Missing Data Restoration using a Human Adaptive Framework

The Cleansing Algorithm

Sjoerd Dijkstra

Delft University of Technology

TU Delft

Missing Data Restoration using an Adaptive Human Framework

by

Sjoerd Dijkstra (4582993)

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 27th of October, 2022 at 10:00.

Student number:	4582993	
Project duration:	November, 2021 – October, 2022	
Thesis committee:	Dr. P. Chen,	TU Delft, supervisor
	Prof. Dr. Ir. G. Jongbloed,	TU Delft
	Prof. Dr. A. Papapantoleon,	TU Delft
	P. van Buuren,	Aegon Nederland, supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This work serves the purpose of completing the Master Applied Mathematics at the Delft University of Technology. In this work is provided my endeavor to construct an algorithm to restore missing data: the Cleansing Algorithm. The research was conducted in cooperation with Aegon Nederland N.V.

I am thankful to Aegon Nederland N.V. and the TU Delft for the opportunity to research this topic. More specifically, I would like to personally thank Philip van Buuren and Bastiaan Scheutjens from Aegon Nederland for the opportunity they have granted me to conduct this research where other institutions were not as willing to put in the effort to making it possible.

I would like to thank my supervisors dr. Piao Chen from the TU Delft and Philip van Buuren from Aegon Nederland for their support. For if it was not for their patience and dedication to supporting my endeavors, this work would not have been possible.

I would like to thank my roommates: David van den Ouden, Riche Mol and David Maffoes, my fellow students of Kern: Rutger, Oskar, Chiel, Mats and Jelle, and my friends: Davy Kreuk and Koen Mulder; as they have not only supported me through the duration of this work, but have all but been a second family away from home during my entire study at the TU Delft.

Most importantly, I would like to thank my family: my parents Harm and Désirée and my two sisters Vivianne and Marilène. They have provided me with endless support for all of my life. Without this close-knit bunch of Dijkstra's, I would have never become who I am today.

For myself, this thesis marks the end of my time studying and I welcome the next step with open arms.

Sjoerd Dijkstra
Delft, October 2022

Abstract

Improving data quality is of the utmost importance for any data-driven company, as data quality is unmistakably tied to business analytics and processes. One method to improve upon data quality is to restore missing and wrong data entries. The goal of this research is construct an algorithm such that it is possible to restore missing and wrong data entries, while making use of a human adaptive framework. This algorithm has been constructed in a modular fashion and consists of three main modules: Data Transformation, Data Structure Analysis and Model Selection. Data Transformation has concerned itself with conversion of raw data to data types and forms the other modules can use. Data Structure Analysis has been designed to deal with correctly missing data and dichotomy in the target feature by making use of three clustering algorithms: DBSCAN, K-Means and Diffusion Maps. DBSCAN is used to determine the necessity of clustering as well as the initialisation of the K-Means algorithm. K-Means and Diffusion Maps have been used as clustering methods in the one-dimensional target feature and the two-dimensional input-target feature pairs, respectively. Data Structure Analysis has further been designed to perform feature selection through three filter methods: CorrCoef, FCBF and Treelet. Model Selection has proposed a novel approach to selection of the best model of a candidate set through the optimisation of a conditional model ranking strategy based on the prior construction of theoretical testing. Our candidate set consisted of Expectation Maximisation, K-Means, Multi-Layer Perceptron, Nearest Neighbor, Random Forest, Linear Regression, Polynomial Regression, ElasticNet Regression. In terms of restorability, it was shown that the optimal configuration of the Cleansing Algorithm for the restoration of missing data, was provided by opting not to use clustering, using a custom alteration to the Treelet algorithm for feature selection and making use of the model selection strategy. This not only lead to the greatest restorability of 56.90% on Aegon data sets, which was an improvement of 44.83% when compared to not using the Cleansing Algorithm, but also to the reduction of computation time by over 400%. A more realistic restorability due to the presence of correctly missing data, was given by the same configuration making use of one-dimensional output clustering. This resulted in a restorability on Aegon data sets of 43.10%. As such it was deemed possible to restore missing data on Aegon data sets. With respect to the human adaptive framework, it was determined that the construction of the algorithm be modular in the sense that any alternate feature selection or clustering approach can be implemented with ease. Furthermore, the model selection module allows us to customize the theoretical testing and choice of regression or classification models for the restoration of missing data. In doing so, the algorithm has laid the foundations for human adaptivity of the Cleansing Algorithm.

Contents

1 Introduction

Data quality is essential in making sure an organization's data is fit for purpose. As such data quality plays an integral part in data governance, but what is data quality? Data quality is a methodology with which the condition of data can be measured. In this way data quality can be used to identify data errors that need to be fixed. Data quality has become an increasingly important point of interest as for many organizations the processing of data has become unmistakably linked to business processes [1]. An example of this linkage is shown in how organizations increasingly depend on data analytics to drive business decision making. If the data quality of data analyzed in this decision making is poor, it is clear that the decision making based on this data may lead to unpredicted results and potential risks. Poor data quality can as such lead to operational problems, inaccurate analytics and bad business strategies. All of these problems can lead to increased expenses, greater risk and potentially even losses for organizations. However, not only internally does bad data quality lead to problems. De Nederlandsche Bank as financial regulator in the Netherlands obligates insurers to adopt a policy-based approach in the performance of core activities. In this policy-based approach the financial regulator asks to provide the definitions of data quality with regards to interpretations of Solvency II data quality requirements in completeness, accuracy and appropriateness [2]. It is in this that the Dutch financial regulator expects insurers to constantly monitor and where necessary improve upon data quality. If data quality policies are considered to be unsatisfactory this can lead to increased monitoring and pressure from the DNB (De Nederlandse Bank) until these policies are again appropriate. In the worst case, insufficient data quality can lead to fines from the financial regulator and this again often leads to reputation damage with long-term consequences for insurers.

Data quality operates in six core dimensions: accuracy, completeness, consistency, timeliness, validity and uniqueness. Accuracy teaches us in what way data reflects the real-world objects and/or events it is intended to model. Completeness teaches us whether all records and values are available. Consistency teaches us whether the same data drawn from two places agree. Timeliness teaches is whether our data is up-to-date. Validity teaches us whether the data fits the requirements defined by business rules and agrees with allowable parameters when these rules apply. Finally, uniqueness teaches us about the existence of duplicate records. Of these six core dimensions of data quality, three dimensions are concerned with wrong and/or missing values within records: accuracy, completeness and validity. In one way or another these three core dimensions verify data through other data that is known to be of high data quality. We shall refer to this process as missing data restoration. In other words, missing data restoration concerns itself with the action of returning wrong or missing records/data to a condition in which this data can be considered correct or non-missing based on some requirements set on the missing data.

These questions will be answered based on arbitrary sets of data as provided by Aegon. There is no prior knowledge to underlying relations within the data and the algorithm should be able to handle any set of data indiscriminately. This raises the question whether it is possible to restore missing data using a human adaptive framework. We interpret this question to consist of two main parts. On the one hand, we ask ourselves if it is possible to restore missing data. On the other hand, we ask ourselves if we can handle human input based on the missing data restoration to adapt and update the algorithm.

In order to handle the first part of this question, we need to define what we understand under being able to restore missing data. We shall consider two different types of missing data: numeric missing data or categorical missing data. Categorical variables concern variables that can take on one of a limited and often fixed number of positive values. Numeric variables concern variables that can take on any real value, possibly on some bounded or semi-bounded interval. For categorical variables, we shall consider a missing value restored when a model returns a test score with 95% accuracy, that is of 100 missing variables we expect our model to be able to correctly restore 95 of these missing variables. For numeric variables, we shall consider a missing value restored when a model returns a test score with 95% accuracy based on a 95% mean absolute percentage error, that is if a value our model returns is within the 5% upper and lower bounds of the true value then we consider this value restored. That is, of 100 missing variables we expect our model to be within the $95\% - 105\%$ range with respect to the true value for 95 of these missing values.

In order to handle the second part of the question, concerning the handling of human input. We consider our algorithm of having a human adaptive framework, if, based on output generated by the algorithm for the restoration of missing data, an expert can provide feedback on aspects of the restoration, such that this feedback, whether positive or negative, can be processed by the algorithm. This feedback processing should result in the algorithm being updated using this expert judgement.

1.1 Literature Review

The restoration of missing data, or more commonly referred to as missing data imputation is a well-studied field. For more classical methods, one looks at Arithmetic Mean Imputation, median imputation or other methods that aim to impute missing data using some singular data statistic. These methods are of course not viable for the restoration of Aegon data.
A great deal of other missing data imputation research concerns itself with the imputation of missing values in the setting of time series, such as signal processing [35] and monitoring [34]. Another set of research focuses on the missing data imputation through probabilistic methods with the goal of as accurately as possible sampling from the underlying distribution such that this data can be used for further analysis with respect to risk and compliance [5]. Finally, research has been done to restore missing data as accurately as possible through methodologies of classification [10] and regression [9]. It is this branch of research that best matches the objectives set for the Aegon data sets.
Several comparative studies have been performed to judge the performances of machine learning methodologies for regression problems. In the work of C. Wang [9], a new approach to missing data imputation is proposed and compared to proven existing methods such as Random Forests, Multi-Layer Perceptrons, Expectation Maximization and Nearest Neighbors. From their results it was shown that though there were merits to the use of their model in terms of time complexity, it was made clear that many of the regression models considered would excel under specific circumstances and there was no clear best model for the task of regression.
In the work of B. Heung [10], a comparison of several classification methods was done for the purpose of digital soil mapping. In their results it is shown that models for classification such as Random Forests, Neural Networks and Nearest Neighbors perform exceptionally well in the classification task.
Based on these works, we are able to consider a subset of the plethora of machine learning algorithms that exist for regression and classification tasks. There is no clear winner under all conditions and as such the determination of the best model for regression and classification are deemed data specific.
This leaves the problem of model selection open. Though, there do exist methodologies for model selection after the models have been trained and tested, such as the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method [32], these methods do not present frameworks to reduce the number of models that need to be constructed.

In light of the presence of correctly missing data within the Aegon dataset and the possibility of other piece-wise continuous relationships for regression, O. Limwattanapibool [23] proposes the use of the Density-based spatial clustering of applications with noise (DBSCAN) algorithm to determine the number of clusters needed for further k-means clustering of data. This framework simultaneously is able to determine the optimal number of clusters to be used for the k-means clustering, but is also capable of determining whether clustering is deemed necessary at all based on analysis of region based clusters.
In the work of E. S. Roozemond [25], a different type of clustering is proposed, making use of diffusion map representations of data. As shown in the work combining the diffusion map process with a clustering algorithm such as k-means clustering, one is able to construct non-linear bounds for clusters. In this way, the clustering representation in a target feature can be determined based on higher dimensional non-linear structures.

The importance of feature selection is illustrated in the work of J. Li, et al. [28]. Feature selection plays a crucial role for building simpler and more comprehensible models, improving data-mining performance, and preparing clean, understandable data. In their work, they also provide an overview of a plethora of feature selection methodologies. In their overview many of these methodologies are concerned with the ranking of feature importance and those that provide subsets of features, such as the Fast Correlation-Based Filter (FCBF). The works of Q. Song, et al. [30] proposes the use and tests the

performance of the Fast Correlation-Based Filter, where B. Senliol, et al. [29] propose an alternative strategy for the selection process that allows not only the feature selection using FCBF, but also aims to remove redundancy from the features selected during the FCBF selection process.
Further work on feature selection with the emphasis on the removal of redundant features occurs in the work of Hui-Huang Hsu and Cheng-Wei Hsieh [31]. In their work they propose the use of correlation coefficients to cluster features together and only choose select features from such high-correlated clusters. Similar to the work of Hui-Huang Hsu and Cheng-Wei Hsieh, A. Lee et al. [27] propose an algorithm based on wavelets called Treelet. Their algorithm is similar to the correlation coefficient feature selection process in that it aims to generate a clustered representation of all features within high-dimensional data. The Treelet algorithm extends on this premise by generating a complete hierarchical tree such that at any level of correlation, a clustered representation of the features can be retrieved.

1.2 Structure of the Work

In this thesis, we shall start by providing an analysis of the Aegon data sets in order to convey the necessity of steps taking in the construction of the algorithm used to restore missing data on Aegon data sets using a human adaptive framework, aptly named the Cleansing Algorithm.

Second, we shall start by providing a schematic overview of the Cleansing Algorithm that has been used to restore missing data and explain globally the ideas behind the different sections within the Cleansing Algorithm process.

Following this, we shall handle each node of the Cleansing Algorithm following the schematic representation. For each section, we shall discuss the reasoning for the node and the theory behind the processes within the nodes, as well as the steps taken within the nodes. We round off each section by explaining how the theory is used within the algorithm and providing an example of the nodes inner workings on a data set from Aegon.

The results from the use of the Cleansing Algorithm and the strategies within the algorithm are provided in the Results section. After which, we end this thesis with an answer to the research question and relevant conclusions and a discussion of the Cleansing Algorithm and recommended future research.

1.3 Overall Procedures for Aegon Data

Before we start with the analysis of Aegon data sets it is important to note the framework that was in place prior with regards to Aegon data procedures. A schematic overview of this is provided in the following figure:

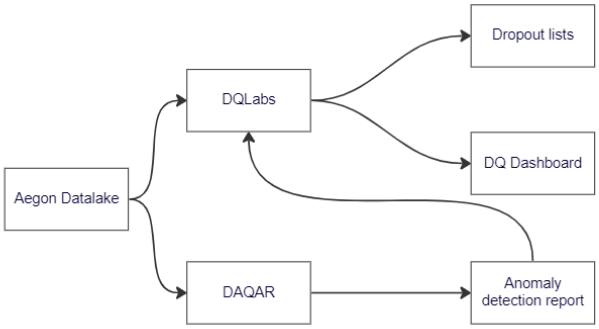


Figure 1.1: Prior overall procedures for Aegon data.

In figure 1.1, we can see that connected to the Aegon Datalake are the DQLabs and DAQAR engines. DAQAR is a tool developed by Aegon that is used to perform anomaly detection on the Aegon data sets. DQLabs on the other hand is used to construct decision rules that are used to determine if the Aegon

data sets adhere to rules and regulations in accordance with Data Quality standards. DQLabs is able to use the anomaly detection output to update its Data Quality checks, which are the decision rules that determine the quality of Aegon data.

It is in this existing framework of procedures that the Cleansing Algorithm as a derivative of this thesis is to be implemented. This has resulted in the following overview:

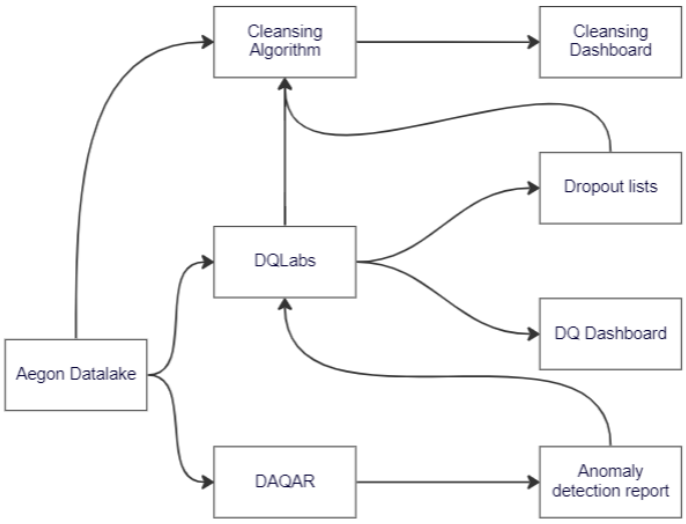


Figure 1.2: Current overall procedures for Aegon data.

In figure 1.2, it is clear that the Cleansing Algorithm has been integrated in the existing Aegon framework, where it takes data directly from the Datalake as well as outputs from the DQLabs engine to further enhance Data Quality.

2 The Cleansing Algorithm

2.1 AEGON Dataset

Before we can dive into modelling, we first need to analyze the data we are working with.
For the imputation of missing data it is of the utmost importance to first determine what missing data we are working with and whether we can analyze its origin.

2.1.1 Data Quality Checks

Aegon is mandated by the watchdog De Nederlandse Bank (DNB) to monitor the quality of their data and where possible make improvements. Apart from this mandate, data quality directly also impacts over business processes, one can think of anything and everything from risk modelling to customer contact.
Therefore Aegon has created the DQLabs project to monitor data quality. DQLabs introduces data quality checks in the form of rules that are constructed by business experts and regulation. If data passes the check it is rewarded with a score of 1 and if not its score is set to 0. That is, if a data point has a score of 0 for some data quality check, then either the value is wrong according to this rule or missing. In either case the value with respect to the data quality check is wrong and needs to be restored. An example of a data quality check is provided:

$$\text{If class} = A \text{ and value} \neq 0, \text{ then DQ} = 0$$

$$\text{If class} \neq A \text{ and value} \leq 0, \text{ then DQ} = 0$$

$$\text{Else, then DQ} = 1$$

This data quality check describes a rule that any instance in the dataset with a class A should have a value assigned to it of 0. If an instance has any other class than class A the value should be greater than 0 to pass. If either of these is not satisfied for some instance of the dataset it fails the data quality check and is assigned a value of 0 for this data quality check. Further analyzing this data quality check we can observe something more, namely that based on the information in this data quality check we know for certain that we are working with categorical and numeric data types. The *value* in the data quality check clearly describes some numeric feature, whereas the *class* clearly describes some categorical feature.
We provide another example of a data quality check:

$$\text{If date} < 2013:01:01 \text{ and response} \neq N, \text{ then DQ} = 0$$

$$\text{If date} >= 2013:01:01 \text{ and response} \notin \{J, N\}, \text{ then DQ} = 0$$

$$\text{Else, then DQ} = 1$$

This data quality check indicates that the *response* feature should always return N if the date feature indicates a date prior to the first of January 2013. If the *date* feature returns a date after the first of January 2013 the response can take on either a J or N. We once more notice the existence of categorical features, in this case the *response* feature. We also note the existence of features that consist of dates.

From analysis of a subset of 100 datasets linked to 100 data quality checks, we therefore have recognized the existence of these three main data types: numeric, categorical and date features. From analysis of 100 data quality checks we have also noted, as with the prior mentioned two data quality checks, that all data quality checks are static. By static we mean that no data quality checks evolve over time and all requirements within the data quality checks are constant. This analysis allows us to handle the features of the type date in a special way. It is therefore possible to convert date features to numeric features, namely daily distances to some arbitrary early date prior to all observations in the feature.

Finally, we provide a crucial example of a data quality check:

$$\text{If status} = n/a \text{ and date} \neq n/a, \text{ then DQ} = 0$$

$$\text{If status} = G \text{ and date} = n/a, \text{ then DQ} = 0$$

$$\text{Else, then DQ} = 1$$

From this data quality check we observe the existence of missing values within the data sets. However, as can be seen from the data quality check if *status* and *date* features are missing then the data quality check passes with score 1, indicating that the Aegon data set contains missing values that are actually correct. It is for this reason that considering missing instances within the data set for restoration is not viable. This makes it clear to us that apart from some data transformation to handle these correct missing values, we are dependent of the scores returned by data quality checks with respect to the restoration of missing data. That is, whenever a data quality check returns a score of zero, we will consider the data to be incorrectly missing, even if an instance exists and is wrong according to the check.

2.1.2 Feature Analysis

This now allows us to further analyze the data sets in terms of missing data type, number of feature/-variables, number of categorical and numeric features and the number of missing and non-missing data points. We provide these general statistics for a subset of the data quality checks:

Missing variable type	#variables	#cat variables post preprocessing	#cat variables pre preprocessing	#num variables	#non-missing datapoints	#missing datapoints
'num'	246	194	59	58	50174	5
'cat'	246	191	58	59	50160	19
'num'	224	271	55	30	303876	0
'cat'	321	356	98	76	37343	0
'cat'	131	123	23	55	110533	0
'cat'	131	123	23	55	110446	87
'cat'	131	123	23	55	110446	87
'num'	131	126	24	54	110533	0
'num'	131	126	24	54	110533	0
'num'	131	126	24	54	110533	0
'num'	131	126	24	54	110533	0
'num'	224	271	55	30	303876	0
'num'	224	271	55	30	303844	32
'num'	226	269	55	32	299616	4260
'cat'	224	284	57	31	358223	0
'cat'	224	284	57	31	358223	0
'cat'	224	299	58	31	358220	3
'cat'	224	300	58	31	358223	0
'num'	258	246	61	76	50619	50
'num'	258	246	61	76	50669	0
'num'	246	194	59	58	49936	52
'num'	382	353	86	87	31588	153
'num'	246	194	59	58	49944	44
'cat'	246	194	59	59	49984	4
'cat'	246	194	59	59	49986	2
'cat'	246	193	59	58	49934	54
'num'	20	32	4	11	341159	0
'num'	321	356	98	75	37202	0
'num'	131	126	24	54	110102	22
'num'	321	354	98	75	37057	145
'num'	321	356	98	75	37199	3
'cat'	233	201	62	54	39721	82
'num'	246	191	59	58	49958	30
'cat'	225	184	31	21	812743	0
'num'	123	129	36	26	50669	0

Figure 2.1: General statistics of a subset of data quality rules in the Aegon dataset.

Based on these general statistics we can make some key observations.
The first thing we notice is that the number of variables/features is relatively high. In all but one instance within this subset of data quality checks, the data has a dimensionality of over 100 and in many cases over 200. Though possible, it is highly unlikely that every variable is informative and non-redundant with respect to the feature with missing data, but we cannot just blindly assume this to be the case. We shall take a closer look at the data set represented by the first row of our general statistics table 2.1:

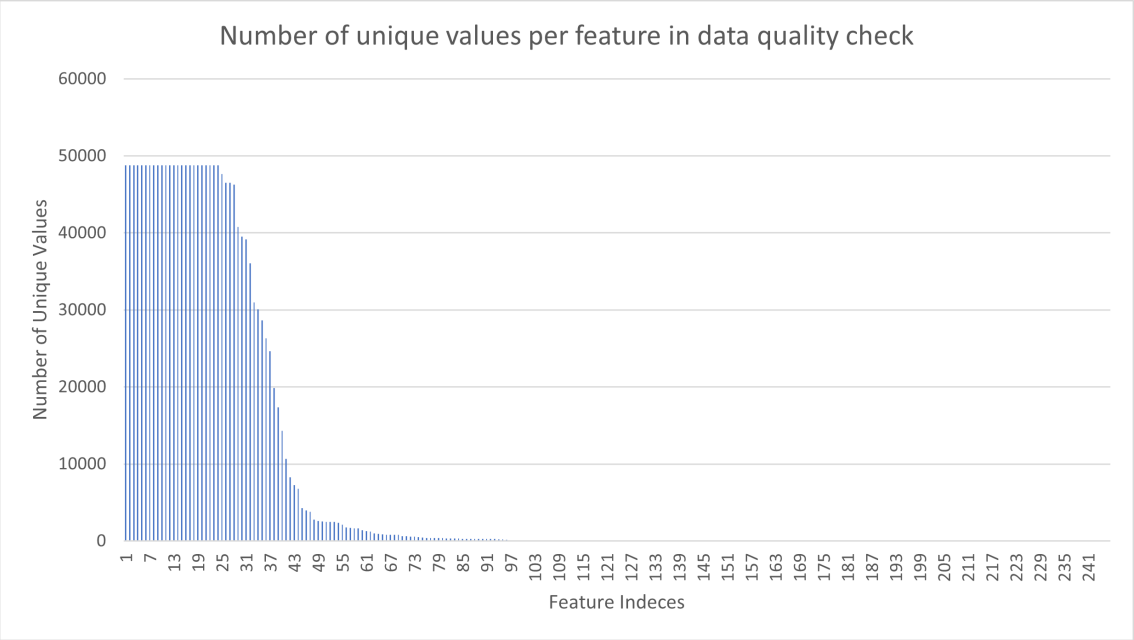


Figure 2.2: Number of unique values in all features of data quality check.

This figure consists of the number of unique values within each feature, sorted by the number of unique instances within features. From the figure we can see that there is quite a few features with a unique element for every data point within the feature. The most logical reasoning for these features is that they are of numeric type. From closer inspection of some subset of these features with unique counts equal to the number of data points, we notice that there exist more than one so-called identifier features. These are features that are either numeric or categorical and unique for each data point and apart from their monotonic, constant increase over time, they may not be informative for the restoration of our target feature. Furthermore, all of these identifiers, in the case they are numeric, will share a very high linear correlation and thus considering all these features may lead to redundancies.
We also notice that after the hundredth feature the number of unique values becomes very low. To better illustrate the differences at these lower unique counts, we provide a log-scaled version of figure 2.2:

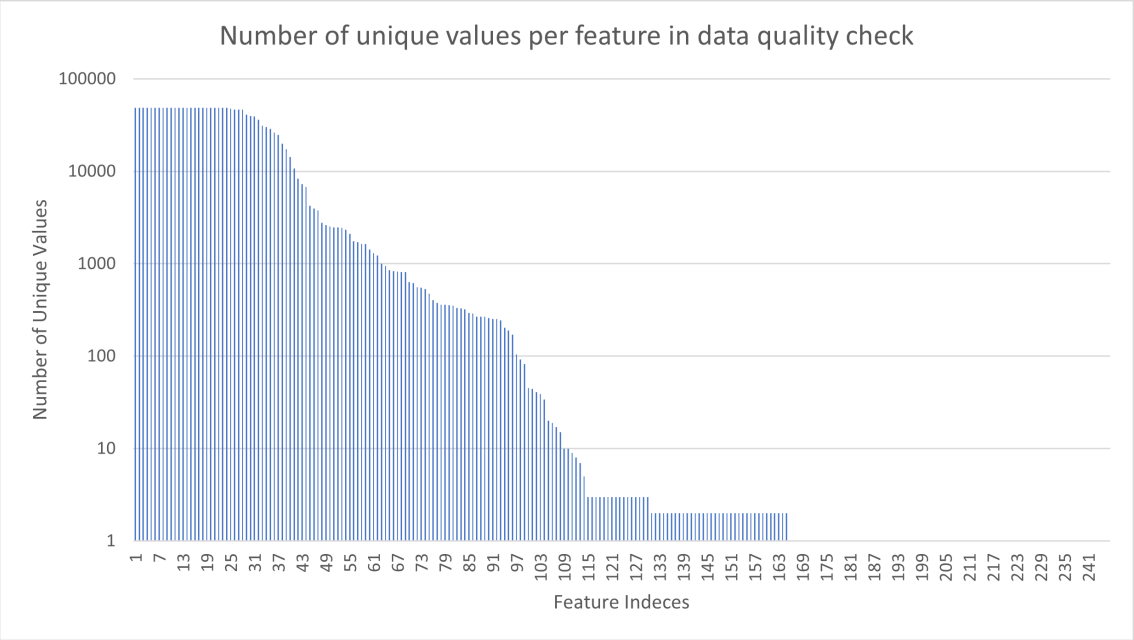


Figure 2.3: Number of unique values in all features of data quality check on logscale.

From this figure we can see that the number of unique instances becomes 1 after feature number 166. Features that only contain one value for every data instance within a feature, of course, are non-informative and thus redundant. This does, however, still leave us with a dimensionality of 166 for this dataset. It is highly unlikely all of these remaining features are informative. This is the reason that dimensionality reduction and/or feature selection will play a crucial role in the restoration of missing data within the Aegon data sets. These techniques will allow us to significantly reduce the dimensionality of our data by only considering informative and non-redundant features for the restoration of missing data within the target feature.

The second observation we make with respect to the figure containing general statistics, is that the number of categorical and numeric features is given. Within the Aegon data sets, features are considered to be categorical if they consist of less than or equal to 20 unique values and if features are number instances other than that, they are considered numeric variables. Features with only one unique instance are not considered in these counts and neither are variables that contain more than the 20 pre-defined unique instances of non-number features (in example strings). In terms of relative frequency of the numeric versus categorical feature counts, there is no clear rule or winner, they can vary indiscriminately, though they do seem to be relatively close to one another before pre-processing. The column containing categorical variable counts post pre-processing are an indication of the increase in dimensionality when using the one-hot encoding technique for categorical features. This allows each class within the feature to represent its own information independently of the other classes. This does, however, assume that all categorical instances are nominal. Furthermore, when analyzing the column containing the type of the feature containing missing data points, we notice that categorical and numeric features both occur with similar rates from this sample. This teaches us that any missing data restoration approach should consider separate methodologies for the numeric and categorical cases.

Type	Unique Obs.	Average (StandardDev)	Median	Missing Obs.	Min./25%q/75%q/Max.
Numeric	7491	60537.55 (85524.51)	35000.0	53	-1.0; 17500.0; 65000.0; 1362528.0
Numeric	26433	32158.67 (98579.55)	0.0	265	-855650.0; 0.0; 39153.0; 1538051.0
Numeric	18214	214586.04 (156156.87)	197250.0	72	4124.0; 74716.0; 310000.0; 966720.0
Numeric	2265	156256.93 (95099.22)	138000.0	72	4970.0; 88740.0; 203000.0; 655000.0
Numeric	42355	232700.53 (179837.83)	201675.0	125	-1.0; 78800.0; 345508.0; 1550000.0
Numeric	49375	1347956.14 (15651.06)	1347974.5	0	1320875.0; 1334381.25; 1361494.75; 1375083.0
Numeric	353	0.14 (0.17)	0.15	89	-5.4; 0.15; 0.15; 1.0
Numeric	276153	71233615.96 (21888446.2)	78652377.0	0	10470.0; 58815019.75; 88178322.0; 95264677.0
Numeric	220149	229893.3 (131473.82)	215889.5	4292	3202.69; 147000.0; 295000.0; 4038643.92
Numeric	80937	232520.62 (141281.44)	212000.0	4260	3202.69; 143352.06; 293922.5; 4038643.92
Numeric	52	-0.22 (0.16)	-0.15	0	-0.8; -0.32; -0.1; -0.01
Date	357	2021-12-31 (157643.9)	2653-08-22	68	2021-06-28; 2021-10-13; 2022-03-17; 2022-06-24
Numeric	1662	321.73 (288.61)	232.02	180	9.72; 111.83; 447.26; 2066.76
Numeric	75633	409.25 (395.42)	277.08	180	1.94; 124.54; 570.69; 4123.36
Numeric	1194	2.21 (0.7)	1.95	0	0.1; 1.72; 2.48; 7.9
Numeric	16600	71354.45 (73128.61)	47549.5	25	0.32; 20898.5; 100000.0; 950000.0

Figure 2.4: Descriptive statistics of a subset of data quality rules in the Aegon dataset.

2.1.2.1 Correctly Missing Data

As can be seen from the first line of descriptive statistics in figure 2.4, we encounter a numeric feature that always contains some number significantly greater than zero, unless the value is negative one. These occurrences within the Aegon data set are data points with missing values that are supposed to be missing, duly named correctly missing data. In order to handle these missing occurrences, it occurs that these missing values are filled with zeros or negative ones so descriptive statistics such as these can be gathered more easily. We provide an example of a histogram containing data adhering to this phenomenon.

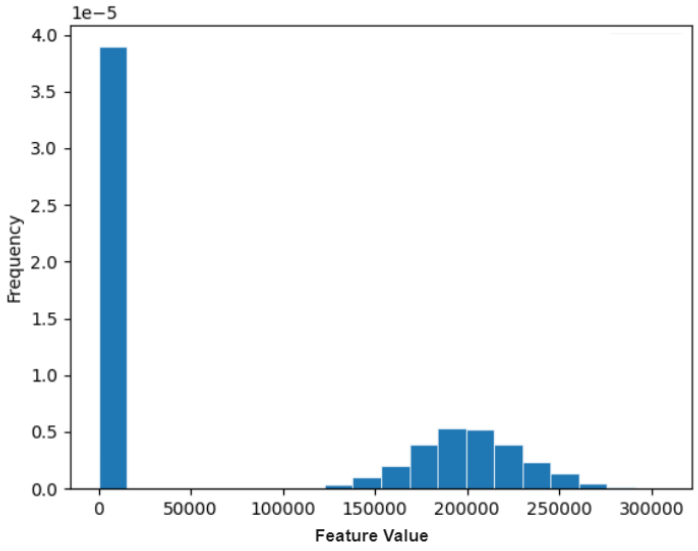


Figure 2.5: Example of feature containing correctly missing data.

From this histogram it is clear that the majority of cases within this feature are correctly missing or set to be some constant value for all cases. These types of phenomenon can negatively influence the processes of normalization or maybe even feature selection. It can even directly influence the performance of models used for restoration as imputing values of zero for every case can result in high test scores. To tackle this, one would need to find a methodology that can separate the data points that are only zero-valued from those with non-zero values.

2.1.3 Missing Data

Analyzing once more figure 2.1 containing the general statistics , we concern ourselves with the missing and non-missing data.

In general there exist three types of missing data: MCAR, MAR and MNAR [46]. Before considering methods to restore missing data it is important to know in which setting the Aegon data sets exist. Missing Completely At Random (MCAR) is the case when the probability of an observation being missing does not depend on observed or unobserved measurements. Missing At Random is the case when the probability of an observation being missing depends only on observed measurements. Finally, Missing Not At Random occurs when the probability of an observation being missing depends on unobserved measurements. In this scenario, the value of the unobserved responses depends on information not available for the analysis [33].
Determination of a general missing data setting may not be advisable in some circumstances. However, looking at the number of missing data points with respect to the number of non-missing data points it is difficult to assume anything other than Missing (Completely) At Random as on such small samples any test to determine MNAR will not yield strong results.
Looking back at the examples of data quality checks that lead to these numbers and analysis of 100 other data quality checks teaches us that each data quality check does consider all cases within a feature when providing a data quality score. As such the data quality checks do not influences the patterns missing data occurrences in this manner.
A fact that can influence the type of missing data, is that personnel at Aegon have manually started restoring missing data points. It can be the case for some features that the "easier" parts of data quality checks are restored manually, while the more sophisticated and broader restoration spaces are left alone. This may lead to some missing data instances to be Missing At Random, though there is no direct way to prove this as, as stated before, the number of missing data points are too small to assume anything other than Missing At Completely Random.

Furthermore, we add that even though quite some data quality checks do not lead to missing data points, the possibility does exist that this may occur in the future and with the static nature of our data sets and demand from Aegon, restoration models should also be constructed for these features.

2.1.4 Restorability

Since data Aegon handles, concerns but is not limited to customer data, the restoration of missing data needs to be as accurate as possible. That is, it does not suffice to restore some missing value in a rough ball-park as this may mean if such a value is approved for restoration, pricing and other considerations based on these value may impact customers and/or Aegon strategies and analyzes directly. It is for this reason it was decided that the minimum requirement for restored missing data to be considered restorable, if it has at least a 95% accuracy with respect to classification and is within the 95% to 105% interval with respect to the true value in 95% of cases for regression.

2.2 Structure of the Cleansing Algorithm

In this section we shall provide a schematic overview of the algorithm used to restore missing data using a human adaptive framework. In figure 2.6 a schematic overview of this "Cleansing Algorithm" is provided. First, the algorithm is provided with data with missing values through the thick arrow on the left. In the first node Data Transformation this data is then transformed in such a manner that the data is usable in the Data Structure Analysis, Model Selection, Model Training and Model Testing nodes. The second node is that of Data Structure Analysis. The aim of this node is to capture lower dimensional structures within the transformed data as to allow us to increase model complexity and lower model strain in Model Selection node by means of feature selection and clustering. This node uses a framework to calibrate its structure analysis based on data quality checks. The third node is Model Selection. In this node we aim to discover the underlying data relation between the input and output features by comparing restoration performance of various candidate models based on our proposed model selection strategy. The fourth node is Model Training. In this node our goal is to train our selected model on a training data set such that we achieve the highest accuracy or lowest error possible. The fifth node is Model Testing. In this node we verify the now trained model on a test data set. The sixth node is Results. This node aims to provide the output based on all previous nodes. From Data Structure Analysis it is provided with data structures, from Model Selection it is provided with the model used and from the Model Training and Model Testing phase the Results node restores missing data. This node also provides

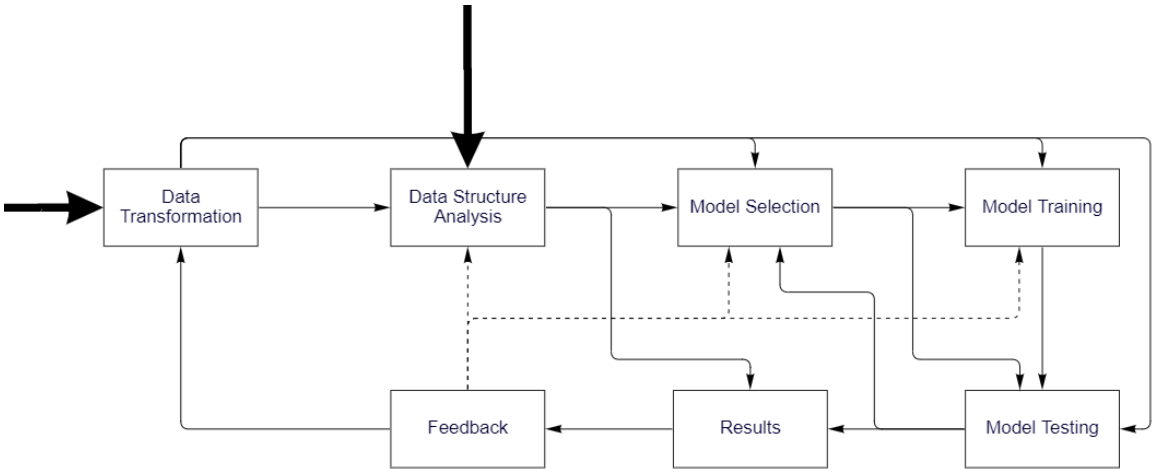


Figure 2.6: Schematic Representation of the Cleansing Algorithm

the necessary restoration rates on the restored missing data based on the previous two nodes. The sixth node is Feedback. This node concerns itself with the human input based on the Results node. If feedback is positive, then data structures, the model and results are stored. In this scenario restored data can be fed through and used in the algorithm. If feedback is negative, a reason can be given based on features used, model used or restored data values. Based on this feedback the Cleansing Algorithm is able to adapt itself and try again.

3 Data Transformation Node

In this section we discuss the Data Transformation node locally. The Data Transformation node concerns two processes: the transformation of data into forms in which they can be used in the procedures of following nodes and the scraping of features from the data quality checks associated to the data.

3.1 Transformation of Data

The transformation of Aegon data focuses on converting raw Aegon data into data types and forms the following nodes in the Cleansing Algorithm will be able to use. In light of this, Data Transformation starts by converting all features containing non-numeric values into numeric features that the other nodes are able to use.

The first insight we will consider based on analysis of Aegon data, is presence of features containing string values. As an example, one may think of a feature named *Gender* in which the terms $Male$ and $Female$ will occur as strings. These are not numeric features and as such will be transformed. First, variables containing strings are converted to numbers. The first string in a variable is converted to value 1 and each re-occurrence of this string acquires this same value. The next unique string acquires the value 2, then this process is repeated until all strings are converted. An example of this is given in Table 1.

Gender	Number
Male	1
Female	2
Male	1
Female	2
Female	2
Female	2
Male	1
Male	1

Table 1: Example conversion strings to numbers.

The second insight we will consider is the presence of features containing dates. Dates can occur in many forms within the Aegon data sets. They may occur in "Datetime" form or as strings with various different orderings for days, months and years, and even a plethora of symbols used to divide these three elements of the date. As such, the transformation of date features starts through recognition of whether or not a feature contains dates. Once a date feature is successfully found, the transformation node determines which symbols are used to split the day, month and year parts. If this is completed, the node will determine which part of the string represents each sub-part. The year sub-part is the only part that is able to take on values greater than 31 and is the only part that can consist of 4 digits. Once the year part is isolated, the other two parts can be distinguished by analysing the domain of each part. The feature that never exceeds the value 12 will be the month part and the remainder is deemed the day part of the date. A variable containing dates is then transformed into the number of days from the date occurrence that is the farthest in the past. An example of this is given in Table 2, where the 23rd of October 1998 shall be considered the date that lies farthest in the past.

Date	ΔDays
23-10-1998	0
14-01-2022	8484
27-02-2020	7797

Table 2: Example conversion dates to numbers.

When all values are numbers of some form, we sort variables based on whether we consider them to be of numeric or categorical nature. A variable is considered to be categorical if the cardinality of that variable is much lower than the length of that variable. In other words, we consider a variable to be categorical if we observe a small amount of unique values relative to the total amount of values. This threshold is set to be at 20 unique values or less in accordance with Aegon definitions for a categorical feature. In the case of string values, we also consider those to be categorical regardless of its cardinality, though they are filtered out at the start of the Data Structure Analysis node. A variable is considered to be numeric if it is not categorical. One can consider this to be all variables with more than 20 unique values and only containing numeric values.
Finally, variables with numeric domains are normalized to the [0,1] finite interval as to perform regression on common scales. This is done by taking each value of a variable and subtracting the variables minimum value and dividing by the difference between the variables maximum and minimum values. An example of this is given in Table 3.

Income Value	Normalized Income Value
55000	1.0000
25000	0.2603
32500	0.4452
14444	0.0000

Table 3: Example Normalized numeric variable.

Variables with categorical domains are one-hot encoded to support regression and classification [3]. However, when performing classification, we refrain from turning the target variable, with missing data to restore, to one-hot encoded variables. This transformation consists of converting each unique class to a unique variable with value 1 if it is of that class and if not it acquires value 0. A categorical variable consisting of five unique classes is converted to five unique variables consisting of two states: 0 and 1. An example of this is given in Table 4.

Category Number	Category 1	Category 2	Category 3
1	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
3	0	0	1
2	0	1	0

Table 4: Example one-hot encoding of categorical variable.

3.2 Scraping Data Quality Checks

As shown in section 2.1.1, data quality checks are used to determine whether or not a data point is to be considered missing or has a wrong value. As such it is expected from features within data quality checks to contain information regarding the output/target feature that contains the missing value. For further use in the Data Structure Analysis node, we thus take the SQL-string as an input containing the data quality check and scrape for all occurrence of features within the check. As such if one of the features within the data quality check is deemed to contain missing data, it is then assigned as the output/target feature and all other features that have been scraped from the data quality check shall be assigned as input features. We provide a censored example of the SQL-string associated to the first data quality check provided in section 2.1.1:

```
SELECT x, x AS x, class, value,
CASE WHEN class = 'A' THEN CASE WHEN value IS
NULL THEN 0 ELSE 1 END ELSE 1 END AS dq_check FROM
x INNER JOIN x ON x = x
INNER JOIN x ON x."element" = x INNER JOIN
x ls ON x = x INNER JOIN x ON
x = x."element" AND (x > CAST(DATEADD(YEAR,
-1, GETDATE()) as DATE)) AND x = 'A'
```

Figure 3.1: SQL-string associated to first Data Quality Check in section 2.1.1.

Within the SQL-string of this data quality check, we have emphasized the *class* and *value* features as they occur in the string. It is these features that we wish to scrape from the string. To do so, we start by reducing the string such that it just contains the relevant data quality check. This is done by recognizing that the actual relevant data quality check starts from the first occurrence of "CASE WHEN" and ends as soon as we recognize the *dq_check* feature. The shortened SQL-string is provided below:

```
CASE WHEN class = 'A' THEN CASE WHEN value IS
NULL THEN 0 ELSE 1 END ELSE 1 END AS dq_check
```

Figure 3.2: Shortened SQL-string associated to first Data Quality Check in section 2.1.1.

Once the SQL-string associated with the data quality check is reduced to this form, we need to recognize the features within the check. For this purpose, we have setup a small database containing all possible occurrences of mathematical operators and SQL functionalities that may contain a feature somewhere in their vicinity. For example, an equals sign "=" can contain a feature to its left and right, thus this is where we search for features. From there we remove any string that was captured that cannot be considered a feature. An example of such a string may occur if we expect a feature before a functionality, but that feature is nested within other SQL functionalities.

Once complete, this then provides the Data Structure Analysis node with all feature occurrences within the data quality check corresponding to the now transformed data set. After performing the transformation of data, we also use the labelling of data points that is provided by the data quality check to split the data in a data set that contains only non-missing data and one where the output feature is deemed to have missing or wrong data in accordance with the data quality check. These sets are passed on to the Data Structure Analysis node.

3.3 Transforming Aegon Data

In this subsection, we shall handle an example of using the transformation process, as described in 3.1, on the actual Aegon data.
First, we provide the data quality check for the simulated Aegon data set:

$$\text{If Residual} \geq \text{Total} - \text{Used, then DQ} = 1$$

$$\text{Else, then DQ} = 0$$

This data quality check teaches us that the *Residual* feature should at all times be greater than or equal to the difference between the $Total$ and $Used$ features. In the context of a loan, for example, we expect that money left in the *Residual* feature to be equal to the total money provided in the loan in the $Total$ feature minus the money of the total loan used in the $Used$ feature. If this residual is less than the indicated difference, money will have gone missing. This should not be a possibility and as such we expect the inverse to be the ground truth. On the other hand, if the residual is strictly greater

than the difference between the total loan and the money of this loan that is in use, money will have been generated. At least this would be the case if the three mentioned features in the data quality check captured the entire relationship. In order to gain a better understanding of this idea, we shall first provide a subset of features on a sample of size 22:

Used	Buffer	Contract Age	Total	Active	Identifier	Date	Residual	DQ
30554.18		1.0	68479.73	N	7689d745-...	2021-04-19	37925.55	1
28219.37	9479.79	4.0	52796.67	Y	42cbf7f6-...	2018-10-25	24577.29	1
16560.57		3.0	75305.98	N	37686f16-...	2019-08-08	58745.41	1
2512.19		10.0	60898.42	N	3ca1777f-...	2012-04-20	58386.23	1
24552.02		18.0	63534.9	N	ecc6b450-...	2004-05-07	38982.88	1
23255.29		4.0	57862.87	N	4db67f01-...	2018-10-04	34607.58	1
14685.44		7.0	64325.56	N	5ee19b4e-...	2015-11-10	49640.12	1
12873.25		9.0	84255.29	N	a85125dc-...	2013-01-09	71382.04	1
3355.06	249.19	2.0	71246.45	Y	a1840a28-...	2020-10-02	67891.39	1
26425.97		10.0	73671.5	N	ae851acd-...	2012-01-15	47245.53	1
24666.74	2892.95	1.0	78639.19	Y	3e16723c-...	2021-05-21	53972.45	0
30717.65		2.0	64913.79	N	829b9939-...	2020-03-26	34196.14	1
3426.7		7.0	59471.58	N	edf41bc8-...	2015-02-06	56044.88	1
29820.3		15.0	68985.32	N	2ab22851-...	2007-11-25	39165.02	1
16931.93		22.0	61332.03	N	966626a8-...	2000-05-11	44400.1	1
22886.77		1.0	80668.11	N	b110a80d-...	2021-08-07	57781.35	1
6047.94		2.0	55967.46	N	0afa5b1a-...	2020-08-18	50940.35	1
28561.3	1051.36	3.0	67352.89	Y	e305f444-...	2019-11-21	38791.6	1
16808.96	4941.84	5.0	62331.74	Y	fe735f64-...	2017-05-15	45522.78	1
32415.22		6.0	70308.96	N	cab8918d-...	2016-11-19	37893.74	1
4112.61		4.0	59444.65	N	a404d9a5-...	2018-05-09	55332.04	1
29666.81	1498.18	9.0	62422.12	Y	499ccf30-...	2013-12-15	32755.31	0

Table 5: Sample from Aegon data set.

The features selected consist of all those mentioned in the data quality check itself, in that features $Residual$, $Used$, $Total$ and DQ all occur. Here, the desired relationship between the features $Residual$, $Used$ and $Total$ is provided by the data quality check. If this relationship is satisfied, the value of the DQ feature will be positive and return 1 and if not return the value 0. Apart from this, we have provided a feature $Identifier$ that consists of unique identification strings for each contract, a feature $Date$ that describes when the contract started and a feature $ContractAge$ providing us with the age of the contract in years. Finally, some features for the contracts are provided, such as the $Active$ feature that describes whether or not a buffer is active, if so, the feature $Buffer$ stores this value.

Before the data can actually be transformed, we first need to split the data set into a set containing only those data points that return a value in the DQ feature of 1 and a set containing the missing data for occurrences with value 0 in feature DQ. This leaves us with the following sample of 20 data points that contain no missing values:

Used	Buffer	Contract Age	Total	Active	Identifier	Date	Residual	DQ
30554.18		1.0	68479.73	N	7689d745-...	2021-04-19	37925.55	1
28219.37	9479.79	4.0	52796.67	Y	42cbf7f6-...	2018-10-25	24577.29	1
16560.57		3.0	75305.98	N	37686f16-...	2019-08-08	58745.41	1
2512.19		10.0	60898.42	N	3ca1777f-...	2012-04-20	58386.23	1
24552.02		18.0	63534.9	N	ecc6b450-...	2004-05-07	38982.88	1
23255.29		4.0	57862.87	N	4db67f01-...	2018-10-04	34607.58	1
14685.44		7.0	64325.56	N	5ee19b4e-...	2015-11-10	49640.12	1
12873.25		9.0	84255.29	N	a85125dc-...	2013-01-09	71382.04	1
3355.06	2490.19	2.0	68756.26	Y	a1840a28-...	2020-10-02	67891.39	1
26425.97		10.0	73671.5	N	ae851acd-...	2012-01-15	47245.53	1
30717.65		2.0	64913.79	N	829b9939-...	2020-03-26	34196.14	1
3426.7		7.0	59471.58	N	edf41bc8-...	2015-02-06	56044.88	1
29820.3		15.0	68985.32	N	2ab22851-...	2007-11-25	39165.02	1
16931.93		22.0	61332.03	N	966626a8-...	2000-05-11	44400.1	1
22886.77		1.0	80668.11	N	b110a80d-...	2021-08-07	57781.35	1
6047.94		2.0	55967.46	N	0afa5b1a-...	2020-08-18	50940.35	1
28561.3	1051.36	3.0	67352.89	Y	e305f444-...	2019-11-21	38791.6	1
32415.22		6.0	70308.96	N	cab8918d-...	2016-11-19	37893.74	1
4112.61		4.0	59444.65	N	a404d9a5-...	2018-05-09	55332.04	1
16808.96	4941.84	5.0	62331.74	Y	fe735f64-...	2017-05-15	45522.78	1

Table 6: Sample from Aegon data set where data points deemed missing are removed.

Then, we split the data containing no missing values in a training, test and validation set. These sets consist of randomly selected data points such that the training set contains 70% of the non-missing data points, the test and validation sets both have 15% of this data. The training set is then given by the following 14 data points:

Used	Buffer	Contract Age	Total	Active	Identifier	Date	Residual
30554.18		1.0	68479.73	N	7689d745-...	2021-04-19	37925.55
28219.37	9479.79	4.0	52796.67	Y	42cbf7f6-...	2018-10-25	24577.29
16560.57		3.0	75305.98	N	37686f16-...	2019-08-08	58745.41
2512.19		10.0	60898.42	N	3ca1777f-...	2012-04-20	58386.23
24552.02		18.0	63534.9	N	ecc6b450-...	2004-05-07	38982.88
23255.29		4.0	57862.87	N	4db67f01-...	2018-10-04	34607.58
14685.44		7.0	64325.56	N	5ee19b4e-...	2015-11-10	49640.12
12873.25		9.0	84255.29	N	a85125dc-...	2013-01-09	71382.04
3355.06	2490.19	2.0	68756.26	Y	a1840a28-...	2020-10-02	67891.39
26425.97		10.0	73671.5	N	ae851acd-...	2012-01-15	47245.53
30717.65		2.0	64913.79	N	829b9939-...	2020-03-26	34196.14
3426.7		7.0	59471.58	N	edf41bc8-...	2015-02-06	56044.88
29820.3		15.0	68985.32	N	2ab22851-...	2007-11-25	39165.02
16931.93		22.0	61332.03	N	966626a8-...	2000-05-11	44400.1

Table 7: Sample from Aegon data set where data points deemed missing are removed.

Next, based on the features named in the data quality check, we select the feature *Residual* to be our target feature for which we wish to restore wrong values. Before the other features are handled, the target feature is handled separately. Here, the feature *Residual* consists of float values, indicating it is of numeric nature. This is further confirmed in accordance with the Aegon definition of a numeric feature, since its unique occurrence rate is 100%. Using this information we determine that the target feature should be normalized. To better grasp the transformation process, we shall use the maximum and minimum values as provided in table 15. The maximum and minimum values that occur within the feature are then given by 71382.04 and 19053.47 respectively. Normalization is then performed for all

non-missing occurrences within the sample by subtracting the minimum sample value and then dividing by the difference between the maximum and minimum sample occurrences. For the first data point in table 15, this results in:

$$\text{Normalized Data Point} = \frac{37925.55 - 24577.29}{71382.04 - 24577.29} \approx 0.2852$$

We continue this process for all values in the feature, resulting in the following normalized feature:

Residual
0.285190
0.000000
0.730014
0.722340
0.307781
0.214301
0.535476
1.000000
0.925421
0.484315
0.205510
0.672316
0.311672
0.423521

Table 8: Normalized Residual feature.

With respect to the transformation of this data to forms that the following nodes in the Cleansing Algorithm will be able to use, we shall first analyze further the input features provided. The $Used$ and $Total$ features consist of numeric values and more specifically floats. In terms of the occurrence rates of unique values within these features, we recognize these to be 100% for both. Based on this information, the process described in section 3.1, labels these features to be numeric. Furthermore, we recognize that the feature's occurrence rates of correctly missing data are 0%, thus no imputation is necessary. These features shall thus be handled similarly to the target feature, where we store the training minimum and maximum values and normalize all occurrences using these values.

A feature that is handled differently is provided in the $Buffer$ feature. For this feature, we note the existence of missing values. As the DQ feature for this training set deemed this to be correct, we set this feature to contain correctly missing values. Analysis of those values within the feature that are not correctly missing, we note they are all of type float and are all unique. On the entire feature, including values outside of our training set, we note that the occurrence rate of correctly missing values is approximately equal to 90%. Within our training sample, however, it is closer to 85%. The two data points that do not contain a correctly missing value are both greater than 0. As such there is no natural occurrence of zero within the feature and correctly missing data is imputed with the value 0. Once the missing values are imputed, we perform normalization on the feature based on the new imputed sample minimum 0 and maximum 9479.79. This then results in the following transformed feature $Buffer$:

Buffer
0.000000
1.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.262684
0.000000
0.000000
0.000000
0.000000
0.000000

Table 9: Imputed and Normalized Buffer feature.

The next input feature is a special case and is handled differently in this sample setting. In the setting of our training sample, we observe that the feature contains only integer values. Furthermore, these integer values are not all unique, but the unique value occurrence rate is 71.43% or there are 10 unique values within the feature of size 14. This would be considered a categorical feature in this setting, as the number of unique values is less than 20. However, if we are to consider a training set containing 70% of the entire feature, we have observed that this feature can take on any integer value from 1 to 30. This is in accordance with the fact that a contract can last up to 30 years, as such a contract age will fall into the indicated range. In the full data setting this feature thus contains more than 20 unique values and as such it would considered to be numeric. To remain true to the true data set and in order to keep this example clear, we shall consider this feature to be numeric and assume that the maximum and minimum values occur within the considered sample. The maximum is then given by 22 and the minimum by 1. This feature is then normalized using the same process as in table 8.

Next, we consider the *Active* feature. This feature clearly consists of only two unique values, namely Y and N. This tells us that this feature is categorical in accordance with our definitions. Following the process in section 3.1, we first transform the categories into numbers such that a Y value corresponds to the value 1 and a N corresponds to the value 2. This is done in order to save which one-hot encoded feature belongs to which category. After this is completed, we perform one-hot encoding, where values of 1 and values of 2 are converted to two new features. The first of these features contains a value of 1 as the original category of the *Active* feature contained a Y and the second contains the same response value for N. The one-hot encoded features for the original *Active* feature are thus provided by:

Active_1	Active_2
0	1
1	0
0	1
0	1
0	1
0	1
0	1
0	1
1	0
0	1
0	1
0	1
0	1
0	1

Table 10: One-hot encoded Active feature.

Another feature that is considered to be categorical in accordance with the Aegon definition of a categorical feature, is the $Identifier$ feature. This feature contains only string values. However, unlike the *Active* feature, has a unique occurrence rate of 100%. As such one-hot encoding this feature will lead to nothing but many features containing no mutual information. This feature will be discarded.

Finally, the *Date* feature remains. This feature contains values of type "datetime", indicating that this feature contains dates. The unique occurrence rate for this sample of the feature is 100%. On the entire feature space, this rate is not close to 100%, but still much greater than the 20 unique values required by the definition for a categorical feature. As such this feature is of type date and numeric in nature. True to the process described in section 3.1, we first search for the earliest available date. In this case the earliest date occurrence is $2000 - 05 - 11$. This will be our theoretical starting point. We now determine the difference in days with respect to this earliest date. This results in the following feature:

Date
7648
6741
7028
4362
1457
6720
5661
4626
7449
4266
7259
5384
2754
0

Table 11: Date to Number transformation of Date feature.

After this transformation is performed, we are left with a numeric feature with more than 20 unique values and thus can be normalized as well.

Now that each feature is transformed, we provide the sample from table 7 in its transformed form:

Active_1	Active_2	Used	Buffer	Contract Age	Total	Date	Residual
0	1	0.994204	0.000000	0.000000	0.49853	1.000000	0.285190
1	0	0.911426	1.000000	0.142857	0.000000	0.881407	0.000000
0	1	0.498073	0.000000	0.0952381	0.715521	0.918933	0.730014
0	1	0.000000	0.000000	0.428571	0.257537	0.570345	0.722340
0	1	0.781403	0.000000	0.809524	0.341345	0.190507	0.307781
0	1	0.735429	0.000000	0.142857	0.161043	0.878661	0.214301
0	1	0.431592	0.000000	0.285714	0.366478	0.740194	0.535476
0	1	0.367342	0.000000	0.380952	1.000000	0.604864	1.000000
1	0	0.0298832	0.262684	0.047619	0.816047	0.97398	0.925421
0	1	0.847842	0.000000	0.428571	0.663565	0.557793	0.484315
0	1	1.000000	0.000000	0.047619	0.385176	0.949137	0.205510
0	1	0.0324232	0.000000	0.285714	0.212181	0.703975	0.672316
0	1	0.968185	0.000000	0.666667	0.514601	0.360094	0.311672
0	1	0.511239	0.000000	1.000000	0.27132	0.000000	0.423521

Table 12: Transformed sample from Aegon data set.

It is in this form that the features will enter the next node, whereby the *Residual* feature is marked as the target feature. Furthermore, transformation information is also carried over to the results node, where this information is needed to undo the transformations performed on the target feature. It is of note that we have now only transformed the training data set. If we wish to transform the test and validation data sets, we do so using the information of the transformations as acquired by the process on the training set. To illustrate this idea, we first provide the test set corresponding to our sample training set:

Used	Buffer	Contract Age	Total	Active	Identifier	Date	Residual
22886.77		1.0	80668.11	N	b110a80d-...	2021-08-07	57781.35
6047.94		2.0	55967.46	N	0afa5b1a-...	2020-08-18	50940.35
28561.3	1051.36	3.0	67352.89	Y	e305f444-...	2019-11-21	38791.60

Table 13: Test set of sample from Aegon data set.

If we were to process this test set as done with the training set of data, we recognize that this will lead to normalization for numeric features based on only the three values provided in the test set. However, if we do so, the values between 0 and 1 for this normalization do not coincide with those as provided by the training set. This means that if we determine there to be an underlying relationship between input and output features on the training set, then using this alternatively normalized data, we would have an inappropriate result. As such we use the information from transforming training data. We provide an overview of information stored:

Information	Used	Buffer	Contract Age	Total	Active	Date	Residual
Maximum	30717.65	9479.79	22	84255.29		7648	71382.04
Minimum	2512.19	0	1	52796.67		0	24577.29
Categories					$\{Y, N\}$		
Type	Numeric	Numeric	Numeric	Numeric	Categorical	Date	Numeric
Start Date						$2015-02-06$	
Imputation		0					

Table 14: Transformation information stored of sample from Aegon data set.

In table 14 we notice that for numeric features the maximum and minimum training sample values are stored, whereas categorical features are supported by storing the set of categories in order. The type of the feature is provided. For dates the earliest date occurrence from the training sample is provided. In the case imputation has taken place, when a feature contains correctly missing values, then we provide

the value with which the correctly missing values are imputed. Using this information to transform the test set, results in the following table:

Active_1	Active_2	Used	Buffer	Contract Age	Total	Date	Residual
0	1	0.722363	0.000000	0.000000	0.885971	0.358133	0.709416
0	1	0.125357	0.000000	0.047619	0.100792	0.264121	0.563256
1	0	0.923548	0.110905	0.095238	0.462710	0.228687	0.303694

Table 15: Transformed test set of sample from Aegon data set.

As can be seen from the transformed test data, the only occurrences of 0 and 1 in the numeric features are achieved if the minimum and maximum of the training data set are achieved. From this small test set it is shown that every transformed value takes on some value from 0 to 1, however, this need not necessarily be the case. In the case that the maximum value of a numeric feature in the test set exceeds the maximum value of that numeric feature on the training set, then the transformed value can exceed 1. The opposite holds for the minimum value, where a lower minimum in the test set results in a transformed value less than 0.
On the other hand, for new categorical occurrences in the test set, that do not occur in the training set, it is more difficult. If we discover a new category in the test data, then we need another one-hot encoded feature to capture information for this category. If we wish to do so, we need to have available the information of that category prior to discovering its existence in the test set. As such, unlike numeric features, we consider the entire data set for the determination of the number of one-hot encoded features are necessary. If a category then does not occur in the training data set, but does show up in the test set, then during the feature selection process, which we will discuss hereafter, the one-hot encoded feature containing information for this category is removed.

4 Data Structure Analysis Node

In this section we discuss the data structure analysis node as provided in the schematic representation of the Cleansing Algorithm 2.6. The Data Structure Analysis node consists of two main parts: first we shall discuss the feature selection process, after which we discuss clustering within the node.

4.1 Feature Selection

Feature selection is the process of selecting a subset of features such that the subset contains all, if not most, relevant features for use in the construction of a model. There are a number of reasons one may opt to use feature selection. Feature selection plays a crucial role in terms of interpretation of models that make use of the subset of selected features. A model that uses a hundred different features, as opposed to a model that uses five features that yield the same accuracy, may give a less clear consensus with respect to which features within the hundred input features has lead to the given output. If the model with five features captures all or most of the influence the input features have on the output, then this model is easier to interpret and verify if features used are adequate or not. As the feature selection process selects a subset of the original total of input features, models based on the selected features use less features and less redundant data for training, reducing the training times of models used. By reducing the number of features that enter our models, the curse of dimensionality is tackled as well. By concluding that a smaller subset of the data contains all relevant information for a models objective, we need less data to generalize the data accurately, leading to more accurate models. Thus the main objective of feature selection is captured. A priori our data contains features that are either irrelevant or redundant and by removing these features, we do not incur much loss of information. Here the distinction is made between redundancy and irrelevance. That is, a feature is considered irrelevant if it contains no or very little information with respect to an output. It is clear that removal of such a feature does not led to loss of information with respect to the output. A feature is considered redundant if after removing all irrelevant features, there are features that contain the same information with respect to the output, removing all but one feature that contain the same information, we incur no information loss for our modelling purposes.

Feature selection can be performed through filter, wrapper and embedded methods [47]. Wrapper methods are based on greedy search algorithms. Wrapper methods aim to search through all possible subsets of features and selects the subset that produces the best result for a specific model. As such, wrapper methods for feature selection are computationally expensive as for each subset of features the wrapper method has to train a model. Embedded methods effectively reduce the computational complexity of wrapper methods, by simultaneously performing feature selection and training models. The most infamous example of an embedded feature selection is that of LASSO regression [48], in which the model adds a regularization penalty to the cost function. The disadvantage of embedded methods is that they are model specific and in some cases are ambiguous to construct. Filter methods are a methodology that makes use of some external criteria, such as correlation or entropy of features, to construct the subsets of features. By allowing for external criteria, the filter methods remove the model specificity and dependency of embedded methods and allow the generalization of the feature selection process. The removal of model dependency also leads to an advantage of wrapper methods in terms of computational complexity, as the selection process is not dependent on model accuracy.

In section 2.1.2, we analyzed the dimensionality of features within the Aegon data sets. From one such data set, we observed upwards of two hundred features that can be used within our models to restore missing data. From observation of the number of unique values within features, we observed that there were nearly 75 features that only contained one unique value, indicating that these features are irrelevant. None of these features contains any unique information for a feature with missing values. From the remaining 166 features it is deemed unlikely that every single feature contains relevant and non-redundant information and as such feature selection is crucial. Furthermore, we note that there are many unique data quality checks that concern different data sets of high dimensionality. If we were to use wrapper methods for feature selection, this increases the time complexity of the restoration process greatly, leading to inefficacy of the Cleansing Algorithm as a process to restore missing data. On the other hand, the use of an embedded method for each data quality check, of which there are hundreds, is

an ambiguous task. This is not considering the increased complexity due to many more hyperparameter configurations and thus model evaluations needed to find the optimal settings.
For the Aegon data sets, the generalization of the feature selection process and independence of a great number of model evaluations are the prime reasons for the necessity of a filter based feature selection process.

The first steps that can be taken are obvious from first analysis of section 2.1.2. That is, if a feature within the data set only contains one unique value within all data points, we consider this feature to be redundant as it contains no information with respect to any output. Another step we are able to take with respect to this general analysis, is the distinguishing between categorical and numeric features. We have defined a categorical feature to be a feature that contains no more than 20 unique values in accordance with Aegon definitions. If a feature contains numeric values and contains more than twenty unique values, it is deemed a numeric feature. This leaves features that contain non-numeric values and over twenty unique values to be deemed redundant as it is not considered numeric nor categorical. Of course if a feature contains some (most likely) unique non-numeric identifier, such as ones full names or an address, it will be unique for each individual data point and one-hot encoding leads to these features also not containing any information with respect to any output. Removal of these features thus leads to no loss of information with respect to the output. Having removed the most obvious redundant and irrelevant features, we are still left with a great number of features. Further filtering of features is less evident and we will opt to introduce some methods with criteria that further the feature selection process. All methods we will propose will make use of the output feature to determine whether a feature is relevant through some criterion relative to the output feature. During this process redundant features are removed as well. In the following subsections, we shall first discuss the Correlation coefficients method as a modest feature selection approach based on hypothesis testing. The second approach we discuss is that of Treelet feature selection, where we propose an alternate use of the Treelet algorithm [27]. Finally, we shall discuss a tried and proven method for feature selection in that of the Fast Correlation-Based Filter [30].

4.1.1 Correlation Coefficients

A correlation coefficient can be considered a numerical measure associated to a type of correlation, where the correlation is some sort of relationship between features. There are many different correlation coefficients with unique characteristics. Most correlation coefficients, however, do agree with respect to their values. Correlation coefficients take on values between -1 and 1. In the case a correlation coefficient returns a -1 or 1 it indicates a strongest possible agreement with respect to the correlation, whereas a 0 indicates the strongest possible disagreement. There are exceptions to this rule, such as the distance correlation coefficient we will discuss, that takes on values between 0 and 1, 0 indicating disagreement and 1 indicating agreement.

4.1.1.1 Pearson Correlation Coefficient

The first coefficient considered, is that of the Pearson correlation coefficient. This coefficient provides a measure of linear correlation between two features. We provide the definition of the Pearson correlation coefficient for a sample [49]:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

- r_{xy} is the sample Pearson correlation coefficient between datasets x and y.
- n is the sample size.
- x_i is an individual point in dataset x.
- y_i is an individual point in dataset y.
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean of dataset x.
- $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the sample mean of dataset y.

In essence the Pearson correlation coefficient is a normalized measurement of covariance and as such will always be valued between -1 and 1, where a coefficient value of 1 indicates a perfect positive linear relationship between two features, -1 indicates that the relationship is a negative linear one and a coefficient of 0 teaches us that the two features are linearly independent. In general, if the value of the coefficient is positive, we expect an increase in one variable to result in increase in the other variable. It is the opposite for a negative coefficient, an increase in one variable is expected to result in a decrease in the other variable.
This leaves the question as to determine the decision boundary for the value of the coefficient such that we may consider a variable to be linearly independent and/or linearly insignificant with respect to another variable. One such method would be to discard any variable that results in a coefficient value of $r_{xy} < \alpha$ for some fixed $\alpha \in [0, 1]$.
One other method for determining whether or not the linear effect is significant, is to perform hypothesis testing.
Assume that x and y are independently drawn from a normal distribution, then the probability density function of the sample Pearson correlation coefficient can be shown to be:

$$f(r) = \frac{(1 - r^2)^{\frac{n}{2}-2}}{B\left(\frac{1}{2}, \frac{n}{2} - 1\right)} \tag{2}$$

- $B(\cdot)$ is the Beta function.
- n is the sample size.

Using this probability density, we can now test the null hypothesis that x and y are independent versus the alternative that x and y are not independent. That is, the p-value is defined to be the probability that, given a sample correlation coefficient r as determined by samples x and y, the value $|r'|$, where r' is a correlation coefficient as generated from some sample of size n from two independent variables x' and y', is greater than or equal to $|r|$. By taking the absolute value of the correlation coefficient and considering the probability density function takes on values on the interval $[-1, 1]$, this p-value is two-sided. Though, the probability density function does assume the samples to be normally distributed, by the law of large numbers, if the sample size n is chosen sufficiently large, this assumption has minor influence on the performance of this hypothesis test [50].
This testing setup, now gives us access to another strategy in determining whether the linear correlation between variables is significant. Namely, if the p-value as determined from the hypothesis testing is below some low threshold. For example, we may opt to discard variables that are shown to have $p < 0.05$ as variables that show significant linear influence on the other variable.
The upside of using the Pearson correlation coefficient is that we can now determine the features that have a significant linear correlation with respect to a variable with missing data by using the value of the correlation coefficient and the p-value. The downside of using the Pearson correlation coefficient is that this methodology is only capable of capturing linear relationships in our data and variables. Though, linear relationships are some of the most common found in data, they are far from the only type of relationship that can occur.

4.1.1.2 Spearman Correlation Coefficient

In order to compensate for the deficiency of the Pearson correlation coefficient, in that it strictly is able to determine the significance of linear relationships between features, we introduce the Spearman rank correlation coefficient. This methodology is an example of non-parametric rank correlation. That is, the Spearman correlation coefficient gives an indication as to how well the relation between two features or variables can be represented by a monotonic function. We provide the definition of the Spearman rank correlation coefficient for samples:

$$\begin{aligned} r_{xy} &= \frac{cov\left(R(X)R(Y)\right)}{\sigma_{R(X)}\sigma_{R(Y)}} \\ &= \frac{\sum_{i=1}^{n}(R(x_i) - R\bar{(}x))(R(y_i) - R\bar{(}y))}{\sqrt{\sum_{i=1}^{n}(R(x_i) - R\bar{(}x))^2}\sqrt{\sum_{i=1}^{n}(R(y_i) - R\bar{(}y))^2}} \end{aligned} \tag{3}$$

- r_{xy} is the sample Spearman correlation coefficient between datasets x and y.
- $\sigma_{R(X)}$ is the standard deviation of the ranks of dataset x.
- $\sigma_{R(Y)}$ is the standard deviation of the ranks of dataset y.
- $cov(R(X), R(Y))$ is the covariance of the ranks of datasets x and y.
- n is the sample size.
- $R(x_i)$ is the rank of an individual point in dataset x.
- $R(y_i)$ is the rank of an individual point in dataset y.
- $\bar{R(x)} = \frac{1}{n}\sum_{i=1}^{n} R(x_i)$ is the sample mean of ranks of dataset x.
- $\bar{R(y)} = \frac{1}{n}\sum_{i=1}^{n} R(y_i)$ is the sample mean of ranks of dataset y.

Similar to the Pearson correlation coefficient, the Spearman rank correlation coefficient takes on values between -1 and 1. However, for the Spearman coefficient, values -1 and 1 indicate that there exists an exact monotonic relationship between features of x and y. A value of 0 indicates that there is no correlation between the ranks of the features. A positively valued coefficient indicates that whenever the value of one variable increases, so does the other and for negatively valued coefficients, when one variable increases, the other decreases. This is conform the definition of monotonicity.
Similar to the Pearson correlation coefficient, we may adopt two different strategies in determining redundancy of variables for the determination of target variables. On the one hand, we may decide on some fixed constant boundary for the minimum value of the Spearman coefficient necessary for a variable to be deemed to have a significant monotonic relation to the target variable. On the other hand, we may construct a hypothesis test to determine a p-value, indicating the probability that the rank correlation would occur with respect to uncorrelated features. Based on this p-value, we can define some $\alpha \in (0, 1)$ such that, if the p-value is lower than this value, we consider the monotonic relation to be significant, rejecting the null hypothesis. It is of note that in this scenario, by assuming ranks, the Spearman rank correlation coefficient becomes non-parametric and does not suffer from the assumption of normality from the Pearson correlation coefficient.
The Spearman rank correlation coefficient now allows us to capture any monotonic relationship within our data, increasing the power of our feature selection in the sense that we are able to capture more relationships, at the cost of allowing more features to pass through to our models, increasing complexity. The Spearman rank correlation coefficient is also capable of capturing the linear relationships of the Pearson correlation at the cost of more time complexity due to having to rank the different observations in the samples.
Though the set of monotonic relationships is much more vast than that of just linear relations, a vast number of monotonic relations show some significant form of linear influence and thus can still be captured with the Pearson correlation coefficient. However, the set of monotonic relationships is vast and diverse enough that the Spearman rank correlation coefficient provides us with a stronger test to uncover underlying relationships between different features. However, the set of monotonic relationships is still just a relatively small subset of all possible relations two features can exhibit. One may think of common non-monotone relations such as, but not limited to, sines, cosines, many polynomials of even order.

4.1.1.3 Distance Correlation Coefficient

An even more powerful coefficient is given by the distance correlation coefficient. Distance correlation is a measure of dependence between two random vectors [4]. Their dimensions need not necessarily be the same.
In order to be able to determine the distance correlation coefficient, we shall start by computing the pairwise distance matrices:

$$a_{j,k} = ||x_j - x_k|| \tag{4}$$

$$b_{j,k} = ||y_j - y_k|| \tag{5}$$

- $||\cdot||$ is the Euclidean norm.
- x_j and x_k are an individual points in dataset x.
- y_j and y_k are an individual points in dataset y.

Then from this pairwise distance matrix, we are able to determine doubly centered distance matrices:

$$A_{j,k} = a_{j,k} - \bar{a}_{j,.} - \bar{a}_{.,k} + \bar{a}_{.,.} \tag{6}$$

$$B_{j,k} = b_{j,k} - \bar{b}_{j,.} - \bar{b}_{.,k} + \bar{b}_{.,.} \tag{7}$$

- $\bar{a}_{j,.}$ is the j-th row mean of matrix $(a_{j,k})$.
- $\bar{b}_{j,.}$ is the j-th row mean of matrix $(b_{j,k})$.
- $\bar{a}_{.,k}$ is the k-th column mean of matrix $(a_{j,k})$.
- $\bar{b}_{.,k}$ is the k-th column mean of matrix $(b_{j,k})$.
- $\bar{a}_{.,.}$ is the grand mean (the mean of all row/column means) of matrix $(a_{j,k})$.
- $\bar{b}_{.,.}$ is the grand mean (the mean of all row/column means) of matrix $(b_{j,k})$.

The squared sample distance covariance and sample distance variance is then given by:

$$dCov^2(X,Y) = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}B_{j,k} \tag{8}$$

$$\begin{aligned} dVar(X) &= dCov(X,X) \\ &= \sqrt{dCov^2(X,X)} \end{aligned} \tag{9}$$

Taking the square root of the squared sample distance (co)variances yields the non-squared counterparts. Finally, we are now able to formulate the actual sample distance correlation:

$$\begin{aligned} r_{xy} &= dCor(X,Y) \\ &= \frac{dCov^2(X,Y)}{\sqrt{dVar(X)}\sqrt{dVar(Y)}} \end{aligned} \tag{10}$$

What makes this sample distance correlation coefficient so powerful, is that whenever the coefficient becomes zero-valued, the two features considered are independent. This is in contrast to the two previous coefficients, where only no linear or monotone relation can be concluded. Furthermore, the distance correlation coefficient can thus measure both linear and non-linear relations between two random variables. Just like the Pearson and Spearman correlation coefficients, the distance correlation coefficient can directly be used in combination with some arbitrary decision boundary to determine which features should be considered for the restoration of missing features, and which not. In addition, we are also able to perform hypothesis testing with respect to distance correlation as well, though here we opt to use a permutation test. The permutation test tests the null hypothesis that both samples come from the same distribution. Under the null hypothesis, the distribution of the test statistic is obtained by calculating all possible values of the test statistic under possible rearrangements of the observed data. That is, first the difference between the two means of the input features is determined, then for many, if not all, different permutations of the combined pool of both features split up into two new features, the difference in the means is calculated. Then from this distribution of mean differences, we can determine the p-value to be the percentage of permutations that has an absolute value greater than the difference in means of the original feature. So, if we encounter a p-value of 0.12, then 12% of permutations have a mean difference greater than that of the original feature. By performing this test on the doubly centered distance matrices, this test now allows us to determine whether any significant form of dependence exists.

Now if we are to employ the strategy as with the other hypothesis tests, that we filter out features that are not under some arbitrary threshold, we are able remove independent features with respect to our target feature from the dataset. The distance correlation coefficient, in removing independent features, is able to capture a much wider range of possible relations within our data with respect to target features than the Pearson correlation coefficient and the Spearman rank correlation coefficient, enabling a more comfortable and less restrictive filtering of non-informative features. It should, however, be noted that the time complexity of the determination of the distance correlation coefficient is much greater than that of the Pearson and Spearman coefficients. The distance correlation coefficient clearly exhibits $O(n^2)$ time complexity, where the Pearson correlation coefficient and Spearman rank correlation coefficient can be done in $O(n)$.

4.1.1.4 Correlation Coefficient Feature Selection

Each correlation coefficient discussed allows us to employ a filter approach to feature selection that draws its inspiration from the work of J. Biesiada and W. Duch [51]. For example, we may decide to only consider features that have a p-value that is less than 0.05 associated to its Pearson correlation coefficient with respect to the output feature that contains missing data. Alternatively, one may opt to use the criterion that the absolute value of the correlation coefficient itself needs to exceed some threshold before it is deemed relevant. By leaving out features that are not deemed to have a significant relation to the output feature, with a p-value greater than 0.05, we are able to filter out some of the irrelevant features. The Pearson correlation coefficient also provides us with a tool to remove redundancy. Let us assume some feature has a p-value less than 0.05 associated to its Pearson correlation coefficient. If next, one other feature also has a p-value less than 0.05, we can observe the value of the Pearson correlation coefficient between both features that have been deemed to have a significant linear relationship. If the value of their shared Pearson correlation coefficient passes a threshold close to 1 or -1, we may deem one of the two feature redundant. In other words, if two features are shown to have a significant linear relationship with an output feature and they are nearly perfectly linearly correlated, they will both yield similar information with respect to the output feature.

If we were to employ such a strategy for each of these correlation coefficients, then Pearson would be computationally the least expensive followed by the Spearman ranking correlation coefficient and then the distance correlation coefficient. However, the inverse ordering suggests the power of hypothesis tests conducted using the coefficients. If we are to use the distance correlation coefficient, then the filter strategy based on hypothesis testing leads to a strong filter of all features that are shown to have some non-negligible dependency with the output feature. On the other hand, the Spearman correlation coefficient is only able to filter for features that are shown to have a non-negligible monotonic relation with the output feature. This of course then forms a more modest filtering of features. Finally, the Pearson correlation coefficient is only able to filter those features that show a non-negligible linear relationship to the output filter, again being a subset of the features the Spearman ranking correlation coefficient is able to filter out.

Based on these insights, one can choose the correlation coefficient as filter strategy that is either most thorough, being the distance correlation coefficient, or computationally the least expensive, being the Pearson correlation coefficient, or a balance in both, being the Spearman rank correlation coefficient. However, for the purposes of restoration of missing data with high accuracy, we would opt to go for the most thorough approach of the distance correlation coefficient, aiming to capture those features that are shown to have some non-negligible dependency to the output feature. This filter is computationally the most expensive and there is a strategy we may consider that can reduce this to some extent. This strategy we propose, makes use of the much lower computational complexity of the other two mentioned correlation coefficients to enhance the filtering process.

The strategy we propose, shall iteratively pass over all features. For each feature we will first perform the hypothesis testing for the Pearson correlation coefficient. If the p-value returned by the test is less than 0.05, then we consider the linear relation between the feature and the output to be significant and we continue to the next feature. If the p-value of a feature is greater or equal to 0.05, we deem the linear relation to be negligible and pass the feature to the next hypothesis test. Next, we test the feature

against the output using hypothesis testing for the Spearman rank correlation coefficient. If the p-value returned by this test is less than 0.05, we deem the monotonic relation between the feature and output feature to be significant and move to the next feature. In the case the p-value for this hypothesis test is greater than 0.05, we move to the final hypothesis test. Finally, we test the hypothesis corresponding to the distance correlation coefficient between the feature and the output feature. If the p-value of this test is less than 0.05, we deem there to be a non-negligible dependence between the feature and the output feature. In the case the p-value of this test is greater or equal to 0.05, we conclude that the dependence is insignificant and the feature is filtered out.

This strategy works under the insight that any significant linear relation, as deduced by the Pearson correlation coefficient, is a subgroup of monotonic relations, as deduced by the Spearman rank correlation coefficient. Whereas the monotonic relations, as deduced by the Spearman rank correlation coefficient, are a subset of the dependency relations, as deduced by the distance correlation coefficient. Thus if a feature shows to have a significant linear relation to the output feature, we know prior that it will show to have a significant monotonic relation and dependency as well. The same goes for a significant monotonic relation, that will be deemed to have significant dependency by the distance correlation hypothesis test. Thus by using this hierarchical strategy for correlation coefficient filtering of features, we are able to exploit the thoroughness of the distance correlation coefficient, while reducing computational strain by filtering out linear and then monotonic relations prior.

4.1.1.5 Correlation Coefficients in the Cleansing Algorithm

Having discussed the feature selection filter strategy relation to correlation coefficients, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical feature, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the output feature. For the correlation coefficient based feature selection process, we would iteratively pass through each input feature and perform the previously described strategy with respect to the output feature. Once a feature is deemed significant with respect to any of the three correlation coefficients, we would store this feature as well as an index as to which correlation coefficient allowed the feature to pass through the filter. The next feature that would be considered, will be checked for linear redundancy before entering the filter. That is, for a new feature that passes through the filter, the Pearson correlation coefficient is determined for each feature that has passed through prior. If any of these pairwise Pearson correlation coefficients were greater than 0.99, we deem that pair of features to contain the same linear information for the output feature, then we opt to filter out the new feature.

Once the processed has finished, we are left with a subset of the original input features that have successfully passed through the correlation coefficient filter and the index of the correlation coefficient that led to this decision and are passed on to the Model Selection node.

4.1.2 Treelets

The Treelet algorithm as proposed by A.B. Lee, B. Nadler and L. Wasserman is inspired by both hierarchical clustering trees and wavelets [27]. They identify two main goals for the algorithm. The first goal is to find a "natural" system of coordinates that reflects the underlying internal structure of the data and that is robust to noise. The second goal as proposed by Lee, Nadler and Wasserman is to improve the performance of conventional regression and classification techniques in the "large p, small n" regime by finding a reduced representation of the data prior to learning.

4.1.2.1 Treelet Algorithm

The algorithm as proposed in [27] works as follows:
We start with some measure of similarity between two variables s_i and s_j, namely M_{ij}. Though this

similarity matrix can be chosen to be any measure of similarity, it is proposed to choose

$$M_{ij} = |\rho_{ij}| + \lambda |\Sigma_{ij}|,$$

where the parameter λ is a nonnegative number. For this similarity

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}$$

represents the correlation between the two variables and

$$\Sigma_{ij} = \mathbb{E}[(s_i - \mathbb{E}s_i)(s_j - \mathbb{E}s_j)]$$

to be the covariance between the two variables.
At level $l = 0$ we set

$$x^{(0)} = [s_{0,1}, ..., s_{0,p}]^T,$$

with $s_{0,k} = x_k$. We set the Dirac basis,

$$B_0 = [\phi_{0,1}, \phi_{0,2}, ..., \phi_{0,p}],$$

where B_0 is the p by p identity matrix. We determine $\Sigma^{(0)}$ and $M^{(0)}$. We set $S = \{1, 2, ..., p\}$.
Next, three steps are defined that are repeated for $l = 1, ..., L$.

Step 1

In the first step, the aim is to find the variable pair that are shown to be most similar with respect to the similarity matrix. In other words, we find (s_i, s_j) such that M_{ij} has the greatest value:

$$(\alpha, \beta) = \arg\max_{i,j \in S} M_{ij}^{(l-1)}$$

where, $i < j$, and we only consider those variable pairs that are still in S.

Step 2

In the second step, the aim is to remove the correlation between the two variables found in the previous step. This is done through finding a Jacobi rotation matrix:

$$J(\alpha, \beta, \theta_l) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

where $c = \cos(\theta_l)$ and $s = \sin(\theta_l)$. In other terms, we aim to find $|\theta_l| \leq \frac{\pi}{4}$ such that $\Sigma_{\alpha\beta}^{(l)} = 0$ with $\Sigma^{(l)} = J^T \Sigma^{(l-1)} J$. We then update the Dirac basis and new coordinates accordingly: $B_l = B_{l-1} J$ and $x^{(l)} = J^T x^{(l-1)}$. Finally, the similarity matrix $M^{(l)}$ is updated.

Step 3

Now we assume that $\Sigma^{(l)}_{\alpha\alpha} \geq \Sigma^{(l)}_{\beta\beta}$ after the Jacobi rotation. We define the sum and difference variables at level l as $s_l = x^{(l)}_{\alpha}$ and $d_l = x^{(l)}_{\beta}$. We define the scaling and detail functions ϕ_l and ψ_l as columns α and β of the matrix B_l. The variable with lower covariance is removed from $S = S \setminus \{\beta\}$. At level l, we then acquire the orthonormal treelet decomposition:

$$x = \sum_{i=1}^{p-l} s_{l,i}\phi_{l,i} + \sum_{i=1}^{l} d_i\psi_i$$

where we recognize the first sum as a coarse-grained representation of the data, it forms the projections in the main principal directions. The second sum represents the differences between the node representations at two levels in the tree.

4.1.2.2 Treelet Feature Selection

The Treelet algorithm as described was proposed with the main goals in mind to discover a subset of features that reflect the underlying internal structure of the data and acquire a reduced representation of the data prior to learning. However, we will, rather than use the Treelet algorithm for its intended purpose, opt to use the methodology of the Treelet algorithm to construct an alternate filtering approach for feature selection within the Cleansing Algorithm.

In order to understand the ideas behind the use of the Treelet algorithm as a filter, we will first provide a toy example of output of the Treelet algorithm in figure 4.1:

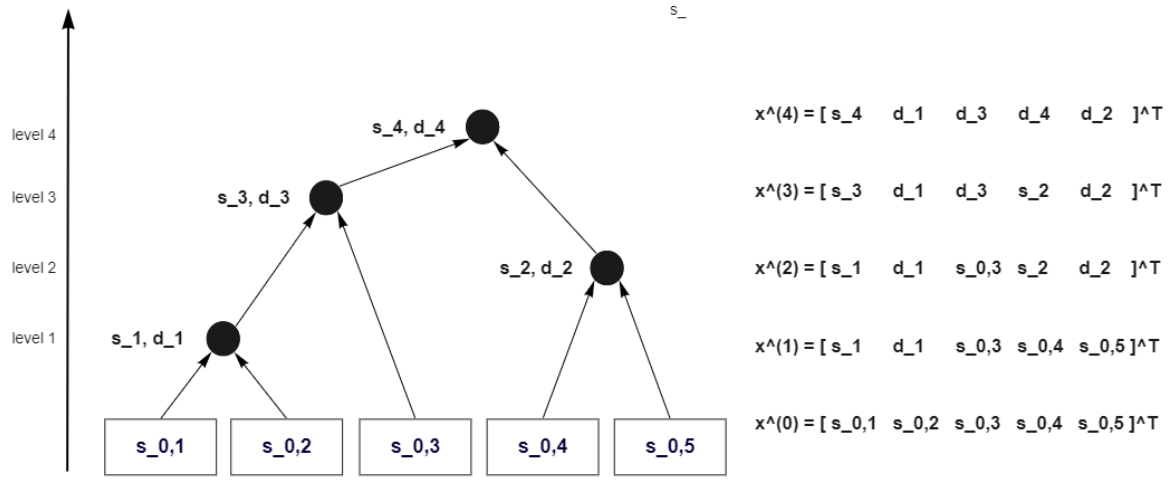


Figure 4.1: Example of a hierarchical tree resulting from the Treelet algorithm.

In the figure, we provide a hierarchical tree that results from the pairwise merging of features into clusters at each level l. In the figure we observe that, at level $l = 0$, the Treelet algorithm decides to cluster feature $s_{0,1}$ and $s_{0,2}$ as these would have the greatest similarity score. Similarly, at level $l = 1$, the Treelet algorithm decides to cluster feature $s_{0,4}$ and $s_{0,5}$ as these would have the second greatest similarity score. At level $l = 2$, we observe that the residual feature that was left after having decorrelated the original feature $s_{0,1}$ with respect to feature $s_{0,2}$, has the third greatest similarity score with respect to $s_{0,3}$ and thus this feature is merged into the already existing cluster containing features $s_{0,1}$ and $s_{0,2}$. The Treelet algorithm ends with the merging of the two cluster to merge all features into one big cluster, based on the similarity score between some pair of features, one from each cluster, being the greatest.

By observing the hierarchical tree resulting from the Treelet algorithm process, the original purpose of the algorithm becomes clear. At each level the original five features have a lower dimensional representation. For example at level $l = 2$, we have a 3-dimensional representation of the original data consisting of the first, third and fourth indices within $x^{(2)}$. However, for filtering purpose, we do not know prior what the appropriate number of features will be with respect to a output feature, it can occur at any of the levels in the Treelet process. For this reason, if we are to use the Treelet algorithm as a filter for feature selection, we need to propose an alternative criterion to determine the optimal lower dimensional representation. This criterion will determine whether we should continue the Treelet process to further levels or when we should stop clustering features and thus will be aptly named a "stopping criterion" as such.

Since the level of the Treelet process is not a sufficient stopping criterion, we will need to analyze different measures available to us at each level. The other main measure to our disposal, is that of the similarity score. The similarity score is more suited to be used as a stopping criterion for the feature selection process as it can be set prior. For example, we can set a stopping criterion for the similarity score to be 0.1. This would mean that once, at some level l, the next greatest similarity score is less than 0.1, we stop the Treelet process and consider this clustering of features. The resulting clusters will then contain features whose mutual similarity with each other is greater than 0.1. Using this clustering, we would need to find the cluster that contains the desired output feature and we would consider all features within the same cluster as this output feature to contain most of the significant information for this output feature. This type of stopping criterion, we shall refer to as a "hard" stopping criterion in that it requires a "hard" value to be set prior to be all-encompassing for filtering features on all data sets.

As an alternative to this hard stopping criterion, we shall propose an alternative. At any level l in the Treelet process, we have access to the similarity score that led to the clustering at level $l - 1$. If the percentage change with respect to the similarity score at level $l - 1$ and level l is greater than 75%, we opt to stop clustering as any feature added to clusters from level l will have minor impact on the output feature based on their similarity score. This type of stopping criterion, though still requiring a value to be set prior, is less "hard" in that it will handle cases where there are a large number of features around the otherwise "hard" stopping criterion better.

A similar process to the "hard" stopping criterion can be used to further filter out redundancy. If we set this stopping criterion to a value close to 1, we are left with a "redundancy" clustering of all features such that their similarity scores indicate them to be as similar as they can be. Using this representation of the features in combination with the features that were selected from the "relevancy" clusterings as proposed above, we can filter out redundancy. For each pair of features, within the cluster of relevant features, that share a cluster in the redundancy clustering, we filter out one of those features.

4.1.2.3 Treelet in the Cleansing Algorithm

Having discussed the feature selection filter strategy relation to the Treelet algorithm, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical feature, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the output feature.

For the feature selection filter based on the Treelet algorithm, we first add the output feature to the collection of input features. After, the similarity matrix is constructed using $\lambda = 10^{-5}$. In doing so, the similarity matrix is an absolute correlation representation between features, that is sorted based on their absolute covariance in the case two similarity scores would otherwise be equal. Rather than constructing and storing the entire similarity matrix, we note the fact that it is symmetrical and only consider the upper triangular matrix as representation.

Next, the greatest similarity score is found through a hard search through the upper triangle of the similarity matrix with exception of the diagonal entries. Having found the indices of the two features that have the greatest similarity score, we decorrelate the two through a binary search for the value of θ_l that leads to $|\Sigma^{(l)}_{\alpha\beta}| = 0 + \epsilon$ for $\epsilon = 10^{-8}$.
Following this, we cluster the features together in a new cluster, if in the current clustering neither of the features already exist. If one of the features is present in an existing cluster, we add the other feature to that cluster. If both features are present in different existing clusters, these two clusters are merged into one.
The Treelet process, then, continues iteratively until the "soft" stopping criterion of 90% is met and those features within the same cluster as the output feature are considered the new filtered input features or the "hard" stopping criterion of 10% is met, whichever occurs first.
During the Treelet process, at the "hard" stopping criterion of 0.99, the redundant clustering is stored. This clustering is then used to remove the redundancy from the subset of features that passed through Treelet process.
Once the process has finished, we are left with a subset of the original input features that have successfully passed through the Treelet filter. These are then passed on the the Model Selection node.

4.1.3 Fast Correlation Based Filter

The final feature selection methodology we propose is that of the Fast Correlation Based Filter or FCBF for short. This methodology for feature selection was first proposed by L. Yu and H. Liu in 2003 [36]. In their work they compare their FCBF approach extensively on various sets of data with respect to some of the more popular feature selection filters, such as ReliefF [6], CorrSF [7] and ConsSF [8]. It was shown to significantly reduce the number of features that pass through its filter compared to the other methods, while maintaining a high accuracy of models using the features that pass through the filter. Most notably the authors use of FCBF reduced the running time of filtering on average by 80% when compared to the next fastest average given by ReliefF. It is for the reason that its running time is shown to reduce dramatically, its accuracy remains competitive and the strict filtering of features, that we consider the FCBF approach for the restoration of missing data. It should, however, be noted that the FCBF approach was originally only made for the task of classification using categorical features. Thus we propose a modification to the existing FCBF approach such that it is adequately able to handle numeric features simultaneously.

4.1.3.1 Fast Correlation Based Filter Algorithm

In their work [36], L. Yu and H. Liu first propose the use of symmetrical uncertainty rather than a Pearson correlation coefficient . They deem it not safe to always assume linear correlation between features in the real world as linear correlation measures may not be able to capture correlations that are non-linear in nature. Symmetrical uncertainty as measure tackles this problem by making use of entropy. The entropy of a variable X is defined as

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)) \tag{11}$$

- $P(x_i)$ is the prior probabilities for all values of X.

The entropy of a variable X after observing values of another variable Y, on the other hand, is given by

$$H(X|Y) = -\sum_j P(y_i)\sum_i P(x_i|y_j)\log_2(P(x_i|y_j)) \tag{12}$$

- $P(x_i|y_j)$ is the posterior probabilities for all values of X given values of Y.

Then, from the entropy, we are able to compute the so-called information gain or mutual information [41], which is then given by

$$IG(X|Y) = H(X) - H(X|Y). \tag{13}$$

Finally, the symmetrical uncertainty is calculated using:

$$SU(X,Y) = \frac{2IG(X|Y)}{H(X)+H(Y)}. \tag{14}$$

This measure provides us with a tool with which we are able to perform feature selection. As stated in [30], symmetrical uncertainty compensates for information gains bias towards more values. A symmetrical uncertainty with value 1 indicates that the value of either one predicts the value of the other completely and a value of 0 indicates that the two features are independent.

The Fast Correlation Based Filter algorithm itself starts by setting some threshold δ beforehand. For example if $\delta = 0.3$, we filter out all features that have a symmetric uncertainty less than or equal to 0.3. Thus the fast correlation based filter iteratively passes through all features, storing those that have a greater symmetrical uncertainty than the given threshold δ with respect to the output feature. The symmetric uncertainty of feature i with respect to target feature c is noted as $SU(i|c)$.

The second step involves the process of removing redundant features from the filter set of features from the first step. To start, the second step sorts the set of features from the first step based on the value of their symmetric uncertainty with respect to the output feature. This set of sorted features starts with the feature that has the greatest symmetric uncertainty with respect to the output feature, which is the feature, according to this metric, that contains the most information with respect to the output feature.

Next, the algorithm iteratively passes through the set of sorted features. The first feature is automatically added to a new set of non-redundant features. For the second feature, we determine the symmetric uncertainty of this second feature with respect to the first feature $SU(i|j')$, which is the only feature of the non-redundant set. Then if $SU(i|j') > SU(i|c)$, we consider that feature i contains more information with respect to the feature within the non-redundant set j', than it does for the output feature c. In this case FCBF states that the feature with index i is redundant and eliminated. If $SU(i|j') \leq SU(i|c)$, the feature i contains more information with respect to the output feature c than it does for the non-redundant feature j'. In this case, feature i is added to the set of non-redundant features. For every feature, we repeat this process, where we determine the symmetric uncertainty between each feature within the non-redundant set, where if for any one of the non-redundant features the symmetric uncertainty is greater than the symmetric uncertainty with respect to the output feature, we consider the feature to be redundant. If the feature contains more information with respect to the output feature, than with respect to all non-redundant features, the feature is considered non-redundant and added to the set for the next feature to be considered.

Once this filtering process completes, we are left with a set of relevant and non-redundant features in accordance to the fast correlation based filter process.

4.1.3.2 Fast Correlation Based Filter Feature Selection

The Fast Correlation Based Filter is an alternative filter methodology for feature selection that was shown to outperform the popular ReliefF and other methodologies in terms of number of features filtered out, time complexity and accuracy of models using this filtering process. Apart from that, the process discussed above shows that FCBF aims to remove irrelevant features through the setting of an arbitrary boundary on the symmetric uncertainty and removes redundancy from this set of relevant features through the comparison of symmetric uncertainty of other relevant features and the output feature. On paper, the Fast Correlation Based Filter may be ideal for the purpose of feature selection, however, it has a major deficiency for use in the restoration of missing data on Aegon data sets. This deficiency occurs due to the fact that entropy is difficult to determine for numeric features. One method to tackle this deficiency is to use the maximum entropy as an approximation for the entropy for numeric features. Under this assumption one would use the maximum entropy of the normal distribution associated with the feature being considered. This maximum entropy for the normal distribution is given by:

$$H(X) = \frac{1}{2}(1 + \log(2\pi\sigma^2)) \tag{15}$$

Here σ^2 denotes the sample variance of feature X. Using the maximum entropy as an approximation for entropy is not computationally expensive, but the entropy resulting from this calculation also does

not well capture the underlying distribution of the feature and serves more as a methodology to compare the numeric features universally under the same assumption. The normality case as maximum entropy especially fails in the case of semi-numeric features such as the one discussed in section 2.1, where the majority of cases occur on exactly the value zero, whereas a subset looks to exhibit a bell-shaped curve on values much greater. This disparity in density of the data on its domain is not well captured by using maximum entropy.
Alternatively, for a better approximation one can approximate the mutual information for numeric features by performing binning to discretize these features. In the work of A. Kraskov, H. Stöfbaur, and P. Grassberger, an improvement to the binning approach is proposed, such that the entropy estimates are based on k-nearest neighbor distances [37]. The idea behind the use of k-nearest neighbors, is that when the distances required to find the first k neighbors is large, the dispersion of the data is large and thus the features entropy will also be large. Their estimator for mutual information was shown to be data efficient in that they were able to resolve structures down to the smallest possible scales (for the case k=1), it was adaptive in that the resolution was higher where data were more numerous and it had minimal bias.

This approximator for mutual information is thus well-suited for use on the mutual information between two numeric features on the Aegon data sets as it can adequately and efficiently handle the semi-numeric features and features in which multiple distinct clusters of data may occur.
This leaves us with the case where we need to determine the mutual information between two features, one of which being a numeric feature and one of them being a categorical feature. An appropriate approach to tackling this case is presented in the work of B. C. Ross [38]. In this work the approximation of mutual information between a categorical and numeric feature is determined through a combination of the nearest neighbor approach employed by A. Kraskov, H. Stöfbaur, and P. Grassberger and the standard mutual information determination between two categorical features.

By the symmetry of the mutual information, we need not determine a separate approach for the determination of mutual information approximation for the categorical and numeric feature case and for the numeric and categorical feature case, i.e. $IG(X, Y) = IG(Y, X)$.
To summarize the use of the FCBF approach within the setting of the Cleansing Algorithm. We start by determining the symmetric uncertainty of input features and output feature entering the Data Structure Analysis node.
If an input and output feature pair are both categorical in nature, we directly compute the entropy of the input and output feature and the entropy of the input feature given the output feature. From these calculations, we are then able to calculate the information gain or mutual information of the input and output feature pair. The symmetric uncertainty is then calculated using this information gain and the entropies of the input and output features.
In the case the input and output feature pair, concern a numeric and categorical feature, we approximate the entropy of the numeric feature using the nearest neighbor approach as proposed in the work of A. Kraskov, H. Stöfbaur, and P. Grassberger and determine the entropy of the categorical feature in accordance with the definition of entropy. The mutual information between the in- and output pair is determined through the methodology proposed in the work of B. C. Ross. This leaves us with the direct computation of the symmetric uncertainty between the input and output feature using the exact entropy for the categorical feature, the approximated entropy for the numeric feature and the approximated information gain between the categorical and numeric features.
In the case the input and output feature pair, concern two numeric features, we approximate the entropy of the input and output features as well as the approximation for the information gain as per the work of A. Kraskov, H. Stöfbaur, and P. Grassberger. Direct computation of the symmetric uncertainty is performed using these three approximations.

Within the Cleansing Algorithm the boundary for symmetrical uncertainty is set to $\delta = 0.01$. As such each symmetrical uncertainty of a feature calculated with respect to the output feature that is greater than this δ is stored an shall be considered, whereas all other features are discarded. As the approximations for symmetric uncertainty can lead to negative values, which are mapped to the value zero, it is especially important to set the δ parameter greater than zero.

Next, for each feature, the Fast Correlation Based Filter process is followed, in that we sort the fea-

tures based on the value of their symmetrical uncertainty. The feature with the greatest symmetrical uncertainty with respect to the output feature is stored in the final filtered feature list. After, we iteratively pass through all other input features in descending order, adding any feature that is shown to have no symmetrical uncertainty, with respect to any feature within the final filtered feature list, greater than its symmetrical uncertainty with respect to the output feature. Any feature that has but one symmetrical uncertainty with an existing feature in the final filtered feature list greater than its symmetrical uncertainty with respect to the output feature, will be discarded. To understand the idea behind this process that removes redundancy, we need to note that if one input feature X has a greater symmetrical uncertainty with respect to the output feature Z than the input feature Y has, the input feature X is thought to contain more information with respect to Z than Y. If the input feature Y then shows to have a symmetrical uncertainty with respect to X greater than Z, then it is thought that feature Y contains more information on X than Z, thus the information Y contains on Z is most-likely already captured within the information of feature X on Z.

4.1.3.3 Fast Correlation Based Filter in the Cleansing Algorithm

Having discussed the feature selection filter strategy related to the Fast Correlation Based Filter algorithm, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical feature, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the output feature.

For the feature selection filter based on the Fast Correlation Based Filter algorithm, we enter the set of input features and the output feature into the algorithm. The features then follow the FCBF process as discussed above until we are finally left with a filtered set of features in that we only consider features with a symmetrical uncertainty greater than the boundary δ as well as removing redundancy by removing those features that are deemed to contain more information with respect to an existing input feature than the output feature. The filtered features are then passed on to the Model Selection node.

4.2 Clustering

The main objective of clustering is generating distinct clusters of data within a data set. The idea behind generation of these clusters, is that these different clusters contain subsets of the data set that are more similar to other data points within their mutual cluster, rather than those of a different cluster. The similarity can take on many forms, though some popular notions include distances between data points, local data point densities and intervals. Using such similarity measures, clustering allows us to discover structures in unlabeled data. That is, clustering provides us with an unsupervised tool with which we may recognize structures in our data prior to data being entered into other supervised models.

From the analysis of features within the Aegon data sets as in 2.1.2.1, we observed an example of a feature that consists of for one part many data points that had the same constant value zero and for another part of data points with greater but varying values. Such a feature is an example of a feature containing correctly missing data. A feature as this one will be recognized by the Cleansing Algorithm as being numeric of nature as there are enough occurrences of uniquely valued data points in accordance with the definition of a numeric feature as defined by Aegon. As discussed previously, the processes of normalization for numeric features and even feature selection can be negatively influenced by the presence of these zero-valued correctly missing values. As such it is crucial to find a methodology that can separate the data points that are only zero-valued from those with non-zero values. Clustering provides us with the tools necessary to tackle this problem.

Even though the feature may be recognized as being numeric, the presence of correctly missing data leads this feature to be partly categorical, thus making the feature semi-numeric. It is in essence implied within the feature that there exists some split within the data such that a data point either belongs to the subset containing correctly missing data or the rest of the data. Now let us assume we wish to restore missing data within such a feature, in example this is our output feature. If we wish to directly generate a supervised model for the restoration of missing data, we need to start with feature selection.

Performing feature selection with such an output feature, will lead to features being selected that are mainly able to recognize whenever a missing data point should be restored with a zero-value. This is due to the fact that the correctly missing data dominates the rest of the data. The generation of a supervised model for restoration also struggles with the majority portion of zero-values. If model training is able to generate a model that can successfully recognize when a data point should be restored as a zero-value, it is already able to achieve a high accuracy in terms of restoration and will find difficulty in further improving on this base accuracy due to the small influence an error on a non-zero value exhibits. The model training may even not be able to generate a model that can restore non-zero values at all as features passed to it from feature selection do not contain the information needed to determine the non-zero values. If we are able to successfully separate the two subsets of data within the output feature, we can perform feature selection to determine which features contains information for the subset of the data that is not zero-valued and we can generate two separate models based on the type of missing data we need to restore. In the case a data point is deemed to belong to the correctly missing data subset, we can restore this with a zero-value and we can generate a supervised model for the other subset of the data.

As there are many different types of clustering algorithm that are able to handle semi-numeric features within the Aegon data, we shall focus on trying to automate and generalize the clustering process to handle many different types of relationships that may lead to the disparities such as the example previously discussed. As such, we first propose the use of a relatively simple and effective algorithm known as K-Means.

4.2.1 K-Means

The K-Means algorithm is an example of a centroid-based clustering algorithm. In this subset of clustering algorithms, each cluster can be represented by a centroid, which can be seen as a data point that lays centrally within a cluster that need not necessarily be part of the original data set. For the process of K-Means the parameter k needs to be set prior to using a K-Means algorithm. It is this parameter k that determines the set number of clusters one wishes to cluster a set of data with. The most popular algorithm used for approximating the solution to the optimization problem related to K-Means is that of LLoyd's algorithm [39]. It is for this reason LLoyd's algorithm is often directly referred to as the K-Means algorithm.

4.2.1.1 K-Means Algorithm

The Lloyd's algorithm consists of three main steps:

The first step is the initialization of k means m_i, by sampling k random data points within the training data.

The second step is to assign each other data point within the training set to the class of which the m_i is closest based on some distance measure. This then results in k distinct sets of data S_i partitioning the training set: $S = \{S_1, \ldots, S_k\}$.

The third step is to update the means m_i to be the means of the sets S_i:

$$m_i = \arg\min_S \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{x_i, x_j \in S_i} ||x_i - x_j||^2 \tag{16}$$

Afterwards, the second and third steps are repeated until the means m_i no longer change.

4.2.1.2 One-Dimensional K-Means Clustering

For use as a clustering algorithm within the Cleansing Algorithm, we need first recognize some drawbacks of Lloyd's algorithm. The use of a Euclidean distance measure does not bode well if we wish to cluster our output feature, while making use of multiple input features of these data points due to the Euclidean distance becoming inflated in high-dimensional spaces. Even in two-dimensional spaces, the algorithm is

shown to not perform well under certain circumstances that arise due to clusters not containing similar numbers of points, clusters not being circular, the choice of an inappropriate value for k and initialization of the k centroids. In light of this and the fact that for features containing correctly missing data, the correctly missing data takes on a value outside of the realised interval of the rest of the data, we propose the use of the k-means algorithm in the one-dimensional space. Within the Cleansing Algorithm the k-means algorithm is used to cluster the one-dimensional output feature into k distinct clusters.

This then leaves us to determine the appropriate value for k for clustering of the one-dimensional output feature. We propose the use of the Silhouette method as proposed by P. Rousseeuw [40]. Silhouette is a method of validation of consistency within clusters of data. The Silhouette method uses a silhouette value as a measure to determine how well a data point matches the cluster it is assigned to versus how well it matches the clusters it is not assigned to. The silhouette value is given by:

$$s(i) = \begin{cases} \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}, & \text{if } |C_i| > 1 \\ 0, & \text{if } |C_i| = 1 \end{cases} \tag{17}$$

Where,

- $a(i)$ is the average intra-cluster distance.
- $b(i)$ is the average inter-cluster distance.
- C_i is the set of data points within the same cluster as i.

In order to calculate the silhouette value for a data point, we thus need to determine the average intra-cluster and inter-cluster distances. We shall first provide the average intra-cluster distance:

$$a(i) = \frac{1}{|C_I|-1} \sum_{j \in C_I, i \neq j} d(i,j) \tag{18}$$

Where,

- C_I is the cluster of data points containing i.

Secondly, the average inter-cluster distance is defined as:

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i,j) \tag{19}$$

Where,

- C_J is a cluster of data points outside the cluster containing i.

The silhouette value ranges from -1 to 1, where a silhouette value of 1 indicates that a data point matches its assigned cluster well and matches other clusters poorly. By taking the mean over these silhouette values, for all data points, we determine the silhouette score. Similarly to the silhouette value, the silhouette score gives us an indication as to how well data points match their assigned clusters versus how the data points match other clusters. As such the silhouette score can be used as a measure to determine the optimal value of k for the k-means algorithm. In order to determine the optimal k-value for clustering the data points within the output feature, the k-means algorithm is applied on the output feature for a range of different values of k. The optimal value of k is given by that k-means clustering that led to the greatest silhouette score.

4.2.1.3 Output Clustering in the Cleansing Algorithm

Having discussed the clustering strategy related to k-means, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical feature, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the output feature.

Once the output feature is recognized to be a numeric feature by the Data Transformation step, it can be used for clustering by means of K-Means on the one-dimensional output feature. K-Means is then performed for values of k ranging from two to twenty. Due to the fact that the k-means algorithm can converge to different local minima dependent on the initialization of the centroids, we opt to run the k-means algorithm ten times with different starting centroids. For these ten runs per value of k, we calculate the silhouette scores. The best value of k for clustering is determined to be that k for which the average silhouette score is the greatest.

Once the optimal k is determined, we label the data based on the centroid of the cluster that the value of the output feature for a specific data point belongs to. The data points with their respective input and output feature values, are then split into k clusters, where each cluster is passed along to Feature Selection separately.

The k-means algorithm on the one-dimensional output features, as discussed previously, is capable of discovering clusters of data with respect to the output feature. In doing so, the algorithm can capture a wide range of underlying relationships between input features and the output feature. This method of clustering does, however, have a downside. If we wish to restore the missing value of a data point within the output feature, we need to first determine to which output cluster the data point belongs. Since the value of this data point in the output feature is missing, we cannot directly use the k-means model that was constructed.

As such we would employ the use of a dissimilarity metric such as that of the silhouette score to determine to which cluster the data point best matches based on this metric. In the higher dimensional space concerning the many input features, however, the choice of the distance metric is important. As the number of input features available can differ greatly between different data quality checks, we need an adaptive distance metric, such as the Minkowski distance as used by L. Schleider, et al [42].

Another method of determining to which cluster the missing data point belongs, is through the use of a supervised machine learning model that can perform classification. One would train the model based on the clustering as determined by the k-means algorithm and then aim to learn the labelling based on the values of the input features. Once trained, the model can then be used to classify missing data points to one of the clusters. In the Cleansing Algorithm this classification is performed by means of random forest.

Finally, since we have performed the scraping of the data quality checks, we have access to the features within the data quality check. If we are to encounter a semi-numeric feature containing correctly missing data, we can match the other features found in the data quality check to determine to which cluster the true missing data points belong. This can be done both through the silhouette score as well as the machine learning model to perform classification.

As this approach to clustering is based purely on the output feature, we shall also refer to this methodology as output clustering in the rest of this work.

4.2.2 Diffusion Maps

As an alternative to the clustering approach through the k-means algorithm, we propose the use of Diffusion Maps. Where the k-means algorithm proposed needs data points containing missing values to first

be clustered before determining which cluster is appropriate, the Diffusion Map clustering shall aim to determine the feature that gives rise to the clustering in the output feature.

Diffusion maps are a dimensionality reduction algorithm introduced by Coifman and Lafon which computes a family of embeddings of a dataset into Euclidean space whose coordinates can be computed from the eigenvectors and eigenvalues of a diffusion operator on the data [26]. In essence the diffusion map embeddings are found by considering the data points as nodes of a graph with their corresponding distances between data points, according to some metric, as the weighted edges of the graph. In doing so, the underlying geometry of the set can be estimated. In order to find local geometry efficiently, it is of importance to give more weight to paths over data points, than to an ordinary Euclidean distance measure.

4.2.2.1 Diffusion Map Algorithm

The construction of a diffusion map works in accordance the diffusion map algorithm as provided in the work of E. S. Roozemond [25]. First we start by requiring:

- A finite set X with n m-dimensional elements x_i, for $i \in 1, 2, 3, \ldots, n$.
- A scale parameter $\epsilon \in \mathbb{R}^+$.
- A cut-off parameter $\delta \in \mathbb{R}^+$.
- The dimension of diffusion embedding m.

Finally, a kernel k is required that satisfies the following properties:

- $k(x, y)$ is symmetric: $k(x, y) = k(y, x)$.
- $k(x, y)$ is positivity preserving: $\forall x, y \in X : k(x, y) \geq 0$.
- The kernel can be normalized to a probability, that is the integral: $\int_X k(x, y)q(y)dy$ is finite for any x.

The first step in the process of constructing the diffusion map starts by calculating the matrix:

$$K_{ij} = k_\epsilon(x_i, x_j) \tag{20}$$

Next, this matrix of kernel measures between all data points, is row normalized to convert the kernel distances to probabilities:

$$P_{ij} = \frac{k_\epsilon(x_i, x_j))}{\sum_{x_j \in X} k_\epsilon(x_i, x_j))} \tag{21}$$

Following this, we determine the eigenvalues $\{\lambda_k\}_{k\in\mathbb{N}}$ and eigenvectors $\{\psi_k\}_{k\in\mathbb{N}}$ of this matrix P and order them in descending order based on the absolute eigenvalue. Once completed we have all needed to construct the diffusion map:

$$\Psi_t(x_i) \; = \; \begin{bmatrix} \lambda_1\psi_1(i) \\ \lambda_2\psi_2(i) \\ \lambda_3\psi_3(i) \\ \cdots \\ \lambda_{s(\delta,t)}\psi_{s(\delta,t)}(i) \end{bmatrix}$$

Where,

- $s(\delta, t) = \{\max k \in \mathbb{N} : |\lambda_k|^t > \delta|\lambda_1|^t\}$ is a integer to determine dimensionality of the embedding.
- λ_j is the eigenvalue corresponding to eigenvector ψ_j.
- $\psi_j(i)$ is the i-th element of eigenvector ψ_j.

In essence, the diffusion map process has used the kernel to represent the geometric properties of the data set X. The matrix K induces a graph based on these kernel distances on our data and the matrix P associates a Markov chain to this graph representation. It is possible to run the Markov chain forward in time, which then corresponds to taking a random walk with more steps.

4.2.2.2 Two-Dimensional Diffusion Map Clustering

For use as a clustering algorithm within the Cleansing Algorithm, we propose the use of the diffusion map representation in combination with k-means in the two-dimensional setting. In the work of E. S. Roozemond [43], this application of diffusion map clustering is visualized on a data set consisting of two circles with different radii, where k-means on its own fails to discover the two distinct circles within the data set. The k-means algorithm used on the diffusion map representation of the data set, however, is able to capture this distinct structures in the data.
This application of diffusion maps opens the door to the handling of many more complex structures within our Aegon data sets. It provides us with a tool to be able to cluster (non-)linear structures in the two-dimensional setting such that we may recognize distinct clusters that should be handled by separate models when restoring their missing values. An example of a two-dimensional data set that may occur is provided in figure 4.2, in which we use k-means on the diffusion map of this data. The colors indicate to which cluster data points belong.

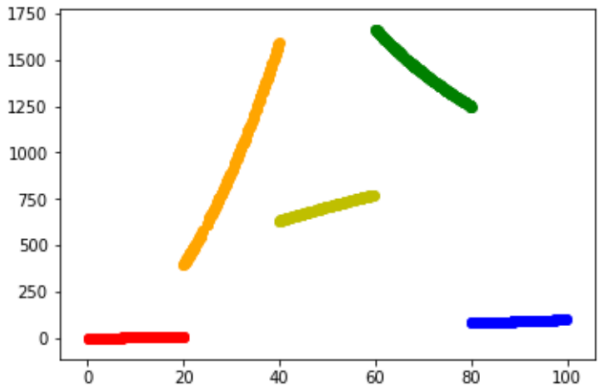


Figure 4.2: Example of two-dimensional data set with 5 unique clusters.

However, the main application for the use of diffusion maps is to be able to recognize which features give rise to the semi-numeric nature of output features containing correctly missing data. Contrary to the one-dimensional k-means approach, the use of k-means on the two-dimensional diffusion map representation of the an input feature and the output feature, allows us to immediately recognize which feature is responsible for the dichotomy within the output feature.

4.2.2.3 Diffusion Map Clustering in the Cleansing Algorithm

Having discussed the clustering strategy related to diffusion maps, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical features, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the

output feature.

Once the output feature is recognized to be a numeric feature by the Data Transformation step, it can be used for clustering by means of diffusion map clustering.

In the Cleansing Algorithm we thus iteratively analyze the geometric structures between the different input features and the output feature. For each input and output feature pair we construct the diffusion map on the two-dimensional data in accordance with the algorithm proposed by E. S. Roozemond. Once a diffusion map has been constructed, we apply the k-means algorithm on this new representation. This is done ten distinct times for values of parameter $k \in \{2, \ldots, 20\}$. For each run the silhouette score is determined based on the diffusion map representation of the data. The k-means clustering with the value of parameter k that acquires the greatest mean silhouette score over their ten runs is deemed as the best choice for k.

As we are iteratively checking for distinct clusters in each of the input features with the output feature, analysis of all features may lead to the recognition of many different sub-clusters of the data. To prevent that our data is clustered in too many very small clusters, we only consider those clusterings that lead to clusters with a greater size than a thousand data points or that consist of at least 5% of data. If the diffusion map clustering has then led to multiple clusterings based on different features, we sort these clusterings based on their silhouette scores in descending order. We start with the clustering that has the greatest silhouette score and merge it with the clustering with the second greatest silhouette score. This is done by creating a new labelling for each label pair between the two clusterings. The sizes of the clusters are once again verified to either be greater than a thousand data points or 5% of data. This process is continued until either the cluster sizes no longer satisfy the size conditions or all clusterings are merged.

Once the clustering process is complete, we are left with a number of distinct clusters of data and the features that led to these clusterings. Now if we wish to restore the value of a missing data point within the output feature, we need to pass the missing data point through the k-means models used for each feature, leading to a unique combination of labelings such that the data point can be assigned to a specific cluster of the data set.

The downside of using such an approach to the clustering of the data, is in its time complexity. Having to construct diffusion maps for each input and output feature pair and clustering data based on these pairs, leads to much greater time complexity than just performing k-means. Though this impact can be minimized whenever multiple features occur within the data quality check of the data set. It is highly likely that features mentioned in the data quality check give rise to the semi-numeric nature of the output feature. As such we also consider to only analyze the diffusion map clusterings of the output feature with respect to other input features mentioned in the data quality check.

4.2.3 Necessity of Clustering

Both the one-dimensional k-means and the two-dimensional diffusion map clustering fail to handle a highly important insight. Namely, both algorithms are unable to determine whether or not clustering is even necessary. The silhouette score cannot handle the case where only one cluster exists and thus this metric is insufficient to determine when it is necessary to cluster. Though it is possible to recognize some semi-numeric features through analysis of the number and frequency of unique value occurrences within the output feature, it is not universally possible. Some semi-numeric feature can contain upwards of 95% of data to be correctly missing and some below 10%. An alternative can be to keep track of the imputation of zeros and negative ones in the output feature during the Data Transformation step of the Cleansing Algorithm. If for example we recognize that a numeric feature contains no zero values originally, we would impute correctly missing data with the value zero. As such the occurrence of the value zero indicates us that there is a correctly missing data occurrence within the feature, if not we need not consider clustering. The same process can be followed for the case where imputation occurs with negative one.

This methodology, even though it deals well with the main problem of the semi-numeric output

features, reduces the possible impact clustering can have on the restoration of missing data significantly. We wish to find a methodology with which we can automatically recognize whether we are dealing with semi-numeric features or numeric features with distinct clusters. One concept that comes to mind, is to analyze the density over the space spanned by the output feature. If the output feature contains correctly missing data, then on either the zero or the negative one value we expect to encounter a high density of points relative to the interval around them. The same can be considered for two or more distinct clusters occurring within the output feature. We expect on two or more distinct interval in the output feature, to encounter higher density areas surrounded by lower density intervals.

4.2.3.1 Density-Based Spatial Clustering of Application with Noise

The Density-Based Spatial Clustering of Application with Noise (or DBSCAN for short) is a data clustering algorithm proposed by M. Ester et al. [24], that is based on this principle. The DBSCAN algorithm, unlike the k-means algorithm that assumes clusters to be of convex shape, can find clusters of any shape. This is due to the fact that the DBSCAN process determines clusters to be those areas of high density that are surrounded by areas of low density. At the basis of DBSCAN are so-called core points. These core points are those data points that have at least some number of other data points within a certain distance to it. The core points thus exist in those areas with high density. Apart from core points, we also recognize those points that lay close to a core point, but are themselves not core points. A cluster can thus be recognized by DBSCAN as those core points that are close to each other by some measure of distance as well as the non-core points that are close to some core point within the cluster. The DBSCAN process can also handle noise in that data points that exist in low density areas that are thus not close to any core points are labelled as noise.

4.2.3.2 Density-Based Spatial Clustering of Application with Noise Algorithm

The DBSCAN algorithm as proposed by by M. Ester et al. [24], starts by determining the minimum number of points $MinPts$ and the distance in which this minimum number of points should be encountered ϵ. Using this we can provide definitions for core points, non-core or boundary points and noise or outliers.

- A point x_i is considered to be a *core point* if at least $MinPts - 1$ other data points are within distance ϵ of it.
- A point x_j is considered to be a *boundary point* if it is reachable from another point and not a core point.
- A point x_k is considered to be *noise* if it is not reachable from any other point.

In order to define what it means to be reachable, we first provide the definition of directly reachable:

- A point x_j is *directly reachable* from x_i if x_j is within a distance of ϵ from a core point x_i.
- A point x_j is *reachable* from x_i if there exists a path $x_i, x_{i+1}, \ldots, x_{i+n}$, where $x_{i+n} = x_j$, such that each x_{l+1} is directly reachable from x_l.

Using these definitions, the algorithm starts by selecting some arbitrary point x_i from the data set X. From this data point, all points that are reachable from x_i are retrieved based on $MinPts$ and ϵ. Then if x_i is a core point, due to there being at least $MinPts$ other data points within distance of ϵ from it, the set of points retrieved is considered a cluster and the point x_i is assigned to said cluster. If x_i is a boundary point or noise, no points are reachable from x_i and DBSCAN continues to the next unlabeled data point. At the end of this process, once all data points have been handled, all unlabeled data is considered to be noise and all labelled data consists of clusters that themselves contain the core and boundary points.

4.2.3.3 DBSCAN Clustering

For use in the Cleansing Algorithm, DBSCAN has an invaluable property. DBSCAN does not need to have an appropriately chose parameter indicating the number of clusters within the data. In light of this, we recognize to two derivations that can be used to supplement the clustering processes of output and diffusion map clustering. The first derivation concerns itself with the fact that DBSCAN is able to handle data in which there does not exist a clustering. Where we needed to specify that there were at least $k = 2$ clusters for the other methodologies to be able to use silhouette scores and k-means itself, DBSCAN can be used in both the output and diffusion map clustering settings to check if clustering is necessary. That is, for output clustering, if we run the DBSCAN algorithm on the one-dimensional output feature and that there exist no distinct clusters within the output feature, then the DBSCAN process shall recognize this and return only labellings for one cluster and possibly some noise. Thus as DBSCAN teaches us that there only exists one cluster in the data, we do not opt to use the output clustering on this output feature. Whereas if the DBSCAN algorithm does recognize at least two subsets of the data to be distinct clusters, we can use output clustering on this feature. For diffusion map clustering a similar process can be followed. For each input and output feature pair, we first perform DBSCAN and use it to determine if there exists one or more than one cluster. If there is no more than one distinct cluster, we can skip clustering this pair and move to the next input feature to consider. If the DBSCAN algorithm finds more than one cluster, the diffusion map clustering is performed. Based on this derivation of the DBSCAN algorithm, we perform a type of filtering to determine if clustering needs to take place or not.

The second derivation draws inspiration from the work of O. Limwattanapibool and S. Arch-Int [23]. In their work they note that the k-means algorithm can be highly accurate whenever the appropriate number of clusters k is chosen prior. An inappropriately chosen number of clusters or initial centers decreases the accuracy of k-means greatly. In light of this, they propose to supplement the k-means clustering algorithm through use of insights that can be gathered from performing DBSCAN on the data. As discussed in the first derivations concerning the determination of when or not to perform clustering, DBSCAN already provides us with an indication of the number of clusters we can expect to encounter in the feature. This number can be used as the parameter of k in k-means. This then leaves use with the choice of initialization of the centroids for k-means. If we use the labellings as provided by DBSCAN and its recognition of noise, we choose as centroids a point within each recognized cluster thus avoiding the choice of noise as centroid.

One important thing to note when making use of the DBSCAN algorithm is the fact that the duplicate points within the one-dimensional output feature arising due to the presence of correctly missing data, will be recognized as noise rather than a distinct cluster. Thus it is of the utmost importance for the use of DBSCAN that we analyze the size of the set of data points that are recognized as noise and if this is greater than 5% of data or a thousand data points, we need to consider the noise as a cluster as well to be determined by k-means. This notion forms the basis as to why DBSCAN cannot be directly applied for clustering within output or diffusion map clustering. DBSCAN, apart from the correctly missing data, also adds data points considered to be noise with respect to the rest of the data. Thus we cannot assume that if the size of noise is greater than the threshold, it could be an appropriate cluster for restoration. Thus we only use DBSCAN to verify whether clustering is appropriate or not.

4.2.3.4 DBSCAN Clustering in the Cleansing Algorithm

Having discussed the clustering strategy related to DBSCAN, we discuss how this process ties into the global structure of the Cleansing Algorithm.

In the Data Transformation section 3, we discussed how the raw Aegon data is transformed. In that section, we ended with a data set comprised of features of categorical and numeric nature as predefined by Aegon. For categorical features, we performed one-hot encoding and numeric features were normalized. These features were the so-called input features, whereas we provided a separate feature that was the output feature.

Once the output feature is recognized to be a numeric feature by the Data Transformation step, it can be used for clustering by means of DBSCAN.

The DBSCAN algorithm assigns data points to clusters or to noise. Using these labellings we recognize whether it is needed to perform clustering. This is either indicated by the presence of correctly missing data, where DBSCAN assigns many data points to be considered noise, or whenever DBSCAN returns labellings for more than one cluster. In the case of correctly missing data, we determine the parameter for the number of clusters in k-means to be equal to the number of unique labellings found by DBSCAN plus one for the correctly missing data. In the case there is no correctly missing data, this parameter can be set to the number of unique labels encountered in the output of DBSCAN. Furthermore, we set the centroids for k-means to be randomly selected from the clusters recognized by the DBSCAN algorithm. For the diffusion map clustering DBSCAN is performed on the input and output feature pairs in the diffusion map representation to determine the appropriate value of k for k-means and the selection of centroids.

Using DBSCAN as a filter for clustering, we are able to either have unclustered data enter into the Model Selection node of the Cleansing Algorithm, or have clusters of data enter when deemed necessary.

4.3 Structure Analysis on Aegon Data

In section 3.3, we were left with a training data set consisting of transformed features, in table 12, and a test and validation set, as in table 15, that was also transformed in accordance with this training data set. The Data Structure Analysis node makes use of just the training data set to determine underlying structures and perform feature selection.

If we now return to the data quality check from section 3.3 and more specifically the case where our feature *Residual* is strictly greater than that of the difference between the $Total$ and $Used$ features, we notice this occurs within the provided sample in four cases in Table 7. Each time the feature *Active* indicates that a buffer is being used with value Y, a value greater than zero is stored in the feature $Buffer$. For these data points one may notice that, rather than the residual being equal to the difference between the total loan and used part of the loan, the residual is equal to the difference between the sum of the total loan with the buffer value, and the used part of the loan. In other words, the data quality check could alternatively be summarized as:

$$\text{If Active} = \text{N and Residual} = \text{Total} - \text{Used, then DQ} = 1$$

$$\text{If Active} = \text{Y and Residual} = \text{Total} - \text{Used} + \text{Buffer, then DQ} = 1$$

$$\text{Else, then DQ} = 0$$

This provides us with a clear overview of the relationships within the training data that we will expect for this data set. However, before we start the structure analysis process, we shall analyze the relative frequencies of each of the data quality check cases. The first case, in which the residual is equal to the difference between the total and used, has an occurrence rate of approximately 85% on the entire data set. The second case, in which the residual is equal to the difference between the total and used plus the buffer value, has an occurrence rate of approximately 10%. Finally, the actual occurrence of missing data is subject to 5% occurrence rate. With this information, we shall continue the process done in 3.3 and perform clustering and feature selection on the transformed training data. We shall start by performing output clustering, after which we perform diffusion map clustering. Once the clustering step is completed, we shall move towards using the three feature selection methodologies.

4.3.1 Clustering

4.3.1.1 Residual

Before we perform clustering on a target feature, first the target feature must be passed through the DBSCAN algorithm. As can be observed from the training set sample, there is no need for clustering on the *Residual* feature. However, we do not know this prior and it is thus passed through the algorithm. This results in the following DBSCAN output, when used on the training set from the complete data set:

Cluster 1	Cluster 2	Cluster 3
49155	25	23

Table 16: DBSCAN output for Residual feature

Based on this DBSCAN output, following the process as discussed in section 4.2.3.1, we determine that there is no need for clustering as we only recognize the first cluster to be of appropriate size to be considered a cluster.

4.3.1.2 Buffer

In order to show how the clustering section of our Cleansing Algorithm works in the case that clustering is deemed necessary by DBSCAN, we take a small detour to consider an alternative target feature, namely the $Buffer$ feature. Clearly, this $Buffer$ feature contains many correctly missing data values. In the case we wish to restore missing data on this feature we wish to be able to handle the correctly missing and non-missing cases separately, as if we do not distinguish between the two, it shall lead to models that return the imputed value of 0 for 85% of cases that are deemed to restore missing data best.
If we are to pass the $Buffer$ feature through the DBSCAN algorithm, we acquire the following output:

Cluster 1	Cluster 2
4125	45078

Table 17: DBSCAN output for Residual feature

Clearly, this new result indicates the necessity to perform clustering on this feature. Both clusters pass the threshold restrictions to be deemed a cluster. In this setting, considering the $Buffer$ feature as the target feature, we thus move to either using the two-dimensional diffusion map clustering or, alternatively, performing one-dimensional output clustering.

First, we shall handle the case in which we opt to use output clustering. In this case the target feature $Buffer$ is passed on to the k-means algorithm as well as the number of clusters determined by the DBSCAN algorithm, which is in this case 2. Performing k-means on this target feature results in the following cluster sizes:

Cluster 1	Cluster 2
4125	45078

Table 18: K-means output for Residual feature

In this specific setting, in which the correctly missing data are the only duplicate values to occur within the feature, DBSCAN and the k-means clustering generate the same clusters. As such the first cluster forms the cluster containing only the imputed correctly missing values of zero, whereas the second cluster contains the rest of the data. Using the labelling from k-means clustering, which indicates to which of the two clusters a data point is assigned, we confirm this observation by providing the occurrence rate of correctly missing values in both clusters:

Cluster 1	Cluster 2
100.00%	0.00%

Table 19: K-means output for Residual feature

Next, we consider the two-dimensional diffusion map clustering approach. Rather than handle each feature pair separately, we shall run through a case where the feature pair does not lead to clustering

and a feature pair, where this is the case. As an example of the case that no clustering is deemed necessary between a feature pair, we consider the target feature $Buffer$ paired up with the feature $Date$. Intuitively, the value of the feature $Date$ does not determine whether or not the feature $Buffer$ takes on a specific value on its domain. This is further confirmed by our alternative representation of the data quality check for this data set. For this reason, we expect the diffusion map algorithm to deduce that no clustering needs to take place. Performing diffusion map clustering on this feature pair leads to the following cluster sizes:

Cluster 1	Cluster 2
10	99988

Table 20: Diffusion Map output for Residual versus Buffer feature

From these results it is clear that the first cluster does not adhere to the restrictions for it to be deemed a cluster, as it is less than 1000 data points and also less than 5% of the total data. Diffusion Map clustering shall thus not cluster the data based on the relation between the *Residual* and $Buffer$ features.
Following this, we wish to consider the diffusion map clustering between the $Active\_1$ input feature and the $Buffer$ output feature. From the alternate representation of the data quality check, it is clear that if the feature $Active\_1$ returns a value of 1, then the $Buffer$ feature will return a value that is non-missing, otherwise it is missing. Clearly, the feature $Active\_1$ thus contains information regarding the value of the $Buffer$ feature and as such we expect the diffusion map clustering to return two distinct clusters as the output clustering did. Performing diffusion map clustering on this feature pair leads to the following cluster sizes:

Cluster 1	Cluster 2
4125	45078

Table 21: Diffusion Map output for Residual versus Buffer feature

Once more do we thus return to the clustering as found by DBSCAN and output clustering. However, now we know that the feature $Active\_1$ has led to this clustering. In a similar manner performing diffusion map clustering on the $Active\_2$ feature with respect to the $Buffer$ target feature, will lead to the same clustering. The intersect of both of their clusterings thus remains the same. Since the $Active\_1$ feature is one of the one-hot encoded features from the original *Active* feature, we store that the *Active* feature has led to the clustering for the $Buffer$ target feature. Finally, the data is split into two clusters based on the labelling of one of these clustering methodologies and each cluster is handled separately for the following feature selection process.

4.3.2 Feature Selection

In handling to process of feature selection, we shall once more return to the case in which the *Residual* feature forms our target feature. We do note that this process is the same process that would be followed for each distinct cluster as determined by clustering.

4.3.2.1 Correlation Coefficient

The first feature selection approach we will use on our transformed training data, is that of the correlation coefficient feature selection strategy. This process, as stated in section 4.1.1, starts by recognizing that the *Residual* feature is the target feature and the other features form the set of input features. For each feature of the training set, we shall determine its Pearson correlation coefficient with respect to the target feature and corresponding hypothesis test score, which from now on will also be referred to as p-values. These are provided in the following table:

Feature	Coefficient	p-value
Active_1	0.370493	0.000
Active_2	-0.370493	0.000
Used	0.024009	0.018
Buffer	-0.991456	0.000
Contract Age	0.002882	0.777
Total	0.994378	0.000
Date	-0.002568	0.801

Table 22: Pearson correlation coefficients and hypothesis test scores

Following the procedures dictated in 4.1.1, we note that the features $Active\_1$, $Active\_2$, $Buffer$ and $Total$ all show significant linear relation to the target feature due to their p-values being far less than 0.05 and the coefficient value is greater than 0.05. Thus these features are selected based on the Pearson correlation coefficient. The feature $Used$ does show to have a significant linear relationship to the *Residual* feature, however, its correlation is deemed low enough that this linear relationship is not deemed to impact the target feature enough to consider it based on its Pearson correlation. For the remaining three features, we move to the Spearman rank correlation coefficient and its corresponding hypothesis. For the remaining features these values are presented in the following table:

Feature	Coefficient	p-value
Used	-0.578518	0.000
Contract Age	-0.007358	0.469
Date	0.007256	0.475

Table 23: Spearman correlation coefficients and hypothesis test scores

Based on the results for these features with respect to the Spearman correlation coefficient, we note that only the $Used$ feature is shown to have a significant monotonic relationship with respect to the target feature and that this relation is greater than the 0.05 boundary. The remaining two features do not show to have a monotonic relationship and are thus passed on to the distance correlation coefficient. These results are provided in the following table:

Feature	Coefficient	p-value
Contract Age	0.044314	0.675
Date	0.044264	0.668

Table 24: Distance correlation coefficients and hypothesis test scores

The results for the distance correlation coefficient and its corresponding hypothesis testing have deemed that the *ContractAge* and *Date* features are independent of the *Residual* target feature and as such are filtered out.
The final step in the correlation coefficient filtering approach, concerns the removal of redundancy through the determination of the value of the Pearson correlation coefficient between the five selected features. The Pearson correlation coefficients are provided below:

Feature 1	Feature 2	Coefficient	p-value
Active_1	Active_2	-1.000000	0.442
Active_1	Used	0.007824	0.000
Active_1	Buffer	-0.370003	0.000
Active_1	Total	-0.000263	0.000
Active_2	Used	-0.007824	0.979
Active_2	Buffer	0.370003	0.441
Active_2	Total	0.000263	0.000
Used	Buffer	-0.107903	0.979
Used	Total	-0.007685	0.449
Buffer	Total	-0.004393	0.666

Table 25: Pearson correlation between features selected for target feature.

As indicated in section 4.1.1, if we encounter a Pearson correlation coefficient between these features greater than 0.99 then we deem there to be redundancy and remove one of the features. Based on the results from table 25, we thus conclude that there is redundancy in the features *Active_1* and *Active_2*. This is a logical conclusion for a one-hot encoded categorical feature that takes on one of two values. In this case the *Active_2* feature is removed. Finally, the algorithm stores each feature and the reason why it was selected by the coefficient correlation filter. This result is displayed below:

Feature	Reason
Active_1	Pearson
Buffer	Pearson
Total	Pearson
Used	Spearman

Table 26: Correlation coefficient filter information stored by the Cleansing Algorithm.

4.3.2.2 Treelet

The second feature selection approach we will use on our transformed training data, is that of the treelet feature selection strategy. This process disregards the fact that the *Residual* feature is our target feature until treelet clustering of the features is performed.
The first step taken in the treelet algorithm is to construct the similarity matrix. In the Cleansing Algorithm the similarity matrix uses for the parameter λ the inverse of the maximum covariance value divided by 20. This choice of λ has led to the similarity matrix to first choose those features that have the greatest correlation and if these correlations are equal, then the feature with greater covariance has our preference. For our training set the initial similarity matrix is given by:

$$M^{(0)} = \begin{bmatrix} 1.51 & 0.01 & 0.42 & 0.00 & 0.42 & 0.00 & 0.42 \\ 0.01 & 1.17 & 0.12 & 0.01 & 0.14 & 0.01 & 0.03 \\ 0.42 & 0.12 & 1.04 & 0.00 & 1.03 & 0.00 & 1.03 \\ 0.00 & 0.01 & 0.00 & 1.05 & 0.00 & 1.05 & 0.00 \\ 0.42 & 0.14 & 1.03 & 0.00 & 1.03 & 0.00 & 1.03 \\ 0.00 & 0.01 & 0.00 & 1.05 & 0.00 & 1.05 & 0.00 \\ 0.42 & 0.03 & 1.03 & 0.00 & 1.03 & 0.00 & 1.03 \end{bmatrix} \tag{22}$$

The first step in the treelet algorithm opts to find the feature pair within this matrix that has the greatest similarity score. This is done by searching through the top triangle of this matrix, resulting in the feature pair that corresponds to the fourth row and sixth column of the matrix being selected. In our training data, this similarity score corresponds to the features *ContractAge* and *Date*. The relationship between these two features, as discussed in section 3.3, is that the *ContractAge* feature indicates the length in years the contract has been active, whereas the *Date* feature indicates the starting date of the contract. As such the relation between the two is that if the number of years the contract is active for is added to the date, then the year should be 2022.

Once the greatest similarity score is found, it is first checked if this similarity score is greater than our stopping criterion. For this example, the stopping criterion has been set to 0.1. As such the similarity score between the two features is sufficient, thus we cluster these two features together and we move to step 2.

The second step in the treelet algorithm opts to remove the correlation between these two features by means of a Jacobi rotation. Initially, the correlation between the two features is determined to be 1.05. By means of binary search the rotation factor θ is determined with starting points for the binary search being $\frac{-\pi}{4}$ and $\frac{\pi}{4}$. Decorrelation of these two features was determined by the θ value of 0.784074. Using this θ we can update the feature such that $x_{new} = J^T x_{old}$, where J^T denotes the transpose of the Jacobi rotation matrix. The new similarity matrix is determined once more using the x_{new}, this results in the following matrix:

$$M^{(1)} = \begin{bmatrix} 1.51 & 0.01 & 0.42 & 0.00 & 0.42 & 0.00 & 0.42 \\ 0.01 & 1.17 & 0.12 & 0.00 & 0.14 & 0.01 & 0.03 \\ 0.42 & 0.12 & 1.04 & 0.01 & 1.03 & 0.00 & 1.03 \\ 0.00 & 0.00 & 0.01 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.42 & 0.14 & 1.03 & 0.00 & 1.03 & 0.00 & 1.03 \\ 0.00 & 0.01 & 0.00 & 0.00 & 0.00 & 1.10 & 0.00 \\ 0.42 & 0.03 & 1.03 & 0.00 & 1.03 & 0.00 & 1.03 \end{bmatrix} \tag{23}$$

From this matrix the new greatest similarity score is given by the feature pair $Buffer$ and $Used$, given by the similarity score 1.03.
This process is continued until the similarity score no longer is greater than the value of the stopping criterion. This then results in the following hierarchical tree for the Aegon training set:

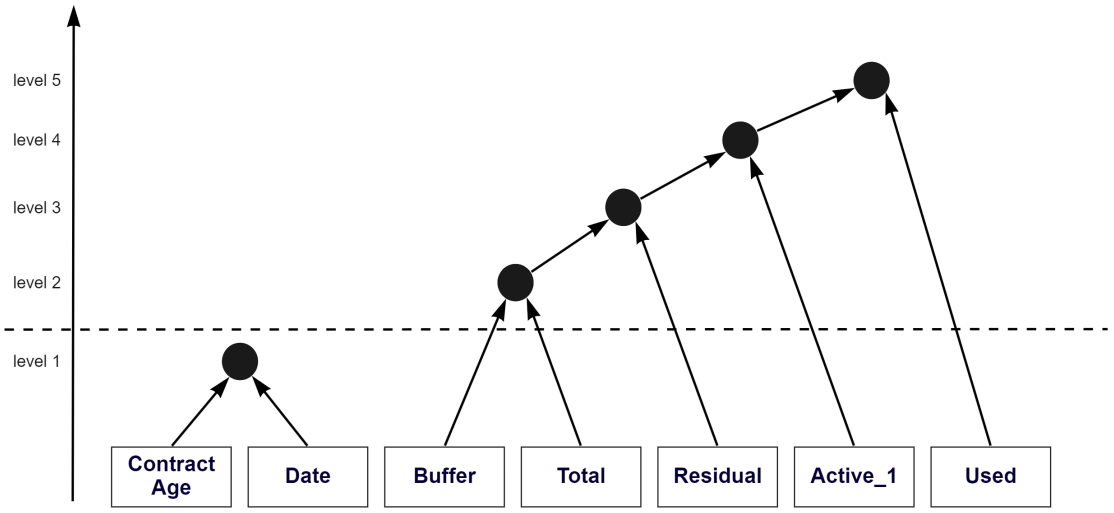


Figure 4.3: Dendrogram of treelet clustering process for Aegon training set.

As can be observed in figure 4.3, at the first level the $ContractAge$ and $Date$ features are clustered together. In light of the fact that the relation between the two is a direct linear one, the fact that these two features are clustered first is a correct one. At the second level the $Buffer$ and $Total$ features are clustered together in a new cluster. This occurring due to the fact that the $Buffer$ and $Total$ feature are linearly related and due to the fact that the contracts allow for larger buffers if the total value is greater, leading to a greater similarity score. At the third level, unlike the two levels prior, no new cluster is formed. Rather, the *Residual* feature is added to the cluster containing the $Buffer$ and $Total$ features. Once the $Buffer$ and $Total$ features are decorrelated, the *Residual* feature is shown to have the greatest similarity score with the decorrelated $Total$ feature. At the fourth level, the $Active\_1$ feature is added to the same cluster as *Residual* in the third level. This is due to the fact that $Active\_1$ has the new greatest similarity score with respect the $Buffer$ feature. Finally, the $Used$ feature completes the Treelet

clustering process, by being added to the cluster containing the $Buffer$, $Total$, $Residual$, $Active\_1$ and $Used$ features.

Once the stopping criterion is met, the search for the target feature *Residual* within the clusters and determine that cluster of features to contain all relevant information for the target feature. From the dendrogram provided in figure 4.3, we observe that once the stopping criterion is met, we are left with two clusters. The first of these clusters contains the *ContractAge* and *Date* features. The second cluster contains the $Buffer$, $Total$, $Residual$, $Active\_1$ and $Used$ features. It is this second cluster that contains the *Residual* target feature and thus other features within this cluster are passed to the next node. Finally, the dotted line in figure 4.3 indicates that levels beneath that line are shown to have correlation close to 1. However, since neither *ContractAge* nor *Date* are in the cluster, no redundancy is removed. If both features were within the desired cluster, as with the correlation coefficient filter, one of the two would have been removed. In the case the dotted line has a cluster of more than two features beneath it, only one feature from that cluster will pass through to the next node.

4.3.2.3 Fast Correlation Based Filter

The third and final feature selection approach we perform, is that of the fast correlation based filter. As with the correlation coefficient approach, FCBF starts by recognizing that the target feature is the *Residual* feature. We pre-define a threshold δ to be equal to 0.01. Next the symmetric uncertainty for each input feature with respect to the target feature is determined. These values are provided in the following table:

Feature	SU
Active_1	0.3524
Active_2	0.3524
Used	0.6441
Buffer	0.3191
Contract Age	0.0000
Total	0.6760
Date	0.0000

Table 27: Symmetric uncertainty of features with respect to the target feature.

In the next step, those features are removed, whose symmetric uncertainty is less than the threshold value of δ. As such the features *ContractAge* and *Date* are removed. The remaining features are sorted with respect to their symmetric uncertainty, resulting in:

Feature	SU
Total	0.6760
Used	0.6441
Active_1	0.3524
Active_2	0.3524
Buffer	0.3191

Table 28: Sorted symmetric uncertainty greater than δ of features with respect to the target feature.

The second step in the FCBF filter process then starts with a cluster containing all features from table 28. From here, we start with the feature $Total$. For this feature, we determine the symmetric uncertainty with respect to each other feature within the cluster and if this symmetric uncertainty is greater than the symmetric uncertainty of the other feature with respect to the target feature, then the feature will be removed from the cluster. The symmetric uncertainties of each other feature within the cluster with respect to the $Total$ feature is provided below:

Feature	SU
Used	0.2087
Active_1	0.2979
Active_2	0.2979
Buffer	0.2805

Table 29: Symmetric uncertainty of features with respect to the Total feature.

From the results in tables 28 and 29, we notice that there is no feature that has a symmetric uncertainty in table 29 that is greater than its symmetric uncertainty in table 28. As such none of these features is deemed to contain more information for the $Total$ feature than the *Residual* feature.

The next feature considered is then the $Used$ feature. Following the process of the previous feature, we provide the symmetric uncertainty with respect to this feature for each remaining feature:

Feature	SU
Active_1	0.0000
Active_2	0.0000
Buffer	0.1544

Table 30: Symmetric uncertainty of features with respect to the Used feature.

In these results we notice that once more the symmetric uncertainty with respect to the $Used$ feature in all cases is less than the symmetric uncertainty with respect to the *Residual* feature.

The third feature considered is the *Active*_1 feature. We provide the symmetric uncertainty with respect to this feature for each remaining feature:

Feature	SU
Active_2	1.0206
Buffer	1.9753

Table 31: Symmetric uncertainty of features with respect to the Active_1 feature.

In the results as provided in table 31, we notice that both features are shown to exhibit a greater symmetric uncertainty with respect to the *Active*_1 feature than with respect to the target feature. For this reason both features are considered to be redundant and thus shall be removed from the cluster. Once both the *Active*_2 and $Buffer$ features have been removed, the iterative process stops as no more features remain to consider in the cluster. The fast correlation based filter then leaves us with the $Used$, *Active*_1 and $Total$ features for the next node.

5 Model Selection Node

The restoration of missing data concerns itself with trying to impute missing values based on existing data. In this sense, we are aiming to determine the underlying mechanisms and processes that could have produced such data if it were not missing. However, the underlying mechanisms and processes can take on a great number of forms. Over the years many unique models have been proposed to identify and reproduce certain types of mechanisms or processes. These brings up the concern of how we can best determine these processes that have produced the data. By comparing the performance of these models according to some metric, we may be able to at least select the model from a set of candidate models that has the best desired performance. This is exactly what model selection concerns itself with.
As we are concerned with restoration of missing data of Aegon data sets, we wish to find the model that best is able to restore missing data. That is, if we are to compare a candidate set of models, we shall do so by considering the restoration rate as defined in 1. Those models that reproduce the greatest restoration rate are the models we expect to best capture the underlying mechanisms and processes with respect to the output feature. In order for the Cleansing Algorithm, as a tool for the restoration of missing data, to be able to capture a broad range of mechanisms, we need to also consider a broad range of candidate models. Though construction and comparison of a large number of candidate models is a tedious and time consuming process. That is why we propose a novel approach to model selection to reduce time complexity.

5.1 Models

Before we are able to discuss the model selection process itself, we shall provide an overview of candidate models that will be considered by the Cleansing Algorithm. All models discussed in this section are implemented using the Sklearn package from Python [45].

5.1.1 Linear Regression

The first model considered for the restoration of numeric features is that of linear regression [11]. Linear regression aims to find a linear relationship between the features of our data and the target feature that is considered to be missing. Similar to the other regressors we shall consider, the aim is to determine the conditional probability distribution of the target variable given all other features. In this setting this is done through the constraint that the conditional probability distribution of the target variable is restricted to multiple linear regression. It is then that we wish to use such a model in the case we expect the underlying data relation to be linear in the sense that $y = f(X)$ for some linear function $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$, in example $f(x_1, x_2, \ldots, x_n) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$, for some $\beta = (\beta_1, \beta_2, \ldots, \beta_p) \in \mathbb{R}^p$.

We shall approximate the linear regression model to the target variable through the method of least-squares estimation. Under this methodology we aim to minimize the mean-squared error of the residuals when comparing the observed dependent variable y to the linear approximation based on the independent variable $\mathbf{x}$. This process leads to a closed form solution for coefficients β_i such that:

$$\beta = (X^T X)^{-1} X^T y \tag{24}$$

By the Gauss-Markov theorem, the β determined above is proven to indeed be the local minimum.
Using this closed form solution we are now able to determine the best linear predictor for the restoration of our missing data.

5.1.2 Polynomial Regression

With the aim to tackle the seemingly restricting assumption of linearity in the linear regression model, we consider the polynomial regression model [12]. As we know, many relations that occur within data are not linear. Polynomial regression aims to fit a polynomial of input features to the target variable rather than a linear function, yielding more flexibility in finding the underlying relation.

This model can be seen as an extension to the aforementioned linear regression model. An example function for fitting a polynomial regression model from one input feature is given:

$$y_i = \sum_{i_1=0}^{d} \sum_{i_2=0}^{d-i_1} \sum_{i_3=0}^{d-i_1-i_2} \cdots \sum_{i_p=0}^{d-\sum_{k=1}^{p-1} i_k} \beta_{i_1 i_2 \cdots i_p} x_1^{i_1} x_2^{i_2} \cdots x_p^{i_p} \tag{25}$$

Conversion of this polynomial model to matrix form as with linear regression yields us the same closed form optimal solution:

$$\beta = (X^T X)^{-1} X^T y \tag{26}$$

However, now the dimensionality of β and X has increased. For example, using a degree of four and number of features equal to ten, results in $\beta \in \mathbb{R}^{1001}$ and $X \in \mathbb{R}^{n \times 1001}$. This illustrates the major drawback to using polynomial regression. On the one hand, polynomial regression allows us to approximate a wide variety of relations between our target feature and input features. On the other hand, by increasing the input feature dimensionality, we require larger data sets and greater computational power. It is also of note that when an input feature x_{ij} takes on values in $\mathbb{R}^+$, then x_{ij}^2 is highly correlated to x_{ij} with correlation greater than 0.99. Though this is not collinearity following a formal definition of the word, it can lead to the same adverse effects of collinearity in the literal sense.

5.1.3 ElasticNet Regression

Elastic Net is an example of a regularized regression method that combines the LASSO and Ridge methods [13]. The main advantage of ridge regression is that it deals with multicollinearity as was one of the drawbacks of using ordinary least squares in the linear regression setting and the advantage of LASSO regression is that it is capable of performing feature selection by allowing coefficients in the aforementioned linear models to become zero-valued. It aims to find a compromise between the two methods. Elastic Net can behave similarly to LASSO regression, while tackling the degenerate and sporadic behavior that occurs in the case of high correlations. Alternatively, Elastic Net can behave like ridge regression, while allowing some highly correlated variables to become zero and keeping a subset of the others. The loss function for the Elastic Net is given by:

$$||X\beta - y||_2^2 + \lambda_2 \alpha ||\beta||_2^2 + \lambda_1 (1-\alpha)||\beta||_1 \tag{27}$$

The choice of the α parameter determines the behavior of the Elastic Net. When $\alpha = 1$ the Elastic Net reduces to the ridge regularization case, where all $\beta_j \neq 0$. As the value of α decreases from 1, the sparsity of β increases down to $\alpha = 0$ where the solution reduces to that of the LASSO regularization and thus maximum sparsity. To solve for any given $\alpha \in (0, 1)$, we have to combine ordinary least squares and coordinate descent. We start by acquiring the original least squares estimate:

$$\tilde{\beta} = (X^T X)^{-1} X^T Y \tag{28}$$

From here we perform coordinate descent as in the LASSO regularization with a modification to our coordinate update:

$$\tilde{\beta_j} = \frac{S\left(\frac{1}{n}\sum_{i=1}^{n} x_{ij}\left(y_i - \tilde{y_i}^{(j)}\right), \lambda(1-\alpha)\right)}{1 + \lambda\alpha} \tag{29}$$

Where,

- $S(u, v) = \begin{cases} u - v \text{ if } u > 0 \text{ and } v < |u| \\ u + v \text{ if } u < 0 \text{ and } v < |u| \\ 0 \text{ if } v \geq |u| \end{cases}$

In essence, for Elastic Net regression, we start with the ordinary least squares solution, then we perform coordinate descent similarly to LASSO regularization, after which the shrinkage of the ridge regularization is performed.

5.1.4 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a class of fully connected feedforward neural networks [14]. Multi-Layer Perceptrons aim to determine some continuous (non-)linear function between the provided input and the output. To do so, MLPs consist of three different types of layers: input, hidden and output layers. An example of a MLP with one hidden layer is provided in figure 5.1

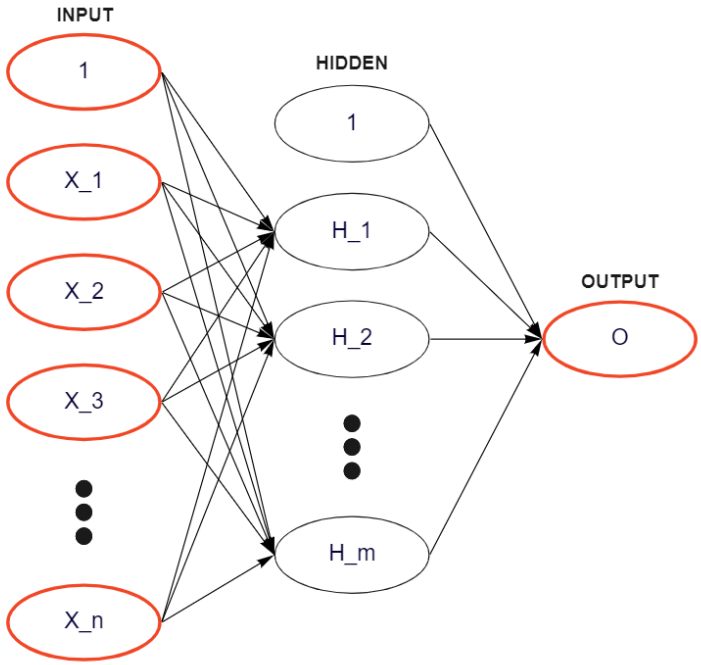


Figure 5.1: Multi-Layer Perceptron consisting of one layer.

The circles left, indicated in red, form the nodes that make up the input layer. This layer has the same dimensionality as the input and each node as such represents one dimension of the input. The second set of nodes forms the hidden layer. Each arrow connects each input node to a node in the hidden layer. This arrow represents some transformation of the value that enters the input node such that a value $H_1 = w^T X + b$ is some linear combination of all input values multiplied by a scalar and a bias term is added. Within the hidden layer a non-linear transformation occurs through a so-called activation function. The most common choices for the activation function are the logistic sigmoid function, the hyperbolic tangent and a rectified linear unit function. Then nodes in the hidden layer are connected through arrows to the final red node, representing the output layer. The arrows connecting each node to the node of the output layer once more represent a linear combination of hidden layer outputs as input for the output layer.
Knowing the architecture of a multi-layer perceptron, we are left with the task of determining all parameters within the MLP. This is done through a process called Adam. Adam is similar to stochastic gradient descent in a sense that it is a stochastic optimizer, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments [15].

One difference between classification and regression using the multi-layer perceptron is defined by the choice of activation function for the output layer. For classification problems concerning binary classification, the logistic function is used, whereas multi-class problems are handled through a softmax function. For regression problems the activation function is the identity function.
The other main difference between classification and regression for MLPs, is in their loss function for training. The loss function for classification is the average cross entropy, which is given by:

$$L = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k}\log(p_{i,k}) + \frac{\alpha}{2N}||W||_2^2 \tag{30}$$

Where,

- N is the number of data points in the input.
- K is the number of distinct classes in the output.
- $y_{i,k} \in \{0, 1\}$ is a binary number indicating whether data point i belongs to class k.
- $p_{i,k} \in [0, 1]$ is the probability estimate of data point i belonging to class k.
- W are the weights.
- α is a parameter that determines the magnitude of the penalty given to the weights by the L_2-regularization.

On the other hand, the loss function used for regression in Adam is the mean square error, given by:

$$L = \frac{1}{2N}\sum_{i=0}^{N}||\hat{y}_i - y_i||_2^2 + \frac{\alpha}{2N}||W||_2^2 \tag{31}$$

Where,

- N is the number of data points in the input.
- $y_i \in \mathbb{R}$ is the true value of the output with index i.
- $\hat{y}_i \in \mathbb{R}$ is the MLP approximation value of the output with index i.
- W are the weights.
- α is a parameter that determines the magnitude of the penalty given to the weights by the L_2-regularization.

.

5.1.5 Random Forests

Random forests is an ensemble learning method for classification and regression of which the operations concern the construction of a number of decision trees [16], [20]. Each decision tree that is part of the random forests ensemble, is constructed from random samples drawn from the complete data set. The construction of these decision trees for classification starts by determining the proportion of each class or category within the data set. Then a measure of impurity is calculated. For our purposes we use the Gini impurity. Then a number of candidate splits are proposed such that for each feature there is some threshold that splits the data set. Of all these candidate splits consisting of a feature and a threshold within the feature, the split is chosen to be the one that minimizes the impurity (or weighted Gini impurity) of the subsets of the data set that the split will construct. This process is then repeated for each subset of the data until some threshold is reached such as the maximum allowable depth of the decision tree, a minimum number of samples, or whenever there is only one data point left after a split. Random forests then construct many decision trees on different samples of the original data and combines the results that come from constructing the decision trees. For classification this is done through the method of majority vote, in example if five decision trees determine the data point to have class 1 and six decision trees deem it to have class 2, class 2 is assigned. For regression this is done through averaging the numeric outputs of the various decision trees. A schematic overview of the random forest process is provided in figure 5.2:

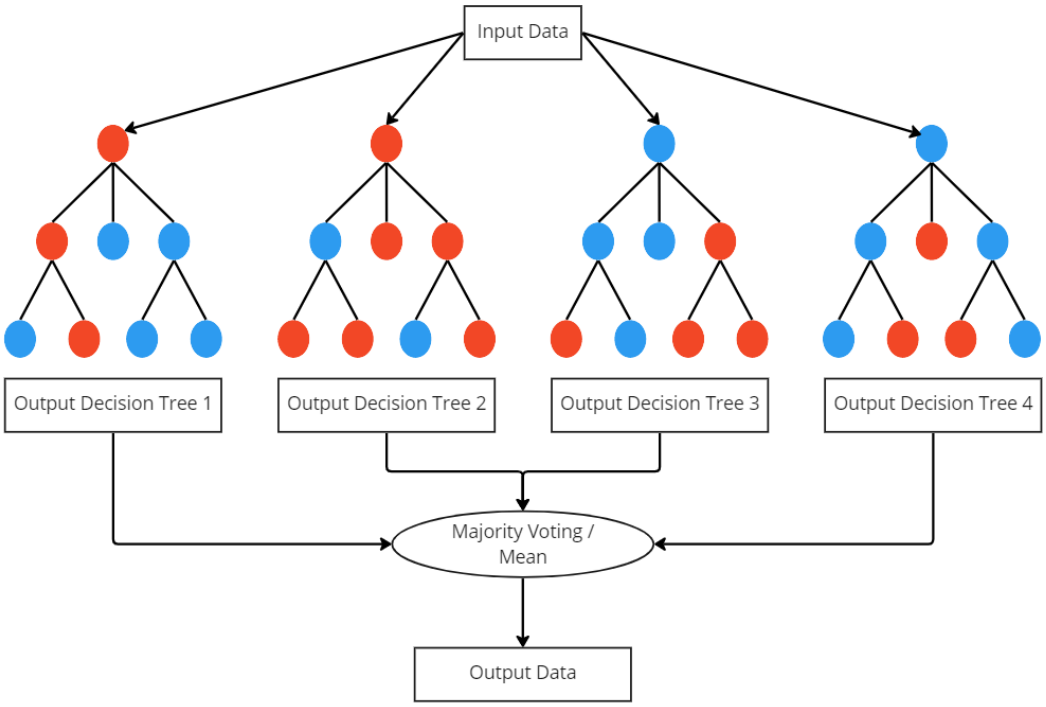


Figure 5.2: Schematic representation of Random Forests.

5.1.6 Nearest Neighbors

The nearest neighbor algorithm (KNN) is a non-parametric supervised learning method that can be used both for regression and classification [17]. The idea behind the nearest neighbor algorithm is to select k data points from a training dataset that are closest with respect to some distance measure $d(\cdot)$. Based on these k nearest data points we expect the value of our target feature to resemble the existing value of the target feature in the training data points. In this way KNN distinguishes itself from the parametric supervised learning models, in that it need only store data points to perform classification and regression, rather than generate and store a model.
That is, given some set of training data with a target feature, the KNN algorithm first determines the k nearest neighbors through either brute force, ball tree or K-D tree dependent on the sparsity, dimensionality and number of neighbors. These neighbors are only determined based on the input features without the target feature. Once the k nearest neighbors have been found we distinguish between classification and regression. For classification problems, majority vote decides to which class in the target feature the data point belongs. This voting can be done uniformly, where each nearest neighbors vote has equal weight to the determination of the class. Alternatively, this can be done based on some other metric such as a distance-based approach that gives greater voting power to closer neighbors. For regression problems, the uniform average of the k nearest neighbors determines the value of the data point. If one wishes to use a distance-based approach, the value of the data point is given by the inverse distance weighted average of the values of the k nearest neighbors for the target feature.

The number of nearest neighbors has to be chosen carefully as it can greatly impact the performance of the algorithm. To briefly illustrate, we provide an example of a KNN classification problem in figure 5.3.

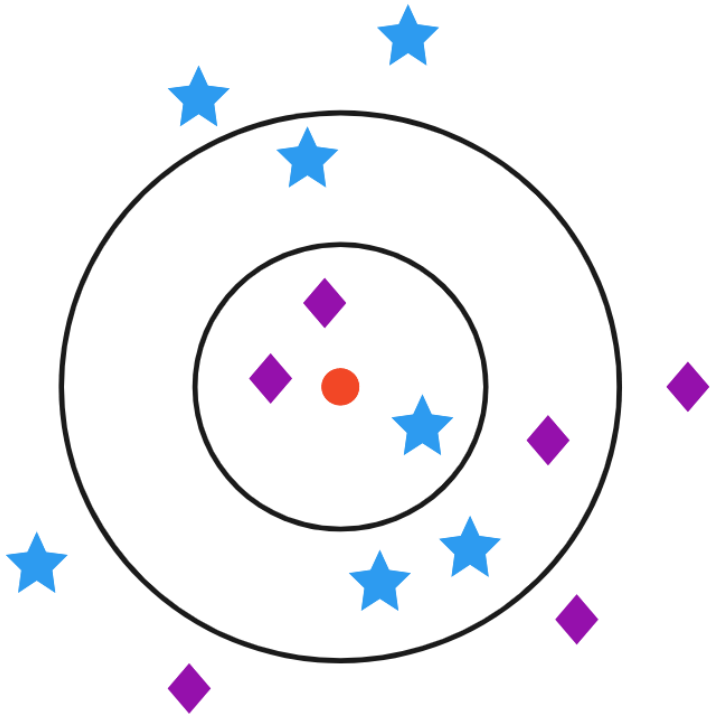


Figure 5.3: Example of K-NN classification task.

The KNN algorithm needs to determine to which class the red circle belongs, either a blue star or purple diamond. If we are to use the three nearest neighbors based on Euclidean distance and uniform weights, the algorithm will return a purple diamond, whereas selecting the five nearest neighbors in the same setting will return a blue star. This illustrates the need for hyperparameter tuning for selecting k such that it most adequately represents the data set.

5.1.7 K-Means

The K-means algorithm is a clustering algorithm and thus its usage in classification is unsupervised. K-means aims to split samples into K groups of equal variance. It does so by minimizing the within-cluster sum-of-squares. By using the number of distinct classes found, a training data fed to the algorithm, we can aim to create K distinct clusters such that each cluster represents some class within the training data. When wishing to determine to which class a new data point belongs, we determine the center of which cluster the new data point is closest to and the data point is assigned the class belonging to that center.
The k-means algorithm consists of three steps:
The first step is the initialization of k means m_i, by sampling k random data points within the training data.
The second step is to assign each other data point within the training set to the class of which the m_i is closest based on some distance measure. This then results in k distinct sets of data S_i partitioning the training set: $S = \{S_1, \ldots, S_k\}$. The third step is to update the means m_i to be the means of the sets S_i:

$$m_i = \arg\min_S \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{x_i, x_j \in S_i} ||x_i - x_j||^2 \tag{32}$$

Afterwards, the second and third steps are repeated until the means m_i no longer change.

5.1.8 Expectation-Maximization

The Expectation-Maximization algorithm is a method used to determine the maximum likelihood of parameters in statistical models in the case the models depend on unobserved latent variables [18]. For use as a classification model we shall consider the Gaussian Mixture model in combination with the Expectation-Maximization algorithm.

The Gaussian Mixture Model, or GMM for short, is a mixture model that uses a combination of Gaussian (Normal) probability distributions and requires the estimation of the mean and standard deviation parameters for each.

The estimation of these parameters can be performed by means of maximum likelihood estimates, where the Expectation-Maximization algorithm comes in.

In the EM algorithm, the estimation-step would estimate a value for the process latent variable for each data point, and the Maximization step would optimize the parameters of the probability distributions in an attempt to best capture the density of the data. The process is repeated until a good set of latent values and a maximum likelihood is achieved that fits the data.

5.1.9 Need for Model Selection

As is clear from the candidate models discussed, we have access to eleven different models of which six can be used for regression purposes to restore numeric features and five models can be used for classification to restore categorical features. Though one might presume that this is not that many models to have to construct for restoration of a missing feature, we point our attention to the hyperparameters within the different models. For example, polynomial regression may be the general model, but in order to appropriately use polynomial regression on our data, we need to decide for which degree of polynomial the regression shows to have the best restoration rate. For all models considered, we choose the best hyperparameters by means of grid search such that all different combinations of hyperparameter configurations of each model need to be constructed and compared. If we thus consider our six models and at least 10 unique hyperparameter configurations on average, we need to train and compare sixty different models. As such we have deemed it necessary to devise a model selection methodology that provides us with the tools necessary to minimize the number of different models we need to construct in order to determine the best model for restoration.

5.2 Model Selection Strategy

In the previous section, we discussed the various different models that can be used for the restoration of missing data. Though the algorithm has access to all these different models and the advantages and disadvantages of the models are known to us, much is unknown with regards to the data that the algorithm will use. For this reason it is not trivial to decide which model to use for restoration in an arbitrary data scenario. However, using every model in every scenario is computationally complex and takes up an extraordinary amount of time, especially considering many models considered will have a great number of sub-variants to choose from based on different selections of hyperparameters in these models.

The process by which the best model for each task of restoration is chosen, is called model selection. One method of model selection splits the input data into a training, validation and test subset. The most common ratio between these subsets of the data set is $70 - 15 - 15$, resulting in 70% of data being used for training the various models, 30% of data being used for validation and test sets. The 15% of data for validation is used to compare the different models and hyperparameter variations of those models by comparing the different model scores to each other. Model scores are determined by comparing output provided by the model based on validation set input to the actual output of the validation set. The test set is used to provide an unbiased estimate of the actual model score for the best selected model variant and is determined in the same manner as the validation set.

Though theoretically sound, the model selection strategy as discussed is computationally and time complex. As such we aim to challenge this model selection strategy. By considering some criterium on the input data provided, we wish to determine which model performs best. This would reduce the problem of model selection from training and comparing all models and their sub-variants to determining for which model a certain criterium holds and then reducing training to only consider sub-variants of the model

for which the criterium holds. It is important for this criterium to be relevant for the restoration of missing data. We propose to use data relations of common occurring mathematical relationships between variables as our criterium.

5.2.1 Data Relations

First, we will define the data relations considered for this model selection methodology. It should be noted that it is near-impossible to capture all possible mathematical relations between some arbitrary set of variables. Even if the possibility existed, the time required to run through all these relations would make model selection based on it useless. The data relations as considered for regression and classification aim to represent a set of common mathematical relationships that the algorithm may encounter during the restoration of missing data on Aegon data sets.

5.2.1.1 Regression Relations

For regressors the data relations are given by:

The first relation we consider is that of a simple linear one.
Given some random variable X, we define this relation as $Z = c_1 + c_2X$ for some $c_1, c_2 \in \mathbb{R}$.

The second relation we consider is that of a simple third degree polynomial. Given some random variable X, we define this relation as $Z = c_1 + c_2X + c_3X^2 + c_4X^3$ for some $c_1, c_2, c_3, c_4 \in \mathbb{R}$.

The third relation we consider is that of a fourth order reciprocal. Given some random variable X, we define this relation as $Z = c_1 + c_2\frac{1}{X} + c_3\frac{1}{X^2} + c_4\frac{1}{X^4}$ for some $c_1, c_2, c_3, c_4 \in \mathbb{R}$.

The fourth relation we consider is that of a logarithmic relation. Given some random variable X, we define this relation as $Z = c_1 + c_2\log(X)$ for some $c_1, c_2 \in \mathbb{R}$.

The fifth relation we consider is that of an exponential relation. Given some random variable X, we define this relation as $Z = c_1 + c_2e^X$ for some $c_1, c_2 \in \mathbb{R}$.

The sixth relation considered is that of a third order polynomial in two variables with interaction terms. Given some random variables X and Y, we define this relation as

$$Z = c_1 + c_2X + c_3Y + c_4X^2 + c_5XY + c_6Y^2 + c_7X^3 + c_8X^2Y + c_9XY^2 + c_{10}Y^3$$

for $c_i \in \mathbb{R}$ for all $i \in \{1, 2, \ldots, 10\}$.

The seventh relation considered is that of a third order reciprocal in two variables with interaction terms. Given some random variables X and Y, we define this relation as

$$Z = c_1 + c_2\frac{1}{X} + c_3\frac{1}{Y} + c_4\frac{1}{X^2} + c_5\frac{1}{XY} + c_6\frac{1}{Y^2} + c_7\frac{1}{X^3} + c_8\frac{1}{X^2Y} + c_9\frac{1}{XY^2} + c_{10}\frac{1}{Y^3}$$

for $c_i \in \mathbb{R}$ for all $i \in \{1, 2, \ldots, 10\}$.

The eighth relation considered is that of an exponential relation between two variables. Given some random variables X and Y, we define this relation as $Z = c_1 + c_2e^{XY}$.

The ninth relation considered is that of an exponential relation between two variables with one as base. Given some random variables X and Y, we define this relation as $Z = c_1 + c_2X^Y$.

The final relation considered is that of a piece-wise continuous relation, a test with maximum restrictions. Given some random variables X, Y and A, we define this relation as

$$Z = I\{Z = 1\}(A) + I\{Z = 2\}\left(c_1 \frac{\log(A)Y}{X^2}\right) + I\{Z = 5\}\left(c_1 A + c_2 X^2 + c_3 X Y^2\right)$$

.

5.2.1.2 Regression Relations in Aegon Data

At first this set of mathematical relationships between numeric variables may seem arbitrary, however, this set was chosen in cooperation with Aegon. It was decided that a subset of mathematical relationships such as this one should be able to capture a broad range of common relationships that may occur within the Aegon data during restoration.

In terms of the simple linear relation, one could consider the relationship between the outstanding value of a linear mortgage and the amount of interest the customer would need to pay. In this setting one would consider the interest rate to be the constant, whereas the outstanding value of the mortgage may be our random variable.

In terms of the polynomial relation, one could consider the case of a compounding interest. Given some random variable to be an interest rate and an arbitrary amount of elapsed time of the contract, this would result in a polynomial relationship between the total compounding interest with respect to the interest rate. Furthermore, the degree of this polynomial is dependent on the amount of elapsed time.

In terms of the reciprocal relation, one could consider the case of the current value of some monetary asset, the original value of this asset and the price change between the two. The original value of this asset would be constant, its current value a random variable and we wish to determine the price difference as a percentage. This then forms a reciprocal relation as we wish to divide the original value of the asset by its current value and then subtracting one to retrieve its percentage price difference.

In terms of the logarithmic relation, one could consider the case of the inverse of the exponential relation. In particular, we focus on the continuously compounding interest rate as an example. In this setting we would need the logarithmic relation to determine the interest rate or elapsed time given some current asset value versus its original value.

In terms of the exponential relation, the inverse holds true as per the logarithmic example. If we wish to determine the current value of an asset given the interest rate, time elapsed and original asset price, we can do so using an exponential relation.

In terms of the polynomial relation with interaction term, we refer once more to the polynomial relation without interaction term, where the example concerns a discretely compounding interest. If we wish to use this value to determine the current price of an asset, we will multiply the polynomial relation with a new random variable, resulting in an example of a polynomial relation with an interaction term.

In terms of the reciprocal relation with interaction term, in similar fashion to the polynomial relation with interaction term, we refer back to the reciprocal relation without interaction term. Rather than determining the price difference between the current value and the original value, we wish to determine the price difference at some point in the future or past. In this setting we would require interaction terms between the current value of the asset and the interest rate at that time.

In terms of the exponential relation between two variables, rather than consider a constant interest rate, we can consider it to be a random variable, the same can be done for the amount of time that has elapsed for an asset. As such, we then return to the continuously compounding interest rate case.

In terms of the exponential relation between two variables with one as base, it serves as a variation to the prior relation, where we allow for a more complex relation. Once more, one could consider the the base variable to represent some expected growth rate, say of a certain job sector and the exponent the elapsed time to consider.

The final relation shows to test the capability of our models to be able to capture piece-wise continuity. In our test settings, it serves as a case where a model would need to use the class provided by a categorical variable to then determine the unique continuous relation per class. In practice, one could consider the case of determining a individuals average tax burden. For incomes lower than 10000 this would result in a constant tax burden of zero. For incomes between 10000 and around the 25000 mark it become linear and from 25000 and greater, it follows a logarithmic trend.

5.2.1.3 Classification Relations

For classifiers the data relations are given by:

The first relation considered is that of a so-called NOT-relation. Given some Bernoulli random variable X, we define this relation as $Z = 1 - X$.

The second relation considered is that of a so-called OR-relation. Given some Bernoulli random variables X and Y, we define this relation as $Z = X + Y - XY$.

The third relation considered is that of a so-called AND-relation. Given some Bernoulli random variables X and Y, we define this relation as $Z = XY$.

The fourth relation considered is a shift test. The shift test considers some categorical random variable and shifts all categories to another one. Let us assume random variable X takes on values from $W = \{1, 2, 3\}$, then this test maps these values to $\{2, 3, 1\}$. Given some categorical random variable X, we define this relation as $Z = (X + 1) \mod |W|$. Here $|W|$ denotes the cardinality of the set W which is the set of categories random variable X can take on.

The fifth relation considered is a binary test. Given some categorical random variables X and Y, we define this relation as $Z = \begin{cases} X, & \text{if } X = Y \\ 100, & \text{if } X \neq Y \end{cases}$.

The sixth relation considered is a boundary test. Given some numeric random variable X and some real constant $a \in \mathbb{R}$, we define this relation as $Z = \begin{cases} 1, & \text{if } X <= a \\ 10, & \text{if } X > a \end{cases}$.

The seventh relation considered is a double boundary test. Given some numeric random variable X and some real constant $a \in \mathbb{R}$, we define this relation as $Z = \begin{cases} 1, & \text{if } X < -\frac{a}{2} \\ 10, & \text{if } -\frac{a}{2} \leq X \leq \frac{a}{2} \\ 0, & \text{if } X > \frac{a}{2} \end{cases}$.

The eighth relation considered is a multinomial test. Given some categorical random variables X, Y, A and B, we define this relation as $Z = \begin{cases} 10A + X, & \text{if } B = 0 \\ 10A + X + 2, & \text{if } B = 1 \\ 10A + Y + 4, & \text{if } B = 2 \\ 10A + Y + 7, & \text{if } B = 3 \end{cases}$.

The ninth relation considered is a fully disjoint test. Given some categorical random variables X and Y, we define this relation as $Z = \begin{cases} 2^{(Y+1)}, & \text{if } X = 1 \\ 5(Y + 1), & \text{if } X = 0 \end{cases}$.

The final relation considered is a partially disjoint test. Given some categorical random variables X and Y, we define this relation as $Z = \begin{cases} 16, & \text{if } X = 1, (Y \mod 2) = 1 \\ 2, & \text{if } X = 1, (Y \mod 2) = 2 \\ 1, & \text{if } X = 0, (Y \mod 4) = 2 \\ 5, & \text{if } X = 0, (Y \mod 4) \neq 2 \end{cases}$.

It is of note that for the categorical relations, X will always take on categories $\{0, 1\}$, Y will always take on categories $\{0, 1, 2\}$, B will always take on categories $\{0, 1, 2, 3\}$, A will always take on categories $\{0, 1, 2, ..., 9\}$.

5.2.1.4 Classification Relations in Aegon Data

We shall provide some examples for these classification relations as they may occur on the Aegon data sets.

In terms of the NOT-relation, we may consider the case that a random variable may only take on one of a binary set of values, namely if we consider a male, we expect its value in a *male* random variable to be one, while its value in the $female$ random variable will be 0 and vice versa.

In terms of the OR-relation, we may consider some check associated to the presence of an ID card or passport in the database. If there is one of either choice or both available, then this check passes, else there is no identification document available.

In terms of the AND-relation, we may consider some check on marital status of an individual. If it is indicated that the individuals partner is married and not divorced, then the individuals marital status should also be married.

In terms of the shift relation, we may consider some random variable containing an index that indicates which iteration of a contract is the current one versus the prior one. We expect the value of this random variable to be shifted to a new value on a renewal of a contract for example.

In terms of the binary relation, we may consider specific jobs in a broader job category. One of these specific jobs may be unique and as such not a lot of data is available in terms of risk. Such a job may thus require custom rates, whereas the other specific jobs in the set can be considered together to have some set rate.

In terms of the boundary relation, we may consider the case where one pays a certain interest rate up to a specified size of a loan. If the desired loan value exceeds this boundary a different interest rate may apply.

In terms of the double boundary relation, we may once more consider varying interest rates dependent on some other value such as the size of a loan.

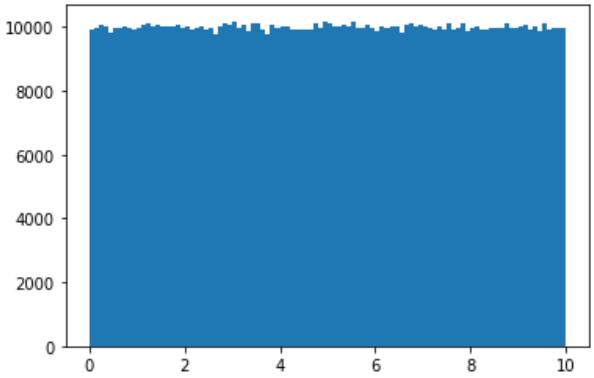
In terms of the multinomial relation, we may consider a categorical variable that indicates using a class the type of mortgage, whether its a mortgage of 1, 5, 10, 20 or 30 years in length. In this setting we can consider our target variable to be the expected end year or month given the starting date and the class value.

In terms of the fully disjoint test, we can consider binary random variable that indicates the underlying relation between the target variable and another input variable. For example, we may require different risk categories assigned dependent on the binary variable that indicates if the individual has a partner or not, given the job category their specific job falls into.
Finally, in terms of the partially disjoint test serves as an extension to the fully disjoint case. One may need to assign a risk class to a specific combination of type of job and their marital status, but for some job classes the marital status does not result in a unique risk class.

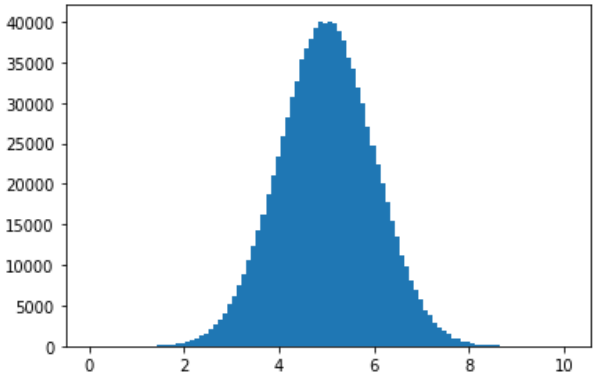
5.2.2 Data Distributions

Though data relations are important, it is not only the relationship between the input and output variables that is key in model selection. The distribution of the data plays a crucial role within these data relations. For each numeric variable we will consider the uniform, gamma, normal, a bi-model and a tri-modal distributions and provide histograms for samples of size one million.

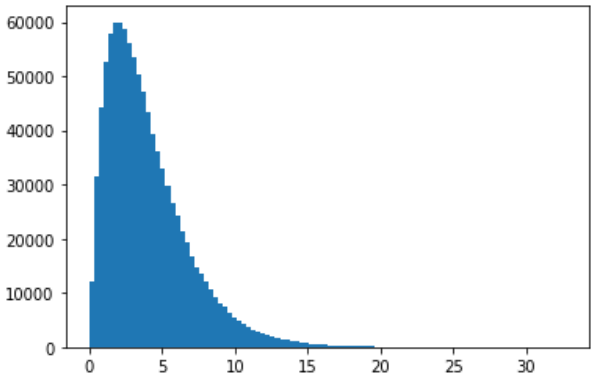
The first distribution we consider for numeric variables is the simplest one, namely the uniform distribution. Using this distribution in numeric testing allows us to best identify which models can identify the underlying relation or data test best.



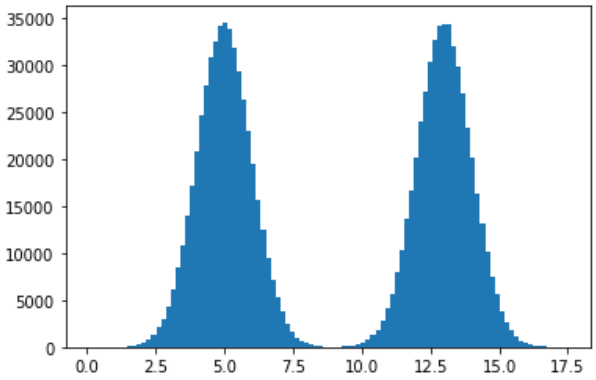
However, the uniform distribution is not realistic in most real-world data scenarios. As such, we consider our second distribution to be the normal distribution. Due to the law of large numbers in combination with the central limit theorem, sampling distributions tend to become normally distributed and as such this distribution is a common occurrence in big data.



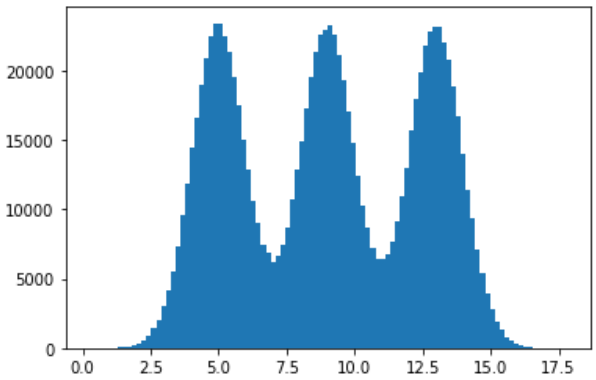
On the other hand, not all data can be considered sufficiently sized for the assumption of normality to hold. For this reason data sets may actually be more skewed and have heavy tails. For this reason we also consider the gamma distribution for our theoretical testing.



All distributions that have been considered thus far have all had one mean, that is one location in the domain of the distribution with the highest density. In order to make our theoretical testing more robust, we shall also consider data that has more than one mean. We consider first a Bi-Modal distribution, namely we sample randomly from the Bernoulli distribution and based on the outcome, we sample from one of two differently centered normal distributions, as such constructing a distribution with two high density areas in its probability density.



Finally, we consider a Tri-Modal distribution for the sake of possible increased complexity. We construct this distribution by first sampling from the standard uniform distribution and based on which third of the zero-one interval the result lands, we sample from a differently centered normal distribution. The distance between the means of first and second normal distribution are equal to that between the second and the third.



For each categorical variable we will consider a uniform, minority dominant and majority dominant distribution. We provide the histograms for one million samples for each distribution and each number

of classes as shown in the theoretical tests.
Once more, the uniform distribution gives equal weight to the occurrence rates of each class and provides us with a test that can focus on aiming to capture underlying relationships within the data.

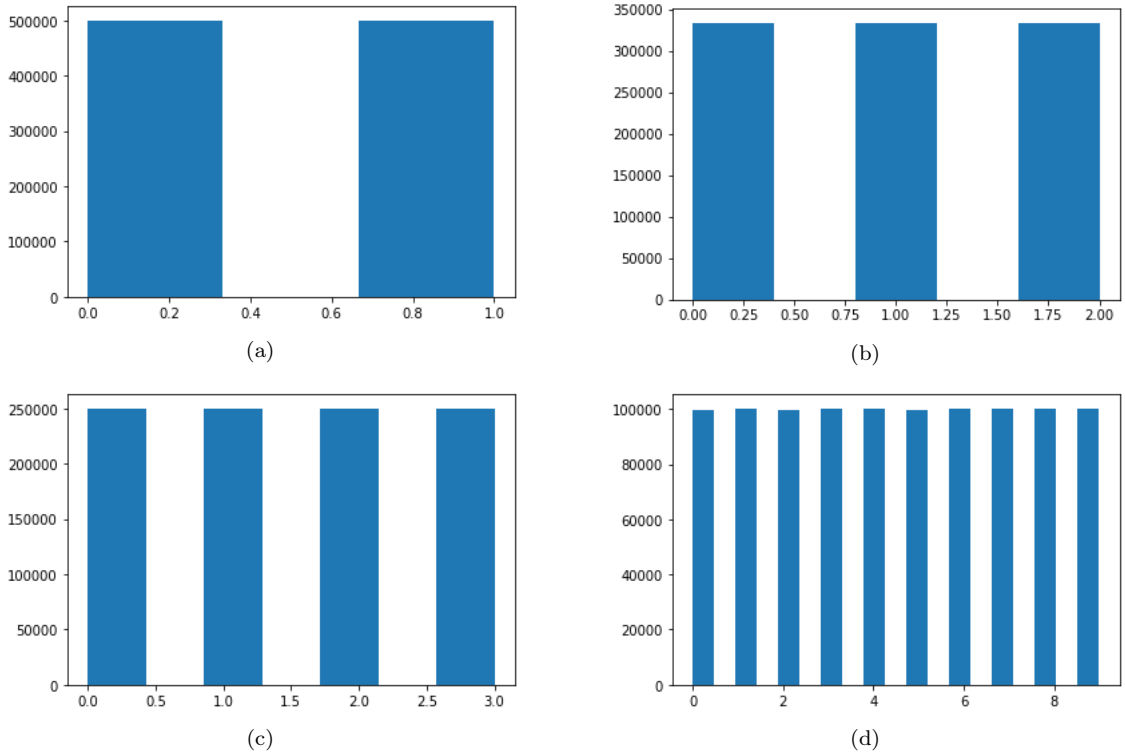


Figure 5.4: Uniform sample probability densities for all test cases.

Alternatively, we consider the occurrence rates of classes to be greatly skewed. We consider a minority of the total number of classes to be dominant, such that less than one third of classes carries 90% of the weight.

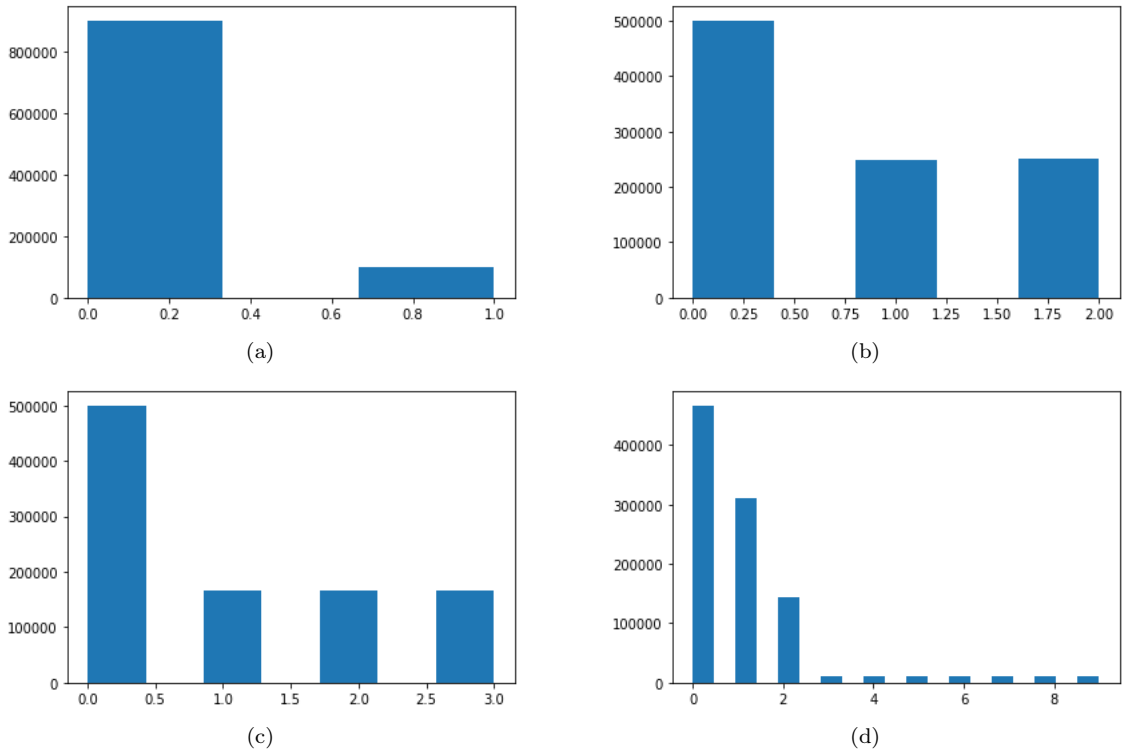


Figure 5.5: Minority Dominant sample probability densities for all test cases.

We opt to also use the opposite approach of the previous distribution and consider a scenario in which a majority of classes carries 90% of the weight. We set majority to be equal to half of the classes in the case we have an even number of classes. In the case of an uneven number of classes it become half of the classes plus a half.

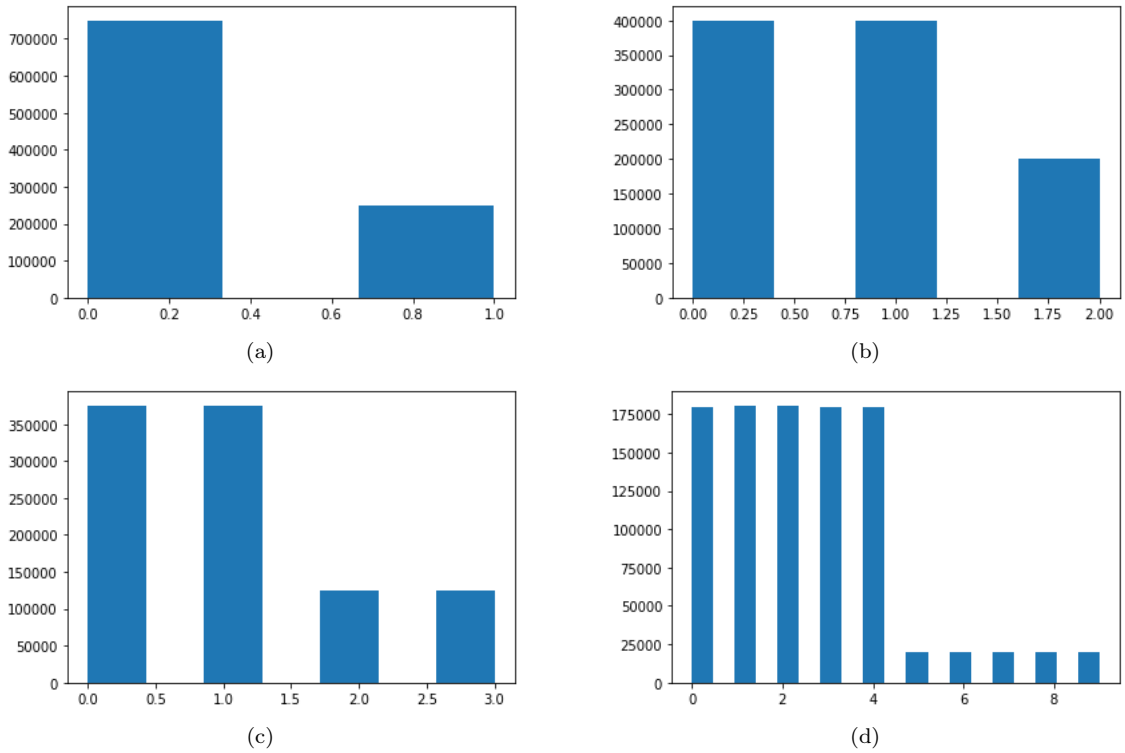


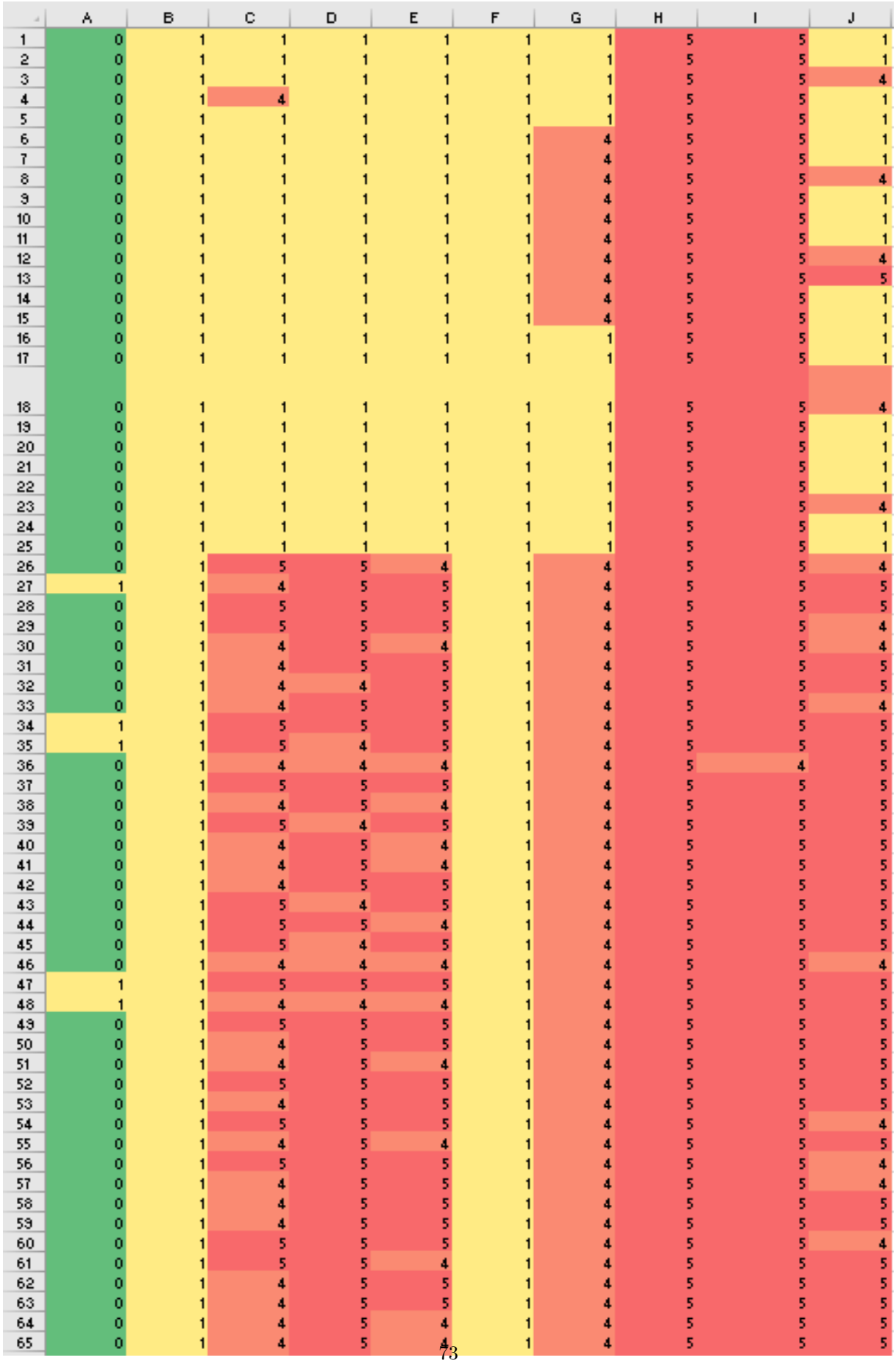
Figure 5.6: Majority Dominant sample probability densities for all test cases.

The choice for these distributions lies in their properties. For regressors using the uniform distribution, we can acquire insight into a models capability to determine the underlying data relation over the entire domain. By using normal distribution, we can acquire insight into a models capability to handle the most common occurring data distribution by the Central Limit Theorem. By using gamma distributions, we gain insight into which model can best handle heavy tails in the data. By using a bi-modal distribution, we gain insight into the capacity of a model to deal with two high density areas. Finally, by using a tri-modal distribution, we gain insight into the capacity of a model to deal with relatively sporadic data, that transitions often from high density to low density areas.
For classifiers, using the uniformly distributed classes, we gain insight in the capacity to discover a relation given the entire domain represented equally. By considering a minority dominant distribution, we gain insight into which model can handle many relatively small representations of classes with a common occurring dominant minority. By considering a majority dominant distribution, we gain insight into which model can deal with a minority of small classes versus a large over-shadowing majority.
By considering 5 distributions for each random variable occurring in the regression data relations, we will consider in total 250 tests. For classification the 3 distributions for each random variable will result in total 117 tests. Though this number may seem large, it is important to note that these are tests that will be conducted once, after which results will be stored for model selection. Having constructed a testing setup for our model selection strategy, we need to quantify our criteria for determining which model to use. Within the testing setup, we recognize three possible criteria: the distribution of input variables, the distribution of the output variable and the data relation.
Considering, however, that the data relation is unknown in the restoration of missing data, we are unable to use this as a criterion for model selection. Using the distribution of input variables, though possible, is difficult. The variables as provided by the feature selection algorithm need not necessarily be relevant for the underlying data relation and can consist of relatively large numbers of variables for distribution matching. Based on these considerations, the use of the distribution of the output variable seems to be the most obvious choice.
Using this theoretical testing approach, we have access to the best model and possibly sub-variant for a restoration task given output distribution. The determination of the output distribution that best

matches the distribution of the actual input data, we use a correlation ranking. That is, the theoretical output that best matches the actual output based on their absolute correlation value. This method of model selection then reduces to determining whether the actual output is either numeric or categorical by nature and then using the model that scored best for the theoretical output distribution with the highest absolute correlation. This novel model selection approach means that instead of having to train hundreds of different models, we need only calculate correlation between two one-dimensional distributions, reducing time and computational complexity significantly.
The main downside of this new approach is the calculations of correlation between the actual output variable and those of theoretical testing. Many of the possible theoretical output distributions themselves can be highly correlated. This indicates that if we select a model based on a marginal difference between the correlation with regards to two theoretical output distributions, that this selection can be sample dependent with respect to the actual output variable.

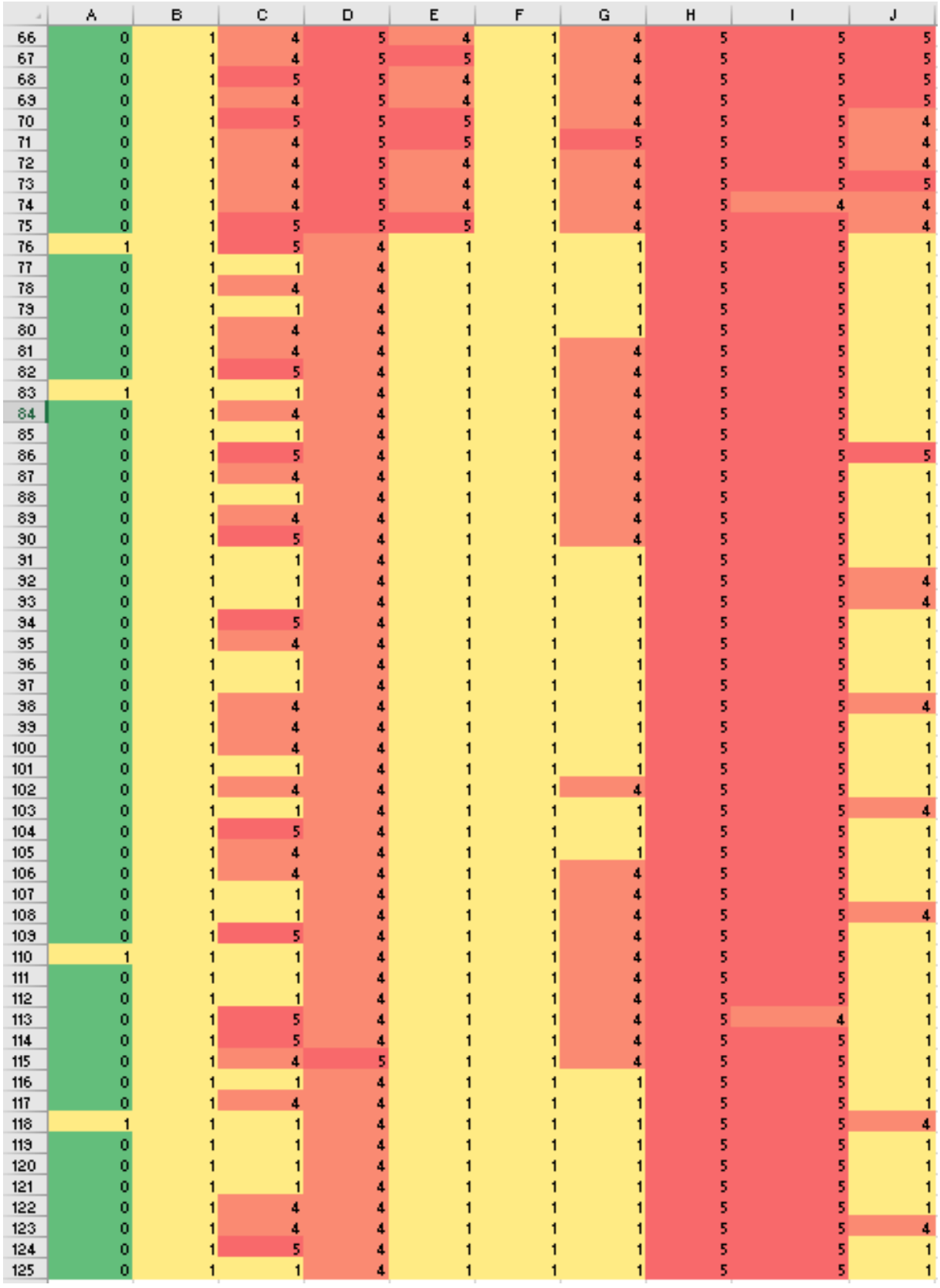
5.2.3 Unconditional Probability

The aim of model selection is selecting the best model for a specific real-world data scenario based on theoretical testing. In many cases, though, the actual data will not perfectly align with this theoretical testing approach. This forms one of the greatest downsides to the previous flowchart model selection approach. However, the theoretical testing can be used in another model selection approached that aims to compensate for shortcomings of the previous approaches while lowering time and computational complexity. In order to do so, we shall first analyze the results of theoretical testing.
For the theoretical testing in the numeric feature setting, we provide the results in the following heatgraphs:



	A	B	C	D	E	F	G	H	I	J
1	0	1	1	1	1	1	1	5	5	1
2	0	1	1	1	1	1	1	5	5	1
3	0	1	1	1	1	1	1	5	5	4
4	0	1	4	1	1	1	1	5	5	1
5	0	1	1	1	1	1	1	5	5	1
6	0	1	1	1	1	1	4	5	5	1
7	0	1	1	1	1	1	4	5	5	1
8	0	1	1	1	1	1	4	5	5	4
9	0	1	1	1	1	1	4	5	5	1
10	0	1	1	1	1	1	4	5	5	1
11	0	1	1	1	1	1	4	5	5	1
12	0	1	1	1	1	1	4	5	5	4
13	0	1	1	1	1	1	4	5	5	5
14	0	1	1	1	1	1	4	5	5	1
15	0	1	1	1	1	1	4	5	5	1
16	0	1	1	1	1	1	1	5	5	1
17	0	1	1	1	1	1	1	5	5	1
18	0	1	1	1	1	1	1	5	5	4
19	0	1	1	1	1	1	1	5	5	1
20	0	1	1	1	1	1	1	5	5	1
21	0	1	1	1	1	1	1	5	5	1
22	0	1	1	1	1	1	1	5	5	1
23	0	1	1	1	1	1	1	5	5	4
24	0	1	1	1	1	1	1	5	5	1
25	0	1	1	1	1	1	1	5	5	1
26	0	1	5	5	4	1	4	5	5	4
27	1	1	4	5	5	1	4	5	5	5
28	0	1	5	5	5	1	4	5	5	5
29	0	1	5	5	5	1	4	5	5	4
30	0	1	4	5	4	1	4	5	5	4
31	0	1	4	5	5	1	4	5	5	5
32	0	1	4	4	5	1	4	5	5	5
33	0	1	4	5	5	1	4	5	5	4
34	1	1	5	5	5	1	4	5	5	5
35	1	1	5	4	5	1	4	5	5	5
36	0	1	4	4	4	1	4	5	4	5
37	0	1	5	5	5	1	4	5	5	5
38	0	1	4	5	4	1	4	5	5	5
39	0	1	5	4	5	1	4	5	5	5
40	0	1	4	5	4	1	4	5	5	5
41	0	1	4	5	4	1	4	5	5	5
42	0	1	4	5	5	1	4	5	5	5
43	0	1	5	4	5	1	4	5	5	5
44	0	1	5	5	4	1	4	5	5	5
45	0	1	5	4	5	1	4	5	5	5
46	0	1	4	4	4	1	4	5	5	4
47	1	1	5	5	5	1	4	5	5	5
48	1	1	4	4	4	1	4	5	5	5
49	0	1	5	5	5	1	4	5	5	5
50	0	1	4	5	5	1	4	5	5	5
51	0	1	4	5	4	1	4	5	5	5
52	0	1	5	5	5	1	4	5	5	5
53	0	1	4	5	5	1	4	5	5	5
54	0	1	5	5	5	1	4	5	5	4
55	0	1	4	5	4	1	4	5	5	5
56	0	1	5	5	5	1	4	5	5	4
57	0	1	4	5	5	1	4	5	5	4
58	0	1	4	5	5	1	4	5	5	5
59	0	1	4	5	5	1	4	5	5	5
60	0	1	5	5	5	1	4	5	5	4
61	0	1	5	5	4	1	4	5	5	5
62	0	1	4	5	5	1	4	5	5	5
63	0	1	4	5	5	1	4	5	5	5
64	0	1	4	5	4	1	4	5	5	5
65	0	1	4	5	4	1	4	5	5	5

Figure 5.7: Overview of best numeric model per test and distribution configuration.



	A	B	C	D	E	F	G	H	I	J
66	0	1	4	5	4	1	4	5	5	5
67	0	1	4	5	5	1	4	5	5	5
68	0	1	5	5	4	1	4	5	5	5
69	0	1	4	5	4	1	4	5	5	5
70	0	1	5	5	5	1	4	5	5	4
71	0	1	4	5	5	1	5	5	5	4
72	0	1	4	5	4	1	4	5	5	4
73	0	1	4	5	4	1	4	5	5	5
74	0	1	4	5	4	1	4	5	4	4
75	0	1	5	5	5	1	4	5	5	4
76	1	1	5	4	1	1	1	5	5	1
77	0	1	1	4	1	1	1	5	5	1
78	0	1	4	4	1	1	1	5	5	1
79	0	1	1	4	1	1	1	5	5	1
80	0	1	4	4	1	1	1	5	5	1
81	0	1	4	4	1	1	4	5	5	1
82	0	1	5	4	1	1	4	5	5	1
83	1	1	1	4	1	1	4	5	5	1
84	0	1	4	4	1	1	4	5	5	1
85	0	1	1	4	1	1	4	5	5	1
86	0	1	5	4	1	1	4	5	5	5
87	0	1	4	4	1	1	4	5	5	1
88	0	1	1	4	1	1	4	5	5	1
89	0	1	4	4	1	1	4	5	5	1
90	0	1	5	4	1	1	4	5	5	1
91	0	1	1	4	1	1	1	5	5	1
92	0	1	1	4	1	1	1	5	5	4
93	0	1	1	4	1	1	1	5	5	4
94	0	1	5	4	1	1	1	5	5	1
95	0	1	4	4	1	1	1	5	5	1
96	0	1	1	4	1	1	1	5	5	1
97	0	1	1	4	1	1	1	5	5	1
98	0	1	4	4	1	1	1	5	5	4
99	0	1	4	4	1	1	1	5	5	1
100	0	1	4	4	1	1	1	5	5	1
101	0	1	1	4	1	1	1	5	5	1
102	0	1	4	4	1	1	4	5	5	1
103	0	1	1	4	1	1	1	5	5	4
104	0	1	5	4	1	1	1	5	5	1
105	0	1	4	4	1	1	1	5	5	1
106	0	1	4	4	1	1	4	5	5	1
107	0	1	1	4	1	1	4	5	5	1
108	0	1	1	4	1	1	4	5	5	4
109	0	1	5	4	1	1	4	5	5	1
110	1	1	1	4	1	1	4	5	5	1
111	0	1	1	4	1	1	4	5	5	1
112	0	1	1	4	1	1	4	5	5	1
113	0	1	5	4	1	1	4	5	4	1
114	0	1	5	4	1	1	4	5	5	1
115	0	1	4	5	1	1	4	5	5	1
116	0	1	1	4	1	1	1	5	5	1
117	0	1	4	4	1	1	1	5	5	1
118	1	1	1	4	1	1	1	5	5	4
119	0	1	1	4	1	1	1	5	5	1
120	0	1	1	4	1	1	1	5	5	1
121	0	1	1	4	1	1	1	5	5	1
122	0	1	4	4	1	1	1	5	5	1
123	0	1	4	4	1	1	1	5	5	4
124	0	1	5	4	1	1	1	5	5	1
125	0	1	1	4	1	1	1	5	5	1

Figure 5.8: Overview of best numeric model per test and distribution configuration.

On the one axis the numbered indexing indicates which data distribution we are in. For example index 1 corresponds to 000, where each individual number within 000 indicates the distribution of each variable. In this case the first zero indicates that X is sampled from the uniform distribution, the second zero indicates that Y is sampled from the uniform distribution and, finally, the third zero indicates that A is sampled from the uniform distribution. The following indexes always try to choose the next distribution for the third variables first (000 goes to 001, turning A to gamma), then if not possible the next distribution is chosen for the second variable and the third one resets to uniform to repeat the process (004 goes to 010, turning A to uniform and Y to gamma). This is done until all 81 combinations pass through the tests. On the other axis the lettered indexing indicates which test we are in. A corresponds to the first numeric test and B to the second up until J. Within the graph, each cell is provided with the model index of the best performing model with respect to their mean-squared error.

In an ideal scenario, we would analyze all rows or columns and determine some fixed relation with which we could select the best model in each scenario. However, in the real data scenario the relation on which the test is based, is not known prior and the different combinations of distributions are equally as hard to deduce from the data as this would imply feature selection would need to yield very precise results: no redundant variables, no unrelated variables, all related variables.
For this reason, we propose the use of a probabilistic approach to the model selection process. We can convert this heatgraph to a probability density of the best model weighted with respect to occurrence in the heatgraph. These results are summarized in the following table:

	LR	PR	EPR	MLP	RF	KNN
Number	116	511	0	0	234	389
Probability	0.09	0.41	0.00	0.00	0.19	0.31

Table 32: Summary of unconditional numeric model results

From this interpretation of the results in the heatgraph, we can deduce a model selection strategy. If we only consider the model that performs best in most test and distribution combinations, we have the best model in the most scenarios with respect to the other models. This means that rather than having to train all 6 models for each scenario, we only use 1 model, in this case polynomial regression. This strategy would then imply, based on theoretical testing, that we would have the best model with a 41% chance.
Though such a strategy does significantly reduce computing time, it also results in the best model less than half of the time based on testing. This could be significantly worse if the real-world data does not adhere to this theoretical testing. We can, however, improve on this strategy by also considering the second best model based on the unconditional density: k-nearest neighbors.
If we use the model selection strategy that constructs both the polynomial regression and k-nearest neighbors regression and then select the best one between the two, we would have to only train two models every time. In this scenario, we get a significant improvement on the 41% chance from just using polynomial regression, namely we get a 72% chance based on theoretical testing. The next logical step is to add the third best model unconditionally, being random forest regression. In this way we would compare the model scores for three different model types yielding a theoretical accuracy of 91%.
Though, again we have no way of verifying the Aegon data adheres to these theoretical rules, so we must aim to devise a strategy that can consider all models. Expanding the current strategy would mean just constructing and training all models yielding 100% theoretical accuracy in terms of picking the best model. This means, we need to propose a different strategy.

When considering the previous strategy, one may intuitively consider a strategy that turns a part of the model selection conditional to reduce time complexity. Rather than taking the best three models in terms of their unconditional probability in being the best model, we start by constructing the first two models. If model 1, in this case polynomial regression, outperforms model 2, in this case nearest neighbors, we opt to use model 1. On the other hand, if model 2 outperforms model 1, we decide to also construct model 3, in this case random forest, and is model 2 outperforms model 3, we use model 3 and otherwise use model 2.
That is, if we extend this strategy to all models, rather than constructing all 5 models every time, we aim

to only construct those models necessary to determine the best one. Using this strategy, we can determine the expected number of different models that will be used to be given by the following formula:

$$\mathbb{E}[N] = P(x_1)P(x_1 > x_2) + \sum_{i=2}^{n-1} P(x_i)P(x_{i-1} < x_i)P(x_i < x_{i+1}) + P(x_n)P(x_{n-1} < x_n)$$

This expectation can of course not be accurately computed from the unconditional probabilities of each model being the best model. Thus we provide the accuracy in determining the best model and expected number of models used based on our theoretical testing in the following table:

Success Rate	Mean Models Used	Std Models Used
100%	2.964	0.983

Table 33: Unconditional strategy results for theoretical testing.

As we can see from the results in the table, this strategy under our current theoretical testing settings will always discover the best model and this strategy lowered the expected number of models used from 5 to 2.964.
Though this strategy holds true for our particular theoretical testing, it does not need to hold true if we wish to enrich the mathematical relations we consider of expand the selection of distributions. We wish to use a more robust strategy that actually analyzes the underlying model rankings.

5.2.4 Unconditional Ranking

Rather than using the unconditional probability from the previous section, this section shall propose an upgraded methodology for model selection based on the theoretical testing.
One strategy we propose is the use of the unconditional mean ranking per model. These are determined by considering the ranking in terms of mean squared error for each model for each test, under each distribution selection. The following table summarizes these results:

	LR	PR	EPR	MLP	RF	KNN
Mean Rank	3.6896	1.6376	4.0584	3.3696	1.0688	1.176
Rank of Mean Rank	5	3	6	4	1	2

Table 34: Summary of unconditional numeric model rank results

The table provides the mean rank for each model over all test settings and the relative rank of these mean ranks for the various models.
Based on these results, if we carry over the same mindset as for the previous strategy, we may propose to start with random forest, then nearest neighbor, then polynomial regression, etc. We also provide the success rate and expected number of models used for this strategy:

Success Rate	Mean Models Used	Std Models Used
100%	3.500	1.087

Table 35: Unconditional ranking strategy results for theoretical testing.

As with the previous proposed selection strategy, we always manage to construct the best model in the theoretical setting. However, this model selection strategy does require the use of more models on average. Thus, purely based on the theoretical testing, our preferred method of model selection would be the unconditional probability density method.
This new unconditional ranking method is more robust, in that it considers the relative rankings of the models. To get a better understanding of why this is the case, we consider a specific scenario. Assume a model 1 dominates all other models in 45% of cases and the relative rankings are constant for all models in this set of cases, for simplicity assume rankings: 1, 2, 3, 4, 5, 6. If in another 45% of cases the ranking is given, such that: 3, 1, 2, 4, 5, 6. Finally, the last 10% give rankings: 5, 6, 4, 2, 3, 1.

The unconditional probabilistic approach teaches us that the best strategy is given by using model 1, then model 2 and finally model 3. If we do so in the 5, 6, 4, 2, 3, 1 ranking setting, this means we consider model 1 to be the best model in this scenario as model 1 outperforms model 2 even though all other models outperform both.
The unconditional ranking approach gives us as optimal strategy using model 2 first, then model 1 and finally, model 3. This does indeed solve the 10% of scenarios and once again returns a perfect selection score.
If we consider another theoretical scenario, similar to the previous one, we can show that even this unconditional ranking approach is far from perfect:
Assume a model 1 dominates all other models in 45% of cases and the relative rankings are constant for all models in this set of cases, for simplicity assume rankings: 1, 2, 3, 4, 5, 6. If in another 45% of cases the ranking is given, such that: 2, 1, 3, 4, 5, 6. Finally, the last 10% give rankings: 4, 6, 5, 2, 3, 1.
In this scenario both the unconditional probability and unconditional ranking strategy fails to discover the best model for the 10% of cases. This is why we propose a final upgrade to the model selection strategy.

5.2.5 Conditional Ranking

To compensate for the unconditional rankings shortcomings, we propose a strategy that makes use of conditional rankings. That is, we wish to know how models performs given a certain model is best. We do so by considering each scenario in which one of the models performs best and then average the rankings of all other models in this scenario. The results of this are presented in the following table:

	LR	PR	EPR	MLP	RF	KNN
LR	0.000	1.000	5.000	4.000	2.233	2.767
PR	4.037	0.000	4.896	3.031	1.280	1.755
EPR	X	X	X	X	X	X
MLP	X	X	X	X	X	X
RF	3.897	3.338	3.432	3.256	0.000	1.077
KNN	4.208	2.956	3.054	3.694	1.087	0.000

Table 36: Mean conditional ranking of regression models given the best model with mean index 0.

Converting these mean rankings to rankings for each scenario, this yields:

	LR	PR	EPR	MLP	RF	KNN
LR	1	2	5	3	4	6
PR	5	1	6	4	2	3
EPR	X	X	X	X	X	X
MLP	X	X	X	X	X	X
RF	6	4	5	3	1	2
KNN	6	3	4	5	2	1

Table 37: Ranking of conditional ranking of regression models given the best model with mean index 0.

This information now gives us the final information we need to determine the new probabilistic model selection approach. Using these rankings we can define an optimization method that selects the best order for constructing and training models. Unlike the previously considered selection strategies, we are not able to consider these scenarios separately, we need to devise a way to determine an ordering for the use of models such that any previously constructed model outperforms the newly constructed model, if and only if, it is the best model for that conditional scenario. This forms the most important part of the optimization objective. If it is the case that multiple orderings satisfy this criterion, then we wish to use the model construction ordering that minimizes the expected number of model constructions needed to get to the best model. For the determination of this expectation, we will use the unconditional probabilities as gathered previously.

We start by constructing a set containing all possible orderings for the regression case separately. Then given one of these orderings, we verify whether it finds the best model for each best model scenario. For example, if we consider the case where we have a ordering $p = \{1, 2, 3, 4, 5, 6\}$ for the regression case and we consider the rankings in the case that random forest regression (RF) performs best, we start at the orderings first index $p_1 = 1$. This implies that this ordering would start by using the model with index 1, which in this case would be linear regression (LR), as indicated by the order of models in the ranking table. This model in the random forest regression scenario has a rank of 6. Next we consider the second index in the ordering $p_2 = 2$, where the ordering would use the second model to be polynomial regression (PR), this model has relative ranking 4. Given our strategy, this would indicate that the second model most likely will outperform the first model in comparison and as such our strategy now involves also verifying for a third model. We take $p_3 = 3$ the third index in our ordering to be the use of elastic net polynomial regression (EPR). This model has relative ranking 5 in the scenario in which the random forest is expected to perform best. Based on our strategy this indicates that polynomial regression will outperform elastic net polynomial regression and we decide to use polynomial regression as the best model for restoration given this ordering. However, random forest is the best model in this scenario, thus this ordering does not satisfy the minimum requirement for our strategy.
Once we have determined all orderings such that we always find the best model in the first local minimum, we determine of these orderings which is the most efficient given our theoretical testing scenario. We do so by using the probability that a certain model is the best model based on the theoretical testing. Using this probability, we can determine the expected number of models each ordering will need to find the best model. We define the following criterium:

$$L(q) = \sum_{i=1}^{k} p_i n_{q,i} \tag{33}$$

- q an ordering.
- p_i is the probability model i is best in theoretical testing.
- $n_{q,i}$ is the number of different models ordering q needs to reach the best model i.

By now choosing the ordering with the minimum value for the expected number of models trained, we acquire our desired ordering for our model selection.
By pure coincidence, the best strategy for this model selection strategy is the same one as constructed by the unconditional probabilistic approach, whose results are summarized here:

Success Rate	Mean Models Used	Std Models Used
100%	2.964	0.983

Table 38: Conditional ranking strategy results for theoretical testing.

This strategy, will now form the basis of our model selection process as this strategy is the most robust. If as human input we allow the consideration of more models or more distributions for the theoretical testing, this strategy guarantees us that under each scenario in which any specific model outperforms the other models, we always find the first local minimum to be the model that best performs.

5.2.6 Classification

We aim to replicate this process for the case of theoretical testing for the classification cases. We first provide the map of best models for each scenario once more:

	A	B	C	D	E	F	G	H	I	J
1	12345	234	234	12345	234	1345	1345	23	12345	2345
2	12345	234	234	12345	234	1345	145	23	12345	2345
3	12345	234	234	12345	234	15	1345	2	12345	2345
4	12345	234	234	12345	234	15	1345	23	12345	2345
5	12345	234	234	12345	234	1345	15	23	12345	2345
6	12345	234	234	12345	234	15	1345	2	12345	2345
7	12345	234	234	12345	234	15	15	23	12345	2345
8	12345	234	234	12345	234	15	1345	2	12345	2345
9	12345	234	234	12345	234	1345	1345	2	12345	2345
10	12345	234	234	12345	234	15	15	2	12345	2345
11	12345	234	234	12345	234	1345	12345	23	12345	2345
12	12345	234	234	12345	234	15	1345	23	12345	2345
13	12345	234	234	12345	234	15	1345	23	12345	2345
14	12345	234	234	12345	234	1345	125	2	12345	2345
15	12345	234	234	12345	234	1345	1345	2	12345	2345
16	12345	234	234	12345	234	145	1345	2	12345	2345
17	12345	234	234	12345	234	15	15	23	12345	2345
18	12345	234	234	12345	234	1345	1345	23	12345	2345
19	12345	234	234	12345	234	1345	1345	2	12345	2345
20	12345	234	234	12345	234	15	135	23	12345	2345
21	12345	234	234	12345	234	15	1345	23	12345	2345
22	12345	234	234	12345	234	125	135	23	12345	2345
23	12345	234	234	12345	234	15	1345	2	12345	2345
24	12345	234	234	12345	234	15	1345	23	12345	2345
25	12345	234	234	12345	234	145	15	2	12345	2345
26	12345	234	234	12345	234	15	15	23	12345	2345
27	12345	234	234	12345	234	135	15	23	12345	2345
28	12345	234	234	12345	234	12345	15	2	12345	2345
29	12345	234	234	12345	234	15	15	2	12345	2345
30	12345	234	234	12345	234	1345	15	2	12345	2345
31	12345	234	234	12345	234	15	15	23	12345	2345
32	12345	234	234	12345	234	15	15	2	12345	2345
33	12345	234	234	12345	234	1345	15	23	12345	2345
34	12345	234	234	12345	234	1345	15	23	12345	2345
35	12345	234	234	12345	234	1345	15	2	12345	2345
36	12345	234	234	12345	234	1345	15	23	12345	2345
37	12345	234	234	12345	234	1345	1345	23	12345	2345
38	12345	234	234	12345	234	135	15	23	12345	2345
39	12345	234	234	12345	234	135	15	23	12345	2345
40	12345	234	234	12345	234	1345	1345	23	12345	2345
41	12345	234	234	12345	234	15	145	23	12345	2345

Figure 5.9: Overview of best categorical model per test and distribution configuration.

	A	B	C	D	E	F	G	H	I	J
41	12345	234	234	12345	234	15	145	23	12345	2345
42	12345	234	234	12345	234	15	15	23	12345	2345
43	12345	234	234	12345	234	15	15	23	12345	2345
44	12345	234	234	12345	234	145	1345	2	12345	2345
45	12345	234	234	12345	234	15	15	23	12345	12345
46	12345	234	234	12345	234	135	1345	23	12345	2345
47	12345	234	234	12345	234	1345	1345	23	12345	2345
48	12345	234	234	12345	234	145	15	2	12345	2345
49	12345	234	234	12345	234	15	135	23	12345	2345
50	12345	234	234	12345	234	1345	1345	23	12345	2345
51	12345	234	234	12345	234	12345	15	23	12345	2345
52	12345	234	234	12345	234	1345	15	23	12345	2345
53	12345	234	234	12345	234	135	135	2	12345	2345
54	12345	234	234	12345	234	15	15	23	12345	2345
55	12345	234	234	12345	234	15	135	23	12345	2345
56	12345	234	234	12345	234	1345	1345	2	12345	2345
57	12345	234	234	12345	234	125	15	23	12345	2345
58	12345	234	234	12345	234	135	1345	3	12345	2345
59	12345	234	234	12345	234	15	1345	23	12345	2345
60	12345	234	234	12345	234	1345	15	2	12345	2345
61	12345	234	234	12345	234	15	1345	23	12345	2345
62	12345	234	234	12345	234	1345	1345	23	12345	2345
63	12345	234	234	12345	234	15	15	2	12345	2345
64	12345	234	234	12345	234	1345	1345	23	12345	2345
65	12345	234	234	12345	234	1345	1345	23	12345	2345
66	12345	234	234	12345	234	145	15	23	12345	2345
67	12345	234	234	12345	234	15	1345	2	12345	2345
68	12345	234	234	12345	234	145	1345	23	12345	2345
69	12345	234	234	12345	234	1345	15	23	12345	2345
70	12345	234	234	12345	234	1345	1345	23	12345	2345
71	12345	234	234	12345	234	1345	15	2	12345	2345
72	12345	234	234	12345	234	1345	1345	23	12345	2345
73	12345	234	234	12345	234	15	1345	23	12345	2345
74	12345	234	234	12345	234	1345	1345	3	12345	2345
75	12345	234	234	12345	234	1345	15	23	12345	2345
76	12345	234	234	12345	234	1345	15	23	12345	2345
77	12345	234	234	12345	234	1345	1345	23	12345	2345
78	12345	234	234	12345	234	15	12345	23	12345	2345
79	12345	234	234	12345	234	145	15	23	12345	2345
80	12345	234	234	12345	234	15	15	2	12345	2345
81	12345	234	234	12345	234	15	1345	23	12345	2345

Figure 5.10: Overview of best categorical model per test and distribution configuration.

As can be seen from the map, for the categorical testing cases, it occurs that two or more models all perform best by yielding the same score for a specific case. As such the categorical testing shows us another scenario in which the unconditional probabilistic strategy considered previously is not robust. There are now many cases for which a subset of models all perform best or even all of them, all the more reason to only consider the conditional ranking strategy.
We start by providing the conditional probabilities each model performs best:

	EM	MLP	RF	KNN	KM
Number	406	653	706	648	486
Probability	0.501	0.806	0.872	0.8	0.6

Table 39: Summary of conditional categorical model results

So rather than use unconditional probabilities to weigh the expected number of models used as in the numeric scenario, we now have to use conditional weighting. The remainder of the conditional ranking selection process remains the same.
Next we consider the conditional rankings for the categorical theoretical testing:

	EM	MLP	RF	KNN	KM
EM	0.000	1.473	0.4335	0.461	0.000
MLP	2.155	0.000	0.0536	0.2588	1.784
RF	1.864	0.449	0.000	0.207	1.552
KNN	1.685	0.4738	0.0417	0.000	1.434
KM	0.658	1.2305	0.3621	0.3848	0.000

Table 40: Mean conditional ranking of classification models given the best model with mean index 0.

It should be noted that in the scenario where more than one model performs best, these both get assigned a ranking of zero and the next best model after those gets a ranking of two. That is, a model is still the third best model is two models before it are both ranked first.
We convert these to conditional rankings to rankings:

	EM	MLP	RF	KNN	KM
EM	1	5	4	3	2
MLP	5	1	2	3	4
RF	5	3	1	2	4
KNN	5	3	2	1	4
KM	4	5	2	3	1

Table 41: Ranking of mean conditional ranking of classification models given the best model with mean index 0.

Applying the same optimization procedure as for regression testing, the optimal strategy for classification is given by: K-means, multi-layer perceptron, random forest, k-nearest neighbors and expectation Maximization. We summarize the results in terms of accuracy in discovering the best model and the number of models needed:

Success Rate	Mean Models Used	Std Models Used
100%	2.402	0.495

Table 42: Conditional ranking strategy results for theoretical testing.

This yields the optimal strategies to be for regression: Polynomial regression, k-nearest neighbors, random forest, linear regression, multi-layer perceptron and elastic net regression. For classification this results in: K-means, multi-layer perceptron, random forest, k-nearest neighbors and expectation Maximization.
Furthermore, for classification and regression we provide the 95% confidence interval for the expected number of models needed:

	95% Lower Bound	95% Upper Bound
Classification	2.368	2.437
Regression	2.910	3.018

Table 43: Confidence intervals for expected number of models used.

By assuming the upper bound to hold true for the worst case scenario based on our theoretical testing, we have lowered the number of models used for classification by at least 48.7% and for regression by at least 50.3%. This results in a greatly reduced time complexity for the model selection process.

5.3 Model Selection for Aegon Data

In section 4.3, clustering was performed on the transformed training set if deemed necessary. Following clustering, feature selection was performed through three considered methodologies and the filtered

features were passed to this Model Selection node. The Model Selection node aims to construct various models to determine the model that is best able to restore missing data on the target feature based on the training data. In this section we shall focus on the features selected as per the correlation coefficient and treelet filter. This means that our input features for our models will consist of the $Buffer$, $Used$, $Active\_1$ and $Total$ features.

In section 5.2, based on theoretical testing, we determined the optimal ordering for the case we wish to restore numeric data and categorical data. If we once more return to the data setting in section 4.3, we note that our target feature was deemed to be numeric in section 3.3. As such the optimal ordering for model construction will consist of first generating polynomial and nearest neighbor regression models. After which random forest regression is considered, followed by linear regression.

The first model we wish to construct is that of polynomial regression. For training the polynomial model we will make use of the training set. Using this training set the polynomial model will aim to find the polynomial relation of specified degree using the filtered features from section 4.3 that best approximates the target feature. Afterwards we will provide the filtered and transformed features of the test data set and enter these into the constructed model. The output of this model is then compared to the actual values as presented by the values of the *Residual* target feature of the test set. The mean absolute percentage error is used to determine the desired score metric as proposed in section 1. That is, if a prediction of a *Residual* value in the test set is within the 95% to 105% interval of the true value within the test set, we deem that value to be successfully restored and return value 1. Summing over all of these values and taking the average thus provides us with the restoration rate of this model, in that is the score returned is 0.99, we have achieved a successful restoration on 99% of test data. For the determination of the polynomial relationship, we will consider polynomials of degrees 2, 3, 4 and 5. We shall consider in detail the second order case.

The first step in fitting the second order polynomial to our training data, involves transforming the selected features up to their second order terms. In the case of our four input features, this will result in 15 new polynomial features, an overview is provided of these features:

Polynomial Feature
1
Active_1
Used
Buffer
Total
Active_1 * Active_1
Active_1 * Used
Active_1 * Buffer
Active_1 * Total
Used * Used
Used * Buffer
Used * Total
Buffer * Buffer
Buffer * Total
Total * Total

Table 44: Second order polynomial features.

From the table the methodology behind constructing higher order features is clear. We start with the intercept, which can be considered a coefficient of order 0, after which we determine the first order features in the order of acquisition by the algorithm. Finally, we second order terms are determined in the order of acquisition. Next, the closed form solution with respect to the mean square error of the linear regression over these features is used to determine the optimal value of coefficients for these features with respect to the target feature. These results are provided below:

Polynomial Feature	Optimal Coefficient
1	300926440.0
Active_1	-872573875.0
Used	-108558021.0
Buffer	321548426.0
Total	-193969657.0
Active_1 * Active_1	645360943.0
Active_1 * Used	571647435.0
Active_1 * Buffer	-36492.986
Active_1 * Total	-321548426.0
Used * Used	549022.524
Used * Buffer	-645360943.0
Used * Total	-192368419.0
Buffer * Buffer	-321548426.0
Buffer * Total	193969656.0
Total * Total	-645360943.0

Table 45: Optimal coefficients of second order polynomial features.

Using the model based on these coefficients, we are now able to test our model on our test set. First, the test set will be transformed to second order polynomial features as was performed on the training data set. Then the input features are used with the coefficients from table 45 to provide predictions for our *Residual* target feature. For a sample of 20 data points from the entire test set these results are provided in the following table, in this table we have undone the transformation from section 3.3:

Prediction	True Value	MAPE
52349.15	52268.10	0.00
32354.25	31198.36	0.04
46840.23	47031.68	0.00
62118.28	62120.23	0.00
43476.38	43043.86	0.01
51326.39	51205.92	0.00
40786.87	40173.01	0.02
35709.83	38491.66	0.07
39266.43	38563.63	0.02
33225.28	32135.73	0.03
60120.54	60128.08	0.00
50876.52	50701.59	0.00
36250.17	35356.82	0.03
34570.47	33548.84	0.03
64298.29	64300.65	0.00
57367.99	57335.32	0.00
80879.30	80225.09	0.01
64283.35	64415.85	0.00
54441.01	54589.07	0.00
53142.57	53093.90	0.00

Table 46: Polynomial model predictions and mean absolute percentage errors on test data versus true values of test data for target feature.

From the results provided in the table, we notice that the predictions seem close to the true values of on the test data. This is further confirmed when analyzing the corresponding MAPE values for these predictions. We deem a data point to be restored in the test set, if the MAPE does not surpass the 5% mark. From this small sample of 20 test data points, only one value does not adhere to this bound and thus if we would consider the restoration rate of this sample, it would be 95%. However, there are many

more values in the entire test set as seen in section 4.3. On the entire test set, the restoration rate of this model was deemed to be 99.95%.

The next step in the model selection phase is to construct polynomial models for the other higher order degrees and compare the model performances on the test data. The restoration rates for all polynomial models considered is provided below:

Order	Restoration Rate
2	99.95%
3	99.95%
4	99.95%
5	99.95%

Table 47: Restoration rates on test data per order of polynomial model.

In this specific data setting, in which the relationship between features exhibits no signs of noise, all polynomial models will return the same restoration rates. Based on these restoration rates, the Cleansing Algorithm now deems the second order polynomial model to be the best model as it has the least number of degrees of freedom and the greatest restoration rate.

Following the construction of the polynomial model, we also construct the nearest neighbor model on the training data and consider the variants of nearest neighbor model with parameter k set to 1, 5, 10 and 20. Similarly to the polynomial model, we generate predictions for test data and compute MAPE values for each prediction with respect to the true value. Once more, we present the predictions versus true values for the nearest neighbor model with parameter set to 1:

Prediction	True Value	MAPE
51528.18	52268.10	0.01
32495.07	31198.36	0.04
47666.24	47031.68	0.01
62561.18	62120.23	0.01
43026.10	43043.86	0.00
51084.21	51205.92	0.00
40096.37	40173.01	0.00
39569.26	38491.66	0.03
38468.35	38563.63	0.00
32956.69	32135.73	0.03
60622.26	60128.08	0.01
50511.60	50701.59	0.00
36194.11	35356.82	0.02
34192.11	33548.84	0.02
64127.00	64300.65	0.00
57292.79	57335.32	0.00
68823.37	80225.09	0.14
61539.16	64415.85	0.04
54104.49	54589.07	0.01
53818.70	53093.90	0.01

Table 48: Nearest neighbor model predictions and mean absolute percentage errors on test data versus true values of test data for target feature.

From the sample data, we note that the restoration rate would be the same as that for the second order polynomial model, being 95%. However, we do note that the one value that does not have an MAPE less than 0.05, has a greater deviation when compared to the results in table 46. This may occur due to the fact that the nearest neighbor model with parameter set to 1, tends to be very sensitive to noise or outliers. In order to verify whether this is a common occurrence within the predictions, we shall

provide the overall model evaluation as well as those for the other parameter settings for the nearest neighbor model.

No. Neighbors	Restoration Rate
1	99.12%
5	99.53%
10	99.53%
20	99.64%

Table 49: Restoration rates on test data per parameter setting of nearest neighbor regression model.

Contrary to the results we observed from the construction of polynomial models, the nearest neighbor models do not all perform exactly the same. Though all recorded restoration rates are very high, the Cleansing Algorithm will opt to use the nearest neighbor model with neighbor parameter k set to 20 as this yields the greatest restoration rate.

Once the best model in terms of restoration rates has been discovered for both the polynomial and nearest neighbor models, their scores on the test set are compared. In this data setting, the polynomial model outperforms the nearest neighbor model in terms of restoration rate and thus the model selection strategy deems the polynomial model the best model we will be able to find based on theoretical testing.

To confirm our findings in determining the best model selection strategy, we shall also construct the random forest and linear models for this test case. The random forest model has a parameter that determines the number of estimators it will use. In the Cleansing Algorithm we consider the settings for this parameter of 10, 100, 1000 and 10000. The restoration rates for this model are presented in the following table:

No. Estimators	Restoration Rate
10	99.64%
100	99.69%
1000	99.69%
10000	99.69%

Table 50: Restoration rates on test data per parameter setting of random forest regression model.

These results show that the greatest restoration rate that the random forest models were able to achieve, was 99.69%. This restoration rate is greater than the best restoration rate observed for the nearest neighbor models, but does not outperform our polynomial model.
Finally, we present the restoration rate for the linear regression model. This model does not have hyperparameters to tune and as such concerns but one model.

Model	Restoration Rate
Linear	93.59%

Table 51: Restoration rate on test data of linear model.

Once more, the restoration rate of the linear model does not outperform the restoration rate achieved by the best polynomial model. As such we have shown that the model selection strategy was able to determine the best model based on theoretical testing. The model selection strategy was able to achieve this by constructing only two main models, whereas no model selection strategy would have led to the construction of four main models not including MLP and ElasticNet models.

In order to present the capabilities of the Cleansing Algorithm to handle categorical features, we will consider a new target feature to be that of *Active* from the original data set. We will not discuss the transformation and structure analysis for this new target feature as these follow the same principles as presented in sections 3.3 and 4.3. Rather, we will move to the construction of models for this classification case. In section 5.2.6 we determined the optimal model selection strategy to be given by the following

model ordering: k-means, multi-layer perceptron, random forest, k-nearest neighbors and expectation Maximization. As such the model selection node starts by constructing a k-means model for the classification of our target feature. For the k-means model there are no parameters to vary as the number of centroids k is chosen such that it is equal to the number of unique classes that are present in the *Active* feature. In this case this parameter is set to 2. However, the initialization of the centroids is chosen at random and as such the k-means model will be constructed 10 times with different initialization. This then results in the following restoration rates:

Initialization	Restoration Rate
1	49.66%
2	49.66%
3	49.66%
4	49.66%
5	49.66%
6	49.66%
7	49.66%
8	49.66%
9	49.66%
10	49.66%

Table 52: Restoration rates on test data per unique initialization of k-means model.

In all ten initialization from the k-means model, we acquired the same restoration rate of less than 50%. This is not a great result as it was shown that the value of Y in the *Active* feature occurred approximately 10% of the time on the training data and the value N had a frequency of the other 90%. A naive model could select the value N to be chosen in all cases and achieve a restoration rate of somewhere closer to 90%.

The next model we consider is the multi-layer perceptron for classification. For this model the main parameter is the hidden layer structure of the underlying network. Based on Introduction to Neural Networks for Java (second edition) by Jeff Heaton [44], it is said that a neural network with no hidden layers only capable of representing linear separable functions or decisions. A network with one hidden layer can approximate any function that contains a continuous mapping from one finite space to another. On the other hand, a network with two hidden layers can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. Based on this, we shall consider one network with no layers, opt for five networks with one layer and two with two layers. Jeff Heaton [44] goes on to say that the number of neurons in the hidden layers should be between the number of input and output features, it should be equal to two thirds of the number of input features plus the number of output features, and, finally, the number of hidden neurons should not be greater than two times the number of input features. In accordance with Heaton [44], we will consider a model with no hidden layers. We will consider a 1-layer model such that the number of hidden neurons passes the rule-of-thumb methods as stated by Heaton [44] and a 2-layer model that also adheres to these rules. To challenge Heaton's rules-of-thumb [44], we will propose the use of a one-layer model with 10, 100 and 1000 hidden neurons and a 2-layer model with 10 hidden neurons in each layer. Furthermore, the maximum number of iterations allowed is set to 1000 and the stopping criterion for the error is set to 10^{-20}. The restoration rates for these architectures are presented in the following table:

Architecture	Restoration Rate
()	86.72%
(3)	97.11%
(10)	98.45%
(100)	100.00%
(1000)	100.00%
(2, 1)	84.29%
(10, 10)	100.00%

Table 53: Restoration rates on test data per hidden layer architecture of multi-layer perceptron model.

From these findings, it is clear that the multi-layer perceptron gained the best restoration rates on the architectures that do not adhere to the rule-of-thumb. All of these architectures outperformed the rule-of-thumb architectures, reaching up to a perfect restoration rate for three of these architectures. In theory, the Cleansing Algorithm could stop generating other models as it is not possible to outperform the best MLP model restoration rate, however, the model selection process indicates that the random forest model should at least be considered in comparison to the best MLP model. As such we once more generate a random forest model, this time for the purpose of classification. These results are presented below:

No. Estimators	Restoration Rate
10	100.00%
100	100.00%
1000	100.00%
10000	100.00%

Table 54: Restoration rates on test data per parameter setting of random forest classification model.

The random forest model is also capable of achieving 100% restoration rate on the test data. All random forest settings achieve this result. As such the Cleansing Algorithm once more prefers the less complex model and will decide to use the random forest model with parameter set to 10. This means that the algorithm will continue the model selection process for the nearest neighbor classification model. These results are presented in the following table:

No. Neighbors	Restoration Rate
1	99.64%
5	98.50%
10	97.83%
20	97.26%

Table 55: Restoration rates on test data per parameter setting of nearest neighbor classification model.

None of the nearest neighbor models are able to match the result of the random forest and multi-layer perceptron models. For this reason, the Cleansing Algorithm will returns the random forest model with number of trees parameter set to 10 as the optimal model. Verification of this result is not necessary for this case, as a perfect restoration rate is achieved. However, for completeness we will present the results for the expectation maximization model, that in a similar fashion to k-means, has a fixed number of centroids set to the number of unique classes, being 2. Once more, the expectation maximization model is sensitive to initialization, so we consider 10 unique initializations. The results are presented below:

Initialization	Restoration Rate
1	48.48%
2	48.48%
3	48.48%
4	48.48%
5	48.48%
6	48.48%
7	48.48%
8	48.48%
9	48.48%
10	48.48%

Table 56: Restoration rates on test data per unique initialization of k-means model.

As was the case for the k-means models, the initialization of the expectation maximization models does not seem to influence the restoration rate. What is more, the restoration rate was exactly the same for all initializations, being 48.48%. This is far below the majority class guess as indicated for the k-means models. The expectation maximization does not outperform nor match the performance of the random forest and multi-layer perceptron models.

For the classification process of the model selection node with respect to the *Active* target feature, we reduced the number of main models needed for model selection from 5 to 4, though, theoretically, if model complexity is no factor, the process could have reduced this number from 5 to 3 by selecting the multi-layer perceptron model over the random forest model.

6 Results

6.1 Model Training and Testing Nodes

In this section we discuss the Model Training & Testing nodes as provided in the schematic representation of the Cleansing Algorithm.

Once the model with the greatest restoration rate is determined through the Model Selection strategy, we are able to start the actual restoration of missing data. Up until now the entire Cleansing Algorithm process has been concerned with analyzing the behavior of input features with respect to the output feature for which we wish to restore missing data. This analysis has been performed making use of those data points that are complete and do not contain missing or wrong values as determined by the Data Quality Checks. In the Model Training & Testing nodes, we aim to use these insights to generate suggestions for the actual missing values of the output feature in those data points that were determined to contain missing or wrong data in accordance with the Data Quality Checks.

The Model Training node takes inputs from the Data Transformation, Data Structure Analysis and Model Selection. From the Data Transformation node, we have access to the data containing actual missing values as determined through Data Quality Checks, it is these values that we wish to restore. From the Data Structure Analysis node, we have gathered information with respect to the clustering, where we know whether or not clustering was performed, which clusters were formed and based on which features those were formed and, finally, the features that were deemed relevant and non-redundant based on the filter feature selection process for the different clusters if any.

Model Training starts by taking the data with missing values and assigning them to the appropriate cluster based on the features and their boundaries that were determined during the Clustering section of the Data Structure Analysis node. Once it is known to which cluster the data point with missing values best belongs, we retrieve the features that were selected by the Feature Selection section of the Data Structure Analysis node. We then filter out all features not passed on by the Feature Selection process.

Once this is completed the data containing missing values enters the Model Testing node, where the model that is assigned to that cluster and those features is retrieved from the Model Selection node output. From here we load the model and generate suggestions for the missing values using the model and the input features from the data with missing values. After the suggestions have been generated, we gather the information from the Data Transformation node concerning the Transformation of Data section. We now wish to undo this transformation process, by first recognizing whether the output feature was numeric or categorical. In the case it was categorical, we undo the one-hot encoding. This is done by assigning the category of which one of the one-hot encoded features is equal to one. In the case the output feature was numeric, we undo the normalization by multiplying with the difference between the maximum and minimum values of the original feature and then adding back the minimum value of the original feature. After, if the numeric feature was originally made up of dates, we add the numeric value of each restored value to the earliest date found in the original feature. Finally, we recognize whether correctly missing data was imputed using zeros or negative ones and if a zero or negative one occurs, they are returned to their correctly missing state.

Once these nodes have completed their tasks, we are left with suggestions for missing data values. Furthermore, we have access to the features used from the Feature Selection section, we have access to the model used from the Model Selection node and have access to the restoration rate of the model used from the Model Selection node. This is all passed on to the Results node.

6.2 Results Node

In the Results node all information a business expert needs, is gathered and sent to a PowerBI dashboard. A snapshot of this dashboard is provided in the following figure:

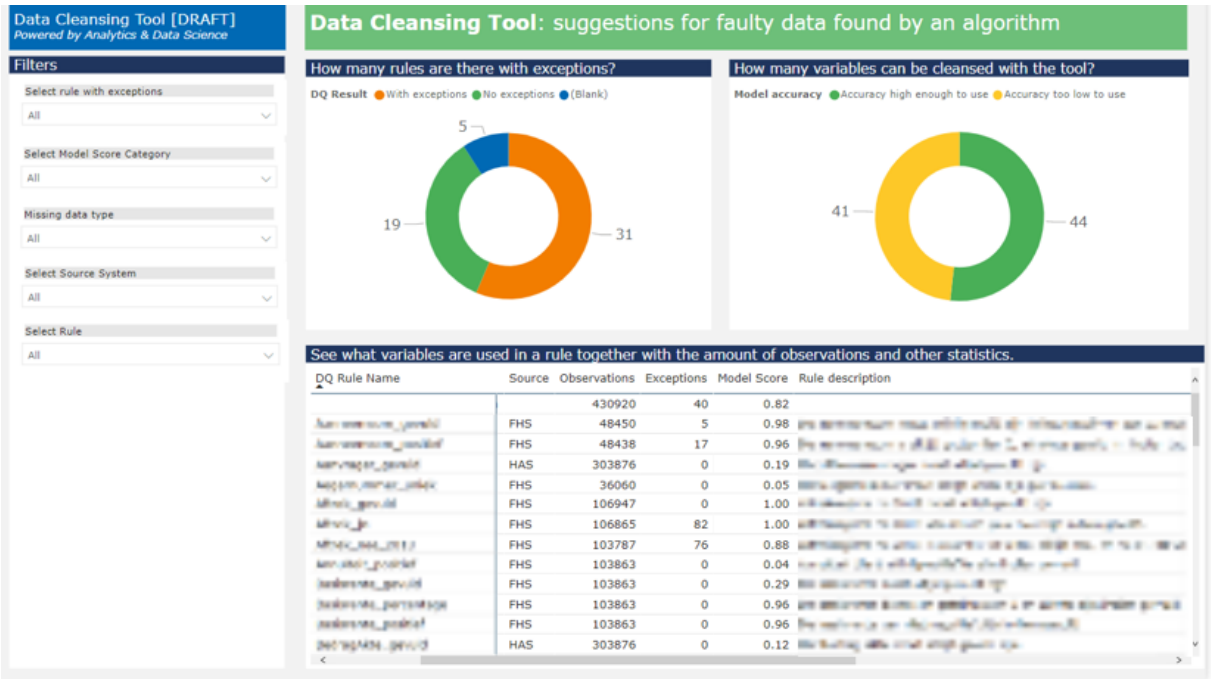


Figure 6.1: Dashboard for general overview of results from Cleansing Algorithm output.

As can be seen in figure 6.13, the business expert is provided with an overview of the number of Data Quality checks that do and do not contain exceptions in the ring chart in the top left of the figure. A data set corresponding to a Data Quality check is considered to contain exceptions if it contains missing/wrong data, which indicates that some values in the *DQ* feature have returned a score of 0. The ring chart on the top right provides an indication of the number of models generated by the algorithm that are deemed to have an adequate accuracy for restoration and those not. An accuracy high enough is achieved once the thesis definitions are met. That is, for categorical variables, a model returned a test score with 95% accuracy. For numeric variables, a missing value is considered restored when a model returned a test score with 95% accuracy based on a 95% mean absolute percentage error. Below the two ring charts, an overview is provided of the name of the Data Quality check, the data base it was sourced from, the number of observations in the data set, the number of exceptions, the score of the generated model and the description of the Data Quality check.
With this panel in the dashboard a business expert is provided with a clear and concise overview of the general status of their data bases with respect to the data quality, as well as possibilities for restoration.

In another panel of the PowerBI dashboard, we provide and overview of all target features associated to the Data Quality checks, providing general statistics of these features.

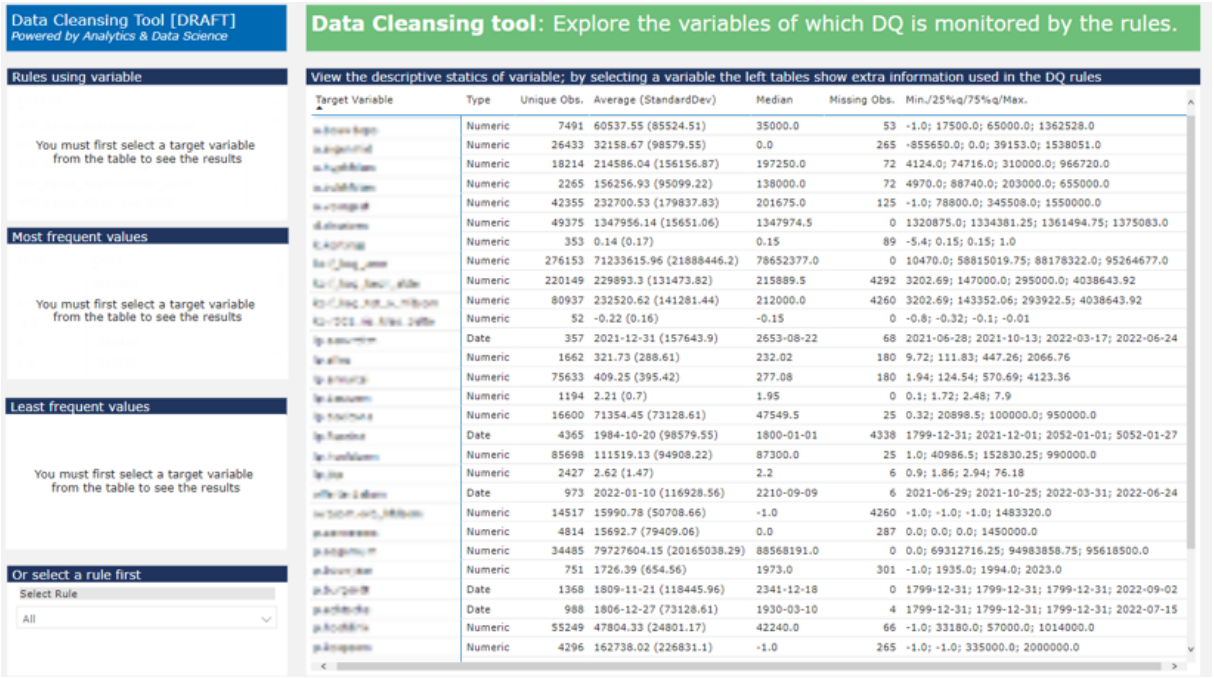


Figure 6.2: Dashboard for general statistics for target features from Cleansing Algorithm output.

That is, in figure 6.14 we provide the type of feature it concerns, the number of unique observations within the target feature, the average and standard deviation, the median, the number of missing observations and the minimum, maximum and first and third quartile values. It is this information that a business expert can consult to check whether the output generated by the Cleansing Algorithm is adequate or not.

If one is to select one of the target features in figure 6.14, one is provided with the following overview:

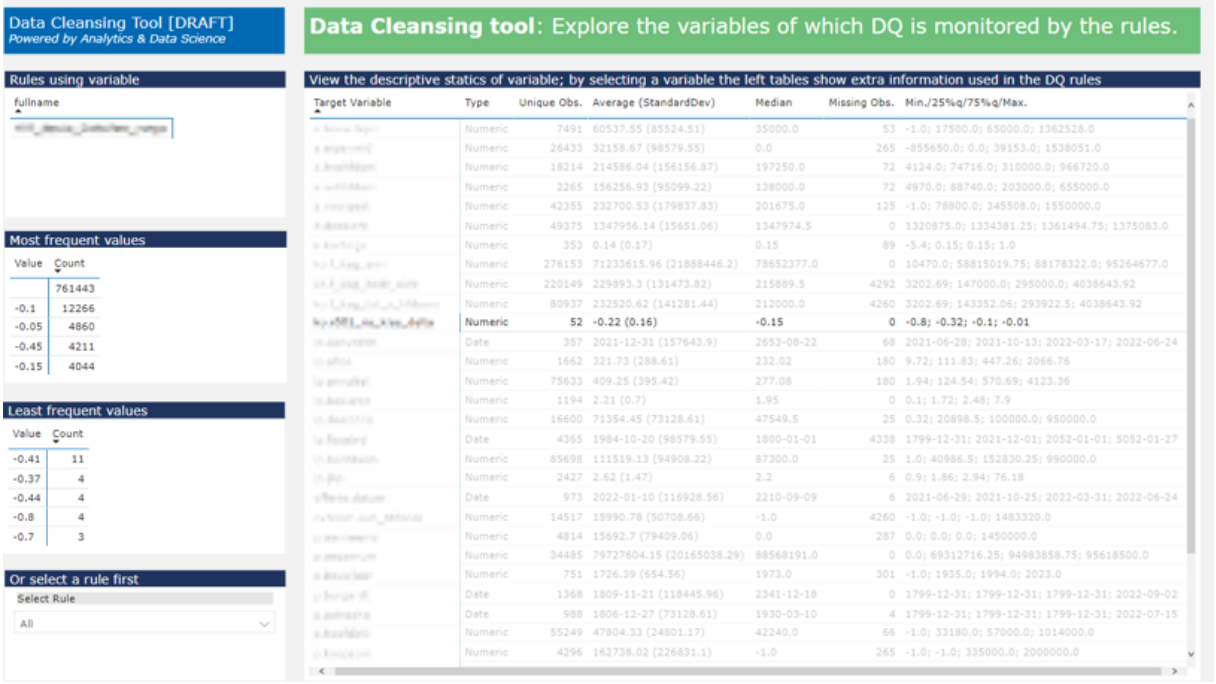


Figure 6.3: Dashboard for general statistics for target features from Cleansing Algorithm output.

Now the business expert is provided on the left of figure 6.3 with an overview of the Data Quality check in which this target feature occurs, the five most frequent occurrences of values within the target feature and the five least frequent occurrences. These occurrences all are provided with their respective counts.

The final panel of the current Data Cleansing Tool dashboard is provided in the following figure:

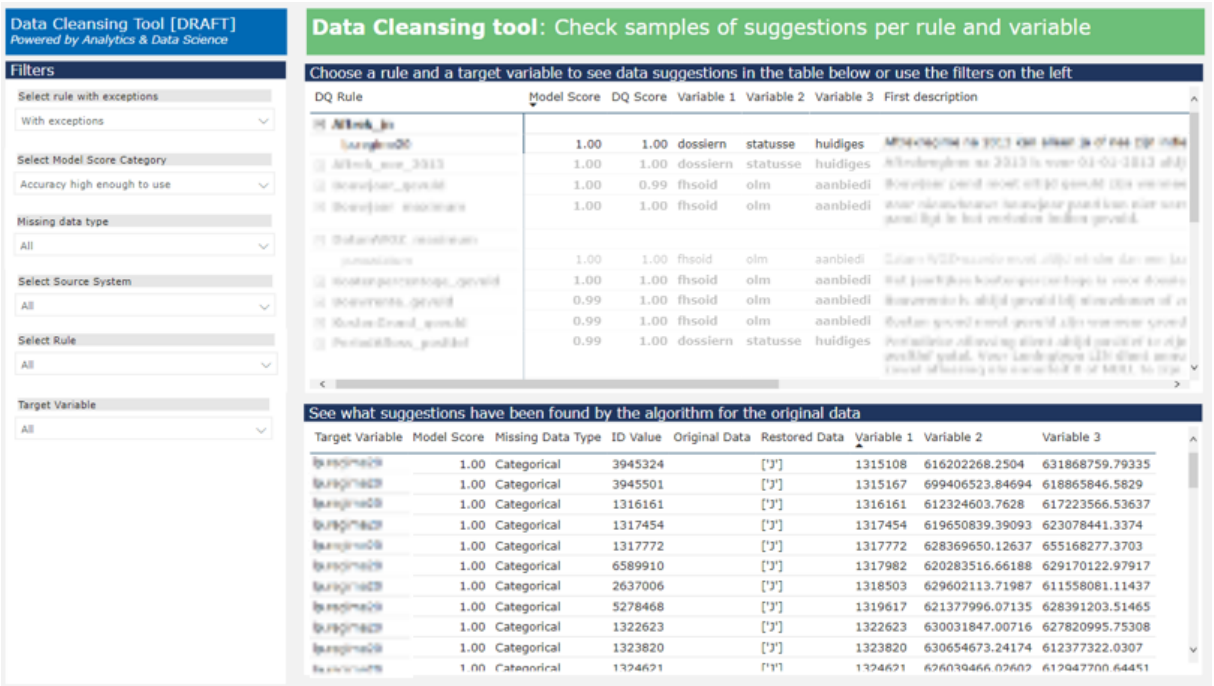


Figure 6.4: Dashboard for restoration suggestions for target features from Cleansing Algorithm output.

It is in this panel that the business expert is provided with actual restoration suggestions for missing data. The user of the dashboard is able to select which subsection of Data Quality checks they wish to analyse, such as whether or not they contain exceptions, whether or not the accuracy is high enough for use, the missing data type, the source system of the data base and specific Data Quality checks and target variables are also possible. Once a selection has been made, the user is provided with the overview on the right side of figure 6.4. In which, for a specific Data Quality check a target feature can be selected. For all target features within Data Quality checks, the model and Data Quality scores are provided, as well as three variables deemed important for the restoration of missing data in said target feature. In the bottom right of the panel in figure 6.4, the user is provided with an ID value of specific entries in the data set that need to be restored. The original values of the target feature is provided as well as the restored value in their respective columns.

6.3 Feedback Node

The final step in the underlying processes of the Cleansing Algorithm concerns itself with feedback. Currently the Cleansing Algorithm is able to handle feedback with regards to the restoration suggestions, as to whether these are deemed correct or incorrect by a business expert. In the case they are deemed incorrect without further information, we restart the Cleansing Algorithm with a different configuration. This can be any choice between not performing clustering if it was performed, selecting one of the other feature selection approaches or removing models from the set of candidate models. It is also possible to provide more specific feedback. If it is indicated that clustering is performed and a business expert deems this inappropriate, then the Cleansing Algorithm shall go through the process again, this time not allowing for clustering. Alternatively, if it is indicated that the features used are inappropriate, the

Cleansing Algorithm starts over trying a different feature selection procedure. If it is indicated that the model used for restoration is inappropriate for the task at hand, the Cleansing Algorithm shall restart, omitting that model from the set of candidate model for the Model Selection node.

Finally, we recognize, due to the node structure the Cleansing Algorithm is clearly modular. This means that someone can easily insert a new feature selection approach or a new model into the Cleansing Algorithm, where if the algorithm is run, the entire process is updated and calibrated to handle these new implementations.

It should be noted that prior to the construction of the Cleansing Algorithm restoration of missing data was generally not possible for Aegon. Due to the time constraints associated to having to restore missing data manually, it was deemed impractical. In this manner the Cleansing Algorithm has provided Aegon with a means to automate and accelerate the process of restoring missing values within the Aegon databases. It is expected that use of the Cleansing Algorithm will, as such, lead to a greater Data Quality for Aegon. Externally, a greater Data Quality will help Aegon adhere to restrictions as imposed on them by the DNB. Internally, the improved Data Quality can further improve business processes and customer support. Apart from this, the manner in which the Cleansing Algorithm is constructed allows for further improvements. The modular construction in the form of nodes discussed through this work allows for human adaptations by addition and removal of features such as approaches in feature selection, clustering and models. Furthermore, the model selection approach allows for addition and removal of data tests and distributions. It this construction in combination with the already implemented methodologies and feedback node that allow for the Cleansing Algorithm to adapt to human input.

6.4 Results on Aegon Data

6.4.1 Performance of feature selection

Testing the performance of feature selection within the algorithm is a difficult task on the Aegon data sets. This is due to the fact that relations between features are unknown prior to restoration. As such testing for the performance of feature selection within the Aegon data sets comes down to comparing the algorithms restoration performance when performing no feature selection to the setting where we do treelet based feature selection, correlation coefficient based feature selection and fast correlation-based filter feature selection methodologies. Provided in the figure below are the results with respect to number of features that each feature selection methodology passes through to the rest of the models for restoration. These are the results of the first 20 data sets.

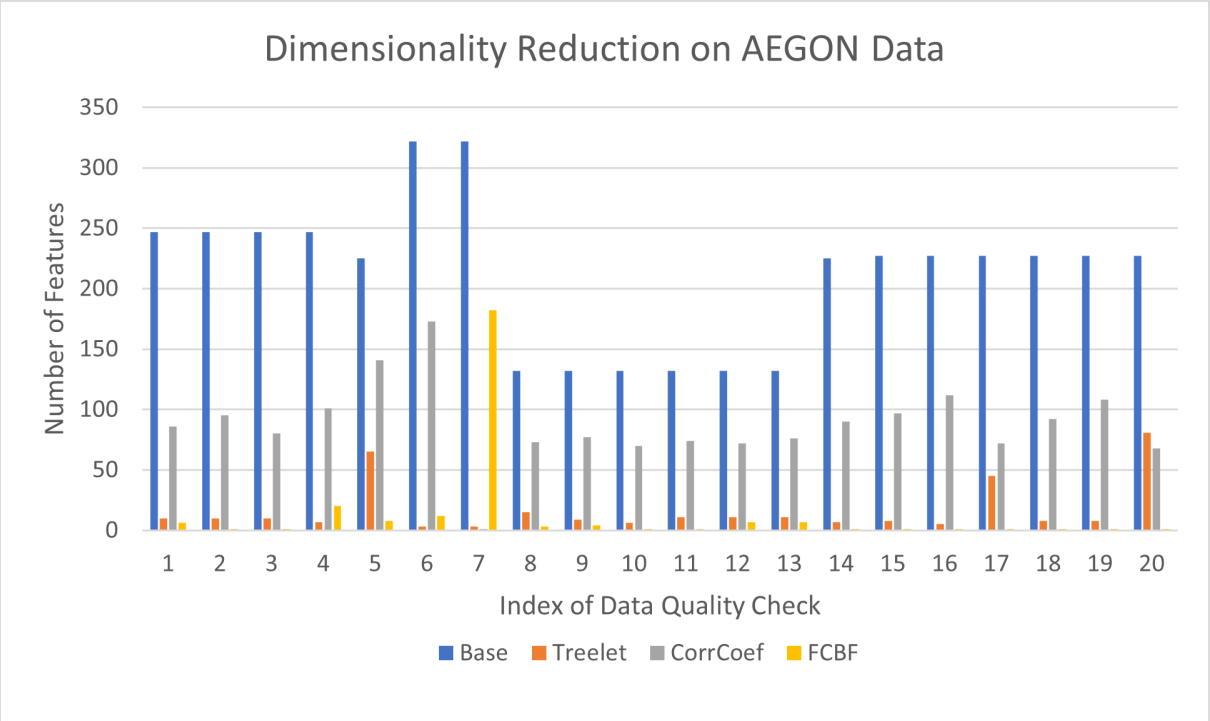


Figure 6.5: Comparison of number of features for feature selection by Treelet, Correlation Coefficient and FCBF.

In figure 6.5 in blue is the original number of features, in orange the number of features after pre-processing and treelet feature selection, in grey the number of features after correlation coefficient feature selection and in yellow the number of features for the fast correlation-based filter.
As seen, all three feature selection approaches significantly reduce the number of features. For this subset of data sets, we provide the maximum, minimum and average percentage decreases of the number of original features in table 57:

	Treelet	CorrCoef	FCBF
Minimum	64.31%	37.33%	43.48%
Mean	91.76%	56.00%	95.22%
Maximum	99.07%	99.68%	99.60%

Table 57: Overview of minimum, maximum and mean percentage dimensionality reduction on the data set in figure 6.5.

The minimum percentage of the original dimensionality of the data sets for this subset was 37.33% achieved by the correlation coefficient filtering for data set with index 5. Likewise, the maximum per-

centage decrease of the original number of features for these data sets was achieved by the correlation coefficient filtering for index 7 and was 99.68%. For this maximum dimensionality reduction case, the Correlation Coefficient filter reduced the number of features to 1, as did the fast correlation-based filter for its maximum reduction, but for indices 2 and 3. The maximum reduction for the Treelet approach led to 3 features remaining for the data set corresponding to indices 6 and 7.

In order to get a better understanding of what these extremes in dimensionality reduction indicate, we once again analyze their corresponding data quality checks. The first two indices are governed by the data quality check we have seen in the section on the Aegon dataset:

$$\text{If class} = A \text{ and value} \neq 0, \text{ then DQ} = 0$$

$$\text{If class} \neq A \text{ and value} \leq 0, \text{ then DQ} = 0$$

$$\text{Else, then DQ} = 1$$

That is, in the data set with index 2, the feature "value" contains missing values, whereas for the data set with index 1 the feature "class" contains missing values. The results of the feature selection for these two indices are provided in the following table:

	Base	Treelet	CorrCoef	FCBF
Dataset 1	247	10	86	6
Dataset 2	247	10	95	1

Table 58: Number of Features for no feature selection (Base) and the Treelet, CorrCoef and FCBF

From this table it is clear for these datasets the correlation coefficient approach is very modest with respect to reducing dimensionality, a great number of features pass through with a p-value less than 5% and at least a coefficient score of 10%. This may be due to the fact that for the Spearman rank correlation coefficient a lot of features are selected. As we know from the real-world indexing always is increasing and monotonic over time and so are most features with a monetary basis. The Spearman correlation coefficient may find that the monotonic relationship is there for all these types of features, even though not all these features yield new information with respect to the target feature.
For the Treelet feature selection approach, we observe that it has found the same cluster containing our target feature, this indicates that both the "class" and "value" features from the data quality rule are found within the direct cluster output from Treelet, rather than being artificially injected into the cluster through the calibration step in our cleansing algorithm.
Finally, for the FCBF feature selection approach, we notice that the number of features selected is lower than both other methods, even resulting in only one significant feature for the case where "value" is our target feature. This teaches us that with respect to FCBF feature selection methodology there is no more information contained in the data set with respect to the target feature "value" other than the feature "class", indicating that restoration of this missing value may only be possible through the letter representing its class. This will of course result in a model that returns zero whenever the class is A and for both classes B and C the best value possible for restoration would be some constant value, being the mean of all data points within each class.

Just judging by the number of features selected by each methodology is not enough to determine which feature selection approach works best for restoration on the data set. It may be that the very strict feature selection approach by the fast correlation based filter is too strict and does not take into account all features that contain information with respect to the missing feature. Alternatively, the Correlation Coefficient approach may be too modest and actually reduce restoration accuracy by taking into account many features that do not yield additional information with respect to the missing feature. We will need to also analyze the performance of the Cleansing Algorithm with respect to the different dimensionality reduction approaches. For the same datasets the results of the cleansing algorithm are given:

	Treelet	CorrCoef	FCBF
Dataset 1	99.60%	99.15%	98.60%
Dataset 2	96.30%	97.20%	95.90%

Table 59: Restoration Rates for Treelet, CorrCoef and FCBF

As we can see, all models achieve restoration rates that are deemed sufficient for these missing data features to be considered restorable. However, it should be noted that the occurrence rate of class A and thus value 0 in the test set is 95.75%. As such it is not deemed difficult to achieve the restorable status and to better compare the models restoration rates, we consider the restoration rates with respect to classes B and C:

	Treelet	CorrCoef	FCBF
Dataset 1	90.59%	80.00%	67.06%
Dataset 2	12.94%	34.12%	3.53%

Table 60: Restoration Rates for Treelet, CorrCoef and FCBF for the subset of classes B and C

From this table, the difference in restoration rates becomes much more significant with respect to the minority classes B and C. Clearly, the Treelet based feature selection with its greater number of features relative to FCBF, is able to capture significantly more correct restoration on classes B and C, indicating that one or more features captured by Treelet, that are not captured by FCBF do in fact contain significant information with respect to the distinction between classes B and C. Furthermore, the much greater number of feature passed by the Correlation Coefficient based approach, seems to capture more information with its feature selection than the FCBF approach, but the sheer amount of features more than Treelet, is shown to reduce the models accuracy in restoring data points of these classes.
On the other hand, for the restoration of the "value" feature, we notice that once again, FCBF is not able to capture the impact of features on the minority classes B and C, whereas Treelet and Correlation Coefficient approaches are able to capture more information. For this dataset it seems to be the case that the Treelet algorithm is missing some features that are significant with respect to the information they contain for the "value" feature compared to Correlation Coefficient methodology. This is no revelation as the Treelet algorithm has a stricter bound on clustering with respect to a correlation of 0.30, whereas Correlation Coefficient captures all features with correlation of at least 0.1 and significant p-value. For this specific setting, the influence of features with minor Pearson correlation can be greatly significant for the restoration of the target feature.

We will look at another data quality check, the one indicated by index 18 in the previous figure. The data quality check associated with this data set is given below:

$$\text{If residual} \geq \text{total} - \text{used, then DQ} = 1$$

$$\text{Else, then DQ} = 0$$

For this data quality check, we notice a more direct dependency between target feature and other features. In this example, "residual" is the target feature, as such we can determine its value up to some accuracy by taking the difference between the "total" and "used" features. If "total" is the target feature then it can be determined using the summation of "residual" and "used" features. In this case the check concerns three possible features, leading to three data sets. We will give the results in the following table for the feature selection methodologies:

	Base	Treelet	CorrCoef	FCBF
Dataset 18	227	8	92	1
Dataset 19	227	8	108	1
Dataset 20	227	81	68	1

Table 61: Number of Features for no feature selection (Base) and the Treelet, CorrCoef and FCBF

Here we notice once again the FCBF reduces the input features to 1, whereas the actual check concerns itself with at least 2 other features. The Treelet algorithm gets a significantly reduced, but more modest 8 input features for dataset 18 and 19, representing target features for "residual" and "total" respectively. For data set 20, we notice that Treelet actually selects more features than our Correlation Coefficient based approach. This at first seems strange as the Correlation Coefficient method, in theory, should at least capture all features in the Treelet input. However, Treelet is able to cluster features that indirectly correlate as well. Let us assume some feature x exists and it is highly correlated with some feature y, the Treelet algorithm will decorrelated the two features, leaving the residual of decorrelating x with respect to y to be used for similarity comparison to other features or residuals. In this way Treelet is capable is producing more input features than the Correlation Coefficient methodology.
Once more we shall compare the restoration rates with respect to each clustering method and present the results in the following table:

	Treelet	CorrCoef	FCBF
Dataset 18	64.90%	83.45%	62.60%
Dataset 19	79.80%	80.55%	66.40%
Dataset 20	4.45%	8.45%	6.55%

Table 62: Restoration Rates for Treelet, CorrCoef and FCBF for datasets with index 18, 19, 20

Here we notice that the FCBF algorithm underperforms with respect to dataset 18, though Treelet only marginally outperforms FCBF, it is significant enough to indicate that FCBF is too strict in terms of feature selection. This is confirmed when further when we notice the superior restoration rate given by the Correlation Coefficient strategy for this dataset.
We see a similar situation for dataset with index 19. Correlation Coefficient strategy performs best, then Treelet and last is FCBF. However, in this case rather than Treelet and FCBF showing close restoration rates, for index 19, Treelet is closer to the Correlation Coefficient method.
Finally, dataset 20 concerning the "used" feature returns really low restoration rates comparatively to the other target features. This is due to the values of which the target feature consists. The target feature in dataset 20 on the whole takes on the value 0 for approximately 85.76% of cases, thus considering the relation in the data:

$$\text{total} - \text{residual} \leq \text{used}$$

For the target feature "residual", which takes on values much greater than 0 in order of magnitude at least 10^4, a similar relation exists. However, the unboundedness of this relation results in relatively less error in terms of restoration rates as the mean absolute percentage error is more lenient with respect to features of greater magnitude. For the target feature "used", with on average a value much closer to order 10^3, the unboundedness of the relation can lead to major error rates in terms of mean absolute percentage error, resulting in the low restoration rate for this target feature, relative to the other to target features in the same data quality check.

We summarize the results of this subsection:

- In terms of dimensionality reduction with respect to feature selection, the Fast Correlation Based Filter achieves the best results over the board, reducing on average the number of features by 95.22%. It is followed by the close second Treelet, which achieves an average reduction of 91.76% and finally, Correlation Coefficient approach reduction is given by 56.00%.

- In terms of restoration rates as a consequence of the feature selection methodologies, there is no clear winner. The Treelet algorithm and Correlation Coefficient methodology both outperform the Fast Correlation Based Filter in terms of restoration rates. On the one hand, Treelet is capable of outperforming Correlation Coefficient, if Correlation Coefficient selects too many features. On the other hand, Correlation Coefficient is more skilled at uncovering most features even containing only small amounts of information to improve the restoration of missing data. All-in-all, it seems that restoration rate is data dependent and the choice between the filters for feature selection should be done per data set.

6.4.2 Performance of clustering

The basis for the opted use of diffusion map and output clustering were observations and analysis of the Aegon data sets. In a significant number of features we noticed that the distribution became some amalgamation of a numeric feature and categorical feature through the data transformations performed on correctly missing values (figure 6.6). The aim of introducing these clustering methods thus had the main purpose to distinguish the subsets of the data that would lead to correctly missing values and those that do not exhibit this behavior in the output feature with missing values we wish to restore. In order to discover if the proposed clustering methods prove fruitful on the Aegon data sets we consider one example of a data set from the Aegon database. The distribution of the feature containing missing values we wish to restore is given in the following figure:

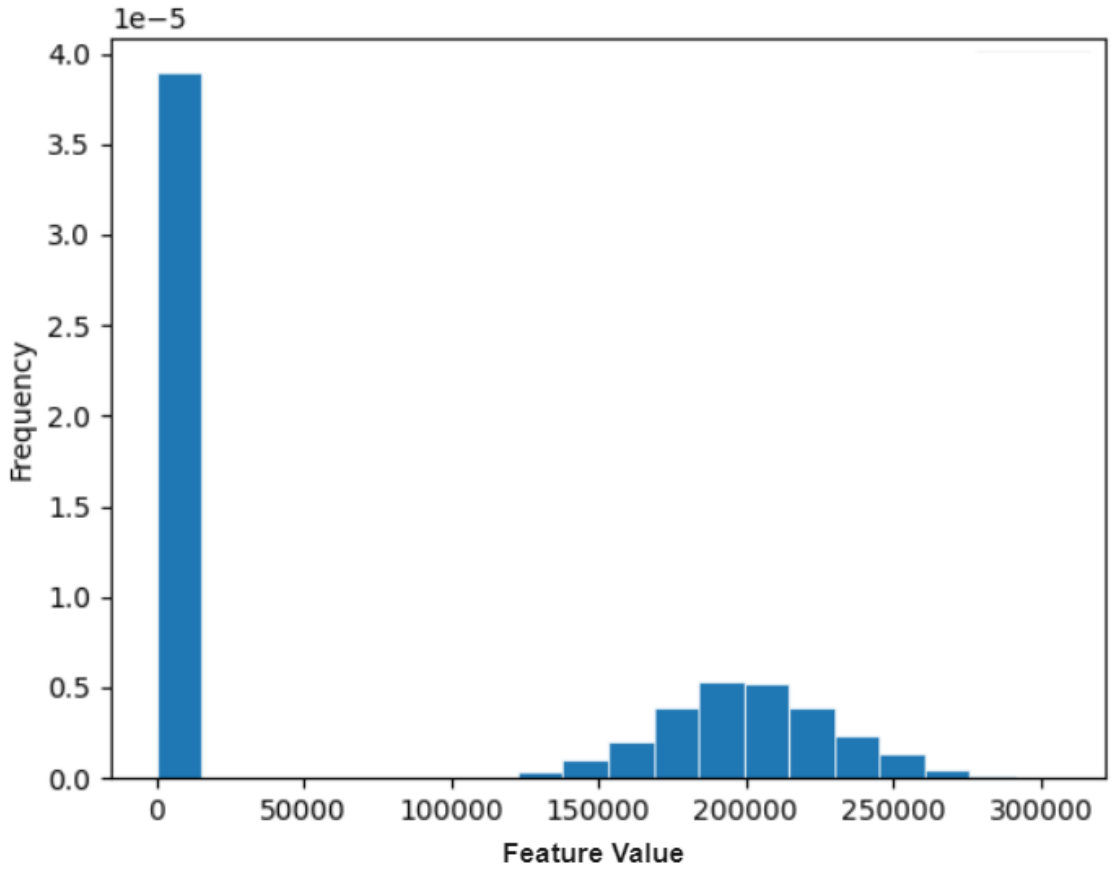


Figure 6.6: Histogram of density of feature containing missing values

As can be seen from the figure, transforming missing values to imputed zeros or negative ones can result in problems when normalizing the feature, as the imputed values may be far from the distribution of non-missing entries. Apart from that, it is relatively easy to get a high accuracy score as any model could just suggest the imputation of zero for every entry in this setting. To tackle this problem we generally wish to discover which features and possibly which conditions on certain features have led to this split in our target feature. To comprehend and prove what led to the distribution of this feature, we take a look at the accompanying data quality check for this feature:

$$\text{If class} = A \text{ and value} \neq 0, \text{ then DQ} = 0$$
$$\text{If class} \neq A \text{ and value} \leq 0, \text{ then DQ} = 0$$
$$\text{Else, then DQ} = 1$$

As can be seen from the data quality check for this feature, if the feature "class" is equal to the "A" class then a value in our feature not equal to zero results in a fail of the data quality check. If the "class" feature is not "A", then the value must be non-negative and greater than zero to pass the data quality check. From this observation it is clear that the "class" feature is the direct deciding factor as to whether a missing data value in the target feature should be 0 or not. To confirm the relation established in the data quality check, we provide a plot of the feature "class" against our target feature:

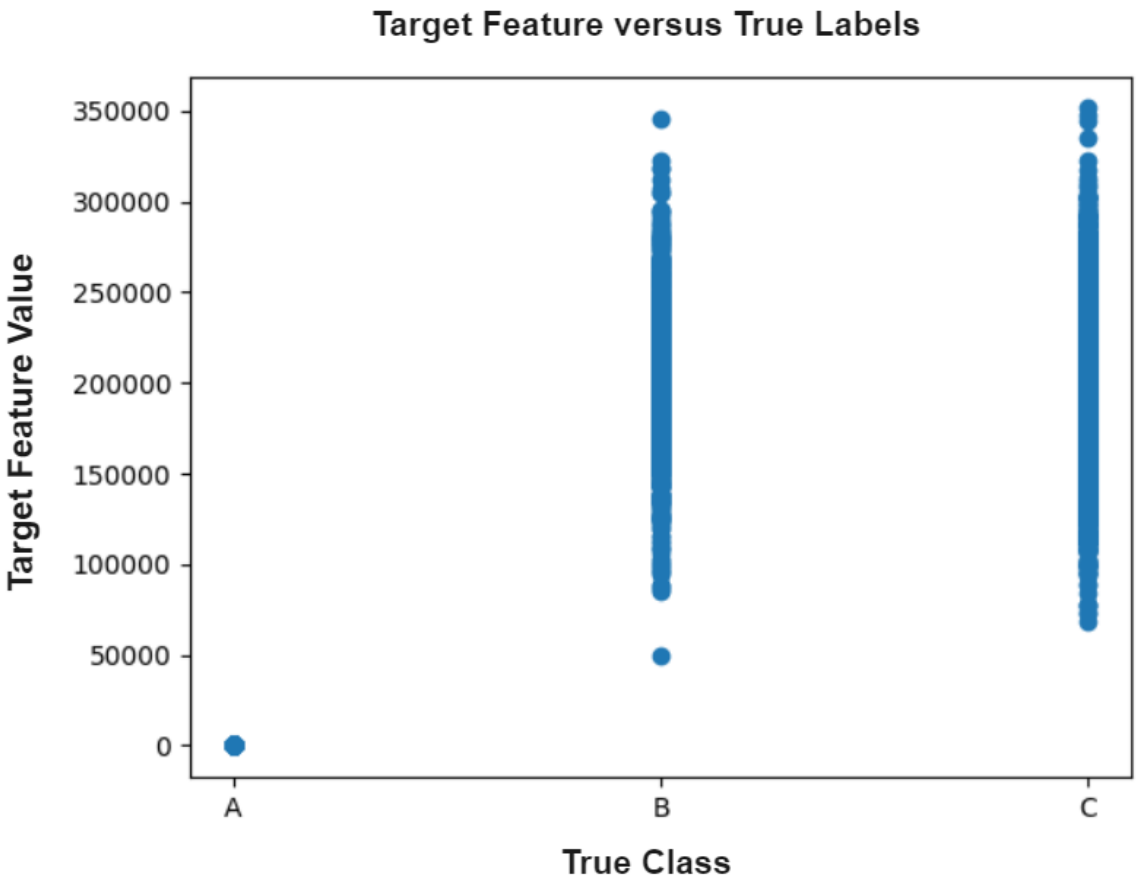


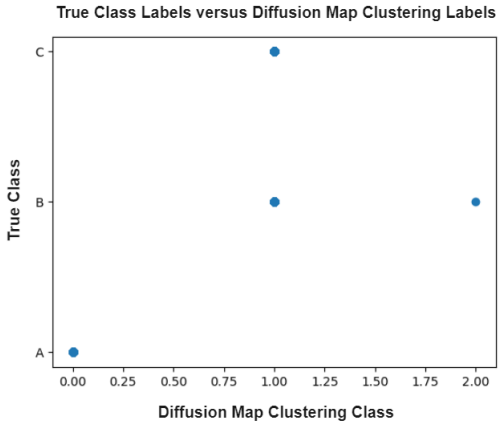
Figure 6.7: Plot of the feature "class" against target feature "value"

From this plot we can confirm that the first class indeed always leads to a value of zero, whereas the other two classes do not seem to be distinguishable from each other purely on the target feature. Having established this, we will analyze the performance of output clustering and diffusion map clustering by comparing their results with respect to this "class" feature and our target feature.

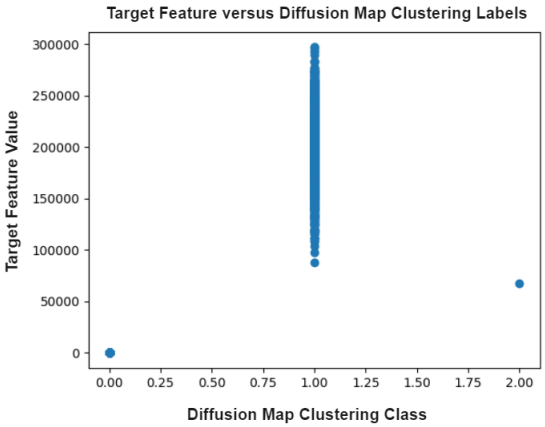
6.4.2.1 Diffusion Map Clustering

Though the Diffusion Map Clustering is performed on all features with respect to the target feature, for this dataset only the "class" feature led to clustering and as such only these results are provided. We provide a plot of the Silhouette score for K-means clusterings with $k = 2$ up to $k = 10$ for the Diffusion Map result using the Minkowski distance function. Next, we provide plots of the labels from the clustering output against the value of the "class" feature and our target feature. Finally, we provide a table with frequencies of each respective class label from this clustering.

(a) Silhouette score for each number of clusters in k-means.



(b) "Class" feature against class labels from k-means.



(c) Target feature against class labels from k-means.

Figure 6.8: Diffusion map clustering statistics

	Cluster 1	Cluster 2	Cluster 3
A	1.000	0.000	0.000
B	0.000	0.995	0.005
C	0.000	1.000	0.000

Table 63: Frequencies of classes per cluster from Diffusion Map Clustering

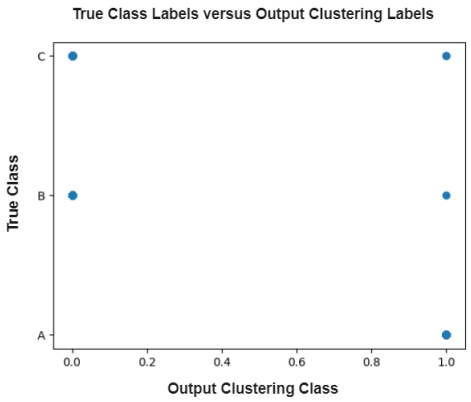
We shall start analyzing the Silhouette score for the various clusterings on the constructed Diffusion Map. From the figure it clearly suggests that the best number of clusters is either 2, 3 or 4, where three clusters slightly outperforms the other two over 10 instances of running each clustering due to randomness in the initiation of k-means. Analyzing the cluster labels plotted against the class labels, we see that class A is completely isolated from the classes B and C, which was the desired result. Furthermore, we notice that class C is completely encapsuled in the second cluster along with some of class B. We do notice, however, that the clustering of class B is not completely pure as some instances of class B have constructed their own third cluster. In order to understand this phenomenon, we look at the figure containing the plot of cluster labels against values of our target feature in combination with the class frequencies per cluster from the table. As we can see, a very small sample of the occurrences of class B have constructed their separate cluster. Analyzing the plot teaches us the reason: the value(s) of class B be that have formed a separate cluster form "light" outliers with respect to the second cluster and of

course is an outlier if we are to consider the first cluster containing only elements of class A. As such the Diffusion Map process as described previously will only construct a Markov chain to these outliers with very small probability resulting in the isolation of these data points.
We do note that though this can be a concern in the more general sense as we do not wish to construct separate clusters for each small outlier cluster, we do not blindly assume Diffusion Map clustering as it stands. This observation strengthens the necessity to only consider clusters containing some significant fraction of the data points considered and if it is not the case, they are not considered to be a separate cluster. By disregarding these data points, we would be removing some outliers from the data set.

6.4.2.2 Output Clustering

Output clustering is an example of clustering we only perform in the one-dimensional setting on the target feature. In doing so, we need not analyze or compute other clusterings except for the k-means on the target feature values. The results for the output clustering are provided below.

(a) Silhouette score for each number of clusters in k-means.



(b) "Class" feature against class labels from k-means.

(c) Target feature against class labels from k-means.

Figure 6.9: Output clustering statistics

	Cluster 1	Cluster 2
A	1.000	0.000
B	0.005	0.995
C	0.003	0.997

Table 64: Frequencies of classes per cluster from Output Clustering

One key difference we observe is that k-means directly on the output results in a clear favorite number of clusters being two. This of course would be the desired number of clusters in the ideal scenario, if we had a pure split between the classes A versus B and C. When analyzing the "class" feature against the cluster labelling, we notice that the clustering is not pure with respect to the ideal scenario. The first cluster does only consist of entries from classes B and C as desired, but the second cluster shows some impurity from both classes B and C into A. As was the case for the Diffusion Map clustering this impurity is very small, however, instead of dedicating the same occurrences to a separate cluster, output clustering adds them to the first cluster, resulting in a minor contamination of this cluster and subsequently on the use of this data.

6.4.2.3 DBSCAN

There is, however, one big caveat as to determine when clustering with respect to an target feature is necessary. Namely, it is impossible for us to just cluster based on K-means on some data set consisting of only one cluster. Furthermore, Silhouette scores cannot be used to compare the clustering of the case of one cluster compared to multiple clusters, as Silhouette scores require at least two clusters to be considered.
We shall consider a target feature, similar to the previous one, but without instances of class A. That is, the target feature has the following histogram:

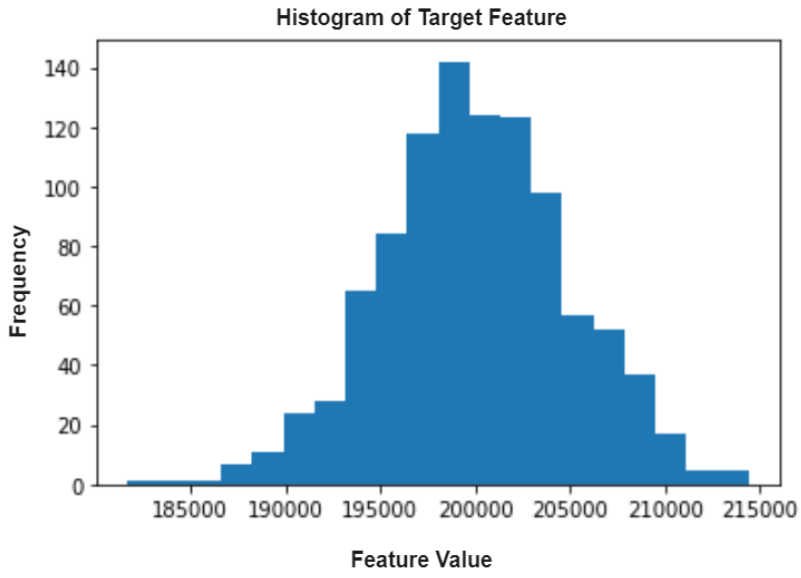


Figure 6.10: Histogram of data consisting of one cluster.

To analyze the problem with running the Output Clustering and Diffusion Map clustering approaches without determining whether it is necessary to cluster with respect to a target feature, we first provide the Output Clustering results for the target feature shown in 6.10:

(a) Silhouette score for each number of clusters in k-means.

(b) Clustering of target feature with one cluster.

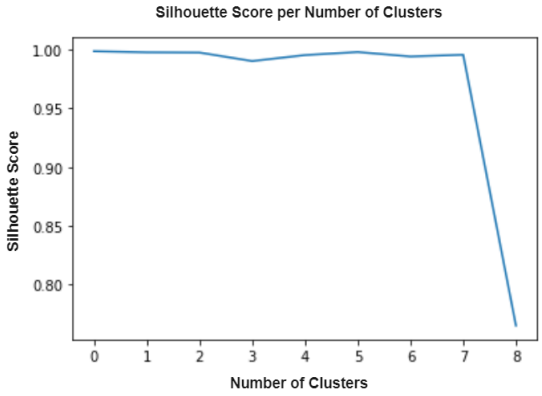
Figure 6.11: Output clustering statistics for target feature with one cluster.

The first observation we make is that the optimal value of the Silhouette score is much lower than those achieved in the setting where we did have a clear split with respect to class A in 6.9. Where earlier, we achieved an optimal Silhouette score above 0.9, we now have one much closer to 0.5. Furthermore, we notice that the optimal number of clusters based on this Silhouette score was two, leading to the clustering as depicted in 6.11b. Knowing a priori that the data in the target feature consists of one cluster, this clustering is not satisfactory and considering both clusters separately for restoration may lead to less accuracy due to loss of generalization. As, if no other feature in the input gives rise to this split in clustering, we are losing information for restoring missing values in either cluster. Finally, we look at the number of data points within each cluster constructed by the Output Clustering methodology:

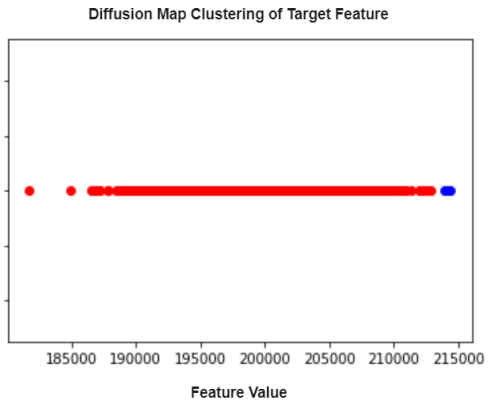
	Cluster 1	Cluster 2
Percentage of data	50.90%	49.10%

Table 65: Percentage of data of target feature in each cluster.

The Output clustering using K-means with two clusters leads to a near even split of the data within the target feature. Just based on these statistics, considering a data point with missing value in the target feature, and classifying it to one of the two clusters, where no clustering should exist, can lead to large reductions of restoration accuracy.
Next, we will consider the clustering of the target feature with respect to Diffusion Map clustering:



(a) Silhouette score for each number of clusters in k-means.



(b) Clustering of target feature with one cluster.

Figure 6.12: Diffusion Map clustering statistics for target feature with one cluster.

From the clustering performed by Diffusion Map methodology, we see that we still have the best Silhouette score achieved by the K-means clustering on the diffusion map for two clusters. However, many subsequently considered number of clusters greater than two up to nine show very similar Silhouette scores. Contrary to the clustering performed by Output Clustering methodology, the Diffusion Map method seems to construct one big cluster containing the majority of the data and one smaller cluster containing some data points that may be considered to be outliers with respect to the diffusion map representation of the data in the target feature. To confirm that which we observe in 6.12b, we provide the percentages of data of the target feature in each cluster:

	Cluster 1	Cluster 2
Percentage of data	99.60%	0.40%

Table 66: Percentage of data of target feature in each cluster.

The table, furthermore, confirms our suspicions, Diffusion Map clustering leads to a scenario in which we would not consider clustering with respect to the target feature, which would be the desired result. However, it should be noted that we perform Diffusion Map clustering using input features separately. This means that in order to determine the scenario for which we do not consider clustering the data in the target feature, the algorithm would need to confirm that for each input feature there is no significant clustering, in example all but one cluster has more less 1% data points. This methodology is cumbersome and abrogates the work done to reduce computational complexity from feature and model selection strategies.

This then leaves us to propose the use of DBSCAN to first determine whether or not there is need for clustering. DBSCAN, as discussed in the clustering subsection of the Data Structure Analysis node, is a density based clustering algorithm. DBSCAN can be directly used on the target feature in the one dimensional setting as with Output Clustering. DBSCAN has no need to know the number of clusters prior to clustering and returns labels concerning different clusters and labels of -1 for outliers. The results of running DBSCAN on our target feature consisting of one cluster are presented in the following figure:

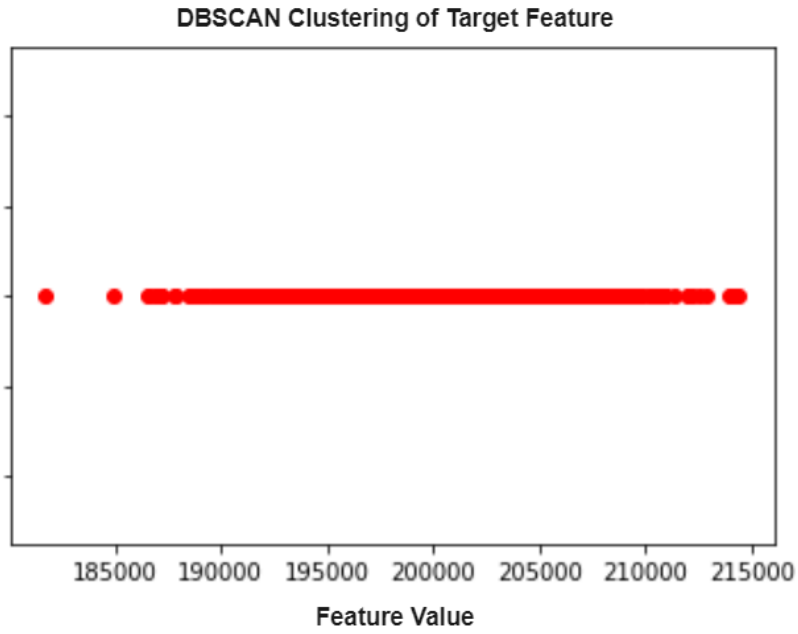


Figure 6.13: DBSCAN clustering of target feature consisting of one cluster.

As can be seen from the figure, DBSCAN considers all instances of the target feature to be part of one and the same cluster and thus DBSCAN is able to handle this scenario appropriately. To confirm, we provide the percentages of data per cluster:

	Cluster 1
Percentage of data	100.00%

Table 67: Percentage of data of target feature in each cluster.

This table indeed confirms all instances of the target feature are part of one and the same cluster. However, now we wish to know whether DBSCAN can indeed also handle the case where more than one cluster exists within the data of the target feature. As such we return to the target feature with density provided in figure 6.6. We once more construct the clusters as provided by DBSCAN for this target feature:

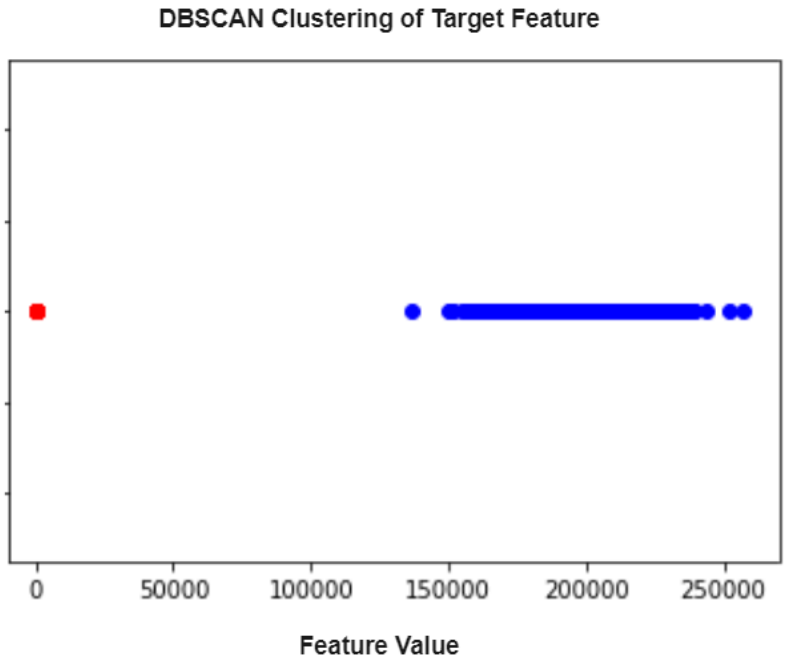


Figure 6.14: DBSCAN clustering of target feature consisting of two clusters.

As we can see from the figure, DBSCAN is able to determine there are clusters within the target feature based on their density, without need to analyze the case where there need to be at least two clusters as performed with Diffusion Map and Output clustering methodologies.
We provide the percentages of zero and non-zero values for each cluster:

	Cluster 1	Cluster 2
Percentage of zero values	100.00%	0.00%
Percentage of non-zero values	0.00%	100.00%

Table 68: Percentage of data of target feature in each cluster.

This confirms the successful clustering of the target feature by DBSCAN for this scenario, confirming DBSCAN is capable of handling both scenarios considered for this target feature.
We will, however, opt to only use the DBSCAN methodology for the determination of the necessity to cluster as DBSCAN has some non-desirable properties. Let us consider the same data set once more introducing the occurrence of 10 duplicate values within the blue cluster containing only non-zero values. These duplicate entries also exist in the true Aegon dataset for this target feature. This then results in the following clustering results for DBSCAN:

	Cluster 1	Cluster 2	Cluster 3
Percentage of zero values	100.00%	0.00%	0.00%
Percentage of non-zero values	0.00%	100.00%	100.00%
Number of data points	6916	3084	10

Table 69: Percentage of data of target feature in each cluster.

As we can observe in this table, the DBSCAN methodology clusters duplicate occurrences, due to their high density in that singular value, to a separate cluster. Though this may not seem to be a problem, omitting the most frequent occurrences of data points within the non-zero cluster can lead to a misrepresentation of the data relative to the true data. As such we have opted for the use of DBSCAN to

determine if there are two or more clusters of size greater than 1000 data points or 5% of the size of the input data, then we will consider the need for using further clustering such as Diffusion Map and Output Clustering approaches.

6.4.3 Performance of the Cleansing Algorithm

In this section we discuss the results as achieved by the cleansing algorithm on the Aegon data sets. First we provide the restoration rates for all different configurations of the cleansing algorithm considered.

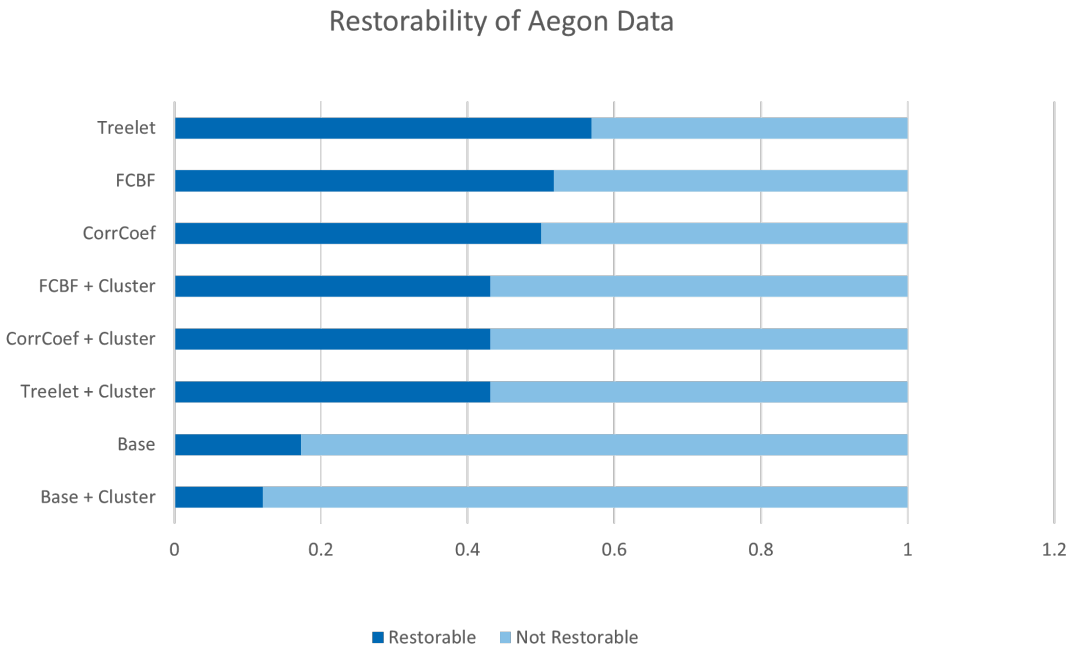


Figure 6.15: Comparison of restoration rates of configurations of the Cleansing Algorithm.

In figure 6.15, *Base* indicates the use of no feature selection and no clustering. From this figure the first thing we observe is that the Treelet feature selection methodology slightly outperforms the other two feature selection methodologies in terms of restoration rates over all Aegon data sets considered. The second highest restoration rate is achieved by the Fast Correlation Based Filter, closely followed by the Correlation Coefficient feature selection strategy. Next, we observe that the use of clustering on the Aegon data sets leads to a lowering of the restoration rate. This is not that odd when we consider the case of correctly missing data in output features. For output features with correctly missing data, we observed that some have percentages of over 90% of their values as imputed zeros or negative ones. If we cluster away all these correctly missing data points, we are left with a cluster with only data points with actual realized values. As we observed in table 60, these other data points do not manage to reach a restoration rates that surpasses the threshold of 95%. Though we do expect that the restorability of these specific data points is better than if we do not perform clustering, the restorability of the Aegon data sets tells us that there are such features that had a restoration rate greater than 95% when the models could restore many correctly missing data points, and now no longer are able to match the restoration rate threshold. Another observation we make is that the restorability is greater for all Cleansing Algorithm configurations than for the base cases. We notice how the restorability of Treelet is 230% greater than that of the Base without clustering. Even if we consider the case where we do perform clustering, we observe a restorability of the fast correlation based filter with clustering that outperforms the Base with clustering by 257.14%. This serves as an indicator as to how irrelevant or redundant features are capable of reducing the efficiency of all machine learning methods used within the algorithm, strengthening the necessity for the use of some sort of feature selection.

Furthermore, we note that the restoration rates for Diffusion map and Output clustering yielded the exact same results. This is due the fact that it is a common occurrence that the disparity in the missing feature is due to some single or small subset of other feature as is seen from the Data Quality check for the example provided in figure 6.7. As such the Diffusion map finds this cluster by comparing the missing feature with the other feature in the two-dimensional space, whereas the Output clustering approach finds it by clustering in the one-dimensional setting only considering the missing feature. Since both approaches yield the same results in terms of restoration rate we would prefer to use the Output clustering approach as the Diffusion map clustering considers all input features with respect to the feature with missing data individually and in two dimensions, whereas the Output clustering need only perform one computation in the one-dimensional setting on the feature with missing data. Even though the output clustering approach requires to use a separate classification model to determine to which cluster the actual missing data belongs, this methodology is shown to reduce time complexity of the cleansing algorithm.

Next we provide the run times for the various configurations of the Cleansing Algorithm excluding the clustering approaches:

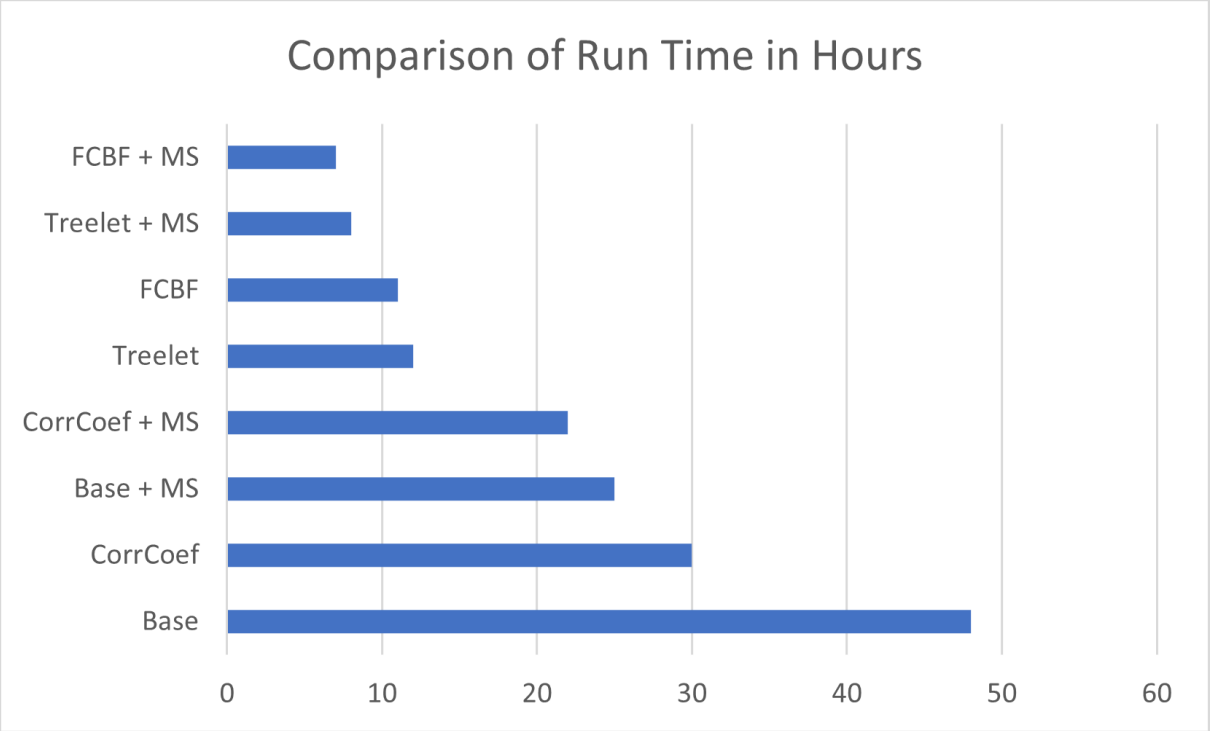


Figure 6.16: Comparison of run times of configurations of the Cleansing Algorithm.

From the run times in figure 6.16, the first observation we make is that the use of our Model Selection approach (+ MS in the figure), always leads to a reduction of the run time of the algorithm on the Aegon data sets. In order to get a better grasp of the reduction in run time that occurs due to the Model Selection approach, we provide a table with the percentage reduction of the run time in ours of the Cleansing Algorithm settings with versus without model selection:

	Base	CorrCoef	Treelet	FCBF
Reduction of run time	47.92%	26.67%	33.33%	36.36%

Table 70: Reduction of run time by using Model Selection strategy.

From these results one may assume that the estimation of a reduction of 48.7% for classification and 50.3% for regression based on the proposed Model Selection strategy was not achieved in practice. This,

however, is difficult to conclude as the run times provided take into consideration the entire process of the Cleansing Algorithm for each configuration. For example if the reduction of Treelet and FCBF feature selection in combination with Model Selection is not the expected result, it could be due to the fact that the nodes prior such as Data Transformation or Data Structure Analysis are the bottleneck in this configuration, such that the Model Selection process itself is indeed adhering to the theoretical reduction. This suspicion is confirmed by the reduction in run time for the Base case. In this case all features pass through to the model selection process, this means that, for the Base configuration, the Cleansing Algorithm will have to construct much more complex models due to the high dimensionality. This causes the process of choosing the model with the highest restoration rate to take the maximum amount of time. As such our Model Selection strategy for the Base case forms the best representation of the theoretical testing setting. This is confirmed by the run time reduction for the Base case, being a reduction of 47.92%, which is close to what was theorized. On the other hand, the FCBF and Treelet feature selection approaches were shown to yield the greatest reduction in the number of features being selected. This then leads to the construction of less complex models and thus a lower base run time for the construction of their models. In terms of the number of hours run time, this then leads to the lowest reduction naturally.

Contrary to this, it is very possible that the theoretical reduction was optimistic in that the true rankings of the models based on Aegon data are different to the ones discovered through theoretical testing. For this reason we shall analyze the types of models that have led to these restorabilities.

Type of Model	LR	PR	EPR	MLP	RF	KNN
Occurrence Rates	29.55%	9.09%	0.00%	0.00%	45.46%	15.90%

Table 71: Occurrence rates of each regression model having the greatest restoration rate.

Type of Model	EM	MLP	RF	KNN	KM
Occurrence Rates	36.59%	0.00%	14.63%	48.78%	0.00%

Table 72: Occurrence rates of each classification model having the greatest restoration rate.

As can be seen in table 71 the unconditional results for models on the Aegon data sets turned out different to those found in the theoretical setting. However, of note is that still ElasticNet Polynomial regression and the Multi-Layer Perceptron regression did not find any representation. From the occurrence rates in table 71, we see that linear regression and random forest regression found greater representation than expected prior, whereas polynomial and nearest neighbor regression are underrepresented compared to the theoretical results in table 32. This disparity may occur due to numerous reasons. We shall discuss two possibilities. The first possibility concerns the shortcomings of the theoretical testing approach. It was clear when the theoretical testing was constructed that we were never going to be able to capture all possible data relationships and distributions. It could be that disparity in model occurrences is due to the Aegon data not adequately adhering to the theoretical data testing. Furthermore, real-world data, such as that from Aegon, can contain noise and outliers and in general is not very "clean". This may also influence the choice to not use polynomial regression as this model is very sensitive to noise, whereas a model such as linear regression is less so.
The second possibility is that the rankings as captured by theoretical testing of the models is similar to the conditional rankings the models adhere to on the Aegon data sets. If this is the case, this indicates that the unconditional occurrences as presented in table 71 were the inaccuracy of theoretical testing. This then would indicate the expected number of models that needs to be constructed to possibly be different to that theorized in section 5.2. If this theory holds true, we expect the restorability of the Cleansing Algorithm when using the Model Selection strategy, to be the same as the one when not using the strategy and choosing the best model in terms of restoration rate from all candidate models. As such, we provide the restorability for the Cleansing Algorithm configurations using the Model Selection strategy versus without:

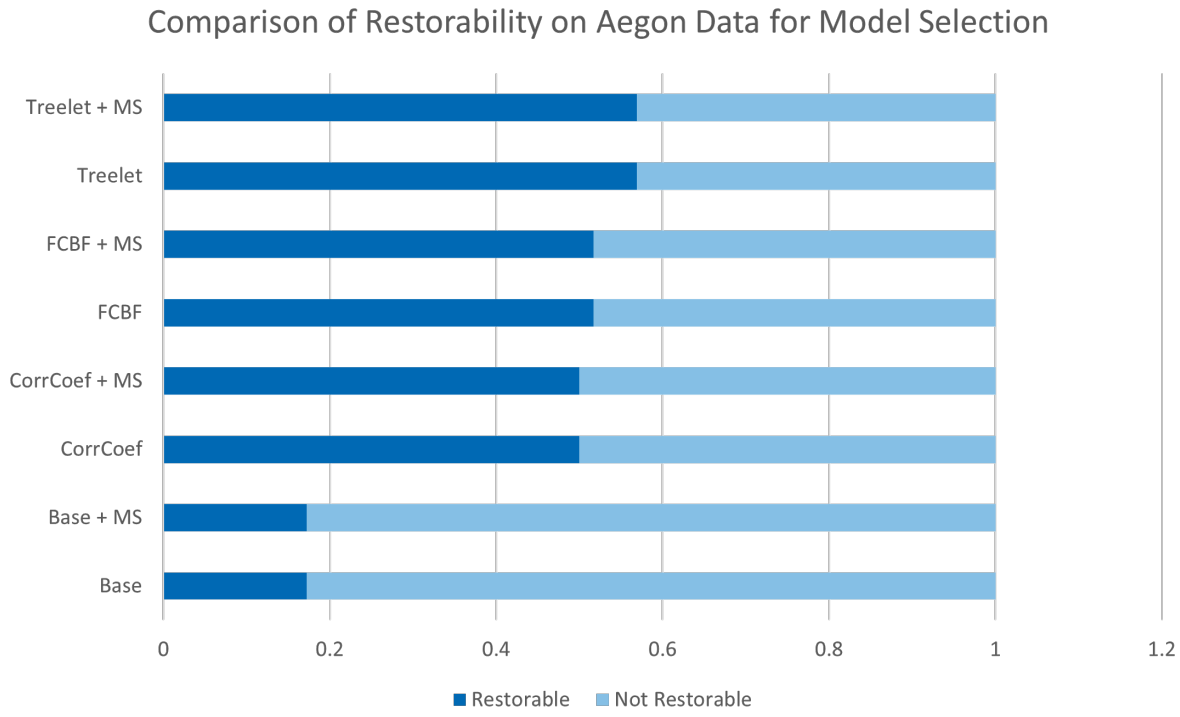


Figure 6.17: Comparison of restorability of using the Model Selection strategy of the Cleansing Algorithm.

As can be observed in figure 6.17, the restorability for configuration of the Cleansing Algorithm with the Model Selection strategy versus those without is identical. For this reason, we theorize that the conditional rankings of the different candidate models as considered in our Model Selection strategy is up to a certain extent similar to those that hold true within the Aegon data sets. Based on the fact that there is no loss in restorability of the Cleansing Algorithm when making use of our Model Selection strategy and the fact that the Model Selection strategy reduces the run time of the Cleansing Algorithm in all cases, we deem the use of the Model Selection strategy to be the preferred choice.

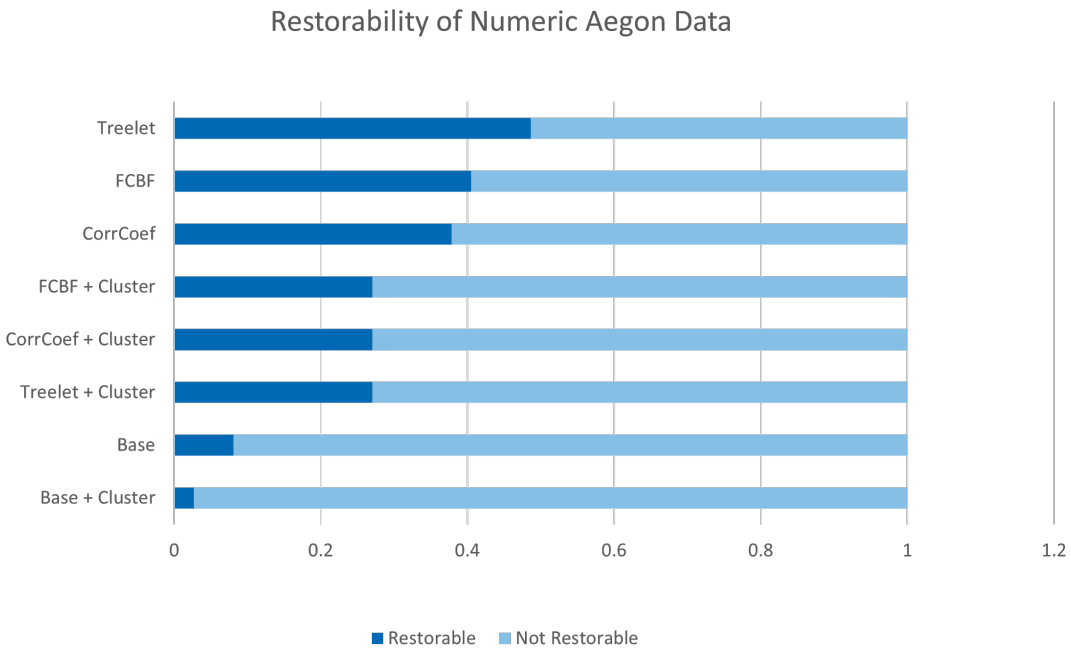


Figure 6.18: Comparison of numeric restorability of configurations of the Cleansing Algorithm.

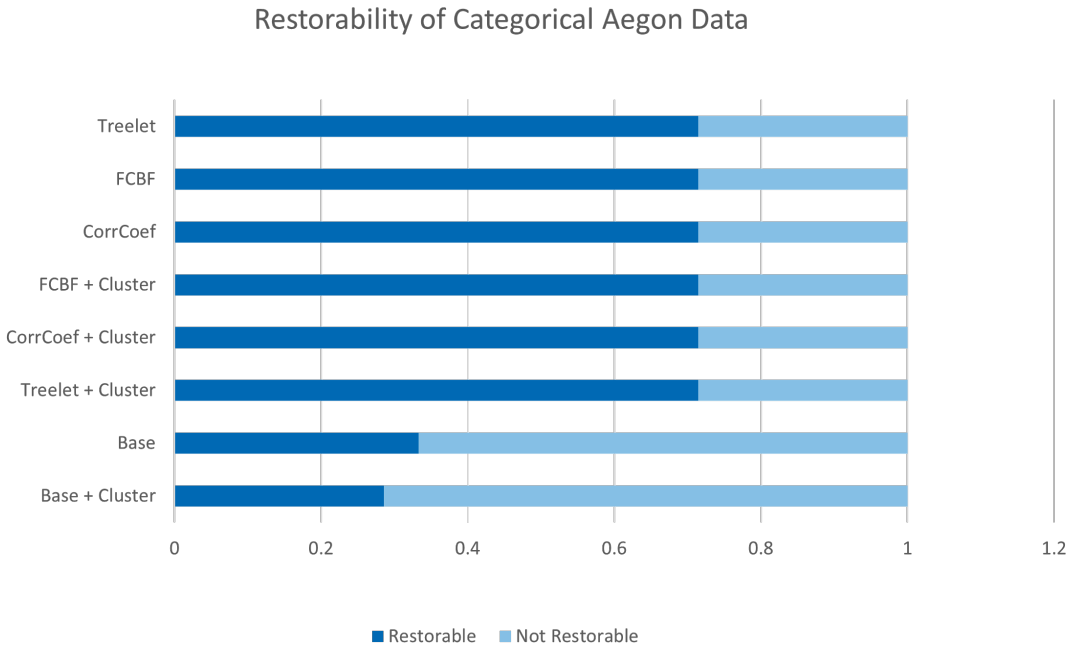


Figure 6.19: Comparison of categorical restorability of configurations of the Cleansing Algorithm.

Finally, we provide the relative frequency and restoration rates for categorical (classification) and numeric (regression) features with missing data. As we observe from figures 6.18 and 6.19, the restoration rate is significantly greater for classification than for regression. This may seem illogical at first considering we are more lenient with respect to regression as we consider everything within 5 percent to be appropriate for restoration, however, this is not the case in practice. In example, for someone to guess the exact price of a house with 5 percent error margin is much more difficult than determining in

which category the house price would fall. Categorical data allows for less margin of error in this way. Categorical data also has a baseline restoration rate equal to 1 over the number of unique categories and even has the possibility to already perform well just suggesting the most frequently occurring category. We also notice that though the restoration rate of the cleansing algorithm decreases, it only seems to decrease with respect to numeric features with missing data. This is conform our discussion of the clustering approaches above.

7 Conclusion & Discussion

7.1 Conclusion

The goal of this research was to determine whether it is possible to restore missing data while allowing space for human input to adapt the algorithm.

Firstly, if we are to consider those configurations of the Cleansing Algorithm that have the greatest restorability, we conclude that it is possible to restore missing data on the Aegon data sets.
The greatest restorability achieved was done by the Cleansing Algorithm making use of the Treelet feature selection, performing no clustering and using our Model Selection strategy. The restorability for this configuration was 56.90% over all data sets of Aegon considered. On the other hand, if we consider the configuration of the Cleansing Algorithm that had the lowest run time, while still yielding the second highest restorability, then the Fast Correlation-Based Filter as feature selection, no clustering and the use of our Model Selection strategy also formed an appropriate choice. This configuration had a restorability of 51.72% on the Aegon data sets and had a 12.50% lower run time when compared to the configuration with highest restorability.
If, however, we consider a more realistic restorability in light of the presence of correctly missing data in semi-numeric output features, the best configuration of the Cleansing Algorithm was given by using either Correlation Coefficient, Treelet or FCBF feature selection approaches, with Output Clustering and our Model Selection strategy. For this configuration the restorability was 43.10%. In comparison, the restorability of not using any strategies employed in the Cleansing Algorithm was 12.07% with a vastly greater run time that was over 400% of run time of the realistic restorability configuration.

Secondly, the cleansing algorithm has been constructed in such a way that any user or expert is able to provide feedback on the workings of the cleansing algorithm. If restored values seem unrealistic, it can be indicated and the algorithm will aim to use output clustering if not already done and otherwise try diffusion map clustering or rather not use a clustering. If the user determines the use of a certain model to be inadequate or recommends the use of a new type of model, the algorithm updates its theoretical testing and restorations accordingly. If a user deems input features used to be missing or inadequate it can be indicated and the algorithm will update restoration. On the other hand, if a user deems a restoration to be logical and correct the model can be stored an upon recognition of the data set and missing feature the algorithm will fetch the model when called upon again. In conclusion, we deem the restoration of missing data using a human adaptive framework to be possible for the Aegon data set.

7.2 Discussion

Even though, the conclusion of our research was a positive one, there is much room for improvement and many more possibilities for the restoration of missing data using a human adaptive framework.

In the Results section, we discussed the use of diffusion map and output clustering to tackle the disparity between distribution of data used in our models compared to the missing data. The original reasoning for clustering was the presence of correctly missing data. The problem with these types of features, is that if a business expert wishes to restore missing data, the easiest values to restore are this correctly missing values. If as such we construct models to restore missing data also considering the correctly missing values, the model restoration rate is not a realistic one as all correctly missing cases have prior been restored by an expert. This then leads to a problem in the assumption that data is missing completely at random. In this case such an output feature's missing values will actually be missing at random as they are only missing if they are not correctly missing. For further research in a similar general data setting, one may wish to determine prior what type of missing data for each data set specifically and adapt the approach to suite each type of missing data separately.

The best, and all, settings for the cleansing algorithm showed to be much better at classification tasks than regression tasks. For future research one could consider more probabilistic models that would determine conditional distributions with respect to features containing missing data and as such provide some sort of confidence intervals rather than singular values for the restoration of missing data.

In the Model Selection section, we observed that elastic net polynomial and multi-layer perceptron regression methods did not ever outperform the other models considered. However, in specific settings, that have not been considered in our theoretical testing approach, these models may be able to outperform the other models. For example the power of the elastic net polynomial regression starts paying off when the number of available data points for the restoration of missing data is far less than the number of parameters the model needs to estimate. A prior filtering in the model selection phase for some arbitrary ratio between number of parameters to determine versus the number of available missing data points can lead to better restoration rates in such settings if they occur.

For the multi-layer perceptron, we only considered some arbitrary hidden layer structures based on literature, though the full power of neural networks in general stems in their ability to adapt to many problem settings. Some algorithms such as Neuroevolution of augmenting topologies (NEAT), allow the use of evolutionary algorithms to determine optimal neural network hidden layer structures for each specific feature with missing data. Though this may be time consuming to tailor the MLP hidden layer structure to each one separately. This also requires the use of very large data sets of hidden layer structures grow increasingly complex.

Furthermore, we only considered some common relations and distributions for the theoretical testing in Model Selection. One may be able to construct a more concise or more all-encompassing combination of distributions and relations to even better capture relations.

In addition, we considered the use of 6 regression models and 5 classifiers, where there exist many more and even custom ones can be made, these may also just be inserted into the algorithm leading to new insights in terms of rankings and model redundancies.

Finally, with respect to our Model Selection strategy, it need not always be the case that the optimization problem of minimizing the number of models used while being able to adhere to all conditional rankings, is possible. In the case this occurs, currently, the Model Selection strategy shall disregard the conditional rankings of the models with the lowest unconditional probabilities. In the case a much greater number of models for regression and classification are considered, this could lead to lower restorability of the Cleansing Algorithm configuration with the Model Selection strategy versus not using it. In light of this, one could start as done during this research with some theoretical basis and then update occurrence distributions and conditional rankings according to frequencies observed on the actual data to counteract

the deficiency of the Model Selection strategy in this case. On a few models these changes may not lead to alternative model selection strategies, but on many models differences may be significant enough to return a new conditional rankings and unconditional probabilities leading to a new optimal ordering.

References

[1] Cai, Li Zhu, Yangyong. (2015). *The Challenges of Data Quality and Data Quality Assessment in the Big Data Era.* Data Science Journal. 14. 10.5334/dsj-2015-002.

[2] Butt, M. (2007). *Insurance, finance, solvency ii and financial market interaction.* Geneva Papers on Risk Insurance, 32(1), 42–42. https://doi.org/10.1057/palgrave.gpp.2510115

[3] Harris, D. M., Harris, S. L. (2013). Digital design and computer architecture (2nd ed., Ser. Engineering professional collection). Morgan Kaufmann. Retrieved October 20, 2022, p. 129. ISBN 978-0-12-394424-5.

[4] Székely, G. J., Rizzo, M. L., Bakirov, N. K. (2007). *Measuring and testing dependence by correlation of distances.* The annals of statistics, 35(6), 2769-2794.

[5] Schunk, D. (2008). *A markov chain monte carlo algorithm for multiple imputation in large surveys.* Asta Advances in Statistical Analysis : A Journal of the German Statistical Society, 92(1), 101–114. https://doi.org/10.1007/s10182-008-0053-6

[6] Kononenko, I., Simec, E., Robnik-sikonja, M. (1997). *Overcoming the myopia of inductive learning algorithms with relieff.* Applied Intelligence, 7(1), 39–55. https://doi.org/10.1023/A:1008280620621

[7] Hall, M.A. (1999) *Correlation-based Feature Subset Selection for Machine Learning.* PhD thesis, Department of Computer Science, University of Waikato, Waikato, N.Z.

[8] Shin, K., Xu, X.M. (2009). *Consistency-Based Feature Selection.* In: Velásquez, J.D., Ríos, S.A., Howlett, R.J., Jain, L.C. (eds) Knowledge-Based and Intelligent Information and Engineering Systems. KES 2009. Lecture Notes in Computer Science, vol 5711. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/978-3-642-04595-0_42`

[9] Wang, C., Shakhovska, N., Sachenko, A., Komar, M. (2020). *A New Approach for Missing Data Imputation in Big Data Interface.* Information Technology and Control, 49(4), 541-555. https://doi.org/10.5755/j01.itc.49.4.27386

[10] Heung, B., Ho, H. C., Zhang, J., Knudby, A., Bulmer, C. E., Schmidt, M. G. (2016). *An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping.* Geoderma, 265, 62–77. https://doi.org/10.1016/j.geoderma.2015.11.014

[11] Weisberg, S. (2005). *Applied linear regression* (Vol. 528). John Wiley Sons. ISBN:9781118386088

[12] Ostertagova, E. (2012). *Modelling using polynomial regression.* Procedia Engineering, 48, 500–506. https://doi.org/10.1016/j.proeng.2012.09.545

[13] Ogutu, J. O., Schulz-Streeck, T., Piepho, H. P. (2012, December). *Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions.* In BMC proceedings (Vol. 6, No. 2, pp. 1-6). BioMed Central. https://doi.org/10.1186/1753-6561-6-S2-S10

[14] Murtagh, F. (1991). *Multilayer perceptrons for classification and regression.* Neurocomputing, 2(5), 183–197. https://doi.org/10.1016/0925-2312(91)90023-5

[15] Kingma, D. P., Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

[16] Breiman, L. *Random Forests.* Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[17] Taunk, K., De, S., Verma, S. and Swetapadma, A. (2019) *A Brief Review of Nearest Neighbor Algorithm for Learning and Classification,* 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[18] Neagoe, V. E., Chirila-Berbentea, V. (2016, July). *Improved Gaussian mixture model with expectation-maximization for clustering of remote sensing imagery.* In 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS) (pp. 3063-3065). IEEE. DOI: 10.1109/I-GARSS.2016.7729792

[19] Papavasileiou, E., Cornelis, J., Jansen, B. (2021). *A systematic literature review of the successors of "neuroevolution of augmenting topologies".* Evolutionary Computation, 29(1), 1-73. DOI: 10.1162/evco_a_00282

[20] Dijkstra, S. (2019). *Analysis of the impact of news on the financial market.* Student Thesis, TU Delft Repository, http://resolver.tudelft.nl/uuid:5d053208-4c51-415f-bd1f-bfc9f819ebf8

[21] Ester, M., Kriegel, H. P., Sander, J. and Xu, X. (1996) *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996

[22] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., Xu, X. (2017). *DBSCAN revisited, revisited: why and how you should (still) use DBSCAN.* ACM Transactions on Database Systems (TODS), 42(3), 19.

[23] Limwattanapibool, O., Arch-int, S. (2017). *Determination of the appropriate parameters for k-means clustering using selection of region clusters based on density dbscan (srcd-dbscan).* Expert Systems, 34(3). https://doi.org/10.1111/exsy.12204

[24] Ester, M., Kriegel, H. P., Sander, J., Xu, X. (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise.* In kdd (Vol. 96, No. 34, pp. 226-231).

[25] Roozemond, E. S. (2021). *Dimensional Reduction using Diffusion Maps.* https://www.diva-portal.org/smash/get/diva2:1570862/FULLTEXT01.pdf

[26] Coifman, R. R., Lafon, S. (2006). *Diffusion maps.* Applied and Computational Harmonic Analysis, 21(1), 5–30. `https://doi.org/10.1016/j.acha.2006.04.006`

[27] Lee, A. B., Nadler, B., Wasserman, L. (2008). *Treelets - an adaptive multi-scale basis for sparse unordered data.* Ann. Appl. Stat, 2(2), 435–471. https://doi.org/10.1214/07-AOAS137

[28] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., Liu, H. (2018). *Feature selection: a data perspective.* Acm Computing Surveys, 50(6).

[29] Senliol, B., Gulgezen, G., Yu, L. and Cataltepe, Z. (2008) *Fast Correlation Based Filter (FCBF) with a different search strategy*, 2008 23rd International Symposium on Computer and Information Sciences, 2008, pp. 1-4, doi: 10.1109/ISCIS.2008.4717949.

[30] Song, Q., Ni, J. and Wang, G. (2013) *A Fast Clustering-Based Feature Subset Selection Algorithm for High-Dimensional Data*, in IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 1, pp. 1-14, Jan. 2013, doi: 10.1109/TKDE.2011.181.

[31] Hsu, Hui-Huang Hsieh, Cheng-Wei. (2010). Feature Selection via Correlation Coefficient Clustering. JSW. 5. 1371-1377. 10.4304/jsw.5.12.1371-1377.

[32] Vazquezl, M.Y.L., Peñafiel, L.A.B., Muñoz, S.X.S., Martinez, M.A.Q. (2021). *A Framework for Selecting Machine Learning Models Using TOPSIS.* In: Ahram, T. (eds) Advances in Artificial Intelligence, Software and Systems Engineering. AHFE 2020. Advances in Intelligent Systems and Computing, vol 1213. Springer, Cham. `https://doi.org/10.1007/978-3-030-51328-3_18`

[33] Pedersen, A. B., Mikkelsen, E. M., Cronin-Fenton, D., Kristensen, N. R., Pham, T. M., Pedersen, L., Petersen, I. (2017). *Missing data and multiple imputation in clinical epidemiological research.* Clinical epidemiology, 9, 157–166. `https://doi.org/10.2147/CLEP.S129785`

[34] Zhang, Z., Luo, Y. (2017), *Restoring method for missing data of spatial structural stress monitoring based on correlation.*, Mechanical Systems and Signal Processing, Volume 91, 2017, Pages 266-277, ISSN 0888-3270, `https://doi.org/10.1016/j.ymssp.2017.01.018.`

[35] Wolfe, P. J. and Godsill, S. J. (2005), *Interpolation of missing data values for audio signal restoration using a Gabor regression model.*, Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005., 2005, pp. v/517-v/520 Vol. 5, doi: 10.1109/ICASSP.2005.1416354.

[36] Yu, L. Liu, H. (2003). *Feature selection for high-dimensional data: A fast correlation-based filter solution.* In Proceedings of the 20th international conference on machine learning (ICML-03) (pp. 856-863).

[37] Kraskov, A., Stögbauer, H., Grassberger, P. (2004). *Estimating mutual information.* Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics, 69(6 Pt 2), 066138–066138.

[38] Ross, B. C., Marinazzo, D. (2014). *Mutual information between discrete and continuous data sets.* Plos One, 9(2), 87357. `https://doi.org/10.1371/journal.pone.0087357`

[39] Lloyd, S. P. (1982), *Least squares quantization in PCM*, IEEE Transactions on Information Theory, 28 (2): 129–137, doi:10.1109/TIT.1982.1056489.

[40] Rousseeuw, P. J. (1987). *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*. Computational and Applied Mathematics. 20: 53–65. doi:10.1016/0377-0427(87)90125-7

[41] Salzberg, S. L. (1994). *C4.5: programs for machine learning by j. ross quinlan.* Morgan Kaufmann Publishers, Inc., 1993. Machine Learning, 16(3), 235–240. https://doi.org/10.1007/BF00993309

[42] Schleider, L., Pasiliao, E. L., Qiang, Z., Zheng, Q. P. (2022). *A study of feature representation via neural network feature extraction and weighted distance for clustering.* Journal of Combinatorial Optimization, (20220221). https://doi.org/10.1007/s10878-022-00849-y

[43] Roozemond, E. S. (2021). *Dimensional Reduction using Diffusion Maps.* Mathematics Department, Uppsala University. `https://www.diva-portal.org/smash/get/diva2:1570862/FULLTEXT01.pdf`

[44] Heaton, J. (2005). *Introduction to neural networks with java (1st ed.).* Heaton Research. ISBN: 097732060X, 9780977320608

[45] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) *Scikit-learn: Machine Learning in Python. Journal Of Machine Learning Research.* **12** pp. 2825-2830

[46] Schafer, J. L., Graham, J. W. (2002). *Missing data: Our view of the state of the art.* Psychological Methods, 7(2), 147–177. `https://doi.org/10.1037/1082-989X.7.2.147`

[47] Stanczyl, U. (2015), Institute of Informatics, Silesian University of Technology, Gliwice, Poland. *Feature selection for data and pattern recognition. In Feature evaluation by filter, wrapper, and embedded approaches (pp. 29–44).* essay, Berlin, Heidelberg : Springer Berlin Heidelberg : Springer.`https://doi.org/10.1007/978-3-662-45620-0_3`

[48] Ranstam, J., Cook, J. A. (2018) *LASSO regression*, British Journal of Surgery, Volume 105, Issue 10, September 2018, Page 1348, `https://doi.org/10.1002/bjs.10895`

[49] Benesty, J., Chen, J., Huang, Y., Cohen, I. (2009). *Pearson correlation coefficient.* In Noise reduction in speech processing (pp. 1-4). Springer, Berlin, Heidelberg.

[50] Bonett, D. G., Wright, T. A. (2000). *Sample size requirements for estimating pearson, kendall and spearman correlations.* Psychometrika, 65(1), 23–28. https://doi.org/10.1007/BF02294183

[51] Biesiada, J., Duch, W., Kurzynski, Marek, Prof., Faculty of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, Wroclaw, 50-370, Poland. (2007). Computer recognition systems 2. In Feature selection for high-dimensional data — a pearson redundancy based filter (pp. 242–249). essay, Berlin, Heidelberg : Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75175-5_30