Detecting Dish Types in Picnic Deliveries

S.C. Noorthoek

TU Delft

Detecting Dish Types in Picnic Deliveries

by

S.C. Noorthoek

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 26, 2022 at 11:00 AM.

Student number:	4375769	
Project duration:	November 8, 2021 – August 26, 2022	
Thesis committee:	Dr. N. Yorke-Smith,	TU Delft, supervisor
	Dr. B. Vlaming,	Picnic, supervisor
	Dr. J. Yang,	TU Delft

TU Delft

Abstract

In addition to delivering groceries at customers' doorsteps, online supermarket Picnic goes the extra mile by aiming to improve customer satisfaction. For instance, by providing cooking inspiration to customers through a recently launched recipe page in the app. This feature presents new recipes weekly and allows customers to easily add the ingredients to their shopping basket. It has raised interest in finding out what dishes customers are cooking as it could be helpful in choosing recipes for the page, predicting which articles are forgotten before checkout, and building a recipe recommender system. Hence, this work proposes two models to detect dish types in Picnic deliveries. The problem is scoped to detect main meals from a specified list of dish types in deliveries which were ordered in the Netherlands. The first model, named the *Frequent Itemset Model*, applies unsupervised learning techniques. First the articles in the deliveries are pre-processed by removing certain articles, choosing the representation of articles, and cleaning the text. The itemsets which represent core ingredients are obtained by applying techniques such as frequent itemset mining, association rule mining, and hierarchical clustering. In the final step itemsets are matched to dish types with the use of programmatic labelling and fuzzy string matching. Newly available labelled data enables the creation of a second model, referred to as the *Supervised Learning Model*, which applies supervised learning techniques. Features are selected and extracted. Multiple machine learning models, some in combination with binary relevance, are compared. The models are evaluated on two datasets: a large, weakly labelled dataset obtained through the recipe page, and a very small, manually labelled dataset with deliveries from a single customer. It is challenging to evaluate the performance of the models, since no large, truly labelled dataset is available. The results do indicate that the Frequent Itemset Model is able to detect common dish types, and that the Supervised Learning Model is able to detect dish types which are similar to the Picnic recipes it has trained on. Multiple suggestions are made for future work, such as obtaining a larger variety of labelled data and redefining the class labels. The contribution of this work is the formulation of the problem, two proposed solutions, insights into the challenges, and suggestions for future work.

Preface

Before you lies the thesis "Detecting Dish Types in Picnic Deliveries", that came into being in collaboration with Picnic. It has been written to complete the Master Computer Science at Delft University of Technology. I began to pursue this master's degree almost two years ago and despite most courses being taught online due to lockdowns, I enjoyed it greatly. I am a big fan of Picnic's product and I was thrilled that I could complete my thesis at this company. The internship surpassed my expectations in multiple ways. From the way that I was welcomed as a team member, to the learning opportunities that were offered in addition to my project. This thesis problem is one that is easy to comprehend, but hard to solve. I could not have come as far nor learned as much as I did, without all of the support that I was given. There are a few people that I would like to thank in particular.

First I want to thank my TU Delft supervisor Dr. Neil Yorke-Smith for giving me the freedom to shape this project, discussing all of my concerns and ideas, and guiding me along the way. I want to thank Dr. Jie Yang for being part of my thesis committee. Secondly, I want to thank my amazing team and all of the people that I met at Picnic who have made this experience so enjoyable. A special thanks goes out to my Picnic supervisor Dr. Bas Vlaming. I always looked forward to our check-ins and I have learned so much from our detailed discussions and the PR reviews. I want to thank you for your guidance, keen eye, and positivity. I would also like to express my gratitude to Tad Slaff for helping me improve my project management skills, and to Sharon Gieske for helping me improve my programming skills. You were always available whenever I had questions, and have always involved me in the team and the company. Finally, I would like to thank my family and friends for their support but also for occasionally distracting me. I am incredibly thankful for my partner Felix, who has been my rock for many years. A very special acknowledgement goes out to my late grandfather Ir Pieter Leonardus van den Bogert, who studied Electrical Engineering at Delft University of Technology and instilled in me an interest for mathematics and engineering.

S.C. Noorthoek
Amsterdam, August 2022

Contents

1 Introduction

Online supermarket Picnic started in the Netherlands in 2015 with its first operations in Amersfoort. The company, which allows customers to order groceries via an app and have them delivered to their doorstep, has been experiencing tremendous growth. By the time of writing, Picnic has also launched in Germany and France. During this growth Picnic has structurally collected data on its operations, which is put into use to improve business processes and increase customer satisfaction. For instance, by using artificial intelligence to suggest relevant articles and to forecast the article demand. As a result, customers can more easily find the articles they are looking for and less often encounter unavailability of articles.

A new, customer-centric feature is the recipe page, where a selection of recipes is presented. The ingredients for these recipes are linked to articles in the app and can be added to the basket at a single click. The goal of the recipe page is to inspire customers and make the process of deciding what to eat for dinner easier and more enjoyable. The process of selecting recipes raised interest into what dishes Picnic customers are interested in. Insights into personal dish preferences, changes in dish preferences over the season, or the most popular dishes per region would help to choose recipes and improve the feature's success. These insights could be obtained using a model that detects dishes which customers are creating using the articles in deliveries. To this author's best knowledge, such a model is not available off the shelf nor has there been any research published on the problem of detecting dishes or dish types in supermarket transaction data. Hence, this research aims at investigating what approaches are the most effective for detecting dish types in Picnic deliveries.

1.1. Problem

The goal of this research project is to create and compare dish detection models. Given a Picnic delivery, the model should be able to detect which dish types were created with the articles from that delivery. The underlying abstract problem of detecting dish types in transaction data is a multi-label classification problem, where every instance (delivery) can be assigned multiple classes (dish types). Multi-label classification is related but intrinsically different from the more classic pattern recognition problem such as binary classification and multi-class classification. In binary classification one of two classes are predicted, for example using a Picnic delivery to predict if a customer's household includes or does not include of children. In multi-class classification more than two classes are predicted. For instance, predicting in which month a Picnic delivery was purchased. One large difference is that in multi-label classification classes are not mutually exclusive. Articles in a delivery can be used to cook pasta carbonara as well as red pepper soup.

1.2. Challenges

Around the world there is an immense number of unique dishes. Even on just a single website like Food.com one can find 500.000+ recipes [22]. The first task is to scope the model by defining what is meant by a dish, which dishes should be considered, and how they should be categorized. The relationships between dishes and ingredients is quite loose, but some patterns can be expected. For example, to make lasagne one probably uses lasagne sheets (although homemade ones can be made with eggs and flour). For a curry the ingredient list is likely to contain numerous spices or a ready-made spice paste. However, other instances can be ambiguous. The same dish can have multiple variations using different sets of ingredients. For example, soup

can be created with tomatoes, onions, and vegetable stock but also with mushrooms, garlic, and parsley. The opposite where the same set of ingredients is used to create different dishes can also occur. For instance, lettuce, bread, tomatoes, and avocado can be used to create a sandwich with lettuce as well as a salad with bread crumbs.

In addition to the unclearly defined dishes and corresponding ingredients, challenges arise from noisy data. Ideally, every delivery contains a complete set of articles/ingredients used to create one specific dish. However, customers also purchase articles unrelated to dishes (e.g. an apple to snack). It could also be that only a subset of the ingredients for a dish is contained in the delivery. The rest might be obtained elsewhere. For example, at another grocery store or during a previous delivery. Some articles, such as pasta and rice, have a long shelf life. These articles might be purchased in large quantities and stored in the pantry.

The final challenge is the availability of labelled data. Given a delivery, how can one know what dishes were actually cooked using a selection of the articles? The first requirement is to obtain the ground truth. Making assumptions about what dish a customers plans to cook based on the purchase of specific ingredients introduces inaccuracies. Secondly, there is a need for a large dataset. It is to be expected that due to the large variety in deliveries and dishes a large number of samples is needed to learn patterns. Altogether, many factors around definitions, noisy data, and availability increase the difficulty of the problem.

1.3. Scope

The problem is scoped by defining the model output, the selection of dish types, and the context of deliveries. Firstly, the output of the model is the name of a dish type such as salad. Specific recipes (i.e. instructions and ingredients) are not used as output of the model. The foremost reason being that it is impossible to collect all existing recipes. Secondly, a selected set of dish types, all main courses, are considered to deal with the huge variety of dishes. The categorization process and resulting selection are discussed in Section 3.2. Thirdly, the model only takes into account Picnic deliveries from the Netherlands. Although Picnic also operates in other countries, the deliveries in these countries are not considered as the recipe page feature was only live in the Dutch app at the time of developing the model. Additionally, different cuisines in different countries could increase the difficulty of the task.

1.4. Motivation

The scientific contribution is the creation and comparison of approaches to solve this novel problem. To the best of the author's knowledge there is no existing work looking into the detection of dishes in shopping baskets. The business contribution is a dish detection model which can be applied to gather information about customer preferences towards cooking meals. The results can serve as inspiration for selecting recipes and meal boxes, but also as input for decisions around product assortment, promotions, and online store layout. The model would be useful when building an article recommender system or a recipe recommender system. The former could, for example, aim predict whether certain articles were forgotten before checkout based on real-time basket data.

1.5. Research Questions

The main goal of this research study is to develop and compare methods to detect what dish types are created from articles in a Picnic delivery. Hence, the main research question is:

What approaches can identify dish types in Picnic deliveries?

The research questions building up to this main research question are:

1. Can an unsupervised learning algorithm detect dish types in deliveries?
2. Which supervised learning algorithms are the most effective for detecting dish types in deliveries?
3. Which features are the most useful for detecting dishes in deliveries?
4. Can programmatic labelling improve the performance of supervised machine learning models in this context?
5. How well does the unsupervised learning algorithm perform compared to supervised learning algorithms?

1.6. Methodology

Two models are created to detect dish types in Picnic deliveries. The first model, named the *Frequent Itemset Model*, uses unsupervised learning algorithms. The choice for an unsupervised learning approach was made since no labelled data was available at the start of the research project. The model aims at discovering articles that are often purchased together, and are highly related. To achieve this, the model pre-processes the deliveries with the help of Picnic's product taxonomy, and applies techniques such as frequent itemset mining, association rule mining, and hierarchical clustering. These combinations of articles are then linked to dish types through a recipe dataset and programmatic labelling. During the project weakly labelled data became available which has allowed the creation of the second model. This model, named the *Supervised Learning Model*, uses supervised learning algorithms. Multiple versions of different machine learning models, some in combination with binary relevance, are implemented and trained on the alternative dataset consisting of weakly labelled data.

1.7. Outline

The remainder of this thesis report is organized into seven chapters. Chapter 2 conducts a literature survey. No research on the problem in the identical context was found. Nonetheless this chapter focuses on related abstract problems in diverse contexts. Chapter 3 is devoted to data. It provides an overview of the source and structure of the data (e.g. deliveries and recipes) and the defined class labels (the dish types). It also explores the data through visualizations. Chapter 4 describes the Frequent Itemset Model, which uses unsupervised learning algorithms. An overview of the approach is provided, after which every section further explains the aspects of the model. Chapter 5 explains the Supervised Learning Model through its features, problem transformation methods, algorithm adaption methods, and supervised learning algorithms. Chapter 6 describes the design and results of the experiments. At the end the results are discussed in Chapter 7 and the conclusion and future work are presented in Chapter 8.

2

Literature Survey

The task of detecting dish types in transaction data is one where, to the best of the author's knowledge, no research has been published on. However, there does exist related work in similar contexts and on related abstract problems. Similar contexts are found in the field of market basket analysis. Data mining is applied to uncover patterns in large databases, often transaction data. Related abstract problem are also multi-label classification problems, where multiple classes can be assigned to instances.

In addition, food computing research has been growing as a result of voluminous, structured, and easily-obtainable data found in the increasing number of recipe sharing websites [45]. Particularly in the field of food image recognition [26, 18, 17, 67], recipe retrieval [12], recipe recommendation [62, 23], and cuisine classification [46, 54]. Although the overall theme is similar, the underlying data and problems are quite different from dish detection in supermarket transactions. Hence, advances in these fields are not discussed. This section is provides an overview on market basket analysis and multi-label classification in Sections 2.1 and 2.2 respectively.

2.1. Market Basket Analysis

Market Basket Analysis is a data mining method which is often used by large retailers to uncover relationships between items. The key concepts are frequent itemset mining (FIM) and association rule mining (ARM), both introduced by Agrawal, Imielinski & Swami in 1993 [31]. Frequent itemset mining aims at extracting frequently occurring itemsets in large transaction databases. For example, if 8% of baskets in a supermarket contain peanut butter and jelly and $\geq 5\%$ is stated as frequent, then this combination of articles is considered a frequent itemset. Association rule mining aims at discovering relationships between sets of items. It might be uncovered that most customers who purchase nachos also purchase avocados and jalapeños. Formal definitions, related concepts, and algorithms are discussed in Sections 4.3 and 4.4. Although market basket analysis can be used in various fields such as the biological field [13], medical purposes [10], drug-drug interactions [21], biodiversity indicators [36], tourism management [35], and insurance industry [53], this section focuses on related applications involving consumer shopping baskets.

2.1.1. Retail Applications

Instead of detecting dish types per delivery, Grive et al. research identifying the "shopping intention" or "shopping mission" behind a customer visit to convenience stores, supermarkets, and mini-hyper markets [27]. Their approach can broadly be described as: adjusting the product taxonomy to focus on relevant articles, removing irrelevant deliveries with too little or too much variety of the articles, and applying k-means clustering to group deliveries by shopping intention. They stress the large influence of the product taxonomy, which refers to the hierarchy of products. For example, the article "Orange Juice Brand X 330 ml" could be categorized under "Orange Juices", which could fall under "Juices" and then potentially under "Non-alcoholic beverages". Which level represents the article best? To optimize for the best results, the authors adjust the product taxonomy through a quantitative approach which takes into account the product variety and basket frequency as well as the business context and experts' opinions. In the next step cluster sampling is applied to eliminate baskets with too little or too much product variety. Then k-means clustering is implemented to segment customer visits. The results are evaluated on a business and a technical level. The results are com-

municated to a group of business expert, and the process is iterated on in terms of the product taxonomy. Custom category levels are created to improve the evaluation by the experts. For the technical evaluation, metrics like cluster's compactness and separation are computed. Some of the identified shopping missions are: meal preparation, breakfast, food and drink on-the-go, and extended visits around food.

Market basket analysis can also be applied on retail data to find customer profiles, improve the supermarket layout of physical stores, and select products to put on sale. For example, Miguéis et al. look into products that are frequently bought together and apply variable clustering in order to assign customers to a lifestyle segmentation [44]. Videla-Cavieres and Ríos extend this approach by performing masker basket analysis based on graph mining techniques [65]. Particularly by focusing on overlapping community detection, which is based around the idea of finding groups of strongly connected nodes in graphs and associating each node to one or multiple subgraphs. They show that this novel approach is successful for identifying communities of products with a meaningful interpretation such as "Grocery" and "Cookies Brand G".

Even though these papers are quite related to our problem, they focus on different characteristics of a shopping basket (or customer). For instance, a community such as "Grocery" or a segment such as "Meal preparation" do not specify any detailed information about what the customer cooked. As an argument to not generate association rules for their purpose, Videla-Cavieres and Ríos mentioned they found "meaningless rules or rules that apply only to a certain group of customers". The rule *coke* → *rum* is given as an example. The rule has a high support and confidence, but was only present in 0.15% of the transactions of two retail chains in Chile. The explanation that is shared for this pattern is that *ron-cola*, made of coke and rum, is a common drink in Chile. It does not match their purpose to define customer characteristics. However, we are interested in information such as this.

2.2. Multi-Label Classification

The problem of detecting dish types in deliveries is identified as a multi-label classification problem, in which an instance can be labelled with multiple classes. This section provides an overview of two multi-label classification tasks.

2.2.1. Multi-Label Text Classification

In multi-label text classification (MLTC) the most relevant labels are predicted for a given text document. Each document can be assigned to zero, one, or multiple labels. Examples of real-world applications are tag recommendation (in scientific bookmarking systems) [30], sentiment analysis (in social media text content) [61], and toxicity identification (in social media text content) [28]. Approaches for multi-label classification can roughly be split into two categories: binary-classifier based methods and global optimization methods [25].

Binary-classifier based methods transform the multi-label classification problem into several binary classification problems. There exist multiple approaches to achieve this, such as binary relevance, label power set, and classifier chains. For each separate problem a binary classifier can be applied, before merging all results. A prominent approach combination is to apply binary relevance with a strong binary classifier such as support vector machines [48]. Two disadvantages of binary-classifier based methods are that some do not make use of dependencies between labels, and that they becomes computationally expensive to train binary models when there are many classes [6]. Several multi-label text categorization algorithms have been proposed which do take these dependencies into account.

McCallum introduced a Bayesian classification approach where classes are represented by a mixture model [42]. Other approaches are BoosTexter [57], and Parametric Mixture Models for Multi-Labeled Text [64]. Later, BP-MLL, a back-propagation neural network adapted for multi-label classification problems, was introduced [77]. Nam et al. build upon BP-MLL and make some adjustments: replacing pairwise ranking loss with cross entropy and make use of rectified linear units (ReLUs), dropout, and AdaGrad [48]. The paper shows that state-of-the-art performance in large-scale multi-label text classification problems can be achieved by a neural network with just a single layer.

There are, however, difference between most multi-label text classifications problems and the problem which this thesis aims to solve The first difference is that in this thesis' problem only 23 class labels are identified as stated in 3.2. Additionally, making use of dependencies might not be beneficial due to the available dataset. There are other important differences between words in a document and articles in a delivery. Generally, there exist more words (including abbreviations, spelling mistakes, and slang) in the text documents than there are articles in a supermarket. It could be argued that combinations of articles represent meals

or dish types in the way that combinations of words represent topics or sentiment. However, a difference is that the order of words in a sentence or document conveys more information than the order of articles in a delivery. In addition, it is likely that there are more unrelated words in a document (e.g. “the") than unrelated articles in a delivery (e.g. dishwashing liquid).

2.2.2. Extreme Multi-Label Text Classification

Extreme multi-label text classification (XMC) is multi-label text classification with an extremely large number of class labels. This causes challenges such as data sparsity, strongly imbalanced datasets, and scalability. Approaches can be grouped into four categories: one-vs-all (OVA), partitioning methods, target-embedding methods, and deep learning [11]. The approaches are discussed in more detail. Since in the current problem definition dishes are grouped into dish types, the number of labels is not extremely large. However, when considering many specific dish types, extreme multi-label text classification might become interesting.

The one-vs-all approach works well in general but is not suitable for extreme multi-label text classification due to the large number of labels and the high computation time. Techniques are introduced to overcome this problem, such as primal and dual sparse approach (PD-Sparse) [70], parallel robust extreme multi-label class (PRoXML) [7], and distributed sparse machines for extreme multi-label classification (DiSMEC) [8]. The partitioning method either partitions the input space or the label space. Examples are Label Partitioning by Sublinear Ranking (LPSR) [68] and Multi-label Random Forest (MLRF) [1]. However, these approaches are also expensive to train. FastXML aims at training faster and predicting more accurately, and is considered the current state-of-the art tree-based XMC method. The general intuition behind it is that it learns a hierarchy over the feature space [49]. Embedding-based approaches represent the label space by a lower dimensional latent space, but have not outperformed the other approaches [11]. Finally, some recently developed deep learning methods are FastText (2016) [33], XML-CNN (2017) [38], AttentionXML (2018) [71], and X-Bert (2019) [11].

3

Data

This chapter covers the data that is related to the project. Section 3.1 describes how the data is obtained and structured. Secondly, the class labels and the process of how they were chosen are explained in Section 3.2. Lastly, an exploratory data analysis on the delivery data and the labels is performed and presented visually in Section 3.3.

3.1. Data Overview

This project uses various data entities including data on deliveries, articles, customers, and recipes. The recipes data is scraped from websites, but all other data comes from Picnic's high quality, anonymized data warehouse. First, Section 3.1.1 describes the most relevant data, which are the deliveries placed via the Picnic app. For some of these deliveries weak labels are obtained. This process is reported in Section 3.1.2. Additional data from Picnic's data warehouse, such as information about articles and customers, and the recipe dataset are described in Sections 3.1.3 and 3.1.4 respectively. The Frequent Itemset Model described in Chapter 4 makes use of unlabelled delivery data and recipe data. The Supervised Learning Model from Chapter 5 uses labelled delivery data as well as additional feature data.

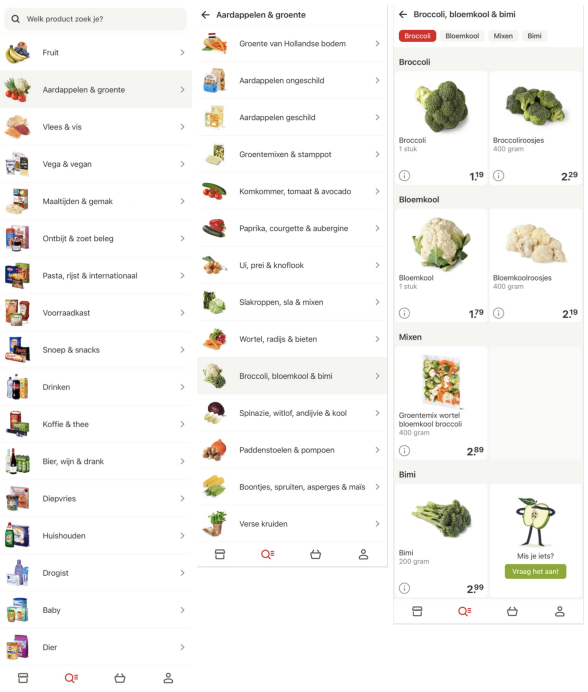

Figure 3.1: Screenshots of a selection of the category tree page in the Picnic app.

3.1.1. Deliveries

Deliveries, more specifically the articles which they contain, are certainly relevant when creating a model to detect dish types in deliveries. Hence, this section briefly describes the context in which deliveries are created and the manner in which they are organized. Picnic customers browse through the app to add articles to their basket. At the moment, articles can be selected 1) by clicking through the article categories in the "category tree page" (Figure 3.1), 2) by using the search bar to find a particular article, 3) through the "previously bought articles" page, or 4) through the several theme pages. Examples of theme pages are "barbecue weather" and "cocktail hour". Customers can place several orders until the evening before the groceries are delivered. A single order or multiple orders combined for the same delivery date are referred to as a delivery.

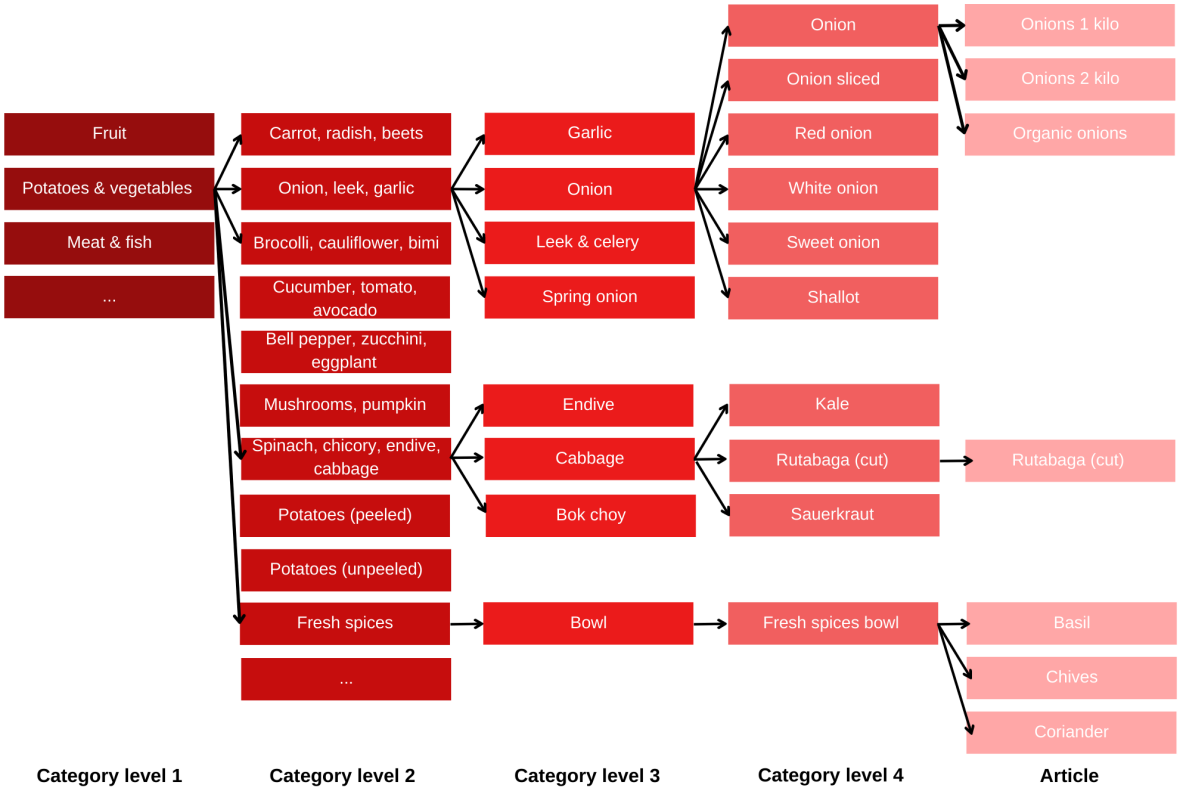

Figure 3.2: Subset of Picnic's product taxonomy.

Articles in the (online) store are being updated frequently. Articles are removed, new articles are added, prices change, etc. This is important to keep in mind when developing a model. If a model places much emphasis on a particular article, its performance will be extra vulnerable to changes in the assortment. Therefore, it is good to know that Picnic has articles already categorized and hierarchized in their product taxonomy. Individual articles are categorized over four different levels. Category level 1 is the least granular level and category level 4 is the most granular level. Figure 3.1 shows how the articles are ordered in the app based on these category levels. The complete product taxonomy has not been added to this thesis due to its size. However, an intuition of how it is structured can be obtained through Figure 3.2, which displays a small subset of the product taxonomy from category level 1 until the individual article level. Additionally, Table 3.1 shows the different category levels of the articles in an example delivery in which the recipes "Indiase rode linzenstoof met bloemkool en naan" and "Tagliatelle met broccoli, courgette en geitenkaas" were ordered.

Article	Category level 4	Category level 3	Category level 2	Category level 1
Mini krieltjes met bistro kruiden	Aardappelen	Krieltjes	Aardappelen geschild	Aardappelen & groente
Bloemkool	Bloemkool	Bloemkool	Broccoli, bloemkool & bimi	Aardappelen & groente
Broccoliroosjes	Broccoli	Broccoli	Broccoli, Bloemkool & bimi	Aardappelen & groente
Courgette	Courgette	Courgette	Paprika, courgette & aubergine	Aardappelen & groente
Heel boeren zonnepit brood	Brood bruin heel	Bruin	Brood	Brood & banket
Afbak Italiaanse bollen	Broodjes afbak	Broodjes	Zelf afbakken	Brood & banket
Perensap	Vruchtensap	Vruchtensap	Sappen & smootjes	Drinken
Bananen	Bananen	Bananen	Bananen, appel & peer	Fruit
Pitloze rode druiven	Druiven	Druiven	Kiwi, drijf, mango & exoten	Fruit
Toiletpapier 4 laags	Toiletpapier 4-laags	Toiletpapier	Papier & tissues	Huishouden
Zachte geitenkaas 50+	Buitenlandse kaas geit	Geit	Kaasspecialiteiten	Kaas
Bio belegen kaas plakken 48+	Plakken kaas belegen verpakt	Belegen	Plakken kaas	Kaas
Smeerkaas 20+	Smeerkaas naturel	Smeerkaas	Smeerkaas & zuivelspread	Kaas
Medium roast koffiebonen	Koffiebonen regular	Regular	Koffiebonen	Koffie & thee
Courgettesoup verspakket	Verpakket soep zomer	Soepen	Verspakketten	Maaltijden & gemak
Krokante muesli appel	Cruesli vruchten	Krokante muesli	Muesli & granola	Ontbijt & zoet beleg
Kokosmelk light	Kokos oosters	Kokos	Oosters	Pasta, rijst & internationaal
Tagliatelle no. 304	Tagliatelle pasta	Tagliatelle	Pasta	Pasta, rijst & internationaal
Naan brood knoflook koriander	Indiaas koken	Indiaas	Surinaams & Indiaas	Pasta, rijst & internationaal
Reuzenmergpijpen	Bakkerskoek mergpijp	Bakkerskoeken	Koek & cake	Snoep & snacks
Gedroogde rode linzen	Peulvruchten gedroogd	Gedroogd	Groente in pot & zak	Voorraadkast
Kipfilet blokjes	Kip blokes & reepjes	Blokjes & reepjes	Kip & gevogelte	Vlees & vis
Kipfilet	Kip	Kip	Gesneden vleeswaren	Vleeswaren, spreads & tapas
Garam masala kruidenmix	Kruidenmix orientaals	Orientaals	Kruidenmix & smaakversterker	Voorraadkast
Olijfolie traditioneel	Olijfolie standaard	Olijfolie	Olie, zuren & azijn	Voorraadkast
Verse slagroom	Slagroom	Slagroom	Room & crème fraîche	Zuivel & eieren
Magere yoghurt	Yoghurt naturel	Natural	Yoghurt	Zuivel & eieren

Table 3.1: Category levels of the articles in an example delivery.

3.1.2. Labels

Ideally, it would be known of deliveries which dish type is cooked using the articles in that delivery. This would facilitate the development of a dish type detection model and allow the results to be evaluated. However, such a dataset is unavailable and difficult to create due to two requirements. The first requirement is that the labelled data needs to be accurate and represent the ground truth. Manual labelling is subjective in this context and hence not a solution. The second requirement is that a lot of data needs to be available. An exact number is undefined beforehand, but one can imagine that e.g. 100 instances is hard to generalize over. Limited resources did not enable interviewing customers one by one.

Fortunately, within the first month of the research project Picnic launched a new feature which displays recipes in the app. Throughout the report, this feature is referred to as the “recipe page”. The data has been structurally gathered as of November 30, 2021. Every week Picnic releases new recipes. As shown in Figure 3.3 a recipe contains a name, preparation steps, an ingredient list. It also contains descriptive information such as an image, the suggested number of people the quantities are used for, the estimated time to cook the dish, and the price of the dish per person. This price can be calculated because the ingredients are linked to articles in the online Picnic store. This also enables customers to add the articles directly to their shopping basket. All items can be added at once or individually. The ingredient list consists of the core ingredients (header “Boodschappenlijst”), and in some cases ingredients for a variation of the recipe (header “Variatietip”) as well as basic ingredients which are commonly present in kitchens (header “Nog niet in huis?”). Whenever a customer visits the page of a recipe, adds at least one of the ingredients to their basket, and also purchase that article, the customer receives an email containing the recipe.

In the background, this information is stored as well. For each delivery it is known if an article was purchased in a recipe context, and for which recipe. Hence, this feature provides links between deliveries and recipes. Since a recipe also falls under a dish type, this links deliveries and dish types as well. As a result, the recipe feature page can be used to label deliveries with dish types. Even though this process can be considered a strong signal for the label, there is no guarantee of all labels being correct. Technically, a customer could have accidentally added the ingredient, or purchased the ingredient without intending to cook the recipe. Still, the recipe page feature unexpectedly provided a way to obtain labelled data while meeting the two requirements.

The recipe page feature has been live since November 2021 and new data is gathered every day. However, the dataset that is used in this project only contains deliveries which were delivered between November 30, 2021 and June 18, 2022.

One critical note is that the resulting dataset is different in several ways compared to a random set of (labelled) deliveries. Firstly, in this dataset customers cook exactly the same recipes from a limited selection of 119 recipes. In an ordinary situation customers would not cook for example a lasagne using articles from

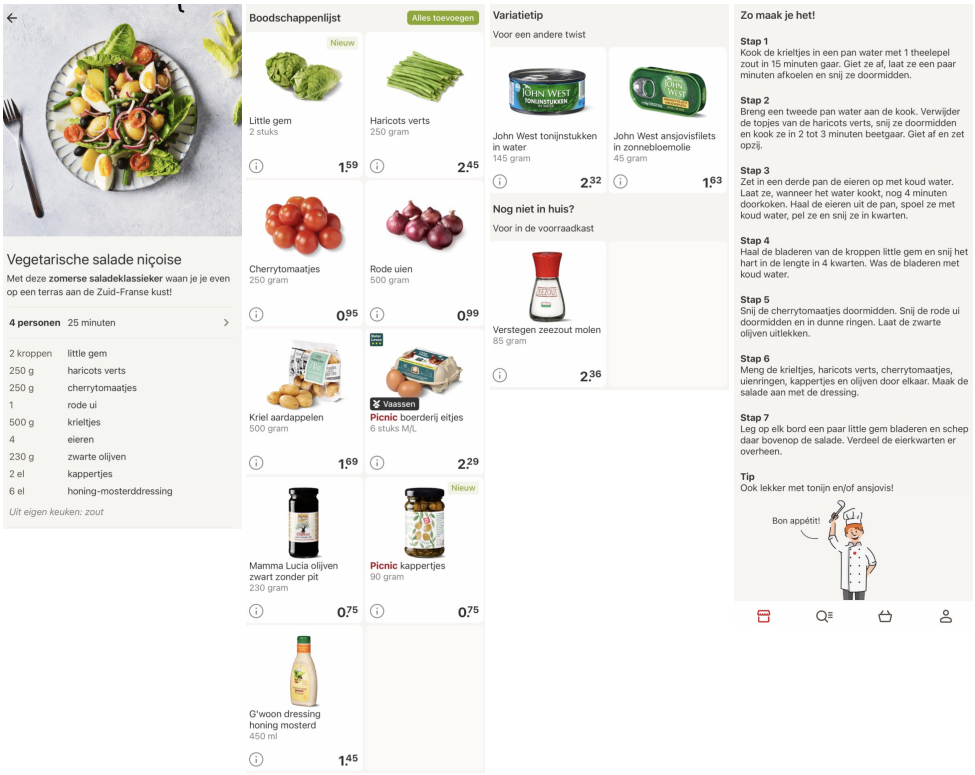

Figure 3.3: Screenshots of a recipe in the Picnic app.

precisely the same ingredient list. Secondly, labels are missing from this dataset. The deliveries have been labelled with Picnic recipes but any other dishes that customers cook from these deliveries are not known and hence not labelled. In addition, specific recipes are always purchased in a certain period. This is because customers can only purchase the recipes when they were visible in the app. A fourth concern, as previously mentioned, is that the labels are weak and do not represent the ground truth. Finally, the dataset is imbalanced because recipes of certain dish types are selected and purchased more often. All together, the available dataset is not a reflection of an ordinary situation. This is expected to be a challenge for the creation and evaluation of a model which makes use of this dataset.

3.1.3. Additional Data

The Supervised Learning Model described in Chapter 5 uses primarily labelled delivery data, but also considers other data as potential features. The feature selection process is described in Section 5.2.1. This section briefly describes the data used in that section. The data includes information about articles, customers, and deliveries.

Information about articles that is used is the popularity of an article and the freshness of an article. The former is measured through “article penetration”. This value is calculated for a specific article by dividing the number of deliveries which contain that article by the total deliveries. The multiplication is not necessary but does make the value easier to comprehend. “Personal article penetration” calculates the article penetration of a specific article for a specific customer. “General article penetration” calculates the article penetration of a specific article over all customers. The freshness of an article is obtained through two data points. The first is whether the freshness guarantee of an article is shown in the Picnic app. The second is the number of days the articles is guaranteed to be fresh as stated by a supplier.

The customer information that is considered includes the “household type“ of a customer, the average number of days between deliveries of a customer, and the total number of deliveries of a customer. The last two are clearly defined. The “household type” groups customers into one of several types. The five types are: single, couple, family, other, and unknown. These categories are derived from the customer’s settings in the app where can be stated how many adults, children, dogs, and cats are part of the household. Note that this information is unvalidated and is optional for customer to provide.

The delivery-level data that is considered concerns the number of distinct category levels present in the delivery. This is an indication of the variety of the products. For each delivery it is available how many distinct level 1, 2, 3, and 4 categories it contains.

3.1.4. Scraped Recipes

As described in Section 4.6 the Frequent Itemset Model makes use of a recipe dataset. This dataset is obtained by scraping various Dutch recipe websites. It contains 12,144 recipes, including the Picnic recipes. For each recipe the name of said recipe, the ingredient list, and URL is obtained. Some websites have descriptive fields, such as dish type tags, for each recipe. However, these dish type tags are not in line with the class labels defined in Section 3.2 and they are not present for all recipes. Hence, these tags have not been used.

3.2. Classes

This section clarifies the class labels and thus the output of the dish detection model. Any dish related to breakfast, sweets, desserts, snacks, drinks, sauces is left out, since only main courses are considered. Despite this scope there still exist too many variations of unique dishes to consider all. Hence, dishes are categorized into a selective set of dish types as shown in Table 3.2. In most cases it is likely trivial what is meant with a dish type, but in other cases it might not be clear what is meant (e.g. *flatbread* and *grains*). Hence, the table includes a description for each dish type. The process of categorizing dishes is utterly subjective. The aim has been to scope the problem and find the best set considering a number of factors. The factors which are taken into account are: the taxonomy of recipes websites, the Dutch cuisine, the availability of labelled data, the input of stakeholders, and (subjective) common knowledge. The rest of this section dives into the process and considerations of defining these dish types.

An important choice is the level of detail in which the dish types are defined. In an ideal situation each dish can be detected in detail. This would include traditional foods such as boeuf bourguignon, pad thai, and pasta carbonara. It would also include detect less traditional food and be able to describe specific combinations in detail. Examples are a lasagne with salmon and zucchini or a risotto with mushrooms and bacon. However, as described before, the endless combinations complicate the problem. The lasagne with salmon and zucchini might also include garlic, broccoli, and peas. Should all ingredients be listed or only key ingredients? How are key ingredients defined? Is the prediction incorrect if the peas are not detected? If a customer intends to cook a pasta with broccoli and a salad with eggplant, how can the model detect this combination instead of detecting a pasta with eggplant and a salad with broccoli? If a customer cooks pasta carbonara with cream and vegetables, is it still considered a pasta carbonara? To clarify and simplify the problem, a comprehensible number of general dish type, based on distinguishable features of dishes, are created.

Taking inspiration from recipes websites, some of these general dish types are considered to be clearly defined and distinguishable from other dish types. Examples are *curry*, *pizza*, and *soup*. However, there remains a lot of gray area. Should *curry*, *risotto*, *paella* be different classes, or should they fall under dish type rice? Why split potato dishes across dish types *potato*, *potato casserole*, and *stamppot*? Which dish type does a chili con carne or carpaccio belong to? Hence, second aspect which is taken into account are the eating habits in the Netherlands. In Dutch culture it is not uncommon to eat fries with fried snacks or pancakes for dinner. Another traditional Dutch dish is “stamppot”. Due to the popularity of these foods, separate dish types are created. The next factor which has been of influence in these considerations is the availability of labelled data, as Section 3.1.2 describes. Dish types are only considered if there are Picnic recipes related to it. This is the case for *risotto* and *paella*, but not for *stew*, *fries* and *pancakes*. This narrows down the list of potential classes. The fourth aspect which has been taken into account are the preferences of the business stakeholders at Picnic. Given the available labelled data and intuition about how distinguishable the dish types are, the dish type were as detailed as possible. As a result, dish type *gnocchi* for example became a separate class. The final decisions around creates class labels was based around common knowledge and a subjective preference of the author. The resulting class labels from Table 3.2 scope and state the problem. This is an essential step which influences the results. It should be noted that the classes can be altered or iterated over based on the results in this work.

Classes (dish types)	Description
curry	Sauce seasoned with spices, associated with Southeast Asia and commonly eaten over rice and/or naan.
flatbread	Middle Eastern-inspired dishes where the main ingredient is a pita bread, Lebanese wrap, etc. Usually also contain hummus, falafel, tahini, shawarma, etc.
gnocchi	Small dough lumps, often created of wheat flour, egg, salt, and potato. Officially not a type of pasta.
grains	Refers to bowls or salads where couscous, bulgur, or quinoa is the main ingredient.
hamburger	Although variations exist, usually a bun or bread roll filled with meat (alternative), cheese, lettuce, vegetables, and sauces.
lasagne	Although officially a type of pasta, a layered, baked dish usually containing cheese, meat, and vegetables.
nasi	Refers to the dish nasi goreng, which is commonly eaten in the Netherlands. Usually made out of fried rice, meat, vegetables, eggs, and satay sauce.
noodles	Dishes where noodles are the main ingredient. This includes noodle soups.
omelet	Any dish with eggs as the main ingredient, such as shakshuka and frittata.
paella	Rice dish which commonly includes ingredients such as saffron, vegetables, chicken, and seafood, and is served in one pan.
pasta	Dishes where any type of pasta (orecchiette, pappardelle, tagliatelle, etc.) is the main ingredient. This includes pasta salads.
pizza	Traditional Italian dish consisting of round wheat-based dough usually with tomatoes, cheese, and other ingredients on top.
potato	Includes dishes with potato a key ingredient and served separately from other ingredients (contrary to stamppot and potato casserole). This includes the typical Dutch dish of potatoes, meat, and vegetables.
potato casserole	Dishes with potato as a key ingredients and mixed with other ingredients. Usually prepared in the oven, and made with cream and cheese.
quiche/tart	A savoury pie made out of pastry crust usually filled with cheese, vegetables, and meat.
rice	Dishes where rice is the main ingredient but which do not fall under curry, paella, or risotto.
risotto	Creamy dish made out of rice cooked with broth. Common ingredients are white wine, parmesan cheese, and onion.
salad	Usually a mixture of (green leafy) vegetables, and other ingredients such as fish, meat, cheese, served with a dressing. Does not include e.g. couscous salads, pasta salads, or potato salads.
soup	Generally, a liquid served warm (can be served cold e.g. gazpacho). It combines meat, fish, and/or vegetables with stock, water and/or milk.
spring roll	Vegetables, rice vermicelli, and fish or meat wrapped in rice paper or dough. Examples are fresh spring rolls as well as fried spring rolls.
stamppot	Traditional Dutch dish made from of mashed potatoes and vegetables, usually served with meat, such as a sausage.
sushi/poke bowl	This dish type include Japanse sushi and Hawaiian poke (bowl). Ingredients usually (raw) seafood, nori, and vegetables such as avocado, cucumber, seaweed, and edamame.
wraps	Usually Mexican dishes where the main ingredients are wraps, tacos, tortillas, and nachos. This includes burritos and quesadillas.

Table 3.2: Overview of classes/dish types.

3.3. Exploratory Data Analysis

The labelled delivery data (Section 3.1.2) is explored through visualizations in order to get a better understanding of the problem. Note that this section only concerns deliveries which are labelled through the recipe page feature, and contain at least one article of a Picnic recipe. The conclusion might very well be different for “true” data. This section focuses on the number of labels in general, the number of labels per instance, the variety between labels, label co-occurrence, and the “confidence" with which deliveries are labelled.

The figures can be found in the confidential version of the thesis.

3.3.1. Many Labels

The figure shows the absolute and cumulative number of labelled deliveries per day. The recipe page is used frequently, and thus a large number of labels is available. The dataset was frozen on June 18, 2022 but new labels are generated daily.

3.3.2. A Light Multi-Label Problem

Although customers can order any number of recipes, the figure shows that if a customer makes use of the recipe page, in most cases a single recipe is ordered. The problem does remain a multi-label classification problem. Note that customers might have intended to cook other dishes besides the recipes offered in the app. This data, however, is unavailable.

3.3.3. Imbalanced Dataset

The first figure displays the number of deliveries for each dish type. Clearly, the dataset is imbalanced. The second figure shows the number of recipes that have been live in the recipe page feature per dish type. For instance, if there would be a recipe for “quiche Lorraine” and “onion tart”, then there would be (at least) two recipes for dish type *quiche*. The figure makes clear that more recipes have been published for certain dish types than for others.

3.3.4. Biased Label Co-Occurrence

The first figure visualizes the proportion of deliveries per dish type per week. It becomes clear that it is not the case that recipes of all dish types are always available in the app. Recipes of some dish types are present each week, but others are only available occasionally. This very much influences the label co-occurrence matrix as shown in the second figure Although it is the case that certain dish types are often purchased within the same delivery, this provides no guarantee for any future deliveries. The label co-occurrence is mainly an effect of which recipes are live when, which is a choice made by the responsible team within Picnic. It is not because customers who for example love to eat pasta, love to eat noodles as well. Such a relationship could still be possible, but it cannot be detected by the current labelled dataset.

3.3.5. Weak Labels

Finally, it is explored how many of the articles of the recipes are purchased. The final figure shows this distribution. The number of single overlapping distinct articles strikes out. In principle all labels are weak. The usage of the recipe page feature is a strong indicator, but none of the labels are confirmed. However, the recipes for which a single distinct articles was purchased can be considered the weakest labels. Note that the values are not normalized in relation to the number of distinct ingredients in the recipes.

4

Frequent Itemset Model

This chapter presents the methodology of the first model, the Frequent Itemset Model, which aims at detecting dish types without the use of supervised learning techniques. An overview of the model is presented at the start of the chapter. The subsequent sections dive deeper into each stage of the model. Section 4.7 provides a summary of the Frequent Itemset Model.

4.1. Overview

Without labelled data it is not an option to feed a model examples it can learn from. Instead, we rely on discovering patterns within the data and transforming these insights into correct conclusions. In a way we are applying phenomenal data mining, a term created by McCartney in 2000 which "finds relations between the data and the phenomena that give rise to data rather than just relations among the data" [43]. As depicted in Figure 4.1, the model consists of multiple stages which are summarized into three phases: 1) pre-processing, 2) obtaining itemsets, and 3) matching these itemsets to dish types. This section briefly describes these phases and the intuition behind them. It also presents the intermediary data of different stages of a sample run.

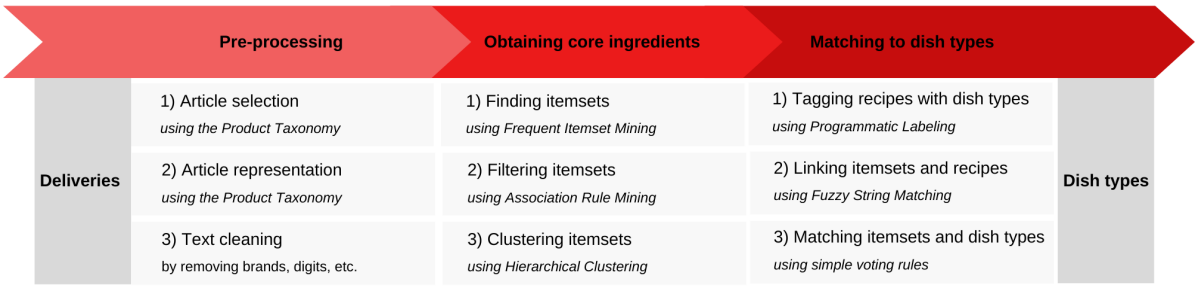

Figure 4.1: Overview of the Frequent Itemset Model.

The Frequent Itemset Model makes uses of the unlabelled delivery dataset, also called transactions, and the recipe dataset. The pre-processing phase consists of three steps: 1) removing irrelevant articles (e.g. cat food), 2) selecting the representation of articles (e.g. tomatoes instead of organic cherry tomatoes 250 grams, and 3) filtering the names of articles in the delivery dataset and ingredients in the recipe dataset (e.g. garlic instead of clove of garlic).

FIM results per step

Raw deliveries (20,000 deliveries)

Key	Delivery
1	{kruidcake mix, cola zero, bio gember, ontbijtspek, ijsthee zero sparkling, toiletblok powerball lavendel, kerriesaus mix, brokjes in gelei adult kalf lever, openhaardblok, lucifers extra lang, keukenpapier 3 laags, mergkoekjes,
2	salami, pangalicious knoflook kruiden} {crème fraîche, bakkers kokosmakronen, augurken zoetzuur middel, half tostibrood wit, zilveruitjes zoetzuur, regular mini, flammkuchendeeg, bio kefir, mini stroopwafels, jonge kaas plakken 48+, afwas middel radijs & bergamot, gingerbeer, Chianti Sangiovese, achterham, ananas geschild, gegrild spek, mini gevulde koek}
…	…

Pre-processed deliveries (19,952 deliveries)

Key	Delivery
1	{gember, ontbijtspek, kerriesaus, salami, visfilet witvis}
2	{crème fraîche, augurken, brood, zilveruitjes, pizzadeeg, plakken kaas, ham, ananas, spek}
…	…

Frequent itemsets (2,019,074 itemsets)

Minimum support = 0.001

Support	Itemset
0.54	{melk}
0.42	{bananen}
…	…
0.24	{melk, bananen}
0.0022	{paksoi, rijst}
0.0016	{mango, tomaten, eieren, gehakt}
0.0016	{kaas, paprika, mango, gehakt, tomaten}
…	…
0.001	{bakmix brood, bananen}

Filtered, frequent itemsets (7,453 itemsets)

Minimum lift = 5

Lift	Support	Itemset
338.15	0.0011	{kipvleugels, kip poten}
131.88	0.0010	{hollandaise saus, asperges}
81.71	0.0016	{kalfsragout, pastei bakjes}
…	…	…
8.89	0.0036	{knoflooksaus, shoarmareepjes}
5.28	0.0247	{wraps, kruidenmix mexicaans, maïs, avocado}
5.17	0.0195	{tomatenblokjes, wraps, kruidenmix mexicaans, maïs}
…	…	…
5.00	0.0026	{pijnboompotten, basilicum}

Clustered, filtered, frequent itemsets

Linkage method = single, cluster number estimation method = Davies-Bouldin Index

Itemset
{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
…

Recipe matches

Dish type	Recipe	Itemset
wraps	Guacamole wraps met tomatensalade	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
wraps	Mexicaanse enchilada's uit de oven	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
wraps	Breakfast burrito	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
wraps	Tostada's met vis en avocado	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
pizza	Tortillapizza's	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
salad	Salade van tomaten en avocado	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
…	…	…

Dish type matches

Dish type	Itemset
wraps	{wraps, kruidenmix mexicaans, maïs, avocado, tomatenblokjes}
…	…

Table 4.1: Selection of results of the different stages of the Frequent Itemset Model.
(single run on 20,000 samples with hyperparameters as shown in the table headers)

After pre-processing the articles, the delivery dataset consists of sets of articles which can be used as ingredients for dinner and are purchased in the same delivery. In order for this model to discover patterns to recognize dish types, a couple of assumptions are made. A first straightforward assumption is that as a result of a customer intending to make a dish, a (sub)set of ingredients for that dish is added to the same delivery. Since many customers follow this pattern, (sub)sets of ingredients of various dish types are expected to be present among deliveries. Although customers add different (sub)sets of ingredients to their deliveries to create the same dish type, an overlap between "core ingredients" is quite probable. Hence, the assumption is made that the same set of core ingredients for a particular dish type is present among multiple deliveries. To obtain core ingredients during this second phase of the model, three techniques are applied: 1) frequent itemset mining, 2) association rule mining, and 3) hierarchical clustering. Using frequent itemset mining the model is able to find sets of articles that occur in numerous deliveries. This is also commonly referred to as frequent itemsets. The FP-Growth algorithm is applied to obtain the frequent itemsets efficiently. Association rule mining is applied using the lift metric to filter thes itemsets with the aim of only keeping itemsets which most likely contain core ingredients. Hierarchical clustering is used to join closely related itemsets. The result of this phase are frequent itemsets which are filtered and clustered, and that represent sets of core ingredients.

A human could easily link these itemsets to dish types, but the model is unaware of which dish type for example itemset {wraps, avocado, tomato, corn} belongs to. Hence, the third phase of the model aims at matching itemsets to dish types by making use of the recipe dataset. This is achieved in three steps: 1) tagging recipes with dish types, and 2) matching itemsets to recipes, and 3) dish type voting.

These three phases link dish types to itemsets which are linked to deliveries. Hence, the model is able to detect dish types in deliveries and, as an extra feature, identifies the articles a prediction in based on. Table 4.1 displays the different steps of a sample run. The remainder of this section is dedicated to describing the workings of the Frequent Itemset Model in more detail.

4.2. Pre-processing

The unlabelled delivery dataset (Section 3.1.1) and the recipe dataset (Section 3.1.4) are pre-processed with the aim of representing the articles in the deliveries and the ingredients in the recipes in such a way that dish types can be detected more easily. The following steps are applied to the delivery dataset. First, unrelated articles are removed, as described in Section 4.2.1. Then a category level representation is selected for all remaining articles through the product taxonomy, as described in Section 4.2.2. Finally, the category level representations are cleaned as stated in Section 4.2.3. The final step of cleaning the text is also applied to the ingredients in the recipe dataset.

4.2.1. Article Selection

Picnic has a large assortment and thus the articles in the delivery dataset contain a large variety of products. It can be assumed of certain products that they are not purchased as ingredients to create a meal. A selection of category and article levels is made to be filtered out from each delivery. A subset of this selection in visible in Table 4.2. If a category level is removed, all of the more granular category levels and articles which fall under that category level are removed as well. For instance, level 1 category "Dier" (includes articles for pets) is filtered out thus level 2 categories "Cat" and "Dog" are also removed automatically. Hence, only a handful of level 4 categories and individual articles present in Table 4.2. Note that it can happen that all articles of a delivery are removed. This is the case for 48 out of the 20,000 deliveries in the example of Table 4.1.

Category level 1	**Category level 2**	**Category level 3**	**Category level 4**	**Article**
Baby	Bakkersspecialiteiten	Banket	Toetjes vers	Avocadosnijder
Bier, wijn & drank	Beschuit	Cake		Speculaaspop glutenvrij
Dier	Candybars & chocolade	Cup-a-soup		
Dranken	Chips & popcorn	Fruitbiscuit		
Drinken	Cornflakes & kinderontbijt	Gebak		
Drogist	Diepvriesmaaltijden	Groentechips		
Huishouden	Drinkontbijt & probiotica	Haverrepen		
Koffie & thee	Drinkyoghurt	Ijsstam		
	…	…		

Table 4.2: Subset of categories and articles which are filtered out from the deliveries during the pre-processing phase.

The selection processes is performed in multiple stages. The first straightforward categories and articles to remove are non-food, and food meant for babies and pets. Since the problem focuses on dishes for main

courses, categories and articles related to breakfast items, bakery items, snacks, and sweets are removed. Some drinks, such as beer and wine, can be used as ingredients, but the decision is made to remove these articles. The reasons are that most recipes do not include drinks as ingredients, and customers mostly purchase drinks with the purpose of drinking instead of cooking. Any ready-made meals, frozen pizzas, and fresh packages are also removed. The latter is a package which contains the core ingredients to cook a recipe (referred to as “verspakketten"). Although the fresh packages are used to create a recipe, they are left out since they do not represent individual ingredients and can simply be queried separately to link the fresh package and respective dish to deliveries.

4.2.2. Article Representation

After removing irrelevant articles from the deliveries, the following step is to decide how the articles are represented. That is, deciding if an article should be represented on the article level or on one of the category levels it falls under within the product taxonomy. These decisions are made while balancing between generalizing over the articles and choosing a representative level of the article.

More general representations of articles ensure that articles are grouped together and that these representations are present in more deliveries. For instance, there are more articles which fall under level 3 category “tomato” than which fall under level 4 category “cherry tomato”. A higher representation makes it easier to obtain frequent itemsets. However, a higher representation might not always be a good representation of the article. A small part of the product taxonomy is visualized by Figure 3.2. This figure clearly shows that there is not one category level representation which works the best to represent all articles. For instance, the article “Onion 1 kilo” can be represented by level 3 category “Onion”. However, representing “Rutabaga” by its level 3 category “Cabbage” loses information about the article type. Category level 4 is a better option in this case. Finally, the different spices can only be distinguished on article level. Hence, the representation for each individual article is selected manually based on two factors. The first factor is the author’s perspective on which level depicts the article best. The second factor is the basket frequency of each article, which indicates how often an article is purchased. In cases that it is unclear which level fits best, the popularity of the article is taken into account. Then a more generalized representation is favoured if the article is not purchased relatively frequently. This selection process is not iterated over, nor reviewed by experts. How articles are represented highly influences which frequent itemsets are obtained.

4.2.3. Text Cleaning

The text cleaning process is applied to the articles in the delivery dataset (after the previous two steps) and to the recipe names and ingredients in the recipe dataset. Firstly, the text is normalized by removing capitalization. Secondly, punctuation marks and other characters are removed and replaced by a space. Then brand names are removed. Brand names are queried from Picnic’s Data Warehouse and some are manually added. There are also brand names which are explicitly not removed, because they would remove the core description of the article/ingredient. Examples of such brands are “Vis Marine”, “The Ketchup Project”, and “Blije Kip”. In the fourth step stop words, words that do not contribute to the meaning of the article/ingredient, are removed. There exist multiple datasets of stopwords since this is a common step within Natural Language Processing (NLP). Examples of popular stopwords are “de” (the), “een” (a), and “ook” (also). However, in this particular context is different from a regular text paragraph and additional words need to be removed. One example is “pitloze druiven” (seedless grapes), where the core word is grapes and it is irrelevant that they are seedless. Other examples are “vers” (fresh), “oma’s” (grandmother’s), and “gezonde” (healthy). Within the recipe dataset there also many adjectives to describe a preparation step in the ingredient list, such as “gesnipperde ui” (chopped onion). These stopwords are identified manually and removed from all strings, independent of the other words in the text. For that reason some issues are identified. For instance, with the word “gehakt”. In Dutch this words refers to the word minced meat but also to the word chopped. Ideally, we would like to remove the word from “amandelen gehakt” (chopped almonds) but not from “gehakt” (minced meat). Another example is “zure” (sour). We would like to remove this word in case of “zure appels” (sour apples), but not in case of “zure room” (sour cream). Depending on the situation, the stop word is either included in all texts or excluded completely. Lastly, any digits are removed.

4.3. Frequent Itemset Mining

As explained in Section 4.1 a subgoal of the model is finding core ingredients. These core ingredients arise from patterns in the data and are sets of closely related articles which are present in multiple deliveries. The

first step is to obtain sets of articles which are present in multiple deliveries without taking into account how closely related the articles are yet. This task is solved by frequent itemsets mining, which aims at extracting frequently occurring events, patterns, or items in data [40]. In this context a frequent itemset is a set of articles which occurs in at least x% of the deliveries. The value of x defines the term "frequent" and is also known as the support. The rest of this section formally states the problem, the complexity, and provides an overview of algorithms to obtain frequent itemsets. Some algorithms are explained in more detail, and the FP-Growth algorithm is selected for the model.

4.3.1. Definitions

The problem of frequent itemset mining was introduced by Agrawal, Imielinski, and Swami in 1993 [31]. Frequent itemset mining aims at finding sets of items, or itemsets, which occur at least as frequently in transactions as a predetermined minimum support. The support for an association rule is the fraction of transactions in D that satisfy the union of the consequent (A) and antecedent (B) of the rule (A ⇒ B). Association rules are introduced in more detail is Section 4.4. The context of frequent itemset mining, its goal, and the definition of support are formally described as follows.

Formal model. Let $I = i_1, i_2, ..., i_m$ be a set of items. Let D be a database consisting of transactions. Each transaction T is a non-empty set of items such that $T \subseteq I$. Let X be an arbitrary set of items. A transaction T is said to contain X if $X \subseteq T$.

Frequent itemset mining task. Given a database D consisting of transactions or sets of items, find all itemsets which are frequent in D given a minimal support threshold.

Support. $support(A \Rightarrow B) = P(A \cup B) = \frac{Transactions\ containing\ both\ A\ and\ B}{Total\ number\ of\ transactions}$, range: $[0, 1]$

Anti-monotone property. If an itemset is not frequent, then its supersets are also not freqeunt. If an itemset is frequent, then its subsets are also frequent.

Apart from frequent itemsets there are also more specific types of frequent itemsets: closed frequent itemsets and maximal frequent itemsets. Frequent itemsets meet the criteria of a minimum support value. Closed frequent itemsets are frequent itemsets which are also closed, meaning that there is no immediate superset with the same support value as the original itemset. For instance, itemset {apple, banana} with support 5 is not a closed frequent itemset if there exists an itemset {apple, banana, pear} with support 5 (and the minimum support is 4). Maximal frequent itemsets are subsets of closed frequent itemsets. A maximum frequent itemset is an itemset if it does not have immediate supersets which are frequent. In this model we collect all frequent itemsets, not just closed or maximal ones. The reason is that we want to consider each itemset. Closed and maximal itemsets tend to filter out smaller itemsets, based on the value of support but not based on how related the items within the itemsets are. It could then for instance happen that itemset {lasagne sheets, mozzarella} with support 100 is removed since there also exists an itemset {lasagne sheets, mozzarella, banana} with support 100. However, when the itemsets are filtered in Section 4.4, the latter itemset is filtered out, while the former itemset is not.

4.3.2. Problem Complexity

The problem of finding frequent itemsets is computationally challenging due to number of candidate itemsets that have to be considered and the main memory that is required [40]. Suppose there are n items in a dataset. For each itemset of size k, with $k \leq n$, there are $\binom{n}{k} = \frac{n!}{k!(n-k)!} = 2^n - 1$ candidate itemsets to consider. This motivates the usage of an efficient algorithm.

4.3.3. Exhaustive Search Algorithms

The same paper which introduced the problem of frequent itemset mining also provided the first algorithm to solve this problem, namely the Apriori algorithm [31]. It is very likely the most-well known algorithm in its field. Many improvements, in terms of time complexity, have been proposed since it was first introduced. Apart from algorithmic solutions, also parallel and distributed computing have been utilized to mine frequent itemsets faster. This section aims at providing an overview of algorithmic solutions for the frequent itemset mining problem. Non-exhaustive search algorithms such as the genetic algorithms Alatas [4], QuantMiner [55], and the genetic programming algorithms G3PARM [39] and G3P-Quantitative [41] are not included.

Such algorithms do not guarantee that the whole search space is explored and are less popular within the research community [40]. Table 4.3 presents an overview of exhaustive algorithms to find frequent itemsets. Note that this overview does not contain all algorithms which have been developed over the years. Many approaches are extensions of other solutions. Apriori, FP-Growth, ECLAT, and dECLAT are identified as the main algorithms. Apriori is the first algorithm, and the other three have achieved the largest improvements. All four are often present in comparative studies [40].

Algorithm	Year	Description
Apriori [2]	1993	The Apriori algorithm applies levelwise breadh first search methodology and makes use of the anti-monotone property. It first evaluates of all sets of individual items if they are frequent itemsets. With the remaining frequent itemsets it creates new candidate itemsets for the next level (an additional item in the set), and also evaluates which of these itemsets are frequent. This process is repeated until no new frequent itemsets are found. In this approach still a lot of candidate itemsets might be generated, and the complete database might be scanned repeatedly.
Partition [56]	1995	The Partition algorithm splits the dataset into non-overlapping partitions, each small enough to be accommodated in main memory. In the first stage frequent itemsets are obtained from each partition one at a time. In the second stage new candidate itemsets are generated from the previously obtained itemsets. Due to the split the partitions are read only once. This approach works well for large datasets and reduces CPU and I/O overheads.
FDM [16]	1996	FDM stands for Fast Distributed Mining of association rules. This approach mines frequent itemsets in a distributed environment. In generates candidate itemsets similarly to Apriori, but it makes advantage of relationships between locally large and globally large sets to generate less candidate itemsets. Local refers to a partition of the dataset linked to a site in a distribution system, while global refers to the complete dataset. Then it applies local and global pruning.
DIC [9]	1997	Dynamic Itemset Counting applies fewer passes over the data compared to the Apriori algorithm and checks relatively few candidate itemsets. Instead of confirming candidate itemsets of size k for each k^{th} pass (as done in Apriori), this algorithm makes use of intervals which consist of M transactions. During each pass M transactions are read, and the support of the itemsets in that subset of transactions is obtained. Itemsets which are already confirmed to be frequent are used to create superitemsets. The frequency of these superitemsets can already start to be counted during the current pass.
FP-Growth [29]	2000	The FP-Growth algorithm introduces a novel frequent pattern tree structure. It applies a divide-and-conquer approach. The dataset is converted into a compressed representation: an FP-tree. The FP-tree is divided into conditional FP-trees, and each one is mined separately.
ECLAT [73]	2000	The ECLAT algorithm (Equivalent CLAss Transformation) has three key features. It makes use of a database where an itemset is linked to a list of transactions is occurs in (tid-list database). Itemsets are enumerated via simple tid-list intersections. The second feature is decomposing the search space into smaller, main memory manageable partitions using prefix-based partition. A bottom-up search strategy is applied to enumerate all frequent itemsets with each partition. The efficiency of this approach falls down when very large lists of transactions are intersected.
dECLAT [72]	2003	dECLAT (Diffset ECLAT) is an improvement of ECLAT where the vertical data representation called Diffset is introduced. It stores changes in the transaction IDs of candidates patterns. dECLAT outperforms the running time of ECLAT and FP-Growth on dense datasets, where items occur in most transactions.

Table 4.3: Overview of exhaustive search algorithms to obtain frequent itemsets [40].

FP-Growth is the selected algorithm to obtain frequent itemsets in the Frequent Itemset Model. Research shows that the algorithms FP-Growth, ECLAT, and dECLAT the other algorithms have the best performance [40]. Additionally, the Python library mlxtend (machine learning extensions) contains an implementation of the FP-Growth algorithm. Hence, this library is used to obtain frequent itemsets through the FP-Growth algorithm. The high-level workings of the FP-Growth algorithm is discussed in Table 4.3. It creates an FP-tree, which it divides and mines separately. In more detail, the algorithm works as follows:

1. The dataset is scanned, and the occurrences of all items are counted.
2. The items are sorted in descending order of the number of occurrences.
3. Items with a frequency below the support value are removed.
4. The FP-tree is constructed as follows. The root of the tree is set to null. The dataset is scanned again. For the first transaction, the tree is extended with a chain of nodes each representing an item in the first transaction (in descending order of occurrences). Each node receives a count of 1. For the next transaction, it is checked for the first item in the transaction (the item with the highest occurrence) if that item is already present as a child node of the root of the tree. If it is not, a new chain is created

similarly to the first transaction. If it is present, it is checked for the second item in the transaction if that item is already present as a child node of the first item. If it not, a new chain is created of the remaining items in the transaction. If it is, the third item is checked, etc. The count of the nodes are incremented by 1 each time the node/item is "shared" by a transaction, or set to 1 when a new node is created. This process is repeated until all transactions have been added to the FP-tree.

5. Conditional pattern bases are constructed as follows. Nodes/items in the FP-tree are considered in order of increasing frequency. For each item (multiple nodes can be linked to the same item) the conditional pattern bases are itemsets which are obtained by traversing from the node to the root of the FP-tree. The count of the paths equals the frequency of the node.

6. A conditional FP-tree is created from the conditional pattern bases of an item. The conditional pattern bases of the same item are added together. That means that the itemsets are merged by adding the counts of each item. If the count of an item is below the support value, it is removed.

7. In the last step, the frequent itemsets are generated from the conditional FP-trees. This is achieved by joining the paths in the conditional FP-trees with the node that tree is linked to.

4.4. Association Rule Mining

After having applied frequent itemset mining as described in Section 4.3.3 to obtain sets of articles which are commonly purchased together, the second step is to filter sets based on how likely they represent core ingredients. This task is completed using association rule mining, which is very much related to frequent itemset mining. Association rule mining discovers correlation among items in frequent itemsets through generating association rules. This section gives some background into association rule mining. Then it is explained how frequent itemsets are filtered.

4.4.1. Definitions

This section formally describes association rules and various ways of measuring the importance of rules.

Association rule. An association rule is an implication of the form (A $\Rightarrow$ B), where A $\subset$ I, B $\subset$ I, A $\neq \emptyset$, B $\neq \emptyset$, and A $\cup$ B = $\emptyset$.

Confidence. The confidence of an association rule $A \Rightarrow B$ can be obtained by dividing the number of transactions containing itemsets A and B by the number of transactions containing itemset A. Intuitively, the confidence of a rule represents how likely it is that A and B are present in a transaction given that A is present.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \Rightarrow B)}{support(A)}, \text{ range: } [0,1]$$

Lift. The lift of an association rule $A \Rightarrow B$ can be derived by first obtaining the number of transactions containing A and B divided by the number of transactions containing A and then dividing this value by the fraction of transactions containing B. Intuitively, the lift metric is the ratio between the probability of B given that A is present and the expected probability of B (without having any information on A). The larger the lift value, the more associated the items are. A lift < 1 indicates that the items are not associated.

$$lift(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)} = \frac{confidence(A \Rightarrow B)}{support(B)}, \text{ range: } [0,\infty]$$

Conviction. The conviction of an association rule aims at indicating if a rule happened by chance or not. It represents the ratio between the expected frequency that A is present in a transaction without B if A and B were independent, and the observed frequency of transactions where A is present and B is not. As a directed measure, conviction is sensitive to rule direction and $conviction(A \Rightarrow B) \neq conviction(B \Rightarrow A)$. Intuitively, it represents the ratio that the association rule would be wrong if the items were independent. For instance, a conviction of 1.7 shows that the association rule would be incorrect 70% more often if the items were independent. Similarly to lift, a conviction of 1 indicates that the items are not association. The larger the conviction, the more associated the items are.

$$conviction(A \Rightarrow B) = \frac{P(A)P(\overline{B})}{P(A \cap \overline{B})} = \frac{1 - support(B)}{1 - confidence(A \Rightarrow B)}, \text{ range: } [0,\infty]$$

Leverage. The leverage of an association rule $A \Rightarrow B$ is obtained by subtracting the number of transactions containing A and B by the multiplication of the number of transactions containing A with the number of transactions containing B. Intuitively, this measure is very much like the lift measure, but captures the difference instead of the ratio between the between the probability of B given that A is present and the expected probability of B (without having any information on A). A and B are negatively correlated if $leverage < 0$, positively correlated if $leverage > 0$, and independent if $leverage = 0$.

$$leverage(A \Rightarrow B) = P(A \cup B) - P(A)P(B) = support(A \Rightarrow B) - support(A) \times support(B), \text{ range: } [-1, 1]$$

4.4.2. Filtering Frequent Itemsets

The FP-growth algorithms returns all frequent itemsets for a given value of support. As can be observed in Figure 4.1 the resulting frequent itemsets do not seem to be commonly combined as ingredients. The frequent itemsets have been filtered on how popular they are, not yet on how related the items are. Filtering the itemsets on based on how related the items are can be done through association rule mining. For each itemset, all possible combinations of sets in the antecedent and consequent can be generated. The itemset {wraps, avocado, banana} is created as an example and shows which association rules are generated from it. An itemset is filtered out if one or multiple of its association rules are below the threshold of a certain measure. Similarly, an itemset is kept if all of its association rules are equal or above this threshold. This ensures that all items within the itemset are highly linked to one another.

The measure lift is selected to filter the frequent itemsets on. Firstly, this metric captures the association between items. The other measures are less suitable for the following reasons. The confidence of an association rule is always high when the consequent if frequent. For instance, the rule {sushi rice} ⇒ {milk} would have a relatively large confidence even though the items are unrelated. The conviction measure, on the other hand, does take the support of the consequent into account. It is generally higher if the support of the consequent is low and the confidence of the consequent given the antecedent is high. However, conviction is sensitive to the direction of the rule and measures the effect of the consequent being false. As observed in the data, the conviction favors association rules where the consequent is a small itemset. The conviction of different association rules of the same itemset can vary quite a bit. It can be quite low if the consequent consists of multiple items, despite all items being highly linked. All association rules of an itemset are required to meet a specific threshold. As a consequence, many relevant itemsets would be unnecessarily removed with conviction as the chosen measure. The lift measure considers a ratio instead of an absolute difference. The leverage measure is less suitable, since it captures the difference instead of the ratio. Lift tends to find strongly related, less frequent items, while leverage tends to prioritize more frequent items. Obtaining strongly related items is the main goal in this context, hence the lift measure is selected.

{wraps, avocado, banana}	**threshold: lift = 5**
{wraps} ⇒ {avocado, banana}	lift = 8
{avocado} ⇒ {wraps, banana}	lift = 6
{banana} ⇒ {wraps, avocado}	lift = 1
{wraps, avocado} ⇒ {banana}	lift = 1
{wraps, banana} ⇒ {avocado}	lift = 6
{avocado, banana} ⇒ {wraps}	lift = 8

To speed up this filtering process, the lift value of an association rules is not calculated whenever this is unnecessary. If one association rule of the itemset does not meet the threshold, the itemset can be filtered out immediately and any remaining association rules do not have to be considered. It holds that $lift(A, B) = lift(B, A)$ since $lift(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B \cup A)}{P(B)P(A)} = lift(B \Rightarrow A)$. Hence, only $lift(A \Rightarrow B)$ or $lift(B \Rightarrow A)$ needs to be calculated, not both.

4.5. Clustering

After filtering the frequent itemsets for a given threshold, only itemsets in which the items are associated remain. Some of these itemsets are very similar. The goal of the clustering step is to merge related itemsets into one. For example, {*bun, hamburger*}, {*bun, hamburger, cheddar*}, {*cheddar, lettuce, tomato*}, and {*hamburger, fries, tomato*} should be clustered into {*bun, hamburger, cheddar, lettuce, tomato*}. This section

provides a general overview of clustering methods and zooms into hierarchical clustering. It also dives into methods to select an optimal number of clusters automatically.

4.5.1. Clustering Categories

There exist many approaches to cluster data points. Clustering algorithms can be grouped into categories. Some of these categories are: 1) partitioning-based, 2) hierarchical-based, 3) density-based, 4) grid-based, and 5) model-based clustering categories [20]. Note that this is not an exhaustive list [69]. This section briefly describes these clustering categories, and explains why hierarchical-based clustering is selected.

In partitioning-based or centroid-based clustering, the number of clusters are determined beforehand and data points are assigned to clusters based on their distance to the centroid of a cluster. Advantages of this cluster type are that they are robust, scalable, and simple. A disadvantages is the difficulty with which to predict the optimal number of clusters. A well-known example of a partitioning-based clustering method is k-means clustering. In hierarchical-based clustering, the data points are organized in a hierarchical way based on the distance between them. The result is a dendrogram, which is a tree that gives an overview of all possible clusters. By cutting the dendrogram for a given number of clusters, this approach can output fixed clusters. There are different metrics and linkage criteria to determine distances between data points and clusters. Advantages are that no input parameters are used, and that the initial phase does not require the number of clusters to be specified. A disadvantage is that hierarchical-based clustering is sensitive to outliers.

In density-based clustering, areas of high densities are clustered together. As a result, arbitrary-shaped distributions are formed. Advantages are that it is not required to specify the number of clusters beforehand and that it is resistent to outliers. Disadvantages are that the algorithm has difficulties with high dimensions as well as varying densities. A well-known density-based clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [19]. In grid-based clustering the data space is changed into a grid structure with a fixed number of clusters, from which clusters are formed. Examples of a grid-based clustering algorithms are STING (Statistical Information Grid) [66], WaveCluster [59], and CLIQUE (Clustering In QUEst) [3]. An advantage is the relatively low computational complexity. A disadvantage is the introduction of boundary points which are a challenge to handle. In model-based clustering a predefined model is fitted to the data. The assumption is made that the data is generated by basic probability distributions. Two approaches of model-based clustering are based on statistical learning and neural networks. Advantages are that the obtained clusters can be interpreted from a statistical perspective, and that prior knowledge can be applied. Disadvantages are that the algorithms do require an underlying model and the performance is very dependent on this choice. A well-known statistical learning approach is Expectation–Maximization (EM) algorithm [47], and a well-known neural network approach is the Self-Organizing Map (SOM) [34].

The dataset which needs to be clustered (the filtered, frequent itemsets) consists of taxonomical data. There is a natural hierarchical structure in the product taxonomy and thus the filtered frequent itemsets. Hence, the hierarchical-based clustering approach is selected to cluster the filtered, frequent itemsets with.

4.5.2. Hierarchical Clustering

This section dives deeper into hierarchical clustering, and explains how it is applied in the Frequent Itemset Model. Hierarchical clustering builds a hierarchy of clusters using one of two approaches: agglomerative and divisive. Agglomerative hierarchical clustering applies a “bottom-up” approach which starts with clusters of individual data points and merges clusters until all data points are contained in one cluster. Divisive hierarchical clustering applies a “top-down” approach which starts with one cluster of all data points and splits clusters until each data point is contained in its separate cluster. The Frequent Itemset Model makes use of agglomerative hierarchical clustering.

The itemsets are one-hot encoded based on the article representations (after text cleaning) and transformed into n-dimensional vectors, where n is the number of unique article representations. Each itemset represents a cluster. Clusters are merged based on how (dis)similar they are compared to other clusters. This is measured by the linkage method and distance metric. The linkage method defines how distances are measured between clusters, and the distance metric defined how a distance is measured between vectors. There exist various distance metrics, such as the Manhattan distance ($\sum_{i=1}^{n} |x_i, y_i|$) and the Chebyshev distance ($\max(|x_i, y_i|)$). The Euclidean distance ($\sqrt{\sum_{i=1}^{n} (x_i, y_i)^2}$) is used in this implementation. Table 4.4 shows the formulas and descriptions of multiple linkage methods. One of these five can be selected when running the Frequent Itemset Model.

The output of this process is a dendrogram, which is a tree-like diagram that shows how the data points are clustered hierarchically. This dendrogram is computed once, and is cut in order to output the final clusters. When cutting the dendrogram, the number of clusters needs to be predefined. Since the optimal number of clusters depends on the data, the next section explains how this number is estimated.

Linkage method	**Formula,** $D(C_I, C_J) =$	**Description**
Single linkage	$\min_{i \in C_I, j \in C_J} d(i,j)$	The distance between two clusters is measured as the smallest distance between any of the data points of both clusters.
Complete linkage	$\max_{i \in C_I, j \in C_J} d(i,j)$	The distance between two clusters is measured as the largest distance between any of the data points of both clusters.
Average linkage	$\sum_{i \in C_I, j \in C_J} \frac{d(i,j)}{\lvert C_I \rvert \lvert C_J \rvert}$	The distance between two clusters is measured as the average distance between all of the data points between both clusters.
Centroid method	$d(\mu_{C_I}, \mu_{C_J})$	The distance between two clusters is measures as the distance between the centroids of the clusters.
Ward's method	$\sum_{i \in C_I \cup C_J} d(i, \mu_{C_I \cup C_J})^2 - (\sum_{i \in C_I} d(i, \mu_{C_I})^2 + \sum_{i \in C_J} d(i, \mu_{C_J})^2)$	The distance between two clusters is measured as the increase in the error sum of squares (ESS) after merging two clusters into one.

Table 4.4: Overview of linkage methods. $D(C_I, C_J)$ is the distance between clusters C_I and C_J. μ_{C_I} is the centroid of cluster C_I.

4.5.3. Estimate the Optimal Number of Clusters

The choice of the number of clusters k highly impacts the clustering results and the optimal value of k varies as the data varies. This section describes how finding the optimal number of clusters k can be automated in the Frequent Itemset Model. Several methods of evaluating the clustering result given a value of k are stated before describing the implementation of choice.

Elbow Method. A heuristic where the output of a cost function (e.g. Within-Cluster-Sum of Squared Errors) is plotted again the number of clusters k and which can resemble an elbow if the output of the cost function decreases relatively quickly for a value k compared to $k-1$ and decreases relatively slowly for larger values of k. This is seen as an indication that k clusters fit the data well.

The Silhouette Method. The silhouette coefficient takes into account the cohesion of items within a cluster and the separation between clusters. The silhouette coefficient is in the range between -1 and 1, a higher coefficient implies a better clustering. Assume that a dataset has been clustered into k clusters. Given a cluster C_I, i is a data point in the cluster. a is the mean distance between point i and all other data points in the cluster.

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i,j) \qquad \text{Average distance from point } i \in C_I \text{ to all other points in the cluster.}$$

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i,j) \qquad \text{Smallest average distance of } i \in C_I \text{ to all points in the other clusters.}$$

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, S(i) = 0 \text{ if } |C| = 1$$

Calinski-Harabasz Index. Calinski-Harabasz (CH) Index, also known as variance ratio criterion, and takes into account the ratio between the within-cluster dispersion (W) and the between-cluster dispersion (B). It is assumed that there is a dataset of n instances which is clustered into k clusters. ce is the center of the dataset.

ce_I is the center of cluster C_I.

$$B = \sum_{i \in C_I} |C_I| \times ||ce_I - ce||^2 \qquad \text{Between-cluster dispersion.}$$

$$W = \sum_{I=1}^{k} \sum_{i \in C_I} ||i - ce_I||^2 \qquad \text{Within-cluster dispersion.}$$

$$CH = \frac{B}{W} \times \frac{n-k}{k-1} \qquad \text{The Calinski-Harabasz Index.}$$

Davies-Bouldin index. The Davies-Bouldin index (DBI) is calculated as shown in the formula, with k as the number of clusters, c_i as the centroid of cluster i, S_i as the average distance of all points in cluster i to c_i. If all combinations of clusters have large distances between them ($d(c_i, c_j)$) and are dense ($S_i + S_j$), the Davies-Bouldin index is lower. The lower the index, the better the clustering.

$$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left(\frac{S_i + S_j}{d(c_i, c_j)}\right)$$

Gap Statistic. The gap statistic method compares the within-cluster dispersion with the expectation under an appropriate reference null distribution of data using the output of a clustering algorithm [63]. Suppose the data consists of n independent observations of p features. The distance between i and i' is denoted by $d(i, i')$. The data is clustered into k clusters $C_1, ..., C_k$ with $n_r = |C_r|$. The estimated optimal number of clusters $\hat{k}$ is the value of k where $\log(W_k)$ is the smallest compared to the expectation of $\log(W_k)$. The gap statistic is implemented by estimating $E_n^*\{\log(W_k)\}$ by an average of B copies of $\log(W_k^*)$ computed from a Monte Carlo sample drawn from the reference distribution. Details can be found in the original paper [63].

$$D_r = \sum_{i, i' \in C_r} d(i, i') \qquad \text{Sum of pairwise distance for all points in cluster } r.$$

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \qquad \text{Pooled within-cluster sum of squares around the cluster means.}$$

$$\text{Gap}_n(k) = E_n^*\{\log(W_k)\} - \log(W_k) \qquad \text{The Gap statistic given } k \text{ clusters.}$$

The Elbow method does not always result in a clear "elbow", and it can be unclear which k fits the data the best. This approach is left out, however, the four other approaches are implemented and one can be selected before running the model. The dendrogram is created once. For multiple values of k the dendrogram is cut and the silhouette coefficient, CH index, DB index, or Gap statistic is calculated. The number of clusters k with the highest clustering score is applied. In the current implementation all values of k between 6 (less than 6 are not clustered) and the number of itemsets are compared. However, one could also skip checking some values of k to speed up this process.

4.6. Matching Itemsets and Dish Types

Frequent itemsets have been found, filtered, and clustered. The final phase of the Frequent Itemset Model is to assign dish types to these itemsets. For instance, the itemset {sushi rice, avocado, salmon, seaweed} should be assigned to dish type "sushi". This is achieved by making use of the recipe dataset described in Section 3.1.4. This dataset can also be viewed at as a collection of examples of dishes. The intuitive idea is to assign dish types to (some of the) itemsets through by matching the articles in the itemsets to the ingredients in the recipes/sample dishes. There are three steps in this phase. The first step is to tag recipes with dish types. The recipes can be viewed as sample dishes. However, the model has no knowledge of a recipe belonging to a specific dish type. The second step is to match itemsets to recipes, through obtaining a similarity score between the articles in the itemsets and the ingredients in the recipes. After these two steps, the itemsets are linked to recipes, and the recipes are tagged with dish types. Based on this information, the most relevant dish type is voted on in the third step.

4.6.1. Tagging Recipes with Dish Types

The first step is to tag recipes with dish types, so that it is known of all recipes/sample dishes to which dish type they belong. Although some recipes from the scraped dataset contain information about a cuisine or

dish type, these categories are not identical to the defined class labels in this context. The dish type of a recipe is new information. This process is automated to avoid the task of manually assigning a dish type to each of the 12,144 recipes and to enable easy addition of more recipes. This section describes the data and the (new) classes used to tag recipes with. Since no suitable labels are available, weak labelling is applied. This section also describes how this is achieved with the use of labelling functions in Snorkel [50]. Finally, three approaches using the Majority Label Voter, Label Model, and Supervised Learning Model, are implemented. The simplest approach, using the Majority Label Voter, performed the best. Hence, this approach is selected in order to tag recipes with dish types.

Dataset. As described in Section 3.1.4 the recipe dataset consists of 12,144 instances. Each recipe consists of a name and an ingredient list. One challenge is the lack of labelled data. None of the recipes have a dish type, as described in Section 3.2, assigned to them. All recipe names and ingredients are filtered as described in Section 4.2.3.

Classes. The classes that the recipes can be tagged with consist of the original classes introduced in Section 3.2 and some additional classes. The additional classes are: “abstain”, “drinks”, “fish”, “meat”, “pancakes”, “poultry”, “sandwich”, “sauce”, “sweets”, and “veggies”. The class “abstain” is required by Snorkel and indicates that no label is assigned. The other additional classes have been added since they represent a selection of the recipes well. For example, there are recipes where only a piece of fish/poultry/meat, or some vegetables are prepared. These do not fall under any of the existing recipes. The idea is that introducing these additional classes decreases the chance that these recipes are incorrectly added to the original classes. The additional classes are used for the weak labelling process, but recipes which are tagged with such a class are discarded.

Labelling Functions (LFs). To avoid labelling all recipes manually, programmatic labelling is applied to obtain a large, lower-quality labelled dataset more quickly. Programmatic labelling is implemented with the use of labelling functions with Snorkel. The Snorkel project started at the Stanford AI Lab as research project in 2015 with the aim of facilitating programmatically label, build, and manage training data. Later on the company Snorkel AI was established and the focus shifted to building Snorkel Flow, a data-centric AI development platform. This platform was made generally available in March 2022. Due to the timing and pricing, the original Snorkel system is used instead of the Snorkel Flow platform. The Snorkel system is based around labelling functions, which are functions that output a label for a subset of the dataset based on domain knowledge inserted into the function. An example of a labelling function is “if one of the ingredients of a recipe is fusilli, penne, or ravioli and none of the ingredients is gnocchi, then return label pasta, else return label abstain”. In the actual implementation these labelling functions are larger. For instance, 62 pasta types are checked instead of 3 as illustrated in the example. For each dish type, multiple labelling functions are created, with a total of 107. These labelling functions have been written by the author. After the labelling functions have been applied to the recipe dataset, each recipe instance has received 107 “votes”. An advantage of this approach is that it absorbs domain knowledge. An disadvantage is that it requires a significant time investment before it can label instances automatically.

Majority Label Voter. The first model that is applied to decide on a final dish type per recipe instance is the Majority Label Voter. It simply chooses the most voted label as the prediction for each instance. One drawback is that labels for which there exist more labelling functions are favoured.

Label Model. Accuracies and correlations between the labelling functions are unknown, and the outputs can overlap as well as conflict. The Label Model aims at solving this by approximating the accuracies of the labelling functions, re-weighting and combining the resulting labels. This is achieved by learning a model of the conditional probabilities of the the true label Y: $P(lf|Y)$. “The approach is based on "Training Complex Models with Multi-Task Weak Supervision” [51].

Supervised Learning Model. Although considered as the third model, this approach is actually the main ambition using Snorkel. In this approach, the dataset is randomly split into a training and test set, with an 80/20-ratio respectively. The Majority Label Voter or the Label Model is applied to the training data to obtain weak labels. The instances in the test set are manually labelled to obtain the ground truth for proper evaluation. Then a machine learning model can be trained to predict dish types for recipes.

4.6.2. Linking Itemsets and Recipes.

The second step is to match the frequent, filtered, and clustered itemsets to recipes. This is achieved by matching the articles in the itemsets to the ingredients in the recipes. Information about an article such as the article name, the brand, the size, and the volume/weight is known. The only available information about an ingredient is a descriptive name. In this context two datasets are joined as it is known that some items belong to the same entity, but a unique identifiers is missing. Such a problem is often referred to as record linkage or data linkage. It is a common, challenging problem. In this case it is extra challenging for multiple reasons: 1) only descriptive names are available is both datasets, 2) descriptive names contain noise (e.g. "2 onions, sliced"), 3) there is not a clear definition of the entities (e.g. do mashed potatoes and new potatoes refer to the same entity?), and 4) similar items can have different descriptions (e.g. "baguette", "pain", and "stokbrood" refer to the same entity. It is clear that matching articles and ingredients with the aim of matching itemsets and recipes is not trivial. Nevertheless the chosen solution is a simple one. It consists of three steps: 1) calculating similarity scores between articles and ingredients, 2) linking articles and ingredients if the similarity score is above a certain threshold ("minimum matching score"), and 3) linking itemsets and recipes if a certain number of articles/ingredients are linked ("minimum matching ingredients"). Both the "minimum matching score" ($0 \leq x \leq 1$) and "minimum matching ingredients" ($x > 0$) are values that can be set before running the Frequent Itemset Model.

The objective of the first step is to create an overview of all article and ingredient combinations and with similarity scores. All articles in Picnic's store are obtained and pre-processed according to Section 4.2. Unrelated articles are removed, the representation is selected according to the product taxonomy, and the text is cleaned. All ingredients from the recipe dataset, only undergo the text cleaning. For each combination of the pre-processed articles and ingredients a similarity score is calculated using the token_set_ratio() function from the FuzzyWuzzy library. After obtaining similarity scores for all article-ingredient combinations, a threshold "minimum matching score" determines which articles and ingredients are considered to refer to the same entity. The third step makes use of this information to link itemsets to recipes. If at least a certain number of items (set as the "minimum matching ingredients") between the articles in an itemset and the ingredients in a recipe are considered the same (i.e. meet the "minimum matching score"), then the itemset and recipe are linked. The result is that each itemset is matched to zero, one, or multiple recipes.

4.6.3. Dish Type Voting.

The previous two steps ensure that itemsets are matched to recipes, and that recipes are tagged with dish types. Hence, each itemset is matched to zero, one, or multiple dish types. In the final step this information is used to match each itemset to one dish type (or none at all). Firstly, itemsets that are not matched to any recipes are removed. Itemsets that are matched a single recipe, are matched to the dish type of that recipe. Itemsets that are matched to multiple recipes but each of a different dish type, are removed. The rest of the itemsets are matched to at least two recipes of the same dish type. For these itemsets a majority vote is applied. In case of a tie, one of the dish types is selected at random. Table 4.5 shares some hypothetical examples.

Itemset	**Dish type of the matched recipe**	**Selected dish type**
{almonds, cashew nuts, walnuts}	-	-
{pear, fig, walnut}	salad	salad
{shrimps, avocado, carrot, cucumber, soy sauce}	spring roll sushi	-
{stock cubes, parsley, risotto}	rice risotto soup	-
{wraps, corn, diced tomatoes, iceberg lettuce}	wraps wraps wraps flatbread flatbread salad	wraps
{basil, tomato, parmesan}	pasta pasta pizza pizza gnocchi	pasta or pizza

Table 4.5: Example dish type voting.

4.7. Summary.

While Section 4 describes the Frequent Itemset Model in detail, this section provides a concise description. The input of the model is a (large) dataset of Picnic deliveries. For each delivery, irrelevant articles are removed, article representations are selected based on Picnic's product taxonomy, and these text representations are cleaned.

Frequent itemsets are mined using the FP-Growth algorithm. The minimum support value can be set and defines what is considered frequent. For each frequent itemset it is checked whether all of its association rules meet a threshold for the lift metric. This threshold, the minimum lift, can be set before running the model and determines how "related" the items in the itemsets are. The itemsets are transformed into vectors and agglomerative hierarchical clustering is applied to cluster similar itemsets. The distance metric is Euclidean distance and any of the available linkage methods (single linkage, complete linkage, average linkage, centroid method, or Ward's method) can be selected. The optimal number of clusters k is be determined by comparing the scores for the obtained clusters for different values of k. The scores are calculated by selecting one of the following methods: the Silhouette method, Calinski-Harabas index, Davies-Bouldin index, or Gap Statistic.

In the next step, the obtained itemsets are linked to dish types. Firstly, 12,144 recipes are tagged with the defined dish types. This is achieved using programmatic labelling through Snorkel. A total of 107 labelling function are defined to "vote" dish types for each recipe. The dish type with the most votes is assigned to the recipe. Secondly, the itemsets are matched to recipes (and their dish types). Each Picnic article and ingredient from the recipe dataset is given a similarity score based on fuzzy string matching. One can select the minimum score that is required in order to assume that the article and ingredient are similar. To match an itemsets and a recipe, a certain number of articles/ingredients much be present in both sets. The minimum similarity score and the minimum number matching items can be defined before running the model. Each itemset is matched to zero, one, or multiple dish types. Thirdly, the final dish types for each itemset are selected based on majority voting. The model has linked dish types to itemsets, and itemsets to deliveries. As a result, deliveries are now linked to dish types.

5

Supervised Learning Model

This chapter presents the second approach for dish type detection in which supervised learning techniques are applied. An overview of the model is shown in Section 5.1. Section 5.2 discusses the feature engineering process and Section 5.3 dives into the supervised learning algorithms as well as problem transformation and algorithm adaptation methods which support multi-label classification problems. Finally, Section 5.4 shares a summary of the Supervised Learning Model.

5.1. Overview

Instead of discovering patterns, this approach is based on learning from examples in order to detect dish types in Picnic deliveries. An overview is provided in Figure 5.1. The specific dish types are defined in Section 3.2. The available labeled dataset does not represent a usual situation. For example, customers cook the exact same 119 recipes in the available labeled dataset. This is important to keep in mind when exploring the data.

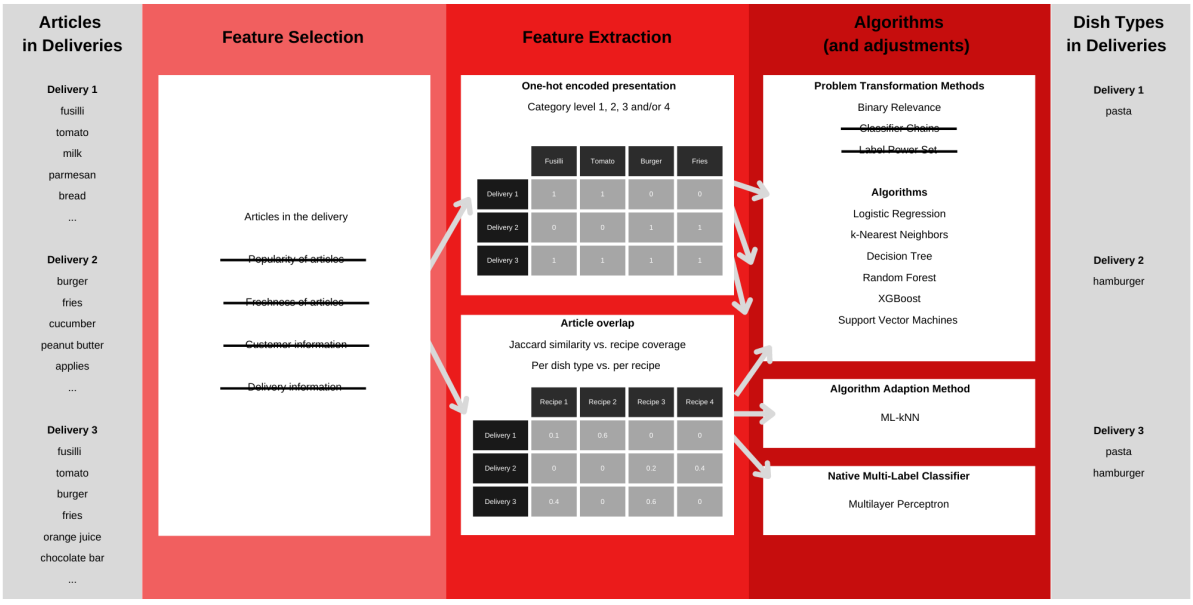

Figure 5.1: Overview of the steps and decision of the Supervised Learning Model.

Another challenge is that the data is noisy. A delivery might not contain the full list of ingredients needed to create a dish. A customer could have obtained some ingredients elsewhere or have some of the ingredients at home already. Secondly, a customer buys articles unrelated to a dish. For some articles it is obvious that they are never ingredients for a dish, like cleaning articles. However, cherry tomatoes could very likely be used to cook a dish as well as for an afternoon snack.

5.2. Feature Engineering

The process of selecting and extracting features that can distinguish dish types in deliveries is described in this section. In Section 5.2.1 several features to detect dish types in deliveries are proposed. To better understand the data and potentially discover patterns, an exploratory data analysis is performed for the identified features. The resulting insights are employed to select features. Finally, the representation of the selected features is discussed in Section 5.2.2.

5.2.1. Feature Selection

Several features which might be useful to detect dish types in deliveries are identified. The most straightforward features are the articles which are purchased in the deliveries. The attributes of the articles are also inspected. In particular, the popularity and the freshness of the articles. In addition, information about the customers might be indicative for customers' dish type preferences. Lastly, delivery information such as the variety of articles may be of relevance. The rest of this section describes these features in more detail and shows the results of examining the data for all identified features. Only the articles in the deliveries are selected as features.

The figures can be found in the confidential version of the thesis.

Articles in the delivery. An apparent feature is the list of articles which are contained in a delivery. Individual articles fall under multiple categories, as described and visualized in Section 3.1.1. Consequently, these the individual articles as well as four category levels can represent five different types of features. For the purpose of generalization, the focus is put on category levels instead of individual articles.

The first figure orders the categories based on how often they appear in the deliveries in the available labeled dataset for all four category levels. The intensity of the red color represents the popularity of the category: the redder the categories, the less often they appear in deliveries. The figure also provides an idea of how granular the categories are and lists the names of (a selection of) categories along the y-axis. This information is useful when observing The second figure in which the ten most popular categories of only deliveries in which a recipe of a certain dish type was purchased are shown per category level. The colors of categories are the same as in the second figure, so they display the popularity of categories before grouping deliveries on the dish types they contain.

The second figure is looked at from two angles: per category level and per dish type. In general, the more granular the category level, the redder the colors and thus the less frequently purchased the top ten categories are in general. In the case of category level 1, the general top ten is similar to the top ten for each dish type. It is concluded that category level 1 contains too little detailed insights. However, the other category levels could be useful to detect dish types.

Observing the second figure per dish type shows multiple interesting insights. The four main takeaways are: 1) popular categories are popular throughout all dish types, 2) customers purchase similar and typical categories for each dish type, 3) there is a larger variety of purchased categories for some dish types, and 4) different category levels complement one another. The second and third points are very much related to this particular dataset. Examples are shared in the confidential version of the thesis.

The first insight that popular categories are popular throughout all dish types is very much observable in the figure. The most frequently purchased categories (in general) often appear in the top ten categories per dish type. The second perception is that the ten most purchased categories, with the exception of the popular categories, seem to be in line with one would expect for each dish type. This is especially the case for category level 4. Thirdly, deliveries of certain dish types contain very much the same unique categories (wide, red bars) while other dish types consist of a broader variety of more popular categories (narrow, white bars). This is visible in the figure through the length and redness of the bars. It is noticeable that dish types which contain wide, red bars are often also the dish types which are linked to fewer recipes. Dish types with less distinguishable categories are mostly dish types to which multiple recipes are linked. For these dish types it is expected that the model will have more difficulty to detect them in unlabeled deliveries. Finally, it is observed that different category levels pick up different views of a dish type. For instance, note that in the case of dish type *pasta*, no level 3 or 4 category related to pasta shows up. The reason for this is that the level 3 and 4 categories are related to multiple pasta types, such as "fusilli", "spaghetti", and "tagliatelle". However, all fall under level 2 category "pasta", and this category does appear in many of the deliveries with dish type *pasta*. So for this dish type using category level 2 as a feature might be more useful than category level 3 or 4. Additionally, dish type *sushi* illustrates that different category levels pick up different unique categories. Category level 2 identifies category "japans", while category level 4 identifies category "zeegroenten vers".

Popularity of the articles in the delivery. Two types of articles can be identified. Firstly, the standard articles, such as yoghurt, bread, milk, eggs, and fruit. It is quietly likely that many of these articles are purchased regularly. The second type of articles are used as ingredients for (new) recipes. It is expected that these articles are not purchased less often. To distinguish between these types of articles, the frequency with which a customer purchases articles can be taken into account. Hence, another potential feature is information about the popularity of articles in the delivery. Article popularity is defined as the "article penetration", which is the number of deliveries which contain the articles divided by the total deliveries. Personal and general article penetration are considered.

The first calculates the article penetration for each individual customer and for each group of articles that are related to either a specific dish type or are not linked to a Picnic recipe at all ("articles not linked to a dish type"). The first figure, shows the empirical cumulative distribution function of the average personal article penetration per dish type (averaged over the articles in a group) and the average of this distribution (averaged over the customers). It is indeed observed that the articles unrelated to a dish type have a higher personal article penetration compared to the articles related to dish types. There is one outlier, which could again be explained by the small proportion of samples.

The second figure shows the general article penetration. Instead of considering how often an individual customer purchases an article, the general article penetration is the number of deliveries which contain the article divided by the total number of deliveries. It is striking that the group of articles which are not linked to a dish type are not purchased relatively most often anymore, as was the case for personal article penetration. However, this is quite likely an effect of the recipe page. Once a recipe is published in the recipe page, many customers purchase the exact same articles. Hence, the general article penetration for these articles becomes relatively high. This is not the case for personal article penetration since it is unlikely that customers order the same recipe multiple times in the short period that the recipe is live.

In conclusion, the general article penetration does not seem to be useful in this context, but the personal article penetration does. For the "true" dataset, this distinction might be smaller if customers regularly cook the same recipes. Although the personal article penetration seems to be a helpful feature it is not taken into account in the model, and could be further explored in future work.

Freshness of the articles in the delivery. Also the freshness of articles is considered as a potential feature. The first figure shows the median number of articles with or without freshness guarantee for all deliveries which contain a recipe of a specific dish type. There seems to be little variation between the dish types. The second figure contains two plots. The first is the empirical cumulative distribution function of the median freshness guarantee of the articles in the deliveries of a certain dish type. The second plot is the average taken over the median freshness guarantee in days of the articles in the deliveries of a certain dish type. The averages are different per dish type, which seems to indicate that the freshness guarantees of articles could be a useful feature. However, it is unclear how much these results are caused by the little variety of recipes in the available dataset. Four outliers also happen to be the dish types which are linked to the least number of recipes.

Customer information. Three types of information on customers is explored: 1) the "household type", 2) the average days between deliveries, and 3) the total number of deliveries. The distribution of the customers' "household types" is also quite similar per dish type as shown in the first figure . The intuition behind the average and total number of deliveries is to distinguish between customers who order regularly and those who do not. Regular customer might be more likely to buy their meals with Picnic. However, in the available dataset there are no instances without labels. The figures show the empirical cumulative distribution function and the average of both the average number of days between deliveries and the total number of deliveries per dish type. In the second figure one might the spot outliers when looking at the averages. However, as Section 3.3.3 describes, these are the four dish types which are purchased the least, which might explain the deviation. The total number of deliveries, as shown in the third figure , is quite similar per dish type.

Delivery information. Lastly, the number of distinct category levels in the delivery is considered as a feature. This potential feature was initially proposed to detect deliveries in which no articles were purchased to cook a main meal. The intuition was that such deliveries might have a smaller variety of articles. However, since the dataset contains no instances without labels, this information is not expected to be useful. The figure which displays the empirical cumulative distribution function of all four category levels confirms this.

5.2.2. Feature Extraction

The type of feature that is the selected are the articles in the deliveries. There are multiple ways in which this information can be represented. This section explains the two representations that are used and referred to as "one-hot encoding" and "article overlap".

One-hot encoding. Articles can be represented by category level 2, 3, and/or 4. These category levels are one-hot encoded, so the presence or absence of each category level is a feature. Depending on which category level representation(s) are chosen, the number of features is quite high. Additionally, the data is sparse.

Article overlap. Another representation which results in less features is named "article overlap". It represents the similarity between (the representation of) articles in a delivery and ingredients in a recipe/dish type. The article overlap can be calculated in four ways, referred to as "the settings", as shown in Table 5.1. For each delivery, a similarity is calculated per Picnic recipe or per dish type. In case of the former there are 119 features, in case of the latter there are 23 features. The first is obtained by calculating the similarity of a delivery with each recipe. The second is obtained by calculating the similarity of a delivery with each recipe, and taking the highest similarity per dish type. The similarity is either calculated as the Jaccard index or recipe coverage. The Jaccard index is defined as $\frac{|A \cap B|}{|A \cup B|}$, where A is the set of (representation of) articles in the delivery and B is the set of ingredients in the recipe. The recipe coverage is the percentage of ingredients of a recipe which are contained in the delivery. Articles can be represented as category level 2, 3, 4, or as individual articles. An example calculation is shown in Table 5.1.

Example Calculation Article Overlap Settings

delivery = {A, B, C, D, E, F}
recipe 1 (dish type 1) = {A, B, C}
recipe 2 (dish type 1) = {B, C, D, Y, Z}
recipe 3 (dish type 2) = {D, E, F, Z}

Article Overlap Settings	Resulting features	Description
Per recipe + Jaccard index	recipe 1 = 3/6 = 0.5 recipe 2 = 3/8 =0.375 recipe 3 = 3/7 = 0.429	Feature per recipe, similarity as the Jaccard index between articles in a delivery and ingredients in a recipe.
Per recipe + Recipe coverage	recipe 1 = 3/3 = 1.0 recipe 2 = 3/5 = 0.6 recipe 3 = 3/4 = 0.75	Feature per recipe, similarity as the percentage of ingredients of a recipe present in a delivery.
Per dish type + Jaccard index	dish type 1 = max(0.5, 0.375) = 0.5 dish type 2 = 0.429	Feature per dish type, similarity as the Jaccard index between articles in a delivery and ingredients in a recipe (max per dish type).
Per dish type + Recipe coverage	dish type 1 = max(1.0, 0.6) = 1.0 dish type 2 = 0.75	Feature per dish type, similarity as the percentage of ingredients of a recipe present in a delivery (max per dish type).

Table 5.1: The four ways to compute the article overlap features shown in an example.

5.3. Modelling

The task of multi-label classification is to assign zero, one, or multiple labels to every instance. Instead of predicting one class out of two classes (binary classification) or multiple classes (multi-class classification), the model needs to predict for each of the classes if it is present in the instance or not. Especially with 23 different classes, that makes the problem more complicated. Two methods are commonly applied to tackle this: problem transformation methods and algorithm adaptation methods. There also exist algorithms which intrinsically support multi-label classification problems. This sections describes the methods and algorithms.

5.3.1. Problem Transformation Methods

Problem transformation methods transform multi-label tasks into binary or multi-class tasks. After such transformation any single-label algorithm can be applied. Binary relevance (BR), label power set (LP), classifier chains, and ensemble of classifier chains (ECC) are some of the methods to transform a multi-label problem to a binary one.

Binary Relevance (BR). The first method, called binary relevance, decomposes one multi-label problem into a binary problem for each class label. There is one binary problem for each class label. Every class has a

dataset which is constructed of instances from the original dataset belonging to that class. The binary problems are considered independently before the results are aggregated into a final prediction [15]. A considerable advantage is that the binary problems can optimize for their own loss functions. Other advantages are that the time complexity is linear with the respect to the number of classes, and that the different models can be parallelized. A common criticism of this method is that it assumes that labels are independent, as it does not take label dependency information into consideration.

Label Power Set (LP). The Label Power Set transforms a multi-label classification problem into one multi-class classification problem where each unique label combination is now considered as a separate class. An advantage of this approach is that it takes into account the correlations between class labels. Disadvantages are that there might be limited training examples for less frequent label combinations and that it has exponential time complexity. With 23 labels in the original problem, the multi-class classification problem can now have up to $2^{23} = 8388608$ labels.

Classifier Chains (CC). Introduced in 2009 the Classifier Chains method provides an alternative to binary relevance and is, on the contrary, able model label correlations while compromising little on computational complexity [52]. Similarly to binary relevance, classifier chains transforms one multi-label classification problem into a single binary classification problem for each class label. Then, contrary to binary relevance, it creates each classifier one by one and incorporates the predictions of the previous classifiers in the feature space of the current classifier. The order of this chain influences performance.

Ensemble of Classifier Chains (ECC). Ensemble of Classifier Chains, introduced in the same paper as Classifier Chains, identifies this issue by training multiple classifiers with a random chain ordering and a random subset of the dataset [52]. The resulting classifier likely give different predictions, which are collected as votes for each label. A threshold over the number of classifiers is applied to obtain the final predictions.

A common characteristic of the problem transformation methods in this section, with the exception of binary relevance, is that they take into account label dependency. As discussed in 3.3.4 current data might indicate that the labels are dependent, but this is actually the result of having selective recipes live in the app. Any dependencies between dish types that might be present are not uncovered. Dependencies that are picked up are not caused by the effect between dish types and can be different for new data that is being collected. Hence, the labels are considered independent and the binary relevance method is selected exclusively. By transforming the problem into multiple single-label problems through binary relevance, many traditional algorithms which are developed for single-label classification problems can be applied to solve the problem. This section provides an overview of these algorithms. The intuition behind these algorithms is briefly discussed, as these are well-known algorithms for which many resources about their workings exist. Relevant settings of parameters are stated per algorithm. This section also shares the general advantages and disadvantages of these algorithm.

Logistic Regression (LR). As an extension of linear regression, logistic regression uses the sigmoid function to output probabilities of a sample belonging to a class (or not): $P(y = 1|\mathbf{x}) = 1/(1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b)))$, with $\mathbf{x}$ as the feature vector, y as the class label, $\mathbf{w}$ as the feature weights, and b as the bias term. The objective function is cross-entropy loss/log loss. Through training, the model learns a vector of weights as well as a bias term/intercept. Note that each weight value is linked to a feature. This value represents the contribution of that feature, and hence insights into indicators of feature importance are easily obtained and logistic regression is often celebrated for its explainability. Another advantage is that it can classify new records fast and it does not make assumptions about the distribution of the class labels. A disadvantage is that it is prone to over-fitting when many features are used compared to observations. This is the case the articles are represented as one-hot encoded features. One may consider regularization techniques to avoid over-fitting in these scenarios. The problem holds complex relationships, which is also a challenge for logistic regression as it constructs linear boundaries.

k-Nearest Neighbours (k-NN). As an instance-based/memory-based learning algorithm, k-nearest neighbours stores the feature vectors and class labels of the training set during training, and compares new instances to it in order to put the instance into context and make a prediction during the classification phase. More specifically, it classifies a new instance by computing a majority vote of the labels of the k nearest neigh-

bours. The value of *k* is set to the default value of 5. Advantages of the algorithm are that it is simple, intuitive and requires no training step. New data can be added to the training set without requiring a long training process. A disadvantages is that the classification phase takes a long time and performance declines for large dataset with more dimensions. It requires a large memory to store the entire training set. Additionally, k-nearest neighbours could have problems with an imbalanced dataset as it favors popular classes. It is also sensitive to outliers and the value of *k* should be chosen carefully when optimizing the model performance.

Decision Tree (DT). As the simplest tree-based model, a decision tree consists of nodes which represent decision rules in the form of if-then statements. A very limited tree for our problem could be as follows. If the delivery contains spaghetti, then predict dish type pasta. Else, if the delivery contains lasagne sheet, then predict dish type *lasagne*. During training the Classification and Regression Tree (CART) algorithm is applied to aim to obtain the optimal tree, which is an NP-complete problem [37]. Gini impurity is used as the splitting criterion to measure the quality of a split. Similarly to logistic regression, an advantage is that the model is interpretable. However, like logistic regression is also has a tendency to over-fit. Other disadvantages are that it is unstable and may be biased towards dominating classes.

Random Forest (RF). Random forest is an ensemble learning method using bootstrap aggregating/bagging. It fits multiple decision trees on subsamples of the dataset. The prediction for an unseen instance is the class that is outputted the most by the individual individual trees. An advantage of random forest is that it can handle large datasets efficiently. Generally, it requires less training time compared to neural networks and is more stable compared to decision trees. Compared to decision trees, it is also more difficult to interpret, however, the scikit-learn implementation facilitates interpretation through the “feature_importances” propery which calculates the Gini importance, which is the (normalized) total reduction of the criterion brought by that feature [58].

Extreme Gradient Boosting (XGBoost). XGBoost is an implementation of the gradient boosted trees algorithm. A gradient boosted tree, similarly to random forest, is an ensemble learning algorithm. The difference is that random forest applies bootstrap aggregating, while gradient boosting trees apply an extended version of bootstrap aggregating where decision trees are generated iteratively. For each tree the aim is to minimize the error compared to the previous one using a gradient descent algorithm. The gradient boosting algorithm is made up of the weighted sum of these trees. XGBoost is an optimized implementation of the gradient boosting method, which uses a different objective function, is parallelizable, and uses L1 and L2 regularization.

Support Vector Machines (SVMs). This algorithm creates a hyperplane in an N-dimensional space, with N as the number of features, which separates classes of instances. The objective is to maximize the margins (between supporting vectors). In other words, the aim is to maximize the minimum distance between the data points of two classes. SVMs apply a linear decision boundary, but not all datasets are linearly separable. One example is a line of alternating data points of a different classes. To solve this the “kernel trick” is applied which maps the training data onto a non-linear decision surface with the help of a kernel function. Hence, these non-linear SVMs can also be applied to non-linearly separable data. Different kernel functions allow this algorithm to be versatile. Another advantage is that SVMs usually work well in high dimensional spaces, which is the case of the one-hot encoded article features. A disadvantage is that SVMs are not suitable for large datasets due to the long training time. This also becomes clear in Section 6.3.2 and results in the algorithm being dropped due to limited resources despite its expectation of good performance.

5.3.2. Algorithm Adaptation Methods

Algorithm adaptation methods are methods which extend already existing algorithms to handle multi-label data directly rather than transforming the problem. This work makes use of one of such methods: ML-kNN.

Multi-Label k-Nearest Neighbours (ML-kNN). ML-kNN is a multi-label lazy learning approach which is derived from k-nearest neighbours and has been adapted for multi-label classification. Multiple k-nearest neighbours algorithms are “trained” before the label set is determined using maximum a posteriori (MAP) principle is applied to the number of neighbouring instances belonging to each possible class. In a bit more detail, this step obtains the prior probabilities of the event that a test instance has a certain label or not, as well as the posterior probabilities that there are exactly a certain number of instances of a specific label given

the event that a test instance has a certain label or not. Both probabilities can be estimated from the training set. The complete explanation and formal notation can be found in the original paper [76]. It is not used here but ML-kNN is able to output a ranking of the labels. This algorithm adaptation method has similar advantages and disadvantages compared to the algorithm its derived from (kNN).

5.3.3. Intrinsic Multi-Label Classifiers

In addition to transforming the problem or existing algorithms, there are exist algorithms which are able to support multi-label classification problems. One well-known algorithm is the multilayer perceptron.

Multilayer Perceptron (MLP). Multilayer perceptron is a fully connected multi-layer feedforward artificial neural network (ANN). The default settings of the scikit-learn implementation have been used: one hidden layer of 100 neurons, Adam optimization algorithm (combining Momentum and RMSprop), learning rate of 0.001, alpha of 0.0001, batch size of 200, and 200 epochs. The output is a vector of raw values, each linked to a class label. The sigmoid or logistic activation function is applied to transform these raw values into probabilities. The default threshold of 0.5 is used to determine which class labels are predicted.

5.4. Summary

This section summarizes the Supervised Learning Model. The articles in deliveries are selected as features to detect dish types. The articles can be represented by category level 1, 2, 3, and/or 4. The articles in deliveries can be one-hot encoded, or can be used to calculate a feature referred to as article overlap. The latter defines how similar a delivery is to a recipe or dish type. A feature is created for each of the 119 recipes or 23 dish type. In case of a feature per dish type, the maximum similarity of all recipes of the same dish type is obtained for that dish type. The similarity can be based on the Jaccard index between the articles in a delivery and the ingredients in a Picnic recipe. It can also be based on the recipe coverage, which is the percentage of the ingredients of the recipe that is present in the delivery. One of multiple algorithms can be selected. The available algorithms are logistic regression, k-nearest neighbours, decision tree, random forest, extreme gradient boosting, support vector machines, multi-label k-nearest neighbours, and multilayer perceptron. With the exception of the last two algorithms, binary relevance is used in order to apply these algorithms to this multi-label classification problem.

6

Experimental Results

This chapter describes the experiments and presents the results of evaluating the models on two types of datasets. First, Section 6.1 provides an overview of evaluation metrics for multi-label classification problems and explains the considerations in this context. The performance of the Frequent Itemset Model and the Supervised Learning Model are analyzed on the weakly labelled dataset obtained from the recipe page in Sections 6.2 and 6.3 respectively. The results of an attempt to estimate the performance of the models on manually labelled deliveries are presented in Section 6.4. Additionally, the Supervised Learning Model is trained on predictions of the Frequent Itemset Model. This approach is referred to as the *Hybrid Model*, and its performance is estimated in Section 6.4. In each section the experiment design is described before the results are presented. For any details on the features and algorithms the reader is referred to Sections 5.2 and 5.3 respectively. Throughout this section multiple abbreviations are used. An overview is shown in Table 6.1.

Metrics	
EMR	Exact match ratio
HL	Hamming loss
Macro-F_1	Macro-averaged F_1 score
Micro-F_1	Micro-averaged F_1-score
Algorithms	
LR	Binary relevance with logistic regression
kNN	Binary relevance with k-nearest neighbours
DT	Binary relevance with decision trees
RF	Binary relevance with random forest
XGB	Binary relevance with XGBoost
SVM	Binary relevance with support vector machines
ML-kNN	Multi-label k-nearest neighbours
MLP	Multilayer perceptron
Models	
FIM	Freqeunt Itemset Model
SLM	Supervised Learning Model
HM	Hybrid Model

Table 6.1: Overview of abbreviations

6.1. Evaluation Metrics

The choice of evaluation metric has great influence on how model performance is perceived. When comparing models, the outcome may be different dependent on the selected evaluation metric. Some conventional metrics for binary and multi-class classification problems are accuracy, error rate, precision, recall/sensitivity, specificity, and F1-score [32]. Each has its advantages and disadvantages, and might be suitable depending on the context. For example, accuracy, (TP+TN)/(P+N), is a simple metric which favours majority class instances, but could be very suitable for e.g. balanced datasets of binary classification problems. Precision, (TP)/(TP+FP), is more appropriate when it is important that a prediction is indeed a correct prediction (measure of quality). Recall, (TP)/(TP+FN), is more appropriate when it is important to capture as many correct predictions as possible (measure of quantity). However, a wrong prediction is not necessarily completely

wrong in the case of multi-label classification. A prediction consists of a set of classes, and it could be that some are incorrect while others are correct. A prediction where only one out of four labels is wrong is better than a prediction where three out of four labels is wrong. There exist two types of multi-label evaluation metric which take this into account: label-based and example-based metrics. The former includes macro-average, micro-average, and weight average versions of conventional metrics. The latter includes example-based versions of conventional metrics, Hamming Loss, and Exact Match Ratio/Subset Accuracy [75]. Descriptions and considerations of evaluation metrics for multi-label classification problems are presented to determine which evaluation metric is selected. Consider the following definitions for the mathematical notations in this section:

$N =$ set of all instances	$TP =$ # instances of the positive class correctly predicted
$C =$ set of all classes	$FP =$ # instances of the positive class incorrectly predicted
$Y =$ set of all targets	$TN =$ # instances of the negative class correctly predicted
$\hat{Y} =$ set of all predictions	$FN =$ # instances of the negative class incorrectly predicted
$I =$ indicator function	

Micro-average. The micro-average method takes the average of the true/false negatives/positives of all classes and then calculates the evaluation metric (e.g. precision) as usual.

$$Precision_{micro} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FP_c}$$

Macro-average. The macro-average method takes the average of the evaluation metric (e.g. precision) over all classes. A characteristic of this method is that it does not take label imbalance into account.

$$Precision_{macro} = \frac{\sum_{c \in C} precision_c}{|C|}$$

Weighted average. The weighted average method is similar to macro average method with the exception that it does take label imbalance into account as individual classes are now weighted by the support of the class in the target set. Any evaluation metric can be chosen, e.g. precision.

$$Precision_{weighted} = \frac{\sum_{c \in C} precision_c * support_c}{\sum_{c \in C} support_c}$$

Samples average. This method applies an evaluation metric, e.g. precision, to each sample (a target instance and respective prediction instance) and averages the results.

$$Precision_{samples} = \frac{\sum_{n \in N} precision_n}{|N|}$$

Hamming Loss. This metric calculates the hamming distance between the targets and predictions and penelizes the individual labels. The output represents the fraction of the number of wrong labels compared to the total number of labels. This approach is quite forgiving given that labels are only penalized on individual level. Note that zero is the optimal value.

$$HammingLoss = \frac{1}{|N||C|} \sum_{i=1}^{|N|} \sum_{j=1}^{|C|} Y_i[j] \oplus \hat{Y}_i[j]$$

Exact Match Ratio/Subset Accuracy. This metric applies the concept of accuracy from a single-label classification problem to a multi-label classification problem. It measures the number of exactly correct predictions. It is the most strict approach as it does not account for partially correct predictions.

$$ExactMatchRatio = \frac{1}{|N|} \sum_{i=1}^{|N|} I(Y_i = \hat{Y}_i)$$

Since different methods of evaluation show different perspectives, the performance of the models are evaluated using multiple methods: Exact Match Ratio, Hamming Loss, macro-averaged F_1-score, and micro-averaged F_1-score. However, it is argued that the results of some evaluation methods weigh more heavily

than others. The goal of the model is to predict dish types in deliveries. It is preferred that more labels are correctly predicted than that there are more instances for which all labels are correctly predicted. Hence, the result of the Hamming Loss metric is in this context considered to be more important than the result of the Exact Match Ratio metric. A disadvantage of the Hamming Loss metric is that is very optimistic and not quite intuitive. In this problem, 23 class labels are defined but instances usually have only a few labels. An instance with label pizza that is incorrectly predicted label soup, has the hamming loss of $2/23 \approx 0.087$. Two labels are predicted incorrectly (1x FP + 1x FN), while the other 21 labels are predicted correctly (21x TN). The value 0.087 seems quite close to the optimal value of 0. However, this is due to the relatively large number of classes. The prediction is actually not as good.

As described in Section 3.3 the data is label imbalanced. Since the division of dish types in the labelled deliveries is very much influenced by Picnic's selection of recipes, it is assumed that this division is not necessarily the same as the division of dish types in the complete set of unlabelled and labelled deliveries. The correct class division is unknown, so the choice is made that all classes are considered of equal importance. As a result of this decision the macro-average method, which does not consider the support of each class, is a better fit than the micro-average, weighted average, and samples average methods. The evaluation metric of choice is the F1-score as it balances precision and recall. Additionally, it is commonly used in multi-label text classification [24, 14].

The above mentioned metrics evaluate the general performance of the model. It is also useful to discover the model performance for each classes and each instance size in an aim to better understand why a specific model performs the way it does. Hence, the precision and recall are obtained per class label.

6.2. Frequent Itemset Model

Evaluating the Frequent Itemset Model is a challenging task due to the nature of model and the available labelled data. It is explained in Section 6.2.1 why evaluating the Frequent Itemset Model on the available labelled data does not prove that the model works well in general. However, it is also explained that such an evaluation can indicate that the model is able to detect dish types when many customers cook the same recipe, as is the case in the available labelled data. Hence, Frequent Itemset Model is evaluated on all available labelled deliveries in Section 6.2.2.

6.2.1. Experiment Design

The Frequent Itemset Model is designed to detect dish types in all kind of deliveries, not just deliveries in which a Picnic recipe was purchased. However, only the labels of deliveries linked to Picnic recipes are known. This makes the dataset irregular in multiple ways as described in Section 3.1.2.

If the Frequent Itemset Model were to be evaluated on this dataset, two outcomes can be expected. Firstly, having examples of identical recipes in the dataset simplifies the problem. For instance, the recipe "Taco's van tilapiafilet met pittige tomatensalsa" includes ingredients "tacos" and "tilapia fillet". Normally, these two articles might not be purchased together often nor be highly "connected". However, the labelled dataset consists of many deliveries in which this recipe was purchased, so an itemset consisting of these articles will likely be detected. Secondly, due to missing labels there is the possibility that predictions are classified as false positives while in fact they are true positives. For instance, a delivery is labelled with sushi but the customer also cooked a stamppot without using Picnic's recipe page. The Frequent Itemset model could predict that the customer cooked a stamppot in this delivery, but the delivery is not labelled with this dish type. Therefore, it will be considered a misclassification during evaluation. As a consequence, any evaluation involving a single evaluation metric using this dataset is not a fair representation of the model performance.

Instead, the attention is focused to design an experiment which can indicate whether the Frequent Itemset Model is able to identify the ingredients used in recipes in the case that many customers cook the same recipes, as is the case in the available labelled dataset. The Frequent Itemset Model is run on all labelled deliveries. The settings with which the model was ran, are listed in the caption of Table 6.2. These settings are obtained by loosely and manually estimating them.

6.2.2. General Performance

The resulting itemsets and predicted dish types are shown in the first two columns of Figure 6.2. This run obtains an exact match ratio of 0.35 and a hamming loss of 0.0380. These values are shared for completeness but are not a good representation of how well the model performs in general. It is indeed the case that multiple of the detected itemsets contain similar articles/ingredients compared to the Picnic recipes. For some

FIM resulting dish types and itemsets

Detected dish type and itemset		**Related recipe**	
Dish type (predicted)	Itemset	Recipe name	Dish type (true)
curry	{currypasta, bimi, aardappelblokjes, kokosmelk, tofu, rundvlees tartaar, naan brood knoflook koriander, pangasius, koriander}		
curry	{currypasta, bimi, kokosmelk, wokgarnalen knoflook, mais}	Thaise rode curry met garnalen, bimi en babymaïs	curry
curry	{garam masala kruidenmix, kokosmelk, naan brood, bloemkool, linzen}	Indiase rode linzenstoof met bloemkool en naan	curry
curry	{saffraan, rijst risotto, peterselie, wokgarnalen knoflook, sperziebonen, kipdijfilet}	Paella met sperziebonen	paella
curry	{garam masala kruidenmix, kokosmelk, naan brood knoflook koriander, bloemkool, linzen, bouillonblokjes groente}		
curry	{currypasta, bimi, kokosmelk, citroengras, pangasius, koriander}	Thaise viscurry met bimi	curry
flatbread	{munt, yoghurt grieks, falafel, hüttenkäse, veldsla melange, wraps}	Bietenwraps met falafel en frisse yoghurt-muntdressing	wraps
flatbread	{shoarmareepjes, munt, ijsbergsla, libanees platbrood, shoarma, veldsla, pitabroodjes}	Libanees platbrood met shoarma en tzatziki	flatbread
grains	{kikkererwten, hot curry, ras hanout kruidenmix, granaatappel, couscous, citroen, feta}	Couscous met granaatappel en paprika	grains
grains	{munt, mascarpone, ras hanout kruidenmix, bouillonblokjes vlees, paneermeel, peterselie, italiaanse pancetta, couscous, citroen}	Gevulde paprika met taboulé	grains
lasagne	{ricotta, pompoen, lasagnebladen, scholfilet, citroen, tomaten}	Lasagne met kip, pompoen../Tagliatelle met ricotta..	lasagne/pasta
nasi	{bloemkoolroosjes, rijst, taugé, spitskool, sperziebonen, uitjes, satésaus}	Gado-gado	salad
nasi	{japanse wokgroenten, gehaktballen, noodles, ketjap, chinese wokgroenten, cashewnoten, soja}		
nasi	{japanse wokgroenten, mie, cayenne pepers, bosui, woksaus}	Eiermie met pittig gehakt en wokgroenten	noodles
noodles	{tempeh, taugé, satésaus}		
noodles	{paksoi, peulen, noodles, pinda's, koriander, woksaus}		
noodles	{paddenstoelen, oosterse wokgroenten, wokgarnalen knoflook, bosui, noodles, sugar snaps, woksaus}	Udonnoedels met garnalen in teriyakisaus	noodles
noodles	{taugé, spitskool, prei, mie, boemboe bami goreng}	Snelle bami met kip	noodles
noodles	{bosui, sesamzaad, shiitake, rundvlees tartaar, woksaus}		
pasta	{braadworst, tomatenblokjes, tomatenpuree, roerbakgroente italiaans, venkelzaad, tomaten}	Spaghetti bolognese met balsamico	pasta
pasta	{gnocchi, pesto, chorizo worst, parmezaan, basilicum, spruiten, chorizo}	Gebakken gnocchi met chorizo en spruitjes	gnocchi
pasta	{soepballetjes, soepgroente, zongedroogde tomaat, stokbroodjes, tomaten, kruidenboter}		
pasta	{asperges groen, rijst risotto, citroen, bieslook, wokgarnalen, sjalot}	Citroenrisotto met garnalen en groene asperges	risotto
pasta	{tomatenblokjes, macaroni spaghetti groenten, jalapeno pepers, maaltijdmix spaghetti, kruidenmix mexicaans}		
pasta	{peterselie, hollandaise saus, ham, aardappelen vastkokend, asperges}	Witte asperges met ham en hollandaisesaus	potato
pasta	{slagroom, spekjes, krulpeterselie, basilicum}		
pasta	{italiaanse kruiden, pesto, mozzarella, parmezaan, basilicum, walnoten}	Rigatoni in een romige avocadopesto	pasta
pasta	{rijst risotto, mozzarella, aubergine, basilicum, parmezaan}	Rode risotto met burrata	risotto
pasta	{tomatenblokjes, tomatenpuree, knolselderij, italiaanse pancetta, krulpeterselie, basilicum}	Pappardelle winter bolognese	pasta
pasta	{pecorino, oregano, aubergine, tomaten}	Spaghetti alla norma	pasta
pasta	{rucola, tijm, romanesco bloemkool, slagroom, parmezaan, citroen, sugar snaps}	Pasta primavera	pasta
pasta	{braadworst, tomatenblokjes, tomatenpuree, roerbakgroente italiaans, azijn balsamico, tomaten}	Spaghetti bolognese met balsamico	pasta
pizza	{rucola, mozzarella, kookroom, aardappelschijfjes, prosciutto crudo, quiche taartdeeg, chorizo}		
potato	{bieslook, aubergine, zalm, citroen, aardappelen krieltjes}	Traybake met zalm, aubergine en krieltjes	potato
quiche	{kookroom, broccoliroosjes, spekjes, feta, bladerdeeg}	Quiche met broccoli, spekjes en feta	quiche
quiche	{ravioli ricotta italiaanse kruiden, pesto, slagroom, paddenstoelen kastagne, pijnboompitten}	Verse ravioli met kastanjechampignons en spinazie	pasta
quiche	{bonen, kidneybonen, maïs}		
soup	{rozemarijn, aardappelblokjes, kookroom, parmezaan}	Ovenschotel van zoete aardappel en kip	potato casserole
stamppot	{boerenkool, rookworsten, aardappel, feta}	Boerenkoolstamppot met vegaballetjes en zoete aardappel	stamppot
stamppot	{aardappelen kruimig, spekjes, hutspot}	Hutspot met geroosterde paprika en basilicum	stamppot
stamppot	{kruimige aardappelen, mosterd, andijvie, chorizo}	Andijviestamppot met chorizospekjes	stamppot
stamppot	{kruimige aardappelen, rundvlees tartaar, zuivelspread, groentespaghetti}		
sushi	{sushirijst, sojabonen, zalm}	Poké bowl met zalm, edamame en avocado	sushi
sushi	{mie, nasi bami groenten, maaltijdmix bami, kipdijfilet, uitjes}		
wraps	{aardappelpartjes, ijsbergsla, kipschnitzel, stokbroodjes, cheddar, kip schnitzel gepaneerd, wraps, citroensap, kipdijreepjes shoarma}	Krokante kipburger/Kipschnitzel-wrap	hamburger/wraps
wraps	{tomatenblokjes, wraps, kruidenmix mexicaans, maïs}		

Table 6.2: FIM identified itemsets and respective dish types in labelled deliveries.
(support=0.003, lift=5, linkage method=average, cluster number estimation method=SC, min matching score=0.7, min matching ingredients=3)

itemsets, the related Picnic recipe and its dish type are shared in the third and fourth column. Although it is not a quantitative approach, it does provide more insights into the intermediary results. The rest of this section shares positive and negative cases. These visible events arise from larger advantages and disadvantages of the Frequent Itemset Model, which are discussed in Section 7.1.

In general, Figure 6.2 shows that the model, for the given dataset and settings, is able to detect articles that are purchased together in the context of cooking a dish. It can be seen that in multiple cases it clusters the ingredients of the Picnic recipes. For instance, for the recipe “Couscous met granaatappel en paprika” it clusters the ingredients For instance, for the not so traditional recipe “Lasagne with chicken, pumpkin, and ricotta” it clusters its ingredients lasagne sheets, pumpkin, ricotta, lemon, and tomatoes. The ingredients it does not pick up are carrot, zucchini, minced chicken, and onion. It also mistakenly adds plaice fillet. These results do not demonstrate how well the Frequent Itemset Model performs exactly, but it does show that the model can detect articles that are often purchased together to create a meal.

Even though the model seems to output itemsets that “make sense” and can be linked to the original recipes, the model also make mistakes of which some can be clearly spotted in Figure 6.2. Firstly, the model has clustered ingredients of similar recipes into one itemset instead of creating two separate itemsets. This, for example, is the case for “Krokante kipburger met rosevalaardappeltjes” which contains ingredients such as chicken schnitzel, iceberg lettuce, tomatoes, and baguette, and “Kipschnitzel-wrap met tzatziki” which contains similar ingredients including chicken schnitzel, iceberg lettuce, tomatoes, and wraps. The opposite, where the two very similar itemsets are created, can also occur. One example is that there exist two similar itemsets which consist of articles such as bratwurst, tomatoes, and Italian vegetables. The only difference it that one itemset contains fennel seed while the other contains vinegar balsamic. Both represent the ingredients of the recipe “Spaghetti bolognese met balsamico”. Thirdly, certain recipes are not detected in this experiment. For instance, the recipe “Plaattaart met roquefort en peer” is not picked up, because the chosen support value is too high . As expected, the figure also shows an example where an itemset, {beans, kidney beans, corn}, is detected which cannot be linked to a Picnic recipe. Although this is not necessarily a mistake by the model, the evaluation set-up does consider it as such. This confirms why an evaluation based on solely evaluation metrics is not appropriate and why the Frequent Itemset Model (and Supervised Learning Model) cannot easily be compared on this dataset. Finally, the model does not always match the correct dish type to an itemset. This is the case for example for itemset {saffraan, rijst risotto, peterselie, wokgarnalen knoflook, sperziebonen, kipdijfilet} which is matched to dish type curry, but should be matched to dish type paella (it matches the ingredients of the recipe “Paella met sperziebonen").

In conclusion, the Frequent Itemset Model is able to detect dish types of popular recipes. A quantitative output is lacking, however, it is clear that the model also fails to detect labels and makes wrong predictions.

6.3. Supervised Learning Model

The Supervised Learning Model is trained and evaluated on labelled deliveries. The design of the experiments, including concerns on how well the results represent the success of this dish type detection model, are discussed in Section 6.3.1. Two preliminary experiments are run to define a subsample size and specifications of an engineered feature in Section 6.3.2. The results of comparing various algorithms in combination with various feature sets are shown in Section 6.3.3. Section 6.3.4 presents the results of the performance per class label.

6.3.1. Experiment Design

As is described in Section 3.1.2, the available labelled dataset does not reflect reality. This raises some concerns on the effectiveness of (training) the model as well as concerns on the evaluation of the model. This section points out these concerns before the goal and the design of the experiments are described.

Two main issues are identified when training the Supervised Learning Model on the available labelled dataset. The first is the little variety of recipes within the dish types. As is described in Section 3.3.3 there are multiple dish types for which there currently exist only one or two Picnic recipes. This makes generalization a challenge. The second issue is that the ingredients in a recipe are linked to individual articles. Customers are not restricted to choose from these specific articles, but Section 3.3.5 does describe that the majority of customers purchase multiple (distinct) articles linked to a recipe. The poke bowl recipe contains the article “cucumber" as one of the ingredients, but the article “organic cucumber" could be used interchangeably. It is argued that training on individual articles should be prevented, as it is assumed not to be generalizable towards new recipes. Additionally, it may become problematic when articles in the store are updated. Alto-

gether, training on the current dataset might not prepare the model to detect dish types in all Picnic deliveries.

These insights are important to point out since an evaluation using the available labelled dataset is expected to not clearly uncover such shortcomings. Evaluating the model on this dataset might very well output perfect results, but that does not imply that the model is able to detect dish types correctly in all Picnic deliveries. It would, however, imply that the model can detect the dish type of recipes in deliveries of the exact same recipes it has trained on. Hence, the goal of the experiments in this section is to find out how well the Supervised Learning Model is able to detect the dish types of in deliveries which contain the same recipes that have specifically been fed as examples. This situation can be regarded as a simplified version the true problem. The results could be used as an indication on the upper bound on how well the model would be able to perform on “real" data, given the current methodology.

The main experiment evaluates the performance of the model on different feature sets and algorithms. The results of this experiment are influenced by the number of subsamples and the settings of the article overlap feature. Therefore, two preliminary experiments are run. The goal of the first experiment is to obtain a subsample size for which the different algorithms can be compared, because training on all available instances is practically infeasible. Hence, the first experiment compares different algorithms and features with different subsample sizes. The list of subsample sizes is {1000, 2000, 4000, ..., 64000}. All available algorithms are compared, of which the full names are shown in Table 6.1. Both the one-hot encoding (with category levels 2 and 3) and the articles overlap (with settings category levels 2 and 3, per recipe, recipe coverage) feature representations are compared. For each sample size five runs where the samples are retrieved at random are applied. The mean and standard deviation of each run is observed. The evaluation metric is macro-averaged F_1-score. The second preliminary experiment aims at finding out which settings for the article overlap feature can best be used for each algorithm. For all settings of the article overlap features (per recipe/per dish type + recipe coverage/Jaccard index), different algorithms are compared on different category level representations. This includes all available algorithms except SVM, and category level representations 2, 3, 2+3, 4, and 2+3+4. The experiments are run on a fixed subset of the dataset using 5x repeated 5-fold cross validation. The main experiment also uses a fixed subset of the dataset in combination with 5x repeated 5-fold cross validation and evaluation metric macro-averaged F_1-score. It compares the different algorithms with different feature sets. The used sample size is decided by the first preliminary experiment. The used settings of the article overlap can differ per algorithms and is decided by the second preliminary experiment.

6.3.2. Preliminary Results

Two aspects are taken into account before comparing the performance of various algorithms and feature sets: the sample size and the settings of the article overlap feature. Table A.1 and Figures A.1 and A.2 indicate that the performance of the algorithms seem to converge. A sample size of 20,000 is selected for the main experiment. The available resources did not allow a larger sample size nor the Support Vector Machine algorithm to be considered in the comparisons in the remainder of this section.

Table A.2 presents the results of running various combinations of settings, algorithms, and feature sets in order to examine the effect of the settings on the model performance. The evaluation metric is F_1-score. The term “settings of the article overlap feature” refers to two binary decisions. Firstly, to define an overlap score (used as feature) for each recipe per dish type or per recipe (per dish type vs. per recipe). Secondly, to define the overlap score as the percentage of unique recipe items purchased in a delivery compared to the total unique items of a recipe, or the Jaccard index of the unique items in a delivery and a recipe (recipe coverage vs. Jaccard index). Details can be found in Section 5.2.2. The results show that different algorithms perform better given different settings. A theory is shared in the appendix. In the next measurements the settings are set depending on the algorithm. Per recipe/recipe coverage: LR, DT, XGB, MLP; per dish type/jaccard index: kNN, ML-kNN; per dish type/recipe coverage: RF.

6.3.3. General Performance

The goal is to learn for which feature set and algorithm the Supervised Learning Model performs the best. It is theorized that training on article level will lead to the model not being generalizable for all Picnic deliveries. Table 6.3 shows the results for Macro-F_1. Tables A.3 and A.4 show the same results but for evaluation metrics EMR, HL, and Micro-F_1. XGBoost outperforms the other algorithms on almost all feature sets given the metric Macro-F_1, but also given the other evaluation metrics. The feature set using one-hot encoding of category level 4 (0.8501) and category levels 2+3+4 (0.8491) give the best result. The feature set using article overlap using category level 4 (0.8228) and category level 2+3+4 (0.8202) also return good results.

metric: Macro-F_1	LR	kNN	DT	RF	XGB	ML-kNN	MLP
One-Hot Encoding							
Category level 2	0.53 (0.009)	0.34 (0.008)	0.45 (0.009)	0.27 (0.008)	**0.61 (0.010)**	0.46 (0.008)	0.52 (0.009)
Category level 3	0.75 (0.007)	0.54 (0.010)	0.69 (0.009)	0.52 (0.010)	**0.82 (0.007)**	0.68 (0.008)	0.72 (0.008)
Category levels 2+3	0.75 (0.010)	0.55 (0.010)	0.69 (0.008)	0.52 (0.009)	**0.82 (0.008)**	0.67 (0.012)	0.73 (0.011)
Category level 4	0.79 (0.005)	0.57 (0.012)	0.74 (0.006)	0.63 (0.011)	**0.85 (0.008)**	0.72 (0.012)	0.78 (0.006)
Category levels 2+3+4	0.79 (0.006)	0.60 (0.013)	0.74 (0.007)	0.65 (0.008)	**0.85 (0.005)**	0.72 (0.009)	0.77 (0.006)
Article Overlap							
Category level 2	0.46 (0.011)	0.36 (0.012)	0.41 (0.010)	0.37 (0.012)	0.54 (0.010)	0.41 (0.009)	**0.55 (0.012)**
Category level 3	0.64 (0.009)	0.65 (0.009)	0.64 (0.009)	0.67 (0.008)	**0.77 (0.005)**	0.68 (0.008)	0.75 (0.005)
Category levels 2+3	0.69 (0.011)	0.59 (0.009)	0.64 (0.008)	0.65 (0.011)	**0.77 (0.007)**	0.65 (0.009)	0.76 (0.008)
Category level 4	0.70 (0.011)	0.74 (0.011)	0.71 (0.008)	0.76 (0.012)	**0.82 (0.007)**	0.76 (0.010)	0.80 (0.007)
Category levels 2+3+4	0.76 (0.009)	0.69 (0.009)	0.71 (0.009)	0.76 (0.007)	**0.82 (0.007)**	0.73 (0.011)	0.80 (0.006)

Table 6.3: General Model Performance: Supervised Learning Model performance per metric, feature set, and model. See Tables A.3 and A.4 for evaluation metrics EMR, HL, and Micro F_1. The experiments are run on a fixed dataset of 20,000 samples using 5x repeated 5-fold cross validation.

6.3.4. Performance per Class

Table 6.4 shows the precision and recall per class for four different models using XGBoost from the same experiments as Section 6.3.3. The cell colors are in line with the values in the cells. The model with the highest value per class is highlighted in bold. In most cases XGBoost with one-hot encoding feature set category level 2+3+4 outperforms the other models. The precision of all classes is quite high. Dish type *sushi* has the lowest precision. The recall is relatively low compared to the precision. Particularly, dish types *hamburger*, *soup*, *spring roll*, and *stamppot* have a low recall. Overall, the macro-averaged mean of precision is 0.935, and 0.783 for recall for XGBoost with one-hot encoding feature set category level 2+3+4.

	XGB							
	Precision				Recall			
	One-Hot Encoding		Article Overlap		One-Hot Encoding		Article Overlap	
Class	**2+3**	**2+3+4**	**2+3**	**2+3+4**	**2+3**	**2+3+4**	**2+3**	**2+3+4**
curry	0.93 (0.015)	**0.94 (0.017)**	0.93 (0.017)	0.93 (0.016)	0.86 (0.026)	**0.87 (0.021)**	0.84 (0.022)	0.86 (0.017)
flatbread	0.90 (0.021)	**0.94 (0.014)**	0.89 (0.022)	0.93 (0.016)	0.78 (0.021)	**0.85 (0.019)**	0.65 (0.028)	0.76 (0.024)
gnocchi	0.92 (0.026)	0.93 (0.025)	0.94 (0.021)	**0.94 (0.021)**	0.88 (0.025)	**0.88 (0.025)**	0.83 (0.029)	0.85 (0.031)
grains	0.93 (0.019)	0.94 (0.014)	0.94 (0.018)	**0.95 (0.014)**	0.82 (0.020)	**0.83 (0.023)**	0.79 (0.020)	0.82 (0.024)
hamburger	0.91 (0.038)	0.92 (0.038)	0.89 (0.047)	0.92 (0.039)	0.60 (0.067)	0.62 (0.068)	0.56 (0.064)	**0.65** (0.060)
lasagne	0.97 (0.039)	**0.98 (0.035)**	0.95 (0.041)	0.96 (0.036)	0.79 (0.082)	0.83 (0.060)	0.75 (0.083)	**0.87 (0.078)**
nasi	0.90 (0.022)	**0.91 (0.027)**	0.86 (0.036)	0.88 (0.031)	0.74 (0.028)	**0.80 (0.028)**	0.69 (0.040)	0.77 (0.032)
noodles	0.90 (0.014)	0.90 (0.013)	0.89 (0.013)	**0.91 (0.11)**	0.86 (0.011)	**0.87 (0.011)**	0.84 (0.011)	0.86 (0.017)
omelet	0.93 (0.031)	**0.96 (0.019)**	0.93 (0.031)	0.94 (0.024)	0.68 (0.043)	**0.74 (0.040)**	0.59 (0.036)	0.66 (0.061)
paella	**0.96 (0.030)**	0.96 (0.035)	0.94 (0.046)	0.94 (0.037)	0.79 (0.043)	**0.83 (0.062)**	0.77 (0.057)	0.82 (0.048)
pasta	0.90 (0.007)	**0.91 (0.006)**	0.88 (0.010)	0.91 (0.008)	0.81 (0.010)	**0.84 (0.009)**	0.79 (0.008)	0.83 (0.011)
pizza	0.87 (0.032)	0.90 (0.033)	0.88 (0.033)	**0.92 (0.034)**	0.68 (0.044)	**0.72 (0.039)**	0.55 (0.044)	0.66 (0.058)
potato	0.89 (0.022)	0.92 (0.018)	0.87 (0.025)	**0.92 (0.018)**	0.73 (0.024)	**0.77 (0.024)**	0.65 (0.029)	0.72 (0.023)
potato casserole	0.95 (0.017)	**0.95 (0.018)**	0.93 (0.021)	0.95 (0.017)	0.80 (0.037)	**0.81 (0.022)**	0.76 (0.019)	0.79 (0.033)
quiche	0.93 (0.029)	**0.94 (0.018)**	0.93 (0.024)	0.94 (0.021)	0.82 (0.032)	**0.83 (0.033)**	0.77 (0.022)	0.80 (0.032)
rice	0.94 (0.019)	**0.94 (0.017)**	0.94 (0.016)	0.94 (0.013)	0.76 (0.026)	0.78 (0.022)	0.76 (0.021)	**0.78 (0.029)**
risotto	0.93 (0.026)	**0.94 (0.023)**	0.93 (0.025)	0.93 (0.029)	0.77 (0.024)	**0.79 (0.034)**	0.65 (0.036)	0.72 (0.038)
salad	0.96 (0.014)	**0.98 (0.012)**	0.93 (0.019)	0.95 (0.016)	0.77 (0.024)	0.80 (0.023)	0.76 (0.033)	**0.81 (0.025)**
soup	0.93 (0.053)	**0.96 (0.036)**	0.93 (0.055)	0.94 (0.034)	0.51 (0.060)	**0.58** (0.061)	0.36 (0.054)	0.47 (0.053)
spring roll	0.92 (0.076)	**0.96 (0.051)**	0.88 (0.099)	0.94 (0.066)	0.60 (0.097)	**0.69 (0.090)**	0.41 (0.092)	0.62 (0.121)
stamppot	0.92 (0.021)	**0.93 (0.024)**	0.89 (0.032)	0.92 (0.023)	0.71 (0.027)	**0.77 (0.029)**	0.58 (0.039)	0.67 (0.036)
sushi	0.80 (0.086)	**0.84 (0.072)**	0.74 (0.157)	0.75 (0.112)	0.54 (0.107)	**0.64** (0.099)	0.45 (0.115)	0.55 (0.100)
wraps	0.93 (0.013)	**0.95 (0.012)**	0.91 (0.015)	0.94 (0.008)	0.82 (0.017)	**0.84 (0.011)**	0.78 (0.023)	0.82 (0.021)
(macro-)mean	0.919	**0.935**	0.905	0.924	0.745	**0.783**	0.677	0.745

Table 6.4: Precision (left) and recall (right) per class and model (one-hot encoded features).

6.4. Estimated Performance on Unlabelled Deliveries

In this section an attempt is made to estimate the performance of the models on unlabelled deliveries. These unlabelled deliveries are deliveries which are not linked to Picnic recipes. The customer did not purchase articles from the recipe feature page. A small manually labelled dataset is obtained, and predictions of various

versions of the Frequent Itemset Model, Supervised Learning Model, and Hybrid Model are observed. The Hybrid Model is the Supervised Learning Model which is trained on the results of the Frequent Itemset Model (and partly on available labelled dataset from the recipe page). The experiment design is described in Section 6.4.1. Information on how the manually labelled dataset is obtained and how the experiments are designed are described in Section 6.4.2. The results of the Frequent Itemset Model, Supervised Learning Model, and Hybrid model are shared in Section 6.4.3, 6.4.4, and 6.4.5 respectively.

6.4.1. Experiment Design

Previous experiments have focused on evaluation on labelled data, which only considers deliveries linked to Picnic recipes from the recipe feature page. However, the goal of the dish type detection model is to perform well on all deliveries. Hence, an attempt is made to evaluate the models on truly labelled deliveries. A very small dataset is obtained through the author's personal account and labelled manually before applying the models. The Frequent Itemset Model, the Supervised Learning, and a combination of both (the Hybrid Model) make predictions for all 19 instances in the small dataset. The Frequent Itemset Model is ran twice with different settings and sample sizes. The settings are manually chosen based on multiple sample runs, and are not proven to output optimal results. The Supervised Learning Model is trained on 50,000 random samples before the predictions are made. The Hybrid model is trained once on 50,000 labels including 9,163 weak labels which are predicted by the Frequent Itemset Model, and once on these 9,163 weak labels alone. The predicted classes are displayed instead of evaluation metrics to have more insights into misclassifications. Since the test set is extremely small and linked to a single customer, the results of this section serve as a rough estimation.

6.4.2. Small Manually Labelled Dataset

Table 6.5 shows the 19 manually labelled deliveries. For each delivery the identified dish types are shown together with a description of the dish, the articles that were purchased to cook the dish, and relevant notes about the delivery. Note that some dishes can be regarded as quite standard dishes in the Netherlands, such as nasi goreng, "Mexican" wraps, and classic hamburger, while others are probably less popular, such as shakshuka, and tacos with mushrooms. It also stands out that multiple dishes, such as ceviche, eggplant with pomegranate, and chicken stew do not belong to any of the defined classes.

Dish type	Dish description	Related purchased articles/ingredients (intuitive description)	Notes
nasi	Nasi goreng	boemboe, bami/nasi vegetables, rice, vegan chicken, satay sauce, kroepoek	
grains	Pearl couscous with halloumi	pearl couscous, halloumi, peaches, broccoli, tomatoes	
potatoes	New potatoes with jackfruit stew	new potatoes, jackfruit, carrot, celery, clove, ginger, tomato paste, garlic	
pasta	Pasta with tuna	broccoli, roasted bell pepper, olives, pecorino, tuna	Did not purchase pasta.
wraps	Wraps with salmon	wraps, salmon, spring onions, avocado, tomatoes, mango	
curry	Non-traditional curry	naan, rice, diced tomatoes, halloumi, cooking cream	
gnocchi	Pesto gnocchi	gnocchi, basil, pine nuts, pecorino, mozzarella, zucchini	
pizza	Burrata pizza	vegetable pizza base, burrata, buffalo mozerella, pecorino, basil, tomatoes	Purchased four types of pasta.
rice	Rice with salmon	brown rice, salmon, spring onion, lettuce, lemon	
salad	Halloumi salad	halloumi, peaches, broccoli, peaches, tomatoes	Salad does not contain lettuce.
wraps	"Mexican" wraps	wraps, corn, bell pepper, avocado, black beans, tortilla chips, taco sauce	
-	Parmigiana di melanazane	eggplant, buffalo mozzerella, tomatoes	
flatbread	"Pita gyros"	pita, vegan gyros, garlic sauce, tomatoes, lettuce	
-	Ceviche	salmon, lime juice, cayenne peppers, cucumber, avocado, coriander, pomegranate	
omelet	Shakshuka	cannellini beans, bell pepper, diced tomatoes, feta, chili, flat bread	Did not purchase egg for the shakshuka.
pizza	Beetroot pizza	vegetable pizza base, beetroot, zucchini, rocket, hazelnuts, goat cheese	Purchased flatbread for the shakshuka.
salad	Salad of salmon, beetroot, and lentils	smoked salmon, beetroot, red lentils	Salad does not contain lettuce.
sushi	Poke bowl	sushi rice, wakame salad, surimi, seaweed sheets, edamame, avocado	
-	Eggplant	eggplant, pomegranate, feta, mint	
potato	Fish, potatoes, and veggies	new potatoes, fish, tomatoes, broccoli	
salad	Salade Niçoise	lettuce, tuna, olives, eggs, green beans, cherry tomatoes, baguette	
curry	Chicken curry	vegan chicken, ginger, coconut milk, yoghurt, cardamom, cinnamon sticks, nutmeg, cilantro, tomato	Did not purchase rice or naan.
-	Kapsalon	vegan shawarma, fries, lettuce, garlic sauce, cucumber, tomatoes	Similar ingredients to class flatbread.
gnocchi	Pesto gnocchi	gnocchi, basil, pecorino, almonds, green beans	
noodles	Noodles with salmon	egg noodles, salmon, bean sprouts, bell pepper, zucchini	
curry	Curry with chickpeas	chickpeas, coconut milk, sweet potatoes, onion	Did not purchase rice nor naan.
quiche	Vegetable pithivier	puff pastry, asparagus, green beans, peas, lemon, goat cheese, thyme, eggs	
wraps	Pulled paddo tacos	taco shells, oyster mushrooms, red cabbage, chilli, caster sugar, ketchup	
pizza	Eggplant pizza	vegetable pizza base, passata, buffalo mozzarella, eggplant, basil	
quiche	Quiche Lorraine	puff pastry, cooking cream, vegan bacon, goat cheese, mushrooms, leek	Purchased wraps but not related to a recipe.
noodles	Teriyaki salmon noodles	noodles, salmon, ginger, broccoli, carrot, caster sugar	
pizza	Vegetarian + salami pizza	vegetable pizza base, passata, buffalo mozzarella, salami, eggplant, mushrooms	
potato	Traybake with fish and potatoes	potato wedges, cod fillet, tomatoes, red onions, zucchini	
potato	Mashed potatoes with vegetarian chicken stew	floury potatoes, vegetarian chicken, mushrooms, mustard, red onion, cooking cream	
hamburger	Classic hamburger	hamburger buns, vegetarian burger, tomatoes, onions	
wraps	"Mexican" wraps	wraps, dices tomatoes, corn, bell peppers, black beans, onions, goat cheese	

Table 6.5: Manually labelled deliveries. These deliveries previously fall under the "unlabelled deliveries" as they do not contain articles from the Picnic recipe page.

6.4.3. Predictions Frequent Itemset Model

Depending on the settings of the model, the results can be different. Table 6.6 shows the obtained itemsets and the matched dish types for two runs. As a result, the predictions in Table 6.7 are different. In green cells all dish types are predicted correctly, in orange cells some dish types are predicted correctly, in red cells no dish types are predicted correctly, and in grey cells no predictions are made. Dish types *gnocchi*, *pasta*, and *pizza* are confused with one another. The first run detects dish type *flatbread* for a delivery in which the dish kapsalon was cooked. This can be interpreted as correct, however, the dish was not labeled with a dish type.

Run 1	
20,000 + 19 unlabelled deliveries	
support=0.005, lift=3, average, Silhouette coefficient, min matching score=0.7, min matching ingredients=3	
Dish type	**Itemset**
flatbread	garlic sauce, pita breads, iceberg lettuce
grains	blackberries, berries, strawberries, pomegranate, raspberries
nasi	meal mix nasi, nasi bami vegetables, satay sauce, rice, kroepoek
noodles	spring onions, wok sauce, noodles
pasta	tomatoes, zucchini, eggplant, mozzarella
pasta	tomatoes, tomato paste, diced tomatoes
pasta	basil, eggplant, diced tomatoes
pizza	mozzarella, pizza dough, rocket, basil, pizza base, pesto
stamppot	sauerkraut (“zuurkool"), endive (“andijvie"), smoked sausages, floury potatoes, bacon, kale (“boerenkool")
wraps	crème fraîche, beans, corn, mexican sauce, mexican spice mix, wraps
wraps	beans, corn, diced tomatoes, wraps, kidney beans

Run 2	
50,000 + 19 unlabelled deliveries	
support=0.005, lift=3, average, Silhouette coefficient, min matching score=0.7, min matching ingredients=3	
Dish type	**Itemset**
curry (1437)	zucchini, curry pasta, eggplant, coconut milk, chickpeas
curry (544)	coriander, lime, coconut milk, spring onions, garlic
nasi (1247)	satay sauce, kroepoek, rice, nasi bami vegetables, meal mix nasi
pasta (521)	pine nuts, pesto, rocket
pasta (914)	tomato pasta, chickpeas, kidney beans, diced tomatoes
pizza (883)	pizza dough, salami, mozzarella, basil
stamppot (1997)	smoked sausages, bacon, sauerkraut, “hutspot", gravy, floury potatoes, kale
stamppot (776)	smoked sauges, smoked sauge, bacon, endive, floury potatoes
wraps (385)	indian chicken tandoori, crème fraîche, mexican burritos
wraps (2273)	mexican sauce, beans, wraps, corn, kidney beans, crème fraîche, tacos, mexican spice mix
wraps (529)	parsley, cayenne pepper, spring onions, garlic
11,005 predictions over 9,172 deliveries	

Table 6.6: Resulting itemsets and their dish types of the Frequent Itemset Model for different runs.

	FIM	
	Run 1	**Run 2**
{nasi}	{nasi}	{nasi}
{grains, potatoes}		
{pasta, wraps}		
{curry, gnocchi}	{pasta, pizza}	{pizza}
{pizza}	{pizza}	{pizza}
{rice}		
{salad, wraps}	{wraps}	{wraps}
{flatbread}	{flatbread}	
{omelet, pizza, salad}		
{sushi}		
{potato, salad}		
{curry}	{flatbread}	{curry}
{gnocchi, noodles}		
{curry}		{curry}
{quiche, wraps}		
{pizza}	{pasta (2x), pizza}	{pizza}
{quiche, noodles}		
{pizza, potato}	{pasta (2x), pizza }	{ curry, pizza }
{hamburger, wraps}	{wraps}	{wraps}

Table 6.7: FIM predictions on manually labelled dataset. The settings are shown in Table 6.6.

6.4.4. Predictions Supervised Learning Model

The predictions of various versions of the Supervised Learning Model (different feature sets and algorithms) are shown in Table A.5. Overall, it is clear that the Supervised Learning Model trained on the labelled deliveries does not perform as well on this dataset compared to the results from Table 6.3. The best predictions are made by the logistic regression algorithm in combination with the article overlap feature set category levels 2+3+4. Table 6.8 provides an overview. It also represents the results of the same model trained on features sets category level 3, category level 4, and article. As expected, training on the individual articles does not seem to generalize well enough. Each prediction is of dish type *pasta* and each prediction is incorrect. Similar to the results of the Frequent Itemset Model in Section 6.4.3, dish type *nasi* is predicted correctly. These results also show that dish types *gnocchi*, *pasta*, and *pizza* are confused with one another, and that dish type *flatbread* is detected in the delivery in which the dish kapsalon was made. Table 6.8 also shows that dish type *potato casserole* is detected where the correct label is actually *potato*. It is also striking that for most instances no predictions are made. In this case training on weak labels only causes similar results but with additional, wrong predictions.

	SIM (LR, article overlap)				
	Category level				
	2	**3**	**4**	**2+3+4**	**article**
{nasi}	{salad}	{nasi}	{nasi}	{nasi}	
{grains, potatoes}					
{pasta, wraps}					
{curry, gnocchi}	{pasta}	{pasta}	{pasta}	{pasta}	{pasta}
{pizza}	{pasta}	{pasta}	{pasta}	{pasta}	{pasta}
{rice}					
{salad, wraps}					
{flatbread}					
{omelet, pizza, salad}			{pasta}		
{sushi}					
{potato, salad}	{salad}	{salad}		{salad}	
{curry}			{flatbread}	{flatbread}	
{gnocchi, noodles}	{pasta, noodles}				{pasta}
{curry}					
{quiche, wraps}					{pasta}
{pizza}		{pasta}	{pasta}	{pasta}	{pasta}
{quiche, noodles}	{grains}	{quiche}		{quiche}	
{pizza, potato}	{potato cass.}	{pasta}	{potato cass.}	{potato cass.}	
{hamburger, wraps}	{wraps}				

Table 6.8: SIM predictions on manually labelled dataset.

6.4.5. Predictions Hybrid Model

The Hybrd Model is created by combining the Frequent Itemset Model and the Supervised Learning Model. Predictions of the Frequent Itemset Model are used as weak labels for the SLM to train on (in addition to the already existing labels). In this experiment the predictions of the second run from Table 6.7 are used as weak labels. The Frequent Itemset Model is applied to 50,0019 deliveries, which results in 11,005 predictions over 9,172 deliveries. However, this includes the manually labelled deliveries. To not train on the test set, predictions related to these deliveries are removed. This results in 10,995 predictions over 9,163 deliveries. As shown in Table 6.6, the class labels are curry (1838), nasi (1246), pasta (1420), pizza (879), stamppot (2507), and wraps (3105).

Table A.6 shows the predictions when training the Hybrid Model on a dataset of 50,000 instances using different algorithms and feature sets. This dataset consists of 9,163 weak label from the Frequent Itemset and 40,837 weak labels from the alternative dataset (via the recipe page). The best results are obtained with the article overlap feature set and logistic regression. Table 6.9 shows results for this feature set and algorithm, as well as the results when training only on the 9,163 weak label from the Frequent Itemset Model. Previous results from Table 6.7 and 6.8 have been added for comparison.

A very expected observation is that the results of the Hybrid Model are more similar to the results of the Frequent Itemset Model when training only on the predictions of the Frequent Itemset Model. Similarly, the results of the Hybrid Model are more similar to the results of the Supervised Learning Model when also training on the alternative dataset. More surprisingly, it seems that more predictions are made for the Hybrid Model (9,163 weak labels) compared to the results of the other two models. However, this evaluation involves

19 deliveries and cannot conclude which model has the best performance.

	HM (LR, article overlap, category levels 2+3+4)		FIM (run 2)	SLM (LR, overlap, 2+3+4))
	50,000 (incl. 9.163 weak labels)	**9,163 weak labels**		
{nasi}	{nasi}	{nasi}	{nasi}	{nasi}
{grains, potatoes}				
{pasta, wraps}				
{curry, gnocchi}	{pasta}	{pasta}	{pizza}	{pasta}
{pizza}	{pasta, pizza}	{pizza}	{pizza}	{pasta}
{rice}				
{salad, wraps}	{wraps}	{wraps}	{wraps}	
{flatbread}		{wraps}		
{omelet, pizza, salad}	{pasta}	{pasta}		
{sushi}		{curry}		
{potato, salad}				{salad}
{curry}	{flatbread}	{curry, wraps}	{curry, wraps}	{flatbread}
{gnocchi, noodles}		{curry}		
{curry}		{curry}	{curry}	
{quiche, wraps}				
{pizza}	{pasta}	{curry, pizza}	{pizza}	{pasta}
{quiche, noodles}				{quiche}
{pizza, potato}	{potato cass.}	{curry, pizza.}	{curry, pizza}	{potato cas.}
{hamburger, wraps}	{wraps}	{wraps}	{wraps}	

Table 6.9: HM predictions on manually labelled dataset. Trained on labelled data as well as the 9,163 predictions of run 2 from Table 6.7.

7
Discussion

This chapter discusses in more detail the results of the various models, whose core results have been reported in Chapter 6. The main results are summarized and the pros and cons are discussed for both the Frequent Itemset Model as well as the Supervised Learning Model. In addition to discussing the methodology, the choices of scoping the problem, in particular the class labels, are also reviewed.

7.1. Frequent Itemset Model

The Frequent Itemset Model is created to solve the problem of detecting dish types in Picnic deliveries. It is able to work around the issue of unavailable labels, since it applies unsupervised learning techniques. In short, it removes irrelevant articles from deliveries, selects the article representations using Picnic's product taxonomy, and cleans the article texts. It aims to find core ingredients in three steps: 1) finding frequent itemsets using frequent itemset mining, 2) filtering out itemsets in which the items are not related using association rule mining, and 3) grouping similar itemsets using hierarchical clustering. Itemsets are matched to dish types with the help of a recipe dataset and programmatic labelling. The main findings from the two experiments from Sections 6.2.2 and 6.4.3 are summarized and interpreted in Section 7.1.1. Then in Sections 7.1.2 and 7.1.2 the pros and cons, besides the performance of the model, are identified.

7.1.1. Main Findings

The first experiment described in Section 6.2.2 involves a large, weakly labelled dataset. This dataset is not representative of a true scenario as explained in Section 3.1.2. However, the results show that the model can identify multiple Picnic recipes in the itemsets it retrieves. This indicates that the model is able to identify combinations of articles which are often combined when preparing a meal. It also makes mistakes when finding the itemsets and matching these to dish types. For instance, it does not always correctly cluster the itemsets. Additionally, it does not detect infrequent recipes/dishes if the level of support is too high. It also matches the itemsets to the wrong dish type occasionally.

The second experiment from Section 6.4.3 involves a small, manually labelled dataset. This dataset is more representative, because all labels are confirmed. Since the experiment is based on very few data points, no hard conclusions should be made. However, it can be discussed what this experiment seems to indicate. It shows that results are quite different depending on the dataset and settings. In terms of the performance of the model it seems that relatively popular dishes with distinctive ingredients, such as nasi goreng and pizza, are detected. Less popular dishes with less distinctive ingredients, such as a beetroot salad and shakshuka, are not detected.

Although it is not quantified, the Frequent Itemset Model seems to be able to detect the dish types of common dishes. It identifies patterns that were not discovered before. Besides its performance, the model has additional advantages which make it useful. There are also disadvantages which make the model less practical to use.

7.1.2. Pros

Multiple advantages and strengths of the Frequent Itemset Model are identified. Some aspects are common pros of unsupervised learning, while others are more specific for this particular model, context, and results.

- **No labelled data required.**
  The model applies unsupervised learning techniques, and does not require information on the dishes that customers cooked. Since no truly representative labelled data is available currently, this is considered to be a large advantage. It is also the foremost reason this model has been created.

- **Find hidden patterns.**
  The first phase of the model discovers itemsets purely based on what the data looks like. It does not have any expectations and simply "let's the data speak".

- **Intuitive.**
  Even though there are multiple phases within the model, each phase is quite intuitive. No deep understanding on complex algorithms is required to follow the process. Additionally, this facilitates debugging and improving the model.

- **Adjustable settings.**
  The Frequent Itemset Model require six settings to be set: 1) support, 2) lift, 3) linkage method, 4) optimal number of cluster method, 5) minimum article-ingredient matching score, and 6) minimum number of ingredients to match itemsets and recipes. These settings could be considered as "hyperparameters", since they influence how well the model performs on the labelled dataset. However, these settings can also be viewed as an opportunity to adjust the model according to one's goal. For instance, by using a high support to detect common dishes, or using a high lift to obtain typical dishes.

- **Multiple use cases.**
  An additional advantage is that the model can be applied for different goals aside from the current scope. The steps of finding and filtering frequent itemsets already obtain useful patterns. The retrieved itemsets give insights into complementary items. These insights can be used, for example, to recommend articles or to build a model which predicts uplift in article demand during promotions.

7.1.3. Cons

Disadvantages and weaknesses linked to the model are related to maintenance, usage, performance, and some to robustness.

- **Dependence on product taxonomy.**
  The model is dependent on the product taxonomy, as the product taxonomy is used to select the level of representation of articles. Without the product taxonomy, the category level 4 would be selected for all articles. Since the representation of articles has a large effect on the results, the performance would not be optimized. For each new article that is introduced in the app, the product taxonomy needs to be updated to avoid the default representation of category level 4.

- **Dependence on recipe dataset.**
  The model is also dependent on the recipe dataset. Without the recipe dataset, dish types could not get assigned to the obtained itemsets and the model would not be able to output dish types. One could assign dish types to itemsets by hand, but human intervention is costly. The composition of the recipe dataset influences the matching process between itemsets and dish types. More recipes of a specific dish type makes the matching process biased towards that dish type. This could explain why the itemset related to dish type risotto (120 recipes) is identified as pasta (958 recipes) and the itemset related to dish type paella (14 recipes) is identified as curry (290 recipes). If there is no recipe which matches an itemset that itemset is removed. Hence, variety of recipes that cover the obtained itemsets is essential.

- **No individual predictions.**
  The model is based on discovering patterns in large dataset, hence it is not capable of making a prediction on a single delivery without taking into account other deliveries.

- **Long runtime.**
  Generally, the model does not make predictions fast. The exact runtime depends on the settings. A lower support results in more frequent itemsets, which results in a longer runtime. The output of filtering the recipes and obtaining the article-ingredient matching scores are already stored to save time.

- **No prediction certainty estimation.**
  The model has a binary output of whether a dish type is present in a delivery or not. It does not share the certainty of a prediction, even though some predictions are based on more factors than others. For instance, it can happen that the frequent itemset {corn, tomato} is clustered into the itemset {corn, tomato, avocado, wraps} which is then matched to dish type wraps. As a result the delivery is labelled with dish type wraps purely based on just two articles. Additionally, sometimes a dish type is matched to an itemset based on ten recipes while in other cases it is based on two recipes.
- **Highly dependent on specific phases.**
  One mistake in the matching process between itemsets and dish types (Section 4.6) has disproportionate consequences. If an itemset is matched to the wrong dish type, then all deliveries linked to that itemset obtain the wrong dish type, and all of these predictions are incorrect. Additionally, during clustering it can happen that many (unrelated) items are put into one cluster. Predictions that result from this are also incorrect.
- **Limited country/language scope.**
  Although in line with the scope of the project, the model can only be applied to Picnic deliveries within the Netherlands. All data, that includes the article names, category level descriptions, recipe names, recipe ingredients, and labelling functions, are in Dutch.

7.2. Supervised Learning Model

The Supervised Learning Model applies supervised learning techniques to a dataset with alternative labels obtained through the recipe page in the Picnic app. Multiple versions, i.e. different combinations of feature sets and algorithms, of the Supervised Learning Model are implemented. The features are all related to the articles in the deliveries, but the level of representation varies (e.g. category level 2 vs 3). Most algorithms are combined with problem transformation method binary relevance in order to output a set of labels. Some algorithms are adjusted or are already intrinsically suitable for multi-label classification problems. The main findings from experiments 6.3.3 and 6.4.4 are summarized and interpreted in Section 7.2.1. Sections 7.2.2 and 7.2.3 state the pros and cons of the model.

7.2.1. Main Findings

The first experiment from Section 6.3.3 makes use of a weak labelled data obtained through the recipe page. This dataset is irregular and the exact same recipes are present in many deliveries. Table 6.3 shows that the macro-averaged F_1-score of 0.85 can be reached using XGBoost and one-hot encoding feature set with category level 4 or category levels 2+3+4. The data is trained on 16,000 samples in each fold. Training on a larger training set could still improve performance. The results imply that the model can detect dish types in deliveries in which very similar articles were purchased compared to the recipes it has trained on. Table 6.4 shows that the precision per class is quite high (at most 0.935 on average), while the recall per class is relatively low (at most 0.783 on average).

Even though the experiment involve the alternatively labelled dataset outputs good results, that does not mean that the model performs similarly for all deliveries. The second experiment from Section 6.4.4 shows that the model does not perform as well on a dataset with “true labels”. The best results are obtained using category level 2+3+4 in this dataset. Out of the 19 manually labelled deliveries, the Supervised Learning Model is only able to make one correct prediction involving a common dish. The model does seem to go into “the right direction”. Related dish type are often confused with one another, but apart from these no wrong predictions are made. However, the model also makes relatively few predictions.

7.2.2. Pros

The following pros of the Supervised Learning Model are common advantages of supervised learning.

- **Potential for higher accuracy.**
  Table 6.8 indicates that the Supervised Learning Model does not perform quite well in general. In this experiment only one out of 19 instances have been classified correctly. However, Table 6.3 shows that the model works quite well when detecting the exact same dishes it has trained on. A macro-averaged F_1-score of $> 80\%$ is obtained, and Table A.1 shows that training on more data can further improve performance. Although the current version of the Supervised Learning Model is not accurate, this approach has the potential to be highly accurate if trained on a representative dataset.

- **Individual, fast predictions.**
  The Supervised Learning Model is able to make a prediction for a single delivery in a short amount of time (once it is trained). Hence, it has the potential to be applied in real-time in the app while a customer is adding articles to their basket.

7.2.3. Cons

Besides the performance of the Supervised Learning Model, other general disadvantages are listed.

- **Labelled data required.**
  The model applies supervised learning techniques, which require labelled data to learn from. This is considered as a disadvantage of this model because high quality data needed in this context is hard to obtain, even though it has a large influence on the performance of model.

- **Long training times.**
  Training the model, especially using XGBoost, can take hours or days depending on the feature set and the size of the training set.

7.3. Class Labels

The project uses manually defined class labels (dish types). There is a certain arbitrariness here, and some classes (though distinct) may show remarkable similarity. These class labels have been defined at the start of the project and have not been iterated over. However, they highly influence how well the models perform. In this section findings which concern the class labels are discussed. Some dish types are too similar in terms of ingredients, some dish types do not have recognizable ingredients, and some dishes do not fall under any of the defined dish types. Note that no investigation is performed to support how often these findings are observed, nevertheless the insights can be useful as a starting point for future work.

There are some dish types that are generally hard to distinguish: potato casserole vs potato, and gnocchi vs pasta vs pizza. This is could be explained by the overlap between related ingredients. In the case of the former, the key difference between the two dish types lies in the preparation step instead and not between the ingredients. This is difficult to detect and a known problem in the similar context of cuisine classification [74]. Distinguishing between gnocchi, pasta, and pizza is difficult due to overlapping articles such as tomatoes, tomato pasta, eggplant, mozzarella, and basil. In this case the ingredients pasta, gnocchi, and pizza base would be very indicative. However, both models do not utilize this by, for instance, assigning a larger weight on specific ingredients.

A number of dishes are observed to be classified correctly relatively easily by the Frequent Itemset Model. Examples are curry, nasi, pasta, pizza, stamppot, and wraps. On the other hand, several other dishes are much harder to identify; this holds for e.g. omelet, quiche and soup. This is likely related to how recognizable the ingredients related to the dish are. The same findings are not observed when the Supervised Learning Model is evaluated on the alternatively labelled dataset. In that case the precision is quite high for all dish types. The recall is lowest for dish types soup, spring roll, and sushi. It is unclear why this is the case, but it is noticed that the dish types with a low recall also tend to be the dish types with less samples/deliveries present in the dataset. When the Supervised Learning Model is evaluated on the manually labelled dataset, it shows that it is only able to detect dish types correctly whenever a delivery contains articles which are similar to the ingredients of one of the Picnic recipes. This makes sense given that the article overlap is used for the features.

Lastly, some dishes do not fall under any of the defined class labels. For example, in dishes where meat, poultry, or fish are the main ingredients, such as carpaccio and coq au vin. Additionally, some dishes are not categorized under a class labels even though they are quite related. For instance, the fast food dish “kapsalon” which is usually prepared with fries, meat, cheese, iceberg lettuce, tomato, and cucumber. This dish does not match the defined dish types well, but it does use similar ingredients as dish type flatbread. In one manually labelled delivery this dish was cooked, and both FIM and SLM indeed detect dish type flatbread. Although this insight in based on a single example, it does point out there reconsidering the dish types, and thus problem, could be redefined while taking the ingredients more into consideration.

8

Conclusion

This work is a first attempt towards solving the novel problem of detecting dish types in deliveries from an online supermarket. The lack of labelled data and fuzzy definitions around dishes are identified as the main challenges. Two models have been created with the aim of solving the problem and investigating the usefulness of these approaches in this context.

The first model, named the Frequent Itemset Model, is a unique approach which combines various techniques. It makes use of frequent itemset mining, association rule mining, hierarchical clustering, programmatic labelling, and fuzzy string matching. The experiments indicate that this model is able to detect the dish types of common dishes. The model requires no labelled data, discovers hidden patterns, is intuitive, can be adjusted to specific goals, and can be applied for multiple use cases. However, it is dependent on a product taxonomy and recipe dataset, cannot make individual predictions, does not provide a certainty estimation, and certain mistakes that can be made have a disproportionate effect on the output.

The second model is referred to as the Supervised Learning Model and applies various supervised learning algorithms on various feature sets. It is trained on an alternative labelled dataset obtained through the recipe page in the Picnic app. This dataset is not representative of the true scenario. For instance, due to the extremely improbable situation that all customers cook the exact same recipes from a small selection. Hence, a challenge that arises is generalizing over the available training set. The best results for the weakly labelled dataset are achieved using extreme gradient boosting trees and one-hot encoding feature set with category level 4. A macro-averaged F_1-score of 0.85 is obtained. It should be noted that detecting the dish types Picnic recipes in this dataset is different from the main goal of detecting dish types in all Picnic deliveries. The best results for the manually labelled dataset (but trained on the weakly labelled dataset) are achieved using logistic regression and article overlap feature set with category levels 2+3+4. The results indeed show that the model performance worse on "real data". The model does have the potential to make highly accurate predictions, however, it is expected that a larger variety of labelled data is required to achieve this. Training the model takes a some time, but then individual predictions can be made quickly.

The lack of labelled data limits the Supervised Learning Model as well as the evaluation of both models. More labels from a larger variety of recipes can be obtained over time through the recipe page in the Picnic app. Both approaches can be enhanced. The contribution of this work is the formulation of the problem, two proposed solutions, insights into the challenges, and suggestions for future work.

8.1. Future Work

This thesis has demonstrated that the problem of detecting dish types in supermarket deliveries can be solved to some extend. However, it has also discussed shortcomings of the current approaches. This section makes suggestions for future work based on these shortcomings as well as new ideas. The suggestions concern improvements for the Frequent Itemset Model, the Supervised Learning Model, and new approaches.

It is expected that the performance of the Frequent Itemset Model can be improved by investigating the influence of the product taxonomy, updating the recipe dataset, adding extra weight to important ingredients, and by considering article alternatives when matching articles and ingredients. The product taxonomy is used to select relevant articles and select the representation of articles. How it is defined affects which itemsets are found and hence affects the results. This work did not optimize the product taxonomy, but do-

ing so could potentially improve the results. Secondly, matching the itemsets to recipes is dependent on the recipe dataset. Adding more recipes to this dataset, removing irrelevant recipes, and balancing the division of dish types would likely increase the performance of the Frequent Itemset Model. It might also be an option to cluster the recipes before matching them to itemsets. Options such as this could be explored. Thirdly, adding extra weight to important ingredients is another way to advance the model. In the current approach, all articles and ingredients are considered equally important when matching itemsets and recipes. However, one can imagine that certain ingredients (e.g. lasagne sheets) hold more valuable information than others (e.g. garlic). Incorporating weights based on article relevance for example is recommended as future work. Lastly, the Frequent Itemset Model calculates similarity scores between articles and ingredients using fuzzy string matching. Some articles/ingredients are very related but the text descriptions are quite different. For instance, "baguette" and "stokbrood". An article alternatives dataset can be useful to improve this matching process.

There are also various suggestions for future work to improve the Supervised Learning Model. The foremost advice is to train on a dataset which contains a wider variety of recipes. Ideally, all labels are confirmed. Another option is to gather more labels through the recipe page and investigating whether training on a wider variety of weakly labelled data improves the performance of the model. In that case it can also be explored if some weak labels should be removed. For instance, if a customer only purchases one ingredient from a recipe. Another way to obtain deliveries of a wider variety of recipes, could be through data augmentation. Secondly, more features can be added and the current features can be improved. Information about how often a customer purchases an articles has good potential as a feature based on the analysis run in this work. Exploring how this information can be applied, and whether it is useful can be done in future work. One option could be to remove articles from a customer's delivery if that customer purchases the article relatively often. Additionally, information about previous deliveries, app event information, and customers' location can be explored for usefulness in detecting dish types. The current features are based on the category levels of the articles in the deliveries. If for example category level 3 is selected, then all category level 3 representations are used as features. However, when developing the product taxonomy for the Frequent Itemset Model it became clear that different articles are best represented by different category levels. Hence, future work could explore the best representations per article for instance through feature learning. Possibly certain category levels can be grouped or split. Also similar to the Frequent Itemset Model, the Supervised Learning model could filter out certain irrelevant articles such as cleaning supplies. To solve the problem of an imbalanced delivery dataset (certain dish types occur more often than others), the effect of undersampling could be explored. A limitation of the current work is that no hyperparameter tuning of the algorithms in the Supervised Learning Model has been performed. Given that a more representable dataset is available, this step could improve performance.

This work has created two approaches, however, different approaches are also possible. Neither approach has made use of natural language process (NLP) techniques. However, the problem is suitable to apply NLP techniques to. Another approach would be to develop a Bayesian network classifier [5, 60]. Additionally, the problem itself can be reformed by adjusting the class labels. In conclusion, this work has made progress into solving the problem of detecting dish types in delivery and many more steps can be made.

A

Results

A.1. Supervised Learning Model

A.1.1. Preliminary Results: Sample Size

Table A.1 shows the mean and standard deviation per sample size and per algorithm over five runs. For each run random samples of the dataset are obtained. The models are trained on this dataset and evaluated using a train set of 80% and a test set of 20%. For example, sample size 8,000 is evaluated on a test set of 1,600 instances, while sample size 64,000 is evaluated on a test of 12,800 instances. Two feature sets are applied. The first feature set are the one-hot encoding of category level 2 and 3. The second feature set is the article overlap with category levels 2, 3, and 4 with settings "recipe coverage" and "per recipe". Figures A.1 and A.2 visualize the results from Table A.1.

Sample size	LR	kNN	DT	RF	XGB	ML-kNN	MLP	SVM
One-Hot Encoding (category levels 2+3)								
1,000	0.40 (0.025)	0.20 (0.046)	0.52 (0.031)	0.16 (0.013)	0.57 (0.045)	0.34 (0.033)	0.38 (0.029)	0.18 (0.013)
2,000	0.50 (0.047)	0.29 (025)	0.56 (0.035)	0.23 (0.016)	0.68 (0.034)	0.46 (0.035)	0.50 (0.045)	0.31 (0.006)
4,000	0.65 (0.027)	0.39 (0.027)	0.65 (0.019)	0.34 (0.011)	0.76 (0.015)	0.58 (0.018)	0.61 (0.034)	0.45 (0.011)
8,000	0.69 (0.028)	0.46 (0.020)	0.67 (0.007)	0.42 (0.024)	0.78 (0.010)	0.62 (0.012)	0.67 (0.024)	0.58 (0.021)
16,000	0.74 (0.008)	0.54 (0.018)	0.69 (0.014)	0.51 (0.014)	0.81 (0.009)	0.69 (0.013)	0.71 (0.009)	0.67 (0.007)
32,000	0.77 (0.007)	0.60 (0.014)	0.71 (0.007)	0.58 (0.003)	0.83 (0.007)	0.71 (0.006)	0.73 (0.011)	0.74 (0.007)
64,000	0.75 (0.004)	0.62 (0.005)	0.67 (0.006)	0.67 (0.004)	0.81 (0.004)	0.67 (0.005)	0.81 (0.006)	-
Article Overlap (category levels 2+3+4, recipe coverage, per recipe)								
1,000	0.39 (0.028)	0.32 (0.023)	0.56 (0.058)	0.33 (0.017)	0.54 (0.032)	0.35 (0.030)	0.42 (0.035)	0.23 (0.013)
2,000	0.49 (0.041)	0.37 (0.026)	0.59 (0.015)	0.45 (0.040)	0.66 (0.035)	0.40 (0.026)	0.58 (0.022)	0.35 (0.021)
4,000	0.64 (0.031)	0.49 (0.014)	0.66 (0.025)	0.57 (0.026)	0.72 (0.019)	0.53 (0.009)	0.70 (0.031)	0.49 (0.018)
8,000	0.70 (0.012)	0.55 (0.018)	0.68 (0.017)	0.65 (0.016)	0.78 (0.014)	0.61 (0.009)	0.77 (0.009)	0.60 (0.021)
16,000	0.74 (0.018)	0.60 (0.013)	0.70 (0.015)	0.70 (0.013)	0.81 (0.009)	0.65 (0.011)	0.79 (0.005)	0.69 (0.016)
32,000	0.78 (0.007)	0.65 (0.008)	0.72 (0.005)	0.75 (0.007)	0.83 (0.006)	0.70 (0.003)	0.82 (0.005)	0.76 (0.009)
64,000	0.80 (0.003)	0.69 (0.060)	0.73 (0.006)	0.78 (0.005)	0.85 (0.005)	0.73 (0.003)	0.84 (0.004)	0.80 (0.004)

Table A.1: The effect of sample size on performance. Each data point is based on a five runs and shows the mean (and standard deviation). The evaluation metric is F_1-score.

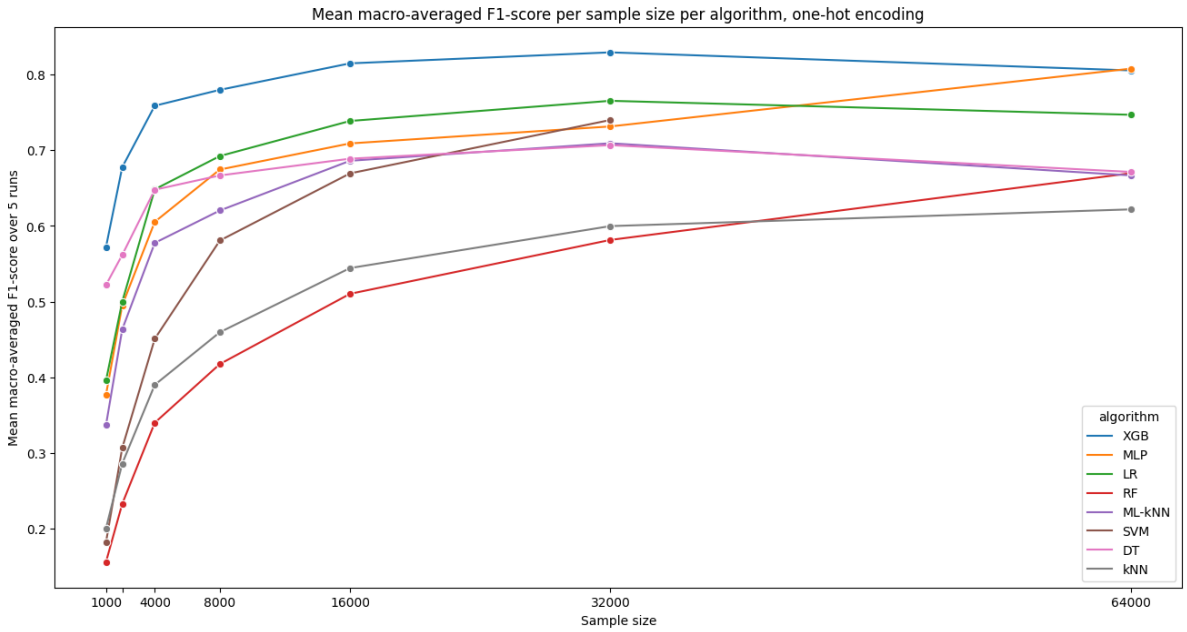

Figure A.1: The effect of sample size on performance. Feature set: one-hot encodings of category levels 2 and 3. Each data point is based on a five runs. The evaluation metric is F_1-score.

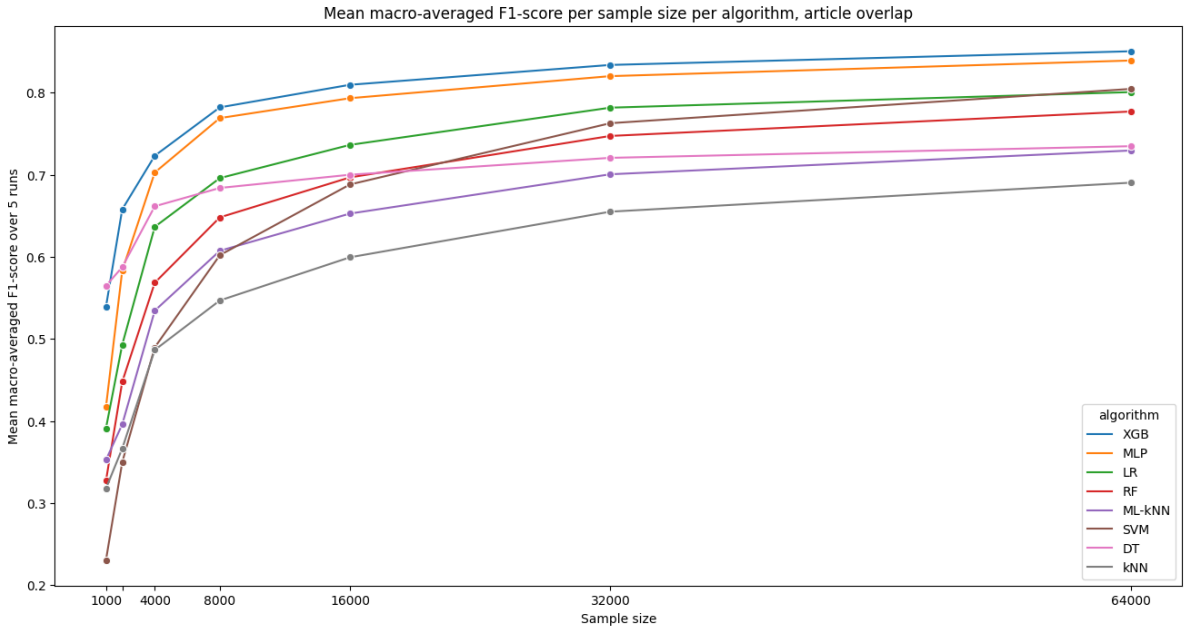

Figure A.2: The effect of sample size on performance. Feature set: article overlap with category levels 2, 3, and 4 with settings “recipe coverage” and “per recipe”. Each data point is based on a five runs. The evaluation metric is F_1-score.

A.1.2. Preliminary Results: Settings Article Overlap

	metric: Macro-F_1	**Per dish type**		**Per recipe**	
	Algorithm	**Recipe coverage**	**Jaccard index**	**Recipe coverage**	**Jaccard index**
Category level 2	LR	0.32 (0.011)	0.18 (0.005)	0.46 (0.009)	0.30 (0.009)
	kNN	0.34 (0.009)	0.36 (0.009)	0.38 (0.009)	0.38 (0.008)
	DT	0.37 (0.009)	0.33 (0.009)	0.41 (0.010)	0.36 (0.009)
	RF	0.37 (0.009)	0.31 (0.010)	0.36 (0.009)	0.32 (0.010)
	XGB	0.48 (0.010)	0.44 (0.009)	0.54 (0.010)	0.50 (0.009)
	ML-kNN	0.40 (0.009)	0.41 (0.012)	0.43 (0.011)	0.43 (0.011)
	MLP	0.44 (0.011)	0.42 (0.009)	0.55 (0.011)	0.52 (0.013)
Category level 3	LR	0.59 (0.011)	0.32 (0.007)	0.64 (0.014)	0.37 (0.007)
	kNN	0.60 (0.013)	0.65 (0.011)	0.56 (0.008)	0.59 (0.010)
	DT	0.60 (0.009)	0.56 (0.009)	0.63 (0.009)	0.59 (0.009)
	RF	0.67 (0.011)	0.62 (0.012)	0.63 (0.012)	0.56 (0.009)
	XGB	0.73 (0.009)	0.70 (0.011)	0.77 (0.010)	0.73 (0.008)
	ML-kNN	0.66 (0.008)	0.68 (0.008)	0.61 (0.010)	0.64 (0.011)
	MLP	0.70 (0.011)	0.71 (0.010)	0.75 (0.011)	0.72 (0.010)
Category level 2+3	LR	0.64 (0.010)	0.41 (0.010)	0.69 (0.008)	0.48 (0.009)
	kNN	0.58 (0.014)	0.59 (0.012)	0.54 (0.010)	0.54 (0.011)
	DT	0.60 (0.010)	0.55 (0.011)	0.64 (0.007)	0.58 (0.008)
	RF	0.65 (0.011)	0.60 (0.011)	0.58 (0.006)	0.53 (0.010)
	XGB	0.74 (0.010)	0.71 (0.013)	0.77 (0.008)	0.74 (0.011)
	ML-kNN	0.65 (0.013)	0.65 (0.09)	0.60 (0.009)	0.59 (0.012)
	MLP	0.73 (0.010)	0.73 (0.011)	0.76 (0.008)	0.75 (0.009)
Category level 4	LR	0.70 (0.008)	0.38 (0.007)	0.70 (0.009)	0.39 (0.009)
	kNN	0.71 (0.008)	0.74 (0.011)	0.65 (0.013)	0.69 (0.010)
	DT	0.69 (0.007)	0.65 (0.008)	0.71 (0.007)	0.67 (0.009)
	RF	0.76 (0.007)	0.72 (0.009)	0.73 (0.010)	0.67 (0.012)
	XGB	0.80 (0.008)	0.78 (0.008)	0.82 (0.007)	0.80 (0.009)
	ML-kNN	0.75 (0.007)	0.76 (0.009)	0.70 (0.011)	0.73 (0.009)
	MLP	0.79 (0.008)	0.79 (0.009)	80 (0.008)	0.78 (0.007)
Category level 2+3+4	LR	0.75 (0.010)	0.53 (0.008)	0.76 (0.010)	0.56 (0.009)
	kNN	0.67 (0.011)	0.68 (0.011)	0.61 (0.011)	0.61 (0.011)
	DT	0.69 (0.009)	0.64 (0.009)	0.71 (0.009)	0.66 (0.010)
	RF	0.76 (0.012)	0.72 (0.009)	0.71 (0.011)	0.65 (0.009)
	XGB	0.81 (0.007)	0.79 (0.010)	0.82 (0.005)	0.80 (0.008)
	ML-kNN	0.73 (0.011)	0.73 (0.010)	0.67 (0.009)	0.67 (0.014)
	MLP	0.80 (0.007)	0.80 (0.010)	0.80 (0.008)	0.80 (0.009)

Table A.2: Supervised Learning Method preliminary results per article overlap setting, feature set, and model. Experiments are run on a fixed dataset of 20,000 samples using 5x repeated 5-fold cross validation. The results show the mean (and standard deviation). The metric is macro averaged F_1-score.

A.1.3. General Performance

One-Hot Encoding							
	LR	**kNN**	**DT**	**RF**	**XGB**	**ML-kNN**	**MLP**
Category level 2							
EMR	0.41 (0.007)	0.35 (0.006)	0.30 (0.007)	0.33 (0.007)	0.49 (0.008)	0.41 (0.006)	0.39 (0.008)
HL	0.037 (662e-6)	0.042 (475e-6)	0.055 (926e-6)	0.039 (604e-6)	0.031 (662e-6)	0.039 (547e-6)	0.043 (693e-6)
Micro-F_1	0.63 (0.006)	0.55 (0.005)	0.55 (0.006)	0.54 (0.008)	0.70 (0.006)	0.61 (0.006)	0.62 (0.006)
Category level 3							
EMR	0.59 (0.007)	0.49 (0.010)	0.51 (0.008)	0.50 (0.008)	0.69 (0.007)	0.58 (0.010)	0.55 (0.009)
HL	0.024 (441e-6)	0.030 (682e-6)	0.031 (578e-6)	0.027 (440e-6)	0.017 (392e-6)	0.025 (674e-6)	0.028 (614e-6)
Micro-F_1	0.79 (0.004)	0.69 (0.008)	0.74 (0.004)	0.71 (0.005)	0.85 (0.004)	0.76 (0.007)	0.76 (0.005)
Category level 2+3							
EMR	0.59 (0.008)	0.50 (0.007)	0.50 (0.008)	0.49 (0.007)	0.69 (0.007)	0.57 (0.006)	0.56 (0.009)
HL	0.023 (520e-6)	0.029 (489e-6)	0.031 (539e-6)	0.027 (432e-6)	0.016 (472e-6)	0.026 (488e-6)	0.026 (630e-6)
Micro-F_1	0.79 (0.004)	0.70 (0.005)	0.74 (0.004)	0.71 (0.005)	0.85 (0.005)	0.76 (0.004)	0.77 (0.005)
Category level 4							
EMR	0.64 (0.007)	0.53 (0.013)	0.57 (0.006)	0.59 (0.007)	0.73 (0.007)	0.62 (0.007)	0.62 (0.006)
HL	0.020 (398e-6)	0.026 (476e-6)	0.025 (494e-6)	0.021 (394e-6)	0.014 (361e-6)	0.022 (517e-6)	0.022 (525e-6)
Micro-F_1	0.82 (0.005)	0.73 (0.012)	0.78 (0.006)	0.78 (0.011)	0.87 (0.008)	0.79 (0.012)	0.80 (0.006)
Category level 2+3+4							
EMR	0.63 (0.008)	0.54 (0.007)	0.55 (0.008)	0.59 (0.006)	0.73 (0.006)	0.61 (0.007)	0.62 (0.008)
HL	0.021 (472e-6)	0.026 (549e-6)	0.027 (652e-6)	0.022 (379e-6)	0.014 (392e-6)	0.023 (439e-6)	0.022 (603e-6)
Micro-F_1	0.81 (0.004)	0.73 (0.006)	0.77 (0.005)	0.78 (0.004)	0.87 (0.003)	0.79 (0.004)	0.80 (0.005)

Table A.3: Supervised Learning Model performance per metric, feature set (using one-hot encoding), and model. The experiments are run on a fixed dataset of 20,000 samples using 5x repeated 5-fold cross validation. The results show the mean (and standard deviation).

Article Overlap							
	LR	**kNN**	**DT**	**RF**	**XGB**	**ML-kNN**	**MLP**
Category level 2							
EMR	0.38 (0.008)	0.33 (0.007)	0.26 (0.006)	0.34 (0.007)	0.45 (0.006)	0.36 (0.010)	0.46 (0.008)
HL	0.038 (609e-6)	0.046 (697e-6)	0.060 (558e-6)	0.041 (488e-6)	0.034 (419e-6)	0.045 (698e-6)	0.034 (632e-6)
Micro-F_1	0.61 (0.007)	0.50 (0.008)	0.50 (0.005)	0.54 (0.007)	0.66 (0.005)	0.53 (0.009)	0.67 (0.007)
Category level 3							
EMR	0.53 (0.006)	0.54 (0.008)	0.45 (0.009)	0.53 (0.006)	0.65 (0.006)	0.56 (0.007)	0.62 (0.007)
HL	0.026 (429e-6)	0.029 (586e-6)	0.037 (766e-6)	0.027 (464e-6)	0.020 (428e-6)	0.028 (591e-6)	0.022 (420e-6)
Micro-F_1	0.74 (0.004)	0.71 (0.005)	0.69 (0.005)	0.73 (0.004)	0.82 (0.004)	0.73 (0.005)	0.80 (0.004)
Category level 2+3							
EMR	0.56 (0.007)	0.51 (0.006)	0.45 (0.008)	0.52 (0.007)	0.65 (0.005)	0.53 (0.007)	0.63 (0.006)
HL	0.024 (474e-6)	0.032 (432e-6)	0.037 (663e-6)	0.027 (467e-6)	0.019 (417e-6)	0.030 (468e-6)	0.021 (465e-6)
Micro-F_1	0.77 (0.005)	0.68 (0.004)	0.69 (0.005)	0.73 (0.005)	0.82 (0.004)	0.70 (0.004)	0.81 (0.004)
Category level 4							
EMR	0.58 (0.006)	0.64 (0.007)	0.53 (0.007)	0.62 (0.007)	0.71 (0.005)	0.65 (0.005)	0.68 (0.009)
HL	0.023 (481e-6)	0.022 (527e-6)	0.029 (591e-6)	0.021 (468e-6)	0.016 (317e-6)	0.021 (414e-6)	0.018 (520e-6)
Micro-F_1	0.77 (0.005)	0.79 (0.005)	0.75 (0.004)	0.80 (0.005)	0.86 (0.003)	0.80 (0.004)	0.83 (0.005)
Category level 2+3+4							
EMR	0.62 (0.006)	0.59 (0.006)	0.52 (0.008)	0.61 (0.008)	0.71 (0.008)	0.61 (0.007)	0.68 (0.006)
HL	0.021 (355e-6)	0.026 (420e-6)	0.030 (612e-6)	0.021 (513e-6)	0.016 (464e-6)	0.024 (471e-6)	0.018 (426e-6)
Micro-F_1	0.80 (0.003)	0.75 (0.004)	0.75 (0.004)	0.80 (0.005)	0.86 (0.004)	0.77 (0.005)	0.84 (0.004)

Table A.4: Supervised Learning Model performance per metric, feature set (using article overlap), and model. The experiments are run on a fixed dataset of 20,000 samples using 5x repeated 5-fold cross validation. The results show the mean (and standard deviation).

A.2. Estimated Performance on Unlabeled Deliveries

	Predictions Supervised Learning Model						
	LR	**kNN**	**DT**	**RF**	**XGB**	**ML-kNN**	**MLP**
	Article overlap, category level 2						
{nasi}	{salad}	{grains}	{omelet}			{grains}	
{grains, potato}		{curry}			{grains}		{grains}
{pasta, wraps}						{grains}	{wraps}
{curry, gnocchi}	{pasta}	{wraps}				{wraps}	
{pizza}	{pasta}	{pasta}		{pasta}	{pasta}	{pasta}	{pasta}
{rice}			{potato, cass.}	{pasta}			
{salad, wraps}						{grains}	
{flatbread}			{rice}			{noodles}	
{omelet, pizza, salad}		{pizza}				{pizza}	
{sushi}							
{potato, salad}	{salad}	{salad}			{salad}	{salad}	{salad}
{curry}		{pizza}	{pizza}		{pizza, omelet}	{pizza}	{pizza}
{gnocchi, noodles}	{pasta, noodles}		{salad}	{salad}	{salad}	{salad}	{salad}
{curry}		{pasta}	{noodles}		{pasta, noodles}	{pasta}	{pasta, noodles}
{quiche, wraps}		{wraps}	{wraps}			{wraps}	
{pizza}		{pasta}	{pasta, soup}				
{quiche, noodles}	{grains}		{flatbread}				
{pizza, potato}	{cass.}	{flatbread}	{curry}	{wraps}		{flatbread}	{wraps}
{hamburger, wraps}	{wraps}	{pasta}	{pasta}	{pasta}	{pasta}	{pasta}	{pasta}
	Article overlap, category levels 2+3+4						
{nasi}	{nasi}					{grains}	
{grains, potato}		{curry}	{quiche, pasta }			{pasta, curry }	{quiche}
{pasta, wraps}							
{curry, gnocchi}	{pasta}	{flatbread}				{flatbread}	
{pizza}	{pasta}	{pasta}	{pasta}	{pasta}	{pasta}	{risotto}	{pizza}
{rice}		{pasta, cass.}	{curry}			{pasta, cass.}	{quiche, cass.}
{salad, wraps}						{grains}	{flatbread}
{flatbread}			{curry}		{curry}		{curry}
{omelet, pizza, salad}		{soup}				{soup}	{soup}
{sushi}							
{potato, salad}	{salad}	{nasi}		{nasi}	{nasi}	{nasi}	{nasi}
{curry}	{flatbread}	{pasta}				{pasta}	
{gnocchi, noodles}		{salad}		{salad}		{salad}	{salad}
{curry}			{noodles}	{noodles}	{noodles}		{noodles}
{quiche, wraps}			{flatbread}				{flatbread}
{pizza}	{pasta}						{grains}
{quiche, noodles}	{quiche}						{rice}
{pizza, potato}	{cass.}						
{hamburger, wraps}		{pasta}		{pasta}	{pasta}	{pasta}	{pasta}
	One-hot encoding, category level 2						
{nasi}		{noodles}	{ pasta, nasi}			{noodles, salad}	{pasta}
{grains, potato}							{salad}
{pasta, wraps}			{noodles}				
{curry, gnocchi}			{salad, nasi}				
{pizza}							
{rice}						{potato}	{potato}
{salad, wraps}			{pasta, nasi}			{noodles}	{quiche}
{flatbread}	{quiche}	{pasta}				{stamppot}	
{omelet, pizza, salad}							{quiche}
{sushi}							
{potato, salad}		{pasta}	{noodles}			{pasta}	
{curry}			{curry}				
{gnocchi, noodles}		{potato}	{nasi}				
{curry}			{potato}				
{quiche, wraps}		{pasta}	{nasi}				
{pizza}			{salad, noodles}				
{quiche, noodles}			{pasta}				
{pizza, potato}			{noodles, wrap}				
{hamburger, wraps}							
	One-hot encoding, category levels 2+3+4						
{nasi}	{curry, rice}						{risotto}
{grains, potato}			{potato}				{potato}
{pasta, wraps}			{rice}			{noodles}	
{curry, gnocchi}							
{pizza}							
{rice}			{gnocchi}		{potato}	{noodles}	{curry, salad}
{salad, wraps}			{pasta}				{nasi, risotto}
{flatbread}			{grains}				
{omelet, pizza, salad}	{stamppot}						
{sushi}							{pasta}
{potato, salad}							
{curry}							
{gnocchi, noodles}						{noodles}	{nasi, noodles}
{curry}			{pizza}				
{quiche, wraps}	{potato}						
{pizza}			{pasta}				
{quiche, noodles}							
{pizza, potato}	{noodles}	{noodles}				{noodles}	
{hamburger, wraps}			{pasta}				

Table A.5: SLM predictions on manually labeled dataset.
(Trained on 50,000 labeled deliveries.)

	Predictions Hybrid Model						
	LR	**kNN**	**DT**	**RF**	**XGB**	**ML-kNN**	**MLP**
	Article overlap, category level 2						
{nasi}	{nasi, salad}		{curry, wraps, pasta}				
{grains, potato}			{hamburger}				{grains}
{pasta, wraps}			{wraps}	{wraps}	{wraps}	{grains}	{wraps}
{curry, gnocchi}	{pasta}	{wraps}				{wraps}	
{pizza}	{pasta}	{pasta}	{pizza}	{pizza}	{pizza, pasta}		{pasta}
{rice}			{curry, pizza}	{curry, pizza}			{pizza, potato cass.}
{salad, wraps}	{wraps}		{quiche}	{pasta}	{pasta}	{grains}	
{flatbread}			{curry}	{curry}			
{omelet, pizza, salad}							
{sushi}			{pasta}				
{potato, salad}		{nasi }	{nasi}	{nasi}	{nasi}	{nasi}	
{curry}		{pizza}	{pizza}	{pizza}	{pizza}		{pizza}
{gnocchi, noodles}	{noodles, pasta}		{curry}	{salad}	{salad}	{salad}	{salad}
{curry}		{pasta}	{curry, pasta}		{pasta, noodles}	{pasta}	{pasta, noodles}
{quiche, wraps}		{wraps}	{curry}	{curry}		{curry, wraps}	
{pizza}		{wraps}					
{quiche, noodles}			{noodles}				
{pizza, potato}	{potato cass.}	{wraps}	{wraps}	{wraps}	{wraps}		{wraps}
{hamburger, wraps}	{wraps}	{pasta}	{pizza}	{pizza}	{pasta}	{pasta, grains}	{pasta}
	Article overlap, category levels 2+3+4						
{nasi}	{nasi}	{curry}	{pasta}			{curry}	{quiche}
{grains, potato}						{pasta}	
{pasta, wraps}						{curry}	
{curry, gnocchi}	{pasta}	{grains}	{pasta}			{grains}	{wraps}
{pizza}	{pasta, pizza }		{wraps}		{wraps}	{wraps}	{wraps}
{rice}		{pasta}	{risotto, pasta}	{pasta}	{pasta}	{pasta}	{pasta}
{salad, wraps}	{wraps}	{pasta}	{pasta}		{pasta}	{risotto, pasta}	{pasta}
{flatbread}		{nasi}	{nasi}	{nasi}	{nasi}	{nasi}	{nasi}
{omelet, pizza, salad}	{pasta}		{curry, pasta}				{curry}
{sushi}		{pasta, cass.}				{pasta}	{cass.}
{potato, salad}							{wraps}
{curry}	{flatbread}	{wraps}	{wraps}			{wraps}	
{gnocchi, noodles}						{pizza}	
{curry}		{salad}	{potato, salad}	{salad}		{salad}	
{quiche, wraps}		{grains}	{flatbread, grains, wraps}			{grains}	{pasta}
{pizza}	{pasta}						
{quiche, noodles}			{noodles}			{potato}	
{pizza, potato}	{potato cass.}	{curry}				{curry}	{flatbread}
{hamburger, wraps}	{wraps}		{sushi}				{grains, sushi}
	One-hot encoding, category level 2						
{nasi}			{curry, pasta, salad}		{pasta}		{salad}
{grains, potato}							
{pasta, wraps}		{noodles}	{omelet}			{noodles}	
{curry, gnocchi}						{noodles}	{pasta}
{pizza}			{pasta, potato, omelet, noodles}				
{rice}							
{salad, wraps}			{grains}				
{flatbread}							
{omelet, pizza, salad}			{pasta}				
{sushi}			{stamppot}				
{potato, salad}						{noodles}	
{curry}					{pasta}		
{gnocchi, noodles}			{curry, grains, omelet}				
{curry}			{salad}				
{quiche, wraps}			{pasta}				
{pizza}			{omelet}				{wraps}
{quiche, noodles}			{quiche, pasta}				
{pizza, potato}					{pasta}		
{hamburger, wraps}			{omelet, pasta, quiche}		{pasta}	{wraps}	{pasta}
	One-hot encoding, category levels 2+3+4						
{nasi}	{curry}						
{grains, potato}							{curry}
{pasta, wraps}	{potato}	{noodles}					
{curry, gnocchi}	{pizza}						
{pizza}		{noodles}	{stamppot, lasagne}				
{rice}			{stamppot}				{pizza}
{salad, wraps}							
{flatbread}							{curry}
{omelet, pizza, salad}			{stamppot, pasta}				{curry, pasta }
{sushi}			{pasta}				
{potato, salad}		{noodles}	{potato, pasta, lasagne}				
{curry}	{pizza}		{potato}				
{gnocchi, noodles}	{pizza}						
{curry}	{curry}						{pizza}
{quiche, wraps}							
{pizza}			{potato}				{salad}
{quiche, noodles}			{lasagne}				{pizza}
{pizza, potato}			{paella, grain s}			{pasta}	{wraps, pizza, nasi}
{hamburger, wraps}	{curry}	{noodles}	{lasagne}				

Table A.6: HM predictions on manually labeled dataset.
(Trained on 50,000 deliveries (40,837 labeled deliveries and 9163 weakly labeled deliveries which are the FIM predictions.)

B

Labeled Picnic recipes

The tables can be found in the confidential version of the thesis.

Bibliography

[1] Rahul Agrawal et al. “Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages”. In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 13–24.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. “Mining association rules between sets of items in large databases”. In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 1993, pp. 207–216.

[3] Rakesh Agrawal et al. “Automatic subspace clustering of high dimensional data for data mining applications”. In: *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. 1998, pp. 94–105.

[4] Bilal Alataş and Erhan Akin. “An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules”. In: *Soft Computing* 10.3 (2006), pp. 230–237.

[5] Alessandro Antonucci et al. “An ensemble of bayesian networks for multilabel classification”. In: *Twenty-third international joint conference on artificial intelligence*. 2013.

[6] Hosein Azarbonyad et al. “Learning to rank for multi-label text classification: combining different sources of information”. In: *Natural Language Engineering* 27.1 (2021), pp. 89–111.

[7] Rohit Babbar and Bernhard Schölkopf. “Data scarcity, robustness and extreme multi-label classification”. In: *Machine Learning* 108.8 (2019), pp. 1329–1351.

[8] Rohit Babbar and Bernhard Schölkopf. “Dismec: Distributed sparse machines for extreme multi-label classification”. In: *Proceedings of the tenth ACM international conference on web search and data mining*. 2017, pp. 721–729.

[9] Sergey Brin et al. “Dynamic itemset counting and implication rules for market basket data”. In: *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. 1997, pp. 255–264.

[10] Patricia B Cerrito. “Choice of antibiotic in open heart surgery”. In: *Intelligent Decision Technologies* 1.1-2 (2007), pp. 63–69.

[11] Wei-Cheng Chang et al. “X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers”. In: *arXiv preprint arXiv:1905.02331* (2019).

[12] Jingjing Chen and Chong-Wah Ngo. “Deep-based ingredient recognition for cooking recipe retrieval”. In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 32–41.

[13] Qingfeng Chen and Yi-Ping Phoebe Chen. “Mining frequent patterns for AMP-activated protein kinase regulation on skeletal muscle”. In: *BMC bioinformatics* 7.1 (2006), pp. 1–14.

[14] Xiaolong Chen et al. “A Survey of Multi-label Text Classification Based on Deep Learning”. In: *International Conference on Adaptive and Intelligent Systems*. Springer. 2022, pp. 443–456.

[15] Everton Alvares Cherman, Maria Carolina Monard, and Jean Metz. “Multi-label problem transformation methods: a case study”. In: *CLEI Electronic Journal* 14.1 (2011), pp. 4–4.

[16] David W Cheung et al. “A fast distributed algorithm for mining association rules”. In: *Fourth International Conference on Parallel and Distributed Information Systems*. IEEE. 1996, pp. 31–42.

[17] Takumi Ege and Keiji Yanai. “Multi-task learning of dish detection and calorie estimation”. In: *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management*. 2018, pp. 53–58.

[18] Takumi Ege and Keiji Yanai. “Simultaneous estimation of food categories and calories with multi-task CNN”. In: *2017 fifteenth IAPR international conference on machine vision applications (MVA)*. IEEE. 2017, pp. 198–201.

[19] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[20] Adil Fahad et al. “A survey of clustering algorithms for big data: Taxonomy and empirical analysis”. In: *IEEE transactions on emerging topics in computing* 2.3 (2014), pp. 267–279.

[21] Reza Ferdousi, Ali Akbar Jamali, and Reza Safdari. “Identification and ranking of important bio-elements in drug-drug interaction by Market Basket Analysis”. In: *BioImpacts: BI* 10.2 (2020), p. 97.

[22] *Food.com - Recipes, Food Ideas And Videos*. Accessed = 2022-03-10. URL: `https://www.food.com/`.

[23] Jill Freyne and Shlomo Berkovsky. “Intelligent food planning: personalized recipe recommendation”. In: *Proceedings of the 15th international conference on Intelligent user interfaces*. 2010, pp. 321–324.

[24] Akinori Fujino, Hideki Isozaki, and Jun Suzuki. “Multi-label text categorization with model combination based on f1-score maximization”. In: *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*. 2008.

[25] Siddharth Gopal and Yiming Yang. “Multilabel classification with meta-level features”. In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 2010, pp. 315–322.

[26] Abhishek Goswami and H Liu. “Deep Dish: Deep learning for classifying food dishes”. In: *Stanford Univ* (2017).

[27] Anastasia Griva et al. “Retail business analytics: Customer visit segmentation using market basket data”. In: *Expert Systems with Applications* 100 (2018), pp. 1–16.

[28] Isuru Gunasekara and Isar Nejadgholi. “A review of standard text classification practices for multi-label toxicity identification of online content”. In: *Proceedings of the 2nd workshop on abusive language online (ALW2)*. 2018, pp. 21–25.

[29] Jiawei Han, Jian Pei, and Yiwen Yin. “Mining frequent patterns without candidate generation”. In: *ACM sigmod record* 29.2 (2000), pp. 1–12.

[30] Hebatallah A Mohamed Hassan et al. “Semantic-based tag recommendation in scientific bookmarking systems”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, pp. 465–469.

[31] Zahid Hossain et al. *Mining association rules between sets of items in large databases*. 1993.

[32] Mohammad Hossin and Md Nasir Sulaiman. “A review on evaluation metrics for data classification evaluations”. In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[33] Armand Joulin et al. “Bag of tricks for efficient text classification”. In: *arXiv preprint arXiv:1607.01759* (2016).

[34] Teuvo Kohonen. “The self-organizing map”. In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[35] Rob Law et al. “Identifying changes and trends in Hong Kong outbound tourism”. In: *Tourism Management* 32.5 (2011), pp. 1106–1114.

[36] Pedro Leote et al. “Are data-mining techniques useful for selecting ecological indicators in biodiverse regions? Bridges between market basket analysis and indicator value analysis from a case study in the neotropics”. In: *Ecological Indicators* 109 (2020), p. 105833.

[37] Roger J Lewis. “An introduction to classification and regression tree (CART) analysis”. In: *Annual meeting of the society for academic emergency medicine in San Francisco, California*. Vol. 14. Citeseer. 2000.

[38] Jingzhou Liu et al. “Deep learning for extreme multi-label text classification”. In: *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 2017, pp. 115–124.

[39] José M Luna, José Raúl Romero, and Sebastián Ventura. “Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules”. In: *Knowledge and Information Systems* 32.1 (2012), pp. 53–76.

[40] José Marıa Luna, Philippe Fournier-Viger, and Sebastián Ventura. “Frequent itemset mining: A 25 years review”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.6 (2019), e1329.

[41] José Marıa Luna et al. “Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm”. In: *Integrated Computer-Aided Engineering* 21.4 (2014), pp. 321–337.

[42] Andrew Kachites McCallum. “Multi-label text classification with a mixture model trained by EM”. In: *AAAI 99 workshop on text learning*. Citeseer. 1999.

[43] John McCarthy. “Phenomenal data mining”. In: vol. 43. 8. ACM New York, NY, USA, 2000, pp. 75–79.

[44] Vera L Miguéis, Ana S Camanho, and João Falcão e Cunha. “Customer data mining for lifestyle segmentation”. In: *Expert Systems with Applications* 39.10 (2012), pp. 9359–9366.

[45] Weiqing Min et al. “A survey on food computing”. In: *ACM Computing Surveys (CSUR)* 52.5 (2019), pp. 1–36.

[46] Weiqing Min et al. “Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration”. In: *IEEE transactions on multimedia* 19.5 (2016), pp. 1100–1113.

[47] Todd K Moon. “The expectation-maximization algorithm”. In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[48] Jinseok Nam et al. “Large-scale multi-label text classification—revisiting neural networks”. In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer. 2014, pp. 437–452.

[49] Yashoteja Prabhu and Manik Varma. “Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 263–272.

[50] Alexander Ratner et al. “Snorkel: Rapid training data creation with weak supervision”. In: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. Vol. 11. 3. NIH Public Access. 2017, p. 269.

[51] Alexander Ratner et al. “Training complex models with multi-task weak supervision”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4763–4771.

[52] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Machine learning* 85.3 (2011), pp. 333–359.

[53] M Roodpishi and R Nashtaei. “Market basket analysis in insurance industry”. In: *Management Science Letters* 5.4 (2015), pp. 393–400.

[54] Sina Sajadmanesh et al. “Kissing cuisines: Exploring worldwide culinary habits on the web”. In: *Proceedings of the 26th international conference on world wide web companion*. 2017, pp. 1013–1021.

[55] Ansaf Salleb-Aouissi, Christel Vrain, and Cyril Nortet. “QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules.” In: *IJCAI*. Vol. 7. 2007, pp. 1035–1040.

[56] Ashok Savasere, Edward Robert Omiecinski, and Shamkant B Navathe. *An efficient algorithm for mining association rules in large databases*. Tech. rep. Georgia Institute of Technology, 1995.

[57] Robert E Schapire and Yoram Singer. “BoosTexter: A boosting-based system for text categorization”. In: *Machine learning* 39.2 (2000), pp. 135–168.

[58] *Scikit-learn: Machine learning in Python, sklearn.ensemble.RandomForestClassifier*. Accessed = 2022-06-25. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

[59] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. “Wavecluster: A multi-resolution clustering approach for very large spatial databases”. In: *VLDB*. Vol. 98. 1998, pp. 428–439.

[60] L Enrique Sucar et al. “Multi-label classification with Bayesian network-based chain classifiers”. In: *Pattern Recognition Letters* 41 (2014), pp. 14–22.

[61] Jie Tao and Xing Fang. “Toward multi-label sentiment analysis: a transfer learning based approach”. In: *Journal of Big Data* 7.1 (2020), pp. 1–26.

[62] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. “Recipe recommendation using ingredient networks”. In: *Proceedings of the 4th annual ACM web science conference*. 2012, pp. 298–307.

[63] Robert Tibshirani, Guenther Walther, and Trevor Hastie. “Estimating the number of clusters in a data set via the gap statistic”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[64] Naonori Ueda and Kazumi Saito. “Parametric mixture models for multi-labeled text”. In: *Advances in neural information processing systems* 15 (2002).

[65] Ivan F Videla-Cavieres and Sebastián A Rıos. “Extending market basket analysis with graph mining techniques: A real case”. In: *Expert Systems with Applications* 41.4 (2014), pp. 1928–1936.

[66] Wei Wang, Jiong Yang, Richard Muntz, et al. “STING: A statistical information grid approach to spatial data mining”. In: *Vldb*. Vol. 97. Citeseer. 1997, pp. 186–195.

[67] Yunan Wang et al. “Mixed dish recognition through multi-label learning”. In: *Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities*. 2019, pp. 1–8.

[68] Jason Weston, Ameesh Makadia, and Hector Yee. “Label partitioning for sublinear ranking”. In: *International conference on machine learning*. PMLR. 2013, pp. 181–189.

[69] Dongkuan Xu and Yingjie Tian. “A comprehensive survey of clustering algorithms”. In: *Annals of Data Science* 2.2 (2015), pp. 165–193.

[70] Ian En-Hsu Yen et al. “Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification”. In: *International conference on machine learning*. PMLR. 2016, pp. 3069–3077.

[71] Ronghui You et al. “Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks”. In: *arXiv preprint arXiv:1811.01727* 137 (2018), pp. 138–187.

[72] Mohammed J Zaki and Karam Gouda. “Fast vertical mining using diffsets”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 326–335.

[73] Mohammed Javeed Zaki. “Scalable algorithms for association mining”. In: *IEEE transactions on knowledge and data engineering* 12.3 (2000), pp. 372–390.

[74] Mabel Mengzi Zhang. “Identifying the cuisine of a plate of food”. In: *Univ. of California at San Diego, La Jolla, CA, USA, Tech. Rep. CSE* 190 (2011).

[75] Min-Ling Zhang and Zhi-Hua Zhou. “A review on multi-label learning algorithms”. In: *IEEE transactions on knowledge and data engineering* 26.8 (2013), pp. 1819–1837.

[76] Min-Ling Zhang and Zhi-Hua Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[77] Min-Ling Zhang and Zhi-Hua Zhou. “Multilabel neural networks with applications to functional genomics and text categorization”. In: *IEEE transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351.