Explainable Survival Analysis

for Urothelial Cancer

Sukhleen Kaur

Technische Universiteit Delft

TU Delft

Explainable Survival Analysis
for Urothelial Cancer

by

Sukhleen Kaur

to obtain the degree of Master of Science
at Technische Universiteit Delft,
to be defended publicly on on Thursday the 26th of August 2021 at 15:00.

Student number:	5053307	
Project duration:	November 2020 - August 2021	
Thesis committee:	Prof. Dr. Marcel Reinders	Pattern Recognition & Bioinformatics
	Dr. Joana Gonçalves	Pattern Recognition & Bioinformatics, **supervisor**
	Dr. Thomas Höllt	Computer Graphics & Visualization

An electronic version of this thesis is available at: `?iiT,ff`2TQbBiQ`vXim/2H7iXMHf`.

Cover Image: Eric Snyder 2015 for National Cancer Institute

TU Delft

Preface

I was always interested in diagnostics and wanted to work on a project with applicability to the real world. I approached my supervisor with this in mind, and she happened to have the perfect project for me. This thesis is the result of 8 months of work that started last November. It has been quite the experience to do a project at this scale with many struggles, especially in the beginning when I could not figure out the package I was using. Nonetheless, I have learned many new things, and this project has shaped me to be a better data scientist. I am proud of the hard work I have put into this project, and I hope that as you read my thesis, you can see it too.

I would like to express my most profound appreciation to my supervisor, **Professor Joana Gonçalves**, whose immense guidance and support have made this thesis possible. I would also like to thank my daily supervisor, **Dr. Attila Csala**, for giving me critical insights and helping me throughout the whole process. I would also like to thank both my supervisors for answering all of the silly questions I had and lending me an ear when I had anything to share!

I would also like to thank my father, **Dr. Harvinder Singh**, and my mother, **Sukhpreet Kaur**. They encouraged me and believed in me throughout this project and beyond. I would not have been able to make it this far if it was not for them.

I would like to thank my partner, **Sho Cremers**, for all of his love and cheerleading! I would also like to thank my friends, **Lianne**, **Josefine**, **Sofia**, and **Zoe**, for showering me with uplifting messages and pepping me up as I got close to the finish line.

Last but not least, I want to thank **Kanya** and **Olivia**. The three of us started our thesis together and have endured the same struggle to keep positive amidst the COVID-19 pandemic. Their company has made this process a little less stressful and a lot more fun!

Sukhleen Kaur
Delft, August 2021

Explainable Survival Analysis for Urothelial Cancer

Sukhleen Kaur
Pattern Recognition & Bioinformatics
Technische Universiteit Delft
Delft, The Netherlands 2628 CD
Email: S.Kaur@student.tudelft.nl

Abstract—**Survival analysis is a statistical method used to predict when an event will occur. Machine learning survival models have been used in many cancer studies. However, machine learning models may not always be interpretable. The current lack of research for explainable survival analysis for urothelial cancer prompted this study. This study offers an insight into the generalizability and explainability of machine learning models for urothelial cancer. We also determine how we can make the models interpretable in the presence of collinearity. In this study, we compared the performance of the models; Rank Linear Support Vector Machine (SVM), Rank Kernel SVM, Coxnet, Random Survival Forest (RSF), and Gradient Boosting (Gboost). We used the Memorial Sloan Kettering (MSK) and The Cancer Genome Atlas (TCGA) datasets. We used gene expression variables and clinical variables to train our models. We evaluated these models based on the C-index. We used Permutation Feature Importance (PFI), a model-agnostic method, to explain our models and used Principal Component Analysis (PCA) to deal with collinearity. We determined that the best linear model was Rank Linear SVM (C-index = 0.58) and the best non-linear model was RSF (C-index = 0.63). Using PFI showed that some of the top-most important genes were expressed in urothelial cancer, one of them even being a prognostic marker. With PCA, we were able to deal with collinearity, and the performance using PCA was comparable to models not using it. PFI with PCA showed that processes exhibited in the top genes were prevalent in cancer.**

***Keywords*: survival analysis, urothelial, generalizability, explainability, model-agnostic, Coxnet, Rank SVM, RSF, GBoost, C-index, PFI, PCA.**

1. Introduction

Survival analysis, also known as time-to-event prediction, is a statistical method used to predict when an event will occur. It is applicable in many areas, and also prevalent in medicine where it is used to indicate when a patient with a particular disease may die or when a patient may be in remission of a particular disease [1]. With increasing machine learning trends and data availability, one would assume that standard machine learning models can be used for survival analysis. However, this is not the case since there is absence of event information in survival data, i.e., there are a lot of samples in the data that have not observed the event of interest. This may be because, for instance, the patient, while being observed, did not experience the event of interest. This is known as censoring, and it is important that we take into account censored data as they carry valuable information.

In 2020 alone, there were 19.3 million new cancer cases with 10.0 million new deaths [2]. Current methods in cancer survival diagnosis involve making predictions based on the average survival rate of a patient with a specific type of cancer [3]. This method just indicates the patient's survival, and it only involves a few factors such as age and tumor stage. Every individual is different, so why would one not look into patient-specific survival rates instead? This is where survival analysis based on machine learning comes in which has the ability to provide patient-specific survival. Many studies have made use of survival analysis with machine learning and have had good results. There have been promising results within the realm of 3 of the topmost occurring cancers; breast cancer [4], [5], [6], lung cancer [7] and prostate cancer [8].

Urothelial cancer made up 3% of the 19.3 million new cancer cases in 2020, with over 200 000 deaths. While there have been some studies that look at predicting survival [9], [10] for urothelial cancer, not many look at the gene expression data while using machine learning survival methods. Gene expression data can considerably improve our understanding of cancer [11]. Gene expression data may contain hidden information that may help predict the survival of a cancer patient and may also help provide patient-specific results [12].

A major issue with machine learning models, in general, is that some of the models are not interpretable and are complicated [13]. They are, so to say, black-box models. These models learn from the data given directly, and even the people who implement/use these models are unable to explain how it came to the prediction that it made [13]. These models have been designed to be good predictors, but unfortunately, not good explainers. Explainable models are necessary as they can help in providing insight as to what is important for the model to make predictions. This can then, for instance, help in creating target treatment for patients [14]. Now, certain machine learning methods such

as Random Forest and Linear Regression provide some sort of interpretability. But this interpretability only lies within the particular model and cannot be extended to other models. Therefore, we need a more model-agnostic approach to model explainability. Model-agnostic approaches provide model flexibility, explanation flexibility and allow us to compare multiple models [15]. In addition to to using a model-agnostic approach, it is important to think about when feature variables may be correlated to one another as this leads to problems with interpretability. Unfortunately, there is not much research for survival analysis that offers model explainability, and that also deals with model explainability with the presence of collinearity.

The shortcomings mentioned above are what prompted this study. The contributions of this study involve using machine learning for survival analysis of urothelial cancer while using mRNA gene expression data and clinical data. This study also provides an insight into a model-agnostic approach for explaining these models. We then also show how we can potentially deal with collinearity.

Therefore, the research questions we try to answer are: 1) *Which machine learning survival models perform and generalize the best when predicting survival for urothelial cancer?* 2) *Can we then explain how these models were able to predict survival?* 3) *Further, can we make the models more explainable in the presence of collinearity?*

2. Methodology

In this section, we first describe the task at hand. Then, we provide details about the dataset, pre-processing steps, the experiments, and the models and techniques that were used.

2.1. The Task

A common objective for survival analysis in clinical research is to predict the time of death. In a way, such a problem can be seen as a regression, but it cannot simply be solved using regression techniques due to censored data. As briefly mentioned in Section 1, censoring refers to the absence of event information. There are different types of censoring, but in this study, we deal with right-censoring [16] and hence, simply refer to it as censoring. Right censoring occurs when the actual survival time is more than the time the patient was observed. For instance, assume that there are 4 patients in a COVID-19 clinic and are observed for a certain amount of time, say t_{max}. Here, we are interested in when a patient might die of COVID-19. If we look at Figure 1, we see that patient 1 and 4 died before t_{max} whereas patient 2 and 3 did not. We then say that patient 2 and 3 are censored.

Survival data is then defined using a tuple: (X_i, t_i, δ_i) [17]. X_i is the feature vector for patient i, δ_i is the event indicator stating whether the death was observed ($\delta_i = 1$) or not ($\delta_i = 0$). t_i indicates the time component. If death is observed t_i is the observed time of death whereas if death is not observed then t_i is the censored time.

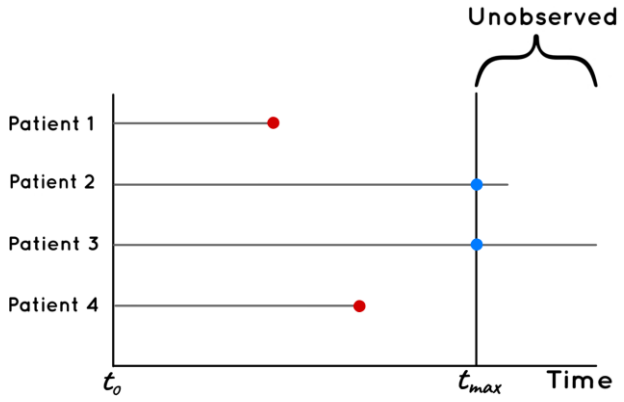


Figure 1: Right-Censoring. The red dots indicate that the patient experienced death. The blue dots indicate right-censoring since the patient did not death within the observed time t_{max}.

The objective of the machine learning survival models is to predict then a survival function that provides the risk score. The risk score is the probability of dying beyond a given time.

In this study, the models are trained on a feature vector containing gene expression variables as well as clinical variables to compute the risk score.

2.2. Dataset

The urothelial cancer dataset for this study was retrieved from cBioPortal [18], [19]. We make use of two cohorts, namely, Memorial Sloan Kettering (MSK) [20] and The Cancer Genome Atlas (TCGA) [21]. This study refers to the MSK dataset as cohort 1 and the TCGA dataset as cohort 2. In cohort 1, there are 476 patient samples, and in cohort 2 there are 413 patient samples. There are over 20000 genes measured from each patient and over 10 clinical variables in each of the cohorts. The values for the gene expression data is given as Transcripts Per Million (TPM) values. Both cohorts contain the event indicator variable as well as the time variable. A sample of the dataset can be seen in Table 1.

2.3. Pre-processing

Since there were overlapping patients between the two cohorts, the first step involved removing these overlaps. All of the overlapping patients were removed from cohort 2. This was important as having repeat observations would cause the machine learning model to give extra weight to those observations creating a biased model. This step then left us with 381 samples in total.

There were some missing hugo symbols (indicating gene symbols) which were then labeled based on their corresponding entrez gene ID number. To do this, the library

Patient ID	Survival Time	Event Indicator	$Gene_1$	...	$Gene_p$	$Clinical_1$	...	$Clinical_k$
TCGA-FD-A3B4	16.75	Deceased	452	...	11047	55	...	Female
TCGA-YF-AA3M	13.64	Living	214	...	3903	57	...	Male
TCGA-DK-AA6L	38.21	Deceased	67	...	2237	48	...	Male

TABLE 1: Dataset sample.

mygene [22] was used. This was done in order to be able to identify the features and their biological processes during feature analysis.

Next, the 2 samples with zero or unknown survival time were dealt with since the machine learning models required these values to be non-zero.

We followed current best practices for pre-processing the gene expression data, that is we applied log-transformation with a pseudo-count of 1 [23].

Then, all the genes with 0 variance were removed, as this was important for the machine learning models not to encounter zero-divide situations. Next, we also made sure to keep only the same features between the two cohorts. This step left us with around 17500 features containing both gene expression and clinical variables. The clinical variables we used for this study involved; tumor stage, age, sex, and race. We removed samples with unknown tumor stage. The tumor stage variable was then ordinal encoded while the sex and race were label encoded.

2.4. Exploratory Analysis

Due to the low number of samples between the two cohorts (only 379), we chose to create a dataset by combining the two cohorts together. However, this cannot be done simply as you may need to account for batch effect. To check whether the correction for batch effect is necessary, we use Principal Component Analysis (PCA) and check for any clusters that form based on cohort type. Figure 2 shows a plot where the gene expression features of two cohorts have been decomposed to 2-dimensions using PCA. From this figure, we can see that the two cohorts do not seem to cluster or have any separation in the 2-dimensions. Based on this, the two cohorts were combined by simply adding cohort 2 to cohort 1.

2.5. Experiments, Training and Parameter Tuning

For this study, 80% of the combined dataset is used for training and 20% of the combined dataset is used for testing. We performed more analysis where the training and testing were done on separate cohorts. These results can be seen in Supplementary Section A.1.

The training set is undersampled such that there are an equal number of samples of each class (death or no death). This is done in order to reduce bias within the machine learning models [24]. The machine learning models are 10 fold cross-validated on the training set in order to check for overfitting as well as tune hyper-parameters.

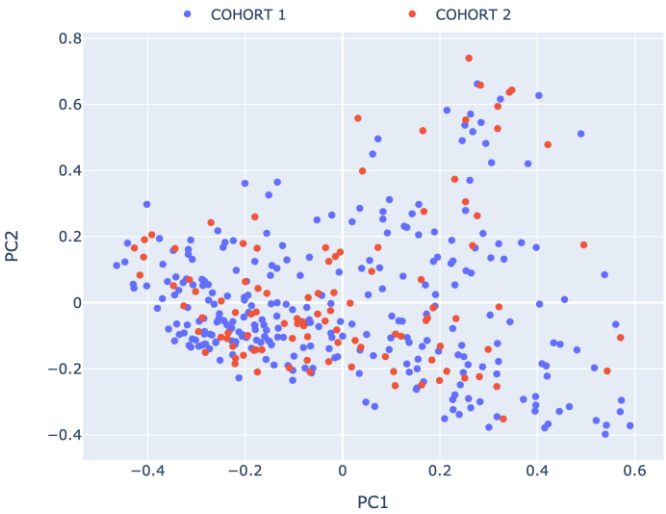


Figure 2: Plot depicting the separation of cohort 1 and cohort 2 on a 2-dimensional space. Principal component 1 (PC1) can be seen on the horizontal axis and principal component 2 (PC2) on the vertical axis.

2.6. Hyper-parameter Tuning

As mentioned earlier in Section 2.5, hyper-parameters for each of the models were tuned. To do this we made use of grid search with cross-validation. Grid search is an exhaustive search method to tune hyper-parameters. It searches over all the combinations of parameters that are given beforehand and returns the parameters that yield the best results. To perform grid search, we made use of the GridSearchCV function from the scikit-learn [25] library.

The working of grid search is illustrated with a simple example. Assume a model that has hyper-parameters A and B and a set of values where $A = \{0.1, 0.5\}$ and $B = \{10, 100\}$. Here, there are a total of 4 combinations in which grid search tries to find the optimal value. It would apply each of these 4 combinations to the model and then select the one which helps the model score the best. But, we also apply cross-validation. Given that we apply 5 fold cross-validation, the grid-search would apply all of these combinations on the 5 folds and then yield the best scoring ones. Let's say the first combination is: $A = 0.1$ and $B = 10$. Grid search would then apply this combination to all the 5 folds and average the scores yielded over the 5 folds, which would then tell us how the model performed using this combination of hyper-parameter values.

In this study, during training of each model, we determine the optimal parameters and then use the best model to test. The full range of hyper-parameters used while training the models can be seen in Figure 12 in Supplementary Section A.2.

2.7. Machine Learning Survival Analysis Models

This section summarizes the key elements of the models used in this study. The scikit-survival [26] library was used in this study for all the models. The hyper-parameter values used are mentioned in the appropriate sections for the models. The values that were set to default can be found in Table 4 in Supplementary Section A.2.

2.7.1. Cox Proportional Hazards Model (CoxPH). The Cox Proportional Hazards (CoxPH) [27] model is one of the most popular, classical models and is a semi-parametric model. The model assumes a linear relationship between the feature variables and the survival time. CoxPH tries to model the hazard function (Equation 1) which indicates the probability of an event occurring at a particular time:

$$h(t, X_i) = h_0(t) \exp(X_1\beta_1 + X_2\beta_2 + \ldots + X_n\beta_n) \quad (1)$$

where $h_0(t)$ is a baseline function which is an unspecified function giving the model it's semi-parametric property. $(X_1 + X_2 + \ldots + X_n)$ is the feature vector with $n = p + k$ and β is the coefficient vector of the features.

In order to avoid overfitting, regularization is applied. L1 and L2 penalties are two of the most popular regularization techniques. L1, also known as LASSO regression, adds a penalty that is the absolute sum of the coefficients of the feature variables. With increasing values of the coefficients, the more the penalty is applied, the cost function that the penalty is applied to decreases the value of the coefficient to reduce loss [28]. L2, also known as ridge regression, adds a penalty to the cost function that is the square of the coefficients. A coefficient to the penalty term is added as well, λ. If $\lambda > 0$, then there is more constraint on the coefficients such that the value of the coefficient will tend to zero [28].

Here, we apply the elastic net penalty [29] to the CoxPH model and we now refer to this model as Coxnet. The elastic net penalty is a linear combination of L1 and L2 penalties. The elastic net penalty is able to use the benefits of both the L1 and L2 penalties to its advantage.

In the case of a dataset containing more features than samples, the L1 penalty will not be able to choose more features than there are samples. In the case of collinearity, the L1 penalty will become unstable [29]. The net elastic penalty is then able to use the L1 term to create a sparse model and is able to use the L2 term to remove the limitation on how many features can be selected as well as introduce a grouping effect in the case of collinearity [29].

In this study, the net elasticity mixing parameter is set to `l1_ratio` is set to 0.0001. Along with this, the number of penalties on the regularization path `n_alphas` is set to 100. All the other parameters are set to their default values.

2.7.2. Ranking Support Vector Machine (Rank SVM). Support Vector Machines (SVMs) have the ability to account for complex and non-linear relationships between features and the outcome. In a classification problem, the basis of SVMs is to find a hyperplane that separates the classes the best. This hyperplane maximizes the distance between itself and the data points [30]. The SVM methods tend to perform the same as or better than the classical CoxPH model [31] for survival analysis.

In this study, we make use of a special case of SVM called Rank SVM [32]. This model converts the survival analysis problem into a ranking problem [33] where the goal is to predict the risk ranks between the patients. It considers all the possible pairs of patients in the dataset based on that, it gives a rank to the sample [33]. The shorter the survival time for a patient, the lower the rank assigned to that patient.

This study looks at two Rank SVM models; Linear SVM and Radial Basis Function (RBF) Kernel SVM. The Linear SVM assumes a linear relationship between the features and outcome. In contrast, the Kernel SVM is able to model complex, non-linear relationships. We also apply L2 regularization in order to avoid overfitting.

Here, for Rank Linear SVM, the L2 penalty, `alpha`, is set to 0.01 with a maximum iteration, `max_iter` of 10000. For Rank Kernel SVM, the L2 penalty, `alpha`, is set to 1×10^{-7}, the maximum iteration, `max_iter` is set to 1500 and the kernel coefficient, `gamma`, is set to 0.0001. All of the other parameters are set to their default values.

2.7.3. Random Survival Forest (RSF). The Random Survival Forests (RSF) [34] model is yet another classical model with slightly better performance than CoxPH [35] when applied to survival data. RSF functions similarly to the standard Random Forest model. RSF, though, has the ability to take into account censored data. RSF is an ensemble of tree-based predictors that can create un-correlated trees and, therefore, internally control error. It can do so as it builds each tree on a different dataset sample [34]. The RSF takes bootstrapped samples from the dataset and grows a tree on each of those samples. It then randomly selects some features and a split value that maximizes a particular statistic. It then repeats this step until a terminal node has been reached and then computes the Cumulative Hazard Function (CHF) [36] for each tree. It then averages it over all the samples [37].

A penalty like L1 and L2 cannot be applied to the RSF model, but in order to prevent overfitting, different parameters such as minimum samples needed to split the node are tuned. Here, the minimum samples needed to split the node, `min_samples_split` is set to 2, the minimum number of samples required to be at leaf node, `min_samples_leaf` is set to 2 and the maximum number of features used, `max_features`, is set to 1 (all features). All of the other parameters are set to their default values.

2.7.4. Gradient Boosting (GBoost). The basis of Gradient Boosting (GBoost) [38] lies in the idea that a sequence of weak base learners can become strong predictors. It com-

bines the prediction of the many base learners in an additive manner to 'boost' its predictive power [38]. Here, the base learners are decision trees, and more decision trees are added one at a time while keeping the previous trees the same. The aim is to optimize a loss function. Here the loss function to optimize is the Cox partial likelihood function, and it is optimized with gradient descent [39]. GBoost has been said to outperform existing survival models [40] making it a good candidate to compare performances.

Here, regularization is applied using dropout. The dropout rate, `dropout_rate`, is set to 0.0001 and the shrinkage of contribution of each base learner, `learning_rate`, is set to 0.0001. The tree based learners in Gboost have a parameter where the node will split only if it decreases in impurity at a particular value. This parameter, `min_imp_decrease` is set to 0. The rest of the parameters, are set to their default values.

2.8. Evaluation Metric: The C-index

To evaluate the models, we look at the C-index [41]. The C-index is a generalization of the Area Under Curve (AUC) accounting for censored data. The interpretation of the C-index is like so: the higher the value, the better the model. If the value is 0.5, then the model is said to perform at random.

The intuition behind the C-index is that a patient with a high risk score will have a shorter survival time. Given two patients i and j and that $i\,j$ [42]:

- If the survival time for i and j is known, then we know a death occurs.
    - If the risk score of $i > j$ and the survival time of $i < j$ then i and j are said to be **concordant**.
    - If the risk score of $i > j$ and the survival time of $i > j$ then i and j are said to be **discordant**.
- If the survival time for i and j are censored, we cannot assume anything about death and hence, we do not consider this pair.
- If the survival time for either i or j is censored then one death occurs. Assume, i dies and hence j is censored.
    - If the survival time for $i > j$, then it is uncertain who dies first and hence we do not consider the pair.
    - If the survival time for $i < j$, then i dies first.
        - If the risk score of $i > j$, then the pair is **concordant**.
        - If the risk score of $i < j$, then the pair is **discordant**.

The C-index is then given as:

$$\text{C-index} = \frac{\#\text{ concordant pairs}}{\#\text{ concordant pairs} + \#\text{ discordant pairs}}$$

Formally, it is written as [43]:

$$c = \frac{\sum_{i\,j} (\eta_i < \eta_j)\ (t_i > t_j)\delta_j}{\sum_{i\,j} (t_i > t_j)\delta_j} \tag{2}$$

where, η is the risk score, t is the survival time and δ is the event indicator.

2.9. Permutation Feature Importance (PFI)

Permutation feature importance [44] is a model-agnostic way of computing feature importance. It has the ability to virtually work on all models without the need of re-training them which is one of its main advantages. This method works by looking at how much the prediction error changes after permuting a feature. The algorithm works as follows [45]:

Given a trained model m, feature vector X of size n, target vector y and the evaluation metric (here C-index) $C(y, m)$

1) Compute the original prediction error $E = C(y, m(X))$
2) For each feature q:
    - Create feature matrix X_{perm} by permuting q in X.
    - Compute $E_{\text{perm}} = C(y, m(X_{\text{perm}}))$.
    - Compute importance as $E_{\text{perm}} - E$

For this study, we make use of the ELI5 [46] package. We perform permutation feature importance on two models; the best performing linear model and the best performing non-linear model. We use all the features for the permutation and perform the analysis on both the train and test set. Performing the analysis on the train set gives us an indication of which features are important in making a prediction, whereas performing the analysis on the test set gives us an indication of which features are important to generalize [45].

We also need to set the iteration number. This iteration indicates how many times the algorithm is performed. The score for each of the features is then the average over these iterations. If the number of iterations is high, it is said to provide better estimates, whereas if the iterations are low, then the algorithm would run faster [46]. Due to the large number of features and in the interest of time, the number of iterations is set to 15.

Collinearity is when feature variables are correlated with other feature variables. This can affect the interpretability of the model. If features are correlated, PFI can lead to unrealistic results. With two correlated features, when PFI permutes their value, it may lead to unrealistic values for those features. We would end up using these unrealistic observations to compute importance which then would not lead to anything meaningful if those values for the features do not occur in reality [45]. Also, adding correlated features would end up decreasing the importance of the associated features as the importance would be split amongst them [45]. Given, two highly correlated features and we train a model on them, say random forest. Some trees in random forest may find the first feature important whereas the others would

find the second feature more important [45]. This may then bring down the importance of both of these features as their importance may mean more together than when the features are used alone.

To combat this issue of collinearity, we make use of Principal Component Analysis (PCA). When PCA decomposes the features to Principal Components (PCs) it does so by finding the line that best fits the data while making the features orthogonal [30] to each other. This orthogonal property then means that the features are uncorrelated.

For this study, we decomposed all of the features (the gene expression data and the clinical variables) and chose the number of PCs based on how many PCs it took to explain 90% of the variance. We chose 71 PCs and this can be seen in Figure 13 in Supplementary Section A.3.

The dataset was then curated to have the 71 PCs as the feature matrix. The dataset was split into train (80%) and test (20%) and the best performing linear model and the best performing non-linear model from the previous analysis were trained on the training set. Like our original analysis, the models were 10-fold cross-validated on the train set and grid search was used to tune to hyper-parameters of the models. The values of the hyper-parameters can be found in Table 3 in Supplementary Section A.2. The performance was then evaluated on the test set and the performance was also compared to the performance of models based on the original analysis. Then, PFI was performed on these 71 PCs.

2.10. Gene Set Enrichment Analysis

Gene Set Enrichment Analysis (GSEA) [47] allows us to determine what kind of biological processes a set of genes (that are ranked) share based on an annotated gene set. GSEA determines how the annotated genes are represented in the ranked set of genes, whether they are over-expressed (on top of the list) or under-expressed (at the bottom of the list) [47]. The GSEA algorithm involves three main steps [47], which are described in general below:

1) Determine the enrichment score. This tells us how much the annotated genes are over or under-expressed in the ranked set of genes.
2) Estimate the statistical significance of the enrichment score by a ”phenotypic-based permutation test” [47].
3) Normalize the enrichment score for each gene set that is being analyzed.

In this study, it helps us to indicate what kind of biological process the important genes exhibit. We use the GSEAPY [48] library for this study and for the annotated gene set we use Gene Ontology (GO) Biological Processes 2018 [49], [50].

Here, we look into the loadings of the top PCs in the feature analysis. We took the intersection of the top 20 most important PCs between the train and test dataset for Linear SVM and the top 20 most important PCs between the train and test dataset for RSF. We then compute the loadings of these PCs and choose the top 20 genes to perform the GSEA.

3. Results and Discussions

3.1. Survival and Feature Analysis Using Gene Expression Data and Clinical Variables

We first look at the results on the cross-validation performances of the best models obtained using grid search for each of the models. Figure 3 shows that all the complex, non-linear models have high C-indexes in the training folds (C-index = 0.96 to C-index = 1.00) while their performances in the validation folds (C-index = 0.50 to C-index = 0.62) drop significantly. Looking at the linear models, we see that Coxnet has a much higher performance score in the training folds (C-index = 0.94) than in the validation folds (C-index = 0.58). In contrast, while the performance of the Rank Linear SVM drops in the validation folds (C-index = 0.63) from the training folds (C-index = 0.72), this drop is much less. We can therefore see that the non-linear models and the Coxnet model tend to overfit. In contrast, the Rank Linear SVM is less susceptible to overfitting, and overall, tends to generalize the best.

We now look at the performances of the models on the testing set (See Figure 4). The performances for all the models range from a C-index of 0.50 to a C-index of 0.63. The best performing model here is RSF (C-index = 0.63) and both the linear models; Coxnet (C-index = 0.59) and Rank Linear SVM (C-index = 0.58) performs similarly. Based on the results seen in Figure 3, Rank Linear SVM is the better linear model. And, while RSF tends to perform the best in the testing set, the performance in the validation folds in Figure 3 tends to be quite unstable. In contrast, the performance of Rank Linear SVM tends to be more stable.

In Section 2.5, we briefly mentioned that we performed more analysis where cohort 1 was used for training and cohort 2 was used for testing. This was to see how the models would generalize to new, unseen data. The results of this can be seen in Supplementary Section A.1. Again, we first look at the performances of the best models on the cross-validation folds. From Figure 10, we can again see that that all the non-linear models and the Coxnet model have a much higher performance score in the training folds (C-index = 0.92 to C-index = 1.00) than the validation folds (C-index = 0.50 to C-index = 0.66). Again, we see that while the performance of Rank Linear SVM drops in the validation folds (C-index = 0.61) from the training folds (C-index = 0.80), it is less compared to the other models. These results, again indicate that the non-linear models and Coxnet model tend to overfit whereas the Rank Linear SVM model is less susceptible to overfitting.

We now look at the performances of the models on the testing set in Supplementary Section A.1. There are two cases for the testing set where the models are tested on samples only from cohort 1 (Figure 11a) and then again on samples only from cohort 2 (Figure 11b). Looking at Figure 11a, we can see that the best performing model is GBoost (C-index = 0.73) and the best performing linear

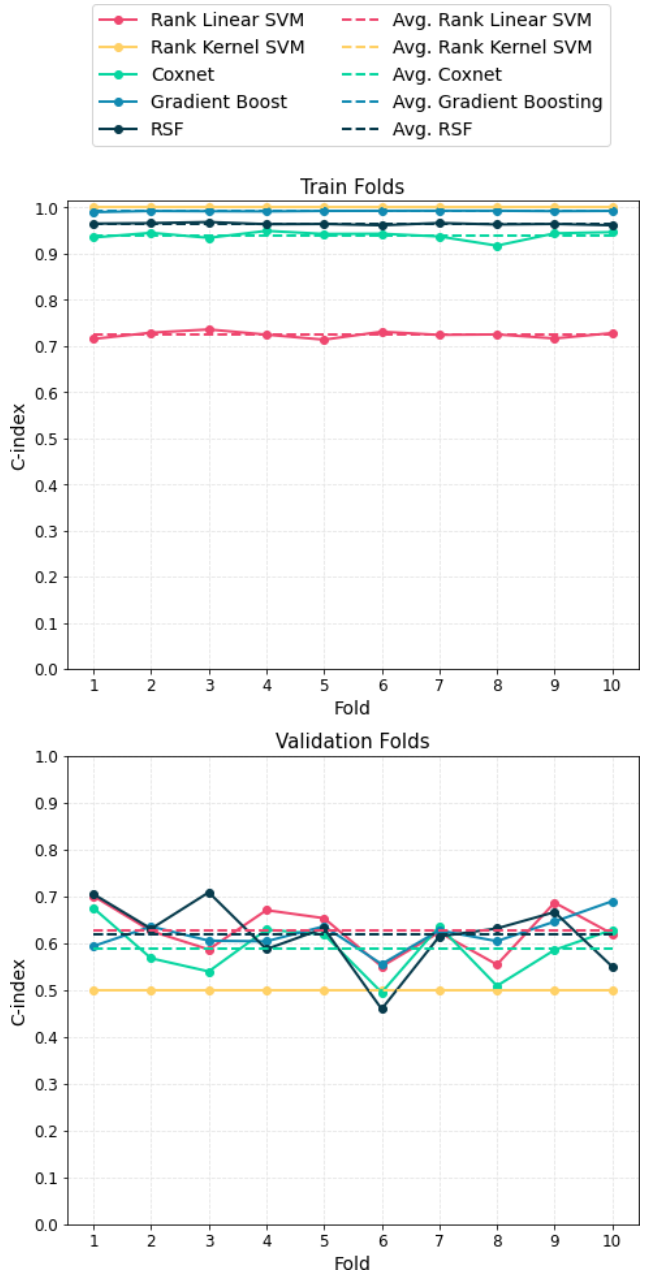


Figure 3: Performance (vertical axis) of all the models in the training set which are 10 fold cross validated (horizontal axis). Top shows the performance on the training folds and bottom shows the performance on the validation folds.

model in Rank Linear SVM (C-index = 0.68). From Figure 11b, we see that the best performing model is RSF (C-index = 0.64) and the best performing linear model is Rank Linear SVM (C-index = 0.58). These results indicate that the best non-linear model is RSF and the best linear model is Rank Linear SVM when it comes to generalizability.

Overall, we see here that all of the non-linear models and the Coxnet model tend to overfit; their performances in the

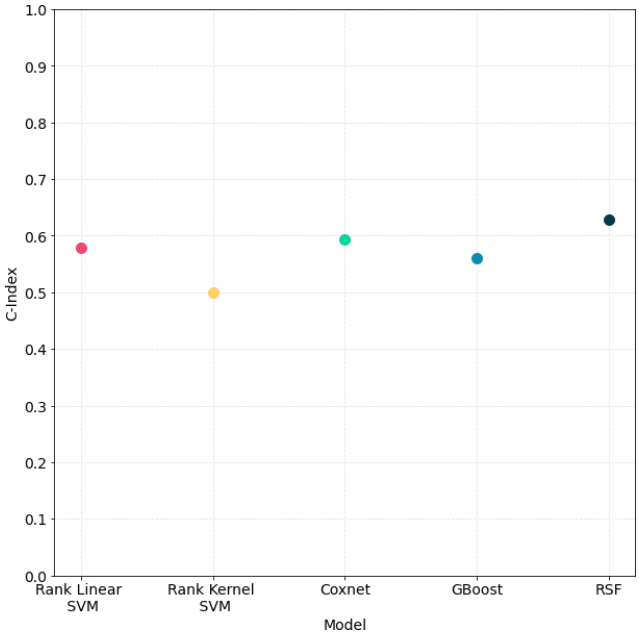


Figure 4: Performance (vertical axis) of all the models (horizontal axis) on the testing set.

training folds are much higher than in the validation folds. We see that while the models show some generalizability characteristics compared to one another, they do not tend to generalize that well overall. Generalizability depends highly on the representativeness of the dataset. If a model spends a lot of its computational effort in obtaining an accurate fit on the training set, there would be an increase in error on new data [51]. In this study, it could be the case that the dataset was not representative enough for the models to learn general patterns. Another reason could be that the models are simply too complex and cannot distinguish noise from the valuable parts of the data. Yet, another reason is that we do not have enough data. The number of samples is much less than the number of features (356 samples vs. 17500 features). However, we do see that the Rank Linear SVM model tends to perform rather similarly between the training folds and the validation folds. This can be because this model does not learn any non-linear relationships between the features and the outcome and therefore does not add any degree of complexity. It is also interesting to see that Rank Linear SVM performed comparably to even the highest performing model (RSF) in the testing set. With it being less susceptible to overfitting and its ability to perform comparably, if one were to choose just one model from this analysis, it may be the Rank Linear SVM model, especially if you consider the Occam's Razor principle.

Based on the results above, we perform further analysis with RSF and Rank Linear SVM. Here, we look at the feature importance scores of RSF and Rank Linear SVM computed using the Permutation Feature Importance (PFI) method for both the training set and the testing set. Figure 5 shows the result of the top 50 most important gene

expression features plus the clinical variables. It is sorted by the importance in the testing set. We see that the clinical variables are not deemed important and that all the features tend to have a low score in general. We can also see from the figures that the feature importance in the testing set tends to vary a lot. This is a reflection of how differently these models perform on the training set and testing set. We have seen that the models tend to have a higher performance in the training set than in the testing set. So, the features required to learn a prediction are not the same as those needed to generalize.

Looking at the results for the feature analysis in Figure 5, we also explored the top 10 genes between the training and testing set for each of the two models. We look into whether they are expressed in urothelial cancer and whether they act as a prognostic marker. To check for this, we make use of the Human Protein Atlas [52]. For the top 10 genes between the training and testing set of Rank Linear SVM, we get four genes; B2M, FN1, KRT17, KRT5. For the top 10 genes between the training and testing set of RSF, we get one gene; ASTE1. All of these genes are expressed in urothelial cancer with ASTE1 being a prognostic marker as well. These results indicate that the models were able to learn something from the given dataset and that there is some promise in modeling explainability using the PFI method.

3.2. Survival and Feature Analysis Using Decomposed Features (PCA)

In this section, we show the results of using PCA to deal with collinearity. We perform this analysis on the two models. Figure 6 shows the performance of the models; Rank Linear SVM and RSF on the training set. Here, the gene expression features and the clinical variables are decomposed using PCA. Again, the training set was 10-fold cross-validated which indicates the models' performance in each fold for both the train and validation folds. In Figure 6, we can see that the models trained on the regular features and the decomposed features perform quite similarly. We see that the performance in the training folds for RSF (C-index = 0.94) is much higher than in the validation folds (C-index = 0.62), and also that the performance in the validation folds is unstable. We also see that the performance of Rank Linear SVM, while higher in the training folds (C-index = 0.73) than in the validation folds (C-index = 0.59), the difference is still much less than RSF.

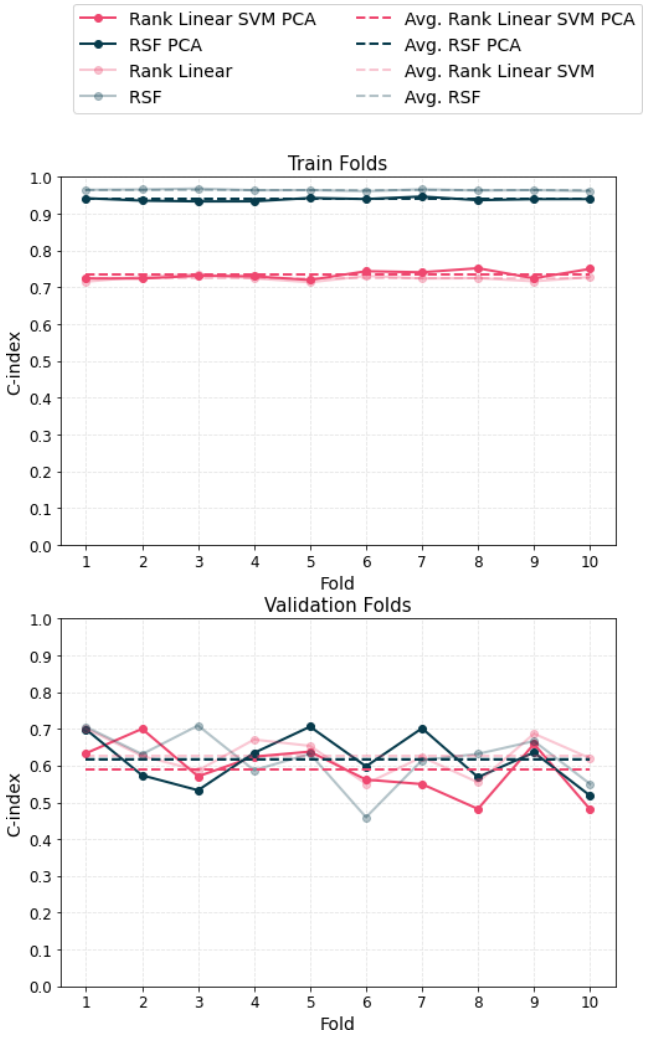
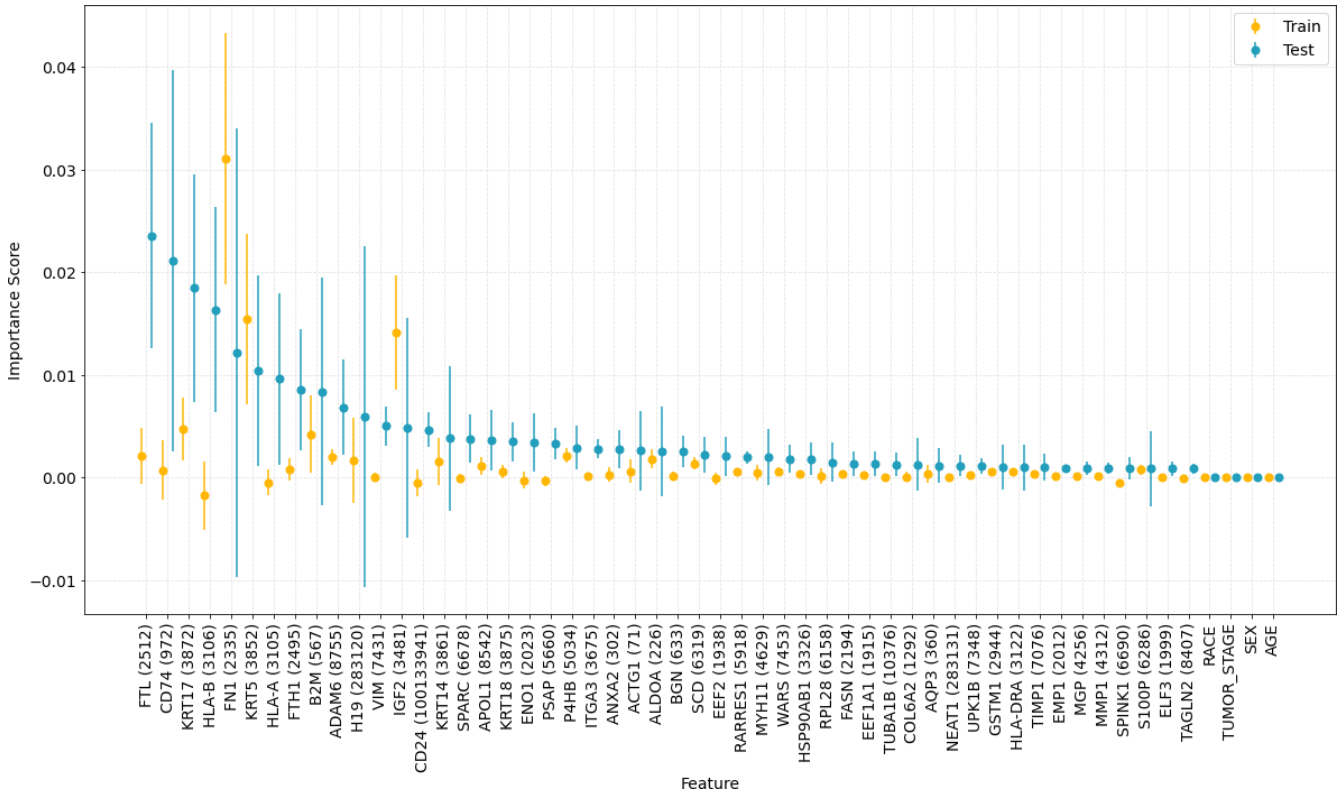
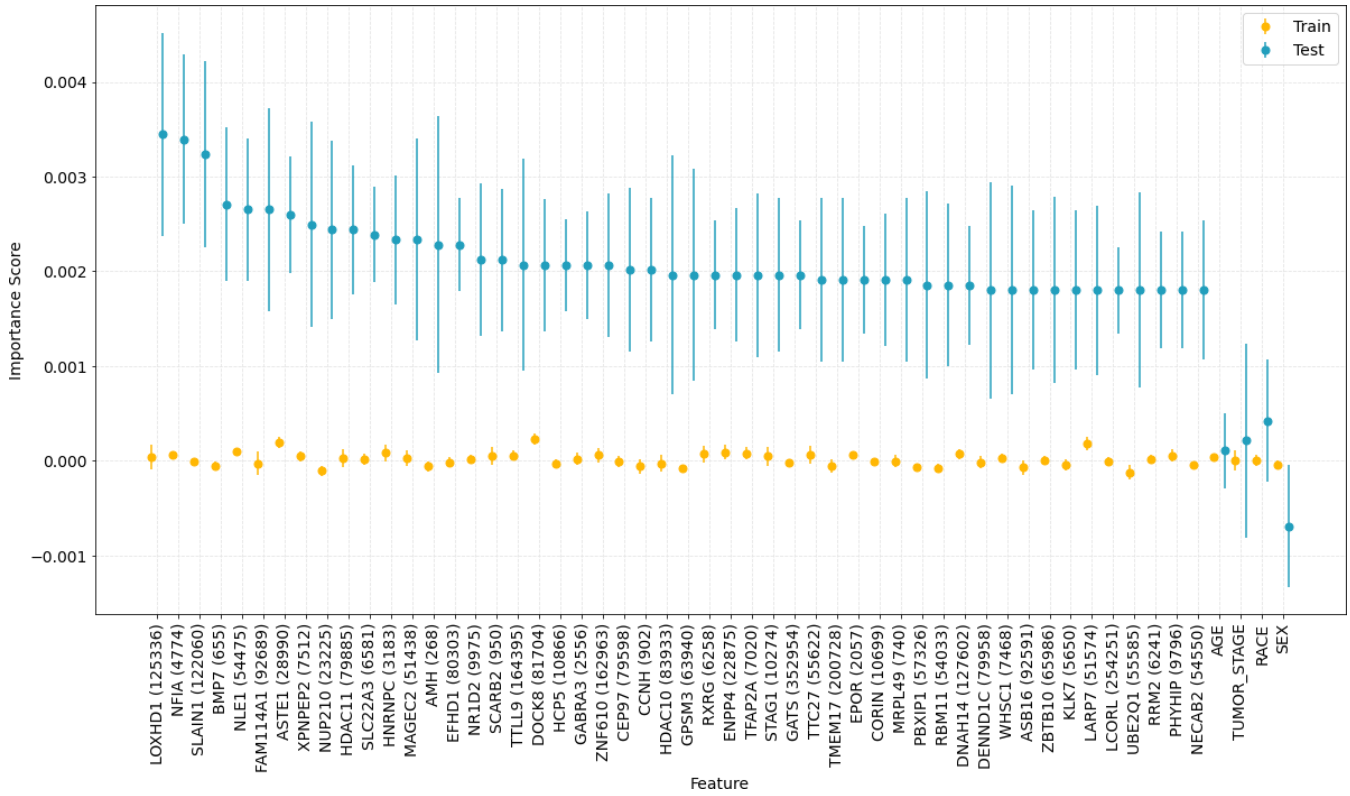


Figure 6: Performance (vertical axis) of Rank Linear SVM and RSF with and without PCA in the training set which are 10-fold cross validated (horizontal axis). Top shows the performance on the training folds and bottom shows the performance on the validation folds.

We can see the results of the models in the training set in Figure 7. Here, we can see that the performances of the PCA versions of the models are comparable to the original versions of the models. Looking closely, we see that the performance of Rank Linear SVM with PCA (C-index = 0.64) is higher compared to the performance of the original Rank Linear SVM model (C-index = 0.58). In contrast, we see that the performance of RSF with PCA (C-index = 0.50) is lower compared to the performance of the original RSF model (C-index = 0.63). The reason for this decrease may be because PCA creates a linear combination of the feature variables which would make sense for a linear model but not for RSF since it looks for complex, non-linear relationship.



(a) SVM



(b) RSF

Figure 5: Feature analysis plot for (a) SVM and (b) RSF showing the importance score (vertical axis) of the top 50 most important gene expression features plus clinical variables (horizontal axis) for both the training and testing set. They are sorted by the importance in the testing set.

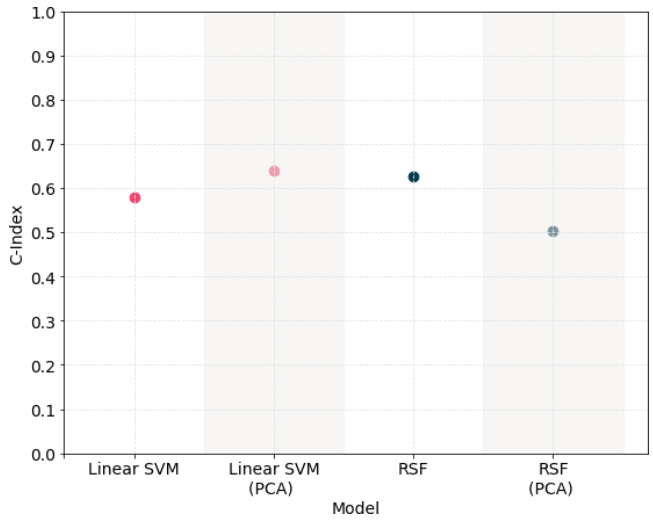


Figure 7: Performance (vertical axis) of SVM and RSF with and without using PCA (horizontal axis) on the testing set.

For further analysis, we look into the feature importances of the two models using the PCA decomposed features. We can see these results in Figure 8 which show the importance score of all the 71 principal components. Again, we notice here that there is a lot of variance between the training set and testing set. This, again, is a reflection of the differences between the performances in the training set and testing set. We also notice here that there seems to be more overlap between the principal components in the training and testing set for Rank Linear SVM than for RSF. In Table 2, we can see these overlaps. The table shows the overlap between the top 20 PCs in the training and testing set for both the models. We can see here that most of the PCs chosen tend to occur in the top 10 for Rank Linear SVM whereas for RSF they tend to be lowly ranked.

Methods	Dataset	PC1	PC4	PC10	PC13
SVM	Train	5	17	2	6
SVM	Test	4	5	2	1
RSF	Train	12	11	3	15
RSF	Test	6	2	10	1

TABLE 2: Rank of overlapping PCs from the top 20 PCs in the training and testing set for both Rank Linear SVM and RSF models.

3.3. Gene Set Enrichment Analysis

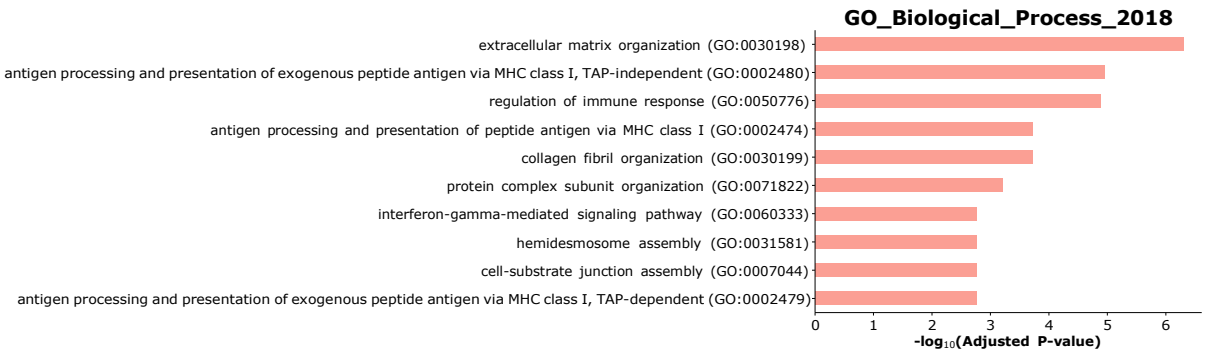
We finally look at the GSEA results. Here, we took the top 20 genes from each PC mentioned in Table 2. This was determined based on their loadings in the PC. We chose 20 genes as the threshold since the loadings tend to drop a lot after the top 20 genes. This can be seen in Figure 14 in Supplementary Section A.4. We can see the results in Figure 9 of the GSEA. From Figure 9a, it can be seen that the genes chosen from PC1 seem to exhibit immune response as well as extracellular matrix organization. Similarly, from Figure 9b, for PC4, the genes exhibit extracellular matrix organization process. From Figure 9c for PC10 and 9d for PC13, the genes exhibit immune response and cell growth process, and metabolic processes, respectively.

The processes exhibited by the genes above do have a relationship with cancer in general. The extracellular matrix organization process serves as a 'niche' [53] for cancer stem cells which then initiate tumors. It provides structural support to regulate proliferation and also differentiation of the cancer stem cells. The immune system response process is something that is prevalent whenever one is suffering from a disease. In cancer, damaged DNA cells produce tumor antigens which signal to the immune system to target the cancer cells to eliminate them [54]. Cancer occurs due to cell proliferation which is also one of the processes that seemed to be exhibited. Urothelial cancer cells need a shift to glycolysis-dependent metabolism in order to proliferate, it is the main energy source [55]. This process is also exhibited by the genes in the PCs here.

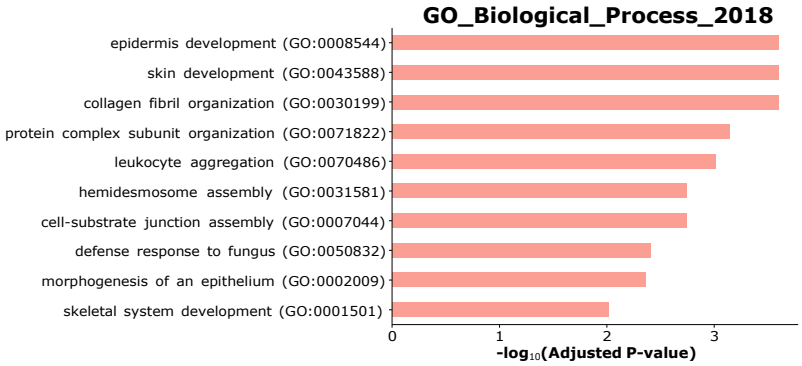
(a) SVM

(b) RSF

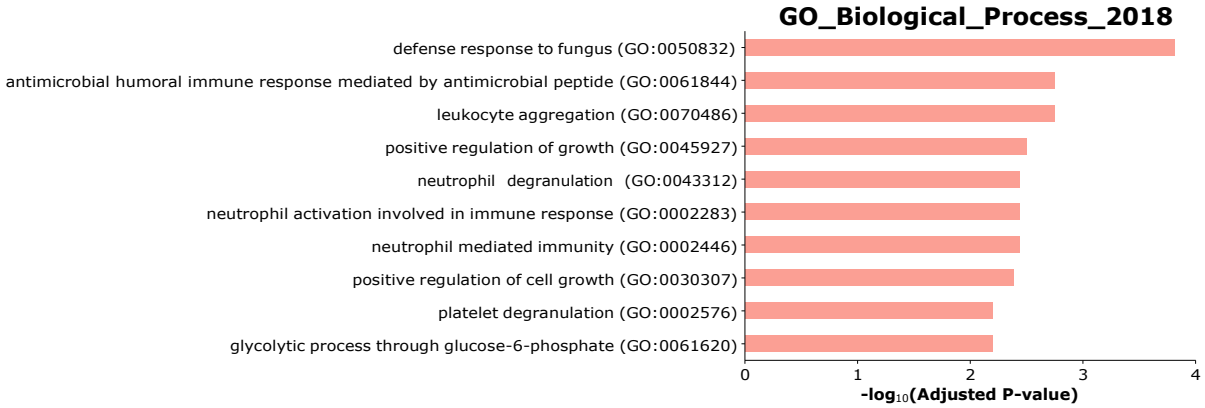
Figure 8: Feature analysis plot for (a) SVM and (b) RSF showing the importance score (y-axis) of the PCs (x-axis) for both the training (yellow) and testing set (blue). They are sorted by the importance in the testing set.



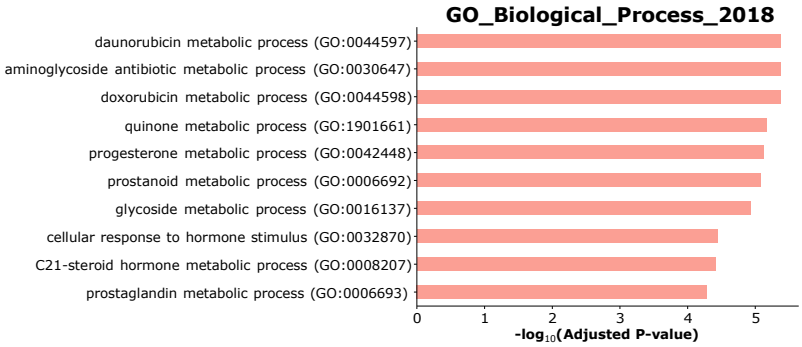
(a) PC1



(b) PC3



(c) PC9



(d) PC12

Figure 9: GSEA plots for top 20 genes in each PC.

4. Conclusion

This study aimed to determine the generalizability and explainability of machine learning survival analysis models and to determine whether they can be explained under the presence of collinearity. Overall, we can see that the models do not generalize as well but do seem to show some promise in the approach when it comes to their explainability. They are able to detect genes that are expressed in urothelial cancer, with one of them even being a prognostic marker. Through the PCA analysis, we are able to deal with collinearity, and the performance seems to be comparable to the models that do not use the decomposed features. The feature analysis with models using the PCA features also gives us an insight into which kind of processes seem to be important.

For future work, more regularization can be added. In this study, grid search was performed, so the number of parameter values that were exhausted was not that many. Hence, with more regularization, overfitting may be avoided. More data should be gathered as that would improve performance (an instance of this can be seen in Figure 15 in Section A.5) of the models. Increasing the number of iterations for permutation feature importance, may also yield more precise results. Also, other forms of model-agnostic feature importance can be used, for instance, Shapley values.

5. Tools and Packages Used

Python 3.7 and Jupyter Notebook [56] were used for data processing and running the analysis. The following libraries were used (along with their function):

- Pandas (V. 1.2.3) [57]: data processing
- NumPy (V. 1.20.1) [58]: data processing
- Scikit-Survival (V. 0.14.0)n [26]: survival modeling
- Scikit-Learn (V. 0.24.2) [25]: hyperparameter tuning, cross-validation, PCA
- Joblib (V. 1.0.1) [59]: parallelization
- Ray (V. 1.2.0) [60]: parallelization
- ELI5 (V. 0.11.0) [46]: feature analysis
- GSEAPY (V. 0.10.4) [48]: gene set enrichment analysis
- MyGen (V. 3.1.0) [22]: labeling unknown genes

References

[1] J. Emmerson and J. Brown, "Understanding survival analysis in clinical trials," *Clinical Oncology*, vol. 33, no. 1, pp. 12–14, Jan. 2021. [Online]. Available: https://doi.org/10.1016/j.clon.2020.07.014

[2] H. Sung, J. Ferlay, R. L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, and F. Bray, "Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *CA: A Cancer Journal for Clinicians*, vol. 71, no. 3, pp. 209–249, Feb. 2021. [Online]. Available: https://doi.org/10.3322/caac.21660

[3] T. Hakulinen, K. Seppä, and P. C. Lambert, "Choosing the relative survival method for cancer survival estimation," *European Journal of Cancer*, vol. 47, no. 14, pp. 2202–2210, Sep. 2011. [Online]. Available: https://doi.org/10.1016/j.ejca.2011.03.011

[4] O. S. Tătaru, M. D. Vartolomei, J. J. Rassweiler, O. Virgil, G. Lucarelli, F. Porpiglia, D. Amparore, M. Manfredi, G. Carrieri, U. Falagario, D. Terracciano, O. de Cobelli, G. M. Busetto, F. D. Giudice, and M. Ferro, "Artificial intelligence and machine learning in prostate cancer patient management—current trends and future perspectives," *Diagnostics*, vol. 11, no. 2, p. 354, Feb. 2021. [Online]. Available: https://doi.org/10.3390/diagnostics11020354

[5] A. Abadi, P. Yavari, M. Arani, H. Alavi Majd, E. Ghasemi, F. Amanpour, and C. Bajdik, "Cox models survival analysis based on breast cancer treatments," *Iranian journal of cancer prevention*, vol. 7, pp. 124–9, 03 2014.

[6] I. K. Omurlu, M. Ture, and F. Tokatli, "The comparisons of random survival forests and cox regression analysis with simulation and an application related to breast cancer," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8582–8588, May 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2008.10.023

[7] L. Cui, H. Li, W. Hui, S. Chen, L. Yang, Y. Kang, Q. Bo, and J. Feng, "A deep learning-based framework for lung cancer survival analysis with biomarker interpretation," *BMC Bioinformatics*, vol. 21, no. 1, Mar. 2020. [Online]. Available: https://doi.org/10.1186/s12859-020-3431-z

[8] O. S. Tătaru, M. D. Vartolomei, J. J. Rassweiler, O. Virgil, G. Lucarelli, F. Porpiglia, D. Amparore, M. Manfredi, G. Carrieri, U. Falagario, D. Terracciano, O. de Cobelli, G. M. Busetto, F. D. Giudice, and M. Ferro, "Artificial intelligence and machine learning in prostate cancer patient management—current trends and future perspectives," *Diagnostics*, vol. 11, no. 2, p. 354, Feb. 2021. [Online]. Available: https://doi.org/10.3390/diagnostics11020354

[9] M. Kolasa, R. Wojtyna, R. Długosz, and W. Jóźwicki, "Application of artificial neural network to predict survival time for patients with bladder cancer," *Advances in Soft Computing Computers in Medical Activity*, p. 113–122, 2009.

[10] A. Y. Abuhelwa, G. Kichenadasse, R. A. McKinnon, A. Rowland, A. M. Hopkins, and M. J. Sorich, "Machine learning for prediction of survival outcomes with immune-checkpoint inhibitors in urothelial cancer," *Cancers*, vol. 13, no. 9, p. 2001, Apr. 2021. [Online]. Available: https://doi.org/10.3390/cancers13092001

[11] W. N. van Wieringen, D. Kun, R. Hampel, and A.-L. Boulesteix, "Survival prediction using gene expression data: A review and comparison," *Computational Statistics & Data Analysis*, vol. 53, no. 5, pp. 1590–1603, Mar. 2009. [Online]. Available: https://doi.org/10.1016/j.csda.2008.05.021

[12] A. N. Burska, K. Roget, M. Blits, L. S. Gomez, F. van de Loo, L. D. Hazelwood, C. L. Verweij, A. Rowe, G. N. Goulielmos, L. G. M. van Baarsen, and F. Ponchel, "Gene expression analysis in RA: towards personalized medicine," *The Pharmacogenomics Journal*, vol. 14, no. 2, pp. 93–106, Mar. 2014. [Online]. Available: https://doi.org/10.1038/tpj.2013.48

[13] C. Rudin and J. Radin, "Why are we using black box models in AI when we don't need to? a lesson from an explainable AI competition," *1.2*, vol. 1, no. 2, Nov. 2019. [Online]. Available: https://doi.org/10.1162/99608f92.5a8a3a3d

[14] U. Pawar, D. O'Shea, S. Rea, and R. O'Reilly, "Explainable AI in healthcare," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, Jun. 2020. [Online]. Available: https://doi.org/10.1109/cybersa49311.2020.9139655

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," *ArXiv*, vol. abs/1606.05386, 2016.

[16] J. Klein, *Survival analysis : techniques for censored and truncated data*. New York: Springer, 2003.

[17] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey," vol. 51, no. 6, 2019. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3214306

[18] E. Cerami, J. Gao, U. Dogrusoz, B. E. Gross, S. O. Sumer, B. A. Aksoy, A. Jacobsen, C. J. Byrne, M. L. Heuer, E. Larsson, Y. Antipin, B. Reva, A. P. Goldberg, C. Sander, and N. Schultz, "The cBio cancer genomics portal: An open platform for exploring multidimensional cancer genomics data: Figure 1." *Cancer Discovery*, vol. 2, no. 5, pp. 401–404, May 2012. [Online]. Available: https://doi.org/10.1158/2159-8290.cd-12-0095

[19] J. Gao, B. A. Aksoy, U. Dogrusoz, G. Dresdner, B. Gross, S. O. Sumer, Y. Sun, A. Jacobsen, R. Sinha, E. Larsson, and et al., "Integrative analysis of complex cancer genomics and clinical profiles using the cbioportal," *Science Signaling*, vol. 6, no. 269, 2013.

[20] A. Kamoun, A. de Reyniès, Y. Allory, G. Sjödahl, A. G. Robertson, R. Seiler, K. A. Hoadley, C. S. Groeneveld, H. Al-Ahmadie, W. Choi, M. A. Castro, J. Fontugne, P. Eriksson, Q. Mo, J. Kardos, A. Zlotta, A. Hartmann, C. P. Dinney, J. Bellmunt, T. Powles, N. Malats, K. S. Chan, W. Y. Kim, D. J. McConkey, P. C. Black, L. Dyrskjøt, M. Höglund, S. P. Lerner, F. X. Real, F. Radvanyi, M. Aine, H. Al-Ahmadie, Y. Allory, J. Bellmunt, I. Bernard-Pierrot, P. C. Black, M. A. Castro, K. S. Chan, W. Choi, B. Czerniak, C. P. Dinney, L. Dyrskjøt, P. Eriksson, J. Fontugne, E. A. Gibb, C. S. Groeneveld, A. Hartmann, K. A. Hoadley, M. Höglund, A. Kamoun, J. Kardos, J. Kim, W. Y. Kim, D. J. Kwiatkowski, T. Lebret, S. P. Lerner, F. Liedberg, N. Malats, D. J. McConkey, Q. Mo, T. Powles, F. Radvanyi, F. X. Real, A. de Reyniès, A. G. Robertson, A. Siefker-Radtke, N. Sirab, R. Seiler, G. Sjödahl, A. Taber, J. Weinstein, and A. Zlotta, "A consensus molecular classification of muscle-invasive bladder cancer," *European Urology*, vol. 77, no. 4, pp. 420–433, Apr. 2020. [Online]. Available: https://doi.org/10.1016/j.eururo.2019.09.006

[21] A. G. Robertson, J. Kim, H. Al-Ahmadie, J. Bellmunt, G. Guo, A. D. Cherniack, T. Hinoue, P. W. Laird, K. A. Hoadley, R. Akbani, M. A. Castro, E. A. Gibb, R. S. Kanchi, D. A. Gordenin, S. A. Shukla, F. Sanchez-Vega, D. E. Hansel, B. A. Czerniak, V. E. Reuter, X. Su, B. de Sa Carvalho, V. S. Chagas, K. L. Mungall, S. Sadeghi, C. S. Pedamallu, Y. Lu, L. J. Klimczak, J. Zhang, C. Choo, A. I. Ojesina, S. Bullman, K. M. Leraas, T. M. Lichtenberg, C. J. Wu, N. Schultz, G. Getz, M. Meyerson, G. B. Mills, D. J. McConkey, J. N. Weinstein, D. J. Kwiatkowski, S. P. Lerner, R. Akbani, H. Al-Ahmadie, M. Albert, I. Alexopoulou, A. Ally, T. Antic, M. Aron, M. Balasundaram, J. Bartlett, S. B. Baylin, A. Beaver, J. Bellmunt, I. Birol, L. Boice, M. S. Bootwalla, J. Bowen, R. Bowlby, D. Brooks, B. M. Broom, W. Bshara, S. Bullman, E. Burks, F. M. Cárcano, R. Carlsen, B. S. Carvalho, A. L. Carvalho, E. P. Castle, M. A. Castro, P. Castro, J. W. Catto, V. S. Chagas, A. D. Cherniack, D. W. Chesla, C. Choo, E. Chuah, S. Chudamani, V. K. Cortessis, S. L. Cottingham, D. Crain, E. Curley, B. A. Czerniak, S. Daneshmand, J. A. Demchok, N. Dhalla, H. Djaladat, J. Eckman, S. C. Egea, J. Engel, I. Felau, M. L. Ferguson, J. Gardner, J. M. Gastier-Foster, M. Gerken, G. Getz, E. A. Gibb, C. R. Gomez-Fernandez, D. A. Gordenin, G. Guo, D. E. Hansel, J. Harr, A. Hartmann, L. M. Herbert, T. Hinoue, T. H. Ho, K. A. Hoadley, R. A. Holt, C. M. Hutter, S. J. Jones, M. Jorda, R. J. Kahnoski, R. S. Kanchi, K. Kasaian, J. Kim, L. J. Klimczak, D. J. Kwiatkowski, P. H. Lai, P. W. Laird, B. R. Lane, K. M. Leraas, S. P. Lerner, T. M. Lichtenberg, J. Liu, L. Lolla, Y. Lotan, Y. Lu, F. R. Lucchesi, Y. Ma, R. D. Machado, D. T. Maglinte, D. Mallery, M. A. Marra, S. E. Martin, M. Mayo, D. J. McConkey, A. Meraney, M. Meyerson, G. B. Mills, A. Moinzadeh, R. A. Moore, E. M. M. Pinero, S. Morris, C. Morrison, K. L. Mungall, A. J. Mungall, J. B. Myers, R. Naresh, P. H. O'Donnell, A. I. Ojesina, D. J. Parekh, J. Parfitt, J. D. Paulauskis, C. S. Pedamallu, R. J. Penny, T. Pihl, S. Porten, M. E. Quintero-Aguilo, N. C. Ramirez, W. K. Rathmell, V. E. Reuter, K. Rieger-Christ, A. G. Robertson, S. Sadeghi, C. Saller, A. Salner, F. Sanchez-Vega, G. Sandusky, C. Scapulatempo-Neto, J. E. Schein, A. K. Schuckman, N. Schultz, C. Shelton, T. Shelton, S. A. Shukla, J. Simko, P. Singh, P. Sipahimalani, N. D. Smith, H. J. Sofia, A. Sorcini, M. L. Stanton, G. D. Steinberg, R. Stoehr, X. Su, T. Sullivan, Q. Sun, A. Tam, R. Tarnuzzer, K. Tarvin, H. Taubert, N. Thiessen, L. Thorne, K. Tse, K. Tucker, D. J. V. D. Berg, K. E. van Kessel, S. Wach, Y. Wan, Z. Wang, J. N. Weinstein, D. J. Weisenberger, L. Wise, T. Wong, Y. Wu, C. J. Wu, L. Yang, L. A. Zach, J. C. Zenklusen, J. J. Zhang, J. Zhang, E. Zmuda, and E. C. Zwarthoff, "Comprehensive molecular characterization of muscle-invasive bladder cancer," *Cell*, vol. 171, no. 3, pp. 540–556.e25, Oct. 2017. [Online]. Available: https://doi.org/10.1016/j.cell.2017.09.007

[22] "mygene." [Online]. Available: https://pypi.org/project/mygene/

[23] M. D. Luecken and F. J. Theis, "Current best practices in single-cell RNA-seq analysis: a tutorial," *Molecular Systems Biology*, vol. 15, no. 6, Jun. 2019. [Online]. Available: https://doi.org/10.15252/msb.20188746

[24] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *Journal of Biomedical Informatics*, vol. 90, p. 103089, Feb. 2019. [Online]. Available: https://doi.org/10.1016/j.jbi.2018.12.003

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

[26] S. Pölsterl, "scikit-survival: A library for time-to-event analysis built on top of scikit-learn," *Journal of Machine Learning Research*, vol. 21, no. 212, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-729.html

[27] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, jan 1972.

[28] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2016.

[29] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. [Online]. Available: http://www.jstor.org/stable/3647580

[30] J. R. LESKOVEC, *MINING OF MASSIVE DATA SETS*. CAMBRIDGE UNIV PRESS, 2020.

[31] S. Goli, H. Mahjub, J. Faradmal, H. Mashayekhi, and A.-R. Soltanian, "Survival prediction and feature selection in patients with breast cancer using support vector regression," *Computational and Mathematical Methods in Medicine*, vol. 2016, pp. 1–12, 2016. [Online]. Available: https://doi.org/10.1155/2016/2157984

[32] V. Van Belle, K. Pelckmans, J. Suykens, and S. Van Huffel, "Support vector machines for survival analysis," in *Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007)*, 2007, pp. 1–8.

[33] V. V. Belle, K. Pelckmans, S. V. Huffel, and J. A. Suykens, "Support vector methods for survival analysis: a comparison between ranking and regression approaches," *Artificial Intelligence in Medicine*, vol. 53, no. 2, pp. 107–118, Oct. 2011. [Online]. Available: https://doi.org/10.1016/j.artmed.2011.06.006

[34] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, M. S. Lauer *et al.*, "Random survival forests," *The annals of applied statistics*, vol. 2, no. 3, pp. 841–860, 2008.

[35] I. K. Omurlu, M. Ture, and F. Tokatli, "The comparisons of random survival forests and cox regression analysis with simulation and an application related to breast cancer," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8582–8588, May 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2008.10.023

[36] C. Kartsonaki, "Survival analysis," *Diagnostic Histopathology*, vol. 22, no. 7, pp. 263–270, Jul. 2016. [Online]. Available: https://doi.org/10.1016/j.mpdhp.2016.06.005

[37] Sukhleen Kaur, "Survey: Latest machine learning techniques for survival analysis," 2020. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.35710.10568

[38] G. Ridgeway, "The state of boosting," 1999.

[39] ——, “The state of boosting,” 1999.

[40] M. Bai, Y. Zheng, and Y. Shen, “Gradient boosting survival tree with applications in credit scoring,” *Journal of the Operational Research Society*, pp. 1–17, Jun. 2021. [Online]. Available: https://doi.org/10.1080/01605682.2021.1919035

[41] J. Harrell, Frank E., R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, “Evaluating the Yield of Medical Tests,” *JAMA*, vol. 247, no. 18, pp. 2543–2546, 05 1982. [Online]. Available: https://doi.org/10.1001/jama.1982.03320430047030

[42] K. Tay, “What is harrell’s c-index?” Oct 2019. [Online]. Available: https://statisticaloddsandends.wordpress.com/2019/10/26/what-is-harrells-c-index/

[43] M. Schmid, M. N. Wright, and A. Ziegler, “On the use of harrell’s c for clinical risk prediction via random survival forests,” *Expert Systems with Applications*, vol. 63, pp. 450–459, nov 2016.

[44] A. J. Fisher, C. Rudin, and F. Dominici, “All models are wrong, but many are useful: Learning a variable’s importance by studying an entire class of prediction models simultaneously,” *J. Mach. Learn. Res.*, vol. 20, pp. 177:1–177:81, 2019.

[45] C. Molnar, *Interpretable machine learning: a guide for making black box models explainable*. Leanpub, 2020.

[46] “Eli5.” [Online]. Available: https://eli5.readthedocs.io/en/latest/overview.html

[47] “Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 43, pp. 15 545–15 550, 2005. [Online]. Available: http://www.jstor.org/stable/4143472

[48] “Gseapy.” [Online]. Available: https://gseapy.readthedocs.io/en/latest/introduction.html

[49] “Gene ontology: tool for the unification of biology,” *Nature Genetics*, vol. 25, no. 1, pp. 25–29, May 2000. [Online]. Available: https://doi.org/10.1038/75556

[50] and Seth Carbon, E. Douglass, B. M. Good, D. R. Unni, N. L. Harris, C. J. Mungall, S. Basu, R. L. Chisholm, R. J. Dodson, E. Hartline, P. Fey, P. D. Thomas, L.-P. Albou, D. Ebert, M. J. Kesling, H. Mi, A. Muruganujan, X. Huang, T. Mushayahama, S. A. LaBonte, D. A. Siegele, G. Antonazzo, H. Attrill, N. H. Brown, P. Garapati, S. J. Marygold, V. Trovisco, G. dos Santos, K. Falls, C. Tabone, P. Zhou, J. L. Goodman, V. B. Strelets, J. Thurmond, P. Garmiri, R. Ishtiaq, M. Rodríguez-López, M. L. Acencio, M. Kuiper, A. Lægreid, C. Logie, R. C. Lovering, B. Kramarz, S. C. C. Saverimuttu, S. M. Pinheiro, H. Gunn, R. Su, K. E. Thurlow, M. Chibucos, M. Giglio, S. Nadendla, J. Munro, R. Jackson, M. J. Duesbury, N. Del-Toro, B. H. M. Meldal, K. Paneerselvam, L. Perfetto, P. Porras, S. Orchard, A. Shrivastava, H.-Y. Chang, R. D. Finn, A. L. Mitchell, N. D. Rawlings, L. Richardson, A. Sangrador-Vegas, J. A. Blake, K. R. Christie, M. E. Dolan, H. J. Drabkin, D. P. Hill, L. Ni, D. M. Sitnikov, M. A. Harris, S. G. Oliver, K. Rutherford, V. Wood, J. Hayles, J. Bähler, E. R. Bolton, J. L. D. Pons, M. R. Dwinell, G. T. Hayman, M. L. Kaldunski, A. E. Kwitek, S. J. F. Laulederkind, C. Plasterer, M. A. Tutaj, M. Vedi, S.-J. Wang, P. D’Eustachio, L. Matthews, J. P. Balhoff, S. A. Aleksander, M. J. Alexander, J. M. Cherry, S. R. Engel, F. Gondwe, K. Karra, S. R. Miyasato, R. S. Nash, M. Simison, M. S. Skrzypek, S. Weng, E. D. Wong, M. Feuermann, P. Gaudet, A. Morgat, E. Bakker, T. Z. Berardini, L. Reiser, S. Subramaniam, E. Huala, C. N. Arighi, A. Auchincloss, K. Axelsen, G. Argoud-Puy, A. Bateman, M.-C. Blatter, E. Boutet, E. Bowler, L. Breuza, A. Bridge, R. Britto, H. Bye-A-Jee, C. C. Casas, E. Coudert, P. Denny, A. Estreicher, M. L. Famiglietti, G. Georghiou, A. Gos, N. Gruaz-Gumowski, E. Hatton-Ellis, C. Hulo, A. Ignatchenko, F. Jungo, K. Laiho, P. L. Mercier, D. Lieberherr, A. Lock, Y. Lussi, A. MacDougall, M. Magrane, M. J. Martin, P. Masson, D. A. Natale, N. Hyka-Nouspikel, S. Orchard, I. Pedruzzi, L. Pourcel, S. Poux, S. Pundir, C. Rivoire, E. Speretta, S. Sundaram, N. Tyagi, K. Warner, R. Zaru, C. H. Wu, A. D. Diehl, J. N. Chan, C. Grove, R. Y. N. Lee, H.-M. Muller, D. Raciti, K. V. Auken, P. W. Sternberg, M. Berriman, M. Paulini, K. Howe, S. Gao, A. Wright, L. Stein, D. G. Howe, S. Toro, M. Westerfield, P. Jaiswal, L. Cooper, and J. Elser, “The gene ontology resource: enriching a GOld mine,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D325–D334, Dec. 2020. [Online]. Available: https://doi.org/10.1093/nar/gkaa1113

[51] G. Paris, D. Robilliard, and C. Fonlupt, “Exploring overfitting in genetic programming,” in *Artificial Evolution*, P. Liardet, P. Collet, C. Fonlupt, E. Lutton, and M. Schoenauer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 267–277.

[52] “The human protein atlas.” [Online]. Available: https://www.proteinatlas.org/

[53] S. Nallanthighal, J. P. Heiserman, and D.-J. Cheon, “The role of the extracellular matrix in cancer stemness,” *Frontiers in Cell and Developmental Biology*, vol. 7, Jul. 2019. [Online]. Available: https://doi.org/10.3389/fcell.2019.00086

[54] C. Australia, “The immune system and cancer,” Dec 2014. [Online]. Available: https://www.edcan.org.au/edcan-learning-resources/supporting-resources/biology-of-cancer/defining-cancer/immune-system

[55] F. Massari, C. Ciccarese, M. Santoni, R. Iacovelli, R. Mazzucchelli, F. Piva, M. Scarpelli, R. Berardi, G. Tortora, A. Lopez-Beltran, L. Cheng, and R. Montironi, “Metabolic phenotype of bladder cancer,” *Cancer Treatment Reviews*, vol. 45, pp. 46–57, Apr. 2016. [Online]. Available: https://doi.org/10.1016/j.ctrv.2016.03.005

[56] “Jupyter.” [Online]. Available: https://jupyter.org/

[57] J. R. W. M. jbrockmendel; Joris Van den Bossche; Tom Augspurger; Phillip Cloud; gfyoung; Sinhrks; Adam Klein; Matthew Roeschke; Simon Hawkins; Jeff Tratner; Chang She; William Ayd; Terji Petersen; Marc Garcia; Jeremy Schendel; Andy Hayden; MomIsBestFriend; Vytautas Jancauskas; Pietro Battiston; Skipper Seabold; chris-b1; h-vetinari; Stephan Hoyer; Wouter Overmeire; alimcmaster1; Kaiqi Dong; Christopher Whelan; Mortada Mehyar, “pandas-dev/pandas: Pandas,” Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[58] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[59] [Online]. Available: https://joblib.readthedocs.io/en/latest/

[60] “Ray v1.4.1.” [Online]. Available: https://docs.ray.io/en/master/joblib.html

Appendix A.
Supplementary Material

A.1. Training on Cohort 1 and Testing on Cohort 2

In Figure 10 we can see the results of the models when they were trained on samples from cohort 1. In Figure 11 we can see the performance of the models in the testing set. Figure 11a shows the performance of the models when tested on samples from cohort 1 and Figure 11b shows the performance of the models when tested on samples from cohort 2. All the non-linear models and the Coxnet model tend to overfit. Rank Linear SVM is less susceptible to overfitting. Looking at Figure 10 and Figure 11 we can see that Rank Linear SVM and RSF tend to perform the best.

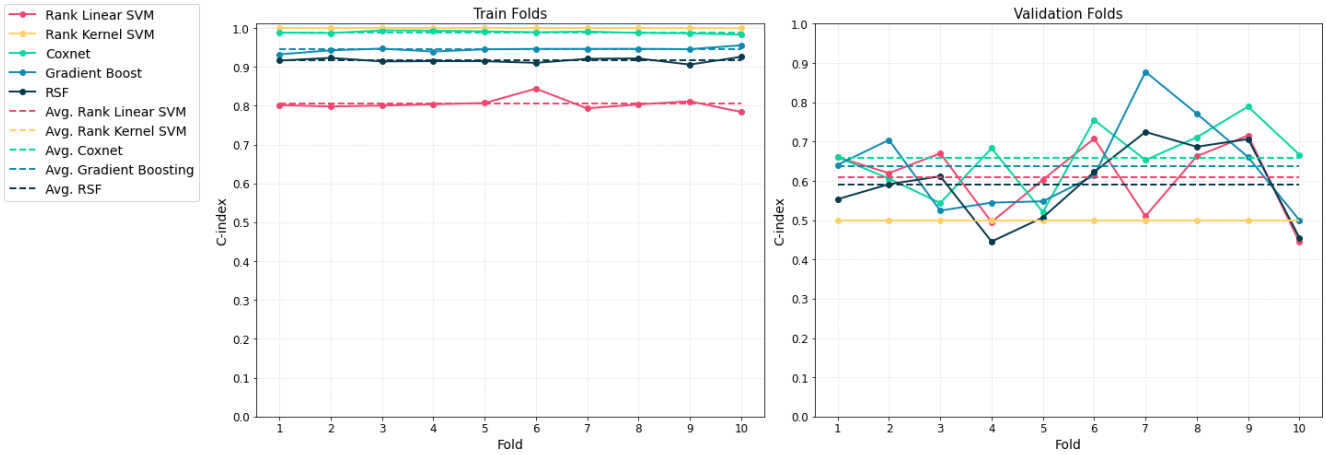
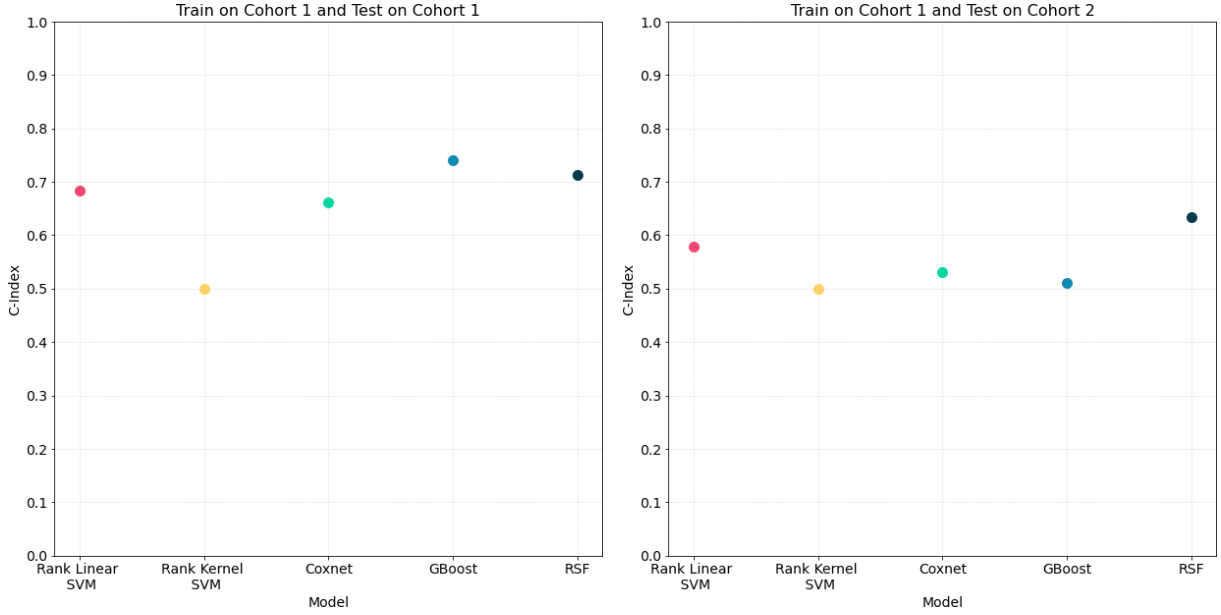


Figure 10: Performance (vertical axis) of all the models in the training set which are 10 fold cross validated (horizontal axis). Left shows the performance on the training folds and right shows the performance on the validation folds.



(a) Test scores on cohort 1. (b) Test scores on cohort 2.

Figure 11: Performance (vertical axis) of all the models (horizontal axis) on the testing set.

A.2. Hyperparameter Tuning

Model	Parameter	Explanation	Values							
Rank Linear SVM	alpha	weight of L2 penalty	0.00000001	0.0001	**0.01**	0.05	0.1	0.5	0.9	3
	max_iter	maximum number of iterations	1500	**10000**						
Rank Kernel SVM	alpha	weight of L2 penalty	**0.00000001**	0.0001	0.01	0.05	0.1	0.5	0.9	3
	max_iter	maximum number of iterations	**1500**	10000						
	gamma	kernel coefficient	**0.0001**	1	100					
Coxnet	n_alphas	number of penalties on regularization path	**100**	1000						
	l1_ratio	elastic net ratio, the smaller, the more it tends towards L2 penalty	0.00000001	**0.0001**	0.5					
RSF	max_features	number of features to consider	sqrt	**1**	log2					
	min_samples_leaf	minimum number of samples required to be at leaf node	**2**	5	15					
	min_samples_split	minimum number of samples needed to split the internal node	**2**	5	15					
GBoost	learning_rate	shrinkage of contribution of each base learner	0.00001	**0.0001**	0.5	0.1				
	dropout_rate	percentage of base learners dropped	**0.0001**	0.1	0.5					
	min_imp_decrease	node will split if it decreases impurity at this value.	**0.0**	0.5	1					

Figure 12: Table showing range of the values used for hyperparameter tuning. The bolded values are the values that were chosen as the best estimators.

For the analysis with PCA, the models were trained again using the PCA features and hence the hyper-parameters were tuned again. Table 3 shows the values of the best models. The range of parameters is the same as in Figure 12.

Model	Parameter	Value
RSF	`max_features`	log2
	`min_samples_leaf`	5
	`min_samples_split`	2
	`n_estimators`	1500
Rank Linear SVM	`alpha`	0.01
	`max_iter`	10000

TABLE 3: Hyper-parameter values of the models trained using PCA.

Model	Parameter	Default Value
Rank Linear SVM	rank_ratio	1.0
	fit_intercept	False
	optimizer	avltree
Kernel Linear SVM	rank_ratio	1.0
	fit_intercept	False
	optimizer	rbtree
Coxnet	alpha_min_ratio	auto
	normalize	False
	copy_X	True
	tol	0.00000001
	max_iter	100000
	fit_baseline_model	False
RSF	max_depth	None
	min_weight_fraction_leaf	0
	max_leaf_nodes	None
	bootstrap	True
	n_jobs	None
GBoost	loss	coxph
	n_estimatore	100
	criterion	friedman_mse
	min_samples_split	2
	min_samples_leaf	2
	min_weight_fraction_leaf	0.0
	max_depth	3
	max_features	None
	max_leaf_nodes	None
	subsample	1.0
	ccp_alpha	0.0

TABLE 4: Hyper-parameter with their default values.

A.3. Principal Component Analysis (PCA) for Multi-collinearity

Figure 13 shows the selection of the number of PCs to decompose our feature variables to. We chose 71 PCs as it is able to explain 90% of the variance.

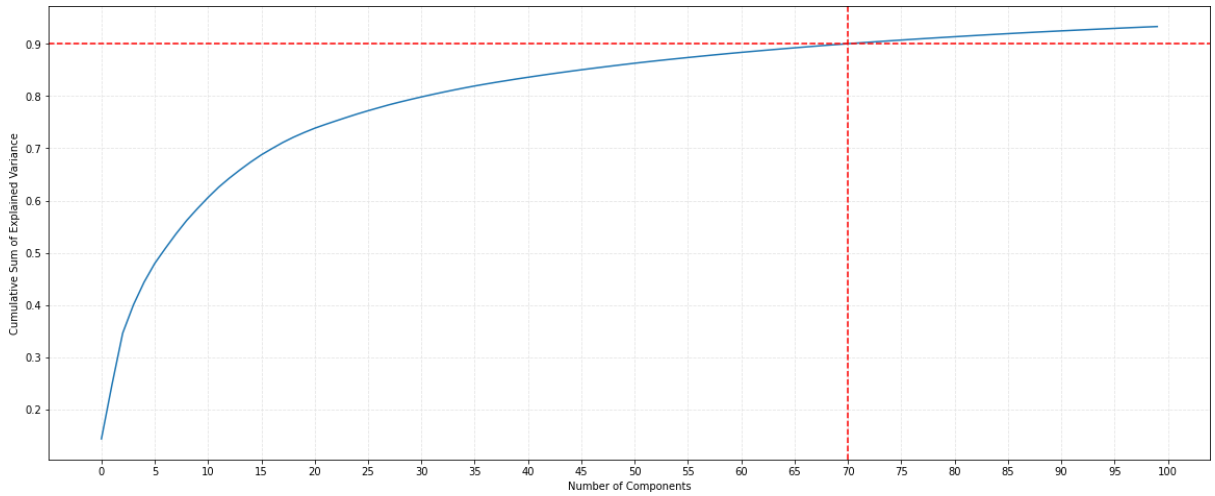


Figure 13: Cumulative sum of explained variance on the vertical axis and the number of components on the horizontal axis. The red dashed line indicate that 90% of the variance is explained using 71 PCs.

A.4. Threshold for Number of Genes to Select for GSEA

Figure 14 shows how the loadings change for each PC with increasing number of genes. There is a drop till 20 genes after which they all tend towards zero.

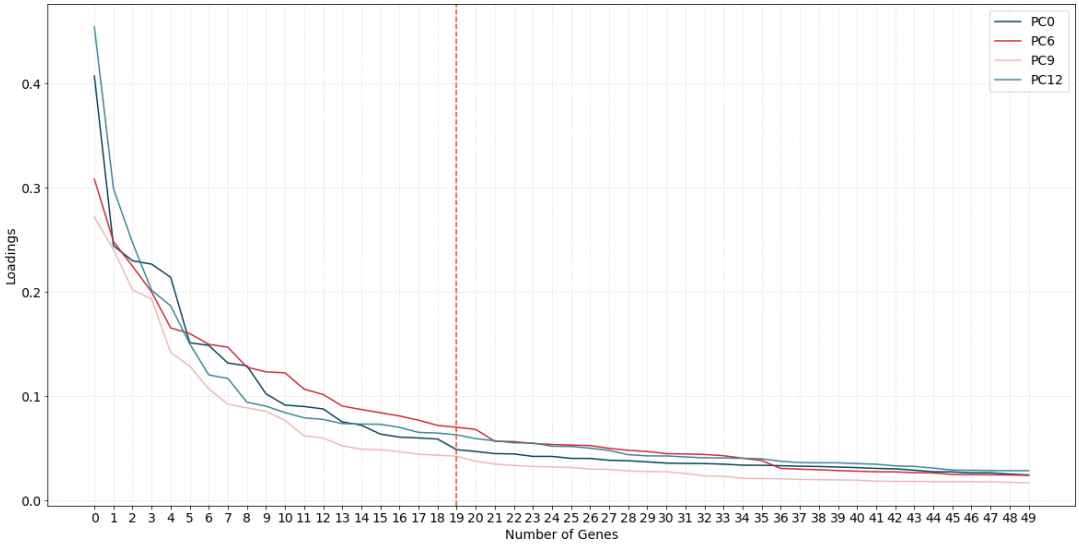


Figure 14: Loadings for each of the PCs on the vertical axis and the number of genes on horizontal axis. The red dashed line represents the number of genes to use for the GSEA analysis – 20 genes.

A.5. Effect of Adding More Samples to Training Set

Figure 15 shows the effect of adding more samples to the training data. Excluding Kernel SVM which again performs completely at random, we can see here that the mean performance increases overall (except for Coxnet) while the variance decreases for all of the models. This indicates that increasing sample size would increase overall performance. We may not the see the same effect for Coxnet model since it may be more sensitive to differences between the two cohorts.

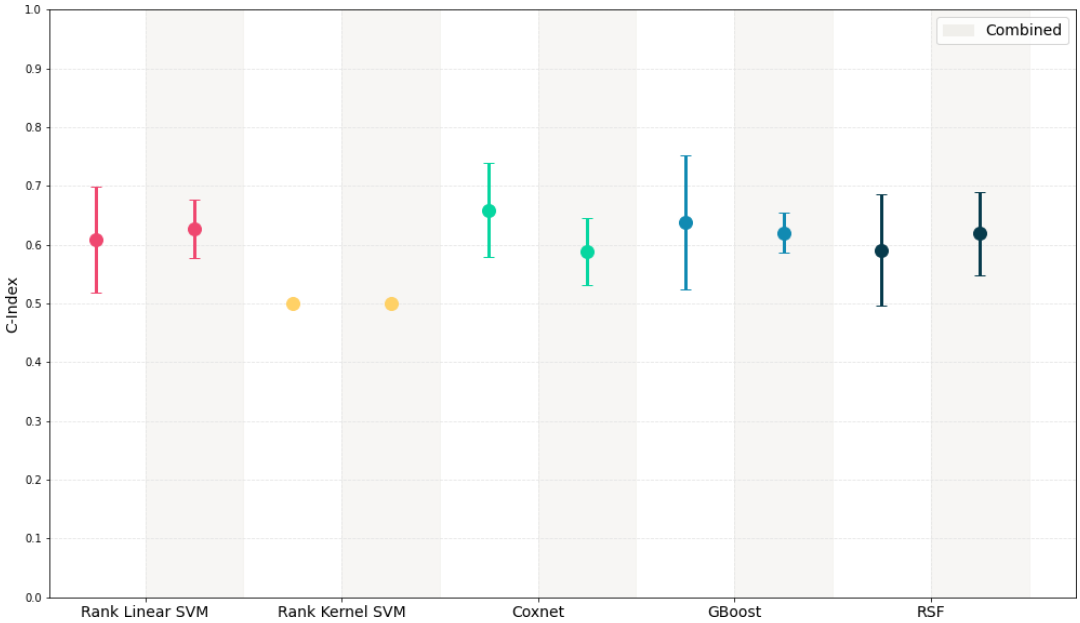


Figure 15: Performance (vertical axis) of all the models (horizontal axis) on the training set. The non-shaded regions indicate the models trained on cohort 1 and the shaded regions indicate the models trained on the combined dataset.