Feature engineering for real-world data mining challenges

Submitted May 2020, in partial completion of the degree

Bachelor of Computer Science

Man Kuan Io

School of Computing
Newcastle University

Word count: 9360

Supervised by Dr Jaume Barcadit

I declare that this dissertation represents my own work
except where otherwise stated.

Signature _____

Date _____ / _____ / _____

Acknowledgements

I would like to express my sincere gratitude for my dissertation supervisor, Dr Jaume Barcadit, for he has given me this opportunity to do this project on feature engineering, and has given me valuable advice and support throughout the entire academic year.

Abstract

Both facets of science and society produce vast amount of data that needs to be turned into useful information. Given that the unprecedented amount of data generated from biological laboratories that could potentially improve our daily lives, appropriate data modelling methods are essential. This dissertation aims to highlight the importance of the use of feature engineering in bioinformatics. It provides a thorough analysis of the performance of several well-known feature selection techniques, in search of a method that keeps or increases the predictive power by creating information-rich features that facilitate further analysis. It also assesses the quality of the selected features by comparing them against publicly accessible online disease databases.

Contents

List of Figures

List of Tables

List of Abbreviations

FE . . . . . Feature engineering

GNB . . . . . Gaussian naive bayes

KNN . . . . . K-nearest neighbours

NMF . . . . . Negative matrix factorisation

PCA . . . . . Principal component analysis

PLS-DA . . . . Partial least squares - discriminant analysis

RF . . . . . Random forest

RFE . . . . . Recursive feature elimination

SCV . . . . . Stratified cross-validation

SVC . . . . . Support vector classifier

SVM . . . . . Support vector machine

1 Introduction

Contents

In recent years, high-throughput sequencing technologies has made collecting genetic data possible; and it is estimated that by 2025, 25% of the population in developed nations and 12.5% in less-developed nations will be able to have their genomes sequenced and stored[1]. Given the unprecedented amount of data generated from biological laboratories that could potentially improve our daily lives, the ability to extract information from enormous amount of data has become ever more crucial.

High-throughput sequencing technologies have enabled scientists to have the power of modelling the whole human genome, but the said power also came with a problem: only a selected few number of genes could be analysed at a time by geneticists. Therefore, determining which gene is important is becoming an ever more important task. But because of the amount of noise in the data, and its high dimensionality, sophisticated data reduction methods are desperately needed. On the other side of the coin, DNA microarray analysis is a hot topic

for researchers today because of its ability to model the whole human genome[2]. Because of their high dimensionality and small sample size, microarray datasets pose a great challenge for computational analysis. The need of dimensionality reduction techniques was soon realised, and substantial effort was put into adapting and creating new ways to solve this problem.

Data mining approaches initially seemed to be ideally suited for the task in hand as it is a great tool for discovering trends, patterns, and relationships from the data. but the performance of the algorithm tends to drop significantly when presented with a small number of samples, or a huge number of features. Thus, a new way of data reduction is needed, which prompted the birth of bioinformatics: a discipline that combines statistics, information theory, and biology to answer questions using huge biological datasets[3]. It employs methods that performs well on storing and analysing high dimensionality datasets. This allows us to filter an enormous amount of redundant or irrelevant features efficiently and effectively, to pinpoint specific genes that could then be further studied upon by geneticists.

1.1 Motivation

Traditionally, it falls upon geneticists to identify and create useful features from the existing data. But when presented with a huge number of genes, manual methods are costly and time-consuming, if not downright impractical. Therefore, it is now a common practice to use Feature Engineering to automate the process of reducing the number of features. Feature Engineering can then be further categorized into feature selection and feature transformation[4]. Since their usage and performance are entirely based on context, finding the optimal feature engineering technique might not be trivial.

Therefore, we will compare and contrast a selection of the most successful feature engineering methods, in search of a method that keeps or increases the predictive power by creating information-rich features that facilitate the machine learning process. Afterwards, said method also has to be easily interpretable, which means the output generated still has to show a clear relation to the original dataset.

Following the said rule will ensure that the reduced dataset is still interpretable. So that it can be further analysed by domain experts to extract knowledge and insights that could potentially solve a problem that presents in the field of biology.

1.2 Contribution

Our goal is to find the most optimal combination of feature engineering technique and classifier suited for a classification task in a biomedical context. The quality of the reduced datasets is then ranked by how informative, quantitative, and representative it is compared to the original dataset. The aims of this dissertation are as follows:

1. Explore and identify feature engineering techniques that are effective at processing high-throughput omics data.

2. A thorough analysis of feature engineering techniques and classifiers in terms of disease classification conducted.

3. A comparison of how effective the feature engineering techniques and classifiers are on retrieving biological insights.

1.3 Outline

- **Chapter 2** provides the background of four feature engineering techniques and four classifiers which are used in all our experiments conducted in this research.

- **Chapter 3** evaluates the performance of different combinations of feature engineering technique and classifier in terms of its classification ranking metrics.

- **Chapter 4** evaluates the results obtained from Chapter 3 and compares it against online biomedical disease databases to find the correlation of the knowledge extracted by each combination of feature engineering technique and classifier against experimental results.

- **Chapter 5** presents conclusions and discusses future work to be done.

2
Background

Contents

2.1 Feature Engineering

Feature engineering is the process of altering a given data in order to increase its predictive performance with respect to the task that is to be performed. If done correctly, it could potentially reduce the algorithms computational cost, improve the trained model's performance, and allow one to gain insights into the dataset[5]. Manual feature engineering can be a very tedious process - It takes a considerable amount of effort to fill in missing values and extract information from incompatible data columns. Even after a dataset is pre-processed and cleaned, it may still be slow to run the model on the entire dataset. Traditionally, it falls upon domain experts to identify and create useful features from the existing data. Each of these tasks is difficult to automate and often become bottlenecks in machine learning

projects. Therefore, automated feature engineering methods are being used by machine learning practitioners to shorten the time needed used on preprocessing. We hand-picked four of the more popular techniques that uses different approaches to tackle this problem: Principal Component Analysis, Partial Least Squares, Non-negative Matrix Factorisation, and Recursive Feature Elimination. Further explanations and definitions of the techniques will be provided in Section 2.2.

2.1.1 Feature Transformation

Feature Transformation is a process that creates a new set of features by reformatting, combing and extracting information from the raw dataset. It can be further categorized into feature construction and feature extraction. Feature construction discovers missing relationships among features by creating additional features.[6] For example, a new feature could be created by combing two features from the raw dataset by using local operators, which replaces or complements the original features. In the context of feature reduction, a multi-dimensional problem could have its dimensions reduced after a relationship is discovered. Feature extraction creates new attributes from the original features through functional mappings to reduce the number of features. The new set of attributes created is a concise representation of the original features that is still able to completely describe the original dataset.

2.1.2 Feature Selection

Contrary to Feature Transformation, feature selection techniques do not alter the dataset's original representation, but merely select a subset of features that are the most informative to the prediction task on hand. Therefore, feature selection is helpful for us to understand the relationship of features and reveal insights that are buried in a high-dimensionality dataset[6, 7]. In theory, it can be shown that in order to acquire the best possible subset for our task, an exhaustive search on all possible subsets are needed. But the said method is impractical on datasets with a huge number of features. Therefore, for real-life applications, iterative methods

are usually used to reduce the computational complexity on the task by sacrificing the predictive performance of the subset.

Feature selection can be further categorized into feature ranking and subset selection. Feature ranking ranks the feature by a performance metric, then decides to keep or eliminate a feature based on its score on the metric. Differing from feature ranking, subset selection ranks the performance on each subset instead of each feature. Subset selection it can be further broken down into three categories – filter, wrapper, and embedded methods. Filter methods aim to create a subset using various statistical methods to evaluate the importance of a feature. It is able to remove features that are irrelevant or are highly correlated with a very low computational cost. Wrapper methods evaluates all possible subsets in the search space by using machine learning algorithms. Each subset is evaluated through cross-validation based on its performance on performing a certain task. Wrapper methods captures feature interactions and find the optimal subset for the task on hand, but is prone to over-fitting and is not efficient on high dimensionality datasets. Embedded methods perform subset selection while training the data on a machine learning algorithm. Like wrapper methods, it uses cross-validation to compare the subsets and are able to capture feature interactions, but it is able to do that at a much lower cost.

2.2 Feature Engineering Techniques

Principal Component Analysis

Principal Component Analysis (PCA) is one of the most used feature extraction techniques in computational biology. It is often used to reduced and explore high-dimensionality datasets in areas such as mass spectrometry and microarray analysis.[8] It achieves dimensionality reduction by transforming the variables into a new set of variables, which are known as the principal components. Principal components are linear combinations of the original features, they are ordered in a way such that the amount of variation held each component decreases as we move down in the order.

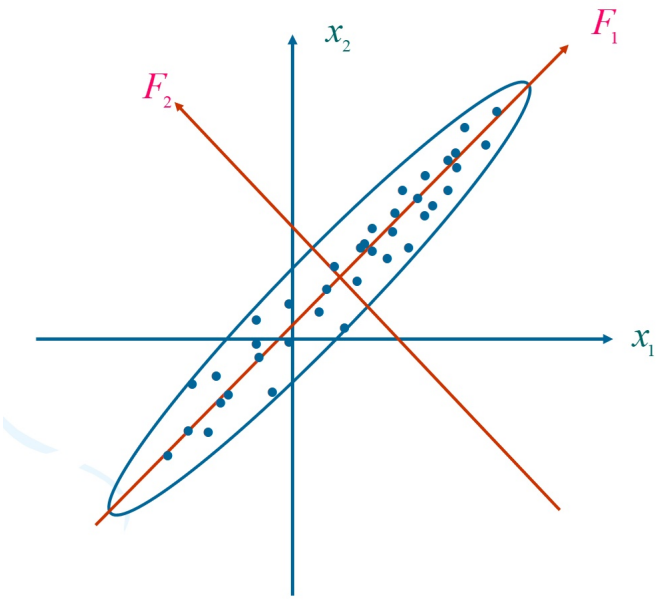

Figure 2.1: An illustration of the PCA method[9].

PCA is usually implemented using eigenvalue decomposition or singular value decomposition (SVD) using projection, the latter is more versatile than the former since its eigenvalue decomposition only works on square matrices. One thing to note is that the dataset on which the PCA technique is to be used must be scaled as the results are sensitive to scaling. An implementation of the PCA algorithm is shown below[10]:

Algorithm 1 Principal Component Analysis

Input: $m \times n$ matrix A, size k
Output: $m \times k$ matrix B

1: R = Remove the average of feature from A
2: Covariance C = $R^T R$
3: E = Eigenvector Decomposition of C
4: Projection P = AE
5: B = First k principal components in P

Partial Least-Squares Discriminant Analysis

Partial least squares (PLS) is a feature transformation technique that specialises in dealing with multicollinear small sample datasets. It was first introduced for regression tasks and then evolved into a classification method that is well known as PLS-discriminant analysis (PLS-DA). It has been frequently used for predictive and descriptive modelling, as well as discriminative variable selection, since it can be used to understand which variables carry the class separating information.

PLS-DA is performed to sharpen the separation between groups of observations by rotating PCA components such that a maximum separation among classes is obtained.[11] It can be thought of as supervised PCA since that it achieves dimensionality reduction by keeping the class labels in mind. It is even considered to be better than PCA since as parameters will not change when new samples are added to the population (Falk & Miller, 1992, Geladi & Kowalski, 1986). The pseudocode of the PLS algorithm is shown below[12]:

Algorithm 2 Partial Least Squares

Input: $m \times n$ matrix A, size k
Output: $m \times k$ matrix B

1: K = Centred Gram Matrix of A
2: K_{res} = first k score vectors in K
3: $t_i = K_{res} Y$
4: $\|t_i\| = 1$
5: $u_i = Y(Y^T t_i)$
6: $Y = Y - t_i(t_i^T Y)$
7: $B = K_{res} - t_i(t_i^T K_{res})$

Non-negative Matrix Factorisation

Non-negative Matrix Factorisation is a powerful feature extraction tool that has been popular in areas such as bioinformatics, molecular pattern discovery and chemometrics.[13] The method is related to PCA, except that it employs the constraint of non-negativity instead of orthogonality, and the results obtained using NMF are strictly positive. NMF tries to extract localised latent information that corresponds to a part of a generalised feature of the original data; As a result, NMF solutions are less uniquely defined but are more interpretable.

NMF is an algorithm that tries to obtain a linear approximation that represents the original data in two parts. Given a n x m matrix X, the algorithm aims to create two non-negative matrices (W, H) that when multiplied, approximates to original matrix X[14]:

$$A_{ij} \simeq (WH)_{ij} = \sum_{k=1}^{r} W_{ik}H_{ij},\ W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n}$$

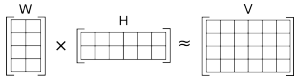

Figure 2.2: An illustration of the NMF method[15].

The matrix W consists of r positive basis vectors and H represents the weights that are used to approximate X. In a dimensionality reduction context, r is usually a small number. The pseudocode of the NMF algorithm is shown below:

Algorithm 3 Non-negative Matrix Factorisation

Input: $m \times n$ matrix A, rank r
Output: $m \times r$ matrix W, $r \times n$ matrix H

1: Initialise W as a random dense matrix.
2: **for** i = 1 ... max_iter **do**
3: H = $W^T A(W^T W)^{-1}$
4: Set all negative elements in H to be 0.
5: W = $HA^T(WW^T)^{-1}$
6: W = W^T
7: Set all negative elements in W to be 0.
8: **end for**

Recursive Feature Elimination

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Recursive Feature Elimination is commonly used together with many classification algorithms to build more efficient classifiers. It attempts to eliminate dependencies and collinearity that may exist in the model by producing a ranking of features as well as candidate subsets[16]. A small number or a small percentage of features are recursively eliminated per loop based on their ranking.

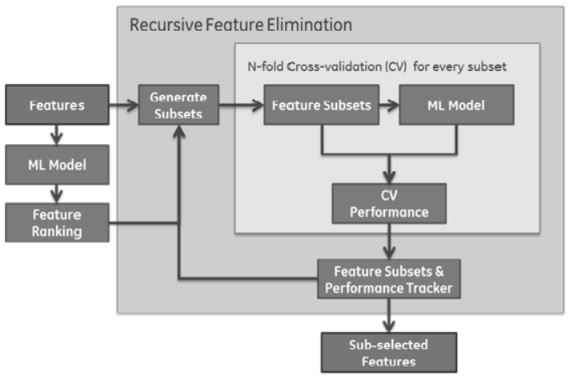

Figure 2.3: RFE Framework for Feature Selection.[17].

Recursive Feature Elimination is commonly used together with many classification algorithms as the combination of RFE with classification algorithms leads to a lower data dimension and higher computational efficiency. Support Vector Machine (SVM) based Recursive Feature Elimination (SVM-RFE) selects patterns using SVM's weight. Support Vector Machine (SVM) models are a powerful tool to identify predictive models or classifiers, not only because they accommodate well sparse data but also because they can classify groups or create predictive rules for data that cannot be classified by linear decision functions. The pseudocode of the RFE algorithm is shown below:

Algorithm 4 Recursive Feature Elimination

Input: $m \times n$ matrix A, Machine Learning Model m, Subset size k
Output: $m \times k$ matrix B

1: $B = A$
2: **while** *number of features in* $A > k$ **do**
3: Calculate the rank of each feature using m on a ranking criteria.
4: r = feature with the lowest value on the ranking criteria.
5: Remove r from B.
6: **end while**

2.3 Machine Learning

Machine learning is a data analysis technique that uses computational methods to extract information from datasets without requiring explicit programming. Broadly speaking, there are three types of machine learning algorithms – supervised, unsupervised, and reinforcement learning[18].

In the context of supervised learning, the algorithm learns from a dataset that are already labelled with the correct answer and tries to make predictions on unforeseen instances[19, 20]r. For unsupervised learning, the dataset does not have any labelled data, and it is up to the algorithm to capture the hidden rules or structures that best describes the data. Reinforcement learning aims to train an agent, usually through trial and error, to choose a series of appropriate actions in order to reach its designated goal[21].

In our thesis, we used four supervised algorithms to see how well our datasets could classify new instances from our dataset. We then evaluated and compared the performance of four different feature engineering techniques (Support Vector Machines, Gaussian Naïve Bayes, K-Nearest Neighbours and Random Forest) by conducting the same classification task on the reduced datasets generated by our techniques to assess how well they performed against the raw dataset and each other.

2.4 Machine Learning Classifiers

Support Vector Machines

Support Vector Machines (SVMs) are a set of related methods for supervised learning, applicable to both classification and regression problems[22]. It has a good performance in a wide variety of real-world applications. SVM is robust and efficient on high dimensionality datasets, which makes it a popular method to use within the field of bioinformatics, especially on pattern recognition and function classification tasks.

SVM classifiers aims to orient a hyperplane that lies in a transformed input space in a way that the members from both classes are separated as far as possible from the hyperplane. For multi-class classification problems, the classifier employs a One-vs-Rest strategy, which the classifier will try to create a hyperplane for each class, treating the other classes as negative samples.

Gaussian Naïve Bayes

A Naive Bayes (NB) classifier is a probabilistic machine learning model based on the Bayes theorem that has also achieved a high performance in bioinformatics. Gaussian Naïve Bayes (GNB) assumes the features form the dataset are independent from each other and it the dataset has a Gaussian distribution (normal distribution)[23].

Gaussian Naïve Bayes classifier uses the probability of the trained instance and computes the probability of an unknown sample, the outcome of the sample is then classified to be a certain class that has the highest conditional probability that is calculated using its features.

K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a non-parametric instance-based learning method that classifies an instance using previous instances. The simplicity and its ability at learning nonlinear relationships made KNN a good classifier to be incorporated on different highly complex datasets[24].

K-nearest neighbour algorithm assigns a class to an unknown sample by comparing it against the trained dataset. The class of the new unknown sample will then be assigned by considering the class labels of the k most similar instances in the trained dataset.

Random Forest

Random Forest (RF) is a stochastic classification algorithm based on decision trees. Its ability to adapt to datasets that are collinear or does not have a large number of samples made RF a popular classifier for high-dimensional genomics data analysis.

Random Forest ensembles a large number of individual decision trees. Each tree in the forest predicts the instance and the class with the most votes will be selected. The algorithm usually has a certain amount of randomness added to the model whilst growing the trees. Instead of searching over the whole dataset for a feature that is the most important, the algorithm will search in a small random subset of features for the optimal feature to use. The individual nodes will then operate together to buffer each other from their errors, and thus create a better model in general.

3 Implementation

Contents

3.1 Introduction

This chapter presents an analysis of 12 different publicly available microarray datasets. Each dataset contains a huge set of molecular probes, and the objective is to classify if a patient has a particular disease based on the probes provided. We first assessed the default performance of the dataset by running it through four different machine learning classifiers. Next, we applied a set of four feature engineering techniques – PCA, PLS-DA, NMF, RFE to treat the dataset before it was passed into the classifiers. Finally, we formally compared to performances

acquired using different statistical measurements. This process is then repeated multiple times to find the optimal set of hyperparameters using grid search.

3.2 Experimental Setup

In this paper, we used three feature transformation and one feature selection methods to reduce the size of the dataset: principal component analysis (PCA), partial least squares discriminant analysis (PLS-DA), Non-negative matrix factorisation (NMF), and recursive feature elimination using support vector machines (SVM-RFE). F_1 score is used as the performance metric for our experiments. It is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

F_1 score is more descriptive than accuracy or other evaluation metrics in this context since the datasets we acquired all have uneven class distributions, F_1 score takes it into account and allows us to benchmark the datasets justly.

We assessed the performances of our datasets using four different classifiers: random forest (RF), gaussian naive Bayes (GNB), Support Vector Machines (SVM) and K-nearest neighbours (KNN). For experiments involving SVM, we used the linear kernel as it is reported to have the best performance out of the few kernel options available. Each classifier we chose uses a different approach and criterion for its prediction. The different approaches simulate different classification scenarios in order to maximise the coverage and applicability of our results.

We used 10-fold stratified cross-validation (SCV) scheme to evaluate the performance of the model and its hyperparameters. Cross-validation is a resampling procedure that splits the dataset randomly into k folds, in which one of the groups is treated as the validation set, and the rest as the training set. SCV builds on the concept of normal k-fold cross validation and seeks to produce folds that are representative of all strata of the data, so that each fold contains approximately the same proportion of data that represents a certain class. SCV is

a necessary step in pipeline, as all the datasets we used for this experiment has a very limited number of samples. The SCV scheme minimises the potential bias of our results created by manual partitions and helps up to achieve a more generalised relationship between our feature engineering techniques and the classifier. The feature engineering techniques will be applied to each fold separately to adjust for the different distributions created by our k-fold SCV procedure. The tuning parameters are generated using grid search per fold to ensure that the best hyper-parameters are used for the model in hopes of yielding better overall performance.

All experiments are performed on an AWS Machine Learning ASIC instance with 4vCPU and 8GB of memory. We implemented all our feature engineering techniques and classifiers using scikit-learn, a powerful open source Python-based machine learning library.

3.2.1 Datasets Used

This chapter presents an analysis of twelve different cancer-related transcriptomics datasets[25, 26] to compare the performances of various feature engineering techniques. A brief summary of the datasets is shown in Table 3.1.

Dataset	# Total Probes	# Instances	# Classes
Breast Cancer	24481	97	2
Central Nervous System	7129	60	2
Colon Tumor	2000	62	2
ALL-AML	7129	72	2
ALL-AML-3	7129	72	3
ALL-AML-4	7129	72	4
Lung Cancer	12600	203	5
Lymphoma	4026	66	3
MLL	12582	72	3
Ovarian Cancer	15154	253	2
Prostate Cancer	2135	102	2
SRBCT	2308	83	4

Table 3.1: Description of 12 Microarray Datasets

For data pre-processing, we did two steps: filling missing values and scaling. For all training and testing datasets, missing values are filled using the mean value of

the column. Then we scaled the datasets using a min-max Scaler such that every probe expression will have a value between 0 and 1. We chose a min-max scaler instead of the conventional normalisation using mean and standard deviation. We did this because NMF does not allow the presence of negative values.

3.3 Exploratory Analysis

Before we conducted our experiments, a formal comparison was held on different aspects of our selected techniques. It is an essential step before we held our experiments as it provides the proper context needed to develop an optimal model and interpret our results objectively. In this section, we explored four different aspects that we thought is essential for this context: Algorithmic Performance, raw dataset performance, and the variables and hyperparameters used for our SCV-GS pipeline.

3.3.1 Algorithmic Performance Comparison

We assessed the algorithmic performance of the four different techniques by comparing the time taken for each technique to complete its reduction task. The independent variable is selected to be the number of attributes that the reduced dataset needs to create or keep. We measured the performance by timing when it starts and finishes, then we took the difference between the two timers to get our results. This process is repeated 30 times, such that the sample distribution would tends towards normal. The results of our findings are shown in Figure 3.1.

Overall, the average time taken for PCA, PLS-DA and NMF rose when we increased the number of features, while a slight downwards trend can be seen on RFE. PCA, NMF, and PLS-DA has a better performance on lower dimensionalities, yet the opposite holds true for RFE. This is expected behaviour since, for RFE to select less features, more iterations are needed to for the elimination process. It can also be observed that PCA is the fastest technique out of the four, with PLS-DA followed closely behind. NMF is slowest technique out of the four, it is the only technique that shows an exponential growth on time complexity when the number

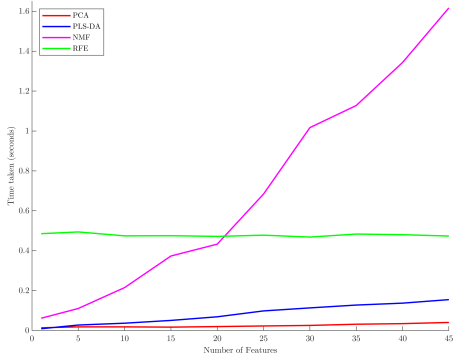

Figure 3.1: Average time taken for feature engineering techniques to run

of features increase. Therefore, in terms of time complexity, PCA and PLS-DA achieved the best performances for the reduction task on hand.

3.3.2 Performance on Raw Datasets

Before we can properly compare the datasets, we first evaluated the performance of our raw dataset perform under our four classifiers. The purpose of such evaluation is to provide a basis for the subsequent tests, in which we will assess how well an reduced dataset would perform in comparison to the original. The performances of the raw datasets are shown in Table 3.2.

dataset	SVC	GNB	KNN	RF	Average F_1
Breast	0.664	0.434	0.583	*0.693	0.594
CNS	0.645	*0.647	0.643	0.644	0.645
Colon	*0.835	0.525	0.763	0.709	0.708
ALL-AML	0.981	*1.000	0.808	0.820	0.902
ALL-AML-3	0.950	*0.965	0.788	0.851	0.889
ALL-AML-4	0.891	*0.904	0.729	0.746	0.818
Lung	*0.959	0.887	0.869	0.876	0.898
Lymphoma	*1.000	0.910	*1.000	0.958	0.967
MLL	*0.970	0.957	0.873	0.797	0.899
Ovarian	*1.000	0.918	0.919	0.961	0.950
Prostate	*0.907	0.740	0.869	0.891	0.852
SRBCT	*1.000	*1.000	0.788	0.8857	0.918

Table 3.2: Base Classification Score of 12 Microarray Datasets

We can see that 7 out of 11 raw datasets are able to achieve an average classification performance above 85%, and breast cancer has the lowest classification performance out of all datasets with a 59.4% overall performance. The best performing classifier for each of the categories are marked with an asterisk (*). In hindsight, we can see that the support vector classifier (with linear kernel) and gaussian naive bayes are the best performing classifiers in this context. We will further inspect their performances in subsequent sections.

3.3.3 Deciding the Number of Variables

Finding the optimal number of variables to reduce the dataset into is generally very difficult task since the optimal number of features vary between each dataset. A set of experiments are first conducted on the three datasets by varying the number of components extracted or features selected. We chose the top three performing datasets in our previous experiment where we compared each datasets base performance for this experiment. The datasets are Ovarian, Lymphoma and SRBCT, each representing a binary, ternary and quaternary classification problem. The three datasets we selected also each represent datasets with different sizes. A brief summary of the three datasets is shown in Table 3.3.

dataset	# Total Probes	# Instances	# Classes	Average F_1
Ovarian	15154	253	2	0.950
Lymphoma	4026	66	3	0.967
SRBCT	2308	83	4	0.918

Table 3.3: Dataset used to Decide the Number of Variables

Four line graphs were used to illustrate the F1-scores acquired by the four feature selection techniques. Each graph captures how each feature engineering technique performed under a certain classifier, and 10-fold cross validation to evaluate their performances. We kept the size of the subset below the number of instances in the training set since PCA and methods derived from it (PLS-DA, NMF) cannot produce datasets that has over min(n_samples, n_features) components. Lymphoma has the smallest number of samples, being at 62. Since a 10-fold cross

validation model was used, therefore we cannot allow the number of attributes selected to be higher than 90% of the number of samples in the Lymphoma dataset. Such measure has to be employed in order to produce comaprable results for our analysis. Therefore, the maximum number of attributes selected must be kept under 55. But instead of iterating through possible subset sizes, we chose to cap the maximum number of selected attributes at 45 as a compromise of computation time and search space coverage.

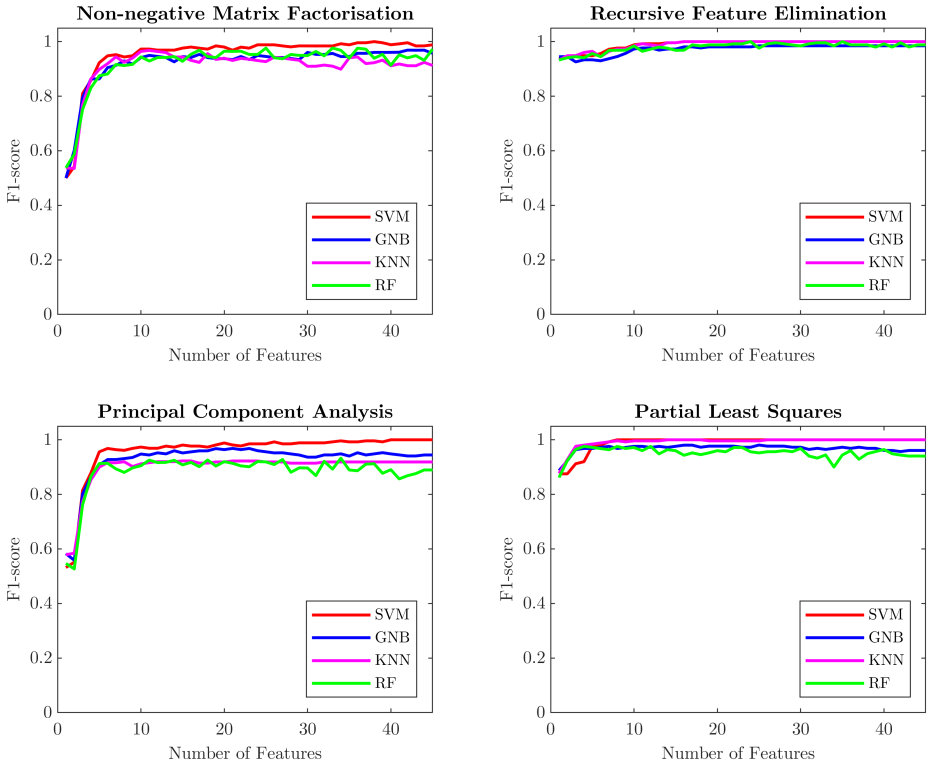

Figure 3.2: Effect of Different Feature Engineering Methods (Ovarian)

From Figure 3.2, we can see that most feature engineering techniques are able to achieve excellent performance on the Ovarian dataset. All four methods are able to produce models with outstanding classification performances. When less than five features were used, it can be observed that recursive feature elimination achieved the best performance out of the four, achieving a stunning F_1 score on all four classifiers.

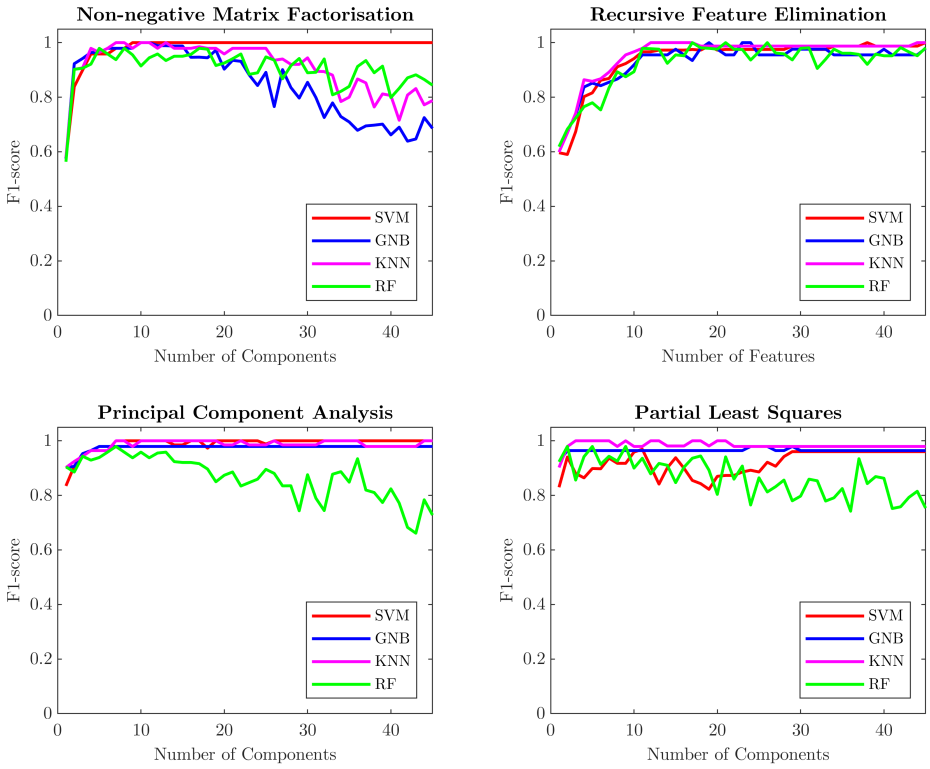

Figure 3.3: Effect of Different Feature Engineering Methods (Lymphoma)

From Figure 3.3, it can be noticed that the performance of the random forest classifier dropped when more than 20 attributes were used. It shows a possibility that the random forest classifier is prone to under-fitting the model when too many features are presented to a tree with a low depth. It can also seen that NMF did not perform well when supplied with a reduce dataset of with a high dimensionality - Except for SVM, all feature engineering techniques' performance starts to plummet when the size of the reduced dataset has over 20 components.

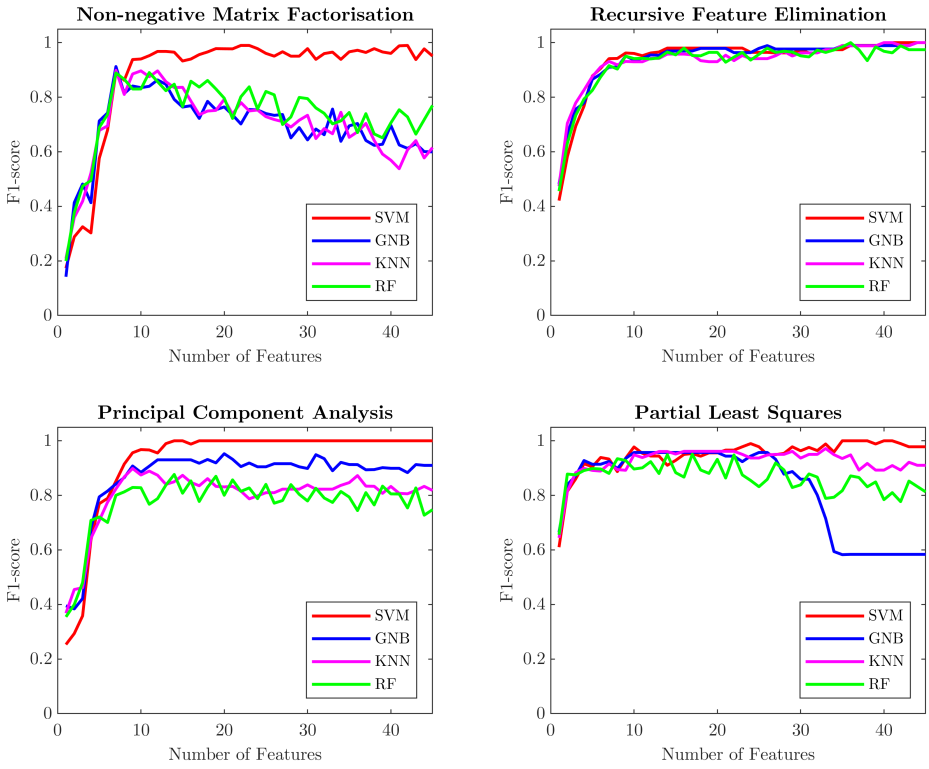

Figure 3.4: Effect of Different Feature Engineering Methods (SRBCT)

From Figure 3.4, a same trend can be observed for NMF, performance decreases as the size of the reduced dataset increases. SVM was able to converge to a high performance and it did not drop when more attributes are introduced to the classifier. In terms of feature engineering techniques, once again RFE achieved excellent performance when paired with all feature engineering techniques.

Overall, we can see that SVM is able to achieve the best and most consistent performance out of the four feature engineering techniques we tested. Random forest's performance fluctuates the most out of the four feature engineering techniques, this is an expected behaviour because of the stochastic nature of the algorithm. For most combination of feature engineering technique and classifier, the pair was able to reach its maximum performance when the reduced dataset has approximately 5 to 6 features, and the performance starts to plateau afterwards. the classification performance drops when more features are added to NMF.

The results of our experiments strongly suggest that only a small number of probes is needed for our classifiers to accurately classify our instances. This makes sense intuitively since each cancer only affects a small subset of genes, therefore most of the probes in the database are irrelevant to this classification context. Based on our observations, SVM has the maximum flexibility out of the four classifiers, as its performance did not drop when a larger subset was passed into the classifier. For feature engineering techniques, RFE is chosen to be the best performing technique because of its high adaptability to different classifiers, and its flexibility to produce subsets of different sizes without sacrificing performance. The optimal subset size to reduce our dataset into is around 5 to 20 features, as for most classifiers we tested, their predictive power decreases when more than 20 attributes are used.

3.3.4 Deciding the Hyperparameters

Grid search was used in pipeline to find the best set of hyper-parameters to ensure that each all combinations of feature engineering technique and classifier reaches its optimal performance. Grid search is an optimisation algorithm that, when given multiple lists of possible hyperparameters, it searches through all possible combinations in order to find a combination of hyperparameters that best optimises the classifier. It is a computationally intensive algorithm to use since the number of models that needs to be tested increases exponentially to the size of the lists passed in. Therefore, we also used scikit-learn's implementation of grid search that uses the library joblib to run the code using single-host, process-based parallelism such that the time needed to train our models would decrease significantly.

The hyperparameters were passed into our grid search are based on the results in Section 3.3.3. We used the same list for the number of components to be extracted, and the number of features to be selected by our techniques in order to ensure a more consistent comparison between the two types of feature engineering. We used [5, 7, 9, 11, 15, 20] as the number of features/components to be generated, based on the fact that all of the feature engineering techniques were able to converge to its optimal performance when the reduced dataset has approximately 5 to 10 dimensions.

Therefore, the list we used contains more sizes from said range. We also considered the fact that most techniques performed worse when more than 20 attributes are used. Therefore, the maximum number of attributes we considered in our pipeline will be capped at 20. We also modified the cost value C in linear Support Vector Classifier (SVC), the value k and the weight function in K-nearest neighbours (KNN), the maximum depth and the number of estimators in Random Forest, in hopes on finding a model that could improve the performance of our machine learning pipeline.

In total 12 datasets × 10 folds × 4 FE techniques × 24 grid search models × 6 different numbers of features = 69120 models are built by our pipeline.

3.4 Results

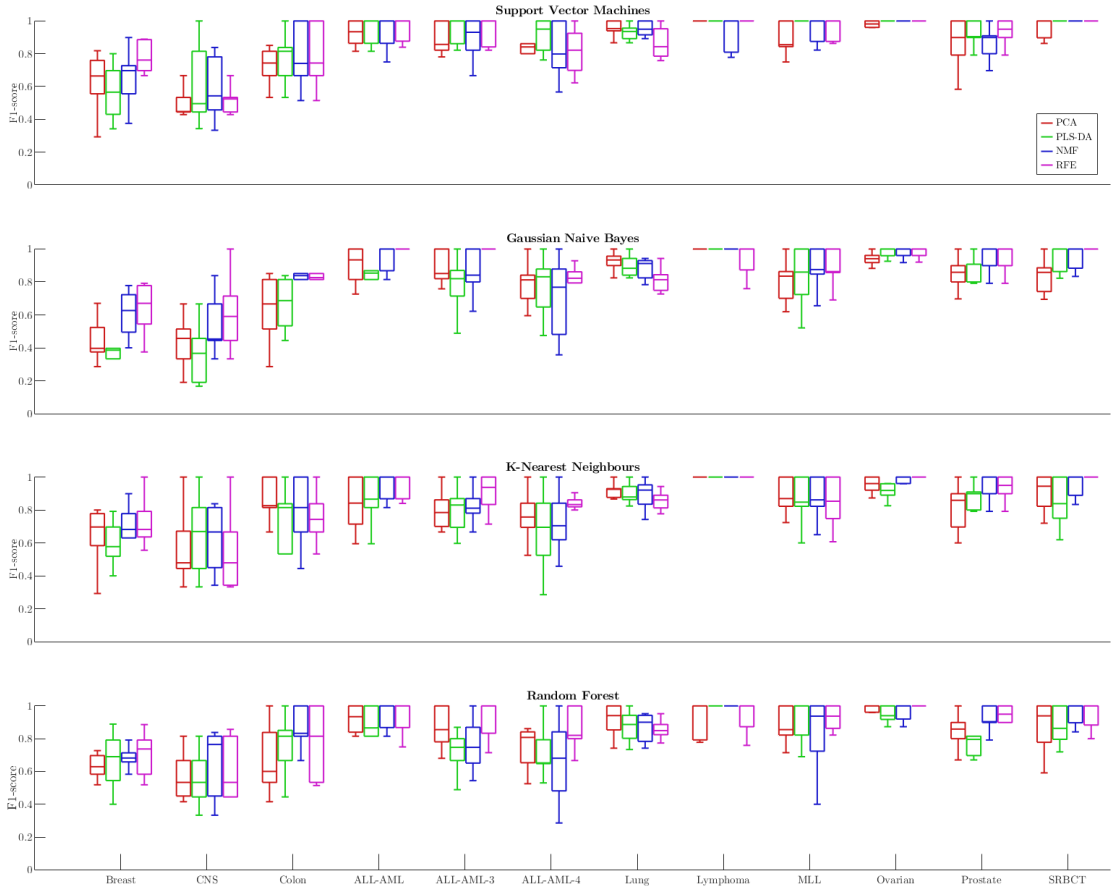

Figure 3.5: Distribution of Performance Calculated using 10-fold SCV-GS Pipeline.

The results of the our classification experiments are shown below. In Figure 3.5, Each plot represents the feature engineering techniques' performance on the twelve

datasets with respect to different classifiers. Overall, we noticed a higher spread on datasets that did not perform well in our previous test on Section 3.3.2. In terms of classifiers, a higher spread can be observed on all datasets using KNN and RF. None of the technique has a visibility better than the other methods. Therefore, further analysis is needed to compare the performance of each technique. The full table showing all classification accuracies we acquired are shown in Table 3.4 and Table 3.5:

FE Tech.	Classifier	Breast	CNS	Colon	ALL-AML	ALL-AML-3	ALL-AML-4
PCA	SVM	0.566 (14)	0.597 (6)	0.784 (5)	0.942 (4.5)	0.952 (2)	**0.911 (1)**
	GNB	0.413 (16)	0.367 (16)	0.678 (14)	0.849 (13)	0.801 (12)	0.777 (9)
	KNN	0.590 (13)	**0.660 (1)**	0.743 (12)	0.832 (16)	0.766 (13)	0.703 (14)
	RF	0.667 (5)	0.543 (10)	0.776 (8)	0.841 (14)	0.723 (16)	0.682 (15)
PLS-DA	SVM	0.663 (6)	0.588 (7)	0.774 (9)	0.948 (2)	0.896 (7)	0.816 (5)
	GNB	0.607 (12)	0.533 (12)	0.803 (4)	0.910 (11)	0.851 (10)	0.723 (13)
	KNN	0.650 (8)	0.624 (3)	0.779 (7)	0.942 (4.5)	0.819 (11)	0.724 (12)
	RF	0.670 (4)	0.657 (2)	**0.848 (1)**	0.913 (9)	0.757 (15)	0.667 (16)
NMF	SVM	0.632 (9)	0.519 (14)	0.731 (13)	0.922 (8)	0.898 (6)	0.830 (4)
	GNB	0.437 (15)	0.420 (15)	0.630 (16)	0.909 (12)	0.875 (8)	0.781 (8)
	KNN	0.657 (7)	0.560 (9)	0.833 (3)	0.838 (15)	0.761 (14)	0.749 (10)
	RF	0.628 (11)	0.572 (8)	0.664 (15)	0.923 (7)	0.862 (9)	0.747 (11)
RFE	SVM	0.711 (3)	0.536 (11)	0.782 (6)	0.925 (6)	0.937 (3)	0.813 (6)
	GNB	0.631 (10)	0.613 (4)	0.845 (2)	**0.969 (1)**	**0.971 (1)**	0.810 (7)
	KNN	0.715 (2)	0.531 (13)	0.767 (11)	0.912 (10)	0.911 (5)	0.836 (3)
	RF	**0.745 (1)**	0.600 (5)	0.773 (10)	0.943 (3)	0.924 (4)	0.848 (2)

Table 3.4: Classification Scores (Breast - ALL-AML-4)

FE Tech.	Classifier	Lung	Lymphoma	MLL	Ovarian	Prostate	SRBCT
PCA	SVM	0.921 (4)	0.985 (3)	**0.957 (1)**	0.992 (4.5)	0.909 (6)	**1.000 (1)**
	GNB	0.899 (8)	0.979 (5.5)	0.844 (15)	0.960 (12)	0.819 (14.5)	0.940 (11)
	KNN	0.897 (9)	0.964 (9.5)	0.856 (12)	0.918 (16)	0.860 (11)	0.844 (15)
	RF	0.880 (11)	0.979 (5.5)	0.911 (5)	0.942 (14)	0.806 (16)	0.870 (14)
PLS-DA	SVM	**0.951 (1)**	0.928 (15)	0.942 (2)	**1.000 (2)**	0.880 (9)	0.976 (3)
	GNB	0.889 (10)	0.964 (9.5)	0.898 (6)	0.980 (8.5)	0.890 (8)	0.946 (10)
	KNN	0.904 (7)	**1.000 (1)**	0.877 (11)	0.984 (7)	0.918 (5)	0.950 (9)
	RF	0.870 (12)	0.960 (11)	0.850 (14)	0.967 (11)	0.900 (7)	0.961 (7)
NMF	SVM	0.950 (2)	0.958 (12)	0.897 (7)	0.980 (8.5)	0.867 (10)	0.964 (6)
	GNB	0.924 (3)	0.979 (5.5)	0.813 (16)	0.941 (15)	0.850 (12)	0.810 (16)
	KNN	0.918 (5)	0.979 (5.5)	0.887 (9)	0.951 (13)	0.819 (14.5)	0.908 (12)
	RF	0.911 (6)	0.921 (16)	0.891 (8)	0.988 (6)	0.849 (13)	0.883 (13)
RFE	SVM	0.867 (13)	0.973 (8)	0.935 (3)	**1.000 (2)**	0.929 (2.5)	0.980 (2)
	GNB	0.792 (16)	0.939 (13.5)	0.883 (10)	0.977 (10)	**0.938 (1)**	0.970 (4.5)
	KNN	0.856 (14)	0.987 (2)	0.855 (13)	**1.000 (2)**	0.928 (4)	0.970 (4.5)
	RF	0.852 (15)	0.939 (13.5)	0.927 (4)	0.992 (4.5)	0.929 (2.5)	0.953 (8)

Table 3.5: Classification Scores (Lung - SRBCT)

To formally compare our results, we ranked the F_1 score produced by different pairs of feature engineering technique and classifier. In cases where two pairs have the same rank, the average rank of the two is used. Afterwards, we used the Friedman rank based test and the Nemenyi post-hoc test to check if the differences in the performances are statistically significant[27]. We conducted the Friedman test on the list of performances acquired using our SCV-GS pipeline and acquired a p value of 8.3×10^6. Therefore, it provided evidence to show that all methods did not produce the same results. Next, we conducted the Nemenyi test on the same table and the results produced are shown in Table 3.6. From the table, we noticed that PCA-SVM is statistically different from other PCA variants.

	PCA.SVM	PCA.GNB	PCA.KNN	PCA.RF	PLS-DA.SVM	PLS-DA.GNB	PLS-DA.KNN	PLS-DA.RF	NMF.SVM	NMF.GNB	NMF.KNN	NMF.RF	RFE.SVM	RFE.GNB	RFE.KNN
PCA.GNB	**0.006**	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PCA.KNN	**0.012**	1.000	-	-	-	-	-	-	-	-	-	-	-	-	-
PCA.RF	**0.040**	1.000	1.000	-	-	-	-	-	-	-	-	-	-	-	-
PLS-DA.SVM	1.000	0.065	0.113	0.264	-	-	-	-	-	-	-	-	-	-	-
PLS-DA.GNB	0.358	0.993	0.999	1.000	0.841	-	-	-	-	-	-	-	-	-	-
PLS-DA.KNN	0.989	0.403	0.547	0.792	1.000	0.998	-	-	-	-	-	-	-	-	-
PLS-DA.RF	0.514	0.971	0.992	1.000	0.931	1.000	1.000	-	-	-	-	-	-	-	-
NMF.SVM	0.805	0.829	0.917	0.987	0.994	1.000	1.000	1.000	-	-	-	-	-	-	-
NMF.GNB	**0.012**	1.000	1.000	1.000	0.113	0.999	0.547	0.992	0.917	-	-	-	-	-	-
NMF.KNN	0.276	0.998	1.000	1.000	0.765	1.000	0.994	1.000	1.000	1.000	-	-	-	-	-
NMF.RF	0.151	1.000	1.000	1.000	0.580	1.000	0.967	1.000	1.000	1.000	1.000	-	-	-	-
RFE.SVM	1.000	**0.046**	0.083	0.207	1.000	0.779	1.000	0.892	0.987	0.083	0.692	0.498	-	-	-
RFE.GNB	0.998	0.252	0.373	0.628	1.000	0.987	1.000	0.998	1.000	0.373	0.971	0.901	1.000	-	-
RFE.KNN	0.994	0.344	0.481	0.737	1.000	0.996	1.000	0.999	1.000	0.481	0.989	0.949	1.000	1.000	-
RFE.RF	1.000	0.113	0.187	0.387	1.000	0.924	1.000	0.974	0.999	0.187	0.873	0.722	1.000	1.000	1.000

Table 3.6: Friedman-Nemenyi Test p-values

Subsequently, we tested the performances of the reduced datasets against the raw datasets to test that if feature engineering and grid search increases the classifiers' performance. A paired, one-tail T-test is used to see if there is a statistically significant improvement in performance for our reduced dataset. The results are shown in Table 3.7:

From our results, one can see that there is a statistically significant increase in the model's predictive performance when feature engineering methods are used on KNN and RF. For SVM and GNB, although a significant increase in its predictive performance cannot be observed, the reduced datasets did not have a negative impact on the two classifiers.

Table 3.8 showed the average rank each combination achieved. In terms of classifiers, RFE clearly performed the best out of the four feature engineering techniques; it is ranked first for GNB, KNN, and RF, and ranked second for SVM. In contrast, NMF is declared the worst feature engineering technique out of the four,

Dataset	SVM		GNB		KNN		RF	
	Before	After	Before	After	Before	After	Before	After
Breast	0.664	0.711	0.434	0.631	0.583	0.715	0.693	0.745
CNS	0.645	0.597	0.647	0.613	0.643	0.660	0.644	0.657
Colon	0.835	0.784	0.525	0.845	0.763	0.833	0.709	0.848
ALL-AML	0.981	0.948	1.000	0.969	0.808	0.942	0.820	0.943
ALL-AML-3	0.950	0.952	0.965	0.971	0.788	0.911	0.851	0.924
ALL-AML-4	0.891	0.911	0.904	0.810	0.729	0.836	0.746	0.848
Lung	0.959	0.951	0.887	0.924	0.869	0.918	0.876	0.911
Lymphoma	1.000	0.985	0.910	0.979	1.000	1.000	0.958	0.979
MLL	0.970	0.957	0.957	0.898	0.873	0.887	0.797	0.927
Ovarian	1.000	1.000	0.918	0.980	0.919	1.000	0.961	0.992
Prostate	0.907	0.929	0.740	0.938	0.869	0.928	0.891	0.929
SRBCT	1.000	1.000	1.000	0.970	0.788	0.970	0.885	0.961
p-value	7.80E-02		1.49E-01		**6.09E-04**		**2.46E-04**	

Table 3.7: Classification Scores Comparison

	PCA	PLS-DA	NMF	RFE
SVM	**4.333 (1)**	5.667 (3)	8.292 (4)	5.458 (2)
GNB	12.167 (4)	9.500 (2)	11.792 (3)	**6.667 (1)**
KNN	11.792 (4)	7.125 (2)	9.750 (3)	**6.958 (1)**
RF	11.125 (4)	9.083 (2)	10.250 (3)	**6.042 (1)**

Table 3.8: Classification Average Rank Comparison

the reasoning behind is that it achieved some of the lowest average rankings out of the 16 pairs, NMF ranked fourth with paired with SVM, and ranked third for the rest.

In terms of the classification performance, SVM-PCA achieved the best overall performance, but it should be emphasised that other classifiers did not pair well with PCA and achieved the worst results out of the 16 pairs we assessed. Therefore, it might not the most versatile feature engineering method to use for all contexts. SVM-PLS, SVM-RFE, and RF-RFE has a slightly worse performance, but both PLS and RFE were able to pair well with all the classification algorithms, and each have their own advantages: PLS can create a reduced dataset in a very short amount of time, and RFE has a very stable performance with regards to the number of features selected. Therefore, we would recommend the three combinations of

feature engineering technique and classifiers to deal extract concise information from high-throughput omics datasets.

3.5 Conclusion

In this chapter, we used the 10-fold SCV-GS model to compare the performance between four different feature engineering techniques, and four different machine learning classifiers. Several approaches were used to evaluate the effectiveness of feature engineering, such as by comparing the performances of the raw dataset against the reduced dataset created by our feature engineering techniques, and comparisons between multiple reduced datasets. We assessed the performance we acquired using statistical tests and our results show that various classification models performed better after feature engineering was used, and the techniques were able to achieve a similar, or better results than a raw dataset running on the same pipeline.

Our findings concluded that SVM is the best classifier to use in a biological context out of the four. It is a stable classifier that were able to create highly effective models to evaluate our datasets. KNN and RF are also classifiers that are worth considering. GNB is the worst performing classifier out of the four, and we do not recommend using GNB to classify biological instances due to its lower accuracy and stability compared to the other classifiers. In terms of feature engineering techniques, RFE performed the best. It was consistently ranked the highest out of all feature engineering techniques and had a stable performance when matched with all classifiers. PLS-DA and RF have similar traits to RFE. Although they did not perform as good as RFE, they are still viable techniques to use for feature engineering tasks.

For the reasons mentioned above, here we present our recommended combinations of feature engineering technique and classifier for the future data analysis projects in the bioinformatics. Performance wise, PCA-SVM is the best combination out of the 16 pairs, but PCA does not pair well with other classifiers. If stability is your highest concern, SVM-RFE, SVM-PLS, and RF-RFE are the best pairs to be used, as they are statistically proven to provide consistent yet excellent results.

4
Knowledge Extraction

Contents

4.1 Introduction

This chapter evaluated the significance of the biological knowledge extracted by different combinations of feature engineering technique and classifier using the Lung dataset as a case study. The dataset contained 203 samples and it runs on the AffyMetrix U133 Plus 2.0 gene chip. We converted the probes into genes using DAVID, an online analysis wizard[28] that uses a module-centric approach to analyse large gene lists by their functions. Afterwards, we associated extracted genes with several online disease databases to check if the genes we extracted are related to Lung cancer.

4.2 Methods

The first step to knowledge extraction is to select feature from our dataset. For feature engineering technique, a set of features will be selected via a criterion that differs from technique to technique. The feature selected, in this case, molecular probes, were converted to identifiable genes that could then be analysed. We used the AFFYMETRIX_3PRIME_IVT_ID identifier provided in DAVID for our conversion. In the following sections, we explain in detail on the technique we applied to extract the best probes from the reduced dataset.

As mentioned in Chapter 3, feature engineering was applied on each fold in our SCV-GS pipeline to take in account of the dataset's different distributions. Therefore, each fold will have a slightly different reduced dataset. Therefore, to ensure we captured all biological knowledge from the method, we combined all sets of probes extracted from the pipeline for our analysis. The probes are selected via different criteria for each feature engineering technique. Therefore, in the following sections, we will provide a brief explanation of the criterion we used for each of our technique.

4.2.1 Feature Selection

Selecting the best probes from feature selection methods is simple, since feature selection algorithms reduces the size of the dataset by removing features with respect to the task being performed. The features are not modified in any way, therefore can be directly used for our analyses.

4.2.2 Feature Transformation

Feature Transformation techniques reduces the size of the dataset by combining or reformatting features. Therefore, it is comparatively harder than feature selection techniques to extract knowledge out from the transformed attributes. Traditionally, if one were to extract knowledge from the reduced dataset, each attribute must be manually interpreted order to explain the information captured by the attribute. But in this paper, we used several different approaches that automates this process. The methods we chose for our feature transformation methods are described below:

Principal Component Analysis

For PCA, the components are ordered by the amount of variance explained by the feature. Therefore, the first three components are used for our analysis. We ranked the genes of each component by their its value. Then the top 10 genes that has the highest absolute value in each of the component are selected.

Partial Least Squares

For PLS, we incorporated a similar approach to that of PCA. But instead of using absolute values, the probes are ranked using the VIP score. VIP scores are calculated using the weighted sum of the correlations between the components and the original feature. The top 5 features with the highest VIP-score will be selected.

Non-negative Matrix Factorisation

For NMF, a suitable method for extracting and ranking the genes does not exist. For knowledge extraction, NMF is mainly used to reveal the latent information hidden in dataset, which made it not suitable for our selecting genes as it is impossible to extract genes from the transformed matrices[29]. Therefore, we did not use NMF for knowledge extraction.

4.2.3 Quantifying Captured Biological Insights

Figure 4.1 illustrates the average number of genes extracted from the twelve datasets using the 10-fold SCV-GS pipeline. It is important to note that PCA and PLS-DA retrieved less probes than RFE because we arbitrarily defined the number of probes selected above. RFE can retrieve as much as 20 probes per fold, as it can select as much as 20 features from the original dataset. PCA extracted an average of approximately 15 features for all twelve datasets, which shows that almost all the features selected from each fold are of the same features. PLS-DA showed little to no variation in terms of the number of features selected, which shows that the probe selection process was not affected by the classifier at all.

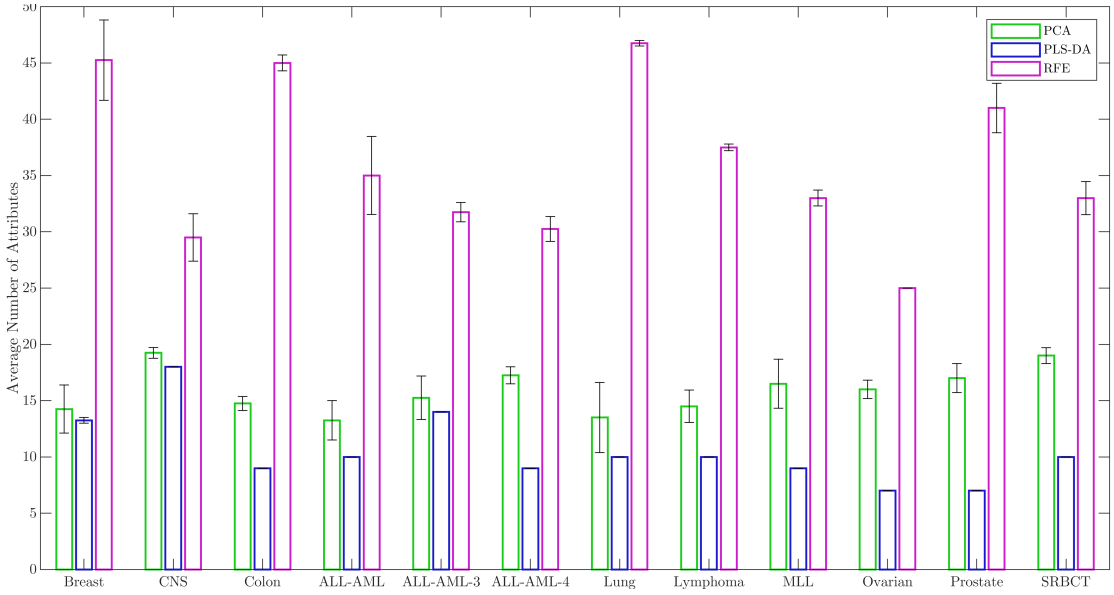

Figure 4.1: Average Number of Probes Extracted using 10-fold SCV-GS Pipeline.

After the probes were converted to gene IDs, we assessed the relevance of the gene by comparing the genes extracted by our feature engineering techniques to experimental results available online. We used a combination of four online disease databases: Malacards[30], Uniprot[31], Orphanet[32] and CTD[33].

4.3 Case Study

In order to test the correlation of the reduced dataset generated by our combination of feature selection technique and classifier, we used the Lung dataset to conduct our experiments. The samples were on the Affymetrix U-133 Plus 2.0 gene chip, which contains 54,613 probes derived from approximately 20,000 different genes.

4.3.1 Results

We assessed the performance of the selected feature engineering techniques, with respect to the amount of biological knowledge it captured. A comparison of the feature engineering techniques were conducted by the percentage of genes that matched with our combined database. The results of our comparison is shown below:

From Table 4.1, we can see that both PLS-DA and RFE were able to extract genes that matches the combined dataset. RFE and PLS-DA both performed well in this

Lung	Positives	Negatives	Accuracy
PCA	0.000 ± 2.708	16.000 ± 0.000	0.000 ± 0.000
PLS-DA	5.000 ± 0.000	2.000 ± 0.000	0.714 ± 0.000
RFE	42.75 ± 0.500	10.500 ± 0.577	0.754 ± 0.012

Table 4.1: Performance of Extracted Genes against Experimental Results

context, approximately 75% of the genes that isolated matches with our database. Upon further investigation, we found two genes that sparked our interest. Both PLS-DA and RFE selected the gene TFAP2C. Transcription factor AP-2 gamma (TFAP2C) is a gene that is involved in activating various developmental genes. It is also a common cell marker for breast and prostate cancer [34]. Peroxiredoxin (PRDX1) is a gene that protects cells from antioxidants, and it is shown as an important biomarker for lung cancer[35]. The probes extracted using PCA could not be converted into genes. Upon further inspection, we noticed that the probes extracted using PCA were the control probes used for technical quality checks and no gene are associated to said probe. Therefore, it is suspected that a better knowledge extraction method needs to be used such that important probes can be isolated out from the principal components. A list of all genes extracted by PLS-DA and RFE can be found in the appendix section.

4.3.2 Assessing Gene Data

After a decision is made on the genes we wish to inspect, a comparison is done between each the five classes in our classification task. From literature[36], we found out that the type 1 to 5 in our dataset represents lung adenocarcinomas (ADEN) , NORMAL, small-cell lung carcinomas (SCLS), squamous cell lung carcinomas (SQUA) and pulmonary carcinoids (COID) respectively. four out of the five classes are variants of lung cancer. To formally assess the genes we selected, we extracted the value of each group, and we compared the values using their mean and standard deviation. The comparison is shown in Table 4.2.

	ADEN	SQUA	COID	SCLC	NORMAL
TFAP2C	38.17 ± 34.44	49.82 ± 38.24	15.47 ± 26.9	2.23 ± 18.59	6.76 ± 13.36
PRDX1	1007.42 ± 399.98	1224.83 ± 491.89	424.16 ± 124.64	708.42 ± 261.58	882.76 ± 94.75
Samples	139	21	20	6	17

Table 4.2: Database Values of the Selected Genes

We can clearly see that the class of patients that does not have lung cancer has the lowest standard deviation out of all the classes. To assess the information captured from our data, we compared our values against the behaviours shown by the cancer variants. Out of the four variants, SCLC is the smallest and the most moble variant, and this could explain why our data shows a and a low activity but a strong variability. ADEN and SQUA are Non-small-cell lung carcinomas (NSCLC), NSCLCs are very large cancer cells, therefore a stronger intensity could be expected from the two variants. Our last variant, COID, can be classified into two types: typical and atypical carcinoids. Typical carcinoids tend to have a slower growth rate and spread rate, and vice versa for atypical carcinoids. Since the two types has two completely different behaviours, which explains its mean value and the high standard deviation shown.

4.4 Conclusion

In this section, we evaluated our model's accuracy on extracting biological knowledge from the microarray datasets. Our case study on Lung cancer shows that PLS-DA and RFE were not only just highly performing feature engineering techniques, they were also able to extract relevant molecular probes from the dataset. Further analysis on two promising genes, TFAP2C and PRDX1, has also shown that they were able to capture the behaviours of the four cancer variants - ADEN, SQUA, COID and SCLC. Due to the reasons mentioned, we can conclude that PLS-DA and RFE are methods that biologists could use to extract knowledge from large biological datasets.

5 Conclusion

5.1 Conclusion

To summarize, feature engineering techniques should be considered as an integral part of high throughput omics datasets analysis. We concluded that feature transformation techniques are able to encapsulate most of the insights from the dataset using a few components, and feature selection techniques are also able to summarise the whole dataset using a few selected samples.

In terms of classifiers, SVM was able to perform exceptionally with paired with all four of our feature engineering techniques, RF is another stable yet highly performing classifier to consider for analysing biological data. The results also suggest avoiding GNB, as its algorithmic simplicity does not allow it to capture the complex structures buried in our data.

In terms of feature engineering techniques, although all techniques were able to keep or increase the performance and reduce the training time needed of the classifiers, RFE stood out the most compared to the other techniques based on its excellent performance and high feature interpretability. We also recommend PLS-DA, although it has a slightly worse accuracy and the features are harder to extract, it compensates this by being several magnitudes faster than RFE. Out of the four techniques, NMF performed the worst. The latent information

captured by it could not be translated back into the original features, and the performance it acquired was the lowest in general. Therefore, we would strongly advise practitioners to avoid NMF for microarray analysis.

Based on our findings, when biologists are faced with the task of analysing real-world biological datasets, we would recommend one to use the SVM-RFE, or RF-RFE pipeline if performance were to be prioritised. If not, SVM-PLS would be a good compromise between performance and computational cost.

5.2 Future Work

In this paper, we dove into the intricacies of four well-known feature engineering techniques and classifiers. But there are still plenty of feature engineering techniques and classifiers that we did not cover in our scope. In our studies, experiments on several different ideas were not conducted due to our limited time constraints and its cost. Since grid search is very computationally intensive, the experiment would take about three days to run for all datasets.

Future work can focus its analysis on deep learning, which has been an active field of research in bioinformatics, models such as neural networks has shown high promises in both its classification performance and its ability to adapt to huge datasets. On the other hand, instead of arbitrarily defining the number of genes that we select, machine learning methods can be employed to decide the optimal number of genes automatically, which could potentially improve the performance of the datasets.

Extra work can also be done on balancing the distributions on each dataset, such that the genes selected using different folds will always remain the same. Further studies can also be done on different datasets to reinforce our finding's integrity. More case studies could be done on different microarray databases. If a larger subset of genes were extracted on more feature engineering methods, a more definitive comparison and explanation could be established. Last but not least, extra validation work could be done on the genes selected from our feature

engineering techniques. Visualisation techniques such as signature induced networks is a great tool for spotting the relationships between genes.

A
Appendix

Availability of data:

The datasets we used in our experiments are available at the following sources:

Prostate Cancer: https://ico2s.org/datasets/microarray.html

Rest of the datasets: http://csse.szu.edu.cn/staff/zhuzx/Datasets.html

PLS-DA				RFE			
SVM	GNB	KNN	RF	SVM	GNB	KNN	RF
NOS2	**NOS2**	**NOS2**	**NOS2**	POLG	POLG	POLG	POLG
STARD13	**STARD13**	**STARD13**	**STARD13**	**TGFBR3**	LOC100506403	LOC100506403	LOC100506403
SDC1	**SDC1**	**SDC1**	**SDC1**	LOC100506403	HIST1H2BG	HIST1H2BG	HIST1H2BG
KRT6A	**KRT6A**	**KRT6A**	**KRT6A**	HIST1H2BG	TUBB8	TUBB8	TUBB8
PPP1R26	**PPP1R26**	**PPP1R26**	**PPP1R26**	TUBB8	**LGMN**	**LGMN**	**LGMN**
USP8	USP8	USP8	USP8	**LGMN**	**CKMT1B**	**CKMT1B**	**CKMT1B**
TFAP2C	**TFAP2C**	**TFAP2C**	**TFAP2C**	**CKMT1B**	H1FX	H1FX	H1FX
HERC2P2	HERC2P2	HERC2P2	HERC2P2	H1FX	**PDE2A**	**PDE2A**	**PDE2A**
CNP	**CNP**	**CNP**	**CNP**	**PDE2A**	TCEB3	TCEB3	TCEB3
				TCEB3	ALG8	ALG8	ALG8
				ALG8	**NPR1**	**NPR1**	**NPR1**
				NPR1	**TPMT**	**TPMT**	**TPMT**
				TPMT	HEG1	HEG1	HEG1
				HEG1	FAM30A	FAM30A	FAM30A
				FAM30A	**MVD**	**MVD**	**MVD**
				MVD	**KDM4C**	**KDM4C**	**KDM4C**
				KDM4C	**ACTR3**	**ACTR3**	**ACTR3**
				ACTR3	TTC22	TTC22	TTC22
				TTC22	**NCOA1**	**NCOA1**	**NCOA1**
				NCOA1	**CAPNS1**	**CAPNS1**	**CAPNS1**
				CAPNS1	**SPOCK2**	**SPOCK2**	**SPOCK2**
				SPOCK2	MFHAS1	MFHAS1	MFHAS1
				MFHAS1	**TBX19**	**TBX19**	**TBX19**
				TBX19	**NR2C2**	**NR2C2**	**NR2C2**
				NR2C2	**CAPZB**	**CAPZB**	**CAPZB**
				CAPZB	**MDN1**	**MDN1**	**MDN1**
				MDN1	**RAB5B**	**GAMT**	**RAB5B**
				GAMT	**GAMT**	**GALK2**	**GAMT**
				GALK2	**GALK2**	**EGR2**	**GALK2**
				EGR2	**EGR2**	**PRKD3**	**EGR2**
				PRKD3	**PRKD3**	**CAPG**	**PRKD3**
				CAPG	**CAPG**	**ISLR**	**CAPG**
				ISLR	**ISLR**	**COL13A1**	**ISLR**
				COL13A1	**COL13A1**	**SOCS2**	**COL13A1**
				SOCS2	**SOCS2**	**ING2**	**SOCS2**
				ING2	**ING2**	**STATH**	**ING2**
				STATH	**STATH**	**TFAP2C**	**STATH**
				TFAP2C	**TFAP2C**	**PRDX1**	**TFAP2C**
				PRDX1	**PRDX1**	**KITLG**	**PRDX1**
				KITLG	**KITLG**	**RND2**	**KITLG**
				RND2	**RND2**	**CNP**	**RND2**
				CNP	**CNP**	**MX2**	**CNP**
				MX2	**MX2**		**MX2**

Table A.1: Gene selected using PLS-DA and RFE.
Genes that matches the online databases are marked in bold.

References

[1] Zachary Stephens et al. “Big Data: Astronomical or Genomical?” In: *PLoS biology* (2015).

[2] Govindarajan, Kaliyappan Duraiyan, and Palanisamy. “Microarray and its applications”. In: *J Pharm Bioallied Sci.* (2012).

[3] Ardeshir Bayat. “Bioinformatics”. In: *BMJ.* (2002).

[4] Sowjanya Latha. “Feature Extraction Mechanisms in Bioinformatics data”. In: *Proceedings of International Conference on Recent trends in Business*. 2011.

[5] Udayan Khurana, Horst Samulowitz Fatemeh Nargesian, and Deepak Turaga Elias Khalil. “Automating Feature Engineering”. In: *NIPS2016* (2016).

[6] Francisco Escolano. *Information theory in computer vision and pattern recognition*. Springer, 2009, pp. 211–269.

[7] Z. Hira and D. Gillies. “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data”. In: *Advances in Bioinformatics* (2015), pp. 1–13.

[8] M. Ringner. “What Is Principal Component Analysis?” In: *Nat Biotechnol* 26 (2008). URL: https://doi.org/10.1038/nbt0308-303.

[9] ZhiHu. *Illustration of the PCA model*. 2018. URL: https://zhuanlan.zhihu.com/p/32665213.

[10] R. Lazcano et al. “Parallelism exploitation of a PCA algorithm for hyperspectral images using RVC-CAL”. In: *Proc. SPIE 10007*. High-Performance Computing in Geoscience and Remote Sensing, Oct. 2016, p. 5.

[11] D. Pirouz. *An Overview of Partial Least Squares*. SSRN Electronic Journal, 2006.

[12] R. Rosipal. “Overview And Some Aspects Of Partial Least Squares”. 2005.

[13] H. Yin, J. Costa, and G. Barreto. “Intelligent Data Engineering And Automated Learning”. In: *IDEAL 2009* (2009), pp. 300–302.

[14] J. Kim, Y. He, and H. Park. “Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework”. In: *Journal of Global Optimization* 58 (2013), p. 2.

[15] Wikipedia. *Illustration of NMF*. 2013. URL: https://en.wikipedia.org/wiki/Non-negative_matrix_factorization#/media/File:NMF.png.

[16] H. Sanz et al. “SVM-RFE: selection and visualization of the most relevant features through non-linear kernels”. In: *BMC Bioinformatics* 19 (2018).

[17] Ravishankar Hariharan and Madhavan Radhika. "Recursive feature elimination for biomarker discovery in resting-state functional connectivity". In: *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2020), p. 4072.

[18] Expert System Team. *What is Machine Learning? A definition*. 2020. URL: `https://expertsystem.com/machine-learning-definition/`.

[19] Sciencedirect.com. *Supervised Learning - An Overview*. 2020. URL: `https://www.sciencedirect.com/topics/computer-science/supervised-learning`.

[20] M. Rouse. *What Is Supervised Learning?* SearchEnterpriseAI, 2019. URL: `https://searchenterpriseai.techtarget.com/definition/supervised-learning`.

[21] R. Sutton. "Introduction To Reinforcement Learning". In: (), pp. 15–20.

[22] C. Vanitha, D. Devaraj, and M. Venkatesulu. "Gene Expression Data Classification Using Support Vector Machine and Mutual Information-based Gene Selection". In: *Procedia Computer Science* 47 (2015), pp. 13–21.

[23] P. Feng et al. "Naive Bayes Classifier with Feature Selection to Identify Phage Virion Proteins". In: *Computational and Mathematical Methods in Medicine* (2013), pp. 1–6.

[24] Z. Yao and W. Ruzzo. "A Regression-based K nearest neighbor algorithm for gene function prediction from heterogeneous data". In: *BMC Bioinformatics* 7 (2006).

[25] Y. S. Ong Zexuan Zhu and M. Dash. "Markov Blanket-Embedded Genetic Algorithm for Gene Selection". In: *Pattern Recognition* 49.11 (2007).

[26] Enrico Glaab et al. "Using Rule-Based Machine Learning for Candidate Disease Gene Prioritization and Sample Classification of Cancer Gene Expression Data". In: *PLoS ONE* 7.7 (July 2012), e39932.

[27] Francisco Herrera Salvador Garcıa. "An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons". In: *Journal of Machine Learning Research* 9 (2008).

[28] Huang Da Wei et al. "The DAVID Gene Functional Classification Tool: A Novel Biological Module-Centric Algorithm to Functionally Analyze Large Gene Lists." In: *Genome biology* 8 (2007).

[29] A. Hudson. *Advanced Data Mining Technologies In Bioinformatics*. US: Tritech Digital Media, 2018.

[30] Noa Rappaport et al. *MalaCards: A Comprehensive Automatically-Mined Database of Human Diseases*. 2014.

[31] M. Magrane and U. Consortium. *Uniprot knowledgebase: a hub of integrated protein data*. Database. 2011.

[32] "Orphanet". In: *Orphanet: an Online Database of Rare Diseases and Orphan Drugs* 1997 (1997). URL: `http://www.orpha.net`.

[33] Allan Peter Davis et al. "The Comparative Toxicogenomics Database: update 2019". In: *Nucleic Acids Research* 47 (Jan. 2019), pp. D948–D954.

[34] 2020. URL: `https://www.genecards.org/cgi-bin/carddisp.pl?gene=TFAP2C`.

[35] 2020. URL: `http://ctdbase.org/detail.go?type=gene&acc=5052`.

[36] 2020. URL:
http://leo.ugr.es/elvira/DBCRepository/LungCancer/LungCancer-Harvard1.html.