Comparison of binary classification and feature selection methods for bio-medical data

ST443 - Machine Learning and Data Mining
London School of Economics and Political Science
London, United Kingdom

Candidate Numbers:
37984
37724
49872
45251

Group 17

Abstract

This paper considers two case studies for the comparison of various binary classification and feature selection methods. The first case study evaluates the performance of Linear Discriminant Analysis (LDA), Logistic, Quadratic Discriminant Analysis (QDA, *k* Nearest Neighbour (*k*-NN), Gradient Boosting Decision Trees (GBDT), Random Forest (RF), and Support Vector Classifiers (SVC) for immune cell classification based on RNA expression levels in genes. The results indicate that the LDA, GBDT and SVC models have the best performance for this data. Based on these results, we trained three improved classifiers, the L2 Logistic, Stacked, and Weighted Voting Classifiers with the aim of optimizing their F1-scores. The results indicate that all three models, combined with their optimal training data structures, improve the F1-scores compared to the baseline models. In the end, the Weighted Voting Classifier, combining Support Vector, Logistic, and Gradient Boosting Decision Tree Classifiers that were trained on 200 principal components resulted in the best F1-score. For the second case study, we perform feature selection and binary classification on a high-dimensional imbalanced dataset containing three-dimensional molecule properties. We consider six feature selection methods, namely Forward Selection with Logistic Classifier, Elastic Net, Recursive Feature Elimination with Random Forests and Logistic Classifiers, a Mutual Information Classifier, and a Random Forest Classifier with a SelectFromModel feature selection. Based on the selected features for each approach, we train QDA, k-NN, Logistic, Random Forest, LDA, and SVC for each feature selection method, and select the classifier that performs best. The results indicate that the highest balanced accuracy is obtained with a QDA classifier based on 50 features selected by Forward Selection with a Logistic Classifier.

1 Introduction

Binary data classification is a common task in machine learning that can be applied in various settings, including customer churn analysis, document categorization, and medical diagnosis [Yin et al., 2013, Zeng et al., 2011]. With the rise of big-data systems, innovations in data generation and computer science techniques allow us to collect more complex data, resulting in high-dimensional data, where the number of features is higher than the number of observations. High-dimensionality is a common

Preprint. Under review.

problem in bio-medical settings, for example for gene and molecule-related data, as advances in technology now allow for gene expression measurements in individual cells. This paper considers two case studies. The first case study considers data, where the number of features is smaller, but relatively close to the number of observations, and focuses on binary classification of immune cells based on RNA expression levels in genes. For this dataset, we would like to answer the following questions:

1. How do the performance of LDA, Logistic, QDA, *k*NN, GBDT, RF, and SVC classifiers compare for various evaluation metrics on the full dataset, and on the 10 principal components?
2. Based on the baseline classification models, we developed three models with the goal of improving their F1-score. How do the performance of the L2 Logistic, Stacking, and Weighted Voting Classifiers compare for the F1-score and other evaluation metrics?

To answer these questions, we train and evaluate seven binary classifiers; Linear Discriminant Analysis (LDA), Logistic Classifier, Quadratic Discriminant Analysis (QDA), *k* Nearest Neighbour Classifier (*k*NN), Gradient Boosting Decision Trees (GBDT), Random Forests, and Support Vector Machine (SVC). We evaluate these classifiers both on the full dataset and the ten principal components summarizing the features. Based on this performance of the classifiers, we construct three improved models, the L2 Logistic, Stacking, and Weighted Voting Classifiers (WVC), with the goal of improving their $F1$-score.

The second case study focuses on binary classifiers combined with feature selection methods for a high-dimensional dataset describing three-dimensional properties of a molecule that can be used to distinguish binding compounds from nonbinding ones. This data is not only high-dimensional, but also relatively sparse and there is a class imbalance. The latter refers to the situation where the majority class has significantly more data points than the minority class [Yin et al., 2013]. In high-dimensional class-imbalanced datasets, feature selection is quite important, because high-dimensional data decreases the efficiency of many classifiers, also known as the curse of dimensionality [Bhavani et al., 2008, Yin et al., 2013]. However, feature selection is a complicated problem, and finding the optimal subset of variables is considered to be NP-hard [Maldonado et al., 2014].

For this dataset, six feature selection methods are applied to select features. Based on these selected features we train the QDA, k-NN, Logistic, Random Forest, LDA, and SVC classifiers and select the best performing classifier, based on the balanced accuracy metric. The feature selection methods include Forward Selection with Logistic Classifier (Forward Logistic), Elastic Net, Recursive Feature Elimination with Random Forests (RFE-RF) and Logistic (RFE-LOG) Classifiers, a Mutual Information (MI) Classifier, and a Random Forest Classifier with a SelectFromModel feature selection (SFM-RF). Our research aims to answer the following research questions for this second case study:

1. Using Forward Logistic, Elastic Net, RFE-Log, RFE-RF, MI, and SFM-RF feature selection methods, how do we obtain the highest balanced accuracy??

The rest of this paper is structured as follows. In Section 2, we provide descriptions and summary information on the datasets used in both case studies. In Section 3, we introduce the classifiers for both case studies, the training approaches are explained. Finally, Section 4 highlights the results of our analysis and Section 5 answers the research questions and highlights some limitations of our research.

2 Data Description

In this Section, we introduce the two datasets. First we focus on the cell classification case, for which we use a subset from the dataset used by Suo et al. [2022]. The dataset contains 5471 observations and 4124 features, and there are no values missing, where each observation correspond to a cell belonging to either the TREG or CD4+T class. CD4+T cells are a class of immune cells within the human body. TREG cells are a subclass of the CD4+T cells, which play an important role in controlling autoimmune diseases. 3356 observations belong to the CD4+T class, and there are 2115 cells classified as TREG. Even though the CD4+T class is a bit larger, we do not identify this dataset as imbalanced. In our analysis, we encode TREG as the positive class, and CD4+T as the negative class.

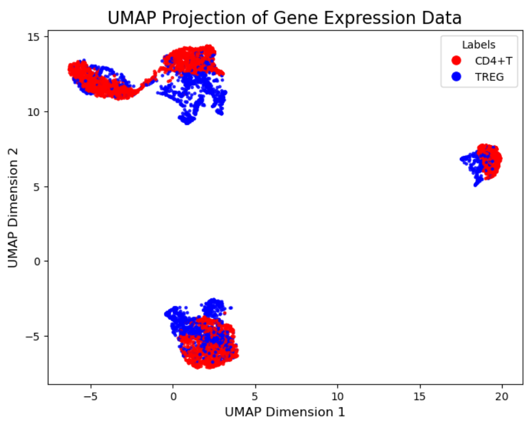

Figure 1: UMAP Projection of Gene Expression Data.

Figure 1 shows the two-dimensional UMAP projection of the RNA gene expression data. The plot aims to separate the two classes, but as shown in Figure 1, the classes are clustered together. This indicates that there are similarities in the gene expression patterns from both clusters, hence it's hard to visualise the data in 2D.

For the second case study, we use the DOROTHEA dataset which was used in the NIPS 2003 challenge [Guyon et al., 2004]. The dataset contains 800 observations on $100,000$ features and there are no values missing. Each observation has the label 1 or -1. The dataset is characterized by a large class imbalance, as 722 observations have the label -1, and 78 observations have label 1. This imbalance can significantly affect the model performance, as many classification models tend to favor the majority class. All features are binary vectors, implying that they can only take values 0 or 1. As shown in Figure 2, all features contain large fractions of zeros, making the data relatively scarce. Scarcity in the data can negatively affect classifiers in producing accurate predictions Bissmark and Wärnling [2017].

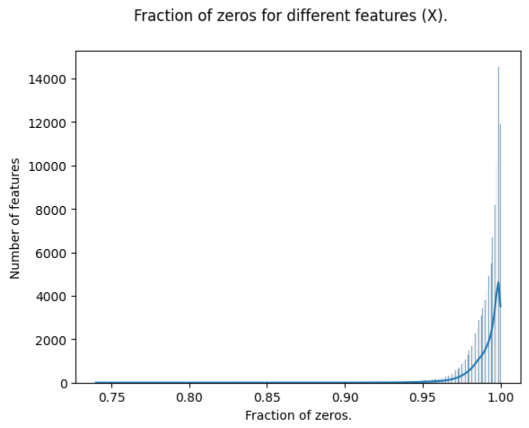

Figure 2: Count plot showing the presence of zero's in the features.

For the cell classification problem, we standardize the data using StandardScaler() in Python. Standardization removes the potential effect that features with large value ranges could have on various computations, including Euclidean distance. After standardization, PCA and classifiers depending on the Euclidean distance, such as the *k*NN classifier, perform better. For the molecule property dataset, we do not have to standardize the data, as the features are binary vectors already.

3 Methodology

In this Section, we will introduce the methodology used in this paper. Similar to the previous Section, the methodology of the cell classification case study is discussed first, after which we elaborate on the approach for the feature selection case study.

3.1 Cell Classification

For the this case study, we consider seven baseline classification models: Linear Discriminant Analysis (LDA), Logistic Classifier, Quadratic Discriminant Analysis (QDA), *k* Nearest Neighbour Classifier (*k*NN), Gradient Boosting Decision Trees (GBDT), Random Forests, and Support Vector Machine (SVC). As each of these methods were introduced within the course, we omit their introduction below, but this can be found in Appendix 5. To train and evaluate these models, we split the dataset in training (80%) and testing (20%) sets. Besides training the model on the training data set, we perform a principal component analysis on the training data to obtain ten principal components. Principal Component Analysis (PCA) selects the eigenvectors of the sample covariance matrix $\hat{\Sigma} = \frac{1}{n} X X^T$ that summarize most of the variability in the original data. PCA reduces the dimensionality of the data significantly, which can reduce the computational complexity of implementing the classification models. Hence, for each classification model, we obtain one model trained on the training set, and one model trained on the ten principal components, and the model performances are evaluated on the accuracy, balanced accuracy, AUC and F1 score metrics for their performance on classifying the testing set.

Based on the results of the baseline models for these metrics, we create three improved models, a Logistic Regression with L2 regularization (L2 Logistic Classifier), a stacking method with random forest and logistic regression (Stacking Classifier), and a Weighted Voting Classifier with Support Vectors, Gradient Boosting Classifiers and Logistic Regression (WVC). We train these models with the aim of improving the F1-scores. These models are explained below.

L2 Logistic Classifier In Section 3.1, the Logistic classifier was introduced. The L2 Logistic Classifier is based on the same log-likelihood function as in 8, but adds an L2 penalty term to this formula, which penalizes large coefficients:

$$l(\beta_1, ..., \beta_{K-1}) = \sum_{i=1}^{n} log(\frac{e^{\beta_{Y_i}^T X_i}}{1 + \sum_{k'=1}^{K-1} e^{\beta_{k'}^T X_i}}) + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{1}$$

where λ is the regularization parameter, which we estimate through a grid search (see notebook). The L2 penalty can decrease the variance of the model, as coefficients that otherwise might inflate are now restricted, but increases the bias of the model, as the penalty term added to the model increases.

Stacking Method The second improvement model is a stacking method combining Random Forest and Logistic Classification models. Stacking combines estimators to reduce their biases Wolpert [1992]. First, we introduce the stacking method, after which we explain our choice for the Random Forest and Logistic Classification models. The explanation and notation of the stacking method are based on Džeroski and Ženko [2004].

Stacking involves the process of combining multiple classifiers obtained with different learning algorithms $L_1, ..., L_J$, where J is the number of learning algorithms considered. We consider $J = 2$, where the learning algorithms are Random Forest and Logistic Classifier. Using training data $(X_1, Y_1), ..., (X_n, Y_n)$, we obtain base-level classifiers C_1 and C_2, where $C_j = L_j(X, Y)$. After this, a meta-level classifier combining the output of the base classifiers is trained. To train the meta-level classifier, we first need to generate data for this model. We perform a 5-fold cross-validation. For each fold, we train all base-level classifiers on the training data excluding the leave-out set, and use the trained base-level classifiers to obtain predictions for the labels of the leave-out set, $\hat{y}_i^j = C_j^i(x_i)$, for $i \in$ leave-out set, $j \in 1, ...J$. Then, for each fold, the meta-level data consists of $(\hat{y}_i^j, ..., \hat{y}_i^J, y_i)$ for $i \in$ leave-out set, where the covariates are the predicted classes from the base-level classifiers, and the dependent variable is the true class. This data is used by the meta-level classifier, in our case the Logistic Classifier, to combine these predictions into one final prediction.

Therefore, the final classifier can be defined as:

$$\psi^{Stacking}(x) := \arg\max_{\mathbf{k}} \beta_k^T \hat{y} \tag{2}$$

with β_k^T being a (J+1) dimensional coefficient vector and $\hat{y} = (1, \hat{y}^j, ..., \hat{y}^J)$ includes one intercept term.

We have chosen for Random Forest and Logistic Classifier base models, and the Logistic Classifier as the meta model. Random Forests are robust to redundant features and can handle non-linear relationships and complex interactions between features. The latter is especially convenient for the RNA expression data in this case study, as this data potentially has intricate gene relationships. The Logistic classifier is able to handle high-dimensional data and performs well when the relationships between the label and explanatory features is linear. Furthermore, it is computationally efficient. The Logistic Classifier and Random Forests complement each other well, as Random Forest compensates for the fact that the Logistic Classifier cannot capture non-linear relations, while the Logistic Regression can better follow simpler relation structures than Random Forests. The Logistic Classifier is also chosen as the meta-model, because this classifier ensures that the stacking of the base-models remains relatively simple and interpretable.

As this stacked model includes various layers, it is prone to overfitting and a high number of explanatory variables in the original training data can contribute to this. Therefore, we did not use the full training dataset for this model, but instead we first computed the first 50 principal components, and used these as dataset for this stacked model.

Weighted Voting Classifier Finally, we use a Weighted Voting Classifier with soft voting rules and weighted contributions that combines a Support Vector Classifier, Logistic Regression, and Gradient Boosting Classifier. A weighted voting classifier is an ensembling method that combines several classifiers and computes the average predicted probabilities, which it uses to predict the class labels. We introduce the Weighted Voting Classifier based on the framework introduced by Kuncheva and Rodríguez [2014]. Assume we want to combine classifiers $C_1, ..., C_J$, where J is the total number of classifiers to combine, and each classifier is individually trained. For the Weighted Voting Classifier, we use the same training data as for the baseline models. For each point, $x \in \{x_1, ..., x_m\}$ in the testing set, we can compute the predicted class probabilities $P(s_j = k|C_j, x)$, where $(s_1, ..., s_J)$ are the labels predicted by $C_1, ..., C_J$, where each $s_j \in \{1, ..., K\}$. In this case, the Weighted Voting Classifier can be defined as:

$$\psi^{WVC}(x) := \arg\max_{\mathbf{k}} \frac{1}{\sum_{j=1}^{J} w_j} \sum_{j=1}^{J} w_j P(s_j = k|C_j, x) \tag{3}$$

where $w_j,\ j \in \{1, ..., J\}$ are the weights assigned to each classifier $C_1, ..., C_J$.

We have chosen three baseline models to be combined in the Weighted Voting Classifier: the Support Vector Classifier, Logistic Classifier, and Gradient Boosting Decision Tree classifier. Each model has its unique strenghts, implying that they can complement each other effectively in this ensembling method. The Support Vector Classifier is reliable for linearly separable patterns and can capture patterns that are missed by other models focusing on non-linear trends or overfitting. The Logistic Classifier performs well for linear classification boundaries too. On the other hand, the Gradient Boosting Decision Tree classifier captures non-linear trends which are missed by the other two models. The GBDT Classifier adds robustness to the ensemble, as the model is able to capture patterns that depend on high-order feature interactions. The weights assigned to the SVC, Logistic and GBDT Classifiers are $3, 2, 1$ respectively, corresponding to their individual performance as baseline models (see Section **??**. As the goal of the improved model is to optimize the F1-score of the classifier, we first train the model using the training dataset consisting of 80% of the available data. Then, to further improve the F1-score, the Weighted Voting Classifier is trained on PCA data for several number of selected components. Then, the model resulting in the highest F1-score is selected.

Once we have evaluated the L2 Logistic, Stacking and Weighted Voting Classifiers, we evaluate their performance for improving the F1-score and choose the function with the highest F1-score. This function is then trained on the initial full dataset provided for this case study, such that it can make predictions on unseen test datasets.

3.2 Molecule Property Selection

For the second case study, we perform binary classification after we have reduced the dimensionality by performing feature selection methods. In this Section, we briefly introduce the methods and explain the how the methods are implemented.

The first feature selection method that we consider is Forward Stepwise Selection (FSS). FSS performs a greedy search, where we start with a model without features, and then add the feature that reduces the test error the most. Now that we have the model with this feature, we look at the $p - 1$ features that are left, and we add the feature that now produces the lowest test error. This process continues, until a stopping criterion is reached, which in our case is a pre-specified number of features to be selected.

Performing Forward Stepwise Selection on the entire dataset is computationally intensive due to the large feature set. To address this, we first train a Random Forest classifier with 100 estimators on the training data. Using the model's feature importance scores, we narrow the feature set down to the top 500 most important features. We then apply a forward selection process using a Logistic Regression model on this reduced subset to identify the 50 most significant features.

The elastic net method, introduced by Zou and Hastie [2005], introduces a penalty of the form

$$\sum_{j=1}^{p}(\alpha|\beta_j| + (1-\alpha)\beta_j^2) \tag{4}$$

which can be implemented in linear classification models. The first term shrinks some of the coefficients to zero, making elastic net a feature selection method, while the second term restricts the size of the coefficients, which can average the effect of highly-correlated features. In high-dimensional settings, using the Elastic Net penalty is preferred over using an L1 penalty term alone, as the latter can only select n features, n being the number of observations, while Elastic Net can select more than n features Hastie [2009]. On the other hand, only using the L2 penalty term would not result in any selected features, as the L2 penalty only restricts the coefficients and does not shrink them to zero. For the Elastic Net method, we apply a 80%-20% split to the dataset First, we apply 3-fold cross-validation to tune the hyper-parameter α. We select the features that have a non-zero coefficient in the fitted model.

Next, we introduce Recursive Feature Elimination (RFE). This procedure starts with a classifier fitting the data with all available features, which ranks the features based on some importance measure obtained through the classifier. Then, it removes the features with the lowest ranks, and fits the classifier again. This procedure continues until a pre-specified number of features remains [Granitto et al., 2006]. As with Forward Stepwise Selection, we shrink the included features to 50 features. The importance measure used to rank the features depends on the underlying classification model, for which we use a Random Forest and Logistic Classifiers.

When applying RFE to Random Forests, the feature importance measure is constructed as follows Granitto et al. [2006]. For any tree estimated in an Random Forest, not all the training data is used, as the tree is fitted on a bootstrap sample. These remaining training data points are called out-of-bag (OOB) subsets. These subsets can give unbiased measures of the prediction error. The relative importance of each feature is then defined as follows: for each feature, we compare the OOB prediction error on the regular data, compared to the OOB prediction error when this one specific feature is shuffled. The rationale behind this approach is that for irrelevant features, the difference between the prediction errors is not large, while for relevant features, this may cause a difference between the predictions on the original and shuffled data sets. As this importance measuring approach in RF-RFE is part of the Random Forest algorithm, the computational efforts are not increased Granitto et al. [2006].

When applying RFE in combination with the Logistic Classifier, the importance measure is defined as follows Zhu and Hastie [2004]:

$$\Delta P_j = \frac{1}{2}\frac{\delta^2 P}{\delta\hat{\beta}_j^2}\hat{\beta}_j^2 \tag{5}$$

where P is the loss function calculated on the training data, and $\hat{\beta}_j$ is the coefficient corresponding to feature j. As we consider a Logistic Classifier, the loss function P is the log-likelihood of a logistic model.

Finally, we apply the SelectFromModel (SFM) Method using a Random Forest. The SFM method is quite similar to RFE, as it also ranks the features based on some importance measure determined by an underlying classifier. Then, it computes the mean of the importance measures, and selects the features that have an importance score higher than the mean. For obtaining the importance measures, we use a Random Forest with 10 estimators as the base model. Using a small number of estimators strikes a balance between efficiency and effectiveness for feature selection, as even with just 10 trees, Random Forest can provide reliable feature importance rankings while avoiding unnecessary computational cost. Within the Random Forest model, a similar approach for computing the weights is used as in RFE.

Finally, We implement feature selection based on Mutual Information (MI). Mutual information measures the dependency between the feature and the class label. MI is defined as follows. Assume that we have $(X_1, Y_1), ..., (X_n, Y_n) \in \mathbb{R}^p \times \{1, ..., K\}$, which are $i.i.d$ realizations of an random variable $Z = (X, y)$, with distribution $\mu(x, y)$. Then, defining the marginal distributions of X and Y as $\mu_x(x) = \int dy\mu(x, y)$ and $\mu_y(y) = \int dx\mu(x, y)$, we can define Mutual Information as follows:

$$I(X, Y) = \int \int dxdy\mu(x, y)log(\frac{\mu(x, y)}{\mu_x(x)\mu_y(y)}) \tag{6}$$

One of the advantages of Mutual Information is that it is able to capture non-linear correlations when selecting features Yin et al. [2013]. We apply the mutual information function to the training data, store the resulting scores in a dataframe, and then sort them in descending order. From this sorted list, the top 100 features are selected for model training.

For each of the feature selection techniques introduced above, we implement a grid search to tune hyperparameters (see notebook) and determine which features will be selected. Then, for each feature selection approach, we estimate various classification models, QDA, *k*-NN, Logistic, Random Forest, LDA, and SVC, and identify the best-performing model based on balanced-accuracy.

4 Results

In this Section, we discuss the results that we have obtained per case study. Table 4 shows the performance metrics for the baseline classification models on the standardized data set and the ten principcal components.

Table 1: Performance of the baseline classification models for accuracy, balanced accuracy, AUC, and F1 evaluation metrics on the standardized dataset and 10 principal components.

	Standardized Data				PCA			
Model	Accuracy	Balanced Accuracy	AUC	F1	Accuracy	Balanced Accuracy	AUC	F1
LDA	0.685	0.684	0.716	0.622	0.921	0.901	0.986	0.888
Logistic	**0.946**	**0.939**	0.990	**0.928**	0.942	0.935	**0.988**	0.923
QDA	0.511	0.519	0.519	0.463	0.939	0.928	0.986	0.917
k-NN	0.709	0.660	0.704	0.545	0.918	0.900	0.958	0.884
GBDT	0.933	0.922	0.986	0.909	0.922	0.908	0.981	0.893
SVC	0.943	0.931	**0.989**	0.922	**0.949**	**0.940**	**0.988**	**0.931**
RF	0.913	0.891	0.983	0.875	0.922	0.910	0.977	0.894

Table 4 highlights the strong performance of Logistic Regression, Gradient Boosting Decision Trees, and Support Vector Classifier for the RNA expression dataset on the standardized dataset, which demonstrates the ability of these models to handle data where the dimension is close to the number of observations. The Logistic classifier has the highest metric values for accuracy, balanced accuracy, and F1-score, 0.946, 0.939, and 0.928 respectively. SVC has the highest value, 0.989, for the AUC metric. On the other hand, the performance of LDA, QDA, and *k*-NN are relatively poor for the standardized data, as shown in Table 4, where QDA has the lowest score for each performance metric.

Overall, the application of PCA improved the model performance compared to the performance on the standardized data set. Especially the performance of the weak classifiers in the standardized data, LDA, QDA, and *k*-NN, bringing them on par with the stronger baseline models. For example, the accuracy score of QDA improved from 0.511 for the standardized data set to 0.928 for the PCA analysis. Even for the models that already performed well on the standardized data, most individual evaluation metrics improved. The accuracy of the SVC increases from 0.943 to 0.949. These results support the idea that dimensionality reduction can effectively mitigate issues related to relatively high-dimensional data, such as overfitting.

An important note to be made is that the ten principal components explain only 9% of the variance in the original training data. To obtain an explained variance of 90%, we needed to run PCA with 2280 components, which was not feasible due to computational constrains. This indicates that a lot of information was lost in the dimension reduction, and therefore can form an explanation for the slight decrease in the performance of some of the individual methods, such as the Logistic Classifier.

Table 2: Performance of the L2 Logistic Classifier on the training data set, improved baseline models for accuracy, balanced accuracy, AUC, and F1 evaluation metrics.

Model	Accuracy	Balanced Accuracy	AUC	F1
L2 Logistic Regression	0.960	0.955	0.995	0.945
Stacking Method	0.969	0.964	0.994	0.957
Voting Classifier	**0.974**	**0.970**	**0.995**	**0.965**

Table 4 shows the performance of the L2 Logistic, Stacking, and Weighted Voting Classifiers. The goal of these models was to improve the F1-scores of the individual models. Note that each of these models was trained on different data to optimize the F1-scores of the models. L2 Logistic was trained on the training data, while the Stacking and Weighted Voting Classifiers were trained on 50 and 200 principal components respectively. The L2 Logistic regression improves the performance of the Logistic model on the train data (see Table 4 for all metrics. This indicates the L2 penalty term is effective by reducing the noise from less informative genes, which improves the performance metrics. The Stacking Classifier performs slightly better than the L2 Logistic Classifier, and also outperforms the individual Random Forest and Logistic Classifiers, especially on the F1-scores. This indicates that the Stacking Classifier is able to combine the strengths and smooth out the limitations of the individual models, resulting in better performance. Finally, Table 4 shows that the Voting Classifier obtains the best performance metrics. The combination of the SVC, Logistic, and GBDT Classifiers results in a balanced and flexible approach, both suitable for linear and non-linear decision boundaries. Combining this with the dimension reduction as the model is trained on PCA data with 50 components, this model obtains the strongest F1-score.

Table 4 shows the results for the second case study focusing on molecule property feature selection. Note that for Forward Selection, RFE, and Mutual Information approaches, the number of selected features was given as an input and not determined by the models themselves. Looking at the selected models, we see that, besides for MI and Elastic Net, relatively simple classification models are chosen after the feature selection was performed. Table 4 indicates that the Forward Selection method using a QDA classifier obtains the highest balanced accuracy of 0.951, while it also uses a minimal number of selected features compared with the other methods. Especially for the methods based on MI, SFM and Elastic Net, we see that the number of selected features is much higher, while the balanced accuracy is lower than for Forward Selection. This result aligns with the insight that including more features in your model does not always result in better performance.

Table 3: Balanced Accuracy scores for the best-performing classifier for each Feature Selection Method.

Feature Selection Method	# Selected Features	Classifier	Balanced Accuracy
Forward Selection	50	QDA	**0.951**
RFE with Random Forest	50	QDA	0.896
RFE with Logistic Regression	50	K-NN	0.900
Mutual Information	100	Logistic	0.892
SFM with Random Forest	421	KNN	0.871
Elastic Net	500	SVC	0.924

5 Conclusion

In this paper, we considered two case studies to compare binary classification models. For the first case study, we looked at cell classification based on RNA expression levels in genes. For this case study, we aimed to answer two questions:

1. How do the performance of LDA, Logistic, QDA, *k*NN, GBDT, RF, and SVC classifiers compare for various evaluation metrics on the full dataset, and on the 10 principal components?
2. Based on the baseline classification models, we developed three models with the goal of improving their F1-score. How do the performance of the L2 Logistic, Stacking, and Weighted Voting Classifiers compare for the accuracy, balanced accuracy, AUC and F1-scores?

The results indicate that for the first question, The Logistic, GBDT and SVC Classifiers have the best performances when trained both on the standardized dataset and the PCA data. LDA *k*-NN and especially QDA perform worse for the standardized dataset, but the dimension reduction offered by PCA improves their performance when trained on the PCA data. A potential explanation for the relatively bad performance of the LDA and QDA classifiers is that the signal-to-noise ratio might be relatively high in the data. Generally, we know that LDA and QDA are suitable when the signal-to-noise ratios in the data are low, such that the data can only support simple model classifiers, but here it seems that more complex and flexible models outperform these simple classifiers. Similarly *k*-NN is known for modeling highly irregular decision boundaries in the presence of noise, but here the classifier is outperformed by more complex models, which can be explained by a low presence of noise in the data.

For the second question, we conclude that the L2 Logistic, Stacking and WVC classifiers, combined with their respective optimal structures of the training data, the full training data, 50 and 200 PCA components respectively, all improve the performance metrics, and especially the F1-scores, of the individual baseline models. The L2 penalty term in the Logistic Classifier reduces the noise from less informative features, improving the performance compared to the Logistic Classifier. The Stacking and WVC each combine several individual classifiers, such that the combined classifiers leverage on the strength and smooth out the weaknesses of the individual classifiers, resulting in better performing classifiers. Overall, the WVC classifier, when trained on 200 principal components, results in the highest F1-score of 0.965.

One limitation of this research, is that for each improved model, we tried to optimize the F1-score, and in this process looked at the optimal format of the training data (full training data or with reduced dimensions). Therefore, the performance of the three improved models is not based on the same training data, which forms a limitation for the comparability of the results. For further research, it would be interesting to compare these models with other combined classifiers.

For the second case study, we looked at a high-dimensional, sparse and imbalanced dataset including molecule properties. For this case study, we aimed to answer the following question:

1. Using Forward Logistic, Elastic Net, RFE-Log, RFE-RF, MI, and SFM-RF feature selection methods, how do we obtain the highest balanced accuracy??

The results indicate that the best balanced accuracy is obtained for a QDA classifier fitted with 50 features selected by a Forward Stepwise feature selection method using a Logistic Classifier. This method obtained a balanced accuracy of 0.951. Additionally, the results show that classifiers based on feature selection methods that select a substantially larger number of features, such as MI, SFM-RF, and Elastic Net, do not obtain higher balanced accuracies.

One limitation for our research for the second case study is that we have not compared the feature selection methods and corresponding classifiers on the same number of features. The main reason for omitting this analysis is computational constraints, as for example, the computational complexity of Forward Selection Method increases fast with the number of features that needs to be selected. On the other hand, we cannot restrict the number of features to be selected by Elastic Net, which selects a high number of features, such that it becomes difficult to compare all the feature selection methods and corresponding balanced accuracies. Further research could try to compare these models for the same number of features.

A Explanation of Baseline Classification Models

Consider a situation in which we have observed training data $(X_1, Y_1), ..., (X_n, Y_n) \in \mathbb{R}^p \times \{1, ..., K\}$, which are $i.i.d.$ copies of $(X, Y) \sim P$, with P unknown. The **Logistic Classifier** is a linear classifier that assumes that the regression function $\eta_k = P(Y = k|X = x)$ follows a logistic model. This allows us to define log odds ratios, on which the logistic classifier is based:

$$\psi^{logit}(x) := \arg\max_{\mathbf{k}} \beta_k^T x \tag{7}$$

The coefficients $\beta_1, ..., \beta_K$ are typically estimated by maximizing the likelihood function shown below.

$$l(\beta_1, ..., \beta_{K-1}) = \sum_{i=1}^{n} log(\frac{e^{\beta_{Y_i}^T X_i}}{1 + \sum_{k'=1}^{K-1} e^{\beta_{k'}^T X_i}}) \tag{8}$$

The **Linear Discriminant Analysis** (LDA) classifier is defined as follows:

$$\psi^{LDA}(x) := \arg\max_{\mathbf{k}} log\hat{\pi}_k - \frac{1}{2}(x - \hat{\mu}_k)^T \hat{\Sigma}^{-1}(x - \hat{\mu}_k) \tag{9}$$

where $\hat{\pi}$, $\hat{\mu}_k$ and $\hat{\Sigma}$ are estimates based on the training sample. LDA is defined as the logarithmic transformation of the product of the prior distribution π_k and the conditional density $f_k = f(X|Y = k)$ for $k \in \{1, ...K\}$, which can be used to model the unknown distribution P. LDA, assumes that f_k follows a multivariate normal distribution, $N_p(\mu_k, \Sigma)$, where μ_k depends on k and all classes have the same covariance matrix Σ. The LDA classifier does not look linear in the second term, but once two label classes are being compared, the quadratic term cancels as all classes have the same covariance matrix $\hat{\Sigma}$.

Quadratic Discriminant Analysis (QDA) is a non-linear classifier, based on LDA. QDA, relaxes the covariance assumption by allowing the covariance matrix to differ between classes, implying $f_k \sim N_p(\mu_k, \Sigma_k)$. This results in a slightly different classifier compared to LDA:

$$\psi^{QDA}(x) := \arg\max_{\mathbf{k}} log(\hat{\pi}_k) - \frac{1}{2}(x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1}(x - \hat{\mu}_k) - \frac{1}{2}\log(\det \hat{\Sigma}_k) \tag{10}$$

where $\hat{\pi}$, $\hat{\mu}_k$ and $\hat{\Sigma}_k$ are estimates based on the training sample. As we need to estimate $\hat{\Sigma}_k$ for each class individually, QDA requires many more parameter estimations than LDA. QDA will therefore only improve LDA if we have enough data points to accurately estimate the $\hat{\Sigma}_k$'s, and the true Σ_k's are not too close to each other.

***k* Nearest Neighbour** (*k*-NN) classifier is a non-parametric classifier, where *k* defines the number of neighbours to be considered in the classifier, and not one of the classes k. If we reorder the training data, $(X_{(1)}, Y_{(1)}), ..., (X_{(n)}, Y_{(n)})$ such that $||X_{(1)} - x||_2 \leq ... \leq ||X_{(n)} - x||_2$, then we can define the *k*NN classifier as:

$$\psi^{kNN}(x) = \arg\max_{\mathbf{k}} \sum_{i=1}^{k} \mathbb{1}(Y_{(i)} = k) \tag{11}$$

The intuition behind this approach is that we classify a new point x based on the majority class of the *k* closest neighbours to x. In case of a tie, the class is chosen at random. In this approach, *k* is a hyper-parameter that needs to be tuned, which is chosen through cross-validation.

The **Random Forest** (RF) Classifier can be defined as follows:

$$\hat{f}_{RF} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{12}$$

The RF Classifier obtains B bootstrap estimates from the training data, and for each sample it constructs a decision tree. To make sure that those trees obtained for the different bootstrap samples are not too highly correlated, RF applies a random sample predictor, such that at each split in a tree, the model can only chose from m features to split the covariate space on, instead of the full set p. Tree-based approaches are likely to overfit the training data, resulting in high variance. Averaging several trees improves the accuracy of the model over using a single tree, and the random sample predictor reduces the variance of the average tree.

The idea behind the **Gradient Boosting Decision Tree** (GBDT) Classifier is similar, as it also averages trees. However, in GBDT, the trees are not all fully grown, but each added tree uses information from the previous trees by fitting the pseudo-residuals defined as negative gradients, which are based on the previously added tree. For each added model, GBDT aims to find the tree that minimizes the loss function:

$$L_m := \sum_{i=1}^{n}(r_{m,i} - T(x_i; \theta_m))^2 \tag{13}$$

where $r_{m,i}$ are the pseudo-residuals. This approach depends strongly on the choice of the loss function. As we are dealing with a classification problem, popular loss functions such as the squared residual are not appropriate. Therefore, we choose the multinomial deviance, also known as the negative log-likelihood:

$$L(f) = -\sum_{k=1}^{K} \mathbb{1} f_k(x) + log(\sum_{k=1}^{K} e^{f_k(x)}) \tag{14}$$

Finally, Support Vector Classifiers (SVC) search for hyperplanes that "optimally" separate the two classes. the SVC optimization problem can be written in the following form:

$$\min_{(\beta_0,\beta)\in\mathbb{R}X\mathbb{R}^p} \sum_{i=1}^{n} l_H(y_i, f(x; \beta_0, \beta)) + \lambda||\beta||_2^2 \tag{15}$$

where $l_H(y, f) = max\{1-yf, 0\}$ is the hinge loss function, and $\lambda \geq 0$ is a tuneable hyperparameter.

References

S Durga Bhavani, T Sobha Rani, and Raju S Bapi. Feature selection using correlation fractal dimension: Issues and applications in binary classification problems. *Applied Soft Computing*, 8 (1):555–563, 2008.

Johan Bissmark and Oscar Wärnling. The sparse data problem within classification algorithms: The effect of sparse data on the naïve bayes algorithm, 2017.

Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54:255–273, 2004.

Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and intelligent laboratory systems*, 83(2):83–90, 2006.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. *Advances in neural information processing systems*, 17, 2004.

Trevor Hastie. The elements of statistical learning: data mining, inference, and prediction, 2009.

Ludmila I Kuncheva and Juan J Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and information systems*, 38:259–275, 2014.

Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information sciences*, 286:228–246, 2014.

Chenqu Suo, Emma Dann, Issac Goh, Laura Jardine, Vitalii Kleshchevnikov, Jong-Eun Park, Rachel A Botting, Emily Stephenson, Justin Engelbert, Zewen Kelvin Tuong, et al. Mapping the developing human immune system across organs. *Science*, 376(6597):eabo0510, 2022.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Liuzhi Yin, Yong Ge, Keli Xiao, Xuehua Wang, and Xiaojun Quan. Feature selection for high-dimensional imbalanced data. *Neurocomputing*, 105:3–11, 2013.

Wencai Zeng, Jiong Jia, Zhonglong Zheng, Chenmao Xie, and Li Guo. A comparison study: Support vector machines for binary classification in machine learning. In *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, volume 3, pages 1621–1625. IEEE, 2011.

Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.