Task 1: Binary Classification

Abstract

The dataset contains RNA expression level measurements for p = 4123 genes across n = 5471 cells (a subset of the dataset from Suo et al. (2022)). The first column specifies the cell type (TREG or CD4+T), while each of the remaining columns records the logarithmically normalized RNA expression level for a specific gene. This is a classical high-dimensionality binary classification problem. The best classifier we find is an Elastic Net Logistic Regression with a test F1 score of 0.939. Considering the high-dimensional nature of the dataset, we find it appropriate to work with PCA-transformed data and this improves the accuracy of a few classifiers. In fact, all classifiers have F1 Scores > 0.9 and the differences are marginal. We also tune the hyperparameters for the more promising classifiers with a GridSearchCV and are, in some cases, able to improve our classifiers. Our key finding is that a relatively simple method like Logistic Regression outperforms other more complex procedures in this dataset.

1 Exploratory Data Analysis

We first notice that both classes are prevalent in the data set with $N_{TREG} = 2115$ and $N_{CD4+T} = 3356$, meaning there is no heavy skew and we do not have to carry out resampling techniques. We then answer a series of 5 questions which helps us frame our thinking on how the data looks and how we should formulate the rest of our approach.

1. Is there multicollinearity in the data set? Need full rank to have a reliable QDA classifier.

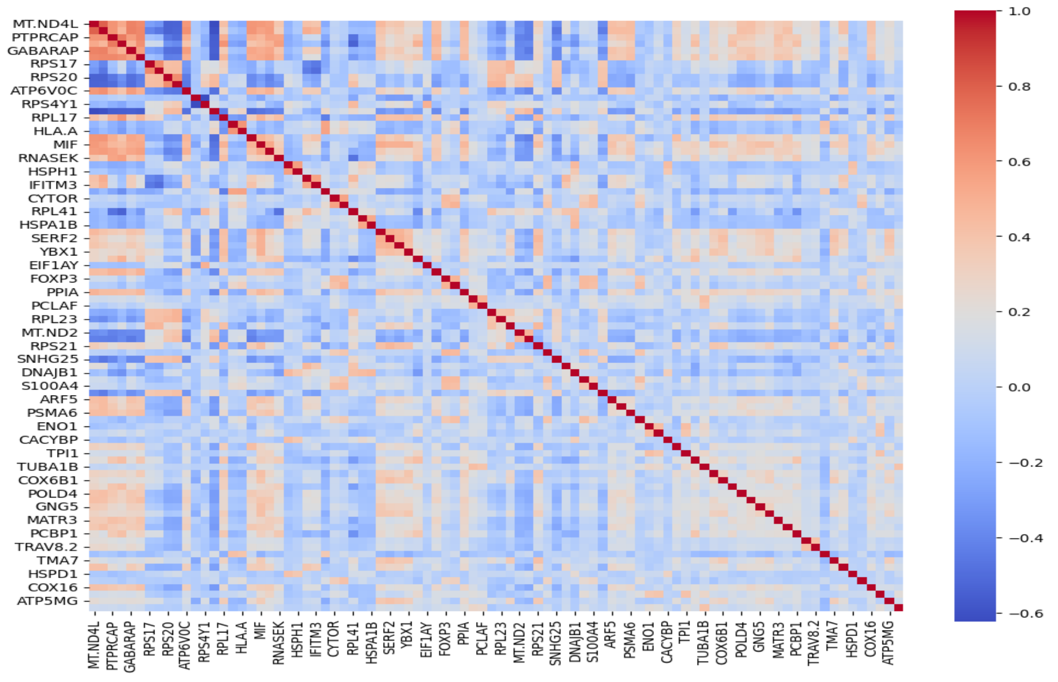

Figure 1: Almost no strong correlation between any variables

We set a threshold of Pearson coefficient > 0.4 before variables are even plotted in the above heatmap. Considering almost all the features are not present on this heatmap and even amongst those present the highest is around 0.7, we can conclude there is minimal mutli-collinearity in the data and we do not need to pursue clustering techniques/other methods to reduce multi-collinearity.

2. Are the classes separable with the current features?

With $p = 4123$, we need to explore dimensionality reduction of some form to have interpretable results. We first consider a Uniform Manifold Approximation and Projection (UMAP) visualisation (McInnes et al. (2018)). UMAP is a dimensionality reduction technique that helps visualize data by projecting it into a lower-dimensional space. It preserves both the global structure (how clusters relate to each other) and local structure (internal consistency within clusters) of the data.

Poor separability in the UMAP plot (Fig 2) informs us that we would need to try some method of feature engineering (eg. PCA) to improve our classifier. Nevertheless, we are still interested in exploring which specific genes might be important in classification. This leads us to our next question.

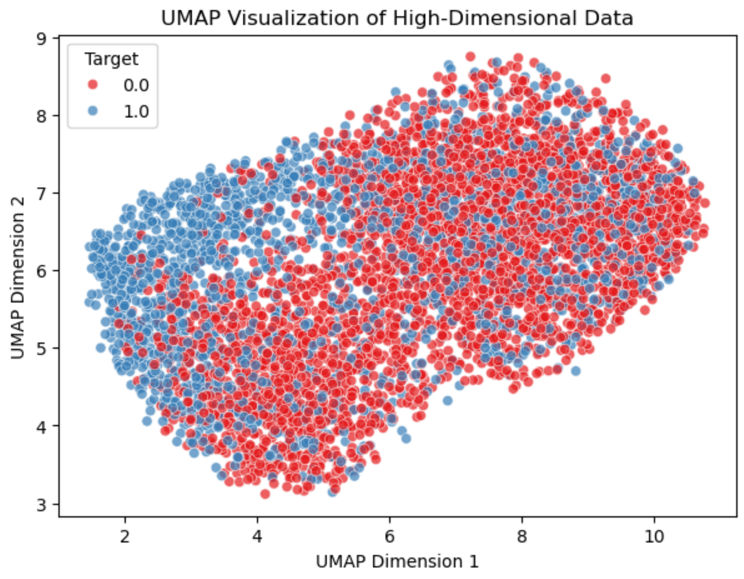

Figure 2: *Minimal separability across classes despite trying different hyperparameters.*

3. Which genes are important in classification? For a first pass at answering this question, we run a random forest and plot a variable importance measure.

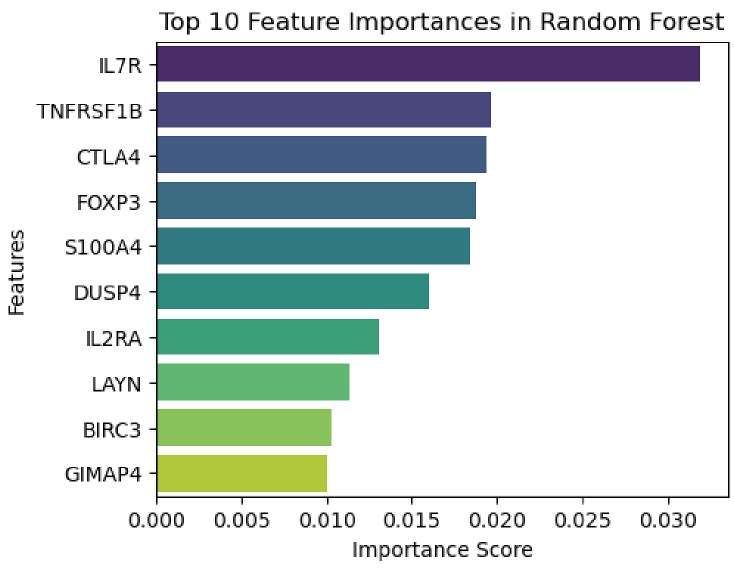

Figure 3: *IL7R is the most important feature but even then only has around 3% importance.*

Our next 2 questions look to further zoom in on specific genes and find the best "explanatory" genes. For both questions we plot boxplots to see if those individual features have much predictive power over the binary Y.

4. In our 10-component PCA, which genes turn out to be influential?

We define influential as the top 8 most common features across the top 25 highest absolute value loadings for each component. In other words these are the 8 features that are most influencing the nature of our principal components. This approach is in part inspired by the work of Jin et al. (2016) which looked into ranking features in PCA. We then plot boxplots of the top 8 features to see if we can they are good at separating both classes.

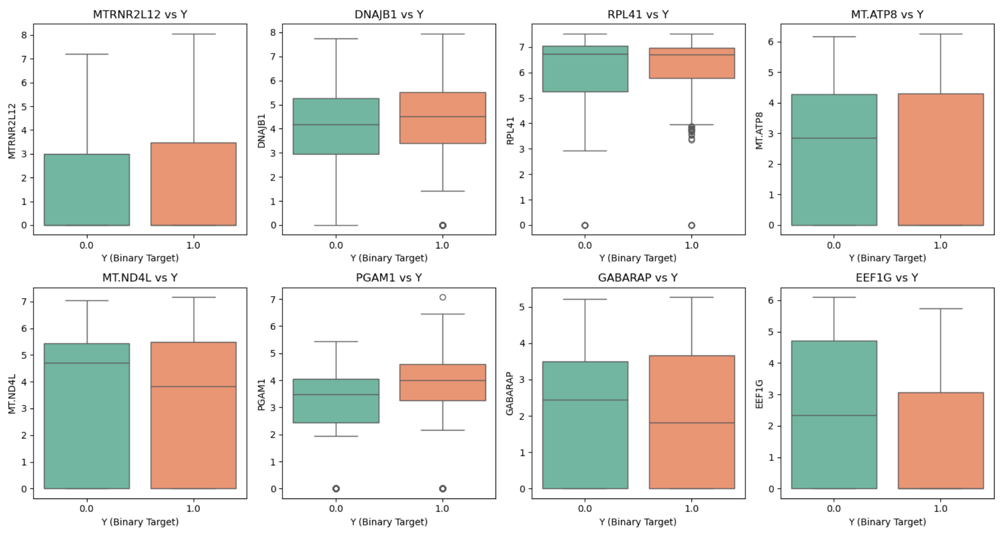

Figure 4: Even influential features are barely distinguishable across classes

5. When we run a naive logistic regression (with L1 penalty to ensure feature selection), which features have the largest coefficients? Do these features on their own offer strong predictive ability?

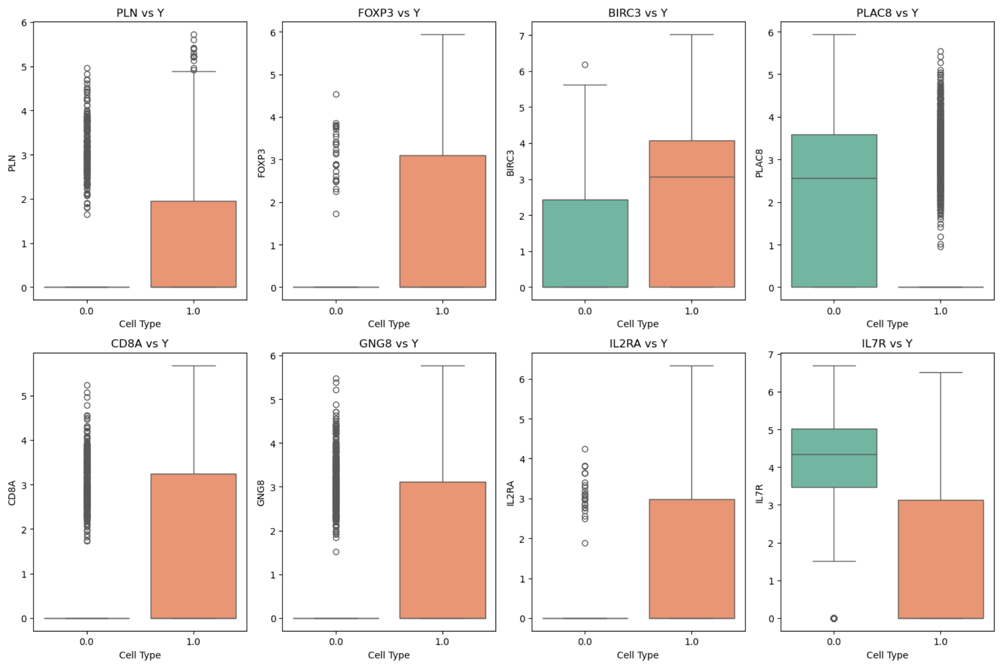

Figure 5: IL7R (bottom right) again stands out here just like in question 3!

While the boxplots of even the best genes tend to overlap, their means are at least distinguishable. This gives us some insight into the fact that a logistic regression might be a good classifier here.

We have 3 major conclusions from our exploratory data analysis. Firstly, the classes are not easily separable with all the current features, which would motivate PCA later. Secondly, there is minimal multi-collinearity meaning we can expect the variance-covariance matrixes to be invertible. Thirdly, a logistic regression is likely to work well as shown in question 5 but this is merely intuition for now and we continue to test this in later sections.

2 Train and evaluate classifiers

We randomly 70-30 split the data set into training and test. We will also do cross-validation later on the same 70% training set.

Pre-PCA Results

Table 1: Results before PCA and tuning hyperparameters.

Classifier	Accuracy	Balanced Accuracy	AUC	F1 Score
LDA	0.681	0.683	0.719	0.618
QDA	0.540	0.534	0.534	0.454
Logistic Regression	0.944	0.942	0.991	0.926
k-NN	0.717	0.653	0.711	0.511
GBDT	0.944	0.933	0.989	0.922
Random Forest	0.932	0.912	0.988	0.901
SVM	**0.958**	**0.946**	**0.993**	**0.941**

We discuss some notable findings below:

- Firstly, SVM is the best classifier across all metrics. This intuitively makes sense as SVM can handle very high-dimensional data well, thanks to the kernel trick. This allows SVM to implicitly map data into higher-dimensional spaces, making it easier to find a separating hyperplane, even in complex and non-linearly separable data.

- Secondly, as we suspected in the exploratory data analysis, logistic regression does perform very well. Logistic regression, despite being a relatively simple method, still outperforms some other more complex classifiers.
- We also notice the poor performance of k-NN which does align with the theory where k-NN performs poorly with highly-dimensional data. This is because if data is high-dimensional, and noise is present in all coordinates, then the distance can have high variance which reduces the accuracy of the k-NN classifier.
- Finally, QDA and LDA also perform poorly which informs us that the underlying dataset might not have homoskedasticity (assumption for LDA) and also might not follow a multivariate Gaussian normal distribution (assumption for QDA)

As discussed in section 1.1, we then proceed to do a 10 component PCA to reduce the dimensionality. While the task specifies us to do a 10-component PCA, we still would like to check if all 10 components are indeed important or if we can further regularise. We could do Forward or Backward Stepwise Selection here but as p = 10 now after the PCA, we opt to do best subset selection as p is not large and this is now computationally feasible. We used AIC and BIC as our scoring metrics and they both agreed on a model which **retains all 10 of the PCA components**. Other possible options here in regularisation were to use cross-validated prediction error to select the "best model".

Post-PCA Results

Table 2: Results after PCA but before tuning hyperparameters

Classifier	Accuracy	Balanced Accuracy	AUC	F1 Score
LDA	0.943	0.930	0.988	0.922
QDA	0.944	0.938	0.987	0.926
Logistic Regression	**0.953**	**0.950**	**0.989**	**0.939**
k-NN	0.925	0.905	0.977	0.894
GBDT	0.934	0.927	0.983	0.913
Random Forest	0.928	0.921	0.979	0.905
SVM	0.952	0.947	0.988	0.937

Our immediate observation is that PCA has massively improved accuracy scores across all metrics so we only proceed from here on with PCA-transformed data (using all 10 components). By reducing the number of features, PCA reduces the complexity of the classifiers, which significantly reduces the risk of overfitting. While Logistic regression now performs the best overall, we also note strong improvement in LDA, QDA and k-NN compared to pre-PCA.

3 Parameter tuning

We proceed with hyperparameter tuning for (1) Logistic Regression, (2) SVM (3) k-NN. We picked the first two as they had the most promising results in our previous stage while we picked k-NN as it was the worst of the classifiers (admittedly still doing pretty well at 0.894 F1 Score) and had the most room for improvement.

Firstly, we implement logistic regression with Elastic Net regularization and optimize it based on F1 Score. We run a GridSearch over values of regularization strength $C = [0.01, 0.1, 1, 10, 100]$ and the L1 ratio = $[0.1, 0.3, 0.5, 0.7, 0.9]$ where the L1 ratio mixes the penalty term between the L1 (Lasso) and L2 (Ridge). We use 10-fold CV F1 scores to pick hyper-parameters and the **best parameters we find are C = 1 and L1 ratio = 0.1**. The best CV F1 Score we find is 0.942, a slight improvement from pre-tuning but nothing to write home about. **This is our chosen classifier in the mypredict() function.**

For SVM, we again run a GridSearchCV over regularisation strength (which balances misclassification penalty vs margin size) $C = [0.1, 1, 10, 100]$, Kernel type = $[linear, rbf, poly]$ and finally Kernel coefficient $\gamma = [scale, auto, 0.001, 0.01, 0.1]$. **Out of our 60 candidates, the best combination was a Radial Basis Function kernel with $\mathbf{C = 100, \gamma = 0.01}$.**

Finally for k-NN, we iterate over number of neighbour values for k-NN from 1-30 and based on 10-fold CV F1 Scores, pick k = 5 as it maximises CV. We notice the spiky behaviour of the CV F1 Score which spikes down at even numbers of K and surmise this is due to arbitrary tie breaking for even number of neighbours which might be leading to lower F1 scores. Picking an odd number of 5 also avoids this problem.

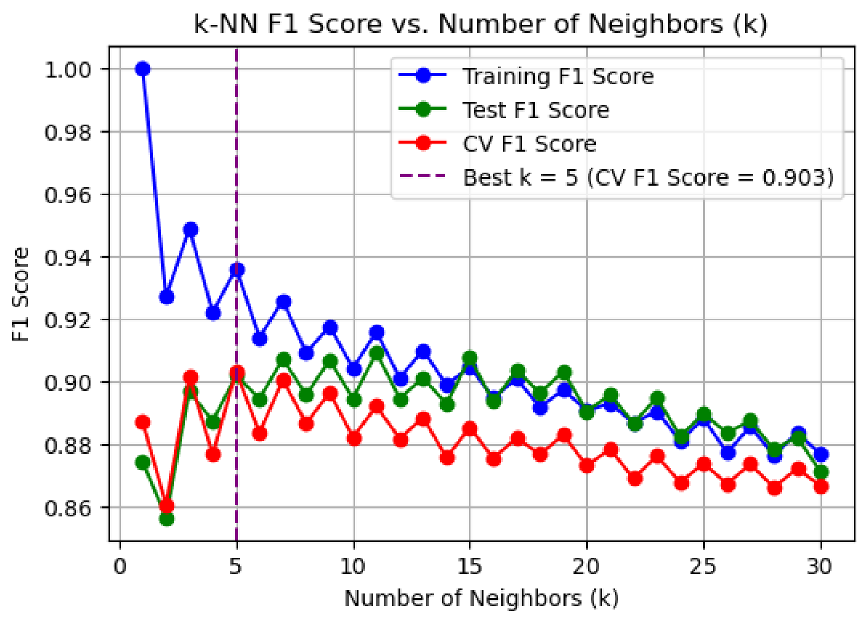

Figure 6: $k = 5$ has highest CV F1 Score

Task 2: Feature Selection

Abstract

We aim to predict whether different compounds bind to Thrombin using the DOROTHEA dataset of 100,000 molecular features. We compare a range of filter, embedded and wrapper methods and report their performance using cross-validation to obtain variance measures and additional robustness. The highest balanced accuracy is achieved with a combination of Mutual Information, Lasso and Support Vector Classifier, with Random Forests presenting a viable but more computationally intensive alternative.

1 Introduction

DOROTHEA is a drug discovery dataset, where different chemical compounds are represented by their structural molecular features and must be classified as "active", i.e. binding to Thrombin, or inactive. The dataset contains information on $n = 800$ different compounds and $p = 1 + 100,000$ columns where the first indicates whether the compound bound to Thrombin, labeled as "1" for the positive and "-1" for the negative class, and the remaining $100,000$ binary columns reflect structural features of the molecules that are used for prediction as well as random probes introduced for additional noise. We aim to predict the first column using the remaining 100,000 features.

The main challenge consists in the high dimensionality of the dataset which prevents the application of common estimation techniques such as OLS. We therefore perform feature selection with the primary goal of achieving highly accurate and reliable predictions. As a secondary objective, we aim to understand which and how many features are particularly influential as this information can guide the development of new drugs that reliably bind to Thrombin and fulfil other desirable properties.

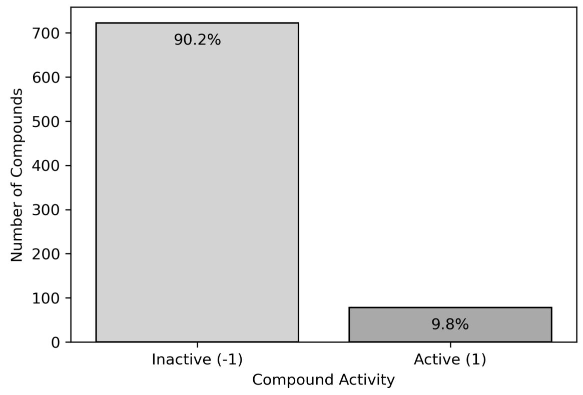

Figure 1: Class imbalance in target label. The plot reports the number of compounds in the positive and negative class and their percentage proportions.

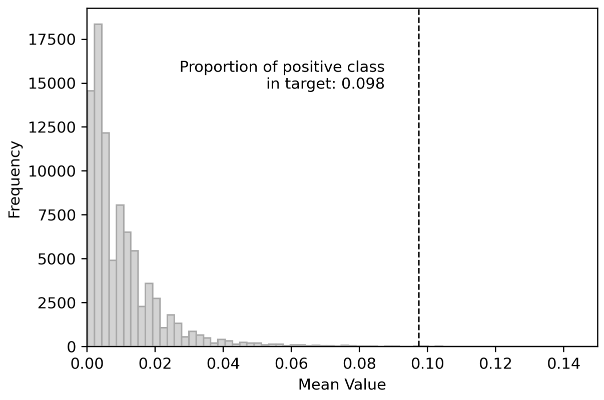

Figure 2: Feature Means. The plot shows the distribution of feature means (proportion of 1s as they are binary) and the proportion of the positive class in the target for comparison.

Besides the high dimensionality, the DOROTHEA dataset has two additional difficulties. First, the classes are highly imbalanced as less than 10% of the compounds in the sample successfully bind to Thrombin as illustrated in Figure 1. Second, the dataset is exceptionally sparse with a sparsity score of 99.1%, and most features are more sparse than the target variable. This property of the data is presented in Figure 2.

We proceed as follows from here. The second section comments on pre-processing and the third presents an overview of feature selection algorithms and their initial performance on the dataset. The fourth section offers two advanced methods which perform particularly well on the data. The fifth section concludes and highlights limitations and further avenues.

2 Pre-Processing

To address these challenges, we perform a series of pre-processing steps. First, we transform the dataset into a sparse matrix which retains the same information as the dense representation without loss. The sparse matrix enhances memory usage by decreasing the size of the dataset by an approximate factor of 80 in our case and renders some of the computationally intensive feature selection algorithms feasible with our resources. To reduce computational costs further, we do not rescale features as all features already have the same order of magnitude.

Second, given the stark class imbalance, we perform all cross-validation through stratified k-folds, i.e. the distribution of class-labels in each fold mirrors that of the overall dataset. In our case, this means that approximately 10% of observations in each fold should be of the positive class. To ensure valid results, we evaluate all models on a separate test set.

Third, to manage the high-dimensionality of the data, we preface some of the computationally expensive feature selection methods with so-called "filters". Feature selection algorithms are typically grouped in three distinct classes. *Filters* are fast and easy to implement and select features based on model-free metrics such as correlation or a minimum within-feature variance to discard constant features; examples include Variance Thresholding, or Mutual Information. *Embedded methods*, such as L1-penalized Logistic Regression (Lasso) or Random Forests are classification algorithms that automatically perform feature selection as part of their classification process. *Wrappers* refer to methods that repeatedly train models and select features based on the model's performance. As such, wrappers such as Forward and Backward Sequential Selection are sensitive to the choice of the underlying classification model, in addition to being more computationally expensive than the other two classes.

3 Initial Feature Selection

In this section, we test the performance of three filter methods, two embedded methods and two wrappers. The embedded methods and wrappers, i.e. Lasso, Random Forest, and Forward/Backward Stepwise Selection, were covered in the course. The filter methods are examined as new alternatives and for pre-processing.

The *Correlation Filter* evaluates the correlation between each feature and the target variable, selecting features with the highest absolute correlations. It assumes linear relationships and discards redundant or irrelevant features based on their weak correlation. *Variance Thresholding* removes features with variance below a specified threshold, assuming low-variance features are less informative. The *Chi-Square Filter* evaluates the independence between features and the target variable by calculating the chi-square statistic. Features with the highest scores are selected, as they show the strongest dependency on the target. *Random Forest* ranks features based on their importance score, which is determined by the reduction in impurity (e.g., Gini or entropy) they provide across the trees and can then be used to discard features below some threshold. It is robust to multicollinearity. Finally, *Lasso* is a logistic regression method that adds an L1 penalty to shrink feature coefficients. It sets the coefficients of less important features to zero, effectively performing feature selection while optimizing the model.

We evaluate each model at different levels of the hyperparameter governing the number of features selected. In the case of the Correlation Filter and Chi-Square Filter, this is controlled directly through the k parameter. For the Random Forest, we vary the threshold for feature importance and for Variance Tresholding we use different values for the minimum variance of a feature to be included in the model. Finally, for Lasso, we vary the regularization parameter λ, which is inversely related to the number of features selected. To reduce the dependence of our methods' performance on a particular split of the data, we perform each method across $k = 10$ stratified folds to ensure similar class balance across the splits and report the mean validation score and an interval of one standard deviation in each direction in Figure 3. Given previous results on this dataset in the literature and computational constraints, we focus on selecting 1000 or fewer features. To ensure comparability, we perform feature selection and then use Logistic Regression as the same classifier across all methods to obtain a balanced accuracy score. With the exception of Variance Tresholding and Lasso, all methods perform well even for a small number of features. The outcomes on the separate test set are reported in the table at the end. The Chi-Square filter selects only a single feature, though we note that this result should be treated carefully as it is subject to high variance.

Subsequently, we test the wrapper methods *Forward* and *Backward Stepwise Selection*. As both prove computationally infeasible on the full dataset, we initially perform Variance Tresholding for fast filtering to reduce the number of potential features and perform FSS and BSS afterwards. This proved particularly beneficial because the cost increases exponentially in the number of features. We find that both algorithms perform poorly with a maximum balanced accuracy of 0.556, potentially due to the fact that the algorithms were not able to access the entire dataset and only the pre-selected features through Variance Tresholding, which might not be those that would be chosen optimally. Due to the weak initial results and the extensive computing costs, we do not pursue these models further.

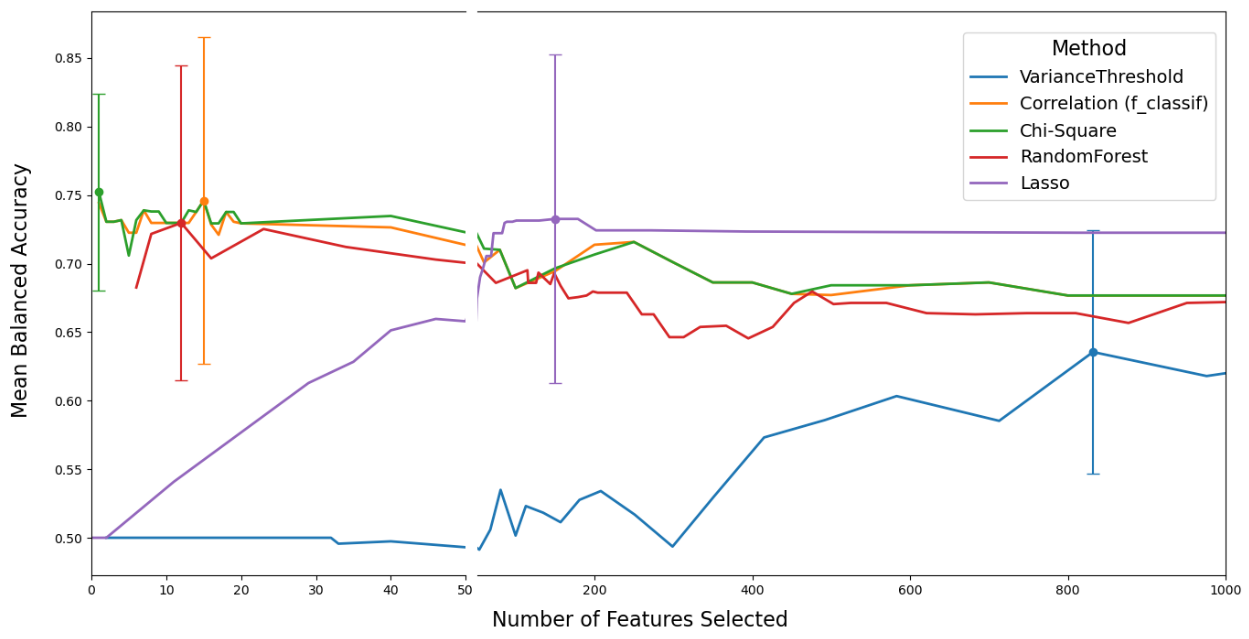

Figure 3: Results of Filter and Embedded Methods. The plot illustrates the accuracy of the tested methods against different numbers of features selected. The solid line reports the mean balanced accuracy across ten folds per method. The error bars correspond to one standard deviation in each direction for the respective method at the optimal point.

4 Advanced approaches

We now consider two advanced approaches: a Mutual Information filter with an optimised Random Forest Classifier, and a pipeline of Mutual Information Filtering, Lasso and Support Vector Classifier (SVC).

First, we implement pipelines of several consecutive feature selection methods. We take inspiration from Hamim et al. (2021) and Zhou et al. (2015) who suggest a three-phase approach of Mutual Information (MI) Filtering, Lasso and Support Vector Classifier (SVC) and achieve impressive accuracy scores on high-dimensional microarray data. MI quantifies the amount of information features provide about each other, capturing both linear and non-linear relationships in contrast to the Correlation Filter. This makes it particularly effective in DOROTHEA, where it helps filter out the irrelevant noise introduced by the random probes. We run a grid search to identify the best parameter combination from $k \in [50, 100, 200, \mathbf{500}]$ for MI, the regularization parameter λ using LassoCV, and the regularization $C \in [0.01, \mathbf{0.1}, 1, 10]$ as well as kernel types from $\{"linear", \textbf{"rbf"}, "poly"\}$ for the SVC. This three-phase method proves to be our most successful approach with a balanced accuracy of 92.7%.

Second, we optimize a Random Forest Classifier by first performing MI and subsequently grid-searching for the optimal parameters. Overall, this method also proves promising even if slightly less accurate and parsimonious than MI-L-SVC. Interestingly, we obtain a graphical representation in Figure 4 which illustrates the structure of the selected features after discarding features with an importance score below the mean importance score of all features. While there is no direct interpretation of this result, it reveals similarities between the selected features and the target that were not discernible in the original data.

We also considered a wide range of alternative feature selection approaches, inspired by both the NeurIPS challenge winners and a literature review. Bayesian approaches, like Automatic Relevance Determination (ARD) priors, and Nature-Inspired Optimisation Algorithms (NIOAs), like the bat and gray wolf algorithms (Al-Thanoon et al. (2022)), stand out as the most promising candidates. ARD priors effectively let the model choose the most relevant features by assigning a relevance parameter (prior distribution) to each feature. This works well in high-dimensional, sparse environments like ours, but is also computationally intensive. The Bat Algorithm by Yang (2010) uses bats' echolocation behaviour to balance exploration and exploitation, and the Binary Bat Algorithm (Mirjalili et al. (2014)) adapts this method to feature selection by letting each bat represent a feature subset. The evidence base supporting their value for feature selection in high-dimensional, sparse environments is strong, but the amount of computational power required to tune these models proved infeasible. And, as noted by Al-Thanoon et al. (2022), these models only demonstrate their value after considerable hyperparameter tuning.

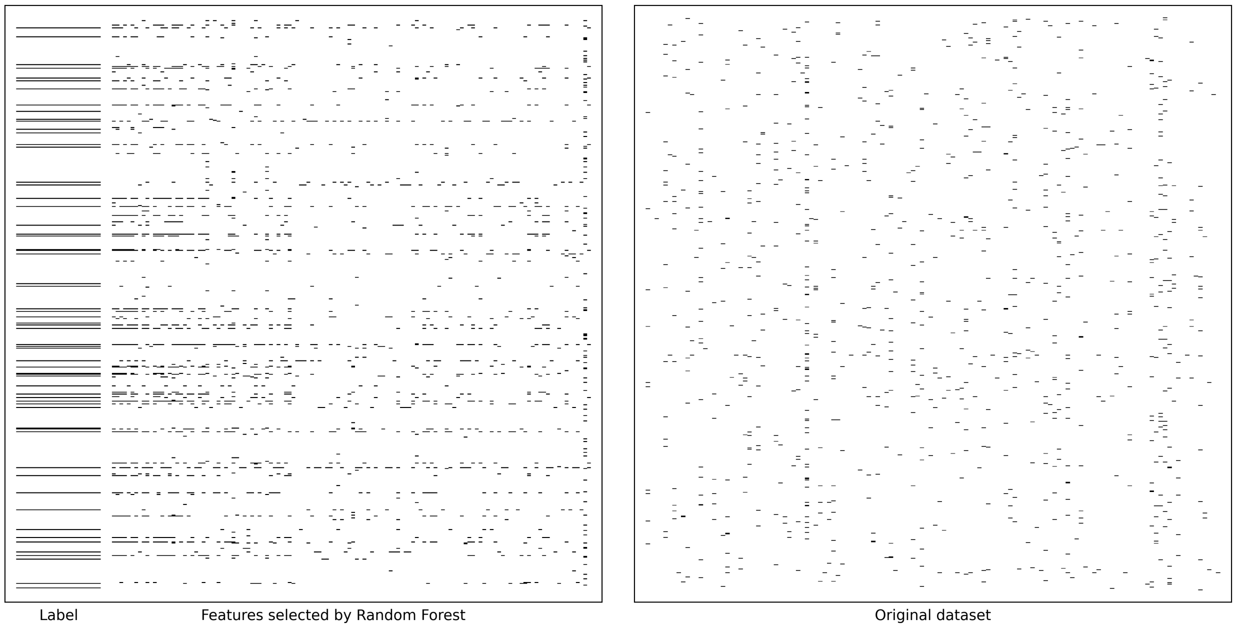

Figure 4: Visual representation of Random Forest Selection. Each black line in the left-most column represents a positive class in the target label. Each black dot in both panels denotes the value 1 for that feature and that observation. The left panel shows the features selected by the Random Forest sorted by feature importance, the right panel shows the first 100 features the initial dataset.

5 Conclusion

Table 3: Final Results of Feature Selection Methods on Test Set

Approach	**Balanced Accuracy**	**Number of Features**
Variance Thresholding	0.559	876
Chi-Square-Filter	0.708	1
Correlation-Filter	0.708	15
Random Forest	0.681	23
Lasso	0.715	257
MI-L-SVC	0.927	44
MI-RF	0.888	128

We test a variety of feature selection models to obtain a parsimonious and accurate predictive model on the DOROTHEA dataset. Among the cost-efficient filter methods, the Chi-Squared filter and Correlation filter perform well but fail to achieve accuracies higher than approximately 0.7. To achieve higher accuracy, we employ more complex models based on the existing literature and find the best outcomes with an optimized Random Forest and a pipeline of Mutual Information, Lasso and SVC.

As further avenues for research, we suggest exploring Bayesian approaches and Nature-Inspired Optimization Algorithhms (NIOAs). Bayesian approaches such as Automatic Relevance Determination priors demonstrated their value in the NeurIPS challenge, and NIOAs have significant empirical support in the literature.

References

[1] M. Hamim et al. “A Hybrid Mutual Information-LASSO-Genetic Algorithm Selection Approach for Breast Cancer Classification”. In: *Proceedings of the International Conference on Advanced Intelligent Systems for Sustainable Development*. 2021, pp. 429–439. DOI: `10.1007/978-981-16-2275-5_36`. URL: `https://link.springer.com/chapter/10.1007/978-981-16-2275-5_36`.

[2] J. Jin et al. “Influential features PCA for high dimensional clustering”. In: *The Annals of Statistics* 44.6 (2016), pp. 2323–2359. DOI: `10.1214/15-AOS1423`.

[3] L. McInnes et al. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: *arXiv preprint* arXiv:1802.03426 (2018). `https://arxiv.org/abs/1802.03426`.

[4] S. Mirjalili et al. “Binary bat algorithm”. In: *Neural Computing and Applications* 25 (2014), pp. 663–681. DOI: `10.1007/s00521-013-1525-5`.

[5] N.A. Al-Thanoon et al. “Improving nature-inspired algorithms for feature selection”. In: *Journal of Ambient Intelligence and Humanized Computing* 13 (2022), pp. 3025–3035. DOI: `10.1007/s12652-021-03429-4`.

[6] X.S. Yang. “A New Metaheuristic Bat-Inspired Algorithm”. In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Ed. by J.R. González et al. Vol. 284. Studies in Computational Intelligence. Springer, Berlin, Heidelberg, 2010, pp. 65–74. DOI: `10.1007/978-3-642-12538-6_6`.

[7] Zhi-Hua Zhou et al. “Feature Selection Algorithm Based on Mutual Information and Lasso for Microarray Data”. In: *The Open Biotechnology Journal* 10 (2015), pp. 278–283. DOI: `10.2174/1874070701510010278`. URL: `https://openbiotechnologyjournal.com/VOLUME/10/PAGE/278/FULLTEXT/`.