

**Computer Science**

**Data Mining Project (CS 590)**

Predict Disease Classes Using Genetic Microarray Data

**Contributors**

Alfred Ntiamoah (002317287)

July, 18th 2022

**1.0 Introduction**

The best approach to accurately identify disease-causing genes is monitoring gene expression values in different samples using microarray technology [1]. One of the challenges associated with this dataset generated by this technology is that the data is skewed in favor of the gene expressions compared to the number of samples i.e. large number of genes versus small sample size. Classification results on such data often do not generalized well enough to categorize unseen data due to small sample size. The research was to learn from a genetic microarray data and then predict disease classes using machine learning technologies. The research was conducted by comparing the performance of the various machine learning algorithms predicting disease classes on two different datasets, generated with T-value filtering and Principal component analysis. Based on their performance, the best algorithm was selected and further used to predict unknown samples.

**1.1 Tools and Libraries**

Python language and its libraries was used to perform this research. Numpy library was used for algebraic (matrix) calculations, Pandas library provided the data framework for loading data from disk, structuring and manipulating in-memory data. Sklearn machine learning library also provided the algorithms to perform task such as preprocessing, metrics, cross validation, principal component analysis etc. It provided the models – algorithms- (Support Vector Machine, KNeighbor Classifier etc.) that was used for data classification as well. Matplotlib and the Seaborn libraries was used to perform data visualization and graphing.

**2.0 Methodology**

The first part of this research work was focused on data loading, cleaning and structuring. This was done with the pandas’ library. The dataset contains 60 samples with 7070 features. The set was loaded into memory and then duplicated for cleaning. Duplicated because two different analyses were used to filter the dataset which could modified the original data in memory.

The training data was threshold between 20 and 16000, thus all genes that expressed value below 20 is set as 20 and all values greater than 16000 was trimmed to 16000.

Fold difference is mostly used to analyses how values in a single record in a dataset e.g. row or column deviate from each other hence introducing variation. Fold difference was calculated from the training data set for the various genes. A fold difference for each gene is calculated by  
dividing its measurement in one group by its measurement in the other group [1]. All genes with fold difference less than two (2) was dropped thus they did not introduce enough variation among the samples.

**2.1 Dimensionality Reduction with T-Value and/or Principal Component Analysis**

Most data mining algorithms work poorly in high dimensional space a popular phenomenon called the curse of dimensionality [2]. The dataset has about 7070 features which makes it very difficult to visualize. Also, may be not all the features are necessary to discriminate one disease from the other so there should be a way to select the best feature set that can discriminate the samples. Reducing dimension helps to speed up the training process thus reducing computation resources required though it may be the expense of performance (accuracy). In other to reduce the dimension of the dataset, T-value and principal component analysis (PCA) were used to select features (set of genes expression) the caused the most variation to discriminate the sample.

**2.1.1 T-Value**

If the whole data is combined and sorted out based on T-value, there will be chance that features that favored only for example, MED data will be selected and other diseases will be ignored. This can occur if most of the features discriminate MED disease class from other classes but introduce very little variation among other disease classes. So to ensure this never happened, equal parts from the data sets (features) is selected based on how much variation they introduce in to the dataset in favor of each class (Disease). T-value is used to select the top N features (genes) and then merged to compose equally proportioned data which will help to prevent skewing predictions to favor disease classes that introduced must variations into the data set. Top 2 means selecting top 2 of the individual features that introduced the most variation to separate a disease class from other classes and then combining those features to form a new dataset. Top 2 will have a maximum feature count of 10, thus 2 times number of classes. The count could be less since a single feature may contribute to the variation of different class hence introducing duplication. Top 3 will result in at most 15 feature set since we are dealing with 5 disease classes.

**2.1.2 PCA**

In other to get the best prediction I performed PCA on the other copy of the dataset and selected the top components to represent top N thus top 2 means the best two components which introduced more variation to discriminate the samples (Disease). The dataset used for the PCA was not threshold instead, it was scaled between 20 and 16000 using the Min-Max scaler. The standard scaler was further applied to the dataset to center the mean since mean-centering has been often used to create models of multivariate data in multivariate statistical process control though not a necessity as described by Neal B. etal [3, 10]. Centering the mean reduced the training time significantly.

The scalers fitted on the training dataset are also used to transform the test dataset that will be used to validate to validate the performance of the models. The number of features that can be generated was restricted by the number of samples but we only considered up to top 30 components.

The various top N dataset was written to file which will further served as the input to the next phase, Model building.

**3.0 Model Building**

The next phase of the research was to build the models. As restricted by the research requirement, the algorithms that were considered are Naïve Bayes, Decision tree (J48 algorithm), K-Nearest Neighbor (KNN- IBk algorithm), Neural Network (Multilayered Perceptron) and Support Vector Classifier as an additional algorithm. I performed parameter tuning on the various algorithms with the help of grid-search-cv tool to obtained the best parameters that yielded the highest accuracy. These parameter was further used to configure the algorithm for the final training, validation and testing. Using pipeline [4] from the Sklearn library made it easy for bundling different phases of the learning process in a single function call.

* 1. **Cross validation**

The basic approach to cross validation, the training set is split into k smaller sets hence the name K-fold validation. The following procedure is followed for each of the k “folds”:

* A model is trained using k−1 of the folds as training data;
* the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy). [5]

We first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining v − 1 subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The training data was split into five (5) folds, the model is train with four (4) folds and then validated on the remaining one (1) part of the dataset. Thus the other one fold is used as a test set to compute a performance measure such as accuracy. The performance measure reported by K-fold cross-validation is the average of the accuracies of the model on each split used as a test set. [5, 17]

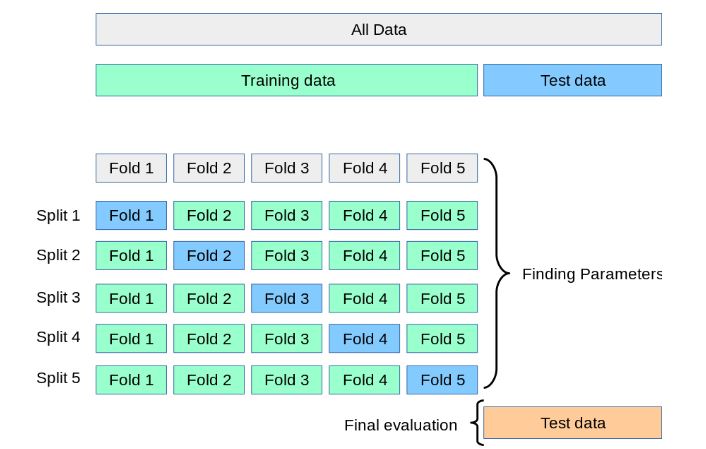


Figure 1 <https://scikit-learn.org/stable/_images/grid_search_cross_validation.png>

Combining Grid-Search with Cross-validation help in selecting the best parameters while reducing the chance of overfitting the dataset. With the k-fold cross validation, the algorithm has to learn from the dataset K times thus five (5) times for this research. This approach can be computationally expensive, but does not waste too much data, which is a major advantage in this research project since the number of samples are far less that the features.

**3.2 Metrics**

Metrics the mechanism to quantify or measure the performance of a model. This research work used the cross\_val\_score function to compute the accuracy of the models prediction. The cross\_val\_score used the f1\_macro (used for multiclass problems) which computed the f1 score by the formula

f1= 2\*(precision\*recall)/(precision+recall) [5].

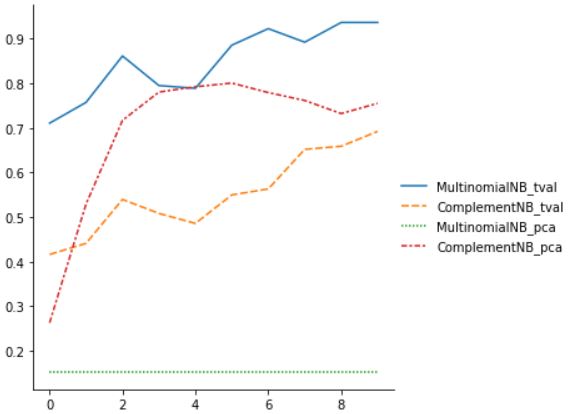
The output of the cross\_val\_score is K (5) f1 scores, and in other to obtain the final accuracy, the arithmetic mean of the various accuracies was calculated.

**3.3 Naïve Bayes Classifier**  
The classifier uses a probabilistic approach to predict the membership of a class by a given sample. It’s based on the Bayesian’s conditional probability theorem and the maximum posteriori hypothesis [6]. Detail explanation on these topics are beyond the scope of this research. This classifier similar to other classifiers, works better when there is an increase in the number of attributes in the data set as it had a great performance in speed and accuracy on the Leukemia (Cancer gene expression dataset) of about 7,129 records [7].

Using two variants of the Naïve Bayes classifier provided by the Sklearn library; Multinomial and Complementary, I fitted these algorithms to the datasets (t-val, pca). Cross validation score was computed for each algorithm per the dataset. It took about 6 seconds to run. The output of the results is summarized in the table below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TopN** | **MultinomialNb\_tval** | **ComplementNB\_tval** | **MultinomialNB\_pca** | **ComplementNB\_pca** |
| 2 | 0.710 | 0.416 | 0.153 | 0.263 |
| 4 | 0.757 | 0.441 | 0.153 | 0.529 |
| 6 | 0.861 | 0.539 | 0.153 | 0.717 |
| 8 | 0.794 | 0.508 | 0.153 | 0.780 |
| 10 | 0.789 | 0.486 | 0.153 | 0.792 |
| 12 | 0.885 | 0.550 | 0.153 | 0.800 |
| 15 | 0.922 | 0.563 | 0.153 | 0.779 |
| 20 | 0.892 | 0.652 | 0.153 | 0.761 |
| 25 | 0.936 | 0.659 | 0.153 | 0.732 |
| 30 | 0.936 | 0.692 | 0.153 | 0.755 |

Table 1.0 Naïve Bayes Classifier output.



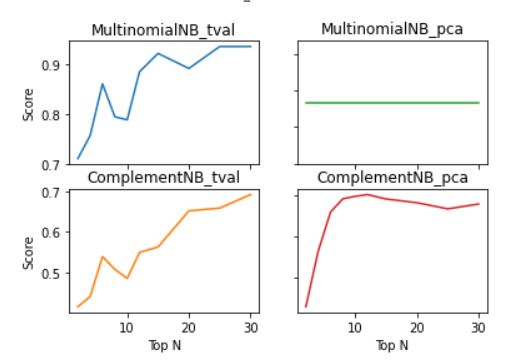


Figure 2 Comparing the Naive Bayes classifier variants

No parameter tuning was performed for this algorithm because there were very few parameters to configure. It took about 6 seconds to train the model. It indicated from the plot that Multinomial algorithm on the t-value reduced dataset had the best performance in terms of accuracy, so it was selected as the best algorithm to represent the Naïve Bayes classifier in the contest. Also it was seen from the graph that the accuracy was growing linearly proportional in relation to the top N dataset. This indicates that adding more features may increase the accuracy but for this research purposes, top-30 t-value reduced training dataset will be considered as the best dataset for this classifier.

**3. 4 Decision Tree (J48 Algorithm)**

A decision tree is a tree-based technique in which any path beginning from the root is described by a data separating sequence until a Boolean outcome at the leaf node is achieve. The nodes and branches are composed of each tree. It consists of nodes that form a *rooted tree*, meaning it is a *directed tree* with a node called “root” that has no incoming edges. Each node represents features in a category to be classified and each subset defines a value that can be taken by the node.It can be used to make assumptions regarding categorical class names, to classify knowledge on the basis of training sets and class labels, and to classify newly obtainable data. [8,9]

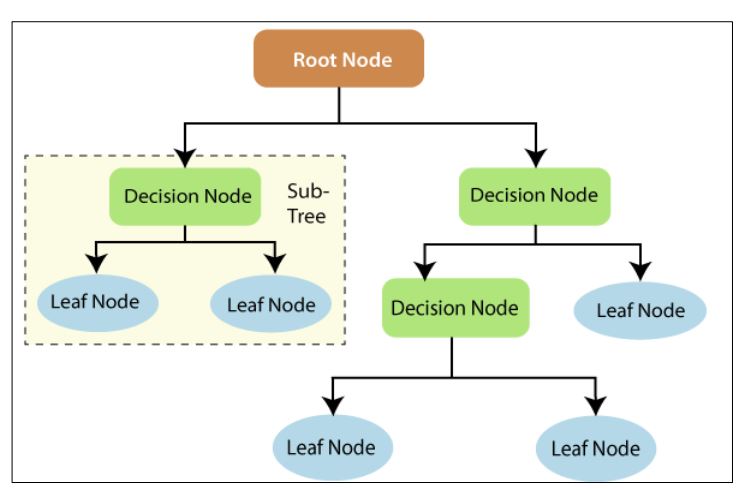


Figure 3. Bahzad T. J., Adnan M. A., Classification Based on Decision Tree Algorithm for  
Machine Learning

**3.4.1 Advantages and Disadvantages**

Some important strong hold of decision tree is that, they are very easy to comprehend, easy to build new principles for production, classify both categorical and numerical values though the generated attribute should always be categorical. It’s also good at handling sparse dataset or data set with outliers or errors. [8,9]

Finding the optimal decision tree algorithms is feasible only in small problems; considered as NP-Hard problem and even not without heuristics. The algorithm tends to perform well on few highly relevant attributes but not so well with many attributes with complex interaction. Its greedy nature makes it vulnerable thus over sensitive to noise [9].

**3.4.2 Approach**

First, the algorithm was fitted to the dataset with its default parameters and it had an average score of 0.781. After performing parameter turning, the following parameters below yield much better accuracy of 0.905.

{'criterion': 'entropy', 'max\_features': 'log2', 'min\_samples\_split': 2, 'random\_state': 0, 'splitter': 'best'}

These parameters performed better on the top-10 t-value reduced training dataset. The decision tree classifier will be configured with the above parameters and will be use as representative for this research work.

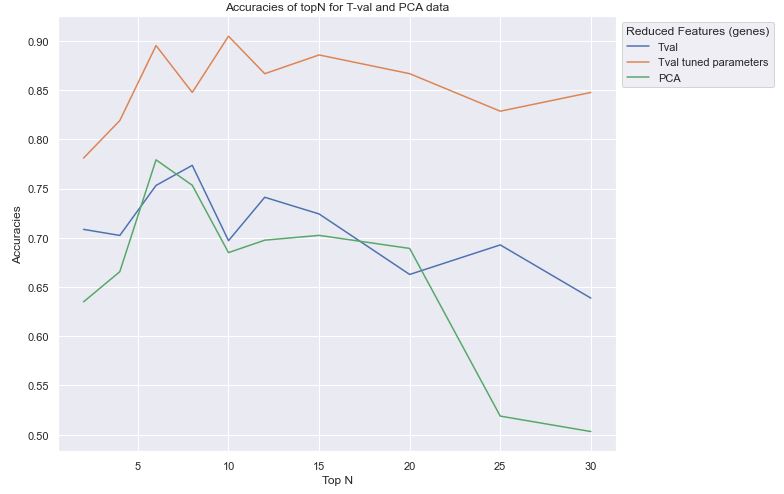


Figure 4 Decision tree performance on the Top N features (Genes)

**3.5 K-Nearest Neighbor Classifier**

K−nearest neighbor’s algorithm, considers the k−nearest point to a sample and then categorizes the sample to that point base on its distance. The k-NN approach leverages this distance formula as its basis for finding the closest neighbor in a Cartesian coordinate system. Small value of K is accurate by does not generalize very well which may lead to unstable boundaries. The k-nearest neighbors are universally consistent if k → ∞ and k/n → 0 [11]

**3.5.1 Advantages and disadvantages**

KNN is a very simple algorithm which makes it very easy to use and also understand the results. It’s also easy to implement with no need to estimate parameters. No or very little training is required and very suitable for the classification of rare events; particularly suitable for multi-classification issues. [12]

As a non-parametric and lazy learner algorithm, it suffers from the curse of dimensionality. Thus for large training sets will require large computing resources memory which can leads to slow performance when making a prediction. The prediction accuracy can quickly degrade when number of attributes grows. It requires similarity measure and attributes that “match” target function which is not often obvious. [12, 13]

**3.5.2 Approach**  
  
Grid search was used to find optimal parameters including the best k value. Below are the best parameters obtained from the grid search.

*{'algorithm': 'auto', 'leaf\_size': 10, 'metric': 'manhattan', 'n\_jobs': 8, 'n\_neighbors': 2, 'p': 1, 'weights': 'distance'}*

From grid search, the best K value was seen to be two (2) for most of the top N dataset. It was also seen that from Top12, the accuracies did not improve.

Table 1 Performance of the K-nearest neighbor on the top N

|  |  |  |
| --- | --- | --- |
| Top N | Accuracies | Best K |
| 2 | 0.895 | 3 |
| 4 | 0.914 | 2 |
| 6 | 0.952 | 2 |
| 8 | 0.981 | 2 |
| 10 | 0.962 | 2 |
| 12 | 0.981 | 2 |
| 15 | 0.981 | 2 |
| 20 | 0.981 | 2 |
| 25 | 0.981 | 2 |
| 30 | 0.981 | 2 |

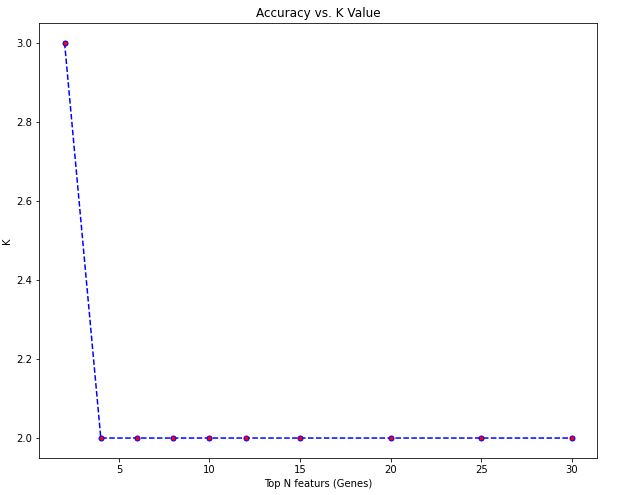


Figure 5. Best K vs top N features (genes)

The training process took about 5min to complete since it combined the parameter tuning with cross validation through the pipeline. The K-nearest neighbor classifier was fitted to the two dataset to determine the best accuracies yet, t-value reduced dataset yielded much higher accuracy compared to the PCA reduced version.

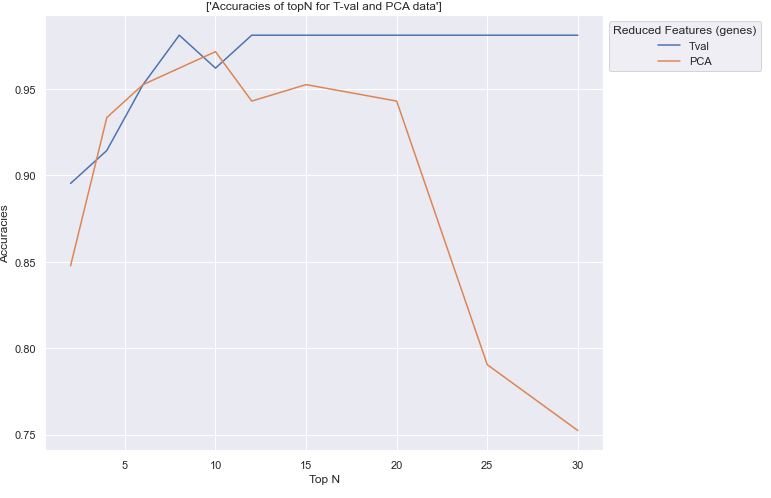


Figure 6 Performance of the KNN on the Top N datasets

**3.5 Support Vector Classifier:**  
Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize accuracy while automatically avoiding over-fit to the data. [14]

Classification in SVM is an example of Supervised Learning thus known labels help indicate the correctness of the system’s performance. The labels direct the algorithm to a desired response, validating the accuracy of the system, or be used to help the system learn to act correctly. SVM finds a linear separating hyperplane with the maximal margin in higher dimensional space. It’s one of the algorithms that uses kernels to perform feature extraction and dimensionality reduction.

**3.5.1 Advantages and Disadvantages**

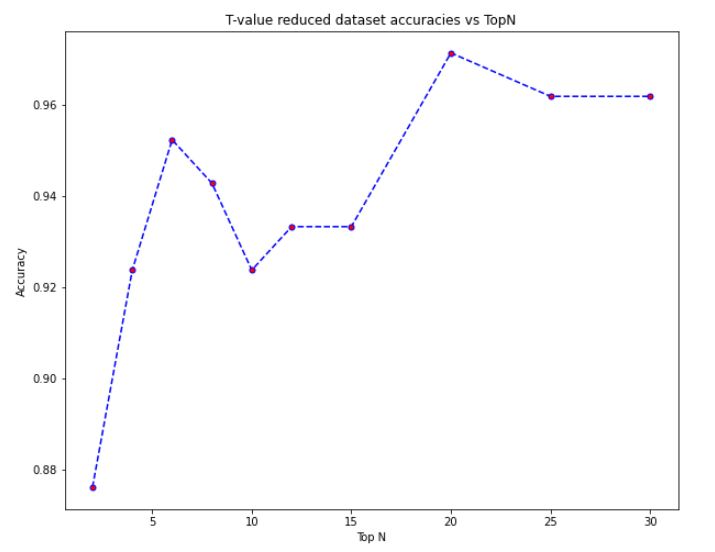
The major strengths of SVM are the training is relatively easy to use. It avoids local optima during learning unlike in neural networks. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly. The weakness includes the need for a good kernel function. The radial basis function (RBF) kernel and it’s variant like the Gaussian RBF is often used in most of applications. [15, 16]

**3.5.2 Approach**

First, grid search was also performed to determine the best parameters, and the following results was obtained.

*{'C': 10, 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'random\_state': 0}*

Using a regularization parameter of 0.1 thus the inverse of the parameter C, game value set to scale, with the radial base function kernel produce the best accuracy. The degree was ignored because its only used by the polynomial kernel. A random state of zero (0) is used by the pseudo random generator to make the result reproducible. The classifier yielded the best accuracy of 0.971 on the top 20 t-value reduced dataset.



There was a slight problem with this classifier fitting the PCA reduced dataset i.e. the training was taking too long to converge. Scaling with min-max Scaler within the range of 20, and 16000 and also using the Standard scaler to adjust the mean yet the classifier did not converge in a reasonable time which indicates the requirement of much more computing power. Considering the lagging performance of the other classifiers on the PCA reduced dataset, I decided to use the T-Val data to conclude on the analysis for this classifier.

**3.6 Neural Network (Multilayered Perceptron):**

A multilayer Perceptron is a variant of the original Perceptron model proposed by Rosenblatt in the 1950 [18, 19], Learning or training of ANN is equivalent to finding the values of all weights such that the desired output is generated to corresponding input, it can be viewed as the minimization of error function computed by the difference between the output of the network and the desired output of a training observations set. Errors generated at the output of the network is propagated through the network backwards, signaling all neural units to adjust its weight to minimize the error. This is known as back-propagation.

layers between its input and output layers, the neurons are organized in layers, the connections are always directed from lower layers to upper layers. Learning for the MLP is the process to adapt the connections weights in order to obtain a minimal difference between the network output and the desired output.

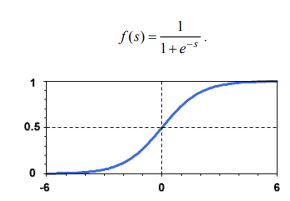
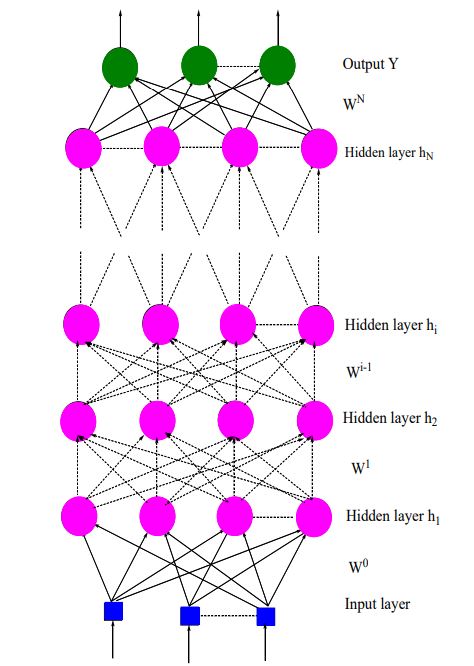


Figure 7. Feed-forward neural network with 4 hidden layers [18] Figure 8. Sigmoid activation function

The power of the multilayer perceptron comes precisely from non-linear activation functions. Almost any non-linear function can be used for this purpose, except for polynomial functions. [18, 19, 20]

**3.6.1 Advantages and Disadvantages**

The advent of big data has make neural network the best choice to learn from such large amount of data. It performs well with most problems compared to other algorithms.

Training neural network is difficult compared to other algorithms this is because it is often treated as a black box. Selecting the hyper-parameters and tuning them to achieve the desired outcome takes a lot of time. Example selecting high learning rate may jump around the global minimum and will never converge while selecting small learning rate may also be locked up in a local minimum. It uses a lot of computing power in training compared to other algorithms and also requires large amount of data to function well else it will easily over-fitting. [20]

**3.6.2 Approach**

Grid search was also performed on the perceptron classifier. There were several parameters to be configured and due to that the training was taking too long to converge. I randomly selected group of parameters to search from.

*{'max\_iter':[5000], 'hidden\_layer\_sizes':[(50,),(100,)],*

*'activation':['tanh', 'relu'], 'solver':[ 'sgd', 'adam'],*

*'random\_state':[1], 'alpha':[10\*\*-6, 10\*\*-4,],*

*'learning\_rate':['constant', 'adaptive'], 'shuffle':[True]}*

From this group, the best parameters to configure the classifier was obtained on the top 10 t-value reduced dataset.

*{'activation': 'relu', 'alpha': 1e-06, 'hidden\_layer\_sizes': (100,), 'learning\_rate': 'constant', 'max\_iter': 5000, 'random\_state': 1, 'shuffle': True, 'solver': 'sgd'*}

These best parameters yielded an accuracy of 0.991 on the top10 t-value reduced dataset. The classifiers used the rectified linear unit activation function (RELU) [21], regularization parameter of 1e06, 100 hidden layers, constant learning rate, and stochastic gradient descent (SGD) solver for cost minimization.

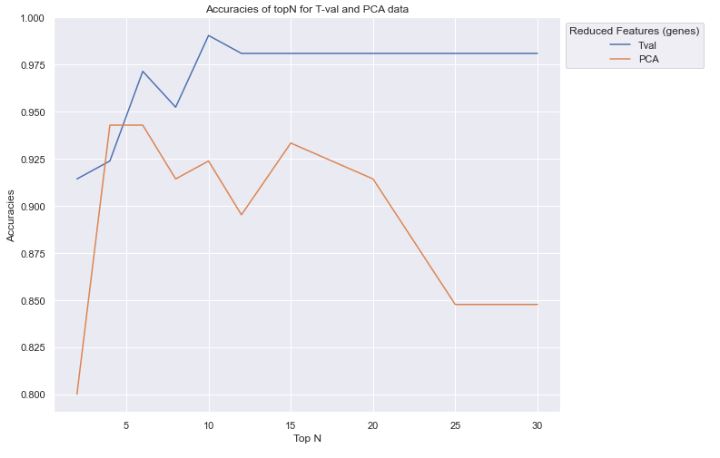


Figure 9 Performance of the ANN (MLP) on the Top N data sets

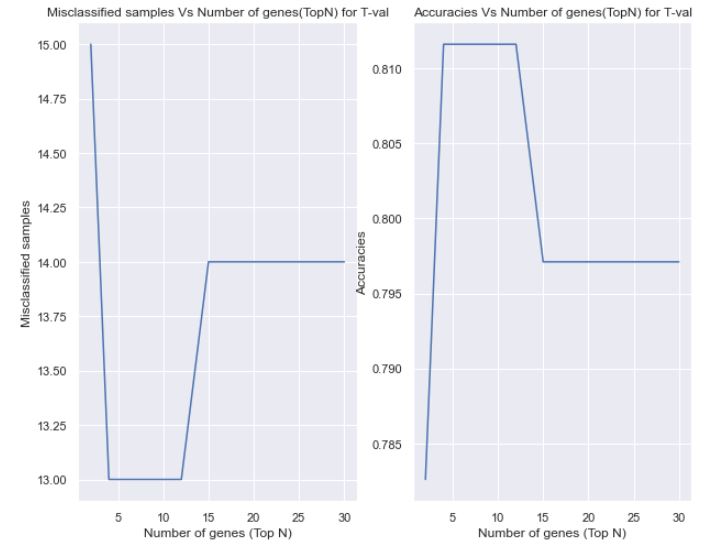
**4.0 Adaptive Boosting**

The goal of boosting is to improve the performance of the weak learning algorithm, weak in the sense that the performance of the algorithm is close to the performance of random guessing though we can assume that the algorithm is a little bit better than classifier whose every prediction is a random guess. [22] Adaptive boosting is nothing but a forest of stumps rather than trees which works by putting more weights on difficult problems and less weight on problems that can easily be handled. A stump is a weak classifier thus on its own gives less accuracy in predictions but a combination of many stumps can be used to make a stronger classifier that can achieve a much higher accuracy compared to that of a single stump. In boosting with stumps, the current stump learns from the errors of the previous stump and this run iteratively to reduce the error in every step until the final stump hence improving the accuracy.

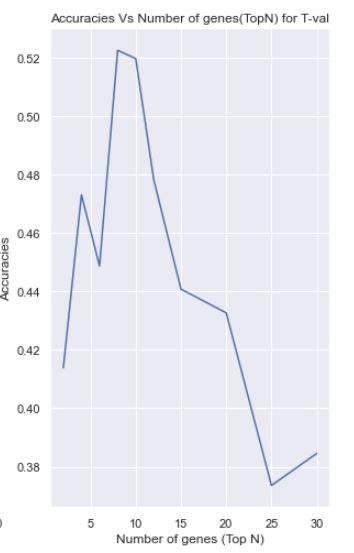
Sklearn provides a package for adaptive boosting which is based on the two algorithms SAMME and SAMME.R.Comparing the performance of SAMME and SAMME.R algorithms, SAMME.R uses the probability estimates to update the additive model, while SAMME uses the classifications only. This makes SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations. Boosting the decision tree classifier using the SAMME.R algorithm achieved a better performance (higher accuracy) than using decision tree classifier without boosting. Cross-validation was combine with adaptive boosting to help prevent overfitting. This research will use basic stump (decision tree classifier) with following parameters. [23, 24, 25]

criterion **=** "gini", random\_state **=** 0, max\_depth**=**3

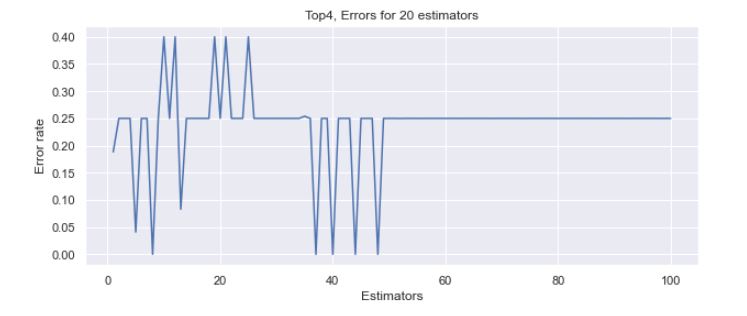
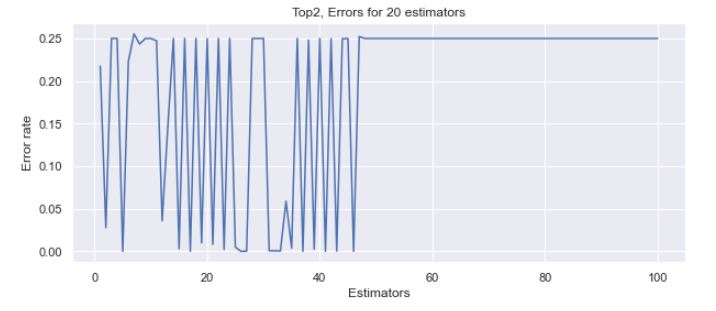
The graph below is the performance of a single stumps on the Top-N t-value reduced dataset.

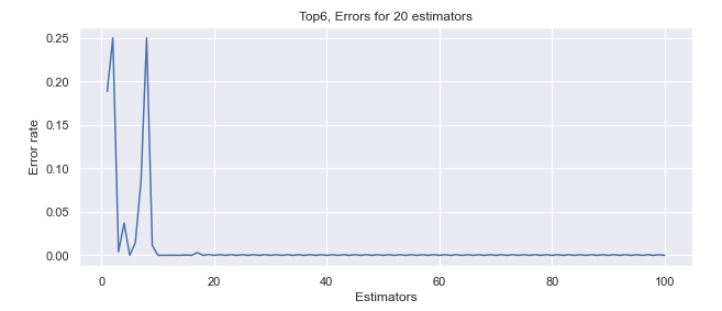


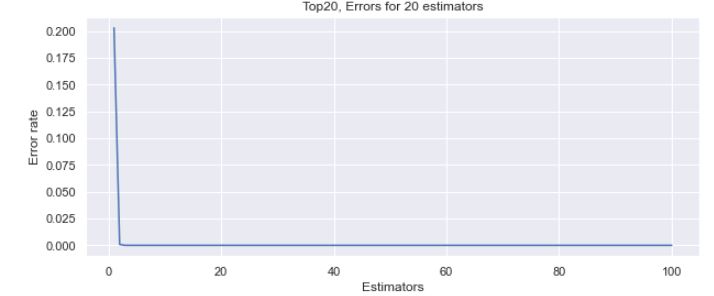
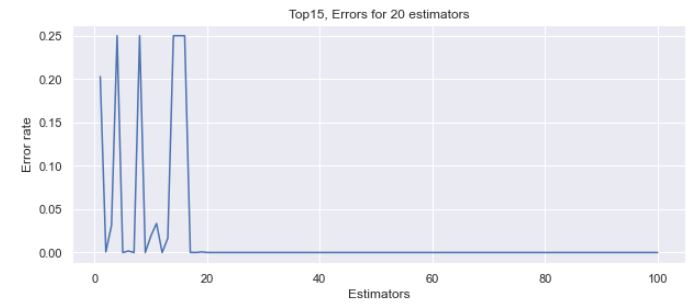
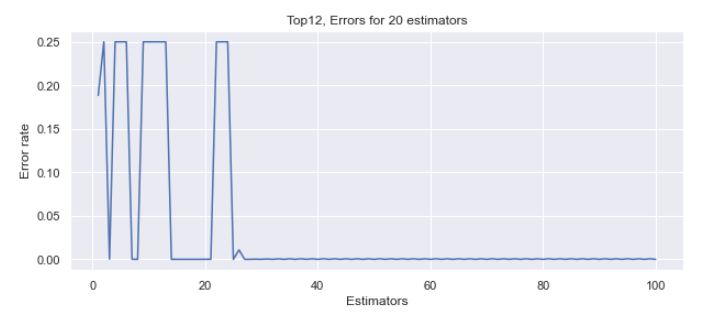
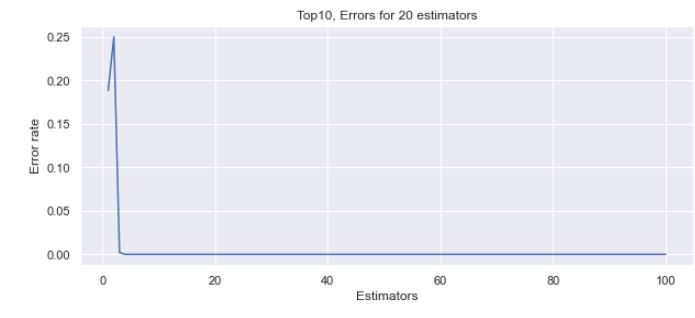
A single stump has a higher error rate since it misclassified about 13 samples which is about 20% error rate. This yields an accuracy of about 80%. When the stump was fitted with cross validation split, it performed worse with about 50% accuracy which occurred with the top6 dataset.



The stump was able to achieve about 100% accuracy without cross validation and 97% accuracy with cross validation with adaptive boosting. Below are the graphs of error rates versus the top N genes using 100 estimators with the SAMME.R algorithm. The graph of top-20 was similar to top-25 and top 30 so they were omitted for brevity.







Combining adaptive boosting with cross validation an accuracy of about 97% was obtained. Below is the graph of performance on the Top-N dataset.

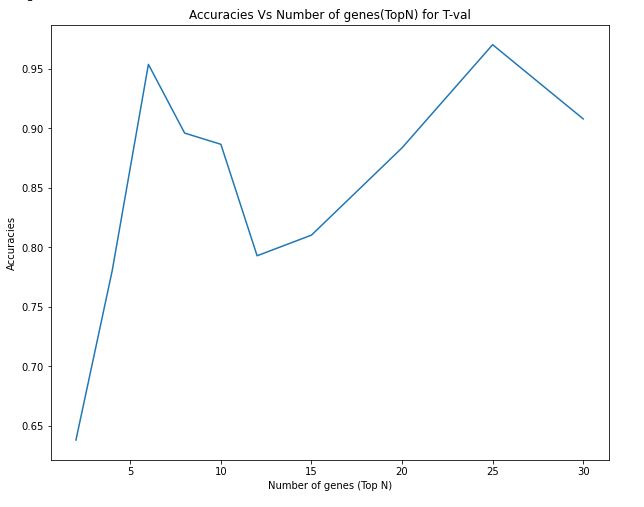


Figure performance of Adaptive boosting on the to-N genes

The output predicted by the Decision tree with adaptive boosting on the t-value reduced Top-25 test dataset are

['MED' 'EPD' 'MED' 'MED' 'EPD' 'MED' 'MED' 'MED' 'EPD' 'JPA' 'JPA' 'MED'

'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'RHB' 'RHB' 'MED' 'MED']

**5.0 Results**

It is found that almost all the model performed well on the t-value reduced dataset, so it will be used for the final decision in selecting the best model and they are all cross validated. The performance of the classifiers is summarized in the table below.

|  |  |  |
| --- | --- | --- |
| **Classifier** | **Highest Accuracy obtained** | **Top N Dataset (T-value reduced)** |
| Naïve Bayes (Multinomial) | 0.936 | Top25, Top30 |
| Decision Tree (J48) | 0.905 | Top-10 |
| K-Nearest Neighbor | 0.981 | Top-12 to Top30 |
| Support Vector | 0.971 | Top-20 |
| Neural network (Multilayered Perceptron) | 0.991 | Top-10 |
| Adaptive Boosting (Decision tree) | 0.971 | Top-25 |

The multilayered perceptron had the best performance and it is then used for the final prediction on the test data.

This is the output prediction of the Neural network algorithm on the Top-10 T-value reduced dataset.

['MGL' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED'

'MED' 'MED' 'RHB' 'MED' 'MED' 'MED' 'MED' 'MED' 'MED' 'RHB' 'MED']

**6.0 Conclusion**

The research was performed with no assumption except one human misclassified error in the dataset. This shouldn’t affect the performance of the algorithms in case one need to replicate the experiments with different dataset.

It was found that, neural networks are good for most problems including the problem posed in this research. K-nearest neighbor performed well with a small training or prediction time. With Adaptive boosting, the performance of decision tree algorithm was improved about 7%.

All the algorithms performed really well on the t-value reduced dataset than the principal components.

The research can be improved by considering other algorithms or other variants that were not used in this paper. A dataset with much higher sample size might be beneficial to improve the algorithms performance.

**7.0 References**

1. Butte A. J., Niederfellner G., etal., DETERMINING SIGNIFICANT FOLD DIFFERENCES IN GENE EXPRESSION ANALYSIS, 2001,
2. Yufei Tao, Dimensionality Reduction with PCA, Department of Computer Science and Engineering, Chinese University of Hong Kong, <https://www.cse.cuhk.edu.hk/~taoyf/course/cmsc5724/18-fall/lec/pca.pdf>
3. Gallagher N. B, O’Sullivan D., Palacios M., The Effect of Data Centering on PCA Models, 196 Hyacinth Road Manson, WA 98831, [www.Eigenvector.com](http://www.Eigenvector.com)
4. <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>
5. <https://scikit-learn.org/stable/user_guide.html>
6. K. Ming Leung, Naïve Bayesion Classifier, November 28, 2007
7. Gopala Krishna M. N., Bhara K. P., etal, Performance analysis and Evaluation of Different Data Mining Algorithms used for Cancer Classification, International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No.5, 2013
8. *Bahzad T. J., Adnan M. A.,* Classification Based on Decision Tree Algorithm for  
   Machine Learning, *Vol. 02, No. 01, pp. 20 – 28 (2021),* received: 1/11/2021, ISSN: 2708-0757
9. Lior R. and Oded M., DECISION TREES, Department of Industrial EngineeringTel-Aviv University, 03 August 2015, https://www.researchgate.net/publication/225237661
10. Sebastian R., The Effect of Scaling and Mean Centering Prior to a Principal Component Analysis, October 2, 2014
11. Jim Adams, USING THE K-NEAREST NEIGHBOR ALGORITHM, April 8, 2019, link: <https://www.prometheandatasolutions.com/2018files/pdfs/kNN-White-Paper-Adams-V2.pdf>
12. Jingwen Sun,Weixing Du,Niancai Shi, *A Survey of kNN Algorithm*. 2018, Available from: <https://www.researchgate.net/publication/348305327_A_Survey_of_kNN_Algorithm> [accessed Aug 01 2022].,
13. Pierre Gaillard, Lecture Notes on K-Nearest Neighbors, July 2018.
14. Vikramaditya Jakkula, Tutorial on Support Vector Machine (SVM) School of EECS, Washington State University, Pullman 99164.
15. Burges C., “A tutorial on support vector machines for pattern recognition”, In “Data Mining and Knowledge Discovery”. Kluwer Academic Publishers, Boston, 1998, (Volume 2).
16. Nello C. and John S. T., “An Introduction to Support Vector Machines and Other Kernel-based Learning Methods”, Cambridge University Press, 2000.
17. Chih-Wei H., Chih-Chung C., and Chih-Jen L., A Practical Guide to Support Vector Classification, 2003, last updated on: May 19, 2016, Department of Computer Science National Taiwan University,
18. Hassan R. Mohammed A. J. I., Youssef G., Mohamed E., Multilayer Perceptron: Architecture, Optimization and Training, International Journal of Interactive, January 2016, Multimedia and Artificial Intelligence, Vol. 4, Nº1
19. ROSENBLATT F., THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN, Psychological Review Vol. 65, No. 6, 1958.
20. Marius-Constantin P., Valentina E. B., Liliana P., Nikos M., Multilayer Perceptron and Neural Networks. July 2009, Faculty of Electromechanical and Environmental Engineering, University of Craiova
21. <https://deepai.org/machine-learning-glossary-and-terms/relu>
22. Robert E. S.. Yoav F., Adaptive Computation and Machine Learning (Boosting, Foundation and Algorithms), MIT Press, 2012, ISBN 978-0-262-01718-3
23. <https://scikit-learn.org/stable/modules/ensemble.html#adaboost>
24. <https://www.cs.toronto.edu/~mbrubake/teaching/C11/Handouts/AdaBoost.pdf>
25. <https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_multiclass.html# sphx-glr-auto-examples-ensemble-plot-adaboost-multiclass-py>
26. <https://github.com/ritesh-suhag/Classification-Using-PCA>
27. <https://www.biologyforlife.com/t-test.html>
28. <https://matplotlib.org/stable/api/index.html>

<https://seaborn.pydata.org/tutorial.html>

**8.0 Source code snippet**

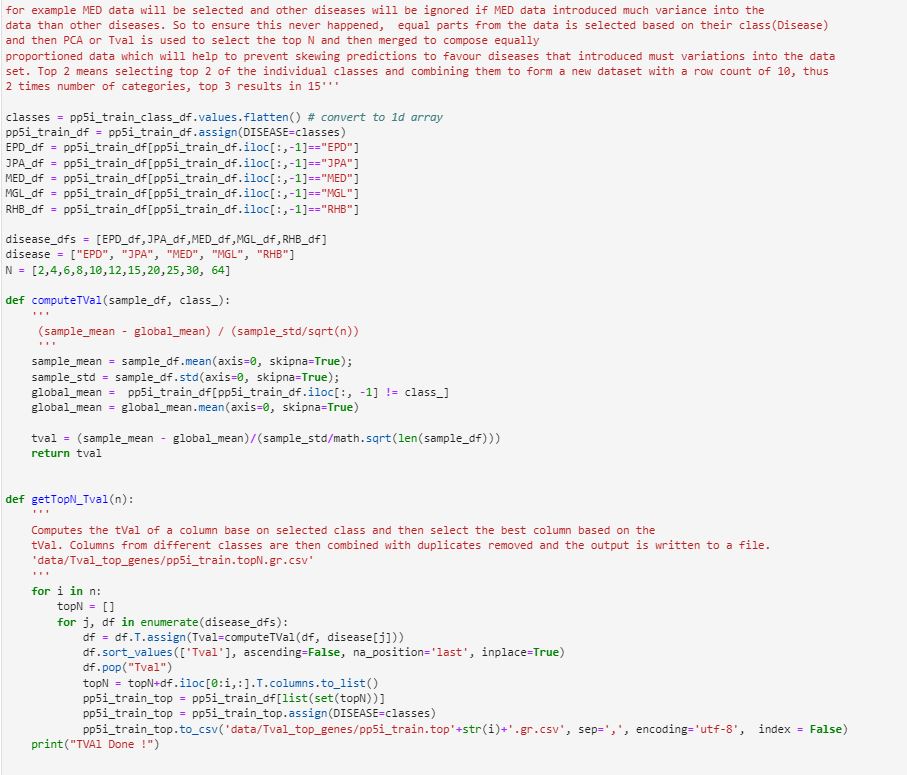


Figure 11. Data cleaning generating T-value

Figure Adaboost



Figure 13 Neural network (Multi-layered perceptron)



Figure 14 Support vector classifier