Predict Disease Classes Using Genetic Microarray Data

Bhoopalsinh Musale
Bishop's University
BMUSALE19@ubishops.ca
Student ID: 002269332

Syed Malik Muzaffar
Bishop's University
msyed19@ubishops.ca
Student ID - 002269955

Abstract:
There have been a lot of researches conducted on prediction and identification of diseases using the genetic microarray data. The classification of cancer at molecular level was performed by T. R Golub, the research in this filed has increased spontaneously. One of the most challenging tasks in the post-genomic era is the identification of disease genes from a vast amount of genetic data. Complex diseases also present a highly heterogeneous genotype, which renders it difficult to identify biological markers. Machine learning methods are widely used to identify those markers, but their performance depends heavily on the size and quality of the data available. The main objective of the study is to predict disease classes using genetic microarray data with help of KNN, Decision Tree, MLP, Naïve Bayes and Ada Boost Classifiers. All these classifiers are also evaluated with different hyperparameters to find out best estimator.

Keywords:
Microarray, KNN, Decision Tree, MLP, Naïve Bayes and Ada Boost

1. Introduction:

Prevention is better than treatment they say and yes, it is true. But in order to prevent we need to identify first. The most important area of medical research is identification of diseases and disease-causing genes. Identification of disease-causing genes can help to diagnose the disease and also, we may find the treatment of the diseases. Till the time it is observed that genetic mutations are a cause of life-threatening diseases. Genetic mutations affect the level of gene expression. Gene expression is the process by which the DNA sequence of a gene is transcribed into RNA. The level of expression of a gene indicates the approximate number of copies of the RNA produced in a cell of that gene and is correlated with the amount of the respective proteins produced [1]. The recent advent of microarray technology has made simultaneous monitoring of thousands of gene expressions possible. Analysing gene expression data can indicate the genes which are differentially expressed in the diseased tissues [2].

In this paper we have used KNN, Decision Tree, MLP, Naïve Bayes and Ada Boost classifiers to predict the disease classes by using the genetic microarray data. We have used Pandas library which is extensively supported by python to load and transpose the data. Next, we performed data cleaning by thresholding both train and test data to a minimum value of 20, maximum of 16,000. For feature selection we have used t-test transformer on each class. The genes with highest t-test values are selected from each gene class. Next, we applied classifiers on the data and then we found out the best model by evaluating all the classifiers.

2. Background:

a) Classification:

Classification is a task involving the use of machine learning algorithms that learn how to apply a class mark to problems domain examples. It is a process whereby a given set of data is categorised into classes and can be performed on both structured and unstructured data. The process begins with predicting the type of data points given. Sometimes the classes are called goals, marks or divisions. Predictive modelling of the classification is the job of approximating the mapping function from input variables to a discrete output variable. The main objective is to classify which class / category the new data falls under.

b) KNN Classifier:

The K-nearest Neighbours (KNN) algorithm is a simple, easy to implement supervised machine learning algorithm which can be used to solve both classification and regression problems. KNN is simple and has no assumptions and training steps involved. KNN is also constantly evolving as it adapts to the newly collected data. Even though it is easy to implement, its slow. It works really well with a low number of input variables but when the variables are increased. It has a hard time predicting new data points.

c) Decision Tree Classifier:

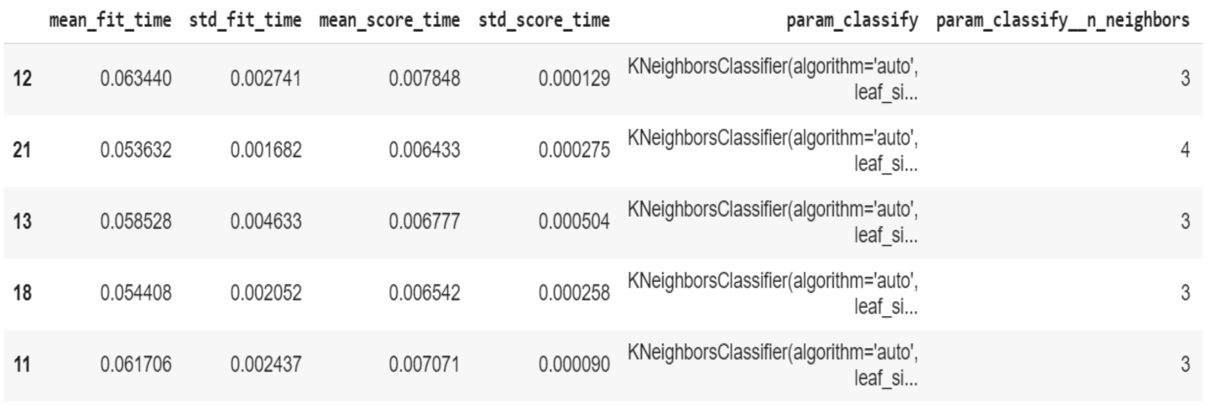
Decision Tree Classifier is a simple classification technique which is widely used. To solve the problem of classification it applies a straightforward theory. Decision Tree Classifier introduces a set of carefully constructed questions about test record attributes. Each time it receives a reply, a follow-up question is asked before it draws a conclusion about the record's class name. The decision trees require less effort during the pre-processing and the missing values don’t have an impact on the decision tree. Even a small change in the data leads to a major structural shift in the decision tree which ultimately leads to instability. They require more time comparatively to train model and also they are complex.

d) MLP Classifier:

MLP Classifier stands for multi-layer Perceptron classifier connected to a Neural Network in the name itself. MLP Classifier depends on an underlying Neural Network to perform the classification function as opposed to other classification algorithms such as Support Vectors or Naive Bayes Classifier. The main advantage of Neural network is that they have the ability to out-perform almost every other machine learning algorithm. And also the ability to work even with missing information. Neural network is often considered as a black box because of its unpredictable nature. It is almost impossible to figure out how it came to a certain conclusion. NN require way more data compared to other machine learning problems which can be solved with less amount of data if using other algorithms.

e) Naïve Bayes Classifier:

Naive Bayes classifiers are a set of Bayes' Theorem-based classification algorithms. It is not a single algorithm but a family of algorithms in which they all share a common principle, i.e. each pair of characteristics being classified is independent of each other. In cases where the independent predictors are true, the



	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_classify	param_classify__n_neighbors
12	0.063440	0.002741	0.007848	0.000129	KNeighborsClassifier(algorithm='auto', leaf_si...	3
21	0.053632	0.001682	0.006433	0.000275	KNeighborsClassifier(algorithm='auto', leaf_si...	4
13	0.058528	0.004633	0.006777	0.000504	KNeighborsClassifier(algorithm='auto', leaf_si...	3
18	0.054408	0.002052	0.006542	0.000258	KNeighborsClassifier(algorithm='auto', leaf_si...	3
11	0.061706	0.002437	0.007071	0.000090	KNeighborsClassifier(algorithm='auto', leaf_si...	3

1: KNN Classifier

Naïve Bayes classifier performs better when compared to the other models. It is easy to implement. The training period is less as it only requires a small amount of training data to estimate the test data. It relies too much on independent predictors.

f) Ada Boost Classifier:

The classifier Ada-boost blends a weak classifier algorithm to form a solid classifier. A single algorithm could misclassify the objects. But if we combine multiple classifiers with selection of training set at each iteration and assign the right amount of weight in the final vote, we can have good overall classifier accuracy score. It is so flexible that it can be combined with any machine learning algorithms. It is fast, simple and easy to program. It is vulnerable to uniform noise and also the weak classifiers tend to lead to low margins and overfittings.

3. Methodology:

Step 1- Load Data: In this step we load the gene data which has comma-separated values. Using the Pandas library which is extensively supported by python, we load the data and perform the transpose of the data.

Step 2- Data Cleaning: Here we threshold the training and testing data to a minimum value of 20 and a maximum of 16,000.

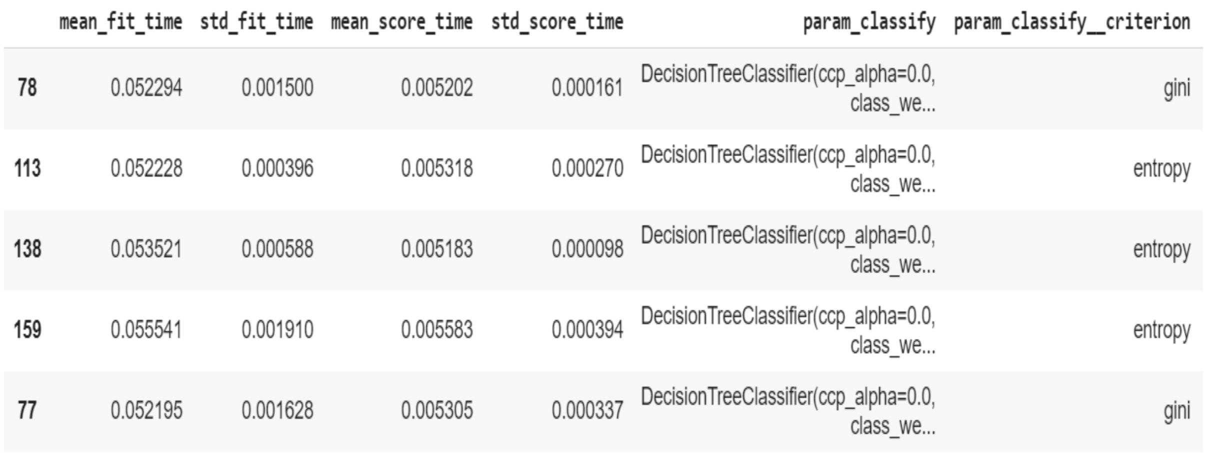
Step 3- Feature Selection and T- test transformer: Function Selection selects features that contribute significantly to our predictive output. This project is an important step. For the performance of the set of features in this project, we carried out a t-test for each class from that class sample and in the remaining classes. For each gene class the genes with the highest absolute t-values are picked. We got t-values measured as:

$$t = \frac{\overline{X_1} + \overline{X_2}}{\sqrt{s_1^2/_{n1} + s_2^2/_{n2}}}$$

Welch's t-test can be used when two population variances are not considered to be equal. Transformer which is compatible with sklearn.pipelines is introduced. For each class, it selects top w features with the highest t-scores, and selects features using t-score. Implementation of scypy library t-test has been used here.

Step 4- Classification and their Individual Result:

1. KNN (K = 2, 3, 4):



	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_classify	param_classify__criterion
78	0.052294	0.001500	0.005202	0.000161	DecisionTreeClassifier(ccp_alpha=0.0, class_we...	gini
113	0.052228	0.000396	0.005318	0.000270	DecisionTreeClassifier(ccp_alpha=0.0, class_we...	entropy
138	0.053521	0.000588	0.005183	0.000098	DecisionTreeClassifier(ccp_alpha=0.0, class_we...	entropy
159	0.055541	0.001910	0.005583	0.000394	DecisionTreeClassifier(ccp_alpha=0.0, class_we...	entropy
77	0.052195	0.001628	0.005305	0.000337	DecisionTreeClassifier(ccp_alpha=0.0, class_we...	gini

2: Decision Tree Classifier

KNN is supervised learning algorithm. We are given a training dataset here that has training observations (x, y) and would like to capture the relationship between x and y. Our motive here is to learn a function h: X→Y so that h(x) can predict the corresponding output y when given an unseen observation x. Figure 1 shows results of KNN classifier with K= [2,3,4] and K=3 gives highest mean test score. But we have tried different values of K also.

2. Decision Tree Classifier:

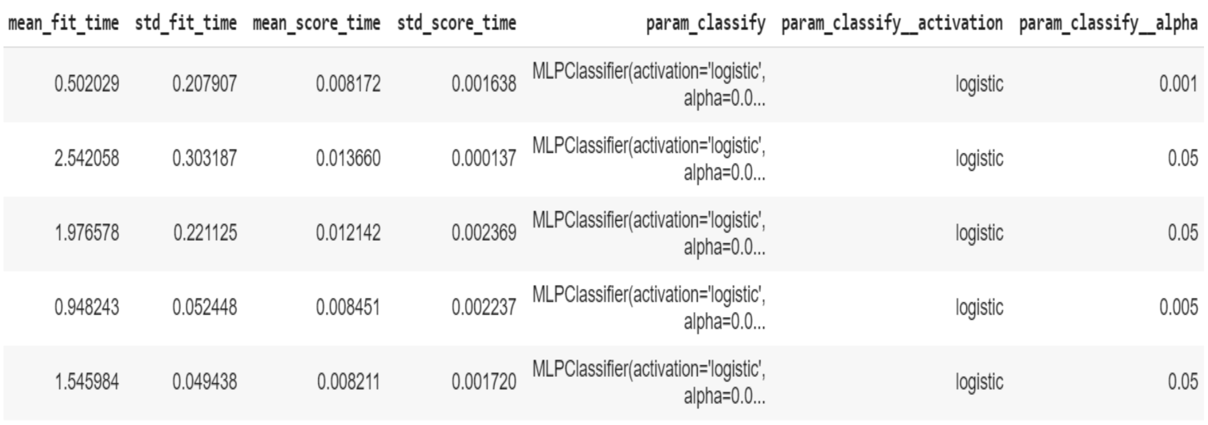
By learning basic decision rules that are placed from prior data, i.e. training data, the Decision Tree model may construct a training model that predicts the target variable class or value. We start from a root of tree to predict class label for a record. Then we equate root attribute values with attribute of record. Using comparison, we can follow the corresponding branch of value and jump to next node. Figure 2 shows the output for the classifier.

3. MLP Classifier:

Implementation of MLP Classifier takes no more effort than implementation of Support Vectors or Naive Bayes or any other classifiers from Scikit-Learn. Figure 3 shows the results for MLP.

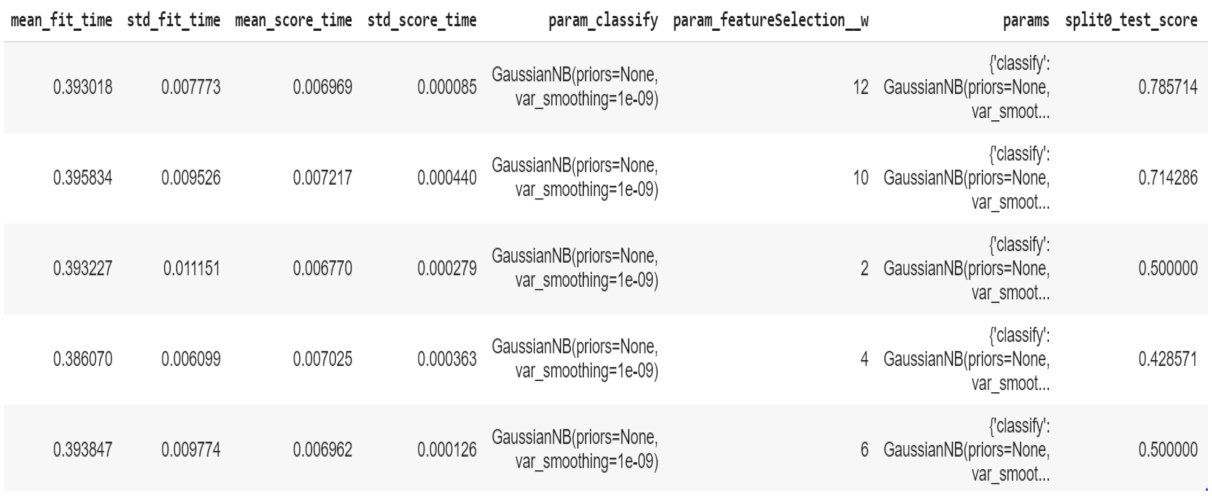
4. Naïve Bayes Classifier:

Naive Bayes, a probabilistic model of machine learning which is used for classification tasks. The classification is based on the theorem of the Bayes.



mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_classify	param_classify__activation	param_classify__alpha
0.502029	0.207907	0.008172	0.001638	MLPClassifier(activation='logistic', alpha=0.0...	logistic	0.001
2.542058	0.303187	0.013660	0.000137	MLPClassifier(activation='logistic', alpha=0.0...	logistic	0.05
1.976578	0.221125	0.012142	0.002369	MLPClassifier(activation='logistic', alpha=0.0...	logistic	0.05
0.948243	0.052448	0.008451	0.002237	MLPClassifier(activation='logistic', alpha=0.0...	logistic	0.005
1.545984	0.049438	0.008211	0.001720	MLPClassifier(activation='logistic', alpha=0.0...	logistic	0.05

3:MLP Classifier



mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_classify	param_featureSelection__w	params	split0_test_score
0.393018	0.007773	0.006969	0.000085	GaussianNB(priors=None, var_smoothing=1e-09)	12	{'classify': GaussianNB(priors=None, var_smoot...	0.785714
0.395834	0.009526	0.007217	0.000440	GaussianNB(priors=None, var_smoothing=1e-09)	10	{'classify': GaussianNB(priors=None, var_smoot...	0.714286
0.393227	0.011151	0.006770	0.000279	GaussianNB(priors=None, var_smoothing=1e-09)	2	{'classify': GaussianNB(priors=None, var_smoot...	0.500000
0.386070	0.006099	0.007025	0.000363	GaussianNB(priors=None, var_smoothing=1e-09)	4	{'classify': GaussianNB(priors=None, var_smoot...	0.428571
0.393847	0.009774	0.006962	0.000126	GaussianNB(priors=None, var_smoothing=1e-09)	6	{'classify': GaussianNB(priors=None, var_smoot...	0.500000

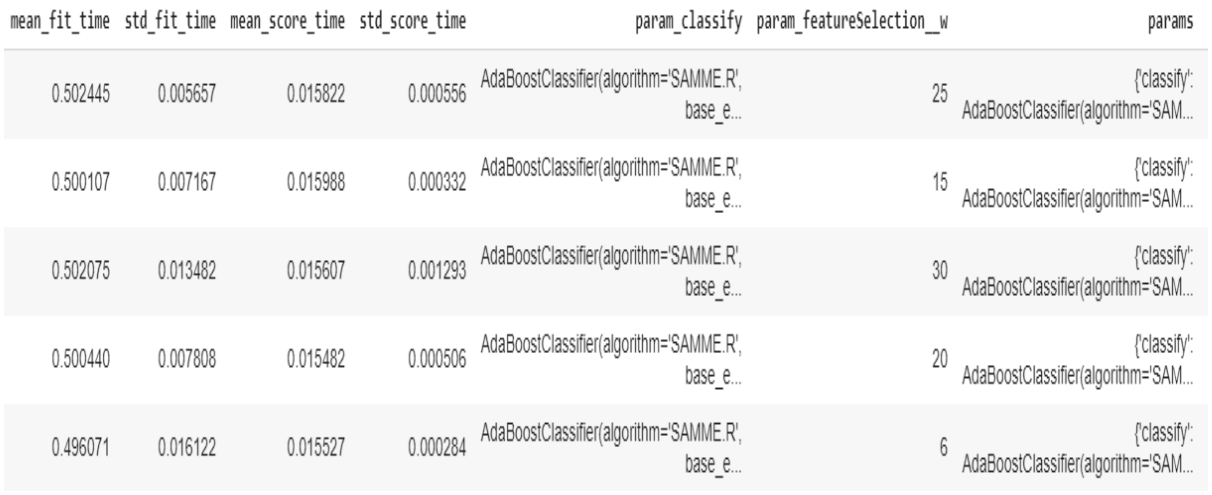
4: Naive Bayes Classifier

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can determine the likelihood that A will happen, given that B has happened. Here the hypothesis is A, and proof is B. Here the presumption is that the functions are autonomous. The presence of one trait won't affect the other. So, it's called naive. Figure 4 shows output for Naïve Bayes Classifier.

5. AdaBoost Classifier:

Bagging indicates grouping of the bootstrap. Thus, that the uncertainty of predictions it is a mixture of several learners. For example, M Decision Tree random forest trains, you can train M different trees on different random sub-sets of the data and vote for final prediction. Bagging consists of the Random Forest and Extra Trees algorithms. Boosting algorithms, set of the low accurate classifier combined to create a highly accurate classifier. Low accuracy classifier (or weak classifier) offers the accuracy better than we can say the flipping of a coin. Highly accurate classifier (or strong classifier) offers error rate closer to 0. This algorithm is able to track the model who failed accurate prediction. The following three



mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_classify	param_featureSelection__w	params
0.502445	0.005657	0.015822	0.000556	AdaBoostClassifier(algorithm='SAMME.R', base_e...	25	{'classify': AdaBoostClassifier(algorithm='SAM...
0.500107	0.007167	0.015988	0.000332	AdaBoostClassifier(algorithm='SAMME.R', base_e...	15	{'classify': AdaBoostClassifier(algorithm='SAM...
0.502075	0.013482	0.015607	0.001293	AdaBoostClassifier(algorithm='SAMME.R', base_e...	30	{'classify': AdaBoostClassifier(algorithm='SAM...
0.500440	0.007808	0.015482	0.000506	AdaBoostClassifier(algorithm='SAMME.R', base_e...	20	{'classify': AdaBoostClassifier(algorithm='SAM...
0.496071	0.016122	0.015527	0.000284	AdaBoostClassifier(algorithm='SAMME.R', base_e...	6	{'classify': AdaBoostClassifier(algorithm='SAM...

5:Ada Boost Classifier

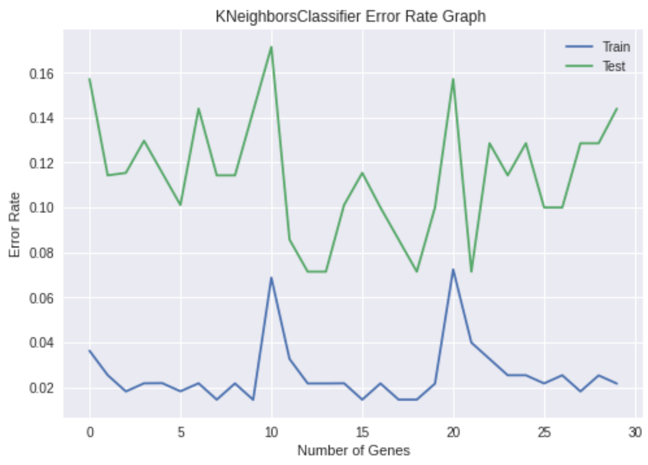
algorithms have gained massive popularity in data science competitions: AdaBoost (Adaptive Boosting), Gradient Tree Booting and XGBoost. Stacking (or stacked generalization) is an ensemble learning technique. It combines multiple base classification models' predictions to form a new data set. This new data is treated as the input data to another classifier. The classifier is employed to solve this problem. Stacking is also called as blending.

Figure 5 shows the output for ada boost classifier.

Step 5- Best Model Selection: We evaluate all above classifiers and with different hyperparameters to find out best estimator. For this, we made use of sklearn's Pipeline class. It sequentially applies a list of transforms and a final estimator. Intermediate steps of the pipeline must implement fit and transform methods. The final estimator only needs to implement fit. The transformers in the pipeline can be cached using memory argument. The motive of the pipeline is to assemble several steps that can be cross validated altogether while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name separated by a '__'. A step's estimator may be replaced by setting the parameter with its name to another estimator, or a transformer removed by setting it to 'passthrough' or None. Feature selection and the model training are done using cross-validation, in order to avoid data leakage. We were able to find that MLP is the best model in this study.

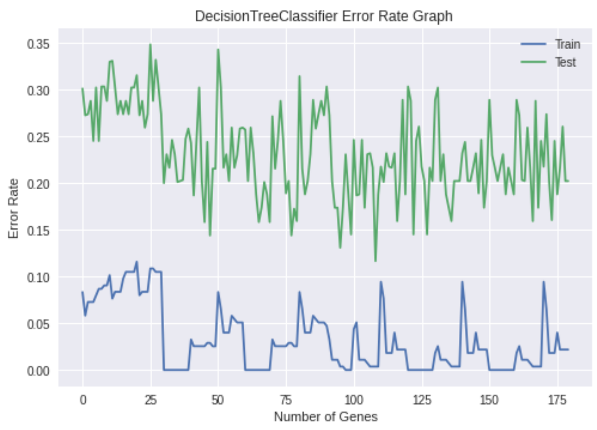
4. Experimental Results:

Figure 6 shows the error rate graph for KNN classifier. We have plotted number of genes on Y- axis and error rate on X- axis.



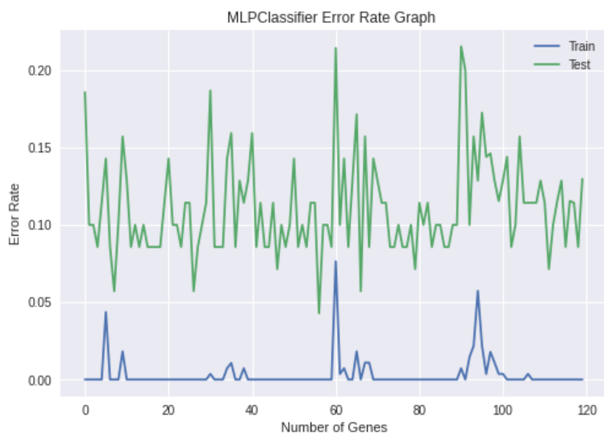
6:Graph for KNN

Figure 7 shows the error rate graph for Decision Tree classifier. We have plotted number of genes on Y- axis and error rate on X- axis.



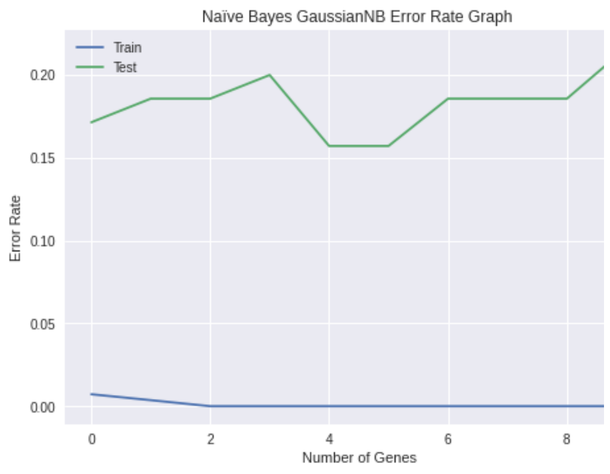
7: Graph for Decision Tree

Figure 8 shows the error rate graph for MLP classifier. We have plotted number of genes on Y- axis and error rate on X- axis.



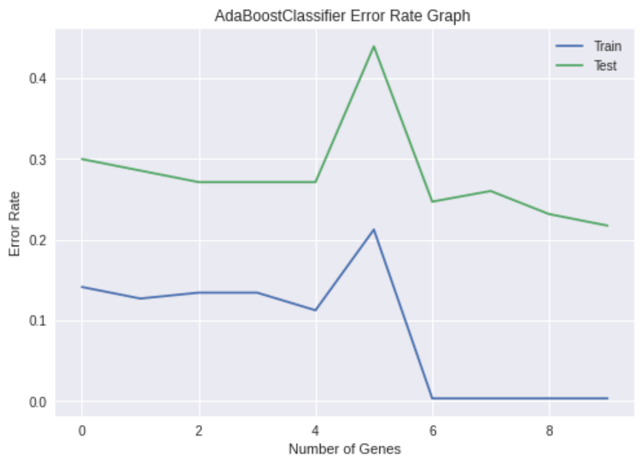
8: Graph for MLP

Figure 9 shows the error rate graph for Naïve Bayes classifier. We have plotted number of genes on Y- axis and error rate on X- axis.



9: Graph for Naïve Bayes

Figure 10 shows the error rate graph for Ada Boost classifier. We have plotted number of genes on Y- axis and error rate on X- axis.



10: Graph for Ada Boost

```
best_estimator_ = model.best_estimator_
best_estimator_
```

```
Pipeline(memory='/tmp/tmp_ab1za2f',
         steps=[('featureSelection', TScoreSelection(w=15)),
                ('classify',
                 MLPClassifier(activation='logistic', alpha=0.05,
                               batch_size='auto', beta_1=0.9, beta_2=0.999,
                               early_stopping=False, epsilon=1e-08,
                               hidden_layer_sizes=(64,),
                               learning_rate='constant',
                               learning_rate_init=0.001, max_fun=15000,
                               max_iter=1000, momentum=0.9, n_iter_no_change=10,
                               nesterovs_momentum=True, power_t=0.5,
                               random_state=None, shuffle=True, solver='lbfgs',
                               tol=0.0001, validation_fraction=0.1,
                               verbose=True, warm_start=False))],
         verbose=False)
```

11: Best Estimator (MLP)

Figure 11 shows the best estimator as MLP classifier with maximum accuracy compared to the rest of the classifiers.

5. Conclusions:

In this paper we have predicted the disease classes using genetic microarray data by using KNN, Decision Tree, MLP, Naïve Bayes and Ada Boost classifiers. All the classifiers were able to predict the disease classes with different accuracy scores. Out of those MLP was evaluated as the best model because the accuracy of the MLP model was higher than other classifiers.

References:

1. Weaver RF. Molecular Biology. Boston: McGraw-Hill; 2003.
2. Zhang A. Advanced Analysis of Gene Expression Microarray Data. Danvers: World Scientific Publishing Co; 2006.
3. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). 2012; 771-778
4. Piatetsky-Shapiro, G. (1996). Advances in knowledge discovery and data mining (Vol. 21). U. M. Fayyad, P. Smyth, & R. Uthurusamy (Eds.). Menlo Park: AAAI press.
5. https://www.biologyforlife.com/t-test.html
6. Hand, D. J. (2007). Principles of data mining. Drug safety, 30(7), 621-622
7. https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06
8. Sujata Joshi, A Deeptha, K Prathibha, N Hema, J Priyanka; Classification and Prediction of Disease Classes using Gene Microarray Data; International Journal of Data Mining Techniques and Applications Volume 5, Issue 1, June 2016, Pages: 7-10 ISSN: 2278-2419
9. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3037837/
10. R. L. Somorjai, B. Dolenko and R. Baumgartner; Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions; Bioinformatics; Vol. 19 no. 12 2003, pages 1484–1491 DOI: 10.1093/bioinformatics/btg182