Disease Prediction on Genetic Microarray Data using Machine Learning Algorithms

***Abstract*— One of the leading fields for the implementation of machine learning is healthcare. Most medical facilities and research institutes strives to be advanced, so that they can make better decision for patient diagnosis and care. The machine learning predictions models enables us to process volume of complex medical datasets and provides better predication on disease diagnosis. Thus, identifying various critical and terminal diseases, allowing medical professionals to start treatment at an earlier stage. Any form of disease prediction would significantly increase the treatment progress which could avoid terminal stage or even help in identification of a potential outbreak. This paper mainly focuses on working on a genetic dataset obtained from a gene microarray, to better understand how different classifiers can be applied on it and to compare the results.**

Keywords — Disease Prediction, Machine Learning Classifiers, Neural Network, Genetic data.

I. INTRODUCTION

Our current time poses a unique opportunity to explore the massive data obtained from various healthcare systems and research institutes. Understanding the genetic makeup of a disease can lead to tremendous development in treatment of the disease and even could be of great medical value in preventing an epidemic outbreak.

Data mining technique has been widely used in the past decades to provide various clinical systems that aids with the prediction and diagnosis of various diseases with high accuracy rate. These techniques are highly effective as every disease has a naturally occurring genetic makeup, which could be defined and identified based on the collected gene samples for any disease. Such clinical prediction system would help medical professionals give patients accurate diagnosis and warning about their disease. The goal of our project is to train various classifiers on the provided genetic data for disease classification. Each classifier works well on certain type of datasets and finding the best suitable prediction classifier for the given dataset.

The scope of this project is primarily to analyze the performance of disease prediction approaches using variants of supervised machine learning classifiers. Our focus involved selecting the specific genes of the various classes of disease to obtain the best accuracy on a predictive disease modelling. Classification models used in this project utilized labelled training data separated from the gene dataset. Comparison of results from the different supervised machine learning algorithms would be more accurate, comprehensive, and less prone to biased results

II. DATA AND METHODOLOGY

The dataset used for this project was collected from DNA microarray which measures expression levels for large number of genes simultaneously. The data set contains the below details to be trained and tested by the prediction models.

No of samples: 69 in training dataset and 23 in test dataset
No of features: 7070 Gene Expressions
No of classes (targets): 5 (MED, RHB, MGL, EPD, JPA)

The microarray dataset was preprocessed in following steps for pre-processing and cleaning of the training dataset.

A. *Loading and processing dataset*

The data set was split into training and testing files 'pp5i_train.gr.csv' and 'pp5i_test.gr.csv' which was kept locally and loaded into the python source code using the pandas library. The library allows us to do data manipulations and analysis, used specially for reading data from csv files.

B. *Label classes*

The target classes were loaded from 'pp5i_train_class.txt' and was encoded into individual labels using 'preprocessing' from Scikit-learn library

C. *Remove fold difference*

The genes with less than two-fold difference were removed from the test samples. The minimum, maximum

threshold for the gene expression values were limited between 2 and 16000

D. Extract subsets of top genes in each class

For the test samples, the ranks (ANOVA F-value) were generally based on the provided features (genes) (Fig 2.1). The 'feature_selection' sub-library from Scikit-learn library is used of this purpose.

Training Data:

	SNO	1	2	3	4	...	66	67	68	69	rank
2506	U00921_at	20	20	20	20	...	118	115	102	119	222.388504
6528	U59877_s_at	20	45	20	20	...	923	1358	1081	1120	166.874021
6663	AF000424_s_at	20	20	20	20	...	73	102	94	92	162.284150
5503	D13631_s_at	21	54	20	31	...	331	396	314	293	148.242194
3688	U79242_at	20	20	20	20	...	153	192	185	121	139.143388
...	...	...	..	...	...	...	...	...	...	...	...
6861	X54489_rna1_at	20	20	20	20	...	20	20	20	20	0.102060
1241	L13042_at	50	53	48	42	...	48	40	41	30	0.099092
1658	M14159_cds2_at	48	85	108	27	...	20	62	48	38	0.092244
3355	U56816_at	122	73	79	120	...	114	73	76	103	0.081504
4156	X51757_at	20	20	20	21	...	50	25	20	32	0.065644

[6413 rows x 71 columns]

Fig 2.1 – Ranking samples in training dataset

The data is then sorted to extract the subsets of top genes. The subsets contain top 2,4,6,8,10,12,15,20,25, and 30 top genes with the highest absolute T-value.

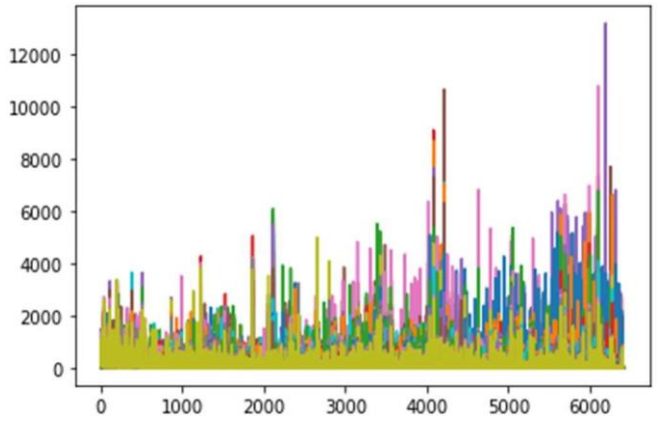


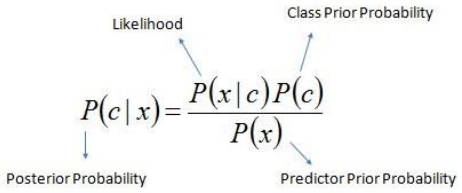
Fig 2.2 – Data distribution of gene samples

III. APPLYING MACHINE LEARNING CLASSIFIERS

Each machine learning classifier uses its programmed algorithm that gets trained and optimized by passing input data to obtain prediction value within defined acceptable range. A labelled dataset is first used to train the underlying algorithm in a classifier, then the unlabeled test data is fed to categorize them under similar trained groups.

A. Gaussian Naïve Bayes Classifier

Naive Bayes Classifiers are based on the Bayes Theorem which is used to calculate conditional probability.



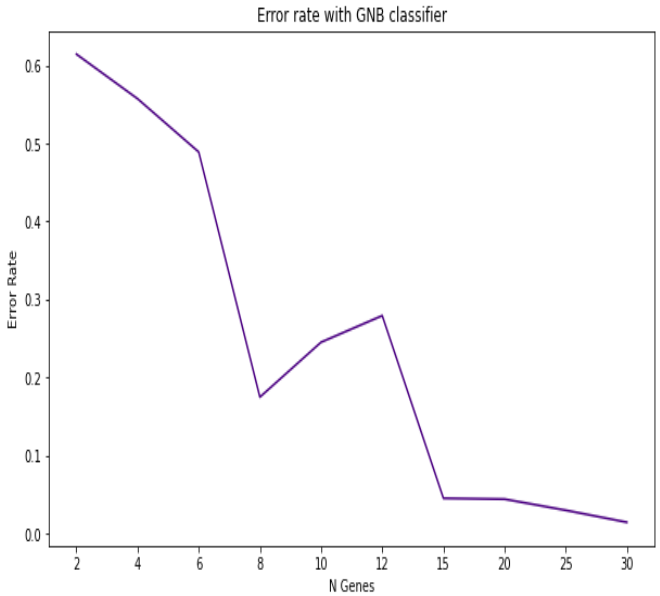
$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

One assumption taken is the strong independence assumptions between the features. Gaussian Naive Bayes is a variant of naïve Bayes that follows Gaussian normal distribution and supports continuous data. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently, and they need only a small training data to be trained for parameter estimation needed for classification. When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal/Gaussian distribution.

The likelihood of the features is assumed to be

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Naïve Bayes classifier works effectively for classifying emails, texts, symbols, and names. But is it very difficult to get the set of independent predictors for developing a model using Naive Bayes.



Library used: *GaussianNB* from scikit-learn (Python)

B. Decision Tree Classifier

A Decision Tree classifier algorithm works on the principle of continuously splitting data according to certain split criteria. The data is split into nodes(features) that provides largest information gain; this happens until the leaf nodes of the tree is pure. The depth of the tree is restricted to avoid the overfitting.

The performance of a decision tree classifier depends on how well the tree is constructed from the training data. Starting from the root node, the node is split into subsets based on feature values. Our decision tree classifier uses C4.5 algorithm which measures information as 'gain' and 'gain ratio'

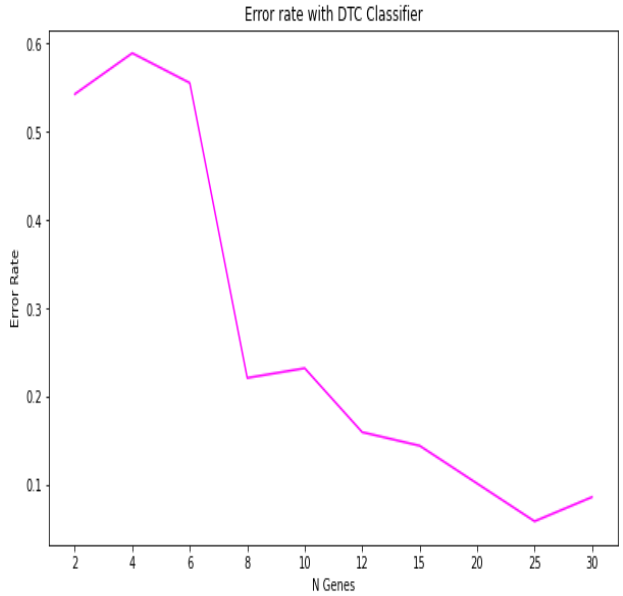
If S is Current state, and Pi is Probability of an event *i* of state S or Percentage of class *i* in a node of state S. Then, entropy is

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

If X is the feature and T is its current state, then gain is

Information Gain(T,X) = Entropy(T) - Entropy(T, X)

The main advantage of the decision tree classifier is its ability to using different feature subsets and decision rules at different stages of classification



Library used: *DecisionTreeClassifier* from scikit-learn (Python)

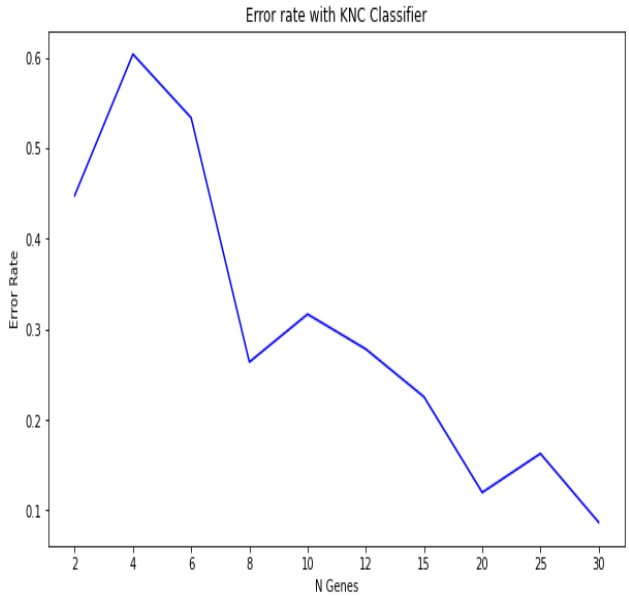
C. K Nearest Neighbor Classifier

The KNN classifier predicts and groups a new sample's class based on the Euclidian distance *d* calculated with the labelled trained data, the distance between two points is calculated by the below expression, if n is the number of dimensions or features the distance is formulated by the following expression

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

In KNN classifier, the model structure determined from the dataset and it is very helpful in practice where most of the real-world datasets do not follow mathematical theoretical assumptions. Training data used mainly in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data



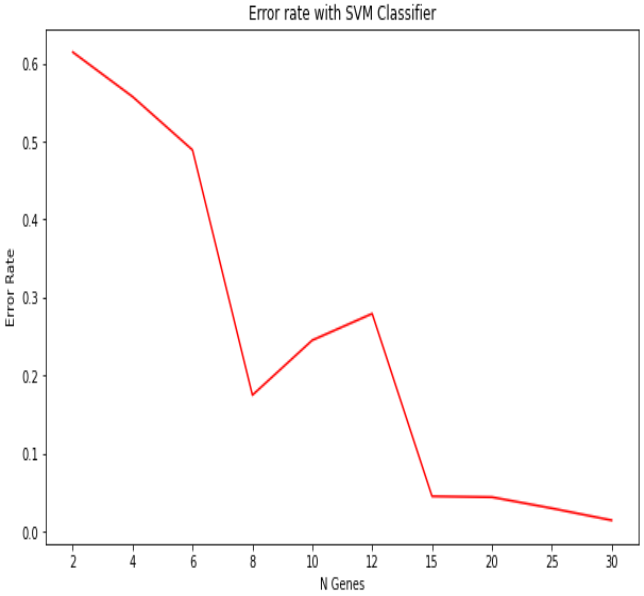
Library used: *KNeighborsClassifier* from scikit-learn (Python)

D. Support Vector Machine

SVM: Support Vector Machine is a supervised classification algorithm where we draw a line between two different categories to differentiate between them. SVM is also known as the support vector network.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.



Library used: SVC from scikit-learn (Python)

E. Neural Network - Multi-Layer Perceptron

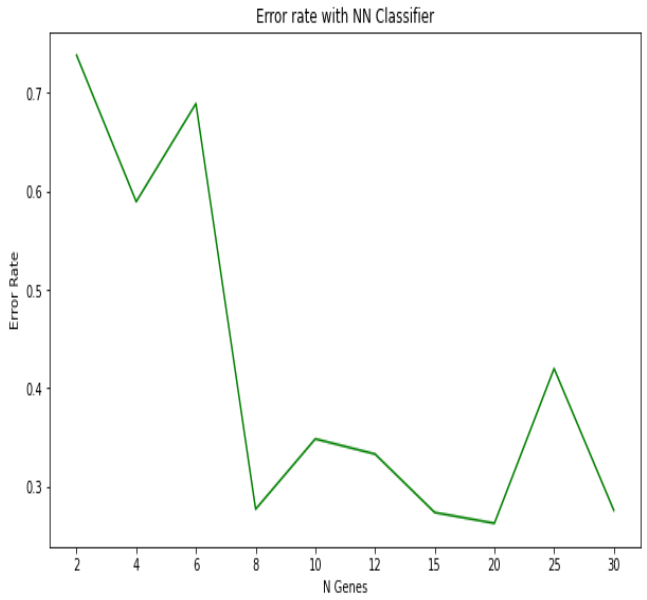
Multilayer perceptron contains collection of perceptron (Neurons), which are simple computational units that have weighted input and produces outputs using an activations function (ReLu).

Multiple neurons are arranged in layers (Input, Hidden and Output layers). Input data fed into the neural network are numerical values and these inputs are to be scaled in a consistent way. The weights in the networks are updated from the error calculated for each training sample, for higher efficient learning process the weight changes are saved and updated as batches in the network. Once trained, the network can be used to make predictions

For our training samples, our neural network model is set to the below hyperparameters which predicts the disease class with high accuracy rate.

Neurons – 25
Hidden Layers – 25
Learning Rate – 0.01
Activation Function – ReLu
Max Iterations - 250

Neural Networks has the ability to learn and model nonlinear and complex relationships, they don't usually have any restrictions of how the input data should be normalized, the ability t0 12o generalize data by the model enables them to predict on unseen data to infer unseen relationships.



Library used: *MLPClassifier* from scikit-learn (Python)

IV. RESULTS AND MODEL COMPARISON

A. Model Accuracy and Error rates of Machine Learning Classifiers

Almost all the classifiers used on for this project, provides good prediction accuracies with less error rates, if fed input genes of N>8 top ranking samples.

As illustrated in the graph (Fig 4.2), the error rate of the classifiers tends to reduce on the higher N-genes values.

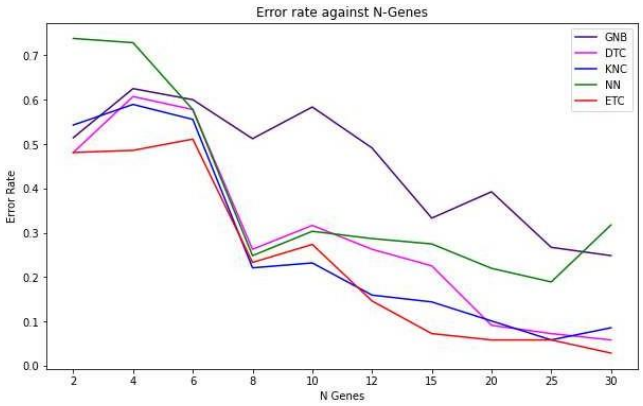


Fig 4.2 - Error rates against N-Genes

Among the classifiers trained, the SVM classifier predict the disease class with best accuracy rate of 98.5% for the subset of top gene samples with N value of 30.

```
Best N          :  30
Best Clasifier  :  SVC
Best Accuracy   :  0.9857142857142858
```

Fig 4.2 – Result of Best Classifier with N value

For the unlabeled test dataset given to us for prediction, we fed it into the classification model and obtained the below prediction results

```
Test dataset predictions :
 ['MED' 'EPD' 'MED' 'MED' 'EPD' 'MED' 'MED' 'MED' 'EPD'
 'MED' 'MED' 'MED' 'MED' 'EPD' 'MED' 'MED' 'RHB' 'EPD'
```

Fig 4.3 – Predicted results of test dataset

V. CONCLUSION

In this project, we developed and compared several machine learning classifiers for predicting disease using dataset collected from gene microarray. The classifiers are trained in the labelled training gene samples and predicted on the provided unlabeled test sample. The most efficient classifier among them was identified as Support vector machine with best accuracy rate. Based on the proposed classification model, the disease prediction can be done for any sample collected over the microarray and the patient can be diagnosed in a most efficient manner. As a future work to this project, we can assess the classifiers on more datasets and disease classes, examining its efficiency on predicting the disease on more complicated gene datasets.

VI. REFERENCES

[1] Decision Trees As Probabilistic Classifiers by J.R.Quinlan, Proceedings of the Fourth International Workshop on machine learning

[2] Machine learning and its application in microscopic image analysis, F. Xing, L. Yang, Machine Learning and Medical Imaging, 2016

[3] Extremely randomized trees, Pierre Geurts, Damien Erns, Louis Wehenkel,, Springer Science, 2005

[4] A Multilayer Perceptron-Based Medical Decision Support System for Heart Disease Diagnosis, Hongmei Yan, Yiangtao Jiang, Jun Zheng, Chenglin Peng, Qinghui Li, 2006.

[5] Prediction System for Heart Disease Using Naive Bayes, Shadab Adam Pattekari, Asma Parveen, International Journal of Advanced Computer and Mathematical Sciences, 2012, pp 290-294,