Disease Prediction Using Machine Learning

Vivek Kumar Bharati, 22M1081[EE1]; Aman Patel, 22M1087[EE2]; Arif Raza, 22M1088 [EE2]
Under the Guidence of Prof. Amit Sethi, Electrical Engineering Department, IIT Bombay

***Abstract*—The paper mainly focusing on prediction of the disease of a human based on the symptoms that the person possesses. So, we are implementing robust Machine Learning Model to predict it. In this, we are predicting with 6 models, which are SVM(Support Vector Machine), RBF kernel SVM, KNN(K-Nearest Neighbors), Random Forest, Gaussian Naive Bayes and Decision Tree. We will be using a confusion matrix to determine the quality of the models. After training the six models with the same data-set, we will be predicting the disease for the input symptoms by combining the predictions of all six models. This makes our overall prediction more robust and accurate.**

***Index Terms*—SVM, RBF kernel SVM, KNN, Random Forest, Gaussian Naive Bayes, Decsion Tree, Confusion Matrix**

I. Introduction

Nowadays, we have Machine Learning by which we can predict many things like we can estimate the value of a house based on the features. In the same way, we can estimate good idea that a person is infected by some disease or not. To estimate disease, we have to know the symptoms and based on the given symptoms we create a data-sets in which many features are there, based on the features Machine Learning Algorithms create a formula to predict whether a person is infected or not. If infected then this algorithms also predict with what disease a person is going through and for this all we need is data-sets. In this project, we are using the data-sets from https://www.kaggle.com/kaushil268/disease-prediction-using-machine-learning. First, we prepare the data which is primary steps of any Machine Learning Problem. In our data-sets there are 133 total columns out of which 132 columns represent the symptoms and the last column is the prognosis. second, we pre-process the data to improve the quality of Machine Learning Model because some features are containing null- values. So, either we remove the rows but it is the best instead we impute the value. But in the data which we are using in this project contains no null value, that means no need of imputation. In our data-set all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a label encoder. After gathering and pre-processing of data-set, the data is ready and can be used to train a machine learning model. We will be using a confusion matrix to determine the quality of the models. After training the six models mentioned in the index terms, we will be predicting the disease for the input symptoms by combining the predictions of all six models. This makes our overall prediction more robust and accurate.

II. Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. In SVM we are minimizing weight w and biases b:

$$\underset{w,b}{min}\frac{1}{2} \parallel w \parallel^2 + c\frac{1}{n}\sum_i \xi_i \tag{1}$$

subject to

$$\begin{cases} y_i(x.w+b) \geq (1-\xi_i) \quad \text{for} \quad i = 1, \cdots, n \\ \xi_i \geq 0 \quad \text{for} \quad i = 1, \cdots, n \end{cases} \tag{2}$$

III. The RBF Kernel

RBF short for Radial Basis Function Kernel is a very powerful kernel used in SVM. Unlike linear or polynomial kernels, RBF is more complex and efficient at the same time that it can combine multiple polynomial kernels multiple times of different degrees to project the non-linearly separable data into higher dimensional space so that it can be separable using a hyperplane.
The RBF kernel works by mapping the data into a high-dimensional space by finding the dot products and squares of all the features in the dataset and then performing the classification using the basic idea of Linear SVM. For projecting the data into a higher dimensional space, the RBF kernel uses the so-called radial basis function which can be written as:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2}) \tag{3}$$

Here $||X_1 - X_2||^2$ is known as the Squared Euclidean Distance and σ is a free parameter that can be used to tune the equation.

The equation is really simple here, the Squared Euclidean Distance is multiplied by the $\frac{1}{2\sigma^2}$ parameter and then finding the exponent of the whole. This equation can find the transformed inner products for mapping the data into higher dimensions directly without actually transforming the entire dat-aset which leads to inefficiency. And this is why it is known as the RBF kernel function.

IV. K-Nearest Neighbor(KNN)

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the data set and at the time of classification, it performs an action on the data-set. KNN algorithm at the training phase just stores the data-set and when it gets new data, then it classifies that data into a category that is much similar to the new data. Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category. Euclidean Distance is calculated by the given formula where d is the number of points and x_{1i} is the points in n-dimensional space.

Euclidean Distance =

$$(\sum_{i=1}^{d}(x_{1i} - x_{2i})^2)^{\frac{1}{2}} \tag{4}$$

V. Random Forest

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.
A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms. The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome. A random forest eradicates the limitations of a decision tree algorithm. It reduces the over-fitting of data-sets and increases precision.

VI. Decision Forest

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree, classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification. A decision tree which is also known as prediction tree refers a tree structure to mention the sequences of decisions as well as consequences. Considering the input $X = (X_1, X_2, \dots X_n)$, the aim is to predict a response or output variable Y. Each element in the set $(X_1, X_2, \dots X_n)$ is known as input variable.It is possible to achieve the prediction by the process of building a decision tree which has test points as well as branches. At each test point, it is decided to select a particular branch and traverse down the tree. Ultimately, a final point is reached, and it will be easy to make prediction. In a decision tree, all the test points exhibit testing specific input variables (or attributes), and the developed decision tree is represented by the branches. Because of flexibility as well as simple visualization, decision trees are mostly probably deployed in data mining applications for the purpose of classification. In the decision tree, the input values are considered as categorical or continuous.

VII. Gaussian Naive Bayes Classifier

Naive Bayes Classifiers are based on the Bayes Theorem. One assumption taken is the strong independence assumptions between the features. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. Naive Bayed classifiers need a small training data to estimate the parameters needed for classification. Naive Bayes Classifiers have simple design and implementation and they can applied to many real life situations. When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \tag{5}$$

where, σ_y is the variance and μ_y is the mean from x_i. Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution. An approach to create a simple model is to assume that the data is described by a Gaussian distribution

with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

VIII. EXPLORATORY DATA ANALYSIS

This section contains result of different exploratory data analysis in which we checked our data-sets contains null values on not. So, In the beginning of introduction it is stated that our data-set contains no null values so no data imputation is required. Now, we checked the variance in different feature[Fig.1] and then checking for balanced class classification[Fig.2] which shows it is balanced class classification. Heat Map is shown in Fig.3.

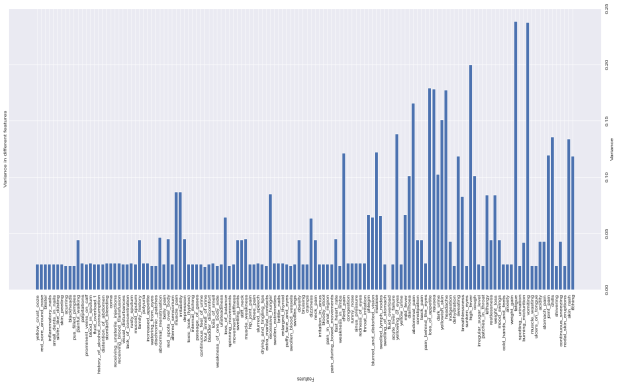

Fig. 1: Variance in different features.

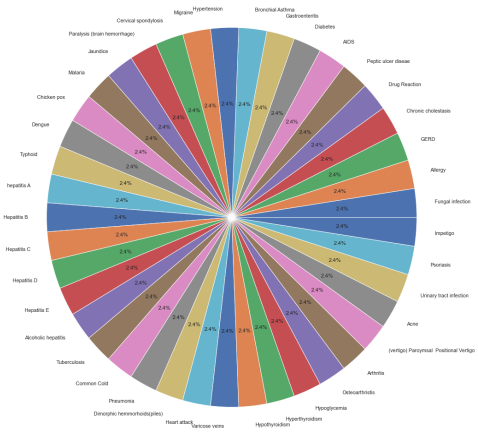

Fig. 2: Balanced Class Classification.

IX. BUILDING MODELS AND SELECTING OPTIMAL MODEL

We tested our data on the above six models. For Linear SVM Classifier we got accuracy of 100 percentage on test data. After that we performed Hyper-parameter Tuning to get the optimal Hyper-parameter values. Similarly we test each model on training data and got accuracy of 100 percentage for RBF kernel SVM, 96 percentage for Random Forest, 100

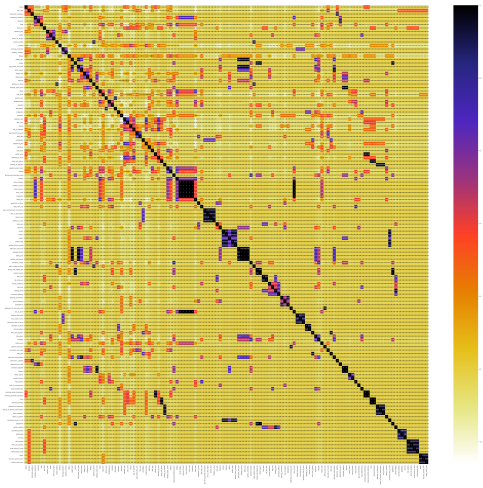

Fig. 3: Heat Map showing correlation between features.

percentage for Gaussian Naive Bayes, 100 percentage for KNN and 86 percentage for Decision Tree Classifier which is shown in Fig.4 As shown in Fig.4, it is depicted that

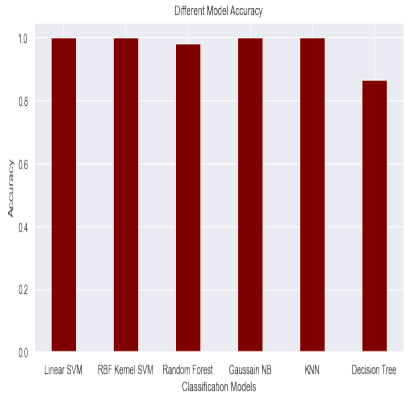

Fig. 4: Heat Map showing correlation between features.

Other five models performs better except Decision Tree.

X. FEATURE ENGINEERING AND FEATURE SELECTION

Now, our aim is to reduce the computational time without loosing accuracy which is achieved by feature engineering also called important feature extraction. The least important variables are eliminated while the complete collection of features is used to form the model in recursive feature elimination, a backward selection strategy. The least significant variables are again removed when the model is rebuilt, and so on. Here, the tuning parameter is the subset size. The final model is then trained using the best subset. After feature engineering we came to know that some features like high fever, acute lever failure, joint pain, fatigue, mild fever, etc are highly important for best performing models which is shown in figure below. Using Recursive Feature Elimination Cross-Validation (RFECV), we perform feature selection to see the accuracy of our models. After training our models on relevant features obtained by RFECV we found the promising results which is shown in figure above.

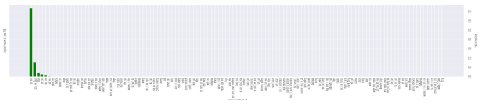

Fig. 5: Highly important feature for Linear SVM.

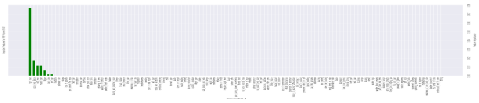

Fig. 6: Highly important feature for RBF K SVM.

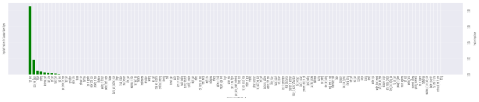

Fig. 7: Highly important feature for Gaussian Naive Bayes.

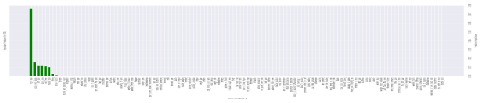

Fig. 8: Highly important feature for KNN.

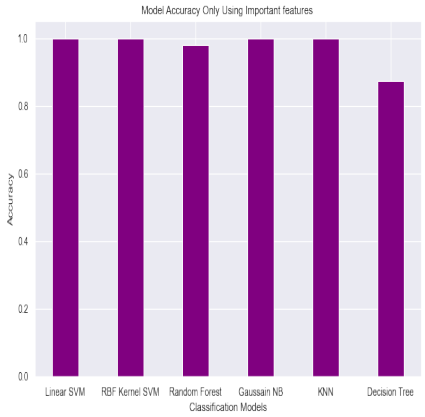

Fig. 9: Accuracy of models after performing RFECV.

XI. Results

So, we combine best performing models based on Feature Engineering and Recursive Feature Elimination Cross-Validation (RFECV). The model is then re-built and again the least important variables are removed. This process is repeated until the optimal subset of features is obtained. The optimal subset is then used to train the final model. We can see that all of the data points were correctly categorised by our combined model. The last step in the implementation process will involve building a function that accepts a list of symptoms, as input and outputs the disease predicted by the combined model based on the input symptoms and result is shown in figure below. Here is one sample which illustrate the output predicted by our combined model.

XII. Conclusion

A real-world problem's data was used to gradually implement machine learning. The path that was followed was as follows:

1) Exploratory Data Analysis
   - Deal with data issues

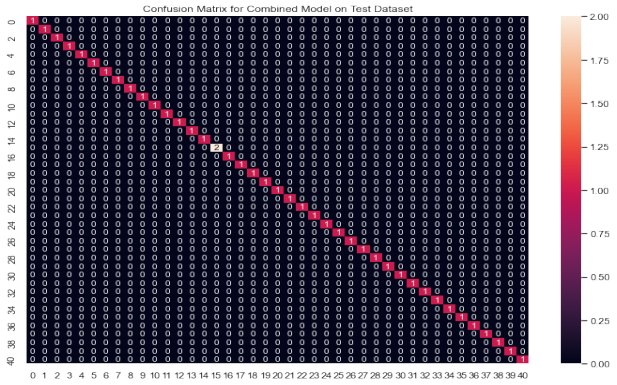

Fig. 10: Confusion matrix showing accuracy for Combined Model.

Testing by User

```
In [323]: user = predictDisease("Itching,Skin Rash,Nodal Skin Eruptions")

In [324]: user

Out[324]: {'lsvm_model_prediction': 'Fungal infection',
           'rbf_k_prediction': 'Fungal infection',
           'rf_model_prediction': 'Fungal infection',
           'gnb_model_prediction': 'Fungal infection',
           'nn_model_prediction': 'Fungal infection',
           'final_prediction': 'Fungal infection'}
```

Fig. 11: Illustrative result

   - Fill in missing values
   - Check variance and correlation in features
   - Check for class balance
2) Feature Engineering and Feature Selection
3) Building Model
4) Model Optimization using RFECV
5) Implementing best combined model

We have solved the problem of identifying disease based on the symptoms.

XIII. References

[1] Christopher M. Bishop "Pattern Recognition and Machine Learning".
[2] https://www.geeksforgeeks.org/disease-prediction-using-machine-learning.
[3] https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm.
[4]https://www.youtube.com/watch?v=1dlb4cdGtVIlist=PLZKhx5nBXhfhl