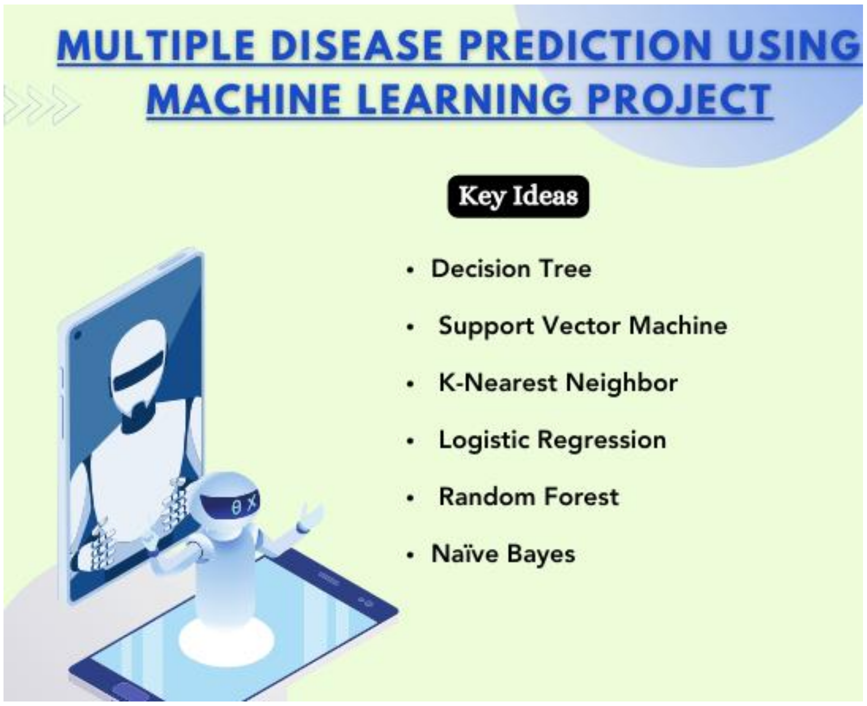


Disease Prediction

USING MACHINE LEARNING

Abhay Sachan | cc24mtech11004@iith.ac.in | 02/11/2024

Disease Prediction Using Machine Learning

- This model aims to implement a robust machine-learning model that can efficiently predict the disease of a human, based on the symptoms that he/she possesses.

- **Approach:**

- **Gathering the Data:** Data preparation is the primary step for any machine learning problem. I will be using a dataset from Kaggle for this problem. This dataset consists of two CSV files one for training and one for testing. There is a total of 133 columns in the dataset out of which 132 columns represent the symptoms, and the last column is the prognosis.

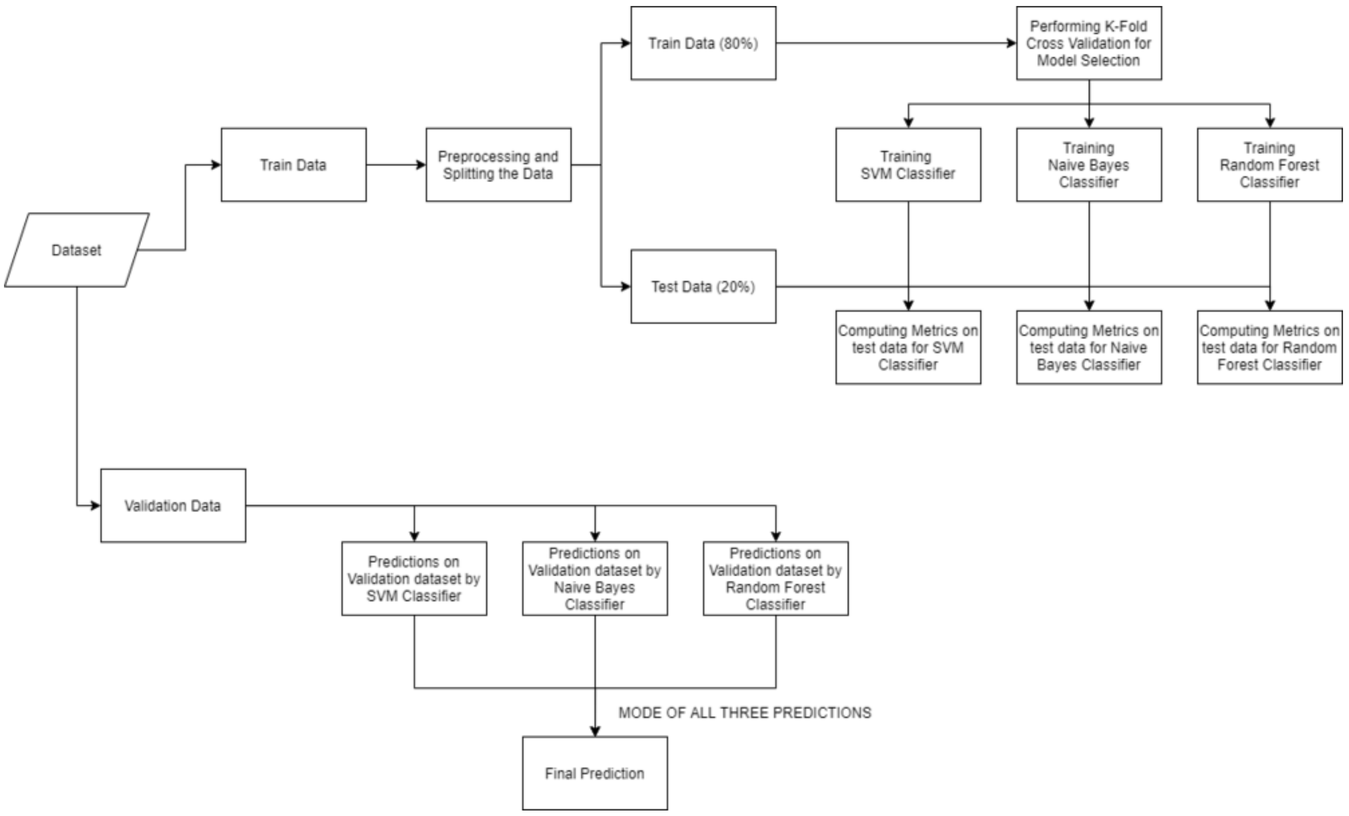
- **Cleaning the Data:** The quality of our data determines the quality of our machine-learning model. So, it is always necessary to clean the data before feeding it to the model for training. In our dataset all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a label encoder.

- **Model Building:** After gathering and cleaning the data, the data is ready and can be used to train a machine learning model. I will be using this cleaned data to train the Support Vector Classifier, Naive Bayes Classifier, and Random Forest Classifier. I will be using a confusion matrix to determine the quality of the models.

- **Inference:** After training the three models we will be predicting the disease for the input symptoms by combining the predictions of all three models. This makes our overall prediction more robust and accurate.

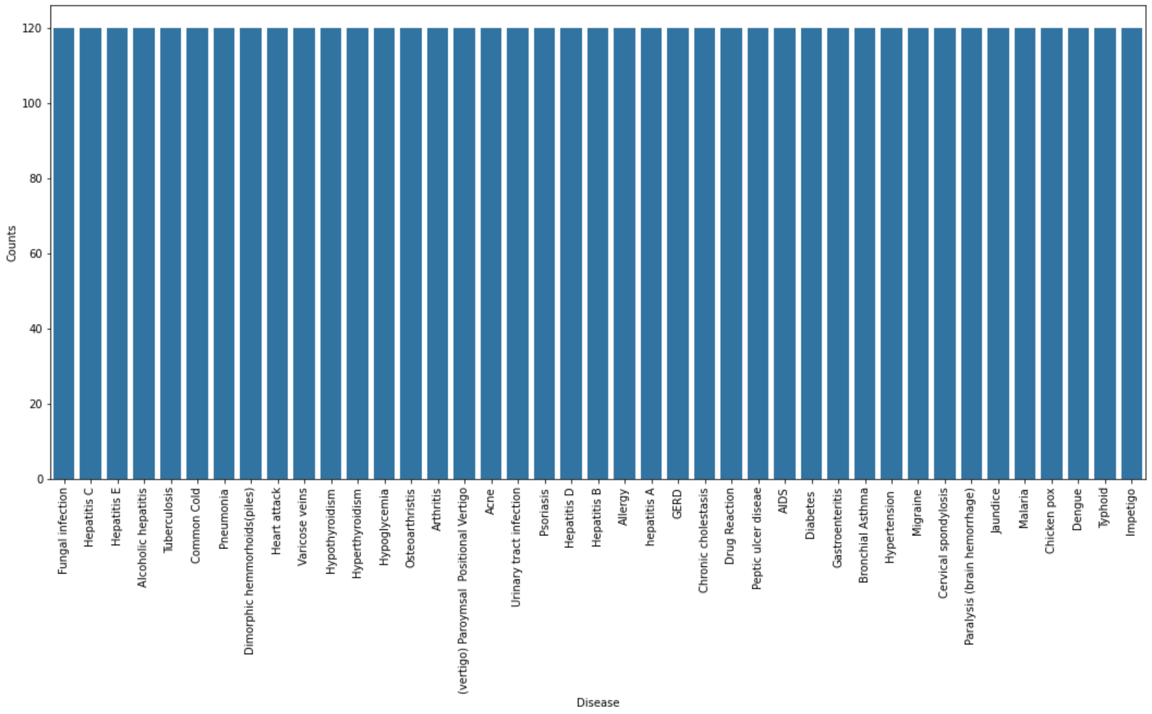
- At last, we will be defining a function that takes symptoms separated by commas as input, predicts the disease based on the symptoms by using the trained models, and returns the predictions in a JSON format.

IMPLEMENTATION


Dataset
Train Data
Preprocessing and Splitting the Data
Train Data (80%)
Test Data (20%)
Performing K-Fold Cross Validation for Model Selection
Training SVM Classifier
Training Naive Bayes Classifier
Training Random Forest Classifier
Computing Metrics on test data for SVM Classifier
Computing Metrics on test data for Naive Bayes Classifier
Computing Metrics on test data for Random Forest Classifier
Validation Data
Predictions on Validation dataset by SVM Classifier
Predictions on Validation dataset by Naive Bayes Classifier
Predictions on Validation dataset by Random Forest Classifier
MODE OF ALL THREE PREDICTIONS
Final Prediction

READING THE DATASET

Firstly, we will be loading the dataset from the folders using the pandas library. While reading the dataset we will be dropping the null column. This dataset is a clean dataset with no null values and all the features consist of 0's and 1's. Whenever we are solving a classification task it is necessary to check whether our target column is balanced or not. We will be using a bar plot, to check whether the dataset is balanced or not.



From the above plot, we can observe that the dataset is a balanced dataset i.e. there are exactly 120 samples for each disease, and no further balancing is required. We can notice that our target column i.e. prognosis column is of object datatype, this format is not suitable to train a machine learning model. So, we will be using a label encoder to convert the prognosis column to the numerical datatype. Label Encoder converts the labels into numerical form by assigning a unique index to the labels. If the total number of labels is n, then the numbers assigned to each label will be between 0 to n-1.

SPLITTING THE DATA FOR TRAINING AND TESTING THE MODEL

Now that we have cleaned our data by removing the Null values and converting the labels to numerical format. We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models.

Train: (3936, 132), (3936,)

Test: (984, 132), (984,)

MODEL BUILDING

After splitting the data, we will be now working on the modelling part. We will be using K-Fold cross-validation to evaluate the machine-learning models. We will be using Support Vector Classifier, Gaussian Naive Bayes Classifier, and Random Forest Classifier for cross-validation. Before moving into the implementation part let us get familiar with k-fold cross-validation and the machine learning models.

- **K-Fold Cross-Validation:** K-Fold cross-validation is one of the cross-validation techniques in which the whole dataset is split into k number of subsets, also known as folds, then training of the model is performed on the k-1 subsets and the remaining one subset is used to evaluate the model performance.
- **Support Vector Classifier:** Support Vector Classifier is a discriminative classifier i.e. when given a labelled training data, the algorithm tries to find an optimal hyperplane that accurately separates the samples into different categories in hyperspace.
- **Gaussian Naive Bayes Classifier:** It is a probabilistic machine learning algorithm that internally uses Bayes Theorem to classify the data points.
- **Random Forest Classifier:** Random Forest is an ensemble learning-based supervised machine learning classification algorithm that internally uses multiple decision trees to make the classification. In a random forest classifier, all the internal decision trees are weak learners, and the outputs of these weak decision trees are combined i.e. mode of all the predictions is as the final prediction.

Using K-Fold Cross-Validation for model selection

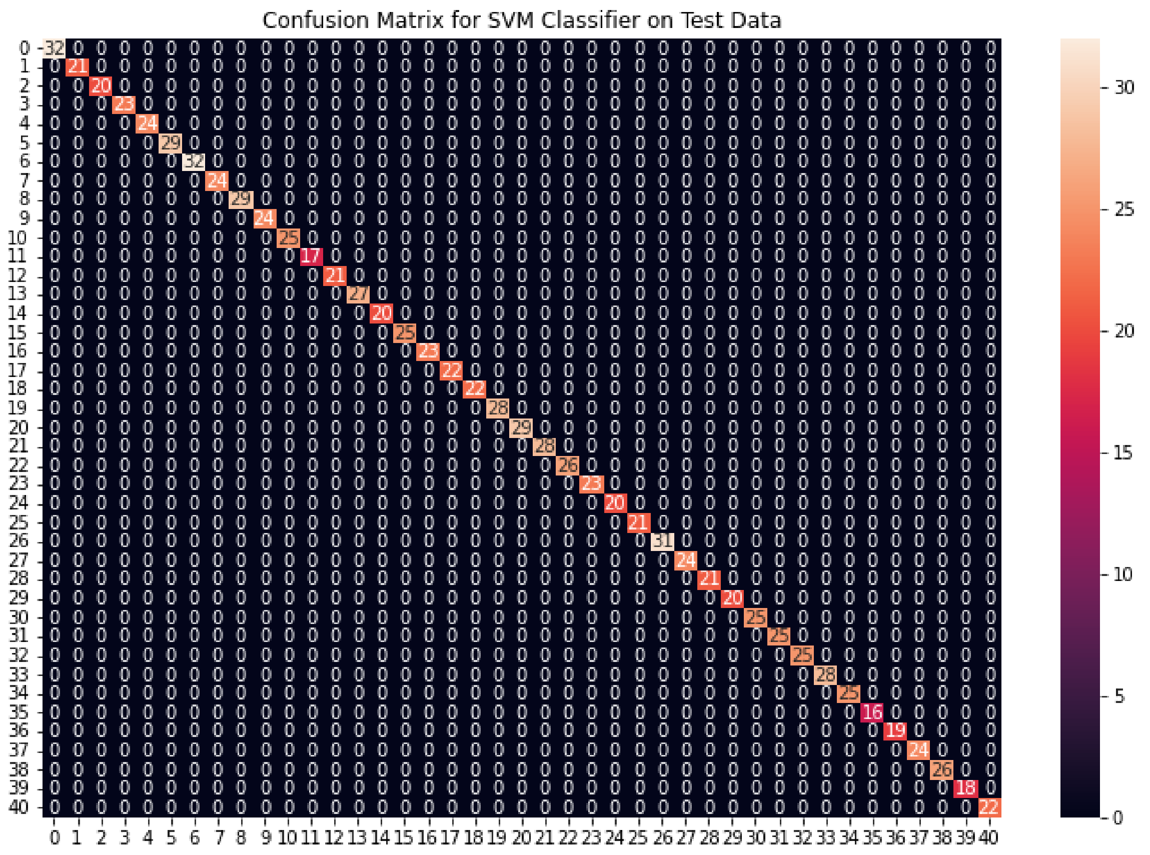
```
============================================================
SVC
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Gaussian NB
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Random Forest
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
```

From the above output, we can notice that all our machine learning algorithms are performing very well and the mean scores after k fold cross-validation are also very high. To build a robust model we can combine i.e. take the mode of the predictions of all three models so that even one of the models makes wrong predictions and the other two make correct predictions then the final output would be the correct one. This approach will help us to keep the predictions much more accurate on completely unseen data. We will be training all the three models on the train data, checking the quality of our models using a confusion matrix, and then combining the predictions of all three models.

BUILDING ROBUST CLASSIFIER BY COMBINING ALL MODELS:

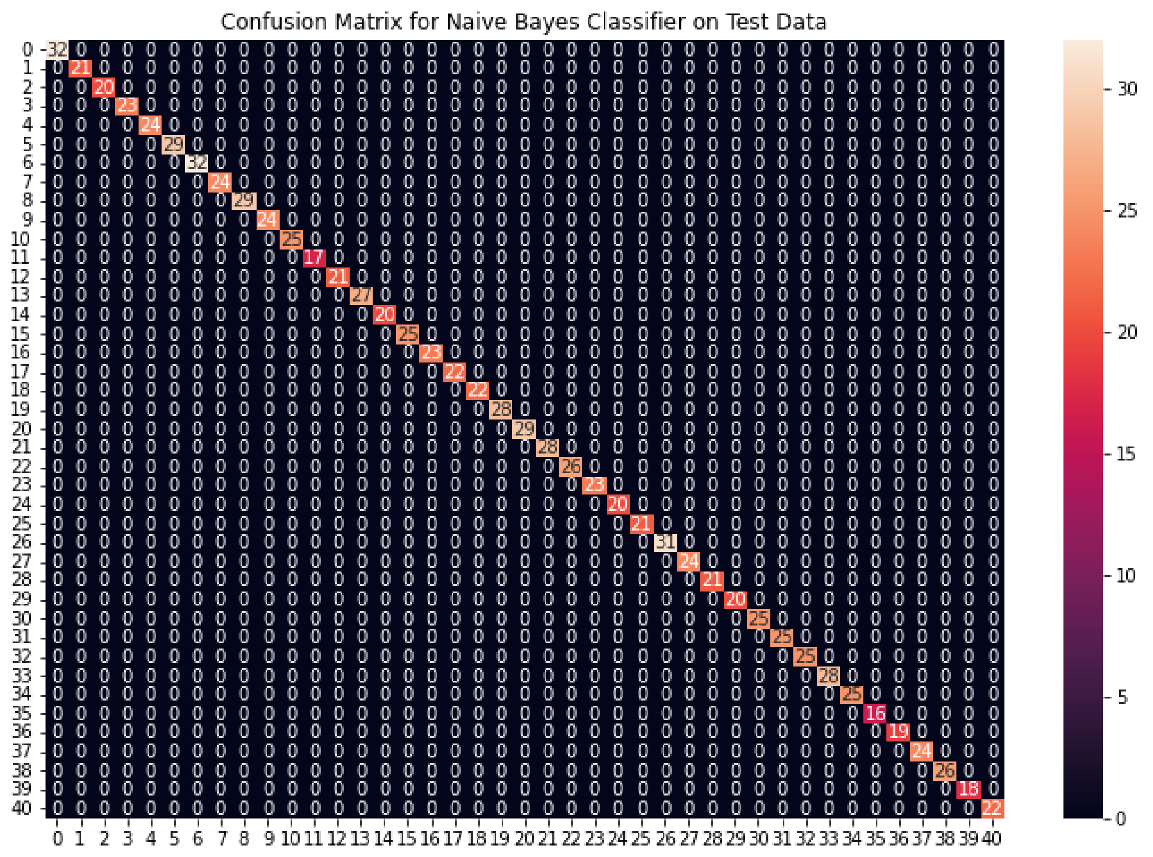
Accuracy on train data by SVM Classifier: 100.0

Accuracy on test data by SVM Classifier: 100.0



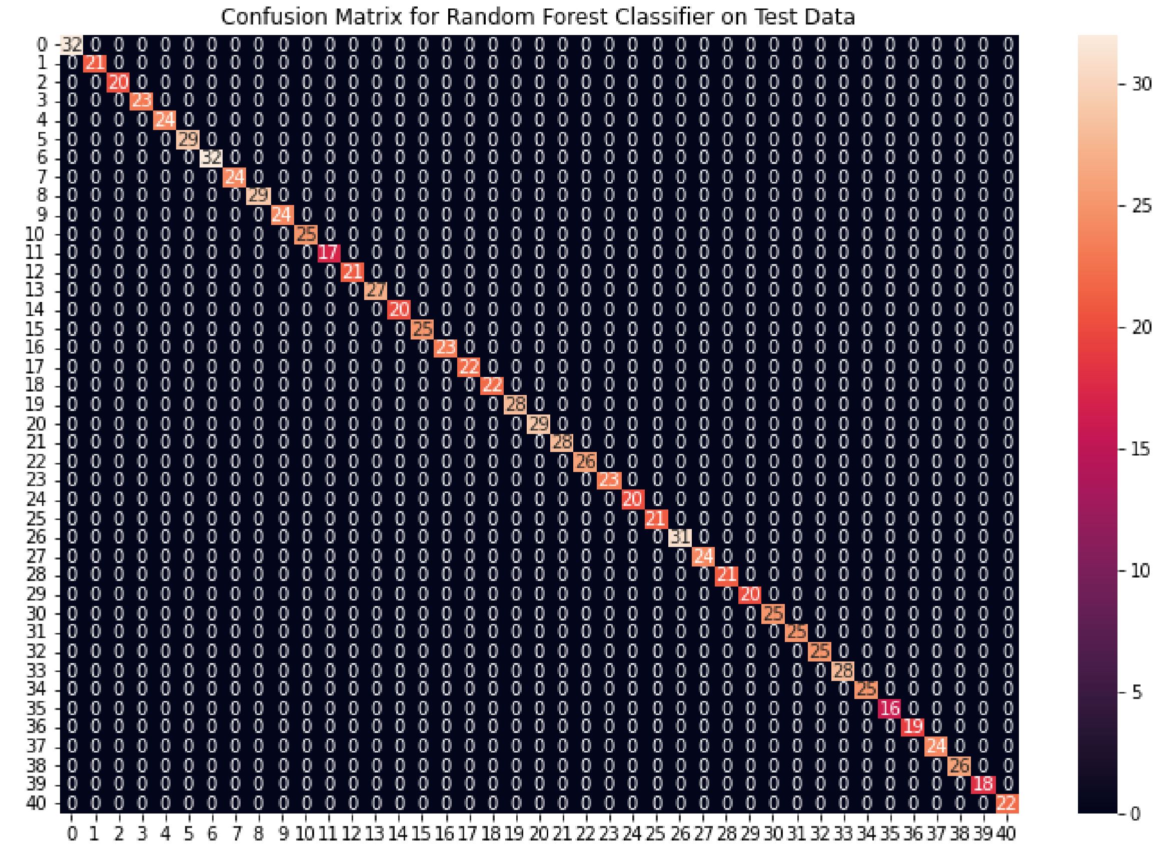
Accuracy on train data by Naive Bayes Classifier: 100.0

Accuracy on test data by Naive Bayes Classifier: 100.0



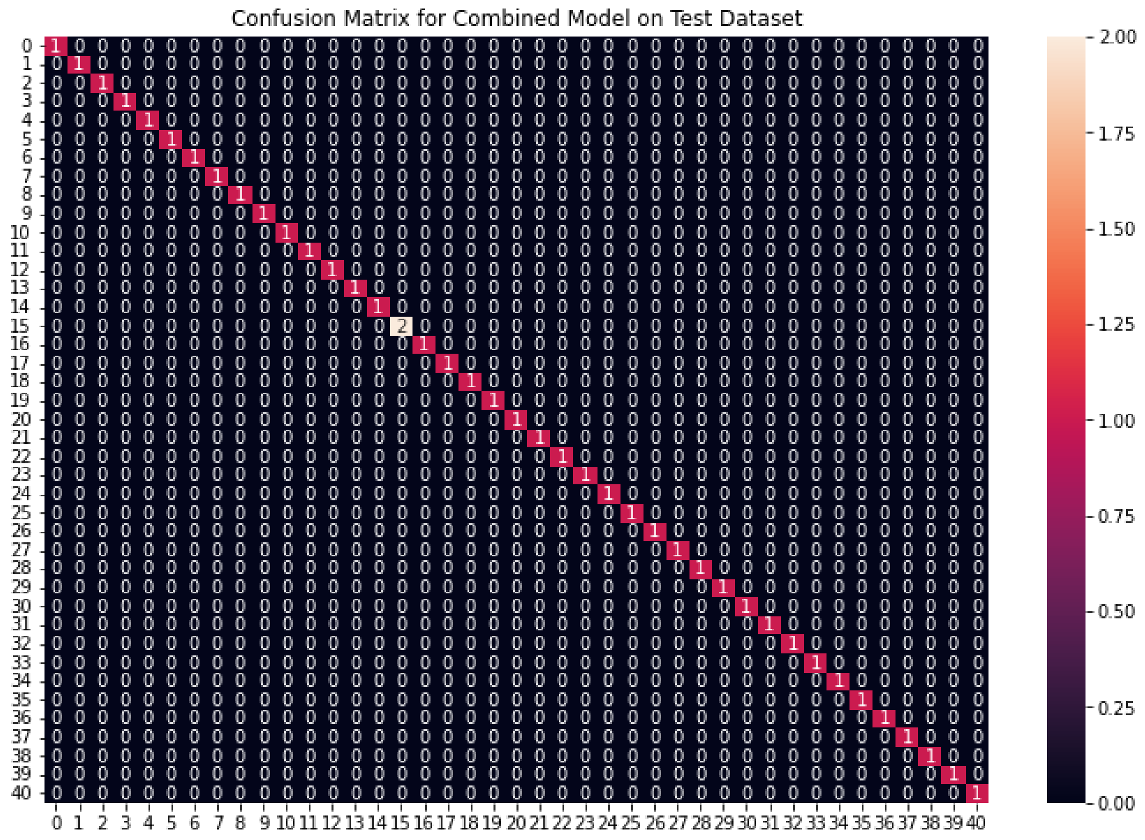
Accuracy on train data by Random Forest Classifier: 100.0

Accuracy on test data by Random Forest Classifier: 100.0



Fitting the model on whole data and validating on the Test dataset:

Accuracy on Test dataset by the combined model: 100.0



Creating a function that can take symptoms as input and generate predictions for disease

Note: *The symptoms that are given as input to the function should be exactly the same among the 132 symptoms in the dataset.*

Output:

```
'rf_model_prediction': 'Fungal infection',

'naive_bayes_prediction': 'Fungal infection',

'svm_model_prediction': 'Fungal infection',

'final_prediction': 'Fungal infection'
```