DISEASE-PREDICTION USING MACHINE LEARNING

GROUP MEMBERS:

Komal Kesav Nenavath (TL)

Aryan Chandrakant Bande

Peram Lokesh

Harsh Vishesh

Adarsh Verma

Prakhar Gangwar

For Naïve Bayes:

The code reads in the training and testing datasets as pandas DataFrames, and then uses scikit-learn's SimpleImputer to impute any missing values with the mean of the feature. The code then applies OneHotEncoder to encode categorical features as binary features. The encoded training and testing data are then concatenated, and the training data is used to train a Gaussian Naive Bayes model.

The trained model is then used to predict the class labels of the test data, and the accuracy, precision, recall, and F1 score are computed using scikit-learn's metrics functions. These metrics provide an evaluation of the model's performance on the test set.

For Random Forest:

The code reads in the training and testing datasets as pandas DataFrames. It then extracts the input features (stored in columns 0 to 131) and class labels (stored in column 132) from both the training and testing data using iloc indexing.

The Random Forest classifier is trained with 100 trees using the RandomForestClassifier class from scikit-learn. The fit method of the RandomForestClassifier object is called with the training data to train the model.

The trained model is then used to predict the class labels of the test data using the predict method of the RandomForestClassifier object. The accuracy, precision, recall, and F1 score of the model's predictions are then computed using scikit-learn's metrics functions.

The confusion matrix, which shows the number of true positives, false positives, true negatives, and false negatives, is also printed to evaluate the performance of the model on each class.

For Support vector machine:

This code builds a Support Vector Machine (SVM) model using linear kernel and evaluates its accuracy using various metrics on a dataset. Here's a brief description of what each step does:

Import required libraries: This code imports the necessary libraries such as numpy, pandas, sklearn, and metrics to work with datasets, SVM model and evaluate its performance.

Read the dataset: This code reads the training and testing datasets using pandas' read_csv() function and assigns them to data_train and data_test respectively.

Prepare the data: This code extracts the input features (X) and output target (y) from the datasets and assigns them to x_test, y_test, x_train, and y_train respectively.

Create SVM model: This code creates an SVM model using sklearn's svm.SVC() function with the 'linear' kernel.

Train SVM model: This code trains the SVM model on the training data using the fit() method of the SVM object.

Make predictions on test data: This code uses the predict() method of the SVM object to make predictions on the test data.

Evaluate model accuracy: This code evaluates the performance of the SVM model using metrics such as confusion matrix, accuracy, precision, recall, and F1 score.

Result:

According to these predictions we decided to select Naïve Bayes and SVM since both of them were getting the highest accuracy.