The Disease Prediction System

**Introduction**:

Disease prediction system using machine learning (ML) is a system that can efficiently predict the disease of a human, based on the symptoms that he/she possesses. It uses various ML algorithms to analyze large datasets of medical records and identify key features for each disease. It can also use deep learning techniques to process complex data such as images, text, or speech. The main goal of such a system is to provide accurate and timely diagnosis, as well as to suggest possible treatments and preventive measures.

**Process:**

The code is a Python program that uses Tkinter to create a graphical user interface (GUI) for a disease prediction system. The program uses three machine learning algorithms: Decision Tree, Random Forest, and Naive Bayes, to predict the outcome of diabetes based on the symptoms entered by the user. The program also displays the accuracy of each algorithm on a test set of data. Here is a brief explanation of each part of the code:

- The first part of the code imports the necessary modules and libraries, such as Tkinter, sklearn, pandas, and numpy.

- The second part of the code loads the dataset from a CSV file called 'diabetes.csv'. The dataset contains 768 rows and 9 columns, where the first 8 columns are the features (such as glucose level, blood pressure, etc.) and the last column is the outcome (0 for negative and 1 for positive). The code also creates a list of the unique values in the outcome column, which are 0 and 1.

- The third part of the code defines some variables that will store the user input from the GUI. The variables are Name, Symptom1, Symptom2, Symptom3, Symptom4, and Symptom5, and they are all of type StringVar, which is a Tkinter class that can hold strings and update the GUI when the value changes.

- The fourth part of the code defines three functions: DecisionTree, randomforest, and NaiveBayes. Each function performs the following steps:

- Split the data into features (X) and target (y) by dropping the outcome column from the data.

- Split the data into train and test sets using the train\_test\_split function from sklearn, with a test size of 0.2 and a random state of 42. This means that 20% of the data will be used for testing and the rest for training, and the random state ensures that the same split is done every time the code is run.

- Create a classifier object from the corresponding algorithm. For example, DecisionTree uses the DecisionTreeClassifier class from sklearn.tree, randomforest uses the RandomForestClassifier class from sklearn.ensemble, and NaiveBayes uses the GaussianNB class from sklearn.naive\_bayes.

- Train the classifier on the train set using the fit method.

- Make predictions on the test set using the predict method.

- Calculate the accuracy of the predictions using the accuracy\_score function from sklearn.metrics, which compares the predicted values with the actual values in the test set.

- Display the accuracy on the GUI by deleting the previous content of a text widget (t1, t2, or t3) and inserting the new accuracy value using the delete and insert methods. The text widgets are defined later in the code.