Report for heart disease prediction machine learning model.

By: Ganna Magdy

Importing necessary libraries:

First, we used a code to import essential libraries for data manipulation, visualization, and machine learning, including pandas, numpy, matplotlib, seaborn, and various modules from scikit-learn. Additionally, we imported specific classifiers and tools from scikit-learn, xgboost, and lightgbm for building and evaluating machine learning models for heart disease prediction.

Data features:

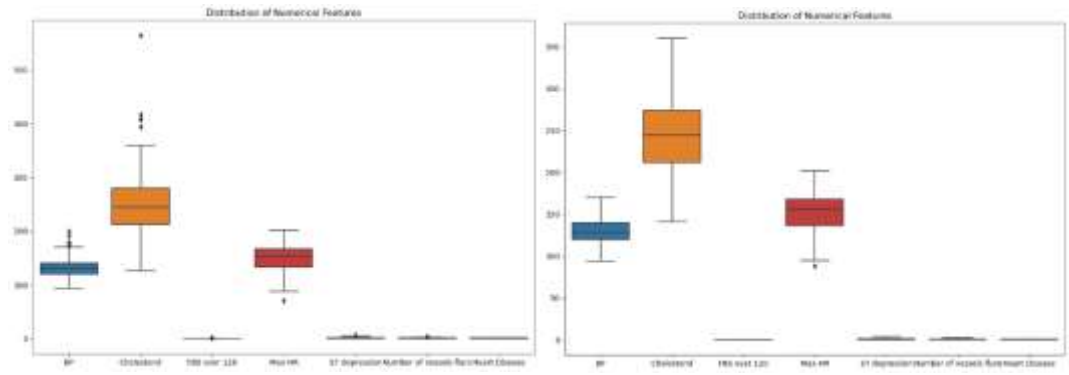
We then used df = pd.read_csv('Heart_Disease_Prediction.csv') to load the data and visualized its features using df and checked for any null values and description for that data type using print(df.isnull().sum()) and df.info() respectively.

Feature engineering:

After checking the description of data, we found categorical data in Heart disease column. We transferred it to numerical using dummies into 'Heart Disease_Absence', 'Heart Disease_Presence and then later grouped them again into Heart_disease as 0 and 1.

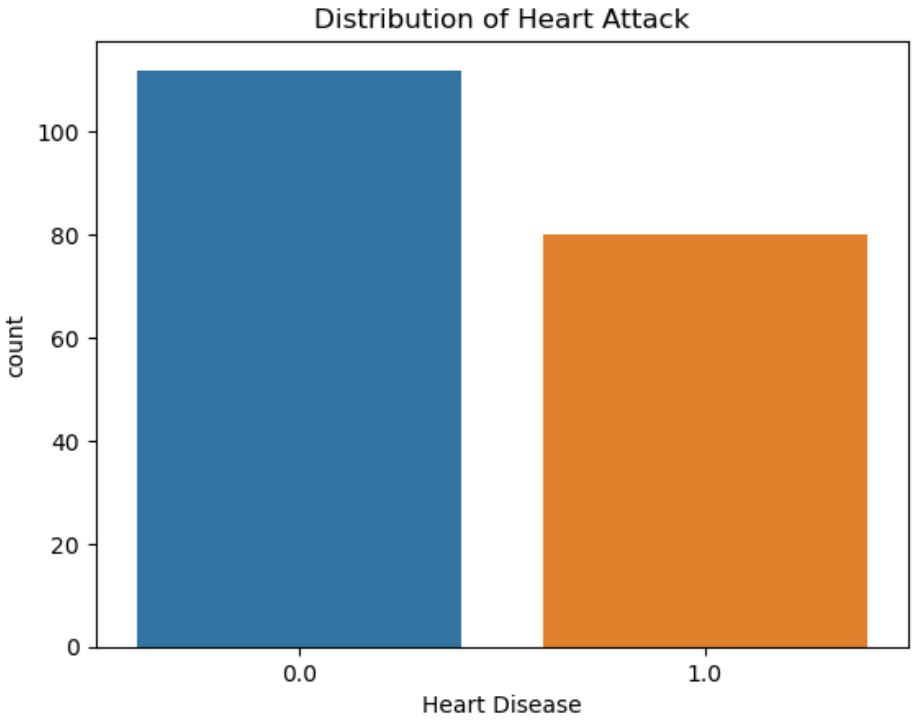
Cleaning and preprocessing:

We checked the distribution of the data using box plot and checked for outliers and iden tified some we later used quantiles method to remove the outliers then visualized the bo x plot again to make sure most of the outliers were in fact removed.

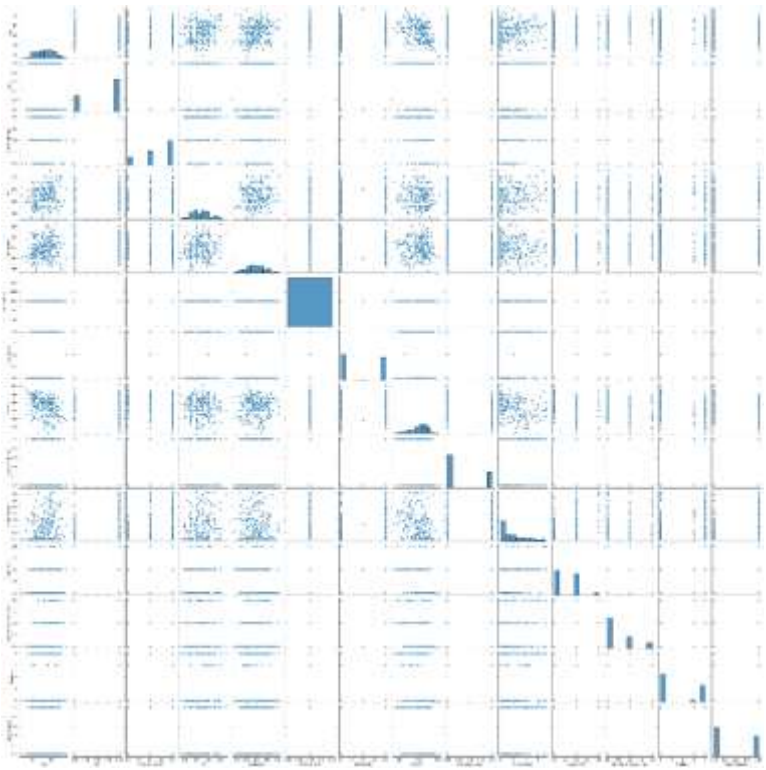


Further data visualization:

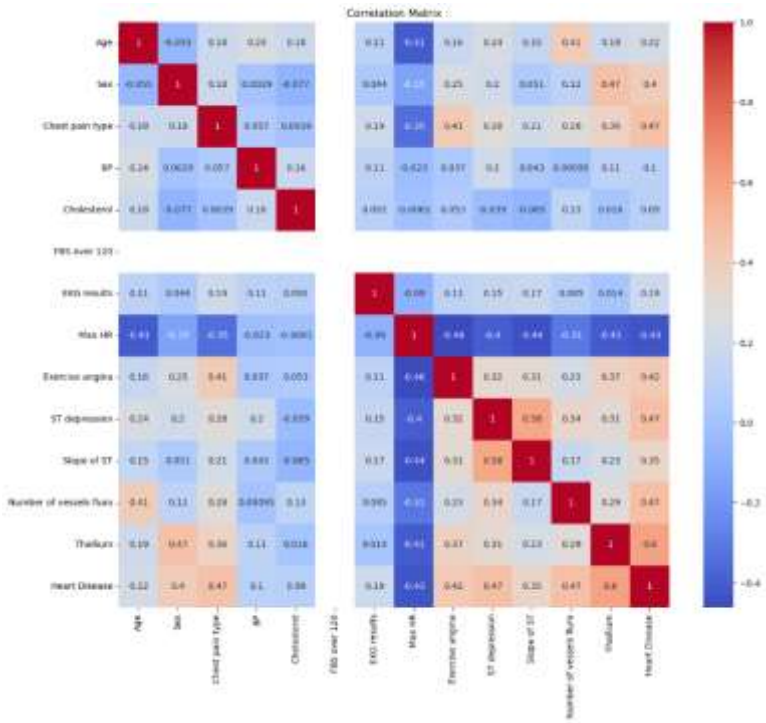
We used count plot to check the diseased vs the normal patients.



We used pair plot which visualizes pairwise relationships between variables in the Data Frame.



Finally, we used heatmap function from Seaborn which visualizes the correlation matrix as a heatmap, where each cell's colour represents the strength and direction of the correlation. The annot=True parameter adds the correlation values to the plot.



Machine Learning Model Development:

Data Preprocessing and Splitting:

- Separates the features (X) and target variable (y).
- Splits the dataset into training and testing sets using train_test_split.

Data Imputation and Scaling:

- Uses SimpleImputer to handle missing values by imputing the mean for each feature.
- Scales the features using StandardScaler to standardize them.

Model Training and Evaluation:

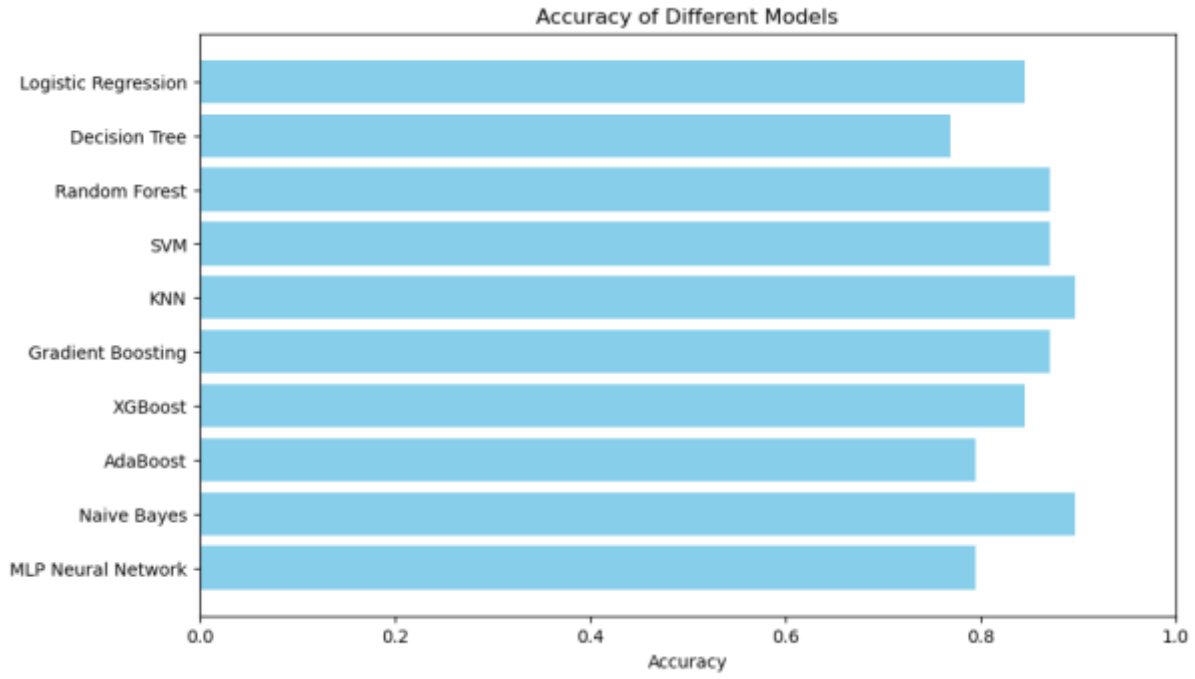
Initializes various classifiers and stores them in a dictionary named models and Iterates through each model.

Models used:

'Logistic Regression', 'Decision Tree', 'Random Forest', 'SVM', 'KNN', 'Gradient Boosting', 'XGBoost', 'AdaBoost' , 'Naive Bayes' , 'MLP Neural Network'

Model Evaluation:

- After training, the model makes predictions (y_pred) on the testing data (X_test).
- Calculates the accuracy of the model using accuracy_score.
- Prints the accuracy score for the model.
- Generates and prints a classification report using classification_report, which includes precision, recall, F1-score, and support for each class.
- Calculates the confusion matrix (cm) using confusion_matrix.
- Prints the confusion matrix for the model, providing a detailed breakdown of true positives, false positives, true negatives, and false negatives.
- We then used a code to write the accuracy of each model and deploy them into a graph.



Discovery:

We found out most of the models had high accuracy except for decision tree. We decided to rely on logistic regression for the model deployment as we believed other models with higher accuracy seemed too overfit.

Model deployment:

we utilized Streamlit, a web application framework, to create an interactive web application for heart disease prediction.

- We Defined a filename for the saved model ('Heart_disease_prediction.sav').
- Uses pickle.dump() to serialize and save the model to a file in binary format ('wb' mode).

We then transferred to visual studio to work on the model deployment. The Streamlit application allows users to input various features related to heart health and predicts the likelihood of heart disease based on those inputs using a pre-trained logistic regression model.

Model Loading:

The pre-trained logistic regression model is loaded using **pickle.load()** from the saved file **"Heart_disease_prediction.sav"**.

Feature selection:

Users input their data for heart disease prediction through the text input fields provided for features like age, sex, blood pressure, cholesterol levels, etc.

Prediction:

Upon clicking the "confirm" button, the application triggers the model to make predictions based on the input data.The result of the prediction is displayed on the sidebar. If the prediction is 0, it indicates no risk of heart disease, and an image of a heart character with a thumbs-up is displayed. If the prediction is 1, indicating a risk of heart disease, an image of a heart character with a sad expression is displayed.

Output Display:

The result of the prediction, along with the corresponding image, is displayed on the sidebar. Users can interact with the application by inputting their data and confirming, receiving an instant prediction and visualization indicating the likelihood of heart disease based on their input.