heart disease prediction report

Salma Deiaa Elgezawy

Bioinformatics major

Machine learning and data mining

**Introduction:**

In this project, we focus on leveraging machine learning algorithms to analyze a dataset pertaining to various attributes and risk factors associated with heart disease. Our goal is to develop a predictive model that can accurately classify individuals as either having or not having heart disease based on their relevant health indicators.

**Dataset:**

The dataset contains information on various attributes and risk factors related to heart disease and it was retrieved from Kaggle website. Each row represents an individual and includes the following features:

1. Age: The age of the individual
2. Sex: Gender of the individual (0 = female, 1 = male)
3. Chest pain type: Type of chest pain experienced by the individual (1 to 4)
4. Resting blood pressure: Resting blood pressure in mm Hg
5. Serum cholesterol: Serum cholesterol level in mg/dl
6. Fasting blood sugar: Fasting blood sugar level > 120 mg/dl (1 = true, 0 = false)
7. Resting electrocardiographic results: Results of resting electrocardiogram (0 to 2)
8. Max heart rate: Maximum heart rate achieved.
9. Exercise induced angina: Exercise-induced angina (1 = yes, 0 = no)
10. Old peak: ST depression induced by exercise relative to rest.
11. ST segment: ST segment measurement (1 to 3)
12. Major vessels: Number of major vessels colored by fluoroscopy (0 to 3)
13. Thal: Thalassemia blood disorder type (3 = normal; 6 = fixed defect; 7 = reversible defect)
14. Heart disease: Target variable indicating the presence of heart disease (0 = no, 1 = yes, 2 = yes)

**Data Info:**

1. The dataset was examined for missing values using the **isnull().sum()** function, and the result showed that there are no missing values in any of the columns. This indicates that the dataset is complete and ready for further analysis and modeling without the need for any data imputation or handling of missing values.
2. The **df.info()** function was used to provide an overview of the dataset's structure and data types. The output reveals that the dataset consists of 270 entries and 14 columns. The data types of the features are either integer or float, with 'old peak' being the only feature represented as a float64 data type, while the rest are int64. This information is crucial for understanding the nature of the data and selecting appropriate data preprocessing and modeling techniques.
3. The **df.describe()** function was used to generate descriptive statistics of the dataset's numerical columns. The output provides valuable insights into the central tendency, dispersion, and shape of the distribution of each feature.

**Exploratory data analysis (EDA):**

1. The code **df['age'].min(), df['age'].max()** was used to determine the minimum and maximum values present in the 'age' column of the dataset. The output (29, 77) reveals that the age range of the individuals in the dataset spans from 29 to 77 years. This information is important as it provides insights into the age distribution of the study population.
2. The code **df['age'].describe()**was utilized to generate descriptive statistics for the 'age' column in the dataset. The output provides key summary statistics that offer insights into the distribution of ages among the individuals in the dataset.
3. A histogram was created using Seaborn (sns) to visualize the distribution of the 'age' column in the dataset. The histogram includes a kernel density estimate (kde) to show the underlying probability density function. Additionally, vertical lines were added to the histogram to represent the mean, median, and mode of the 'age' column. The mean is indicated by a red line, the median by a green line, and the mode by a blue line.
4. The gender distribution in the dataset was examined by calculating the percentage of males and females. These percentages provide insights into the gender composition of the dataset, indicating that males make up a significant majority compared to females.
5. Then I had to identify columns with missing values in the dataset. The output indicates that all columns are complete, with no missing values present. Each column is shown with its corresponding percentage of missing values, which, in this case, is 0% for all columns. This suggests that the dataset is clean and does not require imputation or handling of missing data before further analysis can be performed.
6. A bar chart is then made to show the count of heart attacks for two categories labeled as '1' and '2' in heart disease feature. The bar corresponding to category '1' is taller, indicating a higher count of heart attacks, approximately 140. The bar corresponding to category '2' has a lower count, which is approximately 100. This distribution suggests that category '1' experienced more heart attacks than category '2'.
7. A pair plot is also made to show the relationships between all variables in the dataset. We have diagonal histograms, and off diagonal scatter plots to show whether there is positive, negative, or no correlation between the variables.
8. Then a correlation matrix is done. The correlation matrix heatmap provides a visual representation of the relationships between various cardiovascular-related variables. The heatmap displays correlations ranging from -1 to 1, with colors representing the strength and direction of the correlations. Blue indicates negative correlations, red indicates positive correlations, and white represents little to no linear correlation.

**Dealing with outliers:**

* 1. A boxplot analysis was conducted to visualize the distribution of various numerical features in the dataset. These insights into central tendency, variability, and outlier presence in each feature are crucial for initial data exploration and potential preprocessing steps before further analysis.
  2. Outliers were handled using the Interquartile Range (IQR) method. The first quartile (Q1) and third quartile (Q3) were calculated, and the IQR was determined as the difference between Q3 and Q1. Data points outside the range of (Q1 - 1.5 \* IQR) to (Q3 + 1.5 \* IQR) were considered outliers and removed from the dataset. The cleaned dataset, **df\_cleaned**,was generated by excluding rows containing outliers. A summary of the cleaned dataset's descriptive statistics is presented through the print **(df\_cleaned. describe ())** command. This process aims to ensure a more robust and reliable analysis by mitigating the impact of extreme values on statistical measures.

**Model building and Evaluating:**

1. We split the data into features (**X**) and the target variable (**y**). The features (**X**) are all the columns in the dataset except for the 'heart disease' column, which is our target variable (**y**). This separation allows us to train our machine learning model on the features to predict the target variable.
2. The dataset is divided into two subsets: training and testing sets. The training set (denoted as **X\_train** and **y\_train**) is used to train the machine learning model, allowing it to learn patterns and relationships within the data. The testing set (**X\_test** and **y\_test**) is kept separate and used to evaluate the model's performance and generalization ability on unseen data. The **test\_size** parameter, set to 0.2, specifies that 20% of the data will be allocated to the testing set, while the remaining 80% will be used for training. The **random\_state** parameter ensures reproducibility by fixing the random seed used for shuffling the data before splitting.
3. Missing values in the dataset, if any, are addressed using the **SimpleImputer** from the scikit-learn library. The strategy parameter is set to 'mean', indicating that missing values will be replaced with the mean value of the respective feature calculated from the non-missing values in the training set. This imputation strategy helps to maintain the integrity of the dataset by ensuring that missing values do not adversely affect the model training process.
4. Feature standardization is performed using the **StandardScaler** from the scikit-learn library. Standardization is a preprocessing technique that rescales the features to have a mean of 0 and a standard deviation of 1, making the features more comparable and ensuring that they have similar scales.
5. The target variable is transformed into a binary format where class 1 is mapped to 1 and class 2 is mapped to 0. This transformation is carried out to suit the binary classification nature of the problem, simplifying the evaluation and interpretation of the model's performance. After the transformation, a suite of classification algorithms is trained using the transformed target variable and the preprocessed feature data. Each algorithm is evaluated in terms of **accuracy** and other classification metrics, such as **precision**, **recall**, and **F1-score**, on the testing set to assess its performance in distinguishing between the two classes.
6. Multiple machine learning models have been trained and evaluated for their performance in classifying the dataset this comprehensive evaluation helps in understanding how each model performs in distinguishing between the two classes.
7. The trained machine learning models are evaluated solely based on their accuracy scores. Each model's accuracy in correctly classifying the test data is reported, providing a simplified overview of their performance.
8. After collecting the accuracy scores, a horizontal bar plot is generated using Matplotlib to visually compare the accuracies of different models. Each bar represents the accuracy of a specific model. We then can indicate that the model with highest accuracy is Naive Bayes of 0.90 accuracy.
9. Then the data is trained again but only with the model of highest accuracy which is Naive Bayes.

**Model deployment:**

1. The **pickle.dump()** function is utilized to serialize the trained Naive Bayes model object and write it to a binary file named 'dataset\_heart.sav'. Serializing the model in this manner allows for easy storage and later retrieval. The saved model file can be loaded and used for making predictions on new data without the need to retrain the model each time.
2. Streamlit, a Python library for creating web applications, is imported as **st**.
3. The title of the application is set to 'Heart Disease Prediction app', providing a clear indication of its purpose.
4. An informational message is displayed using **st.info()**, informing users that the application is for heart disease prediction.
5. A sidebar is added to the application interface using **st.sidebar.header()**, allowing users to select features for prediction or perform other interactive actions.
6. The Streamlit application is extended to include text input fields for users to input various features related to heart disease prediction. These input fields allow users to input their personal health information or clinical test results, which will be used as features for heart disease prediction.
7. A Pandas DataFrame is constructed using the **pd.DataFrame()** function, where the data consists of user-inputted values for various features related to heart disease prediction.
8. The **pickle.load()** function is used to deserialize the trained model object from the specified file path 'D:\Education\university\6 term\machine learning\dataset\_heart.sav'. The model is loaded in binary mode (**'rb'**).
9. A button named 'confirm' is added to the sidebar using Streamlit's **st.sidebar.button()** function. This button serves as a trigger for initiating the prediction process.
10. When the 'confirm' button is clicked (Con is True), the user-inputted data stored in the DataFrame df is passed to the loaded model's **predict()** function. The predicted result is obtained from the model and displayed using Streamlit's **st.write()** function.

- The link to the explanatory video: [Meeting with Salma Deiaa Elgezawy-20240513\_231840-Meeting Recording.mp4](https://nileuniversity-my.sharepoint.com/:v:/g/personal/s_deiaa2110_nu_edu_eg/EX68_1GEeXNPj-mvxUxZrWUBsuvbp6iXN8PLpCSBdqf6YA?e=YaNJGK&nav=eyJwbGF5YmFja09wdGlvbnMiOnt9LCJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbE1vZGUiOiJtaXMiLCJyZWZlcnJhbFZpZXciOiJwb3N0cm9sbC1jb3B5bGluayIsInJlZmVycmFsUGxheWJhY2tTZXNzaW9uSWQiOiIzNDU1OWMxZi01Y2RkLTQxYWYtOThkMy02NmJkZDJkYWUwMTEifX0%3D)