

Heart Disease Prediction Model Using Machine Learning

Machine Learning

Saeedullah Sabawon
Sayedullah411@gmail.com

Table of Contents

Introduction	2
Tools and Libraries	2
1. NumPy:.....	2
2. Scikit-learn (sklearn):	2
3. Pandas:.....	2
4. Matplotlib:	2
5. Jupyter Notebook:	3
Data Collection.....	3
Data source	3
Data Exploration	4
Attribute Of the Data Sets.....	4
Attribute of cols cauterization	5
Pre-Processing of Data	6
Missing value.....	6
Python missing value filling and checking.....	6
Transforming of the data	7
Distribution of the target variable	8
Data Set Ready for Training Model.....	9
Splitting Data Set.....	10
Optimal Machine Learning Algorithm for Predicting Heart Disease	10
Logistic regression.....	10
Random forest	11
K-Nearest Neighbors (KNN)	11
Decision Tree.....	11
Support Vector Machines	11
Artificial Neural Networks.....	11
Training model	12
Model Evaluation	12
Prediction:.....	14
Conclusion.....	14
Bibliography	16

Introduction

Heart disease is one of the most critical and dangerous illnesses nowadays worldwide, with many people suffering from this disease. To address this issue, we propose to develop a machine learning model that is capable of predicting heart disease based on patient data. This model will help people easily predict their disease. To develop this model, we will use different machine learning algorithms such as logistic regression, decision trees, random forest, Artificial Neural Networks, Support Vector Machines, Decision Tree classifier, K-Nearest Neighbors, but I select the logistic regression because we want to just predict data whether the patient has the heart disease there are two classes one is the disease and no disease and for the binary classification the logistic regression is the best selection base on decision and we will also use the python and related libraries to train the model effectively. The model will provide a more efficient, faster, and less expensive solution for diagnosing heart disease.

Tools and Libraries

We will use some tools for this project such as Jupyter notebook and python library for this below is the short description of each one

1. NumPy:

NumPy is a powerful Python library for scientific computing and numerical operations. It provides efficient multidimensional array objects, along with various mathematical functions to operate on these arrays. With NumPy, you can perform array manipulation, element-wise operations, linear algebra operations, and mathematical computations efficiently. It is widely used as the foundation for many other libraries in the data science ecosystem.

2. Scikit-learn (sklearn):

Scikit-learn is a popular machine learning library in Python. It provides a wide range of algorithms for various tasks such as classification, regression, clustering, and dimensionality reduction. Sklearn is built on top of NumPy and SciPy, making it easy to integrate with other data science libraries. It also offers consistent APIs, making it simple to use and experiment with different models. Sklearn includes functions for data preprocessing, model selection, evaluation metrics, and feature extraction, making it a comprehensive library for machine learning tasks.

3. Pandas:

Pandas is a versatile data manipulation and analysis library in Python. It offers robust data structures such as Data Frame and Series, which allow efficient handling of structured data. Pandas provides functions for data cleaning, transformation, merging, and analysis. It enables easy indexing, filtering, and grouping of data, making it convenient to perform operations on datasets. Pandas is commonly used for data preprocessing, exploratory data analysis, and preparing data for machine learning tasks.

4. Matplotlib:

Matplotlib is a powerful visualization library in Python. It provides a wide variety of customizable plotting functions and styles, making it suitable for creating publication-quality charts, graphs, and visualizations. With Matplotlib, you can generate line plots, scatter plots, bar plots, histograms, and more. It also offers fine-grained control over plot elements and supports visualization of multiple subplots. Matplotlib is widely used in data analysis, presentations, and creating visualizations to gain insights from data.

5. Jupyter Notebook:

Jupyter Notebook is an interactive computing environment that allows you to create and share documents containing code, visualizations, and explanatory text. It provides a web-based interface where you can write and execute code in cells, making it an excellent tool for data exploration, analysis, and collaborative work. Jupyter Notebook supports various programming languages, including Python, R, and Julia. It allows you to combine code execution with rich text elements, such as Markdown, to create dynamic and interactive narratives. Jupyter Notebook is widely used in data analysis, research, and educational settings, providing an accessible and flexible environment for data scientists and analysts.

These libraries, NumPy, sklearn, pandas, and Matplotlib, are essential tools in the machine learning system in Python. They provide the necessary functionality for data manipulation, analysis, machine learning modeling, and visualization, contributing significantly to the effectiveness and efficiency of data-driven projects.

Data Collection

Data collection is typically the starting point of the process. Datasets can take various forms, including structured and unstructured data, which may contain missing or noisy information. Each data type and format requires specific methods for data handling and management. In addition to this, it's important to mention that the project will also involve the identification and implementation of data cleaning techniques to ensure the accuracy and reliability of the collected data.

Data source

As part of our project, we aimed to collect data from multiple sources for accurate model training. However, due to time constraints, we obtained the dataset from a single online source for use in our machine learning class project. The dataset is publicly available on the Kaggle website and originates from an ongoing study on heart health in Framingham, Massachusetts. The study's primary goal is to predict a patient's 10-year risk of developing coronary heart disease. It includes information on over 4,000 patients and encompasses 15 different attributes, covering demographic, behavioral, and medical factors. We utilized this dataset to train our models and analyze heart health patterns within the scope of our project. If you're interested in accessing the dataset for your own research or projects, you can find the download link below.

URL: <https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/download?datasetVersionNumber=2>

Data Exploration

In this part where we looked at the data, we checked out the information we got from the Kaggle website about heart health in Framingham, Massachusetts. This information had details about more than 1025 patients and included things like age, behavior, and health info.

We took a close look at how the different details were spread out, if anything was missing, and if there were any really unusual numbers. We also made some graphs to see how the different details were connected to each other and if we could find any patterns.

We also did some basic math to understand the main things about the patients and what could predict if they might have heart problems in the next 10 years. This part helped us get a good idea about the information we had and how we could use it to build and test our models for the project.

Attribute Of the Data Sets

1. Age: displays the age of the individual.
2. Sex: displays the gender of the individual using the following format: 1 = male 0 = female.
3. Chest-pain type : displays the type of chest-pain experienced by the individual using the following format : 1 = typical angina 2 = atypical angina 3 = non - anginal pain 4 = asymptotic
4. Resting Blood Pressure : displays the resting blood pressure value of an individual in mmHg (unit)
5. Serum Cholestrol : displays the serum cholestrol in mg/dl (unit)
6. Fasting Blood Sugar : compares the fasting blood sugar value of an individual with 120mg/dl. If fasting blood sugar > 120mg/dl then : 1 (true) else : 0 (false)
7. Resting ECG : 0 = normal 1 = having ST-T wave abnormality 2 = left ventricular hypertrophy
8. Max heart rate achieved : displays the max heart rate achieved by an individual.
9. Exercise induced angina : 1 = yes 0 = no
10. ST depression induced by exercise relative to rest : displays the value which is integer or float.
11. Peak exercise ST segment : 1 = upsloping 2 = flat 3 = downsloping
12. Number of major vessels (0-3) colored by flourosopy : displays the value as integer or float.
13. Thal : displays the thalassemia : 3 = normal 6 = fixed defect 7 = reversable defect
14. Diagnosis of heart disease: Displays whether the individual is suffering from heart disease or not : 0 = absence 1,2,3,4 = present.

Attribute of cols cauterization

```
In [8]: df  
#it has the 1025 rows x 14 columns
```

```
Out[8]:
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	Male	0	125	212	0	1	168	No	1.0	2	2	3	No
1	53	Male	0	140	203	1	0	155	Yes	3.1	0	0	3	No
2	70	Male	0	145	174	0	1	125	Yes	2.6	0	0	3	No
3	61	Male	0	148	203	0	1	161	No	0.0	2	1	3	No
4	62	Female	0	138	294	1	1	106	No	1.9	1	3	2	No
...
1020	59	Male	1	140	221	0	1	164	Yes	0.0	2	0	2	Yes
1021	60	Male	0	125	258	0	0	141	Yes	2.8	1	1	3	No
1022	47	Male	0	110	275	0	0	118	Yes	1.0	1	1	2	No
1023	50	Female	0	110	254	0	0	159	No	0.0	2	0	2	Yes
1024	54	Male	0	120	188	0	1	113	No	1.4	1	1	3	No

1025 rows × 14 columns

```
In [9]: df.head()
```

```
Out[9]:
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	Male	0	125	212	0	1	168	No	1.0	2	2	3	No
1	53	Male	0	140	203	1	0	155	Yes	3.1	0	0	3	No
2	70	Male	0	145	174	0	1	125	Yes	2.6	0	0	3	No
3	61	Male	0	148	203	0	1	161	No	0.0	2	1	3	No
4	62	Female	0	138	294	1	1	106	No	1.9	1	3	2	No

```
In [10]: print(f"Total row: {len(df)}")
```

Total row: 1025

```
In [11]: print(f"Total row: {len(df.columns)}")
```

Total row: 14

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   age        1025 non-null   int64  
 1   sex        1025 non-null   object 
 2   cp         1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol       1025 non-null   int64  
 5   fbs        1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   object 
 9   oldpeak   1025 non-null   float64
 10  slope      1025 non-null   int64  
 11  ca         1025 non-null   int64  
 12  thal       1025 non-null   int64  
 13  target     1025 non-null   object 
dtypes: float64(1), int64(10), object(3)
memory usage: 112.2+ KB
```

Pre-Processing of Data

In the part where we prepared the data for our project, we made sure that the information we had was ready to be used by our models. This meant we had to fix any missing or wrong data, and make sure the numbers were all on the same scale.

We also changed some of the information to make it easier for our models to understand, like turning words into numbers. We also made some new information that could help our models do a better job of predicting if someone might have heart problems in the future.

This part was really important because it helped us make sure that the data we used was good enough for our models to learn from and make accurate predictions.

Missing value

Handling missing values is important because it helps us avoid making mistakes in our analysis. If we ignore missing values, it can lead to wrong conclusions and unreliable models. By properly dealing with missing values, we can ensure that our analysis is accurate and our decisions are based on reliable information.

Below is the reason for the missing value filling:

- **Accurate analysis:** Missing values can lead to biased or inaccurate analysis, as they can skew the results of statistical tests or machine learning models.
- **Data quality:** Missing values can affect the overall quality of the dataset, making it less reliable for making decisions or drawing conclusions.
- **Model performance:** Missing values can cause issues when building predictive models, as many machine learning algorithms cannot handle missing data.
- **Interpretability:** Missing values can affect the interpretability of the results, making it difficult to understand the true nature of the relationships within the data.
- **Ethical and legal considerations:** In some cases, handling missing values is necessary to comply with ethical or legal standards, especially when dealing with sensitive or regulated data.

Python missing value filling and checking

After importing the dataset, it is important to check for missing values. This ensures the data's completeness and reliability. In Python, various methods can be used to identify and handle missing values in the dataset. Below are a few of these methods.

1. We use the `isnull()` method to check the missing value for it below is the example :

```
# Check for missing values in the DataFrame
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
age          0
sex          0
cp           0
trestbps    0
chol         0
fbs          0
restecg     0
thalach     0
exang        0
oldpeak     0
slope        0
ca           0
thal         0
target       0
dtype: int64
```

In the above snapshot, we used the isnull () method to find any missing values in each column. We can see that there are no missing values in any of the columns. If there were any missing values, the count of missing values would be displayed in the second column of the table. If the count is zero, it means there are no missing values in that column.

Transforming of the data

In the May dataset, we have categorical and text data that is important for model training. Therefore, we need to find and replace these with suitable numerical values. This is necessary because most models for prediction work only with numerical values. Upon checking each column's values, I observed that the "sex" and "exang" columns should be filled with numerical values. For example, "sex" has "Male" and "Female" values, and "exang" has "Yes" and "No" values. We can replace "Yes" with 1 and "No" with 0, and "Male" with 1 and "Female" with 0. Additionally, the target column's values "Yes" and "No" can be replaced with 1 and 0, respectively.

Below snapshot for more details

```
# Replacing the sex cols value Male = 1 and Female = 0  
df['sex'] = df['sex'].replace({'Male': 1, 'Female': 0})
```

```
df.head()
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	No	1.0	2	2	3	No
1	53	1	0	140	203	1	0	155	Yes	3.1	0	0	3	No
2	70	1	0	145	174	0	1	125	Yes	2.6	0	0	3	No
3	61	1	0	148	203	0	1	161	No	0.0	2	1	3	No
4	62	0	0	138	294	1	1	106	No	1.9	1	3	2	No

```
#Replace the exang or exercise cols value Yes = 1 and No = 0  
df['exang'] = df['exang'].replace({'Yes': 1, 'No': 0})
```

```
df.head()
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	No
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	No
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	No
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	No
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	No

```
# Let's convert the target column to numerical format. For example, if the column has values 'Yes' and 'No',  
# we can represent 'Yes' as 1 and 'No' as 0.  
df['target'] = df['target'].replace({'Yes':1 , 'No': 0})
```

After replacing the value we have the data set like this

```
In [365]: df.tail()
```

```
Out[365]:
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

Distribution of the target variable

The distribution of the target variable refers to the frequency of each value in the target column. This is important because it can affect the model's accuracy and performance. In this section, we will examine the distribution of the target variable to ensure that the dataset is balanced and scaled properly. This will help to avoid bias in the model and ensure that the predictions are accurate.

Below is the snapshot for it:

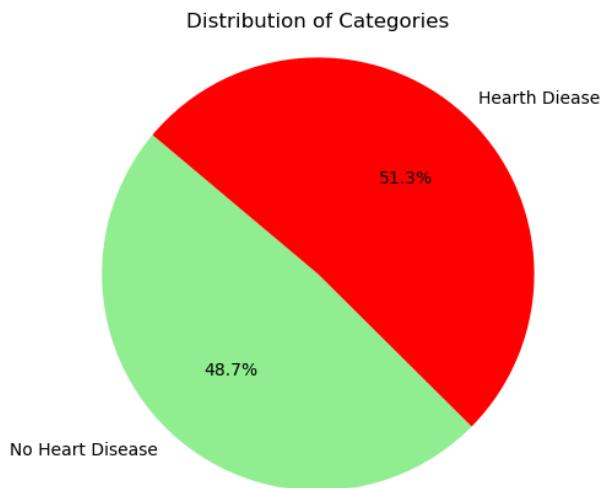
```

labels = ['No Heart Disease', 'Hearth Disease']
sizes = [num_patients_withno_heart_disease,num_patients_with_heart_disease] # percentages or counts for each category

# Create a pie chart
plt.figure(figsize=(5, 5))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=['lightgreen', 'red'])
plt.title('Distribution of Categories')
plt.axis('equal') # Equal aspect ratio ensures that the pie is drawn as a circle.

# Display the pie chart
plt.show()

```



As we see the data set is balance and scaled

Data Set Ready for Training Model

After performing several steps, the dataset has been prepared for training model. This involved cleaning the data, handling missing values, and transforming categorical variables into numerical values. Additionally, outliers were identified and addressed to ensure the integrity of the dataset. The prepared dataset is now ready for further exploration and modeling. The steps taken have contributed to enhancing the quality and reliability of the dataset for accurate analysis and predictions.

Splitting Data Set

Splitting data into training and testing datasets is important in machine learning and data analysis. It involves dividing the available data into two separate sets - one for training our models and another for evaluating their performance. The training dataset is used to build and optimize our models, while the testing dataset is used to assess how well these models will generalize to new, unseen data. Splitting the data helps us get an unbiased estimate of how well our models will perform in the real world and prevents overfitting, where the model memorizes the training data but fails to generalize to new data.

In order to train and evaluate our models, we split the dataset into two parts: the training set and the testing set. Typically, we allocate 80% of the data to the training set and reserve the remaining 20% for testing purposes. The training set allows us to optimize and adjust our models based on the available data. Then, we employ the testing set to assess how well our trained models perform on unseen data, as a way to estimate their real-world predictive capabilities.

Below is the snapshot of code which used for the division

```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.20 , stratify=Y , random_state=2)
#in the above X is the independent Variable
# and Y is the dependent variable

print("Total Data set row :" + str(len(X)))
print("Training Data set :" + str(len(X_train)))
print("Testing Data set :" + str(len(X_test)))

Total Data set row :1025
Training Data set :820
Testing Data set :205
```

Optimal Machine Learning Algorithm for Predicting Heart Disease

When it comes to predicting heart disease, selecting the most optimal machine learning algorithm is crucial. There are several algorithms that can be utilized for this task, each with its own strengths, weaknesses, and suitability for specific types of data. One popular algorithm for heart disease prediction is Logistic Regression. This algorithm is well-suited for binary classification problems, which makes it effective for determining whether or not an individual is likely to have heart disease based on various input features such as age, cholesterol levels, blood pressure, and so on. Logistic Regression provides interpretable results and can highlight the importance of each feature in the prediction process.

Below is the list of algorithm we can use for the hearth disease predication but I select the logistic regression

[Logistic regression](#)

Logistic regression is an effective algorithm for predicting heart disease. It estimates the probability of an individual having heart disease based on input features like age, blood pressure, and cholesterol levels. It provides interpretable results with coefficients indicating the impact of each feature. By fitting

a logistic function to the data, it calculates the odds of heart disease. Preprocessing, training, and evaluation using metrics like accuracy and AUC-ROC are essential steps. Regularization and feature selection can enhance model performance. Overall, logistic regression is suitable for our heart disease prediction project.

Random forest

The random forest classifier is a useful and robust algorithm used for classification tasks with tabular data. It is widely used in areas such as fraud detection, loan risk prediction, and heart disease prediction. By combining results from multiple decision trees, it optimizes training and enhances predictive accuracy. [1]

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a classifier that creates clusters of data samples with the same target value and assigns new values to classes using a distance metric. It is robust and efficient for heart disease detection, especially with only two classes. However, the choice of distance metric can impact the classifier's speed, and for larger datasets, KNN is relatively slower than other methods. [1]

Decision Tree

Decision Tree classifier is a model that creates a tree using dataset attributes, with branches leading to leaves made up of target values. The tree identifies leading features of class labels using visual components and an information gain index. Decision trees are fast and robust for disease prediction if the dataset has powerful features for a simple use-case. [1] [2]

Support Vector Machines

Support Vector Machines (SVM) is a non-probabilistic classifier that generates hyperplanes to divide data points of two classes in vector space. For N features and M targets, SVM creates $M-1 \times N$ -dimensional hyperplanes to separate data points of different classes. SVM optimizes the margin metric to find the best hyperplane for all categories, making it popular for disease prediction as it can effectively categorize tabular data into different categories.

Artificial Neural Networks

Artificial Neural Networks (ANN) are widely used in deep learning, with a standard architecture consisting of a hidden layer made up of linear nodes. Adding more layers and nodes increases the model's ability to learn complex relationships between variables and input features. This capability makes the network effective at capturing relationships between biological and personal markers that affect the probability of heart disease.

In the field of heart disease prediction, the Python libraries Numpy, Pandas, Matplotlib, and Scikit-learn (Sklearn) play crucial roles. Numpy and Pandas are used for data manipulation and analysis, while Matplotlib is working for data visualization. Sklearn offers a wide range of machine learning algorithms and tools for model building and evaluation, making it essential for heart disease prediction using machine learning models.

Training model

After collecting, cleaning, and splitting the data, we now need to choose an algorithm and teach the model using the training data.

Logistic regression has been chosen as the algorithm for the training model. It is effective for classification tasks and can accurately predict outcomes in discrete categories. The model learns from the training data, adjusting its parameters through iterations to minimize errors. Logistic regression captures patterns and dependencies in the data, enabling it to make accurate predictions on unseen data. The algorithm can be fine-tuned and its performance evaluated using techniques like cross-validation. Once trained and evaluated, the logistic regression model can be tested on a separate dataset for reliable real-world performance.

Below is the snapshot

```
#Let create the object of the Logistic regression
model = LogisticRegression()

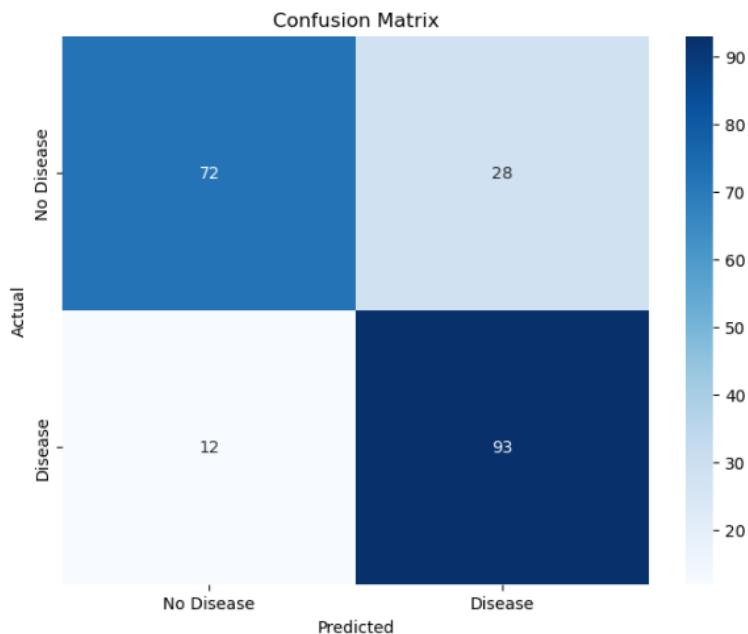
## then we will pass our training data into the Logistic regression model for traing model
model.fit(X_train , Y_train)

LogisticRegression()
```

Model Evaluation

Model evaluation is an essential step in assessing the performance and effectiveness of a logistic regression model or trained model. It helps us understand how well the model generalizes to new, unseen data and provides insights into its accuracy and reliability. There are various techniques and metrics that can be used to evaluate the logistic regression model. Some commonly methods include:

- 1. Confusion Matrix:** The confusion matrix presents a tabular representation of the model's performance by comparing its predictions against known ground truth labels. It contains four metrics: true positives (correctly predicted positive instances), true negatives (correctly predicted negative instances), false positives (incorrectly predicted positive instances), and false negatives (incorrectly predicted negative instances). Below is the snapshot it



Interpretation

- **True Negatives (TN):** 72 cases correctly predicted as "No Disease."
- **False Positives (FP):** 28 cases incorrectly predicted as "Disease" when they were actually "No Disease."
- **False Negatives (FN):** 12 cases incorrectly predicted as "No Disease" when they were actually "Disease."
- **True Positives (TP):** 93 cases correctly predicted as "Disease."

2. Accuracy: Accuracy is a simple yet widely-used measure of overall model performance. It calculates the percentage of correctly predicted instances out of the total number of instances in the evaluation dataset. However, it may not be the most reliable metric, especially when dealing with imbalanced datasets.

```
#let check the accuracy for the training and testing value
#training data
X_train_predication = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_predication , Y_train)

print("Accuracy of the Training data : " ,training_data_accuracy)
Accuracy of the Training data :  0.8524390243902439

#testing data
Y_train_predication = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_train_predication , Y_test)

print("Accuracy of the Testing data : " ,test_data_accuracy)
Accuracy of the Testing data :  0.8048780487804879

# Note: We need to ensure that the accuracy of the testing and training datasets doesn't have a significant difference.
# If the difference is too high, it could indicate overfitting.
```

Prediction:

The two inputs of the prediction component are the model and the prediction set. The prediction result shows the predicted data, actual data, and the probability of different results in each group.

Below snap shot we pass some data into the model for predication and it will predicate the base on the data the patient is the disease or not

```
input_data = (65,0,2,140,417,1,0,157,0,0.8,2,1,2)
#change the input data into numpy array for better predication
input_data_as_numpy_array = np.asarray(input_data)
#let's reshape the numpy array as we are predicting for only one instance it mean predict for the one row
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

#Please test this with Model which taken from the orginal data set
# 60,1,0,145,282,0,0,142,1,2.8,1,2,3    ---> 0  it mean no hearth disease
# 65,0,2,140,417,1,0,157,0,0.8,2,1,2  ---> 1 it mean has the hearth disease
```

```
predication = model.predict(input_data_reshaped)

if(predication == 0):
    print("The Patinet does not have Hearth Disease")
else :
    print("The Patinet has the Hearth Disease")
```

The Patinet has the Hearth Disease

Conclusion

In this project, we addressed the critical issue of heart disease by developing a logistic regression model capable of predicting the likelihood of a patient having heart disease. We compared several machine learning algorithms, but logistic regression was selected due to its suitability for binary classification tasks and interpretability.

We used essential tools and libraries such as NumPy, Pandas, Matplotlib, Scikit-learn (Sklearn), and Jupyter Notebook for data manipulation, analysis, visualization, and model building.

The dataset used in this project was downloaded from the Kaggle platform and provided valuable insights into patient demographics, behavioral factors, and medical history attributes. Data pre-processing was performed, which included handling missing values, transforming categorical data into numerical values, and ensuring the dataset was ready for model training.

The logistic regression model was trained on the prepared dataset, adjusting parameters to minimize errors and capture patterns in the data. The model's performance was evaluated using metrics such as the confusion matrix, accuracy, and prediction results. The confusion matrix provided detailed information about true positives, true negatives, false positives, and false negatives, while accuracy measured overall correctness.

The logistic regression model showed promising results in predicting heart disease. However, further improvements can be made by refining the model, exploring alternative algorithms, and incorporating additional features for enhanced accuracy. Continuous monitoring and updates to the model will ensure its relevance and effectiveness in diagnosing heart disease.

Overall, this project contributes to the development of an efficient, faster, and cost-effective solution for heart disease prediction, potentially aiding early diagnosis and intervention. Further research and improvement in this field are crucial for better healthcare outcomes and patient well-being.

Bibliography

- [1] A. D. ,. M. K. R. V.V Ramalingam, "Hearth Diesase predication using Machine Learning Techniques," *International Journal of Engineering & Technology* , vol. 7, no. 684-687, pp. 2-8, 2018.
- [2] J. Soni, U. Ansari, D. Sharma and S. Soni, "Predictive Data Mining for Medical Diagnosis: An Overview," *International Journal of Computer Applications (0975 – 8887)*, vol. 17, no. 8, p. 6, 211.
- [3] L. Yahaya, N. D. Oye and A. Adamu, "PERFORMANCE ANALYSIS OF SOME SELECTED MACHINE LEARNING ALGORITHMS ON HEART DISEASE PREDICTION USING THE NOBLE UCI DATASETS," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 1, pp. 36-46, 2020.