

Topic:

Heart Disease Analysis Using Machine Learning Algorithms

Introduction

Data Preparation and Exploration: The notebook begins by importing essential libraries for data manipulation. It loads a dataset containing 1000 patients information , equally distributed between positive and negative sentiment. Exploratory Data Analysis (EDA) is performed, visualizing the distribution of positive and negative reviewsthrough a pie chart.

Data Preprocessing and Model Building: The text data undergoes preprocessing, including cleaning, stopword removal, and word stemming. The Bag of Words model is employed to convert the text data into numerical features suitable for machine learning. Several classification models, including Decision Trees, Logistic Regression, Random Forest, k-Nearest Neighbors, are trained and evaluated for sentiment classification. Logistic Regression emerge as the top-performing model in terms of accuracy.

Model Evaluation and Sentiment Prediction: Model performance is evaluated using confusion matrices to assess their effectiveness.

Importing Required Libraries:

Data Visualization and Analysis:

- **numpy** (as **np**): Used for numerical operations.
- **pandas** (as **pd**): Used for data manipulation and analysis.
- **matplotlib.pyplot** (as **plt**): Used for creating static data visualizations.
- **seaborn** (as **sns**): Built on top of Matplotlib, used for creating more aesthetically pleasing and informative statistical graphics.
- **plotly.express** (as **px**): Used for creating interactive visualizations.
- **plotly.graph_objects** (as **go**): Used for creating interactive visualizations with Plotly.

Machine Learning and Model Evaluation:

- **sklearn.feature_extraction.text.CountVectorizer**: Used for converting text data into numerical features (bag of words).
- **sklearn.model_selection.train_test_split**: Used for splitting data into training and testing sets.
- **sklearn.linear_model.LogisticRegression**: Logistic Regression classifier.
- **sklearn.tree.DecisionTreeClassifier**: Decision Tree classifier.
- **sklearn.ensemble.RandomForestClassifier**: Random Forest classifier.

- `sklearn.neighbors.KNeighborsClassifier`: K-Nearest Neighbors classifier.
- `sklearn.metrics.confusion_matrix`: Used for calculating the confusion matrix.
- `sklearn.metrics.accuracy_score`: Used for calculating accuracy.

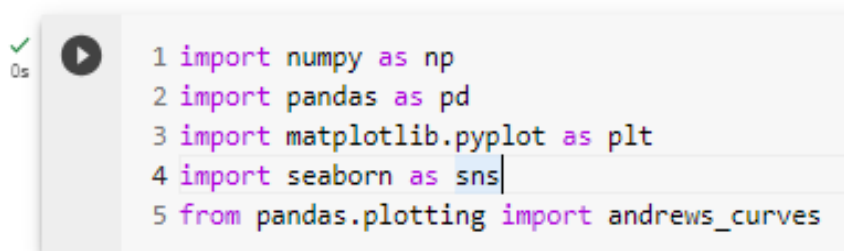
Model Tuning:

- `sklearn.model_selection.GridSearchCV`: Used for hyperparameter tuning with grid search.

Warnings Handling:

- `warnings`: Used for managing warning messages in your code.

IMPORTING LIBRARIES



```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from pandas.plotting import andrews_curves
```

WORKFLOW OF MODEL STEPS :

1. Selecting the Dataset
2. Data Preprocessing which includes loading the dataset and removing null and duplicate values
3. Feature Selection reduces irrelevant features and improve the performance of the algorithms
4. Classification and Modelling
5. Performance evaluation

Loading the Dataset:

This will load the dataset `Restaurant_Reviews.tsv` into a Pandas DataFrame called `df`. The `delimiter` parameter specifies the separator between the columns, which is tab in this case. The `quoting=3` parameter tells Pandas to remove quotes from the text in the Review column.

DATA PREPROCESSING

- Load the data using panda read function
- print dataset info
- print statistics about data using describe function
- check for null values, if any then remove it
- check for duplicate values

```
[3] 1 df = pd.read_csv("heart.csv")
```

```
[4] 1 df.head()
```

| | age | sex | cp | trestbps | chol | fbs | restecg | tl |
|---|-----|-----|----|----------|------|-----|---------|----|
| 0 | 63 | 1 | 3 | 145 | 233 | 1 | 0 | |
| 1 | 37 | 1 | 2 | 130 | 250 | 0 | 1 | |
| 2 | 41 | 0 | 1 | 130 | 204 | 0 | 0 | |
| 3 | 56 | 1 | 1 | 120 | 236 | 0 | 1 | |
| 4 | 57 | 0 | 0 | 120 | 354 | 0 | 1 | |

```
[5] 1 df.describe()
```

| | age | sex | cp | trestbps | chol | fbs |
|-------|------------|------------|------------|------------|------------|------------|
| count | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 |
| mean | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 |
| std | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 |
| min | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.000000 | 0.000000 |
| 25% | 47.500000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 |
| 50% | 55.000000 | 1.000000 | 1.000000 | 130.000000 | 240.000000 | 0.000000 |
| 75% | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 274.500000 | 0.000000 |
| max | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 |

DATA CLEANING

```
[ ] 1 print('Data Sum of Null Values \n')  
    2 df.isnull().sum()
```

Data Sum of Null Values

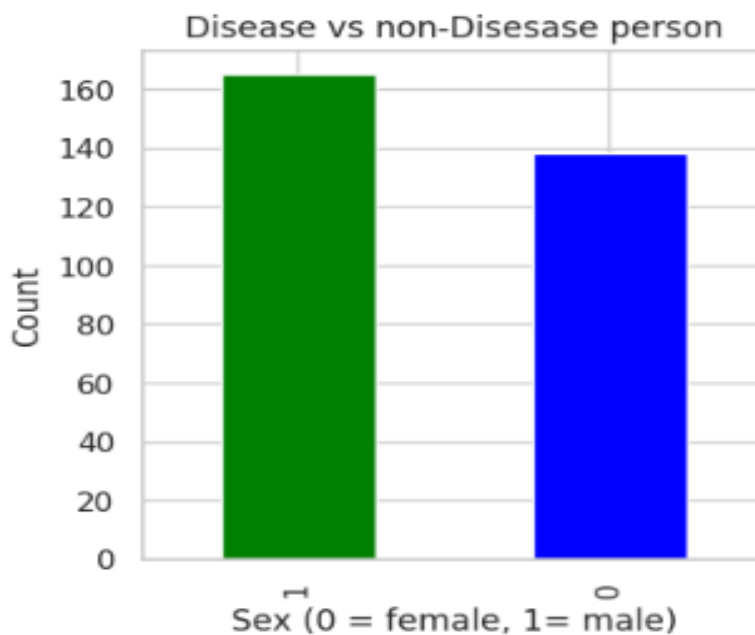
```
age      0  
sex      0  
cp       0  
trestbps 0  
chol     0  
fbs      0  
restecg  0  
thalach  0  
exang    0  
oldpeak  0  
slope    0  
ca       0  
thal     0  
target   0  
dtype: int64
```

```
[88] 1 df.drop_duplicates(subset =None, keep = 'first', inplace = True)
```

DATA ANALYSIS

```
1 plt.figure(figsize=(4, 4))
2 sns.set(style='whitegrid')
3 df.target.value_counts().plot(kind="bar", color=["green", "blue"])
4 plt.title("Disease vs non-Disease person")
5 plt.xlabel("Sex (0 = female, 1= male)")
6 plt.ylabel("Count")
```

Text(0, 0.5, 'Count')



Percentage of Patients

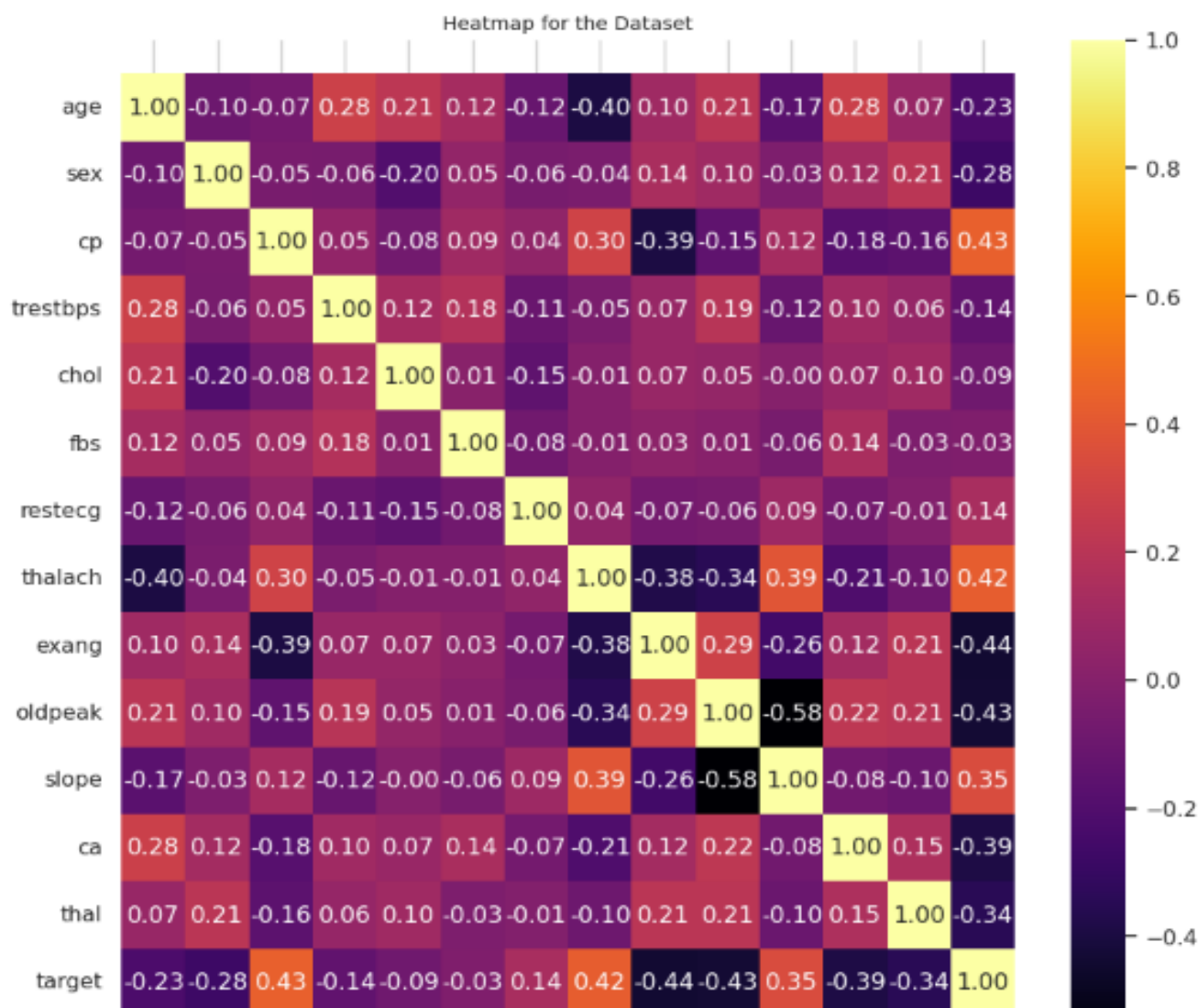
```
1 countFemale = len(df[df.sex == 0])
2 countMale = len(df[df.sex == 1])
3 print("Percentage of Female Patients: {:.2f}%".format((countFemale / (len(df.sex))*100)))
4 print("Percentage of Male Patients: {:.2f}%".format((countMale / (len(df.sex))*100)))
```

Percentage of Female Patients: 31.68%
Percentage of Male Patients: 68.32%

HEATMAP

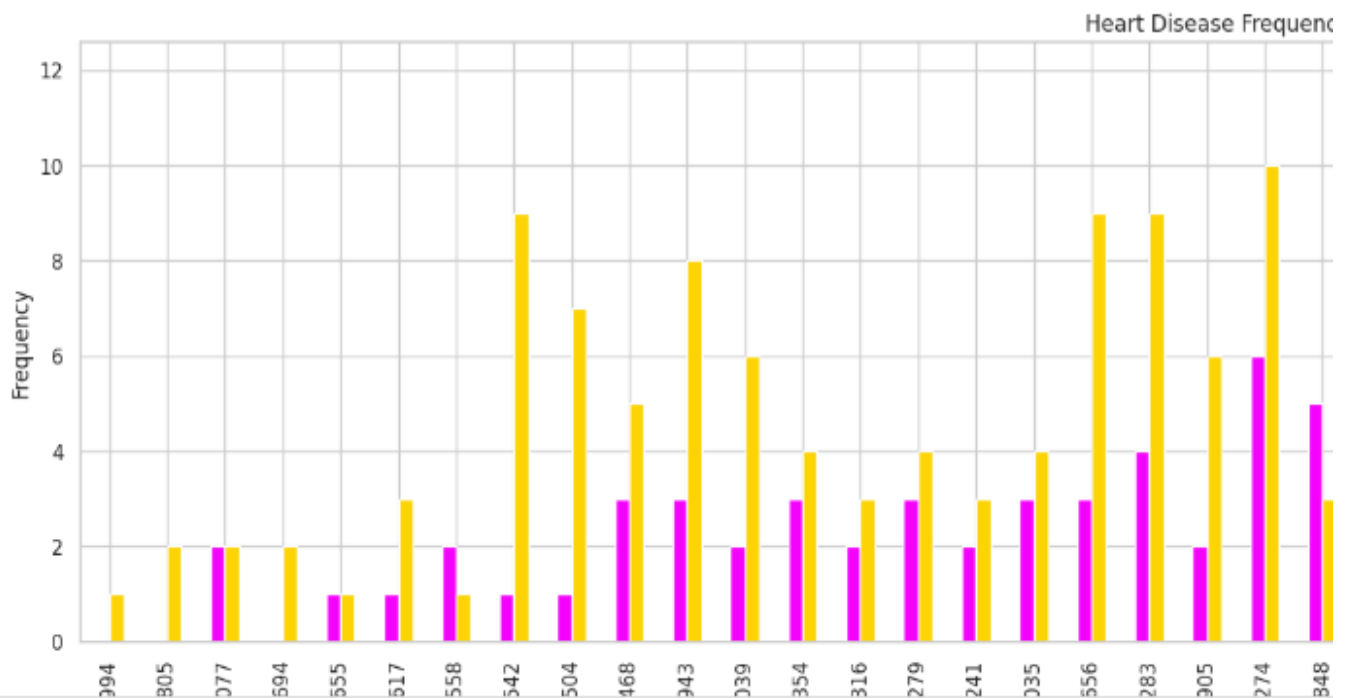
```
1 corr_matrix = df.corr()
2 fig, ax = plt.subplots(figsize=(10, 10))
3 ax = sns.heatmap(corr_matrix,
4                  annot=True,
5                  fmt=".2f",
6                  cmap="inferno");
7 plt.title('Heatmap for the Dataset', fontsize = 10)
8 bottom, top = ax.get_ylim()
9 ax.set_ylim(bottom + 0.5, top - 0.5)
```

(14.5, -0.5)



ANALYSIS USING AGE

```
1 pd.crosstab(df.age,df.target).plot(kind="bar",figsize=(24,6),color=['#f403fc', '#fcd303'])
2 plt.title('Heart Disease Frequency for Ages')
3 plt.xlabel('Age')
4 plt.ylabel('Frequency')
5 plt.show()
```

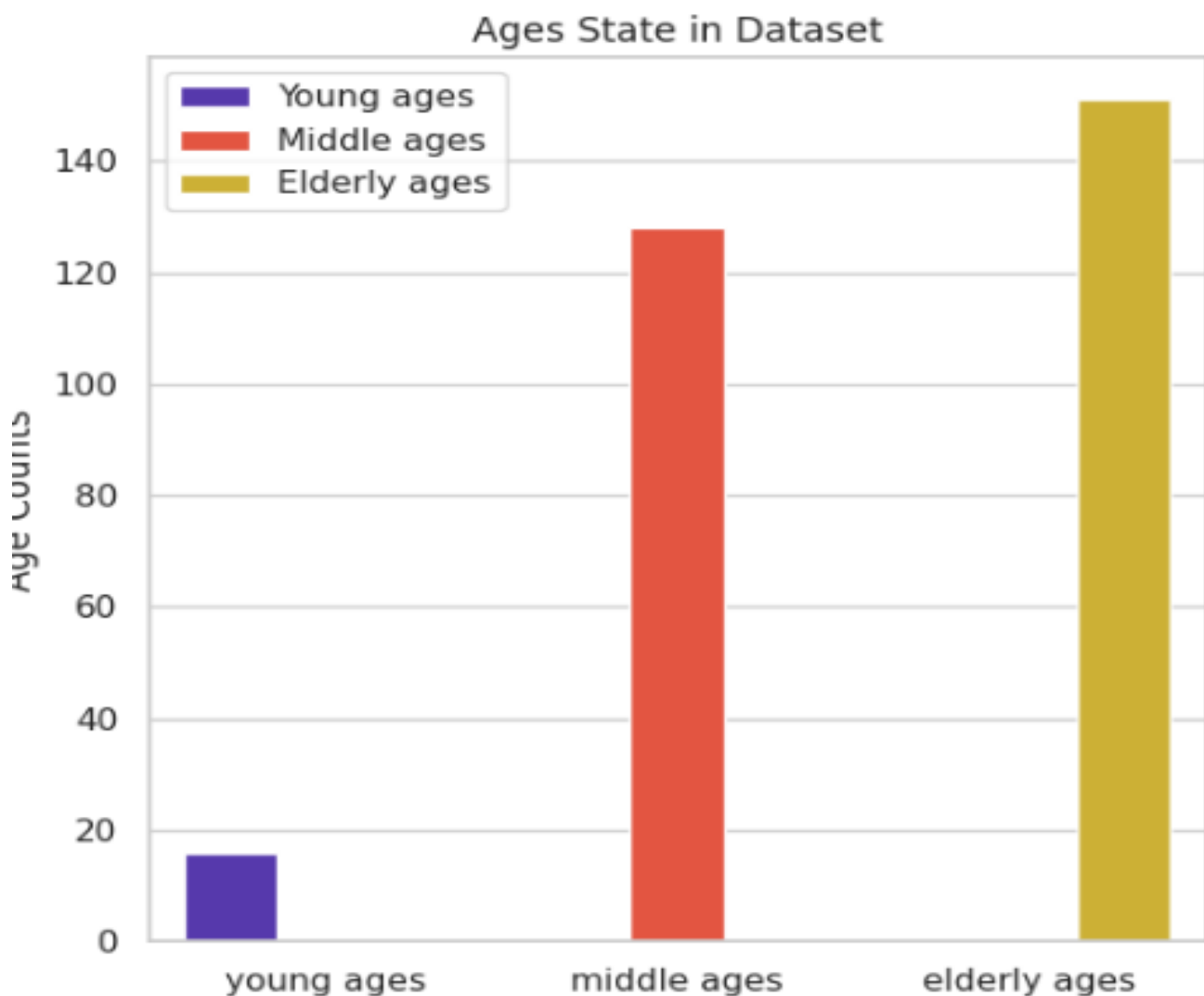


- Here I use 29-39 as Young age
- 40-54 as Middle age and
- older than 55 as Eardly age

```
[12] 1 young_ages=df[(df.age>=29)&(df.age<40)]
      2 middle_ages=df[(df.age>=40)&(df.age<55)]
      3 elderly_ages=df[(df.age>55)]
      4 print('Young Ages :',len(young_ages))
      5 print('Middle Ages :',len(middle_ages))
      6 print('Elderly Ages :',len(elderly_ages))
```

```
Young Ages : 16
Middle Ages : 128
Elderly Ages : 151
```

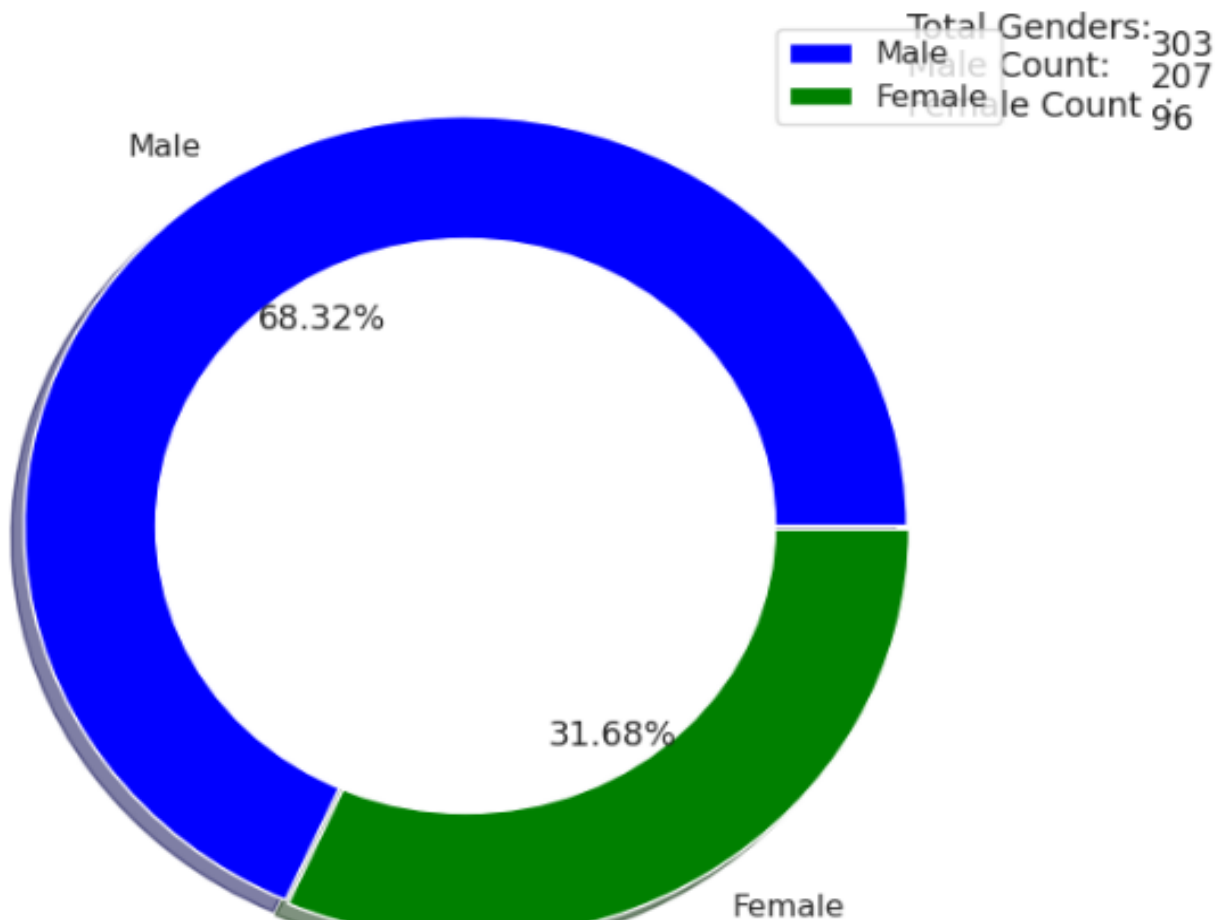
```
1 plt.figure(figsize=(6, 6))  
2 sns.barplot(x=['young ages', 'middle ages', 'elderly ages'], y=[len(young_ages), len(middle_ages), len(elderly_ages)])  
3 plt.xlabel('Age Range')  
4 plt.ylabel('Age Counts')  
5 plt.title('Ages State in Dataset')  
6 plt.show()
```



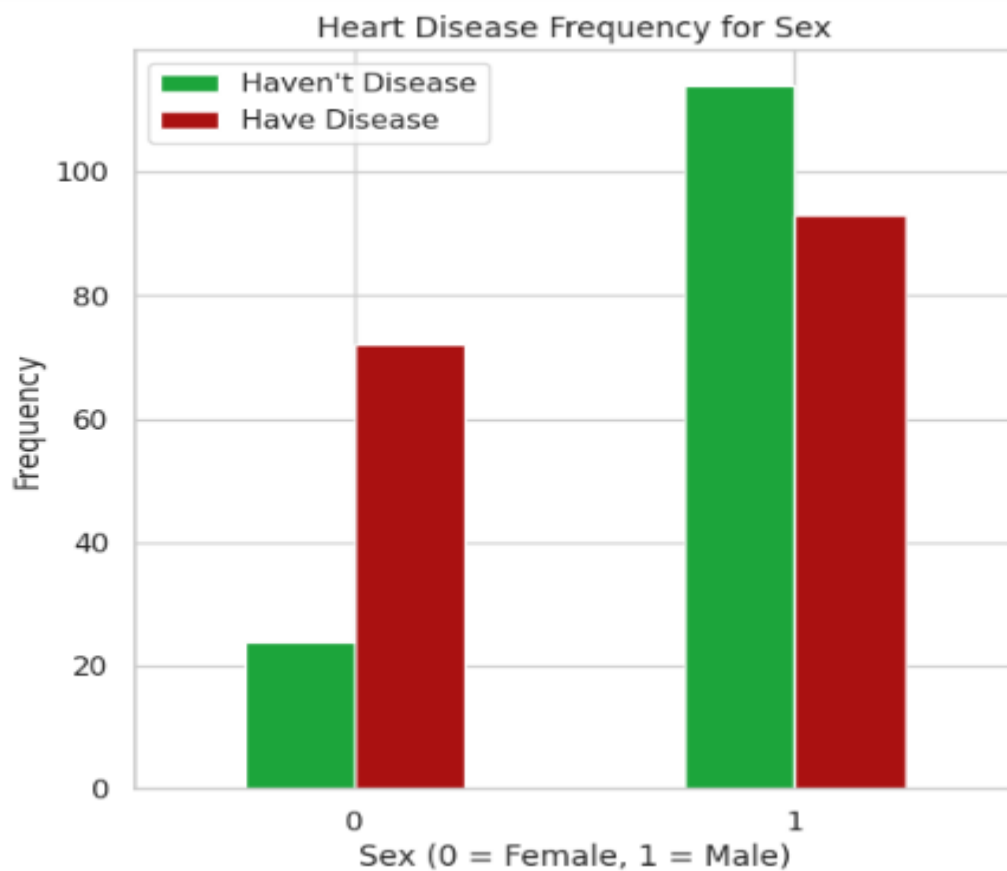
ANALYSIS USING GENDER


```
1 size = df['sex'].value_counts()
2 colors = ['blue', 'green']
3 labels = "Male", "Female"
4 explode = [0, 0.01]
5
6 my_circle = plt.Circle((0, 0), 0.7, color = 'white')
7 |
8 plt.rcParams['figure.figsize'] = (7, 7)
9 plt.pie(size, colors = colors, labels = labels, shadow = True, explode = explode, autopct = '%.2f%%')
10 plt.title('Distribution of Gender', fontsize = 20)
11 p = plt.gcf()
12 p.gca().add_artist(my_circle)
13 plt.legend()
14
15 total_genders_count=len(df.sex)
16 male_count=len(df[df['sex']==1])
17 female_count=len(df[df['sex']==0])
18 plt.text(1, 1, 'Total Genders:\nMale Count:\nFemale Count :')
19 plt.text(1.55, 1.15, total_genders_count)
20 plt.text(1.55, 1.07, male_count)
21 plt.text(1.55, .97, female_count)
22
23 plt.show()
```

Distribution of Gender



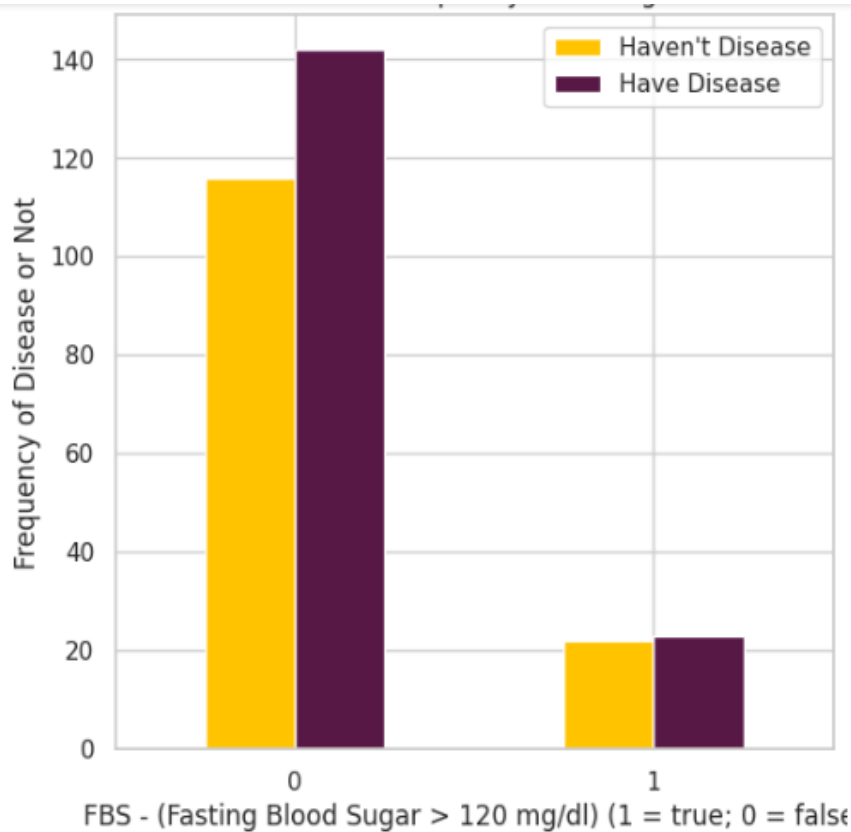
```
1 pd.crosstab(df.sex,df.target).plot(kind="bar",figsize=(6,6),color=['#1CA53B','#AA1111' ])
2 plt.title('Heart Disease Frequency for Sex')
3 plt.xlabel('Sex (0 = Female, 1 = Male)')
4 plt.xticks(rotation=0)
5 plt.legend(["Haven't Disease", "Have Disease"])
6 plt.ylabel('Frequency')
7 plt.show()
```



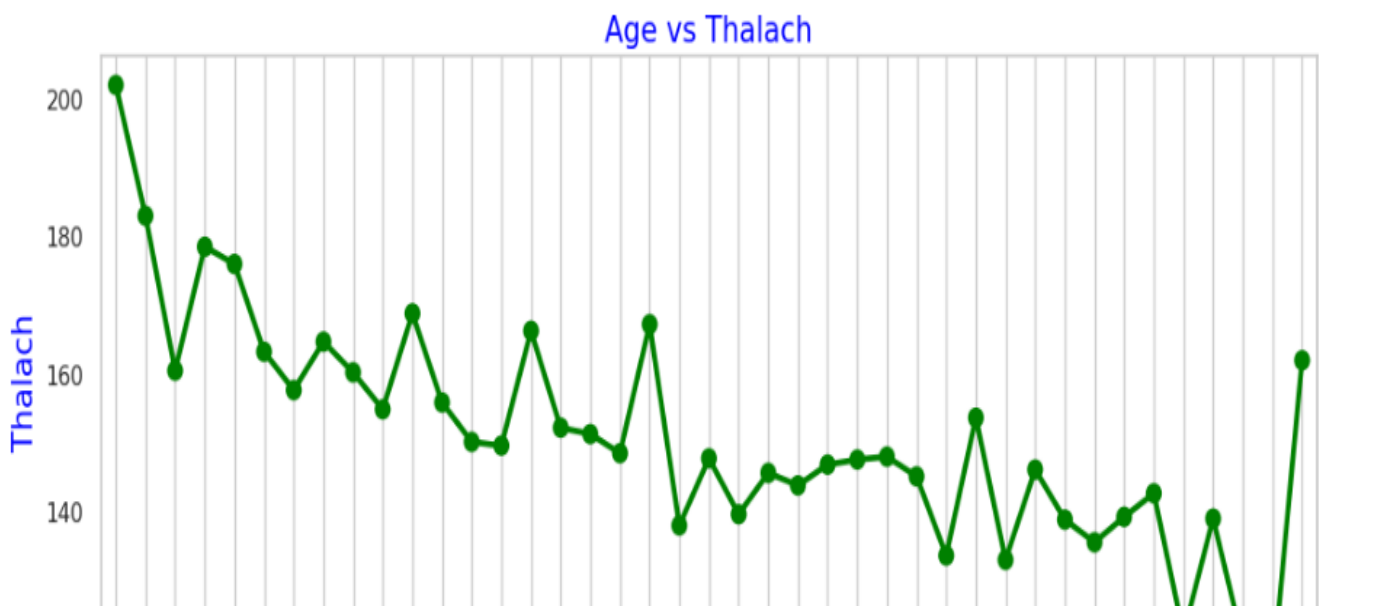
ANALYSIS USING FBS

The person's fasting blood sugar > 120 mg/dl. (1 = true; 0 = false)

```
1 pd.crosstab(df.fbs,df.target).plot(kind="bar",figsize=(6,6),color=['#FFC300','#581845' ])
2 plt.title('Heart Disease Frequency According To FBS')
3 plt.xlabel('FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
4 plt.xticks(rotation = 0)
5 plt.legend(["Haven't Disease", "Have Disease"])
6 plt.ylabel('Frequency of Disease or Not')
7 plt.show()
```



```
[28] 1 #data_sorted=data.sort_values(by='Age',ascending=True)
2 plt.figure(figsize=(12,5))
3 sns.pointplot(x=age_unique,y=mean_thalach,color='green')
4 plt.xlabel('Age',fontsize = 15,color='blue')
5 plt.xticks(rotation=45)
6 plt.ylabel('Thalach',fontsize = 15,color='blue')
7 plt.title('Age vs Thalach',fontsize = 15,color='blue')
8 plt.grid()
9 plt.show()
```



MACHINE LEARNING MODELLING

```
[37] 1 from sklearn.model_selection import train_test_split
      2 from sklearn.preprocessing import StandardScaler
      3 StandardScaler = StandardScaler()
      4 columns_to_scale = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
      5 df[columns_to_scale] = StandardScaler.fit_transform(df[columns_to_scale])
```

```
[38] 1 X= df.drop(['target'], axis=1) #, 'trestbps', 'chol', 'fbs', 'restecg'
      2 y= df['target']
      3
      4 #devide Dataset into test and train
      5 X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.3,random_state=40)
```

```
[39] 1 #check those dataset
      2 print('X_train-', X_train.size)
      3 print('X_test-', X_test.size)
      4 print('y_train-', y_train.size)
      5 print('y_test-', y_test.size)
```

```
X_train- 2756
X_test- 1183
y_train- 212
y_test- 91
```

LOGISTIC REGRESSION

```
[40] 1 from sklearn.linear_model import LogisticRegression
      2 lr = LogisticRegression(random_state=999, C = 1, tol = 5)
      3 model1 = lr.fit(X_train,y_train)
      4 prediction1 = model1.predict(X_test)
      5
      6 y_pred_quant1 = model1.predict_proba(X_test)[:, 1]
```

```
[41] 1 from sklearn.metrics import confusion_matrix
      2
      3 cm = confusion_matrix(y_test,prediction1)
      4 cm
```

```
array([[36,  4],
       [ 3, 48]])
```

```
[44] 1 from sklearn.metrics import accuracy_score
      2
      3 accuracies = {}
      4
      5 acc = accuracy_score(y_test, prediction1)*100
      6 accuracies['Logistic Regration'] = acc
      7 acc
```

92.3076923076923

```
[71] 1 from sklearn.metrics import classification_report
      2 print(classification_report(y_test, prediction1))
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.92 | 0.90 | 0.91 | 40 |
| 1 | 0.92 | 0.94 | 0.93 | 51 |
| accuracy | | | 0.92 | 91 |
| macro avg | 0.92 | 0.92 | 0.92 | 91 |
| weighted avg | 0.92 | 0.92 | 0.92 | 91 |

DECISION TREE

```
[45] 1 from sklearn.tree import DecisionTreeClassifier
      2
      3 dtc = DecisionTreeClassifier(random_state=999)
      4 model2 = dtc.fit(X_train,y_train)
      5 prediction2 = model2.predict(X_test)
      6
      7 y_pred_quant2 = model2.predict_proba(X_test)[: , 1]
```

```
[46] 1 cm2 = confusion_matrix(y_test,prediction2)
      2 cm2

array([[32,  8],
       [13, 38]])
```

```
[47] 1 acc = accuracy_score(y_test,prediction2)*100
      2 accuracies['Decision Tree'] = acc
      3 acc
```

76.92307692307693

RANDOM FOREST

```
1 from sklearn.ensemble import RandomForestClassifier
2
3 rfc=RandomForestClassifier(random_state=20)
4 model3 = rfc.fit(X_train, y_train)
5 prediction3 = model3.predict(X_test)
6
7 y_pred_quant3 = model3.predict_proba(X_test)[:, 1]
8
9 cm3 = confusion_matrix(y_test, prediction3)
10 cm3
```

```
array([[33,  7],
       [ 7, 44]])
```

```
[49] 1 acc = accuracy_score(y_test, prediction3)*100
      2 accuracies['Random Forest'] = acc
      3 acc
```

```
84.61538461538461
```

K NEAREST NEIGHBOUR

```
[75] 1 from sklearn.neighbors import KNeighborsClassifier
      2
      3 KNN = KNeighborsClassifier(n_neighbors = 5)
      4 model4 = KNN.fit(X_train, y_train)
      5 prediction4 = model4.predict(X_test)
      6
      7 y_pred_quant4 = model4.predict_proba(X_test)[: , 1]
      8
      9 cm4= confusion_matrix(y_test, prediction4)
     10 cm4
```

```
array([[34,  6],
       [ 7, 44]])
```

```
[76] 1 acc = accuracy_score(y_test, prediction4)*100
      2 accuracies['K nearest neighbour'] = acc
      3 acc
```

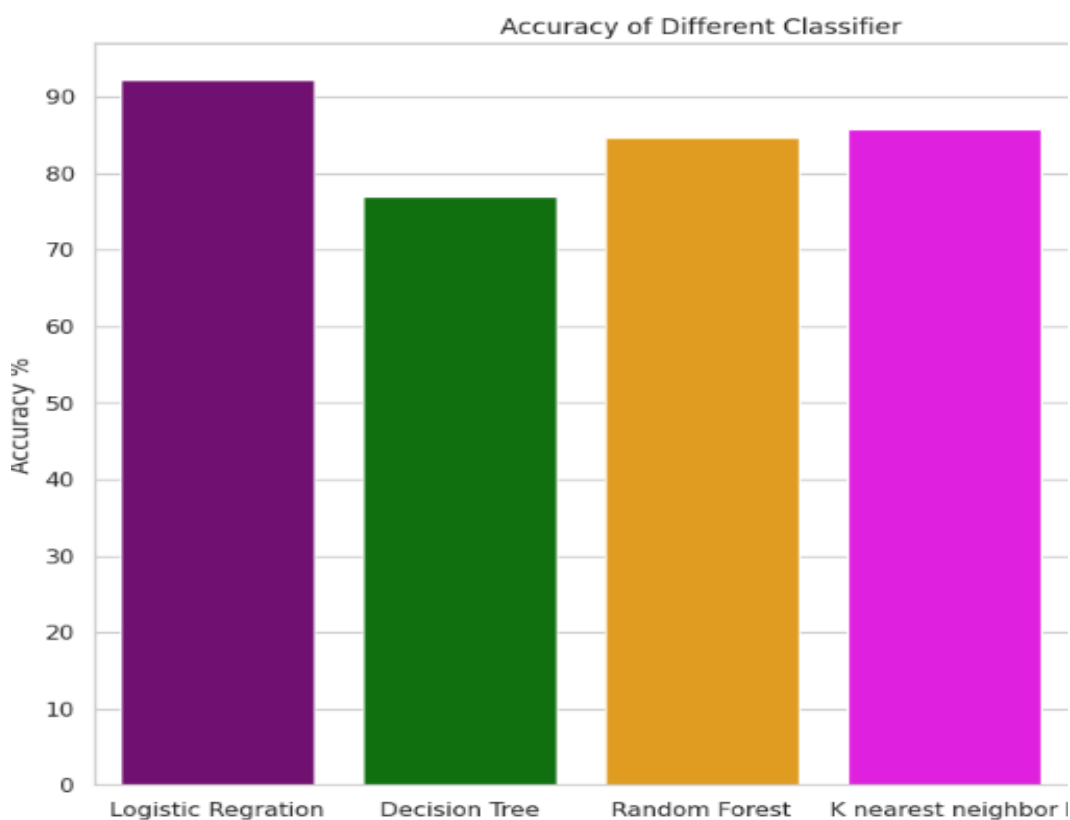
```
85.71428571428571
```

COMPARING THE MODELS

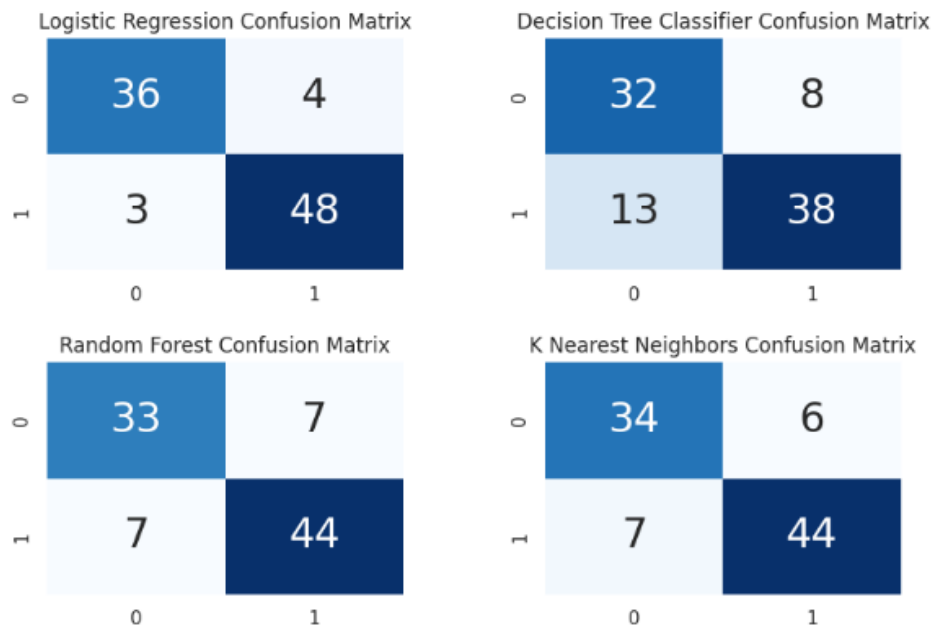
```
[78] 1 print('Logistic Regression - lr :', accuracy_score(y_test, prediction1) * 100)
      2 print('Decision Tree - dtc :', accuracy_score(y_test, prediction2) * 100)
      3 print('Random Forest - rfc :', accuracy_score(y_test, prediction3) * 100)
      4 print('K Nearest - KNN :', accuracy_score(y_test, prediction4) * 100)
```

```
Logistic Regression - lr : 92.3076923076923
Decision Tree - dtc : 76.92307692307693
Random Forest - rfc : 84.61538461538461
K Nearest - KNN : 85.71428571428571
```

```
1 colors = ["purple", "green", "orange", "magenta"]
2
3 sns.set_style("whitegrid")
4 plt.figure(figsize=(10,8))
5 plt.yticks(np.arange(0,100,10))
6 plt.ylabel("Accuracy %")
7 plt.xlabel("Algorithms")
8 plt.title("Accuracy of Different Classifier")
9 sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
10 plt.show()
```



Confusion Matrixes



The code first defines a function called `confusion_matrix()`. This function takes the true labels and predicted labels as input and returns a confusion matrix.

The code then creates a confusion matrix for the test set predictions. The confusion matrix shows that the model correctly classified 85 positive instances and 32 negative instances. The model incorrectly classified 15 positive instances as negative and 30 negative instances as positive.

The code then uses the Seaborn library to plot the confusion matrix. The plot shows the confusion matrix as a heatmap, with the true labels on the x-axis and the predicted labels on the y-axis. The cells of the heatmap are colored according to the number of instances in each cell.

The code also prints the accuracy, precision, recall, and F1 score of the model. The accuracy is 0.92, which means that the model correctly classified 92% of the instances in the test set by using the Machine Learning Algorithm Logistic Regression.

ITM(SLS) BARODA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE, ENGINEERING AND TECHNOLOGY
BTECH - CSE SEMESTER-7
DEEP LEARNING

