Introduction to Data Analytics

Project Report

Title - Probabilistic Classification using Statistical Naïve Bayes Classifier (Topic - 6)

Date of Submission: 24 November 2022

Group Members -

- ➔ Kakarla Venkata Seshasai Pavan Teja - S20190020216 - UG4
- ➔ Kumar Vamsi Krishna Kurakula - S20190020227 - UG4
- ➔ Kamarthi Litheesh Kumar - S20190020218 - UG4
- ➔ Banu Theja V - S20190020258 - UG4

Abstract -

A heart attack (Cardiovascular disease) occurs when the flow of blood to the heart muscle suddenly becomes blocked. From WHO statistics every year 17.9 million die from a heart attack. The medical study says that the human lifestyle is the main reason behind this heart problem. Apart from this, many vital factors warn that the person may/may not get a chance of a heart attack.

This dataset contains some medical information of patients which tells whether that person getting a heart attack chance is less or more. Using the info explore the dataset and classify the target variable using different Machine Learning models and find out which algorithm is suitable for this dataset.

1. Introduction -

A myocardial infarction (MI), commonly known as a heart attack, occurs when blood flow decreases or stops to the coronary artery of the heart, causing damage to the heart muscle. The most common symptom is chest pain or discomfort which may travel into the shoulder, arm, back, neck or jaw. Often it occurs in the center or left side of the chest and lasts for more than a few minutes. The discomfort may occasionally feel like heartburn. Other symptoms may include shortness of breath, nausea, feeling faint, cold sweat, or feeling tired.

Data Science and machine learning (ML) can be very helpful in the prediction of heart attacks in which there are different risk factors like high blood pressure, high cholesterol, abnormal pulse rate, diabetes, etc... can be considered. The objective of this study is to optimize the prediction of heart disease using ML.

In our project, we are going to use the Naive Bayes Classifier to predict heart attacks.

Features that we have in our project Heart attack dataset:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholesterol in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise-induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- the number of major vessels (0-3) colored by fluoroscopy
- thal: 0 = normal; 1 = fixed defect; 2 = reversible defect
- Target

2. Key concepts for project implementation:

2.1. Naive Bayes:

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes models are a group of extremely fast and simple classification algorithms that are often suitable for very high-dimensional datasets. Because they are so fast and have so few tunable parameters, they end up being very useful as a quick-and-dirty baseline for a classification problem.

2.2. Bayesian classification:

Naive Bayes classifiers are built on Bayesian classification methods. These rely on Bayes's theorem, which is an equation describing the relationship of conditional probabilities of statistical quantities.

$$\boxed{P(A|B)= \frac{P(B|A)P(A)}{P(B)}}$$

Where

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: The probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: The probability of the hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

2.3. Advantages of Naïve Bayes Classifier:

- ➔ It is used for **Credit Scoring**. It is used in **medical data classification**.
- ➔ It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- ➔ It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

2.4. Correlation Analysis:

- ➔ In statistics, the word correlation is used to denote some form of association between two variables.
- ➔ The correlation may be positive, negative, or zero. The correlation coefficient(r) is used to measure the degree of correlation.
- ➔ A correlation coefficient is a number between -1 and 1 that tells you the strength and direction of a relationship between variables.
- ➔ we must choose the most relevant and non-redundant features from the original feature set to reduce the number of features.
- ➔ Here we use correlation analysis.

2.5. Confusion Matrix:

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier.

		Actuals	
		+	-
Predicted	+	True Positive (TP)	False Positive (FP)
	-	False Negative (FN)	True Negative (TN)

2.5.1. True Positives (TP):

➔ When the actual value is Positive and the predicted is also Positive.

2.5.2. True Negatives (TN):

➔ When the actual value is Negative and the prediction is also Negative.

2.5.3. False Positives (FP):

➔ When the actual is Negative but the prediction is Positive. Also known as the Type 1 error.

2.5.4. False Negatives (FN):

➔ When the actual is Positive but the prediction is Negative. Also known as the Type 2 error.

2.5.5. Accuracy

➔ It's the ratio between the number of correct predictions and the total number of predictions.

$$\boxed{Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Correct\ Predictions}{Total\ Predictions}}$$

2.5.6. Precision

➔ The ratio of the total number of correctly classified positive classes divided by the total number of predicted positive classes.

$$\boxed{Precision = \frac{TP}{TP + FP} = \frac{Predictions\ Actually\ Positive}{Total\ Predicted\ Positive}}$$

2.5.7. Recall

➔ The ratio of the total number of correctly classified positive classes divided by the total number of positive classes.

$$\boxed{Recall = \frac{TP}{TP + FN} = \frac{Predictions\ Actually\ Positive}{Total\ Actual\ Positive}}$$

2.5.8. F1-Score

- ➔ The F1 score is a number between 0 and 1 and is the harmonic mean of precision and recall.
- ➔ In practice, when we try to increase the precision of our model, the recall goes down and vice-versa.
- ➔ The F1-score captures both trends in a single value.

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

2.6. K-Fold Cross Validation:

- ➔ In k-fold cross-validation, the original sample is randomly partitioned into k equal-sized subsamples.
- ➔ Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data.
- ➔ The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data.
- ➔ The k results can then be averaged to produce a single estimation.

2.7. Gaussian Naïve Bayes algorithm:

- ➔ In Gaussian Naïve Bayes, the assumption is made that the continuous numerical attributes are distributed normally.
- ➔ The attribute is first segmented based on the output class, and then the variance and mean of the attribute are calculated for each class.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

3. Implementation of Naive Bayer's Classifier:

3.1. Steps of implementation:

1. Library like
   a. pandas - for data cleaning and analysis
   b. Numpy - for working with arrays
   c. Matplotlib - for visualization
   d. Seaborn - to visualize random distributions
   e. EDA - it is applied to investigate the data and summarize the key insights. It will give you a basic understanding of your data, its distribution, null values, and much more.
   f. Sklearn - for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction. Here, used for data splitting, Naive Bayer's Classifier, K-fold cross-validation,confusion_matrix,accuracy_score,classification_report, and roc_curve.
2. Importing, and visualization(like Histogram, etc..) of the data.
3. Data preprocessing
a. Correlation
4. Splitting data (in 80% training data and 20% testing data) and scaling using standardscaler()
5. Modeling - Naive Bayes classifier from sklearn
6. Accuracy, Confusion Matrix, Precision, F1_score, Recall from sklearn
7. K fold cross validation
8. Calculate average of each k value's accuracy and find the maximum of all K value's average
9. Printing the max accuracy and k value at max accuracy

3.2. Data Preparation:

1. Read, analyze and preprocess data.
2. Run correlation analysis on the columns.
3. Remove features that are highly correlated to others.
4. During correlation analysis, the columns are numerical type.
5. Split the data into training and testing sets.

3.3. Building and using the Naive Bayesian Classifier:

1. Use the Gaussian Naive Bayes in the sklearn tool.
2. Predict the test data.
3. Build a Confusion Matrix and compute Accuracy, Precision, Recall, and F1-Score.
4. Run k-fold cross-validation and capture Accuracy for each of k runs.
5. Calculate average accuracy from k-folds.
6. Find the Maximum accuracy from all K values.

4. Experimental Results:

4.1. Data Visualization:

4.1.1. Structure of data:

```
df.info()
```

As you can see there are

No null values in attributes

```
<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   int64
 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   int64
 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   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

4.1.2. Dataset Attributes:

```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

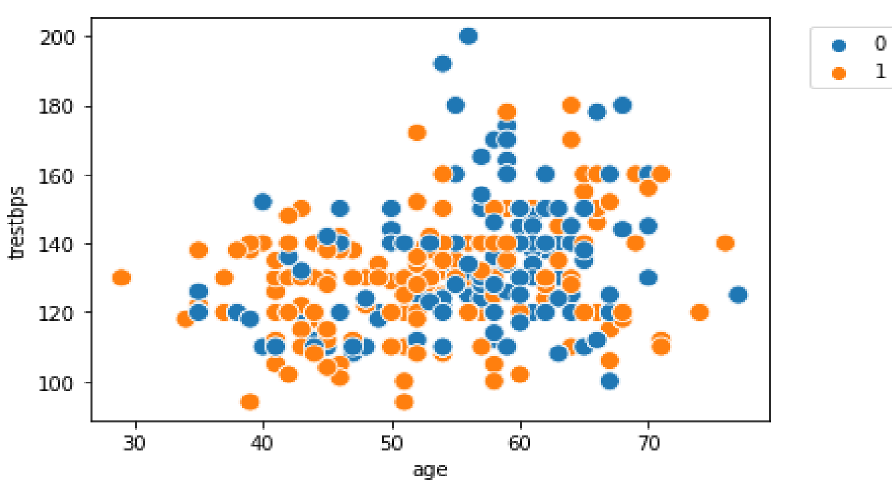
4.1.3. Summary:

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.754146	2.323902	0.513171
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660	0.500070
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

4.1.4. Scatter Plot:

```
sns.scatterplot(x = 'age', y = 'trestbps', s = 100, hue ='target', data=df);
plt.legend(bbox_to_anchor=(1.2, 1))
```

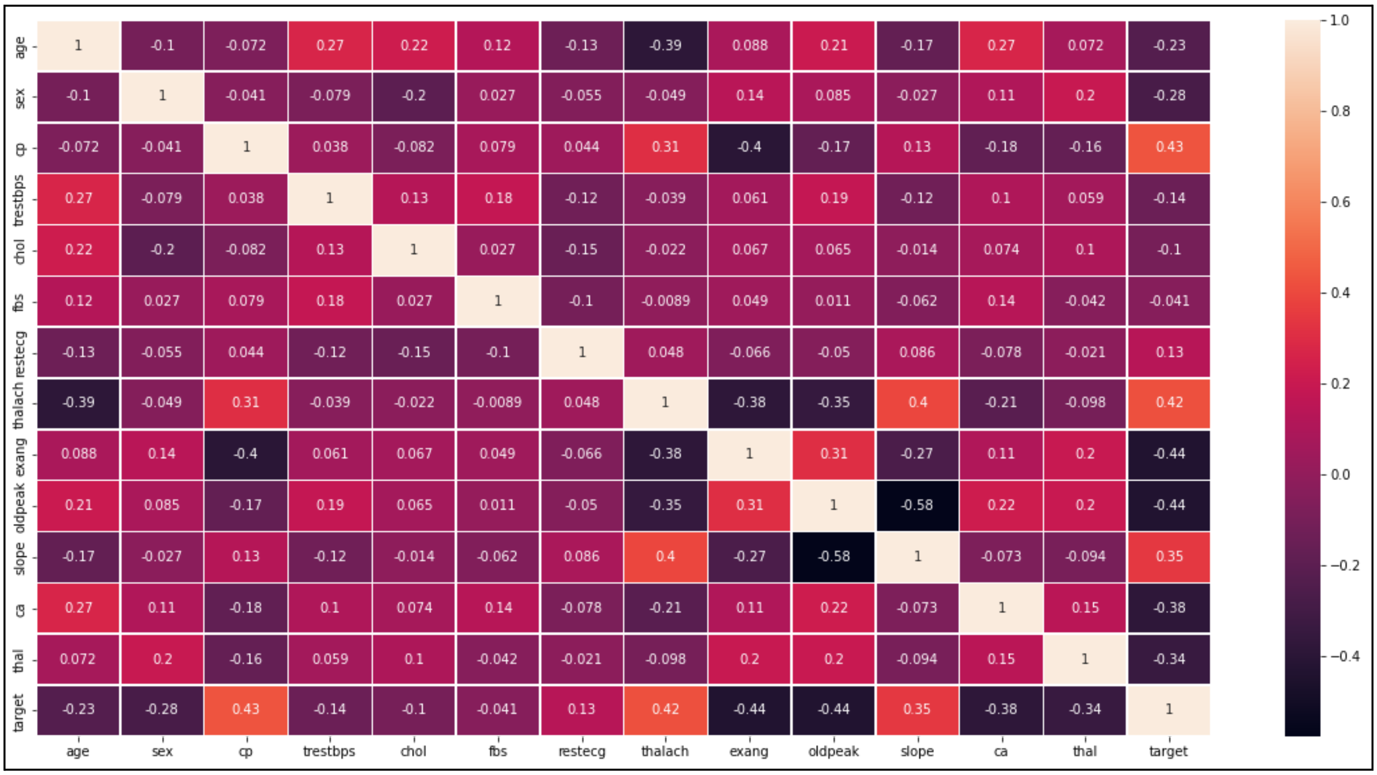


4.2. Data Preprocessing:

4.2.1. Correlation:

```
corr = df.corr()
```

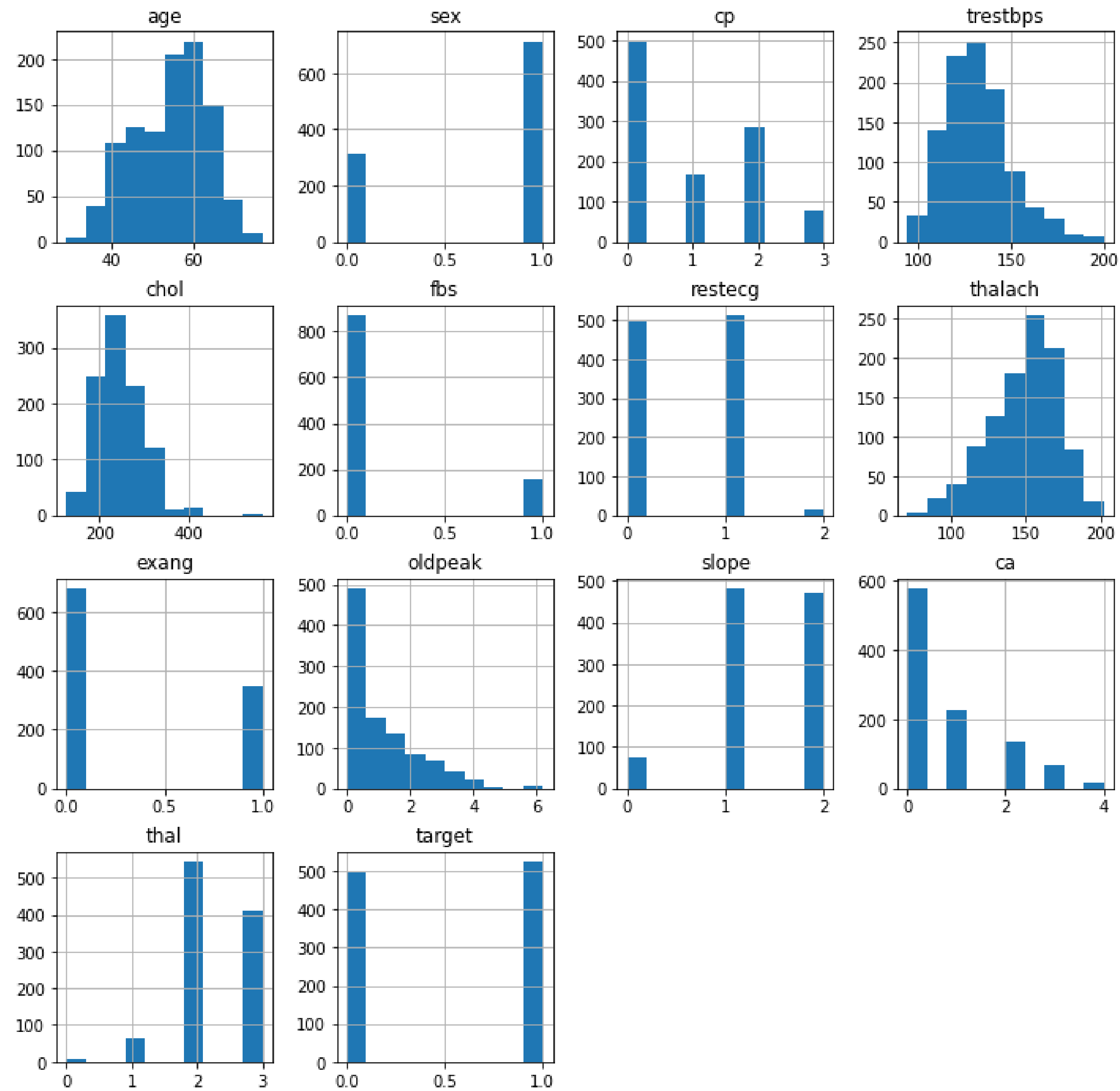
```
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot = True, linewidth = 0.5)
```



Here, the correlation between the attributes is observed, and found every attribute has less correlation with each other. No need to drop any attribute from the dataset.

4.2.2. Histogram:

```
df.hist(grid = True,figsize=(12,12))
plt.show()
```



4.2.3. Splitting Data:

➔ Splitting data into training and testing using the sklearn tool.

```
[186] y = df["target"]
      X = df.drop('target',axis=1)
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state = 0)
```

➔ Training data - 80% - (419+401 = 820 instances in training set)

```
[187] Counter(y_train)

      Counter({1: 419, 0: 401})
```

➔ Testing data - 20% - (107+98 = 205 instances in testing set)

```
[188] Counter(y_test)

      Counter({1: 107, 0: 98})
```

4.2.4. Scaling:

```
[189] scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

4.3. Modeling Using Naive Bayes:

```
# training the model on training set
nb = GaussianNB()
nb.fit(X_train,y_train)

# making predictions on the testing set
nbpred = nb.predict(X_test)

#Confusion Matrix
nb_conf_matrix = confusion_matrix(y_test, nbpred)

# comparing actual response values (y_test) with predicted response values (y_pred)
nb_acc_score = accuracy_score(y_test, nbpred)
```

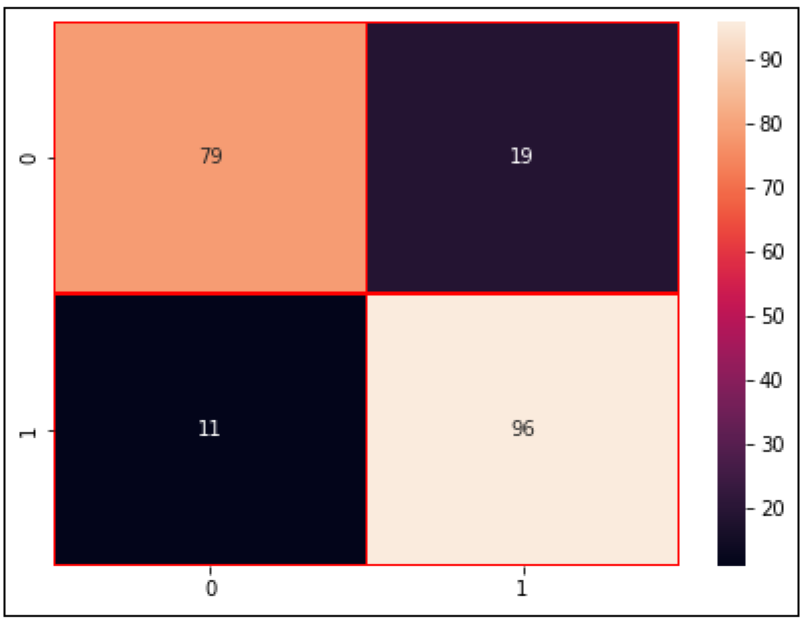
4.3.1. Accuracy:

```
print("Accuracy of Naive Bayes model:",nb_acc_score*100,'\n')
```

```
Accuracy of Naive Bayes model: 85.36585365853658
```

4.3.2. Confusion Matrix:

```
print("confusion matrix")
f,ax = plt.subplots(figsize=(7, 5))
sns.heatmap(nb_conf_matrix, annot=True, linewidths=0.5,linecolor="red", fmt= '.0f',ax=ax)
plt.show()
```



4.3.3. Precision Score:

```
from sklearn.metrics import  precision_score
p_score = precision_score(y_test, nbpred)
print("Precision Score:",p_score)
```

```
Precision Score: 0.8347826086956521
```

4.3.4. Recall Score:

```
from sklearn.metrics import  recall_score
r_score = recall_score(y_test, nbpred)
print("Recall:",r_score)
```

```
Recall: 0.897196261682243
```

4.3.5. F1 Score:

```
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, nbpred)
print("F1 Score:",f1_score)
```

```
F1 Score: 0.8648648648648648
```

4.4. K-Fold Cross Validation: K = 5,6,7,8,9,10,11,12

```
X = np.array(X_train)
y = np.array(y_train)
split = [5,6,7,8,9,10,11,12]
kfoldacc = []
max_acc_index=0
max_acc=0
for s in split:
  kf = KFold(n_splits=s)
  avg=[]
  for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    nb1 = GaussianNB()
    nb1.fit(X_train,y_train)
    nbpred1 = nb.predict(X_test)
    nb_acc_score1 = accuracy_score(y_test, nbpred1)
    avg.append(nb_acc_score1)
  kfoldacc.append((sum(avg)/len(avg))*100)
  average=(sum(avg)/len(avg))*100
  if average>max_acc:
    max_acc=average
    max_acc_index=s
print("K-Fold Max Acc - ", max_acc)
print("K =", max_acc_index)
```

```
K-Fold Max Acc -  81.71428571428572
K = 7
```

4.5. Prior and Posterior Probabilities:

4.5.1. Contingency table for all Attributes V/S Target(class):

```
for i in df.columns:
  if i!="target":
    ct = pd.crosstab(df[i], df['target'], margins = True)
    print("\nContigency Table of",i)
    print("\n")
    print(ct)

    ct.columns = ["0","1","rowtotal"]
    l=list([str(j) for j in list(Counter(df[i]).keys())])+["coltotal"]
    ct.index= l
    print("-----------------------------------------------")
```

```
Contigency Table of slope


target    0    1   All
slope
0        46   28    74
1       324  158   482
2       129  340   469
All     499  526  1025
-------------------------
```

```
Contigency Table of thal


target    0    1   All
thal
0         4    3     7
1        43   21    64
2       132  412   544
3       320   90   410
All     499  526  1025
-------------------------
```

```
Contigency Table of ca


target    0    1   All
ca
0       163  415   578
1       160   66   226
2       113   21   134
3        60    9    69
4         3   15    18
All     499  526  1025
-------------------------
```

```
Contigency Table of sex


target    0    1   All
sex
0        86  226   312
1       413  300   713
All     499  526  1025
---------------------------
```

```
Contigency Table of cp


target    0    1   All
cp
0       375  122   497
1        33  134   167
2        65  219   284
3        26   51    77
All     499  526  1025
---------------------------
```

4.5.2. Prior and Posterior Probabilities: (restecg and target)

```
ct1 = pd.crosstab(df['restecg'], df['target'], margins = True)
print(ct1)
print("----------------------------------------------------")
ct1.columns = ["0","1","rowtotal"]
ct1.index= ["0","1","2","coltotal"]
print(ct1 / ct1.loc["coltotal","rowtotal"])
print("----------------------------------------------------")
bayesposterior(prior = ct1.iloc[1,1]/ct1.iloc[3,1],
               likelihood = ct1.iloc[3,1]/ct1.iloc[3,2],
               evidence = ct1.iloc[1,2]/ct1.iloc[3,2],
               string = 'Posterior Probability =')
```

```
target     0    1   All
restecg
0        283  214   497
1        204  309   513
2         12    3    15
All      499  526  1025
----------------------------------------------------
                 0         1  rowtotal
0         0.276098  0.208780  0.484878
1         0.199024  0.301463  0.500488
2         0.011707  0.002927  0.014634
coltotal  0.486829  0.513171  1.000000
----------------------------------------------------
Prior= 0.5874524714828897
Likelihood= 0.5131707317073171
Evidence= 0.5004878048780488
Equation = (Prior*Likelihood)/Evidence
Posterior Probability = 0.6023391812865497
```

4.5.3. Prior and Posterior Probabilities: (slope and target)

```
ct2 = pd.crosstab(df['slope'], df['target'], margins = True)
print("Contigency Table of slope\n")
print(ct2)
print("----------------------------------------------------------")
ct2.columns = ["0","1","rowtotal"]
ct2.index= ["0","1","2","coltotal"]
print(ct2 / ct2.loc["coltotal","rowtotal"])
print("----------------------------------------------------------")
bayesposterior(prior = ct2.iloc[1,1]/ct2.iloc[3,1],
               likelihood = ct2.iloc[3,1]/ct2.iloc[3,2],
               evidence = ct2.iloc[1,2]/ct2.iloc[3,2],
               string = 'Posterior Probability =')
```

```
Contigency Table of slope

target    0    1   All
slope
0        46   28    74
1       324  158   482
2       129  340   469
All     499  526  1025
----------------------------------------------------------
                 0         1  rowtotal
0         0.044878  0.027317  0.072195
1         0.316098  0.154146  0.470244
2         0.125854  0.331707  0.457561
coltotal  0.486829  0.513171  1.000000
----------------------------------------------------------
Prior= 0.30038022813688214
Likelihood= 0.5131707317073171
Evidence= 0.47024390243902436
Equation = (Prior*Likelihood)/Evidence
Posterior Probability = 0.3278008298755187
```

4.5.4. Prior and Posterior Probabilities: (ChestPain and target)

```
ct3 = pd.crosstab(df['cp'], df['target'], margins = True)
print("Contigency Table of ChestPain\n")
print(ct3)
print("----------------------------------------------------------")
ct3.columns = ["0","1","rowtotal"]
ct3.index= ["0","1","2","3","coltotal"]
print(ct3 / ct3.loc["coltotal","rowtotal"])
print("----------------------------------------------------------")
bayesposterior(prior = ct3.iloc[2,1]/ct3.iloc[4,1],
               likelihood = ct3.iloc[4,1]/ct3.iloc[4,2],
               evidence = ct3.iloc[2,2]/ct3.iloc[4,2],
               string = 'Posterior Probability =')
```

```
Contigency Table of ChestPain

target    0    1   All
cp
0       375  122   497
1        33  134   167
2        65  219   284
3        26   51    77
All     499  526  1025
----------------------------------------------------------
                 0         1  rowtotal
0         0.365854  0.119024  0.484878
1         0.032195  0.130732  0.162927
2         0.063415  0.213659  0.277073
3         0.025366  0.049756  0.075122
coltotal  0.486829  0.513171  1.000000
----------------------------------------------------------
Prior= 0.41634980988593157
Likelihood= 0.5131707317073171
Evidence= 0.27707317073170734
Equation = (Prior*Likelihood)/Evidence
Posterior Probability of Tennis given Rain = 0.7711267605633803
```