Heart Disease Prediction

Shahzada Muhammad Ali (MSCS-4)

Hamza Tariq (MSCS-4)

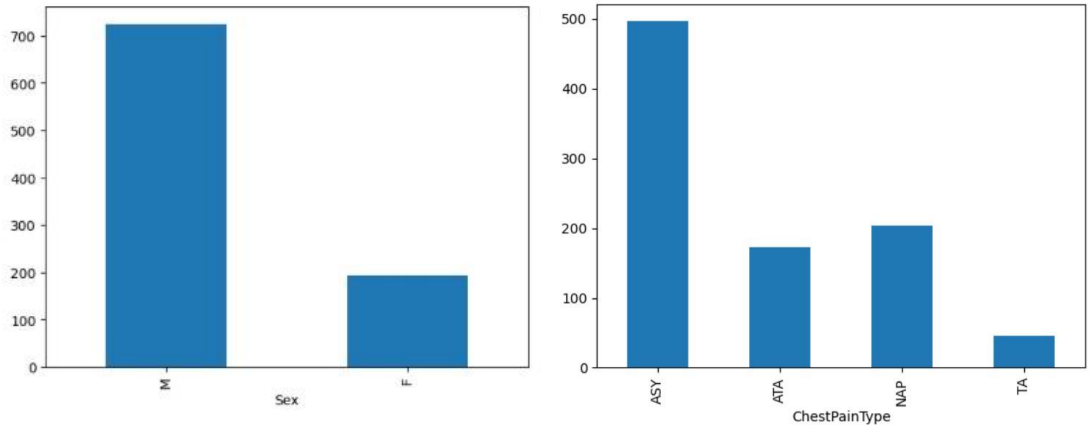
Hammad Ahmed (MSCS-4)

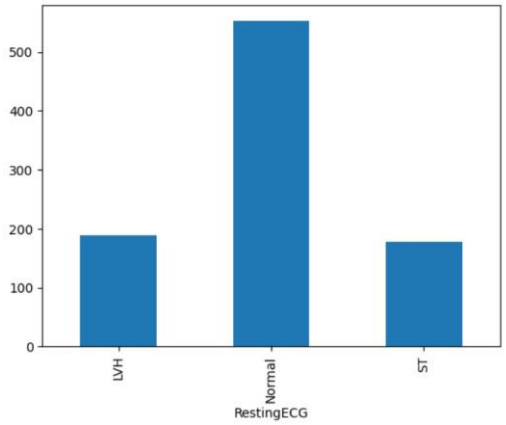
Work Flow:

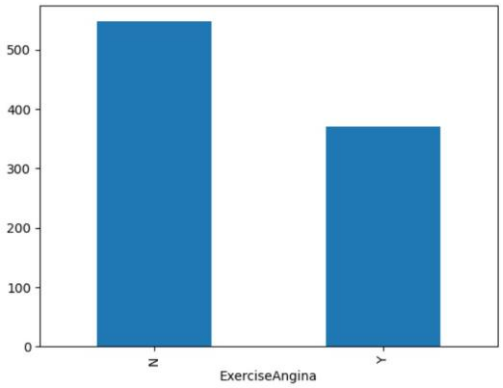
1. We first started by importing all the necessary libraries
2. We then checked for the **missing values**

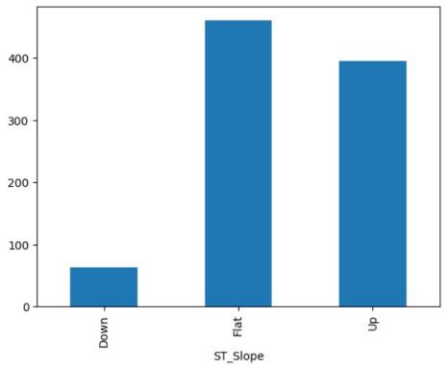
```
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
missing values
```

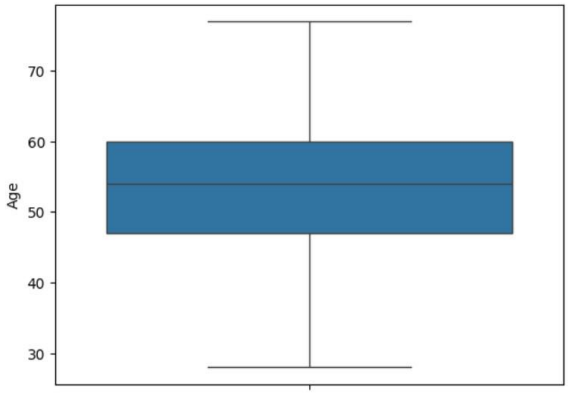
3. Since there were no missing values, we moved on to treat the **outliers**

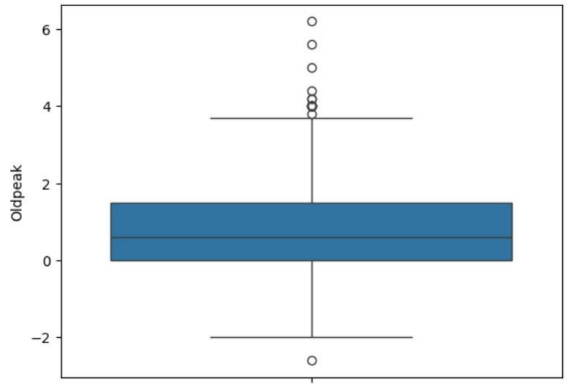


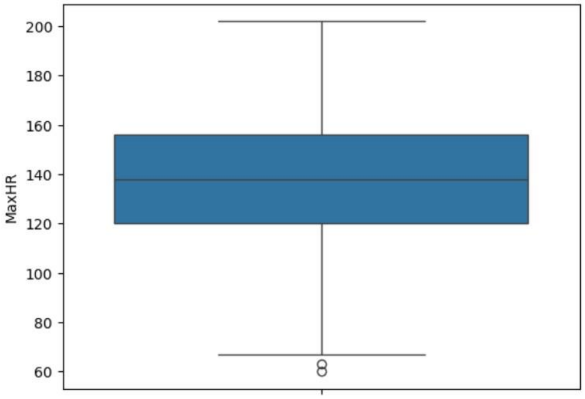

500
400
300
200
100
0
LVH
Normal
ST
RestingECG

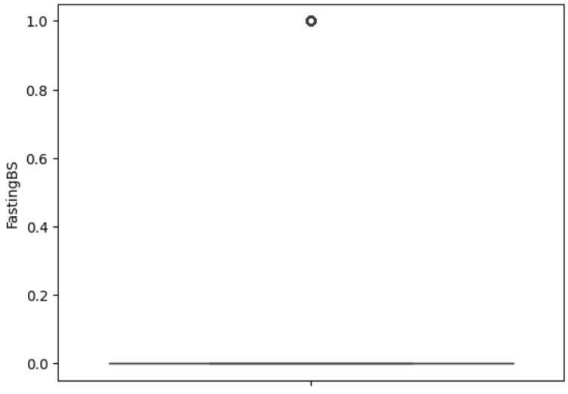

500
400
300
200
100
0
N
Y
ExerciseAngina

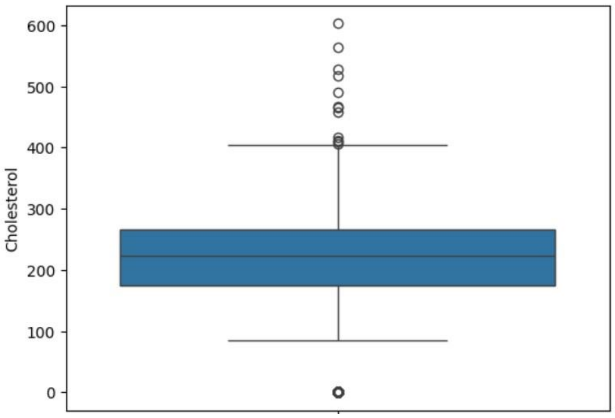

400
300
200
100
0
Down
Flat
Up
ST_Slope

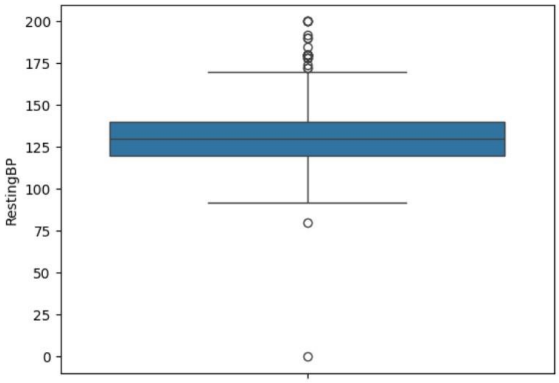

70
60
50
40
30
Age


6
4
2
0
−2
Oldpeak

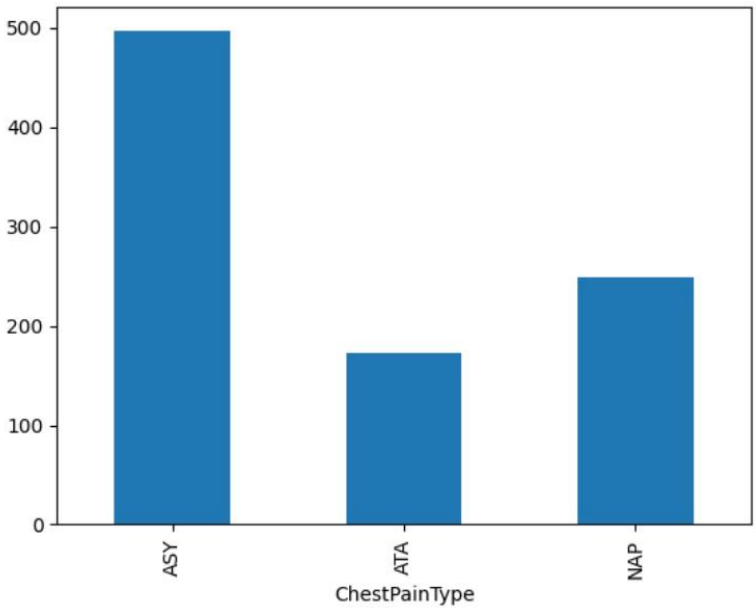

200
180
160
140
120
100
80
60
MaxHR


1.0
0.8
0.6
0.4
0.2
0.0
FastingBS


600
500
400
300
200
100
0
Cholesterol

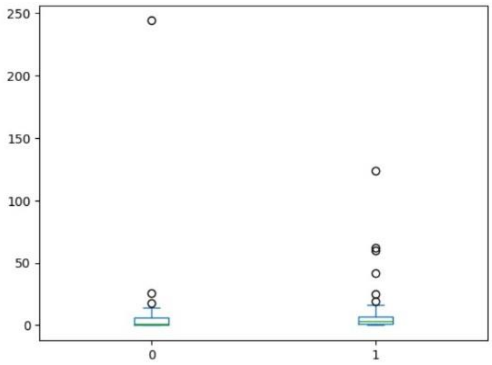


4. **ChestPainType outliers "TA"** had a total percentage of more than 5%. We extracted those outliers and looked at the dataset from the point of view of those outliers. There was no change in the mean and median of any of the columns as compared to the mean and median of these columns when considering the entire dataset. But there was a change in Heart Disease Prediction column. The median was changed from 1 to 0. This meant that these outliers carried some information. So we checked for that information by comparing the Heart Disease prediction for all the values in ChestPainType column. The closest values to TA were NAP, so we merged TA with NAP with no effect to the overall Heart Disease Prediction.

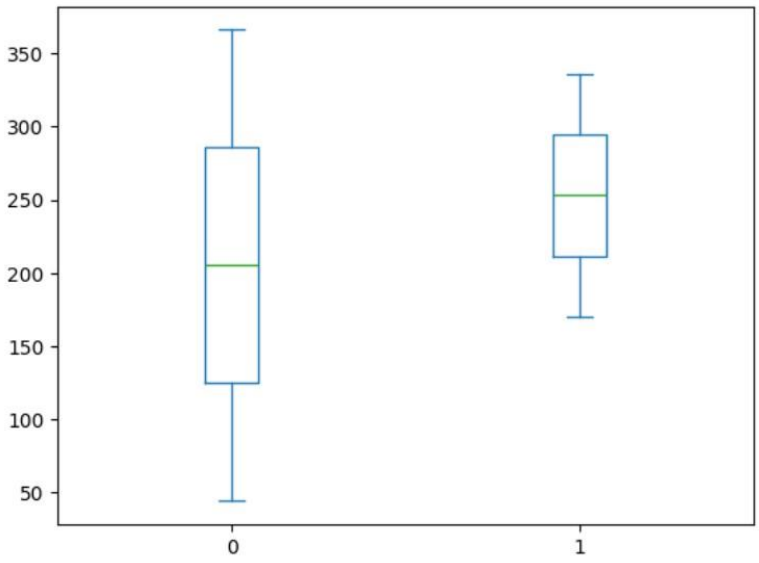


5. **ST_Slope outliers "Down"** had a total percentage of more than 6%. We again checked these values against all columns in the dataset and we noticed that all values of ST_Slope column were significantly changing the mean and median of the Oldpeak column. We thought that they carry an important information, so we kept them.

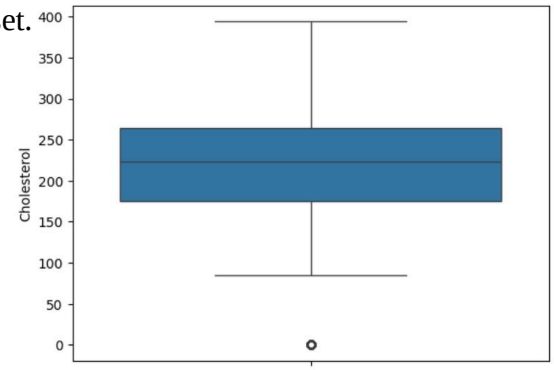
6. **Oldpeak outliers** were 16 in total. Upon further mining, we found out that most of them had ChestPainType “ASY” and most of them predicted Heart Disease as positive. Most of them also had high cholesterol levels so we kept these outliers as they carried an important information.

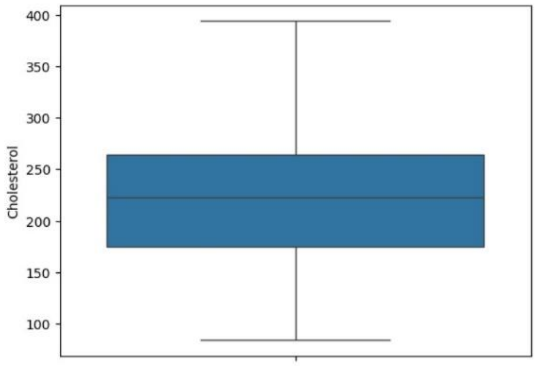


7. **MaxHR outliers** were only two in number. So we removed them.
8. We could not understand the readings in FastingBS column because they were mostly 0s and 1s but since they were normally distributed between 0s and 1s of Heart Disease column so we kept them as it is.

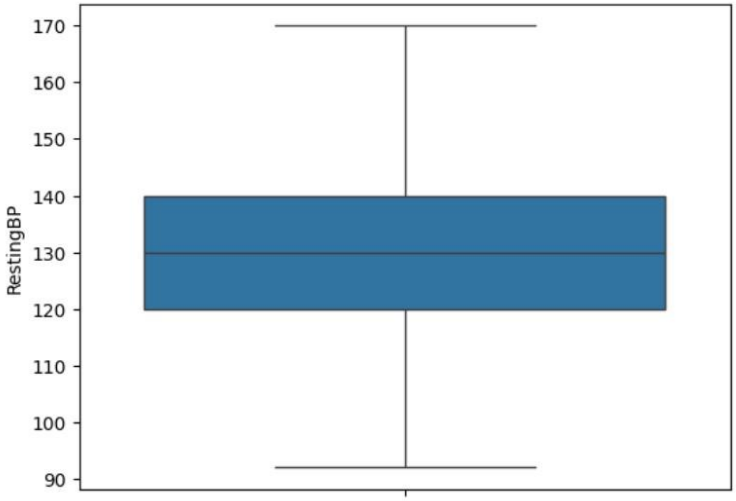


9. **Cholesterol outliers** were a total of 182 data points. The outliers above the upper bound were only 12 and they were not significantly altering the results or the mean and medians of other columns, so we replaced them with the median values of cholesterol. The outliers below the lower bound were 170 but most of them were 0s. Total cholesterol levels of 0 are usually linked with no Heart Disease risk but in our dataset they were showing increases risk of Heart Disease so we thought that the data is faulty or these are calcium levels, LDL or HDL. But since there was no way of knowing for sure so we replaced all the values of 0 with 84, so that they just fall within the lower whisker without any damage to the overall dataset.





10. **RestingBP outliers** were a total of 28 values. Since they had 0 values and a systolic blood pressure of 0 is not possible, we assumed that these are diastolic blood pressure values. Since they were not significantly altering the mean and median of other columns nor were they significantly altering the results, we simply replaced them with median RestingBP values.



11. After treating the outliers, we moved to **<u>One-Hot Encoding</u>**

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	M	ATA	NAP	Normal	ST	Y	Flat	Up
0	40	140	289	0	172	0.0	0	1	1	0	1	0	0	0	1
1	49	160	180	0	156	1.0	1	0	0	1	1	0	0	1	0
2	37	130	283	0	98	0.0	0	1	1	0	0	1	0	0	1
3	48	138	214	0	108	1.5	1	0	0	0	1	0	1	1	0
4	54	150	195	0	122	0.0	0	1	0	1	1	0	0	0	1

12. Then we moved to feature scaling. We used **<u>RobustScalar</u>** because we were keeping some outliers and we didn't want our model to get affected because of them.

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	M	ATA	NAP	Normal	ST	Y	Flat	Up
0	3.076923	7.0	3.238095	0	4.777778	0.000000	0	1	1	0	1	0	0	0	1
1	3.769231	8.0	2.016807	0	4.333333	0.666667	1	0	0	1	1	0	0	1	0
2	2.846154	6.5	3.170868	0	2.722222	0.000000	0	1	1	0	0	1	0	0	1
3	3.692308	6.9	2.397759	0	3.000000	1.000000	1	0	0	0	1	0	1	1	0
4	4.153846	7.5	2.184874	0	3.388889	0.000000	0	1	0	1	1	0	0	0	1

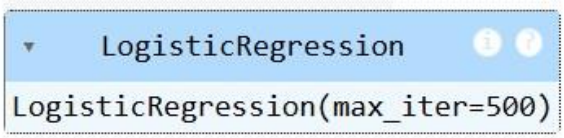
13. The we checked for **Multi-collinearity using Variance Inflation Factor (VIF)**

	variables	VIF
0	Age	33.480017
1	RestingBP	61.417911
2	Cholesterol	10.489477
3	FastingBS	1.472418
4	MaxHR	27.140383
5	Oldpeak	2.207944
6	M	5.014841
7	ATA	1.849939
8	NAP	1.779452
9	Normal	3.837220
10	ST	1.995399
11	Y	2.601169
12	Flat	2.580027

14. Then we kept removing one column after the other to check for VIF values. We stopped when we got VIF values less than 4 for all columns. So we were left with only 9 columns at the end.

	variables	VIF
0	FastingBS	1.319693
1	Oldpeak	1.994701
2	ATA	1.709852
3	NAP	1.622900
4	Normal	3.492627
5	ST	1.843964
6	Y	2.384977
7	Flat	3.556525
8	Up	3.269055

15. Then we moved on to applying different **Models/Algorithms** on our dataset.
16. We first applied **Logistic Regression** and got 86% accuracy on train and 82% on test. We checked the model with different values of VIF and adding some of the removed features back and this is the best result that we got.

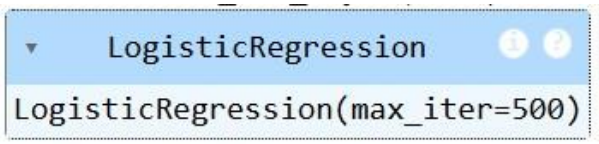



```
0.8611544461778471
0.8290909090909091
```

17. We then applied **Principal Component Analysis (PCA)** and got 84% accuracy on train and 80% accuracy on test at 2 principal components. We checked the model with 3 and 4 principal components but these were the best results that we got.

	PCA 1	PCA 2
0	-1.213511	0.269172
1	0.211047	0.579801
2	-1.038932	-1.015214
3	0.863286	0.630806
4	-1.084786	0.203612
...	...	...
911	0.298576	0.560761
912	1.407609	0.302624
913	0.731993	0.659366
914	-0.247883	-0.017288
915	-1.084786	0.203612

916 rows × 2 columns




```
0.8455538221528861
0.8
```

18. We then applied **Apriori Algorithm** and got 92-98% confidence at 1.67-1.77 lift on our training data and 90-94% confidence at 1.62-1.69 on our test data.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty
143	(Y, FastingBS, Flat)	(HeartDisease)	0.078003	0.550702	0.076443	0.980000	1.779547	1.0	0.033487	22.464899	0.475120	0.138418	0.955486
130	(Normal, FastingBS, Flat)	(HeartDisease)	0.070203	0.550702	0.068643	0.977778	1.775511	1.0	0.029982	20.218409	0.469761	0.124294	0.950540
60	(FastingBS, Flat)	(HeartDisease)	0.138846	0.550702	0.134165	0.966292	1.754655	1.0	0.057703	13.329173	0.499431	0.241573	0.924977
54	(Y, FastingBS)	(HeartDisease)	0.104524	0.550702	0.096724	0.925373	1.680352	1.0	0.039162	6.020593	0.452147	0.173184	0.833903
42	(FastingBS, ST)	(HeartDisease)	0.060842	0.550702	0.056162	0.923077	1.676182	1.0	0.022656	5.840874	0.429540	0.101124	0.828793

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty
130	(Y, FastingBS, Flat)	(HeartDisease)	0.065455	0.556364	0.061818	0.944444	1.697531	1.0	0.025402	7.985455	0.439689	0.110390	0.874772
155	(Y, ST, Flat)	(HeartDisease)	0.061818	0.556364	0.058182	0.941176	1.691657	1.0	0.023788	7.541818	0.435804	0.103896	0.867406
48	(Y, FastingBS)	(HeartDisease)	0.112727	0.556364	0.105455	0.935484	1.681425	1.0	0.042737	6.876364	0.456755	0.187097	0.854574
112	(Y, Flat)	(HeartDisease)	0.280000	0.556364	0.254545	0.909091	1.633987	1.0	0.098764	4.880000	0.538889	0.437500	0.795082
142	(Normal, Y, Flat)	(HeartDisease)	0.152727	0.556364	0.138182	0.904762	1.626206	1.0	0.053210	4.658182	0.454484	0.242038	0.785324

19. We then applied **K-Nearest Neighbor (KNN)** and got 83% accuracy on our test data at k=4. We checked the classifier for values of k ranging from 1 to 9. And these were the best results that we got. We used cosine similarity as a measure because these were 0/1 values.

```
k=4
0.8327272727272728
/usr/local/lib/python3.10/dist-packages,
  return self._fit(X, y)
```

KNeighborsClassifier

```
KNeighborsClassifier(metric='cosine')
```

20. We then applied **Decision Tree** on our dataset using both Gini Index and Entropy as a criterion and got an accuracy of 81.8% and an F-1 score of 0.83 by using Gini Index and got an accuracy of 82.1% and an F-1 score of 0.83 by using Entropy.

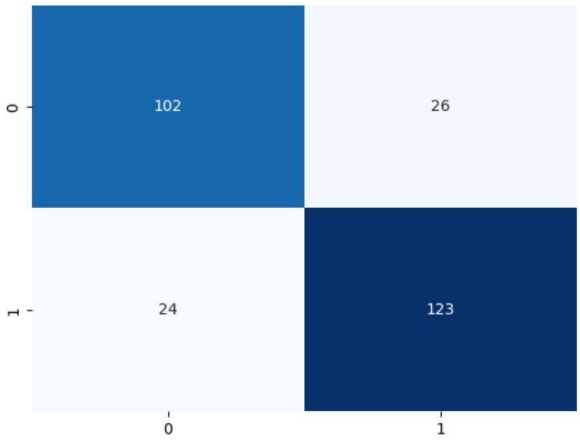
Gini Index

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
0.8181818181818182
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.80      0.80       128
           1       0.83      0.84      0.83       147

    accuracy                           0.82       275
   macro avg       0.82      0.82      0.82       275
weighted avg       0.82      0.82      0.82       275

F1 Score : 0.831081081081081
```



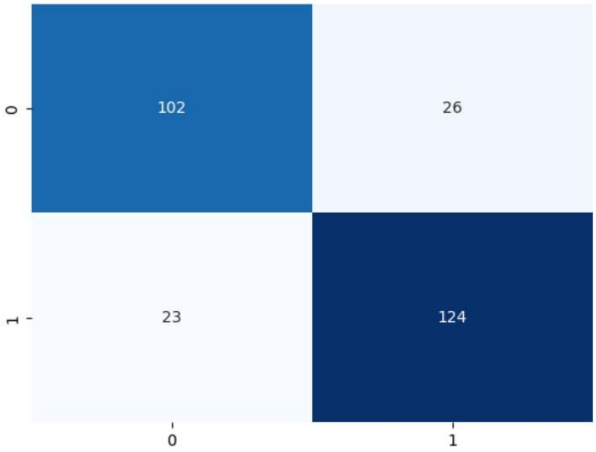
Entropy

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
0.8218181818181818
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.80      0.81       128
           1       0.83      0.84      0.84       147

    accuracy                           0.82       275
   macro avg       0.82      0.82      0.82       275
weighted avg       0.82      0.82      0.82       275

F1 Score : 0.835016835016835
```



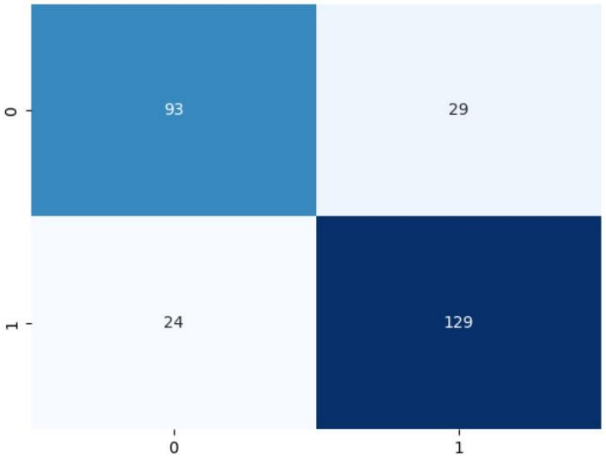
21. Lastly we applied **Gaussian Naïve Bayesian** on our dataset and got 80.7% accuracy and a 0.83 F-1 score.

```
GaussianNB
GaussianNB()
0.8072727272727273
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.76      0.78       122
           1       0.82      0.84      0.83       153

    accuracy                           0.81       275
   macro avg       0.81      0.80      0.80       275
weighted avg       0.81      0.81      0.81       275

F1 Score : 0.8295819935691319
```



Comparisons of Models:

Model Name	Accuracy (Train)	Accuracy (Test)	Other
Logistic Regression	86%	82%	
PCA with Log Reg	84%	80%	
Apriori Algorithm	92-98% confidence	90-94% confidence	
KNN			83% at k=4
Decision Tree (Gini Index)	81.8%		0.83 F-1 Score
Decision Tree (Entropy)	82.1%		0.83 F-1 Score
Gaussian Naïve Bayesian	80.7%		0.83 F-1 Score