Heart Disease Prediction Using Machine Learning

Skill Vertex : Feb batch

Domain: Data Science

Final Major Project, **Mentor: Akash Maurya**

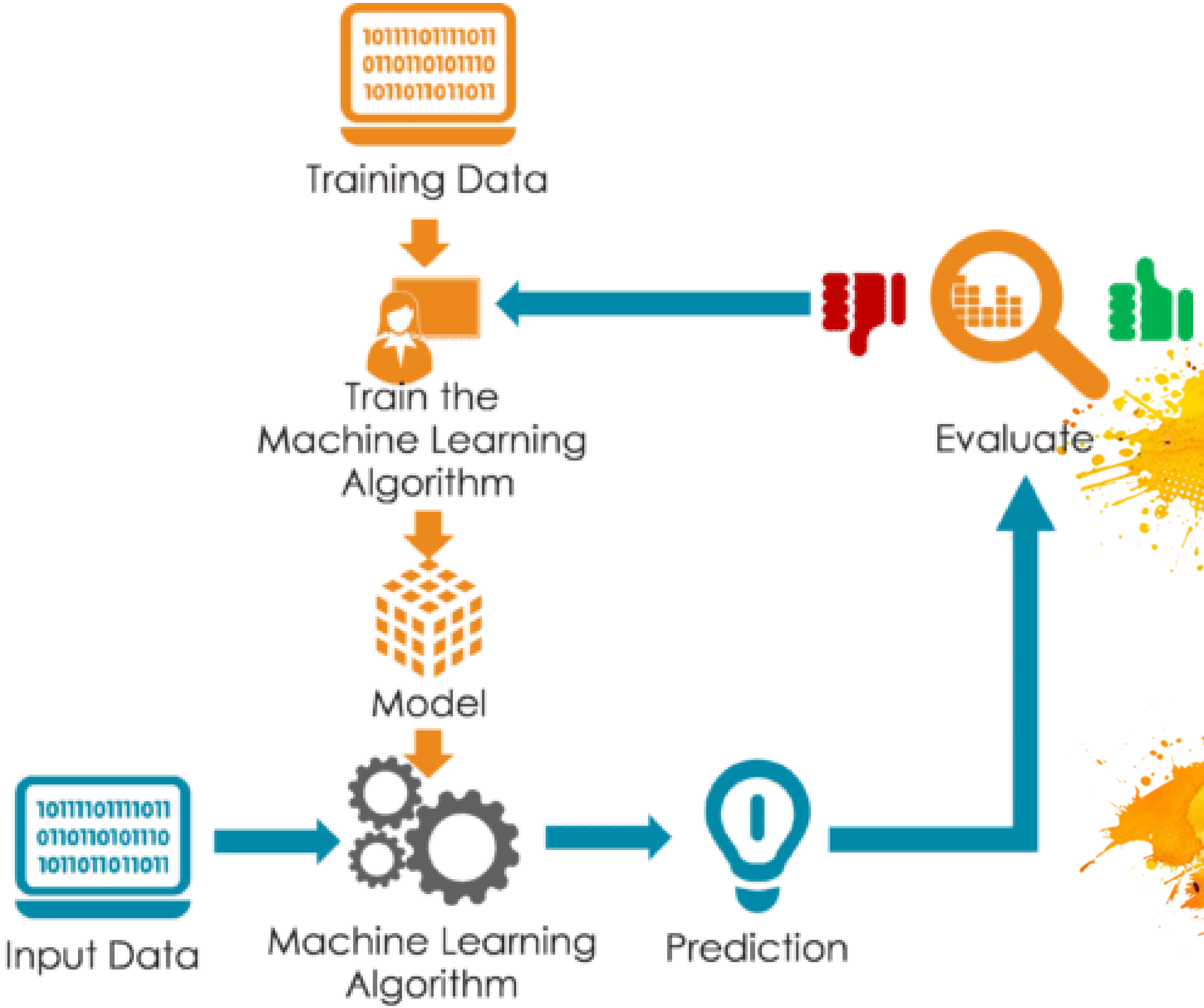
Team member:

1.Yash Tambat 2. Harshana Sundaram
3.Vishali Sudhakar 4.Dhanush Subramanian
5.Maithre Sunitha 6.Arya SS

INTRODUCTION

What is Heart Disease ?

: A type of disease that affects the heart or blood vessels. The risk of certain heart diseases may be increased by smoking, high blood pressure, high cholesterol, unhealthy diet, lack of exercise, and obesity. The most common heart disease is coronary artery disease (narrow or blocked coronary arteries), which can lead to chest pain, heart attacks, or stroke. Other heart diseases include congestive heart failure, heart rhythm problems, congenital heart disease (heart disease at birth), and endocarditis (inflamed inner layer of the heart). Also called cardiovascular disease.



Training Data

Train the Machine Learning Algorithm

Model

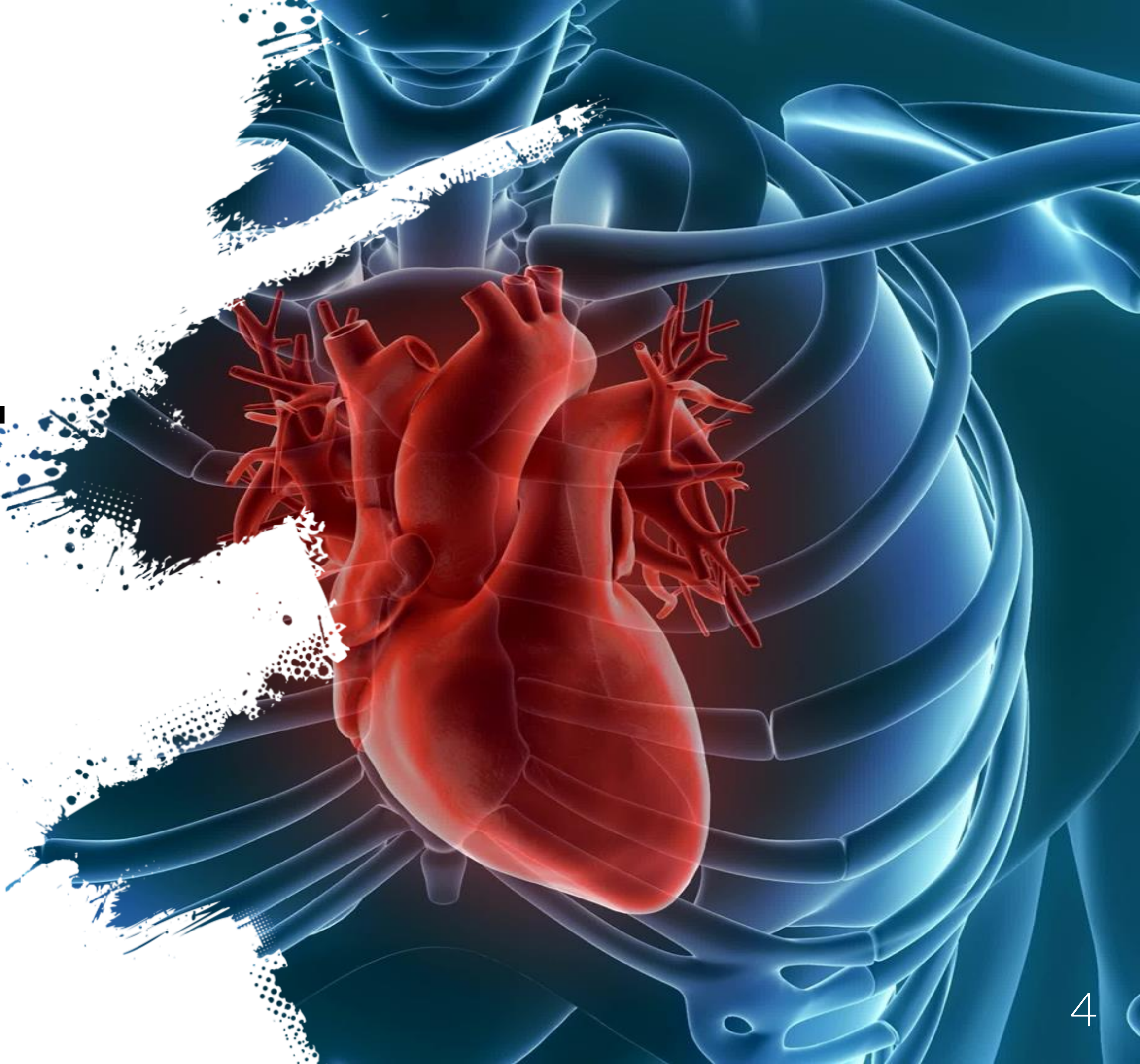
Evaluate

Input Data

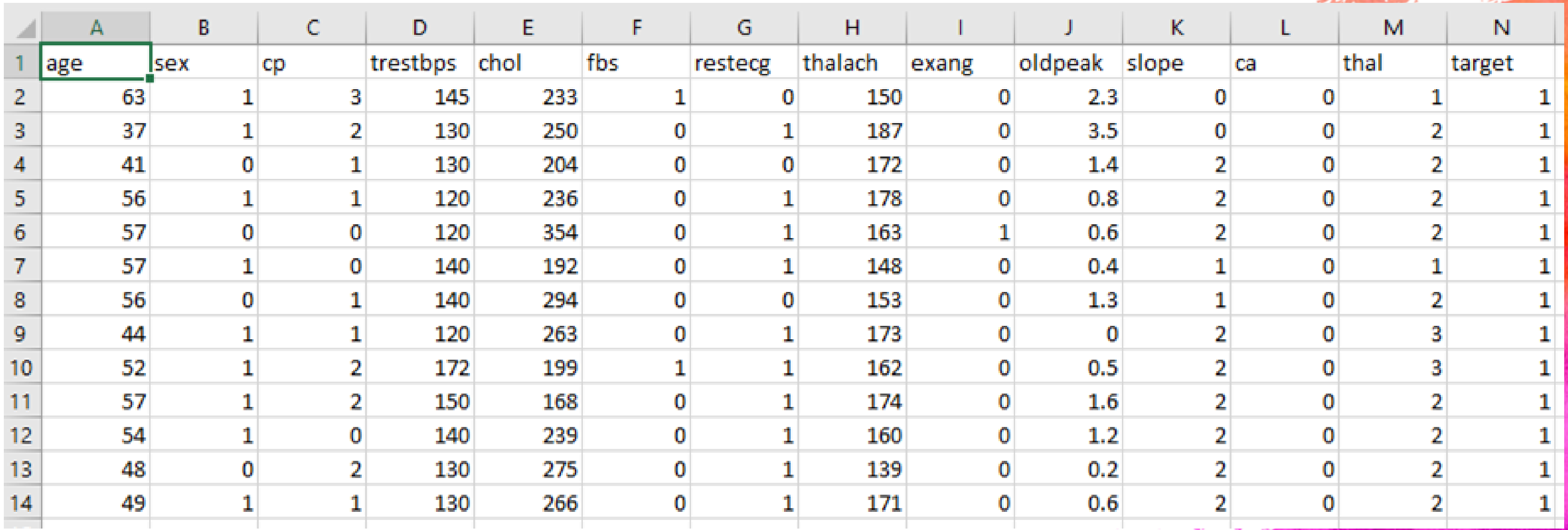
Machine Learning Algorithm

Prediction

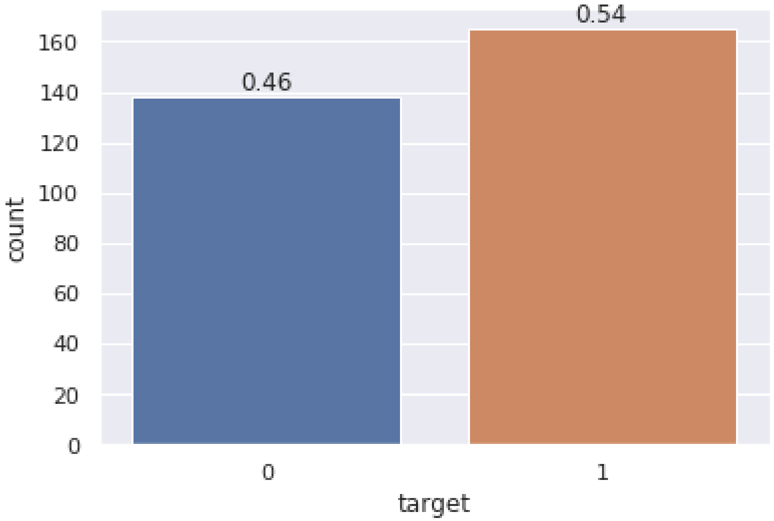
The Working of the Project



Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
2	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
3	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
4	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
5	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
6	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
7	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
8	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
9	44	1	1	120	263	0	1	173	0	0	2	0	3	1
10	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
11	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
12	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
13	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
14	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1



```
1      165
0      138
```

3 step process

Split the dataset

Train the dataset

Compare the Algos

scikit learn

NumPy

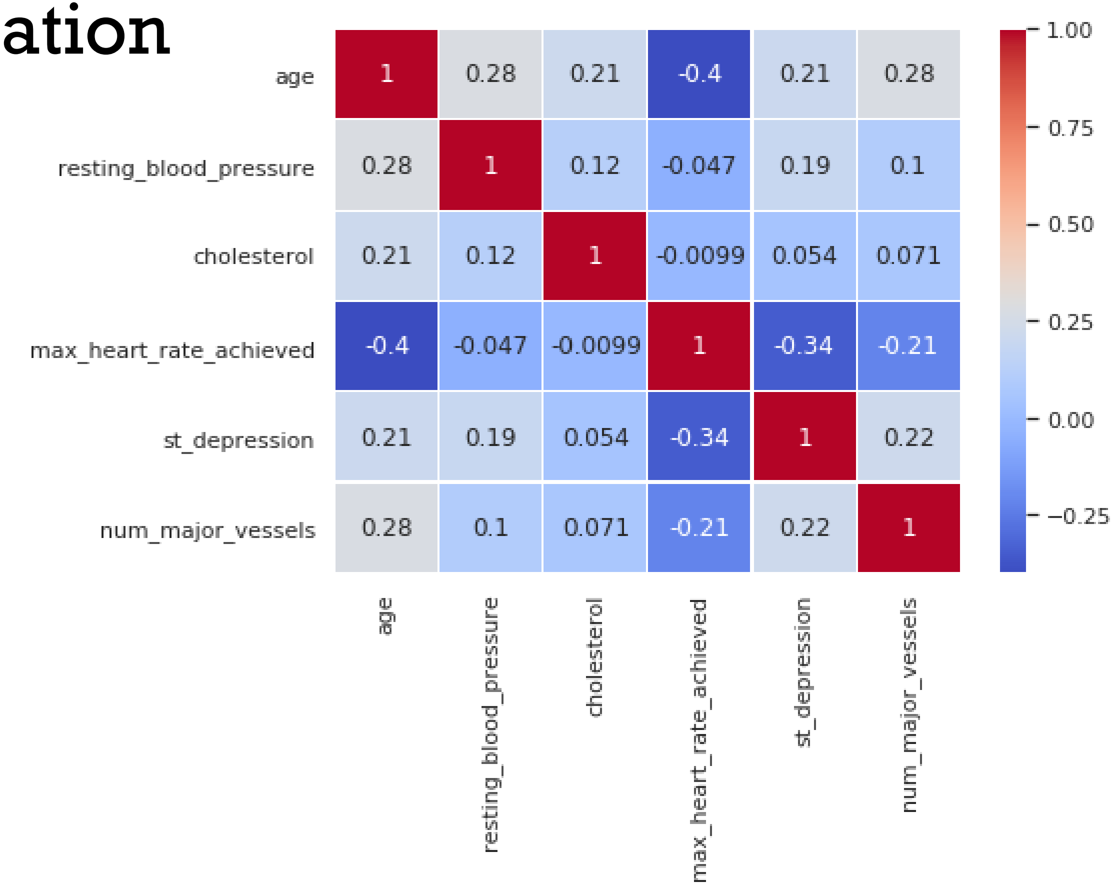
pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Seaborn

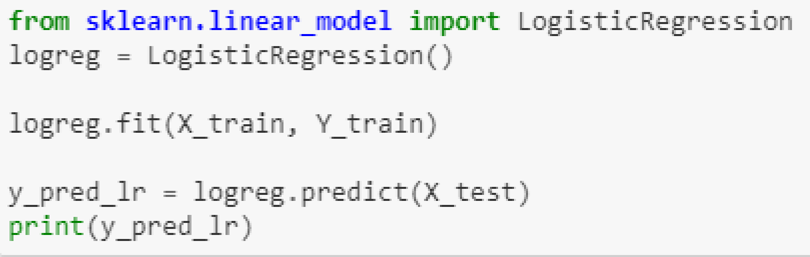
matplotlib

Correlation Plot



	age	resting_blood_pressure	cholesterol	max_heart_rate_achieved	st_depression	num_major_vessels
age	1	0.28	0.21	-0.4	0.21	0.28
resting_blood_pressure	0.28	1	0.12	-0.047	0.19	0.1
cholesterol	0.21	0.12	1	-0.0099	0.054	0.071
max_heart_rate_achieved	-0.4	-0.047	-0.0099	1	-0.34	-0.21
st_depression	0.21	0.19	0.054	-0.34	1	0.22
num_major_vessels	0.28	0.1	0.071	-0.21	0.22	1

Logistic Regression

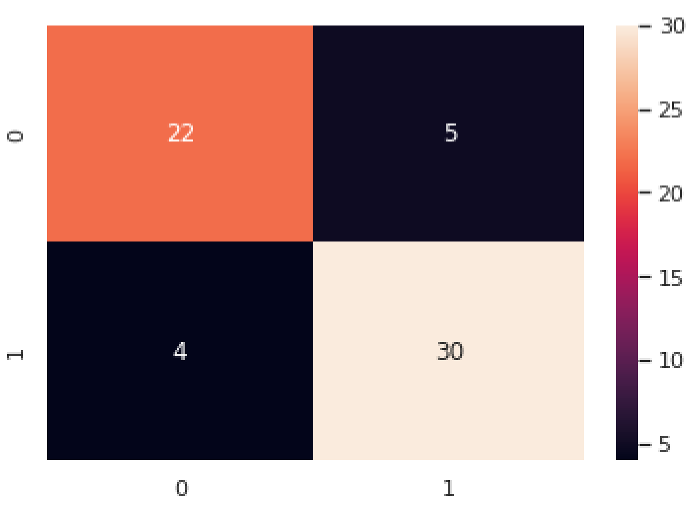


```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()

logreg.fit(X_train, Y_train)

y_pred_lr = logreg.predict(X_test)
print(y_pred_lr)
```

accuracy score : 85.25 %

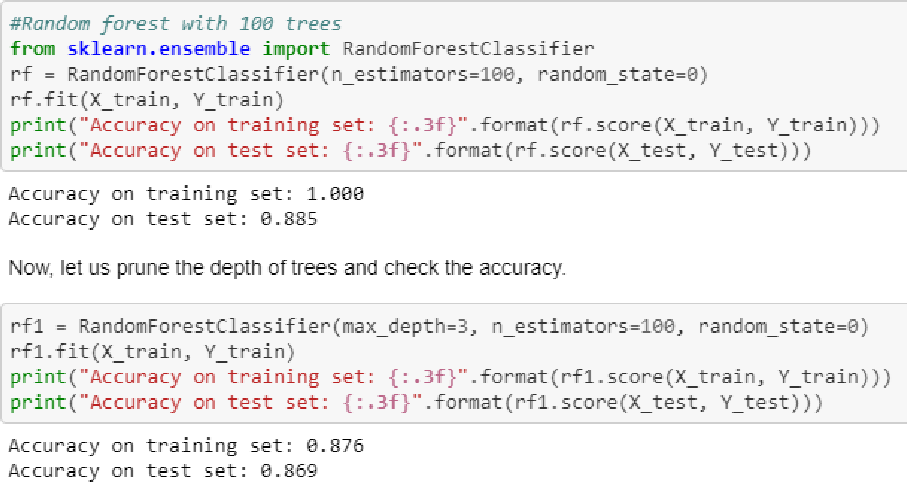


Precision: 0.85

Recall is: 0.88

F-Score: 0.86

Random Forest



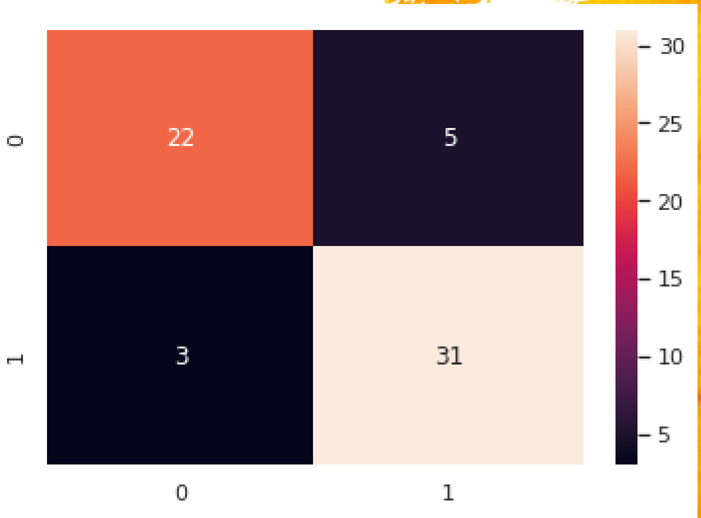
```
#Random forest with 100 trees
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X_train, Y_train)
print("Accuracy on training set: {:.3f}".format(rf.score(X_train, Y_train)))
print("Accuracy on test set: {:.3f}".format(rf.score(X_test, Y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.885
```

Now, let us prune the depth of trees and check the accuracy.

```
rf1 = RandomForestClassifier(max_depth=3, n_estimators=100, random_state=0)
rf1.fit(X_train, Y_train)
print("Accuracy on training set: {:.3f}".format(rf1.score(X_train, Y_train)))
print("Accuracy on test set: {:.3f}".format(rf1.score(X_test, Y_test)))
```

```
Accuracy on training set: 0.876
Accuracy on test set: 0.869
```

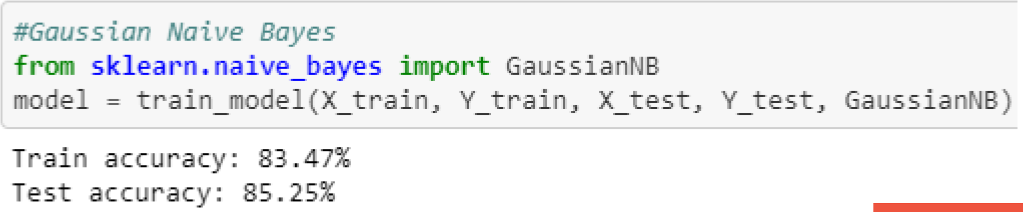


Precision: 0.86

Recall is: 0.91

F-Score: 0.88

Naive Bayes



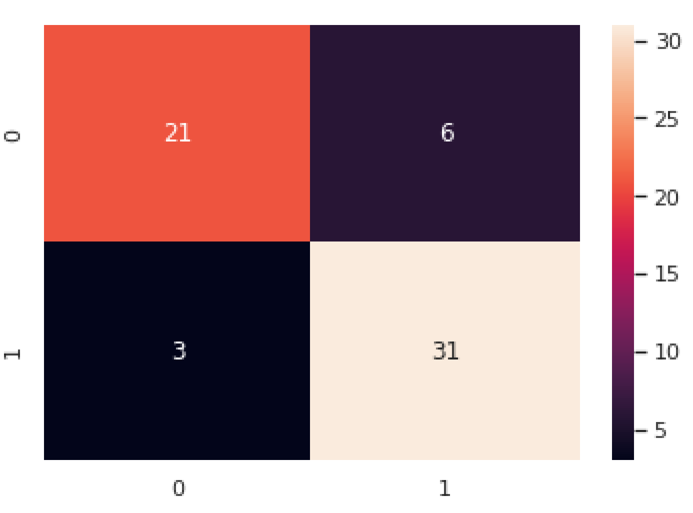
```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
model = train_model(X_train, Y_train, X_test, Y_test, GaussianNB)
```

```
Train accuracy: 83.47%
Test accuracy: 85.25%
```

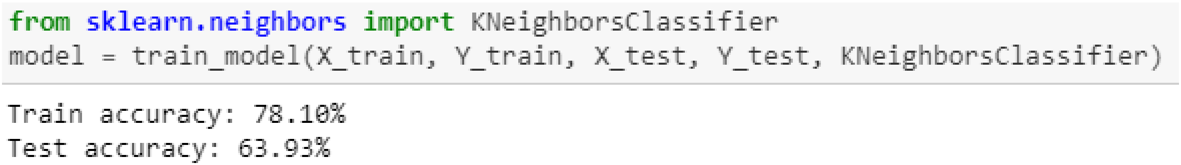
Precision: 0.83

Recall is: 0.91

F-Score: 0.87

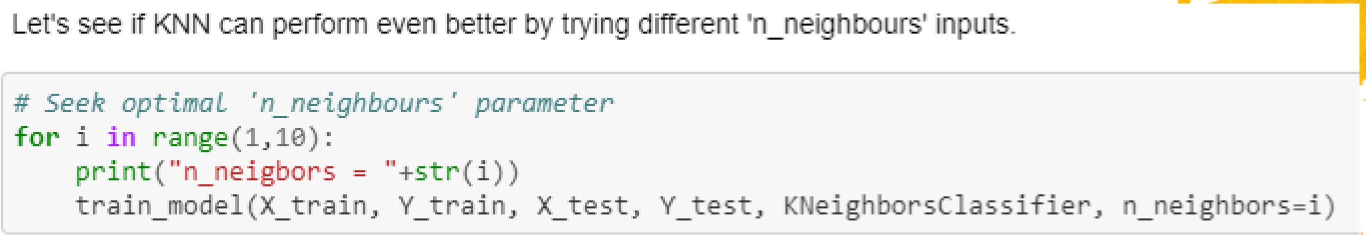


K-Nearest Neighbor



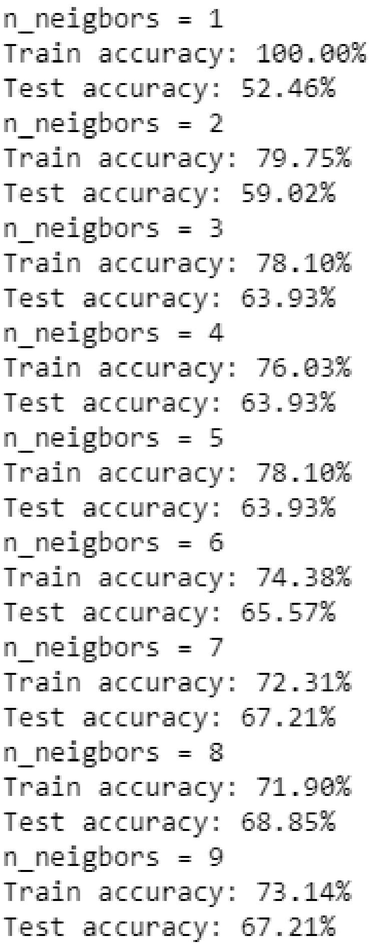
```
from sklearn.neighbors import KNeighborsClassifier
model = train_model(X_train, Y_train, X_test, Y_test, KNeighborsClassifier)
```

```
Train accuracy: 78.10%
Test accuracy: 63.93%
```



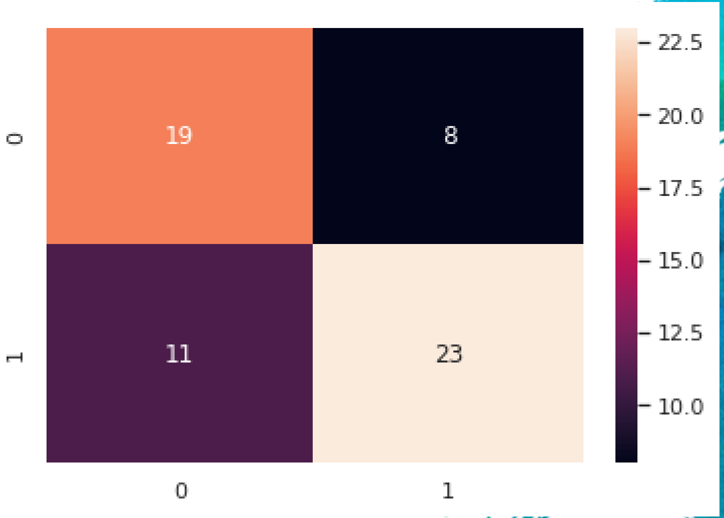
Let's see if KNN can perform even better by trying different 'n_neighbours' inputs.

```
# Seek optimal 'n_neighbours' parameter
for i in range(1,10):
    print("n_neigbors = "+str(i))
    train_model(X_train, Y_train, X_test, Y_test, KNeighborsClassifier, n_neighbors=i)
```



```
n_neigbors = 1
Train accuracy: 100.00%
Test accuracy: 52.46%
n_neigbors = 2
Train accuracy: 79.75%
Test accuracy: 59.02%
n_neigbors = 3
Train accuracy: 78.10%
Test accuracy: 63.93%
n_neigbors = 4
Train accuracy: 76.03%
Test accuracy: 63.93%
n_neigbors = 5
Train accuracy: 78.10%
Test accuracy: 63.93%
n_neigbors = 6
Train accuracy: 74.38%
Test accuracy: 65.57%
n_neigbors = 7
Train accuracy: 72.31%
Test accuracy: 67.21%
n_neigbors = 8
Train accuracy: 71.90%
Test accuracy: 68.85%
n_neigbors = 9
Train accuracy: 73.14%
Test accuracy: 67.21%
```

It turns out that value of n_neighbours (8) is optimal.

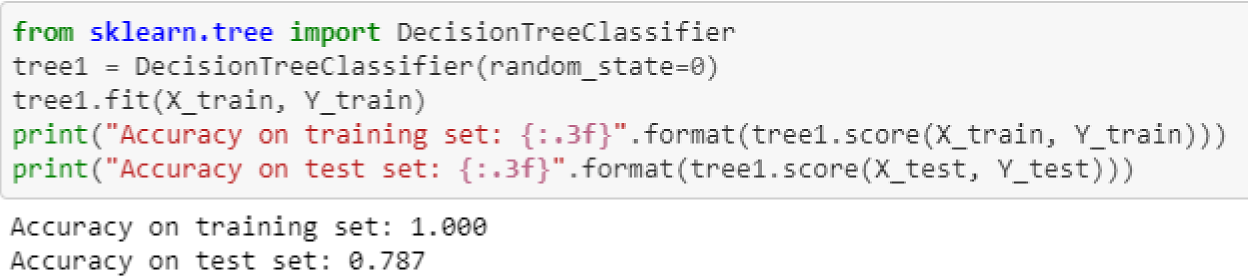


Precision: 0.74

Recall is: 0.67

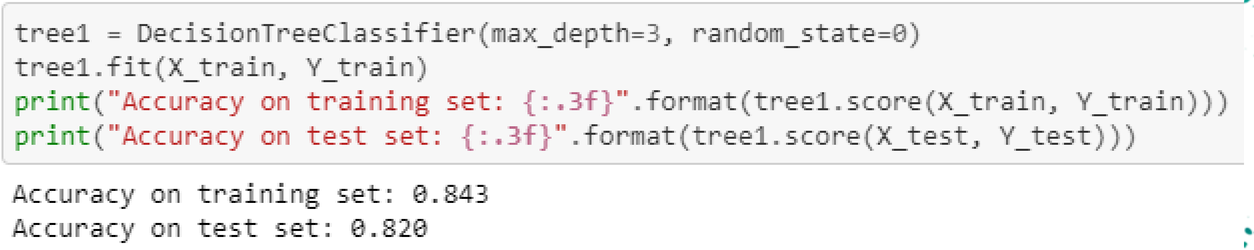
F-Score: 0.70

Decision Tree



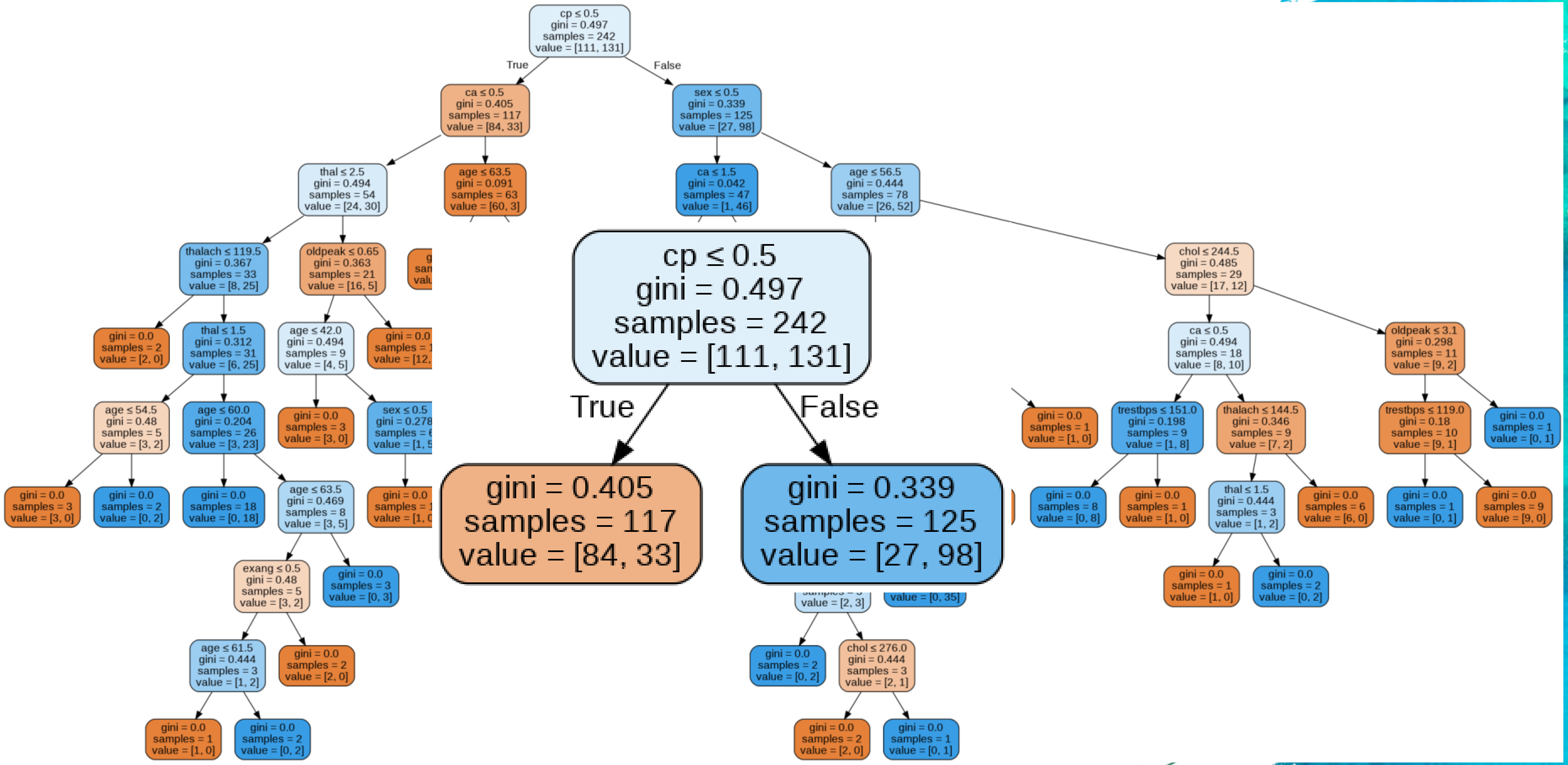
```
from sklearn.tree import DecisionTreeClassifier
tree1 = DecisionTreeClassifier(random_state=0)
tree1.fit(X_train, Y_train)
print("Accuracy on training set: {:.3f}".format(tree1.score(X_train, Y_train)))
print("Accuracy on test set: {:.3f}".format(tree1.score(X_test, Y_test)))
```

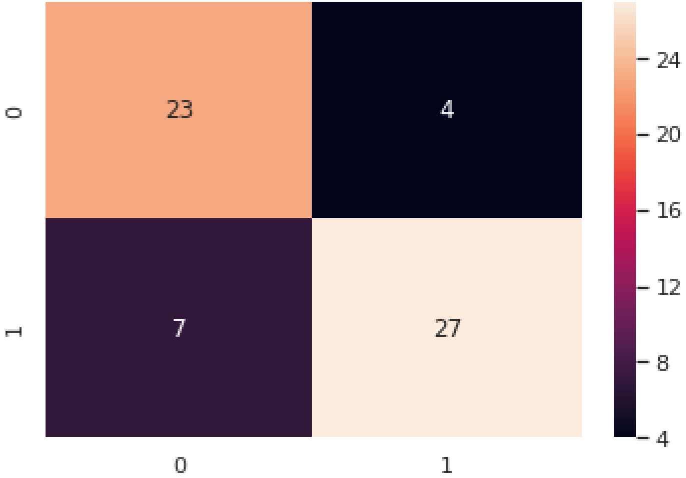
```
Accuracy on training set: 1.000
Accuracy on test set: 0.787
```



```
tree1 = DecisionTreeClassifier(max_depth=3, random_state=0)
tree1.fit(X_train, Y_train)
print("Accuracy on training set: {:.3f}".format(tree1.score(X_train, Y_train)))
print("Accuracy on test set: {:.3f}".format(tree1.score(X_test, Y_test)))
```

```
Accuracy on training set: 0.843
Accuracy on test set: 0.820
```


cp ≤ 0.5 gini = 0.497 samples = 242 value = [111, 131]
True
False
ca ≤ 0.5 gini = 0.405 samples = 117 value = [84, 33]
sex ≤ 0.5 gini = 0.339 samples = 125 value = [27, 98]
thal ≤ 2.5 gini = 0.494 samples = 54 value = [24, 30]
age ≤ 63.5 gini = 0.091 samples = 63 value = [60, 3]
ca ≤ 1.5 gini = 0.042 samples = 47 value = [1, 46]
age ≤ 56.5 gini = 0.444 samples = 78 value = [26, 52]
thalach ≤ 119.5 gini = 0.367 samples = 33 value = [8, 25]
oldpeak ≤ 0.65 gini = 0.363 samples = 21 value = [16, 5]
g san valu
chol ≤ 244.5 gini = 0.485 samples = 29 value = [17, 12]
gini = 0.0 samples = 2 value = [2, 0]
thal ≤ 1.5 gini = 0.312 samples = 31 value = [6, 25]
age ≤ 42.0 gini = 0.494 samples = 9 value = [4, 5]
gini = 0.0 samples = 1 value = [12,
ca ≤ 0.5 gini = 0.494 samples = 18 value = [8, 10]
oldpeak ≤ 3.1 gini = 0.298 samples = 11 value = [9, 2]
age ≤ 54.5 gini = 0.48 samples = 5 value = [3, 2]
age ≤ 60.0 gini = 0.204 samples = 26 value = [3, 23]
gini = 0.0 samples = 3 value = [3, 0]
sex ≤ 0.5 gini = 0.278 samples = 6 value = [1, 5
gini = 0.0 samples = 1 value = [1, 0]
trestbps ≤ 151.0 gini = 0.198 samples = 9 value = [1, 8]
thalach ≤ 144.5 gini = 0.346 samples = 9 value = [7, 2]
trestbps ≤ 119.0 gini = 0.18 samples = 10 value = [9, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 18 value = [0, 18]
age ≤ 63.5 gini = 0.469 samples = 8 value = [3, 5]
gini = 0.0 samples = 1 value = [1, 0
gini = 0.0 samples = 8 value = [0, 8]
gini = 0.0 samples = 1 value = [1, 0]
thal ≤ 1.5 gini = 0.444 samples = 3 value = [1, 2]
gini = 0.0 samples = 6 value = [6, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 9 value = [9, 0]
exang ≤ 0.5 gini = 0.48 samples = 5 value = [3, 2]
gini = 0.0 samples = 3 value = [0, 3]
value = [2, 3]
value = [0, 35]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 2 value = [0, 2]
age ≤ 61.5 gini = 0.444 samples = 3 value = [1, 2]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 2 value = [0, 2]
chol ≤ 276.0 gini = 0.444 samples = 3 value = [2, 1]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 1 value = [0, 1]
cp ≤ 0.5 gini = 0.497 samples = 242 value = [111, 131]
True
False
gini = 0.405 samples = 117 value = [84, 33]
gini = 0.339 samples = 125 value = [27, 98]

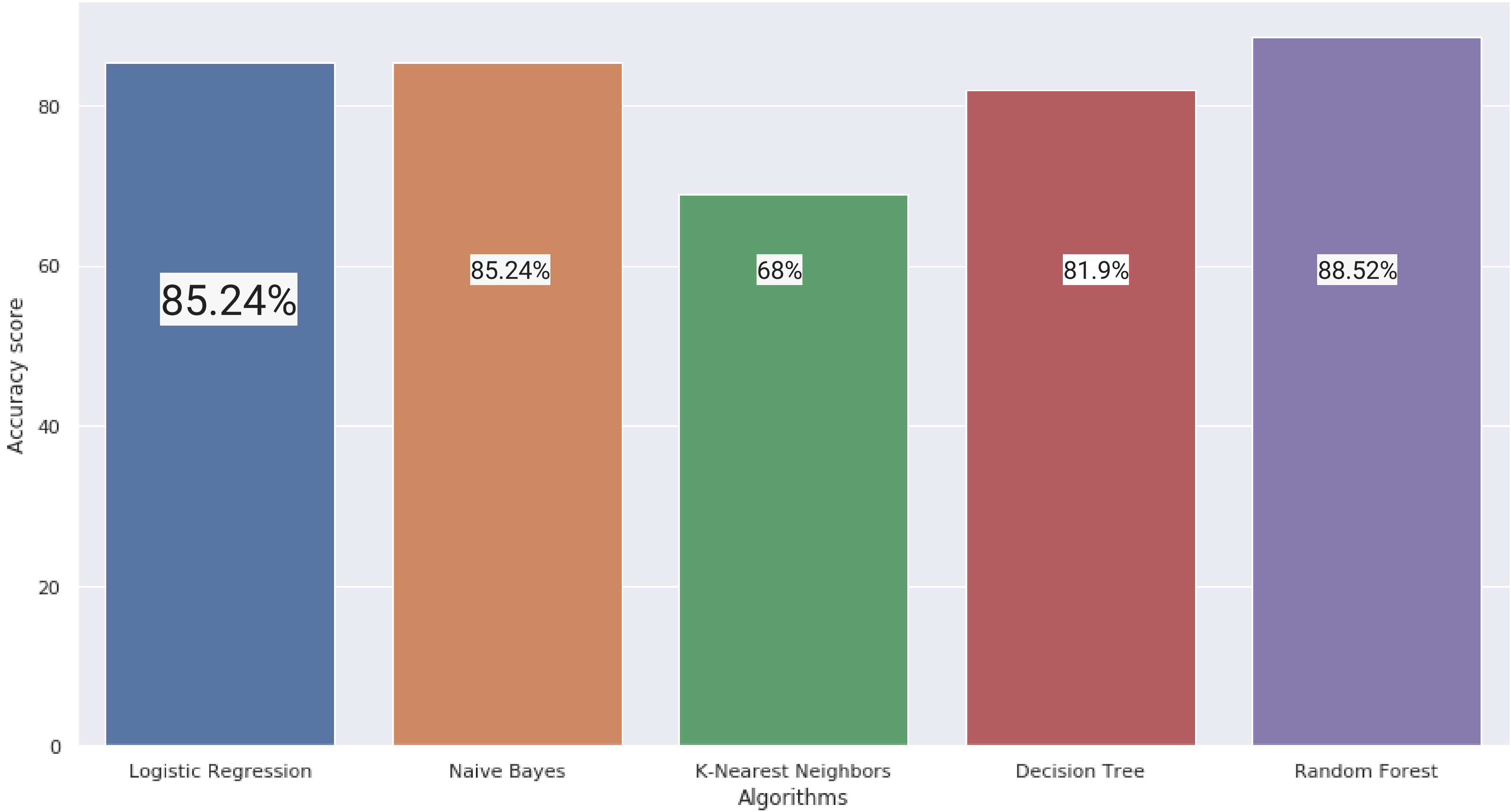


Precision: 0.87

Recall is: 0.79

F-Score: 0.83

Results


Accuracy score
0
20
40
60
80
85.24%
85.24%
68%
81.9%
88.52%
Logistic Regression
Naive Bayes
K-Nearest Neighbors
Decision Tree
Random Forest
Algorithms

Business profit for heart disease prediction:

The business profit for heart disease prediction using machine learning can be significant, and here are some potential benefits for businesses in the healthcare industry:

1. Improved Patient Outcomes: Early detection and treatment of heart disease can significantly improve patient outcomes and reduce the risk of complications. By using machine learning models for heart disease prediction, healthcare providers can identify high-risk patients earlier and intervene promptly, leading to better patient outcomes and increased patient satisfaction.

2 Cost Savings: Machine learning models can help healthcare providers to allocate resources efficiently and effectively. By identifying high-risk patients early, healthcare providers can optimize their care delivery and reduce the need for costly hospitalizations and procedures. This can lead to significant cost savings for both patients and healthcare providers.

Conclusion:

In conclusion, heart disease prediction using machine learning has the potential to revolutionize the field of healthcare. By analyzing large amounts of patient data, machine learning models can accurately predict the likelihood of developing heart disease and identify high-risk patients who may benefit from early intervention.

thank you