EXPERIMENT 3

NAME:PARTH PAREKH **SAP ID:60004200006**

BRANCH:COMPUTER ENGINEERING **BATCH:A1**

Aim:

Implementation of Classification algorithm Using

1. Decision Tree ID3

2. Naïve Bayes algorithm

Theory:

Naïve Bayes Algorithm

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.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for plaing golf.

Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes

	Outlook	Temperature	Humidity	Windy	Play Golf
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, feature matrix and the response vector.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of dependent features. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
- Response vector contains the value of class variable(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent

- equal

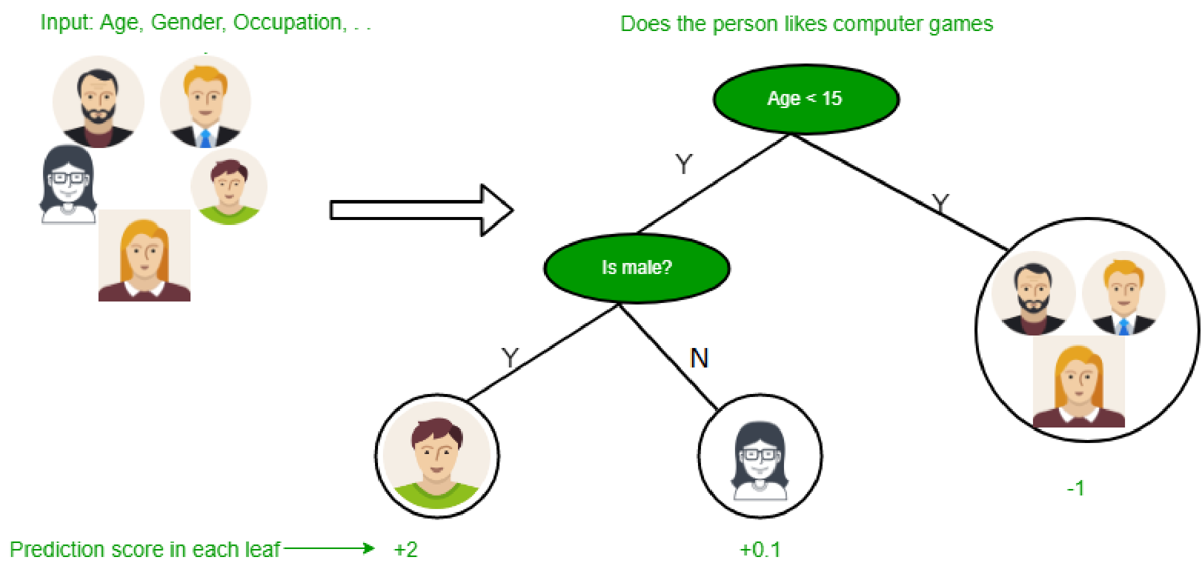
contribution to the outcome.

Decision Tree Algorithm

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

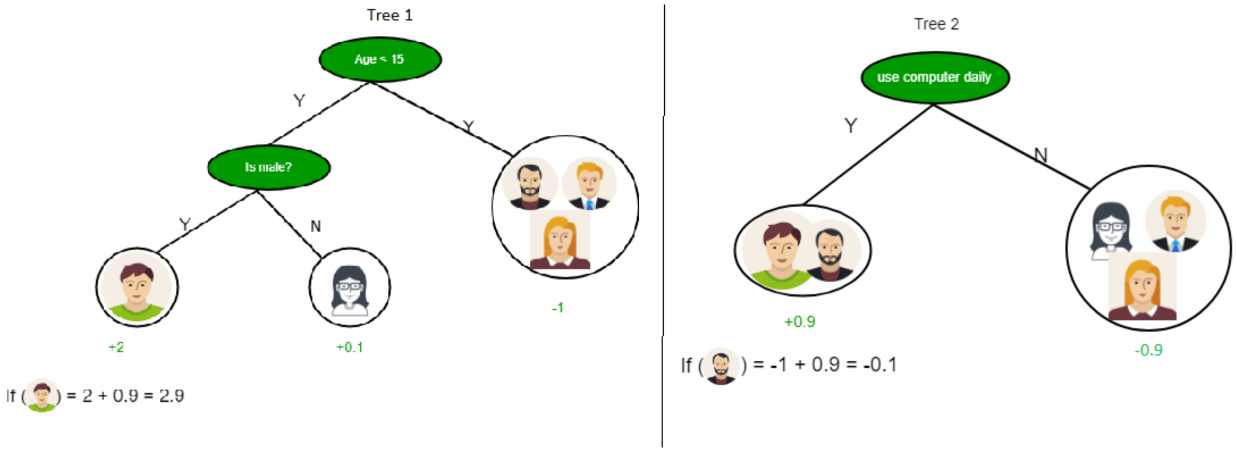
Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

We can represent any boolean function on discrete attributes using the decision tree.



Below are some assumptions that we made while using decision tree:

1. At the beginning, we consider the whole training set as the root.
2. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
3. On the basis of attribute values records are distributed recursively.
4. We use statistical methods for ordering attributes as root or the internal node.



As you can see from the above image that Decision Tree works on the Sum of Product form which is also known as Disjunctive Normal Form. In the above image, we are predicting the use of computer in the daily life of the people.

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

Performance:

Part A & B

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.utils import resample
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accur
acy_score
import sklearn.metrics as metrics

# LabelEncoding Function
encoder = LabelEncoder()
```

```
def label_encoder(df,columns):
  for i in columns:
    df[i] = encoder.fit_transform(df[i])
  df.head()
  return df

# Splitting Data
def split_dataset(X,y):
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
20, random_state = 0)
  return X_train, X_test, y_train, y_test

# Scaling Dataset
sc = StandardScaler()
def scaling(X_train,X_test):
  X_train = sc.fit_transform(X_train)
  X_test = sc.transform(X_test)
  return X_train,X_test

# Model Building
def Classifier(X_train, X_test, y_train, y_test,x):
  if x=="NB":
    model = GaussianNB()
    print("For Naive Bayes: ")
  elif x=="DT":
    model = DecisionTreeClassifier(criterion="entropy",random_state=0)
    print("For Decision Tree: ")
  model.fit(X_train,y_train)
  y_pred = model.predict(X_test)
  print("Classification Report: ")
  print(classification_report(y_test, y_pred))
  conf_matrix = confusion_matrix(y_test, y_pred)
  conf_mat(conf_matrix,x)
  print()
  acc = accuracy_score(y_test, y_pred)
  print()
  print("Accuracy: " + str(acc))
  print()
  return acc,model

# Confusion matrix
def conf_mat(conf_matrix,x):
  print("Confusion Matrix for", x)
  plt.figure(figsize=(24,12))
  plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
```

```
  plt.subplot(2,3,1)
  plt.title("Confusion Matrix")
  sns.heatmap(conf_matrix,annot=True,cmap="Blues",fmt="d",cbar=False)
  plt.show()

# AUROC and Curve
def AUROC(X_test,y_test,model):
  probs = model.predict_proba(X_test)
  preds = probs[:,1]
  fpr, tpr, _ = metrics.roc_curve(y_test, preds)
  roc_auc = metrics.auc(fpr, tpr)
  return fpr,tpr,roc_auc

def roc_auc(fpr_nb,tpr_nb,roc_auc_nb,fpr_dtc,tpr_dtc,roc_auc_dtc):
  fig, ax = plt.subplots(figsize=(7.5, 7.5))

  plt.plot(fpr_nb, tpr_nb, label='ROC Curve Naive (AUC = %0.2f)' % (roc_au
c_nb))
  plt.plot(fpr_dtc, tpr_dtc, label='ROC Curve DTC (AUC = %0.2f)' % (roc_au
c_dtc))
  plt.plot([0, 1], [0, 1], linestyle='--
', color='red', label='Random Classifier')
  plt.plot([0, 0, 1], [0, 1, 1], linestyle=':', color='green', label='Perf
ect Classifier')
  plt.xlim([-0.05, 1.05])
  plt.ylim([-0.05, 1.05])
  plt.xlabel('False positive rate')
  plt.ylabel('True positive rate')
  plt.legend(loc="lower right")
  plt.show()

# Graph Comparison Between Naive Bayes and Decision Tree
def comparisonBar(dataset, acc_nb, acc_dtc):
  accuracy = [acc_nb*100,acc_dtc*100]
  plt.title("For Dataset " + dataset)
  methods = ["Naive Bayes", "Decision Tree" ]
  colors = ["red", "blue"]
  sns.set_style("whitegrid")
  plt.yticks(np.arange(0,100,10))
  plt.ylabel("Accuracy %")
  plt.xlabel("Algorithms")
  sns.barplot(x=methods, y=accuracy, palette=colors)

def cancer():
  from sklearn.datasets import load_breast_cancer
```

```
  breast_cancer = load_breast_cancer()
  df = pd.DataFrame(
    np.c_[breast_cancer.data, breast_cancer.target],
    columns = [list(breast_cancer.feature_names)+ ['target']]
                 )
  print("Overview of Dataset: ")
  print(df.head())
  print()

  print("Seeing dataset stats: ")
  print(df.describe())
  X = df.iloc[:, 0:-1]
  y = df.iloc[:,-1]

  X_train, X_test, y_train, y_test = split_dataset(X,y)
  X_train, X_test = scaling(X_train,X_test)


  acc_nb, nb = Classifier(X_train, X_test, y_train, y_test,"NB")
  acc_dtc, dtc = Classifier(X_train, X_test, y_train, y_test,"DT")

  fpr_nb,tpr_nb,roc_auc_nb = AUROC(X_test,y_test,nb)
  fpr_dtc,tpr_dtc,roc_auc_dtc = AUROC(X_test,y_test,dtc)
  print()

  print("AUROC Curve: ")
  roc_auc(fpr_nb,tpr_nb,roc_auc_nb,fpr_dtc,tpr_dtc,roc_auc_dtc)
  print()

  print("Comparing the two models: ")
  comparisonBar("Breast Cancer Prediction",acc_nb,acc_dtc)
  return float("{:.2f}".format(acc_nb*100)),float("{:.2f}".format(acc_dtc*
100))

def bank_note():
  df = pd.read_csv("BankNote_Authentication.csv")
  print("Overview of Dataset: ")
  print(df.head())
  print()

  print("Checking if null values exist in the Dataset: ")
  print(df.isnull().sum())
  print()

  print("Seeing dataset stats: ")
```

```
  print(df.describe())
  print()

  # Data preprocessing
  X = df.iloc[:, :-1].values
  y = df.iloc[:, -1].values

  X_train, X_test, y_train, y_test = split_dataset(X,y)
  X_train, X_test = scaling(X_train,X_test)

  acc_nb, nb = Classifier(X_train, X_test, y_train, y_test,"NB")
  acc_dtc, dtc = Classifier(X_train, X_test, y_train, y_test,"DT")

  fpr_nb,tpr_nb,roc_auc_nb = AUROC(X_test,y_test,nb)
  fpr_dtc,tpr_dtc,roc_auc_dtc = AUROC(X_test,y_test,dtc)
  print()

  print("AUROC Curve: ")
  roc_auc(fpr_nb,tpr_nb,roc_auc_nb,fpr_dtc,tpr_dtc,roc_auc_dtc)
  print()

  print("Comparing the two models: ")
  comparisonBar("Bank Note Authentication",acc_nb,acc_dtc)
  return float("{:.2f}".format(acc_nb*100)),float("{:.2f}".format(acc_dtc*
100))

def social_network():
  df = pd.read_csv("Social_Network_Ads.csv")
  print("Overview of Dataset: ")
  print(df.head())
  print()

  print("Checking if null values exist in the Dataset: ")
  print(df.isnull().sum())
  print()

  print("Seeing dataset stats: ")
  print(df.describe())
  print()
  X = df.iloc[:, [2, 3]].values
  y = df.iloc[:, 4].values


  X_train, X_test, y_train, y_test = split_dataset(X,y)
  X_train, X_test = scaling(X_train,X_test)
```

```
  acc_nb, nb = Classifier(X_train, X_test, y_train, y_test,"NB")
  acc_dtc, dtc = Classifier(X_train, X_test, y_train, y_test,"DT")

  fpr_nb,tpr_nb,roc_auc_nb = AUROC(X_test,y_test,nb)
  fpr_dtc,tpr_dtc,roc_auc_dtc = AUROC(X_test,y_test,dtc)
  print()

  print("AUROC Curve: ")
  roc_auc(fpr_nb,tpr_nb,roc_auc_nb,fpr_dtc,tpr_dtc,roc_auc_dtc)
  print()

  print("Comparing the two models: ")
  comparisonBar("Social Network Ads",acc_nb,acc_dtc)
  return float("{:.2f}".format(acc_nb*100)),float("{:.2f}".format(acc_dtc*
100))

def heart():
  df = pd.read_csv("heart.csv")
  print("Overview of Dataset: ")
  print(df.head())
  print()

  print("Checking if null values exist in the Dataset: ")
  print(df.isnull().sum())
  print()

  # Data preparation
  bins = [25,35,45,55,65,75,85]
  labels = ['25-35','35-45','45-55','55-65','65-75', '75-85']
  df['age'] = pd.cut(df['age'], bins=bins, labels=labels)

  # Getting categorical values for non-categorical features like Age
  from sklearn.preprocessing import LabelEncoder
  encoder = LabelEncoder()
  column = ['age']
  df[column] = encoder.fit_transform(df[column])
  print("Seeing dataset stats: ")
  print(df.describe())
  print()
  X = df.iloc[:, :-1].values
  y = df.iloc[:, -1].values
  X_train, X_test, y_train, y_test = split_dataset(X,y)
  X_train, X_test = scaling(X_train,X_test)
```

```
  acc_nb, nb = Classifier(X_train, X_test, y_train, y_test,"NB")
  acc_dtc, dtc = Classifier(X_train, X_test, y_train, y_test,"DT")

  fpr_nb,tpr_nb,roc_auc_nb = AUROC(X_test,y_test,nb)
  fpr_dtc,tpr_dtc,roc_auc_dtc = AUROC(X_test,y_test,dtc)
  print()

  print("AUROC Curve: ")
  roc_auc(fpr_nb,tpr_nb,roc_auc_nb,fpr_dtc,tpr_dtc,roc_auc_dtc)
  print()

  print("Comparing the two models: ")
  comparisonBar("Heart Disease",acc_nb,acc_dtc)
  return float("{:.2f}".format(acc_nb*100)),float("{:.2f}".format(acc_dtc*
100))

def iris():
  df = pd.read_csv("Iris.csv")
  print("Overview of Dataset: ")
  print(df.head())
  print()

  print("Checking if null values exist in the Dataset: ")
  print(df.isnull().sum())
  print()
  print("Seeing dataset stats: ")
  print(df.describe())
  X = df.iloc[:, :-1].values
  y = df.iloc[:, -1].values
  X_train, X_test, y_train, y_test = split_dataset(X,y)
  X_train, X_test = scaling(X_train,X_test)

  acc_nb, nb = Classifier(X_train, X_test, y_train, y_test,"NB")
  acc_dtc, dtc = Classifier(X_train, X_test, y_train, y_test,"DT")

  print("Comparing the two models: ")
  comparisonBar("Social Network Ads",acc_nb,acc_dtc)
  return float("{:.2f}".format(acc_nb*100)),float("{:.2f}".format(acc_dtc*
100))

def plotComparison(a,b,c,d,e,f,g,h,i,j):
  labels = ['Heart Disease', 'Iris', 'Social Ads', 'Bank Note', 'Breast Ca
ncer']
  nb_means = [a, b, c, d, e]
  dtc_means = [f, g, h, i, j]
```

```
  x = np.arange(len(labels))
  width = 0.2
  fig, ax = plt.subplots()
  rects1 = ax.bar(x - width/2, nb_means, width, label='Naive Bayes')
  rects2 = ax.bar(x + width/2, dtc_means, width, label='Decision Tree')
  ax.set_ylabel('Accuracy')
  ax.set_title('Comparison Graph(Blue=Naive Bayes,Orange=Decision Tree)')
  ax.set_xticks(x)
  ax.set_xticklabels(labels)

  fig.tight_layout()

  plt.show()

def main():
    print("Heart Disease Prediction")
    print()
    nb_heart, dtc_heart = heart()
    print()

    print("Iris Prediction")
    print()
    nb_iris, dtc_iris = iris()
    print()

    print("Social Network Ads Prediction")
    print()
    nb_social_ad, dtc_social_ad = social_network()
    print()

    print("Bank Note Authentication Prediction")
    print()
    nb_bank_note, dtc_bank_note = bank_note()

    print("Breast Cancer Prediction")
    print()
    nb_cancer, dtc_cancer = cancer()
    print()

    print()
    plotComparison(nb_heart,nb_iris,nb_social_ad,nb_bank_note,nb_cancer,dt
c_heart,dtc_social_ad,dtc_iris,dtc_bank_note,dtc_cancer)

main()
```

Part C

```
import pandas as pd
from sklearn.model_selection import KFold
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier,BaggingClassifier,AdaBoostCl
assifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

df = pd.read_csv("BankNote_Authentication.csv")

X = df.iloc[:,:-1]
y = df.iloc[:,-1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
 random_state = 1)
# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
classifier_dtc = DecisionTreeClassifier(criterion='entropy', random_state=
0)
classifier_dtc.fit(X_train, y_train)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier_naive = GaussianNB()
classifier_naive.fit(X_train, y_train)

k = 10
kf = KFold(n_splits=k, random_state=0, shuffle=True)
nb = GaussianNB()
dtc = DecisionTreeClassifier()
```

```
acc_score_nb = []
acc_score_dtc = []

for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    nb.fit(X_train,y_train)
    dtc.fit(X_train,y_train)
    pred_values_nb = nb.predict(X_test)
    pred_values_dtc = dtc.predict(X_test)

    acc_nb = accuracy_score(pred_values_nb , y_test)
    acc_dtc = accuracy_score(pred_values_dtc , y_test)
    acc_score_nb.append(acc_nb)
    acc_score_dtc.append(acc_dtc)

avg_acc_score_nb = sum(acc_score_nb)/k
avg_acc_score_dtc = sum(acc_score_dtc)/k

print("For Naive Bayes: ")
print('accuracy of each fold - {}'.format(acc_score_nb))
print('Avg accuracy : {}'.format(avg_acc_score_nb))

print("For Decision Tree: ")
print('accuracy of each fold - {}'.format(acc_score_dtc))
print('Avg accuracy : {}'.format(avg_acc_score_dtc))
print("Accuracy after KFold Cross Validation for Naive Bayes : %0.4f"%((av
g_acc_score_nb)*100))
print("Accuracy after KFold Cross Validation for Decision Tree : %0.4f"%((
avg_acc_score_dtc)*100))

#Accuracy of Naive Bayes
print("Naive Bayes Test Accuracy {:.2f}%".format(classifier_naive.score(X_
test, y_test)*100))

#Accuracy of Decision tree
print("Decision Tree Test Accuracy {:.2f}%".format(classifier_dtc.score(X_
test, y_test)*100))

methods = ["Naive Bayes", "Naive Bayes After K folds","Decision Tree","Dec
ision Tree After K folds" ]
accuracy = [classifier_naive.score(X_test, y_test)*100,(avg_acc_score_nb)*
100,classifier_dtc.score(X_test, y_test)*100,(avg_acc_score_dtc)*100]
```

```
colors = ["orange", "blue"]
sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,100,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=methods, y=accuracy, palette=colors)
plt.show()

bg_dtc=BaggingClassifier(dtc,max_samples=0.5,max_features=1.0,n_estimators
=20)
bg_dtc.fit(X_train,y_train.values.ravel())

print("Bagging Ensemble Model for Decision Tree Classifier: %0.4f"%(bg_dtc
.score(X_test,y_test.values.ravel())*100))
bg_nb=BaggingClassifier(nb,max_samples=0.5,max_features=1.0,n_estimators=2
0)
bg_nb.fit(X_train,y_train.values.ravel())

print("Bagging Ensemble Model for Naive Bayes Classifier: %0.4f"%(bg_nb.sc
ore(X_test,y_test.values.ravel())*100))
ada_dtc=AdaBoostClassifier(dtc,n_estimators=10,learning_rate=1)
ada_dtc.fit(X_train,y_train.values.ravel())

print("AdaBoost Ensemble Model for Decison Tree Classifier: %0.4f"%(ada_dt
c.score(X_test,y_test.values.ravel())*100))
ada_nb=AdaBoostClassifier(nb,n_estimators=1,learning_rate=1)
ada_nb.fit(X_train,y_train.values.ravel())

print("AdaBoost Ensemble Model for Naive BayesClassifier: %0.4f"%(ada_nb.s
core(X_test,y_test.values.ravel())*100))
VC = VotingClassifier(estimators=[('lr',nb),('dt',dtc)],voting='hard')
VC.fit(X_train,y_train.values.ravel())

print("VotingClassifier Ensemble Model: %0.4f"%(VC.score(X_test,y_test.val
ues.ravel())*100))

methods = ["Naive Bayes Bagging", "Naive Bayes Adaboost","Decision Tree Ba
gging","Decision Tree Adaboost","Voting Classifier" ]
accuracy = [(bg_nb.score(X_test,y_test.values.ravel())*100),(ada_nb.score(
X_test,y_test.values.ravel())*100),(bg_dtc.score(X_test,y_test.values.rave
l())*100),(bg_nb.score(X_test,y_test.values.ravel())*100),(VC.score(X_test
,y_test.values.ravel())*100)]
colors = ["orange", "blue"]
sns.set_style("whitegrid")
```

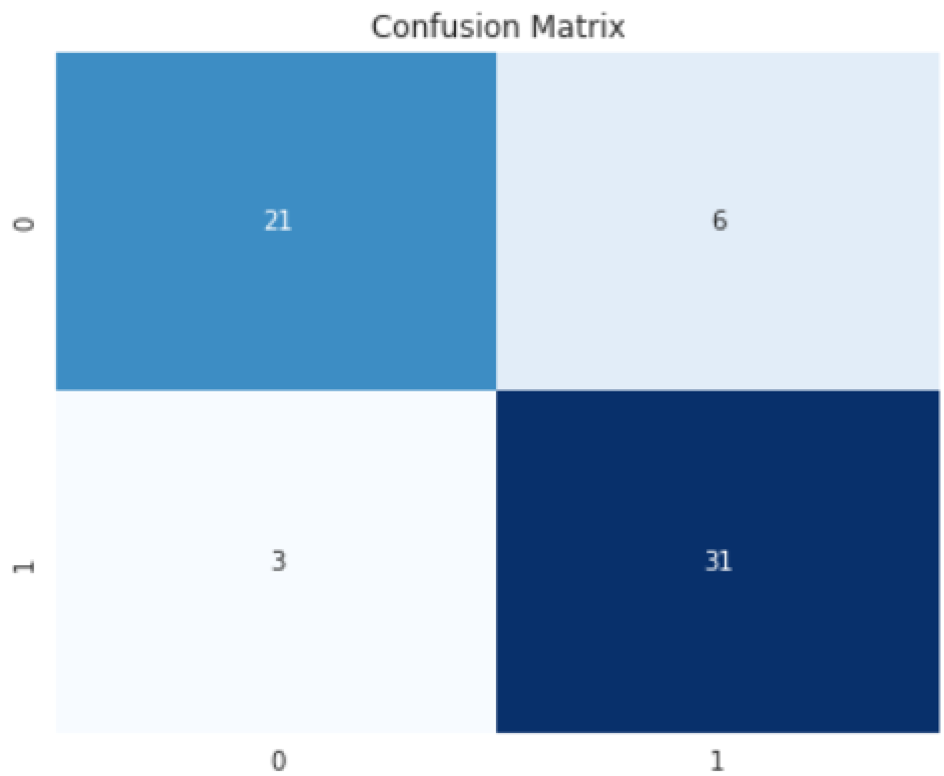
```
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,100,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=methods, y=accuracy, palette=colors)
plt.show()
```

Output

Part A & B

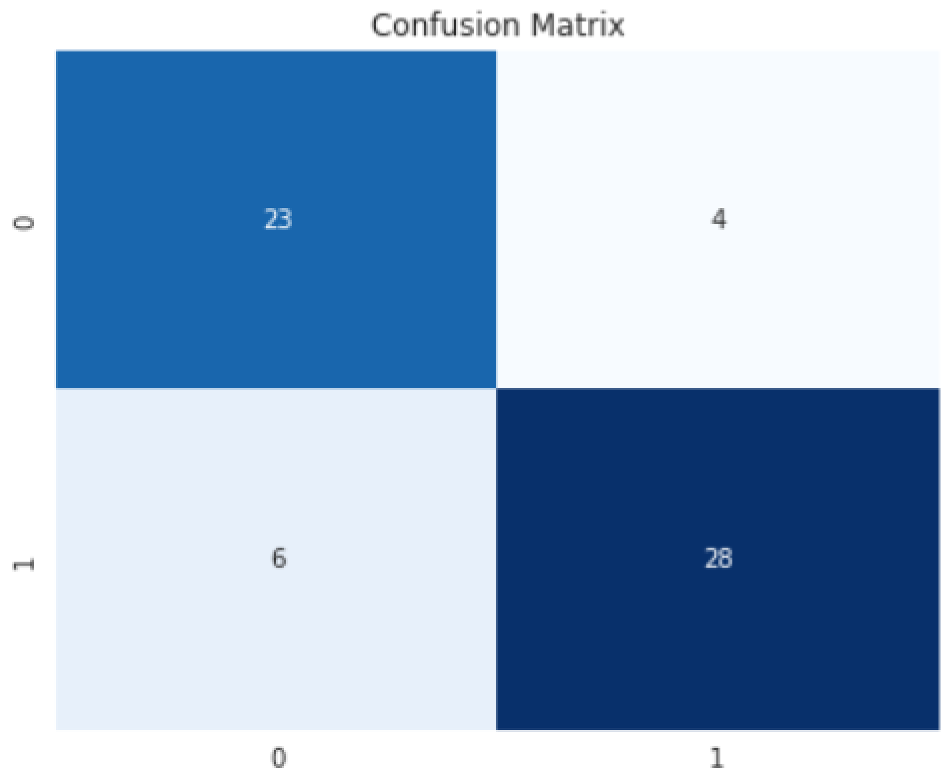
```
Heart Disease Prediction

Overview of Dataset:
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target
0   63    1   3       145   233    1  ...      0      2.3      0   0     1       1
1   37    1   2       130   250    0  ...      0      3.5      0   0     2       1
2   41    0   1       130   204    0  ...      0      1.4      2   0     2       1
3   56    1   1       120   236    0  ...      0      0.8      2   0     2       1
4   57    0   0       120   354    0  ...      1      0.6      2   0     2       1

[5 rows x 14 columns]

Checking if null values exist in the Dataset:
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
```

```
Seeing dataset stats:
              age         sex          cp  ...          ca        thal      target
count  303.000000  303.000000  303.000000  ...  303.000000  303.000000  303.000000
mean     2.379538    0.683168    0.966997  ...    0.729373    2.313531    0.544554
std      0.998928    0.466011    1.032052  ...    1.022606    0.612277    0.498835
min      0.000000    0.000000    0.000000  ...    0.000000    0.000000    0.000000
25%      2.000000    0.000000    0.000000  ...    0.000000    2.000000    0.000000
50%      2.000000    1.000000    1.000000  ...    0.000000    2.000000    1.000000
75%      3.000000    1.000000    2.000000  ...    1.000000    3.000000    1.000000
max      5.000000    1.000000    3.000000  ...    4.000000    3.000000    1.000000

[8 rows x 14 columns]

For Naive Bayes:
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.78      0.82        27
           1       0.84      0.91      0.87        34

    accuracy                           0.85        61
   macro avg       0.86      0.84      0.85        61
weighted avg       0.85      0.85      0.85        61

Confusion Matrix for NB
```



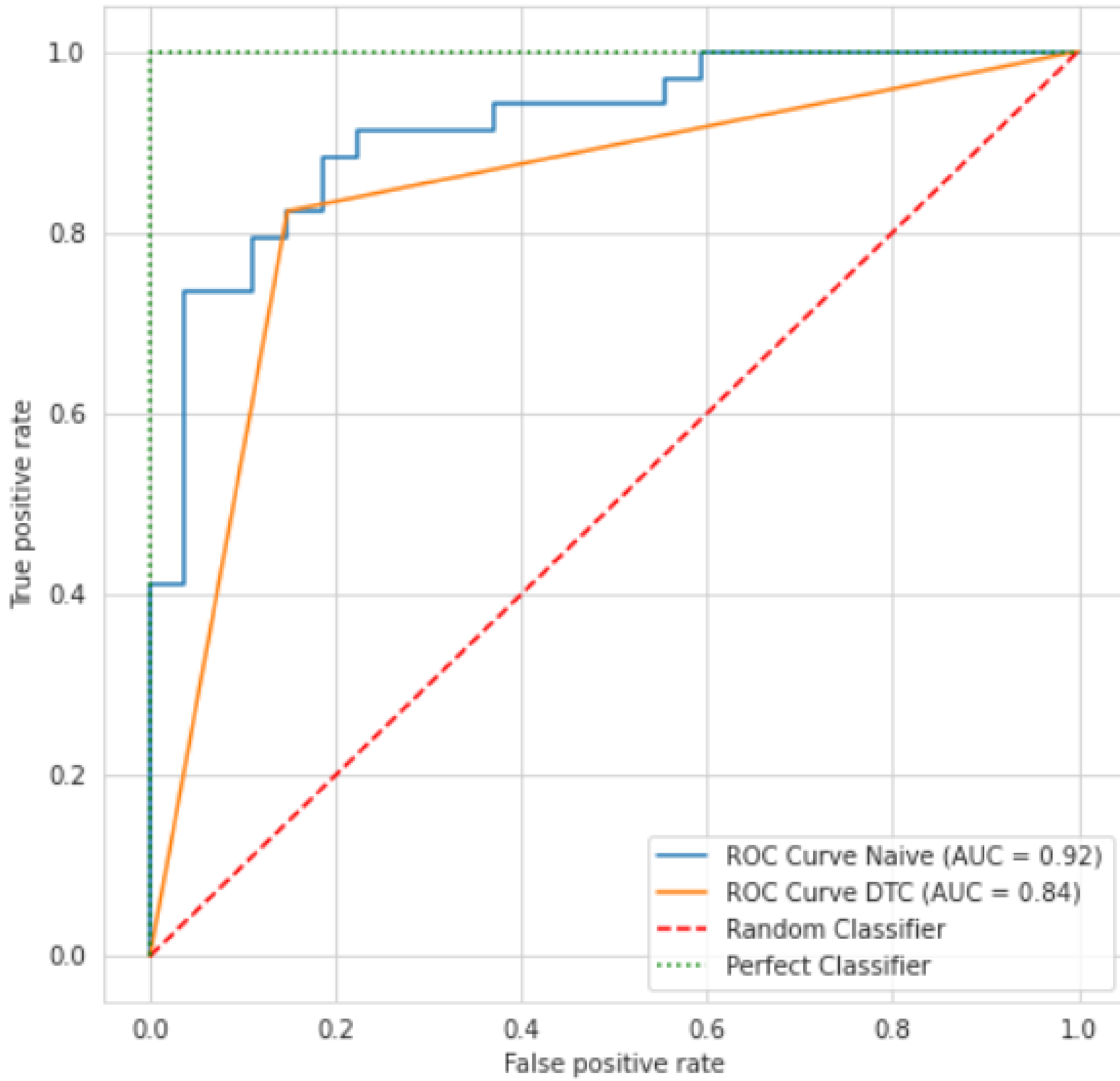

```
Accuracy: 0.8524590163934426

For Decision Tree:
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.85      0.82        27
           1       0.88      0.82      0.85        34

    accuracy                           0.84        61
   macro avg       0.83      0.84      0.83        61
weighted avg       0.84      0.84      0.84        61
```

Confusion Matrix for DT



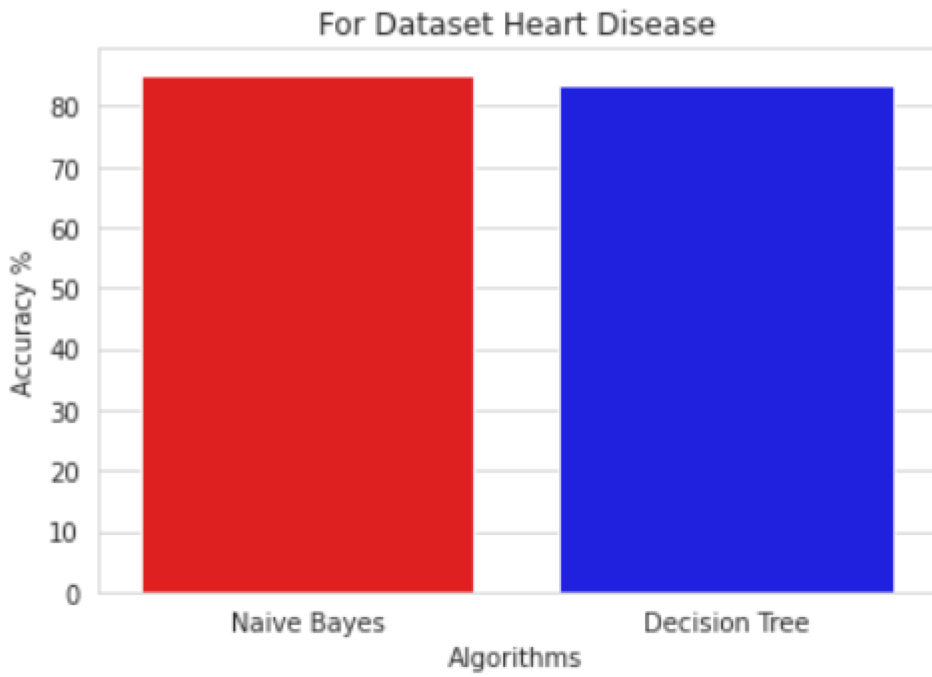
Accuracy: 0.8360655737704918

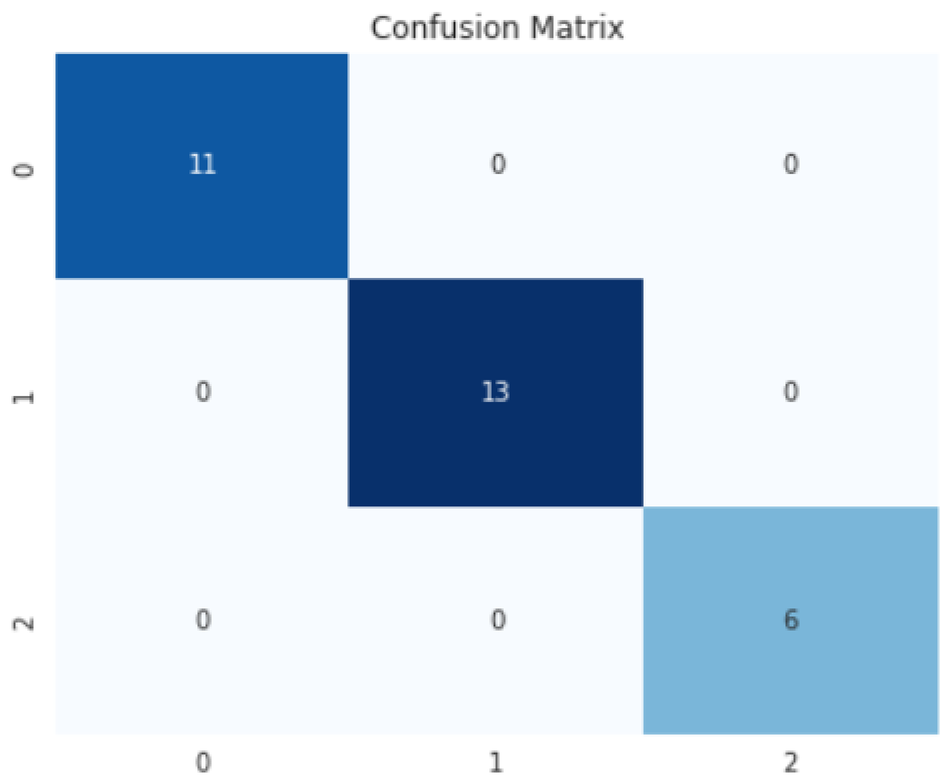
AUROC Curve:


1.0
0.8
0.6
0.4
0.2
0.0
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
ROC Curve Naive (AUC = 0.92)
ROC Curve DTC (AUC = 0.84)
Random Classifier
Perfect Classifier

```
Comparing the two models:

Iris Prediction

Overview of Dataset:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa

Checking if null values exist in the Dataset:
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64

Seeing dataset stats:
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
```

```
For Naive Bayes:
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

Confusion Matrix for NB
```



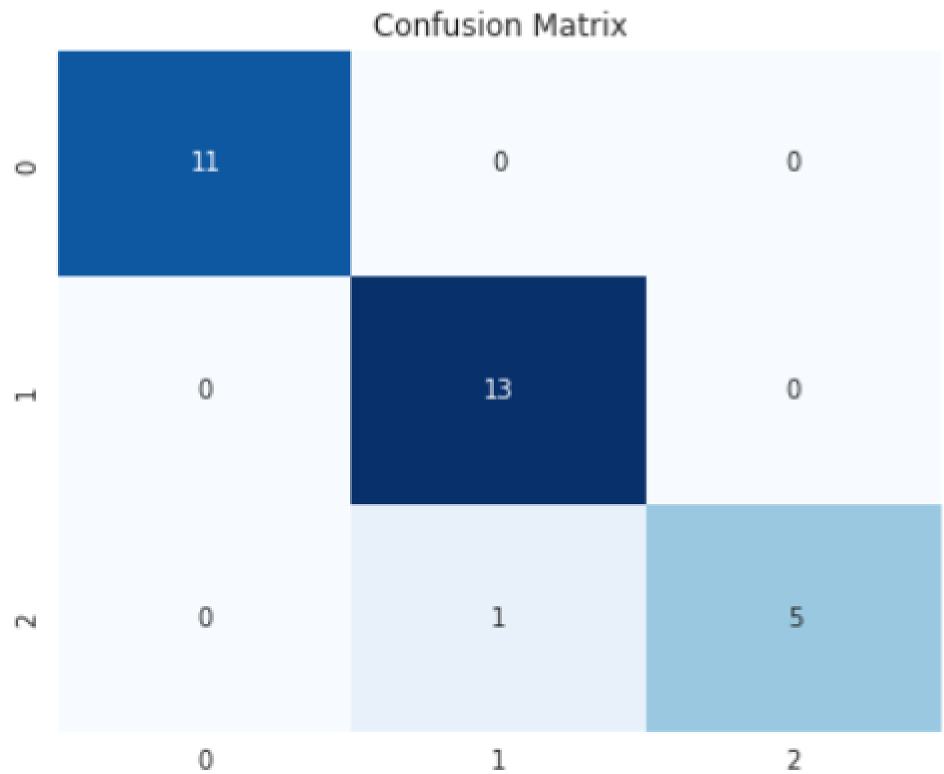



```
Accuracy: 1.0
```

For Decision Tree:
Classification Report:

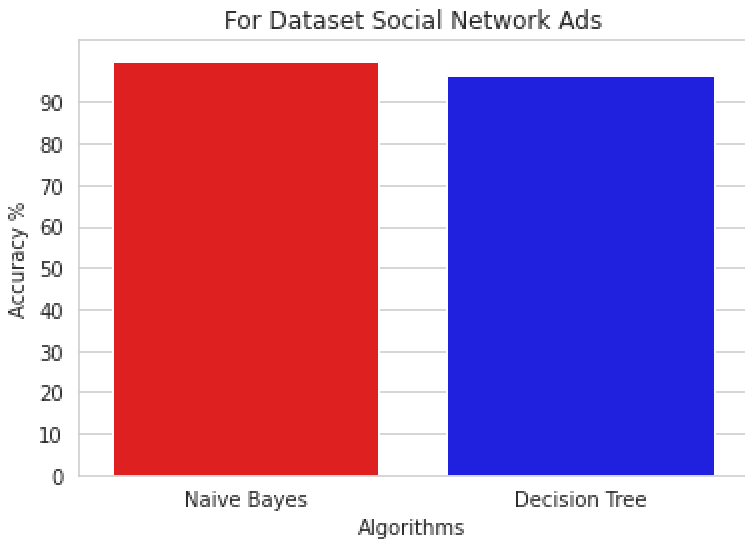
	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	0.93	1.00	0.96	13
Iris-virginica	1.00	0.83	0.91	6
accuracy			0.97	30
macro avg	0.98	0.94	0.96	30
weighted avg	0.97	0.97	0.97	30

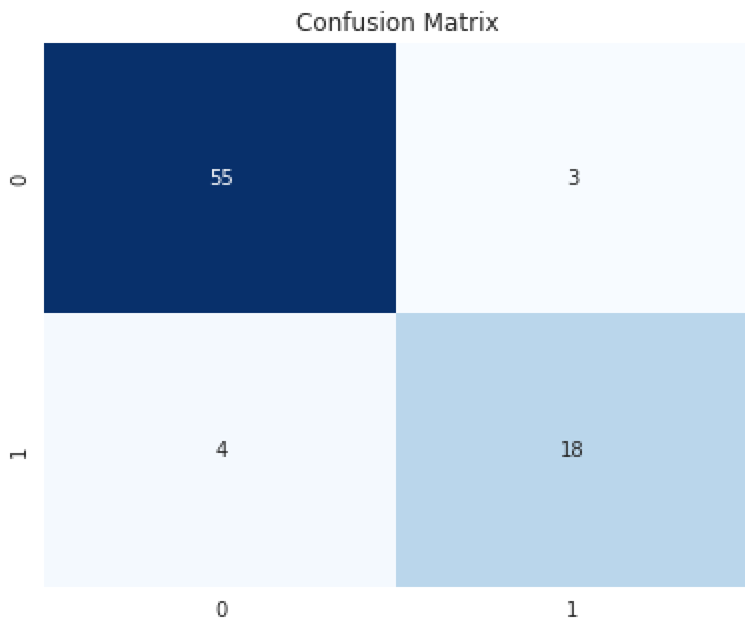
Confusion Matrix for DT



```
Accuracy: 0.9666666666666667

Comparing the two models:

Social Network Ads Prediction

Overview of Dataset:
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0

Checking if null values exist in the Dataset:
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

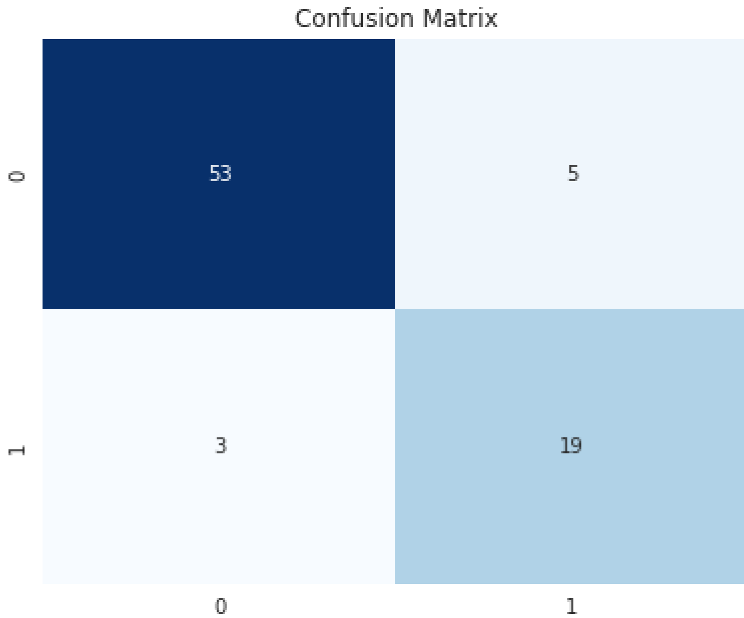
```
Seeing dataset stats:
            User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000

For Naive Bayes:
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94        58
           1       0.86      0.82      0.84        22

    accuracy                           0.91        80
   macro avg       0.89      0.88      0.89        80
weighted avg       0.91      0.91      0.91        80

Confusion Matrix for NB
```





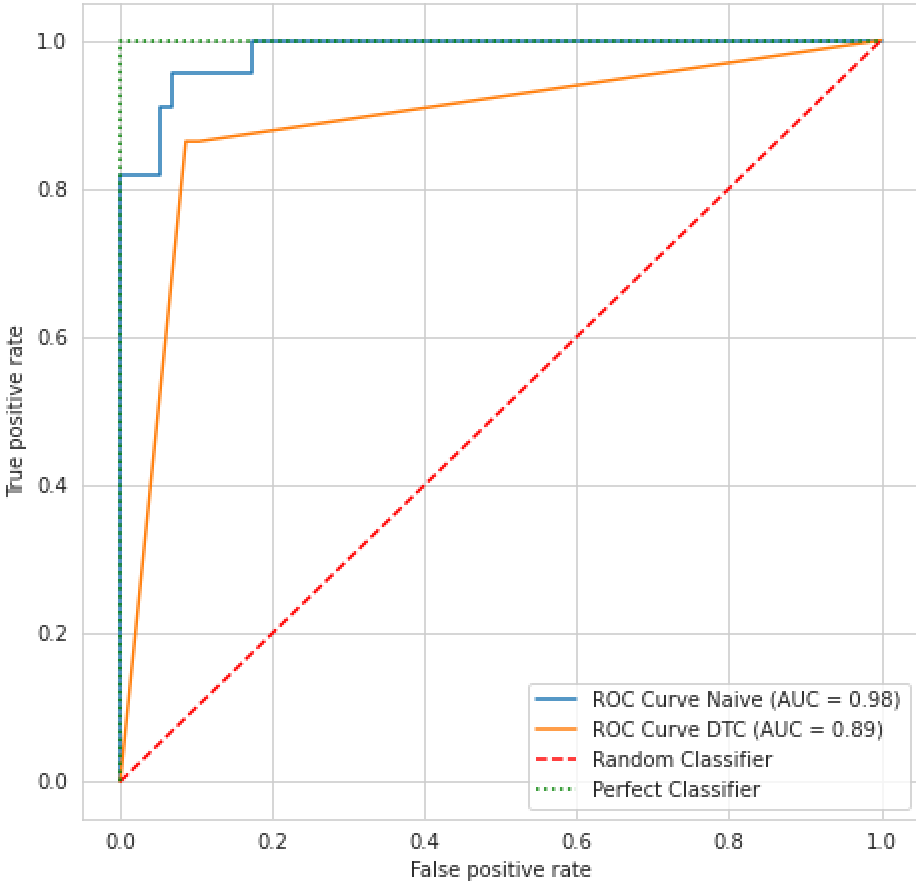

```
Accuracy: 0.9125
```

```
For Decision Tree:
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        58
           1       0.79      0.86      0.83        22

    accuracy                           0.90        80
   macro avg       0.87      0.89      0.88        80
weighted avg       0.90      0.90      0.90        80

Confusion Matrix for DT
```



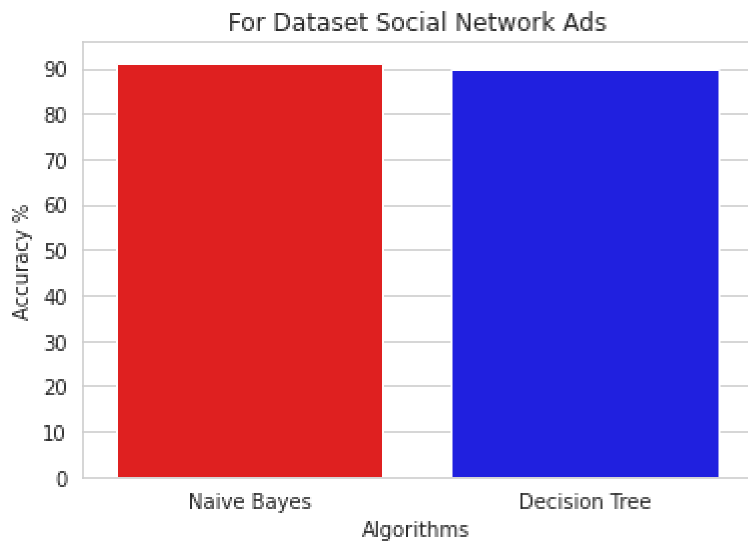

```
Accuracy: 0.9
```

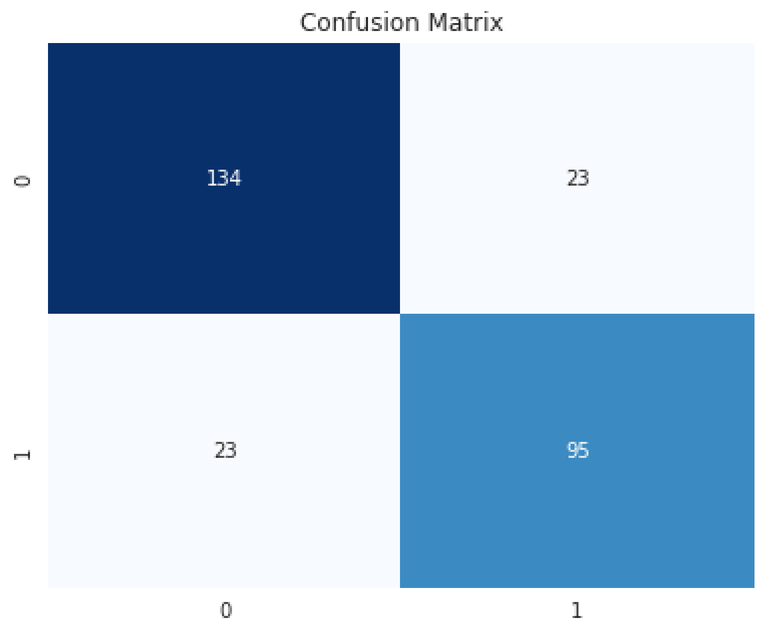
```
AUROC Curve:
```




```
Comparing the two models:

Bank Note Authentication Prediction

Overview of Dataset:
   variance  skewness  curtosis  entropy  class
0   3.62160    8.6661   -2.8073 -0.44699      0
1   4.54590    8.1674   -2.4586 -1.46210      0
2   3.86600   -2.6383    1.9242  0.10645      0
3   3.45660    9.5228   -4.0112 -3.59440      0
4   0.32924   -4.4552    4.5718 -0.98880      0
```

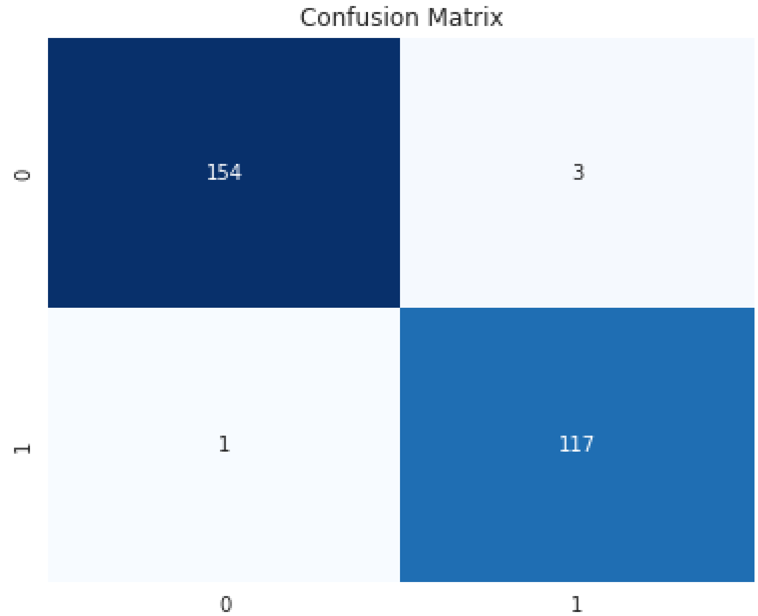
```
Checking if null values exist in the Dataset:
variance    0
skewness    0
curtosis    0
entropy     0
class       0
dtype: int64

Seeing dataset stats:
          variance     skewness     curtosis      entropy        class
count  1372.000000  1372.000000  1372.000000  1372.000000  1372.000000
mean      0.433735     1.922353     1.397627    -1.191657     0.444606
std       2.842763     5.869047     4.310030     2.101013     0.497103
min      -7.042100   -13.773100    -5.286100    -8.548200     0.000000
25%      -1.773000    -1.708200    -1.574975    -2.413450     0.000000
50%       0.496180     2.319650     0.616630    -0.586650     0.000000
75%       2.821475     6.814625     3.179250     0.394810     1.000000
max       6.824800    12.951600    17.927400     2.449500     1.000000

For Naive Bayes:
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       157
           1       0.81      0.81      0.81       118

    accuracy                           0.83       275
   macro avg       0.83      0.83      0.83       275
weighted avg       0.83      0.83      0.83       275

Confusion Matrix for NB
```





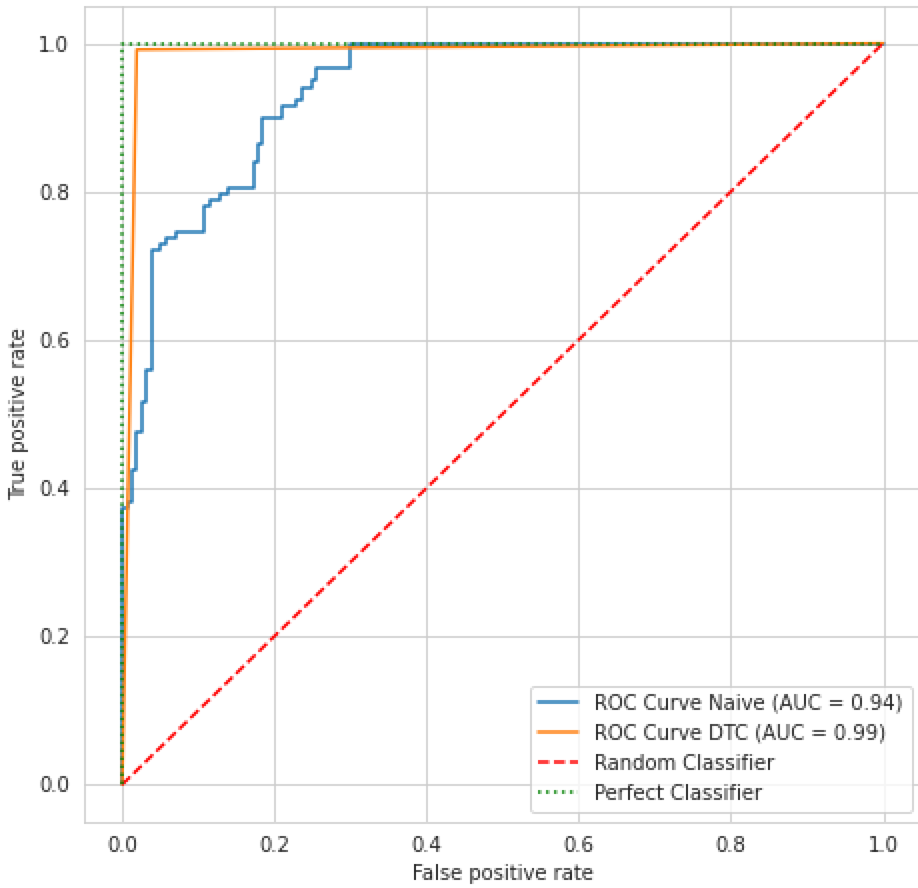

```
Accuracy: 0.8327272727272728
```

```
For Decision Tree:
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       157
           1       0.97      0.99      0.98       118

    accuracy                           0.99       275
   macro avg       0.98      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275

Confusion Matrix for DT
```




```
Accuracy: 0.9854545454545455
```

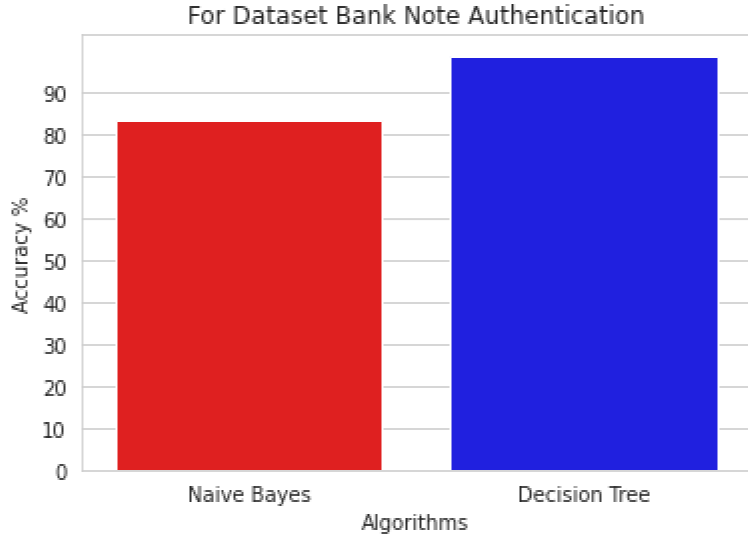
AUROC Curve:

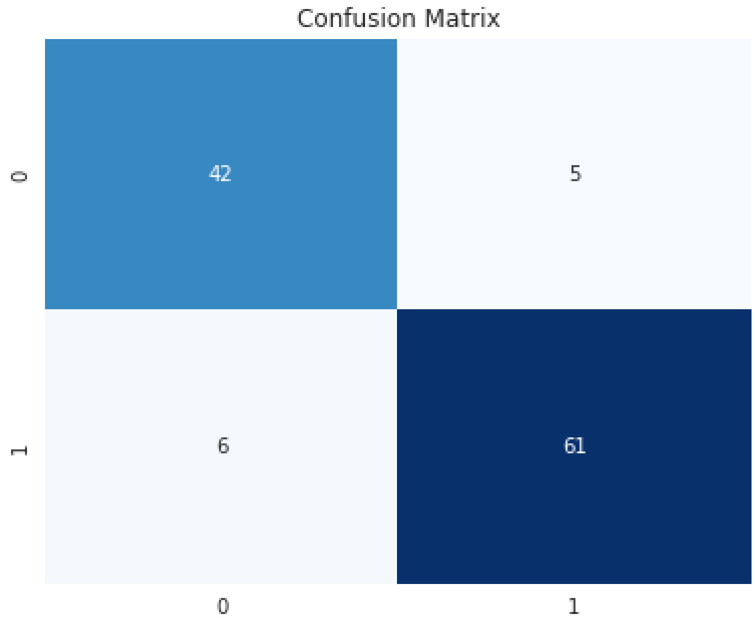



```
Comparing the two models:
Breast Cancer Prediction

Overview of Dataset:
   mean radius  mean texture  ...  worst fractal dimension  target
0        17.99         10.38  ...                  0.11890     0.0
1        20.57         17.77  ...                  0.08902     0.0
2        19.69         21.25  ...                  0.08758     0.0
3        11.42         20.38  ...                  0.17300     0.0
4        20.29         14.34  ...                  0.07678     0.0

[5 rows x 31 columns]
```

```
Seeing dataset stats:
       mean radius mean texture  ... worst fractal dimension      target
count   569.000000   569.000000  ...               569.000000  569.000000
mean     14.127292    19.289649  ...                 0.083946    0.627417
std       3.524049     4.301036  ...                 0.018061    0.483918
min       6.981000     9.710000  ...                 0.055040    0.000000
25%      11.700000    16.170000  ...                 0.071460    0.000000
50%      13.370000    18.840000  ...                 0.080040    1.000000
75%      15.780000    21.800000  ...                 0.092080    1.000000
max      28.110000    39.280000  ...                 0.207500    1.000000

[8 rows x 31 columns]
For Naive Bayes:
Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.89      0.88        47
         1.0       0.92      0.91      0.92        67

    accuracy                           0.90       114
   macro avg       0.90      0.90      0.90       114
weighted avg       0.90      0.90      0.90       114

Confusion Matrix for NB
```

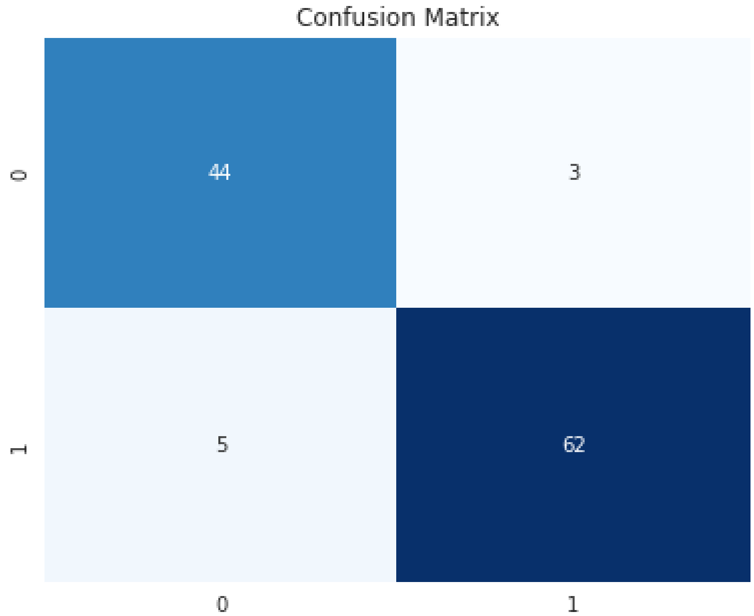
For Dataset Bank Note Authentication





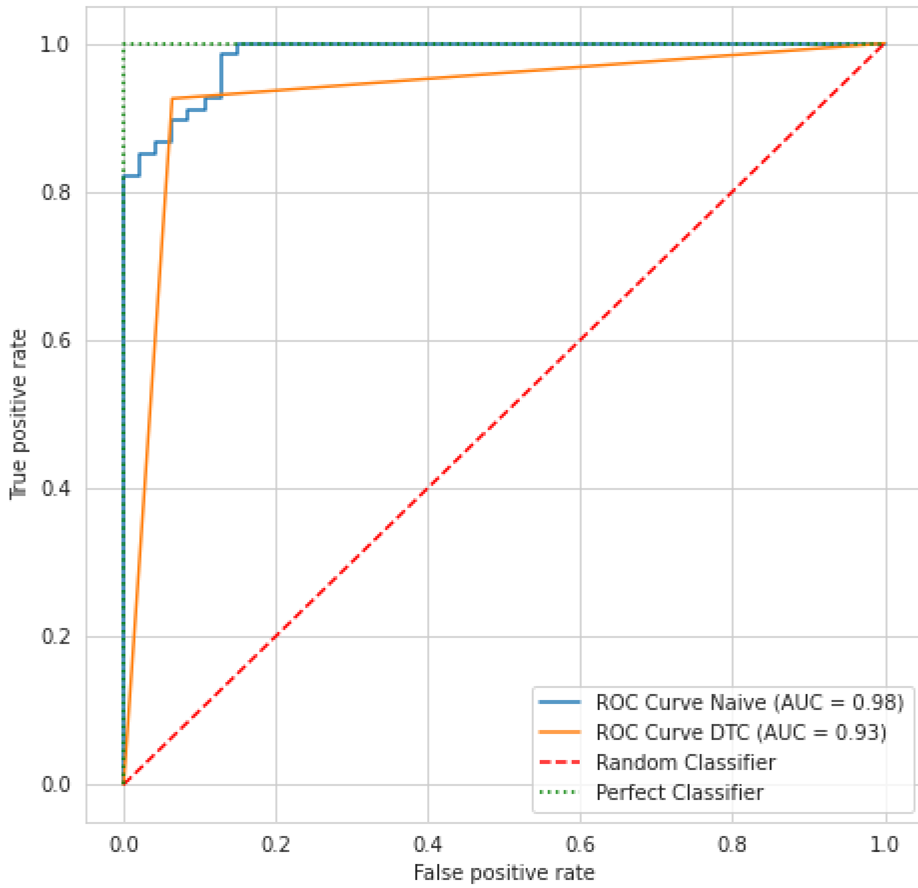

```
Accuracy: 0.9035087719298246

For Decision Tree:
Classification Report:
              precision    recall  f1-score   support

         0.0       0.90      0.94      0.92        47
         1.0       0.95      0.93      0.94        67

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114

Confusion Matrix for DT
```

```
Confusion Matrix for DT
```

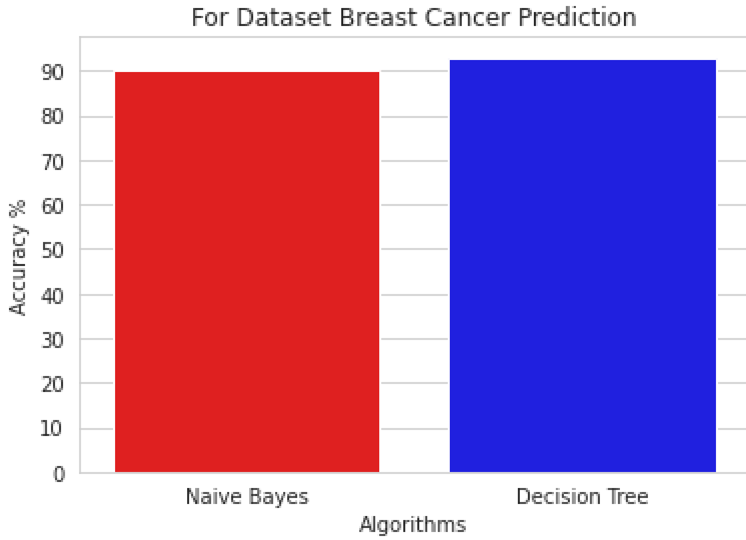


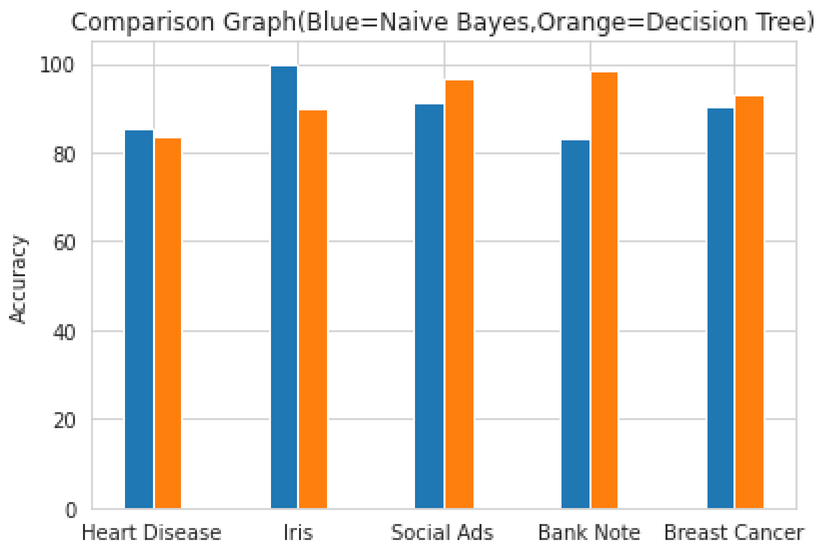

```
Accuracy: 0.9298245614035088
```

AUROC Curve:



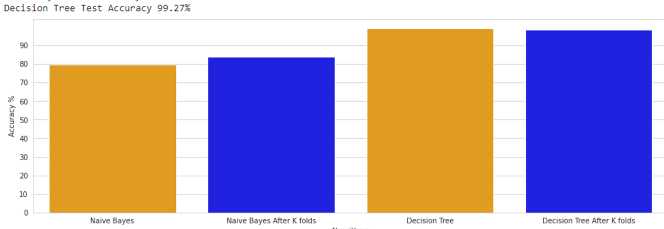
Comparing the two models:



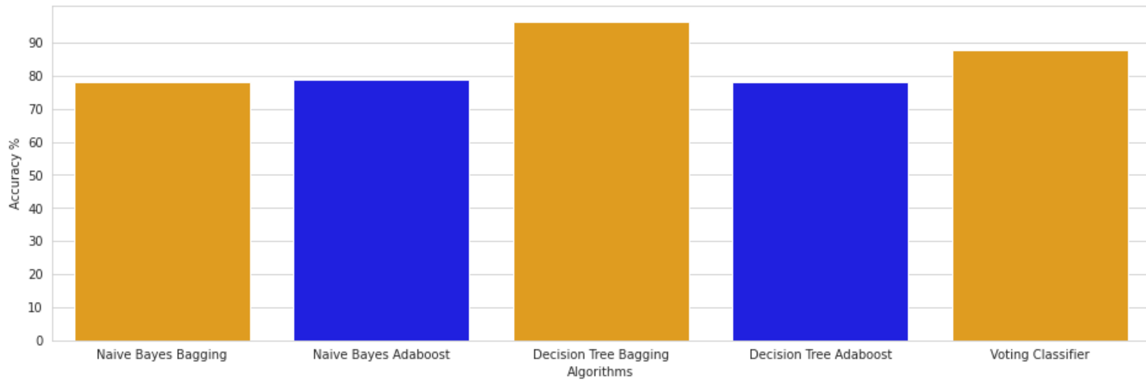


Part C

```
For Naive Bayes:
accuracy of each fold - [0.855072463768116, 0.8188405797101449, 0.8321167883211679, 0.868613138686l314, 0.8102189781021898, 0.8394160583941606, 0.8686131386861314, 0.8613138686131386, 0.8467153284671532, 0.7883211678832117]
Avg accuracy : 0.8389241510631547
For Decision Tree:
accuracy of each fold - [0.9927536231884058, 0.9927536231884058, 0.9854014598540146, 0.9781021897810219, 0.9854014598540146, 0.9781021897810219, 0.9927007299270073, 1.0, 0.9708029197080292, 0.9635036496350365]
Avg accuracy : 0.9839521844916957
Accuracy after KFold Cross Validation for Naive Bayes : 83.8924
Accuracy after KFold Cross Validation for Decision Tree : 98.3952
Naive Bayes Test Accuracy 79.56%
Decision Tree Test Accuracy 99.27%
```



```
Bagging Ensemble Model for Decision Tree Classifier: 96.3504
Bagging Ensemble Model for Naive Bayes Classifier: 78.1022
AdaBoost Ensemble Model for Decison Tree Classifier: 97.0803
AdaBoost Ensemble Model for Naive BayesClassifier: 78.8321
VotingClassifier Ensemble Model: 87.5912
```



Conclusion:

Hence, we learnt that

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods :

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.
- Decision tree can be computationally expensive to train.

For naive bayes:

The following are some of the benefits of the Naive Bayes classifier:

- It is simple and easy to implement
- It doesn’t require as much training data
- It handles both continuous and discrete data
- It is highly scalable with the number of predictors and data points
- It is fast and can be used to make real-time predictions
- It is not sensitive to irrelevant features

Disadvantages of Naive Bayes:

- Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.
- This algorithm faces the ‘zero-frequency problem’ where it assigns zero probability to a categorical variable whose category in the test data set wasn’t available in the training dataset. It would be best if you used a smoothing technique to overcome this issue.
- Its estimations can be wrong in some cases, so you shouldn’t take its probability outputs very seriously.