NAME: **Siddharth Unny**

SAP ID: **60004200080**

BRANCH: **A4**

SUBJECT: **Data Warehousing and Mining**

EXPERIMENT – 3

Implementation of Classification algorithm

AIM: Implementation of Classification algorithm using

1. Decision Tree ID3
2. Naïve Bayes algorithm

THEORY :

PART A) Program using inbuilt functions.

Predict class of unseen samples.

Results should display

1. Confusion matrix
2. Classifier accuracy

CODE :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
train_df = pd.read_csv(titanic_path)train_df.head()
```

```
train_df = pd.read_csv(titanic_path)
train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
#preprocessing of dataset
train_df["Name"]
train_df["Title"]=''
for i in range(len(train_df)):
  title_search = train_df["Name"][i]
  title = str(title_search).split(",")
  finaltitle = title[1].split(".")
  train_df["Title"][i]=finaltitle[0]
```

```
#remove nan values
nan_values = train_df.isna()
more = nan_values.mean().round(5)*100
print(more)
for i in range(len(more)):
  if(more[i] >= 50):
    remove = train_df.columns[i]
train_df.drop(remove, inplace=True, axis=1)
#filling nans
embark_col = train_df["Embarked"]
for i in range(len(embark_col)):
  if train_df["Embarked"].isnull().any():
    train_df["Embarked"]=train_df["Embarked"].fillna(train_df["Embarked"].mode()[0])
for i in range(len(train_df)):
  if train_df["Age"].isnull().any():     train_df["Age"]=train_df["Age"].fillna(train_df["Age"].mean())
#merging family_mebers
train_df["Family_members"]=train_df["SibSp"]+train_df["Parch"]
train_df.drop("SibSp", inplace=True, axis=1)
train_df.drop("Parch", inplace=True, axis=1)
train_df.drop("Name", inplace=True, axis=1)
train_df.drop("Ticket", inplace=True, axis=1)
train_df.drop("PassengerId" , inplace = True ,axis=1)
#Normalizing
def normalize(data_list):
  X_new = []
  X_max = data_list[0]
  for i in range(1, len(data_list)):
      if data_list[i] > X_max:
          X_max = data_list[i]
  X_min = min(data_list)
  for i in data_list:
    X_new.append( (i - X_min)/(X_max - X_min) )
  return X_new
train_df["Age"] = normalize(train_df["Age"])
train_df["Fare"]  =  normalize(train_df["Fare"])
```

```
train_df["Family_members"] = normalize(train_df["Family_members"])
train_df = pd.get_dummies(train_df, columns=["Pclass", "Sex", "Title", "Embarked"])

titanic_df = pd.DataFrame(train_df)
y = titanic_df["Survived"]
titanic_df.drop("Survived", inplace=True, axis=1)
X = titanic_df
#Gaussian Naive Bayes Theorem

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.30)

from sklearn.naive_bayes import GaussianNB
gauss = GaussianNB()
gauss.fit(X_train, y_train)

prediction_score = gauss.predict(X_test)
print('Prediction size : ' , prediction_score.size)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test , prediction_score)
print('Confusion matrix : ')
print(cm)

from sklearn import metrics
print('Classification Report : ')
print(metrics.classification_report(y_test,prediction_score))

print('Gaussian Naive Bayes Accuracy:',(metrics.accuracy_score(y_test,prediction_score)))
```

```
Prediction size :  268
Confusion matrix :
[[130  25]
 [ 30  83]]
Classification Report :
              precision    recall  f1-score   support

           0       0.81      0.84      0.83       155
           1       0.77      0.73      0.75       113

    accuracy                           0.79       268
   macro avg       0.79      0.79      0.79       268
weighted avg       0.79      0.79      0.79       268

Gaussian Naive Bayes Accuracy: 0.7947761194029851
```

```
#Decision Tree Classifier

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.30)

from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)

prediction_score = classifier.predict(X_test)
print('Prediction size : ' , prediction_score.size)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test , prediction_score)
print('Confusion matrix : ')
print(cm)

from sklearn import metrics
print('Classification  Report  : ')
print(metrics.classification_report(y_test,prediction_score))

print('Decision  Tree  Classifier  Accuracy:',(metrics.accuracy_score(y_test,prediction_score)))
```

```
Prediction size :  268
Confusion matrix :
[[139  35]
 [ 22  72]]
Classification Report :
              precision    recall  f1-score   support

           0       0.86      0.80      0.83       174
           1       0.67      0.77      0.72        94

    accuracy                           0.79       268
   macro avg       0.77      0.78      0.77       268
weighted avg       0.80      0.79      0.79       268

Decision Tree Classifier Accuracy: 0.7873134328358209
```

PART B)

1. Compare results of DT and ND for 5 datasets.
2. Plot AUROC
3. Plot comparison graphs using the results of DT and NB

1. Wine Dataset

```
train_df = pd.read_csv(wine_path)
train_df.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.30)
from sklearn.tree import DecisionTreeClassifier
clf_tree= DecisionTreeClassifier(criterion='entropy', random_state=0)
clf_tree.fit(X_train, y_train)
from sklearn.naive_bayes import GaussianNB
gauss = GaussianNB()
gauss.fit(X_train, y_train)
y_score1 = clf_tree.predict_proba(X_test)[:,1]
y_score2 = gauss.predict_proba(X_test)[:,1]
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn import metrics
def confusionMatrix(y_test,prediction_score,name):cm
  = confusion_matrix(y_test , prediction_score)
  print('Confusion matrix of  '+ name)
  print(cm)
  print('Classification Report of '+ name)
  print(metrics.classification_report(y_test,prediction_score))
```

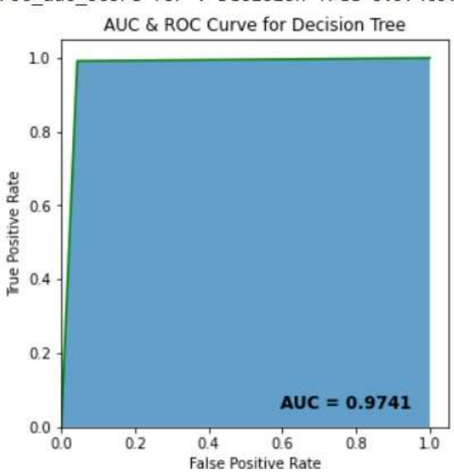
```
confusionMatrix(y_test, clf_tree.predict(X_test),"Decision Tree")
confusionMatrix(y_test, gauss.predict(X_test),"Naive Bayes")
def plotfigures(y_score , y_test,X_test, name):
  #Creating False and True Positive Rates and printing Scores
  false_positive_rate, true_positive_rate, threshold1 = roc_curve(y_test, y_score)
  print('roc_auc_score for : ' + name , roc_auc_score(y_test, y_score))
  #Plotting ROC Curves
  plt.subplots(1, figsize=(8,5))
  plt.axis('scaled')
  plt.xlim([0, 1.05])
  plt.ylim([0,  1.05])
  plt.title("AUC & ROC Curve for "+ name)
  plt.plot(false_positive_rate, true_positive_rate, 'g')
  plt.fill_between(false_positive_rate, true_positive_rate, alpha=0.7)
  plt.text(0.95, 0.05, 'AUC = %0.4f' % roc_auc_score(y_test, y_score), ha='right', fontsize=12,
weight='bold' )
  plt.xlabel("False Positive Rate")
  plt.ylabel("True Positive Rate")
  plt.show()

plotfigures(y_score1 , y_test,X_test, "Decision Tree")
plotfigures(y_score2 , y_test,X_test, "Naive Bayes")
def Comparison(y_test,y_score1,y_score2,X_test):
  false_positive_rate, true_positive_rate, thresolds = roc_curve(y_test, y_score1)
  plt.plot(false_positive_rate, true_positive_rate, lw=2, alpha=0.3, label='ROC Decision Tree (AUC
= %0.4f)' % roc_auc_score(y_test, y_score1))
  false_positive_rate, true_positive_rate, thresolds = roc_curve(y_test, y_score2)
  plt.plot(false_positive_rate, true_positive_rate, lw=2, alpha=0.3, label='ROC Naive Bayes (AUC =
%0.4f)' % roc_auc_score(y_test, y_score2))
  plt.title('ROC Curve Comparison')
  plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
  plt.plot([0,1],[0,1],'r--')
  plt.xlim([0,1.1])
  plt.ylim([0,1.1])
  plt.ylabel('True Positive Rate')
```

```
plt.xlabel('False Positive Rate')
plt.show()

Comparison(y_test,y_score1,y_score2,X_test)
```

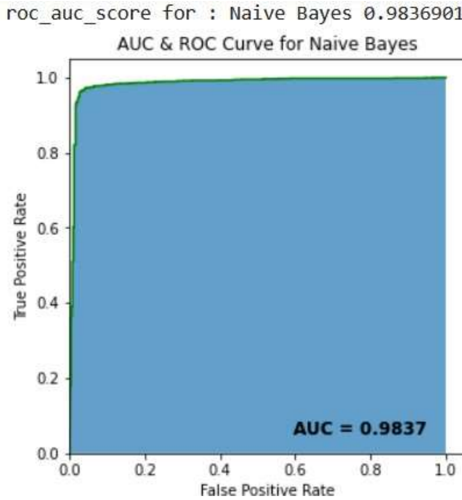
```
Confusion matrix of  Decision Tree
[[ 487   17]
 [  29 1406]]
Classification Report of  Decision Tree
              precision    recall  f1-score   support

         0.0       0.94      0.97      0.95       504
         1.0       0.99      0.98      0.98      1435

    accuracy                           0.98      1939
   macro avg       0.97      0.97      0.97      1939
weighted avg       0.98      0.98      0.98      1939
```

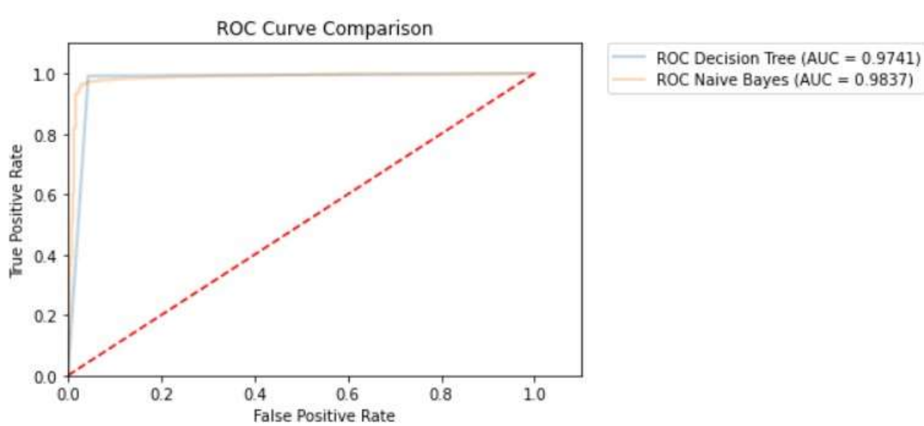
```
Confusion matrix of  Naive Bayes
[[ 472   32]
 [  35 1400]]
Classification Report of  Naive Bayes
              precision    recall  f1-score   support

         0.0       0.93      0.94      0.93       504
         1.0       0.98      0.98      0.98      1435

    accuracy                           0.97      1939
   macro avg       0.95      0.96      0.96      1939
weighted avg       0.97      0.97      0.97      1939
```

```
roc_auc_score for : Decision Tree 0.9740977168455618
```




```
roc_auc_score for : Naive Bayes 0.9836901266457252
```





2.Diabetes Dataset

```
train_df = pd.read_csv(diabetes_path)
train_df.head()
```

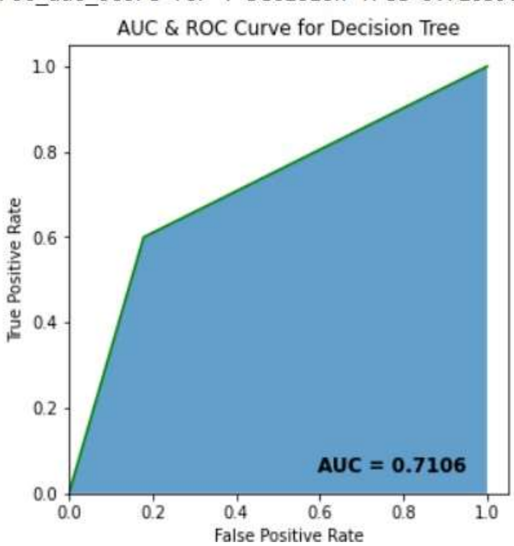
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
y = train_df["Outcome"]
train_df.drop("Outcome", inplace=True, axis=1)
X = train_df
```

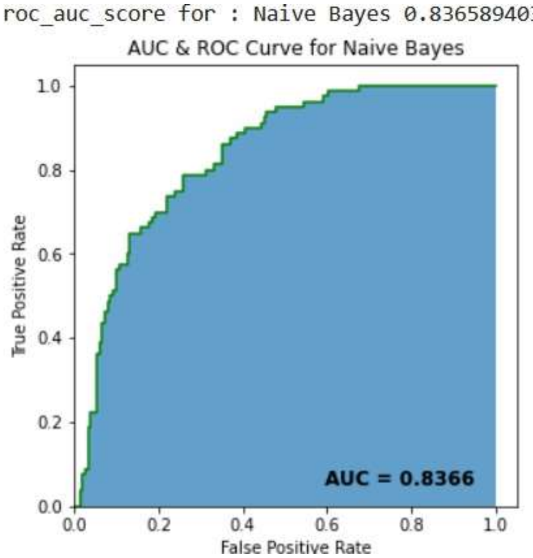
```
Confusion matrix of  Decision Tree
[[124  27]
 [ 32  48]]
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       0.79      0.82      0.81       151
           1       0.64      0.60      0.62        80

    accuracy                           0.74       231
   macro avg       0.72      0.71      0.71       231
weighted avg       0.74      0.74      0.74       231
```

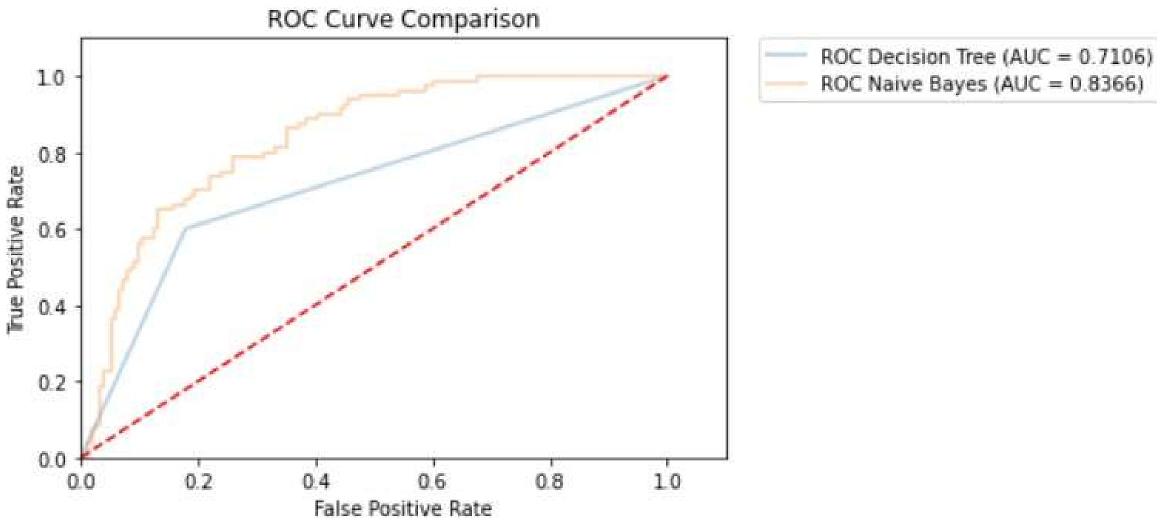
```
Confusion matrix of  Naive Bayes
[[131  20]
 [ 29  51]]
Classification Report of  Naive Bayes
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       151
           1       0.72      0.64      0.68        80

    accuracy                           0.79       231
   macro avg       0.77      0.75      0.76       231
weighted avg       0.78      0.79      0.78       231
```

```
roc_auc_score for : Decision Tree 0.7105960264900663
```




```
roc_auc_score for : Naive Bayes 0.83658940397351
```





3.Star Dataset

```
train_df = pd.read_csv(star_path)
train_df.tail()
```

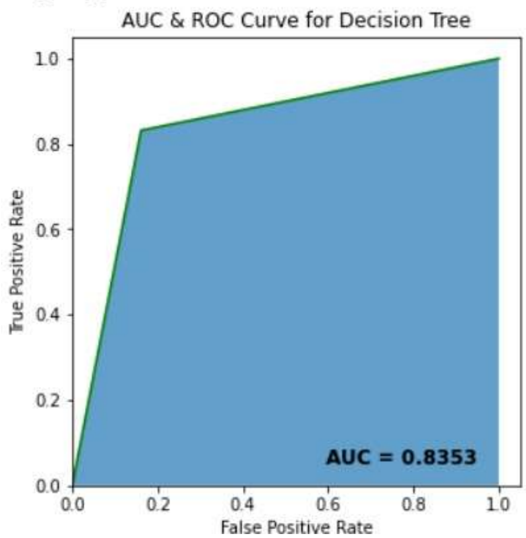
	Vmag	Plx	e_Plx	B-V	SpType	Amag	TargetClass
3637	7.29	3.26	0.95	1.786	K4III	14.856089	0
3638	8.29	6.38	1.00	0.408	F2IV/V	17.314104	1
3639	6.11	2.42	0.79	1.664	M0/M1IIICNp	13.029078	0
3640	7.94	4.94	2.90	0.210	A5V	16.408636	1
3641	8.81	1.87	1.23	1.176	K1/K2III	15.169209	0

```
y = train_df["TargetClass"]
train_df.drop(["TargetClass","SpType"], inplace=True, axis=1)
X = train_df
```

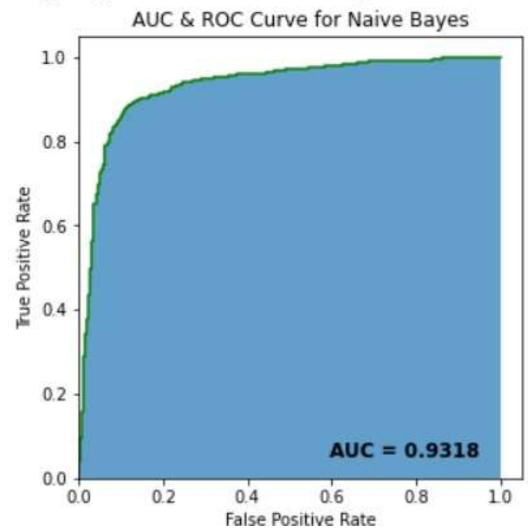
```
Confusion matrix of  Decision Tree
[[463  89]
 [ 91 450]]
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       0.84      0.84      0.84       552
           1       0.83      0.83      0.83       541

    accuracy                           0.84      1093
   macro avg       0.84      0.84      0.84      1093
weighted avg       0.84      0.84      0.84      1093
```

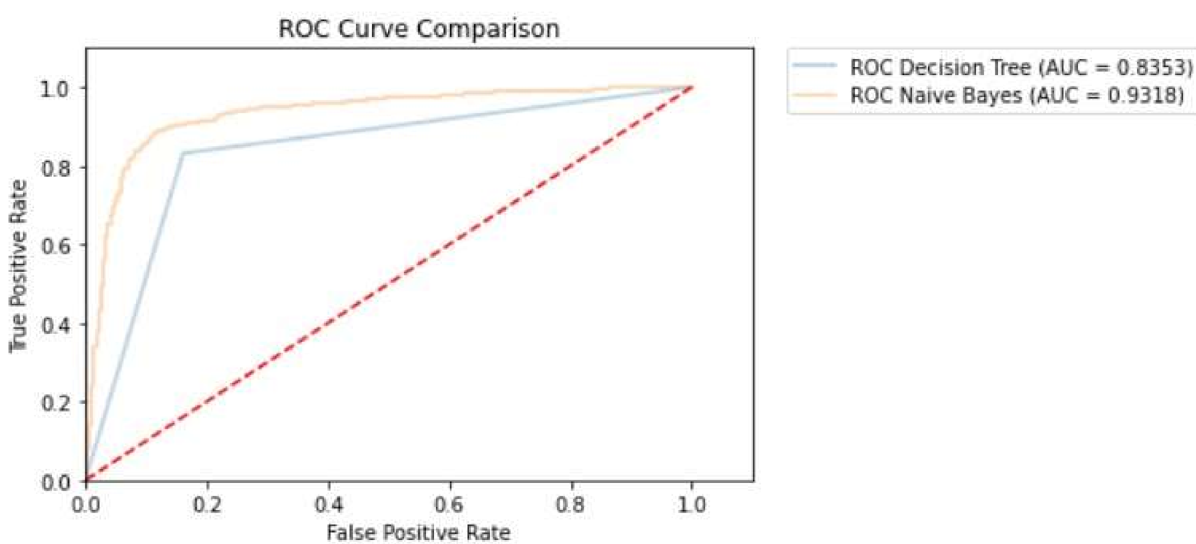
```
Confusion matrix of  Naive Bayes
[[513  39]
 [110 431]]
Classification Report of  Naive Bayes
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       552
           1       0.92      0.80      0.85       541

    accuracy                           0.86      1093
   macro avg       0.87      0.86      0.86      1093
weighted avg       0.87      0.86      0.86      1093
```

```
roc_auc_score for : Decision Tree 0.835280545956227
```




```
roc_auc_score for : Naive Bayes 0.9317588202202041
```





4.Gender Dataset

```
train_df = pd.read_csv(gender_path)
train_df.head()
```

	Favorite Color	Favorite Music Genre	Favorite Beverage	Favorite Soft Drink	Gender
0	Cool	Rock	Vodka	7UP/Sprite	F
1	Neutral	Hip hop	Vodka	Coca Cola/Pepsi	F
2	Warm	Rock	Wine	Coca Cola/Pepsi	F
3	Warm	Folk/Traditional	Whiskey	Fanta	F
4	Cool	Rock	Vodka	Coca Cola/Pepsi	F

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in train_df.columns:
    train_df[i]=le.fit_transform(train_df[i])
train_df.head()
```

	Favorite Color	Favorite Music Genre	Favorite Beverage	Favorite Soft Drink	Gender
0	0	6	3	0	0
1	1	2	3	1	0
2	2	6	5	1	0
3	2	1	4	2	0
4	0	6	3	1	0

y = train_df["Gender"]

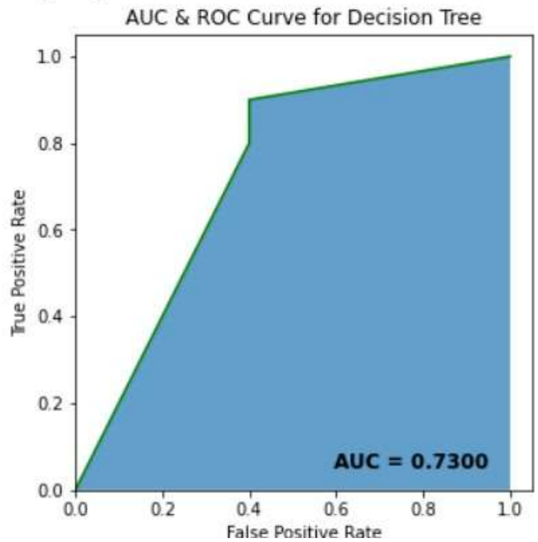
train_df.drop("Gender", inplace=True, axis=1)X

= train_df

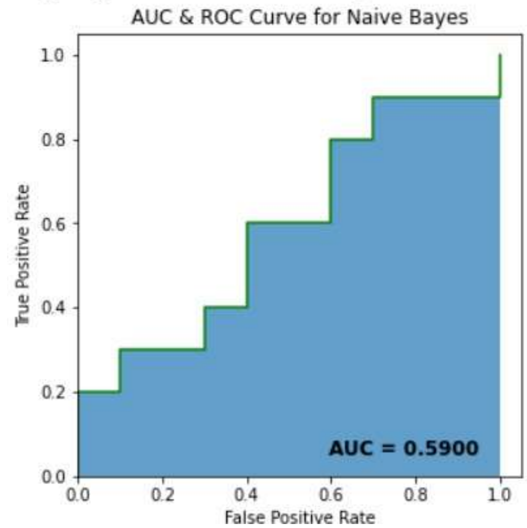
```
Confusion matrix of  Decision Tree
[[6 4]
 [2 8]]
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       0.75      0.60      0.67        10
           1       0.67      0.80      0.73        10

    accuracy                           0.70        20
   macro avg       0.71      0.70      0.70        20
weighted avg       0.71      0.70      0.70        20
```

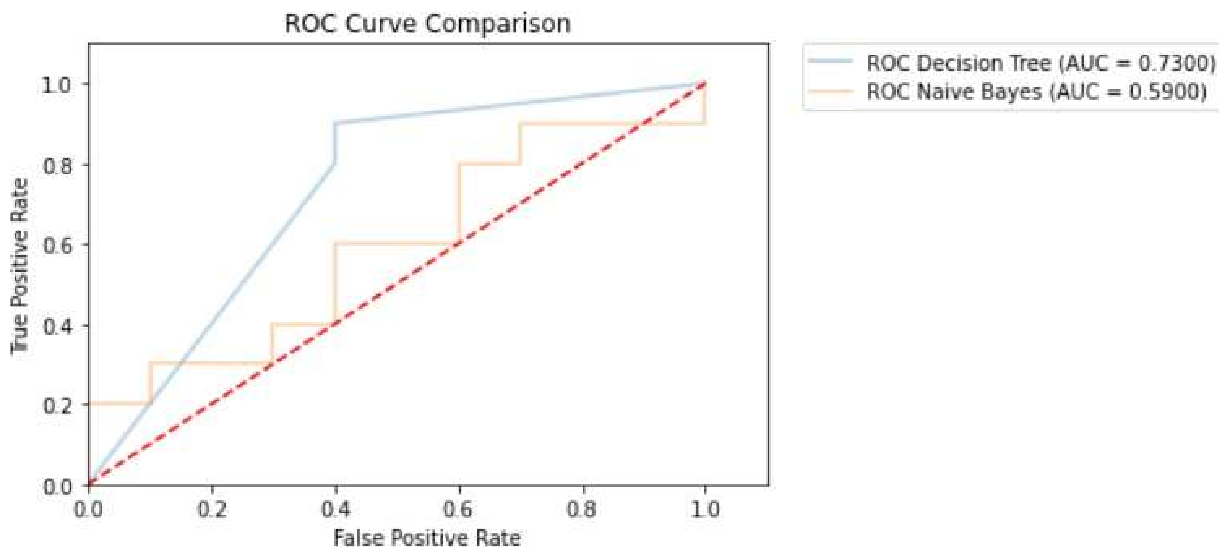
```
Confusion matrix of  Naive Bayes
[[7 3]
 [6 4]]
Classification Report of  Naive Bayes
              precision    recall  f1-score   support

           0       0.54      0.70      0.61        10
           1       0.57      0.40      0.47        10

    accuracy                           0.55        20
   macro avg       0.55      0.55      0.54        20
weighted avg       0.55      0.55      0.54        20
```

```
roc_auc_score for : Decision Tree 0.73
```




```
roc_auc_score for : Naive Bayes 0.5900000000000001
```





5.Mushroom Dataset

```
train_df = pd.read_csv(mushroom_path)
train_df.head()
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat	class
0	x	s	n	t	p	f	c	n	k	e	...	w	w	p	w	o	p	k	s	u	p
1	x	s	y	t	a	f	c	b	k	e	...	w	w	p	w	o	p	n	n	g	e
2	b	s	w	t	l	f	c	b	n	e	...	w	w	p	w	o	p	n	n	m	e
3	x	y	w	t	p	f	c	n	n	e	...	w	w	p	w	o	p	k	s	u	p
4	x	s	g	f	n	f	w	b	k	t	...	w	w	p	w	o	e	n	a	g	e

5 rows × 23 columns

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in train_df.columns:
    train_df[i]=le.fit_transform(train_df[i])
train_df.head()
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat	class
0	5	2	4	1	6	1	0	1	4	0	...	7	7	0	2	1	4	2	3	5	1
1	5	2	9	1	0	1	0	0	4	0	...	7	7	0	2	1	4	3	2	1	0
2	0	2	8	1	3	1	0	0	5	0	...	7	7	0	2	1	4	3	2	3	0
3	5	3	8	1	6	1	0	1	5	0	...	7	7	0	2	1	4	2	3	5	1
4	5	2	3	0	5	1	1	0	4	1	...	7	7	0	2	1	0	3	0	1	0

5 rows × 23 columns

y = train_df["class"]

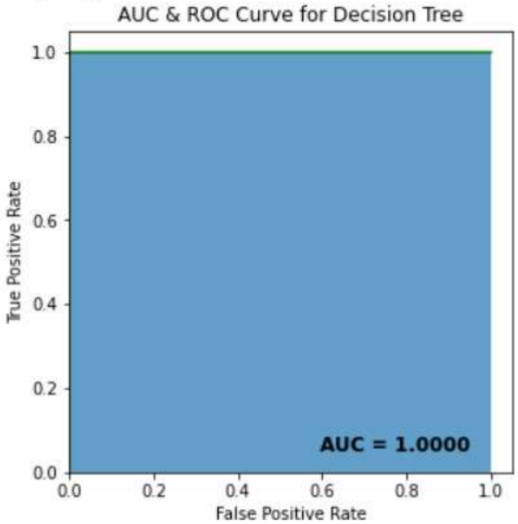
train_df.drop("class", inplace=True, axis=1)X

= train_df

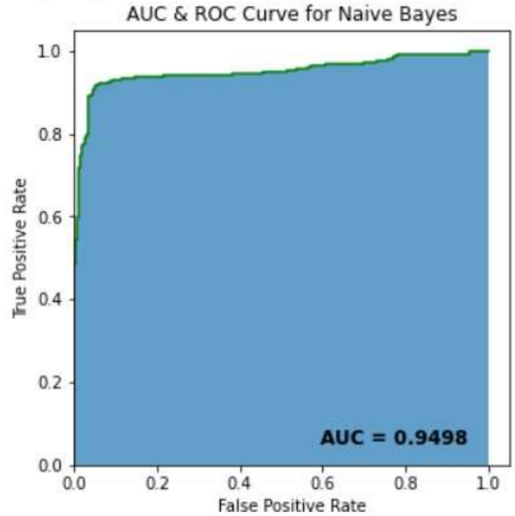
```
Confusion matrix of  Decision Tree
[[1265    0]
 [   0 1173]]
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1265
           1       1.00      1.00      1.00      1173

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

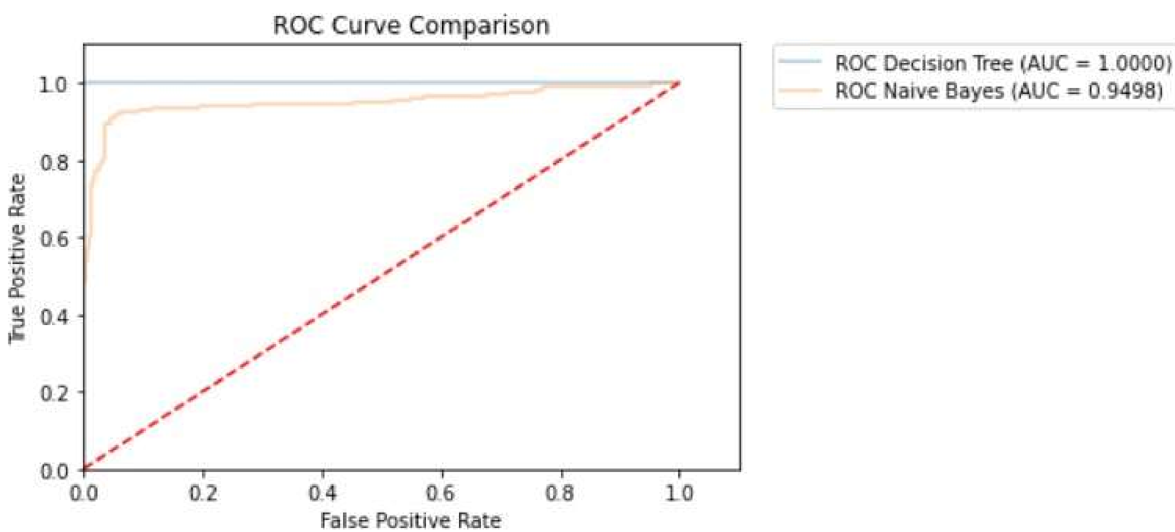
```
Confusion matrix of  Naive Bayes
[[1154  111]
 [  86 1087]]
Classification Report of  Naive Bayes
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1265
           1       0.91      0.93      0.92      1173

    accuracy                           0.92      2438
   macro avg       0.92      0.92      0.92      2438
weighted avg       0.92      0.92      0.92      2438
```

```
roc_auc_score for : Decision Tree 1.0
```




```
roc_auc_score for : Naive Bayes 0.9498411222196388
```





1. In Wine Quality Data, both Naïve Bayes and Decision Tree have approximately the same accuracy .
2. In Diabetes Data, Naive Bayes has better accuracy than Decision Tree so it is the better classifier.
3. In Star Data, Naive Bayes has better accuracy than Decision Tree so it is the better classifier.
4. In Gender Data, Decision Tree has best accuracy so it is the better classifier .
5. In Mushroom Data , both Naïve Bayes and Decision Tree have approximately the same accuracy

PART C) Modify DT/NB to use k-fold cross validation and ensemble models

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve,
mean_squared_error
from sklearn.model_selection import KFold, cross_val_score
from sklearn.ensemble import VotingClassifier
#Using star dataset
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20, random_state=0)
print("Using 7 folds cross validation")
k = 7
kf = KFold(n_splits=k, random_state=None)
nb = GaussianNB()
dt = DecisionTreeClassifier(random_state=0)
result = cross_val_score(nb , X, y, cv = kf)
print("Avg accuracy for Naive Bayes : {}".format(result.mean()))
result = cross_val_score(dt , X, y, cv = kf)
print("Avg accuracy for Decision Tree : {}".format(result.mean()))
print()
final_model = VotingClassifier(
estimators=[('nb', nb), ('dt', dt)], voting='hard')
result = cross_val_score(final_model, X, y, cv = kf)
print("\nEnsemble Using Max votings")
print("Avg accuracy with k-folds cross validation : {}".format(result.mean()))
final_model.fit(X_train, y_train)
pred_final = final_model.predict(X_test)
cm2 = confusion_matrix(y_test, pred_final)
print("\nConfusion matrix :\n",cm2)
print("Classifier Accuracy without cross validation : ",accuracy_score(y_test, pred_final))
```

OUTPUT :

```
Using 7 folds cross validation
Avg accuracy for Naive Bayes : 0.8682077998776655
Avg accuracy for Decision Tree : 0.8426667861888592


Ensemble Using Max votings
Avg accuracy with k-folds cross validation : 0.852554786863808

Confusion matrix :
 [[346  24]
 [ 89 270]]
Classifier Accuracy without cross validation :  0.8449931412894376
```

CONCLUSION :

Python was used to implement the classification algorithms Decision Tree (ID3) and Naive Bayes. We learnt different ways to evaluate the classification model using : Confusion Matrix and AUROC curve .

The confusion matrix provides us a matrix/table as output and describes the performance of the model.It is also known as the error matrix. The matrix consists of predictions resulting in a summarized form, which has a total number of correct predictions and incorrect predictions.

ROC curve stands for Receiver Operating Characteristics Curve and AUC stands for Area Under the Curve.It is a graph that shows the performance of the classification model at different thresholds.To visualize the performance of the multi-class classification model, we use the AUC-ROC Curve.The algorithms were modified using k-fold cross validation and ensemble models, and the AUROC and comparison graphs were generated.