Name: - Abhijit Saha *Batch: - B1*
SAP ID: - 60004200112 *Subject: - Data Mining & Warehousing*

Experiment – 3

Aim: - Implementation of Classification algorithm Using 1. Decision Tree ID3 and 2. Naïve Bayes algorithm

Theory: -

Classification Algorithm: - The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observations into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called targets/labels or categories.

Decision Tree: - Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Naïve Bayes: - Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Implementation: -

```
# Connecting drive
from google.colab import drive
drive.mount('/content/drive')

# Importing packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate
```

```
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier,
RandomForestClassifier
from sklearn import metrics
from sklearn import tree
```

Part A:

```
# Variables for making graph
dataset, gnb_acc, tree_acc =[], [], []
```

Iris Dataset:

```
# Pre-processing
dataset.append('Iris')
df = pd.read_csv('/content/drive/MyDrive/Iris.csv')

for i in df.columns:
  print(i)
```

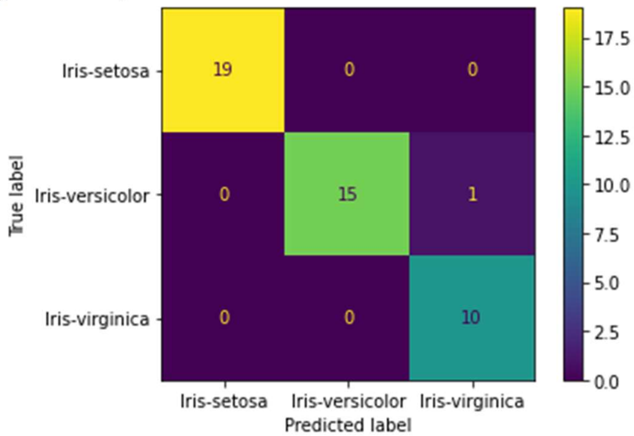
```
Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
```

```
df.drop(columns = 'Id', inplace = True)
df.replace(to_replace = 'Iris-setosa', value = 0, inplace = True)
df.replace(to_replace =  'Iris-versicolor', value = 1, inplace = True)
df.replace(to_replace = 'Iris-virginica', value = 2, inplace = True)

# Splitting into training and testing set
X = pd.DataFrame(df, columns = ['SepalLengthCm',
'SepalWidthCm','PetalLengthCm', 'PetalWidthCm'])
y = df['Species'].to_numpy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
1002)

# Gaussian Naïve Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
```

```
acc = accuracy_score(y_pred, y_test)
gnb_acc.append(acc)
print(f'Accuracy using Gaussian Naive Bayes: {acc}')
```

```
Accuracy using Gaussian Naive Bayes: 0.9777777777777777
```

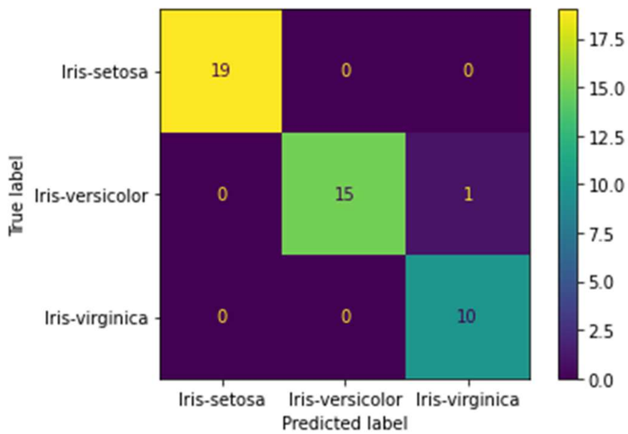
```
matrix = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = matrix,
display_labels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
cm_display.plot()
plt.show()
```




```
# Decision Tree
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, y_train)
predicted_value = model.predict(X_test)
acc = accuracy_score(predicted_value, y_test)
tree_acc.append(acc)
print(f'Accuracy using Decision Tree: {acc}')
```

```
Accuracy using Decision Tree: 0.9555555555555556
```

```
tree_matrix = metrics.confusion_matrix(y_test, y_pred)
tree_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = tree_matrix,
display_labels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
tree_cm_display.plot()
plt.show()
```



Stars Dataset:

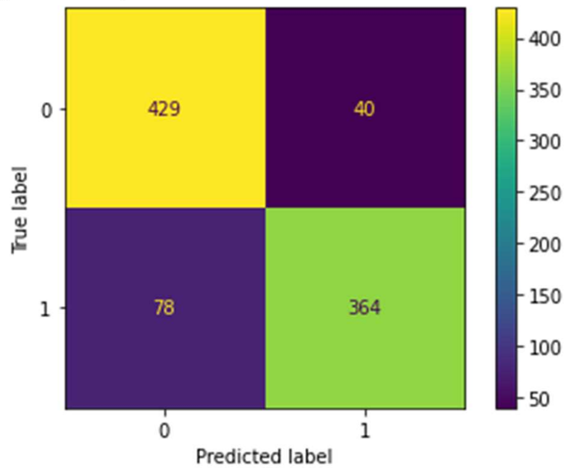
```
# Pre-processing
dataset.append('Stars')
df = pd.read_csv('/content/drive/MyDrive/Star3642_balanced.csv')

for i in df.columns:
  print(i)
```

```
Vmag
Plx
e_Plx
B-V
SpType
Amag
TargetClass
```

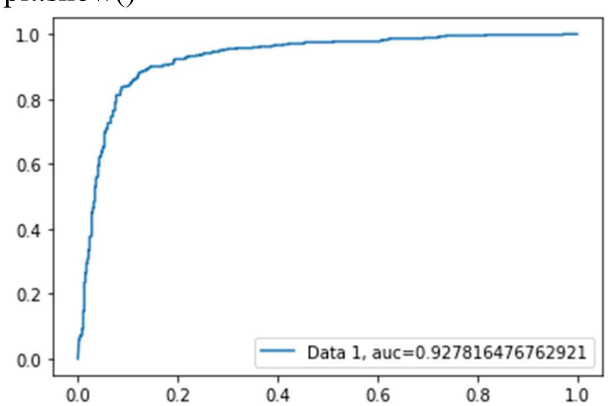
```
df.drop('SpType', axis = 1)

# Splitting into training and testing set
X = pd.DataFrame(df, columns = ['Vmag', 'Plx', 'e_Plx', 'B-V', 'Amag'])
y = df['TargetClass'].to_numpy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state
= 10242)

# Gaussian Naïve Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
acc = accuracy_score(y_pred, y_test)
gnb_acc.append(acc)
print(f'Accuracy using Gaussian Naive Bayes: {acc}')
```

```
Accuracy using Gaussian Naive Bayes: 0.8704720087815587
```

```
matrix = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = matrix, display_labels = ['0', '1'])
cm_display.plot()
plt.show()
```



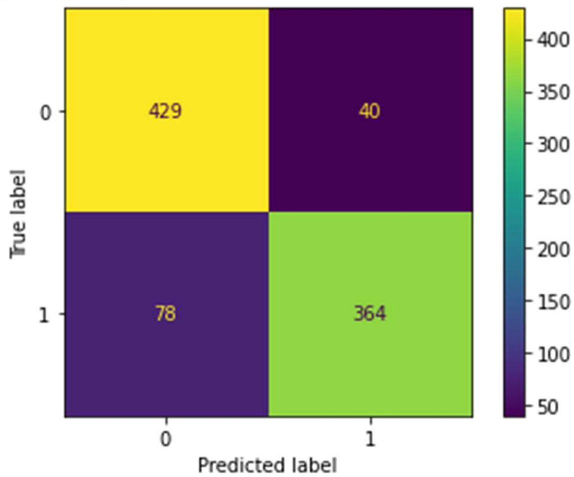

```
# ROC Curve
y_pred_proba = gnb.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



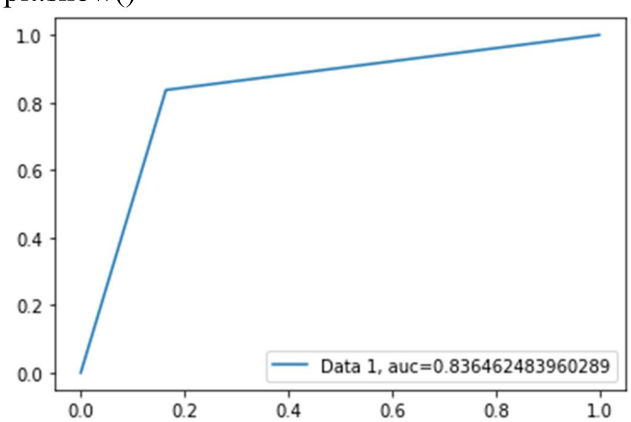

```
# Decision Tree
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, y_train)
predicted_value = model.predict(X_test)
acc = accuracy_score(predicted_value, y_test)
tree_acc.append(acc)
print(f'Accuracy using Decision Tree: {acc}')
```

```
Accuracy using Decision Tree: 0.8353457738748628
```

```
tree_matrix = metrics.confusion_matrix(y_test, y_pred)
tree_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = tree_matrix,
display_labels = ['0', '1'])
tree_cm_display.plot()
plt.show()
```




```
# ROC Curve
y_pred_proba = model.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



Heart Dataset:

 # Pre-processing
dataset.append('Heart Attack')
df = pd.read_csv('/content/drive/MyDrive/heart.csv')

for i in df.columns:
  print(i)

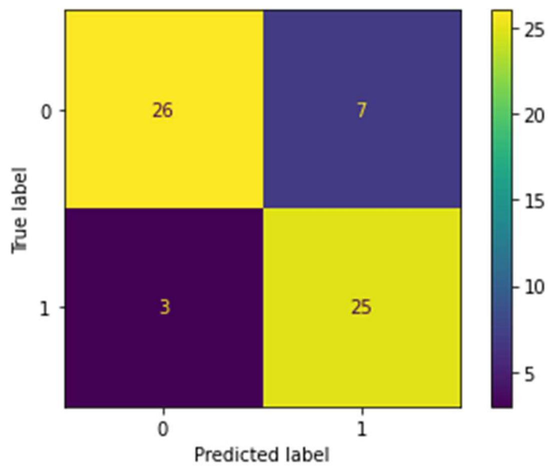
```
age
sex
cp
trtbps
chol
fbs
restecg
thalachh
exng
oldpeak
slp
caa
thall
output
```

 # Splitting into training and testing set
X = pd.DataFrame(df, columns = ['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh', 'exng', 'oldpeak', 'slp', 'caa', 'thall'])
y = df['output'].to_numpy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 10242)

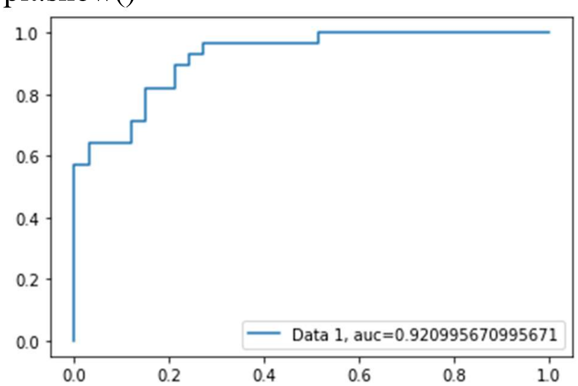
 # Gaussian Naïve Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
acc = accuracy_score(y_pred, y_test)
gnb_acc.append(acc)
print(f'Accuracy using Gaussian Naive Bayes: {acc}')

```
Accuracy using Gaussian Naive Bayes: 0.8360655737704918
```

matrix = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = matrix, display_labels = ['0', '1'])
cm_display.plot()
plt.show()



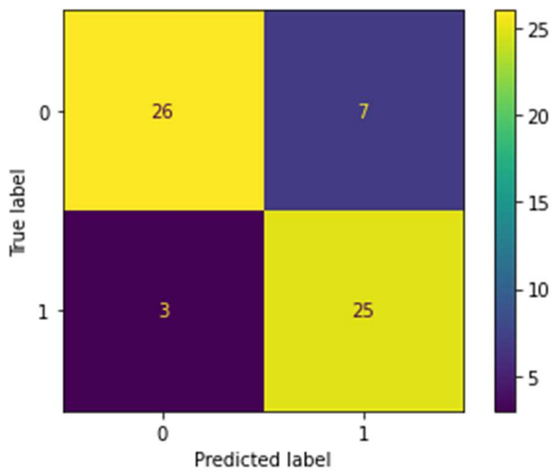

```
# ROC Curve
y_pred_proba = gnb.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



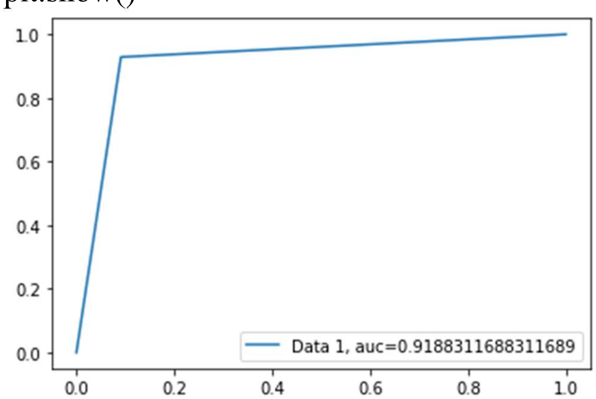

```
# Decision Tree
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, y_train)
predicted_value = model.predict(X_test)
acc = accuracy_score(predicted_value, y_test)
tree_acc.append(acc)
print(f'Accuracy using Decision Tree: {acc}')
```

```
Accuracy using Decision Tree: 0.9180327868852459
```

```
tree_matrix = metrics.confusion_matrix(y_test, y_pred)
tree_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = tree_matrix,
display_labels = ['0', '1'])
tree_cm_display.plot()
plt.show()
```




```
# ROC Curve
y_pred_proba = model.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



Mushroom Dataset:

 # Pre-processing

```
dataset.append('Mushroom')
df = pd.read_csv('/content/drive/MyDrive/mushrooms.csv')
for i in df.columns:
  print(i)
```

```
class
cap-shape
cap-surface
cap-color
bruises
odor
gill-attachment
gill-spacing
gill-size
gill-color
stalk-shape
stalk-root
stalk-surface-above-ring
stalk-surface-below-ring
stalk-color-above-ring
stalk-color-below-ring
veil-type
veil-color
ring-number
ring-type
spore-print-color
population
habitat
```

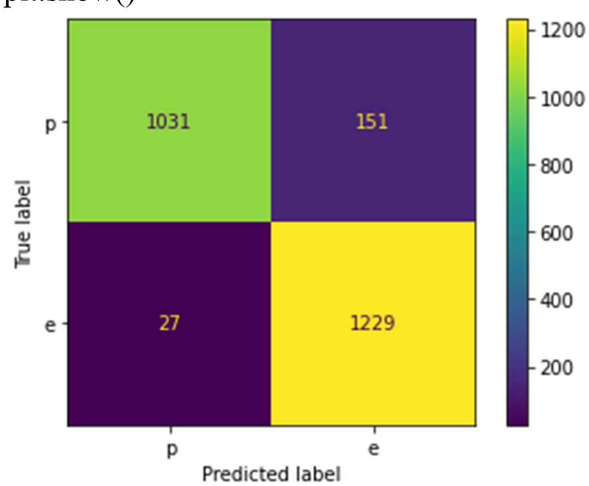
 # Replacing strings with numbers

```
df.replace({'cap-shape': {'x': 0, 'b': 1, 's': 2, 'f': 3, 'k': 4, 'c': 5},
           'cap-surface': {'s': 0, 'y': 1, 'f': 2, 'g': 3},
           'cap-color': {'n': 0, 'y': 1, 'w': 2, 'g': 3, 'e': 4, 'p': 5, 'b': 6, 'u': 7, 'c': 8, 'r': 9},
           'bruises': {'t': 0, 'f': 1},
           'odor': {'p': 0, 'a': 1, 'l': 2, 'n': 3, 'f': 4, 'c': 5, 'y': 6, 's': 7, 'm': 8},
           'gill-attachment': {'f': 0, 'a': 1},
           'gill-spacing': {'c': 0, 'w': 1},
           'gill-size': {'n': 0, 'b': 1},
           'gill-color': {'k': 0, 'n': 1, 'g': 2, 'p': 3, 'w': 4, 'h': 5, 'u': 6, 'e': 7, 'b': 8, 'r': 9, 'y': 10,
'o': 11},
           'stalk-shape': {'e': 0, 't': 1},
           'stalk-root': {'e': 0, 'c': 1, 'b': 2, 'r': 3, '?': 4},
           'stalk-surface-above-ring': {'s': 0, 'f': 1, 'k': 2, 'y': 3},
           'stalk-surface-below-ring': {'s': 0, 'f': 1, 'k': 2, 'y': 3},
           'stalk-color-above-ring': {'w': 0, 'g': 1, 'p': 2, 'n': 3, 'b': 4, 'e': 5, 'o': 6, 'c': 7, 'y':
8},
           'stalk-color-below-ring': {'w': 0, 'g': 1, 'p': 2, 'n': 3, 'b': 4, 'e': 5, 'o': 6, 'c': 7, 'y':
8},
           'veil-type': {'p': 0},
           'veil-color': {'w': 0, 'n': 1, 'o': 2, 'y': 3},
           'ring-number': {'o': 0, 't': 1, 'n': 2},
           'ring-type': {'p': 0, 'e': 1, 'l': 2, 'f': 3, 'n': 4},
           'spore-print-color': {'k': 0, 'n': 1, 'u': 2, 'h': 3, 'w': 4, 'r': 5, 'o': 6, 'y': 7, 'b': 8},
           'population': {'s': 0, 'n': 1, 'a': 2, 'v': 3, 'y': 4, 'c': 5},
```

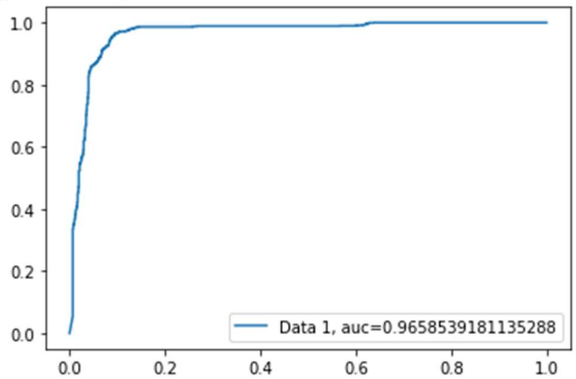
```
        'habitat': {'u': 0, 'g': 1, 'm': 2, 'd': 3, 'p': 4, 'w': 5, 'l': 6},
        'class': {'p': 0, 'e': 1}
        }, inplace = True)


# Splitting into training and testing set
X = pd.DataFrame(df, columns = ['cap-shape', 'cap-surface', 'cap-color', 'bruises',
'odor',
    'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
    'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
    'stalk-surface-below-ring', 'stalk-color-above-ring',
    'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number',
    'ring-type', 'spore-print-color', 'population', 'habitat'])
y = df['class'].to_numpy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
10242)

# Gaussian Naïve Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
acc = accuracy_score(y_pred, y_test)
gnb_acc.append(acc)
print(f'Accuracy using Gaussian Naive Bayes: {acc}')
```

```
Accuracy using Gaussian Naive Bayes: 0.9269893355209188
```

```
matrix = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = matrix,
display_labels = ['p', 'e'])
cm_display.plot()
plt.show()
```



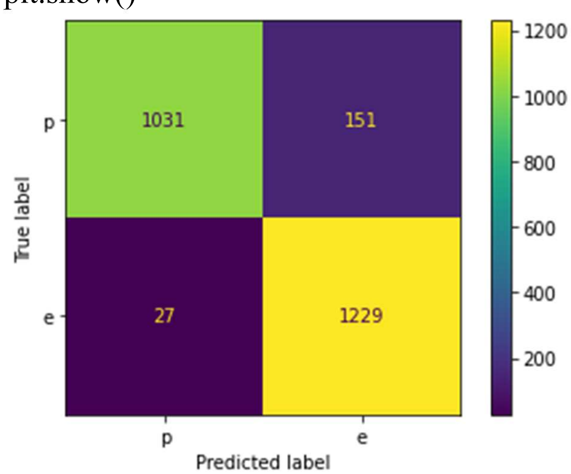
 # ROC Curve
y_pred_proba = gnb.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()



 # Decision Tree
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, y_train)
predicted_value = model.predict(X_test)
acc = accuracy_score(predicted_value, y_test)
tree_acc.append(acc)
print(f'Accuracy using Decision Tree: {acc}')

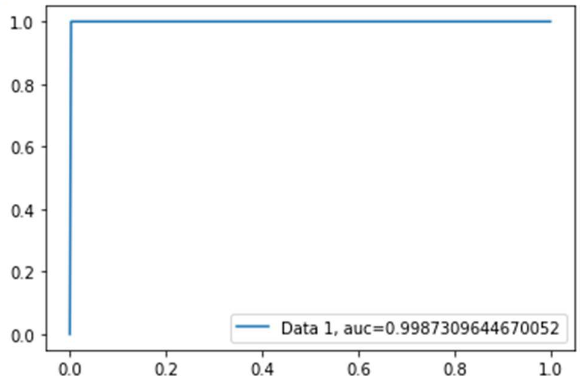
```
Accuracy using Decision Tree: 0.9987694831829368
```

tree_matrix = metrics.confusion_matrix(y_test, y_pred)
tree_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = tree_matrix, display_labels = ['p', 'e'])
tree_cm_display.plot()
plt.show()



 # ROC Curve

```
y_pred_proba = model.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



Customer Dataset:

 # Pre-processing

```
dataset.append('Mushroom')
df = pd.read_csv('/content/drive/MyDrive/Customer_Behaviour.csv')

for i in df.columns:
  print(i)
```

```
User ID
Gender
Age
EstimatedSalary
Purchased
```

```
df.drop('User ID', axis = 1)
df.replace(to_replace = 'Male', value = 0, inplace = True)
df.replace(to_replace =  'Female', value = 1, inplace = True)
```

 # Splitting into training and testing set

```
X = pd.DataFrame(df, columns = ['Gender', 'Age', 'EstimatedSalary'])
y = df['Purchased'].to_numpy()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 10242)
```

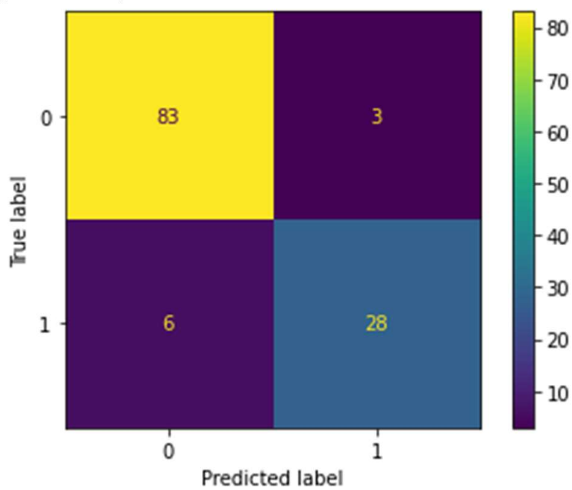
 # Gaussian Naïve Bayes

```
gnb = GaussianNB()
```

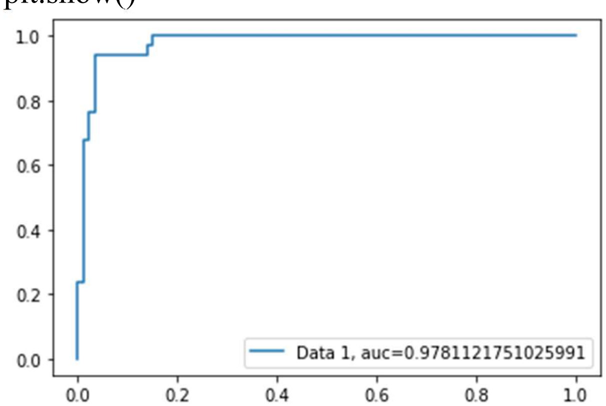
```
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
acc = accuracy_score(y_pred, y_test)
gnb_acc.append(acc)
print(f'Accuracy using Gaussian Naive Bayes: {acc}')
```

```
Accuracy using Gaussian Naive Bayes: 0.925
```

```
matrix = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = matrix,
display_labels = ['0', '1'])
cm_display.plot()
plt.show()
```



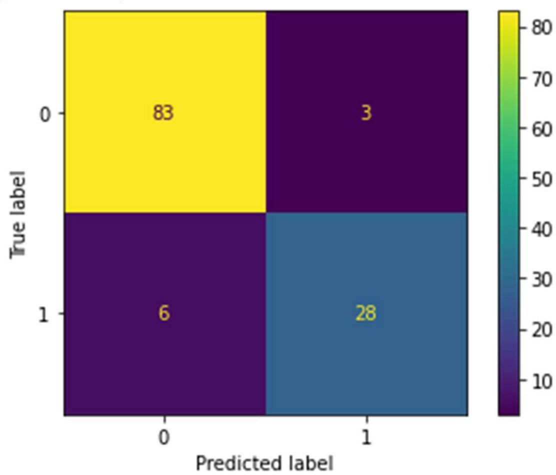

```
# ROC Curve
y_pred_proba = gnb.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()
```



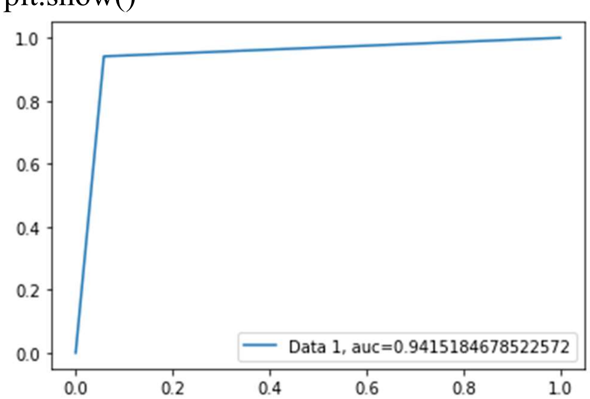
# Decision Tree
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, y_train)
predicted_value = model.predict(X_test)
acc = accuracy_score(predicted_value, y_test)
tree_acc.append(acc)
print(f'Accuracy using Decision Tree: {acc}')

```
Accuracy using Decision Tree: 0.9416666666666667
```

tree_matrix = metrics.confusion_matrix(y_test, y_pred)
tree_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = tree_matrix, display_labels = ['0', '1'])
tree_cm_display.plot()
plt.show()

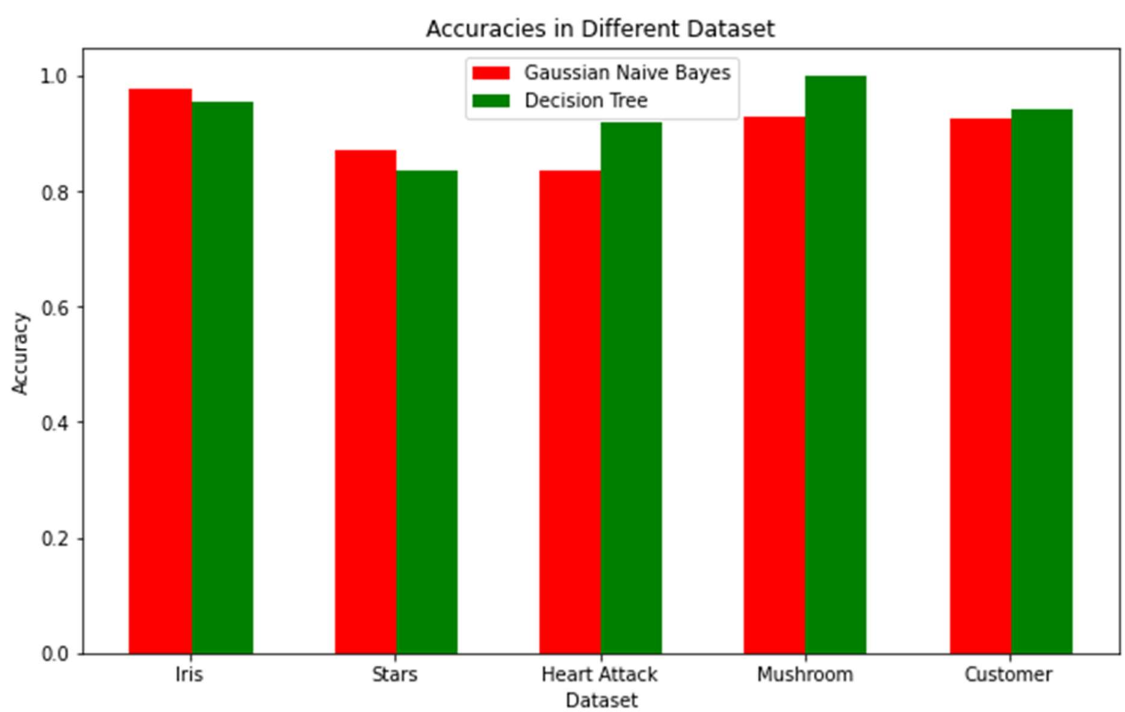


# ROC Curve
y_pred_proba = model.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label = "Data 1, auc=" + str(auc))
plt.legend(loc = 4)
plt.show()



Part B:

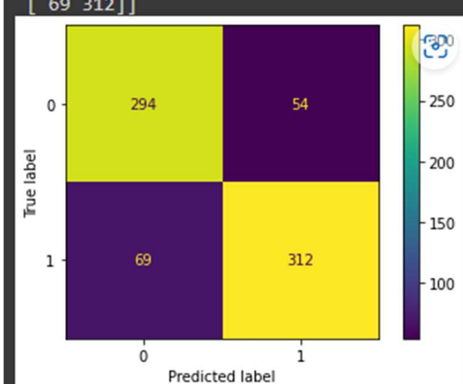
```
# Comparison in Different Datasets
N = 3
ind = np.arange(len(dataset[::1]))
width = 0.3

fig = plt.figure()
ax = fig.add_subplot(111)

rects1 = ax.bar(ind, gnb_acc, width, color = 'r')
rects2 = ax.bar(ind+width, tree_acc, width, color = 'g')

ax.set_ylabel('Accuracy')
ax.set_xlabel('Dataset')
ax.set_xticks(ind + width - 0.155)
ax.set_xticklabels(dataset)
ax.legend((rects1[0], rects2[0]), ('Gaussian Naive Bayes', 'Decision Tree') )

ax.set_title('Accuracies in Different Dataset')
fig.set_figwidth(10)
fig.set_figheight(6)

plt.show()
```



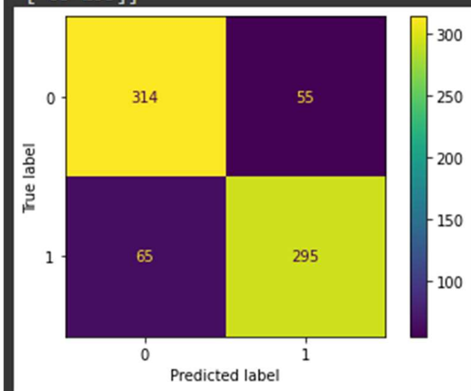
Part C:

```
# Choosing the Stars Dataset Decision Tree model

# K-Fold Cross Validation
final = 0
k_folds = KFold(n_splits = 10, shuffle = True, random_state = 2002)
for train_index, test_index in k_folds.split(X):
  X_train, X_test = X.iloc[train_index, :], X.iloc[test_index, :]
  y_train, y_test = y[train_index], y[test_index]
  rfc_model = tree.DecisionTreeClassifier()
  rfc_model.fit(X_train, y_train)
  test_preds = rfc_model.predict(X_test)
  test_accuracy = accuracy_score(y_test, test_preds)
  final += test_accuracy
  test_confusion_matrix = metrics.confusion_matrix(y_test, test_preds)
  print(f'Accuracy Score: {test_accuracy}')
  print(f'Confusion Matrix: \n{test_confusion_matrix}')
  k_fold_cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
test_confusion_matrix, display_labels = ['0', '1'])
  k_fold_cm_display.plot()
  plt.show()
  print('\n')

print(f'Average accuracy: {final / 10}')
```

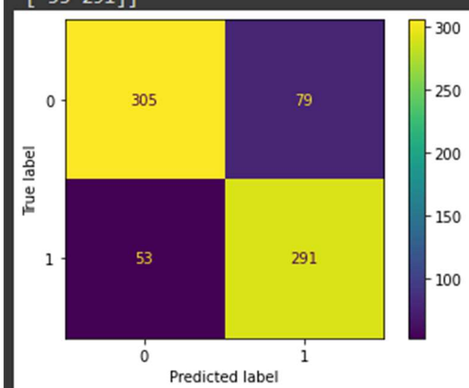
```
Accuracy Score: 0.831275720164609
Confusion Matrix:
[[294  54]
 [ 69 312]]
```



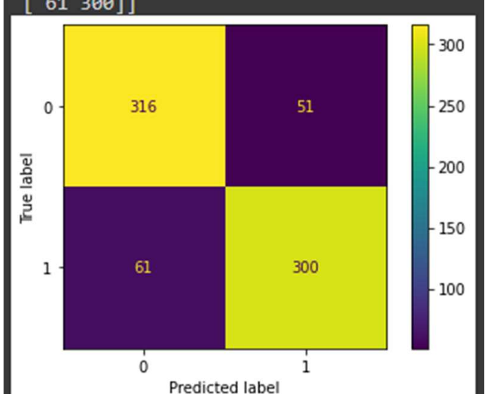
```
Accuracy Score: 0.8353909465020576
Confusion Matrix:
[[314  55]
 [ 65 295]]
```



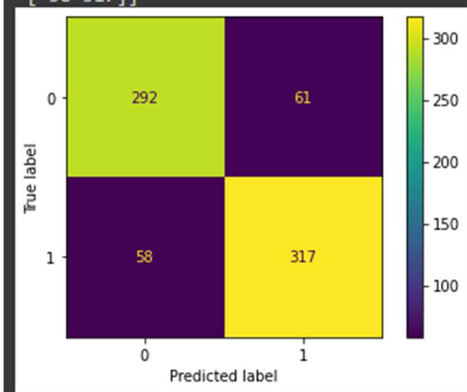

```
Accuracy Score: 0.8186813186813187
Confusion Matrix:
[[305  79]
 [ 53 291]]
```




```
Accuracy Score: 0.8461538461538461
Confusion Matrix:
[[316  51]
 [ 61 300]]
```



```
Accuracy Score: 0.8365384615384616
Confusion Matrix:
[[292  61]
 [ 58 317]]
```




```
Average accuracy: 0.8336080586080585
```

```
# Ensembling Methods

# Bagging Classifier
bagging = BaggingClassifier(base_estimator = model, n_estimators = 5, max_samples = 50, bootstrap = True)
bagging.fit(X_train, y_train)
print(f'Train score: {bagging.score(X_train, y_train)}')
print(f'Test score: {bagging.score(X_test, y_test)}')
```

```
Train score: 0.8853809196980096
Test score: 0.8763736263736264
```

```
# Adaboost Classifier
adaboost = AdaBoostClassifier(base_estimator = model, n_estimators = 5, learning_rate = 0.1, random_state = 25210)
adaboost.fit(X_train, y_train)
print(f"Train score: {adaboost.score(X_train, y_train)}")
print(f"Test score: {adaboost.score(X_test, y_test)}")
```

```
Train score: 1.0
Test score: 0.8475274725274725
```

```
clf = RandomForestClassifier(n_estimators = 100, max_features = "auto", random_state = 4627)
clf.fit(X_train, y_train)
print(f"Train score: {clf.score(X_train, y_train)}")
print(f"Test score: {clf.score(X_test, y_test)}")
```

```
Train score: 1.0
Test score: 0.885989010989011
```

```
# Comparing the scores
all_data = {'Classifier': ['Decision Treee', 'K-Fold', 'Bagging Classifier', 'Adaboost',
'Random Forest'],
            'Accuracy': scores}
print(tabulate(all_data, tablefmt = 'grid'))
```

```
+--------------------+----------+
| Decision Treee     | 0.835346 |
+--------------------+----------+
| K-Fold             | 0.837454 |
+--------------------+----------+
| Bagging Classifier | 0.876374 |
+--------------------+----------+
| Adaboost           | 0.847527 |
+--------------------+----------+
| Random Forest      | 0.885989 |
+--------------------+----------+
```

Conclusion: -

Implemented Gaussian Naïve Bayes and Decision Tree Classifier. Also learned K-Fold cross validation and different ensembling methods to improve the accuracy of a model.