Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	T. Y. B.Tech (Computer Engineering)
Course:	Data Mining and Warehouse Laboratory
Course Code:	DJ19CEL501
Experiment No.:	03

AIM: Implementation of Classification algorithm Using

1. Decision Tree ID3
2. Naïve Bayes algorithm

THEORY:

IMPLEMENTATION OF CLASSIFICATION ALGORITHM USING DECISION TREE ID3:

- Decision Tree ID3 is a machine learning algorithm that builds a decision tree for classification by selecting the best attributes to split the data based on information gain.
- It recursively divides the data into subsets until a stopping criterion is met.
- The resulting tree can be used to classify new instances based on their feature values.

IMPLEMENTATION OF CLASSIFICATION ALGORITHM USING NAÏVE BAYES:

- The Naïve Bayes algorithm is a probabilistic classifier that applies Bayes' theorem with the "naïve" assumption that features are conditionally independent.
- It calculates the probability of an instance belonging to a particular class based on the probabilities of its features.
- Naïve Bayes is simple, efficient, and works well with text classification and other applications where independence assumptions hold.

PROCEDURE:

IMPORT LIBRARIES:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.datasets import load_iris
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

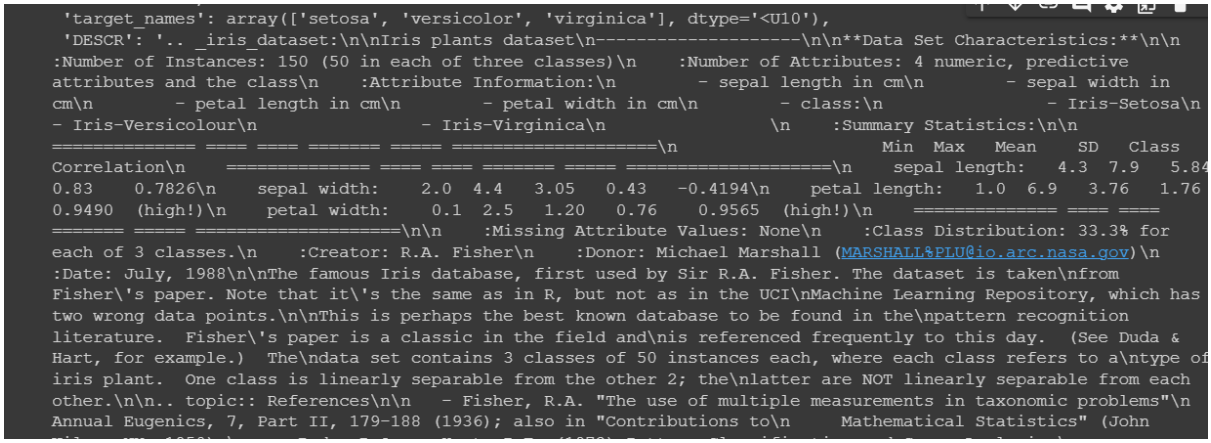
```
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier

from ucimlrepo import fetch_ucirepo  # For fetching UCI datasets
```

Part A: Applying Gaussian Naive Bayes and Decision Tree Classifier on Dataset

Dataset 1 - Iris Floris Dataset

```
df1 = load_iris()
df1
```

OUTPUT:



```
'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 150 (50 in each of three classes)\n    :Number of Attributes: 4 numeric, predictive
attributes and the class\n    :Attribute Information:\n        - sepal length in cm\n        - sepal width in
cm\n        - petal length in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa\n
- Iris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n
============== ==== ==== ======= ===== ====================\n                    Min  Max   Mean    SD   Class
Correlation\n    ============== ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84
0.83    0.7826\n    sepal width:    2.0  4.4   3.05    0.43   -0.4194\n    petal length:   1.0  6.9   3.76    1.76
0.9490  (high!)\n    petal width:    0.1  2.5   1.20    0.76    0.9565  (high!)\n    ============== ==== ====
======= ===== ====================\n\n    :Missing Attribute Values: None\n    :Class Distribution: 33.3% for
each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n
:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom
Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has
two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recognition
literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to this day.  (See Duda &
Hart, for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of
iris plant.  One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each
other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n
Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistics" (John
```

```
dataset1 = pd.DataFrame(df1.data)
dataset1.columns = df1.feature_names
dataset1['species'] = df1.target
dataset1
```

OUTPUT:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2

```
# independent and dependent features
x = dataset1.iloc[:,:-1]
y = dataset1.iloc[:,-1]
```

```
x
```

OUTPUT:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9

```
y
```

OUTPUT:

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int32
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.33, random_state=42)
# Gaussian Naive Bayes Classifier
treemodel = GaussianNB()
treemodel
```

OUTPUT:

```
GaussianNB
GaussianNB()
```

```
treemodel.fit(x_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
[14] # prediction
y_pred = treemodel.predict(x_test)
```

```
acc_nb1 = accuracy_score(y_pred, y_test)
acc_nb1
```

```
0.96
```

```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.93      0.93      0.93        15
           2       0.94      0.94      0.94        16

    accuracy                           0.96        50
   macro avg       0.96      0.96      0.96        50
weighted avg       0.96      0.96      0.96        50
```

```
[17] # DecisionTree Classifier
decisionTree = DecisionTreeClassifier()
decisionTree
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[18] decisionTree.fit(x_train, y_train)
```

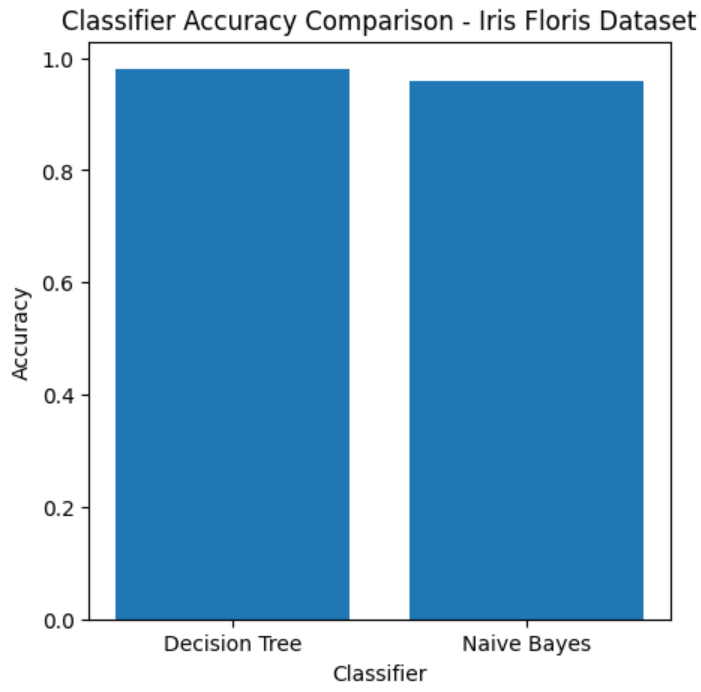
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
# prediction
y_pred = decisionTree.predict(x_test)
```

```
[21] acc_dt1 = accuracy_score(y_pred, y_test)
     acc_dt1

     0.98

     print(classification_report(y_pred, y_test))

                   precision    recall  f1-score   support

                0       1.00      1.00      1.00        19
                1       1.00      0.94      0.97        16
                2       0.94      1.00      0.97        15

         accuracy                           0.98        50
        macro avg       0.98      0.98      0.98        50
     weighted avg       0.98      0.98      0.98        50

                                           ✓ 0s  completed at 12:50
```

```
plt.figure(figsize=(5,5))
labels = ['Decision Tree', 'Naive Bayes']
accuracy = [acc_dt1, acc_nb1]
plt.bar(labels, accuracy)
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.title('Classifier Accuracy Comparison - Iris Floris Dataset')
plt.show()
```

OUTPUT:



Part B: Applying Gaussian Naive Bayes and Decision Tree Classifier on 4 Datasets

Dataset 2 - Wine Dataset

```
# fetch dataset
df2 = fetch_ucirepo(id=109)

# data (as pandas dataframes)
x = df2.data.features
y = df2.data.targets
```

```
[25] df2
```


```
'intro_paper': {'title': 'Comparative analysis of statistical pattern recognition methods in high dimensional
settings',
   'authors': 'S. Aeberhard, D. Coomans, O. Vel',
   'published_in': 'Pattern Recognition',
   'year': 1994,
   'url': 'https://www.semanticscholar.org/paper/83dc3e4030d7b9fbdbb4bde03ce12ab70ca10528',
   'doi': '10.1016/0031-3203(94)90145-7'},
  'additional_info': {'summary': 'These data are the results of a chemical analysis of wines grown in the same
region in Italy but derived from three different cultivars. The analysis determined the quantities of 13
constituents found in each of the three types of wines. \r\n\r\nI think that the initial data set had around 30
variables, but for some reason I only have the 13 dimensional version. I had a list of what the 30 or so
variables were, but a.)  I lost it, and b.), I would not know which 13 variables are included in the
set.\r\n\r\nThe attributes are (dontated by Riccardo Leardi, riclea@anchem.unige.it )\r\n1) Alcohol\r\n2) Malic
acid\r\n3) Ash\r\n4) Alcalinity of ash  \r\n5) Magnesium\r\n6) Total phenols\r\n7) Flavanoids\r\n8) Nonflavanoid
phenols\r\n9) Proanthocyanins\r\n10)Color intensity\r\n11)Hue\r\n12)OD280/OD315 of diluted wines\r\n13)Proline
\r\n\r\nIn a classification context, this is a well posed problem with "well behaved" class structures. A good
```


x

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoids	Nonflavanoid_phenols	Proanthocyanins
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82
...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35

y

	class
0	1
1	1
2	1
3	1
4	1
...	...
173	3
174	3
175	3

```
x.isnull().sum()
```

```
Alcohol                         0
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavanoids                      0
Nonflavanoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
OD280_OD315_of_diluted_wines    0
Proline                         0
dtype: int64
```

```
[29] y.isnull().sum()
```

```
class    0
dtype: int64
```

```
[30] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

```
# Gaussian Naive Bayes Classifier
treemodel = GaussianNB()
treemodel
```

```
GaussianNB()
```

```
[ ] treemodel.fit(x_train, y_train)
```

```
[32] treemodel.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning:
  y = column_or_1d(y, warn=True)
```

```
GaussianNB()
```

```
[33] # prediction
y_pred = treemodel.predict(x_test)
```

```
acc_nb2 = accuracy_score(y_pred, y_test)
acc_nb2
```

```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        20
           2       1.00      1.00      1.00        24
           3       1.00      1.00      1.00        15

    accuracy                           1.00        59
   macro avg       1.00      1.00      1.00        59
weighted avg       1.00      1.00      1.00        59
```

```
# DecisionTree Classifier
decisionTree = DecisionTreeClassifier()
decisionTree
```

```
DecisionTreeClassifier()
```

```
[37] decisionTree.fit(x_train, y_train)
```

```
DecisionTreeClassifier()
```

```
[38] # prediction
y_pred = decisionTree.predict(x_test)
```

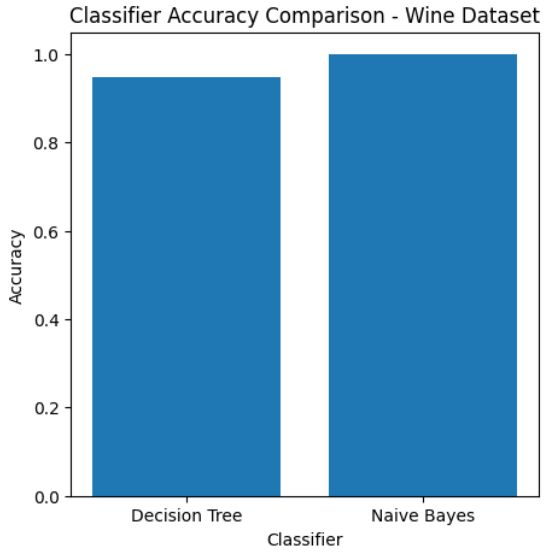
```
acc_dt2 = accuracy_score(y_pred, y_test)
acc_dt2
```

```
0.9491525423728814
```

```
[40] print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           1       0.90      1.00      0.95        18
           2       1.00      0.89      0.94        27
           3       0.93      1.00      0.97        14

    accuracy                           0.95        59
   macro avg       0.94      0.96      0.95        59
weighted avg       0.95      0.95      0.95        59
```

```
plt.figure(figsize=(5,5))
labels = ['Decision Tree', 'Naive Bayes']
accuracy = [acc_dt2, acc_nb2]
plt.bar(labels, accuracy)
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.title('Classifier Accuracy Comparison - Wine Dataset')
plt.show()
```



Dataset 4 - Rice Evaluation

```
[ ] # fetch dataset
    df4 = fetch_ucirepo(id=545)

    # data (as pandas dataframes)
    x = df4.data.features
    y = df4.data.targets
```

```
[ ] df4
```

```
{'data': {'ids': None,
  'features':        Area   Perimeter  Major_Axis_Length  Minor_Axis_Length  Eccentricity  \
  0     15231  525.578979         229.749878          85.093788      0.928882
  1     14656  494.311005         206.020065          91.730972      0.895405
  2     14634  501.122009         214.106781          87.768288      0.912118
  3     13176  458.342987         193.337387          87.448395      0.891861
  4     14688  507.166992         211.743378          89.312454      0.906691
  ...     ...         ...                ...                ...           ...
  3805  11441  415.858002         170.486771          85.756592      0.864280
  3806  11625  421.390015         167.714798          89.462570      0.845850
  3807  12437  442.498993         183.572922          86.801979      0.881144
  3808   9882  392.296997         161.193985          78.210480      0.874406
  3809  11434  404.709992         161.079269          90.868195      0.825692

        Convex_Area    Extent
  0           15617  0.572896
```

x

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent
0	15231	525.578979	229.749878	85.093788	0.928882	15617	0.572896
1	14656	494.311005	206.020065	91.730972	0.895405	15072	0.615436
2	14634	501.122009	214.106781	87.768288	0.912118	14954	0.693259
3	13176	458.342987	193.337387	87.448395	0.891861	13368	0.640669
4	14688	507.166992	211.743378	89.312454	0.906691	15262	0.646024
...	...	...	...	...	...	...	...
3805	11441	415.858002	170.486771	85.756592	0.864280	11628	0.681012
3806	11625	421.390015	167.714798	89.462570	0.845850	11904	0.694279
3807	12437	442.498993	183.572922	86.801979	0.881144	12645	0.626739
3808	9882	392.296997	161.193985	78.210480	0.874406	10097	0.659064
3809	11434	404.709992	161.079269	90.868195	0.825692	11591	0.802949

```
[ ] y
```

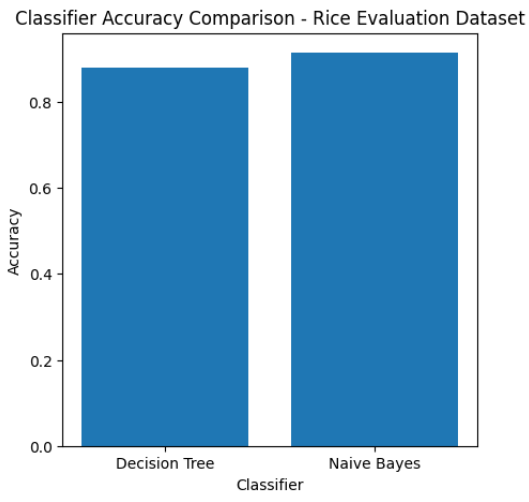
	Class
0	Cammeo
1	Cammeo
2	Cammeo
3	Cammeo
4	Cammeo
...	...
3805	Osmancik
3806	Osmancik
3807	Osmancik
3808	Osmancik
3809	Osmancik

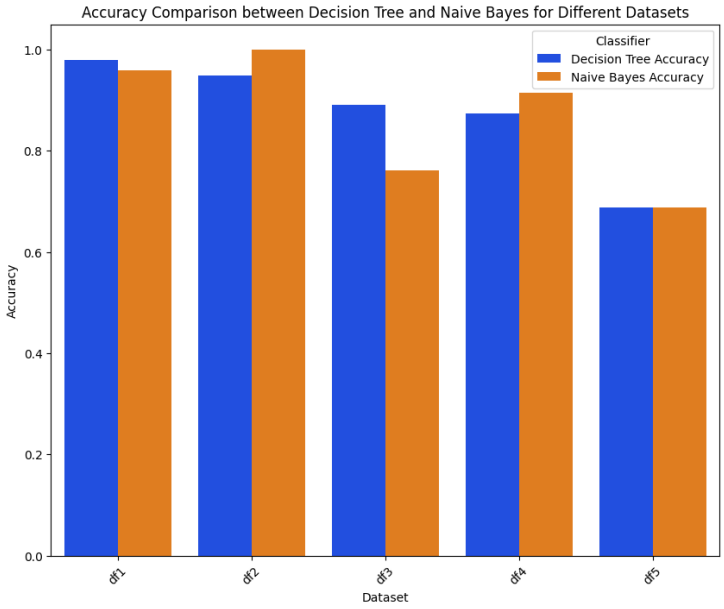
```
[46] x.isnull().sum()

Area                0
Perimeter           0
Major_Axis_Length   0
Minor_Axis_Length   0
Eccentricity        0
Convex_Area         0
Extent              0
dtype: int64

[47] y.isnull().sum()

Class    0
dtype: int64

[48] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

# Gaussian Naive Bayes Classifier
treemodel = GaussianNB()
treemodel

GaussianNB()
```

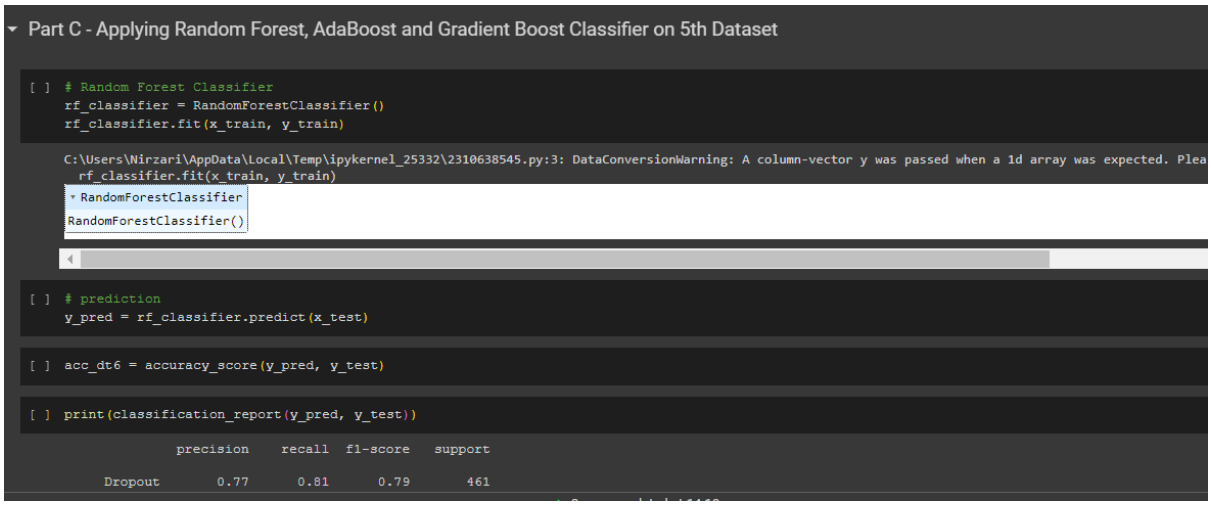
```
[50] treemodel.fit(x_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was
  y = column_or_1d(y, warn=True)
GaussianNB()

[51] # prediction
y_pred = treemodel.predict(x_test)

[53] acc_nb4 = accuracy_score(y_pred, y_test)
acc_nb4

0.9141494435612083

print(classification_report(y_pred, y_test))

              precision    recall  f1-score   support

      Cammeo       0.89      0.92      0.90       546
    Osmancik       0.94      0.91      0.92       712

    accuracy                           0.91      1258
   macro avg       0.91      0.91      0.91      1258
weighted avg       0.92      0.91      0.91      1258
```

```
[55] # DecisionTree Classifier
     decisionTree = DecisionTreeClassifier()
     decisionTree

     DecisionTreeClassifier()

[56] decisionTree.fit(x_train, y_train)

     DecisionTreeClassifier()

[57] # prediction
     y_pred = decisionTree.predict(x_test)

     acc_dt4 = accuracy_score(y_pred, y_test)
     acc_dt4

     0.8799682034976153
```

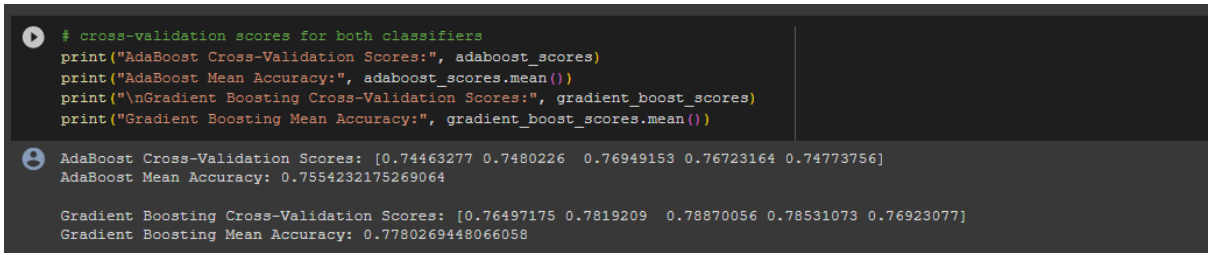
```
[59] print(classification_report(y_pred, y_test))

                precision    recall  f1-score   support

       Cammeo       0.84      0.89      0.86       535
     Osmancik       0.91      0.87      0.89       723

     accuracy                           0.88      1258
    macro avg       0.88      0.88      0.88      1258
 weighted avg       0.88      0.88      0.88      1258

     plt.figure(figsize=(5,5))
     labels = ['Decision Tree', 'Naive Bayes']
     accuracy = [acc_dt4, acc_nb4]
     plt.bar(labels, accuracy)
     plt.xlabel('Classifier')
     plt.ylabel('Accuracy')
     plt.title('Classifier Accuracy Comparison - Rice Evaluation Dataset')
     plt.show()
```







```
Part C - Applying Random Forest, AdaBoost and Gradient Boost Classifier on 5th Dataset

[ ] # Random Forest Classifier
    rf_classifier = RandomForestClassifier()
    rf_classifier.fit(x_train, y_train)

    C:\Users\Nirzari\AppData\Local\Temp\ipykernel_25332\2310638545.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Plea
      rf_classifier.fit(x_train, y_train)
    ▾ RandomForestClassifier
    RandomForestClassifier()

[ ] # prediction
    y_pred = rf_classifier.predict(x_test)

[ ] acc_dt6 = accuracy_score(y_pred, y_test)

[ ] print(classification_report(y_pred, y_test))

                  precision    recall  f1-score   support

         Dropout       0.77      0.81      0.79       461
```



```
# cross-validation scores for both classifiers
print("AdaBoost Cross-Validation Scores:", adaboost_scores)
print("AdaBoost Mean Accuracy:", adaboost_scores.mean())
print("\nGradient Boosting Cross-Validation Scores:", gradient_boost_scores)
print("Gradient Boosting Mean Accuracy:", gradient_boost_scores.mean())

AdaBoost Cross-Validation Scores: [0.74463277 0.7480226  0.76949153 0.76723164 0.74773756]
AdaBoost Mean Accuracy: 0.7554232175269064

Gradient Boosting Cross-Validation Scores: [0.76497175 0.7819209  0.78870056 0.78531073 0.76923077]
Gradient Boosting Mean Accuracy: 0.7780269448066058
```

CONCLUSION:

- In conclusion, the implementation of classification algorithms using Decision Tree ID3, and Naïve Bayes has provided effective tools for making predictions and categorizing data.
- Decision Tree ID3 creates a tree-like structure to make decisions, while Naïve Bayes leverages probability and independence assumptions.
- These methods offer versatile options for classification tasks in machine learning.