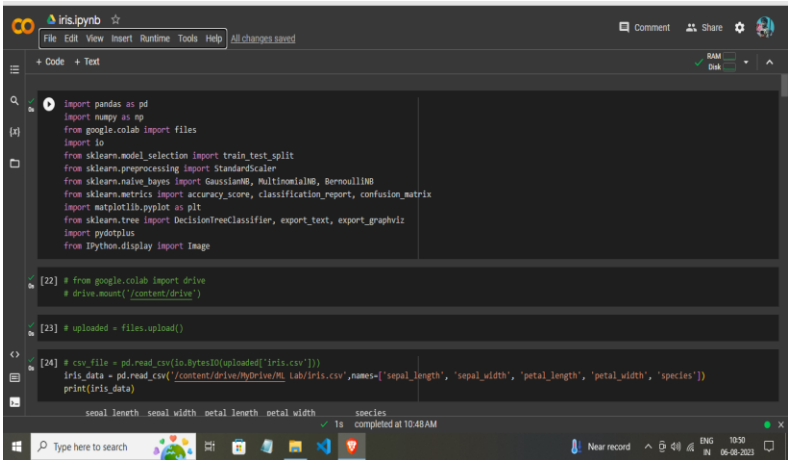
ML LAB ASSIGNMENT 1
NAME - KUNTAL DAS
ROLL - 002011001032

IRIS_PLANTS : →



```
import pandas as pd
import numpy as np
from google.colab import files
import io
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, export_text, export_graphviz
import pydotplus
from IPython.display import Image
```

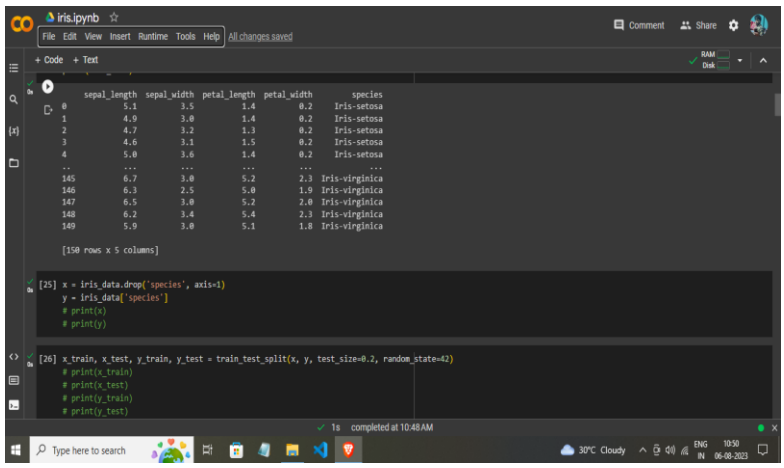
```
[22] # from google.colab import drive
     # drive.mount('/content/drive')
```

```
[23] # uploaded = files.upload()
```

```
[24] # csv_file = pd.read_csv(io.BytesIO(uploaded['iris.csv']))
     iris_data = pd.read_csv('/content/drive/MyDrive/ML Lab/Iris.csv',names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])
     print(iris_data)
```

```
     sepal_length  sepal_width  petal_length  petal_width         species
```

1s completed at 10:48 AM

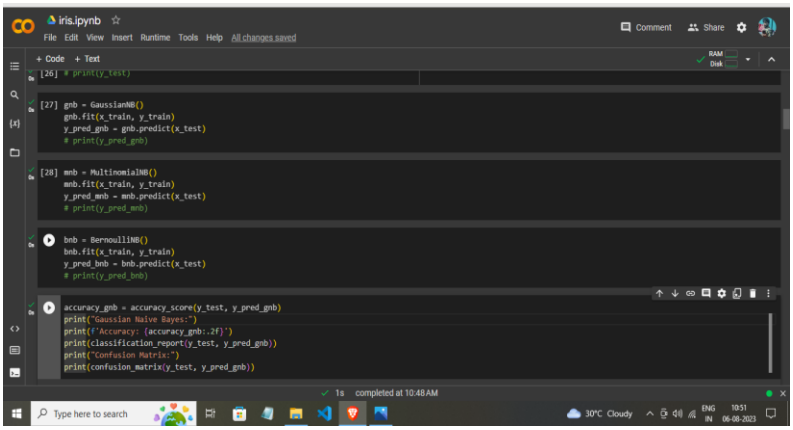


```
     sepal_length  sepal_width  petal_length  petal_width         species
0             5.1          3.5           1.4          0.2     Iris-setosa
1             4.9          3.0           1.4          0.2     Iris-setosa
2             4.7          3.2           1.3          0.2     Iris-setosa
3             4.6          3.1           1.5          0.2     Iris-setosa
4             5.0          3.6           1.4          0.2     Iris-setosa
..            ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

```
[25] x = iris_data.drop('species', axis=1)
     y = iris_data['species']
     # print(x)
     # print(y)
```

```
[26] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
     # print(x_train)
     # print(x_test)
     # print(y_train)
     # print(y_test)
```

1s completed at 10:48 AM



```
[26] # print(y_test)
```

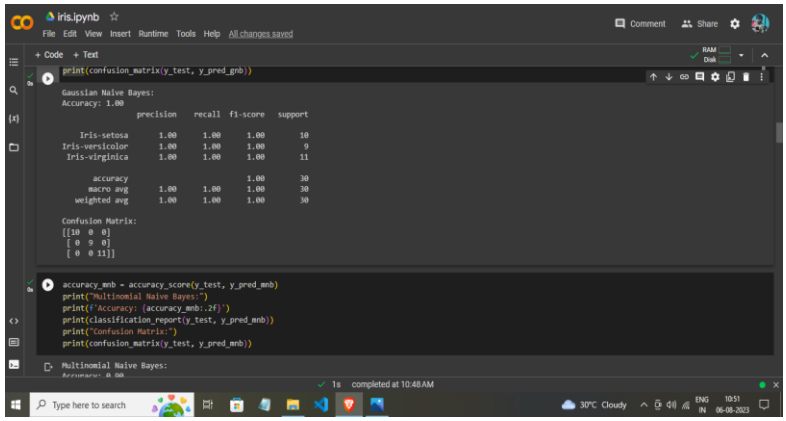
```
[27] gnb = GaussianNB()
     gnb.fit(x_train, y_train)
     y_pred_gnb = gnb.predict(x_test)
     # print(y_pred_gnb)
```

```
[28] mnb = MultinomialNB()
     mnb.fit(x_train, y_train)
     y_pred_mnb = mnb.predict(x_test)
     # print(y_pred_mnb)
```

```
     bnb = BernoulliNB()
     bnb.fit(x_train, y_train)
     y_pred_bnb = bnb.predict(x_test)
     # print(y_pred_bnb)
```

```
     accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
     print("Gaussian Naive Bayes:")
     print(f"Accuracy: {accuracy_gnb:.2f}")
     print(classification_report(y_test, y_pred_gnb))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_gnb))
```

1s completed at 10:48 AM



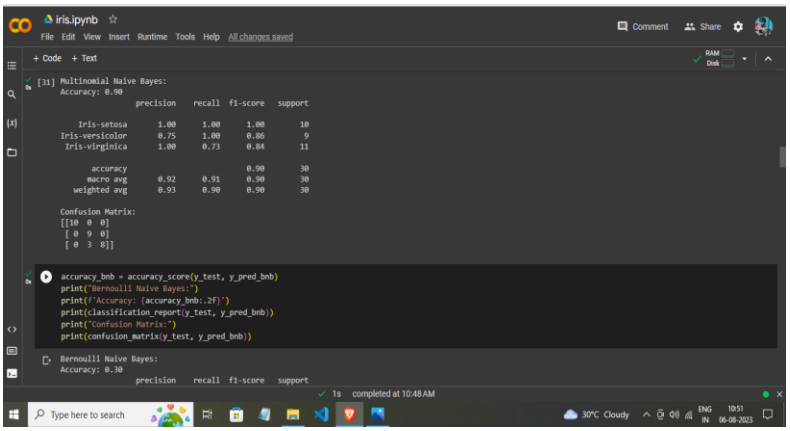
```
print(confusion_matrix(y_test, y_pred_gnb))
```

```
Gaussian Naive Bayes:
Accuracy: 1.00
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

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

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
     accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
     print("Multinomial Naive Bayes:")
     print(f'Accuracy: {accuracy_mnb:.2f}')
     print(classification_report(y_test, y_pred_mnb))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_mnb))
```

```
Multinomial Naive Bayes:
```

1s completed at 10:48 AM

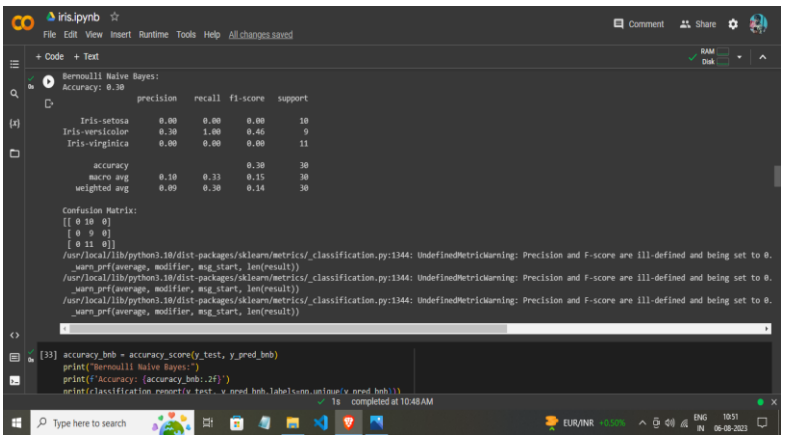


```
[31] Multinomial Naive Bayes:
     Accuracy: 0.90
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        10
 Iris-versicolor       0.75      1.00      0.86         9
  Iris-virginica       1.00      0.73      0.84        11

        accuracy                           0.90        30
       macro avg       0.92      0.91      0.90        30
    weighted avg       0.93      0.90      0.90        30

    Confusion Matrix:
    [[10  0  0]
     [ 0  9  0]
     [ 0  3  8]]
```

```
     accuracy_bnb = accuracy_score(y_test, y_pred_bnb)
     print("Bernoulli Naive Bayes:")
     print(f"Accuracy: {accuracy_bnb:.2f}")
     print(classification_report(y_test, y_pred_bnb))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_bnb))
```

```
     Bernoulli Naive Bayes:
     Accuracy: 0.30
                   precision    recall  f1-score   support
```

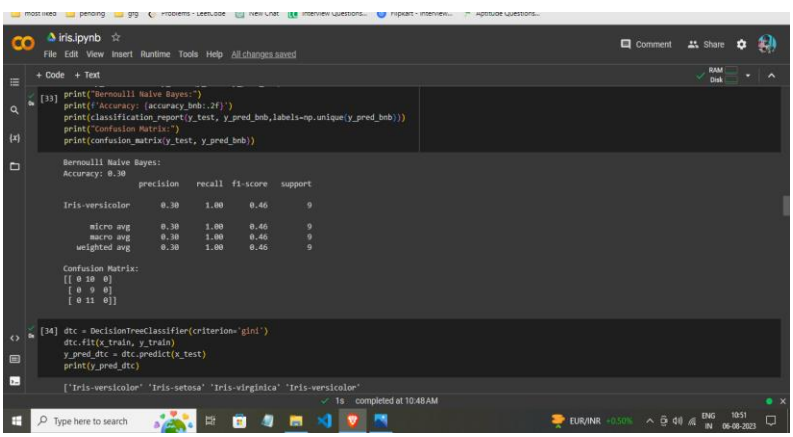
1s completed at 10:48 AM



```
Bernoulli Naive Bayes:
Accuracy: 0.30
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00        10
Iris-versicolor       0.30      1.00      0.46         9
 Iris-virginica       0.00      0.00      0.00        11

       accuracy                           0.30        30
      macro avg       0.10      0.33      0.15        30
   weighted avg       0.09      0.30      0.14        30

Confusion Matrix:
[[ 0 10  0]
 [ 0  9  0]
 [ 0 11  0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[33] accuracy_bnb = accuracy_score(y_test, y_pred_bnb)
     print("Bernoulli Naive Bayes:")
     print(f"Accuracy: {accuracy_bnb:.2f}")
     print(classification_report(y_test, y_pred_bnb,labels=np.unique(y_pred_bnb)))
```

1s completed at 10:48 AM



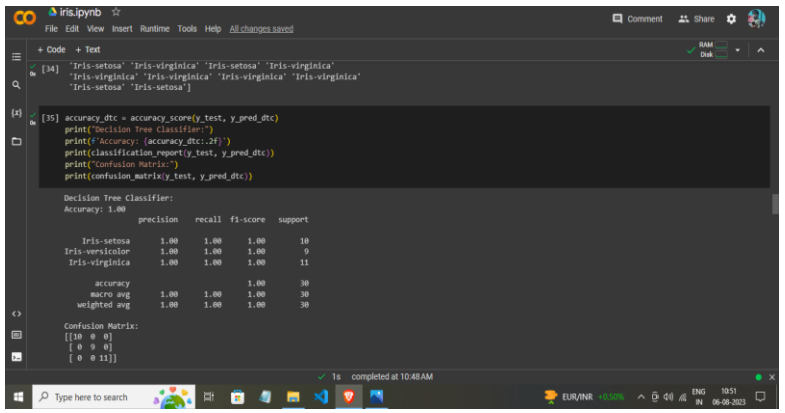
```
[33] print("Bernoulli Naive Bayes:")
     print(f"Accuracy: {accuracy_bnb:.2f}")
     print(classification_report(y_test, y_pred_bnb,labels=np.unique(y_pred_bnb)))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_bnb))
```

```
     Bernoulli Naive Bayes:
     Accuracy: 0.30
                   precision    recall  f1-score   support

     Iris-versicolor       0.30      1.00      0.46         9

           micro avg       0.30      1.00      0.46         9
           macro avg       0.30      1.00      0.46         9
        weighted avg       0.30      1.00      0.46         9

     Confusion Matrix:
     [[ 0 10  0]
      [ 0  9  0]
      [ 0 11  0]]
```

```
[34] dtc = DecisionTreeClassifier(criterion='gini')
     dtc.fit(x_train, y_train)
     y_pred_dtc = dtc.predict(x_test)
     print(y_pred_dtc)
```

```
     ['Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
```

1s completed at 10:48 AM



```
[34] 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
     'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
     'Iris-setosa' 'Iris-setosa']
```

```
[35] accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
     print("Decision Tree Classifier:")
     print(f'Accuracy: {accuracy_dtc:.2f}')
     print(classification_report(y_test, y_pred_dtc))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_dtc))
```

```
Decision Tree Classifier:
Accuracy: 1.00
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

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

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

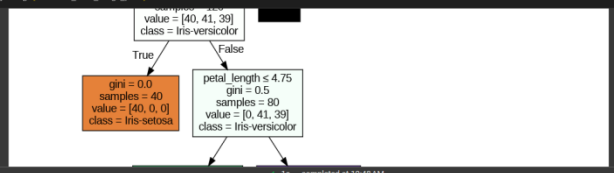
1s completed at 10:48 AM

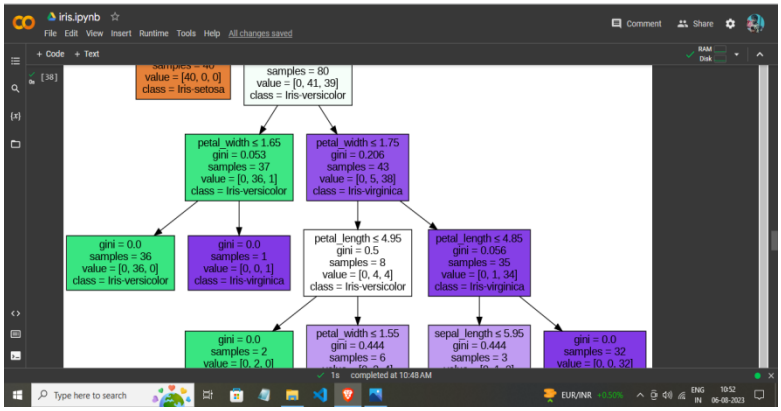
```
dot_data = export_graphviz(dtc, out_file=None, feature_names=x.columns, class_names=y.unique(),
                           filled=True, rounded=False, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
# print(graph)
```

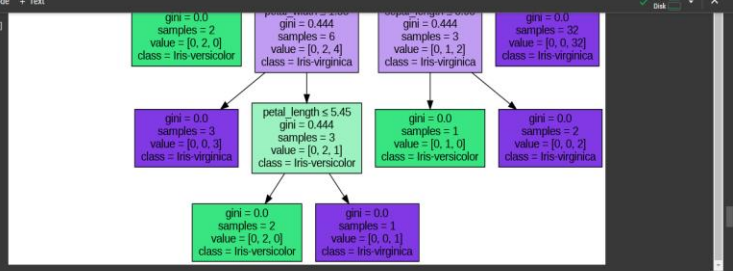
```
decision_tree_image_path = "iris_decision_tree.png"
graph.write_png(decision_tree_image_path)
```

```
True
```

```
Image(decision_tree_image_path)
```





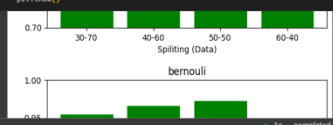



```
import numpy as np
import matplotlib.pyplot as plt

iris_data = pd.read_csv('/content/drive/MyDrive/ML Lab/iris.csv',names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])
```

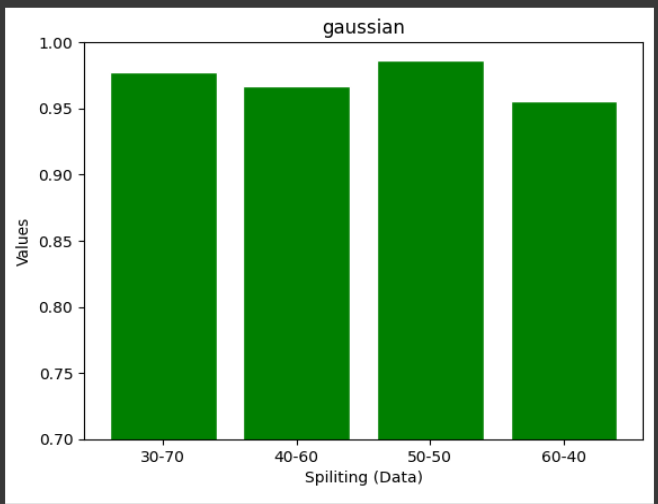
```
import numpy as np
import matplotlib.pyplot as plt

iris_data = pd.read_csv('/content/drive/MyDrive/ML Lab/iris.csv',names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])
# print(iris_data)

X = iris_data.drop('species', axis=1)
y = iris_data['species']

bernouli=[]
gaussian=[]
multinomial=[]
D_t=[]

for i in range(3,7):
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i/10, random_state=42)

  gnb = GaussianNB()
  gnb.fit(X_train, y_train)
  y_pred_gnb = gnb.predict(X_test)
  accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
  gaussian.append(accuracy_gnb)

  mnb = MultinomialNB(alpha=1.2,fit_prior=True,class_prior=None)
  mnb.fit(X_train, y_train)
  y_pred_mnb = mnb.predict(X_test)
```

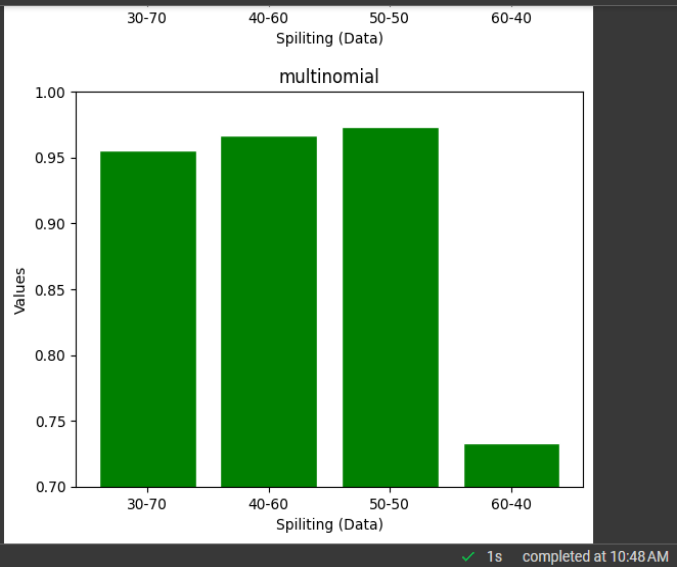
```
y_pred_gnb = gnb.predict(X_test)
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
gaussian.append(accuracy_gnb)

mnb = MultinomialNB(alpha=1.2,fit_prior=True,class_prior=None)
mnb.fit(X_train, y_train)
y_pred_mnb = mnb.predict(X_test)
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
multinomial.append(accuracy_mnb)

bnb = BernoulliNB(alpha=1.2,binarize=1.7,fit_prior=True,class_prior=None)
bnb.fit(X_train, y_train)
y_pred_bnb = bnb.predict(X_test)
accuracy_bnb = accuracy_score(y_test, y_pred_mnb)
bernouli.append(accuracy_bnb)

dtc = DecisionTreeClassifier(criterion='entropy')  # You can also use 'entropy' for criterion
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)
accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
D_t.append(accuracy_dtc)


bins = ["30-70","40-60","50-50","60-40"]
names=["gaussian","multinomial","bernouli","Decission_tree"]
```

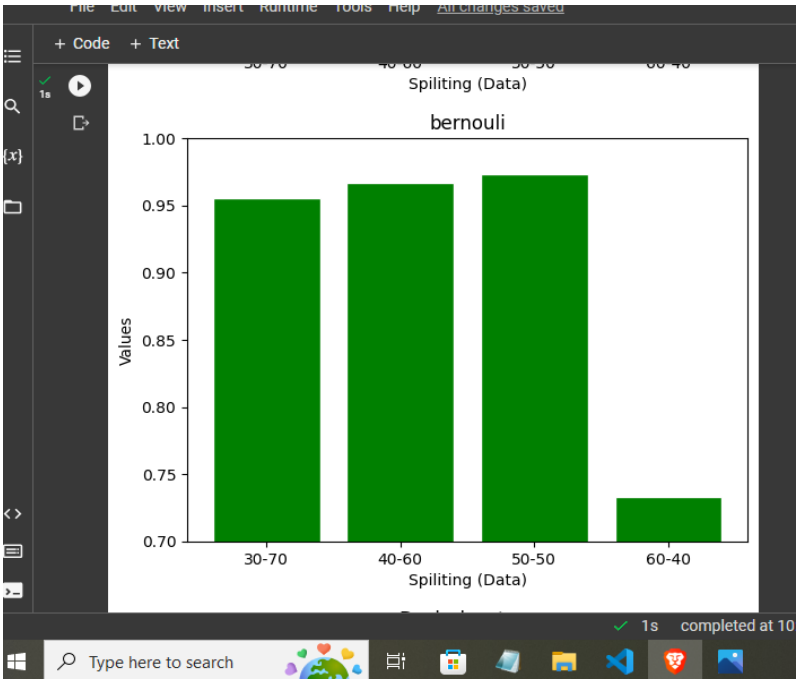
```
names=["gaussian","multinomial","bernouli","Decission_tree"]
total=[gaussian,multinomial,bernouli,D_t]


for i in range(0,4):
  plt.bar(bins, total[i], color='green', edgecolor='white')

  plt.ylim(0.7, 1)


  # Add axis labels and a title
  plt.xlabel('Spiliting (Data)')
  plt.ylabel('Values')
  plt.title(names[i])

  # Show the plot
  plt.show()
```



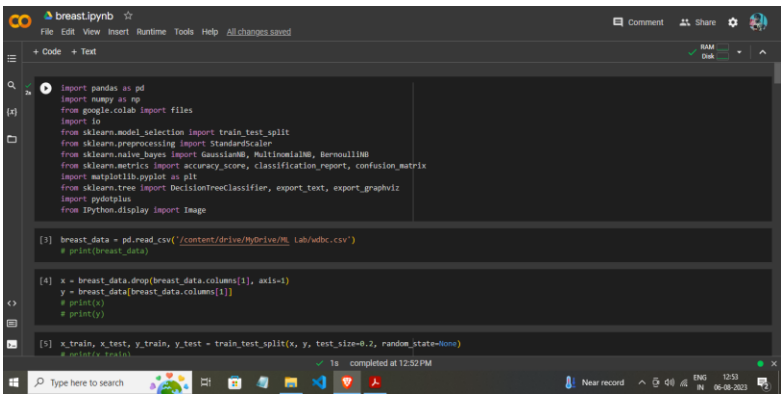

```
plt.show()
```

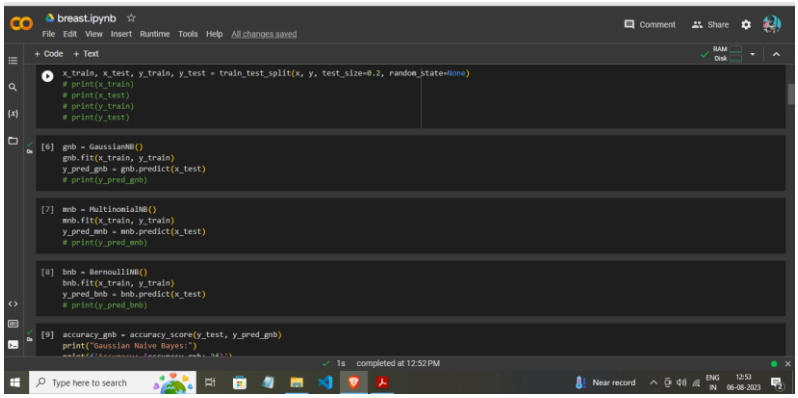


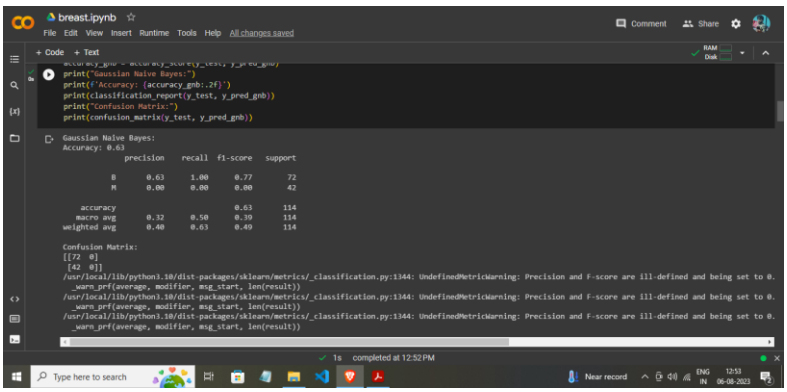


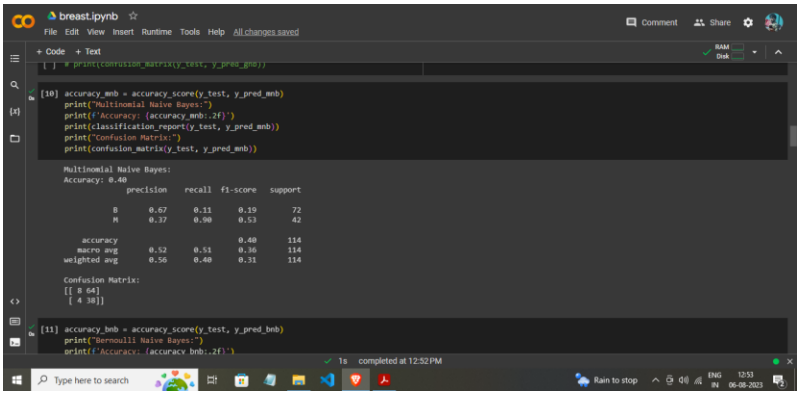


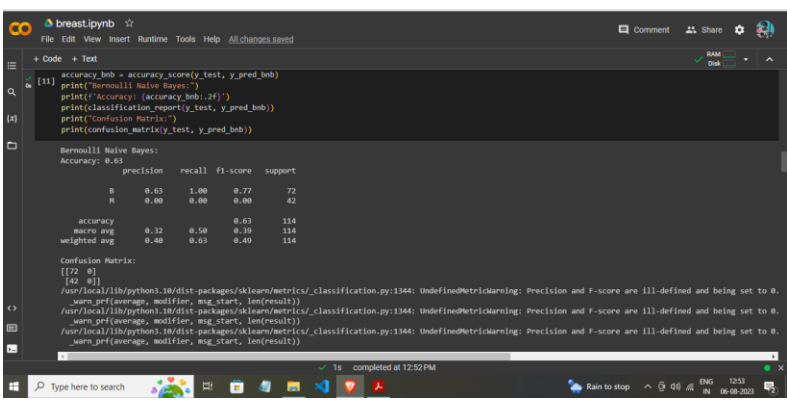
BREAST_CANCER : →

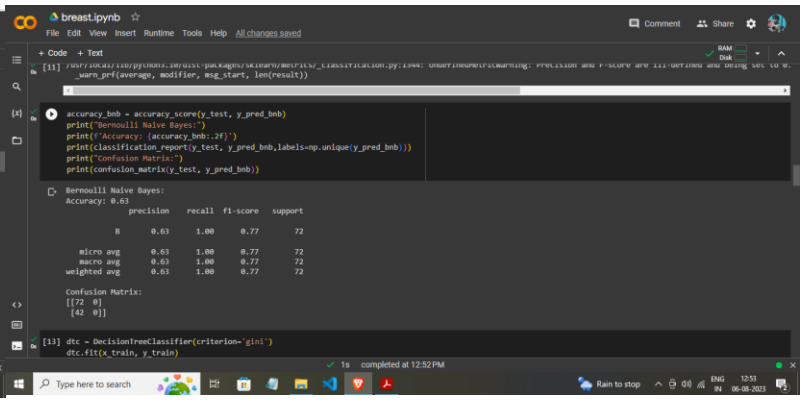






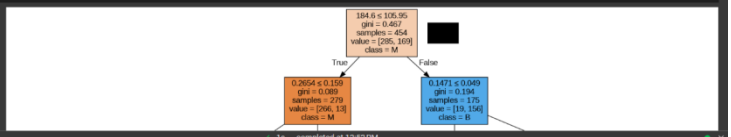




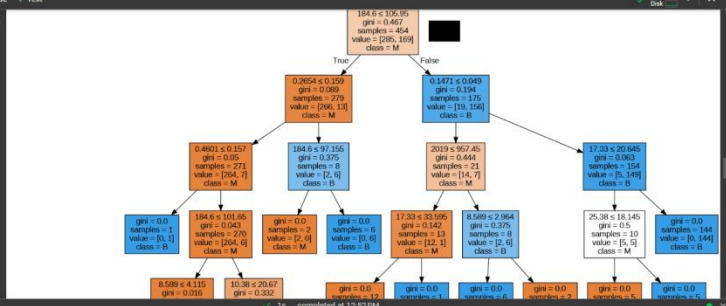


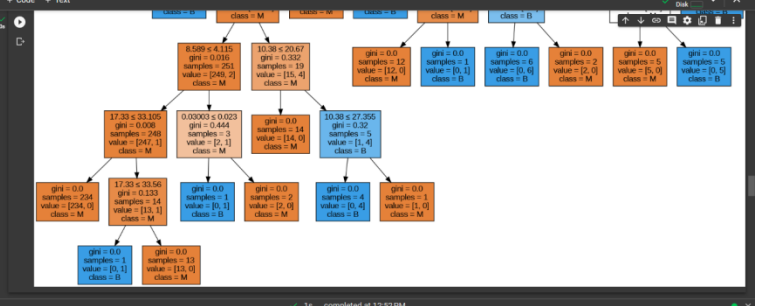
```
[13] dtc = DecisionTreeClassifier(criterion='gini')
     dtc.fit(x_train, y_train)
     y_pred_dtc = dtc.predict(x_test)

[14] accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
     print("Decision Tree Classifier:")
     print(f'Accuracy: {accuracy_dtc:.2f}')
     print(classification_report(y_test, y_pred_dtc))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred_dtc))

     Decision Tree Classifier:
     Accuracy: 0.90
                   precision    recall  f1-score   support

                B       0.93      0.92      0.92        72
                M       0.86      0.88      0.87        42

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

     Confusion Matrix:
     [[66  6]
      [ 5 37]]
```

```
      [ 5 37]]

[17] dot_data = export_graphviz(dtc, out_file=None, feature_names=x.columns, class_names=y.unique(),
                                filled=True, rounded=False, special_characters=True)
     graph = pydotplus.graph_from_dot_data(dot_data)

[18] decision_tree_image_path = "breast_tree.png"
     graph.write_png(decision_tree_image_path)

     True

[19] Image(decision_tree_image_path)
```



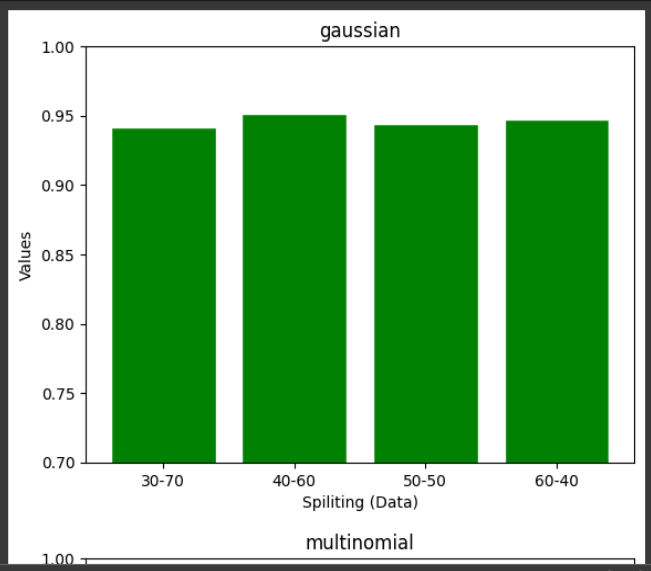

```
[19]
```

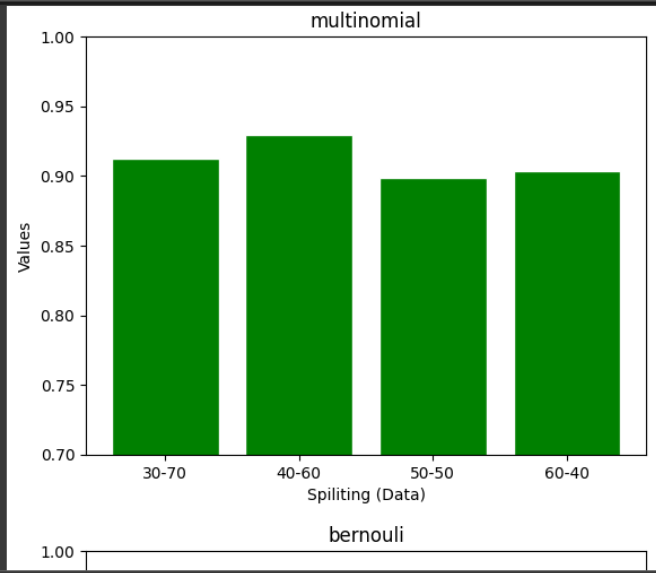


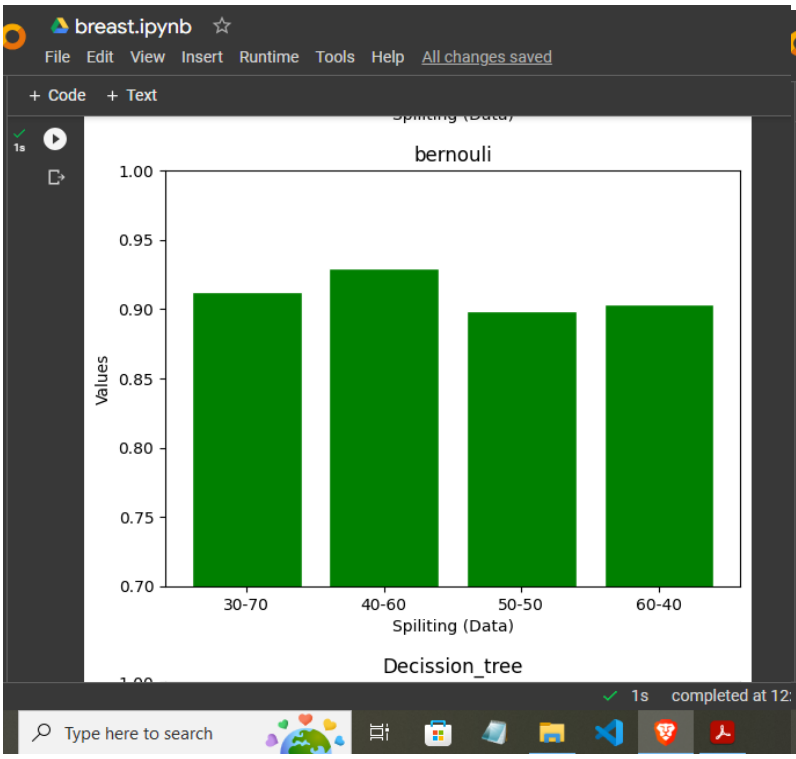


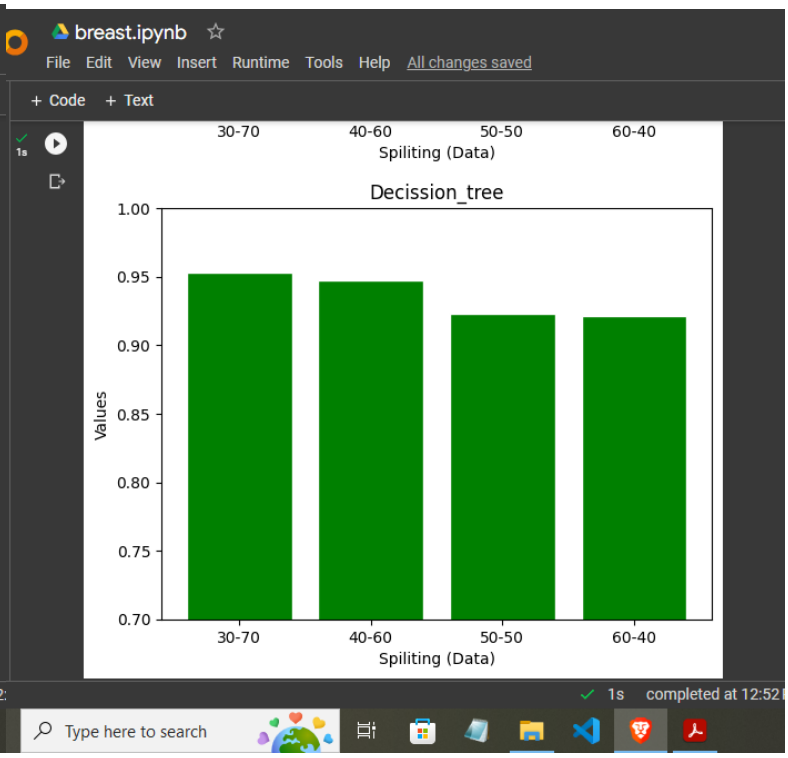

```
columns=[]
for i in range(0,32):
  columns.append(i)
breast_data = pd.read_csv('/content/drive/MyDrive/ML Lab/wdbc.csv',names=columns)
# print(breast_data)

x = breast_data.drop(breast_data.columns[[0,1]], axis=1)
y = breast_data[breast_data.columns[1]]

bernouli=[]
gaussian=[]
multinomial=[]
D_t=[]

for i in range(3,7):
  X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=i/10, random_state=42)

  gnb = GaussianNB()
  gnb.fit(X_train, y_train)
  y_pred_gnb = gnb.predict(X_test)
  accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
  gaussian.append(accuracy_gnb)

  mnb = MultinomialNB(alpha=1.2,fit_prior=True,class_prior=None)
  mnb.fit(X_train, y_train)
  y_pred_mnb = mnb.predict(X_test)
```

```
[22]  y_pred_mnb = mnb.predict(X_test)
      accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
      multinomial.append(accuracy_mnb)

      bnb = BernoulliNB(alpha=1.2,binarize=1.7,fit_prior=True,class_prior=None)
      bnb.fit(X_train, y_train)
      y_pred_bnb = bnb.predict(X_test)
      accuracy_bnb = accuracy_score(y_test, y_pred_mnb)
      bernouli.append(accuracy_bnb)

      dtc = DecisionTreeClassifier(criterion='entropy')
      dtc.fit(X_train, y_train)
      y_pred_dtc = dtc.predict(X_test)
      accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
      D_t.append(accuracy_dtc)

    bins = ["30-70","40-60","50-50","60-40"]
    names=["gaussian","multinomial","bernouli","Decission_tree"]
    total=[gaussian,multinomial,bernouli,D_t]

    for i in range(0,4):
      plt.bar(bins, total[i], color='green', edgecolor='white')
      plt.ylim(0.7, 1)

      # Add axis labels and a title
      plt.xlabel('Spiliting (Data)')
```

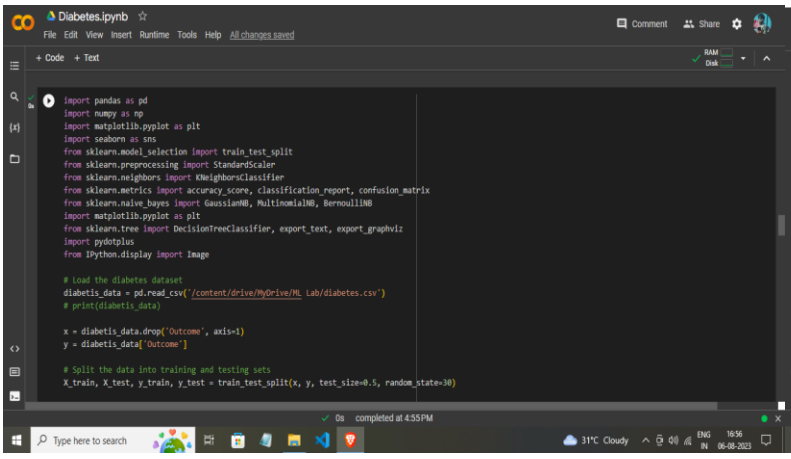


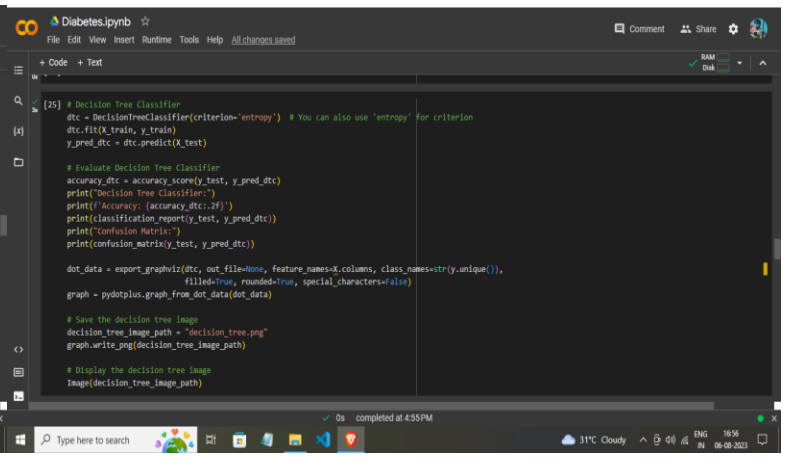


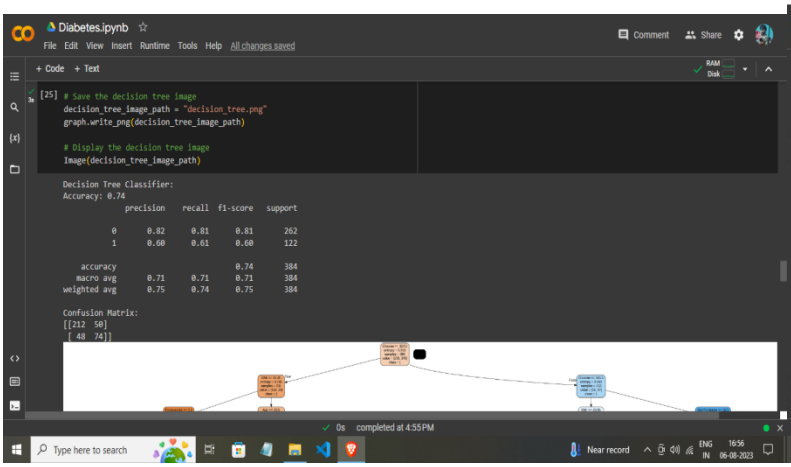

breast.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
bernouli
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Values
30-70
40-60
50-50
60-40
Spiliting (Data)
Decission_tree
1s completed at 12:
Type here to search

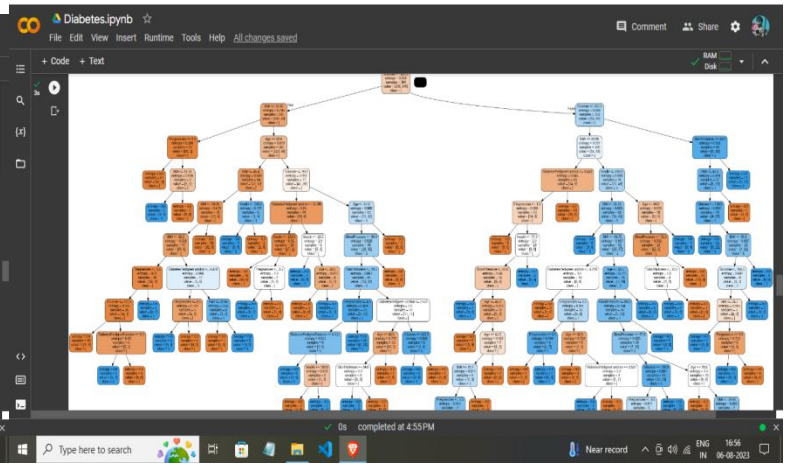

breast.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
30-70
40-60
50-50
60-40
Spiliting (Data)
Decission_tree
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Values
30-70
40-60
50-50
60-40
Spiliting (Data)
1s completed at 12:52
Type here to search

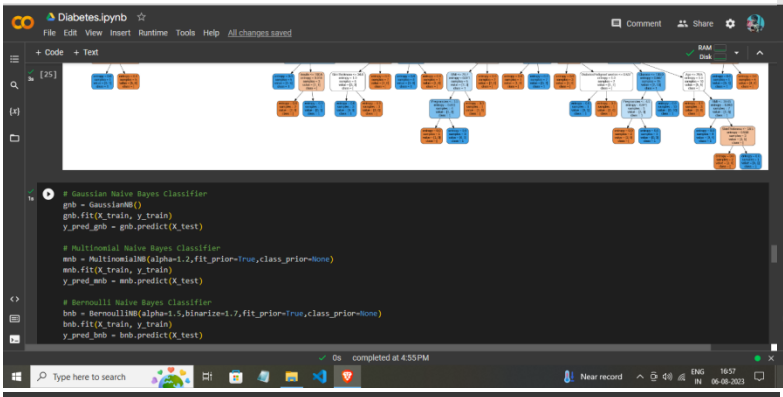
DIABETES : →

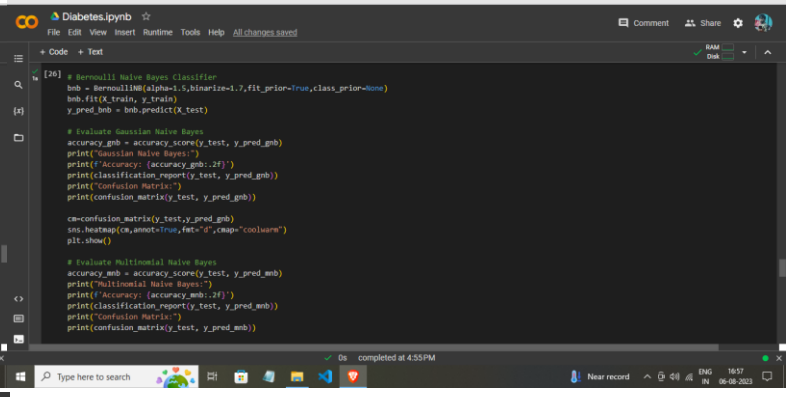

Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, export_text, export_graphviz
import pydotplus
from IPython.display import Image
# Load the diabetes dataset
diabetis_data = pd.read_csv('/content/drive/MyDrive/ML Lab/diabetes.csv')
# print(diabetis_data)
x = diabetis_data.drop('Outcome', axis=1)
y = diabetis_data['Outcome']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=30)
0s completed at 4:55PM
Type here to search

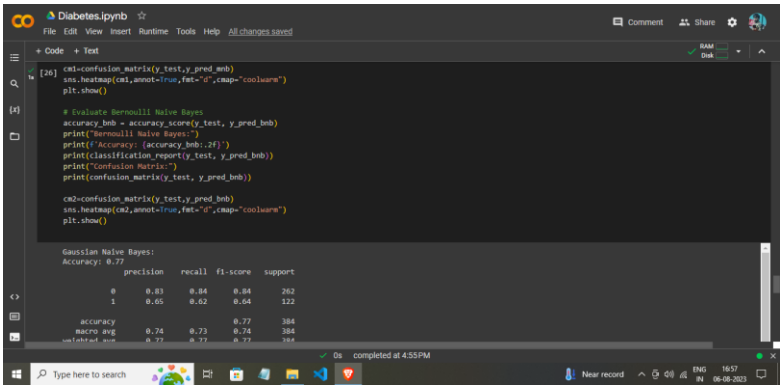

Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[25] # Decision Tree Classifier
dtc = DecisionTreeClassifier(criterion='entropy') # You can also use 'entropy' for criterion
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)
# Evaluate Decision Tree Classifier
accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
print("Decision Tree Classifier:")
print(f'Accuracy: {accuracy_dtc:.2f}')
print(classification_report(y_test, y_pred_dtc))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_dtc))
dot_data = export_graphviz(dtc, out_file=None, feature_names=X.columns, class_names=str(y.unique()),
filled=True, rounded=True, special_characters=False)
graph = pydotplus.graph_from_dot_data(dot_data)
# Save the decision tree image
decision_tree_image_path = "decision_tree.png"
graph.write_png(decision_tree_image_path)
# Display the decision tree image
Image(decision_tree_image_path)
0s completed at 4:55PM
Type here to search

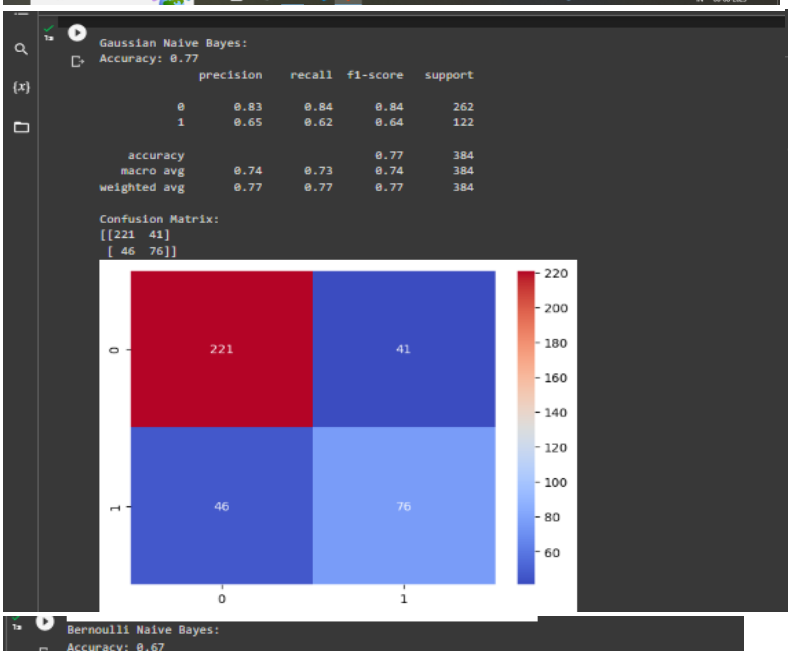

Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[25] # Save the decision tree image
decision_tree_image_path = "decision_tree.png"
graph.write_png(decision_tree_image_path)
# Display the decision tree image
Image(decision_tree_image_path)
Decision Tree Classifier:
Accuracy: 0.74
precision recall f1-score support
0 0.82 0.81 0.81 262
1 0.60 0.61 0.60 122
accuracy 0.74 384
macro avg 0.71 0.71 0.71 384
weighted avg 0.75 0.74 0.75 384
Confusion Matrix:
[[212 50]
[48 74]]
0s completed at 4:55PM
Type here to search


Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s completed at 4:55PM
Type here to search

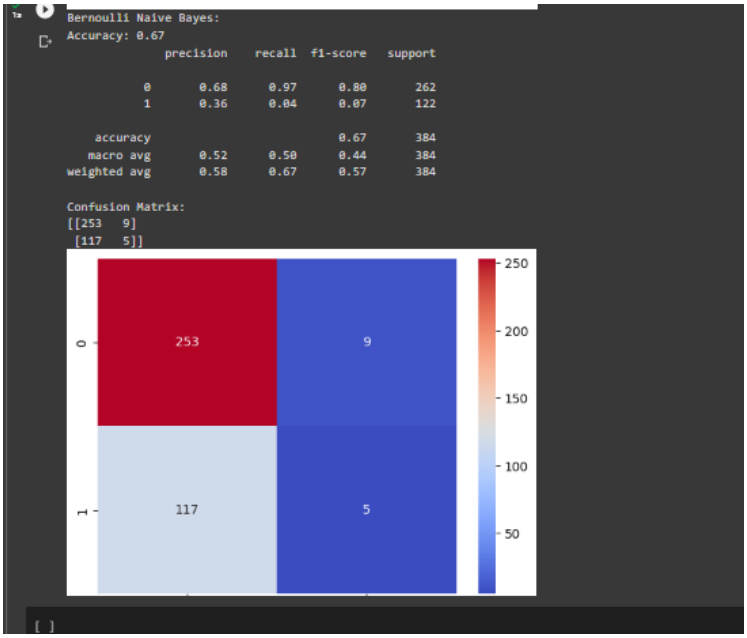

Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[25]
# Gaussian Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_gnb = gnb.predict(X_test)
# Multinomial Naive Bayes Classifier
mnb = MultinomialNB(alpha=1.2,fit_prior=True,class_prior=None)
mnb.fit(X_train, y_train)
y_pred_mnb = mnb.predict(X_test)
# Bernoulli Naive Bayes Classifier
bnb = BernoulliNB(alpha=1.5,binarize=1.7,fit_prior=True,class_prior=None)
bnb.fit(X_train, y_train)
y_pred_bnb = bnb.predict(X_test)
0s completed at 4:55PM


Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[26]
# Bernoulli Naive Bayes Classifier
bnb = BernoulliNB(alpha=1.5,binarize=1.7,fit_prior=True,class_prior=None)
bnb.fit(X_train, y_train)
y_pred_bnb = bnb.predict(X_test)
# Evaluate Gaussian Naive Bayes
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
print("Gaussian Naive Bayes:")
print(f"Accuracy: {accuracy_gnb:.2f}")
print(classification_report(y_test, y_pred_gnb))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_gnb))
cm=confusion_matrix(y_test,y_pred_gnb)
sns.heatmap(cm,annot=True,fmt="d",cmap="coolwarm")
plt.show()
# Evaluate Multinomial Naive Bayes
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
print("Multinomial Naive Bayes:")
print(f"Accuracy: {accuracy_mnb:.2f}")
print(classification_report(y_test, y_pred_mnb))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_mnb))
0s completed at 4:55PM


Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[26]
cm1=confusion_matrix(y_test,y_pred_mnb)
sns.heatmap(cm1,annot=True,fmt="d",cmap="coolwarm")
plt.show()
# Evaluate Bernoulli Naive Bayes
accuracy_bnb = accuracy_score(y_test, y_pred_bnb)
print("Bernoulli Naive Bayes:")
print(f"Accuracy: {accuracy_bnb:.2f}")
print(classification_report(y_test, y_pred_bnb))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_bnb))
cm2=confusion_matrix(y_test,y_pred_bnb)
sns.heatmap(cm2,annot=True,fmt="d",cmap="coolwarm")
plt.show()
Gaussian Naive Bayes:
Accuracy: 0.77
precision recall f1-score support
0 0.83 0.84 0.84 262
1 0.65 0.62 0.64 122
accuracy 0.77 384
macro avg 0.74 0.73 0.74 384
0s completed at 4:55PM


Gaussian Naive Bayes:
Accuracy: 0.77
precision recall f1-score support
0 0.83 0.84 0.84 262
1 0.65 0.62 0.64 122
accuracy 0.77 384
macro avg 0.74 0.73 0.74 384
weighted avg 0.77 0.77 0.77 384
Confusion Matrix:
[[221 41]
[46 76]]
221
41
46
76
0
1
220
200
180
160
140
120
100
80
60


Diabetes.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Accuracy: 0.62
precision recall f1-score support
0 0.73 0.69 0.71 262
1 0.41 0.47 0.44 122
accuracy 0.62 384
macro avg 0.57 0.58 0.57 384
weighted avg 0.63 0.62 0.62 384
Confusion Matrix:
[[180 82]
[65 57]]
180
82
65
57
0
1
180
160
140
120
100
80
60


Bernoulli Naive Bayes:
Accuracy: 0.67
precision recall f1-score support
0 0.68 0.97 0.80 262
1 0.36 0.04 0.07 122
accuracy 0.67 384
macro avg 0.52 0.50 0.44 384
weighted avg 0.58 0.67 0.57 384
Confusion Matrix:
[[253 9]
[117 5]]
253
9
117
5
0
1
250
200
150
100
50