Name - Rishik Varma
Roll - 001811001050
Dept- I.T 4th yr 1st sem
Sub - ML Lab Evaluation

Google Colab Link :

- **Question 1.1 →** https://colab.research.google.com/drive/1fqcX9CC8a4Mr0Hs9tM4N2G6zrvzKfjbd?usp=sharing
- **Question 1.2 →** https://colab.research.google.com/drive/1s3ut32N4In_ufZ05sbYK6vQaUyc3iRNY?usp=sharing
- **Question 2 →** https://colab.research.google.com/drive/1-CPvmMM4jzRWuY2qZCvKCUZzbsSkw-Td?usp=sharing
- **Question 5 →** https://colab.research.google.com/drive/1PapEFoBVCFzjA5jrHEUCU_YfAxm_lAxX?usp=sharing

Github Repo Link → https://github.com/Knightrv/ML_Lab_Evaluation

Question 1.1 starts :

Name --> Rishik Varma

Roll --> 001811001050

Sub --> ML Lab Evaluation Question Number - 1.1

Year --> IT 4th Yr 1st sem

[ ] ↳ *1 cell hidden*

Load Dataset

```
1 wine = datasets.load_wine() # it's source is same as : https://archive.ics.uci.edu/ml/d
```

```
1 dir(wine)
```

```
['DESCR', 'data', 'feature_names', 'frame', 'target', 'target_names']
```

```
1 wine.data
```

```
array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

```
1 print(wine.feature_names)
2 print(wine.target_names)
3 print(wine.target)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols',
['class_0' 'class_1' 'class_2']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
1 df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
2 df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69

```
1 df["target"] = wine.target
2 df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69

```
1 wine.target_names
```

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

DataFrame ready to perform

```
1 len(df)
```

```
178
```

```
1 X = df.drop(["target"], axis="columns")
2 y = df.target
3 print(X.head())
4 print(y.head())
```

```
   alcohol  malic_acid   ash  ...   hue  od280/od315_of_diluted_wines  proline
0    14.23        1.71  2.43  ...  1.04                          3.92   1065.0
1    13.20        1.78  2.14  ...  1.05                          3.40   1050.0
2    13.16        2.36  2.67  ...  1.03                          3.17   1185.0
```

```
3    14.37        1.95  2.50  ...  0.86                          3.45   1480.0
4    13.24        2.59  2.87  ...  1.04                          2.93    735.0

[5 rows x 13 columns]
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int64
```

SVC Classfier

Linear SVC Classifier

```
1 linear_SVC_classifier = SVC(kernel='linear')
2 linear_SVC_classifier
```

```
SVC(kernel='linear')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

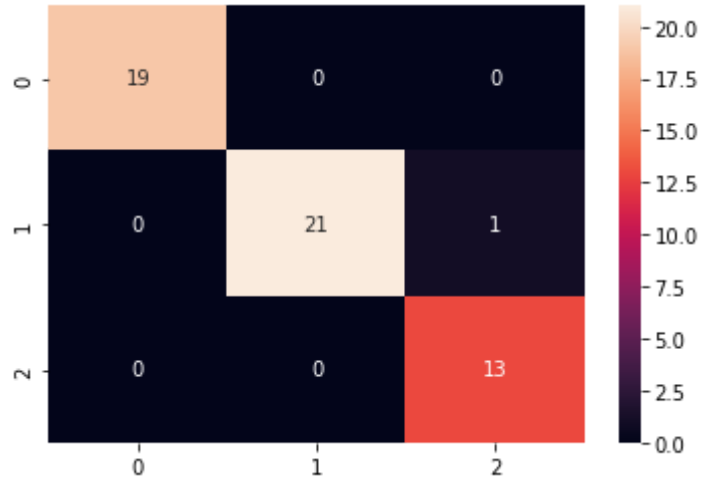
```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 98.14814814814815%

Confusion Matrix:

[[19  0  0]
```

```
 [ 0 21  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        22
           2       0.93      1.00      0.96        13

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1eb2a10>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

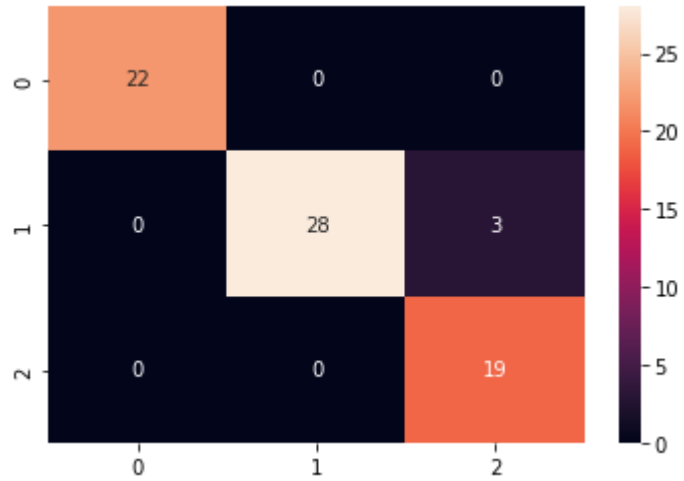
```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
```

```
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.83333333333334%

Confusion Matrix:
[[22  0  0]
 [ 0 28  3]
 [ 0  0 19]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       1.00      0.90      0.95        31
           2       0.86      1.00      0.93        19

    accuracy                           0.96        72
   macro avg       0.95      0.97      0.96        72
weighted avg       0.96      0.96      0.96        72
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1a13b10>
```



train size : test size = 50% : 50%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
89
89
```

```
linear_SVC_classifier.fit(X_train, y_train)
```

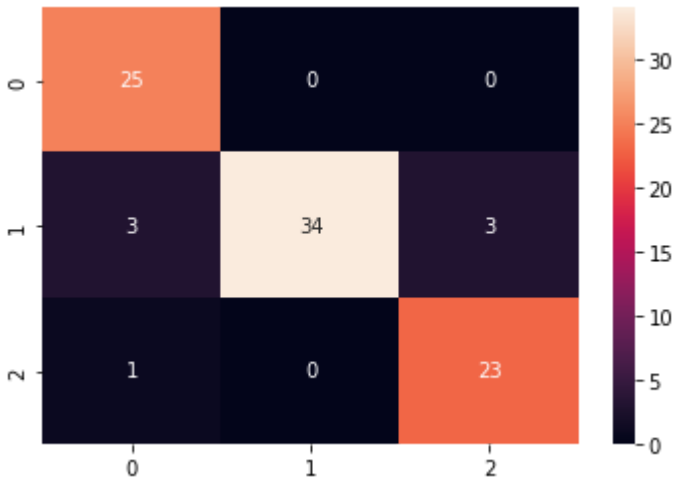
```
SVC(kernel='linear')
```

```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 92.13483146067416%

Confusion Matrix:
[[25  0  0]
 [ 3 34  3]
 [ 1  0 23]]

Classification Report:

              precision    recall  f1-score   support

           0       0.86      1.00      0.93        25
           1       1.00      0.85      0.92        40
           2       0.88      0.96      0.92        24

    accuracy                           0.92        89
   macro avg       0.92      0.94      0.92        89
weighted avg       0.93      0.92      0.92        89
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1941b90>
```



train size : test size = 40% : 60%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

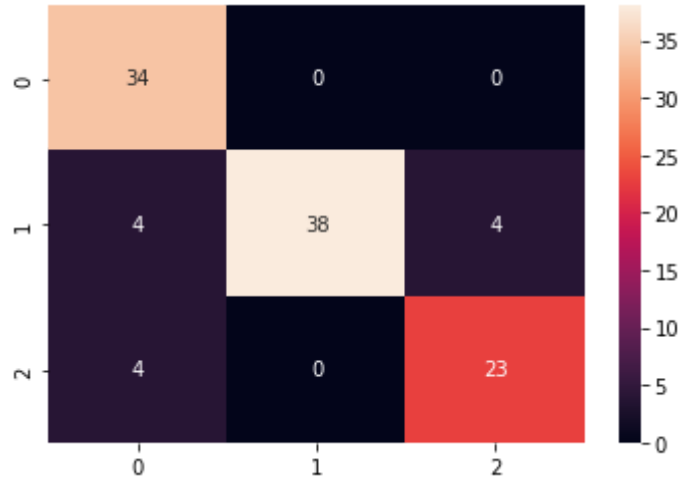
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.78504672897196%

Confusion Matrix:

[[34  0  0]
 [ 4 38  4]
 [ 4  0 23]]

Classification Report:

              precision    recall  f1-score   support

           0       0.81      1.00      0.89        34
           1       1.00      0.83      0.90        46
           2       0.85      0.85      0.85        27

    accuracy                           0.89       107
   macro avg       0.89      0.89      0.88       107
weighted avg       0.90      0.89      0.89       107
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1890490>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

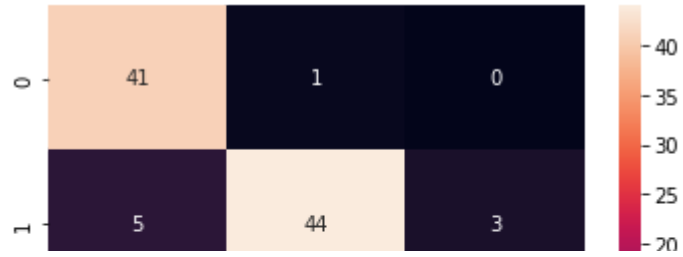
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.0%

Confusion Matrix:

[[41  1  0]
 [ 5 44  3]
 [ 5  1 25]]

Classification Report:

              precision    recall  f1-score   support

           0       0.80      0.98      0.88        42
           1       0.96      0.85      0.90        52
           2       0.89      0.81      0.85        31

    accuracy                           0.88       125
   macro avg       0.88      0.88      0.88       125
weighted avg       0.89      0.88      0.88       125
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb17cc450>
```



Polynomial SVC Classifier


```
1 poly_SVC_classifier = SVC(kernel='poly')
2 poly_SVC_classifier
```

```
SVC(kernel='poly')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='poly')
```

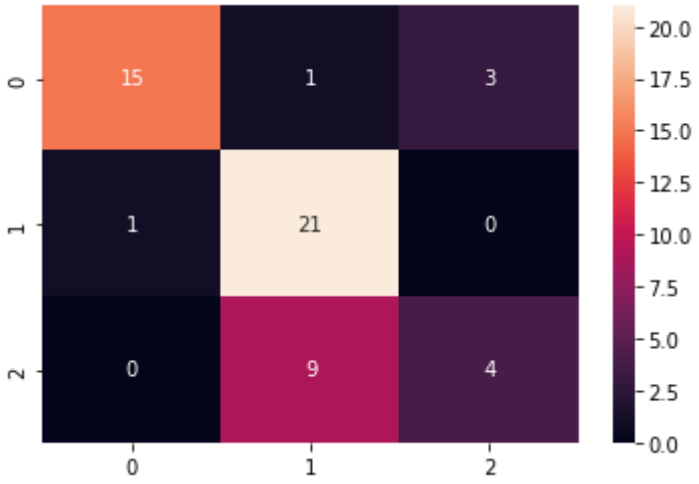
```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 74.07407407407408%

Confusion Matrix:
 [[15  1  3]
 [ 1 21  0]
 [ 0  9  4]]

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.79      0.86        19
           1       0.68      0.95      0.79        22
```

```
              2       0.57      0.31      0.40        13

       accuracy                           0.74        54
      macro avg       0.73      0.68      0.68        54
   weighted avg       0.74      0.74      0.72        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb170bad0>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='poly')
```

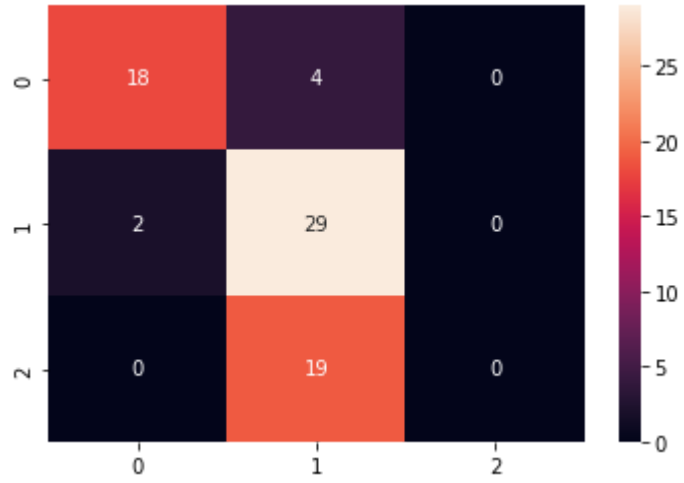
```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 65.27777777777779%

Confusion Matrix:
 [[18  4  0]
 [ 2 29  0]
 [ 0 19  0]]
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.90      0.82      0.86        22
           1       0.56      0.94      0.70        31
           2       0.00      0.00      0.00        19

    accuracy                           0.65        72
   macro avg       0.49      0.58      0.52        72
weighted avg       0.52      0.65      0.56        72

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1640450>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

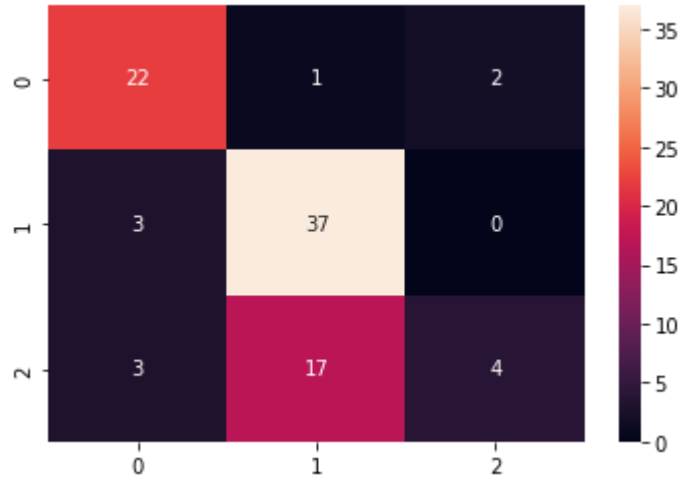
```
SVC(kernel='poly')
```

```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 70.78651685393258%

Confusion Matrix:
 [[22  1  2]
 [ 3 37  0]
 [ 3 17  4]]

Classification Report:

              precision    recall  f1-score   support

           0       0.79      0.88      0.83        25
           1       0.67      0.93      0.78        40
           2       0.67      0.17      0.27        24

    accuracy                           0.71        89
   macro avg       0.71      0.66      0.63        89
weighted avg       0.70      0.71      0.66        89
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1560950>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

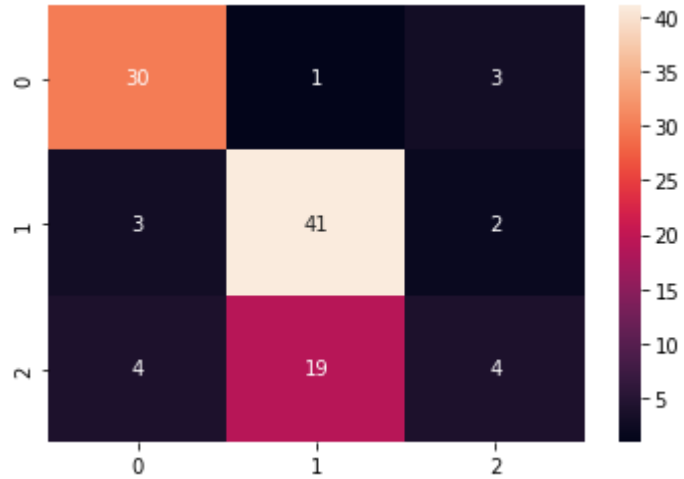
```
SVC(kernel='poly')
```

```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 70.09345794392523%

Confusion Matrix:
 [[30  1  3]
 [ 3 41  2]
 [ 4 19  4]]

Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.88      0.85        34
           1       0.67      0.89      0.77        46
           2       0.44      0.15      0.22        27

    accuracy                           0.70       107
   macro avg       0.64      0.64      0.61       107
weighted avg       0.66      0.70      0.65       107
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb14a3210>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='poly')
```

```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 70.39999999999999%

Confusion Matrix:
 [[38  1  3]
 [ 3 45  4]
 [ 4 22  5]]

Classification Report:

              precision    recall  f1-score   support

           0       0.84      0.90      0.87        42
           1       0.66      0.87      0.75        52
           2       0.42      0.16      0.23        31

    accuracy                           0.70       125
   macro avg       0.64      0.64      0.62       125
weighted avg       0.66      0.70      0.66       125
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

Gaussain SVC Classifier

```
1 gaussain_SVC_classifier = SVC(kernel='rbf')
2 gaussain_SVC_classifier
```

```
SVC()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

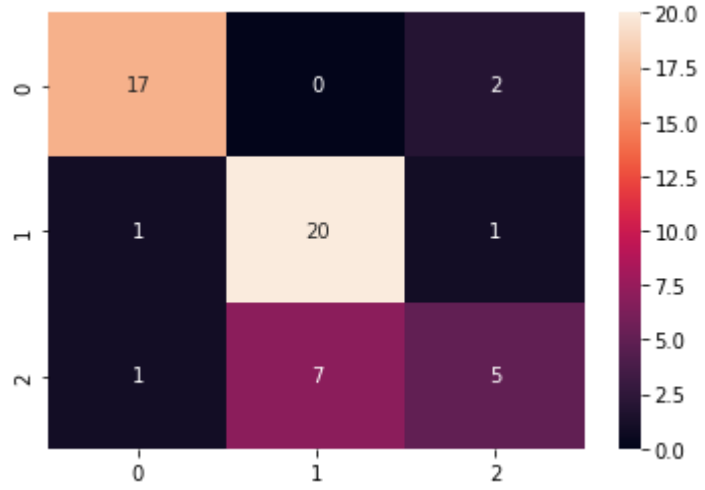
```
SVC()
```

```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 77.77777777777779%

Confusion Matrix:
 [[17  0  2]
 [ 1 20  1]
 [ 1  7  5]]

Classification Report:

              precision    recall  f1-score   support

           0       0.89      0.89      0.89        19
           1       0.74      0.91      0.82        22
           2       0.62      0.38      0.48        13

    accuracy                           0.78        54
   macro avg       0.75      0.73      0.73        54
weighted avg       0.77      0.78      0.76        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb13225d0>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

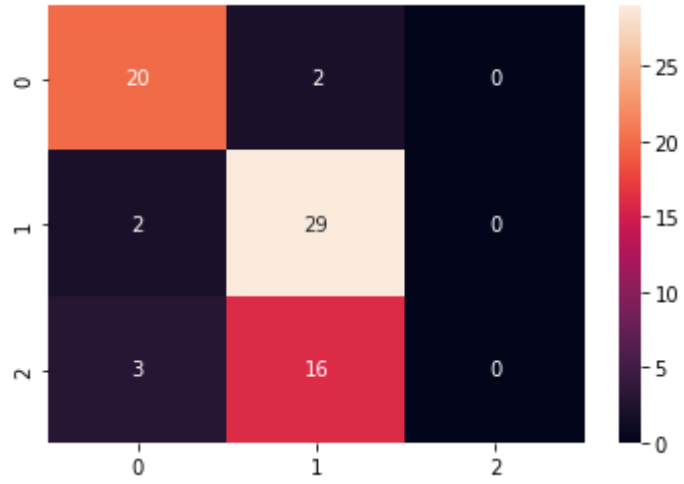
```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 68.05555555555556%

Confusion Matrix:
 [[20  2  0]
 [ 2 29  0]
 [ 3 16  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.80      0.91      0.85        22
           1       0.62      0.94      0.74        31
           2       0.00      0.00      0.00        19
```

```
    accuracy                           0.68        72
   macro avg       0.47      0.61      0.53        72
weighted avg       0.51      0.68      0.58        72

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb12db450>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

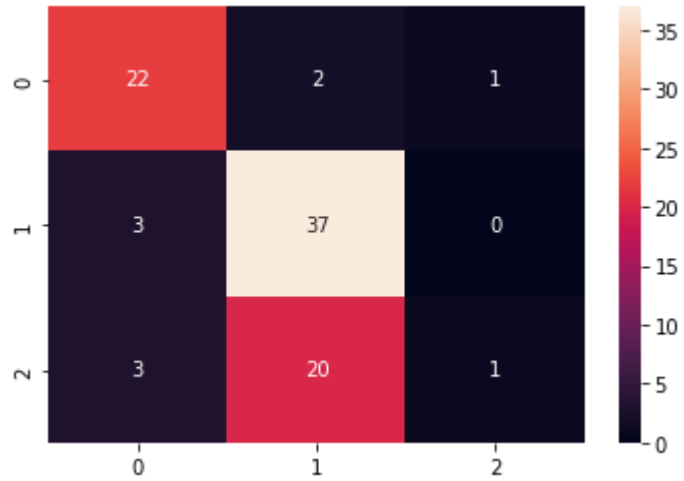
```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 67.41573033707866%
```

```
Confusion Matrix:
 [[22  2  1]
 [ 3 37  0]
 [ 3 20  1]]

Classification Report:

              precision    recall  f1-score   support

           0       0.79      0.88      0.83        25
           1       0.63      0.93      0.75        40
           2       0.50      0.04      0.08        24

    accuracy                           0.67        89
   macro avg       0.64      0.62      0.55        89
weighted avg       0.64      0.67      0.59        89
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb11df5d0>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

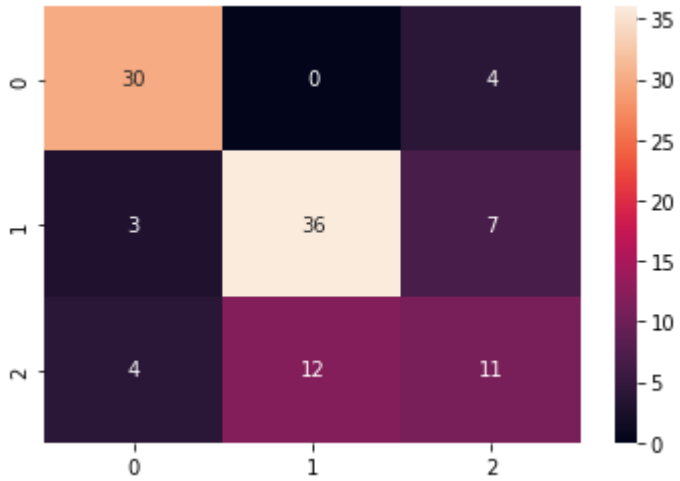
```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
```

```
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 71.96261682242991%

Confusion Matrix:
 [[30  0  4]
 [ 3 36  7]
 [ 4 12 11]]

Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.88      0.85        34
           1       0.75      0.78      0.77        46
           2       0.50      0.41      0.45        27

    accuracy                           0.72       107
   macro avg       0.69      0.69      0.69       107
weighted avg       0.71      0.72      0.71       107
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb113a590>
```



train size : test size = 30% : 70%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
53
125
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

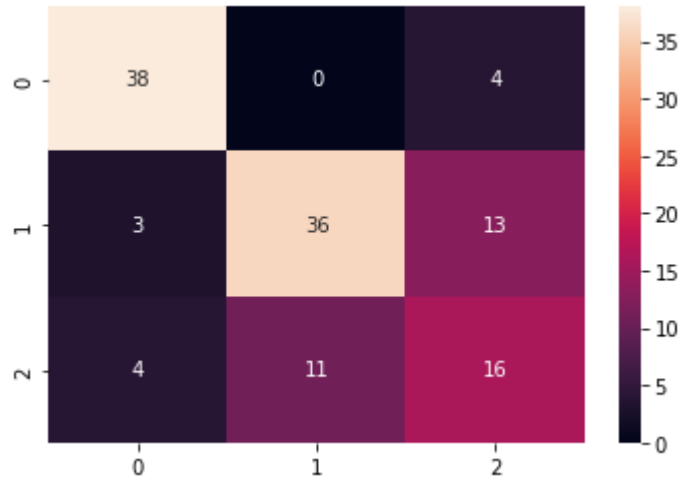
```
SVC()
```

```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 72.0%

Confusion Matrix:
 [[38  0  4]
 [ 3 36 13]
 [ 4 11 16]]

Classification Report:

              precision    recall  f1-score   support

           0       0.84      0.90      0.87        42
           1       0.77      0.69      0.73        52
           2       0.48      0.52      0.50        31

    accuracy                           0.72       125
   macro avg       0.70      0.70      0.70       125
weighted avg       0.72      0.72      0.72       125
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb106bc50>
```



Sigmoid SVC Classifier

```
1 sigmoid_SVC_classifier = SVC(kernel='sigmoid')
2 sigmoid_SVC_classifier
```

```
SVC(kernel='sigmoid')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

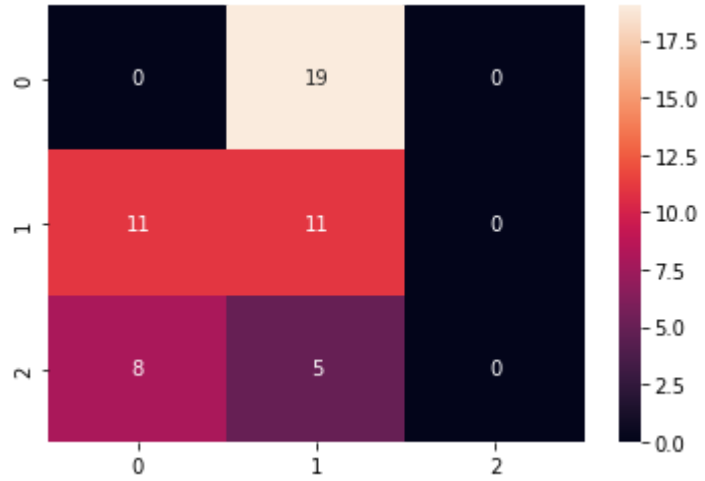
```
SVC(kernel='sigmoid')
```

```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 20.37037037037037%

Confusion Matrix:
 [[ 0 19  0]
 [11 11  0]
 [ 8  5  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00        19
           1       0.31      0.50      0.39        22
           2       0.00      0.00      0.00        13

    accuracy                           0.20        54
   macro avg       0.10      0.17      0.13        54
weighted avg       0.13      0.20      0.16        54

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0faf1d0>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='sigmoid')
```

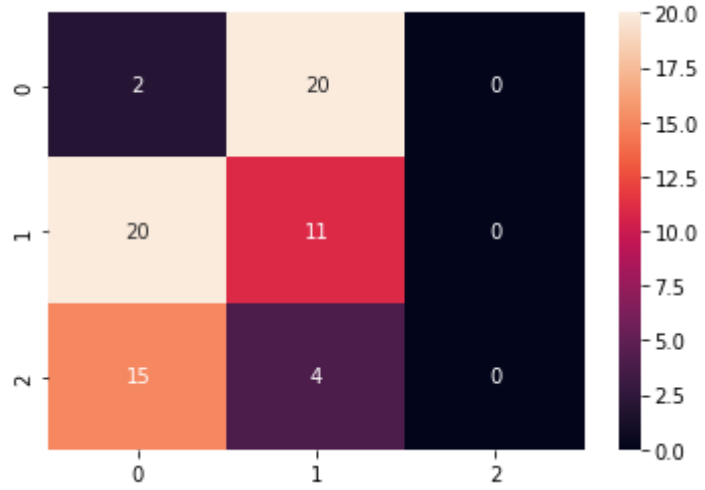
```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 18.055555555555554%

Confusion Matrix:
 [[ 2 20  0]
 [20 11  0]
 [15  4  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.05      0.09      0.07        22
           1       0.31      0.35      0.33        31
           2       0.00      0.00      0.00        19

    accuracy                           0.18        72
   macro avg       0.12      0.15      0.13        72
```

```
weighted avg       0.15      0.18      0.16         72

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb1a01250>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='sigmoid')
```

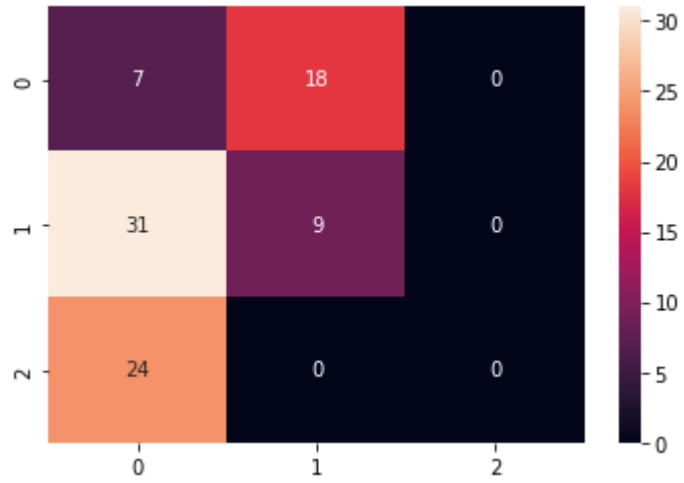
```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 17.97752808988764%

Confusion Matrix:
```

```
[[ 7 18  0]
 [31  9  0]
 [24  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.11      0.28      0.16        25
           1       0.33      0.23      0.27        40
           2       0.00      0.00      0.00        24

    accuracy                           0.18        89
   macro avg       0.15      0.17      0.14        89
weighted avg       0.18      0.18      0.17        89

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0e88310>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

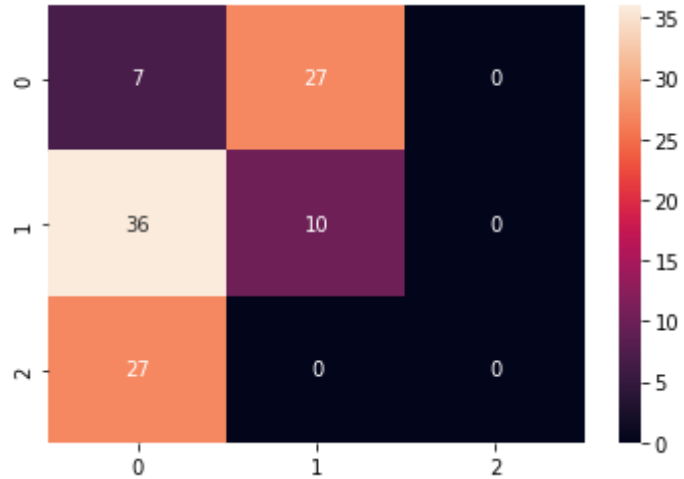
```
SVC(kernel='sigmoid')
```

```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 15.887850467289718%

Confusion Matrix:
 [[ 7 27  0]
 [36 10  0]
 [27  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.10      0.21      0.13        34
           1       0.27      0.22      0.24        46
           2       0.00      0.00      0.00        27

    accuracy                           0.16       107
   macro avg       0.12      0.14      0.13       107
weighted avg       0.15      0.16      0.15       107

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0db1c10>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

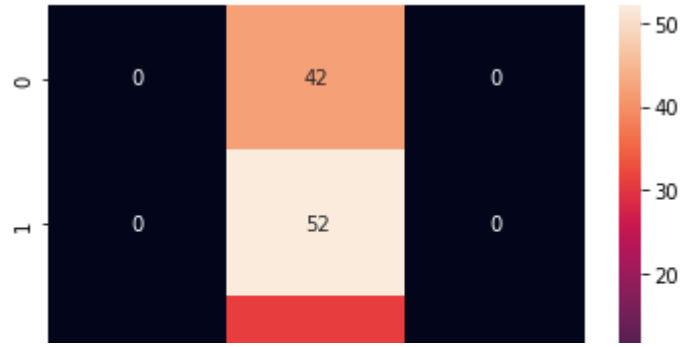
```
SVC(kernel='sigmoid')
```

```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 41.6%

Confusion Matrix:
 [[ 0 42  0]
 [ 0 52  0]
 [ 0 31  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00        42
           1       0.42      1.00      0.59        52
           2       0.00      0.00      0.00        31

    accuracy                           0.42       125
   macro avg       0.14      0.33      0.20       125
weighted avg       0.17      0.42      0.24       125

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0ce6050>
```



Decision Tree

```
1 from sklearn.tree import DecisionTreeClassifier
2 clf = DecisionTreeClassifier()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

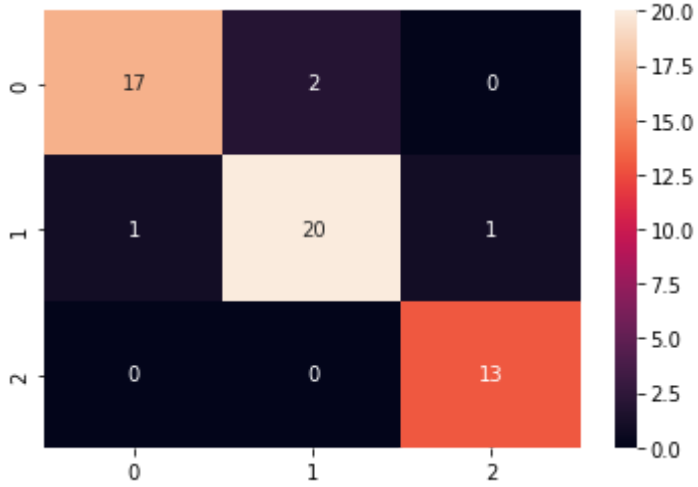
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 92.5925925925926%

Confusion Matrix:

[[17  2  0]
 [ 1 20  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.89      0.92        19
```

```
                1       0.91      0.91      0.91        22
                2       0.93      1.00      0.96        13

         accuracy                           0.93        54
        macro avg       0.93      0.93      0.93        54
     weighted avg       0.93      0.93      0.93        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0c2bf50>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

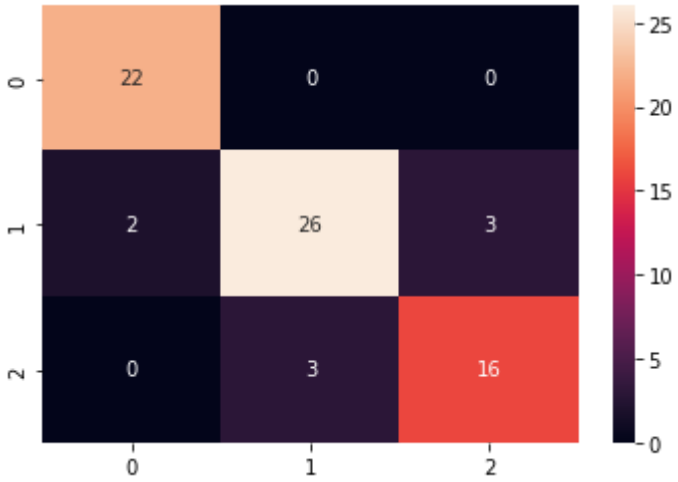
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.88888888888889%

Confusion Matrix:
[[22  0  0]
 [ 2 26  3]
 [ 0  3 16]]
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.92      1.00      0.96        22
           1       0.90      0.84      0.87        31
           2       0.84      0.84      0.84        19

    accuracy                           0.89        72
   macro avg       0.89      0.89      0.89        72
weighted avg       0.89      0.89      0.89        72
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0bdb0d0>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

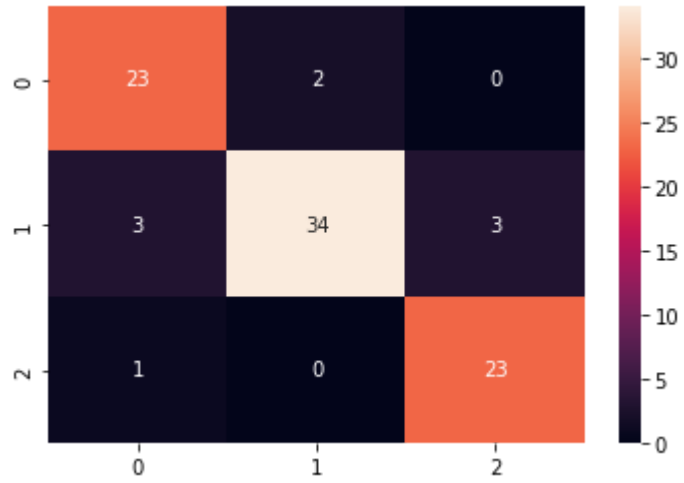
```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 89.8876404494382%

Confusion Matrix:
[[23  2  0]
 [ 3 34  3]
 [ 1  0 23]]

Classification Report:

              precision    recall  f1-score   support

           0       0.85      0.92      0.88        25
           1       0.94      0.85      0.89        40
           2       0.88      0.96      0.92        24

    accuracy                           0.90        89
   macro avg       0.89      0.91      0.90        89
weighted avg       0.90      0.90      0.90        89
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0af0910>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

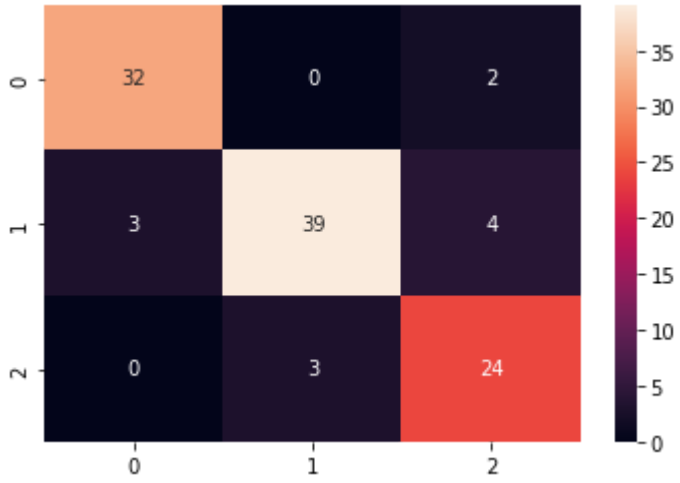
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
```

```
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.78504672897196%

Confusion Matrix:

[[32  0  2]
 [ 3 39  4]
 [ 0  3 24]]

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.94      0.93        34
           1       0.93      0.85      0.89        46
           2       0.80      0.89      0.84        27

    accuracy                           0.89       107
   macro avg       0.88      0.89      0.89       107
weighted avg       0.89      0.89      0.89       107
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0a5b090>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

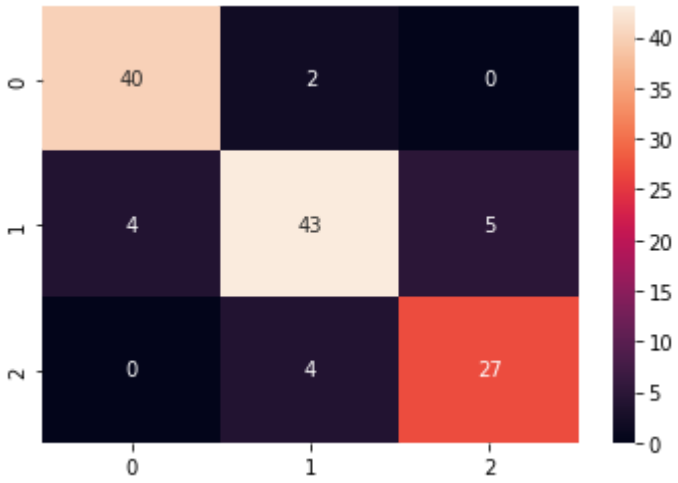
```
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

```
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.0%

Confusion Matrix:

[[40  2  0]
 [ 4 43  5]
 [ 0  4 27]]

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.95      0.93        42
           1       0.88      0.83      0.85        52
           2       0.84      0.87      0.86        31

    accuracy                           0.88       125
   macro avg       0.88      0.88      0.88       125
weighted avg       0.88      0.88      0.88       125
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb098ec50>
```



Random Forest Classifier

```
rfc_classifier = RandomForestClassifier(n_estimators=20)
```

```
2 rfc_classifier
```

```
RandomForestClassifier(n_estimators=20)
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 rfc_classifier.fit(X_train, y_train)
```

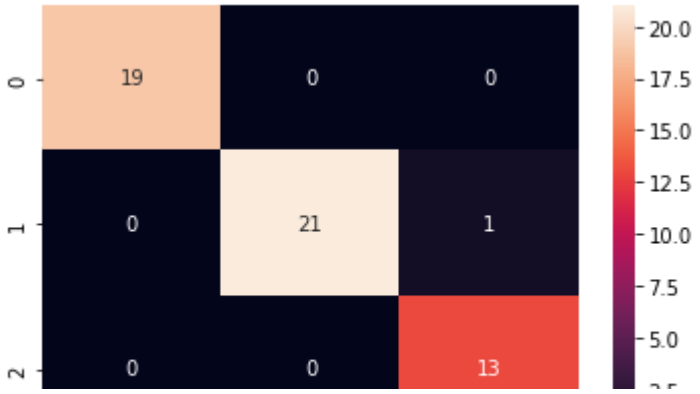
```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 98.14814814814815%

Confusion Matrix:

[[19  0  0]
 [ 0 21  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        22
           2       0.93      1.00      0.96        13

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb08c1d10>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 rfc_classifier.fit(X_train, y_train)
```

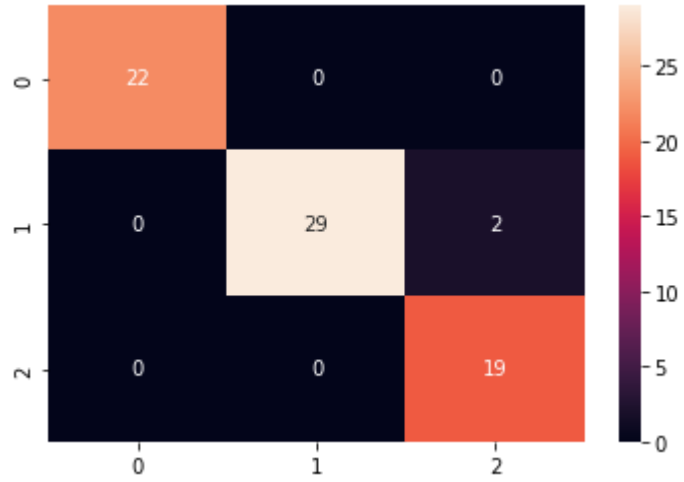
```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.22222222222221%

Confusion Matrix:
[[22  0  0]
 [ 0 29  2]
 [ 0  0 19]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       1.00      0.94      0.97        31
           2       0.90      1.00      0.95        19

    accuracy                           0.97        72
   macro avg       0.97      0.98      0.97        72
weighted avg       0.97      0.97      0.97        72
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb08103d0>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

```
1 rfc_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

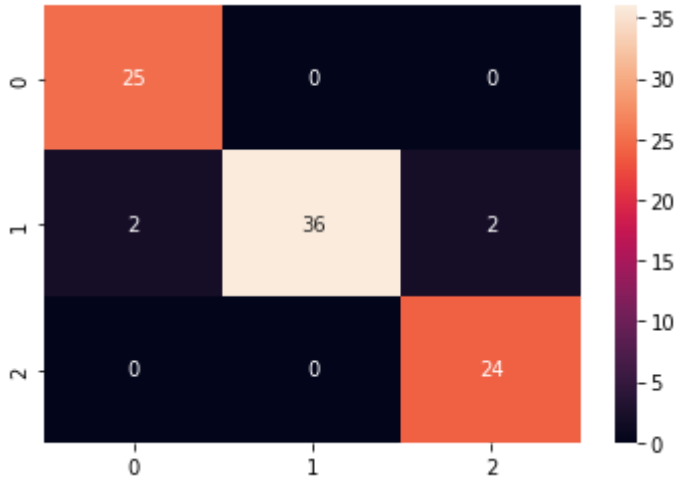
```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.50561797752809%

Confusion Matrix:
[[25  0  0]
 [ 2 36  2]
 [ 0  0 24]]

Classification Report:

              precision    recall  f1-score   support
```

```
              0       0.93      1.00      0.96        25
              1       1.00      0.90      0.95        40
              2       0.92      1.00      0.96        24

       accuracy                           0.96        89
      macro avg       0.95      0.97      0.96        89
   weighted avg       0.96      0.96      0.95        89
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb071db10>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 rfc_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

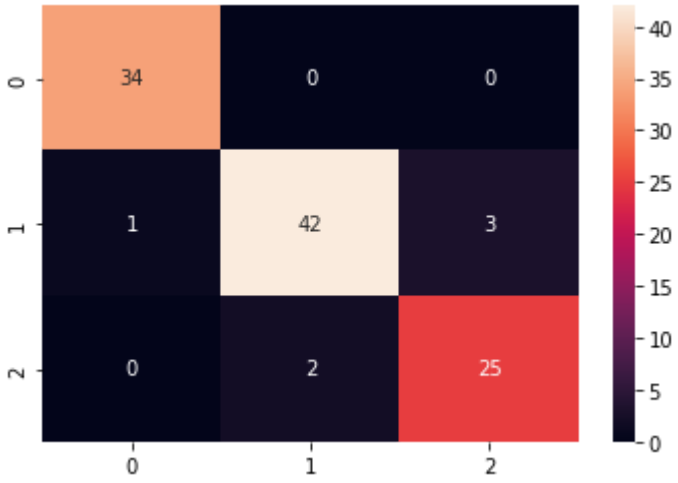
```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.39252336448598%

Confusion Matrix:
```

```
[[34  0  0]
 [ 1 42  3]
 [ 0  2 25]]

Classification Report:

              precision    recall  f1-score   support

           0       0.97      1.00      0.99        34
           1       0.95      0.91      0.93        46
           2       0.89      0.93      0.91        27

    accuracy                           0.94       107
   macro avg       0.94      0.95      0.94       107
weighted avg       0.94      0.94      0.94       107
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb06647d0>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

```
1 rfc_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

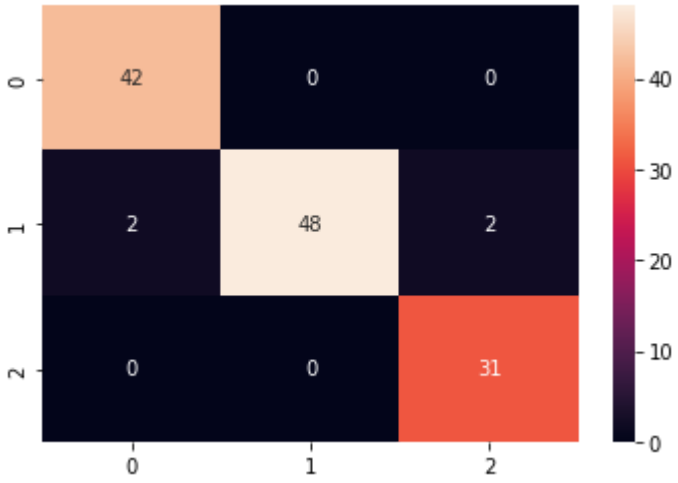
```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
```

```
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 96.8%

Confusion Matrix:

[[42  0  0]
 [ 2 48  2]
 [ 0  0 31]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      1.00      0.98        42
           1       1.00      0.92      0.96        52
           2       0.94      1.00      0.97        31

    accuracy                           0.97       125
   macro avg       0.96      0.97      0.97       125
weighted avg       0.97      0.97      0.97       125
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb05b9c90>
```



Naive Bayes

```
1 from sklearn.naive_bayes import GaussianNB
2 gnb = GaussianNB()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
124
54
```

```
1 gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

```
1 y_pred = gnb.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.44444444444444%

Confusion Matrix:

[[19  0  0]
 [ 2 19  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95        19
           1       1.00      0.86      0.93        22
           2       0.93      1.00      0.96        13

    accuracy                           0.94        54
   macro avg       0.94      0.95      0.95        54
weighted avg       0.95      0.94      0.94        54
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb04f1a90>
```

train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
106
72
```

```
1 gnb.fit(X_train, y_train)
```

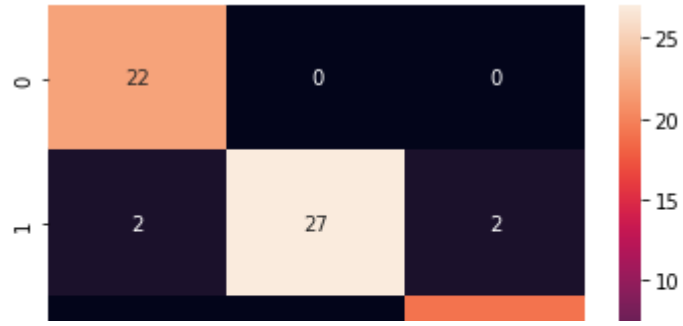
```
GaussianNB()
```

```
1 y_pred = gnb.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.44444444444444%

Confusion Matrix:
[[22  0  0]
 [ 2 27  2]
 [ 0  0 19]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      1.00      0.96        22
           1       1.00      0.87      0.93        31
           2       0.90      1.00      0.95        19

    accuracy                           0.94        72
   macro avg       0.94      0.96      0.95        72
weighted avg       0.95      0.94      0.94        72
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb042c0d0>
```



train size : test size = 50% : 50%


```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
89
89
```

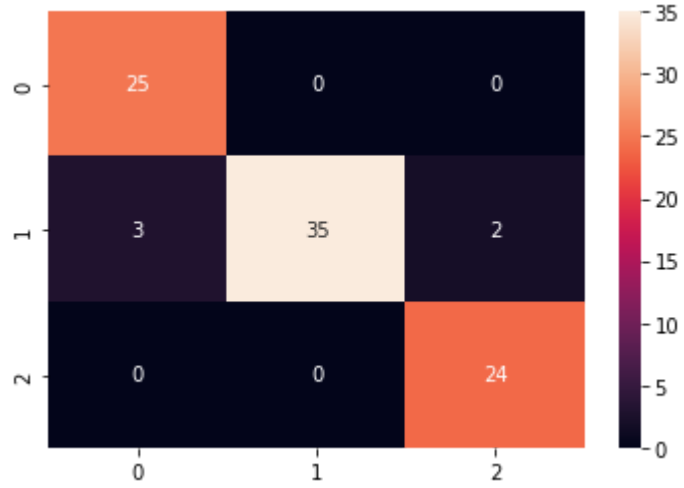
```
1 gnb.fit(X_train, y_train)
2 y_pred = gnb.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.3820224719101%

Confusion Matrix:
[[25  0  0]
 [ 3 35  2]
 [ 0  0 24]]

Classification Report:

              precision    recall  f1-score   support

           0       0.89      1.00      0.94        25
           1       1.00      0.88      0.93        40
           2       0.92      1.00      0.96        24

    accuracy                           0.94        89
   macro avg       0.94      0.96      0.95        89
weighted avg       0.95      0.94      0.94        89
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb04e4e10>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
71
107
```

```
1 gnb.fit(X_train, y_train)
2 y_pred = gnb.predict(X_test)
```

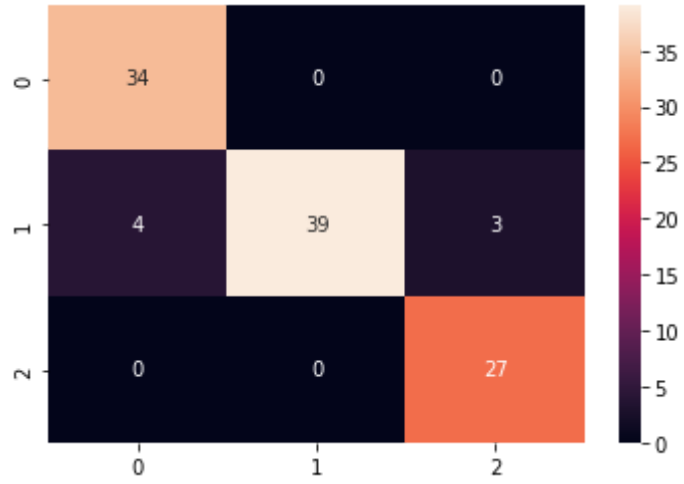
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.45794392523365%

Confusion Matrix:

[[34  0  0]
 [ 4 39  3]
 [ 0  0 27]]

Classification Report:

              precision    recall  f1-score   support

           0       0.89      1.00      0.94        34
           1       1.00      0.85      0.92        46
           2       0.90      1.00      0.95        27

    accuracy                           0.93       107
```

```
   macro avg       0.93      0.95      0.94       107
weighted avg       0.94      0.93      0.93       107
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0310810>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
53
125
```

```
1 gnb.fit(X_train, y_train)
2 y_pred = gnb.predict(X_test)
```

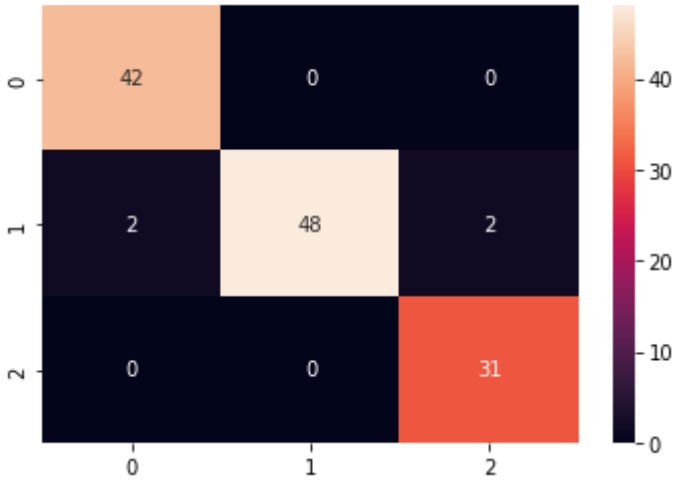
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 96.8%

Confusion Matrix:

[[42  0  0]
 [ 2 48  2]
 [ 0  0 31]]

Classification Report:
```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	42
1	1.00	0.92	0.96	52
2	0.94	1.00	0.97	31
accuracy			0.97	125
macro avg	0.96	0.97	0.97	125
weighted avg	0.97	0.97	0.97	125

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cb0238550>
```



Question 1.2 starts :

Name --> Rishik Varma

Roll --> 001811001050

Sub --> ML Lab Evaluation Question Number - 1.2

Year --> IT 4th Yr 1st sem

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
```

Load Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving ionosphere.csv to ionosphere (3).csv

```
df = pd.read_csv('ionosphere.csv')
df.head()
```

	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	fe
0	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	
1	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	
2	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	
3	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	
4	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	

```
# df.column_ai.value_counts()
```

DataFrame ready to perform

```
1 len(df)
```

```
351
```

```
1 X = df.drop(["label"], axis="columns")
2 y = df.label
3 print(X.head())
4 print(y.head())
```

```
   feature1  feature2  feature3  ...  feature32  feature33  feature34
0         1         0   0.99539  ...   -0.54487    0.18641   -0.45300
1         1         0   1.00000  ...   -0.06288   -0.13738   -0.02447
2         1         0   1.00000  ...   -0.24180    0.56045   -0.38238
3         1         0   1.00000  ...    1.00000   -0.32382    1.00000
4         1         0   1.00000  ...   -0.59573   -0.04608   -0.65697

[5 rows x 34 columns]
0    g
1    b
2    g
3    b
4    g
Name: label, dtype: object
```

SVC Classfier

Linear SVC Classifier

```
1 linear_SVC_classifier = SVC(kernel='linear')
2 linear_SVC_classifier
```

```
SVC(kernel='linear')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

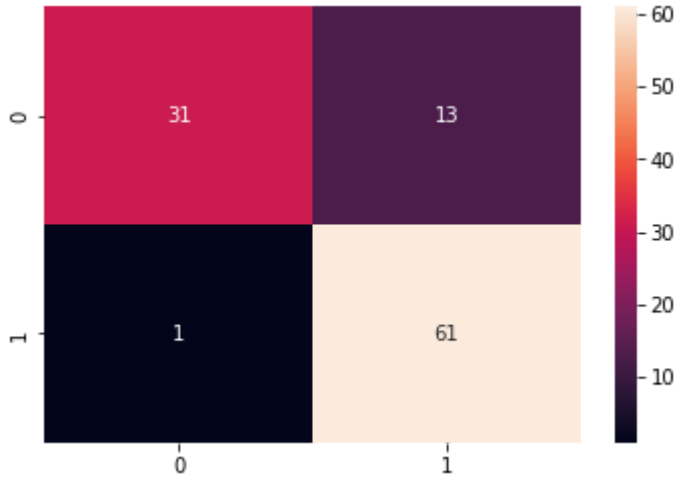
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 86.79245283018868%

Confusion Matrix:

[[31 13]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.70      0.82        44
           g       0.82      0.98      0.90        62

    accuracy                           0.87       106
   macro avg       0.90      0.84      0.86       106
weighted avg       0.88      0.87      0.86       106
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e47bb2790>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
210
141
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

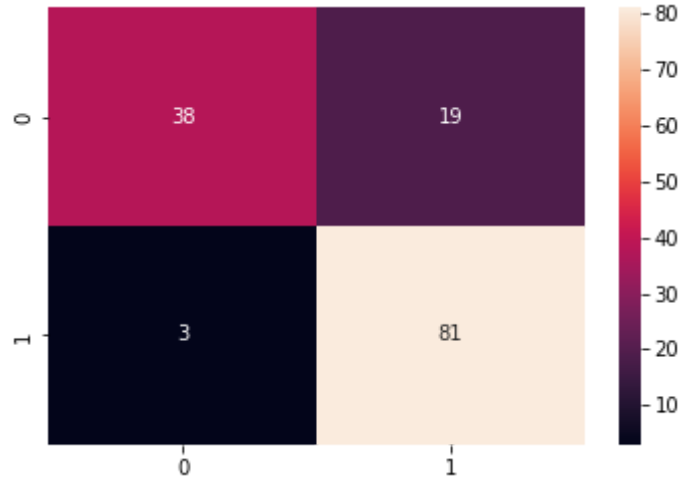
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 84.39716312056737%

Confusion Matrix:
[[38 19]
 [ 3 81]]

Classification Report:

              precision    recall  f1-score   support

           b       0.93      0.67      0.78        57
           g       0.81      0.96      0.88        84

    accuracy                           0.84       141
   macro avg       0.87      0.82      0.83       141
weighted avg       0.86      0.84      0.84       141
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e4765cd90>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
175
176
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

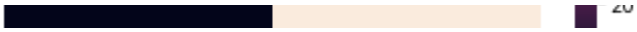
```
Accuracy: 81.25%

Confusion Matrix:
[[44 31]
 [ 2 99]]

Classification Report:

              precision    recall  f1-score   support

           b       0.96      0.59      0.73        75
           g       0.76      0.98      0.86       101

    accuracy                           0.81       176
   macro avg       0.86      0.78      0.79       176
weighted avg       0.84      0.81      0.80       176
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46d96b10>
```

train size : test size = 40% : 60%


```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```



```
1 print(len(X_train))
2 print(len(y_test))
```

```
140
211
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

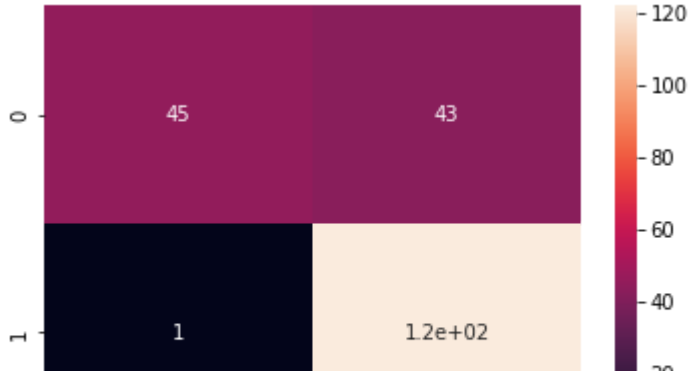
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 79.14691943127961%

Confusion Matrix:

[[ 45  43]
 [  1 122]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.51      0.67        88
           g       0.74      0.99      0.85       123

    accuracy                           0.79       211
   macro avg       0.86      0.75      0.76       211
weighted avg       0.84      0.79      0.77       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46d14190>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

```
1 linear_SVC_classifier.fit(X_train, y_train)
```

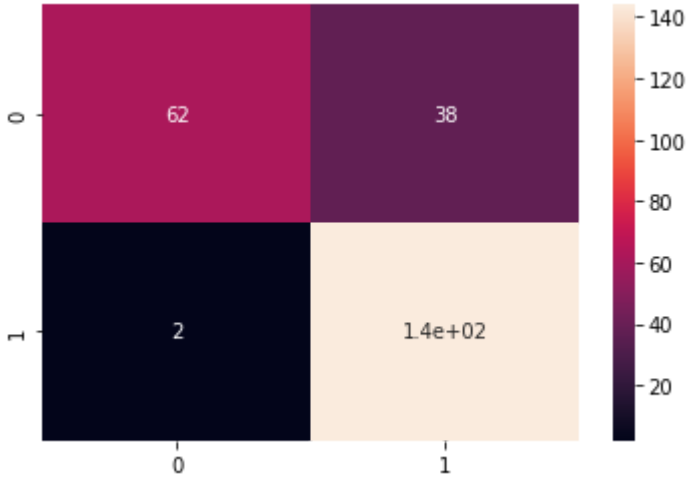
```
SVC(kernel='linear')
```

```
1 y_pred = linear_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 83.73983739837398%

Confusion Matrix:

[[ 62  38]
 [  2 144]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.62      0.76       100
           g       0.79      0.99      0.88       146

    accuracy                           0.84       246
   macro avg       0.88      0.80      0.82       246
weighted avg       0.86      0.84      0.83       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46c50910>
```



Polynomial SVC Classifier

```
1 poly_SVC_classifier = SVC(kernel='poly')
2 poly_SVC_classifier
```

```
SVC(kernel='poly')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='poly')
```

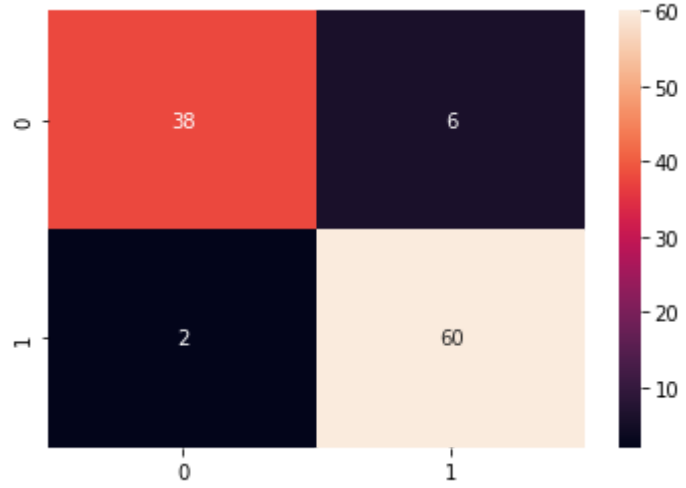
```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 92.45283018867924%

Confusion Matrix:
 [[38  6]
```

```
 [ 2 60]]

    Classification Report:

                  precision    recall  f1-score   support

               b       0.95      0.86      0.90        44
               g       0.91      0.97      0.94        62

        accuracy                           0.92       106
       macro avg       0.93      0.92      0.92       106
    weighted avg       0.93      0.92      0.92       106
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f9e46b7a910>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
    210
    141
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

```
    SVC(kernel='poly')
```

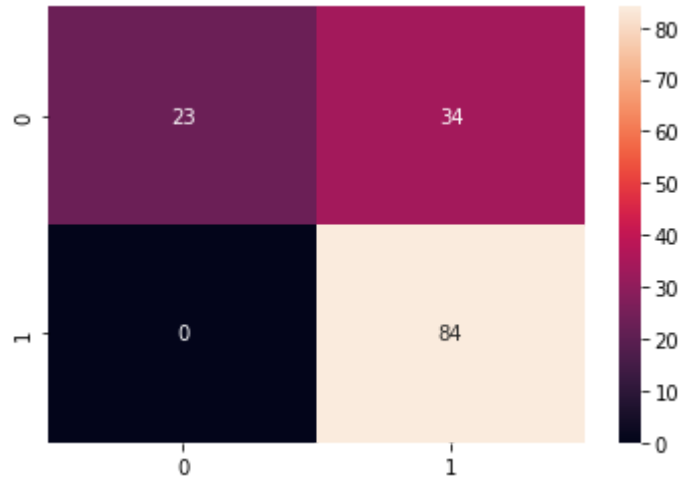
```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
```

```
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 75.88652482269504%

Confusion Matrix:
 [[23 34]
 [ 0 84]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.40      0.57        57
           g       0.71      1.00      0.83        84

    accuracy                           0.76       141
   macro avg       0.86      0.70      0.70       141
weighted avg       0.83      0.76      0.73       141
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46b22050>
```



train size : test size = 50% : 50%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

```
poly_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='poly')
```

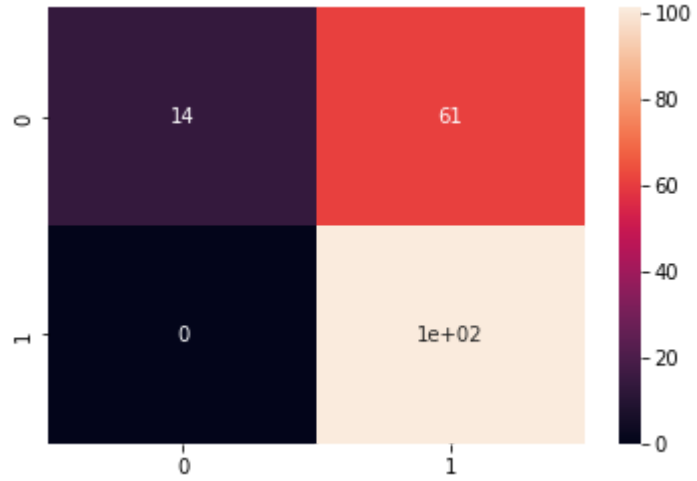
```
y_pred = poly_SVC_classifier.predict(X_test)
```

```
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 65.3409090909091%

Confusion Matrix:
 [[ 14  61]
 [  0 101]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.19      0.31        75
           g       0.62      1.00      0.77       101

    accuracy                           0.65       176
   macro avg       0.81      0.59      0.54       176
weighted avg       0.78      0.65      0.57       176
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46a49090>
```



train size : test size = 40% : 60%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

```
poly_SVC_classifier.fit(X_train, y_train)
```

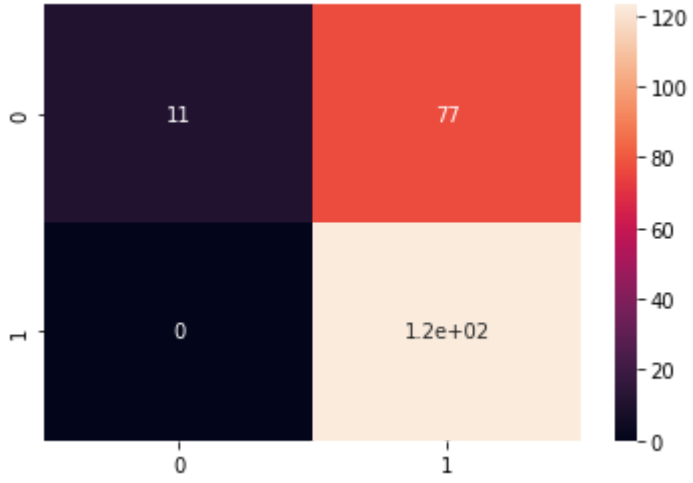
```
SVC(kernel='poly')
```

```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 63.507109004739334%

Confusion Matrix:
 [[ 11  77]
 [  0 123]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.12      0.22        88
           g       0.61      1.00      0.76       123

    accuracy                           0.64       211
   macro avg       0.81      0.56      0.49       211
weighted avg       0.78      0.64      0.54       211
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e469e3e10>
```



train size : test size = 30% : 70%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
105
```

```
    246
```

```
1 poly_SVC_classifier.fit(X_train, y_train)
```

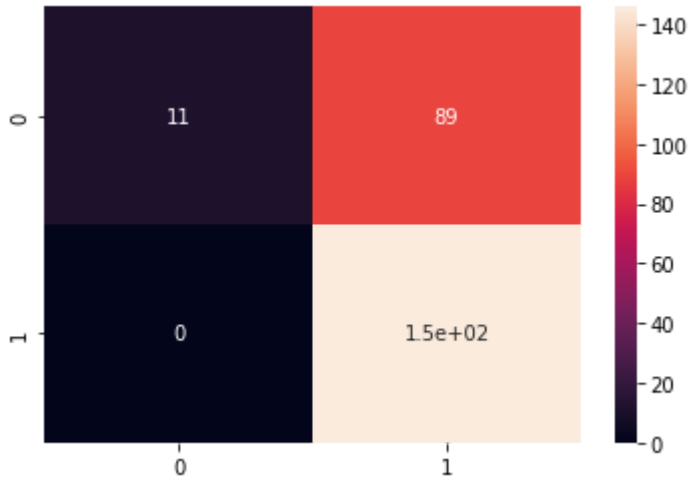
```
    SVC(kernel='poly')
```

```
1 y_pred = poly_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
    Accuracy: 63.82113821138211%

    Confusion Matrix:
     [[ 11  89]
     [  0 146]]

    Classification Report:

                  precision    recall  f1-score   support

               b       1.00      0.11      0.20       100
               g       0.62      1.00      0.77       146

        accuracy                           0.64       246
       macro avg       0.81      0.56      0.48       246
    weighted avg       0.78      0.64      0.54       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f9e4690fb50>
```



Gaussain SVC Classifier

```
1 gaussain_SVC_classifier = SVC(kernel='rbf')
2 gaussain_SVC_classifier
```

```
SVC()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.28301886792453%

Confusion Matrix:
 [[40  4]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.91      0.94        44
           g       0.94      0.98      0.96        62

    accuracy                           0.95       106
   macro avg       0.96      0.95      0.95       106
weighted avg       0.95      0.95      0.95       106
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46841390>
```

train size : test size = 60% : 40%


```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
210
141
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

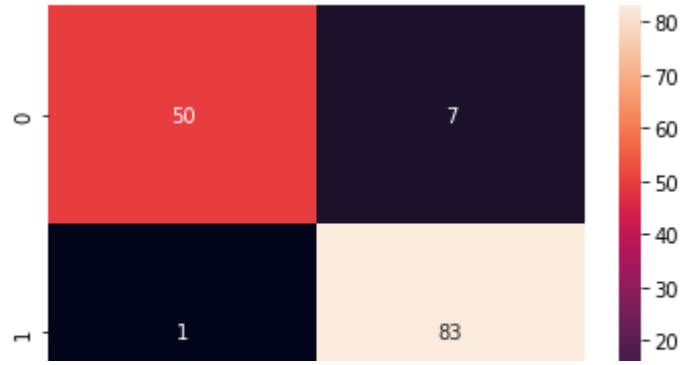
```
SVC()
```

```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.32624113475178%

Confusion Matrix:
 [[50  7]
 [ 1 83]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.88      0.93        57
           g       0.92      0.99      0.95        84

    accuracy                           0.94       141
   macro avg       0.95      0.93      0.94       141
weighted avg       0.95      0.94      0.94       141
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e467e20d0>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
175
176
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

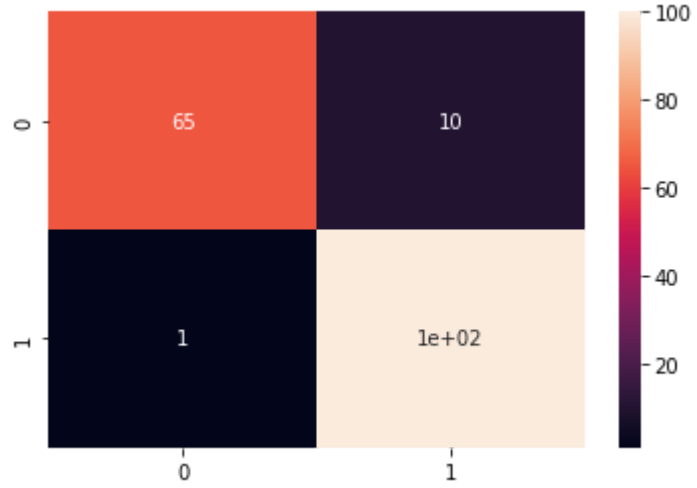
```
SVC()
```

```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.75%

Confusion Matrix:
 [[ 65  10]
 [  1 100]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.87      0.92        75
           g       0.91      0.99      0.95       101

    accuracy                           0.94       176
   macro avg       0.95      0.93      0.93       176
weighted avg       0.94      0.94      0.94       176
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e4670d850>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
140
211
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

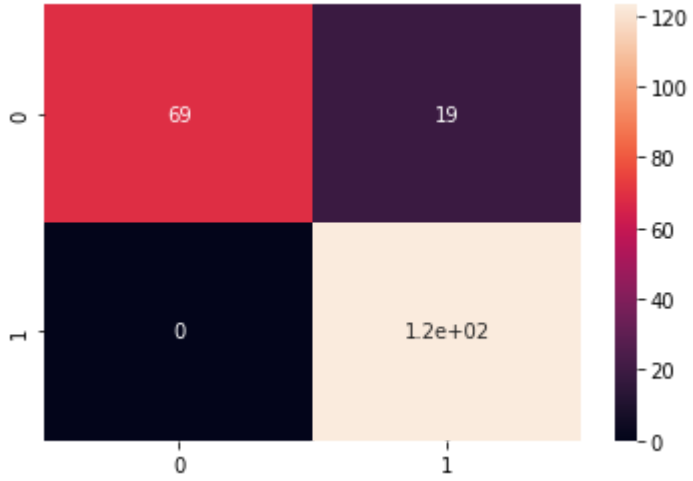
```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.99526066350711%

Confusion Matrix:
 [[ 69  19]
 [  0 123]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.78      0.88        88
           g       0.87      1.00      0.93       123

    accuracy                           0.91       211
   macro avg       0.93      0.89      0.90       211
```

```
weighted avg       0.92      0.91      0.91       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e466a2210>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

```
1 gaussain_SVC_classifier.fit(X_train, y_train)
```

```
SVC()
```

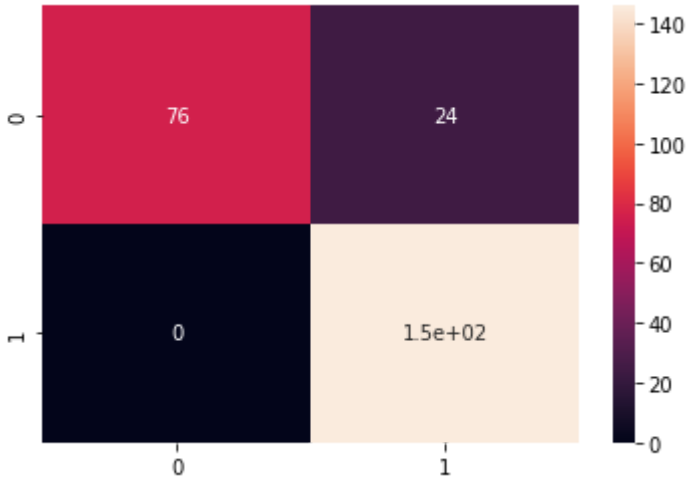
```
1 y_pred = gaussain_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.2439024390244%

Confusion Matrix:
 [[ 76  24]
 [  0 146]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.76      0.86       100
           g       0.86      1.00      0.92       146
```

```
          accuracy                           0.90       246
         macro avg       0.93      0.88      0.89       246
      weighted avg       0.92      0.90      0.90       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e465c2bd0>
```



Sigmoid SVC Classifier

```
1 sigmoid_SVC_classifier = SVC(kernel='sigmoid')
2 sigmoid_SVC_classifier
```

```
SVC(kernel='sigmoid')
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

```
SVC(kernel='sigmoid')
```

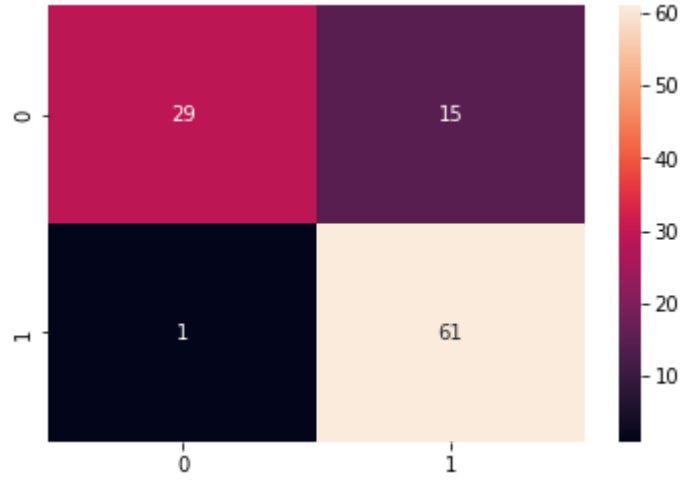
```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
```

```
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 84.90566037735849%

Confusion Matrix:
 [[29 15]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.66      0.78        44
           g       0.80      0.98      0.88        62

    accuracy                           0.85       106
   macro avg       0.88      0.82      0.83       106
weighted avg       0.87      0.85      0.84       106
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46504a90>
```



train size : test size = 60% : 40%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

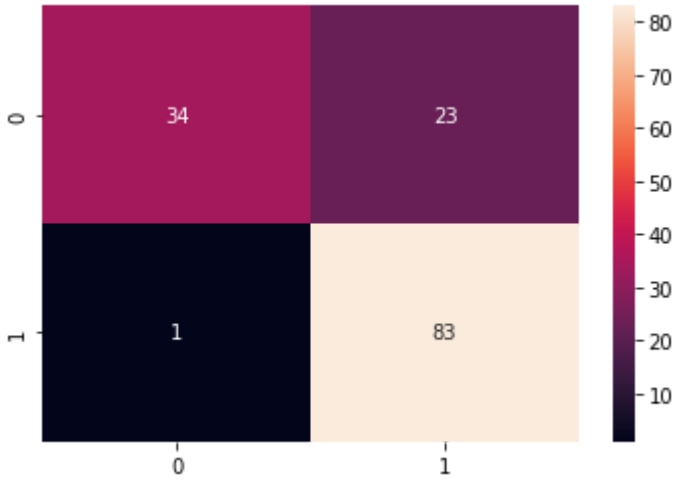
```
SVC(kernel='sigmoid')
```

```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 82.97872340425532%

Confusion Matrix:
 [[34 23]
 [ 1 83]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.60      0.74        57
           g       0.78      0.99      0.87        84

    accuracy                           0.83       141
   macro avg       0.88      0.79      0.81       141
weighted avg       0.86      0.83      0.82       141
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46498ad0>
```



train size : test size = 50% : 50%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

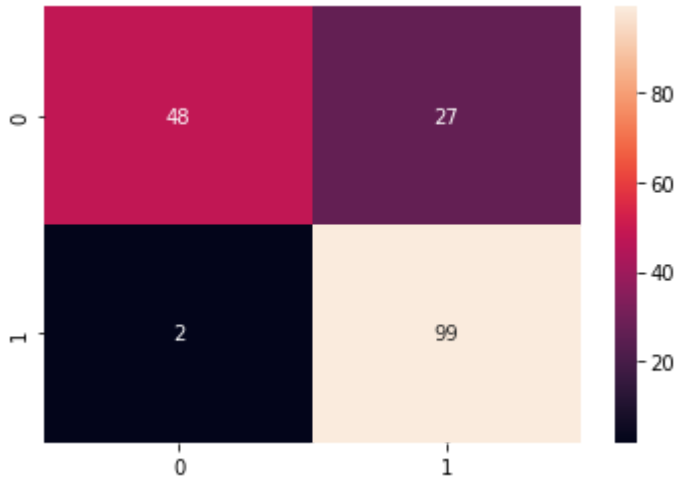
```
SVC(kernel='sigmoid')
```

```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 83.52272727272727%

Confusion Matrix:
 [[48 27]
 [ 2 99]]

Classification Report:

              precision    recall  f1-score   support

           b       0.96      0.64      0.77        75
           g       0.79      0.98      0.87       101

    accuracy                           0.84       176
   macro avg       0.87      0.81      0.82       176
weighted avg       0.86      0.84      0.83       176
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e463ccb50>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
```

```
2 print(len(y_test))
```

```
140
211
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

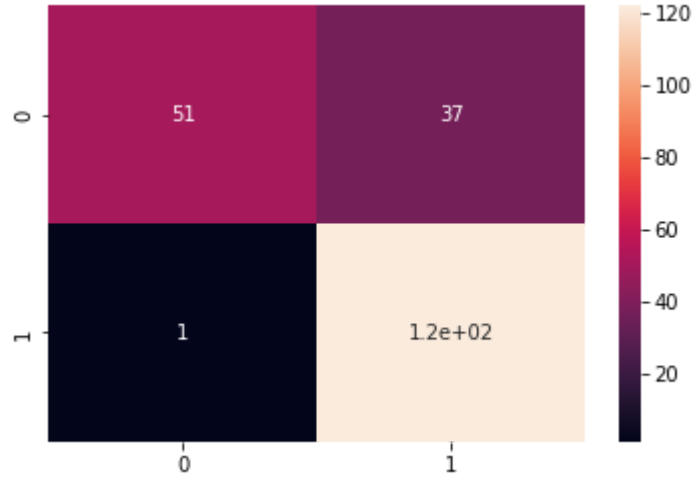
```
SVC(kernel='sigmoid')
```

```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 81.99052132701422%

Confusion Matrix:
 [[ 51  37]
 [  1 122]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.58      0.73        88
           g       0.77      0.99      0.87       123

    accuracy                           0.82       211
   macro avg       0.87      0.79      0.80       211
weighted avg       0.86      0.82      0.81       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e4634b450>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

```
1 sigmoid_SVC_classifier.fit(X_train, y_train)
```

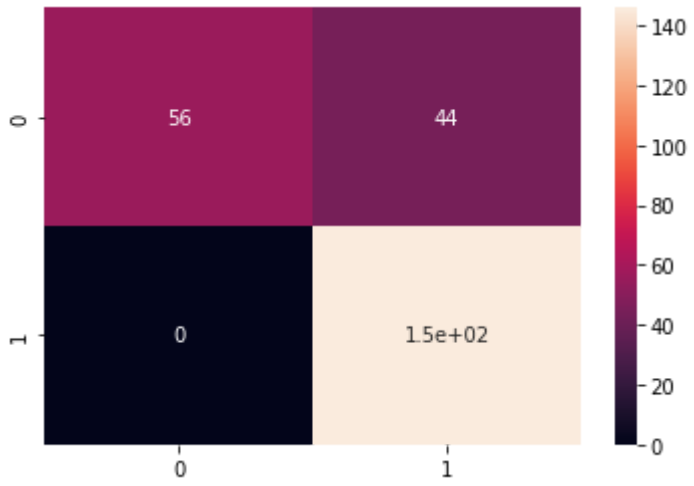
```
SVC(kernel='sigmoid')
```

```
1 y_pred = sigmoid_SVC_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n", cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 82.11382113821138%

Confusion Matrix:
 [[ 56  44]
 [  0 146]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.56      0.72       100
           g       0.77      1.00      0.87       146

    accuracy                           0.82       246
   macro avg       0.88      0.78      0.79       246
weighted avg       0.86      0.82      0.81       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e463d51d0>
```



Decision Tree

```
1 from sklearn.tree import DecisionTreeClassifier
2 clf = DecisionTreeClassifier()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 91.50943396226415%

Confusion Matrix:

[[36  8]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.82      0.89        44
           g       0.88      0.98      0.93        62

    accuracy                           0.92       106
   macro avg       0.93      0.90      0.91       106
weighted avg       0.92      0.92      0.91       106
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46232d50>
```

train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
210
141
```

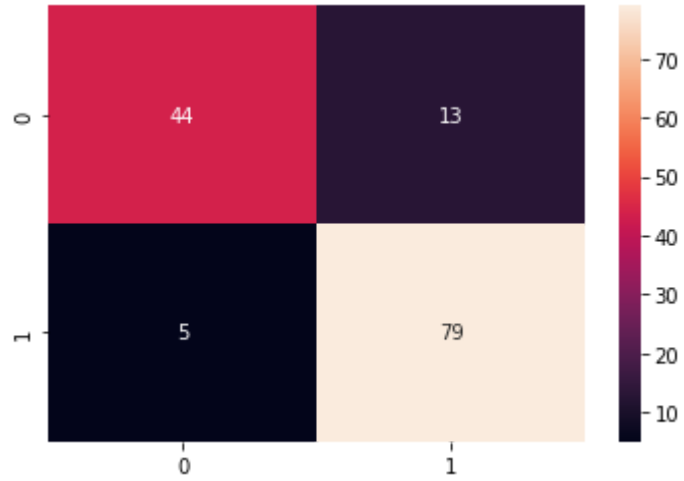
```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 87.2340425531915%

Confusion Matrix:
[[44 13]
 [ 5 79]]

Classification Report:

              precision    recall  f1-score   support

           b       0.90      0.77      0.83        57
           g       0.86      0.94      0.90        84

    accuracy                           0.87       141
   macro avg       0.88      0.86      0.86       141
weighted avg       0.87      0.87      0.87       141
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46157350>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
175
176
```

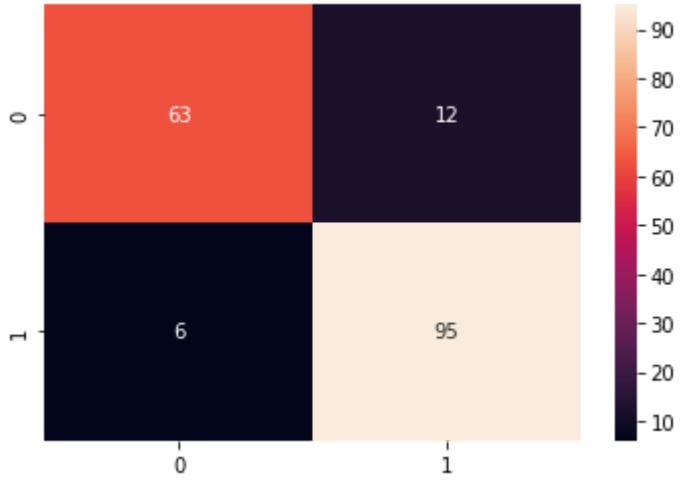
```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 89.77272727272727%

Confusion Matrix:
[[63 12]
 [ 6 95]]

Classification Report:

              precision    recall  f1-score   support

           b       0.91      0.84      0.87        75
           g       0.89      0.94      0.91       101

    accuracy                           0.90       176
   macro avg       0.90      0.89      0.89       176
weighted avg       0.90      0.90      0.90       176
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e46077d10>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
140
211
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

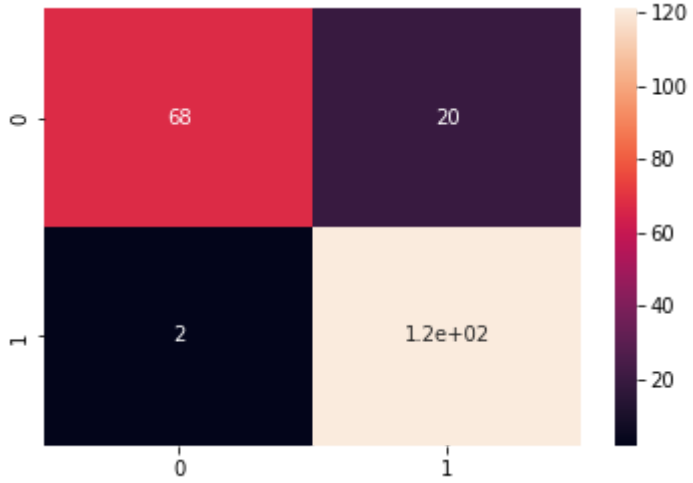
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 89.57345971563981%

Confusion Matrix:

[[ 68  20]
 [  2 121]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.77      0.86        88
           g       0.86      0.98      0.92       123
```

```
          accuracy                           0.90       211
         macro avg       0.91      0.88      0.89       211
      weighted avg       0.91      0.90      0.89       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e460254d0>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

```
1 clf = clf.fit(X_train,y_train)
2 y_pred = clf.predict(X_test)
```

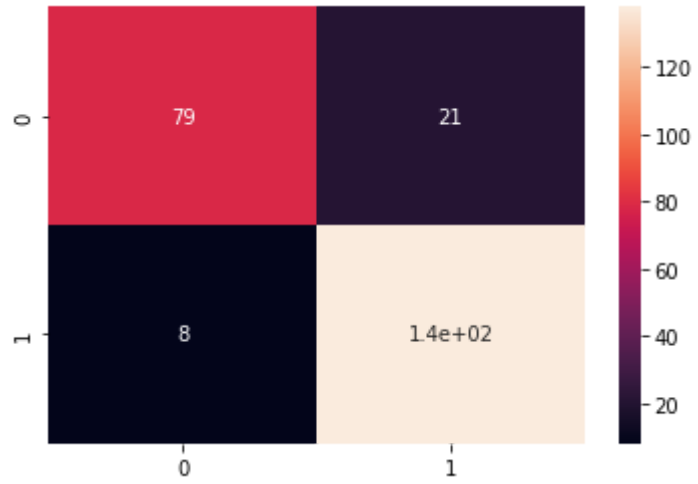
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 88.21138211382113%

Confusion Matrix:

[[ 79  21]
 [  8 138]]

Classification Report:
```

```
                precision    recall  f1-score   support

           b       0.91      0.79      0.84       100
           g       0.87      0.95      0.90       146

    accuracy                           0.88       246
   macro avg       0.89      0.87      0.87       246
weighted avg       0.88      0.88      0.88       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e478e8e90>
```



Random Forest Classifier

```
1 rfc_classifier = RandomForestClassifier(n_estimators=20)
2 rfc_classifier
```

```
RandomForestClassifier(n_estimators=20)
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 rfc_classifier.fit(X_train, y_train)
```

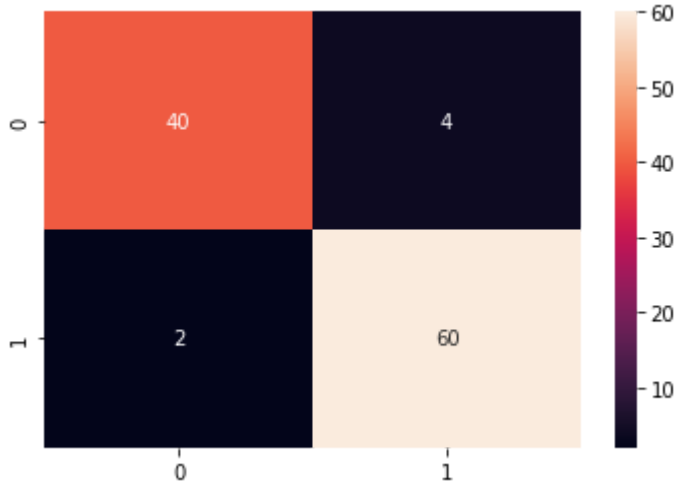
```
RandomForestClassifier(n_estimators=20)
```

```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.33962264150944%

Confusion Matrix:

[[40  4]
 [ 2 60]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.91      0.93        44
           g       0.94      0.97      0.95        62

    accuracy                           0.94       106
   macro avg       0.94      0.94      0.94       106
weighted avg       0.94      0.94      0.94       106
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45ecc090>
```



train size : test size = 60% : 40%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

```
1 rfc_classifier.fit(X_train, y_train)
```

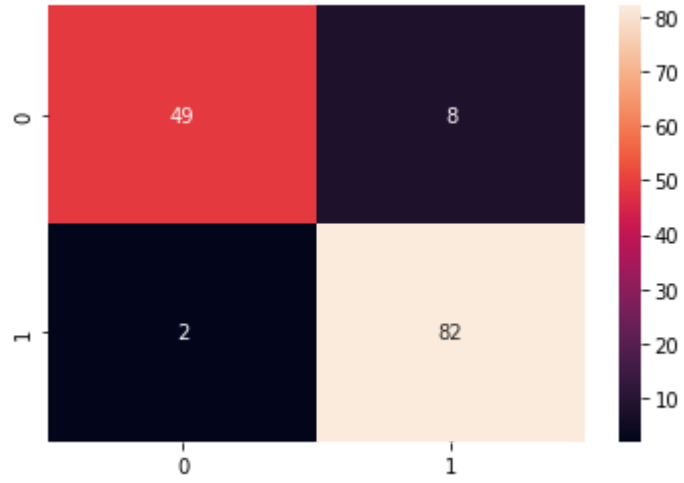
```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 92.90780141843972%

Confusion Matrix:
[[49  8]
 [ 2 82]]

Classification Report:

              precision    recall  f1-score   support

           b       0.96      0.86      0.91        57
           g       0.91      0.98      0.94        84

    accuracy                           0.93       141
   macro avg       0.94      0.92      0.92       141
weighted avg       0.93      0.93      0.93       141
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45eafa50>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
175
176
```

```
1 rfc_classifier.fit(X_train, y_train)
```

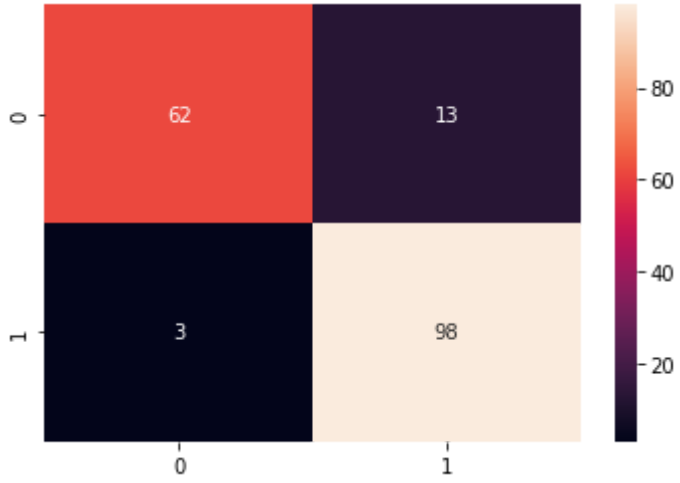
```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.9090909090909%

Confusion Matrix:
[[62 13]
 [ 3 98]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.83      0.89        75
           g       0.88      0.97      0.92       101

    accuracy                           0.91       176
   macro avg       0.92      0.90      0.91       176
weighted avg       0.91      0.91      0.91       176
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45d854d0>
```



train size : test size = 40% : 60%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
140
211
```

```
1 rfc_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.99526066350711%

Confusion Matrix:

[[ 74  14]
 [  5 118]]

Classification Report:

              precision    recall  f1-score   support

           b       0.94      0.84      0.89        88
           g       0.89      0.96      0.93       123

    accuracy                           0.91       211
   macro avg       0.92      0.90      0.91       211
weighted avg       0.91      0.91      0.91       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45d27a50>
```

train size : test size = 30% : 70%


```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

```
1 rfc_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

```
1 y_pred = rfc_classifier.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.65040650406505%

Confusion Matrix:

[[ 81  19]
 [  4 142]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.81      0.88       100
           g       0.88      0.97      0.93       146

    accuracy                           0.91       246
   macro avg       0.92      0.89      0.90       246
weighted avg       0.91      0.91      0.90       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45c471d0>
```

Naive Bayes

```
1 from sklearn.naive_bayes import GaussianNB
2 gnb = GaussianNB()
```

train size : test size = 70% : 30%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
245
106
```

```
1 gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

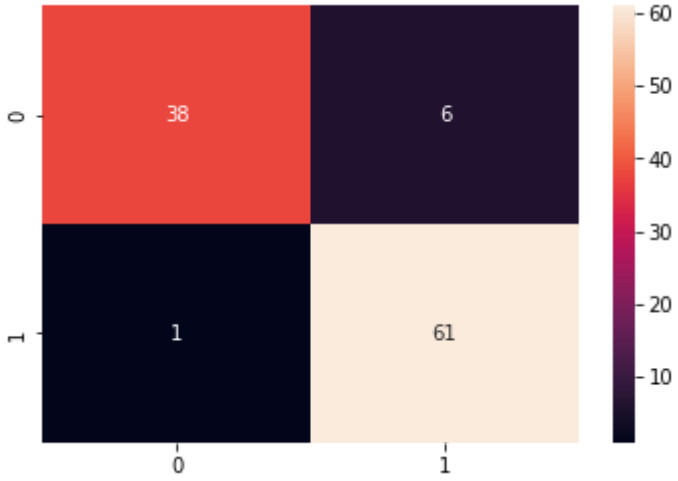
```
1 y_pred = gnb.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:\n")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.39622641509435%

Confusion Matrix:

[[38  6]
 [ 1 61]]

Classification Report:
```

```
                precision    recall  f1-score   support

           b       0.97      0.86      0.92        44
           g       0.91      0.98      0.95        62

    accuracy                           0.93       106
   macro avg       0.94      0.92      0.93       106
weighted avg       0.94      0.93      0.93       106
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45bf0a90>
```



train size : test size = 60% : 40%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
210
141
```

```
1 gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

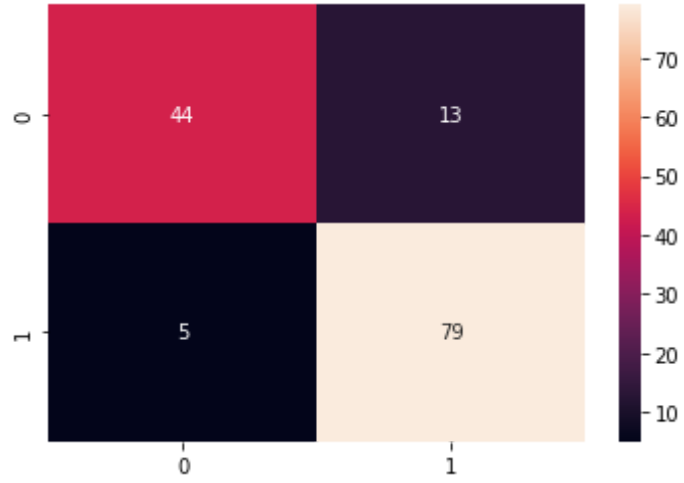
```
1 y_pred = gnb.predict(X_test)
2 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
3 cf_matrix = confusion_matrix(y_test,y_pred)
4 print("Confusion Matrix:")
5 print(cf_matrix)
6 print("\nClassification Report:\n")
7 print(classification_report(y_test,y_pred))
```

```
Accuracy: 87.2340425531915%
```

```
Confusion Matrix:
[[44 13]
 [ 5 79]]

Classification Report:

              precision    recall  f1-score   support

           b       0.90      0.77      0.83        57
           g       0.86      0.94      0.90        84

    accuracy                           0.87       141
   macro avg       0.88      0.86      0.86       141
weighted avg       0.87      0.87      0.87       141
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45b1cdd0>
```



train size : test size = 50% : 50%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
175
176
```

```
1 gnb.fit(X_train, y_train)
2 y_pred = gnb.predict(X_test)
```

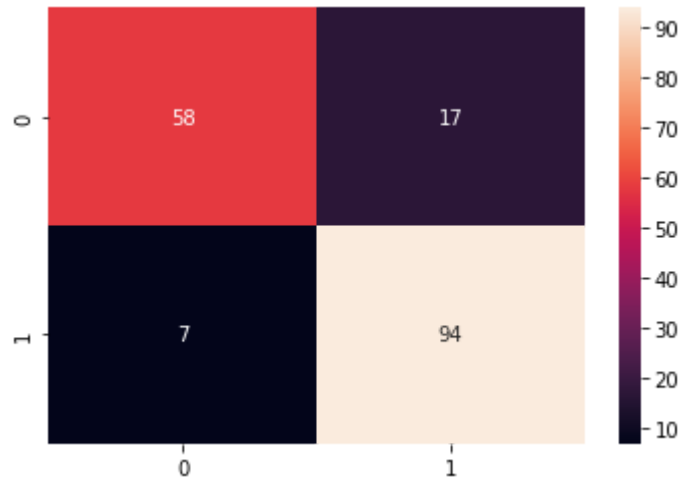
```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:")
4 print(cf_matrix)
```

```
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 86.36363636363636%

Confusion Matrix:
[[58 17]
 [ 7 94]]

Classification Report:

              precision    recall  f1-score   support

           b       0.89      0.77      0.83        75
           g       0.85      0.93      0.89       101

    accuracy                           0.86       176
   macro avg       0.87      0.85      0.86       176
weighted avg       0.87      0.86      0.86       176
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e45a44990>
```



train size : test size = 40% : 60%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=0
```

```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

```
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
```

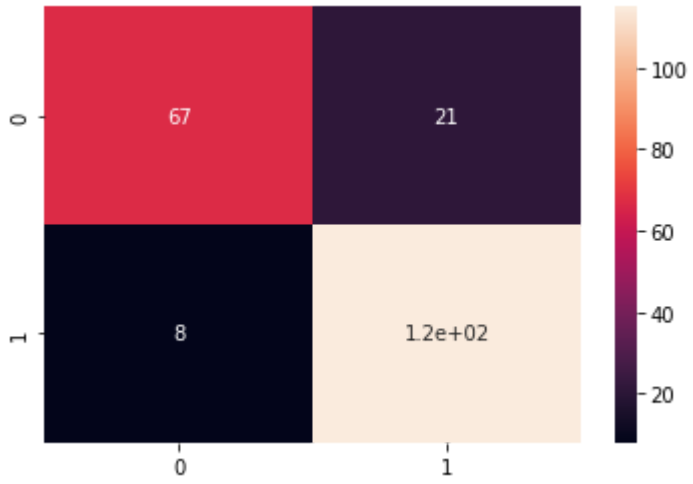
```
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 86.25592417061611%

Confusion Matrix:

[[ 67  21]
 [  8 115]]

Classification Report:

              precision    recall  f1-score   support

           b       0.89      0.76      0.82        88
           g       0.85      0.93      0.89       123

    accuracy                           0.86       211
   macro avg       0.87      0.85      0.86       211
weighted avg       0.87      0.86      0.86       211
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e459bab10>
```



train size : test size = 30% : 70%

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0
```

```
1 print(len(X_train))
2 print(len(y_test))
```

```
105
246
```

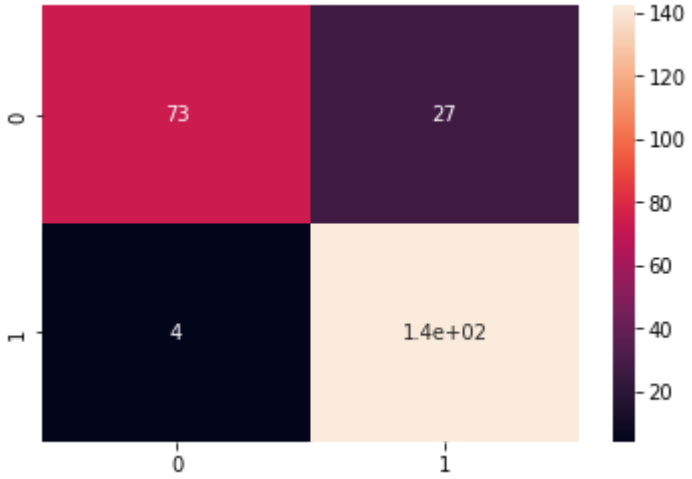
```
1 gnb.fit(X_train, y_train)
2 y_pred = gnb.predict(X_test)
```

```
1 print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
2 cf_matrix = confusion_matrix(y_test,y_pred)
3 print("Confusion Matrix:\n")
4 print(cf_matrix)
5 print("\nClassification Report:\n")
6 print(classification_report(y_test,y_pred))
```

```
Accuracy: 87.39837398373984%

Confusion Matrix:

[[ 73  27]
 [  4 142]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.73      0.82       100
           g       0.84      0.97      0.90       146

    accuracy                           0.87       246
   macro avg       0.89      0.85      0.86       246
weighted avg       0.88      0.87      0.87       246
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e458f7d50>
```



Question 2 starts :

Name --> Rishik Varma

Roll --> 001811001050

Sub --> ML Lab Evaluation Question Number - 2

Year --> IT 4th Yr 1st sem

```
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from pprint import pprint
from sklearn.model_selection import GridSearchCV
```

Iris Dataset

```


iris = datasets.load_iris()


X=iris.data
Y=[ iris.target_names[x] for x in iris.target]

X_train, X_test, y_train, y_test = train_test_split(X,Y,train_size=0.7,test_size=0.30,r


# Feature Scaling


sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```


classifier = MLPClassifier()
classifier.fit(X_train,y_train)
```

```
 5
 6 y_pred = classifier.predict(X_test)
 7
 8
 9
10 print("Confusion Matrix:")
11 print(confusion_matrix(y_test, y_pred))
12
13 print("----------------------------------------------------")
14 print("----------------------------------------------------")
15
16
17 print("Performance Evaluation")
18 print(classification_report(y_test, y_pred))
19
20
21 print("----------------------------------------------------")
22 print("----------------------------------------------------")
23
24 print("Accuracy:")
25 print(accuracy_score(y_test, y_pred))
26
27
28 plot_confusion_matrix(classifier, X_test, y_test)
29 plt.show()
```

```
Confusion Matrix:
[[13  1  0]
 [ 0 17  0]
 [ 0  0 14]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

      setosa       1.00      0.93      0.96        14
```

Diabetes Dataset

```
      macro avg       0.98      0.98      0.98        45
```

```
1
2
3 # preparing the dataset
4
5 dataset = datasets.load_diabetes()
6
7 X = np.delete(dataset.data,1,1)
8 y = dataset.data[:,1]
9
10 # as in the dataset Male or Female is not mentioned properly so we assume the first uni
11
12 data_sex_type = np.unique(y);
13 y = list(map(lambda x : 'M' if x == data_sex_type[0] else 'F' , y));
14
15 target_name = ['M','F']
16 feature_name = list(filter(lambda x : x != 'sex',dataset.feature_names));
17
18 X_train , X_test , y_train , y_test = train_test_split(X,y,test_size = 0.3)
```

```
   virginica          0          0         14
```

```
1 # Classification
2
3 classifier = MLPClassifier(max_iter=100)
4
5 ###############################################################################
6 # Showing all the parameters
7
8
9 # Look at parameters used by our current forest
10 print('Parameters currently in use:\n')
11 pprint(classifier.get_params())
12
13 ###############################################################################
14 # Creating a set of important sample features
15
16 parameter_space = {
17     'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
18     'activation': ['tanh', 'relu'],
19     'solver': ['sgd', 'adam'],
20     'alpha': [0.0001, 0.05],
21     'learning_rate': ['constant','adaptive'],
```

```
}
pprint(parameter_space)

##############################################################################


# Use the random grid to search for best hyperparameters
# First create the base model to tune
classifier = MLPClassifier(max_iter=100)
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores

rf_random = GridSearchCV(classifier, parameter_space, n_jobs=-1, cv=3)
rf_random.fit(X_train, y_train)

y_pred = rf_random.predict(X_test)


print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("-------------------------------------------------------")
print("-------------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, y_pred))


print("-------------------------------------------------------")
print("-------------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, y_pred))


plot_confusion_matrix(rf_random, X_test, y_test)
plt.show()
```

```
Parameters currently in use:

{'activation': 'relu',
 'alpha': 0.0001,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_fun': 15000,
 'max_iter': 100,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': None,
 'shuffle': True,
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
{'activation': ['tanh', 'relu'],
 'alpha': [0.0001, 0.05],
 'hidden_layer_sizes': [(50, 50, 50), (50, 100, 50), (100,)],
 'learning_rate': ['constant', 'adaptive'],
 'solver': ['sgd', 'adam']}
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
  warnings.warn(msg, category=FutureWarning)
Confusion Matrix:
[[47 18]
 [25 43]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           F       0.65      0.72      0.69        65
           M       0.70      0.63      0.67        68

    accuracy                           0.68       133
   macro avg       0.68      0.68      0.68       133
weighted avg       0.68      0.68      0.68       133

--------------------------------------------------
--------------------------------------------------
Accuracy:
```

Wisconsin Breast Cancer Dataset

1

```
 2
 3 # Dataset Preparation
 4 dataset = datasets.load_breast_cancer()
 5 df = pd.DataFrame(data=dataset.data,columns=dataset.feature_names)
 6 df['target'] = pd.DataFrame(data= np.array([ dataset.target_names[x] for x in dataset.t
 7
 8 X = df.drop(['target'], axis=1)
 9 y = df['target']
10
11
12
13 X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7,test_size=0.30,r
14
15
16 # Feature Scaling
17
18
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Classification using MLP
24
25
26 classifier = MLPClassifier()
27 classifier.fit(X_train,y_train)
28
29 y_pred = classifier.predict(X_test)
30
31
32 print("Confusion Matrix:")
33 print(confusion_matrix(y_test, y_pred))
34
35 print("-------------------------------------------------")
36 print("-------------------------------------------------")
37
38
39 print("Performance Evaluation")
40 print(classification_report(y_test, y_pred))
41
42
43 print("-------------------------------------------------")
44 print("-------------------------------------------------")
45
46 print("Accuracy:")
47 print(accuracy_score(y_test, y_pred))
48
49 plot_confusion_matrix(classifier, X_test, y_test)
50 plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
  warnings.warn(msg, category=FutureWarning)
Confusion Matrix:
[[111   1]
 [  2  57]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

      benign       0.98      0.99      0.99       112
   malignant       0.98      0.97      0.97        59

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
```

1

Question 5 starts :

Name --> Rishik Varma

Roll --> 001811001050

Sub --> ML Lab Evaluation Question Number - 5

Year --> IT 4th Yr 1st sem

K-means versus K-medoids

K-means

```
1 import numpy as np
2 import pandas as pd
3 import sklearn as sk
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 %matplotlib inline
7 from sklearn.cluster import KMeans
8 from sklearn.datasets import load_wine
```

```
1 wine=load_wine()
2 wine
```

```
{'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n------------------------\n\
 'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
```

```
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline'],
'frame': None,
'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2]),
'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7')}
```

```
1 x=wine.data
```

```
1 df=pd.DataFrame(data=wine.data, columns=['alcohol',
2   'malic_acid',
3   'ash',
4   'alcalinity_of_ash',
5   'magnesium',
6   'total_phenols',
7   'flavanoids',
8   'nonflavanoid_phenols',
9   'proanthocyanins',
10  'color_intensity',
11  'hue',
12  'od280/od315_of_diluted_wines',
13  'proline'])
14 df
```

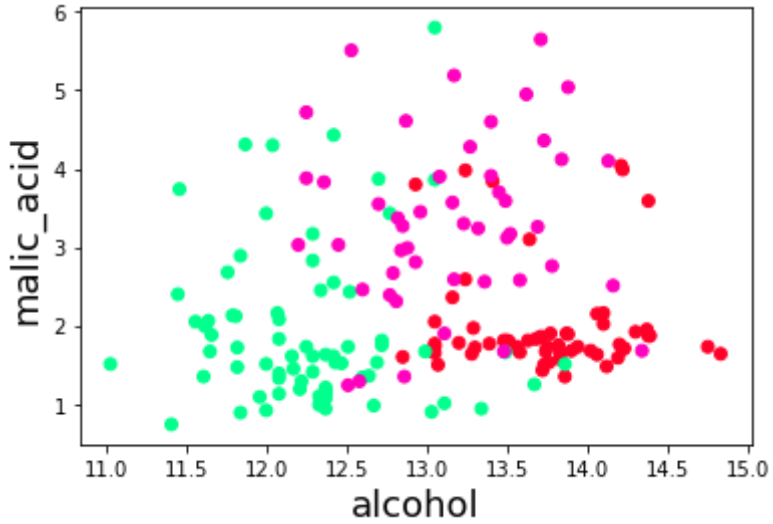
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoi
0	14.23	1.71	2.43	15.6	127.0	2.80	3.
1	13.20	1.78	2.14	11.2	100.0	2.65	2.

Double-click (or enter) to edit

3	14.37	1.95	2.50	16.8	113.0	3.85	3.

```
1 plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=wine.target, cmap='gist_rainbow')
2
3 plt.xlabel('alcohol', fontsize=18)
4 plt.ylabel('malic_acid', fontsize=18)
```

```
Text(0, 0.5, 'malic_acid')
```



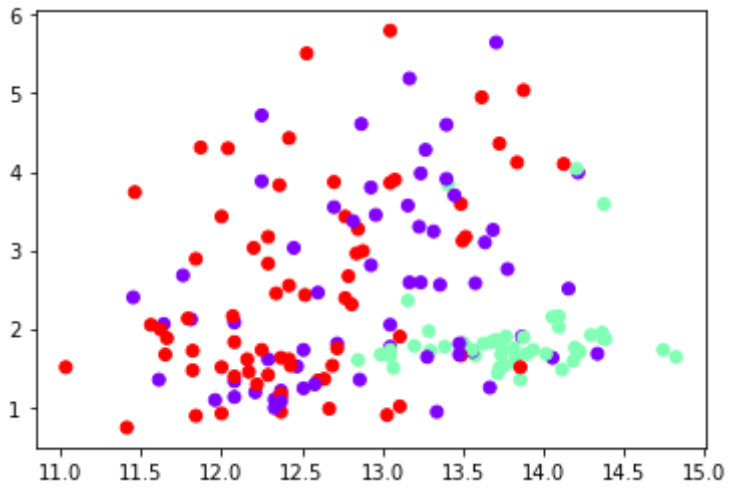

```
1 kmeans = KMeans(init="random", n_clusters=3, n_init=10, max_iter=300, random_state=42)
2 y = kmeans.fit_predict(x)
```

```
1 print("K-Means Cluster Centers")
2 print(kmeans.cluster_centers_)
3 print("Cluster Labels")
4 print(kmeans.labels_)
```

```
K-Means Cluster Centers
[[1.29298387e+01 2.50403226e+00 2.40806452e+00 1.98903226e+01
  1.03596774e+02 2.11112903e+00 1.58403226e+00 3.88387097e-01
  1.50338710e+00 5.65032258e+00 8.83967742e-01 2.36548387e+00
  7.28338710e+02]
 [1.38044681e+01 1.88340426e+00 2.42617021e+00 1.70234043e+01
  1.05510638e+02 2.86723404e+00 3.01425532e+00 2.85319149e-01
  1.91042553e+00 5.70255319e+00 1.07829787e+00 3.11404255e+00
  1.19514894e+03]
 [1.25166667e+01 2.49420290e+00 2.28855072e+00 2.08231884e+01
  9.23478261e+01 2.07072464e+00 1.75840580e+00 3.90144928e-01
  1.45188406e+00 4.08695651e+00 9.41159420e-01 2.49072464e+00
  4.58231884e+02]]
Cluster Labels
[1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0
 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 0 2 2 0 2 2 0 0 0 2 2 1
 0 2 2 2 0 2 2 0 0 2 2 2 2 2 2 0 0 2 2 2 2 2 0 0 2 0 2 0 2 2 2 0 2 2 2 2 0 2
```

```
2 0 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 0 2 2 0 0 0 0 2 2 2 0 0 2 2 0 0 2 0
0 2 2 2 2 0 0 0 2 0 0 0 2 0 2 0 0 2 0 0 0 0 2 2 0 0 0 0 0 0 2]
```

```
1 plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=kmeans.labels_, cmap='rainbow')
2 plt.show()
```




```
 1 fig, axes = plt.subplots(1, 2, figsize=(14,6))
 2 axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k',
 3 axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target, cmap='jet',edgecolo
 4 axes[0].set_xlabel('alcohol', fontsize=18)
 5 axes[0].set_ylabel('malic_acid', fontsize=18)
 6 axes[1].set_xlabel('alcohol', fontsize=18)
 7 axes[1].set_ylabel('malic_acid', fontsize=18)
 8 axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
 9 axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
10 axes[0].set_title('Actual', fontsize=18)
11 axes[1].set_title('Predicted', fontsize=18)
12
```

```
Text(0.5, 1.0, 'Predicted')
```


```
1 from sklearn.metrics import silhouette_score
2 print("The silhouette score is :")
3 silhouette_score(x, kmeans.labels_)
```

```
The silhouette score is :
0.5711381937868844
```

```
1 from sklearn.metrics import calinski_harabasz_score
2 print("The calinski harabasz score is :  ")
3 calinski_harabasz_score(x, kmeans.labels_)
```

```
The calinski harabasz score is :
561.815657860671
```

```
1 from sklearn.metrics import davies_bouldin_score
2 print("The davies bouldin score is :  ")
3 davies_bouldin_score(x, kmeans.labels_)
```

```
The davies bouldin score is :
0.5342431775436273
```

New section

K-medoids

```
1 !pip install scikit-learn-extra
```

```
Requirement already satisfied: scikit-learn-extra in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scikit-learn>=0.23.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
```

```
1 import numpy as np
2 import pandas as pd
3 import sklearn as sk
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 %matplotlib inline
7 from sklearn_extra.cluster import KMedoids
8 from sklearn.datasets import load_iris
```

```
1 wine=load_wine()
```

```
2 wine
```

```
{'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n------------------------\n\
 'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline'],
 'frame': None,
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2]),
 'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7')}
```

```
1 x=wine.data
```

```
1
2 df=pd.DataFrame(data=wine.data, columns=['alcohol',
3   'malic_acid',
4   'ash',
5   'alcalinity_of_ash',
6   'magnesium',
7   'total_phenols',
8   'flavanoids',
9   'nonflavanoid_phenols',
10  'proanthocyanins',
11  'color_intensity',
12  'hue',
```

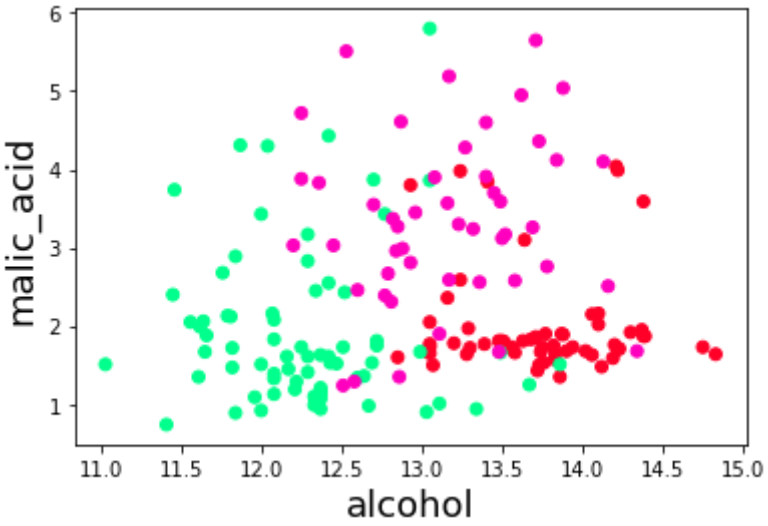
```
13   'od280/od315_of_diluted_wines',
14   'proline'])
15 df
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoi
0	14.23	1.71	2.43	15.6	127.0	2.80	3.
1	13.20	1.78	2.14	11.2	100.0	2.65	2.
2	13.16	2.36	2.67	18.6	101.0	2.80	3.
3	14.37	1.95	2.50	16.8	113.0	3.85	3.
4	13.24	2.59	2.87	21.0	118.0	2.80	2.
...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.
174	13.40	3.91	2.48	23.0	102.0	1.80	0.
175	13.27	4.28	2.26	20.0	120.0	1.59	0.
176	13.17	2.59	2.37	20.0	120.0	1.65	0.
177	14.13	4.10	2.74	24.5	96.0	2.05	0.

178 rows × 13 columns

```
1 plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=wine.target, cmap='gist_rainbow')
2
3 plt.xlabel('alcohol', fontsize=18)
4 plt.ylabel('malic_acid', fontsize=18)
```

```
Text(0, 0.5, 'malic_acid')
```



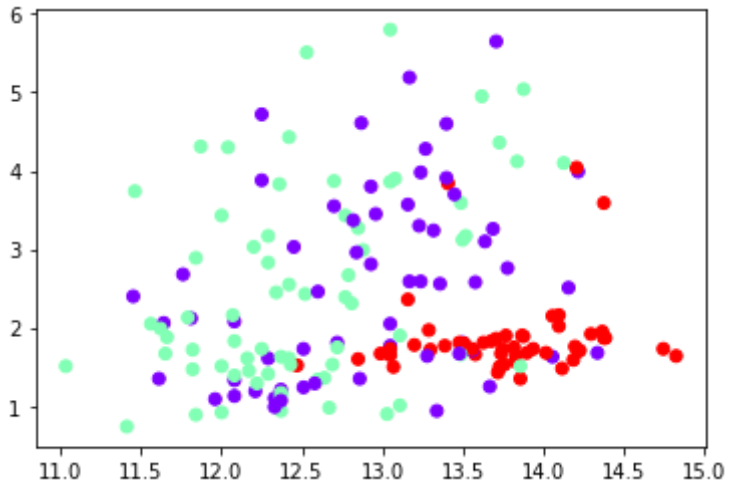

```
1 kmedoid = KMedoids(init="heuristic", n_clusters=3, max_iter=300, random_state=42)
2 y = kmedoid.fit_predict(x)
```

```
1 print("K-Medoids Cluster Centers")
2 print(kmedoid.cluster_centers_)
```

```
print("Cluster Labels")
print(kmedoid.labels_)
```

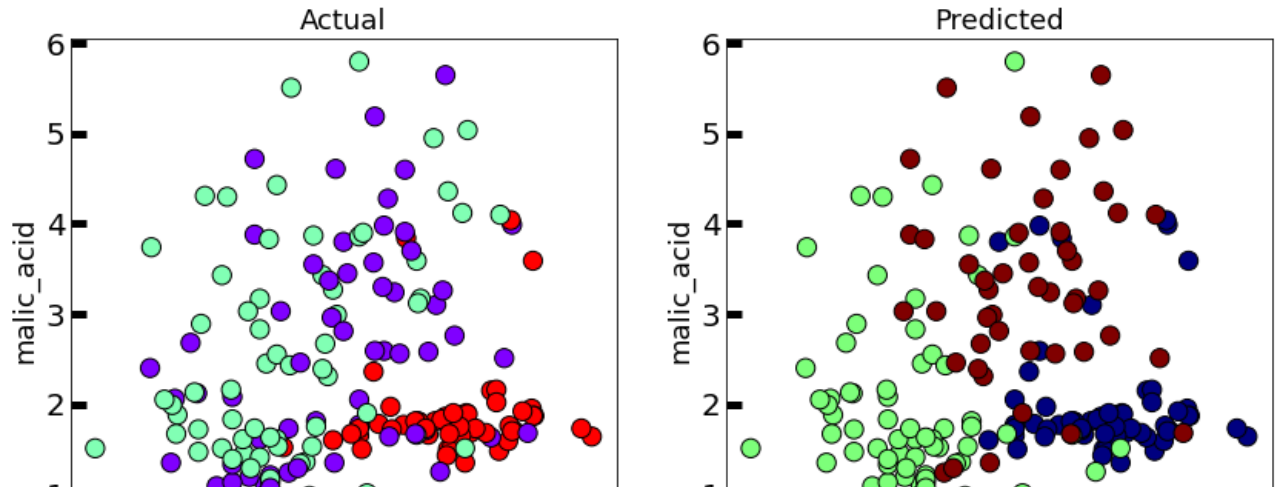
```
K-Medoids Cluster Centers
[[1.260e+01 2.460e+00 2.200e+00 1.850e+01 9.400e+01 1.620e+00 6.600e-01
  6.300e-01 9.400e-01 7.100e+00 7.300e-01 1.580e+00 6.950e+02]
 [1.349e+01 1.660e+00 2.240e+00 2.400e+01 8.700e+01 1.880e+00 1.840e+00
  2.700e-01 1.030e+00 3.740e+00 9.800e-01 2.780e+00 4.720e+02]
 [1.383e+01 1.570e+00 2.620e+00 2.000e+01 1.150e+02 2.950e+00 3.400e+00
  4.000e-01 1.720e+00 6.600e+00 1.130e+00 2.570e+00 1.130e+03]]
Cluster Labels
[2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 0 0 2 2 2 2 2 2 2 2 2 0
 2 2 0 0 2 2 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 1 0 1 0 1 1 0 1 1 0 0 0 1 1 2
 0 1 1 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 2 0 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0
 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1]
```

```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=kmedoid.labels_, cmap='rainbow') #tr
plt.show()
```




```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k',
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target, cmap='jet',edgecolo
axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18)
axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

```
Text(0.5, 1.0, 'Predicted')
```




```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, kmedoid.labels_)
```

```
The silhouette score is :
0.5666480408636575
```

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, kmedoid.labels_)
```

```
The calinski harabasz score is :
539.3792353535451
```

```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, kmedoid.labels_)
```

```
The davies bouldin score is :
0.529239412600317
```

```
print("It is observed that TSS=SSE+SSB is a constant. Hence we will calculate the TSS a

print("The value of SSE is: ")
print(kmedoid.inertia_)



# Finding the overall centroid of the data points
centers = kmedoid.cluster_centers_
center_x = []
for center in centers:
  center_x.append(center[0])
center_x
overall_center = sum(center_x)/len(center_x)

tss = 0
for i in range(len(df)):
```

```
  a = df.iloc[i][0] - overall_center
  b = pow(a,2)
  tss = tss+b



print("The value of SSB is: ")
print(tss - kmedoid.inertia_)
```

```
It is observed that TSS=SSE+SSB is a constant. Hence we will calculate the TSS ans su
The value of SSE is:
16376.969320536637
The value of SSB is:
-16243.642776092192
```

DBSCAN versus OPTICS

DBSCAN

```
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.datasets import load_wine
```

```
wine=load_wine()
wine
```

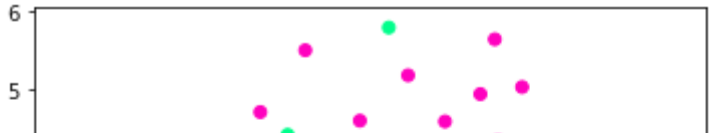
```
{'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n------------------------\n\
 'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
```

```
       'magnesium',
       'total_phenols',
       'flavanoids',
       'nonflavanoid_phenols',
       'proanthocyanins',
       'color_intensity',
       'hue',
       'od280/od315_of_diluted_wines',
       'proline'],
      'frame': None,
      'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2]),
      'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7')}
```

```
1 x=wine.data
```

```
1 df=pd.DataFrame(data=wine.data, columns=['alcohol',
2   'malic_acid',
3   'ash',
4   'alcalinity_of_ash',
5   'magnesium',
6   'total_phenols',
7   'flavanoids',
8   'nonflavanoid_phenols',
9   'proanthocyanins',
10  'color_intensity',
11  'hue',
12  'od280/od315_of_diluted_wines',
13  'proline'])
14 df
15
16 plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=wine.target, cmap='gist_rainbow') #t
17
18 plt.xlabel('alcohol', fontsize=18)
19 plt.ylabel('malic_acid', fontsize=18)
```

```
Text(0, 0.5, 'malic_acid')
```

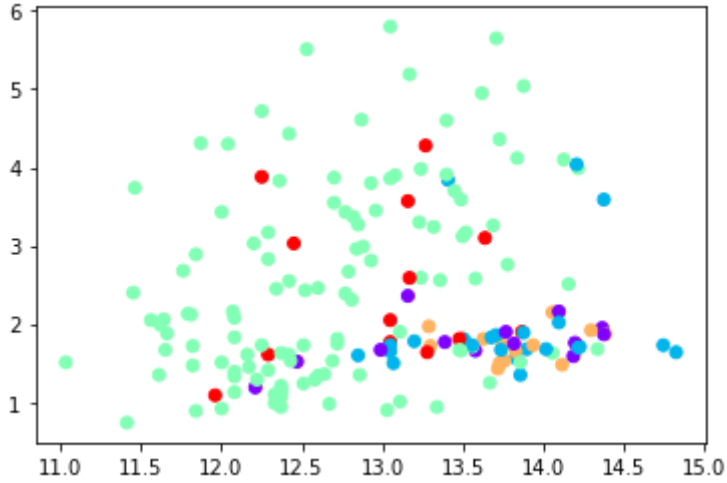


```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(x)
```

```
print("Cluster Labels")
print(dbscan.labels_)
```

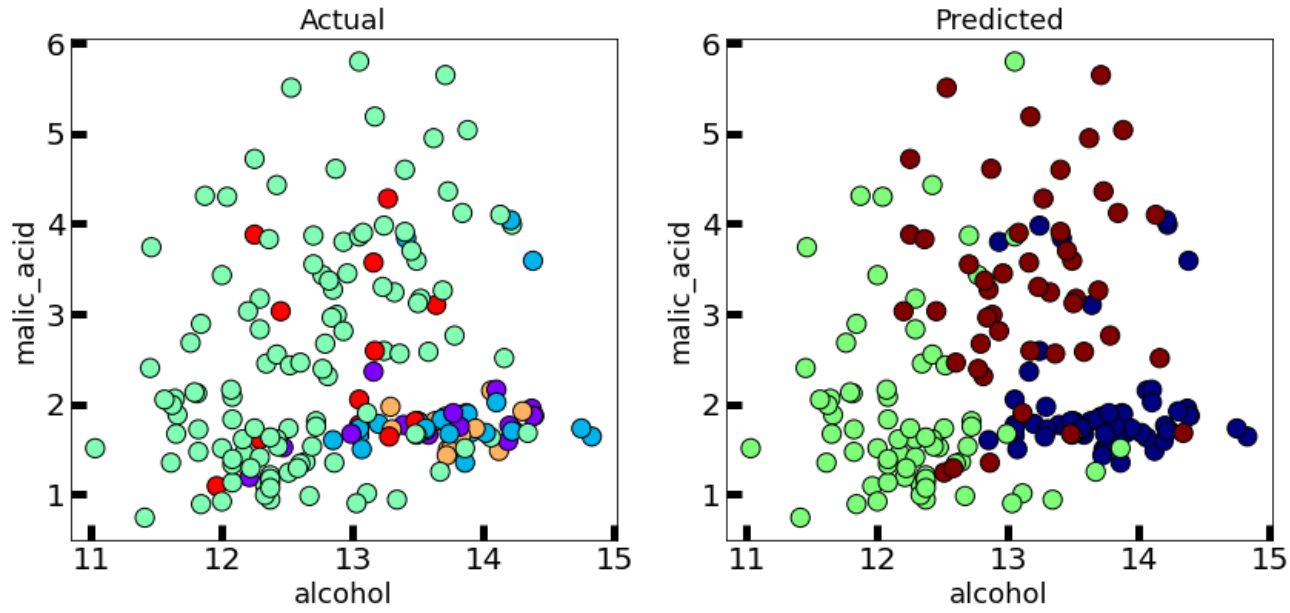
```
Cluster Labels
[ 0  0 -1 -1  1 -1  2  2  0  0 -1  2  2  0 -1  2  2  0 -1  3  1  1  0  0
  3  3 -1  2  3  0  2 -1  0  2  0  3  3  0  0  1  1  0  0  1  3  0  0  0
  0  2  0  2 -1 -1  0  0  0  2  2  1  1  1  1  1  1  1  1  1 -1  3  1
  1 -1  3  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  3  3  1  1  1  1  1  1  1  1  1  1  1  3  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  3  3  1]
```

```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=dbscan.labels_, cmap='rainbow')
plt.show()
```




```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k',
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target, cmap='jet',edgecolo
axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18)
axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

```
Text(0.5, 1.0, 'Predicted')
```




```
1 from sklearn.metrics import silhouette_score
2 print("The silhouette score is :  ")
3 silhouette_score(x, dbscan.labels_)
```

```
The silhouette score is :
0.4413295944891938
```

```
1 from sklearn.metrics import calinski_harabasz_score
2 print("The calinski harabasz score is :")
3 calinski_harabasz_score(x, dbscan.labels_)
```

```
The calinski harabasz score is :
208.9449395725058
```

```
1 from sklearn.metrics import davies_bouldin_score
2 print("The davies bouldin score is :")
3 davies_bouldin_score(x, dbscan.labels_)
```

```
The davies bouldin score is :
7.812129203041904
```

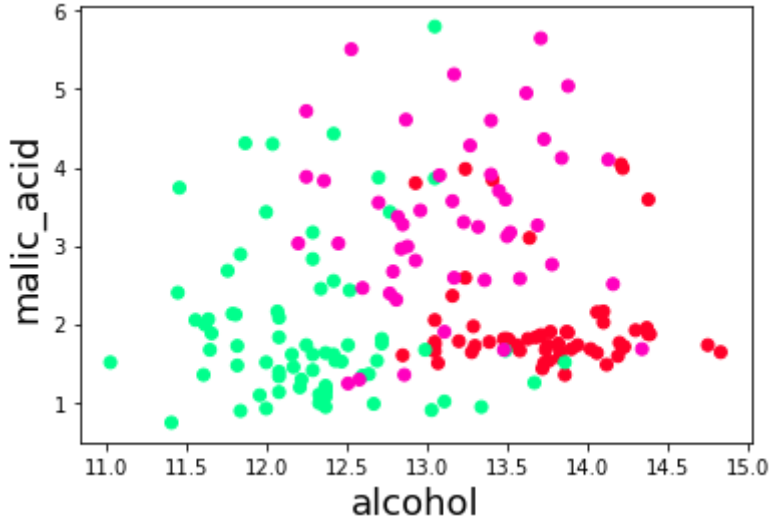
OPTICS

```
1 import numpy as np
2 import pandas as pd
3 import sklearn as sk
4 import matplotlib.pyplot as plt
5 import seaborn as sns
```

```
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.datasets import load_wine
```

```
wine=load_wine()
wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline'])
df

plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=wine.target, cmap='gist_rainbow') #t

plt.xlabel('alcohol', fontsize=18)
plt.ylabel('malic_acid', fontsize=18)
```

```
Text(0, 0.5, 'malic_acid')
```

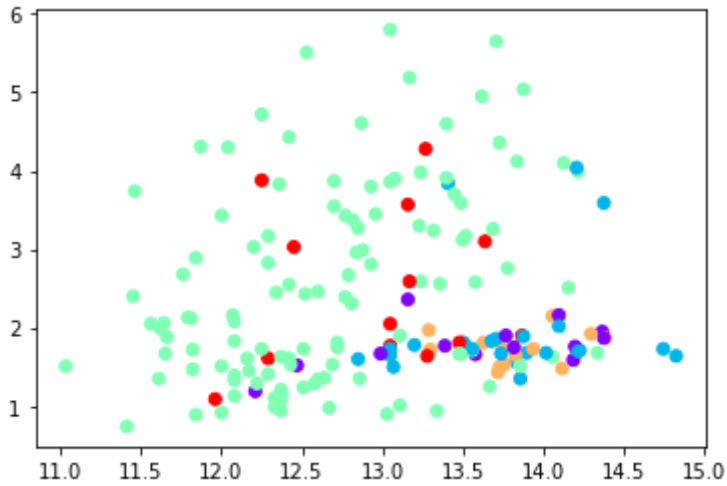



```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(x)
```

```
print("Cluster Labels")
print(dbscan.labels_)

```

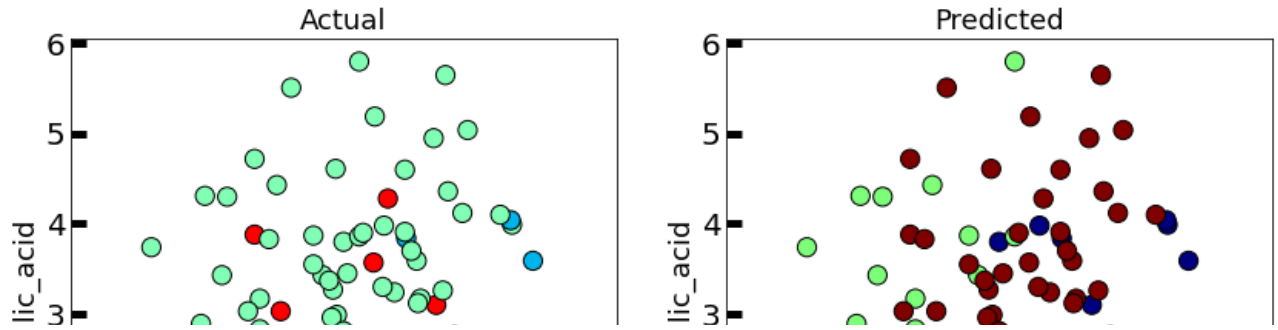
```
Cluster Labels
[ 0  0 -1 -1  1 -1  2  2  0  0 -1  2  2  0 -1  2  2  0 -1  3  1  1  0  0
  3  3 -1  2  3  0  2 -1  0  2  0  3  3  0  0  1  1  0  0  1  3  0  0  0
  0  2  0  2 -1 -1  0  0  0  2  2  1  1  1  1  1  1  1  1  1 -1  3  1
  1 -1  3  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  3  3  1  1  1  1  1  1  1  1  1  1  1  3  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  3  3  1]
```

```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=dbscan.labels_, cmap='rainbow') #try
plt.show()

```




```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k',
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target, cmap='jet',edgecolo
axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18)
axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

```
Text(0.5, 1.0, 'Predicted')
```




```
1 from sklearn.metrics import silhouette_score
2 print("The silhouette score is :")
3 silhouette_score(x, dbscan.labels_)
```

```
The silhouette score is :
0.4413295944891938
```

alcohol alcohol

```
1 from sklearn.metrics import calinski_harabasz_score
2 print("The calinski harabasz score is :")
3 calinski_harabasz_score(x, dbscan.labels_)
```

```
The calinski harabasz score is :
208.9449395725058
```

```
1 from sklearn.metrics import davies_bouldin_score
2 print("The davies bouldin score is :")
3 davies_bouldin_score(x, dbscan.labels_)
```

```
The davies bouldin score is :
7.812129203041904
```

Comparisons for the clustering Algorithms

Algorithm	Dataset	Silhouette Score	Calinski Harabasz Score	Davies Bouldin Score
K-means	WINE DATASET	0.5711381938	561.8156579	0.5342431775
K-medoids	WINE DATASET	0.5666480409	539.3792354	0.5292394126
DBSCAN	WINE DATASET	0.4413295945	208.9449396	7.812129203
OPTICS	WINE DATASET	0.4413295945	208.9449396	7.812129203