SHUVRASISH ROY
ROLL NO.: 001811001012

MACHINE LEARNING LAB
ASSIGNMENT 2

import all necessary modules:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scikitplot as skplt
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
, plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
import plotly.graph_objects as go
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

creating the functions

```
def preprocess(X,y,te_size,label=False,scale=False,pca=False):

  if label:
    y = LabelEncoder().fit_transform(y)


  X_tr,X_te,y_tr,y_te = train_test_split(X,y,test_size=te_size)

  if scale:
    sc = StandardScaler()
    X_tr = sc.fit_transform(X_tr)
    X_te = sc.transform(X_te)

  if pca:
    pca = PCA(n_components='mle')
    X_tr = pca.fit_transform(X_tr)
    X_te = pca.transform(X_te)

  return X_tr,X_te,y_tr,y_te

def tester(classi,X_t,y_t,y_p):

  print("Confusion Matrix")
  print(confusion_matrix(y_t,y_p))

  print('-----------------------------------')
  print('-----------------------------------')
```

```
print('Performance Evaluation:')
print(classification_report(y_t,y_p))

print('-----------------------------------')
print('-----------------------------------')

print('Accuracy Score:')
print(accuracy_score(y_t,y_p))

plot_confusion_matrix(classi,X_t,y_t)
plt.title('Heat map for confusion matrix')
plt.show()

y_p_proba = classifier.predict_proba(X_t)

skplt.metrics.plot_roc(y_t,y_p_proba)
plt.show()
```

WINE DATASET

Loading the data:

Using wine dataset.

```
df1 = pd.read_csv('wine.data', header=None)
X = df1.iloc[:,1:]
y = df1.iloc[:,0]
```

Creating and comparing models:

Code for 70:30 split:

We can just use params as 0.3 to 0.7 to split the data into different sized training and testing splits.

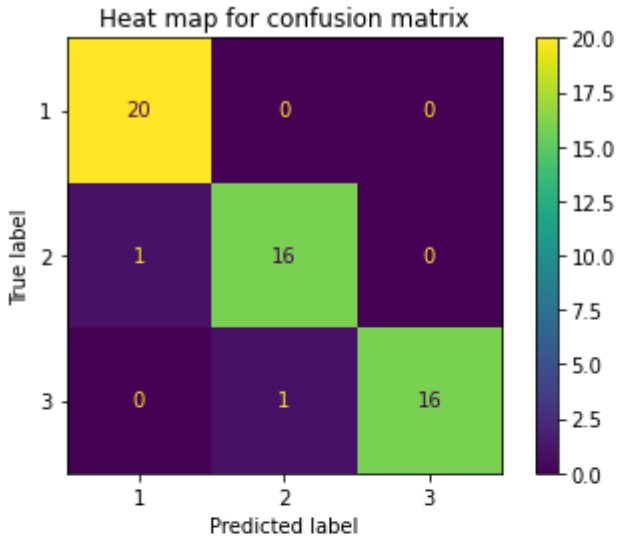
```
#70:30 split
X_train,X_test,y_train,y_test = preprocess(X,y,0.3,scale=True,pca=True)
```

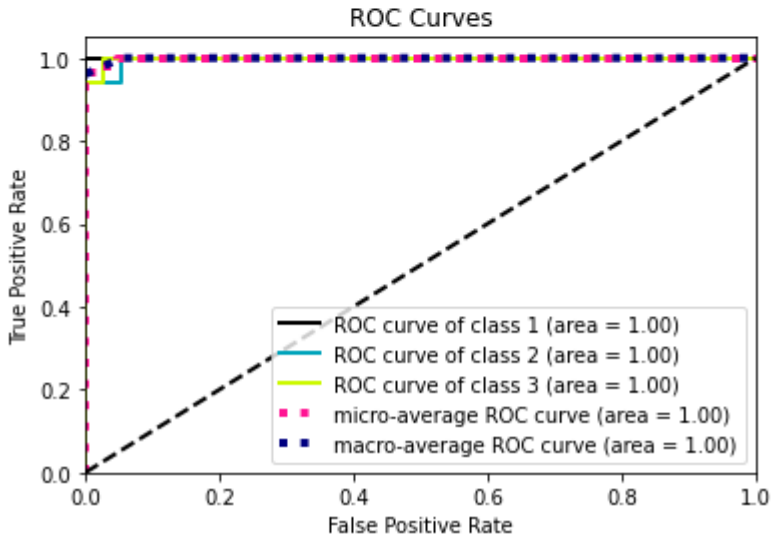
Without Parameter Tuning:

```
#linear
classifier = SVC(kernel='linear', probability=True)
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('SVC Linear:')
tester(classifier,X_test,y_test,y_pred)
#polynomial 2
classifier = SVC(kernel='poly', degree=2, probability=True)
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('SVC Polynomial degree 2:')
tester(classifier,X_test,y_test,y_pred)
#polynomial 3
classifier = SVC(kernel='poly', degree=3, probability=True)
classifier.fit(X_train,y_train)
```

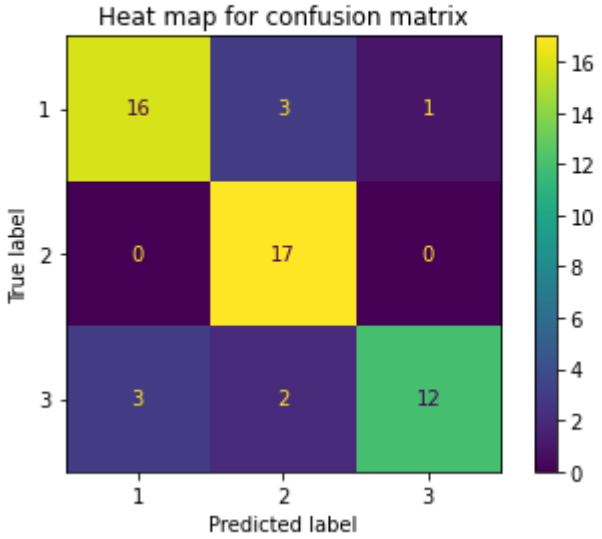
```
y_pred = classifier.predict(X_test)

print('SVC Polynomial degree 3:')
tester(classifier,X_test,y_test,y_pred)
#gaussian
classifier = SVC(kernel='rbf', probability=True)
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('SVC Gaussian:')
tester(classifier,X_test,y_test,y_pred)
#sigmoid
classifier = SVC(kernel='sigmoid', probability=True)
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('SVC Sigmoid:')
tester(classifier,X_test,y_test,y_pred)

#mlp

classifier = MLPClassifier(max_iter=500)
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('MLP:')
tester(classifier,X_test,y_test,y_pred)

#random forest
classifier=RandomForestClassifier()
classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

print('Random Forest Classifier:')
tester(classifier,X_test,y_test,y_pred)
```

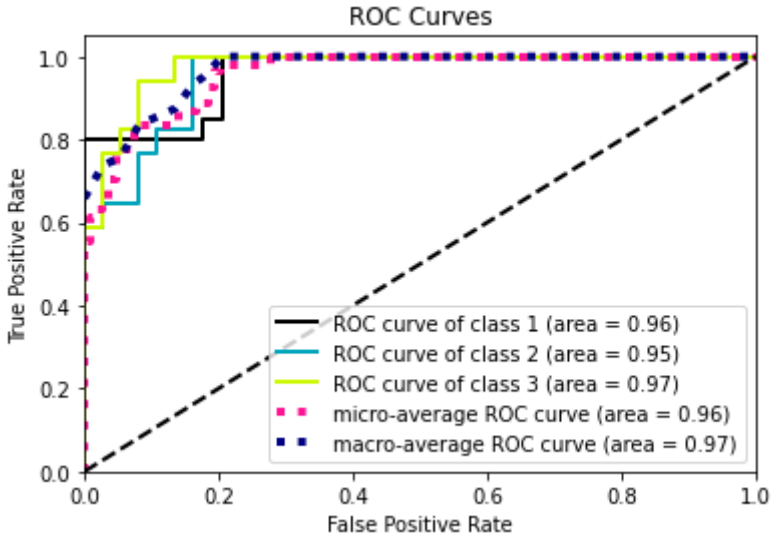
```
SVC Linear:
Confusion Matrix
[[20  0  0]
 [ 1 16  0]
 [ 0  1 16]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       0.95      1.00      0.98        20
           2       0.94      0.94      0.94        17
           3       1.00      0.94      0.97        17

    accuracy                           0.96        54
   macro avg       0.96      0.96      0.96        54
weighted avg       0.96      0.96      0.96        54

-----------------------------------
-----------------------------------
Accuracy Score:
0.9629629629629629
```



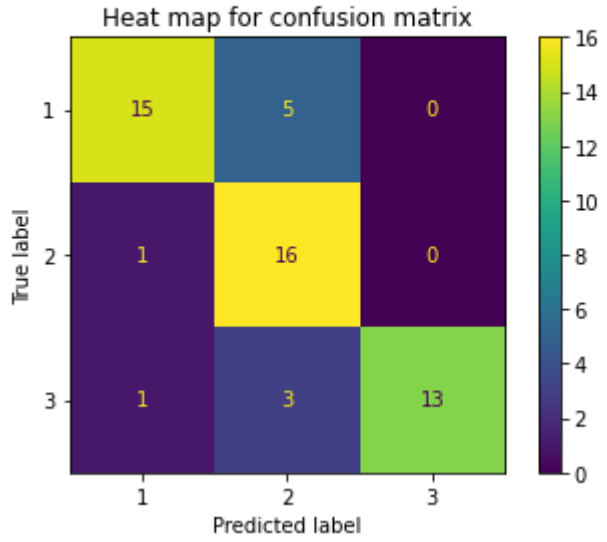


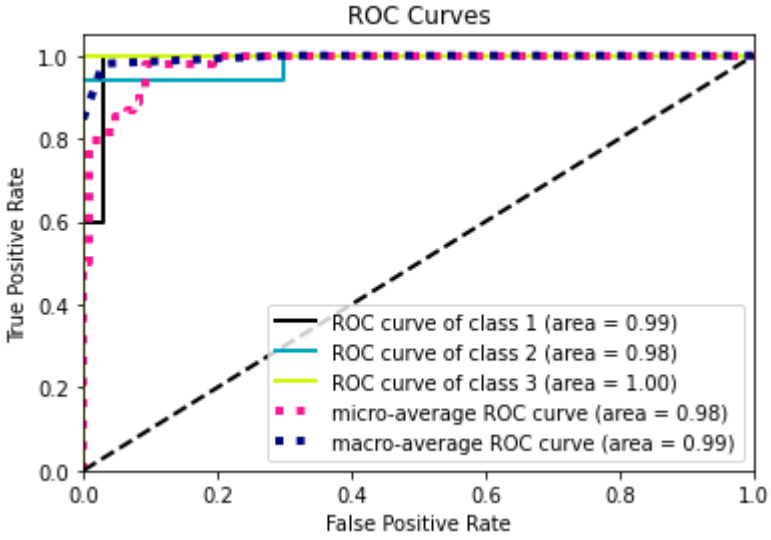
```
SVC Polynomial degree 2:
Confusion Matrix
[[16  3  1]
 [ 0 17  0]
 [ 3  2 12]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       0.84      0.80      0.82        20
           2       0.77      1.00      0.87        17
           3       0.92      0.71      0.80        17

    accuracy                           0.83        54
   macro avg       0.85      0.84      0.83        54
weighted avg       0.85      0.83      0.83        54

-----------------------------------
-----------------------------------
Accuracy Score:
0.8333333333333334
```



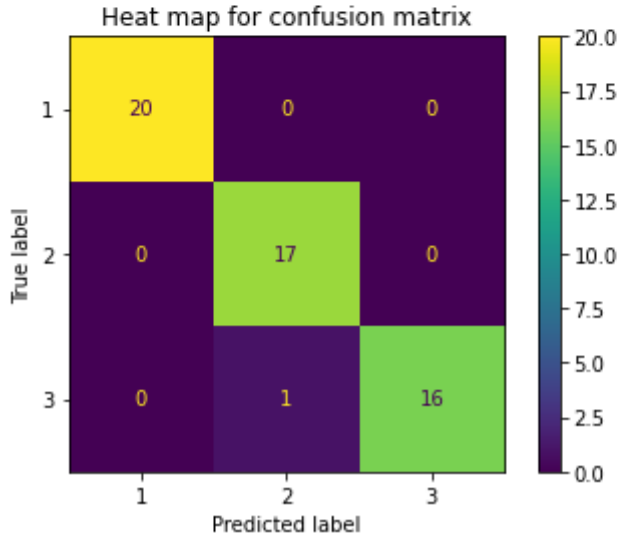


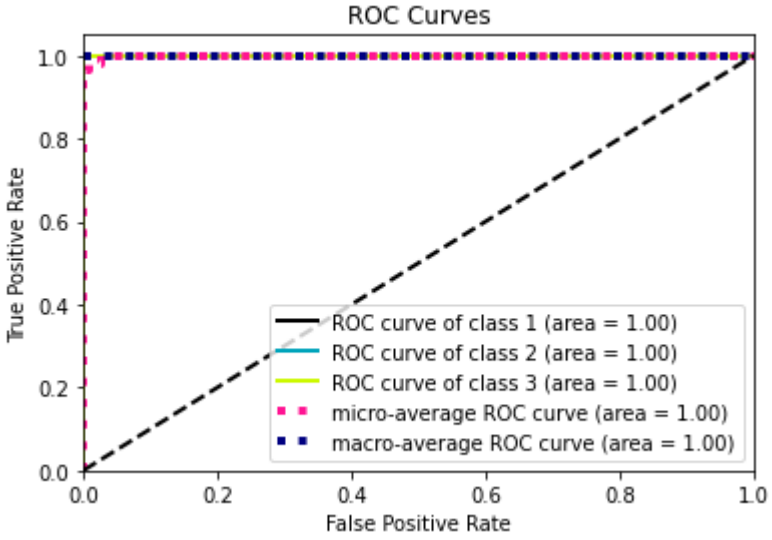
```
SVC Polynomial degree 3:
Confusion Matrix
[[15  5  0]
 [ 1 16  0]
 [ 1  3 13]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       0.88      0.75      0.81        20
           2       0.67      0.94      0.78        17
           3       1.00      0.76      0.87        17

    accuracy                           0.81        54
   macro avg       0.85      0.82      0.82        54
weighted avg       0.85      0.81      0.82        54

-----------------------------------
-----------------------------------
Accuracy Score:
0.8148148148148148
```



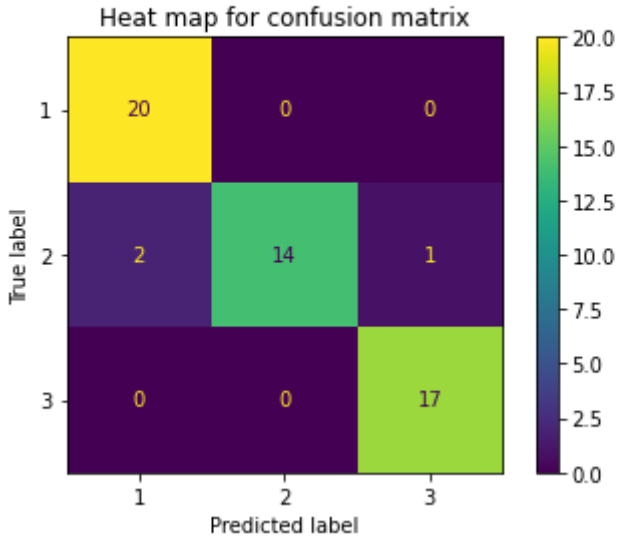


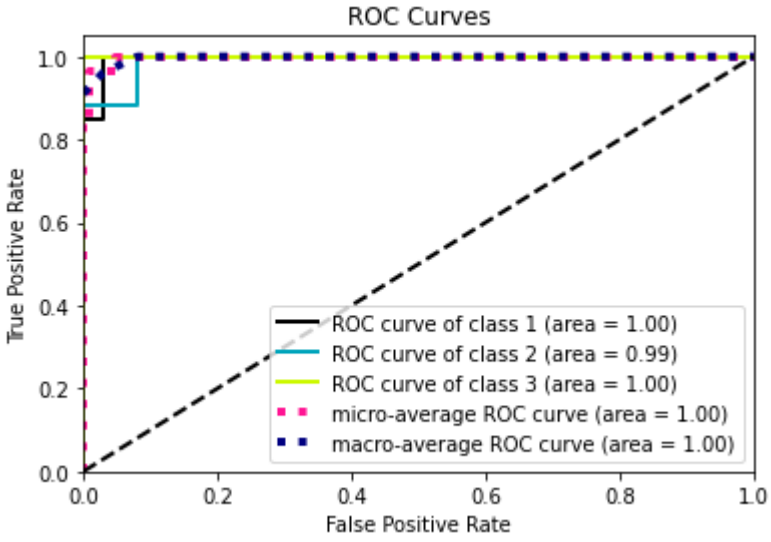
```
SVC Gaussian:
Confusion Matrix
[[20  0  0]
 [ 0 17  0]
 [ 0  1 16]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        20
           2       0.94      1.00      0.97        17
           3       1.00      0.94      0.97        17

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

-----------------------------------
-----------------------------------
Accuracy Score:
0.9814814814814815
```



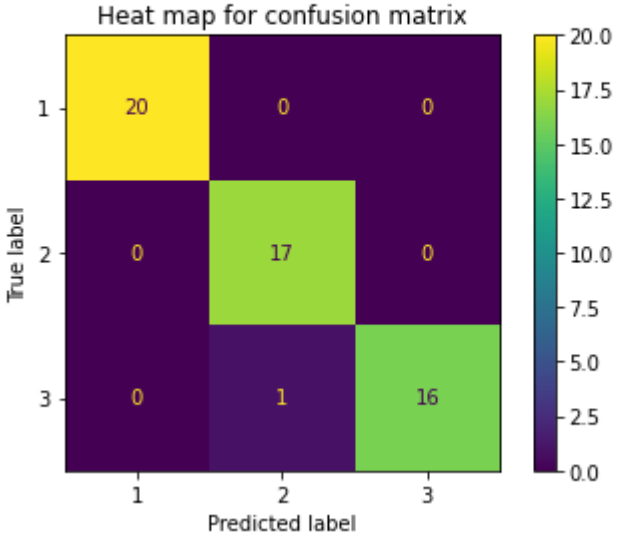


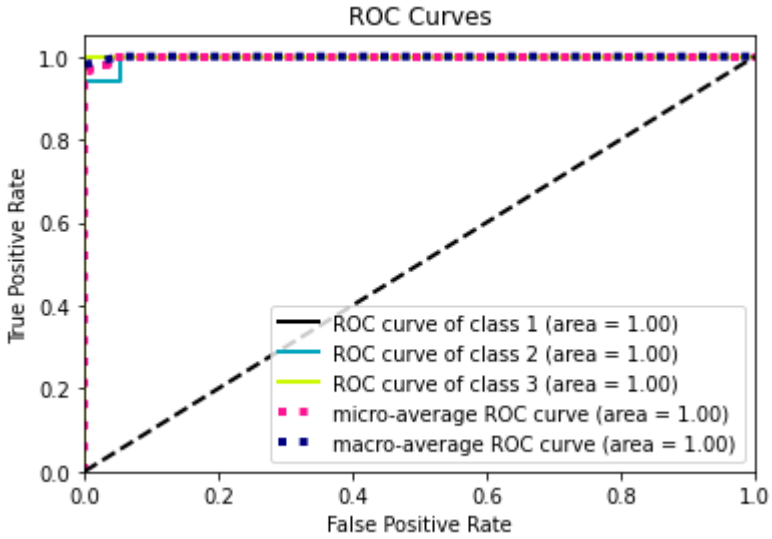
```
SVC Sigmoid:
Confusion Matrix
[[20  0  0]
 [ 2 14  1]
 [ 0  0 17]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       0.91      1.00      0.95        20
           2       1.00      0.82      0.90        17
           3       0.94      1.00      0.97        17

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

-----------------------------------
-----------------------------------
Accuracy Score:
0.9444444444444444
```



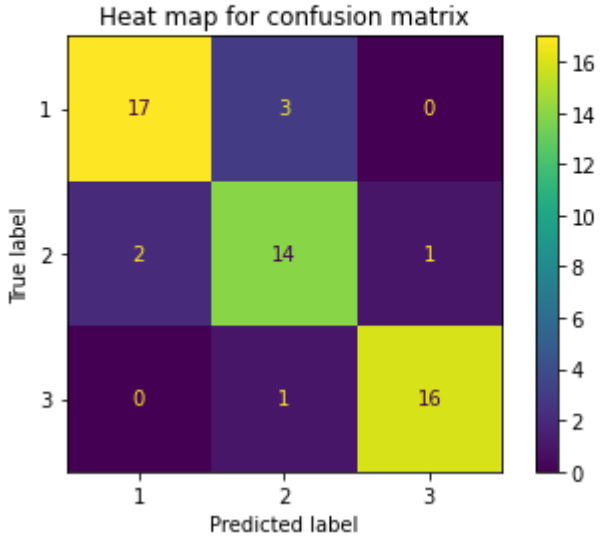


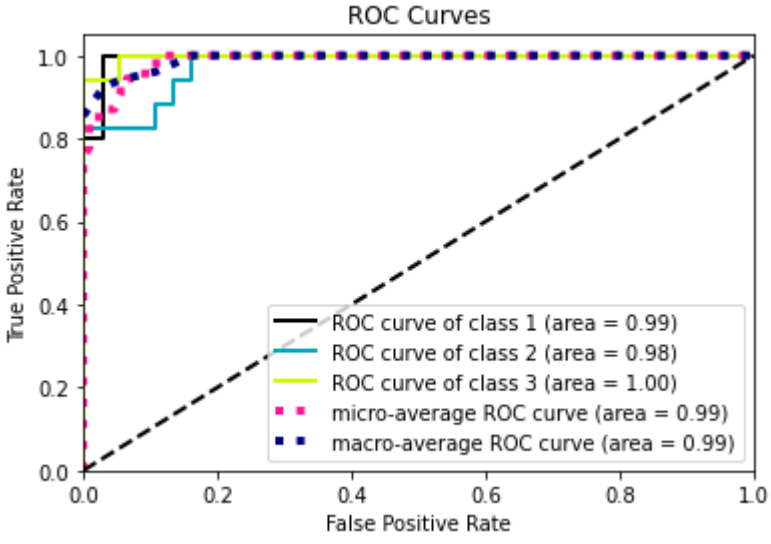
```
MLP:
Confusion Matrix
[[20  0  0]
 [ 0 17  0]
 [ 0  1 16]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        20
           2       0.94      1.00      0.97        17
           3       1.00      0.94      0.97        17

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

-----------------------------------
-----------------------------------
Accuracy Score:
0.9814814814814815
```





```
Random Forest Classifier:
Confusion Matrix
[[17  3  0]
 [ 2 14  1]
 [ 0  1 16]]
-----------------------------------
-----------------------------------
Preformance Evaluation:
              precision    recall  f1-score   support

           1       0.89      0.85      0.87        20
           2       0.78      0.82      0.80        17
           3       0.94      0.94      0.94        17

    accuracy                           0.87        54
   macro avg       0.87      0.87      0.87        54
weighted avg       0.87      0.87      0.87        54

-----------------------------------
-----------------------------------
Accuracy Score:
0.8703703703703703
```





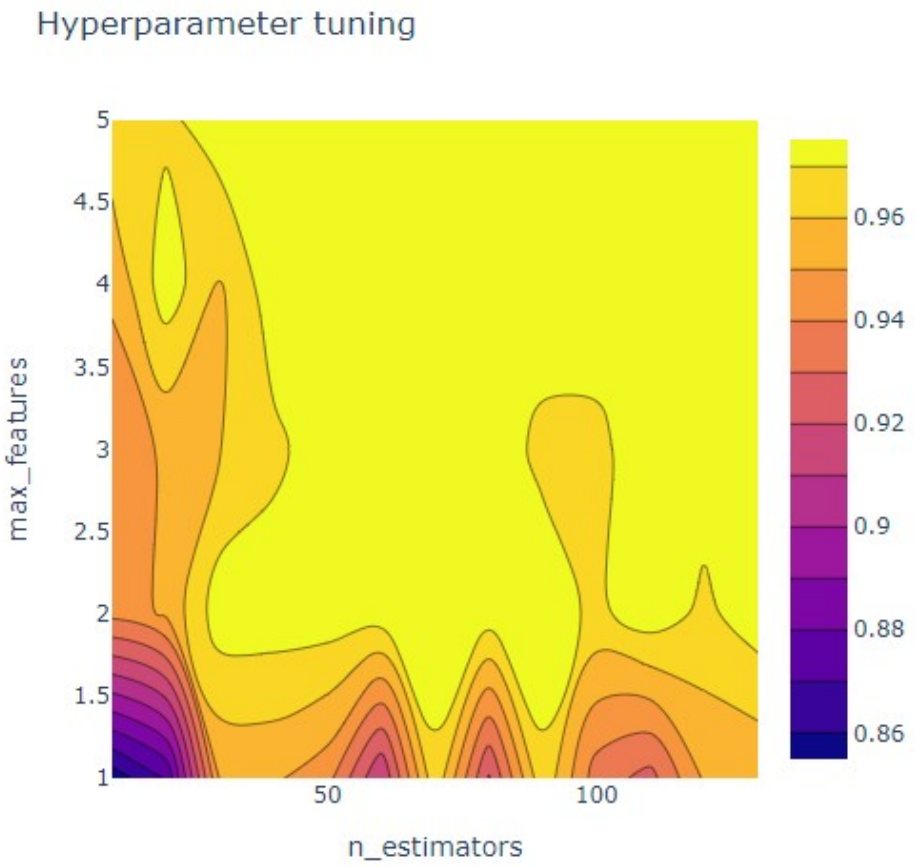
With Parameter Tuning:

```
max_features_range = np.arange(1,6,1)
n_estimators_range = np.arange(10,140,10)
param_grid = dict(max_features=max_features_range, n_estimators=n_estimators_range)

rf = RandomForestClassifier()

grid = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)


print("The best parameters are %s with a score of %0.2f" % (grid.best_params_, grid
.best_score_))

grid_results = pd.concat([pd.DataFrame(grid.cv_results_["params"]),pd.DataFrame(gri
d.cv_results_["mean_test_score"], columns=["Accuracy"])],axis=1)
grid_results.head()

grid_contour = grid_results.groupby(['max_features','n_estimators']).mean()
grid_contour

grid_reset = grid_contour.reset_index()
grid_reset.columns = ['max_features', 'n_estimators', 'Accuracy']
grid_pivot = grid_reset.pivot('max_features', 'n_estimators')
grid_pivot

x = grid_pivot.columns.levels[1].values
y = grid_pivot.index.values
z = grid_pivot.values

layout = go.Layout(
       xaxis=go.layout.XAxis(
        title=go.layout.xaxis.Title(
        text='n_estimators')
       ),
       yaxis=go.layout.YAxis(
        title=go.layout.yaxis.Title(
        text='max_features')
       ) )

fig = go.Figure(data = [go.Contour(z=z, x=x, y=y)], layout=layout )
fig.update_layout(title='Hyperparameter tuning', autosize=False,
          width=500, height=500,
          margin=dict(l=65, r=50, b=65, t=90))

fig.show()

fig = go.Figure(data= [go.Surface(z=z, y=y, x=x)], layout=layout )
```

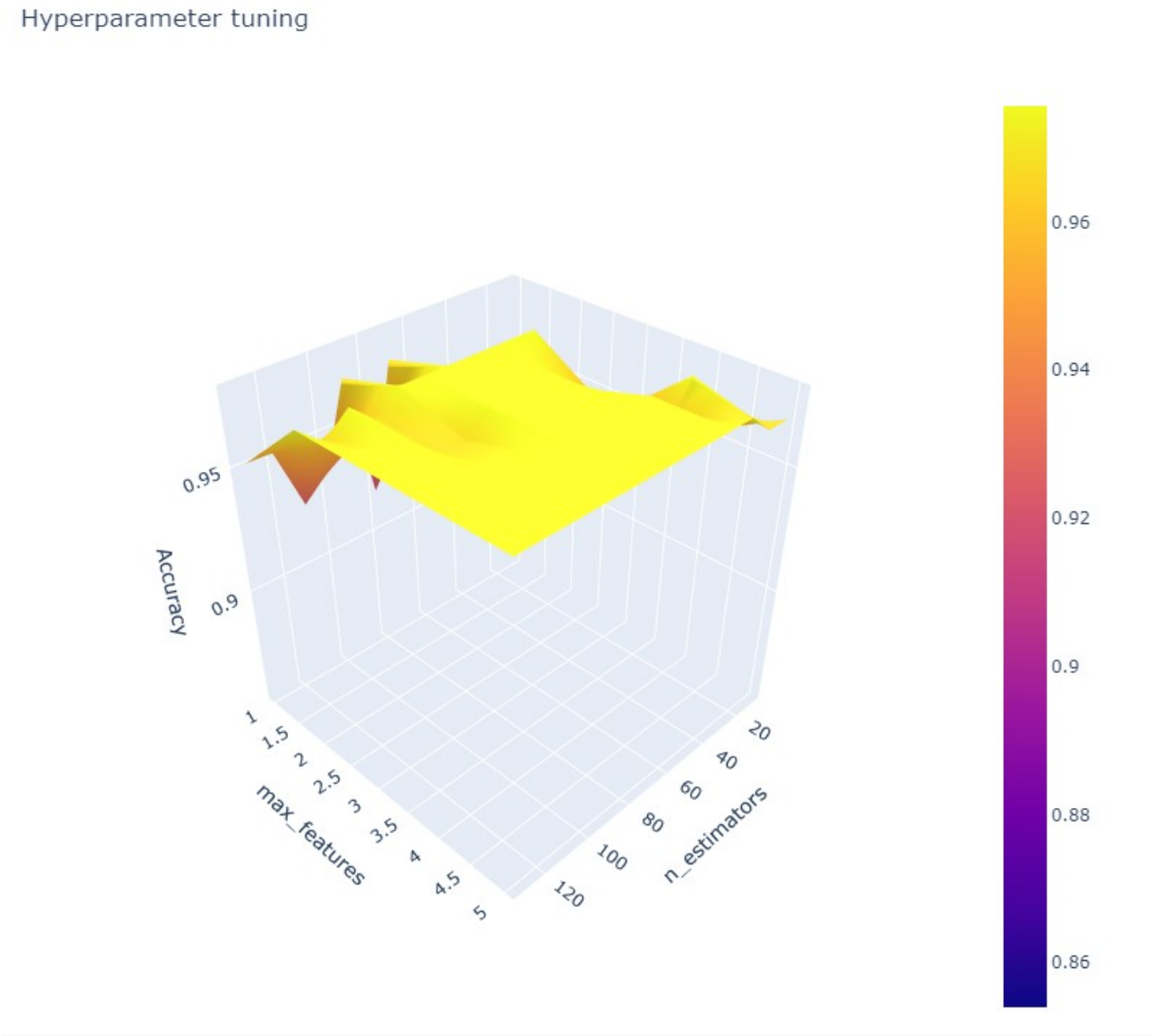
```
fig.update_layout(title='Hyperparameter tuning',
          scene = dict(
           xaxis_title='n_estimators',
           yaxis_title='max_features',
           zaxis_title='Accuracy'),
          autosize=False,
          width=800, height=800,
          margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

2D plot.

```
The best parameters are {'max_features': 2, 'n_estimators': 30} with a score of 0.98
```



3D plot



The model works best for 70:30 split as this provides the best accuracy as it is trained on more data than any other split.

For 60:40 split:

SVC

SVC Linear:
Accuracy score: 0.9305555555555556

SVC Polynomial:
2nd degree accuracy score: 0.8888888888888888
3rd degree accuracy score: 0.9305555555555556

SVC Gaussian:
Accuracy score: 0.9583333333333334

SVC Sigmoid:
Accuracy score: 0.9583333333333334

MLP
Accuracy score: 0.9583333333333334

Random Forest:
Accuracy score: 0.9305555555555556

For 50:50 split:

SVC

SVC Linear:
Accuracy score: 0.9775280898876404

SVC Polynomial:
2nd degree accuracy score: 0.8539325842696629
3rd degree accuracy score: 0.8764044943820225

SVC Gaussian:
Accuracy score: 0.9887640449438202

SVC Sigmoid:
Accuracy score: 0.9550561797752809

MLP
Accuracy score: 0.9775280898876404

Random Forest:
Accuracy score: 0.9887640449438202

For 40:60 split:

SVC

SVC Linear:
Accuracy score: 0.9719626168224299

SVC Polynomial:
2nd degree accuracy score: 0.8878504672897196
3rd degree accuracy score: 0.9158878504672897

SVC Gaussian:
Accuracy score: 0.9626168224299065

SVC Sigmoid:
Accuracy score: 0.9439252336448598

MLP
Accuracy score: 0.9719626168224299

Random Forest:
Accuracy score: 0.9626168224299065

For 30:70 split:

SVC

SVC Linear:
Accuracy score: 0.952

SVC Polynomial:
2nd degree accuracy score: 0.824
3rd degree accuracy score: 0.888

SVC Gaussian:
Accuracy score: 0.968

SVC Sigmoid:
Accuracy score: 0.944

MLP
Accuracy score: 0.952

Random Forest:
Accuracy score: 0.912

COMPARISON TABLE

Splits	70:30	60:40	50:50	40:60	30:70
Classifiers					
RFC	0.870	0.930	0.988	0.962	0.912
SVC					
Linear	0.962	0.930	0.977	0.971	0.952
Polynomial					
2nd	0.833	0.888	0.853	0.887	0.824
3rd	0.814	0.930	0.876	0.915	0.888
Gaussian	0.981	0.958	0.988	0.962	0.968
Sigmoid	0.944	0.958	0.955	0.943	0.944
MLP	0.984	0.958	0.977	0.971	0.952

NAME - SHUVRASISH ROY

ROLL NO. - 001811001012

MACHINE LEARNING LAB ASSIGNMENT 2

Import required modules

```
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

```
b_cancer = datasets.load_breast_cancer() # it's source is same as : https://archive.ics.uc
```

```
dir(b_cancer)
```

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

```
b_cancer.data
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

```
print(b_cancer.feature_names)
print(b_cancer.target_names)
print(b_cancer.target)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
['malignant' 'benign']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 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 0 0 0 0 0 0 1]
```

```
df = pd.DataFrame(data=b_cancer.data, columns=b_cancer.feature_names)
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

```
df["target"] = b_cancer.target
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	

```
df.target.value_counts()
```

```
1    357
0    212
Name: target, dtype: int64
```

4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

DataFrame ready to perform

```
len(df)
```

```
569
```

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

```
   mean radius  mean texture  ...  worst symmetry  worst fractal dimension
0        17.99         10.38  ...          0.4601                  0.11890
1        20.57         17.77  ...          0.2750                  0.08902
2        19.69         21.25  ...          0.3613                  0.08758
3        11.42         20.38  ...          0.6638                  0.17300
4        20.29         14.34  ...          0.2364                  0.07678

[5 rows x 30 columns]
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int64
```

SVC Classfier

Linear SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
```

```
        max_iter=-1, probability=False, random_state=None, shrinking=True,
        tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

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

```
Accuracy: 95.90643274853801%

Confusion Matrix:

[[ 61   2]
 [  5 103]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.97      0.95        63
           1       0.98      0.95      0.97       108

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed652ca90>
```

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))
```

```
341
228
```

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

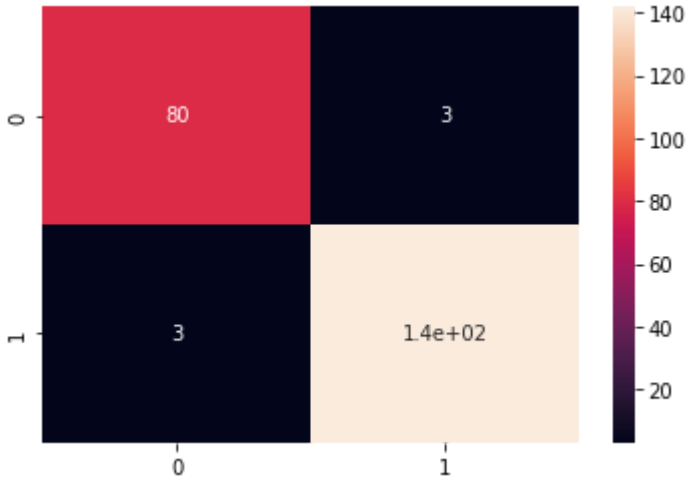
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 97.36842105263158%

Confusion Matrix:
[[ 80   3]
 [  3 142]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.96      0.96        83
           1       0.98      0.98      0.98       145

    accuracy                           0.97       228
   macro avg       0.97      0.97      0.97       228
weighted avg       0.97      0.97      0.97       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed64fe090>
```



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))
```

```
284
285
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

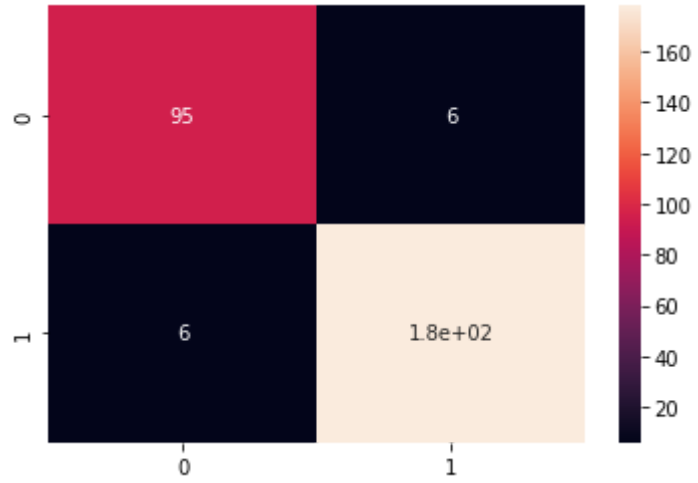
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.78947368421052%

Confusion Matrix:
[[ 95   6]
 [  6 178]]

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.94      0.94       101
```

```
                1       0.97      0.97      0.97       184

         accuracy                           0.96       285
        macro avg       0.95      0.95      0.95       285
     weighted avg       0.96      0.96      0.96       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed642f3d0>
```



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))
```

```
227
342
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

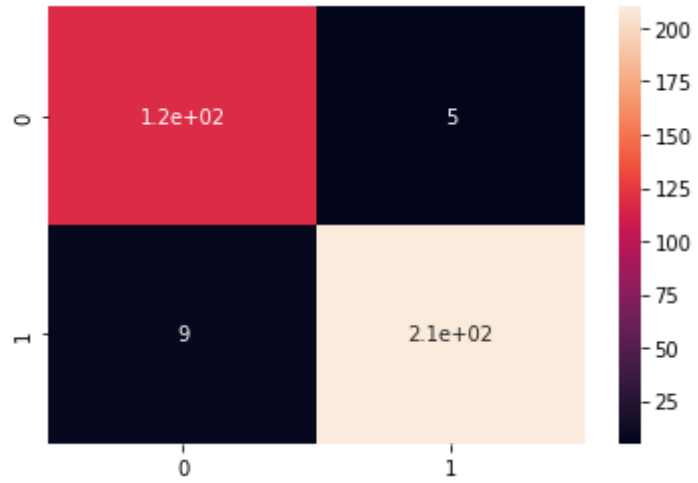
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.90643274853801%
```

```
Confusion Matrix:

[[118   5]
 [  9 210]]

Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.96      0.94       123
           1       0.98      0.96      0.97       219

    accuracy                           0.96       342
   macro avg       0.95      0.96      0.96       342
weighted avg       0.96      0.96      0.96       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed63ca210>
```



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))
```

```
170
399
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

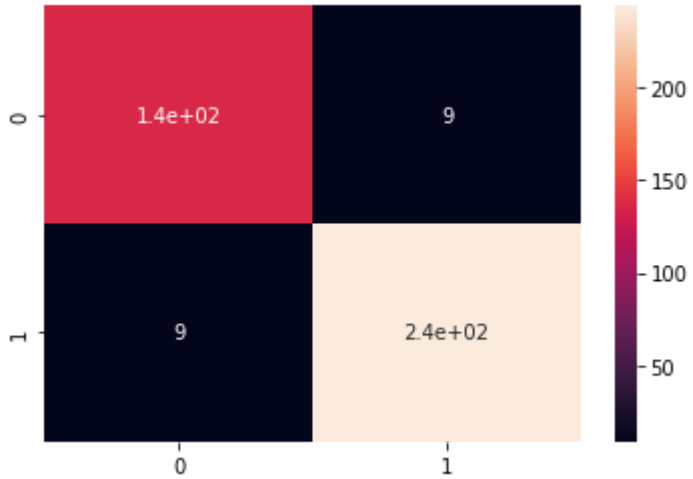
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test, y_pred)}%\n")
```

```
print("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: 95.48872180451127%

Confusion Matrix:

[[137   9]
 [  9 244]]

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.94      0.94       146
           1       0.96      0.96      0.96       253

    accuracy                           0.95       399
   macro avg       0.95      0.95      0.95       399
weighted avg       0.95      0.95      0.95       399
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed62fe750>
```



Polynomial SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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))
```

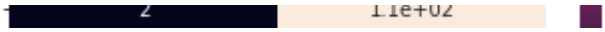
```
Accuracy: 91.81286549707602%

Confusion Matrix:
 [[ 51  12]
 [  2 106]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.81      0.88        63
           1       0.90      0.98      0.94       108

    accuracy                           0.92       171
   macro avg       0.93      0.90      0.91       171
weighted avg       0.92      0.92      0.92       171
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6225310>
```

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))
```

```
341
228
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 91.66666666666666%

Confusion Matrix:
 [[ 65  18]
 [  1 144]]

Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.78      0.87        83
           1       0.89      0.99      0.94       145

    accuracy                           0.92       228
   macro avg       0.94      0.89      0.91       228
weighted avg       0.92      0.92      0.91       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed621b950>
```

train size : test size = 50% : 50%

```
                0                   1
```

```
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))
```

```
284
285
```

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

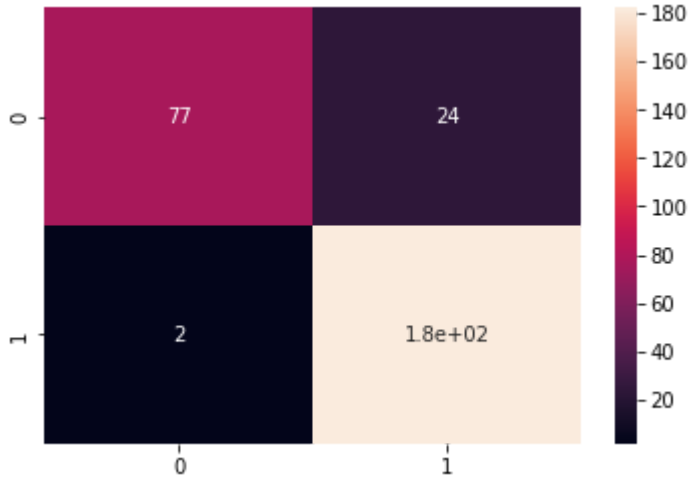
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 90.87719298245615%

Confusion Matrix:
 [[ 77  24]
 [  2 182]]

Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.76      0.86       101
           1       0.88      0.99      0.93       184

    accuracy                           0.91       285
   macro avg       0.93      0.88      0.89       285
weighted avg       0.92      0.91      0.91       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6159050>
```



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))
```

```
227
342
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

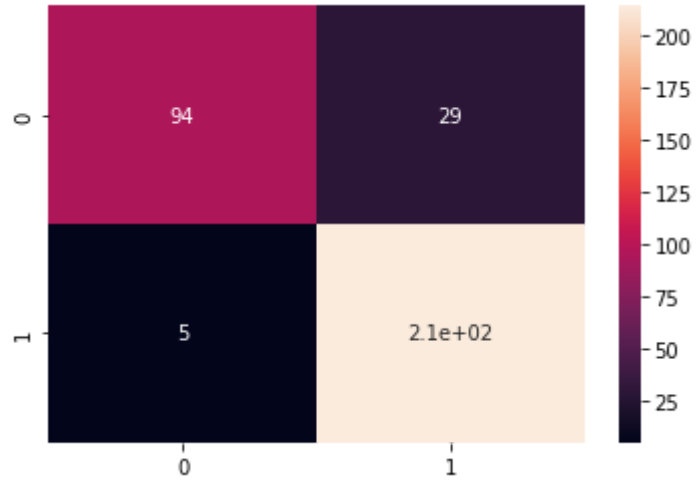
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.05847953216374%

Confusion Matrix:
 [[ 94  29]
 [  5 214]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.76      0.85       123
```

```
                1       0.88      0.98      0.93       219

         accuracy                           0.90       342
        macro avg       0.92      0.87      0.89       342
     weighted avg       0.91      0.90      0.90       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed608b150>
```



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))
```

```
170
399
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

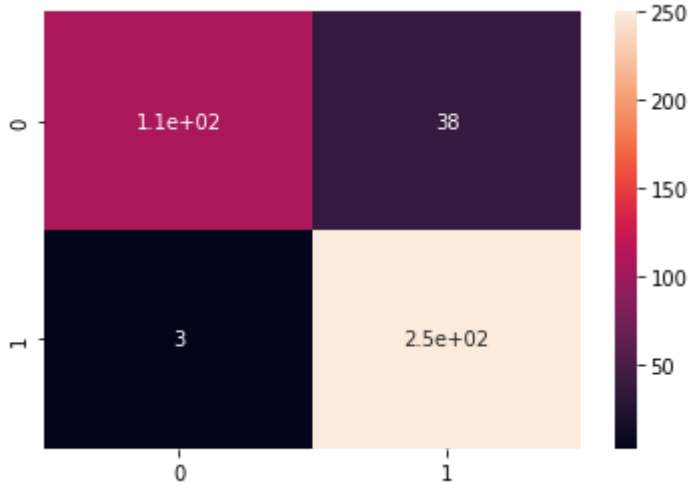
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 89.72431077694235%

Confusion Matrix:
 [[108  38]
```

```
 [  3 250]]

Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.74      0.84       146
           1       0.87      0.99      0.92       253

    accuracy                           0.90       399
   macro avg       0.92      0.86      0.88       399
weighted avg       0.91      0.90      0.89       399
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed5fb4ad0>
```



Gaussain SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

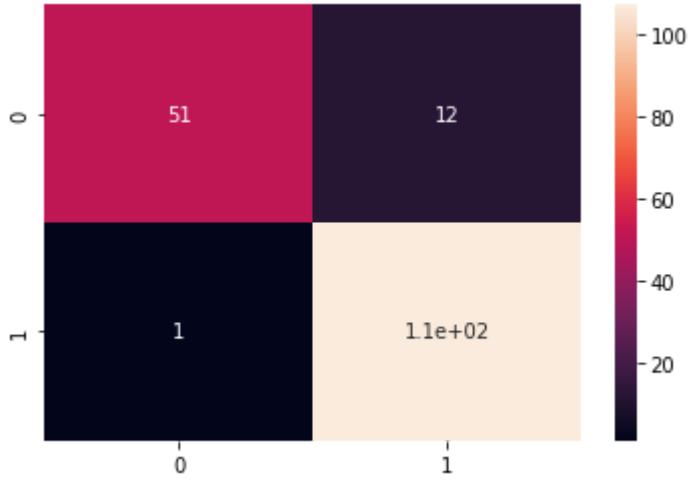
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 92.39766081871345%

Confusion Matrix:
 [[ 51  12]
 [  1 107]]

Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.81      0.89        63
           1       0.90      0.99      0.94       108

    accuracy                           0.92       171
   macro avg       0.94      0.90      0.91       171
weighted avg       0.93      0.92      0.92       171
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed5f5d4d0>
```



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))
```

```
341
228
```

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

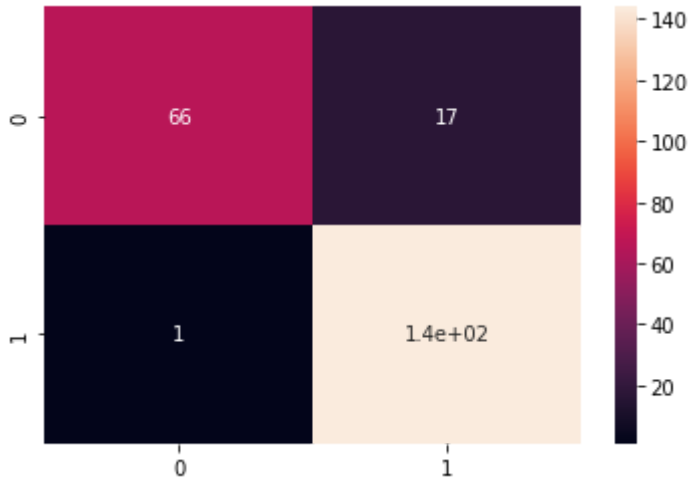
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 92.10526315789474%

Confusion Matrix:
 [[ 66  17]
 [  1 144]]

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.80      0.88        83
           1       0.89      0.99      0.94       145

    accuracy                           0.92       228
   macro avg       0.94      0.89      0.91       228
weighted avg       0.93      0.92      0.92       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6fcf4d0>
```



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))
```

```
284
285
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

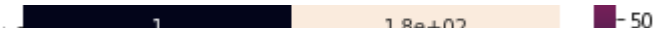
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 91.57894736842105%

Confusion Matrix:
 [[ 78  23]
 [  1 183]]

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.77      0.87       101
           1       0.89      0.99      0.94       184

    accuracy                           0.92       285
   macro avg       0.94      0.88      0.90       285
weighted avg       0.92      0.92      0.91       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed5df8210>
```

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))
```

```
227
342
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 90.64327485380117%

Confusion Matrix:
 [[ 94  29]
 [  3 216]]

Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.76      0.85       123
           1       0.88      0.99      0.93       219

    accuracy                           0.91       342
   macro avg       0.93      0.88      0.89       342
weighted avg       0.91      0.91      0.90       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed5d6d350>
```

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))
```

```
170
399
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

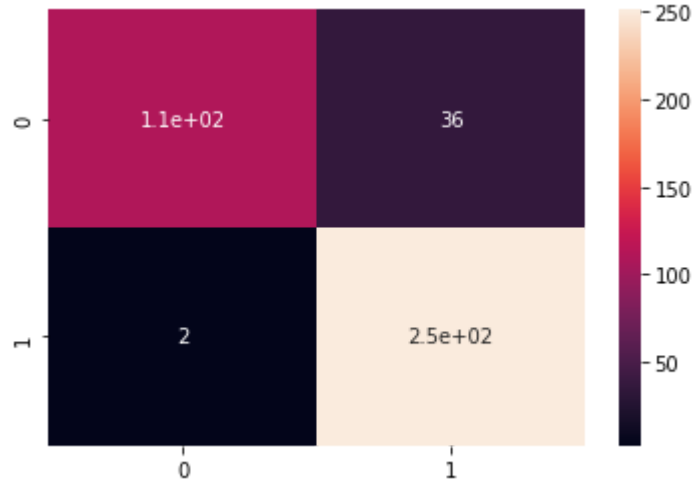
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.47619047619048%

Confusion Matrix:
 [[110  36]
 [  2 251]]

Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.75      0.85       146
           1       0.87      0.99      0.93       253

    accuracy                           0.90       399
   macro avg       0.93      0.87      0.89       399
```

```
weighted avg       0.91      0.90      0.90       399
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed5cb6990>
```



Sigmoid SVC Classifier

```
sigmoid_SVC_classifier = SVC(kernel='sigmoid', C=0.9)
sigmoid_SVC_classifier
```

```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 46.198830409356724%

Confusion Matrix:
 [[10 53]
 [39 69]]

Classification Report:

              precision    recall  f1-score   support

           0       0.20      0.16      0.18        63
           1       0.57      0.64      0.60       108

    accuracy                           0.46       171
   macro avg       0.38      0.40      0.39       171
weighted avg       0.43      0.46      0.44       171
```

```
from sklearn.model_selection import GridSearchCV
```

```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] C=0.1, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=0.1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=0.1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=0.1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=0.1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=0.1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=0.1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=0.1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=0.1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=0.1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=0.1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=sigmoid ..............................
```

```
[CV] .............. C=0.1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=sigmoid .............................
[CV] .............. C=0.1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=sigmoid .............................
[CV] .............. C=0.1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=sigmoid .............................
[CV] .............. C=0.1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=sigmoid .............................
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[CV] .............. C=0.1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid ....................................
[CV] .................... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid ....................................
[CV] .................... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid ....................................
[CV] .................... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid ....................................
[CV] .................... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid ....................................
[CV] .................... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid ..................................
[CV] .................. C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid ..................................
[CV] .................. C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid ..................................
[CV] .................. C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
```

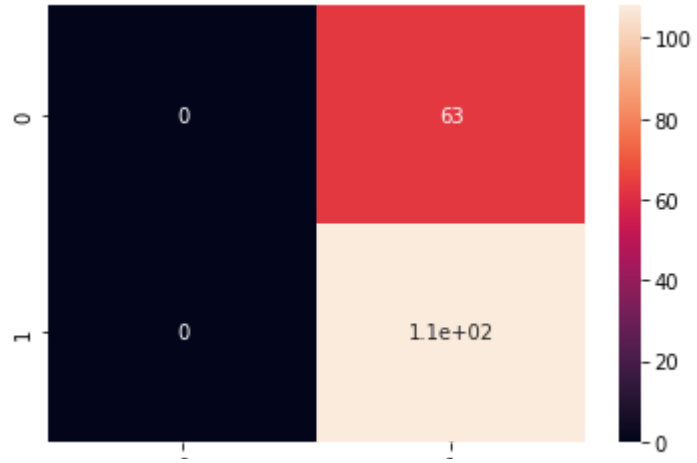
```
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['sigmoid']}
```

```
print(grid.best_estimator_)
```

```
SVC(C=0.1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

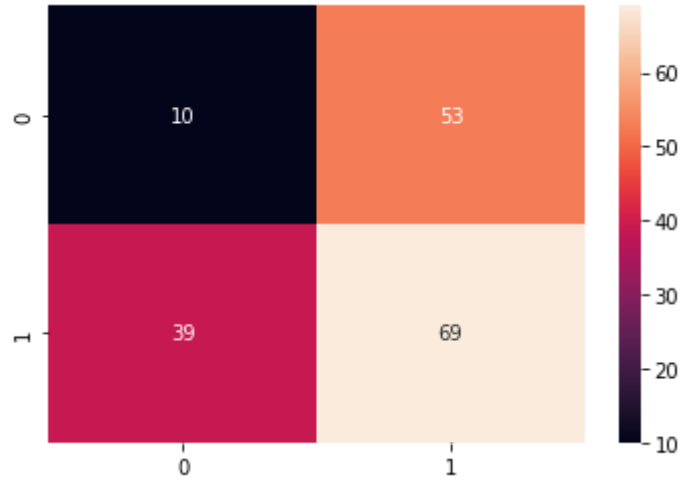
```
import matplotlib.pyplot as plt
grid_predictions = grid.predict(X_test)
print(confusion_matrix(y_test,grid_predictions))
plt.show(sns.heatmap(confusion_matrix(y_test,grid_predictions), annot=True))
print(classification_report(y_test,grid_predictions))
print("Accuracy Score of RBF kernel", accuracy_score(y_test,grid_predictions))
```

```
[[  0  63]
 [  0 108]]
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6f6be90>
```



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))
```

```
341
228
```

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

```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

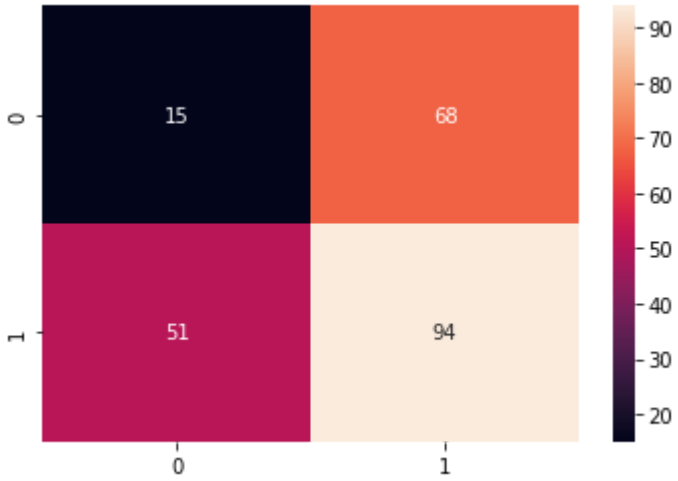
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

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

```
Accuracy: 47.80701754385965%

Confusion Matrix:
 [[15 68]
 [51 94]]

Classification Report:

              precision    recall  f1-score   support

           0       0.23      0.18      0.20        83
           1       0.58      0.65      0.61       145

    accuracy                           0.48       228
   macro avg       0.40      0.41      0.41       228
weighted avg       0.45      0.48      0.46       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6f10c50>
```



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))
```

```
284
285
```

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

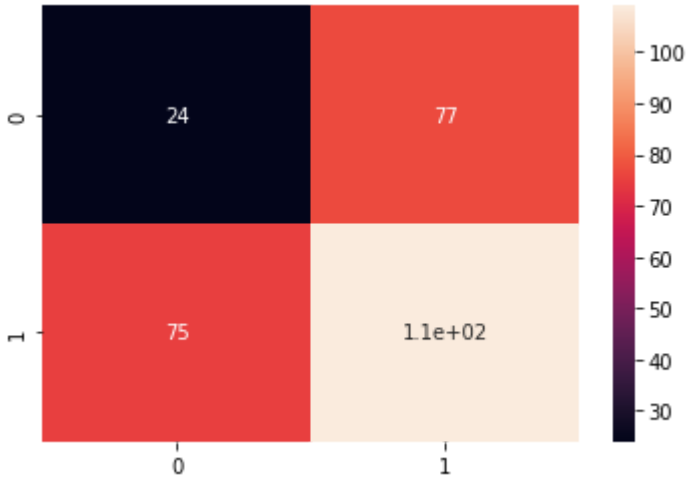
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
```

```
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)

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: 46.666666666666664%

Confusion Matrix:
 [[ 24  77]
 [ 75 109]]

Classification Report:

              precision    recall  f1-score   support

           0       0.24      0.24      0.24       101
           1       0.59      0.59      0.59       184

    accuracy                           0.47       285
   macro avg       0.41      0.42      0.41       285
weighted avg       0.46      0.47      0.47       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6e50050>
```



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))
```

```
227
342
```

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

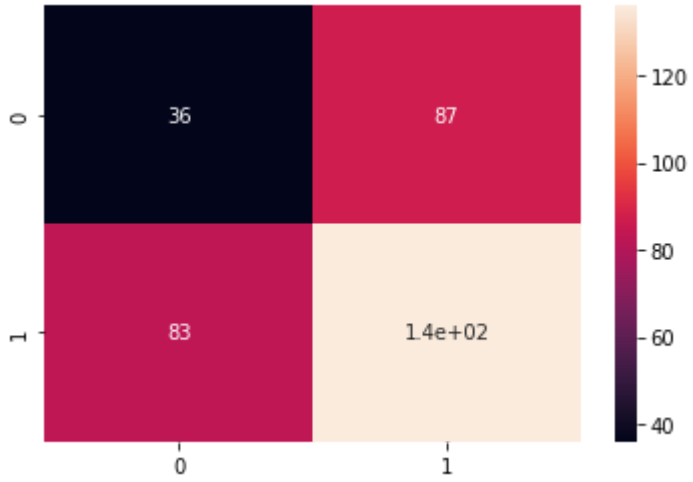
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 50.29239766081871%

Confusion Matrix:
 [[ 36  87]
 [ 83 136]]

Classification Report:

              precision    recall  f1-score   support

           0       0.30      0.29      0.30       123
           1       0.61      0.62      0.62       219

    accuracy                           0.50       342
   macro avg       0.46      0.46      0.46       342
weighted avg       0.50      0.50      0.50       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6d7d590>
```



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))
```

```
170
399
```

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

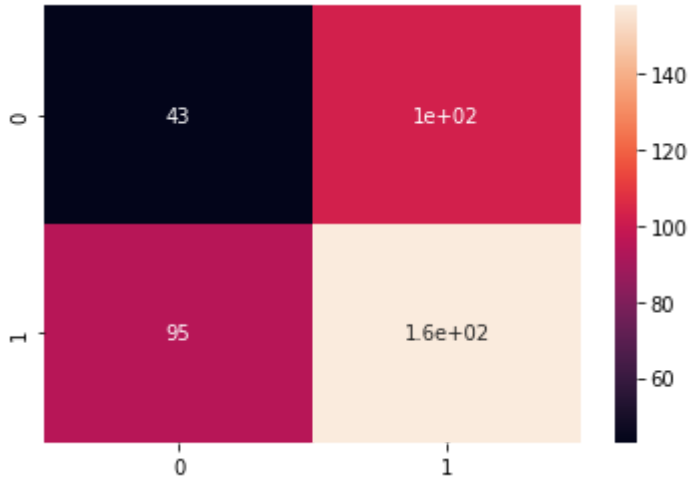
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 50.37593984962406%

Confusion Matrix:
 [[ 43 103]
 [ 95 158]]

Classification Report:

              precision    recall  f1-score   support

           0       0.31      0.29      0.30       146
           1       0.61      0.62      0.61       253

    accuracy                           0.50       399
   macro avg       0.46      0.46      0.46       399
weighted avg       0.50      0.50      0.50       399
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6cb9650>
```



MLP Classifier

```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
mlp_classifier
```

```
MLPClassifier(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=600,
              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)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

```
MLPClassifier(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=600,
              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)
```

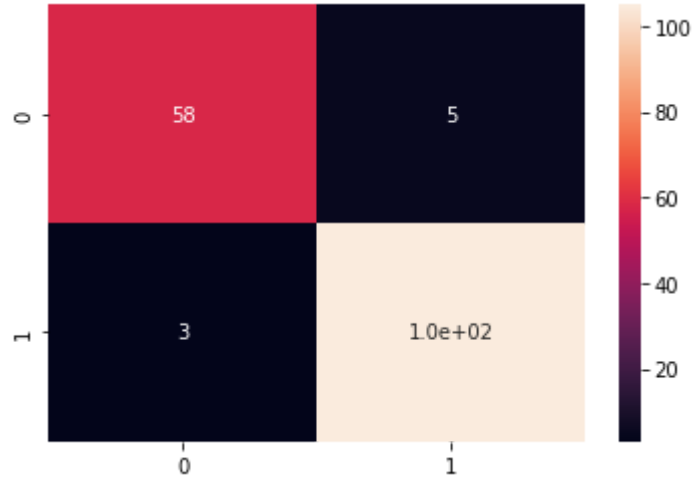
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.32163742690058%

Confusion Matrix:

[[ 58   5]
 [  3 105]]
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.92      0.94        63
           1       0.95      0.97      0.96       108

    accuracy                           0.95       171
   macro avg       0.95      0.95      0.95       171
weighted avg       0.95      0.95      0.95       171
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6d08790>
```



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))
```

```
341
228
```

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

```
MLPClassifier(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=600,
              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)
```

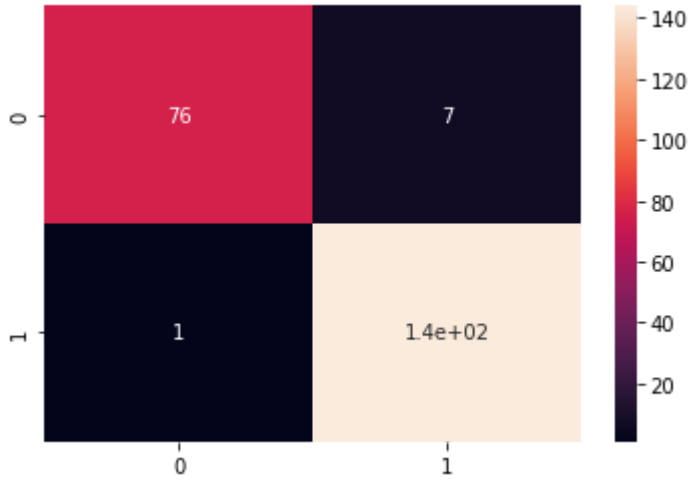
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test, y_pred)
```

```
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: 96.49122807017544%

Confusion Matrix:
[[ 76   7]
 [  1 144]]

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.92      0.95        83
           1       0.95      0.99      0.97       145

    accuracy                           0.96       228
   macro avg       0.97      0.95      0.96       228
weighted avg       0.97      0.96      0.96       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6b6fe90>
```



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))
```

```
284
285
```

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

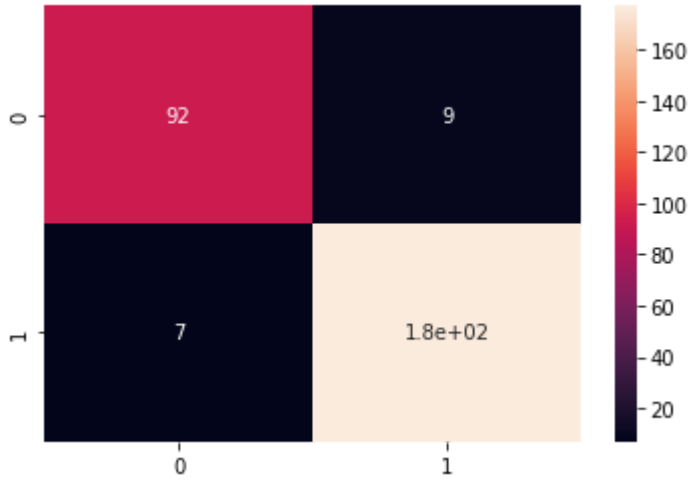
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9, 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)
```

```
y_pred = mlp_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: 94.38596491228071%

Confusion Matrix:
[[ 92   9]
 [  7 177]]

Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.91      0.92       101
           1       0.95      0.96      0.96       184

    accuracy                           0.94       285
   macro avg       0.94      0.94      0.94       285
weighted avg       0.94      0.94      0.94       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6aaf5d0>
```



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))
```

```
227
342
```

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

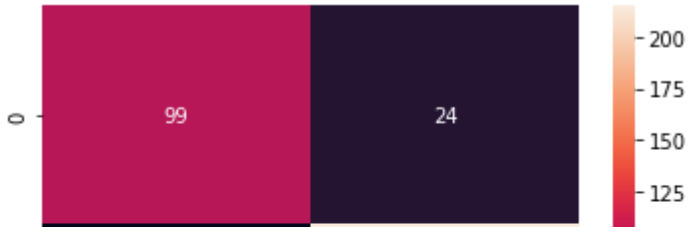
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9, 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)
```

```
y_pred = mlp_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))
```

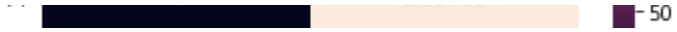
```
Accuracy: 91.81286549707602%

Confusion Matrix:

[[ 99  24]
 [  4 215]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.80      0.88       123
           1       0.90      0.98      0.94       219

    accuracy                           0.92       342
   macro avg       0.93      0.89      0.91       342
weighted avg       0.92      0.92      0.92       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6a37550>
```



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))
```

```
170
399
```

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

```
MLPClassifier(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=600,
              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)
```

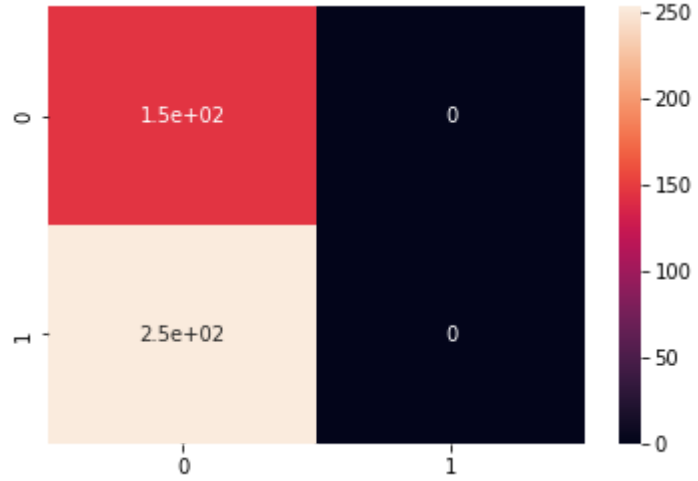
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 36.59147869674185%

Confusion Matrix:

[[146   0]
 [253   0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.37      1.00      0.54       146
           1       0.00      0.00      0.00       253

    accuracy                           0.37       399
   macro avg       0.18      0.50      0.27       399
weighted avg       0.13      0.37      0.20       399
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed693b090>
```



Random Forest Classifier

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

train size : test size = 70% : 30%

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

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

```
398
171
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
```

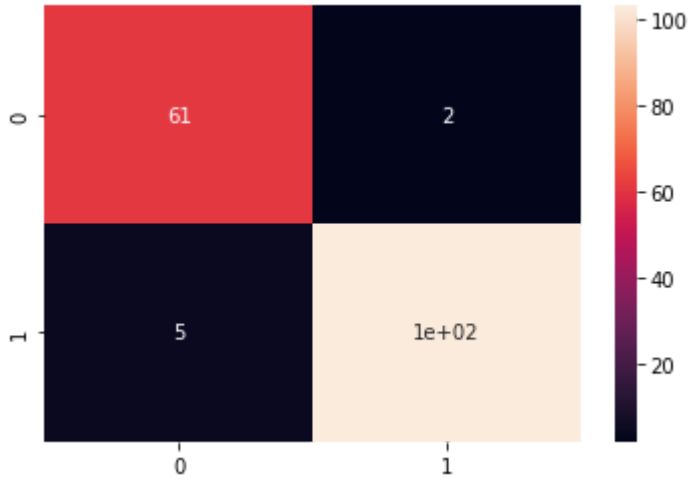
```
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
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: 95.90643274853801%

Confusion Matrix:

[[ 61   2]
 [  5 103]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.97      0.95        63
           1       0.98      0.95      0.97       108

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed68df550>
```



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))
```

```
341
228
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

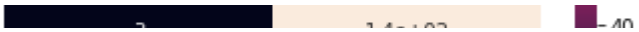
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 96.05263157894737%

Confusion Matrix:
[[ 77   6]
 [  3 142]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.93      0.94        83
           1       0.96      0.98      0.97       145

    accuracy                           0.96       228
   macro avg       0.96      0.95      0.96       228
weighted avg       0.96      0.96      0.96       228
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed73f1cd0>
```

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))
```

```
284
285
```

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

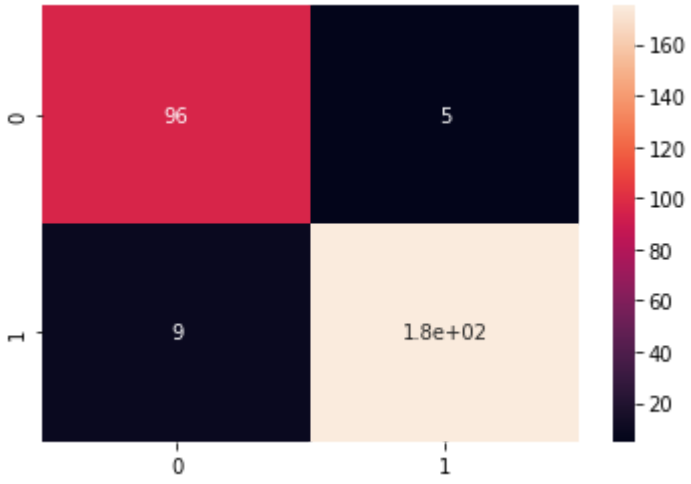
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

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

```
Accuracy: 95.08771929824562%

Confusion Matrix:
[[ 96   5]
 [  9 175]]

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.95      0.93       101
           1       0.97      0.95      0.96       184

    accuracy                           0.95       285
   macro avg       0.94      0.95      0.95       285
weighted avg       0.95      0.95      0.95       285
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6781390>
```



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))
```

```
227
342
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

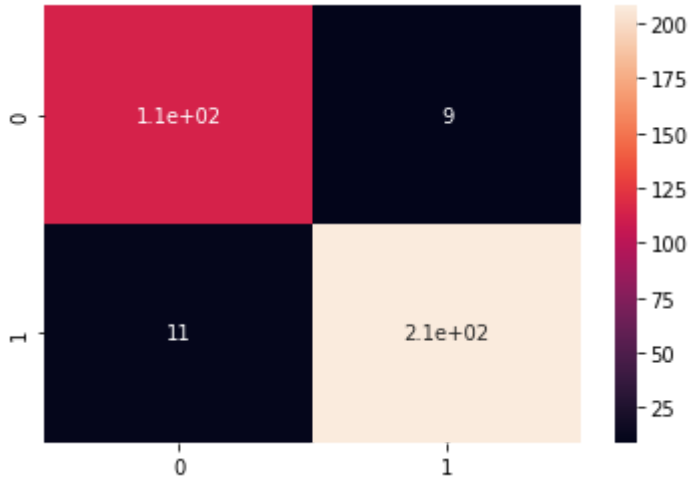
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 94.15204678362574%

Confusion Matrix:

[[114   9]
 [ 11 208]]
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.93      0.92       123
           1       0.96      0.95      0.95       219

    accuracy                           0.94       342
   macro avg       0.94      0.94      0.94       342
weighted avg       0.94      0.94      0.94       342
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed66b9990>
```



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))
```

```
170
399
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

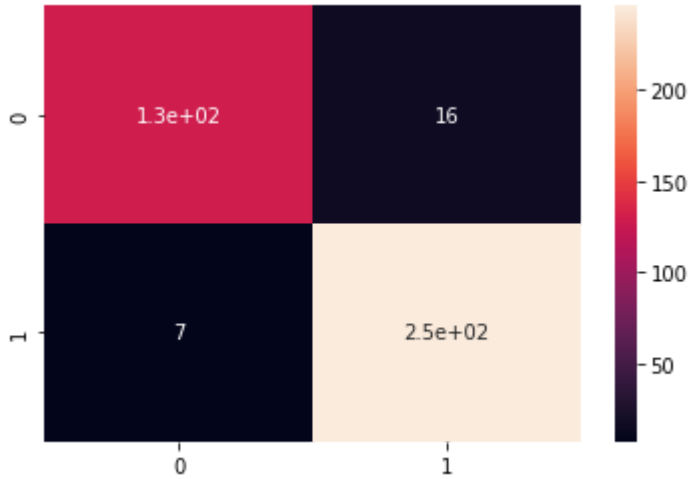
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test, y_pred)}%\n")
```

```
print("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: 94.23558897243107%

Confusion Matrix:

[[130  16]
 [  7 246]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.89      0.92       146
           1       0.94      0.97      0.96       253

    accuracy                           0.94       399
   macro avg       0.94      0.93      0.94       399
weighted avg       0.94      0.94      0.94       399
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed665e0d0>
```



NAME - SHUVRASISH ROY

ROLL NO. - 001811001012

MACHINE LEARNING LAB ASSIGNMENT 2

Import required modules

```
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

```
df = pd.read_csv('/content/ionosphere_data.csv')
df.head()
```

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	cc
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	

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

```
g    225
b    126
Name: column_ai, dtype: int64
```

DataFrame ready to perform

```
len(df)
```

```
351
```

```
X = df.drop(["column_ai"], axis="columns")
y = df.column_ai
print(X.head())
print(y.head())
```

```
   column_a  column_b  column_c  ...  column_af  column_ag  column_ah
0      True     False   0.99539  ...   -0.54487    0.18641   -0.45300
1      True     False   1.00000  ...   -0.06288   -0.13738   -0.02447
2      True     False   1.00000  ...   -0.24180    0.56045   -0.38238
3      True     False   1.00000  ...    1.00000   -0.32382    1.00000
4      True     False   1.00000  ...   -0.59573   -0.04608   -0.65697

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

SVC Classfier

Linear SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

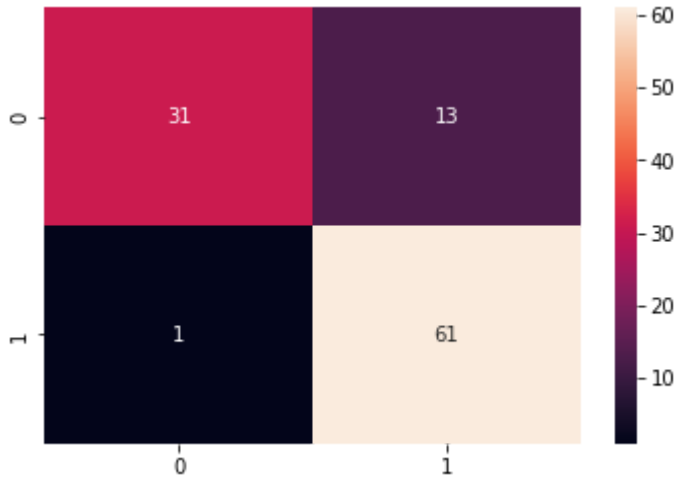
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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:\n")
print(cf_matrix)
print("\nClassification Report:\n")
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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fd4a58650>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fca484750>
```

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
```


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

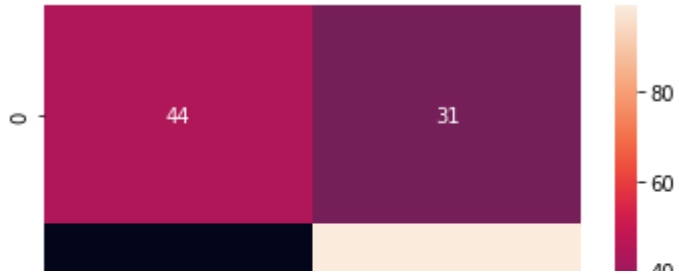
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9f5da90>
```



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
```

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

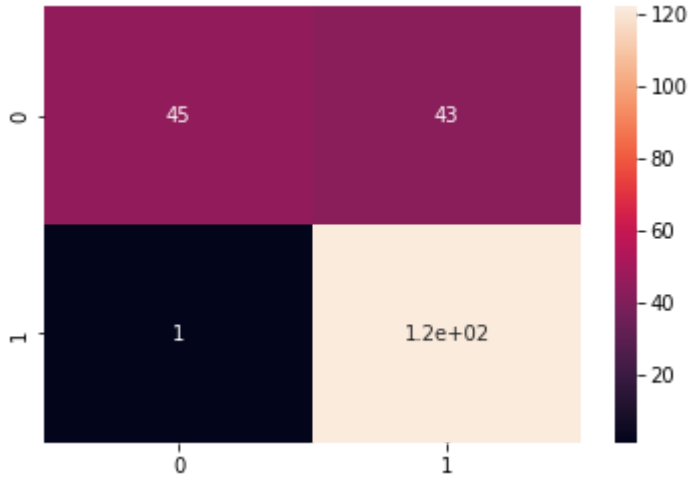
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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:\n")
print(cf_matrix)
print("\nClassification Report:\n")
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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9e95450>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

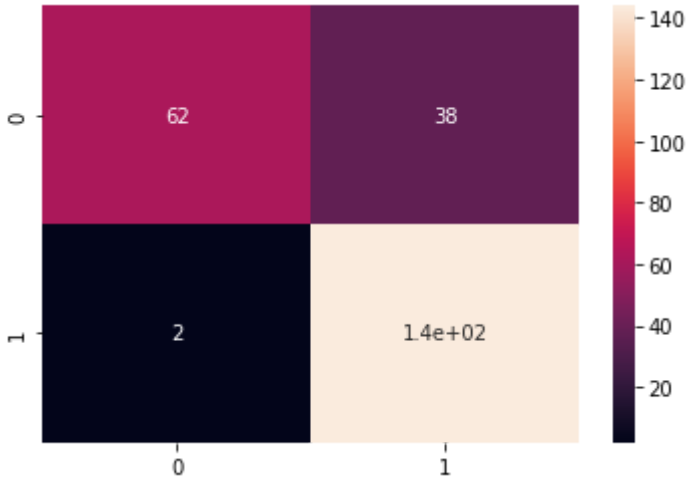
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9f65a50>
```



Polynomial SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

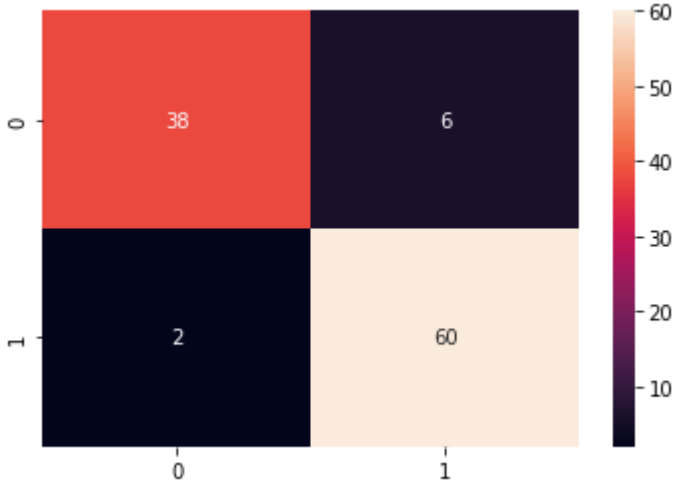
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
```

```
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)

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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9d7b0d0>
```



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
```

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

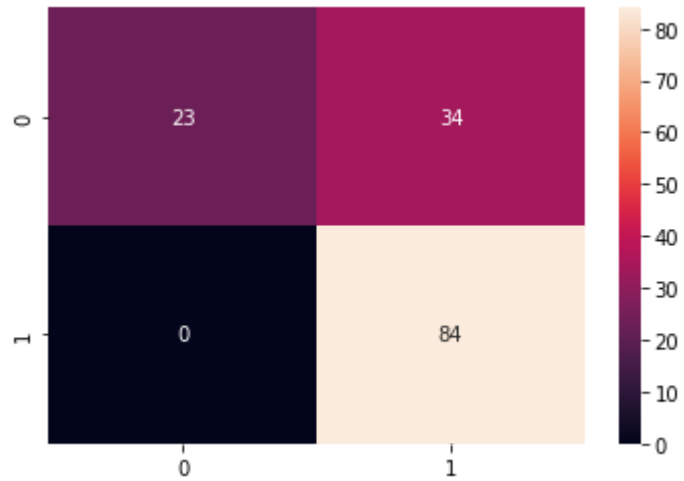
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 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 0x7f0fc9d0a3d0>
```



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(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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 0x7f0fc9c31bd0>
```

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)
```

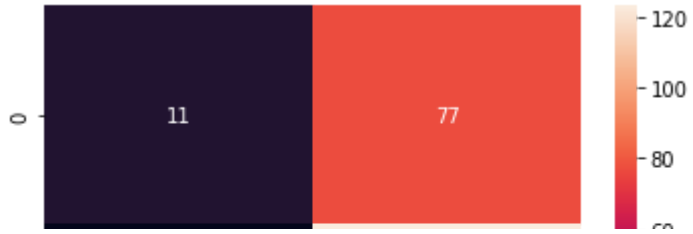
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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))
```

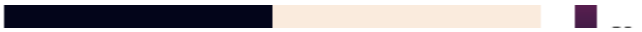
```
Accuracy: 63.507109004739334%

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 0x7f0fc9c8f610>
```



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
```

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

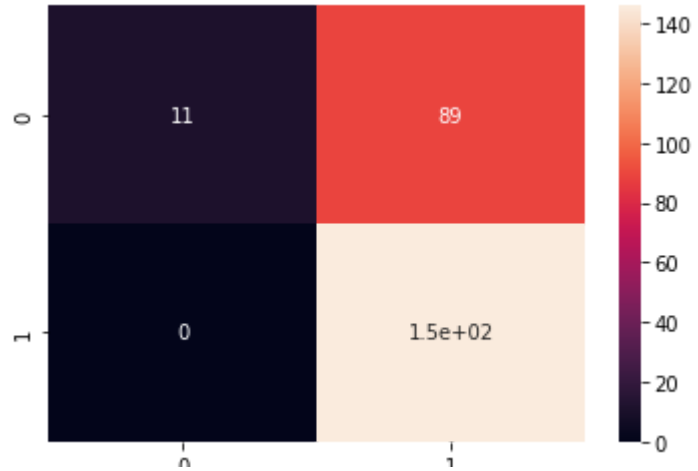
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 63.82113821138211%

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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9b11050>
```



Gaussain SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

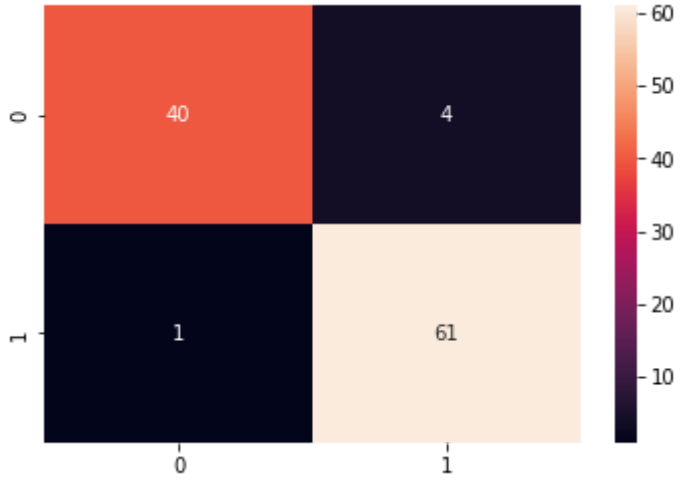
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9a3d310>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

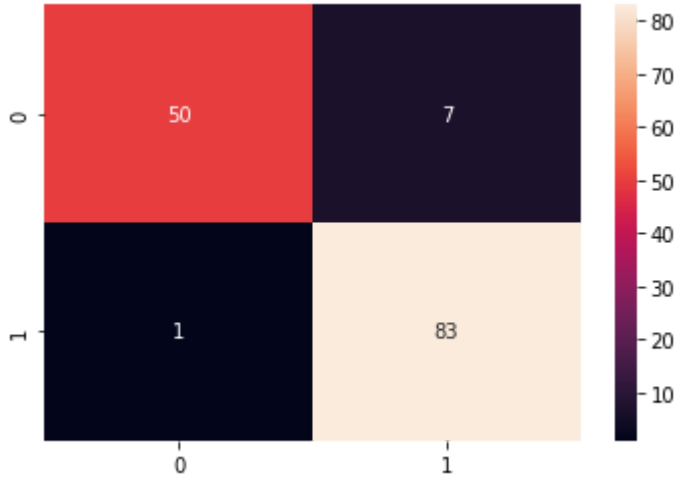
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test, y_pred)
```

```
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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc995f510>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
```

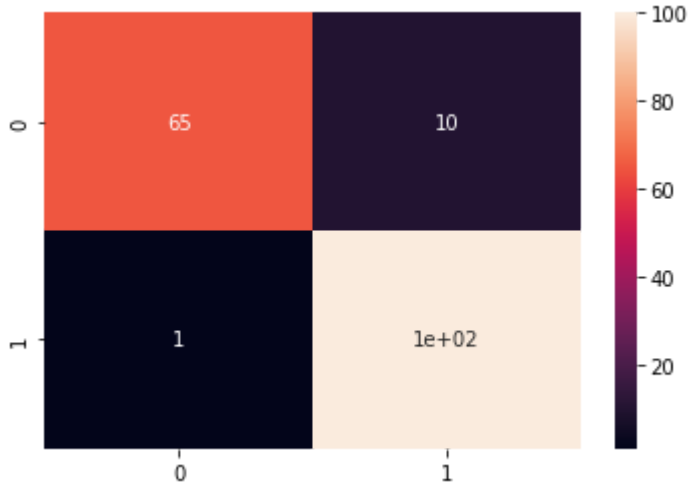
```
          decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9908110>
```



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
```

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

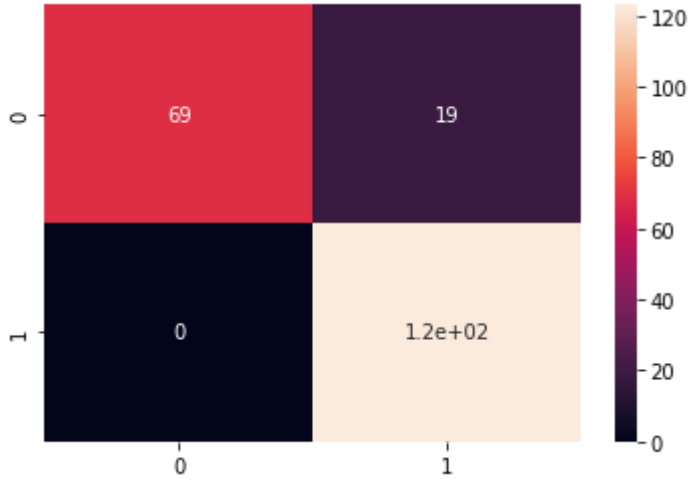
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9cea990>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc97ad810>
```

Sigmoid SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

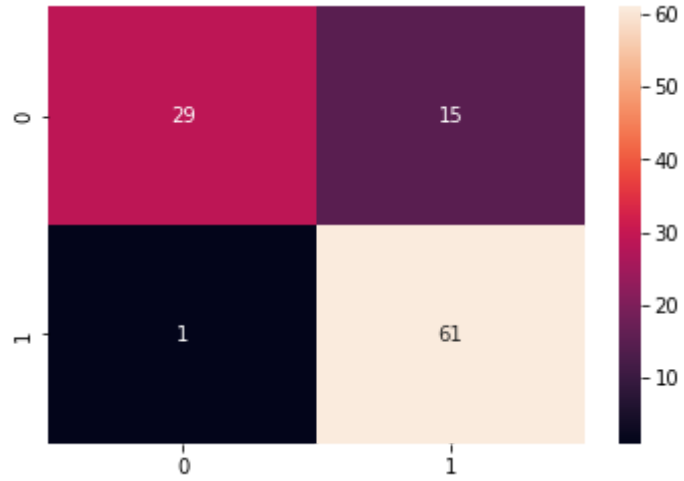
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 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 0x7f0fc96dcdd0>
```



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(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

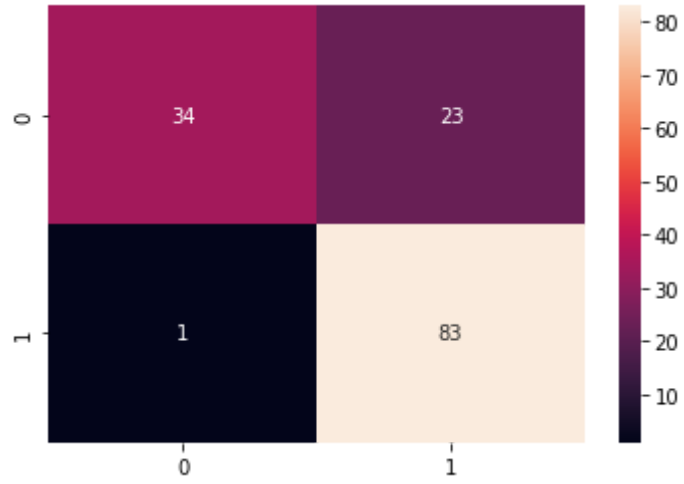
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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 0x7f0fc9673850>
```



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
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

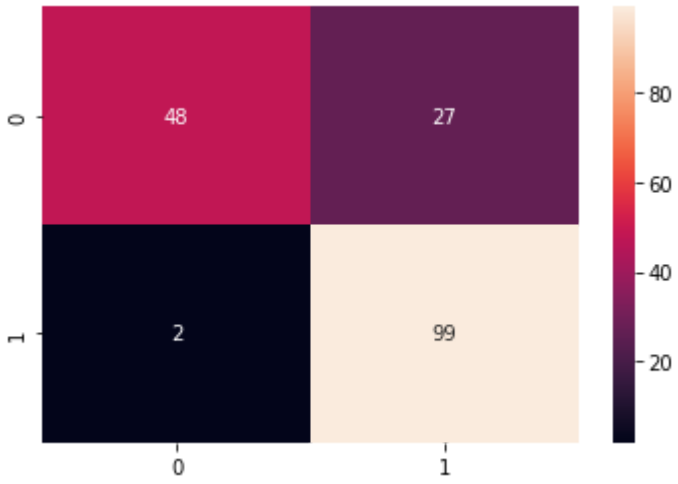
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
```

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

```
Accuracy: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc95aa290>
```



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
```

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

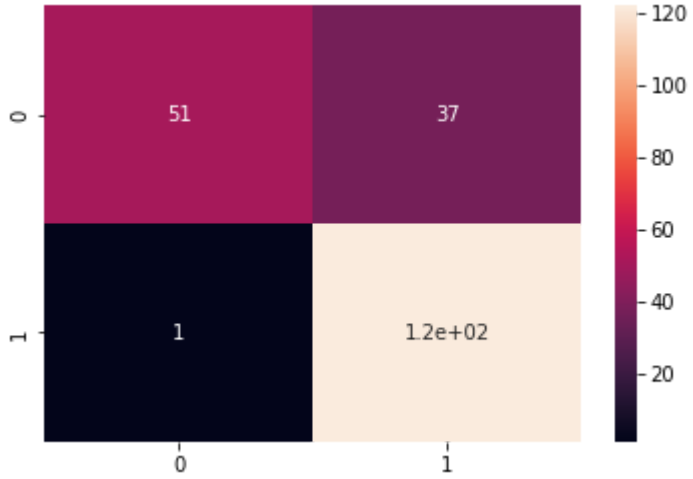
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc94da4d0>
```



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
```

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

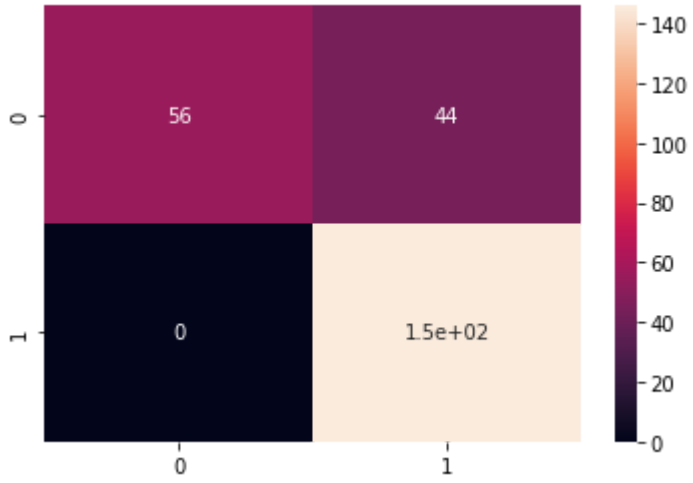
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 82.11382113821138%

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
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9476dd0>
```



MLP Classifier

```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
mlp_classifier
```

```
MLPClassifier(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=600,
              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)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

```
MLPClassifier(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=600,
              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)
```

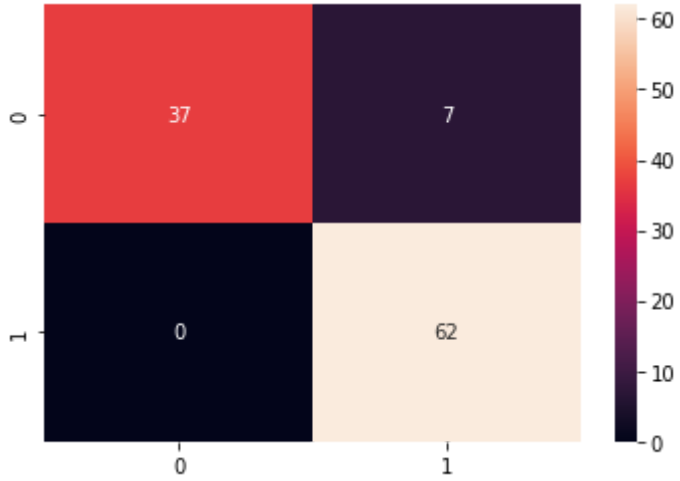
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.39622641509435%

Confusion Matrix:

[[37  7]
 [ 0 62]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.84      0.91        44
           g       0.90      1.00      0.95        62

    accuracy                           0.93       106
```

```
   macro avg       0.95      0.92      0.93       106
weighted avg       0.94      0.93      0.93       106
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc93b3250>
```



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
```

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

```
MLPClassifier(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=600,
              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)
```

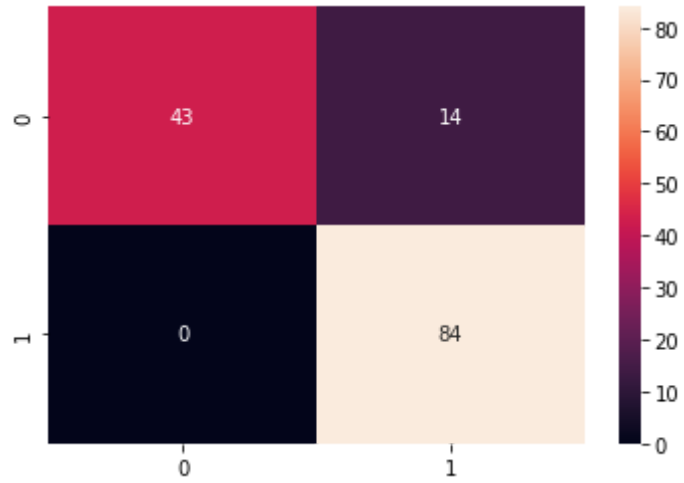
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.0709219858156%
```

```
Confusion Matrix:
[[43 14]
 [ 0 84]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.75      0.86        57
           g       0.86      1.00      0.92        84

    accuracy                           0.90       141
   macro avg       0.93      0.88      0.89       141
weighted avg       0.91      0.90      0.90       141
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9340950>
```



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
```

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

```
MLPClassifier(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=600,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
```

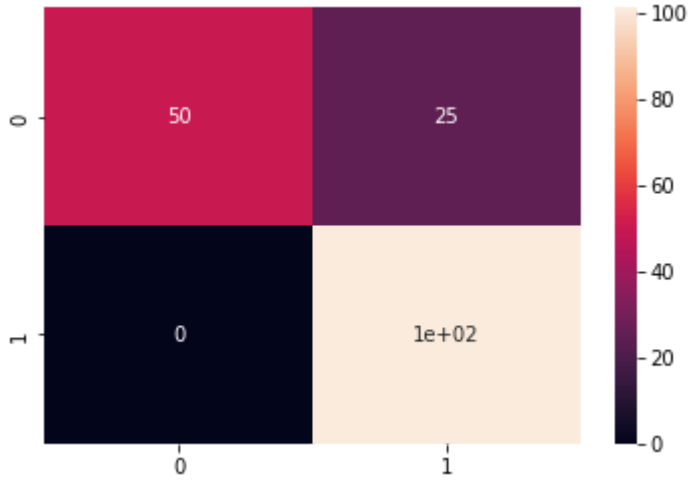
```
                   tol=0.0001, validation_fraction=0.1, verbose=False,
                   warm_start=False)
```

```
y_pred = mlp_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: 85.79545454545455%

Confusion Matrix:
[[ 50  25]
 [  0 101]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.67      0.80        75
           g       0.80      1.00      0.89       101

    accuracy                           0.86       176
   macro avg       0.90      0.83      0.84       176
weighted avg       0.89      0.86      0.85       176
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9277f50>
```



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
```

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

```
MLPClassifier(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=600,
              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)
```

```
y_pred = mlp_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: 84.36018957345972%

Confusion Matrix:

[[ 56  32]
 [  1 122]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.64      0.77        88
           g       0.79      0.99      0.88       123

    accuracy                           0.84       211
   macro avg       0.89      0.81      0.83       211
weighted avg       0.87      0.84      0.84       211
```

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

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
```

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

```
MLPClassifier(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=600,
              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)
```

```
y_pred = mlp_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: 84.5528455284553%

Confusion Matrix:

[[ 62  38]
 [  0 146]]

Classification Report:

              precision    recall  f1-score   support

           b       1.00      0.62      0.77       100
           g       0.79      1.00      0.88       146

    accuracy                           0.85       246
   macro avg       0.90      0.81      0.83       246
weighted avg       0.88      0.85      0.84       246
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc90e4a50>
```

Random Forest Classifier

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

train size : test size = 70% : 30%

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

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

```
245
106
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

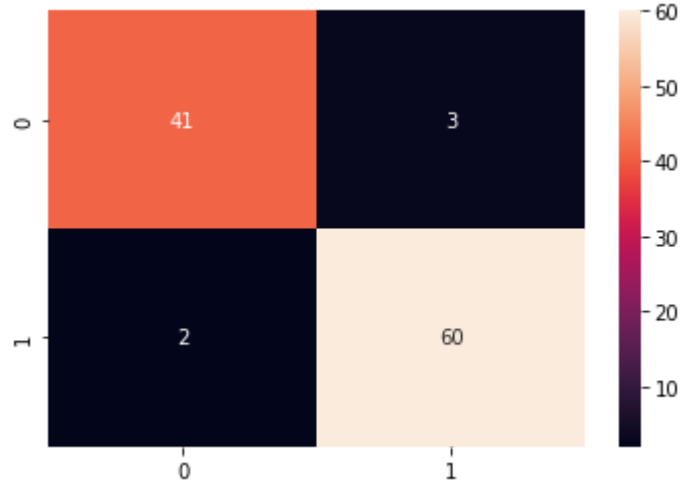
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf matrix = confusion matrix(y test,y pred)
```

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

```
Accuracy: 95.28301886792453%

Confusion Matrix:

[[41  3]
 [ 2 60]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.93      0.94        44
           g       0.95      0.97      0.96        62

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc95aa750>
```



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
```

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

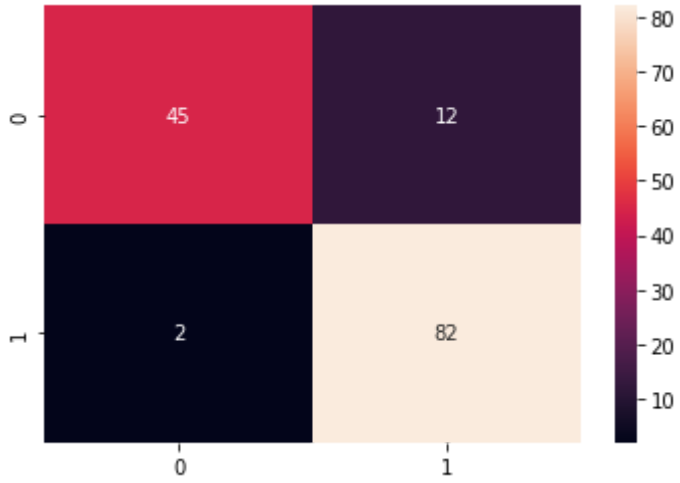
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

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

```
Accuracy: 90.0709219858156%

Confusion Matrix:
[[45 12]
 [ 2 82]]

Classification Report:

              precision    recall  f1-score   support

           b       0.96      0.79      0.87        57
           g       0.87      0.98      0.92        84

    accuracy                           0.90       141
   macro avg       0.91      0.88      0.89       141
weighted avg       0.91      0.90      0.90       141
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8f97410>
```



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
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

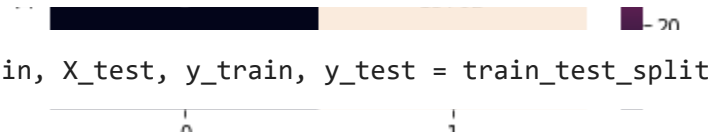
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 92.04545454545455%

Confusion Matrix:
[[ 62  13]
 [  1 100]]

Classification Report:

              precision    recall  f1-score   support

           b       0.98      0.83      0.90        75
           g       0.88      0.99      0.93       101

    accuracy                           0.92       176
   macro avg       0.93      0.91      0.92       176
weighted avg       0.93      0.92      0.92       176
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8f42090>
```

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
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

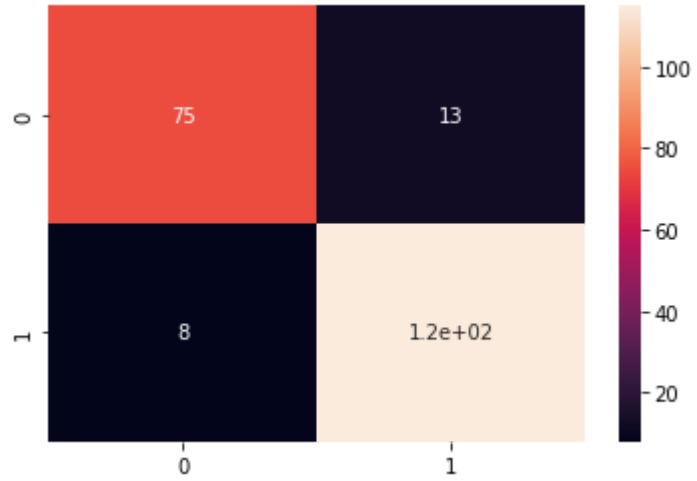
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 90.04739336492891%

Confusion Matrix:

[[ 75  13]
 [  8 115]]

Classification Report:

              precision    recall  f1-score   support

           b       0.90      0.85      0.88        88
           g       0.90      0.93      0.92       123

    accuracy                           0.90       211
   macro avg       0.90      0.89      0.90       211
```

```
weighted avg       0.90      0.90      0.90       211
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8e6b350>
```



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
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

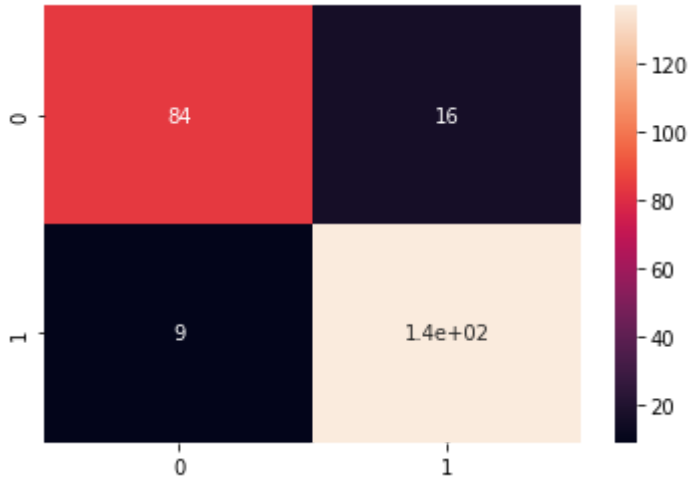
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 89.83739837398373%

Confusion Matrix:
```

```
[[ 84  16]
 [  9 137]]

Classification Report:

              precision    recall  f1-score   support

           b       0.90      0.84      0.87       100
           g       0.90      0.94      0.92       146

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8df4810>
```



NAME - SHUVRASISH ROY

ROLL NO. - 001811001012

MACHINE LEARNING LAB ASSIGNMENT 2

Import required modules

```
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

```
iris = datasets.load_iris() # it's source is same as : https://archive.ics.uci.edu/ml/data
```

```
dir(iris)
```

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

```
iris.data
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
```

```
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
```

```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
df["target"] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

DataFrame ready to perform

```
df["flower_names"] = df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_names
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
len(df)
```

```
150
```

```
X = df.drop(["target", "flower_names"], axis="columns")
y = df.target
print(X.head())
print(y.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
0    0
1    0
2    0
3    0
```

```
4    0
Name: target, dtype: int64
```

SVC Classfier

Linear SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

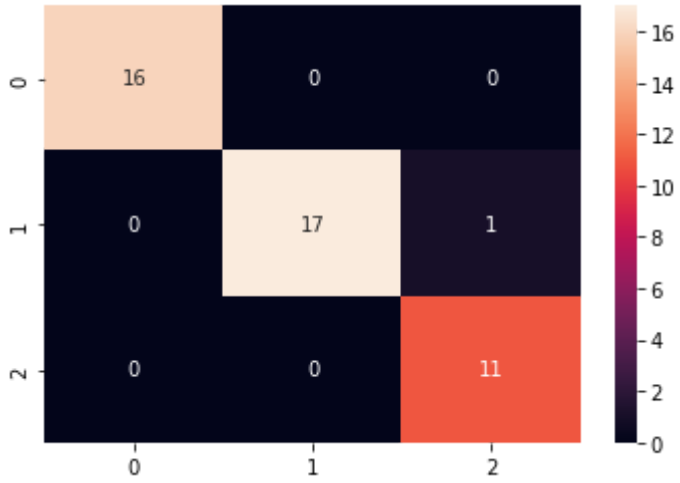
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.77777777777777%

Confusion Matrix:

[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]
```

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f21bc5290>
```



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))
```

```
90
60
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

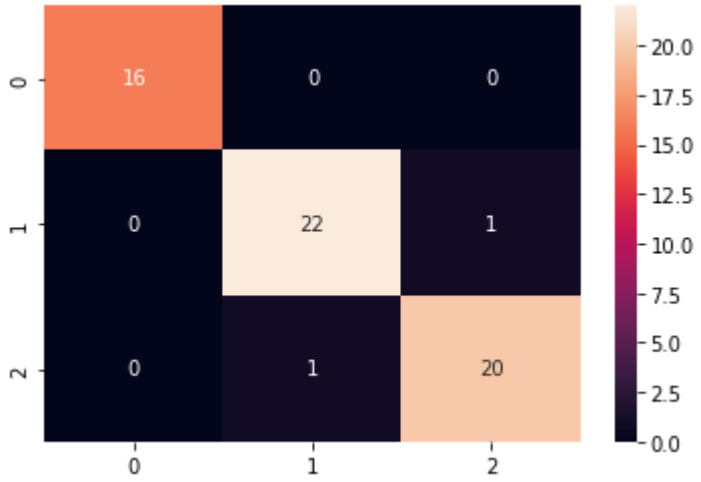
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
```

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

```
Accuracy: 96.66666666666667%

Confusion Matrix:
[[16  0  0]
 [ 0 22  1]
 [ 0  1 20]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.96      0.96      0.96        23
           2       0.95      0.95      0.95        21

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19629690>
```



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))
```

```
75
75
```

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

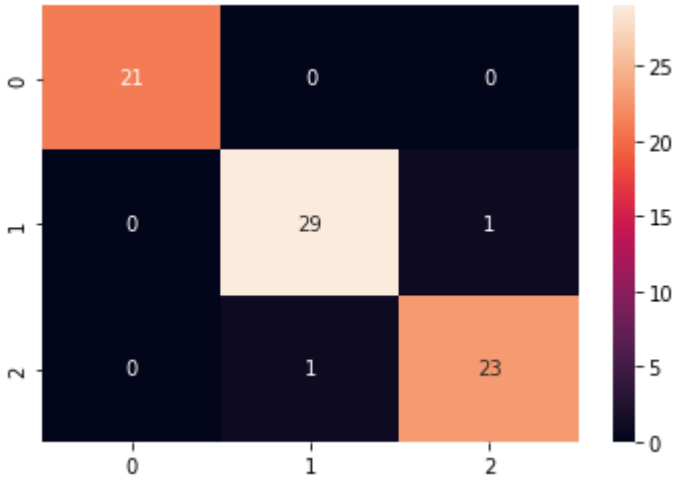
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 97.33333333333334%

Confusion Matrix:
[[21  0  0]
 [ 0 29  1]
 [ 0  1 23]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.97      0.97      0.97        30
           2       0.96      0.96      0.96        24

    accuracy                           0.97        75
   macro avg       0.98      0.98      0.98        75
weighted avg       0.97      0.97      0.97        75
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f195676d0>
```



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))
```

```
60
90
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

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

```
Accuracy: 96.66666666666667%

Confusion Matrix:

[[26  0  0]
 [ 0 32  1]
 [ 0  2 29]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.94      0.97      0.96        33
           2       0.97      0.94      0.95        31

    accuracy                           0.97        90
   macro avg       0.97      0.97      0.97        90
weighted avg       0.97      0.97      0.97        90
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19515250>
```

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))
```

```
45
105
```

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

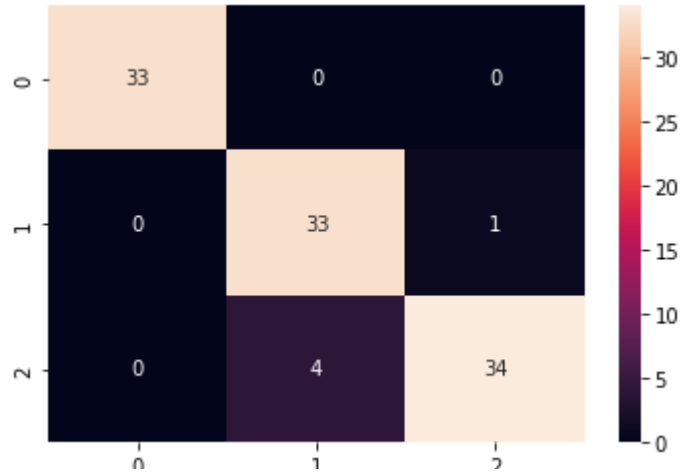
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

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

```
Accuracy: 95.23809523809523%

Confusion Matrix:

[[33  0  0]
 [ 0 33  1]
 [ 0  4 34]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.89      0.97      0.93        34
           2       0.97      0.89      0.93        38

    accuracy                           0.95       105
   macro avg       0.95      0.96      0.95       105
weighted avg       0.95      0.95      0.95       105
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f193e25d0>
```



Polynomial SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

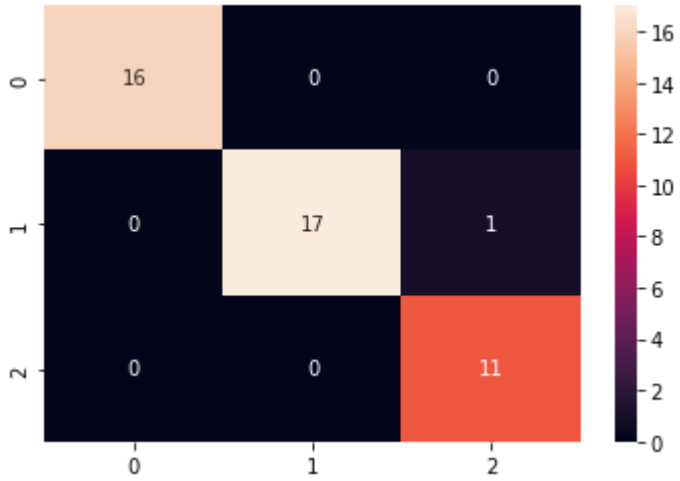
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.77777777777777%
```

```
Confusion Matrix:
 [[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1939d910>
```



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))
```

```
90
60
```

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

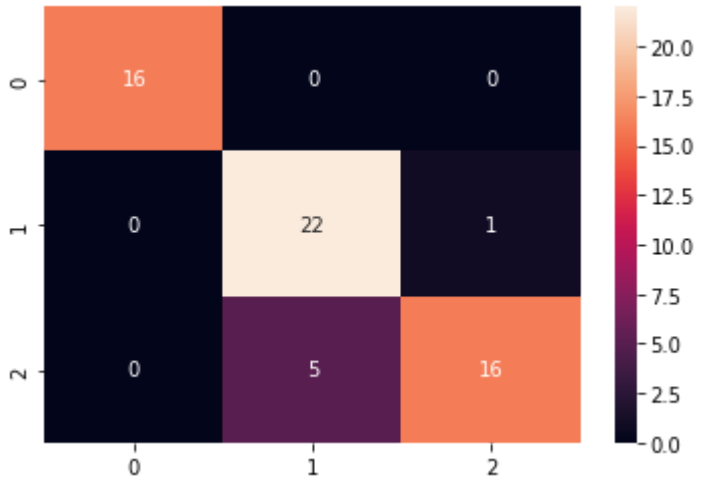
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 90.0%

Confusion Matrix:
 [[16  0  0]
 [ 0 22  1]
 [ 0  5 16]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.81      0.96      0.88        23
           2       0.94      0.76      0.84        21

    accuracy                           0.90        60
   macro avg       0.92      0.91      0.91        60
weighted avg       0.91      0.90      0.90        60
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19265ad0>
```



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))
```

```
75
75
```

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

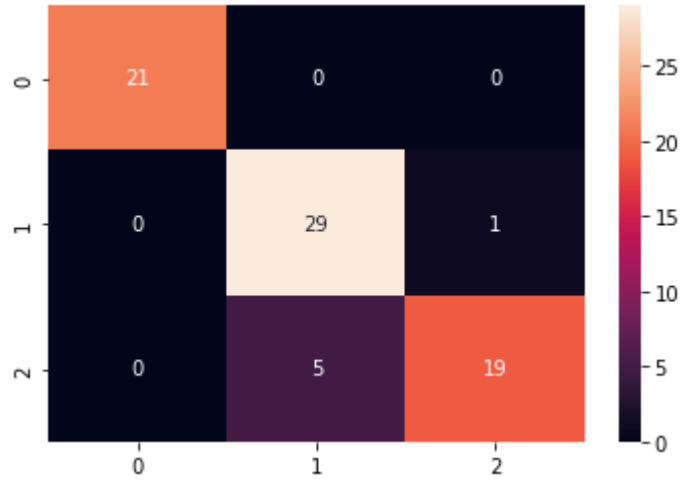
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 92.0%

Confusion Matrix:
 [[21  0  0]
 [ 0 29  1]
 [ 0  5 19]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.85      0.97      0.91        30
           2       0.95      0.79      0.86        24

    accuracy                           0.92        75
   macro avg       0.93      0.92      0.92        75
weighted avg       0.93      0.92      0.92        75
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1919ee10>
```



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))
```

```
60
90
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

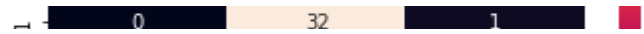
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.33333333333333%

Confusion Matrix:
 [[26  0  0]
 [ 0 32  1]
 [ 0  5 26]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.86      0.97      0.91        33
           2       0.96      0.84      0.90        31

    accuracy                           0.93        90
   macro avg       0.94      0.94      0.94        90
weighted avg       0.94      0.93      0.93        90
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19158210>
```

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))
```

```
45
105
```

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

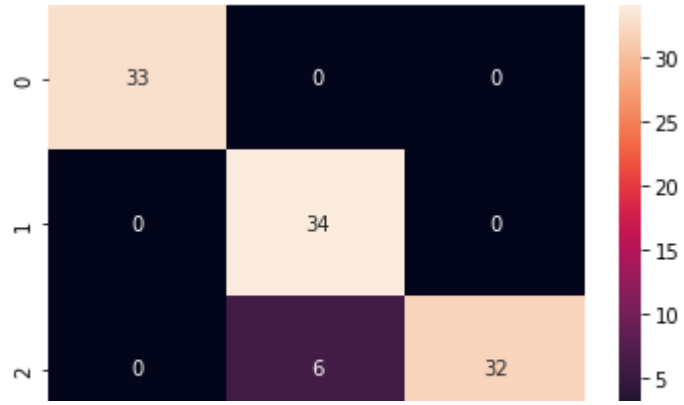
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 94.28571428571428%

Confusion Matrix:
 [[33  0  0]
 [ 0 34  0]
 [ 0  6 32]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.85      1.00      0.92        34
           2       1.00      0.84      0.91        38

    accuracy                           0.94       105
   macro avg       0.95      0.95      0.94       105
weighted avg       0.95      0.94      0.94       105
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f190d7310>
```



Gaussain SVC Classifier

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

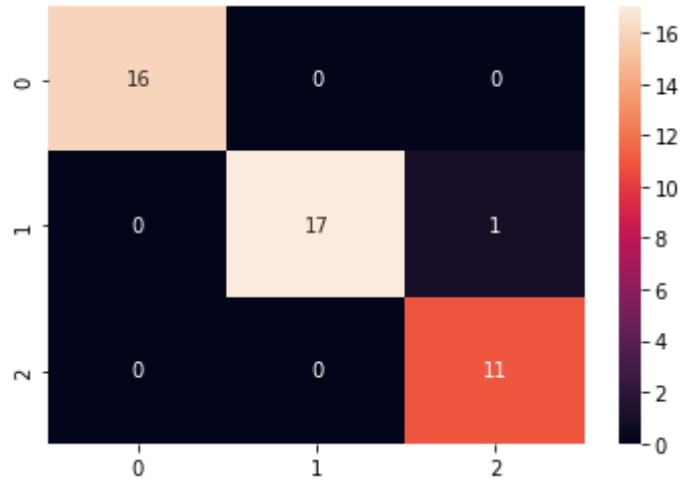
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.77777777777777%

Confusion Matrix:
```

```
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18fa53d0>
```



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))
```

```
90
60
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

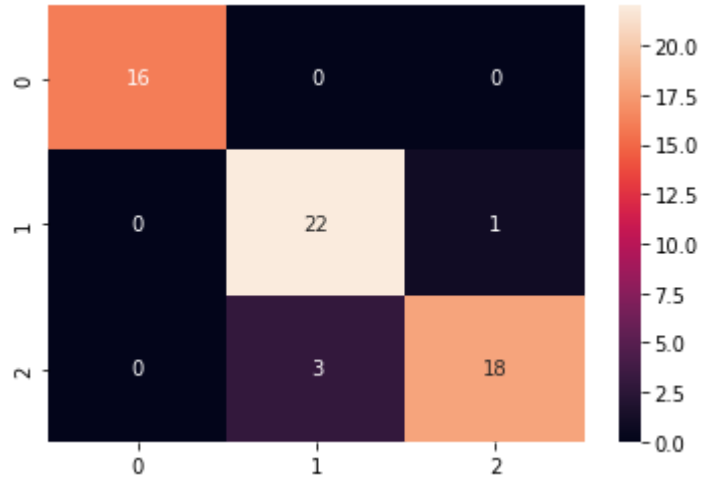
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test, y_pred)}%\n")
```

```
print("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: 93.33333333333333%

Confusion Matrix:
 [[16  0  0]
 [ 0 22  1]
 [ 0  3 18]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.88      0.96      0.92        23
           2       0.95      0.86      0.90        21

    accuracy                           0.93        60
   macro avg       0.94      0.94      0.94        60
weighted avg       0.94      0.93      0.93        60
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18efba90>
```



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))
```

```
75
75
```

```
gaussain SVC classifier.fit(X train, y train)
```

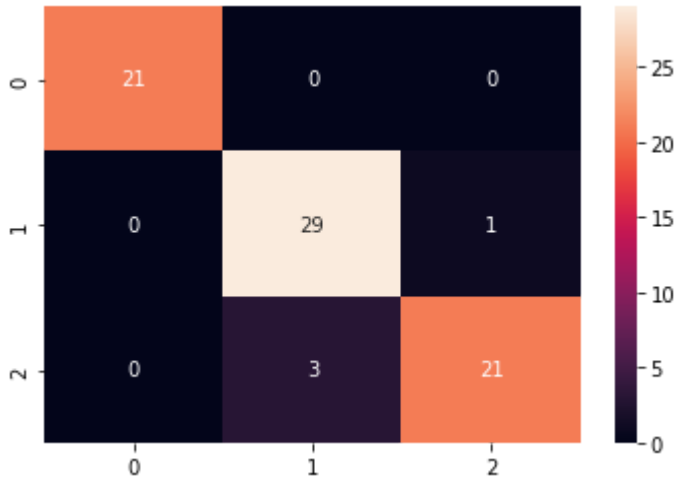
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 94.66666666666667%

Confusion Matrix:
 [[21  0  0]
 [ 0 29  1]
 [ 0  3 21]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.91      0.97      0.94        30
           2       0.95      0.88      0.91        24

    accuracy                           0.95        75
   macro avg       0.95      0.95      0.95        75
weighted avg       0.95      0.95      0.95        75
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18e43e50>
```



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))
```

```
60
90
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 93.33333333333333%

Confusion Matrix:
 [[26  0  0]
 [ 0 32  1]
 [ 0  5 26]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.86      0.97      0.91        33
           2       0.96      0.84      0.90        31

    accuracy                           0.93        90
   macro avg       0.94      0.94      0.94        90
weighted avg       0.94      0.93      0.93        90
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1929f510>
```

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))
```

```
45
105
```

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

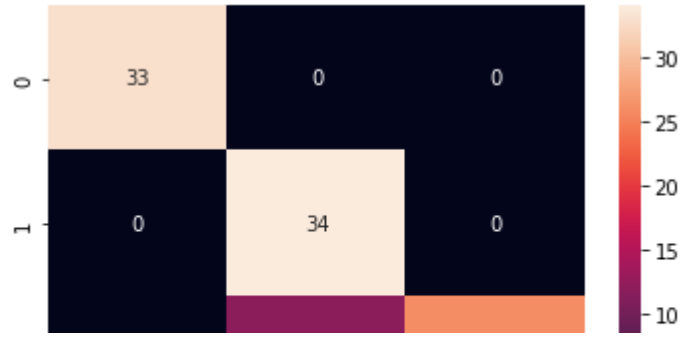
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = gaussain_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: 88.57142857142857%

Confusion Matrix:
 [[33  0  0]
 [ 0 34  0]
 [ 0 12 26]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.74      1.00      0.85        34
           2       1.00      0.68      0.81        38

    accuracy                           0.89       105
   macro avg       0.91      0.89      0.89       105
weighted avg       0.92      0.89      0.88       105
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18d13e50>
```



Sigmoid SVC Classifier

```
0 1 2
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

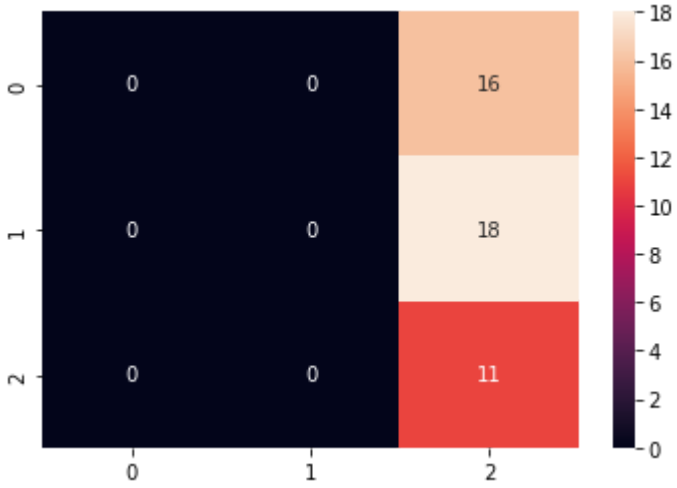
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 24.444444444444443%

Confusion Matrix:
 [[ 0  0 16]
 [ 0  0 18]
```

```
 [ 0  0 11]]

Classification Report:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.00      0.00      0.00        18
           2       0.24      1.00      0.39        11

    accuracy                           0.24        45
   macro avg       0.08      0.33      0.13        45
weighted avg       0.06      0.24      0.10        45

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18bfce50>
```



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))
```

```
90
60
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

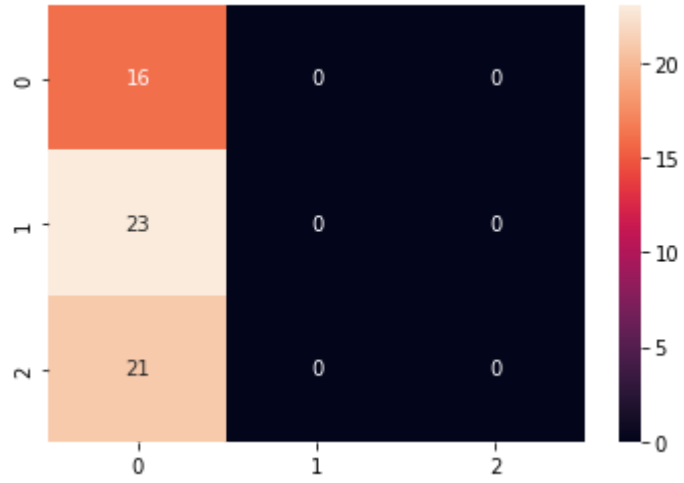
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
```

```
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: 26.666666666666668%

Confusion Matrix:
 [[16  0  0]
 [23  0  0]
 [21  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.27      1.00      0.42        16
           1       0.00      0.00      0.00        23
           2       0.00      0.00      0.00        21

    accuracy                           0.27        60
   macro avg       0.09      0.33      0.14        60
weighted avg       0.07      0.27      0.11        60

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f240a5590>
```



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))
```

```
75
75
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 28.000000000000004%

Confusion Matrix:
 [[21  0  0]
 [30  0  0]
 [24  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.28      1.00      0.44        21
           1       0.00      0.00      0.00        30
           2       0.00      0.00      0.00        24

    accuracy                           0.28        75
   macro avg       0.09      0.33      0.15        75
weighted avg       0.08      0.28      0.12        75

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18a70c90>
```

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))
```

```
60
90
```

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

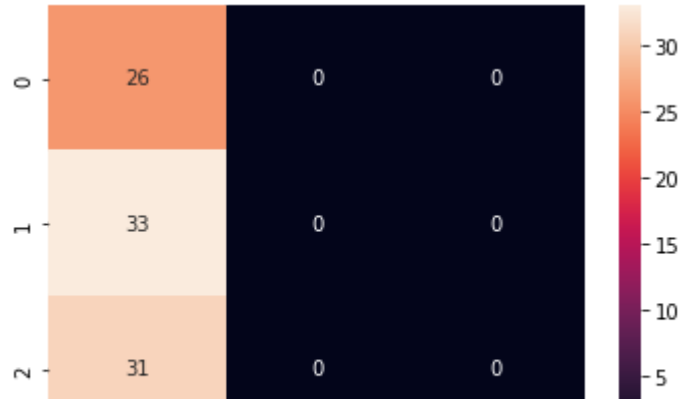
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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: 28.888888888888886%

Confusion Matrix:
 [[26  0  0]
 [33  0  0]
 [31  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.29      1.00      0.45        26
           1       0.00      0.00      0.00        33
           2       0.00      0.00      0.00        31

    accuracy                           0.29        90
   macro avg       0.10      0.33      0.15        90
weighted avg       0.08      0.29      0.13        90

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f189b4190>
```



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))
```

```
45
105
```

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

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

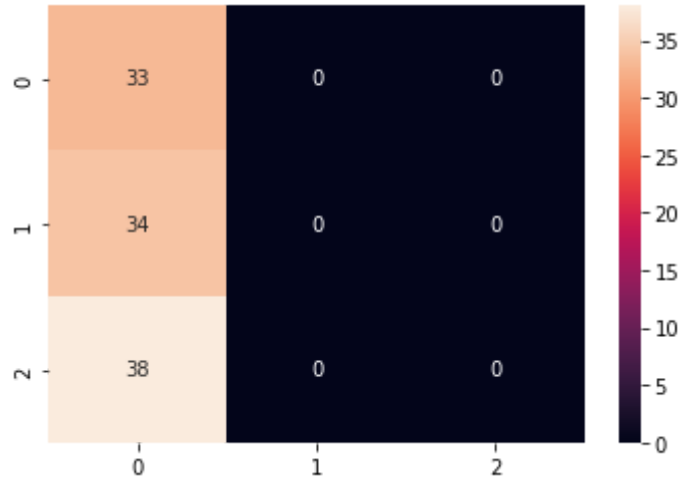
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 31.428571428571427%

Confusion Matrix:
 [[33  0  0]
 [34  0  0]
 [38  0  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.31      1.00      0.48        33
           1       0.00      0.00      0.00        34
           2       0.00      0.00      0.00        38

    accuracy                           0.31       105
   macro avg       0.10      0.33      0.16       105
```

```
weighted avg       0.10      0.31      0.15       105

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f188f7390>
```



MLP Classifier

```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
mlp_classifier
```

```
MLPClassifier(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=600,
              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)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

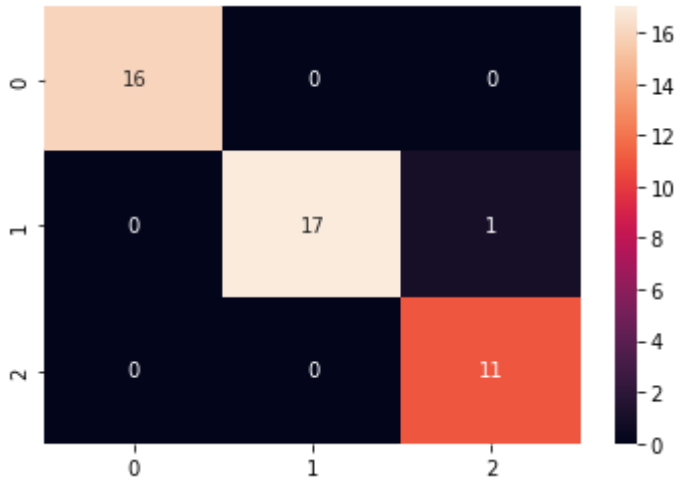
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9, 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)
```

```
y_pred = mlp_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: 97.77777777777777%

Confusion Matrix:

[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18839510>
```



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))
```

```
90
60
```

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

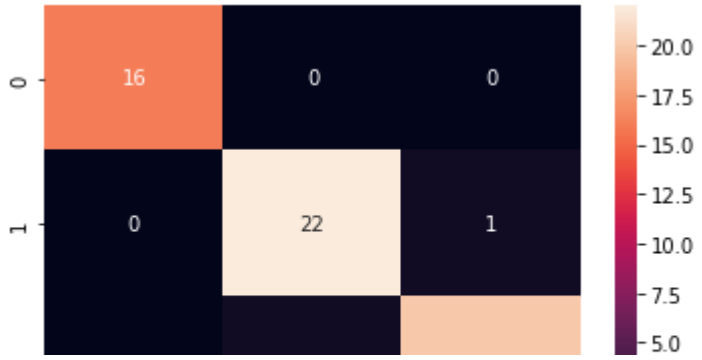
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9, 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)
```

```
y_pred = mlp_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: 96.66666666666667%

Confusion Matrix:
[[16  0  0]
 [ 0 22  1]
 [ 0  1 20]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.96      0.96      0.96        23
           2       0.95      0.95      0.95        21

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1876f890>
```



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))
```

```
75
75
```

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

```
MLPClassifier(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=600,
              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)
```

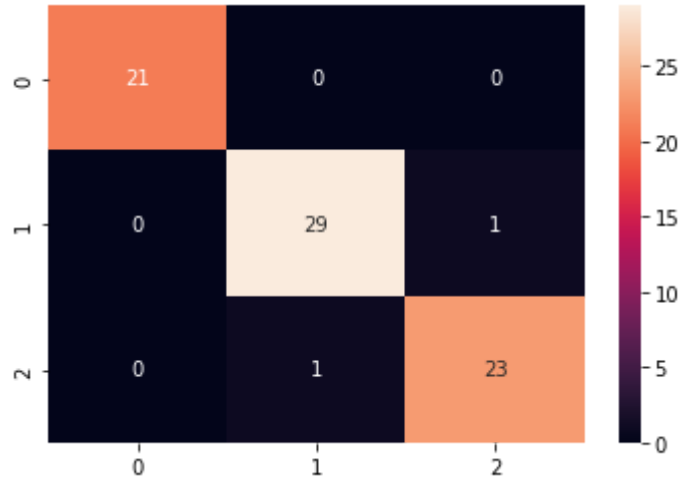
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.33333333333334%

Confusion Matrix:
[[21  0  0]
 [ 0 29  1]
 [ 0  1 23]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.97      0.97      0.97        30
```

```
                2       0.96      0.96      0.96        24

         accuracy                           0.97        75
        macro avg       0.98      0.98      0.98        75
     weighted avg       0.97      0.97      0.97        75
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f186af650>
```



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))
```

```
60
90
```

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

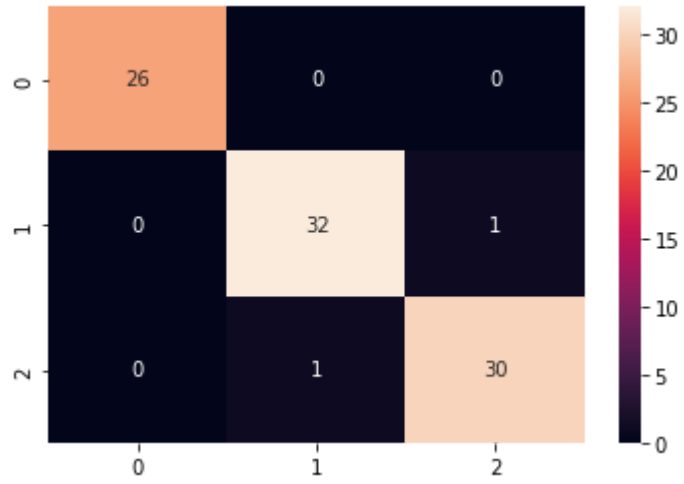
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9, 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)
```

```
y_pred = mlp_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: 97.77777777777777%

Confusion Matrix:

[[26  0  0]
 [ 0 32  1]
 [ 0  1 30]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.97      0.97      0.97        33
           2       0.97      0.97      0.97        31

    accuracy                           0.98        90
   macro avg       0.98      0.98      0.98        90
weighted avg       0.98      0.98      0.98        90
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f240a50d0>
```



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))
```

```
45
105
```

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

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
```

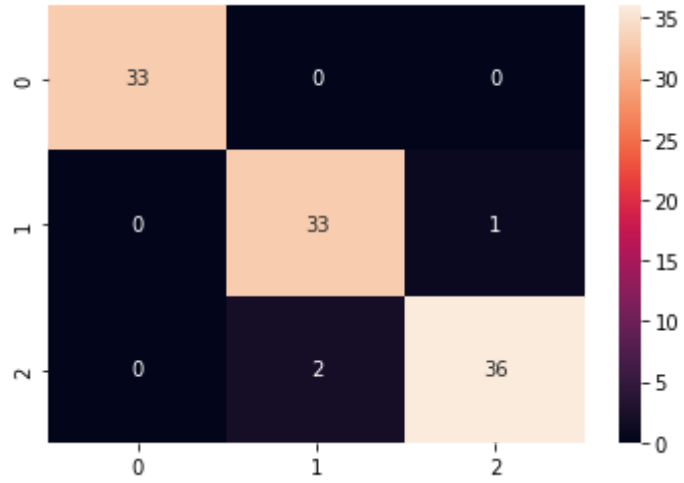
```
                    hidden_layer_sizes=(100,), learning_rate='constant',
                    learning_rate_init=0.001, max_fun=15000, max_iter=600,
                    momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                    power_t=0.5, random_state=None, shuffle=True, solver='adam',
                    tol=0.0001, validation_fraction=0.1, verbose=False,
                    warm_start=False)
```

```
y_pred = mlp_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: 97.14285714285714%

Confusion Matrix:

[[33  0  0]
 [ 0 33  1]
 [ 0  2 36]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.94      0.97      0.96        34
           2       0.97      0.95      0.96        38

    accuracy                           0.97       105
   macro avg       0.97      0.97      0.97       105
weighted avg       0.97      0.97      0.97       105
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18b7a050>
```



Random Forest Classifier

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

train size : test size = 70% : 30%

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

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

```
105
45
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

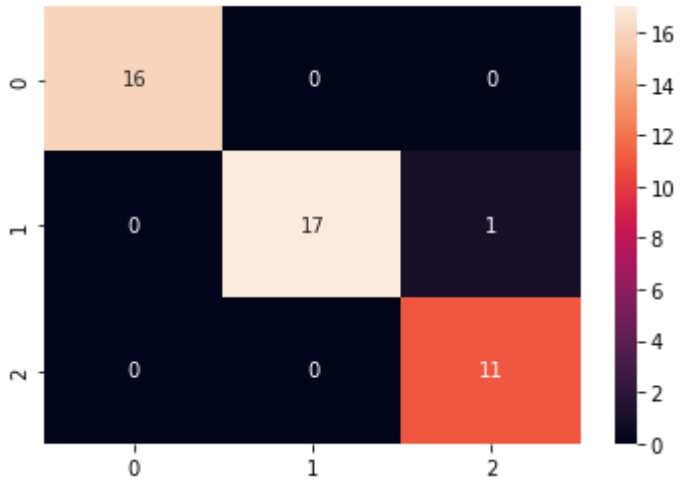
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 97.77777777777777%

Confusion Matrix:

[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]

Classification Report:

              precision    recall  f1-score   support
```

```
              0       1.00      1.00      1.00        16
              1       1.00      0.94      0.97        18
              2       0.92      1.00      0.96        11

       accuracy                           0.98        45
      macro avg       0.97      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f184b8750>
```



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))
```

```
90
60
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

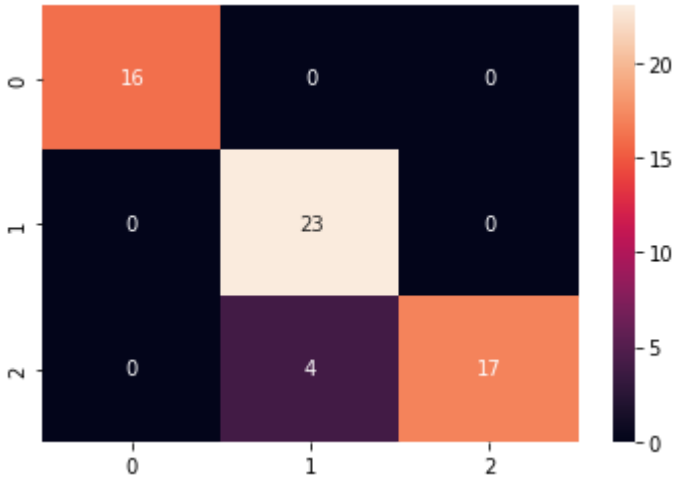
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
```

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

```
Accuracy: 93.33333333333333%

Confusion Matrix:
[[16  0  0]
 [ 0 23  0]
 [ 0  4 17]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.85      1.00      0.92        23
           2       1.00      0.81      0.89        21

    accuracy                           0.93        60
   macro avg       0.95      0.94      0.94        60
weighted avg       0.94      0.93      0.93        60
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f183f9850>
```



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))
```

```
75
75
```

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

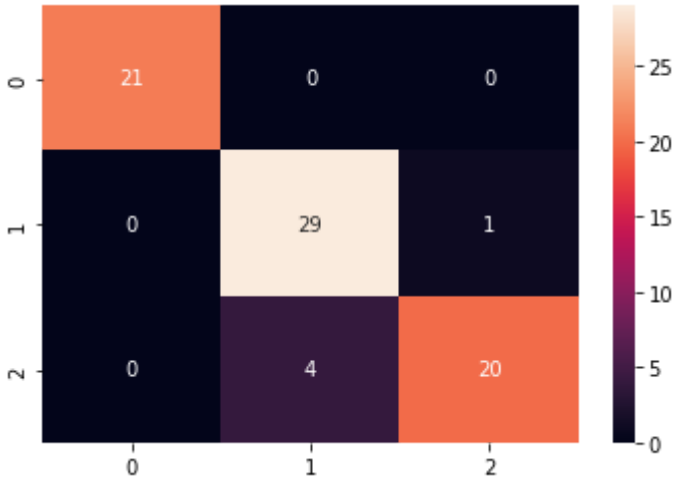
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

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

```
Accuracy: 93.33333333333333%

Confusion Matrix:
[[21  0  0]
 [ 0 29  1]
 [ 0  4 20]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.88      0.97      0.92        30
           2       0.95      0.83      0.89        24

    accuracy                           0.93        75
   macro avg       0.94      0.93      0.94        75
weighted avg       0.94      0.93      0.93        75
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1832e890>
```



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))
```

```
60
90
```

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

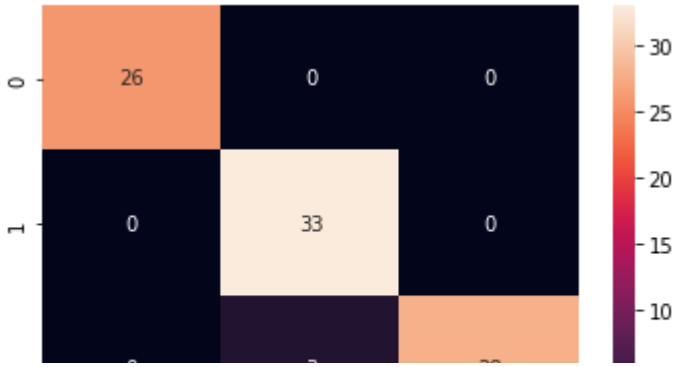
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
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: 96.66666666666667%

Confusion Matrix:

[[26  0  0]
 [ 0 33  0]
 [ 0  3 28]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.92      1.00      0.96        33
           2       1.00      0.90      0.95        31

    accuracy                           0.97        90
   macro avg       0.97      0.97      0.97        90
weighted avg       0.97      0.97      0.97        90
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18263650>
```



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))
```

```
45
105
```

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

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
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: 95.23809523809523%

Confusion Matrix:

[[33  0  0]
 [ 0 33  1]
 [ 0  4 34]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
```

```
                1       0.89      0.97      0.93        34
                2       0.97      0.89      0.93        38

         accuracy                           0.95       105
        macro avg       0.95      0.96      0.95       105
     weighted avg       0.95      0.95      0.95       105
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f181a1890>
```