JADAVPUR UNIVERSITY BE IT 4TH YEAR

NAME - JATIN SINGH CHUG

ROLL NO. - 001811001074

ML 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	colum
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77

```
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)
```

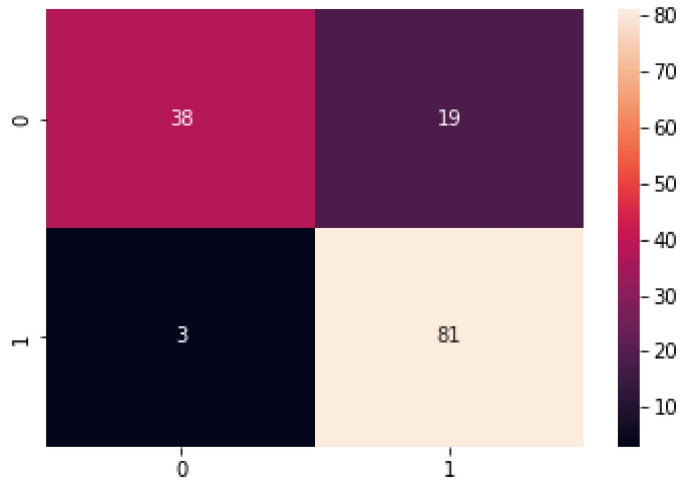
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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)
```

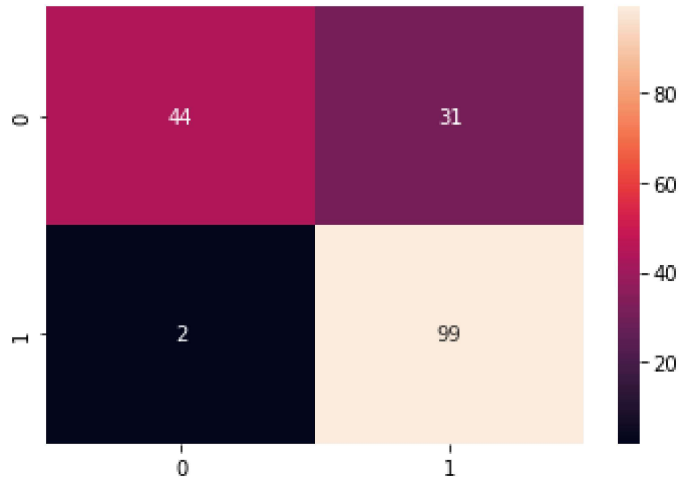
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

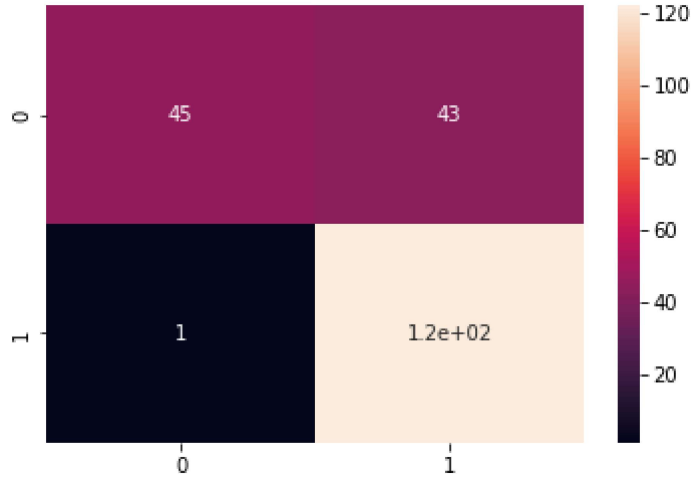
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
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)
```

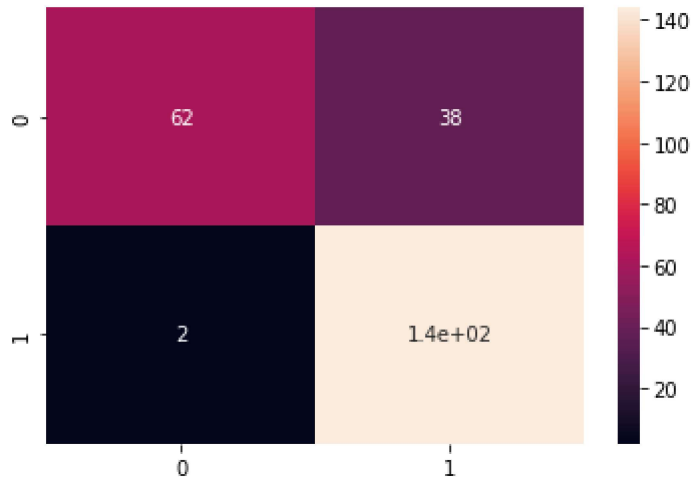
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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)
```

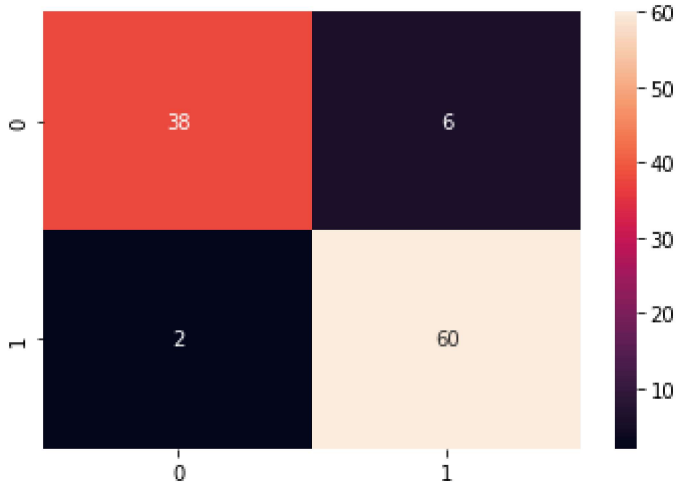
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

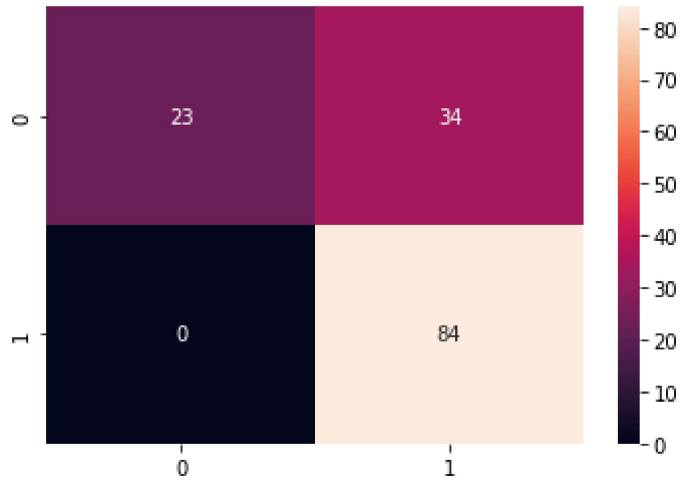
```
y_pred = poly_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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',
```

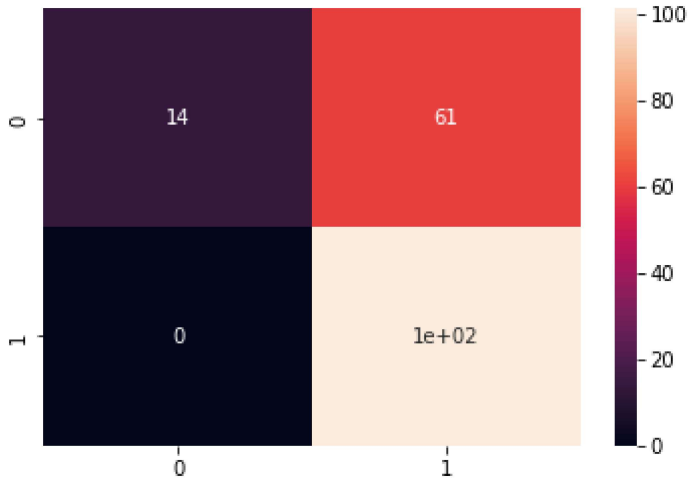
```
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)
```

```
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)
```

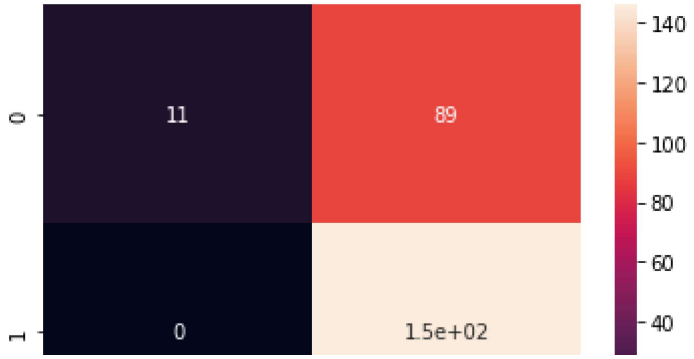
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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)
```

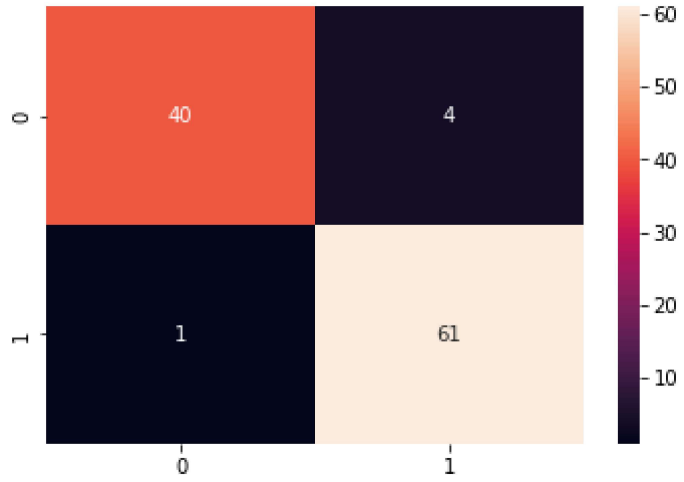
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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',
```

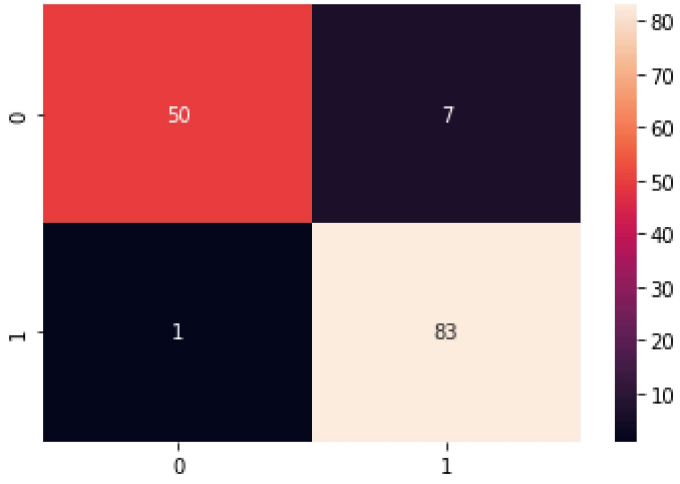
```
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)
```

```
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)
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)
```

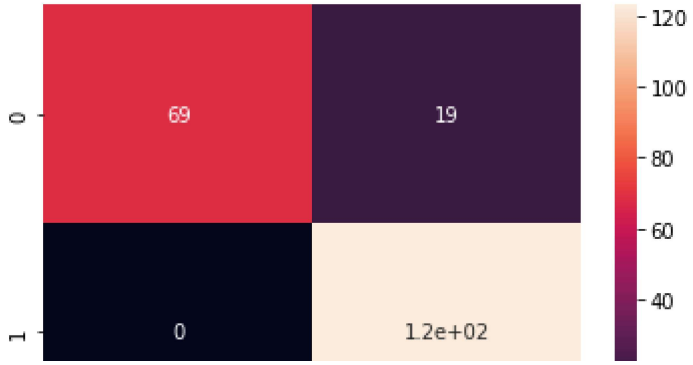
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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)
```

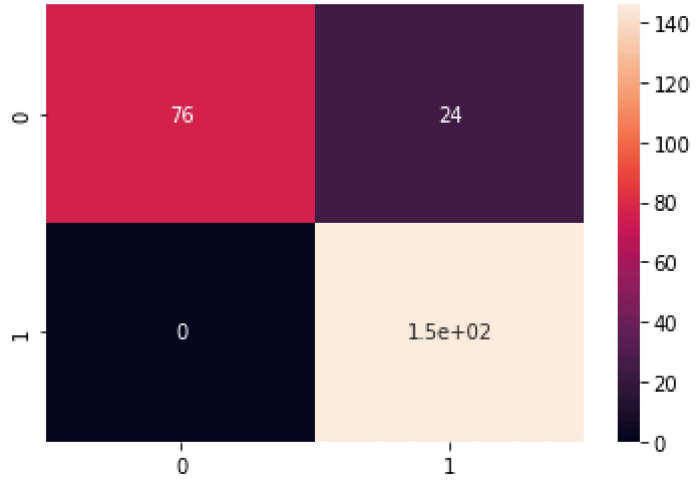
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

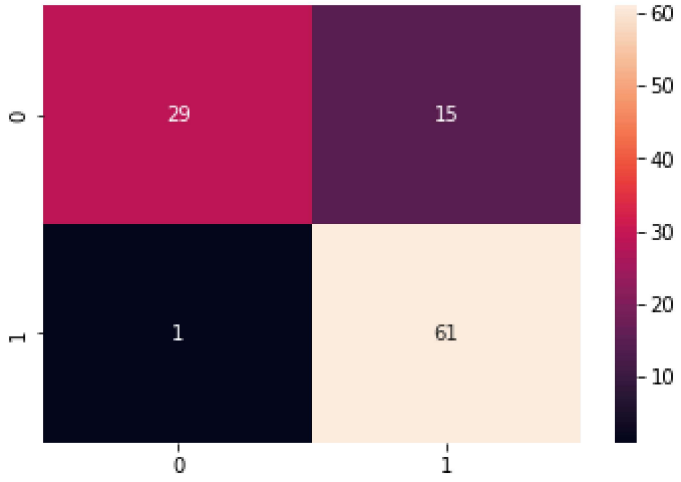
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
```

```
          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))
```

```
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)
```

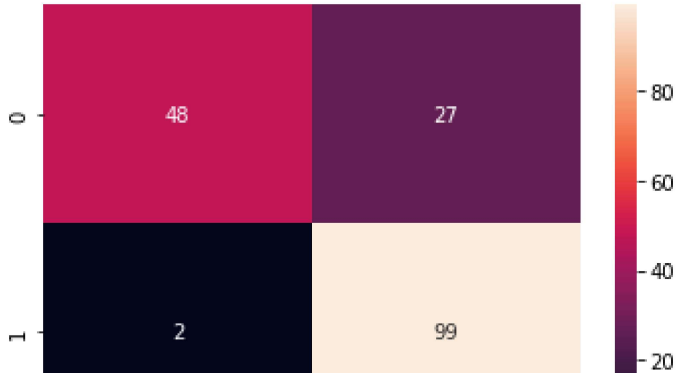
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
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)
```

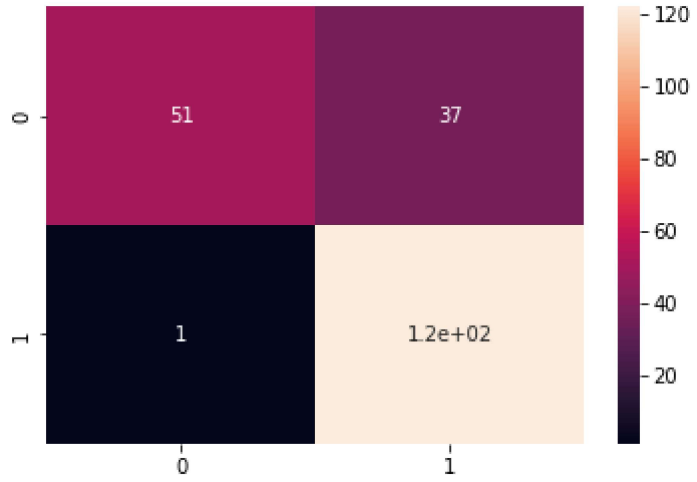
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

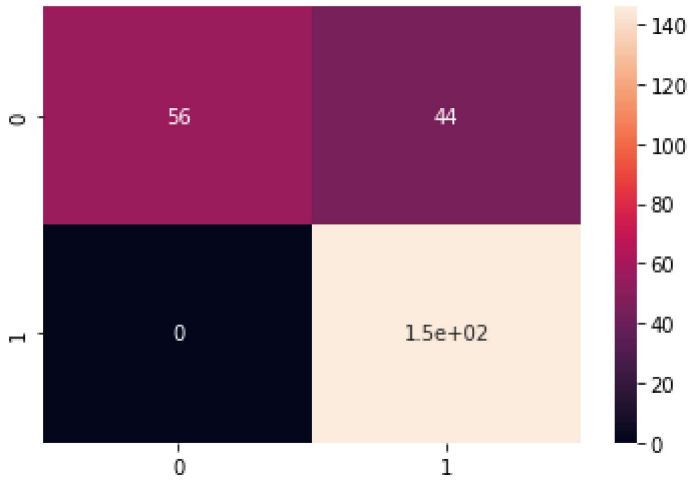
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

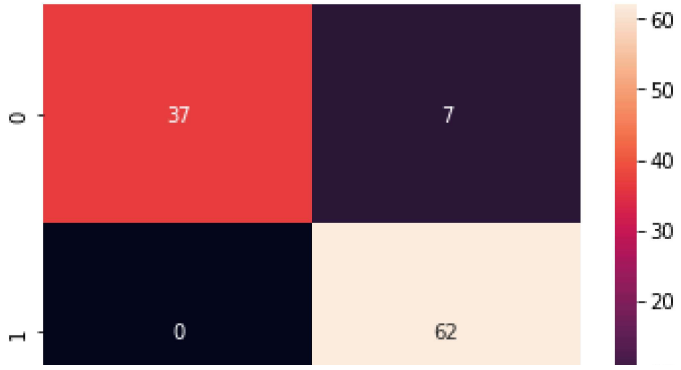
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              momentum=0.9,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

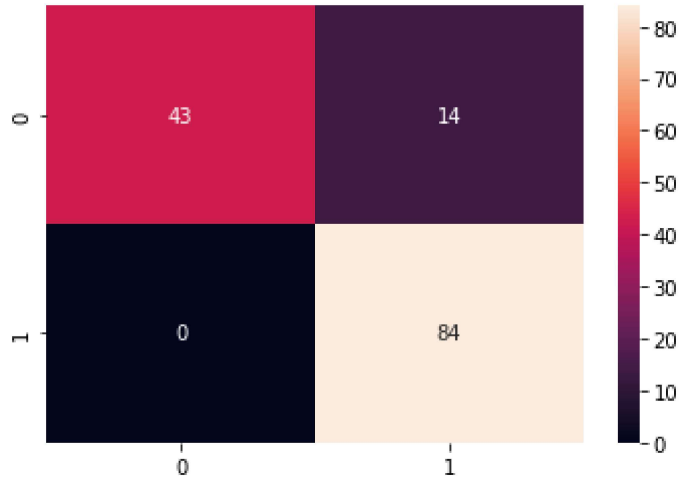
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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)
```

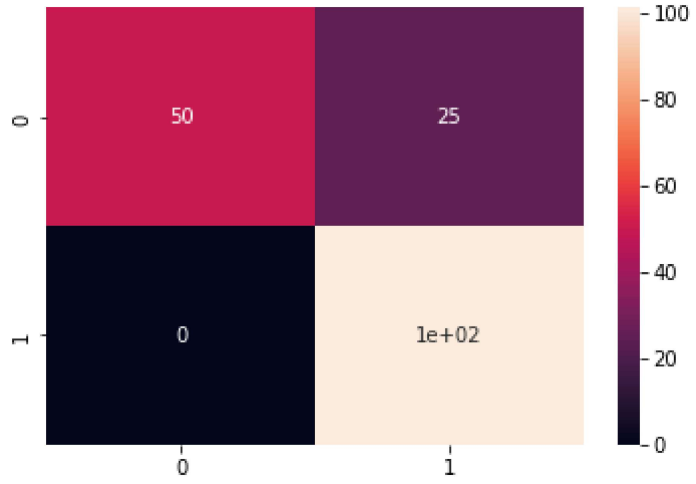
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
```

```
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)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc91ac8d0>
```

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)
```

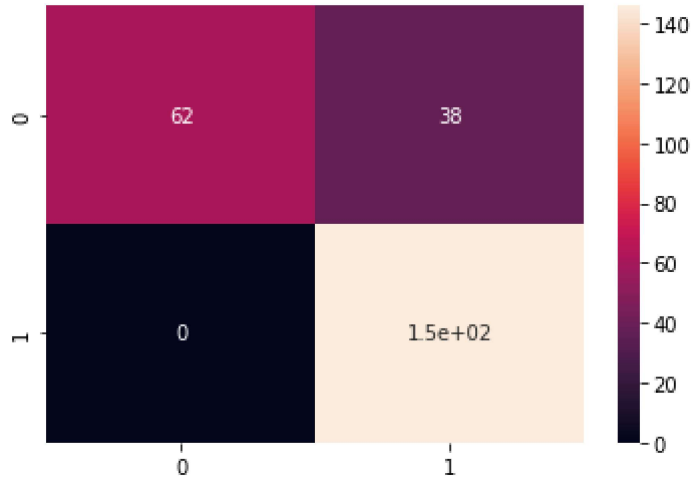
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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,
```

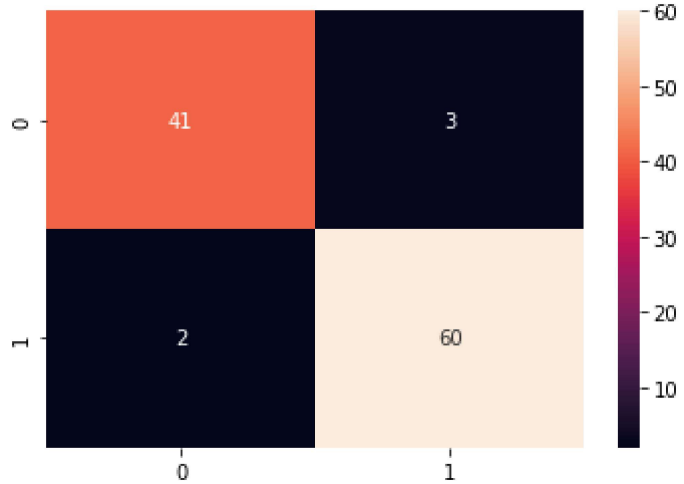
```
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       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)
```

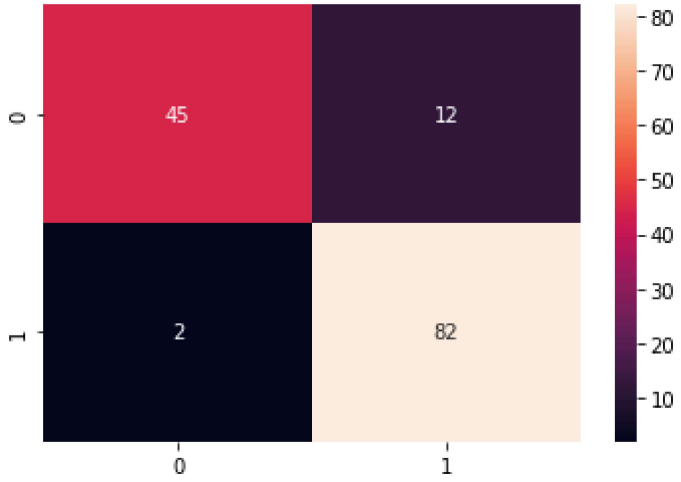
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

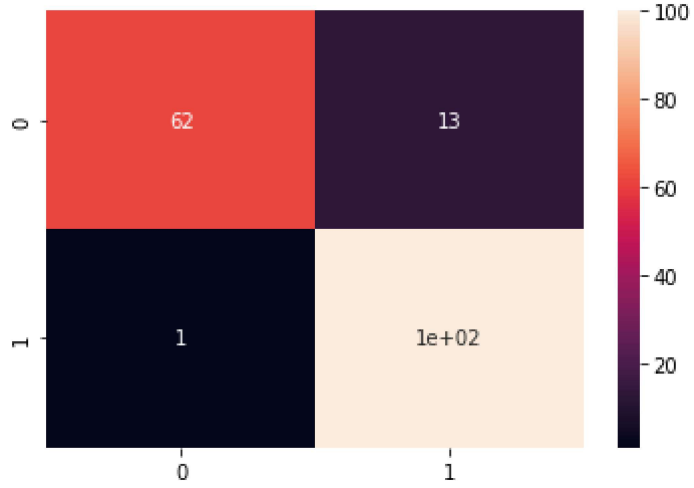
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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,
```

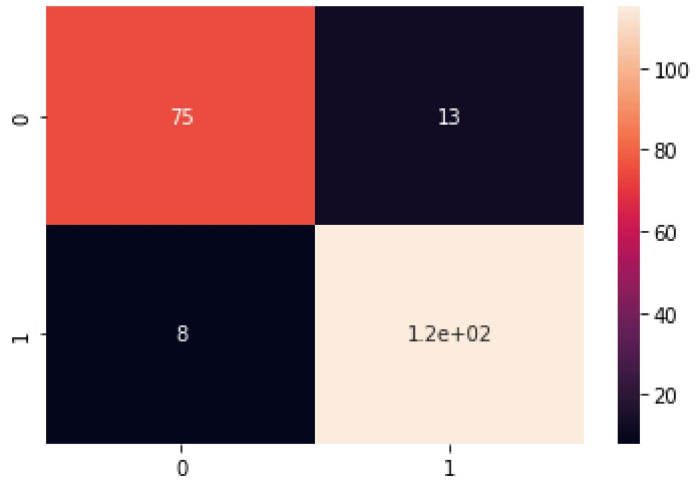
```
                               n_jobs=None, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)
```

```
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>
```