Swapnil Ghosh JU IT 001911001067

Import required modules

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.metrics import accuracy_score, classification_report, confusio
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

In [ ]:
```
iris = datasets.load_iris() # it's source is same as : https://archive.ics.
```

In [ ]:
```
dir(iris)
```

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

In [ ]:
```
iris.data
```

Out[ ]:
```
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],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
```

```
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
```

```
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [ ]:
```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df.head()
```

Out[ ]:

	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

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

Out[ ]:

	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

In [ ]:
```
iris.target_names
```

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

DataFrame ready to perform

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

Out[ ]:

	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

In [ ]:
```
len(df)
```

Out[ ]:
```
150
```

In [ ]:
```
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

In [ ]:
```
linear_SVC_classifier = SVC(kernel='linear')
linear_SVC_classifier
```

Out[ ]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

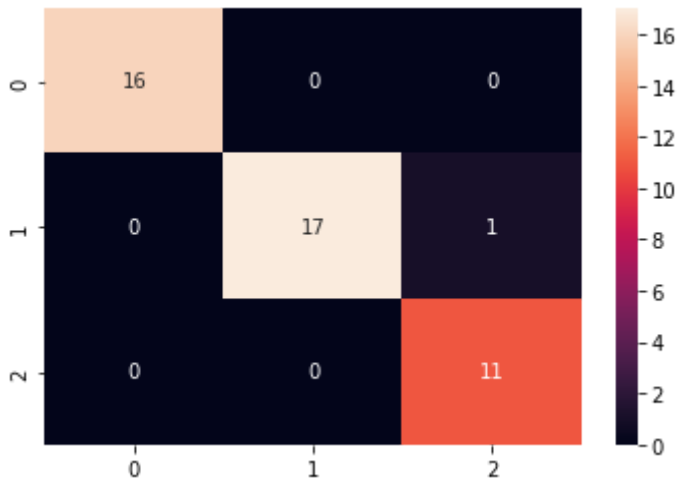
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f21bc5290>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

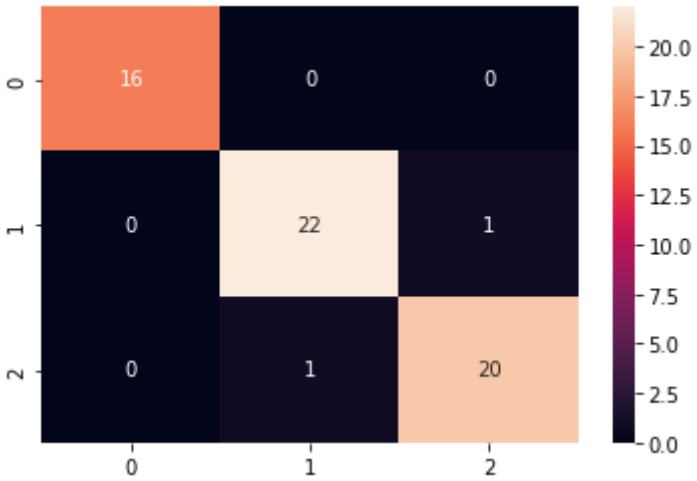
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19629690>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

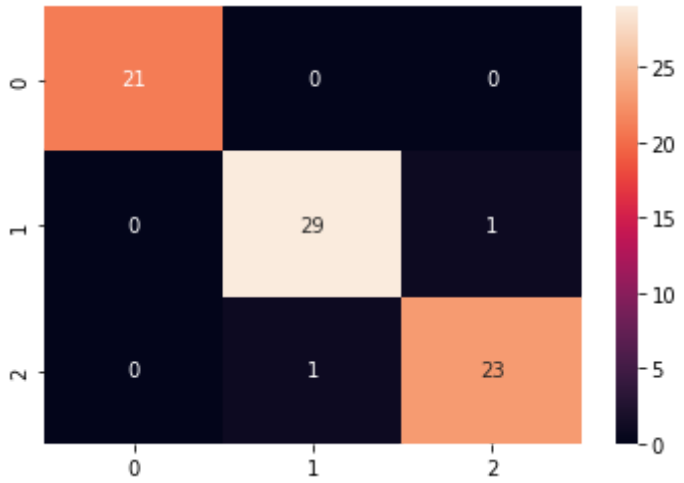
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f195676d0>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
60
90
```

In [ ]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

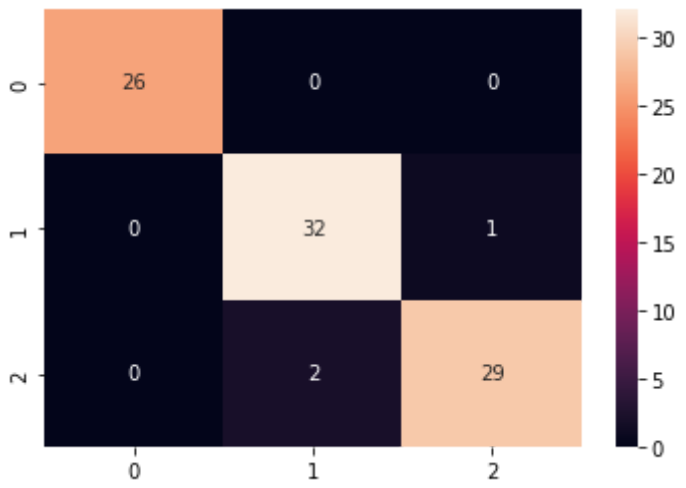
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19515250>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

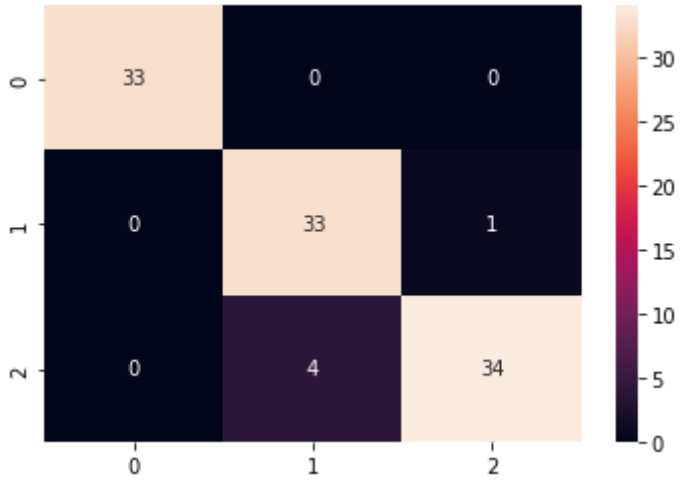
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f193e25d0>
```



Polynomial SVC Classifier

In [ ]:
```
poly_SVC_classifier = SVC(kernel='poly')
poly_SVC_classifier
```

Out[ ]:
```
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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

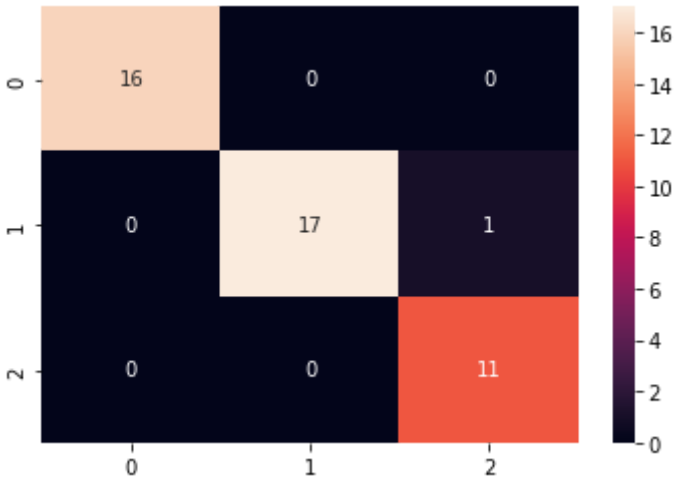
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1939d910>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

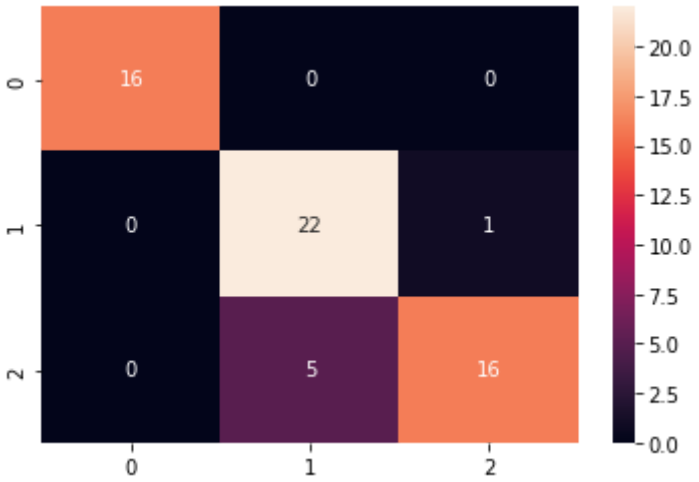
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19265ad0>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

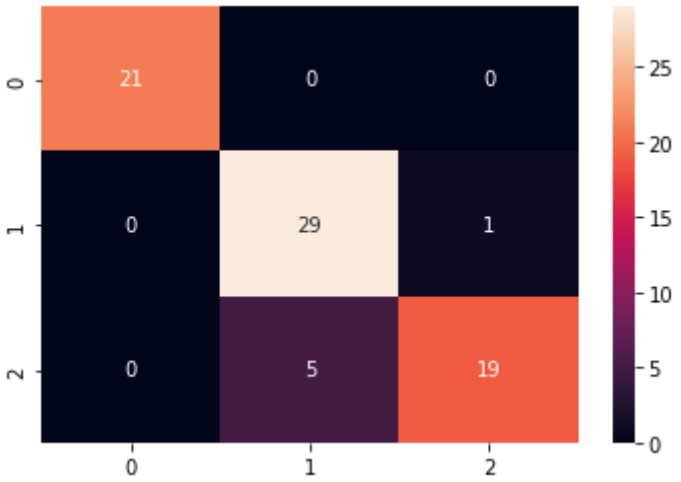
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1919ee10>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
60
90
```

In [ ]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

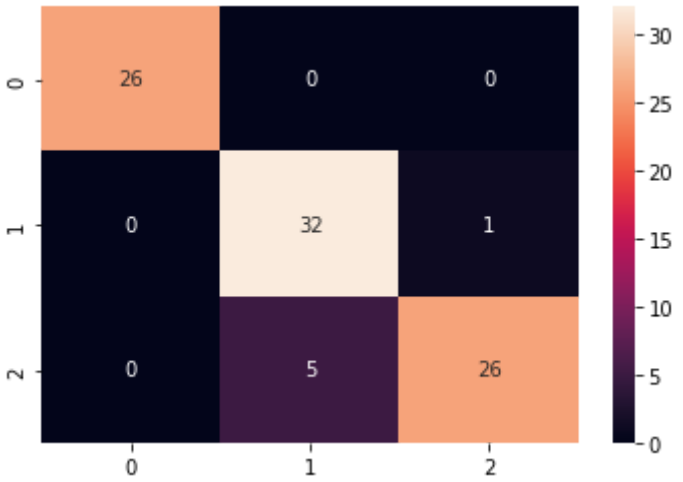
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f19158210>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

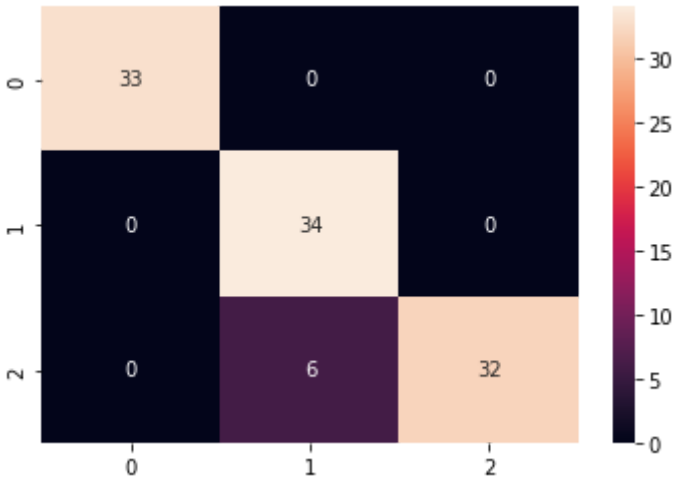
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f190d7310>
```



Gaussain SVC Classifier

In [ ]:
```
gaussain_SVC_classifier = SVC(kernel='rbf')
gaussain_SVC_classifier
```

Out[ ]:
```
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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

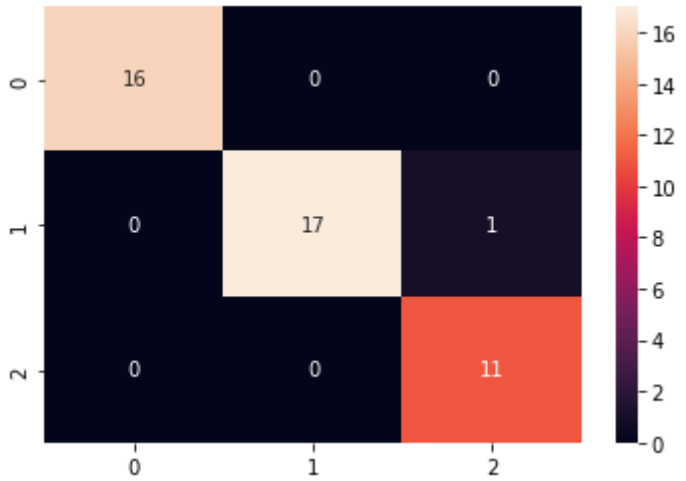
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18fa53d0>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

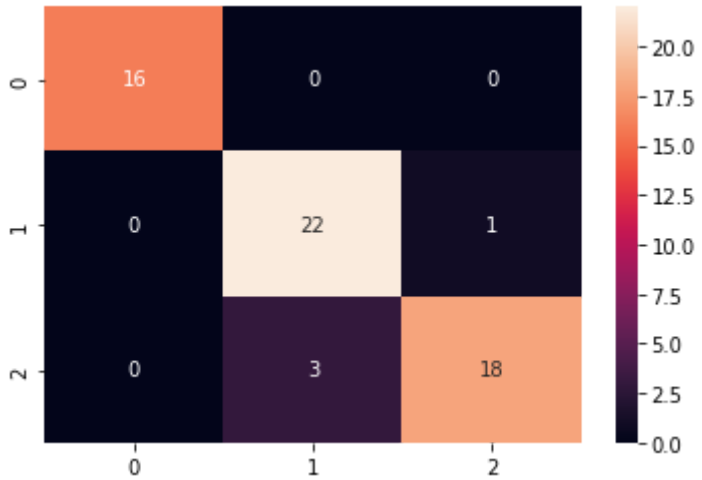
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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:
 [[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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18efba90>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

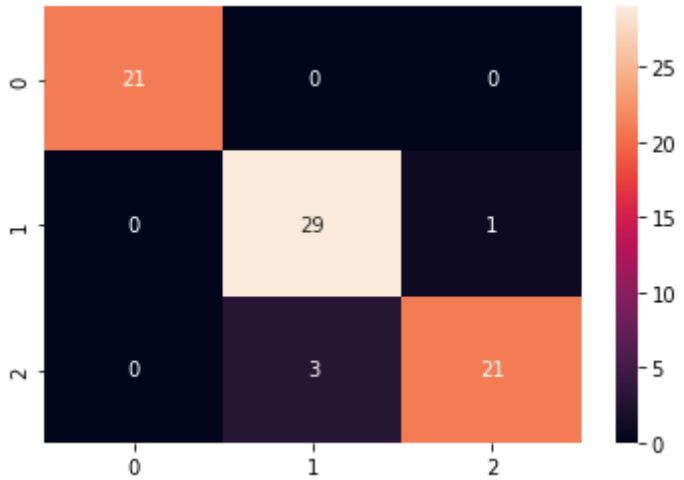
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18e43e50>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
60
90
```

In [ ]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

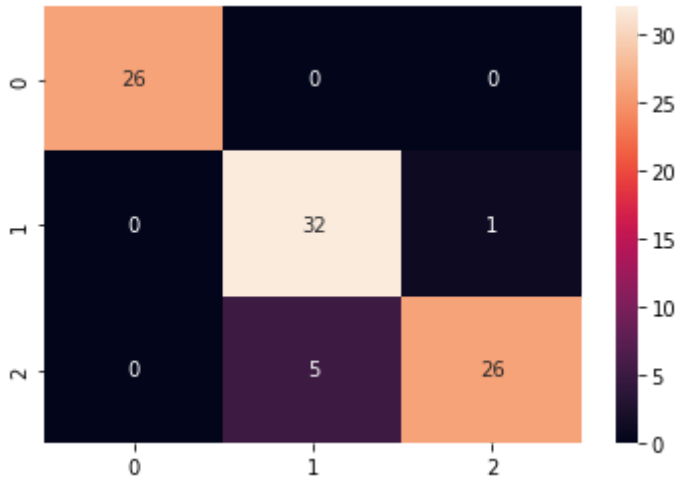
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1929f510>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

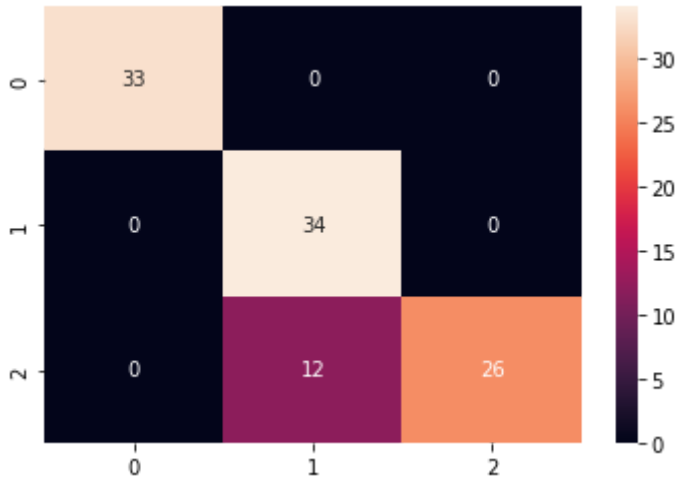
Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18d13e50>
```



Sigmoid SVC Classifier

In [ ]:
```
sigmoid_SVC_classifier = SVC(kernel='sigmoid')
sigmoid_SVC_classifier
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

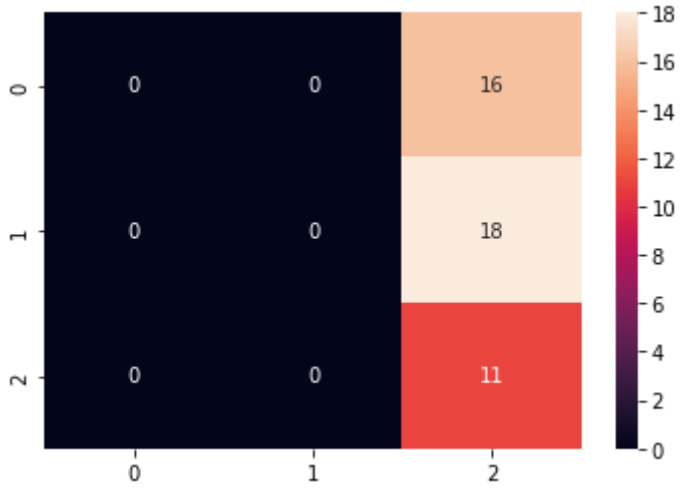
In [ ]:
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18bfce50>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

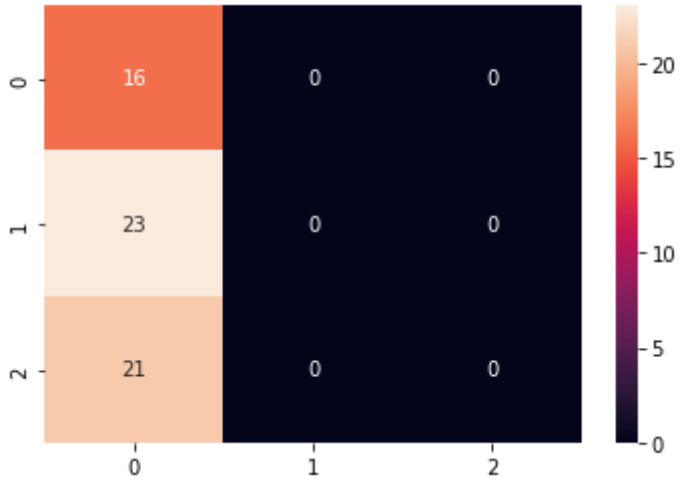
In [ ]:
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f240a5590>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

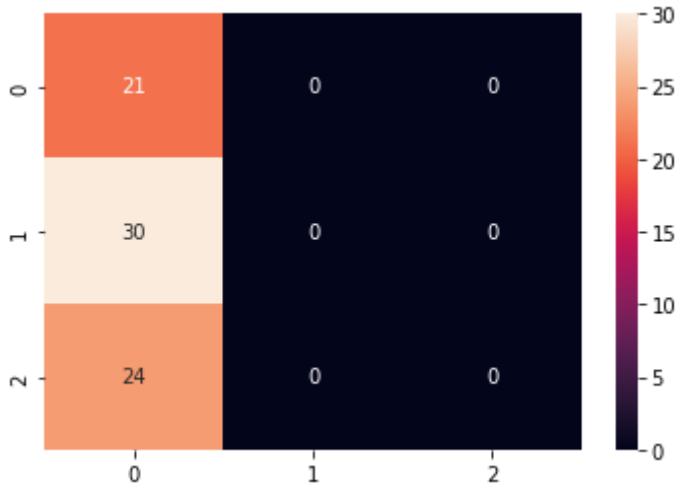
In [ ]:
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18a70c90>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
60
90
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

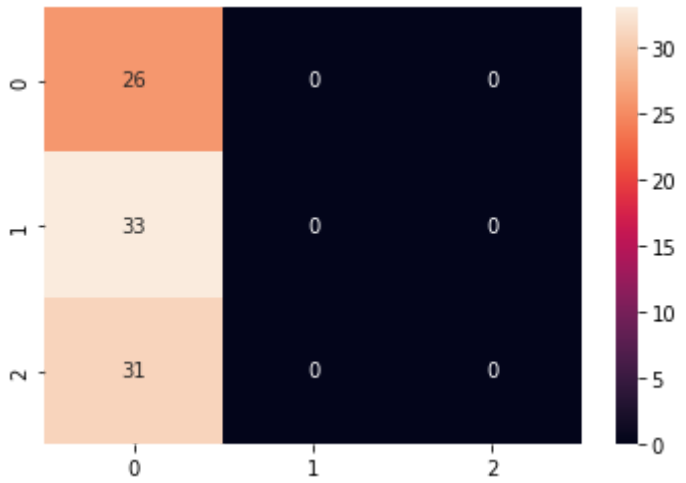
In [ ]:
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f189b4190>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

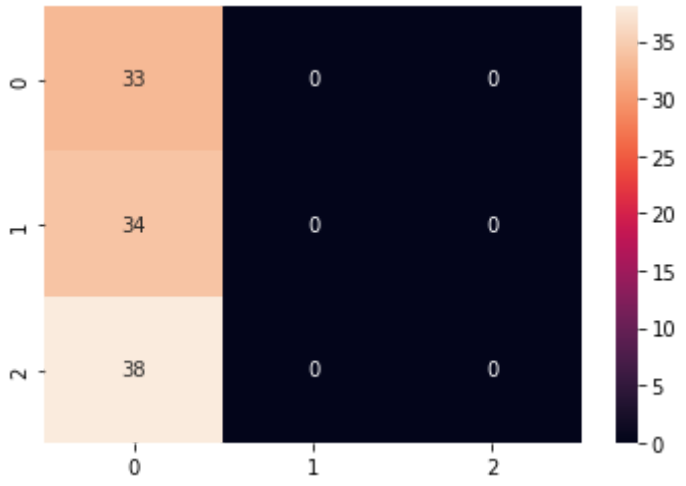
In [ ]:
```
y_pred = sigmoid_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n", cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f188f7390>
```



MLP Classifier

In [ ]:
```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
```

```
mlp_classifier
```

Out[ ]:
```
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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

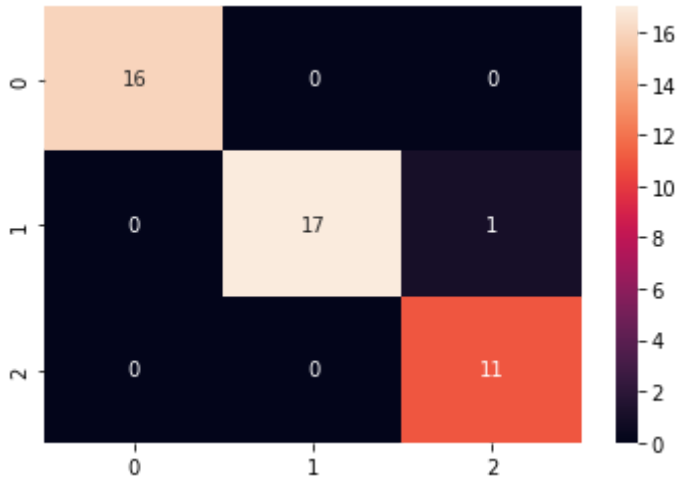
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18839510>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

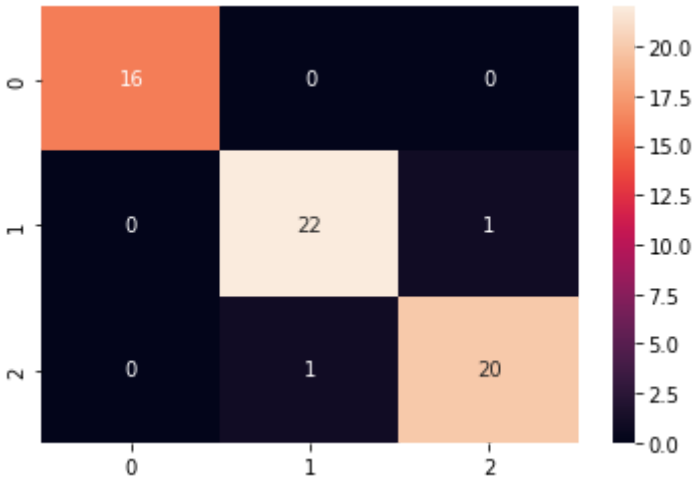
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1876f890>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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)
```

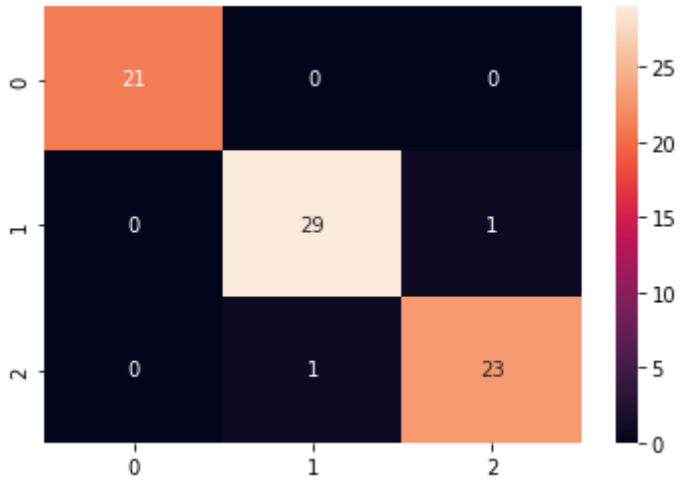
In [ ]:
```
y_pred = mlp_classifier.predict(X_test)
```

```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f186af650>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
60
90
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

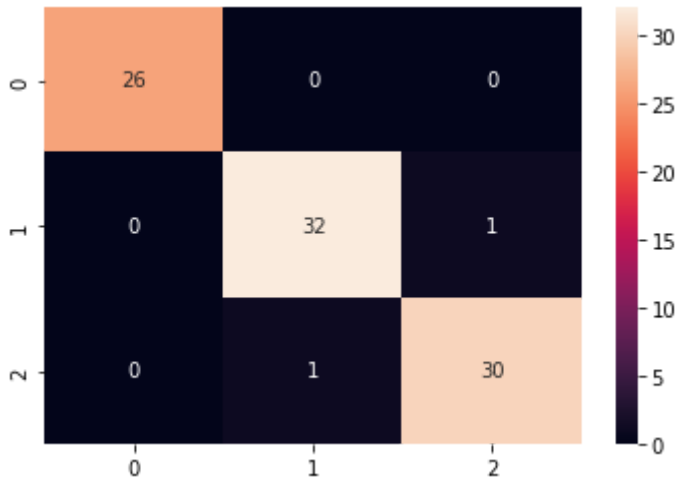
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f240a50d0>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

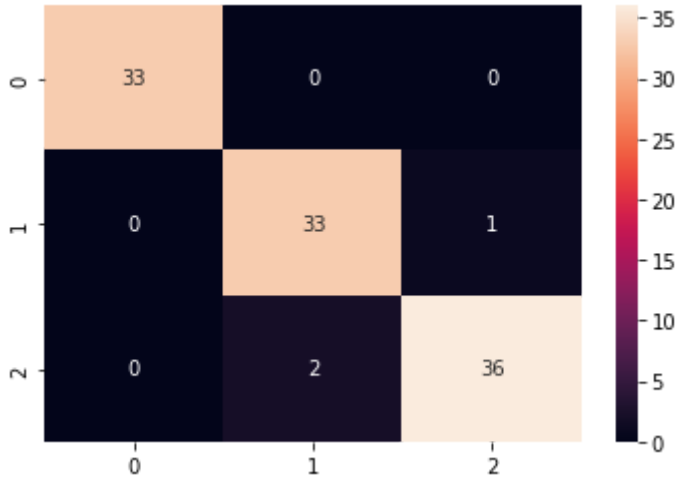
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18b7a050>
```



Random Forest Classifier

In [ ]:
```
rfc_classifier = RandomForestClassifier(n_estimators=20)
rfc_classifier
```

Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
105
45
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

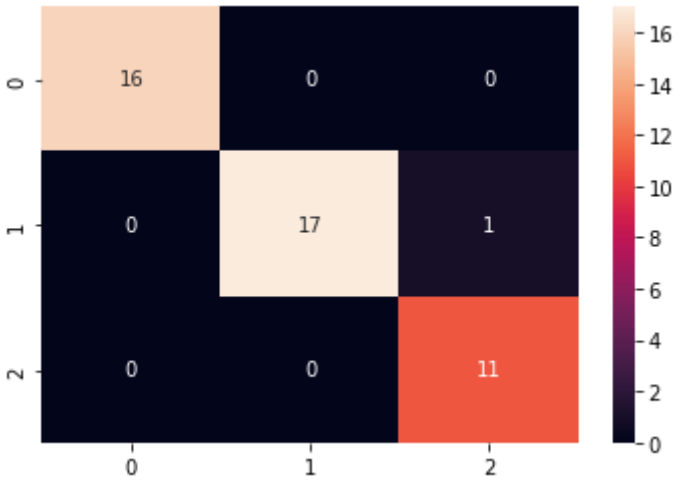
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f184b8750>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
90
60
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

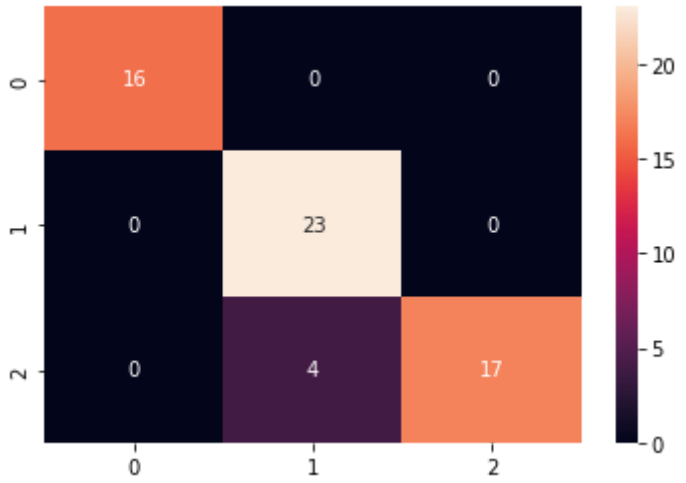
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f183f9850>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
75
75
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

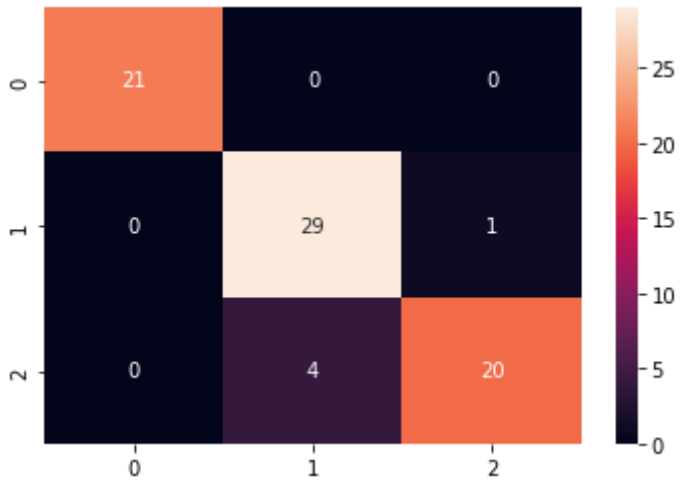
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1832e890>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
```

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

```
60
90
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

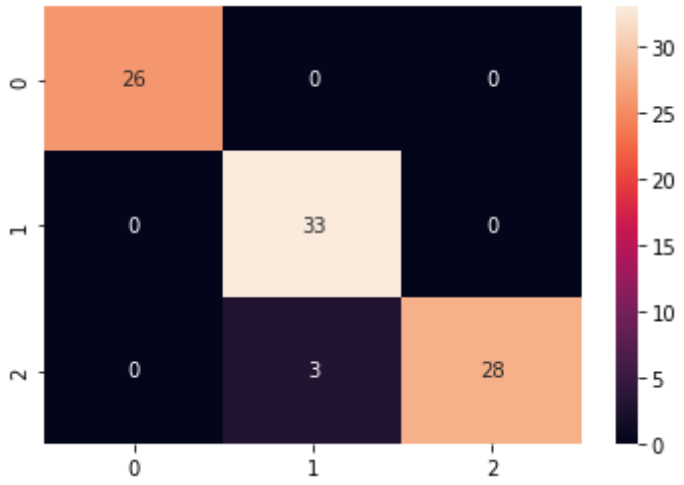
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f18263650>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
45
105
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

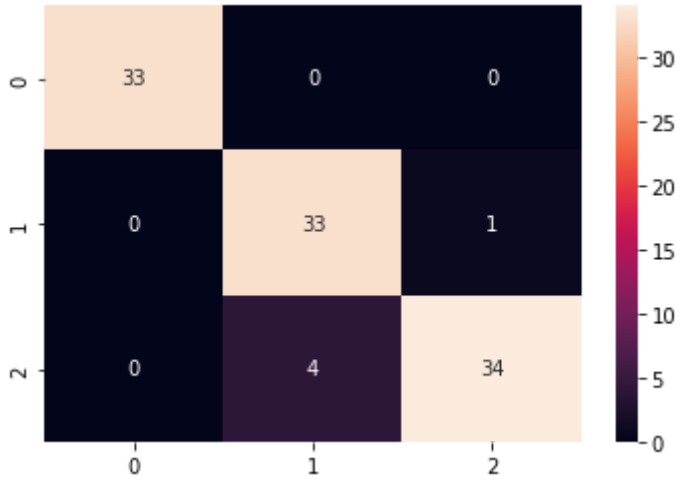
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f181a1890>
```


0
1
2
33
0
0
0
33
1
0
4
34
0
1
2
30
25
20
15
10
5
0

Swapnil Ghosh JU IT 001911001067

Import required modules

In [1]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.metrics import accuracy_score, classification_report, confusio
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

In [2]:
```
df = pd.read_csv('/content/ionosphere_data.csv')
df.head()
```

Out[2]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	colu
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.(
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.(
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.8
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.(
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.1

In [3]:
```
df.column_ai.value_counts()
```

Out[3]:
```
g    225
b    126
Name: column_ai, dtype: int64
```

DataFrame ready to perform

In [4]:
```
len(df)
```

Out[4]:
```
351
```

In [5]:
```
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

In [6]:
```
linear_SVC_classifier = SVC(kernel='linear')
linear_SVC_classifier
```

Out[6]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [8]:
```
print(len(X_train))
print(len(y_test))
```

```
245
106
```

In [9]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

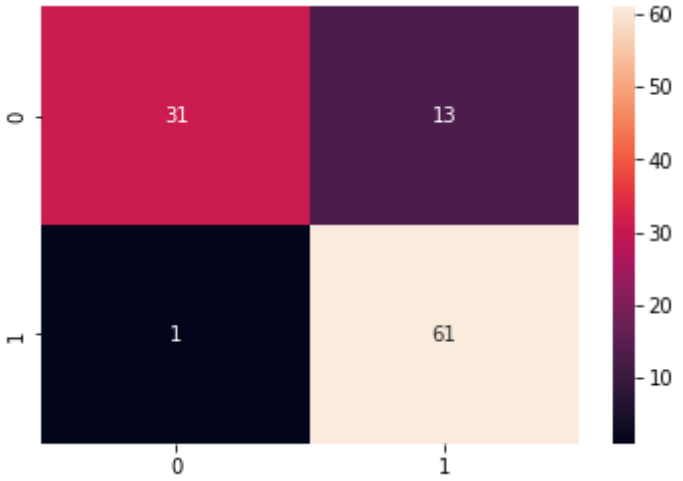
Out[9]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [10]:
```
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
```

In [11]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fd4a58650>
```



train size : test size = 60% : 40%

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [13]:
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [14]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

Out[14]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

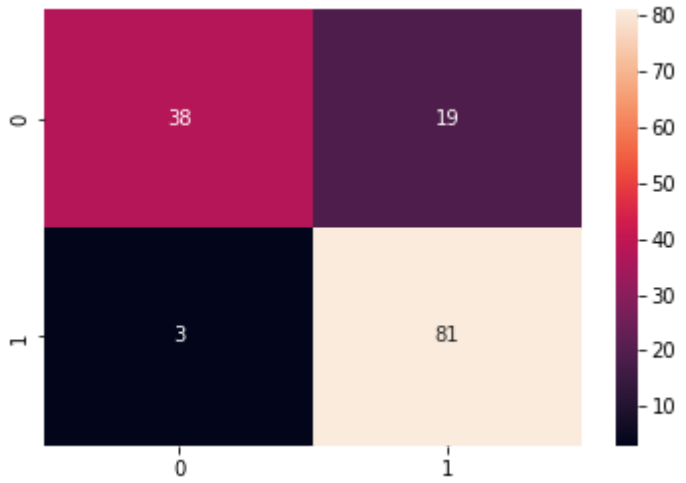
In [15]:
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
```

```
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
```

In [16]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fca484750>
```



train size : test size = 50% : 50%

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [18]:
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [19]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

Out[19]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

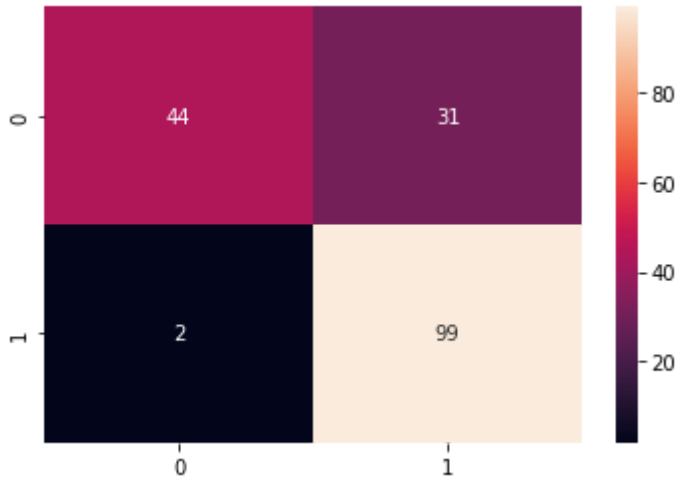
In [20]:
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
```

```
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
```

In [21]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9f5da90>
```



train size : test size = 40% : 60%

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [23]:
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [24]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

Out[24]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

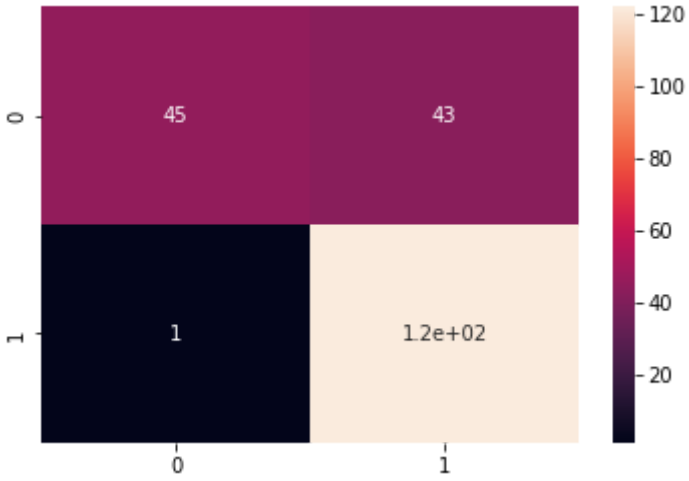
In [25]:
```
y_pred = linear_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
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
```

In [26]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9e95450>
```



train size : test size = 30% : 70%

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [28]:
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [29]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

Out[29]:
```
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='linea
r',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

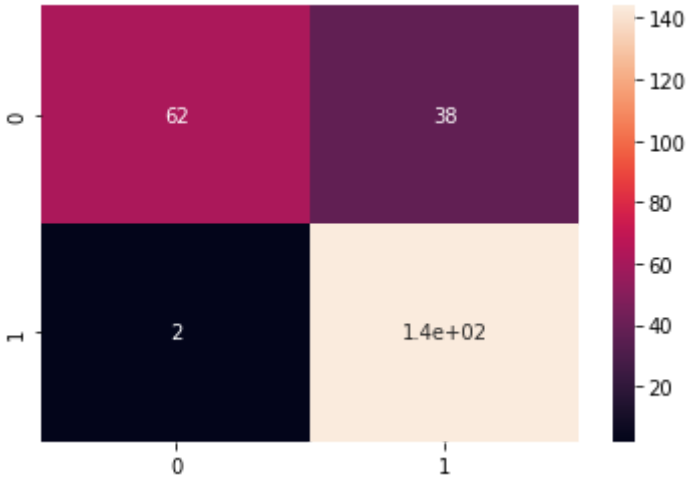
In [30]:
```
y_pred = linear_SVC_classifier.predict(X_test)
```

```
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
```

In [31]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9f65a50>
```



Polynomial SVC Classifier

In [32]:
```
poly_SVC_classifier = SVC(kernel='poly')
poly_SVC_classifier
```

Out[32]:
```
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%

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [34]:
```
print(len(X_train))
```

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

```
245
106
```

In [35]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

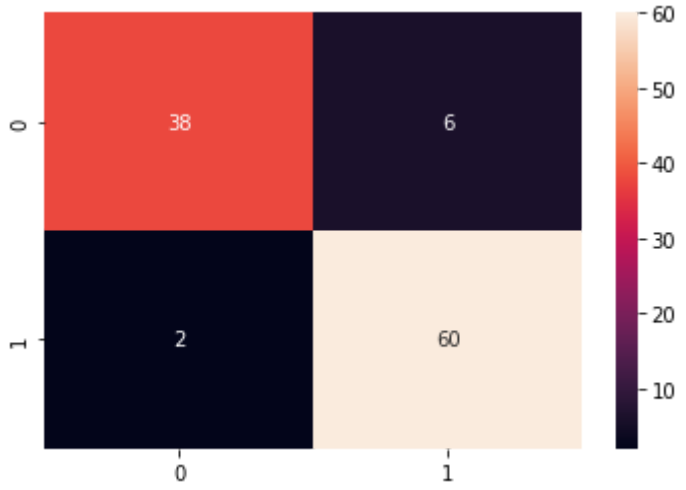
Out[35]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [36]:
```
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
```

In [37]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9d7b0d0>
```



train size : test size = 60% : 40%

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [39]:
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [40]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

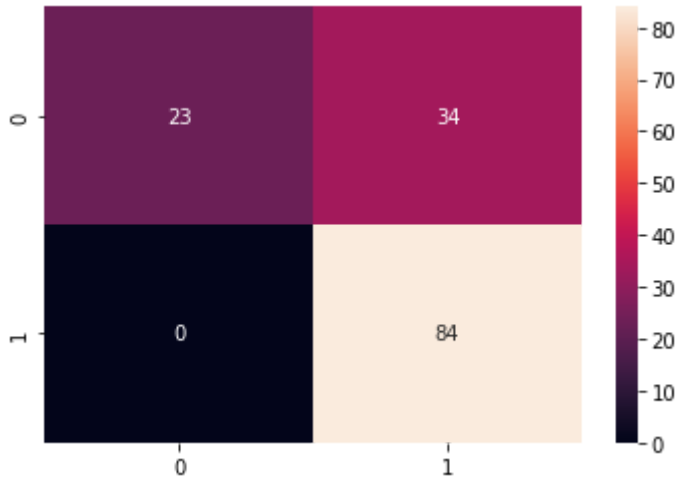
Out[40]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [41]:
```
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
```

In [42]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9d0a3d0>
```



train size : test size = 50% : 50%

In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [44]:
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [45]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

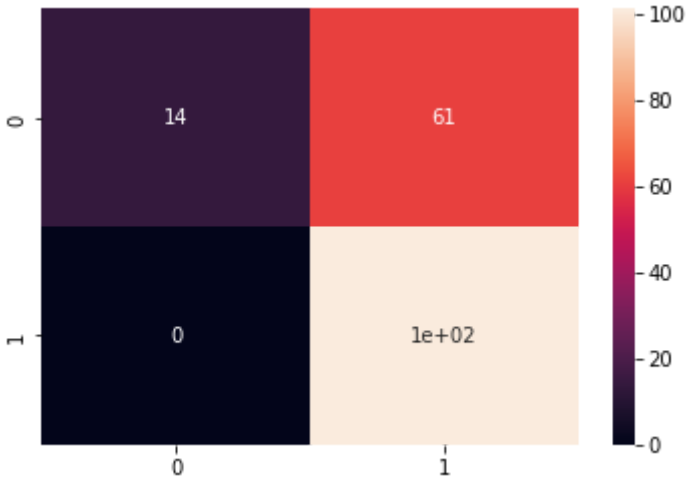
Out[45]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [46]:
```
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
```

In [47]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9c31bd0>
```



train size : test size = 40% : 60%

In [48]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [49]:
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [50]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

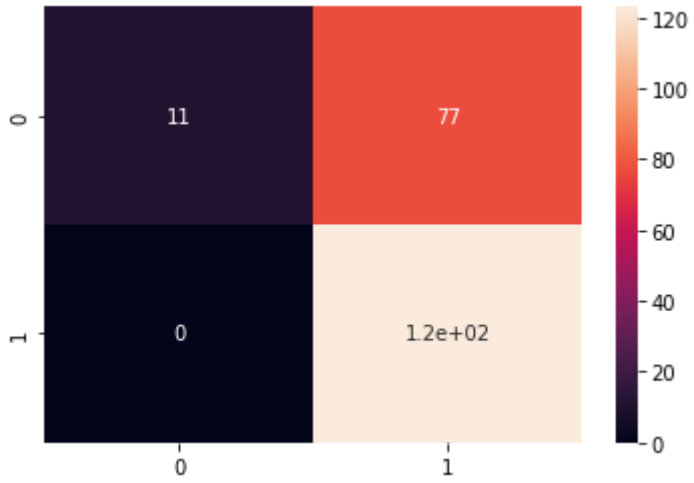
Out[50]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [51]:
```
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
```

In [52]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9c8f610>
```



train size : test size = 30% : 70%

In [53]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [54]:
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [55]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

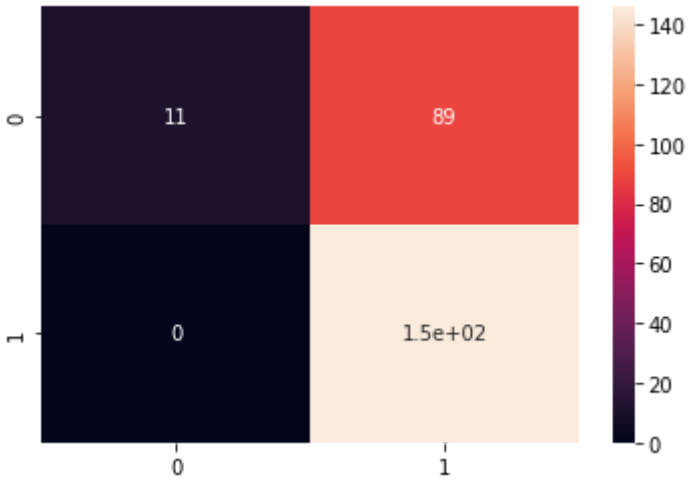
Out[55]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [56]:
```
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
```

In [57]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9b11050>
```



Gaussain SVC Classifier

In [58]:
```
gaussain_SVC_classifier = SVC(kernel='rbf')
gaussain_SVC_classifier
```

Out[58]:
```
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%

In [59]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [60]:
```
print(len(X_train))
print(len(y_test))
```

```
245
106
```

In [61]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

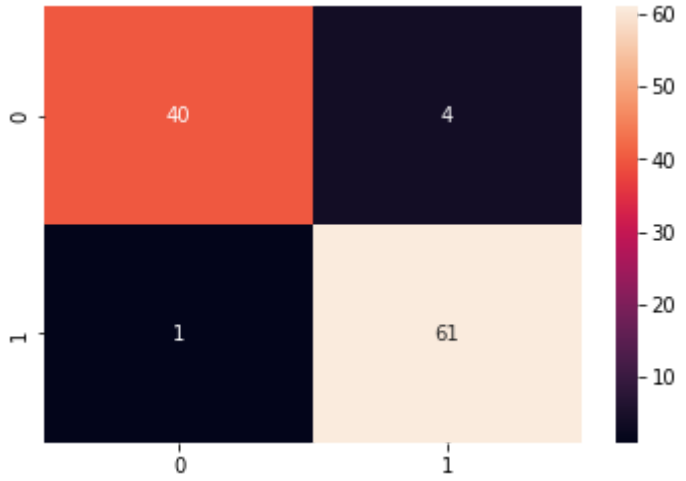
Out[61]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [62]:
```
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
```

In [63]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9a3d310>
```



train size : test size = 60% : 40%

In [64]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [65]:
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [66]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

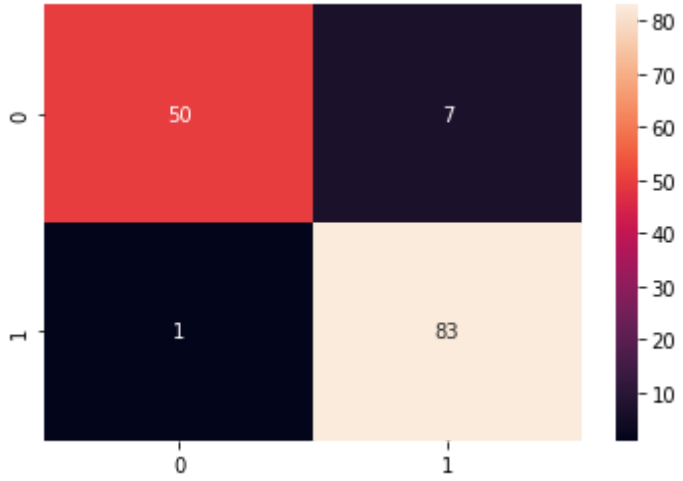
Out[66]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [67]:
```
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
```

In [68]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc995f510>
```



train size : test size = 50% : 50%

In [69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [70]:
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [71]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

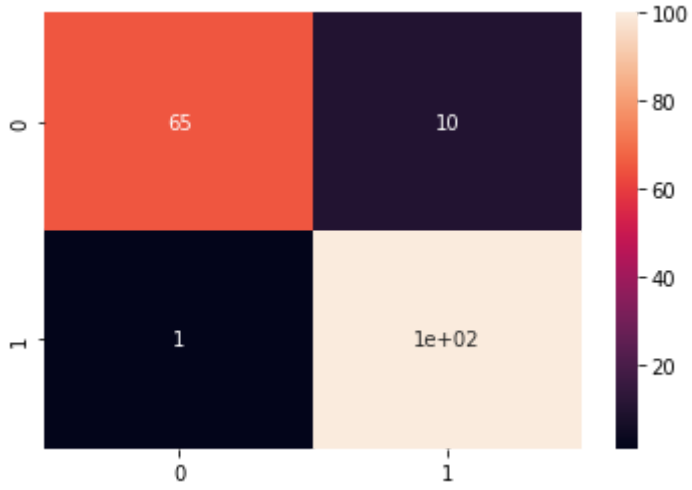
Out[71]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [72]:
```
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
```

In [73]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[73]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9908110>
```



train size : test size = 40% : 60%

In [74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [75]:
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [76]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

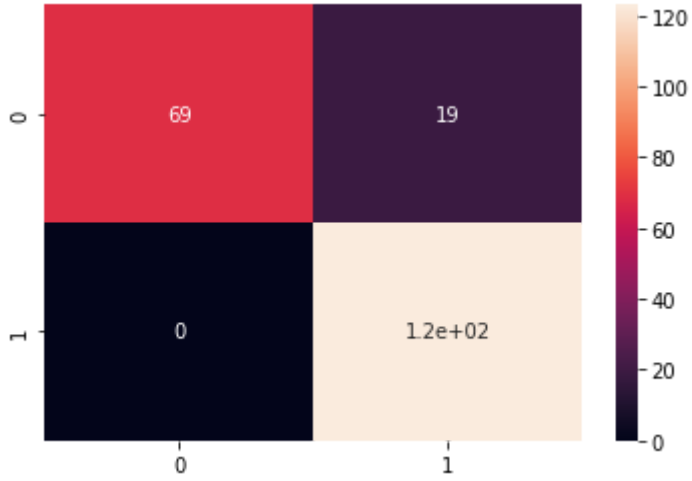
Out[76]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [77]:
```
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
```

In [78]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[78]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9cea990>
```



train size : test size = 30% : 70%

In [79]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [80]:
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [81]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

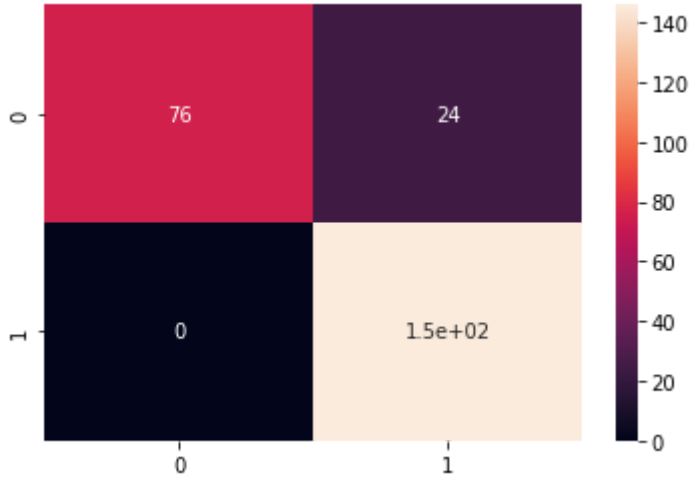
Out[81]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [82]:
```
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
```

In [83]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[83]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc97ad810>
```



Sigmoid SVC Classifier

In [84]:
```
sigmoid_SVC_classifier = SVC(kernel='sigmoid')
sigmoid_SVC_classifier
```

Out[84]:
```
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='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

train size : test size = 70% : 30%

In [85]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [86]:
```
print(len(X_train))
print(len(y_test))
```

```
245
106
```

In [87]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

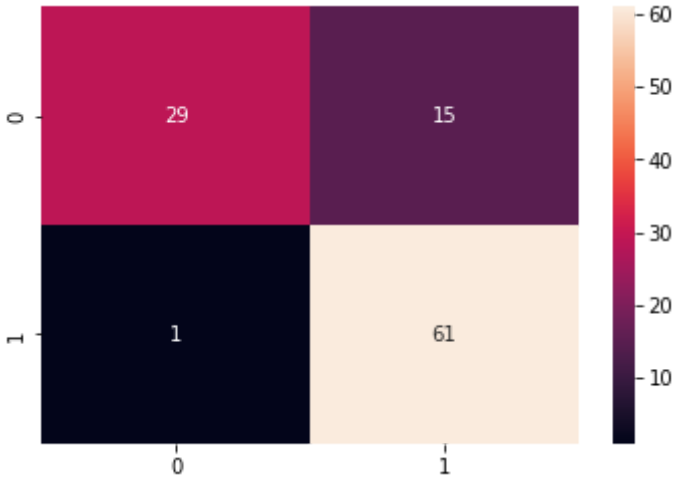
Out[87]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [88]:
```
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
```

In [89]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[89]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc96dcdd0>
```



train size : test size = 60% : 40%

In [90]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [91]:
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [92]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

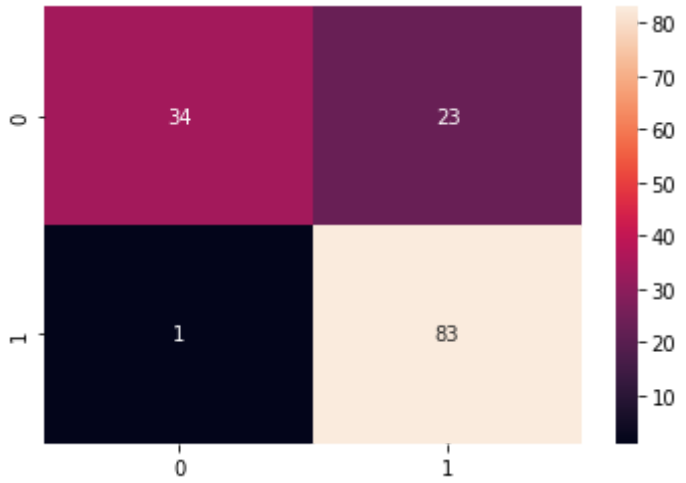
Out[92]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [93]:
```
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
```

In [94]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[94]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9673850>
```



train size : test size = 50% : 50%

In [95]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [96]:
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [97]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

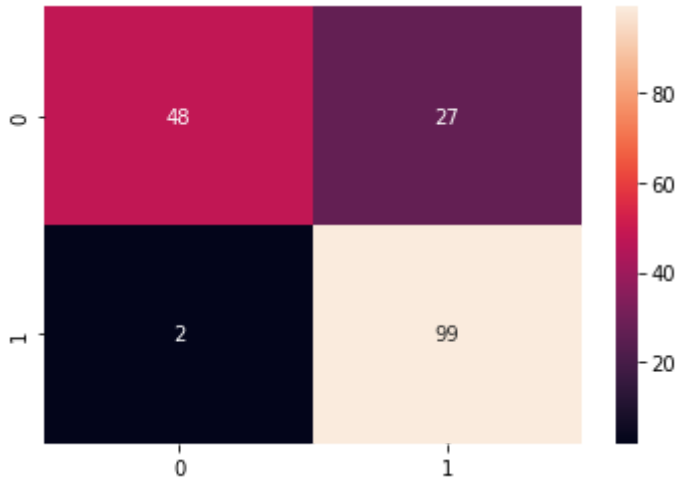
Out[97]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [98]:
```
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
```

In [99]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[99]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc95aa290>
```



train size : test size = 40% : 60%

In [100…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [101…
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [102…
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

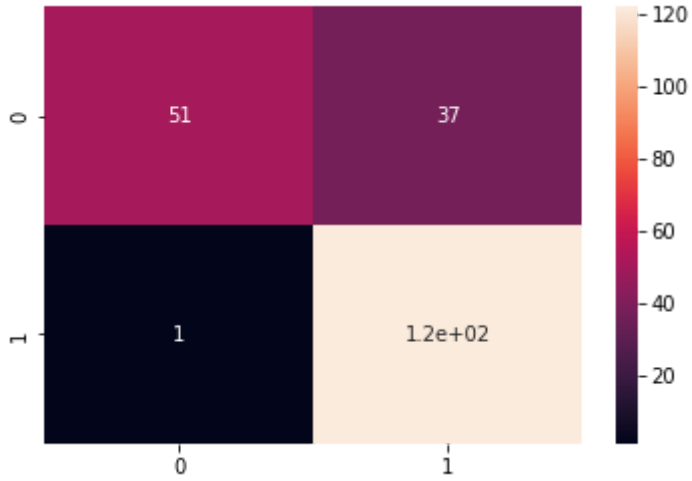
Out[102]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [103…
```
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
```

In [104…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[104]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc94da4d0>
```



train size : test size = 30% : 70%

In [105…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [106…
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [107…
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

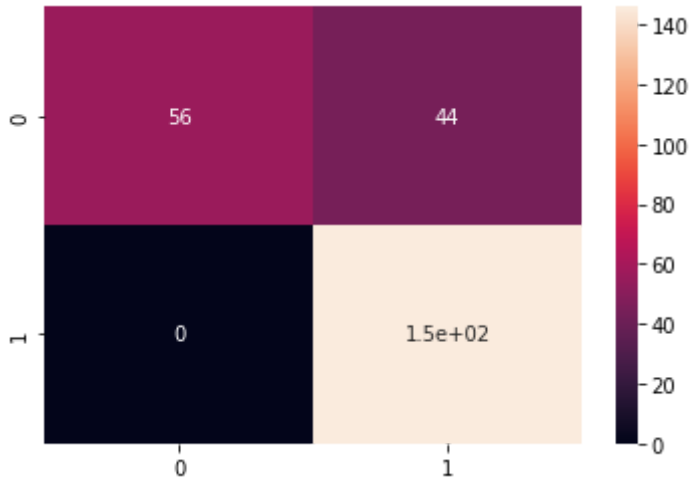
Out[107]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [108…
```
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
```

In [109…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[109]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9476dd0>
```



MLP Classifier

In [110…
```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
mlp_classifier
```

Out[110]:
```
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%

In [111…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [112…
```
print(len(X_train))
print(len(y_test))
```

```
245
106
```

In [113…
```
mlp_classifier.fit(X_train, y_train)
```

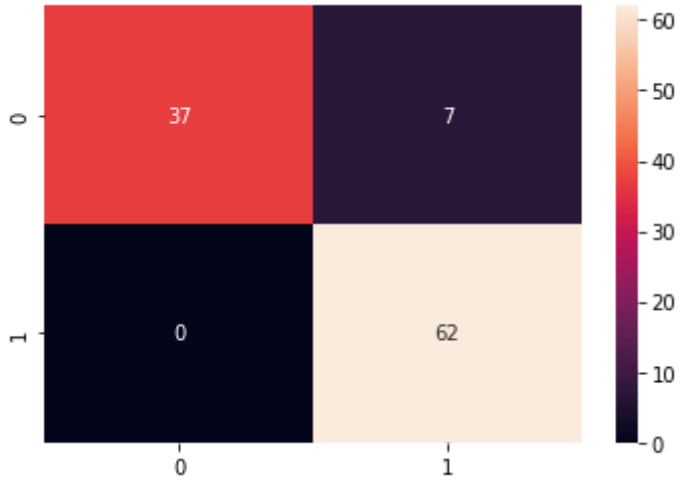
Out[113]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [114…
```
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
```

In [115…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[115]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc93b3250>
```



train size : test size = 60% : 40%

In [116…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [117…
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [118…
```
mlp_classifier.fit(X_train, y_train)
```

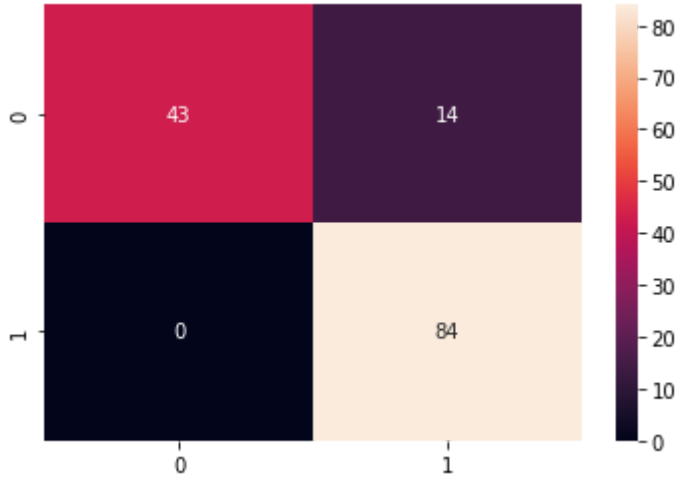
Out[118]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [119…
```
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
```

In [120…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[120]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9340950>
```



train size : test size = 50% : 50%

In [121…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [122…
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [123…
```
mlp_classifier.fit(X_train, y_train)
```

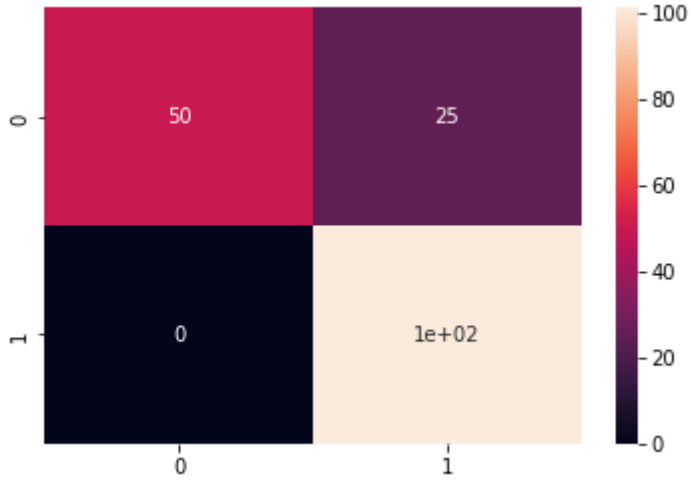
Out[123]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [124…
```
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
```

In [125…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[125]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9277f50>
```



train size : test size = 40% : 60%

In [126…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [127…
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [128…
```
mlp_classifier.fit(X_train, y_train)
```

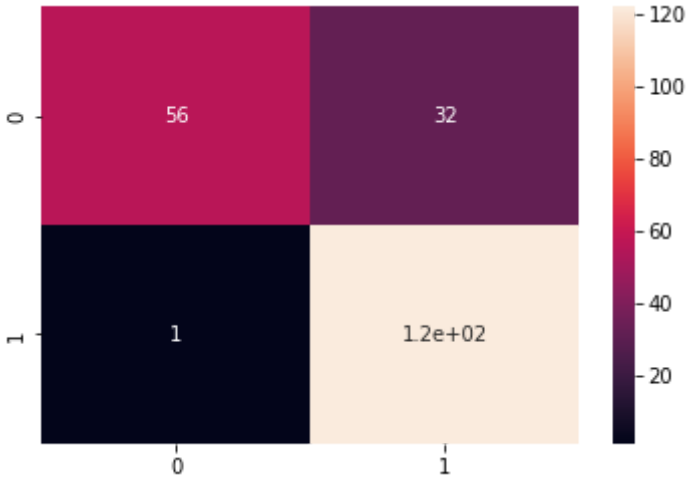
Out[128]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [129…
```
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
```

In [130…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[130]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc91ac8d0>
```



train size : test size = 30% : 70%

In [131…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [132…
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [133…
```
mlp_classifier.fit(X_train, y_train)
```

Out[133]:
```
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)
```

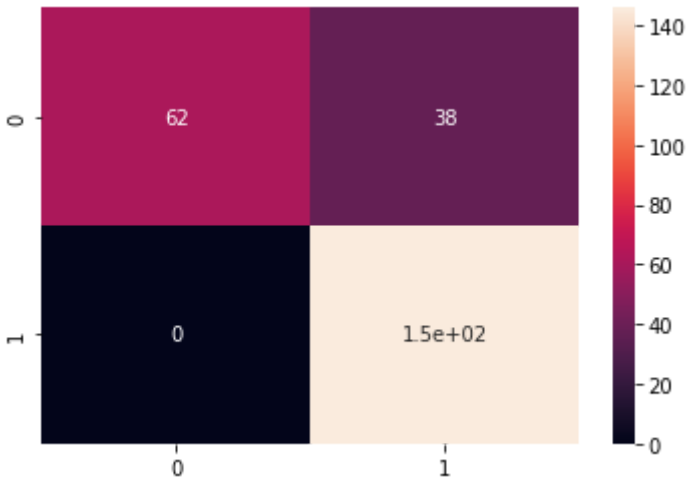
In [134…
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
```

```
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
```

In [135…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[135]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc90e4a50>
```



Random Forest Classifier

In [136…
```
rfc_classifier = RandomForestClassifier(n_estimators=20)
rfc_classifier
```

Out[136]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       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%

In [137…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [138…
```
print(len(X_train))
print(len(y_test))
```

```
245
106
```

In [139…
```
rfc_classifier.fit(X_train, y_train)
```

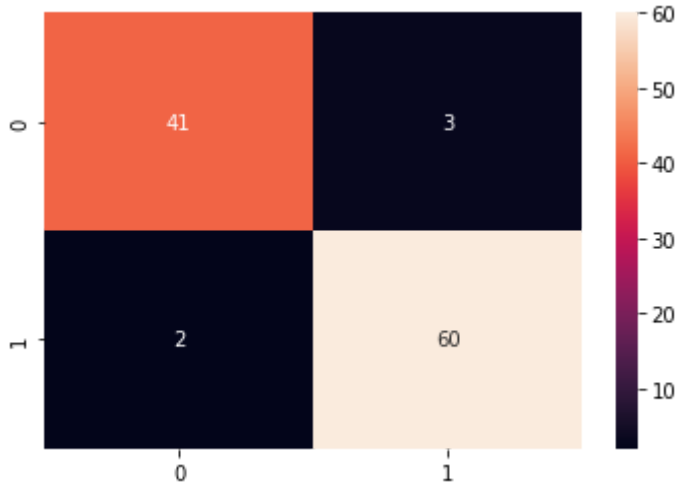
Out[139]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [140…
```
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
```

In [141…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[141]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc95aa750>
```



train size : test size = 60% : 40%

In [142…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [143…
```
print(len(X_train))
print(len(y_test))
```

```
210
141
```

In [144…
```
rfc_classifier.fit(X_train, y_train)
```

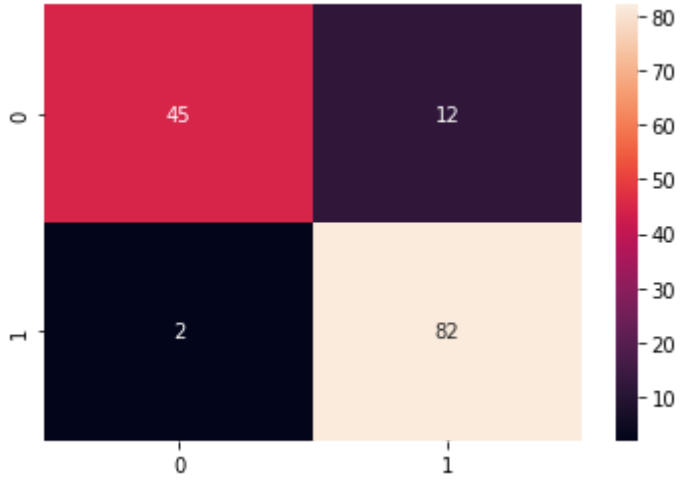
Out[144]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [145…
```
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
```

In [146…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[146]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8f97410>
```



train size : test size = 50% : 50%

In [147…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [148…
```
print(len(X_train))
print(len(y_test))
```

```
175
176
```

In [149…
```
rfc_classifier.fit(X_train, y_train)
```

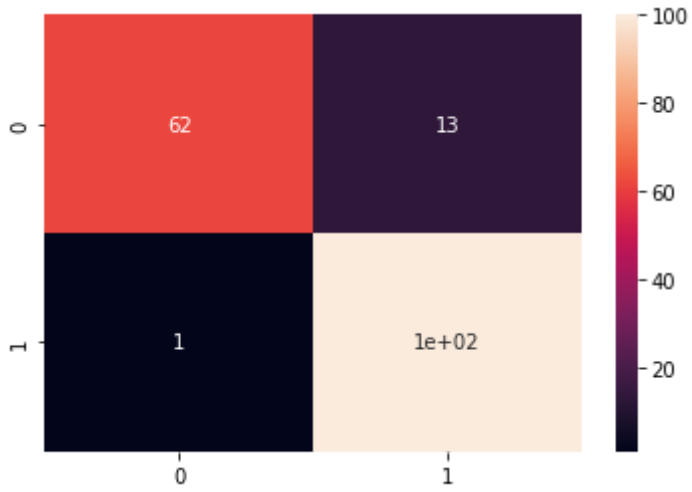
Out[149]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [150…
```
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
```

In [151…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[151]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8f42090>
```



train size : test size = 40% : 60%

In [152…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [153…
```
print(len(X_train))
print(len(y_test))
```

```
140
211
```

In [154…
```
rfc_classifier.fit(X_train, y_train)
```

Out[154]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       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)
```

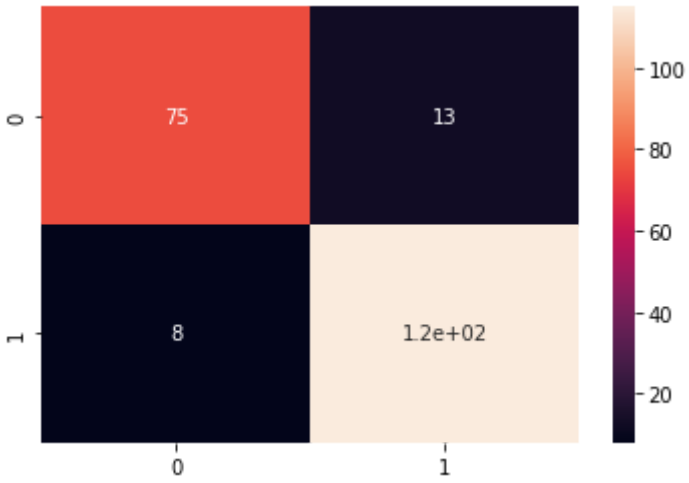
In [155…
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
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
```

In [156…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[156]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8e6b350>
```



train size : test size = 30% : 70%

In [157…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [158…
```
print(len(X_train))
print(len(y_test))
```

```
105
246
```

In [159…
```
rfc_classifier.fit(X_train, y_train)
```

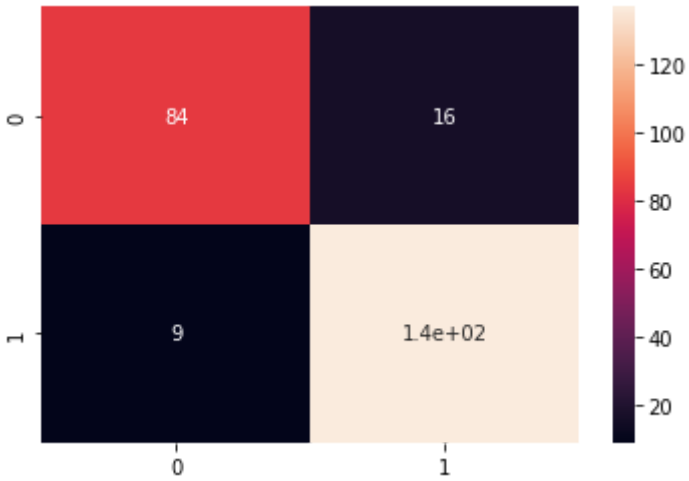
Out[159]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [160…
```
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
```

In [161…
```
sns.heatmap(cf_matrix, annot=True)
```

Out[161]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc8df4810>
```



Swapnil Ghosh JU IT 001911001067

Import required modules

In [3]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.metrics import accuracy_score, classification_report, confusio
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
```

In [2]:
```
!pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
     |████████████████████████████████| 292 kB 1.5 MB/s eta 0:00:01
Requirement already satisfied: scipy>=1.0 in /home/swapnil/Desktop/sample_p
roject_1/env/lib/python3.9/site-packages (from seaborn) (1.7.3)
Requirement already satisfied: pandas>=0.23 in /home/swapnil/Desktop/sample
_project_1/env/lib/python3.9/site-packages (from seaborn) (1.4.2)
Requirement already satisfied: numpy>=1.15 in /home/swapnil/Desktop/sample_
project_1/env/lib/python3.9/site-packages (from seaborn) (1.22.3)
Requirement already satisfied: matplotlib>=2.2 in /home/swapnil/Desktop/sam
ple_project_1/env/lib/python3.9/site-packages (from seaborn) (3.5.1)
Requirement already satisfied: cycler>=0.10 in /home/swapnil/Desktop/sample
_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn)
(0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /home/swapnil/Desktop/sampl
e_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn)
(9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in /home/swapnil/Deskto
p/sample_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->s
eaborn) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/swapnil/Desktop/s
ample_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seab
orn) (1.4.2)
Requirement already satisfied: fonttools>=4.22.0 in /home/swapnil/Desktop/s
ample_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seab
orn) (4.25.0)
Requirement already satisfied: packaging>=20.0 in /home/swapnil/Desktop/sam
ple_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seabor
n) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in /home/swapnil/Desktop/sa
mple_project_1/env/lib/python3.9/site-packages (from matplotlib>=2.2->seabo
rn) (3.0.4)
Requirement already satisfied: pytz>=2020.1 in /home/swapnil/Desktop/sample
_project_1/env/lib/python3.9/site-packages (from pandas>=0.23->seaborn) (20
22.1)
Requirement already satisfied: six>=1.5 in /home/swapnil/Desktop/sample_pro
ject_1/env/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotl
ib>=2.2->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.11.2
```

Load Dataset

In [4]:
```
b_cancer = datasets.load_breast_cancer() # it's source is same as : https:/
```

In [5]:
```
dir(b_cancer)
```

Out[5]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [6]:
```
b_cancer.data
```

Out[6]:
```
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]])
```

In [7]:
```
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 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 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 0 0 1 0 1 1 0 0 1 1 0 0 1 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 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 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 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 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 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 0 0 0 0 0 0 1]
```

In [8]:
```
df = pd.DataFrame(data=b_cancer.data, columns=b_cancer.feature_names)
```

```
df.head()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 30 columns

In [9]:
```
df["target"] = b_cancer.target
df.head()
```

Out[9]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 31 columns

In [10]:
```
df.target.value_counts()
```

Out[10]:
```
1    357
0    212
Name: target, dtype: int64
```

DataFrame ready to perform

In [11]:
```
len(df)
```

Out[11]:
```
569
```

In [12]:
```
X = df.drop(["target"], axis="columns")
y = df.target
print(X.head())
print(y.head())
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst radius  worst texture  worst perimete
r  \
0                 0.07871  ...         25.38          17.33           184.6
0
1                 0.05667  ...         24.99          23.41           158.8
0
2                 0.05999  ...         23.57          25.53           152.5
0
3                 0.09744  ...         14.91          26.50            98.8
7
4                 0.05883  ...         22.54          16.67           152.2
0

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          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

In [13]:
```
linear_SVC_classifier = SVC(kernel='linear')
linear_SVC_classifier
```

Out[13]:
```
SVC(kernel='linear')
```

train size : test size = 70% : 30%

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [15]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [16]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

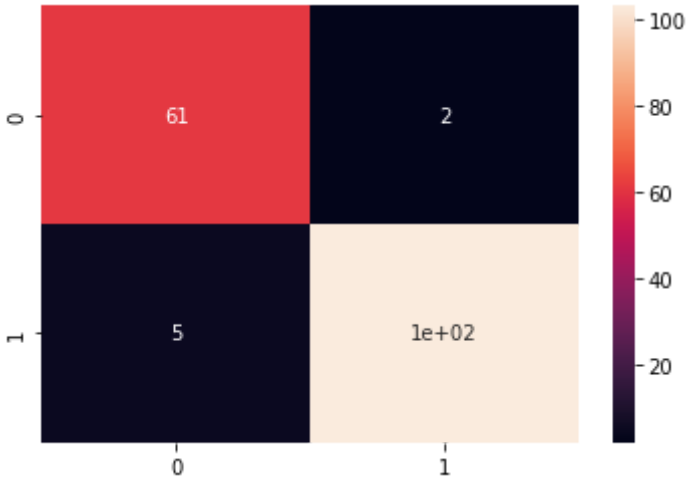
Out[16]:
```
SVC(kernel='linear')
```

In [17]:
```
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
```

In [18]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[18]:
```
<AxesSubplot:>
```



train size : test size = 60% : 40%

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [20]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [21]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

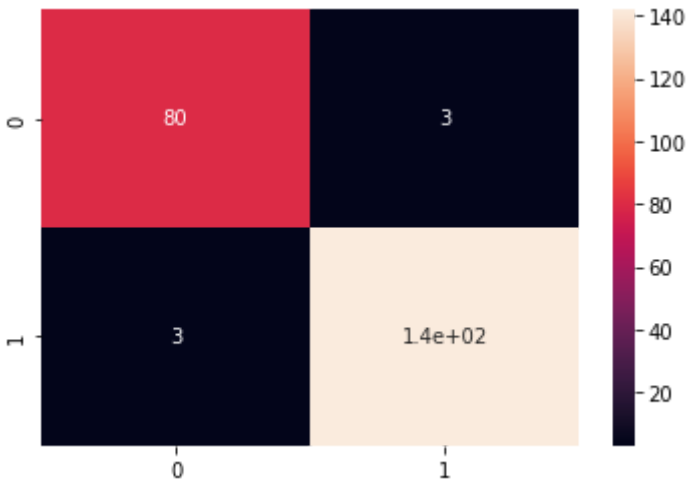
Out[21]:
```
SVC(kernel='linear')
```

In [22]:
```
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
```

In [23]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[23]:
```
<AxesSubplot:>
```



train size : test size = 50% : 50%

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [25]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [26]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

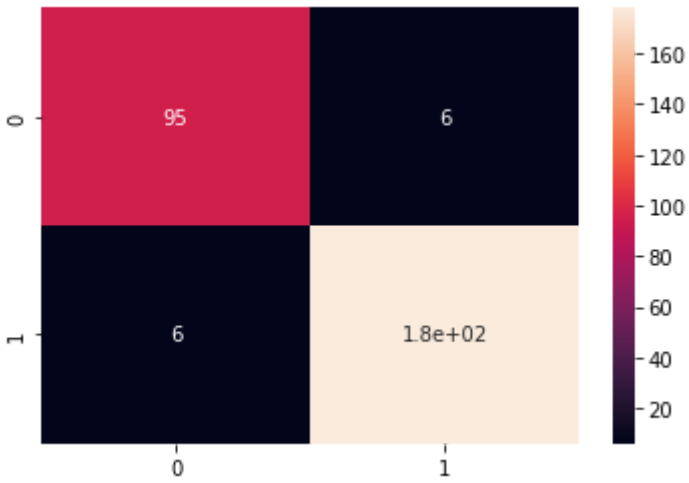
Out[26]:
```
SVC(kernel='linear')
```

In [27]:
```
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
```

In [28]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[28]:
```
<AxesSubplot:>
```



train size : test size = 40% : 60%

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [30]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [31]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

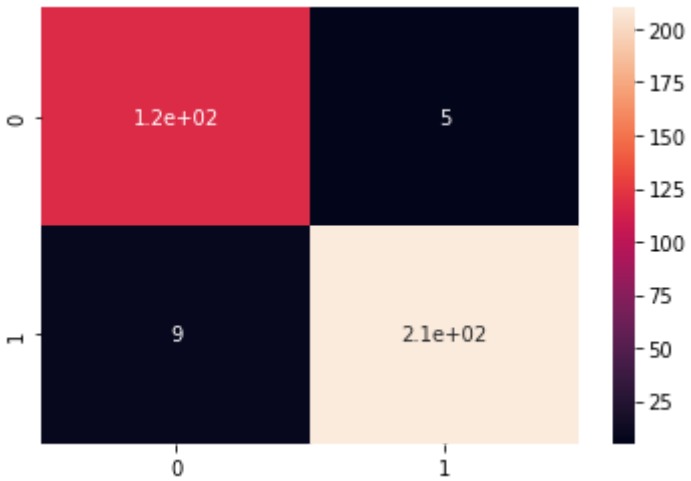
Out[31]:
```
SVC(kernel='linear')
```

In [32]:
```
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
```

In [33]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[33]:
```
<AxesSubplot:>
```



train size : test size = 30% : 70%

In [34]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [35]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [36]:
```
linear_SVC_classifier.fit(X_train, y_train)
```

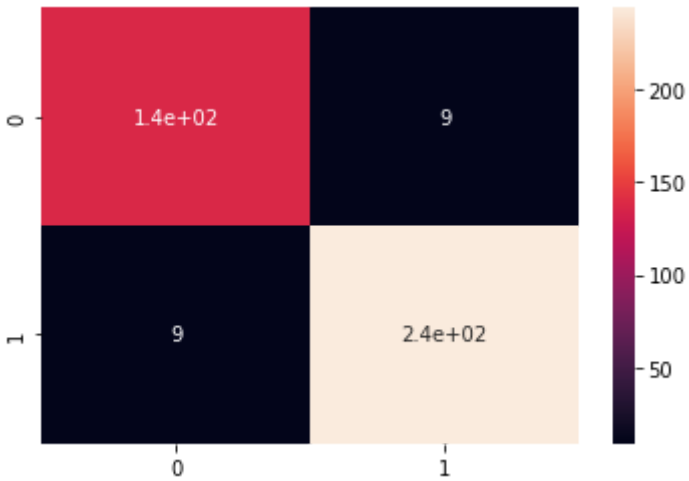
Out[36]:
```
SVC(kernel='linear')
```

In [37]:
```
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: 95.48872180451127%

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
```

In [38]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[38]:
```
<AxesSubplot:>
```



Polynomial SVC Classifier

In [39]:
```
poly_SVC_classifier = SVC(kernel='poly')
poly_SVC_classifier
```

Out[39]:
```
SVC(kernel='poly')
```

train size : test size = 70% : 30%

In [40]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [41]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [42]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

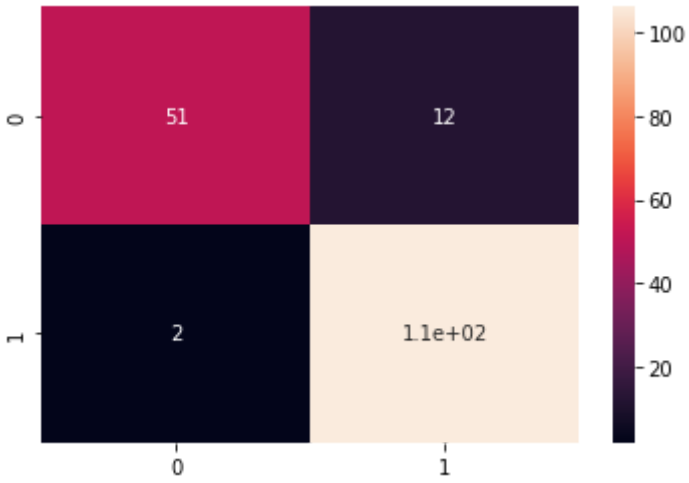
Out[42]:
```
SVC(kernel='poly')
```

In [43]:
```
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
```

In [44]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[44]:
```
<AxesSubplot:>
```



train size : test size = 60% : 40%

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [46]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [47]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

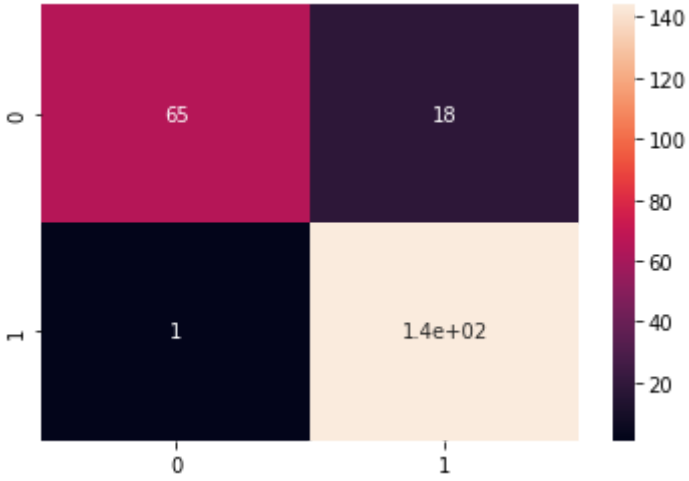
Out[47]:
```
SVC(kernel='poly')
```

In [48]:
```
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
```

In [49]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[49]:
```
<AxesSubplot:>
```



train size : test size = 50% : 50%

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [51]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [52]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

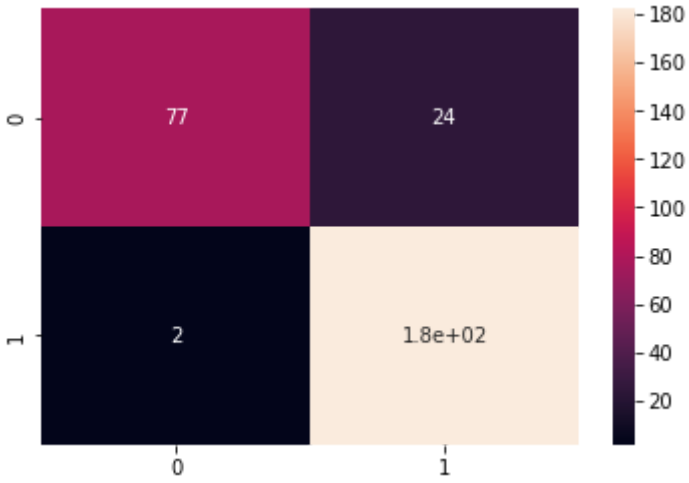
Out[52]:
```
SVC(kernel='poly')
```

In [53]:
```
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
```

In [54]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[54]:
```
<AxesSubplot:>
```



train size : test size = 40% : 60%

In [55]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [56]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [57]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

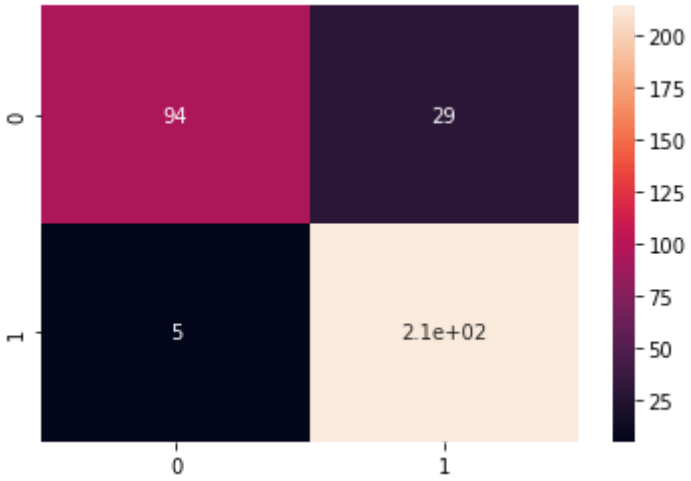
Out[57]:
```
SVC(kernel='poly')
```

In [58]:
```
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
```

In [59]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[59]:
```
<AxesSubplot:>
```



train size : test size = 30% : 70%

In [60]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [61]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [62]:
```
poly_SVC_classifier.fit(X_train, y_train)
```

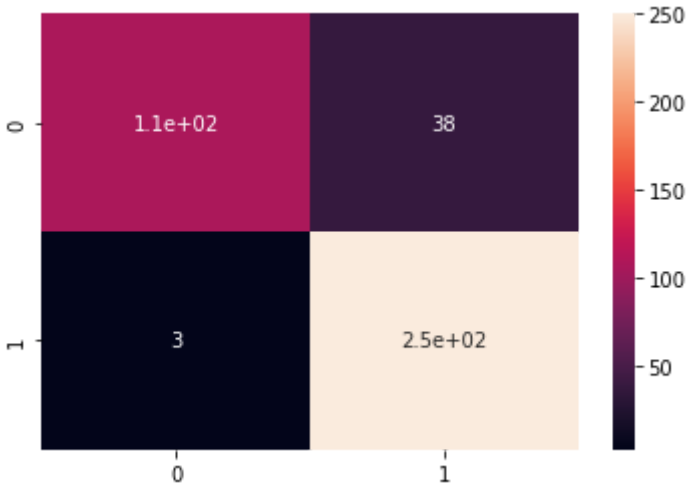
Out[62]:
```
SVC(kernel='poly')
```

In [63]:
```
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
```

In [64]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[64]:
```
<AxesSubplot:>
```



Gaussain SVC Classifier

In [65]:
```
gaussain_SVC_classifier = SVC(kernel='rbf')
gaussain_SVC_classifier
```

Out[65]:
```
SVC()
```

train size : test size = 70% : 30%

In [66]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [67]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [68]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

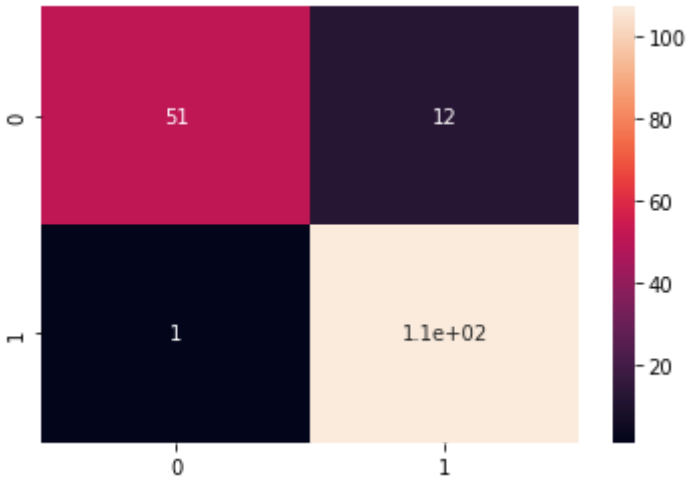
Out[68]:
```
SVC()
```

In [69]:
```
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
```

In [70]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[70]:
```
<AxesSubplot:>
```



train size : test size = 60% : 40%

In [71]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [72]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [73]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

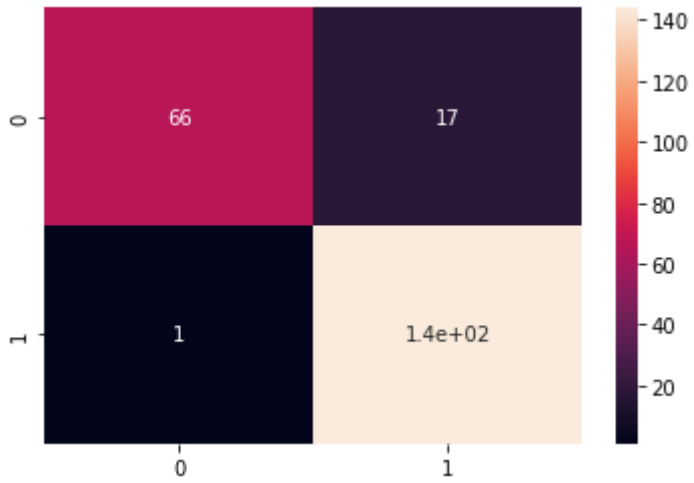
Out[73]:
```
SVC()
```

In [74]:
```
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
```

In [75]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[75]:
```
<AxesSubplot:>
```



train size : test size = 50% : 50%

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [77]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [78]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

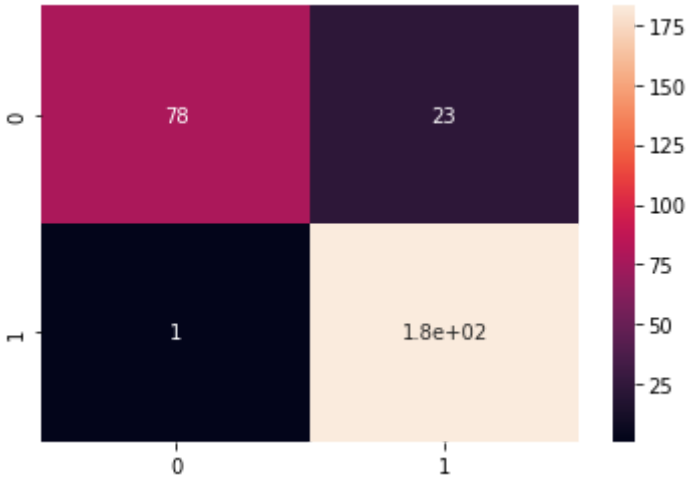
Out[78]:
```
SVC()
```

In [79]:
```
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
```

In [80]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[80]:
```
<AxesSubplot:>
```



train size : test size = 40% : 60%

In [81]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [82]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [83]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

Out[83]:
```
SVC()
```

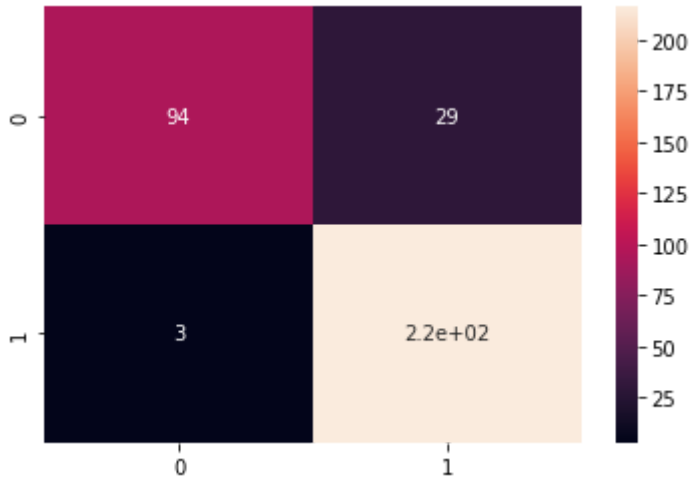
In [84]:
```
y_pred = gaussain_SVC_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
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
```

In [85]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[85]:
```
<AxesSubplot:>
```



train size : test size = 30% : 70%

In [86]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [87]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [88]:
```
gaussain_SVC_classifier.fit(X_train, y_train)
```

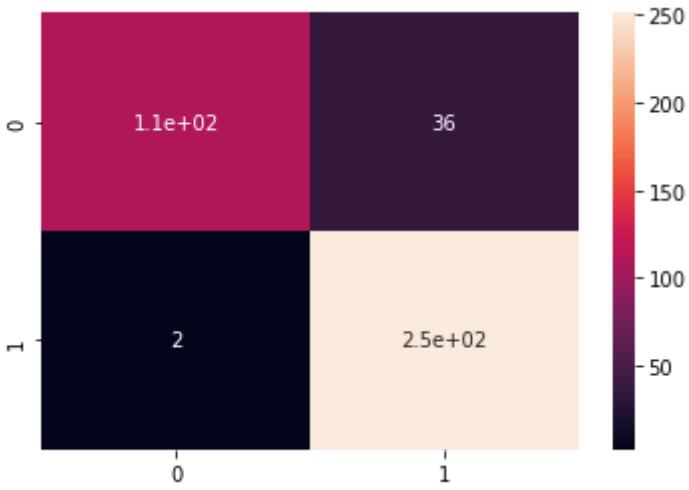
Out[88]:
```
SVC()
```

In [89]:
```
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
```

In [90]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[90]:
```
<AxesSubplot:>
```



Sigmoid SVC Classifier

In [91]:
```
sigmoid_SVC_classifier = SVC(kernel='sigmoid', C=0.9)
sigmoid_SVC_classifier
```

Out[91]:
```
SVC(C=0.9, kernel='sigmoid')
```

train size : test size = 70% : 30%

In [92]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [93]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [94]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

Out[94]:
```
SVC(C=0.9, kernel='sigmoid')
```

In [95]:
```
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
```

In [96]:
```
from sklearn.model_selection import GridSearchCV
```

In [97]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [97], in <cell line: 1>()
----> 1 grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
      2 grid.fit(X_train,y_train)

NameError: name 'param_grid' is not defined
```

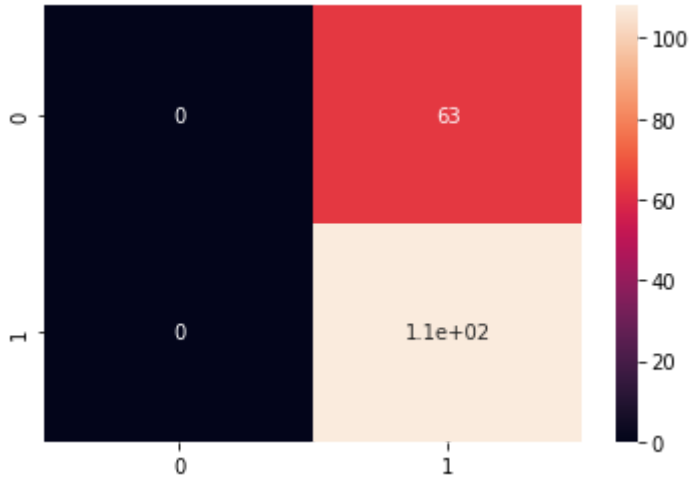
In [ ]:
```
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel':
```

In [ ]:
```
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)
```

In [ ]:
```
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_prediction
```

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



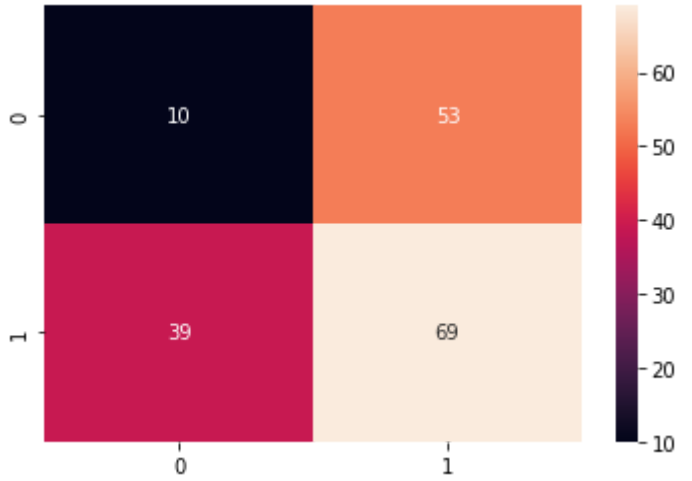

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        63
           1       0.63      1.00      0.77       108

    accuracy                           0.63       171
   macro avg       0.32      0.50      0.39       171
weighted avg       0.40      0.63      0.49       171
```

```
Accuracy Score of RBF kernel 0.631578947368421
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6f6be90>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

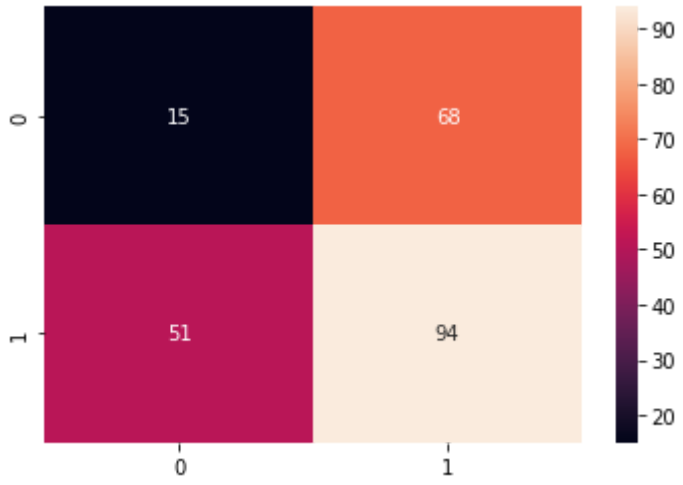
Out[ ]:
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6f10c50>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

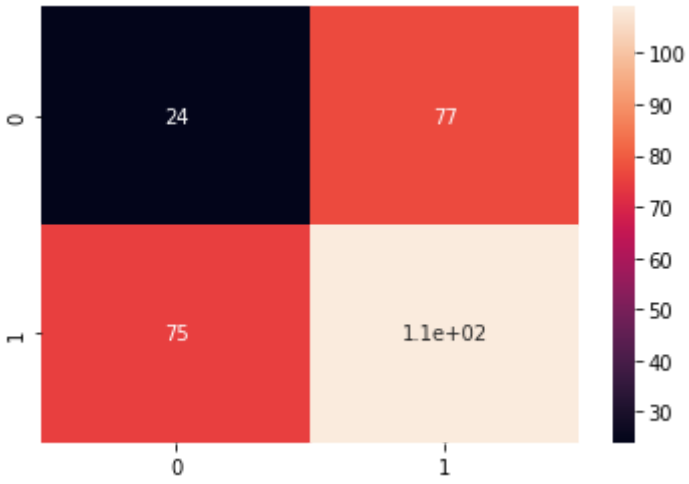
Out[ ]:
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6e50050>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

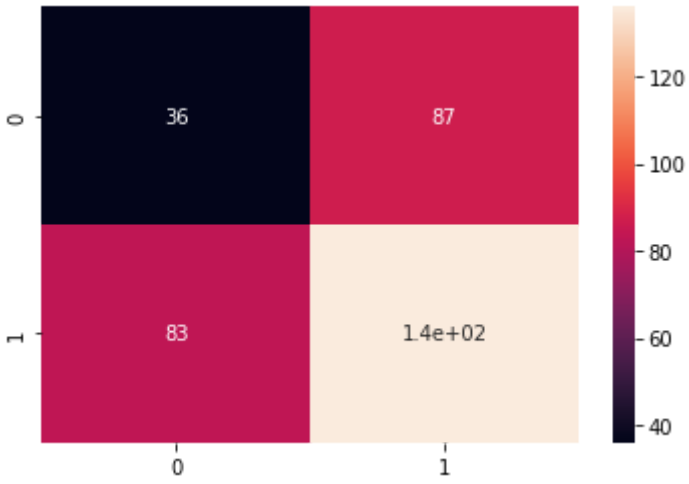
Out[ ]:
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6d7d590>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [ ]:
```
sigmoid_SVC_classifier.fit(X_train, y_train)
```

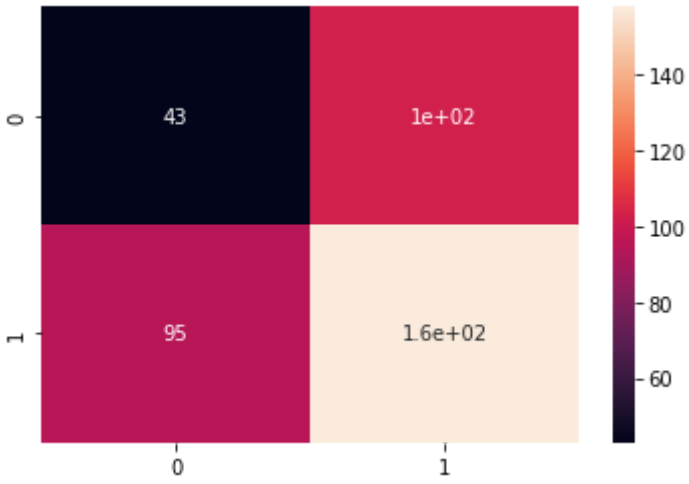
Out[ ]:
```
SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoi
d',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6cb9650>
```



MLP Classifier

In [ ]:
```
mlp_classifier = MLPClassifier(learning_rate='constant', max_iter=600)
mlp_classifier
```

Out[ ]:
```
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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

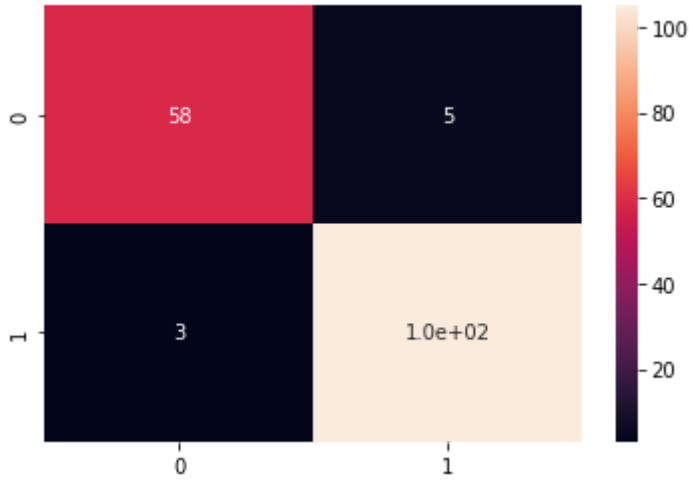
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6d08790>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

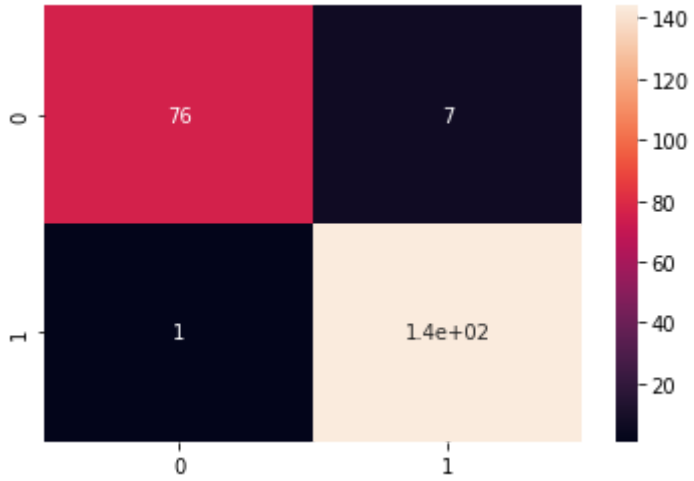
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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: 96.49122807017544%

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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6b6fe90>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

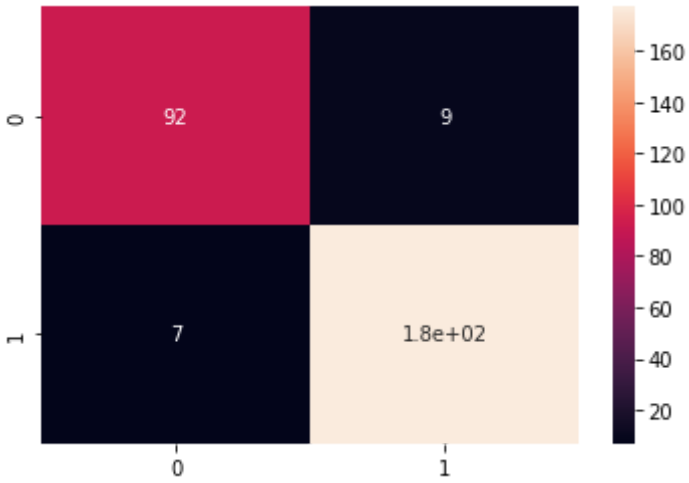
Out[ ]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=600,
              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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6aaf5d0>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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)
```

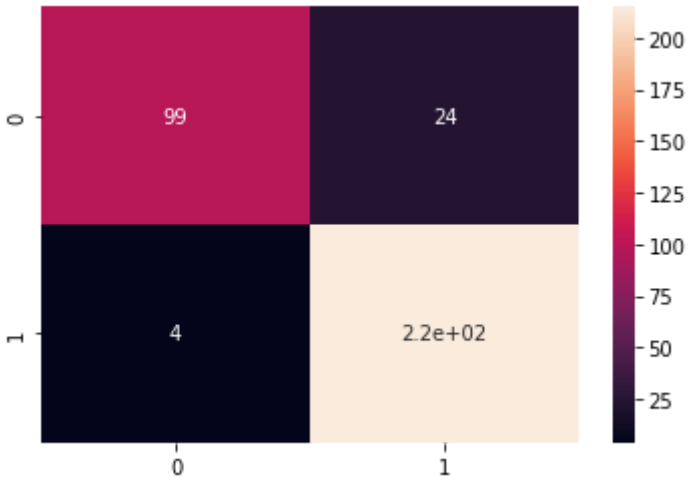
In [ ]:
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6a37550>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [ ]:
```
mlp_classifier.fit(X_train, y_train)
```

Out[ ]:
```
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)
```

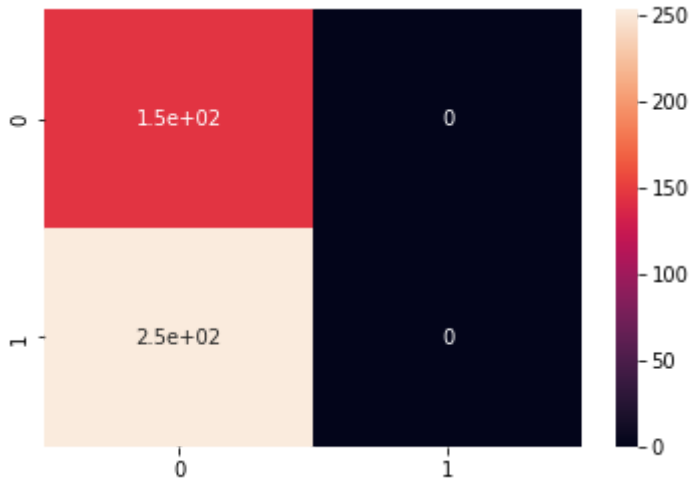
In [ ]:
```
y_pred = mlp_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
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:1
272: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed693b090>
```



Random Forest Classifier

In [ ]:
```
rfc_classifier = RandomForestClassifier(n_estimators=20)
rfc_classifier
```

Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       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%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
398
171
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

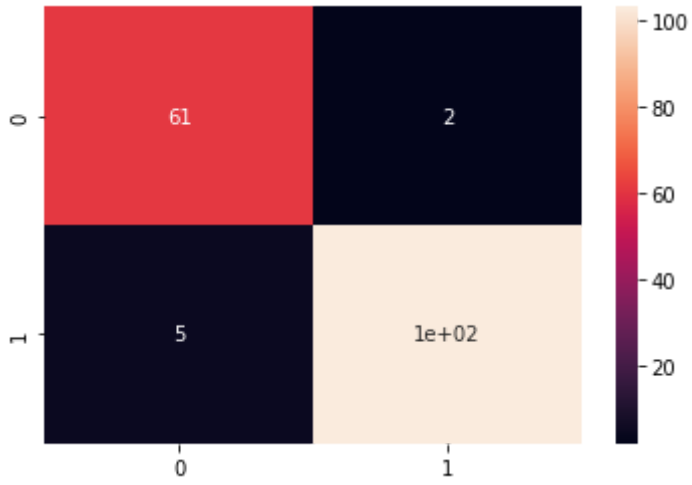
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed68df550>
```



train size : test size = 60% : 40%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
341
228
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       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)
```

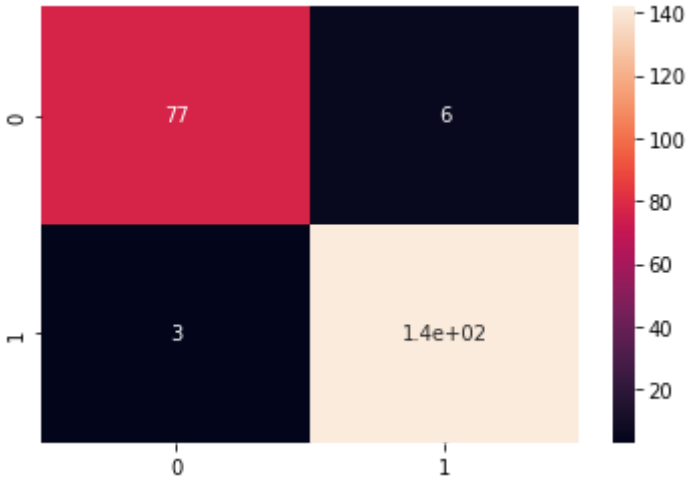
In [ ]:
```
y_pred = rfc_classifier.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
```

```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed73f1cd0>
```



train size : test size = 50% : 50%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
284
285
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

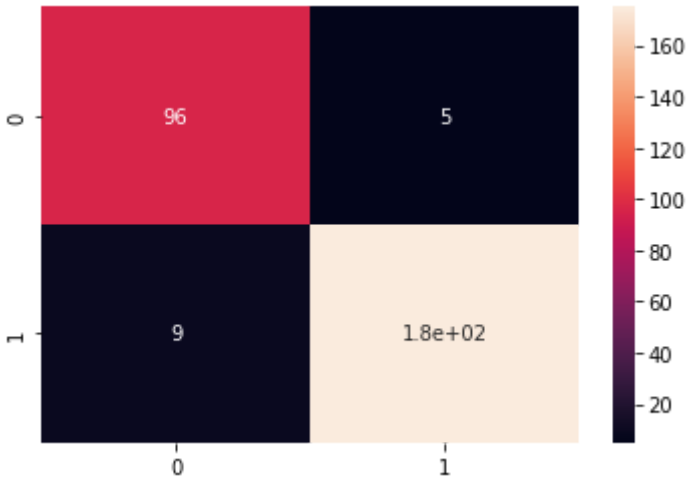
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed6781390>
```



train size : test size = 40% : 60%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
227
342
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

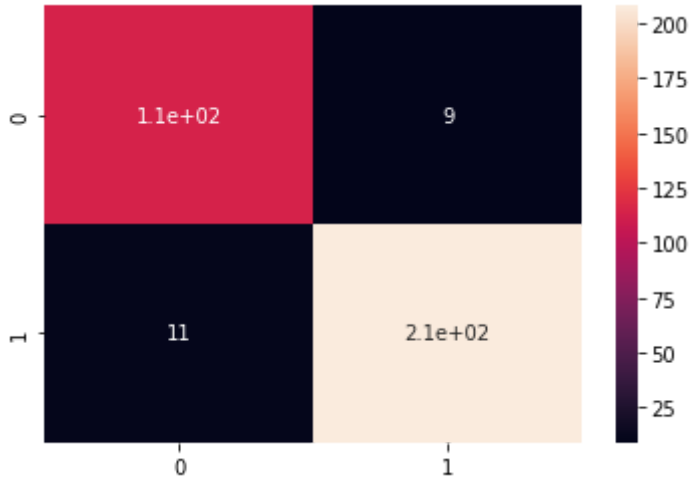
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed66b9990>
```



train size : test size = 30% : 70%

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, ra
```

In [ ]:
```
print(len(X_train))
print(len(y_test))
```

```
170
399
```

In [ ]:
```
rfc_classifier.fit(X_train, y_train)
```

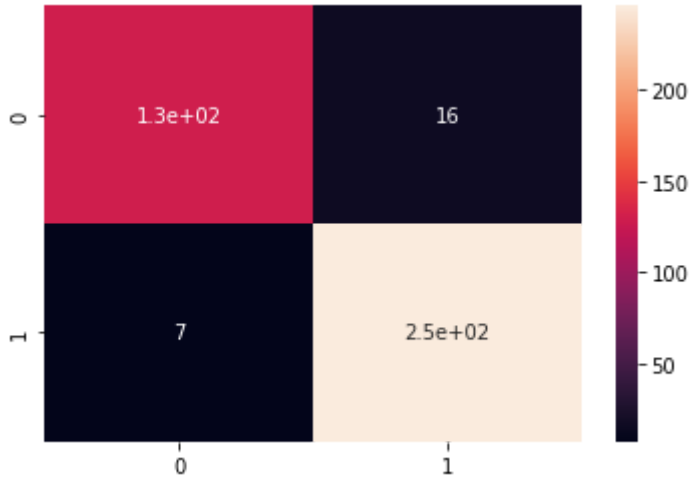
Out[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       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)
```

In [ ]:
```
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: 94.23558897243107%

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
```

In [ ]:
```
sns.heatmap(cf_matrix, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed665e0d0>
```


1.3e+02
16
7
2.5e+02
0
1
0
1
200
150
100
50