Name : Anmol Raj

Roll no :001811001069

IRIS DATASET

Importing modules

In [2]:

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

Load Dataset

In [3]:

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

In [4]:

```
dir(iris)
```

Out[4]:

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

In [5]:

```
iris.data
```

Out[5]:

```
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 [6]:

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

Out[6]:

	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 [7]:

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

Out[7]:

	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 [8]:

```
iris.target_names
```

Out[8]:

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

DataFrame ready to perform

In [9]:

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

Out[9]:

	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 [10]:

```
len(df)
```

Out[10]:

```
150
```

In [11]:

```
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 [12]:

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

Out[12]:

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

train size : test size = 70% : 30%

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [15]:

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

```
105
45
```

In [17]:

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

Out[17]:

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

In [19]:

```
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
```

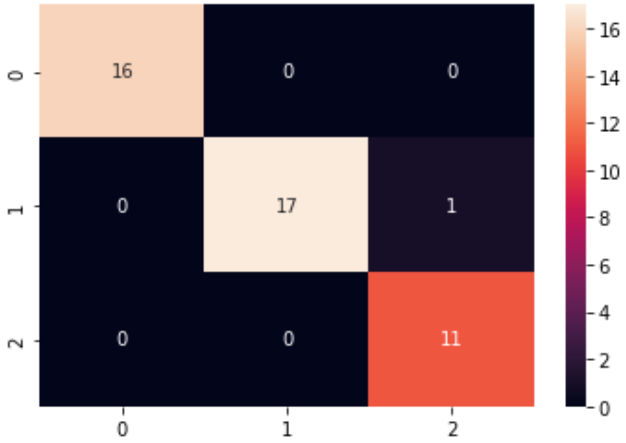
```
           2       0.92      1.00      0.96        11

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

In [20]:

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

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1a8258590>
```



train size : test size = 60% : 40%

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [22]:

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

```
90
60
```

In [23]:

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

Out[23]:

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

In [24]:

```
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]
```

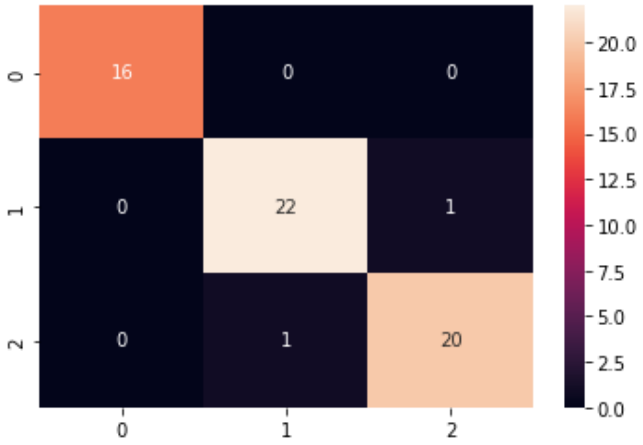
```
 [ 0 22  1]
 [ 0  1 20]]

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 [25]:

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

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1a80c9510>
```



train size : test size = 50% : 50%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [26]:

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

```
90
60
```

In [27]:

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

Out[27]:

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

In [28]:

```
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))
```

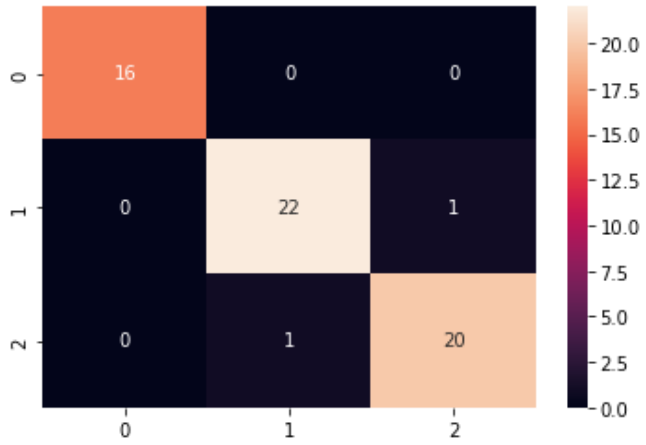
```
Accuracy: 96.66666666666667%

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 [29]:

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

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19fad3c50>
```



train size : test size = 40% : 60%

In [30]:

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

In [31]:

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

```
60
90
```

In [32]:

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

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='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
```

```
    tol=0.001, verbose=False)
```

In [33]:

```
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))
```

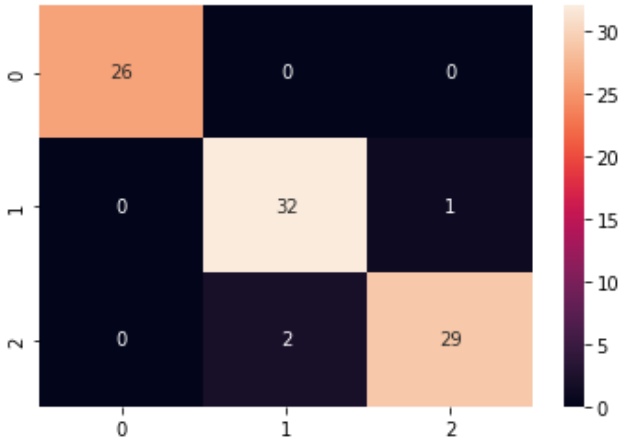
```
Accuracy: 96.66666666666667%

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 [34]:

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

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1a81b4890>
```



train size : test size = 30% : 70%

In [35]:

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

In [36]:

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

```
45
105
```

In [37]:

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

Out[37]:

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

In [38]:

```
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))
```

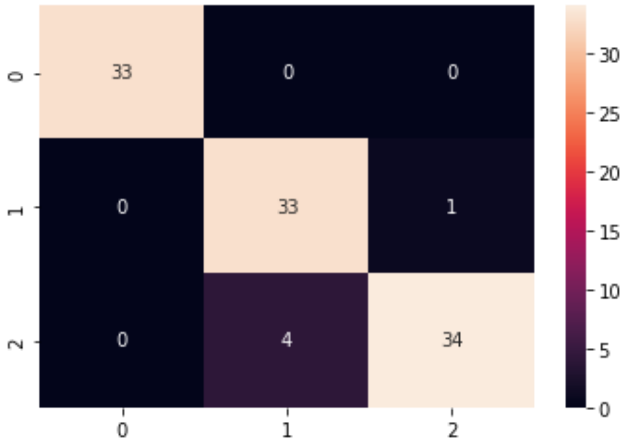
```
Accuracy: 95.23809523809523%

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 [39]:

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

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f953750>
```



Polynomial SVC Classifier

In [40]:

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

```
poly_SVC_classifier
```

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)
```

train size : test size = 70% : 30%

In [41]:

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

In [42]:

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

```
105
45
```

In [43]:

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

Out[43]:

```
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 [44]:

```
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))
```

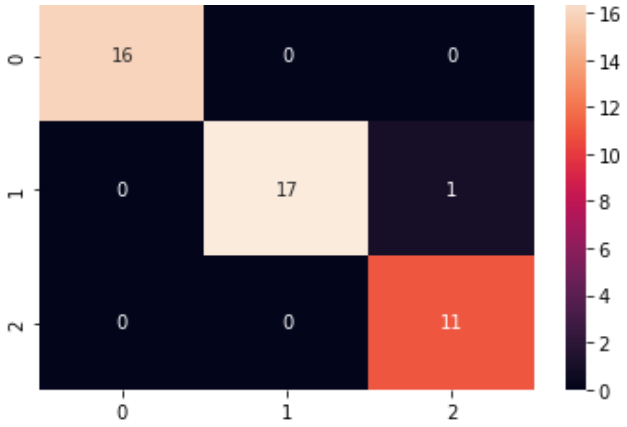
```
Accuracy: 97.77777777777778%

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 [45]:

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

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f88f390>
```



train size : test size = 60% : 40%

In [46]:

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

In [47]:

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

```
90
60
```

In [48]:

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

Out[48]:

```
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 [49]:

```
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))
```

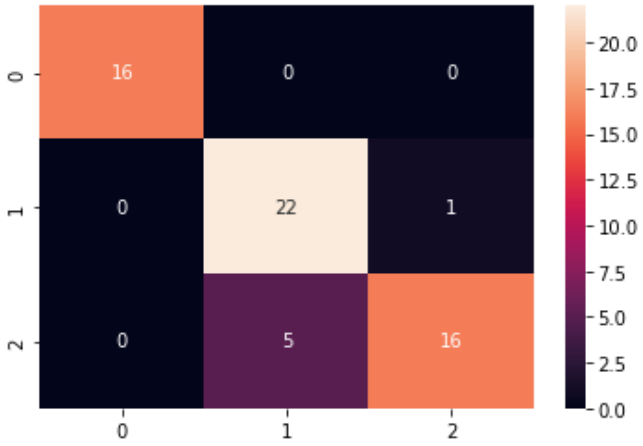
```
Accuracy: 90.0%

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 [50]:

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

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f7b5b90>
```



train size : test size = 50% : 50%

In [51]:

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

In [52]:

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

```
75
75
```

In [53]:

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

Out[53]:

```
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 [54]:

```
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
```

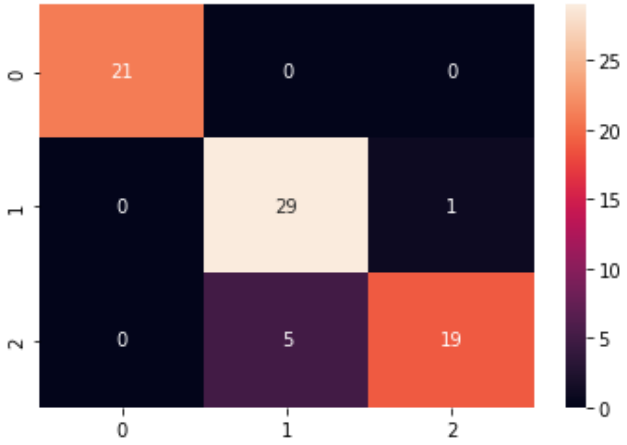
```
   macro avg       0.93      0.92      0.92         75
weighted avg       0.93      0.92      0.92         75
```

In [55]:

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

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f709150>
```



train size : test size = 40% : 60%

In [66]:

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

In [67]:

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

```
60
90
```

In [68]:

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

Out[68]:

```
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 [59]:

```
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:
```

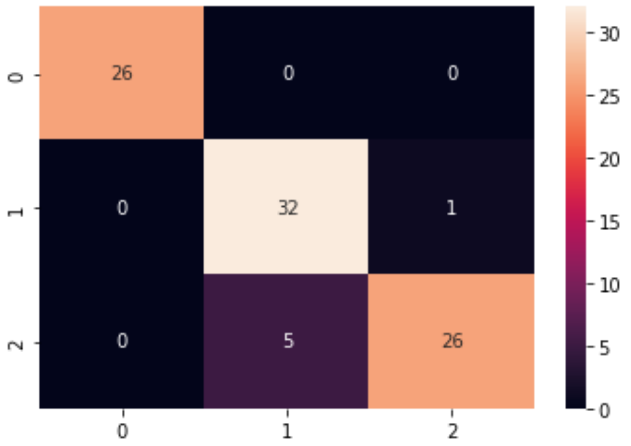
```
              precision    recall  f1-score   support

           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 [60]:

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

Out[60]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f62e450>
```



train size : test size = 30% : 70%

In [61]:

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

In [62]:

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

```
45
105
```

In [63]:

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

Out[63]:

```
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 [64]:

```
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:
```

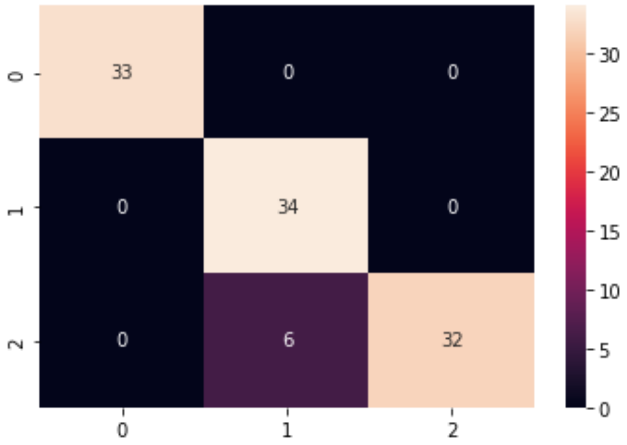
```
 [[33  0  0]
 [ 0 34  0]
 [ 0  6 32]]

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 [65]:

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

Out[65]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f571150>
```



Gaussain SVC Classifier

In [69]:

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

Out[69]:

```
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 [70]:

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

In [71]:

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

```
105
45
```

In [72]:

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

Out[72]:

```
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 [73]:

```
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))
```

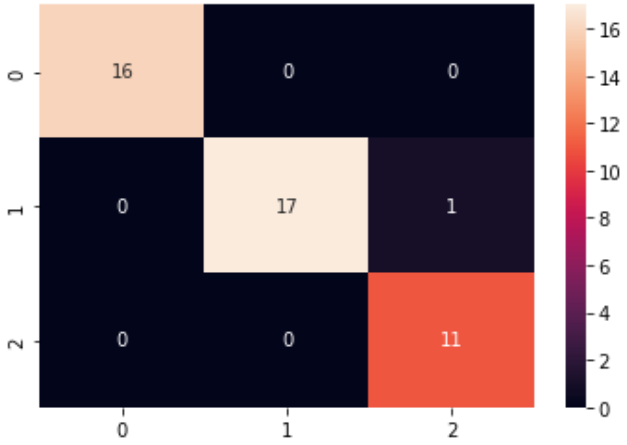
```
Accuracy: 97.77777777777778%

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 [74]:

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

Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f7adfd0>
```



train size : test size = 60% : 40%

In [80]:

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

In [81]:

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

```
90
60
```

In [82]:

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

Out[82]:

```
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 [83]:

```
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))
```

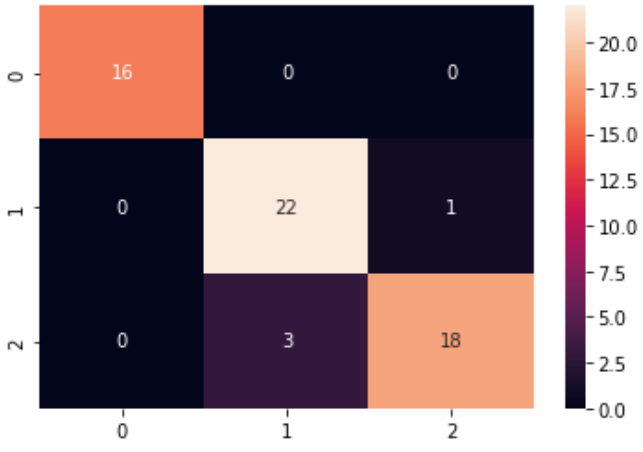
```
Accuracy: 93.33333333333333%

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 [84]:

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

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f325a10>
```



train size : test size = 50% : 50%

In [85]:

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

In [86]:

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

```
75
75
```

In [87]:

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

In [88]:

```
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))
```

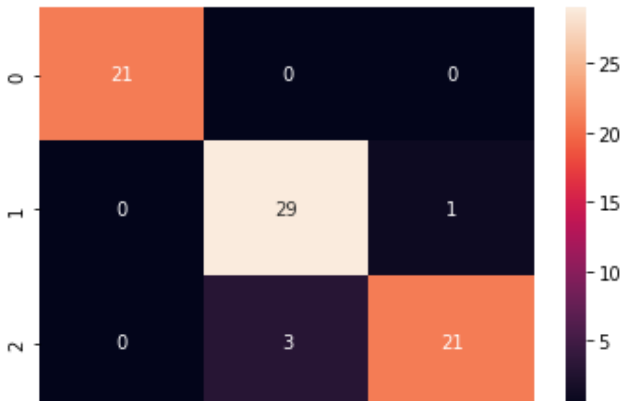
```
Accuracy: 94.66666666666667%

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 [89]:

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

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f7ba610>
```



train size : test size = 40% : 60%

In [90]:

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

In [91]:

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

```
60
90
```

In [92]:

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

In [93]:

```
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))
```

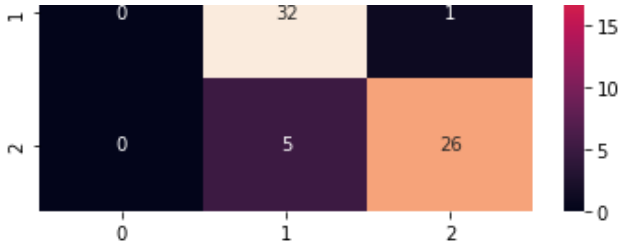
```
Accuracy: 93.33333333333333%

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 [94]:

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

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f1e38d0>
```



train size : test size = 30% : 70%

In [95]:

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

In [96]:

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

```
45
105
```

In [97]:

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

In [98]:

```
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))
```

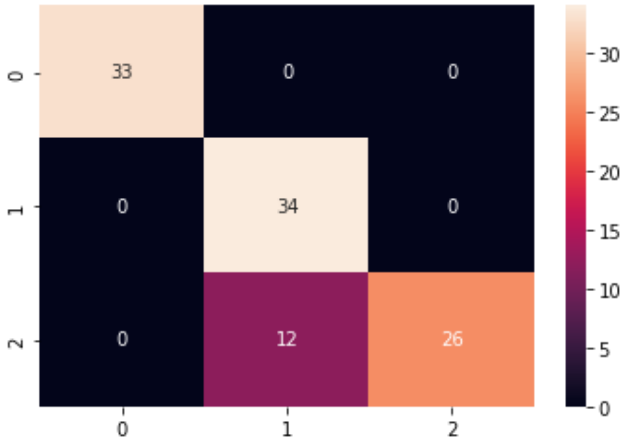
```
Accuracy: 88.57142857142857%

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 [99]:

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

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f1982d0>
```



Sigmoid SVC Classifier

In [100]:

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

Out[100]:

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

train size : test size = 70% : 30%

In [101]:

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

In [102]:

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

```
105
45
```

In [103]:

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

Out[103]:

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

In [104]:

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

```
Accuracy: 24.444444444444443%

Confusion Matrix:
 [[ 0  0 16]
 [ 0  0 18]
 [ 0  0 11]]
```

```
Classification Report:

              precision    recall  f1-score   support

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

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

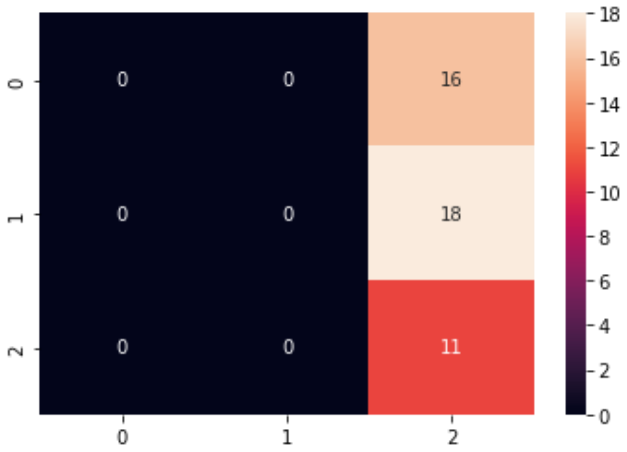
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [105]:

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

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f0d9290>
```



train size : test size = 60% : 40%

In [106]:

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

In [107]:

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

```
90
60
```

In [108]:

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

Out[108]:

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

In [109]:

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

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

```
Accuracy: 26.666666666666668%

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

Classification Report:

              precision    recall  f1-score   support

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

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

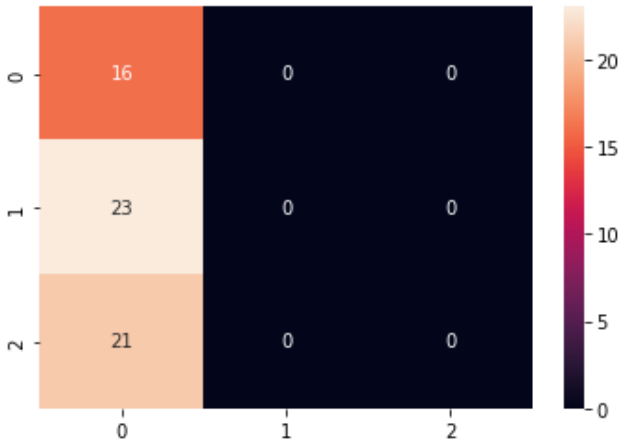
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [110]:

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

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19f015a90>
```



train size : test size = 50% : 50%

In [111]:

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

In [112]:

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

```
75
75
```

In [113]:

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

Out[113]:

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

In [114]:

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

```
Accuracy: 28.000000000000004%

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

Classification Report:

              precision    recall  f1-score   support

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

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

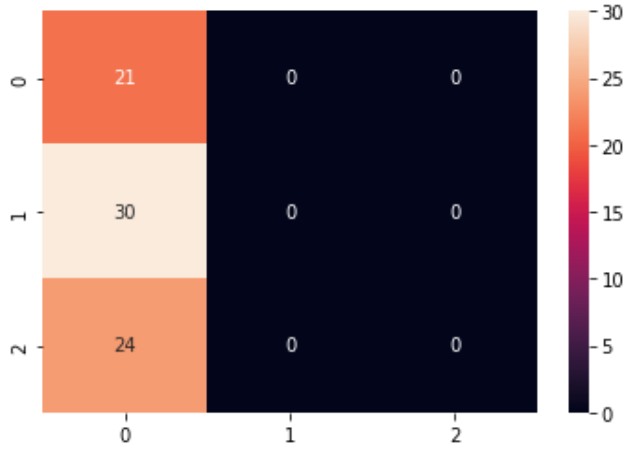
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [115]:

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

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19ef44d50>
```



train size : test size = 40% : 60%

In [116]:

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

In [117]:

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

```
60
90
```

In [118]:

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

Out[118]:

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

In [119]:

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

```
Accuracy: 28.888888888888886%

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

Classification Report:

              precision    recall  f1-score   support

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

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

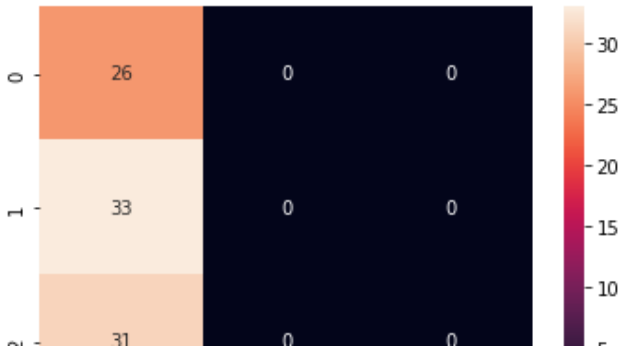
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [120]:

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

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19ee78cd0>
```



train size : test size = 30% : 70%

In [121]:

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

In [122]:

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

```
45
105
```

In [123]:

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

Out[123]:

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

In [124]:

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

```
Accuracy: 31.428571428571427%

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

Classification Report:

              precision    recall  f1-score   support

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

    accuracy                           0.31       105
   macro avg       0.10      0.33      0.16       105
weighted avg       0.10      0.31      0.15       105
```

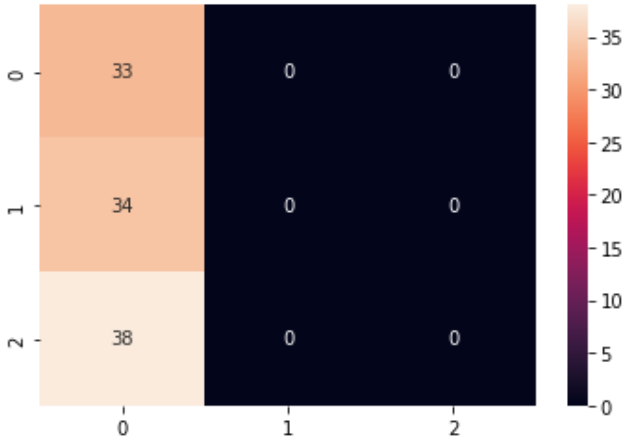
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [125]:

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

Out[125]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19ee1a650>
```



MLP Classifier

In [126]:

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

Out[126]:

```
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 [152]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [153]:

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

```
105
45
```

In [154]:

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

Out[154]:

```
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 [155]:

```
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))
```

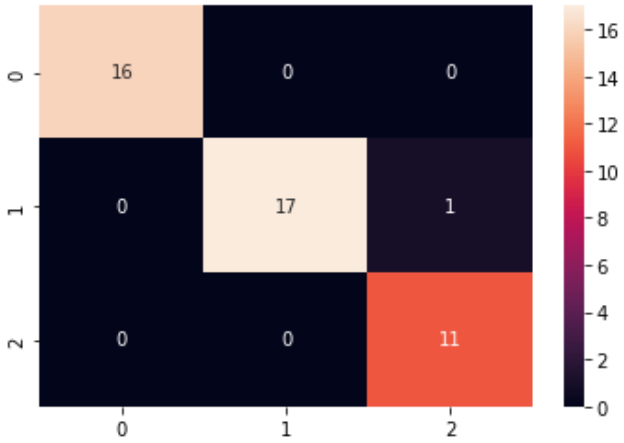
```
Accuracy: 97.77777777777778%

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 [156]:

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

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e9ba750>
```



train size : test size = 60% : 40%

In [157]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [158]:

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

```
90
60
```

In [159]:

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

Out[159]:

```
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 [160]:

```
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))
```

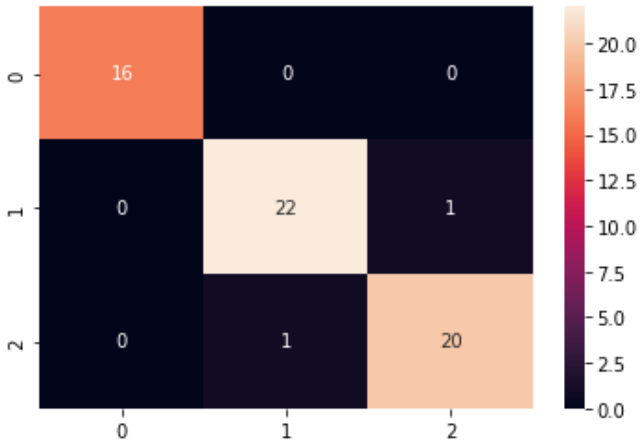
```
Accuracy: 96.66666666666667%

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 [161]:

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

Out[161]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e8b5ad0>
```



train size : test size = 50% : 50%

In [162]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [163]:

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

```
75
75
```

In [164]:

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

Out[164]:

```
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 [165]:

```
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))
```

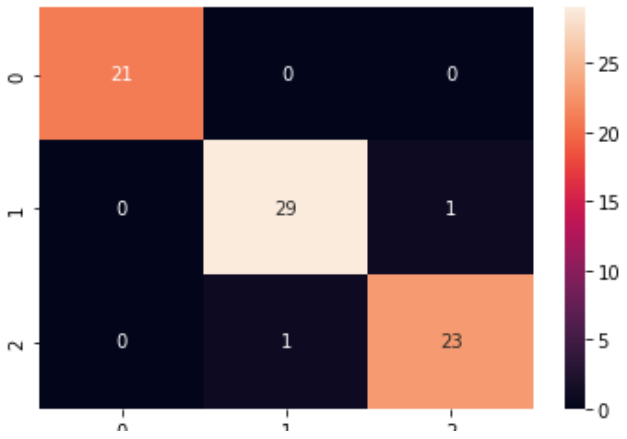
```
Accuracy: 97.33333333333334%

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 [166]:

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

Out[166]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e808690>
```



0 1 2

train size : test size = 40% : 60%

In [167]:

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

In [168]:

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

```
60
90
```

In [169]:

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

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (600) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Out[169]:

```
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 [170]:

```
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))
```

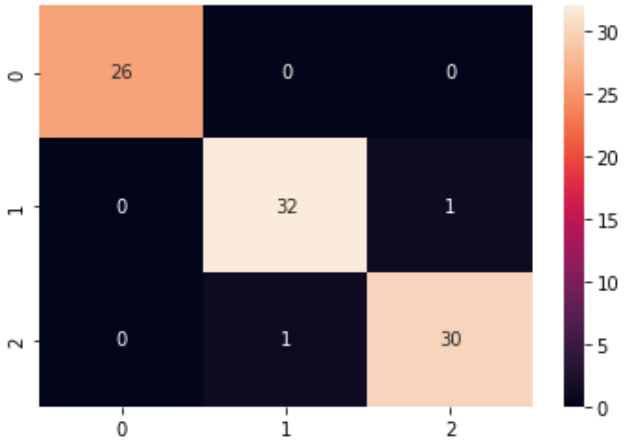
```
Accuracy: 97.77777777777777%

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 [171]:

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

Out[171]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e7391d0>
```



train size : test size = 30% : 70%

In [172]:

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

In [173]:

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

```
45
105
```

In [174]:

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

Out[174]:

```
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 [175]:

```
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
```

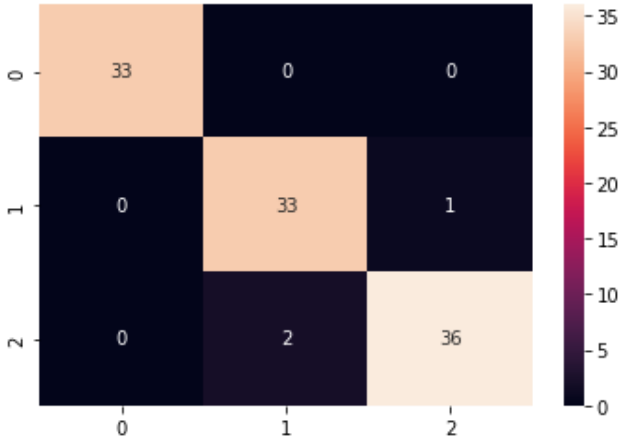
```
           1       0.94      0.97      0.96        34
           2       0.97      0.95      0.96        38

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

In [176]:

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

Out[176]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e66ff50>
```



Random Forest Classifier

In [177]:

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

Out[177]:

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

train size : test size = 70% : 30%

In [178]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [179]:

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

```
105
45
```

In [180]:

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

Out[180]:

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

In [181]:

```
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))
```

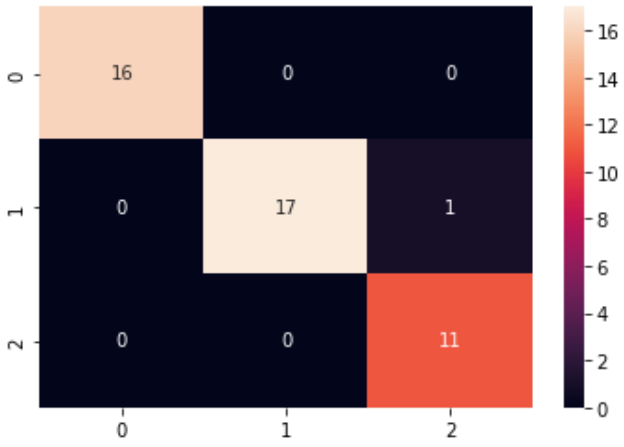
```
Accuracy: 97.77777777777777%

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 [182]:

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

Out[182]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e5a8410>
```



train size : test size = 60% : 40%

In [183]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [184]:

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

```
90
60
```

In [185]:

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

Out[185]:

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

In [186]:

```
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))
```

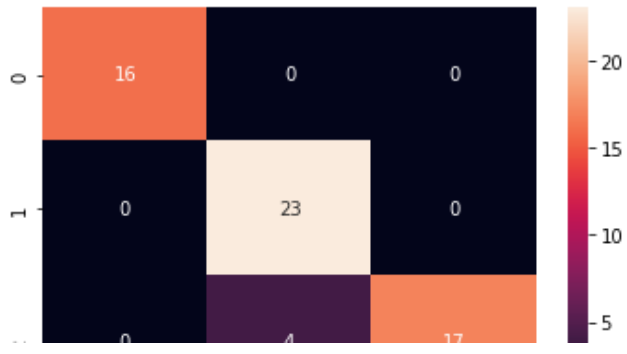
```
Accuracy: 93.33333333333333%

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
```

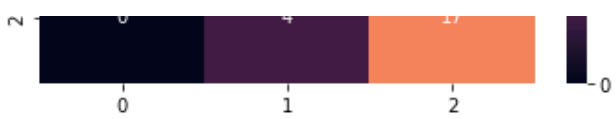
In [187]:

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

Out[187]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e558ad0>
```





train size : test size = 50% : 50%

In [188]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [189]:

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

```
75
75
```

In [190]:

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

Out[190]:

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

In [191]:

```
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))
```

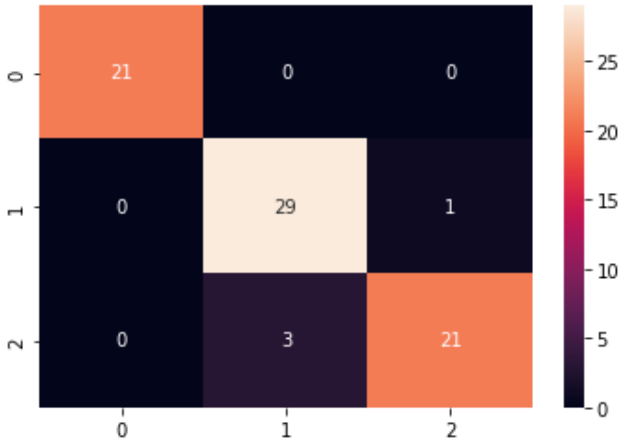
```
Accuracy: 94.66666666666667%

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 [192]:

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

Out[192]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e494d90>
```



train size : test size = 40% : 60%

In [193]:

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

In [194]:

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

```
60
90
```

In [195]:

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

Out[195]:

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

In [196]:

```
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: 94.44444444444444%

Confusion Matrix:

[[26  0  0]
 [ 0 33  0]
 [ 0  5 26]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.87      1.00      0.93        33
           2       1.00      0.84      0.91        31
```

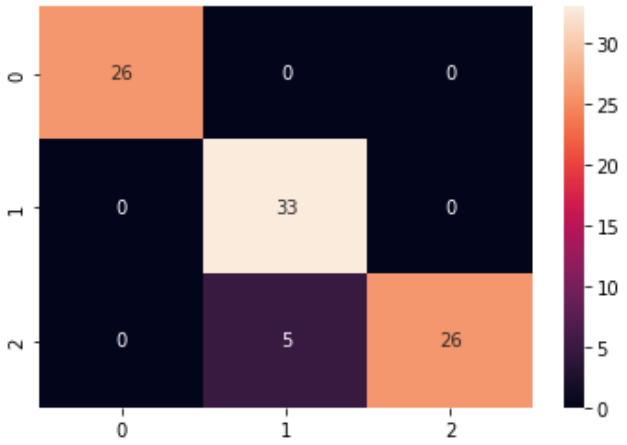
```
    accuracy                           0.94        90
   macro avg       0.96      0.95      0.95        90
weighted avg       0.95      0.94      0.94        90
```

In [197]:

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

Out[197]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e3a37d0>
```



train size : test size = 30% : 70%

In [198]:

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

In [199]:

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

```
45
105
```

In [200]:

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

Out[200]:

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

In [201]:

```
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: 94.28571428571428%
```

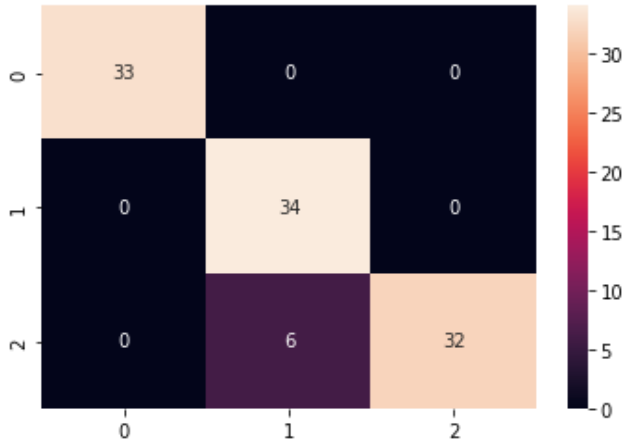
```
Confusion Matrix:

[[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 [202]:

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

Out[202]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd19e300c90>
```



Name : Anmol Raj

Roll no :001811001069

CANCER-DATA

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, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
```

Load Dataset

In [2]:

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

In [3]:

```
dir(b_cancer)
```

Out[3]:

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

In [4]:

```
b_cancer.data
```

Out[4]:

```
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 [5]:

```
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 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 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 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 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 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 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

In [6]:

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

Out[6]:

	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
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	1.0950	0.9053
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	0.7456	0.7869
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	0.4956	1.1560
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	0.7572	0.7813

In [7]:

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

Out[7]:

	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
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	1.0950	0.9053
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	0.7456	0.7869
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	0.4956	1.1560
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	0.7572	0.7813

In [8]:

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

Out[8]:

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

DataFrame ready to perform

In [9]:

```
len(df)
```

Out[9]:

```
569
```

In [10]:

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

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

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

SVC Classfier

Linear SVC Classifier

In [37]:

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

Out[37]:

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

train size : test size = 70% : 30%

In [38]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [39]:

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

```
398
171
```

In [40]:

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

In [41]:

```
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))
```

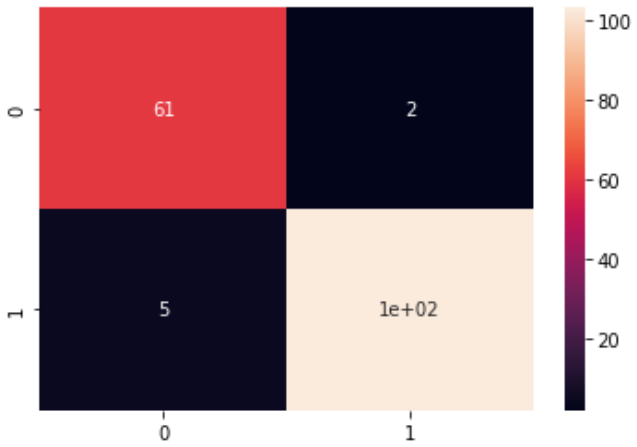
```
Accuracy: 95.90643274853801%

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 [42]:

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

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14a2c9390>
```



train size : test size = 60% : 40%

In [43]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [44]:

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

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

```
341
228
```

In [45]:

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

In [46]:

```
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))
```

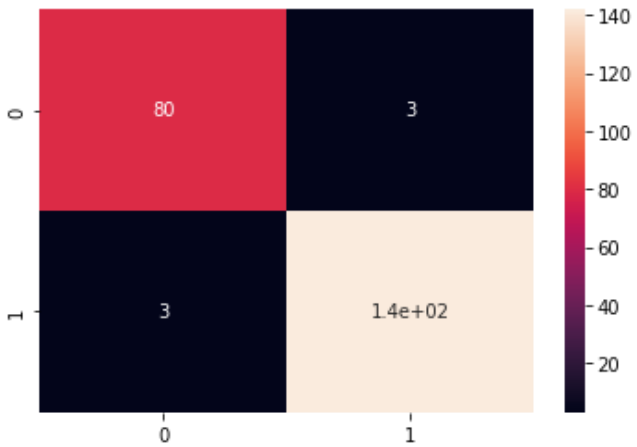
```
Accuracy: 97.36842105263158%

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 [47]:

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

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14a11d290>
```



train size : test size = 50% : 50%

In [48]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [49]:

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

```
284
285
```

In [50]:

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

In [51]:

```
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))
```

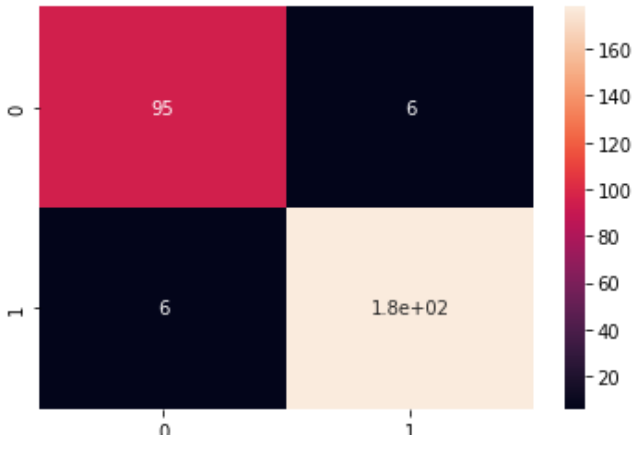
```
Accuracy: 95.78947368421052%

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 [52]:

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

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1529a0690>
```



train size : test size = 40% : 60%

In [53]:

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

In [54]:

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

```
227
342
```

In [55]:

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

In [56]:

```
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))
```

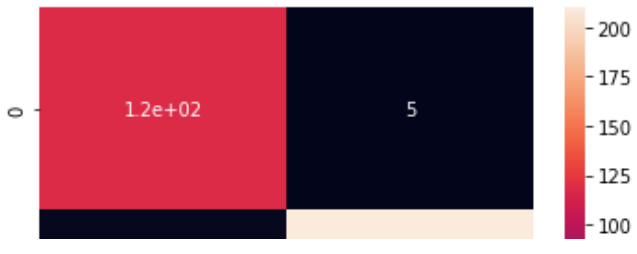
```
Accuracy: 95.90643274853801%

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
```

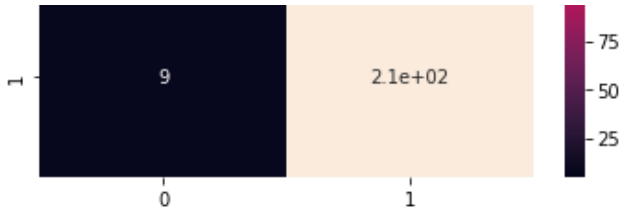
In [57]:

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

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149fecad0>
```





train size : test size = 30% : 70%

In [58]:

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

In [59]:

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

```
170
399
```

In [60]:

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

Out[60]:

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

In [61]:

```
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))
```

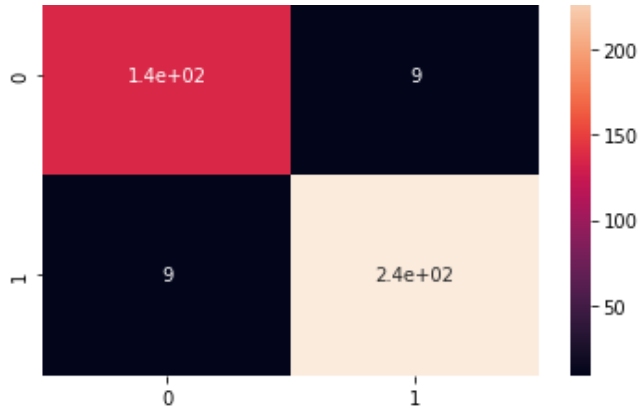
```
Accuracy: 95.48872180451127%

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 [62]:

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

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14a04fdd0>
```



Polynomial SVC Classifier

In [63]:

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

Out[63]:

```
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 [64]:

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

In [65]:

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

```
398
171
```

In [66]:

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

In [67]:

```
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:
```

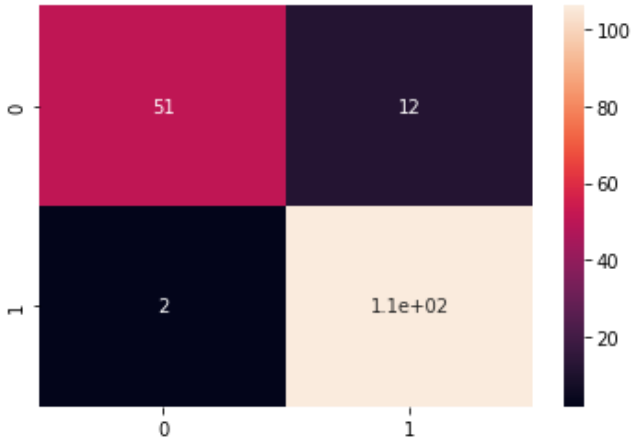
```
              precision    recall  f1-score   support

           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 [68]:

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

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa155d5c110>
```



train size : test size = 60% : 40%

In [69]:

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

In [70]:

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

```
341
228
```

In [71]:

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

In [72]:

```
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:
```

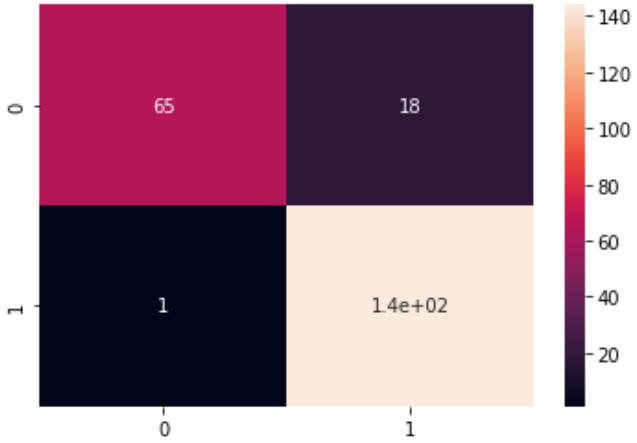
```
 [[ 65  18]
 [  1 144]]

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 [73]:

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

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149ddd550>
```



train size : test size = 50% : 50%

In [74]:

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

In [75]:

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

```
284
285
```

In [76]:

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

In [77]:

```
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))
```

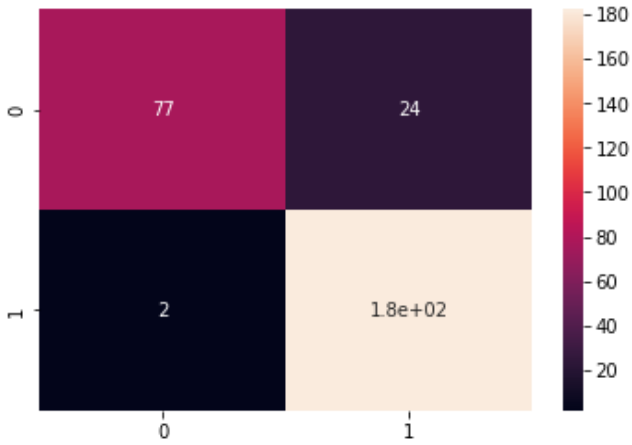
```
Accuracy: 90.87719298245615%

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 [78]:

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

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149d7e310>
```



train size : test size = 40% : 60%

In [79]:

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

In [80]:

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

```
227
342
```

In [81]:

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

In [82]:

```
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))
```

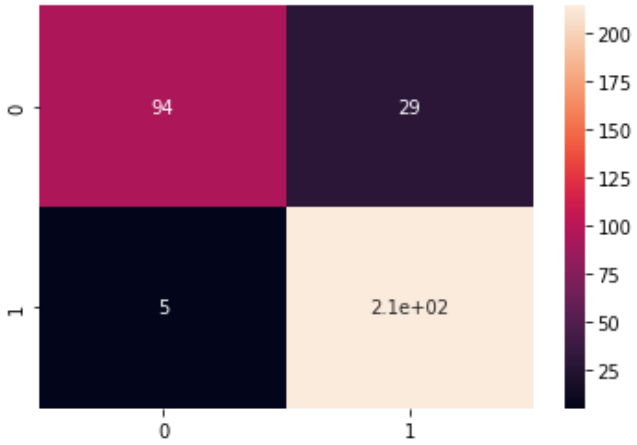
```
Accuracy: 90.05847953216374%

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 [83]:

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

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14a1c5150>
```



train size : test size = 30% : 70%

In [84]:

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

In [85]:

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

```
170
399
```

In [86]:

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

Out[86]:

```
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 [87]:

```
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))
```

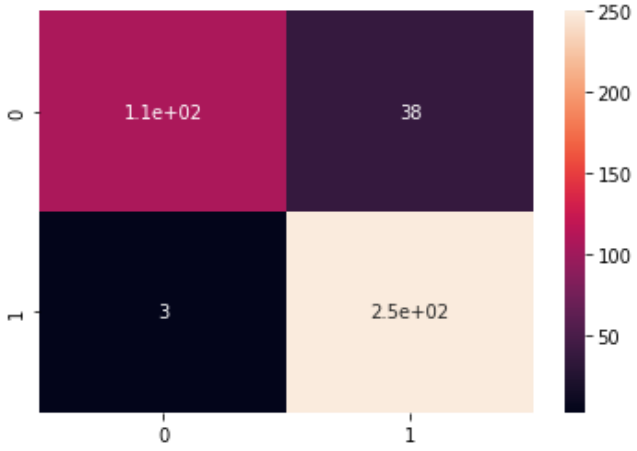
```
Accuracy: 89.72431077694235%

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 [88]:

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

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149c2af10>
```



Gaussain SVC Classifier

In [89]:

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

Out[89]:

```
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 [90]:

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

In [91]:

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

```
398
171
```

In [92]:

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

In [93]:

```
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))
```

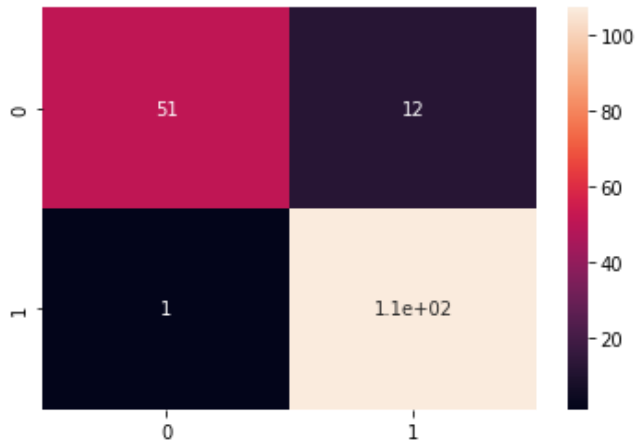
```
Accuracy: 92.39766081871345%

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 [94]:

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

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149b5bb90>
```



train size : test size = 60% : 40%

In [95]:

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

In [96]:

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

```
341
228
```

In [97]:

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

In [98]:

```
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))
```

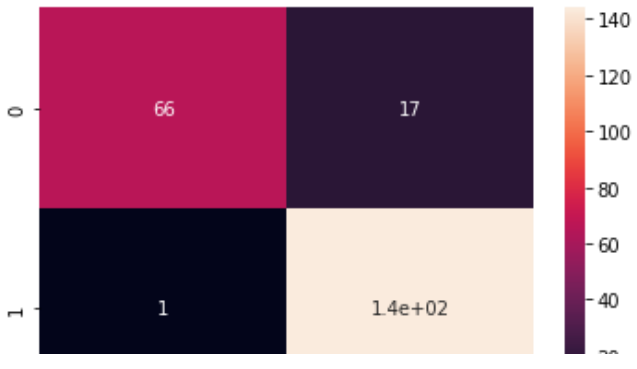
```
Accuracy: 92.10526315789474%

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 [99]:

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

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149aef090>
```



train size : test size = 50% : 50%

In [100]:

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

In [101]:

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

```
284
285
```

In [102]:

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

In [103]:

```
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))
```

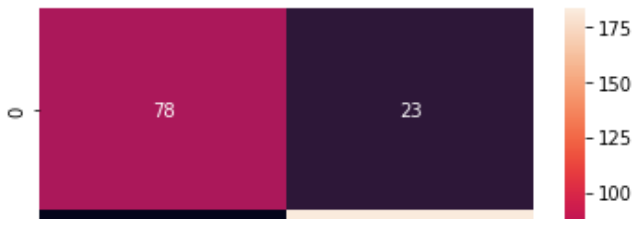
```
Accuracy: 91.57894736842105%

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
```

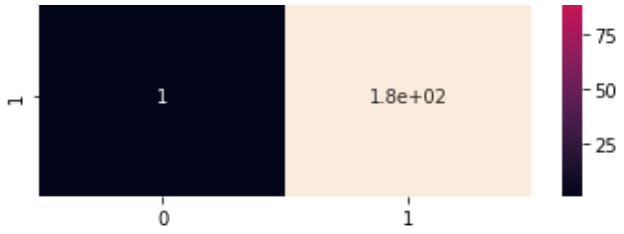
In [104]:

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

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149af1890>
```





train size : test size = 40% : 60%

In [105]:

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

In [106]:

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

```
227
342
```

In [107]:

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

In [108]:

```
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))
```

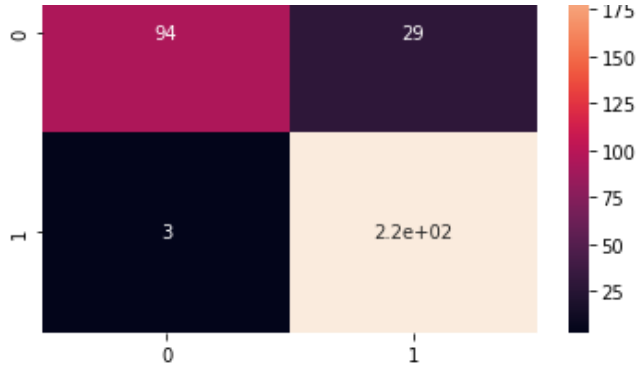
```
Accuracy: 90.64327485380117%

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 [109]:

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

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1499c92d0>
```



train size : test size = 30% : 70%

In [110]:

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

In [111]:

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

```
170
399
```

In [112]:

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

Out[112]:

```
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 [113]:

```
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))
```

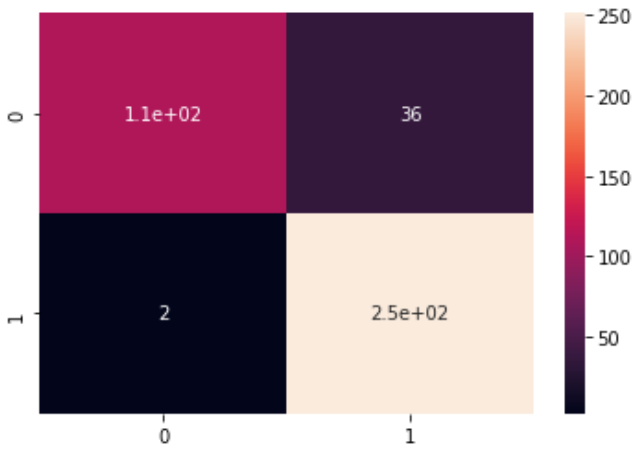
```
Accuracy: 90.47619047619048%

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 [114]:

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

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1498fd910>
```



Sigmoid SVC Classifier

In [115]:

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

Out[115]:

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

train size : test size = 70% : 30%

In [116]:

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

In [117]:

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

```
398
171
```

In [118]:

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

Out[118]:

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

In [119]:

```
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 [120]:

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

In [124]:

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

In [126]:

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

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

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[CV] ..... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid .....
[CV] ..... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid .....
[CV] ..... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=1, kernel=sigmoid .....
[CV] ..... C=1, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.001, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.001, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.001, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=1, gamma=0.001, kernel=sigmoid .....
[CV] ..... C=1, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=1, kernel=sigmoid .....
[CV] ..... C=10, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=1, kernel=sigmoid .....
[CV] ..... C=10, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=1, kernel=sigmoid .....
[CV] ..... C=10, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=1, kernel=sigmoid .....
[CV] ..... C=10, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.1, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.01, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.001, kernel=sigmoid .....
[CV] ..... C=10, gamma=0.001, kernel=sigmoid, total=   0.0s
```

```
[CV] C=10, gamma=0.001, kernel=sigmoid ...............................
[CV] ................ C=10, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.001, kernel=sigmoid ...............................
[CV] ................ C=10, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.001, kernel=sigmoid ...............................
[CV] ................ C=10, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=10, gamma=0.001, kernel=sigmoid ...............................
[CV] ................ C=10, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=100, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=100, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=100, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=100, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=1, kernel=sigmoid ..................................
[CV] ................... C=100, gamma=1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=100, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=100, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=100, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=100, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.1, kernel=sigmoid ................................
[CV] ................. C=100, gamma=0.1, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=100, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=100, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=100, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=100, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.01, kernel=sigmoid ...............................
[CV] ................ C=100, gamma=0.01, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.001, kernel=sigmoid ..............................
[CV] ............... C=100, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.001, kernel=sigmoid ..............................
[CV] ............... C=100, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.001, kernel=sigmoid ..............................
[CV] ............... C=100, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.001, kernel=sigmoid ..............................
[CV] ............... C=100, gamma=0.001, kernel=sigmoid, total=   0.0s
[CV] C=100, gamma=0.001, kernel=sigmoid ..............................
[CV] ............... C=100, gamma=0.001, kernel=sigmoid, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    0.8s finished
```

Out[126]:

```
GridSearchCV(cv=None, error_score=nan,
             estimator=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),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001],
                         'kernel': ['sigmoid']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=2)
```

In [127]:

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

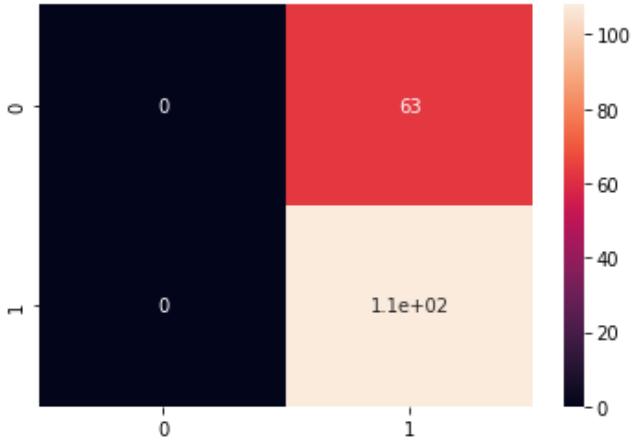
```
SVC(C=0.1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
```

```
    tol=0.001, verbose=False)
```

In [128]:

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

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




```
              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
```

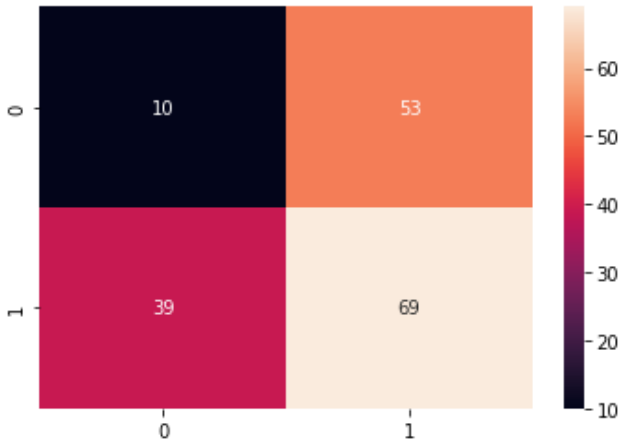
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [129]:

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

Out[129]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14969c310>
```



train size : test size = 60% : 40%

In [130]:

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

In [131]:

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

```
341
228
```

In [132]:

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

Out[132]:

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

In [133]:

```
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))
```

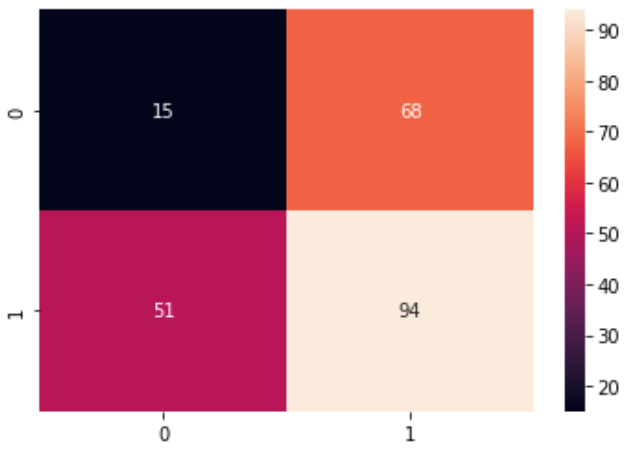
```
Accuracy: 47.80701754385965%

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 [134]:

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

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149633390>
```



train size : test size = 50% : 50%

In [135]:

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

In [136]:

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

```
284
285
```

In [137]:

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

Out[137]:

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

In [138]:

```
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))
```

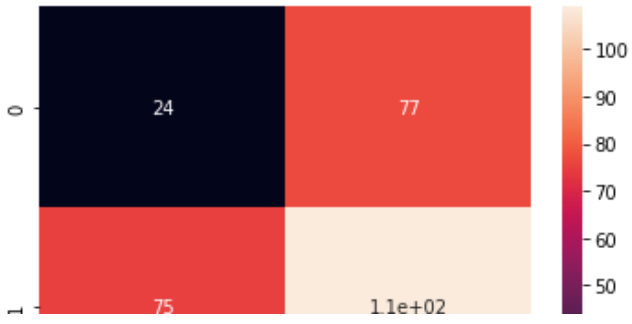
```
Accuracy: 46.666666666666664%

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 [139]:

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

Out[139]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14956a250>
```



train size : test size = 40% : 60%

In [140]:

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

In [141]:

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

```
227
342
```

In [142]:

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

Out[142]:

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

In [143]:

```
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))
```

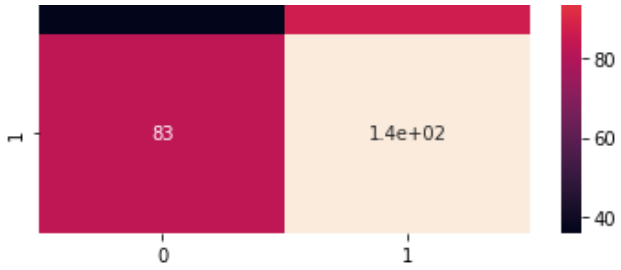
```
Accuracy: 50.29239766081871%

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 [144]:

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

Out[144]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1494df550>
```



train size : test size = 30% : 70%

In [145]:

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

In [146]:

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

```
170
399
```

In [147]:

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

Out[147]:

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

In [148]:

```
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))
```

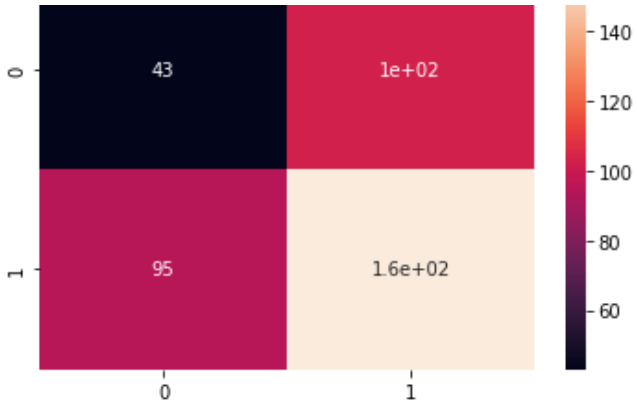
```
Accuracy: 50.37593984962406%

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 [149]:

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

Out[149]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149c649d0>
```



MLP Classifier

In [150]:

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

Out[150]:

```
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 [151]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [152]:

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

```
398
171
```

In [153]:

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

Out[153]:

```
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 [154]:

```
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))
```

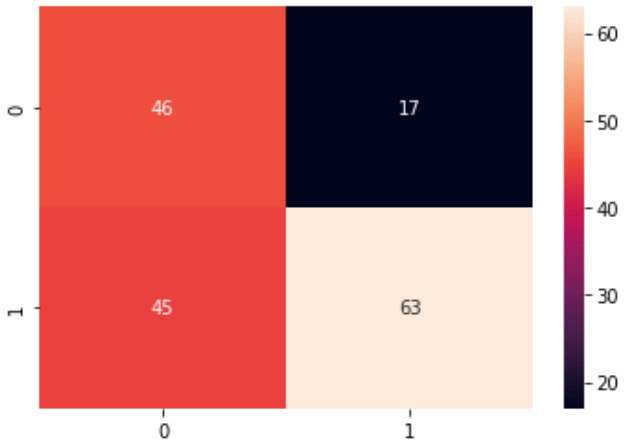
```
Accuracy: 63.74269005847953%

Confusion Matrix:

[[46 17]
 [45 63]]

Classification Report:

              precision    recall  f1-score   support

           0       0.51      0.73      0.60        63
           1       0.79      0.58      0.67       108

    accuracy                           0.64       171
   macro avg       0.65      0.66      0.63       171
weighted avg       0.68      0.64      0.64       171
```

In [155]:

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

Out[155]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149765c50>
```



train size : test size = 60% : 40%

In [156]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [157]:

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

```
341
228
```

In [158]:

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

Out[158]:

```
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 [159]:

```
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))
```

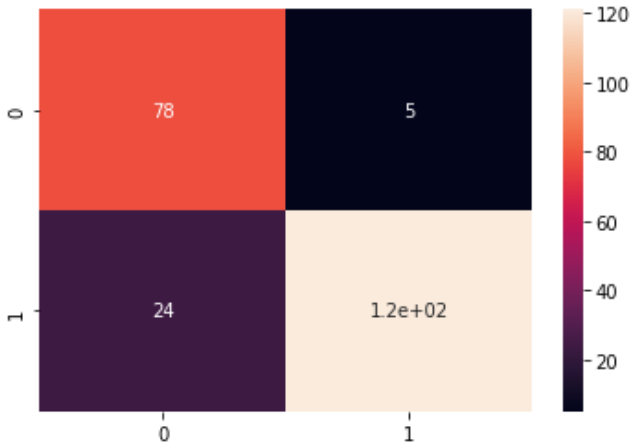
```
Accuracy: 87.28070175438597%

Confusion Matrix:
[[ 78   5]
 [ 24 121]]

Classification Report:

              precision    recall  f1-score   support

           0       0.76      0.94      0.84        83
           1       0.96      0.83      0.89       145

    accuracy                           0.87       228
   macro avg       0.86      0.89      0.87       228
weighted avg       0.89      0.87      0.87       228
```

In [160]:

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

Out[160]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149407290>
```



train size : test size = 50% : 50%

In [161]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [162]:

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

```
284
285
```

In [163]:

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

Out[163]:

```
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 [164]:

```
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))
```

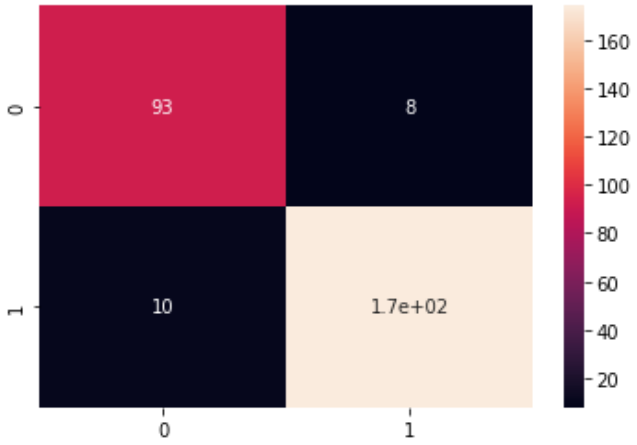
```
Accuracy: 93.6842105263158%

Confusion Matrix:
[[ 93   8]
 [ 10 174]]

Classification Report:

              precision    recall  f1-score   support

           0       0.90      0.92      0.91       101
           1       0.96      0.95      0.95       184

    accuracy                           0.94       285
   macro avg       0.93      0.93      0.93       285
weighted avg       0.94      0.94      0.94       285
```

In [165]:

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

Out[165]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149347a10>
```



train size : test size = 40% : 60%

In [166]:

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

In [167]:

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

```
227
342
```

In [168]:

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

Out[168]:

```
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 [169]:

```
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))
```

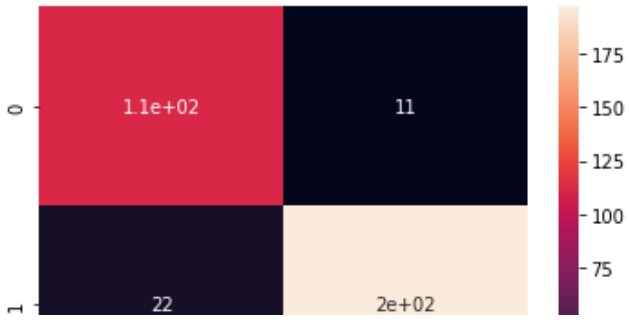
```
Accuracy: 90.35087719298247%

Confusion Matrix:

[[112  11]
 [ 22 197]]

Classification Report:

              precision    recall  f1-score   support

           0       0.84      0.91      0.87       123
           1       0.95      0.90      0.92       219

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

In [170]:

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

Out[170]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149277710>
```



train size : test size = 30% : 70%

In [171]:

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

In [172]:

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

```
170
399
```

In [173]:

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

Out[173]:

```
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 [174]:

```
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))
```

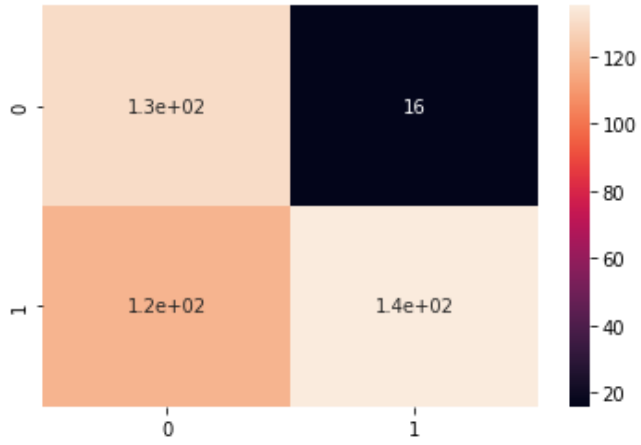
```
Accuracy: 66.41604010025063%

Confusion Matrix:

[[130  16]
 [118 135]]

Classification Report:

              precision    recall  f1-score   support

           0       0.52      0.89      0.66       146
           1       0.89      0.53      0.67       253

    accuracy                           0.66       399
   macro avg       0.71      0.71      0.66       399
weighted avg       0.76      0.66      0.67       399
```

In [175]:

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

Out[175]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149212910>
```



Random Forest Classifier

In [176]:

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

Out[176]:

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

train size : test size = 70% : 30%

In [177]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [178]:

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

```
398
171
```

In [179]:

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

Out[179]:

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

In [180]:

```
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))
```

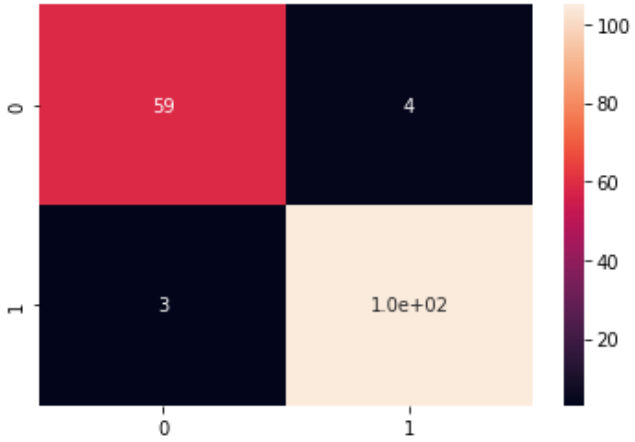
```
Accuracy: 95.90643274853801%

Confusion Matrix:

[[ 59   4]
 [  3 105]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.94      0.94        63
           1       0.96      0.97      0.97       108

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

In [181]:

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

Out[181]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa149140f50>
```



train size : test size = 60% : 40%

In [182]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [183]:

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

```
341
228
```

In [184]:

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

Out[184]:

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

```
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [185]:

```
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))
```

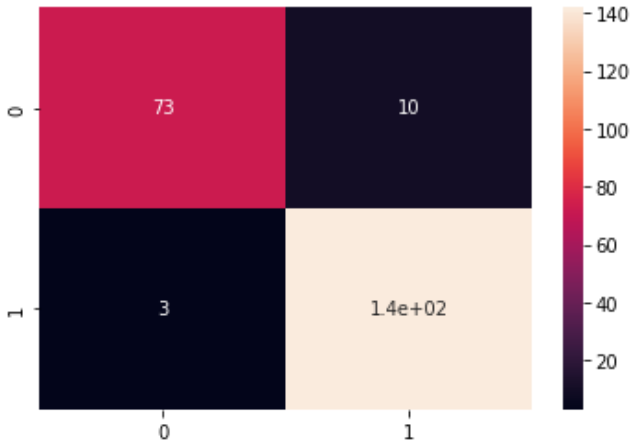
```
Accuracy: 94.2982456140351%

Confusion Matrix:
[[ 73  10]
 [  3 142]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.88      0.92        83
           1       0.93      0.98      0.96       145

    accuracy                           0.94       228
   macro avg       0.95      0.93      0.94       228
weighted avg       0.94      0.94      0.94       228
```

In [186]:

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

Out[186]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1490cca90>
```



train size : test size = 50% : 50%

In [187]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [188]:

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

```
284
```

285

In [189]:

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

Out[189]:

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

In [190]:

```
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))
```

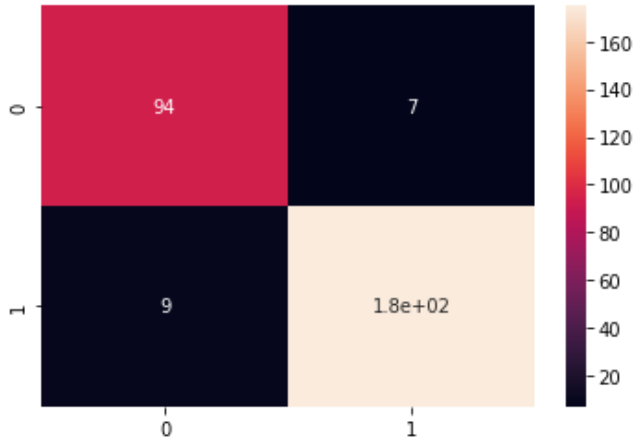
```
Accuracy: 94.38596491228071%

Confusion Matrix:
[[ 94   7]
 [  9 175]]

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.93      0.92       101
           1       0.96      0.95      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 [191]:

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

Out[191]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa14900e710>
```



train size : test size = 40% : 60%

In [192]:

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

In [193]:

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

```
227
342
```

In [194]:

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

Out[194]:

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

In [195]:

```
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))
```

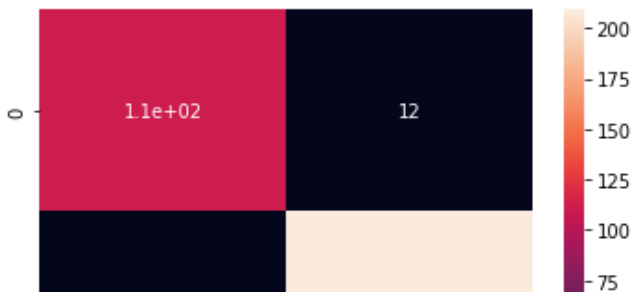
```
Accuracy: 93.56725146198829%

Confusion Matrix:

[[111  12]
 [ 10 209]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.90      0.91       123
           1       0.95      0.95      0.95       219

    accuracy                           0.94       342
   macro avg       0.93      0.93      0.93       342
weighted avg       0.94      0.94      0.94       342
```

In [196]:

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

Out[196]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa148f90c50>
```



train size : test size = 30% : 70%

In [197]:

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

In [198]:

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

```
170
399
```

In [199]:

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

Out[199]:

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

In [200]:

```
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))
```

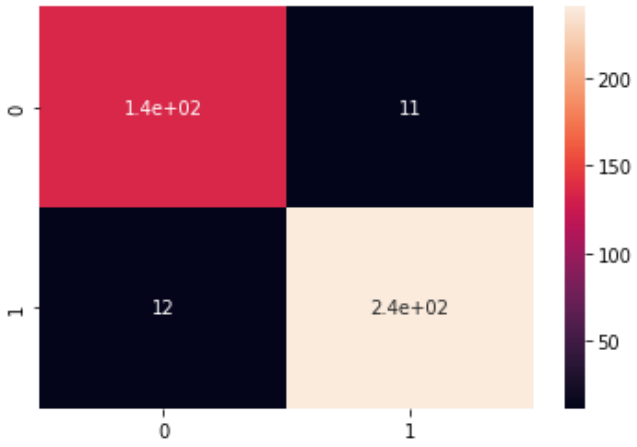
```
Accuracy: 94.23558897243107%

Confusion Matrix:

[[135  11]
 [ 12 241]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.92      0.92       146
           1       0.96      0.95      0.95       253

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

In [201]:

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

Out[201]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa148ea6350>
```


1.4e+02
11
12
2.4e+02
0
1
0
1
200
150
100
50

Name : Anmol Raj

Roll no :001811001069

IONOSPHERE

Import required modules

In [ ]:

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

Load Dataset

In [ ]:

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

Out[ ]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_k	colu
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	-0.1
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.6
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.0
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.2

In [ ]:

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

Out[ ]:

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

DataFrame ready to perform

In [ ]:

```
len(df)
```

Out[ ]:

```
351
```

In [ ]:

```
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 [ ]:

```
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='linear',
    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, random_state=0)
# 70% training data, 30% testing data
```

In [ ]:

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

```
245
106
```

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='linear',
    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))
```

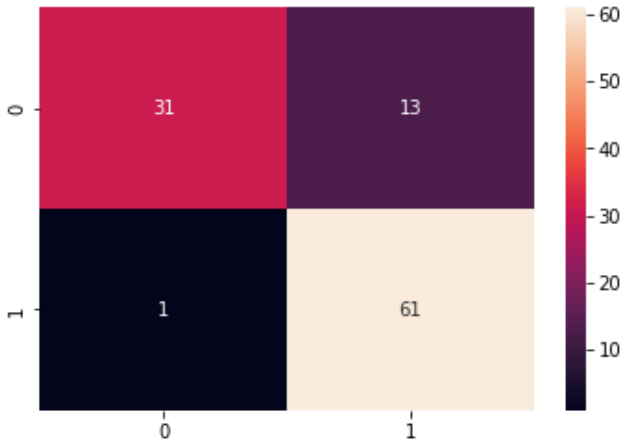
```
Accuracy: 86.79245283018868%

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 [ ]:

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

Out[ ]:

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



train size : test size = 60% : 40%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [ ]:

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

```
210
141
```

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='linear',
    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))
```

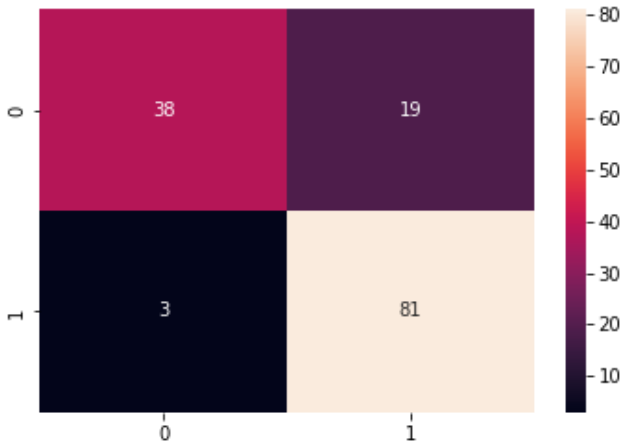
```
Accuracy: 84.39716312056737%

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 [ ]:

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

Out[ ]:

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



train size : test size = 50% : 50%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [ ]:

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

```
175
176
```

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='linear',
    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))
```

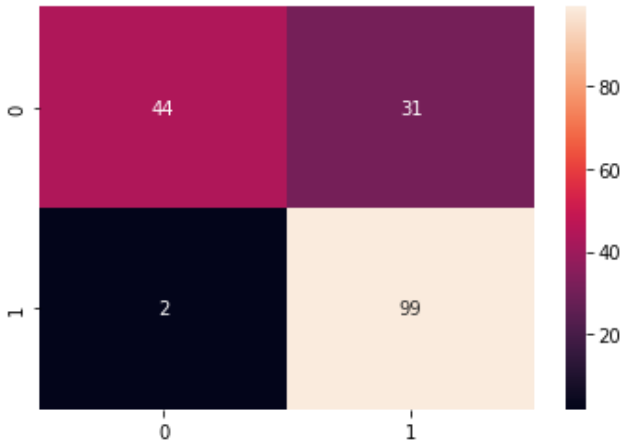
```
Accuracy: 81.25%

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 [ ]:

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

Out[ ]:

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



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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='linear',
    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))
```

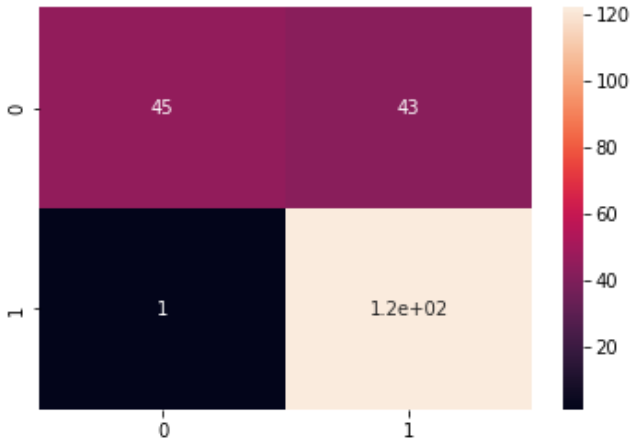
```
Accuracy: 79.14691943127961%

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 [ ]:

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

Out[ ]:

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



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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='linear',
    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))
```

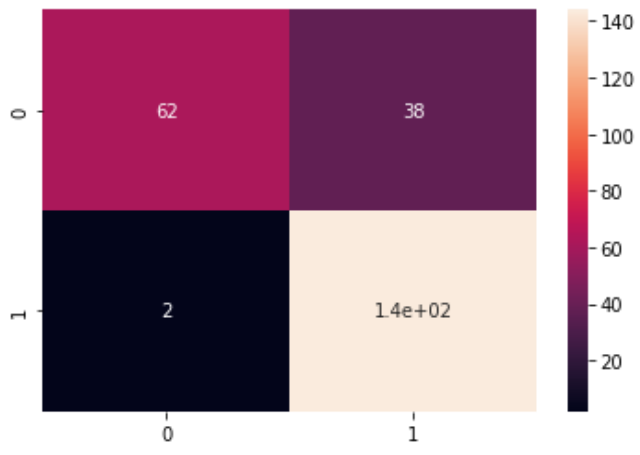
```
Accuracy: 83.73983739837398%

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 [ ]:

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

Out[ ]:

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



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, random_state=0)
```

In [ ]:

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

```
245
106
```

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))
```

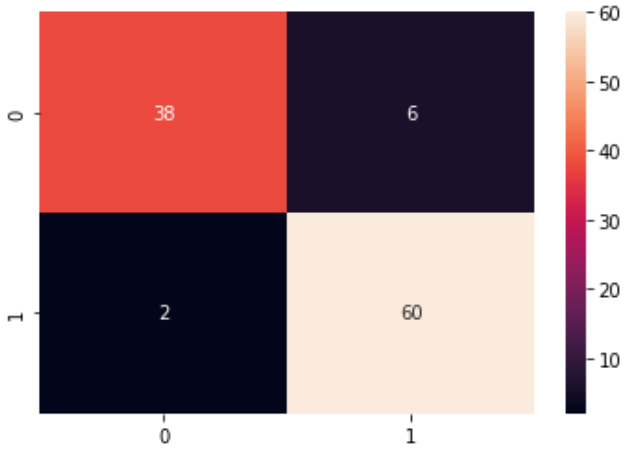
```
Accuracy: 92.45283018867924%

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 [ ]:

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

Out[ ]:

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



train size : test size = 60% : 40%

In [ ]:

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

In [ ]:

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

```
210
141
```

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))
```

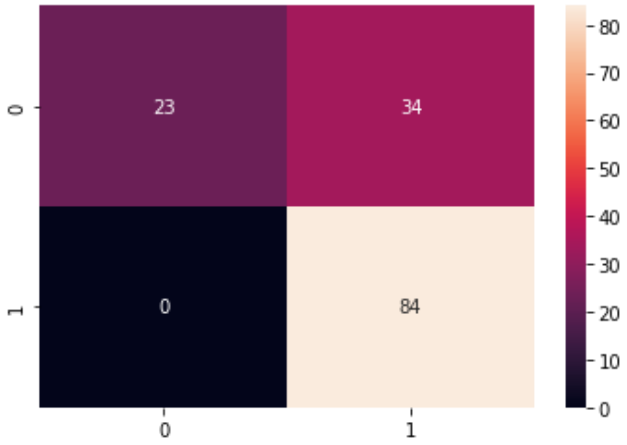
```
Accuracy: 75.88652482269504%

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 [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9d0a3d0>
```



train size : test size = 50% : 50%

In [ ]:

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

In [ ]:

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

```
175
176
```

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: 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
```

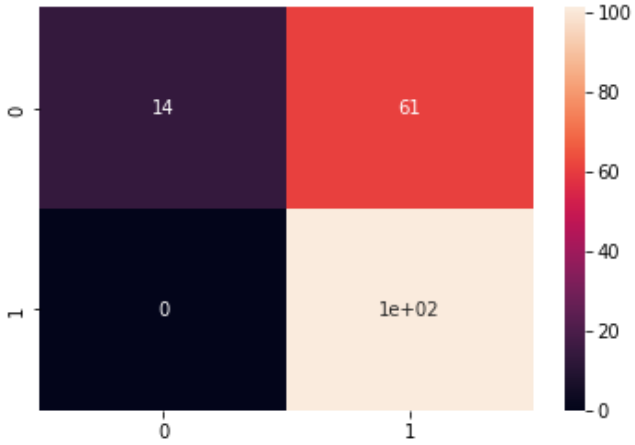
```
    accuracy                           0.65       176
   macro avg       0.81      0.59      0.54       176
weighted avg       0.78      0.65      0.57       176
```

In [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9c31bd0>
```



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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: 63.507109004739334%

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

Classification Report:
```

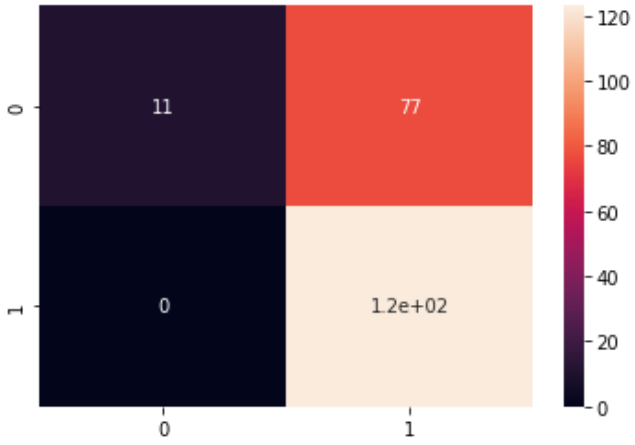
```
              precision    recall  f1-score   support

           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 [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9c8f610>
```



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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: 63.82113821138211%

Confusion Matrix:
 [[ 11  89]
```

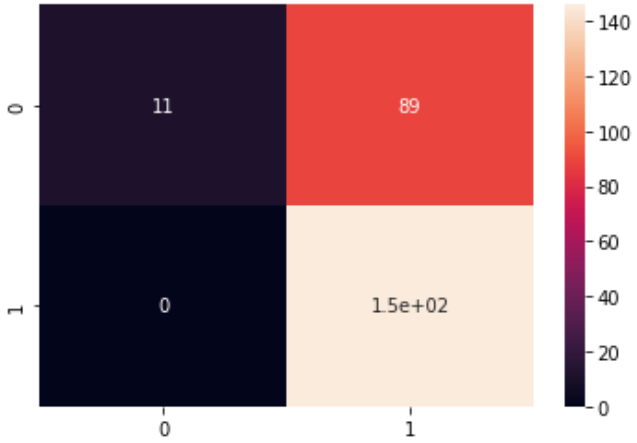
```
 [  0 146]]

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 [ ]:

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

Out[ ]:

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



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, random_state=0)
```

In [ ]:

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

```
245
106
```

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))
```

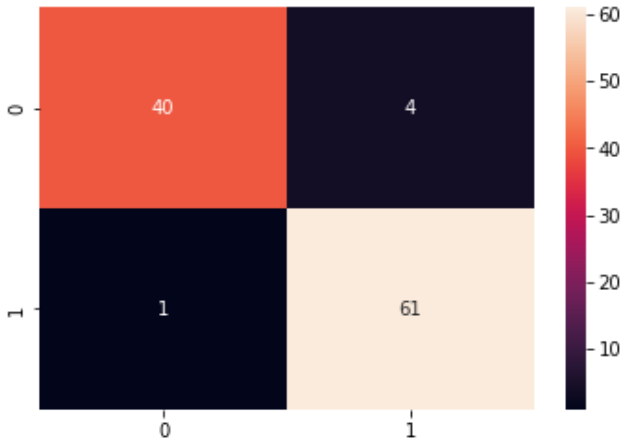
```
Accuracy: 95.28301886792453%

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 [ ]:

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

Out[ ]:

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



train size : test size = 60% : 40%

In [ ]:

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

In [ ]:

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

```
210
141
```

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))
```

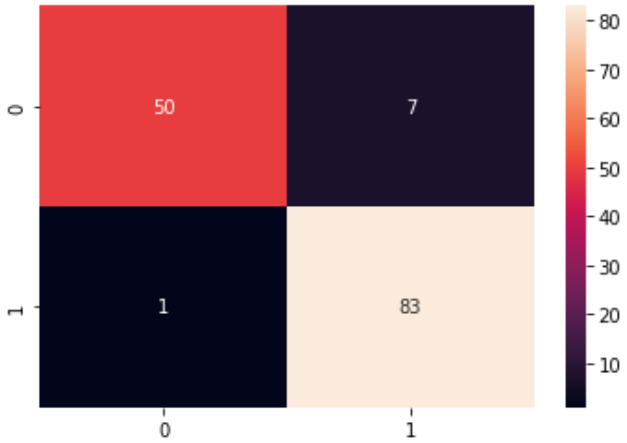
```
Accuracy: 94.32624113475178%

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 [ ]:

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

Out[ ]:

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



train size : test size = 50% : 50%

In [ ]:

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

In [ ]:

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

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

```
175
176
```

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))
```

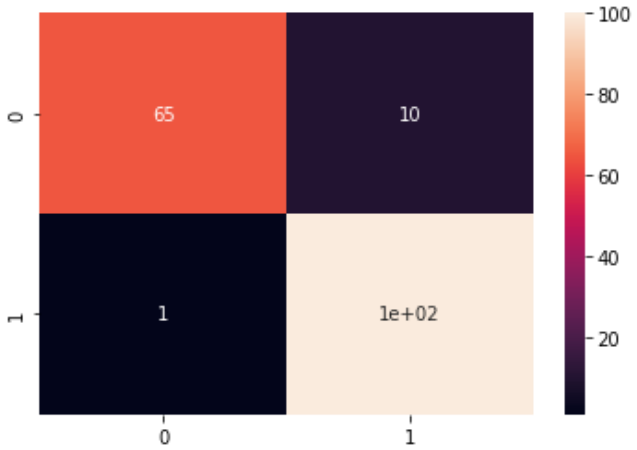
```
Accuracy: 93.75%

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 [ ]:

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

Out[ ]:

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



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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))
```

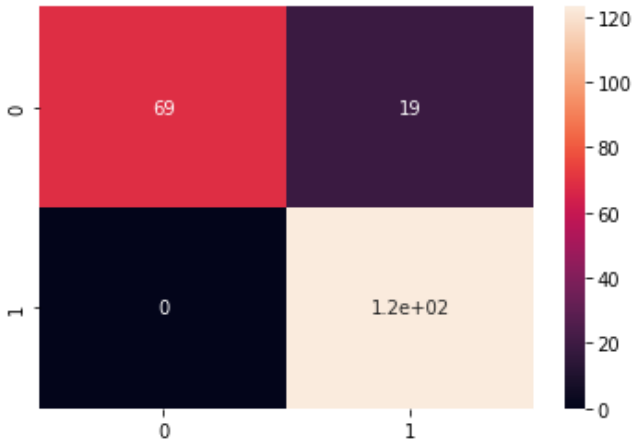
```
Accuracy: 90.99526066350711%

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 [ ]:

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

Out[ ]:

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



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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: 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 [ ]:

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

Out[ ]:

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

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='sigmoid',
    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, random_state=0)
```

In [ ]:

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

```
245
106
```

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='sigmoid',
    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))
```

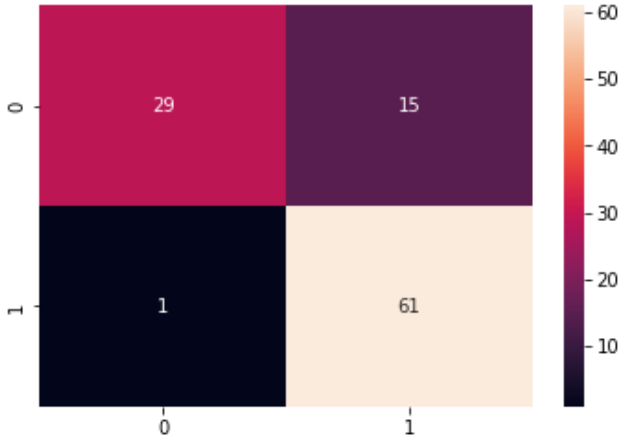
```
Accuracy: 84.90566037735849%

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 [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc96dcdd0>
```



train size : test size = 60% : 40%

In [ ]:

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

In [ ]:

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

```
210
141
```

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='sigmoid',
    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: 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
```

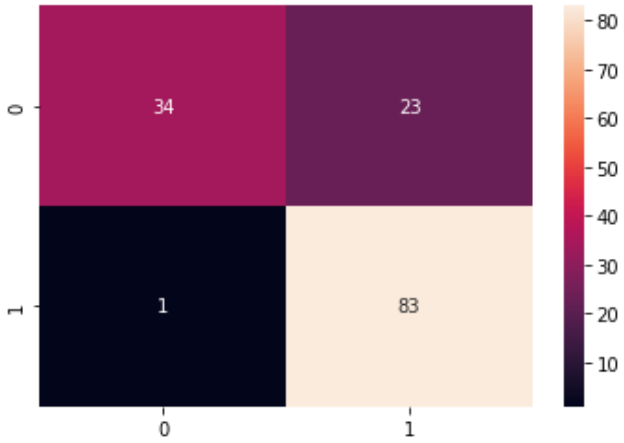
```
   macro avg       0.88      0.79      0.81       141
weighted avg       0.86      0.83      0.82       141
```

In [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fc9673850>
```



train size : test size = 50% : 50%

In [ ]:

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

In [ ]:

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

```
175
176
```

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='sigmoid',
    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: 83.52272727272727%

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

Classification Report:

              precision    recall  f1-score   support
```

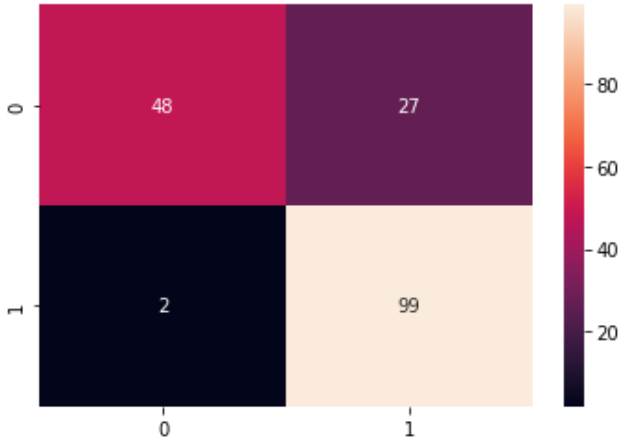
```
           b       0.96      0.64      0.77        75
           g       0.79      0.98      0.87       101

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

In [ ]:

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

Out[ ]:

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



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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='sigmoid',
    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: 81.99052132701422%

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

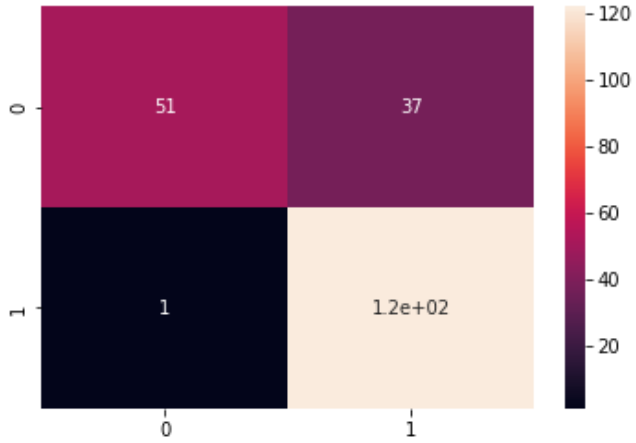
```
Classification Report:

              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 [ ]:

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

Out[ ]:

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



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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='sigmoid',
    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: 82.11382113821138%
```

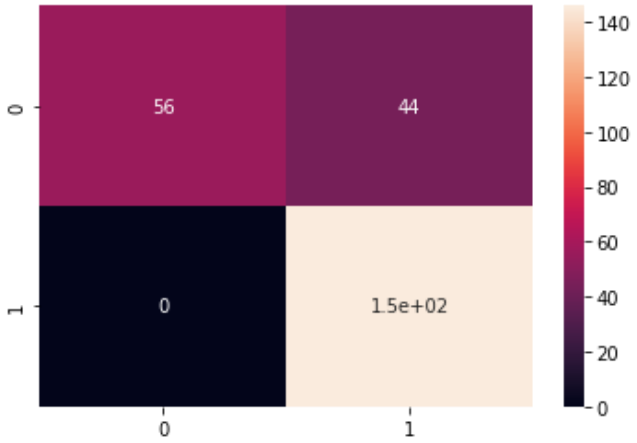
```
Confusion Matrix:
 [[ 56  44]
 [  0 146]]

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 [ ]:

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

Out[ ]:

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



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, random_state=0)
# 70% training data, 30% testing data
```

In [ ]:

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

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

```
245
106
```

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))
```

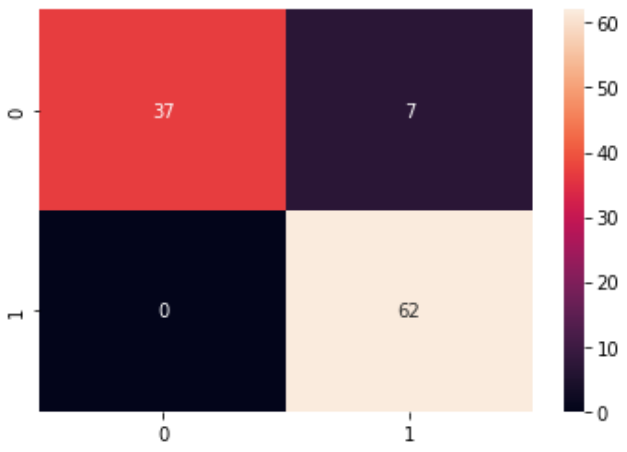
```
Accuracy: 93.39622641509435%

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 [ ]:

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

Out[ ]:

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



train size : test size = 60% : 40%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [ ]:

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

```
210
141
```

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))
```

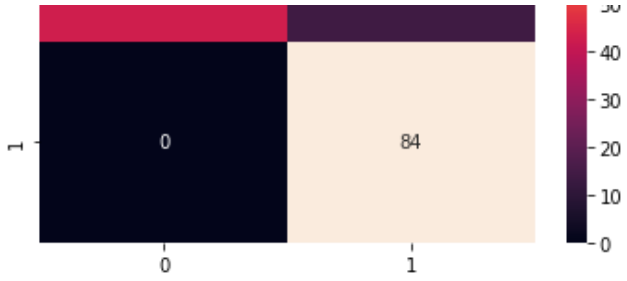
```
Accuracy: 90.0709219858156%

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 [ ]:

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

Out[ ]:

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



train size : test size = 50% : 50%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [ ]:

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

```
175
176
```

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))
```

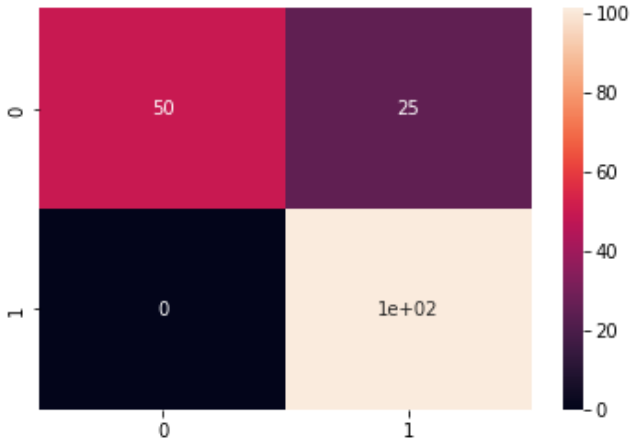
```
Accuracy: 85.79545454545455%

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 [ ]:

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

Out[ ]:

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



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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: 84.36018957345972%

Confusion Matrix:

[[ 56  32]
 [  1 122]]

Classification Report:

              precision    recall  f1-score   support
```

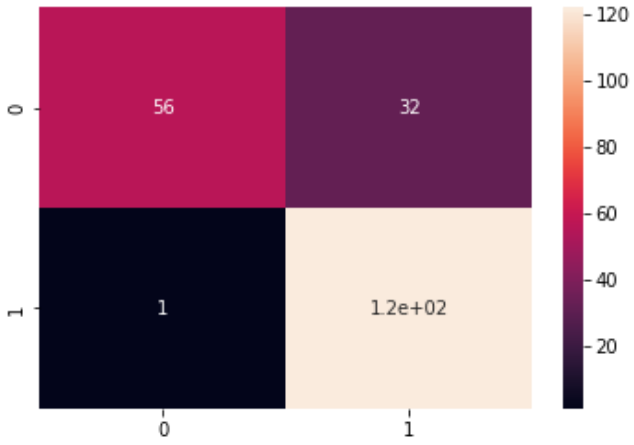
```
           b       0.98      0.64      0.77        88
           g       0.79      0.99      0.88       123

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

In [ ]:

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

Out[ ]:

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



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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: 84.5528455284553%
```

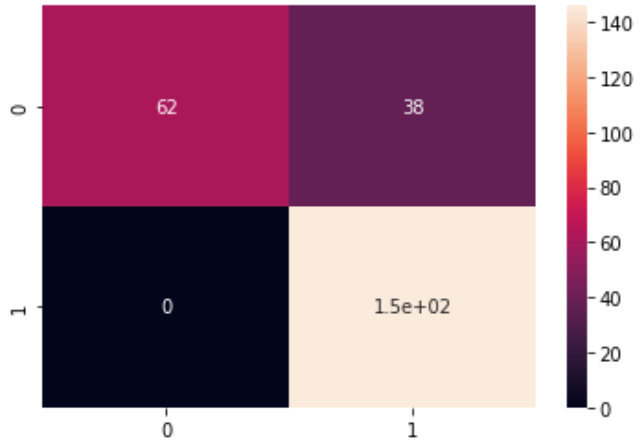
```
Confusion Matrix:

[[ 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 [ ]:

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

Out[ ]:

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



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

train size : test size = 70% : 30%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [ ]:

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

```
245
106
```

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

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))
```

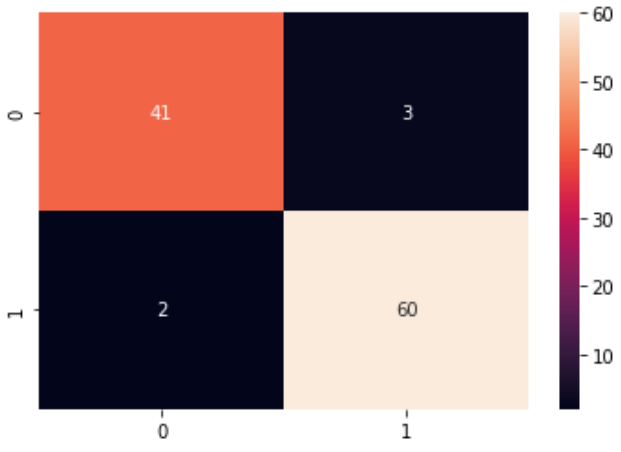
```
Accuracy: 95.28301886792453%

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 [ ]:

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

Out[ ]:

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



train size : test size = 60% : 40%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [ ]:

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

```
210
141
```

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

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))
```

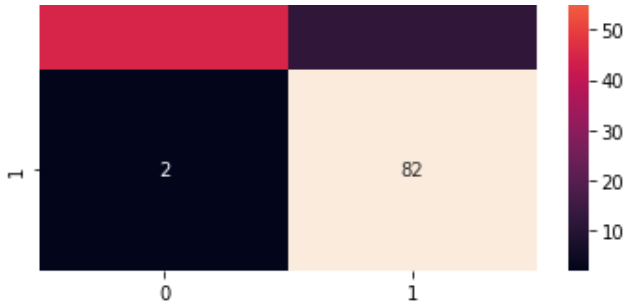
```
Accuracy: 90.0709219858156%

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 [ ]:

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

Out[ ]:

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



train size : test size = 50% : 50%

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [ ]:

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

```
175
176
```

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

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))
```

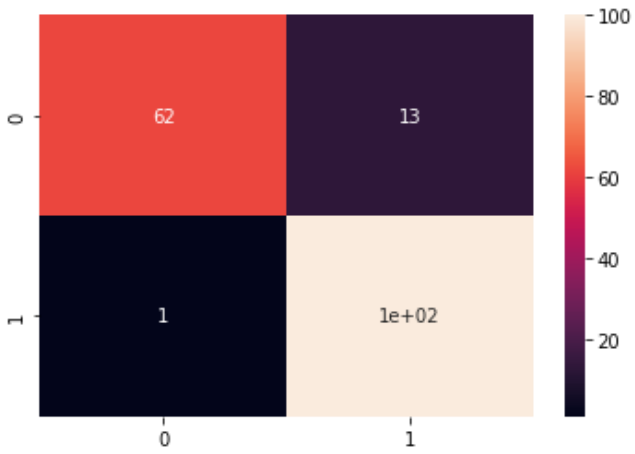
```
Accuracy: 92.04545454545455%

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 [ ]:

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

Out[ ]:

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



train size : test size = 40% : 60%

In [ ]:

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

In [ ]:

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

```
140
211
```

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

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: 90.04739336492891%

Confusion Matrix:

[[ 75  13]
 [  8 115]]

Classification Report:

              precision    recall  f1-score   support
```

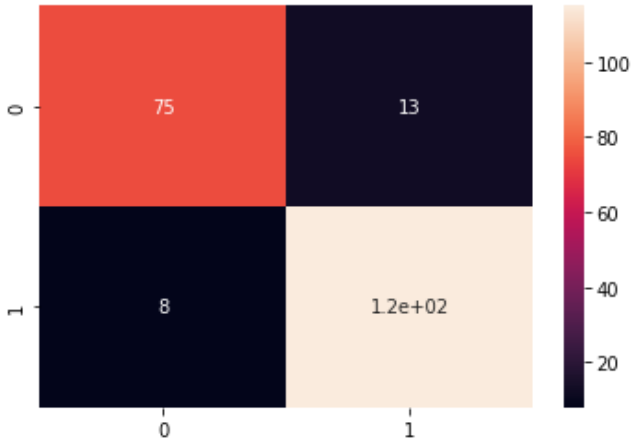
```
           b       0.90      0.85      0.88        88
           g       0.90      0.93      0.92       123

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

In [ ]:

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

Out[ ]:

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



train size : test size = 30% : 70%

In [ ]:

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

In [ ]:

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

```
105
246
```

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

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))
```

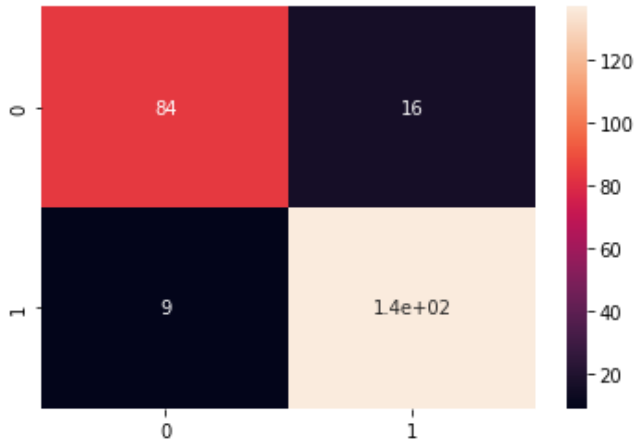
```
Accuracy: 89.83739837398373%

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 [ ]:

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

Out[ ]:

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



Name : Anmol Raj

Roll no : 001811001069

WINE DATASET

Importing modules

In [1]:

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

Load Dataset

In [2]:

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

In [3]:

```
dir(wine)
```

Out[3]:

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

In [4]:

```
wine.data
```

Out[4]:

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

In [5]:

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

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flav
```

```
anoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od31
5_of_diluted_wines', 'proline']
['class_0' 'class_1' 'class_2']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

In [6]:

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

Out[6]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	c
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	

In [7]:

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

Out[7]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	c
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	

In [8]:

```
wine.target_names
```

Out[8]:

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

DataFrame ready to perform

In [9]:

```
len(df)
```

Out[9]:

```
178
```

In [10]:

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

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

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

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

SVC Classfier

Linear SVC Classifier

In [11]:

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

Out[11]:

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

train size : test size = 70% : 30%

In [12]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [13]:

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

```
124
54
```

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

```
Accuracy: 98.14814814814815%

Confusion Matrix:
```

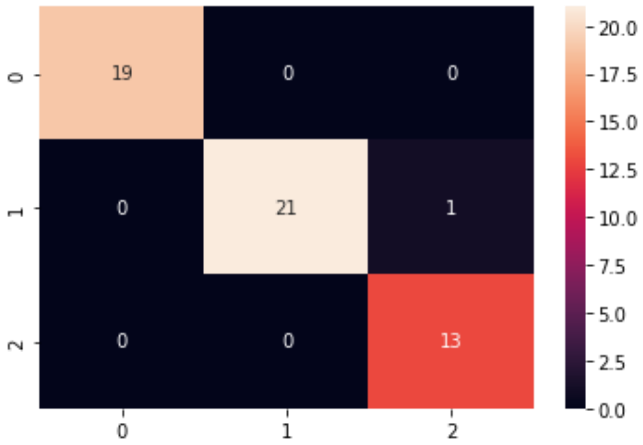
```
[[19  0  0]
 [ 0 21  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

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

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

In [16]:

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

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e43120550>
```



train size : test size = 60% : 40%

In [17]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [18]:

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

```
106
72
```

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='linear',
    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))
```

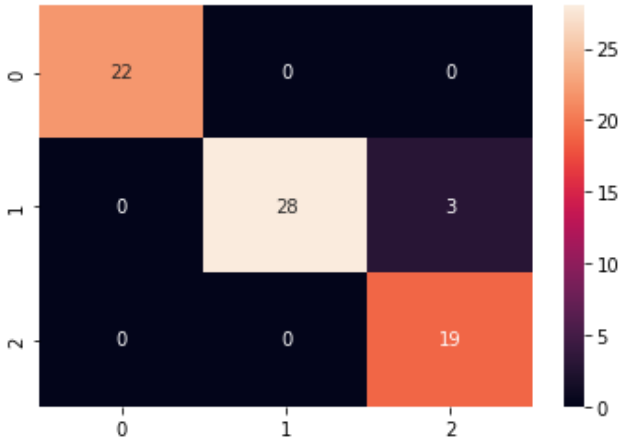
```
Accuracy: 95.83333333333334%

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [21]:

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

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20c82a50>
```



train size : test size = 50% : 50%

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [23]:

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

```
89
89
```

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

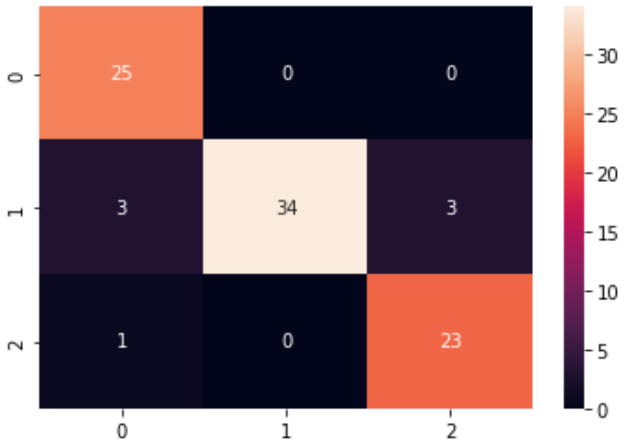
```
Accuracy: 92.13483146067416%

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [26]:

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

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20c2b590>
```



train size : test size = 40% : 60%

In [27]:

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

In [28]:

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

```
71
107
```

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='linear',
    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))
```

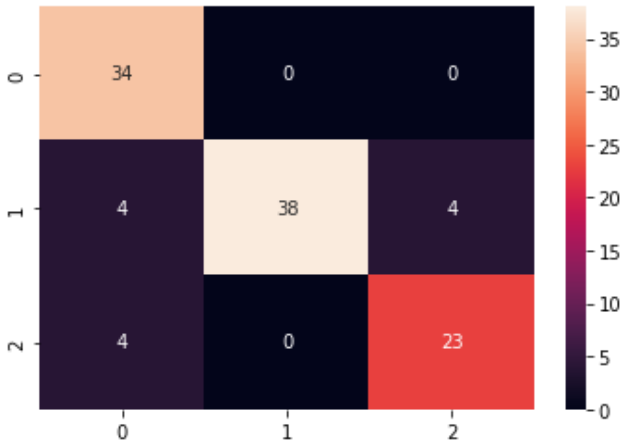
```
Accuracy: 88.78504672897196%

Confusion Matrix:

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [31]:

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

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20b034d0>
```



train size : test size = 30% : 70%

In [32]:

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

In [33]:

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

```
53
125
```

In [34]:

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

Out[34]:

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

In [35]:

```
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))
```

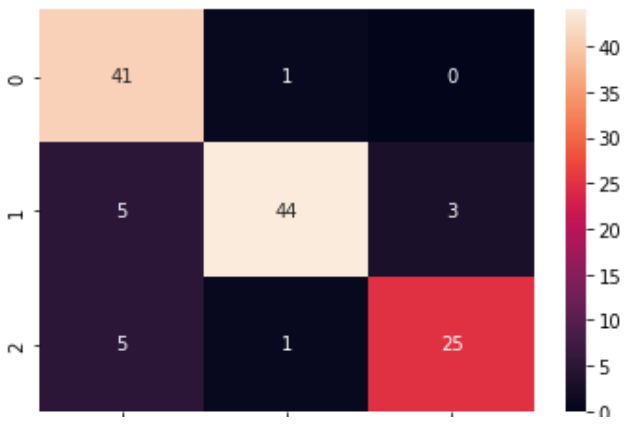
```
Accuracy: 88.0%

Confusion Matrix:

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [36]:

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

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20a402d0>
```



0 1 2

Polynomial SVC Classifier

In [37]:

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

Out[37]:

```
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 [38]:

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

In [39]:

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

```
124
54
```

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))
```

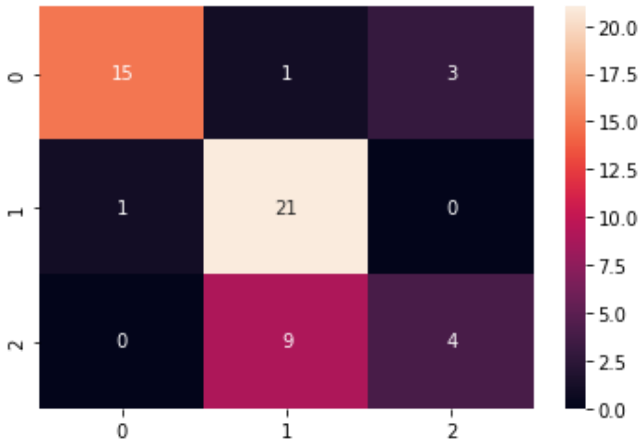
```
Accuracy: 74.07407407407408%

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

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.79      0.86        19
           1       0.68      0.95      0.79        22
           2       0.57      0.31      0.40        13

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

In [42]:

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

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e209ea350>
```



train size : test size = 60% : 40%

In [43]:

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

In [44]:

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

```
106
72
```

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: 65.27777777777779%

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

Classification Report:

              precision    recall  f1-score   support

           0       0.90      0.82      0.86        22
           1       0.56      0.94      0.70        31
```

```
           2       0.00      0.00      0.00        19

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

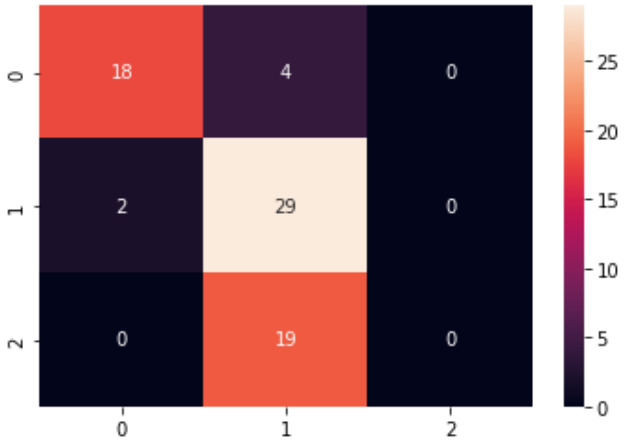
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [47]:

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

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e208806d0>
```



train size : test size = 50% : 50%

In [48]:

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

In [49]:

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

```
89
89
```

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: 70.78651685393258%
```

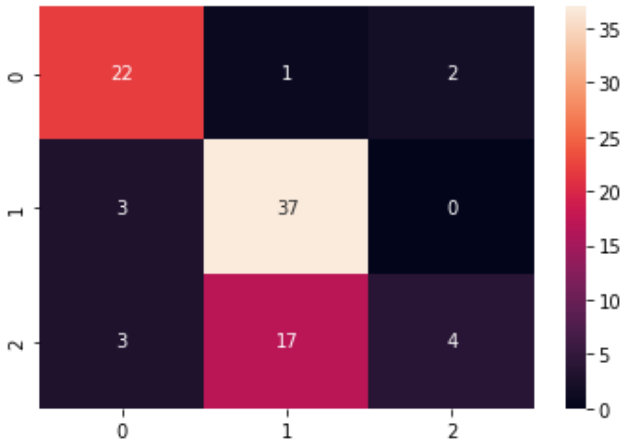
```
Confusion Matrix:
 [[22  1  2]
 [ 3 37  0]
 [ 3 17  4]]

Classification Report:

              precision    recall  f1-score   support

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

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

In [52]:

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

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20828dd0>
```



train size : test size = 40% : 60%

In [53]:

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

In [54]:

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

```
71
107
```

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))
```

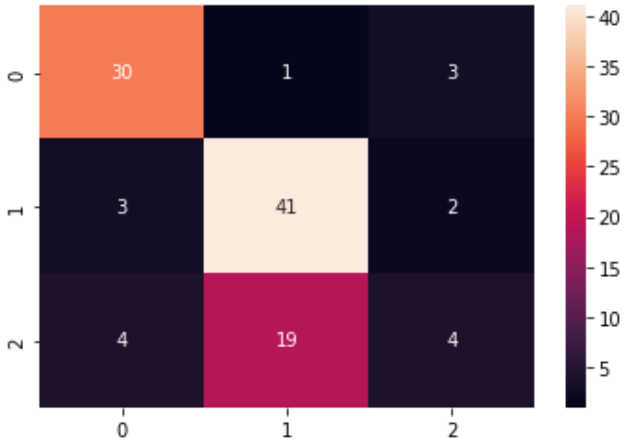
```
Accuracy: 70.09345794392523%

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [57]:

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

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20763e10>
```



train size : test size = 30% : 70%

In [58]:

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

In [59]:

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

```
53
125
```

In [60]:

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

Out[60]:

```
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 [61]:

```
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))
```

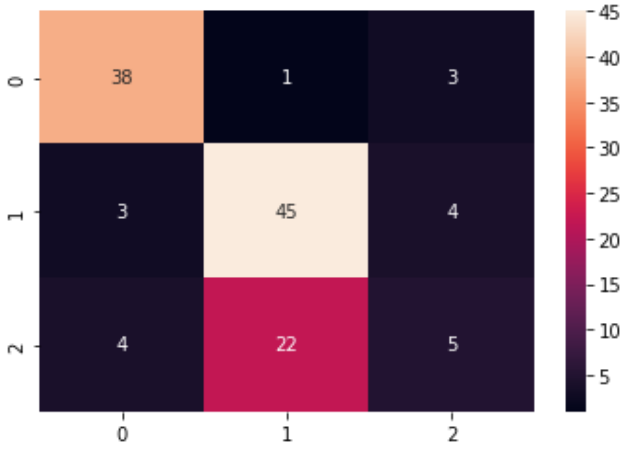
```
Accuracy: 70.39999999999999%

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [62]:

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

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e2069f6d0>
```



Gaussain SVC Classifier

In [63]:

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

Out[63]:

```
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 [64]:

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

In [65]:

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

```
124
54
```

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))
```

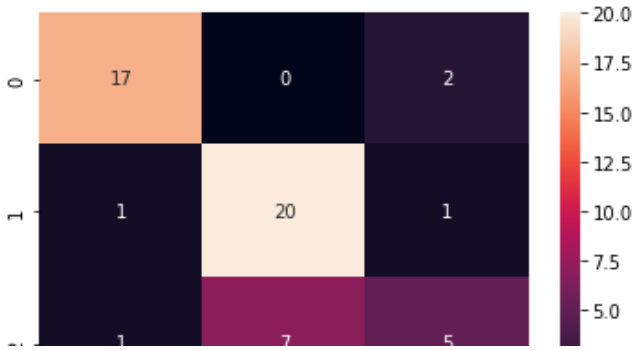
```
Accuracy: 77.77777777777779%

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

Classification Report:

              precision    recall  f1-score   support

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

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

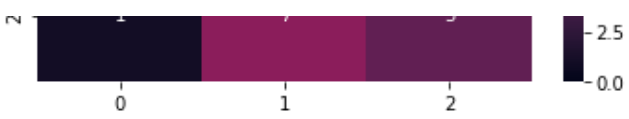
In [68]:

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

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e205db9d0>
```





train size : test size = 60% : 40%

In [69]:

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

In [70]:

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

```
106
72
```

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: 68.05555555555556%

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

Classification Report:

              precision    recall  f1-score   support

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

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

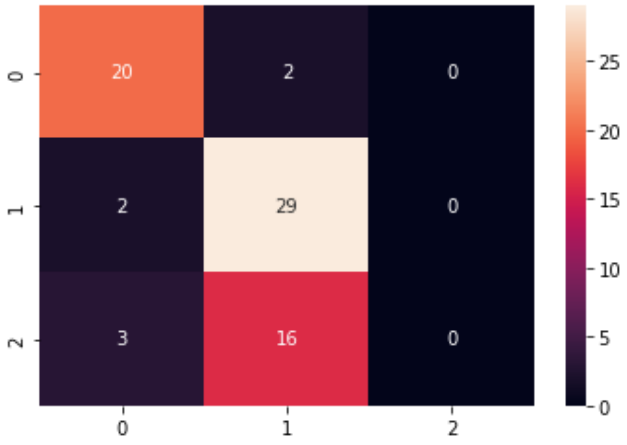
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [73]:

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

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e2051e450>
```



train size : test size = 50% : 50%

In [74]:

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

In [75]:

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

```
89
89
```

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))
```

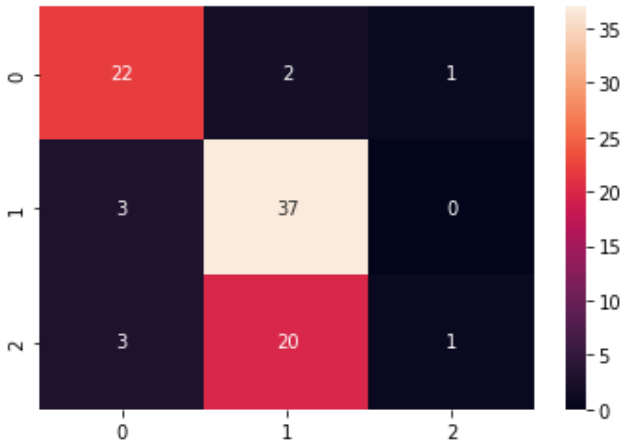
```
Accuracy: 67.41573033707866%

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

Classification Report:

              precision    recall  f1-score   support

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

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

In [78]:

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

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e205651d0>
```



train size : test size = 40% : 60%

In [79]:

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

In [80]:

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

```
71
107
```

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: 71.96261682242991%

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

Classification Report:

              precision    recall  f1-score   support

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

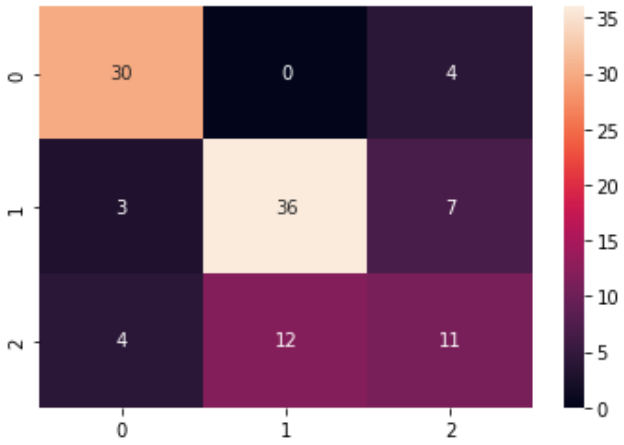
```
    accuracy                           0.72       107
   macro avg       0.69      0.69      0.69       107
weighted avg       0.71      0.72      0.71       107
```

In [83]:

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

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20900d90>
```



train size : test size = 30% : 70%

In [84]:

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

In [85]:

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

```
53
125
```

In [86]:

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

Out[86]:

```
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 [87]:

```
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: 72.0%

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

Classification Report:
```

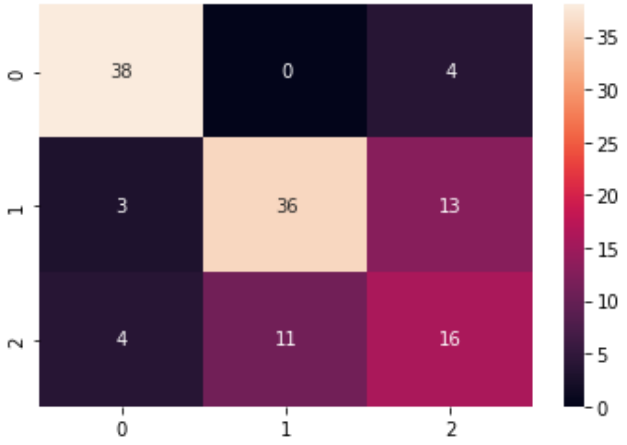
```
              precision    recall  f1-score   support

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

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

In [88]:

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

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e2030e2d0>
```



Sigmoid SVC Classifier

In [89]:

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

Out[89]:

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

train size : test size = 70% : 30%

In [90]:

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

In [91]:

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

```
124
54
```

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='sigmoid',
    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: 20.37037037037037%

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

Classification Report:

              precision    recall  f1-score   support

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

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

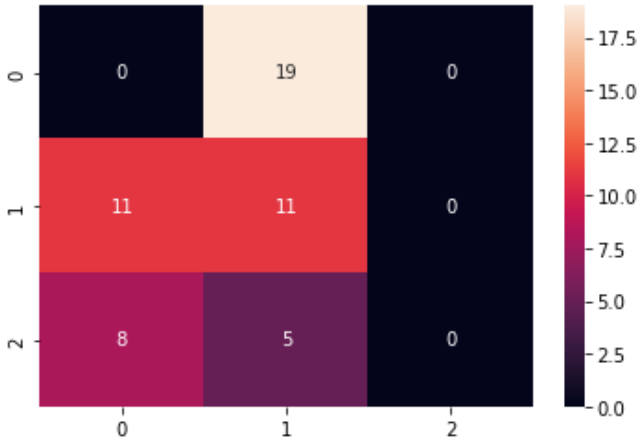
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [94]:

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

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20256410>
```



train size : test size = 60% : 40%

In [95]:

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

In [96]:

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

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

```
106
72
```

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='sigmoid',
    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: 18.055555555555554%

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

Classification Report:

              precision    recall  f1-score   support

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

    accuracy                           0.18        72
   macro avg       0.12      0.15      0.13        72
weighted avg       0.15      0.18      0.16        72
```

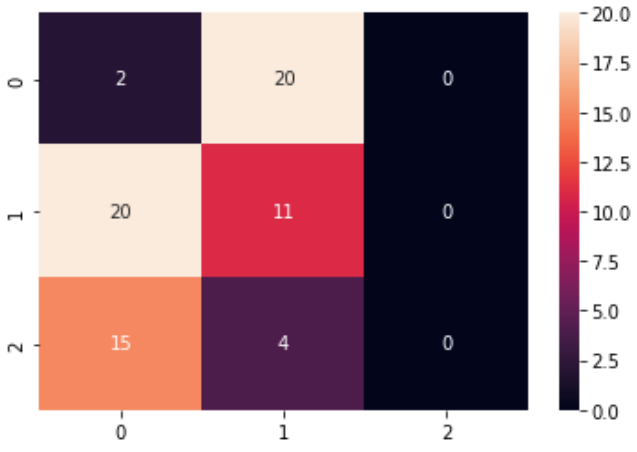
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [99]:

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

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e201a1550>
```



train size : test size = 50% : 50%

In [100]:

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

In [101]:

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

```
89
89
```

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='sigmoid',
    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: 17.97752808988764%

Confusion Matrix:
 [[ 7 18  0]
 [31  9  0]
 [24  0  0]]

Classification Report:

              precision    recall  f1-score   support

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

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

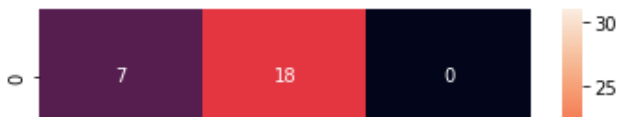
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

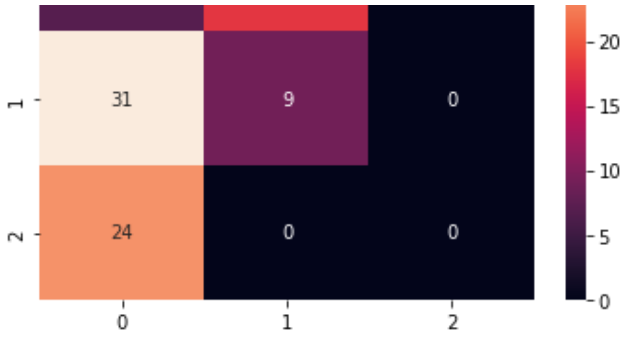
In [104]:

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

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e200d1a50>
```





train size : test size = 40% : 60%

In [105]:

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

In [106]:

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

```
71
107
```

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='sigmoid',
    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: 15.887850467289718%

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

Classification Report:

              precision    recall  f1-score   support

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

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

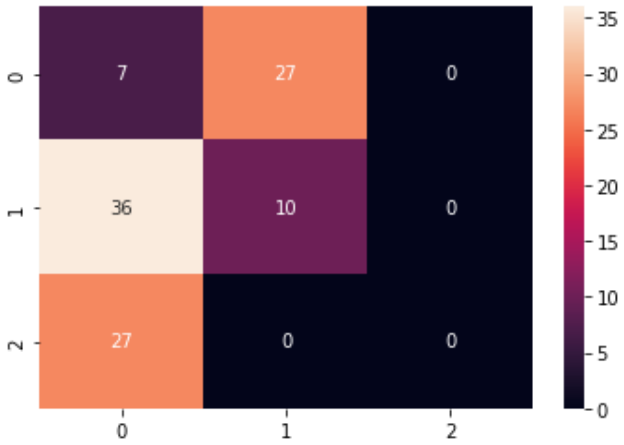
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [109]:

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

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e20016910>
```



train size : test size = 30% : 70%

In [110]:

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

In [111]:

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

```
53
125
```

In [112]:

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

Out[112]:

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

In [113]:

```
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: 41.6%

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

Classification Report:

              precision    recall  f1-score   support

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

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

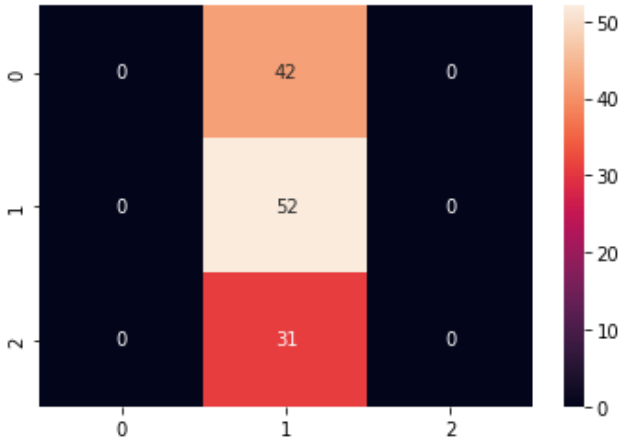
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [114]:

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

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1ff59190>
```



MLP Classifier

In [115]:

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

Out[115]:

```
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 [116]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [117]:

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

```
124
54
```

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

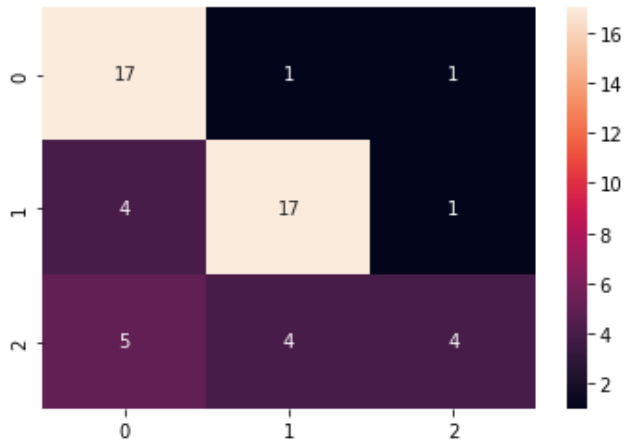
```
Accuracy: 70.37037037037037%

Confusion Matrix:

[[17  1  1]
 [ 4 17  1]
 [ 5  4  4]]

Classification Report:

              precision    recall  f1-score   support

           0       0.65      0.89      0.76        19
           1       0.77      0.77      0.77        22
           2       0.67      0.31      0.42        13

    accuracy                           0.70        54
   macro avg       0.70      0.66      0.65        54
weighted avg       0.71      0.70      0.68        54
```

In [120]:

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

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e208b7cd0>
```



train size : test size = 60% : 40%

In [121]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [122]:

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

```
106
72
```

In [123]:

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

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (600) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

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))
```

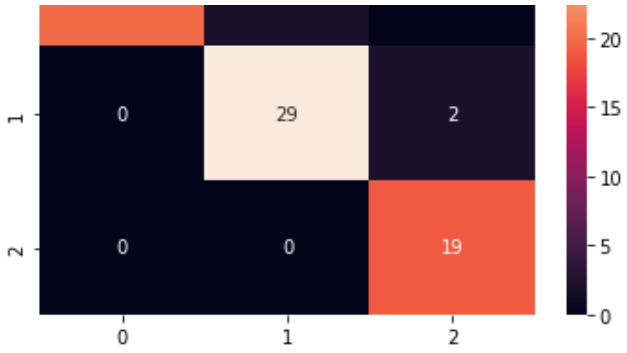
```
Accuracy: 94.44444444444444%

Confusion Matrix:
[[20  2  0]
 [ 0 29  2]
 [ 0  0 19]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      0.91      0.95        22
           1       0.94      0.94      0.94        31
           2       0.90      1.00      0.95        19

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

In [125]:

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

Out[125]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1fdcea10>
```



train size : test size = 50% : 50%

In [126]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [127]:

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

```
89
89
```

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

```
Accuracy: 26.96629213483146%

Confusion Matrix:
[[ 0  0 25]
 [ 0  0 40]
 [ 0  0 24]]

Classification Report:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00        25
           1       0.00      0.00      0.00        40
           2       0.27      1.00      0.42        24

    accuracy                           0.27        89
   macro avg       0.09      0.33      0.14        89
weighted avg       0.07      0.27      0.11        89
```

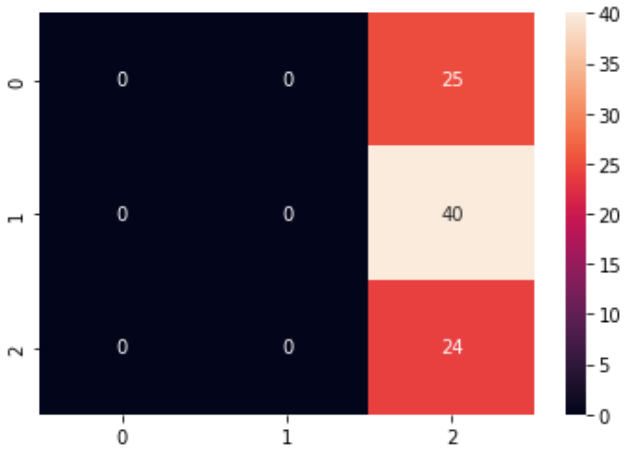
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [130]:

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

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1fd5ae90>
```



train size : test size = 40% : 60%

In [131]:

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

In [132]:

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

```
71
107
```

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: 40.18691588785047%
```

```
Confusion Matrix:

[[34  0  0]
 [19  0 27]
 [18  0  9]]

Classification Report:

              precision    recall  f1-score   support

           0       0.48      1.00      0.65        34
           1       0.00      0.00      0.00        46
           2       0.25      0.33      0.29        27

    accuracy                           0.40       107
   macro avg       0.24      0.44      0.31       107
weighted avg       0.22      0.40      0.28       107
```

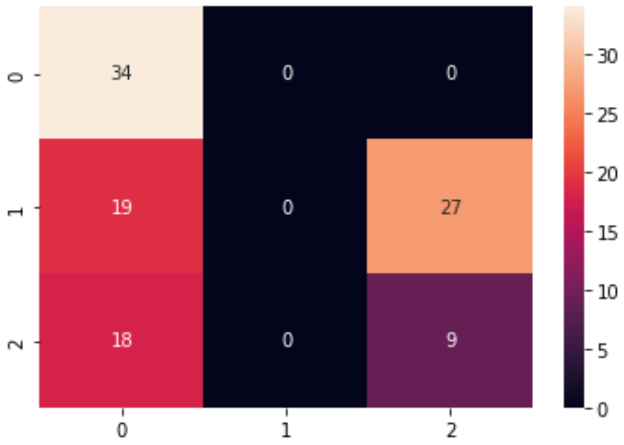
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [135]:

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

Out[135]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1fcf5790>
```



train size : test size = 30% : 70%

In [136]:

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

In [137]:

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

```
53
125
```

In [138]:

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

Out[138]:

```
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 [139]:

```
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: 15.2%

Confusion Matrix:

[[ 2 40  0]
 [35 17  0]
 [14 17  0]]

Classification Report:

              precision    recall  f1-score   support

           0       0.04      0.05      0.04        42
           1       0.23      0.33      0.27        52
           2       0.00      0.00      0.00        31

    accuracy                           0.15       125
   macro avg       0.09      0.12      0.10       125
weighted avg       0.11      0.15      0.13       125
```

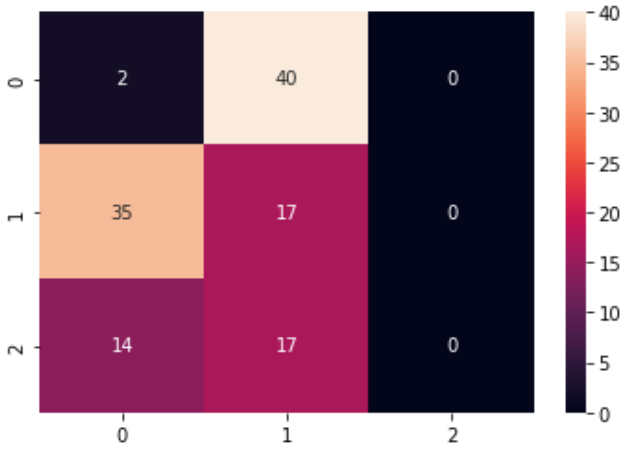
```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [140]:

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

Out[140]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e200c8810>
```



Random Forest Classifier

In [141]:

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

```
rfc_classifier
```

Out[141]:

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

train size : test size = 70% : 30%

In [142]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# 70% training data, 30% testing data
```

In [143]:

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

```
124
54
```

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

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

```
Accuracy: 96.29629629629629%

Confusion Matrix:

[[19  0  0]
 [ 1 20  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      1.00      0.97        19
           1       1.00      0.91      0.95        22
           2       0.93      1.00      0.96        13

    accuracy                           0.96        54
   macro avg       0.96      0.97      0.96        54
```

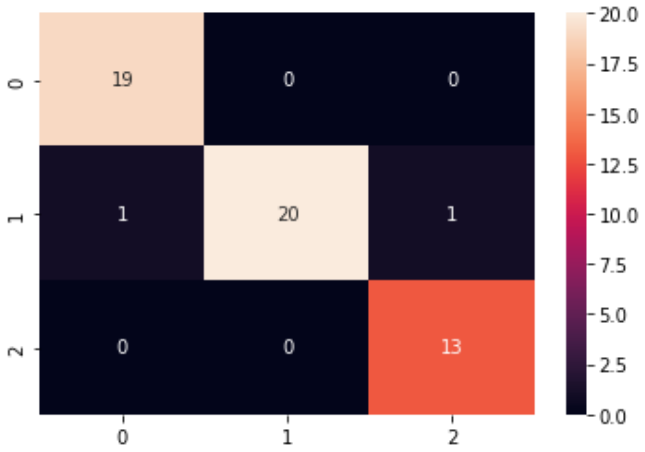
```
weighted avg       0.97      0.96      0.96        54
```

In [146]:

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

Out[146]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1fcb77d0>
```



train size : test size = 60% : 40%

In [147]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
# 60% training data, 40% testing data
```

In [148]:

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

```
106
72
```

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

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: 97.22222222222221%

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

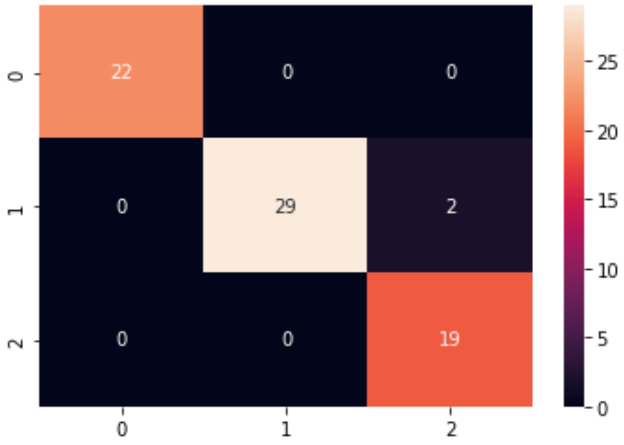
```
 [ 0  0 19]]

Classification Report:

              precision    recall  f1-score   support

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

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

In [151]:

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

Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1faabc10>
```



train size : test size = 50% : 50%

In [152]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
# 50% training data, 50% testing data
```

In [153]:

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

```
89
89
```

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

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

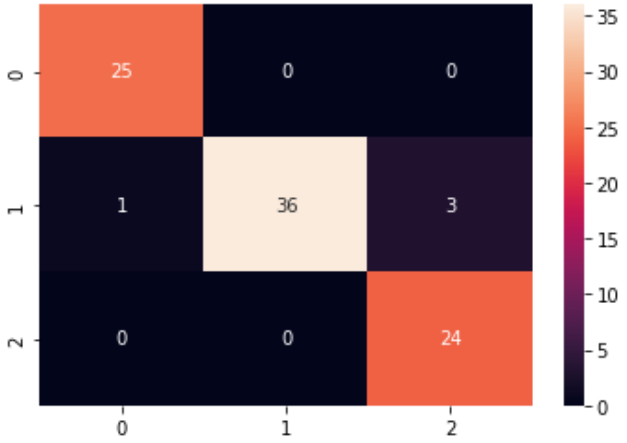
```
Accuracy: 95.50561797752809%

Confusion Matrix:
[[25  0  0]
 [ 1 36  3]
 [ 0  0 24]]

Classification Report:

              precision    recall  f1-score   support

           0       0.96      1.00      0.98        25
           1       1.00      0.90      0.95        40
           2       0.89      1.00      0.94        24

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

In [156]:

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

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1f9ece10>
```



train size : test size = 40% : 60%

In [157]:

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

In [158]:

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

```
71
107
```

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

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))
```

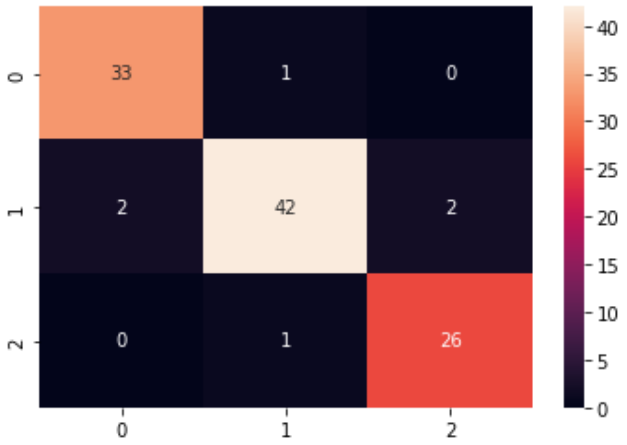
```
Accuracy: 94.39252336448598%

Confusion Matrix:

[[33  1  0]
 [ 2 42  2]
 [ 0  1 26]]

Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.97      0.96        34
           1       0.95      0.91      0.93        46
           2       0.93      0.96      0.95        27

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

In [161]:

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

Out[161]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1f92a690>
```



train size : test size = 30% : 70%

In [162]:

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

In [163]:

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

```
53
125
```

In [164]:

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

Out[164]:

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

In [165]:

```
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))
```

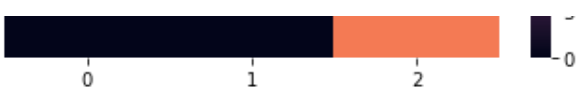
```
Accuracy: 92.80000000000001%

Confusion Matrix:

[[42  0  0]
 [ 2 43  7]
 [ 0  0 31]]

Classification Report:

              precision    recall  f1-score   support

           0       0.95      1.00      0.98        42
           1       1.00      0.83      0.91        52
           2       0.82      1.00      0.90        31

    accuracy                           0.93       125
   macro avg       0.92      0.94      0.93       125
weighted avg       0.94      0.93      0.93       125
```

In [166]:

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

Out[166]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e1f867f50>
```


0
1
2
0