Muhammad Arham Adeel

Naive Bayes Algorithm

Practice and task given in the video on phool dataset.

In [1]:
```
#importing libraries
import pandas as pd
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset("iris")
df.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:
```
x = df.iloc[ : , : -1]
y = df.iloc[ : , -1 : ]
```

In [4]:
```
x.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
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 [5]:
```
y.head()
```

Out[5]:

	species
0	setosa
1	setosa
2	setosa
3	setosa

	species
4	setosa

In [8]:
```
from sklearn.naive_bayes import GaussianNB

model = GaussianNB().fit(x,y)
model
```

```
H:\download\Anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  return f(*args, **kwargs)
```

Out[8]:
```
GaussianNB()
```

In [7]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=0)
```

In [10]:
```
# training the model on training dataset

from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(X_train,y_train)
model

# making prediction on data set

y_predict = model.predict(X_test)
y_predict
```

```
H:\download\Anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  return f(*args, **kwargs)
```

Out[10]:
```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa'], dtype='<U10')
```

In [21]:
```
#Accuracy test

score = model.score(X_test,y_test)
print("The accuracy score is = ",score)
```

```
The accuracy score is =  0.9666666666666667
```

In [13]:
```
#Checking Score

from sklearn import metrics

print("Guassian in Naive Base model accuracy : ", metrics.accuracy_score(y_test,y_predict))
```

```
Guassian in Naive Base model accuracy :  0.9666666666666667
```

In [17]:
```
print("Guassian in Naive Base model accuracy in % : " , metrics.accuracy_score(y_test,y_predict)*1
```

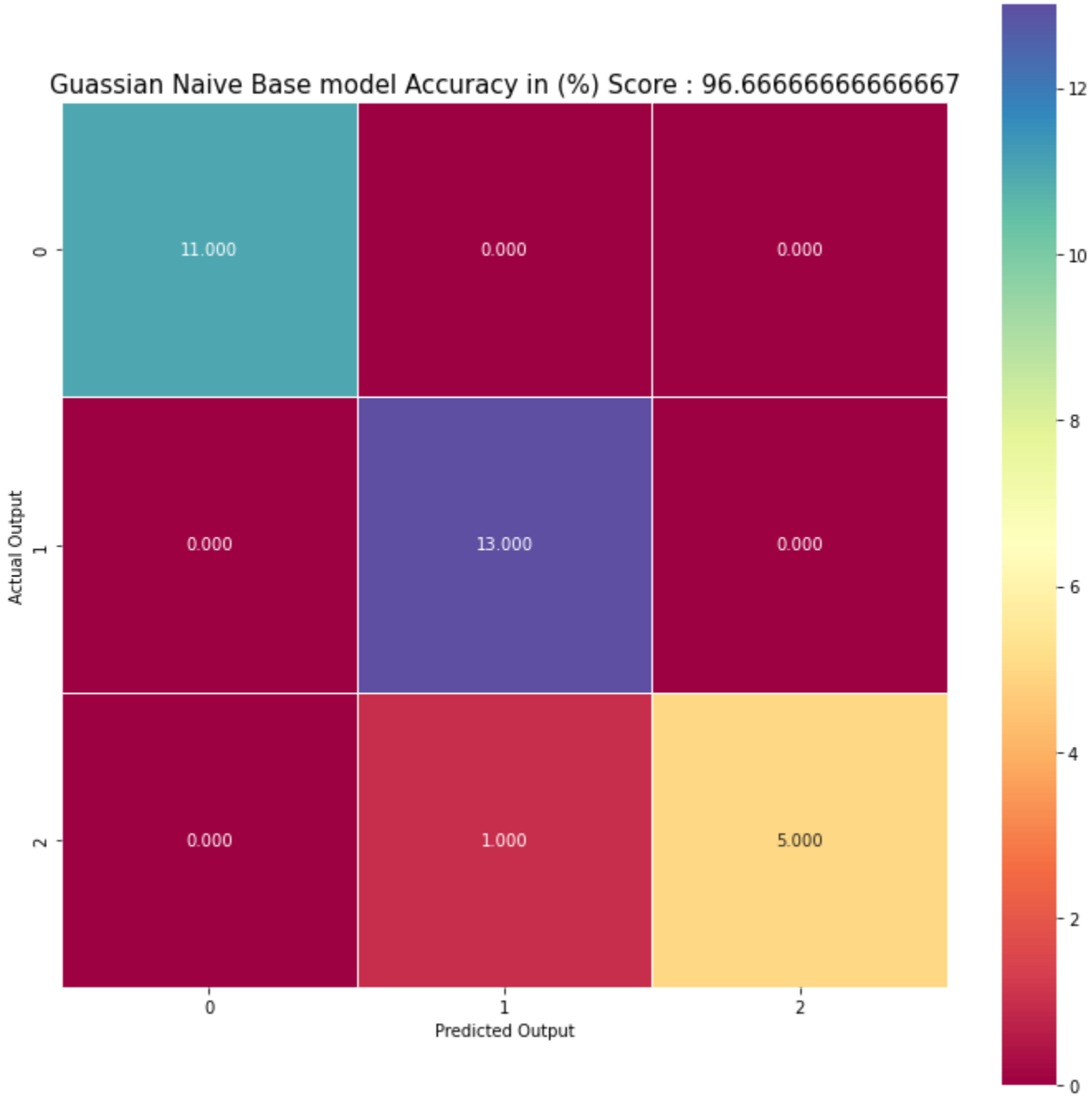
```
Guassian in Naive Base model accuracy in % :  96.66666666666667
```

In [19]:
```
from sklearn import metrics

cm = metrics.confusion_matrix(y_test,y_predict)
cm
```

Out[19]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  1,  5]], dtype=int64)
```

In [26]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
sns.heatmap(cm, annot = True, fmt=".3f",linewidths=".5",square=True,cmap = "Spectral")
plt.xlabel("Predicted Output")
plt.ylabel("Actual Output")

all_sample_title = "Guassian Naive Base model Accuracy in (%) Score : {0}".format(score*100)
plt.title(all_sample_title,size=15)
```

Out[26]:
```
Text(0.5, 1.0, 'Guassian Naive Base model Accuracy in (%) Score : 96.66666666666667')
```



What is Naive Bayes algorithm?

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

Types of Naive Bayes model under the scikit-learn library

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.