DATA SCIENCE Report:

SP20-BCS-071

Muhammad Zain

Section G2:

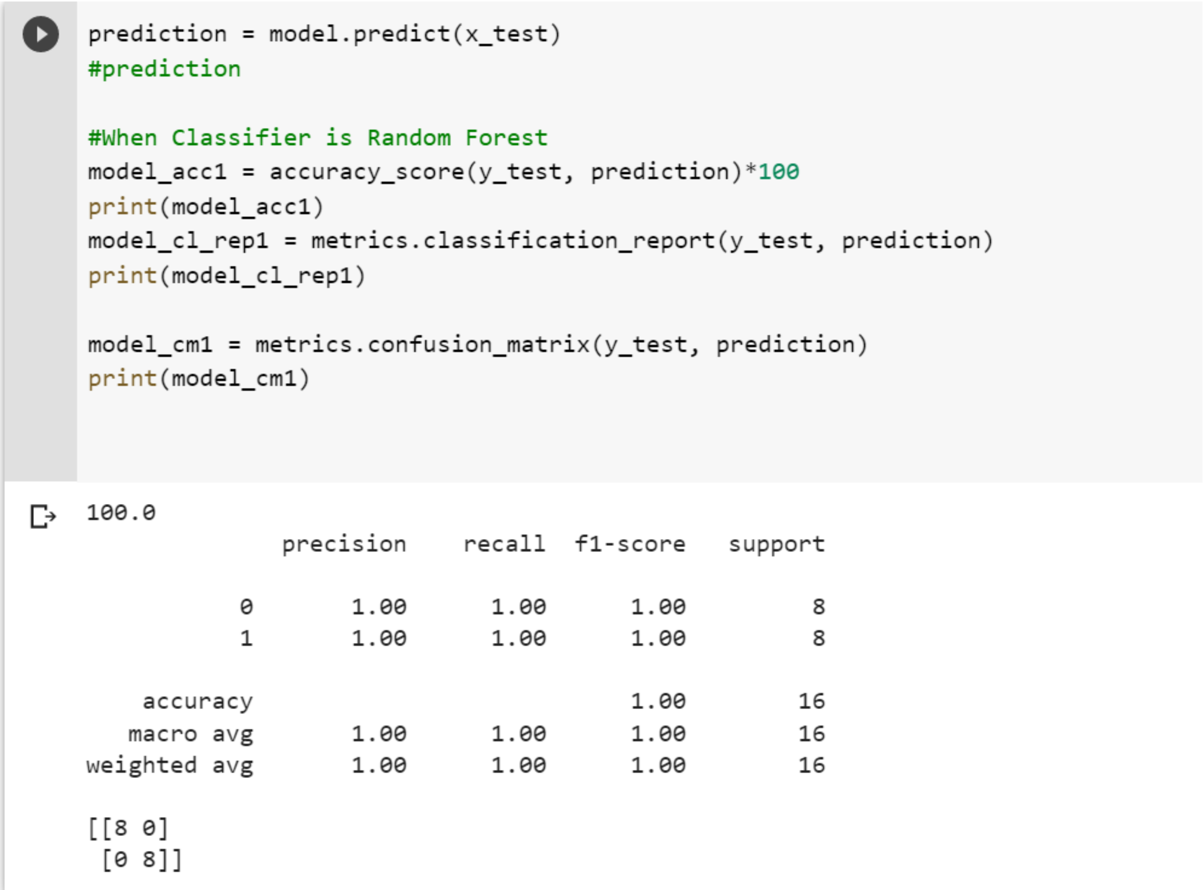
Question-01:

1. 80 instances
2. 7 Inputs Attributes
3. Output has two possible values
   - Male
   - Female
4. There are a total of four attributes are categorical. Below is the list
   - Beard
   - Hair Length
   - Scarf
   - Eye-color
5. The class ratio is 57.5% male and 42.55 female

Question 2:

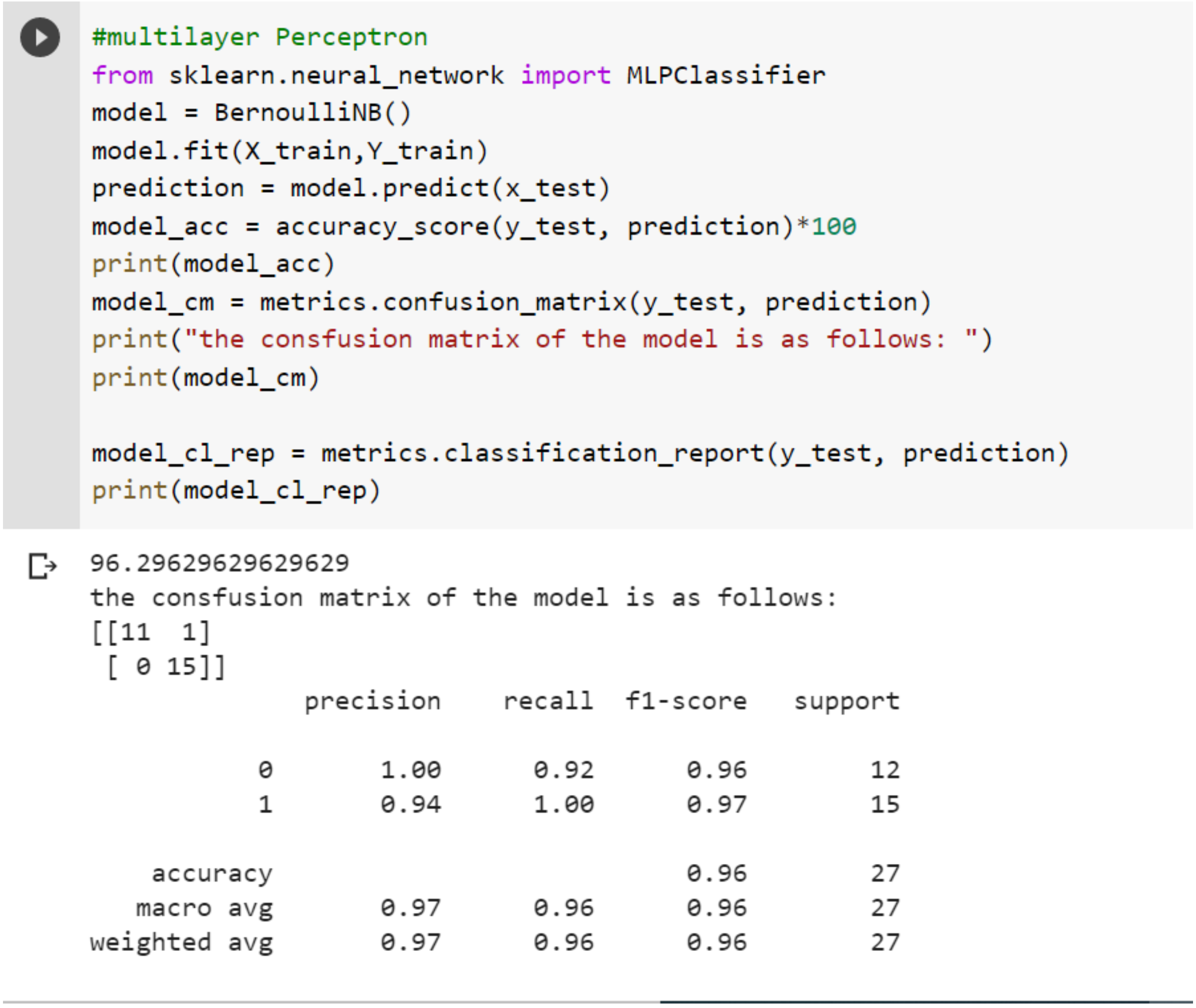
IN Random Forest: The accuracy of the model is 100%, it means the model has classified all the instances correctly.

- Random Forest supported the minimum of 27 instances in 67-33 split. With 100%
- In Support Vector Machine the Accuracy is 70% on 67-33 split.
- In Multilayer perceptron I used two classifiers MultinomialNB() and BernouliNB both has shown the 96.29% accuracy in 67-33 split.



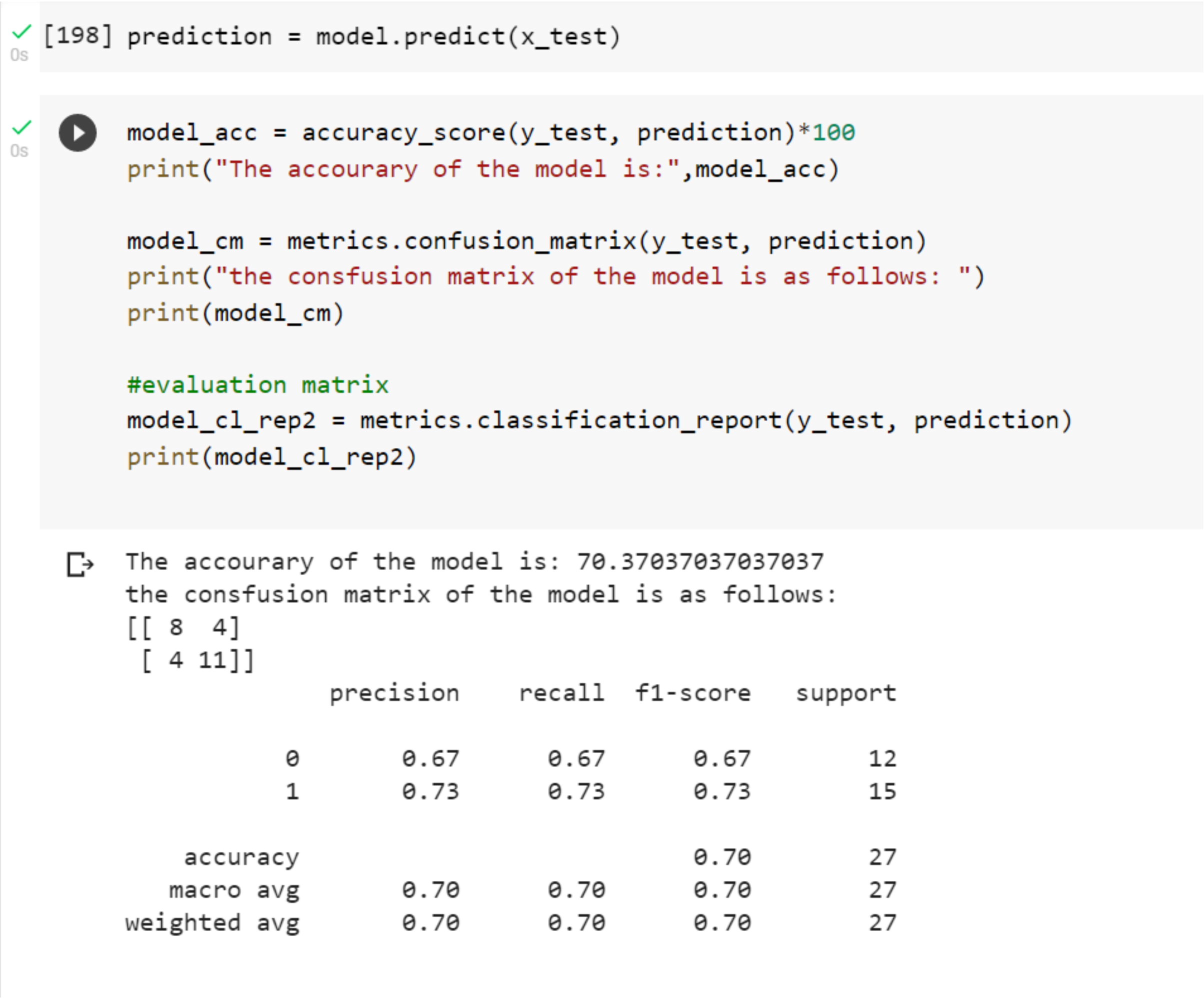
```
prediction = model.predict(x_test)
#prediction

#When Classifier is Random Forest
model_acc1 = accuracy_score(y_test, prediction)*100
print(model_acc1)
model_cl_rep1 = metrics.classification_report(y_test, prediction)
print(model_cl_rep1)

model_cm1 = metrics.confusion_matrix(y_test, prediction)
print(model_cm1)
```

```
100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         8

    accuracy                           1.00        16
   macro avg       1.00      1.00      1.00        16
weighted avg       1.00      1.00      1.00        16

[[8 0]
 [0 8]]
```



```
#multilayer Perceptron
from sklearn.neural_network import MLPClassifier
model = BernoulliNB()
model.fit(X_train,Y_train)
prediction = model.predict(x_test)
model_acc = accuracy_score(y_test, prediction)*100
print(model_acc)
model_cm = metrics.confusion_matrix(y_test, prediction)
print("the consfusion matrix of the model is as follows: ")
print(model_cm)

model_cl_rep = metrics.classification_report(y_test, prediction)
print(model_cl_rep)
```

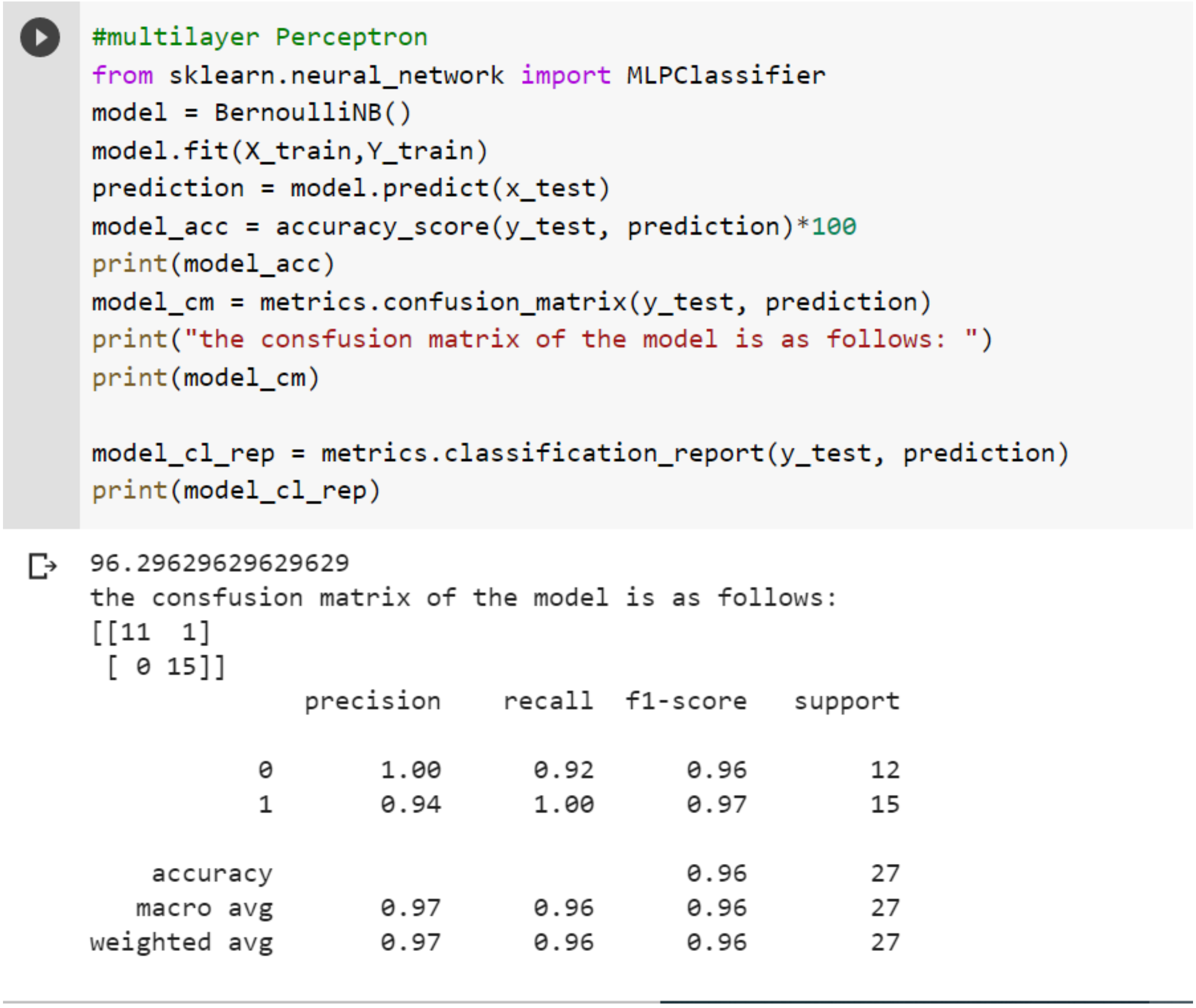
```
96.29629629629629
the consfusion matrix of the model is as follows:
[[11  1]
 [ 0 15]]
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.94      1.00      0.97        15

    accuracy                           0.96        27
   macro avg       0.97      0.96      0.96        27
weighted avg       0.97      0.96      0.96        27
```



```
[198] prediction = model.predict(x_test)
```

```
model_acc = accuracy_score(y_test, prediction)*100
print("The accourary of the model is:",model_acc)

model_cm = metrics.confusion_matrix(y_test, prediction)
print("the consfusion matrix of the model is as follows: ")
print(model_cm)

#evaluation matrix
model_cl_rep2 = metrics.classification_report(y_test, prediction)
print(model_cl_rep2)
```

```
The accourary of the model is: 70.37037037037037
the consfusion matrix of the model is as follows: 
[[ 8  4]
 [ 4 11]]
              precision    recall  f1-score   support

           0       0.67      0.67      0.67        12
           1       0.73      0.73      0.73        15

    accuracy                           0.70        27
   macro avg       0.70      0.70      0.70        27
weighted avg       0.70      0.70      0.70        27
```



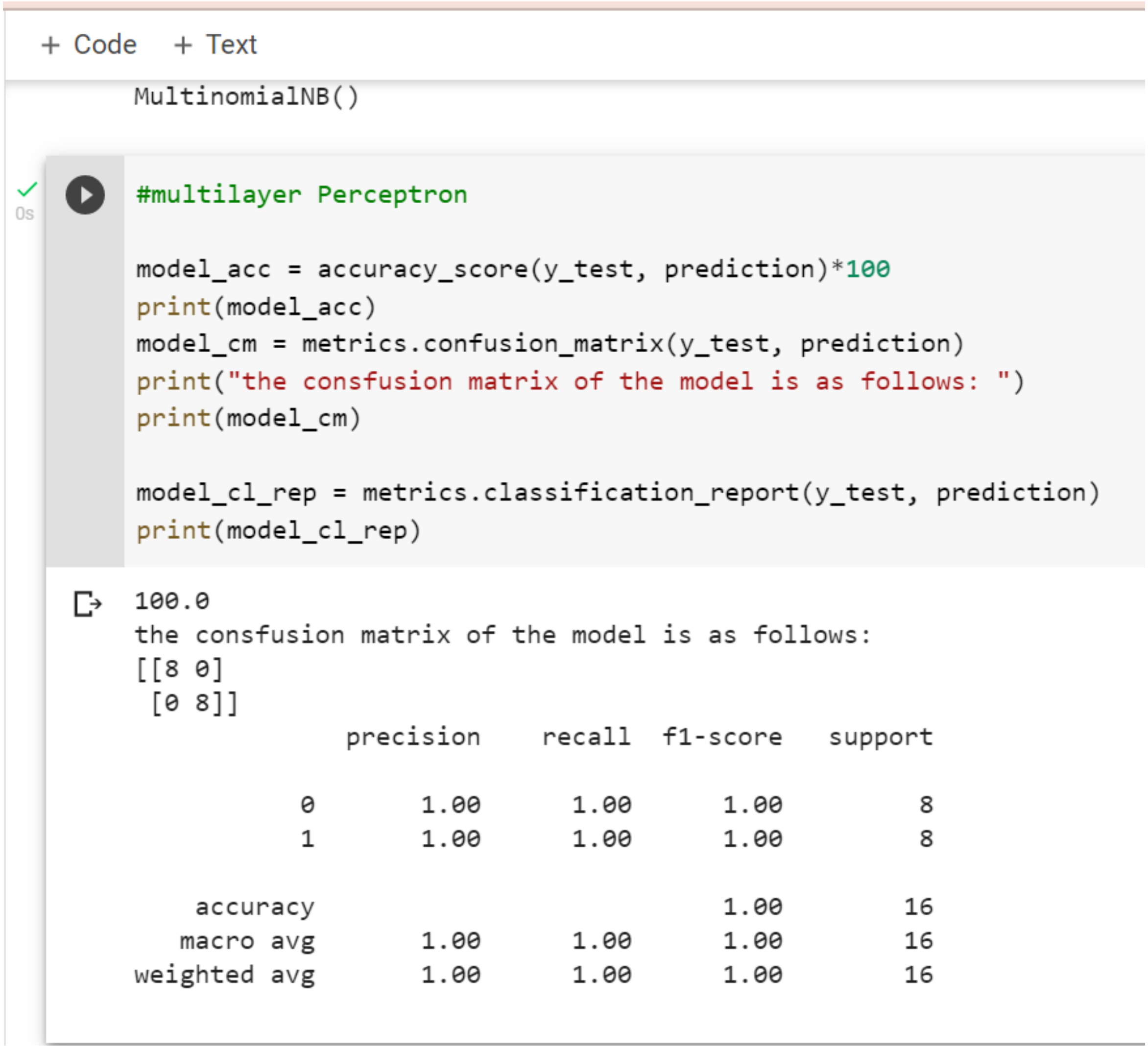
```
#multilayer Perceptron
from sklearn.neural_network import MLPClassifier
model = BernoulliNB()
model.fit(X_train,Y_train)
prediction = model.predict(x_test)
model_acc = accuracy_score(y_test, prediction)*100
print(model_acc)
model_cm = metrics.confusion_matrix(y_test, prediction)
print("the consfusion matrix of the model is as follows: ")
print(model_cm)

model_cl_rep = metrics.classification_report(y_test, prediction)
print(model_cl_rep)
```

```
96.29629629629629
the consfusion matrix of the model is as follows:
[[11  1]
 [ 0 15]]
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.94      1.00      0.97        15

    accuracy                           0.96        27
   macro avg       0.97      0.96      0.96        27
weighted avg       0.97      0.96      0.96        27
```

(B)

After the Rerun of the model at 80-20 split. The results are as follows:

- Random Forest predicted with the accuracy of 100% by supporting 16 instances.
- SVC generated the output with the accuracy of 93.75% same number of 16 instances
- With the Bernoulli classifier the accuracy is 100%
- With the Multinomial the accuracy is 100%.

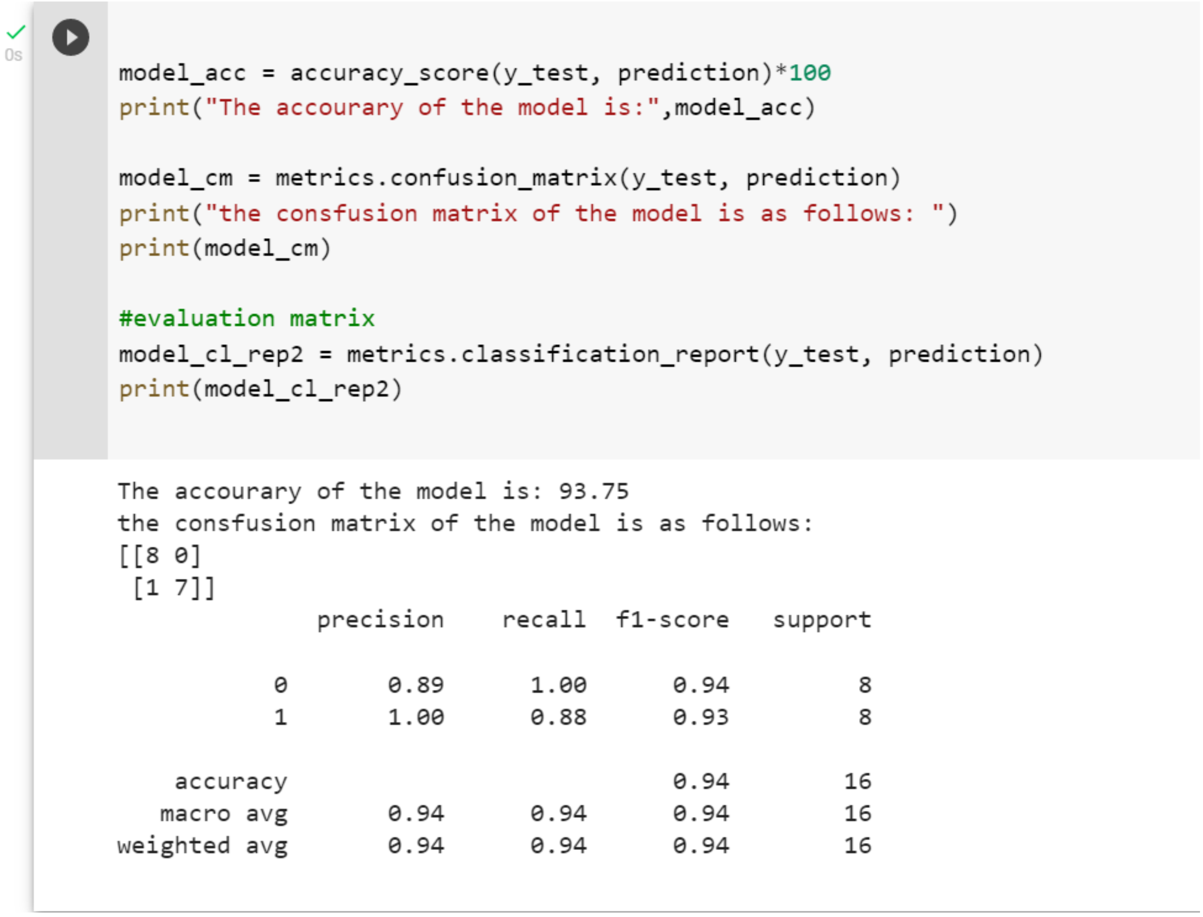


```
MultinomialNB()
```

```
#multilayer Perceptron

model_acc = accuracy_score(y_test, prediction)*100
print(model_acc)
model_cm = metrics.confusion_matrix(y_test, prediction)
print("the consfusion matrix of the model is as follows: ")
print(model_cm)

model_cl_rep = metrics.classification_report(y_test, prediction)
print(model_cl_rep)
```

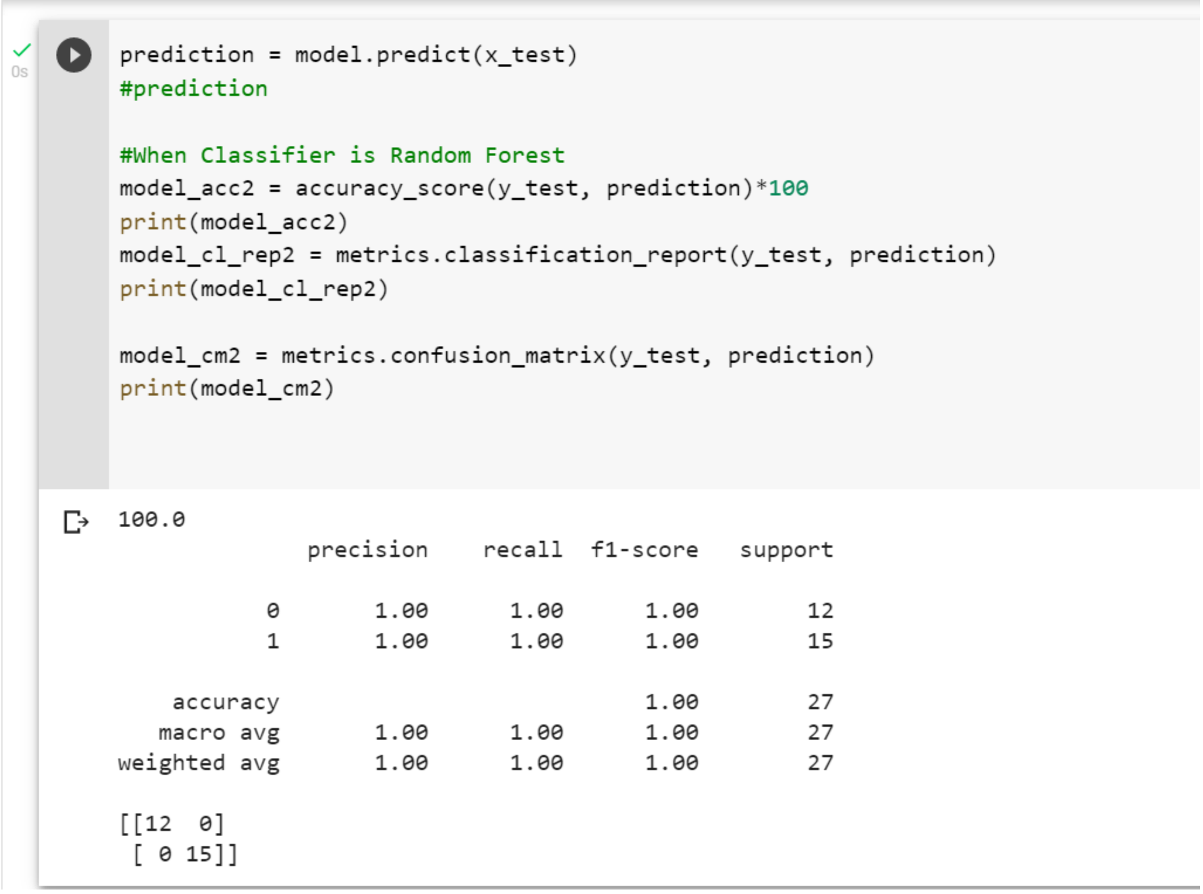
```
100.0
the consfusion matrix of the model is as follows: 
[[8 0]
 [0 8]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         8

    accuracy                           1.00        16
   macro avg       1.00      1.00      1.00        16
weighted avg       1.00      1.00      1.00        16
```

-



```
model_acc = accuracy_score(y_test, prediction)*100
print("The accourary of the model is:",model_acc)

model_cm = metrics.confusion_matrix(y_test, prediction)
print("the consfusion matrix of the model is as follows: ")
print(model_cm)

#evaluation matrix
model_cl_rep2 = metrics.classification_report(y_test, prediction)
print(model_cl_rep2)
```

```
The accourary of the model is: 93.75
the consfusion matrix of the model is as follows:
[[8 0]
 [1 7]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94         8
           1       1.00      0.88      0.93         8

    accuracy                           0.94        16
   macro avg       0.94      0.94      0.94        16
weighted avg       0.94      0.94      0.94        16
```



```
prediction = model.predict(x_test)
#prediction

#When Classifier is Random Forest
model_acc2 = accuracy_score(y_test, prediction)*100
print(model_acc2)
model_cl_rep2 = metrics.classification_report(y_test, prediction)
print(model_cl_rep2)

model_cm2 = metrics.confusion_matrix(y_test, prediction)
print(model_cm2)
```

```
100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00        15

    accuracy                           1.00        27
   macro avg       1.00      1.00      1.00        27
weighted avg       1.00      1.00      1.00        27

[[12  0]
 [ 0 15]]
```

C)

The two Main attributes that are contributing the most are Scarf and Beard. Because almost every male has this feature but not female whereas scarf is the feature that almost every female has but not male

D)

After excluding the main two attributes the classifiers somehow still predict the some of the instances correctly but it has deduced the precision of the model.

In other words, yes by removing the two main attributes , it is effecting the classifiers to predict the values

Question no 3:

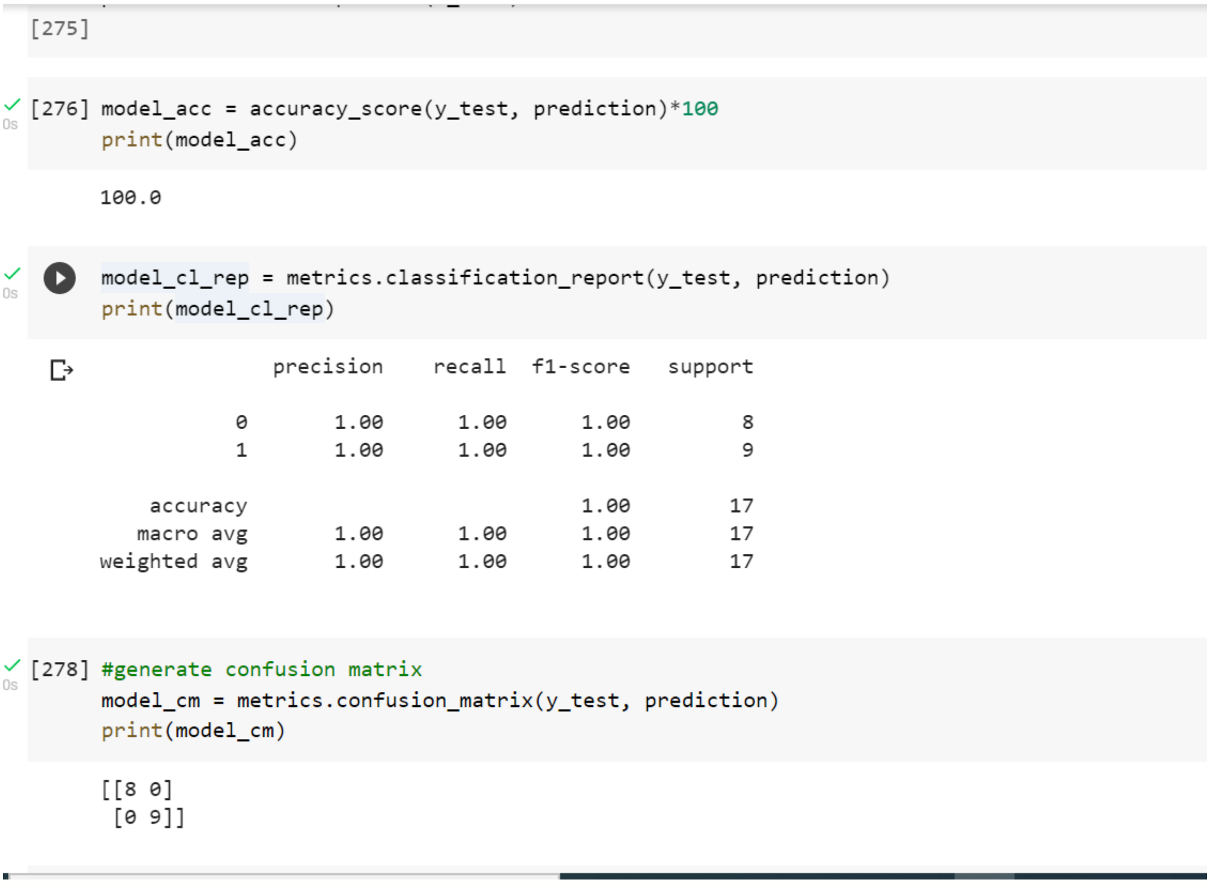
```
cross Validation scores:n [1.        1.        1.        1.        0.81481481]
Average Cross Validation score :0.962962962962963
```

```
shuffle_split=ShuffleSplit(test_size=0.33,train_size = 0.2,n_splits=5)
```

Question -04:

New Instances Respectively according to the Datasets.

Wei	**Hei**	**beard**	**HL**	**SZ**	**Scarf**	**EC**	**Gender**		
1.	85	135	no	long	35	yes	black	female	
2.	45	175	no	short	40	no	blue	female	
3.	80	171	yes	short	44	no	black	male	
4.	85	177	yes	medium		46	yes	black	male
5.	69	165	no	long	38	no	black	female	



```
[275]
```

```
[276] model_acc = accuracy_score(y_test, prediction)*100
      print(model_acc)
```

```
100.0
```

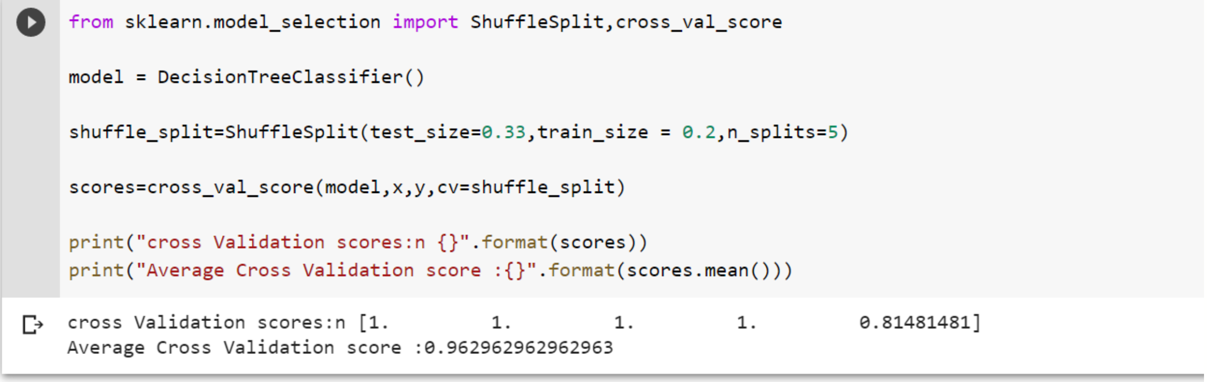
```
model_cl_rep = metrics.classification_report(y_test, prediction)
print(model_cl_rep)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         9

    accuracy                           1.00        17
   macro avg       1.00      1.00      1.00        17
weighted avg       1.00      1.00      1.00        17
```

```
[278] #generate confusion matrix
      model_cm = metrics.confusion_matrix(y_test, prediction)
      print(model_cm)
```

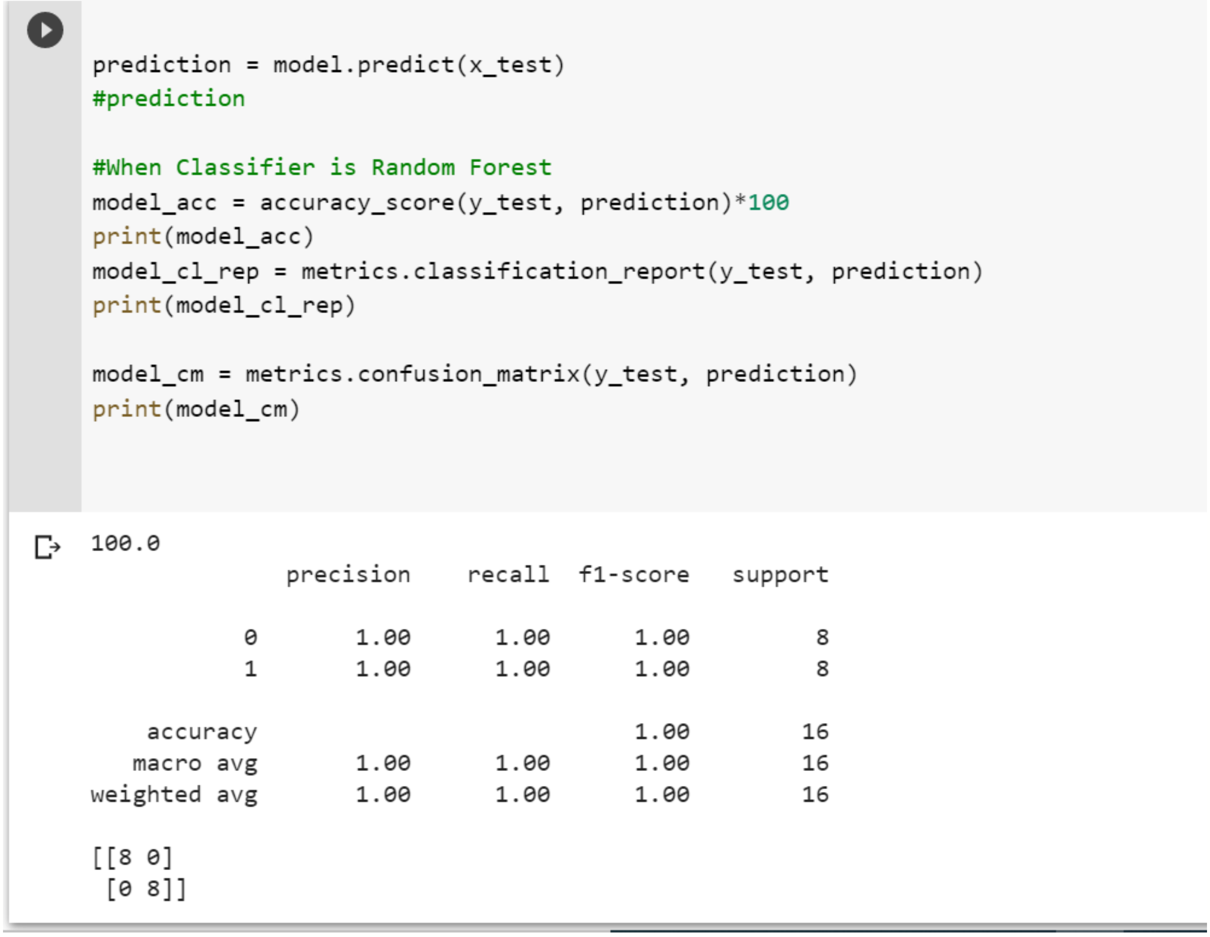
```
[[8 0]
 [0 9]]
```

Some other screenshots listed here too.



```
from sklearn.model_selection import ShuffleSplit,cross_val_score

model = DecisionTreeClassifier()

shuffle_split=ShuffleSplit(test_size=0.33,train_size = 0.2,n_splits=5)

scores=cross_val_score(model,x,y,cv=shuffle_split)

print("cross Validation scores:n {}".format(scores))
print("Average Cross Validation score :{}".format(scores.mean()))
```

```
cross Validation scores:n [1.         1.         1.         1.         0.81481481]
Average Cross Validation score :0.962962962962963
```



```
prediction = model.predict(x_test)
#prediction

#When Classifier is Random Forest
model_acc = accuracy_score(y_test, prediction)*100
print(model_acc)
model_cl_rep = metrics.classification_report(y_test, prediction)
print(model_cl_rep)

model_cm = metrics.confusion_matrix(y_test, prediction)
print(model_cm)
```

```
100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         8

    accuracy                           1.00        16
   macro avg       1.00      1.00      1.00        16
weighted avg       1.00      1.00      1.00        16

[[8 0]
 [0 8]]
```