INTERNAL 2 – ETDS

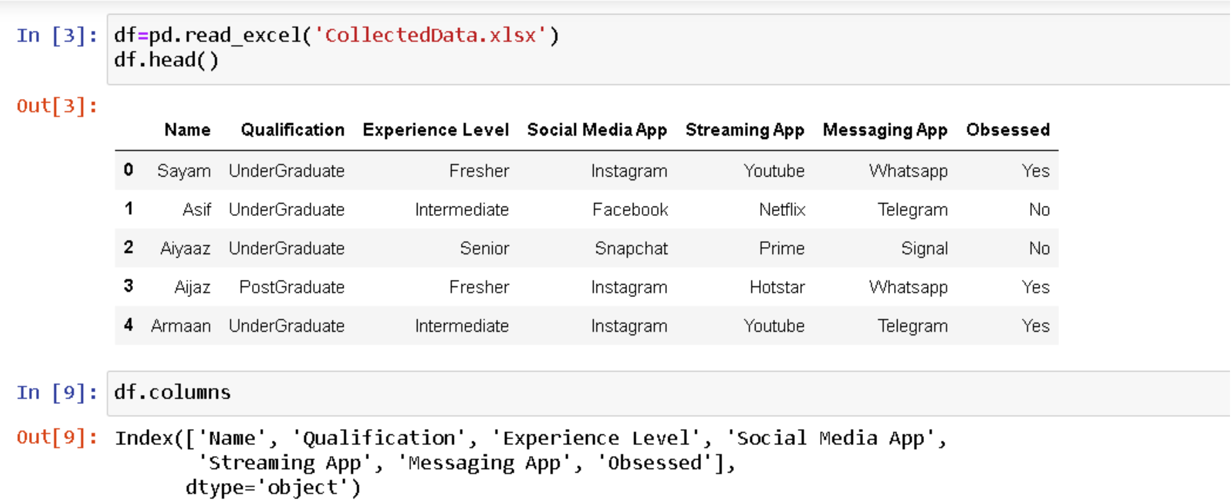
- ## PROBLEM STATEMENT

  The Problems statement is on social media apps usage of professional employees which apps they use for streaming, messaging , social for entertainment purpose. Some important features are Qualification , Experience level based on this evaluate whether they are obsessed or not.

- ## DATA COLLECTION

  The Data was collected as survey questionnare on Google form.

- ## ATTRIBUTES IN DATA



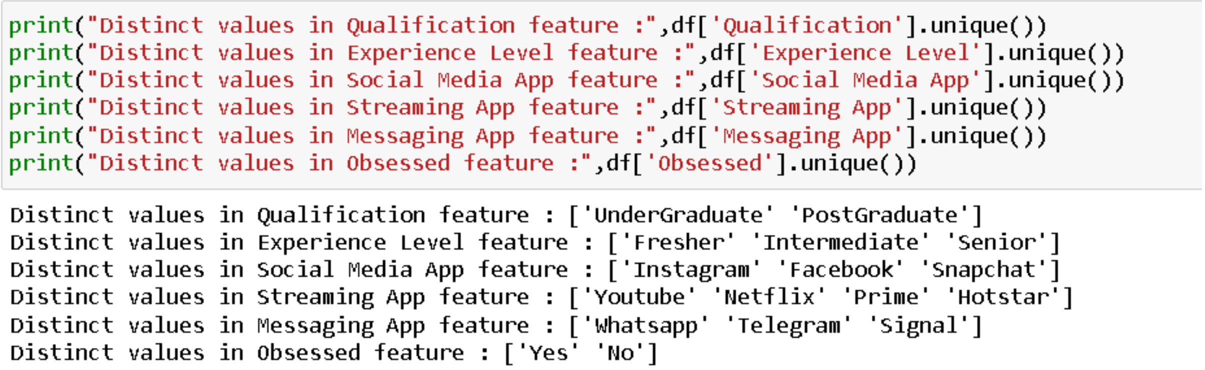
```
In [3]: df=pd.read_excel('CollectedData.xlsx')
        df.head()
```

Out[3]:

	Name	Qualification	Experience Level	Social Media App	Streaming App	Messaging App	Obsessed
0	Sayam	UnderGraduate	Fresher	Instagram	Youtube	Whatsapp	Yes
1	Asif	UnderGraduate	Intermediate	Facebook	Netflix	Telegram	No
2	Aiyaaz	UnderGraduate	Senior	Snapchat	Prime	Signal	No
3	Aijaz	PostGraduate	Fresher	Instagram	Hotstar	Whatsapp	Yes
4	Armaan	UnderGraduate	Intermediate	Instagram	Youtube	Telegram	Yes

```
In [9]: df.columns

Out[9]: Index(['Name', 'Qualification', 'Experience Level', 'Social Media App',
               'Streaming App', 'Messaging App', 'Obsessed'],
              dtype='object')
```

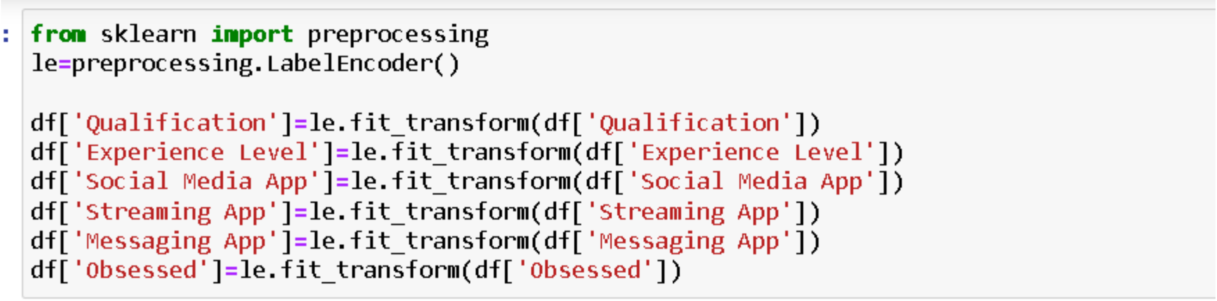
Unique Values of the features



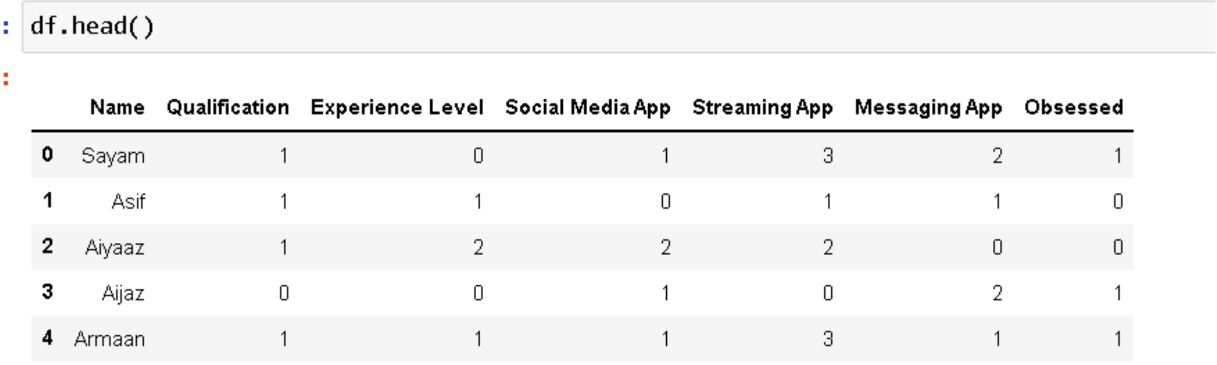
```
print("Distinct values in Qualification feature :",df['Qualification'].unique())
print("Distinct values in Experience Level feature :",df['Experience Level'].unique())
print("Distinct values in Social Media App feature :",df['Social Media App'].unique())
print("Distinct values in Streaming App feature :",df['Streaming App'].unique())
print("Distinct values in Messaging App feature :",df['Messaging App'].unique())
print("Distinct values in Obsessed feature :",df['Obsessed'].unique())
```

```
Distinct values in Qualification feature : ['UnderGraduate' 'PostGraduate']
Distinct values in Experience Level feature : ['Fresher' 'Intermediate' 'Senior']
Distinct values in Social Media App feature : ['Instagram' 'Facebook' 'Snapchat']
Distinct values in Streaming App feature : ['Youtube' 'Netflix' 'Prime' 'Hotstar']
Distinct values in Messaging App feature : ['Whatsapp' 'Telegram' 'Signal']
Distinct values in Obsessed feature : ['Yes' 'No']
```

Encoding the categorical values to numerical with label encoder.



```
: from sklearn import preprocessing
  le=preprocessing.LabelEncoder()

  df['Qualification']=le.fit_transform(df['Qualification'])
  df['Experience Level']=le.fit_transform(df['Experience Level'])
  df['Social Media App']=le.fit_transform(df['Social Media App'])
  df['Streaming App']=le.fit_transform(df['Streaming App'])
  df['Messaging App']=le.fit_transform(df['Messaging App'])
  df['Obsessed']=le.fit_transform(df['Obsessed'])
```

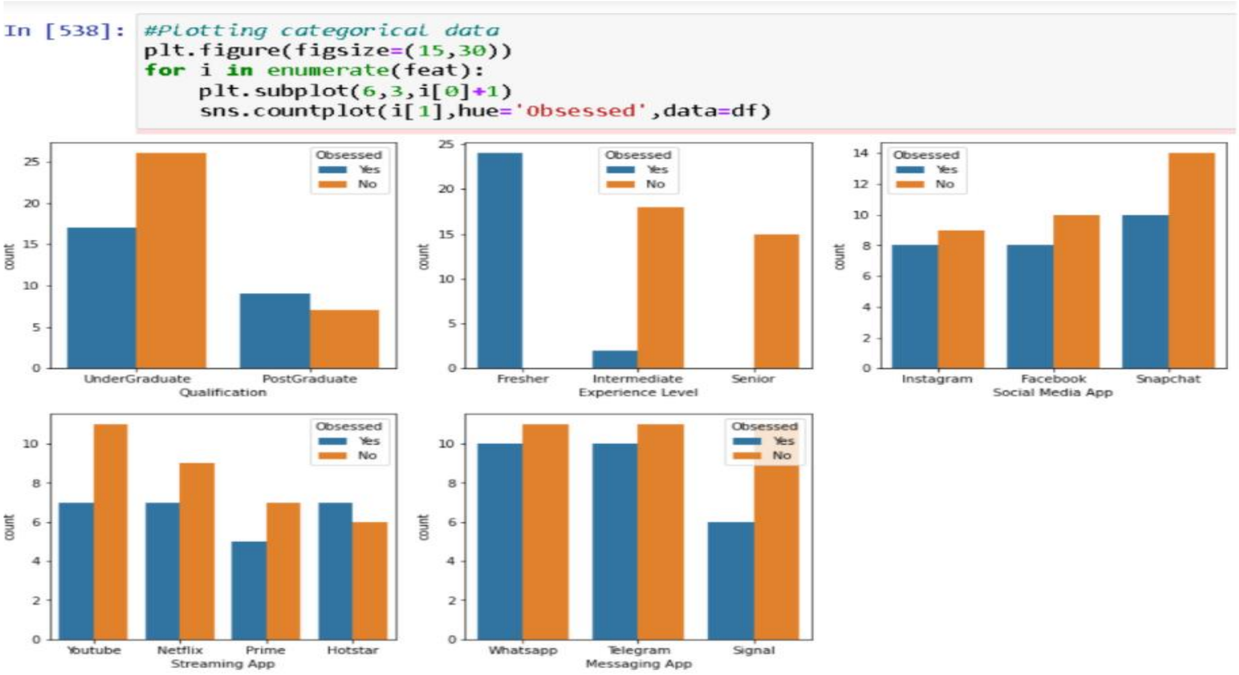


```
: df.head()
```

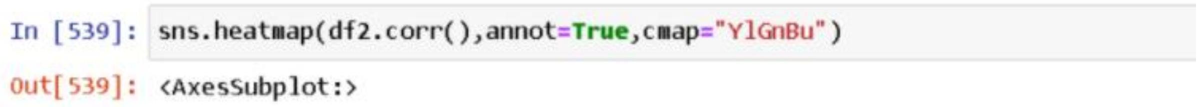
:

	Name	Qualification	Experience Level	Social Media App	Streaming App	Messaging App	Obsessed
0	Sayam	1	0	1	3	2	1
1	Asif	1	1	0	1	1	0
2	Aiyaaz	1	2	2	2	0	0
3	Aijaz	0	0	1	0	2	1
4	Armaan	1	1	1	3	1	1

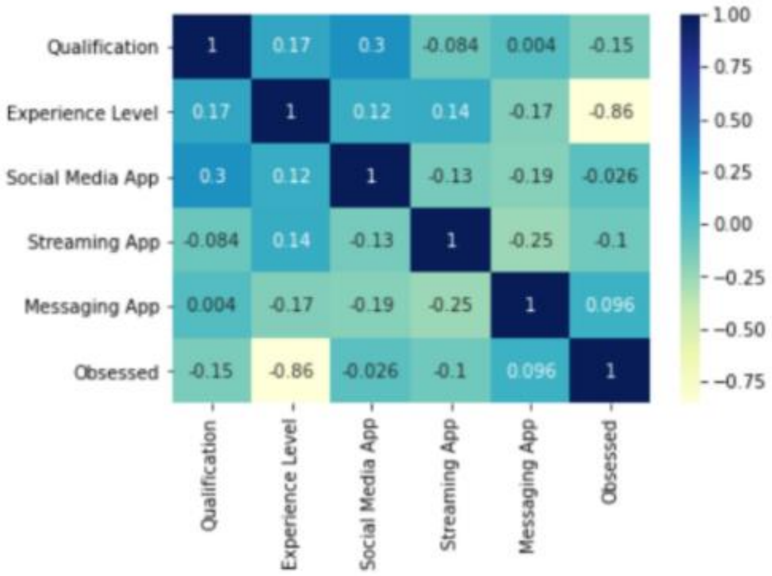
Vizualizing the data.



```
In [538]: #Plotting categorical data
          plt.figure(figsize=(15,30))
          for i in enumerate(feat):
              plt.subplot(6,3,i[0]+1)
              sns.countplot(i[1],hue='Obsessed',data=df)
```

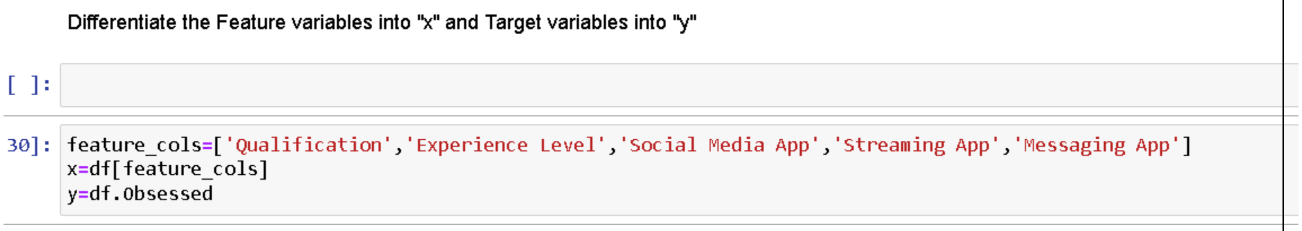


```
In [539]: sns.heatmap(df2.corr(),annot=True,cmap="YlGnBu")

Out[539]: <AxesSubplot:>
```



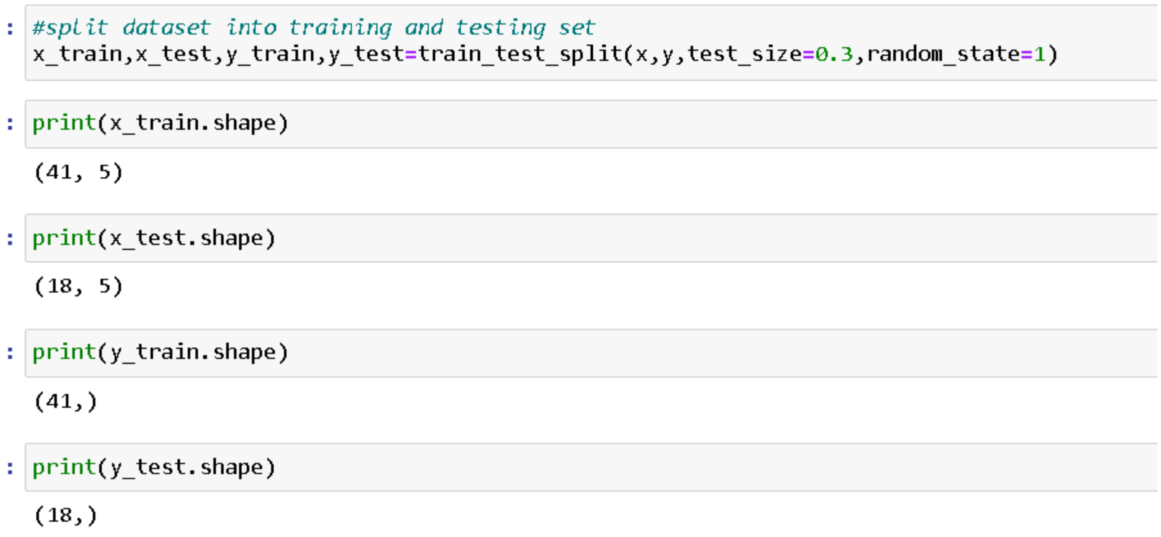
- ## SAMPLING SET (TRAIN AND TEST DATA)

Training =70 % and Testing =30%



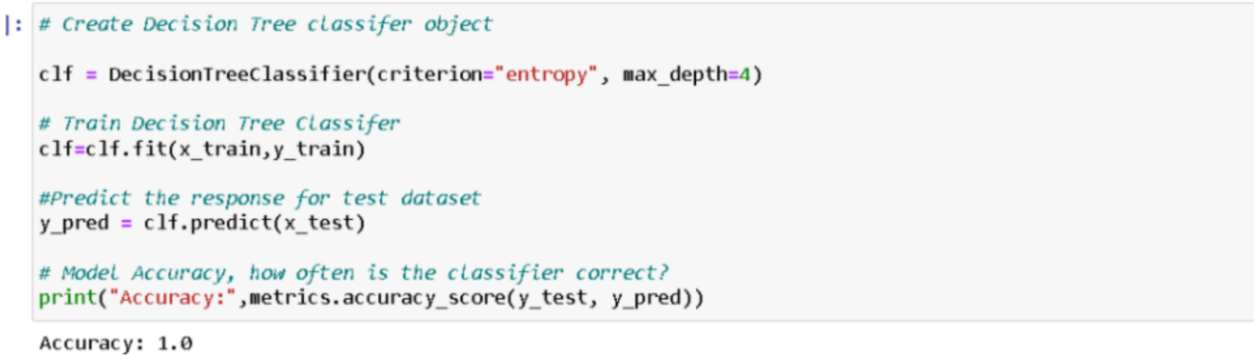
Differentiate the Feature variables into "x" and Target variables into "y"

```
[ ]: 

30]: feature_cols=['Qualification','Experience Level','Social Media App','Streaming App','Messaging App']
     x=df[feature_cols]
     y=df.Obsessed
```

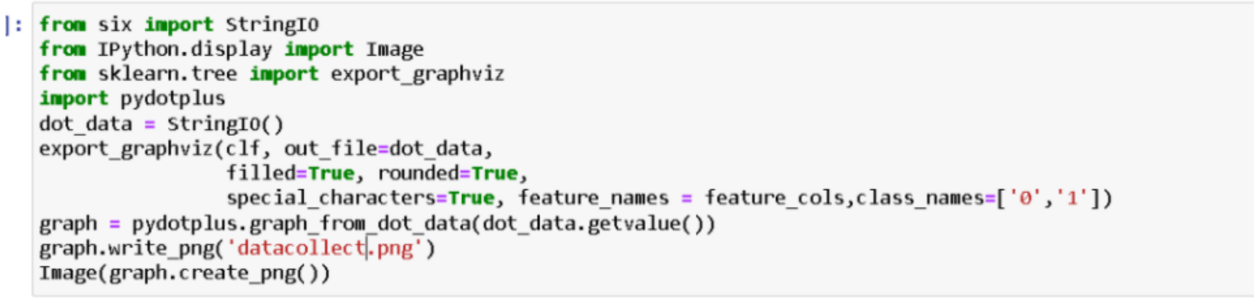


```
: #split dataset into training and testing set
  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)

: print(x_train.shape)

  (41, 5)

: print(x_test.shape)

  (18, 5)

: print(y_train.shape)

  (41,)

: print(y_test.shape)

  (18,)
```

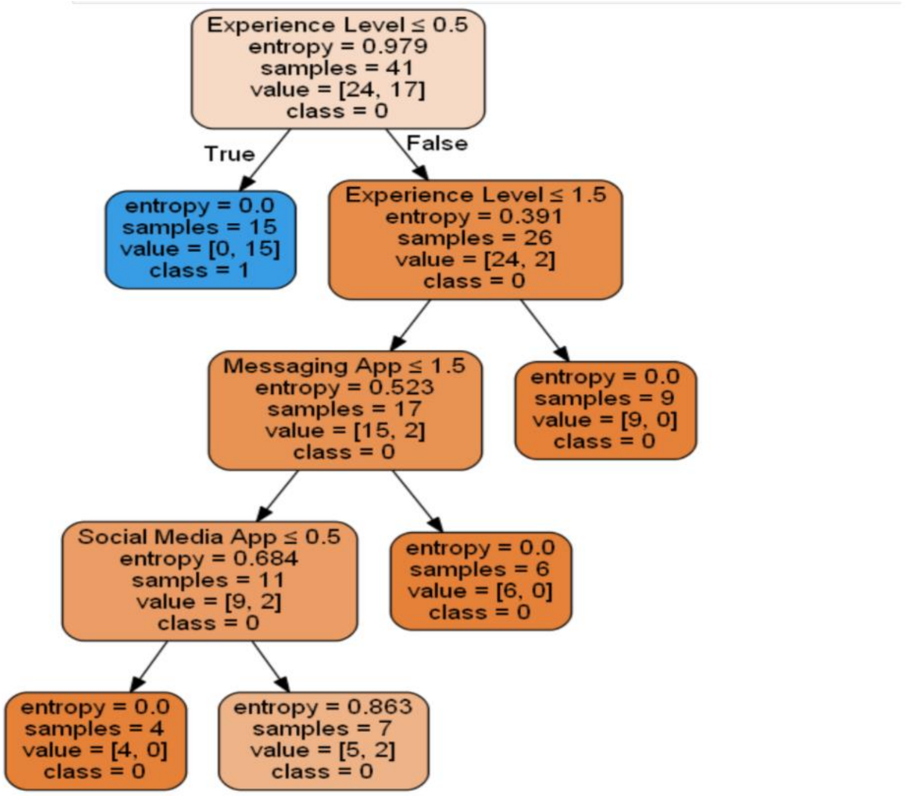
- ## MODELLING WITH ACCURACY.



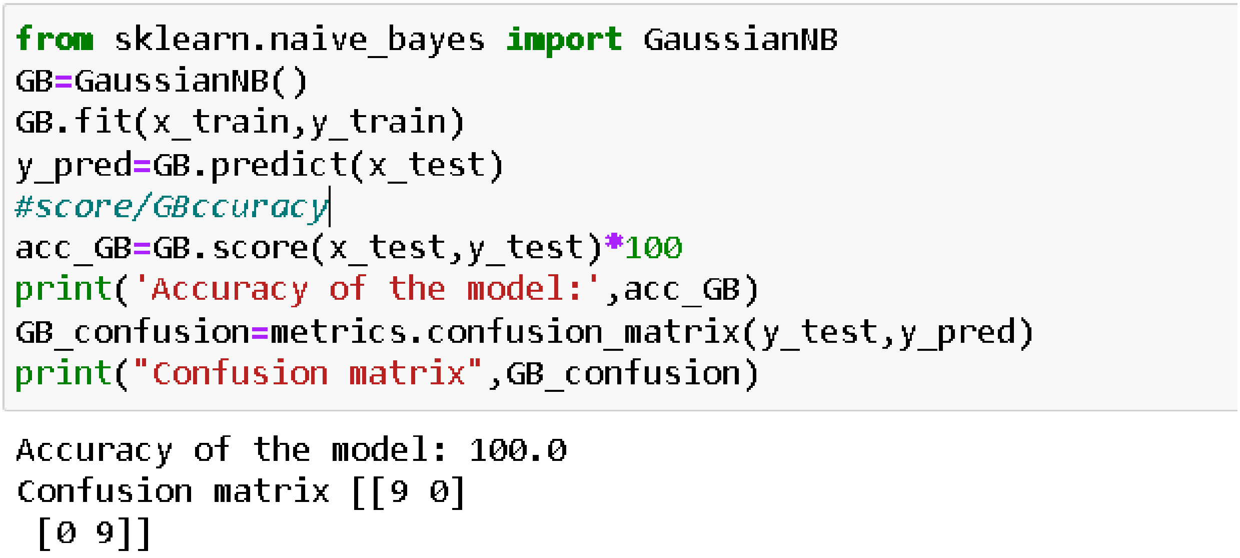
```
]: # Create Decision Tree classifer object

   clf = DecisionTreeClassifier(criterion="entropy", max_depth=4)

   # Train Decision Tree Classifer
   clf=clf.fit(x_train,y_train)

   #Predict the response for test dataset
   y_pred = clf.predict(x_test)

   # Model Accuracy, how often is the classifier correct?
   print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

   Accuracy: 1.0
```



```
]: from six import StringIO
   from IPython.display import Image
   from sklearn.tree import export_graphviz
   import pydotplus
   dot_data = StringIO()
   export_graphviz(clf, out_file=dot_data,
                   filled=True, rounded=True,
                   special_characters=True, feature_names = feature_cols,class_names=['0','1'])
   graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
   graph.write_png('datacollect.png')
   Image(graph.create_png())
```



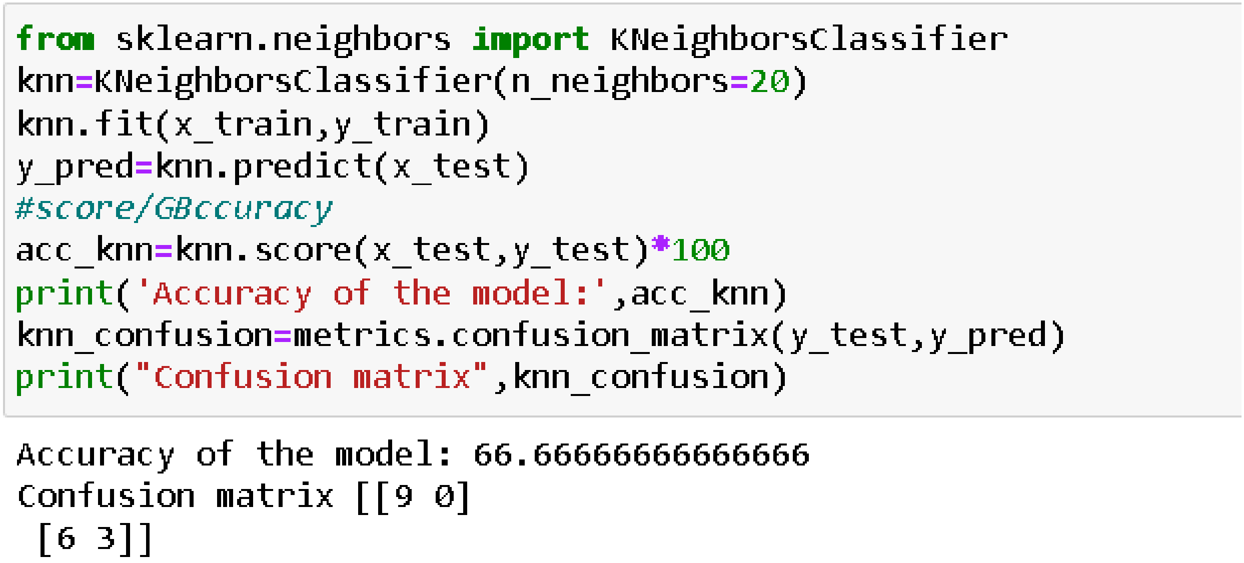
NAÏVE BAYES



```
from sklearn.naive_bayes import GaussianNB
GB=GaussianNB()
GB.fit(x_train,y_train)
y_pred=GB.predict(x_test)
#score/GBccuracy
acc_GB=GB.score(x_test,y_test)*100
print('Accuracy of the model:',acc_GB)
GB_confusion=metrics.confusion_matrix(y_test,y_pred)
print("Confusion matrix",GB_confusion)
```

```
Accuracy of the model: 100.0
Confusion matrix [[9 0]
 [0 9]]
```

KNN



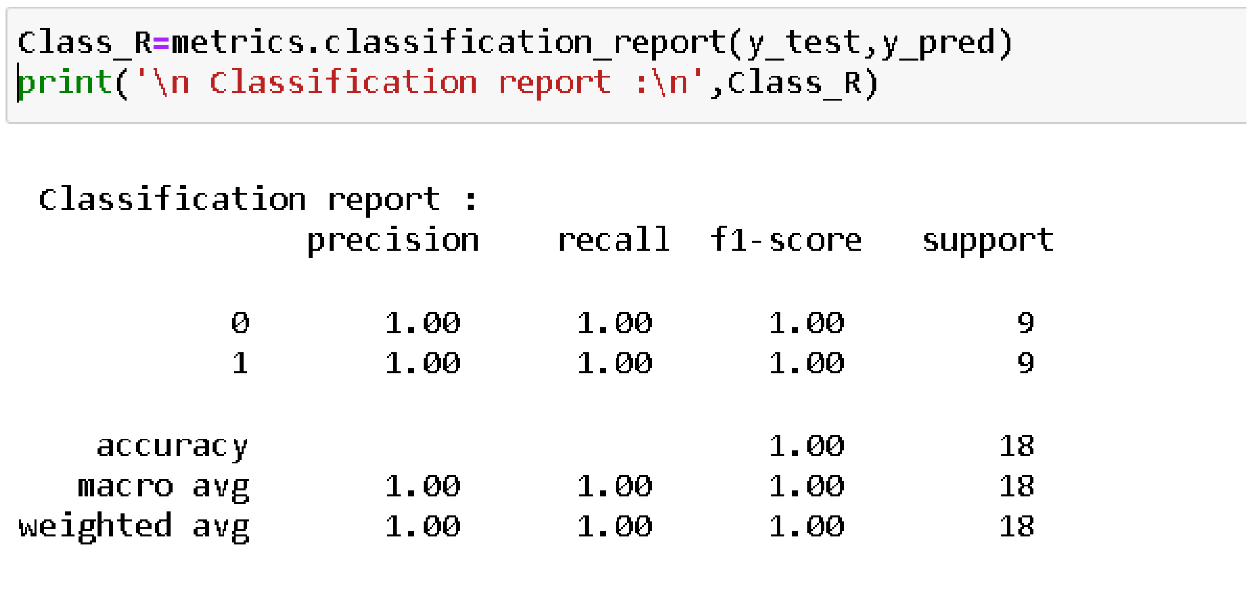
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=20)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
#score/GBccuracy
acc_knn=knn.score(x_test,y_test)*100
print('Accuracy of the model:',acc_knn)
knn_confusion=metrics.confusion_matrix(y_test,y_pred)
print("Confusion matrix",knn_confusion)
```

```
Accuracy of the model: 66.66666666666666
Confusion matrix [[9 0]
 [6 3]]
```

• COMPARATIVE ANALYSIS

MODEL	ACCURACY
KNN	66.66
NAÏVE BAYES	100
DECISION TREE	100

CLASSIFICATION REPORT



```
Class_R=metrics.classification_report(y_test,y_pred)
print('\n Classification report :\n',Class_R)
```

```
 Classification report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00         9

    accuracy                           1.00        18
   macro avg       1.00      1.00      1.00        18
weighted avg       1.00      1.00      1.00        18
```

Among this three models , knn has the lowest accuracy and decision tree and knn has the highest accuracy.

• REFERENCES

Classification Algorithms - NaÃ¯ve Bayes (tutorialspoint.com)
https://www.tutorialspoint.com/machine_learning_with_python/machine.
Scikit Learn - Decision Trees (tutorialspoint.com)