Decision Trees Technique

In [1]:

```
#Importing the libraries "pandas and numpy to my python script
#with the standard short name as "pd and np".

#Uploading the file on google colab and choosing the selected dataset by clicking "choose
files".

import pandas as pd
import numpy as np


from google.colab import files
uploaded = files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving New_Data2.csv to New_Data2.csv
```

In [2]:

```
#In the first step, importing the dataset in the project by the "read_csv" function
#and that reads the data into a pandas dataframe object.

dataset = pd.read_csv('New_Data2.csv')
```

In [3]:

```
#Here showing the full dataset with 1372 rows and 5 columns.

dataset
```

Out[3]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

In [4]:

```
#head() method returns a particular rows from the top and where did not mention
#the number below hence returns first 5 rows.

dataset.head()
```

Out[4]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Predicting the class based on other variables

In [5]:

```
#Dropping the "class" column by drop() function.
#Then, showing the value of x and value of y by using print function which prints
#the specified message to the screen.

x=dataset.drop('Class', axis=1)
y=dataset['Class']
print("value of x: ", x.head())
print("\n value of y: ", y.head())
```

```
value of x:      Variance  Skewness  Curtosis  Entropy
0   3.62160    8.6661   -2.8073 -0.44699
1   4.54590    8.1674   -2.4586 -1.46210
2   3.86600   -2.6383    1.9242  0.10645
3   3.45660    9.5228   -4.0112 -3.59440
4   0.32924   -4.4552    4.5718 -0.98880

 value of y:  0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int64
```

Decision Trees Technique

In [6]:

```
#Decision Trees model is helpful for analyzing quantitative data and making a
#decision based on numbers.

#Importing DecisionTreeClassifier is capable of performing multi-class classification on
a dataset.

#Importing train_test_split function to split arrays or matrices into
#random subsets for train and test data.

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
classifier=DecisionTreeClassifier()
```

In [7]:

```
#Splitting the data using train_test_split.
#The purpose of split is to break the data into training and test or validation set.
#After that can check the validation of the model by fitting the training data and
#predicting using the test or validation by applying the technique.

x_train, x_test, y_train, y_test=train_test_split(x,y,train_size=0.8, test_size=0.2, ran
dom_state=50)
classifier.fit(x_train, y_train)
```

Out[7]:

```
DecisionTreeClassifier()
DecisionTreeClassifier()
```

In [8]:

```
#Predicting the values of test result by using classifier.predict.

prediction=classifier.predict(x_test)
```

In [9]:

```
#Importing confusion_matrix and classification_report.
#we will learn to interpret the confusion matrix and the classification report
#while using them to evaluate the performance.

from sklearn.metrics import confusion_matrix,classification_report
```

In [10]:

```
#Here is the prediction results for the test of confusion_matrix and classification_report.

print(confusion_matrix(y_test,prediction))
print(classification_report(y_test,prediction))
```

```
[[154   1]
 [  3 117]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       155
           1       0.99      0.97      0.98       120

    accuracy                           0.99       275
   macro avg       0.99      0.98      0.99       275
weighted avg       0.99      0.99      0.99       275
```