REPORT

ON

Analyze the difficulty level of a particular course using clustering techniques

Submitted by: Yash

11701070

Programme

B.Tech (Computer Science & Engineering)

Under the Guidance of

Mrs. USHA MITTAL

School of Computer Science & Engineering

Lovely Professional University, Phagwara

(April,2020)

TABLE OF CONTENT

GITHUB REPOSITORY LINK: -

https://github.com/Yash-Pandey07/Clustring

Abstract

In the present serious universe of instructive associations, the colleges and universities are utilizing different information mining apparatuses and methods to improve the understudies' execution. Presently a days, when the quantity of drop out understudies is expanding each year, on the off chance that we become more acquainted with the likelihood of a understudy whether he/she will have the option to adapt up effectively to the course, it is conceivable to take some preventive activities in advance.At the end of the day, on the off chance that we become more acquainted with that an understudy will clear his papers in the course or he will have return in papers, a educator/parent can concentrate more on such understudies.The standard purpose of this errand is to exhibit the opportunity of planning and showing a little dataset size and the common sense of making a gauge model with substantial precision rate. This assessment explores likewise the opportunity of perceiving the key pointers in the little dataset, which will be utilized in making the desire model, using assorted gathering frameworks. Best pointers were dealt with into various AI estimations to evaluate them for the most exact model. The essential aftereffects of this assessment have exhibited the capability of machine learning discriminant examination computations in planning little dataset size and in making a commendable request's exactness and resolute quality test rates.

Keywords

Data Pre Processing, Data Mapping,Logistic Regression Classifier, K-Means,Different types of Clustering. ,pandas ,numpy ,Sklearn ,matplo.lib.

INTRODUCTION

Clustering: is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields.

K-means Clustering • K-means clustering is an algorithm to classify or to group your objects based on attributes/features into K number of group. K is positive integer number. • The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Thus the purpose of K-mean clustering is to classify the data.

K-means Clustering - Example • The basic step of k-means clustering is simple. In the beginning we determine number of cluster K and we assume the centroid or center of these clusters. We can take any random objects as the initial centroids or the first K objects in sequence can also serve as the initial centroids. • Then the K means algorithm will do the three steps below until convergence

Clustering is a Machine Learning technique that involves the grouping of data points. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields.

Broadly speaking, clustering can be divided into two subgroups :

- **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

Thus, this task plans to limit the previously mentioned holes by settling the accompanying examination questions:

What is the best AI bunching model for grouping understudy's exposition venture course , utilizing little dataset size, with a sensible and noteworthy course trouble?

What are the primary key pointers that could help in making the order model for anticipating understudies' thesis venture course?

PROPOSED METHODOLOGY

1. DATA PRE PROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.
Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.It involves the following steps: -

- Importing the libraries.
- Importing the Dataset.
- Handling of Missing Data.
- Handling of Categorical Data.
- Splitting the dataset into training and testing datasets.

Importing Libararies

To make use of the functions in a module, you'll need to import the module with an import statement. An import statement is made up of the import keyword along with the name of the module. In a Python file, this will be declared at the top of the code.

```
import pandas as pd
import os
import matplotlib.pyplot as plt # Graphs & Visualization
import seaborn as sns
from sklearn.cluster import KMeans
print("Start")
```

Importing the DataSet

Pandas is the most popular data manipulation package in Python, and DataFrames are the Pandas data type for storing tabular 2D data.

	0	Termid	RegdNo	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	...	CA_3	CA_4	Height	Weight	ScholarType	Direction	Gen
0	4036	418192	2088776	NTE538	B+	67.0	NaN	NaN	64.0	NaN	...	4.0	18.0	162	93	Day Scholar	North	M
1	4037	418192	2088776	NTE8	B	57.0	0.0	62.0	NaN	95.0	...	20.0	2.0	162	93	Day Scholar	North	M
2	4038	418192	2088776	NTE9	A+	73.0	NaN	NaN	68.0	100.0	...	0.0	7.0	162	93	Day Scholar	North	M
3	18100	418192	5262776	NTE538	F	16.0	NaN	NaN	0.0	NaN	...	1.0	4.0	171	76	Day Scholar	West	M
4	18101	418192	5262776	NTE8	D	60.0	0.0	49.0	NaN	74.0	...	7.0	7.0	171	76	Day Scholar	West	M
5	18102	418192	5262776	NTE9	E	84.0	NaN	NaN	22.0	86.0	...	2.0	0.0	171	76	Day Scholar	West	M
6	45061	418192	11226776	NTE538	F	0.0	NaN	NaN	0.0	NaN	...	0.0	0.0	167	94	Day Scholar	East	Fem
7	45062	418192	11226776	NTE8	F	0.0	0.0	0.0	NaN	0.0	...	0.0	0.0	167	94	Day Scholar	East	Fem
8	45063	418192	11226776	NTE9	F	0.0	NaN	NaN	0.0	0.0	...	0.0	0.0	167	94	Day Scholar	East	Fem
9	49898	718192	12190776	NTE12	B	61.0	NaN	NaN	62.0	NaN	...	8.0	3.0	158	83	Hostler	South	Fem
10	56699	418192	13709776	NTE538	B+	66.0	NaN	NaN	68.0	NaN	...	12.0	2.0	165	94	Day Scholar	North	M
11	56700	418192	13709776	NTE885	E	42.0	0.0	0.0	NaN	82.0	...	0.0	2.0	165	94	Day Scholar	North	M
12	56701	418192	13709776	NTE886	B+	85.0	0.0	70.0	NaN	92.0	...	0.0	2.0	165	94	Day Scholar	North	M

3 rows × 23 columns

Data Filtering from all data set

Making a new csv file to to slect the use full data selecting required data only and which will effect the course difficulty .

```
print("Enter the MHRDName for u want to filter make sure u use whitespace and write exact")
namm = input("Enter the name of required data with proper spaces")
df = pd.read_csv('DATA-FINAL.csv')
if os.path.isfile('My.csv'):
    print("not create again");
else:
    lst =  [['Termid','RegdNo','Course','Grade','CA_100','MTT_50','ETT_100','ETP_100','Course_Att','MHRDName','CA_1','CA_2',
    d = pd.DataFrame(lst)
    for i in range(df.Termid.count()):
        if i == 0:
            d.to_csv('My.csv',mode='a',header=False)
        if df.MHRDName[i] == namm :
            df.iloc[i:i+1,:].to_csv('My.csv',mode='a',header=False)
            print("New file name My.csv is created")
        else:
            pass
```

The basic process of loading data from a CSV file into a Pandas DataFrame is achieved using the "read_csv" function in Pandas:

```
In [56]: df1=pd.read_csv('My.csv')
         df1
```

Out[56]:

	0	Termid	RegdNo	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	...	CA_3	CA_4	Height	Weight	ScholarType	Direction	Gen
0	4036	418192	2088776	NTE538	B+	67.0	NaN	NaN	64.0	NaN	...	4.0	18.0	162	93	Day Scholar	North	M
1	4037	418192	2088776	NTE8	B	57.0	0.0	62.0	NaN	95.0	...	20.0	2.0	162	93	Day Scholar	North	M
2	4038	418192	2088776	NTE9	A+	73.0	NaN	NaN	68.0	100.0	...	0.0	7.0	162	93	Day Scholar	North	M
3	18100	418192	5262776	NTE538	F	16.0	NaN	NaN	0.0	NaN	...	1.0	4.0	171	76	Day Scholar	West	M
4	18101	418192	5262776	NTE8	D	60.0	0.0	49.0	NaN	74.0	...	7.0	7.0	171	76	Day Scholar	West	M
5	18102	418192	5262776	NTE9	E	84.0	NaN	NaN	22.0	86.0	...	2.0	0.0	171	76	Day Scholar	West	M
6	45061	418192	11226776	NTE538	F	0.0	NaN	NaN	0.0	NaN	...	0.0	0.0	167	94	Day Scholar	East	Fem
7	45062	418192	11226776	NTE8	F	0.0	0.0	0.0	NaN	0.0	...	0.0	0.0	167	94	Day Scholar	East	Fem
8	45063	418192	11226776	NTE9	F	0.0	NaN	NaN	0.0	0.0	...	0.0	0.0	167	94	Day Scholar	East	Fem
9	49898	718192	12190776	NTE12	B	61.0	NaN	NaN	62.0	NaN	...	8.0	3.0	158	83	Hostler	South	Fem
10	56699	418192	13709776	NTE538	B+	66.0	NaN	NaN	68.0	NaN	...	12.0	2.0	165	94	Day Scholar	North	M
11	56700	418192	13709776	NTE885	E	42.0	0.0	0.0	NaN	82.0	...	0.0	2.0	165	94	Day Scholar	North	M
12	56701	418192	13709776	NTE886	B+	85.0	0.0	70.0	NaN	92.0	...	0.0	2.0	165	94	Day Scholar	North	M

13 rows × 23 columns

Handling of Missing Data

Real-world data often has missing values.Data can have missing values for a number of reasons such as observations that were not recorded and data corruption.Handling missing data is important as many machine learning algorithms do not support data with missing values.

There are mainly two ways to handle missing data, which are:

By deleting the particular row: The simplest strategy for handling missing data is to remove records that contain a missing value.We can do this by creating a new Pandas DataFrame with the rows containing missing values removed.

Pandas provides the drop() function that can be used to drop either columns or rows with missing data. We can use dropna() to remove all rows with missing data.

```
In [58]: df1=df1.drop(['0','Termid','RegdNo','MHRDName','Direction','ProgramType','Height','Weight','ScholarType','Gender','CourseTy
         df1
```

Out[58]:

	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	CA_1	CA_2	CA_3	CA_4	Medium
0	NTE538	B+	67.0	NaN	NaN	64.0	NaN	14.0	31.0	4.0	18.0	English
1	NTE8	B	57.0	0.0	62.0	NaN	95.0	31.0	4.0	20.0	2.0	English
2	NTE9	A+	73.0	NaN	NaN	68.0	100.0	12.0	54.0	0.0	7.0	English
3	NTE538	F	16.0	NaN	NaN	0.0	NaN	2.0	9.0	1.0	4.0	Hindi
4	NTE8	D	60.0	0.0	49.0	NaN	74.0	7.0	39.0	7.0	7.0	Hindi
5	NTE9	E	84.0	NaN	NaN	22.0	86.0	78.0	4.0	2.0	0.0	Hindi
6	NTE538	F	0.0	NaN	NaN	0.0	NaN	0.0	0.0	0.0	0.0	Regional
7	NTE8	F	0.0	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0	Regional
8	NTE9	F	0.0	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	Regional
9	NTE12	B	61.0	NaN	NaN	62.0	NaN	45.0	5.0	8.0	3.0	English
10	NTE538	B+	66.0	NaN	NaN	68.0	NaN	6.0	46.0	12.0	2.0	English

By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. to impute with mean column

values, these mean column values will need to be stored to file for later use on new data that has missing values.Pandas provides the fillna() function for replacing missing values with a specific value.

```
In [8]: 1 data.fillna(data.mean(),axis=0,inplace=True)
```

Handling of Categorical Data

If you are familiar with machine learning, you will probably have encountered categorical features in many datasets. These generally include different categories or levels associated with the observation, which are non-numerical and thus need to be converted so the computer can process them.

```
In [63]: df1.dtypes

Out[63]: Course         object
         Grade          object
         CA_100        float64
         MTT_50        float64
         ETT_100       float64
         ETP_100       float64
         Course_Att    float64
         CA_1          float64
         CA_2          float64
         CA_3          float64
         CA_4          float64
         Medium         object
         dtype: object
```

Making not null values of all the columns and making null values as zero

```
: df1.isnull().sum()
```

```
: Course        0
  Grade         0
  CA_100        0
  MTT_50        8
  ETT_100       8
  ETP_100       5
  Course_Att    5
  CA_1          0
  CA_2          0
  CA_3          0
  CA_4          0
  Medium        0
  dtype: int64
```

Filling the values if ENTerm and MIDTerm using CA values

10	NTE538	B+	66.0	66.0	NaN	68.0	NaN	6.0	46.0	12.0	2.0	English
11	NTE885	E	42.0	42.0	0.0	NaN	82.0	12.0	28.0	0.0	2.0	English
12	NTE886	B+	85.0	85.0	70.0	NaN	92.0	72.0	11.0	0.0	2.0	English

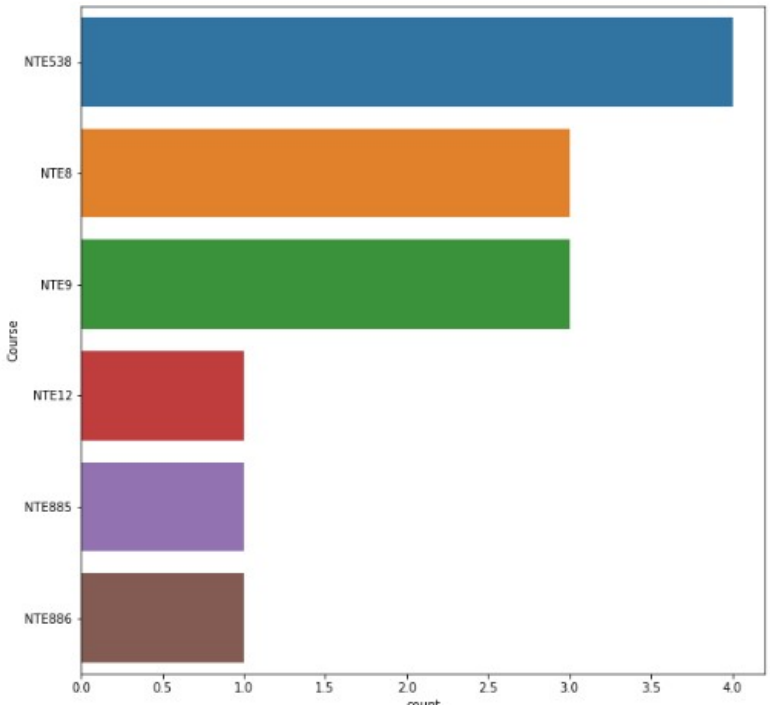
In [68]:
```
df1['EndTerm']=''
new_list=[]
for index,row in df1.iterrows():
    if(pd.isnull(row['ETT_100'])):
        new_list.append(row['ETP_100'])
    elif(pd.isnull(row['ETT_100']) and pd.isnull(row['ETP_100'])):
        new_list.append(0)
    else:
        new_list.append(row['ETT_100'])
```

In [71]:
```
new_list
```

Out[71]:
```
[64.0, 62.0, 68.0, 0.0, 49.0, 22.0, 0.0, 0.0, 0.0, 62.0, 68.0, 0.0, 70.0]
```

Plotting the count of theory and practical subject with Countplot

```
In [20]: plt.figure(1 , figsize = (10 ,10 ))
         sns.countplot(y = 'Course' , data = df1)
         plt.show()
```



Printing all kinds of grades that students have received

```
In [22]: mid_term100=[]
         for index,row in df1.iterrows():
             mid_term100.append((row['MTT_50']/50)*100)
         df1['MTT_100']=mid_term100

In [23]: df1['MTT_100']=mid_term100

In [24]: df1['Grade'].unique()
Out[24]: array(['B+', 'B', 'A+', 'F', 'D', 'E'], dtype=object)
```

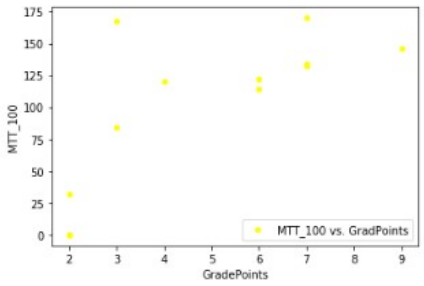
Converting grades to value as to make it use while plotting graph

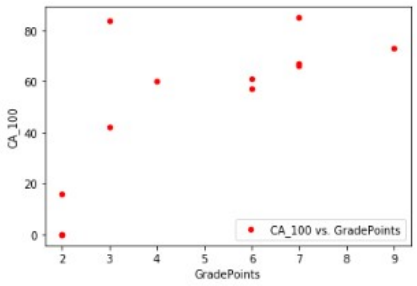
In [25]:
```
df1['GradePoints']=''
new_list_GradePoints=[]
for index,row in df1.iterrows():
    if(row['Grade']=='O'):
        new_list_GradePoints.append(10)
    elif(row['Grade']=='A+'):
        new_list_GradePoints.append(9)
    elif(row['Grade']=='A'):
        new_list_GradePoints.append(8)
    elif(row['Grade']=='B+'):
        new_list_GradePoints.append(7)
    elif(row['Grade']=='B'):
        new_list_GradePoints.append(6)
    elif(row['Grade']=='C'):
        new_list_GradePoints.append(5)
    elif(row['Grade']=='D'):
        new_list_GradePoints.append(4)
    elif(row['Grade']=='E'):
        new_list_GradePoints.append(3)
    elif(row['Grade']=='F'):
        new_list_GradePoints.append(2)
    else:
        new_list_GradePoints.append(1)
```

In [26]:
```
df1['GradePoints']=new_list_GradePoints
```

In [27]:
```
df1.plot(kind="scatter", y="MTT_100",x="GradePoints", color="yellow", label="MTT_100 vs. GradPoints")
df1.plot(kind="scatter",y="CA_100",x="GradePoints", color="red", label="CA_100 vs. GradePoints")
```

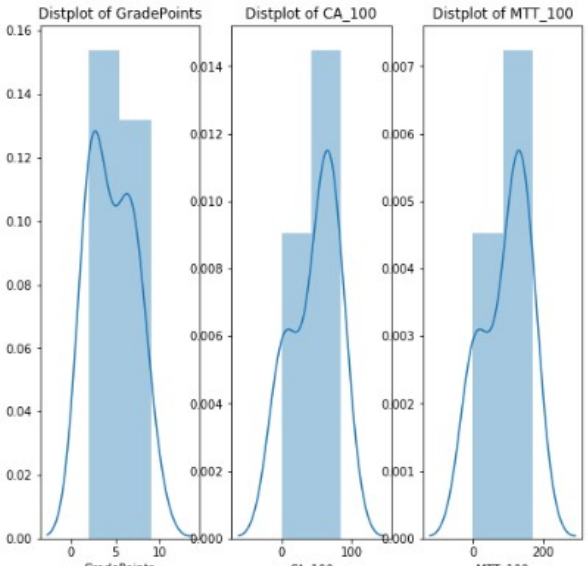
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x2494df70a08>





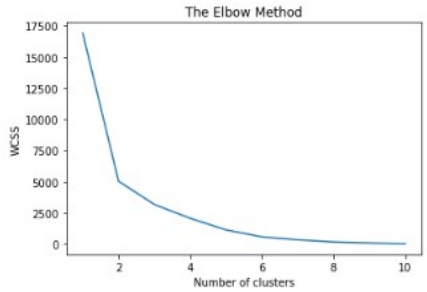
Plotting the distplot to know the skewness of a particular column

```
In [28]: plt.figure(1 , figsize = (17 , 8))
         n = 0
         for x in ['GradePoints','CA_100','MTT_100']:
             n += 1
             plt.subplot(1 , 6 , n)
             sns.distplot(df1[x])
             plt.title('Distplot of {}'.format(x))
         plt.show()
```



Applying KMeans Clustering

```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state=41)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters = 3, init = 'k-means++')
y_kmeans = kmeans.fit_predict(x)
print(y_kmeans)
```

```
[1 1 1 2 1 1 2 0 2 1 1 0 1]
```

```
df_pred = pd.DataFrame(y_kmeans)
```

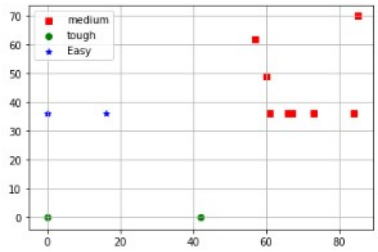
Applying Agglomerative Clustering

In [40]:
```
from sklearn.metrics import silhouette_score
scores['Agglomerative']=silhouette_score(x,y_ac)
print(silhouette_score(x,y_ac))
```

```
0.5906855834241743
```

In [41]:
```
from sklearn.cluster import Birch
model = Birch(threshold=0.01, n_clusters=3)
model.fit(x)
y_bi = model.predict(x)
```

In [42]:
```
pp=np.array(x)
plt.scatter(pp[y_bi==0,0],pp[y_bi==0,1],c='red',marker='s',label='medium')
plt.scatter(pp[y_bi==1,0],pp[y_bi==1,1],c='green',marker='o',label='tough')
plt.scatter(pp[y_bi==2,0],pp[y_bi==2,1],c='blue',marker='*',label='Easy')
plt.legend()
plt.grid()
plt.show()
```



In [43]:
```
from sklearn.metrics import silhouette_score
scores['Birch']=silhouette_score(x,y_bi)
print(silhouette_score(x,y_bi))
```

```
0.5906855834241743
```

RESULT AND CONCLUSION

Here we have now selected the best result of students on the basic of END terms and midterms marks in knn means prediction so we will take the difficulty as per its outcome and divide it as per easy ,medium and hard so we can predict the difficulty of each course. Predicting class of all Student whether that particular subject is easy,medium or tough using KMean as its Silhoutte_Score is same as that of Agglomerative and is high¶

```
output=[]
for index,rows in df1.iterrows():
    if(np.asscalar(kmeans.predict([[rows['MTT_100'],rows['EndTerm']]]))==1):
        output.append('Medium')
    elif(np.asscalar(kmeans.predict([[rows['MTT_100'],rows['EndTerm']]]))==0):
        output.append('Simple')
    else:
        output.append('Hard')
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:5: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  """
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:5: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  """
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
```

```
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\DELL\AppData\Local\Programs\Python\Python37\Lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: np.asscalar(a)
is deprecated since NumPy v1.16, use a.item() instead
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [47]: df1['Difficulty Level of Course']=output

In [48]: df2 = df1[['Course','Difficulty Level of Course']]
         df2.head()

Out[48]:
```

	Course	Difficulty Level of Course
0	NTE538	Medium
1	NTE8	Medium
2	NTE9	Medium
3	NTE538	Simple
4	NTE8	Medium

```
In [ ]:

In [ ]:
```


REFERENCES

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68?gi=e2cb15b48542

https://www.geeksforgeeks.org/clustering-in-machine-learning/

https://towardsdatascience.com/unsupervised-machine-learning-clustering-analysis-d40f2b34ae7e

https://developers.google.com/machine-learning/clustering/clustering-algorithms

https://developers.google.com/machine-learning/clustering

https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

https://machinelearningmastery.com/clustering-algorithms-with-python/

https://data-flair.training/blogs/clustering-in-machine-learning/

http://www.cad.zju.edu.cn/home/zhx/csmath/lib/exe/fetch.php?media=2011:presentation_ml_by_ibrar.pdf