Chapter 1

ABSTRACT

The K-Means clustering algorithm was proposed by Mac Queen in 1967 which is a partition-based cluster analysis method. It is used widely in cluster analysis for that the K-means algorithm has higher efficiency and scalability and converges fast when dealing with large data sets. However it also has many deficiencies: the number of clusters K needs to be initialised, the initial cluster centres are arbitrarily selected, and the algorithm is influenced by the noise points. In view of the shortcomings of the traditional K-Means clustering algorithm, this paper presents an improved K-means algorithm using noise data filter. The algorithm developed density-based detection methods based on characteristics of noise data where the discovery and processing steps of the noise data are added to the original algorithm. By preprocessing the data to exclude these noise data before clustering data set the cluster cohesion of the clustering results is improved significantly and the impact of noise data on K-means algorithm is decreased effectively and the clustering results are more accurate.

Chapter 2

INTRODUCTION

The K-means algorithm is one of the most influential clustering algorithms in the field of data mining. It is widely used in many fields, such as school, daily consumption, transfer, and curriculum arrangement of different student groups. However, the traditional k-means algorithm is relatively sensitive to the initial cluster center, and the clustering result is excessively dependent on the initial center. In order to obtain a more accurate clustering result, we propose a k-means algorithm based on semantic improvement. In this paper, we calculate the mesh density of the sample, set the density threshold to remove the outliers, and divide the core points, boundary points, noise points, optimized clusters according to the grid density of the data points, effectively reduce noise interference, and build the semantic relationship of the data in the cluster and optimizes the selection of the initial cluster center point. The simulation experiment is carried out by using five common datasets provided by the UCI database. The results show that the search method based on the improved k-means algorithm is reduced in the data iteration time compared with the prior art. Improvements have been made in terms of accuracy.

For this project we will attempt to use KMeans Clustering to cluster Universities into two groups, Private and Public.

It is very important to note, we actually have the labels for this data set, but we will NOT use them for the KMeans clustering algorithm, since that is an unsupervised learning algorithm.

When using the K Means algorithm under normal circumstances, it is because you don't have labels. In this case we will use the labels to try to get an idea of how well the algorithm performed, but you won't usually do this for Kmeans, so the classification report and confusion matrix at the end of this project don't truly make sense in a real world setting!.

The Data

We will use a data frame with 777 observations on the following 18 variables.

- Private A factor with levels No and Yes indicating private or public university
- Apps Number of applications received
- Accept Number of applications accepted
- Enroll Number of new students enrolled
- Top 10 Perc Pct. new students from top 10% of H.S. class
- Top 25 Perc Pct. new students from top 25% of H.S. class
- F.Undergrad Number of full time undergraduates
- P.Undergrad Number of part time undergraduates
- Outstate Out-of-state tuition
- Room.Board Room and board costs
- Books Estimated book costs
- Personal Estimated personal spending
- PhD Pct. of faculty with Ph.D.’s
- Terminal Pct. of faculty with terminal degree
- S.F.Ratio Student/faculty ratio
- perc.alumni Pct. alumni who donate
- Expend Instructional expenditure per student
- Grad.Rate Graduation rate

Chapter 3

SOFTWARE REQUIREMENTS

Python :

- Python is a high-level, general-purpose programming language.
- Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Libraries :

- **numpy :-**
- **pandas :-**
- **matplotlib :-**
- **seaborn :-**
- **sklearn :-**

Operating System :

- Program is tested on Windows 10

Hardware Requirements Specification :

- Laptop / Computer

Chapter 4

K MEANS ALGORITHM

K Means algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way k means algorithm works is as follows:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach K Means follows to solve the problem is called Expectation Maximization. The step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a breakdown of how we can solve it mathematically (feel free to skip it).

The objective function is

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

where wik=1 for data point xi if it belongs to cluster k; otherwise, wik=0. Also, μk is the centroid of xi's cluster.

It's a minimization problem of two parts. We first minimize J w.r.t. wik and treat μk fixed. Then we minimize J w.r.t. μk and treat wik fixed. Technically speaking, we differentiate J w.r.t. wik first and update cluster assignments (E-step). Then we differentiate J w.r.t. μk and recompute the centroids after the cluster assignments from previous step (M-step). Therefore, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

In other words, assign the data point xi to the closest cluster judged by its sum of squared distance from cluster's centroid.

And M-step is

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}} \qquad (3)$$

Which translates to recomputing the centroid of each cluster to reflect the new assignments.

Few things to note here:

- Since clustering algorithms including k means use distance-based measurements to determine the similarity between data points, it's recommended to standardize the data to have a mean of zero and a standard deviation of one since almost always the features in any dataset would have different units of measurements such as age vs income.
- Given k means iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since k means algorithm may be stuck in a local optimum and may not converge to global optimum. Therefore, it's recommended to run the algorithm using different initializations of centroids and pick the results of the run that yielded the lower sum of squared distance.
- Assignment of examples isn't changing is the same thing as no change in within-cluster variation:

$$\frac{1}{m_k} \sum_{i=1}^{m_k} \left\| x^i - \mu_{c^k} \right\|^2 \qquad (4)$$

Chapter 4

IMPLEMENTING K MEANS ALGORITHM

4.1 Importing Libraries

We are going to use the Python programming language for this task of implementing K Means Algorithm for creating cluster for private and government colleges so let's start by importing some necessary libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Now let's import the data and move further:

```
data = pd.read_csv("College_Data.csv")
data.head(10)
```

	College_name	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.R
0	Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	
1	Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	
2	Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	
3	Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	
4	Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	
5	Albertson College	Yes	587	479	158	38	62	678	41	13500	3335	500	675	67	73	
6	Albertus Magnus College	Yes	353	340	103	17	45	416	230	13290	5720	500	1500	90	93	
7	Albion College	Yes	1899	1720	489	37	68	1594	32	13868	4826	450	850	89	100	
8	Albright College	Yes	1038	839	227	30	63	973	306	15595	4400	300	500	79	84	
9	Alderson-Broaddus College	Yes	582	498	172	21	44	799	78	10468	3380	660	1800	40	41	

4.2 Data analysis

Before creating clusters of private and government colleges we need to observe and analyse the data to see what we are going to work with. The goal here is to learn more about the data

find answers to some important questions such as:

- What question (s) are you trying to solve?
- What kind of data do we have and how do we handle the different types?
- What is missing in the data and how do you deal with it?
- Where are the outliers and why should you care?
- How can you add, change, or remove features to get the most out of your data?

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   College_name  777 non-null    object 
 1   Private       777 non-null    object 
 2   Apps          777 non-null    int64  
 3   Accept        777 non-null    int64  
 4   Enroll        777 non-null    int64  
 5   Top10perc     777 non-null    int64  
 6   Top25perc     777 non-null    int64  
 7   F.Undergrad   777 non-null    int64  
 8   P.Undergrad   777 non-null    int64  
 9   Outstate      777 non-null    int64  
 10  Room.Board    777 non-null    int64  
 11  Books         777 non-null    int64  
 12  Personal      777 non-null    int64  
 13  PhD           777 non-null    int64  
 14  Terminal      777 non-null    int64  
 15  S.F.Ratio     777 non-null    float64
 16  perc.alumni   777 non-null    int64  
 17  Expend        777 non-null    int64  
 18  Grad.Rate     777 non-null    int64  
dtypes: float64(1), int64(16), object(2)
memory usage: 115.5+ KB
```

Here we dropping “College Name” Column because it have all static values, so for better visualization we are removing “College Name” column

```
cdata=data.drop(['College_name'], axis = 1)
cdata.head()
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio
0	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1
1	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2
2	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12.9
3	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7
4	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9

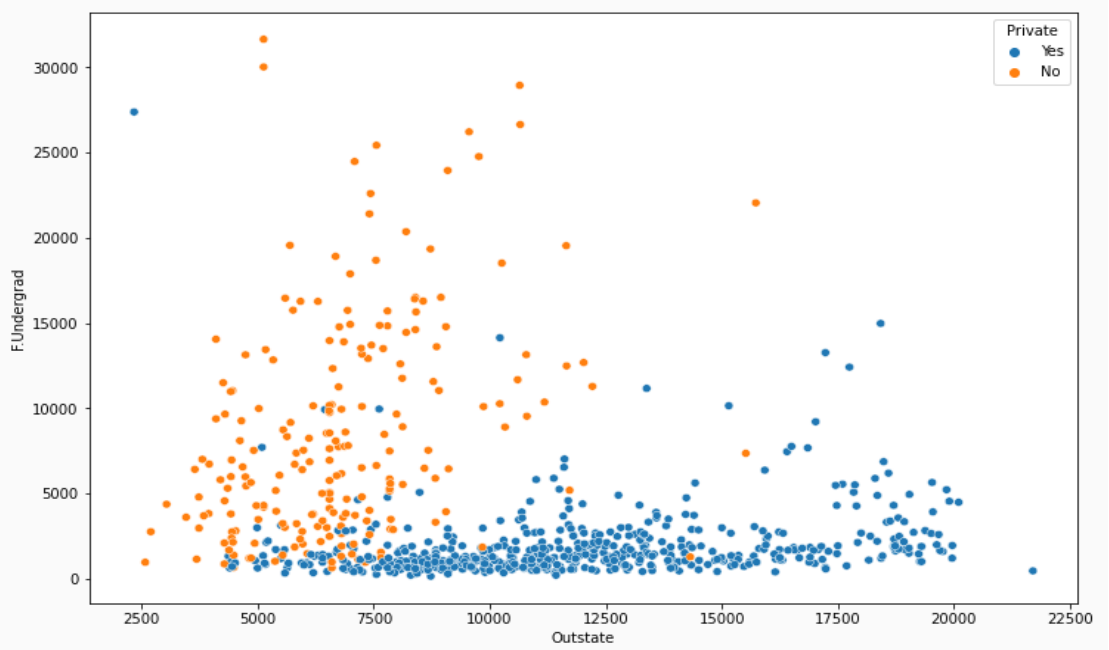
In it we finding if any column have null value, if have then fill it with ‘0’

```
cdata.isnull().sum()
```

```
Private        0
Apps           0
Accept         0
Enroll         0
Top10perc      0
Top25perc      0
F.Undergrad    0
P.Undergrad    0
Outstate       0
Room.Board     0
Books          0
Personal       0
PhD            0
Terminal       0
S.F.Ratio      0
perc.alumni    0
Expend         0
Grad.Rate      0
dtype: int64
```

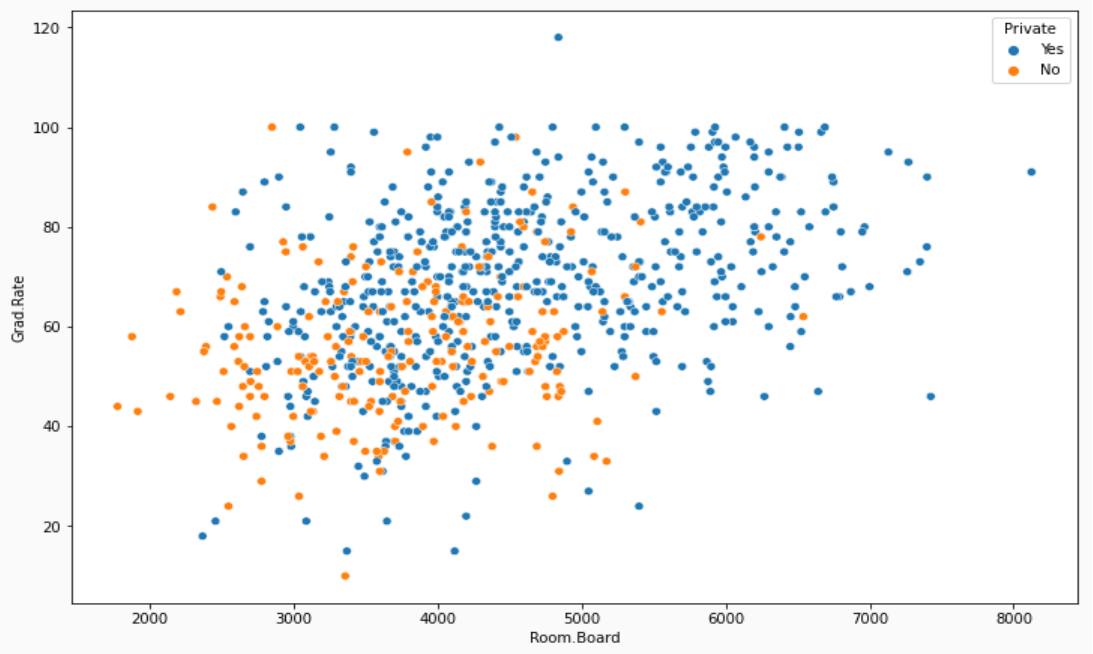
Here we plotting Scatter plot graph which shows how many private colleges and government colleges are provide full time undergraduate and Out-of-state tuition facility

```
plt.figure(figsize=(12,8))
sns.scatterplot(data=cdata,x='Outstate',y='F.Undergrad',hue='Private')
plt.show()
```



And this scatter represents graduation rate with respect to room board cost

```
plt.figure(figsize=(12,8))
sns.scatterplot(data=cdata,x = 'Room.Board', y = 'Grad.Rate',hue='Private')
plt.show()
```



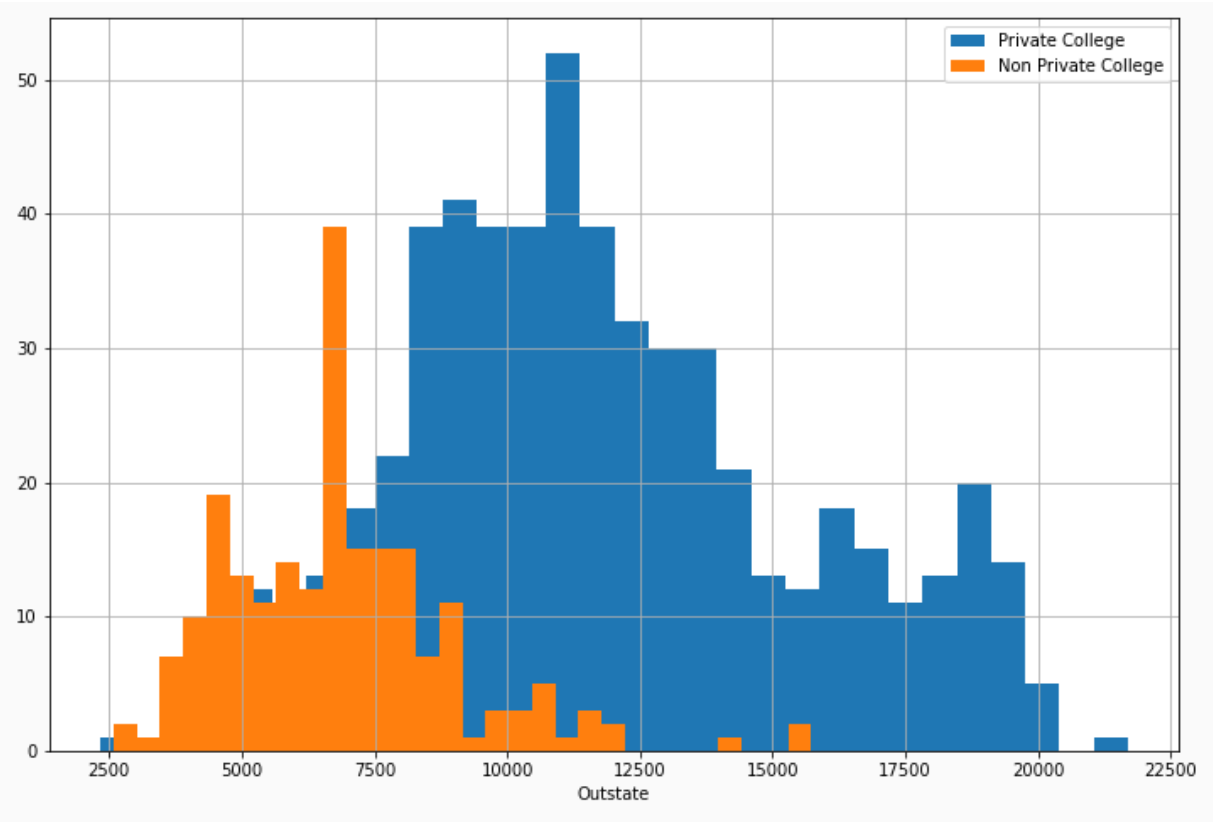
Here we plotting histogram of private and government college for outstate

```
plt.figure(figsize=(12, 8))

cdata.loc[cdata.Private == 'Yes', 'Outstate'].hist(label="Private College", bins=30)
cdata.loc[cdata.Private == 'No', 'Outstate'].hist(label="Non Private College", bins=30)

plt.xlabel('Outstate')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x238c8614ca0>
```



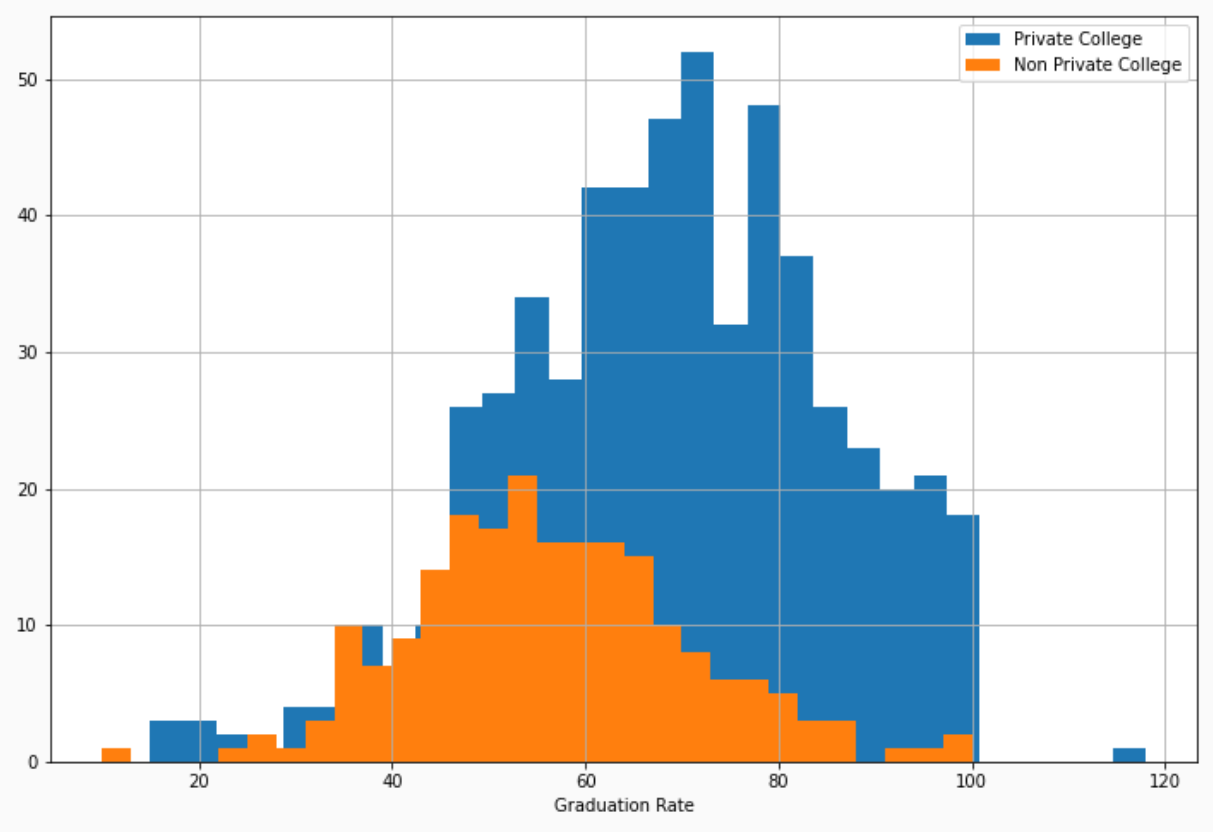
Here we plotting histogram of private and government college for graduation rate

```
plt.figure(figsize=(12, 8))

cdata.loc[cdata.Private == 'Yes', 'Grad.Rate'].hist(label="Private College", bins=30)
cdata.loc[cdata.Private == 'No', 'Grad.Rate'].hist(label="Non Private College", bins=30)

plt.xlabel('Graduation Rate')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x238c840b280>
```



Importing KMeans from sklearn library , Create an instance of a K Means model with 2 clusters.

K Means Cluster

```
from sklearn.cluster import KMeans
kmeans = KMeans(2)
```

```
kmeans.fit(cdata.drop('Private', axis=1))
```

```
KMeans(n_clusters=2)
```

```
kmeans.cluster_centers_
```

```
array([[1.03631389e+04, 6.55089815e+03, 2.56972222e+03, 4.14907407e+01,
        7.02037037e+01, 1.30619352e+04, 2.46486111e+03, 1.07191759e+04,
        4.64347222e+03, 5.95212963e+02, 1.71420370e+03, 8.63981481e+01,
        9.13333333e+01, 1.40277778e+01, 2.00740741e+01, 1.41705000e+04,
        6.75925926e+01],
       [1.81323468e+03, 1.28716592e+03, 4.91044843e+02, 2.53094170e+01,
        5.34708520e+01, 2.18854858e+03, 5.95458894e+02, 1.03957085e+04,
        4.31136472e+03, 5.41982063e+02, 1.28033632e+03, 7.04424514e+01,
        7.78251121e+01, 1.40997010e+01, 2.31748879e+01, 8.93204634e+03,
        6.51195815e+01]])
```

We have ‘Yes’ and ‘No’ static value in column ‘Private’, so we can’t use that value for clustering or further calculation, to solve this problem we add new column ‘cluster’ with ‘1’ and ‘0’, cluster column have 1 value when private column have ‘Yes’ , and cluster column have 0 when private column have ‘No’ value

```
cdata['Cluster'] = cdata['Private'].apply(lambda x: 1 if x == 'Yes' else 0)
cdata
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.a
0	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1	
1	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2	
2	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12.9	
3	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7	
4	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
772	No	2197	1515	543	4	26	3089	2029	6797	3900	500	1200	60	60	21.0	
773	Yes	1959	1805	695	24	47	2849	1107	11520	4960	600	1250	73	75	13.3	
774	Yes	2097	1915	695	34	61	2793	166	6900	4200	617	781	67	75	14.4	
775	Yes	10705	2453	1317	95	99	5217	83	19840	6510	630	2115	96	96	5.8	
776	Yes	2989	1855	691	28	63	2988	1726	4990	3560	500	1250	75	75	18.1	

777 rows × 19 columns

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(cdata['Cluster'], kmeans.labels_))
```

```
[[ 74 138]
 [ 34 531]]
```

```
print(classification_report(cdata['Cluster'], kmeans.labels_))
```

```
              precision    recall  f1-score   support

           0       0.69      0.35      0.46       212
           1       0.79      0.94      0.86       565

    accuracy                           0.78       777
   macro avg       0.74      0.64      0.66       777
weighted avg       0.76      0.78      0.75       777
```

```
print(accuracy_score(cdata.Cluster, kmeans.labels_))
```

```
0.7786357786357786
```

Chapter 5

CONCLUSION

We successfully built a cluster using the k mean algorithm in Machine Learning.

Chapter 6

REFERENCES

Books :-

Research issues on K-means Algorithm: An Experimental Trial Using Matlab by Joaquin Perez Ortega, Ma. Del

Rocio Boone Rojas and Maria J. Somodevilla Garcia.

The k-means algorithm - Notes by Tan, Steinbach, Kumar Ghosh.

Website :-

https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm

https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

https://towardsdatascience.com/k-means-clustering-of-university-data-9e8491068778

https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

https://www.kaggle.com/faressayah/k-means-clustering-private-vs-public-universities