Name: Ankit Sharma

Email: kumarankitx022@gmail.com

Mob no.: +91-7677241423

Week: 04

ASSIGNMENT – 10

TOPIC: K-NEAREST NEIGHBORS (K-NN)

Q.1. Distances which we can use in K-NN model and also learn about mathematical formula for those distances.

Q.2. Alter the n_neighbors value and see the difference in the output and also change the type of distance by altering the p value of the distance metric.

Ans. 1) A number of Machine Learning Algorithms - Supervised or Unsupervised, use Distance Metrics to know the input data pattern in order to make any Data Based decision. A good distance metric helps in improving the performance of Classification, Clustering and Information Retrieval process significantly.

Below are the commonly used distance metrics –

Minkowski Distance: Minkowski distance is a metric in Normed vector space. What is Normed vector space? A Normed vector space is a vector space on which a norm is defined. Suppose X is a vector space then a norm on X is a real valued function $||x||$which satisfies below conditions -

1. **Zero Vector-** Zero vector will have zero length.
2. **Scalar Factor-** The direction of vector doesn't change when you multiply it with a positive number though its length will be changed.
3. **Triangle Inequality-** If distance is a norm then the calculated distance between two points will always be a straight line.

The distance can be calculated using below formula -

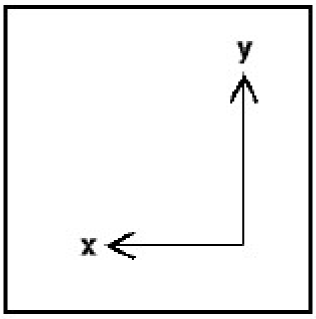
$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Minkowski distance is the generalized distance metric. Here generalized means that we can manipulate the above formula to calculate the distance between two data points in different ways.

As mentioned above, we can manipulate the value of *p* and calculate the distance in three different ways-
p = 1, Manhattan Distance
p = 2, Euclidean Distance
p = ∞, Chebychev Distance

Manhattan Distance:
We use Manhattan Distance if we need to calculate the distance between two data points in a grid like path. As mentioned above, we use Minkowski distance formula to find Manhattan distance by setting p’s value as 1.

Let’s say, we want to calculate the distance, d, between two data points: x and y.



Manhattan

Distance d will be calculated using an absolute sum of difference between its cartesian co-ordinates as below :

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

where, n- number of variables, xi and yi are the variables of vectors x and y respectively, in the two dimensional vector space. i.e.

x = (x1,x2,x3,...) and y = (y1,y2,y3,…).

Now the distance d will be calculated as-

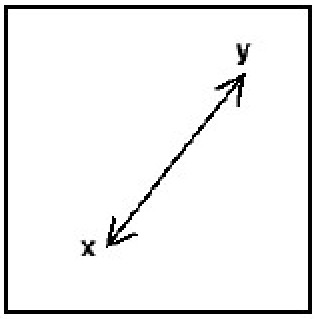
(x1 - y1) + (x2 - y2) + (x3 - y3) + … + (xn - yn).

Euclidean Distance:
Euclidean distance is one of the most used distance metric. It is calculated using Minkowski Distance formula by setting ***p's*** value to ***2***. This will update the distance ***'d'*** formula as below :

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Let's stop for a while! Does this formula look familiar? Well yes, we just saw this formula above in this article while discussing ***"Pythagorean Theorem".***

Euclidean distance formula can be used to calculate the distance between two data points in a plane.



Euclidean

Cosine Distance:
Mostly Cosine distance metric is used to find similarities between different documents. In cosine metric we measure the degree of angle between two documents/vectors(the term frequencies in different documents collected as metrics). This particular metric is used when the magnitude between vectors does not matter but the orientation.

Cosine similarity formula can be derived from the equation of dot products :-

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Now, you must be thinking which value of cosine angle will be helpful in finding out the similarities.

$$\text{Cos } 0' = 1 \qquad \text{Cos } 90' = 0$$
$$\text{Cos } 180' = -1$$

Now that we have the values which will be considered in order to measure the similarities, we need to know what do 1, 0 and -1 signify.

Here cosine value 1 is for vectors pointing in the same direction i.e. there are similarities between the documents/data points. At zero for orthogonal vectors i.e. Unrelated(some similarity found). Value -1 for vectors pointing in opposite directions(No similarity).

Q. no. 2) Alter the n_neighbors value and see the difference in the output and also change the type of distance by altering the p value of the distance metric

Using k ranging from 1 to 10 and distance p = 1 (Manhattan)

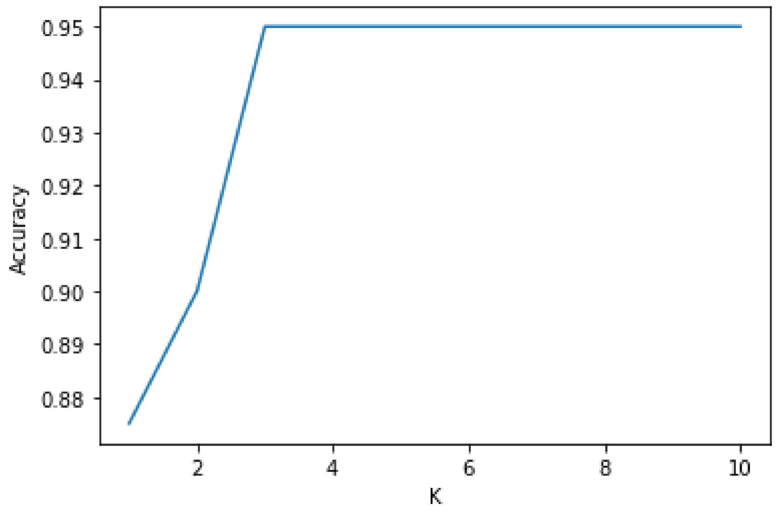
In [20]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
acc = []
k_rng = range(1,11)
for k in k_rng:
    clf = KNeighborsClassifier(n_neighbors = k, p =1)
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    acc.append(metrics.accuracy_score(y_test, y_pred))
```

In [21]:

```
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.plot(k_rng,acc)
print(acc)
```

```
[0.875, 0.9, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95]
```



Using k ranging from 1 to 10 and distance p = 2 (Euclidean)

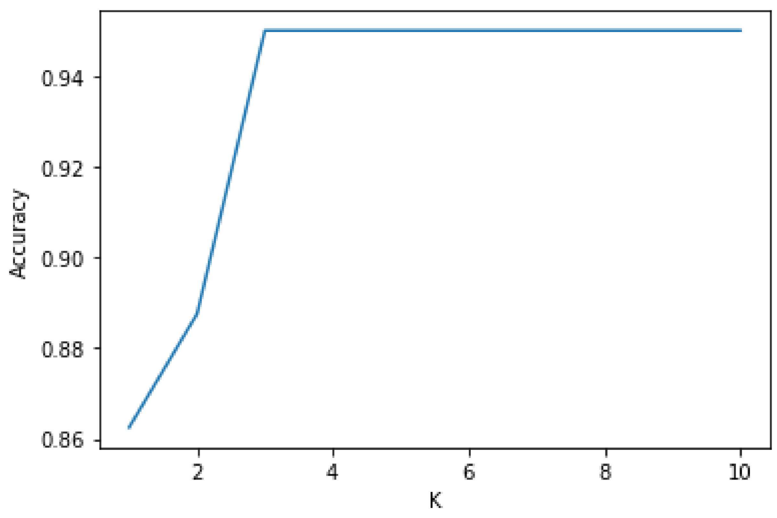
In [22]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
acc1 = []
k_rng1 = range(1,11)
for k in k_rng1:
    clf1 = KNeighborsClassifier(n_neighbors = k, p =2)
    clf1.fit(X_train,y_train)
    y_pred1 = clf1.predict(X_test)
    acc1.append(metrics.accuracy_score(y_test, y_pred1))
```

In [23]:

```
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.plot(k_rng1,acc1)
print(acc1)
```

```
[0.8625, 0.8875, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95]
```



ASSIGNMENT – 11

TOPIC: K-MEANS

Q.1. To study and prepare notes for k-means++ algorithm.

Q.2. Find out other methods we can use to choose the number of clusters for the k-means.

Q.3. Try to use k-means algorithm with different combinations of the columns for the datasets.

Ans. 1) Drawback of standard K-means algorithm:
One disadvantage of the K-means algorithm is that it is sensitive to the initialization of the centroids or the mean points. So, if a centroid is initialized to be a "far-off" point, it might just end up with no points associated with it and at the same time more than one clusters might end up linked with a single centroid. Similarly, more than one centroids might be initialized into the same cluster resulting in poor clustering. For example, consider the images shown below.

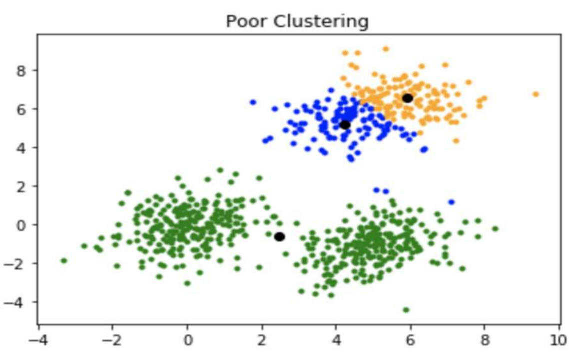


Fig (a)

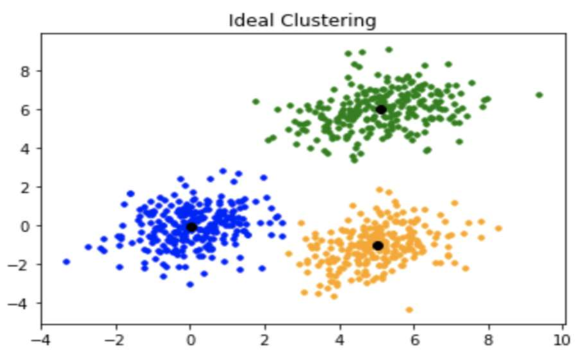


Fig (b)

Fig (a): Shows a poor initialisation of centroids resulted in poor clustering.

Fig (b): This is how the clustering should have been and ideal clustering:

K-mean++:
To overcome the above-mentioned drawback we use K-means++. This algorithm ensures a smarter initialization of the centroids and improves the quality of the clustering. Apart from initialization, the rest of the algorithm is the same as the standard K-means algorithm. That is K-means++ is the standard K-means algorithm coupled with a smarter initialization of the centroids.

Intuition:
By following the above procedure for initialization, we pick up centroids which are far away from one another. This increases the chances of initially picking up centroids that lie in different clusters. Also, since centroids are picked up from the data points, each centroid has some data points associated with it at the end.

Note: Although the initialization in K-means++ is computationally more expensive than the standard K-means algorithm, the run-time for convergence to optimum is drastically reduced for K-means++. This is because the centroids that are initially chosen are likely to lie in different clusters already.

Ans. 2) Different methods we can use to choose the number of clusters for the k-means:

Elbow Method:
Elbow method gives us an idea on what a good *k* number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids. We pick *k* at the spot where SSE starts to flatten out and forming an elbow.

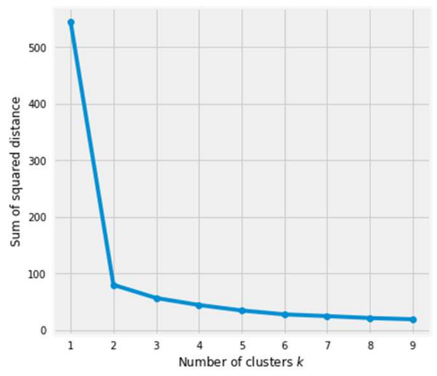


Fig (a) – Elbow at 2

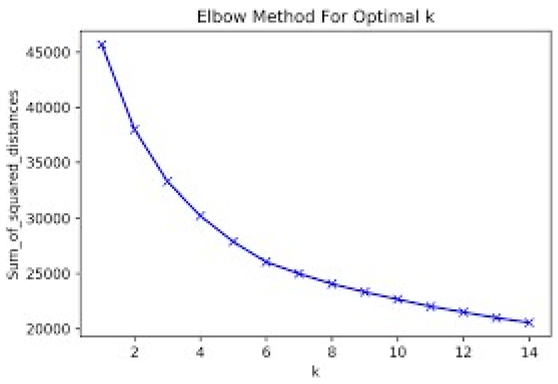


Fig (b) – No elbow

The graph, Fig (a) above shows that k=2 is not a bad choice. Sometimes it's still hard to figure out a good number of clusters to use because the curve is monotonically decreasing and may not show any elbow (as in case of Fig b) or has an obvious point where the curve starts flattening out.

Silhouette Analysis:
Silhouette analysis can be used to determine the degree of separation between clusters. For each sample:

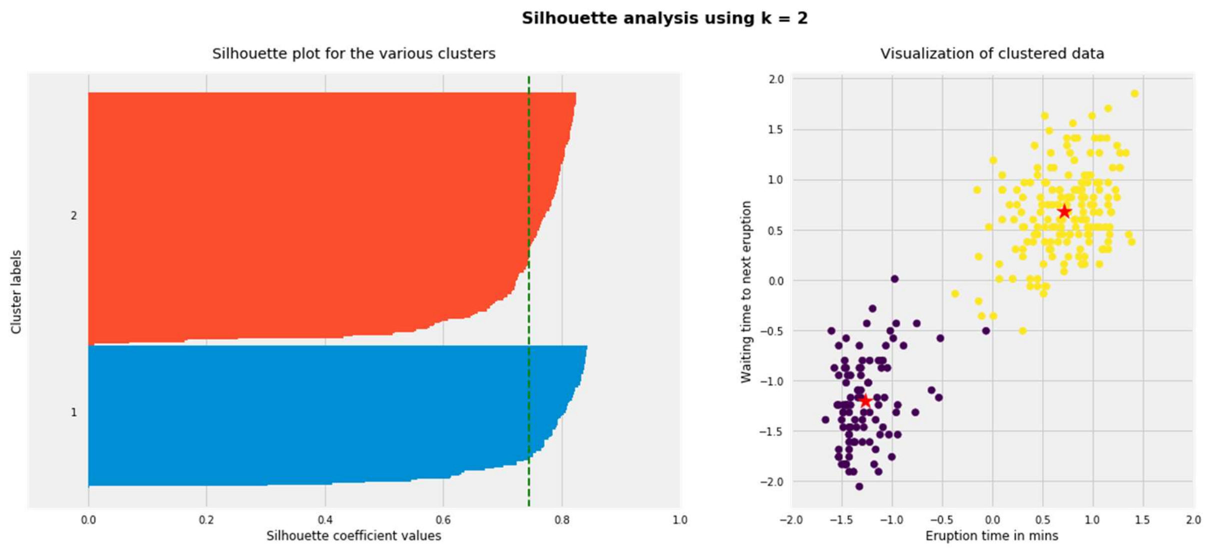
- Compute the average distance from all data points in the same cluster (ai).
- Compute the average distance from all data points in the closest cluster (bi).
- Compute the coefficient:

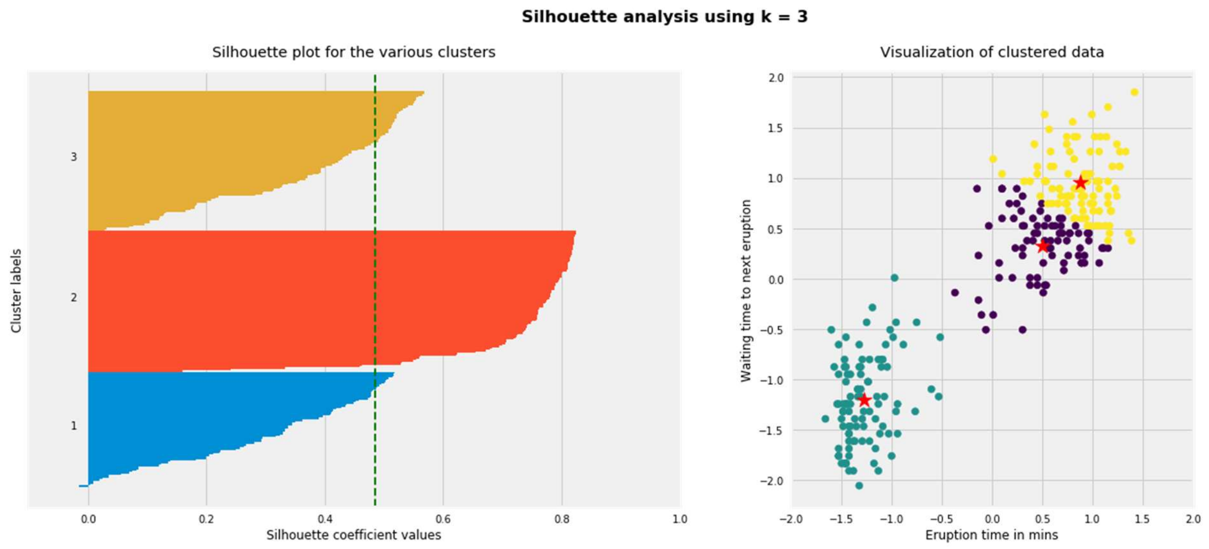
$$\frac{b^i - a^i}{max(a^i, b^i)}$$

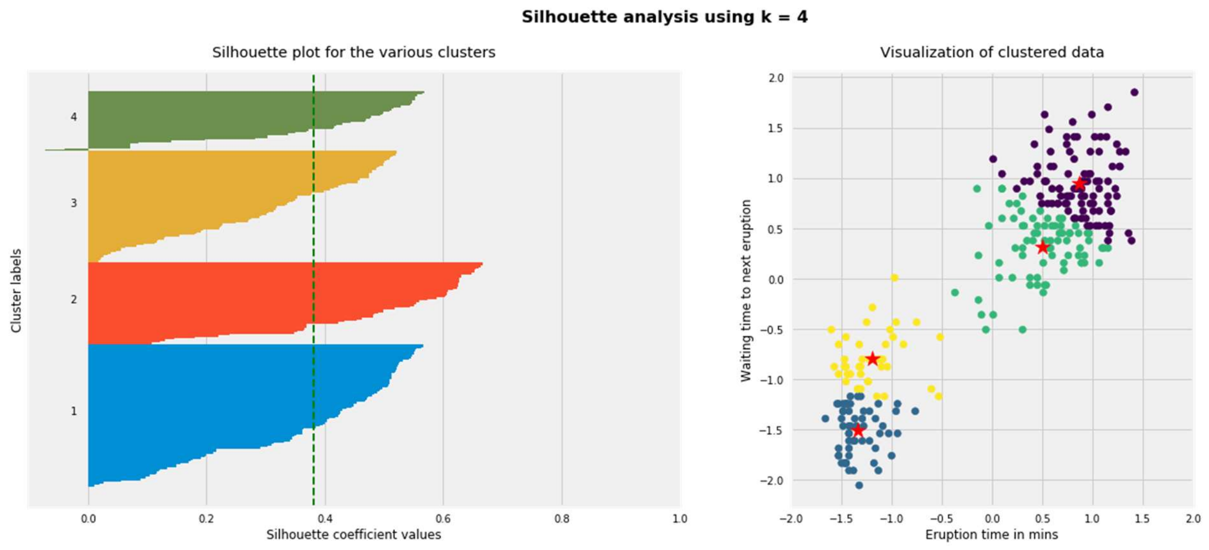
The coefficient can take values in the interval [-1, 1].

- If it is 0 –> the sample is very close to the neighboring clusters.
- It it is 1 –> the sample is far away from the neighboring clusters.
- It it is -1 –> the sample is assigned to the wrong clusters.

Therefore, we want the coefficients to be as big as possible and close to 1 to have a good clusters.

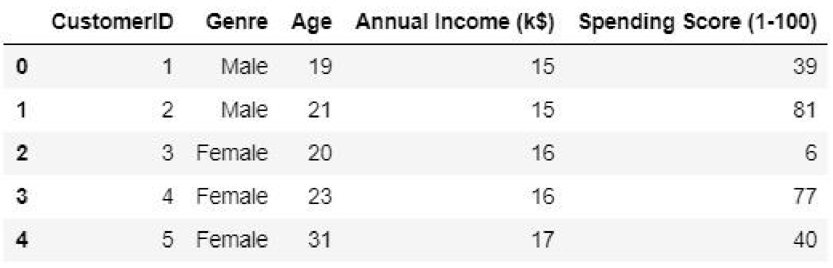






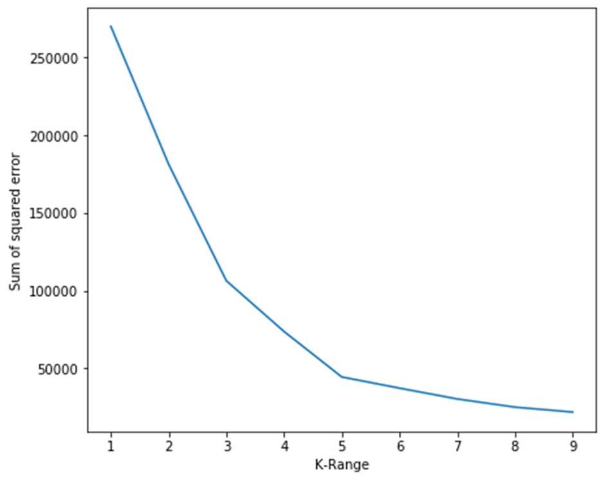
As the above plots show, n_clusters=2 has the best average silhouette score of around 0.75 and all clusters being above the average shows that it is actually a good choice. Also, the thickness of the silhouette plot gives an indication of how big each cluster is. The plot shows that cluster 1 has almost double the samples than cluster 2. However, as we increased n_clusters to 3 and 4, the average silhouette score decreased dramatically to around 0.48 and 0.39 respectively. Moreover, the thickness of silhouette plot started showing wide fluctuations. The bottom line is: Good n_clusters will have a well above 0.5 silhouette average score as well as all of the clusters have higher than the average score.

Ans. 3) Given dataset: (Represnted only 5 rows)

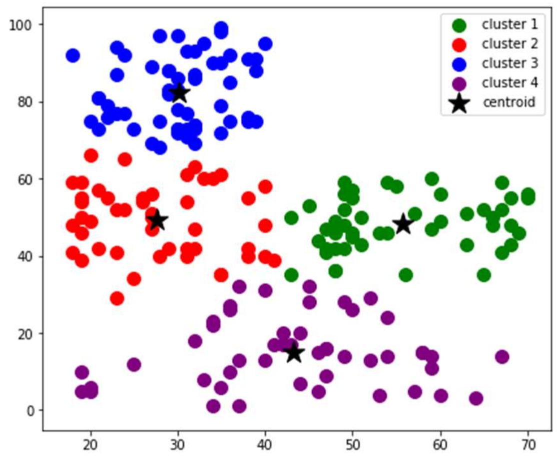


	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Case 1: Considering columns Annual Income (k$) and Spending Score (1-100)



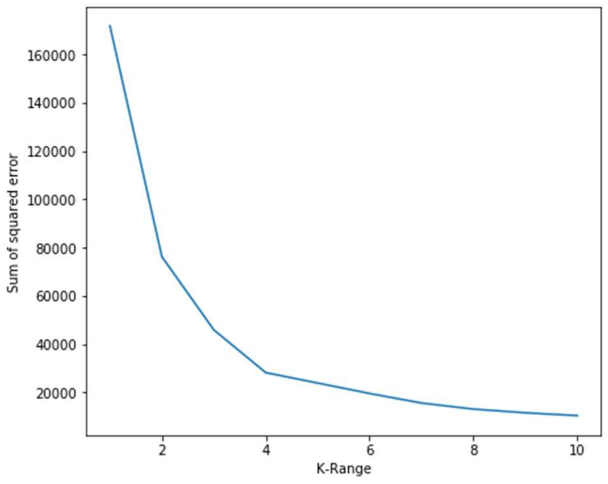
Fig(a)

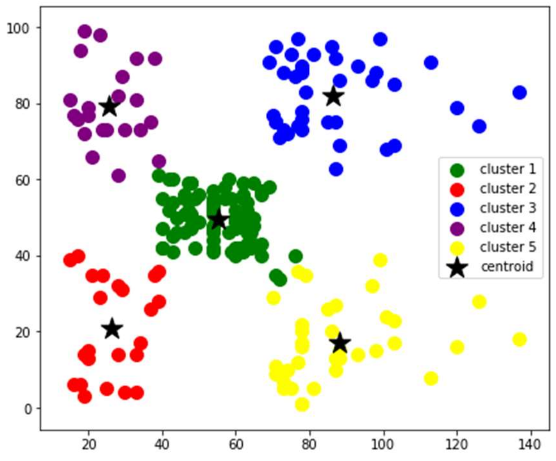


Fig(b)

Here, in Fig (a), we see that after using elbow method k = 5 is the best value for number of clusters and then using n_cluster = 5 we plot a graph with 5 clusters with their respective centroid as shown in the Fig (b)

Case 2: Considering columns Age and Spending Score (1-100)





Fig(c) **Fig(d)**

Here, in Fig (c), we see that after using elbow method k = 4 is the best value for number of clusters and then using n_cluster = 4 we plot a graph with 4 clusters with their respective centroid as shown in the Fig (d)

ASSIGNMENT – 12

TOPIC: <u>REINFORCEMENT LEARNING</u>

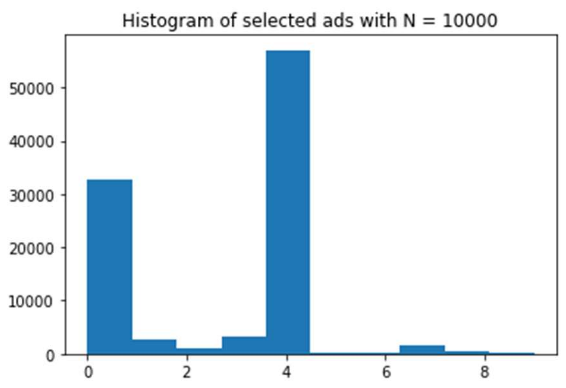
Q.1. Experiment with N and try to figure out the minimum number of rounds it took to select the 5th ad(index 4 in the graph) for every change in N make a histogram for it in our original code we used N = 10000, is to reduce the number of N and try to figure in what minimum no. of rounds was our UCB able to identify the most selected ad.

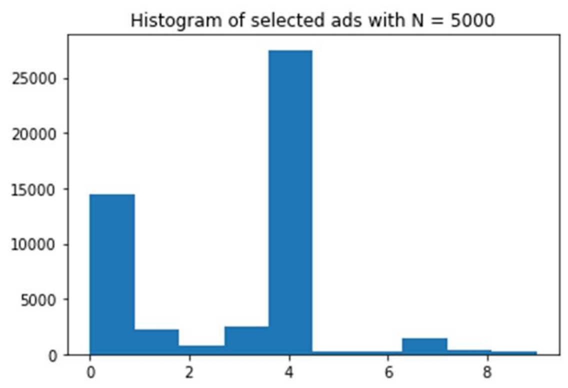
Do not do this N = 9000 N = 8000 N = 7000 YOU CAN DO THIS N = 5000 N = 2500 N = 1250 N = 600 N = 300

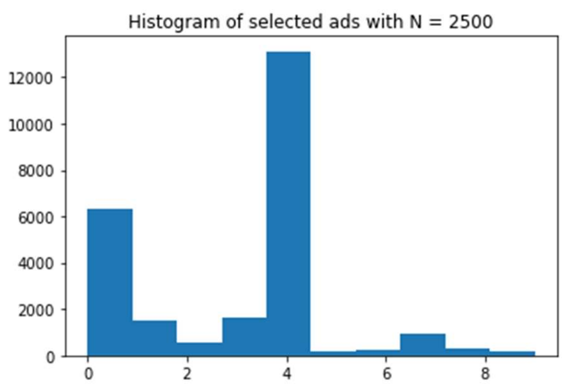
Ans. 1) By plotting a histogram for different values of N i.e.,

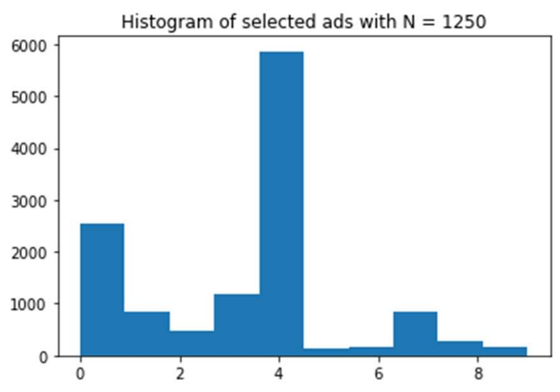
- **10000**
- **5000**
- **1250**
- **600**
- **300**
- **150**
- **75**

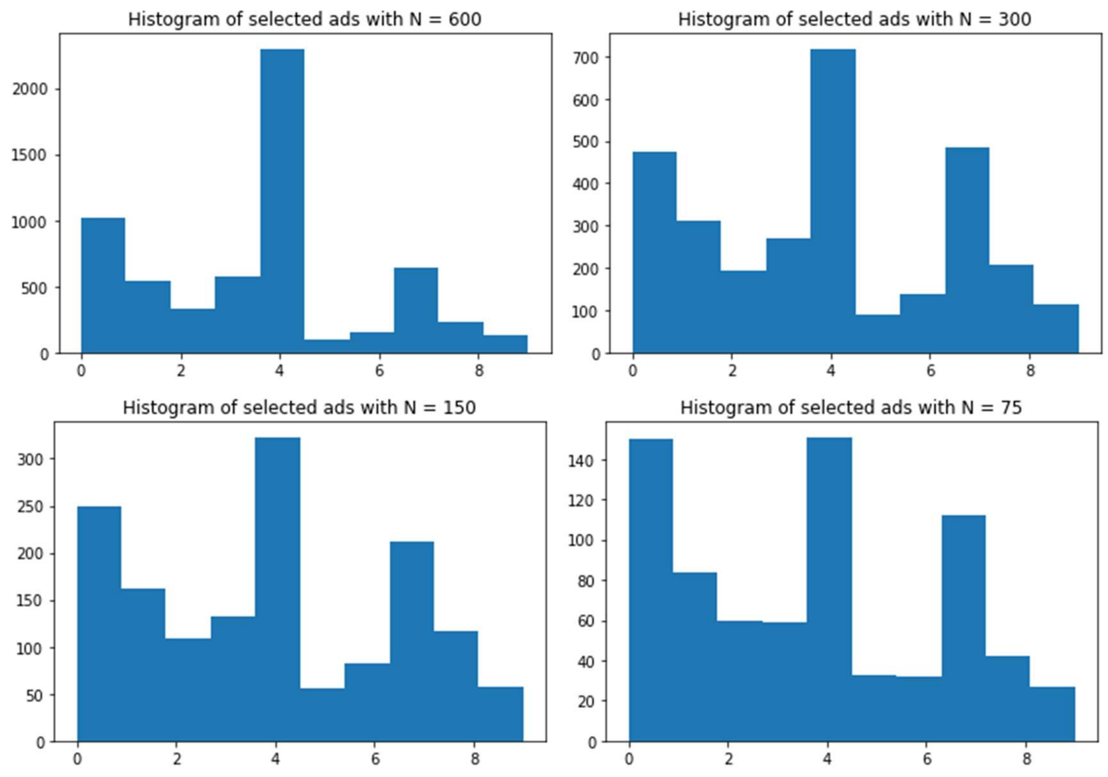
We can identify at what minimum rounds was our UCB able to identify the 5th ad(index 4 in the graph) as the most selected ad.











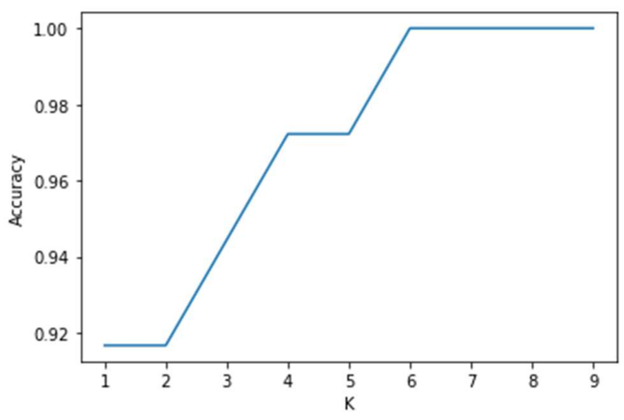
We see that till 150 rounds our UCB was able to identify the 5th ad (index 4 in the graph) as the most selected ad. And Also at 75 rounds it identifies ad 1 ad and 4 as equally selected.

ASSIGNMENT – 13

TOPIC: PRINCIPAL COMPONENT ANALYSIS (PCA)

Q.1. Try the same PCA with KNN model and check its accuracy.

Ans. 1) Before Applying PCA to our KNN model:



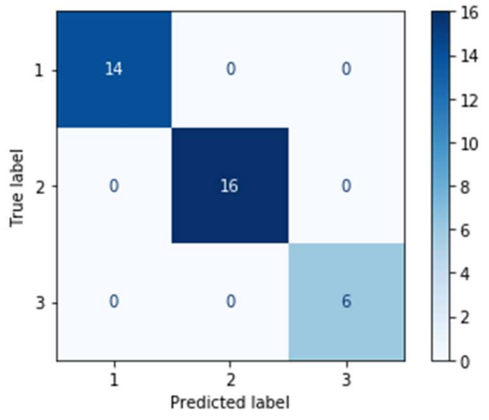


Fig (a)- K vs Accuracy **Fig (b)- Confusion matrix for k = 6**

Our KNN model shows that for k = 6 onwards it's having accuracy score 1.00 which is 100%. And below that the accuracy is different for every different values of k ranging from 1 to 5.

After Applying PCA to our KNN model:

PCA: PCA is considered to be one of the most used unsupervised algorithms and can be seen as the most popular dimensionality reduction algorithm.

After anlysis of our pca.explained_variance_ratio_ i.e. ,

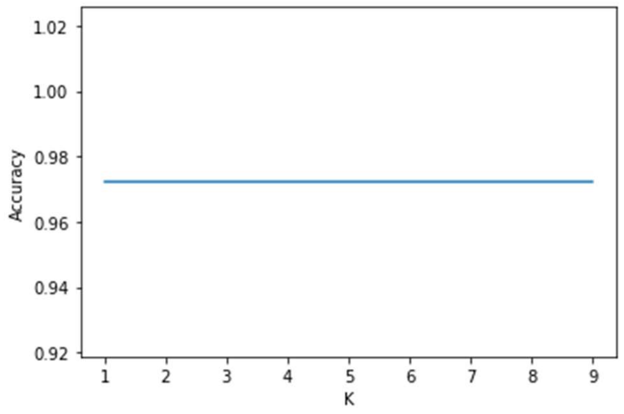
```
[0.36722576 0.19231879 0.10830194 0.07414597 0.06288414 0.05059778
 0.0419487  0.02518069 0.02222384 0.01858596 0.01712304 0.01277985
 0.00668354]
```

These 13 results that we get are the explained_variance_ratio_, out of these we can see that the first and the second results **0.36722576 0.19231879** consists of more of the percentage of the data. And since plotting the data in 2D space is easier to undestand we'll use these two values which is total **0.55954439** of the total data i.e., approx **56 %.**

Since, now that we that our these two are the most essential or we can say the principal components of our data. We can now train our KNN model.

After training our KNN model we get an accuracy score of **0.97222222** i.e., **97.22%.** Also our KNN model shows that (Fig c.) that for any value of K we are getting same accuracy result.

Result after Applying PCA to our KNN model:



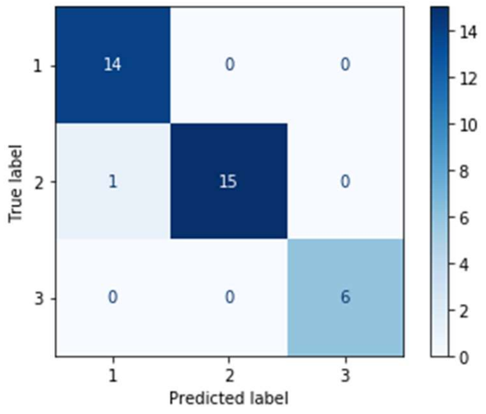
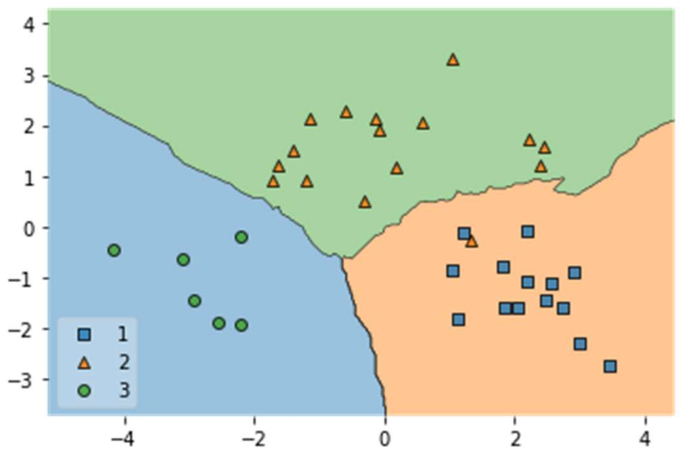


Fig (c)- K vs Accuracy with PCA **Fig (d)- Confusion matrix for any k**



Fid (e) – Decision Region with PCA KNN Model