University of Windsor

GENG-8900 MACHINE LEARNING

Instructor: Dr. Yasser Alginahi

K-Means Clustering

Harshkumar Zaveri

Aeshita Dhiman

Gurpreet Singh Saini

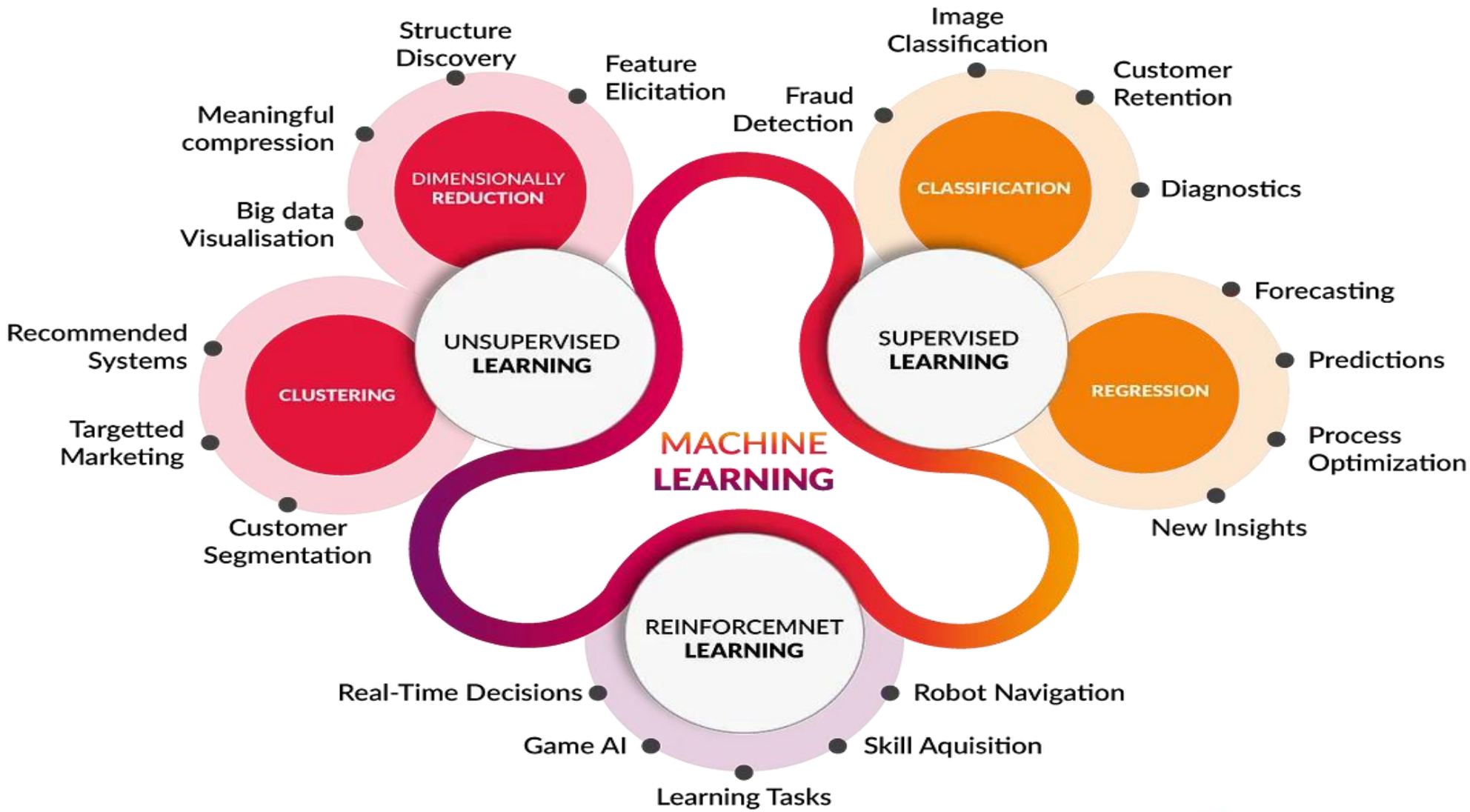
Date: 03-Nov-2023

University of Windsor

Overview

- Types of Machine Learning
- Clustering
- What is K-means clustering
- Key Terminologies
- K means algorithm
- Evaluate quality of clusters
- Advantages and Disadvantages
- Implementation and code
- Applications
- References

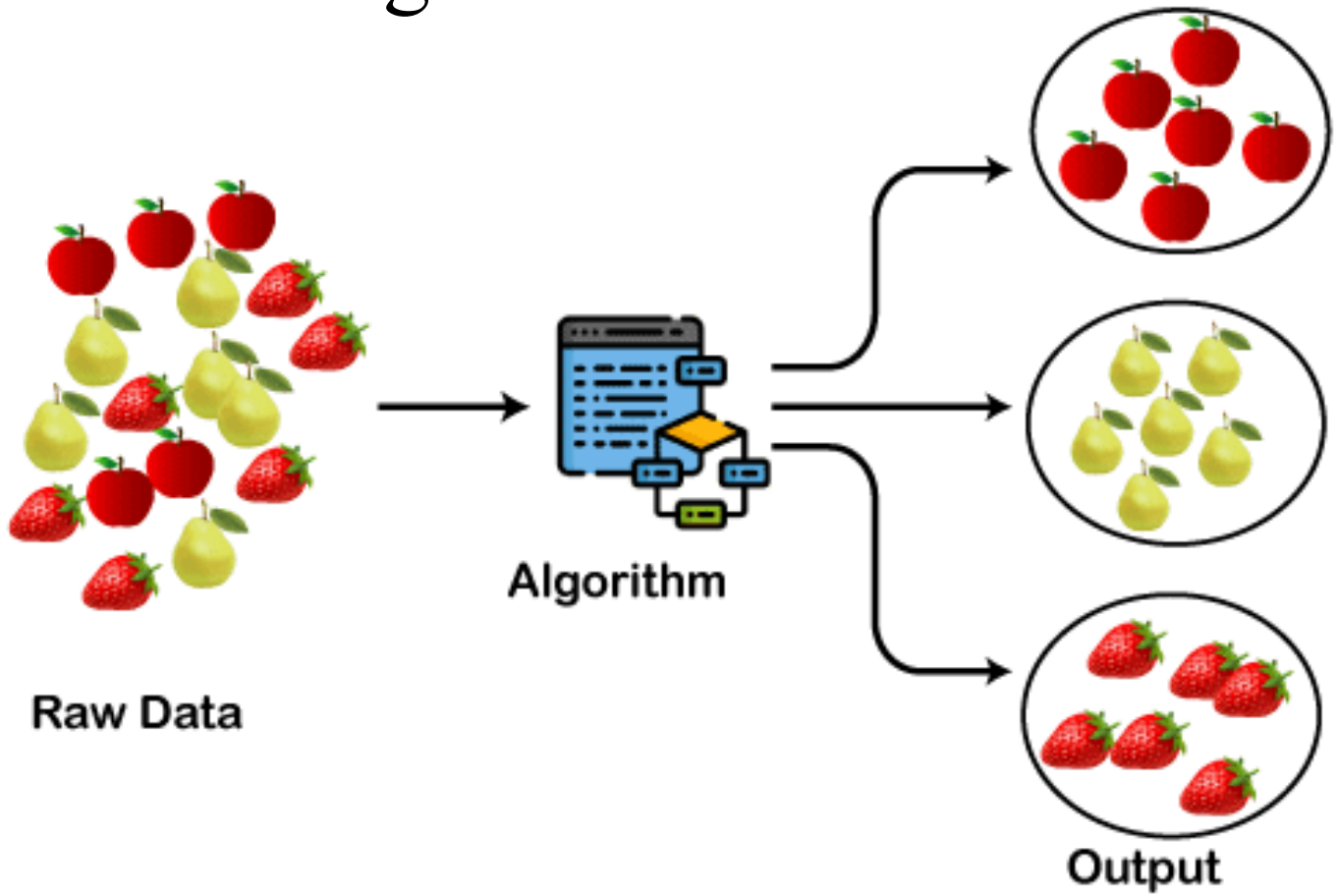
Types of Machine Learning



Types of Machine Learning [https://shorturl.at/pzNT9]

Clustering

- Clustering is the process of arranging a group of objects in such a manner that the objects in the same group are more similar to each other than to the objects in any other group



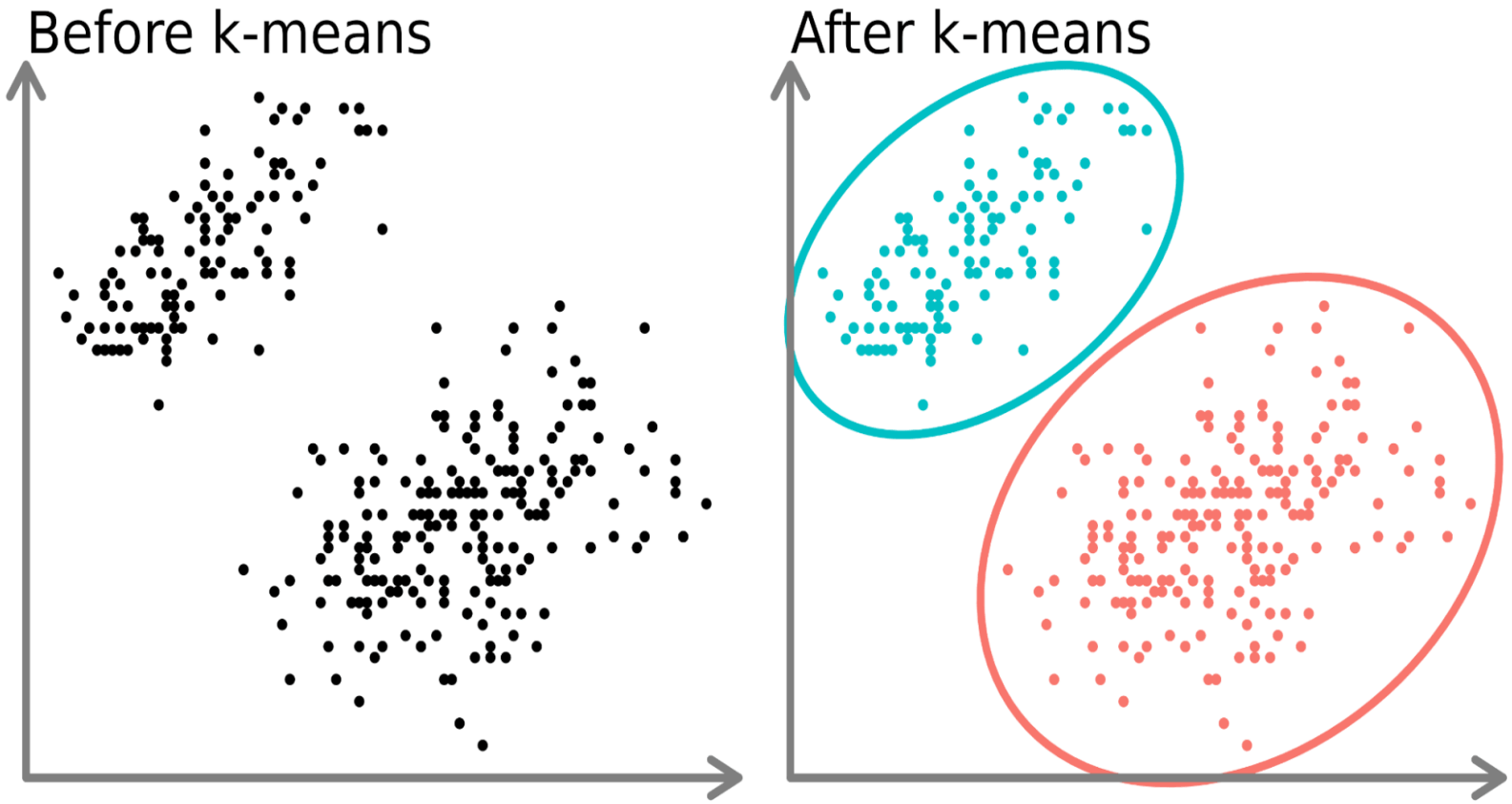
Clustering [https://shorturl.at/hpyDE]

Clustering

- 5 essential clustering algorithms
  - K-Means
  - MeanShift
  - DBSCAN
  - Hierarchical clustering
  - BIRCH

What is K-mean clustering?

- K-means clustering is an unsupervised machine learning algorithm that partitions a dataset into K clusters
- Such models aim to assign similar data points into distinct clusters or groups



K-means clustering [https://shorturl.at/kvPS6]

Key Terminologies

- In ideal conditions for clustering to perform well the inter-cluster distance should be very high and the intra-cluster distance should be very low.
- Based on these two distances we can measure the performance of clustering techniques using the Dunn index.
- The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance.

Inter-cluster and Intra-cluster distance [https://shorturl.at/fzBU2]

Key Terminologies

- **Data Points**: These are individual data instances or observations that are grouped into clusters based on their similarities.
- **Centroids**: Centroids are the central points of each cluster. They represent the mean or average of the data points within that cluster.
- **Clusters**: Clusters are groups of data points that are similar to each other and dissimilar to data points in other clusters. Each data point belongs to one and only one cluster.
- **K-value**: K refers to the number of clusters you want to create in the dataset. It is a hyperparameter that you must choose before running the K-means algorithm.

Keywords [https://shorturl.at/fzBU2]

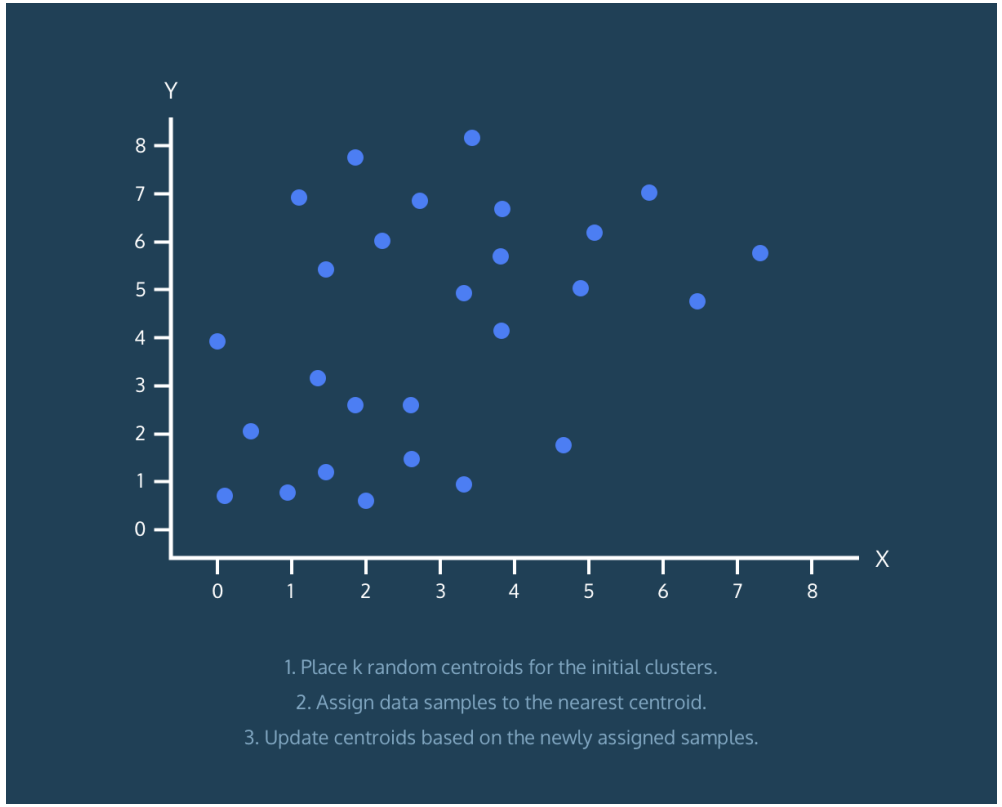
Key Terminologies

- This algorithm takes the data points and gives the predefined set of clusters. Consider dataset D={x_1,x_2……,x_n} has n points with k clusters(S_1,S_2..S_k) and each cluster assigned with one centroid(C_1,C_2, C_3..C_k).
- Every point in the dataset is assigned to the set (Si) depending on the nearest centroid to it.
- The centroid(C_i) is the mean point to all the other points in the given set of points (S_i). It is calculated as below:

$$Ci = \frac{1}{n} \sum_{Xj \epsilon Si} Xj$$

K-Means Algorithm

- Step 1: Identify number of clusters
- Step 2: Initialize Cluster Centroids
- Step 3: Assign Datapoints to Centroids
- Step 4: Re-evaluate cluster mean
- Step 5: Repeat steps 3 – 4 till convergence or a certain number of iterations



K-means Algorithm[https://shorturl.at/ouMOW]

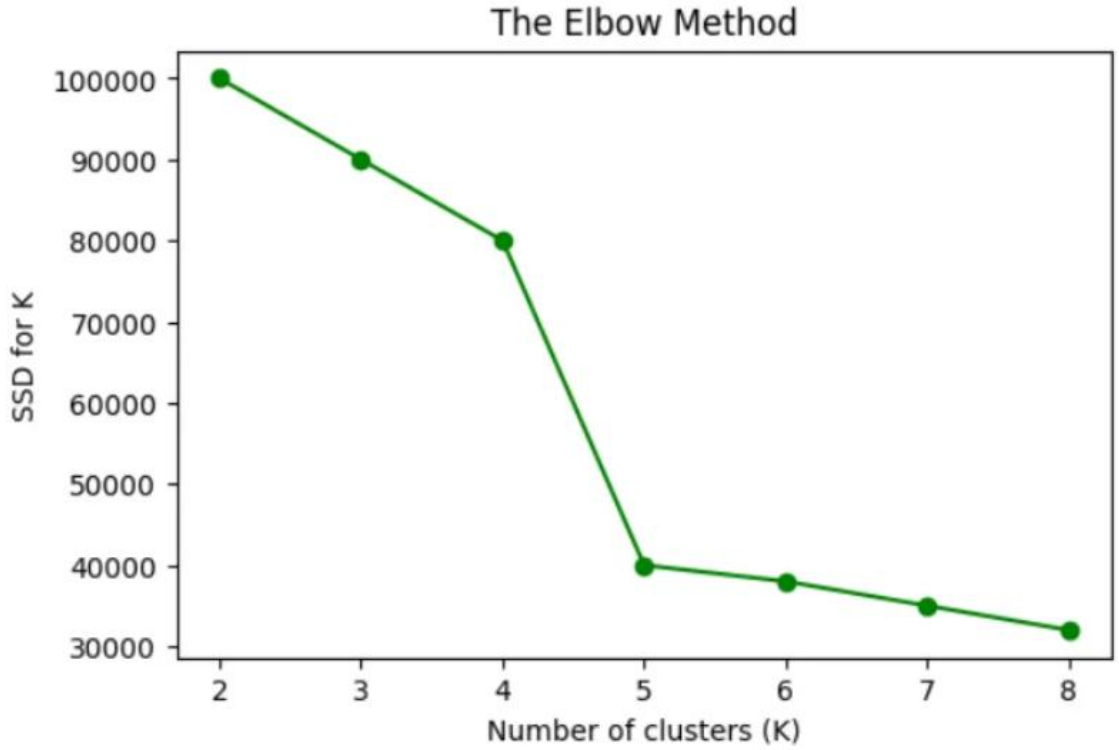
K-Means Algorithm

Identify number of clusters

- Elbow Method
  - Find optimal number of clusters using Sum of Squared Distances (SSD)

- Methodology
  - Start with a defined range of K
  - Run the KMeans Algorithm on your dataset
  - Calculate SSD – Euclidean distance
    - The sum of the squared distance from centroid.
    - $d = (x_1 - c_1)^2 + (x_2 - c_2)^2$
  - Sum up the squares
  - Repeat for all clusters

$$\arg min_{(c1,c2...ck)} = \sum_{i=1}^{K} \sum_{x \in Si} ||x - Ci||^2$$



Elbow Method [https://shorturl.at/glJM8]

K-Means Algorithm

Assign initial Centroids

- At Random
  - Basic method

- KMeans ++
  - Smart centroid initialization
  - Initial centroid assigned randomly
  - Rest of the centroids based on maximum squared distance to spread out centroids

K-Means Iteration [https://shorturl.at/jwWZ7]

K-Means Algorithm

Assign data points to clusters

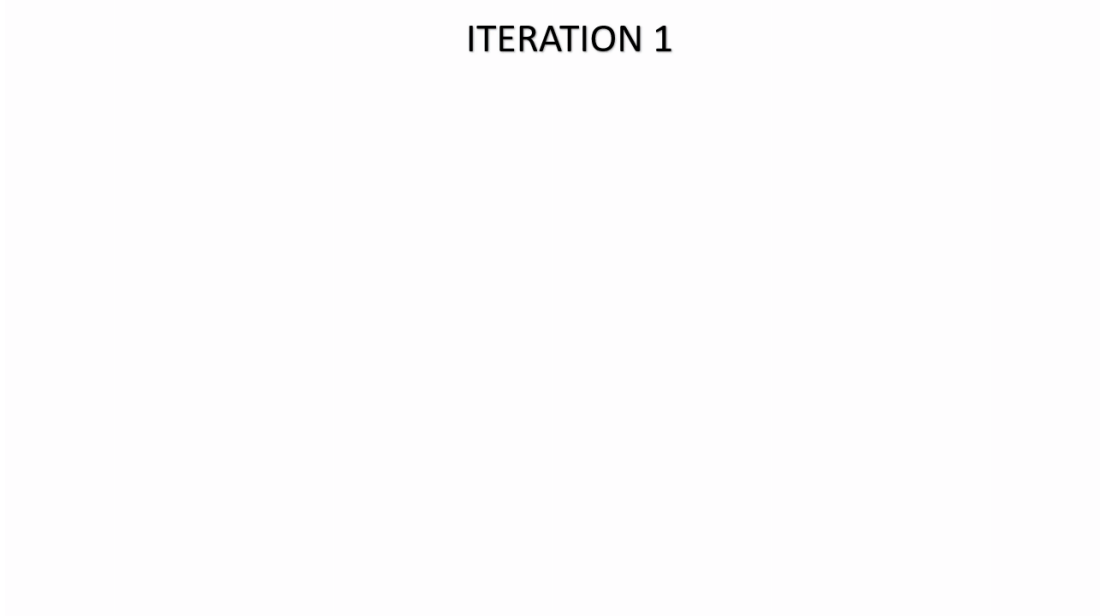
- Using distance measures like Euclidian distance to find the nearest centroid to the data point

- What Happens if same distance?
  - Choose any cluster at random

K-Means Iteration [https://shorturl.at/jwWZ7]

K-Means Algorithm

Assign data points to nearest centroid to create clusters



K-Means Iteration [https://shorturl.at/jwWZ7]

K-Means Algorithm

Update centroids and clusters

- Recalculate the centroid for each cluster using :
  - $c_j = \frac{1}{|s_j|} \sum_{x_i \in s_j} x_i$

- Reassign data points to the new centroids

K-Means Iteration [https://shorturl.at/jwWZ7]

K-Means Algorithm

Repeat data point assignment and centroid update until convergence

- Convergence :The state where there is a very small update in the new centroids over the old centroids

K-Means Iteration [https://shorturl.at/jwWZ7]

Evaluate quality of Cluster

- **Inertia**
  - It tells us how far the points within a cluster are
  - the sum of distances of all the points within a cluster from the centroid of that cluster
  - Euclidean distance is often used as the distance metric
  - We calculate this for all the clusters; the final inertial value is the sum of all these distances
  - the distance between them should be as low as possible
- **Dunn Index**
  - Dunn index is the ratio of the minimum of inter-cluster distances and maximum of intra-cluster distances.
  - The more the value of the Dunn index, the better the clusters will be

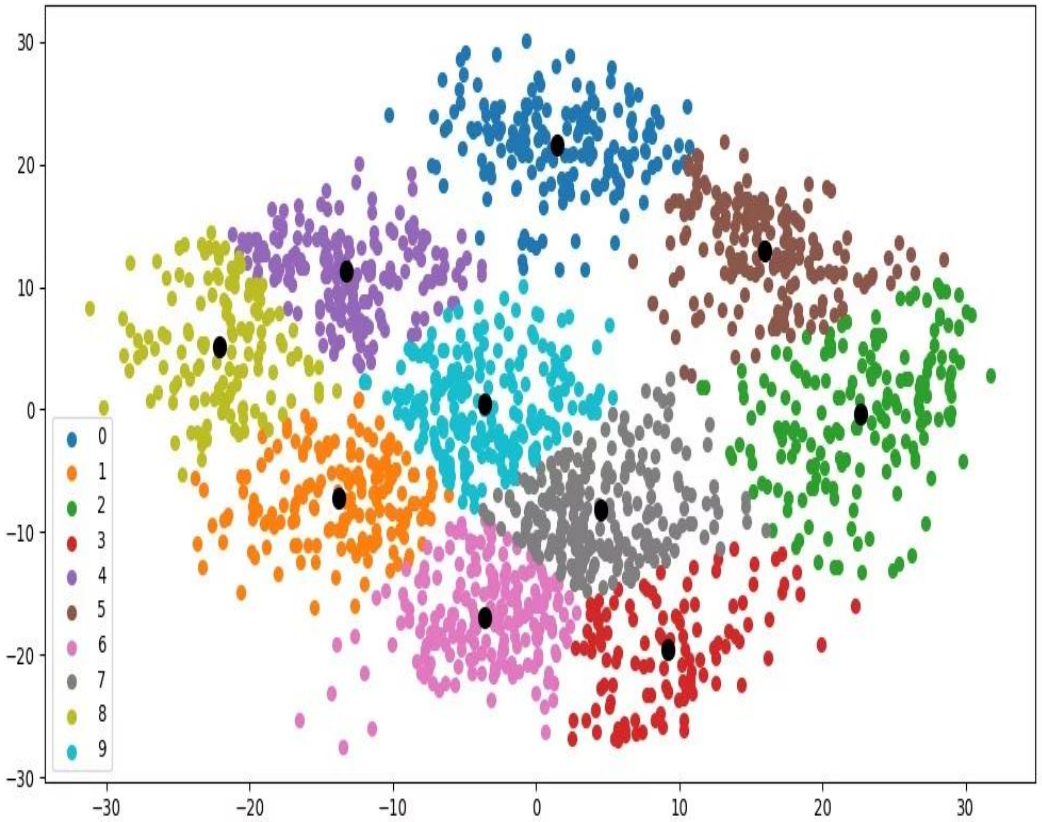
$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\max(\text{Intra cluster distance})}$$

Clusters are far apart

Clusters are compact

Advantages

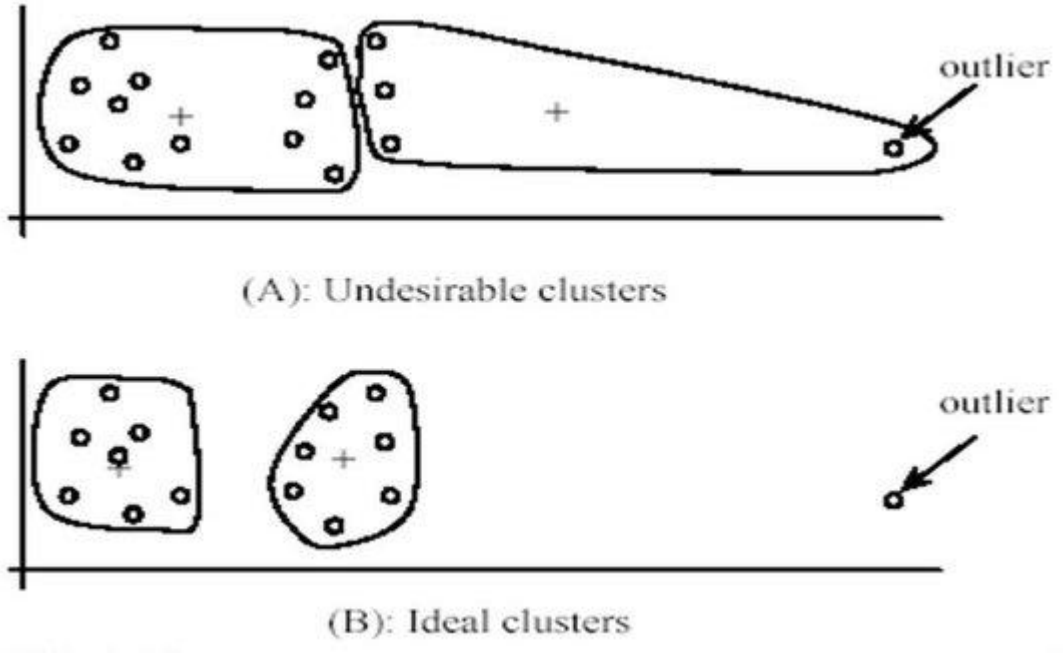
- Easy to implement
- Result can be Interpreted seamlessly
- Produce consistent results
- Can adapt to new examples
- The biggest advantage: UNSUPERVISED LEARNING



K-mean clustering graph [https://tinyurl.com/2s3zsdrx]

Disadvantages

- Choosing K manually
- Clustering outliers
- Different random seeds can cause the algorithm to converge to different local minima instead of the global minimum
- It assumes that clusters are spherical and have similar variance



Effect of outlier on clustering [https://tinyurl.com/c29dyzc4]

Implementing K-Mean on a real dataset

- The dataset is from an automobile company, which have divided there existing customer into four groups and want to predict the group for new customer.
- The dataset have following columns: Unique ID, Gender, Marital status, Age, Graduated, profession, Work experience, Credit score, Family size and category group
- What are we going to do? Remove the category that's already defined and cluster it by ourself.

Code!!!

```
Import pandas as pd
from sklearn.cluster import Kmeans

df = pd.read_csv("Train.csv")
df = df.drop(["Segmentation", "ID"], axis="columns")
df=df.dropna()
df = df.reset_index()
df = df.drop("index", axis="columns")
df_new = pd.get_dummies(df)
df_kmeans = pd.get_dummies(df, drop_first=True)
kmeans_model = KMeans(n_clusters=3)
clusters = kmeans_model.fit_predict(df_kmeans)
df_kmeans.insert(df_kmeans.columns.get_loc("Age"), "Cluster", clusters)
df_kmeans.sample(5, random_state=44)
```

K-mean Implementation and code [http://bit.ly/49kADke]

Implementation

Read the dataset

```
df = pd.read_csv("Train.csv")
df.sample(5, random_state=44)
```

	ID	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1	Segmentation
1627	466681	Male	No	19	No	Healthcare	2.0	Low	4.0	Cat_6	D
805	462758	Male	No	18	No	Healthcare	0.0	Low	3.0	Cat_6	D
3725	467212	Female	No	28	Yes	Doctor	1.0	Low	3.0	Cat_6	B
6581	460636	Male	Yes	45	Yes	Artist	2.0	Average	3.0	Cat_3	D
7462	459157	Male	No	37	Yes	Engineer	1.0	Low	1.0	Cat_6	D

Implementation

Remove columns which we do not need

```
df = df.drop(["Segmentation", "ID"], axis="columns")
df.sample(5, random_state =40)
```

	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1
3757	Female	No	38	No	Homemaker	13.0	Low	NaN	Cat_6
4692	Female	Yes	45	Yes	Artist	1.0	High	5.0	Cat_6
5629	Male	Yes	80	Yes	Lawyer	NaN	Low	1.0	Cat_6
341	Male	Yes	51	No	Entertainment	1.0	Average	4.0	Cat_7
6150	Female	Yes	50	Yes	Artist	1.0	Average	3.0	Cat_6

Implementation

Detail information about the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8068 entries, 0 to 8067
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Gender           8068 non-null   object 
 1   Ever_Married     7928 non-null   object 
 2   Age              8068 non-null   int64  
 3   Graduated        7990 non-null   object 
 4   Profession       7944 non-null   object 
 5   Work_Experience  7239 non-null   float64
 6   Spending_Score   8068 non-null   object 
 7   Family_Size      7733 non-null   float64
 8   Var_1            7992 non-null   object 
dtypes: float64(2), int64(1), object(6)
memory usage: 378.3+ KB
```

Implementation

Data Cleaning

```
df=df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6665 entries, 0 to 8067
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Gender           6665 non-null   object 
 1   Ever_Married     6665 non-null   object 
 2   Age              6665 non-null   int64  
 3   Graduated        6665 non-null   object 
 4   Profession       6665 non-null   object 
 5   Work_Experience  6665 non-null   float64
 6   Spending_Score   6665 non-null   object 
 7   Family_Size      6665 non-null   float64
 8   Var_1            6665 non-null   object 
dtypes: float64(2), int64(1), object(6)
memory usage: 364.5+ KB
```

Implementation

Dataset after Data cleaning and preprocessing

```
df = df.reset_index()
df = df.drop("index", axis="columns")
df.head()
```

	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1
0	Male	No	22	No	Healthcare	1.0	Low	4.0	Cat_4
1	Female	Yes	67	Yes	Engineer	1.0	Low	1.0	Cat_6
2	Male	Yes	67	Yes	Lawyer	0.0	High	2.0	Cat_6
3	Male	Yes	56	No	Artist	0.0	Average	2.0	Cat_6
4	Male	No	32	Yes	Healthcare	1.0	Low	3.0	Cat_6

Implementation

Formatting the data for K-Mean clustering

```
df_new = pd.get_dummies(df)
df_new.head()
```

	Age	Work_Experience	Family_Size	Gender_Female	Gender_Male	Ever_Married_No	Ever_Married_Yes	Graduated_No	Graduated_Yes	Pro
0	22	1.0	4.0	0	1	1	0	1	0	
1	67	1.0	1.0	1	0	0	1	0	1	
2	67	0.0	2.0	0	1	0	1	0	1	
3	56	0.0	2.0	0	1	0	1	1	0	
4	32	1.0	3.0	0	1	1	0	0	1	

Implementation

Formatting the data for K-Mean clustering

```
df_kmeans = pd.get_dummies(df, drop_first=True)
df_kmeans.head()
```

	Age	Work_Experience	Family_Size	Gender_Male	Ever_Married_Yes	Graduated_Yes	Profession_Doctor	Profes
0	22	1.0	4.0	1	0	0	0	
1	67	1.0	1.0	0	1	1	0	
2	67	0.0	2.0	1	1	1	0	
3	56	0.0	2.0	1	1	0	0	
4	32	1.0	3.0	1	0	1	0	

Implementation

Create K-mean clustering model

```
kmeans_model = KMeans(n_clusters=3)
clusters = kmeans_model.fit_predict(df_kmeans)
df_kmeans.insert(df_kmeans.columns.get_loc("Age"), "Cluster", clusters)
df_kmeans.sample(5, random_state=44)
```

	Cluster	Age	Work_Experience	Family_Size	Gender_Male	Ever_Married_Yes	Graduated_Yes	Profession_Doctor	Profession_Engineer	Profession_Entertainment
6434	0	36	13.0	1.0	0	0	1	0	0	0
1190	0	18	2.0	4.0	1	0	0	0	0	0
1298	2	40	9.0	2.0	0	0	1	0	0	0
6591	0	35	9.0	2.0	1	1	1	0	0	1
6054	1	77	0.0	2.0	0	1	1	0	0	0

Applications

- Image Compression
- Image segmentation
- Evaluating performance – Academic, work or credit

Image compression using K-mean [http://tiny.cc/52gdvz]

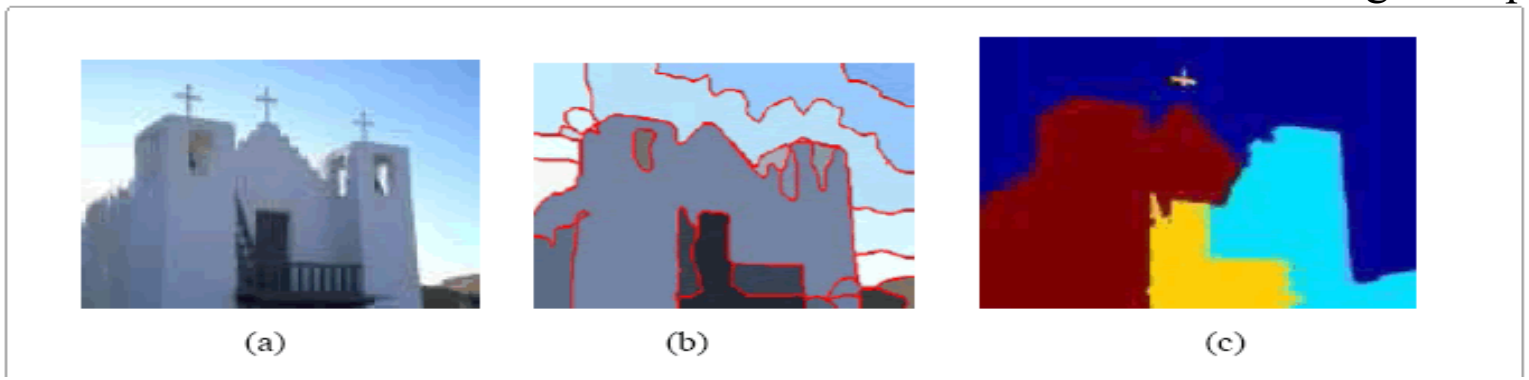


Image segmentation using K mean [http://tiny.cc/52gdvz]

References

- What is K-means Clustering [https://rb.gy/eqh727]
- K-means Algorithm [https://rb.gy/anbipp]
- Elbow Method [https://rb.gy/jdjn28]
- K-mean Implementation and code [http://bit.ly/49kADke]
- Advantages and Disadvantages [https://bit.ly/40qKkda]


End of Presentation
ANY QUESTIONS?
Meme creator.org