Omkar Gurav
BE IT
8048

classmate
Date
Page

Assignment 5

Answer the following questions

i) Explain unsupervised learning.

→

- In machine learning task of unsupervised learning is that of trying to find hidden structure in unlabelled data.

- The training data is unlabelled, so there is no error or reward signal to evaluate a partial solution.

- Unsupervised learning, the training data the instances are not bounded the corresponding output variables.

- The objective of unsupervised learning is to find hidden structure of the data.

- Unsupervised learning problems can further separated into clustering & association problems.

Clustering:- The task of dividing the data into similar group is known as clustering, such as divide the students as per scoring of marks like first class, second class pass class.

Association:- The task of finding out rules that describe large portions of the data is known as association, such as people that buy X also tends to buy Y.

ii) Explain k means algorithm.

→

- k-means is one of the simplest unsupervised learning algorithm that solves the clustering problem.

- This algorithm attempts to improve the inter group similarity while keeping the groups as far as possible from each other.

- Basically k-means runs on distance calculations, which uses "Euclidean Distance" to calculate the distance between 2 given instances.

- For given instances (X_1, Y_1) & (X_2, Y_2) the formula is
$Dist_2(X,Y) = \sqrt{\sum_{i=1}^{d} (X_i - Y_i)^2}$ where d gives dimension of features.

- "k" in k-means represents the number of clusters in which the given instances are divided.

- The basic restriction for k-Means algorithm is that your data should be continuous in nature.

- k-Means is an iterative process of clustering, which keeps iterating until it reaches the best solution or clusters for given instances.

- k-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

iii) Explain hierarchical clustering.

→

- It is a tree-based clustering algorithm.
- Hierarchical clustering does not use exemplars to perform the task of clustering.
- It partitions the given data rather than the entire instance space & hence represent a descriptive clustering rather than a predictive one.
- Mainly 2 concepts are used in hierarchical clustering.

① Dendogram:-

Given a data set D, a dendogram is a binary tree representation. In dendrogram the actual instances are represented by its leaf nodes. An internal node represents the subset of elements in the leaves of the subtree rooted at that node. The level of a node is the distance between the two clusters represented by the children of the node. Leaves have level 0.

② Linkage Function:-

To calculate the distance between the two clusters means is not the straight forward class. This has led to the introduction of the so-called linkage function, which is a general way to turnpair wise instance distances into pair wise cluster distances. The linkage method defines how the distance between 2 clusters is measured.

The most common linkage functions are as follows:

① Single linkage:-
Defines the distance between 2 clusters as the smallest pairwise distance between elements from each cluster.

② Complete linkage:-
Defines the distance between 2 clusters as the largest pointwise distance.

③ Average linkage:-
Defines the cluster distance as the average pointwise distance.

④ Centroid linkage:-
Defines the cluster distance as the point distance between the cluster means.

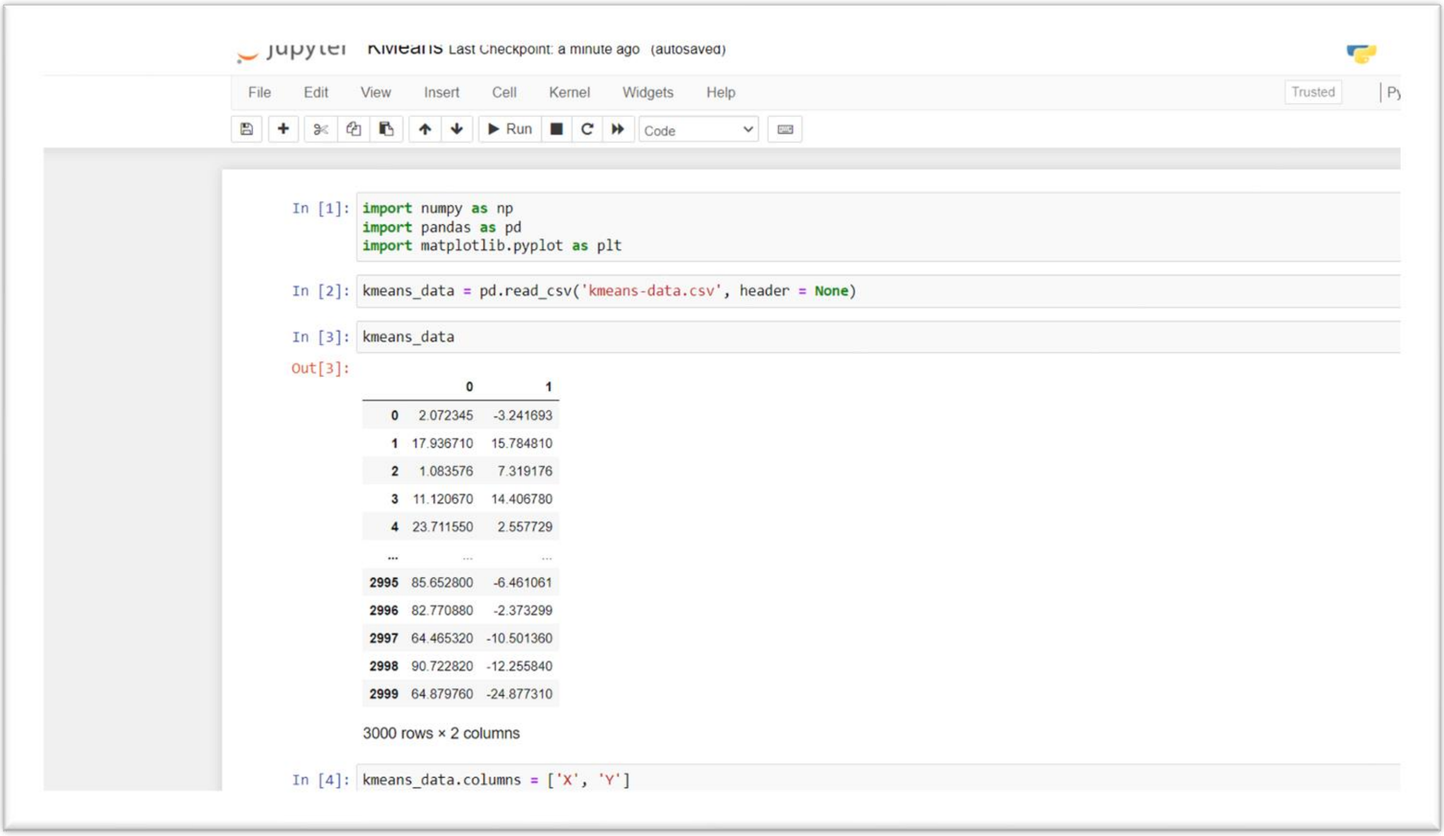
KMeans.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

kmeans_data = pd.read_csv('kmeans-data.csv', header = None)

kmeans_data

kmeans_data.columns = ['X', 'Y']

kmeans_data.isnull().sum()

array = kmeans_data.iloc[:, [0,1]].values

array

# Finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss = []

for i in range(1,11):

    kmeans = KMeans(n_clusters = i, random_state = 42)
    kmeans.fit(array)
    # Appending within-cluster-sum-of-squares
    wcss.append(kmeans.inertia_)
```

```
plt.plot(range(1, 11, 1), wcss)
plt.title('The Elbow method graph')
plt.xlabel('Number of clusters')
plt.ylabel('wcss')
plt.xticks(np.arange(1,11), np.arange(1,11))
plt.show()

kmeans = KMeans(n_clusters = 3, random_state = 42)
y_predict = kmeans.fit_predict(array)

plt.scatter(array[y_predict == 0,0], array[y_predict == 0,1], s = 100, c = 'blue', label = 'Cluster 1')

plt.scatter(array[y_predict == 1,0], array[y_predict == 1,1], s = 100, c = 'red', label = 'Cluster 2')

plt.scatter(array[y_predict == 2,0], array[y_predict == 2,1], s = 100, c = 'cyan', label = 'Cluster 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid' )

plt.title('Clusters')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()
```

plt.show()

Output :



jupyter KMeans Last Checkpoint: a minute ago (autosaved)

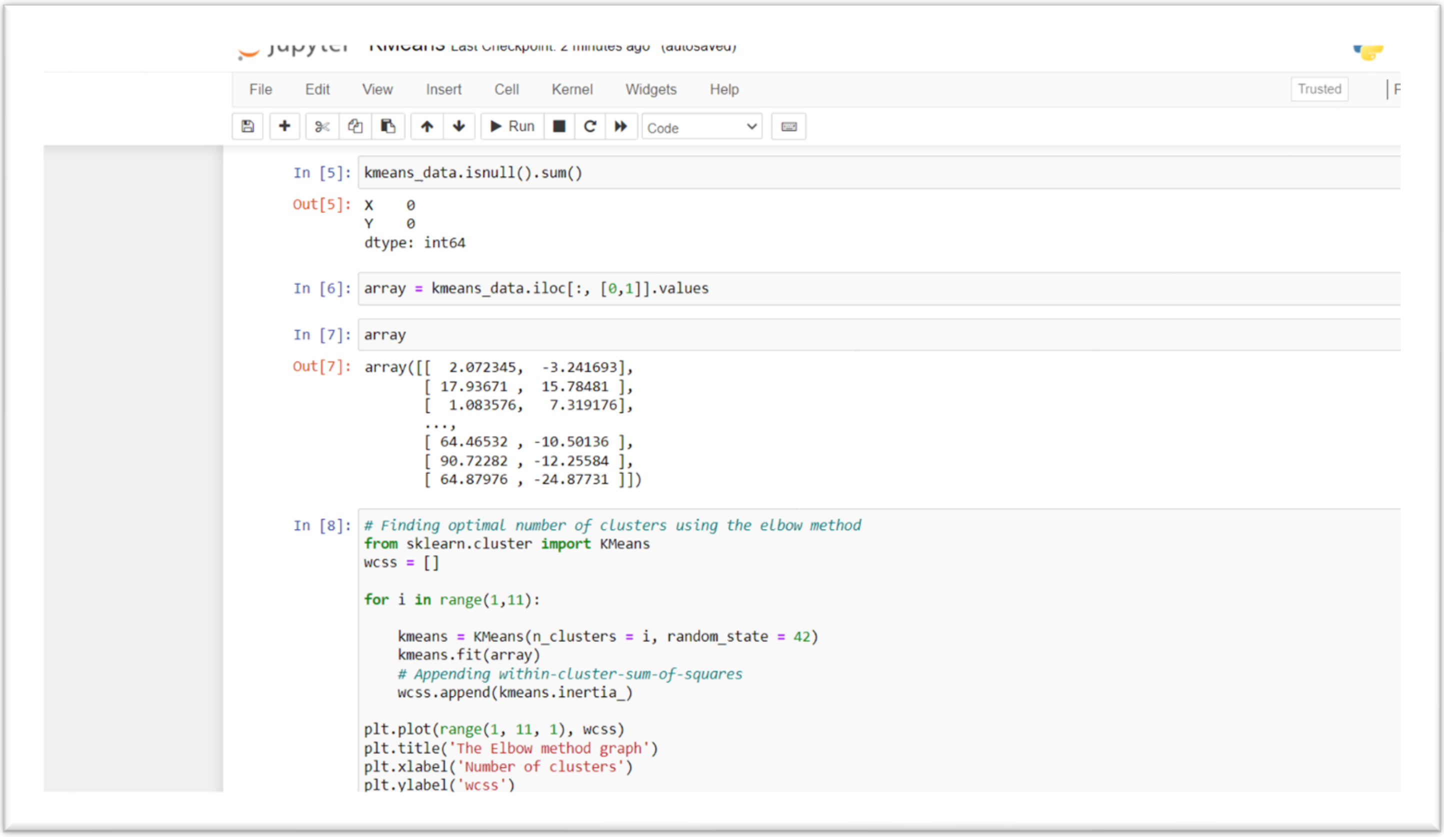
File Edit View Insert Cell Kernel Widgets Help Trusted

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

In [2]: kmeans_data = pd.read_csv('kmeans-data.csv', header = None)

In [3]: kmeans_data
Out[3]:
```

	0	1
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729
...	...	...
2995	85.652800	-6.461061
2996	82.770880	-2.373299
2997	64.465320	-10.501360
2998	90.722820	-12.255840
2999	64.879760	-24.877310

3000 rows × 2 columns

```
In [4]: kmeans_data.columns = ['X', 'Y']
```



jupyter KMeans Last Checkpoint: 2 minutes ago (autosaved)

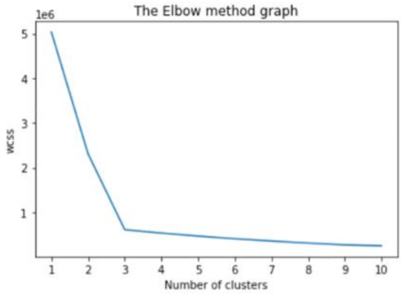
File Edit View Insert Cell Kernel Widgets Help Trusted

```
In [5]: kmeans_data.isnull().sum()
Out[5]: X    0
        Y    0
        dtype: int64

In [6]: array = kmeans_data.iloc[:, [0,1]].values

In [7]: array
Out[7]: array([[  2.072345,  -3.241693],
               [ 17.93671 ,  15.78481 ],
               [  1.083576,   7.319176],
               ...,
               [ 64.46532 , -10.50136 ],
               [ 90.72282 , -12.25584 ],
               [ 64.87976 , -24.87731 ]])

In [8]: # Finding optimal number of clusters using the elbow method
        from sklearn.cluster import KMeans
        wcss = []

        for i in range(1,11):

            kmeans = KMeans(n_clusters = i, random_state = 42)
            kmeans.fit(array)
            # Appending within-cluster-sum-of-squares
            wcss.append(kmeans.inertia_)

        plt.plot(range(1, 11, 1), wcss)
        plt.title('The Elbow method graph')
        plt.xlabel('Number of clusters')
        plt.ylabel('wcss')
```

jupyter KMeans Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

```python
plt.plot(range(1, 11, 1), wcss)
plt.title('The Elbow method graph')
plt.xlabel('Number of clusters')
plt.ylabel('wcss')
plt.xticks(np.arange(1,11), np.arange(1,11))
plt.show()
```



In [9]:
```python
kmeans = KMeans(n_clusters = 3, random_state = 42)
y_predict = kmeans.fit_predict(array)
```

In [10]:
```python
plt.scatter(array[y_predict == 0,0], array[y_predict == 0,1], s = 100, c = 'blue', label = 'Cluster 1')

plt.scatter(array[y_predict == 1,0], array[y_predict == 1,1], s = 100, c = 'red', label = 'Cluster 2')

plt.scatter(array[y_predict == 2,0], array[y_predict == 2,1], s = 100, c = 'cyan', label = 'Cluster 3')
```

jupyter KMeans Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

In [10]:
```python
plt.scatter(array[y_predict == 0,0], array[y_predict == 0,1], s = 100, c = 'blue', label = 'Cluster 1')

plt.scatter(array[y_predict == 1,0], array[y_predict == 1,1], s = 100, c = 'red', label = 'Cluster 2')

plt.scatter(array[y_predict == 2,0], array[y_predict == 2,1], s = 100, c = 'cyan', label = 'Cluster 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid' )

plt.title('Clusters')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()
```