Experiment 7

Aim: To implement different clustering algorithms.

Problem Statement: a) Clustering algorithm for unsupervised classification (K-means, density based (DBSCAN), Hierarchical clustering)

About Dataset: This is the dataset used in the second chapter of Aurélien Géron's recent book 'Hands-On Machine learning with Scikit-Learn and TensorFlow'. It serves as an excellent introduction to implementing machine learning algorithms because it requires rudimentary data cleaning, has an easily understandable list of variables and sits at an optimal size between being too toyish and too cumbersome. The data contains information from the 1990 California census. So although it may not help you with predicting current housing prices like the Zillow Zestimate dataset, it does provide an accessible introduction dataset for teaching people about the basics of machine learning.

Theory:

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns. It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset. After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML systems can use this id to simplify the processing of large and complex datasets.

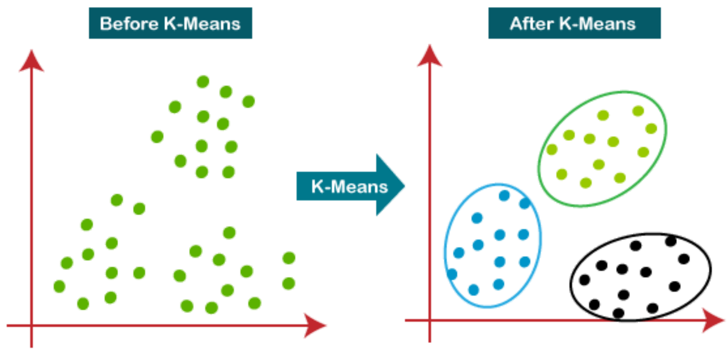
1. K Means

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs to only one group that has similar properties. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimise the sum of distances between the data point and their corresponding clusters. The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has data points with some commonalities, and it is away from other clusters. The below diagram explains the working of the K-means Clustering Algorithm:

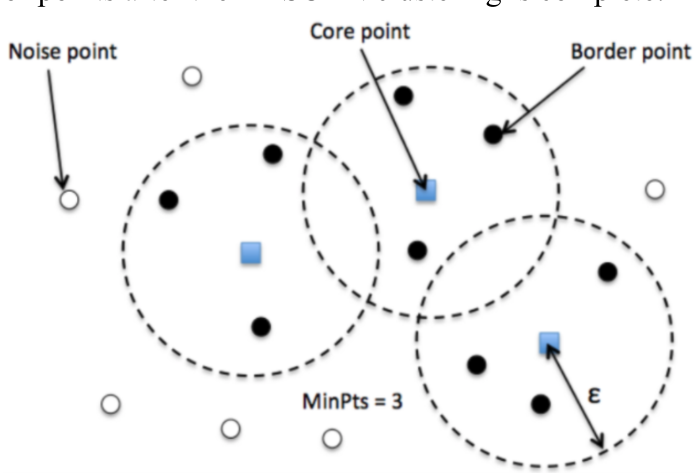


2. DBSCAN

DBSCAN is a clustering algorithm that defines clusters as continuous regions of high density and works well if all the clusters are dense enough and well separated by low-density regions. In the case of DBSCAN, instead of guessing the number of clusters, it will define two hyperparameters: epsilon and minPoints to arrive at clusters.

- Epsilon (ε): A distance measure that will be used to locate the points/to check the density in the neighbourhood of any point.
- minPoints(n): The minimum number of points (a threshold) clustered together for a region to be considered dense.

There are three types of points after the DBSCAN clustering is complete:



- Core — This is a point that has at least m points within distance n from itself.
- Border — This is a point that has at least one Core point at a distance n.
- Noise — This is a point that is neither a Core nor a Border. And it has less than m points within distance n from itself.

Algorithmic steps for DBSCAN clustering

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least 'minPoint' points within a radius of 'ε' to the point then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighbourhood calculation for each neighbouring point

3. Hierarchical clustering

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA. In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram. Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm. This hierarchy of clusters is represented in the form of the dendrogram.

The hierarchical clustering technique has two approaches:

- Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach.

Agglomerative Hierarchical clustering: The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

Divisive Hierarchical clustering: Also known as a top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

Output:

K Means:

```
[ ] df = pd.read_csv("/kaggle/input/california-housing-prices/housing.csv")
```

```
[ ] df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

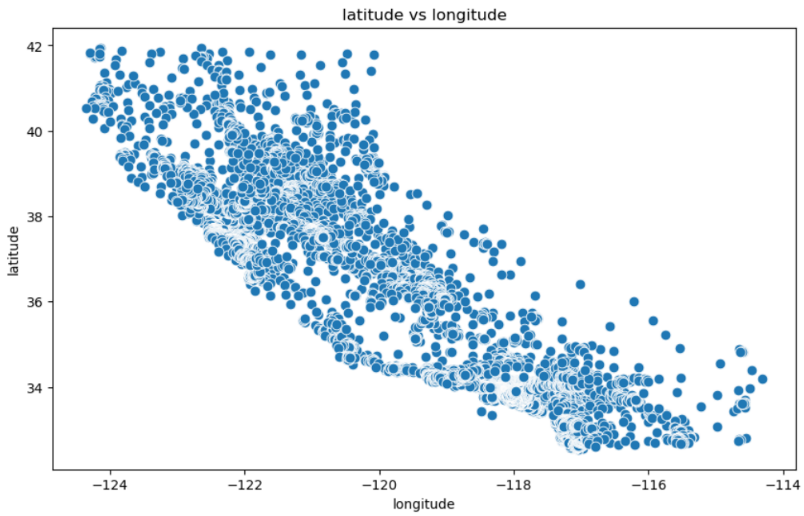
```
[ ] #Importing the necessary libraries
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from mpl_toolkits.mplot3d import Axes3D
    %matplotlib inline
```

```
[ ] #Importing KMeans from sklearn
    from sklearn.cluster import KMeans
```

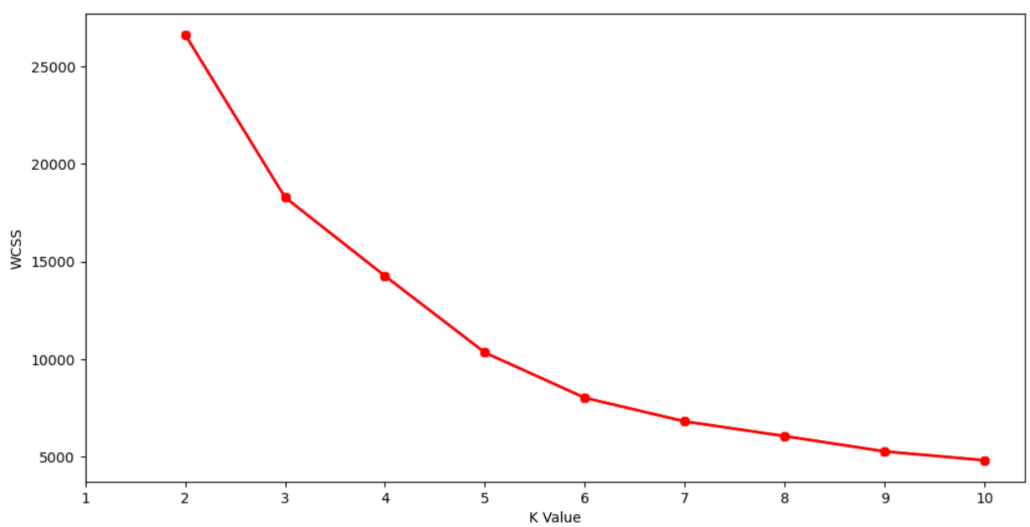
```
df.corr()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069608	0.099773	0.055310	-0.015176	-0.045967
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066983	-0.108785	-0.071035	-0.079809	-0.144160
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.320451	-0.296244	-0.302916	-0.119034	0.105623
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.930380	0.857126	0.918484	0.198050	0.134153
total_bedrooms	0.069608	-0.066983	-0.320451	0.930380	1.000000	0.877747	0.979728	-0.007723	0.049686
population	0.099773	-0.108785	-0.296244	0.857126	0.877747	1.000000	0.907222	0.004834	-0.024650
households	0.055310	-0.071035	-0.302916	0.918484	0.979728	0.907222	1.000000	0.013033	0.065843
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007723	0.004834	0.013033	1.000000	0.688075
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049686	-0.024650	0.065843	0.688075	1.000000

```
X=df[["longitude","latitude"]]
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'longitude',y = 'latitude',  data = X  ,s = 60 )
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('latitude vs longitude')
plt.show()
```



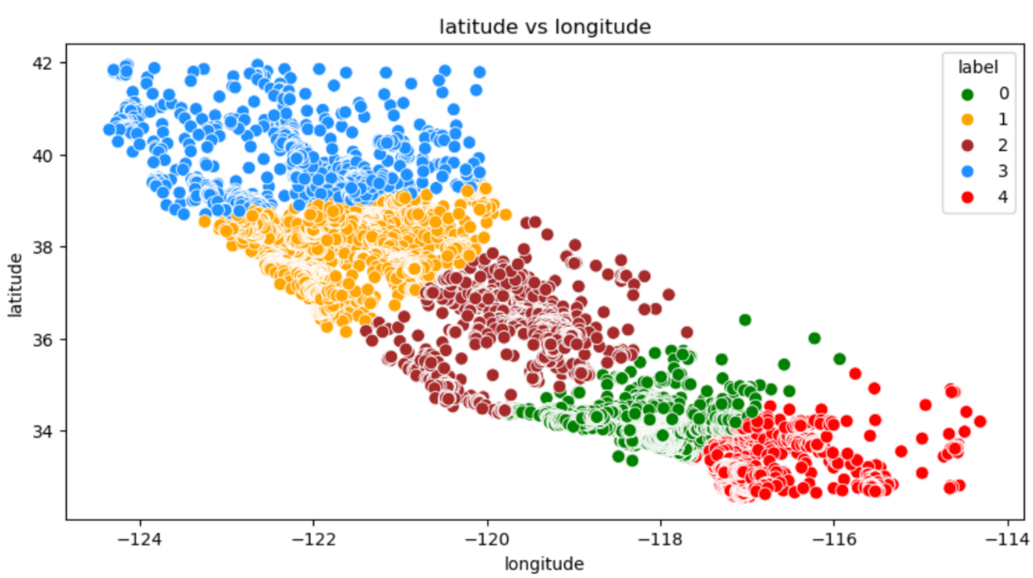

```python
from sklearn.cluster import KMeans
wcss=[]
for i in range(2,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    wcss.append(km.inertia_)
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(2,11),wcss)
plt.plot(range(2,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```



```python
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df["label"] = y
#The new dataframe with the clustering done
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	label
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY	1
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY	1
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY	1
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY	1

```python
#Scatterplot of the clusters
plt.figure(figsize=(10,5))
sns.scatterplot(x = 'longitude',y = 'latitude',hue="label",
                 palette=['green','orange','brown','dodgerblue','red'], legend='full',data = df  ,s = 60 )
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('latitude vs longitude')
plt.show()
```



DBSCAN

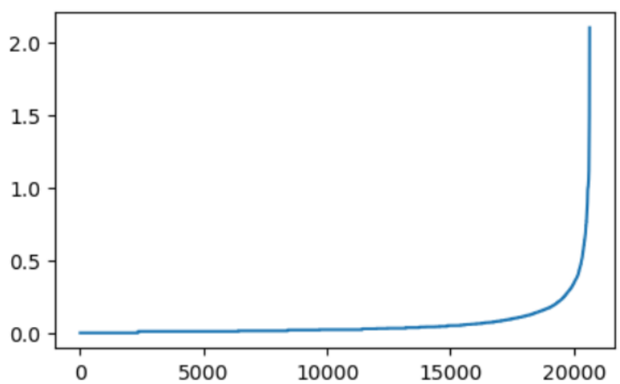
DBSCAN

Compute data proximity from each other using Nearest Neighbours

```
[ ] from sklearn.neighbors import NearestNeighbors # importing the library
    neighb = NearestNeighbors(n_neighbors=2) # creating an object of the NearestNeighbors class
    nbrs=neighb.fit(X) # fitting the data to the object
    distances,indices=nbrs.kneighbors(X) # finding the nearest neighbours
```

Sorting and plot the distances between the data points

```
# Sort and plot the distances results
distances = np.sort(distances, axis = 0) # sorting the distances
distances = distances[:, 1] # taking the second column of the sorted distances
plt.rcParams['figure.figsize'] = (5,3) # setting the figure size
plt.plot(distances) # plotting the distances
plt.show() # showing the plot
```

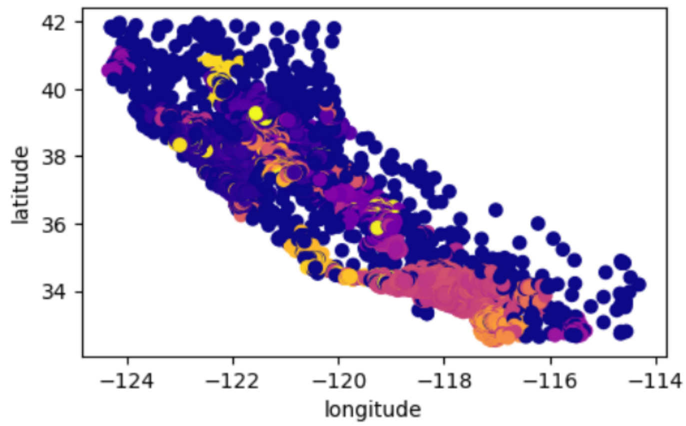



```
# extracting the above mentioned columns
X1 = data.loc[:, ['longitude',
                  'latitude']].values
```

Implementing the DBSCAN model

```
[ ] from sklearn.cluster import DBSCAN
    # cluster the data into five clusters
    dbscan = DBSCAN(eps = 0.3, min_samples = 4).fit(X) # fitting the model
    labels = dbscan.labels_ # getting the labels
```

```
plt.scatter(X1[:, 0], X1[:,1], c = labels, cmap= "plasma") # plotting the clusters
plt.xlabel("longitude") # X-axis label
plt.ylabel("latitude") # Y-axis label
plt.show() # showing the plot
```

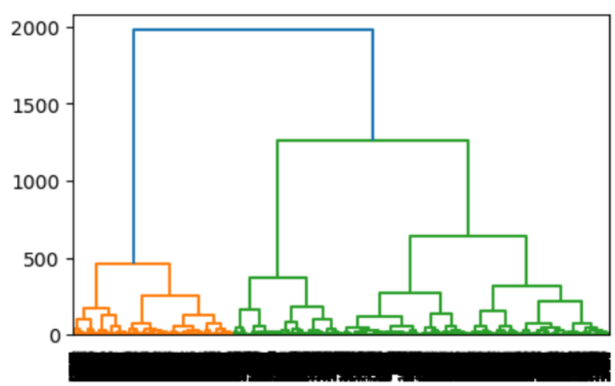


Hierarchical Clustering

Hierarchical Clustering

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
```

```
[ ] dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
```



Conclusion: Thus we studied and understood various clustering algorithms such as k-means, hierarchical, DBSCAN techniques and also visualised them using scatterplot and dendograms for better understanding of data