**CHAPTER 3: Unsupervised Learning (Clustering)**

**Theory**

In the previous chapter, we studied bias-variance trade-off, what is regression, what is simple linear regression, what is multiple linear regression, evaluating regression models, and finally Logistic Regression for classification. In this chapter, we will see what Unsupervised Learning is, K-Means Clustering, Hierarchical Clustering, and Principal Component Analysis.

Let’s start our chapter with Unsupervised Learning.

**Introduction to Unsupervised Learning**

In Unsupervised learning, the information used to train is neither classified nor labeled in the dataset. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The main task of unsupervised learning is to find patterns in the data.

Once a model learns to develop patterns, it can easily predict patterns for any new dataset in the form of clusters. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Often it is easier to get unlabelled data compared to labeled data, and in such cases, we can use unsupervised machine learning to work on the data. Data, which needs categorization can be categorized with the help of unsupervised machine learning.

Clustering is a type of unsupervised machine learning in which the algorithm processes our data and divide them into “cluster”.

**Clustering**

Let’s suppose we give a child a different object to a group. How does a child make a group? The child may group over the color, over the shape, over the hardness or softness of the object, etc. The basic idea here is that the child tries to find out similarities and dissimilarities between different objects and then tries to make a group of similar objects. This is called **Clustering**, the method of identifying similar instances and keeping them together. In other words, clustering identifies homogeneous subgroups among the observations.



Example:

Chart, scatter chart

Description automatically generated

In the figure above, we can easily identify 4 different clusters. The clustering criteria here is distance. Whichever points are near to each other are kept in the same cluster and the faraway points belong to a different cluster.

**The Goal of Clustering**

The goal of clustering is to determine the intrinsic group in unlabelled data. The question is: What constitutes a good cluster? It can be shown that there is no absolute “best” criterion for cluster validation. Consequently, it is the user who must supply the criterion for validating the cluster. For example, We might be interested in finding representatives of homogeneous instances for finding the “natural clusters” and identifying their unknown properties (like “natural” data types), for finding appropriate groupings, or in finding unusual (which are from all other data) data objects (outlier detection).

**Applications**

The scikit-learn book describes the various applications of clustering as follows:

* **For customer segmentation:** You can cluster your customers based on their purchases, their activity on your website, and so on. This is useful to understand who your customers are and what they need, so you can adapt your products and marketing campaigns to each segment. For example, this can be useful in recommender systems to suggest content that other users in the same cluster enjoyed.
* **For data analysis:** When analyzing a new dataset, it is often useful to first discover clusters of similar instances, as it is often easier to analyze clusters separately.
* **As a dimensionality reduction technique:** Once a dataset has been clustered, it is usually possible to measure each instance’s affinity with each cluster (affinity is any measure of how well an instance fits into a cluster). Each instance’s feature vector x can then be replaced with the vector of its cluster affinities. If there are k clusters, then this vector is k dimensional. This is typically much lower-dimensional than the original feature vector, but it can preserve enough information for further processing.
* **For anomaly detection (also called outlier detection):** Any instance that has a low affinity to all the clusters is likely to be an anomaly. For example, if you have clustered the users of your website based on their behavior, you can detect users with unusual behavior, such as an unusual number of requests per second, and so on. Anomaly detection is particularly useful in detecting defects in manufacturing, or for fraud detection.
* **For semi-supervised learning:** If you only have few labels, you could perform clustering and propagate the labels to all the instances in the same cluster. This can greatly increase the number of labels available for a subsequent supervised learning algorithm and thus improve its performance.
* **For search engines:** For example, some search engines let you search for images that are like a reference images. To build such a system, you would first apply a clustering algorithm to all the images in your database: similar images would end up in the same cluster. Then when a user provides a reference image, all you need to do is to find this image’s cluster using the trained clustering model, and you can then simply return all the images from the cluster.
* **To segment an image:** By clustering pixels according to their color, then replacing each pixel’s color with the mean of its cluster, it is possible to reduce the number of different colors in the image considerably. This technique is used in many detection and tracking system, as it makes it easier to detect the contour of each object.

**Main Requirements:**

The primary requirements that should be met by a clustering algorithm are:

* It should be scalable
* It should be able to deal with attributes of different types
* It should be able to discover arbitrary shape clusters
* It should have an inbuilt ability to deal with noise and outliers
* The clusters should not vary with the order of input records
* It should be able to handle data of high dimensions
* It should be easy to interpret and use

**Approaches for Clustering**

The clustering approaches can be broadly divided into two categories: Agglomerative and Divisive.

**Agglomerative:** This approach first considers all the points as individual clusters and then finds out the similarity between two points, puts them into a cluster. Then it goes on to find similar points and clusters until there is only one cluster left i.e., all points belong to a big cluster. This is also called the bottom-up approach.

**Divisive:** It is the opposite of the agglomerative approach. It first considers all the points to be part of one big cluster and in the subsequent steps tries to find out the points/clusters which are least like each other and then breaks the bigger cluster into smaller ones. This continues until there are as many clusters as there are data points. This is also called the top-down approach.

**K-Means Clustering**

K-Means is a clustering approach in which the data is grouped into K distinct non-overlapping clusters based on their distances from the K centers. The value of K needs to be specified first and then the algorithm assigns the points to exactly one cluster.

The algorithm takes raw unlabelled data as an input and divides the dataset into clusters and the process is repeated until the best clusters are found.

K-Means is very easy and simple to implement. It is highly scalable, can be applied to both small and large datasets. There is, however, a problem with choosing the number of clusters or K. Also, with the increase in dimensions, stability decreases. But, overall K Means is a simple and robust algorithm that makes clustering very easy.

The theory discussed above can be mathematically expressed as:

* Let C1, C2, CK be the K Clusters
* Then we can write:



i.e., each data point has been assigned to a cluster.

* Also,



This means that the clusters are non-overlapping.

* The idea behind the K-Means clustering approach is that the within-cluster variation amongst the point should be minimum. The within-cluster variance is denoted by: W(Ck). Hence, according to the statement above, we need to minimize this variance for all the clusters.

Mathematically it can be written as:

Text

Description automatically generated with low confidence

* The next step is to define the criterion for measuring the within-cluster variance. Generally, the criterion is the Euclidean distance between two data points.

A picture containing text, watch, clock

Description automatically generated

* The above formula says that we are calculating the distances between all the points in a cluster, then we are repeating it for all the K clusters (That’s why two summation signs), and then we are dividing it by the number of observations in the clusters (Ck is the number of observations in the Kth cluster) to calculate the average.

So, ultimately our goal is to minimize:

A picture containing text, clock

Description automatically generated

The following algorithm steps are used to solve this problem

Algorithm:

1. Randomly assign K centres.
2. Calculate the distance of all the points from all the K Centres and allocate the points to the cluster based on the shortest distance. The model’s inertia is the mean squared distance between each instance and its closest centroid. The goal is to have a model with the lowest inertia.
3. Once all the points are assigned to clusters, recompute the centroids.
4. Repeat steps 2 and 3 until the locations of the centroids stop changing and the cluster allocation of the points becomes constant.

**Exercise**

Importing all the required libraries

Text

Description automatically generated

Consider a collection of points that are sampled from three different densities, in this case, normal densities with the same covariances but different means.

Text

Description automatically generated

A picture containing chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Here is a scatterplot of this data. We clearly see three mixture components.

How can we recover these clusters?

This is the job of clustering algorithms.

Mixture densities arise in both unsupervised learning and in supervised learning. In both cases, they commonly represent a problem structure in which data is generated from several ideal prototypes (The cluster centres) but then corrupted by noise.

* When each cluster has a distinct class label, we have a regular classification problem with normal densities.
* When no cluster has any labels, we can hope to recover the underlying clusters with a clustering algorithm and then assign labels to these clusters. This is a form of semi-supervised learning.
* When there is training data with labels available, often each class is a mixture of multiple clusters. That is, each class is generated by multiple prototypes (think characters in different fonts).

You can perform clustering either at the class level or across all samples and then label each cluster with its corresponding class label.

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Obviously, those centres are wrong, but let’s keep going.

Now, we compute the assignment of the data points to the prototypes (array Closet). This is also wrong, but we’re going to be using it anyway.

Graphical user interface, text

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

Graphical user interface, text

Description automatically generated

Now we pretend that the cluster assignments are correct and recompute the location of the centres.

Text

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

As we can see, the centres have moved, and it looks like they have generally moved in the right direction.

Now, let’s just repeat this process multiple times

Graphical user interface, text, application

Description automatically generated

We can now plot the path that the prototype guesses have taken.

Text

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

As you can see, the final location of the prototype centres (red) is nicely in the centre of the classes. The algorithm doesn’t give us exactly the cluster centres because there are three clusters, but we postulated four cluster centres.

We can also look at the partition of the data included by these cluster centres.

Graphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

The diagram above shows the step-by-step implementation of the K-Means algorithm.

As we saw earlier, we need to provide the value of K before ahead. But the question is how to get a good value of K. An optimum value of k is obtained using the Elbow Method.

**The Elbow-Method**

This method is based on the relationship between the within-cluster sum of squared distances (WCSS or Inertia) and the number of clusters. It is observed that first with an increase in the number of clusters WCSS decreases steeply and then after a certain number of clusters the drop in WCSS is not that prominent. The point after which the graph between WCSS and the number of clusters becomes comparatively smother is termed as the elbow and the number of clusters at that point are the optimum number of clusters as even after increasing the cluster after that point the variation is not decreasing by much i.e., we have accounted for almost all the dissimilarity in the data. An elbow-curve looks like this:

Chart, line chart

Description automatically generated

**Case Study**

**Mall Customer Data**

Mall customer data is an interesting dataset that has hypothetical customer data. It puts you in the shoes of the owner of a supermarket. You have customer data, and on this basis of the data, we must divide the customers into various groups.

**Importing the necessary libraries**

Graphical user interface, text

Description automatically generated

We have imported the necessary libraries. Let’s go ahead and import the data.

A picture containing company name

Description automatically generated

Table

Description automatically generated

Let’s do some basic data exploration.

Graphical user interface, text, application, chat or text message

Description automatically generated

We have a total of 200 observations and 5 features.

Text

Description automatically generated

All the features are imported with correct data types and there are no null values in the dataset.

Let’s check the correlation between the numeric values.

Table

Description automatically generated

The data seems to be interesting as there is no correlation between the variables.

**Visualizing the Distributions**

**Annual Income Distribution**

Text

Description automatically generated

Chart, line chart

Description automatically generated

Most of the customer’s income falling between $50K to $80K. a small number of customers having an income greater than $80K.

**Age Distribution**

Text

Description automatically generated

Chart, line chart, histogram

Description automatically generated

There are customers of a wide variety of ages.

**Spending Score Distribution**

Text

Description automatically generated

Chart, line chart

Description automatically generated

The maximum spending score is in the range of 40 to 60.

**Gender Analysis:**

Text

Description automatically generated

Chart, treemap chart

Description automatically generated

Most of the customers are female customers than males.

**Clustering based on 2 features**

First, we work with two features only, annual income and spending score.

Text

Description automatically generated

We subset the dataset with only two features (Annual Income, Spending Score) and saved it to the variable “X”.

Table

Description automatically generated

Let’s check the correction between two variables using a scatter plot.

A picture containing diagram

Description automatically generated

Chart, scatter chart

Description automatically generated

Text

Description automatically generated

Now we calculate Within Cluster Sum of Squared Errors (WCSSE) for different values of k. Next, we choose the k for which WCSSE first starts to diminish. This value of K gives us the best number of clusters to make from the raw data.

Graphical user interface, text

Description automatically generated

Chart, line chart

Description automatically generated

This is known as the elbow graph, the x-axis being the number of clusters, the number of clusters is taken at the elbow joint point. This point is the point where making clusters is most relevant as here the value of WCSSE suddenly stops decreasing. Here in the graph, after 5 the drop is minimal, so we take 5 to be the number of clusters.

Text

Description automatically generated

The labels added to the data.

Table

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

We can clearly see that 5 different clusters have been formed from the data. The red cluster is the customers with the least income and least spending score, similarly, the blue cluster is the customers with the most income and most spending score.

Now let’s print the customer ID according to the groups

A picture containing calendar

Description automatically generated

Table

Description automatically generated with medium confidence

**K-Means Clustering based on 3D data**

Now, we shall be working on 3 types of data. Apart from the spending score and annual income of customers, we shall also take in the age of the customers.

A picture containing text

Description automatically generated

Table

Description automatically generated

Filtering the features



Now we calculate the WCSSE for different values of k.

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Chart, line chart

Description automatically generated

Assigning cluster labels to the clusters.

Text

Description automatically generated

Table

Description automatically generated

Plotting 3D plot as we did the clustering based on 3 input features.

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

A picture containing calendar

Description automatically generated

Text

Description automatically generated

**Hierarchical Clustering**

One main disadvantage of K-Means is that it needs us to pre-enter the number of clusters (K). Hierarchical clustering is an alternative approach that does not need us to give the value of K beforehand and, it creates a beautiful tree-based structure for visualization.

Here, we are going to discuss the bottom-up (or Agglomerative) approach of cluster building. We start by defining any sort of similarity between the data points. Generally, we consider the Euclidean distance. The points which are closer to each are more similar than the points which are farther away. The Algorithm starts with considering all points as separate clusters and then grouping points together to form clusters.

**The Algorithm:**

1. Begin with n observations and a measure (such as Euclidean distance) of all the n(n-1)/2 pairwise dissimilarities (or the Euclidean distances generally). Treat each observation as its own cluster. Initially, we have n clusters.
2. Compare all the distances and put the two closest points/clusters in the same cluster. The dissimilarity (or The Euclidean distance) between these two clusters indicates the height in the dendrogram at which the fusion line should be placed.
3. Compute the new pairwise inter-cluster dissimilarities (or The Euclidean distance) among the remaining clusters.
4. Repeat steps 2 and 3 till we have only one cluster left.

**Setting up the Example**

Suppose a teacher wants to divide her students into different groups. She has the marks scored by each student in an assignment and based on these marks, she wants to segment them into groups. There’s no fixed target here as to how many groups to have. Since the teacher does not know what type of students should be assigned to which group, it cannot be solved as a supervised learning problem. So, we will try to apply hierarchical clustering here and segment the students into different groups.

Let’s take a sample of 5 students:

Table

Description automatically generated

**Creating a Proximity Matrix**

First, we will create a proximity matrix that tells us the distance between each of these points. Since we are calculating the distance of each point from each of the other points, we will get a square matrix of shape nxn (where n is the number of observations).

Let’s make the 5x5 proximity matrix for our example:

A picture containing calendar

Description automatically generated

The diagonal elements of this matrix will always be 0 as the distance of a point with itself is always 0. We will use the Euclidean distance formula to calculate the rest of the distances. So, let’s say we want to calculate the distance between points 1 and 2:



Similarly, we can calculate all the distances and fill the proximity matrix.

**Steps to Perform Hierarchical Clustering**

**Step 1:** First, we will assign all the points to an individual cluster:



Different colors represent different clusters. You can see that we have 5 different colors for the 5 points in our data.

**Step 2:** Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance. We then update the proximity matrix:

Calendar

Description automatically generated

Here, the smallest distance is 3 and hence we will merge points 1 and 2:

Chart, bubble chart

Description automatically generated

Let’s look at the updated clusters and accordingly update the proximity matrix:

Table

Description automatically generated

Here, we have taken the maximum of the two marks (7,10) to replace the marks for this cluster. Instead of the maximum, we can also take the minimum value or the average values as well. Now we will again calculate the proximity matrix for these clusters:

A picture containing telephone, white

Description automatically generated

**Step 3:** We will repeat step 2 until only a single cluster is left.

So, we will first look at the maximum distance in the proximity matrix and then merge the closest pair of clusters. We will get the merged clusters as shown below after repeating these steps:

Chart, bubble chart

Description automatically generated

We started with 5 clusters and finally have a single cluster. This is how agglomerative hierarchical clustering works. But the burning question still remains – how do we decide the number of clusters? Let’s understand that in the next section.

**How should we Choose the Number of Clusters in Hierarchical Clustering?**

Ready to finally answer this question that’s been hanging around since we started learning? To get the number of clusters for hierarchical clustering, we make use of an awesome concept called a Dendrogram.

A dendrogram is a tree-like diagram that records the sequences of merges or splits.

Let’s get back to our teacher-student example. Whenever we merge two clusters, a dendrogram will record the distance between these clusters and represent it in graph form. Let’s see how a dendrogram looks like:

Chart

Description automatically generated

We have the samples of the dataset on the x-axis and the distance on the y-axis. **Whenever two clusters are merged, we will join them in this dendrogram, and the height of the join will be the distance between these points.** Let’s build the dendrogram for our example:

Chart, bubble chart

Description automatically generated

Take a moment to process the above image. We started by merging samples 1 and 2 and the distance between these two samples was 3. Let’s plot in the dendrogram.

Chart, histogram, box and whisker chart

Description automatically generated

Here, we can see that we have merged sample 1 and 2. The vertical line represents the distance between these samples. Similarly, we plot all the steps where we merged the clusters and finally, we get a dendrogram like this:

Chart, box and whisker chart

Description automatically generated

We can clearly visualize the steps of hierarchical clustering. **More the distance of the vertical lines in the dendrogram, the more the distance between those clusters.**

Now, we can set a threshold distance and draw a horizontal line (Generally, we try to set the threshold in such a way that it cuts the tallest vertical line). Let’s set this threshold as 12 and draw a horizontal line:

Chart, box and whisker chart

Description automatically generated

**The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold.** In the above example, since the red line intersects 2 vertical lines, we will have 2 clusters. One cluster will have a sample (1,2,4) and the other will have a sample (3,5). Pretty straightforward, right?

This is how we can decide the number of clusters using a dendrogram in Hierarchical Clustering.

**Example:**

Importing the necessary libraries

Graphical user interface, text

Description automatically generated

Now let’s define a method to plot the clusters.

Text

Description automatically generated

A simple clustering problem

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

The idea behind linkage clustering or hierarchical clustering is to put things that are close together into the same cluster. Linkage clustering is usually based on distances only.

Compute the distance between each pair of the two collections of inputs i.e., calculate distance between all the points in data.

Text

Description automatically generated

Based on pairwise distances, we can now compute a linkage matrix. We successively merge the closest points/clusters into the same cluster. The linkage “matrix” is simply a table listing which pairs of points are merged at what step and what distance.

Graphical user interface, text

Description automatically generated

A picture containing letter

Description automatically generated



Chart, histogram

Description automatically generated

We can “cut” the dendrogram to form a flat cluster. If we cut the above diagram into two clusters, we get a good answer.

Plotting the individual clusters

Text

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

As you can see that now the clusters are shown in two different colors. It means that the algorithm has created two separate groupings based on some similarity criteria.

In the lines above, have talked about the linkage matrices and we have written in our code “lm = linkage (ds, “single”)”. So let’s discuss the different types of linkages that we generally use.

**Linkage Methods**

Taking data for example

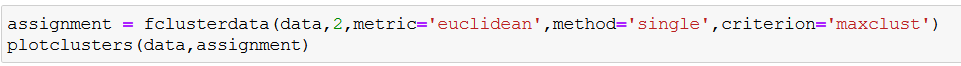
Text

Description automatically generated

Initially, hierarchical clustering starts out with clusters consisting of individual points. Later, it compares clusters with each other and merges the two “closest” clusters.

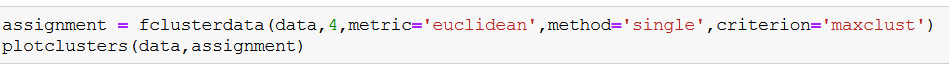
Since clusters are a set of points, there are many kinds of linkage methods:

* **Single Linkage:** Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one at a time.
  + Cluster distance is the smallest distance between any point in cluster1 and any point in cluster 2.
  + Highly sensitive to outliers when forming flat clusters.
  + Works well for low-noise data with an unusual structure.



Chart, scatter chart

Description automatically generated



Chart, scatter chart

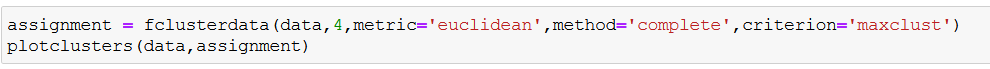
Description automatically generated

* **Complete Linkage:** Maximal Intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B and record the largest of these dissimilarities.
  + Cluster distance is the largest distance between any point in cluster1 and any point in cluster 2
  + Less sensitive to outliers than single linkage.



Chart, scatter chart

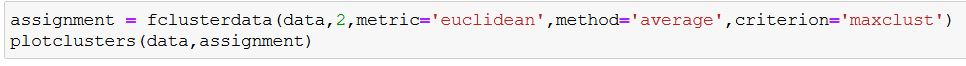
Description automatically generated



Chart, scatter chart

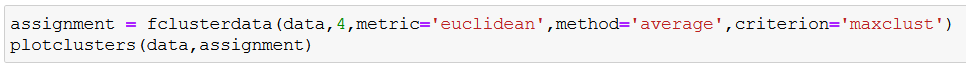
Description automatically generated

* **Average Linkage:** Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observation in cluster A and the observation in cluster B, and record the average of these dissimilarities.
  + Cluster distance is the average distance of all pairs of points in clusters 1 and 2



Chart, scatter chart

Description automatically generated



Chart, scatter chart

Description automatically generated

* **Centroid Linkage:** The dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linage can result in undesirable inversions.
  + Cluster distance is the distance of the centroids of both clusters

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

* **Ward Linge:** Wikipedia says Ward’s minimum variance criterion minimizes the total within-cluster variance. To implement this method, at each step find the pair of clusters that leads to a minimum increase in total within-cluster variance after merging.
  + Based on minimizing a variance criterion before and after merging.

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

**Case Study**

We will be working on a wholesale customer segmentation problem. The data is hosted on the UCI Machine Learning repository. The aim of this problem is to segment the Client of a wholesale distributor based on their annual spending on diverse product categories like milk, grocery, region, etc.

**Importing all the necessary libraries**

Graphical user interface, text, application

Description automatically generated

**Load the data and look at the first few rows.**

A picture containing text

Description automatically generated

Table

Description automatically generated

There are multiple product categories – Fresh, Milk, Grocery, etc. The values represent the number of units purchased by each client for each product. **Our aim is to make clusters from this data that can segment similar clients together.** We will, of course, use Hierarchical Clustering for this problem.

But before applying Hierarchical Clustering, we must normalize the data so that the scale of each variable is the same. Why is this important? Well, if the scale of the variables is not the same, the model might be biased towards the variables with a higher magnitude like Fresh or Milk (refer to the above table).

So, let’s first normalize the data and bring all the variables to the same scale:

Text

Description automatically generated

Text

Description automatically generated with low confidence

Here, we can see that the scale of all the variables is almost similar. Now, we are good to go. Let’s first draw the dendrogram to help us decide the number of clusters for this problem:

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

The x-axis contains the samples and the y-axis represents the distance between these samples. The vertical line with maximum distance is the blue line and hence we can decide on a threshold of 6 and cut the dendrogram.

Graphical user interface, application, Word

Description automatically generated

A picture containing chart

Description automatically generated

We have two clusters as this line cuts the dendrogram at two points. Let’s now apply hierarchical clustering for 2 clusters:

Text, Word

Description automatically generated

A picture containing calendar

Description automatically generated

We can see the values of 0s and 1s in the output since we defined 2 clusters. 0 represents the points that belong to the first cluster and 1 represents points in the sending cluster.

Let’s now visualize the two clusters:

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Awesome! We can clearly visualize the two clusters here. This is how we can implement hierarchical clustering in Python.

Now let’s get into a new topic called PCA.

**The Curse of Dimensionality**

Humans are bound by their perception of a maximum of three dimensions. We can’t comprehend shapes/graphs beyond three dimensions. Often, data scientists get datasets that have thousands of features. They give birth to two kinds of problems:

* **Increase in computation time:** The majority of the machine learning algorithms rely on the calculation of distance for model building and as the number of dimensions increase, it becomes more and more computation-intensive to create a model out of it. For example, if we must calculate the distance between two points in just one dimension, like two points on the number line, we will just subtract the coordinate of one point from another and then take the magnitude:



What if we need to calculate the distance between two points in two dimensions? The same formula translates to:



What if we need to calculate the distance between two points in three dimensions? The same formula translates to:



And for N-dimensions, the formula becomes:



This is the effort of calculating the distance between two points. Just imagine the number of calculations involved for all the data points involved.

One more point to be considered is that as the number of dimensions increases, points are going far away from each other. This means that any new point that comes when we are testing the model is going to be farther away from our training points. This leads to a less reliable model, and it makes our model overfitted to the training data.

* **Hard (or almost impossible) to visualize the relationship between features:** As stated above, humans can not comprehend things beyond three dimensions. So, if we have an n-dimensional dataset, the only solution left to us is to create either a 2-D or 3-D graph out of it. Let’s say for simplicity, we are creating 2-D graphs. Suppose we have 1000 features in the dataset. That results in a total (1000 \* 999)/2 = 499500 combinations possible for creating the 2-D graph.

is it humanly possible to analyze all those graphs to understand the relationship between the variables?

**The questions that we need to ask at this point are:**

* Are all the features really contributing to decision-making?
* Is there a way to come to the same conclusion using a lesser number of features?
* Is there a way to combine features to create a new feature and drop the old ones?
* Is there a way to remodel features in a way to make them visually comprehensible?

The answer to all the above questions is – Dimensionality Reduction Techniques.

**What is a Dimensionality Reduction Technique?**

Dimensionality reduction is a feature selection technique using which we reduce the number of features to be used for making a model without losing a significant amount of information compared to the original dataset. In other words, a dimensionality reduction technique projects data of higher dimensions to a lower-dimensional subspace.

**When to use Dimensionality Reduction?**

Dimensionality reduction shall be used before feeding the data to a machine-learning algorithm to achieve the following:

* It reduces the size of the space in which the distances are calculated, thereby improving machine learning algorithm performance.
* It reduces the degrees of freedom for our dataset avoiding chances of overfitting.
* Reducing the dimensionality using dimensionality reduction techniques can simplify the dataset facilitating a better description, visualization, and insight.

**Principal Component Analysis**

**Introduction**

Principal Component Analysis (PCA) is one of the prominent dimensionality reduction techniques. It is valuable when we need to reduce the dimension of the dataset while retaining maximum information. As the name suggests, it finds out the principal components from the data. PCA transforms and fits the data from higher-dimensional space to a new, lower-dimensional subspace. This results in an entirely new coordinate system of the points where the first axis corresponds to the first principal component that explains the most variance in the data.

**What are the principal components?**

Principal components are the derived features that explain the maximum variance in the data. The first principal component explains the most variance, the second a bit less, and so on. Each of the new dimensions found using PCA is a linear combination of the old features.

Let’s take the following example where the data is distributed like the diagram on the left:

Chart, line chart, scatter chart

Description automatically generated

In the above diagram, we are considering 3 orthogonal (C3 is in the third dimension) axes to show the distribution of data. If you notice the diagram on the right, the first two axes **C1** and **C2** successfully explain the maximum variation in the data whereas the axes **C3** only consist of a fewer number of points. Hence, while considering the principal components **C1** and **C2** will be our choices.

**Is there any pre-processing step required before applying PCA?**

We need to keep the below points in our mind before applying PCA

* PCA can not be applied to the dataset with null values. Hence, you need to treat null values before proceeding with PCA. There are different ways of treating null values such as dropping the variables and imputing the missing data using mean or median.
* We shouldn’t apply PCA on the dataset having attributes on different scales. We need to standardize variables before applying PCA.

**Case Study**

This dataset has 19 columns (or dimensions) we will try to reduce its dimension using PCA.

**Importing all the necessary libraries**

Text

Description automatically generated

**Importing data and exploring the top five observations**

Text

Description automatically generated

Table

Description automatically generated

Let’s check the statistical summary of our dataset to find the scale of different attributes.

Table

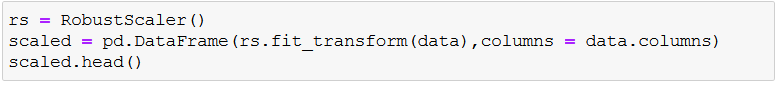
Description automatically generated

Above we can see that every attribute is on a different scale. Therefore, we can not jump to PCA directly without changing the scales of attributes.

We see that column “Post Weekday” has less variance and column “Lifetime Post Total Reach” has comparatively more variance.

Therefore, if we apply PCA without standardization of data then more weightage will be given to the “Lifetime Post Total Reach” column during the calculation of “eigenvectors” and “eigenvalues” and we will get biased principal components.

Now we will standardize the dataset using “RobustScaler” of the “sklearn” library. Other ways of standardizing data are provided in “sklearn” like “StandardScaler” and “MinMaxScaler” and can be chosen as per the requirement.



Table

Description automatically generated

**Who decides the number of Principal Components?**

Unless specified, the **number of principal components will be equal to the number of attributes.**

Our dataset has 18 attributes initially hence we get 18 principal components. These components are new variables that are in fact a linear combination of input variables.

Once we get the amount of variance explained by each principal component, we can decide how many components we need for our model based on the amount of information we want to retain.

Principal components are uncorrelated with each other. These principal components are known as eigenvectors and the variance explained by each **eigenvector** is known as **eigenvalues**.

Below we have applied PCA on the scaled datasets. If we want a predefined number of components, then we can do that it using PCA (n\_components)

Text

Description automatically generated

Text, letter

Description automatically generated

Here the output is the variance explained by each principal component. We have 18 attributes in our dataset and hence we get principal components.

Always remember that the first principal component will always hold maximum variance.

You can observe the same in the output that the first principal component holds maximum variance followed by subsequent components.

**Interpretation of Principal Component**

Now we have 18 principal components, and we will try to find out how these components are influenced by each attribute. We can check the influence of the top 3 attributes (both positive and negative) for the first principal component.

Let’s check the influence of attributes on principal components by changing the number of features and number of components.

Text

Description automatically generated

Chart, waterfall chart

Description automatically generated

Chart

Description automatically generated with medium confidence

We can interpret here that our first principal component is mostly influenced by engagement to the post (like, share, and comment)

Likewise, we can interpret other principal components as per the understanding of data using the above plot.

**Plot to visualize variance by each principal component: Scree Plot**

Below you can see a scree plot that depicts the variance explained by each principal component.

Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

Here we can see that the top 8 components account for more than 95% variance. We can use these 8 principal components for our modeling purpose.

Finally, we reduce the number of attributes to 8 from the initial 18 attributes. We were also able to retain 95% information of our dataset.

**SUMMARY**

We have started with understanding what is unsupervised machine learning then moved onto cluster analysis technique, which is an unsupervised machine learning model, walked through the complete process of clustering using K-Means and Hierarchical cluster algorithms, seen methods to select the appropriate number of clusters, and finally, we have understood the concept of dimensionality reduction with the help principal component analysis algorithm.

**Program Assignment**

Build a Churn algorithm to predict which customers are at risk of leaving the brand.

Follow the below step by step approach:

1. Data importing
2. Data exploration
3. Data cleaning
4. Feature Scaling
5. Cluster analysis

**Assessment**

**Choose the appropriate option**

1. Movie Recommendation systems are an example of:
   1. Classification
   2. Clustering
   3. Reinforcement Learning
   4. Regression
2. Which of the following can act as possible termination conditions in K-Means?
   1. For a fixed number of iterations
   2. Assignment of observations to clusters does not change between iterations. Except for cases with a bad local minimum.
   3. Centroids do not change between successive iterations.
   4. Terminate when WCSSE does not change further.
   5. All of the above
3. What is the most appropriate no. of clusters for the data points represented by the following dendrogram:

Chart, shape

Description automatically generated

* 1. 2
  2. 4
  3. 6
  4. 8

1. Which of the following metrics, do we have for finding dissimilarity between two clusters in hierarchical clustering?
   1. Single-link
   2. Complete-link
   3. Average-link
   4. All of the above
2. The most popularly used dimensionality reduction algorithm is Principal Component Analysis (PCA). Which of the following is/are true about PCA?
   1. PCA is an unsupervised method
   2. It searches for the directions that data have the largest variance.
   3. Maximum number of principal components <= number of features
   4. All principal components are orthogonal to each other
   5. All of the Above

**Fill in the spaces with appropriate answers**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ clustering algorithms consider each data point as an individual cluster in this technique. After each iteration, the similar clusters merge with other clusters and the merging wills stop until one cluster or k clusters are formed.
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ method, the distance between two clusters is defined as the minimum distance between two data points in each cluster.
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ clustering algorithm is most sensitive to outliers as it uses the mean of cluster data points to find the cluster center.
4. Feature scaling is an important step before applying \_\_\_\_\_\_\_\_\_\_\_\_\_ algorithm.
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ method is used for finding optimal of cluster in K-Mean algorithm?

**True or False**

1. For two runs of K-Means clustering is it expected to get the same clustering results?
   1. True
   2. False
2. Is it possible that the Assignment of observations to clusters does not change between successive iterations in K-Means?
   1. True
   2. False
3. Imagine, you have 1000 input features and 1 target feature in a machine learning problem. You must select the 100 most important features based on the relationship between input features and the target features. Do you think, this is an example of dimensionality reduction?
   1. Ture
   2. False
4. In Divisive Hierarchical clustering, we consider all the data points as a single cluster, we separate the data points from the cluster which are not similar.
   1. True
   2. False
5. In the Average-linkage method, we find the centroid of cluster 1 and the centroid of cluster 2 and then calculate the distance between the two before merging.
   1. True
   2. False

**Assessment Solutions**

**Choose the appropriate option**

1. B
2. E
3. B
4. D
5. E

**Fill in the spaces with appropriate answers**

1. Agglomerative
2. Single-linkage
3. K-Means
4. Clustering algorithms
5. Elbow Method

**True or False**

1. False
2. True
3. True
4. True
5. False