# **ML Interview Task 3**

# **Unsupervised Learning- Synopsis**

## Pre-processing data, handling null values

1. First of all, “poutcome” can be separated from the other columns, since it is not an independent variable that affects the dataset.
2. Then, the missing data (null values) need to be handled. This can be done by various ways, and the choice of method largely depends on the type of column.
   * The rows containing the null values can be removed. However, if the number of such rows is large, then the amount of loss of data will be profound.
   * If a certain column has a significant number of null values, and it doesn't seem of enough value, then the column can be dropped.
   * If there are null values in columns- [job, marital, education, default, housing, loan], then the value should be set to unknown, as that is an available option.
   * If there are null values in columns like “age", then they can be assigned the mean of the column.
   * In case of columns- [contact, month and day\_of\_week], a new variable that will represent the “null” condition should be formed (if deleting the data will cause a lot of loss).
3. For the categorization of non-numeric columns into something that the machine-learning algorithm can conduct its calculations about, they should be assigned labels. For example:
   * Yes or no values can be assigned ‘1’ and ‘0’. Unknown can be assigned ‘2’.
   * Columns containing “day\_of\_week” and “month” can be custom encoded using one hot encoding and given labels of natural integers.
   * For important and influential attributes like “job”, “marital”, “housing”, etc., OneHotEncoder() can be performed on it, to make the data clearer.
4. For an attribute like “age”, data scaling is important. This can be done by data normalization. This is done by:

where x is any value from the feature x and min(x) is the minimum value from the feature and max(x) is the maximum value of the feature.

## Clustering algorithm

DBSCAN- Density Based Spatial Clustering of Applications with Noise.

It is a density-based clustering non-parametric algorithm. Given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbours), marking as outliers points that lie alone in low-density regions (whose nearest neighbours are too far away).

It is a very useful algorithm, and in my opinion, it is better than other clustering algorithms like K-Means because it isn’t confined to a specific shape, which generally tends to be circular/spherical because of the distance-based algorithms. For the data sets that are as vast and unpredictable as the given data set, DBSCAN can perform better.

Another big advantage is the exclusion of outliers by rightly characterizing them as noise. In the given dataset, there will definitely be a lot of randomness, and if it is tried to be included in the clusters, can bring a lot of deviation from the required goal. Hence, DBSCAN is chosen.

There are 2 basic hyper-parameters in DBSCAN:

1. Epsilon(ε)

It defines the distance from the point in question (say ‘P’) to any point in its vicinity (say ‘Q’), in order to determine if point ‘Q’ is a neighbour of ‘P’.

It basically gives the radius of the circle in which, if point ‘Q’ lies, it becomes a neighbour.

The value of ‘ε’ needs to be chosen very carefully, as if it is too large, then most of the points will be labelled as the ‘neighbour’, and the number of clusters will reduce, which is an undesired result. However, if it is too small, then a large number of points will not be able to have a neighbour in their vicinity, and will become outliers. This too, is highly undesirable.

1. MinPts

It gives the minimum number of points which neighbour the given data point ‘P’ within a radius of ‘ε’ (epsilon).

This value, too, needs to be chosen carefully. If this is too small, then the boundary of the cluster will change, and the data points which are supposed to be outliers may change. In fact, if the clusters are close, then it might even merge two clusters.

However, if it is too large, it might limit the capacity of the cluster to hold enough data points. The number of clusters present will then increase drastically.

Based on these 2 hyper-parameters, there are 3 types of data-points:

1. A picture containing diagram

   Description automatically generatedCore point (labelled ‘A’ in the diagram)

It is a point which has atleast ‘MinPts’ number of data points in its vicinity (a circle with the radius of ‘ε’ (epsilon).

1. Border point (labelled ‘B’ and ‘C’ in the diagram)

It is a point which has atleast 1 core point in its vicinity, but less than the ‘MinPts’ number of points.

1. Noise point (labelled ‘N’ in the diagram)

It is a point which doesn’t have any data point in its vicinity.

The algorithm first identifies the core points with the given values of ‘ε’ (epsilon) and ‘MinPts’.

It then takes any one core point (say ‘C’) and assigns it a new cluster. Any other core points in its neighbourhood are assigned to the same cluster. This process goes on till all the core points connected in chain with the original point ‘C’ are assigned the same cluster.

Next, the border points having a core point of the cluster in its neighbourhood is assigned to the cluster.

The algorithm then goes on to start this process with another core point to form a new cluster. Once all the clusters are formed, the remaining data points are noise points.

## Number of clusters

In this algorithm, there is no need to input the number of clusters. However, the values of epsilon and ‘MinPts’ need to be chosen with care. The reasons have already been mentioned above. The choosing of the values does, in fact, require a trial-and-error approach. It is not expected that the first values input will precisely give the required result.

A good way of measuring whether the values give desirable output is by using the ‘Silhouette Coefficient’ or ‘Silhouette score’. It depends on the distance of data points from other data points in their cluster and the next nearest cluster.

Its value ranges from -1 to 1. Values close to 0 indicate overlapping clusters, while negative values usually indicate that data points have been assigned to the wrong clusters. The closer the value to 1, the better the model.