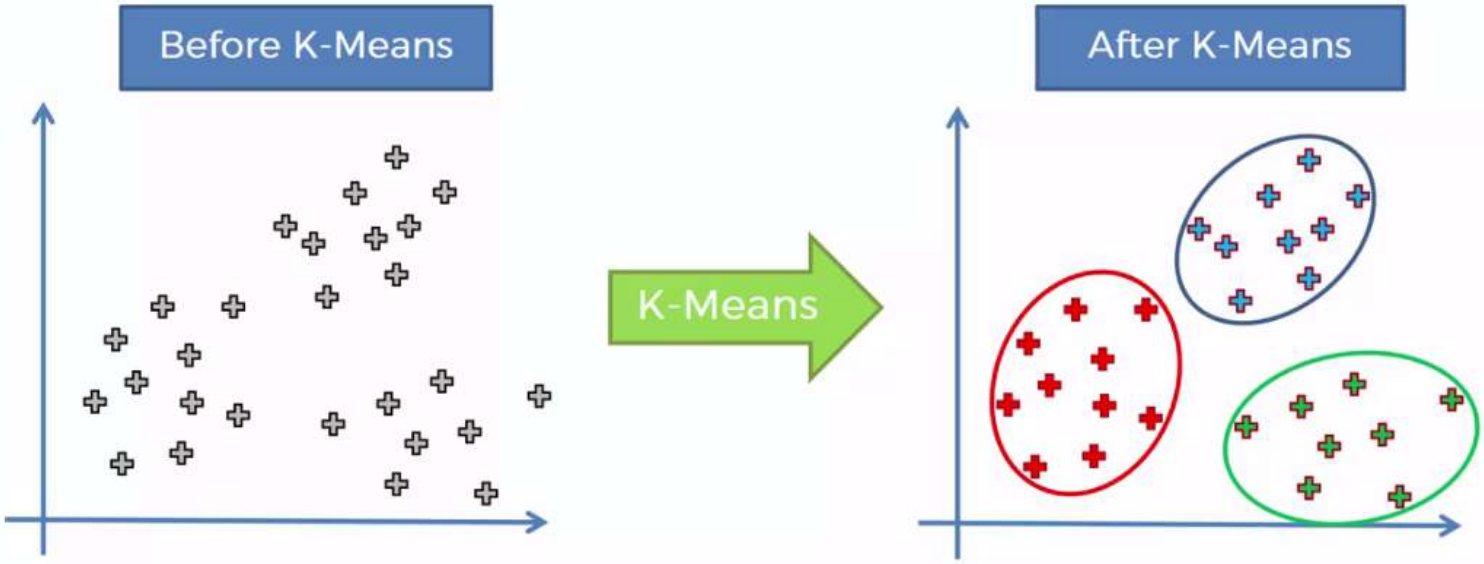


Table Of Contents

Main Requirements for clustering Algorithms

1. It should be scalable.
2. It should be able to deal with attributes of different types.
3. It should be able to discover arbitrary shape clusters.
4. It should have an inbuilt ability to deal with Noise & outliers.
5. The cluster should not vary with the order of input records.
6. It should be able to handle data of high dimensions.
7. It should be easy to interpret & use.

10 K-means

Old Algorithm

Proposed in 1957 by Stuart Lloyd, AUO
In 1965 by Edward W. Forgy. so
Sometimes called Lloyd-Forgy Algo.

→ K-means is a clustering approach in which data is grouped into K distinct non overlapping clusters based on their distances from the K centres.

→ The value of 'K' needs to be specified first & then the algorithm allocates the points to exactly one cluster.

Theory of K-means

→ Let $C_1, C_2, \ldots C_k$ be the K-cluster.

→ then we can write :-

$C_1 \cup C_2 \cup C_3 \cup \ldots \cup C_k = \{1, 2, 3, \ldots n\}$ → i.e, Each data point has been assigned to a cluster

And also, $C_k \cap C_{k'} = \phi$ ⟶

this means, cluster are non-overlapping.

→ the idea behind the K-means approach is that within-cluster variation amongst the point should be minimum.

The within-cluster variance is denoted by :- $W(C_k)$

→ Hence, according to statement above, we need to minimize the variance for all cluster mathematically,

$$\underset{C_1, C_2, \ldots C_k}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

WCSS :- within cluster summation of squares

→ the next step is to define the criterion for measuring the within cluster variance. Generally, the criterion is the Euclidian distance between two datapoints.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

→ the above formula says that we are ① calculating the distances b/w all the points in a cluster ② then we are repeating it for all the K-clusters (so two summations)

Eg:-

Let's take 4 points

Case ①:-

4 Points
4 clusters

→ Now, each point acts as a cluster and so it is centroid or itself.

∴ Distance b/w the point & centroid (same point) is zero (0).

→ And it is same for all the 4 clusters on this.

i.e) $w(C_4) = 0+0+0+0 = 0$

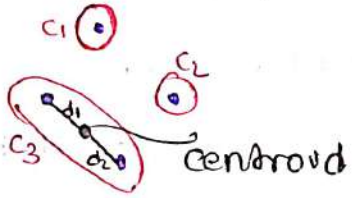
$C_1\ C_2\ C_3\ C_4$

↓

(Summation of distances)

Case ②:-

4 Points
3 clusters



→ Now we need distance b/w points & the centroid of cluster it is present in.

$w(C_3) = C_1 + C_2 + C_3$

$w(C_3) = 0 + 0 + (d_1 + d_2)$

(w) (w) (summation of distances in a cluster)

① Individual sum of distances of a cluster

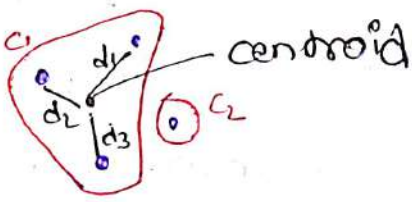
$\sum_{i=1}^{n} (x - c)^2$ ← centroid

② All the clusters sum

$\sum_{k=1}^{k} \sum_{i=1}^{n} (x - c)^2$ //

Case ③:-

4 Points
2 clusters



$w(C_4) = C_1 + C_2$

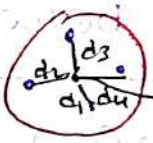
↓

$w(C_4) = (d_1 + d_2 + d_3) + 0$

(distance b/w point & centroid) (summation of distances in a cluster.)

Summation of the individual summations of distances in cluster

Case ④:- 4 Points, 1 cluster



$w(C_1) = C_1$

↓

$w(C_4) = (d_1 + d_2 + d_3 + d_4)$

Goal is to minimize $w(c_1)+w(c_2)+w(c_3)+w(c_4)$ → low variance for all clusters

So, we got ↓

$w(c_4) = 0$

$w(c_3) = d_1 + d_2$

$w(c_2) = d_1' + d_2' + d_3'$

$w(c_1) = d_1'' + d_2'' + d_3'' + d_4''$

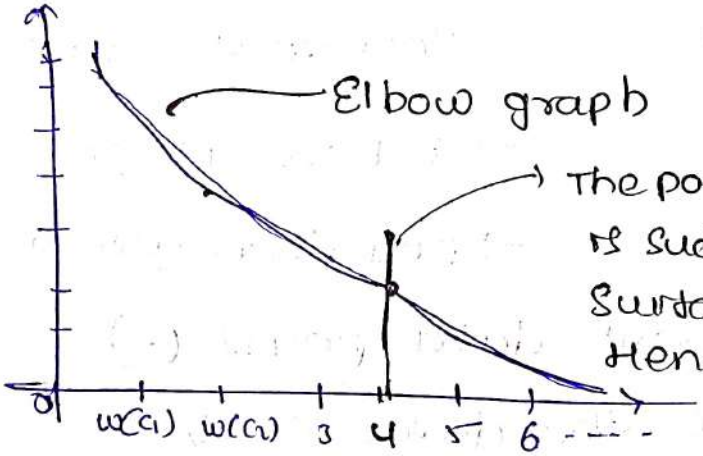
→ By observation, we can say that

As no. of cluster increases, $w(c_k)$ value decreases & tends to '0'

$$\therefore w(c_k) \propto \frac{1}{\text{no. of cluster}(k)}$$

$w(c_1) > w(c_2) > w(c_3) > \cdots > w(c_k)$

and the graph goes as -



within cluster summation of squares (WCSS)

→ this is how we find the value of 'k' in k-means.

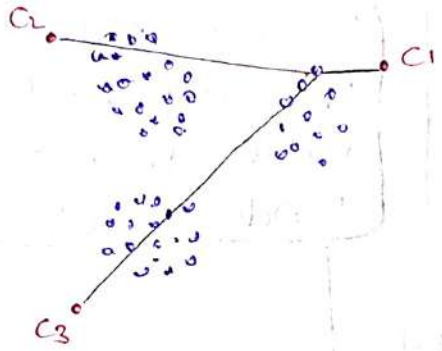
But, here is the problem in k-means,

(*) After selecting k value, it takes k-points as centroids and starts building the cluster normally.

But thing is where do we place those k-points ?

(*) cluster building process

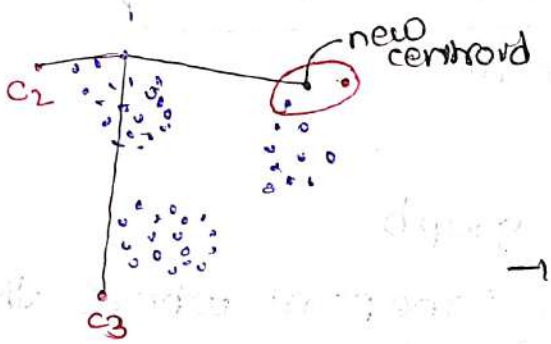
Let's say we got: (K=3)



→ After, randomly placing the K points,

→ It takes a point from our data (·) and measures the distance to all the K-centroid points (·),

And whichever distance is smaller, the point goes that cluster.



→ So, here it goes to C_1

→ And then gets a new centroid.

→ Now for next data point (·) we measure the distance of point to the "new centroids"

→ And in this way, cluster keeps building and centroid gets updated.

→ This process continues until all the data points are finished.

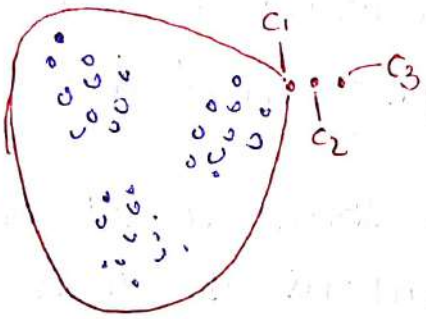
this is how grouping happens.

Edge case scenario

Problem with K-means

→ In the above example, we took K=3 and then it generated 3 random points and build cluster.

→ So, what if all the 3 points are close?



If all are close to only C_1.

And,
$C_2 = 0$
$C_3 = 0$
} elements

In this case, all data points would go to cluster ① (C_1)

which is not what we are expecting!!

Algorithm

① Find WCSS for K-groups possible (max K = n) (n = no. of elements)

② Build Elbow graph, and understand K-value (point where there's distinction)

③ Randomly assign K-centres.

④ calculate distance of all points from all K-centres and allocate the points to cluster based on shortest distance to "cluster centroid".
The model's inertia is the mean squared distance b/w each instance & closest centroid.
Goal is to have model with low inertia.

⑤ Once all points assigned to cluster recompute centroids.

(6) Repeat steps 4 & 5 until the location of the centroids stop changing and the cluster allocation of the points becomes constants.

Note:-

→ K-means is only recommended for spherical data (or) symmetrical data.

→ For, non spherical data there won't be fair chance for k clusters to form clusters, and it will become high biased cluster.

(*) Challanges and Improvements in K-means:-

(1) we need to specify the no. of clusters beforehand.

(2) It is required to run the algorithm multiple times to avoid a suboptimal solution.

(3) K-means does not behave very well when the clusters have varying sizes, different densities or non-spherical shapes.

(4) An improvement to K-means is propoled in 2003 by charles Elkan.

→ It accelerates the algorithm by many unnecessary distance calculations ; which is achieved by exploiting the "TRIANGLE INEQUALITY"

In a triangle with sides a, b & c i.e. $a+b>c$
the straight line is always shortest

⑤ Another important improvement to k-means algorithm called "k-means++", was proposed in 2006 by David Arthur & Sergei.

→ They introduced a smarter initialization step that tends to select centroids that are distant from one another, and this makes the k-means algorithm much less likely to converge to a suboptimal solution.

⑥ Yet another important variant of the k-means algorithm was proposed in a 2010 paper by David Sculley.

→ Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroids just slightly at each iteration. This speeds up the algorithm typically by a factor of 3 or 4 and makes it possible to cluster huge datasets that do not fit in memory. Scikit-Learn implements this algorithm in the "MiniBatchKMeans" class.

Initially, All points as seperate clusters

(11) Hierarchical Clustering

Bottom-up / Agglomerative approach

→ one main disadvantage of K-means is that it needs us to pre-enter the no. of clusters (k).

→ Hierarchical clustering is an alternative approach which does not need us to give the value of k beforehand and also, it creates a beautiful 'tree-based structure' for visualisation).

→ we start by defining any sort of similarity between the datapoints. Generally, we consider Euclidian distance. the points which are closer to each are more similar than the points which are farther away.

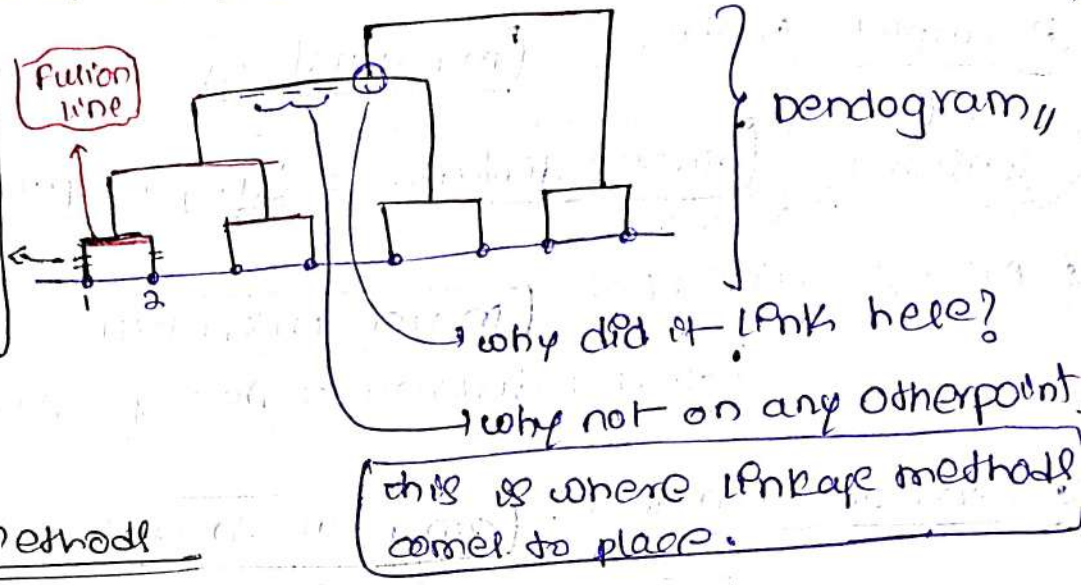
Algorithm

① Begin with n observations and a measure (such as Euclidian distance) of all the $n(n-1)/2$ pairwise dissimilarities (distances). Treat each observation as its own cluster. So, initially we have n clusters.

② compare all the distances and put two closest points/clusters in the same cluster. the dissimilarity between these two clusters indicates the height in the dendogram at which "fusion line" should be placed.

③ compute the new pairwise inter-cluster dissimilarities among the remaining clusters.

④ Repeat steps 2 & 3 till we have only one cluster left.



Linkage methods

→ Based on pairwise distances, we can now compute a linkage matrix.
the linkage matrix is simply a table listing which pairs of points are merged at what step & what distance.

→ we can cut dendogram to form flat clusters.

* we know, initially HC starts with clusters consisting of individual points.

→ later it compares the clusters with each other and merges the two "closest clusters".

→ since clusters are pair of points, there are many different kinds of linkage methods.

Notes

⊙ Single Linkage:-

cluster distance = smallest pairwise distance

⊙ complete Linkage:- maximal intercluster distance

less sensitive to outliers

cluster distance = Largest pairwise distance

⊙ Average Linkage:- mean intercluster dissimilarity

cluster distance = Average pairwise distance

⊙ centroid Linkage:- can result in undesirable inversions

cluster distance = distance between the centroids of clusters

⊙ ward's Linkage:- Before & after merging

cluster distance = minimize the variance in cluster.

Simple Linkage:-

minimal intercluster dissimilarity.

→ Single linkage can result in extended, trailing cluster in which single observations are fused one-at-a-time.

→ cluster distance is the smallest distance b/w any point in cluster ① & any point in cluster ②

→ High sensitive to outliers when forming flat clusters.

→ works well for low-noise data with unusual structures

(12) DBSCAN (Density Based Spatial Clustering of Applications with Noise.)

→ It is an unsupervised machine learning algorithm. This algorithm defines clusters as continuous regions of high density.

Key words :-

⊙ **Epsilon :-** (EPS)

This is the distance fn which we look for the neighbouring points.

⊙ **min-points :-**

The min no. of points specified by the user

⊙ **core point :-**

If the no. of points inside the epsilon radius of a point is greater than or equal the min_points then it is called a **core point**.

⊙ **Border point :-**

If the no. of points inside the epsilon radius of a point is less than the min points and it lies within the eps radius region of a core point, then it is called border point.

⊙ **Noise :-**

A point which is neither core nor a border point is a noise point.

Algorithm steps

① The algorithm starts with a random point in the dataset which has not been visited yet and its neighbouring points are identified based on the eps value.

② If the point contains ≥ points than min points then the cluster formation starts and this point becomes a core point;
else it's considered as Noise.

→ The point to note here is that a point initially classified as noise can later become a border point if it's in the eps radius of a core point.

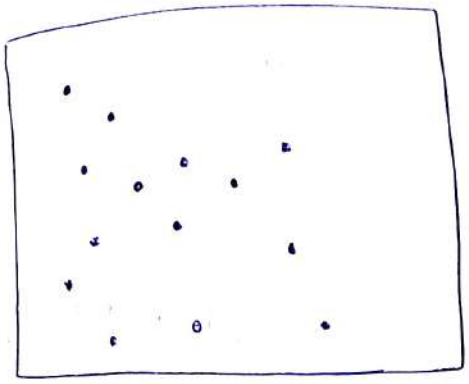
③ If the point is a core point, then all its neighbours become a part of cluster. If the points in the neighbourhood turn out to be core points then their neighbours are also part of cluster.

④ Repeat the steps above until all points are classified into different clusters or noises.

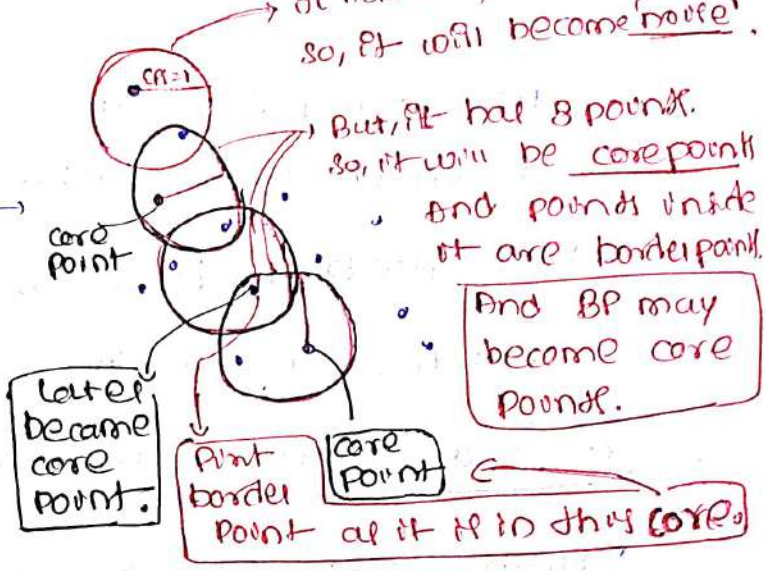
This algo works well if all clusters are dense enough, and they are well separated by low dense regions

Eg:-

Let's say eps = 1, min point = 3.



data.



→ In short, DBSCAN is a very simple yet powerful algorithm, capable of identifying any no. of clusters of any shape, it is robust to outliers, and it has only two hyperparameters.

(eps & min samples)

→ However, if the density varies significantly across the clusters, it can be impossible for it to capture all the clusters properly. Moreover its computational complexity is roughly $O(m \log m)$, making it pretty close to linear with regards to the no. of instances.

//

⊛ Evaluating of clustering:

Cluster validity:

The validation of cluster created is a troublesome task.

→ The problem here is "cluster are in the eyes of the beholder".

A good cluster will have:

- High inter class similarity
- Low inter class similarity.

Aspects of cluster validation:

- External: compare your cluster to ground truth.
- Internal: Evaluating the cluster without reference to the external data.
- Reliability: the clusters are not formed by chance (randomly) — Some statistical framework can be used

⊙External measures

→ No. of objects in the data P :- $P_1, P_2, \ldots P_m$

the set of ground truth clusters C :- $C_1, C_2 \ldots C_n$

the set of clusters formed by the algorithm.

the Incidence matrix N * N matrix.

→ $P_{ij} = 1$, if the two points O_i & O_j belong to the same cluster in the ground truth

else, $P_{ij} = 0$.

→ $C_{ij} = 1$, if the two points O_i & O_j belong to the same cluster in the ground truth

else, $C_{ij} = 0$

Now there can be following scenario

1) $C_{ij} = P_{ij} = 1$ ⟶ Both the points belong to the same cluster for both our algorithm & ground truth (Agree) — SD

2) $C_{ij} = P_{ij} = 0$ ⟶ Both the points don't belong to same cluster for both our algorithm & ground truth (Agree) — DD

3) $C_{ij} = 1$ but $P_{ij} = 0$ → the points belong in the same cluster for our algorithm but different cluster in ground truth (Disagree) — SD

4) $C_{ij} = 0$ but $P_{ij} = 1$ → the points don't belong in same cluster for our Algo but same cluster in ground truth (disagree) — DS

↑ Just like accuracy score

$$\text{Rand Index} = \frac{\text{total Agree}}{\text{total Disagree}} = \frac{(SS+DD)}{(SS+DD+DS+SD)}$$

→ the disadvantage of this is it can be dominated by DD.

$$\text{Jaccard coefficient} = \frac{SS}{(SS+DS+SD)}$$

∴ A higher value of Rand Index & Jaccard coefficient mean that the clusters generated by our algorithm mostly agree to the ground truth.

confusion matrix

n = no. of points
m_i = Points in cluster i
C_j = Points in class j
n_{ij} = Points in cluster i coming from class j.
$P_{ij} = \frac{n_{ij}}{m_i}$

(Probability of element from cluster i to be assigned to class j

	class 1	class 2	class 3	
cluster 1	n_{11}/P_{11}	n_{12}/P_{12}	n_{13}/P_{13}	m_1
cluster 2	n_{21}/P_{21}	n_{22}/P_{22}	n_{23}/P_{23}	m_2
cluster 3	n_{31}/P_{31}	n_{32}/P_{32}	n_{33}/P_{33}	m_3
	C_1	C_2	C_3	n

→ Entropy of cluster P_i

$$e_i = -\sum P_{ij} (\log P_{ij})$$

→ For entering clustering algorithm, entropy can be given as

$$e = \sum \frac{m_i}{n} e_i$$

4

→ Purity of cluster i, $p_i = \max(p_{ij})$

And for entire cluster it is $P(C) = \sum \frac{m_i}{n} \cdot p_i$

→ Purity, is the total percentage of data points "clustered correctly."

→ A high value of purity score means that our clustering algorithm performs well against the ground truth.

Note:-

→ In calculating External measures, most of time we don't have ground truth,

---

⊙ External measures:-

These are the methods we do measure the quality of cluster without external reference. there are two aspects of it.

Cohesion:-

How closely the objects in the same cluster are related to each other. It is the within-cluster sum of squared distances. It is the same metric that we used to calculate for K-means Algorithm.

$$WCSS = \sum \sum (x - m_i)^2$$

Separation:-

How different objects in different clusters are and how distinct a well separated cluster is from other clusters.

It is the between cluster sum of squared distance

$$BSS = \sum c_i (m - m_i)^2$$

where, c is the size of individual cluster & m is centroid of all data points

Note:-

✓ BSS + WSS is always a constant

→ the silhouette can be calculated as:-

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

where,

a(x) is avg. distance of x from all other point in same cluster.

b(x) is avg. distance of x from all other points in other clusters.

And silhouette coefficient is,

$$SC = \frac{1}{N} \sum s(x)$$

Higher s(x) means that the ____ inter cluster similarity is less and the intra cluster dissimilarity is more (Good clustering)

(*) The curse of Dimensionality.

→ In the real world datasets, we often encounter data with hundreds or even thousands of features. Problems with more features/dimensions.

① As the majority of the machine learning algorithms rely on the calculation of distance for model building, and as the number of dimensions increases, it becomes more & more computationally intensive to create a model out of it.

Eg:

→ To calculate the distance b/w two points in just one dimension, we can just subtract the co-ordinates of one point from another.

→ For 2D, $= \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$

→ And for ND, it becomes: $\sqrt{(a_1-a_2)^2+(b_1-b_2)^2+\cdots+(n_1-n_2)^2}$

→ This is the effort for just two points, just imagine the no. of calculations involved for all datapoints.

② Hard to visualize the relationship b/w features: If we have an n-Dimensional dataset, the only solution left is to create either a 2D/3D graph out of it. Suppose we have 1000 features in the dataset, if we plan to create 2D graphs, we could need $(1000 \times 999)/2 = 499500$ combinations!! And we know it is not possible to spend time to analyze that many graphs to understand the relationship b/w the variables.

→ And like this, when we have huge no. of features, we need to ask the questions like -

① Are all the features really contributing to decision making.

② Is there a way to come to the same conclusion using lesser no. of features.

③ Is there a way to combine features to create a new feature and drop the old ones.

④ Is there a way to remodel features in a way to make them visually comprehensible.

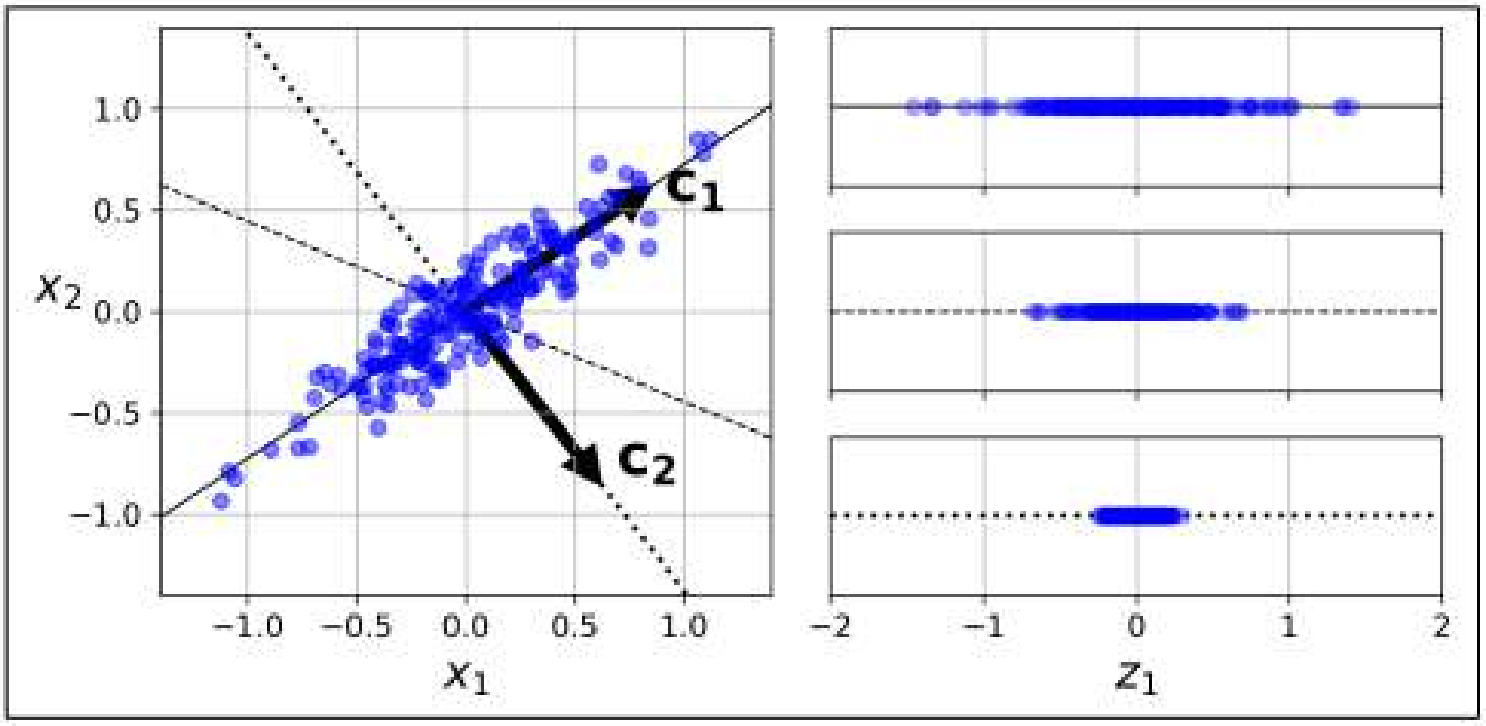
The answer to all the above questions is :-

⊙ Dimensionality Reduction Technique

→ Dimensionality reduction is a feature selection technique using which we reduce the no. of features to be used for making a model without losing significant amount of information compared to original dataset.

→ In other words, a dimensionality reduction technique projects a data of higher dimension to a lower dimension subspace.

→ DR technique should be used before feeding the data to a machine learning algorithm, it reduces the space in which the distances are calculated, thereby improving ML algorithm performance



Understanding Principal Component Analysis

Table Of Contents

(*) Principal component Analysis

→ The most popular dimensionality reduction technique is the principal component analysis, it is an unsupervised algorithm used for feature selection.

Idea behind PCA

→ PCA transforms and fits the data from a higher dimensional space to a lower dimensional subspace, this results into an entirely new co-ordinates system of the points where the first axis explains the most variance in the data (first principle component), this is the idea behind PCA.

what are principal components

→ Principal components are the axis, that explain the maximum variance in the data, the first principal component explains the most variance, the 2nd a bit less and so on..

→ Each of new axis / principal component found using PCA is a linear combination of the original features

→ All the principal component are uncorrelated and orthogonal to each other.

(*) Math Behind PCA

→ Principal component analysis can be implemented in two approaches, each with its unique focus and method. They are as follows:

① Eigen decomposition Method

② singular value decomposition Method,

Lets delve into them

① Eigen Decomposition Method

→ Eigen decomposition is a process that decomposes a square matrix A into a set of eigenvectors and eigenvalues.

→ For a given square matrix A, the eigenvectors are the non-zero vectors that satisfy the equation.

$$\boxed{A = \lambda V}$$ where,

V is the eigen vector and

λ is the corresponding eigenvalue.

Steps of Eigen-decomposition approach

① Standardize columns of A, so that each feature has a mean of zero, this ensures that each feature contributes equally to the analysis.

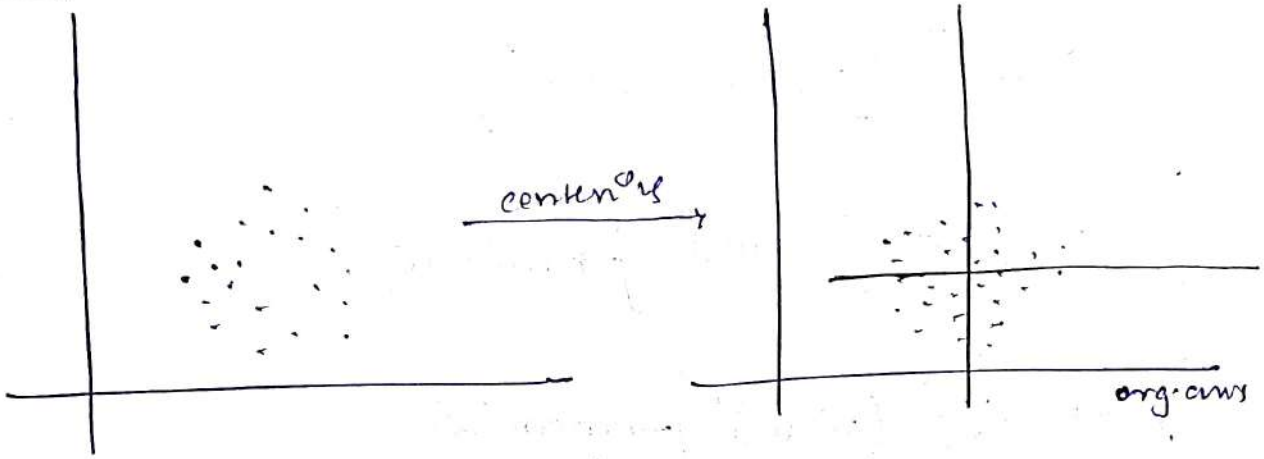
(2) compute co-variance matrix from the normalized data, $\Sigma = A^T A / (m-1)$

(3) perform eigen decomposition of Σ [covariance matrix], to obtain eigen values & eigenvectors.

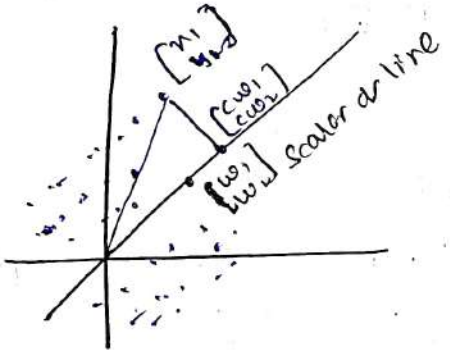
(4) choose the top K eigen vectors corresponding to the largest eigen values to construct the principal components as AX_K → Kth eigen vector.

Proof

Step 1



we have, $D = \{x_1, x_2, \ldots x_n\}$ $x_i \in \mathbb{R}^d$



the projection point will be, $(x^T w) \cdot w$, considering w is a scalar or the best fit line/ the line that has least reconstruction error.

$$\text{Error}(\text{line}, \text{dataset}) = \sum_{i=1}^{m} \text{error}(\text{line}, x_i)$$

$$= \sum_{i=1}^{m} \| x_i - (x^T w) w \|^2$$

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} \| x - (x^T w)\cdot w \|^2$$

This is the mean of the residuals, we can think of this as the avg. error.

$$= \frac{1}{n}\sum_{i=1}^{n} \left[(x - (x^T w)\cdot w)^T \cdot (x - (x^T w)\cdot w) \right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left[x_i^T x_i - (x_i^T w)^T - (x_i^T w)^2 + [x_i^T w^2] \right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left[x_i^T x_i - (x_i^T w)^2 \right]$$

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} -(x_i^T w)^2 \rightarrow \min w$$

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} (x_i^T w)^2 \rightarrow \max w$$

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} (x_i^T w)(x_i^T w) = \frac{1}{n}\sum_{i=1}^{n} w^T (x_i \cdot x_i^T) w$$

$$J(w) = w^T \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} x_i \cdot x_i^T \right)}_{\text{we know this is covariance matrix}} \cdot w$$

Each column corresponds to one data point.

$$\therefore \max(w) = w^T C \cdot w$$

where, C – covariance matrix
w – Eigen vector corresponding to max eigen value of C_{11}

→ so, Basically we can say that, the line which represents the data forms or minimum reconstruction error is same as the line which maximizes the $w^T C w$, as C - covariance matrix,
and the line is given by eigen vector corresponding to max. eigen value of C, and this line is the principal component!!

★ First principal component $(w_1) = \arg\max_w (w^T C \cdot w)$

→ Now, any line which minimizes the sum of errors / residuals, must also be orthogonal to w_1.
Hence, second principal component will be orthogonal to the first $w_2^T w_1 = 0$ ★

→ By, continuing this procedure, we get $\{w_1, w_2 \dots w_d\}$
such that,

$$\|w_k^2\| = 1 \quad \forall k \text{ and}$$
$$w_i^T w_j = 0 \quad \forall i \neq j$$

Residuals after d rounds

$$\forall i \rightarrow n_i - \left[(x_i^T w_1) w_1 + (x_i^T w_2) w_2 + \dots + (x_i^T w_k) w_d\right] = \vec{0}$$

$$\forall i \rightarrow x_i = (x_i^T w_1) w_1 + (x_i^T w_2) w_2 + \dots + (x_i^T w_d) \cdot w_d$$

→ If original dataset is of $(d \times n)$, then based on above eqn, after k rounds of PCA, we get $(d \times k) + (k \times n)$

Eg

→ Suppose we have an initial dataset with 50 features and 100 samples. $d=50$, $n=100$

Data dimensions $50 \times 100 = 1500$ data points to represent entire dataset

After 5 PCA rounds, $K=5$

data dimension $= (50 \times 5) + (5 \times 100) = 750$

So, now we only need half of datapoints to represent entire dataset

→ $\{w_1\ w_2\ w_3\ w_4\ w_5\}$ → Representation, common for dataset

→ $x_i \rightarrow [x_i^T w_1 \quad x_i^T w_2 \quad x_i^T w_3 \quad x_i^T w_4 \quad x_i^T w_5]$ → Data Point Specific

→ Hence, to reconstruct x_i we don't need 1500 numbers like naive way features

we only need 5 principal components

→ And through Eigen decomposition approach, we calculate the covariance matrix, and find its eigenvalues & eigenvectors. And define the principal components from eigen vectors. we can reconstruct by

$A \times k \times k^T$

② Singular value decomposition Method

→ SVD method factorises the dataset in such a way that it becomes a product of the multiplication of 3 individual matrices.

$$X(\text{original data}) = U * \Sigma * V^T$$

where, U is the matrix that contains the principal components.

Steps in SVD approach

① Standardize columns of A, to ensure each feature contributes equally to analysis.

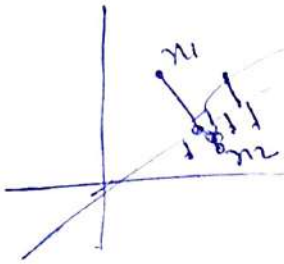
② compute SVD of the centered data matrix to obtain matrices U, Σ, and V^T, such that $X = U \Sigma V^T$ where U contains left singular vectors, Σ holds the singular values & V^T includes the right singular vectors.

③ select principal components: The principal components are obtained by selecting the columns of U corresponding to the largest singular values in Σ, these principal components represent the directions of maximum variance in the dataset.

Note:-

→ what does high variance mean?

when making a proxy,
what if two points get same proxy.



→ Now, How do we distinguish x_1 & x_2.

→ And How do we reconstruct x_1 & x_2.

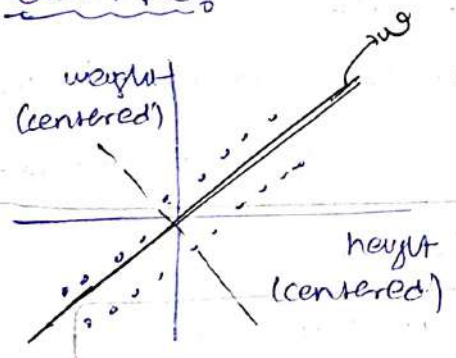
→ So, if all the datapoints are crowded, then we can't reconstruct and the information is lost.

This happens due to less variance / small variance.

Hence, we want directions where projections not crowded.

variance has to be high,
so the datapoints will be spread out
and reconstruction of points can be done properly.

example:



- height gives some information weight

→ we have two directions now w_1, w_2..

→ And these two directions are decorrelated, every direction is giving a new information from the other.

→ Hence, we can say these directions are de-correlated.

this's what the algorithm is essentially doing & it's called Principal Component Analysis.

* once we get principal components, Project the original data onto the subspace spanned by the selected principal components, effectively reducing the dimensionality of the dataset.

(*) What is the optimal no. of PC's needed?

* Before we ask this qn, we need to ask how much of the information in a given dataset is lost by projecting the observations onto first few principal components, i.e,

> the proportion of variance explained by each principal component.

Explained variance ratio = $\frac{\text{variance by } m^{th} \text{ PC}}{\text{total variance}}$

$$EVR = \frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} w_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

→ EVR of PC1 = $\frac{\text{Distance of PC1 Points}}{\text{Distance of PC1 Points + PC2 + ... PCn}}$

If EVR of PC1 = 0.91, then that means PC1 explains 91% of the variance of data.

→ Now that we have a metrics to get the variance explained, we can choose the no. of dimensions that add up to a sufficiently large proportion of the variance (eg- as %).

Eg:

EVR of PC1 = 0.84

EVR of PC2 = 0.10

EVR of PC3 = 0.05

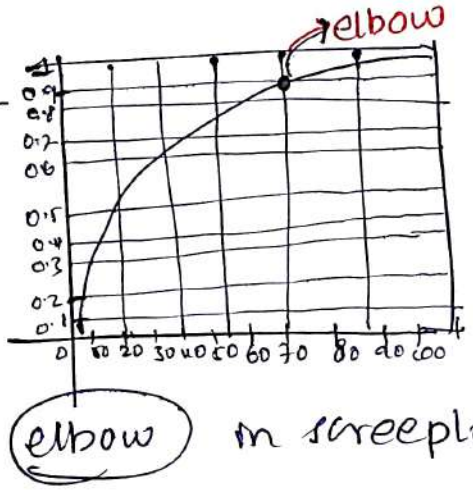
→ Now, if we want to preserve (>90%) variance, then we can just take PC1 & PC2, as they sum up to 0.94, 94%, >90%.

→ Instead if we want to preserve (>95%) of variance, then we should take PC1, PC2 & PC3. as the sum up to 0.99, 99%. >95%.

① Scree plots

→ Scree plots are the graphs that convey how much variance is explained by corresponding principal components.

→ So, by eyeballing the screeplot and looking for a point at which the proportion of variance explained by each subsequent PC drops off. This is often referred as elbow in screeplots.

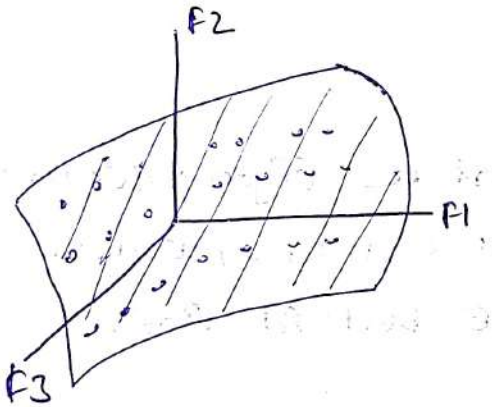


Issues / concerns with PCA

1. Time complexity
   - In particularly, covariance matrix.
   - Usually, it takes $O(d^3)$
   - If no. of features were large, (like 32000 for image) then d^3 is going to be very very large.

   Finding the eigen vectors and eigen values of covariance matrix is the time consuming step.

   Eg: Face recognition (Eigen faces)

2. 

   Data may not necessarily live in a low-dimensional Linear subspa

   → features will not be linearly related.

Notes

- we will have same solution for both the above issues, solving one will solve the other.

Kernel PCA

- Input — $\{x_1, \dots x_n\}$ $x_i \in \mathbb{R}^d$,

Kernel $K: \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$

Step ① compute $K \in \mathbb{R}^{n \times n}$,
where $K_{ij} = K(x_i, x_j)$ $\forall i, j$.

And then we need to center the kernel

Step ② compute $\beta_1, \dots \beta_l$ eigen vectors & $n\lambda_1 \geq \dots n\lambda_l$ eigen values.

~~step~~ and normalize to get, $\alpha_k = \frac{\beta_k}{\sqrt{n\lambda_k}}$ $\forall k$.

Step ③ $w_k = \phi(x) \cdot \alpha_k$

Eigen computation of kernel matrix. Hence, thus defeats the purpose, as it needs $\phi(x)$, which we tried to avoid all long!!

we cannot reconstruct the eigen vectors of the covariance matrix

→ But we can still compute the compressed representation.

$$\phi(x_i)^T \cdot w_k = \phi(x_i)^T \cdot \left(\sum_{j=1}^{n} \phi(x_j) \alpha_{kj} \right) = \sum_{j=1}^{n} \alpha_{kj} \phi(x_i)^T \phi(x_j)$$

$$\phi(x_i)^T w_k = \sum_{j=1}^{n} \alpha_{kj} \cdot K_{ij}$$

→ for downstream task, this is usually good enough.

→ Typically, the reason why we do kernel PCA is to get these projections and throw away original points and only retain these projections along each of these data points.

⚠ This is where, dimensionality Reduction happens.

(eg → original datapoint was 100 dimension, but then you only care about top 5.

→ So, we map the 100 dimensional data into a 5 dimensional projection onto these 5 most important directions.

→ That becomes a 5 dimensional representation of your 100 dimensional data

→ once, you learn these representation in a low dimensional space, as just the projection onto these eigen directions you can use these projection values as your data point and work this for a downstream task (supervised learning task).

modified

Step (3) compute $\sum_{j=1}^{n} \alpha_{kj} \cdot k_{ij} \quad \forall k$

$$x \in \mathbb{R}^d \longrightarrow \left[\sum_{j=1}^{n} \alpha_{1j} \cdot k_{ij}, \sum_{j=1}^{n} \alpha_{2j} \cdot k_{ij} \cdots \sum_{j=1}^{n} \alpha_{kj} \cdot k_{ij} \right]$$

→ It might increase the dimension, nevertheless it will allow you to capture more complicated relationships.

Note:

→ After step 2, we have to center the kernel.

→ Given, $K \in \mathbb{R}^{n \times n}$

$K_{ij} = k(x_i, x_j) \quad \forall i, j$

create a new kernel K^c — centered.

$$K^c_{ij} = K_{ij} - \theta_i 1_j^T - 1_i \theta_j^T + P 1_i \cdot 1_j^T$$

where,

$$\theta_i = \frac{1}{n} \sum_{k=1}^{n} K_{ik}$$

$$P = \frac{1}{n^2} \sum_{i,j} K_{ij}$$

→ Now, we can say we are able to solve both the issues by using kernel PCA.

– Supervised Algorithms –

Regression Models

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
Linear Regression	Linear Regression models a linear relationship between input variables and a continuous numerical output variable. The default loss function is the mean square error (MSE).	1. Fast training because there are few parameters. 2. Interpretable/Explainable results by its output coefficients.	1. Assumes a linear relationship between input and output variables. 2. Sensitive to outliers. 3. Typically generalizes worse than ridge or lasso regression.
Polynomial Regression	Polynomial Regression models nonlinear relationships between the dependent, and independent variable as the n-th degree polynomial.	1. Provides a good approximation of the relationship between the dependent and independent variables. 2. Capable of fitting a wide range of curvature.	1. Poor interpretability of the coefficients since the underlying variables can be highly correlated. 2. The model fit is nonlinear but the regression function is linear. 3. Prone to overfitting.
Support Vector Regression	Support Vector Regression (SVR) uses the same principle as SVMs but optimizes the cost function to fit the most straight line (or plane) through the data points. With the kernel trick it can efficiently perform a non-linear regression by implicitly mapping their inputs into high-dimensional feature spaces.	1. Robust against outliers. 2. Effective learning and strong generalization performance. 3. Different Kernel functions can be specified for the decision function.	1. Does not perform well with large datasets. 2. Tends to underfit in cases where the number of variables is much smaller than the number of observations.
Gaussian Process Regression	Gaussian Process Regression (GPR) uses a Bayesian approach that infers a probability distribution over the possible functions that fit the data. The Gaussian process is a prior that is specified as a multivariate Gaussian distribution.	1. Provides uncertainty measures on the predictions. 2. It is a flexible and usable non-linear model which fits many datasets well. 3. Performs well on small datasets as the GP kernel allows to specify a prior on the function space.	1. Poor choice of kernel can make convergence slow. 2. Specifying specific kernels requires deep mathematical understanding

Classification Models

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
SVM	In its simplest form, support vector machine is a linear classifier. But with the kernel trick. it can efficiently perform a non-linear classification by implicitly mapping their inputs into high-dimensional feature spaces. This makes SVM one of the best prediction methods.	1. Effective in cases with a high number of variables. 2. Number of variables can be larger than the number of samples. 3. Different Kernel functions can be specified for the decision function.	1. Sensitive to overfitting, regularization is crucial. 2. Choosing a "good" kernel function can be difficult. 3. Computationally expensive for big data due to high training complexity. 4. Performs poorly if the data is noisy (target classes overlap).
Negrest Neighbors	Nearest Neighbors predicts the label based on a predefined number of samples closest in distance to the new point.	1. Successful in situations where the decision boundary is irregular. 2. Non-parametric approach as it does not make any assumption on the underlying data.	1. Sensitive to noisy and missing data. 2. Computationally expensive because the entire set of n. points for every execution is required.
Logistic Regression (and its extensions)	The logistic regression models a linear relationship between input variables and the response variable. It models the output as binary values (o or rather than numeric values.	1. Explainable & Interpretable. 2. Less prone to overfitting using regularization. 3. Applicable for multi-class predictions.	1. Makes a strong assumption about the relationship between input and response variables. 2. Multicollinearity can cause the model to easily overfit without regularization.
Linear Discriminant Analysis	The linear decision boundary maximizes the separability between the classes by finding a linear combination of features.	1. Explainable & Interpretable. 2. Applicable for multi-class predictions.	1. Multicollinearity can cause the model to overfit. 2. Assuming that all classes share the same covariance matrix. 3. Sensitive to outliers. 4. Doesn't work well with small class sizes.

Both Regression and Classification Models

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
Decision Trees	Decision Tree models learn on the data by making decision rules on the variables to separate the classes in a flowchart like a tree data structure. They can be used for both regression and classification.	1. Explainable and interpretable. 2. Can handle missing values.	1. Prone to overfitting. 2. Can be unstable with minor data drift. 3. Sensitive to outliers.
Random Forest	Random Forest classification models learn using an ensemble of decision trees. The output of the random forest is based on a majority vote of the different decision trees.	1. Effective learning and better generalization performance. 2. Can handle moderately large datasets. 3. Less prone to overfit than decision trees.	1. Large number of trees can slow down performance. 2. Predictions are sensitive to outliers. 3. Hyperparameter tuning can be complex.
Gradient Boosting	An ensemble learning method where weak predictive learners are combined to improve accuracy. Popular techniques include XGBoost, LightGBM and more.	1. Handling of multicollinearity. 2. Handling of non-linear relationships. 3. Effective learning and strong generalization performance. 4. XGBoost is fast and is often used as a benchmark algorithm.	1. Sensitive to outliers and can therefore cause overfitting. 2. High complexity due to hyperparameter tuning. 3. Computationally expensive.
Ridge Regression	Ridge Regression penalizes variables with low predictive outcomes by shrinking their coefficients towards zero. It can be used for classification and regression.	1. Less prone to overfitting. 2. Best suited when data suffers from multicollinearity. 3. Explainable & Interpretable.	1. All the predictors are kept in the final model. 2. Doesn't perform feature selection.
Lasso Regression	Lasso Regression penalizes features that have low predictive outcomes by shrinking their coefficients to zero. It can be used for classification and regression.	1. Good generalization performance. 2. Good at handling datasets where the number of variables is much larger than the number of observations. 3. No need for feature selection.	1. Poor interpretability/explainability as it can keep a single variable. from a set of highly correlated variables.
AdaBoost	Adaptive Boosting uses an ensemble of weak learners that is combined into a weighted sum that represents the final output of the boosted classifier.	1. Explainable & Interpretable. 2. Less need for tweaking parameters. 3. Usually outperforms Random Forest.	1. Less prone to overfitting as the input variables are not jointly optimized. 2. Sensitive to noisy data and outliers.

– Unsupervised Algorithms –

Clustering Algorithms

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
K-Means	Most common clustering approach which assumes that the closer data points are to each other, the more similar they are It determines K clusters based on Euclidean distances.	1. Scales to large datasets 2. Interpretable & explainable results 3. Can generate tight clusters	1. Requires defining the expected number of clusters in advance. 2. Not suitable to identify clusters with non-convex shapes.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise can handle non-linear cluster structures, purely based on density. It can differentiate and separate regions with varying degrees of density, thereby creating clusters.	1. No assumption on the expected number of clusters. 2. Can handle noisy data and outliers 3. No assumptions on the shapes and sizes of the clusters 4. Can identify clusters with different densities	1. Requires optimization of two parameters 2. Can struggle in case of very high dimensional data
HDBSCAN	Family of the density-based algorithms and has roughly two steps: finding the core distance of each point, and expands clusters from them. It extends DBSCAN by converting it into a hierarchical clustering algorithm.	1. No assumption on the expected number of clusters 2. Can handle noisy data and outliers. 3. No assumptions on the shapes and sizes of the clusters. 4. Can identify clusters with different densities	1. Mapping of unseen objects in HDBSCAN is not straightforward. 2. Can be computationally expensive
Agglomerative Hierarchical Clustering	Uses hierarchical clustering to determine the distance between samples based on the metric, and pairs are merged into clusters using the linkage type.	1. There is no need to specify the number of clusters. 2. With the right linkage, it can be used for the detection of outliers. 3. Interpretable results using dendrograms.	1. Specifying metric and linkages types requires good understanding of the statistical properties of the data 2. Not straightforward to optimize 3. Can be computationally expensive for large datasets
OPTICS	Family of the density-based algorithms where it finds core sample of high density and expands clusters from them. It operates with a core distance (e) and reachability distance.	No assumption on the expected number of clusters. 2. Can handle noisy data and outliers. 3. No assumptions on the shapes and sizes of the clusters. 4. Can identify clusters with different densities. 5 Not required to define fixed radius as in DBSCAN.	1. It only produces a cluster ordering. 2. Does not work well in case of very high dimensional data. 3. Slower than DBSCAN.

Dimensionality Reduction Techniques

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
PCA	Principal Component Analysis (PCA) is a feature extraction approach that uses a linear function to reduce dimensionality in datasets by minimizing information loss.	1. Explainable Interpretable results. 2 New unseen datapoints can be mapped into the existing PCA space 3. Con be used as dimensionality reduction technique as preliminary step to other machine learning tasks 4 Helps reduce overfitting 5. Helps remove correlated features	1. Sensitive to outliers 2. Requires data standardization
t-SNE	t-distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction method that converts similarities between data points to joint probabilities using the Student t-distribution in the low-dimensional space	1. Helps preserve the relationships seen in high dimensionality 2. Easy to visualise the structure of high dimensional data in or 3 dimensions 3. Very effective for visualizing clusters or groups of data points and their relative proximities	1. The cost function is not convex: different initializations can get different results. 2. Computationally intensive for large datasets. 3. Default parameters do not always achieve the best results
UMAP	Uniform Manifold Approximation and Projection (UMAP) constructs a high-dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.	1. It can be used as general-purpose dimension reduction technique as o preliminary step to other machine learning tasks. 2. Can be very effective for visualizing clusters or groups of data points and their relative proximities. 3 Able to handle high dimensional sparse datasets	1. Default parameters do not always achieve the best results
ICA	Independent Component Analysis (ICA) is a linear dimensionality reduction method that aims to separate a multivariate signal into additive subcomponents under the assumption that independent components are non-gaussian. Where PCA "compresses" the data, ICA "separates" the information.	1. Can separate multivariate signals into its subcomponents 2 Clear aim of the method: only applicable if there are multiple independent generators of information to uncover. 3. Can extract hidden factors in the data by transforming a set of variables to new set that maximally independent.	1. Without any prior knowledge, determination of the number of independent components or sources can be difficult. 2. PCA is often required as a pre-processing step.

Association Rules

ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
Apriori algorithm	The Apriori algorithm uses the join and prune step iteratively to identify the most frequent itemset in the given dataset. Prior knowledge (apriori) of frequent itemset properties is used in the process.	1. Explainable & interpretable results. 2. Exhaustive approach based on the confidence and support.	1. Requires defining the expected number of clusters or mixture components in advance 2. The covariance type needs to be defined for the mixture of component
FP-growth algorithm	Frequent Pattern growth (FP-growth) is an improvement on the Apriori algorithm for finding frequent itemsets. It generates a conditional FP-Tree for every item in the data.	1. Explainable & interpretable results. 2. Smaller memory footprint than the Apriori algorithm	1. More complex algorithm to build than Apriori 2. Can result in many (incremental) overlapping/trivial itemsets
FP-Max Algorithm	A variant of Frequent pattern growth that is focused on finding maximal itemsets.	1. Explainable & Interpretable results. 2. Smaller memory footprint than the Apriori and FP-growth algorithms	1. More complex algorithm to build than Apriori