Chap 4. Dimensionality Reduction Techniques

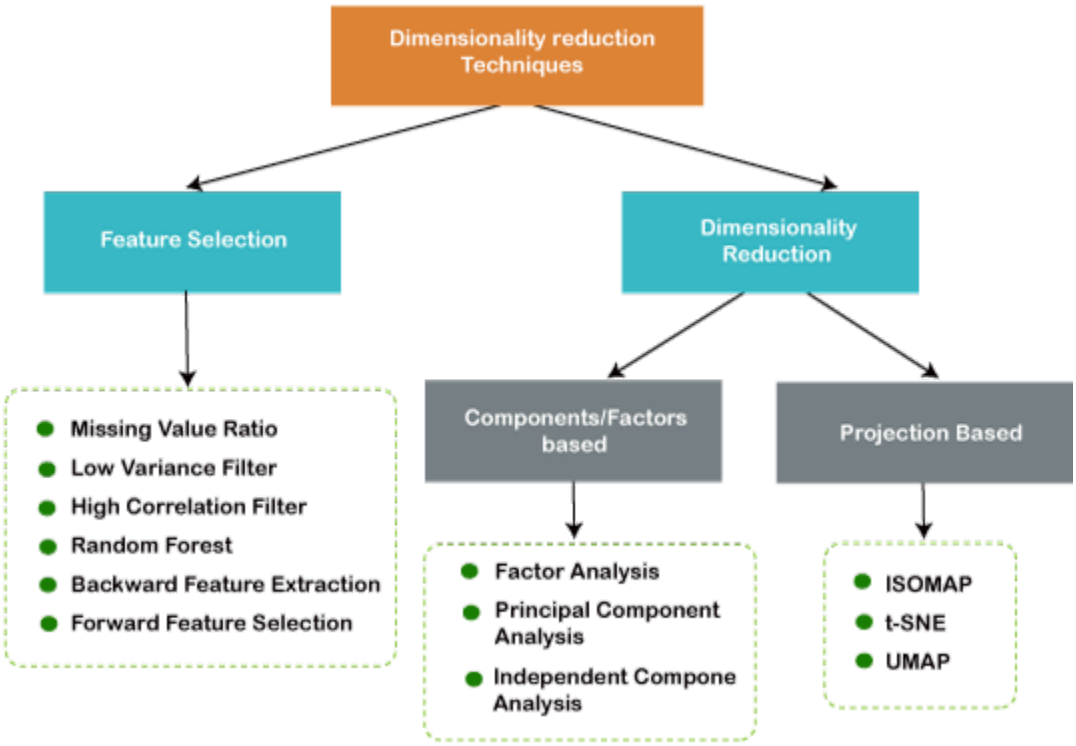
What is Dimensionality Reduction?

The number of input features, variables, or columns present in a given dataset is known as **dimensionality**, and the process to reduce these features is called **dimensionality reduction.**

A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."*** These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.



The Curse of Dimensionality

Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance. Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction. To address the curse of dimensionality, Feature engineering techniques are used which include feature selection and feature extraction. Dimensionality reduction is a type of feature extraction technique that aims to reduce the number of input features while retaining as much of the original information as possible.

Benefits of applying Dimensionality Reduction

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- **I**t removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of dimensionality Reduction

There are also some disadvantages of applying the dimensionality reduction, which are given below:

- **Loss of Information** – Important data may be discarded, which can reduce model accuracy or predictive power.
- **Reduced Interpretability** – New features (like principal components) are often abstract and hard to understand.
- **Computational Cost** – Techniques like t-SNE and UMAP can be slow and resource-intensive on large datasets.
- **Irreversibility** – Once reduced, it's difficult to reconstruct the original data accurately.

Approaches of Dimension Reduction

There are two ways to apply the dimension reduction technique, which are given below:

1. Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset. (Discussed in detail later)

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation Coefficient**
- **Chi-Square Test**
- **ANOVA**
- **Information Gain** , **etc.**

---

2. Wrappers Methods

In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection
- Bi-directional Elimination

---

3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

---

2. Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

1. Principal Component Analysis
2. Linear Discriminant Analysis
3. Kernel PCA
4. Quadratic Discriminant Analysis

Common techniques of Dimensionality Reduction

1. **Principal Component Analysis**
2. **Backward Elimination**
3. **Forward Selection**
4. **Score comparison**
5. **Missing Value Ratio**
6. **Low Variance Filter**
7. **High Correlation Filter**
8. **Random Forest**
9. **Factor Analysis**
10. **Auto-Encoder**

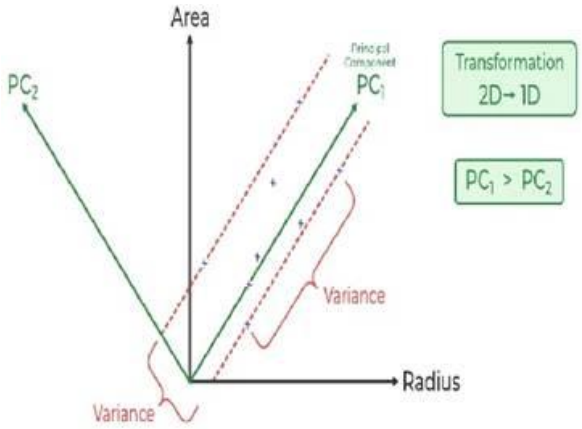
Principal Component Analysis(PCA)

Need:

As the number of features or dimensions in a dataset increases, the amount of data required to obtain a statistically significant result increases exponentially. This can lead to issues such as overfitting, increased computation time, and reduced accuracy of machine learning models; this is known as the curse of dimensionality problems that arise while working with high-dimensional data. It becomes expensive to perform tasks such as clustering or classification because the algorithms need to process a much larger feature space, which increases computation time and complexity. Additionally, some machine learning algorithms can be sensitive to the number of dimensions, requiring more data to achieve the same level of accuracy as lower-dimensional data.

What is Principal Component Analysis(PCA)?

Principal Component Analysis(PCA) technique was introduced by the mathematician Karl Pearson in 1901. It works on the condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum. Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, Principal Component Analysis (PCA) is an unsupervised learning algorithm technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit. The main goal of Principal Component Analysis (PCA) is to reduce the dimensionality of a dataset while preserving the most important patterns or relationships between the variables without any prior knowledge of the target variables. Principal Component Analysis (PCA) is used to reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables, retaining most of the sample's information, and useful for the regression and classification of data.



Applications of Principal Component Analysis

- **Dimensionality Reduction**
  PCA reduces the number of variables in datasets while retaining most of the original variance, making data easier to visualize and process.

- **Data Visualization**
  High-dimensional data can be projected onto 2D or 3D space using PCA for easy

visualization and pattern recognition.

- **Noise Reduction**
  By keeping only the principal components with the most variance, PCA can eliminate noise and improve the quality of data.

- **Feature Extraction**
  PCA helps in identifying and creating new features (principal components) that are combinations of the original features, which can improve machine learning performance.

- **Image Compression**
  PCA is used in compressing images by reducing the number of correlated pixels, thus saving storage while retaining important visual features.

Mathematical Approach to PCA

The main guiding principle for Principal Component Analysis is FEATURE EXTRACTION i.e. "Features of a data set should be less as well as the similarity between each other is very less." In PCA, a new set of features are extracted from the original features which are quite dissimilar in nature. So, an n-dimensional feature space gets transformed into an m-dimensional feature space., where the dimensions are orthogonal to each other.

Concept of Orthogonality: Vector Space is a set of vectors. They can be represented as a linear combination of the smaller set of vectors called BASIS VECTORS. So any vector 'v' in a vector space can be represented as:

$$v = \sum_{i=1}^{n} a_i u_i$$

where *a* represents 'n' scalars and u represents the basis vectors. Basis vectors are orthogonal to each other. Orthogonality of vectors can be thought of an extension of the vectors being perpendicular in a 2-D vector space. So our feature vector (data-set) can be transformed into a set of principal components (just like the basis vectors).

Objectives of PCA:

The new features are distinct i.e. the covariance between the new features (in case of PCA, they are the principal components) is 0. The principal components are generated in order of the variability in the data that it captures. Hence, the first principal component should capture the

maximum variability, the second one should capture the next highest variability etc. The sum of the variance of the new features / the principal components should be equal to the sum of the variance of the original features.

Working of PCA:

PCA works on a process called Eigenvalue Decomposition of a covariance matrix of a data set. The steps are as follows:

1. First, calculate the covariance matrix of a data set.
2. Then, calculate the eigenvectors of the covariance matrix.
3. The eigenvector having the highest eigenvalue represents the direction in which there is the highest variance. So this will help in identifying the first principal component.
4. The eigenvector having the next highest eigenvalue represents the direction in which data has the highest remaining variance and is also orthogonal to the first direction. So, this helps in identifying the second principal component.
5. Like this, identify the top 'k' eigenvectors having top 'k' eigenvalues to get the 'k' principal components.

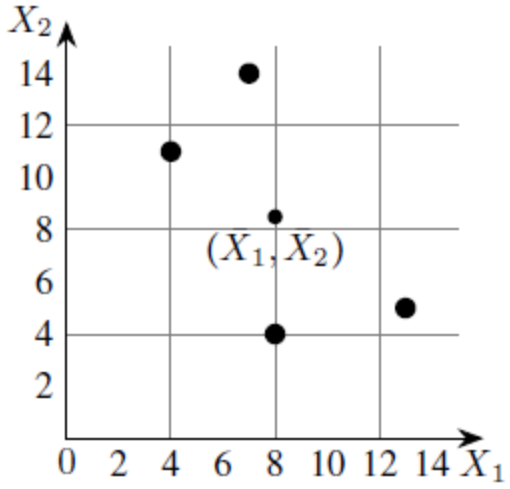
Numerical for PCA :

Consider the following dataset

Given the data in Table, reduce the dimension from 2 to 1 using the Principal Component Analysis (PCA) algorithm.

Feature	Example 1	Example 2	Example 3	Example 4
X_1	4	8	13	7
X_2	11	4	5	14

Step 1: Calculate Mean
The figure shows the scatter plot of the given data points.



Calculate the mean of X_1 and X_2 as shown below.

$$\bar{X}_1 = \tfrac{1}{4}(4+8+13+7) = 8,$$
$$\bar{X}_2 = \tfrac{1}{4}(11+4+5+14) = 8.5.$$

Step 2: Calculation of the covariance matrix.
The covariances are calculated as follows:

$$\begin{aligned}
\mathrm{Cov}(X_1, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)^2 \\
&= \tfrac{1}{3}\left((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right) \\
&= 14 \\
\mathrm{Cov}(X_1, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)(X_{2k} - \bar{X}_2) \\
&= \tfrac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) \\
&\quad + (13-8)(5-8.5) + (7-8)(14-8.5) \\
&= -11 \\
\mathrm{Cov}(X_2, X_1) &= \mathrm{Cov}(X_1, X_2) \\
&= -11 \\
\mathrm{Cov}(X_2, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{2k} - \bar{X}_2)^2 \\
&= \tfrac{1}{3}\left((11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2\right) \\
&= 23
\end{aligned}$$

The covariance matrix is,

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix}$$
$$= \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 3: Eigenvalues of the covariance matrix
The characteristic equation of the covariance matrix is,

$$0 = \det(S - \lambda I)$$
$$= \begin{vmatrix} 14 - \lambda & -11 \\ -11 & 23 - \lambda \end{vmatrix}$$
$$= (14 - \lambda)(23 - \lambda) - (-11) \times (-11)$$
$$= \lambda^2 - 37\lambda + 201$$

Solving the characteristic equation we get,

$$\lambda = \tfrac{1}{2}(37 \pm \sqrt{565})$$
$$= 30.3849, 6.6151$$
$$= \lambda_1, \lambda_2 \quad \text{(say)}$$

Step 4: Computation of the eigenvectors
To find the first principal components, we need only compute the eigenvector corresponding to the largest eigenvalue. In the present example, the largest eigenvalue is λ_1 and so we compute the eigenvector corresponding to λ_1.
The eigenvector corresponding to $\lambda = \lambda_1$ is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

satisfying the following equation:

$$\begin{bmatrix}0\\0\end{bmatrix} = (S-\lambda_1 I)X$$
$$= \begin{bmatrix}14-\lambda_1 & -11\\ -11 & 23-\lambda_1\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$
$$= \begin{bmatrix}(14-\lambda_1)u_1 - 11u_2\\ -11u_1 + (23-\lambda_1)u_2\end{bmatrix}$$

This is equivalent to the following two equations:

$$(14-\lambda_1)u_1 - 11u_2 = 0$$
$$-11u_1 + (23-\lambda_1)u_2 = 0$$

Using the theory of systems of linear equations, we note that these equations are not independent and solutions are given by,

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda_1} = t,$$

that is,

$$u_1 = 11t, \quad u_2 = (14-\lambda_1)t,$$

where t is any real number.
Taking t = 1, we get an eigenvector corresponding to λ_1 as

$$U_1 = \begin{bmatrix}11\\14-\lambda_1\end{bmatrix}.$$

To find a unit eigenvector, we compute the length of X_1 which is given by,

$$\|U_1\| = \sqrt{11^2 + (14-\lambda_1)^2}$$
$$= \sqrt{11^2 + (14-30.3849)^2}$$
$$= 19.7348$$

Therefore, a unit eigenvector corresponding to λ_1 is

$$e_1 = \begin{bmatrix} 11/\|U_1\| \\ (14-\lambda_1)/\|U_1\| \end{bmatrix}$$
$$= \begin{bmatrix} 11/19.7348 \\ (14-30.3849)/19.7348 \end{bmatrix}$$
$$= \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

By carrying out similar computations, the unit eigenvector e_2 corresponding to the eigenvalue $\lambda = \lambda_2$ can be shown to be,

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

Step 5: Computation of first principal components

let,

$$\begin{bmatrix} X_{1k} \\ X_{2k} \end{bmatrix}$$

be the k^{th} sample in the above Table (dataset). The first principal component of this example is given by (here "T" denotes the transpose of the matrix)

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix}$$
$$= 0.5574(X_{1k} - \bar{X}_1) - 0.8303(X_{2k} - \bar{X}_2).$$

For example, the first principal component corresponding to the first example

$$\begin{bmatrix} X_{11} \\ X_{21} \end{bmatrix} = \begin{bmatrix} 4 \\ 11 \end{bmatrix}$$

is calculated as follows:

$$\begin{bmatrix}0.5574 & -0.8303\end{bmatrix}\begin{bmatrix}X_{11}-\bar{X}_1\\X_{21}-\bar{X}_2\end{bmatrix} = 0.5574(X_{11}-\bar{X}_1) - 0.8303(X_{21}-\bar{X}_2)$$
$$= 0.5574(4-8) - 0.8303(11-8,5)$$
$$= -4.30535$$

The results of the calculations are summarized in the below Table.

X_1	**4**	**8**	**13**	**7**
X_2	**11**	**4**	**5**	**14**
First Principle Components	**-4.3052**	**3.7361**	**5.6928**	**-5.1238**

Step 6: Geometrical meaning of first principal components

First, we shift the origin to the "center"

$$(\bar{X}_1, \bar{X}_2)$$

and then change the directions of coordinate axes to the directions of the eigenvectors e_1 and e_2.

Kernel Principal Component Analysis (KPCA)

The PCA technique is linear ,i.e., it is limited to datasets that are separable linearly and performs admirably. However, the outcome may not be the best dimensionality reduction if we apply it to nonlinear datasets. With kernel PCA, a dataset is projected onto a linearly separated kernel function. The concept bears similarities to those of Support Vector Machines. When working with data that shows nonlinear structures or patterns, kernel PCA is especially helpful. It is crucial to remember that the kernel and its settings have a big influence on the outcome and that fine-tuning these settings is required to get the best performance.

- One method for reducing nonlinear dimensionality in machine learning is Kernel Principal Component Analysis (KPCA). It is a development of the traditional Principal Component Analysis (PCA) methodology, which captures more intricate and nonlinear interactions between the data points, KPCA first applies a nonlinear mapping function to the data before using PCA.
- There are several kernel techniques, including **Gaussian**, **polynomial**, and **linear**.
- In KPCA, the input data is mapped, which makes it easier for linear techniques like PCA to capture the nonlinear correlations between the data points. Next, the altered data's principal components are computed, which can be applied to data visualization, clustering, and classification tasks.
- The ability of KPCA to accommodate nonlinear connections between the input features gives it an edge over regular PCA, especially for jobs like voice or image recognition. KPCA can reduce the dimensionality of the data while maintaining the most crucial information, making it capable of handling high-dimensional datasets with numerous characteristics.
- Using a kernel characteristic in kernel PCA, the dataset is projected directly into a higher-dimensional space where it is linearly separable. Finally, we followed the kernel PCA to a nonlinear dataset using scikit-learn.
- The kernel trick enables implicit computation of the high-dimensional representation of the data without explicit computation of the transformation and is also commonly employed in support vector machines (SVMs). Polynomial, sigmoid, and radial basis function (RBF) kernels are examples of frequently used kernel functions.

Using the idea of kernel functions in machine learning by transforming it into a high-dimensional feature space. By identifying the primary components of the data's covariance matrix, classical PCA converts the data into a lower-dimensional space. Using a nonlinear mapping function known as a kernel function, the data is converted into a high-dimensional feature space for kernel PCA. The principal components are then located in this high-dimensional feature space.

In simpler terms, Kernel PCA:

---

1. Maps non-linear data to a higher dimension.
2. Uses a kernel function to define the similarity between data points.
3. Performs PCA in the new, higher-dimensional space.
4. Identifies principal components that reveal non-linear relationships.

The following are the general steps in Kernel PCA:

---

1. **Select a kernel function:** Choose an appropriate kernel function according to the properties of the data. The challenge at hand and the data's underlying structure determine which kernel should be used.
2. **Compute the kernel matrix:** Using the selected kernel function, determine the pairwise similarity (or distance) between data points. As a result, the kernel matrix-a symmetric positive semi-definite matrix-is produced.

3. **Choose the main elements:** The greatest eigenvalues' top k eigenvectors, or primary components, are selected to create the reduced-dimensional representation of the data.

Advantages of Kernel PCA include:

- **Non-linearity:** In contrast to conventional linear PCA, kernel PCA can identify nonlinear patterns in the data.
- **Robustness:** Kernel PCA may be more resistant to outliers and noise in the data since it considers the entire structure of the information rather than just the close distances between record points.
- **Versatility:** Various kernel functions can be applied in kernel PCA to accommodate various data kinds and goals.
- In contrast to traditional linear PCA, kernel PCA may control nonlinear relationships between the input features, enabling more precise feature extraction and dimensionality reduction.
- It can make statistics easier to see and interpret by reducing the dimensionality of the data while maintaining the most important information in excessively dimensional datasets.
- Kernel PCA can be used for a variety of tasks, including categorization, grouping, and information visualisation.

Disadvantages of kernel PCA

- **Complexity:** Because kernel PCA needs the computation of kernel matrix for every pair of data points, eigenvectors and eigenvalues, it can be computationally costly, particularly for big datasets.
- **Model choice:** Selecting the appropriate kernel function and component count can be difficult and may call for specialized expertise or trial and error.
- **No Inverse Transform :** Unlike regular PCA, kernel PCA doesn't have a straightforward inverse transformation. This means you can't easily map transformed data back to the original space, which limits interpretability and reconstruction.
- **Less Interpretable :**The new dimensions (components) in kernel PCA are not easy to interpret because they are **not simple linear combinations** of the original features.
- **Sensitive to Noise :**Kernel PCA can sometimes overfit to noise in the data, especially when using complex kernels like RBF with small bandwidth.

Difference between PCA and KPCA:

Aspect	PCA (Principal Component Analysis)	KPCA (Kernel Principal Component Analysis)
Type of Method	Linear dimensionality reduction	Nonlinear dimensionality reduction
Feature Transformation	Projects data onto linear components (directions of max variance)	Projects data into a high-dimensional space using a kernel
Handles Nonlinearity	Cannot capture nonlinear patterns	Can capture complex nonlinear structures
Kernel Function Used	Not used	Uses kernel functions (e.g., RBF, polynomial, sigmoid)
Computational Complexity	More efficient; scales well with large datasets	Less efficient; involves computing large kernel matrices

Low Rank Approximation (LRA)

In machine learning and data analysis, **low rank approximation** refers to representing a matrix (or a dataset) with a **smaller number of components** or features, while still capturing most of the important information. This is done by approximating the matrix with another matrix that has a **lower rank** than the original one.

Low-rank approximations are a commonly used technique in data analysis and machine learning for reducing the dimensionality of high-dimensional data. Given a matrix A, a low-rank approximation involves finding a matrix B that is of lower rank than A but approximates A as closely as possible. This is typically done by finding the truncated singular value decomposition (SVD) of A, which provides a low-rank approximation of A by keeping only the largest singular values and corresponding singular vectors.

Low-rank approximations can be computed using various numerical algorithms, including the randomized SVD and the alternating least squares (ALS) algorithm. These methods are particularly useful for handling large-scale datasets that are difficult to store or process in their original high-dimensional form.

Key Concepts:

- **Rank** of a matrix refers to the number of linearly independent rows or columns. A **low rank** matrix has fewer linearly independent rows or columns.

- The goal of **low rank approximation** is to find a matrix with **lower rank** that is **close** to the original matrix in some sense, while simplifying the data representation.

How Does It Work?

1. **Matrix Decomposition**: You can use methods like **Singular Value Decomposition (SVD)** or **Principal Component Analysis (PCA)** to break the matrix into components.

2. **Truncate**: You then keep only the **largest singular values** or principal components (the most important features), discarding the smaller, less significant ones.

3. **Reconstruct**: The resulting matrix is the low-rank approximation of the original one.

Based on the above two discussed method, we define LRA as:

For a matrix A of shape m x n and rank (A) << min {m, n}, the low-rank approximation of A is to find another matrix B such that rank (B) ≤ rank (A). Intuitively, we tend to see how linearly similar matrix B is to the input matrix A. Mathematically, LRA is a minimization problem, in which we measure the fit between a given matrix (the data) and an approximating matrix (the optimization variable).

Motivation for LRA

Let us assume 3 matrices X, Y, Z of dimensions (50 x 40), (50 x 10), (40 x 10) respectively where rank(X) = 10. Therefore, as per matrix factorization, the three matrices are of the form A = BCT. We observe that:

$$A_{mxn} = BmxrCTrxn$$

No of elements/pixels in mat(A) = (50 x 40) = 2000
while
No of elements/pixels in mat(BC^T) = (50 x 10) + (10 x 40) = 900.

Thus, by using low-rank approximation, we can reduce the number of pixels of input data by a significant amount. (1100 fewer pixels than input image in the above case)

Application of LRA

This method is commonly used in Image Processing tasks where the input image (data) is very large and needs to be compressed before any processing task. Observe the two images of the same person given below.

After LRA image vs Input image

After applying LRA, we are able to obtain an image that is very similar to the original image by discarding some unimportant pixels, hence reducing the size of the image as well. After doing this, it becomes much easier to handle such large datasets and perform a variety of pre-processing tasks on them.

Eigenvalue and eigenvector computation is a fundamental problem in linear algebra with numerous applications in various fields, including machine learning, signal processing, and quantum mechanics, among others. Given a square matrix A, an eigenvector v is a non-zero vector that satisfies the equation $Av = \lambda v$, where λ is a scalar known as the eigenvalue corresponding to v.

To compute eigenvectors, we typically use numerical algorithms that involve iterative methods, such as the power method or the inverse power method. These methods can efficiently compute the eigenvectors associated with the largest or smallest eigenvalues of a matrix, respectively. Additionally, there are also methods that can compute all the eigenvectors of a matrix simultaneously, such as the QR algorithm or the Jacobi method.

Advantages of Eigenvector Computation:

- Eigenvectors provide a concise representation of a matrix, as they capture the most important features of the matrix.
- Eigenvectors can be used to solve linear systems of equations, which arise in many practical applications.
- Eigenvectors can be used to diagonalize a matrix, which simplifies many computations and allows us to easily compute powers and exponentials of the matrix.

Disadvantages of Eigenvector Computation:

- Eigenvector computation can be computationally expensive, particularly for large matrices.
- Eigenvectors may not exist for all matrices, or they may be complex numbers.
- Eigenvectors may not be unique, as a matrix may have multiple eigenvectors corresponding to the same eigenvalue.

Advantages of Low-Rank Approximations:

- Low-rank approximations can significantly reduce the dimensionality of high-dimensional data, making it easier to store and process.
- Low-rank approximations can reduce noise and remove irrelevant features, improving the accuracy of data analysis and machine learning models.
- Low-rank approximations can be computed efficiently using numerical algorithms, particularly for large datasets.

Disadvantages of Low-Rank Approximations:

- Low-rank approximations may not capture all the information in the original high-dimensional data, leading to some loss of information.
- Choosing an appropriate rank for the low-rank approximation can be challenging, and different choices can lead to different results.
- Low-rank approximations may not work well for all types of data, particularly if the data is highly irregular or has a complex structure.

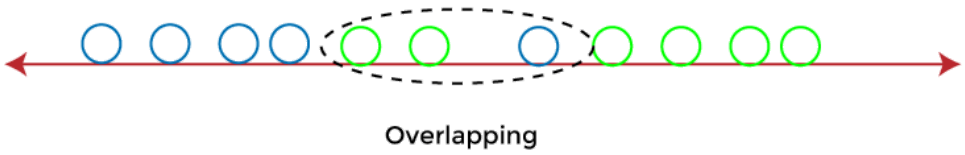
Linear Discriminant Analysis (LDA) in Machine Learning

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA). This can be used to project the features of higher dimensional space into lower-dimensional space in order to reduce resources and dimensional costs.

What is Linear Discriminant Analysis (LDA)?

Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems. ***Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning***. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification.

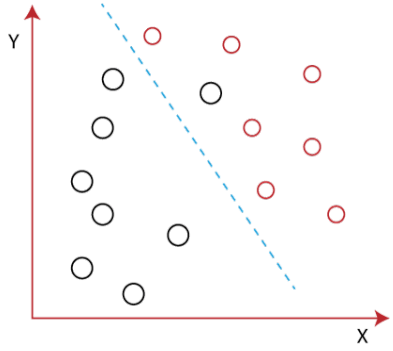
Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.



To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

Example:

Let's assume we have to classify two different classes having two sets of data points in a 2-dimensional plane as shown below image:



However, it is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis; we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

Why LDA?

- Logistic Regression is one of the most popular classification algorithms that perform well for binary classification but falls short in the case of multiple classification problems with well-separated classes. At the same time, LDA handles these quite efficiently.
- LDA can also be used in data pre-processing to reduce the number of features, just as PCA, which reduces the computing cost significantly.
- LDA is also used in face detection algorithms. In Fisherfaces, LDA is used to extract useful data from different faces. Coupled with eigenfaces, it produces effective results.

Drawbacks of Linear Discriminant Analysis (LDA)

Although, LDA is specifically used to solve supervised classification problems for two or more classes which are not possible using logistic regression in machine learning. But LDA also fails in some cases where the Mean of the distributions is shared. In this case, LDA fails to create a new axis that makes both the classes linearly separable. To overcome such problems, we use **non-linear Discriminant analysis** in machine learning.

Extension to Linear Discriminant Analysis (LDA)

Linear Discriminant analysis is one of the most simple and effective methods to solve classification problems in machine learning. It has so many extensions and variations as follows:

1. **Quadratic Discriminant Analysis (QDA):** For multiple input variables, each class deploys its own estimate of variance.
2. **Flexible Discriminant Analysis (FDA):** it is used when there are non-linear groups of inputs are used, such as splines.
3. **Flexible Discriminant Analysis (FDA):** This uses regularization in the estimate of the variance (actually covariance) and hence moderates the influence of different variables on LDA.

Real-world Applications of LDA

Some of the common real-world applications of Linear discriminant Analysis are given below:

- **Face Recognition** : Face recognition is the popular application of computer vision, where each face is represented as the combination of a number of pixel values. In this case, LDA is used to minimize the number of features to a manageable number before going through the classification process. It generates a new template in which each dimension consists of a linear combination of pixel values. If a linear combination is generated using Fisher's linear discriminant, then it is called Fisher's face.

- **Medical** :In the medical field, LDA has a great application in classifying the patient's disease on the basis of various parameters of patient health and the medical treatment which is going on. On such parameters, it classifies disease as mild, moderate, or severe. This classification helps the doctors in either increasing or decreasing the pace of the treatment.
- **Customer Identification** : In customer identification, LDA is currently being applied. It means with the help of LDA; we can easily identify and select the features that can specify the group of customers who are likely to purchase a specific product in a shopping mall. This can be helpful when we want to identify a group of customers who mostly purchase a product in a shopping mall.
- **For Predictions** :LDA can also be used for making predictions and so in decision making. For example, "will you buy this product" will give a predicted result of either one or two possible classes as a buying or not.
- **In Learning** : Nowadays, robots are being trained for learning and talking to simulate human work, and it can also be considered a classification problem

Difference between Linear Discriminant Analysis and PCA

Feature	PCA (Principal Component Analysis)	LDA (Linear Discriminant Analysis)
1. Purpose	PCA is used to reduce the dimensionality of data by finding new axes (principal components) that **capture the highest variance** in the dataset, irrespective of any class labels. This helps retain most of the original data's information.	LDA reduces dimensionality by finding new axes that **maximize the distance between different classes** while minimizing the variance within each class, improving class separability for classification tasks.
2. Supervision	PCA is an **unsupervised** method. It does not consider the class labels of the data. It only looks at the features and their correlations to find directions of maximum variance.	LDA is a **supervised** method. It uses class labels to understand which features are most effective at distinguishing between classes, and it optimizes the projection accordingly.
3. Feature Reduction Limit	PCA can produce as many components as there are original features (e.g., from 100D to 10D or even 1D), depending on how much variance you want to retain.	LDA can reduce data to a maximum of **(C – 1)** dimensions, where **C** is the number of distinct classes in the data. For example, if there are 3 classes, LDA can give at most 2 components.

4. Projection Focus	PCA finds directions in the data space where the **spread (variance) is greatest**, regardless of what classes the data points belong to. The goal is to represent the data efficiently.	LDA finds the axes that **maximize the distance between class means** and **minimize the variance within each class**, making it ideal for separating different categories.
5. Use Cases	PCA is commonly used for **data compression, noise filtering, feature visualization, and speeding up machine learning models** on large datasets without needing to consider output labels.	LDA is ideal for **classification tasks**, such as face recognition, medical diagnosis, and document classification, where distinguishing between categories is important.

Feature Selection Techniques in Machine Learning

What is Feature Selection?

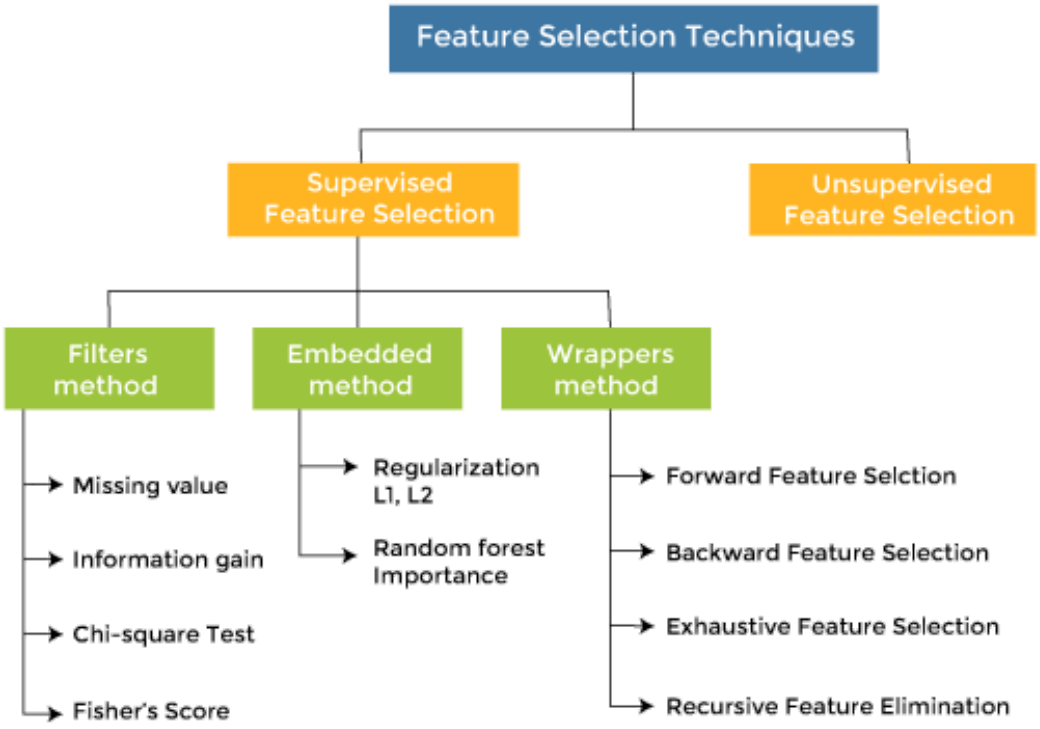
A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

So, we can define feature Selection as, "***It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building***." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

Feature Selection Techniques

There are mainly two types of Feature Selection techniques, which are:

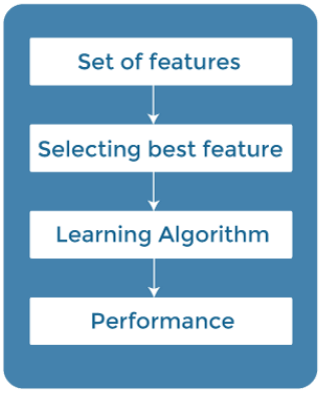
- **Supervised Feature Selection technique**
  Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection technique**
  Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.



There are mainly three techniques under supervised feature Selection:

1. Filter Methods

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step. The filter method filters out the irrelevant features and redundant columns from the model by using different metrics through ranking. The advantage of using filter methods is that it needs low computational time and does not overfit the data.



Some common techniques of Filter methods are as follows:

Information Gain: Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

Chi-square Test: Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

Fisher's Score:

Fisher's score is one of the popular supervised techniques of feature selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

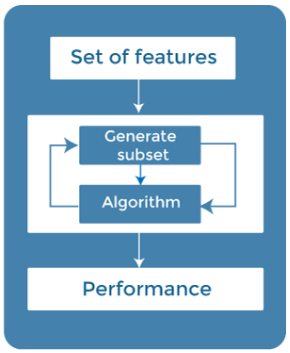
Missing Value Ratio:

The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable having more than the threshold value can be dropped.

$$\text{Missing Value Ratio} = \frac{Number\ of\ Missing\ values * 100}{Total\ number\ of\ observations}$$

2. Wrapper Methods

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



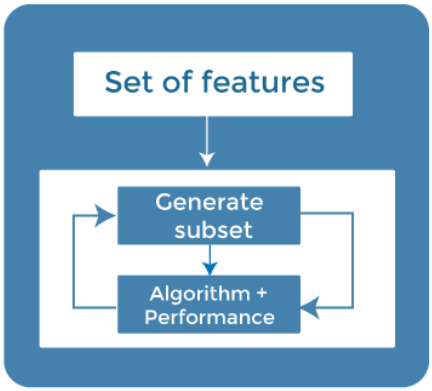
On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

Some techniques of wrapper methods are:

- **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new variable/feature does not improve the performance of the model.
- **Backward elimination** - Backward elimination is also an iterative approach, but it is the opposite of forward selection. This technique begins the process by considering all the features and removes the least significant feature. This elimination process continues until removing the features does not improve the performance of the model.
- **Exhaustive Feature Selection-** Exhaustive feature selection is one of the best feature selection methods, which evaluates each feature set as brute-force. It means this method tries & makes each possible combination of features and returns the best performing feature set.
- **Recursive Feature Elimination-**
  Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features. Now, an estimator is trained with each set of features, and the importance of each feature is determined using *coef_attribute* or through a *feature_importances_attribute.*

3. Embedded Methods

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.



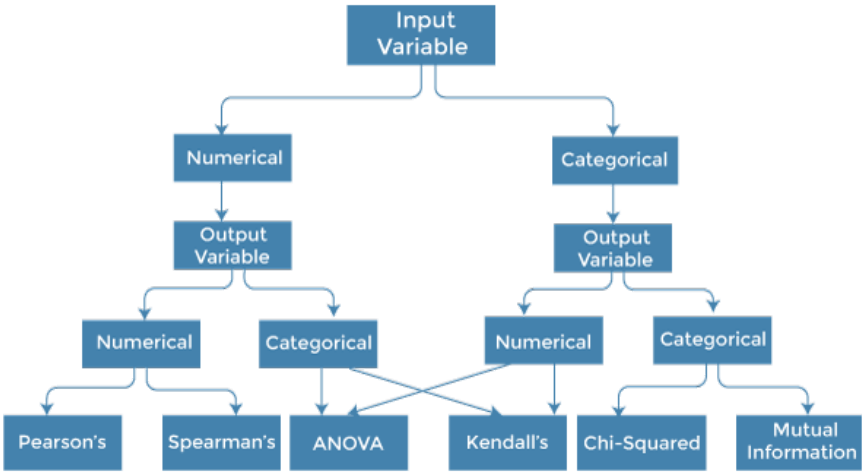
These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:

- **Regularization-** Regularization adds a penalty term to different parameters of the machine learning model for avoiding overfitting in the model. This penalty term is added to the coefficients; hence it shrinks some coefficients to zero. Those features with zero coefficients can be removed from the dataset. The types of regularization techniques are L1 Regularization (Lasso Regularization) or Elastic Nets (L1 and L2 regularization).

- **Random Forest Importance -** Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, which is a type of bagging algorithm that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the impurity (Gini impurity) over all the trees. Nodes are arranged as per the impurity values, and thus it allows pruning of trees below a specific node. The remaining nodes create a subset of the most important features.

How to choose a Feature Selection Method?

For machine learning engineers, it is very important to understand which feature selection method will work properly for their model. The more we know the data types of variables, the easier it is to choose the appropriate statistical measure for feature selection.



To know this, we need to first identify the type of input and output variables. In machine learning, variables are of mainly two types:

- **Numerical Variables:** Variable with continuous values such as integer, float
- **Categorical Variables:** Variables with categorical values such as Boolean, ordinal, nominals.

Below are some univariate statistical measures, which can be used for filter-based feature selection:

1. Numerical Input, Numerical Output:

Numerical Input variables are used for predictive regression modelling. The common method to be used for such a case is the Correlation coefficient.

- Pearson's correlation coefficient (For linear Correlation).
- Spearman's rank coefficient (for non-linear correlation).

---

2. Numerical Input, Categorical Output:

Numerical Input with categorical output is the case for classification predictive modelling problem**s.** In this case, also, correlation-based techniques should be used, but with categorical output.

- **ANOVA correlation coefficient (linear).**
- **Kendall's rank coefficient (nonlinear).**

3. Categorical Input, Numerical Output:

This is the case of regression predictive modelling with categorical input. It is a different example of a regression problem. We can use the same measures as discussed in the above case but in reverse order.

4. Categorical Input, Categorical Output:

This is a case of classification predictive modelling with categorical Input variables.

The commonly used technique for such a case is Chi-Squared Test. We can also use Information gain in this case.

We can summarise the above cases with appropriate measures in the below table:

Input Variable	Output Variable	Feature Selection technique
Numerical	Numerical	○ Pearson's correlation coefficient (For linear Correlation). ○ Spearman's rank coefficient (for non-linear correlation).
Numerical	Categorical	○ ANOVA correlation coefficient (linear). ○ Kendall's rank coefficient (nonlinear).
Categorical	Numerical	○ Kendall's rank coefficient (linear). ○ ANOVA correlation coefficient (nonlinear).
Categorical	Categorical	○ Chi-Squared test (contingency tables). ○ Mutual Information.

Feature Mapping

Feature mapping is a technique used in data analysis and machine learning to transform input data from a lower-dimensional space to a higher-dimensional space, where it can be more easily analyzed or classified. Feature mapping involves selecting or designing a set of functions that map the original data to a new set of features that better capture the underlying patterns in the data. The resulting feature space can then be used as input to a machine learning algorithm or other analysis technique. Feature mapping can be used in a wide range of applications, from natural language processing to computer vision, and is a powerful tool for transforming data into a format that can be analyzed more easily. However, there are also potential issues to consider, such as the curse of dimensionality, overfitting, and computational complexity.

Feature mapping, also known as feature engineering, is the process of transforming raw input data into a set of meaningful features that can be used by a machine learning algorithm. Feature mapping is an important step in machine learning, as the quality of the features can have a significant impact on the performance of the algorithm.

There are several techniques for feature mapping, including:

1. **Feature extraction**: This involves transforming the input data into a new set of features that capture the most important information. For example, in image processing, features such as edges, corners, and textures can be extracted from the image.
2. **Feature transformation:** This involves applying mathematical functions to the input data to transform it into a new set of features. For example, in text classification, the input text can be transformed into a bag-of-words representation, where each word in the text is represented by a count of its frequency.
3. **Feature selection**: This involves selecting a subset of the available features that are most relevant to the task at hand. This can be done using techniques such as mutual information, correlation, or regularization.
4. **Feature scaling**: This involves scaling the features to ensure that they have similar ranges and are on the same scale. This is important for some algorithms, such as those based on distance metrics.
5. **Feature mapping** can be a time-consuming and iterative process, as different feature combinations and transformations may need to be tried to find the best set of features for a given task. However, effective feature mapping can significantly improve the performance of a machine learning algorithm and enable it to make more accurate predictions.
6. **Feature engineering:** This is the process of creating new features from the existing ones in order to improve the performance of a machine learning algorithm. Feature engineering can involve a combination of feature extraction, transformation, and selection techniques. For example, in a fraud detection task, a new feature could be created by calculating the ratio of the transaction amount to the average transaction amount for that user.
7. **Dimensionality reduction:** This involves reducing the number of features in a dataset while retaining the most important information. This can be done using techniques such as principal component analysis (PCA) or t-distributed stochastic neighbor embedding

(t-SNE). Dimensionality reduction can help to reduce overfitting and improve the computational efficiency of a machine learning algorithm.

8. **Embeddings:** An embedding is a vector representation of a feature or object that captures its semantic meaning. For example, in natural language processing, word embeddings can be used to represent words as dense vectors in a high-dimensional space, where words with similar meanings are closer together. Embeddings can be learned using techniques such as word2vec or GloVe.
9. **Data augmentation:** This involves creating new examples by applying transformations to the existing data. For example, in image classification, data augmentation can involve rotating, flipping, or cropping the images to create new variations. Data augmentation can help to increase the size of the training set and improve the robustness of a machine learning model.
10. **Domain knowledge:** Expert knowledge about the domain in which the machine learning algorithm will be applied can be used to guide the feature mapping process.

Overfitting and Underfitting in Machine Learning

Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

The main goal of each machine learning model is **to generalize well**. Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output. Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

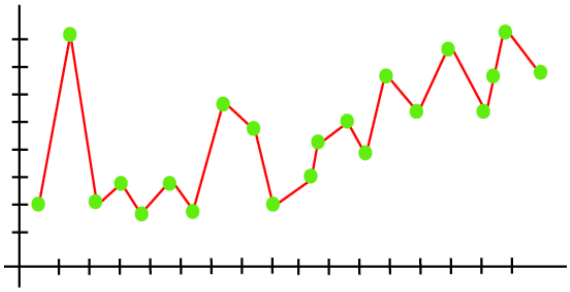
- **Signal:** It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.
- **Noise:** Noise is unnecessary and irrelevant data that reduces the performance of the model.
- **Bias:** Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.
- **Variance:** If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

Overfitting

Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has **low bias** and **high variance.** The

chances of occurrence of overfitting increase as we provide training to our model. It means the more we train our model, the more chances of occurring the overfitted model. Overfitting is the main problem that occurs in supervised learning.

Example: The concept of the overfitting can be understood by the below graph of the linear regression output:



As we can see from the above graph, the model tries to cover all the data points present in the scatter plot. It may look efficient, but in reality, it is not so. Because the goal of the regression model is to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors.

How to avoid the Overfitting in Model

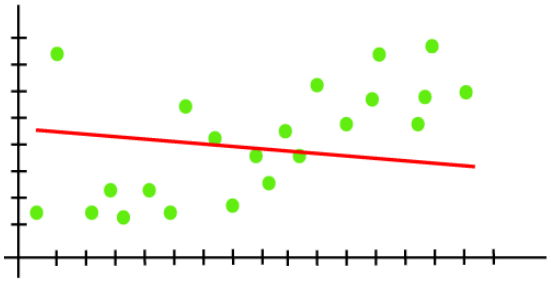
Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

- **Use Regularization** – Apply techniques like L1 (Lasso) or L2 (Ridge) to penalize model complexity.
- **Train with More Data** – Feeding more diverse data helps the model generalize better.
- **Use Cross-Validation** – Techniques like k-fold cross-validation help monitor model performance and detect overfitting.
- **Simplify the Model** – Reduce model complexity (e.g., fewer layers or parameters in neural networks).
- **Apply Dropout or Early Stopping** – In neural networks, dropout randomly disables neurons during training, and early stopping halts training when validation error starts increasing.

Underfitting

Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the feed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data.In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions. An underfitted model has high bias and low variance.

Example: We can understand the underfitting using below output of the linear regression model:



As we can see from the above diagram, the model is unable to capture the data points present in the plot.

How to avoid underfitting:

- **Use a more complex model** – Switch to models with higher capacity (e.g., from linear to decision trees or deep neural networks).
- **Train for more epochs** – Ensure the model has enough time to learn the patterns in the data.
- **Reduce regularization** – Lower regularization parameters (like alpha in Lasso/Ridge) to allow the model more flexibility.
- **Add relevant features** – Include additional informative features that help the model make better predictions.
- **Ensure proper data preprocessing** – Normalize, scale, and clean the data to help the model learn effectively.