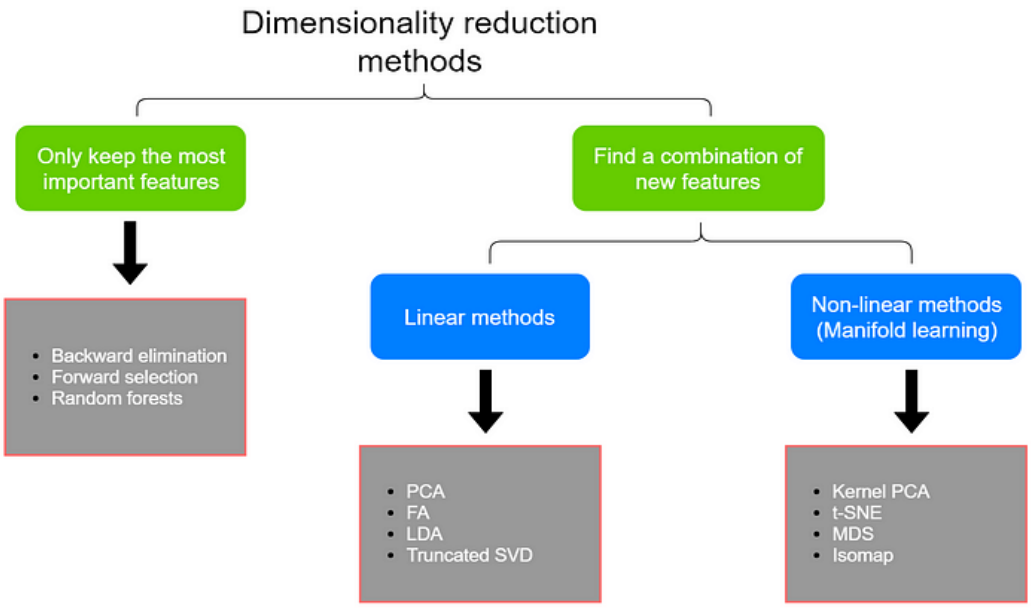
Dimensionality Reduction Techniques

Definition:

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of variables or features present in a dataset. It involves transforming high-dimensional data into a lower-dimensional space while preserving as much information as possible (extract some information as derived feature from original feature).

The main goal of dimensionality reduction is to simplify the data by reducing the number of features, which can be helpful in reducing noise, improving performance, precent the curse of dimensionality reducing overfitting, and making it easier to visualize and understand the data.



Only keep most important features:

Backward Elimination Method/ Backward Feature Selection:

Follow the below steps to understand and use the 'Backward Feature Elimination' technique:

- We first take all the n variables present in our dataset and train the model using them
- We then calculate the performance of the model
- Now, we compute the performance of the model after eliminating each variable (n times), i.e., we drop one variable every time and train the model on the remaining n-1 variables
- We identify the variable whose removal has produced the smallest (or no) change in the performance of the model, and then drop that variable
- Repeat this process until no variable can be dropped

Forward Elimination Method/ Forward Feature Selection:

This is the opposite process of the Backward Feature Elimination we saw above. Instead of eliminating features, we try to find the best features which improve the performance of the model. This technique works as follows:

- We start with a single feature. Essentially, we train the model n number of times using each feature separately
- The variable giving the best performance is selected as the starting variable
- Then we repeat this process and add one variable at a time. The variable that produces the highest increase in performance is retained
- We repeat this process until no significant improvement is seen in the model's performance

NOTE: Both Backward Feature Elimination and Forward Feature Selection are time consuming and computationally expensive**.** They are practically only used on datasets that have a small number of input variables. The techniques we have seen so far are generally used when we do not have a very large number of variables in our dataset.

Random Forest:

Random Forest is one of the most widely used algorithms for feature selection. It comes packaged with in-built feature importance so you don't need to program that separately. This helps us select a smaller subset of features.

We need to convert the data into numeric form by applying one hot encoding, as Random Forest (Scikit-Learn Implementation) takes only numeric inputs.

Linear Methods:

Principal Component Analysis (PCA):

PCA (Principal Component Analysis) is a statistical method used to reduce the dimensions of a large dataset by transforming it into a lower-dimensional space. It is used to identify patterns and relationships among variables, by identifying the most important features (or components) that explain the maximum variance in the dataset.

Here are the steps to calculate PCA:

1. **Standardize the data:** Calculate the mean and standard deviation of each feature in the dataset. Then, subtract the mean from each data point and divide the result by the standard deviation.

2. **Calculate the covariance matrix:** Create a matrix of all possible feature combinations and calculate the covariance of each pair of features.

3. **Calculate the eigenvectors and eigenvalues of the covariance matrix:** Eigenvectors are the directions of the maximum variance in the dataset, and eigenvalues represent the magnitude of the variance in the direction of the eigenvector.

4. **Choose the number of principal components:** Determine the number of components that explain most of the variance in the data. A common rule of thumb is to choose the number of components that explain at least 70-80% of the variance.

5. **Calculate the principal components:** Multiply the standardized data by the eigenvectors corresponding to the selected principal components. This will transform the data into the lower-dimensional space.

PCA can be implemented in several programming languages such as Python, R, MATLAB, etc. There are also many libraries available to perform PCA such as Scikit-learn, NumPy, SciPy, etc.

Advantages:

- **Reduces the number of features:** PCA helps in reducing the number of features without losing too much information. This is useful when dealing with datasets with a large number of features, which can make it difficult to analyse and visualize the data.
- **Helps in identifying important features:** PCA helps in identifying the most important features in the dataset. These features can be used to build more efficient models and improve accuracy.
- **Reduces overfitting:** PCA can help reduce overfitting by removing noise and redundant features from the dataset.
- **Improves performance:** PCA can help improve the performance of machine learning algorithms by reducing the dimensionality of the dataset, which in turn reduces the computational requirements.

Disadvantages:

- **May not preserve all the information:** PCA may not preserve all the information in the original dataset. This can lead to some loss of accuracy in certain cases.
- **May be difficult to interpret:** PCA can sometimes be difficult to interpret, especially when the number of principal components is large.
- **Requires scaling of data:** PCA requires the data to be scaled before applying the algorithm. This can be time-consuming and may require additional pre-processing steps.
- **May not work well with categorical variables**: PCA may not work well with datasets that have categorical variables. In such cases, other techniques like Factor Analysis or Multiple Correspondence Analysis may be more appropriate.

Linear Decrement Analysis (LDA):

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that is used to find a linear combination of features that separates two or more classes of objects or events. It aims to find a

projection of the data that maximizes the separation between classes while minimizing the variance within each class.

LDA is a supervised learning algorithm that is used for classification tasks where the classes are known a priori. It can be used to reduce the dimensionality of the data while preserving most of the information in the original dataset. It is commonly used in pattern recognition, machine learning, and computer vision applications.

LDA is typically used for multi-class classification. It can also be used as a dimensionality reduction technique. LDA best separates or discriminates (hence the name LDA) training instances by their classes. The major difference between LDA and PCA is that LDA finds a linear combination of input features that optimizes class separability while PCA attempts to find a set of uncorrelated components of maximum variance in a dataset. Another key difference between the two is that PCA is an unsupervised algorithm whereas LDA is a supervised algorithm where it takes class labels into account.

There are some limitations of LDA. To apply LDA, the data should be normally distributed. The dataset should also contain known class labels. The maximum number of components that LDA can find is the number of classes minus 1. If there are only 3 class labels in your dataset, LDA can find only 2 (3–1) components in dimensionality reduction. It is not needed to perform feature scaling to apply LDA. On the other hand, PCA needs scaled data. However, class labels are not needed for PCA. The maximum number of components that PCA can find is the number of input features in the original dataset.

LDA for dimensionality reduction should not be confused with LDA for multi-class classification. Both cases can be implemented using the Scikit-learn **LinearDiscriminantAnalysis ()** function. After fitting the model using **fit (X, y)**, we use the **predict(X)** method of the LDA object for multi-class classification. This will assign new instances to the classes in the original dataset. We can use the **transform(X)** method of the LDA object for dimensionality reduction. This will find a linear combination of new features that optimizes class separability.

The main steps involved in LDA are:

- **Standardize the dataset:** The first step is to standardize the dataset by scaling each feature to have zero mean and unit variance.
- **Compute the mean vectors:** The mean vector of each class is computed.
- **Compute the scatter matrices:** The within-class scatter matrix and the between-class scatter matrix are computed.
- **Compute the eigenvectors and eigenvalues**: The eigenvectors and eigenvalues of the matrix product of the inverse of the within-class scatter matrix and the between-class scatter matrix are computed.
- **Select the principal components:** The eigenvectors with the highest eigenvalues are selected as the principal components.
- **Transform the data:** The data is transformed by projecting it onto the new space defined by the selected principal components.

Advantages:

- LDA can reduce the dimensionality of the data while preserving most of the information in the original dataset.

- LDA is a supervised learning algorithm and takes into account the class information, which can result in better separation of classes.
- LDA can be used for feature extraction, which can be useful in image and speech recognition.

Disadvantages:

- LDA assumes that the classes have the same covariance matrix, which may not be true in some cases.
- LDA is sensitive to outliers, which can affect the results.
- LDA is computationally expensive for large datasets.

Difference between PCA and LDA:

PCA aims to transform the original dataset into a new set of variables called principal components that are linear combinations of the original variables. The principal components are chosen in such a way that the first component explains the largest variance in the data, and each subsequent component explains as much of the remaining variance as possible. PCA is an unsupervised technique, which means it doesn't use any class information.

LDA, on the other hand, is a supervised technique that aims to find a linear combination of the original variables that maximizes the separation between different classes. It finds the directions in the data that best separates the classes, and projects the data onto those directions. The goal is to reduce the within-class variance and increase the between-class variance. LDA is commonly used for classification problems where the goal is to predict the class of a new observation based on the available features.

- PCA performs dimensionality reduction by maximizing the variance of the data. Therefore, in most cases, feature standardization is necessary before applying PCA (see the exceptions here).
- LDA performs dimensionality reduction by maximizing the class separability of classification datasets. Therefore, feature standardization is optional here (we will verify this shortly).
- PCA does not require class labels. So, it can be used with classification, regression and even with unlabelled data!
- LDA requires class labels. So, it is used with classification datasets.
- PCA finds a set of uncorrelated features in a lower dimensional space. Therefore, PCA automatically removes multicollinearity in the data (learn more here).
- As explained earlier, LDA can be used for both supervised and unsupervised tasks. PCA can only be used for unsupervised dimensionality reduction.
- The maximum number of components that PCA can find is equal to the number of input features (original dimensionality) of the dataset! We often prefer to find a considerably low number of components that captures as much of the variance in the original data as possible.
- The maximum number of components that LDA can find is equal to the number of classes minus one in the classification dataset. For example, if there are only 3 classes in the dataset, LDA can find the maximum of 2 components.

- LDA is more effective than PCA for classification datasets because LDA reduces the dimensionality of the data by maximizing class separability. It is easier to draw decision boundaries for data with maximum class separability.

Truncated Singular Value Decomposition (SVD):

This method performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). It works well with sparse data in which many of the row values are zero. In contrast, PCA works well with dense data (non zero data). Truncated SVD can also be used with dense data. Another key difference between truncated SVD and PCA is that factorization for SVD is done on the data matrix while factorization for PCA is done on the covariance matrix.

The Scikit-learn implementation of truncated SVD is much easy. It can be done using the **TruncatedSVD()** function. The following Python code describes the implementation of truncated SVD and PCA techniques to the Iris dataset.

Truncated Singular Value Decomposition (SVD) is a dimensionality reduction technique that is commonly used in data science and machine learning. It is a modified version of Singular Value Decomposition (SVD) that is used to reduce the dimensionality of a large, sparse dataset while preserving important information.

SVD is a matrix factorization technique that decomposes a matrix into three other matrices, U, S, and V. The U and V matrices are orthogonal matrices, while the S matrix is a diagonal matrix containing the singular values of the original matrix. These singular values represent the importance of the corresponding features or variables in the data.

Truncated SVD is a modified version of SVD that only retains the top k singular values and the corresponding columns of U and V. This results in a lower-dimensional representation of the data that still captures the most important information. Truncated SVD is often used in natural language processing (NLP) tasks, such as text classification and topic modelling, where the data is typically high-dimensional and sparse.

The main advantage of Truncated SVD over other dimensionality reduction techniques is that it can handle sparse matrices efficiently. Additionally, it can be used with a wide range of data types, including numerical, categorical, and text data. However, Truncated SVD can sometimes lose important information during the dimensionality reduction process, and the optimal value of k (i.e., the number of retained singular values) may not always be clear.

Non-Linear Methods:

Kernal Principal Component Analysis (Kernal-PCA):

Kernel PCA is a non-linear dimensionality reduction technique that uses ***kernels***. It can also be considered as the non-linear form of normal PCA. Kernel PCA works well with non-linear datasets where normal PCA cannot be used efficiently.

The intuition behind Kernel PCA is something interesting. The data is first run through a kernel function and temporarily projects them into a new higher-dimensional feature space where the classes become

linearly separable (classes can be divided by drawing a straight line). Then the algorithm uses the normal PCA to project the data back onto a lower-dimensional space. In this way, Kernel PCA transforms non-linear data into a lower-dimensional space of data which can be used with linear classifiers.

In the Kernel PCA, we need to specify 3 important hyperparameters — the number of components we want to keep, the type of kernel and the kernel coefficient (also known as the *gamma*). For the type of kernel, we can use ***'linear', 'poly', 'rbf', 'sigmoid', 'cosine'***. The ***rbf kernel*** which is known as the **radial basis function kernel** is the most popular one.

One limitation of using the Kernel PCA for dimensionality reduction is that we have to specify a value for the ***gamma*** hyperparameter before running the algorithm. It requires implementing a hyperparameter tuning technique such as Grid Search to find an optimal value for the ***gamma***. It is beyond the scope of this article. But you can get help with the hyperparameter tuning process by using **"k-fold cross-validation"**.

t-distributed Stochastic Neighbour Embedding (t-SNE)

This is also a non-linear dimensionality reduction method mostly used for data visualization. In addition to that, it is widely used in image processing and NLP. The Scikit-learn documentation recommends you to use PCA or Truncated SVD before t-SNE if the number of features in the dataset is more than 50. The following is the general syntax to perform t-SNE after PCA. Also, note that feature scaling is required before PCA.

Multi-Dimensional Scaling (MDS):

MDA is another non-linear dimensionality reduction technique that tries to preserve the distances between instances while reducing the dimensionality of non-linear data. There are two types of MDS algorithms: Metric and Non-metric. The **MDS()** class in the Scikit-learn implements both by setting the ***metric*** hyperparameter to True (for Metric type) or False (for Non-metric type).

Isometric Mapping (Isomap):

This method performs non-linear dimensionality reduction through Isometric mapping. It is an extension of MDS or Kernel PCA. It connects each instance by calculating the *curved* or *geodesic* distance to its nearest neighbours and reduces dimensionality. The number of neighbours to consider for each point can be specified through the **n_neighbors** hyperparameter of the **Isomap()** class which implements the Isomap algorithm in the Scikit-learn.