1. What is feature engineering, and how does it work? Explain the various aspects of feature engineering in depth.

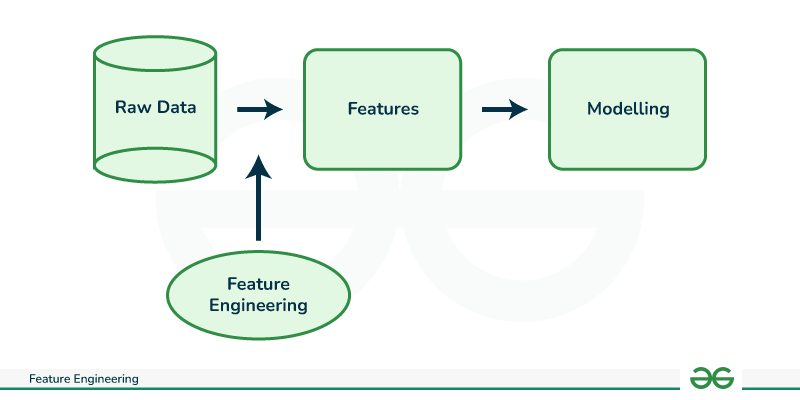
**What is Feature Engineering?**

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

**What is Feature Engineering?**

Feature engineering is the process of **transforming raw data into features that are suitable for machine learning models**. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.



**What is a Feature?**

In the context of machine learning, a feature (also known as a variable or attribute) is an individual measurable property or characteristic of a data point that is used as input for a machine learning algorithm. Features can be numerical, categorical, or text-based, and they represent different aspects of the data that are relevant to the problem at hand.

* For example, in a dataset of housing prices, features could include the number of bedrooms, the square footage, the location, and the age of the property. In a dataset of customer demographics, features could include age, gender, income level, and occupation.
* The choice and quality of features are critical in machine learning, as they can greatly impact the accuracy and performance of the model.

**Need for Feature Engineering in Machine Learning?**

We engineer features for various reasons, and some of the main reasons include:

* **Improve User Experience:** The primary reason we engineer features is to enhance the user experience of a product or service. By adding new features, we can make the product more intuitive, efficient, and user-friendly, which can increase user satisfaction and engagement.
* **Competitive Advantage:**Another reason we engineer features is to gain a competitive advantage in the marketplace. By offering unique and innovative features, we can differentiate our product from competitors and attract more customers.
* **Meet Customer Needs:**We engineer features to meet the evolving needs of customers. By analyzing user feedback, market trends, and customer behavior, we can identify areas where new features could enhance the product’s value and meet customer needs.
* **Increase Revenue:** Features can also be engineered to generate more revenue. For example, a new feature that streamlines the checkout process can increase sales, or a feature that provides additional functionality could lead to more upsells or cross-sells.
* **Future-Proofing:**Engineering features can also be done to future-proof a product or service. By anticipating future trends and potential customer needs, we can develop features that ensure the product remains relevant and useful in the long term.

**Processes Involved in Feature Engineering**

Feature engineering in Machine learning consists of mainly 5 processes: Feature Creation, Feature Transformation, Feature Extraction, Feature Selection, and Feature Scaling. It is an iterative process that requires experimentation and testing to find the best combination of features for a given problem. The success of a machine learning model largely depends on the quality of the features used in the model.

**1. Feature Creation**

Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

**Types of Feature Creation:**

1. **Domain-Specific:**Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.
2. **Data-Driven:**Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.
3. **Synthetic:**Generating new features by combining existing features or synthesizing new data points.

**Why Feature Creation?**

1. **Improves Model Performance:**By providing additional and more relevant information to the model, feature creation can increase the accuracy and precision of the model.
2. **Increases Model Robustness:**By adding additional features, the model can become more robust to outliers and other anomalies.
3. **Improves Model Interpretability:**By creating new features, it can be easier to understand the model’s predictions.
4. **Increases Model Flexibility:** By adding new features, the model can be made more flexible to handle different types of data.

**2. Feature Transformation**

[Feature Transformation](https://www.geeksforgeeks.org/feature-transformation-techniques-in-machine-learning/) is the process of transforming the features into a more suitable representation for the machine learning model. This is done to ensure that the model can effectively learn from the data.

**Types of Feature Transformation:**

1. [**Normalization**](https://www.geeksforgeeks.org/what-is-data-normalization/)**:**Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.
2. **Scaling:**Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.
3. **Encoding:**Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.
4. **Transformation:**Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

**Why Feature Transformation?**

1. **Improves Model Performance:**By transforming the features into a more suitable representation, the model can learn more meaningful patterns in the data.
2. **Increases Model Robustness:**Transforming the features can make the model more robust to outliers and other anomalies.
3. **Improves Computational Efficiency:**The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By transforming the features, it can be easier to understand the model’s predictions.

**3. Feature Extraction**

[Feature Extraction](https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/)is the process of creating new features from existing ones to provide more relevant information to the machine learning model. This is done by transforming, combining, or aggregating existing features.

**Types of Feature Extraction:**

1. **Dimensionality Reduction:** Reducing the number of features by transforming the data into a lower-dimensional space while retaining important information. Examples are [PCA](https://www.geeksforgeeks.org/principal-component-analysis-pca/)and[t-SNE](https://www.geeksforgeeks.org/ml-t-distributed-stochastic-neighbor-embedding-t-sne-algorithm/).
2. **Feature Combination:**Combining two or more existing features to create a new one. For example, the interaction between two features.
3. **Feature Aggregation:**Aggregating features to create a new one. For example, calculating the mean, sum, or count of a set of features.
4. **Feature Transformation:**Transforming existing features into a new representation. For example, log transformation of a feature with a skewed distribution.

**Why Feature Extraction?**

1. **Improves Model Performance:**By creating new and more relevant features, the model can learn more meaningful patterns in the data.
2. **Reduces Overfitting:** By reducing the dimensionality of the data, the model is less likely to overfit the training data.
3. **Improves Computational Efficiency:** The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By creating new features, it can be easier to understand the model’s predictions.

**4. Feature Selection**

[Feature Selection](https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/) is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model. It is an important step in the feature engineering process as it can have a significant impact on the model’s performance.

**Types of Feature Selection:**

1. **Filter Method:**Based on the statistical measure of the relationship between the feature and the target variable. Features with a high correlation are selected.
2. **Wrapper Method:**Based on the evaluation of the feature subset using a specific machine learning algorithm. The feature subset that results in the best performance is selected.
3. **Embedded Method:**Based on the feature selection as part of the training process of the machine learning algorithm.

**Why Feature Selection?**

1. **Reduces Overfitting:** By using only the most relevant features, the model can generalize better to new data.
2. **Improves Model Performance:** Selecting the right features can improve the accuracy, precision, and recall of the model.
3. **Decreases Computational Costs:**A smaller number of features requires less computation and storage resources.
4. **Improves Interpretability:** By reducing the number of features, it is easier to understand and interpret the results of the model.

**5. Feature Scaling**

[Feature Scaling](https://www.geeksforgeeks.org/ml-feature-scaling-part-1/) is the process of transforming the features so that they have a similar scale. This is important in machine learning because the scale of the features can affect the performance of the model.

**Types of Feature Scaling:**

1. [**Min-Max Scaling**](https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/)**:**Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.
2. **Standard Scaling:** Rescaling the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation.
3. **Robust Scaling:** Rescaling the features to be robust to outliers by dividing them by the interquartile range.

**Why Feature Scaling?**

1. **Improves Model Performance:**By transforming the features to have a similar scale, the model can learn from all features equally and avoid being dominated by a few large features.
2. **Increases Model Robustness:**By transforming the features to be robust to outliers, the model can become more robust to anomalies.
3. **Improves Computational Efficiency:**Many machine learning algorithms, such as k-nearest neighbors, are sensitive to the scale of the features and perform better with scaled features.
4. **Improves Model Interpretability:** By transforming the features to have a similar scale, it can be easier to understand the model’s predictions.

**What are the Steps in Feature Engineering?**

The steps for feature engineering vary per different Ml engineers and data scientists. Some of the common steps that are involved in most machine-learning algorithms are:

1. **Data Cleansing**
   * Data cleansing (also known as data cleaning or data scrubbing) involves identifying and removing or correcting any errors or inconsistencies in the dataset. This step is important to ensure that the data is accurate and reliable.
2. **Data Transformation**
3. **Feature Extraction**
4. **Feature Selection**
   * Feature selection involves selecting the most relevant features from the dataset for use in machine learning. This can include techniques like correlation analysis, mutual information, and stepwise regression.
5. **Feature Iteration**
   * Feature iteration involves refining and improving the features based on the performance of the machine learning model. This can include techniques like adding new features, removing redundant features and transforming features in different ways.

*Overall, the goal of feature engineering is to create a set of informative and relevant features that can be used to train a machine learning model and improve its accuracy and performance. The specific steps involved in the process may vary depending on the type of data and the specific machine-learning problem at hand.*

**Techniques Used in Feature Engineering**

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. There are various techniques that can be used in feature engineering to create new features by combining or transforming the existing ones. The following are some of the commonly used feature engineering techniques:

**One-Hot Encoding**

[One-hot encoding](https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/) is a technique used to transform categorical variables into numerical values that can be used by machine learning models. In this technique, each category is transformed into a binary value indicating its presence or absence. For example, consider a categorical variable “Colour” with three categories: Red, Green, and Blue. One-hot encoding would transform this variable into three binary variables: Colour\_Red, Colour\_Green, and Colour\_Blue, where the value of each variable would be 1 if the corresponding category is present and 0 otherwise.

**Binning**

[Binning](https://www.geeksforgeeks.org/binning-in-data-mining/) is a technique used to transform continuous variables into categorical variables. In this technique, the range of values of the continuous variable is divided into several bins, and each bin is assigned a categorical value. For example, consider a continuous variable “Age” with values ranging from 18 to 80. Binning would divide this variable into several age groups such as 18-25, 26-35, 36-50, and 51-80, and assign a categorical value to each age group.

**Scaling**

The most common scaling techniques are standardization and normalization. Standardization scales the variable so that it has zero mean and unit variance. Normalization scales the variable so that it has a range of values between 0 and 1.

**Feature Split**  
[Feature splitting](https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/) is a powerful technique used in feature engineering to improve the performance of machine learning models. It involves dividing single features into multiple sub-features or groups based on specific criteria. This process unlocks valuable insights and enhances the model’s ability to capture complex relationships and patterns within the data.

**Text Data Preprocessing**  
Text data requires special preprocessing techniques before it can be used by machine learning models. Text preprocessing involves removing stop words, stemming, lemmatization, and vectorization. Stop words are common words that do not add much meaning to the text, such as “the” and “and”. Stemming involves reducing words to their root form, such as converting “running” to “run”. Lemmatization is similar to stemming, but it reduces words to their base form, such as converting “running” to “run”. Vectorization involves transforming text data into numerical vectors that can be used by machine learning models.

**Feature Engineering Tools**

There are several tools available for feature engineering. Here are some popular ones:

**1. Featuretools**

Featuretools is a Python library that enables automatic feature engineering for structured data. It can extract features from multiple tables, including relational databases and CSV files, and generate new features based on user-defined primitives. Some of its features include:

* Automated feature engineering using machine learning algorithms.
* Support for handling time-dependent data.
* Integration with popular Python libraries, such as pandas and scikit-learn.
* Visualization tools for exploring and analyzing the generated features.
* Extensive documentation and tutorials for getting started.

**2. TPOT**

TPOT (Tree-based Pipeline Optimization Tool) is an automated machine learning tool that includes feature engineering as one of its components. It uses genetic programming to search for the best combination of features and machine learning algorithms for a given dataset. Some of its features include:

* Automatic feature selection and transformation.
* Support for multiple types of machine learning models, including regression, classification, and clustering.
* Ability to handle missing data and categorical variables.
* Integration with popular Python libraries, such as scikit-learn and pandas.
* Interactive visualization of the generated pipelines.

**3. DataRobot**

DataRobot is a machine learning automation platform that includes feature engineering as one of its capabilities. It uses automated machine learning techniques to generate new features and select the best combination of features and models for a given dataset. Some of its features include:

* Automatic feature engineering using machine learning algorithms.
* Support for handling time-dependent and text data.
* Integration with popular Python libraries, such as pandas and scikit-learn.
* Interactive visualization of the generated models and features.
* Collaboration tools for teams working on machine learning projects.

**4. Alteryx**

Alteryx is a data preparation and automation tool that includes feature engineering as one of its features. It provides a visual interface for creating data pipelines that can extract, transform, and generate features from multiple data sources. Some of its features include:

* Support for handling structured and unstructured data.
* Integration with popular data sources, such as Excel and databases.
* Pre-built tools for feature extraction and transformation.
* Support for custom scripting and code integration.
* Collaboration and sharing tools for teams working on data projects.

**5. H2O.ai**

H2O.ai is an open-source machine learning platform that includes feature engineering as one of its capabilities. It provides a range of automated feature engineering techniques, such as feature scaling, imputation, and encoding, as well as manual feature engineering capabilities for more advanced users. Some of its features include:

* Automatic and manual feature engineering options.
* Support for structured and unstructured data, including text and image data.
* Integration with popular data sources, such as CSV files and databases.
* Interactive visualization of the generated features and models.
* Collaboration and sharing tools for teams working on machine learning projects.

Overall, these tools can help streamline and automate the feature engineering process, making it easier and faster to create informative and relevant features for machine learning models.

1. What is feature selection, and how does it work? What is the aim of it? What are the various methods of function selection?

**Feature Selection Techniques in Machine Learning**

**Feature selection:**

Feature selection is a process that chooses a subset of features from the original features so that the feature space is optimally reduced according to a certain criterion.

 Feature selection is a critical step in the feature construction process. In text categorization problems, some words simply do not appear very often. Perhaps the word “groovy” appears in exactly one training document, which is positive. Is it really worth keeping this word around as a feature ? It’s a dangerous endeavor because it’s hard to tell with just one training example if it is really correlated with the positive class or is it just noise. You could hope that your learning algorithm is smart enough to figure it out. Or you could just remove it.

There are three general classes of feature selection algorithms: **Filter methods, wrapper methods and embedded methods**.

The role of feature selection in machine learning is,

1. To reduce the dimensionality of feature space.

2. To speed up a learning algorithm.

3. To improve the predictive accuracy of a classification algorithm.

4. To improve the comprehensibility of the learning results.

**Features Selection Algorithms are as follows:**

**1**. **Instance based approaches:** There is no explicit procedure for feature subset generation. Many small data samples are sampled from the data. Features are weighted according to their roles in differentiating instances of different classes for a data sample. Features with higher weights can be selected.

**2. Nondeterministic approaches:**Genetic algorithms and simulated annealing are also used in feature selection.

**3. Exhaustive complete approaches:**Branch and Bound evaluates estimated accuracy and ABB checks an inconsistency measure that is monotonic. Both start with a full feature set until the preset bound cannot be maintained.

While building a machine learning model for real-life dataset, we come across a lot of features in the dataset and not all these features are important every time. Adding unnecessary features while training the model leads us to reduce the overall accuracy of the model, increase the complexity of the model and decrease the generalization capability of the model and makes the model biased. Even the saying “Sometimes less is better” goes as well for the machine learning model. Hence, **feature selection** is one of the important steps while building a machine learning model. Its goal is to find the best possible set of features for building a machine learning model.

Some popular techniques of feature selection in machine learning are:

* Filter methods
* Wrapper methods
* Embedded methods

**Filter Methods**

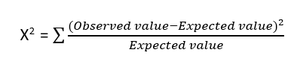
These methods are generally used while doing the pre-processing step. These methods select features from the dataset irrespective of the use of any machine learning algorithm. In terms of computation, they are very fast and inexpensive and are very good for removing duplicated, correlated, redundant features but these methods do not remove multicollinearity. Selection of feature is evaluated individually which can sometimes help when features are in isolation (don’t have a dependency on other features) but will lag when a combination of features can lead to increase in the overall performance of the model.

https://media.geeksforgeeks.org/wp-content/uploads/20201204094030/15.PNG

*Filter Methods Implementation*

Some techniques used are:

* **Information Gain –** It is defined as the amount of information provided by the feature for identifying the target value and measures reduction in the entropy values. Information gain of each attribute is calculated considering the target values for feature selection.
* **Chi-square test —** Chi-square method (X2) is generally used to test the relationship between categorical variables. It compares the observed values from different attributes of the dataset to its expected value.

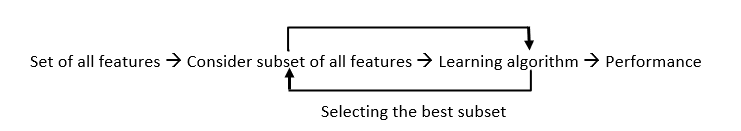


*Chi-square Formula*

* **Fisher’s Score –** Fisher’s Score selects each feature independently according to their scores under Fisher criterion leading to a suboptimal set of features. The larger the Fisher’s score is, the better is the selected feature.
* **Correlation Coefficient –** Pearson’s Correlation Coefficient is a measure of quantifying the association between the two continuous variables and the direction of the relationship with its values ranging from *-1 to 1*.
* **Variance Threshold –** It is an approach where all features are removed whose variance doesn’t meet the specific threshold. By default, this method removes features having zero variance. The assumption made using this method is higher variance features are likely to contain more information.
* **Mean Absolute Difference (MAD) –** This method is similar to variance threshold method but the difference is there is no square in MAD. This method calculates the mean absolute difference from the mean value.
* **Dispersion Ratio –** Dispersion ratio is defined as the ratio of the Arithmetic mean (AM) to that of Geometric mean (GM) for a given feature. Its value ranges from *+1 to ∞ as AM ≥ GM* for a given feature. Higher dispersion ratio implies a more relevant feature.
* **Mutual Dependence –**This method measures if two variables are mutually dependent, and thus provides the amount of information obtained for one variable on observing the other variable. Depending on the presence/absence of a feature, it measures the amount of information that feature contributes to making the target prediction.
* **Relief –** This method measures the quality of attributes by randomly sampling an instance from the dataset and updating each feature and distinguishing between instances that are near to each other based on the difference between the selected instance and two nearest instances of same and opposite classes.

**Wrapper methods:**

Wrapper methods, also referred to as greedy algorithms train the algorithm by using a subset of features in an iterative manner. Based on the conclusions made from training in prior to the model, addition and removal of features takes place. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features has been achieved. The main advantage of wrapper methods over the filter methods is that they provide an optimal set of features for training the model, thus resulting in better accuracy than the filter methods but are computationally more expensive.



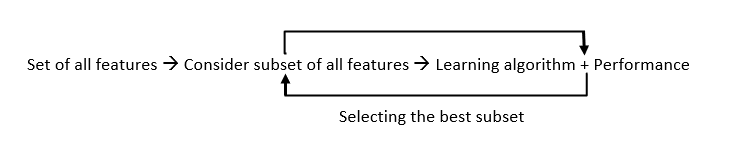
*Wrapper Methods Implementation*

Some techniques used are:

* **Forward selection –**This method is an iterative approach where we initially start with an empty set of features and keep adding a feature which best improves our model after each iteration. The stopping criterion is till the addition of a new variable does not improve the performance of the model.
* **Backward elimination –** This method is also an iterative approach where we initially start with all features and after each iteration, we remove the least significant feature. The stopping criterion is till no improvement in the performance of the model is observed after the feature is removed.
* **Bi-directional elimination –** This method uses both forward selection and backward elimination technique simultaneously to reach one unique solution.
* **Exhaustive selection –** This technique is considered as the brute force approach for the evaluation of feature subsets. It creates all possible subsets and builds a learning algorithm for each subset and selects the subset whose model’s performance is best.
* **Recursive elimination –** This greedy optimization method selects features by recursively considering the smaller and smaller set of features. The estimator is trained on an initial set of features and their importance is obtained using feature\_importance\_attribute. The least important features are then removed from the current set of features till we are left with the required number of features.

**Embedded methods:**

In embedded methods, the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods. Embedded methods encounter the drawbacks of filter and wrapper methods and merge their advantages. These methods are faster like those of filter methods and more accurate than the filter methods and take into consideration a combination of features as well.



*Embedded Methods Implementation*

Some techniques used are:

* **Regularization –** This method adds a penalty to different parameters of the machine learning model to avoid over-fitting of the model. This approach of feature selection uses Lasso (L1 regularization) and Elastic nets (L1 and L2 regularization). The penalty is applied over the coefficients, thus bringing down some coefficients to zero. The features having zero coefficient can be removed from the dataset.
* **Tree-based methods –**These methods such as Random Forest, Gradient Boosting provides us feature importance as a way to select features as well. Feature importance tells us which features are more important in making an impact on the target feature.

**Conclusion:**

Apart from the methods discussed above, there are many other methods of feature selection. Using hybrid methods for feature selection can offer a selection of best advantages from other methods, leading to reduce in the disadvantages of the algorithms. These models can provide greater accuracy and performance when compared to other methods. Dimensionality reduction techniques such as Principal Component Analysis (PCA), Heuristic Search Algorithms, etc. don’t work in the way as to feature selection techniques but can help us to reduce the number of features.

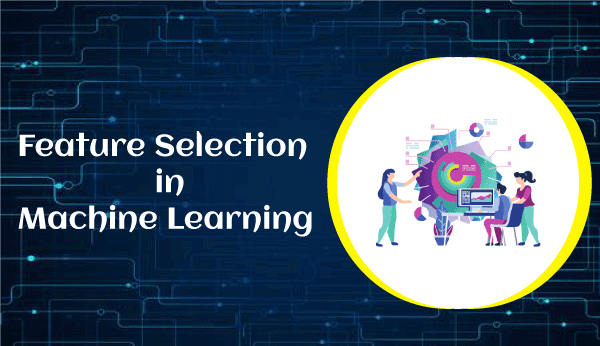
Feature selection is a wide, complicated field and a lot of studies has already been made to figure out the best methods. It depends on the machine learning engineer to combine and innovate approaches, test them and then see what works best for the given problem.

3. Describe the function selection filter and wrapper approaches. State the pros and cons of each approach?

Feature Selection Techniques in Machine Learning

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.

While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.



Feature selection is one of the important concepts of machine learning, which highly impacts the performance of the model. As machine learning works on the concept of "Garbage In Garbage Out", so we always need to input the most appropriate and relevant dataset to the model in order to get a better result.

In this topic, we will discuss different feature selection techniques for machine learning. But before that, let's first understand some basics of feature selection.

* **What is Feature Selection?**
* **Need for Feature Selection**
* **Feature Selection Methods/Techniques**
* **Feature Selection statistics**

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.

Need for Feature Selection

Before implementing any technique, it is really important to understand, need for the technique and so for the Feature Selection. As we know, in machine learning, it is necessary to provide a pre-processed and good input dataset in order to get better outcomes. We collect a huge amount of data to train our model and help it to learn better. Generally, the dataset consists of noisy data, irrelevant data, and some part of useful data. Moreover, the huge amount of data also slows down the training process of the model, and with noise and irrelevant data, the model may not predict and perform well. So, it is very necessary to remove such noises and less-important data from the dataset and to do this, and Feature selection techniques are used.

Selecting the best features helps the model to perform well. For example, Suppose we want to create a model that automatically decides which car should be crushed for a spare part, and to do this, we have a dataset. This dataset contains a Model of the car, Year, Owner's name, Miles. So, in this dataset, the name of the owner does not contribute to the model performance as it does not decide if the car should be crushed or not, so we can remove this column and select the rest of the features(column) for the model building.

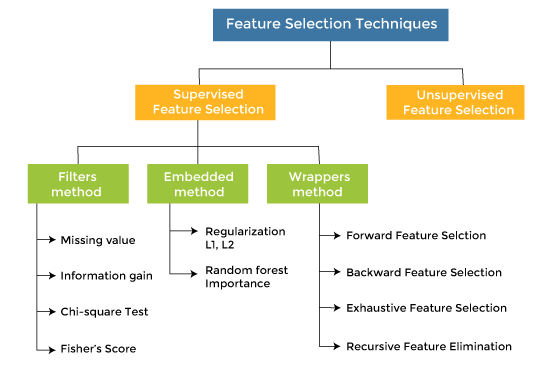
Below are some benefits of using feature selection in machine learning:

* **It helps in avoiding the curse of dimensionality.**
* **It helps in the simplification of the model so that it can be easily interpreted by the researchers.**
* **It reduces the training time.**
* **It reduces overfitting hence enhance the generalization.**

Feature Selection Techniques

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

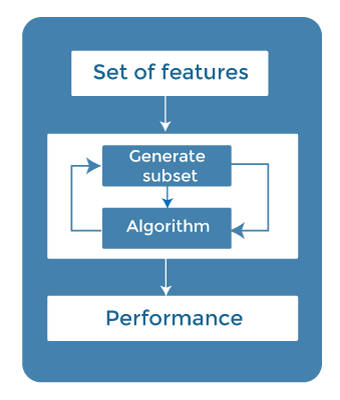
* **Supervised Feature Selection technique**  
  Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
* **Unsupervised Feature Selection technique**  
  Unsupervised Feature selection techniques ignore the target variable and can be used for the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



There are mainly three techniques under supervised feature Selection:

1. 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:

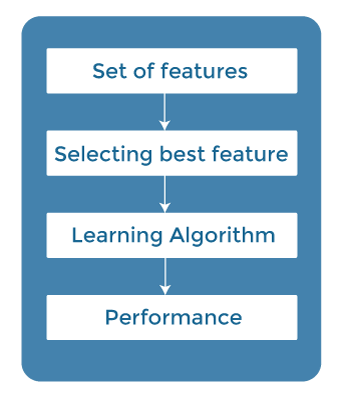
* **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new 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 & make each possible combination of features and return 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.*

2. 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 feature 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
* Chi-square Test
* Fisher's Score
* Missing Value Ratio

**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.

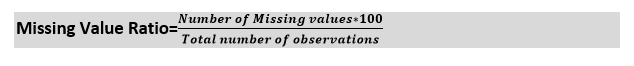
**Chi-square Test:** Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

**Fisher's Score:**

Fisher's score is one of the popular supervised technique of features 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.

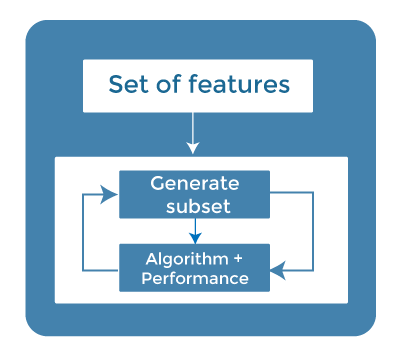
**Missing Value Ratio:**

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 is having more than the threshold value can be dropped.



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.

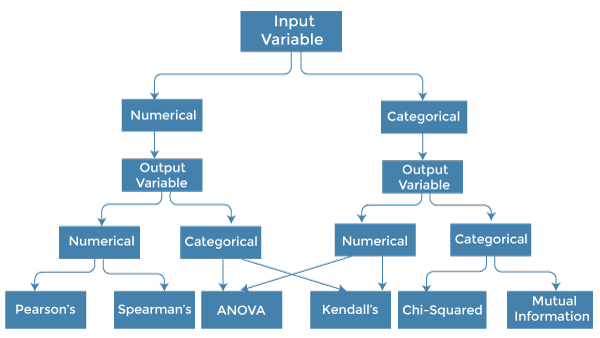


These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:

* **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 to 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 that which feature selection method will work properly for their model. The more we know the datatypes 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. |

4.

1. Describe the overall feature selection process.

Feature Selection In Machine Learning: All You Need to Know



Table of Contents

The input variables that we give to our [machine learning models](https://www.simplilearn.com/machine-learning-models-article) are called features. Each column in our dataset constitutes a feature. To train an optimal model, we need to make sure that we use only the essential features. If we have too many features, the model can capture the unimportant patterns and learn from noise. The method of choosing the important parameters of our data is called Feature Selection.

In this article titled ‘Everything you need to know about Feature Selection’, we will teach you all you need to know about feature selection.

Your AI/ML Career is Just Around The Corner!

Why Feature Selection?

Machine learning models follow a simple rule: whatever goes in, comes out. If we put garbage into our model, we can expect the output to be garbage too. In this case, garbage refers to noise in our data.

To train a model, we collect enormous quantities of data to help the machine learn better. Usually, a good portion of the data collected is noise, while some of the columns of our dataset might not contribute significantly to the performance of our model. Further, having a lot of data can slow down the training process and cause the model to be slower. The model may also learn from this irrelevant data and be inaccurate.

Feature selection is what separates good [data scientists](https://www.simplilearn.com/tutorials/data-science-tutorial/how-to-become-a-data-scientist) from the rest. Given the same model and computational facilities, why do some people win in competitions with faster and more accurate models? The answer is Feature Selection. Apart from choosing the right model for our data, we need to choose the right data to put in our model.

Consider a table which contains information on old cars. The model decides which cars must be crushed for spare parts.

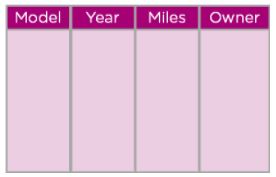


Figure 1: Old cars dataset

In the above table, we can see that the model of the car, the year of manufacture, and the miles it has traveled are pretty important to find out if the car is old enough to be crushed or not. However, the name of the previous owner of the car does not decide if the car should be crushed or not. Further, it can confuse the [algorithm](https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm) into finding patterns between names and the other features. Hence we can drop the column.

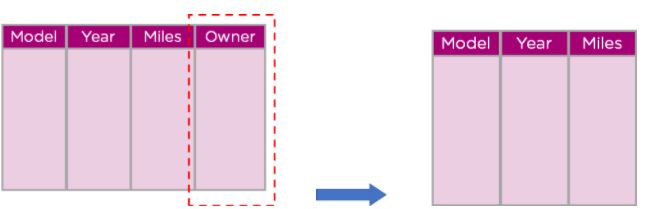


 Figure 2: Dropping columns for feature selection

Your AI/ML Career is Just Around The Corner!

What is Feature Selection?

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.

It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.



Figure 3: Feature Selection

Feature Selection Models

Feature selection models are of two types:

1. Supervised Models: Supervised feature selection refers to the method which uses the output label class for feature selection. They use the target variables to identify the variables which can increase the efficiency of the model
2. Unsupervised Models: Unsupervised feature selection refers to the method which does not need the output label class for feature selection. We use them for unlabelled data.

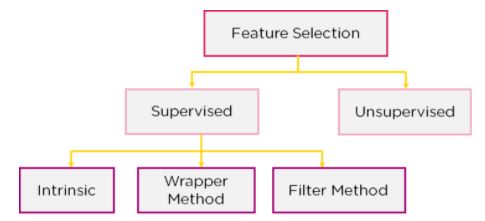


Figure 4: Feature Selection Models

We can further divide the supervised models into three :

1. Filter Method: In this method, features are dropped based on their relation to the output, or how they are correlating to the output. We use correlation to check if the features are positively or negatively correlated to the output labels and drop features accordingly. Eg: Information Gain, [Chi-Square Test](https://www.simplilearn.com/tutorials/statistics-tutorial/chi-square-test), Fisher’s Score, etc.

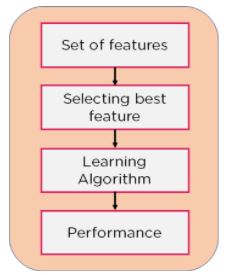


Figure 5: Filter Method flowchart

2. Wrapper Method: We split our data into subsets and train a model using this. Based on the output of the model, we add and subtract features and train the model again. It forms the subsets using a greedy approach and evaluates the accuracy of all the possible combinations of features. Eg: Forward Selection, Backwards Elimination, etc.

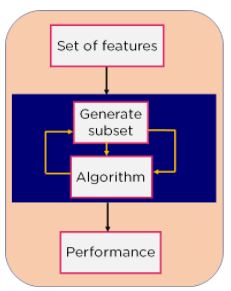


                                              Figure 6: Wrapper Method Flowchart

3. Intrinsic Method: This method combines the qualities of both the Filter and Wrapper method to create the best subset.

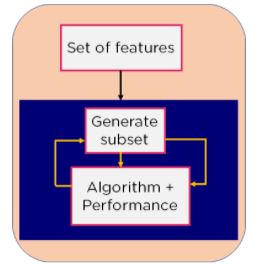


                                                Figure 7: Intrinsic Model Flowchart

This method takes care of the machine training iterative process while maintaining the computation cost to be minimum. Eg: Lasso and Ridge Regression.

How to Choose a Feature Selection Model?

How do we know which feature selection model will work out for our model? The process is relatively simple, with the model depending on the types of input and output variables.

Variables are of two main types:

* Numerical Variables: Which include integers, float, and numbers.
* Categorical Variables: Which include labels, strings, boolean variables, etc.

Based on whether we have numerical or categorical variables as inputs and outputs, we can choose our feature selection model as follows:

|  |  |  |
| --- | --- | --- |
| Input Variable | Output Variable | Feature Selection Model |
| Numerical | Numerical | * Pearson’s correlation coefficient * Spearman’s rank coefficient |
| 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. |

                                            Table 1: Feature Selection Model lookup

Feature Selection With Python

Let’s get hands-on experience in feature selection by working on the Kobe Bryant Dataset which analyses shots taken by Kobe from different areas of the court to determine which ones will go into the basket.

The dataset is as shown:

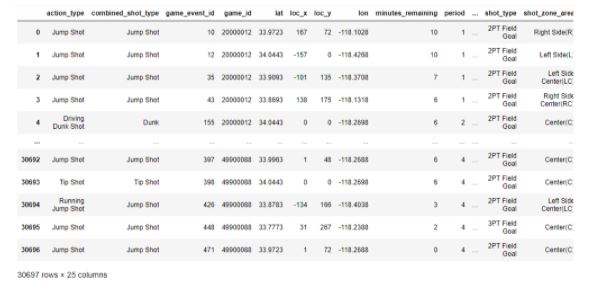


                                                        Figure 8: Kobe Bryant Dataset

As we can see, the dataset has 25 different columns. We will not need all of them.

We first begin by loading in the necessary modules.

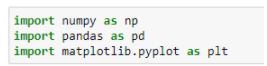


                                                        Figure 9: Importing modules

First, let's check out the loc\_x and loc\_y columns. They probably represent longitude and latitude.



                                     Figure 10: Plotting the latitude and longitude columns in our dataset

The figure is as shown:

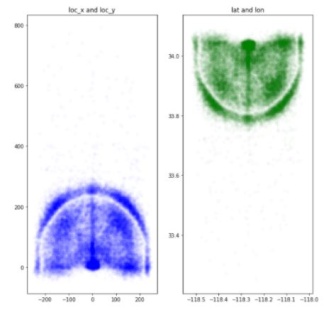


                                                 Figure 11: Plotting Latitude and Longitude

From the above figures, we can see that they resemble the two ‘D’s on a basketball court. Instead of having two separate columns, we can change the coordinates into polar form and have a single column [‘angle’].

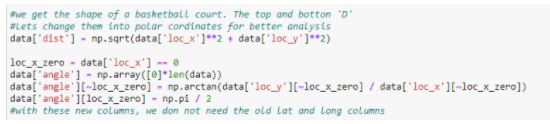


                                          Figure 12: Changing Latitude and Longitude into polar form

Your AI/ML Career is Just Around The Corner!

We can combine the minutes and seconds columns into a single column for time.



                                           Figure 13: Combining two columns

Let’s look at the unique values in the ‘team\_id’ and ‘team\_name’ columns:

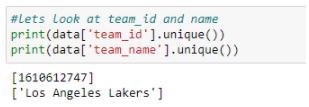


                                          Figure 14: Unique values in ‘team\_id’ and ‘team\_name’

The entire column contains only one value and can be dropped. Let’s take a look at the ‘match\_up’ and ‘opponent’ columns :

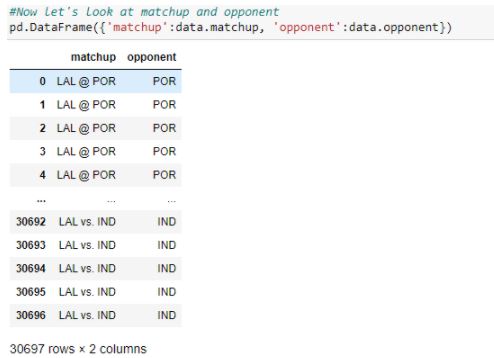


                                          Figure 15: ‘match\_up’ and ‘opponent’ columns

Again, they contain the same information. Let’s plot the values of ‘dist’ and ‘shot\_distance’ columns on the same graph to see how they differ:

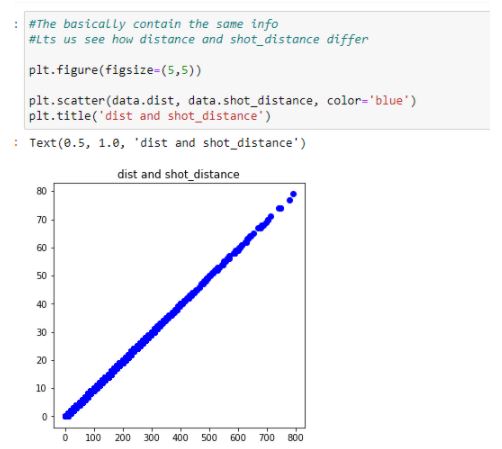


                                                   Figure 16: Plotting ‘dist’ and ‘shot\_distance’ columns

Again, they contain exactly the same information. Let’s take a look at columns shot\_zone\_area, shot\_zone\_basic and shot\_zone\_range.



                                                Figure 17: Plotting the different shot zones columns

The figure depicted below shows the plots :

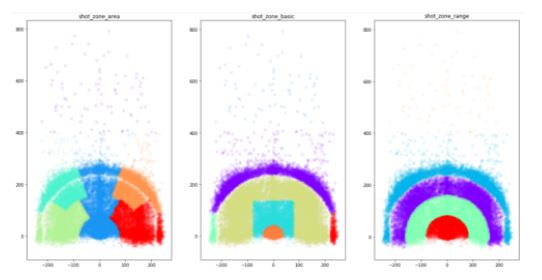


                                                        Figure 18: Different shot zones

We can see that they contain the different parts of the court from where the shots were taken. This information already exists in the angle and dist columns.

Now, let’s drop all the useless columns.



                                                        Figure 19: Dropping Columns

After merging columns and removing useless columns, we get a dataset that contains only 11 important columns.



                                                             Figure 20: Final Dataset

Your AI/ML Career is Just Around The Corner!

Conclusion

In this article titled ‘Everything you need to know about Feature Selection’, we got an idea of how important it is to select the best features for our machine learning model. We then took a look at what feature selection is and some feature selection models. We then moved onto a simple way to choose the right feature selection model based on the input and output values. Finally, we saw how to implement feature selection in [Python](https://www.simplilearn.com/learn-the-basics-of-python-article) with a demo. If you are looking to learn more about feature selection and related fundamental features of Python, Simplielarn’s [Python Certification Course](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCoursepages) would be ideal for you. This python certification course covers the basics fundamentals of python including data operations, conditional statements, shell scripting, and [Django](https://www.simplilearn.com/tutorials/python-tutorial/python-django" \o "Django" \t "_blank) and much more, and prepares you for a rewarding career as a professional [Python programmer.](https://www.simplilearn.com/tutorials/python-tutorial/how-to-become-python-developer)

Was this article on feature selection useful to you? Do you have any doubts or questions for us? Mention them in this article's comments section, and we'll have our experts answer them for you at the earliest!

ii. Explain the key underlying principle of feature extraction using an example. What are the most widely used function extraction algorithms?

**Feature Extraction Techniques – NLP**

Last Updated : 01 Feb, 2023

**Introduction :**

This article focuses on basic feature extraction techniques in NLP to analyse the similarities between pieces of text. Natural Language Processing (NLP) is a branch of computer science and machine learning that deals with training computers to process a large amount of human (natural) language data. Briefly, NLP is the ability of computers to understand human language. **Need of feature extraction techniques** Machine Learning algorithms learn from a pre-defined set of features from the training data to produce output for the test data. But the main problem in working with language processing is that machine learning algorithms cannot work on the raw text directly. So, we need some feature extraction techniques to convert text into a matrix(or vector) of features. Some of the most popular methods of feature extraction are :

* Bag-of-Words
* TF-IDF

**Bag of Words:**

The bag of words model is used for text representation and feature extraction in natural language processing and information retrieval tasks. It represents a text document as a multiset of its words, disregarding grammar and word order, but keeping the frequency of words. This representation is useful for tasks such as text classification, document similarity, and text clustering.

Bag-of-Words is one of the most fundamental methods to transform tokens into a set of features. The BoW model is used in document classification, where each word is used as a feature for training the classifier. For example, in a task of review based sentiment analysis, the presence of words like *‘fabulous’, ‘excellent’* indicates a positive review, while words like *‘annoying’, ‘poor’* point to a negative review . There are 3 steps while creating a BoW model :

1. The first step is **text-preprocessing** which involves:
   1. converting the entire text into lower case characters.
   2. removing all punctuations and unnecessary symbols.
2. The second step is to **create a vocabulary** of all unique words from the corpus. Let’s suppose, we have a hotel review text. Let’s consider 3 of these reviews, which are as follows :
3. *good movie*
4. *not a good movie*
5. *did not like*
6. Now, we consider all the unique words from the above set of reviews to create a vocabulary, which is going to be as follows :

*{good, movie, not, a, did, like}*

1. In the third step, we **create a matrix of features** by assigning a separate column for each word, while each row corresponds to a review. This process is known as **Text Vectorization**. Each entry in the matrix signifies the presence(or absence) of the word in the review. We put **1** if the word is present in the review, and **0** if it is not present.

For the above example, the matrix of features will be as follows :

| **good** | **movie** | **not** | **a** | **did** | **like** |
| --- | --- | --- | --- | --- | --- |
| **1** | **1** | 0 | 0 | 0 | 0 |
| **1** | **1** | **1** | **1** | 0 | 0 |
| 0 | 0 | **1** | 0 | **1** | **1** |

A major drawback in using this model is that the order of occurrence of words is lost, as we create a vector of tokens in randomised order.However, we can solve this problem by considering **N-grams**(mostly bigrams) instead of individual words(i.e. unigrams). This can preserve local ordering of words. If we consider all possible bigrams from the given reviews, the above table would look like:

| **good movie** | **movie** | **did not** | **a** | **…** |
| --- | --- | --- | --- | --- |
| **1** | **1** | 0 | 0 | … |
| **1** | **1** | 0 | **1** | … |
| 0 | 0 | **1** | 0 | … |

However, this table will come out to be very large, as there can be a lot of possible bigrams by considering all possible consecutive word pairs. Also, using N-grams can result in a huge **sparse**(has a lot of 0’s) matrix, if the size of the vocabulary is large, making the computation really complex!! Thus, we have to remove a few N-grams based on their frequency. Like, we can always remove **high-frequency N-grams**, because they appear in almost all documents. These high-frequency N-grams are generally articles, determiners, etc. most commonly called as **StopWords**. Similarly, we can also remove low frequency N-grams because these are really rare(i.e. generally appear in 1 or 2 reviews)!! These types of N-grams are generally typos(or typing mistakes). Generally, medium frequency N-grams are considered as the most ideal. However, there are some N-grams which are really rare in our corpus but can highlight a specific issue. Let’s suppose, there is a review that says – *“Wi-Fi breaks often”.* Here, the N-gram *‘Wi-Fi breaks* can’t be too frequent, but it highlights a major problem that needs to be looked upon. Our BoW model would not capture such N-grams since its frequency is really low. To solve this type of problem, we need another model i.e. **TF-IDF Vectorizer**, which we will study next. **Code : Python code for creating a BoW model is:**

* Python3

|  |
| --- |
| # Creating the Bag of Words model  word2count = {}  for data in dataset:      words = nltk.word\_tokenize(data)      for word in words:          if word not in word2count.keys():              word2count[word] = 1          else:              word2count[word] += 1 |

**Issues of Bag of Words:**

The following are some of the issues with the Bag of Words model for text representation and analysis:

1. High dimensionality: The resulting feature space can be very high dimensional, which may lead to issues with overfitting and computational efficiency.
2. Lack of context information: The bag of words model only considers the frequency of words in a document, disregarding grammar, word order, and context.
3. Insensitivity to word associations: The bag of words model doesn’t consider the associations between words, and the semantic relationships between words in a document.
4. Lack of semantic information: As the bag of words model only considers individual words, it does not capture semantic relationships or the meaning of words in context.
5. Importance of stop words: Stop words, such as “the”, “and”, “a”, etc., can have a large impact on the bag of words representation of a document, even though they may not carry much meaning.
6. Sparsity: For many applications, the bag of words representation of a document can be very sparse, meaning that most entries in the resulting feature vector will be zero. This can lead to issues with computational efficiency and difficulty in interpretability.

**TF-IDF Vectorizer :**

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used for information retrieval and natural language processing tasks. It reflects the importance of a word in a document relative to an entire corpus. The basic idea is that a word that occurs frequently in a document but rarely in the entire corpus is more informative than a word that occurs frequently in both the document and the corpus.

TF-IDF is used for:

1. Text retrieval and information retrieval systems  
2. Document classification and text categorization  
3. Text summarization  
4. Feature extraction for text data in machine learning algorithms.

TF-IDF stands for *term frequency-inverse document frequency*. It highlights a specific issue which might not be too frequent in our corpus but holds great importance. The TF–IFD value increases proportionally to the number of times a word appears in the document and decreases with the number of documents in the corpus that contain the word. It is composed of 2 sub-parts, which are :

1. Term Frequency (TF)
2. Inverse Document Frequency (IDF)

**Term Frequency(TF) :**Term frequency specifies how frequently a term appears in the entire document.It can be thought of as the probability of finding a word within the document.It calculates the number of times a word occurs in a review , with respect to the total number of words in the review .It is formulated as:

A different scheme for calculating tf is *log normalization*. And it is formulated as:

where,  is the frequency of the term t in document D. **Inverse Document Frequency(IDF) :**The inverse document frequency is a measure of whether a term is rare or frequent across the documents in the entire corpus. It highlights those words which occur in very few documents across the corpus, or in simple language, the words that are rare have high IDF score. IDF is a log normalised value, that is obtained by dividing the total number of documents in the corpus by the number of documents containing the term , and taking the logarithm of the overall term.

where,  is the frequency of the term t in document D.  is the total number of documents in the corpus.  is the count of documents in the corpus, which contains the term t. Since the ratio inside the IDF’s log function has to be always greater than or equal to 1, so the value of IDF (and thus tf–idf) is greater than or equal to 0.When a term appears in large number of documents, the ratio inside the logarithm approaches 1, and the IDF is closer to 0. **Term Frequency-Inverse Document Frequency(TF-IDF)** TF-IDF is the product of TF and IDF. It is formulated as:

A high TF-IDF score is obtained by a term that has a high frequency in a document, and low document frequency in the corpus. For a word that appears in almost all documents, the IDF value approaches 0, making the tf-idf also come closer to 0.TF-IDF value is high when both IDF and TF values are high i.e the word is rare in the whole document but frequent in a document. Let’s take the same example to understand this better:

1. *good movie*
2. *not a good movie*
3. *did not like*

In this example, each sentence is a separate document. Considering the bigram model, we calculate the TF-IDF values for each bigram :

|  | **good movie** | **movie** | **did not** |
| --- | --- | --- | --- |
| **good movie** | 1\*log(3/2) = 0.17 | 1\*log(3/2) = 0.17 | 0\*log(3/1) = 0 |
| **not a good movie** | 1\*log(3/2) = 0.17 | 1\*log(3/2) = 0.17 | 0\*log(3/1) = 0 |
| **did not like** | 0\*log(3/2) = 0 | 0\*log(3/2) = 0 | 1\*log(3/1) = 0.47 |

Here, we observe that the bigram *did not* is rare(i.e. appears in only one document), as compared to other tokens, and thus has a higher tf-idf score. **Code : Using the python in-built function *TfidfVectorizer* to calculate tf-idf score for any corpus**

* Python3

|  |
| --- |
| # calculating tf-idf values  from sklearn.feature\_extraction.text import TfidfVectorizer  import pandas as pd   texts = {  "good movie", "not a good movie", "did not like"  }   tfidf = TfidfVectorizer(min\_df = 2, max\_df = 0.5, ngram\_range = (1, 2))  features = tfidf.fit\_transform(texts)   pd.Dataframe{       features.todense(),       columns = tfidf.get\_feature\_names()  } |

On a concluding note, we can say that though Bag-of-Words is one of the most fundamental methods in feature extraction and text vectorization, it fails to capture certain issues in the text. However, this problem is solved by TF-IDF Vectorizer, which also is a feature extraction method, that captures some of the major issues which are not too frequent in the entire corpus.

**Issues of TF-IDF :**

The following are some of the issues with using TF-IDF for text representation and analysis:

1. High dimensionality: The resulting feature space can be very high dimensional, which may lead to issues with overfitting and computational efficiency.
2. Lack of context information: TF-IDF only considers the frequency of words in a document, disregarding the context and meaning of words.
3. Domain dependence: The results of TF-IDF can be domain-specific, as the frequency and importance of words can vary greatly depending on the domain of the text.
4. Insensitivity to word associations: TF-IDF doesn’t consider the associations between words, and the semantic relationships between words in a document.
5. Lack of semantic information: As TF-IDF only considers individual words, it does not capture semantic relationships or the meaning of words in context.

Importance of stop words: Stop words, such as “the”, “and”, “a”, etc., can have a large impact on the TF-IDF representation of a document, even though they may not carry much meaning.

**Reference :**

Here are some references for further reading on the Bag of Words and TF-IDF models in natural language processing and information retrieval:

1. “Text Classification and Naive Bayes” by Debasis Das, Journal of Emerging Trends in Computing and Information Sciences
2. “A comparative study of feature selection and classification algorithms for sentiment analysis” by Esmaeilpour, M. and others, Journal of Ambient Intelligence and Humanized Computing
3. “Information Retrieval” by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, Cambridge University Press
4. “Natural Language Processing with Python” by Steven Bird, Ewan Klein, and Edward Loper, O’Reilly Media
5. “Speech and Language Processing” by Daniel Jurafsky and James H. Martin, Prentice Hall.

5. Describe the feature engineering process in the sense of a text categorization issue.

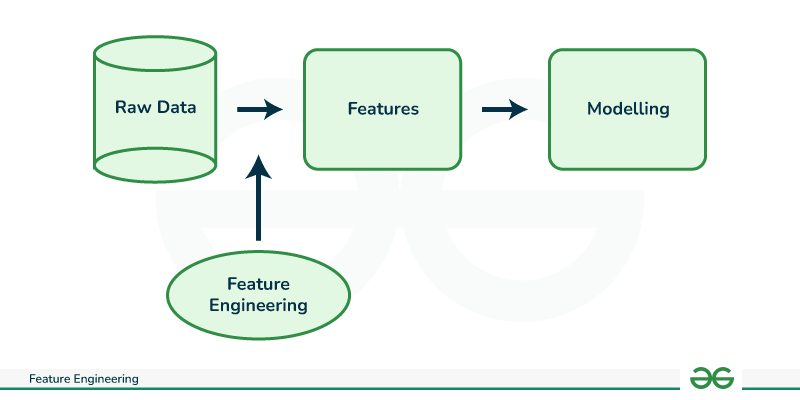
**What is Feature Engineering?**

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

**What is Feature Engineering?**

Feature engineering is the process of **transforming raw data into features that are suitable for machine learning models**. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.



**What is a Feature?**

In the context of machine learning, a feature (also known as a variable or attribute) is an individual measurable property or characteristic of a data point that is used as input for a machine learning algorithm. Features can be numerical, categorical, or text-based, and they represent different aspects of the data that are relevant to the problem at hand.

* For example, in a dataset of housing prices, features could include the number of bedrooms, the square footage, the location, and the age of the property. In a dataset of customer demographics, features could include age, gender, income level, and occupation.
* The choice and quality of features are critical in machine learning, as they can greatly impact the accuracy and performance of the model.

**Need for Feature Engineering in Machine Learning?**

We engineer features for various reasons, and some of the main reasons include:

* **Improve User Experience:** The primary reason we engineer features is to enhance the user experience of a product or service. By adding new features, we can make the product more intuitive, efficient, and user-friendly, which can increase user satisfaction and engagement.
* **Competitive Advantage:**Another reason we engineer features is to gain a competitive advantage in the marketplace. By offering unique and innovative features, we can differentiate our product from competitors and attract more customers.
* **Meet Customer Needs:**We engineer features to meet the evolving needs of customers. By analyzing user feedback, market trends, and customer behavior, we can identify areas where new features could enhance the product’s value and meet customer needs.
* **Increase Revenue:** Features can also be engineered to generate more revenue. For example, a new feature that streamlines the checkout process can increase sales, or a feature that provides additional functionality could lead to more upsells or cross-sells.
* **Future-Proofing:**Engineering features can also be done to future-proof a product or service. By anticipating future trends and potential customer needs, we can develop features that ensure the product remains relevant and useful in the long term.

**Processes Involved in Feature Engineering**

Feature engineering in Machine learning consists of mainly 5 processes: Feature Creation, Feature Transformation, Feature Extraction, Feature Selection, and Feature Scaling. It is an iterative process that requires experimentation and testing to find the best combination of features for a given problem. The success of a machine learning model largely depends on the quality of the features used in the model.

**1. Feature Creation**

Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

**Types of Feature Creation:**

1. **Domain-Specific:**Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.
2. **Data-Driven:**Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.
3. **Synthetic:**Generating new features by combining existing features or synthesizing new data points.

**Why Feature Creation?**

1. **Improves Model Performance:**By providing additional and more relevant information to the model, feature creation can increase the accuracy and precision of the model.
2. **Increases Model Robustness:**By adding additional features, the model can become more robust to outliers and other anomalies.
3. **Improves Model Interpretability:**By creating new features, it can be easier to understand the model’s predictions.
4. **Increases Model Flexibility:** By adding new features, the model can be made more flexible to handle different types of data.

**2. Feature Transformation**

[Feature Transformation](https://www.geeksforgeeks.org/feature-transformation-techniques-in-machine-learning/) is the process of transforming the features into a more suitable representation for the machine learning model. This is done to ensure that the model can effectively learn from the data.

**Types of Feature Transformation:**

1. [**Normalization**](https://www.geeksforgeeks.org/what-is-data-normalization/)**:**Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.
2. **Scaling:**Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.
3. **Encoding:**Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.
4. **Transformation:**Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

**Why Feature Transformation?**

1. **Improves Model Performance:**By transforming the features into a more suitable representation, the model can learn more meaningful patterns in the data.
2. **Increases Model Robustness:**Transforming the features can make the model more robust to outliers and other anomalies.
3. **Improves Computational Efficiency:**The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By transforming the features, it can be easier to understand the model’s predictions.

**3. Feature Extraction**

[Feature Extraction](https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/)is the process of creating new features from existing ones to provide more relevant information to the machine learning model. This is done by transforming, combining, or aggregating existing features.

**Types of Feature Extraction:**

1. **Dimensionality Reduction:** Reducing the number of features by transforming the data into a lower-dimensional space while retaining important information. Examples are [PCA](https://www.geeksforgeeks.org/principal-component-analysis-pca/)and[t-SNE](https://www.geeksforgeeks.org/ml-t-distributed-stochastic-neighbor-embedding-t-sne-algorithm/).
2. **Feature Combination:**Combining two or more existing features to create a new one. For example, the interaction between two features.
3. **Feature Aggregation:**Aggregating features to create a new one. For example, calculating the mean, sum, or count of a set of features.
4. **Feature Transformation:**Transforming existing features into a new representation. For example, log transformation of a feature with a skewed distribution.

**Why Feature Extraction?**

1. **Improves Model Performance:**By creating new and more relevant features, the model can learn more meaningful patterns in the data.
2. **Reduces Overfitting:** By reducing the dimensionality of the data, the model is less likely to overfit the training data.
3. **Improves Computational Efficiency:** The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By creating new features, it can be easier to understand the model’s predictions.

**4. Feature Selection**

[Feature Selection](https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/) is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model. It is an important step in the feature engineering process as it can have a significant impact on the model’s performance.

**Types of Feature Selection:**

1. **Filter Method:**Based on the statistical measure of the relationship between the feature and the target variable. Features with a high correlation are selected.
2. **Wrapper Method:**Based on the evaluation of the feature subset using a specific machine learning algorithm. The feature subset that results in the best performance is selected.
3. **Embedded Method:**Based on the feature selection as part of the training process of the machine learning algorithm.

**Why Feature Selection?**

1. **Reduces Overfitting:** By using only the most relevant features, the model can generalize better to new data.
2. **Improves Model Performance:** Selecting the right features can improve the accuracy, precision, and recall of the model.
3. **Decreases Computational Costs:**A smaller number of features requires less computation and storage resources.
4. **Improves Interpretability:** By reducing the number of features, it is easier to understand and interpret the results of the model.

**5. Feature Scaling**

[Feature Scaling](https://www.geeksforgeeks.org/ml-feature-scaling-part-1/) is the process of transforming the features so that they have a similar scale. This is important in machine learning because the scale of the features can affect the performance of the model.

**Types of Feature Scaling:**

1. [**Min-Max Scaling**](https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/)**:**Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.
2. **Standard Scaling:** Rescaling the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation.
3. **Robust Scaling:** Rescaling the features to be robust to outliers by dividing them by the interquartile range.

**Why Feature Scaling?**

1. **Improves Model Performance:**By transforming the features to have a similar scale, the model can learn from all features equally and avoid being dominated by a few large features.
2. **Increases Model Robustness:**By transforming the features to be robust to outliers, the model can become more robust to anomalies.
3. **Improves Computational Efficiency:**Many machine learning algorithms, such as k-nearest neighbors, are sensitive to the scale of the features and perform better with scaled features.
4. **Improves Model Interpretability:** By transforming the features to have a similar scale, it can be easier to understand the model’s predictions.

**What are the Steps in Feature Engineering?**

The steps for feature engineering vary per different Ml engineers and data scientists. Some of the common steps that are involved in most machine-learning algorithms are:

1. **Data Cleansing**
   * Data cleansing (also known as data cleaning or data scrubbing) involves identifying and removing or correcting any errors or inconsistencies in the dataset. This step is important to ensure that the data is accurate and reliable.
2. **Data Transformation**
3. **Feature Extraction**
4. **Feature Selection**
   * Feature selection involves selecting the most relevant features from the dataset for use in machine learning. This can include techniques like correlation analysis, mutual information, and stepwise regression.
5. **Feature Iteration**
   * Feature iteration involves refining and improving the features based on the performance of the machine learning model. This can include techniques like adding new features, removing redundant features and transforming features in different ways.

*Overall, the goal of feature engineering is to create a set of informative and relevant features that can be used to train a machine learning model and improve its accuracy and performance. The specific steps involved in the process may vary depending on the type of data and the specific machine-learning problem at hand.*

**Techniques Used in Feature Engineering**

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. There are various techniques that can be used in feature engineering to create new features by combining or transforming the existing ones. The following are some of the commonly used feature engineering techniques:

**One-Hot Encoding**

[One-hot encoding](https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/) is a technique used to transform categorical variables into numerical values that can be used by machine learning models. In this technique, each category is transformed into a binary value indicating its presence or absence. For example, consider a categorical variable “Colour” with three categories: Red, Green, and Blue. One-hot encoding would transform this variable into three binary variables: Colour\_Red, Colour\_Green, and Colour\_Blue, where the value of each variable would be 1 if the corresponding category is present and 0 otherwise.

**Binning**

[Binning](https://www.geeksforgeeks.org/binning-in-data-mining/) is a technique used to transform continuous variables into categorical variables. In this technique, the range of values of the continuous variable is divided into several bins, and each bin is assigned a categorical value. For example, consider a continuous variable “Age” with values ranging from 18 to 80. Binning would divide this variable into several age groups such as 18-25, 26-35, 36-50, and 51-80, and assign a categorical value to each age group.

**Scaling**

The most common scaling techniques are standardization and normalization. Standardization scales the variable so that it has zero mean and unit variance. Normalization scales the variable so that it has a range of values between 0 and 1.

**Feature Split**  
[Feature splitting](https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/) is a powerful technique used in feature engineering to improve the performance of machine learning models. It involves dividing single features into multiple sub-features or groups based on specific criteria. This process unlocks valuable insights and enhances the model’s ability to capture complex relationships and patterns within the data.

**Text Data Preprocessing**  
Text data requires special preprocessing techniques before it can be used by machine learning models. Text preprocessing involves removing stop words, stemming, lemmatization, and vectorization. Stop words are common words that do not add much meaning to the text, such as “the” and “and”. Stemming involves reducing words to their root form, such as converting “running” to “run”. Lemmatization is similar to stemming, but it reduces words to their base form, such as converting “running” to “run”. Vectorization involves transforming text data into numerical vectors that can be used by machine learning models.

**Feature Engineering Tools**

There are several tools available for feature engineering. Here are some popular ones:

**1. Featuretools**

Featuretools is a Python library that enables automatic feature engineering for structured data. It can extract features from multiple tables, including relational databases and CSV files, and generate new features based on user-defined primitives. Some of its features include:

* Automated feature engineering using machine learning algorithms.
* Support for handling time-dependent data.
* Integration with popular Python libraries, such as pandas and scikit-learn.
* Visualization tools for exploring and analyzing the generated features.
* Extensive documentation and tutorials for getting started.

**2. TPOT**

TPOT (Tree-based Pipeline Optimization Tool) is an automated machine learning tool that includes feature engineering as one of its components. It uses genetic programming to search for the best combination of features and machine learning algorithms for a given dataset. Some of its features include:

* Automatic feature selection and transformation.
* Support for multiple types of machine learning models, including regression, classification, and clustering.
* Ability to handle missing data and categorical variables.
* Integration with popular Python libraries, such as scikit-learn and pandas.
* Interactive visualization of the generated pipelines.

**3. DataRobot**

DataRobot is a machine learning automation platform that includes feature engineering as one of its capabilities. It uses automated machine learning techniques to generate new features and select the best combination of features and models for a given dataset. Some of its features include:

* Automatic feature engineering using machine learning algorithms.
* Support for handling time-dependent and text data.
* Integration with popular Python libraries, such as pandas and scikit-learn.
* Interactive visualization of the generated models and features.
* Collaboration tools for teams working on machine learning projects.

**4. Alteryx**

Alteryx is a data preparation and automation tool that includes feature engineering as one of its features. It provides a visual interface for creating data pipelines that can extract, transform, and generate features from multiple data sources. Some of its features include:

* Support for handling structured and unstructured data.
* Integration with popular data sources, such as Excel and databases.
* Pre-built tools for feature extraction and transformation.
* Support for custom scripting and code integration.
* Collaboration and sharing tools for teams working on data projects.

**5. H2O.ai**

H2O.ai is an open-source machine learning platform that includes feature engineering as one of its capabilities. It provides a range of automated feature engineering techniques, such as feature scaling, imputation, and encoding, as well as manual feature engineering capabilities for more advanced users. Some of its features include:

* Automatic and manual feature engineering options.
* Support for structured and unstructured data, including text and image data.
* Integration with popular data sources, such as CSV files and databases.
* Interactive visualization of the generated features and models.
* Collaboration and sharing tools for teams working on machine learning projects.

Overall, these tools can help streamline and automate the feature engineering process, making it easier and faster to create informative and relevant features for machine learning model

6. What makes cosine similarity a good metric for text categorization? A document-term matrix has two rows with values of (2, 3, 2, 0, 2, 3, 3, 0, 1) and (2, 1, 0, 0, 3, 2, 1, 3, 1). Find the resemblance in cosine.

**Cosine Similarity**

Prerequisite – [Measures of Distance in Data Mining](https://www.geeksforgeeks.org/measures-of-distance-in-data-mining/) In [Data Mining](https://www.geeksforgeeks.org/data-mining/), similarity measure refers to distance with dimensions representing features of the data object, in a dataset. If this distance is less, there will be a high degree of similarity, but when the distance is large, there will be a low degree of similarity. Some of the popular similarity measures are –

1. Euclidean Distance.
2. Manhattan Distance.
3. Jaccard Similarity.
4. Minkowski Distance.
5. Cosine Similarity.

**Cosine similarity** is a metric, helpful in determining, how similar the data objects are irrespective of their size. We can measure the [similarity between two sentences in Python](https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/) using Cosine Similarity. In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine similarity between two vectors is –

(x, y) = x . y / ||x|| ||y||

where,

* **x . y** = product (dot) of the vectors ‘x’ and ‘y’.
* **||x||**and**||y||** = length (magnitude) of the two vectors ‘x’ and ‘y’.
* **||x||  ||y||** = regular product of the two vectors ‘x’ and ‘y’.

**Example :**Consider an example to find the similarity between two vectors – **‘x’** and**‘y’**, using Cosine Similarity. The ‘x’ vector has values, **x = { 3, 2, 0, 5 }** The ‘y’ vector has values, **y = { 1, 0, 0, 0 }** The formula for calculating the cosine similarity is : **(x, y) = x . y / ||x||  ||y||**

x . y = 3\*1 + 2\*0 + 0\*0 + 5\*0 = 3

||x|| = √ (3)^2 + (2)^2 + (0)^2 + (5)^2 = 6.16

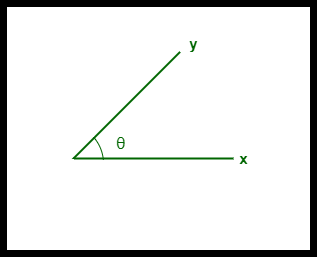
||y|| = √ (1)^2 + (0)^2 + (0)^2 + (0)^2 = 1

∴ (x, y) = 3 / (6.16 \* 1) = 0.49

The dissimilarity between the two vectors ‘x’ and ‘y’ is given by –

∴ (x, y) = 1 - (x, y) = 1 - 0.49 = 0.51

* The cosine similarity between two vectors is measured in ‘θ’.
* If θ = 0°, the ‘x’ and ‘y’ vectors overlap, thus proving they are similar.
* If θ = 90°, the ‘x’ and ‘y’ vectors are dissimilar.



*Cosine Similarity between two vectors*

**Advantages :**

* The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
* When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

7.

i. What is the formula for calculating Hamming distance? Between 10001011 and 11001111, calculate the Hamming gap.

**The Hamming distance between the two strings is 4.**

**Given:**

First value = 11111000

Second value = 10000111

**To Find:**

The hamming distance between 11111000 and 10000111

**Solution:**

A measurement of the difference among two varied strings of identical length is the Hamming distance.

It is calculated by keeping track of how many different places there are between corresponding items in the strings.

Calculating the Hamming distance between two given values by  comparing each corresponding position:

1 1 1 1 1 0 0 0

1 0 0 0 0 1 1 1

There are in total four positions where the elements differ with respect to positions of 0 and 1.

**Answer: The Hamming distance between the two strings is 4.**

#SPJ1

ii. Compare the Jaccard index and similarity matching coefficient of two features with values (1, 1, 0, 0, 1, 0, 1, 1) and (1, 1, 0, 0, 0, 1, 1, 1), respectively (1, 0, 0, 1, 1, 0, 0, 1).

# Install packages qvalue and jaccard and load

# the library

library(qvalue)

library(jaccard)

# Binary vectors A and B depicting purchase of

# items by customers

Binary\_A <- c(0,1,0,0,0,1,0,0,1,1)

Binary\_B <- c(0,0,1,0,0,0,0,0,1,1)

# Computing jaccard similarity between 2 binary

# vectors A and B

jaccard(Binary\_A,Binary\_B)

# Computing jaccard distance between 2 binary

# vectors A and B

Jaccard\_distance <- 1 - jaccard(Binary\_A,Binary\_B)

Jaccard\_distance

1. State what is meant by "high-dimensional data set"? Could you offer a few real-life examples? What are the difficulties in using machine learning techniques on a data set with many dimensions? What can be done about it?

We have discussed so far that high-dimensional data analysis can be challenging since variables are difficult to visualise, leading to challenges identifying relationships between variables and suitable response variables; we may have relatively few observations compared to features, leading to over-fitting; and features may be highly correlated, leading to challenges interpreting models. We therefore require alternative approaches to examine whether, for example, groups of observations show similar characteristics and whether these groups may relate to other features in the data (e.g. phenotype in genetics data).

In this course, we will cover four methods that help in dealing with high-dimensional data: (1) regression with numerous outcome variables, (2) regularised regression, (3) dimensionality reduction, and (4) clustering. Here are some examples of when each of these approaches may be used:

1. Regression with numerous outcomes refers to situations in which there are many variables of a similar kind (expression values for many genes, methylation levels for many sites in the genome) and when one is interested in assessing whether these variables are associated with a specific covariate of interest, such as experimental condition or age. In this case, multiple univariate regression models (one per each outcome, using the covariate of interest as predictor) could be fitted independently. In the context of high-dimensional molecular data, a typical example are differential gene expression analyses. We will explore this type of analysis in the Regression with many outcomes episode.
2. Regularisation (also known as regularised regression or penalised regression) is typically used to fit regression models when there is a single outcome variable or interest but the number of potential predictors is large, e.g. there are more predictors than observations. Regularisation can help to prevent overfitting and may be used to identify a small subset of predictors that are associated with the outcome of interest. For example, regularised regression has been often used when building epigenetic clocks, where methylation values across several thousands of genomic sites are used to predict chronological age. We will explore this in more detail in the Regularised regression episode.
3. Dimensionality reduction is commonly used on high-dimensional datasets for data exploration or as a preprocessing step prior to other downstream analyses. For instance, a low-dimensional visualisation of a gene expression dataset may be used to inform quality control steps (e.g. are there any anomalous samples?). This course contains two episodes that explore dimensionality reduction techniques: Principal component analysis and Factor analysis.
4. Clustering methods can be used to identify potential grouping patterns within a dataset. A popular example is the identification of distinct cell types through clustering cells with similar gene expression patterns. The K-means episode will explore a specific method to perform clustering analysis.

8. Make a few quick notes on:

1. PCA is an acronym for Personal Computer Analysis.

**What Is Principal Component Analysis?**

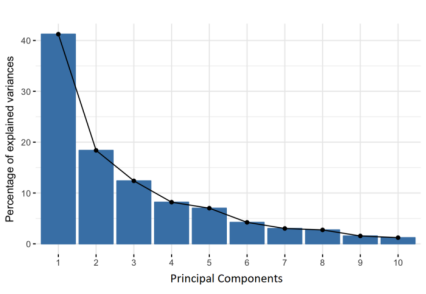
Principal component analysis, or PCA, is a [dimensionality reduction](https://builtin.com/data-science/dimensionality-reduction-python) method that is often used to reduce the dimensionality of large [data sets](https://builtin.com/data-science), by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize, and thus make analyzing data points much easier and faster for [machine learning algorithms](https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies) without extraneous variables to process.

So, to sum up, the idea of PCA is simple: **reduce the number of variables of a data set, while preserving as much information as possible.**

**What Are Principal Components?**

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the scree plot below.

Percentage of Variance (Information) for each by PC.

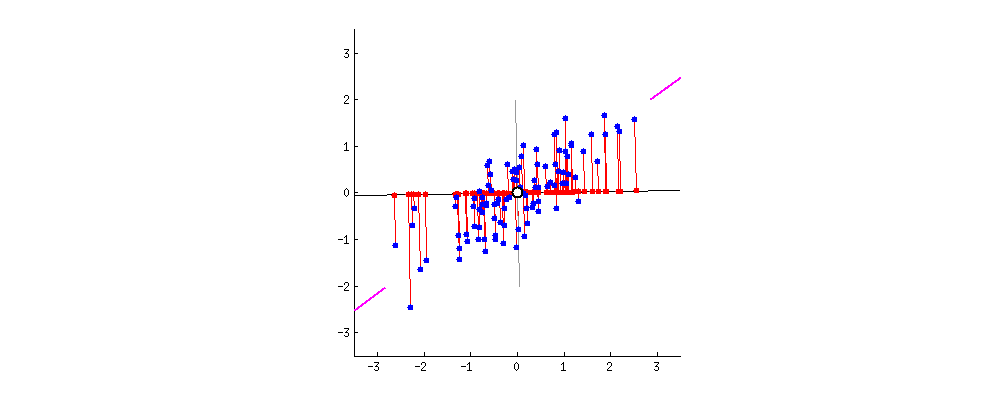
Organizing information in principal components this way will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

An important thing to realize here is that the principal components are less interpretable and don’t have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

**How PCA Constructs the Principal Components**

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the **largest possible variance** in the data set. For example, let’s assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it’s approximately the line that matches the purple marks because it goes through the origin and it’s the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it’s the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

This continues until a total of *p* principal components have been calculated, equal to the original number of variables.

**Step-by-Step Explanation of PCA**

**Step 1: Standardization**

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (for example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

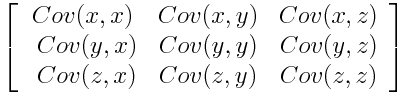
Principal Component Analysis Standardization

Once the standardization is done, all the variables will be transformed to the same scale.

**Step 2: Covariance Matrix Computation**

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the [covariance matrix](https://builtin.com/data-science/mahalanobis-distance).

The covariance matrix is a *p* × *p*symmetric matrix (where *p*is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables *x*, *y*, and *z*, the covariance matrix is a 3×3 data matrix of this from:

Covariance Matrix for 3-Dimensional Data.

Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

**What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?**

It’s actually the sign of the covariance that matters:

* If positive then: the two variables increase or decrease together (correlated)
* If negative then: one increases when the other decreases (Inversely correlated)

Now that we know that the covariance matrix is not more than a table that summarizes the correlations between all the possible pairs of variables, let’s move to the next step.

**Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components**

Eigenvectors and eigenvalues are the [linear algebra](https://builtin.com/data-science/basic-linear-algebra-deep-learning) concepts that we need to compute from the covariance matrix in order to determine the ***principal components*** of the data.

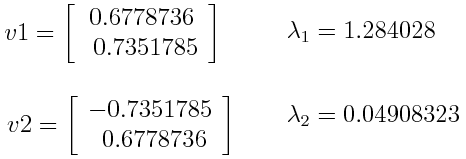
What you first need to know about eigenvectors and eigenvalues is that they always come in pairs, so that every eigenvector has an eigenvalue. Also, their number is equal to the number of dimensions of the data. For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.

It is eigenvectors and eigenvalues who are behind all the magic of principal components because the eigenvectors of the Covariance matrix are actually *the**directions of the axes where there is the most variance*(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the *amount of variance carried in each Principal Component*.

By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

**Principal Component Analysis Example:**

Let’s suppose that our data set is 2-dimensional with 2 variables ***x,y***and that the eigenvectors and eigenvalues of the covariance matrix are as follows:



If we rank the eigenvalues in descending order, we get λ1>λ2, which means that the eigenvector that corresponds to the first principal component (PC1) is *v1*and the one that corresponds to the second principal component (PC2) is *v2.*

After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96 percent and 4 percent of the variance of the data.

**Step 4: Create a Feature Vector**

As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step, what we do is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

So, the [feature vector](https://builtin.com/machine-learning/siamese-network) is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

**Principal Component Analysis Example**:

Continuing with the example from the previous step, we can either form a feature vector with both of the eigenvectors *v*1 and *v*2:

Principal Component Analysis eigen vectors

Or discard the eigenvector *v*2, which is the one of lesser significance, and form a feature vector with *v*1 only:

Principal Component Analysis eigen vectors 2

Discarding the eigenvector *v2*will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that *v*2 was carrying only 4 percent of the information, the loss will be therefore not important and we will still have 96 percent of the information that is carried by *v*1.

So, as we saw in the example, it’s up to you to choose whether to keep all the components or discard the ones of lesser significance, depending on what you are looking for. Because if you just want to describe your data in terms of new variables (principal components) that are uncorrelated without seeking to reduce dimensionality, leaving out lesser significant components is not needed.

**Step 5: Recast the Data Along the Principal Components Axes**

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

Principal Component Analysis feature vector

1. Use of vectors

## ****Practical Application of Vector in Everyday Life****

### ****1. MILITARY USAGE****

Any piece of artillery that fires a projectile by employing gun power or any other type of typically explosive-based propellant is considered to be a cannon. The calibre, range, mobility, rate of fire, and angle of fire of cannons all differ from one another. Depending on the role that each type of cannon is supposed to play on the battlefield, different types of cannon mix and balance these characteristics to differing degrees. It is required to make use of this vector. Vectors decide where the projectile will head and hit on the ground.

### ****2. PROJECTILE****

The baseball vector is utilized automatically by players in sports like basketball and baseball. In the end, students either shot the target or threw the ball in a direction at an angle, both of which were accomplished by using their understanding of vectors. In fact, in games like Javelin throw, it is necessary to judge the angle that the projectile vector makes with the ground so that the javelin can travel as far as possible.

### ****3. VECTOR IN GAMING****

Vectors are utilized in the storage of locations, directions, and velocities in video games. The position vector tells us how far away the object is, the velocity vector tells us how long time it will take or how much force we need to apply, and the direction vector tells us how we should apply that force.

### ****4. ROLLER COASTER****

The majority of the motion that occurs during a roller coaster ride is a reaction to the gravitational pull that the earth exerts. When a train hits a high point and subsequently rides downhill, it will have gained sufficient speed to be affected by gravitational attraction. At this rate, it will reach the top of the next hill. This process repeats again until all of the train’s energy has been dissipated due to friction. However, the train would be able to continue operating so long as there was no point on the track that was higher than the initial peak. It is essential to the construction of the safety system that vectors of forces, acceleration, and velocity be considered. This employs the use of vectors in designing the roller coaster ride.

### ****5. WHEN PLAYING CRICKET****

 If a batsman in cricket hits a lob shot, there are three possible outcomes: catch out, drop before the fielder, or maximum score. When the batsman hits the ball, the angle at which he shoots and the amount of velocity with which he hits the ball are both important factors in determining whether or not the ball goes over the boundary. If both of these factors meet the requirements for maximum force, the ball goes over the boundary. Because of the ball, everything revolves around the vector.

### ****6. BOAT CROSSING A RIVER****

When a boat travels over a river, the speed of the boat and the speed of the river both contribute to the total speed of the boat. When the current speed of the river changes, so does the course that the boat takes. Therefore, the boatman must determine an angle for crossing the river in order to access the shore of the river in a direct manner. Vector plays an important role here.

### ****7. CROSSWIND****

The concept of a crosswind is one that is familiar to us. A wind that blows in a direction that is perpendicular to the path that one is travelling is referred to as a crosswind. When a plane finally touches down, it will at times experience challenges due to the crosswind. With the use of a vector, a pilot is able to determine the resultant velocity as well as the direction.

1. Embedded technique

**What are Embedding in Machine Learning?**

In recent years, embeddings have emerged as a core idea in machine learning, revolutionizing the way we represent and understand data. In this article, we delve into the world of embeddings, exploring their importance, applications, and the underlying techniques used to generate them.

**Table of Content**

* [What are Embedding?](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#what-are-embedding)
* [Key terms used for Embedding](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#key-terms-used-for-embedding)
* [Why Embedding is so important?](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#why-embedding-is-so-important)
* [What Object can be embedded?](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#what-object-can-be-embedded)
* [How do embeddings work?](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#how-do-embeddings-work)
* [Visualization of Word Embeddings using t-SNE](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#visualization-of-word-embeddings-using-tsne)
* [Frequently Asked Questions on Embedding](https://www.geeksforgeeks.org/what-are-embeddings-in-machine-learning/#frequently-asked-questions-on-embedding)

**What are Embedding?**

**Embedding can be defined as the mathematical representation of discrete objects or values as dense vectors within a continuous vector space.**

These objects can vary widely, including words, paragraphs, documents, images, audio, and more. The key idea behind embeddings is to encode semantic and contextual information in a compact and meaningful way, allowing machine learning algorithms to effectively analyze and understand the data.

Embedding vectors are generated using [machine learning](https://www.geeksforgeeks.org/machine-learning-algorithms/) techniques like [**neural network**](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/)-based training on large datasets to learn the relationships between words or other objects and represent them as **dense vectors** in a **continuous vector space**.

In[**natural language processing**](https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/) tasks, for instance, words are embedded into **dense vectors**, with similarities between **vectors** indicating semantic similarities between the corresponding words. Similarly, **embedding** images and audio allows for the extraction and representation of significant features and relationships within these data modalities.

For example, A simple word **embedding** graph is shown below, generated using [**Word2Vec**](https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/) to obtain the word embeddings. To visualize these embeddings in 2D plots, [t-SNE](https://www.geeksforgeeks.org/ml-t-distributed-stochastic-neighbor-embedding-t-sne-algorithm/) (t-distributed Stochastic Neighbor Embedding) has been employed to reduce the dimensionality of the embedding vectors.

In the above graph, we observe distinct clusters of related words. For instance, “computer,” “software,” and “machine” are clustered together, indicating their semantic similarity. Similarly, “lion,” “cow,” “cat,” and “dog” form another cluster, representing their shared attributes. Notably, there exists a significant gap between these clusters, highlighting their dissimilarity in meaning or context.

**Key terms used for Embedding**

Now, let’s understand the key terms one by one, which we have frequently used in above definintions of embedding.

**Vector**

* **Definitions**: A vector is a mathematical object that represents a quantity with both magnitude and direction. In machine learning, a vector typically refers to an ordered set of numerical values representing a data point or features in a multi-dimensional space.
* **Example**: In a 2D space, a vector [3, 4] represents a quantity with magnitude 5 (from the Pythagorean theorem) and direction, where the x-component is 3 and the y-component is 4.

**Dense Vector**

* **Definition**: A dense vector is a type of vector where most of its elements are non-zero.  
  In the context of embeddings, dense vectors are used to represent discrete objects or values within a continuous multi-dimensional vector space.  
  Dense vectors contain information about the attributes or features of the represented objects, and they are often utilized in machine learning tasks for their ability to capture intricate relationships and patterns within data.
* **Example**: Consider a dense vector [2000, 3, 5, 9.8] representing features of a house, where each element represents a different attribute such as size in square feet (2000), number of bedrooms (3), number of bathrooms (5), and age of the house in years (9.8).

**Vector space**

* **Definition**: A vector space, or linear space, is a mathematical structure consisting of a set of vectors that can be added together and multiplied by scalars, satisfying certain properties.  
  It satisfy the certain properties like:
  + **Closure under addition:** The sum of any two vectors within the space results in another vector within the same space.
  + **Scalar multiplication:** Multiplying a vector by a scalar yields another vector within the same space.
* **Example**: The set of all 3D vectors with real-number coordinates forms a vector space. For example, the vectors [1, 0, 0], [0, 1, 0], and [0, 0, 1] constitute a basis for the 3D vector space.

**Continuous Vector space**

* **Definition**: A continuous vector space or continuous multi-dimensional vector space is a vector space where each vector represents an object or value with multiple attributes, and these attributes can take on continuous values.  
  In the context of embeddings, objects or values are mapped to dense vectors within this continuous multi-dimensional vector space.  
  Continuous vector spaces enable the representation of complex data structures and relationships, allowing machine learning algorithms to analyze and extract meaningful patterns from the data.
* **Example**: Consider a continuous vector space representing colors in the RGB (Red, Green, Blue) color model. In this space, each vector corresponds to a color, and its attributes represent the intensity values for the red, green, and blue channels. For instance, The vector [0.9, 0.3, 0.1] represents a shade of red, with higher intensity in the red channel, some intensity in the green channel, and the least intensity in the blue channel.

**Why Embedding is so important?**

Embeddings are used across various domains and tasks for several reasons:

1. **Semantic Representation:** Embeddings capture semantic relationships between entities in the data. For example, in word embeddings, words with similar meanings are mapped to nearby points in the vector space. This semantic representation enables models to understand and reason about the underlying concepts in the data.
2. **Dimensionality Reduction:** Embeddings reduce the dimensionality of data while preserving important features and relationships. This is crucial for processing large datasets efficiently and for tasks where high-dimensional data is problematic.
3. **Generalization:**Embeddings generalize well to unseen data. Models trained using embeddings can leverage the semantic similarities encoded in the embeddings to make predictions on new, unseen examples, even if they were not present in the training data.
4. **Transfer Learning:** Embeddings learned from one task or domain can be transferred and fine-tuned for use in related tasks or domains. This allows leveraging knowledge gained from one dataset to improve performance on another, potentially smaller dataset.
5. **Efficient Computations:** Embeddings enable efficient computations by representing data in a compact, dense format. This is particularly important for machine learning models, as it reduces the computational complexity of training and inference processes.
6. **Feature Engineering:** Embeddings automatically extract meaningful features from raw data, reducing the need for manual feature engineering. This is particularly advantageous for tasks where handcrafted features may be difficult to define or time-consuming to create.
7. **Interpretability**: In some cases, embeddings provide interpretable representations of data. For example, in word embeddings, the direction and distance between word vectors can correspond to meaningful relationships, such as gender, tense, or sentiment.

Overall, embeddings offer a powerful framework for representing and processing data in various domains, leading to improved performance, efficiency, and generalization capabilities in machine learning and artificial intelligence applications

**What Object can be embedded?**

From textual data to images and beyond, embeddings offer a versatile approach to encoding information into dense vector representations. Some of the major types of objects or values that can be embedded include:

**1. Words**

[Word embedding](https://www.geeksforgeeks.org/word-embeddings-in-nlp/) are numerical representations of words in a continuous vector space, where words with similar meanings or contexts are mapped to nearby points. These representations capture semantic relationships between words and are learned from large amounts of text data using techniques like neural networks. Word embeddings enable computers to process and understand natural language more effectively by transforming words into dense, low-dimensional vectors that encode semantic information. They have become a fundamental tool in natural language processing tasks such as sentiment analysis, machine translation, and document classification.

**Some of the Popular word embeddings include:**

* [Word2Vec](https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/)
* [GloVe (Global Vectors for Word Representation)](https://www.geeksforgeeks.org/pre-trained-word-embedding-using-glove-in-nlp-models/)
* [FastText](https://www.geeksforgeeks.org/fasttext-working-and-implementation/)
* [BERT (Bidirectional Encoder Representations from Transformers)](https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/)
* GPT

**2. Complete Text Document**

Text embeddings, also known as document embeddings or document representations, extend the concept of word embeddings to represent entire units of text, such as sentences, paragraphs, or documents, in a continuous vector space. Unlike word embeddings, which represent individual words, text embeddings capture the semantic meaning and contextual information of longer segments of text. They encode semantic meaning and context, unlike word embeddings which focus on individual words. Used in NLP tasks such as sentiment analysis and machine translation, text embeddings capture the essence of text in fixed-size vectors, facilitating efficient processing and comparison of textual data.

**Some of the Popular text embedding models include:**

* [Doc2Vec](https://www.geeksforgeeks.org/doc2vec-in-nlp/)
* [Universal Sentence Encoder (USE)](https://www.geeksforgeeks.org/word-embedding-using-universal-sentence-encoder-in-python/)
* BERT
* [ELMO](https://www.geeksforgeeks.org/what-are-some-key-strengths-of-bert-over-elmo-ulmfit/)

**3. Audio Data**

[Audio](https://www.geeksforgeeks.org/audio-processing-with-transformer/) data presents a diverse set of objects that can be embedded, including individual sound samples, audio clips, and entire audio recordings. By representing audio as dense vectors in a continuous vector space, embedding techniques effectively capture acoustic features and relationships. This enables a wide range of audio processing tasks, such as speech recognition, speaker identification, emotion detection, and music genre classification.

**Some of the popular Audio embedding techniques may include:**

* VGGish
* OpenL3
* [Wav2Vec](https://www.geeksforgeeks.org/wav2vec2-self-a-supervised-learning-technique-for-speech-representations/)

**4. Image Data**

Image embeddings are numerical representations of images in a continuous vector space, extracted by processing images through [convolutional neural networks (CNNs)](https://www.geeksforgeeks.org/introduction-convolution-neural-network/" \t "_blank). These embeddings encode the visual content, features, and semantics of images, facilitating efficient understanding and processing of visual information by machines. They capture semantic meaning, object presence, and spatial relationships within images, enabling tasks such as image classification, object detection, similarity search, and content-based image retrieval.

**Some of the popular CNNs based Image embedding techniques include:**

* [VGG](https://www.geeksforgeeks.org/vgg-16-cnn-model/)
* [ResNet](https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/)
* [Inception](https://www.geeksforgeeks.org/ml-inception-network-v1/)
* EfficientNet

**5. Graph Data**

Graph embedding refers to the process of transforming the nodes and edges of a graph into numerical vectors in a continuous vector space. These embeddings capture the structural and relational information of the graph, allowing complex graph data to be represented in a format suitable for machine learning algorithms. Graph embedding techniques enable various graph-based tasks, such as node classification, link prediction, and graph clustering, by encoding the topological properties and semantic relationships within the graph into vector representations.

**Some popular graph embedding techniques include:**

* [Node2Vec](https://www.geeksforgeeks.org/node2vec-algorithm/)
* [DeepWalk](https://www.geeksforgeeks.org/deepwalk-algorithm/)
* GraphSAGE (Graph Sample and Aggregation)
* LINE (Large-scale Information Network Embedding)
* Graph Convolutional Networks (GCNs)
* TADW (Topological Attributed Deep Walk)

These techniques are widely used in various applications such as social network analysis, recommendation systems, biological network analysis, and link prediction.

**6. Structured Data**

Structured data, including feature vectors and tabular data, can be embedded to capture complex relationships and patterns. This conversion enables machine learning models to process structured data more effectively. Techniques include Entity Embeddings, which map categorical variables to dense vector representations, and [Autoencoders](https://www.geeksforgeeks.org/auto-encoders/" \t "_blank), which learn compressed representations of structured data through unsupervised learning. These embeddings facilitate tasks like regression, classification, and clustering on structured datasets.

**How do embeddings work?**

Embeddings work by transforming high-dimensional and sparse data into dense, low-dimensional representations in a continuous vector space. These representations capture meaningful relationships and patterns in the data, making it easier for machine learning algorithms to process, analyze, and learn from the data effectively.

The process of generating embeddings varies depending on the type of data being used, but here, we are defining a general overview of how to create an embeddings work :

1. **Define the Embedding Space:**  
   Before generating embeddings, it’s necessary to establish the embedding space, which refers to the dimensionality of the continuous vector space. This dimensionality of the embedding space is a hyperparameter that needs to be chosen based on the characteristics of the data and the requirements of the task.
2. **Learn Embeddings:**  
   Embeddings are learned using neural networks approach. The specific approach depends on the type of data being used:  
   * **For textual data:** Word embeddings are learned by training neural network models on large text corpora. These models, such as Word2Vec, GloVe, FastText, or BERT, learn to predict words based on their context or to capture co-occurrence statistics of words in the corpus. The weights of the neural network, which represent the learned embeddings, are then used as the word embeddings.
   * **For image data:** Image embeddings are learned by training convolutional neural networks (CNNs) on large image datasets, such as ImageNet. CNNs learn to extract meaningful visual features from images, and the output of intermediate layers or the final layer of the network can be used as image embeddings.
   * **For audio data:** Audio embeddings can be learned using neural network models trained on spectrograms or raw audio waveforms. These models, such as VGGish, OpenL3, or Wav2Vec, learn to capture acoustic features and relationships in the audio data, producing embeddings that represent the content of the audio.
   * **For graph data:** Graph embeddings are learned using techniques such as Node2Vec, DeepWalk, or Graph Convolutional Networks (GCNs). These techniques learn to encode the structural and relational information of the graph into vector representations by considering the connectivity patterns and properties of the nodes and edges.
   * **For Structured Data:** Embeddings for structured data involve mapping categorical variables or feature vectors into dense, low-dimensional representations. Techniques such as Entity Embeddings or Autoencoders are commonly used for this purpose. Entity Embeddings transform categorical variables into continuous vectors, capturing relationships between different categories. Autoencoders learn compressed representations of structured data through unsupervised learning.
3. **Optimize Embeddings:**During the training process, embeddings are optimized to minimize a loss function that measures the discrepancy between the predicted outputs (e.g., word predictions, image classifications) and the ground truth labels or targets. This optimization process adjusts the embedding vectors to capture meaningful patterns and relationships in the data, making them more suitable for the intended task.
4. **Apply Embeddings:** Once the embeddings are learned, they can be applied to various machine learning tasks, such as classification, clustering, similarity search, recommendation systems, or information retrieval. In these tasks, embeddings are used as input features to machine learning models or algorithms, enabling them to operate more effectively in the embedding space and leverage the captured patterns and relationships in the data.

**Visualization of Word Embeddings using t-SNE**

Visualizing word embeddings can provide insights into how words are positioned relative to each other in a high-dimensional space. In this code, we demonstrate how to visualize word embeddings using t-SNE (t-distributed Stochastic Neighbor Embedding), a technique for dimensionality reduction, after training a Word2Vec model on the ‘text8’ corpus.

**Code Steps:**

1. Import necessary libraries.
2. Load the ‘text8’ corpus.
3. Train a Word2Vec model on the corpus.
4. Define sample words for visualization.
5. Filter words existing in the model’s vocabulary.
6. Retrieve word embeddings for sample words.
7. Convert embeddings to a numpy array.
8. Print original embedding vector shape.
9. Use t-SNE to reduce embeddings to 2D.
10. Print the shape of reduced embeddings.
11. Plot word embeddings using Matplotlib.
12. Set plot attributes.
13. Save the plot as an image file.
14. Display the plot.

Python3

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn.manifold** **import** TSNE

**import** **gensim.downloader** **as** **api**

**from** **gensim.models** **import** Word2Vec

*# Load the text8 corpus from gensim*

corpus = api.load('text8')

*# Train a Word2Vec model on the text8 corpus*

model = Word2Vec(corpus)

*# Sample words for visualization*

words = ['cat', 'dog', 'elephant', 'lion', 'bird', 'rat', 'wolf', 'cow',

'goat', 'snake', 'rabbit', 'human', 'parrot', 'fox', 'peacock',

'lotus', 'roses', 'marigold', 'jasmine', 'computer', 'robot',

'software', 'vocabulary', 'machine', 'eye', 'vision',

'grammar', 'words', 'sentences', 'language', 'verbs', 'noun',

'transformer', 'embedding', 'neural', 'network', 'optimization']

*# Filter words that exist in the model's vocabulary*

words = [word **for** word **in** words **if** word **in** model.wv.key\_to\_index]

*# Get word embeddings for sample words from the pre-trained model*

word\_embeddings = [model.wv[word] **for** word **in** words]

*# Convert word embeddings to a numpy array*

embeddings = np.array(word\_embeddings)

*# Print original embedding vector shape*

print('Original embedding vector shape', embeddings.shape)

*# Use t-SNE to reduce dimensionality to 2D with reduced perplexity*

tsne = TSNE(n\_components=2, perplexity=2) *# Reduced perplexity value*

embeddings\_2d = tsne.fit\_transform(embeddings)

*# Print the shape of the embeddings after applying t-SNE*

print('After applying t-SNE, embedding vector shape', embeddings\_2d.shape)

*# Plot the word embedding graph*

*# Set figure size and DPI for high-resolution output*

plt.figure(figsize=(10, 7), dpi=1000)

plt.scatter(embeddings\_2d[:, 0], embeddings\_2d[:, 1], marker='o')

*# Add labels to data points*

**for** i, word **in** enumerate(words):

plt.text(embeddings\_2d[i, 0], embeddings\_2d[i, 1], word,

fontsize=10, ha='left', va='bottom') *# Adjust text placement for better readability*

plt.xlabel('t-SNE Dimension 1')

plt.ylabel('t-SNE Dimension 2')

plt.title('Word Embedding Graph (t-SNE with Word2Vec)')

plt.grid(**True**)

plt.savefig('embedding.png') *# Save the plot as an image file*

plt.show()

**Output:**

Original embedding vector shape (37, 100)  
After applying tsne embedding vector shape (37, 2)

**Conclusion**

Embeddings have revolutionized machine learning by providing compact, dense representations of data across various domains. From capturing semantic relationships in natural language to extracting features from images and audio, embeddings play a crucial role across diverse domains. Their ability to generalize, transfer knowledge, and facilitate efficient computations makes them indispensable in modern [AI](https://www.geeksforgeeks.org/ai-algorithms/) applications. With embeddings, we unlock new possibilities for understanding and processing data, driving innovation and advancement in AI technology.

10. Make a comparison between:

* 1. Sequential backward exclusion vs. sequential forward selection

**equential Feature Selection**

Last Updated : 06 Sep, 2023

Feature selection is a process of identifying and selecting the most relevant features from a dataset for a particular predictive modeling task. This can be done for a variety of reasons, such as to improve the predictive accuracy of a model, to reduce the computational complexity of a model, or to make a model more interpretable. This article focuses on a sequential feature selector, which is one such feature selection technique.

Sequential feature selection (SFS) is a greedy algorithm that iteratively adds or removes features from a dataset in order to improve the performance of a predictive model. SFS can be either forward selection or backward selection.

**Sequential Feature Selector**

SequentialFeatureSelector class in Scikit-learn supports both forward and backward selection. The SequentialFeatureSelector class in scikit-learn works by iteratively adding or removing features from a dataset in order to improve the performance of a predictive model. The process is as follows:

1. The selector is initialized with a predictive model, the number of features to select, the scoring metric, and the tolerance for improvement.
2. The selector fits the predictive model on the full set of features.
3. The model is evaluated on the training set using the scoring metric.
4. The feature that most improve the model’s cross-validation score is added to the selected features set, or the feature that least reduces the model’s cross-validation score is removed from the selected features set, whichever one gives the greatest improvement in the scoring metric.
5. The selector repeats steps 2-4 until the desired number of features has been selected.

The process is reversed if the selector is doing backward selection. During backward selection, selector starts with the entire set of features and iteratively removes the feature that has the least impact on the predictive model’s performance. The process is repeated until the required number of features is chosen or until no additional features can be eliminated without significantly decreasing the model’s performance.

The required number of features can be specified via the n\_features\_to\_select argument, which specifies the number of features to select, or the tol parameter, which specifies the tolerance for improvement. The selector will only add or remove a feature if it improves the scoring metric by at least tol.

**Code implementation**

* Python3

|  |
| --- |
| #Code for demostrating use of SFS on iris data. written by Tapendra Kumar    from sklearn.datasets import load\_iris  from sklearn.linear\_model import LogisticRegression  from sklearn.feature\_selection import SequentialFeatureSelector    iris = load\_iris(as\_frame=True)  X = iris.data  y = iris.target    # Create a logistic regression model  logreg = LogisticRegression()    # Create a sequential feature selector  selector = SequentialFeatureSelector(      logreg, n\_features\_to\_select=2, scoring='accuracy')    # Fit the selector to the data  selector.fit(X, y)    # Get the selected features  selected\_features = selector.get\_support()    print('The selected features are:', list(X.columns[selected\_features])) |

**Output :**

The selected features are: ['petal length (cm)', 'petal width (cm)']

**Advantages and Disadvantages**

The advantages of sequential feature selection include:

* It is a simple and efficient algorithm.
* It can be used with any type of predictive model.
* It can be used to select features for both classification and regression tasks.

The disadvantages of sequential feature selection include:

* It can be sensitive to the choice of the scoring metric.
* It can be biased towards features that are highly correlated with the target feature.
* It can be computationally expensive for large datasets.

**Conclusion**

Sequential feature selection is a powerful tool that can be used to improve the performance of predictive models. However, it is important to be aware of its limitations and to use it appropriately.

1. Function selection methods: filter vs. wrapper

What is the difference between filter, wrapper, and embedded methods for feature selection?

Wrapper methods measure the “usefulness” of features based on the classifier performance. In contrast, the filter methods pick up the intrinsic properties of the features (i.e., the “relevance” of the features) measured via univariate statistics instead of cross-validation performance. So, wrapper methods are essentially solving the “real” problem (optimizing the classifier performance), but they are also computationally more expensive compared to filter methods due to the repeated learning steps and cross-validation. The third class, embedded methods, are quite similar to wrapper methods since they are also used to optimize the objective function or performance of a learning algorithm or model. The difference to wrapper methods is that an intrinsic model building metric is used during learning. Let me give you a – off the top of my head – list of examples from these three categories.

Filter methods:

* information gain
* chi-square test
* fisher score
* correlation coefficient
* variance threshold

Wrapper methods:

* recursive feature elimination
* sequential feature selection algorithms
* genetic algorithms

Embedded methods:

* L1 (LASSO) regularization
* decision tree

(Note that I would count transformation and projection techniques such as Principal Component Analysis as a feature *extraction* approach, since we are projecting the data into a new feature space.) To give you a more hands-on illustration, let me pick one algorithm from each category and explain w

**1). A Filter method Example: Variance Thresholds**

Here, we simply compute the variance of each feature, and we select the subset of features based on a user-specified threshold. E.g., “keep all features that have a variance greater or equal to *x*” or “keep the the top *k* features with the largest variance.” We assume that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature variables or feature and target variables into account, which is one of the drawbacks of filter methods.

**2). A Wrapper Method Example: Sequential Feature Selection**

Sequential Forward Selection (SFS), a special case of sequential feature selection, is a greedy search algorithm that attempts to find the “optimal” feature subset by iteratively selecting features based on the classifier performance. We start with an empty feature subset and add one feature at the time in each round; this one feature is selected from the pool of all features that are not in our feature subset, and it is the feature that – when added – results in the best classifier performance. Since we have to train and cross-validate our model for each feature subset combination, this approach is much more expensive than a filter approach such as the variance threshold, which we discussed above.

**3). An Embedded Method Example: L1 Regularization**

L1 (or LASSO) regression for generalized linear models can be understood as adding a penalty against complexity to reduce the degree of overfitting or variance of a model by adding more bias. Here, we add a penalty term directly to the cost function,

regularized\_cost = cost + regularization\_penalty

In L1 regularization, the penalty term is

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L1 : λ Σki | wi | = λ | **w** | 1, |

where **w** is our *k-dimensional* feature vector. Through adding the L1 term, our objective function now becomes the minimization of the regularized cost, and since the penalty term grows with the value of the weight parameters (λ is just a free parameter to fine-tune the regularization strength), we can induce sparsity through this L1 vector norm, which can be considered as an intrinsic way of feature selection that is part of the model training step.

1. SMC vs. Jaccard coefficient

he **simple matching coefficient (SMC)** or **Rand similarity coefficient** is a [statistic](https://en.wikipedia.org/wiki/Statistic) used for comparing the [similarity](https://en.wikipedia.org/wiki/Similarity_measure) and [diversity](https://en.wikipedia.org/wiki/Diversity_index) of [sample](https://en.wikipedia.org/wiki/Sample_(statistics)) sets.[[1]](https://en.wikipedia.org/wiki/Simple_matching_coefficient#cite_note-1)[[*better source needed*](https://en.wikipedia.org/wiki/Wikipedia:NOTRS)]

|  |  |  |  |
| --- | --- | --- | --- |
|  | | ***A*** | |
| **0** | **1** |
| ***B*** | **0** | 𝑀00 | 𝑀10 |
| **1** | 𝑀01 | 𝑀11 |

Given two objects, A and B, each with *n* binary attributes, SMC is defined as:

SMC=number of matching attributestotal number of attributes=𝑀00+𝑀11𝑀00+𝑀11+𝑀01+𝑀10

where

* 𝑀00 is the total number of attributes where *A* and *B* both have a value of 0,
* 𝑀11 is the total number of attributes where *A* and *B* both have a value of 1,
* 𝑀01 is the total number of attributes where *A* has value 0 and *B* has value 1, and
* 𝑀10 is the total number of attributes where *A* has value 1 and *B* has value 0.

The **simple matching distance (SMD)**, which measures dissimilarity between sample sets, is given by 1−SMC.[[2]](https://en.wikipedia.org/wiki/Simple_matching_coefficient#cite_note-2)[[*better source needed*](https://en.wikipedia.org/wiki/Wikipedia:NOTRS)]

SMC is linearly related to Hamann similarity: SMC=(Hamann+1)/2. Also, SMC=1−𝐷2/𝑛, where 𝐷2 is the squared Euclidean distance between the two objects (binary vectors) and *n* is the number of attributes.

The SMC is very similar to the more popular [Jaccard index](https://en.wikipedia.org/wiki/Jaccard_index" \o "Jaccard index). The main difference is that the SMC has the term 𝑀00 in its numerator and denominator, whereas the Jaccard index does not. Thus, the SMC counts both mutual presences (when an attribute is present in both sets) and mutual absence (when an attribute is absent in both sets) as matches and compares it to the total number of attributes in the universe, whereas the Jaccard index only counts mutual presence as matches and compares it to the number of attributes that have been chosen by at least one of the two sets.

In market basket analysis, for example, the basket of two consumers who we wish to compare might only contain a small fraction of all the available products in the store, so the SMC will usually return very high values of similarities even when the baskets bear very little resemblance, thus making the Jaccard index a more appropriate measure of similarity in that context. For example, consider a supermarket with 1000 products and two customers. The basket of the first customer contains salt and pepper and the basket of the second contains salt and sugar. In this scenario, the similarity between the two baskets as measured by the Jaccard index would be 1/3, but the similarity becomes 0.998 using the SMC.

In other contexts, where 0 and 1 carry equivalent information (symmetry), the SMC is a better measure of similarity. For example, vectors of demographic variables stored in [dummy variables](https://en.wikipedia.org/wiki/Dummy_variable_(statistics)), such as binary gender, would be better compared with the SMC than with the Jaccard index since the impact of gender on similarity should be equal, independently of whether male is defined as a 0 and female as a 1 or the other way around. However, when we have symmetric dummy variables, one could replicate the behaviour of the SMC by splitting the dummies into two binary attributes (in this case, male and female), thus transforming them into asymmetric attributes, allowing the use of the Jaccard index without introducing any bias. By using this trick, the Jaccard index can be considered as making the SMC a fully redundant metric. The SMC remains, however, more computationally efficient in the case of symmetric dummy variables since it does not require adding extra dimensions.

The Jaccard index is also more general than the SMC and can be used to compare other data types than just vectors of binary attributes, such as [probability measures](https://en.wikipedia.org/wiki/Probability_measure).