Machine Learning

An Engineer’s Handbook

The Sarasvati Project

PUBLISHED BY PUBLISHER

BOOK-WEBSITE.COM

First printing, March 2022

Contents

List of Figures

List of Tables

1. Dimentionality Reduction

1.1 What is Dimensionality Reduction?

Dimensionality reduction is a technique used to reduce the number of features in a dataset while retaining as much of the important information as possible. In other words, it is a process of transforming high-dimensional data into a lower-dimensional space that still preserves the essence of the original data.

In machine learning, high-dimensional data refers to data with a large number of features or variables. The curse of dimensionality is a common problem in machine learning, where the performance of the model deteriorates as the number of features increases. This is because the complexity of the model increases with the number of features, and it becomes more difficult to find a good solution. In addition, high-dimensional data can also lead to overfitting, where the model fits the training data too closely and does not generalize well to new data.

Dimensionality reduction can help to mitigate these problems by reducing the complexity of the model and improving its generalization performance. There are two main approaches to dimensionality reduction: feature selection and feature extraction.

1.1.1 Feature Selection

Feature selection involves selecting a subset of the original features that are most relevant to the problem at hand. The goal is to reduce the dimensionality of the dataset while retaining the most important features. There are several methods for feature selection, including filter methods, wrapper methods, and embedded methods. Filter methods rank the features based on their relevance to the target variable, wrapper methods use the model performance as the criteria for selecting features, and embedded methods combine feature selection with the model training process.

1.1.2 Feature Extraction

Feature extraction involves creating new features by combining or transforming the original features. The goal is to create a set of features that captures the essence of the original data in a lower-dimensional space. There are several methods for feature extraction, including principal component analysis (PCA), linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE). PCA is a popular technique that projects the original features onto a lower-dimensional space while preserving as much of the variance as possible.

1.1.2.1 Principal Component Analysis

PCA is an unsupervised method of dimensionality reduction that aims to find the directions of maximum variance in a dataset. The idea is to find a smaller set of variables or features that capture the most important patterns in the data. PCA is often used to preprocess data for machine learning algorithms. PCA works by first centering the data around its mean and then finding the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the directions of maximum variance, while the eigenvalues represent the amount of variance explained by each eigenvector. The eigenvectors are then used to project the data onto a lower-dimensional space. The number of principal components to keep is determined by the amount of variance we want to retain. Typically, we keep the principal components that explain a certain percentage of the total variance in the data. After a certain number, the variance explained by the PCA components, cumulatively, does not increase much with the increasing components.

1.1.2.2 Linear Discriminant Analysis

LDA is a supervised method of dimensionality reduction that aims to find the linear combination of features that best separates the classes in a dataset. The idea is to reduce the dimensionality of the data while preserving the information that is most relevant for class discrimination. LDA works by first calculating the mean and covariance matrix for each class in the data. It then calculates the between-class scatter matrix and the within-class scatter matrix. The goal is to find a projection that maximizes the ratio of the between-class scatter matrix to the within-class scatter matrix. This projection is the linear discriminant function. Once the linear discriminant function is found, we can project the data onto this function to obtain the reduced representation of the data. The resulting transformed data can be used for classification.

1.1.2.3 PCA vs LDA — When to use what?

PCA and LDA are both powerful techniques for dimensionality reduction, but they have different objectives and assumptions. One of the main differences is in their objectives. PCA aims to find the directions of maximum variance in the data, while LDA aims to find the projection that best separates the classes in the data. Another difference is in their assumptions. PCA is an unsupervised method that does not take into account the class labels in the data. LDA, on the other hand, is a supervised method that assumes that the data is normally distributed and that the covariance matrices for each class are equal. Finally, PCA is often used for exploratory data analysis and preprocessing of data for machine learning algorithms, while LDA is often used for classification and feature selection.

2. Model Basics

2.1 Bias

Bias is simply defined as the inability of the model because of that there is some difference or error occurring between the model's predicted value and the actual value. These differences between actual or expected values and the predicted values are known as error or bias error or error due to bias. Bias is a systematic error that occurs due to wrong assumptions in the machine learning process.

- Low Bias: Low bias value means fewer assumptions are taken to build the target function. In this case, the model will closely match the training dataset.
- High Bias: High bias value means more assumptions are taken to build the target function. In this case, the model will not match the training dataset closely.

The high-bias model will not be able to capture the dataset trend. It is considered as the underfitting model which has a high error rate. It is due to a very simplified algorithm.

2.2 Variance

Variance is the measure of spread in data from its mean position. In machine learning variance is the amount by which the performance of a predictive model changes when it is trained on different subsets of the training data. More specifically, variance is the variability of the model that how much it is sensitive to another subset of the training dataset. i.e. how much it can adjust on the new subset of the training dataset.

- Low variance: Low variance means that the model is less sensitive to changes in the training data and can produce consistent estimates of the target function with different subsets of data from the same distribution. This is the case of underfitting when the model fails to generalize on both training and test data.
- High variance: High variance means that the model is very sensitive to changes in the training data and can result in significant changes in the estimate of the target function when trained on different subsets of data from the same distribution. This is the case of overfitting when the model performs well on the training data but poorly on new, unseen test data. It fits the training data too closely that it fails on the new training dataset.

2.3 Different Combinations of Bias-Variance

1. High Bias, Low Variance: A model with high bias and low variance is said to be underfitting.

2. High Variance, Low Bias: A model with high variance and low bias is said to be overfitting.
3. High-Bias, High-Variance: A model has both high bias and high variance, which means that the model is not able to capture the underlying patterns in the data (high bias) and is also too sensitive to changes in the training data (high variance). As a result, the model will produce inconsistent and inaccurate predictions on average.
4. Low Bias, Low Variance: A model that has low bias and low variance means that the model is able to capture the underlying patterns in the data (low bias) and is not too sensitive to changes in the training data (low variance). This is the ideal scenario for a machine learning model, as it is able to generalize well to new, unseen data and produce consistent and accurate predictions. But in practice, it's not possible.

Now we know that the ideal case will be Low Bias and Low variance, but in practice, it is not possible. So, we trade off between Bias and variance to achieve a balanced bias and variance

2.4 Bias Variance Tradeoff

If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance.

2.5 Overfitting

Overfitting (also known as high-variance) is the process where the models perform very well on the training data but not on the evaluation/testing data. This happens because in the training stage the model memorizes the data that is present, but is incapable to generalize unseen examples. When the training algorithm runs on the dataset, through multiple iterations we permit the minimization of overall distance, but if it runs for long it will lead to the minimal overall distance. If it reaches minimal distance, can also be said as cost, the line we get will fit all the points even grabbing the secondary patterns, noise as well that may not be required for the model generalization.

2.6 Underfitting

Underfitting occurs when the model could not learn enough patterns, and performs poorly on the training data because it was unable to capture the dominant trend. This case is called underfitting. The model's poor performance on training data happens because the model is too simple (the input features are not expressive enough) to narrate or describe the target well.

2.7 Reduce Overfitting

1. Data Augmentation:
   - Artificially increase the size and diversity of your training data by applying transformations like flipping, rotating, zooming, or adding noise to your existing data points. This helps the model learn generalizable features instead of memorizing specific details.
2. Regularization:
   - Penalizes complex models, encouraging them to fit the data without memorizing noise. Common techniques include L1 (Lasso) and L2 (Ridge) regularization, which add the sum of absolute or squared feature weights to the cost function.
3. Early Stopping:
   - Monitor the model's performance on a separate validation set during training. Stop training before the validation error starts increasing, even if the training error keeps

decreasing. This prevents the model from memorizing the training data at the expense of generalizability.

4. Dropout:
   - Randomly drop out neurons during training, forcing the model to rely on other features and combinations instead of memorizing specific connections. This encourages robustness and prevents overfitting in neural networks.
5. Feature Selection:
   - Identify and remove irrelevant or redundant features that contribute little to the model's predictions but increase its complexity. This can be done through techniques like correlation analysis or feature importance scores.
6. Model Complexity Control:
   - Choose a model with the right level of complexity for your problem. Using a model that is too complex for the data is more prone to overfitting. Consider simpler models like linear regression or decision trees for smaller datasets or less complex relationships.
7. Ensembling:
   - Combine predictions from multiple, diverse models (e.g., bagging, boosting) to reduce variance and improve generalization. This can often outperform individual models and be less susceptible to overfitting.
8. Data Quality and Preprocessing:
   - Ensure your data is clean and properly preprocessed before training. Missing values, outliers, and inconsistencies can lead to overfitting. Standardize features or scale them to similar ranges to prevent some features from dominating the model.

2.8 Reduce Underfitting

1. Increase model complexity:
   - Increase the number of features: Add relevant features that were not considered initially. This could involve feature engineering to create new features from existing ones.
   - Choose a more complex model: For nonlinear relationships, consider switching to models like decision trees, random forests, or neural networks, which have higher expressive power.
2. Increase the amount of training data:
   - Collect more data, ensuring it is representative of the overall population you want to make predictions for.
   - Augment your existing data through techniques like oversampling or data synthesis to enrich your training set.
3. Reduce regularization:
   - Regularization penalties can bias models towards simpler solutions to prevent overfitting. Relaxing the regularization strength (e.g., hyperparameter tuning) can allow the model to capture more complex patterns.
4. Train for longer:
   - Underfitting can sometimes be due to insufficient training time. Experiment with longer training epochs, but be careful not to overfit.
5. Address data quality issues:
   - Ensure your data is clean and preprocessed properly. Missing values, outliers, and irrelevant features can hinder model performance.
6. Use ensemble methods:
   - Combining multiple simpler models like decision trees through techniques like bagging or boosting can often lead to better generalization and reduced underfitting.

2.9 Evalutaion Metric

2.9.1 For Classifiation

2.9.1.1 Accuracy

- The most basic metric, measuring the proportion of correct predictions made by the model.
- Calculated as: $(TruePositives + TrueNegatives)/TotalPredictions$
- It's easy to understand, but can be misleading in imbalanced datasets.

2.9.1.2 Precision

- The proportion of true positives among the predicted positives.
- It measures how accurate the model is when it predicts a positive class.
- Calculated as: $TruePositives/(TruePositives + FalsePositives)$
- High precision means fewer false positives, crucial for avoiding unnecessary costs or actions.

2.9.1.3 Recall (Sensitivity)

- The proportion of true positives correctly identified by the model.
- It measures how well the model captures all actual positive cases.
- Calculated as: $TruePositives/(TruePositives + FalseNegatives)$
- High recall is essential in critical tasks like disease detection or fraud prevention.

2.9.1.4 F1-Score

- The harmonic mean of precision and recall, balancing both aspects of performance.
- It provides a single value for evaluating models when both precision and recall are important.
- Calculated as: $2 * (Precision * Recall)/(Precision + Recall)$

2.9.1.5 Confusion Matrix

- A table visualizing the distribution of predictions (true positives, true negatives, false positives, false negatives).
- It provides a detailed breakdown of model performance across different classes.

2.9.1.6 ROC Curve (Receiver Operating Characteristic)

- What it is:
  - A plot visualizing the model's performance at various decision thresholds.
  - It shows the trade-off between true positive rate (recall) and false positive rate (1 - specificity).
- How it's created:
  - The model assigns a probability or score to each data point, indicating its likelihood of belonging to the positive class.
  - Various decision thresholds are applied to these scores.
  - For each threshold, TPR and FPR are calculated based on the resulting true positives, false positives, true negatives, and false negatives.
  - The ROC curve is then plotted by plotting TPR (y-axis) against FPR (x-axis) for all possible thresholds.
- Interpreting the ROC curve:
  - Good performance: A curve that climbs steeply towards the top-left corner (high TPR, low FPR) indicates better performance.
  - Diagonal line: A diagonal line (AUC-ROC = 0.5) represents random guessing.
  - Area under the curve (AUC-ROC): A single value summarizing the curve's overall performance. Higher AUC-ROC values (closer to 1) indicate better discrimination between classes.

2.9.1.7 AUC-ROC (Area Under the ROC Curve)

- A single value summarizing the ROC curve, representing the model's overall ability to distinguish between classes.
- A higher AUC-ROC (closer to 1) indicates better performance.

2.9.1.8 Choosing the Right Metrics

- Problem Context: Consider the specific goals of your problem. Do you prioritize identifying all positive cases (high recall) or minimizing false positives (high precision)?
- Data Imbalance: If classes are imbalanced, accuracy can be misleading. Use precision, recall, F1-score, and AUC-ROC for better evaluation.
- Visualization: Use confusion matrices and ROC curves to visualize model performance and identify areas for improvement.

2.9.2 For Regression

2.9.2.1 Mean Squared Error (MSE)

- Measures the average squared difference between the predicted and actual values.
- Lower MSE indicates better fit.
- Sensitive to outliers due to squaring.

2.9.2.2 Root Mean Squared Error (RMSE)

- The square root of MSE, providing the error in the same units as the target variable.
- More interpretable than MSE as it measures average error magnitude.

2.9.2.3 Mean Absolute Error (MAE)

- Measures the average absolute difference between predicted and actual values.
- Less sensitive to outliers than MSE/RMSE.

2.9.2.4 R-squared (R^2)

- Represents the proportion of variance in the target variable explained by the model.
- Values range from 0 to 1, with higher values indicating better fit.
- However, be cautious of overfitting, as complex models can have high R^2 without generalizing well.

2.9.2.5 Adjusted R-squared

- A modified version of R^2 that penalizes for the number of features in the model.
- It prevents overestimation of model fit in complex models with many features.

3. Ensemble Methods

3.1 Ensemble Methods

Ensemble Learning is a machine learning technique where the predictions from various predictors, such as classifiers or regressors, are combined by aggregating the predictions of a set of models to produce outcomes that are superior to those of any individual predictor. The group of predictors is known as an ensemble, where their combined predictions help to improve performance. and the overall algorithm of this ensemble learning is known as an ensemble method. The basic principle of ensemble learning is to combine a number of weak learners into strong learners. There are two main types of ensemble methods:

1. Bagging (Bootstrap Aggregating): In bagging, multiple models are trained on different random subsets of the training data with replacement. The average or majority vote of all the distinct models' predictions is used to determine the final prediction. Examples of bagging methods are Bagged Decision Trees, Extra Trees, and Random Forests.
2. Boosting: In boosting, multiple models are trained sequentially, with each one aiming to fix the mistakes caused by the one before it by assigning higher weights to the misclassified instances and adjusting them during training. The final prediction is produced by weighting each individual model's prediction according to how accurate it was. Examples of Boosting method is AdaBoost, Gradient Boosting, and XGBoost.
3. Stacking (Stacked Generalization): In stacking, The predictions of various models are combined using a meta-model. During stacking, various base models are trained on training data, and their predictions are then used as input features for a more sophisticated model, known as a blender or meta-model. The meta-model develops the ability to synthesize the basic models' predictions to arrive at the ultimate conclusion. Stacking enables more complex interactions among the base models and provides enhanced performance.

3.2 Similarities Between Bagging and Boosting

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

3.3 Differences Between Bagging and Boosting

Bagging	Boosting
The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
Each model receives equal weight.	Models are weighted according to their performance.
Each model is built independently.	New models are influenced by the performance of previously built models.
Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.

4. Regularization

4.1 What is regularization?

Regularization means restricting a model to avoid overfitting by shrinking the coefficient estimates to zero. When a model suffers from overfitting, we should control the model's complexity. Technically, regularization avoids overfitting by adding a penalty to the model's loss function

4.2 Regularization Techniques

4.2.1 Ridge Regularization or L2 Regularization

A linear regression that uses the L2 regularization technique is called ridge regression. In other words, in ridge regression, a regularization term is added to the cost function of the linear regression, which keeps the magnitude of the model's weights (coefficients) as small as possible. The L2 regularization technique tries to keep the model's weights close to zero, but not zero, which means each feature should have a low impact on the output while the model's accuracy should be as high as possible.

$$\text{Ridge Regression Cost Function} = \text{Loss Function} + \frac{1}{2}\lambda \textstyle\sum_{j=1}^{m} w_j^2$$

5. Everything About Regression

5.1 what is regression?

1. A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables.
2. A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.
3. It does this by essentially fitting a best-fit line and seeing how the data is dispersed around this line.
4. Regression captures the correlation between variables observed in a data set and quantifies whether those correlations are statistically significant or not.

There are 2 types of Regression:

1. Simple linear regression: Simple linear regression uses one independent variable to explain or predict the outcome of the dependent variable Y.
2. Multiple linear regression: Multiple linear regression uses two or more independent variables to predict the outcome (while holding all others constant).

5.2 Calculating Regression

Linear regression models often use a least-squares approach to determine the line of best fit. The least-squares technique is determined by minimizing the sum of squares created by a mathematical function. A square is, in turn, determined by squaring the distance between a data point and the regression line or mean value of the data set. Once this process has been completed (usually done today with software), a regression model is constructed. The general form of each type of regression model is:

- Simple linear regression: $Y = a + bX + u$
- Multiple linear regression: $Y = a + b_1X_1 + ... + b_nX_n + u$
- where:
  - Y=The dependent variable you are trying to predict or explain
  - X=The explanatory independent variables you are using to predict or associate with Y
  - a=The y-intercept
  - b=beta coefficient is the slope of the explanatory variables
  - u=The regression residual or error term

5.3 Assumptions of regression models

In order to properly interpret the output of a regression model, the following main assumptions about the underlying data process of what you analyzing must hold:

- The relationship between variables is linear
- Homoskedasticity, or that the variance of the variables and error term must remain constant
- All explanatory variables are independent of one another
- All variables are normally-distributed. Normal distribution, also known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean.

6. Everything About Descision Trees

6.1 What is a Descision Tree?

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems.

6.2 What Is the Structure of a Decision Tree?

A tree consists of 2 major components:

- Decision node: the point where you make a decision
- Leaf node: the output of said decision; it does not contain any further branches

The algorithm starts from the first decision node, known as the root node. It represents the entire dataset, which is further divided into 2 or more homogeneous sets. The decision nodes represent the dataset's features, branches denote the decision rules, and each leaf node signifies the outcome.

6.3 Decision Tree Terminologies

1. Root Node: It is the topmost node in the tree, which represents the complete dataset. It is the starting point of the decision-making process. The feature with the highest inforation gain is choses as a root node.
2. Decision/Internal Node: A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
3. Leaf/Terminal Node: A node without any child nodes that indicates a class label or a numerical value.
4. Pure Node: Pure nodes are those that have one class — hence the term pure.
5. Splitting: The process of splitting a node into two or more sub-nodes using a split criterion and a selected feature.
6. Branch/Sub-Tree: A subsection of the decision tree starts at an internal node and ends at the leaf nodes.
7. Parent Node: The node that divides into one or more child nodes.
8. Child Node: The nodes that emerge when a parent node is split.
9. Impurity: A measurement of the target variable's homogeneity in a subset of data. It refers to the degree of randomness or uncertainty in a set of examples. The Gini index and entropy

are two commonly used impurity measurements in decision trees for classifications task

10. Variance: Variance measures how much the predicted and the target variables vary in different samples of a dataset. It is used for regression problems in decision trees. Mean squared error, Mean Absolute Error, friedman mse, or Half Poisson deviance are used to measure the variance for the regression tasks in the decision tree.
11. Information Gain: Information gain is a measure of the reduction in impurity achieved by splitting a dataset on a particular feature in a decision tree. The splitting criterion is determined by the feature that offers the greatest information gain, It is used to determine the most informative feature to split on at each node of the tree, with the goal of creating pure subsets
12. Pruning: The process of removing branches from the tree that do not provide any additional information or lead to overfitting.

6.4 Attribute Selection

6.4.1 Entropy and Information Gain

Here's a simple explanation of entropy in a decision tree, using a fruit basket analogy:

Imagine a basket full of mixed fruits:

1. If it's all apples, there's no surprise when you pick one—it's always an apple. This is like a node with low entropy (more purity).
2. If it's half apples and half oranges, there's more uncertainty—you might get either fruit. This is like a node with higher entropy (less purity).

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content. Entropy can be though of as a measure of disorder or impurity in a node. Information Gain = $Entropy_{Parent} - Entropy_{Children}$ Decision trees use entropy to make choices:

1. They try to split nodes into groups that are more "pure," like sorting the fruits into separate baskets.
2. A node with high entropy (mixed fruits) is a good candidate for splitting, as it can lead to more certainty in the branches.

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

- Suppose S is a set of instances,
- A is an attribute
- S_v is the subset of S
- v represents an individual value that the attribute A can take and Values (A) is the set of all possible values of A, then
- Refer Formula Sheet.

6.4.2 Gini Index

The Gini Index is a measure of the inequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithms. It ranges from 0 to 1, where 0 represents perfect equality (all values are the same) and 1 represents perfect inequality (all values are different). In other words, Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with a lower Gini index should be preferred. Some additional features and characteristics of the Gini Index are:

- It is calculated by summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.
- A lower Gini Index indicates a more homogeneous or pure distribution, while a higher Gini Index indicates a more heterogeneous or impure distribution.

- In decision trees, the Gini Index is used to evaluate the quality of a split by measuring the difference between the impurity of the parent node and the weighted impurity of the child nodes.
- Compared to other impurity measures like entropy, the Gini Index is faster to compute and more sensitive to changes in class probabilities.
- One disadvantage of the Gini Index is that it tends to favor splits that create equally sized child nodes, even if they are not optimal for classification accuracy.
- In practice, the choice between using the Gini Index or other impurity measures depends on the specific problem and dataset, and often requires experimentation and tuning.

6.4.3 Difference between Gini Index and Entropy

Gini Index	Entropy
It is the probability of misclassifying a randomly chosen element in a set.	While entropy measures the amount of uncertainty or randomness in a set.
The range of the Gini index is [0, 1], where 0 indicates perfect purity and 1 indicates maximum impurity.	The range of entropy is [0, log(c)], where c is the number of classes.
Gini index is a linear measure.	Entropy is a logarithmic measure.
It can be interpreted as the expected error rate in a classifier.	It can be interpreted as the average amount of information needed to specify the class of an instance.
It is sensitive to the distribution of classes in a set.	It is sensitive to the number of classes.
It is less robust than entropy.	It is more robust than Gini index.
It is sensitive.	It is comparatively less sensitive.
It has a bias toward selecting splits that result in a more balanced distribution of classes.	It has a bias toward selecting splits that result in a higher reduction of uncertainty.
Gini index is typically used in CART (Classification and Regression Trees) algorithms	Entropy is typically used in ID3 and C4.5 algorithms.

6.5 Hyperparameters

1. Max Depth: The name of hyperparameter max depth is suggested the maximum depth that we allow the tree to grow to. The deeper you allow, the more complex our model will become; in the case of training error, it is easy to see what will happen. If we increase the max depth value, training error will always go down but let us consider a case for testing error; if we set the max depth too high, then without capturing the useful patterns, the decision tree might simply overfit the training data, this will cause the testing error to increase, but if we set the value of it to too low that is also not good then the decision tree might be having little flexibility to capture the patterns and interaction in the training data. This will also cause the testing error to increase, which is a case of underfitting so we have to find the right max depth using hyperparameter tuning either grid search or at the best possible values of max depth the random search might arrive.
2. Min Samples Split: In a decision tree, there are multiple nodes, some of which are internal nodes, while others are called leaf nodes. Internal nodes can further split into child nodes. The min samples split hyperparameter specifies the minimal number of samples required to divide a node. This hyperparameter can accept various values: you can specify an integer

to denote the minimum number of samples required in an internal node, or you can use a fraction to denote the minimum percentage of samples required in an internal node.

3. Min Samples Leaf: Let us see another parameter which is called min samples leaf, as we have already seen a leaf node is a node without any children, so we cannot split a leaf node any further, so min samples leaf is the minimum number of samples that we can specify to term a given node as a leaf node so that we do not want to split it further. For example, we start off with 10,000 samples, and we reach a node wherein we just have 100 samples; there is no point in splitting further as you would tend to overfit the training data that you have, so by using this hyperparameter smartly, we can also avoid overfitting, and this parameter is similar to min samples splits. However, this describes the minimum number of samples at the leaf that is a base of the tree.
4. Min Weight Fraction Leaf: This is also another type of decision tree hyperparameter, which is called min weight fraction, it is the fraction of the input samples that are required at the leaf node where sample weight determined weight, in this way, we can deal with class unbalancing, and the class unbalancing can be done by sampling an equal number of samples from each class. Also, when we biased it towards dominant classes then that are not aware of the sample weights like min sample leaf, by using weight-based criteria, it is easier to optimize the structure of the tree if the samples are weighted, min weight fraction leaf, in which the leaf nodes contain at least a fraction of the overall sum of the weights.
5. Class Weight: This is also a hyperparameter called class weight in which weight associated with classes or it is used to provide weight for each output class; this actually means when the algorithm calculates impurity to make the split at each node, then the resulting child node are weighted by class weight by giving the child sample weight, distribution of our classes has been start then the weight of class and then depending on where our tree lean, we can try to increase the weight of the other class so that algorithm penalizes the sample of one class relative to the other.

6.6 Advantages

- Decision trees are easy to understand. Because of their structure, which follows the natural flow of human thought, most people will have little trouble interpreting them. In addition, visualizing the model is effortless and allows you to see exactly what decisions are being made.
- There is little to no need for data preprocessing. Unlike other algorithms, decision trees take less time to model as they require less coding, analysis, or even dummy variables. The reason is that the technique looks at each data point individually instead of the set as a whole.
- Versatile when it comes to data. In other words, standardizing the collected data is not a necessity. You can imbue both numerical and categorical data into the model as it's able to work with features of both types.

6.7 Disadvantages

- There is a tendency to overfit. Essentially, the model performs so well on the training data that it compromises the decision-making process. You can prevent this by either stopping the decision tree before it has a chance to do so or, alternatively, letting it grow and then pruning the decision tree after overfitting occurs.
- Mathematical equations are more costly. Not only does the decision tree require more time to calculate, but it also consumes more memory. This is not ideal as sometimes you will have to work with substantial amounts of data and stricter deadlines , efficiency is of the essence.
- Decision trees can be unstable. For example, a minor modification of the data can lead to significant changes – perhaps even generating a new tree with contrary results. Another

instance is the model producing biased decisions if some of the classes dominate over the rest.

7. Everything About K Means

7.1 What is K-means Clustering?

Unsupervised Machine Learning is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine's job in this case is to organize unsorted data according to parallels, patterns, and variations.

7.2 What is the objective of k-means clustering?

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

7.3 How k-means clustering works?

We are given a data set of items, with certain features, and values for these features like a vector. The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm, an unsupervised learning algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

It will help if you think of items as points in an n-dimensional space. The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean, and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The 'points' mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x, the items have values in [0,3], we will initialize the means with values for x at [0,3]).

The above algorithm in pseudocode is as follows:

- Initialize k means with random values
- For a given number of iterations:
  - Iterate through items:
    - Find the mean closest to the item by calculating the euclidean distance of the item with each of the means
    - Assign item to mean
    - Update mean by shifting it to the average of the items in that cluster

7.4 Choosing K

If the true label is not known in advance, then K-Means clustering can be evaluated using Elbow Criterion , Silhouette Coefficient , cross-validation, information criteria, the information theoretic jump method, and the G-means algorithm. .

7.4.1 Elbow Criterion Method

The idea behind elbow method is to run k-means clustering on a given dataset for a range of values of k (e.g k=1 to 10), for each value of k, calculate sum of squared errors (SSE).

Calculate the mean distance between data points and their cluster centroid. Increasing the number of clusters(K) will always reduce the distance to data points, thus decrease this metric, to the extreme of reaching zero when K is as same as the number of data points. So the goal is to choose a small value of k that still has a low SSE.

We run the algorithm for different values of K(say K = 10 to 1) and plot the K values against SSE(Sum of Squared Errors). And select the value of K for the elbow point.

7.4.2 Silhouette Coefficient Method

A higher Silhouette Coefficient score relates to a model with better-defined clusters. The Silhouette Coefficient is defined for each sample and is composed of two scores:

- The mean distance between a sample and all other points in the same class.
- The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient is for a single sample is then given as:

To find the optimal value of k for KMeans, loop through 1 to n for $n_{clusters}$ in KMeans and calculate Silhouette Coefficient for each sample.

A higher Silhouette Coefficient indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

7.5 Advantages and Disadvantages

7.5.1 Advantages

- It is very easy to understand and implement.
- If we have large number of variables then, K-means would be faster than Hierarchical clustering.
- On re-computation of centroids, an instance can change the cluster.
- Tighter clusters are formed with K-means as compared to Hierarchical clustering.

7.5.2 Disadvantages

- It is a bit difficult to predict the number of clusters i.e. the value of k.
- Output is strongly impacted by initial inputs like number of clusters (value of k)
- Order of data will have strong impact on the final output.
- It is very sensitive to rescaling. If we will rescale our data by means of normalization or standardization, then the output will completely change.

- It is not good in doing clustering job if the clusters have a complicated geometric shape.