MACHINE LEARNING

ASSIGNMENT-5

1.The residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term.

In general terms, the sum of squares is a statistical technique used in regression analysis to determine the dispersion of data points. In a regression analysis, the goal is to determine how well a data series can be fitted to a function that might help to explain how the data series was generated. The sum of squares is used as a mathematical way to find the function that best fits (varies least) from the data.

The RSS measures the amount of error remaining between the regression function and the data set after the model has been run**.**

Calculate the Residual Sum of Squares

$RSS = \sum^{n}_{i=1} (y^i - f(x_i))^2$

Where:

y_i = the i^{th} value of the variable to be predicted

$f(x_i)$ = predicted value of y_i

n = upper limit of summation

2. Total sum of squares

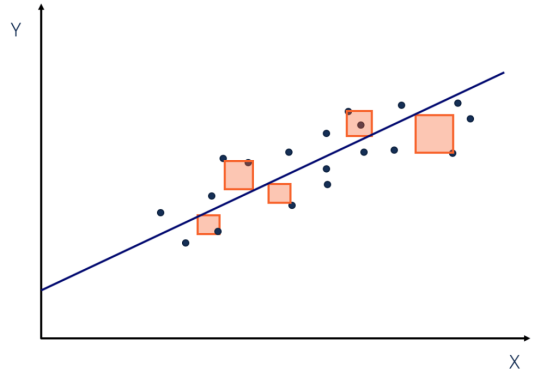
The total sum of squares is a variation of the values of a dependent variable from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a sample. It can be determined using the following formula:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where:

- y_i – the value in a sample
- $\bar{y}$ – the mean value of a sample.

Sum of squares (SS) is a statistical tool that is used to identify the dispersion of data as well as how well the data can fit the model in regression analysis. The sum of squares got its name because it is calculated by finding the sum of the squared differences.



The sum of squares is one of the most important outputs in regression analysis. The general rule is that a smaller sum of squares indicates a better model, as there is less variation in the data.

Types of Sum of Squares

In regression analysis, the three main types of sum of squares are the total sum of squares, regression sum of squares, and residual sum of squares.

1. Total sum of squares

The total sum of squares is a variation of the values of a dependent variable from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a sample. It can be determined using the following formula:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where:

- y_i – the value in a sample
- $\bar{y}$ – the mean value of a sample

Regression sum of squares (also known as the sum of squares due to regression or explained sum of squares)

The regression sum of squares describes how well a regression model represents the modeled data. A higher regression sum of squares indicates that the model does not fit the data well.

The formula for calculating the regression sum of squares is:

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

Where:

- $\hat{y}_i$ – the value estimated by the regression line
- $\bar{y}$ – the mean value of a sample

3. Residual sum of squares (also known as the sum of squared errors of prediction)

The residual sum of squares essentially measures the variation of modeling errors. In other words, it depicts how the variation in the dependent variable in a regression model cannot be explained by the model. Generally, a lower residual sum of squares indicates that the regression model can better explain the data, while a higher residual sum of squares indicates that the model poorly explains the data.

The residual sum of squares can be found using the formula below:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

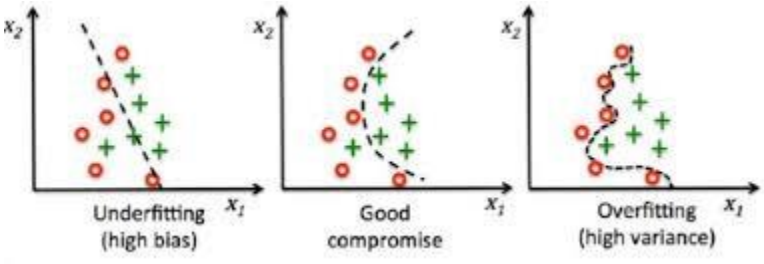
Where:

- y_i – the observed value
- $\hat{y}_i$ – the value estimated by the regression line

The relationship between the three types of sum of squares can be summarized by the following equation:

$$TSS = SSR + SSE$$

This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.



3.Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it. Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features."

Pros of Regularization

We can use a regularized model to reduce the dimensionality of the training dataset. Dimensionality reduction is important because of three main reasons:

- Prevents Overfitting: A high-dimensional dataset having too many features can sometimes lead to overfitting (model captures both real and random effects).
- Simplicity: An over-complex model having too many features can be hard to interpret especially when features are correlated with each other.
- Computational Efficiency: A model trained on a lower dimensional dataset is computationally efficient (execution of algorithm requires less computational time).

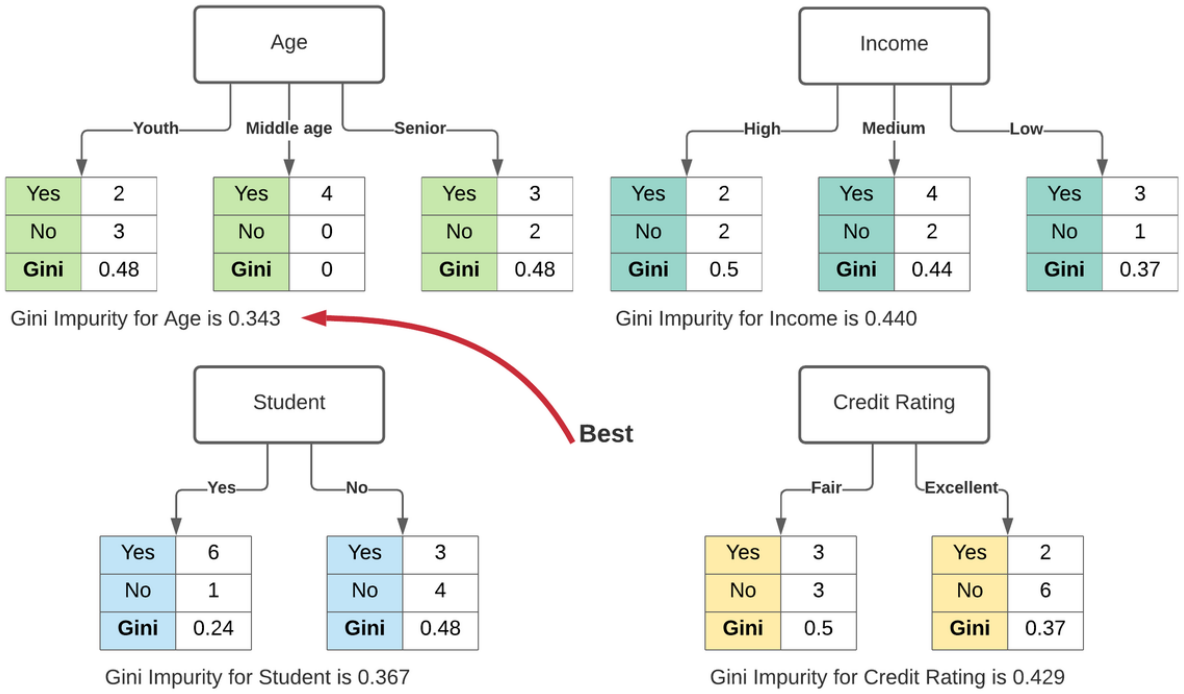
Cons of Regularization

- Regularization leads to dimensionality reduction, which means the machine learning model is built using a lower dimensional dataset. This generally leads to a high bias errror.
- If regularization is performed before training the model, a perfect balance between bias-variance tradeoff must be used.

4.Gini Index, also known as Gini impurity, **calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly**. If all the elements are linked with a single class then it can be called pure.

Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution

in the dataset.



For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labeled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – you could calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the **impurity** of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node. This process would continue for each subsequent node using the remaining features.

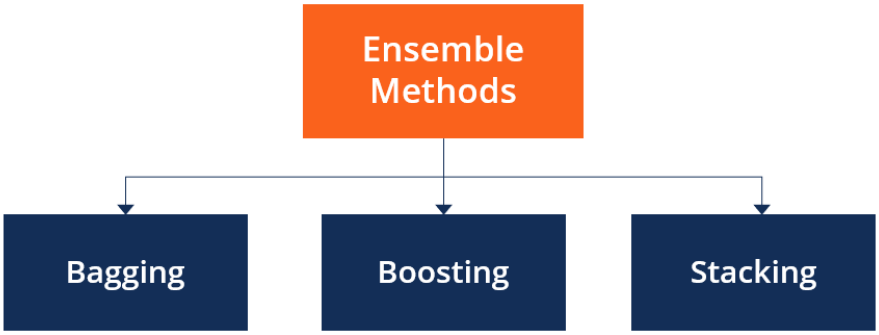
In the image above, Age has minimum gini impurity, so age is selected as the root in the decision tree.

5. Over-fitting is the phenomenon in which the learning system tightly fits the given training data so much that it would be inaccurate in predicting the outcomes of the untrained data.

In irregular decision trees, over-fitting occurs when the tree is designed so as to perfectly fit all samples in the training data set. Thus it ends up with branches with strict rules of sparse data. Thus this effects the accuracy when predicting samples that are not part of the training set.

One of the methods used to address over-fitting in decision tree is called **pruning** which is done after the initial training is complete. In pruning, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.

6. The ensemble is a method used in the machine learning algorithm. In this method, multiple models or 'weak learners' are trained to rectify the same problem and integrated to gain desired results. Weak models combined rightly give accurate models. Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine_learning.



- Ensemble methods aim at improving predictability in models by combining several models to make one very reliable model.
- The most popular ensemble methods are boosting, bagging, and stacking.
- Ensemble methods are ideal for regression and classification, where they reduce bias and variance to boost the accuracy of models.

Categories of Ensemble Methods

Ensemble methods fall into two broad categories, i.e., sequential ensemble techniques and parallel ensemble techniques. **Sequential ensemble techniques** generate base learners in a sequence, e.g., Adaptive Boosting (AdaBoost). The sequential generation of base learners promotes the dependence between the base learners. The performance of the model is then improved by assigning higher weights to previously misrepresented learners.

In **parallel ensemble techniques**, base learners are generated in a parallel format, e.g., random forest. Parallel methods utilize the parallel generation of base learners to encourage

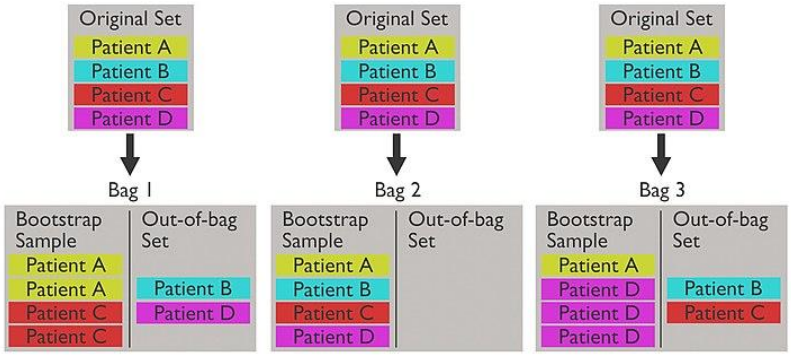
independence between the base learners. The independence of base learners significantly reduces the error due to the application of averages.

The majority of ensemble techniques apply a single algorithm in base learning, which results in homogeneity in all base learners. Homogenous base learners refer to base learners of the same type, with similar qualities. Other methods apply heterogeneous base learners, giving rise to heterogeneous ensembles. Heterogeneous base learners are learners of distinct types.

7.Bagging is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average. Bagging is an acronym for 'Bootstrap Aggregation' and is used to decrease the variance in the prediction model. Bagging is a parallel method that fits different, considered learners independently from each other, making it possible to train them simultaneously.
Bagging generates additional data for training from the dataset. This is achieved by random sampling with replacement from the original dataset. Sampling with replacement may repeat some observations in each new training data set. Every element in Bagging is equally probable for appearing in a new dataset.
These multi datasets are used to train multiple models in parallel. The average of all the predictions from different ensemble models is calculated. The majority vote gained from the voting mechanism is considered when classification is made. Bagging decreases the variance and tunes the prediction to an expected outcome.

Boosting is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm. Boosting is a sequential ensemble method that iteratively adjusts the weight of observation as per the last classification. If an observation is incorrectly classified, it increases the weight of that observation. The term 'Boosting' in a layman language, refers to algorithms that convert a weak learner to a stronger one. It decreases the bias error and builds strong predictive models.
Data points mispredicted in each iteration are spotted, and their weights are increased. The Boosting algorithm allocates weights to each resulting model during training. A learner with good training data prediction results will be assigned a higher weight. When evaluating a new learner, Boosting keeps track of learner's errors.

8.The out-of-bag (OOB) error is **the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample**. This allows the RandomForestClassifier to be fit and validated whilst being trained.
Out-of-bag (**OOB**) **error**, also called **out-of-bag estimate**, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from. OOB error is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

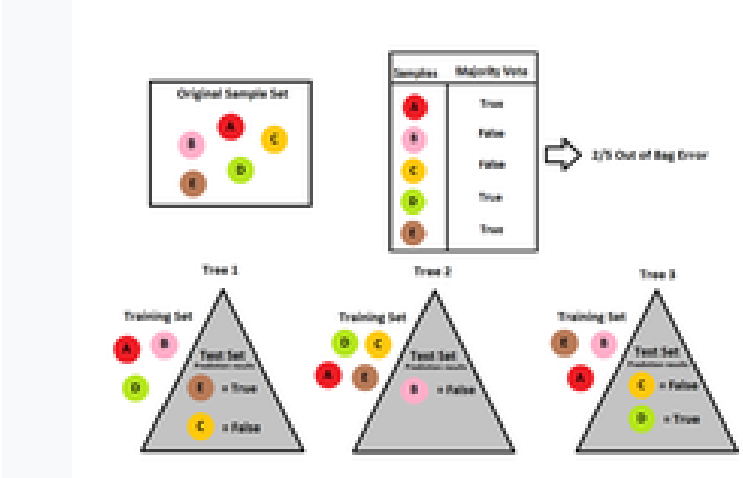


This example shows how bagging could be used in the context of diagnosing disease. A set of patients are the original dataset, but each model is trained only by the patients in its bag. The patients in each out-of-bag set can be used to test their respective models. The test would consider whether the model can accurately determine if the patient has the disease.

Calculating out-of-bag error

Since each out-of-bag set is not used to train the model, it is a good test for the performance of the model. The specific calculation of OOB error depends on the implementation of the model, but a general calculation is as follows.

1. Find all models (or trees, in the case of a random forest) that are not trained by the OOB instance.
2. Take the majority vote of these models' result for the OOB instance, compared to the true value of the OOB instance.
3. Compile the OOB error for all instances in the OOB dataset.



The bagging process can be customized to fit the needs of a model. To ensure an accurate model, the bootstrap training sample size should be close to that of the original set.[2] Also, the number of iterations (trees) of the model (forest) should be considered to find the true OOB error. The OOB error will stabilize over much iteration so starting with a high number of iterations is a good idea.[3]

Shown in the example to the right, the OOB error can be found using the method above once the forest is set up.

9. Cross validation is an evaluation method used in machine learning to find out how well your machine learning model can predict the outcome of unseen data. It is a method that is easy to comprehend; works well for a limited data sample and also offers an evaluation that is less biased, making it a popular choice. The data sample is split into 'k' number of smaller samples, hence the name: K-fold Cross Validation. You may also hear terms like four fold cross validation, or ten fold cross validation, which essentially means that the sample data is being split into four or ten smaller samples respectively.

k-fold cross validation performed as:

The general stratergy is quite straight forward and the following steps can be used:

1. First, shuffle the dataset and split into k number of subsamples. (It is important to try to make the subsamples equal in size and ensure k is less than or equal to the number of elements in the dataset).

2. In the first iteration, the first subset is used as the test data while all the other subsets are considered as the training data.

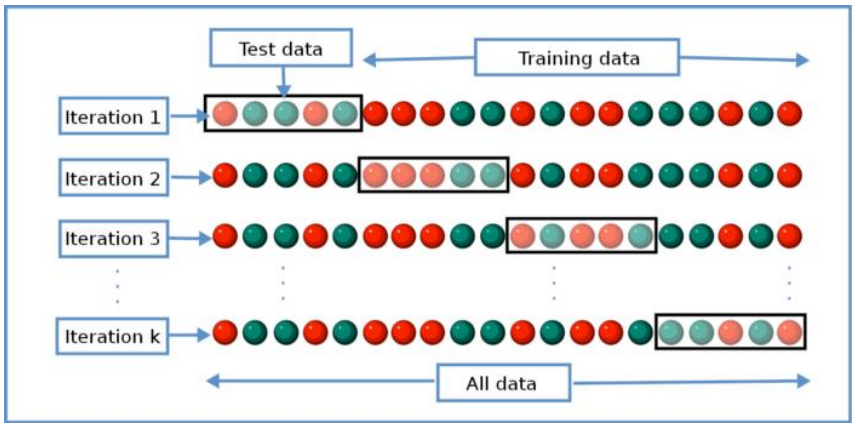
3. Train the model with the training data and evaluate it using the test subset. Keep the evaluation score or error rate, and get rid of the model.

4. Now, in the next iteration, select a different subset as the test data set, and make everything else (including the test set we used in the previous iteration) part of the training data.

5. Re-train the model with the training data and test it using the new test data set, keep the evaluation score and discard the model.

6. Continue iterating the above k times. Each data subsamples will be used in each iteration until all data is considered. You will end up with a k number of evaluation scores.

7. The total error rate is the average of all these individual evaluation scores.



Determine the best value for 'k' in K-Fold Cross Validation

Chosing a good value for k is important. A poor value for k can result in a poor evaluation of the model's abilities. In other words, it can cause the measured ability of the model to be overestimated (high bias) or change widely depending on the training data used (high variance).

Generally, there are three ways to select k:

- Let k = 5, or k =10. Through experimentation, it has been found that selecting k to be 5 or 10 results in sufficiently good results.
- Let k = n, where n is the size of the dataset. This ensures each sample is used in the test data set.

- Another way is to chose k so that every split data sample is sufficiently large, ensuring they are statistically represented in the larger dataset.

Types of cross validation

Cross validation can be divided into two major categories:

- Exhaustive, where the method learn and test on every single possibility of dividing the dataset into training and testing subsets.
- Non-exhaustive cross validation methods where **all** ways of splitting the sample are **not** computed.

Exhaustive cross-validation

Leave-p-out cross validation is a method of exhaustive cross validation. Here, p number of observations (or elements in the sample dataset) are left out as the training dataset, everything else is considered as part of the training data. For more clarity, if you look at the above image, p is equal to 5, as shown by the 5 circles in the 'test data'.

Leave-one-out cross validation a special form of leave-p-out exhuastive cross validation method, where p = 1. This is also a specific case for k-fold cross validation, where k = N(number of elements in the sample dataset).

Non-exhaustive cross-validation

K-fold cross validation where k is not equal to N, Stratified cross validation and repeated random sub-sampling validation are non-exhaustive cross validation methods.

10. Hyperparameter tuning consists of **finding a set of optimal hyper parameter values for a learning algorithm while applying this optimized algorithm to any data set**. That combination of hyper parameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.
However, there is another kind of parameter, known as ***Hyper parameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.
Some examples of model hyper parameters include:

1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
2. The learning rate for training a neural network.
3. The C and sigma hyperparameters for support vector machines.
4. The k in k-nearest neighbors.

11. In order for Gradient Descent to work, we must set the learning rate to an appropriate value. This parameter determines how fast or slow we will move towards the optimal weights. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values. So using a good learning rate is crucial.

Gradient descent is the popular optimization algorithm used in machine learning to estimate the model parameters. During training a model, the value of each parameter is guessed or assigned random values initially. The cost function is calculated based on the initial values and the parameter estimates are improved over several steps such that the cost function assumes a minimum value eventually. Learning rate explained through a child's interaction

To understand this better let's consider an example.

If a child sees ten dogs and all of them are black in color, he might believe that all dogs are black and would consider this as a feature when trying to identify a dog.

Imagine he's shown a white dog, and his parents tell him that it's a dog. With a desirable learning rate, he would quickly understand that black color is not an important feature of dogs and would look for another feature.

But with a low learning rate, he would consider the white dog as an outlier and would continue to believe that all dogs are black.

And if the learning rate is too high, he would instantly start to believe that all dogs are white even though he has seen more black dogs than white ones

The point is it's' really important to achieve a desirable learning rate because:

- both low and high learning rates results in wasted time and resources
- A lower learning rate means more training time
- more time results in increased cloud GPU costs
- a higher rate could result in a model that might not be able to predict anything accurately.

  A desirable learning rate is one that's low enough so that the network converges to something useful but high enough so that it can be trained in a reasonable amount of time.

12.Logistic Regression has traditionally been used as a linear classifier, i.e. when the classes can be separated in the feature space by linear boundaries.

13.AdaBoost

AdaBoost or Adaptive Boosting is the first Boosting ensemble model. The method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.

In practice, this boosting technique is used with simple classification trees or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.

Gradient Boost

Gradient Boost is a robust machine learning algorithm made up of Gradient descent and Boosting. The word 'gradient' implies that you can have two or more derivatives of the same

function. Gradient Boosting has three main components: additive model, loss function and a weak learner.

The technique yields a direct interpretation of boosting methods from the perspective of numerical optimisation in a function space and generalises them b

The Comparison

Loss Function:
The technique of Boosting uses various loss functions. In case of Adaptive Boosting or AdaBoost, it minimises the exponential loss function that can make the algorithm sensitive to the outliers. With Gradient Boosting, any differentiable loss function can be utilised. Gradient Boosting algorithm is more robust to outliers than AdaBoost.

Flexibility

AdaBoost is the first designed boosting algorithm with a particular loss function. On the other hand, Gradient Boosting is a generic algorithm that assists in searching the approximate solutions to the additive modelling problem. This makes Gradient . Boosting more flexible than AdaBoost

Benefits
AdaBoost minimises loss function related to any classification error and is best used with weak learners. The method was mainly designed for binary classification problems and can be utilised to boost the performance of decision trees. Gradient Boosting is used to solve the differentiable loss function problem. The technique can be used for both classification and regression problems.

14. In statistics and machine learning, the **bias–variance tradeoff** is the property of a model that the variance of the parameter estimated across samples can be reduced by increasing the bias in the estimated parameters. The **bias–variance dilemma** or **bias–variance problem** is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set.

The bias–variance tradeoff is a central problem in supervised learning. Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also generalizes well to unseen data. Unfortunately, it is typically impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well but are at risk of overfitting to noisy or unrepresentative training data. In contrast, algorithms with high bias typically produce simpler models that may fail to capture important regularities (i.e. underfit) in the data

15. Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

1. Polynomial Kernel Function

The polynomial kernel is a general representation of kernels with a degree of more than one. It’s useful for image processing.

There are two types of this:

- **Homogenous Polynomial Kernel Function**

$\mathbf{K(x_i,x_j) = (x_i.x_j)^d}$, where ‘.’ is the dot product of both the numbers and d is the degree of the polynomial.

- **Inhomogeneous Polynomial Kernel Function**

$\mathbf{K(x_i,x_j) = (x_i.x_j + c)^d}$ where c is a constant.

2. Gaussian RBF Kernel Function

RBF is the radial basis function. This is used when there is no prior knowledge about the data.

It’s represented as:

$$\mathbf{K(x_i,x_j) = exp(-\gamma||x_i - x_j||)^2}$$

3.Linear Kernel Function

This kernel is one-dimensional and is the most basic form of kernel in SVM. The equation is:

$$\mathbf{K(x_i,x_j) = x_i.x_j + c}$$