

Boosting Techniques

What is Boosting?

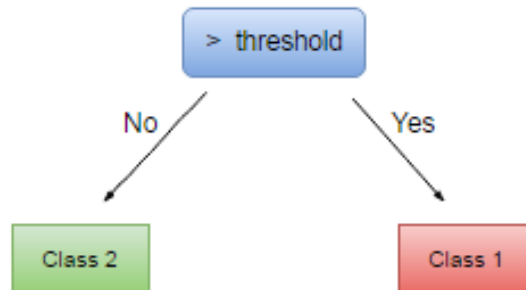
Ensemble Learning is a method that is used to enhance the performance of Machine Learning model by combining several learners. Boosting is a technique which combines many models (weak learners) to create a strong model (strong learner). Basic idea is to correct the predecessor by training predictors sequentially. The boosting algorithms are primarily used in machine learning for reducing bias and variance.

What is Ada boosting Algorithm?

Ada boost or adaptive boosting is the first stepping stone in the world of boosting algorithms. Ada boost is the first boosting algorithm among all the boosting algorithms. The main idea behind this algorithm is to combine multiple weak learners into a strong classifier. It can be used for both classification and regression tasks.

Ada Boost classifier, a first base classifier (such as a Decision Tree) is trained and used to make predictions on the training set. The relative weight of misclassified training instances is then increased. A second classifier is trained using the updated weights and again it makes predictions on the training set, weights are updated, and so on.

Base model is called as Stump. (decision trees have a depth of 1 (i.e. 2 leaves))



Steps in Ada Boosting:

1. Initialize with Sample Weights for each observations in a dataset.
2. Build a decision tree with each feature
3. Calculate the significance of the trees developed
4. Update the sample weights of misclassified observations and give more weightage to misclassified samples and reduce weights of correctly classified samples.
5. Normalize the weights.
6. The dataset with updated weights will act as our input data. Repeat steps 2 to 5 until all observations are correctly classified.

Gradient Boosting

The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. The objective here is to minimize this loss function by adding weak learners using gradient descent. Since it is based on loss function hence for regression problems, we'll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g log-likelihood.

Steps in Gradient Boosting:

1. The first step in gradient boosting is to build a base model to predict the observations in the training dataset. For simplicity we take an average of the target column and assume that to be the predicted value.
2. The next step is to calculate the pseudo residuals which are (observed value – predicted value)
3. we will build a model on these pseudo residuals and make predictions. We want to minimize these residuals and minimizing the residuals will eventually improve our model accuracy and prediction power.
4. In this step we find the output values for each leaf of our decision tree. That means there might be a case where 1 leaf gets more than 1 residual, hence we need to find the final output of all the leaves. Average of all the numbers in a leaf, doesn't matter if there is only 1 number or more than 1.
5. This is finally the last step where we must update the predictions of the previous model. It can be updated as:

New weight = Average weight + (learning rate x new prediction1) + (learning rate x new prediction2)

XGBoost Algorithm

The XGBoost is having a tree learning algorithm as well as linear model learning, and because of that, it is able to do parallel computation on a single machine. This makes it 10 times faster than any of the existing gradient boosting algorithms.

1. System Optimization

- **Tree Pruning** – The XGBoost algorithm uses the depth-first approach, unlike the stopping criterion for tree splitting used by GBMS, which is greedy in nature and it also depends upon the negative loss criterion. The XGBoost instead uses the max depth feature/parameter, and hence it prunes the tree in a backward direction.
- **Parallelization** – The process of sequential tree building is done using the parallelized implementation in the XGBoost algorithm. This is made possible due to the outer and inner loops that are interchangeable. The outer loop lists the leaf nodes of a tree, while the inner loop will calculate the features. Also, in order for the outer loop to start, the inner loop must get completed. This process of switching improves the performance of the algorithm.
- **Hardware Optimization** – Hardware optimization was also considered during the design of the XGBoost algorithm. Internal buffers are allocated for each of the threads to store the gradient statistics.

2. Algorithmic Enhancements

- **Awareness of Sparsity** – XGBoost is known to handle all different types of sparsity patterns very efficiently. This algorithm learns the next missing value by seeing the training loss.
- **Regularization** – In order to prevent overfitting, it corrects more complex models by implementing both the LASSO (also called L1) and Ridge regularization (also called L2).
- **Cross-Validation** – It is having built-in cross-validation features that are being implemented at each iteration in the model creation. This prevents the need to calculate the number of boosting iterations needed.
- **Distributed Weighted Quantile Sketch** – It uses the distributed weighted quantile sketch to get the optimal number of split points among the weighted datasets.

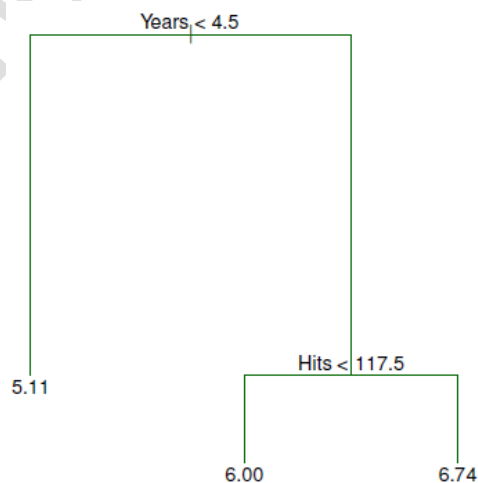
Decision trees

Introduction

Decision trees are one of the very important algorithms in the field of machine learning. It is influenced by the real-world tree structure. As the name suggests, decision trees are used in decision making based on the condition split (refer the example).



Example: Consider a sports player salary, first condition to split is years of experience. Other features are of least importance, if the years of experience is less than 4.5 years. If years of experience is greater than 4.5 years, then number of Hits is the factor of split.



MACHINE LEARNING

The conditions where there are splits are called **internal nodes** (Years, Hits). Based on the condition, the tree is split into **branches**. The end of the branch where further split is not possible is called as **decision** or **leaf** (5.11, 6.00, 6.74 are the decisions).

Recursive binary splitting

This is the approach which is used for splitting the branches in tree. In recursive binary splitting, at each condition the tree is split into two new branches.

Top-down approach

It's a top-down approach where split begins at the top of the tree, the split is successively carried out further until we reach the final decision or leaf. Each split will result in two new branches.

Greedy approach

At each step of the tree-building process, the best split is made at that step, rather than looking ahead and picking a split that will lead to a better tree in some future step. The split is based on cost, the one with the least cost function is chosen as the split. This method continuous for the subsequent splits until it reaches the final decision or leaf. This is called as 'Greedy' as it has greed to reduce the cost function at every point.

Linear regression vs Decision tree regression

Linear regression fits well when the response variable has linear relationship with the predictors, while it does not perform well if there is non-linear relationship.

If there is a highly non-linear and complex relationship between the features, then decision trees may outperform classical approaches.

Classification tree

For classification problems, where the outcome is categorical such as yes/no, male/female, spam/ham, etc. the decision tree classifier is used to make decision.

Regression tree

For regression problems where the outcome is a continuous value like price of house, salary of a person, etc. the decision tree regressor is used to make decisions.

Working

Decision trees use multiple strategies to split a node into sub nodes. Decision tree splits considering all the available features and tries to split which results in similar set of values on either side. For example, if we split fruits based on colour then orange colour will be on one side and red coloured fruits will be on other side.

Algorithm selection will be based on the type of target variables.

Advantages

1. Its very simple to understand and visualize
2. Implicit feature selection takes place
3. Handles both categorical and numerical data
4. Data preparation takes lesser effort

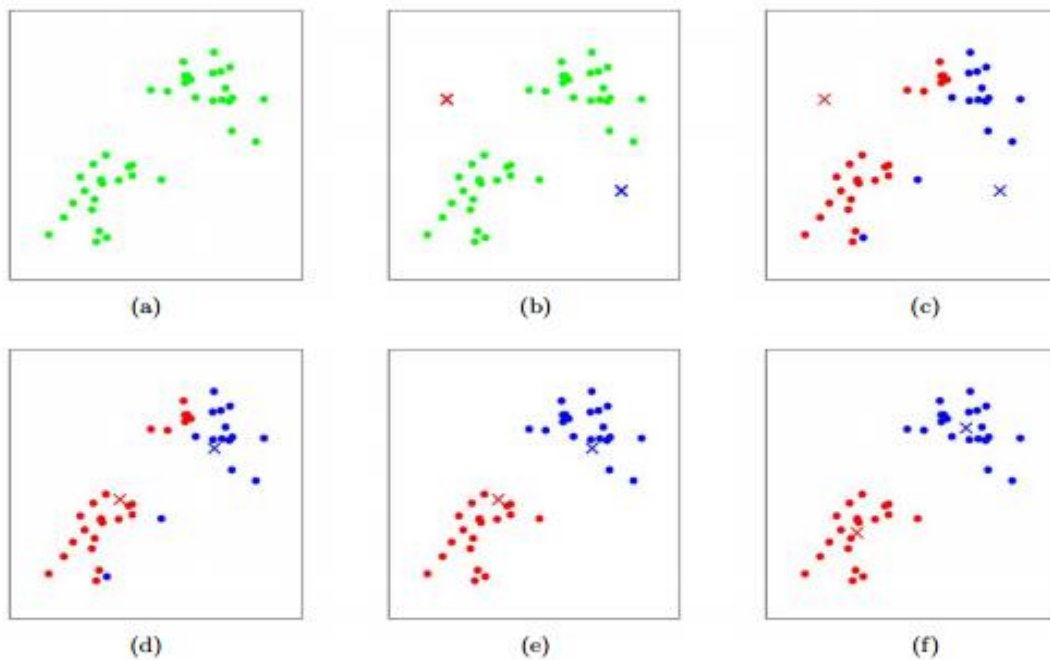
Disadvantages

1. The algorithm will not assure global optimal decision tree.
2. It creates biased trees if the data is dominated by one class. It works well if the dataset is balanced.
3. Small changes in the data will result in complete change in the decision tree generated.
4. It generates over complex trees which might not work well on unseen data.

K Means Clustering

The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. There are many different types of clustering methods, but k-means is one of the oldest and most approachable.

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group.



Steps in K means algorithm:

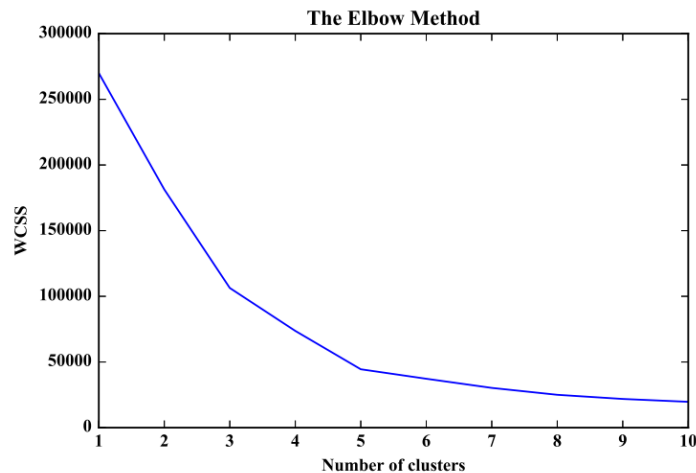
1. Select the "k" value.
2. Randomly initialize "k" Centroids.
3. Take the average of all the points which are nearest to the centroid points
4. Move the centroids to the mean value of group
5. Repeat the step 3 and step 4, as long as there is no movement of centroids.

Choosing K Value:

For each value of K, within-cluster sum of squares (WCSS) is calculated and plotted with K on X axis and WCSS on Y axis.

WCSS is calculated with below formula

$$WCSS = \sum_{i \in n} (X_i - Y_i)^2$$



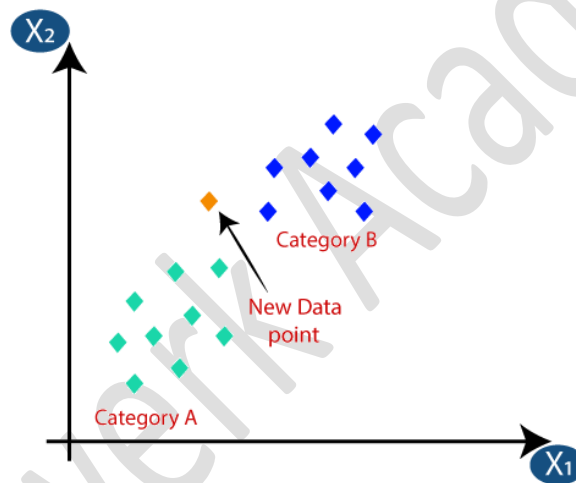
The point where we see a bend and an elbow like curve is generally considered as best indicator of K. In the above case we have it at 3 and 5, we go with 5 as K value as the K value should not be too small as well.

K-Nearest Neighbours

Introduction



- K-nearest neighbours popularly known as KNN algorithm is most widely used for classification problems, even though it can be used for both regression and classification problems.
- KNN is a Supervised Machine Learning Algorithm that classifies the data based on the input labels learned by the model (ex: Distance).

Example: We will teach a child to recognize an apple by telling few characteristics of it such as shape, colour, etc. Next time when we show an apple picture, the child can easily recognize its an apple based on characteristics.



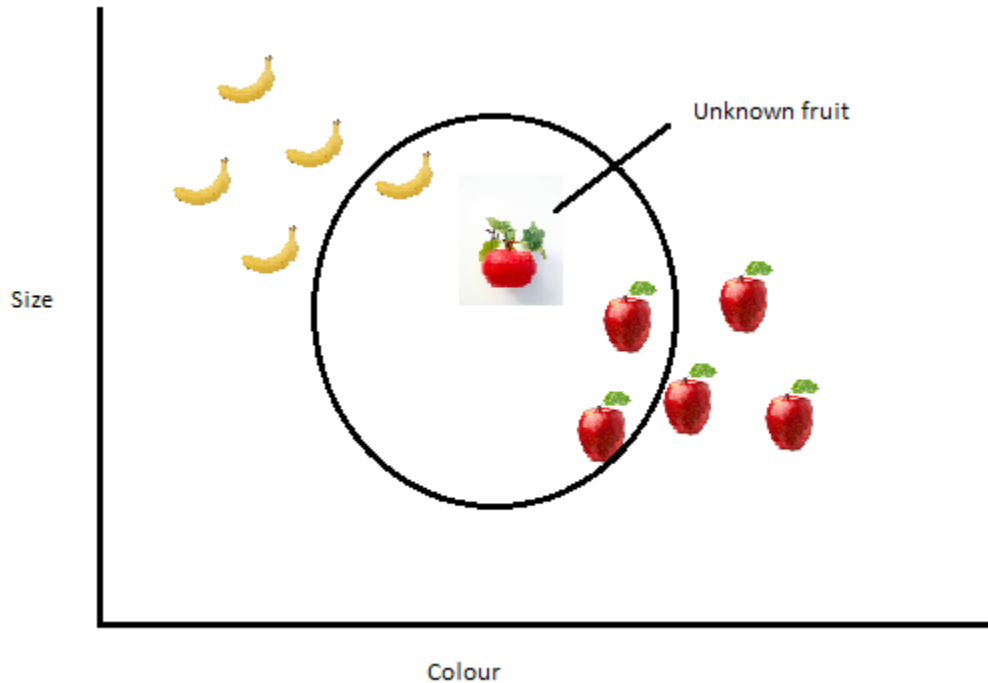
Why KNN?

KNN is based on feature similarity, the classification can be done using KNN. Let us look at an examples.

	
Yellow in colour	Red in colour
Long shaped	Round shaped
No stem attached	Have stem attached
Comparatively soft	Comparatively hard
Cannot eat with skin	Can eat with skin

MACHINE LEARNING

Hence with the feature comparison, if we had to classify an unknown fruit as banana or apple, we can use KNN to do it. Following is an example with $K = 3$.



Working of KNN:

1. First step is to select a K -value (say $K = 3$)
2. Whenever a new data point comes, the distance of the new data point is calculated with all the other existing data points.
3. Based on the distances all the records are sorted in ascending order
4. Based on selected K -value, we select the first K -records(distances). For example, if we select $K = 3$ then first 3 distances(records) are selected from the sorted distances
5. a) For KNN classifier, we will check the classes(output) of the records selected from above step 4 and based on majority voting we will assign the new data point to the majority class
b) For KNN Regressor, we can take the mean/median/mode of the output values that are selected from the above step 4.

How distances are calculated?

- 1) **Minkowski**: It is the generalized distance metric and is given by

$$\left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

- 2) **Manhattan Distance**: We use Manhattan distance if we need to calculate the distance between two data points in a grid like path. If we substitute $p = 1$ in Minkowski formula, we get the Manhattan distance.

$$d = \sum_{i=1}^n |x_i - y_i|$$

- 3) **Euclidean distance**: Euclidean distance formula can be used to calculate the distance between two data points in a plane. This is the most widely used distance metric.

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- 4) **Hamming distance**: This distance metric is used when dealing with categorical data. If both the categorical values are same we assign 0 or else 1

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

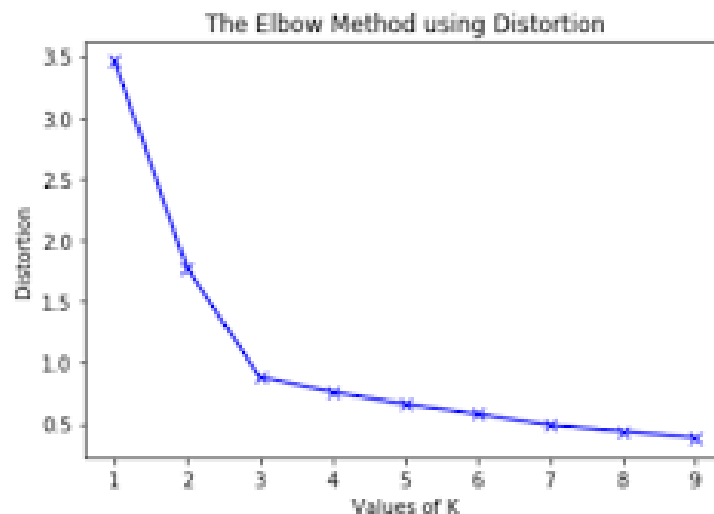
Choosing K value

KNN algorithm classifies the unknown class based on its similarity with the nearest neighbours, choosing k value has significant effect on the accuracy of the model. There are two ways to choose k value

1. \sqrt{n} where n is the total number of data points.
2. Odd value is chosen to avoid the confusion between the classes of data points

Choose K value as odd number and try to increase the K value to get smoother and better results, once the accuracy starts decreasing which means error will increase, choose the K with the best training accuracy.

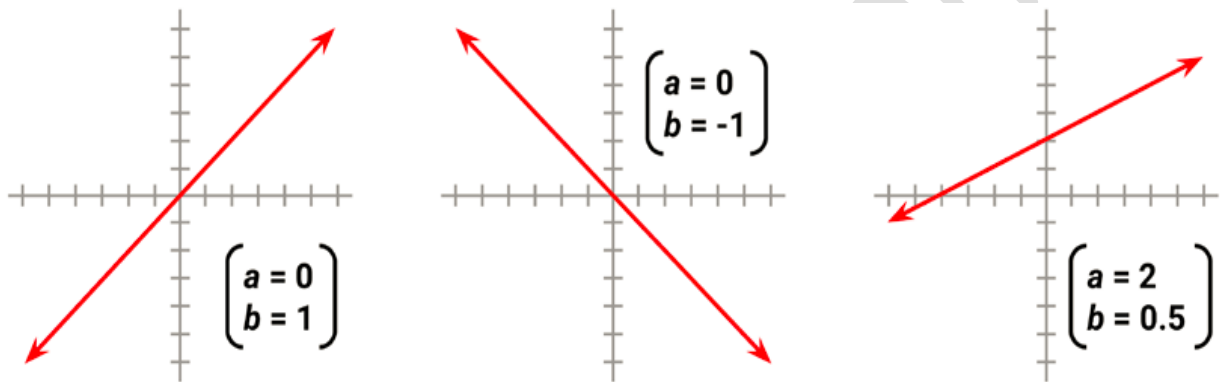
Usually elbow method is followed to get the best value, the following diagram we can see a sharp bend near the value K which is the optimal point after which the decrease in the value is not much. This value is chosen as K value.



Linear Regression

Introduction

Linear regression refers to estimating the given function using a linear combination of input variables. Regression is concerned with specifying the relationship between a **dependent variable** (the value to be predicted) and one or more numeric **independent variables** (the predictors). The dependent variable depends upon the value of the independent variable. The simplest regression assumes that the relationship between the dependent and independent variables follows a straight line.



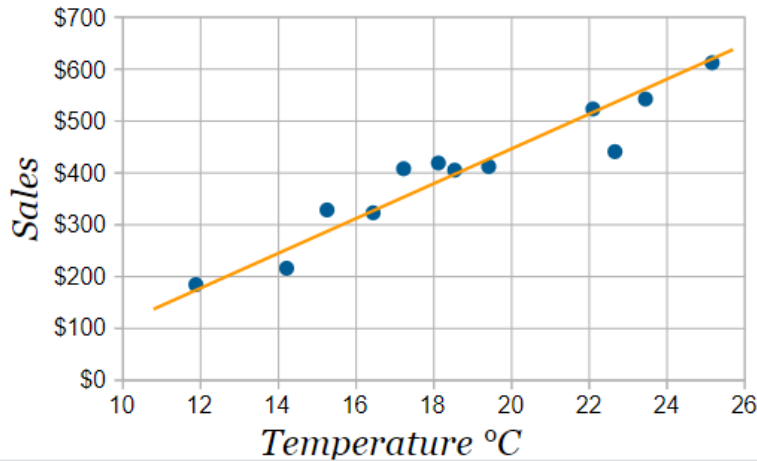
Examples:

- Performance of a student in exam depends on no. of hours studied, no. of classes attended, no. of problems solved etc. Here we can assume that these variables are linearly related to each other.
- Sales of a car and the advertising are linearly related. Higher the advertising done, higher is the sales of a car.
- Price of a house depends on location, age of house, no. of bedrooms etc.

MACHINE LEARNING

Least Squares method (Line of best fit)

Considering the data points, we want to fit the line which best covers the data points.



We can achieve this by just looking at the data points and drawing line which is as close to the data points as possible, but we cannot assure if it's the best line possible. To achieve better accuracy, we can calculate Least squares regression which is measure of the best fit.

Steps

To find the best fit line for N number of points

- For each data point given by the coordinates (x,y), we will calculate x^2 and xy .
- Once the calculation is done for all the data points, will add x , y , xy , x^2 individually which will result in summation $\sum x$, $\sum y$, $\sum x^2$, $\sum xy$.
- Next step is to calculate slope

$$m = \frac{N \sum(xy) - \sum x \sum y}{N \sum(x^2) - (\sum x)^2}$$

N is the number of data points

- Now calculate the intercept

$$b = \frac{\sum y - m \sum x}{N}$$

- The final equation will be

$$y = mx + b$$

Note: The python package scikit-learn will do all the calculations internally, you need to know how it is done.

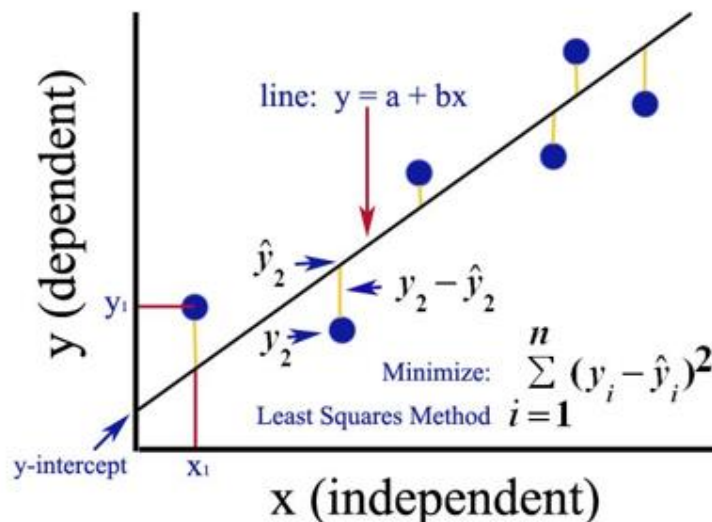
MACHINE LEARNING

Task: For the following case, find the best line equation and find what will be y value for given x = 8.

x (Hours of sunshine)	y (Number of cold drinks sold)
2	4
3	5
5	7
9	15
7	10

Cost Function

The line which is fit will not pass through all the data point and thus there will be difference between the actual line and the data point which is called as error. The purpose of fitting the best line is to get the least square of errors.

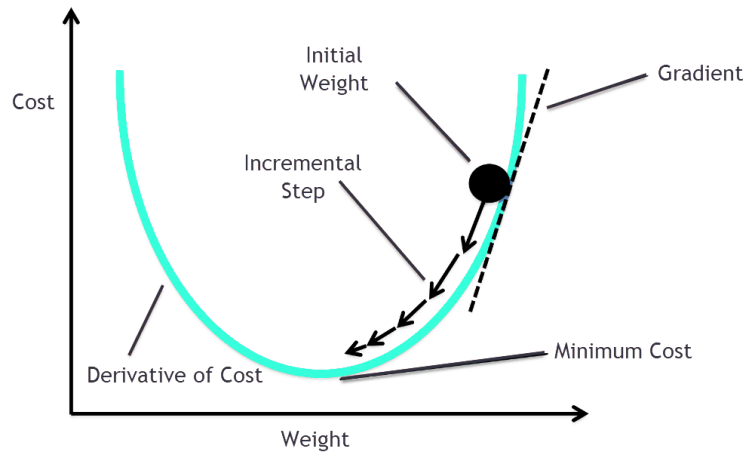


Note: Outliers will cause significant effect on the best line, it will change the intercept and the slope.

Cost function or loss function is the error in predicted m and c value in line equation. Let us consider MSE (Mean squared error) as loss function. It is the average of the squared errors; error is the difference between the predicted and the actual output. The MSE is calculated for given set of weights (m and c value). The weights are adjusted such that we get the least possible MSE. Mathematically MSE is represented by

$$MSE = \frac{1}{n} \sum \left(\underbrace{y - \hat{y}}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}} \right)^2$$

Gradient Descent



Gradient descent is a method where the weights of model are adjusted so that an optimal value of cost is reached. The incremental step makes sure the weights are changed and this helps to reach the minimum cost, the step size is very important if the step size is very high then there are chances that the optimal point is skipped on the other hand if the step size is very low then it takes lot of time to converge to the minima.

Logistic Regression

Introduction

Logistic regression is very similar to linear regression; it tries to find an equation which can predict binary outcome for set of two or more response variables. Unlike linear regression, the response variables can be categorical.

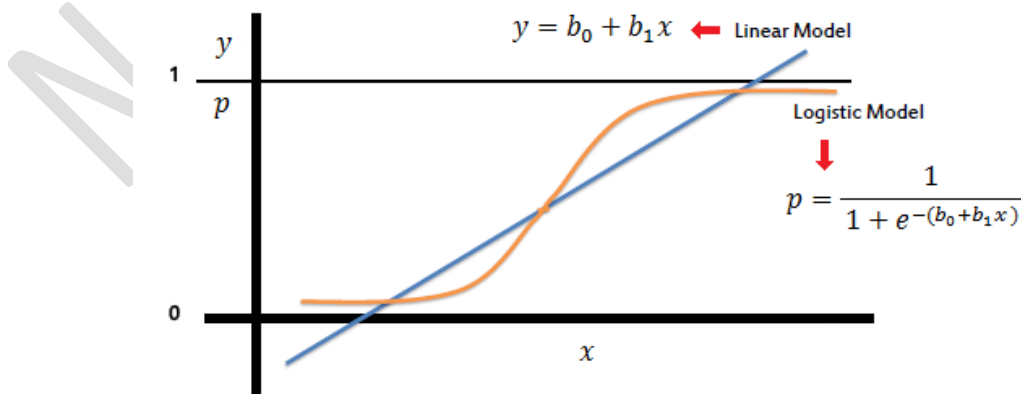
Example:

Logistic regression can be used to predict:

1. Credit card fraud detection
Based on features such as date, amount spent, place, type of transaction, etc. the banks will use logistic regression to predict whether it's a valid transaction or a fraud. For instance, if a person never spends high amount and the transaction is recorded which has high amount spent then it's classified as fraud.
2. Spam or Ham
Based on the text content present in the mail, the mails are classified. If the email contains free offers, payment related email from unregistered mails is classified as Spam.

Why not linear regression?

Linear regression fails when the output value to be predicted is categorical, which means the labels have a fixed outcome. The outcome of logistic regression is the probabilities and thus the values must be between 0 and 1. Linear regression will have the values above 1 and below 0, thus it does not fit well for the categorical outcomes. Using logistic regression, any values between $-\infty$ (-infinity) to $+\infty$ (+infinity) can be fit between 0 and 1. Also linear regression is sensitive to the data points the line shifts if there are any outliers or if there are any additional data points coming in.



Types of logistic regression

There are mainly three types of logistic regression:

1. **Simple logistic regression:** This is also called as binomial type where the number of outcomes is two; the target variable can have only two values. Ex: When we are predicting whether a person survived or not, the outcome can be either survived or not survived. The logistic equation is given by
2. **Multiclass or Multinomial regression:** The type of logistic regression where there are chances of more than two outcomes. Instead of predicting {0,1}, we will be predicting {0, 1, 2, ..., n} where n+1 represent the number of classes (since we are starting from 0 as first class).

Logistic regression equation

The logistic regression equation builds on the assumption that log-odds of observations can be represented in the form of linear equation.

$$\log \frac{P(\mathbf{x})}{1 - P(\mathbf{x})} = \sum_{j=0}^K b_j x_j$$

Where b_0 is constant, $x_0=1$

The LHS of the equation is known as logit(P), thus the name is logistic regression.

In the next step, take exponent on both sides

$$\begin{aligned} \frac{P(\mathbf{x})}{1 - P(\mathbf{x})} &= \exp\left(\sum_{j=0}^K b_j x_j\right) \\ &= \prod_{j=0}^K \exp(b_j x_j) \end{aligned}$$

Exponential of sum of the linear equation is the product of exponential of linear equation by product rule of exponents.

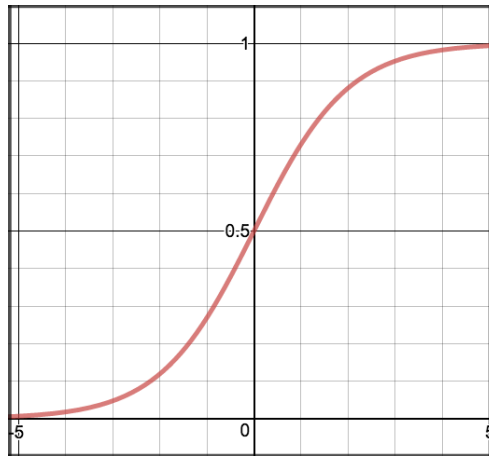
If we invert the odds ratio, then the equation turns out to be

$$P(\mathbf{x}) = \frac{\exp z}{1 + \exp z}$$

where

$$z = \sum_{j=0}^K b_j x_j$$

Graphical representation



The output of sigmoid function lies in the range of zero and one for any value of x.

Use case

Let us say we want to know whether a person would pass or fail based on the number of hours he studied. We calculate the probability using sigmoid function, if we take yes as the class 1, we calculate $P(\text{pass})$ or $P(\text{class} = 1)$. If $P(\text{class}=1)$ is greater than 0.5, we consider it as pass else we consider it as fail.

Loss function

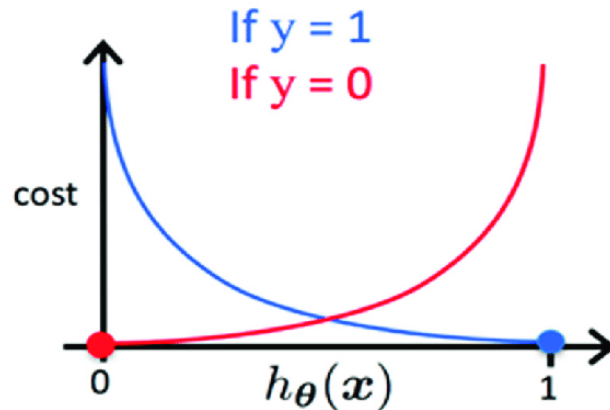
The loss function or the cost function used in case of logistic regression is Cross-entropy. It is also known as log loss. The cross-entropy loss is divided into two equations based on the output. For $y = 1$, we have separate equation and for $y = 0$ we have separate equation.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$
$$\begin{aligned} \text{Cost}(h_{\theta}(x), y) &= -\log(h_{\theta}(x)) && \text{if } y = 1 \\ \text{Cost}(h_{\theta}(x), y) &= -\log(1 - h_{\theta}(x)) && \text{if } y = 0 \end{aligned}$$

The combined equation which can be used irrespective of class is represented as

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

The purpose of using log function in the equation is to smoothen the gradient curve to ease the calculation of gradient and minimize the cost.



Multinomial Regression

Like the binomial regression as seen above, we can use the equation to calculate probability with slight modification and slightly modified steps.

The function which is used for multinomial regression is known as softmax function.

Softmax

Unlike sigmoid, this function takes input vector of K real numbers and converts them into probability distribution of K probabilities which is proportional to the exponential of the inputs. The function is given below

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } z = z_1, \dots, z_K$$

In case of multinomial regression, we calculate probability of each class and the class with the highest probability will be assigned to the output.

Advantages

1. It is one of the simplest machine learning algorithms and is easy to implement.
2. Provides great efficiency in some classification problems.
3. In a low dimensional dataset having a enough training examples, logistic regression is less prone to over-fitting
4. Logistic Regression proves to be very efficient when the dataset has features that are linearly separable.
5. Rather than straight away starting with a complex model, logistic regression is sometimes used as a benchmark model to measure performance, as it is relatively quick and easy to implement.

Disadvantages

1. Non-linear problems can't be solved with logistic regression since it has a linear decision surface. Linearly separable data is rarely found in real world scenarios. So the transformation of non-linear features is required which can be done by increasing the number of features such that the data becomes linearly separable in higher dimensions.
2. It is difficult to capture complex relationships using logistic regression. More powerful and complex algorithms such as Neural Networks can easily outperform this algorithm.
3. Only important and relevant features should be used to build a model otherwise the probabilistic predictions made by the model may be incorrect and the model's predictive value may degrade.
4. In Linear Regression independent and dependent variables should be related linearly. But Logistic Regression requires that independent variables are linearly related to the log odds ($\log(p/(1-p))$).

Multicollinearity

Multicollinearity is a statistical concept where several independent variables in a model are correlated. This means that an independent variable can be predicted from another independent variable.

Multicollinearity may not affect the accuracy of the model as much but we might lose reliability in determining the effects of individual independent features on the dependent feature in your model and that can be a problem when we want to interpret your model.

Variance Inflation Factor (VIF)

Although correlation matrix and scatter plots can also be used to find multicollinearity, their findings only show the bivariate relationship between the independent variables. VIF is preferred as it can show the correlation of a variable with a group of other variables.

VIF score of an independent variable represents how well the variable is explained by other independent variables.

R² value(R-squared) is determined to find out how well an independent variable is described by the other independent variables. A high value of **R²** means that the variable is highly correlated with the other variables. This is captured by the **VIF** which is denoted below:

$$VIF = 1 / (1 - R^2)$$

So, the closer the **R²** value to 1, the higher the value of VIF and the higher the multicollinearity with the particular independent variable.

- VIF starts at 1 (when $R^2=0$, $VIF=1$ – minimum value for VIF) and has no upper limit.
- $VIF = 1$, no correlation between the independent variable and the other variables.
- VIF exceeding 5 or 10 indicates high multicollinearity between this independent variable and the others

Performance Metrics

Cost Function

Cost function is a measure of the error term between the actual value and the predicted value of the model. There are different cost functions which can be used and is used as measure of how good the model is performing on the unknown data.

Regression

1. Mean Absolute Error

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

2. Mean Squared Error

$$MSE = \frac{1}{n} \sum \underbrace{(y - \hat{y})^2}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}}$$

3. Root Mean Squared Error

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

Classification

1. Binary Cross Entropy

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

2. Categorical Cross Entropy

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Performance metrics in classification

1. Confusion matrix

	Predicted Negative	Predicted Positive
Observed Negative (default=0)	True Negative (TN)	False positive (FP)
Observed Positive (default=1)	False negative (FN)	True positive (TP)

2. Sensitivity or Recall

Sensitivity (Recall or True positive rate) is calculated as the number of correct positive predictions divided by the total number of positives.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

3. Specificity

Specificity (true negative rate) is calculated as the number of correct negative predictions divided by the total number of negatives.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

4. Precision

Precision (Positive predictive value) is calculated as the number of correct positive predictions divided by the total number of positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

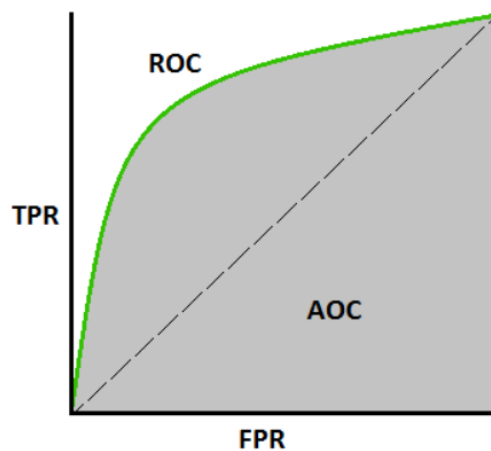
5. F1-Score

F1 score is used as a measure when both precision and recall are important as part of performance metrics.

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

AUC-ROC curve

- AUC - ROC curve is a performance measurement for classification problem at various thresholds settings.
- Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.



Principal Component Analysis

Introduction

Consider a dataset with huge number of predictors, a greater number of predictors might cause problems which can be listed as follows

1. Difficult to understand the relationships between the predictors.
2. Many variables might lead to overfitting of the model.

Is it possible to consider all the predictors and only concentrate on few predictors? It is possible to reduce the dimension of the feature space. The high dimensional feature space is reduced to lesser dimensional feature space without losing the important information and is done using PCA.

Reducing the dimension of the feature space is called as 'Dimensionality reduction'. This can be achieved in many ways, but mainly fall under two categories.

1. Feature extraction
2. Feature elimination

In feature elimination, we remove the unwanted predictors and keep only the important ones. The problem with the feature elimination we cannot extract any information if present from the predictors dropped.

On the other hand, in feature extraction we will not face the issue of losing information. Let's say we have 100 features and we are reducing it to 10 new features, these 10 new features will have the important information extracted from each of the 100 features.

PCA identifies list of principal axes to define the dataset to rank them based on the amount of variance present in them.

$$PC1 = w_{1,1}(\text{Feature A}) + w_{2,1}(\text{Feature B}) + w_{3,1}(\text{Feature C}) \dots + w_{n,1}(\text{Feature N})$$

$$PC2 = w_{1,2}(\text{Feature A}) + w_{2,2}(\text{Feature B}) + w_{3,2}(\text{Feature C}) \dots + w_{n,2}(\text{Feature N})$$

$$PC3 = w_{1,3}(\text{Feature A}) + w_{2,3}(\text{Feature B}) + w_{3,3}(\text{Feature C}) \dots + w_{n,3}(\text{Feature N})$$

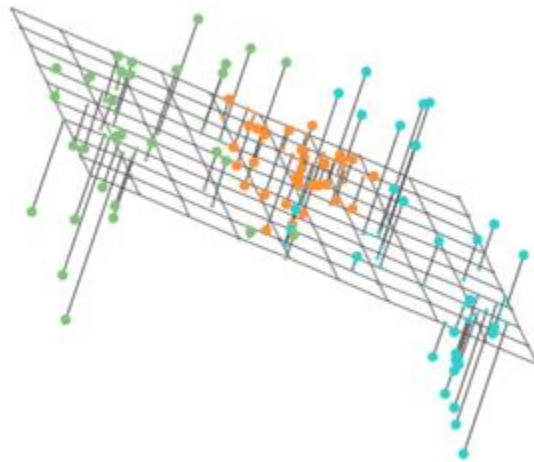
In this case, PC1 which is the first principal component formed by the linear combination of the features to get the magnitude and the direction, this component explains the highest variance of the features and thus contains the maximum information.

MACHINE LEARNING

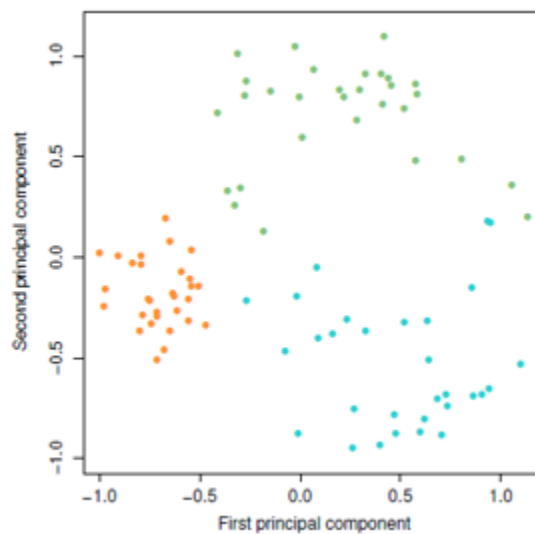
The second component PC2 is also a combination of features but with the lesser variance than PC1, PC2 has no correlation with PC1. Successive components capture variance of remaining variation without being correlated with the other components.

PCA – Representation

First let us represent three-dimensional feature space.



Now the same features can be converted to two principal components.



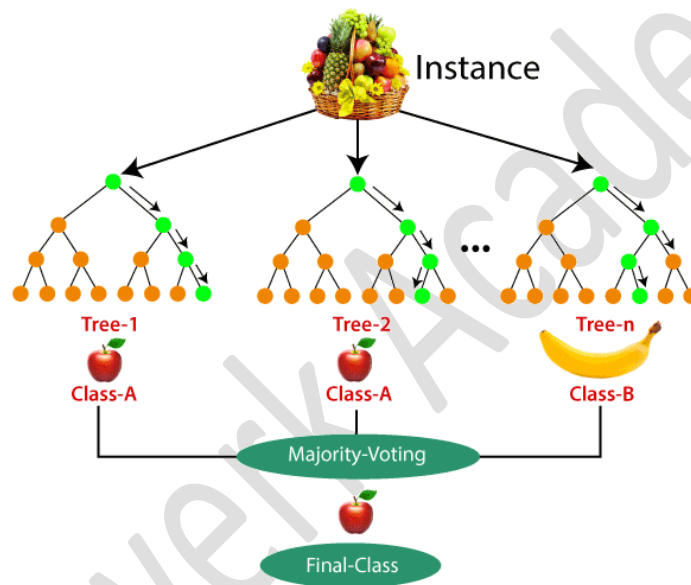
MACHINE LEARNING

The two principal components will explain most of the variance which is explained by the features which are present in the original dataset. Similarly the higher dimension features can be converted using PCA to get 2,3,4,...n components, where n is less than the original number of features.

Random Forest

Introduction

Random forest is a supervised learning algorithm which can be used for classification as well as regression problems, but mainly used for classification problems. The name random forest as it comprises of many decision trees which are created on random data samples and the best possible solution is chosen based on the majority voting. Rather than using a single decision tree where there is chance of overfitting, multiple random trees are built, and the results are averaged which overcomes the problem of overfitting.



The group of uncorrelated models produce better predictions than the individual predictions. In the group of predictions, few predictions might be wrong, but many other predictions will be correct and thus will lead to correct predictions.

Working of Random Forest Algorithm

Let us consider a dataset having S observations (rows) and K features.

- 1) Randomly select " S " samples from total " s " samples where $S \ll s$. (For instance, if S is 10000 observations, select s 100).
- 2) From the above selected " S " samples, randomly select " K " features from total " m " features where $k \ll m$. (For instance, if m is 50 features, k would be 5).
- 3) Among the " K " features, calculate the node " d " using the best split point
- 4) Split the node into daughter nodes using the best split
- 5) Repeat the steps 2 to step 4 until " l " number of nodes has been reached
- 6) Build forest by repeating steps 1 to 5 for " n " number of times to create " n " number of trees

Real Life scenario

Imagine one of your friends wants to decide which mobile phone he has to buy. He will have certain criteria to decide, he will approach you. You will ask him certain questions like what the budget is, which brand he is looking for, what is the type of phone, etc. Based on his answers you will create a virtual decision tree which classifies, and you suggest him some mobile.

Here in this case the suggestion given by you is the result of the answer filters. Likewise, he will ask more and more friends, based on the highest number of suggestions he will decide which mobile to buy.

Regularization

Overfitting is a phenomenon that occurs when a Machine Learning model is constraint to training set and not able to perform well on unseen data.

Regularization works by adding a penalty or complexity term or shrinkage term with Residual Sum of Squares (RSS) to the complex model.

Consider below cost function for simple linear regression.

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 .$$

We will add an extra penalty term or shrinkage term which will minimize the effect of overfitting which is caused when the model tries to cover the noise term.

Techniques in Regularization

1. Ridge Regression
2. Lasso Regression

Ridge Regression

Ridge regression is one of the types of linear regression in which we introduce a small amount of bias, known as Ridge regression penalty so that we can get better long-term predictions.

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2$$

We have added an extra penalty term which is lambda multiplied by summation of squared coefficients. This is also called L2-Norm.

When to use Ridge:

- When we have the independent variables which are having high collinearity (problem of multicollinearity) between them, at that time general linear or polynomial regression will fail so to solve such problems, Ridge regression can be used.
- If we have more parameters than the samples, then Ridge regression helps to solve the problems.

Lasso Regression

Lasso regression is another variant of the regularization technique used to reduce the complexity of the model. It stands for Least Absolute and Selection Operator.

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|.$$

In this case the penalty term which is added is an absolute value of coefficients instead of squared terms. This method is also called L1-Norm.

Key differences between Lasso and Ridge

1. Ridge regression helps us to reduce only the overfitting in the model while keeping all the features present in the model. It reduces the complexity of the model by shrinking the coefficients whereas Lasso regression helps in reducing the problem of overfitting in the model as well as automatic feature selection.
2. Lasso Regression tends to make coefficients to absolute zero whereas Ridge regression never sets the value of coefficient to absolute zero.