Linear Regression

A machine learning algorithm based on supervised learning is linear regression. It executes a regression operation. Regression uses independent variables to model a goal prediction value. It is mostly used to determine how variables and forecasting relate to one another. Regression models vary according to the number of independent variables they utilise and the type of relationship they consider between the dependent and independent variables. The job of predicting a dependent variable's value (y) based on an independent variable is carried out using linear regression (x). Therefore, x (the input) and y (the output) are found to be linearly related using this regression approach (output). Thus, the term "linear regression" was coined.

In this mainly an optimal line needs to be found out to fit the data. The best fit of the line is seen by how close the line is from all the data points. The line with least distance is the best fit for a dataset. The total distance between the line and the data point is calculated by the sum of the squared difference between y intercept of the line and the datapoint. This distance from the line to the data point is called a “Residual.”

There are three main points in linear regression least-square to fit the line, calculate R² and p-value for R². First horizontal line is taken into consideration by takin the average of the y intercepts. Then the residuals are calculated for the line. The line is rotated and many residuals are found out for different angles. By this a graph can be plotted for rotation vs sum of squared residuals. From the graph we can find out the rotation which has the least sum of square. That is why this method is called “Least square method.”

The line equation contains a y-axis intercept and a slope. After this R² of the line is calculated by dividing the difference between variance of the mean and fitted values of the variable that needs to be predicted with the variance if the mean. The R² explains the percentage of the variation in the predicted values by the model. This equation can be applied to any equation even if it is more complicated than a simple linear regression equation. The p-value of the R² is to see if the R² value calculate for the data is significant or not. It is calculated by dividing the numerator with the denominator where numerator is the variance explained by the extra parameter and the denominator is the variation that is not explained by the extra parameter in the fit line. After this a histogram is plotted with the output got from this formula for all the data point. Then the p-value is calculated by dividing the extreme values from the histogram with all other values.

By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum.

Decision Tree

Decision tree is a simple flow chart which states a statement and then decides based on the statement weather it is true or false. There are two types of trees the first one is the classification tree, in which the decision tree classifies things into categories. Second type of tree deals with the numerical values, meaning that the tree predicts numeric values which is called Regression tree.

Decision tree classifier building is ideal for exploratory knowledge discovery since it does not require parameter configuration or domain understanding. High-dimensional data may be handled via decision trees. Decision tree classifiers are often accurate. A popular inductive method for learning classification information is decision tree induction.

By dividing the source set into subgroups based on an attribute value test, a tree may be "trained". It is known as recursive partitioning to repeat this operation on each derived subset. When the split no longer improves the predictions or when the subset at a node has the same value for the target variable, the recursion is finished.

The algorithm calculates the sum of square residual of all the possible data points in the dataset. After this the nodes are selected based on the residual values. In the case of dataset with many columns, all the columns are individually compared with the prediction column. And the difference between the regressor line and the data point is calculated. This is done for all the datapoints and the sum of square residual is calculated at each step. By doing this, SSR is obtained for all columns, which is further compared to all the SSR values and the lowest one is selected as the root node. Next node is selected in the same basis, lowest of all the SSR values. All the possible observation are grouped below the nodes. If the there is no further path from the node, then that node is called ‘leaf.’ For prediction of a certain value the tree path is followed till the end, i.e., till there is no further options (leaf). That leaf value is approximately the predicted values.

Pruning can improve a tree's performance even further. It entails eliminating the branches that depend on insignificant qualities. By doing this, we lower the complexity of the tree and, by decreasing overfitting, raise its predictive ability. Further enhancing a tree's performance is pruning. It requires cutting off the branches that rely on unimportant characteristics. By doing this, we reduce the tree's complexity while increasing its capacity for prediction by reducing overfitting.

Random Forest

The bagging approach is extended by the random forest algorithm, which builds an uncorrelated forest of decision trees using both feature randomness and bagging. A random subset of features is generated via feature randomness, also known as feature bagging or "the random subspace method “which guarantees minimal correlation across decision trees. The main distinction between decision trees and random forests is this.

Decision tree is not much flexible with the data used to create them, which means that they are not flexible when it comes to classifying new samples. So, the development of Random Forest took place which could combine the simplicity of decision trees with the flexibility resulting in a vast improvement in accuracy.

First step is making a bootstrapped dataset out of the original dataset. In this random samples are picked out and a small dataset is generated which is called bootstrapped data. In bootstrapped dataset single sample can be selected multiple time in the same bootstrapped dataset.

Now a decision tree is generated using the bootstrapped dataset with the use of a random subset of the variables at each steps including the selection of the root node.

This is followed for different bootstrapped data. Every time a new bootstrapped data is generated and a tree is built on the subset of the variables at each step. Nearly 100s of tree are made. This variety in the tree is the uniqueness of the random forest making it more effective than decision tree.

To predict the value of a sample the sample data is run through all the trees and the results are tabulated. Votes are given to the results obtained from each tree and the observation with the maximum vote is the final result. For the numerical values the mean of all the result is calculated and given as the final result.

When a sample entry does not go in the bootstrapped dataset, that entry is termed as “Out-of-bag-dataset.” This out-of-bag-samples are then tested for all the decision tree and the result is verified. By this the we can measure how much accurate the random forest mode is by the proportion of the out of bag data samples which were correctly predicted by the model.

Decision trees tend to tightly fit all the samples in the training data, which increases the danger of overfitting. The classifier will not, however, overfit the model when there are a large number of decision trees in a random forest since the averaging of uncorrelated trees reduces the total variance and prediction error.

Random forest is mainly used for the application such as evaluating customers with high credit risk, fraud detection and healthcare applications which helps doctors in bioscience and medications.

Gradient boosting

Gradient boosting model is one of the powerful models for analysing of the data. In the field of machine learning this model is one of the most used as it has a boosting factor which reduces the error. As is well known, bias and variance mistakes may be used to broadly categorise errors in machine learning systems. As one of the boosting strategies, gradient boosting is used to reduce the model's bias error. Both continuous and categorical target variables may be predicted using the gradient boosting approach (as a Regressor) (as a Classifier). The cost function is Mean Square Error (MSE) when it is used as a regressor, while it is Log loss when it is used as a classifier.

Gradient boosting model starts with the AdaBoost model, in which a single layer tree is made with only one split. Then the Adaboost make another tree based on the error on the previous stump. Like this number of trees are made one after another from their previous stumps.

On the other hand, Gradient Boost makes a start with a single leaf. In this the first leaf is taken as the mean of all the samples which we are trying to predict, that could be a continuous value in case of regression and other for classification. Then the tree is made same as Adaboost, i.e., it makes the tree based on the previous errors but the size of the tree is larger than the AdaBoost.

Gradient boost makes all the tree equal to a fixed size. This is not done in the case of AdaBoost, in which the tree can be bigger than the previous stump. The scaling of the tree occurs in both the boosting models but in Gradient Boosting the scaling factor is done on same by the same amount for all the tree. Mostly the number of leaf’s present in tree are from 8 to 12. In this way the tree is made from the previous error and then scaled. Tree is continuously made till the number of tree user has asked are not created.

In Gradient boosting the problem of overfitting the model is dealt by using learning rate to scale the contribution from the new tree. The value of the learning rate is between zero and one. Learning rate makes the prediction little better than the original leaf. Taking small steps in the right direction results in better prediction with a testing data, i.e., low variance.

While making of the tree from the previous error Pseudo residuals is found out by calculating the difference between the observed and the predicted values. A new column is made by calculating the residual for all the samples with the predicted values from a tree. Now a new tree is made from tis residual values. The tree will have the same learning rate as the previous tree. Now we combine all the tree with the new tree and prediction is made. By this method we get new residuals. Each time a tree is added to the prediction the residual gets smaller. Like this the chain of tree is added till we get the maximum specified, or adding additional trees does not significantly reduce the size of the Residuals.

In short, we start with the mean value as the leaf, after which we add tree based in the residuals, the difference between observed and the predicted values and scaled by the learning rate. Tree is added based on the previous errors.

Extreme Gradient Boosting

Extreme gradient boosting model is one of the most use models in machine learning. The theory of extreme gradient boosting is a big machine learning algorithm with lots of steps and part which makes it capable of extreme boosting of the model. They have a unique methos for creating of decision tree. Mainly XGBoost was developed for handling big and complicated data.

First step in XGBoost is to take the initial prediction by fitting the training data into the model. By default, this prediction is equal to 0.5, This value is same for XGBoost regression and classification. A straight line for the initial predicted value is plotted on the graph and residual, i.e., the difference between the observed and predicted values is calculated. This plotting of the graph shows us how good is the initial prediction compared to the dataset.

Unlike the gradient tree, XGBoost uses a unique tree. There are many ways to build a tree for XGBoost. The most common way for building a tree for XGBoost regression is by starting each tree as single leaf. This single leaf contains all the residuals of the dataset. Mainly default lever of the tree is 6. After this a Quality score or Similarity score is calculated for the residuals of the data.

Similarity score is calculated by squaring the sum of the residuals and dividing it by the sum of number of residuals and λ(lambda). ‘λ' Lambda is a regularization parameter which is present to reduce the vulnerability of the prediction to an individual observation. In short λ(lambda) prevents the overfitting of the training data. So, in the numerator the sum of the residual is calculated by add all the values that are above and below the initial prediction value. Values below the initial predicted values come out to be negative while the values above the initial predictive data is positive in nature. As the squaring is done after the addition of all the residuals, they cancel out each other which means the sample residuals with same values but on the opposite side of each other cancel each other due to positive and negative nature. So, like wise similarity is found out for the leaf.

Now we cluster the similar residuals by splitting it into groups. For doing this first few residuals are taken into consideration and average of predict value it calculated. So, the split is done according to the average value, i.e., greater, and less than the average value. The smaller values than the average value goes in the left leaf and all the greater values go in the right leaf. Then same Similarity score is computed for both the right and left leaf.

Similarity score mainly depends on the number of residuals and the they cancel each other making the Similarity score relatively less. On the contras when the residuals are similar or only one is present in the leaf node, the Similarity score comes out to be relatively large.

After this a term ‘Gain’ is added to the tree, Grain is calculated by subtracting the Similarity score of the root node from the sum of Similarity score of the right and left node. Likewise, tree is created by taking average of next few residual samples till the average thresholds reaches the last samples.

After calculating the gain value of all the average threshold, they are compared and the threshold with the highest gain value comes out to be best at splitting the residuals into cluster of similar values.

To further divide the tree leaf, the residuals present in the leaf are only considered and the whole process if again followed for further bifurcation the leaf node. The Default levels of the tree is 6, but can be changed accordingly.

After the tree is created ‘Pruning’ comes into picture. Pruning in XGBoost is done based on the grain value. A new variable is ‘γ’ (gamma) is set with a number. After which the difference between the gain of the lowest branch of the tree and gamma is calculated. If the result comes out to be negative then the branch is removed. The branch is kept if a positive result is the outcome. Likewise, the pruning is done for all the branches in the tree.

At last, for making the prediction the average of the elements present in the end leaf node are taken as the output value. If there is only one element then it is taken as it is as the output value. So, when the value is to be predicted for a sample, the starting initial prediction is added to the product of learning rate and the output value got from the tree. In XGBoost the learning rate is default 0.3, It is called ε (eta). By this a new residual is obtained which are smaller than before. Likewise new tree is made based on the new residual. This is done till the required number of trees are created or till the residual are super small.

KNN

The KNN algorithm can be applied to issues involving classification and regression. To forecast the values of any new data points, the KNN algorithm makes advantage of "feature similarity." In other words, the value given to the new point depends on how much it matches the points in the training set. KNN is famous for classification problems but it works fine with continuous values. Working of KNN is the simplest of all the machine learning models. Despite being simple, the capacity of the model is very effective on a certain task.

In this algorithm the data is stored with all the samples based on the similarities. This helps to classify the new set of data into categories easily. In the case of regression, the sample is just plotted on a graph to apply the k nearest algorithm on the data. There are two types in KNN one is non-parametric in which not many assumptions are made on the data. In this the model does not learn anything from the training set instead it stores the data and then performs action when the prediction needs to be done. While in the parametric algorithm few if the assumptions are made based on the data set samples which are given as the training set for the model.

So, the working of the model starts by defining the value of ‘K’ which indicates the number of neighbours to be considered while calculating the prediction regression value. The KNN technique may be applied to issues involving classification and regression. To forecast the values of any new data points, the KNN algorithm makes advantage of "feature similarity." In other words, the value given to the new point depends on how much it matches the points in the training set. The value of K needs to be defined properly as the number of k nearest neighbour of an instance will predicted the output value. If the K value is too small then very few samples will be taken into consideration and the error may be very less i.e., the model may overfit the training dataset. On the other hand, if K value is high then many sample will be grouped together and the result calculated will vary by a high value, i.e., it may underfit the training dataset.

The distance between the instance and the data samples is calculated by distance formula by two methods one is Manhattan distance formula and the other is Euclidean distance formula. After calculating the distances, k number of nearest points are taken into considerations. All the values nearest to the instance are grouped and the average of them is given as the output value of the instance the needs to be predicted.