TABLE OF CONTENTS

|  |  |
| --- | --- |
| **Topic** | **Pool** |
| Linear Regression | 1 |
| KNN | 1 |
| K-Means | 1 |
| Decision Tree | 1 |
| Random Forest | 2 |
| Gradient Boost | 2 |

## LINEAR REGRESSION

Linear regression is a type of predictive analysis where we try to find out a linear relation between a set of input variables and a target value. The target value is numeric. It helps determine whether the chosen inputs/predictors help predict the outcome correctly or not. Its not possible to accurately determine the outcome with a set of predictors. There will always be error. The linear regression model must be good such that this error/offset with respect to the actual output is minimum. If in a linear regression model, only a single predictor is chosen to determine the outcome, it is referred to as Simple linear regression. For any linear model that has more than one predictor, it is a multiple regression.

MODEL:

**Y = B1X 1+ B0** for simple linear regression

**Y = B1X1 + B2X2 + B3X3 + … + B0** for multiple linear regression

where Y = target, X1 = predictor, B1 = coefficient of predictor, B0 = intercept

In this approach, we get coefficients of the predictors and the intercept of the model. The model must have at-least one slope that is nonzero. The following statistic values help in determining

* if the model provides a better fit than one that has no independent variable,
* there is at-least one slope which is nonzero.

**F TEST:**

F test compares the model with a one that doesn’t have independent variables. If the value of the regression is significant, means the model fits the data better than a model without independent variables. Here all the coefficients of the predictors are assessed jointly. If the F test is significant, it means the coefficient of determination is not zero and the correlation between the model and the dependent variable is significant. The degrees of freedom indicate the number of independent variables.

**T TEST**:

It is the coefficient divided by standard error. Standard error is estimate of standard deviation of coefficient. If the coefficient is large compared to its standard error, then it’s different from 0. The P value indicates whether the slope is significant. The value of the coefficient gives the effect it has on the dependent variable i.e it tells how much the dependent variable changes with the change in the independent variable.

**Coefficient of Determination**:

It’s the percentage of variation explained in the target variable that is explainable by the predictor. It is a metric for evaluating performance of a regression model.

R2 = 1 – (Sum square of errors)/(Sum Square Total)

Sum of square errors = ∑(y-yhat)2

Sum Square Total = Sum of Square Errors + Sum of squared deviations of predicted(SSR)

There are four key assumptions that must be satisfied for a linear regression model. They are as follows

* The model must be linear. Data points must be scattered above and below the residual line.
* The error terms are independent. Plot against time.
* The error terms have constant variances.
* The error terms are normally distributed. Verify with help of Quantile-Quantile plot. Quantiles are cut points dividing the probability distribution space into intervals with equal probabilities.

Non – normality and heteroscedasticity can be fixed by applying transformation such as log.

**Outliers**:

In a given data set, we may have points that tend to influence the model entirely on its own. Such points are usually the extremes. For a good model, it’s imperative that such points are removed.

Cooks distance: It’s an estimate of influence of a data point. It identifies those points that negatively affect the regression. It’s a combination of each observations, residual and leverage. Higher the leverage and residual, higher is the cooks’ distance.

Leverage(h) = (1 + z\*z) /n

Cooks Distance (D) = ( (standard residual)\*(standard residual)\* h)/p\*(1-h) where p is the number of parameters.

**METRICS**

Following are the metrics that can be used to evaluate the predictions made by the model

Mean Squared Error

It’s the average of the squared difference between the target value and predicted value by the model

MSE = ∑(y-yhat)2/n

Root Mean Squared Error

It’s the square root of MSE. This is useful where large errors are undesirable.

RMSE = √MSE

Mean Absolute Error

Difference between the target value and predicted value. Its more robust to outliers and does not penalize errors extremely.

MAE = (1/n) \* ∑ |y-yhat|

Adjusted R2

R2 suffers from the problem that it increases with the increase in terms even though may not have improved. This issue is handled by Adjusted R2

**Code:**

from sklearn.linear\_model import LinearRegression

regressor1 = LinearRegression()

regressor1.fit(X\_train, y\_train) # training the model. Using this, predictions made.

## K NEAREST NEIGHBORS (KNN)

KNN is a supervised learning algorithm that can be used in a classification problem or regression problem. Ideally used in classification problems. In this model approach, we try to determine the state/value of a given point depending upon its immediate neighbors which are similar. For a classification problem, the output is a discrete value whereas for a regression problem it’s a real value. The ‘K’ stands for the number of neighbors. KNN is mostly used in problems related to finance, healthcare, Handwritten detection, predicting credit ratings of customers, classifying potential voters. KNN can also be used to impute missing categorical and continuous variables.

**MODEL:**

The structure of the model is determined from the data set. All of the training data is used in training phase. The training phase is much costlier and slower. More time and memory required. The computational complexity for classifying new samples grows with the number of samples in the training dataset. KNN works best if the model is normalized i.e between values between 0 and 1.

**WORKING:**

KNN works on the assumption that similar data points are close to each other. Here similarity is measured in terms of distance. Distance can be obtained by any of the following

* Euclidian
* Hamming
* Manhattan
* Minkowski

Euclidian is a good measure if the inputs are of the same type. Manhattan distance is good to use if input variables are not of same type. Below is an example of how a new point is classified into either class A or class B depending upon it’s neighbors similarity.

A close up of text on a white background

Description automatically generated

The variance/bias changes with the change in the number of neighbors in the algorithm. As the value of K increases, the variance decreases and bias increases. Also, with the increase in ‘K’, the boundary that separates one class from another becomes more distinguished.

A close up of a map

Description automatically generated

As we can see in the diagram above, the red line represents the error on training data while the green line represents the error on test data. For test data, at k=1, we are overfitting. Hence the error rate initially decreases and reaches a minima then increases with increasing K.

**Optimal value of K:**

K is a hyper parameter in this algorithm. We can either use cross validation or grid search to find the optimal value of K. To begin with K=1, the model has high variance. As the value of K increases, the model complexity also increases.

KNN is also referred to as Lazy learning as no learning of the model is required and all work happens at time of prediction.

**Curse of dimensionality:**

KNN performs better with lower number of input features. When number of features increases, more data is required. When we increase the amount of data, it leads to overfitting. To avoid overfitting, data has to grow exponentially as we increase the number of dimensions.

In order to solve this problem, we can use PCA before applying the KNN algorithm. In large feature dataset, Euclidean distance is of no use. We can use cosine similarity instead.

**Code:**

* To build a classifier

A screenshot of a social media post

Description automatically generated

Above, we have built a general KNN classifier with k=5 and using the Euclidean distance to calculate similarity. Here we have overridden the default algorithm and metric.

* To build a classifier using Grid search

A screenshot of a social media post

Description automatically generated

Above we have used Grid search to find the optimal value of K. Also we are instructing the algorithm to do parallel computing.

## K MEANS

K Means is an unsupervised clustering algorithm. It takes a set of data points and tries to group them into ‘K’ number of clusters. It is unsupervised because the points do not have external classification.

**Working:**

Depending on the number of clusters specified i.e K, we will randomly pick K points as centroids. Centroid is a cluster center. Ex: If we K=2, then we will have 2 centroids. Following are steps how K Means works

STEP 1

Rows grouped into K number of clusters

STEP 2

Randomly generate K cluster centers

STEP 3

Assign each observation to the nearest centroid

STEP 4

Observations are colored as per the color of the closest cluster centroid

STEP 5

Recompute the centroids of clusters

STEP 6

Repeat step 4 as the centroids are updated

Continue updating the centroids until the cluster centroids stabilize i.e no change.

A close up of a piece of paper

Description automatically generated

A close up of text on a white background

Description automatically generated

A close up of text on a white background

Description automatically generated

In the above diagrams, have briefly illustrated the way how KNN works.

**Hyper parameter**

The number of clusters ‘K’ is the hyperparameter. Ideal value of K can be chosen by the following ways

* Elbow plot: It’s a plot of number of clusters vs sum of squared distances from each point to its cluster.

A close up of a piece of paper

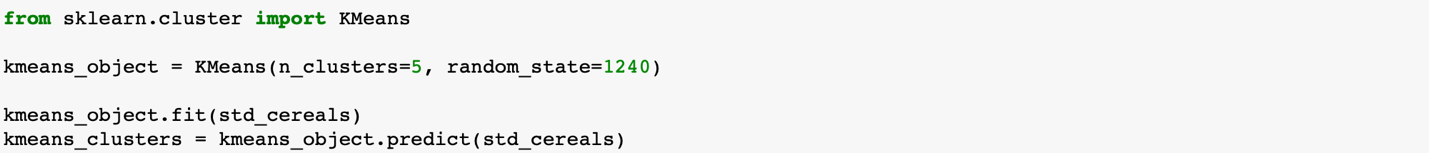
Description automatically generated

* Silhouette Analysis: method of interpretation and validation of consistency within clusters of data. The Silhouette value is a measure of how similar a point is to its own cluster compared to other clusters. The Silhouette score gives the average value of all samples.

A close up of text on a white background

Description automatically generated

**Code:**



Here we have passed 2 arguments to the K-Means initializer, the number of clusters and random state(seed). We can also pass the following to the initializer

* n\_init: number of times algorithm will be run with different centroid seeds.
* max\_iter: maximum number of iterations per run.
* n\_jobs: for parallel computing

## DECISION TREES

This is a supervised learning approach. Here, the feature space is greedily divided into distinct non overlapping regions. As the name suggests, it has a tree like structure with root as the starting point. Each branch represents an outcome of the test and each leaf node represents holds a value. The value could be categorical or numerical. Decision tree algorithms are referred to as CART i.e classification and regression trees.

**Working**:

Decision trees work by splitting the data into homogeneous regions. For this purpose, each point in the feature space is picked and a split is performed. Impurity refers to a node having more than one type of classification. The split that results in less impurity is chosen. At every split, the decision tree ensures that the impurity is reduced in a node so that it gives a more homogeneous set of nodes. At a given point in time, the decision tree considers only the current split i.e input variable along which the feature space is split. We can illustrate this with Titanic data set shown below

A picture containing text

Description automatically generated

Here, ‘**is sex male**’ is chosen as the root node based on which the split takes place as this resulted in less impurity. ‘**Died**’ and ‘**Survived**’ are the two classes or target variable values. These are represented as the leaf nodes in the Decision Tree. The result of a split is either a leaf node or another decision split node. In this case, ‘**is age > 9.5**’ is another decision node. Similarly, ‘**is sibsp > 2.5**’ is another decision node. Here we also observe that, the decision of an input variable on which split is made is either binary or continuous value.

**Impurity Measures**:

Impurity can be measured in any of the following ways

* Classification error rate: **1 – max(probability of I, probability of j)**
* Gini Index: **∑ (probability of j) \* (1 – probability of j)**
* Cross Entropy: **∑ (probability of j) \* log(1/probability of j)**

**Recursive binary splitting**:

* All the features are considered
* Different split points are tried and tested using a cost function. The split with the best cost function is selected. Makes the root node the best predictor/classifier

**Pruning**:

As the depth of the tree increases, it results in overfitting. This issue can be resolved by pruning. Here we remove the branches that make use of features that have low importance. Pruning can start at the root or at leaf. It reduces the size of the tree without reducing the predictive accuracy. There are a couple of types of pruning

* Reduced error pruning: starting at the leaf, each node is replaced with its most popular class.
* Cost complexity pruning: learning parameter (α) is used to weigh whether nodes can be removed based on the size of the sub tree.

**Hyper parameters**:

* Criterion: the cost function i.e gini index or cross entropy
* Splitter: the strategy to choose split at each node. Supported strategies are ‘best’ and ‘random’.
* Max\_depth: the maximum depth of the tree. Default value is none i.e nodes split until all are pure. The deeper the tree grows, the more complex the model becomes. This results in overfitting.
* Min\_samples\_split: minimum number of samples required to split a node.
* Min\_samples\_leaf: minimum number of samples required at leaf node.
* Max\_features: number of features to consider when looking for best split.

**Code**:

The above hyperparameters are passed to the Decision Tree initializer. It could a classifier or regression

A screenshot of a social media post

Description automatically generated

## RANDOM FOREST

Random forest is a supervised learning model. Decision tree is the base of the random forest model. It is an ensemble of ‘n’ number of decision tree. The trees in Random Forest run parallel i.e there is no interaction between the ‘n’ decision trees. This is a bagging technique as it prevents overfitting. Random forest can be used in classification or regression problems. The final output of the Random Forest model is result of collective responses of the ‘n’ decision trees. It is fast to train with test data.

**Working:**

Before applying the Random forest algorithm, we need to ensure any categorical variable is encoded. We can use One-Hot encoding. A large number of uncorrelated trees working as a group will outperform any individual model. The result model of all these ‘n’ trees would have been trained on different sets of data. Here, we are not sub-setting the data i.e we don’t provide chunks of data from original data set to the ‘n’ decision trees. Instead, we provide each tree with the same size of data as the original dataset along with replacement. Each tree in the random forest can pick only from a random subset of features. This ensures that there is variation in the trees that make up the model and that the model does not rely on any individual feature. As a result of this, there will be low correlation between the trees.

The number of features that can be split on at each node is a percentage of the total number of features. This is controlled by a hyperparameter. For a classification problem, the output of the Random forest model is the class with maximum number of occurrences in the decision trees. For regression, it is the average value of all outcomes of the decision trees.

Mathematical formula:

For regression problems, we used Mean squared errors to decide which branch is better for the forest.

A screenshot of a cell phone

Description automatically generated

For classification problem, we use

* Gini Index
* Cross Entropy

**Hyper parameters:**

* n\_estimators: number of trees in the forest
* max\_features: maximum number of features considered for splitting a node
* max\_depth: maximum depth of each tree
* min\_samples\_split: minimum number of data points in a node before it is split
* min\_samples\_leaf: number of data points in a leaf
* bootstrap: method for sampling data. With or without replacement.

We can use the above hyperparameters in model building using Grid search.

**Code:**

A screenshot of a cell phone

Description automatically generated

## GRADIENT BOOST

Boosting is an ensemble in which the predictors are not made independently but sequentially. In this technique, the subsequent predictors learn from the previous predictors. Because new predictors are learning from mistakes committed by previous predictors, it takes less time/iterations to reach close to actual predictions. Gradient boost can be used for regression and classification problems. It produces a model which is an ensemble of weak prediction models.

**Working**:

The main idea behind GBM is to set target outcomes for the next model to minimize the error. The target outcome for each case in the data set depends on how much a change in that case’s prediction impacts the overall prediction error i.e whether it tends to minimize the error or not. Following are the steps involved

* Initialize the outcome
* Iterate from 1 to total number of trees   
  a) Update the weights for targets based on previous run (higher for the ones mis-classified)   
  b) Fit the model on selected subsample of data   
  c) Make predictions on the full set of observations   
  d) Update the output with current results taking into account the learning rate
* Return the final output.

**Hyper parameters**:

* learning rate: Determines impact of each tree on final outcome. GBM works with an initial estimate which is updated with the output of each tree. The learning rate controls the magnitude of change in these estimates
* n\_estimators: number of sequential trees to be modelled
* subsample: Fraction of the observations that need to be selected for each tree. Selection is done by random sampling.

**Code:**

