|  |  |
| --- | --- |
| Regularization | * bias in a model is high when it does not perform well on the training data itself, and variance is high when the model does not perform well on the test data. Please note that a model failing to fit on the test data means that the model results on the test data varies a lot as the training data changes. This may be because the model coefficients do not have high reliability. * Another point to keep in mind is that the model coefficients that we obtain from an ordinary-least-squares (OLS) model can be quite unreliable if among all the predictors that we used to build our model, only a few are related significantly to the response variable * An underfitting model is not a suitable model, as it will show poor performance with the training data. Usually, a model that is an underfit would have high training as well as testing error. |
|  | * The ‘linear’ in linear regression does not stand for the relation between the target variable and the predictor variable. In fact, it stands for the coefficients of the predictor terms in the linear regression solution. It is entirely logical to have a linear regression solution as z= a\*sinx + b\*cosy, however, it will not be a linear regression model if the relation can be defined as y=2a\*y\*x^\*sinx + y^2\*x^2\*sinxy. |
|  | * Overfitting and non-linearity in data can be checked with residual analysis. Value of bo & b1 after taking the derivatives:   + - bo=¯y−b1¯x  and     - b1= (∑ni=1(x−¯x)(y−¯y))/ (∑ni=1(x−¯x)2)     - where ¯y and ¯x are the sample means. |
|  | * For Simple Linear Regression   + Plot the independent variable against the dependent variable to check for nonlinear patterns. * For Multiple Linear Regression, since there are multiple predictors, we, instead, plot the residuals versus the predicted values. Ideally, the residual plot will show no observable pattern. In case a pattern is observed, it may indicate a problem with some aspect of the linear model. Apart from that:   + Residuals should be randomly scattered around 0.   + The spread of the residuals should be constant.   + There should be no outliers in the data   If nonlinearity is present, then we may need to plot each predictor against the residuals to identify which predictor is nonlinear. |
|  | * Non-linearity is determined by co-efficient B0 & B1 i.e. response variable has to be of linear relationship of co-efficient. * After applying any equation data-transformation, polynomial or non-linear regression residual plots has to be analysed against predictor |
|  | * If there is a non-linear trend in data, first thing we do is to transform predictor variable * When the problem is the non-normality of error terms and/or un-equal variances are the problems, then consider transforming the response variable. When we transform the response variable, we will change both its errors and variance. Hence, we should transform the response variable only if the error terms are not normal or if the residuals exhibit non-constant variance, as seen in the residual plots * When the regression function is not linear and the error terms are not normal and have un-equal variances, then transform both the response and the predictor. * Polynomial and data-transformation both are linear in terms of the parameters (i..e linear in terms of the beta’s). For models where response variable is related non-linearly with the parameters or the model co-efficient, we use non-linear regression. |
|  | * A model complexity is determined by magnitude of co-efficient or number of co-efficient * Regularization “Cost function = RSS + penalty” * With regularization, models work well on unseen data without losing out on identifying they underlying patterns in the data. |
|  | * In ridge/lasso regression, for each value of lambda we get different values of model coefficients. * For high lambda the model will underfit, less lambda model will overfit. (bias/variance trade-off) * Cost function  1. RSS = square sum of error terms (linear regression) 2. Cost function = RSS + lambda \* (square sum of co-efficient) 3. Cost function = RSS + lambda \* (sum of absolute values of coefficients) |
|  | * Generally, Lasso should perform better in situations where only a few among all the predictors that are used to build our model have a significant influence on the response variable. So, feature selection, which removes the unrelated variables, should help. But Ridge should do better when all the variables have almost the same influence on the response variable. * Overfitting can be solved by using regularisation. |

|  |  |
| --- | --- |
| Tree Models   * Decision trees * Random forest | * Decision trees and random forests can be used to solve both classification and regression problems. * Ensemble is called a collection of models. * Decision trees mimic human decision-making process and are efficient in dealing with categorical data. * Unlike other algorithms such as logistic and support vector machine, decision trees do not help in finding a linear relationship between the independent variable and the target variable. However, they can be used to **model highly non-linear data**. * Random forest is very successful algorithm for classification. |
|  | * **Root node** is top most node. Arrow points away from the node * The node that cannot be further classified or split is called the **leaf node/terminal node.** * The intermediate nodes between the root and the leaf nodes are called **internal nodes**. * If all leaf nodes are perfectly homogenous i.e. they no error in classifying the labels training error is considered as zero. |
|  | * In decision tree, if an attribute is nominal categorical, then there are (2\*\*(k−1) −1) possible splits where k is the number of classes. * In an attribute is ordinal categorical or continuous in nature with n different values there are (n-1) different possible splits. Each value of the attribute is sorted from smallest to the largest, and candidate splits based on the individual values is examined to determine the best split point which maximizes the homogeneity at a node. * Algorithm: CART algorithm, calculating percentiles and midpoints of the sorted values for handling continuous features in different algorithm and this process is known as discretization. |
|  | * Homogeneity of the resultant partitions is measured by methods such as “classification error, gini index and entropy”.   + 1. Classification error: 1- max(pi)     2. Gini index = sum of Pi \* (1-Pi) = 1 – sum of Pi\*\*2     3. Entropy = -1 \* sum of [pi \* log(Pi) where base is 2] * Entropy and Gini index is high when impurity is high. * Higher the homogeneity (i.e. low impurity) lower the. Entropy * Change in impurity(purity gain) = impurity (pre-split) – Impurity\*(post-split) |
|  | * Recall that the Gini index is calculated as follows: * G=∑k/i=1. Pi\*(1−pi) = 1−∑k/i=1 pi\*\*2 * where pi is the probability of finding a point with the label i, and k is the number of classes. |
|  | * Homegeneity > Threshold then stop else go.. think in terms of similar data points it will be less homogeneous then keep splitting. |
|  | * Bagging works well with high variance algorithms and is easy to parallelise. By high variance, we mean algorithms which change a lot with slight changes in the data as a result of which these algorithms very easily overfit if not controlled * Disadvantages: Loss of interpretability , Also, it does not work well when any of the features dominate because of which all the trees look similar and hence the property of diversity in ensembles is lost. Sometimes bagging can be computationally expensive and is applied depending on the case. |

|  |  |
| --- | --- |
| Ad boost classifier | * The most important parameters are base\_estimator, n\_estimators and learning\_rate. * 1. base\_estimator - It is the learning algorithm to use to train the weak models. The default Learning Algorithm is DecisionTreeClassifier with Max Depth of 1 * n\_estimators - It is the number of models to iteratively train. * learning\_rate - It is the contribution of each model to the weights and default value for it is 1. There is a trade-off between learning\_rate and n\_estimators. Reducing the learning rate will forcing the model train slower (but sometimes resulting in better performance scores. Decreasing the learning rate L makes the coefficients α\_m smaller, which reduces the amplitude of the sample\_weights at each step (As per weight formula we use at each step for updating weights. |
| Adaboost | * Assigns weights to each of the data points and model, where weights to each data points are same in beginning; for model making more error or assigned less weights in final prediction. * Every model works on errors made by previous learner. Data points which has been mis classified assigned more weights. Next learner work on resampled data (i.e. some of data points can appear multiple times as a row in training data) |
|  | * Chart    Description automatically generated with medium confidence |
|  | * Lowest gini impurity = max gini score * Find variable which results in max gini score/lowest gini impurity that becomes root node * Find initial weight 🡪 calculate (total error) 🡪 calculate (alpha) 🡪 calculate new weights 🡪 calculate new normalized weights * Total error = data points mis-classified/number of data points * Alpha (say/importance) = 0.5\*ln ([1 - total-error]/total-error) * New weights = initial weight \* e (\*\* alpha) where alpha will be (+) for in-correct classified and (-) for negatively classified. * Normalized weight = new weight/(sum of new weight) * Create a distribution range and get the random value of x between 0 & 1 * Choose the row and put in res-smapled data (do it for n observation so it will have more weightage.) * Again initialize the initial weight = 1/n (nb of data points) |
|  | * Chart, line chart    Description automatically generated |
|  | Table  Description automatically generated |
| Gradient boosting | * In gradient boosting we create decision tree based on residual |
|  | * Take average of target variable (This is our base mode – initial weight)🡪 Calculate residual (error: target-average value) 🡪 make decision tree based on residual 🡪 calculate average for nodes if it has more than one sample on leaf node (column residual in below diagram)🡪 sample ending on any of the leaf node will have new residual value defined from here 🡪 calculate new predicted (average target of response + 0.1(LR) \* residual error) = **New prediction = Initial prediction + Learning Rate\*Residuals (For ex:** 27.0 + 0.1\*(-12.7), ) * Start process again make tree on residual * Buidl sequential model to minimize losses, since our model is based on residuals * Final model = 27.7(base model) + 0.1\*-12.7(residual) + 0.1\*-11.4 (N. residual) * i.e. F(t+1) = F(t) + lambda\*h(t+1) |
|  | Table  Description automatically generated |
| Gradient boosting classification: | * calculate odds = log(survived/not-survived) * initial prediction = e \*\* log(odds)/ [1 + e\*\* log(odds)] |
|  | * assign initial prediction 🡪 calculate residual (actual – predicted probability) 🡪 build trees with this residual 🡪 calculate leaf output where the sample ends 🡪 Calculate new log odds = [initial log(odds) + 0.6(LR)\*(-leaf output)] 🡪 Calculate probability of surviving/new prediction (use the formula above of initial prediction i.e. *e \*\* new log(odds)/ [1 + e\*\* new log(odds)])* 🡪 calculate new residuals (from intial survivided value) 🡪 plot the tree and calculate process again * final prediction = initial log odds (0.29) + 0.6\*(h1) + 0.6\* (h2) |
|  | Table  Description automatically generated  Output value of a leaf = sum of residuals / sum of [previous probabbility \* (1 – previous probability)] |
|  | Table  Description automatically generated |
|  | Table  Description automatically generated  New probabiilty fo surviving. |
|  | Table  Description automatically generated |
| XG Boost | * Calculate residual = actual value -predicted value (0.5 assumed with an arbitrary prediction) For ex: -8.5 -.5 = -.9   Make decision tree based on time and calculate similarity score based on residual   * Calculate similarity score = squared of [sum of residuals] / [number of residuals + lamba(regularization param)]   = (-.9 + 5.5 – 7.5 + 7.5) / 4 + Y(assumed zero) 🡺 3.06  Make decision tree based on time and calculate similar score for each node. *Our aim is to maximize it*.  Calculate the gain = [sum of similarity score of splite nodes]) - previus node similarity score   * Further split based on time and calculate similarity score, aim is to maxiize the gain. If (gain -y > 0) split further else tune * Calculate output value of a leaf node = (sum of residuals / number of residuals + lambda) * Final prediction = initial preduction + learning-rate \* output value * Calculate new residuals and repeat the process [marks – prediction]   Regularisation parameter (lambda and gama) where (gain – gama) > 0 split it else prune it |
|  | Chart  Description automatically generated |
|  | Diagram  Description automatically generated |
|  | Diagram  Description automatically generated |
|  | A picture containing diagram  Description automatically generated |