**Machine Learning 5**

1. **R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?**

Ans: [R-squared](https://statisticsbyjim.com/glossary/r-squared/) is a goodness-of-fit measure for linear [regression](https://statisticsbyjim.com/glossary/regression-analysis/) models. This statistic indicates the percentage of the variance in the [dependent variable](https://statisticsbyjim.com/glossary/response-variables/) that the [independent variables](https://statisticsbyjim.com/glossary/predictor-variables/) explain collectively. R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale.

After fitting a linear regression model, you need to determine how well the model fits the data. Does it do a good job of explaining changes in the dependent variable? There are several key goodness-of-fit [statistics](https://statisticsbyjim.com/glossary/statistics/) for [regression analysis](https://statisticsbyjim.com/glossary/regression-analysis/). In this post, we’ll examine R-squared (R2 ), highlight some of its limitations, and discover some surprises. For instance, small R-squared values are not always a problem, and high R-squared values are not necessarily good!

**R-squared and the Goodness-of-Fit**

R-squared evaluates the scatter of the data points around the fitted regression line. It is also called the [coefficient](https://statisticsbyjim.com/glossary/regression-coefficient/) of determination, or the coefficient of multiple determination for multiple regression. For the same data set, higher R-squared values represent smaller differences between the observed data and the fitted values.

R-squared is the percentage of the dependent variable variation that a linear model explains.

1. As we discussed before, RSS gives us the total square of the distance of actual points from the regression line. But if we focus on a single residual, we can say that it is the distance that is not captured by the regression line. Therefore, RSS as a whole gives us the variation in the target variable that is not explained by our model.

### Calculate R-Squared

Now, if TSS gives us the total variation in Y, and RSS gives us the variation in Y not explained by X, then **TSS-RSS gives us the variation in Y that is explained by our model**! We can simply divide this value by TSS to get the proportion of variation in Y that is explained by the model. And this our **R-squared statistic**!

R-squared = (TSS-RSS)/TSS

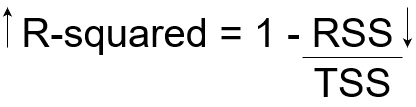
                    = Explained variation/ Total variation

                    = 1 – Unexplained variation/ Total variation

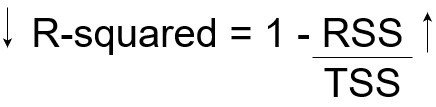
So R-squared gives the degree of variability in the target variable that is explained by the model or the independent variables. If this value is 0.7, then it means that the independent variables explain 70% of the variation in the target variable.

R-squared value always lies between 0 and 1. A higher R-squared value indicates a higher amount of variability being explained by our model and vice-versa.

If we had a really low RSS value, it would mean that the regression line was very close to the actual points. This means the independent variables explain the majority of variation in the target variable. In such a case, we would have a really high R-squared value.



On the contrary, if we had a really high RSS value, it would mean that the regression line was far away from the actual points. Thus, independent variables fail to explain the majority of variation in the target variable. This would give us a really low R-squared value.



So, this explains why the R-squared value gives us the variation in the target variable given by the variation in independent variables.

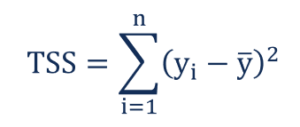
1. **What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.**

### Ans: 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](https://corporatefinanceinstitute.com/resources/knowledge/terms/dependent-variable/) from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a [sample](http://www.webmath.com/sampledata.html). It can be determined using the following formula:



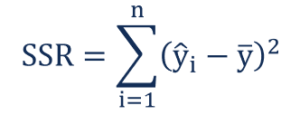
Where:

* yi – the value in a sample
* ȳ – the mean value of a sample

#### 2. 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:



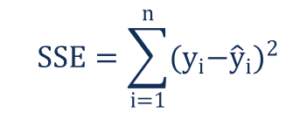
Where:

* ŷi – the value estimated by the regression line
* ȳ – 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:



Where:

* yi – the observed value
* ŷi – the value estimated by the regression line

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

Relationship Formula

1. What is the need of regularization in machine learning?

Ans: In mathematics, statistics, finance, computer science, particularly in machine learning and inverse problems, regularization is the process of adding information in order to solve an ill-posed problem or to prevent overfitting. Regularization applies to objective functions in ill-posed optimization problems.

Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values.

In general, regularization means to make things regular or acceptable. ... In the context of machine learning, regularization is the process which regularizes or shrinks the coefficients towards zero. In simple words, regularization discourages learning a more complex or flexible model, to prevent overfitting.

1. **What is Gini–impurity index?**

Ans: Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. But what is actually meant by 'impurity'? If all the elements belong to a single class, then it can be called pure.

Gini index and entropy are the criteria for calculating information gain. ... Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure. Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder.

1. **Are unregularized decision-trees prone to overfitting? If yes, why?**

Ans: 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 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.

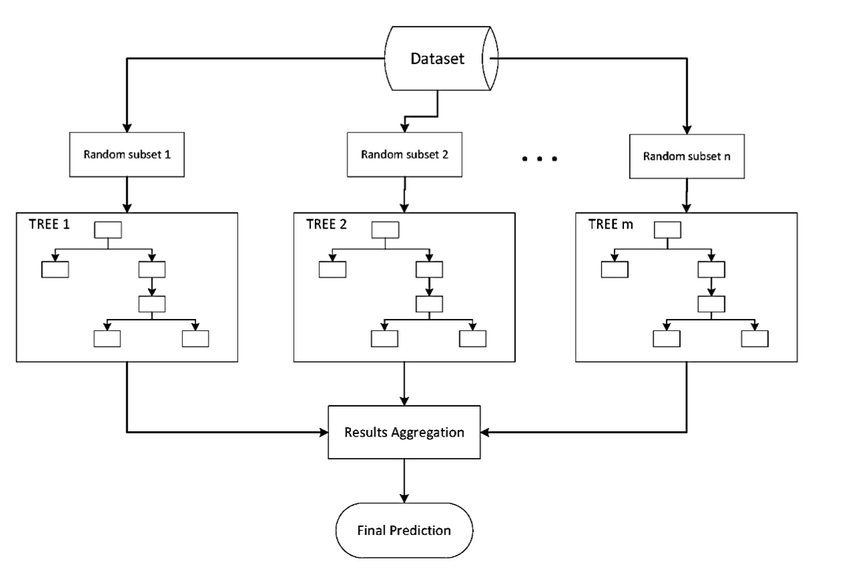
Details of decision tree over-fitting and pruning is given in the book Machine Learning by Tom Mitchell. It should be simple for beginners to understand.

1. **What is an ensemble technique in machine learning?**

Ans: Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods usually produces more accurate solutions than a single model would. This has been the case in a number of machine learning competitions, where the winning solutions used ensemble methods.

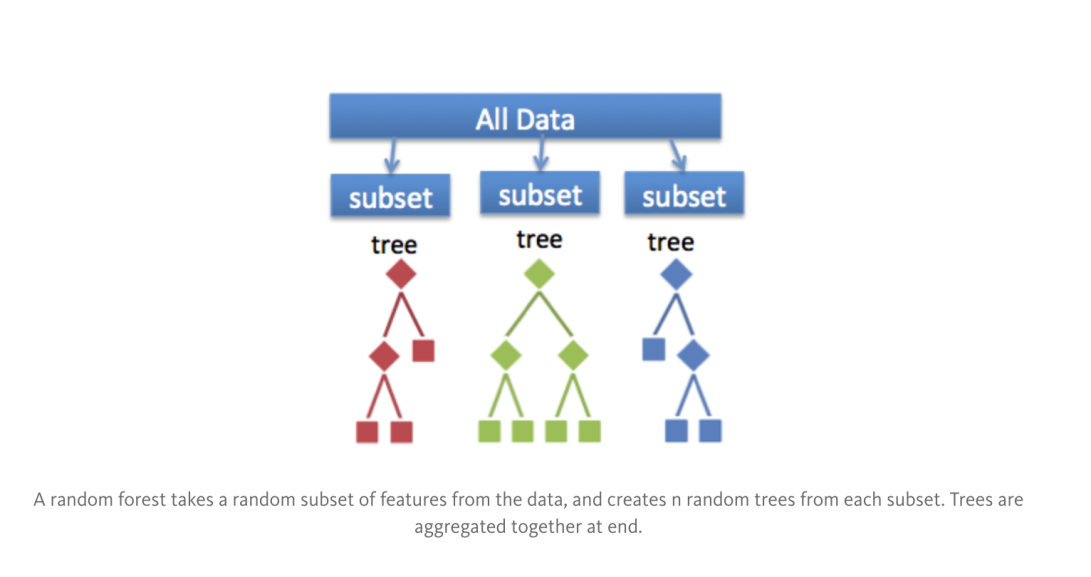
# Types of Ensemble Methods

1. *BAGG*ing, or *B*ootstrap *AGG*regating. **BAGG**ing gets its name because it combines *B*ootstrapping and *Agg*regation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor. The image below will help explain:



Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

2. Random Forest Models. Random Forest Models can be thought of as BAGGing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor. Refer to the image for a better understanding.



Similar to BAGGing, bootstrapped subsamples are pulled from a larger dataset. A decision tree is formed on each subsample. HOWEVER, the decision tree is split on different features (in this diagram the features are represented by shapes).

1. **What is the difference between Bagging and Boosting techniques?**

|  |  |
| --- | --- |
| Simplest way of combining predictions that belong to the same type. | A way of combining predictions that belong to the different types. |
| 2. | Aim to decrease variance, not bias. | Aim to decrease bias, not variance. |
| 3. | Each model receives equal weight. | Models are weighted according to their performance. |
| 4. | Each model is built independently. | New models are influenced by performance of previously built models. |
| 5. | Different training data subsets are randomly drawn with replacement from the entire training dataset. | Every new subsets contains the elements that were misclassified by previous models. |
| 6. | Bagging tries to solve over-fitting problem. | Boosting tries to reduce bias. |
| 7. | If the classifier is unstable (high variance), then apply bagging. | If the classifier is stable and simple (high bias) the apply boosting. |
| 8. | Random forest. | Gradient boosting. |

1. **What is out-of-bag error in random forests?**

Ans: Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of [random forests](https://en.wikipedia.org/wiki/Random_forest), [boosted decision trees](https://en.wikipedia.org/wiki/Gradient_boosting), and other [machine learning](https://en.wikipedia.org/wiki/Machine_learning) models utilizing [bootstrap aggregating](https://en.wikipedia.org/wiki/Bootstrap_aggregating) (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from.

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.

1. **What is K-fold cross-validation?**

Ans: Cross Validation is a very important technique that is used widely by data scientists. The problem with machine learning models is that you won’t get to know how well a model performs until you test its performance on an independent data set (the data set which was not used for training the machine learning model).

Cross Validation comes to the rescue here and helps you estimate the performance of your model. One type of cross validation is the K-Fold Cross Validation. Keep reading to learn more!

What is Cross Validation?

Cross Validation is a very useful technique for assessing the performance of machine learning models. It helps in knowing how the machine learning model would generalize to an independent data set. You want to use this technique to estimate how accurate the predictions your model will give in practice.

When you are given a machine learning problem, you will be given two type of data sets — known data (training data set) and unknown data (test data set). By using cross validation, you would be “testing” your machine learning model in the “training” phase to check for overfitting and to get an idea about how your machine learning model will generalize to independent data, which is the test data set given in the problem.

In one round of cross validation, you will have to divide your original training data set into two parts:

* Cross validation training set
* Cross validation testing set or Validation set
* You will train your machine learning model on the cross validation training set and test the model’s predictions against the validation set. You will get to know how accurate your machine learning model’s predictions are when you compare the model’s predictions on the validation set and the actual labels of the data points in the validation set.
* For reducing the variance, several rounds of cross validation are performed by using different cross validation training sets and cross validation testing sets. The results from all the rounds are averaged to estimate the accuracy of the machine learning model.

## K-Fold Cross Validation

**K-Fold Cross Validation** is a common type of cross validation that is widely used in [machine learning](https://datascience.magoosh.com/?utm_source=data-science-blog&utm_medium=blog&utm_campaign=data-science-home&utm_term=inline).

K-fold cross validation is performed as per the following steps:

1. Partition the original training data set into k equal subsets. Each subset is called a **fold**. Let the folds be named as f1, f2, …, fk .
2. For i = 1 to i = k
   1. Keep the fold fi as Validation set and keep all the remaining *k-1* folds in the Cross validation training set.
   2. Train your machine learning model using the cross validation training set and calculate the accuracy of your model by validating the predicted results against the validation set.
3. Estimate the accuracy of your machine learning model by averaging the accuracies derived in all the*k* cases of cross validation.

In the k-fold cross validation method, all the entries in the original training data set are used for both training as well as validation. Also, each entry is used for validation just once.

Generally, the value of *k* is taken to be 10, but it is not a strict rule, and *k* can take any value.

1. **What is hyper parameter tuning in machine learning and why it is done?**

Ans: In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

Model optimization is one of the toughest challenges in the implementation of machine learning solutions. Entire branches of machine learning and deep learning theory have been dedicated to the optimization of models.

Hyperparameter optimization in machine learning intends to find the hyperparameters of a given machine learning algorithm that deliver the best performance as measured on a validation set. Hyperparameters, in contrast to model parameters, are set by the machine learning engineer before training. The number of trees in a random forest is a hyperparameter while the weights in a neural network are model parameters learned during training. I like to think of hyperparameters as the model settings to be tuned so that the model can optimally solve the machine learning problem.

Some examples of model hyperparameters include:

* The learning rate for training a neural network.
* The C and 𝛾 hyperparameters for support vector machines.
* The k in k-nearest neighbors.

Hyperparameter optimization finds a combination of hyperparameters that returns an optimal model which reduces a predefined loss function and in turn increases the accuracy on given independent data.

1. **What issues can occur if we have a large learning rate in Gradient Descent?**

Ans: When the learning rate is too large, gradient descent can inadvertently increase rather than decrease the training error. […] When the learning rate is too small, training is not only slower, but may become permanently stuck with a high training error.

If your learning rate is set too low, training will progress very slowly as you are making very tiny updates to the weights in your network. However, if your learning rate is set too high, it can cause undesirable divergent behavior in your loss function. I'll visualize these cases below - if you find these visuals hard to interpret, I'd recommend reading (at least) the first section in my post on [gradient descent](https://www.jeremyjordan.me/gradient-descent/).



1. **Can we use Logistic Regression for classification of Non-Linear Data? If not, why?**

Ans: Logistic regression is known and used as a linear classifier. ... It is used to come up with a hyperplane in feature space to separate observations that belong to a class from all the other observations that do not belong to that class. The decision boundary is thus linear.

Logistic regression is considered a generalized linear model because the outcome always depends on the sum of the inputs and parameters. Or in other words, the output cannot depend on the product (or quotient, etc.) of its parameters!

1. **Differentiate between Adaboost and Gradient Boosting.**

Ans: Gradient boosting is a method that allows a computer/system to learn based on regression analysis and classification. The result is that the machine might be able to predict outcomes based on history. Some of the methods used include decision trees and use cases.

Adaboost is short for adaptive boosting, which describes an algorithm for using other methods to produce a statistically weighted outcome, which the machine can then use to boost classification of use cases.

AdaBoost stands for Adaptive Boosting. So, basically, we will see the differences between Adaptive Boosting and Gradient Boosting. The basic idea of boosting (an ensemble learning technique) is to combine several weak learners into a stronger one. The general idea of boosting algorithms is to try predictors sequentially, where each subsequent model attempts to fix the errors of its predecessor.  
  
As per documentation:  
  
In Adaboost, ‘shortcomings’ are identified by high-weight data points.  
  
In Gradient Boosting, ‘shortcomings’ (of existing weak learners) are identified by gradients.  
  
Lets elaborate the above statements:  
  
Adaboost is more about ‘voting weights’ and Gradient boosting is more about ‘adding gradient optimization’.   
  
Adaboost increases the accuracy by giving more weightage to the target which is misclassified by the model. At each iteration, Adaptive boosting algorithm changes the sample distribution by modifying the weights attached to each of the instances. It increases the weights of the wrongly predicted instances and decreases the ones of the correctly predicted instances.  
  
Gradient boosting calculates the gradient (derivative) of the Loss Function with respect to the prediction (instead of the features). Gradient boosting increases the accuracy by minimizing the Loss Function (error which is difference of actual and predicted value) and having this loss as target for the next iteration.  
  
Gradient boosting algorithm builds first weak learner and calculates the Loss Function. It then builds a second learner to predict the loss after the first step. The step continues for third learner and then for fourth learner and so on until a certain threshold is reached.

1. **What is bias-variance trade off in machine learning?**

Ans:

* Bias is the simplifying assumptions made by the model to make the target function easier to approximate.
* Variance is the amount that the estimate of the target function will change given different training data.
* Trade-off is tension between the error introduced by the bias and the variance.

**What is bias?**

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

**What is variance?**

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it’s going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can’t be more complex and less complex at the same time.

1. **Give short description each of Linear, RBF, Polynomial kernels used in SVM.**

Ans: The linear, polynomial and RBF or Gaussian kernel are simply different in case of making the hyperplane decision boundary between the classes. The kernel functions are used to map the original dataset (linear/nonlinear ) into a higher dimensional space with view to making it linear dataset.

In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. ... Training a SVM with a Linear Kernel is Faster than with any other Kernel.

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.