ASSIGNMENT\_NO\_5

QUESTION\_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?**

ANSWER

Residual sum of Squares (RSS)

The Residual sum of Squares (RSS) is defined as below and is used in the Least Square Method in order to estimate the regression coefficient .The smallest residual sum of squares is equivalent to the largest r Squared .The deviance calculation is a generalization of residual sum of squares.

Squared loss = (y−y^)2

The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model.

The smaller the residual sum of squares, the better your model fits your data; the greater the residual sum of squares, the poorer your model fits your data.

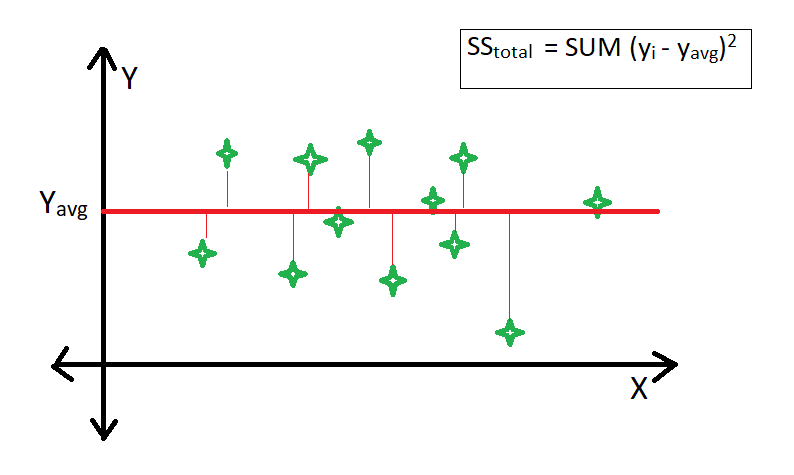
A value of zero means your model is a perfect fit.

Statistical models are used by investors and portfolio managers to track an investment's price and use that data to predict future movements.

The RSS is used by financial analysts in order to estimate the validity of their econometric models.

R-squared

R-square is a comparison of the residual sum of squares *(SSres)* with the total sum of squares*(SStot)*. The total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.



The residual sum of squares is calculated by the summation of squares of perpendicular distance between data points and the best-fitted line.

R square is calculated by using the following formula :

https://media.geeksforgeeks.org/wp-content/uploads/20190415232942/CodeCogsEqn-10.gif

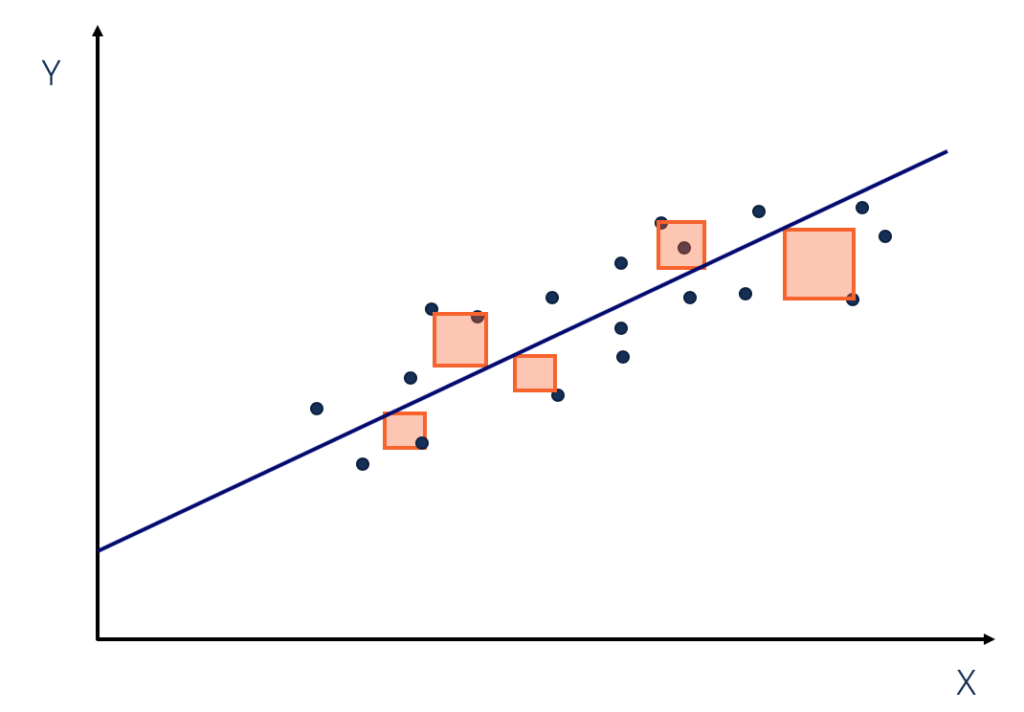
The residual sum of squares (RSS) is the absolute amount of explained variation, whereas R-squared is the absolute amount of variation as a proportion of total variation. R-squared is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

QUESTION\_2

**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.**

ANSWER

Sum of squares (SS) is a statistical tool that is used to identify the dispersion of data as well as how well the data can fit the model in regression analysis . The sum of squares got its name because it is calculated by finding the sum of the squared differences.

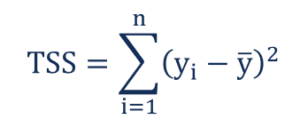


The sum of squares is one of the most important outputs in regression analysis. The general rule is that a smaller sum of squares indicates a better model, as there is less variation in the data.

### Types of 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:



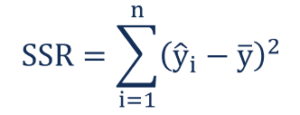
Where:

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

#### 2. 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:



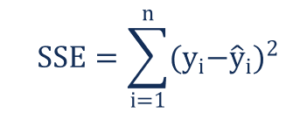
Where:

* ŷi– the value estimated by the regression line
* ȳ – the mean value of a sample

3. Residual sum of squares

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

QUESTION\_3

**What is the need of regularization in machine learning?**

ANSWER

Regularization refers to techniques that are used to calibrate machine learning models in order to**minimize the adjusted loss function and prevent overfitting or underfitting**. Using Regularization, we can fit our machine learning model appropriately on a given test set and hence reduce the errors in it.

Sometimes the [machine learning](https://www.javatpoint.com/machine-learning) model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called over fitted . This problem can be deal with the help of a regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features."

Avoiding overfitting. The model will have a low accuracy if it is overfitting. This happens because your model is trying too hard to capture the noise in your training dataset. By noise we mean the data points that don’t really represent the true properties of your data, but random chance. Learning such data points, makes your model more flexible, at the risk of overfitting.

The concept of balancing bias and variance, is helpful in understanding the phenomenon of overfitting.

*One of the ways of avoiding overfitting is using cross validation, that helps in estimating the error over test set, and in deciding what parameters work best for your model.*

QUESTION\_4

**What is Gini–impurity index?**

ANSWER

Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

Gini Index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen.

If all the elements belong to a single class, then it can be called pure. The degree of Gini Index varies between 0 and 1,

where,  
'0' denotes that all elements belong to a certain class or there exists only one class (pure), and  
'1' denotes that the elements are randomly distributed across various classes (impure).

A Gini Index of '0.5 'denotes equally distributed elements into some classes.

Consider a dataset D that contains samples from K classes. The probability of samples belonging to class i at a given node can be denoted as Pi. Then the Gini Impurity of D is defined as:

The node with uniform class distribution has the highest impurity. The minimum impurity is obtained when all records belong to the same class. Several examples are given in the following table to demonstrate the Gini Impurity computation.

An attribute with the smallest Gini Impurity is selected for splitting the node.

## Calculation of Gini Index

We will now calculate the Gini Index with the following -

* Calculating the Gini Index for past trend
* Calculating the Gini Index for open interest

Gini Index is a powerful measure of the randomness or the impurity or entropy in the values of a dataset. Gini Index aims to decrease the impurities from the root nodes (at the top of decision tree) to the leaf nodes (vertical branches down the decision tree) of a decision tree model.

QUESTION\_5

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

ANSWER

A decision tree is an algorithm for supervised learning. It uses a tree structure, in which there are two types of nodes: decision node and leaf node. A decision node splits the data into two branches by asking a Boolean question on a feature. A leaf node represents a class. The training process is about finding the “best” split at a certain feature with a certain value. And the predicting process is to reach the leaf node from root by answering the question at each decision node along the path.

Overfitting can be one problem that describes if your model no longer generalizes well.

Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it’s possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree’s they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting.

Overfitting means that a model is giving a good fit on a dataset (whatever the measure you use to assess fit), but this is not a general case (i.e. when new data comes in or on another dataset, the error will explode. Or said otherwise, the model variance is high).

In the case of trees, adding a node to a leave based on one feature should be done only if the feature really brings information at this level. The feature could be random though and this would deteriorate greatly the fit.

As a simplistic example on a classification task; if we want to sort out apples and oranges based on some features, including one of the features that is a value, 0 or 1 chosen at random. If it happens that in our dataset the values 1 correspond in 80% of the time to apples, then we would be tempted to add a node saying "if value is 1, then apple", but you can see that this is absolutely not a generality: the tree wouldn't fit at all to another dataset. Hence this node shouldn't be added, i.e. the tree should be pruned.

QUESTION\_6

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

ANSWER

Ensemble learning is a technique in machine learning which takes the help of several base models and combines their output to produce an optimized model. This type of machine learning algorithm helps in improving the overall performance of the model. Here the base model which is most commonly used is the Decision tree classifier. A decision tree basically works on several rules and provides a predictive output, where the rules are the nodes and their decisions will be their children and the leaf nodes will constitute the ultimate decision.

Different types of ensembles, but our major focus will be on the below two types:

* Bagging
* Boosting

These methods help in reducing the variance and bias in a machine learning model. Now let us try to understand what is bias and variance. Bias is an error that occurs due to incorrect assumptions in our algorithm; a high bias indicates our model is too simple/underfit. Variance is the error that is caused due to sensitivity of the model to very small fluctuations in the data set; a high variance indicates our model is highly complex/overfit. An ideal ML model should have a proper balance between bias and variance.

### Bootstrap Aggregating/Bagging

Bagging is an ensemble technique that helps in reducing variance in our model and hence avoids overfitting. Bagging is an example of the parallel learning algorithm. Bagging works based on two principles.

* **Bootstrapping:** From the original data set, different sample populations are considered with replacement.
* **Aggregating:** Averaging out the results of all the classifiers and providing single output, for this, it uses majority voting in the case of classification and averaging in the case of the regression problem. One of the famous machine learning algorithms which use the concept of bagging is a random forest.

#### Random Forest

In random forest from the random sample withdrawn from the population with replacement and a subset of features is selected from the set of all the features a decision tree is built. From these subsets of features whichever feature gives the best split is selected as the root for the decision tree. The features subset must be chosen randomly at any cost otherwise we will end up producing only correlated tress and the variance of the model will not be improved.

Now we have built our model with the samples taken from the population, the question is how do we validate the model? Since we are considering the samples with replacement hence all the samples will not be considered and some of it will not be included in any bag these are called out of bag samples. We can validate our model with this OOB (out of bag) samples. The important parameters to be considered in a random forest is the number of samples and the number of trees. Let us consider ‘m’ as the subset of features and ‘p’ is the full set of features, now as a thumb rule, it’s always ideal to choose

* m as√and a minimum node size as 1 for a classification problem.
* m as P/3 and minimum node size to be 5 for a regression problem.

The m and p should be treated as tuning parameters when we deal with a practical problem. The training can be terminated once the OOB error stabilizes. One drawback of the random forest is that when we have 100 features in our data set and only a couple of features are important then this algorithm will perform poorly.

### Boosting

Boosting is a sequential learning algorithm that helps in reducing bias in our model and variance in some cases of supervised learning. It also helps in converting weak learners into strong learners. Boosting works on the principle of placing the weak learners sequentially and it assigns a weight to each data point after every round; more weight is assigned to the misclassified data point in the previous round. This sequential weighted method of training our data set is the key difference to that of bagging

Ensemble learning techniques are widely used in improving the model accuracy; we have to decide on which technique to use based on our data set. But these techniques are not preferred in some cases where interpretability is of importance, as we lose interpretability at the cost of performance improvement. These have tremendous significance in the health care industry where a small improvement in performance is very valuable.

QUESTION\_7

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

ANSWER

Bagging and Boosting, both being the popularly used methods, have a universal similarity of being classified as ensemble methods. Here we will highlight more similarities between them, followed by the differences they have from each other. Let us first start with similarities as understanding these will make understanding the differences easier.

## Bagging and Boosting: Differences

Bagging is a method of merging the same type of predictions. Boosting is a method of merging different types of predictions.

Bagging decreases variance, not bias, and solves over-fitting issues in a model. Boosting decreases bias, not variance.

In Bagging, each model receives an equal weight. In Boosting, models are weighed based on their performance.

Models are built independently in Bagging. New models are affected by a previously built model’s performance in Boosting.

In Bagging, training data subsets are drawn randomly with a replacement for the training dataset. In Boosting, every new subset comprises the elements that were misclassified by previous models.

Bagging is usually applied where the classifier is unstable and has a high variance. Boosting is usually applied where the classifier is stable and simple and has high bias.

## Bagging and Boosting: Similarities

1. Bagging and Boosting are ensemble methods focused on getting N learners from a single learner.
2. Bagging and Boosting make random sampling and generate several training data sets
3. Bagging and Boosting arrive upon the end decision by making an average of N learners or taking the voting rank done by most of them.
4. Bagging and Boosting reduce variance and provide higher stability with minimizing errors.

QUESTION\_8

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

ANSWER

A random forest is an ensemble machine-learning model that is composed of multiple decision trees. A decision tree is a model that makes predictions by learning a series of simple decision rules based on the features of the data. A random forest combines the predictions of multiple decision trees to make more accurate and robust predictions.

[Random Forests](https://www.geeksforgeeks.org/random-forest-regression-in-python/)are often used for classification and regression tasks. In classification, the goal is to predict the class label (e.g., “cat” or “dog”) of each sample in the dataset. In regression, the goal is to predict a continuous target variable (e.g., the price of a house) based on the features of the data.

Random forests are popular because they are easy to train, can handle high-dimensional data, and are highly accurate. They also have the ability to handle missing values and can handle imbalanced datasets, where some classes are more prevalent than others.

To train a random forest, you need to specify the number of decision trees to use (the **n\_estimators** parameter) and the maximum depth of each tree (the **max\_depth** parameter). Other hyperparameters, such as the minimum number of samples required to split a node and the minimum number of samples required at a leaf node, can also be specified.

Once the random forest is trained, you can use it to make predictions on new data. To make a prediction, the random forest uses the predictions of the individual decision trees and combines them using a majority vote or an averaging technique.

### What is the difference between the OOB Score and the Validation score?

OOB (out-of-bag) score is a performance metric for a [machine learning](https://www.geeksforgeeks.org/machine-learning/) model, specifically for ensemble models such as random forests. It is calculated using the samples that are not used in the training of the model, which is called out-of-bag samples. These samples are used to provide an unbiased estimate of the model’s performance, which is known as the OOB score.

The validation score, on the other hand, is the performance of the model on a validation dataset. This dataset is different from the training dataset and is used to evaluate the model’s performance after it has been trained on the training dataset.

In summary, the OOB score is calculated using out-of-bag samples and is a measure of the model’s performance on unseen data. The validation score, on the other hand, is a measure of the model’s performance on a validation dataset, which is a set of samples that the model has not seen during training.

### **OOB (out-of-bag) Errors**

OOB (out-of-bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data. In scikit-learn, the OOB error can be obtained using the oob\_score\_ attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees. This is different from the error computed using the usual training and validation sets, which are used to tune the hyperparameters of the random forest.

The OOB error can be useful for evaluating the performance of the random forest on unseen data. It is not always a reliable estimate of the generalization error of the model, but it can provide a useful indication of how well the model is performing.

### What are some of the use cases of the OOB error?

One of the main use cases of the OOB error is to evaluate the performance of an [ensemble model](https://www.geeksforgeeks.org/ensemble-methods-in-python/), such as a random forest. Because the OOB error is calculated using out-of-bag samples, which are samples that are not used in the training of the model, it provides an unbiased estimate of the model’s performance.

Another use case of the OOB error is to tune the [hyperparameters](https://www.geeksforgeeks.org/hyperparameter-tuning/) of a model. By using the OOB error as a performance metric, the hyperparameters of the model can be adjusted to improve its performance on unseen data.

Additionally, the OOB error can be used to diagnose whether a model is overfitting or underfitting. If the OOB error is significantly higher than the validation score, it may indicate that the model is overfitting and not generalizing well to unseen data. On the other hand, if the OOB error is significantly lower than the validation score, it may indicate that the model is underfitting and not learning the underlying patterns in the data.

Overall, the OOB error is a useful tool for evaluating the performance of an ensemble model and for diagnosing issues such as overfitting and underfitting.

QUESTION \_9

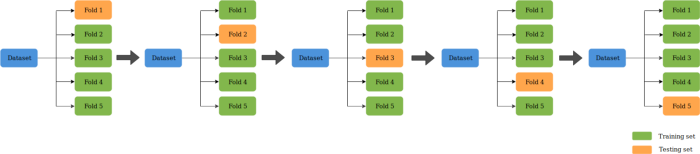
**What is K-fold cross-validation?**

ANSWER

Evaluating a Machine Learning model can be quite tricky. Usually, we split the data set into training and testing sets and use the training set to train the model and testing set to test the model. We then evaluate the model performance based on an error metric to determine the accuracy of the model. This method however, is not very reliable as the accuracy obtained for one test set can be very different to the accuracy obtained for a different test set. K-fold Cross Validation(CV) provides a solution to this problem by dividing the data into folds and ensuring that each fold is used as a testing set at some point. This article will explain in simple terms what K-Fold CV is and how to use the sklearn library to perform K-Fold CV.

# **What is K-Fold Cross Validation?**

K-Fold CV is where a given data set is split into a K number of sections/folds where each fold is used as a testing set at some point. Let’s take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds has been used as the testing set.



The following is done in this technique for training, validating, and testing the model:

1. The dataset is split into training and test dataset.
2. The training dataset is then split into K-folds.
3. Out of the K-folds, (K-1) fold is used for training
4. 1 fold is used for validation
5. The model with specific hyperparameters is trained with training data (K-1 folds) and validation data as 1 fold. The performance of the model is recorded.
6. The above steps (step 3, step 4, and step 5) is repeated until each of the k-fold got used for validation purpose. This is why it is called k-fold cross-validation.
7. Finally, the mean and standard deviation of the model performance is computed by taking all of the model scores calculated in step 5 for each of the K models.
8. Step 3 to Step 7 is repeated for different values of hyperparameters.
9. Finally, the hyperparameters which result in the most optimal mean and the standard value of model scores get selected.
10. The model is then trained using the training data set (step 2) and the model performance is computed on the test data set (step 1).

### Why use the Cross-validation technique?

The conventional technique for training and testing the model is to split the data into two different splits which are termed training and test split. For a decent size of data, the training and test split is taken as 70:30. **Here are a few challenges** due to which the cross-validation technique is used:

* Challenges with training-test split: In order to train the model for optimal performance, the hyperparameters are tweaked appropriately to achieve good model performance with the test data. However, this technique results in the risk of overfitting the test set. This is because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can “leak” into the model, and evaluation metrics no longer report on generalization performance.
* Challenges with training-validation-test split: In order to take care of the above issue, there are three splits that get created. They are training, validation, and test split. The model hyperparameters get tuned using a training and validation set. And, finally, the model generalization performance is determined using a test data split. However, this technique also has shortcomings. By partitioning the data into three sets, the number of samples which can be used for learning the model gets reduced. The results depend on a particular random choice for the pair of (train, validation) sets.

To overcome the above challenges, the cross-validation technique is used. As described earlier in this section, two different splits such as training and test split get created. However, cross-validation is applied to the training data by creating K-folds of training data in which (K-1) fold is used for training and the remaining fold is used for testing. This process is repeated for K times and the model performance is calculated for a particular set of hyperparameters by taking the mean and standard deviation of all the K models created. The hyperparameters giving the most optimal model are calculated. Finally, the model has trained again on the training data set using the most optimal hyper parameter and the generalization performance is computed by calculating model performance on the test dataset. The diagram given below represents the same.

QUESTION \_10

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

ANSWER

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.   
However, there is another kind of parameter, known as *Hyperparameters*, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

The hyper-parameter tuning process is a tightrope walk to achieve a balance between**underfitting and overfitting**. Underfitting is when the machine learning model is unable to reduce the error for either the test or training set. An underfitting model is not powerful enough to fit the underlying complexities of the data distributions.

Some examples of model hyperparameters include:

1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
2. The learning rate for training a neural network.
3. The C and sigma hyperparameters for support vector machines.
4. The k in k-nearest neighbors.

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:

* [GridSearch CV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)
* [Randomized Search CV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html)

GridSearchCV   
In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

For example, if we want to set two hyperparameters C and Alpha of the Logistic Regression Classifier model, with different sets of values. The grid search technique will construct many versions of the model with all possible combinations of hyperparameters and will return the best one.

***Drawback***: GridSearchCV will go through all the intermediate combinations of hyperparameters which makes grid search computationally very expensive.

**RandomizedSearchCV**   
RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in a random fashion to find the best set of hyperparameters. This approach reduces unnecessary computation.

QUESTION \_11

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

ANSWER

Gradient Descent is an iterative process that finds the minima of a function. Gradient descent is an optimization algorithm mainly used to find the minimum of a function. In machine learning, gradient descent is used to update parameters in a model. Parameters can vary according to the algorithms, such as coefficients in [Linear Regression](https://www.knowledgehut.com/blog/data-science/linear-regression-for-machine-learning) and weights in Neural Networks.

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function f(x) parameterized by x. Similarly, in machine learning, optimization is the task of minimizing the cost function parameterized by the model's parameters. The main objective of gradient descent is to minimize the convex function using iteration of parameter updates. Once these machine learning models are optimized, these models can be used as powerful tools for Artificial Intelligence and various computer science applications.

The learning rate can seen as step size, η(bar in all). As such, gradient descent is taking successive steps in the direction of the minimum. If the step size η� is too large, it can (plausibly) "jump over" the minima we are trying to reach, ie. we overshoot. This can lead to osculations around the minimum or in some cases to outright divergence. It is important to note that the step gradient descent takes is a function of step size η� as well as the gradient values g�. If we are in a local minimum with zero gradient the algorithm will not update the parameters p� because the gradient is zero, similarly if p� is in a "steep slope", even a small η� will lead to a large update in p�'s values.

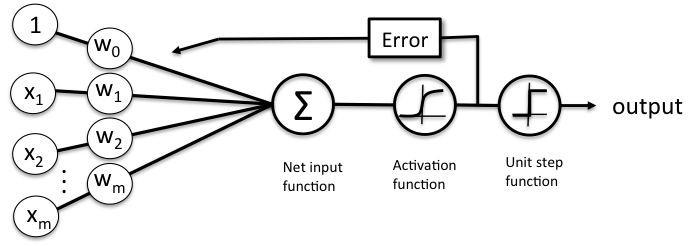
QUESTION \_12

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

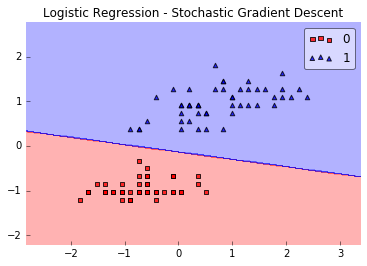
ANSWER

The short answer is: 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!

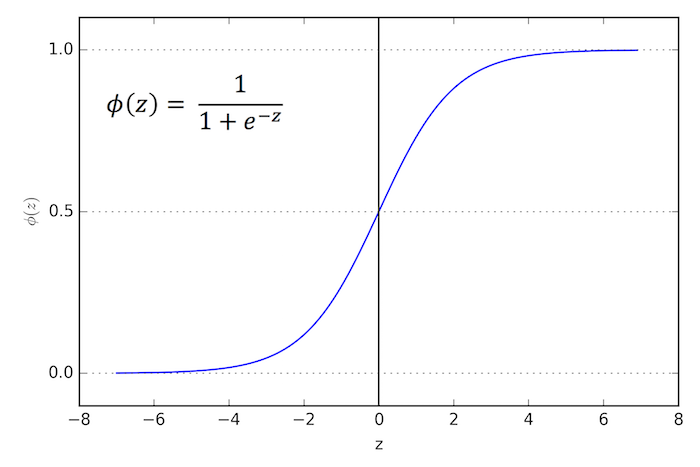
So, why is that? Let’s recapitulate the basics of logistic regression first, which hopefully makes things more clear. Logistic regression is an algorithm that learns a model for binary classification. A nice side-effect is that it gives us the *probability* that a sample belongs to class 1 (or vice versa: class 0). Our objective function is to minimize the so-called logistic function Φ (a certain kind of sigmoid function); it looks like this:



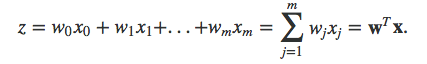
Now, if *φ(z)* is larger than *0.5* (alternatively: if *z* is larger than *0*), we classify an input as class 1 (and class 0, otherwise). Although logistic regression produces a linear decision surface (see the classification example in the figure below) this logistic (activation) function doesn’t look very linear at all, right!?doesn’t look very linear at all, right!?



So, let’s dig a bit deeper and take a look at the equation we use to compute *z* – the net input function!



The net input function is simply the dot product of our input features and the respective model coefficients w:



Here, x0 refers to the weight of the bias unit which is always equal to 1 (a detail we don’t have to worry about here). I know, mathematical equations can be a bit “abstract” at times, so let’s look at a concrete example.

Let’s assume we have a sample training point x consisting of 4 features (e.g., *sepal length*, *sepal width*, *petal length*, and *petal width* in the [*Iris dataset*](https://archive.ics.uci.edu/ml/datasets/Iris)):

x = [1, 2, 3, 4]

Now, let’s assume our weight vector looks like this:

w = [0.5, 0.5, 0.5, 0.5]

Let’s compute *z* now!

z = wTx = 1*0.5 + 2*0.5 + 3*0.5 + 4*0.5 = 5

Not that it is important, but we have a 99.3% chance that this sample belongs to class 1:

*Φ(z=148.41) = 1 / (1 + e-5) = 0.993*

The key is that our model is *additive* our outcome *z* depends on the additivity of the weight parameter values, e.g., :

*z = w1x1 + w2x2*

There’s no interaction between the feature values, nothing like w1x1 \* w2x2

Or so, which would make our model non-linear!

QUESTION \_13

**Differentiate between Adaboost and Gradient Boosting**

ANSWER

### **AdaBoost**

AdaBoost or Adaptive Boosting is the first [Boosting ensemble model](https://analyticsindiamag.com/ensemble-modeling-explained-through-music/). The method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.

In practice, this boosting technique is used with [simple classification trees](https://arxiv.org/pdf/1403.1452.pdf) or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.

### **Gradient Boosting**

Gradient Boost is a robust [machine learning algorithm](https://analyticsindiamag.com/pseudo-labelling-a-guide-to-semi-supervised-learning/) made up of Gradient descent and Boosting. The word ‘gradient’ implies that you can have two or more derivatives of the same function. Gradient Boosting has three main components: additive model, loss function and a weak learner.

The technique yields a direct interpretation of boosting methods from the perspective of numerical optimisation in a function space and generalises them by allowing optimisation of an arbitrary loss function.

### **The Comparison**

#### **Loss Function:**

The technique of Boosting uses various loss functions. In case of Adaptive Boosting or AdaBoost, it minimises the exponential loss function that can make the algorithm sensitive to the outliers. With Gradient Boosting, any differentiable loss function can be utilised. Gradient Boosting algorithm is more robust to outliers than AdaBoost.

#### **Flexibility**

AdaBoost is the first designed boosting algorithm with a particular loss function. On the other hand, Gradient Boosting is a generic algorithm that assists in searching the approximate solutions to the additive modelling problem. This makes Gradient Boosting more flexible than AdaBoost.

#### **Benefits**

AdaBoost minimises loss function related to any classification error and is best used with weak learners. The method was mainly designed for binary classification problems and can be utilised to boost the performance of decision trees. Gradient Boosting is used to solve the differentiable loss function problem. The technique can be used for both classification and regression problems.

#### **Shortcomings**

In the case of Gradient Boosting, the shortcomings of the existing weak learners can be identified by gradients and with AdaBoost, it can be identified by high-weight data points.

### **Wrapping Up**

Though there are several differences between the two boosting methods, both the algorithms follow the [same path](https://arxiv.org/pdf/1403.1452.pdf) and share similar historic roots. Both the algorithms work for boosting the performance of a simple base-learner by iteratively shifting the focus towards problematic observations that are challenging to predict.

In the case of AdaBoost, the shifting is done by up-weighting observations that were misclassified before, while Gradient Boosting identifies the difficult observations by large residuals computed in the previous iterations.

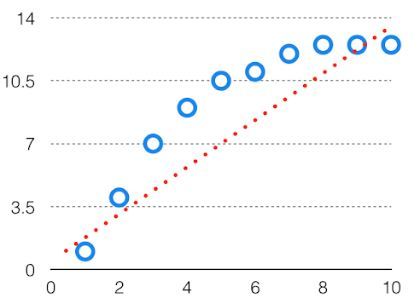
QUESTION \_14

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

ANSWER

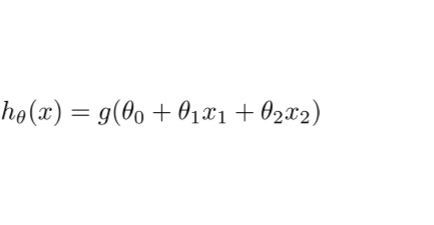
It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm. There is a tradeoff between a model’s ability to minimize bias and variance which is referred to as the best solution for selecting a value of Regularization constant. Proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

Bias  
The bias is known as the difference between the prediction of the values by the ML model and the correct value. Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.  
By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as Underfitting of Data. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below ,for an example of such a situation.



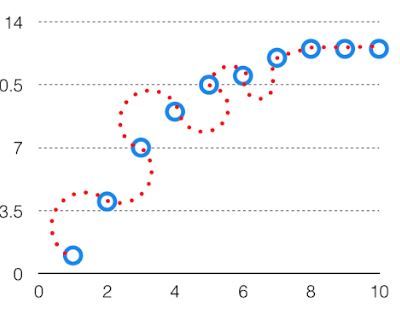
*High Bias*

In such a problem, a hypothesis looks like follows.

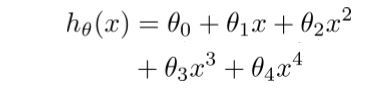
  
Variance  
The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.

When a model is high on variance, it is then said to as Overfitting of Data. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high.  
While training a data model variance should be kept low.

The high variance data looks like follows.

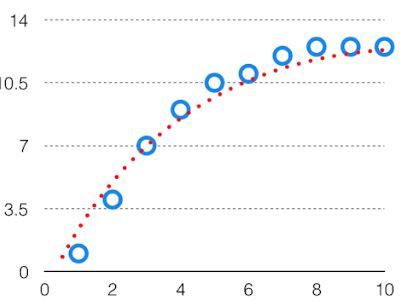


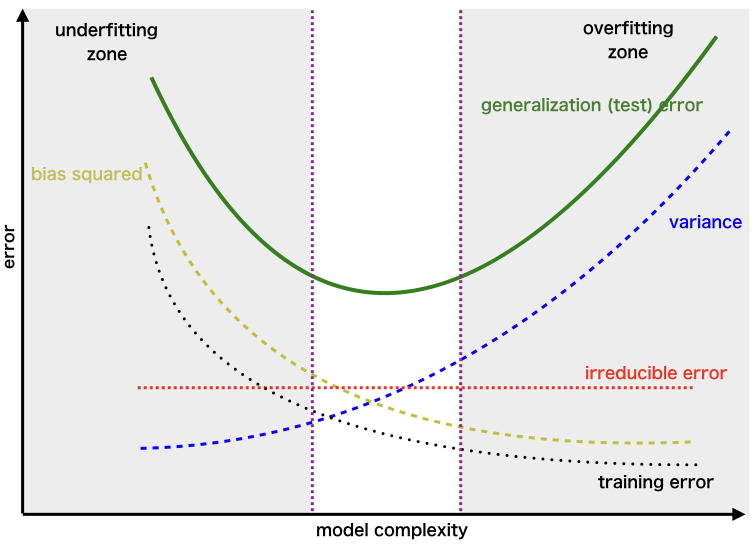
*High Variance*

In such a problem, a hypothesis looks like follows.  
  
Bias Variance Tradeoff

If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as Trade-off or Bias Variance Trade-off.

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. For the graph, the perfect tradeoff will be like.

  
The best fit will be given by hypothesis on the tradeoff point.

The error to complexity graph to show trade-off is given as –  
   
This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

QUESTION \_15

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

ANSWER

**Linear Kernel**

**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. One of the examples where there are a lot of features, is **Text Classification**, as each alphabet is a new feature. So we mostly use Linear Kernel in Text Classification.

It is the most basic type of kernel, usually one dimensional in nature. It proves to be the best function when there are lots of features. The linear kernel is mostly preferred for [**text-classification problems**](https://dataaspirant.com/nlp-text-preprocessing-techniques-implementation-python/) as most of these kinds of classification problems can be linearly separated.

Linear kernel functions are **faster** than other functions.

#### Linear Kernel Formula

**F(x, xj) = sum( x.xj)**

Here, **x, xj** represents the data you’re trying to classify.

### Polynomial Kernel

It is a more generalized representation of the linear kernel. It**is not** as preferred as other kernel functions as it is **less efficient** and accurate.

#### Polynomial Kernel Formula

**F(x, xj) = (x.xj+1)^d**

Here ‘.’ shows the **dot product** of both the values, and **d** denotes the degree.

F(x, xj) representing the **decision boundary** to separate the given classes.

### Gaussian Radial Basis Function (RBF)

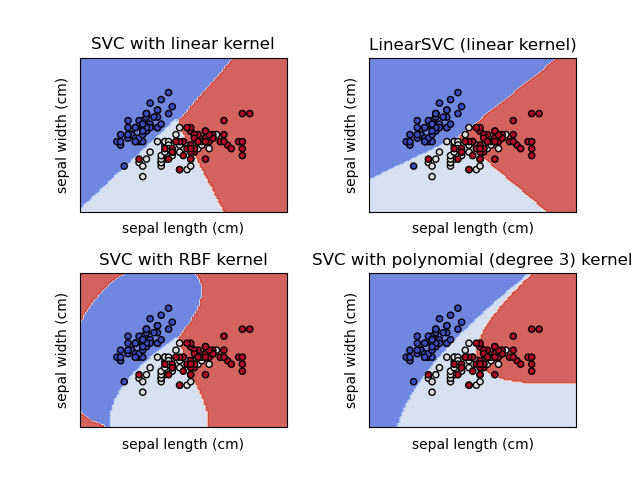
 It is one of the most preferred and used kernel functions in svm. It is usually chosen for non-linear data. It helps to make proper separation when there is no prior knowledge of data.

#### Gaussian Radial Basis Formula

**F(x, xj) = exp(-gamma \* ||x - xj||^2)**

The value of gamma varies from **0 to 1**. You have to manually provide the value of gamma in the code. The most preferred value for **gamma is 0.1**.

[SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC), [NuSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC) and [LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC) are classes capable of performing binary and multi-class classification on a dataset



STATISTICS WORKSHEET-5

Q1 to Q10 are MCQs with only one correct answer. Choose the correct option.

1. Using a goodness of fit ,we can assess whether a set of obtained frequencies differ from a set of frequencies.

a) Mean

b) Actual

c) Predicted s

**d) Expected**

2. Chisquare is used to analyze

a) Score

b) Rank

**c) Frequencies**

d) All of these

3. What is the mean of a Chi Square distribution with 6 degrees of freedom?

a) 4

b) 12

**c) 6**

d) 8

4. Which of these distributions is used for a goodness of fit testing?

a) Normal distribution

**b)Chi-square distribution**

c) Gamma distribution

d) Poission distribution

5. Which of the following distributions is Continuous

a) Binomial Distribution

b) Hypergeometric Distribution

**c) F Distribution**

d) Poisson Distribution

6. A statement made about a population for testing purpose is called?

a) Statistic

**b) Hypothesis**

c) Level of Significance

d) Test Statistic

7. If the assumed hypothesis is tested for rejection considering it to be true is called?

**a) Null Hypothesis**

b) Statistical Hypothesis

c) Simple Hypothesis

d) Composite Hypothesis

8. If the Critical region is evenly distributed then the test is referred as?

**a) Two tailed**

b) One tailed

c) Three tailed

d) Zero tailed

9. Alternative Hypothesis is also called as?

a) Composite hypothesis

**b) Research Hypothesis**

c) Simple Hypothesis

d) Null Hypothesis WORKSHEET

10. In a Binomial Distribution, if ‘n’ is the number of trials and ‘p’ is the probability of success, then the mean value is given by

**a) Np**

b) n