FLIP ROBO

ASSIGNMENT - 5

MACHINE-LEARNING

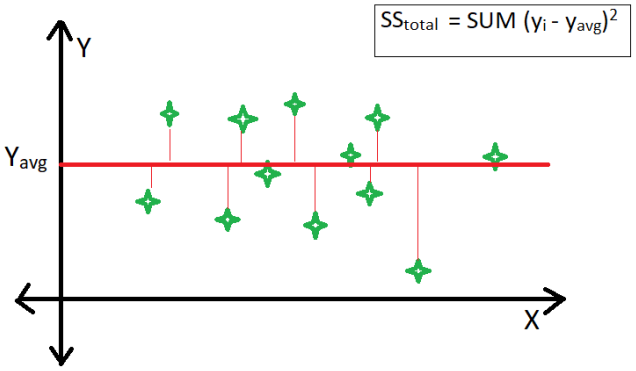
Q1 to Q15 are subjective answer type questions, Answer them briefly.

Q1. 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?

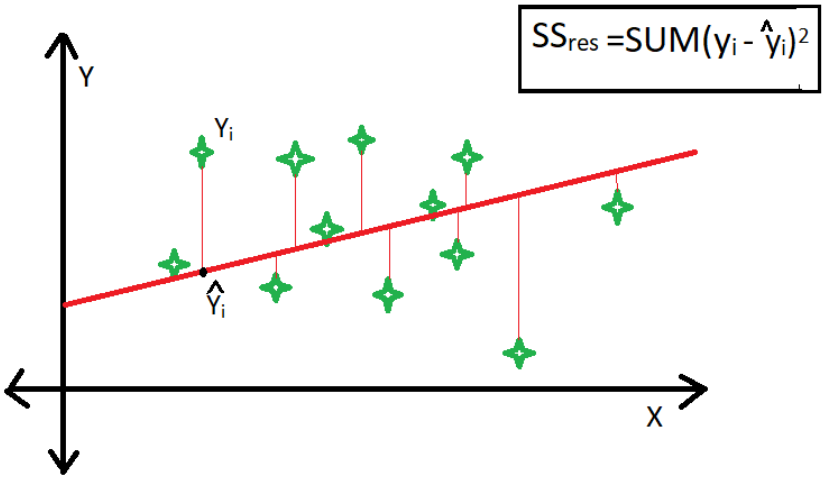
Ans.: R-squared represents the proportion of the variance in your data which is explained by your model; the closer to one, the better the fit.

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.

R-square is a comparison of the residual sum of squares (RSS or SSres) with the total sum of squares(TSS or 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 :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where, SSres is the residual sum of squares and SStot is the total sum of squares.

The goodness of fit of regression models can be analyzed on the basis of the R-square method. The more the value of r-square near 1, the better is the model.

The **Residual sum of squares (RSS)** is the absolute amount of explained variation, 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.

The RSS, also known as the sum of squared residuals, essentially determines how well a regression model explains or represents the data in the model.

The actual number you get depends largely on the scale of your response variable. Taken alone, the RSS isn't so informative.

Hence, R-squared is a better measure of goodness of fit model in regression.

Q2. 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.: ***TSS (Total Sum of Squares):*** The first formula we'll look at is the Sum of Squares Total (denoted as SST or TSS). TSS finds the squared difference between each variable and the mean.

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where, yi = The *i*th term in the set

ȳ = the mean of all items in the set

What this means is for each variable, you take the value and subtract the mean, then square the result. This gives you the distance from the linear line drawn to each particular variable. You could also describe TSS as the dispersion of observed variables around the mean, or the variance. So, the goal of TSS is to measure the total variability of the dataset.

ESS (Explained Sum of Squares) : The next formula we'll talk about is Sum of Squares Regression (denoted as SSR), also known as Explained Sum of Squares (denoted as ESS). SSR is used to describe the difference between the predicted value and the mean of the dependent variable.

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

Where, ŷi — the value estimated by the regression line

ȳ — the mean value of a sample

To start, we will again need the mean. The estimated value is the one that lies on the regression line. That means instead of the actual value of each variable, take the value of where that variable would be on the line of regression. This will tell us how well the line fits the data. If the SSR matches the TSS, then that line would be a perfect fit.

RSS (Residual Sum of Squares): The final formula to discuss is the Sum of Squares Error (denoted SSE), also known as Residual Sum of Squares (RSS). SSE finds the difference between the observed, or actual value of the variable, and the estimated value, which is what it should be according to the line of regression.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where, yi — the observed value

ŷi — the value estimated by the regression line

In the case of a perfect fit, the error would be 0, meaning the estimated value is the same as the actual value. Any value above 0 shows the error, or to what degree the line is inaccurate according to the values. The lower the value, the better line of regression fits the data. A high residual sum would demonstrate that the model poorly represents the data.

Now that we have all three explained, we can represent their relationship:

$$TSS = SSR + SSE$$

Q3. What is the need of regularization in machine learning?

Ans.: To train our machine learning model, we give it some data to learn from. The process of plotting a series of data points and drawing the best fit line to understand the relationship between the variables is called Data Fitting. Our model is the best fit when it can find all necessary patterns in our data and avoid the random data points and unnecessary patterns called **Noise**.

Overfitting: If we allow our machine learning model to look at the data too many times, it will find a lot of patterns in our data, including the ones which are unnecessary. It will learn really well on the test dataset and fit very well to it. It will learn important patterns, but it will also learn from the noise in our data and will not be able to predict on other datasets.

A scenario where the machine learning model tries to learn from the details along with the noise in the data and tries to fit each data point on the curve is called **Overfitting**.

In the figure depicted below, we can see that the model is fit for every point in our data. If given new data, the model curves may not correspond to the patterns in the new data, and the model cannot predict very well in it.

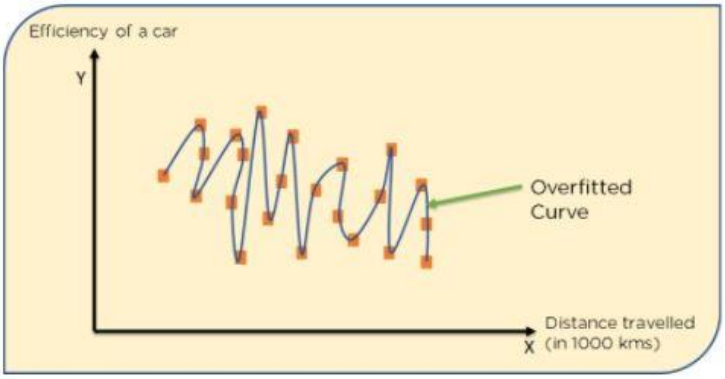


Figure 1: Overfitted Model

Underfitting: Conversely, in a scenario where the model has not been allowed to look at our data a sufficient number of times, the model won't be able to find patterns in our test dataset. It will not fit properly to our test dataset and fail to perform on new data too.

A scenario where a machine learning model can neither learn the relationship between variables in the testing data nor predict or classify a new data point is called **Underfitting.**

The below diagram shows an under-fitted model. We can see that it has not fit properly to the data given to it. It has not found patterns in the data and has ignored a large part of the dataset. It cannot perform on both known and unknown data.

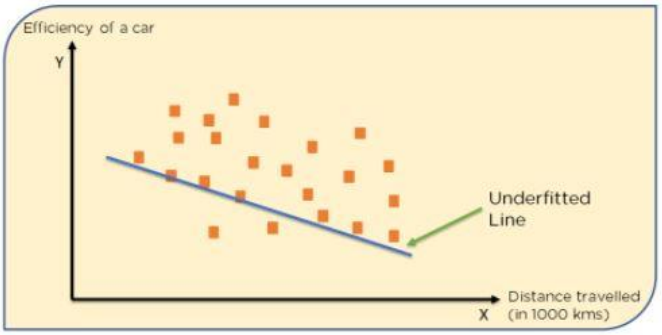
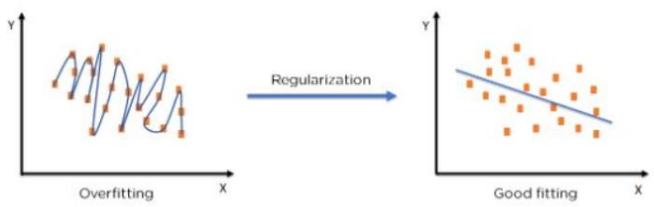


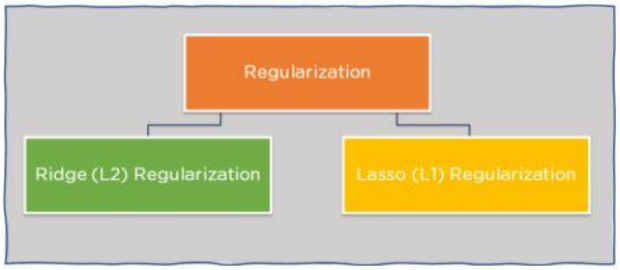
Figure 2: Underfitted Model

Regularization is the most used technique to penalize complex models in machine learning, it is deployed for reducing overfitting (or, contracting generalization errors) by putting network weights small. Also, it enhances the performance of models for new inputs.



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.

There are two main types of **Regularization Techniques**: Ridge Regularization and Lasso Regularization.



L1 regularization gives output in binary weights from 0 to 1 for the model's features and is adopted for decreasing the number of features in a huge dimensional dataset.

L2 regularization disperse the error terms in all the weights that leads to more accurate customized final models.

Q4. What is Gini–impurity index?

Ans.: Gini Index, also known as Gini impurity, calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly. If all the elements are linked with a single class, then it can be called pure. Gini impurity is a function that determines how well a decision tree was split. Basically, it helps us to determine which splitter is best so that we can build a pure decision tree. the Gini index varies between values 0 and 1, where 0 expresses the purity of classification, i.e. All the elements belong to a specified class or only one class exists there. And 1 indicates the random distribution of elements across various classes. The value of 0.5 of the Gini Index shows an equal distribution of elements over some classes.

The Gini Index is determined by deducting the sum of squared of probabilities of each class from one, mathematically, Gini Index can be expressed as:

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

Where, Pi denotes the probability of an element being classified for a distinct class.

Q5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans.: Yes, unregularized decision-trees are prone to overfitting, especially when a tree is particularly deep. This is due to the amount of specificity we look at leading to smaller sample of events that meet the previous assumptions. This small sample could lead to unsound conclusions.

An example of this could be predicting if the Boston Celtics will beat the Miami Heat in tonight's basketball game. The **first level** of the tree could ask if the Celtics are playing home or away. The **second level** might ask if the Celtics have a higher win percentage than their opponent, in this case the Heat. The **third level**

asks if the Celtic's leading scorer is playing? The **fourth level** asks if the Celtic's second leading scorer is playing. The **fifth level** asks if the Celtics are traveling back to the east coast from 3 or more consecutive road games on the west coast.

While all of these questions may be relevant, there may only be two previous games where the conditions of tonight's game were met. Using only two games as the basis for our classification would not be adequate for an informed decision. One way to combat this issue is by setting a max depth. This will limit our risk of overfitting; but as always, this will be at the expense of error due to bias. Thus, if we set a max depth of three, we would only ask if the game is home or away, do the Celtics have a higher winning percentage than their opponent, and is their leading scorer playing. This is a simpler model with less variance sample to sample but ultimately will not be a strong predictive model.

Ideally, we would like to minimize both error due to bias and error due to variance.

Q6. What is an ensemble technique in machine learning?

Ans.: Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. It is a general meta-approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

Although there are a seemingly unlimited number of ensembles that you can develop for the predictive modeling problem, there are three methods that dominate the field of ensemble learning. So much so, that rather than algorithms per se, each is a field of study that has spawned many more specialized methods.

The three main classes of ensemble learning methods are **bagging**, **stacking**, and **boosting**, and it is important to consider them on the predictive modeling project.

Q7. What is the difference between Bagging and Boosting techniques?

Ans.: Bagging and Boosting are two types of Ensemble Learning. These two decrease the variance of a single estimate as they combine several estimates from different models. So, the result may be a model with higher stability. Let's understand these two terms in a glimpse.

Bagging: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.

Boosting: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

Differences Between Bagging and Boosting

S.NO	Bagging	Boosting
1.	The 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 the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the 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.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Q8. What is out-of-bag error in random forests?

Ans.: The OOB score is calculated using out-of-bag samples and is a measure of the model's performance on unseen data. OOB (out-of-bag) score is a performance metric for a 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.

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.

Q9. What is K-fold cross-validation?

Ans.: Cross validation is an evaluation method used in machine learning to find out how well your machine learning model can predict the outcome of unseen data. It is a method that is easy to comprehend, works well for a limited data sample and also offers an evaluation that is less biased, making it a popular choice.

The data sample is split into 'k' number of smaller samples, hence the name: K-fold Cross Validation. You may also hear terms like four fold cross validation, or ten fold cross validation, which essentially means that the sample data is being split into four or ten smaller samples respectively.

Q10. What is hyper parameter tuning in machine learning and why it is done?

Ans.: 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.

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:

1. GridSearchCV
2. RandomizedSearchCV

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.

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.

Q11. What issues can occur if we have a large learning rate in Gradient Descent?

Ans.: Learning rate (λ) is one such hyper-parameter that defines the adjustment in the weights of our network with respect to the loss gradient descent. It determines how fast or slow we will move towards the optimal weights

The Gradient Descent Algorithm estimates the weights of the model in many iterations by minimizing a cost function at every step.

Here is the algorithm:

Repeat until convergence

{ Wj = Wj - λ θF(Wj) / θWj }

Where:

Wj is the weight

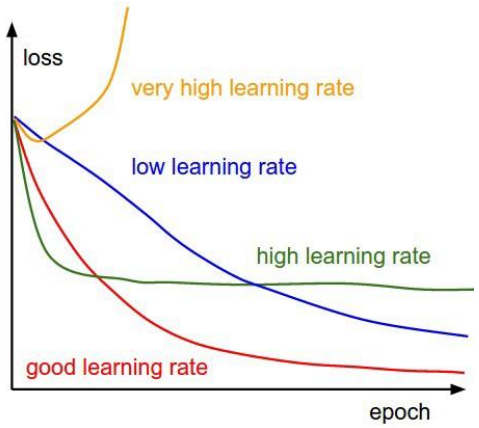
θ is the theta

F(Wj) is the cost function respectively.

In order for Gradient Descent to work, we must set the learning rate to an appropriate value. This parameter determines how fast or slow we will move towards the optimal weights. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values. So using a good learning rate is crucial.

The learning rate is the most important hyper-parameter for tuning neural networks. A good learning rate could be the difference between a model that doesn't learn anything and a model that presents state-of-the-art results.

The below diagram demonstrates the different scenarios one can fall into when configuring the learning rate.



The obvious way to find a desirable or optimal learning rate is through trial and error. To do this efficiently, there are a few ways that we should adhere to.

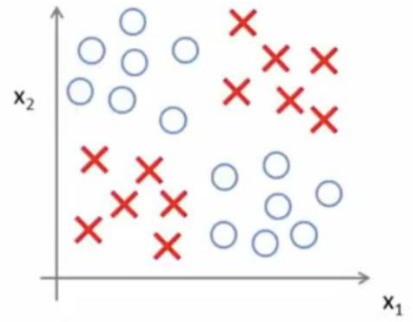
Q12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Ans.: We cannot use Logistic Regression for classification of Non-Linear Data because:

→ Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Logistic regression is considered a generalized linear model because the outcome always depends on the sum of the inputs and parameters.

→ Non-Linear Classification refers to categorizing those instances that are not linearly separable.

→ Some of the classifiers that use non-linear functions to separate classes are Quadratic Discriminant Classifier, Multi-Layer Perceptron (MLP), Decision Trees, Random Forest, and K-Nearest Neighbours (KNN).



→ In the figure above, we have two classes, namely 'O' and 'X.' To differentiate between the two classes, it is impossible to draw an arbitrary straight line to ensure that both the classes are on distinct sides.

→ We notice that even if we draw a straight line, there would be points of the first-class present between the data points of the second class.

→ In such cases, piece-wise linear or non-linear classification boundaries are required to distinguish the two classes.

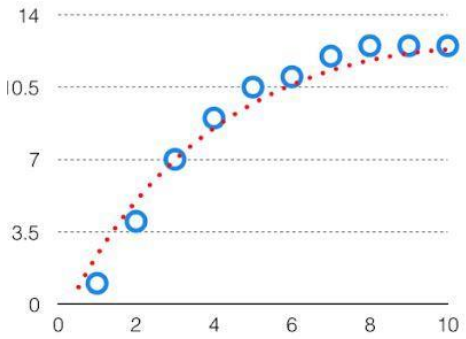
Q13. Differentiate between Adaboost and Gradient Boosting.
Ans.: Comparison between AdaBoost and Gradient Boost

S.No	Adaboost	Gradient Boost
1	An additive model where shortcomings of previous models are identified by high-weight data points.	An additive model where shortcomings of previous models are identified by the gradient.
2	The trees are usually grown as decision stumps.	The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.
3	Each classifier has different weights assigned to the final prediction based on its performance.	All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy.
4	It gives weights to both classifiers and observations thus capturing maximum variance within data.	It builds trees on previous classifier's residuals thus capturing variance in data.

Q14. What is bias-variance trade off in machine learning?

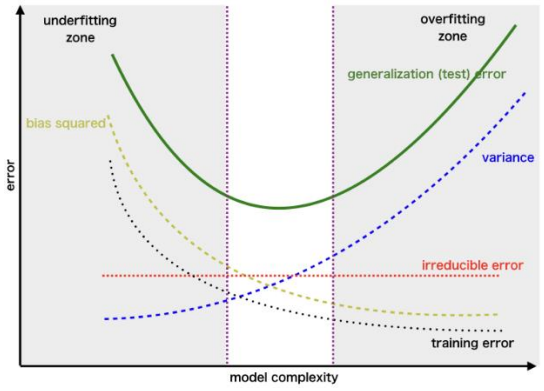
Ans.: If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will 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.

Q15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans.: *SVM Kernel Functions:*

The kernel functions play a very important role in SVM for helping to solve problems.

Their job is to take data as input and they transform it in any required form.

They are significant in SVM as they help in determining various important things.

They provide shortcuts to avoid complex calculations.

The amazing thing about kernel is that we can go to higher dimensions and perform smooth calculations with the help of it.

We can go up to an infinite number of dimensions using kernels.

Sometimes, we cannot have a hyperplane for certain problems. This problem arises when we go up to higher dimensions and try to form a hyperplane.

A kernel helps to form the hyperplane in the higher dimension without raising the complexity.

1. ***Polynomial Kernel Function:*** The polynomial kernel is a general representation of kernels with a degree of more than one. It's useful for image processing.

There are two types of this:

- *Homogenous Polynomial Kernel Function*

K(xi,xj) = (xi.xj)d, where '.' is the dot product of both the numbers and d is the degree of the polynomial.

- *Inhomogeneous Polynomial Kernel Function*

K(xi,xj) = (xi.xj + c)d where c is a constant.

2. ***Gaussian RBF Kernel Function:*** RBF is the radial basis function. This is used when there is no prior knowledge about the data.

It's represented as:

K(xi,xj) = exp(-γ||xi – xj||)2

3. ***Linear Kernel Function:*** This kernel is one-dimensional and is the most basic form of kernel in SVM. The equation is:

K(xi,xj) = xi.xj + c