Machine learning

(Solution)

Q.1 - Both R-squared and Residual Sum of Squares (RSS) are measures of goodness of fit in regression analysis, but they capture different aspects of the model's performance.

R-squared (also known as the coefficient of determination) measures the proportion of variation in the dependent variable that is explained by the independent variables in the model. In other words, it indicates how well the model fits the data, with values ranging from 0 to 1. Higher R-squared values indicate a better fit, as they mean that a larger proportion of the variation in the dependent variable is explained by the independent variables in the model.

On the other hand, RSS measures the total sum of squared differences between the actual values of the dependent variable and the predicted values by the model. It represents the amount of unexplained variation in the data, and lower RSS values indicate a better fit, as they mean that the model is able to explain more of the variation in the data.

Q. 2 – Total sum of square (TSS) is the sum of squared differences between the observed *dependent variables* and the overall mean. Think of it as the dispersion of the observed variables around the Mean—similar to the variance in descriptive statistics. But SST measures the total variability of a dataset, commonly used in regression analysis and ANOVA.

The sum of square due to regression(SSR) or explained sum of square(ESS**)** is the sum of the differences between the *predicted value* and the mean of the *dependent variable*. In other words, it describes how well our line fits the data.

The sum of squares error (SSE) or residual sum of squares (RSS)**,** Where residual means remaining or unexplained**)** is the difference between the *observed* and *predicted* values.

Mathematically, **SST** = **SSR** + **SSE**.

The rationale is the following:

The total variability of the dataset is equal to the variability explained by the regression line plus the unexplained variability, known as error.

Q.3 -Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

Sometimes the 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 overfitted. 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.

Q.4 -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.

For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labeled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – you could calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the impurity of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node. This process would continue for each subsequent node using the remaining features.

Q.5 - 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 to nights 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 strongpredictive model.

Q.6 -The ensemble methods in machine leaíning combine the insights obtained fíom multiple leaíning models to facilitate accuíate and impíoved decisions. These methods follow the same píinciple as the example of buying an aií-conditioneí cited above.

In leaíning models, noise, vaíiance, and bias aíe the majoí souíces of eííoí. The ensemble methods in machine leaíning help minimize these eííoí-causing factoís, theíeby ensuíing the accuíacy and stability of machine leaíning (ML) algoíithms.

Similaíly, ensemble methods in machine leaíning employ a set of models and take advantage of the blended output, which, compaíed to a solitaíy model, will most ceítainly be a supeíioí option when it comes to píediction accuíacy.

Q.7 -Bagging and boosting are different ensemble techniques that use multiple models to reduce error and optimize the model. The bagging technique combines multiple models trained on different subsets of data, whereas boosting trains the model sequentially, focusing on the error made by the previous model. In this article, we will discuss the difference between bagging and boosting.

	Bagging	Boosting
Basic Concept	Combines multiple models trained on different subsets of data.	Train models sequentially, focusing on the error made by the previous model.
Objective	To reduce variance by averaging out individual model error.	Reduces both bias and variance by correcting misclassifications of the previous model.
Data Sampling	Use Bootstrap to create subsets of the data.	Re-weights the data based on the error from the previous model, making the next models focus on misclassified instances.
Model Weight	Each model serves equal weight in the final decision.	Models are weighted based on accuracy, i.e., better-accuracy models will have a higher weight.
Error Handling	Each model has an equal error rate.	It gives more weight to instances with higher error, making subsequent model focus on them.
Overfitting	Less prone to overfitting due to average mechanism.	Generally not prone to overfitting, but it can be if the number of the model or the iteration is high.
Performance	Improves accuracy by reducing variance.	Achieves higher accuracy by reducing both bias and variance.
Common Algorithms	Random Forest	AdaBoost, XGBoost, Gradient Boosting Mechanism
Use Cases	Best for high variance, and low bias models.	Effective when the model needs to be adaptive to errors, suitable for both bias and variance errors.

Difference Between Bagging and Boosting: Bagging vs Boosting

Q.8.- Generally, in machine learning and data science, it is crucial to create a trustful system that will work well with the new, unseen data. Overall, there are a lot of different approaches and methods to achieve this generalization. Out-of-bag error is one of these methods for validating the machine learning model.

This approach utilizes the usage of bootstrapping in the random forest. Since the bootstrapping samples the data with the possibility of selecting one sample multiple times, it is very likely that we won't select all the samples from the original data set. Therefore, one smart decision would be to exploit somehow these unselected samples, called out-of-bag samples.

Correspondingly, the error achieved on these samples is called out-of-bag error. What we can do is to use out-of-bag samples for each decision tree to measure its performance. This strategy provides reliable results in comparison to other validation techniques such as train-test split or cross-validation.

Q.9 -Let's say that you have trained a machine learning model. Now, you need to find out how well this model performs. Is it accurate enough to be used? How does it compare to another model? There are several evaluation methods to determine this. One such method is called K-fold cross validation.

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.

Q.10 -Hyperparameters are the knobs or settings that can be tuned before running a training job to control the behavior of an ML algorithm. They can have a big impact on model training as it relates to training time, infrastructure resource requirements (and as a result cost), model convergence and model accuracy.

Model parameters are learnt as part of training process, whereas the values of hyperparameters are set before running the training job and they do not change during the training.

Different types of Hyperparameters

Hyperparameters can be broadly divided into 3 categories —

- Model hyperparameters — defines the fundamental construct of a model itself for ex. attributes of a neural network architecture like filter size, pooling, stride, padding.
- Optimizer hyperparameters — are related to how the model learn the patterns based on data. These types of hyperparameters include optimizers like gradient descent and stochastic gradient descent (SGD), Adam, RMSprop, Adadelta and so on.

Q.11 -Gradient Problems are the ones which are the obstacles for Neural Networks to train. Usually you can find this in Artificial Neural Networks involving gradient based methods and back-propagation. But today in deep learning era, various alternate solutions are introduced eradicating the flaws of network learning. This blog will give you a deep insight experience of all sorts of Gradient Problems detailing about its causal situations and solutions. Also, blog will infer with an idea about Neural Network architecture and learning process along with key computations.

I. Vanishing Gradient

Vanishing gradient is a scenario in the learning process of neural networks where model doesn't learn at all. It is due to when gradient becomes too small, almost vanishes leads to weights got stuck and never reach the optimal value for minimal loss(global minima). Thus network not able to learn and converge. And especially during chain rule differentiation, back-propagating from last to initial layer may lead to no updates of weights at all.

II. Exploding Gradient

Exactly opposite to vanishing gradient when model keeps on learning, weights keep on updating large but model never gets converged.Computes gradient (loss) with respect to weights which becomes extremely large in the earlier layers in such a way that it explodes. Keeps on oscillating, taking large step size as shown in above third figure and diverge from the convergence point while moving away from it.

Q.12 -The reason we switch from a linear regression model to a non-linear regression model is because of the output feature variable. While studying linear regression last week, I got a data set in which the dependent variable has categories.

In this article, we will discuss basic concepts regarding logistic regression and learn how we will about maximum likelihood estimates and log(odds). A good understanding is very much important and it saves lots of our time.

First, we need to know why linear regression is not suitable for categories of data. From the graph below we observe that the first one is for linear regression and the second one is also for linear but with binary category values. The insights from these two graphs we can take are that the first graph has values in linearly approach i.e. the independent variable increases the dependent variable is also increased. But, the second graph doesn't tell this type of behaviour rather the dependent variable values are spotted on two values i.e. "0" and "1" only.

If we use the linear approach on the second values the error rate will increase and our model won't fit well and one more thing to be noticed that the linear line is more above and more below the data points that we don't need for prediction. So, we need an approach in which the prediction will be in "0" and "1" only.

Q.13 -Here, we have compared two of the popular Boosting algorithms, Gradient Boosting and AdaBoost.

AdaBoost

AdaBoost or Adaptive Boosting is the first Boosting ensemble model. 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 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 algorithms 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.

Q.14 -In statistics and machine learning, the **bias–variance trade off** describes the relationship between a model's complexity, the accuracy of its predictions, and how well it can make predictions on previously unseen data that were not used to train the model. In general, as we increase the number of tunable parameters in a model, it becomes more flexible, and can better fit a training data set. It is said to have lower error, or bias. However, for more flexible models, there will tend to be greater **variance** to the model fit each time we take a set of samples to create a new training data set. It is said that there is greater variance in the model's estimated parameters.

The **bias–variance dilemma** or **bias–variance problem** is the conflict in trying to simultaneously minimize these two sources of error that prevent surpervised learning algorithms from generalizing beyond their training set.

- The bias error is an error from erroneous assumptions in the learning algorithm . High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- The variance is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).

The **bias–variance decomposition** is a way of analyzing a learning algorithm's expected generalization error with respect to a particular problem as a sum of three terms, the bias, variance, and a quantity called the *irreducible error*, resulting from noise in the problem itself.

Q.15-RBF and Polynomial kernel functions which you can use in SVM are given below:

1. **Polynomial Kernel**

Following is the formula for the polynomial kernel:

$$f(X1, X2) = (X1^T . X2 + 1)^d$$

Here d is the degree of the polynomial, which we need to specify manually.

Suppose we have two features X1 and X2 and output variable as Y, so using polynomial kernel we can write it as:

$$X1^T . X2 = \begin{bmatrix} X1 \\ X2 \end{bmatrix} . [X1 \quad X2]$$

$$= \begin{bmatrix} X1^2 & X1.X2 \\ X1.X2 & X2^2 \end{bmatrix}$$

So we basically need to find X_1^2 , X_2^2 and X1.X2, and now we can see that 2 dimensions got converted into 5 dimensions.

. **RBF Kernel**

What it actually does is to create non-linear combinations of our features to lift your samples onto a higher-dimensional feature space where we can use a linear decision boundary to separate your classes It is the most used kernel in SVM classifications, the following formula explains it mathematically: