MACHINE LEARNING

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

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:- Although they evaluate different aspects of model performance, R-squared and Residual Sum of Squares (RSS) are often used metrics of goodness of fit in regression models.

   The coefficient of determination, or R-squared, quantifies the percentage of the dependent variable's variance that can be accounted for by the independent variables in the model. Higher values suggest a better fit. The range is 0 to 1. R-squared is helpful when comparing models and comprehending the model's overall explanatory power. Its drawbacks include its sensitivity to the quantity of predictors and inability to quantify the accuracy of a single prediction.

   RSS, on the other hand, calculates the total sum of squared residuals, or the variations in the dependent variable's predicted and observed values. It is a representation of the unexplained variation in the data and a tool for evaluating how well the model reduces prediction errors. A better match is indicated by lower RSS values. When assessing the precision of individual forecasts and contrasting various models according to their predictive efficacy, RSS is especially helpful.

   To summarise, RSS concentrates on the precision of individual predictions, whereas R-squared offers a comprehensive assessment of the model's ability to explain the variance in the dependent variable. The particular objectives and specifications of the analysis will determine which of these two measurements is best. It is frequently advised to take into account both metrics simultaneously in order to obtain a thorough grasp of the model's goodness of fit.

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:- TSS, which is computed as the sum of the squared deviations between each observed Y value and the mean of Y, denotes the total variance in the dependent variable (Y). It calculates the dependent variable's overall deviation from the mean.

   The variance in the dependent variable that the regression model explains is represented by ESS. It is computed as the total of the squared discrepancies between the mean of Y and the expected Y values (as determined by the regression model). The expected value divergence from the Y mean is measured by ESS.

   The sum of the squared discrepancies between the observed and predicted values of the dependent variable, or RSS, denotes the unexplained variance in the dependent variable. It calculates how far the observed values deviate from the expected values.

   The following is the equation that links these three metrics:

   $$ESS + RSS = TSS$$

   This formula demonstrates how the explained sum of squares (ESS) and residual sum of squares (RSS) can be separated from the total sum of squares (TSS). The percentage of the total variation in Y that the regression model can explain is represented by the ESS, and the percentage that cannot be explained is represented by the RSS.

   We may determine how much of the variation in the dependent variable is explained by the model and how much is left unexplained by comparing the magnitudes of ESS and RSS.

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

   Answer:- Regularization is a crucial machine learning approach that aids in resolving the overfitting issue. When a model learns the training data too well and performs poorly on unknown data, this is known as overfitting. By including a penalty term in the model's objective function, regularization encourages it to strike a compromise between keeping things simple and fitting the training data as accurately as possible, hence preventing overfitting.

   Regularization is necessary for a number of reasons:

   1. Controlling complexity: Large parameter or high degree of freedom machine learning models are prone to become unduly complex and fitting the noise in the training set. L1 (Lasso) and L2 (Ridge) regularization are two examples of regularization approaches that impose a penalty on the model's parameters to deter them from taking disproportionately high or low values. This keeps overfitting at bay and aids in managing model complexity.

   2. Generalisation: Developing models with strong ability to generalise to previously unknown data is the ultimate goal of machine learning. Regularization lessens a model's susceptibility to noise and outliers in the training set, which helps the model become more generalizable. Regularization helps the model provide more trustworthy and robust predictions on fresh data by preventing it from relying unduly on certain training samples.

   3. Feature selection: One other advantage of regularisation methods such as L1 regularisation is their ability to automatically choose features. L1 regularisation can efficiently remove superfluous or unnecessary features from the model by driving some model coefficients to zero by applying a penalty that promotes sparsity. This lowers the likelihood of overfitting and simplifies the model while also making it easier to interpret.

      In conclusion, machine learning requires regularisation to control model complexity, enhance generalisation, and carry out feature selection. Regularisation aids in the development of more durable, dependable, and precision-predicting models by finding a compromise between fitting the training data and preserving simplicity.

4. What is Gini–impurity index?

   Answer:- The Gini impurity index is a measure used in decision tree algorithms to evaluate the impurity or disorder of a set of data. It quantifies the probability of misclassifying a randomly chosen element in the dataset if it were randomly labeled according to the class distribution of that set.

   Mathematically, the Gini impurity index is calculated by summing the squared probabilities of each class label being chosen, subtracted from 1. It ranges from 0 to 1, where 0 represents a perfectly pure set (all elements belong to the same class) and 1 represents a completely impure set (elements are evenly distributed across all classes).

   In the context of decision trees, the Gini impurity index is used to determine the best split point for dividing the data based on different features. The goal is to minimize the impurity at each split, resulting in more homogeneous subsets of data. By repeatedly splitting the data based on the feature with the lowest Gini impurity, decision trees can effectively classify or predict the target variable.

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

   Answer:- It is true that overfitting occurs in unregularized decision trees. When a model learns training data too well and is unable to generalise successfully to new, unknown data, this is known as overfitting. Overfitting can occur when decision trees generate extremely detailed and specialised rules to match the training set.

   Unrestricted in both structure and complexity, irregularized decision trees are able to develop deeply rooted, intricate trees that precisely match the training set. As a result, the training set of data may contain noise or outliers that the model fails to recognise as actual underlying patterns.

   Unregularized decision trees may therefore find it difficult to generalise and produce reliable predictions when applied to fresh data. The model might have a large variance, which would indicate that it generates inconsistent results and is sensitive to even slight variations in the training set.

Regularization methods can be used to reduce overfitting in decision trees. By limiting the maximum depth, establishing a minimum number of samples needed to divide a node, or pruning the tree after it has been constructed, these strategies impose limitations on the tree's growth. Regularisation improves the model's capacity to generalise to new data, lowers the likelihood of overfitting, and simplifies and reduces the complexity of the model.

6. What is an ensemble technique in machine learning?

Answer:- An ensemble technique in machine learning is the merging of several separate models into a single, more potent, and more precise predictive model. The concept underlying ensemble approaches is that overall performance can be enhanced over the use of a single model by combining the predictions of numerous models.

Various kinds of ensemble approaches exist, such as:

1. Bagging: Using various subsets of the training data, several models are trained independently using this technique. A random subset of the data is used to train each model, and the final forecast is obtained by averaging or voting over the predictions made by each model.

2. Boosting: Boosting is an iterative method in which models are trained one after the other, concentrating on the samples that the earlier models misclassified. Weighted voting is used to aggregate the forecasts of all models, assigning greater weight to the models with superior performance.

3. Random Forest: This well-liked ensemble method blends the ideas of decision trees and bagging. A random portion of the training data and a random subset of the features are used to train each decision tree in the ensemble that is produced. The total of all the trees' projections is used to get the final forecast.

The ability of ensemble approaches to lessen overfitting, enhance generalisation, and identify various patterns in the data makes them useful. They are commonly employed to improve the precision and resilience of predictive models in a variety of machine learning tasks, including regression, classification, and anomaly detection.

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

Answer:- Machine learning models perform better when ensemble learning techniques like bagging and boosting are applied. But they vary in how they go about it and blend different models together.

Bagging:- By using random sampling with replacement, bagging (also known as bootstrap aggregating) entails generating several subsets of the initial training set. Every subset is used to train a different model, and the average or vote of all the individual models' forecasts yields the final prediction. By lessening the influence of noise and outliers in the data, bagging helps lower variance and increase model stability. The bagging algorithms Random Forest and Extra Trees are two examples.

Boosting:- Boosting, on the other hand, aims to improve the model iteratively by increasing the weight of instances that are incorrectly identified. When boosting, models are trained one after the other in an attempt to fix the errors created by the earlier models. All of the separate models' forecasts are combined to produce the final prediction, usually through the use of a weighted voting mechanism. By concentrating on the challenging cases in the data, boosting aids in the reduction of bias and increases model accuracy. AdaBoost, Gradient Boosting, and XGBoost are a few boosting algorithm examples.

In conclusion, the primary distinction between boosting and bagging is in the way that they generate and blend different models. In bagging, different models are created by random sampling with replacement, and their predictions are combined by voting or averaging. Contrarily, boosting focuses on training models one after the other, assigning greater weight to cases that are misclassified, and aggregating their predictions through the use of a weighted voting system.

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

Answer:- Out-of-bag (OOB) error in random forests is a metric used to assess how well the model predicts using samples that were not used in training.

An ensemble learning technique called random forests uses several decision trees combined to provide

predictions. Every tree is trained on a bootstrap sample a random subset of the initial data with replacement during the training process. This indicates that a certain number of samples are excluded or "out-of-bag" in every bootstrap sample.

By assessing each tree's predictions on the out-of-bag data that weren't used to train that specific tree, the OOB error is determined. To gauge the accuracy of the predictions, they are then contrasted with the actual values of the out-of-bag samples. An approximation of the random forest model's performance on hypothetical data is provided by the OOB error.

One benefit of utilising the OOB error is that it offers an alternative means of evaluating the model's performance without requiring a separate validation set. It can be applied to model selection and optimisation, and it functions as an internal validation measure during the training process. Furthermore, the error rate on fresh, unknown data, or generalisation error, of the random forest model can be estimated using the OOB error.

To summarise, the random forests out-of-bag error quantifies the precision of the predictions made using samples that were excluded from the training procedure. It can be applied to model selection and evaluation as it gives an estimate of the model's performance on unknown data.

9. What is K-fold cross-validation?

Answer:- K-fold cross-validation may be a utilized in machine learning and factual modeling to survey the execution and generalization capacity of a prescient show. It includes isolating the accessible information into K similarly measured subsets or "folds."

The handle at that point emphasizes K times, where each time, one of the folds is utilized as the approval set, and the remaining K-1 folds are utilized as the preparing set. The demonstrate is prepared on the preparing set and assessed on the approval set, coming about in a execution metric (e.g., precision, cruel squared blunder) for that iteration.

This process is rehashed K times, with each overlap serving as the approval set precisely once. The execution measurements from each cycle are at that point found the middle value of to get a single execution gauge for the model.

K-fold cross-validation makes a difference to address the issue of overfitting by giving a more vigorous appraise of the model's execution on concealed information. It permits for a more comprehensive assessment of the model's capacity to generalize to modern information by utilizing distinctive subsets of the information for preparing and validation.

Common choices for K incorporate 5 and 10, but the esteem can be balanced based on the measure of the dataset and computational imperatives. K-fold cross-validation could be a utilized strategy for demonstrate assessment and determination, because it gives a more solid gauge of the model's execution compared to a single train-test part.

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

Answer:- In machine learning, choosing the best values for a machine learning model's hyperparameters is known as hyperparameter tuning. Hyperparameters are settings made by the user prior to the model being trained, rather than ones that are inferred from the data. The regularisation parameter, the number of hidden layers in a neural network, and the learning rate are a few examples of hyperparameters.

The process of hyperparameter tuning is used to raise a machine learning model's efficiency. Through the process of optimising hyperparameter values, we can improve the model's capacity for generalisation and improve the precision of its predictions on hypothetical data. It aids in improving outcomes, minimising overfitting, and optimising the model's performance. Usually, methods like grid search, random search, or Bayesian optimisation are used for hyperparameter tuning.

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

Answer:- Several problems may arise in Gradient Descent if we have a high learning rate:

1. Overshooting the minimum: An algorithm may take a long time to get to the loss function's minimum if the learning rate is high. It might, however, overshoot the minimum and continue to oscillate around it

without converging if the learning rate is too high.

2. Divergence: The method may fail to converge and instead diverge while learning at a high rate. As a result, the loss function will rise rather than fall, producing inconsistent and erratic outcomes.

3. Slow convergence: Despite what would seem obvious, a high learning rate can actually cause the algorithm to converge more slowly. This is due to the possibility that the algorithm would continuously exceed the minimum and oscillate around it, aking longer to reach a stable solution.

4. Unstable gradients: A high learning rate may result in unstable gradients, which can cause the model's weights and biases to change dramatically as it is being trained. The model may find it challenging to learn and generalise well as a result.

Gradient Descent requires careful consideration when selecting a learning rate in order to prevent these problems. The issues brought on by a high learning rate can be lessened by employing strategies like learning rate decay or adaptive learning rate approaches.

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

Answer:- As a linear classification procedure, logistic regression makes the assumption that there is a linear relationship between the target variable and the features. As a result, handling non-linear data directly is not appropriate for it. Logistic regression might not be able to fully represent the complex patterns and relationships found in non-linear data.

Nevertheless, by utilising methods like feature engineering or non-linear feature transformations, one can apply logistic regression to non-linear data. This may entail generating polynomial features or interaction terms, which are new features that are combinations or modifications of the original characteristics. By doing this, we can improve the model's ability to handle non-linear data by adding non-linear relationships to it.

Alternatively, other methods specifically made for non-linear data, including decision trees, support vector machines (SVM), or neural networks, may be more appropriate if the non-linear relationship in the data is too complex to be captured by Logistic Regression. These algorithms are capable of efficiently learning and representing non-linear relationships.

13. Differentiate between Adaboost and Gradient Boosting.

Answer:- By merging weak learners, two ensemble learning techniques Adaboost and Gradient Boosting are employed to create effective predictive models. Their methods and the way they adjust the weights of the weaker students vary, nevertheless.

1. Approach :-

Adaboost (Adaptive Boosting): Adaboost trains weak learners repeatedly using various data subsets. In each iteration, it gives misclassified occurrences a bigger weight, which makes such instances more visible to upcoming weak learners.
Gradient Boosting: In contrast, Gradient Boosting creates an ensemble of weak learners in a step-by-step fashion. By fitting the loss function's negative gradient, it teaches each weak learner how to minimise the loss function.

2. Weight Updates:-

Adaboost: Based on the training instances' categorization mistake, Adaboost modifies their weights. It does this by giving misclassified instances higher weights, which amplifies their influence in later iterations.
Gradient Boosting: This technique determines the loss function's negative gradient in order to update the weights. It allows succeeding learners to concentrate on the remaining errors by fitting each weak learner to the residuals (the difference between the actual and anticipated values) of the preceding weak learner.

3. Weak Learners:-

Adaboost: Decision trees, often known as "stumps" (trees with only one split), are commonly used by

Adaboost as weak learners. Different data subsets are used to train these decision stumps.
Gradient Boosting: This technique makes use of a variety of weak learners, including regression models, decision trees, or even neural networks. These weak learners are typically shallow trees with a small number of nodes.

4. Training Process:-

The Adaboost training method involves training weak learners in a step-by-step fashion, with each student attempting to rectify the faults committed by the learners before them. It uses weighted voting to aggregate all weak learners' predictions.
Gradient Boosting: Gradient Boosting is a stage-wise training method that fits the negative gradient to minimise the loss function for each learner. The final forecast is the sum of the guesses made by all weak learners.

In conclusion, the training procedure, the kind of weak learners employed, and the weight update algorithms employed by Adaboost and Gradient Boosting are different. While Gradient Boosting minimises the loss function by fitting the negative gradient, Adaboost concentrates on misclassified cases and gives them larger weights.

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

Answer:- In machine learning, the balance between two types of errors that can occur in a model bias and variance is referred to as the bias-variance trade-off. When a model continually fails to recognise the genuine underlying patterns in the data, it is said to be biassed. Bias is the error induced by the model's assumptions or simplifications. Contrarily, variance is the inaccuracy brought about by the model's sensitivity to changes in the training set, which can lead to overfitting and poor performance on fresh, untested data.

In machine learning, striking the correct balance between variance and bias is essential. While a model with high variance may overfit the data and perform poorly when applied to new data, a model with high bias may underfit the data and perform poorly when applied to new data. To obtain the greatest feasible model performance, the objective is to minimise both bias and variance.

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

Answer:- The following provides a brief overview of every kernel utilised in Support Vector Machines (SVM):

1. Linear Kernel: The most basic kernel in SVM is the linear kernel. When the data is linearly separable, it performs well and represents the data in its original feature space. The dot product between two data points is computed.

2. Kernel for Radial Basis Function (RBF): In SVM, the RBF kernel is a prominent option. It can handle both linearly and non-linearly separable data since it translates the data into a higher-dimensional space. Based on their separation, it calculates how similar two data points are to one another.

3. Polynomial Kernel: Non-linearly separable data are handled by the polynomial kernel. It uses polynomial functions to map the data into a higher-dimensional space. The decision boundary's difficulty is based on the polynomial's degree.

These kernels offer diverse methods for data transformation and analysis, enabling SVM to successfully categorise data points in a range of situations.