Q-: 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 is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable in a regression model. The residual sum of squares (RSS) measures the difference between your observed data and the model's predictions. It is the portion of variability your regression model does not explain, also known as the model's error. Use RSS to evaluate how well your model fits the data. So, R-squared is the standard goodness-of-fit measure for regression. Its formula incorporates RSS. Typically, you'll evaluate R-squared rather than the RSS because it avoids some of RSS's limitations, making it much easier to interpret.

Q-: 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-: The total sum of squares (TSS) is the sum of squared differences between the observed dependent variables and the overall mean.

The explained sum of squares (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 residual sum of squares is used to help you decide if a statistical model is a good fit for your data. It measures the overall difference between your data and the values predicted by your estimation model.

equation relating these three metrics with each other

Total SS = Explained SS + Residual Sum of Squares.

Q-: What is the need of regularization in machine learning?

Ans-: Regularization in Machine Learning. Regularization is a technique used to reduce errors by fitting the function appropriately on the given training set and avoiding overfitting. In other word regularization is a set of methods for reducing overfitting in machine learning models.

Q-: What is Gini–impurity index?

Ans-: The Gini Index is the additional approach to dividing a decision tree. Decision Tree is one of the most popular and powerful classification algorithms that we use in machine learning. The decision tree from the name itself signifies that it is used for making decisions from the given dataset.

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

Ans:- Unregularized decision trees are prone to overfitting due to their high complexity and tendency to adapt too closely to the training data. Unregularized decision trees can learn the noise in the data, leading to overly complex models.

Q-: What is an ensemble technique in machine learning?

Ans-: Ensemble means 'a collection of things' and in Machine Learning terminology, Ensemble learning refers to the approach of combining multiple ML models to produce a more accurate and robust prediction compared to any individual model. It implements an ensemble of fast algorithms (classifiers) such as decision trees for learning and allows them to vote. Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.

Q-: What is the difference between Bagging and Boosting techniques?

Ans-: Bagging is a machine learning ensemble method that aims to reduce the variance of a model by averaging the predictions of multiple base models. The key idea behind Bagging is to create multiple subsets of the training data (bootstrap samples) and train a separate base model on each of these subsets. These base models can be of any type, such as decision trees, neural networks, or regression models.

Boosting is another ensemble learning method that focuses on improving the accuracy of a model by sequentially training a series of base models. Unlike Bagging, where base models are trained independently, boosting trains each base model in a way that emphasizes the examples that the previous models misclassified. The idea is to give more weight to the misclassified samples so that the subsequent models focus on these challenging cases. The final prediction is then made by combining the predictions of all base models, giving more weight to those that performed better during training.

S. Num	Bagging	Boosting
1	This is the simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types
2	This aims to decrease variance, not bias	This aims to decrease bias, not variance
3	In this each model receives equal weight	In this model are weighted according to their performance
4	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
5	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
6	In this each model is built independently	New models are influenced by the performance of previously built models
7	Different training data subsets are selected using row sampling with replacement and random sampling methods from	Every new subset contains the elements that were misclassified by previous models

	the entire training dataset	
8	In this base classifier are trained parallelly.	In this base classifier are trained sequentially

Q-: What is out-of-bag error in random forests?

Ans-: Out-of-bag errors are an estimate of the performance of a random forest classifier or regressor on unseen data. The Out-of-Bag error can be obtained using the oob-score attribute of the random forest classifier or regressor.

Ans-: k-fold cross-validation is one of the most popular strategies widely used by data scientists. K-fold cross-validation is a technique for evaluating predictive models. It is a data partitioning strategy so that you can effectively use your dataset to build a more generalized model. The dataset is divided into k subsets or folds. The main intention of doing any kind of machine learning is to develop a more generalized model which can perform well on unseen data.

Q-: What is hyperparameter tuning in machine learning and why it is done?

Ans-: Hyperparameters are set manually to help in the estimation of the model parameters. They are not part of the final model equation.

It is an external configuration variable that data scientists use to manage machine learning model training. Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

Hyperparameters directly control model structure, function, and performance. Hyperparameter tuning allows data scientists to tweak model performance for optimal results. This process is an essential part of machine learning, and choosing appropriate hyperparameter values is crucial for success.

Q-: What issues can occur if we have a large learning rate in Gradient Descent?

Ans-: It can lead to several issues, including:

a- Instability: Large learning rates can cause instability in the training process. The updates to the model parameters can be too drastic, leading to erratic behavior and difficulty in finding an optimal solution.

b- Divergence: A large learning rate can cause the optimization process to overshoot the minimum of the loss function, resulting in divergence. Instead of converging to the minimum, the parameter updates may oscillate or move away from the optimum, making the optimization process unstable.

c- Poor Generalization: Large learning rates can prevent the model from generalizing well to unseen data. The model may learn to fit the training data too closely, leading to overfitting and poor performance on new, unseen data.

d- Sensitivity to Initialization: Large learning rates can make the optimization process highly sensitive to the initial values of the model parameters. Small changes in the initial values can lead to significantly different convergence behavior or even divergence.

e- Overshooting the Minimum: With a large learning rate, the optimization algorithm may overshoot the minimum of the loss function. This can lead to a zig-zagging pattern around the minimum or bouncing back and forth across the optimum, preventing convergence.

f- Difficulty Converging: Even if the optimization process does not diverge, a large learning rate can slow down convergence. The algorithm may require more iterations to reach the minimum of the loss function, increasing the computational cost of training.

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

Ans-: Yes, Logistic Regression can still be used for classification of non-linear data by incorporating non-linear transformations of the input features.

Here are some approaches to using Logistic Regression for non-linear classification. (a)Feature Engineering:

(b)Feature Mapping

(c)Ensemble Methods

(d)Regularization

(e)Kernel Logistic Regression

Q-: Differentiate between Adaboost and Gradient Boosting?

Ans-: Gradient Boosting algorithm is more robust to outliers than AdaBoost. 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.

Q-: Give short description each of Linear, RBF, Polynomial kernels used in SVM?

Ans:- Linear SVM: The linear kernel is one of the most basic and widely used kernels. A kernel function essentially computes the inner product of two feature vectors in a higher-dimensional space. The linear kernel, also known as the dot-product kernel, computes this inner product directly in the original feature space. When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line.

RBF-: The Radial Basis Function (RBF) kernel is one of the most powerful, useful, and popular kernels in the Support Vector Machine (SVM) family of classifiers.

Polynomial -: A polynomial kernel is a kind of SVM kernel that uses a polynomial function to map the data into a higher-dimensional space. It does this by taking the dot product of the data points in the original space and the polynomial function in the new space. Polynomial kernels are used when the data is not linearly separable and can be separated by a polynomial function.