MACHINE LEARNING

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 ?

R-squared indicates the proportion of variance explained by the model and is intuitive, but it factors in the number of variables, potentially leading to overfitting. RSS measures the model fit directly by summing squared residuals, with a lower RSS indicating a better fit. The choice between them depends on dataset specifics, and commonly a combined approach alongside other metrics, like adjusted R-squared, provides a clearer measure of model fit. To determine which is a better measure of the goodness of fit model in regression between R-squared and Residual Sum of Squares (RSS), we need to consider both measures in the context of regression analysis. R-squared, or the coefficient of determination, indicates the proportion of variance in the dependent variable that is predictable from the independent variables. It is obtained by comparing the fitted model to a model with no explanatory variables except for the mean of the dependent variable. As R-squared approaches 1.0, it suggests a model with a better fit.

RSS, on the other hand, measures the overall model fit by summing up the squared differences between observed and predicted values, which are known as the squared residuals. In contrast to R-squared, the lower the RSS, the better the model's predictions match the actual data.

While R-squared is commonly used for its interpretability as a percentage of the variance explained, it is not without its limitations. Particularly, R-squared will always increase as more variables are added, regardless of whether those variables are meaningful to the model. This is where the adjusted R-squared comes into play in multiple regression, as it adjusts for the number of variables and helps prevent overfitting. Furthermore, if we consider non-linear models, the mean may not be a feature, making R-squared harder to calculate or interpret

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also the equation relating these three metrics with each other.

Total Sum of Square in regression analysis measures the total variability in your data. It represents the cumulative sum of squares of the differences between each data point and the mean of the dependent variable.

Explained Sum of Squares measures how well a regression model represents the data being modeled. Specifically, it quantifies how much variation in the model’s predicted values (based on explanatory variables) explains the observed data.
In a standard regression model, where we have a response variable

Residual Sum of Squares measures the difference between your observed data and the model productions . It represents the portion of variability that your regression model does not explain , also known as the models error

3. What is the need for regularization in machine learning.

Regularization is a crucial technique used to prevent overfitting, ensuring that models generalize well to new, unseen data. Here's why we When a machine learning model fits tightly to the training data and learns all the details, it may not generalize well to unseen data. Overfitting occurs when the model captures noise and specific patterns in the training data, leading to poor performance on new data. The generalization curve illustrates this trade-off. While minimizing training loss is desirable, validation loss starts increasing after a certain point. This divergence indicates overfitting Regularization helps control model complexity by adding a penalty to the loss function. It encourages simpler models that capture underlying patterns without fitting noise. Common techniques include: Shrinks coefficient estimates by adding a penalty term to the cost function. It keeps the magnitude of model weights small.
Encourages sparsity by forcing some coefficients to exactly zero. It selects relevant features.
Combines L1 and L2 regularization, balancing their effects

4. What is Gini impurity index?
   The Gini Impurity is a measurement used in building Decision Trees. It helps determine how features of a dataset should split nodes to form the tree. Specifically, the Gini Impurity of a dataset is a number between **0** and **0.5**. Imagine you want to build a classifier to predict credit card defaults. You have labeled data with features like age, income, credit rating, and student status. To find the best feature for the first split (the root node) in the decision tree, you calculate how poorly each feature divides the data into the correct classes (default or no default). The feature with the lowest impurity determines the best split. This process continues for subsequent nodes using remaining features.

5. Are unregularized decision trees prone to overlifting? If yes why?

Yes, unregularized decision trees can be prone to overfitting. Here's Overfitting occurs when a model becomes too complex, capturing noise and irrelevant patterns in the training data instead of the underlying true patterns. Essentially, the model becomes too specialized in the training data and fails to generalize well to unseen data.

Decision trees are prone to overfitting because they can create deep, intricate structures that perfectly fit the training data. Without regularization, they may learn to memorize the training examples, including noise, rather than capturing the essential relationships.

Unregularized decision trees have no constraints on their growth. They can keep splitting nodes until each leaf contains only one data point. This leads to a high variance model that doesn't generalize well to new data. To mitigate overfitting, consider pruning the decision tree. Pruning involves removing branches or nodes that don't significantly improve predictive accuracy.

6. What is an ensemble technique in machine learning?

Ensemble learning is a powerful technique that combines the predictions from multiple individual models to improve predictive performance. Here's how it works. Ensemble learning leverages the wisdom of the crowd by aggregating predictions from diverse models. Each individual model may have its own strengths and weaknesses, but when combined, they create a more accurate and robust prediction. Bagging involves training multiple base models (such as decision trees) on different subsets of the same dataset. It then averages their predictions to reduce variance and enhance generalization.

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

Bagging (Bootstrap Aggregating) and Boosting are both ensemble learning techniques that combine multiple models to improve performance, but they differ in approach:

Bagging

Bagging uses parallel modeling and sampling with replacement.
It aims to reduce variance by averaging predictions from diverse base models.

Bagging can use complex base models (e.g., decision trees).
Random Forest, which aggregates decision trees using bagging.

Boosting

Boosting builds models sequentially without samplingIt aims to reduce bias by correcting errors made by previous models.
Boosting typically relies on simple base models (e.g., decision trees).
Gradient Boosting, XGBoost, and AdaBoost.

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

is an estimate of the model's performance on unseen data. Here's how it works:

In random forests, each tree is trained using a bootstrap sample (a subset of the training data). However, some data points are left out during the sampling process for each tree.
The OOB error is calculated using predictions from the trees that do not contain a specific data point in their respective bootstrap samples. Essentially, it measures how well the model performs on the data points that were not included during training.
The OOB error allows the random forest to be fit and validated simultaneously. It provides an estimate of how well the model generalizes to unseen data without the need for a separate validation set

9. What is K fold cross validation ?

K-Fold Cross-Validation is a robust technique used to evaluate the performance of machine learning models. Here's how it works. Each fold serves as the validation set , while the remaining k folds are used for training. The model is trained and evaluated k times, with each fold taking turns as the validation set. This repeated process ensures thorough testing. The final performance metric of the performance across all k iterations. K-Fold CV helps ensure that the model generalizes well to unseen data.
It provides a more reliable estimate of model performance than a single train/test split.

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

in machine learning refers to the process of selecting optimal values for a model's hyperparameters. These hyperparameters control various aspects of the learning process and are set before training begins. Unlike regular model parameters (such as weights and biases), hyperparameters cannot be directly learned from data. Here's why hyperparameter tuning is essential. Choosing the right hyperparameters significantly impacts a model's accuracy, generalization, and other performance metrics. Tuning them ensures the best possible model performance.
Different tasks and datasets require different hyperparameter settings. Tuning allows customization to suit specific problems.

Properly tuned hyperparameters prevent overfitting (when a model performs well on training data but poorly on unseen data) by controlling model complexity. Hyperparameters affect the stability and convergence speed of training. Well-tuned values lead to faster convergence and stable learning.

11. What issues can occur if we have a large learning rate in gradient descent?

When using a large learning rate in gradient descent, several issues can arise:

Gradient descent aims to find the minimum of a loss function. If the learning rate (step size) is too large, the algorithm can "jump over" the minima we want to reach. _The model's parameters become unstable. The algorithm updates them too aggressively, causing them to fluctuate wildly and fail to settle at optimal values. When the learning rate is too large, gradient descent diverges.

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

Traditionally, logistic regression has been used as a linear classifier. It creates a hyperplane in the feature space to separate observations belonging to a class from those that do not. However, when we suspect that the decision boundary is nonlinear, we can explore alternative approaches. If we have a better idea about the shape of the decision boundary, we can attempt nonlinear functions for the logit function. For instance, we can express the decision boundary as a polynomial in feature variables but linear in the weights or coefficients. This can still be modeled within the linear framework using libraries like scikit-learn. In cases where the decision boundary cannot be expressed as a polynomial, we need to find model parameters that maximize the likelihood function. For such scenarios, we can use optimization techniques available in libraries like spicy.

13. Differentiate between Adaboost and Gradient Boosting.

Boosting algorithms are one of the best-performing algorithms among all the other machine learning with the best performance and higher accuracies. All the boosting algorithms work on the basis of learning from the errors of the previous model trained and tried avoiding the same mistakes made by the previously trained weak learning algorithm.

It is also a big interview question that might be asked in data science interviews. In this article, we will be discussing the main difference between GradientBoosting, AdaBoost, XGBoost, CatBoost, and LightGBM algorithms, with their working mechanisms and their mathematics of them. Gr is the boosting algorithm that works on the principle of the stagewise addition method, where multiple weak learning algorithms are trained and a strong learner algorithm is used as a final model from the addition of multiple weak learning algorithms trained on the same dataset.

In the gradient boosting algorithm, the first weak learner will not be trained on the dataset, it will simply return the mean of the particular column, and the residual for output of the first weak learner algorithm will be calculated which will be used as output or target column for next weak learning algorithm which is to be trained.

Following the same pattern, the second weak learner will be trained and the residuals will be calculated which will be used as an output column again for the next weak learner, this is how this process will continue until we reach zero residuals.

In gradient boosting the dataset should be in the form of numerical or categorical data and the loss function using which the residuals are calculated should be differential at all points

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

The bias variance tradeoff is a fundamental concept in machine learning that involves balancing model complexity and generalization performance. Here’s how it works
bias refers to the error introduced by approximating a real-world problem with a simplified model.

High bias occurs when a model is too simplistic and fails to capture the underlying patterns in the data.
High bias models tend to underfit the training data, resulting in poor performance.Variance represents the model’s sensitivity to small fluctuations in the training data.
High variance occurs when a model is overly complex and fits the noise in the data rather than the true signal.
High variance models tend to overfit the training data, performing well on training data but poorly on unseen data.
The bias-variance tradeoff aims to find the right balance between bias and variance.
Increasing model complexity (e.g., adding more features or increasing polynomial degree) reduces bias but increases variance.
Decreasing model complexity (e.g., using simpler models or regularization) increases bias but reduces variance.
The goal is to minimize the total error which is the sum of bias squared and variance.
There exists an optimal point where the total error is minimized.

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

It’s a mathematical function commonly used in machine learning, interpolation, and approximation. In machine learning, RBF networks use these functions as activation functions in neural networks. They’re particularly useful for solving problems involving non-linear relationships. If you have any more questions or need further clarification

This is the simplest kernel function. It is used when the data is linearly separable, meaning a straight line (or hyperplane in higher dimensions) can separate the classes.