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?

Ans) In regression analysis, R-squared is generally considered a better measure of goodness of fit compared to Residual Sum of Squares (RSS) because:

Interpretability: R-squared provides a normalized value between 0 and 1, indicating the proportion of variance in the dependent variable that is predictable from the independent variables. This makes it easier to interpret.

Relative Comparison: R-squared allows for easier comparison between different models. A higher R-squared value indicates a better fit relative to other models, while RSS is an absolute measure that does not provide context for comparison.

Scale Independence: R-squared is scale-invariant, meaning it doesn't depend on the units of measurement. In contrast, RSS can vary significantly depending on the scale of the data.

However, it's important to consider that R-squared can sometimes be misleading, particularly in cases of overfitting, where adding more variables may artificially inflate the R-squared value without improving the model's predictive power. Adjusted R-squared is often used to account for this issue.

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.

Ans) In regression analysis, TSS, ESS, and RSS are metrics that help evaluate the goodness of fit of a model:

Total Sum of Squares (TSS): TSS measures the total variability in the dependent variable. It represents the total variation from the mean of the dependent variable. It is calculated as: $TSS=\sum(yi - \bar{y})^{**}2$

where yi is each observed value and $\bar{y}$ is the mean of the observed values.

Explained Sum of Squares (ESS): ESS measures the variability explained by the regression model. It represents the portion of the total variability that is accounted for by the model. It is calculated as: $ESS=\sum(y\hat{}i-\bar{y})^{**}2$

where $y\hat{}i$ is the predicted value from the regression model.

Residual Sum of Squares (RSS): RSS measures the variability that is not explained by the model. It represents the error in the predictions. It is calculated as: $RSS=\sum(yi-y\hat{}i)^{**}2$

The relationship among these three metrics is given by the equation: TSS=ESS+RSS

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

Ans) Regularization is a set of techniques used in machine learning to prevent overfitting and improve a model's ability to generalize:

Preventing overfitting:- Regularization prevents a model from learning the noise and underlying patterns in training data. Overfitting occurs when a model's error rate on training data decreases, but its error rate on testing data increases or stops decreasing.

Improving generalization:- Regularization helps a model perform well on new data by limiting its complexity. This improves the model's ability to make accurate predictions on new datasets.

Handling multicollinearity:- Regularization is particularly useful when features are highly correlated.

Improving robustness to noise:- Regularization makes a model less sensitive to idiosyncrasies in the training data.

Aiding in convergence:- Regularization can help ensure smoother and more reliable convergence for models trained using iterative optimization techniques.

4) What is Gini–impurity index?

Ans) The Gini impurity index is a measure used in decision tree algorithms to evaluate the purity of a dataset or a node in the tree. It quantifies how often a randomly chosen element from the dataset would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

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

Ans) Yes, unregularized decision trees are indeed prone to overfitting. This tendency arises from several key factors:

a) High Complexity: Decision trees can grow very deep, capturing intricate patterns and noise in the training data. When a tree is allowed to grow without restriction, it may create branches for every minor fluctuation in the data, resulting in a model that fits the training data perfectly but fails to generalize to unseen data.

b) Low Bias and High Variance: Unregularized decision trees have low bias because they can represent a wide variety of functions. However, they also have high variance, meaning that their predictions can change significantly with small changes in the training data. This high variance can lead to overfitting, where the model performs well on training data but poorly on validation or test data.

c) Lack of Pruning: Without regularization techniques like pruning, a decision tree will continue to split until all data points are perfectly classified. This can result in a highly complex tree structure that captures noise instead of the underlying data distribution.

d) Sensitivity to Noise: Decision trees are sensitive to outliers and noisy data. In an unregularized tree, even a small amount of noise can lead to splits that don't reflect the true structure of the data, further contributing to overfitting.

6) What is an ensemble technique in machine learning?

Ans) An ensemble technique in machine learning is a method that combines multiple individual models (often referred to as "base models" or "learners") to produce a more powerful and accurate predictive model. The primary goal of using ensemble methods is to improve overall model performance by leveraging the strengths of various models while mitigating their weaknesses.

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

Ans)

SNO	Bagging	Boosting
1	Bagging is a learning approach that aids in enhancing the performance, execution, and precision of machine learning algorithms.	Boosting is an approach that iteratively modifies the weight of observation based on the last classification.
2	It is the easiest method of merging predictions that belong to the same type.	It is a method of merging predictions that belong to different types.
3	Here, every model has equal weight.	Here, the weight of the models depends on their performance.
4	In bagging, each model is assembled independently.	In boosting, the new models are impacted by the implementation of earlier built models.
5	It helps in solving the over-fitting issue.	It helps in reducing the bias.
6	In the case of bagging, if the classifier is unstable, then we apply bagging.	In the case of boosting, If the classifier is stable, then we apply boosting.

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

Ans) Out-of-bag (OOB) error is a measure of prediction error used specifically in the context of random forests and other bagging techniques. It provides an estimate of the model's performance without the need for a separate validation set.

9) What is K-fold cross-validation?

Ans) K-fold cross-validation is a robust technique used to assess the performance and generalizability of a machine learning model. It involves partitioning the dataset into K subsets or "folds" and systematically training and validating the model.

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

Ans) Hyperparameter tuning in machine learning refers to the process of selecting the optimal set of hyperparameters for a model to improve its performance. Hyperparameters are the parameters that are not learned from the data during training; instead, they are set before the training process begins. Examples include the learning rate, the number of trees in a random forest, the depth of a decision tree, and the number of clusters in k-means clustering.

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

Ans) Using a large learning rate in gradient descent can lead to several significant issues that affect the training process and model performance:

a) Overshooting the Minimum: A large learning rate can cause the updates to overshoot the optimal solution, resulting in the loss function oscillating back and forth rather than converging. This can prevent the algorithm from finding the minimum.

b) Divergence: If the learning rate is too high, the updates can become so large that the loss function starts to increase instead of decrease. In extreme cases, this can lead to divergence, where the loss grows indefinitely, and the algorithm fails to converge.

c) Instability: High learning rates can create instability in the training process, leading to erratic updates and unpredictable behavior. This instability makes it difficult to reliably minimize the loss function.

d) Poor Generalization: Even if the model manages to find a solution with a high learning rate, the learned parameters may be suboptimal. This can result in poor generalization to unseen data, as the model might converge to a suboptimal local minimum.

e) Training Time: The inefficiency caused by oscillations and divergence can significantly increase training time, as the model may take longer to stabilize or may require additional iterations to find an acceptable solution.

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

Ans) Logistic regression is primarily designed for binary classification tasks and assumes a linear relationship between the input features and the log-odds of the output. Therefore, using logistic regression directly for non-linear data can lead to several challenges:

a) Limitations of Logistic Regression for Non-Linear Data: Linear Decision Boundary: Logistic regression creates a linear decision boundary. If the actual relationship between the features and the target variable is non-linear, logistic regression may not capture the underlying patterns effectively.

b) Underfitting: When applied to non-linear data, logistic regression is likely to underfit the model. It may fail to classify the data points correctly, as it cannot adapt to the complexities and interactions present in non-linear relationships.

c) Poor Performance: If the relationship between features and the target variable is non-linear, the model's performance metrics (accuracy, precision, recall, etc.) will typically be lower than those of models that can capture non-linearities.

13) Differentiate between Adaboost and Gradient Boosting.

Ans) The most significant difference is that gradient boosting minimizes a loss function like MSE or log loss while AdaBoost focuses on instances with high error by adjusting their sample weights adaptively.

Gradient boosting models apply shrinkage to avoid overfitting which AdaBoost does not do. Gradient boosting also performs subsampling of the training instances while AdaBoost uses all instances to train every weak learner.

Overall gradient boosting is more robust to outliers and noise since it equally considers all training instances when optimizing the loss function. AdaBoost is faster but more impacted by dirty data since it fixates on hard examples.

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

Ans) The bias-variance trade-off is a fundamental concept in machine learning that describes the balance between two sources of error that affect a model's performance: bias and variance. Understanding this trade-off is crucial for building models that generalize well to unseen data.

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

Ans) In Support Vector Machines (SVM), kernels are functions that enable the algorithm to operate in a higher-dimensional space without explicitly transforming the data. Here’s a short description of three common kernel types: Linear, RBF (Radial Basis Function), and Polynomial kernels.

a) **Linear Kernel:** It is effective for linearly separable data where a straight line (or hyperplane) can separate classes. It is computationally efficient and often used when the data is already linearly separable.

b) **RBF (Radial Basis Function) Kernel:** It is effective for non-linearly separable data and can create complex decision boundaries. The RBF kernel is widely used due to its ability to handle a wide variety of data distributions.

c) **Polynomial Kernel:** It is useful for scenarios where the relationship between classes is polynomial rather than linear. The polynomial kernel can create curved decision boundaries and can be useful for capturing interactions between features.