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. 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. Both measures are useful in evaluating the goodness of fit of a model, but they serve different purposes. R-squared is a useful measure to assess the overall fit of the model and to compare different models, while RSS is useful to identify the degree of the error in the model's predictions.

In general, a good model should have both a high R-squared value and a low RSS value, indicating that it explains a large proportion of the variation in the dependent variable and has a low degree of error in its predictions. However, in some cases, one measure may be more important than the other, depending on the research question and the nature of the data being analysed

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. TSS is the sum of squared differences between the observed dependent variables and the overall mean. ESS is the sum of the squares of the deviations of the predicted values from the mean value of a response variable, in a standard regression model. The residual sum of squares (RSS) is a statistical technique used to measure the variance in a data set that is not explained by the regression model.

TSS = ESS + RSS

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

Ans. While training a machine learning model, the model can easily be over fitted or under fitted. To avoid this, we use regularization in machine learning to properly fit a model onto our test set. Regularization techniques help reduce the chance of over fitting and help us get an optimal model.

4. What is Gini–impurity index?

Ans. Gini impurity is a measure used in decision tree algorithms to quantify a dataset's impurity level or disorder.

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

Ans. Yes, Unregularized decision trees can keep splitting nodes based on the provided data until they have essentially "memorized" it. This can lead to high variance, meaning the model performs well on the training data but poorly on new, unseen data. The model captures noise and specific patterns that might not generalize well. Regularization techniques, such as pruning or using constraints like maximum depth, minimum samples per leaf, or maximum number of leaf nodes, can help prevent overfitting in decision trees. These techniques aim to simplify the tree structure, stopping the growth before it becomes too complex and captures noise.

6. What is an ensemble technique in machine learning?

Ans. Ensemble techniques in machine learning involve combining multiple individual models to produce a stronger, more accurate predictive model than any of the individual models alone. The idea behind ensembling is to leverage the wisdom of the crowd – combining diverse models can often yield better results than relying on a single model.

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

Ans. *Independence vs. Dependence:* Bagging models are trained independently, while Boosting models are trained sequentially and dependent on each other.

Parallel vs. Sequential Learning: Bagging trains models in parallel, while Boosting trains models sequentially.

Weighting of Instances: Boosting assigns more weight to misclassified instances to emphasize learning from them, whereas Bagging treats all instances equally.

Both Bagging and Boosting aim to reduce variance, improve accuracy, and enhance the overall performance of machine learning models, but they achieve these goals through different strategies of combining multiple models.

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

Ans. In Random Forest, the out-of-bag (OOB) error is an estimate of the model's performance without the need for a separate validation set or cross-validation. It's a way to evaluate the performance of the Random Forest model using the data that was not used in the training of each individual decision tree.

9 . What is K-fold cross-validation?

Ans. K-fold cross-validation is a widely used technique in machine learning for assessing the performance of a model and for hyper parameter tuning. It's a resampling procedure used to evaluate models on a limited dataset in a more robust way than simple train-test splits.

Here's a step-by-step explanation of K-fold cross-validation:

Partitioning the Dataset: The dataset is divided into K equally sized subsets or folds.

Iterative Process: The model training and evaluation process is repeated K times.

Training and Validation: In each iteration:

One of the K subsets is used as the validation set.

The model is trained on the remaining K-1 subsets (the training set).

Evaluation Metric: The performance of the model is evaluated using a chosen metric (e.g., accuracy, mean squared error, etc.) on the validation set for that iteration.

Average Performance: After K iterations, K different performance scores are obtained (one for each fold). These scores are averaged to obtain a single estimation of the model's performance.

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

Ans. Hyperparameter tuning in machine learning involves finding the optimal hyperparameters for a given model. Hyperparameters are settings or configurations that are external to the model and cannot be directly learned from the data during training. They govern the learning process, affecting the model's performance and behavior.

Some examples of hyperparameters in machine learning models include learning rates, regularization parameters, tree depths in decision trees, the number of clusters in clustering algorithms, etc.

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

Ans. When using gradient descent, the learning rate is a crucial hyperparameter that determines the step size taken towards the minimum of the loss function during each iteration. If the learning rate is too large, it can lead to several issues, including:

a. Divergence or Overshooting the Minimum

b. Instability and Unstable Convergence

c. Missing the Minimum

d. Inability to Converge

e. Sensitivity to Noise or Fluctuations

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

Ans. Logistic Regression is a linear classification algorithm that's particularly well-suited for binary classification tasks. It models the relationship between the independent variables and the probability of a certain outcome using a linear equation.

The decision boundary in logistic regression is a linear boundary, which means it forms a straight line (or hyper plane in higher dimensions) in the feature space. As a result, logistic regression is inherently limited in its ability to handle non-linear relationships between features and the target variable.

When the relationship between features and the target variable is non-linear, using logistic regression directly may not capture complex patterns present in the data. In such cases, logistic regression might struggle to fit the data well and might not provide accurate predictions.

However, there are ways to use logistic regression for non-linear data by introducing non-linear transformations of the features or by employing techniques like feature engineering or adding polynomial terms to the model. These techniques allow logistic regression to capture non-linear relationships indirectly by transforming the features into a higher-dimensional space where linear separation might be possible.

Additionally, for dealing with inherently non-linear data, more sophisticated models such as decision trees, random forests, support vector machines (SVMs), neural networks, or kernel-based methods (e.g., kernel SVM) are often more suitable. These models have the capacity to learn and represent complex non-linear decision boundaries, making them more effective for handling non-linear data compared to logistic regression.

13. Differentiate between Adaboost and Gradient Boosting.

Ans. Adaboost (Adaptive Boosting) and Gradient Boosting are both popular boosting algorithms used in machine learning for ensemble learning, but they differ in their approach, training procedure, and how they build the ensemble of models.

Here are the key differences between Adaboost and Gradient Boosting:

Training Process:

Adaboost: Adaboost assigns weights to the training instances and trains a series of weak learners (usually decision trees) sequentially. It adjusts the weights of incorrectly classified instances in each iteration, allowing subsequent models to focus more on these misclassified instances.

Gradient Boosting: Gradient Boosting builds a sequence of models, typically decision trees, in a sequential manner, where each subsequent model corrects the errors made by the previous one. It minimizes the errors (residuals) of the previous model by fitting new models to the residuals using gradient descent.

Weighting of Instances:

Adaboost: Adaboost assigns higher weights to misclassified instances to emphasize learning from them in subsequent iterations. It gives more weight to instances that were incorrectly classified by previous models.

Gradient Boosting: Gradient Boosting focuses on minimizing the errors (residuals) of the previous model. It trains each new model on the residuals (the difference between predictions and actual values) of the previous model.

Model Building:

Adaboost: Adaboost combines weak learners into a strong learner by giving more weight to the predictions of those weak learners that perform better on the training data.

Gradient Boosting: Gradient Boosting builds an ensemble of models by sequentially adding models that predict the residuals or errors of the previous models, gradually reducing the overall error.

Base Learners:

Adaboost: Adaboost typically uses decision trees as weak learners, often shallow trees (stumps) with limited depth.

Gradient Boosting: Gradient Boosting commonly uses decision trees as base learners but can employ more complex trees with higher depth. It can also use other base learners like linear regression, neural networks, etc.

Learning Rate:

Adaboost: Adaboost uses a learning rate to control the contribution of each weak learner to the final prediction, adjusting weights based on misclassifications.

Gradient Boosting: Gradient Boosting employs a learning rate to scale the contribution of each tree to the ensemble, affecting the step size during the optimization process.

Both Adaboost and Gradient Boosting aim to improve model accuracy by combining multiple weak learners into a strong ensemble model, but they differ in their approach to training, instance weighting, and model building strategies.

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

Ans. The bias-variance tradeoff is a fundamental concept in machine learning that describes the balance between model complexity and model generalization. It refers to the tradeoff between the model's ability to capture the true underlying patterns in the data (i.e., bias) and its sensitivity to fluctuations or noise in the dataset (i.e., variance).

Bias: Bias refers to the error introduced by approximating a real problem with a simplified model. A high bias means the model oversimplifies the underlying patterns in the data, leading to underfitting. Models with high bias may fail to capture important relationships between features and target variables, resulting in consistently wrong predictions (inaccuracy).

Variance: Variance refers to the model's sensitivity to fluctuations or noise in the training data. A high variance model is overly complex and captures not only the underlying patterns but also the noise in the data, leading to overfitting. Models with high variance perform well on the training data but fail to generalize to new, unseen data, causing high prediction errors.

The tradeoff occurs because decreasing bias often increases variance, and reducing variance can increase bias. Finding the right balance between bias and variance is crucial for building models that generalize well to new data. The goal is to achieve a model that has low bias (to capture the underlying patterns) and low variance (to avoid overfitting).

Ways to address the bias-variance tradeoff include:

Model Complexity: Adjusting the complexity of the model: simpler models tend to have higher bias and lower variance, while complex models often have lower bias but higher variance.

Regularization: Techniques like L1/L2 regularization, dropout, or early stopping can help control model complexity and reduce overfitting by penalizing overly complex models.

Ensemble Methods: Using ensemble methods like bagging, boosting, or stacking can combine multiple models to reduce variance and improve overall performance.

Cross-validation: Properly using cross-validation techniques to evaluate models and tuning hyperparameters can help find the right balance between bias and variance.

Understanding and managing the bias-variance tradeoff is crucial in machine learning to build models that generalize well to new, unseen data while capturing the essential patterns present in the training data.

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

Ans. Sure, here's a brief description of each type of kernel used in Support Vector Machines (SVMs):

Linear Kernel:

The linear kernel is the simplest kernel used in SVMs.

It represents a linear decision boundary between classes in the original feature space.

It works well when the data is linearly separable or when the number of features is large compared to the number of samples.

The linear kernel calculates the inner product between feature vectors in the original space.

RBF (Radial Basis Function) Kernel:

The RBF kernel is a popular choice in SVMs and is suitable for non-linear classification.

It can map the input data into a high-dimensional space where it becomes linearly separable.

It is capable of capturing complex, non-linear relationships between features by using a Gaussian-like function. The RBF kernel has a parameter 'gamma' that determines the influence of individual training samples on the decision boundary.

Polynomial Kernel:

The polynomial kernel is used to handle non-linear classification by mapping data into a higher-dimensional space using polynomial functions.

It computes the similarity between two vectors by the degree of polynomial and a coefficient.

This kernel can capture more complex relationships than a linear kernel but might be sensitive to the choice of the polynomial degree.

The degree parameter in the polynomial kernel controls the degree of the polynomial used for mapping. Each kernel in SVMs has its strengths and weaknesses, and the choice of kernel depends on the nature of the data and the problem at hand. The linear kernel is efficient and works well for linearly separable data, while RBF and polynomial kernels are capable of handling non-linear relationships but might require careful tuning of their hyperparameters to achieve optimal performance.