Q1. 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?

The choice between R-squared and Residual Sum of Squares (RSS) as a measure of goodness of fit in regression depends on the context and the specific goals of the analysis. Each metric provides different insights into the performance of a regression model, and their interpretation can vary. If the primary goal is to assess the overall explanatory power of the model and compare the proportion of variability explained, R-squared is a suitable metric. It provides a standardized measure that is independent of the scale of the dependent variable. If the goal is to focus on the absolute fit of the model and compare models based on the magnitude of residuals, RSS can be useful. It directly measures the sum of squared errors and is interpretable in terms of the scale of the dependent variable. In practice, both metrics are often considered together to provide a comprehensive assessment of model performance. Ultimately, the choice between R-squared and RSS depends on the specific objectives and the aspects of model fit that are of primary interest in a given analysis.

Q2. 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.

In the context of linear regression, Total Sum of Squares (TSS), Explained Sum of Squares (ESS), and Residual Sum of Squares (RSS) are metrics used to assess the goodness of fit of the regression model. These terms are often used in the context of the coefficient of determination provides a measure of the proportion of variability in the dependent variable that is explained by the regression model. A higher value indicates a better fit of the model to the data.

Q3. What is the need of regularization in machine learning?

Regularization in machine learning is a technique used to prevent overfitting and improve the generalization performance of a model. Overfitting occurs when a model learns the training data too well, capturing noise and details specific to the training set but failing to generalize to new, unseen data. Regularization introduces constraints on the model parameters during training to mitigate overfitting and enhance the model's ability to generalize. Common regularization techniques include L1 regularization (Lasso), L2 regularization (Ridge), and a combination of both (Elastic Net) for linear models. Regularization is also applied to other machine learning algorithms, including neural networks and decision trees, with variations specific to each algorithm. Regularization is a crucial tool in machine learning to prevent overfitting, improve generalization, handle noise, and strike a balance between model complexity and simplicity. The choice of the regularization technique and its strength is often determined through experimentation and validation on independent test sets.

Q4. What is Gini–impurity index?

The Gini impurity index is a measure used in the context of decision trees to evaluate the impurity or disorder of a set of data points. It is commonly employed in binary classification problems to assess the homogeneity of a collection of samples with respect to their class labels. The Gini impurity is often used as a criterion for making decisions at each node of a decision tree during the tree-building process. The Gini impurity is a measure of the likelihood of misclassifying a randomly chosen element in the dataset. A lower Gini impurity indicates a purer set (closer to a single class), while a higher Gini impurity suggests a more mixed set. In the context of decision trees, the Gini impurity is used to evaluate the impurity of a dataset and guide the splitting of nodes during the construction of the tree. When choosing a feature to split on, the algorithm considers the reduction in Gini impurity that the split would achieve. The goal is to make splits that result in subsets with lower impurity, leading to more homogeneous child nodes. In summary, the Gini impurity is a criterion used in decision trees to evaluate the impurity of a set of data points. It is a measure of the likelihood of misclassification, and decision tree algorithms use it to make decisions about feature splits during the tree-building process.

Q5. Are unregularized decision-trees prone to overfitting? If yes, why?

Yes, unregularized decision trees are prone to overfitting. Decision trees are capable of capturing intricate details and patterns in the training data, often to the point of memorizing the training set rather than learning the underlying structure. This characteristic makes them highly flexible and able to fit complex relationships, but it also leaves them susceptible to overfitting. By applying appropriate regularization techniques, decision trees can be constrained to better generalize to new data, making them more robust and preventing overfitting.

Q6. What is an ensemble technique in machine learning?

An ensemble technique in machine learning involves combining the predictions of multiple individual models to create a more robust and accurate model. The idea behind ensemble learning is to leverage the collective wisdom of diverse models, each of which may have strengths and weaknesses, to improve overall predictive performance. It's important to note that the success of ensemble techniques depends on factors such as the diversity of the individual models, the quality of the base learners, and the method used to combine their predictions. Ensembles are a powerful and widely used approach in machine learning, providing a way to improve model performance across various tasks.

Q7. What is the difference between Bagging and Boosting techniques?

Bagging (Bootstrap Aggregating) and Boosting are both ensemble learning techniques that aim to improve the performance of machine learning models by combining the predictions of multiple weak learners. Despite their similar goals, these techniques differ in their approaches to training and aggregating weak learners. Common algorithms for bagging include Random Forests, while popular boosting algorithms include AdaBoost, Gradient Boosting (e.g., XGBoost, LightGBM), and others. bagging and boosting are both ensemble techniques that combine multiple weak learners, but they differ in their training processes, weighting of instances, and the way predictions are aggregated. Bagging aims to reduce variance, while boosting focuses on improving accuracy by emphasizing challenging instances.

Q8. What is out-of-bag error in random forests?

The out-of-bag (OOB) error is a concept associated with the training process of Random Forests, a popular ensemble learning algorithm. In a Random Forest, multiple decision trees are trained on subsets of the data, and the OOB error provides a way to estimate the performance of the model without the need for a separate validation set. The out-of-bag error acts as a cross-validation metric within the training process of the Random Forest, providing an unbiased estimate of the model's predictive performance without the need for a separate validation set. It is particularly useful when the available dataset is limited, and setting aside a validation set might reduce the amount of data used for training. The out-of-bag error is often used to monitor the training process, assess model performance, and make decisions about hyperparameters or model complexity in Random Forests. It helps in understanding how well the model is likely to generalize to new, unseen data.

Q9. What is K-fold cross-validation?

K-fold cross-validation is a widely used technique in machine learning for assessing the performance and generalization ability of a model. The main idea behind K-fold cross-validation is to split the dataset into

K subsets or "folds." The model is trained and evaluated K times, each time using a different fold as the validation set and the remaining K-1 folds as the training set. This process helps in obtaining a more robust performance estimate by averaging the results over K iterations.

Q10. What is hyper parameter tuning in machine learning and why it is done?

Hyperparameter tuning in machine learning involves optimizing the hyperparameters of a model to improve its performance. Hyperparameters are configuration settings external to the model itself and are not learned from the training data. Examples include learning rates, regularization parameters, the number of hidden layers in a neural network, and so on. The goal of hyperparameter tuning is to find the optimal combination of hyperparameters that results in the best model performance.

Hyperparameter tuning is a crucial step in the machine learning workflow. It helps to fine-tune models for better performance, improves generalization, and ensures that the chosen hyperparameters are well-suited to the specific characteristics of the data.

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

It's common practice to tune the learning rate during the training process or use adaptive learning rate algorithms. Techniques like learning rate annealing, momentum, and adaptive learning rate methods can help mitigate the challenges associated with selecting an appropriate learning rate. Experimenting with different learning rates and monitoring the training process can help find a balance between fast convergence and stability.

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

While Logistic Regression is a powerful tool for linear classification problems, it may not perform well on inherently non-linear data. If faced with non-linear data, it's often more appropriate to explore models designed to handle non-linearities or consider transforming the features to capture the non-linear relationships effectively.

Q13. Differentiate between Adaboost and Gradient Boosting.

While both Adaboost and Gradient Boosting are ensemble methods that sequentially build a series of weak learners, they differ in their approach to adjusting instance weights, handling residuals, and the choice of weak learners. Adaboost adjusts instance weights to focus on misclassified instances, while Gradient Boosting fits weak learners to the residuals of the combined model to minimize errors.

Q14. What is bias-variance trade off in machine learning?

The tradeoff arises because reducing bias often increases variance and vice versa. A model with high bias and low variance might not fit the training data well, but it might generalize better to new data. Conversely, a model with low bias and high variance might fit the training data very well, but it may not generalize well to new, unseen data. The goal is to find the right level of model complexity that minimizes both bias and variance, striking a balance that results in good generalization performance.

Q15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Linear Kernel: The linear kernel is the simplest form of SVM kernel. It computes the dot product of the feature vectors in the original input space. This kernel is effective when the data is linearly separable, meaning it can be separated by a straight line in the input space.

RBF (Radial Basis Function) Kernel: The RBF kernel is a popular non-linear kernel in SVM. It is also known as the Gaussian kernel. It measures the similarity between data points in a high-dimensional space by transforming the features into an infinite-dimensional space. The RBF kernel is effective in capturing complex, non-linear relationships in the data.

Polynomial Kernel: The polynomial kernel is another non-linear kernel used in SVM. It calculates the similarity between data points by raising the dot product of the feature vectors to a certain power. The degree of the polynomial determines the complexity of the decision boundary. This kernel is useful when the relationship between classes is polynomial and not linear.