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?**

Answer - R-squared is generally considered a better measure of goodness of fit in regression compared to the Residual Sum of Squares (RSS).

1- Interpretability - R-squared provides an intuitive measure of the proportion of variance explained by the regression model.

2 - Adjusted for Sample Size - R-squared is adjusted for the number of predictors in the model, which helps in comparing models with different numbers of predictors

3 - Captures Explained Variation - R-squared measures the proportion of the total variation in the dependent variable that is explained by the independent variables in the model.

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.**

Answer - Total Sum of Squares (TSS) are represents the total variation in the dependent variable (y) explained by the model and It is calculated as the sum of the squared differences between each observed y-value and the mean of all y-values.

Explained Sum of Squares (ESS) are representing the variation in the dependent variable (y) explained by the regression model. It is calculated as the sum of the squared differences between the predicted y-values (based on the regression model) and the mean of all values.

Residual Sum of Squares (RSS) are representing the unexplained variation in the dependent variable (y) after accounting for the effects of the independent variables. It is calculated as the sum of the squared differences between the observed y-values and the predicted values from the regression model.

Formula - TSS=ESS+RSS

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

Answer - While training a machine learning model, the model can easily be overfitted 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 overfitting and help us get an optimal model.

The need for regularization in machine learning arises due to several reasons:

1- Preventing Overfitting
2- Handling High-Dimensional Data
3- Improving Model Stability
4- Balancing Bias and Variance
5- Enhancing Model Interpretability

4 - **What is Gini–impurity index?**

Answer - The Gini impurity index, also called Gini impurity, is a way to measure how mixed up or uncertain a group of categories is. It's often used in decision-making computer programs, especially when sorting things into two groups, to figure out the best way to split them up.

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

Answer - Yes, decision trees without constraints are likely to overfit, especially when they're allowed to grow without limits. This happens because decision trees can learn very detailed patterns from the training data, even including irrelevant noise and outliers. So, the tree might become too complicated, perfectly fitting the training data but also capturing random fluctuations and not just the main patterns.

The main reasons why unregularized decision trees are prone to overfitting

1. **High Variance**: Decision trees have a high variance, meaning they are sensitive to small fluctuations in the training data. When allowed to grow without any constraints, decision trees can become highly specialized to the training data, resulting in different splits and decisions for similar instances.

2. **Complexity**: Unregularized decision trees can grow to be very deep and complex, with many nodes and branches. This complexity allows the tree to capture intricate patterns in the training data, but it also increases the likelihood of overfitting, as the tree may memorize the training data instead of learning generalizable patterns.

3. **Overly Specific Rules**: Decision trees can create overly specific rules to classify instances in the training data, even if those rules do not generalize well to new, unseen data. These specific rules may be based on noise or outliers in the training data, leading to poor performance on test data.

6 - **What is an ensemble technique in machine learning?**

Answer - An ensemble technique in machine learning is a way of combining several individual models to make a better and more reliable prediction. Instead of just using one model, ensemble techniques use a variety of models to improve accuracy and prevent making predictions that are too specific to the training data. There are five commonly used ensemble techniques in machine learning.

– Bagging (Bootstrap Aggregating)

1 – Boosting

2 – Random forest

3 – Stacking

4 – Voting

7 - **What is the difference between Bagging and Boosting techniques?**

Answer - Bagging (Bootstrap Aggregating) and Boosting are both ensemble techniques used in machine learning to improve the performance of predictive models by combining the predictions of multiple base models.

A. **Training Process**:

Bagging: In bagging, multiple instances of the same base model are trained independently on different subsets of the training data, sampled with replacement (bootstrap sampling). Each base model is trained in parallel, and there is no interaction between them during training.

Boosting: In boosting, base models are trained sequentially, where each subsequent model focuses on correcting the errors made by the previous ones. The training process is iterative, and instances in the training data are weighted based on their performance in previous iterations. Examples of boosting algorithms include AdaBoost, Gradient Boosting Machines and XGBoost.

B. **Base Model Weighting**:

Bagging: In bagging, the predictions of individual base models are combined by averaging (for regression) or voting (for classification). Each base model has an equal weight in the final prediction.

Boosting: In boosting, the predictions of individual base models are combined by weighted averaging, where each model's contribution to the final prediction is weighted based on its performance. Models that perform well are given higher weights, while models that perform poorly are given lower weights.

C. **Reduction of Variance vs. Bias**:

Bagging: Bagging aims to reduce variance by averaging the predictions of multiple models trained on different subsets of the data. It helps reduce overfitting and improves stability.

Boosting: Boosting aims to reduce bias by sequentially training weak learners that focus on correcting the errors made by previous models. It helps improve model accuracy and reduce underfitting.

D. **Parallelism**:

Bagging: Bagging can be parallelized, as each base model is trained independently on a subset of the data. This allows for efficient distributed training.

Boosting: Boosting is inherently sequential, as each subsequent model depends on the performance of the previous ones. This makes boosting less suitable for parallelization compared to bagging.

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

Answer - In Random Forests, the out-of-bag error helps estimate how well the model might perform on new data without needing cross-validation. It's figured out using the data points not included in the samples used to train each decision tree in the forest. There are four types of out-of-bag error estimation methods in Random Forests.

1 - Bootstrap Sampling
2 - Out-of-Bag Samples
3 - Estimating Error
4 - Aggregating Errors

9 - **What is K-fold cross-validation?**

Answer - K-fold cross-validation is a common method in machine learning. It helps to check how well a model can predict things. We split the data into K equal parts, and then train and test the model K times. Each time, one part is for testing and the others are for training. There are three types of K-fold cross-validation.

1 Dataset Splitting
2 Model Training and Testing
3 Performance Evaluation

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

Answer - Hyperparameter tuning, also known as hyperparameter optimization, is similar to finding the best settings for a machine learning model. Hyperparameters are settings that control how the model learns, such as its complexity or learning speed. We need to determine these settings before the model begins learning. Unlike other settings that the model learns from data, we must choose hyperparameters ourselves and cannot adjust them while the model is learning.

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

Answer - If we use a big learning rate in gradient descent, it can cause problems that might stop the optimization process from getting to the best solution or even prevent it from working properly.

There are five large learning rates.

1- Overshooting the Minimum
2- Instability and Unpredictability
3- Difficulty in Convergence
4- Skipping the Optimal Solution
5- Sensitivity to Initial Conditions

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

Answer – Logistic Regression is like drawing a straight line to separate things into different groups. It's good when the groups can be separated easily with a straight line. But when things are more complicated and don't follow a straight line, logistic regression might not work well.

In those cases, we use other methods like decision trees, random forests, support vector machines, or neural networks. These methods can handle more complex situations where the relationship between things isn't straightforward.

If we still want to use logistic regression with tricky data, we can try changing the data or adding more information to make it easier for logistic regression to understand. We might also transform the data to look different, so logistic regression can work better with it.

13- Differentiate between Adaboost and Gradient Boosting

Answer - AdaBoost and Gradient Boosting are both ensemble learning techniques used to improve the performance of weak learners, typically decision trees, by combining multiple models. While they share similarities in their boosting approach, they differ in several key.

Algorithm – AdaBoost is an adaptive boosting algorithm that sequentially trains a series of weak learners. Each weak learner is trained on a weighted version of the training data, with the weights adjusted to focus more on instances that were misclassified by previous models. And Gradient Boosting is a general boosting algorithm that builds a series of weak learners in a stage-wise manner. Instead of adjusting the weights of training instances, Gradient Boosting fits each weak learner to the residuals of the previous model.

Objective Function - AdaBoost minimizes the exponential loss function, which penalizes misclassifications exponentially. It focuses on reducing the training error by giving more weight to misclassified instances in subsequent iterations. And Gradient Boosting minimizes a user-specified loss function, such as mean squared error (MSE) for regression tasks or log loss for classification tasks.

Model Training - AdaBoost trains weak learners sequentially, with each subsequent model focusing on the misclassified instances of the previous models. And Gradient Boosting trains weak learners sequentially, with each subsequent model fitting the residuals of the previous models.

Learning Rate - AdaBoost uses a learning rate parameter to control the contribution of each weak learner to the final prediction. A smaller learning rate leads to slower learning but can improve generalization. And Gradient Boosting also uses a learning rate parameter, but its role is different. It controls the step size of each iteration and helps prevent overfitting. A smaller learning rate typically leads to better performance but requires more iterations.

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

Answer - The bias-variance trade-off is an important idea in machine learning. It's about finding the right balance between two things: bias and variance in a model.

Bias is about the mistakes made when a model is too simple. It can't handle the complexity of real-world problems, so it misses important patterns in the data. High bias means the model doesn't fit the data well, which leads to poor performance on both the training and test sets. This is called underfitting.

Variance, on the other hand, is about the model being too sensitive to small changes in the training data. If the model is too complex, it might pick up on random noise in the data and mistake it for important information. This leads to overfitting, where the model works well on the training data but fails when given new data.

So, the bias-variance trade-off is all about finding the sweet spot between a model that's too simple and one that's too complex.

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

Answer - **Linear Kernel** - The linear kernel is the simplest kernel function used in SVMs.

It computes the dot product between the feature vectors in the original feature space.

The decision boundary generated by the linear kernel is a straight line in the input space.

It is suitable for linearly separable datasets or datasets where a linear decision boundary is appropriate.

RBF (Radial Basis Function) Kernel - The RBF kernel is a popular choice for SVMs, especially when dealing with non-linearly separable data.

It maps the input features into a higher-dimensional space using a Gaussian (radial basis) function.

The decision boundary generated by the RBF kernel is non-linear and can capture complex relationships in the data.

It is effective for handling non-linear decision boundaries and can adapt to various types of data distributions.

Polynomial Kernel - The polynomial kernel is another kernel function used in SVMs to handle nonlinearly separable data.

It computes the dot product between the feature vectors in a higher-dimensional space defined by polynomial terms.

The decision boundary generated by the polynomial kernel is polynomial-shaped and can capture non-linear relationships in the data.

The degree parameter of the polynomial kernel determines the degree of the polynomial used in the kernel function. Higher degrees allow the kernel to capture more complex patterns but may also lead to overfitting.