MACHINE LEARNING

Q1 to Q15 are subjective answer type questions, Answer them briefly.

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 - R-squared is a better measure of goodness of fit in regression compared to the Residual Sum of Squares.
   Interpretability: It ranges from 0 to 1, where 1 indicates a perfect fit. This makes it easier to interpret compared to RSS, which is an absolute measure and doesn't provide a clear indication of the model's power.
   Normalization: It is scale-invariant and allows for comparison across different datasets and models. RSS, on the other hand, depends on the scale of the dependent variable and is not comparable across models or datasets without normalization.
   Context: It is widely recognized and understood by practitioners and researchers in the field, making it easier to communicate the performance of the model.
   Model Comparison: It allows for straightforward comparison between different models. A higher R-squared value indicates a better fit to the data, while a lower value suggests that the model may not be capturing as much of the variance in the dependent variable.

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 (Total Sum of Squares), ESS (Explained Sum of Squares), and RSS (Residual Sum of Squares) are important metrics used to assess the goodness of fit of a regression model.

   Total Sum of Squares (TSS): TSS is calculated as the sum of the squared differences between each observed value of Y and the mean of Y

   $$\textbf{TSS} = \sum^{n}_{i=1} (y_i - \bar{y})^2$$

   Explained Sum of Squares (ESS): ESS is calculated as the sum of the squared differences between the predicted values of Y and the mean of Y

   $$\textbf{ESS} = \sum^{n}_{i=1} (\hat{Y}_i - \bar{Y})^2$$

   Residual Sum of Squares (RSS): RSS is calculated as the sum of the squared differences between the observed values of Y and the predicted values of Y:

   $$\textbf{RSS} = \sum^{n}_{i=1} (Y_i - \hat{Y}_i)^2$$

   The relationship between TSS, ESS, and RSS can be expressed by

   $$TSS = ESS + RSS$$

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

   Ans –
   Preventing Overfitting: One of the primary motivations for regularization is to prevent overfitting. Regularization techniques add constraints to the model, limiting its complexity and reducing the risk of overfitting.
   Handling Multicollinearity: n regression analysis, multicollinearity refers to the presence of strong correlations between predictor variables. This can lead to unstable parameter estimates and inflated standard errors.
   Improving Model Generalization: By imposing constraints on the model parameters, regularization techniques promote simpler models that are less sensitive to fluctuations in the training data and more likely to generalize effectively to new observations.
   Feature Selection and Model Interpretability: This can improve model interpretability by identifying the most influential predictors and reducing the dimensionality of the feature space.
   Reducing Model Variance: By constraining the magnitude of the coefficients, regularization prevents large fluctuations in the model's output due to small changes in the training data.

4. What is Gini–impurity index?

Ans - The Gini impurity index is a measure of the impurity or randomness of a dataset, commonly used in decision tree algorithms, particularly for binary classification problems.
For a given node in a decision tree, which represents a subset of the dataset:
Calculate the probability pi of an instance belonging to a particular class i.

$$G=1-\sum^{k}_{i=1}{}_{1k}p_i^2$$

If a node is perfectly pure (i.e., all instances belong to the same class), the Gini impurity is 0.
If a node is completely impure (i.e., instances are evenly distributed across all classes), the Gini impurity is at its maximum value, which occurs when all pi values are equal.

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

Ans – Yes, the reasons why unregularized decision trees can lead to overfitting:

High Variance: Decision trees have high variance, meaning they are highly sensitive to small variations in the training data.
Memorization of Training Data: Unregularized decision trees have the capacity to memorize the training data by creating highly specific decision rules for each instance.
Overly Complex Trees: Decision trees can grow to be very deep and complex, especially when they are allowed to expand without constraints.
Lack of Pruning: Unregularized decision trees typically do not employ pruning techniques to trim back the tree after it has been grown. Pruning helps prevent overfitting by removing branches of the tree that do not contribute significantly to its predictive power. Without pruning, decision trees may continue to grow excessively, resulting in overfitting.

6. What is an ensemble technique in machine learning?

Ans - An ensemble technique in machine learning is a method that combines the predictions of multiple individual models to produce a final prediction.

Ensemble techniques can be broadly categorized into two types:

Bagging (Bootstrap Aggregating): Bagging involves training multiple instances of the same base learning algorithm on different subsets of the training data.
Boosting: Boosting is an iterative ensemble technique that builds a sequence of weak learners (models that perform slightly better than random guessing) and combines them to create a strong learner.

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

Ans - **Model Combination: Bagging**: In bagging, the final prediction is obtained by averaging the predictions of all individual models (for regression tasks) or by taking a majority vote (for classification tasks).
Boosting: In boosting, the final prediction is obtained by combining the predictions of all weak learners, often using a weighted sum. The weights assigned to each weak learner depend on its performance on the training data

Handling of Errors: **Bagging**: Bagging methods do not explicitly focus on correcting errors made by individual models. Instead, they rely on the collective wisdom of diverse models to improve predictive performance.
Boosting: Boosting methods train each subsequent model to correct the errors made by the previous models. This iterative process helps reduce bias and improve overall model performance.

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

Ans - In Random Forests, the out-of-bag (OOB) error is an estimation of the generalization error of the model without the need for an explicit validation set. It is a measure of how well the Random Forest model performs on unseen data.

9. What is K-fold cross-validation?
Ans - K-fold cross-validation is a popular technique used in machine learning for model evaluation and selection, particularly when the dataset is limited and splitting it into separate training and validation sets might lead to overfitting or underfitting.

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

Ans - Hyperparameter tuning, also known as hyperparameter optimization, is the process of selecting the optimal hyperparameters for a machine learning model. Hyperparameters are parameters that are set before the learning process begins and control the behavior of the learning algorithm.
Model Performance Improvement: The choice of hyperparameters can have a significant impact on the performance of a machine learning model.
Avoiding Overfitting and Underfitting: hperparameters play a crucial role in controlling the complexity of the model and preventing overfitting or underfitting.
Adapting to Different Datasets and Tasks: Hyperparameters often need to be adapted to the specific characteristics of the dataset and the task at hand.
Optimizing Computational Resources: Hyperparameter tuning helps optimize the allocation of computational resources by identifying the most effective hyperparameter configurations within a given time or budget constraint.

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

Ans - If a large learning rate is used in gradient descent, several issues can arise, impacting the convergence and stability of the optimization process. Here are some of the issues associated with a large learning rate:

Overshooting the Minimum: A large learning rate can cause the optimization algorithm to take excessively large steps in the direction of the gradient
Instability and Unpredictability: Large learning rates can introduce instability and unpredictability in the optimization process.
Failure to Converge: With a large learning rate, the optimization algorithm may fail to converge to the minimum of the loss function.
Poor Generalization: In machine learning tasks, the goal is often to minimize the loss function on unseen data (generalization).

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

Ans - Logistic Regression is a linear classification algorithm, meaning it models the relationship between the input features and the output (binary or multi-class labels) using a linear function. While Logistic Regression is effective for linearly separable data, it may not perform well when the decision boundary between classes is non-linear.

Linear Decision Boundary: Logistic Regression models the decision boundary as a linear combination of the input features. This means the decision boundary in logistic regression is a hyperplane in the feature space.
Underfitting: When Logistic Regression is applied to non-linear data, it may result in underfitting, where the model fails to capture the underlying patterns in the data.
Limited Expressiveness: Logistic Regression has limited expressiveness compared to non-linear classifiers such as decision trees, support vector machines with non-linear kernels, or neural networks. These models can capture more complex relationships between the input features and the output by employing non-linear transformations or by using more flexible architectures.

13. Differentiate between Adaboost and Gradient Boosting.

Ans - **AdaBoost**: AdaBoost combines multiple weak learners (models that perform slightly better than random guessing) to create a strong learner. The weak learners are typically shallow decision trees, often referred to as "stumps," which are simple and have limited depth.
Gradient Boosting: Gradient Boosting builds a sequence of weak learners in a stage-wise fashion, where each new weak learner is trained to correct the errors made by the previous models. The weak learners can be any model capable of regression or classification, commonly decision trees.
AdaBoost: AdaBoost is simpler and more interpretable compared to Gradient Boosting. It consists of a sequence of weak learners trained sequentially, with each learner focusing on correcting the errors made by the previous ones.
Gradient Boosting: Gradient Boosting is more complex and computationally intensive compared to AdaBoost, as it involves optimizing a differentiable loss function using gradient descent.

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

Ans - he bias-variance tradeoff is a fundamental concept in machine learning that describes the relationship between the bias of a model and its variance, and how they influence the model's predictive performance.
High Bias, Low Variance: Models with high bias tend to be simple and make strong assumptions about the data. They have low variance because they produce similar predictions across different datasets or subsets of the training data. However, high bias models may fail to capture the underlying patterns in the data, resulting in underfitting.

Low Bias, High Variance: Models with low bias are more complex and flexible, allowing them to capture subtle patterns and relationships in the data. They have high variance because they are sensitive to small fluctuations in the training data. While low bias models may perform well on the training data, they may also overfit and generalize poorly to unseen data.

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

Ans - **Linear Kernel**: The linear kernel is the simplest kernel function used in SVM. It computes the dot product between the feature vectors of the input samples.
RBF (Radial Basis Function) Kernel: The RBF kernel is a popular choice for SVMs and is suitable for non-linearly separable datasets. It computes the similarity between samples in a high-dimensional space using the Gaussian (radial basis) function.
Polynomial Kernel: The polynomial kernel is used to handle non-linearly separable data by mapping the original feature space into a higher-dimensional space. It computes the similarity between samples using polynomial functions of the original features. The degree of the polynomial (specified by the degree parameter) controls the complexity of the decision boundary.