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 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It is bounded between 0 and 1, where 1 indicates a perfect fit. This makes it easier to interpret compared to RSS, which is just a measure of the sum of squared residuals without any standardized scale. while RSS is a component of the calculation of R-squared and is important in understanding the distribution of residuals, R-squared provides a more comprehensive measure of goodness of fit in regression models.

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, Total Sum of Squares (TSS), Explained Sum of Squares (ESS), and Residual Sum of Squares (RSS) are important concepts used to understand the variability in the data and the goodness of fit of the regression model. Here's a brief explanation of each along with the equation relating them:

1. Total Sum of Squares (TSS):
TSS represents the total variance in the dependent variable (Y) and measures how much the dependent variable varies around its mean. It is calculated by summing the squared differences between each observed dependent variable value and the mean of the dependent variable.

2. Explained Sum of Squares (ESS):
ESS represents the variance in the dependent variable (Y) that is explained by the regression model. It measures how much of the total variance in the dependent variable is accounted for by the independent variables in the model.

3. Residual Sum of Squares (RSS):
RSS represents the unexplained variance in the dependent variable (Y) that remains after fitting the regression model. It measures the squared differences between the observed dependent variable values and the predicted values (residuals) from the regression model.

The relationship between these three metrics can be expressed by the following equation:

TSS = ESS + RSS

This equation states that the total variability in the dependent variable (TSS) can be decomposed into two parts: the variability explained by the regression model (ESS) and the unexplained residual variability (RSS). In a well-fitting model, ESS would be large relative to RSS, indicating that much of the variability in the dependent variable is accounted for by the model.

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

ANS= Regularization is a technique used in machine learning to prevent overfitting and improve the generalization performance of a model. Here's why regularization is needed:

1. Preventing Overfitting: Overfitting occurs when a model learns to capture noise and patterns specific to the training data, rather than the underlying true relationship between the features and the target variable. Regularization helps to mitigate overfitting by adding a penalty term to the loss function, discouraging the model from fitting the training data too closely.

2. Improving Generalization: Regularization techniques encourage the model to learn simpler patterns that generalize well to unseen data. By penalizing overly complex models, regularization helps to ensure that the model's performance on new, unseen data is similar to its performance on the training data.

3. Handling High-Dimensional Data: In high-dimensional spaces, where the number of features is large relative to

the number of samples, overfitting becomes a significant concern. Regularization methods like L1 (Lasso) and L2 (Ridge) regularization can help in feature selection and reduce the complexity of the model by shrinking the coefficients associated with less informative features.

4. Dealing with Multicollinearity: Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated. This can lead to unstable parameter estimates and difficulties in interpreting the model. Regularization techniques can help to mitigate the effects of multicollinearity by imposing constraints on the parameter estimates.

5. Enhancing Model Interpretability: Regularization can lead to more interpretable models by encouraging sparsity (L1 regularization) or by shrinking the coefficients of less important features (L2 regularization). This can help practitioners better understand the relationships between the features and the target variable.

Overall, regularization is an essential tool in the machine learning toolbox for building models that generalize well to new data, are robust to noise and outliers, and are interpretable. It helps strike a balance between model complexity and generalization performance, leading to more reliable and effective models.

4. What is Gini–impurity index?

ANS= Gini impurity measures the probability of incorrectly classifying a randomly chosen element in a dataset if it were randomly labeled according to the class distribution in the dataset. In other words, it quantifies the degree of uniformity or heterogeneity in the dataset. Gini impurity index is a measure of impurity or randomness commonly used in decision tree algorithms to determine the optimal splits for partitioning datasets in classification tasks. It helps build decision trees that effectively separate the data into homogeneous subsets with respect to the target variable.

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

ANS= Yes, unregularized decision trees are prone to overfitting. Overfitting occurs when a model learns to fit the training data too closely, capturing noise or random fluctuations in the data rather than the underlying patterns. Decision trees are particularly susceptible to overfitting due to their inherent ability to create complex, deep tree structures that can perfectly fit the training data. To mitigate overfitting in decision trees, various regularization techniques can be employed, such as limiting the maximum depth of the tree, setting a minimum number of samples required to split a node, or pruning the tree after it has been fully grown. These techniques help simplify the model and encourage it to capture the most important patterns in the data, rather than fitting the noise. Regularized decision trees, such as Random Forests or Gradient Boosted Trees, incorporate additional techniques to combat overfitting and often yield better generalization performance compared to unregularized decision trees.

6. What is an ensemble technique in machine learning?

ANS= An ensemble technique in machine learning involves combining multiple individual models to create a stronger, more robust predictive model. The basic idea behind ensemble methods is to leverage the wisdom of the crowd: by aggregating the predictions of multiple models, the ensemble can often outperform any of its individual components.
There are several popular ensemble techniques, including:
1.BAGGING(BOOTSTRAP AGGREGATING
2.BOOSTING
3.RANDOM FOREST
Ensemble methods are widely used in practice because they tend to produce more accurate and robust models compared to individual models. They are particularly effective when the base models are diverse (i.e., they make different kinds of errors) and when there is a sufficient amount of data available for training each model.

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

ANS= Bagging (Bootstrap Aggregating) and Boosting are both ensemble techniques used in machine learning to improve the performance of models, but they differ in their approach to combining multiple models and in the way they handle the training process. It use to improve model performance, they differ in their training processes, handling of errors, and approaches to reducing overfitting. Bagging focuses on training diverse models independently and averaging their predictions, while Boosting sequentially trains models to correct the errors made by previous ones.
BAGGING- In bagging, multiple instances of the same base model are trained independently on different subsets of the training data, typically sampled with replacement (bootstrap sampling). Each model learns from a slightly

different perspective of the data. The final prediction is obtained by averaging (for regression) or voting (for classification) the predictions of all models.
BOOSTING- In boosting, models are trained sequentially, and each subsequent model focuses on correcting the errors made by the previous ones. Initially, each data point is given equal weight, but subsequent models pay more attention to the data points that were misclassified by earlier models. The final prediction is a weighted sum of the predictions made by each model.

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

ANS= The out-of-bag (OOB) error in Random Forests is a method used to estimate the performance of the model without the need for an additional validation set. It's a unique feature of Random Forests and is a consequence of the bootstrapping technique used during training. In Random Forests, each decision tree in the ensemble is trained on a bootstrapped sample of the original dataset, which means some observations are left out of each bootstrapped sample. The out-of-bag error is calculated by evaluating each observation using only the trees that did not include that observation in their respective bootstrap samples.

9. What is K-fold cross-validation?

ANS= K-fold cross-validation is a technique used to evaluate the performance of a machine learning model by partitioning the original dataset into K equal-sized subsets (or "folds"). The model is then trained K times, each time using K-1 folds as training data and the remaining fold as validation data. K-fold cross-validation provides a more robust estimate of the model's performance compared to a single train-test split, as it uses multiple splits of the data for training and validation. It helps to reduce the variance in the performance estimate and provides a more accurate assessment of how well the model is likely to generalize to unseen data. Common choices for the value of K include 5-fold and 10-fold cross-validation, but other values can be used depending on the size of the dataset and computational constraints. Additionally, techniques such as stratified K-fold cross-validation can be used to ensure that each fold preserves the class distribution of the original dataset, which is particularly useful for imbalanced datasets.

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 set of hyperparameters for a machine learning model. Hyperparameters are parameters that are set before the learning process begins and are not learned from the data during training. They control the behavior and complexity of the model and can significantly impact its performance. Hyperparameter tuning is done to find the combination of hyperparameters that results in the best performance of the model on unseen data. The goal is to optimize the model's performance metrics, such as accuracy, precision, recall, F1-score, or any other relevant metric, on a validation set or through cross-validation. Hyperparameter tuning can be performed using various techniques, including grid search, random search, Bayesian optimization, and more advanced methods like genetic algorithms or evolutionary strategies. The choice of method depends on factors such as the size of the search space, computational resources, and the desired level of optimization.

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, hindering the convergence of the optimization process and potentially leading to poor performance of the model. Here are some of the issues associated with using a large learning rate:
1. Overshooting the Minimum
2. Instability and Oscillations
3. Failure to Converge
4. Inaccurate Parameter Updates
5. Unstable Gradient

To mitigate these issues, it is crucial to carefully choose an appropriate learning rate for gradient descent. Techniques such as learning rate schedules, adaptive learning rates (e.g., Adam, RMSprop), and gradient clipping can help stabilize the optimization process and improve convergence. Additionally, conducting hyperparameter tuning to find the optimal learning rate for a specific problem can significantly enhance the performance of the optimization algorithm.

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

ANS= Logistic Regression is a linear classification algorithm that is specifically designed for binary classification tasks, although it can be extended to handle multi-class classification as well. Logistic Regression models the

relationship between the independent variables (features) and the probability of belonging to a particular class using a linear function followed by a non-linear transformation (sigmoid function) to ensure that the predicted probabilities are between 0 and 1. While Logistic Regression can effectively model linear decision boundaries and is suitable for problems with linearly separable classes, it may not perform well on datasets with non-linear relationships between the features and the target variable. To handle non-linear data, more complex models such as Support Vector Machines with non-linear kernels, Decision Trees, Random Forests, Gradient Boosting Machines, Neural Networks, and Kernel Logistic Regression can be used. These models are capable of capturing non-linear relationships between the features and the target variable and can provide better performance on datasets with complex, non-linear patterns.

13. Differentiate between Adaboost and Gradient Boosting.

ANS= Adaboost and Gradient Boosting are both ensemble learning techniques used for improving the performance of machine learning models, particularly in the context of boosting.
Adaboost: Adaboost (short for Adaptive Boosting) typically uses a sequence of weak learners (e.g., decision trees with limited depth or stumps, which are single-level decision trees) as base learners. Each weak learner is trained sequentially, and the subsequent learners focus on correcting the errors made by the previous ones.
Gradient Boosting: Gradient Boosting, on the other hand, can use a variety of base learners, but decision trees are commonly employed. However, unlike Adaboost, the decision trees in Gradient Boosting are typically deeper (i.e., they are not constrained to be weak learners). Each decision tree is trained sequentially to correct the residuals (errors) made by the previous trees.

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

ANS= The bias-variance tradeoff is a fundamental concept in machine learning that refers to the balance between two types of error that affect the performance of a predictive model: bias and variance. Understanding this tradeoff is crucial for designing models that generalize well to unseen data.
Bias: Bias measures how closely the predictions of a model match the true values. A model with high bias tends to make simplistic assumptions about the underlying relationship between the features and the target variable, resulting in systematic errors. High bias can lead to underfitting, where the model fails to capture the true structure of the data and performs poorly on both the training and test sets.
Variance: Variance measures the variability of the model's predictions across different training sets. A model with high variance is overly sensitive to small fluctuations in the training data and captures noise rather than the underlying patterns. High variance can lead to overfitting, where the model learns to fit the training data too closely, resulting in excellent performance on the training set but poor generalization to new, unseen data.
The goal in machine learning is to find the right balance between bias and variance that minimizes the model's total error on unseen data. This often involves techniques such as cross-validation, regularization, and model selection to strike the appropriate balance and build models that generalize well to new data.

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

ANS= **Linear Kernel**: The linear kernel is the simplest kernel used in SVMs. It computes the dot product between the feature vectors in the original input space. The linear kernel is suitable for linearly separable datasets or datasets where the classes can be separated by a linear decision boundary. It computes the dot product between the feature vectors in the original input space.
RBF (Radial Basis Function) Kernel: The RBF kernel is a popular choice for SVMs and is capable of capturing non-linear decision boundaries. It measures the similarity between two feature vectors in a high-dimensional space using the Gaussian (radial basis) function. The RBF kernel can map the input space to an infinite-dimensional feature space, allowing it to capture complex non-linear relationships between the features and the target variable. The RBF kernel has two hyperparameters: gamma (γ), which controls the width of the Gaussian function, and C, which controls the trade-off between maximizing the margin and minimizing the classification error.
Polynomial Kernel: The polynomial kernel has a hyperparameter, degree, which determines the degree of the polynomial function used to compute the similarity between feature vectors. The polynomial kernel is another kernel function used in SVMs to capture non-linear relationships. It computes the similarity between two feature vectors in a high-dimensional space using polynomial functions.