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?**

   Answer:-R-squared and Residual Sum of Squares (RSS) are both commonly used metrics to assess the goodness of fit of a regression model, but they measure different aspects of the model's performance.

   R-squared :

   R-squared measures the proportion of the variance in the dependent variable (y) that is explained by the independent variables (x) in the model. It ranges from 0 to 1, where a value closer to 1 indicates that a larger proportion of the variance in the dependent variable is explained by the independent variables, and a value closer to 0 indicates poor model fit.

   Interpretation: R-squared is a relative measure of goodness of fit. A higher R-squared value suggests that the model provides a better fit to the data compared to a model with a lower R-squared value.

   Advantages: R-squared is easy to interpret and widely used. It provides insight into the overall explanatory power of the model.

   Residual Sum of Squares (RSS):

   RSS measures the total squared difference between the observed values of the dependent variable and the predicted values from the regression model. It represents the sum of the squared residuals, where residuals are the differences between the observed and predicted values.

   Interpretation: RSS is an absolute measure of model fit. Lower values of RSS indicate that the model's predictions are closer to the observed values, implying a better fit.

   Advantages: RSS provides a direct measure of the model's prediction accuracy. It is particularly useful for comparing different models or assessing improvements in model fit after model refinement.

---

2. **What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sumof Squares) in regression. Also mention the equation relating these three metrics with each other.**

Answer:-In the regression analysis, TSS (Total Sum of Squares), ESS(Explained Sum of Squares) and RSS (Residual Sum of Squares) are important metrics used to evaluate the goodness of fit of regression model and to understand the variability of the data.

1. Total Sum of Squares:- The total sum of square it is also know as the regression sum of square.TSS represents the total variability in the dependent variable (Y). It is calculated as the sum of the squared differences between the actual values of the dependent variable and the mean of the dependent variable .The formula of Total Sum of Squares is

TSS(Total Sum Of Squares)= $\Sigma (yi - y)^2$

- yi = actual values of the dependent variable
- y = mean of the dependent variable

2. Explained Sum of Squares:-ESS represents the portion of the total variation that is explained by the regression model, indicating how well the independent variable(s) in the model explain the variability in the dependent variable.The Formula of ESS is

ESS(Explained Sum of Squares)=$\Sigma (\hat{y}i - y)2$

$\hat{y}i$= predicted values of the dependent variable from the regression model.

3. Residual Sum of Squares(RSS):-*RSS* represents the portion of the total variation that is not explained by the regression model, often attributed to random error or other factors not captured by the model.

1. $RSS=\sum(Yi-Y\hat{}i)2$

Yi =actual values of the dependent variable, and

$Y\hat{}i$ =predicted values of the dependent variable from the regression model.

The equation relating these three metrics is:

$TSS=ESS+RSS$

This equation illustrates that the total variability in the dependent variable (TSS) can be decomposed into the variability explained by the regression model (ESS) and the unexplained residual variability (RSS).

---

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

Answer:-Regularization in machine learning used to prevent overfitting and it is a crucial technique in machine leaning that helps improve the stability, performance and generalization ability of models especially in situation where the training data is limited or noisy .Overfitting arises when a model excessively fits the nuances and noise within the training data, thereby capturing irrelevant patterns that fail to translate effectively to unseen data .There are various reasons why regularization is important in machine learning :

1. **Preventing Overfitting:-**Regularization helps prevent overfitting by penalizing complex models. It discourages models from fitting the noise or irrelevant patterns in the training data, promoting a more generalized representation.
2. **Improving Generalization:-**The main objective of machine learning models is to effectively generalize to fresh, unobserved data. Regularization helps prevent the model from becoming overly tailored to the training data, thereby enhancing its ability to provide accurate predictions for novel instances
3. **Increasing Stability:-**Regularization can increase the stability of the learning algorithm by reducing the sensitivity of the model to small changes in the training data. This can lead to more robust and reliable models that are less likely to be influenced by outliers or small variations in the data
4. **Handling Noisy Data:**-Regularization enhances the stability of the learning algorithm by decreasing the model's susceptibility to minor fluctuations in the training data. Consequently, this fosters the development of more resilient and dependable models that are less prone to being swayed by outliers or subtle deviations in the data. X

---

4. What is Gini–impurity index?

Answer:-The Gini impurity index also called Gini impurity, quantifies the likelihood of misclassifying a randomly selected element from a set if it were randomly labeled based on the distribution of labels within that subset.It is commonly used in decision tree algorithms, especially for binary classification problems.

$$Gini(S)=1-\sum i=1c(pi)^2$$

$Gini(S)$ is the Gini impurity of the set

$S.c$ is the number of classes.

pi is the probability of randomly choosing an element of class i from the set S.

---

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

Answer:-Yes, unregularized decision trees are prone to overfitting. This is because decision trees have a tendency to grow until they perfectly classify the training data, which can lead to overly complex models that capture noise and irrelevant patterns in the data .

Below are the few reasons why unregularized decision-tree prone to overfitting .

1. Complexity:Without regularization, decision trees can become very deep and complex, resulting in highly specific rules that only apply to the training data. These intricate structures may not generalize well to new data, as they may be overly tailored to the training set.
2. Sensitive to Noise:Decision trees are sensitive to noise and outliers in the training data. Without regularization, they may create branches to accommodate these noisy data points, which can lead to overfitting and poor performance on new data.
3. Memorization:Unregularized decision trees have the capacity to memorize the training data, including noise and outliers, rather than learning the true underlying relationships between features and the target variable. This memorization can result in poor generalization performance on unseen data.

---

6. What is an ensemble technique in machine learning?

Answer:-The ensemble technique in machine learning is method that combines the predictions of multiple individual models to produce the final prediction.There are several types of ensemble technique bagging ,boosting ,stacking and voting .Two mainly used ensemble technique are bagging and boosting.Bagging and Boosting are both ensemble techniques used to improve the predictive performance of machine learning models, but they differ in their approach to combining multiple models and how they address the weaknesses of individual models. Bagging involves training multiple models independently and averaging their predictions, while Boosting focuses on sequentially training multiple models to correct the errors made by the previous models

Bagging:- Bagging involves training multiple instances of the same learning algorithm on different subsets of the training data, each obtained through bootstrap sampling.

Boosting:-Boosting focuses on sequentially training models, where each subsequent model corrects the errors made by the previous ones. It assigns more weight to misclassified instances to improve their importance in subsequent models.

Ensemble techniques are widely used in machine learning because they often result in models with better predictive performance, improved generalization, and increased robustness compared to individual base models.

---

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

Answer:-Bagging and Boosting are both ensemble techniques used to improve the predictive performance of machine learning models, but they differ in their approach to combining multiple models and how they address the weaknesses of individual models. Bagging involves training multiple models independently and averaging their predictions, while Boosting focuses on sequentially training multiple models to correct the errors made by the previous models.

Here are the main differences between Bagging and Boosting techniques.

1. Model Complexity:

Bagging: Bagging tends to reduce overfitting by averaging or voting over multiple models, which helps to smooth out the predictions and reduce variance.

Boosting: Boosting can sometimes lead to overfitting, especially if the base models are too complex or if the boosting process continues for too many iterations. However, boosting algorithms like AdaBoost and Gradient Boosting include regularization techniques to prevent overfitting.

2.Weighting of Models:

Bagging: In Bagging, all base models are given equal weight when making predictions. The final prediction is typically obtained by averaging (in regression) or voting (in classification) the predictions of all base models.

Boosting: In Boosting, each base model is assigned a weight based on its performance during training. Models that perform well are given higher weights, while models that perform poorly are given lower weights. The final prediction is a weighted sum of the predictions of all base models.

3.Final Prediction:

Bagging: The final prediction is often obtained by averaging or voting on the predictions of individual models. Bagging aims to reduce variance and improve stability.

Boosting: The final prediction is a weighted sum of the predictions of all the learners. Boosting aims to reduce both bias and variance, leading to better overall predictive performance.

---

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

Answer:-The out-of-bag (OOB) error is a concept associated with random forests, which is an ensemble learning method. In a random forest, each tree in the forest is trained on a bootstrapped sample of the original dataset, meaning that some observations may be included multiple times, while others may not be included at all.

The OOB error is the prediction error of each individual tree in the random forest, calculated using only the observations that were not included in the training set of that particular tree. Since each tree is trained on a different subset of the data, the OOB error provides an unbiased estimate of the model's performance on unseen data.

The OOB error is useful because it allows for an assessment of the random forest's predictive accuracy without the need for a separate validation set. It serves as a built-in cross-validation measure, providing an indication of how well the model is likely to generalize to new, unseen data.

---

9. What is K-fold cross-validation?

Answer:-K-fold cross-validation is a technique used in machine learning to assess the performance and generalization ability of a model. The dataset is divided into K subsets (or folds), and the model is trained and evaluated K times. In each iteration, one of the K subsets is used as the test set, while the remaining K-1 subsets are used for training the model.

The K-fold cross-validation process can be summarized as follows:

The dataset is divided into K equally-sized folds.
The model is trained K times.
In each iteration, one of the K folds is used as the test set, and the remaining K-1 folds are used as the training set.
The model's performance is evaluated on the test set in each iteration, producing K performance metrics.
The final performance metric is often the average of the K metrics.
K-fold cross-validation helps in obtaining a more robust performance estimate for the model, as it ensures that each data point is used for both training and testing at least once. This helps to reduce the impact of variability in a single train-test split and provides a more reliable assessment of the model's ability to generalize to new, unseen data. Common choices for K include 5-fold and 10-fold cross-validation, but the value of K can be adjusted based on the size and characteristics of the dataset.

---

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

Answer:-Hyperparameter tuning, also known as hyperparameter optimization or model selection, is the process of finding the optimal values for the hyperparameters of a machine learning model. Hyperparameters are external configurations of a model that cannot be learned from the data and must be set prior to the training process. They influence the overall behavior and performance of the model.

Examples of hyperparameters include learning rates, regularization strengths, the number of hidden layers and units in a neural network, the depth of a decision tree, and so on. The choice of hyperparameter values can significantly impact the performance of the model.

Hyperparameter tuning is done for several reasons:

Performance Improvement: Adjusting hyperparameters can lead to better model performance. Fine-tuning these parameters can help the model converge faster during training and achieve better accuracy on the validation or test data.

Avoiding Overfitting or Underfitting: Hyperparameters play a crucial role in controlling the complexity of a model. Tuning them helps strike a balance between overfitting (model fitting the training data too closely but performing poorly on new data) and underfitting (model being too simplistic to capture the underlying patterns).

Generalization: Hyperparameter tuning helps improve the model's ability to generalize well to new, unseen data. A well-tuned model is more likely to perform well on diverse datasets.

Resource Utilization: Tuning hyperparameters can optimize the use of computational resources. For example, selecting an appropriate batch size or the number of parallel processes during training can impact training efficiency.

---

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

Answer:-Using a large learning rate in gradient descent can lead to several issues, impacting the convergence and performance of the optimization process. Here are some common problems associated with a large learning rate:

Divergence: A large learning rate may cause the optimization algorithm to overshoot the minimum of the loss function. Instead of converging to the minimum, the updates may oscillate or diverge, making the optimization process unstable.

Overshooting the Minimum: If the learning rate is too large, the algorithm may take excessively large steps during each iteration. This can cause it to overshoot the optimal parameter values, preventing convergence or causing the algorithm to oscillate around the minimum without reaching it.

Failure to Converge: A high learning rate may prevent the optimization algorithm from converging to a stable solution. The algorithm may keep bouncing back and forth, preventing it from settling at the optimal parameter values.

Skipping the Minimum: With a large learning rate, the algorithm may skip over the minimum of the loss function entirely. This is particularly problematic for non-convex and highly irregular loss surfaces where finding the global minimum is challenging.

Numeric Overflow or Underflow: Large learning rates can lead to numeric instability, causing numerical overflow or underflow issues during the parameter updates. This can result in NaN (Not a Number) values, making the optimization process fail.

---

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

Answer:-Logistic Regression is a linear classification algorithm, meaning it models the relationship between the independent variables and the binary outcome using a linear function. It assumes a linear decision boundary between the classes, which makes it suitable for linearly separable data where the classes can be separated by a straight line.

However, Logistic Regression may not perform well on non-linear data where the decision boundary is not linear. If the relationship between the independent variables and the outcome is non-linear, Logistic Regression will struggle to capture this complex relationship, leading to poor classification performance.

In cases where the data is non-linearly separable, using Logistic Regression directly may result in underfitting, where the model is too simplistic to capture the underlying patterns in the data. In such scenarios, more complex models or non-linear classification algorithms, such as Support Vector Machines (SVMs), Decision Trees, Random Forests, Gradient Boosting Machines, or neural networks, are typically more appropriate.

---

13. Differentiate between Adaboost and Gradient Boosting.

Answer:-AdaBoost (Adaptive Boosting) and Gradient Boosting are both ensemble learning methods used for classification and regression tasks. While they share similarities in their boosting approach, they differ in certain aspects:

Base Learners:
AdaBoost: AdaBoost builds a strong classifier by combining multiple weak classifiers. A weak classifier is typically a simple model, such as a decision tree with only one level of depth (a decision stump).
Gradient Boosting: Gradient Boosting builds an ensemble of weak learners (often decision trees) in a sequential manner, where each new model corrects the errors made by the previous ones. The weak learners in Gradient Boosting are not necessarily as weak as those in AdaBoost and can be more complex, like decision trees with multiple levels.

Weighting of Data Points:
AdaBoost: In AdaBoost, the weights of the misclassified data points are increased at each iteration, focusing on the harder-to-classify instances to improve the model's performance.
Gradient Boosting: Gradient Boosting, on the other hand, uses gradient descent optimization to minimize a loss function (e.g., mean squared error for regression or cross-entropy loss for classification). It assigns weights to the residuals (the differences between the predicted and actual values) and fits the next weak learner to the residuals.

Parallelism:
AdaBoost: AdaBoost can be parallelized as each weak learner can be trained independently.
Gradient Boosting: Gradient Boosting, especially in its most popular implementations like XGBoost and LightGBM, supports parallelization by enabling multiple trees to be built simultaneously. However, the boosting process itself is sequential since each new model depends on the previous one.

---

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

Answer:-The bias-variance tradeoff is a fundamental concept in machine learning that relates to the performance of a model and its ability to generalize to unseen data. It describes the tradeoff between two sources of error: bias and variance.

Bias:

Bias measures how closely the predictions of a model match the true values of the data. A high bias indicates that the model is too simplistic and unable to capture the underlying patterns in the data. Models with high bias tend to underfit the data, meaning they perform poorly not only on the training data but also on new, unseen data.

Variance:

Variance measures the variability of a model's predictions across different datasets. A high variance indicates that the model is too sensitive to small fluctuations in the training data.

Models with high variance tend to overfit the data, meaning they perform well on the training data but generalize poorly to new, unseen data.

The bias-variance tradeoff arises because decreasing one component (bias or variance) often leads to an increase in the other component. Finding the right balance between bias and variance is crucial for building models that generalize well to new data.

---

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

Answer:-Support Vector Machines (SVMs) are a class of supervised learning algorithms that can be used for both classification and regression tasks. They are particularly effective in high-dimensional spaces and are well-suited for situations where the data is not linearly separable. SVMs use kernels to transform the input features into a higher-dimensional space, making it possible to find a hyperplane that effectively separates the data.

Here are short descriptions of three commonly used kernels in SVM:

Linear Kernel:

The linear kernel is the simplest kernel and is suitable for linearly separable data. It computes the dot product of the input features, effectively applying a linear transformation to the data.

RBF (Radial Basis Function) Kernel:

The RBF kernel, also known as the Gaussian kernel, is a popular choice for handling non-linear relationships in the data. It maps the input features into an infinite-dimensional space using a radial basis function. The resulting decision boundary is non-linear and can adapt to complex patterns.

Polynomial Kernel:

The polynomial kernel introduces non-linearity by computing the polynomial combinations of the input features. It allows the SVM to capture complex relationships between features.
