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 and Residual Sum of Squares (RSS) are both measures of the goodness of fit of a regression model, but they capture different aspects of the fit.

R-squared, also known as the coefficient of determination, measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, with a higher value indicating a better fit. R-squared is useful for comparing different models or for determining the proportion of the variability in the dependent variable that is explained by the model.

On the other hand, Residual Sum of Squares (RSS) measures the total amount of unexplained variance in the dependent variable that remains after the model has been fit. It is the sum of the squared differences between the actual and predicted values of the dependent variable. A lower value of RSS indicates a better fit. RSS is useful for evaluating the accuracy of the predictions of the model.

In general, both measures are important and should be considered together when evaluating the goodness of fit of a model. However, R-squared is often considered to be a better measure of goodness of fit than RSS because it provides a single number that summarizes the proportion of variance in the dependent variable that is explained by the model, which is more interpretable and easier to compare across 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.

Answer: In regression analysis, these terms represent measures of variability:

TSS (Total Sum of Squares): It measures the total variability in the dependent variable (Y) and represents the sum of the squared differences between the observed dependent variable values and their mean.
ESS (Explained Sum of Squares): It represents the variability explained by the regression model. It measures the sum of the squared differences between the predicted values and the mean of the dependent variable.
RSS (Residual Sum of Squares): It measures the unexplained variability in the dependent variable by the regression model. It is the sum of the squared differences between the observed values and the predicted values.
The relationship between these three metrics can be expressed as:
TSS=ESS+RSS
This equation indicates that the total variability in the dependent variable can be decomposed into the variability explained by the regression model (ESS) and the unexplained variability or error (RSS).

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

Answer: Regularization in machine learning is used to prevent overfitting, which occurs when a model learns the training data too closely, including its noise and outliers. Regularization techniques add a penalty term to the model's loss function, discouraging overly complex models that might fit the training data perfectly but perform poorly on new, unseen data. This helps in improving the model's generalization capability.

4. What is Gini–impurity index?

Answer: Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. It reaches its minimum (zero) when all cases in the node fall into a single target category.

Let's take the sample data as given in following image:

Obs	Emotion	Temperature	StayHome?
1	*Sick*	*Temp*98	*N*
2	*Sick*	*Temp*105	*Y*
3	*NotSick*	*Temp*98	*Y*
4	*NotSick*	*Temp*98	*N*
5	*NotSick*	*Temp*101	*Y*
6	*Sick*	*Temp*101	*N*
7	*NotSick*	*Temp*99.5	*N*
8	*NotSick*	*Temp*102	*Y*

First, we calculate for the feature **Emotion**

- Sick Gini impurity = 2 * (2/3) * (1/3) = 0.444
- NotSick Gini Impurity = 2 * (3/5) * (2/5) = 0.48
- Weighted Gini Split = (3/8) * SickGini + (5/8) NotSickGini = 0.4665

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

Answer: Yes, unregularized decision trees are prone to overfitting. Overfitting occurs when a model learns the training data too closely, including its noise and outliers, to the extent that it performs poorly on unseen data.

In the case of unregularized decision trees:

Complexity: They can become very deep and complex, leading to capturing noise in the training data rather than the underlying pattern.

High Variance: Without regularization, decision trees can have high variance, meaning they can produce very different models with slight variations in the training data.

Sensitive to Small Changes: Small changes or noise in the training data can result in different splits and tree structures, making the model unstable.

To combat overfitting, techniques like pruning (to reduce tree size), setting a maximum depth, or using ensemble methods like Random Forests (which build multiple trees and average their predictions) are often employed.

6. What is an ensemble technique in machine learning?

Answer: In machine learning, an ensemble technique combines predictions from multiple models to produce a final prediction. The idea is that by combining several models, the ensemble can often achieve better performance than any individual model. Common ensemble methods include bagging, boosting, and stacking.

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

Answer: Both Bagging and Boosting are ensemble learning techniques, but they differ in their approach:

Bagging (Bootstrap Aggregating):

Sampling Technique: Uses bootstrap sampling to create multiple subsets of the training data. Each subset is of the same size as the original dataset.

Model Training: Trains multiple base learners (e.g., decision trees) on these subsets independently and in parallel.

Combining Predictions: For classification, the final prediction is typically made by majority voting. For regression, it might be an average.

Example: Random Forest is a popular ensemble method that uses Bagging with decision trees.

Boosting:

Sampling Technique: Does not use bootstrap sampling. Instead, it focuses on the instances that are harder to classify. The weights of instances are adjusted based on the performance of the previous models.

Model Training: Iteratively trains a sequence of weak learners (e.g., shallow trees) where each subsequent learner corrects the errors of its predecessor.

Combining Predictions: The final prediction is often made by weighted majority voting, where weights depend on the performance of each model.

Example: Ada Boost (Adaptive Boosting) and Gradient Boosting Machines (GBM) are popular boosting algorithms.

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

Answer: In the context of Random Forests, the out-of-bag (OOB) error is an estimate of a model's performance without the need for a separate validation set or cross-validation. Here's how it works:

Bootstrap Sampling: In the process of building each decision tree within the Random Forest, a bootstrap sample (a sample drawn with replacement from the original data) is used. This means some data points from the original dataset will not be included in a particular bootstrap sample.

Out-of-Bag Data: The data points that are not included in the bootstrap sample for a given tree are referred to as out-of-bag (OOB) data for that tree.

OOB Prediction: After training the Random Forest, each data point in the OOB set is passed down all the trees where it was out-of-bag. The majority class (for classification) or average prediction (for regression) across all these trees is used as the OOB prediction for that data point.

OOB Error: The OOB error is then computed by comparing the OOB predictions to the true values in the out-of-bag samples. For classification tasks, it's the error rate; for regression tasks, it's the mean squared error.

The advantage of OOB error in Random Forests is that it provides a built-in estimate of the generalization error, allowing you to assess model performance without needing an additional validation set.

In summary, while both techniques aim to improve model performance by combining multiple learners, Bagging focuses on parallel training of base models using bootstrapped samples, whereas Boosting emphasizes sequential training of models, giving more attention to misclassified instances.

9. What is K-fold cross-validation?

Answer: K-fold cross-validation is a resampling technique used to assess the performance of a machine learning model on different subsets of the training dataset. Here's a brief explanation:

Data Splitting: The original dataset is divided into k subsets, or "folds", of approximately equal size.

Iteration: The model is trained k times. In each iteration:

One of the k subsets is used as the validation set.

The remaining k-1 subsets are combined and used as the training set.

Performance Metrics: After each iteration, the model's performance is evaluated using a chosen metric (e.g., accuracy, mean squared error) on the validation set.

Average Performance: Once all k iterations are completed, the performance results from each fold are averaged to obtain a single estimation of model performance.

Benefits of k-fold cross-validation:

Provides a more robust estimate of model performance compared to a single train-test split.

Utilizes the entire dataset for training and validation, maximizing data usage.

Common choices for k in k-fold cross-validation include 5 or 10, but the value can be adjusted based on the dataset size and specific requirements.

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

Answer: Hyper parameter tuning in machine learning refers to the process of selecting the optimal hyper parameters for a given algorithm. Hyper parameters are parameters that are not learned from the data but are set before the learning process begins. Examples include the learning rate in gradient descent, the depth of a decision tree in random forests, or the number of hidden layers in a neural network.

Here's why hyper parameter tuning is essential:

Optimized Performance: Properly tuned hyper parameters can significantly improve the performance of a model, leading to better accuracy, precision, or other relevant metrics.

Prevent Over fitting/Under fitting: Incorrect hyper parameters can lead to over fitting (model performs well on training data but poorly on new, unseen data) or under fitting (model is too simple to capture underlying patterns). Tuning helps find a balance.

Model Robustness: A well-tuned model is likely to generalize better to new, unseen data, making it more robust and reliable.

Methods for hyper parameter tuning include manual tuning, grid search, random search, Bayesian optimization, and more. The choice of method often depends on the computational resources available and the complexity of the hyper parameter space.

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:

Overshooting: The algorithm may jump across the minimum point, causing it to oscillate or diverge, making it difficult to converge to the optimal solution.

Divergence: Instead of converging to a minimum, the algorithm might move away from it, leading to a failure in convergence and making the model training ineffective.

Instability: High learning rates can lead to unstable training dynamics, where the loss function may show erratic behavior during the learning process.

Poor Generalization: The model might perform well on the training data but fail to generalize well to unseen data, as it may not have reached a stable and optimal solution.

Increased Computational Time: The algorithm may require more iterations to converge, as it might overshoot the optimal solution and then have to backtrack, leading to increased computational time.

It's essential to choose an appropriate learning rate and potentially implement techniques like learning rate decay or adaptive learning rates to ensure stable and efficient training.

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

Answer: Logistic Regression is inherently a linear classifier. This means it assumes that the relationship between the independent variables and the log-odds of the dependent variable is linear. If the data is non-linear, logistic regression might not capture the underlying pattern

effectively. In such cases, more complex models or non-linear transformations of the data might be required to achieve better classification performance.

13. Differentiate between Adaboost and Gradient Boosting.

Answer: Sure, both AdaBoost (Adaptive Boosting) and Gradient Boosting are ensemble methods used to improve the performance of machine learning models, particularly decision trees. However, they differ in several ways:

Loss Function:

AdaBoost: Focuses on the misclassified data points. It assigns higher weights to misclassified points and then adjusts the model to focus more on those points.

Gradient Boosting: Generalizes the concept of boosting by optimizing any arbitrary differentiable loss function. It tries to fit the new model to the residual errors of the previous model.

Weighting:

AdaBoost: Adjusts the weights of instances at each iteration, focusing more on misclassified instances.

Gradient Boosting: Does not typically adjust instance weights. Instead, it fits the new model to the residual errors from the previous model.

Base Learners:

AdaBoost: Typically uses a decision stump (a decision tree with one level).

Gradient Boosting: Can use a variety of base learners, but decision trees are common.

Learning Rate:

AdaBoost: Uses a learning rate to control the contribution of each model to the final ensemble.

Gradient Boosting: Also uses a learning rate, but it's used to control the step size during optimization rather than the contribution of each model.

Parallelization:

AdaBoost: Can be parallelized since each iteration's weights depend only on the previous iteration.

Gradient Boosting: Is typically sequential, making it harder to parallelize because each model's training depends on the previous one.

In summary, while both AdaBoost and Gradient Boosting are boosting algorithms that combine weak learners into a strong learner, they differ in their approach to weighting instances, handling residuals, and optimization strategies.

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

Answer: The bias-variance trade-off is a fundamental concept in machine learning that helps explain the balance between a model's simplicity and its ability to generalize to unseen data:

Bias: Refers to the error introduced by approximating a real-world problem, which means it's the difference between the expected prediction of our model and the correct value. High bias can cause the model to miss relevant relations between features and target outputs, leading to underfitting (too simple).

Variance: Refers to a model's sensitivity to small fluctuations or noise in the training data. High variance can cause the model to model the random noise in the training data, leading to overfitting (too complex).

In essence, the goal is to find a model that has a good trade-off between bias and variance to ensure it performs well on both training data and unseen test data.

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

Answer: Certainly! Here's a brief description of each kernel used in Support Vector Machines (SVM):

Linear Kernel: The simplest kernel, it computes the dot product between the input features. It works well when the data is linearly separable or when the number of features is very high.

RBF (Radial Basis Function) Kernel: Also known as the Gaussian kernel, it maps data into an infinite-dimensional space. It measures the similarity between two samples based on their Euclidean distance. It's effective for capturing complex nonlinear relationships in the data.

Polynomial Kernel: This kernel computes the similarity between samples using polynomial terms. It allows the SVM to model more complex decision boundaries by considering higher-order relationships between data points.

Each of these kernels allows the SVM to transform data into a higher-dimensional space, making it easier to find an optimal separating hyper plane for classification tasks.