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-

TSS (Total Sum of Squares):

Measures the total variability of the dependent variable (y) around its mean ($\bar{y}$).

It considers how much the data points spread out from the overall average.

Calculated as:

$TSS = \Sigma(yi - \bar{y})^2$

ESS (Explained Sum of Squares):

Represents the portion of the total variability that is explained by the regression model.

It measures how well the fitted regression line accounts for the variation in y.

Calculated as:
$ESS = \Sigma(\hat{y}i - \bar{y})^2$

RSS (Residual Sum of Squares):
Represents the remaining variability that is not explained by the model.
It's the sum of the squared differences between the actual y values and the predicted values ($\hat{y}$) from the regression line.
Also known as SSE (Sum of Squared Errors).
Calculated as:
$RSS = \Sigma(yi - \hat{y}i)^2$

Key Relationship:
TSS is always equal to the sum of ESS and RSS:
TSS = ESS + RSS

Interpretation:

- A larger ESS relative to TSS indicates that the model explains a greater proportion of the total variability in the data, suggesting a better fit.
- A smaller RSS indicates that the model's predictions are closer to the actual values, also suggesting a better fit.

Key Uses:

- R-squared (R^2): A measure of the model's goodness of fit, calculated as ESS/TSS. It Answer-
- represents the proportion of the total variability explained by the model.
- F-test: Assesses the overall significance of the regression model, using the relationship between ESS and RSS.
- Model comparison: Helps compare different regression models to select the one that best explains the data.

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 mode

4-What is Gini–impurity index?
Answer-

The Gini impurity index, also known as Gini index or Gini coefficient, is a measure used in decision tree learning to assess the impurity or disorder within a node (data point group) of a decision tree. It essentially tells you how well a node has been separated into homogeneous groups based on their class labels. Here's how it works:

- The Gini index for a node is calculated as the probability of misclassifying a randomly chosen element from that node if the classification was done randomly based on the class distribution within the node.
- It ranges from 0 to 0.5:
- 0: Perfect purity, all elements belong to the same class.
- 0.5: Maximum impurity, classes are equally distributed within the node.
- Lower Gini index indicates a better split, as the elements within the node are more likely to belong to the same class.

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

Answer-

Yes, unregularized decision trees are indeed prone to overfitting. Here's a breakdown of why this happens, accompanied by visual aids:

Understanding Overfitting:

- Overfitting occurs when a model learns the training data too well, including its noise and idiosyncrasies, and fails to generalize to new, unseen data.
- This results in excellent performance on the training set but poor performance on real-world data.

Why Decision Trees Overfit:

1. Flexibility: Decision trees can create highly complex structures with many branches and leaves, allowing them to capture intricate patterns in the training data.
2. No Built-in Regularization: They lack inherent mechanisms to control complexity and prevent overfitting, unlike some other algorithms (e.g., linear regression with L1/L2 regularization).
3. Recursive Splitting: The tree-building process involves recursively splitting data into smaller and smaller subsets, potentially leading to over-adaptation to specific data points and noise.

Visualizing Overfitting in Decision Trees:

Preventing Overfitting in Decision Trees:

- Pruning: Remove unnecessary branches or leaves to simplify the tree structure.
- Regularization: Impose constraints during training to penalize model complexity, such as limiting tree depth or the number of leaves.
- Ensemble Methods: Combine multiple trees (e.g., random forests, boosting) to reduce variance and improve generalizability.
- Cross-Validation: Evaluate model performance on a separate validation set to detect overfitting early and adjust complexity accordingly.

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

Answer-

In machine learning, an ensemble technique combines the predictions of multiple individual models to improve overall performance compared to any single model alone. Think of it like a team of experts working together to make a more informed decision than any one expert could independently.

Here's a breakdown of the key aspects of ensemble techniques:

How they work:

- Multiple models: Instead of relying on a single model, ensemble techniques create several different models (often called "base models") trained on the same data.
- Diversity: These base models can be of different types (e.g., decision trees, neural networks) or trained with different parameters, ensuring they capture different aspects of the data.
- Combining predictions: The results of these individual models are then combined in some way to generate a final prediction. This can involve averaging, voting, or using a more complex meta-model to learn how to best weight each model's contribution.

Benefits of ensemble techniques:

- Improved accuracy: By leveraging the strengths of different models and

minimizing their weaknesses, ensembles can achieve higher accuracy and robustness than any single model.
- Reduced variance: Combining multiple predictions reduces the overall variance of the results, making them less sensitive to noise and outliers in the data.
- Enhanced generalizability: Ensembles tend to generalize better to unseen data, meaning they perform well on data they haven't been specifically trained on.
- Handling complex problems: Ensembles can tackle complex problems with highly non-linear relationships or multi-modal distributions that individual models struggle with.

Common ensemble techniques:

- Bagging (Bootstrap Aggregating): Trains multiple models on different random subsets of the training data and averages their predictions. Examples include Random Forests and Isolation Forests.
- Boosting: Sequentially trains models, each attempting to correct the errors of the previous model, resulting in a strong ensemble. Examples include Gradient Boosting and XGBoost.
- Stacking: Trains a meta-model to combine the predictions of multiple base models, learning how to best weight their contributions.

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

Answer-

Difference Between Bagging and Boosting: Bagging vs Boosting

	Bagging	Boosting
Basic Concept	Combines multiple models trained on different subsets of data.	Train models sequentially, focusing on the error made by the previous model.
Objective	To reduce variance by averaging out individual model error.	Reduces both bias and variance by correcting misclassifications of the previous model.

Data Sampling	Use Bootstrap to create subsets of the data.	Re-weights the data based on the error from the previous model, making the next models focus on misclassified instances.
Model Weight	Each model serves equal weight in the final decision.	Models are weighted based on accuracy, i.e., better-accuracy models will have a higher weight.
Error Handling	Each model has an equal error rate.	It gives more weight to instances with higher error, making subsequent model focus on them.
Overfitting	Less prone to overfitting due to average mechanism.	Generally not prone to overfitting, but it can be if the number of the model or the iteration is high.
Performance	Improves accuracy by reducing variance.	Achieves higher accuracy by reducing both bias and variance.
Common Algorithms	Random Forest	AdaBoost, XGBoost, Gradient Boosting Mechanism
Use Cases	Best for high variance, and low bias models.	Effective when the model needs to be adaptive to errors, suitable for both bias and variance errors.

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

Answer-

> Out-of-bag error, also known as out-of-bag (OOB) error, is a unique feature of random forest models that allows for estimating the model's generalization performance without needing a separate validation set. This makes it a valuable tool for assessing the accuracy and potential for overfitting in random forests.

Out-of-bag error, also known as out-of-bag (OOB) error, is a unique feature of random forest models that allows for estimating the model's generalization performance without needing a separate validation set. This makes it a valuable tool for assessing the accuracy and potential for overfitting in random forests.

Here's a breakdown of how it works:

Random forest basics:

- Random forests consist of an ensemble of decision trees, each trained on a bootstrapped sample of the original data.
- Bootstrapping involves randomly sampling the data with replacement, creating multiple subsets of the same size as the original data.
- Each tree uses a different random subset of features for splitting nodes, further diversifying the ensemble.

OOB error calculation:

- For each data point in the original dataset, it never participates in the training of any of the trees used to predict its OOB error.
- This is because its data point will not be present in any of the randomly drawn bootstrap samples used to train the individual trees.
- For each data point, its OOB error is calculated by averaging the predictions of all the trees in the forest except the ones it was not included in during training.

Benefits of OOB error:

- No need for separate validation set: Eliminates the need for setting aside a portion of the data for validation, potentially increasing the available training data.
- Provides an unbiased estimate: OOB error provides an unbiased estimate of the model's performance on unseen data, as the data points are never used in the training process.
- Helps monitor overfitting: As the number of trees in the forest increases, the OOB error can start to stabilize or even increase, indicating potential overfitting.

Limitations of OOB error:

- Underestimates true error: OOB error tends to underestimate the true error on unseen data, as it doesn't involve completely new data points.
- Not directly comparable to traditional validation metrics: OOB error is based on a different calculation than traditional validation metrics used for other learning algorithms.

9-What is K-fold cross-validation?

Answer-

K-fold cross-validation is a widely used method for evaluating the performance of machine learning models and selecting appropriate hyperparameters. It involves dividing the dataset into k equal-sized folds, or subsets, and then iteratively training and evaluating the model on different combinations of these folds.

Key advantages of K-fold cross-validation:

- Reduces bias: By using all data for both training and validation, it provides a more reliable estimate of model performance compared to using a single validation set.
- Minimizes variance: Iteratively training on different folds helps reduce the influence of random variations in the data split.
- Improves model selection: Allows for comparison of different models or model configurations to select the best performing one.
- Hyperparameter tuning: Helps find the optimal hyperparameters for a model by evaluating different combinations using cross-validation.

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

Answer-

When you're training machine learning models, each dataset and model needs a different set of hyperparameters, which are a kind of variable. The only way to determine these is through multiple experiments, where you pick a set of hyperparameters and run them through your model. This is called *hyperparameter tuning*. In essence, you're training your model sequentially with different sets of hyperparameters. This process can be manual, or you can pick one of several automated hyperparameter tuning methods.

Whichever method you use, you need to track the results of your experiments. You'll have to apply some form of statistical analysis, such as the loss function, to determine which set of hyperparameters gives the best result. Hyperparameter tuning is an important and computationally intensive process.

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

Answer-

The learning rate is an important hyperparameter that greatly affects the performance of gradient descent. It determines how quickly or slowly our model learns, and it plays an important role in controlling both convergence and divergence of the algorithm. When the learning rate is too large, gradient descent can suffer from divergence. This means that weights increase exponentially, resulting in exploding gradients which can cause problems such as instabilities and overly high loss values. On the other hand, if the learning rate is too small, then gradient descent can suffer from slow convergence or even stagnation—which means it may not reach a local minimum at all unless many iterations are performed on large datasets.

In order to avoid these issues with different learning rates for each parameter/variable, we use adaptive techniques such as Adagrad and Adam which adjust their own learning rates throughout training based on real-time observations of parameters during optimization (i.e., they control exploration/exploitation trade-offs). These adaptive measures ensure better results than standard gradient descent while avoiding potential pitfalls in terms of either massive gains or slow losses due to misconfigured static global learning rates like those used with traditional gradient descent algorithms.

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

Answer-
While logistic regression is a powerful classification algorithm, it's not well-suited for directly handling non-linear relationships in the data. Here's why:

Linear Decision Boundary:

- Logistic regression inherently creates a linear decision boundary, meaning it separates classes using a straight line (in 2D) or a hyperplane (in higher dimensions).
- This works well when the data points belonging to different classes are naturally separable by a linear boundary.

Inherent Linearity:

- The model's prediction function, the logistic function, is itself a linear function of the input features. This limits its ability to capture complex, non-linear patterns.

Limitations with Non-Linear Data:

- If the decision boundary between classes is inherently non-linear (e.g., curved, circular), logistic regression will struggle to model it accurately.
- Forcing a linear fit on non-linear data can lead to poor performance and misleading results.

13-. Differentiate between Adaboost and Gradient Boosting.

Answer-
Difference between Gradient Boosting and Adaptive Boosting(AdaBoost)

Gradient boosting	**Adaptive Boosting**
This approach trains learners based upon minimising the loss function of a learner (i.e., training on the residuals of the model)	This method focuses on training upon misclassified observations. Alters the distribution of the training dataset to increase weights on sample observations that are difficult to classify.
Weak learners are decision trees constructed in a greedy manner with split points based on purity scores (i.e., Gini, minimise loss). Thus, larger trees can be used with around 4 to 8 levels. Learners should still remain weak and so they should be constrained (i.e., the maximum number of layers, nodes, splits, leaf nodes)	The weak learners incase of adaptive boosting are a very basic form of decision tree known as stumps.

All the learners have equal weights in the case of gradient boosting. The weight is usually set as the learning rate which is small in magnitude.	The final prediction is based on a majority vote of the weak learners' predictions weighted by their individual accuracy.

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

Answer-

The bias-variance trade-off is a fundamental concept in machine learning, describing the relationship between bias and variance in a model and their combined impact on its performance.

Key players:

- Bias: The tendency of a model to consistently miss the underlying relationship between features and the target variable, leading to systematic errors. Think of it as the model's "prejudice" towards certain outcomes.
- Variance: The amount of fluctuation in a model's predictions for the same input due to sensitivity to small changes in the training data. Imagine a jittery hand drawing a line – that's high variance.

The Trade-off:

- Reducing bias often involves increasing variance and vice versa.
- A low-bias model adheres closely to the training data, potentially missing the general patterns for unseen data (high variance).
- Conversely, a low-variance model captures general patterns well, but might overlook certain nuances in the training data and introduce systematic errors (high bias).

Finding the Sweet Spot:

The goal is to find the sweet spot where the model has a balance between bias and variance, minimizing both without sacrificing one for the other. This "sweet spot" leads to a model that can generalize well to unseen data and avoid both systematic errors and overfitting.

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

Answer-

1. Linear Kernel:

- Projects data points onto a higher-dimensional space using a linear transformation.
- Suitable for: Linearly separable data or when computational efficiency is a priority.
- Equation: K(x, y) = x^T * y (dot product of two vectors)

2. RBF (Radial Basis Function) Kernel:

- Transforms data into an infinite-dimensional space, enabling complex non-linear decision boundaries.
- Suitable for: Non-linear relationships, highly dimensional data, and when generalization ability is crucial.
- Equation: K(x, y) = exp(-gamma * ||x - y||^2), where gamma is a hyperparameter controlling kernel width.

3. Polynomial Kernel:

- Combines the features of linear and non-linear kernels, allowing for flexible decision boundaries.
- Suitable for: Datasets with features that interact with each other in a non-linear way.
- Equation: K(x, y) = (x^T * y + 1)^d, where d is the degree of the polynomial.