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?

R-squared is considered a better measure of goodness of fit for a model in regression
because it is normalized and provides an interpretable percentage of variance explained by the model, which makes it easier to interpret and compare across different models and datasets.

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.

In regression:
TSS (Total Sum of Squares) represents the total variance in the observed data.
ESS (Explained Sum of Squares) represents the portion of TSS that is explained by the regression model.
RSS (Residual Sum of Squares) represents the portion of TSS that is not explained by the regression model.
The equation relating these three metrics is:
TSS = ESS + RSS
This equation shows that the total variability of the data can be partitioned into the variability explained by the model and the variability that is unexplained (residuals).

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

Regularization in machine learning is needed to prevent overfitting. Overfitting happens when a model learns not only the underlying patterns but also the noise in the training data, which can negatively impact its performance on new data. Regularization techniques add a penalty for complexity to the model's loss function, encouraging simpler models that generalize better to new data.

4. What is Gini–impurity index?

The Gini impurity index is a metric used in decision trees to quantify how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. It ranges from 0 to 0.5 .

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

Yes, unregularized decision-trees are prone to overfitting because they can grow indefinitely and create very complex models that fit the noise and specific details of the training data rather than the underlying data distribution. This complexity can result in poor performance on unseen data.

6. What is an ensemble technique in machine learning?

An ensemble technique in machine learning is a strategy that combines multiple models to improve the robustness and accuracy of predictions. By aggregating the predictions of several models, ensemble methods can often achieve better performance than any single model could alone.

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

The difference between Bagging and Boosting techniques is:

Bagging trains multiple models in parallel on different subsets of the data and combines their predictions through averaging or voting to make the final prediction.

Boosting trains models sequentially, with each model trying to correct the errors made by the previous ones. The predictions are combined through a weighted sum approach, where more weight is given to the predictions made by better-performing models.

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

The out-of-bag (OOB) error in random forests is an estimate of the prediction error for the trees in the forest. It is calculated using only the data that was not included (i.e., “out-of-bag”) in the bootstrap sample for each tree when it was being trained. This provides an internal validation mechanism since the OOB error is based on predictions made on unseen data.

9. What is K-fold cross-validation?

K-fold cross-validation is a technique where the dataset is randomly divided into ‘K’ equal-sized folds. For each fold, the model is trained on ‘K-1’ folds and validated on the remaining fold. This process is repeated ‘K’ times, with each fold being used once as the validation set. The results are then averaged to produce a single estimation of model performance.

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

Hyperparameter tuning in machine learning is the process of finding the best combination of hyperparameters (settings) for a learning algorithm. A hyperparameter is a parameter whose value is set before the learning process begins, unlike other parameters which are learned during training.

The goal of hyperparameter tuning is to optimize the performance of a model on unseen data. This is done because different hyperparameters can significantly impact the model’s ability to learn from data and make accurate predictions. By tuning these parameters, we aim to find the most effective settings for the specific problem and data at hand.

11. What issues can occur if we have a large learning rate in Gradient Descent?
If the learning rate in Gradient Descent is too large, it can cause several issues:

1. Overshooting Minimum: The model may overshoot the global minimum or the best solution because large steps may not allow it to settle into the lowest point of the loss function.
2. Divergence: Instead of converging to a solution, the model’s performance may actually worsen over time as the large steps increase the error.
3. Instability: The updates may be too aggressive, causing the model parameters to fluctuate wildly and never settle down.

A large learning rate can prevent the model from learning effectively, leading to poor performance.

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

Logistic Regression is a linear model for classification. It works best when the decision boundary between classes is linear or nearly linear. For non-linear data, where the relationship between features and the target variable is not linear, Logistic Regression may not perform well because it cannot capture the non-linear patterns.

However, it’s possible to use Logistic Regression on non-linear data by applying techniques like feature transformation or polynomial feature expansion to map the non-linear features into a higher-dimensional space where a linear separation is possible. But in general, other algorithms designed for non-linearity, like decision trees or kernel-based methods, might be more suitable for non-linear data.

13. Differentiate between Adaboost and Gradient Boosting.

AdaBoost and Gradient Boosting are both ensemble learning techniques that combine multiple weak learners to form a strong learner, but they differ in their approach:

AdaBoost (Adaptive Boosting):

- Adjusts the weights of incorrectly classified instances so that subsequent classifiers focus more on difficult cases.
- Uses the same learning algorithm for all weak learners.
- The final prediction is made based on the weighted majority vote (for classification) or weighted sum (for regression) of the weak learners.

Gradient Boosting:

- Improves the model by fitting the new predictor to the residual errors made by the previous predictor.
- Can use different types of learning algorithms for weak learners.
- The final prediction is made by summing up the contributions of all predictors.

In essence, AdaBoost adapts by changing the weights of instances, while Gradient Boosting adapts by sequentially fitting to the residual errors.

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

The bias-variance tradeoff is a fundamental concept in machine learning that describes the tradeoff between two types of errors a model can make:

Bias:

- Error from erroneous assumptions in the learning algorithm.
- High bias can cause the model to miss relevant relations between features and target outputs (underfitting).

Variance:

- Error from sensitivity to small fluctuations in the training set.
- High variance can cause the model to model the random noise in the training data (overfitting).

Ideally, you want to find a balance where both bias and variance are as low as possible to achieve good prediction performance. However, decreasing one typically increases the other. A model with low bias must be complex enough to capture the true patterns, which can make it sensitive to noise (high variance), while a model with low variance might be too simple to capture complex patterns (high bias).

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

Linear Kernel:

- It is a simple kernel function that calculates the linear separation in the input feature space.
- Formula: (K(x, y) = x^T y)

RBF (Radial Basis Function) Kernel:

- It is a popular kernel function that can map an input space into an infinite-dimensional space.
- It is useful for non-linear data separation.
- Formula: (K(x, y) = \exp(-\gamma | x - y |^2)), where (\gamma) is a parameter that defines how much influence a single training example has.

Polynomial Kernel:

- It allows learning of non-linear models by applying polynomial transformation to the input features.
- Formula: (K(x, y) = (1 + x^T y)^d), where (d) is the degree of the polynomial.

Each kernel has its own use case and is chosen based on the data distribution and the problem at hand.