1) Choosing between R-squared and Residual Sum of Squares (RSS) as a measure of goodness of fit in regression depends on the specific objectives and considerations of the analysis. Both R-squared and RSS offer valuable insights into the goodness of fit of a regression model, but they serve different purposes. R-squared provides an overall measure of explanatory power, while RSS measures the accuracy of predictions. Therefore, the choice between R-squared and RSS depends on the specific goals and context of the analysis. It is often advisable to consider both measures when evaluating the performance of a regression model.

2) In 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 the regression model.
   TSS measures the total variability in the dependent variable (Y) around its mean.
   ESS measures the variability in the dependent variable that is explained by the regression model while RSS measures the unexplained variability in the dependent variable by the regression model.
   The relationship between these three metrics is given by the following equation, known as the decomposition of variance:

   TSS = ESS + RSS

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

3) Regularization in machine learning is a technique used to prevent overfitting and under fitting thereby improving the generalization performance of predictive models. Overfitting occurs when a model learns the training data too well, capturing noise or irrelevant patterns that do not generalize well to unseen data. Regularization helps address this issue by imposing constraints on the model's parameters, discouraging overly complex or erratic behavior that may lead to overfitting and help us get an optimal model.

4) The Gini impurity index, often referred to simply as Gini impurity, is a measure of the impurity or uncertainty of a set of data points within a decision tree or classification model. It is commonly used in binary classification problems but can be extended to multi-class classification as well. In the context of decision trees, Gini impurity is used to evaluate the quality of a split in the data.

5) Yes, unregularized decision trees are prone to overfitting, especially when dealing with complex datasets or when the tree is allowed to grow without any constraints. Overfitting occurs when a model learns the training data too well, capturing noise or irrelevant patterns that do not generalize well to unseen data.

6) An ensemble technique refers to a method that combines multiple individual models to improve the overall performance and predictive accuracy of the system. Ensemble techniques are based on the principle that combining the predictions of multiple models often leads to better results than relying on any single model alone.
   There are several types of ensemble techniques, including Bagging, Boosting, Stacking and Voting.
7) Here are the key differences between bagging and boosting:
   Different Training Approach:

   Bagging (Bootstrap Aggregating): Bagging involves training multiple instances of the same base model (e.g., decision trees) on different subsets of the training data, sampled with replacement. Each model is trained independently of the others, and their predictions are combined through averaging or voting.

   Boosting: Boosting is an iterative ensemble technique where base models are trained sequentially, with each subsequent model focusing on correcting the errors made by the previous ones. Each model is trained on the entire training dataset, but the weights of the data points are adjusted during training to give more weight to the misclassified instances.

   Different Training Procedure (Bagging: Parallel; Boosting: Sequential)

   Bagging: In bagging, base models are trained independently in parallel, and their predictions are aggregated afterward. Each model is trained on a random subset of the training data, typically using bootstrapping to sample with replacement.

   Boosting: In boosting, base models are trained sequentially in an iterative manner. Each subsequent model is trained to correct the errors made by the previous models, with a focus on the instances that were misclassified or have higher residuals.

8) In Random Forests, each decision tree in the ensemble is trained on a bootstrap sample of the original training data, meaning that some data points from the original dataset may not be included in the training set for a particular tree. These data points that are not included in the bootstrap sample are referred to as "out-of-bag" (OOB) samples.

9) K-fold cross-validation is a resampling technique used to evaluate the performance of a machine learning model and estimate its predictive accuracy on unseen data. It is particularly useful when the dataset is limited in size or when there is a desire to validate the model's performance in a robust manner.

   In K-fold cross-validation, the original dataset is randomly partitioned into K equal-sized subsets or "folds". The process then involves iteratively training the model K times, each time using K-1 folds as the training set and the remaining fold as the validation set. This results in K trained models and K validation sets.

10) Hyperparameter tuning, also known as hyperparameter optimization, is the process of selecting the optimal hyperparameters for a machine learning model. Hyperparameters are configuration settings that are not learned from the training data but are set before the training process begins. Examples of hyperparameters include the learning rate in gradient descent, the number of hidden layers and neurons in a neural network, the depth of a decision tree, and the regularization parameter in a regression model.

    Hyperparameter tuning is done to maximize the performance of the model on unseen data and improve its generalization ability. The goal is to find the hyperparameter values that result in the best model performance in terms of metrics such as accuracy, precision, recall, F1-score, or mean squared error, depending on the specific problem and evaluation criteria.

11) Having a large learning rate in gradient descent can lead to several issues, including:

    Overshooting: With a large learning rate, the updates to the model parameters can be too large, causing the optimization algorithm to overshoot the minimum of the loss function.
    Instability: A large learning rate can make the optimization process unstable, causing the parameter updates to oscillate around the minimum of the loss function rather than converging smoothly.
    Unpredictable Behavior: Large learning rates can lead to unpredictable behavior of the optimization algorithm, making it difficult to determine when the algorithm has converged to a satisfactory solution.
    Difficulty in Tuning: In some cases, a learning rate that is too large may cause the algorithm to diverge, while a learning rate that is too small may result in slow convergence or getting stuck in local minima.
    Difficulty in Escaping: a large learning rate may cause the optimization algorithm to get stuck or converge too slowly.

12) Logistic Regression is typically not suitable for handling non-linear relationships in the data. When the decision boundary between classes is non-linear, Logistic Regression may struggle to capture the underlying patterns effectively, leading to suboptimal performance. Here are a few reasons why Logistic Regression may not perform well for classification of non-linear data:

Linear Decision Boundary: Logistic Regression models assume that the decision boundary separating the classes is a linear function of the input features. This means it can only learn linear decision boundaries, such as straight lines or hyperplanes in higher dimensions. In scenarios where the true decision boundary is non-linear, Logistic Regression may fail to capture the complex relationships between the features and the target variable.

Underfitting: When the data contains non-linear relationships that cannot be adequately captured by a linear model, Logistic Regression may underfit the data.

Limited Expressiveness: Logistic Regression has limited expressiveness compared to more complex models such as decision trees, random forests, or neural networks.

13) Differences between Adaboost and Gradient Boosting

ADABOOST	**GRADIENT BOOSTING**
Adaboost trains a sequence of weak learners (typically decision trees with one level of depth, also known as stumps) sequentially.	Gradient Boosting trains a sequence of weak learners (often decision trees) sequentially as well,
At each iteration, the algorithm adjusts the weights of misclassified training instances to emphasize the difficult-to-classify examples in subsequent iterations.	Instead of adjusting the instance weights, it fits each new learner to the residuals (or gradients) of the predictions made by the ensemble of previous learners.
The final prediction is made by combining the predictions of all weak learners using a weighted majority vote.	The final prediction is the sum of the predictions made by all weak learners, with each prediction weighted by a shrinkage parameter (learning rate).
Focuses on reducing the overall error rate of the ensemble by emphasizing the instances that are difficult to classify correctly.	Focuses on improving the model's predictions by iteratively fitting weak learners to the residuals of the previous ensemble, moving towards the direction of steepest descent in the loss function space.
Adaboost adjusts the weights of misclassified instances to prioritize them in subsequent iterations, allowing the algorithm to focus on the hardest-to-classify examples.	Gradient Boosting assigns equal weights to all training instances but updates the predictions by fitting each new learner to the residuals of the previous ensemble.

14) The bias-variance tradeoff is a fundamental concept in machine learning that describes the balance between two sources of error, bias and variance, which affect the predictive performance of a model.

Bias refers to the error introduced by approximating a real-world problem with a simplified model. A model with high bias tends to oversimplify the underlying patterns in the data and may consistently underpredict or overpredict the target variable.

Variance refers to the variability in model predictions when trained on different subsets of the training data. A model with high variance is sensitive to small fluctuations in the training data and may produce significantly different predictions for different training sets.

15) Short description each of Linear, RBF, Polynomial kernels used in SVM

Linear Kernel:

The linear kernel is the simplest kernel function used in SVM.
It represents a linear decision boundary in the feature space.
The linear kernel is suitable for linearly separable datasets or datasets where a linear decision boundary is appropriate.

RBF (Radial Basis Function) Kernel:

The RBF kernel, also known as the Gaussian kernel, is a popular choice for SVM.
It maps the data into a higher-dimensional space using a non-linear transformation.
The RBF kernel is capable of capturing complex, non-linear relationships between the features.
It is defined by a single parameter, the kernel width (or gamma), which controls the smoothness of the decision boundary.
The RBF kernel is suitable for datasets with non-linear separable classes or when the decision boundary is non-linear.

Polynomial Kernel:

The polynomial kernel maps the data into a higher-dimensional space using polynomial functions.
It allows the SVM to capture non-linear relationships between the features.
The polynomial kernel is defined by two parameters: the degree of the polynomial and an optional coefficient term.
Higher degrees of the polynomial can capture more complex relationships but may also lead to overfitting.