1. What are the differences between hyperparameters and parameters of a machine learning (ML) model?

A: In machine learning, both parameters and hyperparameters play a crucial role, but they serve different purposes:

parameter:

These are internal variables that the model learns during training. They are tuned to minimize the error of the model. For example, in a linear regression model, the coefficients are parameters.
Hyperparameters:

These are external configuration values that you set before training starts. They determine the structure or behavior of the model. Examples include learning rate, regularization strength, and depth for decision tree models.

Use native models:

Decision trees: Parameters are the decision rules for each node, and hyperparameters may include the maximum depth of the tree or the minimum samples required to split a node.

K-Nearest Neighbors (KNN): The model itself does not have parameters like traditional algorithms because it relies on instance-based learning. However, the number of neighbors (k) is a hyperparameter.

2. Prove that Elastic net can be used as either LASSO or Ridge regularizes.

A:

An elastic net is basically a combination of both L1 and L2 regularization. So know elastic net, you can implement both Ridge and Lasso by tuning the parameters.

$$\min\left(\left|\left|Y - X\theta\right|\right|_2^2 + \lambda_1||\theta||_1 + \lambda_2||\theta||_2^2\right)$$

3. Analyse the importance of the features for predicting "class' using two different approaches. Explain the similarity/difference between outcomes.

A: We used two approaches:

Decision Tree Feature Importance: This gives important scores based on how often a feature is used to split the data. It focuses on features that best partition the dataset into its target classes.

Logistic Regression Coefficients: The magnitude of coefficients indicates the importance of features in predicting the target. Larger coefficients have a bigger impact on the model's output.

Similarity/Difference:

While both approaches rank features, the decision tree's ranking is based on data partitioning, while logistic regression's ranking is based on the linear relationship with the log odds of the outcome.

4.Is it possible that the presented result is an overfitted one? Justify.

A: Yes, overfitting occurs when the model performs very well on the training data but poorly on unseen data (like the validation or test set). It means the model has learned the training data noise along with the underlying pattern. Regularization, feature selection, and using simpler models can help mitigate overfitting.

5. Justify different design decisions for each ML model used to answer this question.

A: Decision Tree Classifier:

K nearest neighbor:

KNN is sensitive to feature scale, so we normalize the features before training. KNN will classify a data point based on the majority class of its k nearest neighbors.

Design Decision: KNN assumes that similar data points (based on a distance metric) are more likely to belong to the same class.

Logistic regression:

After standardizing the features, we will use logistic regression to predict the probability of binary outcomes. Design Decision: Assume a linear relationship between the feature and the log-odds of the target variable.

- *K-Nearest Neighbors (KNN):*

KNN is sensitive to feature scales, so we'd normalize the features before training.
KNN would classify data points based on the majority class of their k nearest neighbors.
Design Decision: KNN assumes that similar data points (based on distance metrics) are more likely to belong to the same class.

- *Logistic Regression:*

After normalizing the features, we'd use logistic regression which predicts probabilities of binary outcomes.
Design Decision: Assumes a linear relationship between the features and the log-odds of the target variable.

6. Have you optimized any hyper-parameters for each ML model? What are they? Why have you done that? Explain.

A: Hyper-parameter optimization:

For each model, certain hyperparameters would be critical:

- *Decision Tree: Depth of the tree, minimum samples to split, etc.*
- *KNN: Number of neighbors.*
- *Logistic Regression: Regularization strength and type.*

Optimizing these hyperparameters using techniques like cross-validation can lead to better generalization and model performance.

7. When do you want to use ensemble models over other ML models? Explain based on the models that you have used in Q4 and Q5

A: Better Performance: If one model isn't doing well, combining several can help. Think of it like taking advice from a group instead of just one friend.

Diverse Data: When data is like a mix of apples, bananas, and cherries, ensemble models are like a fruit salad, blending everything nicely.

Stability: Two heads are better than one. Similarly, using multiple models can help in making stable and reliable decisions.

In Practice:

A single tree can memorize and make mistakes. But a forest has many trees, so it's harder for all of them to make the same mistake.

KNN and Logistic Regression are like using a straight ruler to draw curves. They might miss some twists and turns. Ensemble models, like Gradient Boosting, can trace those curves more closely.

8. Similarities or differences among ensemble models used in Q5?

Similarities:

All are ensemble methods and typically use Decision Trees as base models.
They combine multiple models to make a final decision.

Differences:

Their approach to combining models: Random Forest uses bagging, while Gradient Boosting and AdaBoost use boosting.
The way they handle errors and build subsequent models is different.

9. Write a report comparing the performances of models built in question 4 and 5. Report the best method based on model complexity and performance.

A: Ensemble models are often better than individual models because they combine multiple models to capture complex patterns and reduce overfitting. In my analysis, while individual models such as decision trees, KNN, and logistic regression provided insights, ensemble methods such as random forests, gradient boosting, and AdaBoost provided better generalization capabilities.

10. Is there any preferable scenario for using any specific model among the set of ensemble models?

A: Random Forest: Best when you want an interpretable model that's resistant to overfitting.

Gradient Boosting: When performance is paramount, and you have the computational resources for fine-tuning.

AdaBoost: Useful when you have many weak learners and need to focus on challenging-to-classify instances.

Gradient Boosting: When performance is of utmost importance. Gradient Boosting often outperforms other algorithms but might require more tuning.

AdaBoost: When the dataset has many weak learners. AdaBoost focuses on the hard-to-classify examples, making it effective when there's a need for high precision.