True or False:

Ans 1) True: Pruning can make a decision tree smaller and sometimes even better at correctly classifying new, unseen data. While it might perform worse on the training data, its overall classification ability can improve.

Ans 2) In k-fold cross-validation, the dataset is divided into k roughly equal parts. Each part is used as the test set once while the model is trained on the remaining k-1 parts. This is repeated k times, not leaving out k inputs at each step.

Ans 3) Gradient Boosting works by adding models step by step. Each new model tries to fix the errors made by the previous models using gradients, similar to how gradient descent works.

Ans 4): While splitting the data randomly is common, stratified splitting is often used for classification to make sure both training and test sets have a similar distribution of classes. Random splitting is not the only or always the best method.

Ans 5) If a model makes no mistakes on the training data, it might be overfitting. However, it’s very unlikely to make 100% mistakes on the test data because some patterns will usually be similar between training and test sets.

Ans 6) Increasing regularization makes a model simpler, which can lead to underfitting. This means the model's bias increases, not decreases.

Ans 7) The ID3 algorithm without a depth limit can create a very large, overfitted tree, not necessarily a compact one.

Ans 8) All classifiers make some assumptions about the data. The "best" classifier depends on how well these assumptions match the true nature of the data.

Ans 9) Random Forests use a method called bagging to create many decision trees from random subsets of the data. They also add extra randomness when building the trees. This helps reduce the overall error by averaging the results of all the trees.

Ans 10) The k-nearest neighbor algorithm has an error that, at worst, is no more than twice the Bayes error as the dataset size becomes very large.

Ans 11) Squared loss regression trees generally have a time complexity of O(n log n) per split, similar to other decision tree algorithms.

Ans 12) The Bayes optimal error is the lowest possible classification error given the true probability distributions. It includes noise but is not only due to noise.

**ERM and SVM**

1)SVM (Support Vector Machine)

Loss Function: Hinge Loss

Description: Measures how far the prediction is from the correct classification.

Regularizer: L2 Regularization

Description: Penalizes large weights to prevent overfitting.

LASSO (Least Absolute Shrinkage and Selection Operator)

Loss Function: Mean Squared Error (MSE)

Description: Measures the average squared difference between predicted and actual values.

Regularizer: L1 Regularization

Description: Adds absolute values of weights, encouraging some weights to be zero.

Ridge Regression

Loss Function: Mean Squared Error (MSE)

Description: Measures the average squared difference between predicted and actual values.

Regularizer: L2 Regularization

Description: Penalizes large weights to prevent overfitting.

2)

3) Gradient Descent

Least Squares Loss: Works well for minimizing this loss in both LASSO and Ridge Regression.

Hinge Loss: Can be used, but needs careful handling.

Newton’s Method

Least Squares Loss (Ridge Regression): Works efficiently for minimizing this loss.

Hinge Loss and L1 Regularization (LASSO): Not easily used because they are not smooth and can be complex to compute.

**Bias and Variance**

1. High Training and Testing Errors with Limited Depth

* High Bias: The decision trees are too simple (shallow) to capture the complexity of the stock market data.
* Irrelevant Features: The features used might not be useful for predicting stock prices.

1. High Errors with Unlimited Depth

* Noisy Data: The stock market data might be very noisy, making it hard to find useful patterns.
* Overfitting with Limited Data: If there isn't enough training data, even deep trees can't generalize well.

1. Bagging

* Simulation: Bagging simulates using multiple different datasets by creating random samples (with replacement) from the original dataset.
* Variance Reduction: It reduces variance by averaging the predictions from many trees, smoothing out the fluctuations from any single tree.

1. Boosting Shallow Trees

* Bias: Boosting reduces bias over time as it combines many weak models into a stronger one.
* Variance: Initially, variance might increase slightly, but overall, boosting helps in reducing variance by making the model more robust.

**kNN / Curse of Dimensionality**

1. Speeding Up kNN Classifier

* Dimensionality Reduction: Use PCA or t-SNE to lower the number of features.
* Approximate Nearest Neighbours: Use LSH or KD-trees to quickly find nearest neighbours.
* Efficient Data Structures: Use Ball Trees or KD-trees for faster searching.
* Parallelization: Use multiple processors to split the work.
* Data Preprocessing: Normalize or scale data for easier distance calculations.

2. Impact of Using Squared Euclidean Distance

* kNN Accuracy: Using squared Euclidean distance doesn’t change which neighbours are closest. The ranking of distances stays the same.
* b) Validity of Advice: Yes, the suggestions in question 1 still apply. The methods work with squared Euclidean distance too.

3. Curse of Dimensionality

Concern: Yes, Kanye should worry. With very high dimensions (d = 1,000,000) and few data points (n = 5,000), data points become sparse, making it hard for kNN to find meaningful neighbours, leading to poor performance.

4. Neighbourhood Size k Effects

* Bias: Increases with larger k. The model becomes more general, smoothing over details.
* Variance: Decreases with larger k. The model is less sensitive to specific data points.
* Trade-off: Small k means low bias but high variance. Large k means high bias but low variance.

5. Choosing Between kNN and Linear SVM

* kNN Classifier: Best for small datasets with complex, non-linear patterns. Example: Medical diagnosis with few records and complex symptom relationships.
* Linear SVM Classifier: Best for large datasets with linear or near-linear patterns. Example: Text classification like spam detection, where the data is high-dimensional but roughly linear.

**Decision Matrix:**

1. Prediction Value at a Leaf of a Regression Tree

Prediction Value: The prediction value at a leaf of a regression tree with squared-loss impurity is the mean (average) of the target values of the data points in that leaf.

Proof of Optimality: This is optimal because the mean minimizes the squared error.

2. Gini Index for 3 Classes.

Maximized: The Gini Index is maximized when all classes are equally probable. For 3 classes, this happens when p1=p2=p3=1/3.

Minimized: The Gini Index is minimized when all items are of a single class, i.e., when pk​=1 for one class and 0 for others.

3. Decision Trees are "Myopic"

Explanation: Decision trees are considered "myopic" because they make decisions based on the best split at each node without considering the impact on future splits. They optimize locally at each node rather than globally over the entire tree.

4. Preventing Overfitting in Decision Trees

Method 1: Pruning: Remove branches that have little importance and don't contribute significantly to the accuracy on the validation set.

Method 2: Limiting Tree Depth: Set a maximum depth for the tree to prevent it from growing too complex and capturing noise in the training data.

**Boosting and Bagging**

1. Random Forests and Validation/Test Data

Claim: Ludwig van Beethoven says he doesn't need validation or test data for Random Forests.

Answer: False.

Reason:

Generalization: Without validation/test data, you can't ensure the model works well on new data; it might overfit.

Parameter Tuning: Parameters (like tree count and depth) need tuning using validation data.

Cross-Validation: Techniques like k-fold cross-validation split data to tune parameters and check performance.

Conclusion: Exponential loss LLL is an upper bound on training error because it penalizes misclassified points and those near the decision boundary.