Theoretical Assignment

**True or False (Mention reason)**

Ans 1 **True**: Pruning can not only significantly reduce the size but also improve the classification accuracy of unseen objects. It may be the case that the accuracy of the assignment on the train set deteriorates, but the accuracy of the classification properties of the tree increases overall.

Ans 2 **False**: In k-fold cross-validation, the dataset is split into k approximately equal-sized parts. Each part is used as the test set once while the classifier is trained on the remaining k-1 parts. This process is repeated k times, not leaving k inputs out at each step.

Ans 3 **True**: Gradient Boosting involves stage-wise additive modeling where each new model attempts to correct the errors of the previous models by using gradients, which is akin to performing gradient descent in function space.

Ans 4 **False**: While uniform random splitting is common, stratified splitting is often used for classification tasks to ensure that both training and test sets have a similar distribution of classes. Random splitting is not the only method and sometimes not the best.

Ans 5 **True**: If a classifier has 0% training error, it perfectly fits the training data. While it might perform poorly on test data due to overfitting, obtaining 100% testing error is highly unlikely because some patterns from the training data would likely be present in the test data.

Ans 6 **False**: Increasing regularization generally increases the bias of a classifier by simplifying the model and reducing its complexity, which can lead to underfitting.

Ans 7 **False**: The ID3 algorithm without a depth limit tends to create a very large, potentially overfitted tree, not necessarily the maximally compact tree.

Ans 8 **False**: All classifiers make some assumptions about the data (e.g., linearity, independence, distribution) which guide their learning process. The notion of "best" classifiers often depends on how well these assumptions align with the data's true characteristics.

Ans 9 **True**: Random Forests use a technique called bagging, where they create many different decision trees from random subsets of the data. They also add extra randomness when making the trees. This helps to reduce the overall error by averaging the results of all the trees.

Ans 10 **True**: This statement relates to the property of the k-nearest neighbour algorithm where its error approaches no more than twice the Bayes error as the dataset size grows infinitely large.

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

Ans 12 **False**: The Bayes optimal error is the best possible classification error considering the underlying true probability distributions. It represents the irreducible error, which includes noise but is not solely due to noise.

**ERM and SVM**

1.The names of loss functions and regularizers for SVM, LASSO, and Ridge Regression are:

**SVM**:

* Loss Function: Hinge Loss
* Regularizer: L2 Regularization

**LASSO**:

* Loss Function: Least Squares Loss
* Regularizer: L1 Regularization

**Ridge Regression**:

* Loss Function: Least Squares Loss
* Regularizer: L2 Regularization

3. **Gradient Descent**:

* Can be used without modifications for minimizing the **Least Squares Loss** (both LASSO and Ridge Regression).
* Can be used (with care) for minimizing the **Hinge Loss** (SVM).

 **Newton’s Method**:

* Can be efficiently used for minimizing the **Least Squares Loss** (Ridge Regression).
* Is not straightforwardly applicable to **Hinge Loss** (SVM) and **Least Squares Loss with L1 Regularization** (LASSO) due to non-differentiability and computational complexity.

**Bias and Variance**

**1. Scrooge McDuck trains decision trees of limited depth d, but both training and testing errors are high. Name two possible explanations for the problem.**

Possible explanations could include:

* **Underfitting:** Decision trees with limited depth may not capture the complexity of the relationship between features and stock market prices. They may be too simplistic to learn meaningful patterns from the data, leading to high errors both in training and testing.
* **Insufficient Features:** The features (or predictors) Scrooge is using may not be informative enough to predict stock market prices accurately. Without relevant features that correlate well with the target variable, the decision trees may struggle to generalize effectively.

**2. Scrooge now trains trees with unlimited depth but still observes high train and test errors. What explanation can you give him?**

The likely explanation is:

* **Overfitting:** Decision trees with unlimited depth can perfectly fit the training data by memorizing it, resulting in very low training error. However, they generalize poorly to unseen data (test set) because they capture noise and outliers in the training data as well, leading to high test error. Overfitting occurs when the model is too complex relative to the amount of training data available.

**3. What does Bagging simulate and why would it reduce variance?**

Bagging (Bootstrap Aggregating) simulates averaging multiple models trained on different subsets of the data (bootstrap samples). It reduces variance by:

* **Diversifying Models:** Each bootstrap sample is likely to contain different subsets of the training data, leading to diverse models. These models may make different errors on different parts of the data.
* **Averaging Predictions:** By averaging the predictions of these diverse models, Bagging reduces the overall variance because individual errors tend to cancel out when averaged over multiple models.

**4. You boost decision trees with very limited depth (depth = 2). How are bias and variance affected as the boosting iterations increase?**

Boosting with decision trees of very limited depth (depth = 2) typically results in:

* **Decreasing Bias:** Boosting focuses on improving the model's ability to fit the data, reducing bias by iteratively correcting errors made by previous models. The base learners (trees of depth 2) are weak, but boosting increases their complexity over iterations, reducing bias.
* **Potentially Increasing Variance:** While boosting aims to reduce variance by combining multiple models, increasing the number of boosting iterations can sometimes lead to overfitting. This is because boosting iteratively fits the residuals (errors) of the previous model, which can lead to models that capture noise and result in increased variance on the test set.

**kNN / Curse of Dimensionality**

1. To speed up her kNN classifier during test time, Kim K. can consider the following approaches:

* **Dimensionality Reduction**: Techniques like Principal Component Analysis (PCA) or t-SNE can reduce the number of dimensions while retaining most of the relevant information, thus reducing the computational load.
* **Approximate Nearest Neighbours**: Use approximate nearest neighbours algorithms like Locality-Sensitive Hashing (LSH) or KD-trees, which can significantly speed up the nearest neighbour search.
* **Efficient Data Structures**: Implement efficient data structures like Ball Trees or KD-trees to speed up the search process.
* **Parallelization**: Use parallel processing to divide the computation across multiple processors.
* **Data Preprocessing**: Normalize or scale the data to potentially simplify the distance computations.

a) **Impact on kNN Accuracy**: - The missing property of the triangular inequality due to using squared Euclidean distance does not affect the kNN accuracy. The ranking of distances (which neighbours are closer) remains the same because the square root operation is monotonic. The relative distances between points do not change, so the same neighbours are chosen whether using the Euclidean or squared Euclidean distance.

b) **Validity of Advice with Squared Distance**: - Yes, the answer to question 1 is still valid. Techniques such as dimensionality reduction, approximate nearest neighbours, and efficient data structures can still be applied when using squared Euclidean distance. The squared distance simply changes the specific distance measure but does not affect the applicability of these methods for speeding up the classifier.

1. **Curse of Dimensionality**:

* Yes, Kanye W. should be worried about the curse of dimensionality. In high-dimensional spaces, the volume of the space increases exponentially, and the data points tend to become sparse. This sparsity makes it difficult for the kNN classifier to find meaningful neighbours, as the distance between points becomes less informative. With d=1000000d = 1000000d=1000000 dimensions and only n=5000n = 5000n=5000 images, the distance between points might not effectively capture the similarity, leading to poor classification performance.

1. **Neighbourhood Size kkk Effects**:

* **Bias**: As kkk increases, the bias of the classifier increases. A larger kkk means the classifier averages over more points, potentially smoothing over the decision boundary and leading to a more generalized model.
* **Variance**: As kkk increases, the variance of the classifier decreases. A larger kkk reduces the impact of any single training point on the prediction, making the classifier less sensitive to fluctuations in the training data.
* **Trade-off**: There is a bias-variance trade-off when selecting kkk. A small kkk results in low bias but high variance, while a large kkk results in high bias but low variance.

1. **Scenario for kNN vs. Linear SVM**:

* **kNN Classifier**: Use kNN when you have a small dataset with a complex and non-linear decision boundary. kNN is non-parametric and can capture complex relationships without assuming a linear decision boundary. For example, in a medical diagnosis problem with a small number of patient records and non-linear relationships between symptoms.
* **Linear SVM Classifier**: Use a linear SVM when you have a large dataset with a clear linear or nearly linear decision boundary. SVMs are effective in high-dimensional spaces and can handle large datasets efficiently. For example, in text classification tasks (e.g., spam detection), where the data is high-dimensional but the decision boundary is approximately linear.

**Decision Trees**

### Prediction value at a leaf of a regression tree (with squared-loss impurity):

In a regression tree, the prediction value at a leaf node mmm is the average of the target values of the training instances that belong to that leaf node:

**Optimality:**

This prediction value​ is optimal in the sense that it minimizes the squared-loss impurity criterion (often used in regression trees). The squared-loss impurity at leaf node mmm is defined as:

To prove optimality:

* **Minimization of Squared Loss:** The prediction ​ is chosen to minimize the sum of squared differences between the predicted values and the actual target values of training instances in leaf node mmm.
* **Optimal Split Criterion:** During tree construction, nodes are split to minimize impurity (such as squared-loss impurity). The prediction value​ at a leaf node mmm is derived from the training data such that it minimizes this impurity measure, ensuring that the predicted values are as close as possible to the actual values in that leaf.

### Given the definition of the Gini Index of a set, when is G(S) maximized, when minimized? (no derivation necessary)

* **Maximized:** The Gini Index G(S) is maximized when the probabilities pk are as evenly distributed as possible among the classes. This happens when all pk are equal, i.e., pk=1/ c for all k. In this case, G(S) reaches its maximum value of 1−1/c.
* **Minimized:** The Gini Index G(S) is minimized when one of the probabilities pk is 1 (indicating all items belong to the same class) and the rest are 0. In this scenario, G(S)=0G(S) = 0G(S)=0.

### 3. Decision trees are “myopic”:

Decision trees are "myopic" because they make locally optimal decisions at each node without considering the global optimal solution for the entire tree:

* At each node, a decision tree chooses the best split based on a local criterion (like Gini Index or information gain).
* The tree grows recursively, making each decision independently of future nodes.
* Once a split is made, the decision is final and not reconsidered.

This myopic behaviour can lead to suboptimal overall structures if local decisions do not contribute to the best possible global structure of the tree.

### 4. Preventing overfitting in decision trees:

Two effective methods to prevent overfitting in decision trees are:

* **Tree Pruning:** After a tree is built, pruning involves removing unnecessary branches that do not significantly improve predictive accuracy on validation data. This helps prevent the model from fitting noise in the training data.
* **Limiting Tree Complexity:** Setting constraints such as maximum tree depth or minimum samples per leaf node helps control the complexity of the model. Shallower trees or larger leaf sizes reduce the risk of overfitting by restricting the tree's ability to capture noise and outliers in the training set.

**Boosting and Bagging**

### 1. Ludwig van Beethoven claims that when he trains a Random Forests classifier he does not need any validation or test data and can train the classifier on the entire data set, even to select model parameters? Is this true/false? Justify your answer.

**False.** Ludwig van Beethoven's claim is false. Here's why:

* **Validation and Test Data Importance:** It's crucial to evaluate a classifier's performance on unseen data (validation or test set) to ensure it generalizes well to new observations. Training a Random Forest on the entire dataset without validation or test data can lead to overfitting. Overfitting occurs when the model learns the noise and specific details of the training data, which reduces its ability to generalize to new data.
* **Model Parameter Selection:** Random Forests have parameters (such as the number of trees, tree depth, etc.) that can significantly affect performance. These parameters should ideally be tuned using validation data to find the best settings that generalize well. Using the entire dataset for parameter selection can lead to models that perform well on training data but poorly on unseen data.
* **Cross-Validation:** Typically, cross-validation (like k-fold cross-validation) is used to tune model parameters and assess performance. This involves splitting the data into multiple subsets, training on some and validating on others, to obtain an unbiased estimate of model performance.