UNIT 3 ASSIGNMENT

Understanding the Mechanics of   
ML Algorithms

## Instructions

The questions below will prepare you for future interviews as they relate to concepts discussed throughout the unit. You’ve practiced these concepts in the coding activities, exercises, and coding portion of the assignment; let’s now formulate your programming into well-reasoned responses.

Except as indicated, use this document to record all your assignment work and responses to any questions. At a minimum, you will need to turn in a digital copy of this document to your facilitator   
as part of your assignment completion. You may also have additional supporting documents that   
you will need to submit. Your facilitator will provide feedback to help you work through your findings.

**Note:** Though your work will only be seen by those grading the course and will not be used or   
shared outside the course, you should take care to obscure any information you feel might be   
of a sensitive or confidential nature.

*Begin your assignment by completing the questions below. Directions to submit your work can be found on the assignment page. Information about the grading rubric is available on any of the course assignment pages online. Do not hesitate to contact your facilitator if you have any questions about the assignment.*

Unit 3 Written Portion

# Building and Evaluating a Model

Answer the questions below about building and evaluating your models using algorithms such as decision trees and k-nearest neighbors.

## Questions:

1. What are the advantages and disadvantages of decision trees?

|  |
| --- |
| Advantages:   * Easy to understand and interpret, as the decision tree structure can be visualized. * Can handle both numerical and categorical data. * Can handle missing values and outliers. * Decision trees can capture non-linear relationships between features. * Can be used for classification and regression tasks.   Disadvantages:   * Decision trees are prone to overfitting, especially when the tree becomes too deep or complex. * They can be sensitive to small changes in the data, leading to different tree structures. * Decision trees may not generalize well to unseen data if they are too specific to the training set. * They can create biased trees if the training data is imbalanced or has class-specific biases. * Decision trees are not suitable for problems with high-dimensional feature spaces. |

1. What are the advantages and disadvantages of k-nearest neighbors?

|  |
| --- |
| Advantages:   * KNN is a simple and intuitive algorithm that is easy to understand and implement. * It can handle both classification and regression tasks. * KNN can capture complex relationships in the data, including non-linear patterns. * KNN does not make any assumptions about the underlying data distribution. * It can be used for multi-class classification tasks.   Disadvantages:   * KNN can be computationally expensive, especially with large datasets or high-dimensional feature spaces. * The choice of the K value (number of neighbors) is crucial and can affect the performance of the model. * KNN is sensitive to the scale of the features and may require feature scaling or normalization. * The decision boundary of KNN can be irregular and sensitive to noise or outliers in the data. * KNN may struggle with imbalanced datasets where one class dominates. |

1. Explain the difference between k-nearest neighbors and decision trees. When would you decide to use one over the other?

|  |
| --- |
| K-Nearest Neighbors (KNN) is a non-parametric lazy learning algorithm that relies on the similarity between instances to make predictions. It classifies a new data point based on the majority class of its nearest neighbors.  Decision Trees are a non-parametric supervised learning algorithm that partitions the feature space based on a set of if-else conditions, resulting in a tree-like structure that can be used for classification or regression.  When to use KNN:   * KNN is suitable when the decision boundary is expected to be complex or nonlinear. * It can work well with small to medium-sized datasets. * KNN is effective when the training data represents the underlying data distribution well. * It can be used in situations where the feature importance or interpretability of the model is not a primary concern.   When to use Decision Trees:   * Decision trees are useful when interpretability and explainability of the model are important. * They can handle both numerical and categorical features and automatically handle missing values. * Decision trees are efficient for large datasets and can handle high-dimensional feature spaces. * They are effective when the relationships between features and the target variable are nonlinear and hierarchical.   The choice between KNN and Decision Trees depends on the specific problem, dataset characteristics, interpretability requirements, and the underlying relationships in the data. It is recommended to experiment with both algorithms and evaluate their performance to determine the most suitable approach for a given task. |

1. What are hyperparameters? List some hyperparameters in k-nearest neighbors and decision trees.

|  |
| --- |
| Hyperparameters are parameters that are not learned from the data but are set prior to the training process in machine learning algorithms. These parameters control the behavior and performance of the algorithm and need to be specified by the user.  K-Nearest Neighbors:   * n\_neighbors: The number of nearest neighbors to consider for classification or regression. * weights: The weight function used in prediction, such as uniform weights or weights based on distance.   Decision Trees:   * criterion: The function used to measure the quality of a split in the decision tree, such as 'gini' or 'entropy'. * max\_depth: The maximum depth of the decision tree. * min\_samples\_split: The minimum number of samples required to split an internal node. |

1. What is overfitting? How can you avoid overfitting? Give examples using a model discussed   
   so far.

|  |
| --- |
| Overfitting occurs when a machine learning model performs extremely well on the training data but fails to generalize well to new, unseen data. In other words, the model has learned the noise or random fluctuations in the training data rather than the underlying patterns or relationships. Overfitting often leads to poor performance and inaccurate predictions on new data.  To avoid overfitting, some common strategies include:   * Increasing the amount of training data: Having more diverse and representative data can help the model to learn the underlying patterns better and reduce the chances of overfitting. * Feature selection/reduction: Selecting or extracting relevant features and reducing the dimensionality of the dataset can help in reducing overfitting. It focuses on the most informative features and avoids unnecessary noise or irrelevant information. * Regularization: Regularization techniques introduce a penalty term to the model's objective function, discouraging complex or extreme parameter values. This helps to control the model's flexibility and prevents it from fitting the noise in the training data. * Cross-validation: Cross-validation helps to estimate the performance of a model on unseen data. It involves splitting the data into multiple folds and training/evaluating the model on different combinations of folds. This allows for a more robust assessment of the model's performance and can help in identifying overfitting.   Example using Decision Trees:  In the context of decision trees, overfitting can occur when the tree becomes overly deep or complex, capturing noise or specific details of the training data. To avoid overfitting in decision trees, we can use techniques such as:   * Limiting the maximum depth of the tree (max\_depth hyperparameter). * Setting a minimum number of samples required to split an internal node (min\_samples\_split hyperparameter). * Specifying a minimum number of samples required to be at a leaf node (min\_samples\_leaf hyperparameter). * Pruning the tree by removing branches that do not significantly improve the model's performance.   By using these techniques, we can control the complexity of the decision tree and prevent it from overfitting the training data, leading to improved generalization and better performance on new data. |

1. What is the purpose of splitting data into different sets?

|  |
| --- |
| Here are the main purposes of splitting data into different sets:  Training the model: The training set is used to train the machine learning model. It contains labeled examples (input features and corresponding target labels) that are used to learn the underlying patterns and relationships in the data.  Evaluating model performance: The test set, which is separate from the training set, is used to evaluate the performance of the trained model. By applying the trained model to the test set and comparing the predicted outputs with the actual labels, we can assess how well the model generalizes to unseen data.  Assessing model generalization: Splitting the data into separate sets helps us understand how well the model generalizes to new, unseen data. If the model performs well on the test set, it indicates that it has learned the underlying patterns rather than memorizing the training data. This provides an estimate of the model's performance on real-world data.  Preventing overfitting: Splitting the data into separate sets allows us to detect and prevent overfitting. By training the model on the training set and evaluating it on the test set, we can identify if the model is overfitting the training data. If the model performs significantly worse on the test set compared to the training set, it suggests overfitting, and adjustments to the model or its hyperparameters may be needed. |

*To submit this assignment, please refer to the instructions in the course*.