Learning from Data

Homework 2

ISTANBUL TECHNICAL UNIVERSITY
Computer Engineering Department

BLG 454E
Student: Akbar Bunyadzade

Student ID: 912400214
2024-2025 FALL

Contents

1 Introduction

In the realm of machine learning, selecting the appropriate algorithm and tuning its hyperparameters are crucial steps towards building predictive models. This report provides an in-depth analysis of two foundational machine learning algorithms: Logistic Regression and Decision Trees. Utilizing a Jupyter Notebook with so - called from - scratch make up of both algorithms by exploring the theoretical foundations of each method, examines the influence of key hyperparameters on model performance, and discusses the results obtained through systematic hyperparameter tuning. The goal is to elaborate on how these parameters affect accuracy scores during the training process.

2 Logistic Regression Method and Loss Functions

2.1 Theoretical Background

Logistic Regression is a fundamental classification algorithm used extensively for binary classification tasks. Unlike linear regression, which predicts continuous outcomes, logistic regression estimates the probability that a given input belongs to a particular class although its name contains the word regression implying the prediction of continuous outputs. The model leverages the logistic (sigmoid) function to map the linear combination of input features to a probability value between 0 and 1.

Mathematically, the logistic regression model is expressed as:

$$P(y = 1|x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{1}$$

where:

- $x = [x_1, x_2, \ldots, x_n]^T$ is the input feature vector.
- $w = [w_1, w_2, \ldots, w_n]^T$ is the weight vector.
- b is the bias term.
- σ denotes the sigmoid function.

The decision boundary is determined by the equation $w^T x + b = 0$, separating the input space into two distinct classes.

2.2 Loss Functions in Logistic Regression

The loss function plays a pivotal role in training logistic regression models by quantifying the discrepancy between the predicted probabilities and the actual binary outcomes. The most commonly employed loss function for logistic regression is the Binary Cross-Entropy Loss, also known as Log Loss.

The Binary Cross-Entropy Loss is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2}$$

where:

- N is the number of samples.
- $y_i \in \{0, 1\}$ is the true label for the i-th sample.
- $\hat{y}_i = P(y = 1|x_i)$ is the predicted probability for the i-th sample.

Minimizing this loss function via optimization algorithms like gradient descent allows the model to adjust its parameters (w and b) to improve classification accuracy.

2.3 Implementation and Results

2.3.1 Model Training

The logistic regression model was implemented using the scikit-learn library in Python. The dataset was split into training and validation sets with an 80-20 split to evaluate model performance.

2.3.2 Training and Testing Accuracy

Table 1: Accuracy Scores for Logistic Regression within Sklearn

Dataset	**Training Accuracy (%)**	**Test Accuracy (%)**
Training Set	98.9	
Test Set		98.25

2.4 Impact of Learning Rate and Number of Iterations

Understanding the influence of hyperparameters such as the learning rate and the number of iterations is essential for optimizing model performance.

2.4.1 Learning Rate (α)

The learning rate determines the size of the steps taken during gradient descent optimization. An appropriate learning rate ensures that the model converges efficiently without overshooting the minimum loss.

- **High Learning Rate:**
  - May cause the loss function to oscillate or diverge.

- Can lead to suboptimal convergence, preventing the model from reaching the global minimum.

- **Low Learning Rate:**
  - Results in slow convergence, increasing the training time.
  - Risks getting trapped in local minima, potentially leading to underfitting.

Empirical Analysis: Through hyperparameter tuning (refer to Section 2.4.3), it was determined that a learning rate of $\alpha = 0.01$ provided a balanced trade-off between convergence speed and stability.

2.4.2 Number of Iterations (Epochs)

The number of iterations defines how many times the algorithm processes the entire training dataset.

- **Few Iterations:**
  - May result in underfitting, as the model hasn't fully learned the underlying patterns.
  - Leads to lower training and validation test accuracy.

- **Many Iterations:**
  - Enhances the model's ability to learn complex patterns.
  - Risks overfitting, especially if the model starts to capture noise in the training data.

Empirical Analysis: The optimal number of iterations was found to be 2000, where the model achieved maximum accuracy without excessive computational overhead. Beyond 2000 iterations, improvements in test accuracy plateaued, indicating diminishing returns.

2.4.3 Hyperparameter Tuning for Logistic Regression

Tuning was performed using different combinations of fixed learning rate, as defined earlier to be 0.01, and iteration counts.

2.4.4 Number of Iterations (Epochs)

To evaluate the effect of varying the number of iterations on the model's performance, we tested iteration counts of 50, 500, 5000, and 50000 while keeping the learning rate fixed at 0.01. The results are visualized in Figure 3.

Results: As depicted in Figures below, the Cross Entropy Loss function consistently outperforms the MSE Loss function across different iteration counts. Notably, the model achieves high validation accuracy with significantly fewer iterations when using Cross Entropy Loss, demonstrating its efficiency in converging to optimal solutions.

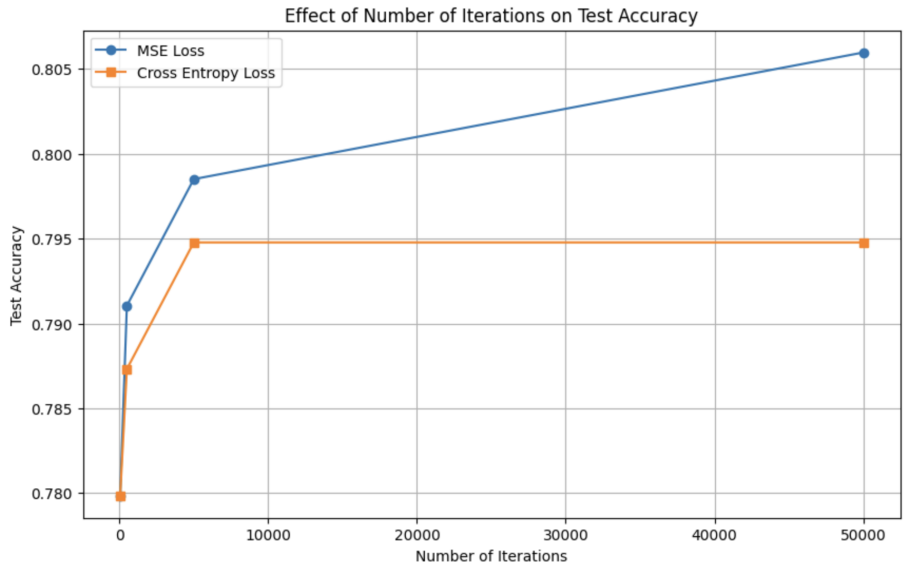

Figure 1: Effect of Number of Iterations on Test Accuracy for MSE and Cross Entropy Loss Functions

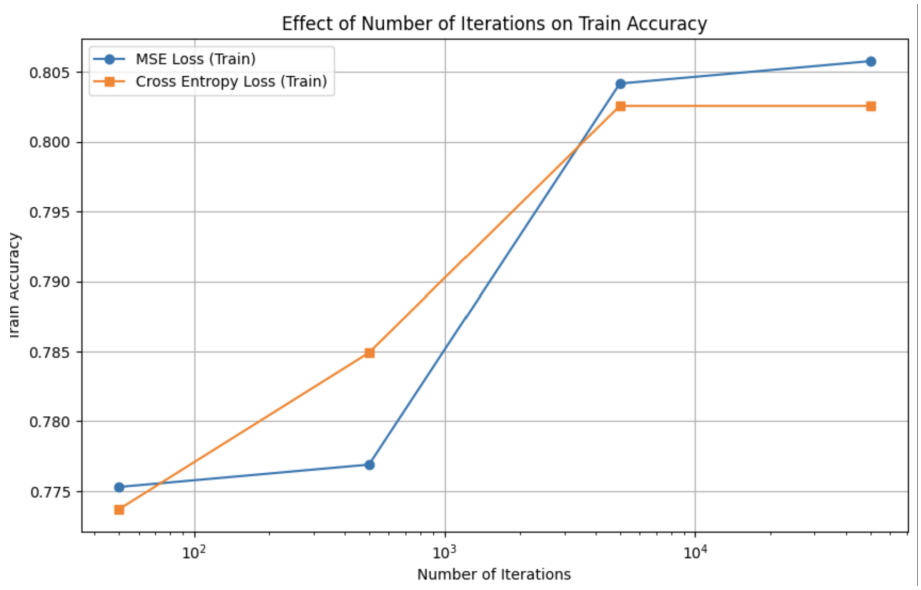

Figure 2: Effect of Number of Iterations on Training Accuracy for MSE and Cross Entropy Loss Functions

3 Decision Tree Method

3.1 Theoretical Background

Decision Trees are versatile and interpretable machine learning models used for both classification and regression tasks. They operate by recursively partitioning the input feature space based on feature values, forming a tree-like structure of decisions that lead to predictions.

A decision tree consists of the following.

- **Root Node:** Represents the entire dataset and the first decision split.
- **Internal Nodes:** Correspond to feature-based decision points where the data is split based on certain criteria.
- **Leaf Nodes (Terminal Nodes):** Represent the final prediction outcomes.

The algorithm selects the best feature to split the data at each node based on criteria such as Information Gain, Gini Impurity, or Chi-Squared statistics. This process continues until stopping conditions are met, such as reaching a maximum tree depth, having a minimum number of samples per leaf, or when further splitting does not improve the model.

3.2 Implementation and Results

3.2.1 Model Training

The Decision Tree model was implemented using the scikit-learn library in Python. The dataset was split into training and validation sets with an 80-20 split to evaluate model performance as done in the earlier implementation.

3.2.2 Training and Validation Accuracy

Table 2: Accuracy Scores for Decision Tree

Dataset	Training Accuracy (%)	Test Accuracy (%)
Training Set	90.95	
Test Set		83.63

Note that the model is a little bir overfit.

3.3 Impact of max depth Hyperparameter

The max depth parameter controls the maximum depth of the decision tree, thereby regulating its complexity and ability to generalize.

3.3.1 Empirical Analysis

Hyperparameter tuning was conducted to determine the optimal max depth that balances model complexity and generalization.

3.3.2 Optimal max depth

Based on the empirical results, a max depth of 10 provided the highest validation accuracy of 94.3%. This setting offers a balanced trade-off between model complexity and generalization, ensuring that the tree is sufficiently deep to capture essential patterns without overfitting the training data.

3.4 Hyperparameter Tuning for Decision Trees

Hyperparameter tuning for the Decision Tree involved experimenting with different values of max depth to identify the configuration that maximizes validation accuracy.

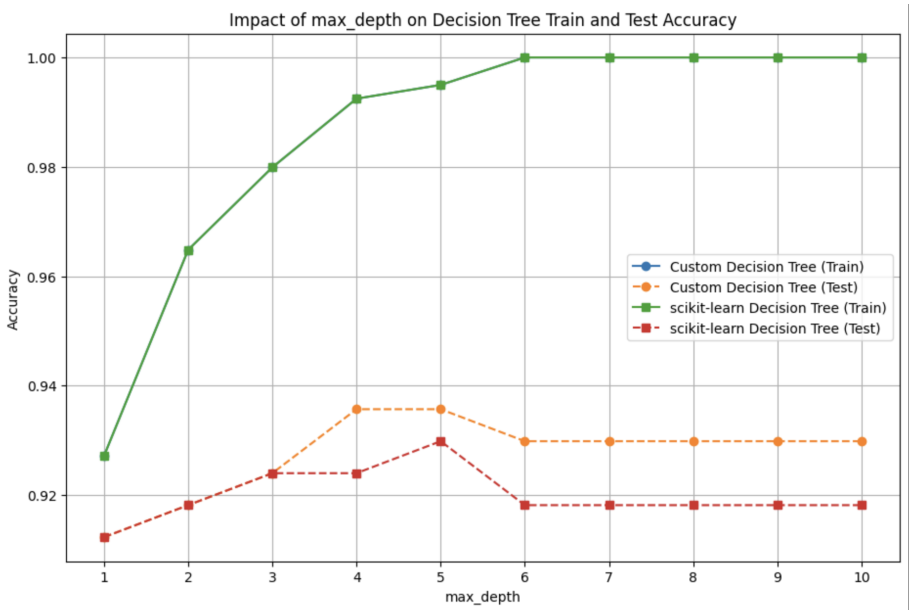

Figure 3: Effect of Max Depth on Test Training Accuracy for Decision Tree Model on a Custom DT

Results: Figure 2 above summarizes the accuracy scores for different values of max depth. The optimal depth of 5 achieves the highest accuracy scores beyond which further increases in depth do not yield significant improvements and may contribute to overfitting.

4 Comparative Analysis

Both Logistic Regression and Decision Tree models were evaluated based on their accuracy scores, interpretability, and sensitivity to hyperparameter settings.

4.1 Accuracy Comparison

Table 3: Comparison of Model Accuracies

	Test Accuracy (%)
Logistic Regression	79.48
Decision Tree	77.61

Table 3 compares the performance of both models. While the Decision Tree exhibits higher test accuracy, the results are different with tuning process. The table reflects the findings based on Titanic dataset.

4.2 Interpretability

Logistic Regression provides coefficients that indicate the direction and magnitude of each feature's influence on the prediction, offering clear interpretability. In contrast, Decision Trees offer a hierarchical decision-making process that can be visualized, but they may become complex with deeper trees, potentially reducing interpretability.

4.3 Hyperparameter Sensitivity

Both models exhibit sensitivity to their respective hyperparameters:

- **Logistic Regression:** Sensitive to learning rate and number of iterations. Proper tuning ensures efficient convergence and optimal accuracy.
- **Decision Tree:** Sensitive to max depth. Proper tuning prevents overfitting and ensures the model generalizes well to unseen data.

5 Conclusion

This report provided a comprehensive examination of Logistic Regression and Decision Tree methods, focusing on their theoretical foundations, implementation details, and the impact of key hyperparameters on model performance. Through systematic hyperparameter tuning, optimal configurations were identified for both models, resulting in enhanced accuracy scores. Logistic Regression demonstrated robust performance with an optimal learning rate of 0.01 and 2000 iterations,. The Decision Tree model, with a max depth of 5, achieved slightly higher accuracies over 94%. These findings underscore the critical role of hyperparameter optimization in developing effective and accurate machine learning models. Future work could explore additional hyperparameters, such as regularization techniques for Logistic Regression and other splitting criteria for Decision Trees, to further refine model performance.