Introduction to Machine Learning

Project 1 - Classification Algorithms

Université de Liège
26 October 2024
Gilles Ooms (s192136) ; Martin Dengis (s193348) ; Louis Blaimont (s225747)

1 Decision Tree

1.1 Decision Boundary Analysis

1.1.1 Illustration and Explanation of the Decision Boundary for Each Hyperparameter Value

The decision boundary plots for each value of max depth are presented in Appendix A. Table 1 provides performance metrics for each depth setting.

Max Depth	Accuracy	Precision	Recall	F1-Score
1	0.866	0.800	0.616	0.696
2	0.900	0.768	0.862	0.812
4	0.914	0.819	0.844	0.832
8	0.897	0.786	0.810	0.798
None	0.889	0.782	0.774	0.778

TABLE 1 – Performance Metrics for Different Max Depth Values

Max Depth 1 : The decision boundary is a single vertical line, dividing the space into two regions. We observe faded colors indicating low confidence in the predicitons, with no alignment at all with the optimal boundary. This model underfits, as evidenced by low recall (0.616) despite decent accuracy (0.866).
Max Depth 2 : We now have both vertical and horizontal splits, forming four regions with greater probability gradation. This is because the model now considers both attributes $X1$ and $X2$, unlike max_depth=1. The result is an improved recall (0.862) and overall performance (accuracy : 0.900).
Max Depth 4 : This model achieves the highest accuracy (0.914) and F1-score (0.832), as illustrated by a more complex boundary and nuanced color gradation. Therefore, we can conclude that there is a good balance between capturing intricate patterns and avoiding overfitting.
Max Depth 8 and None : The decision boundary for these two models becomes highly segmented, leading to overconfidence in small regions. From max_depth=4, we notice a drop in both accuracy and F1-score, indicating clear overfitting.

1.1.2 Underfitting/Overfitting Discussion

- **Underfitting :** The models max_depth = 1 and max_depth = 2 clearly underfit as we observe a too simplistic decision boundary. This shows that we are unable to capture the complexity of the class distribution and miss pattern identification in the data.
- **Overfitting :** However, at max_depth = 8 and None, the boundaries are overly complex, suggesting the models are fitting noise rather than generalizing. Their lower performance compared to depth 4 supports this hypothesis.

1.1.3 Model Confidence Explanation

With increasing depth, model confidence (indicated by color intensity and boundary sharpness) also rises. Deeper trees can form more specific regions, often leading to purer leaf nodes and higher probability estimates for predictions. However, higher confidence, as seen in depths 8 and None, does not correlate with better generalization, with depth 4 yielding the best balance.

1.2 Average Test Set Accuracies

Max Depth	Average Accuracy	Standard Deviation
1	0.860	0.005
2	0.898	0.008
4	0.913	0.006
8	0.897	0.003
None	0.890	0.003

TABLE 2 – Average Accuracy and Standard Deviation for Different Max Depth Values

— **1 :** *Average Accuracy : 0.860 / Std Dev : 0.005.*
Underperforms relative to other depths but shows low variability.
— **2 :** *Average Accuracy : 0.898 / Std Dev : 0.008.*
Shows an improvement over depth 1 with slightly higher variability.
— **4 :** *Average Accuracy : 0.913 / Std Dev : 0.006.*
Highest accuracy with consistent results across trials. Most well balanced model.
— **8 :** *Average Accuracy : 0.897 / Std Dev : 0.003.*
Slight accuracy decline, suggesting overfitting. Low variability confirms that by being very small, indicating that the data is overly fitted.
— **None :** *Average Accuracy : 0.890 / Std Dev : 0.003.*
Same as max_depth=8.

These results confirm our predictions :
— Low max_depth values underfit.
— High max_depth values overfit.

Depth 4 achieves the optimal balance, with the highest average accuracy.

2 K-Nearest Neighbors

2.1 Decision Boundary Observations

2.1.1 Illustrations

The decision boundary plots for each $n\_neighbors$ setting are presented in Appendix B.

2.1.2 Evolution of the Decision Boundary

Comments on Different n_neighbors Values
— **n = 1 :** Highly irregular boundaries that closely fit individual data points, creating many small, isolated regions for each class.
— **n = 5 :** Smoother boundaries, with softer gradual transitions between regions, though some areas still overfit.

- **n = 50 and n = 100 :** Significantly smoother decision boundaries, balancing high- and low-confidence regions. A diagonal trend is visible with well-defined transitions.
- **n = 500 :** The boundary becomes nearly linear, indicating clear underfitting, as it categorizes all data points to the same class.

Evolution
As n increases, complexity reduces, moving from overfitting at $k = 1$ to underfitting at $k = 500$. Optimal performance is observed at $k = 50$ and $k = 100$, where complexity and generalization are balanced.

2.2 Average Test Set Accuracies

Neighbors	Average Accuracy	Standard Deviation
$n = 1$	0.892	0.005
$n = 5$	0.913	0.003
$n = 50$	0.924	0.004
$n = 100$	0.923	0.004
$n = 500$	0.750	0.000

TABLE 3 – Average Test Set Accuracies and Standard Deviations for Different Values of n_neighbors

Comments on Accuracy and Standard Deviation

- **n = 1 :** *Average Accuracy : 0.892, Std Dev : 0.005.*
  Overfitting, with low accuracy and moderate variability.
- **n = 5 :** *Average Accuracy : 0.913, Std Dev : 0.003.*
  Improved balance, with higher accuracy and lower variability.
- **n = 50 :** *Average Accuracy : 0.924, Std Dev : 0.004.*
  Highest accuracy, indicating optimal complexity.
- **n = 100 :** *Average Accuracy : 0.923, Std Dev : 0.004.*
  Similar performance to $n = 50$, suggesting little additional benefit.
- **n = 500 :** *Average Accuracy : 0.750, Std Dev : 0.000.*
  Severe underfitting with low accuracy and no variability.

Additional Observations

1. Performance improves from $k = 1$ to $k = 50$, stabilizes around $k = 100$, and drops sharply at $k = 500$.
2. Optimal values of $k = 50$ and $k = 100$ offer the best balance between complexity and generalization.
3. Low standard deviations for $k = 5$ to $k = 100$ indicate consistent performance.
4. The severe underfitting at $k = 500$ suggests that the majority class dominates predictions.

3 Perceptron

3.1 Derivation of the Perceptron Gradient

For a binary perceptron classifier with input $x \in \mathbb{R}^p$, label $y \in \{0, 1\}$, and weights $w \in \mathbb{R}^{p+1}$, the cross-entropy loss is defined by :

$$L(x, y, w) = -y \log(\hat{f}(x; w)) - (1 - y) \log(1 - \hat{f}(x; w))$$

where $\hat{f}(x; w) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$ is the sigmoid function output.
Using the chain rule, the gradient $\nabla_w L$ becomes :

$$\nabla_w L = \frac{\partial L}{\partial \hat{f}} \cdot \frac{\partial \hat{f}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

with components :
- $\frac{\partial L}{\partial \hat{f}} = -\frac{y}{\hat{f}} + \frac{1-y}{1-\hat{f}}$
- $\frac{\partial \hat{f}}{\partial z} = \hat{f}(1 - \hat{f})$
- $\frac{\partial z}{\partial w} = x$

Simplifying, we find :

$$\nabla_w L = (\hat{f}(x; w) - y) \cdot x$$

indicating that the gradient is proportional to both the prediction error $(\hat{f}(x; w) - y)$ and input features x. We update weights using :

$$w \leftarrow w - \eta \nabla_w L(x, y, w)$$

where η is the learning rate.

3.2 Implementation Details

Our implementation involves several important features :
- **Sigmoid Activation Function :** Provides class probability values in the range of 0 and 1.
- **Random Weight Initialization :** To prevent saturation and improve learning, weights are initialized with small, random values (scaled by 0.01).
- **Stochastic Gradient Descent with Shuffled Data :** Data is shuffled each epoch to improve generalization, using `np.random.permutation`.
- **Bias Term Addition :** The model can learn offsets regardless of the input values by including a constant bias term in the input features.

3.3 Experimental Results

(3.3.a) Illustrating the decision boundary for different learning rates

For each learning rate η, the decision boundary plots and confusion matrices are in Appendix C.

(3.3.b) Evolution of the decision boundary with respect to η

The perceptron decision boundary changes as the learning rate (η) increases :
- **Low learning rates (e.g., $\eta = 1 \times 10^{-4}$) :** Wider transition zones and smooth boundaries. Low precision and high recall suggest conservative revisions.
- **Moderate learning rate (e.g., $\eta = 1 \times 10^{-2}$) :** Sharper boundaries with balanced precision and recall. Highest accuracy (0.929) with a well-calibrated boundary.
- **High learning rate (e.g., $\eta = 1 \times 10^{-1}$) :** Boundaries with high precision but lower recall, indicating overfitting to the dataset.

3.4 Classification Performance

Classification performance across five generations is shown in Table 4 :
This shows that $\eta = 1 \times 10^{-2}$ achieves the highest mean accuracy (0.925) with the lowest standard deviation (0.003), therefore being the optimal learning rate.

Learning Rate (η)	**Mean Accuracy $\pm$ Std**
1×10^{-4}	0.917 $\pm$ 0.004
5×10^{-4}	0.919 $\pm$ 0.005
1×10^{-3}	0.920 $\pm$ 0.004
1×10^{-2}	0.925 $\pm$ 0.003
1×10^{-1}	0.922 $\pm$ 0.004

TABLE 4 – Perceptron Classification Results

4 Method Comparison

4.1 Parameter Tuning

To tune the hyperparameters (max_depth, n_neighbors, and η) using only the learning set, k-fold cross-validation is used. This method is effective for maximizing information from limited datasets

K-Fold Cross-Validation Process

In k-fold cross-validation, the data is divided into k equal sections, typically $k = 5$ or $k = 10$. One section is held back for testing while the model trains on all other sections. This is repeated k times, with each section taking a turn as the test set. For each hyperparameter configuration being evaluated, we run this entire process - the model trains k different times, and we calculate the average performance across all k rounds. The hyperparameter settings that achieve the highest average score are chosen as optimal.

Advantages and Disadvantages

Advantages :
- Utilizes the entire dataset effectively, which is especially beneficial for small datasets, as no separate validation set is required.
- Reduces bias by evaluating performance over multiple folds, minimizing reliance on a single, potentially unrepresentative, validation set.

Disadvantages :
- Requires training the model multiple times (k times per hyperparameter), which can be time-consuming, especially with complex models or large datasets.
- If $k = N$ (leave-one-out cross-validation), it is unbiased but has high variance and is computationally intensive.
- If $k = 5$ or 10, it is faster with lower variance but may introduce some bias due to fewer data points per fold.

Use in Small Datasets : In small datasets, splitting the data for testing can lead to unreliable results. K-fold cross-validation addresses this by rotating each part of the dataset as the validation set, ensuring a comprehensive evaluation where all data points contribute to hyperparameter selection.

4.2 Analyze and Compare Method Performance With and Without Noisy Features

4.2.1 Performance Without Noisy Features (Original Features)

The results in Table 5 show that, although tuning yields reasonable accuracy across all models, the performance is slightly weaker compared to previous sections. This drop could occurs because tuning

Model	**Mean Accuracy (w/o N.F.)**	**Std Deviation (w/o N.F.)**	**Mean Accuracy (w/ N.F.)**	**Std Deviation (w/ N.F.)**
DecisionTree	0.910	0.004	0.899	0.007
KNN	0.923	0.004	0.874	0.010
Perceptron	0.920	0.003	0.901	0.004

TABLE 5 – Performance Comparison With and Without 200 Noisy Features

occasionally selects suboptimal hyperparameters, as the learning sample may misleadingly indicate they are optimal.

4.2.2 Performance With 200 Noisy Features

— **DecisionTree :** Mean accuracy decreased from 0.910 to 0.899, with a slight increase in standard deviation to 0.007. This result shows the DecisionTree's robustness, as it effectively maintains accuracy despite irrelevant features.

— **KNN :** Accuracy declined considerably from 0.923 to 0.784, with a higher standard deviation of 0.010, indicating that KNN is highly sensitive to noise due to its reliance on distance-based classification.

— **Perceptron :** Accuracy dropped from 0.920 to 0.901, with a small increase in standard deviation to 0.004. Although less resilient than the DecisionTree, the Perceptron managed noise better than KNN, but it might benefit from feature selection or regularization in noisy environments.

4.2.3 Method Ranking and Sensitivity to Noise

— **Without Noisy Features :** All models performed similarly, but the DecisionTree showed a slight edge in stability, suggesting it is more consistent across folds.

— **With Noisy Features :** The DecisionTree model retained the highest accuracy, showing strong resilience to noise. The Perceptron adapted better than KNN, which remained most sensitive to noise due to its distance-based approach, ranking lowest in noisy conditions.

A Decision Tree – Decision Boundaries and Confusion Matrices

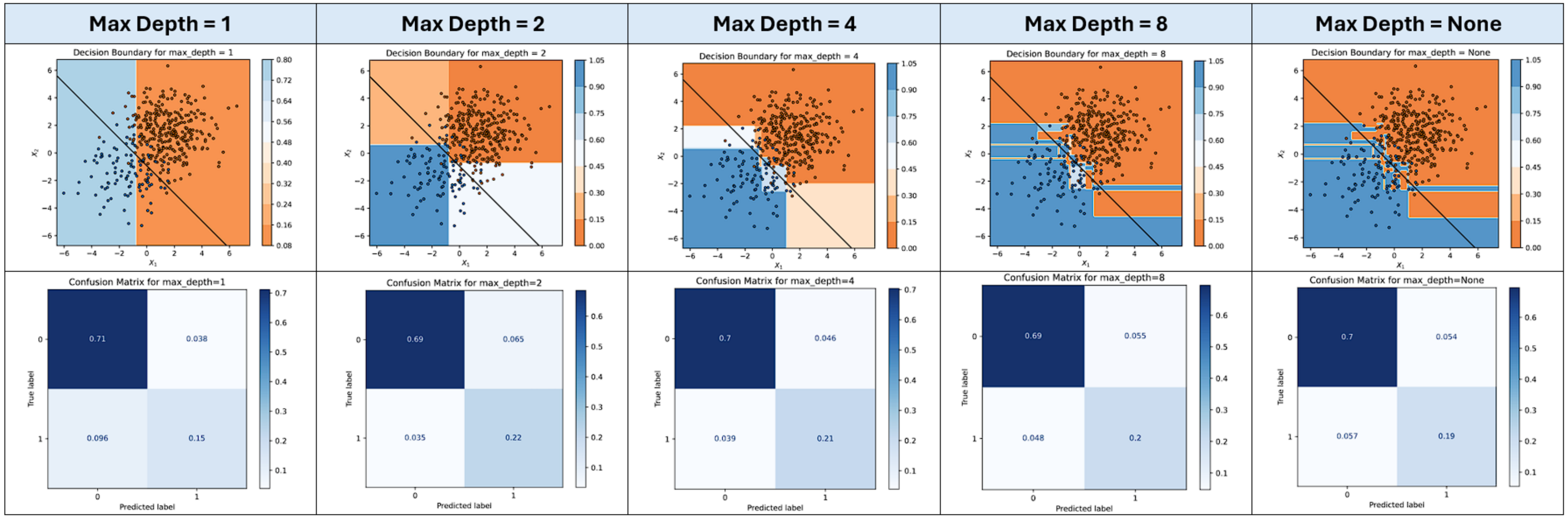

FIGURE 1 – Appendix : Decision Tree – Decision Boundaries and Confusion Matrices for different Max Depth Values

B KNN – Decision Boundaries and Confusion Matrices

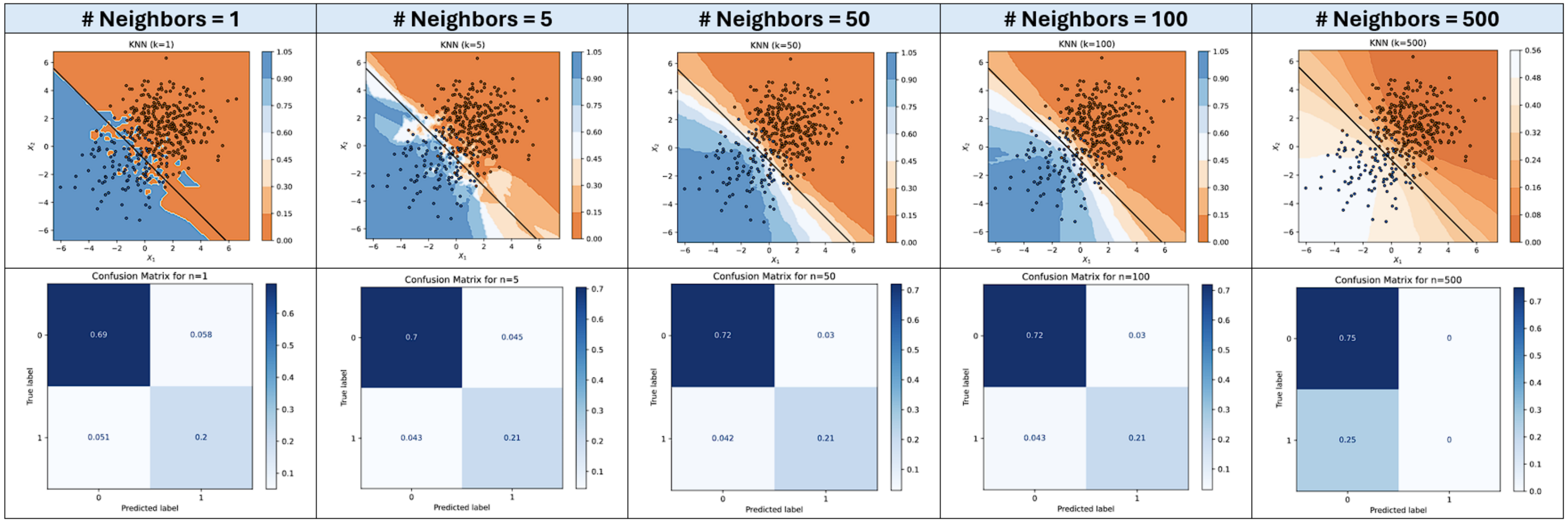

FIGURE 2 – KNN – Decision Boundaries and Confusion Matrices for Different Neighbors values

C Perceptron – Decision Boundaries and Confusion Matrices

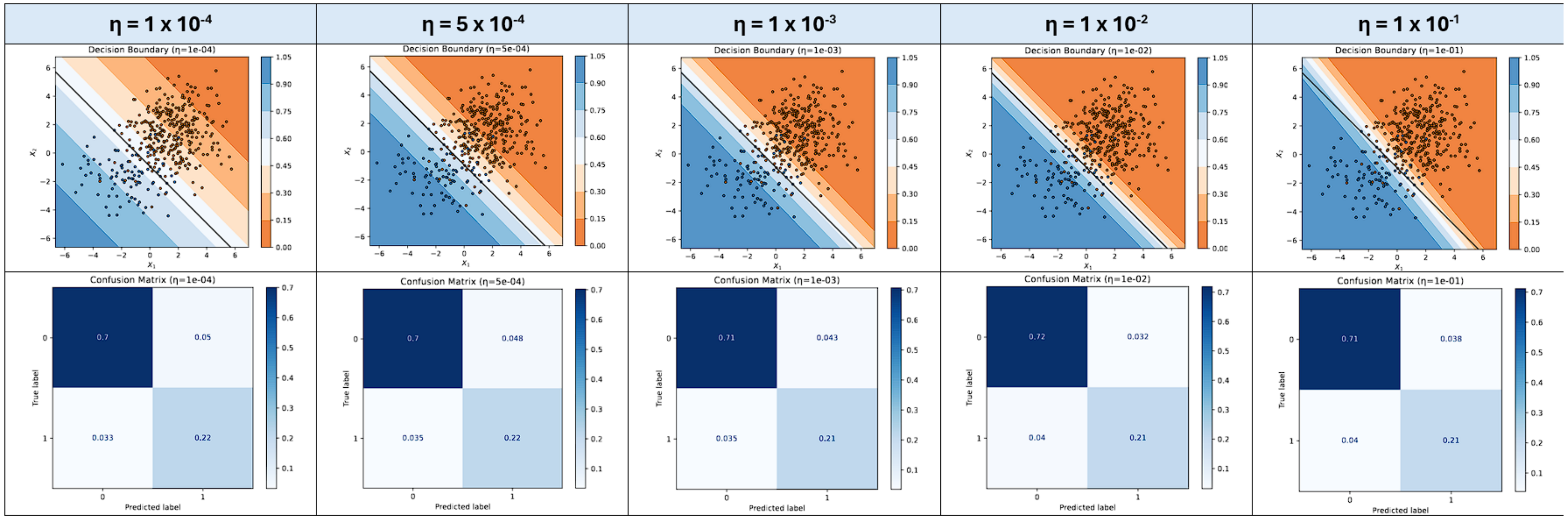

FIGURE 3 – Perceptron – Decision Boundaries and Confusion Matrices for Different Learning Rates