MŰEGYETEM 1782

Budapesti Műszaki és Gazdaságtudományi Egyetem

Csarkovszkij Artyjom

Phaxay Thipphasone

DL-BOTGUARD

BUDAPEST, 2025

Table of contents

1 Introduction **4**
1.1 Motivation 4
1.2 Description 4
1.3 Related Work 4
2 Methods **6**
2.1 Problem 6
2.2 Datasets 6
2.2.1 Our Own Dataset 6
2.2.2 BOUN Dataset 7
2.3 Data Preprocessing 7
2.3.1 Feature Definitions 7
2.3.2 Windowing 8
2.3.3 Dataset Split 8
2.4 Models 8
2.4.1 Baseline: Logistic Regression 8
2.4.2 GRUMouseNet 8
2.5 Hyperparameter Optimization 9
2.6 Training Setup 9
3 Evaluation **10**
3.1 Metrics 10
3.2 Baseline Results 10
3.2.1 Initial baseline on Our dataset 10
3.2.2 Baseline without process_name 10
3.2.3 Baseline on BOUN dataset 11
3.3 Our Dataset Results 12
3.4 BOUN Dataset Results 14
4 Conclusions **16**
4.1 Main Findings 16
4.2 Limitations 16
4.3 Future Work 16
5 References **18**

Appendix 19

1 Introduction

1.1 Motivation

For years automated bots have generated an enormous amount of Internet traffic. Many of these bots are trying to imitate human behavior and interactions to bypass CAPTCHAs, abuse online services or commit crimes like fraud. Traditional detection methods typically rely on network features, browser fingerprints, however as bots are becoming more sophisticated these metrics are easier to spoof.

Mouse and keypress dynamics offer a different type of metric: they capture the fine-grained motor behavior of users - how a cursor accelerates, decelerates, changes direction, and pauses. These patterns are difficult for bots to reproduce perfectly, especially in real time. As a result of this, mouse dynamics have been studied extensively as a behavioral biometric for both continuous authentication and intrusion detection. [1]

1.2 Description

In this project we focus on classification problems: given a short sequence of mouse events decide whether it was produced by a human or by a robot.

We work with two datasets:

- Our own collected dataset that was captured using custom logger software.
- The BOUN Mouse Dynamics Dataset, a public dataset from Boğaziçi University. [2]

1.3 Related Work

Mouse dynamics have been explored previously as behavioral biometrics for both authentication and intrusion detection. Several surveys and empirical studies show that models trained on cursor trajectories can reach competitive performance. [1] [3]

Deep learning models have been applied to mouse dynamics to capture temporal patterns and achieve better performance than classical feature-based models. [4] [5] Other works specifically target bot detection using mouse movement trajectories, often highlighting differences in speed profiles, curvature and micro-pauses between human and automated behavior.

Our project works as the intersection of these topics: we use a GRU-based network to classify short mouse sequences as human or bot, and we analyze how different datasets and threshold choices affect bot detection performance.

.

2 Methods

2.1 Problem

Mouse dynamics are produced continuously over time as the user interacts with the computer. A single raw event does not contain enough information to determine whether the behavior is human-like or not. What matters is the temporal pattern over a sequence of actions. Hence, we define the task as a **window-level binary classification problem.** Each sample is a sequence of T mouse events:

$$x = [(x_1, y_1, t_1), \dots, (x_T, y_T, t_T)]$$

Where (x_i, y_i) are screen coordinates and t_i is a timestamp.

The model outputs a probability $P(bot \mid x)$. After inference we compare this calculated probability with a given threshhold f. If $P(bot \mid x) \geq$ f, we classify the actions as bot actions, otherwise human. The choice of T identifies trade-off between falsely banning humans and allowing bots through.

2.2 Datasets

2.2.1 Our Own Dataset

We built a custom mouse event logger that records mouse events from our PCs while working or gaming:

- Screen coordinates
- Timestamp
- Event type
- Foreground application (process_name)

Also, a custom software to imitate bot behavior was written. It was searching for the web.

Due to the nature of our logger, a key challenge for us was data imbalance. Since bots' behavior is more straightforward, it logs much less data per time browsing, so we were left with only 10k samples for bot and hundreds of thousands for a human. To avoid a severely skewed dataset, we down sampled the human part to roughly match the bot class.

Another issue was label leakage through the *process_name* feature - bots were running in an extremely limited set of applications, while humans used a wider range. Keeping these fields as a feature, leads to situation where the model learned to detect bots simply by checking which application was active, rather than learning mouse behavior. Because of this, process_name and other application-specific attributes were dropped in the final preprocessing.

Overall, our own dataset was useful for initial modelling ideas, but its size and the difficulty of generating realistic bot behavior limited its usefulness for training a robust detector. [6] [7]

2.2.2 BOUN Dataset

The BOUN Mouse Dynamics Dataset is a public dataset that contains long-term mouse data from 24 users with around 2550 hours of active usage. We focus on a single user to keep learning manageable.

Compared to our own data, the BOUN dataset has smoother trajectories with more consistent sampling.

2.3 Data Preprocessing

Both datasets are processed with the help of a common *data_loader* module to ensure consistent feature definitions.

2.3.1 Feature Definitions

For each raw event, we extract:

- *timestamp_ms*
- *x, y*
- *event_type*

Events are sorted by *timestamp_ms*, and rows with missing timestamps are dropped.

2.3.2 Windowing

We slide a fixed-length window of size $T = 100$ events with step size 50. For each window we stack the per-event features into a matrix of shape and assign the window label.

This produces an array $X \in R^{N \times T \times F}$ and a label vector $y \in \{0,1\}^N$.

2.3.3 Dataset Split

We randomly split windows into train, validation and test sets using stratified splits to preserve class balance. After that using statistics from the training set only, we apply scaling per feature dimension.

2.4 Models

2.4.1 Baseline: Logistic Regression

As a simple baseline, we trained a logistic regression classifier. This model is fast to train. However, such simple aggregates lose detailed temporal information, and the performance maximum is reached relatively quickly.

Also, we faced a problem with improper feature engineering that later was solved.

2.4.2 GRUMouseNet

For the better modeling of the temporal structure of mouse movements, we coded GRUMouseNet, a recurrent neural network based on the Gated Recurrent Unit (GRU). Unlike the logistic regression baseline, it processes full event sequences and learns patterns that unfold over time.

Structure:

- Bidirectional GRU for richer temporal context
- Optional multi-layer depth and dropout
- Lightweight classifier for final decision making.

2.5 Hyperparameter Optimization

To avoid manually tuning hyperparameters, we used Weights & Biases Sweeps [8] to automate the process.

Hyperparameters explored:

- Hidden size: $\{32, 64, 128\}$
- Number of layers: $\{1,2\}$
- Dropout: $\{0.0,0.1,0.3\}$
- Learning rate: $\{3e^{-4}, e^{-3}, 3e^{-3}\}$
- Batch size: $\{64,128\}$
- Number of epochs: $\{10, 15\}$

As a sweep strategy we used random search. For each run, wandb.ai recorded loss curves, validation accuracy, and configuration details.

2.6 Training Setup

Training was implemented in PyTorch and executed on CPU or GPU depending on the environment. GPU training was preferred due to speed gains.

3 Evaluation

3.1 Metrics

We use standard classification metrics:

- Accuracy
- Precision
- Recall
- F1-score

We also analyzed the effect of changing the decision threshold ƭ applied to the predicted bot probability. By default, $ƭ = 0.5$, but for practical systems a higher or lower threshold can be chosen depending on whether false bans of humans or missed bots are considered more harmful.

3.2 Baseline Results

Before building an advance model, we created baseline classifier to understand the difficulty of the task and to set reference points for later improvements. We used Logistic Regression as a baseline.

3.2.1 Initial baseline on Our dataset

Initially, using the full feature set including the *process_name* column the classifier appeared to perform extremely well. (Figure 3.1)

These results initially suggested the task was extremely easy.

However, during later analysis we discovered a data leak: Bots were executed only inside a small set of applications, therefore *process_name* uniquely revealed the class label. The model was not learning behavior - it was learning software context. This invalidated the initial baseline.

3.2.2 Baseline without process_name

After removing *process_name* and other metadata features, the baseline performance decreased dramatically:

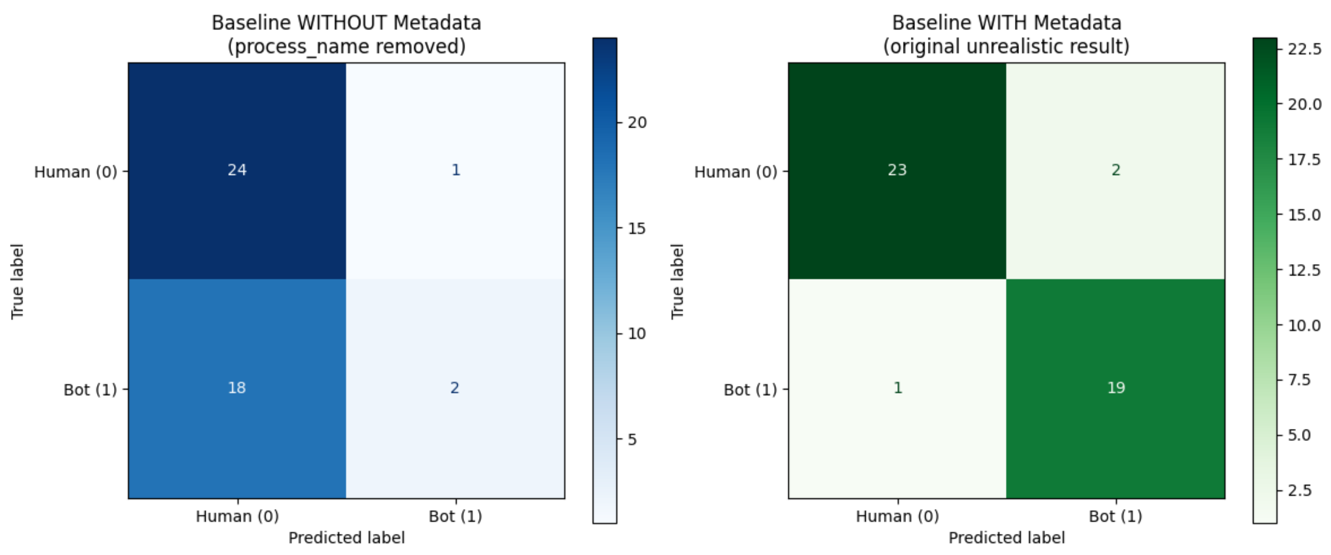

Figure 3.1 Comparison of baseline confusion matrices on our own dataset.

Now the classifier struggles to capture differences. The extremely sharp drop in recall for class 1 (bot) indicates strong underfitting. The model collapses toward classifying most samples as humans. This is expected, because mouse movements are temporal, and a linear model working on aggregated features cannot capture this structure.

3.2.3 Baseline on BOUN dataset

We also trained the same logistic regression baseline on the BOUN Mouse Dynamics Dataset. Unlike our own dataset, BOUN contains millions of raw mouse events. In this setting (Figure 3.2), the baseline reaches an overall accuracy of 52%, which is only slightly above random guessing. The model struggles primarily because the two classes substantial overlap in simple statistical features such as mean displacement or speed. Even though bot windows achieved relatively high recall (73%), the precision for the class remained relatively low (40%), and human windows are frequently misclassified as bots (≈ 5000 false positives). This behavior indicates that simple linear decision boundaries are insufficient for modeling mouse dynamics.

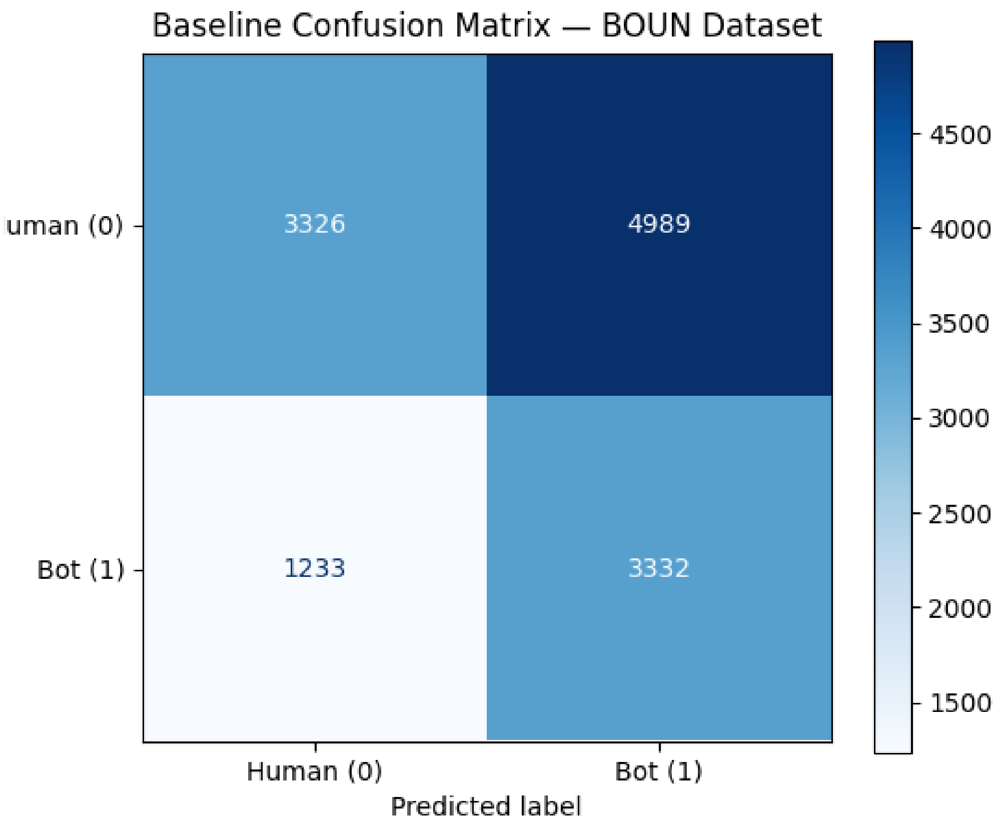

Figure 3.2 Baseline Confusion Matrix on BOUN Dataset

3.3 Our Dataset Results

After training the GRU model on our own dataset, we evaluated its performance across threshold range [0.10 to 0.90] with a step of 0.1. The results show that the model does not generalize well to this dataset and fails to separate humans from bots reliably. (Figure 3.3, Figure 3.4)

Thresholds 0.10 to 0.40 - Model predicts only class 1 (bot)

Threshold = 0.50 - Slightly balanced

Thresholds 0.60 to 0.90 - Model predicts only class 0 (human)

This behavior mirrors the imbalance observed earlier - the model assigns low probabilities to the bot class almost every time. This behavior was rather expected due to key limitations:

- Small amount of training data with high variance and noise
- The movements generated by bot were very uniform.

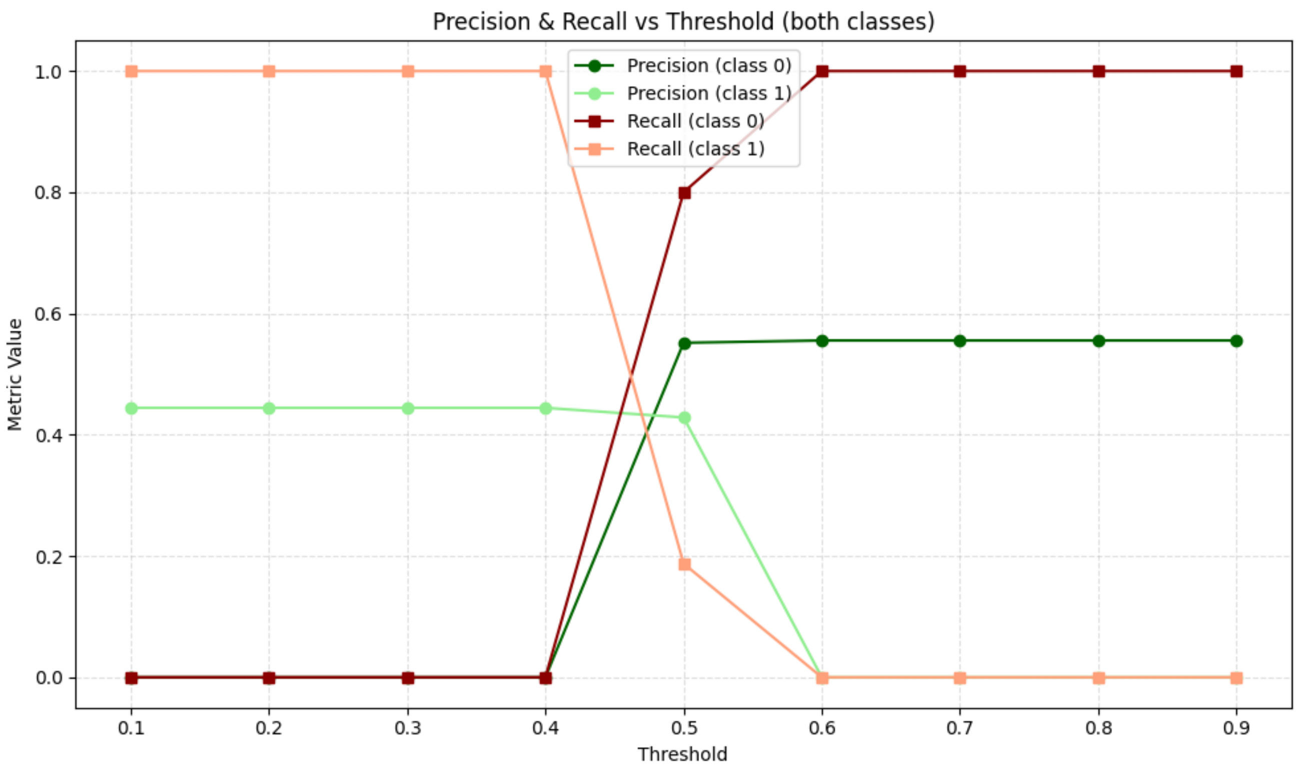

Figure 3.3 Precision & Recall vs Threshold - Our Own Dataset

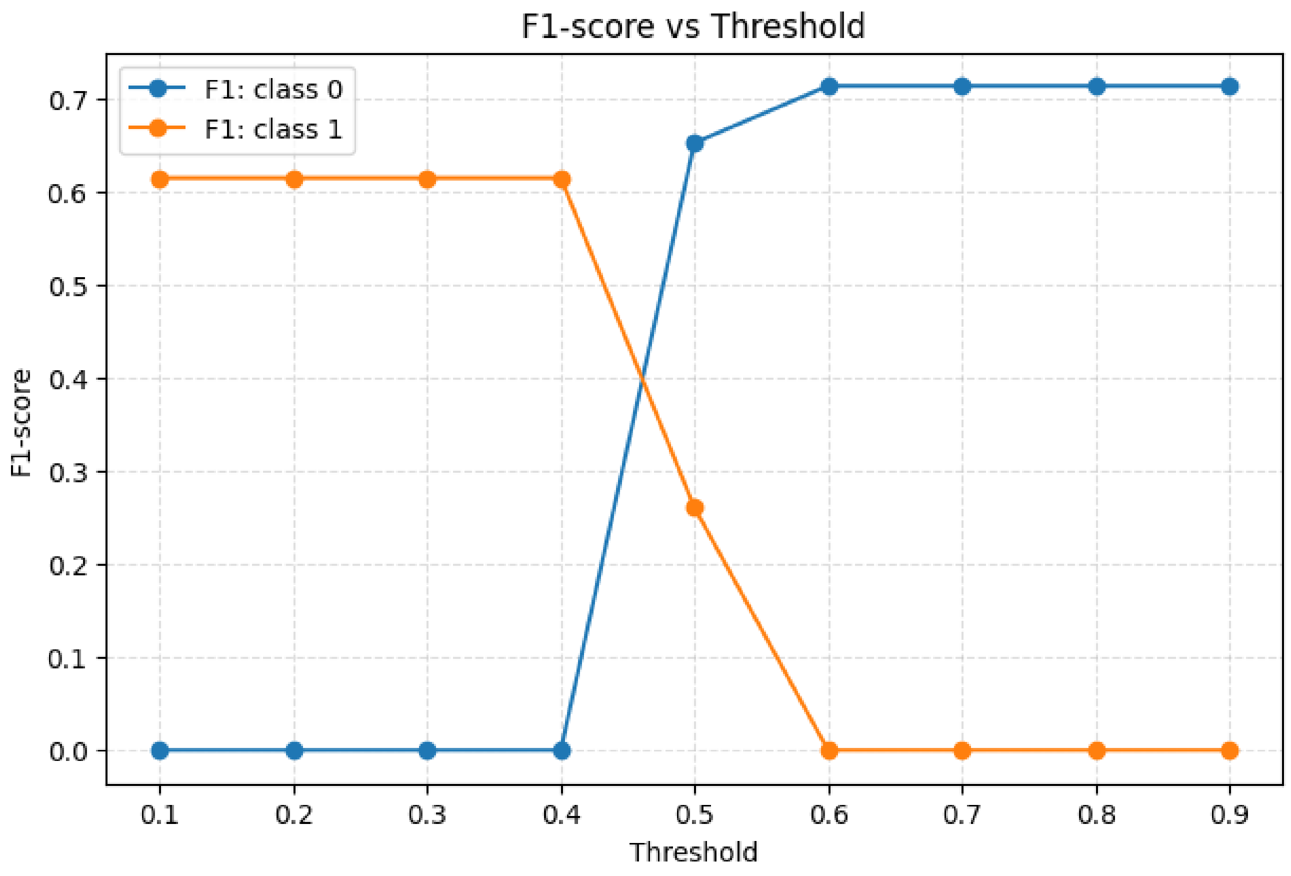

Figure 3.4 F1 score vs Threshold - Our Own Dataset

3.4 BOUN Dataset Results

After training the GRU model on the BOUN dataset, we evaluated its performance across threshold range [0.10 to 0.90] with a step of 0.1. The results show that the model has strong performance on this dataset, especially compared to the baseline model. (Figure 3.5, Figure 3.6)

Across all thresholds, the model achieved its best overall performance near the default threshold of 0.50, providing the most stable trade-off between detecting bots and avoiding false alarms for human users. But for some applications you can consider other thresholds:

- **Low Threshold - Aggressive Bot Detection**

  At low thresholds, the model becomes extremely sensitive and detects almost all bot activity, however, it incorrectly flags extreme number of human users.

  So we can say that it is almost useless.

- **Mid Threshold - Best Overall Balance**

  At these thresholds the model catches most bots without overwhelming users with false warnings. The prediction quality is the most stable and balanced.

  Can be used in scenarios where you just flag possible bots for further investigation, maybe with combination of others traditional detection methods.

- **High Threshold - User-Friendly Mode**

  Here the system becomes conservative, and almost all human users pass without issues But many bots also do, because the system waits for very strong evidence before flagging anything.

  It can be used where avoiding user frustration is the top priority, as a main source of information before banning a user. Sometimes it is better to let half bots through but have close to zero false bans.

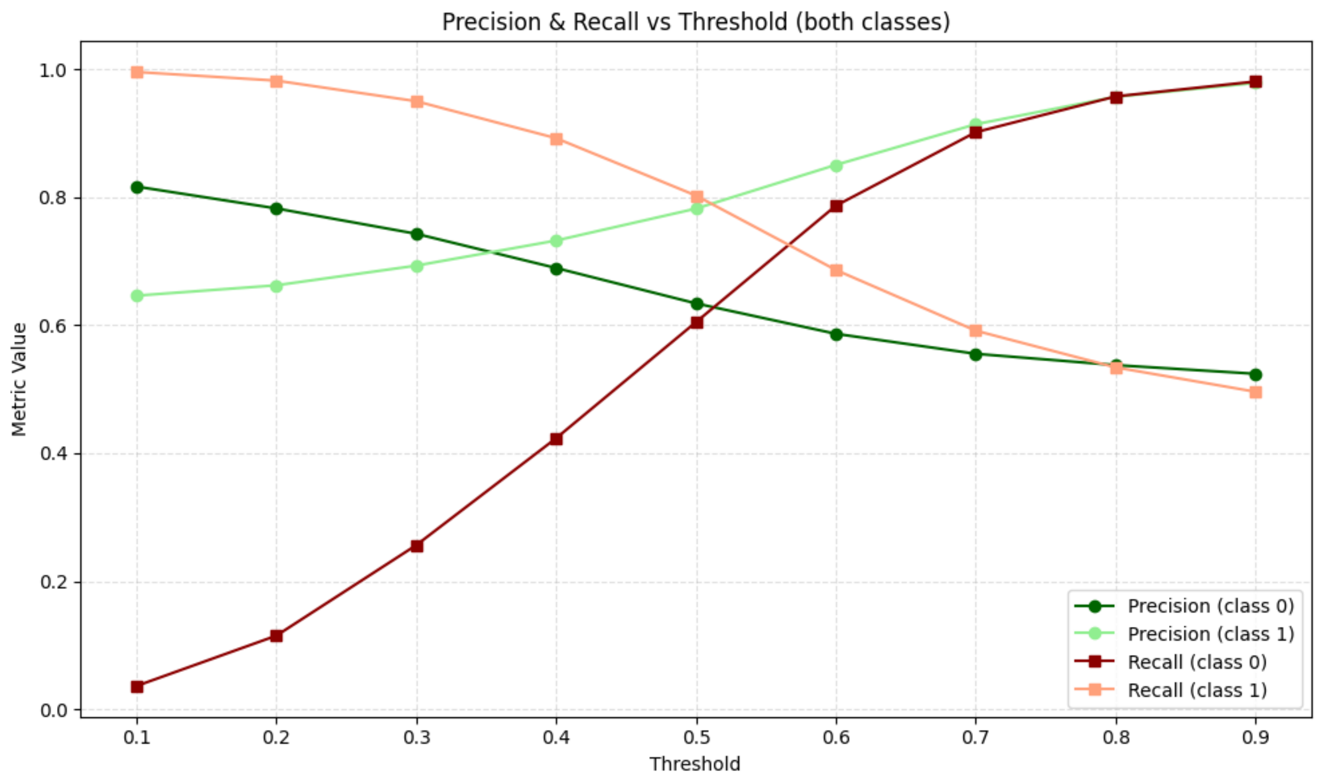

Figure 3.5 Precision & Recall vs Threshold - BOUN Dataset.jpg

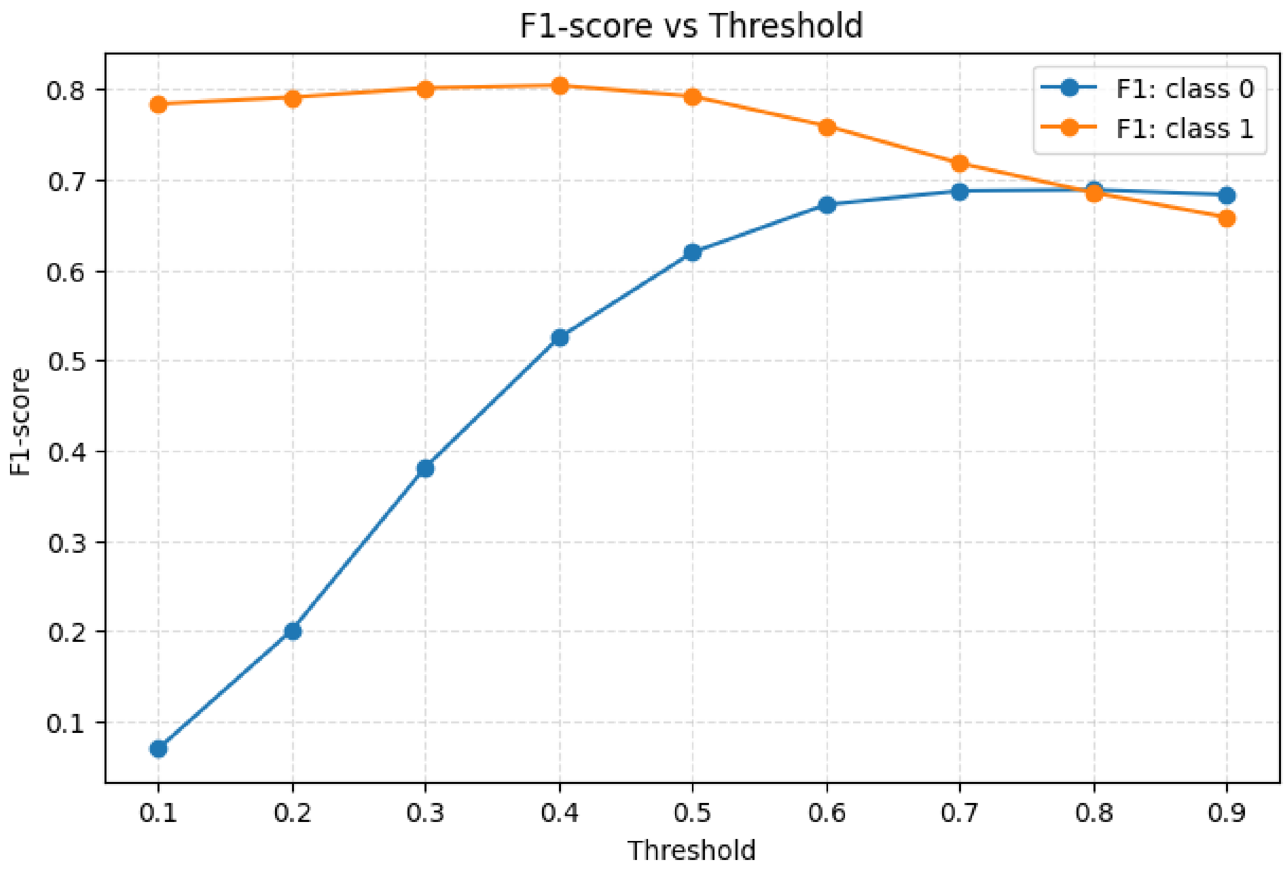

Figure 3.6 F1 score vs Threshold - BOUN Dataset

4 Conclusions

4.1 Main Findings

This project discovered human users and automated bots classification based on mouse-dynamics. The main findings are the following:

- **GRU sequence models perform well on temporal behavioral data.**
- **Window-based sequence framing works well.**
- **Dataset quality matters more than everything.**
- **The decision threshold strongly controls model behavior.**

Overall, our results show that mouse movements can be used for bot detection, but the final performance depends heavily on data quantity, data quality, and appropriate threshold selection.

4.2 Limitations

Despite promising results, we must acknowledge some limitations:

- **Custom dataset limitations**

  Our dataset is small, not diverse enough, and collected from a single environment.

- **BOUN dataset limitations**

  While more complete, BOUN's bot behaviors are synthetic and may not fully represent modern bot strategies that adapt their patterns to imitate human movement more closely.

- **Feature engineering minimalism**

  We have included only basic kinematic features. Advanced behavioral features can theoretically improve accuracy further.

4.3 Future Work

The project has several directions for improvement and evolution:

- **Collect a larger and more diverse real-world dataset.**

Including more users, multiple devices, various applications, and diverse bot strategies would greatly increase model performance.

- Try even more advanced deep models.
- **Cross-dataset generalization experiments**

  A production system must remain effective even when the distribution shifts.
- **Real-time system implementation**

  Integrating the model into a live interface would allow continuous authentication or bot detection.
- **Feature exploration**

  Additional motion metrics could reveal deeper behavioral signatures.

5 References

[1] N. Siddiqui, R. Dave, M. Vanamala and N. Seliya, "Machine and Deep Learning Applications to Mouse Dynamics for Continuous User Authentication," *Machine Learning and Knowledge Extraction,* vol. 4, no. 2, p. 502–518, 2022.

[2] A. A. Kılıç, M. Yıldırım and E. Anarım, "Bogazici mouse dynamics dataset," *Data in Brief,* vol. 36, p. 107094, 2021.

[3] E. Kuric, P. Demcak, M. Krajcovic and P. Nemcek, "Is mouse dynamics information credible for user behavior research? An empirical investigation," *OpenReview,* 2024.

[4] M. Antal and N. Fejér, "Mouse dynamics based user recognition using deep learning," *Acta Universitatis Sapientiae, Informatica,* vol. 12, no. 1, p. 39–50, 2020.

[5] H. A. Lee, N. Prathapani, R. Paturi, S. Parmaksiz and F. Di Troia, "NLP-based user authentication through mouse dynamics," in *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, 2022.

[6] "DL-BotGuard human dataset," [Online]. Available: https://drive.google.com/drive/folders/1o9KxK52oGi1hZoTS82DmheeAk2P6MR10?usp=drive_link.

[7] "DL-BotGuard bot dataset," [Online]. Available: https://drive.google.com/drive/folders/13wx3vIMZ87HQBLQJx5xSQROT4uVVdbP-?usp=drive_link.

[8] "Weights & Biases Sweeps Documentation," [Online]. Available: https://docs.wandb.ai/models/sweeps.

Appendix

According to the policy, we are obligated to explicitly declare how large language models were used:

- All docstrings were generated with the use of LLMs.
- AI help was used for paraphrasing when writing documentation.

All texts were reviewed, adjusted by us before inclusion in the final submission.