User Authentication through Mouse Dynamics

Zohreh Ashtarilarki
Diya Doshi

Fall 2024

Contents

Introduction

The continuous evolution of biometric systems has introduced novel methods for identifying and authenticating users. While traditional biometrics such as fingerprints and facial recognition are well-established, behavioral biometrics like mouse dynamics are gaining traction due to their non-intrusive nature. Mouse dynamics involves analyzing a user's interaction patterns with a mouse, such as movement paths, click behavior, and idle times. These patterns are unique to each individual and serve as reliable indicators for identification or authentication tasks.

This project explores the classification of users based on mouse dynamics using advanced machine learning architectures. By leveraging convolutional neural networks (CNNs) to capture spatial relationships, long short-term memory networks (LSTMs) to extract temporal dependencies, and artificial neural networks (ANNs) to combine diverse feature sets, this study seeks to establish a robust framework for user classification. The experiments span feature engineering, data preprocessing, model development, and performance optimization, offering a comprehensive understanding of the efficacy of mouse dynamics as a biometric modality.

This research is motivated by the growing need for secure, nonintrusive authentication systems in a digitally connected world. Mouse dynamics can be collected passively during routine tasks, making it an ideal candidate for continuous authentication scenarios. This report provides insights into this investigation's methodologies, challenges, and outcomes.

Dataset

The dataset consists of mouse interaction data collected from multiple users, capturing both spatial and temporal aspects of their behavior. The primary features in the original dataset include:

- **Record Timestamp**: The timestamp indicates when the interaction was recorded.
- **Client Timestamp**: The timestamp reflects when the interaction was processed on the client side.
- **Button**: The state of the mouse button (e.g., 'NoButton', 'Left', 'Right').
- **State**: The action being performed (e.g., 'Move', 'Pressed', 'Released').
- **x and y**: The coordinates representing the mouse's position on the screen.
- **User**: The identifier of the user interacting with the mouse.

After preprocessing, additional **derived features** were created to enhance the dataset:

- **Speed**: The mouse's movement rate, is calculated as the distance traveled over time.
- **Acceleration**: The rate of change of speed over time, indicating how rapidly the user is accelerating or decelerating their mouse movement.
- **Angles**: The directional changes in the mouse's path, capturing the angle between consecutive positions.
- **Delta Angle**: The difference in angle between two consecutive positions, highlighting sharp turns or changes in movement direction.
- **Click Frequency**: Click frequency captures how often a user clicks while interacting with the interface. This feature can offer insights into the user's engagement with the system and patterns in their interaction, such as rapid clicking or more deliberate action.
- **Click Duration**: This feature measures the total time a user holds down the mouse button during interactions. It helps to understand how long a user typically spends on actions such as dragging or selecting items. Longer durations could indicate careful attention or slow decision-making, while shorter durations could reflect faster, more casual interactions.
- **Idle Time**: Idle time captures periods when the user is not interacting with the mouse, such as waiting for the next task or momentarily pausing. This feature was derived by calculating the time between interactions (e.g., clicks or movements) and is indicative of how actively or passively a user engages with the system.

Balancing the data

A quick look at the dataset revealed that it was unbalanced.

```
user
user7     426060
user9     415668
user20    294197
user12    246824
user16    240046
user15    149921
user29    131751
user21    125931
user23    124548
user35     98870
Name: count, dtype: int64
```

To address this imbalance, techniques such as **SMOTE** (Synthetic Minority Over-sampling Technique) and **random oversampling** were used to ensure that all classes were represented more equally during model training.

We employed **supervised learning** with the user as the target variable, treating it as the label for classification. The goal was to train a model to predict the user based on various features such as mouse dynamics, including speed, acceleration, and button usage patterns. This approach aimed to leverage user-specific behaviors to classify and differentiate between individuals based on their unique interaction styles with the system.

Since the data was time-series based, we couldn't shuffle the dataset, as shuffling would disrupt the temporal relationships between consecutive observations.

In summary, the dataset was preprocessed and enriched with derived features that captured key aspects of user behavior. Steps were taken to handle missing and infinite values, normalize numerical features, and address the class imbalance, making the dataset suitable for training machine learning models for user classification based on mouse dynamics.

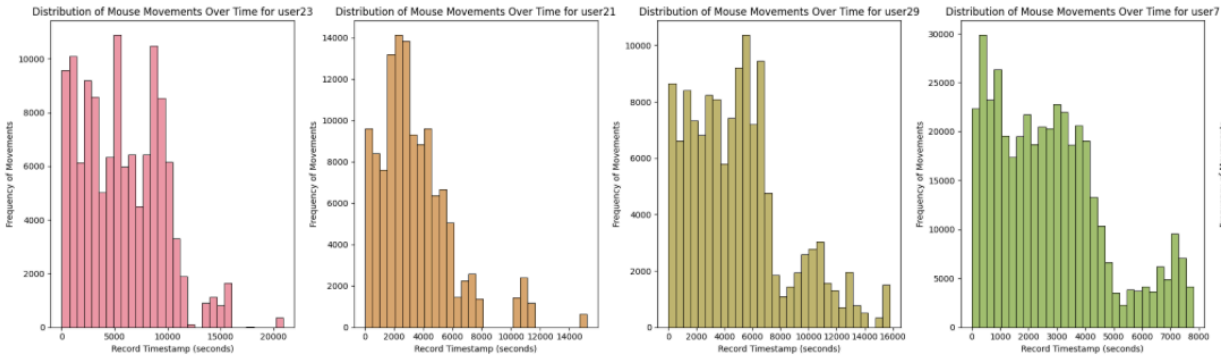
Experiments

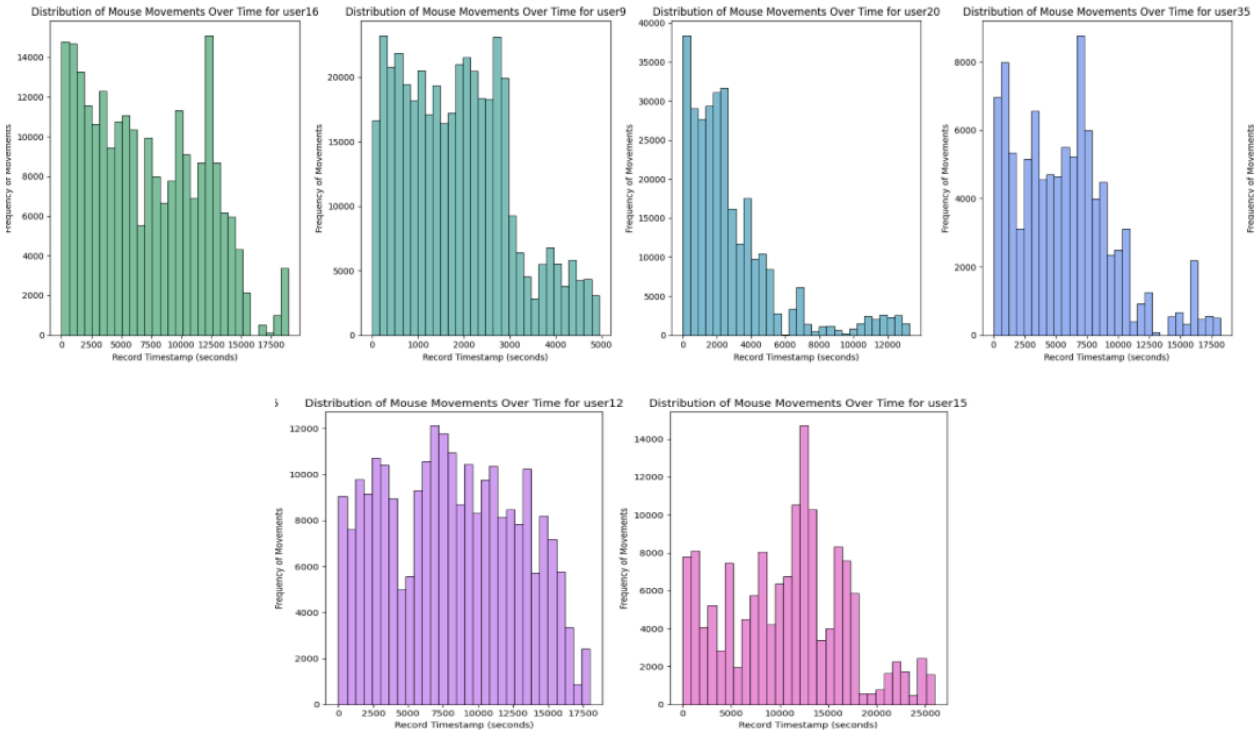
1. Exploring User-Specific Patterns in Mouse Dynamics

Understanding user-specific patterns is crucial for feature engineering. Three experiments were conducted to analyze mouse movements over time, mouse button usage, and mouse state usage.

Mouse Movement Distribution Over Time

Histograms of record timestamps were plotted for each user to examine their frequency of mouse movements. These visualizations highlighted distinct temporal interaction patterns among users, such as consistent activity or sporadic bursts interspersed with idle periods. The unique distributions underscored the importance of temporal features like **idle_time** and **click_duration** for capturing user behaviors. To enhance clarity, each user's plot was assigned a distinct color using Seaborn's "husl" palette.





Mouse Button Usage Patterns

The button feature, representing interactions such as Left, Right, and NoButton, was analyzed using count plots. The results revealed diverse button usage patterns—some users predominantly utilized left clicks, while others demonstrated balanced interactions with multiple button states. This variability suggested the potential of the button feature as a strong discriminator for user classification. The feature was subsequently one-hot encoded for model integration.

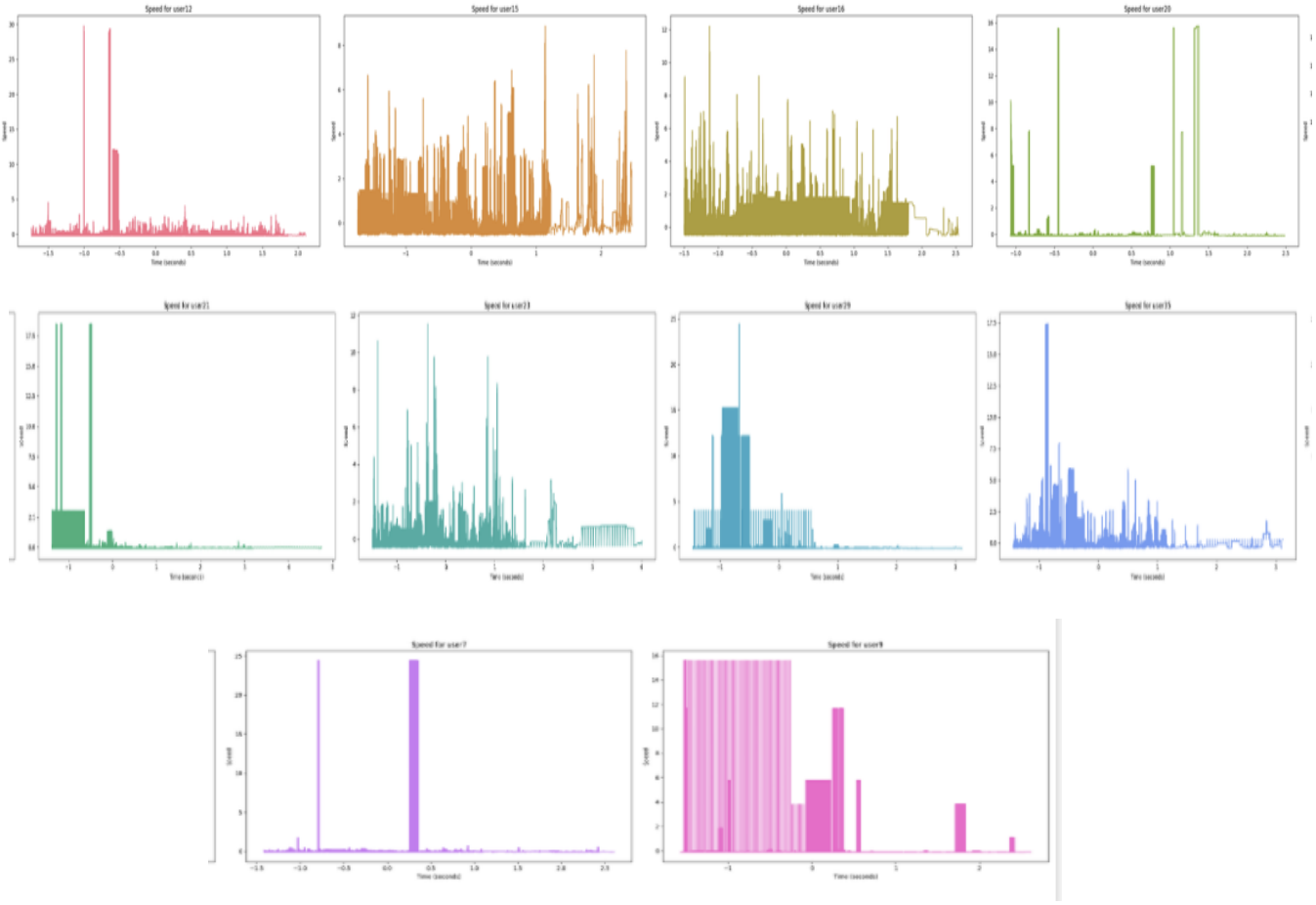
Mouse State Usage Patterns

Similarly, the state feature, representing actions such as Move, Pressed, and Released, was analyzed. Some users primarily moved their mouse, while others alternated between clicks and movements. These patterns emphasized the value of the state as a categorical feature for classification. Like the button, the state feature was one-hot encoded to preserve its categorical information in the model pipeline.

Speed Usage Patterns

The speed feature, which measures the rate of movement of the mouse, was analyzed across different users using line plots. The results revealed varying patterns in users' movement speeds—some users exhibited consistent and steady speeds, while others demonstrated fluctuating or sporadic movements. These differences in speed patterns suggested that the speed feature could be a significant indicator of individual user behavior. As a result, this feature was

carefully examined for its potential in user classification. To enhance its utility in machine learning models, the speed feature was normalized and incorporated into the model for further analysis and prediction.



2. Feature Engineering and Transformation

Applying Rolling Window and Normalization on Numerical Features

A rolling window approach was applied to numerical features (e.g., x, y, record timestamp) to capture local trends and reduce noise. Rolling averages with a window size of 100 smoothed out fluctuations while preserving temporal continuity. Z-score normalization ensured that features were scaled to have a mean of 0 and a standard deviation of 1, improving comparability across features.

Calculating Speed and Acceleration to Capture User Behavior Dynamics

Speed and acceleration were derived as dynamic features to analyze fluidity and responsiveness in mouse movements. Debugging techniques were implemented to handle anomalies like zero or negative time differences, ensuring data consistency. These features enriched the dataset by quantifying user-specific movement patterns.

Capturing Angular Dynamics: Calculating Movement Angles and Directional Changes

Angles and delta angles were computed to capture directional changes in mouse movements. Angles were derived using trigonometric calculations on dx and dy (differences in x and y coordinates). Delta angles highlighted abrupt or smooth transitions, providing insights into user tendencies.

Analyzing Click Patterns: Frequency and Duration of Mouse Clicks

Click patterns were analyzed by introducing **click_frequency** and **click_duration** as features. These metrics quantified the number of clicks and the time spent holding mouse buttons, reflecting user-specific interaction styles. The integration of these features enhanced the dataset's behavioral attributes.

Idle Time Analysis: Measuring Periods of Inactivity

Idle time, representing periods of stationary mouse behavior, was calculated by identifying instances where **x_diff** and **y_diff** were zero. This feature captured the user's rhythm and natural pauses, adding a temporal dimension to the dataset.

3. Data Quality and Preprocessing

Data Quality Check: Null and Infinite Values

Initially, the original dataset had no null values. However, when we used Z-score normalization to standardize the data, null values were introduced. Specifically, when deriving additional features such as speed, acceleration, and delta angles, null values appeared due to division by zero errors (e.g., when the time difference was zero).

Data Cleaning: Handling Null and Infinite Values

To handle Null and infinite values, interpolation and forward-fill (ffill) methods were initially applied. However, these methods resulted in an accuracy of only 0.4, which was suboptimal. As a result, they were handled using a combination of dropping incomplete rows, replacing infinite values with NaN, and applying linear interpolation. This process ensured smooth transitions in the data while preserving feature integrity.

Feature Transformation: One-Hot Encoding for Categorical Features

Categorical features (button and state) were transformed using one-hot encoding. This representation preserved the semantic meaning of categories while enabling the machine learning models to analyze patterns in categorical data effectively.

4. Model Development and Optimization

Initial Model Training

A CNN + LSTM + ANN architecture was implemented to capture spatial and temporal relationships in the data. The CNN component extracted spatial patterns from numerical features, while the LSTM captured temporal dependencies. The ANN component integrated these features with one-hot encoded categorical data for classification. The initial model achieved modest accuracy, with noticeable challenges in handling class imbalance.

Handling Class Imbalance

To address the class imbalance, multiple techniques were applied:

- **Class Weights:** Weights inversely proportional to class frequencies were used during training to emphasize underrepresented classes.
- **SMOTE:** Synthetic Minority Oversampling Technique was applied to generate synthetic samples for minority classes. This improved the dataset's balance while preserving feature relationships.
- **Random Oversampling:** Minority classes were oversampled to match the majority class, ensuring an even distribution across all classes.

Architectural Enhancements

Several architectural improvements were explored:

- **Bidirectional LSTM:** Adding bidirectionality allowed the LSTM to learn dependencies in both forward and backward directions, enhancing temporal feature extraction. However, this took a lot of time and the accuracy derived from unidirectional LSTM was better.
- **Additional CNN and ANN Layers:** Increasing the depth of the CNN and ANN components enabled the model to capture more complex spatial and combinatorial patterns in the data.

Hyperparameter Tuning

The learning rate was initially experimented with a low value of 0.0001; however, 0.001 was ultimately selected as the optimal value for training. Early stopping and learning rate schedulers were employed to prevent overfitting and dynamically adjust the learning rate during training.

To prevent overfitting and improve the model's generalization, early stopping was employed during training. The EarlyStopping callback was configured to monitor the validation loss (val_loss), with a patience of 5 epochs. **Categorical cross-entropy** was maintained as the loss function due to its effectiveness in handling multi-class classification problems.

Combining Oversampling with Training

Oversampling was applied directly to the combined numerical and categorical features, followed by training the model. This approach enhanced class balance without disrupting the structural relationships within the dataset.

Results

The experiments yielded a comprehensive set of results that highlighted both the strengths and limitations of the proposed approach.

1. Initial Model Performance: The initial CNN + LSTM + ANN model achieved modest accuracy (~60%), with significant performance gaps for minority classes. Validation loss was high, indicating overfitting and poor generalization.

2. Impact of Class Balancing: Class weights and oversampling techniques significantly improved overall accuracy, with SMOTE and random oversampling yielding the best results. Random oversampling combined with a balanced class distribution produced a test accuracy of 84%, the highest in all experiments.

3. Enhanced Architectures: Incorporating unidirectional LSTM and deeper CNN/ANN layers further improved the model's ability to capture complex patterns. However, validation performance remained inconsistent, indicating overfitting to the training set.

4. Class-Wise Performance: While majority classes (e.g., 0, 8, 9) were classified with high precision and recall, minority classes (e.g., 6, 7) continued to exhibit poor performance. This highlighted the need for more advanced techniques, such as data augmentation or hybrid balancing strategies.

5. Combined Approaches: The best-performing model used random oversampling, unidirectional LSTM, and additional CNN/ANN layers. It achieved a test accuracy of **84.11%** and significantly reduced loss.

Conclusion

This project demonstrated the potential of mouse dynamics as a biometric modality for user classification. The combination of CNN, LSTM, and ANN architectures proved effective in capturing spatial and temporal patterns in mouse movement data. However, class imbalance and overfitting posed significant challenges, particularly for minority classes.

Key insights from the experiments include:

- **Feature Engineering:** Derived features like speed, acceleration, and angular changes significantly enhanced model performance.
- **Class Balancing:** Oversampling techniques like SMOTE and random oversampling were crucial for addressing class imbalance, with the latter yielding superior results.
- **Architectural Enhancements:** Deeper networks and unidirectional LSTM improved accuracy but required careful regularization to mitigate overfitting.

Future work could explore advanced balancing techniques, such as hybrid sampling, and integrate additional behavioral features for improved classification. This project underscores the value of mouse dynamics in user authentication and paves the way for further research in this domain.
