ITI105 Project Proposal

User Authentication Using Classical Machine Learning: Leveraging Key Typing Dynamics Behavior

Allen Lee and Jacob Abraham

1. Business Problem statement

Computing devices, including mobile phones, utilize various biometric authentication methods like fingerprints or facial features to identify users. However, these methods rely on specific hardware, which can increase the overall cost. An economical alternative is to authenticate users based on their behavior, such as typing dynamics.

2. Deciding on ML

2.1. Machine Learning Problem statement

We want to create a machine learning model that accurately identifies users based on their typing dynamics, achieving a high level of authentication performance and provide a cost-effective alternative to biometric authentication methods for computing devices.

2.2. Current solution

The current solution involves using biometric authentication methods such as fingerprints or facial features, which require specialized hardware and can be costly.

2.3. Ideal outcome

The ideal outcome would be a reliable and cost-effective user authentication system that eliminates the need for specialized hardware, making it accessible and affordable for a wide range of computing devices.

2.4. Success metrics

- High authentication accuracy: The ML system should achieve a high accuracy in correctly identifying authorized users based on their typing dynamics.
- Cost-effectiveness: The ML system should provide a more affordable alternative to specialized hardware-dependent biometric authentication methods.
- Consistent and stable performance over time: The ML model should demonstrate consistent performance in authenticating users based on their typing dynamics over time. It should maintain accuracy and reliability without significant fluctuations or degradation in performance.
- Robustness against impersonation attacks: The ML model should be resilient to impersonation attempts, where attackers try to mimic or impersonate authorized users' typing dynamics. A robust model can accurately differentiate between genuine and fraudulent user patterns, enhancing the security of the authentication system.

- Positive user acceptance: The ML system should be user-friendly and easily adopted by users, without causing significant inconvenience or frustration.

2.5. Failure metrics

- False accept rate: The ML system should have a low rate of incorrectly accepting unauthorized users as authorized, ensuring that only legitimate users are granted access.
- False reject rate: The ML system should have a low rate of incorrectly rejecting authorized users, minimizing instances where legitimate users are denied access.
- Inconsistency in user authentication: The ML system should consistently authenticate authorized users based on their typing dynamics over time, without frequent fluctuations or erroneous results.
- Vulnerability to impersonation: The ML system should be robust against potential attacks, such as attempts to mimic or impersonate authorized users' typing dynamics.
- The user acceptance and satisfaction levels are low: If users find the authentication process cumbersome, frustrating, or inconvenient, leading to dissatisfaction and a reluctance to use the system, it indicates a failure in achieving user acceptance and adoption.

2.6. Output of the ML model

The output of the machine learning model would be a multi-class decision indicating whether the user's typing dynamics match the authorized user profile or not.

2.7. Using the Output

- The output from the ML model will be made available when a user attempts to authenticate themselves on a computing device. When the user enters their credentials, such as username or email, the ML model will be triggered to analyze their typing dynamics and produce the output.
- Access control: The output will determine whether the user is granted access to the computing device or specific applications. If the ML model output indicates a match between the user's typing dynamics and the authorized user profile, access will be granted. Conversely, if the output indicates a mismatch or a low confidence level, access may be denied.
- Security enhancement: By utilizing the ML model's output for authentication, the system can enhance security by validating the user's identity based on their unique typing dynamics. This helps prevent unauthorized access and protects sensitive data or resources.
- User verification: The output can be used to verify the authenticity of the user. By analyzing the typing dynamics, the ML model can provide confidence in the user's identity, ensuring that the person attempting to access the device or application is indeed the authorized user.
- Fraud detection: The ML model's output can also be utilized to detect potential fraud or impersonation attempts. By analyzing the typing dynamics and comparing it to the authorized user profile, the system can identify inconsistencies or anomalies that may indicate fraudulent activity.

2.8. Heuristics

If we didn't use ML, we would approach the problem of user authentication based on typing dynamics using alternative heuristics and techniques. Here are some possible approaches:

- Rule-based matching: We could create a set of predefined rules and patterns that capture the characteristics of each authorized user's typing dynamics. These rules could include factors such as keystroke duration, inter-key timing, and key pressure. By comparing the user's input to these rules, we can decide whether the typing dynamics match the authorized user's profile.

- Threshold-based matching: We could establish threshold values for various typing dynamics parameters based on the authorized user's profile. For example, we could define acceptable ranges for keystroke durations or inter-key timing. If the user's input falls within these ranges, they are considered a match. However, if the input deviates significantly from the thresholds, the authentication may be rejected.

- Scoring system: We could assign scores to different aspects of the user's typing dynamics, such as keystroke timings or pressure. By comparing the user's scores against a predefined threshold, we can decide on their authentication. The scores could be based on statistical analysis of the typing patterns of the authorized user.

- Multi-factor authentication: Instead of relying solely on typing dynamics, we could combine it with other authentication factors such as password-based authentication, security questions, or additional biometric measures like fingerprint or facial recognition. The combination of multiple factors can provide a more robust authentication mechanism.

2.9. Hypotheses, and assumptions

- Typing dynamics are sufficiently distinct and consistent among individuals, and that there are enough data samples available to train and evaluate the machine learning model effectively.

- Individuals have unique typing patterns that can be captured and used for user authentication, and that these patterns remain consistent over time.

3. Framing as a Classical Machine Learning Problem

3.1. Problem articulation

Our problem is best framed as a Supervised Multi-class Classification task, which predicts whether the user’s typing dynamics match the authorized user profile or not.

3.2. Problem formulation

Given the consideration of starting with simpler problem formulations, the problem of user authentication based on key typing dynamics can be initially framed as a binary classification problem. This formulation focuses on distinguishing between two classes: authorized users and unauthorized users.

Binary Classification Problem Formulation: Given input features derived from key typing dynamics, the ML model aims to predict whether the user attempting authentication is authorized or unauthorized. The model learns to classify the user's typing behavior into one of two classes.

By simplifying the problem to binary classification, it allows for a focused implementation and evaluation of the ML model, making it easier to reason about the model's performance and interpretability. Once a binary classification model is established, it can be extended to handle multiclass classification by incorporating additional classes representing different users or user profiles.

3.3. Dataset

3.3.1. Data design for the Model

To design the data for the ML model to make predictions, we need to identify the relevant features that capture the typing dynamics of users. These features can be extracted from the user's typing behavior during the authentication process and used as input for the ML model. By analyzing and learning from these features, the model can make predictions about the match or mismatch between the user's typing dynamics and the authorized user profile.

Here are some potential features that could be included:

1. Keystroke duration: The duration of time a user takes to press and release each key.
2. Inter-key timing: The time interval between consecutive keystrokes.
3. Key pressure: The force or pressure exerted by the user while pressing each key.
4. Key flight time: The time interval between releasing one key and pressing the next key.
5. Key hold time: The duration for which a key is held down before being released.
6. Typing rhythm: The overall pattern and rhythm of the user's typing, including variations in speed and timing.
7. Key combination patterns: Specific patterns of key combinations or sequences that the user commonly types.
8. Error rate: The frequency or rate at which the user makes typing errors, such as incorrect keystrokes or backspacing.
9. Key release latency: The time interval between pressing a key and releasing it.
10. Digraph or trigraph timing: The timing between specific combinations of two or three consecutive keystrokes.

3.3.2. Source of data

- Benchmark data set: http://www.cs.cmu.edu/~keystroke/
- We plan to create own dataset by developing a simple key sniffer program. The features will be finalized based on experiments.

3.3.3. Determine Easily Obtained Inputs

1. Keystroke duration: The duration of time a user takes to press and release each key.
    - Input Source: Keystroke duration can be measured by capturing the timestamps when each key is pressed and released.
    - Data Pipeline: Developing a data pipeline for keystroke duration would involve capturing the timing information for each key press and release event and calculating the duration based on the recorded timestamps. This process may require implementing keyboard event capturing mechanisms or utilizing platform-specific APIs.
2. Inter-key timing: The time interval between consecutive keystrokes.
    - Input Source: Inter-key timing can be derived from the timestamps of consecutive key press events.

- Data Pipeline: Constructing inter-key timing involves capturing the timestamps of successive key press events and calculating the time difference between them. This process can be implemented by tracking and recording the timing information during typing sessions.

3. Key pressure: The force or pressure exerted by the user while pressing each key.
    - Input Source: Key pressure data can be obtained from specialized hardware sensors capable of measuring the force applied to each key.
    - Data Pipeline: Developing a data pipeline for key pressure would involve integrating and interfacing with the hardware sensors capable of capturing the pressure information. This process may require additional hardware components or integration with existing keyboard technologies.
    - **This feature will not be used as it needs additional hardware.**
4. Key flight time: The time interval between releasing one key and pressing the next key.
    - Input Source: Key flight time can be derived from the timestamps of releasing one key and pressing the next key.
    - Data Pipeline: Constructing key flight time involves capturing the timestamps of key release and key press events and calculating the time difference between them. This process can be implemented by recording the timing information during typing sessions.
5. Key hold time: The duration for which a key is held down before being released.
    - Input Source: Key hold time can be derived from the timestamps of key press and key release events.
    - Data Pipeline: Developing a data pipeline for key hold time involves capturing the timestamps of key press and key release events and calculating the duration between them. This process can be implemented by tracking and recording the timing information during typing sessions.
6. Typing rhythm: The overall pattern and rhythm of the user's typing, including variations in speed and timing.
    - Input Source: Typing rhythm can be derived from the collective timing information of multiple keystrokes.
    - Data Pipeline: Constructing typing rhythm involves analyzing the collective timing information of multiple keystrokes and capturing the patterns and variations in speed and timing. This process can be implemented by aggregating and analyzing the timing data from a series of keystrokes.
7. Key combination patterns: Specific patterns of key combinations or sequences that the user commonly types.
    - Input Source: Key combination patterns can be derived from analyzing the sequence and frequency of specific key combinations.

- Data Pipeline: Developing a data pipeline for key combination patterns involves capturing and analyzing the sequence of key combinations, identifying frequently occurring patterns, and recording their occurrence. This process can be implemented by tracking and analyzing the key sequences during typing sessions.

8. Error rate: The frequency or rate at which the user makes typing errors, such as incorrect keystrokes or backspacing.
   - Input Source: Error rate can be derived from analyzing the occurrence and frequency of typing errors during typing sessions.
   - Data Pipeline: Constructing error rate involves monitoring and recording typing errors, such as incorrect keystrokes or backspacing events, and calculating their frequency or rate. This process can be implemented by tracking and analyzing the occurrence of errors during typing sessions.
9. Key release latency: The time interval between pressing a key and releasing it.
   - Input Source: Key release latency can be derived from the timestamps of key press and key release events.
   - Data Pipeline: Developing a data pipeline for key release latency involves capturing the timestamps of key press and key release events and calculating the time difference between them. This process can be implemented by tracking and recording the timing information during typing sessions.
10. Digraph or trigraph timing: The timing between specific combinations of two or three consecutive keystrokes.
    - Input Source: Digraph or trigraph timing can be derived from analyzing the timing between specific combinations of consecutive keystrokes.
    - Data Pipeline: Constructing digraph or trigraph timing involves identifying specific key combinations, capturing the timing information between them, and recording the timings for analysis. This process can be implemented by tracking and analyzing the timing between consecutive keystrokes during typing sessions.

3.4. Methods

3.4.1. Feature Extraction

Preprocess the raw data to extract relevant features from the typing dynamics, such as keystroke duration, inter-key timing, key pressure, etc. This may involve calculating durations, time intervals, or aggregating statistical measures from the raw typing data.

3.4.2. Data Split

Split the dataset into training and testing sets. The training set will be used to train the machine learning model, while the testing set will be used to evaluate its performance.

3.4.3. Model Selection

Choose a suitable supervised machine learning algorithm for binary or multi-class classification, such as logistic regression, support vector machines (SVM), decision trees, or random forests.

A suitable ML approach for the user authentication problem based on key typing dynamics is to utilize a supervised learning algorithm, such as a multi-class classification algorithm. In a multiclass classification scenario, the ML model aims to classify input instances into one of several possible classes. In this case, each class would represent a specific user or user profile. The model would learn to differentiate between the typing dynamics patterns associated with different users and make predictions about the user identity based on the input features.

Other ML approaches could also be considered for this problem:

- Decision Trees: Decision trees are intuitive and interpretable models that can handle multiclass classification problems effectively. They partition the feature space based on the input features and create a tree-like structure to make predictions. Decision trees can handle both numerical and categorical features and handle non-linear relationships.
- Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees. It can handle multiclass classification and provides robustness against overfitting. Random Forest models can capture complex interactions between features and perform well with high-dimensional data.
- Naive Bayes: Naive Bayes algorithms are probabilistic classifiers that assume independence between features. Despite their simplistic assumption, Naive Bayes classifiers perform well in many multiclass classification problems. They are computationally efficient and handle high-dimensional data effectively.
- Gradient Boosting: Gradient Boosting algorithms, such as XGBoost or LightGBM, are powerful ensemble methods that sequentially train weak classifiers and combine them into a strong predictor. They excel in handling imbalanced datasets and offer good predictive performance in multiclass classification scenarios.
- While Support Vector Classification (SVC) can still be valid for multiclass classification, it inherently performs binary classification and requires extension techniques to handle multiclass scenarios, such as One-vs-One or One-vs-Rest approaches. However, depending on the dataset size, complexity, and other factors, other algorithms like decision trees, random forests, neural networks, naive Bayes, or gradient boosting may provide more straightforward implementation or better performance.

Experiments will be conducted with different algorithms and their performance will be compared using appropriate evaluation metrics to identify the most suitable approach for the user authentication problem based on key typing dynamics.

3.4.4. Metrics

- Accuracy: The proportion of correct predictions (both true positives and true negatives) out of the total number of predictions. It gives an overall measure of the model's correctness in classifying authorized and unauthorized users.
- Precision: The ratio of true positives to the sum of true positives and false positives. Precision measures the accuracy of the model in correctly identifying authorized users out of all the instances it predicted as authorized. It assesses the model's ability to minimize false positive errors.

- Recall (Sensitivity or True Positive Rate): The ratio of true positives to the sum of true positives and false negatives. Recall quantifies the model's ability to correctly identify authorized users out of all the instances that are authorized. It evaluates the model's effectiveness in minimizing false negative errors.
- F1-score: The harmonic mean of precision and recall. It provides a balanced measure of both precision and recall, giving an overall assessment of the model's performance.
- False Acceptance Rate (FAR): The ratio of false positive predictions to the sum of false positives and true negatives. It represents the proportion of unauthorized users incorrectly identified as authorized. A lower FAR indicates a more secure model.
- False Rejection Rate (FRR): The ratio of false negative predictions to the sum of false negatives and true positives. It measures the proportion of authorized users incorrectly identified as unauthorized. A lower FRR indicates a more accurate model in correctly authenticating authorized users.

3.4.5. Training

Train the selected model on the labeled training dataset. The model will learn the patterns and relationships between the input features (typing dynamics) and the corresponding labels (authorized or unauthorized).

3.4.6. Model Evaluation

Evaluate the trained model's performance on the testing set using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. This step helps assess how well the model generalizes to unseen data and its ability to accurately classify the user as authorized or unauthorized based on their typing dynamics.

3.5. Deployment

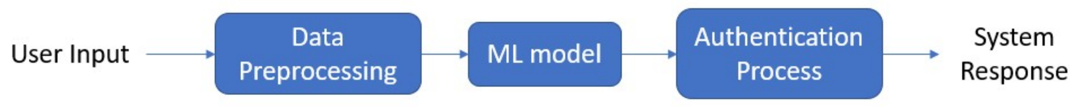
Once the model shows satisfactory performance, it can be deployed in a production environment for user authentication. The model will take the input features (typing dynamics) from a user attempting to authenticate and provide a prediction of whether they are an authorized user or not.

In the overall solution, the ML model for user authentication based on key typing dynamics is used as a component of the system's authentication process. Here is an overview of how the model is used within the solution:

1. User Input: The user attempting to authenticate provides their key typing input, typically by typing a sequence of characters on a keyboard.
2. Data Preprocessing: The system captures the raw typing data from the user's input and preprocesses it to extract relevant features, such as keystroke duration, inter-key timing, key pressure, etc. These features serve as input to the ML model.
3. ML Model: The ML model, trained on labeled data, receives the extracted features as input and predicts whether the user is authorized or unauthorized based on their typing behavior. The model utilizes the learned patterns and relationships between the input features and the corresponding labels.
4. Decision: The model outputs a binary decision indicating whether the user is authorized or unauthorized based on their typing dynamics.

5. Authentication Process: The model's decision is then used as part of the overall authentication process. If the model classifies the user as authorized, the authentication is successful, and the user is granted access to the system. Conversely, if the model classifies the user as unauthorized, the authentication is rejected, and the user is denied access.
6. System Response: The system responds accordingly based on the authentication outcome. If authentication is successful, the user gains access to the desired resources or functionalities. In the case of rejection, the user may be prompted for further verification or denied access, depending on the system's requirements.

Architecture Diagram:



For this project, “Streamlit” or “Gradio” will be used to provide a web interface for inference.

3.6. Monitoring and Iteration

Continuously monitor the model's performance and collect feedback to further refine and improve its accuracy and robustness. This can involve retraining the model with updated data or exploring different algorithms to optimize performance.