**KEYSTROKE DYNAMICS MODULE**

**DONE BY,**

CB.EN.U4CSE18310 - Chandravadhana. A  
CB.EN.U4CSE18328 - Juvvala Hemantha Sai Sandhya  
CB.EN.U4CSE18333 - Akhila Kumari K

**INTRODUCTION**

Keystroke dynamics basically deals with the analysis of keystroke patterns of a user. It is classified as a **behavioural biometric.** In one of its many use cases, keystroke dynamics can enable proctored exams to provide non-intrusive real-time authentication of users.

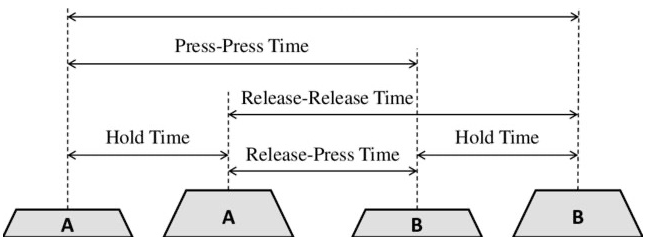
Capturing the duration between 2 keypresses, the duration of pressing down a key, and the duration between the current key release and the next key press can provide insights about the user.

The next time the user logs in, by comparing his/her current typing pattern with his/her previous typing patterns, the platform can authenticate whether the logged in user is legitimate or fraudulent.

It is a classification based problem, where we have matched different typing patterns and clustered them under an umbrella.

**ABOUT THE TECHNIQUE – GENERATE FEATURES**

* Usually a keystroke logger would just log the timestamps when the key is pressed and released.
* Now, by itself, these timestamps would mean nothing.
* However, we can gain insights on user typing patterns by creating features such as:
  1. Press-press duration (PPD)
  2. Hold duration (HD)
  3. Release-press duration (RPD)
* The image shows the duration calculations for pressing 2 keys — A and B.
* Here, the smaller keys represent the key press event and the larger ones represent the key release event. These durations can be used in understanding the user’s typing patterns.



**PART 1: IMPORTING THE NECESSARY LIBRARIES**

* All the necessary libraries are imported for efficient pre-processing and data cleaning.

**PART 2: LOADING THE DATASET**

* First, we read the data from csv.
* The dataset ‘train.csv’ and ‘test.csv’ has been imported.

**PART 3: DESCRIPTION OF THE DATASET**

* The dataset: Keystroke dynamics challenge 1 | Kaggle has been used in the jupytr notebook.
* This dataset captures typing attempts of **110 users.**
* Each user has attempted **8 times** to type the string ‘united states’ and the corresponding **timestamps** of key press and release relative to the first key press have been captured.
* Number of rows in training dataset: 880
* Number of users for which training data is present: 110
* No. of rows in test dataset: 220

**PART 4: DATA PRE-PROCESSING**

* Checking for missing values. But according to the output, there are no missing values.
* **Feature scaling:**
  + We need to perform **Feature Scaling** on the press and release quantities since they have values in hundreds and thousands.
  + **MIN – MAX Normalization** is done to scale the data.
  + **Min**-**max normalization** is one of the most common ways to **normalize** data.
  + For every feature, the **minimum** value of that feature gets transformed into a 0, the **maximum** value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

**PART 5: DEMONSTRATION**

* This data can’t be used directly.
* **Feature generation** is done:
  + Press-press, hold, release-press durations from this dataset.
* Taking a sample of 5 users but this data can’t provide useful information directly. More useful insights can be drawn only from the **time durations** between these timestamps.
* **PRESS – PRESS DURATION**
  + Time between 2 continues presses.
* **RELEASE PRESS DURATION**
  + Time between a release of 1 key and press of other.
* **HOLD DURATION**
  + Time between a press and release of a key.

**PART 6: EXPLORATORY DATA ANALYSIS (EDA)**

* We try to explore this generated data and try to understand the data by its face value.
* Before further analysing the code, let’s consider only 5 of the 110 users i.e. 5x8 = 40 typing patterns.

**WHAT IS SWARM PLOT?**

* Swarm plots depict all the data points. Swarm plots attempt to avoid obscuring points by calculating non-overlapping positions instead of adding random jitter.
* Using seaborn’s swarm plot function, we can generate swarm plots of release-press, press-press, and hold duration for these 5 users.

**PART 7: INFERENCE FROM SWARM PLOT**

* Initially, only two keys/letters were analysed 🡪 ‘U’ and ‘N’. So two key presses were made 8 times for analysis i.e 2 X 8 = 16.
* Similarly, the second graph two keys/letters were analysed 🡪 ‘N’ and ‘I”.
* From the above two graphs, not much can be inferred because it doesn’t give a solid typing pattern.
* Therefore, all 13 keys/letters were analysed 🡪 ‘United States” i.e 13 X 8
* As seen from the swarm plots, the press-press duration, release-press durations are roughly the same across all users.
* Thus, directly using an average duration will not be helpful.
* However, hold duration is roughly different for each user which is correct since each user has a different typing speed according to his familiarity with typing.
* Therefore, using histograms to check if any variations could be identified.
* Since, each typing pattern consists of 13 keystrokes, scatter plots and line plots can be used.

**PART 8: CONVERTING ROW FEATURES TO COLUMN FEATURES FOR PLOTTING**

* Now, each row in the dataset is a typing pattern corresponding to a user, but if we want to analyse the typing patterns across users (timestamps connected by lines), therefore we need to bring these row features into a single column.
* For this purpose, along with few other pandas functions, we need to use the wide\_to\_long feature of pandas.
* For a particular user, there are 12 X 8 times (=96), the typing analysis is done. Therefore, for 5 users: 96 X 5 times (=480). Hence, the shape is 480 X 3.
  + So, for user: one, the iloc values ranges from 0:15, where each row is the analysis between two consecutive keys/letters. Since there are 13 keys in “United states”, we get 12 comparisons.
  + Similarly for user: 5, the iloc values ranges from 32:29.

**PART 9: PLOTTING PRESS, PRESS DURATION VS PRESS TIMESTAMPS USING PLOTS AND LINES**

* Note, here there are 40 line plots i.e. 8 line plots per user and 5 users are being considered.
* Notice the jagged lines for user 4 (red lines).
  + The PPD suddenly increases and then becomes very low for the next key.
  + This means that this user waits for a relatively longer time before typing in 2 keys back to back.
  + So, we could say this user typically types in groups of 2 keys.
* Now, imagine the level of security if we use sophisticated algorithms to generate insights and devise authentication techniques.
* This would be something that the user wouldn’t even need to remember.

**PART 10: FACTORS IMPACTING THE TYPING PATTERNS**

* Time of day for typing.
* Lighting condition of keyboard while typing (especially for non-touch typers).
* Level of familiarity with the keyboard layout.
* Location where typing is done.
* The arm, sitting position of the typer.
* Stress of the typer.

**PART 11: DATA PREPARATION**

* Instead of directly using these durations as inputs, we could group these durations into histograms which would represent groups of different typing speeds.
* For e.g. a slow typer would have his keystroke durations falling in the histogram bucket of larger durations.
* At the same time, a fast typer (or perhaps a touch typer) would have his keystroke durations falling in the histogram bucket of smaller durations.
* Thus, we will convert the entire training and test data by using this histogram technique.
* Note, wide\_to\_long function transforms the dataframe by arranging all the row values of press-\* into a single column.

**PART 12: BUCKETING USING HISTOGRAMS**

* Now, let’s view the resulting histograms to determine which buckets can be created so as to split the data across them uniformly.
* Instead of determining buckets, a pandas function called   
  “qcut” is used.
* In this function we specify the column name over which we want to perform the binning/bucketing.
* This function checks the distribution of values in the specified column and divides the distribution into equal frequencies with q number of bins.
* This ensures that each bin has equal frequency and we can get more meaningful bins.