
# User Identity Verification via Machine Learning, using Keystroke and Mouse Dynamics

Nicholas Kecojevic
1413706

Department of Computing and Technology
Anglia Ruskin University

# Abstract

*In our current society, we require an ever-growing increase in security, with the majority of the modern world moving to digital platforms, the risk of malintent to the everyday user has never been higher. Research previously done shows that the concept of using mouse and keystroke biometrics is not a new one, but no research into said systems uses the combination of both mouse and keystroke biometric systems, but rather one or the other. This report explores the ability to use both mouse and keystroke dynamics as a means of authentication for users. The aim was to produce a piece of software that distinguishes user characteristics from input data and feed the features into a machine learning model to produce an identity. Based on a review of the previous studies and reports completed on the topic of mouse and keystroke biometrics, a simple data collection took place. Participants were instructed to complete a game of solitaire and commence a 5-minute typing test, to simulate everyday computer usage. The input data was collected and analysed to retrieve unique features that were fed into a machine learning model to identify new input data.*

*The results indicate that there does lie the ability to use mouse and keystroke biometrics as a means of verification, but as they are a soft biometric method, there also lies the possibility of inaccurate results. With the addition of the difficulties faced for larger userbases, with using such an approach like the one in this report, it is recommended that mouse and keystroke biometric systems should only be used for a small userbase, where levels of high-security clearance are needed. Further research is needed to identify greater feature extraction and removal of outliers to bring the possibility of software like the one created in this report to the masses for everyday consumer use.*

# Table of contents

# List of Figures, Diagrams & Equations

# 1. Introduction

With this report, I aim to investigate the possibility of using mouse and keystroke biometrics in current systems. The report will serve as an experiment to see the effectiveness of using these measures of authentication and if they will suit their intended purpose.

## 1.1 Aims of the study

The success of this project will be based upon one major factor, is it possible to build a machine learning system, that can take in user characteristics and predict an individual, to a satisfactory standard by comparing the data to a previously harvested machine learning model.

Ideally, the project ought to:

- Log all user input data, then extract a set of generated features from said input data.
- Feed said user features into a machine learning model and output a likely identity of the user.
- Produce an application which could ultimately increase security for organisations and individuals.

## 1.2 Background work

During the authors' studies at university, the author has wondered there would ever be a be-all and end-all to security measures. It would be foolish to think that in an industry that has grown to 35 times its size between 2004 and 2017 (Morgan, 2017) that such a solution is possible. The author hopes that this report and the software produced could stand as a basis for future research and production into methods that can further the divide between cybersecurity and offensive measures.

Biometrics is the measurement and statistical analysis of people's unique physical and behavioural characteristics, biometrics are often divided into two modalities, hard and soft biometrics. Hard biometrics are the traditional biometrics, such as fingerprints or facial scans, they are considered accurate, whereas, soft biometrics, such as gait analysis, comes with a certain degree of inaccuracy as there is a chance of higher false acceptance rate and false

rejection rate. Behavioural biometrics falls under the latter modality, soft biometrics. Soft biometrics are not accurate enough to use as a stand-alone authentication system, but instead to use as a secondary level of authentication.

Behavioural biometrics have been in use since as early as the 1860s, the first telegraph was sent in 1844, as operators became used to the systems, their behavioural patterns emerged, each operator developed their unique signature from their tapping rhythm. In World War II, messages sent using Morse code were identified using a methodology called 'The Fist of the Sender'. The recipient of a transmission could identify the sender of the message, by recognising the latencies in the dots and dashes of the Morse code transmission. (Jenkins et al., 2011)

The earliest known references to computer input devices as a means of authentication comes from a research report titled 'Authentication by Keystroke Timing: Some Preliminary Results' (Gaines et al., 1980). The study took 7 typists and analysed keystroke timings when the typists were presented with a paragraph to copy. The procedure was repeated 4 months later, and the results were analysed to determine the best possible method of distinguishing between participants. The authors found five digraphs, which when considered together could successfully identify between the participants.

The method of mouse and keystroke dynamics have come a long way since 1980, with the advancement of computer systems, analysing thousands of mouse actions and keystrokes becomes a relatively inexpensive task. As a result, we can include as many defining characteristics as we see fit, increasing the likelihood of accurate identity verification.

In 2016, Balabit, a Hungarian IT security firm, published a dataset called the 'Balabit Mouse Dynamics Challenge data set' (Fülöp et al., 2016). The Balabit data set contained 10 user training files, and 10 user test files, containing the details of the timing and position locations of the mouse. The dataset served as a method of improving and evaluating the performance of

behavioural biometric algorithms based on mouse dynamics for user authentication/identification purposes. The dataset has been used by many research papers and journal entries as a benchmark dataset, unfortunately, the dataset only includes mouse information, which therefore for this project, makes it unsuitable for its intended use. The author instead, used the information gathered by papers using this dataset to improve their methods of behavioural biometric algorithms.

More recently, Machine Learning approaches have been implemented to analyse the data set. A recent paper titled Mouse Dynamics as Continuous User Authentication Tool (Kahn et al., 2020) looks at the different algorithms that can be used for mouse dynamics. As well as reviewing previous works of evaluating studies. The authors came upon the conclusion that as there exists no consistent dataset, experimental conditions or methodologies for the experiments it is not possible to conduct a sound comparison of the classifiers. Therefore, perhaps it would be imperative for a varied dataset to be created containing a consistent approach to best evaluate further studies done in the subject of mouse and keystroke biometrics.

The article 'An insider threat detection approach based on mouse dynamics and deep learning' (Hu et al., 2019), explores the difference in recognising mouse actions, with most papers only including three, these three being mouse drag actions, mouse move actions and point-click actions. Hu et al explore the possibility of using six actions, the above mentioned three as well as click actions, and scroll actions and stay actions. These six actions are defined as such;

| Operation | Definition |
|---|---|
| Mouse move | An action defined as a movement of the mouse not ended in a click |
| Point click | An action defined as a movement of the mouse, ending in a click |
| Mouse drag | An action defined as the press of a mouse button, then the subsequent movement, concluded with a release of the mouse button |
| Click | An action defined as a press and release of the mouse button with no movement |
| Scroll | An action defined as a scroll by the user |
| Stay | An action defined as the length of time of the pause in between each action |

(Diagram 1.1 – Hu et al list of mouse operations and definitions)

Hu et al state that most researchers ignore the 3 latter actions, but these are still operations of the mouse movement behaviour, and to ignore them would be to ignore useful information, but one could also argue that the data extracted from these last 3 operations are severely dependent on the task at hand, and would scale poorly in scenarios in which the user is authenticated completing a task that was none alike the training dataset.

Feature extraction is a relatively documented subject with (Yıldırım and Anarım, 2019) detailing 68 total features for mouse movement sequences for extraction of characteristics from the Balabit data set, one can argue that 68 features for each action could be too many and the risk of saturating the useful data with un-useful data exists. Perhaps features such as the time of the action, don't reflect on the user but rather the specific actions. Including features such as this could result in an unfavourable outcome in the machine learning models.

In the research article titled 'Understanding keystroke dynamics for smartphone user' (Lee et al., 2018), Lee et al cover the usage of biometric data as a multifactor authentication combined with pin and pattern recognition on mobile phone devices. They state that the biggest problem that lies with biometric data is that is a leakage were to occur, the user's biometric data as a secure means of authentication is invalidated, possibly for their entire life. Changing such learned behaviours such as mouse and keystroke movement would be almost impossible.

One paper explores the possibility of using both keystroke and mouse dynamics, 'Combining both Keystroke and Mouse dynamics for continuous user authentication and identification' (Mondal and Bours, 2016). Mondal and Bours propose a continuous system of authentication and identification, with users being constantly monitored to analyse the input stream to detect variances from the intended user, with the project able to successfully identify the attacker, if their data exists in the machine learning model. Using a continuous system does make the possibility of malintent much more difficult but does impose restrictions on users, that might not feel comfortable having their every move recorded and analysed.

In the paper ‘Adversarial attacks on remote user authentication using behavioural mouse dynamics’ (Tan et al., 2019), we see the possibility of using machine learning to trick a system into authenticating an incorrect user. The authors conclude that neural network-based attacks are better performing than a statistics-based approach and that authentication models can be adversely affected. The approach taken place in this report already assumes that the attackers have access to all the users' input data, while a man in the middle attack can occur when a user is in a remote desktop connection, for the majority of the global population who uses computers, it would be highly improbable that a would-be attacker would have access to a user’s mouse and keystroke movements. The report did also conclude that if an attacker has access to the architecture of the authentication model, its robustness would be greatly affected, but that in a realistic setting, attacking a system based on behavioural features is harder than copying a fingerprint. Therefore, it is safe to assume that implemented correctly, and securely a behavioural biometric system would be sufficient as a secondary means of authentication.

As of the submission of this report, there exists no application in the public realm that successfully implements mouse and keystroke biometrics as a means of authentication. One such software exists for keystroke biometrics (typingdna.com). One might assume that this technology might be used in a manner within a secure organisation (especially as the United States Military has used a form of it since WWII), that has not disclosed its existence. Every paper on the topic seems to conclude the same thing, that mouse and keystroke dynamics are a suitable form of an authentication method, when implemented correctly.

# 2. Description of Methodology

10 participants were gathered for data collection, participants had a range of different background experience and familiarity with computer systems, with an age range of 18-50. Half of the participants completed the task on the same mouse and keyboard and the other half had personal mouse and keyboards. This was done to ensure we didn't have any biases affecting our results negatively. The task for participants to complete was to play 2 full games of solitaire to simulate average mouse movement, with 3 actions in particular, as defined by Hu et al. Mouse move actions, mouse drag actions and point click actions. Once the participant concluded the games of solitaire, the typing portion of the experiment started, participants were instructed to write out the body of text in front of them for 5 minutes. Half of the participants received the same body of text and the other half were individual bodies of text, again, this was done to remove biases.

While the task was occurring, the software produced in this report was recording all the input data into a raw log file, once the task and recording session has concluded, the resulting log file is converted from its raw format into two 'readable' files, one for mouse and the other for the keyboard. These two separate files are then processed by the software to determine the unique features from the dataset, machine learning models are built using a binary random forest classifier, with a 1 for the desired user traits, and a 0 for all the other undesired user traits. When verifying identity, a similar process occurs, an instance of the logging software is initialised, with the results being fed straight to a data frame that is processed and fed into the different machine learning models, the results are taken and sorted and the output is a predicted user and percentage match.

All of the software needed for this report were produced by the author, the programming language Python was used to build all aspects of the software. Libraries such as pynput were used to collect input data, and then individual features were extracted using NumPy and pandas

to manipulate the data. Finally, the results were fed into scikit-learn libraries for machine learning models, which produced the results detailed in this report. A varied usage of mouse's and keyboards were used for this experiment.

## 2.1 Mouse Data Feature Extraction

To build an accurate machine learning model, detailed feature extraction was needed. For mouse input, each entry in our feature set was defined by one action, of this one action several features were extracted. The first being the straightness of the action which is defined by the equation:

$$S = \frac{\sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}}{\sum_{i=1}^{N} \sqrt{dx_i{}^2 + dy_i{}^2}}$$

(Equation 2.1 – Straightness of an action)

The straightness of an action is the optimal distance divided by the actual distance taken. $x$ represents an array of the $x$ values of an action, $y$ is the same for $y$ values, $dx$ and $dy$ are arrays containing the difference of the current $x$ and $y$ location and the previous. The optimal distance is calculated using Pythagoras theorem to work out the distance between the start and end of the action, whereas the actual distance is the sum of all of the distance travelled throughout an action. This equation results in a value between 0 and 1, with values closer to 1 being perfect straightness and values closer to 0 being nothing alike the optimal distance. The next feature is the initial acceleration; this is defined using the difference in velocity which is defined by equation 2.2, where $vx$ and $vy$ are defined by the velocity of the $x$ and $y$ axis respectively;

$$dv = \sqrt{vx^2 + vy^2}_i - \sqrt{vx^2 + vy^2}_{i-1}$$

(Equation 2.2 – Change in velocity)

We use $dv$ to work out the starting acceleration like such;

```
startAcceleration = 0
for 0 > i >= numberOfEntriesInAction:
      dv = v[i] – v[i-1]
      if dv > 0 and accelerating == True:
            startAcceleration = startAcceleration + dv
      else:
            accelerating = False
```

The above pseudocode shows us that when the difference in velocity is greater than 0, therefore increasing in speed, the value of $dv$ is added to the acceleration value. Once the change in velocity is negative, the acceleration is no longer occurring and dv is no longer added to the value. The remaining features are all generated by finding the mean, standard deviation, minimum and maximum of the arrays; $vx, vy, v, \omega, a, j$ and $c$. Where $vx, vy$ and $v$ are values of each velocity for a sub-action. $\omega$ which is the array containing the angular velocity of each sub-action which is defined by equation 2.2, where $d\theta$ represents the difference in theta between each sub-action and $dt$ the difference in time;

$$\omega = {d\theta}/{dt}$$

(Equation 2.3 – Angular velocity of an action)

$a$ represents the array containing the values for each sub-action of the difference in velocity over the difference in time, shown in equation 2.4, where $dv$ represents the difference in velocity;

$$a = {dv}/{dt}$$

(Equation 2.4 – Acceleration of an action)

$j$ represents the array containing the 'jerkiness' values of each sub-action, it is defined by the difference in angle over the difference in time, shown in equation 2.5, where $da$ is defined by the difference in angle using inverse tangent of the $x$ and $y$ values;

$$j = {da}/{dt}$$

(Equation 2.5 – Jerkiness of an action)

Our last feature is $c$ which is the array containing the curvature of each of the sub-actions. It is defined by the difference in angle over the difference in path length, shown in equation 2.6, where $dp$ is defined by the length of the current sub-action;

$$c = {}^{\mathrm{da}}/_{dp}$$

(Equation 2.6 – Curvature of an action)

## 2.2 Keyboard Data Feature Extraction

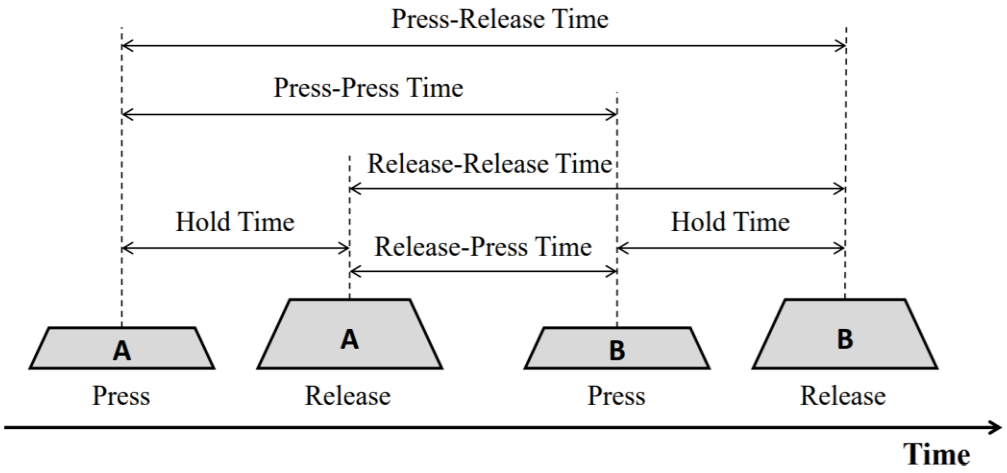
Keyboard data feature extraction was a much more difficult process, with input data only having 3 values, time, key and press or release. The first step was to create a new data frame containing the 'completed' keypress and releases, which gave us a more solid data set to extract from. The first features came from identifying which finger was used to activate the key, the keyboard was split into 10 zones detailed as such in diagram 2.1.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ` | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | - | = | BACKSPACE |
| Tab | Q | W | E | R | T | Y | U | I | O | P | [ | ] | # |
| CAPS | A | S | D | F | G | H | J | K | L | ; | ' | ENTER | |
| SHIFT | Z | X | C | V | B | N | M | , | . | / | SHIFT | | |
| CTRL | START | ALT | SPACE | | | | | | | ALT | FN | CTRL | |

(Diagram 2.1 – Identifying finger and key zone relations)

Where the keyboard is split into two for the left and right hands, the green, yellow, red and blue zones on the left become zones 1, 2, 3, 4 respectively. The purple zone becomes zone 5, and on the right-hand section, the remaining zones become 6, 7, 8 and 9. Zone 10 is any key that did not fall into any of the above-detailed zones. To make the machine learning model not value any certain keypress as a higher rated value, 10 columns were created in our data-set, containing zeros apart from if the key matched a particular zone then that column would register as a 1.

The following metrics of key activation times were established in the paper 'The Wolf of SUTD' (Harilal et al., 2018) as shown in diagram 2.2;




(Diagram 2.2 – Key actuation and de-actuation time metrics)

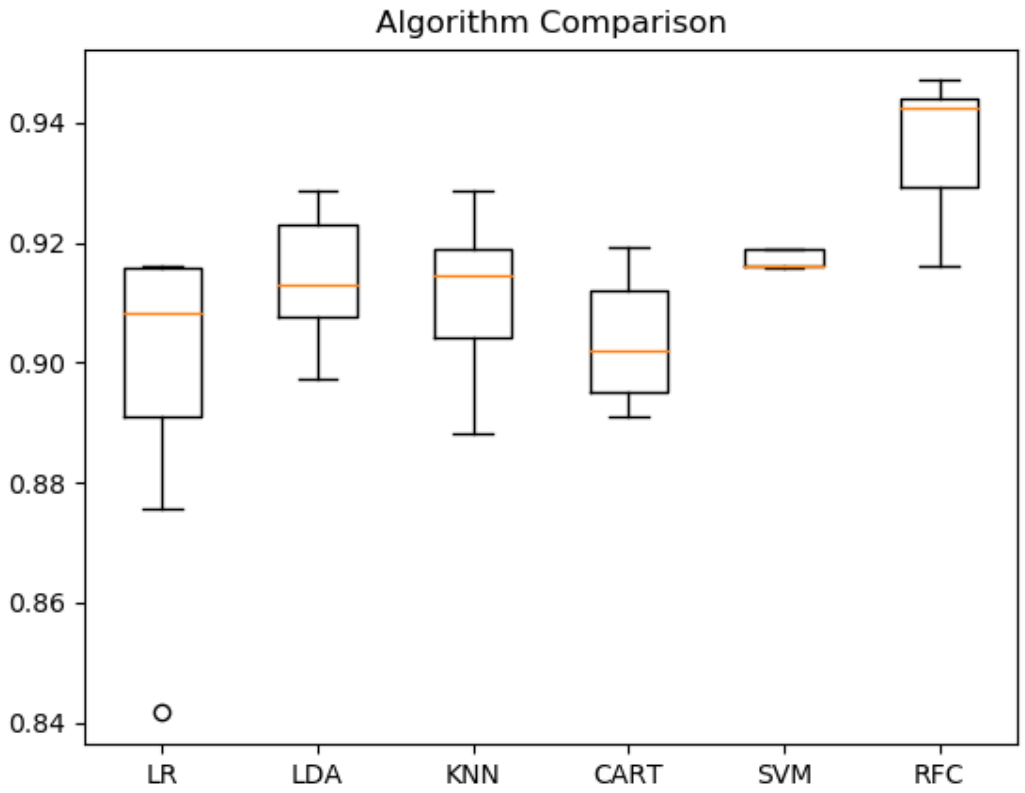
This gives us 5 features which are shown as $H$ (Hold time), $PP$ (Press-Press time), $PR$ (Press-Release time), $RP$ (Release-Press time) and $RR$ (Release-Release time). These metrics are calculated by finding the difference in time between the key actuation and de-actuation time. Another feature is defined by $KPS$ (Keys Per Second), which is calculated by the number of keys that have been pressed within the last full second. The last feature is defined by $RAT20K$ (Rolling Average Time of the surrounding 20 Keys), this is calculated using the following equation, in which $pt$ is defined by the time of the press of the key;

$$RAT20K = \sum_{i=-10}^{10} pt_i - pt_{-10}$$

(Equation 2.7 – Rolling average of surrounding 20 keys time)

## 2.3 Classifiers

During the creation of the software element of this report, many different classifiers were chosen and analysed to find the most accurate models to evaluate the effectiveness of the methods taken, in figure 2.1 we see each algorithms accuracy and their standard deviation for the detailed classifiers.




(Figure 2.1 – Analysis of different classification methods)

With the tested classifiers being, Logistic regression, Linear discriminate analysis, K-Nearest neighbours, Decision tree learning, Support vector machine and Random forest classification respectively. Random forest classification was the chosen model as the outcome was discernibly higher for all of the datasets.

## 2.4 Methodology Evaluation

The chosen feature set for both machine learning models were using every possible discerning feature from the input streams, it is the authors' opinion that there may be a few more features to be extracted but they will not impact the results as heavily as the chosen features.

Due to the current pandemic occurring, the author struggled to find participants for this report, 20 participants completing further tasks than the completed two was the ideal goal. One can argue that the tasks completed do not accurately reflect everyday computer usage, as people are not typically being timed when they type, and everyday mouse usage is not that similar to

solitaire playing, but due to the constraints of the lockdown rules set in place due to the pandemic, it became impossible for the author to supervise the participants completing the tasks. Therefore, it is the author's opinion that the chosen methodology was the best possible action as almost everyone is familiar with the game solitaire and should not struggle with any of the task's requirements.

# 3. Results

Building the machine learning models allow us to test its results. Below is a table showing the precision and recall values for each users' models, for the binary class. 1 being the desired class and 0 being the undesired class. (every other users' features) Precision is the number of true positives over all the true and false positives, recall is the number of true positives over the true positives and false negatives. Precision essentially means the percentage of relevant results, whereas recall refers to the percentage correctness of the predicted results. Numbers closer to 1 signify the model is making fewer incorrect predictions.

| | *Mouse* | | | | *Keyboard* | | | |
|---|---|---|---|---|---|---|---|---|
| | **1 (Desired)** | | **0 (Undesired)** | | **1 (Desired)** | | **0 (Undesired)** | |
| | ***Precision*** | ***Recall*** | ***Precision*** | ***Recall*** | ***Precision*** | ***Recall*** | ***Precision*** | ***Recall*** |
| ***User1*** | 0.76 | 0.37 | 0.94 | 0.99 | 0.90 | 0.36 | 0.89 | 0.99 |
| ***User2*** | 0.91 | 0.94 | 0.98 | 0.97 | 0.80 | 0.53 | 0.91 | 0.97 |
| ***User3*** | 0.75 | 0.50 | 0.93 | 0.97 | 0.60 | 0.13 | 0.88 | 0.99 |
| ***User4*** | 0.75 | 0.68 | 0.95 | 0.97 | 0.67 | 0.18 | 0.88 | 0.99 |
| ***User5*** | 0.94 | 0.89 | 0.99 | 0.99 | 0.81 | 0.38 | 0.91 | 0.99 |
| ***User6*** | 1.00 | 0.02 | 0.95 | 1.00 | 0.59 | 0.23 | 0.93 | 0.99 |
| ***User7*** | 0.66 | 0.20 | 0.92 | 0.99 | 0.66 | 0.12 | 0.93 | 0.99 |
| ***User8*** | 0.99 | 0.87 | 0.99 | 1.00 | n/a | n/a | n/a | n/a |
| ***User9*** | 0.75 | 0.18 | 0.97 | 1.00 | 0.77 | 0.17 | 0.96 | 1.00 |
| ***User10*** | 1.00 | 0.02 | 0.95 | 1.00 | 0.57 | 0.13 | 0.95 | 0.99 |
| ***Average*** | **0.85** | **0.47** | **0.96** | **0.99** | **0.71** | **0.25** | **0.92** | **0.99** |

(Figure 3.1 – Machine learning model precision and recall rate)

As we can see in the table above, the models have almost no issue identifying the undesired features, but for both mouse and keyboard, the models have a respectable precision but the recall is on the lower side. This essentially lets us know that our models will be very strict when predicting identity, but we should have a low rate of false positives. Testing another dataset unseen by the models, user1's test file is compared against each of the models, the output match percentage is sorted and displayed as such in table 3.2;

| | Combined match | Mouse match | Keyboard match |
|---|---|---|---|
| **User1** | 43.53% | 48.66% | 38.4% |
| **User7** | 2.90% | 4.46% | 1.33% |
| **User2** | 2.67% | 0% | 5.33% |
| **User6** | 1.2% | 0% | 2.4% |
| **User5** | 0.67% | 0% | 1.34% |
| **User8** | 0.67% | 1.34% | 0% |
| **User9** | 0.45% | 0.89% | 0% |
| **User4** | 0.4% | 0% | 0.8% |
| **User10** | 0.36% | 0.45% | 0.27% |
| **User3** | 0.13% | 0% | 0.27% |

(Figure 3.2 – Predicting ownership of user1 testing dataset)

From the above table, we can indeed confirm that the models have a very low false acceptance rate for users that are considered undesirable within the model, but the model suffers with predicted the correct user of user 1, with the combined predictions of the models only being 43% sure that it is indeed user1. Compared to the results of the other predictions, 43% is many factors greater and therefore has successfully predicted ownership of the dataset. Comparing this table of results to another, the predicted ownership of user2 testing data, as detailed below;

| | Combined match | Mouse match | Keyboard match |
|---|---|---|---|
| **User2** | 80.87% | 94.29% | 67.45% |
| **User8** | 3.28% | 2.76% | 4.79% |
| **User1** | 3.23% | 1.17% | 5.28% |
| **User4** | 1.58% | 0.23% | 2.92% |
| **User6** | 1.04% | 1.73% | 0.35% |
| **User9** | 0.76% | 0.17% | 1.24% |
| **User10** | 0.58% | 0.95% | 0.21% |
| **User3** | 0.43% | 0% | 0.86% |
| **User5** | 0.05% | 0.09% | 0.01% |
| **User7** | 0.01% | 0.02% | 0% |

(Figure 3.3 – Predicting ownership of user2 testing dataset)

The results are much more favourable, with mouse prediction being as high as 94%, comparing this to the precision and recall of the models detailed above in table 1, we see that having a

higher precision and recall, resulted in the model predicting correctly a lot more of the time. Lastly comparing the tables to the precited ownership of user6 testing dataset, the user model who scored a low value of 0.02 on recall rate for desired mouse traits;

| | **Combined match** | **Mouse match** | **Keyboard match** |
|---|---|---|---|
| **User6** | 14.25% | 0% | 28.51% |
| **User7** | 7.57% | 14.71% | 0.44% |
| **User1** | 4.47% | 3.68% | 5.26% |
| **User2** | 1.1% | 2.21% | 0% |
| **User4** | 0.65% | 0% | 1.32% |
| **User3** | 0.43% | 0% | 0.88% |
| **User10** | 0.43% | 0% | 0.88% |
| **User5** | 0.22% | 0% | 0.44% |
| **User8** | 0% | 0% | 0% |
| **User9** | 0% | 0% | 0% |

(Figure 3.4 – Predicting ownership of user6 testing dataset)

Here we can see that while the software did manage to correctly predict user 6, the mouse prediction was 0% with only the relatively high keyboard percentage match of 29% resulting in its combined prediction, this was expected due to the mouse recall rate of 0.02 for user6, resulting in almost no correct predictions. It is not clear why user6 and user10 received such low recall rates for desired mouse traits, but it is of the authors' opinion that perhaps these participants were not consistent in their usage of the mouse, therefore the model could not distinguish the correct traits to a high enough prediction rate.

# 4. Discussion

With the results of the experiment done within this project, we see that it is indeed possible to verify the identity of a user, but this does come with several limitations. If the user is not consistent with their usage of the input methods, the machine learning models struggle to identify a pattern, and false rejections occur too frequently. The results do seem to indicate that they would be suitable for a secure system environment as the false positive rate is almost 0%, meaning if anything the models will not approve access that does not match the users' pattern, but rather the models will judge the actual user more harshly.

Using a binary classifier gave the best results, but building an individual model containing the features of the desired user, as well as all the other users' undesired features, for each user makes it an unlikely implementation for large operations. With more users, the size of each model will expand in size and each user creates 2 additional models for the mouse and keyboard. Implementations such as the one in this project would best work for smaller sample sizes, possibly academic class groups, or smaller workplaces.

Analysing the results of each metric of measurement, it is clear to see that the mouse data provides a higher accuracy compared to the keyboard data, keyboard machine learning models seem to produce a higher false rejection rate. It is the authors' opinion that this is most likely due to a larger feature extraction available for the mouse biometrics as it is a 2d plane, the keyboard biometrics are generated from a binary input means, resulting in less data for feature extraction.

Another point to make is that the users 1, 6, 7, 9 and 10 all used the same keyboard and mouse for the experiment. As shown in table 3.1, for the mouse portion of the machine learning models, the recall rate is on the lower side for these users. This could offer the possibility that perhaps the models are picking up in the differences of the input methods, rather than the users behind said input methods. While this is a possibility, it is the author’s opinion that the

individual user characteristics still play a part in the model as otherwise for the above 5 users, otherwise the chance of identifying them correctly would be almost non-existent.

Comparing the results to other articles of research done on the topic of mouse and keystroke biometrics, one can assume that perhaps due to the issues with the dataset harvested, the results are not as favourable as some research papers have managed to achieve. It is also possible that the datasets used by some of these papers have some sort of biases that result in the high level of results that they achieve, perhaps their dataset userbase is not that indicative of the general population.

# 5. Conclusion

This project aimed to identify whether or not it was possible to create software that would be able to identify certain user characteristics with mouse and keyboard biometrics, and use this information to predict a user and verify identity. From the aim alone, one can say the project can be deemed a success, as the software successfully manages to perform the task, with favourable results. While the project did not manage to produce a high enough match for the author to be satisfied with the results every time, the software does provide a distinct identity when predicting. It is the author's opinion that this software would not be suitable in its current form for users, as the possibility of rejection of the correct user is too high a risk.

## 5.1 Suggestions for further work

For further work, it would be wise to look into generating results for the undesired user features, that way a model can be built just for the user, with the undesired results being generated randomly. Perhaps another machine learning model to generate features that are unlike the desired user traits. Unfortunately, due to the time constraints of the project, the author was unable to experiment with more feature extraction for both input means, while they both produce semi-favourable results, there may lie more information for harvesting that can result in higher positive authentication rates. It is recommended that more data is gathered that uses users on a different keyboard and mouse from their original training data, so the possible correlation between using different input methods resulting in skewed results can be explored, as well as having all users complete the experiment on the same mouse and keyboard to distinguish between user characteristics and input method characteristics.

# References

Anon 2020. *balabit/Mouse-Dynamics-Challenge*. [online] Balabit. Available at: <https://github.com/balabit/Mouse-Dynamics-Challenge> [Accessed 15 Mar. 2020].

Gaines, R.S., Lisowski, W., Press, S.J. and Shapiro, N., 1980. Authentication by Keystroke Timing: Some Preliminary Results. [online] Available at: <https://www.rand.org/pubs/reports/R2526.html> [Accessed 15 Mar. 2020].

Harilal, A., Toffalini, F., Homoliak, I., Castellanos, J., Guarnizo, J. and Mondal, S., 2018. *The Wolf of SUTD (TWOS): A dataset of malicious insider threat behavior based on a gamified competition*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/324536760_The_Wolf_of_SUTD_TWOS_A_dataset_of_malicious_insider_threat_behavior_based_on_a_gamified_competition> [Accessed 3 May 2020].

Hu, T., Niu, W., Zhang, X., Liu, X., Lu, J. and Liu, Y., 2019. *An Insider Threat Detection Approach Based on Mouse Dynamics and Deep Learning*. [Research Article] Security and Communication Networks. Available at: <https://www.hindawi.com/journals/scn/2019/3898951/> [Accessed 15 Mar. 2020].

Jenkins, J., Nguyen, Q., Reynolds, J. and Szu, H., 2011. (2) (PDF) The physiology of keystroke dynamics. *ResearchGate*. [online] Available at: <https://www.researchgate.net/publication/253642686_The_physiology_of_keystroke_dynamics> [Accessed 15 Mar. 2020].

Khan, A., Quaraishi, S. and Bedi, S., 2019. Mouse Dynamics as Continuous User Authentication Tool. *International Journal of Recent Technology and Engineering*, 8(4), pp.10923–10927.

Lee, H., Hwang, J.Y., Kim, D.I., Lee, S., Lee, S.-H. and Shin, J.S., 2018. *Understanding Keystroke Dynamics for Smartphone Users Authentication and Keystroke Dynamics on Smartphones Built-In Motion Sensors*. [Research Article] Security and Communication Networks. Available at: <https://www.hindawi.com/journals/scn/2018/2567463/> [Accessed 15 Mar. 2020].

Mondal, S. and Bours, P., 2016. Combining keystroke and mouse dynamics for continuous user authentication and identification. In: *2016 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)*. 2016 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA). pp.1–8.

Morgan, S., 2017. Cybersecurity market grows 35X from $3.5B in 2004 to $120B in 2017. Spending predicted to exceed $1T next 5 years. Available at: <https://www.linkedin.com/pulse/cybersecurity-market-grows-35x-from-35b-2004-120b-2017-steve-morgan> [Accessed 3 May 2020].

Tan, Y.X.M., Iacovazzi, A., Homoliak, I., Elovici, Y. and Binder, A., 2019. Adversarial Attacks on Remote User Authentication Using Behavioural Mouse Dynamics. *2019 International Joint Conference on Neural Networks (IJCNN)*, pp.1–10.

Yıldırım, M. and Anarım, E., 2019. Novel Feature Extraction Methods for Authentication via Mouse Dynamics with Semi-Supervised Learning. In: *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2019 Innovations in Intelligent Systems and Applications Conference (ASYU). pp.1–6.

# Programming References

Python. *Welcome to Python.org*. [online] Python.org. Available at: <https://www.python.org/> [Accessed 4 April 2020].

Palmér, Moses. n.d. *Pynput: Monitor and Control User Input Devices*. Available at: <https://pypi.org/project/pynput/> [Accessed 4 April 2020].

NumPy. *NumPy — NumPy*. [online] Available at: <https://numpy.org/> [Accessed 4 April 2020].

Pandas. *pandas - Python Data Analysis Library*. [online] Available at: <https://pandas.pydata.org/> [Accessed 4 April 2020].

SciKit-Learn. *scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/> [Accessed 4 April 2020].

HiLite.me. *Source code beautifier / syntax highlighter – convert code snippets to HTML « hilite.me*. [online] Available at: <http://hilite.me/> [Accessed 4 April 2020].

# Appendix

Included attached to this report are the;

**Interim report**

The initial outline of the project submitted at the start of the academic year

**Presentation poster**

A scaled-down version of the poster presented to colleagues

**Software Instructions**

Instructions on how to use the software produced in this report

**Software source code**

Python source code of the software produced in this report