**Utilizing Machine Learning Techniques for Continuous Authentication**

**Introduction:**

For our CISC 5001 project we researched continuous authentication and how machine learning techniques like principal component analysis (PCA) may help improve the performance of existing continuous authentication systems like HMOG and Touchalytics.

Continuous authentication is the idea of having a device such as a smart phone monitor the user's strokes across the screen as well as movements and grasping of the phone to try and tell if the holder is the authorized user or an attacker. Traditional login authentications only check for any problems at the time of password entry and so are susceptible to having the device accessed after the proper user has entered the code or the attacker getting the code another way. Continuous authentication though works as a second layer of security as it gives the device the ability to detect unwanted use after the initial password login and as long as the device is in use.

**Background:**

Two recent influential papers by Sitova et al. and Frank et al. examined continuous authentication and created new custom datasets: the Touchalytics and the HMOG datasets. The papers proposed custom algorithms to achieve user authentication. The third paper by Gong et al. built on the work of the Touchalytics database in hopes of protecting the system from possible attacks. Specifically, they wanted to defend the system of a case where the attacker has access to the user’s touch data. I shall now summarize the papers on Hand Movement Orientation and Grasp (HMOG), Touchalytics and Forgery Resistant Touch Based Authentication.

The study on the HMOG system consisted of 100 subjects and made use of the phone’s accelerometer, gyroscope and magnetometer to monitor the holding of the phone as well as the tapping of the screen. The study also incorporated biometric key generation as well as tap and keystroke dynamic features. This data was collected by having each subject have 8 sessions where they were asked to type out various responses. 4 of these sessions had the subject sitting while the remaining 4 had them walking. Surprisingly the data showed that the equal error rate (EER) obtained by the authentication system was lower when the subject was walking. Through further examination the study concluded that this was due to the phone being able to collect more information about the user's movements due to the act of walking. For walking the lowest EER obtained was 7.16% and for sitting was 10.05%. This study in particular did make use of PCA where they discovered that the magnetometer made no significant contributions to the authentication system. Also, uniquely this study decided to investigate the extra power consumption continuous authentication would have on the device and so experimented with different sampling rates in order to see which provided the most efficient use of energy for accuracy. The study concluded that 16Hz which gave an energy overhead of 7.9% was optimal as higher sampling rates did not give substantial increases in accuracy but did increase power consumption and lower sampling rates increased the error substantially.

The Touchalytics system was developed under the idea each user interacts with their phone through swipes in a unique way similar to a signature or hand writing and so used different stages to collect this movement data. The system was first prepped with an enrollment phase where it tracked horizontal and vertical swipes separately. The system then compared interactions to the data it had collected. This study consisted of 41 participants and consisted of two sections. The first had participants tasked with reading an article and then answering some simple questions on it while the system recorded their vertical swipes. The second part consisted of a find the difference games where the 2 pictures were separated by a black image so they had to swipe back and forth recording their horizontal swipes. These 2 types of motions were recorded as one usually swipes vertically when reading an article and one usually swipes horizontally when going through a picture gallery. This was how the user data on swiping movements was recorded. The EER for this study was between a median of 0% and 4% depending on the type of session. The 0% median EER reflects the results of intra-session authentication where the system tries to authenticate the same user across multiple sessions in the same day and the 4% reflects an authentication test 1 week after the enrollment stage.

This last paper on Forgery Resistant Touch Based Authentication builds on the Touchalytics dataset and possible flaws in the original design. One member of this study was Mario Frank who also worked on the Touchalytics database. This research was conducted in order to find a way of protecting a device from an attacker who could gain access to the user’s touch data and have a robot replicate the movements as the paper says has been demonstrated as possible. To combat this the group came up with the idea of changing the screen settings randomly each use by doing things like shifting the x-axis a little. The user will naturally adjust their actions to overcome this change but a robot would not be able to. The study theorizes that even if the attacker had access to all of the user’s touch data on all conditions it would still be impossible for them to know which setting the system was on at the time of the attack and which data set to use. The study consisted of 25 subjects where participants played 2 games in order to collect touch data. The first was the same one as the Touchalytics one where the participant had to spot differences between pictures. The second had the participant match images by scrolling up and down. The conclusion of the study found that this way of authentication decreased the EER from 17-18% to 2-9% and was better able to fend off random and targeted attacks.

By looking at these various studies we can see that continuous authentication can serve as a great backup security measure to more traditional forms of authentication like pins or passwords. This is a type of technology that can greatly increase information security which is growing ever important in this day and age. As can be seen in the HMOG study it can also be seen that machine learning can help this technology perform better than originally thought and we would like to implement more machine learning techniques on this type of authentication in hopes of further making the technology more accurate and reliable.

**Project Tools:**

For the project we made use of the Python programming language. Google Collaborate was used as the IDE for developing the code. We used Tensorflow and the Matlab libraries for implementing machine learning algorithms to be used by the continuous authentication system. Also, we used the KNN or K Nearest Neighbors algorithm to interpret the data as well as PCA to see what components provided the most variance.

**Process:**

We decided to conduct our research on the Touchalytics dataset. For our first trial we repurposed code from Eddie Lam, another Brooklyn College student, who did research on a similar project. We also reached out to Frank via email for clarification on the study he conducted. He mentioned that it would be useful to try one-vs-all on the dataset. This means marking 1 user as authentic and setting the rest as attackers. In addition to this we tried one versus one and all versus all. The reason these multiple datasets were created were to check the differences in accuracy obtained depending on the data provided.

Using the KNN algorithm with all users having unique IDs a 67.0% mean accuracy was obtained. The first thing we experimented on was the K value for KNN. By convention K is at first set to 5 and then changed to see what the best value is. After creating a graph of k value versus mean error it seemed that 5 was the best value. We then added confusion matrices for the data to make the results easier to read.

We created subsets of the Touchalytics data set. One had a one versus many criteria. There was one authentic user and the other 40 users were classified as strangers. Another selected 2 random users. The first being the authentic user and the second being the stranger. For one versus many the accuracy drastically decreased. The much larger set of stranger data caused the program to be biased towards strangers. When doing one versus one the accuracy became much higher though. Each time the data was run there was small fluctuations in values but those were mostly irrelevant.

We also tried employing PCA. The data has 31 different components to choose from. We first selected an initial PCA component value of 2. We then fed this new data back into the KNN algorithm. Our results were less than the initial accuracy. We then tried increasing the PCA component value. Increasing it to 10 increased the accuracy attained. This let us know that only taking 2 components into account is not optimal. It would be valuable to try all possible PCA values to see which provides us with the best accuracy.

The accuracy obtained for recognizing the authentic user in the one versus many data set was 61.3%. For one versus one we obtained a 96.0% accuracy. These results made us consider focusing more on one versus one datasets.

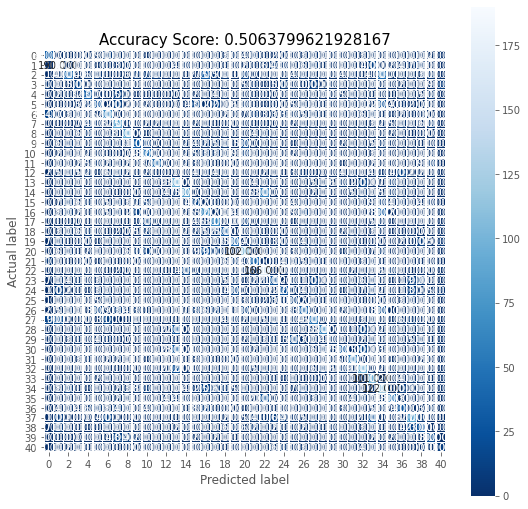
We then applied PCA onto the one versus one data set. For a PCA value of 2 we obtained an 86.0% accuracy rate. This was less that the accuracy with no PCA however. We then tried increasing the number of PCA components to see if that would increase the accuracy. Increasing the number of components to 10 increased the accuracy on the same data set to 90.0%. While still less than the initial accuracy that considers all the components this was a noticeable improvement.

One limitation noticed during this project is the bias of the data. If there is more data classified as a stranger than data that is classified as authentic the algorithm will more likely assume the authentic user is a stranger. While the authentic user will in most cases be accessing the device more than strangers, we must be careful of the reverse problem. If there is too much data labeled as authentic then strangers would be wrongly assumed to be authentic users. This means that the dataset needs to be properly balanced to avoid bias in either direction. This shortcoming can be avoided by manipulating the dataset. The issue with this though is that while this can be controlled in a lab setting this cannot be done in the real world. We can provide the program with an equal number of entries for the user and attacker but in real life there will probably be much more records of the user. This issue can also be compounded by multiple attackers existing in a real world setting while there is still only one authentic user. These are areas worth considering when looking into the practicality of continuous authentication.

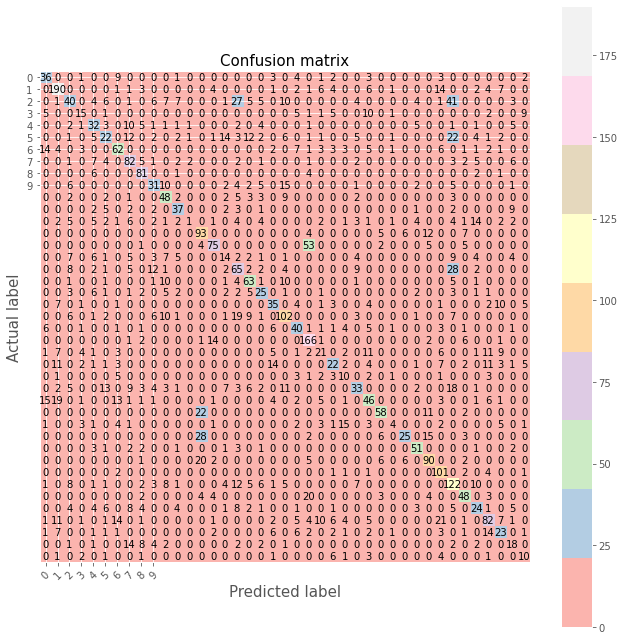
**Project Results:**



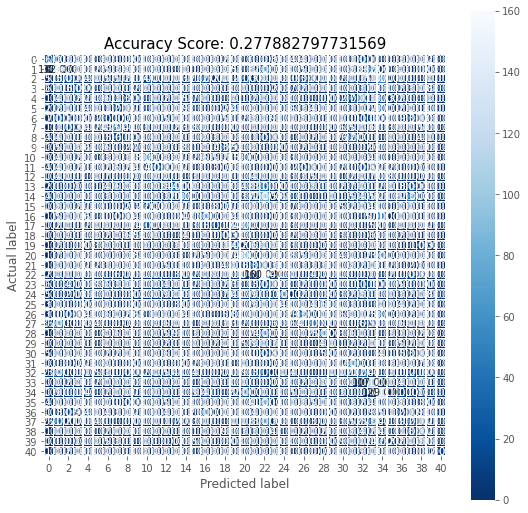
This was the graph produced by plotting error against the K value. It shows that a K value of 5 seems optimal.



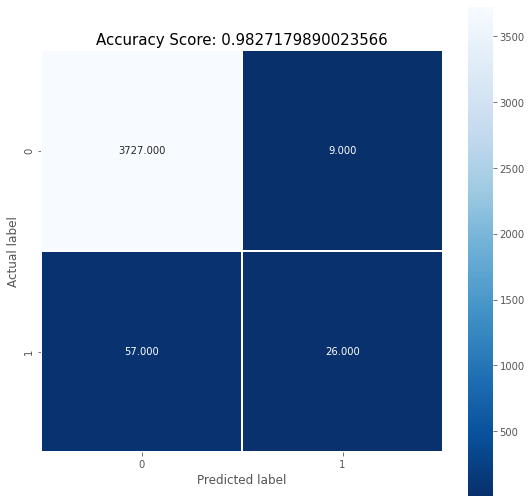
This was the accuracy and confusion matrix obtained when all users were distinct.



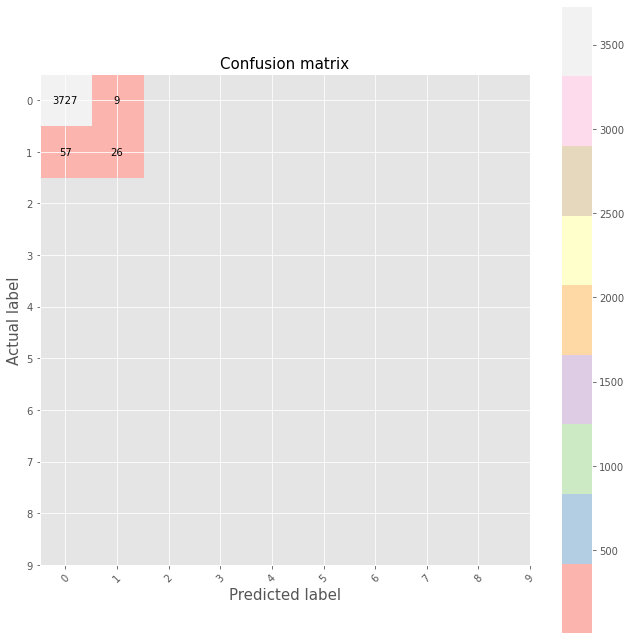
This was the confusion matrix obtained when all users were unique.



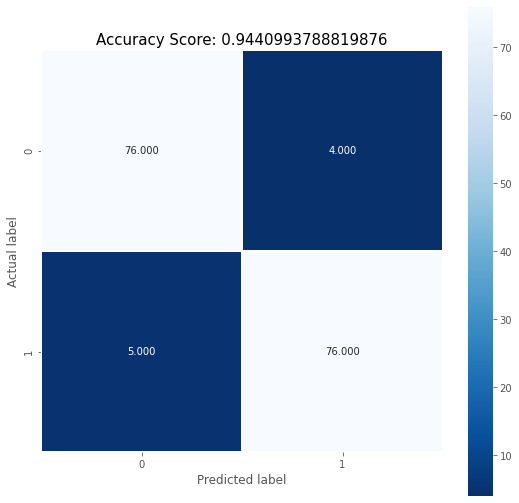
This is the accuracy for all users unique when it was given a PCA of 10 and then fed back into KNN. We can see that it is much smaller than not using PCA.



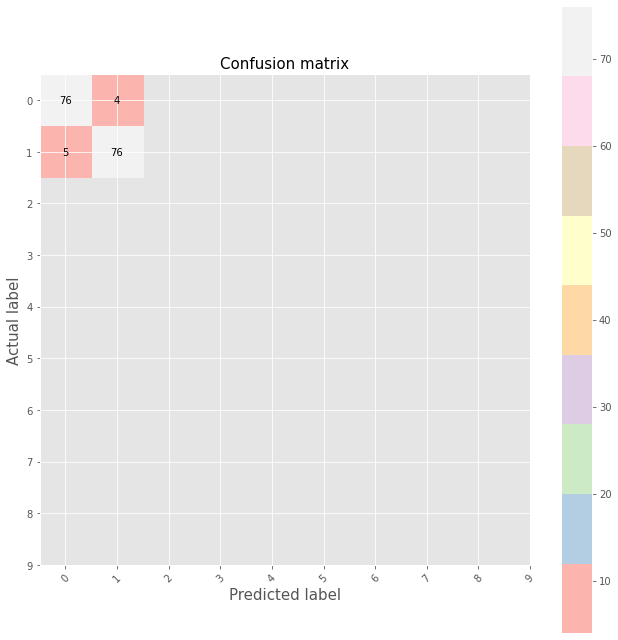
This was the accuracy obtained for one versus all with KNN. The high accuracy is deceptive though because of the bigger amount of stranger data. The accuracy of correctly identifying the authentic user is very low as we can see it only predicted the correct user 26 times out of 92.



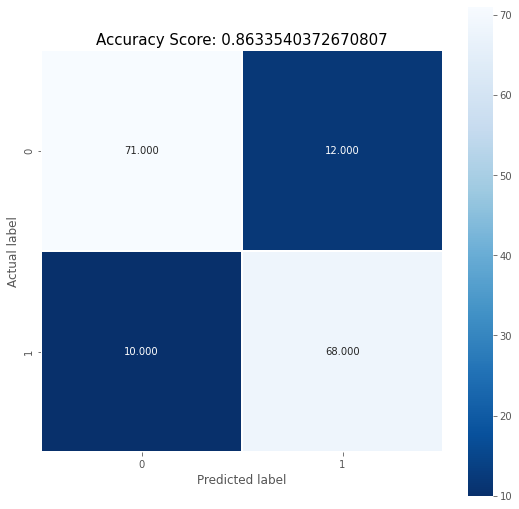
This is the confusion matrix for one versus all.



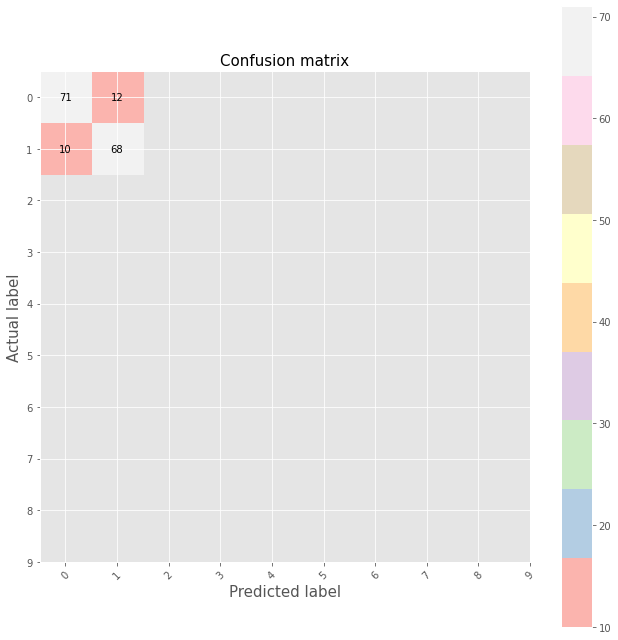
This is the accuracy for one versus one with the same number of entries. As can be seen the accuracy is incredibly high with very few mistakes.



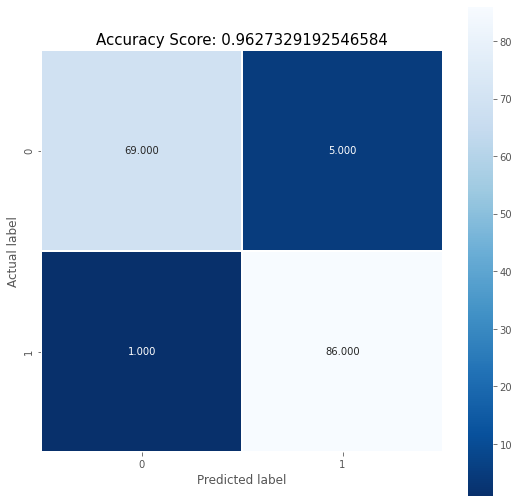
This is the confusion matrix for one versus one



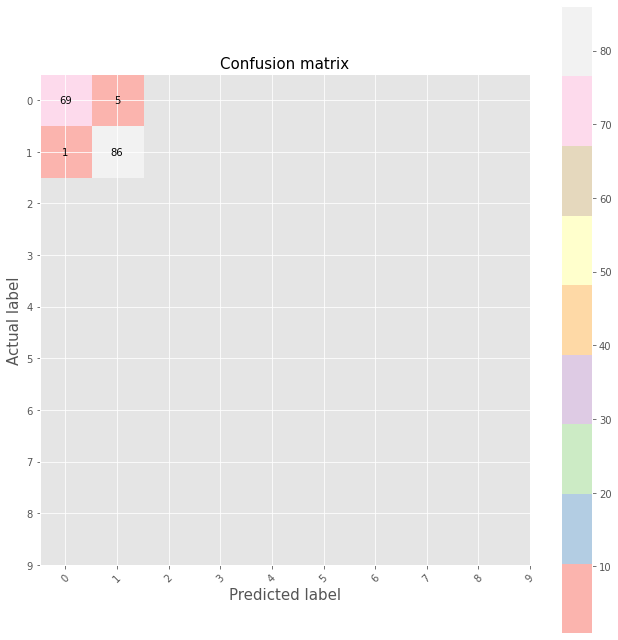
This is the accuracy for one versus one with a PCA value of 2 that is fed back into KNN. As can be seen the accuracy is significantly less that not using PCA.



This is the accuracy for one versus one with a PCA value of 2 that is fed back into KNN.



This is the accuracy for one versus one with a PCA value of 10 that is fed back into KNN. The accuracy obtained is greater than that obtained with a PCA value of 2 but still less than no PCA.



This is the accuracy for one versus one with a PCA value of 10 that is fed back into KNN.

**Data Sets**

Version0 is the unaltered dataset.

Version1 has 1 user as the authentic user with ID 1 and the rest are classified as ID 0.

Version2 is the same as version 0 but has sessions 6 and 7 removed.

Version3 is the same as version 0 but has one authentic user with ID 1 and the rest are classified as ID 0. Version3 also has sessions 6 and 7 removed.

Version4 is the same as version 0 but has columns 2 and 13 removed which are DocID and PhoneID.

Version5 has 1 user as the authentic user with ID 1 and the rest are classified as ID 0. Also, Version5 also has columns 2 and 13 removed which are DocID and PhoneID.

Version6 is the same as version 0 but has sessions 6 and 7 removed. Version6 also has has columns 2 and 13 removed which are DocID and PhoneID.

Version7 is the same as version 0 but has one authentic user with ID 1 and the rest are classified as ID 0. Version3 also has sessions 6 and 7 removed as well as columns 2 and 13 removed which are DocID and PhoneID.

Version8 is the same as Version0 but it only has records of User 1 and User 2. Also, it contains the same number of entries for both users.

**Code Versions**

Each code version builds on the one before it.

What does make each one worthy of examination is that some code versions will be used on different datasets or have different values and so they show how the accuracy changes with different datasets.

Code version1

This is the first working instance of the code that I created. It examines 1 vs all using dataset 7. Because of the great number of entries registered as attackers the data is biased towards assuming the authentic user is an attacker. This means that there is a very high chance that the authentic user will be rejected. Version 1 also plots 2 graphs of a PCA of value 2 which shows variance between the two different classifications by how close they are to other dots.

Code version2

This is identical to version 1 except it has a graph that plots error against the value of K for the KNN algorithm. This was done as the optimal value of K cannot be found except with trial and error. It is by convention 5 at the start though. The program takes a long time to process this section though so it is usually commented out in future versions of the code.

Code version 3

This code makes use of dataset 8. This means it compares 2 different users with the same number of records. Dataset 8 was created manually but It is possible to adjust the program to make it do this by itself. The 1 vs 1 dataset shows a much higher increase in accuracy than 1 vs many. Also, with this code I decided to evaluate the PCA of value 2 by feeding it back into the KNN. The accuracy obtained after is less that the initial KNN which takes into account all components.

Code version 4

This code is the same as version 3 but it increases the PCA value to 10. This was to see if increasing the PCA value would increase the accuracy value when fed back into the KNN. This worked as the matrices and accuracy score was notably increased. This showed that the value of 2 for PCA was not optimal. The value obtained was still less than the accuracy obtained when only using KNN. The number 10 was used as a guess. Ideally though the code should run through all possible PCA values to see which produces the best accuracy. There could even be a PCA value that when fed into the KNN produces a better accuracy than not using PCA at all.

Code version 5

This is similar to version 4 but it uses the original dataset. This was done to see what the accuracy of the original dataset with PCA fed into KNN would look like. This dataset was also being used because after the high accuracy of one vs one we did on users 1 and 2 we wanted to see if we would obtain similar results across all possible pairs.

Code version 7 and 8

These versions were modifications of 5 in order to try comparing all possible pairs. We were unfortunately not currently able to finish it due to time constraints.

Code version 10

This was the final code produced which contains some extra documentation. As mentioned, it builds on all previous versions but can be considered the definitive version. Running this code with different datasets that we created will show the accuracy of the different user and attacker scenarios.

**Link to GitHub with Code and Data Sets**

<https://github.com/HA-work/CISC5001>

**Future areas of research**

The two main areas that I think are worthy of investigation are how the PCA value affects accuracy and the accuracy between all possible pairs. By creating a code that tests all possible PCA values we can see if there is a PCA value that when fed into KNN outperforms only using KNN. This could be very useful as it would mean that there are unneeded components being recorded and examined. This means that removing this could improve accuracy and improve the battery life of the device making use of the implicit authentication as there are less metrics to examine.

The next thing of note is to try and compare all possible pairs to see the average accuracy. This is because while one versus one achieved incredible results it could be possible that the 2 users who were randomly selected just happened to be very different from each other. Examining this would help improve the understanding of distinguishing between users.

**References:**

Sitova, Z., Sedenka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., & Balagani, K. S. (n.d.). HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users. Retrieved from <https://arxiv.org/pdf/1501.01199.pdf>

Frank, M., Biedert, R., Ma, E., Martinovic, I., & Song, D. (n.d.). Touchalytics: On the Applicability of Touchscreen Inputas a Behavioral Biometric for Continuous Authentication. Retrieved from <http://www.mariofrank.net/paper/touchalytics.pdf>

Gong, N. Z., Payer, M., Moazzezi, R., & Frank, M. (n.d.). Forgery-Resistant Touch-based Authentication on MobileDevices. Retrieved from <http://www.mariofrank.net/paper/2016_AsiaCCS_ForgeryResistantTouchAuth.pdf>

**Links to Webpages where we drew code from**

[K-Nearest Neighbors Algorithm in Python and Scikit-Learn](https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/)

[PCA using Python (scikit-learn). My last tutorial went over Logistic… | by Michael Galarnyk | Towards Data Science](https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60)

[PCA, Nearest-Neighbors Classification and Clustering](https://www.numerical-tours.com/matlab/ml_1_pca_nn/)

[classification - Using the eigenvalues from PCA in k-nearest-neighbours - Cross Validated](https://stats.stackexchange.com/questions/70134/using-the-eigenvalues-from-pca-in-k-nearest-neighbours)

[pca - Principal component analysis before nearest neighbor search - Cross Validated](https://stats.stackexchange.com/questions/18608/principal-component-analysis-before-nearest-neighbor-search)

[Multi-mode operation of principal component analysis with k-nearest neighbor algorithm to monitor compressors for liquefied natural gas mixed refrigerant processes - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0098135417302466)

[K-nearest Neighbours Classification in python – Ben Alex Keen](https://benalexkeen.com/k-nearest-neighbours-classification-in-python/)

[Indexing, Slicing and Subsetting DataFrames in Python – Data Analysis and Visualization in Python for Ecologists](https://datacarpentry.org/python-ecology-lesson/03-index-slice-subset/index.html)

[KNN in Python. You will learn about a very simple yet… | by Czako Zoltan | Towards Data Science](https://towardsdatascience.com/knn-in-python-835643e2fb53)

[8 Python Pandas Value\_counts() tricks that make your work more efficient](https://re-thought.com/pandas-value_counts/)