Preprocessing:

The first step in this lab is to preprocess the data given. We create a data object that records: ID, swipe, X, Y, Pressure, Area, Time, Velocity, Acc and features. We go through the files and gather the data by swipe number for each user.

Feature Extraction:

Over a complete touch stroke, a vector corresponding to x-coordinate, y-coordinate, area, pressure, and time was populated. We computed 2 more vectors for velocity and acceleration. We computed the following features from the vectors: from each of the velocity, acceleration, area, and pressure vectors, we computed 5 features: mean, std, 1st, 2nd, and 3rd quartiles. From the coordinates vectors and the time vector we computed average change of x, average change of y, swipe width and height, max and min slope, and total duration. We also computed the coordinates of the start and end points; number of data points; area and pressure in mid-stroke; attack angle, which is the angle by which the user starts the swipe; and leaving angle, the angle by which the user ends the swipe. Here is a table with the 36 features we chose.

|  |  |
| --- | --- |
| **1** | *x-coordinate of the start-point* |
| **2** | *y-coordinate of the start-point* |
| **3** | *x-coordinate of the endpoint* |
| **4** | *y-coordinate of the endpoint* |
| **5** | *Average Change of X position* |
| **6** | *Average Change of Y position* |
| **7** | *Swipe Width* |
| **8** | *Swipe Height* |
| **9** | *Max Slope* |
| **10** | *Min Slope* |
| **11** | *Total Duration* |
| **12** | *Velocity: mean* |
| **13** | *Velocity: std* |
| **14** | *Velocity: Q1* |
| **15** | *Velocity: Q2* |
| **16** | *Velocity: Q3* |
| **17** | *Acc: mean* |
| **18** | *Acc: std* |
| **19** | *Acc: Q1* |
| **20** | *Acc: Q2* |
| **21** | *Acc: Q3* |
| **22** | *Pressure: mean* |
| **23** | *Pressure: std* |
| **24** | *Pressure: Q1* |
| **25** | *Pressure: Q2* |
| **26** | *Pressure: Q3* |
| **27** | *Area: mean* |
| **28** | *Area: std* |
| **29** | *Area: Q1* |
| **30** | *Area: Q2* |
| **31** | *Area: Q3* |
| **32** | *Number of Data Points* |
| **33** | *Attack Angle* |
| **34** | *Leaving Angle* |
| **35** | *Mid-stroke area* |
| **36** | *Mid-stroke pressure* |

Feature Selection:

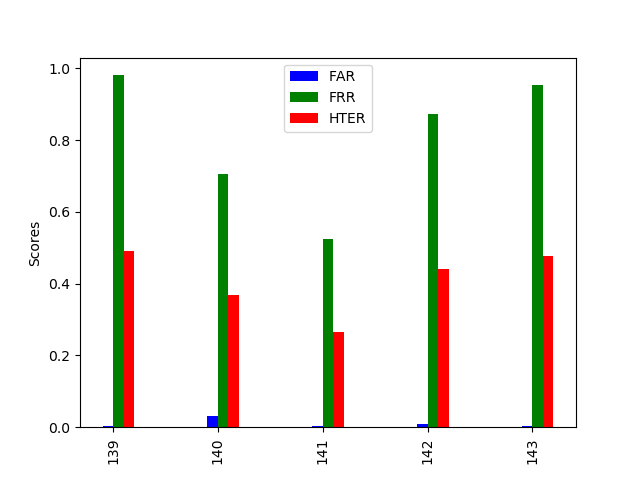
After extracting features, we first transform our data to be compatible with machine learning inputs by putting the features in an X matrix and labels in a y array. Each user has genuine and impostor data with genuine being the data of the user and impostor being the data of all the users that are not the current user. Then, to select the best features, we use SelectKbest in the sklearn package. We specifically use mutual information classification which takes into consideration how dependent the features are on each other. This is a good way to find the best features for the type of data we have because correlated features can distinguish users better.

Machine Learning Algorithms:

We use four machine learning algorithms to train and test our model: KNN, Logistic Regression, SVM and Random Forest. For each algorithm, we use GridSearchCV sklearn function to find the best parameters and refit the trained model. Although this approach makes the code much slower, it is essential to get better results as different parameters work for different types of data. We then predict the labels for genuine and impostor test data and record confusion matrix for each user.

We focus more on KNN when reporting results as it is the best algorithms for the type of data we have. Two class SVM also works well, but since our classes are imbalanced it is not the best approach because the model will be biases towards the impostors.

Data Analysis:

We chose to use a bar graph to analyze performance. For each user we plot a set of 3 bars representing FARs, FRRs, and HTERs. We plot all the users in the same graph, each with its own set. The users are represented by their number on the x-axis and the y-axis represents scores, the values computed for FAR, FRR, and HTER, for each user. Since the data set is very large, we ran the code only in one mode, namely landscape, and only on 5 users using KNN algorithm. A snapshot of the bar graph for the 5 users is shown below. 

Overall, KNN was able to predict labels of impostors and genuine users better with an HTER value between 0.3 and 0.5. The model trained is very conservative as it has a higher false reject rate than a false accept rate. This is a very important in the field of biometrics because users would rather have a system than doesn’t always recognize them than a system that gives impostors access.