CMPT 353: Computational Data Science

Gesture Based Shape Detection

Balrick Gill - 301309400
Quoc tuong Tran - 301297287
Goeun Jang -301341130
CMPT 353 Fall 2021

1. Introduction

1.1 Objective

Most of us have all had one of those days where our hands were too full to dial someone, preoccupied in another task to focus on a tiny screen or wanting a more intuitive method to navigate the labyrinth of functionalities/apps found on smartphones. This project aims to eliminate these instances by allowing the user to physically move the phone in different shapes instead of maneuvering through the phone screen. The following are some examples of possible functions that could be linked to each shape:

- Accepting a phone call
- Directly dialing a number
- Starting a video
- Other programmable feature like start music or increase volume

Since most people are familiar with gesture based controls such as double tap off screen to turn phone on or slide finger from bottom of phone to middle of screen to return to the home screen. This takes a similar approach of physically implementing a gesture for increased navigation capabilities through drawing the shapes O, S, V in the air with a smartphone.

We are particularly interested in accurately detecting the shapes drawn by the user. Hence, the linked functions are beyond the scope of this project.

2. Data Collection

2.1 Tools

To collect the necessary data, we utilized the Physics Toolbox Sensor Suite app on supported Android devices. For this project we combined two sensors to get the necessary data with Linear Acceleration and g-Force. We decided to measure linear acceleration since it enables us to record when the user changes the direction of the phone and the speed at which the device is moving at. We chose to collect g-force measurements because it was similar to what we were measuring with linear acceleration and found through initial trials it gave a visually cleaner image through graphing the collected data of what was happening.

2.2 Collecting Samples

The user has to physically move their phone, when recording, in the air to draw the predetermined shapes (O, S, V) which our project aims to classify. The users set their device to dual sensor mode set to record g-Force and Linear Acceleration. They only record data when drawing to minimize recording unnecessary data. We also collect data on hand used, dominant hand, and age.

Each participant executed the following steps using their dominant hand and non-dominant hand:

1. Hold phone flat (like how it would be on desk)
2. Each shape should be drawn within a 15 x 10 canvas (L x W inches)
3. Move phone around at constant speed (total 2 – 4 seconds drawing each shape) while holding phone flat with no wrist movement (See figure 2.2.1 for shapes)
    a. When drawing V, start at top left
    b. When drawing O, start at top and move clockwise to starting point
    c. When drawing S, start at top right

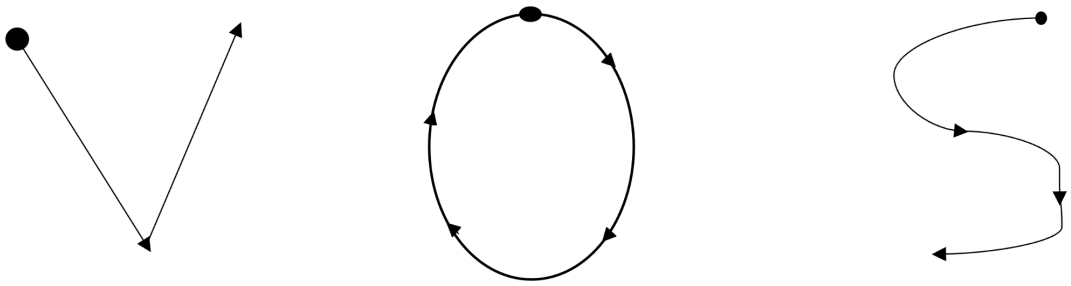


Figure 2.2.1: Visual representation of how to draw shape with black dot as starting point

2.3 Notes on Sampling

It is important to address possible sources of inaccuracy of data points from how each user draws each shape to technical aspects. All sampling was done in person to confirm correct shapes were being drawn in the correct steps. Drawings that did not follow the guidelines were redone to ensure consistency for later steps of the project. Drawer style was preserved such as non-circular curves and shakiness in straight lines, similar to what is seen in variation of hand printed letters.

Noise can enter the recording through just pressing record button, through testing pressing the record button can measure a noise of around 0.25 g in g-Force and 1 m/s^2 in acceleration when quickly trying to press the record button. We decided not to trim the start and end of a recording because the signal is followed directly after the noise which could last for a really small time interval making it difficult to pinpoint. We decided to not use the gyroscope sensor due to how small movements in the wrist can give very different data points for the same shape.

Of course the phone sensors can also vary depending on factors such as phone and outside temperature, battery conditions, phone usage/condition, and quality of sensors from the manufacturer. We used two phones to collect the data from users: a Samsung S20+ with Linear Accelerometer collection rate of 204 Hz and 410 Hz for g-Force, Oneplus 6 with Linear Accelerometer collection rate of 202 Hz and 407 Hz for g-Force.

3. Data Transformations

3.1 Use of Standard Deviations

We recorded the continuous numerical data to a single standard deviation for each sensor and for each axis. We decided to go with standard deviation for a parameter due to the visible clustering. Figure 3.1.1 shows clustering of shapes when plotting their x and z axis linear acceleration standard deviations in a scatter plot and Figure 3.1.2 is an enhanced graph of the chunk at the bottom left of figure 3.1.1.

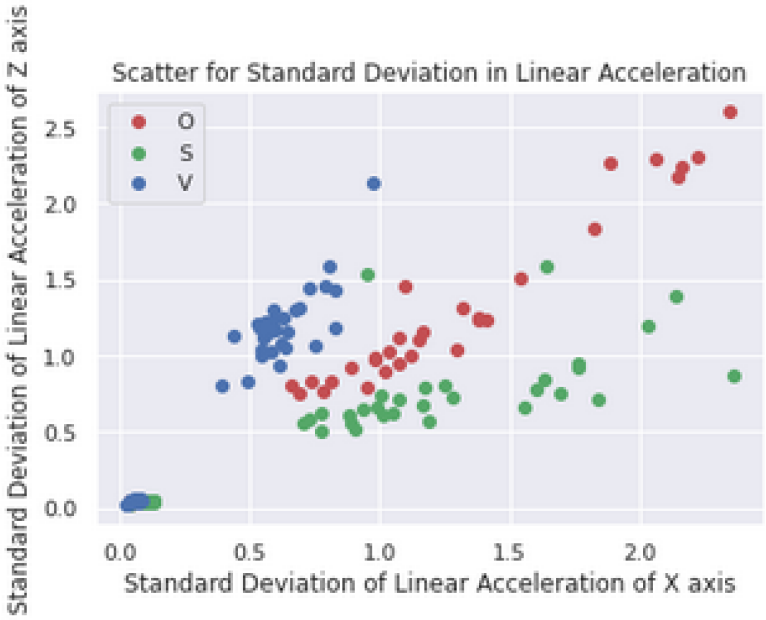


Figure 3.1.1: Scatter plot illustrating cluster of shapes

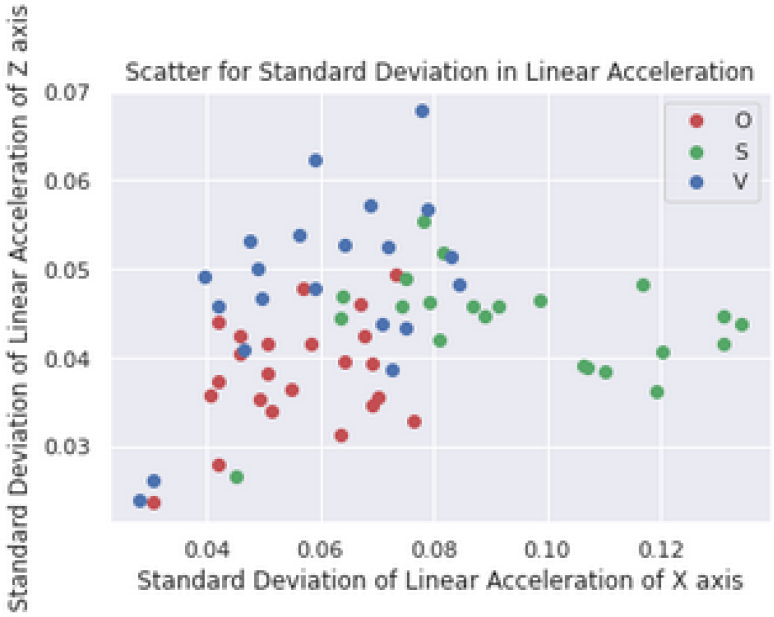


Figure 3.1.2: Enhanced view of cluster in bottom left of figure 3.1.1

3.2 Peaks and Valleys Parameter

Going through our collected data by graphing samples a visible pattern appeared for each of the three shapes being drawn. By looking at the peaks and valleys formed in each drawing in each recorded axis (x, y, z) by each sensor there was a clear pattern of the number of valleys and peaks formed with noise before and after the drawing. There was also a recorded plateau whenever the shape would suddenly change direction like in the cusp of V. The peaks and valleys varied between each shape sample depending on how much force the user applied to each section of the drawing.

This can be illustrated by stepping through the process of counting the peaks and valleys for a drawing of V. Figure 3.2.1 is graphing the raw data from user 3 using their right hand while drawing V using g-Force data on the y-axis. Note graph is zoomed in to give a more detailed image and would appear more smoother on the recording app where the effect is more pronounced. The redline helps to see the V and distinguish it from noise. The green triangles estimate where the peaks and valleys are located in the drawing.

Figure 3.2.1: Plotting raw data from drawing of V with red lines highlighting signal and green triangles for peaks and valleys (drawn by hand).

We decided to use the Pareto Principle (1.28 standard deviations in both directions from mean) to find instances of the user shifting the phone in another direction and not a minor twist in the wrist or other minor movements (see Figure 3.2.2).

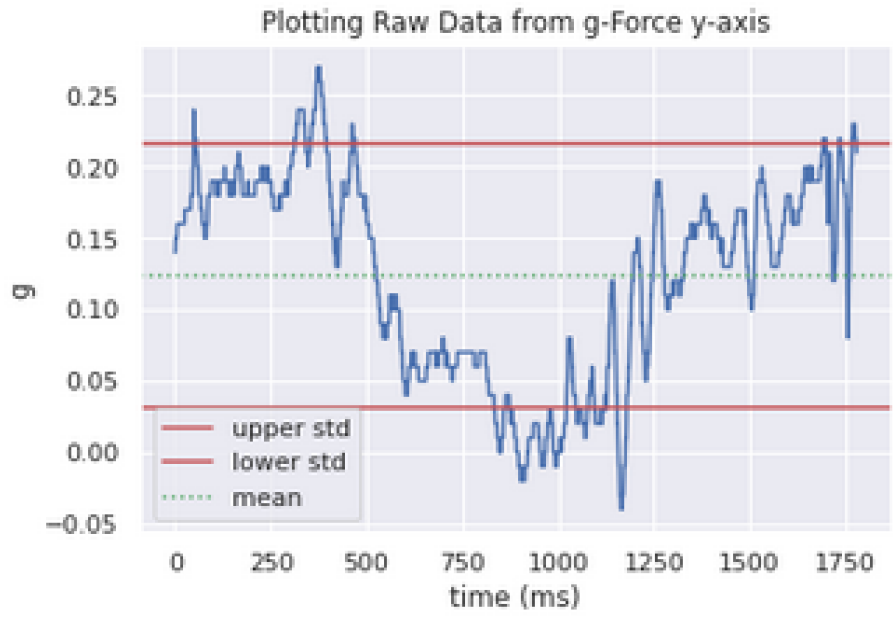


Figure 3.2.2: area between the red lines is 2.6 standard deviations

The continuous blocks outside of the 2.56 standard deviations count as either a peak or valley, if it is above the top redline it is a peak if it is below the bottom redline it is a valley. Figure 3.2.3 highlights this using green dots to highlight peaks and valleys, from the figure the transformation will indicate this drawing will have 7 peaks and 4 valleys.

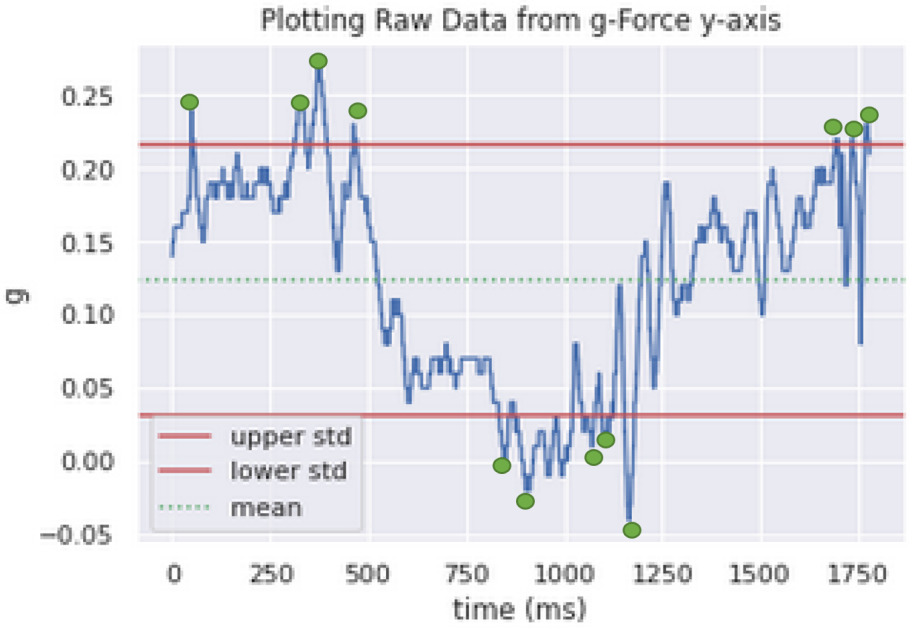


Figure 3.2.3: Green dots represent either a peak or valley (dots drawn by hand).

Similar transformation takes place on the other axis and sensor and the other shapes. Note used V to illustrate this process because it is the simplest to visually decipher. When going over this process with O and S, the shape is very sinusoidal.

Going over Figure 3.2.3 there should ideally be 1 valley with 2 peaks(1 on either side of the valley). We experimented with different standard deviations and were able to increase our max consistency accuracy (at around 1.4 standard deviations at either side of mean) to have the Random Forest model (with scaling) predicting consistently at 96 percent accuracy with a max observed accuracy of 98 percent (inconsistent). We decided to leave the standard deviation at 1.28 (for either side of mean) for reporting since we felt we were overfitting to our data which would not let the models perform well for new samples.

3.3 Scaling Data

Before running the data through all the models we have to make sure to convert all the columns to numbers to let the models read them. This is due to columns such as hand_used, shape or dominant_hand being assigned as either ‘R’ or ‘V’ during the cleaning process. We convert them into integers: 0 or 1 in case of hand related, or 0 to 2 if it is shape related. We found that if these columns are excluded, due to not being converted to numbers, the accuracy of the models dropped up to 10 percent. However, if only 1 or 2 columns are converted, the accuracy barely changes depending on classifier type, it can increase some bit for Neural Network while nothing for Bayes.

We decide to run the data through models with and without scaling to see the difference. With scaling, we have a selection between PolynomialFeature, MinMaxScaler, StandardScaler or FunctionTransformer. We picked out these ones due to their simplicity and generally widely used for prediction models. After running the script a few times for each of the transformation types. The accuracy between each transformation type varies a lot. MinMaxScaler was able to produce the highest accuracy (peak up to 0.96), but it was very inconsistent during our test runs. On the other hand, PolynomialTransformer lowers down the accuracy significantly, especially Neural Network where it was only 0.33. This led us to deciding to go with StandardScaler due to its consistency compared to others.

These observations that we have depend on the amount of samples that we were able to collect, which is only 4. This means that with a much larger amount of samples, we can have a more objective view on the results, and how scaling greatly affects the models.

4. Results

4.1 Machine Learning Analysis Before Scaling

We will first take a look at the different machine learning techniques used to determine which model resulted in optimum scores. Figures 4.1.1- 4.1.4 and 4.2.1- 4.2.4 are based on figure 4.1.5.

4.1.1 Naïve Bayes Classifier

Parameters were not given when using this classifier. We used the automated GaussianNB() function, which yielded a 0.795 accuracy score. Our primary interest was in the next three classifiers; hence we did not tune the function for optimal results.

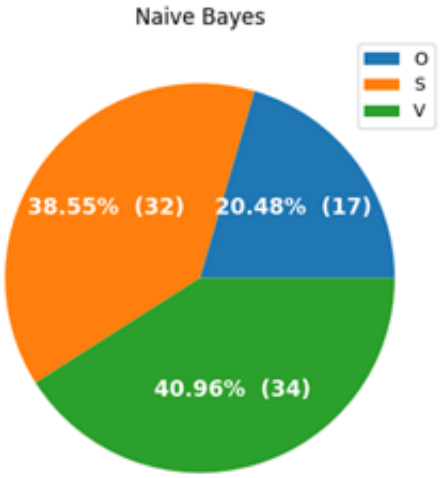
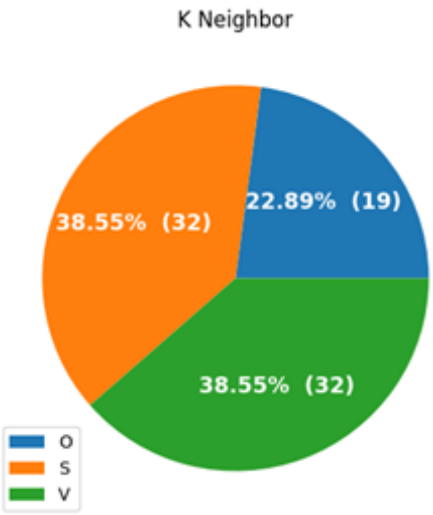


Figure 4.1.1: Naive Bayes Prediction Results before Scaling

4.1.2 K-Neighbors Classifier



Without scaling, with n=7, this classifier resulted in an accuracy score of 0.783. As the value of n got smaller than seven, the accuracy score started to decrease. As the value of n got larger than seven, the accuracy score started to increase. The highest score of 0.795 resulted when n=10. When the value of n went beyond 10, once again, the score started to decrease. With scaling, the k-neighbors classifier actually worked better when n=7 and got worse when n=10. Since we were interested in the accuracy score of models with scaling, we left the parameter at n=7.

Figure 4.1.2: KNeighbor Prediction Results before scaling

4.1.3 Neural Network Classifier

Before the scaling, this classifier had an accuracy score of 0.84. With hidden units greater than six, the classifier performed significantly worse, at a score of 0.675. With smaller alpha values, such as 1e-4 or 1e-5, the accuracy increased. The highest score was at 0.94. However, when tested after scaling, we found that this model actually performed better when the alpha value was 1e-3. Thus, we settled on 1e-3 for the alpha value and one hidden layer with six hidden units.

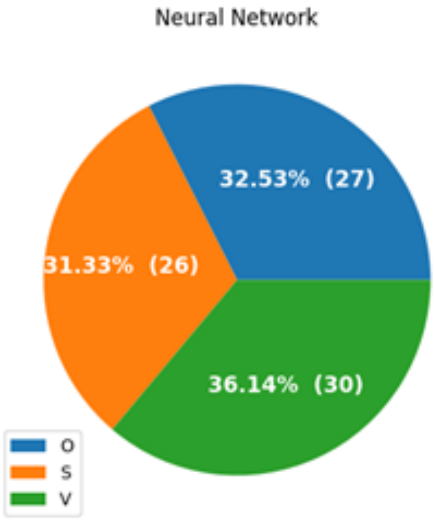


Figure 4.1.3: Neural Network Prediction Results before Scaling

4.1.4 Random Forest Classifier

We tried out different criterions and n_estimators values for this classifier. The various n_estimators values did not display much change in the accuracy values and were disregarded. Before the scaling, both the criterion Gini and Entropy resulted in high accuracy scores at 0.94. Entropy was what was chosen because the scores were quite steady. Gini produced great results but had some fluctuations in the scores and was not as consistent as Entropy for both before and after scaling.

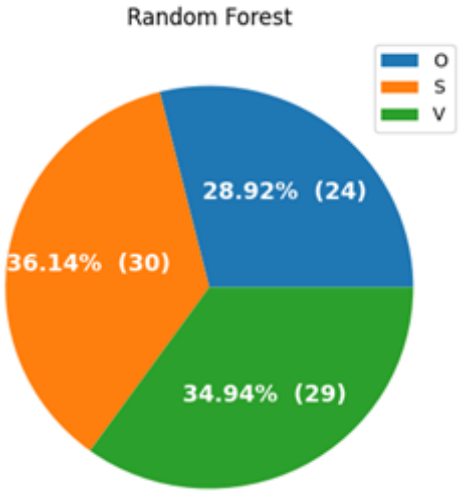


Figure 4.1.4: Random Forest Prediction Results before scaling

4.1.5 Optimal Classifier

Figure 4.1.5 a pie chart of what the actual classification should look like with 100 percent accuracy(how the testing data was split among shapes) and table 4.1 is the score of each model:

Classification Model	Accuracy Score
Naïve Bayes	0.80
K-Neighbors	0.78
Neural Network	0.84
Random Forest	0.94

Table 4.1: Accuracy score of all models used before scaling

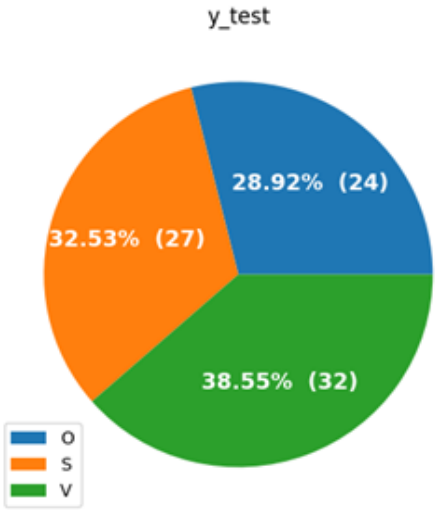


Figure 4.1.5: Data Split in testing data

Before the scaling, the random forest model performed the best with a 10 percent difference with the runner-up, the neural network model. The k-neighbors had the lowest accuracy score but produced nearly the same results as the naïve bayes model. Overall, all models had decent accuracy scores. With more data and extensive tuning, the accuracy of these models will only become more promising.

4.2 Analyzing Classification Models After Scaling

These models that we used are the same as before Scaling, including parameters, except having StandardScaler running before creating the models.

4.2.1 Naïve Bayes Classifier

Surprisingly, after the scaling the accuracy is still the same as without the scaling, which was 0.79. Like previously stated, this model is not of our focus on this project, which means tuning the parameter to optimal can show a difference between with and without scaling.

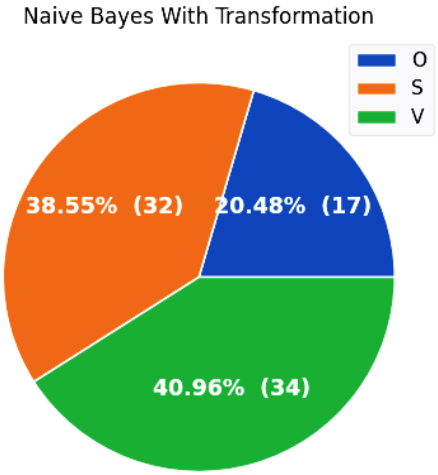
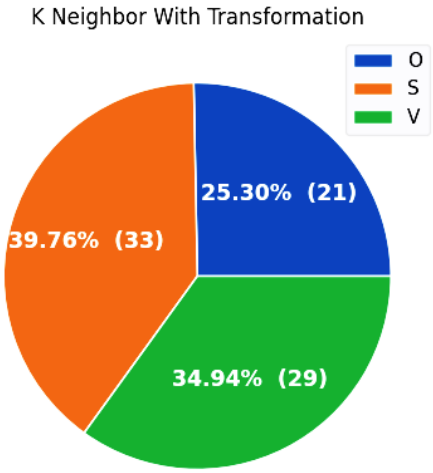


Figure 4.2.1: Naive Bayes Classification after Scaling

4.2.2 K-Neighbors Classifier



The accuracy improved 10 percent compared to not scaling. In this case, we got 0.819 and consistently around this range. In fact, from our observation this model is the one that benefits the most from scaling compared to other models. Explanation can be that when grabbing nearest neighbors, the group gives the model a point of comparison, which gives more accurate prediction.

Figure 4.2.2: KNeighbor Classification after Scaling

4.2.3 Neural Network Classifier

Similar to K-Neighbors, this model also has an increase in accuracy around 6 percent. In this case, we usually received around the range of 0.916. Though this is a huge boost in performance, we have to keep in mind that it is only based on a very limited amount of samples that we have. Since Neural Networks look at the weight of each attribute from data, it is reasonable to think the standard deviation scaling makes the weight more accurate. From this observation, it is possible

to think that this model has the most potential to raise the accuracy further with large amounts of samples. Though with or without scaling, that is unsure.

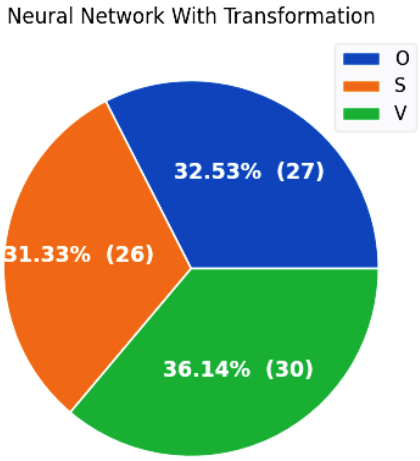
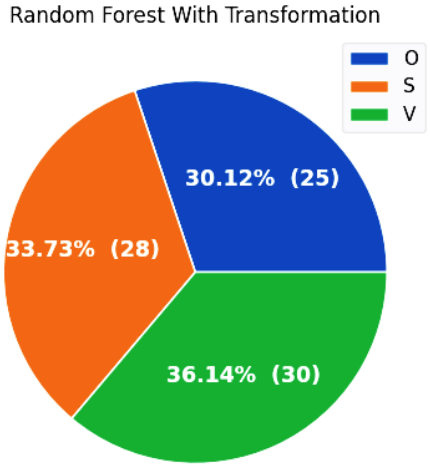


Figure 4.2.3: Neural Network Classification After Scaling

4.2.4 Random Forest Classifier



In this case we got the result the same as before scaling, which is 0.94. However, though many test runs, we find that this can be inconsistent. Sometimes it would boost up the accuracy compared to before scaling, while some other times it would decrease. Though the variation in accuracy is not too far from the original one, up to 5 percent at most. We come to a conclusion that the scaling is not very suitable to this model.

Figure 4.2.4: Random Forest Classification after scaling

4.2.5 Optimal Classifier

Classification Model	Accuracy Score
Naïve Bayes	0.79
K-Neighbors	0.81
Neural Network	0.91
Random Forest	0.94

Table 4.2: Accuracy score of all models used after scaling

Even after scaling, both the Neural Network and Random Forest remain the best models, as seen in table 4.2. It might seem that Random Forest is still the best model in this case, but due to the inconsistency of it after scaling, it would be reasonable to think Neural Network is the best model overall because of its consistency. Though Random Forest can be a lot more consistent if the samples were larger.

5. Limitations and Extensions

One thing which could improve our project was collecting more data points from more users to get a more varied dataset which would help form a model to better interpret real world drawings. Looking back we could have smoothed the data before stepping through the process of counting peaks and valleys to minimize small bumps near a valley which is evident from figure 3.2.3.

By taking a closer examination on data recording, we observed the way in which a user holds the phone when drawing can greatly affect the recording. Through trialing hand positions using the OnePlus 6 found the phone can give a higher reading the more the front of the phone is tilted upwards or downwards (probably due to sensor location). We believe using a similar phone would reduce inconsistencies from where sensors are located on the phone to sensor capabilities.

When gathering data, it would have been optimal if we had an exact procedure to follow. For instance, instead of a rather vague description of the size of shape to draw, we could have printed the shapes out to make sure all participants were drawing the same size. Further, how steady the individual's hand was prior to drawing the shape could have been critical factors. Instead of giving a range of time to draw each shape, we could have set an exact time limit, where each drawing takes over the entire given interval. Of course, doing so would reduce the 'real-world' variance the model would be trained on.

6. Accomplishment Statements

Important to note this report exceeds the 2n page limit due to the number of figures, otherwise exceeds the limit by 1 page without figures.

Balrick Gill

- Extracted useful parameters for modeling from raw user data (peaks and valleys, standard deviations)
- Created plots for parts 3 and diagram for part 2
- Collected user data from 2 individuals
- Wrote section 2, Data collection
- Wrote section 3 Data Transformation except 3.3
- Worked on portions of limitations and extensions (part 5)
- Edited and formatted final document

Quoc tuong Tran

- Setup the skeleton for scripts to run in different stages (cleaning and analyzing). So that it can handle scaling, as long as user samples are in correct format (seen in readme.MD).
- Nurmerized all columns from collected data to improve predictions.
- Provide a scaled version for all models using StandardScaler.
- Script displays stages while running.
- Wrote section 3.3, Scaling Data
- Worte Section 4.2, Analyzing Classification Models After scaling

Goeun Jang

- Collected linear acceleration and g-force data on individuals, used to analyze shape data
- Created models for machine learning analysis including GaussianNB, KNeighbors classifier, Neural Network classifier, Random Forest classifier to determine which model produced optimal results for data before scaling
- Worked with the different parameters of the models to create a steady outcome every time the program ran
- Enumerated predicted values to create pie charts for visualization of models
- Formed the skeleton report as well as writing a portion of the introduction (Part 1), the machine learning analysis (Part 4.1) and a portion of the limitations (part 5) of the report