DATA1030 Final Project Report-- Hand Gesture Recognition

Cangcheng Tang

Banner ID: B01628536

Brown University

GitHub: https://github.com/tangcc35/DATA1030-Project-Cangcheng.git

1. Introduction

The target variable is different "Class" for gestures and "User". This project's main purpose is to train a model to recognize hand gestures based on the coordinates of 12 joints of the hand. There is also an exploratory objective: identifying users.

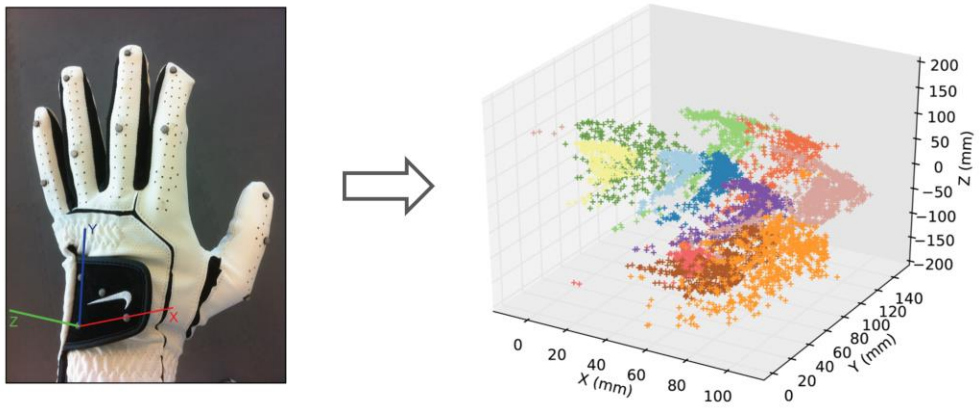
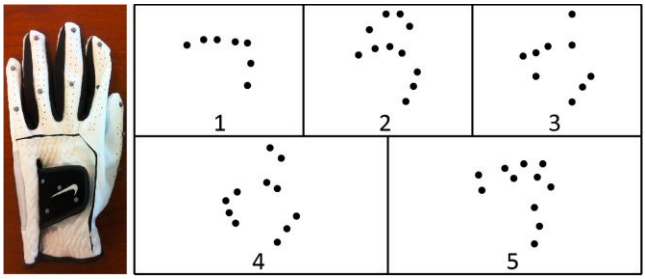


Figure 1: Record markers in a 3D space

As the target variable is categorical, not continuous, classification methods and models will be used to solve the problems above.



$1 \mapsto$ Fist (with thumb out),
$2 \mapsto$ Stop (hand flat),
$3 \mapsto$ Point1 (point with pointer finger),
$4 \mapsto$ Point2 (point with pointer and middle fingers),
$5 \mapsto$ Grab (fingers curled as if to grab).

Figure 2: Gestures and markers

Hand gesture recognition is a very powerful tool in interaction with an electronic device. Such technology allows hands-free control over smartphones, computers, or even smart household appliances

2. EDA

2.1 Missing values

2.1.1 Reason for missing values

Markers are placed on fingernails, joints and knuckles, but are unlabeled, so the camera used to obtain coordinates can't tell which marker belongs to which finger. Therefore, coordinates are more likely to be missing if they belong to a lower-ranked marker. This is missing not at random.

Figure 3: Examples of how markers are label

2.1.2 EDA on missing data

Shown in the figures below, lines are separated by class and user. This implies missing values depend on classes and users.

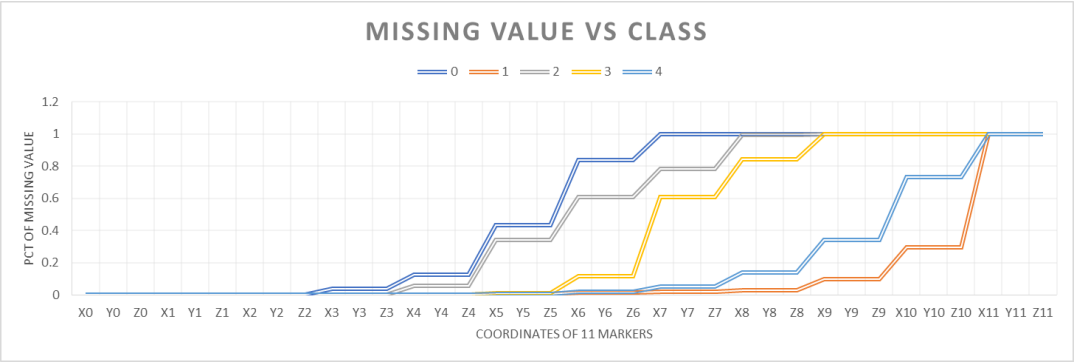


Figure 4: Missing value patterns VS class

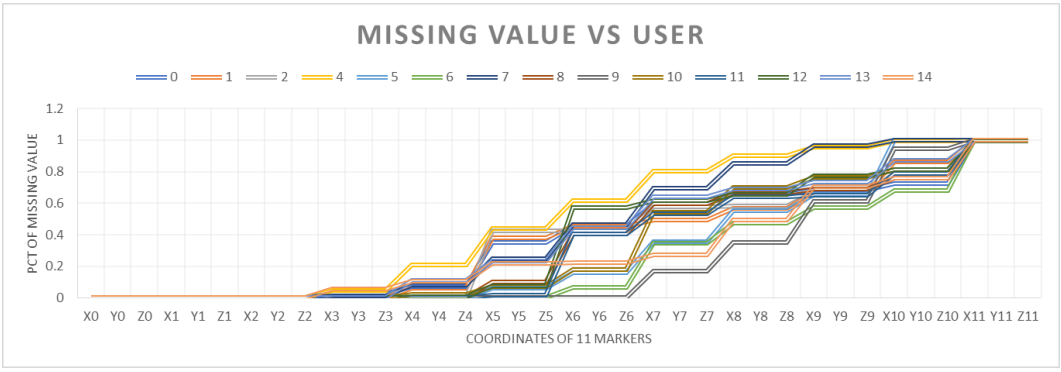


Figure 5: Missing value patterns VS user

To utilize this information, a new column was created to show which markers are missing at each data point. Two stack plots below shows the relationships between missing values and gestures and users.

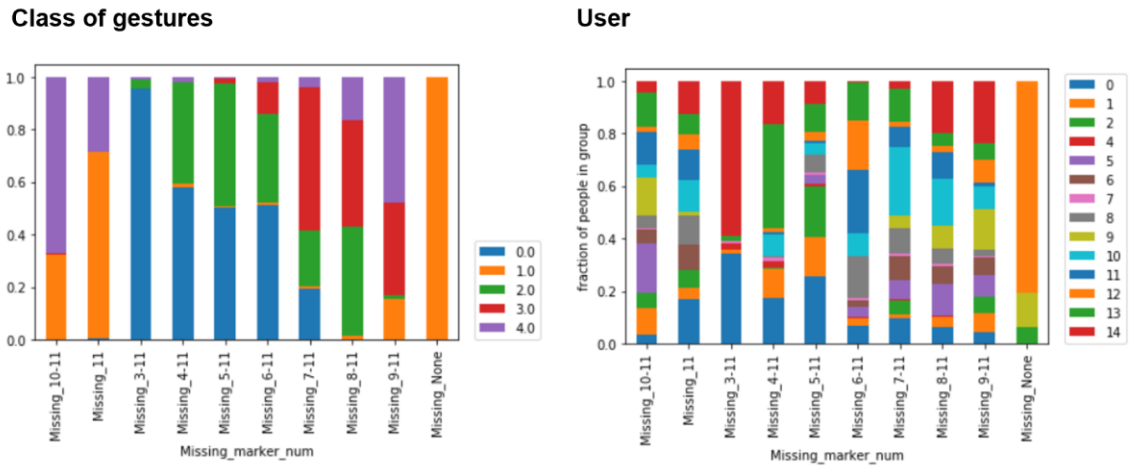


Figure 6: Number of missing values VS class and user in stack

2.2 Balance of Data

For class of gestures data is quite balanced, as shown below. Baseline for gesture prediction is 20.93%.

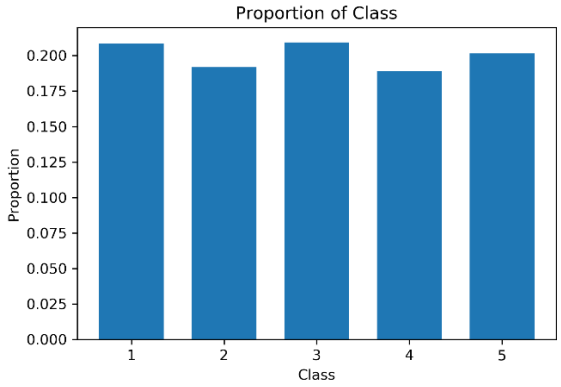


Figure 7: Proportion of Class

For users, data is not very balanced. Baseline for user prediction is 12.25%.

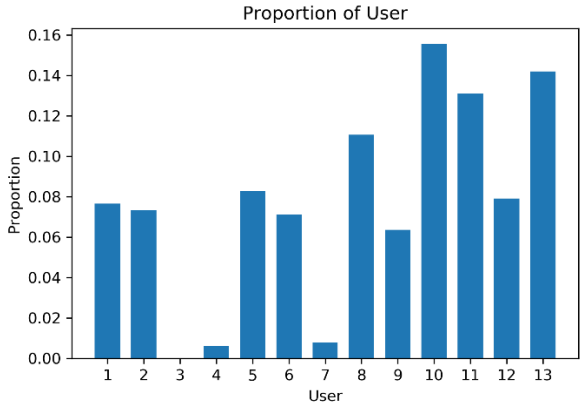


Figure 8: Proportion of User

2.3 EDA on coordinates

2.3.1 Gesture as target

Y and Z coordinates distribute differently on different gestures.

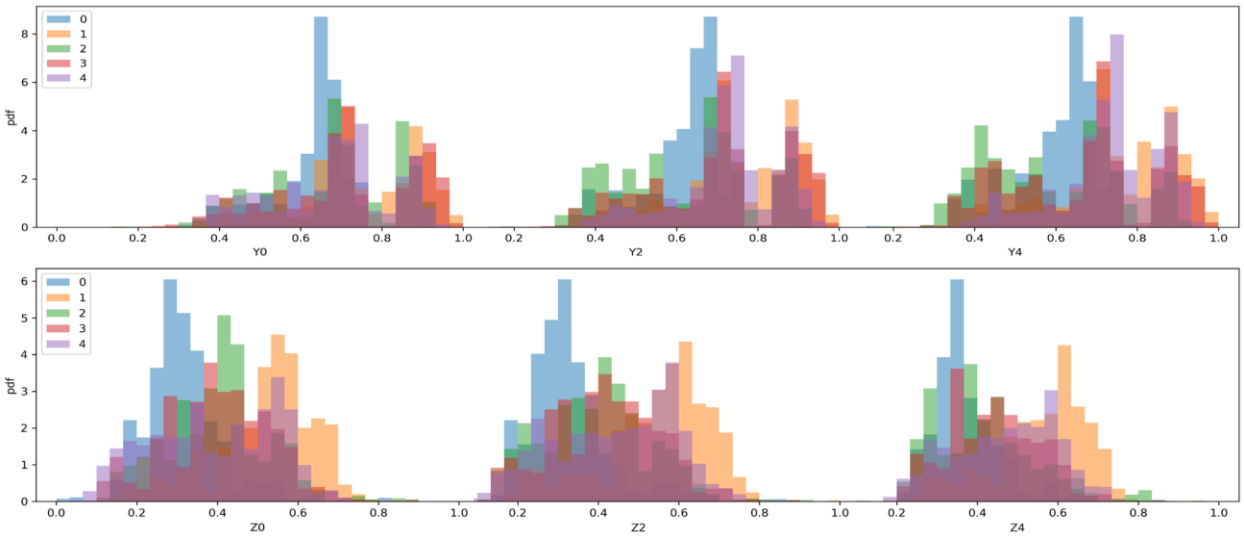


Figure 9: Y and Z's distributions given class of gesture

X coordinates don’t separate gestures that well, especially the first markers. Distributions of different classes are heavily overlapped.

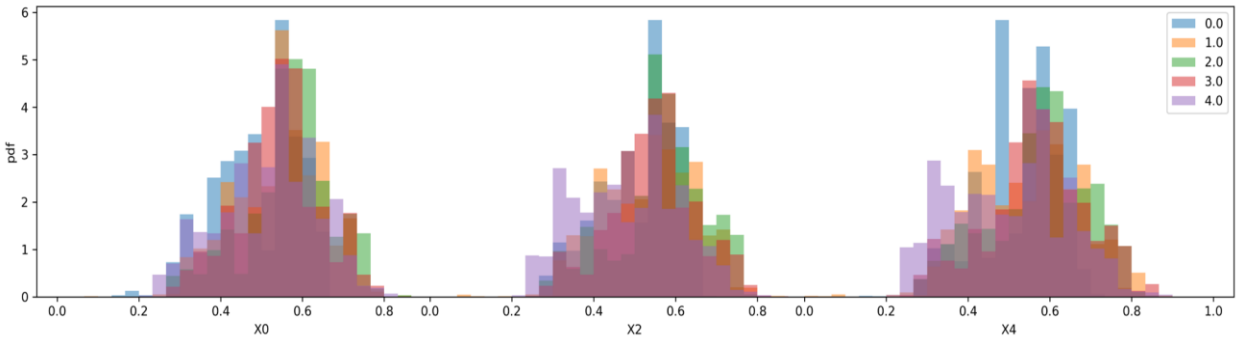


Figure 10: X’s distributions given class of gesture

2.3.2 User as target

X and Z coordinates distribute differently on different users.

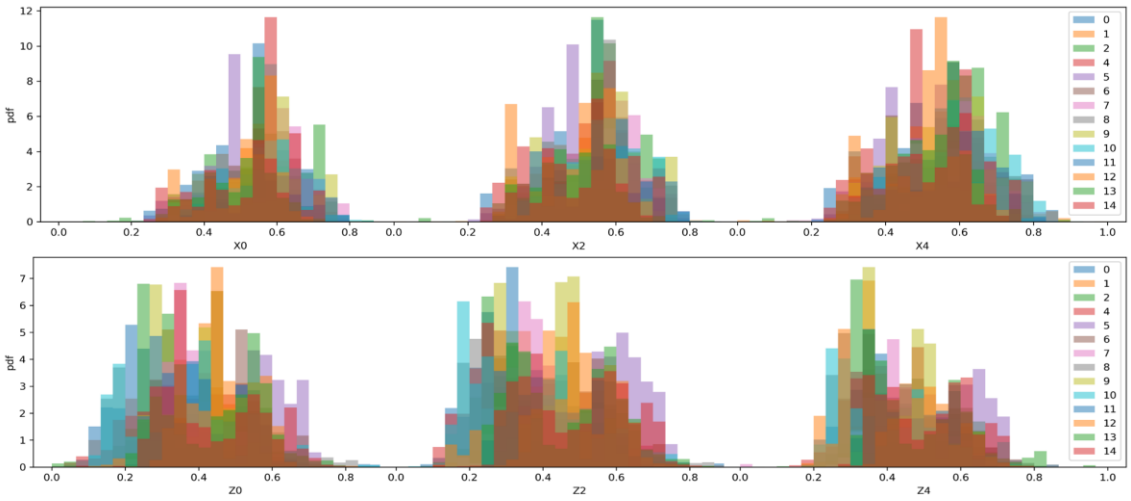


Figure 11: X and Z’s distributions given user

Y coordinates of the markers don't separate users that well. The distributions are similar given users.

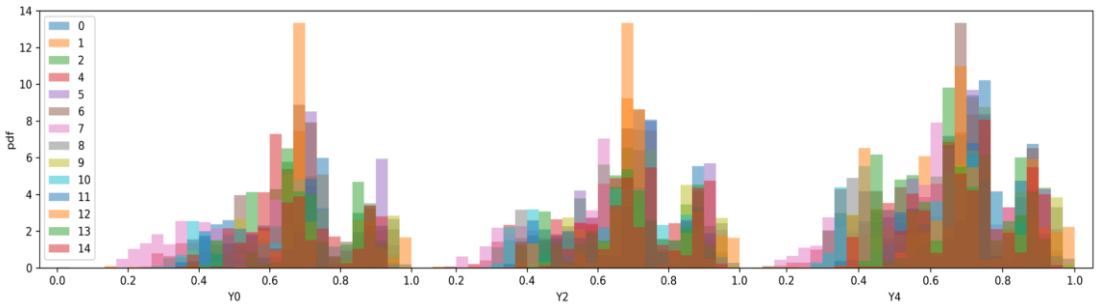


Figure 12: Y's distribution given user

3 Methods

There are too many missing values in this dataset, and they are missing not at random. XGBoost would be a reasonable choice, SVC is also used, but mostly for comparison.

3.1 Predicting class of gestures

3.1.1 Split train, cv and test

To prevent data leakage, train, cv and test sets were split by users. For example, if a user was in train, they can't be in cv, nor test. After the split, `User` is dropped.

K-fold could introduce data leakage in this case. Therefore, the basic train, cv, test sets were split. Train was 60%, cv and test were both 20%.

3.1.2 XGBoost

3.1.2.1 Preprocess

MinMaxscaler was used to preprocess all coordinates and `Missing_marker_num`. The preprocessors were only fitted using training sets, then cv and test were transformed to prevent data leakage.

3.1.2.2 Tuning hyperparameters

λ and `max_depth` were tuned. λ was tuned from 0, 1, 10, 20, 50. `max_depth` was tuned from 4, 5, 6, 7, 8. There were 25 combinations in total.

The best λ and `max_depth` are 10 and 6, both are in the middle of the corresponding ranges.

3.1.2.3 Performance metric

Metric to measure models was accuracy. Since the job is gesture recognition, the most important thing is to classify correctly. Accuracy was the most suitable metric, measuring the overall performance of a classification model.

3.1.3 SVC

3.1.3.1 Preprocess

MinMaxscaler was used to preprocess all coordinates and `Missing_marker_num`. To use an SVC model, no missing values should be in the dataset. Columns with missing values more than 10% were dropped, as it doesn't make sense to impute values missing at random. I tried my best to preserve information of missing values using column: `Missing_marker_num`. The preprocessors were only fitted using training sets, then cv and test were transformed to prevent data leakage.

3.1.3.2 Tuning hyperparameters

The default rbf SVM classifier was used. C and γ were tuned. I tried five Cs, the range was 0 to 6 in log space. I also tried five γ s, the range is -5 to 2 in log space. There were 25 combinations in total.

The best C and ` γ ` are 1000 and $10^{-2/3}$, which are in the middle of the corresponding ranges.

3.1.3.3 Performance metric

Metric to measure the models was accuracy. Reason is the same as above.

3.2 Predicting users

3.2.1 General idea

To get a better prediction on users, the gesture is set to be a known factor. In other words, the following methods are for predicting users given a gesture. To maximize accuracy, XGBoost and SVC were trained on all 5 gestures, so that I could find the best gesture for predicting users, along with the best corresponding model.

3.2.2 Split train, cv and test

I first extracted 5 sets of data by class of gestures. Split the subsets into train, cv and test, where test took up 20% of the whole subset, and cv taking 20% of the rest subset. After the split, column `Class` is dropped.

3.2.3 XGBoost

3.2.2.1 Preprocess

Methods for preprocessing is the same as the XGBoost part for predicting gestures.

3.2.2.2 Tuning hyperparameters

λ and `max_depth` were tuned for all 5 gestures. The range and number of values I tried for each gesture are the same. λ was tuned from 0, 1, 10, 20, 50. `max_depth` was tuned from 4, 5, 6, 7, 8. There were 25 combinations in total for each of the 5 gestures.

The best gesture is Class 5, "grab". Its best λ and `max_depth` are 0 and 6, the depth is in the middle of the corresponding range.

3.2.2.3 Performance metric

Metric to measure the models was accuracy. Reason is the same as above.

3.2.3 SVC

3.2.3.1 Preprocess

Methods for preprocessing is the same as the SVC part for predicting gestures.

3.2.3.2 Tuning hyperparameters

The default rbf SVM classifier was used. C and γ were tuned for all 5 gestures. I tried five Cs, the range was 0 to 6 in log space. I also tried five γs, the range is -5 to 2 in log space. There were 25 combinations in total for each of the 5 gestures.

The best gesture is Class 4, "pointing with two fingers". Its best C and ` γ ` are $10^{1.5}$ and $10^{-2/3}$, they are in the middle of the corresponding ranges.

3.2.3.3 Performance metric

Metric to measure the models was accuracy. Reason is the same as above.

4 Results

4.1 Predicting class of gestures

4.1.1 Average accuracy and uncertainty

Both models performed okay. From tables below, we can see that XGBoost has higher average accuracy and lower standard deviation, and outperformed SVC constantly. XGBoost is better suited for the job here. Lower accuracy for SVC may be caused by too much missing values.

Table 1: Results of predicting gestures

Model Comparison Summary, Predicting Gestures				
Model	Baseline	Average accuracy	Standard deviation	Standard deviation above baseline
XGBoost	20.93%	88.85%	0.0405	16.78
SVC		79.65%	0.0414	14.2

Model Comparison for Each Random State, Predicting Gestures					
Model	Random State 1	Random State 2	Random State 3	Random State 4	Random State 5
XGBoost	91.72%	83.90%	85.45%	88.22%	94.95%
SVC	81.98%	81.98%	74.18%	75.34%	84.77%

4.1.2 Averaged confusion matrix

4.1.2.1 XGBoost

Generally, the model predicts gestures well. It sometimes confuses Class 3 with Class 4, which are pointing with one finger two fingers. Another confusion is on Class 5 and Class 2, which are grab and fist. It makes sense for these gestures to be confused.

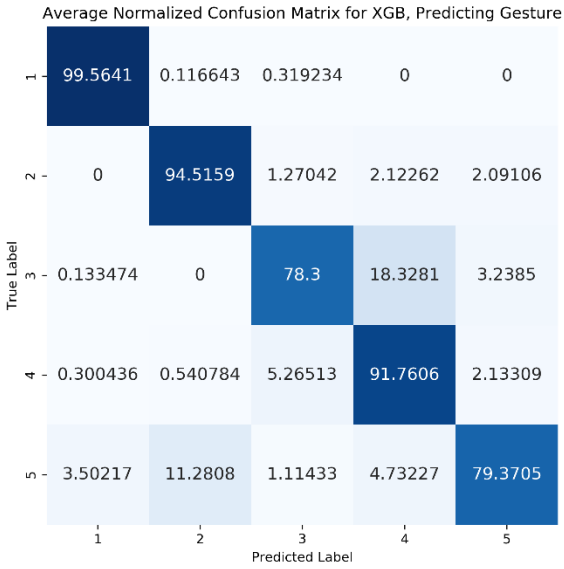


Figure 13: Average normalized confusion matrix for XGB, predicting gestures

4.1.2.2 SVC

Generally, the model predicts gestures less accurately. It also had confusion on recognizing between Class 3 and Class 4, as well as Class 5 and Class 2. In addition, SVC also confuses about Class 5 and Class 4, Class 3 and Class 1. These confusions may be caused by missing values.

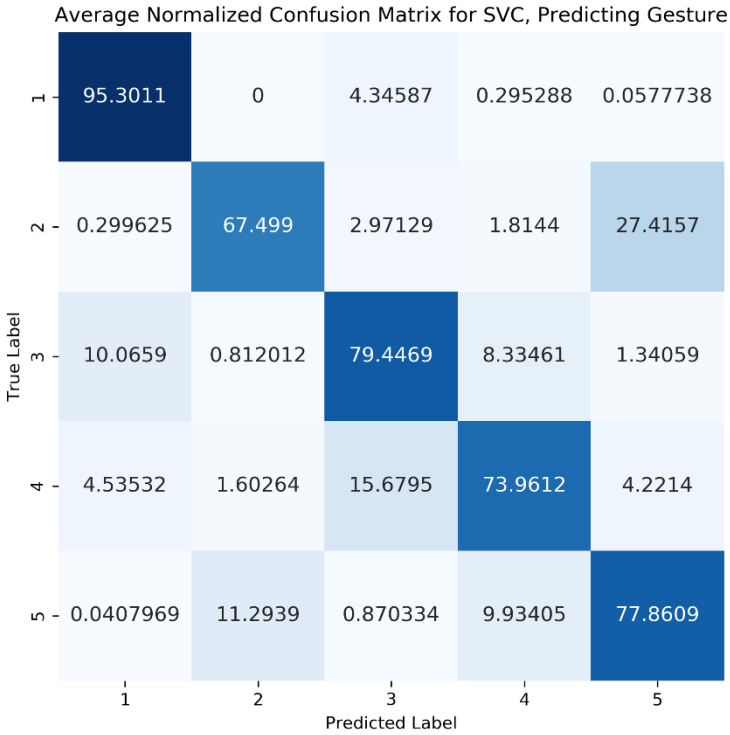


Figure 14: Average normalized confusion matrix for SVC, predicting gestures

4.1.3 Local feature importance

4.1.3.1 XGBoost

Below are the force plots for the 6th data point, ordered from Class 1 to Class 5. True Class is 1. For this point, Y7, Z1, Z2 are important in deciding whether it is Class 1, they all voted for yes. For Class 2, Y7, Y8, Y9, Z4 are important, Y7, Y8, Y9 voted for no, but Z4 voted for yes. Similar interpretations can be drawn from other classes.

Figure 15: Force plot for 6th data point, predicting gestures

4.1.3.2 SVC

Below are the force plots for the 6th data point, ordered from Class 1 to Class 5. The actual Class for this point is 1, SVC didn't get this point right. For Class 1, X3, Z4, X1, Y4 are important in deciding whether it is Class 1. X3, Z4, Y4 voted for no, while X1 voted for yes. For Class 4, X3, X1, Y4, Y3 are more important, only X1 voted for no, Y3, Y4, X3 all voted for yes.

Figure 16: Force plot for 6th data point, predicting gestures

4.2 Predicting user

4.2.1 Average accuracy and uncertainty

From tables below, we can see that both models performed very accurately on recognizing users given their best gesture. Although standard deviations above baseline for SVC is higher, the XGBoost performed better as it constantly outperformed SVC.

Table 2: Results of predicting gestures

Model Comparison Summary, Predicting Users					
Model	Baseline	Best Gesture	Average accuracy	Standard deviation	Standard deviation above baseline
XGBoost	12.26%	Class 5	99.64%	0.0013	674.11
SVC		Class 4	97.67%	0.0007	1237.48

Model Comparison for Each Random State, Predicting Gestures					
Model	Random State 1	Random State 2	Random State 3	Random State 4	Random State 5
XGBoost	99.56%	99.68%	99.75%	99.43%	99.78%
SVC	97.73%	97.66%	97.60%	97.77%	97.60%

4.2.2 Averaged confusion matrix

4.2.2.1 XGBoost

The model performs very well in predicting users. Almost no confusions.

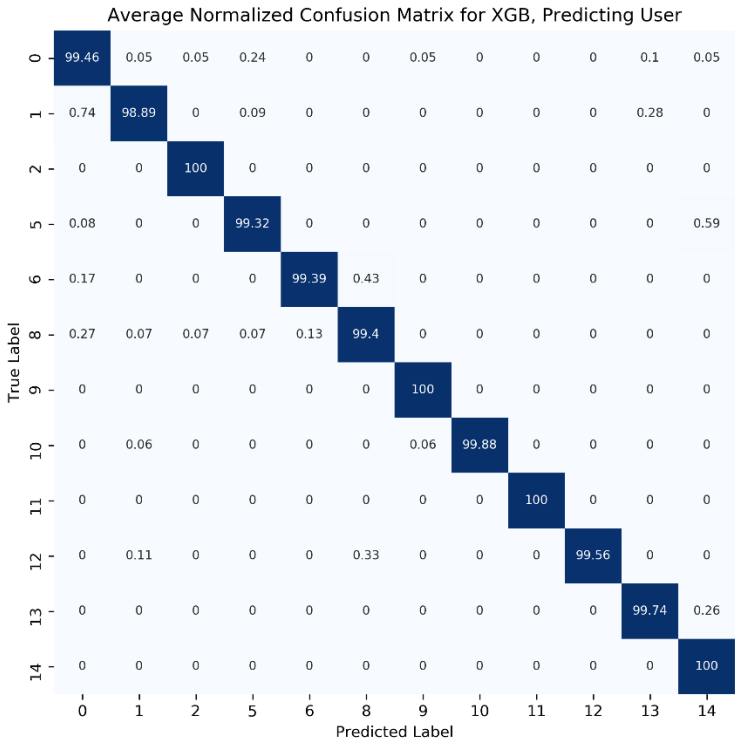


Figure 17: Average normalized confusion matrix for XGB, predicting users

4.2.2.2 SVC

There is some confusion for SVC, but generally it also did a good job, with only a few confusions on user 8 and 10.

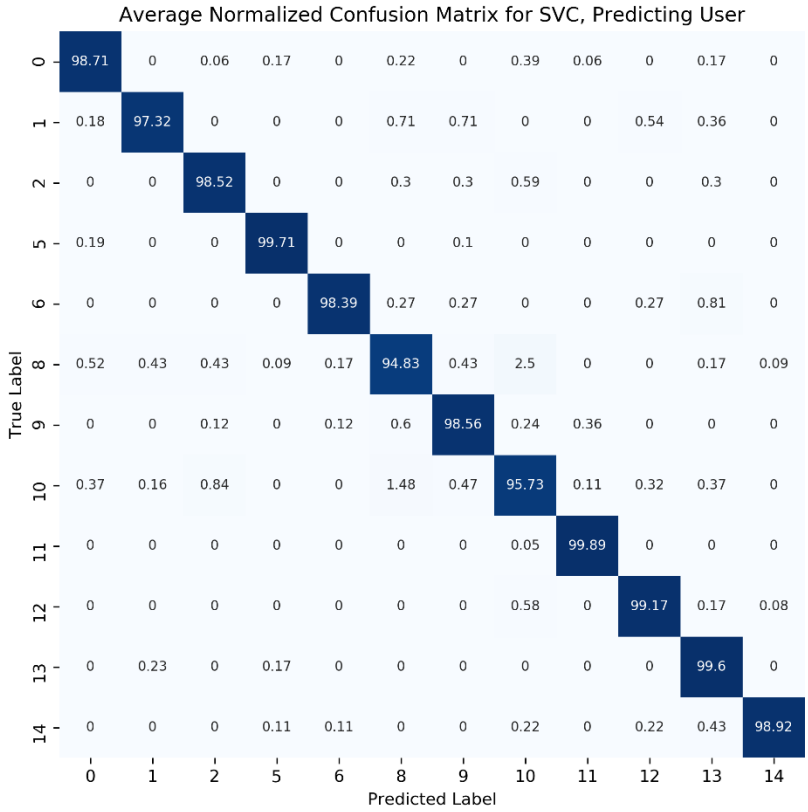


Figure 18: Average normalized confusion matrix for SVC, predicting users

4.2.3 Local feature importance

4.2.3.1 XGBoost

Below are the force plots for the 8th data point. The first plot is on User 0, the second plot is on User 1. True user is 1. For User 0, most of the important features voted for no. For User 1, most of the important features vote for yes.

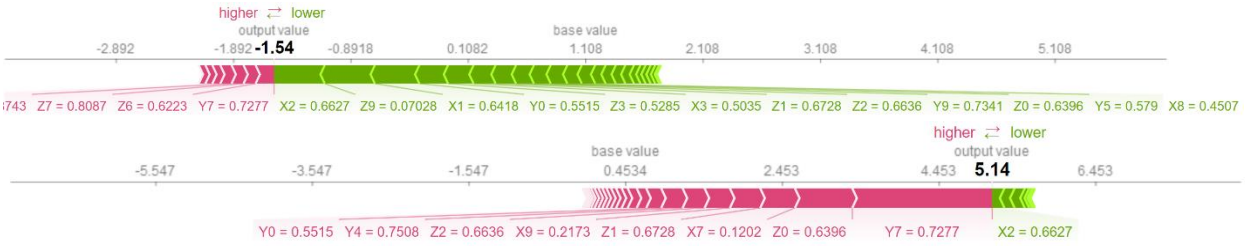


Figure 19: Force plot for 8th data point, predicting users

4.2.3.2 SVC

Below are the force plots for the 8th data point. The first plot is on User 0, the second plot is on User 1. Almost all features voted for no in User 0, yes for User 1.

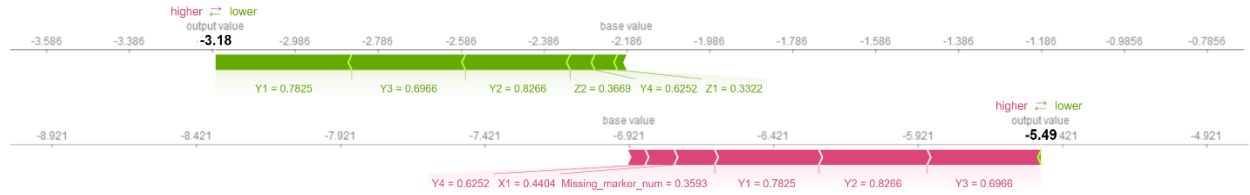


Figure 20: Force plot for 8th data point, predicting users

4.3 Real-life application

With hand gesture recognition, we can control our smartphones, computers, household appliances without touching them. You can answer phone calls with dirty hands in the kitchen. There is also no need to have a controller for each household appliances, everything can be controlled by waving hands.

Recognizing users allows for switching users seamlessly on a device. Therefore, one gesture can mean different commands for different users.

5 Outlook

When predicting gestures, the main weakness is distinguishing “pointing between two fingers” and “pointing with one finger”. This is indeed confusing for the models, especially SVC. A possible solution is adding a column that counts the number of fingers captured by camera through feature engineering.

As for models, I could tune more hyperparameters on XGBoost and SVC, squeeze more predictive power from these two models. Other machine learning models like neural network, logistic regression and KNN may have higher accuracy.

As for features, I could perform a more thorough EDA to better preserve information provided by missing values using feature engineering methods, this can help the models predict both gestures and users more accurately.

6 Reference

Dataset is from UCI machine learning repository:
https://archive.ics.uci.edu/ml/datasets/Motion+Capture+Hand+Postures

[1]. Gardner, A., Kanno, J., Duncan, C.A. and Selmic, R., 2014. Measuring distance between unordered sets of different sizes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 137-143).

[2]. Gardner, A., Duncan, C.A., Kanno, J. and Selmic, R., 2014, October. 3d hand posture recognition from small unlabeled point sets. In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 164-169). IEEE.