UNIVERSITY OF TEXAS ARLINGTON | COLLEGE OF ENGINEERING

Hand Posture Recognition

IE 6318 Project Report Under Guidance of
Dr. Shouyi Wang

By Mrudul Banka

1001547722

Contents

List of Figures

1. Introduction

Hand gesture recognition has gradually become a popular as a means of Human Computer Interaction (HCI). Hand gesture technology makes users' lives simpler, achieving "hands free" interaction through eliminating the need to hold or press the device. We use hand gesture to operate our cell phones, to play videogames like Wii, Xbox and to operate highly secure facilities. Some of the major industries which make use of hand gesture recognition daily are - Automotive sector, Consumer electronics sector, Transit sector, Gaming sector, smartphones, Defence Home automation, and Automated sign language translation.

To understand how to these various sectors, develop their hand recognition technique, we obtained data by performing different gestures in front of a camera. A Vicon motion capture camera system was used to record 12 users performing 5 hand postures with markers attached to a left-handed glove.

A rigid pattern of markers on the back of the glove was used to establish a local coordinate system for the hand, and 11 other markers were attached to the thumb and fingers of the glove. 3 markers were attached to the thumb with one above the thumbnail and the other two on the knuckles. 2 markers were attached to each finger with one above the fingernail and the other on the joint between the proximal and middle phalanx. Their positions were not explicitly tracked. Consequently, there is no a priori correspondence between the markers of two given records. In addition, due to the resolution of the capture volume and self-occlusion due to the orientation and configuration of the hand and fingers, many records have missing markers. Extraneous markers were also possible due to artifacts in the Vicon software's marker reconstruction/recording process and other objects in the capture volume. As a result, the number of visible markers in a record varied considerably.

1.1 Details of dataset:

The data presented here is already partially pre-processed. First, all markers were transformed to the local coordinate system of the record containing them. Second, each transformed marker with a norm greater than 200 millimetres was pruned. Finally, any record that contained fewer than 3 markers was removed. The processed data has at most 12 markers per record and at least 3.

Dataset link https://archive.ics.uci.edu/ml/datasets/MoCap+Hand+Postures - Data is a CSV file. A header provides the name of each attribute. A question mark '?' is used to indicate a missing value. A record corresponds to a single instant or frame as recorded by the camera system.

- Data Set Characteristics: Multivariate
- Number of Instances: 78095
- Number of Attributes: 38
- Attribute Characteristics: Real
- Date Donated: 2017-01-27

1.1.1 Classes

5 types of hand postures from 12 users were recorded. Each class is recorded as an integer. The class ID of the given record ranges from 1 to 5 with

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

Fig: 1 - Fist

Fig:2 - Stop

Fig:3 - Point1

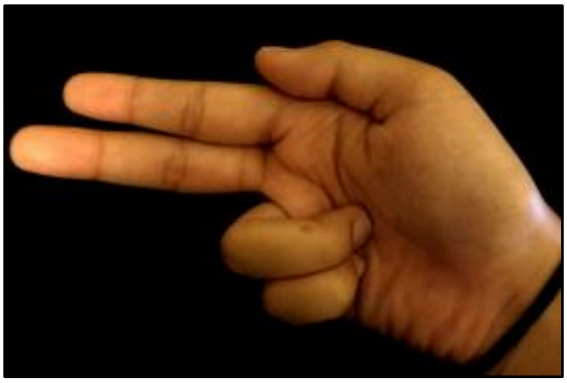


Fig: 4 - Point2

Fig: 5 - Grab

1.1.2 Features

Since there were 12 different people whose postures were recorded, the ID of the user that contributed the record is represented in the user column. Other features are X, Y, Z coordinates of all the markers used. No meaning other than as an identifier.

- 'Xi' – Real Number. The x-coordinate of the i-th marker position. 'i' ranges from 0 to 11.
- 'Yi' - Real Number. The y-coordinate of the i-th marker position. 'i' ranges from 0 to 11.
- 'Zi' - Real Number. The z-coordinate of the i-th marker position. 'i' ranges from 0 to 11.

2. Data Exploration

Data exploration is an approach similar to initial data analysis, whereby we use visual exploration to understand what is in a dataset and the characteristics of the data, rather than through traditional data management systems. We use the following charts to explore our data

2.1 Data Cleaning

On observing the dataset, we find out that there are number of missing values throughout the feature's variables. In order to visualize our data, we first need to get rid of missing and Nan values. We perform this procedure with the help of following program in python.

```
# read in all our data
post_read = pd.read_csv(r"G:\uta_acad\Sem 3\IE 6304 Data mining and analytics\Project\Postures.csv")
df = post_read.iloc[:, 0:14]
# print(df)
# set seed for reproducibility
np.random.seed(0)
# Read ? as Nan
df = df.replace('?', np.nan)
# get the number of missing data points per column
missing_values_count = df.isnull().sum()
total_missing = missing_values_count.sum()
# print(total_missing)
# percent of data that is missing
# print((total_missing/total_cells) * 100, "Percent")
# remove all the rows that contain a missing value
df = df.dropna()
```

Fig 6: Program for cleaning missing values

Now when we look at our dataset without missing values, we find that feature variables X5 to X11, Y5-Y11, Z5 – Z11 have all been removed because each of those features were missing many readings. Hence, to balance our dataset and make it ready for visualization, the above-mentioned features no longer play a part in our project.

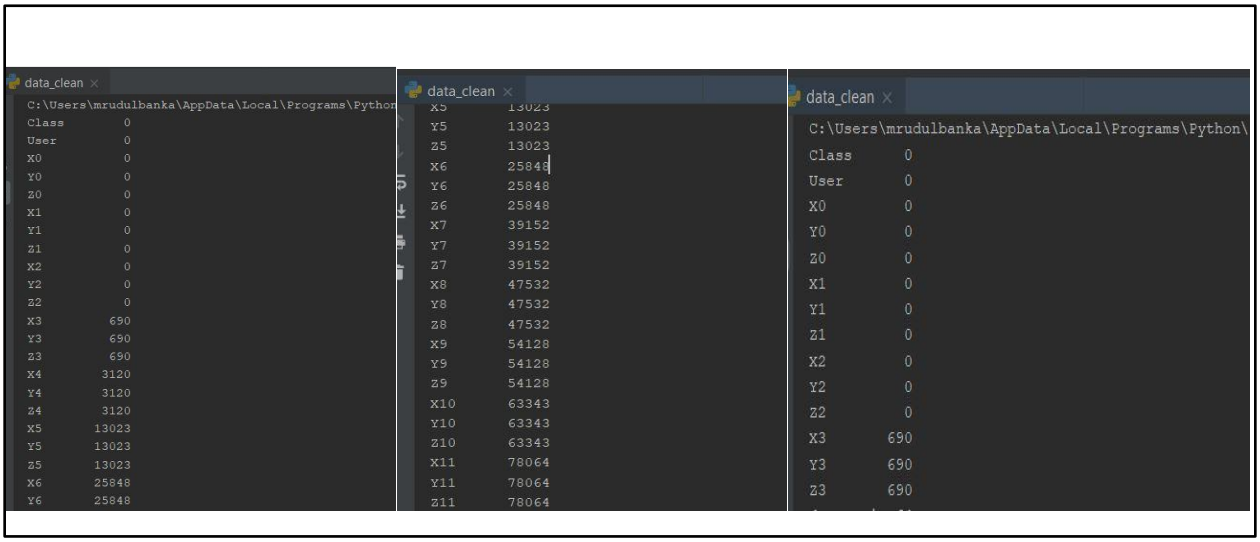
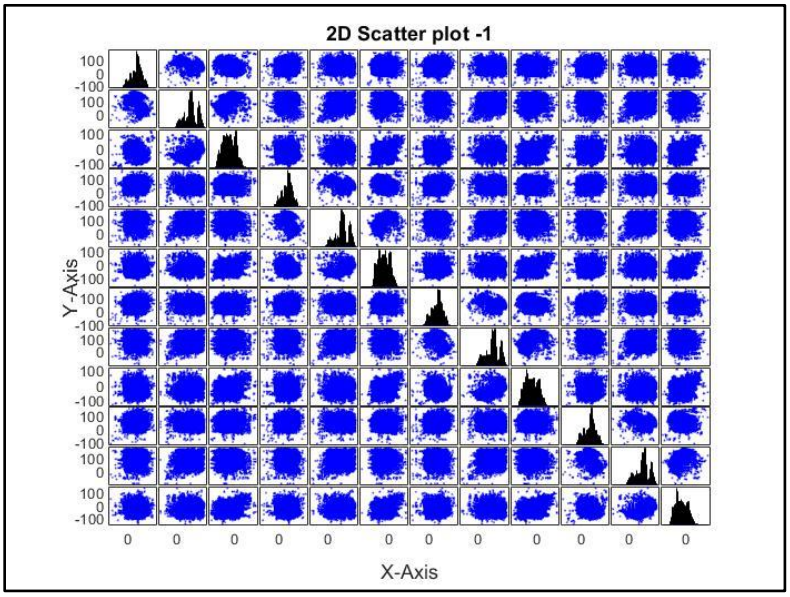


Fig 7: Data Cleaning: Before and After Missing values and truncated Feature matrix

2.2 2-D Scatter Plot



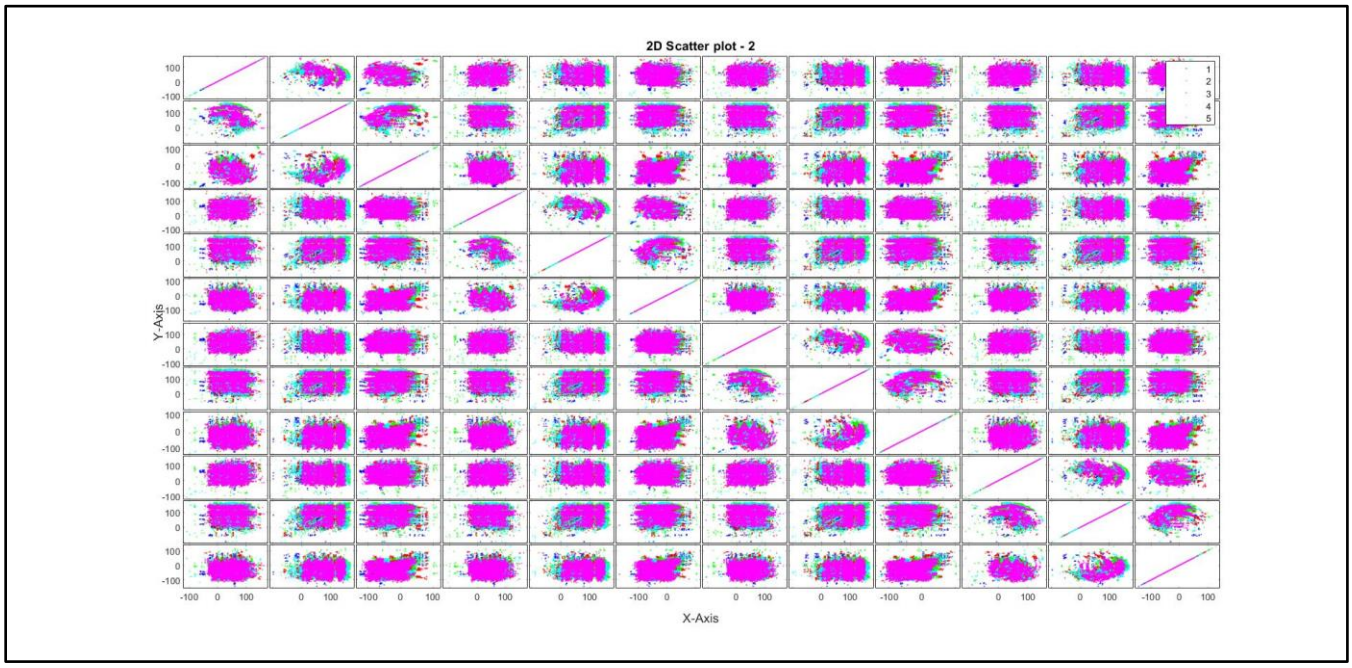


Fig 8: 2-D Scatter plot

Observations:

- From the figure above, we see that no two feature variables show an increasing or decreasing trend i.e. there is no strong correlation between two feature variables.
- All feature variable plots are randomly scattered – each variable provides useful information.

2.3 Parallel Co-ordinates Plot

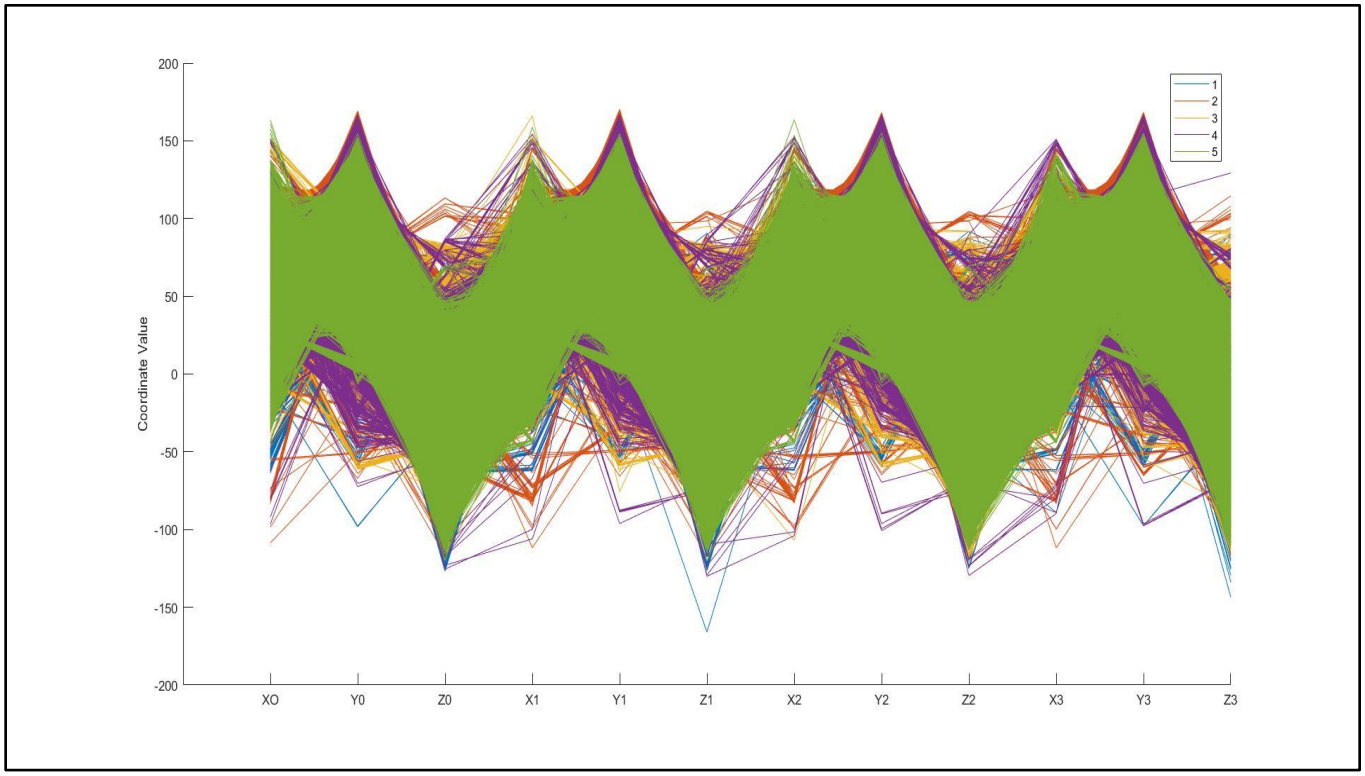


Fig 9: Parallel Coordinates plot

Observations:

- From the figure above, we see that most of our coordinate values lie between 160 to – 160 with noticeable outliers in feature variables of Z coordinate.
- All feature variables are randomly scattered – each variable provides useful information.

2.4 Correlation Matrix

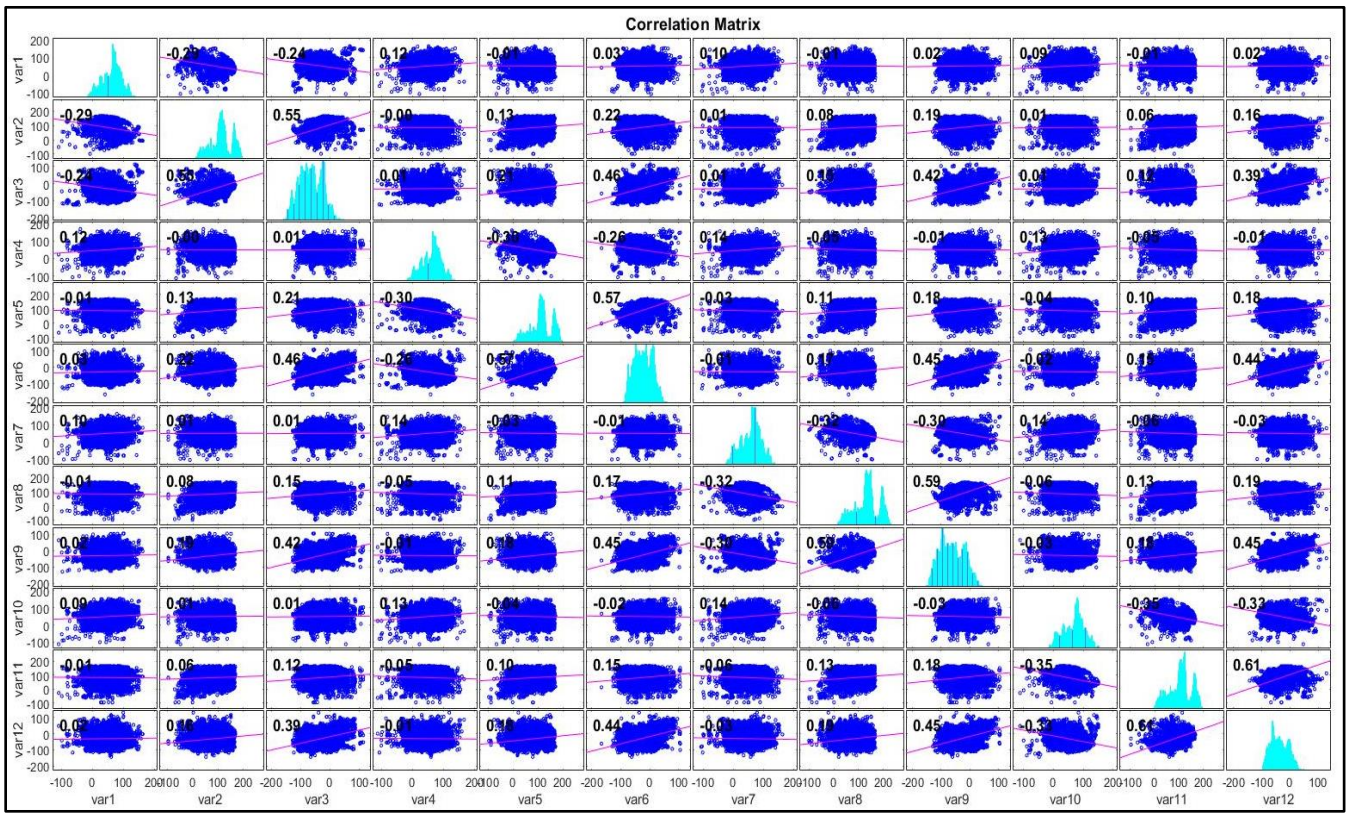


Fig 10: Correlation between all the features

Observations:

- From the figure above, we see that no two feature variables show an increasing or decreasing trend i.e. there is no strong correlation between two feature variables.
- The highest correlation obtained between Y3 and Z3 is 0.61 which is moderate and thus these variables can't be termed as corelated.
- All feature variable plots are randomly scattered – each variable provides useful information.

2.5 Feature Matrix Visualization

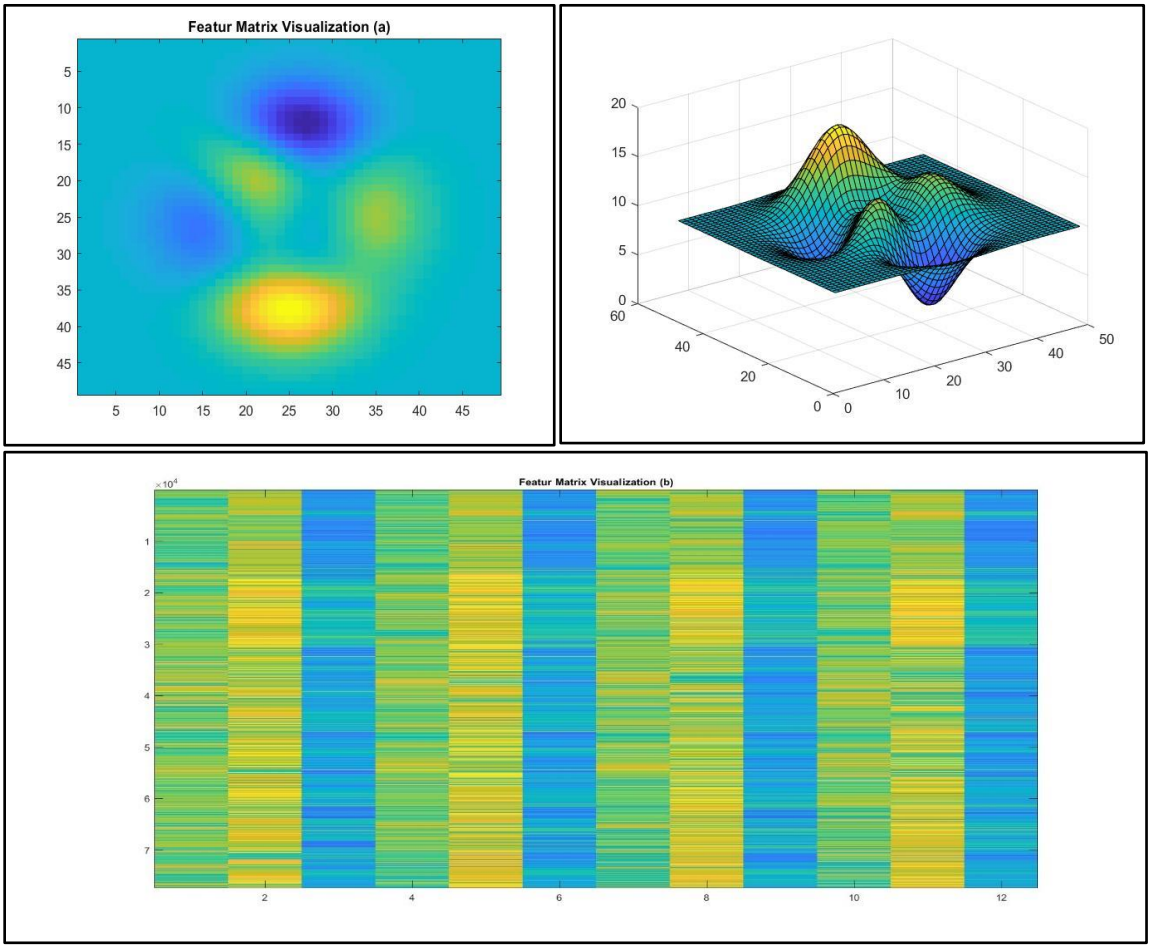


Fig 11: Visualization of our feature Matrix

Observations:

- From the figure above, we see all three coordinates provide important information to construct our classification model.
- The yellow shaded regions indicate points with low standard deviation while the dark blue indicates points with a high standard deviation. For our plots – yellow region lies between (20-30) on X axis, (35,45) on y axis. The blue concentrated region lies between (25,35) on X axis and (5,15) on Y axis.

2.6 Normalised Feature Matrix

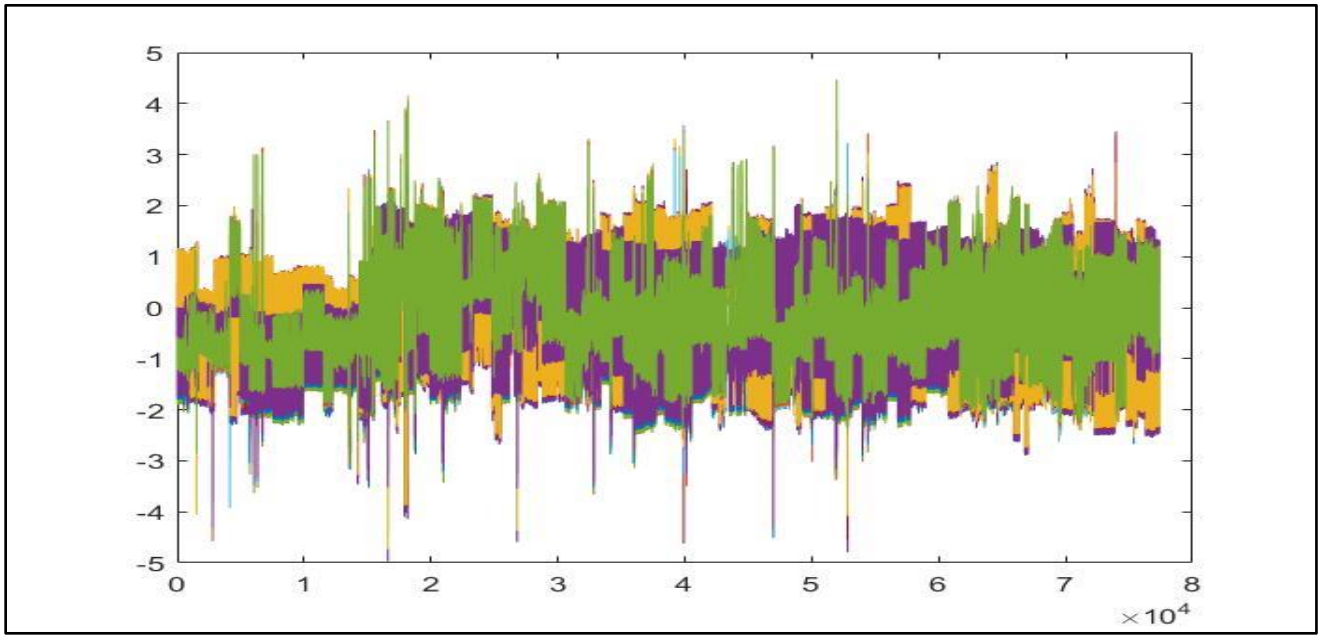


Fig 12: Normalized Feature Matrix

Observations:

- Typically, the attributes are normalized to prevent one attribute from dominating the plot. Histograms.
- From the figure above, we see that the normalized plot has very high standard deviations for some instances. This is due to readings obtained from 12 different users with unequal hand size.

2.7 Histogram

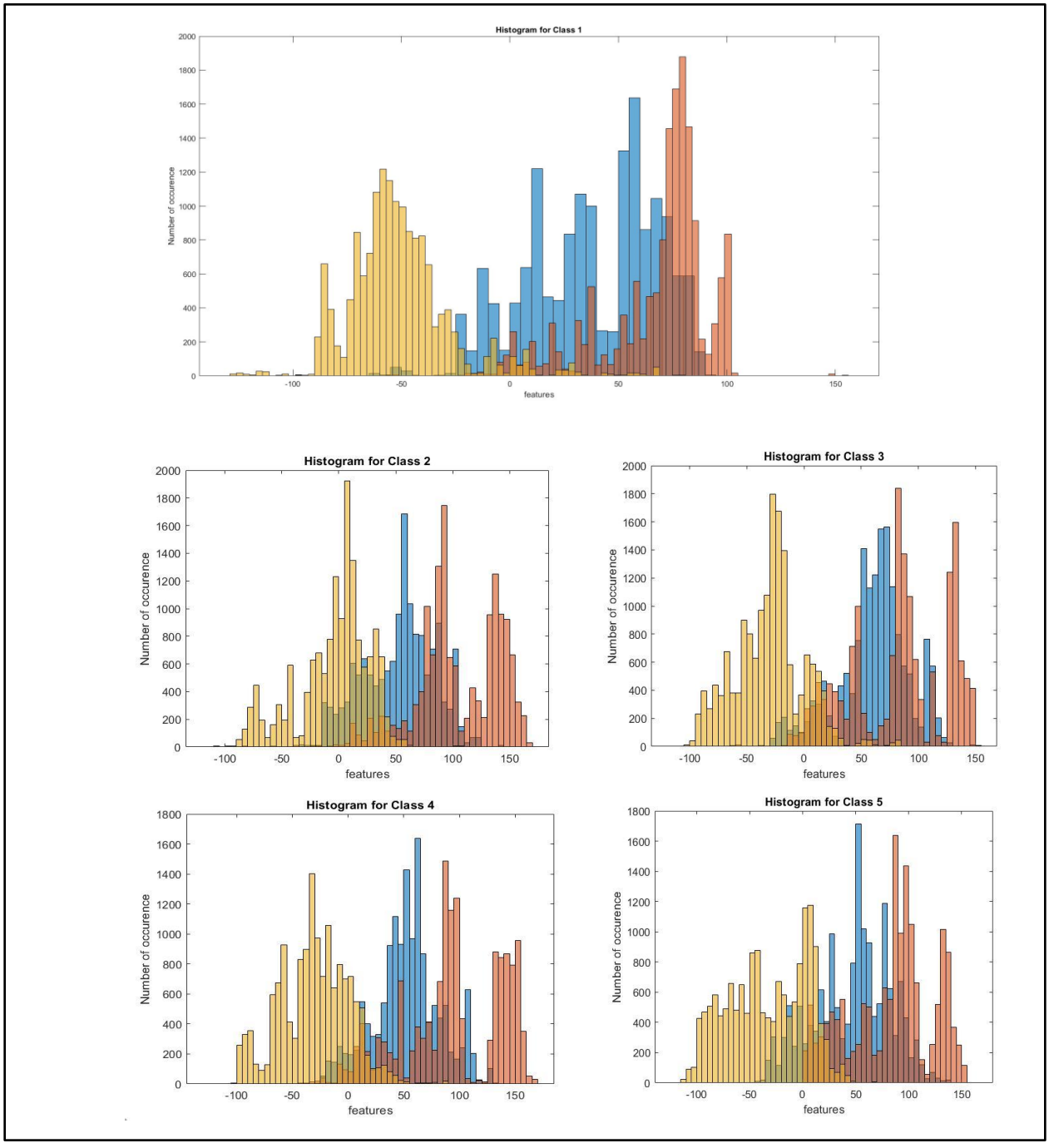


Fig 13: Histograms for features of 5 classes

Observations:

- From the histograms for all classes, we see that there are outliers for every class. Some of which in Class 1 are quite noticeable.
- No class has similar or highly overlapping feature variables.

2.8 Boxplots

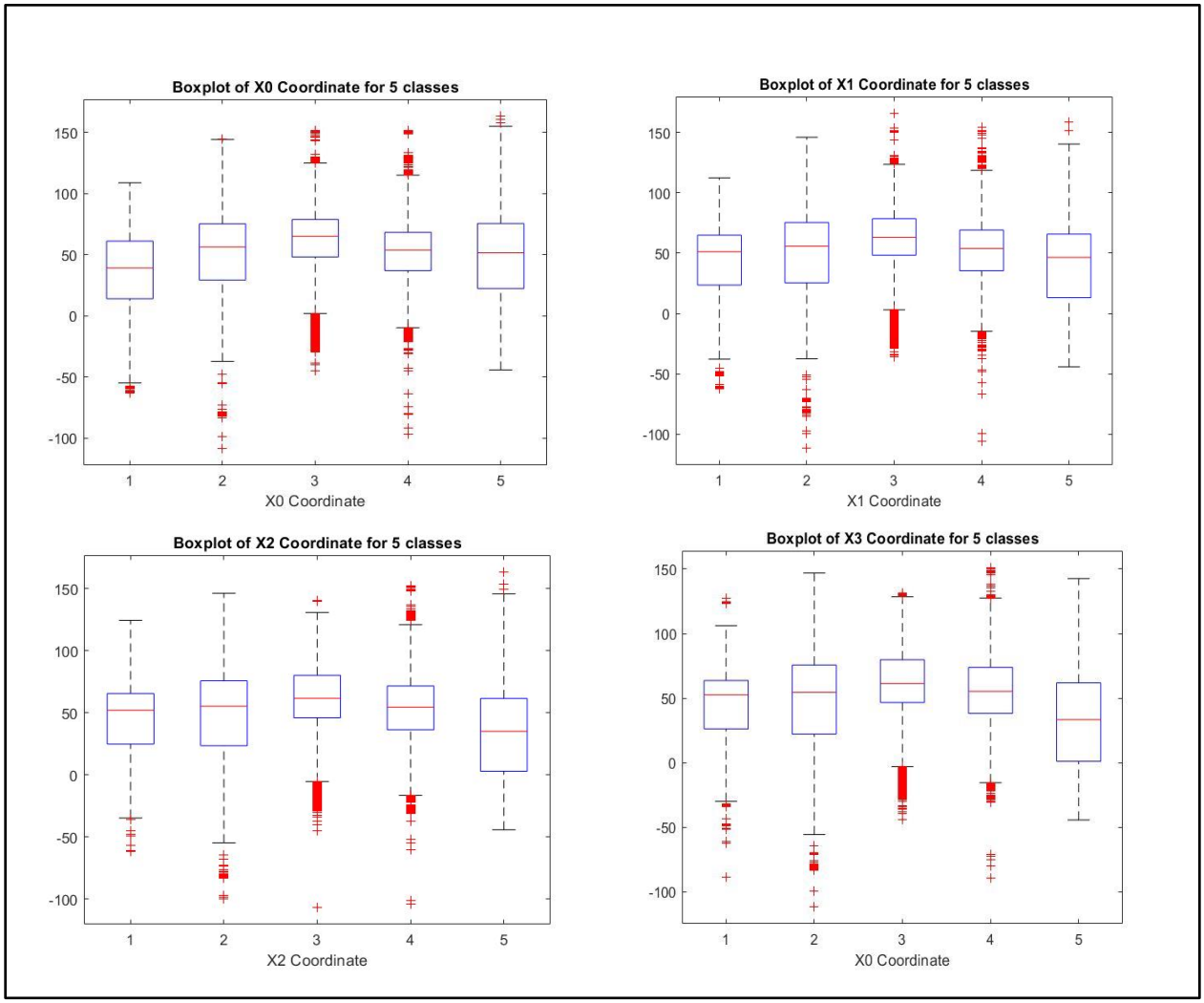


Fig 14: Box plots for X Co-ordinates of 5 classes

Observations:

- From the histograms for all X Coordinates, we see that there are outliers for every class except class 5. Some of which in Class 3 and class 4 are quite noticeable.
- 50^{th} percentile for each of the classes is in and around +50 mark.
- Points shaded in red, in general will be termed as outliers. However, in our dataset they are not because a difference in the size of hand can cause such deviation. Since, we gather data from 12 different people, everyone has different sizes of hands.

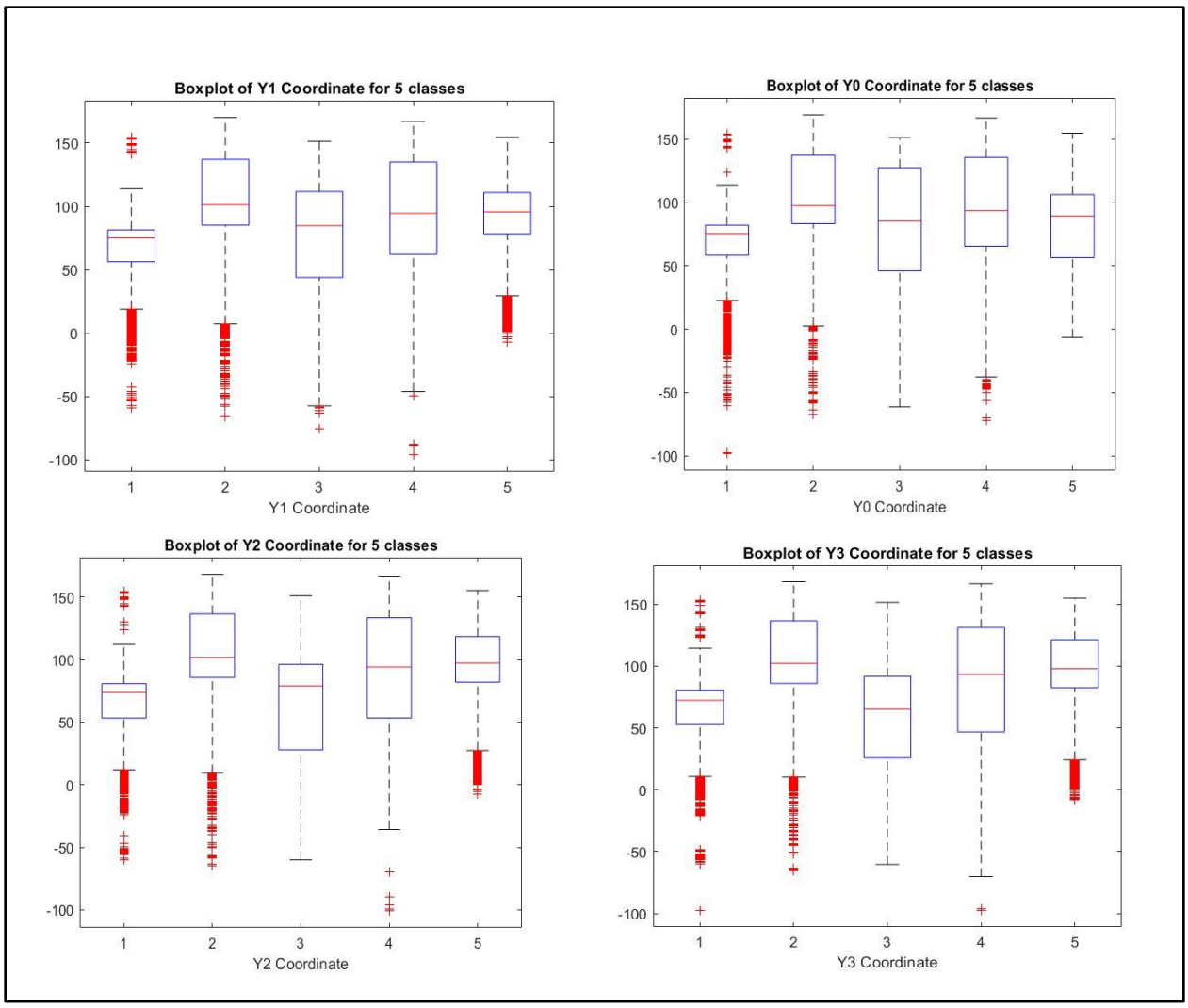


Fig 15: Box plots for Y Co-ordinates of 5 classes

Observations:

- From the histograms for all Y Coordinates, we see that there are outliers for every class except class 4. Some of which in class 1, class 2 and class 4 are quite noticeable.
- 50th percentile for each of the classes is rather scatter compared to X coordinates graph.
- Points shaded in red, in general will be termed as outliers. However, in our dataset they are not because a difference in the size of hand can cause such deviation. Since, we gather data from 12 different people, everyone has different sizes of hands.

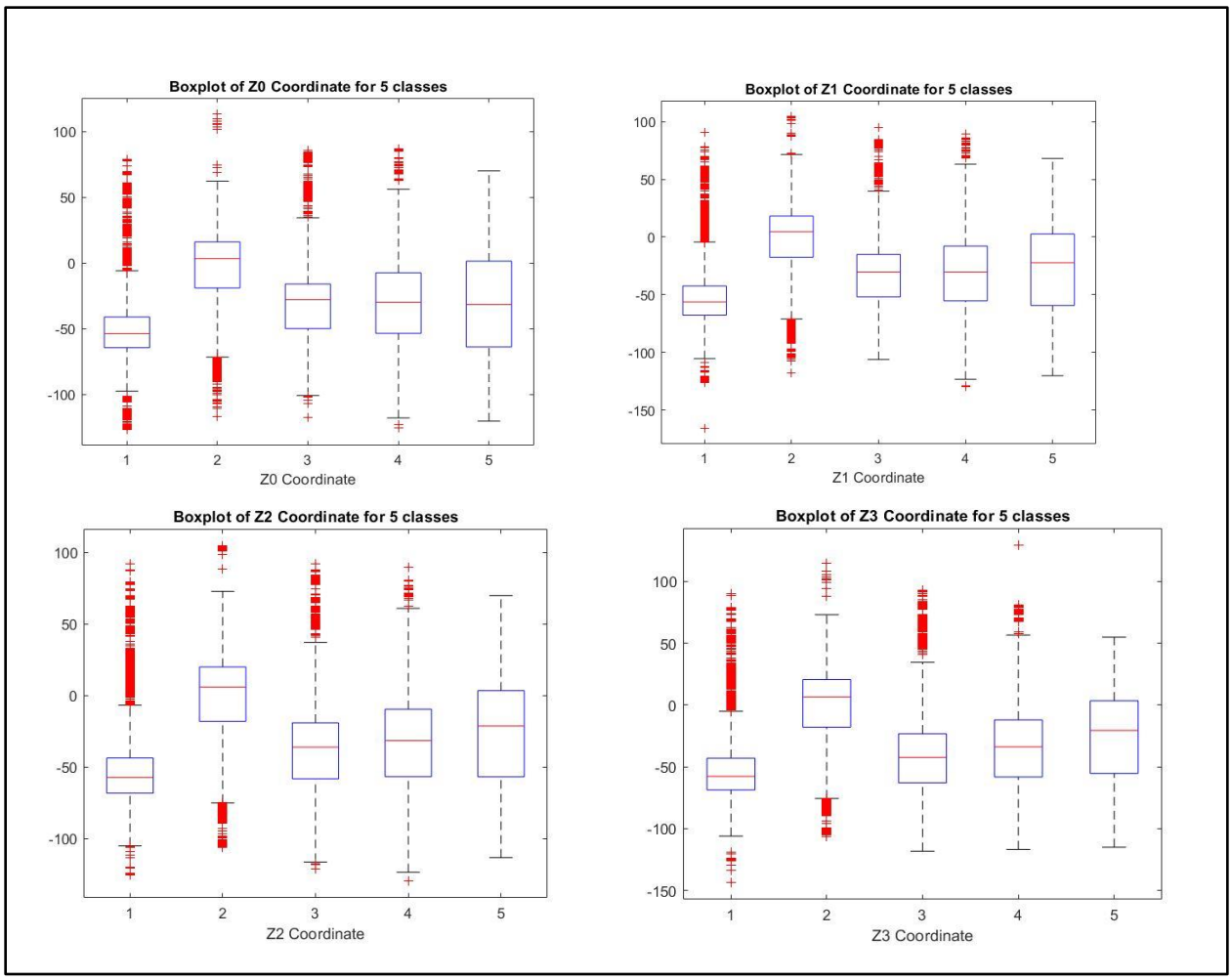


Fig 16: Box plots for Y Co-ordinates of 5 classes

Observations:

- From the histograms for all Y Coordinates, we see that there are outliers for every class except class 5. Some of which in class 1, class 2 and class 3 are quite noticeable.
- 50th percentile for each of the classes is rather scattered between 0 to -50 compared to X coordinates graph.
- Points shaded in red, in general will be termed as outliers. However, in our dataset they are not because a difference in the size of hand can cause such deviation. Since, we gather data from 12 different people, everyone has different sizes of hands.

3. Feature Selection

After data cleaning and visualizations of all our features with respect with all 5 classes, we can determine if it possible to decrease the dimensionality of out feature matrix to save computational time and cost, as well to reduce model complexity. We follow a procedure called principal component analysis to find out which feature variables are more important to their counterparts.

3.1 Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of distinct principal components is equal to the smaller of the number of original variables or the number of observations minus one. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score). Below are the scores and chart for PCA procedure performed on our dataset.

```
ans =

   1.0e+03 *

    4.2720
    2.2357
    2.0179
    1.8746
    1.4205
    0.8284
    0.7739
    0.7692
    0.7286
    0.3963
    0.3454
    0.3294
```

Features	PCA Score	% of total PCA	Relative %
1	4.272	26.71352372	26.71352372
2	2.2357	13.98020248	40.6937262
3	2.0179	12.618263	53.31198919
4	1.8746	11.72218436	65.03417355
5	1.4205	8.882621827	73.91679538
6	0.8284	5.180122437	79.09691781
7	0.7739	4.839324908	83.93624272
8	0.7692	4.80993503	88.74617775
9	0.7286	4.556056504	93.30223426
10	0.3963	2.478129553	95.78036381
11	0.3454	2.159843421	97.94020723
12	0.3294	2.05979277	100

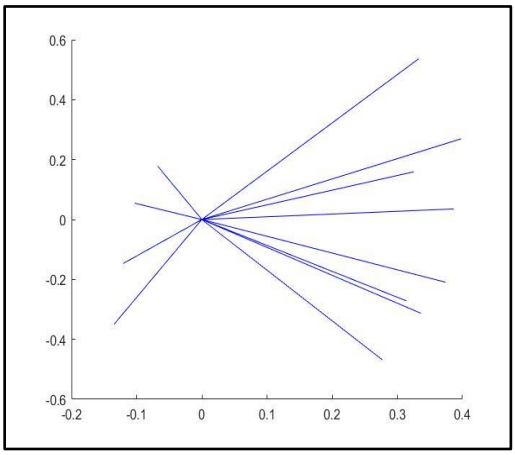


Fig 17: PCA Scores and chart for 12 feature Variables

3.1.1 Observations after performing PCA

- To avoid overfit in our models, we can eliminate features 10,11,12 from our model although it not necessary. Thus, the resulting feature matrix would have only 9 features. We can perform classification with and without PCA.

3.1.2 Results after working with pruned dataset from PCA

- On performing classification through with and without PCA, I was able to achieve better results without PCA. Hence, I have dropped my analysis of PCA for classification.

4. Classification Models

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, based on a training set of data containing observations (or instances) whose category membership is known. To ensure robust classification takes place, we use a technique called cross validation to train and test our data.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. In the models below, we use 5-fold cross validation in each of the models presented below.

4.1 Decision Tree Classifier

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

By using a decision tree classifier, we obtain an accuracy of 56.81% with class 2 and class 5 being highly misclassified.

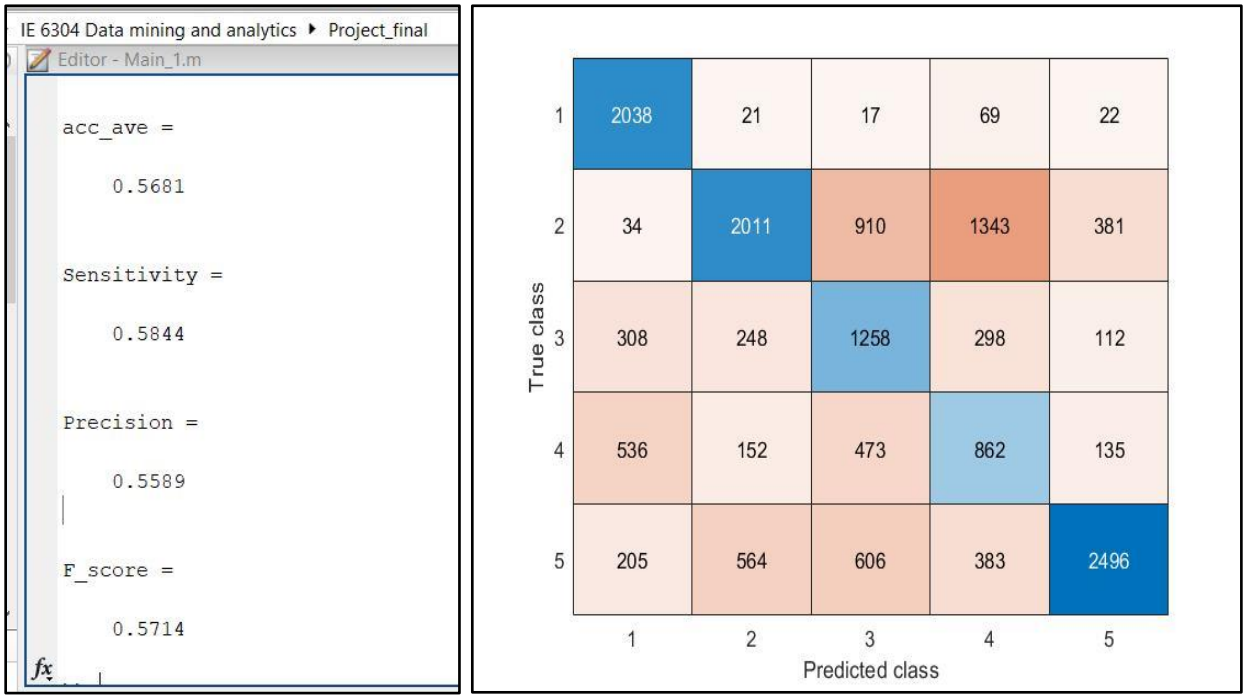


Fig 18: Results and Confusion Matrix for Decision Tree Classifier

4.2 Linear Discriminant Analysis Classifier

Linear Discriminant Analysis (LDA) is a classification method originally developed in 1936 by R. A. Fisher. It is simple, mathematically robust and often produces models whose accuracy is as good as more complex methods. LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made. The model uses Bayes Theorem to estimate the probabilities.

By using LDA classifier, we obtain an accuracy of 54.54% with class 2 and class 5 being highly misclassified. Class 2 is classified as other classes more than it is classified as class 2.

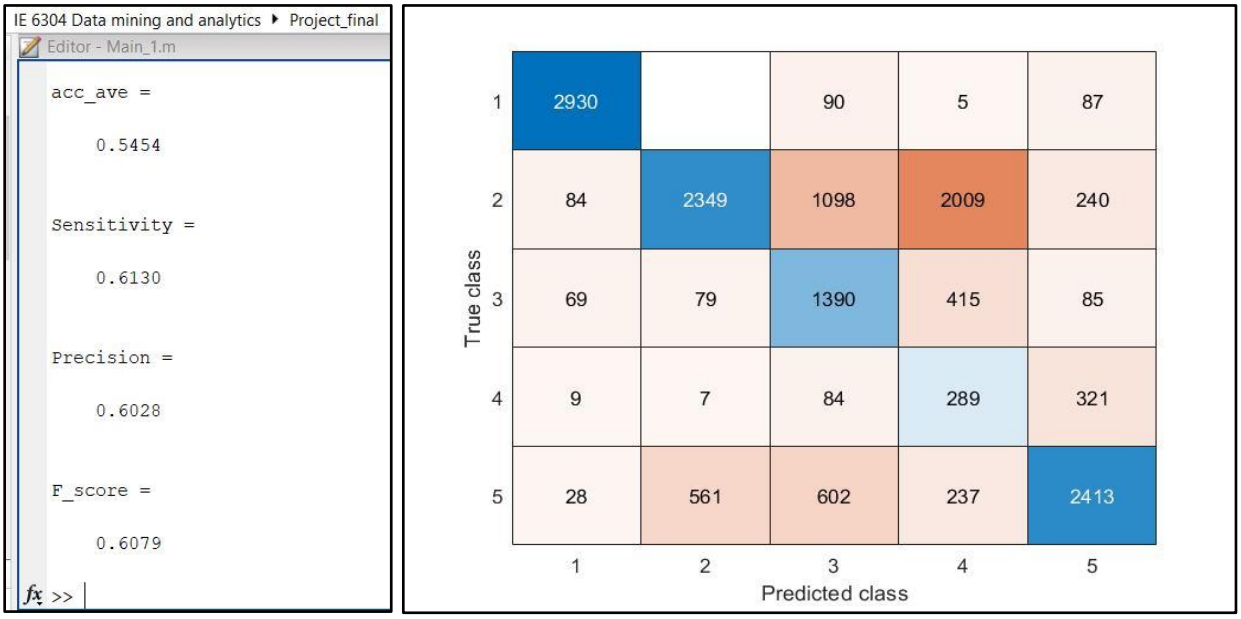


Fig 19: Results and Confusion Matrix for Linear Discriminant Analysis Classifier

4.3 Naïve Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes classifier calculates the probabilities for every factor. Then it selects the outcome with highest probability. This classifier assumes the features are independent.

By using Naïve Bayes classifier, we obtain an accuracy of 58.01% with class 2 and class 4 being highly misclassified. Class 2 is classified as other classes more than it is classified as class 2.

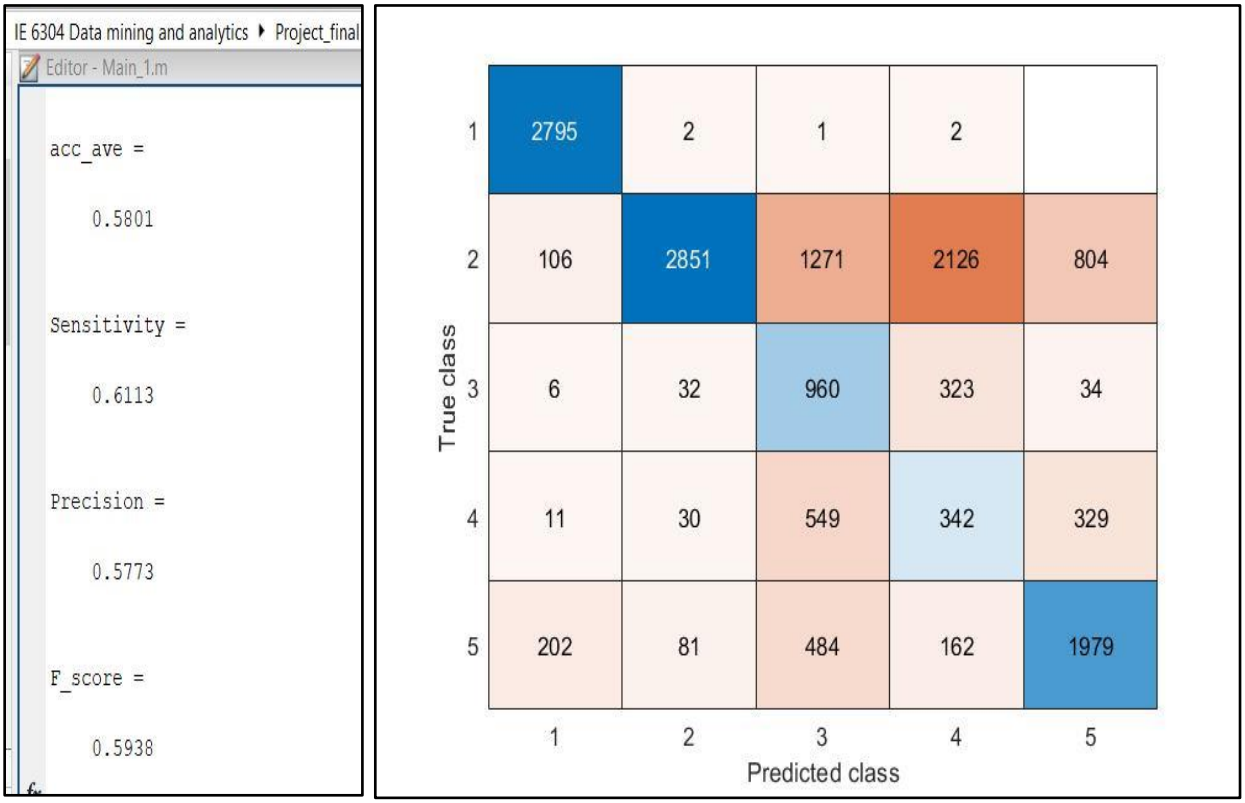


Fig 20: Results and Confusion Matrix for Naïve Bayes Classifier

4.4 K- Nearest Neighbours Classifier

The K-nearest neighbors algorithm (KNN) is a non-parametric method used for classification and regression analysis. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership.

An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. Both for classification and regression, a useful technique can be to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

By using KNN classifier with 3 neighbors are our decision criteria, we obtain an accuracy of 69.09% with class 2 being highly misclassified.

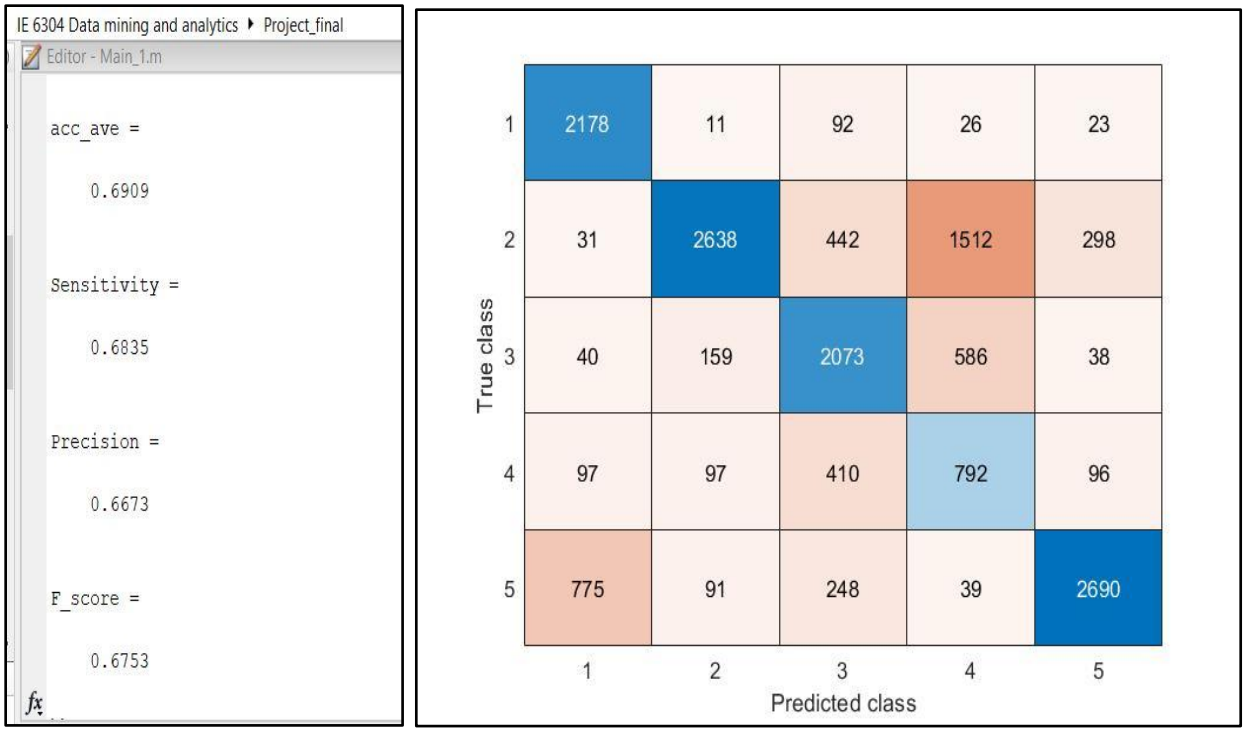


Fig 21: Results and Confusion Matrix for KNN-3 Classifier

By using KNN classifier with 3 neighbors are our decision criteria, we obtain an accuracy of 69.09%. To optimise our KNN we perform classification with an optimised KNN function by standardizing our feature matrix, using Euclidean distance with 3 neighbors each having equal weights. We increase our accuracy to 91.85%. Number of misclassifications for class two have decreased significantly.

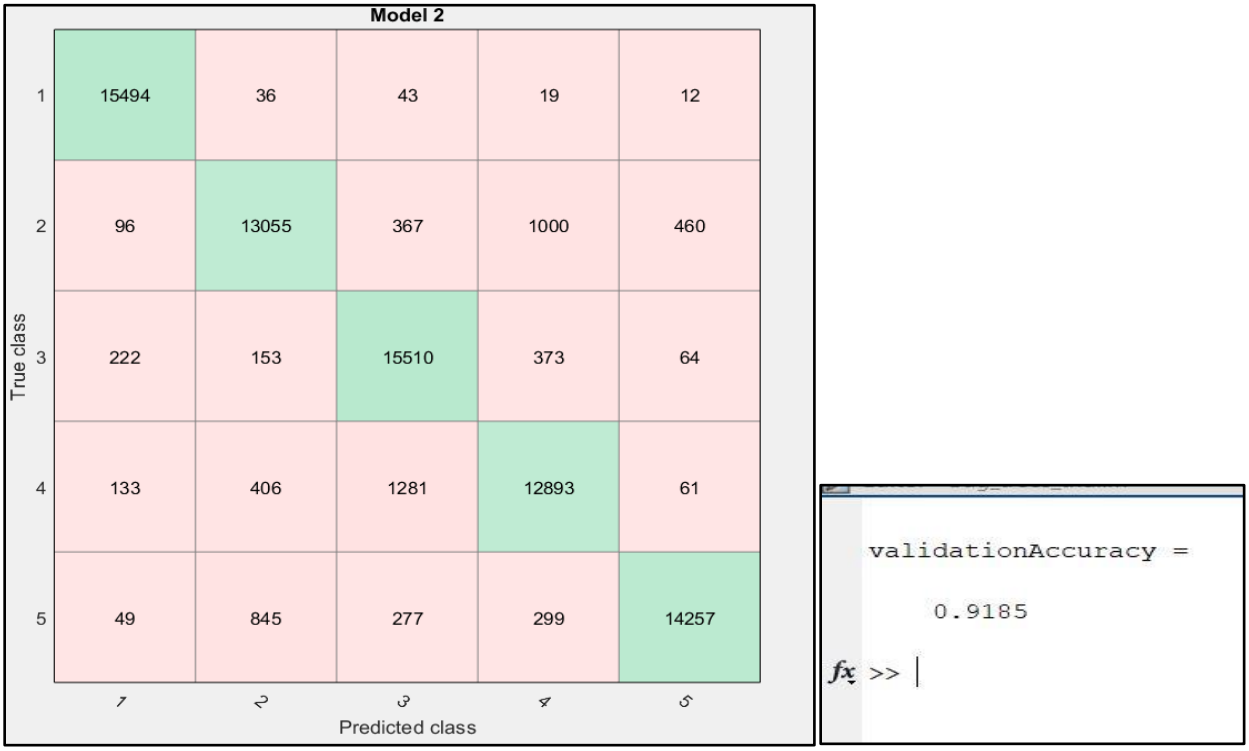


Fig 22: Results and Confusion Matrix for optimised KNN-3 Classifier

4.5 Bagging Ensemble – Bagged Trees Classifier (Bootstrap)

Bootstrap Aggregation is a general procedure that can be used to reduce the variance for those algorithms that have high variance. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees. When bagging with decision trees, we are less concerned about individual trees overfitting the training data. For this reason and for efficiency, the individual decision trees are grown deep (e.g. few training samples at each leaf-node of the tree) and the trees are not pruned. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

While, Random Forests are an improvement over bagged decision tree. It is a simple tweak. In CART, when selecting a split point, the learning algorithm is allowed to look through all variables and all variable values in order to select the most optimal split-point.

In our model, we use this bagging technique with decision tree as our base classifier. By using Bagging tress classifier, we obtain an accuracy of 97.51% with no single class being highly misclassified.

Fig 23: Results and Confusion Matrix for Bagged Tress Classifier

5. Performance Evaluation

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

- True positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.
- True negatives (TN): We predicted no, and they don't have the disease.
- False positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- False negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")
- Accuracy: Overall, how often is the classifier correct? (TP+TN)/total
- Sensitivity: When it's actually yes, how often does it predict yes? TP/actual yes
- Precision: When it predicts yes, how often is it correct? TP/predicted yes
- F Score: This is a weighted average of the true positive rate (Sensitivity) and precision.

The figure below shows the performance of our classification. KNN-3 has the highest parameters of all models and is thus superior to other models.

	Classifier			
Parameters	**Decision Trees**	**LDA**	**KNN - 3**	**Naïve Bayes**
acc_ave	0.5681	0.5454	**0.6909**	0.5801
Sensitivity	0.5844	0.613	**0.6835**	0.6114
Precision	0.5589	0.6028	**0.6673**	0.5773
F_score	0.5714	0.6079	**0.6753**	0.5939

Fig 24: Traditional Classifiers

	Classifier		
Parameters	**KNN - 3**	**Optimized KNN-3**	**Bagged Trees**
Validation Accuracy	0.6909	0.9185	**0.975**

Fig 25: Optimized and Ensemble Classifiers

6. Conclusion

- From the figures in model evaluation section, we observe that no model provides us an accuracy above 70% unless we use an optimised version of the classifier or a bagging approach.
- The bagging trees classifier has best accuracy among all model and is therefore the most reliable model to perform classification 5 hand postures as described earlier.
- Class 2 is noticeably the most highly misclassified class.

7. References

- A. Gardner, J. Kanno, C. A. Duncan, and R. Selmic. 'Measuring distance between unordered sets of different sizes,' in 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 137-143.
- A. Gardner, C. A. Duncan, J. Kanno, and R. Selmic. '3D hand posture recognition from small unlabeled point sets,' in 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC), Oct 2014, pp. 164-169
- Dataset was obtained https://archive.ics.uci.edu/ml/datasets/MoCap+Hand+Postures