Report

Hand Gestures classification

Data Mining 2022-2023 Q2

Authors:

Pol Casacuberta Gil

He Chen

Eric Hurtado

Tommaso Patriti

Alexandru-Ilie Popa

Facultat d'Informàtica de Barcelona

Index

1. Motivation and description of the problem **1**
2. Data source presentation **1**
3. Formal description of data structure and metadata **3**
4. Detailed description of preprocessing and data preparation (including this, here: Evaluation criteria of data mining models) **4**
4.1 Univariate description of the data 4
4.1.1 Missing data analysis 6
4.1.2 Boxplot and histograms for 12 markers 7
4.1.2 Univariate analysis depending on label 11
4.2 Bivariate analysis 26
4.2.1 Correlation Heat Map 26
4.2.2 Scatter Plots 28
4.2.3 Class Distribution 31
4.2.4 Marker Positions 32
4.2.5 Markers heat map 33
4.2.6 Pairwise scatter plots 34
4.3 Evaluation criteria of data mining models **36**
5. Machine learning methods **37**
5.1 Naïve Bayes 37
5.2 K-NN 37
5.2.1 Simple Cross-Validation 38
5.2.2 Evaluating the effect of K using 10-fold cross-validation 40
5.2.3 Accuracy score by K Value for Different Metrics 41
5.2.4 Accuracy score by K Value for Different Weights 42
5.2.5 Grid Search 43
5.3 Decision Trees 44
5.4 Support Vector Machines 53
5.5 Meta-learning algorithms 54
5.5.1 Bagging 54
5.5.2 RandomForest 55
5.5.3 Adaboost 56
5.5.4 Performance Majority Voting 56
6. Comparison **58**
7. General conclusions **60**
Task Completion **62**

1. Motivation and description of the problem

The motivation for choosing this topic is based on the interest in hand posture recognition and analysis using data captured by a motion capture camera system. This field has various applications in areas such as human-computer interaction, virtual reality, robotics, and healthcare, among others.

The ability to utilize marker information on the hand to identify and classify different postures opens the door to developing more intuitive and interactive systems. Additionally, the challenge of dealing with incomplete data and variability among the records presents an interesting problem from a data processing and machine learning algorithm development standpoint.

Exploring this particular database offers the opportunity to work with real data captured in a controlled environment. Furthermore, the availability of user information allows for evaluating the generalization of algorithms to new users, which is relevant for real-world applications.

In summary, the motivation behind choosing this topic lies in the interest in hand posture recognition, the challenge of dealing with incomplete data, and the potential for applying machine learning techniques in a practical and applied context.

Our objective is to develop predictive models that can determine the user who makes the movement based on the recorded movements captured by the Vicon motion capture camera system. By addressing the challenges posed by the dataset, we aim to contribute to advancements in various fields that require accurate hand posture prediction.

2. Data source presentation

Our data collection process began on the UCI Machine Learning Repository, a website known for its vast array of datasets catered towards various research interests. We strived to find a dataset that would engage the interests of all team members while also providing ample data for us to work effectively. After some exploration, we chose the MoCap Hand Posture dataset, which fulfilled all our requirements.

The MoCap Hand Posture dataset is a rich source of information about hand postures, gathered using a Vicon motion capture camera system. This system recorded twelve users performing five different hand postures with markers affixed to a left-handed glove. The markers were arranged in a rigid pattern on the back of the glove to establish a local coordinate system for the hand, and 11 other markers were attached to the thumb and fingers of the glove. This meticulous arrangement of markers provided a detailed representation of various hand postures.

However, the dataset came with its own set of complexities. The markers were unlabeled and not explicitly tracked, leading to inconsistencies between the markers of two given records. Additionally, self-occlusion due to the hand and fingers' orientation, resolution limitations of the capture volume, and software artifacts contributed to missing markers and extraneous markers. Hence, the number of visible markers per record varied significantly.

To make the dataset more manageable, it had already undergone partial preprocessing. This included transforming all markers to the local coordinate system of the record they belonged to, pruning markers with a norm greater than 200 millimeters, and removing records with fewer than 3 markers. As a result, the processed dataset had a maximum of 12 markers per record and at least 3.

The dataset was downloaded from the UCI Machine Learning Repository and loaded into our Google Colab environment for further analysis and manipulation. Using Google Colab's tools, we prepared the data to match the structure needed for our assignment.

This dataset is particularly fascinating as it gathers data about hand postures and their corresponding classifications. The dataset provides a 'Class' attribute, which signifies the class ID of the given record, ranging from 1 to 5, each denoting a specific hand posture. The 'User' attribute indicates the user that contributed the record. The dataset also provides the x, y, and z coordinates of the i-th unlabeled marker position.

In the following sections, we will delve into the different features of this dataset and discuss the nuances in further detail. The dataset’s acquisition source, the UCI Machine Learning Repository, also provides additional information about the data collection process.

3. Formal description of data structure and metadata

The MoCap Hand Postures dataset is a collection of motion capture data on 5 hand postures performed by 12 different users. The data was captured using a Vicon motion capture camera system and includes markers attached to a left-handed glove and fingers. The data is partially preprocessed, with all markers transformed to the local coordinate system of the record containing them and any transformed marker with a norm greater than 200 millimetres pruned. Records with fewer than 3 markers were also removed.

Data structure:

- Each record represents a single instant or frame as recorded by the camera system.
- There are 12 markers per record at most, and at least 3.
- The markers are not labelled, and their positions are not explicitly tracked.
- The 'Class' attribute indicates the class ID of the record, which 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).
- The 'User' attribute indicates the ID of the user that contributed the record.

Metadata:

- 'Class' - Integer. The class ID of the given record. Range from 1 to 5.
- 'User' - Integer. The ID of the user that contributed the record. Range from 0 to 14.
- 'Xi' - Real. The x-coordinate of the i-th unlabelled marker position. 'i' ranges from 0 to 11.
- 'Yi' - Real. The y-coordinate of the i-th unlabelled marker position. 'i' ranges from 0 to 11.
- 'Zi' - Real. The z-coordinate of the i-th unlabelled marker position. 'i' ranges from 0 to 11.
- Missing values are indicated by a question mark '?'.
- The data has been partially preprocessed, with all markers transformed to the local coordinate system of the record containing them, and any record that contained fewer than 3 markers removed. Additionally, any transformed marker with a norm greater than 200 millimeters was pruned.
- Due to the manner in which the data was captured, there may be near-duplicate records originating from the same user, so it is recommended to evaluate classification algorithms on a leave-one-user-out basis.

Example of a sample:

- Class: 1
- User: 0
- X0: 54.2638799540698
- Y0: 71.4667761378817
- Z0: -64.807708780882
- X1: 76.8956347751343
- …
- Z4: -71.2977813147725
- X5: ?
- Y5: ?
- Z5: ?
- …
- Z11: ?

4. Detailed description of preprocessing and data preparation (including this, here: Evaluation criteria of data mining models)

4.1 Univariate description of the data

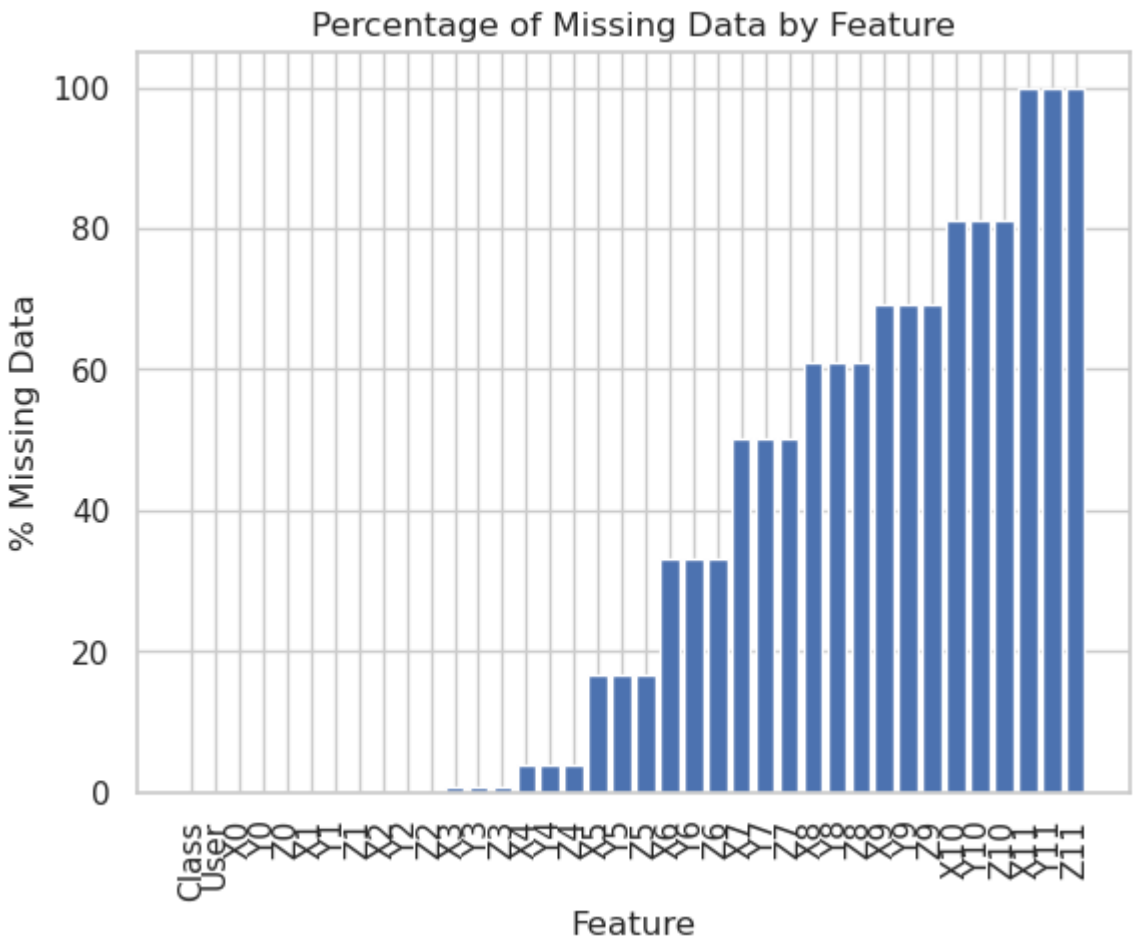
In the following table are some interesting values for uni-variate description.

	count	mean	std	min	25%	50%	75%	max	Missing data (%)
Class	**78096**	**2.984**	**1.421**	**0**	**2**	**3**	**4**	**5**	**0**

User	78096	7.959	4.698	0	5	9	12	14	0
X0	78096	50.346	32.696	-108.553	29.295	54.62	72.489	190.018	0
Y0	78096	85.812	40.204	-98.234	63.494	86.526	113.107	169.175	0
Z0	78096	-29.985	34.362	-126.771	-56.356	-30.864	-1.419	113.345	0
X1	78096	49.595	32.478	-111.685	28.755	54.216	71.762	188.692	0
Y1	78096	86.193	40.453	-96.143	64.155	87.543	116.219	170.209	0
Z1	78096	-29.509	34.764	-166.007	-57.36	-30.184	-0.367	104.698	0
X2	78096	48.612	33.605	-106.887	25.17	53.815	71.562	188.76	0
Y2	78096	83.771	41.024	-100.789	58.052	86.458	106.661	168.186	0
Z2	78096	-30.561	35.12	-129.595	-58.654	-32.352	-0.945	104.591	0
X3	77406	48.492	33.873	-111.761	23.86	54.136	71.435	151.033	0.884
Y3	77406	82.063	41.572	-97.603	54.072	85.75	105.531	168.292	0.884
Z3	77406	-31.146	35.933	-143.541	-59.37	-34.014	-1.439	129.317	0.884
X4	74976	48.405	34.227	-99.108	22.827	53.857	71.968	172.276	3.995
Y4	74976	80.418	42.624	-97.949	48.957	85.777	105.483	168.259	3.995
Z4	74976	-31.993	36.398	-157.199	-60.547	-35.165	-2.142	119.237	3.995
X5	65073	47.042	34.815	-120.658	19.535	51.951	71.889	180.563	16.676
Y5	65073	81.394	42.865	-97.469	49.252	88.136	107.835	167.926	16.676
Z5	65073	-30.27	36.853	-135.699	-58.994	-31.714	0.117	110.899	16.676
X6	52248	45.677	36.342	-100.084	15.485	52.085	72.615	176.409	33.098
Y6	52248	83.731	43.131	-67.284	51.655	92.154	111.472	168.598	33.098
Z6	52248	-26.639	35.71	-153.45	-55.375	-26.456	2.433	117.915	33.098
X7	38944	44.462	38.379	-108.606	13.078	49.601	75.72	189.222	50.133
Y7	38944	88.461	40.505	-64.972	63.541	93.796	119.186	169.127	50.133
Z7	38944	-20.368	33.944	-113.733	-45.676	-19.425	6.591	117.816	50.133
X8	30564	48.208	38.605	-121.182	17.039	55.946	79.631	173.907	60.864
Y8	30564	86.028	41.616	-65.078	53.661	92.183	121.911	169.323	60.864
Z8	30564	-24.364	35.736	-142.654	-52.521	-22.458	6.415	119.213	60.864
X9	23968	54.747	39.971	-99.232	26.114	61.992	83.767	174.054	69.31

Y9	**23968**	**80.542**	**42.676**	**-64.734**	**44.034**	**84.739**	**115.555**	**167.943**	**69.31**
Z9	**23968**	**-27.777**	**36.251**	**-113.397**	**-55.618**	**-26.797**	**3.064**	**123.381**	**69.31**
X10	**14753**	**53.755**	**38.461**	**-80.196**	**20.407**	**61.844**	**83.055**	**149.486**	**81.109**
Y10	**14753**	**73.999**	**42.915**	**-65.019**	**37.697**	**78.987**	**100.948**	**168.352**	**81.109**
Z10	**14753**	**-29.736**	**39.678**	**-112.669**	**-62.953**	**-32.909**	**6.06**	**108.456**	**81.109**
X11	**32**	**-28.77**	**49.754**	**-96.952**	**-62.609**	**-48.524**	**0.646**	**84.683**	**99.959**
Y11	**32**	**25.152**	**36.539**	**-65.432**	**21.795**	**38.748**	**40.703**	**127.945**	**99.959**
Z11	**32**	**1.644**	**19.656**	**-48.275**	**-3.829**	**11.206**	**12.789**	**18.062**	**99.959**

4.1.1 Missing data analysis



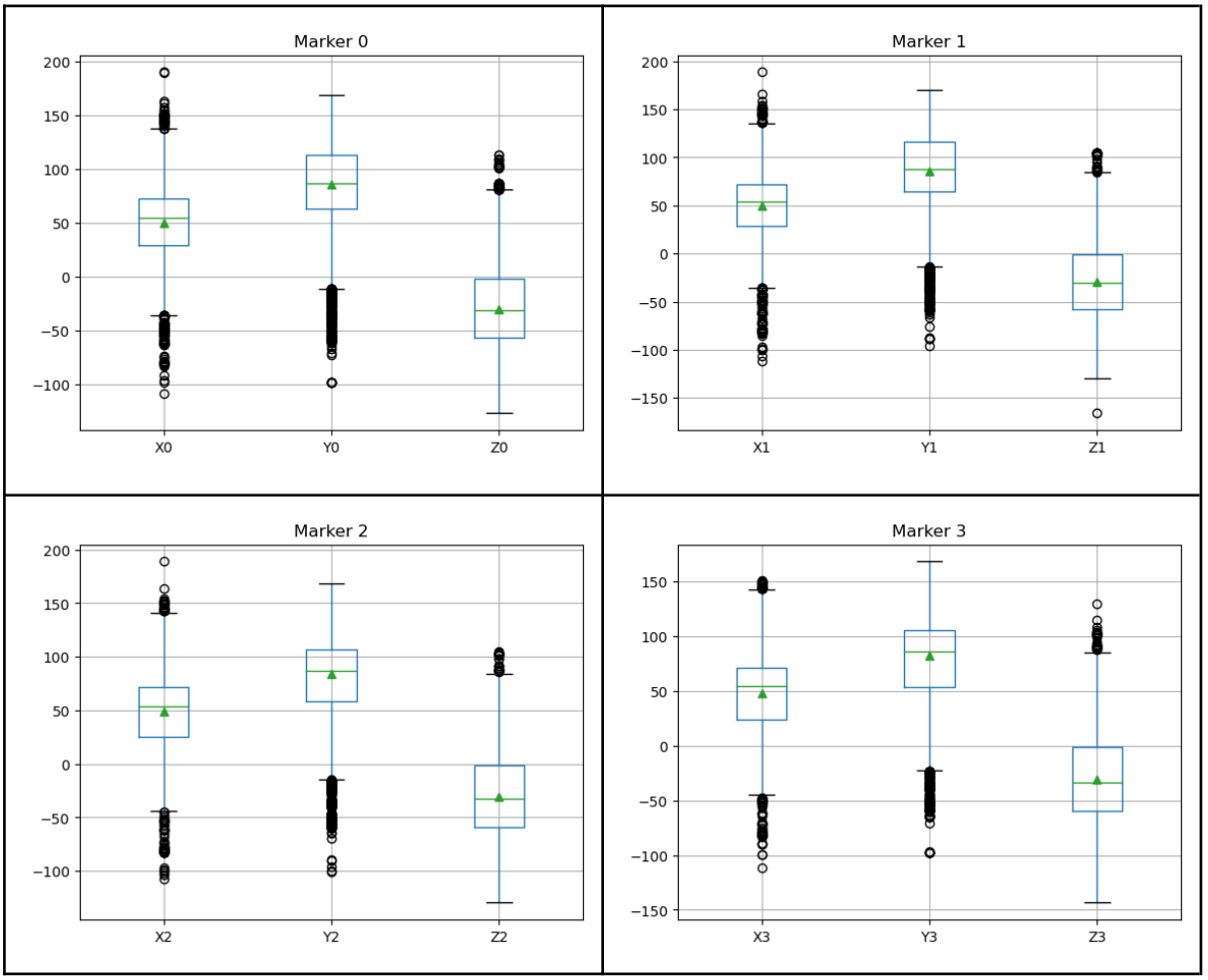
We can observe that the "Class" and "User" features do not have missing data. Also, for the features representing markers 0, 1, and 2, there is no missing data.

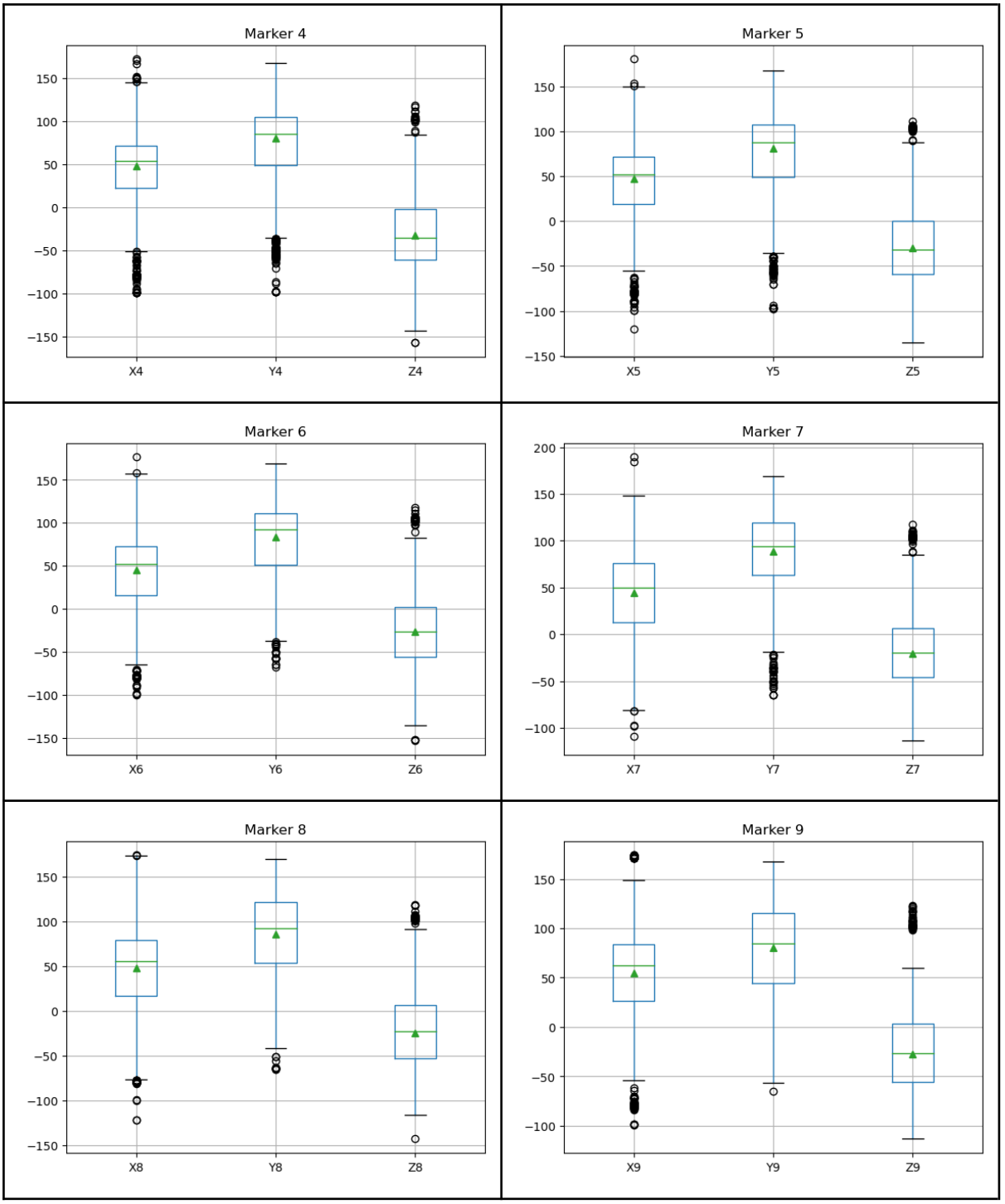
Starting from marker 3, we can see that the percentage of missing data increases as the marker number increases, until the last marker, 11, which has 99.959% missing data. This

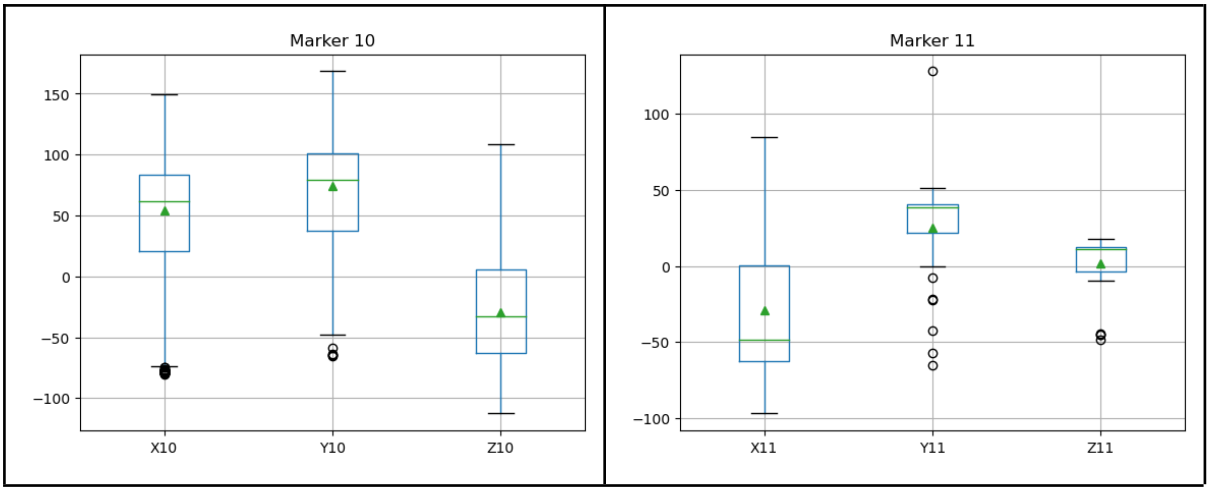
may be an indicator of a systematic issue or error in the data collection process. It is possible that some records have missing markers due to self-occlusion and artifacts in the recording process for some postures. Knowing that markers 0, 1, and 2 are located in the thumb, and markers 3 and 4 in the index finger... In this sense, these markers can represent "the importance of different fingers". For example, in some countries, cutting off someone's thumb is considered a seriously injured, which is severely punished, while cutting off the pinky finger has a lighter punishment.

4.1.2 Boxplot and histograms for 12 markers

In the following images, we have boxplots grouped by {Xi, Yi, Zi} for each marker:

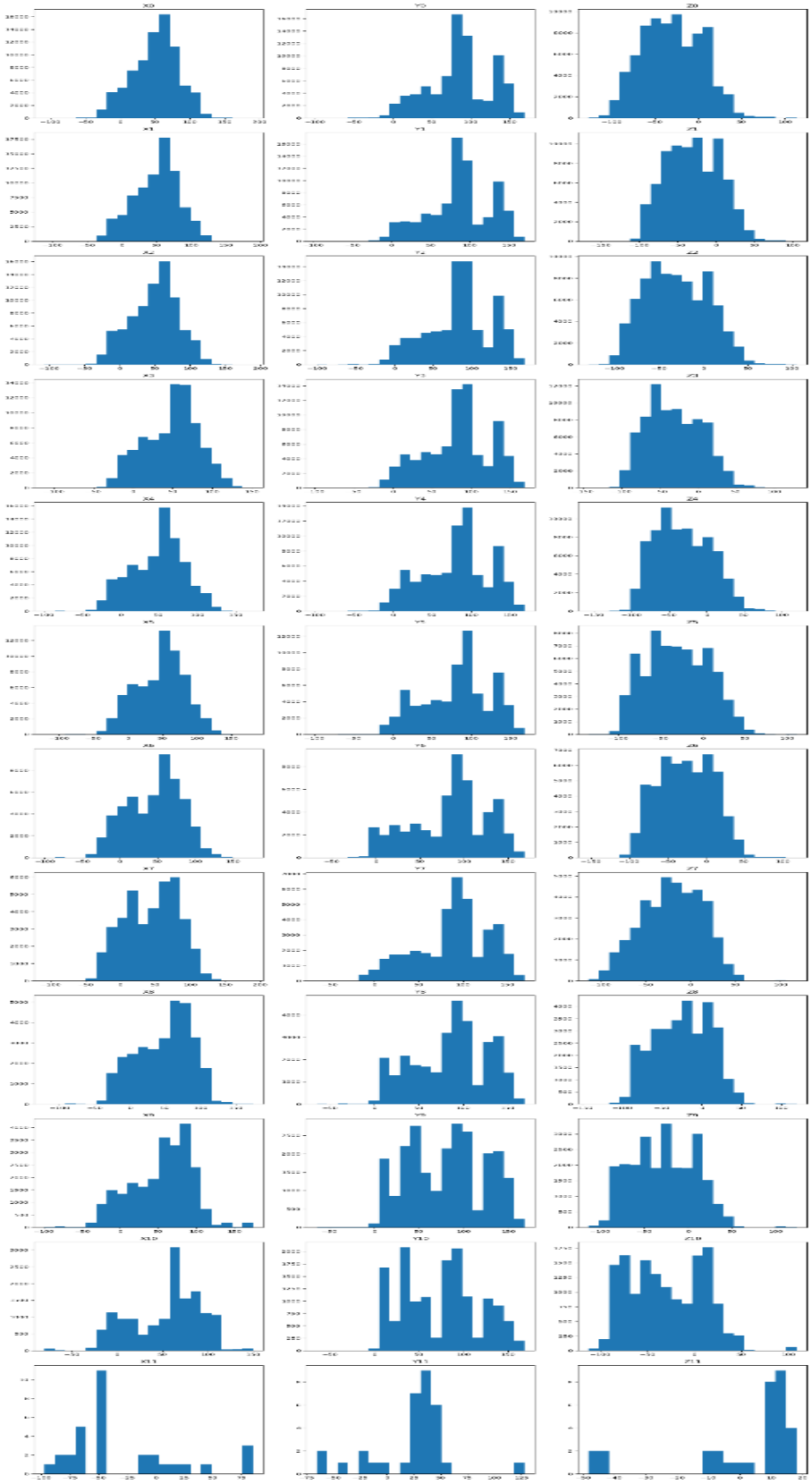



Marker 4
X4
Y4
Z4
Marker 5
X5
Y5
Z5
Marker 6
X6
Y6
Z6
Marker 7
X7
Y7
Z7
Marker 8
X8
Y8
Z8
Marker 9
X9
Y9
Z9



There are a few outliers in the data set for some variables, which are indicated by dots outside of the whiskers in the box plots. These outliers could be investigated further to determine if they are legitimate data points or if they represent errors or unusual events.
We can see that the mean and standard deviation do not vary much for markers with small percentage of missing data.

In the following images, we have histograms for each marker, where each row represents the features {Xi, Yi, Zi} for each marker:

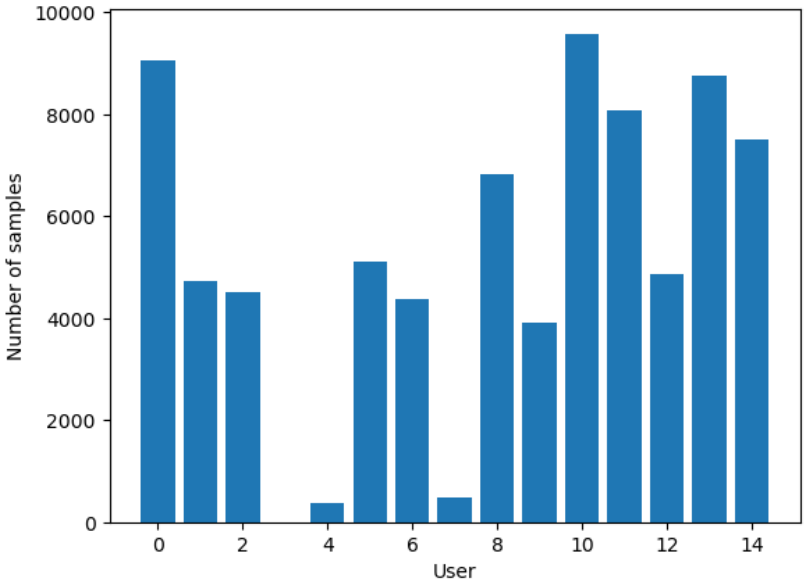


We can see that the distribution of the histograms is not known. For example, the histogram of the Xi column (the first one) resembles a normal distribution, but it is not symmetric, so it cannot be the normal distribution. Another important aspect is that the shape of the histograms for each column remains consistent for the first half of the markers. However, we

see a significant change for the last markers compared to the first ones. This occurs because, as mentioned before, the percentage of missing data increases as the number of markers increases. For instance, for the last marker, Marker 11, with 99.9% missing data, we can say that the histogram shape is completely random since we don't have enough data. Delving into more detail, for Marker 7 with 50% missing data, we already observe indications of variations in the shape, specifically in the X histogram, where we see two peaks. This factor may need to be considered when imputing the missing data.

4.1.2 Univariate analysis depending on label

Group by User

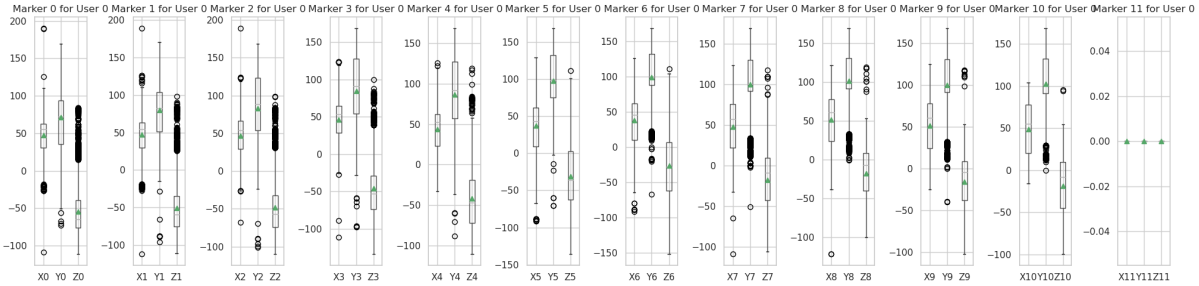


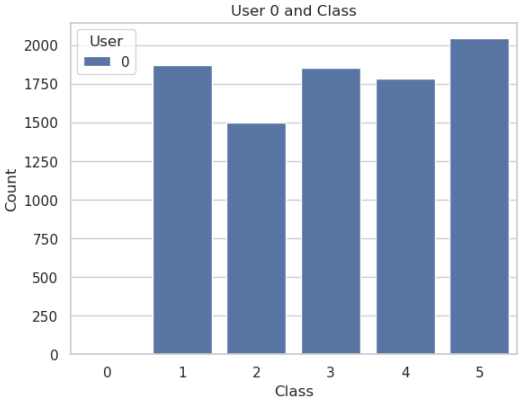
It appears that we have an imbalanced dataset. The number of instances in each class varies, with some classes having significantly more instances than others. Imbalanced datasets can pose challenges for machine learning algorithms, particularly when the minority class is underrepresented.

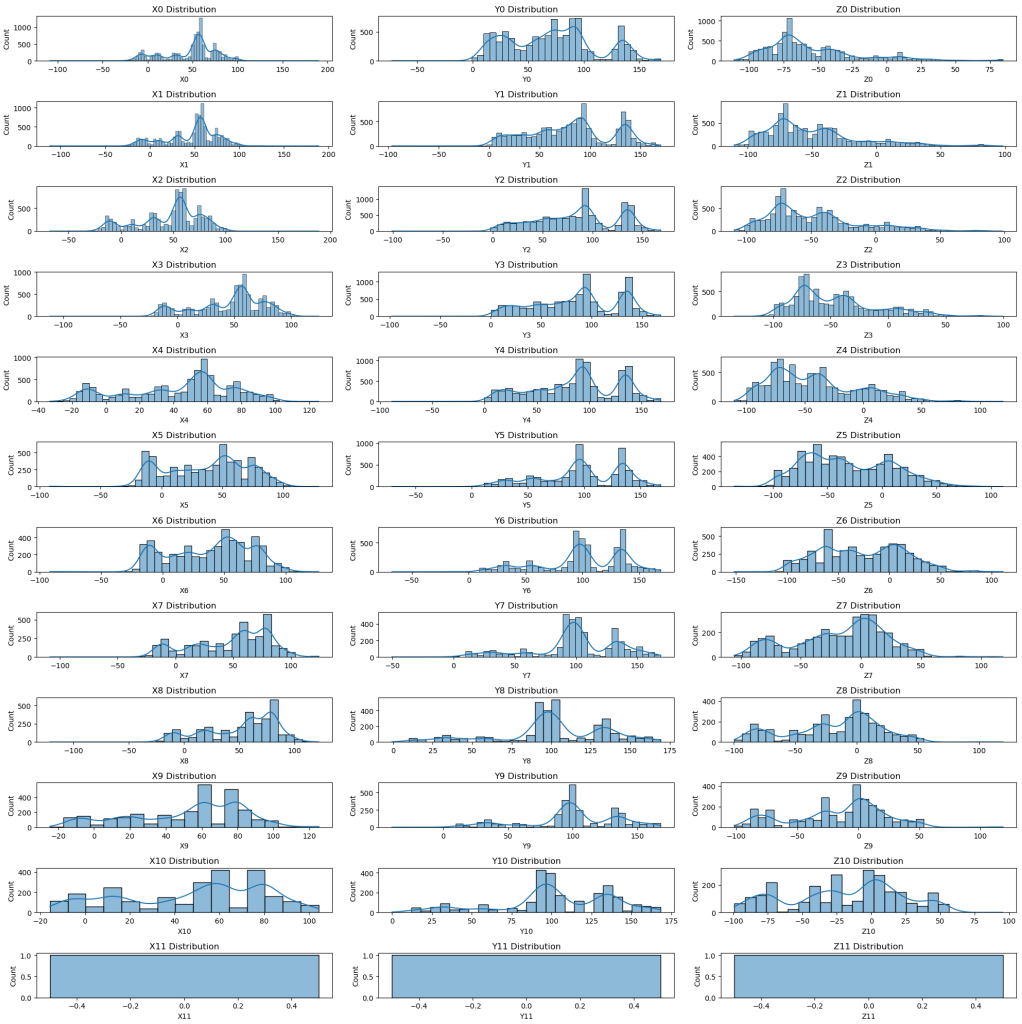
In our dataset, classes 3, 4 and 7 have relatively fewer instances compared to the others. This class imbalance can impact the model's ability to accurately learn patterns and make predictions, as the model may be biased towards the majority class.

Since we have a class imbalance where only a few classes have less than 500 samples, and the majority of classes have samples in the range of 4000 to 9000. When implementing the algorithm we will take care to evaluate model performance: Start by training and evaluating our model on the imbalanced dataset. Assess the performance metrics such as accuracy, precision, recall, and F1-score for each class.

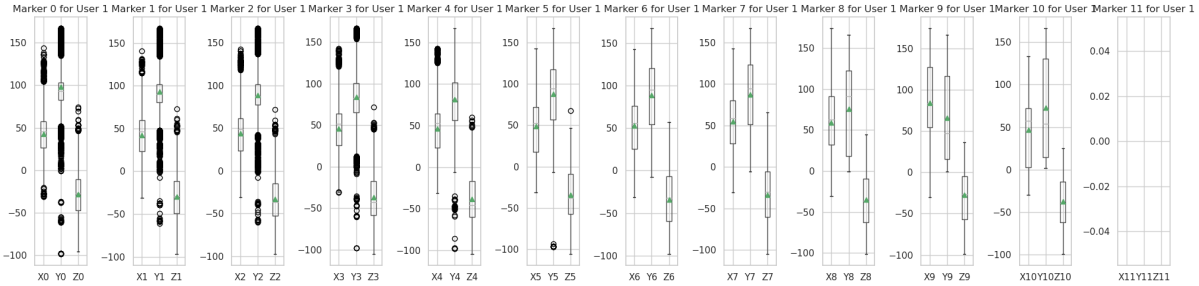
User 0:

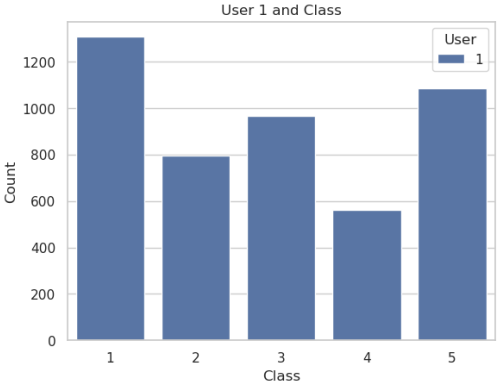

Marker 0 for User 0
Marker 1 for User 0
Marker 2 for User 0
Marker 3 for User 0
Marker 4 for User 0
Marker 5 for User 0
Marker 6 for User 0
Marker 7 for User 0
Marker 8 for User 0
Marker 9 for User 0
Marker 10 for User 0
Marker 11 for User 0
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

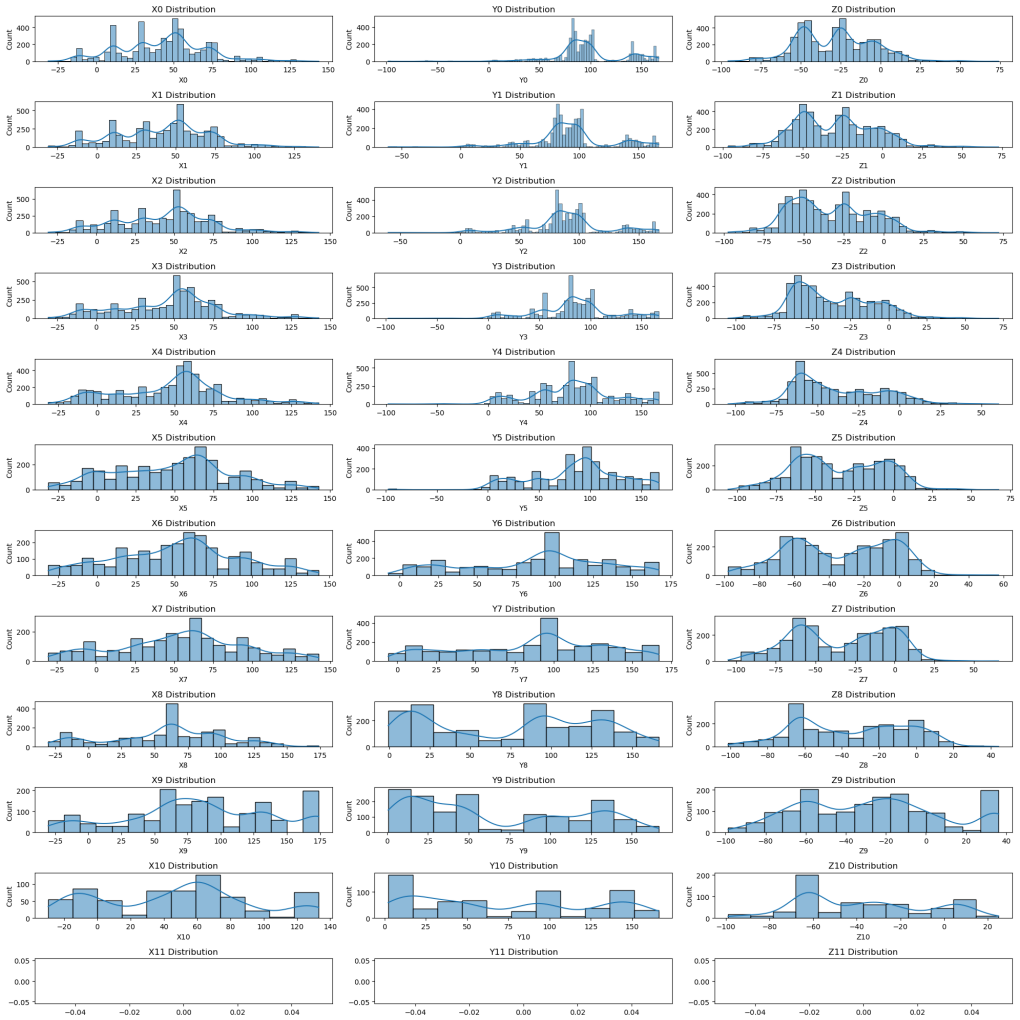

User 0 and Class
User
0
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

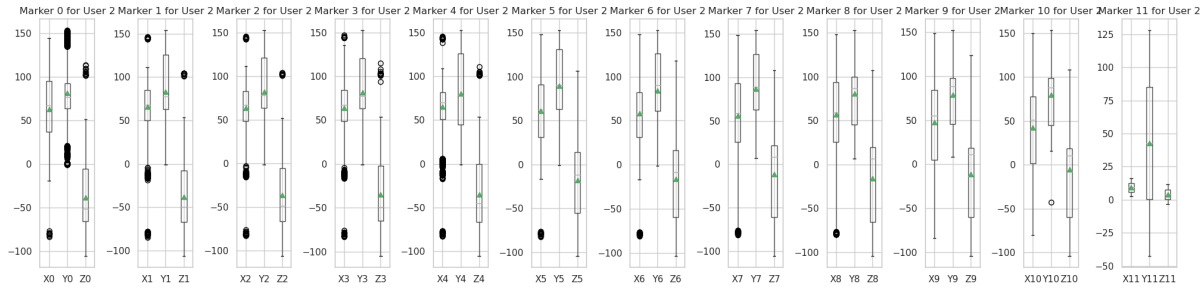
User 1:

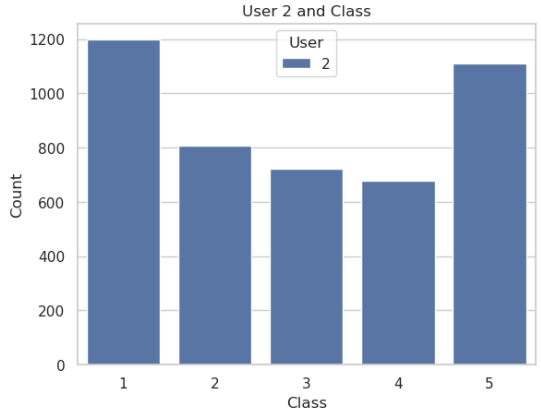

Marker 0 for User 1
Marker 1 for User 1
Marker 2 for User 1
Marker 3 for User 1
Marker 4 for User 1
Marker 5 for User 1
Marker 6 for User 1
Marker 7 for User 1
Marker 8 for User 1
Marker 9 for User 1
Marker 10 for User 1
Marker 11 for User 1
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

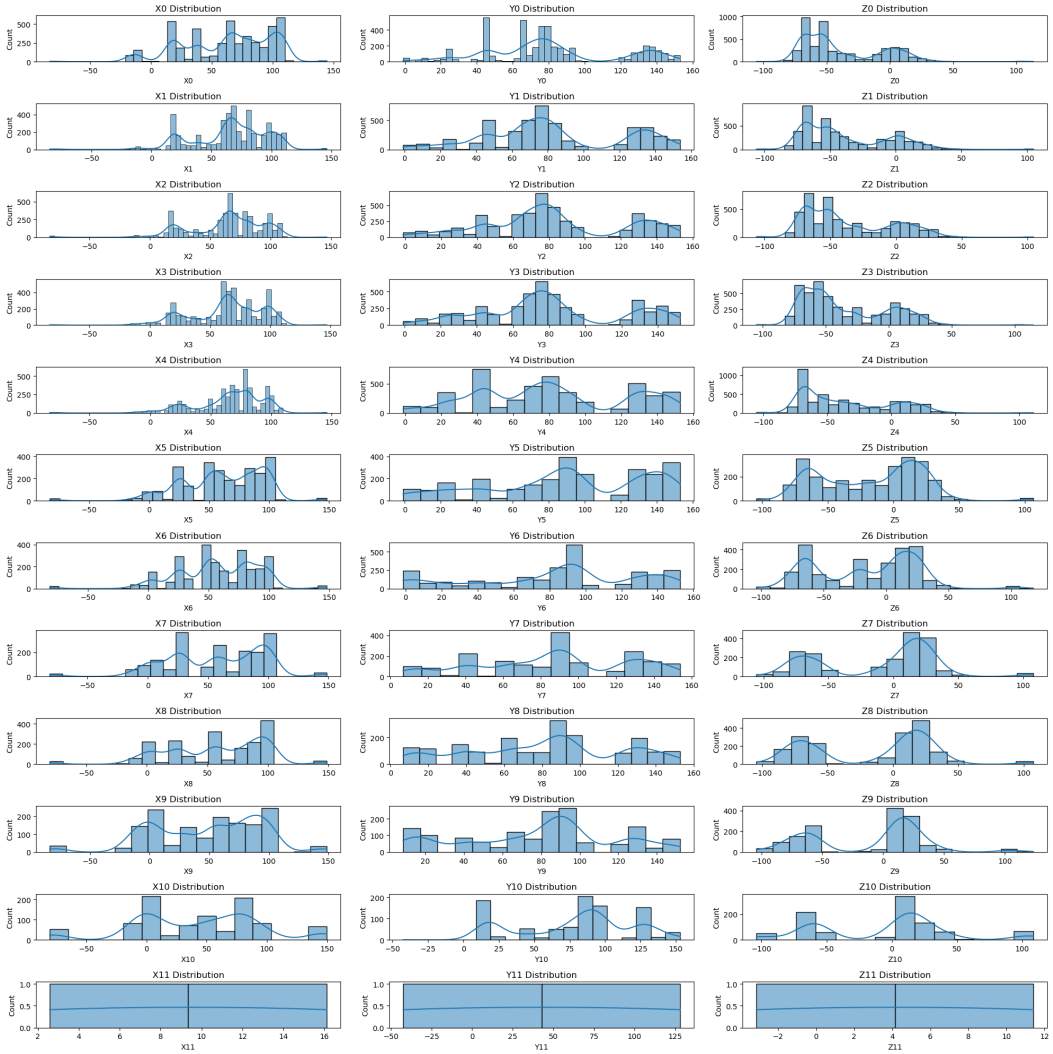

User 1 and Class
User
1
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution

User 2:

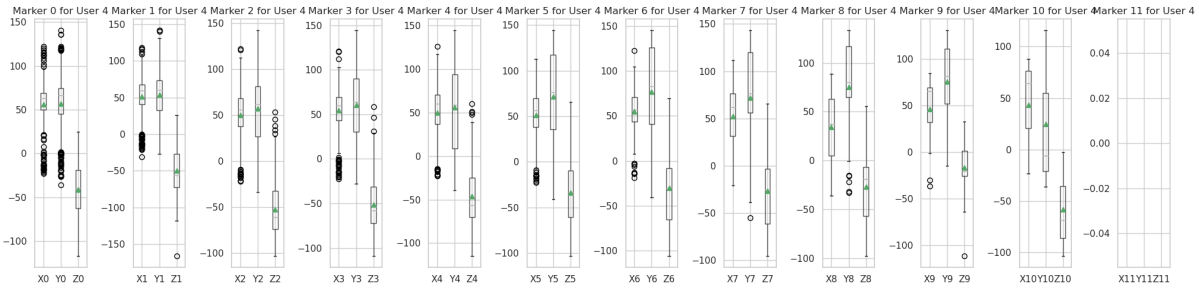


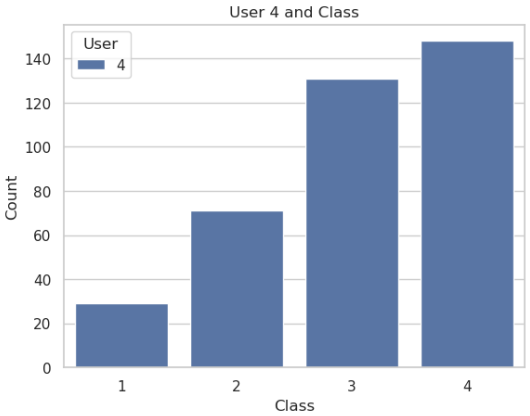


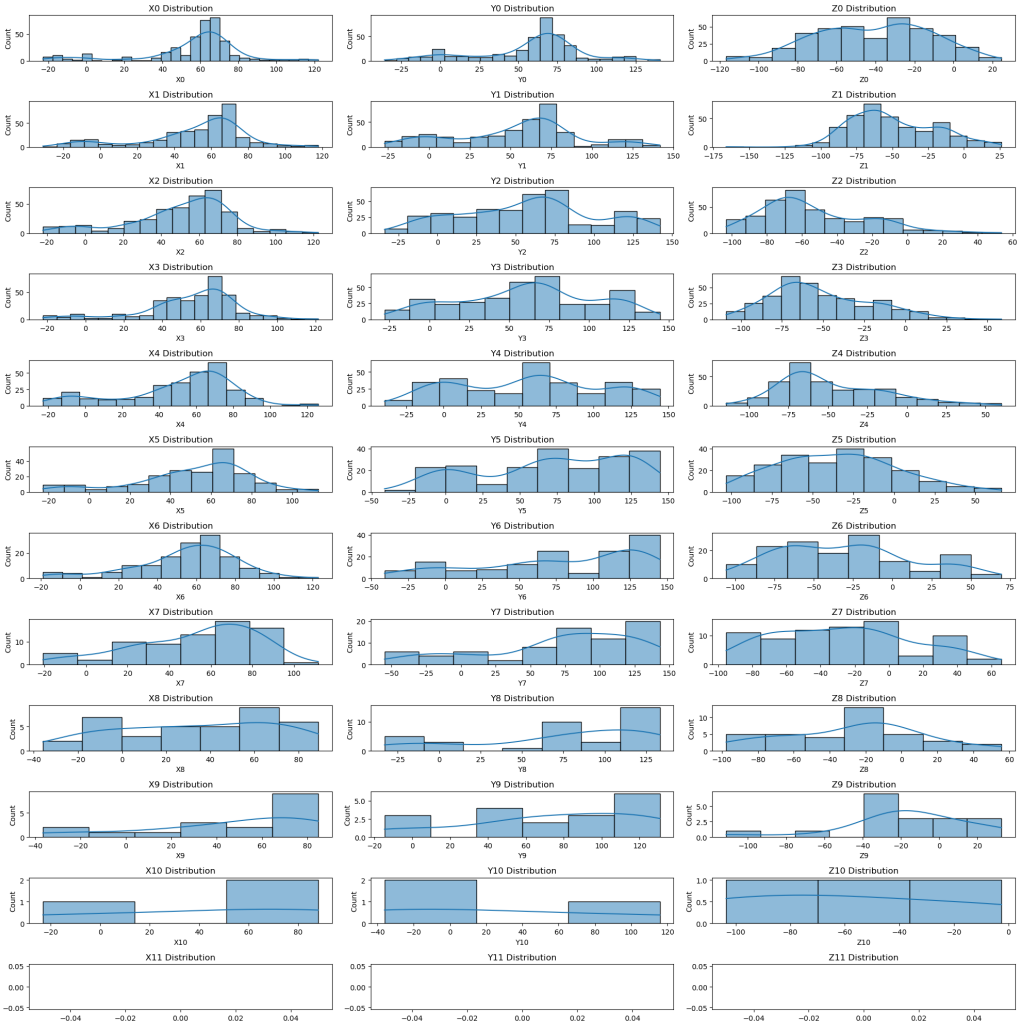


User 3: We don’t have User 3

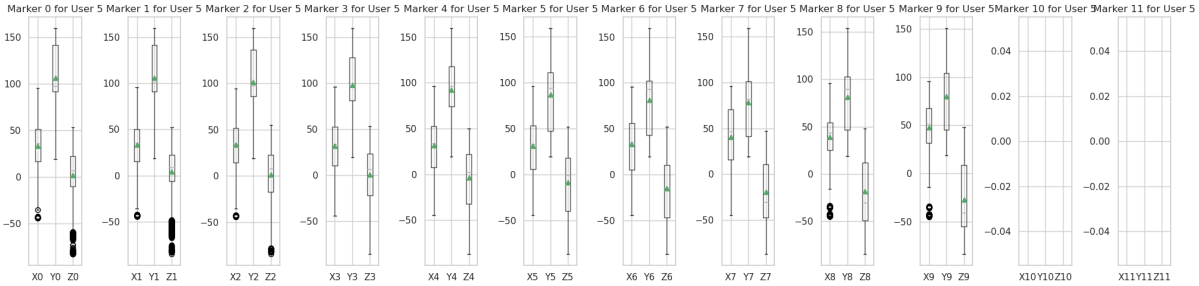
User 4:

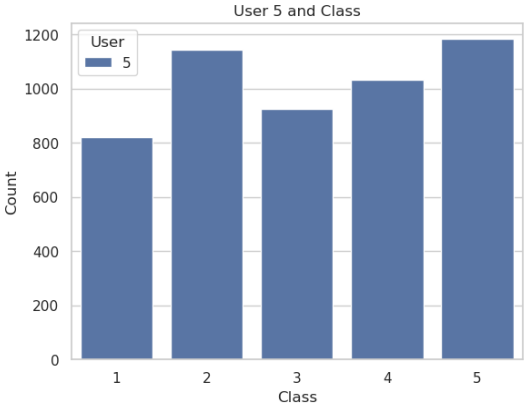

Marker 0 for User 4
Marker 1 for User 4
Marker 2 for User 4
Marker 3 for User 4
Marker 4 for User 4
Marker 5 for User 4
Marker 6 for User 4
Marker 7 for User 4
Marker 8 for User 4
Marker 9 for User 4
Marker 10 for User 4
Marker 11 for User 4
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

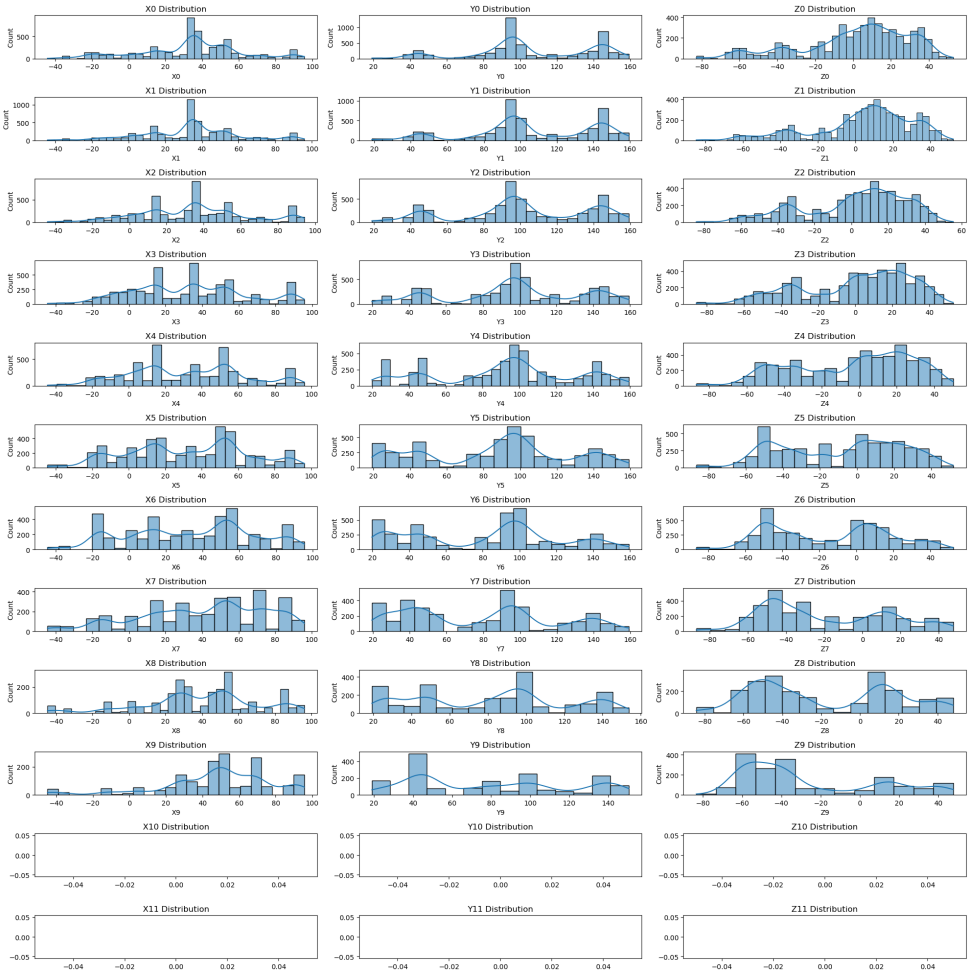

User 4 and Class
User
4
Count
Class
1
2
3
4


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution

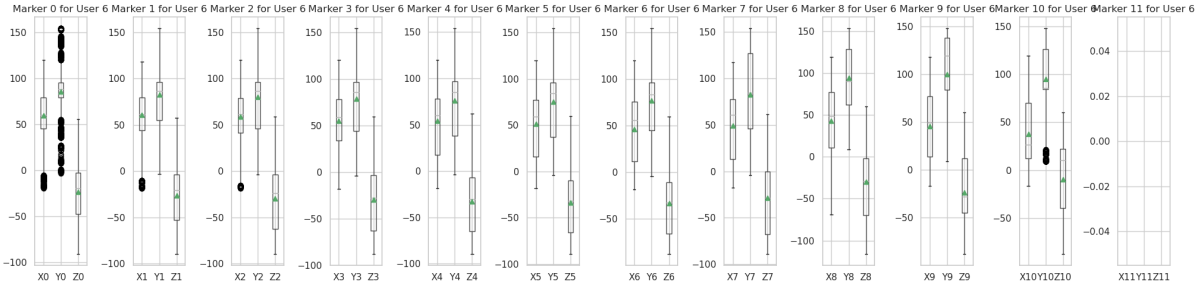
User 5:


Marker 0 for User 5
Marker 1 for User 5
Marker 2 for User 5
Marker 3 for User 5
Marker 4 for User 5
Marker 5 for User 5
Marker 6 for User 5
Marker 7 for User 5
Marker 8 for User 5
Marker 9 for User 5
Marker 10 for User 5
Marker 11 for User 5
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10 Y10 Z10
X11 Y11 Z11

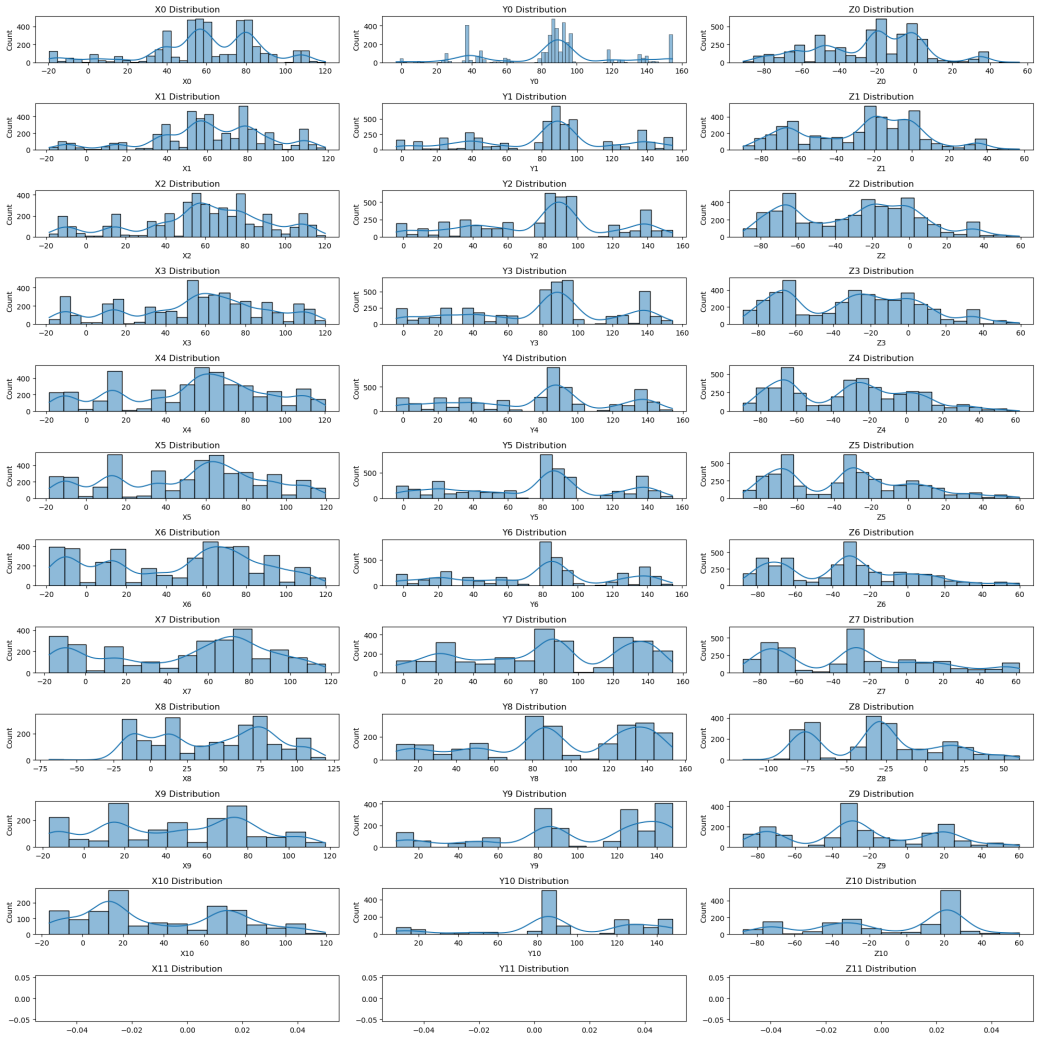

User 5 and Class
User
5
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

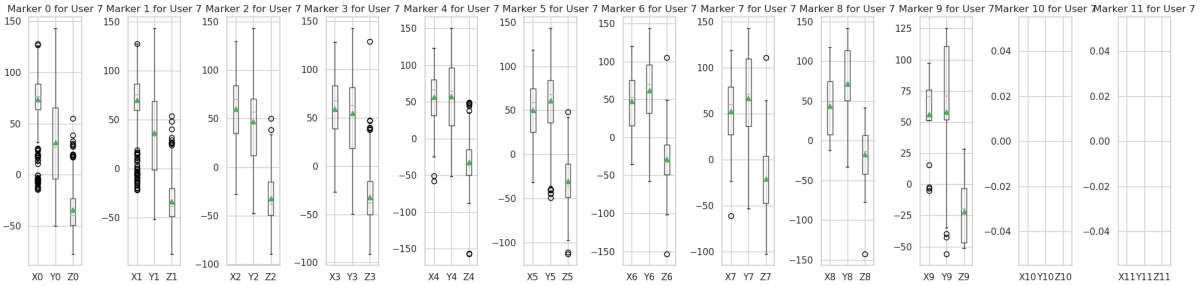
User 6:

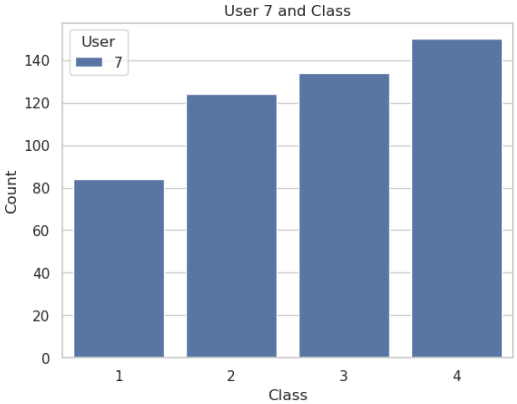

Marker 0 for User 6
Marker 1 for User 6
Marker 2 for User 6
Marker 3 for User 6
Marker 4 for User 6
Marker 5 for User 6
Marker 6 for User 6
Marker 7 for User 6
Marker 8 for User 6
Marker 9 for User 6
Marker 10 for User 6
Marker 11 for User 6
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

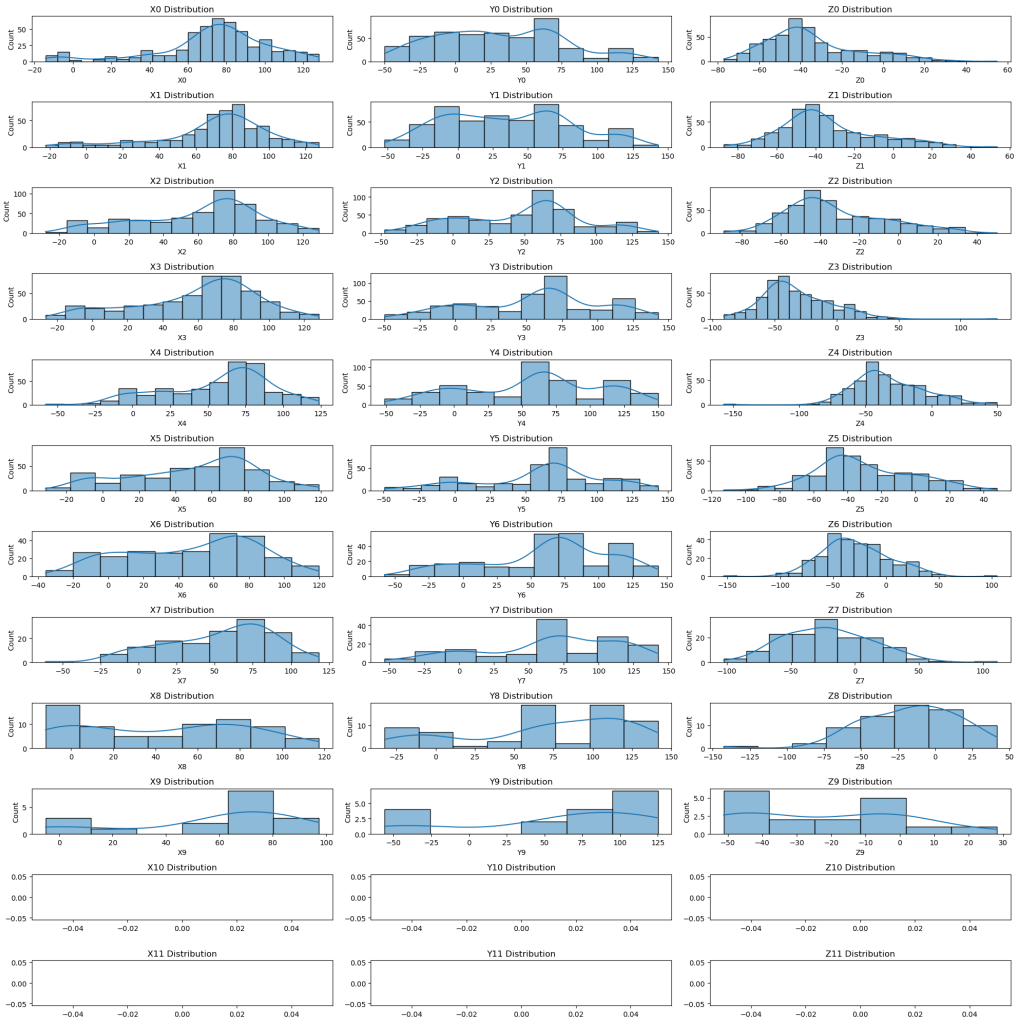
User 6 and Class
User
6
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution

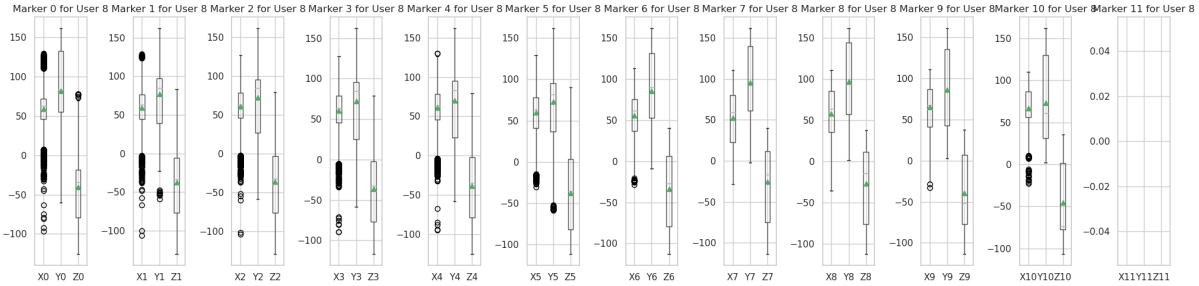
User 7:

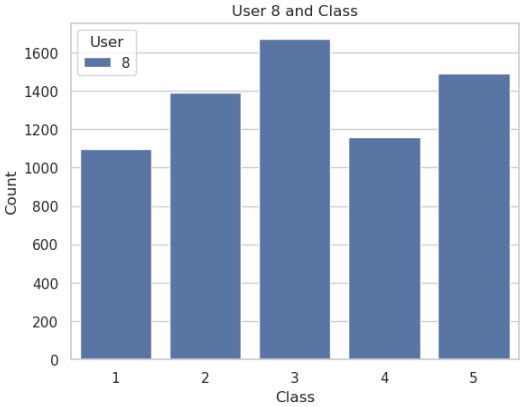

Marker 0 for User 7
Marker 1 for User 7
Marker 2 for User 7
Marker 3 for User 7
Marker 4 for User 7
Marker 5 for User 7
Marker 6 for User 7
Marker 7 for User 7
Marker 8 for User 7
Marker 9 for User 7
Marker 10 for User 7
Marker 11 for User 7
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10 Y10 Z10
X11 Y11 Z11

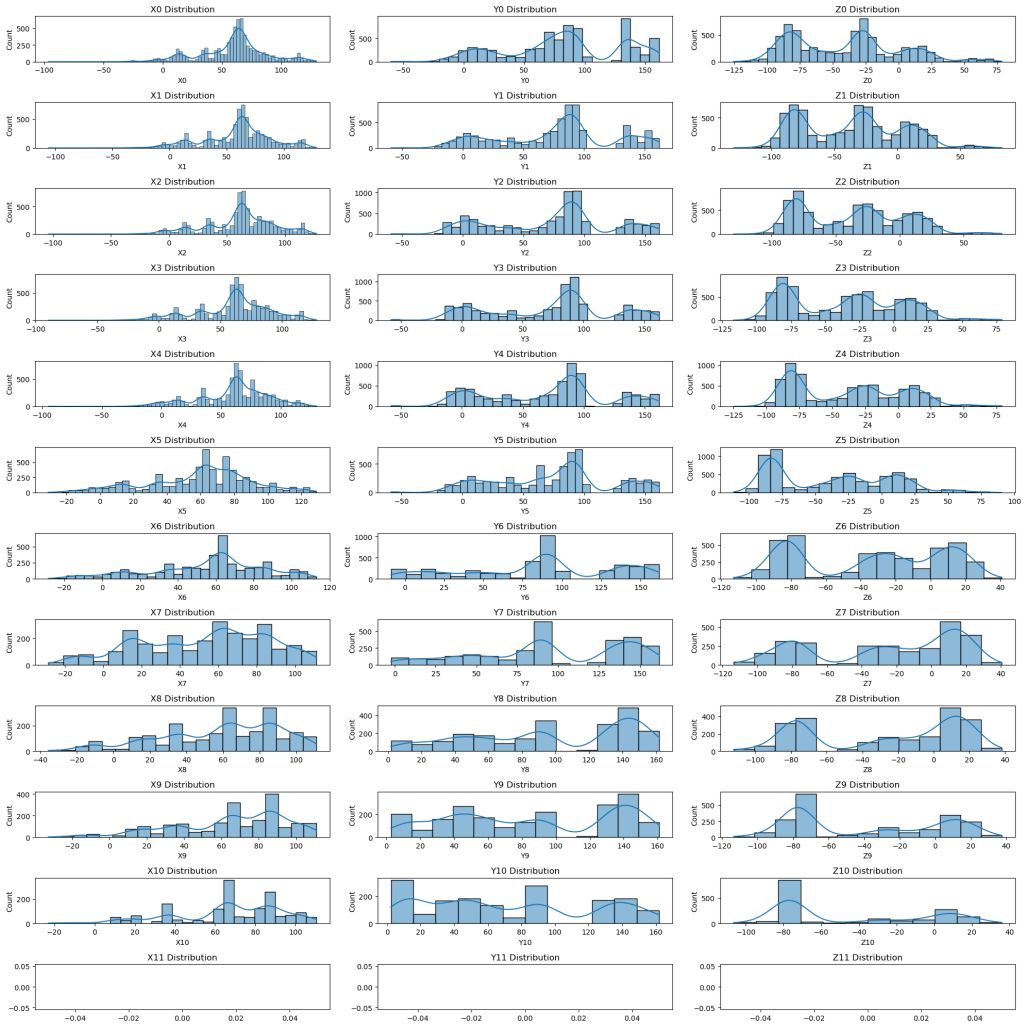

User 7 and Class
User
7
Count
Class
1
2
3
4


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

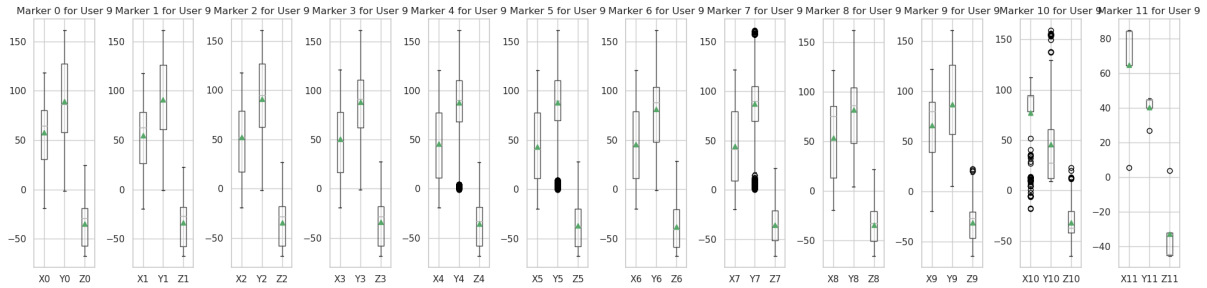
User 8:


Marker 0 for User 8
Marker 1 for User 8
Marker 2 for User 8
Marker 3 for User 8
Marker 4 for User 8
Marker 5 for User 8
Marker 6 for User 8
Marker 7 for User 8
Marker 8 for User 8
Marker 9 for User 8
Marker 10 for User 8
Marker 11 for User 8
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

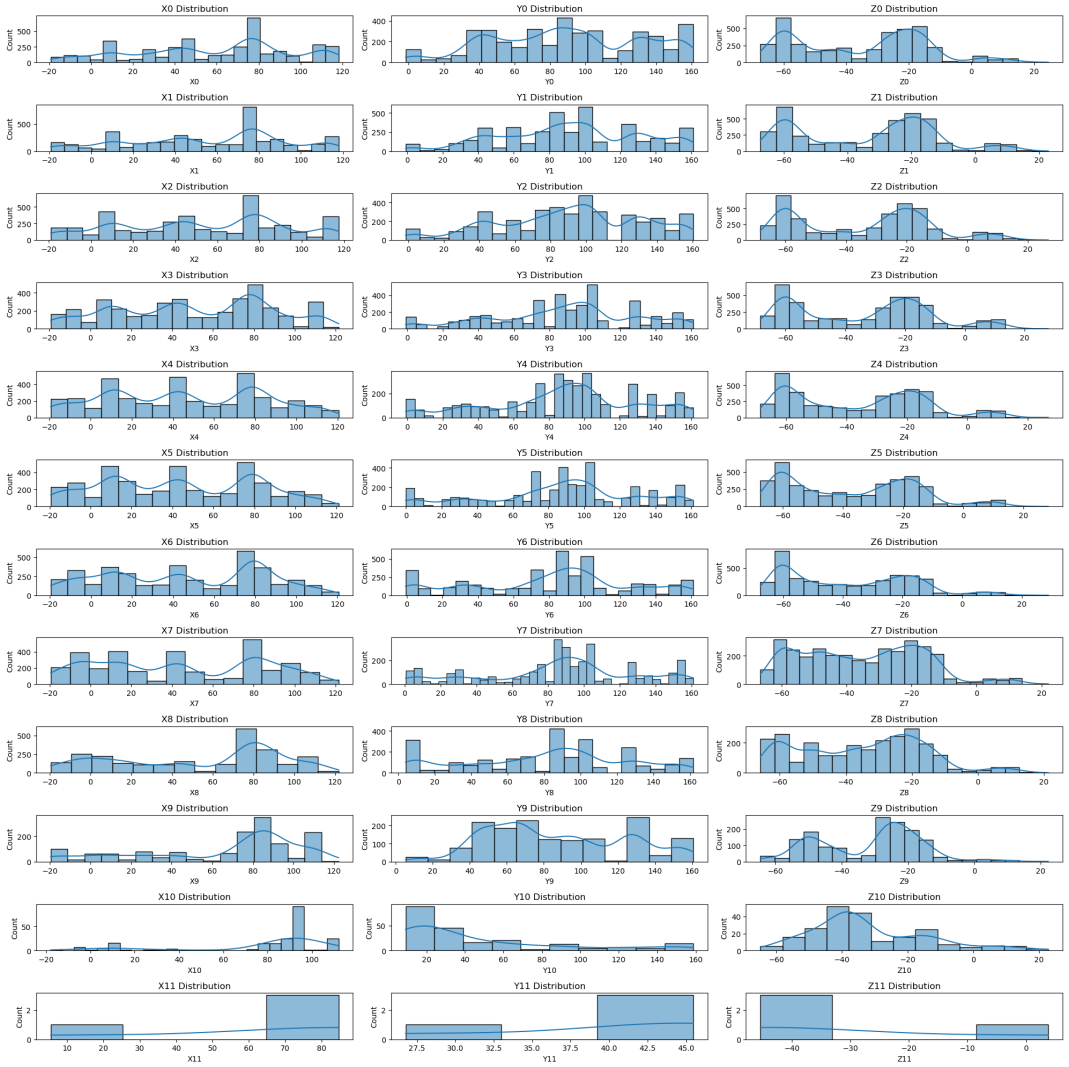

User 8 and Class
User
8
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution

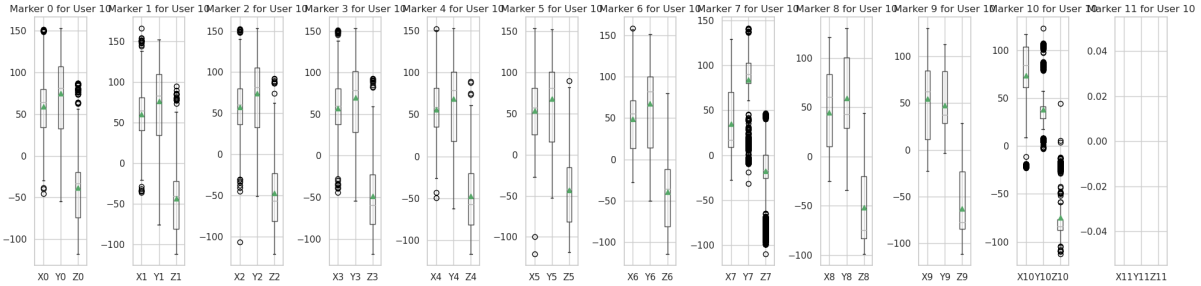
User 9:

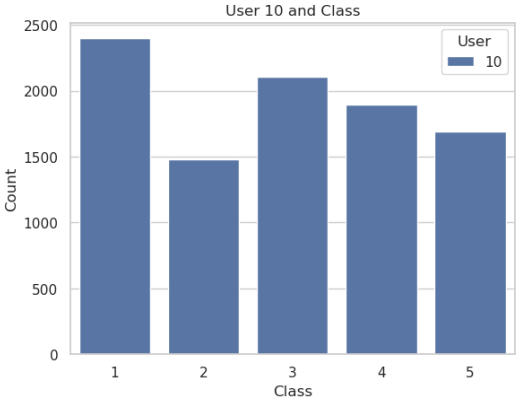

Marker 0 for User 9
Marker 1 for User 9
Marker 2 for User 9
Marker 3 for User 9
Marker 4 for User 9
Marker 5 for User 9
Marker 6 for User 9
Marker 7 for User 9
Marker 8 for User 9
Marker 9 for User 9
Marker 10 for User 9
Marker 11 for User 9
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10 Y10 Z10
X11 Y11 Z11

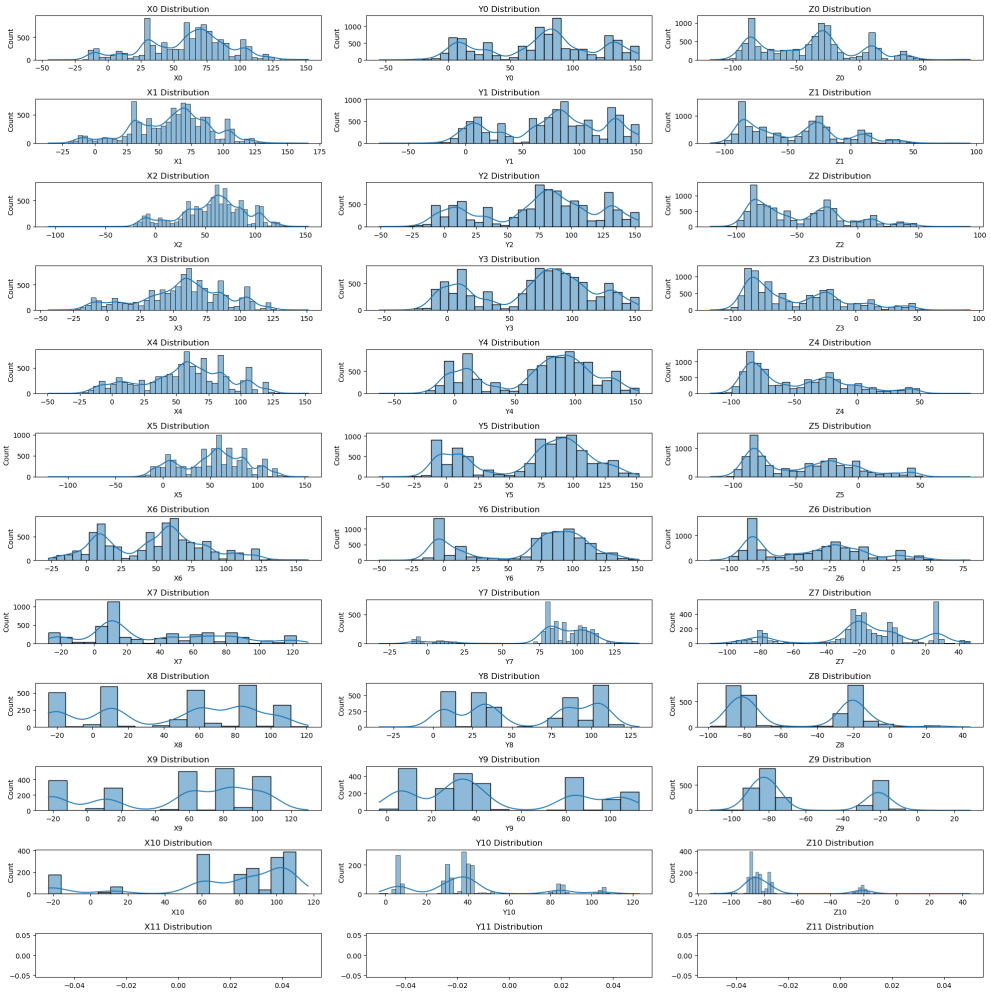
User 9 and Class
User
9
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

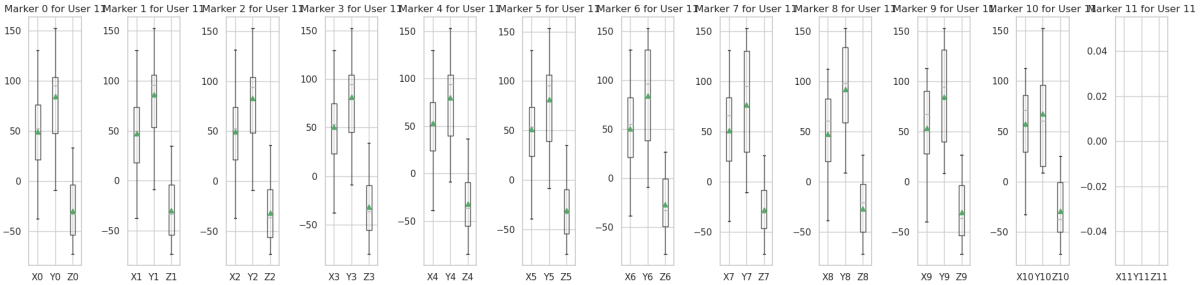
User 10:

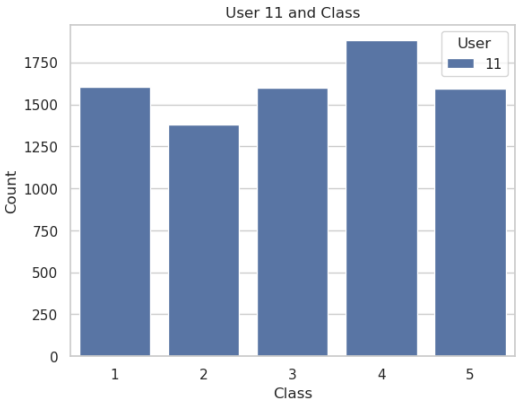

Marker 0 for User 10
Marker 1 for User 10
Marker 2 for User 10
Marker 3 for User 10
Marker 4 for User 10
Marker 5 for User 10
Marker 6 for User 10
Marker 7 for User 10
Marker 8 for User 10
Marker 9 for User 10
Marker 10 for User 10
Marker 11 for User 10
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

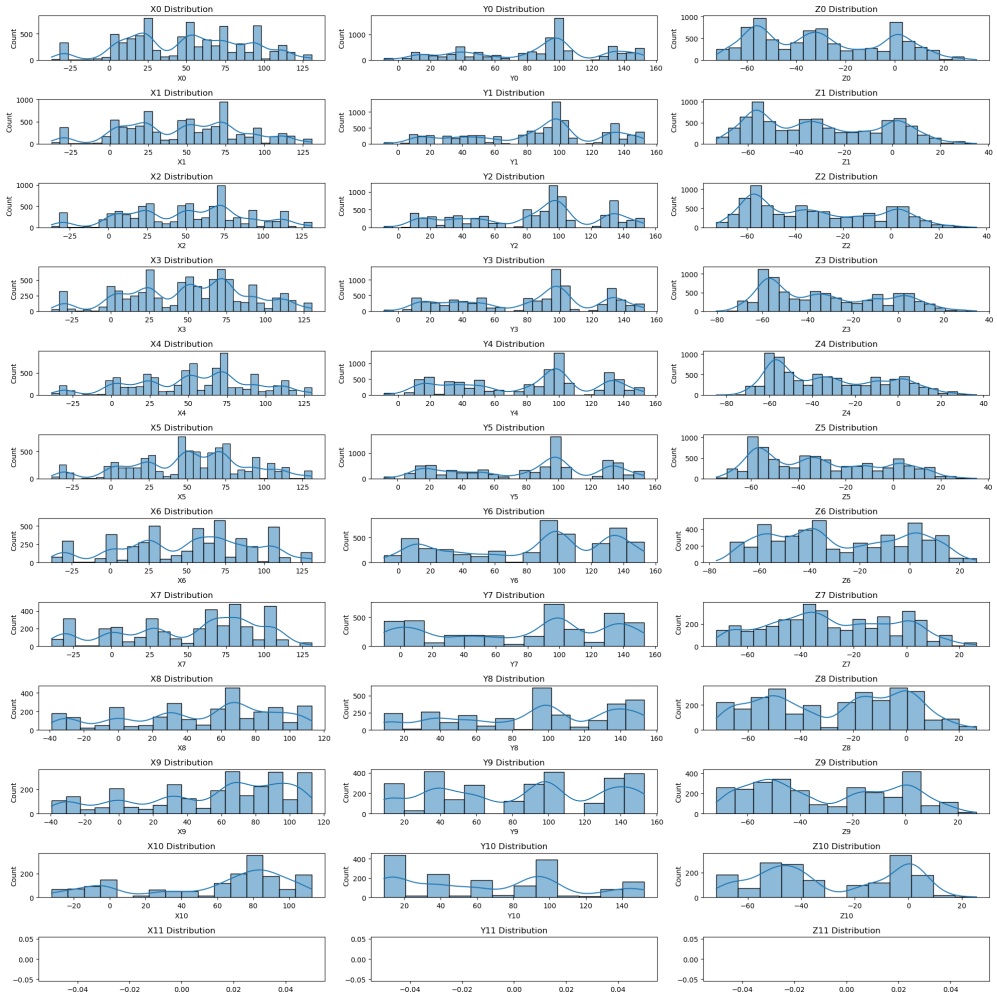

User 10 and Class
User
10
Count
Class
1
2
3
4
5


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

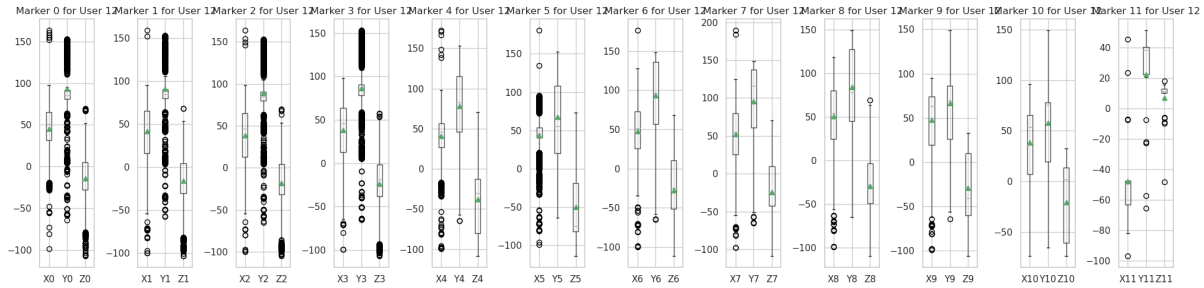
User 11:

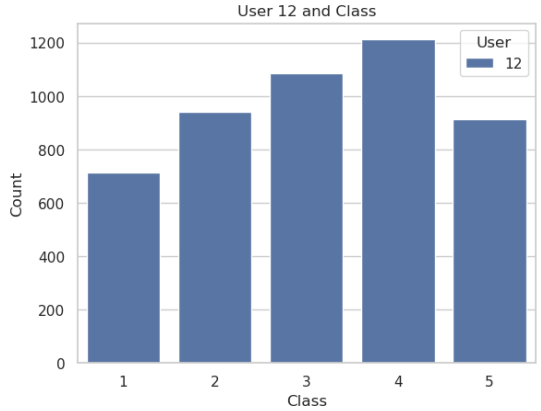


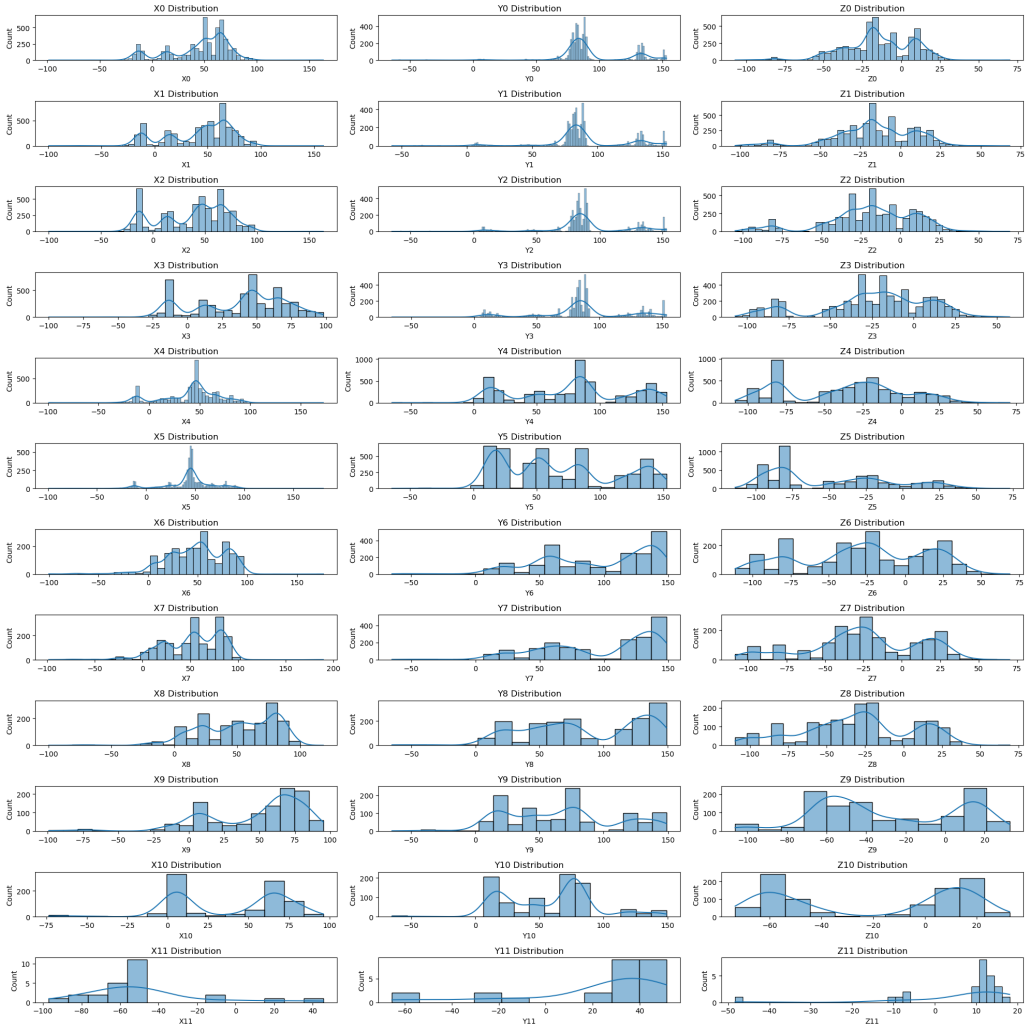




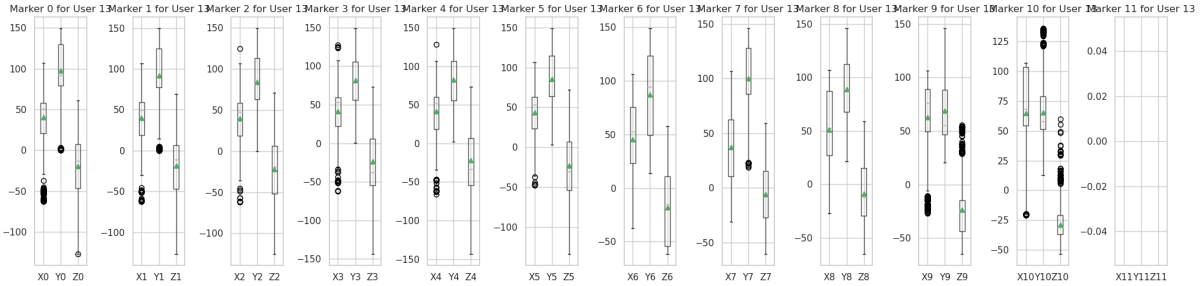
User 12:


Marker 0 for User 12
Marker 1 for User 12
Marker 2 for User 12
Marker 3 for User 12
Marker 4 for User 12
Marker 5 for User 12
Marker 6 for User 12
Marker 7 for User 12
Marker 8 for User 12
Marker 9 for User 12
Marker 10 for User 12
Marker 11 for User 12
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10 Y10 Z10
X11 Y11 Z11

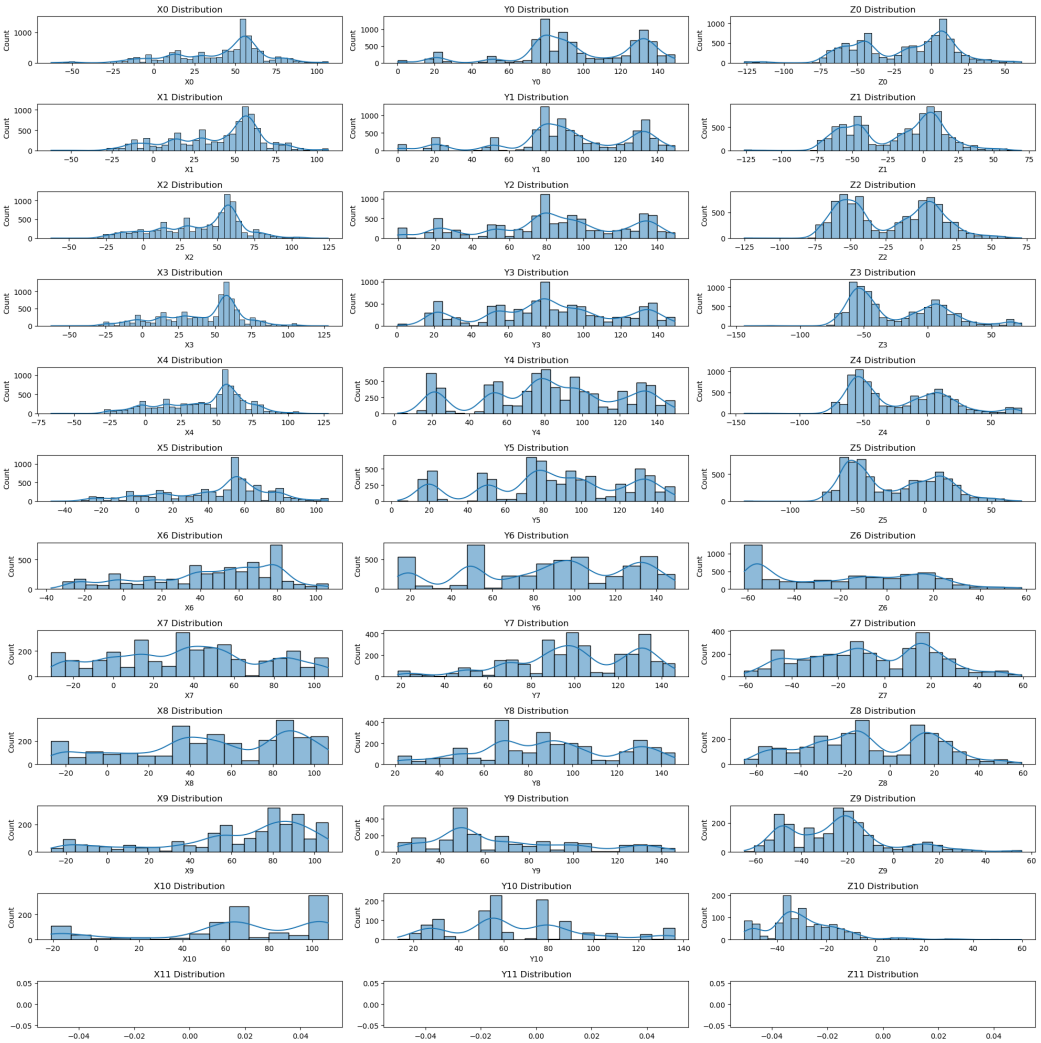

User 12 and Class
User
12
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

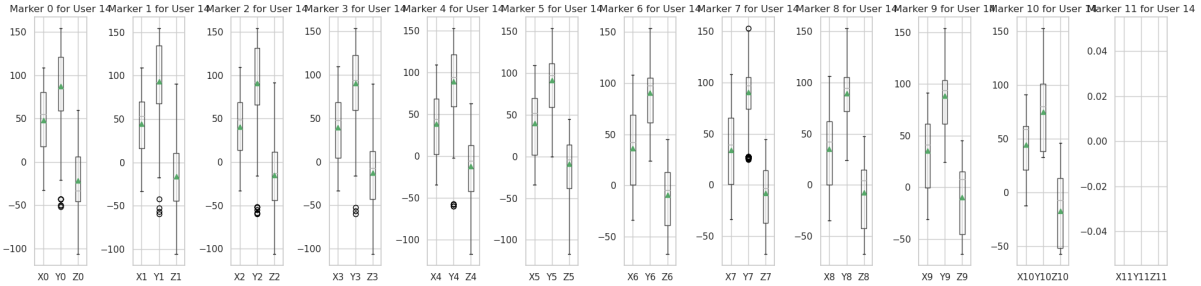
User 13:

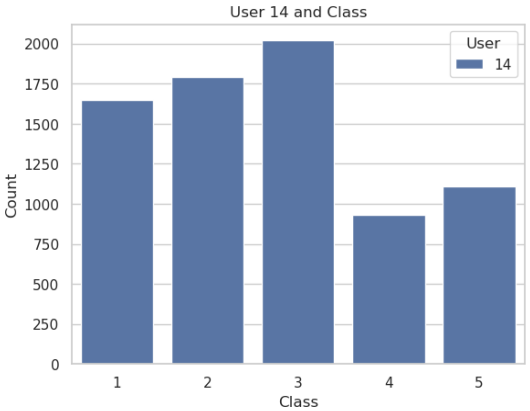

Marker 0 for User 13
Marker 1 for User 13
Marker 2 for User 13
Marker 3 for User 13
Marker 4 for User 13
Marker 5 for User 13
Marker 6 for User 13
Marker 7 for User 13
Marker 8 for User 13
Marker 9 for User 13
Marker 10 for User 13
Marker 11 for User 13
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10Y10Z10
X11Y11Z11

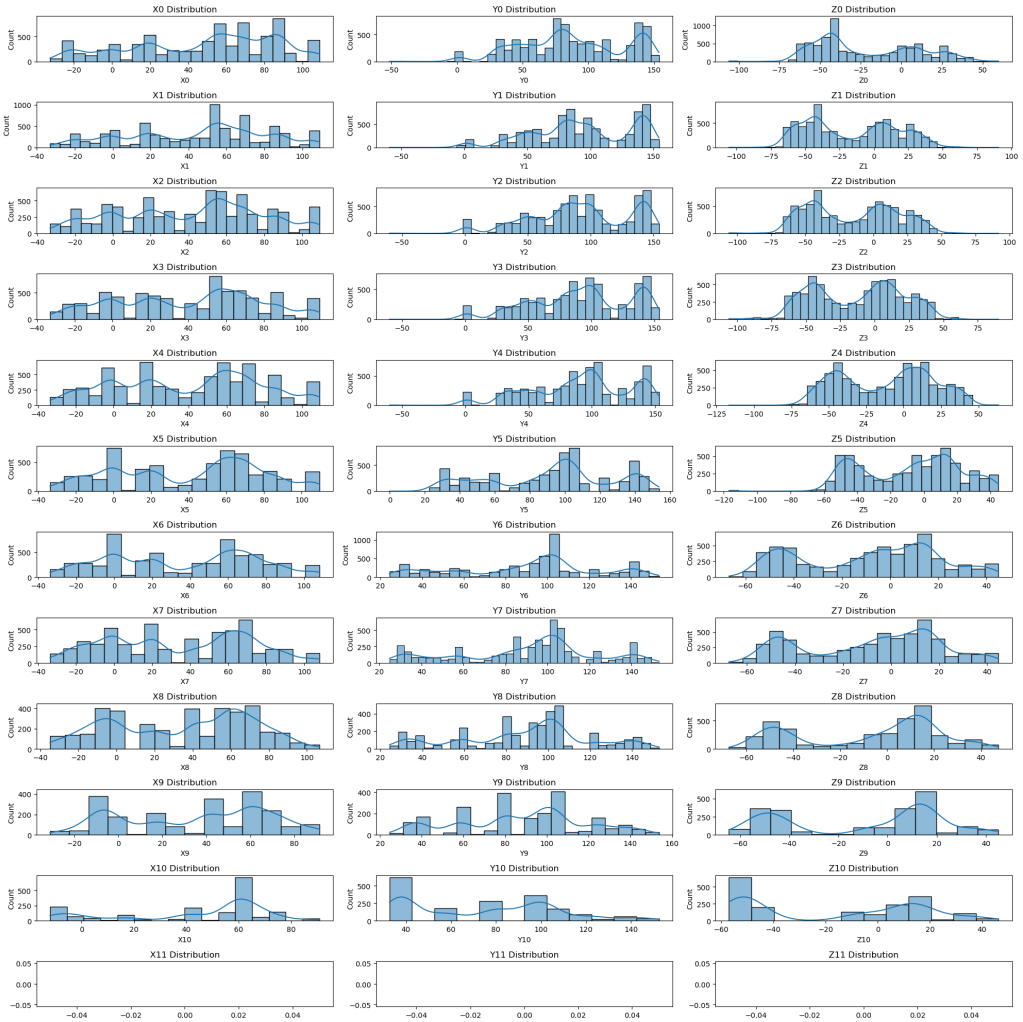
User 13 and Class
User
13
Count
Class


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

User 14:


Marker 0 for User 14
Marker 1 for User 14
Marker 2 for User 14
Marker 3 for User 14
Marker 4 for User 14
Marker 5 for User 14
Marker 6 for User 14
Marker 7 for User 14
Marker 8 for User 14
Marker 9 for User 14
Marker 10 for User 14
Marker 11 for User 14
X0 Y0 Z0
X1 Y1 Z1
X2 Y2 Z2
X3 Y3 Z3
X4 Y4 Z4
X5 Y5 Z5
X6 Y6 Z6
X7 Y7 Z7
X8 Y8 Z8
X9 Y9 Z9
X10 Y10 Z10
X11 Y11 Z11


User 14 and Class
User
14
Count
Class
1
2
3
4
5


X0 Distribution
Y0 Distribution
Z0 Distribution
X1 Distribution
Y1 Distribution
Z1 Distribution
X2 Distribution
Y2 Distribution
Z2 Distribution
X3 Distribution
Y3 Distribution
Z3 Distribution
X4 Distribution
Y4 Distribution
Z4 Distribution
X5 Distribution
Y5 Distribution
Z5 Distribution
X6 Distribution
Y6 Distribution
Z6 Distribution
X7 Distribution
Y7 Distribution
Z7 Distribution
X8 Distribution
Y8 Distribution
Z8 Distribution
X9 Distribution
Y9 Distribution
Z9 Distribution
X10 Distribution
Y10 Distribution
Z10 Distribution
X11 Distribution
Y11 Distribution
Z11 Distribution
Count

As we saw in the previous section, the shape of the histograms for the features within the same set Xi, Yi, Zi is very similar, or even more, they have the same shape. However, as we saw before, as the marker number increases, the histograms of the markers become less similar to those at the beginning. If we pay attention, this difference between the markers becomes very prominent for markers 6 and 7. And if we recall from the missing data analysis, marker 6 has a 33% missing data rate, and marker 7 has a 50% missing data rate. Therefore, this change in the histogram distributions is mainly related to the percentage of missing data.

This difference in the shape of the histograms especially impacts the outliers. Generally, we observe zones in the histograms where the number of samples is concentrated, and zones where the concentration is very low. This difference is clearly visible in markers with smaller numbers (where there is not much missing data). However, for example, in marker 7, this difference in concentrations is hardly noticeable. A suitable example for this situation is when you are drunk (with a lot of missing data) and throwing darts at a target. It is obvious that you will miss more when you are drunk compared to when you are sober.

Another important aspect to mention is that none of the histograms have a known distribution. There are some that resemble the normal distribution, for example, some histograms from the Xi set, but we see that none of them exhibit symmetry. Therefore, it becomes difficult to define a specific value to consider as outliers. In the end, we have decided not to discard these values that deviate from the mean because they may provide useful information, and we do not know the underlying distribution of the features.

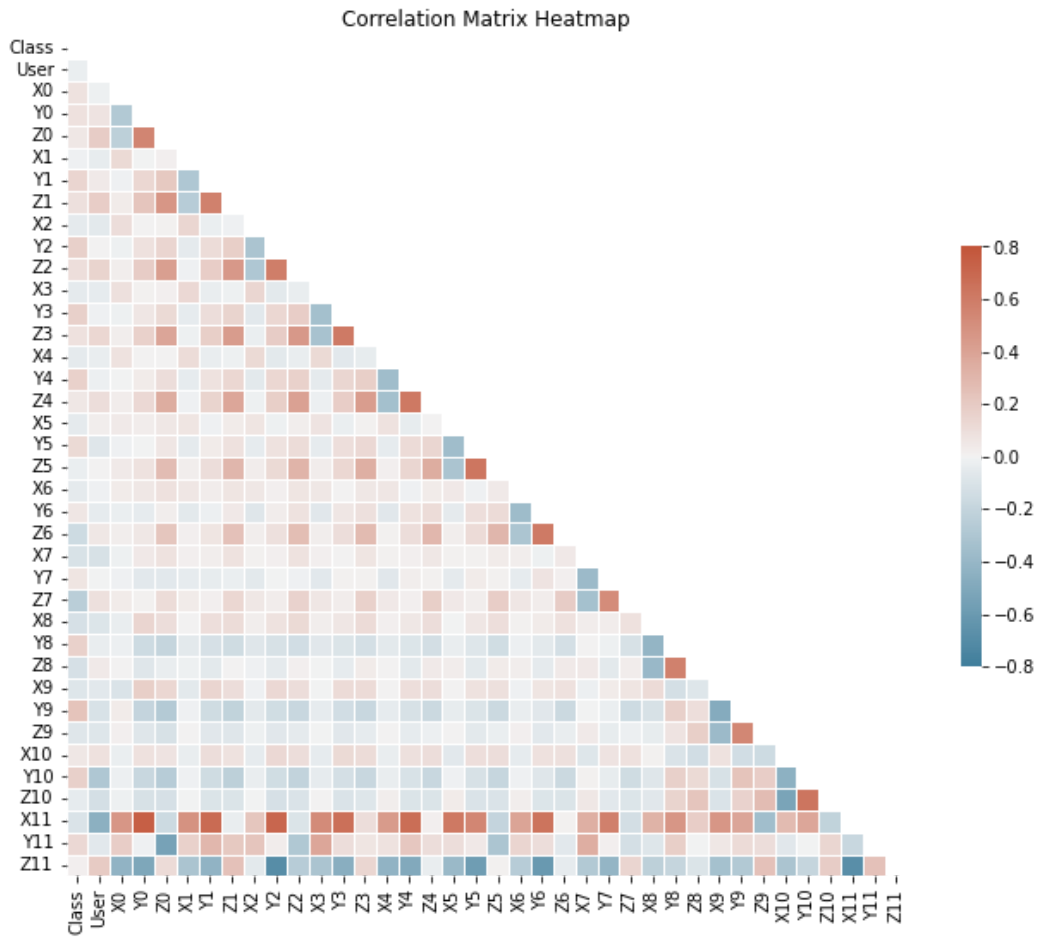
4.2 Bivariate analysis

Bivariate analysis is a crucial step as it seeks to understand the relationships between two variables. The MoCap Hand Postures dataset is complex, featuring multiple variables that represent the x, y, and z coordinates of different markers on a glove.

By visualizing the relationships, it's easier to get a concrete understanding of these aspects. It simplifies the decision-making process when it comes to selecting the most appropriate machine learning algorithms and choosing the right features.

4.2.1 Correlation Heat Map

A correlation matrix heatmap is being created to understand the relationships between the different attributes in the dataset. This correlation matrix is a powerful tool that can reveal much about the data. This produces the following result:



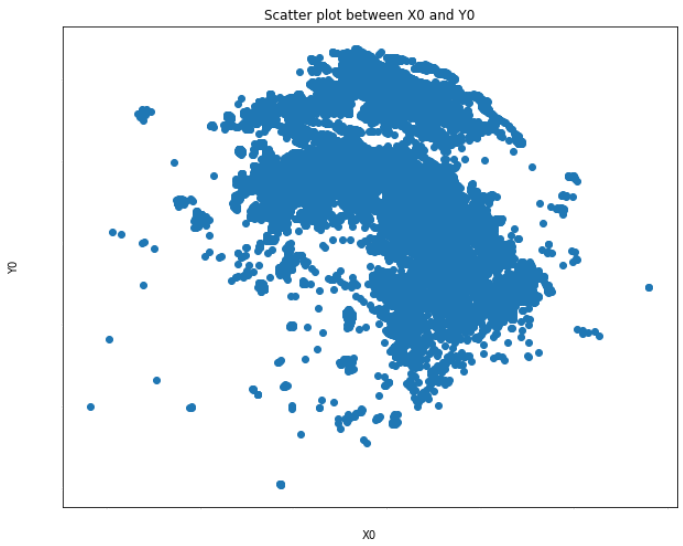
The primary aim is to find variables that have a high correlation (either positive or negative). High correlation (close to 1 or -1) between two variables suggests they change in relation to each other: as one increases, the other either increases (positive correlation) or decreases (negative correlation). If the correlation between two variables is high, they may be carrying similar information, leading to consideration of whether both variables need to be retained.

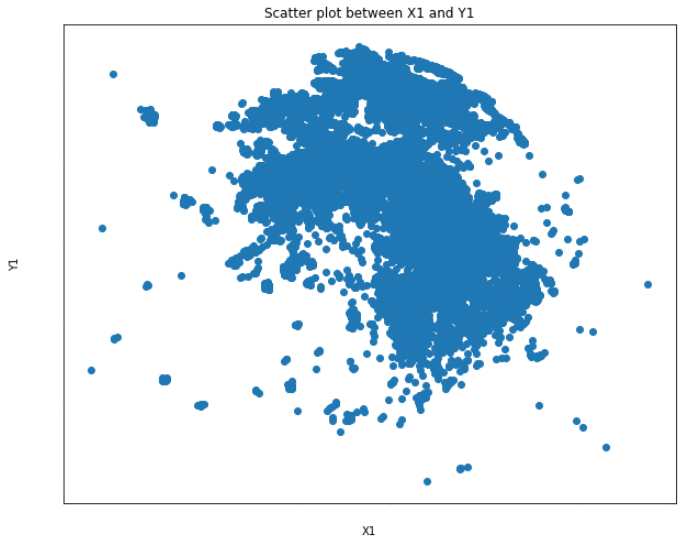
Some conclusions are:

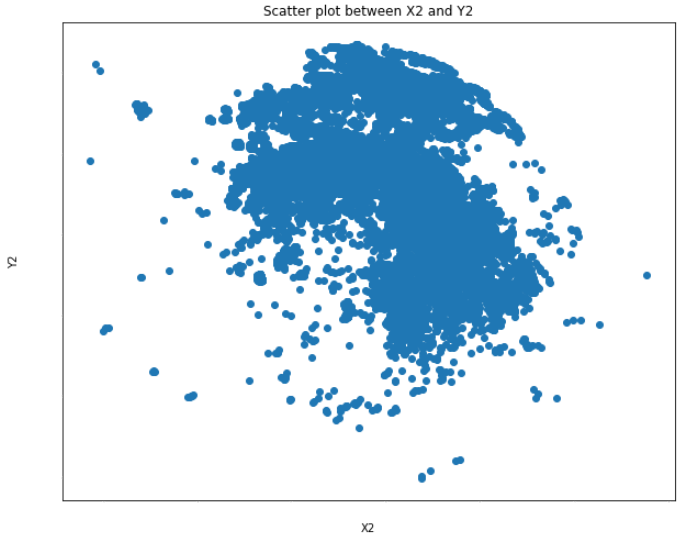
- The 'Z' coordinates (Z0, Z1, Z2, ...) are highly correlated with many other variables, indicating that they might share a lot of common information with those variables. This could potentially be due to the 3D nature of the data, as changes in 'Z' coordinates might often coincide with changes in other dimensions. Alternatively, it might reflect certain common patterns or movements across different users and classes.
- The 'X11' variable also exhibits high correlation with other variables, suggesting that this marker's position might be particularly influential or responsive to hand postures.
- On the other hand, 'Z11' shows a relatively low (blue color) correlation with most other variables. This could mean that the 'Z' coordinate of the 11th marker is more independent and does not change in tandem with other variables. Perhaps, this marker is located at a part of the hand where movements are less likely to cause changes in other parts.

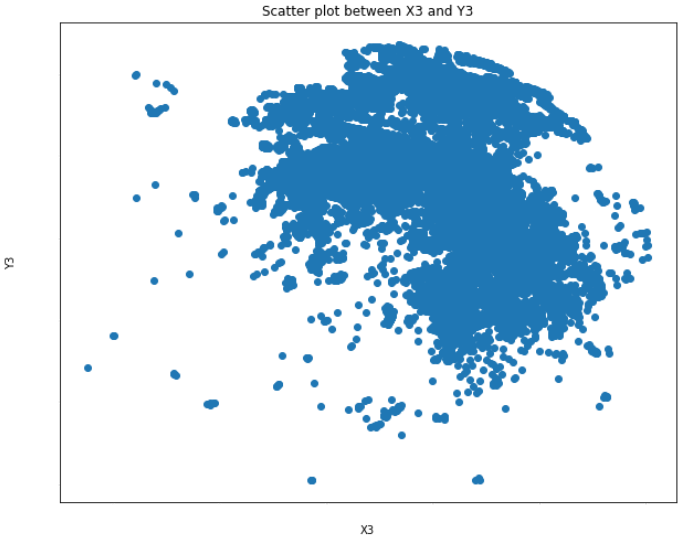
4.2.2 Scatter Plots

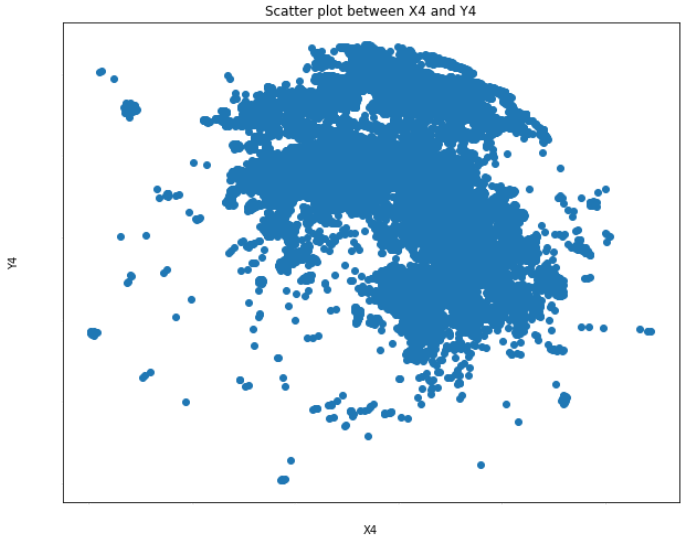
The Scatter plots between pairs of X and Y coordinates can provide insights into the relationships between movements in different parts of the hand. For example, a strong correlation between the X coordinate of and the Y coordinate of could suggest that these two parts of the hand often move together in certain postures.

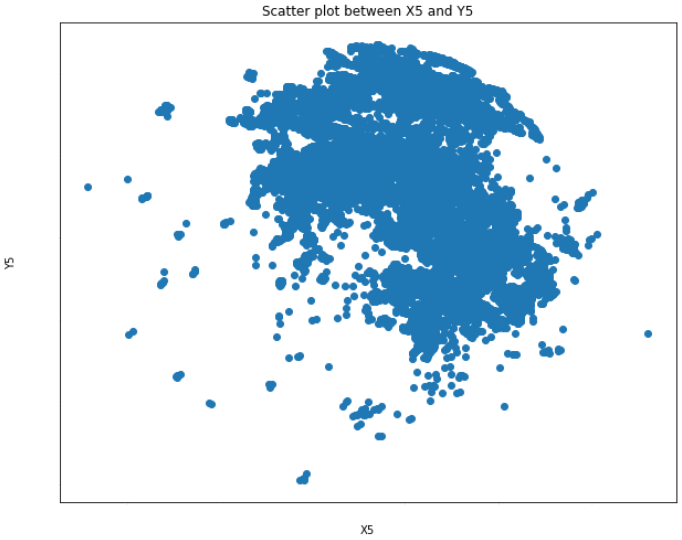


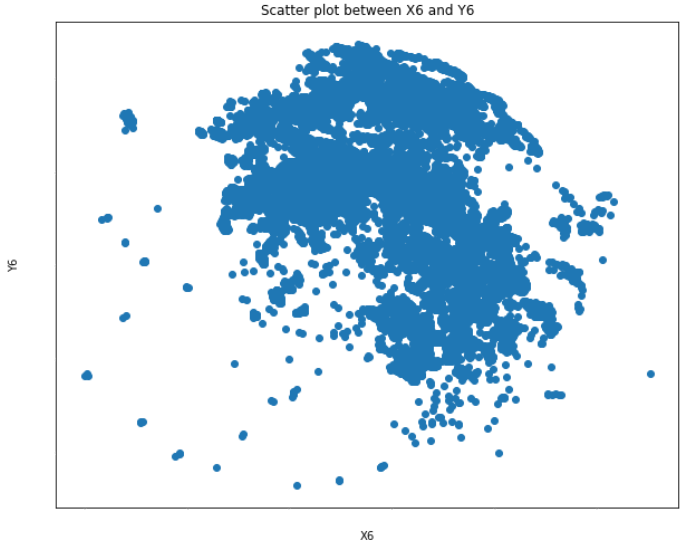


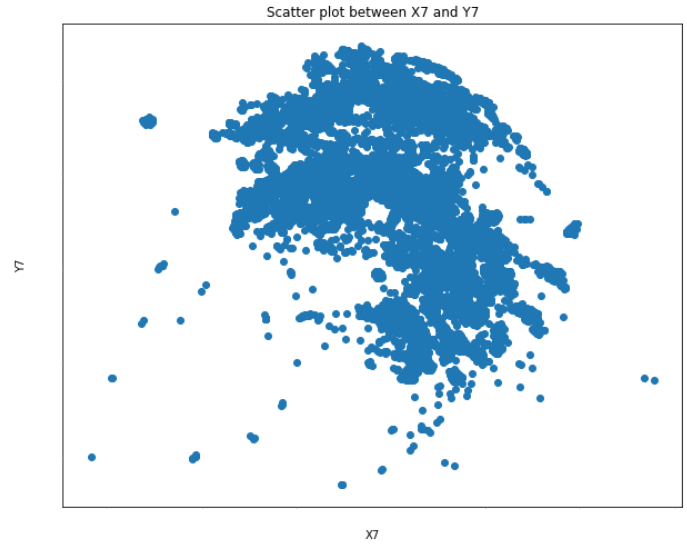

Scatter plot between X2 and Y2
Y2
X2

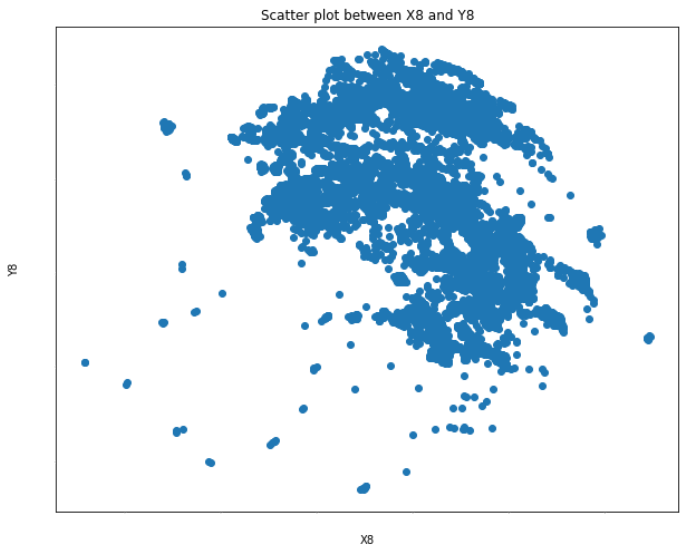

Scatter plot between X3 and Y3
Y3
X3

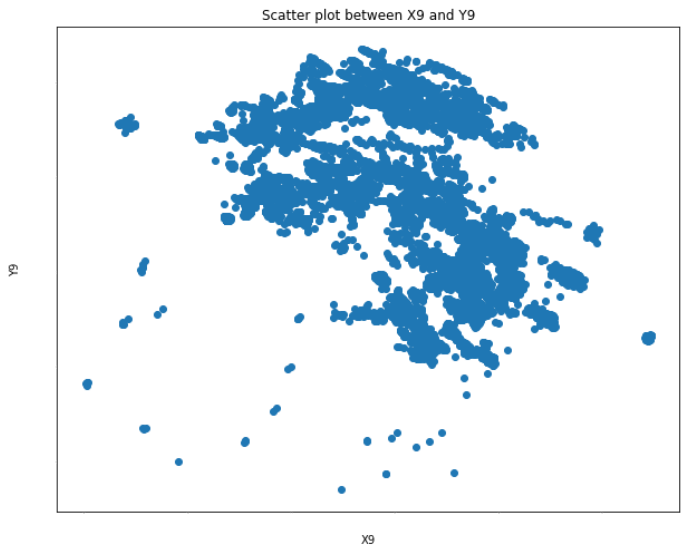

Scatter plot between X4 and Y4
Y4
X4

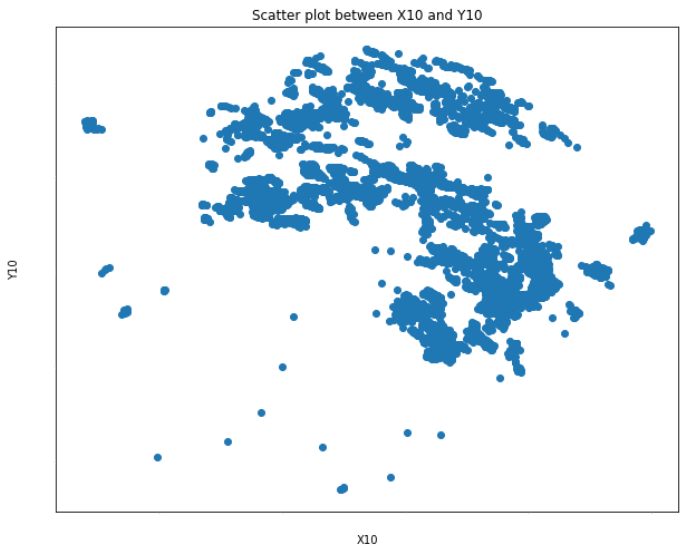

Scatter plot between X5 and Y5
Y5
X5

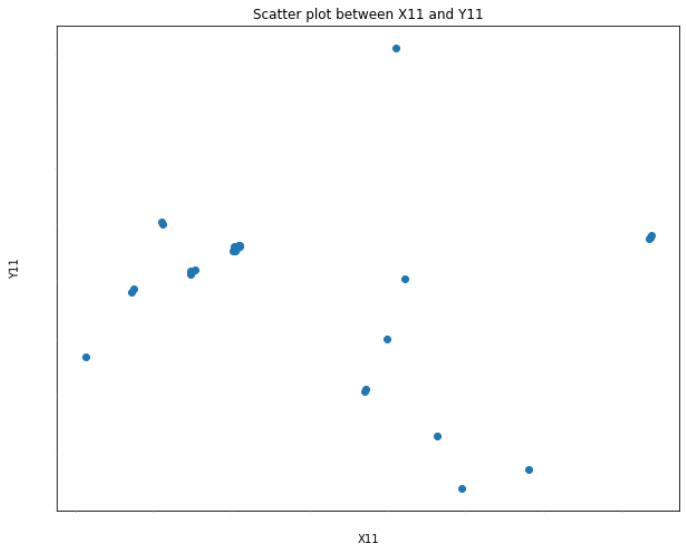

Scatter plot between X6 and Y6
Y6
X6


Scatter plot between X7 and Y7
Y7
X7









Some conclusions are:

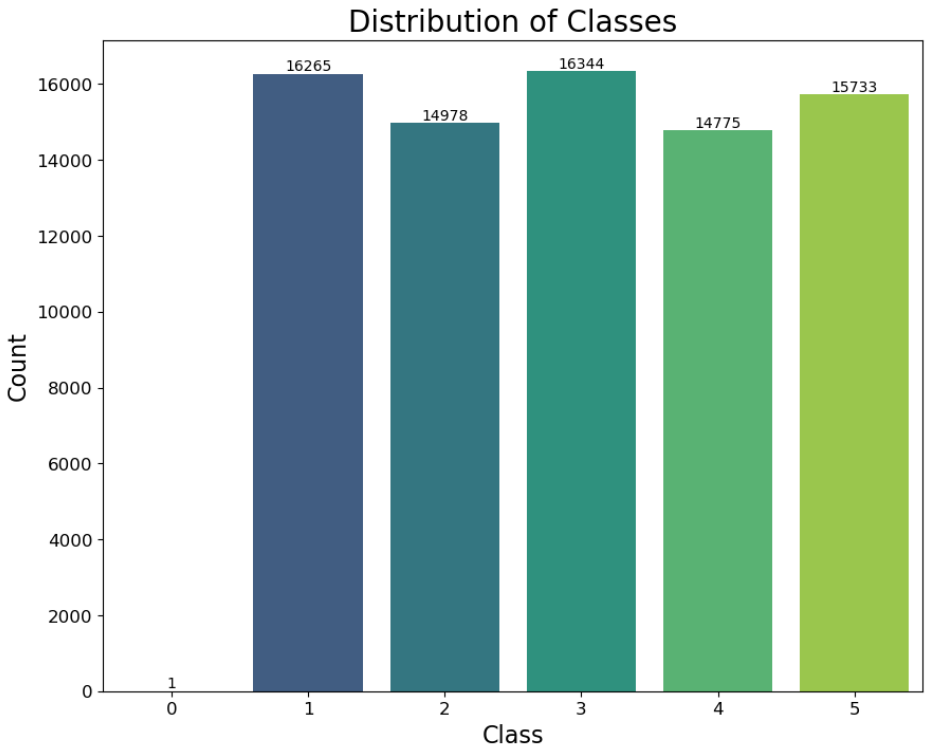
- The scatter plots for pairs X0-Y0 through to X8-Y8 show well-grouped points with few outliers. This implies that these coordinates generally move together in a predictable manner, suggesting a strong correlation between these particular pairs. This could be because these markers are attached to parts of the hand that move in a coordinated manner during the various hand postures.

- From pairs X9-Y9 and X10-Y10, the points start to scatter a bit more, indicating a weaker correlation between these pairs. This could suggest that the corresponding markers are on parts of the hand that move more independently, possibly indicating that these parts of the hand have more flexibility or variability in their movements across different postures.

- The pair X11-Y11 shows no clear pattern in the scatter plot, indicating a weak or no correlation between these coordinates. This could be due to several

reasons. One possibility is that the corresponding marker is on a part of the hand that moves very independently of the other parts, and its movements are not strongly correlated with any particular posture. Another possibility is that the marker could be missing in many records, leading to less data and thus a more scattered and less meaningful plot.

4.2.3 Class Distribution



From the class distribution, we can draw these conclusions:

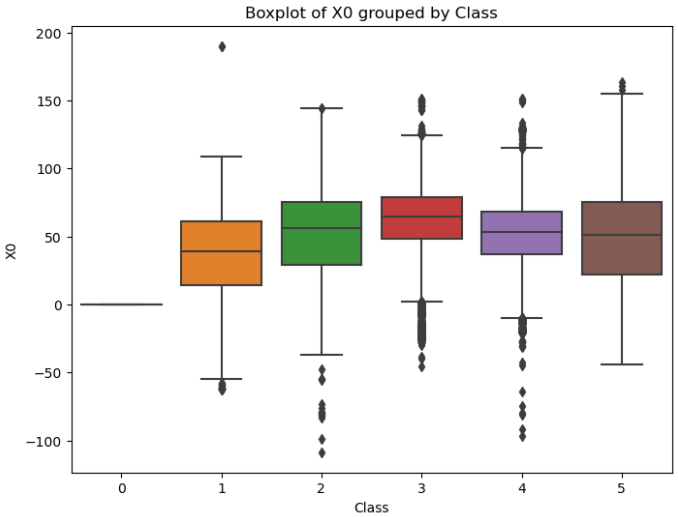
- The data is distributed fairly evenly across classes 1 to 5. Each class represents a specific hand posture(classes in this order from 1 to 5): Fist (with thumb out), Stop (hand flat), Point1 (point with pointer finger), Point2 (point with pointer and middle fingers), and Grab (fingers curled as if to grab). The balanced distribution of the classes allows for good representation during model training and helps to ensure that the model doesn't become biased toward a specific class due to overrepresentation. It's important to note that class '0' should be ignored, as it's just a dummy placeholder and doesn't represent any real data.

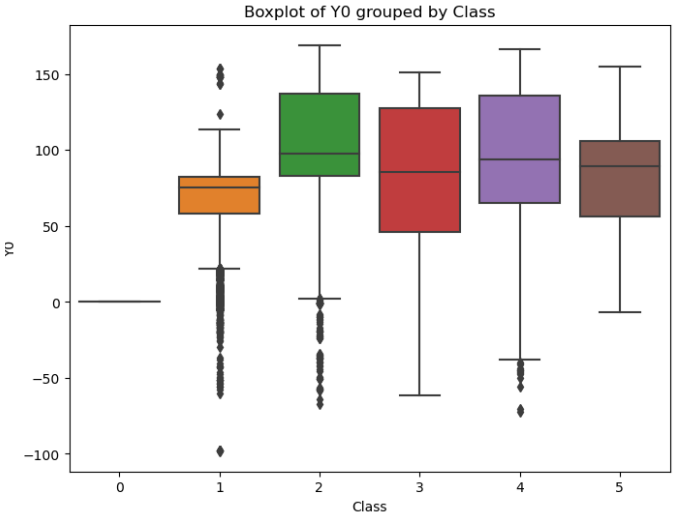
- The structure of the dataset implies that it is possible to use a leave-one-user-out cross-validation strategy when evaluating models. This approach would involve iteratively leaving out data from one user during training and then using that user's data to test the model. This approach tests the model's generalization capability to unseen users.

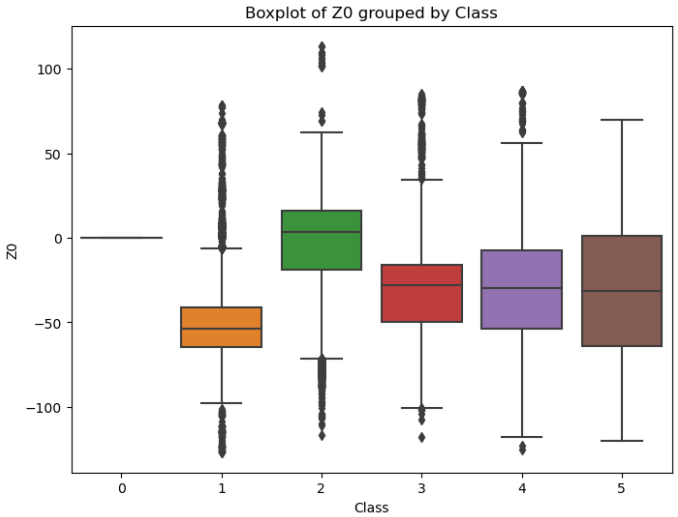
- Given the nature of the data, one can not only work on classifying hand gestures but also explore user identification or other clustering tasks. This reveals the versatility of this dataset for different machine learning tasks.

4.2.4 Marker Positions

The use of box plots for each marker position (X0, Y0, Z0) grouped by class helps to understand how these positions are distributed within each class.







Some conclusions are:

- Classes 3 and 4 in X0, and class 1 in Y0 and Z0, have a significant number of outliers. This could indicate instances where the hand posture deviated substantially from the typical posture for those classes.

- Differences in the distributions of the X0, Y0, and Z0 coordinates among the classes suggest that these coordinates might be useful features for distinguishing between classes. For example, the Z0 coordinate for class 1 is consistently below 0, while it is above 0 for class 2.

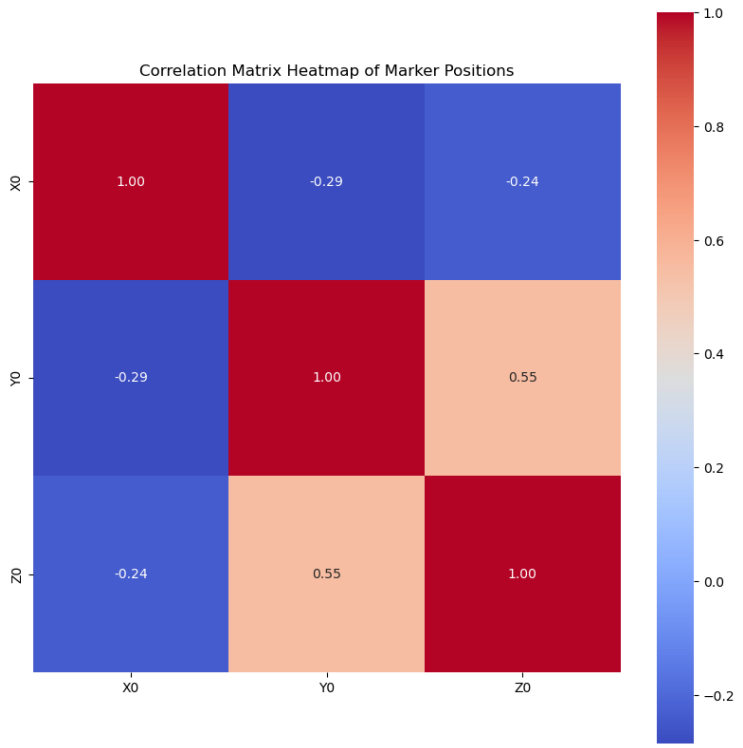
- The different patterns and distributions observed might relate to the nature of the hand postures themselves. For instance, the smaller boxes (less variation) for classes 3 and 4 in the X0 coordinate might indicate a more fixed thumb position in the "Point1" and "Point2" postures.

- The large number of outliers and small box size for class 1, and the larger box size for class 2, both in the Y0 and Z0 coordinates, might reflect inherent complexities of the "Fist" and "Stop" postures. For the "Fist" posture, the user's fingers are tightly curled, likely resulting in less variability in marker positions (small box) but also a higher chance of extreme cases (outliers), due to the individual variations in fist-making. The "Stop" posture, on the other hand, requires an open hand, which might result in more spread out marker positions.

- The Z0 positions of these two classes stand out in their relationship to the zero point. Class 1 is consistently below zero, while class 2 is above zero. This could be related to the hand postures; in a fist (class 1), the fingers (and therefore markers) are likely curled under the hand, hence below zero in the Z coordinate. In contrast, the "Stop" gesture (class 2) involves an open hand with fingers extended upwards, which would likely result in Z positions above zero.

- These insights suggest that the X, Y, and Z positions of the markers, especially those of the thumb, can serve as useful features for distinguishing between different hand postures in the dataset. They also highlight the influence of the nature of the hand postures themselves on the marker positions, which underscores the complexity of the classification task.

4.2.5 Markers heat map



We can draw these conclusions from the heat map:

- There is a moderate positive correlation between the Y0 and Z0 positions. This means that as the Y0 value increases, the Z0 value also tends to increase and vice

versa. This could suggest that certain movements or postures that involve raising the hand (increasing Y0) might also involve moving the hand forward or backward (increasing or decreasing Z0).

- There is a weak negative correlation between the X0 and Y0 positions. This means that as the X0 value increases, the Y0 value tends to decrease slightly and vice versa. This could suggest that certain movements or postures that involve moving the hand to the right (increasing X0) might slightly involve lowering the hand (decreasing Y0).

- There is a weak negative correlation between the X0 and Z0 positions. This means that as the X0 value increases, the Z0 value tends to decrease slightly and vice versa. This could suggest that certain movements or postures that involve moving the hand to the right (increasing X0) might slightly involve moving the hand backwards (decreasing Z0).

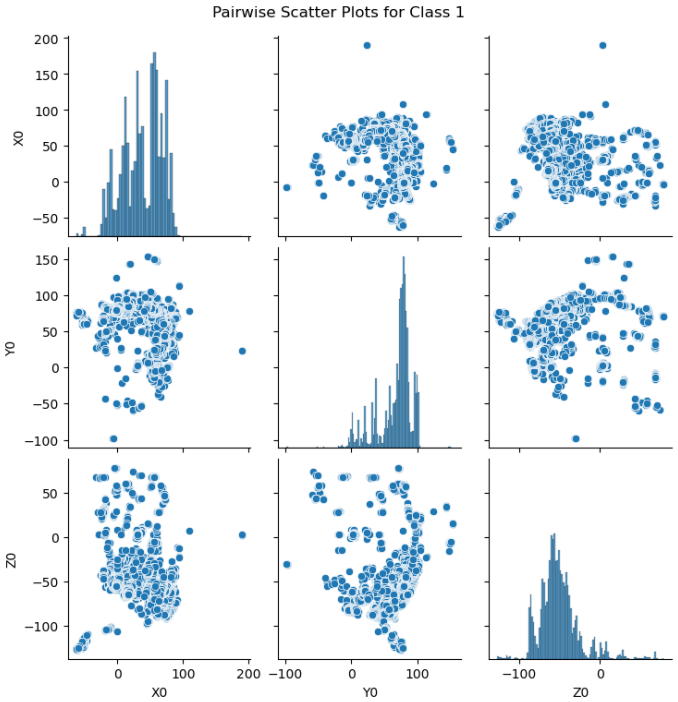
4.2.6 Pairwise scatter plots

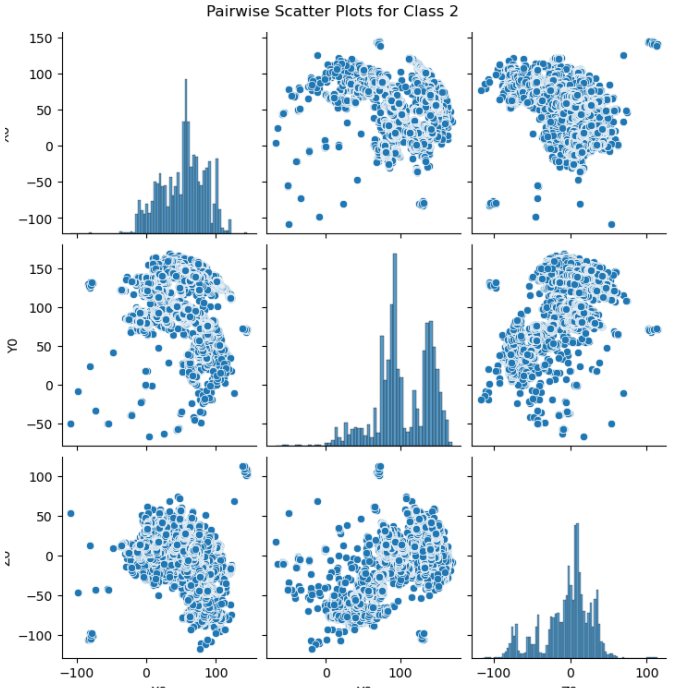
Each scatter plot in the grid shows the relationship between two of the markers' positions. Each dot represents an individual observation in the dataset. The position of the dot at the intersection of X and Y indicates the values of the two variables for that observation. If there is a clear trend in the dots (e.g., upward, downward, linear, or non-linear), it suggests a correlation between the two variables. Outliers can also be identified as dots that are distinctly apart from the main group of dots.

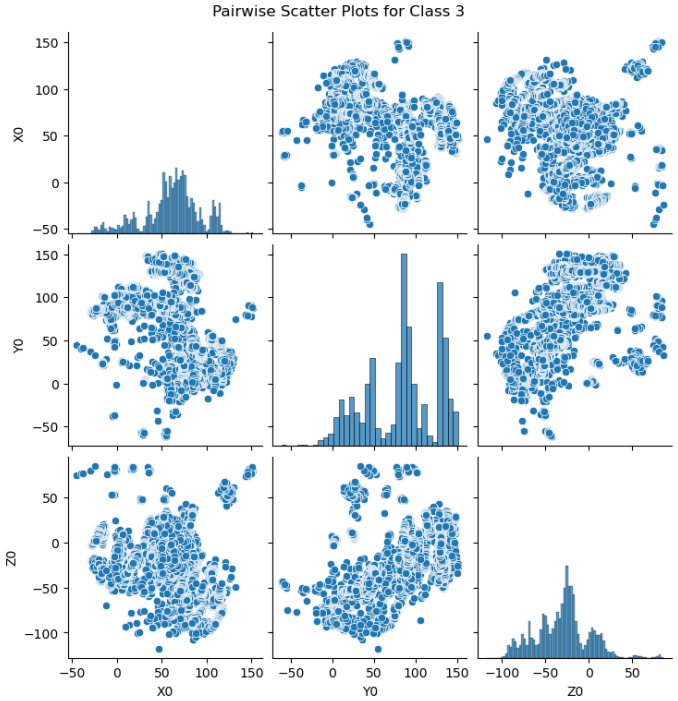
The diagonal plots are histograms, which show the distribution of a single variable. This allows you to see the spread of values and the shape of the distribution for each variable within each class.

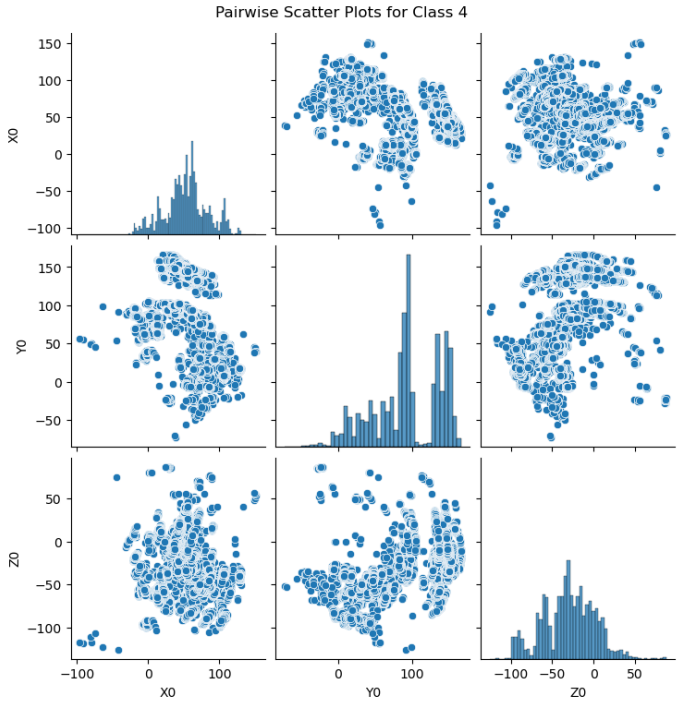
By generating these plots for each class, we can compare how the relationships between variables differ between classes. For instance, a certain pattern or correlation might only appear for one class, suggesting that it's a characteristic of that specific hand posture.

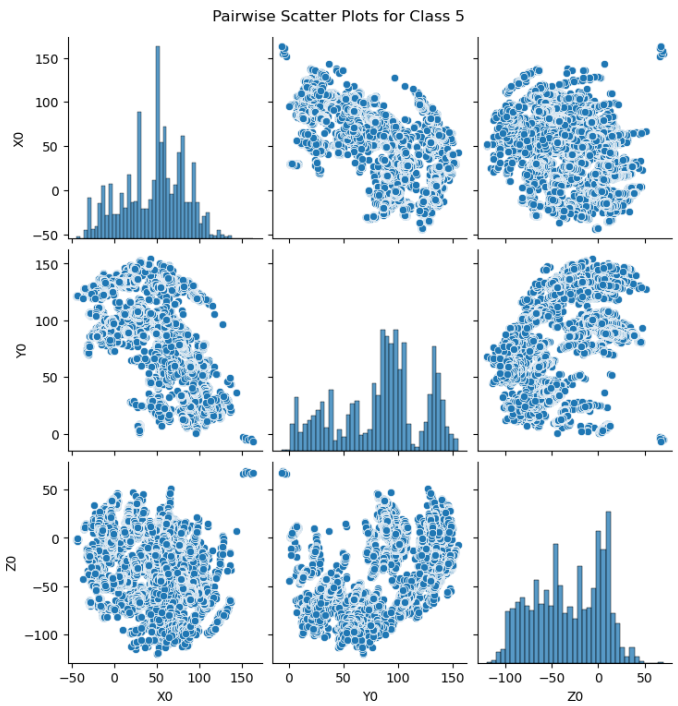
We need to bear in mind that correlations found in scatter plots don't imply causation. Further analysis would be needed to determine whether a change in one variable causes a change in another.


Pairwise Scatter Plots for Class 1
X0
Y0
Z0


Pairwise Scatter Plots for Class 2
Y0


Pairwise Scatter Plots for Class 3
X0
Y0
Z0


Pairwise Scatter Plots for Class 4
X0
Y0
Z0



4.3 Evaluation criteria of data mining models

We have chosen to use the following metrics for evaluating the data mining models, the most important and the one we use to rank the models, the accuracy, this metric provides the percentage of correctly classified instances out of the total instances and is widely used. For the sake of completeness, we have also calculated the precision, which measures the proportion of correctly predicted positive instances out of all the instances that are predicted as positive. We also use recall, which focuses on capturing all positive instances and minimizing false negatives, and test f1 score, which balances both precision and recall.

Cross-validation is used on models that use grid search using the method grid search CV, which tries out all the parameters given, by default it uses the 5-fold cross validation. Why do we use 5? 5 is a common benchmark for these kinds of models, it strikes a good balance between computational cost and a reliable performance estimate, it also means we have sufficient data for training because each fold contains 80% of the data for the training of the model.
So we divide our data into 80% training data, and 20% testing data.

5. Machine learning methods

5.1 Naïve Bayes

Two types of predictions have been performed using the Naive Bayes algorithm: one using the fit method to adjust the model to the training data, and another using cross-validation to evaluate the model's performance on new data. This combination allows for a more comprehensive and reliable understanding of the performance of the Naive Bayes algorithm on the given dataset. The results obtained for each case are as follows:

FIT:

acc	prec	recall	f1 score
0.3823	0.3946	0.3823	0.3474

CROSS VALIDATION:

acc	prec	recall	f1 score
0.2359	0.2797	0.2359	0.2135

It can be observed that both predictions yield relatively low results compared to higher values of precision, recall, and F1 score that would be expected in a highly accurate model. This could indicate that the Naive Bayes model does not fit well with the data used or that the data itself presents significant challenges for prediction.

CONCLUSION:

One possible explanation for the low results could be related to the assumption of independence of variables in the Naive Bayes algorithm. This algorithm assumes that the features are independent of each other, meaning that the presence or absence of one feature does not affect the presence or absence of other features. However, this assumption may not hold true in the case of our data. Given that some features are correlated on other features to accurately predict the target class.

5.2 K-NN

K-Nearest Neighbors (K-NN) is a simple, instance-based learning algorithm. It's called instance-based because it doesn't explicitly learn a model. Instead, it classifies new instances based on their similarity to the existing instances which have been seen during the training phase.

The algorithm calculates the distance between the new instance and all existing instances, and then selects the K closest instances (neighbors). The predicted label of the new instance is then determined by the majority label among those K neighbors.

In our K-NN modeling approach, we employed feature scaling as a vital preprocessing step. Feature scaling is necessary due to the distance-based nature of the K-NN algorithm.
In K-NN, the algorithm determines the class of an unknown instance based on the classes of its nearest neighbors. The "nearness" of neighbors is determined by measuring the distance between the instance and all other instances in the dataset. Consequently, the magnitude and range of each feature heavily influence the computed distances.

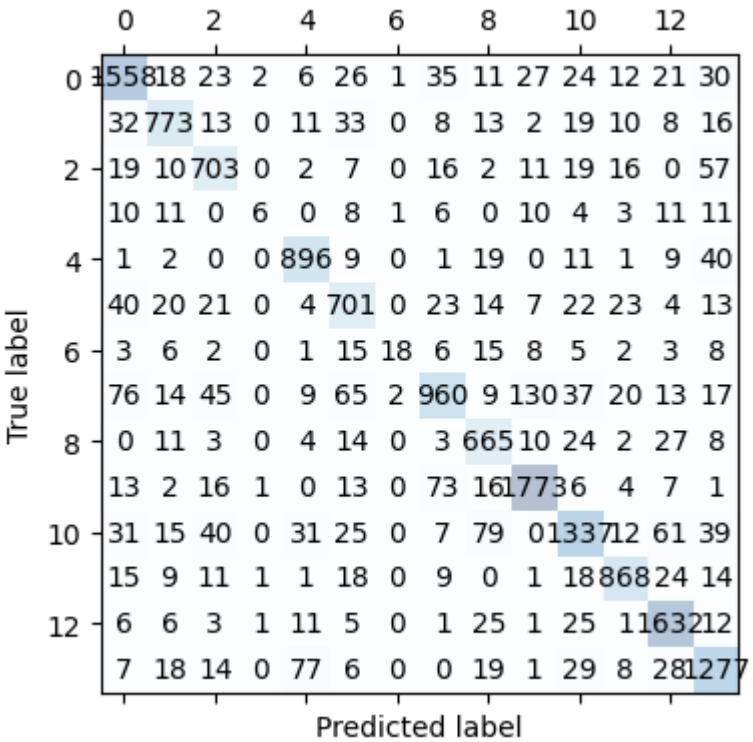
For example, let's consider two features: **WristPosZ** and **ThumbEndPosZ**. These features represent the z-coordinate positions of the wrist and thumb, respectively.
Now, imagine **WristPosZ** typically varies between -30 to 30, whereas **ThumbEndPosZ** varies in a much larger range, say -300 to 300. If we use these features in their original scale in the K-NN algorithm, **ThumbEndPosZ** will disproportionately dominate the distance calculation due to its larger scale. This could potentially diminish the contribution of **WristPosZ** in determining the similarity between instances, even if changes in **WristPosZ** could be equally or more important for differentiating between hand postures.

To prevent such issues, we apply feature scaling, which ensures all features have a similar range of values, thereby providing each feature with an equal opportunity to influence the model's predictions. This leads to more balanced distance calculations and often results in improved model accuracy.

5.2.1 Simple Cross-Validation

In this stage, we are doing a simple train-test split and fitting a basic K-NN model to our data. We then evaluate its performance by making predictions on the test set and comparing these predictions to the actual values.

Accuracy: 0.8429577464788732



Looking at the results of the simple cross-validation, it's clear that the K-NN model performed relatively well. The accuracy of approximately 84.3% indicates that the model correctly classified this percentage of the test instances.

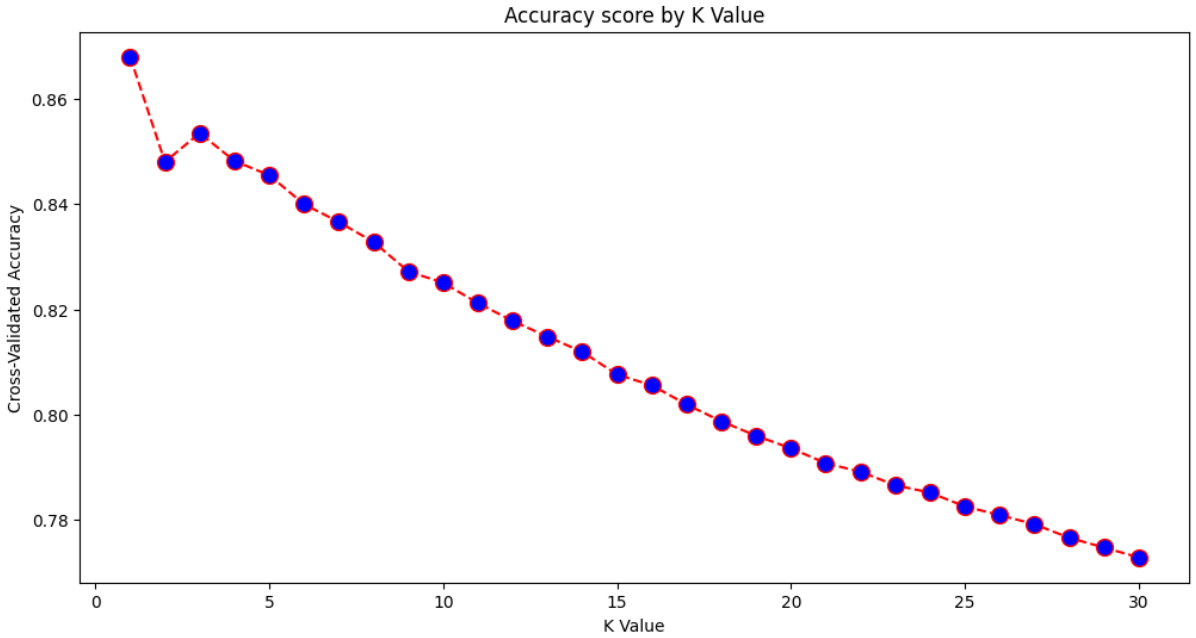
Analyzing the confusion matrix offers a more detailed view of the model's performance. The diagonal values represent the correct predictions made by the model for each class. For instance, for class `, the model made 773 correct predictions; for class 0, it correctly predicted 703 instances, and so on.

However, the model was not flawless, as the off-diagonal elements in the confusion matrix demonstrate. These values represent the instances where the model incorrectly predicted the class. For instance, in the second row for class 1, the model misclassified 32 instances as class 0, 13instances as class 2, 0 instances as class 3, etc.
Upon examining these off-diagonal values more closely, I notice that the model often confuses class 7 with class 9 (130 instances were predicted as class 9 but were actually class 7). Similarly, class 9 was often misclassified as class 8, with 161 instances mislabeled.
Moreover, it seems that the classes with fewer instances, such as class 3 and class 6, are predicted less accurately than others. This could potentially be attributed to a class imbalance problem.

These insights are valuable as they point to the areas where the model can be improved. While the overall accuracy is good, this detailed analysis from the confusion matrix helps to identify ways to enhance the model's performance further.

5.2.2 Evaluating the effect of K using 10-fold cross-validation

The value of K is a hyperparameter that we need to choose, and its optimal value can vary depending on the specific dataset. Too small a value of K can make the model sensitive to noise in the data, while too large a value can make the model too generalized.
In this stage, we run a 10-fold cross-validation for different values of K to see how the choice of K affects the model's performance. 10-fold cross-validation is a robust way to estimate the performance of a model on unseen data.



From the results of the 10-fold cross-validation, it's clear that the choice of K significantly impacts the accuracy of the K-NN model.

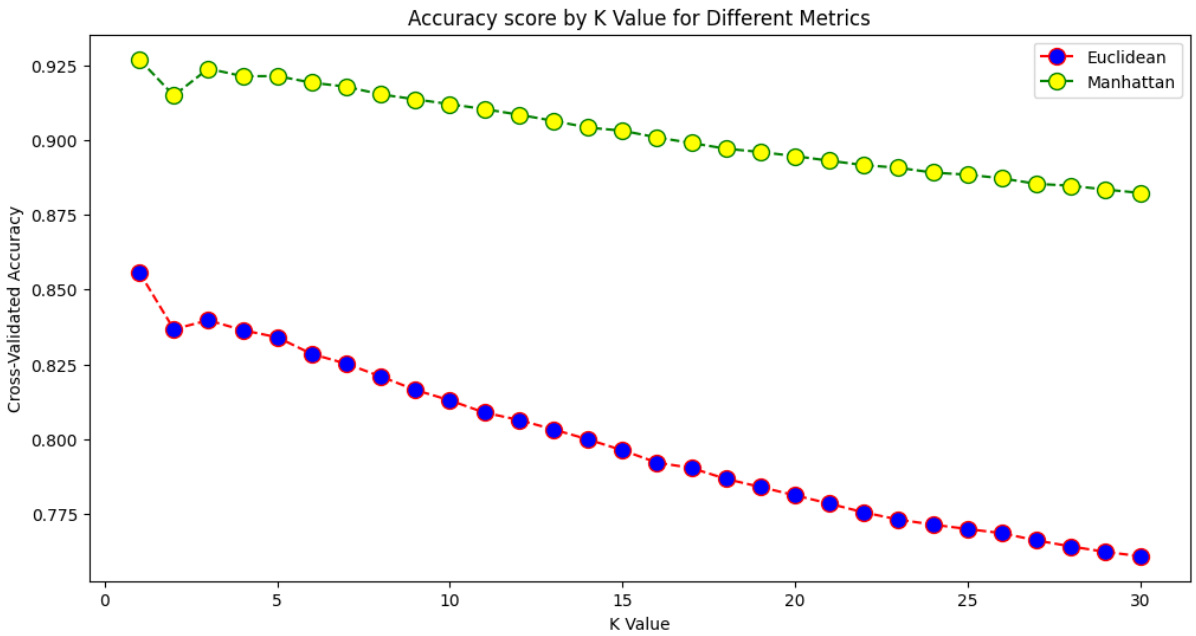
The model's accuracy is highest at K=1, reaching approximately 0.90. This indicates that the nearest single neighbor provides a reliable estimate of the class, despite the common concern that K=1 might make the model overly sensitive to noise. This suggests that there might be a high level of separability in the dataset.

However, when K is increased to 2, the model's accuracy drops sharply to 0.83. This suggests that when the model takes into account the closest two data points, it may start considering points from other classes, which can decrease the accuracy.
Interestingly, as we further increase K to 3, there is a slight increase in accuracy to about 0.85. This suggests that expanding the number of neighbors considered to three might create a more balanced decision boundary that takes into account the structure of the data.
Nevertheless, as we continue to increase K beyond this point, the accuracy starts to decline gradually. This implies that the model is beginning to overgeneralize as it considers more neighbors, many of which might be from other classes.

From these observations, it seems like the optimal value for K in this dataset is around 1, with a possible consideration for K=3. It's intriguing to observe that a smaller K works better for this dataset, which may suggest a high level of class separability. This finding will be instrumental while further tuning the model for improved performance.

5.2.3 Accuracy score by K Value for Different Metrics

K-NN uses distance measures to determine similarity, and different distance measures can give different results. In this stage, we are evaluating how the choice of distance metric (Euclidean vs Manhattan) affects the model's performance.



The results demonstrate that the choice of distance metric significantly impacts the model's performance.

The accuracy for the K-NN model with Euclidean distance starts at 0.855 for K=1. As we increase K, the accuracy decreases gradually, reaching 0.835 for K=2 and 0.84 for K=3. Beyond K=3, the accuracy further decreases as K increases. This pattern suggests that the Euclidean metric model favors a lower K value, where only the nearest neighbors have a more significant impact on the class estimation.

On the other hand, the model using Manhattan distance starts at a higher accuracy of 0.925 for K=1. The accuracy decreases slightly to 0.905 for K=2 but then goes back up to 0.920 for K=3. From there, as we increase K, the accuracy begins to decrease, but the decrease is slower and more linear compared to the model with the Euclidean metric.

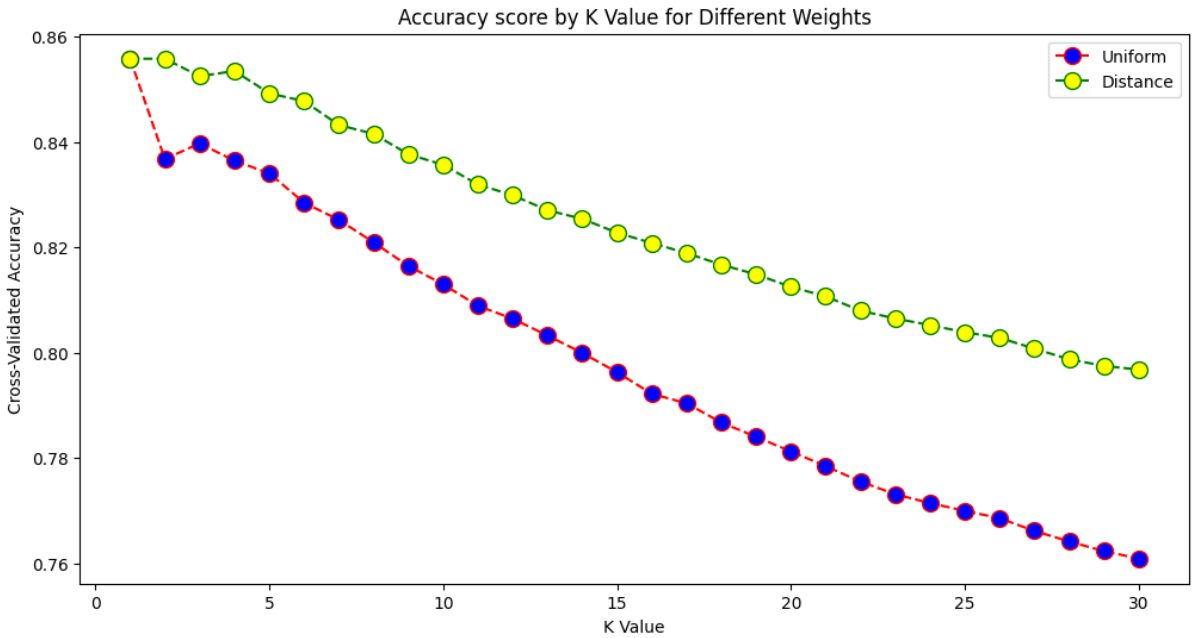
Comparing the two metrics, the Manhattan distance metric consistently delivers higher accuracy scores across different K values. Furthermore, the performance of the Manhattan metric appears to be more stable across a range of K values, as indicated by the more linear decrease in accuracy as K increases.

In summary, the Manhattan distance metric performs better for our dataset, providing higher accuracy across all K values tested. The more gradual decline in accuracy as K increases also suggests that the model using Manhattan distance may be more robust to the choice of K. However, similar to the model with the Euclidean distance, the optimal K value appears to be lower (K=1 or possibly K=3).

These observations suggest that the dataset's structure aligns better with the Manhattan distance computation, emphasizing differences along individual dimensions rather than the aggregate difference (as in the Euclidean distance).

5.2.4 Accuracy score by K Value for Different Weights

This is similar to the previous stage, but instead of distance metrics, we are now looking at different ways of weighing the votes of the neighbors. With 'uniform' weights, all neighbors have equal votes. With 'distance' weights, closer neighbors have more influence on the decision than farther ones.



Some conclusions are:

- For the K-NN model using distance weighting, the accuracy starts at 0.857 for K=1 and slightly increases to 0.858 for K=2. As we increase K beyond 2, the accuracy decreases, but at a slower pace, and occasionally, we see a slight increase in accuracy for some values of K. These results suggest that the distance-weighted model performs relatively well across a range of K values, although there's a slight preference for smaller K values.

- For the K-NN model with uniform weighting, the accuracy begins at 0.837 for K=1 and goes up a little to 0.84 for K=2. However, as we increase K beyond 2, the accuracy decreases more rapidly compared to the distance-weighted model. This suggests that the uniform-weighted model may be more sensitive to the choice of K, especially as K gets larger.

Comparing the two, the distance-weighted model consistently delivers higher accuracy across all K values tested. It also shows a more stable performance as K increases, suggesting it might be more robust to the choice of K compared to the uniform-weighted model.

Overall, the results suggest that weighting the votes by distance leads to better performance for our dataset. This finding indicates that in our dataset, the classes of closer neighbors are more likely to be similar than those of farther neighbors, making the 'distance' weight a more effective choice for our K-NN model.

5.2.5 Grid Search

Given that there are several hyperparameters to optimize, doing it manually can be time-consuming and may not even guarantee the best combination. Grid search is a hyperparameter tuning technique that can search across a specified grid of hyperparameters and find the best combination of hyperparameter values. In this stage, we are using grid search to find the best combination of K, distance metric, and weight function for our K-NN model.

Grid Search Best Score	Grid Search Best Parameters	Grid Search Best Estimator
0.9308536010217043	{'metric': 'manhattan', 'n_neighbors': 4, 'weights': 'distance'}	KNeighborsClassifier(metric='manhattan', n_neighbors=4, weights='distance')

The results of the grid search reveal that the best combination of hyperparameters for our K-NN model on this specific dataset are:

- Distance Metric: Manhattan
- Number of Neighbors (K): 4
- Weight Function: Distance
- This configuration results in a model with an accuracy score of approximately 0.931, which is a significant improvement from the initial models we explored.

These results provide several insights:

- The Manhattan distance metric outperforms the Euclidean metric in our case. This could be attributed to the characteristics of our dataset and the distribution of our data points. The Manhattan distance, calculated as the sum of absolute differences between the points, may be better suited for our high-dimensional data.

- The optimal number of neighbors is 4, suggesting that a small neighborhood leads to better predictions on our dataset. A smaller K reduces the risk of the model being overly generalized, and this value of K may balance the trade-off between bias and variance in our model.

- Lastly, weighting by distance ('distance') is the best option for our weight function, confirming our previous observation. This weighting scheme gives more importance to closer neighbors, which seems to align well with the distribution and nature of our data.

5.3 Decision Trees

We applied the default DecisionTreeClassifier from sklearn directly after preprocessing the data, without imposing any parameter constraints:

```
[[1615   24   14    2    3   18    2   47    5   13   13    7   11   20]
 [  26  815    5    3    7    8    6   11   10    7   20    1   14    5]
 [   4    5  812    4    1    4    1    5    0    7    9    4    4    2]
 [   7    7    3   34    0    1    5    6    0    7    1    1    7    2]
 [   5    4    2    0  925    2    2    2    2    3    9    1   17   15]
 [  18    6    7    5    2  793    1   25    3    9   10    4    6    3]
 [   3    4    2    3    0    2   48    0    6    7    9    1    4    3]
 [  37   18    7    3    5   27    0 1245    0   39    4    6    2    4]
 [   5   12    2    0    4    7    5    3  713    0    4    1    4   11]
 [   6    3    2    4    2    8    2   37    2 1841    7    2    4    5]
 [  12    9   11    4   10   12    0    4   13    3 1565    5   14   15]
 [   6    6    3    5    2    1    3    8    0    4    2  921   12   16]
 [  15   12    1    3    7    4    4    4   10    2    9    5 1640   13]
 [   7   14    4    6   16    4    5    3    5    0   16    9   18 1377]]

Accuracy: 0.9183098591549296

              precision    recall  f1-score   support

           0       0.91      0.90      0.91      1794
           1       0.87      0.87      0.87       938
           2       0.93      0.94      0.93       862
           4       0.45      0.42      0.43        81
           5       0.94      0.94      0.94       989
           6       0.89      0.89      0.89       892
           7       0.57      0.52      0.55        92
           8       0.89      0.89      0.89      1397
           9       0.93      0.92      0.93       771
          10       0.95      0.96      0.95      1925
          11       0.93      0.93      0.93      1677
          12       0.95      0.93      0.94       989
          13       0.93      0.95      0.94      1729
          14       0.92      0.93      0.93      1484

    accuracy                           0.92     15620
   macro avg       0.86      0.86      0.86     15620
weighted avg       0.92      0.92      0.92     15620
```

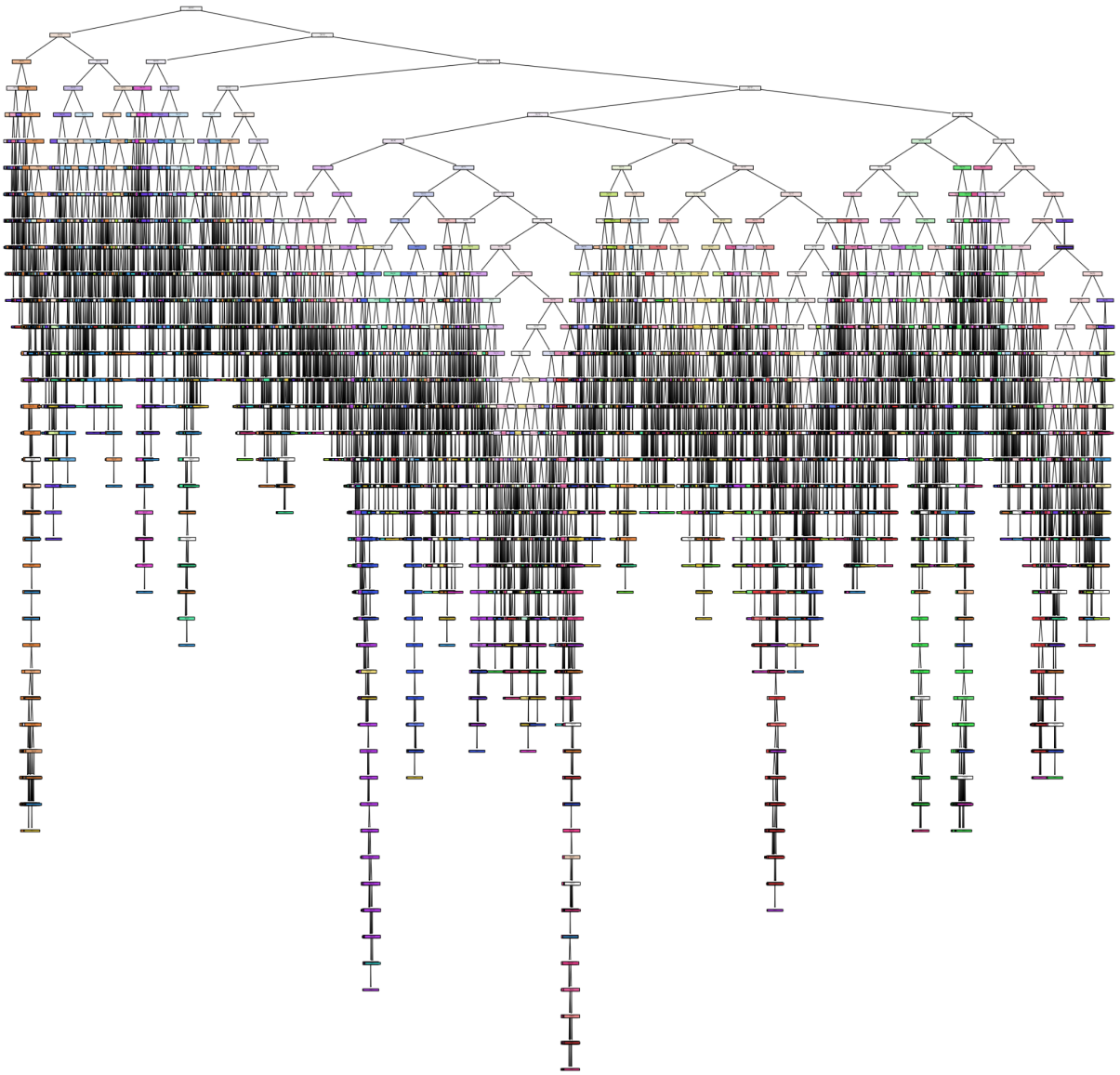
For our database and the prediction task at hand, we will prioritize accuracy as a more important metric since we are predicting postures, and values such as false positives and false negatives do not hold significant importance.

The accuracy of your classifier on the test data is approximately 0.91, which indicates that it is correctly predicting the postures for a significant portion of the samples.

The confusion matrix shows the number of samples that were classified into each posture category. It provides a more detailed view of the classifier's performance for each class. In this case, the majority of the samples are classified correctly, with some misclassifications in each posture category. For users 4 and 7, we can see that the classification is not very good. As we have analyzed in previous sections, we have very few samples for these two users.

In conclusion, if we do not consider users 4 and 7, which have a small number of samples, we can say that we have good performance.

Default version decision tree with 4297 leaves nodes:



Best parameters:
With the default parameters of the algorithm, we obtain a decision tree with thousands of nodes and a considerable depth. Although the accuracy percentage is quite high, It is always a good practice to find a way to simplify the decision tree. A simple tree helps us avoid issues like overfitting, and it is easier to interpret and compute.

We can play the following parameters (if you don't pass any parameter, it will run the algorithm with default values):

```
class DecisionTreeClassifier(
  *,
        criterion: Literal['gini', 'entropy', 'log_loss'] = "gini",
        splitter: Literal['best', 'random'] = "best",
        max_depth: Int | None = None,
        min_samples_split: float | int = 2,
        min_samples_leaf: float | int = 1,
        min_weight_fraction_leaf: Float = 0,
        max_features: float | int | Literal['auto', 'sqrt', 'log2'] | None = None,
        random_state: Int | None = None,
        max_leaf_nodes: Int | None = None,
        min_impurity_decrease: Float = 0,
        class_weight: Mapping | str | Sequence[Mapping] | None = None,
        ccp_alpha: float = 0
)
```

We have focus on the following parameters:

- **criterion**: The function to measure the quality of a split. It can be either "gini" for the Gini impurity or "entropy" for the information gain.
- **max_depth:** The max_depth parameter specifies the maximum depth or levels allowed for the decision tree.
- **min_samples_split:** The minimum number of samples required to be at a leaf node.
- **min_impurity_decrease:** A node will be split if this split induces a decrease of the
- impurity greater than or equal to this value
- **max_leaf_nodes:** The max_leaf_nodes parameter limits the maximum number of leaf nodes in the decision tree. It allows for controlling the complexity and size of the tree.

Applying Grid Search with the following parameters:

- params: {
  'criterion': ['entropy','gini'],
  'min_impurity_decrease': [0.0, 0.01, 0.02, 0.05],
  'min_samples_split': [2, 5, 10, 20],

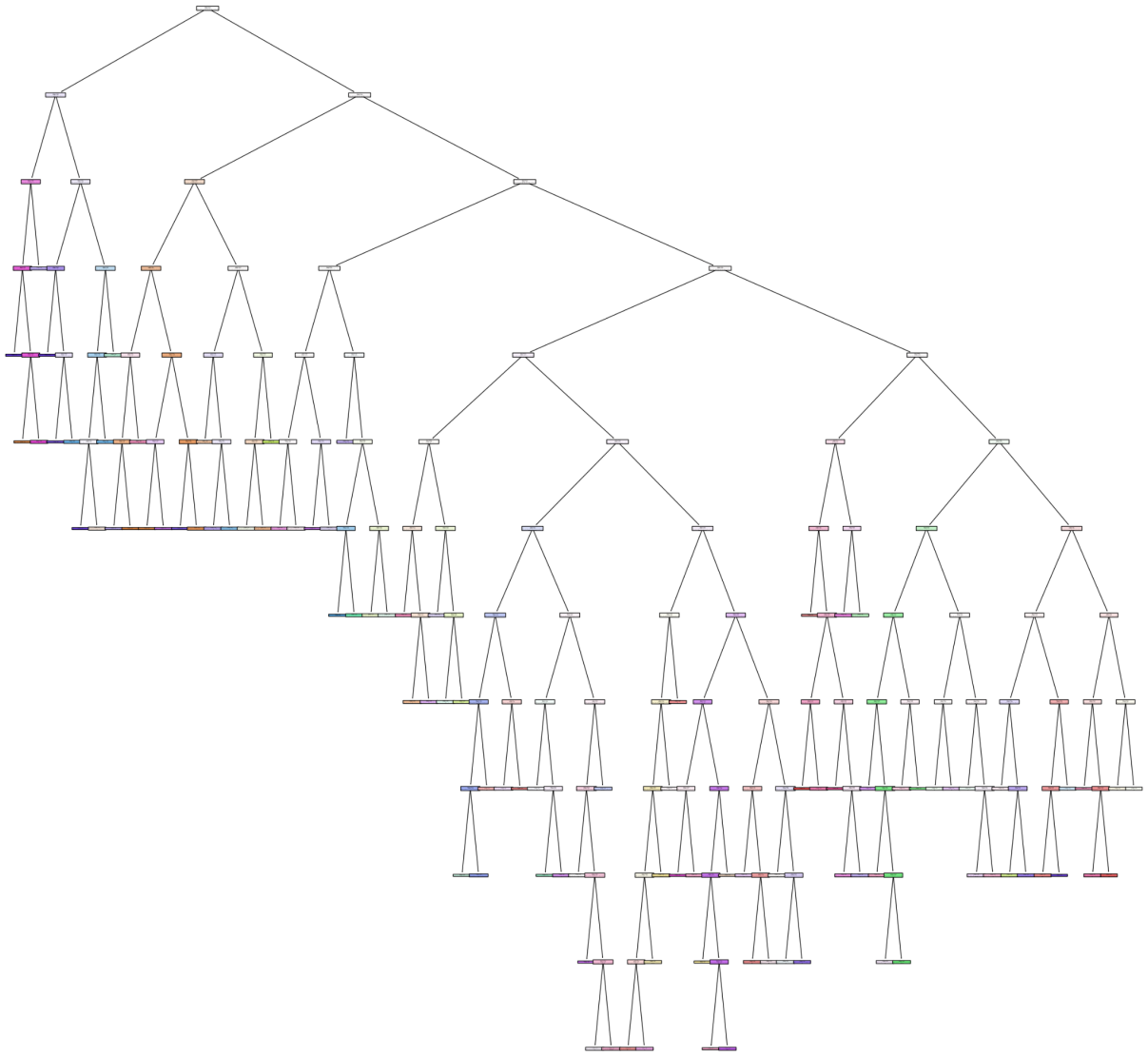
```
    'max_depth': [10, 20, 30, 50],
    'max_leaf_nodes': [20, 40, 60, 80, 100]
  }
```

The best accuracy value is using this values of parameters:

- 'criterion': entropy
- 'max_depth': 20,
- 'max_leaf_nodes': 100,
- 'min_impurity_decrease': 0.0,
- 'min_samples_split': 2

```
[[ 880   49  156    0   12   64    0   66    0  255   80   55  130   47]
 [  20  369   13    0   31   32    0   14   50   82  130   11  106   80]
 [  39   58  479    0   38   89    0   37    8   19   68    9    5   13]
 [  10    7   10    0    0    7    0    4    0    8    7    8   10   10]
 [   6    8   20    0  611    0    0    0   10   24   54   31  140   85]
 [  14   15   48    0    0  365    0   61  155   57   89    0   77   11]
 [  12   15    4    0    0    6    0    0   13   11   10    7   12    2]
 [ 106    4   62    0   16  109    0  639    0  306   55   19   44   37]
 [   0    1    0    0    3   15    0    5  555    3   74    2  105    8]
 [  93    0   27    0    9   33    0   80    8 1493   81   29   54   18]
 [   4   59   19    0   75    8    0   17   73   20 1130    9  169   94]
 [  50   16    0    0    5    0    0    4    1   64   20  759   44   26]
 [   2   64   18    0   66   39    0    8   23   39  133   40 1232   65]
 [   0   30   14    0   99    0    0    3   81    9  114   22  206  906]]

Accuracy: 0.6029449423815622

              precision    recall  f1-score   support

           0       0.71      0.49      0.58      1794
           1       0.53      0.39      0.45       938
           2       0.55      0.56      0.55       862
           4       0.00      0.00      0.00        81
           5       0.63      0.62      0.63       989
           6       0.48      0.41      0.44       892
           7       0.00      0.00      0.00        92
           8       0.68      0.46      0.55      1397
           9       0.57      0.72      0.64       771
          10       0.62      0.78      0.69      1925
          11       0.55      0.67      0.61      1677
          12       0.76      0.77      0.76       989
          13       0.53      0.71      0.61      1729
          14       0.65      0.61      0.63      1484

    accuracy                           0.60     15620
   macro avg       0.52      0.51      0.51     15620
weighted avg       0.60      0.60      0.59     15620
```

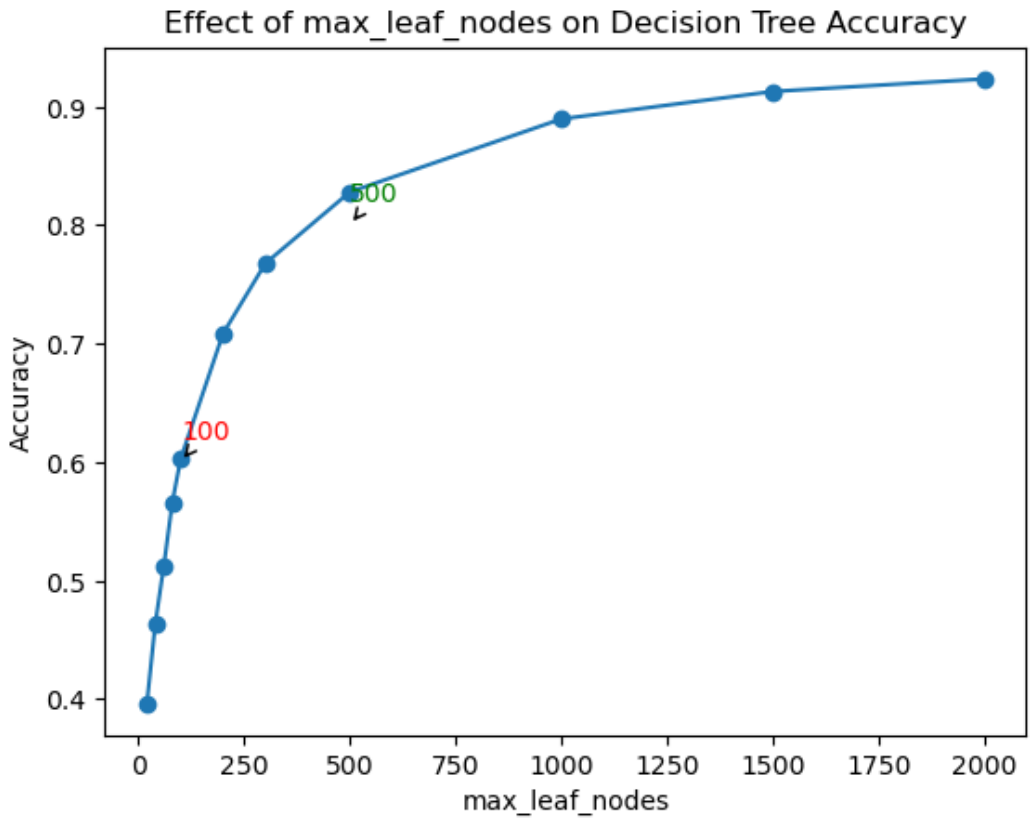
We can see that the accuracy has decreased compared to the default version, from 0.91 to 0.60. However, we believe it is worth it because in the previous version, we had a huge tree with approximately 8000 nodes, out of which 4000 were leaves. Now, it is reduced to 199 nodes, out of which 100 are leaves:



Now, based on the GridSearch we performed, we will focus on studying some parameters that we consider more important and try to improve the last version, as an accuracy of 0.60 seems quite low compared to the default version.

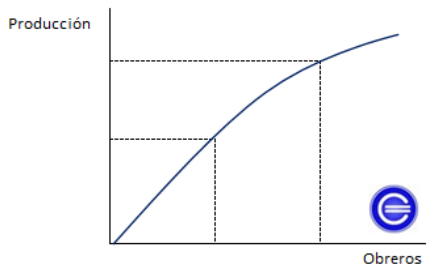
From the GridSearch results, we can see that 'min_impurity_decrease' and 'min_impurity_decrease' have taken the smallest values we provided. On the other hand, 'max_depth' and 'max_leaf_nodes' have increased, with 'max_leaf_nodes' even taking the highest value we provided. We have decided to focus the study on these last two parameters.

Effect of parameter 'max_leaf_nodes':



We have obtained a quite interesting graph that follows the law of diminishing returns: increasing the quantity of a productive factor in the production of a good or service leads to a decrease in the productivity as we increase this factor.

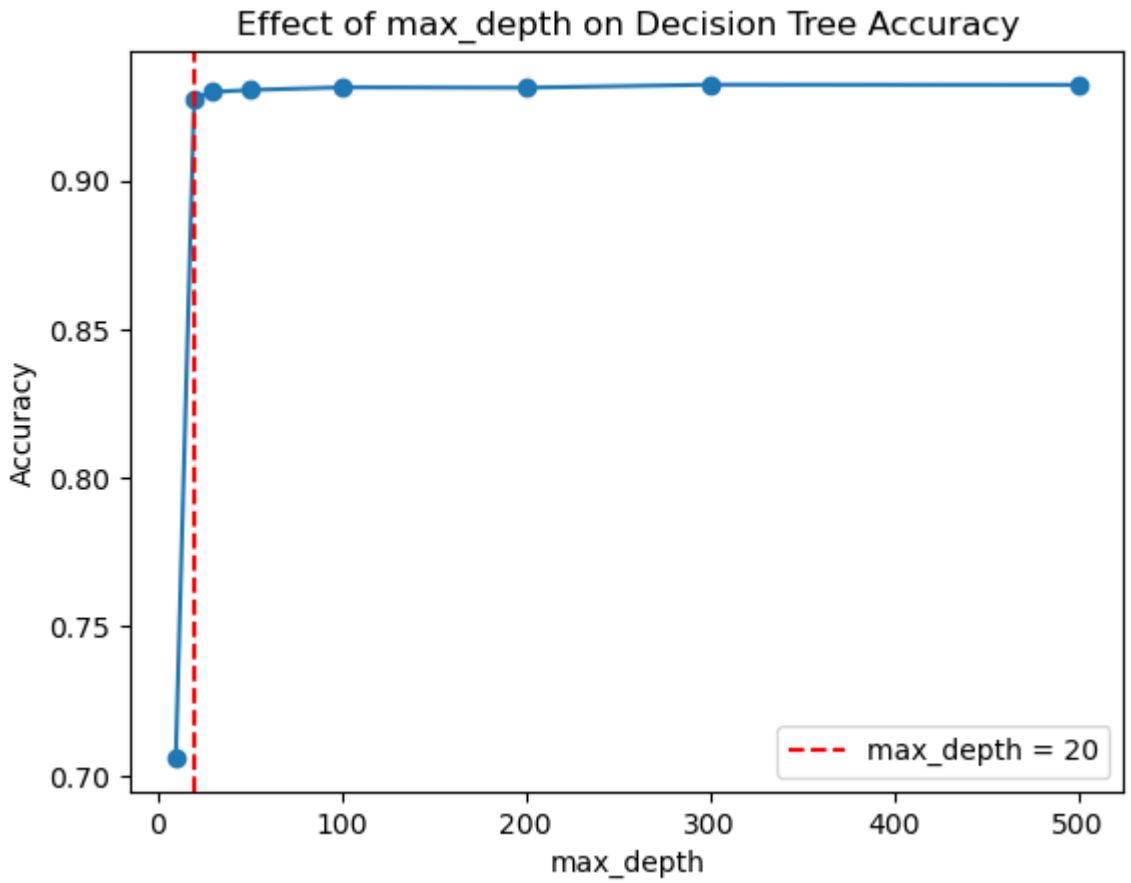
If we increase the quantity of a productive factor while keeping the quantity of the remaining factors fixed, there will come a point where the quantity of the final product obtained decreases as we continue to produce more and more. An example of this is the relationship between production and workers.



Therefore, as capitalists, we are interested in having the fewest number of workers (leaf nodes) to maintain high production (accuracy). In other words, we aim to maintain productivity. We can observe that for an accuracy of 0.6, it is necessary to have a minimum of 100 max_leaf_nodes, which aligns with the previous result. However, to achieve an

accuracy of 0.8, we need around 500 max_leaf_nodes. There is a clear decrease in productivity in this case.

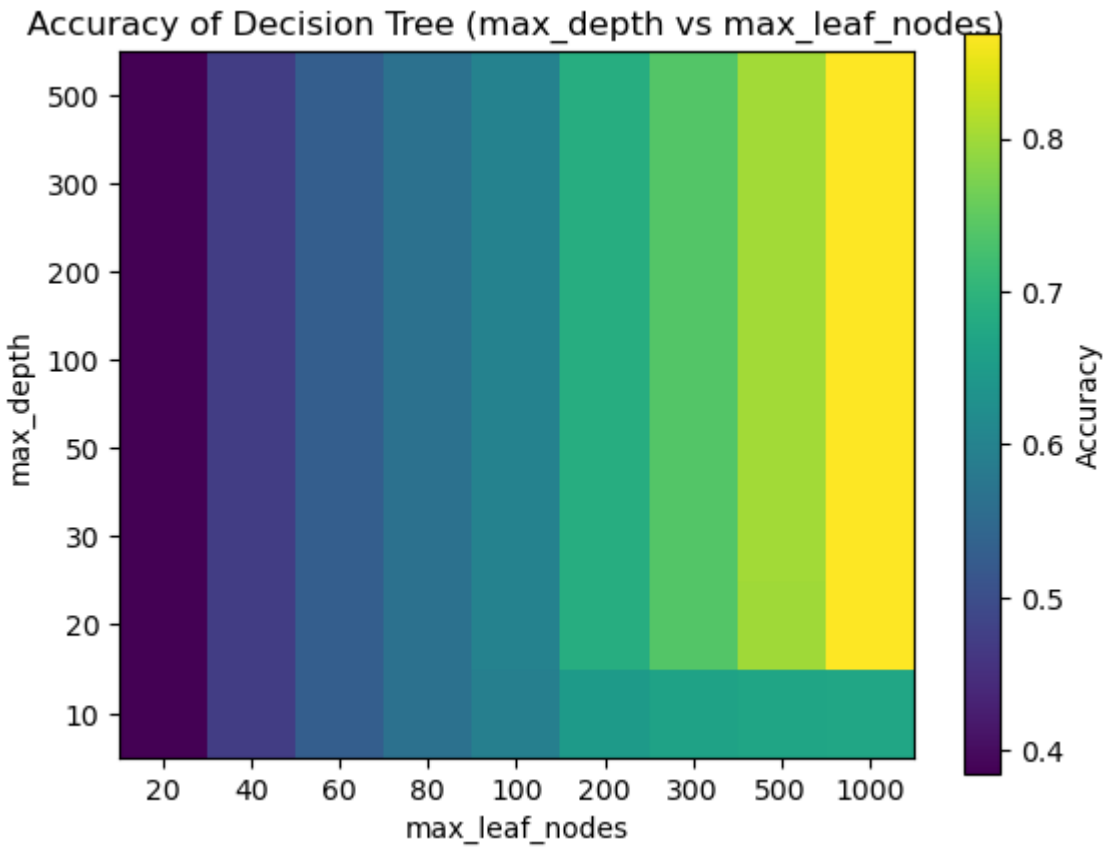
Effect of parameter 'max_depth':



Regarding max_depth, we can see that when it has a value of 20, it can reach an accuracy of over 0.90. It's important to remember that in this case and the previous one, we didn't impose any restrictions on max_leaf_nodes. Therefore, here we are representing the highest accuracy we can achieve without limiting the number of nodes, specifically for max_depth=20.

Next, we will study the behavior of accuracy in relation to max_depth and max_leaf_nodes using a heatmap.

Effect of parameter 'max_leaf_nodes' and 'max_depth':

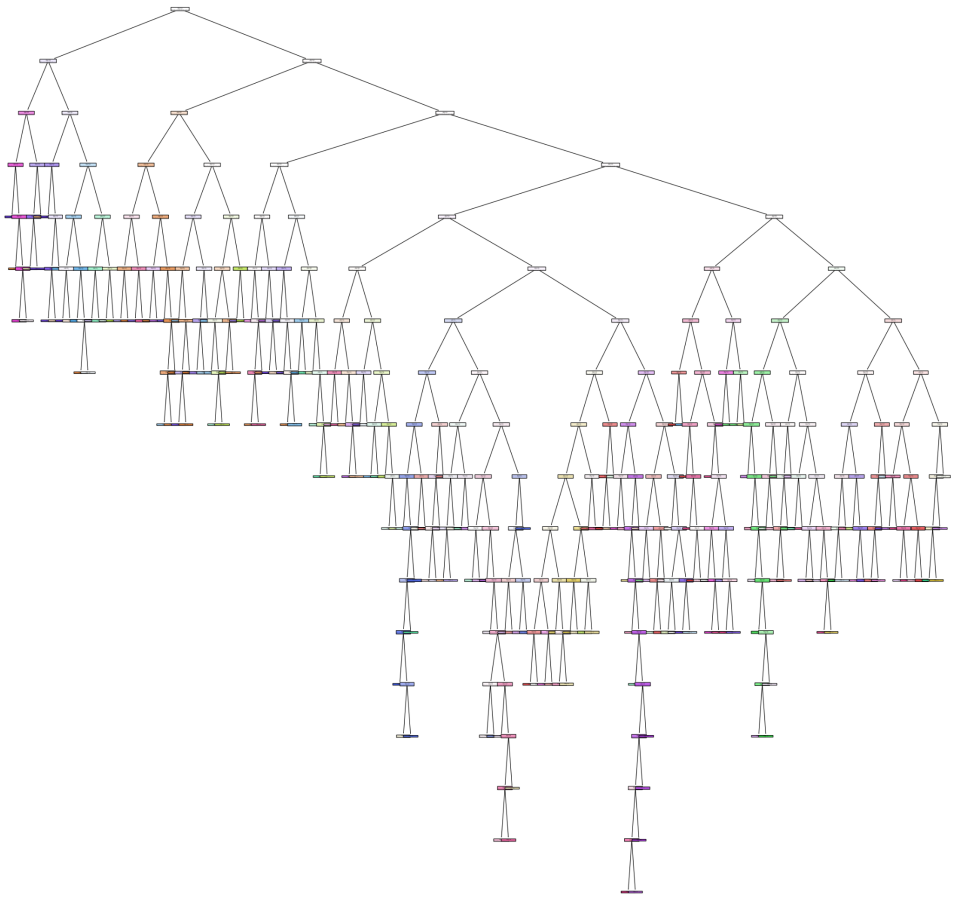


In this heatmap, we can see that the parameter that truly affects the accuracy is 'max_leaf_nodes'. Without changing the value of this parameter, just altering the max_depth barely results in any accuracy variation, which makes sense since without nodes, we cannot expand the tree. The highest value of max_depth with effects on accuracy is 20. Knowing this, we can now generate the decision tree with the desired accuracy, taking into account the graph of the effect of the 'max_leaf_nodes' parameter.

For example if we want a desicion tree with 0.7 of accuracy:

- max_leaf_node = 200
- max_depth = 20

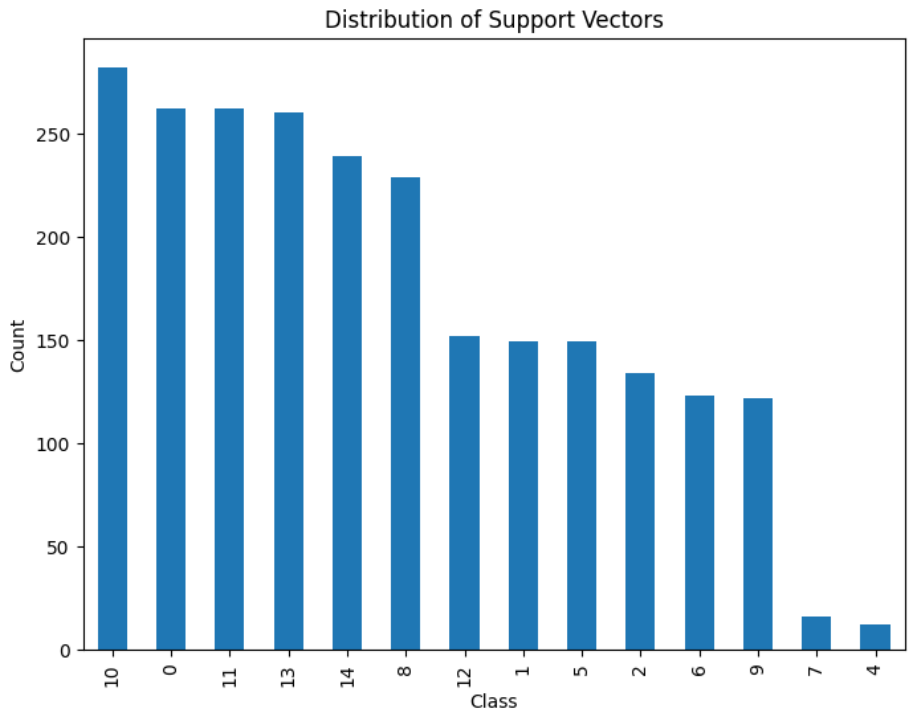
```
[[1164   36  129    0   16   90    7  123    0   43   39   54   63   30]
 [  47  467   16    0   17   63    0   25   11   13  135   12   79   53]
 [  16   23  632    0   19   35    3   34    4   11   55    2   10   18]
 [  14    6   13    0    0    7    1    6    0    7    6    8    6    7]
 [  10   24   20    0  704    0    0    3    0    8   38   10  102   70]
 [  20   32   32    0    5  565    0   92   13   26   47    1   52    7]
 [  17   22    2    0    1    8    7    2    3    9   10    1    7    3]
 [  85   10   19    0   13   75    0  895    0  207   54   20   11    8]
 [   3   21    0    0   10   44    0    5  506    0   98    1   77    6]
 [  92    6   17    0    7   23   11  125    3 1502   51   21   30   37]
 [  20   26   30    0   26    3    0   18   13    5 1343    3   87  103]
 [  63   21    4    0   11    5    1    2    0   32   10  798   26   16]
 [   4   56   30    0   37    1    7    1    9   10   92   17 1416   49]
 [   3   34   15    0  100    0    0   11   19   10   96   11  117 1068]]

Accuracy: 0.7085147247119078

              precision    recall  f1-score   support

           0       0.75      0.65      0.69      1794
           1       0.60      0.50      0.54       938
           2       0.66      0.73      0.69       862
           4       0.00      0.00      0.00        81
           5       0.73      0.71      0.72       989
           6       0.61      0.63      0.62       892
           7       0.19      0.08      0.11        92
           8       0.67      0.64      0.65      1397
           9       0.87      0.66      0.75       771
          10       0.80      0.78      0.79      1925
          11       0.65      0.80      0.72      1677
          12       0.83      0.81      0.82       989
          13       0.68      0.82      0.74      1729
          14       0.72      0.72      0.72      1484

    accuracy                           0.71     15620
   macro avg       0.63      0.61      0.61     15620
weighted avg       0.71      0.71      0.70     15620
```



5.4 Support Vector Machines

In order to run the model, we had to speed it up, because the amount of data was too much to handle in a sensible amount of time. That is why we chose to do random subsampling to drastically reduce the training time, this could unfortunately reduce the quality of our results. In terms of our kernel and parameter choice, We tried many different methods with a grid search for rbf and sigmoid kernels, which turned out to be the fastest. But for the linear and polynomial they took too long to try many different methods and were also not a very good fit for the problem at hand since the problem does not seem to be linear.
These were the best parameters of all:
{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
The parameters we tried were:
C: 1,10,100,1000
Gamma: 1, 0.1, 0.001, 0.0001
Kernel: rbf, sigmoid

In our case there are 2383 support vectors, which implies that a considerable number of samples played a significant role in defining the decision boundaries. Because of this number of support vectors being too high, it is challenging to interpret them individually or extract meaningful insights of each sample.
The following is the distribution of Support Vectors for each class.



The disparity of the support vectors for each class could indicate a few things. If there are a higher number of support vectors for a class, it may indicate that there is a class imbalance in the training data. It could also mean that those classes with more support vectors are more complex or challenging to separate accurately.

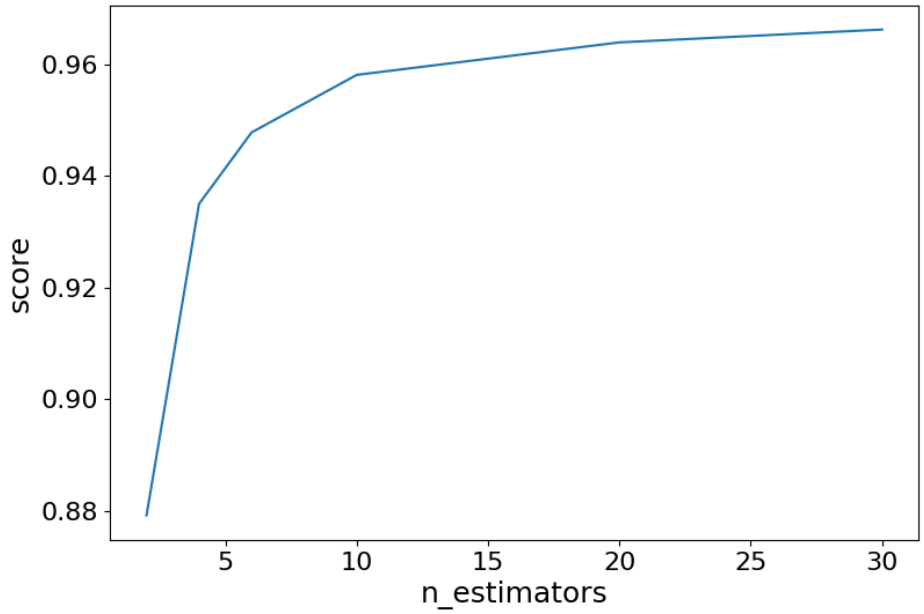
5.5 Meta-learning algorithms

	acc	prec	recall	test f1 score (W)
majority_vote 2	0.989821	0.989856	0.989821	0.949099
AdaBoost	0.989757	0.989702	0.989757	0.950960
Random Forest	0.982714	0.982614	0.982714	0.926240
Bagging	0.966197	0.965651	0.966197	0.915705
majority_vote 1	0.917990	0.917158	0.917990	0.857498

5.5.1 Bagging

One of the most important parameters in this algorithm is:

- base_estimator: It defines the base estimator to fit on random subsets of the dataset.
- n_estimators: The number of base estimators in the ensemble.

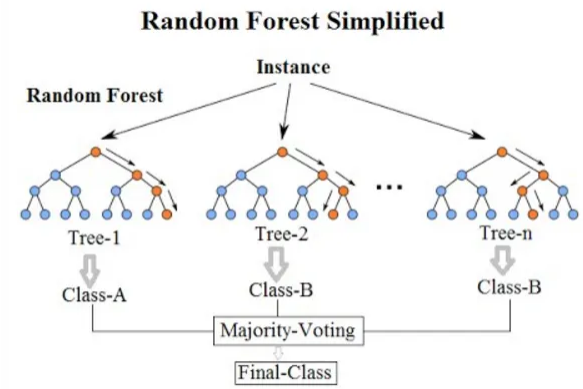


In the graph above it can be seen that as the "n_estimator" parameter increases, the accuracy also increases, but it tends to be asymptotic and settle around 96.5%. As "n_estimator" increases, the computational cost also increases, so in this case I decided to stop at "n_estimator = 30"

BaggingClassifier(n_estimators = 30)

acc	prec	recall	test f1 score (W)
0.966197	0.965651	0.966197	0.915705

5.5.2 RandomForest



There are many possible parameters in a RandomForestClass, for this reason doing a gradsearch on all of these would have required a considerable amount of time, for this reason, the grid search was performed on the following three parameters, obtaining a clear performance increase compared to the default ones.

- **n_estimators:** It defines the number of decision trees to be created in a random forest. Generally, a higher number makes the predictions stronger and more stable, but a very large number can result in higher training time.
- **max_features:** maximum number of splits thats required for split in each decision tree.
- **max_depth:** maximum depth of the decision trees.

RandomForestClassifier(max_depth=200, max_features=4, n_estimators=200)

acc	prec	recall	test f1 score (W)
0.982714	0.982614	0.982714	0.926240

5.5.3 Adaboost

In AdaBoost, Multiple sequential models are created, each correcting the errors from the last model. AdaBoost assigns weights to the incorrect predicted and the subsequent model works to predict these values correctly. As base_estimator I used DecisionTreeClassifier as it is computationally more light and with best results those not based on "multi classifications". The other parameters were decided by grid search.

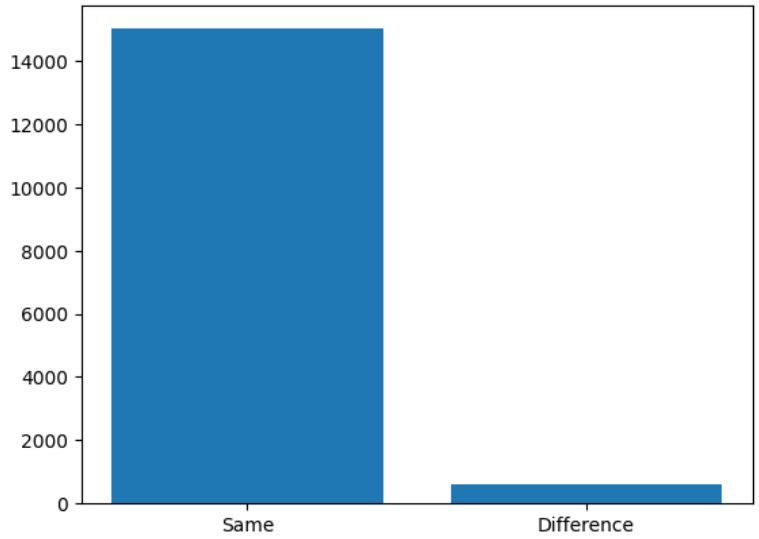
1. **base_estimators**: specifies the base estimator.
2. **n_estimators**: the number of estimators, default is 10 but keeping the value higher gives better performance
3. **learning_rate**: parameter that contributes to the estimator of final combination

AdaBoostClassifier(n_estimators=200,estimator = decision_tree, learning_rate=1)

acc	prec	recall	test f1 score (W)
0.989757	0.989702	0.989757	0.950960

5.5.4 Performance Majority Voting

Considering the excellent level of accuracy of the previous models, using this method is inefficient, in fact, as can be seen from the graph below, the values in which the three classifiers used disagree are very few,and mistakenly predict the same patterns, therefore there is little independence between individual classifiers. In particular I used Adaboost, RandmForest and Bagging, previously trained.



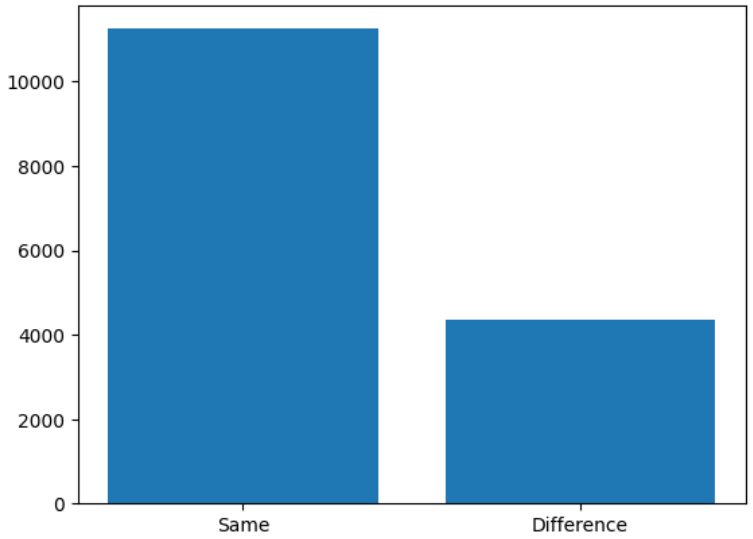
Can see a slight improvement over AdaBoost which had the best score, but not enough to justify its use.

acc	prec	recall	test f1 score (W)
0.989821	0.989856	0.989821	0.949099

A version of lesser performance, but more effective, was found, three models were trained:

knn -> full dataset
DecisionTree -> half dataset
DecisionTree -> half dataset

	acc	prec	recall	test f1 score (W)
Decision_tree2	0.869206	0.868029	0.869206	0.796537
KNN	0.864661	0.863969	0.864661	0.796469
Decision_tree1	0.862100	0.860431	0.862100	0.778553



It can be seen how in this case the independence between the classifiers is very good, so as to increase the accuracy by about 6% bringing an average of the classifiers of 86.5% to 92%.

$$P_N = \sum_{i=m}^{N} \left(\frac{N!}{(N-i)!i!} \right) (p)^i (1-p)^{N-i}$$

According to Condorcet's Jury Theorem, I should have arrived at a theoretical percentage of about 94%, considering that there is no total independence between classifiers, and that it is a real problem, having obtained 92% is consistent.

acc	prec	recall	test f1 score (W)
0.917990	0.917158	0.917990	0.857498

6. Comparison

After evaluating the different data mining methods on the validation dataset, we can now discuss the results and determine the best method for testing on the validation data set. To facilitate the comparison, let's create a comparative table summarizing the performance metrics of each method:

	acc	prec	recall	test f1 score (W)

majority_vote 2	0.989821	0.989856	0.989821	0.949099
AdaBoost	0.989757	0.989702	0.989757	0.950960
Random Forest	0.982714	0.982614	0.982714	0.926240
Bagging	0.966197	0.965651	0.966197	0.915705
KNN	0.937324	0.936993	0.937324	0.867506
Decision Tree	0.918	0.92	0.92	0.92
majority_vote 1	0.917990	0.917158	0.917990	0.857498
SVM	0.670968	0.741763	0.670968	0.563378

KNN:

Simple Cross-Validation	84.3%
10-fold-Cross-Validation	90%
Euclidean Distance	93%
Manhattan Distance	85.5%
Uniform Weight	85.8%
Distance Weight	84%
Grid Search	93%

Upon analyzing the results, it is evident that AdaBoost outperforms other methods across all metrics, exhibiting the highest accuracy, precision, recall, and test F1 score.

Comparing the accuracy of the validation dataset to the results obtained through cross-validation, we can observe that the trends are generally similar. However, there are some notable differences for certain methods. The SVM model, for instance, demonstrates a substantial drop in performance when tested on the validation dataset compared to

cross-validation. This discrepancy could be attributed to the SVM model's sensitivity to the specific distribution of data in the validation set, leading to poor generalization.
SVM Validation Accuracy: 0.6693548387096774, Cross-Validation Accuracy: 0.4419354838709677

Considering the overall performance, the best method among those evaluated appears to be AdaBoost. It consistently achieves the highest scores in all metrics, indicating its effectiveness in handling the given dataset. AdaBoost's ability to combine weak learners and improve their performance through iterative learning iterations might be contributing to its success.

7. General conclusions

1. **Data Preprocessing and Feature Extraction:**

The project successfully implemented data preprocessing and feature extraction techniques to prepare the dataset for machine learning models. The data was analyzed for correlation, and scatter plots were used to visualize the relationship between different variables.
Some important conclusions for this part are:

- High correlation (close to 1 or -1) between two variables suggests they change in relation to each other. If the correlation between two variables is high, they may be carrying similar information, leading to consideration of whether both variables need to be retained.

- The 'Z' coordinates (Z0, Z1, Z2, ...) are highly correlated with many other variables, indicating that they might share a lot of common information with those variables. This could potentially be due to the 3D nature of the data, as changes in 'Z' coordinates might often coincide with changes in other dimensions.

- The 'X11' variable also exhibits high correlation with other variables, suggesting that this marker's position might be particularly influential or responsive to hand postures.

- On the other hand, 'Z11' shows a relatively low correlation with most other variables. This could mean that the 'Z' coordinate of the 11th marker is more independent and does not change in tandem with other variables.

- The scatter plots for pairs X0-Y0 through to X8-Y8 show well-grouped points with few outliers. This implies that these coordinates generally move together in a predictable manner, suggesting a strong correlation between these particular pairs.

- From pairs X9-Y9 and X10-Y10, the points start to scatter a bit more, indicating a weaker correlation between these pairs. This could suggest that the corresponding markers are on parts of the hand that move more independently.

- The pair X11-Y11 shows no clear pattern in the scatter plot, indicating a weak or no correlation between these coordinates.

- There is a weak negative correlation between the X0 and Y0 positions as well as between the X0 and Z0 positions. This could suggest that certain movements or postures that involve moving the hand to the right (increasing X0) might slightly involve lowering the hand (decreasing Y0) or moving the hand backwards (decreasing Z0).

- Correlations found in scatter plots don't imply causation. Further analysis would be needed to determine whether a change in one variable causes a change in another.

2. **Machine Learning Models:**

Several machine learning algorithms were applied in this project, including Naive Bayes, KNN, SVM, Decision Trees, Bagging, Random Forest, and AdaBoost. Each of these models was tuned and evaluated for their performance.
The conclusions for each one are:

- Naive Bayes: The Naive Bayes model achieved an accuracy of 0.67 on the test data. It was noted that the model's performance could be improved by tuning the smoothing parameter.

- KNN: The KNN model achieved an accuracy of 0.93 using the Euclidean distance metric and 0.855 using the Manhattan distance metric. The model's performance was further improved to 0.93 after grid search optimization.

- SVM: The SVM model achieved an accuracy of 0.67 on the test data. However, it was noted that the model's performance dropped significantly when tested on the validation dataset compared to cross-validation, indicating poor generalization.

- Decision Trees: The Decision Tree model achieved an accuracy of 0.91 on the test data. The model was further optimized by tuning parameters like 'max_leaf_nodes' and 'max_depth', which significantly affected the accuracy.

- Bagging: The Bagging model achieved an accuracy of 0.966. The model's performance was influenced by the 'n_estimator' parameter, with the accuracy increasing as 'n_estimator' increased.

- Random Forest: The Random Forest model achieved an accuracy of 0.982. The model's performance was optimized by tuning parameters like 'n_estimators', 'max_features', and 'max_depth'.

- AdaBoost: The AdaBoost model outperformed other models across all metrics, achieving the highest accuracy, precision, recall, and test F1 score. The model was noted for its effectiveness in handling the given dataset and its ability to combine weak learners and improve their performance through iterative learning iterations.

- Meta-learning Algorithms: Meta-learning algorithms like Majority Voting were also evaluated. However, it was found that there was little independence between individual classifiers, making the use of this method inefficient.

- Overall Performance: Considering the overall performance, AdaBoost was identified as the best method among those evaluated. It consistently achieved the highest scores in all metrics, indicating its effectiveness in handling the given dataset.

Task Completion

Participant	Alex	Pol	Tommaso	Eric	He
Selecting dataset and inspect it	X	X	X	X	X
Description of the original data	X			X	X
Description of pre-processing of data	X			X	X
Univariate Analysis					X
Bivariate Analysis	X				
Naïve Bayes				X	
K-NN	X				
Decision Trees					X
Support Vector Machines		X			
Meta-learning algorithms			X		
Evaluation criteria of data mining models		X			
Final comparison and conclusions		X			
Document (word style)	X	X	X	X	X
PowerPoint	X	X	X	X	X
Organization, file structure		X			
Video	X	X	X	X	X