Table of Contents

Introduction

```
In this project i'm experiencing with couple of machine learning
algorithms:
  - Classic ones:
    - Random forest
    - Support Vector Machines
    - Extreme Gradient Boosting trees
  - Neural networks:
    - Multi Layered Perceptron
    - Convolutional Neural Networks
    - Long Short-Term Memory Networks
```

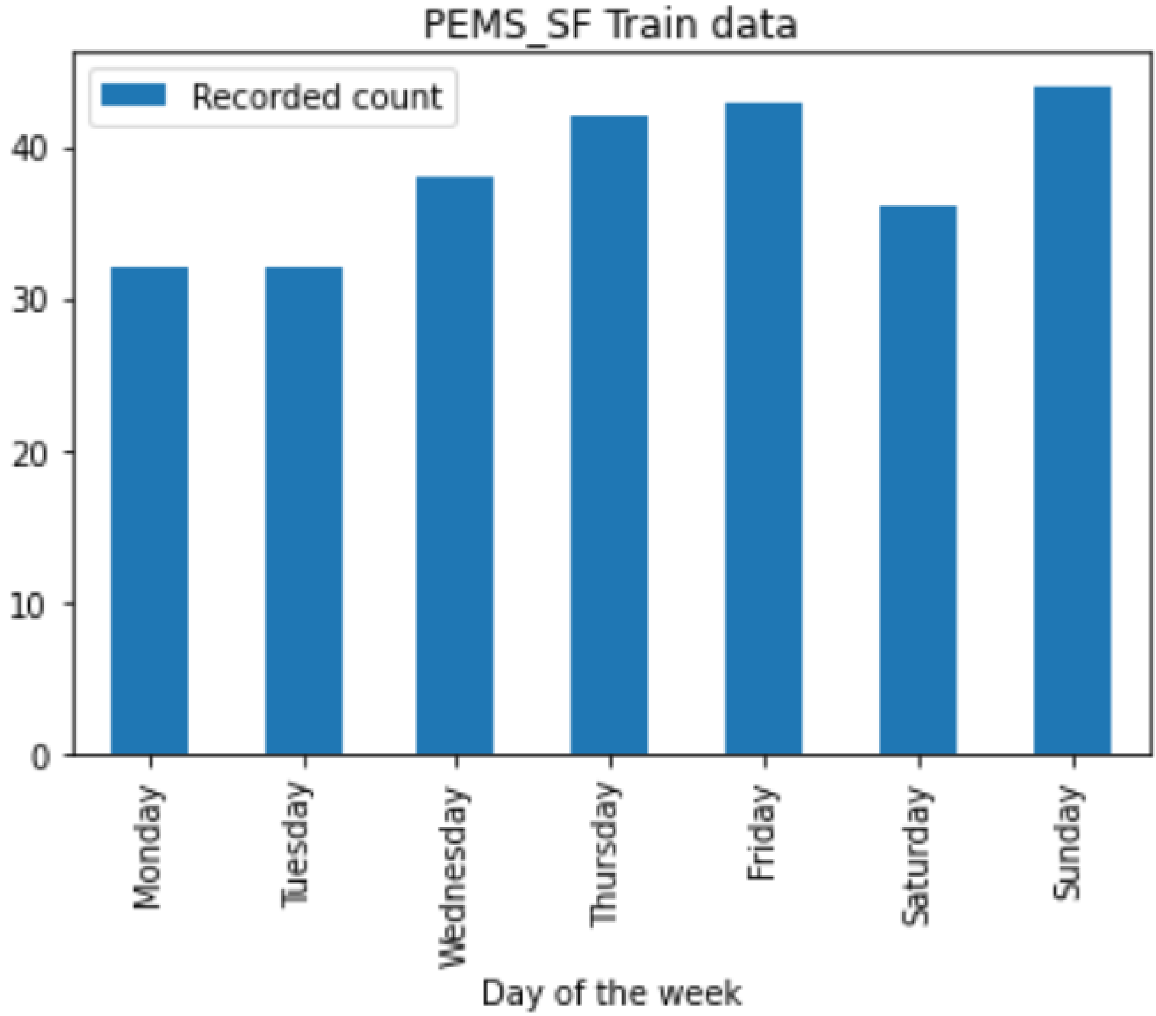
```
These algorithms are used on two datasets:
  - PEMS-SF
    - It contains information about days of the week activity (7 days of
the week).
    - A row contains 963 columns (963 sensors)
    - A dataset element represents a sensor activity (chosen column) in
that particular day (chosen row)

  - U Wave Gesture
    - It contains information acquisitioned by accelerometers.
    - Each row represents a different gesture (We have 8 gestures in total)
    - Each row contains 3 columns (3 axis evolution: ox, oy, oz)
    - A dataset element represents a particular axis evolution in time
(chosen column) for a particular gesture (chosen row)
```

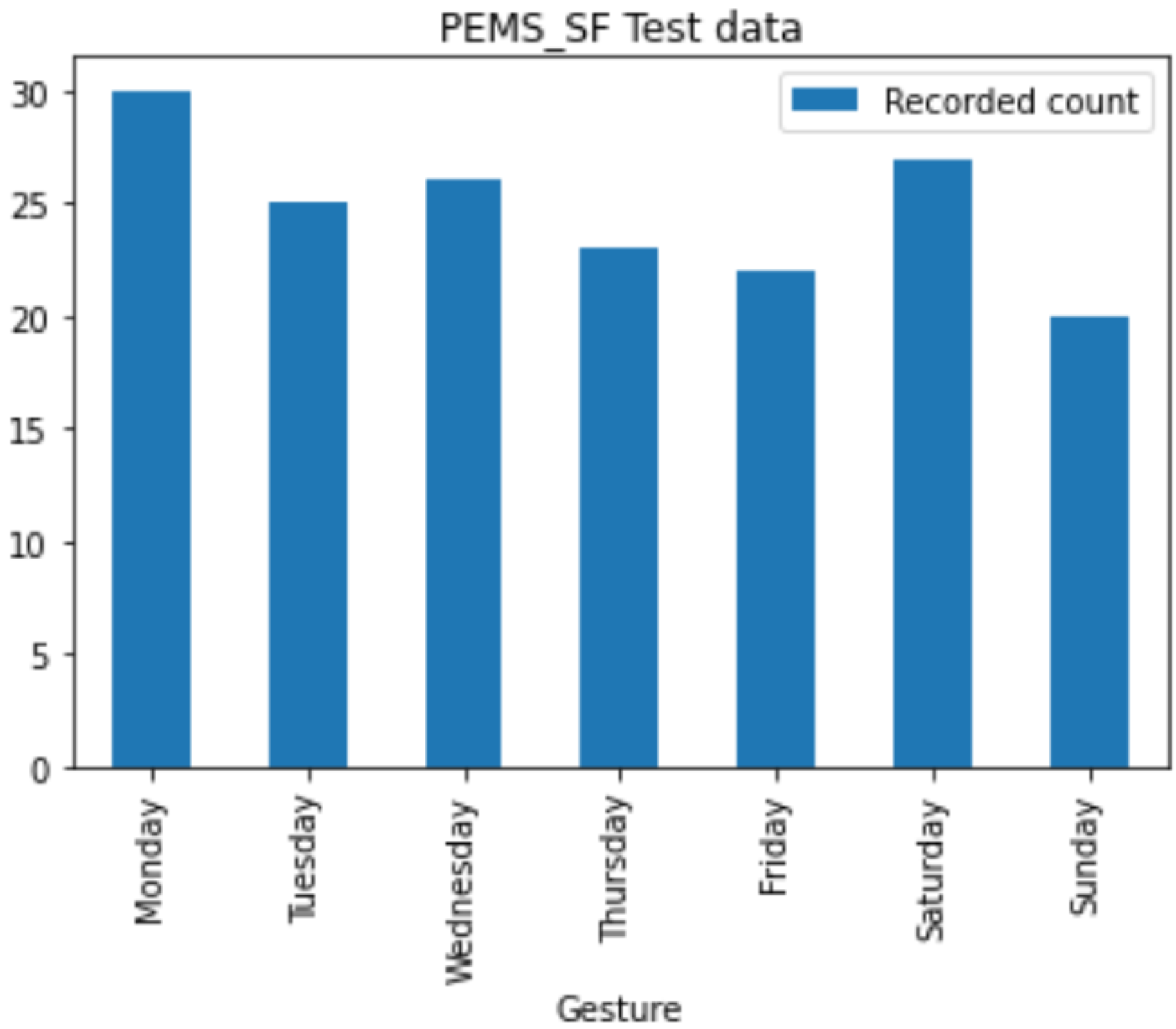
Exploratory Data Analysis

Class Equilibrium

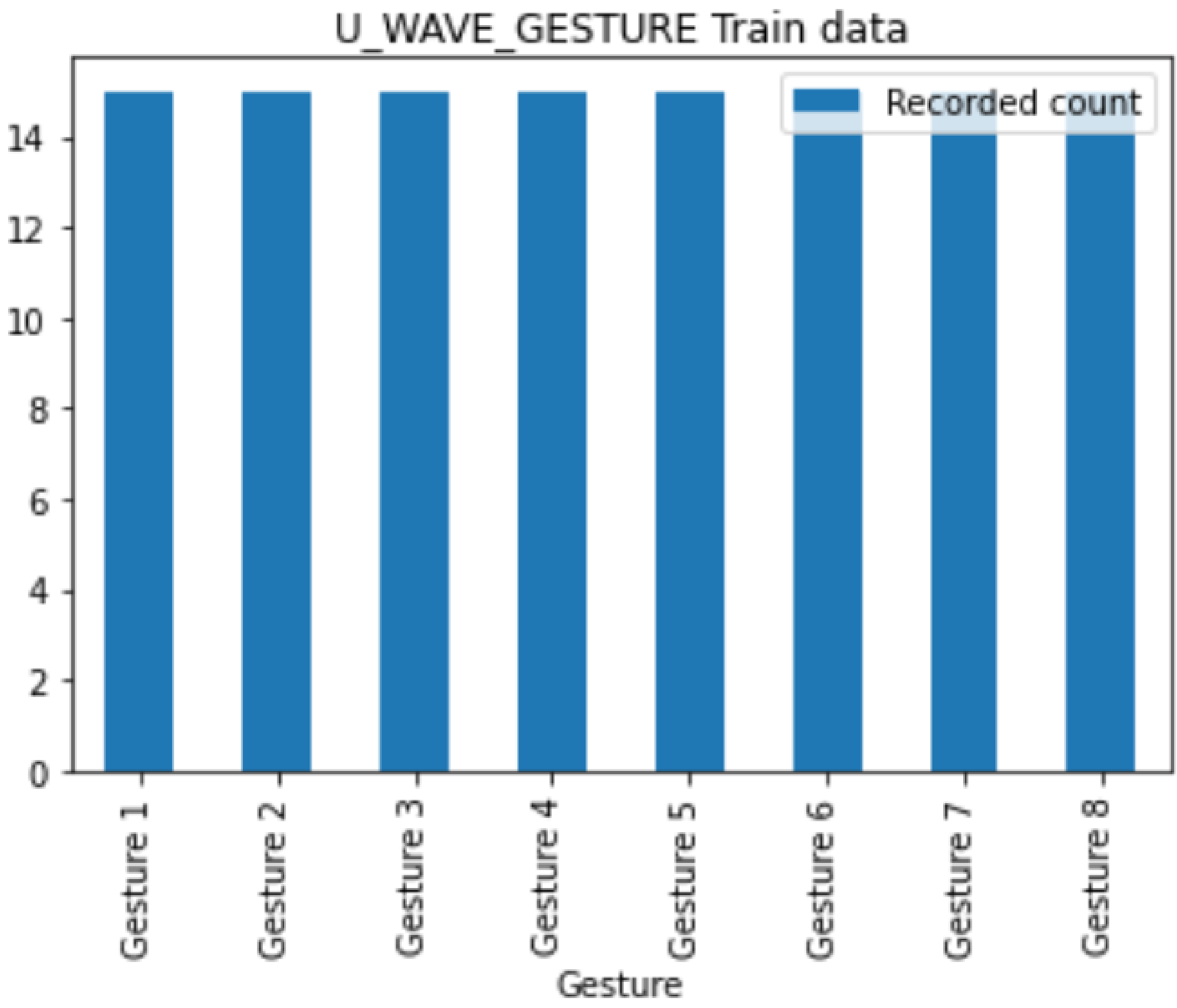
```
We observe that classes are balanced.
The training process will not be affected by an imbalance in the train
data.
```



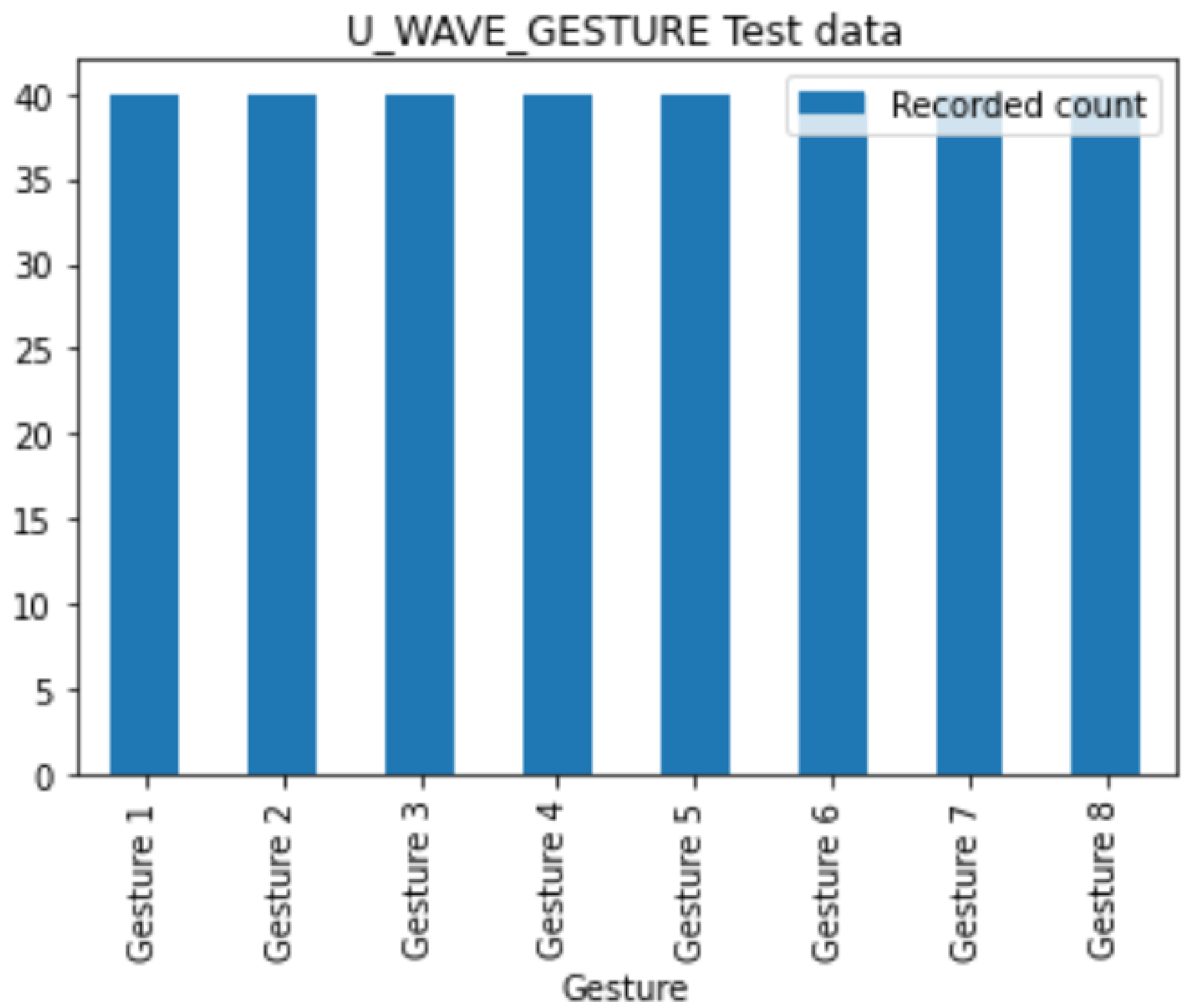

```
We observe that classes are balanced.
Due to this fact, the testing process will produce relevant results.
```




```
We observe that classes are balanced.
The training process will not be affected by an imbalance in the train
data.
```



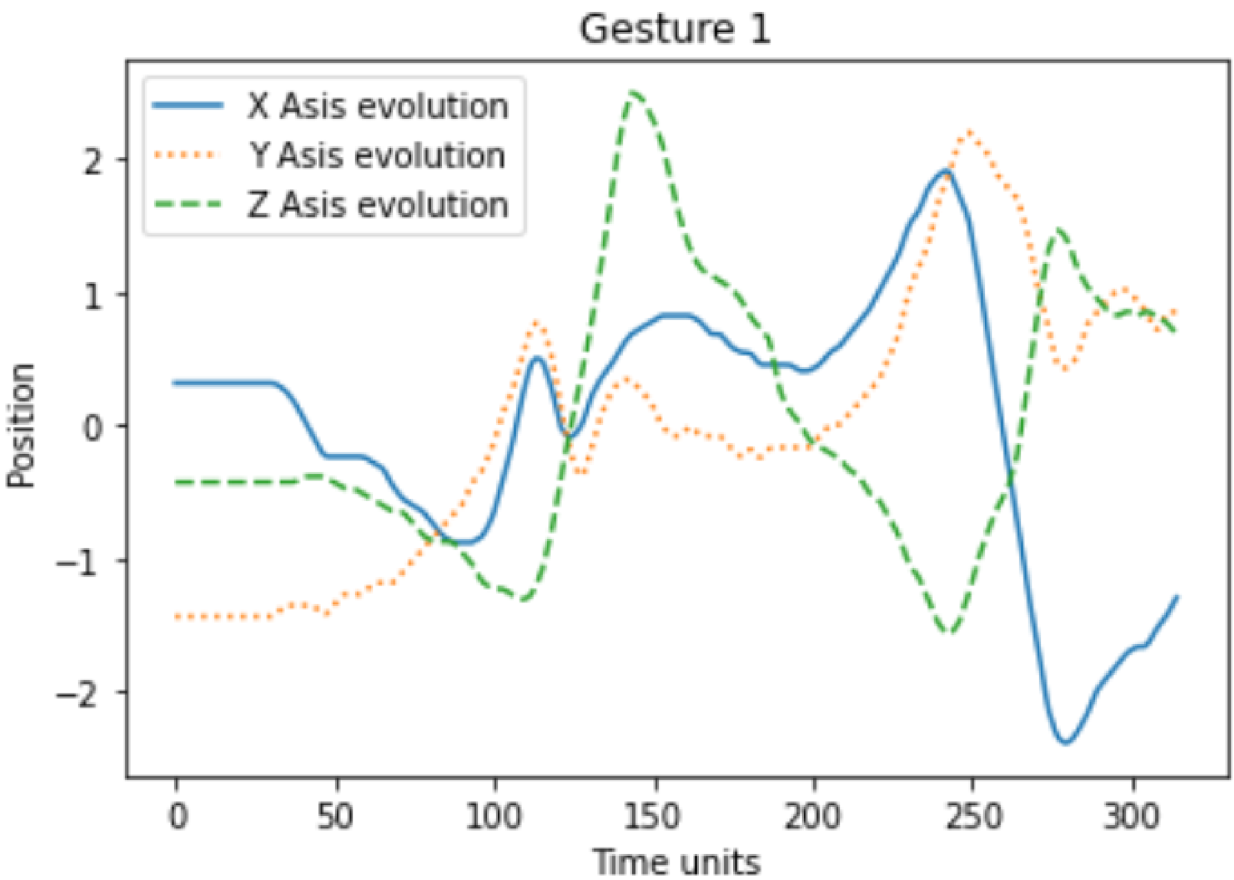

```
We observe that classes are balanced.
Due to this fact, the testing process will produce relevant results.
```

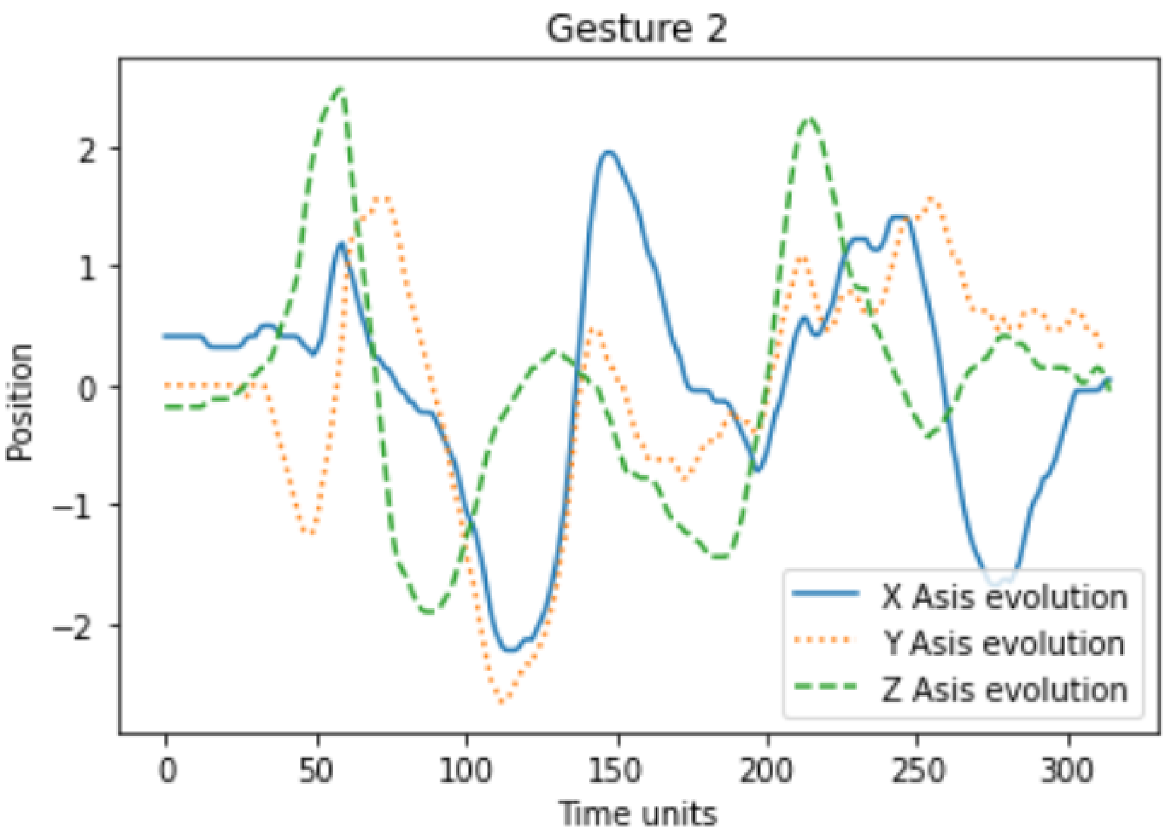


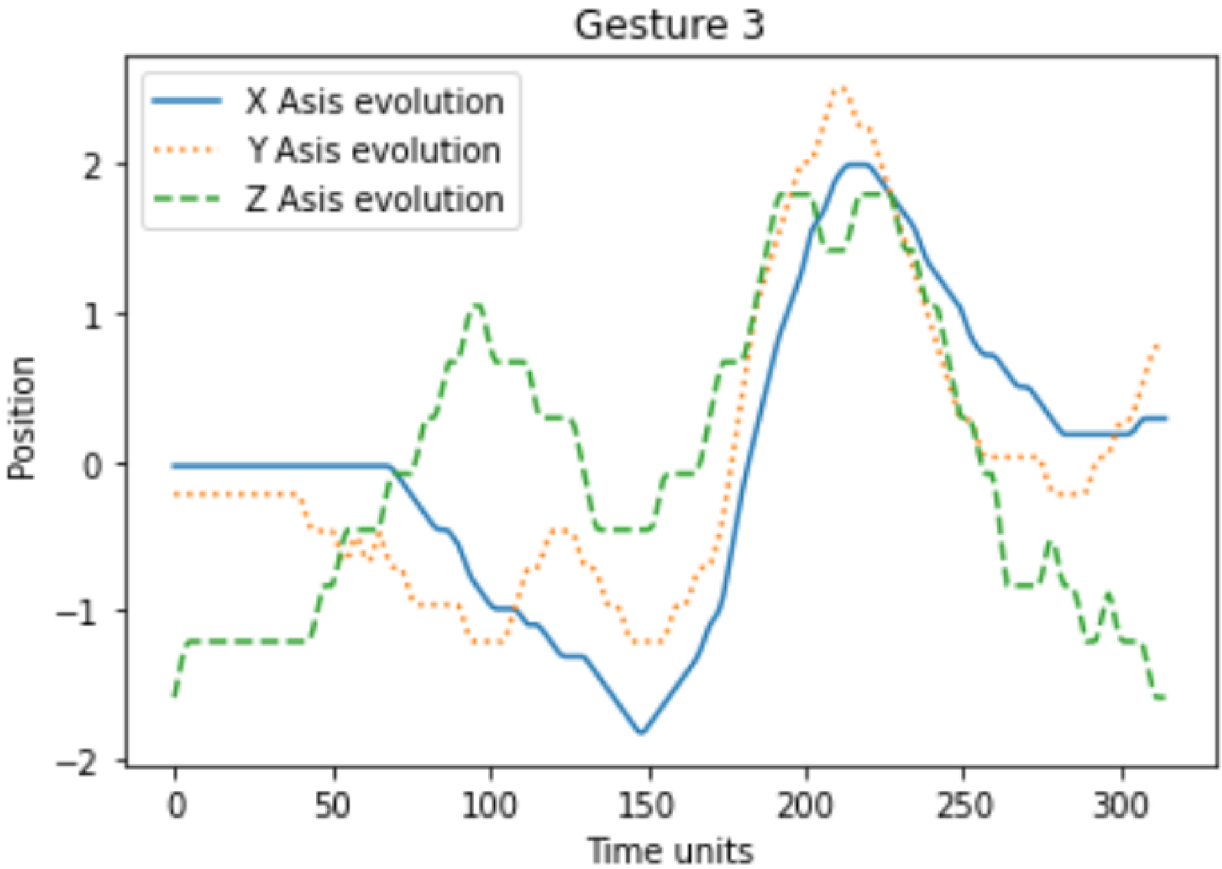
U Wave Gesture axis evolution

Random example for each gesture

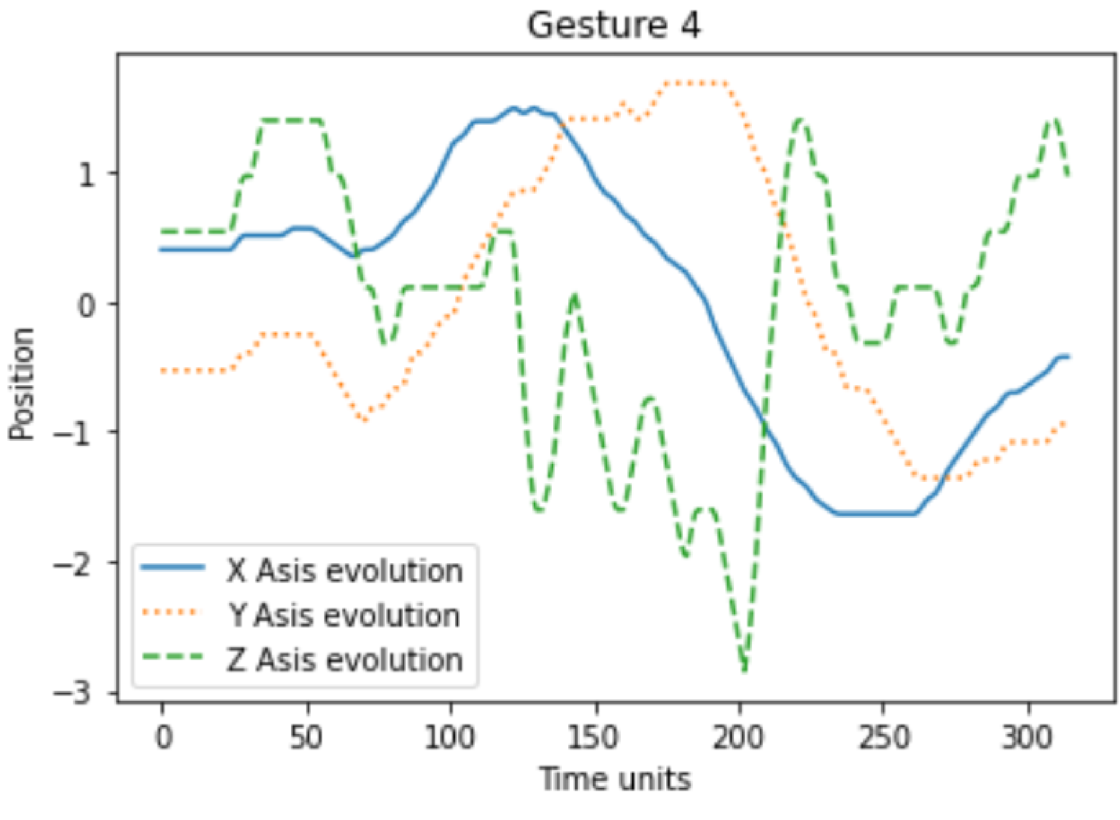
```
Will follow a random a example for each gesture in the dataset.
We can observe the evolution of the 3 axis in time.
We can observe that the Gestures are ochiometrically different
```


Gesture 1
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units
2
1
0
−1
−2
0
50
100
150
200
250
300

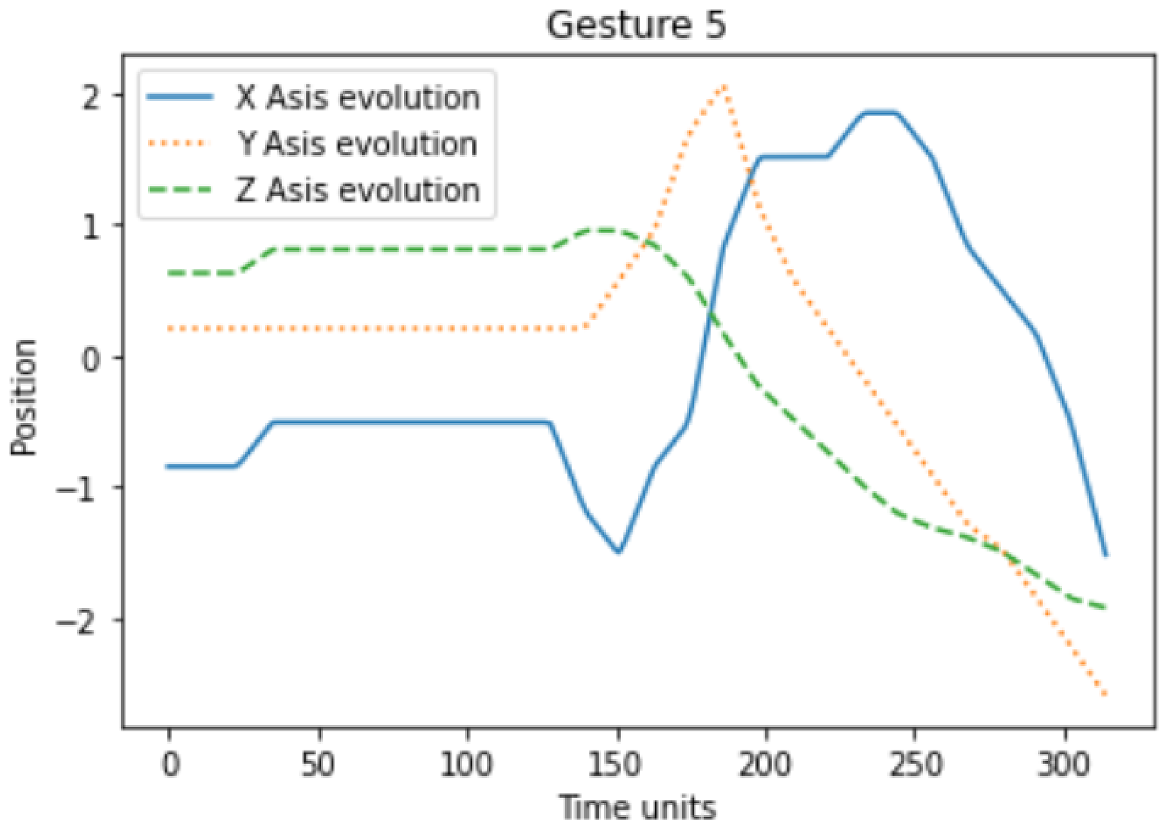

Gesture 2
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units
2
1
0
−1
−2
0
50
100
150
200
250
300

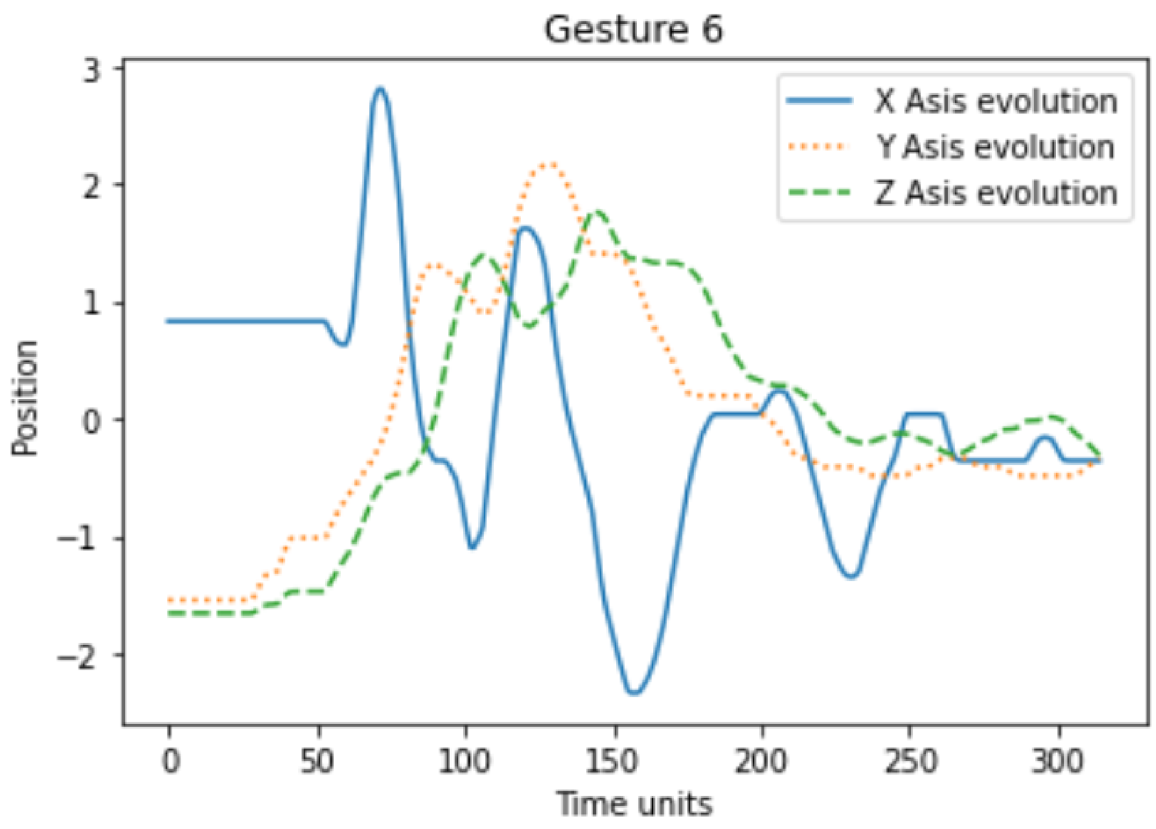


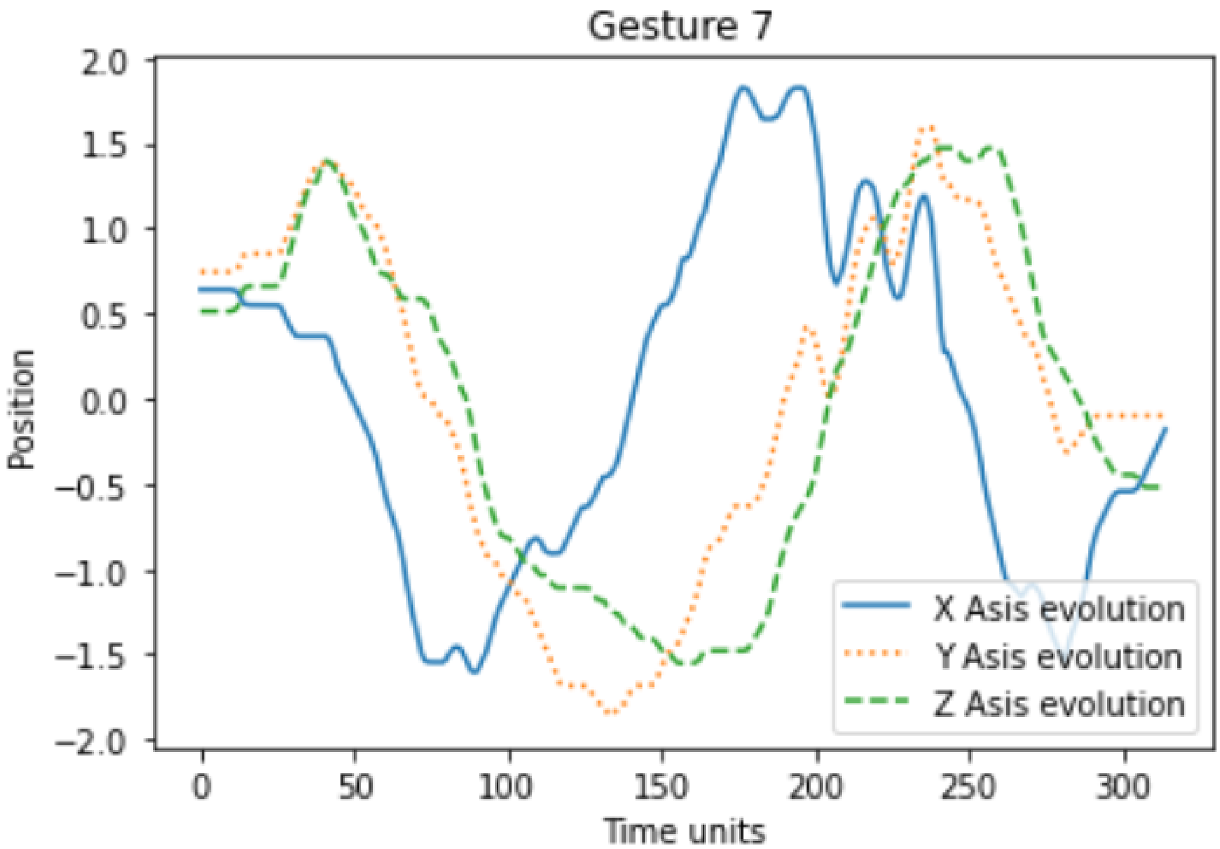
Gesture 3
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units

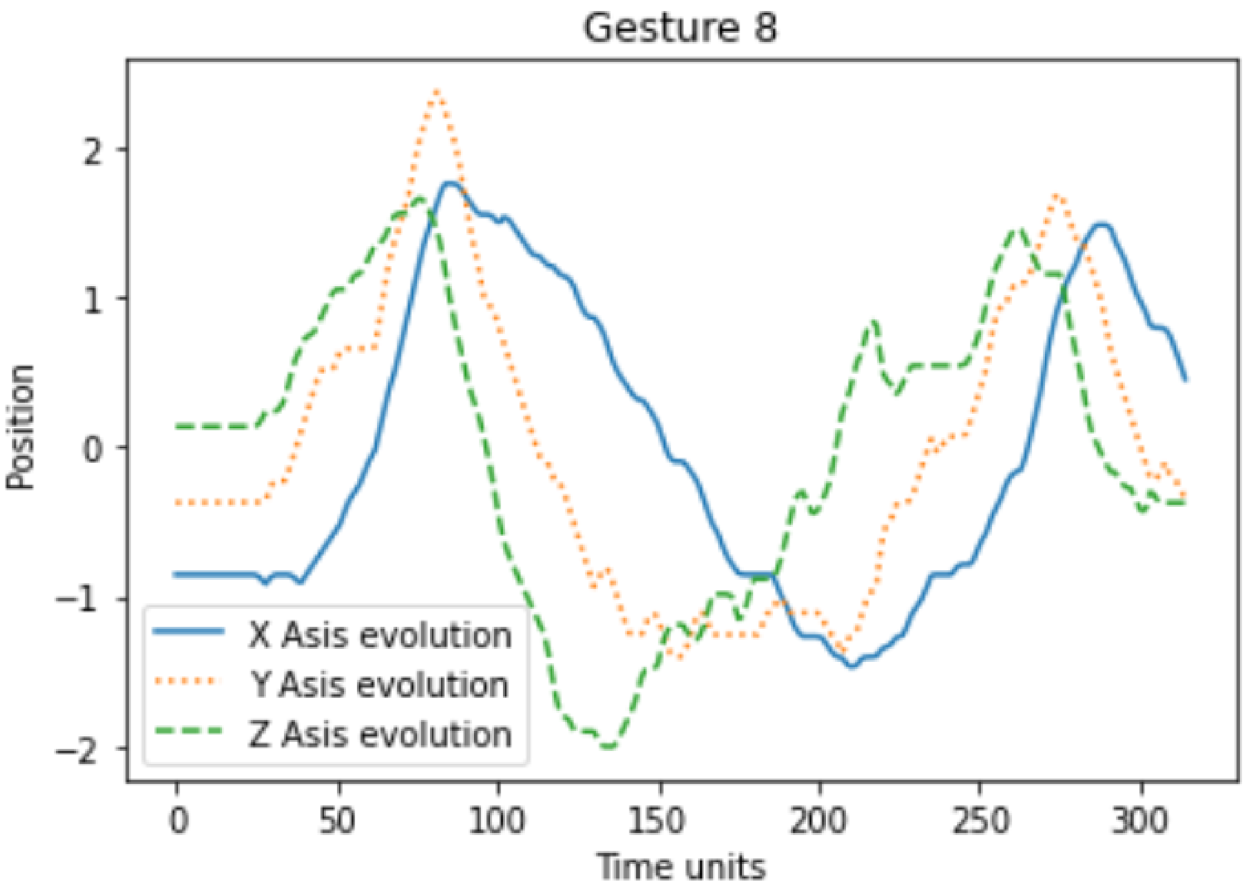


Gesture 4
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units


Gesture 5
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units
2
1
0
−1
−2
0
50
100
150
200
250
300

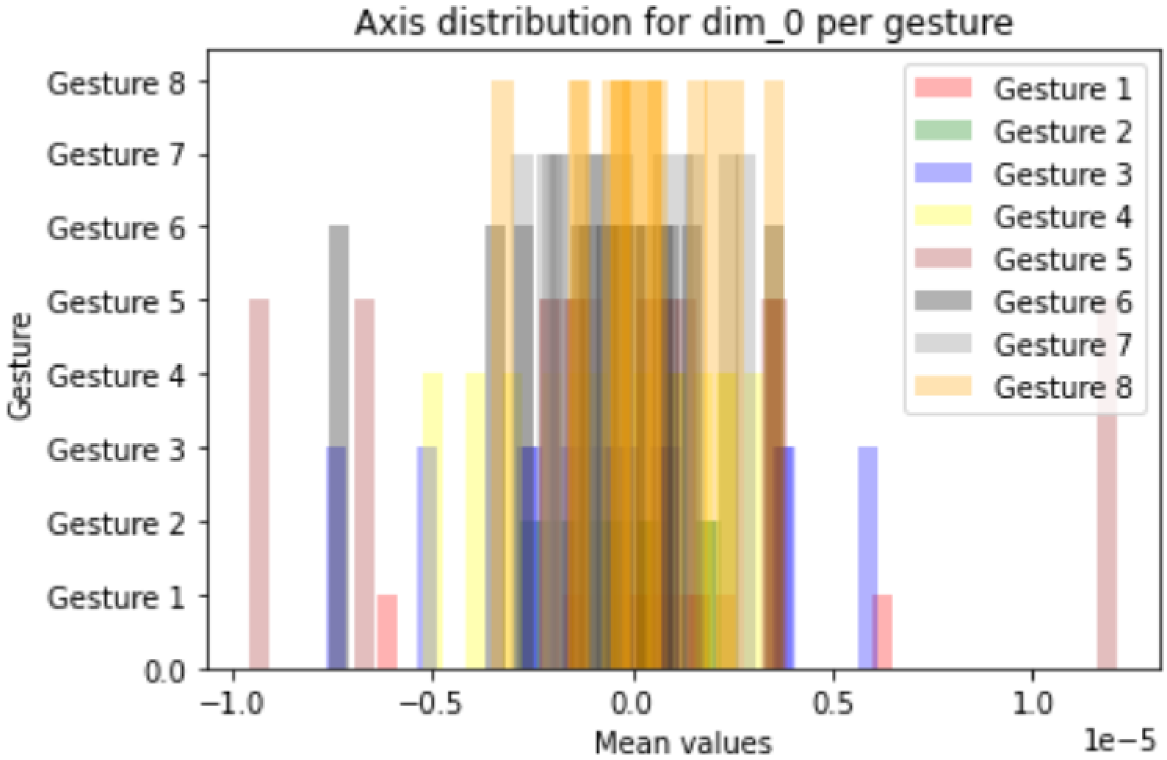

Gesture 6
X Asis evolution
Y Asis evolution
Z Asis evolution
Position
Time units
3
2
1
0
−1
−2
0
50
100
150
200
250
300

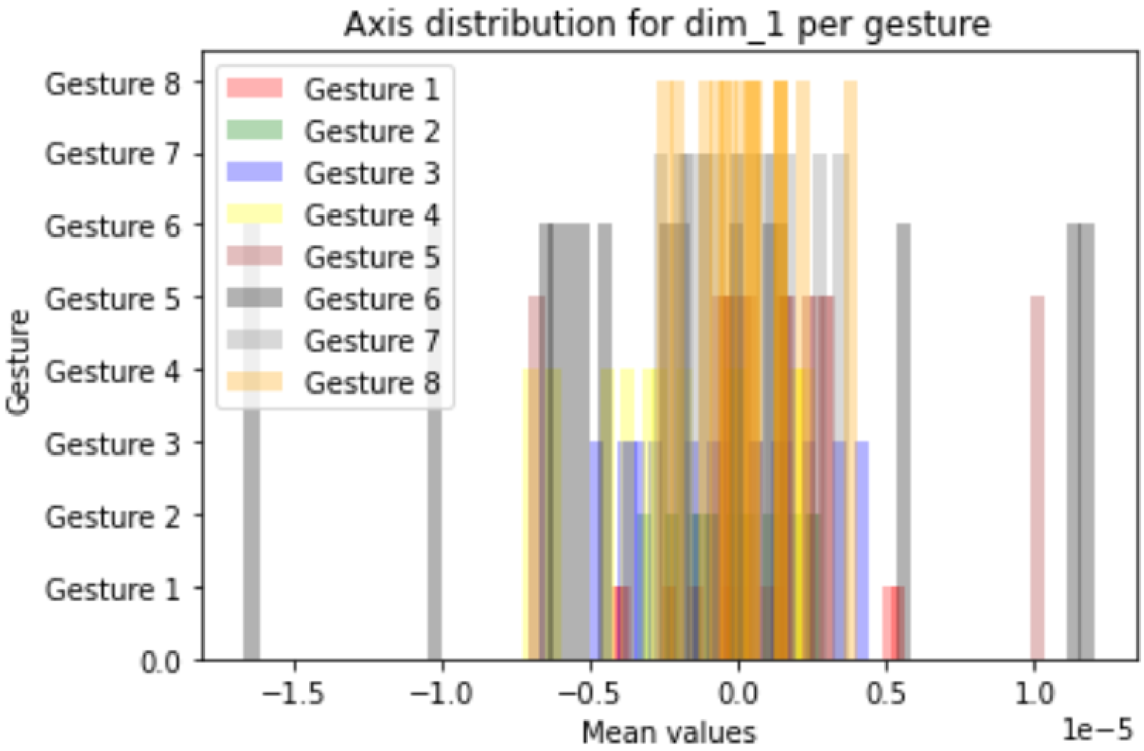


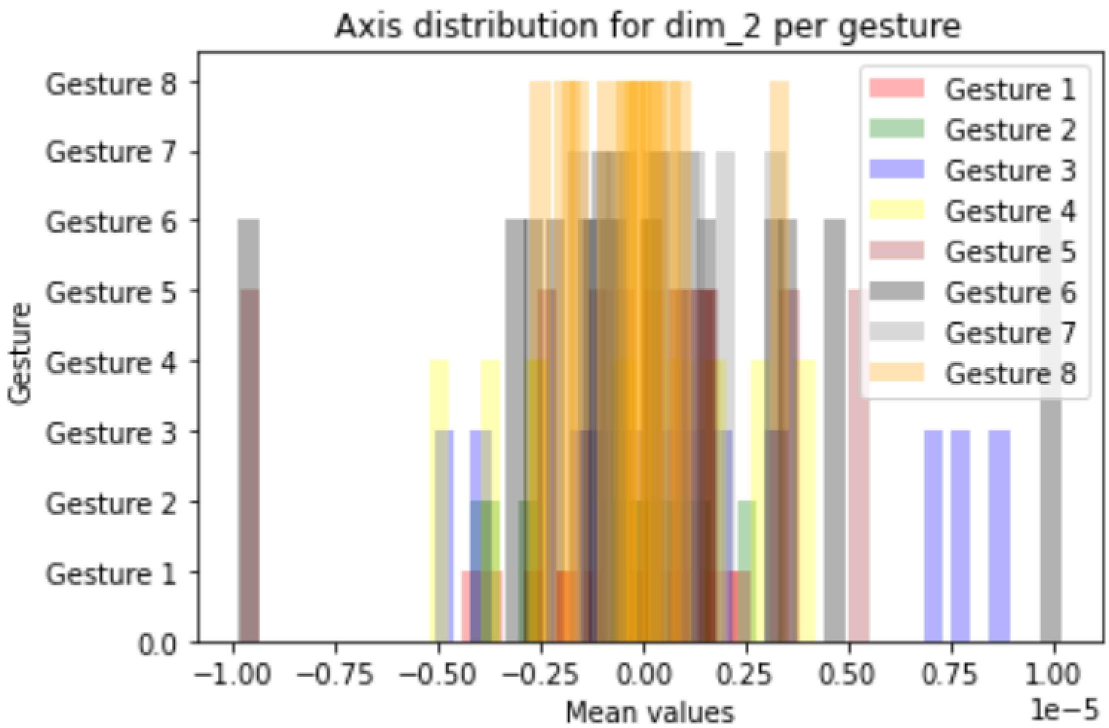


Axis evolution distribution per gesture

```
In the following images we can see the mean value ranges of each axis
evolution with respect to a gesture.
dim_0 is X axis
dim_1 is Y axis
dim_2 is Z axis
```





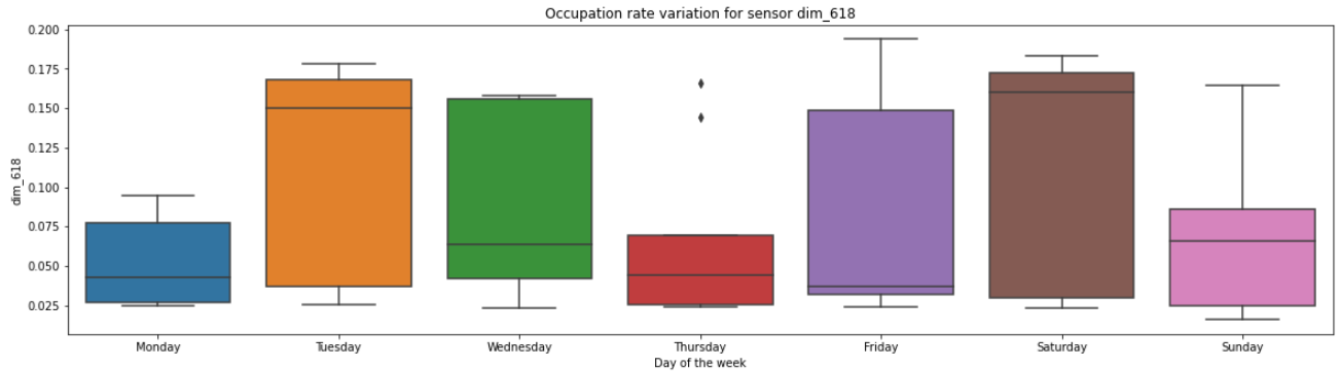


PEMS-SF Analysis

Best sensors deviation week days

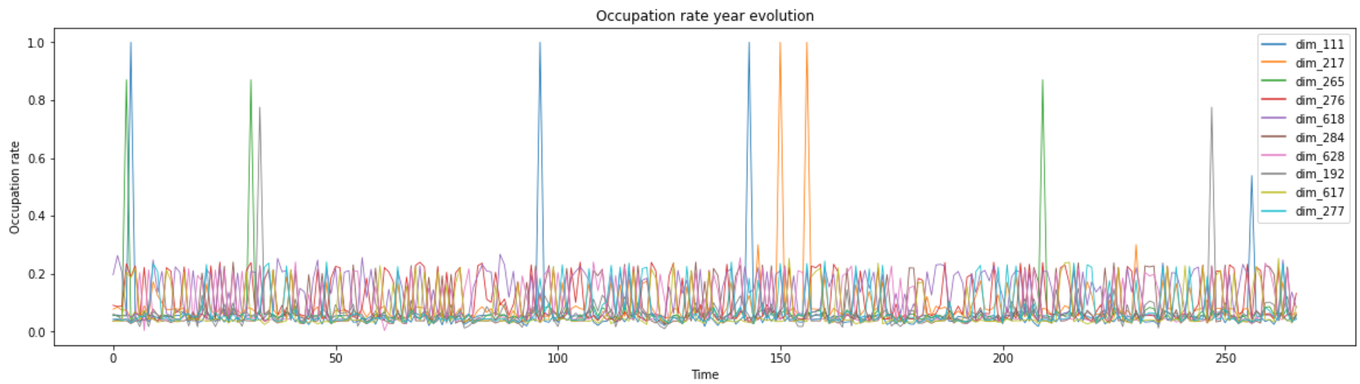
```
I've chosen the sensor with the highest variation in the acquisitioned
data.
```

```
For each day i've chosen 8 examples and printed the range in the occupation
rate.
```



Best sensors occupation rate year evolution

```
For this example i've chosen the sensors with the most interesting
information.
I've printed the mean occupation rate evolution during a year for each of
these sensors.
We can see some rare spikes during the period which may be the results of
big gatherings of people
```



Feature selection & classic algorithms

```
In the case of Random Forest, SVM, and Extreme Gradient Boosted Trees i've
executed a GridSearch over the train data and found the best
hyperparameters.
With these hyperparameters I constructed the corresponding instance of the
classifier, trained it and computed the results with respect to test data.
```

PEMS-SF

Random forest

```
{
    "accuracy_score": 0.8959537572254336,
    "precision_score": [1.0, 1.0, 1.0, 0.82608696, 0.95,
        0.77419355, 0.76923077
    ],
    "recall_score": [0.86666667, 1.0, 0.84615385, 0.82608696, 0.86363636,
        0.88888889, 1.0
    ],
    "f1_score": [0.92857143, 1.0, 0.91666667, 0.82608696, 0.9047619,
        0.82758621, 0.86956522
    ],
    "confusion_matrix": [
        [26, 0, 0, 0, 0, 0, 4],
        [0, 25, 0, 0, 0, 0, 0],
        [0, 0, 22, 4, 0, 0, 0],
        [0, 0, 0, 19, 0, 4, 0],
        [0, 0, 0, 0, 19, 3, 0],
        [0, 0, 0, 0, 1, 24, 2],
        [0, 0, 0, 0, 0, 0, 20]
    ],
    "parameters": {
        "max_depth": 20,
        "min_samples_split": 2,
        "n_estimators": 200
    }
}
```

SVM

```
{
    "accuracy_score": 0.7167630057803468,
    "precision_score": [0.82758621, 0.85714286, 0.66666667, 0.59090909,
0.7,{
    "accuracy_score": 0.930635838150289,
    "precision_score": [1.0, 0.88888889, 0.96, 1.0, 0.91666667,
        0.92592593, 0.83333333
    ],
    "recall_score": [0.86666667, 0.96, 0.92307692, 0.86956522, 1.0,
        0.92592593, 1.0
    ],
    "f1_score": [0.92857143, 0.92307692, 0.94117647, 0.93023256,
0.95652174,
        0.92592593, 0.90909091
    ],
    "confusion_matrix": [
        [26, 0, 0, 0, 0, 2, 2],
        [0, 24, 1, 0, 0, 0, 0],
        [0, 0, 24, 0, 2, 0, 0],
        [0, 3, 0, 20, 0, 0, 0],
        [0, 0, 0, 0, 22, 0, 0],
```

```
        [0, 0, 0, 0, 0, 25, 2],
        [0, 0, 0, 0, 0, 0, 20]
    ]
}
        0.6, 0.81818182
    ],
    "recall_score": [0.8, 0.72, 0.61538462, 0.56521739, 0.63636364,
        0.77777778, 0.9
    ],
    "f1_score": [0.81355932, 0.7826087, 0.64, 0.57777778, 0.66666667,
        0.67741935, 0.85714286
    ],
    "confusion_matrix": [
        [24, 0, 0, 0, 0, 2, 4],
        [3, 18, 0, 2, 0, 2, 0],
        [0, 2, 16, 4, 2, 2, 0],
        [0, 0, 3, 13, 2, 5, 0],
        [0, 0, 2, 3, 14, 3, 0],
        [0, 1, 3, 0, 2, 21, 0],
        [2, 0, 0, 0, 0, 0, 18]
    ],
    "parameters": {
        "C": 2,
        "degree": 2,
        "gamma": "scale",
        "kernel": "poly"
    }
}
```

Gradient Boosted Trees

```
{
    "accuracy_score": 0.930635838150289,
    "precision_score": [1.0, 0.88888889, 0.96, 1.0, 0.91666667,
        0.92592593, 0.83333333
    ],
    "recall_score": [0.86666667, 0.96, 0.92307692, 0.86956522, 1.0,
        0.92592593, 1.0
    ],
    "f1_score": [0.92857143, 0.92307692, 0.94117647, 0.93023256,
0.95652174,
        0.92592593, 0.90909091
    ],
    "confusion_matrix": [
        [26, 0, 0, 0, 0, 2, 2],
        [0, 24, 1, 0, 0, 0, 0],
        [0, 0, 24, 0, 2, 0, 0],
        [0, 3, 0, 20, 0, 0, 0],
        [0, 0, 0, 0, 22, 0, 0],
        [0, 0, 0, 0, 0, 25, 2],
        [0, 0, 0, 0, 0, 0, 20]
```

```
    ],
    "parameters": {
        "learning_rate": 0.1,
        "max_depth": 2,
        "n_estimators": 100
    }
}
```

U Wave Gesture

Random forest


```
{
    "accuracy_score": 0.375,
    "precision_score": [0.30666667, 0.58695652, 0.36956522, 0.11111111,
0.41666667,
        0.35897436, 0.375, 0.375
    ],
    "recall_score": [0.575, 0.675, 0.425, 0.05, 0.25, 0.35, 0.375, 0.3],
    "f1_score": [0.4, 0.62790698, 0.39534884, 0.06896552, 0.3125,
        0.35443038, 0.375, 0.33333333
    ],
    "confusion_matrix": [
        [23, 1, 2, 6, 0, 2, 3, 3],
        [2, 27, 0, 0, 0, 0, 7, 4],
        [9, 0, 17, 5, 5, 4, 0, 0],
        [7, 0, 17, 2, 4, 7, 0, 3],
        [10, 1, 5, 0, 10, 12, 1, 1],
        [10, 0, 5, 5, 5, 14, 0, 1],
        [9, 8, 0, 0, 0, 0, 15, 8],
        [5, 9, 0, 0, 0, 0, 14, 12]
    ],
    "parameters": {
        "max_depth": 15,
        "min_samples_split": 7,
        "n_estimators": 200
    }
}
```


SVM


```
{
    "accuracy_score": 0.403125,
    "precision_score": [0.29487179, 0.60465116, 0.5, 0.3125, 0.43478261,
        0.39473684, 0.39285714, 0.39583333
    ],
    "recall_score": [0.575, 0.65, 0.375, 0.25, 0.25, 0.375, 0.275, 0.475],
    "f1_score": [0.38983051, 0.62650602, 0.42857143, 0.27777778,
```

```
0.31746032,
        0.38461538, 0.32352941, 0.43181818
    ],
    "confusion_matrix": [
        [23, 0, 0, 6, 0, 3, 3, 5],
        [4, 26, 0, 0, 0, 0, 5, 5],
        [6, 1, 15, 12, 2, 4, 0, 0],
        [8, 0, 11, 10, 4, 4, 0, 3],
        [10, 3, 2, 0, 10, 12, 1, 2],
        [11, 0, 2, 4, 7, 15, 0, 1],
        [11, 5, 0, 0, 0, 0, 11, 13],
        [5, 8, 0, 0, 0, 0, 8, 19]
    ],
    "parameters": {
        "C": 4,
        "degree": 1,
        "gamma": "scale",
        "kernel": "poly"
    }
}
```

Gradient Boosted Trees

```
{
    "accuracy_score": 0.35625,
    "precision_score": [0.3442623, 0.55813953, 0.31111111, 0.28571429,
0.33333333,
        0.32692308, 0.32258065, 0.31818182
    ],
    "recall_score": [0.525, 0.6, 0.35, 0.1, 0.25, 0.425, 0.25, 0.35],
    "f1_score": [0.41584158, 0.57831325, 0.32941176, 0.14814815,
0.28571429,
        0.36956522, 0.28169014, 0.33333333
    ],
    "confusion_matrix": [
        [21, 1, 3, 2, 3, 3, 3, 4],
        [5, 24, 0, 0, 0, 0, 4, 7],
        [6, 1, 14, 8, 2, 8, 1, 0],
        [4, 0, 14, 4, 5, 9, 0, 4],
        [6, 1, 6, 0, 10, 15, 1, 1],
        [7, 0, 8, 0, 7, 17, 0, 1],
        [6, 10, 0, 0, 1, 0, 10, 13],
        [6, 6, 0, 0, 2, 0, 12, 14]
    ],
    "parameters": {
        "learning_rate": 0.05,
        "max_depth": 4,
        "n_estimators": 5
    }
}
```

Neural networks

U Wave Gesture

Multi-Layered Perceptron

```
The neural network architecture consists of 5 layers of 100 neurons each.
```


```
{
    "accuracy_score": 0.853125,
    "precision_score": [0.9, 0.90697674, 0.85714286, 0.9, 0.68627451,
        0.75, 1.0, 0.925
    ],
    "recall_score": [0.9, 0.975, 0.9, 0.675, 0.875, 0.825, 0.75, 0.925],
    "f1_score": [0.9, 0.93975904, 0.87804878, 0.77142857, 0.76923077,
        0.78571429, 0.85714286, 0.925
    ],
    "confusion_matrix": [
        [36, 0, 0, 0, 0, 4, 0, 0],
        [0, 39, 0, 0, 0, 1, 0, 0],
        [0, 0, 36, 1, 3, 0, 0, 0],
        [0, 0, 0, 27, 12, 0, 0, 1],
        [0, 1, 1, 0, 35, 3, 0, 0],
        [4, 0, 1, 2, 0, 33, 0, 0],
        [0, 3, 4, 0, 0, 1, 30, 2],
        [0, 0, 0, 0, 1, 2, 0, 37]
    ],
    "parameters": {
        "activation": "relu",
        "solver": "adam",
        "alpha": 0.001,
        "learning_rate": "constant",
        "hidden_layer_sizes": (100, 100, 100, 100, 100),
        "max_iter": 100000
    }
}
```


Convolutional Neural Network

```
First two convolutional layers produce the feature maps.
The Drouput layer is very important. It manages to remove the outliers from
the feature maps
and to keep only the relevant attributes.
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_24 (Conv1D)          (None, 315, 64)           640

 conv1d_25 (Conv1D)          (None, 313, 64)           12352

 dropout_21 (Dropout)        (None, 313, 64)           0

 max_pooling1d_12 (MaxPoolin  (None, 156, 64)          0
 g1D)

 flatten_12 (Flatten)        (None, 9984)              0

 dense_42 (Dense)            (None, 500)               4992500

 dense_43 (Dense)            (None, 8)                 4008

=================================================================
Total params: 5,009,500
Trainable params: 5,009,500
Non-trainable params: 0
_________________________________________________________________
```


```
{
    "accuracy_score": 0.859375,
    "precision_score": [0.94594595, 0.86666667, 0.8372093, 0.86666667,
0.74,
        0.85365854, 0.88235294, 0.925
    ],
    "recall_score": [0.875, 0.975, 0.9, 0.65, 0.925, 0.875, 0.75, 0.925],
    "f1_score": [0.90909091, 0.91764706, 0.86746988, 0.74285714,
0.82222222,
        0.86419753, 0.81081081, 0.925
    ],
    "confusion_matrix": [
        [35, 0, 1, 0, 0, 4, 0, 0],
        [0, 39, 0, 0, 0, 1, 0, 0],
        [0, 0, 36, 1, 1, 0, 2, 0],
        [0, 0, 0, 26, 11, 1, 0, 2],
        [0, 1, 1, 0, 37, 0, 1, 0],
        [0, 0, 2, 2, 0, 35, 1, 0],
        [0, 5, 3, 1, 0, 0, 30, 1],
        [2, 0, 0, 0, 1, 0, 0, 37]
```

```
    ]
}
```

Long short-term memory

```
In this architecture the first layer does all the work.
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_9 (LSTM)               (None, 1000)              5264000

 dropout_22 (Dropout)        (None, 1000)              0

 dense_44 (Dense)            (None, 1000)              1001000

 dense_45 (Dense)            (None, 8)                 8008

=================================================================
Total params: 6,273,008
Trainable params: 6,273,008
Non-trainable params: 0
_________________________________________________________________
```

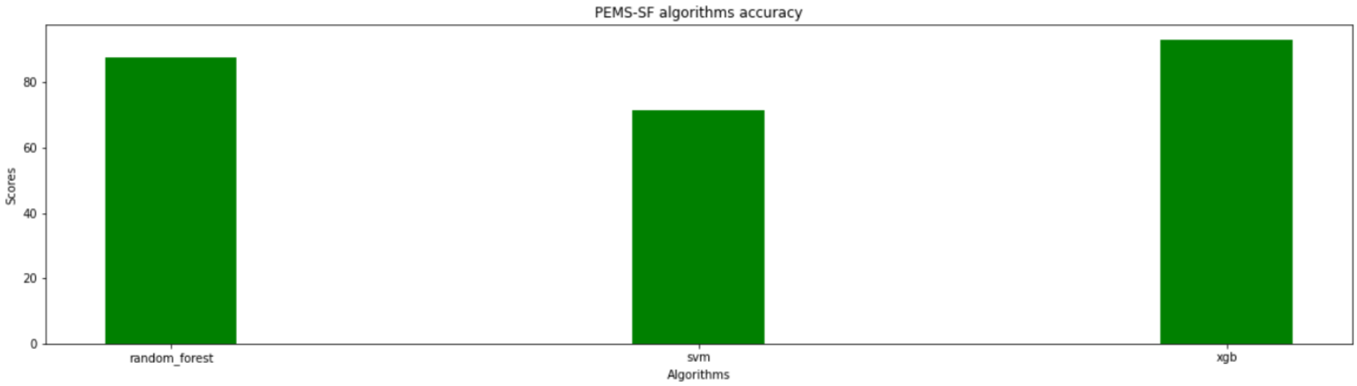

```
{
    "accuracy_score": 0.86875,
    "precision_score": [0.92307692, 0.95238095, 0.85365854, 0.92307692,
0.65454545,
        0.80487805, 1.0, 0.95121951
    ],
    "recall_score": [0.9, 1.0, 0.875, 0.6, 0.9, 0.825, 0.875, 0.975],
    "f1_score": [0.91139241, 0.97560976, 0.86419753, 0.72727273,
0.75789474,
        0.81481481, 0.93333333, 0.96296296
    ],
    "confusion_matrix": [
        [36, 0, 1, 0, 0, 3, 0, 0],
        [0, 40, 0, 0, 0, 0, 0, 0],
        [0, 0, 35, 0, 4, 1, 0, 0],
        [0, 0, 0, 24, 14, 1, 0, 1],
        [0, 0, 1, 0, 36, 2, 0, 1],
        [3, 0, 1, 2, 1, 33, 0, 0],
```

```
        [0, 2, 3, 0, 0, 0, 35, 0],
        [0, 0, 0, 0, 0, 1, 0, 39]
    ]
}
```

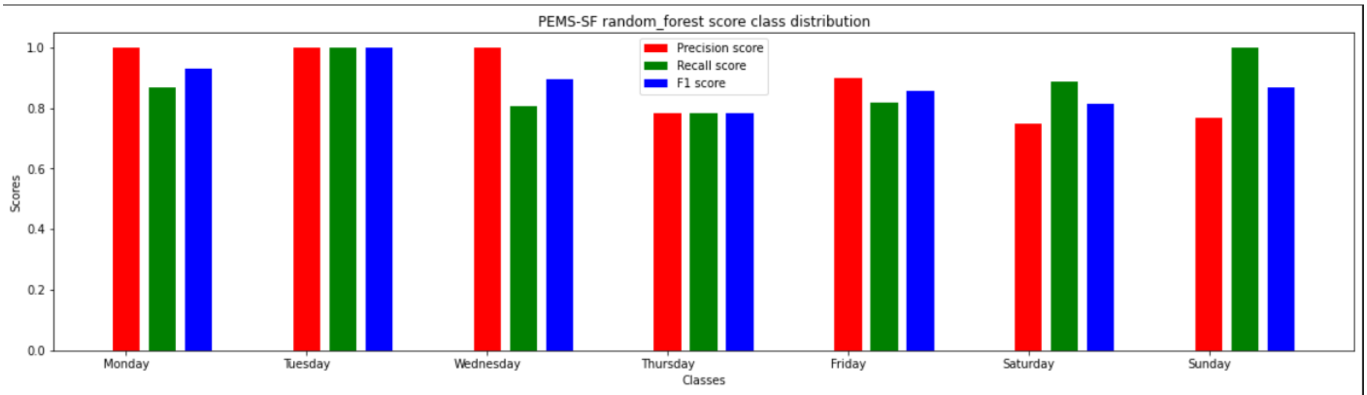
Final overview

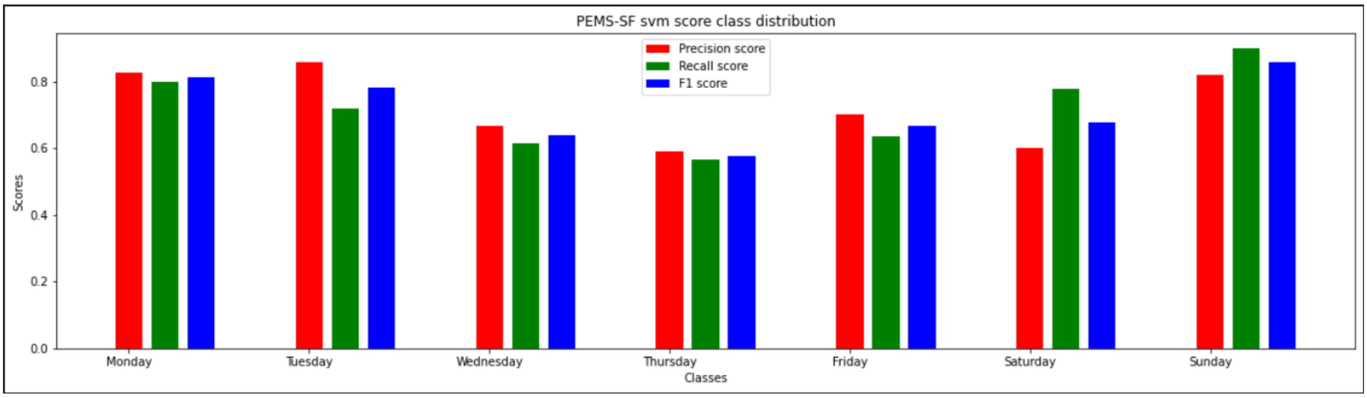
PEMS-SF

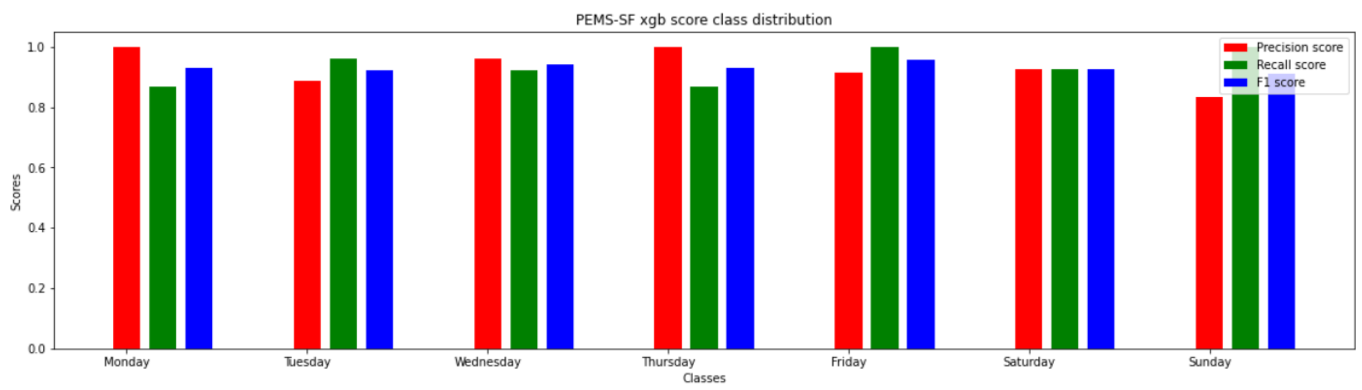
Accuracy



Algorithms efficiency with respect to classes

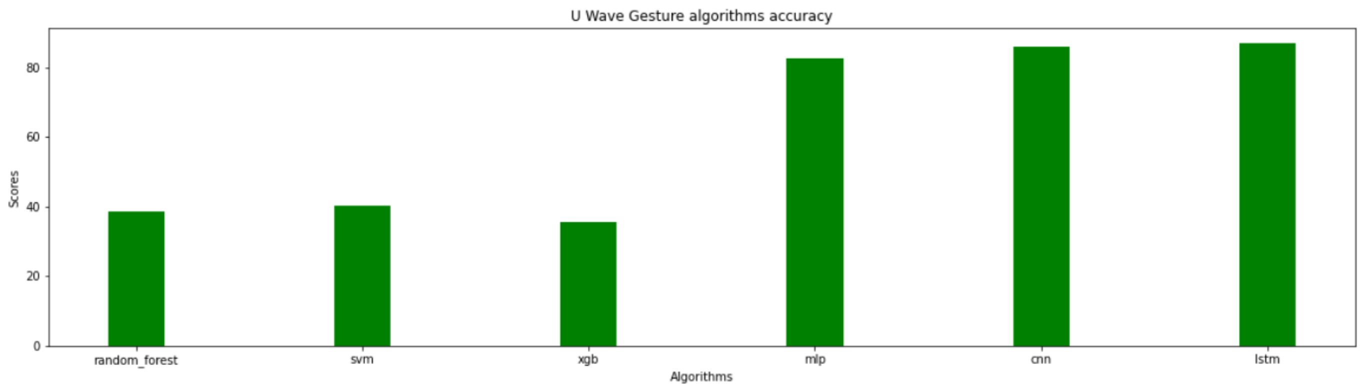




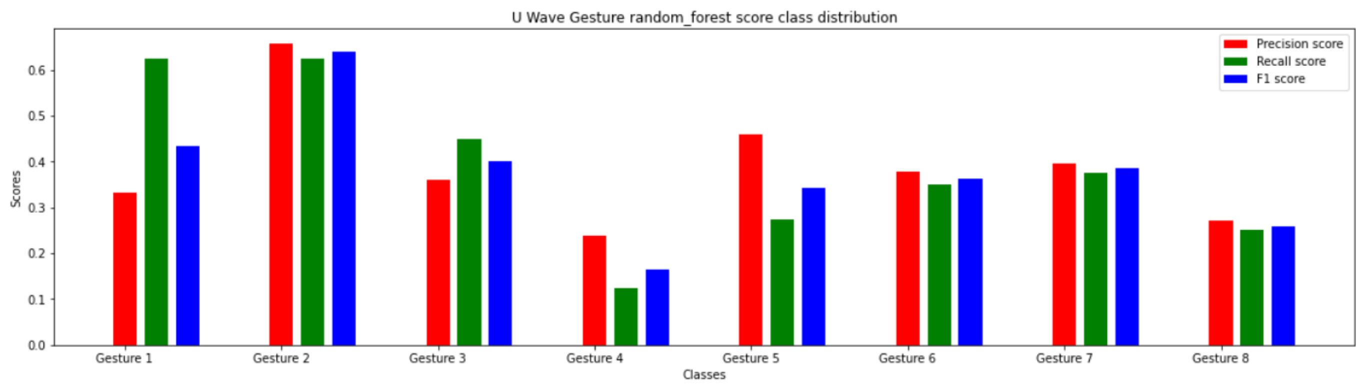


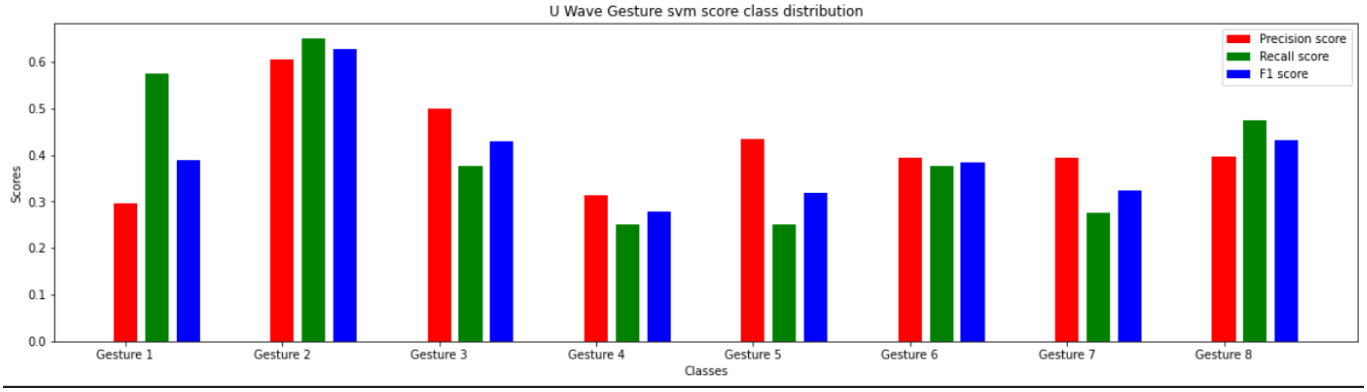
U Wave Gesture

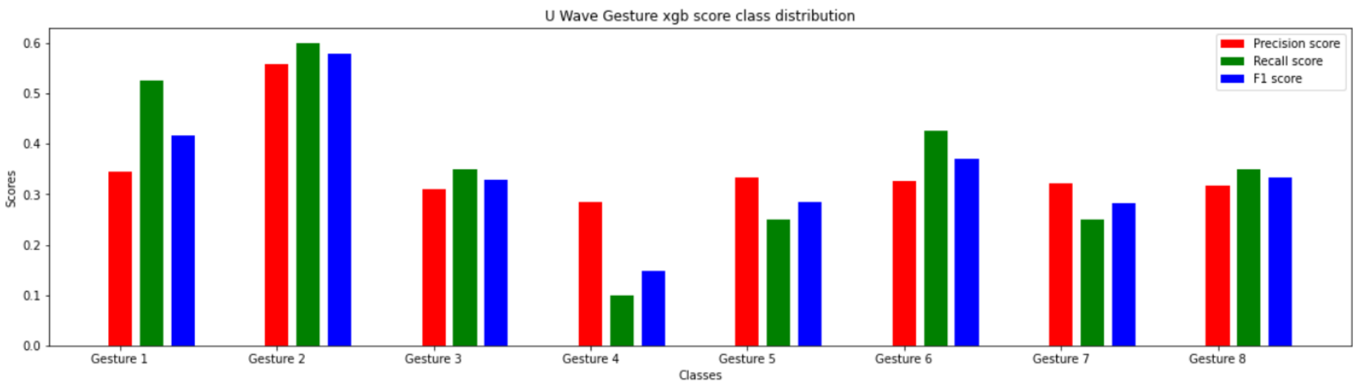
Accuracy

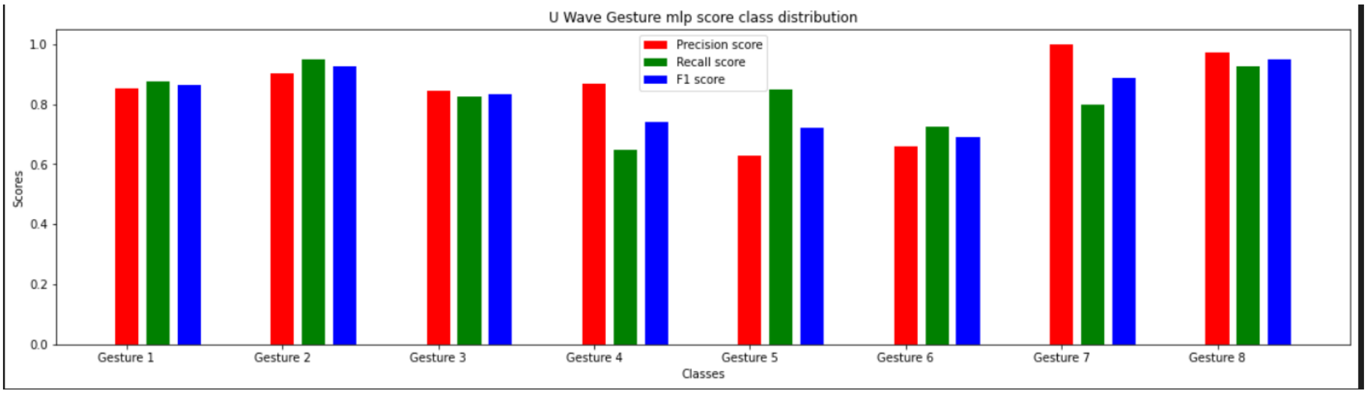


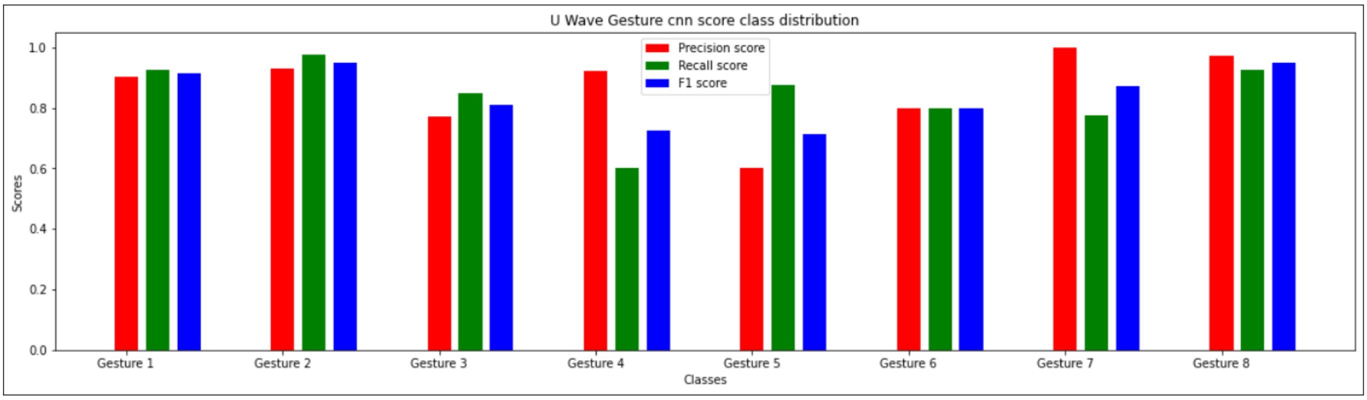
Algorithms efficiency with respect to classes

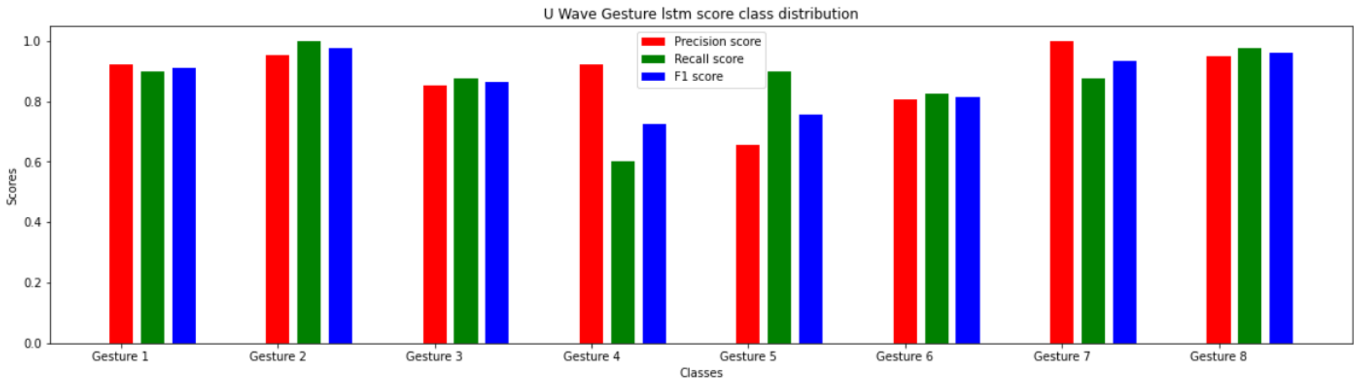



U Wave Gesture svm score class distribution
Precision score
Recall score
F1 score
Scores
Classes
Gesture 1
Gesture 2
Gesture 3
Gesture 4
Gesture 5
Gesture 6
Gesture 7
Gesture 8


U Wave Gesture xgb score class distribution
Precision score
Recall score
F1 score
Scores
Classes
Gesture 1
Gesture 2
Gesture 3
Gesture 4
Gesture 5
Gesture 6
Gesture 7
Gesture 8


U Wave Gesture mlp score class distribution
Precision score
Recall score
F1 score
Scores
Classes
Gesture 1
Gesture 2
Gesture 3
Gesture 4
Gesture 5
Gesture 6
Gesture 7
Gesture 8


U Wave Gesture cnn score class distribution
Precision score
Recall score
F1 score
Scores
Classes
Gesture 1
Gesture 2
Gesture 3
Gesture 4
Gesture 5
Gesture 6
Gesture 7
Gesture 8



Algorithms train loss curves