CAPSTONE 2 PRESENTATION

BC TALKING LINE – ARTIFICIAL INTELLIGENCE

Meet the team:

Yousef Elbayoumi

1722326
Project documentation responsible

Bora Bilgin

1729464
Data scientist

Saeed Al-Maliki

1720304
Project integration responsible

İpek Yasin

1503913
Project planning responsible

Sinem Uzun

1737571
Data scientist

Mohammed Ali

1735170
Project integration responsible

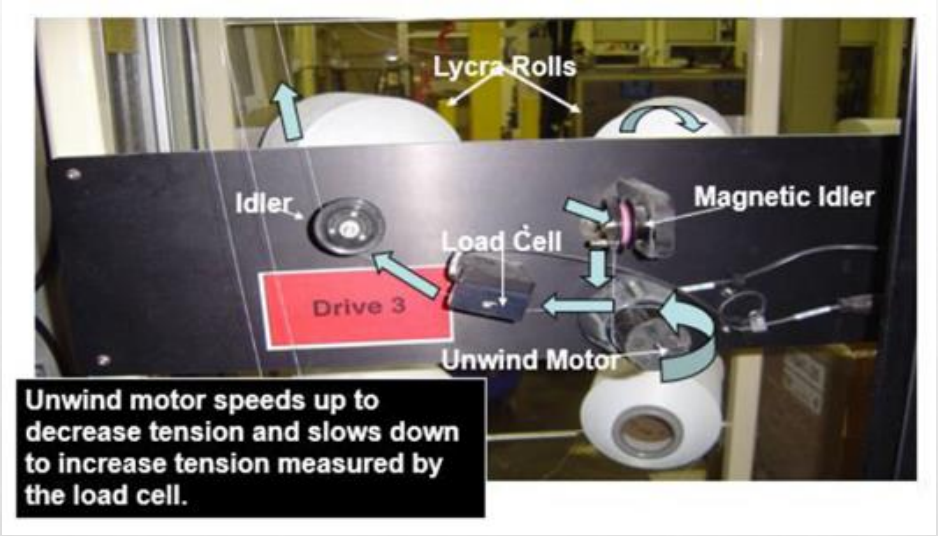
TABLE OF CONTENT

OVERVIEW

Identification of the need

- Reduce the unplanned stops
- Provide significative action suggestions
- Ensure parameters are revealed

Definition of the problem



The main problem:

Elastic Breakout

Taken from (P&G) factory

Standards and constraints

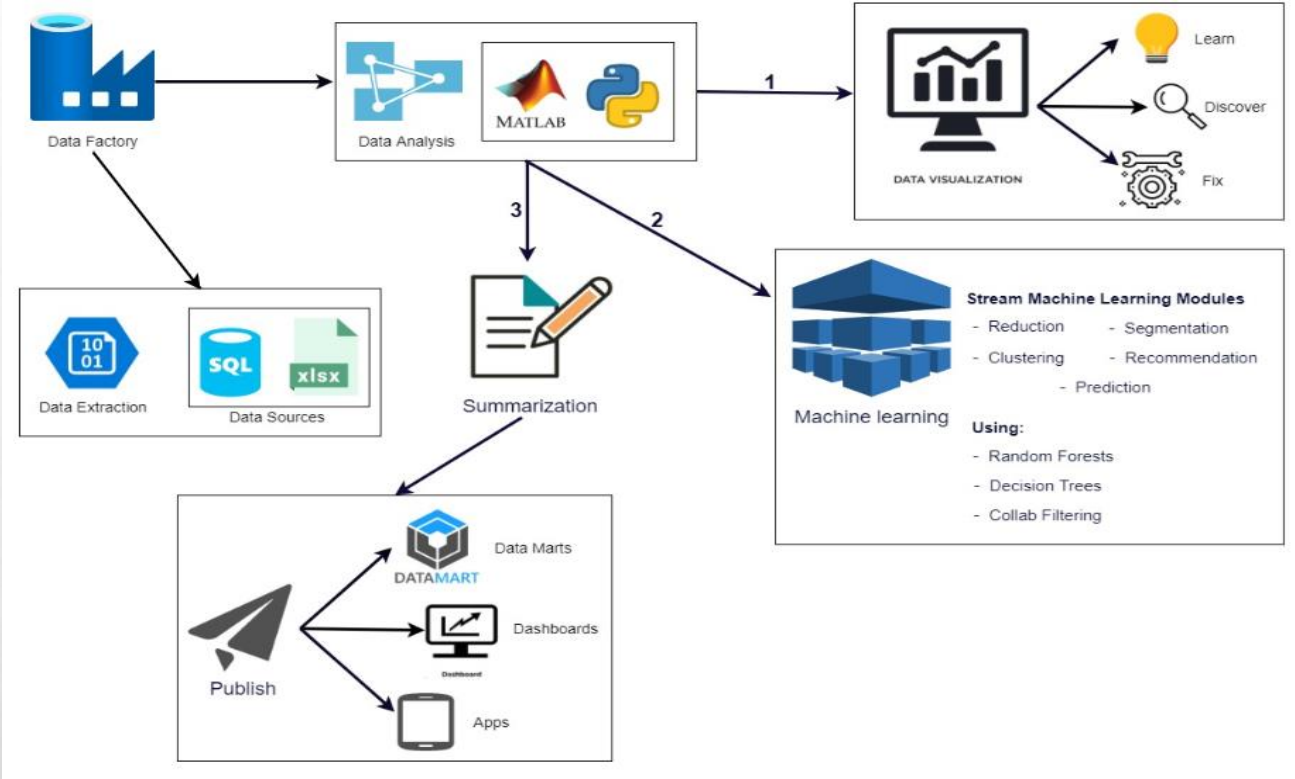
- Environmental effects
- Social effects
- Economic effects
- Ethical issues
- Health and safety

Literature review

[1] On the impact of smart sensor approximations on the accuracy of machine learning tasks, Daniele Jahier Pagliari & Massimo Poncino(2020).
-Signal modifications introduced by smart sensors.
-The effect of the electricity approaches on the efficiency of the ML models has been evaluated.
-Machine Learning models were founded that they are very sensitive to smart sensors changes.

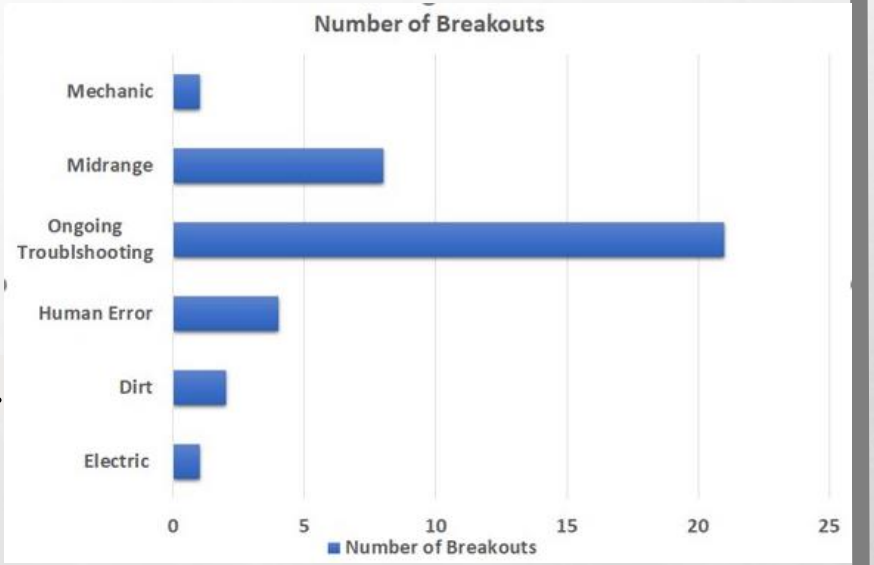
[2] A distributed sensor-fault detection and diagnosis framework using machine learning, Sana Ullah Jan, Young Doo Lee, & In Soo Koo(2020).
-Plan a sensor-fault detection and diagnosis framework.
-Quick Fault Detection (QFD) and Low Computation Cost (LCC).
Presented a system for distributed sensor-fault detection and diagnosis framework.

Overview diagram



First progress EEE

- Elastic breakout unplanned stops.
- Delay in production schedule.
- Various reasons for stops.
- 63 Total stops.
- 41.2 % of the stops due to raw material.
- 1 month worth of data collected.

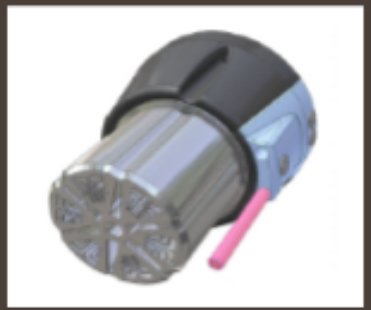


YARN CONTROL SYSTEM

- KTF/RW-MF (Constant Tension Feeder-ReWinder-MicroFeeder).
- Consist of various independent modules.

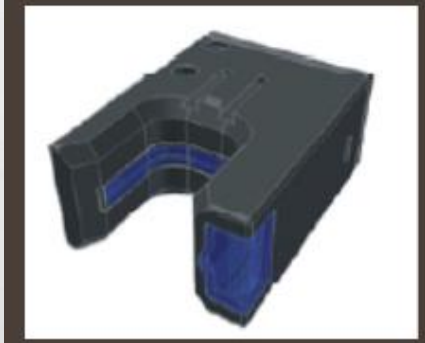
- SM-DIN/RW-MF

- KTF/100/xx



- TS4/xxxyRW

- IS3x/TS



- SMART KTF 2000

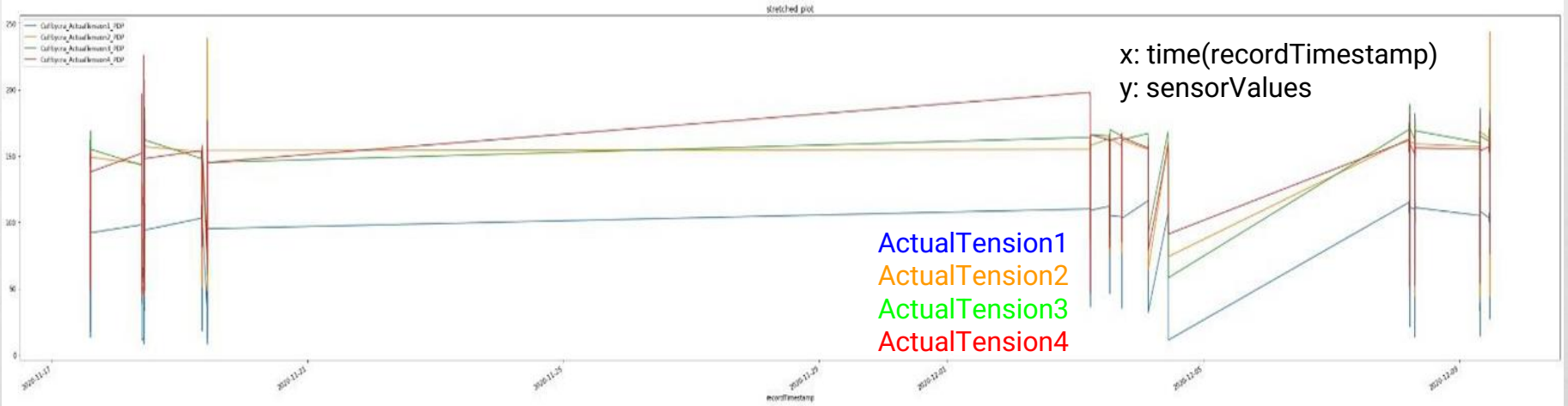
- PC LINK KTF

First progress SEN

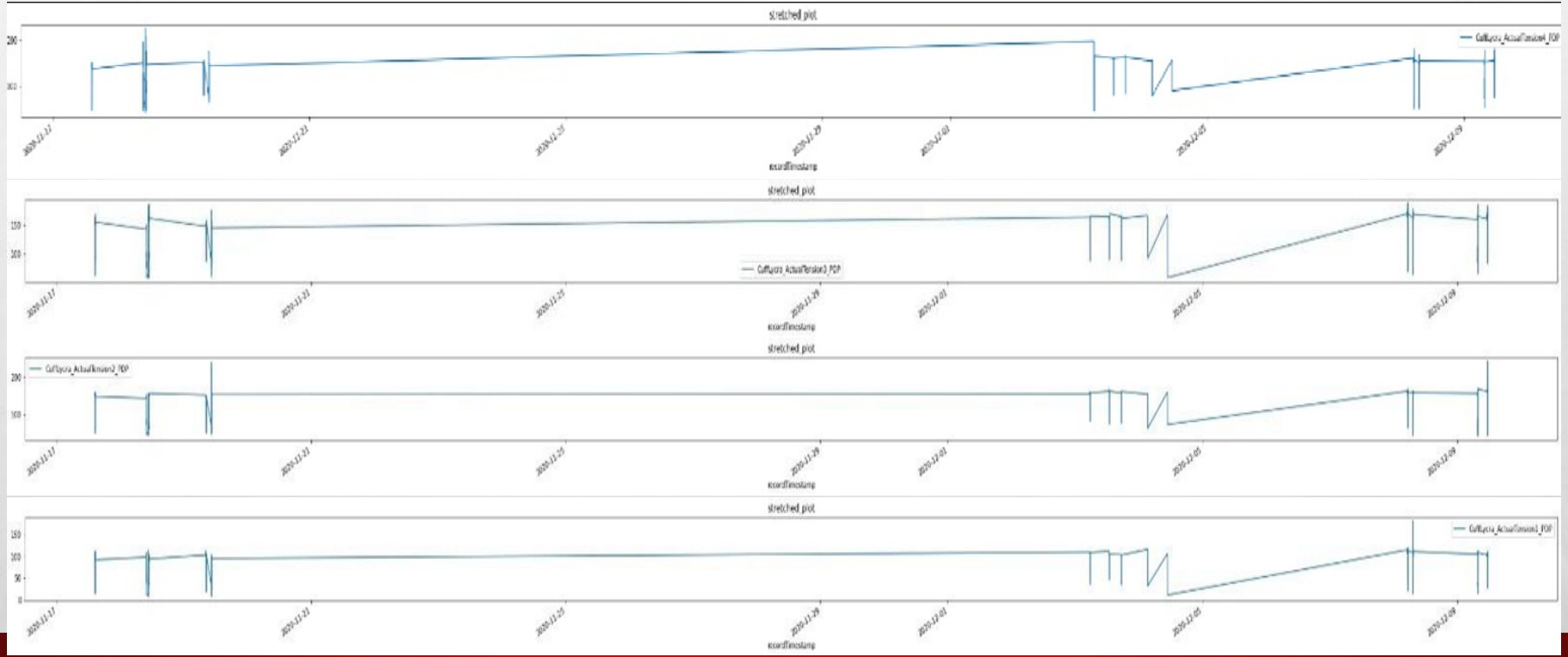
Functional Requirements

Nonfunctional Requirements
- Performance Requirements
- Safety Requirements
- Security Requirements
- Software Quality Attributes
- Business Rules

SENSOR VALUES D104

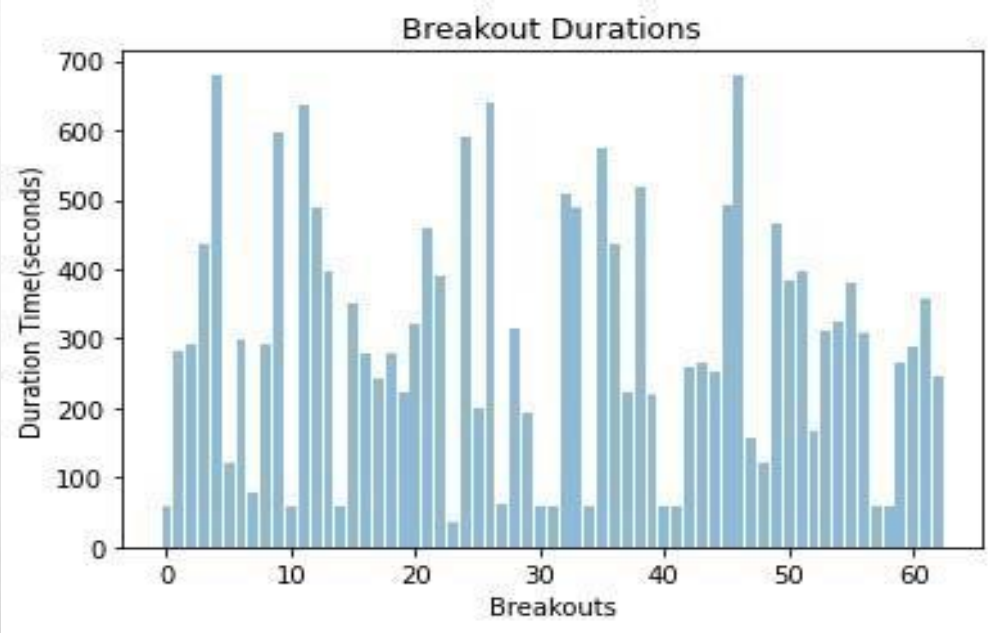


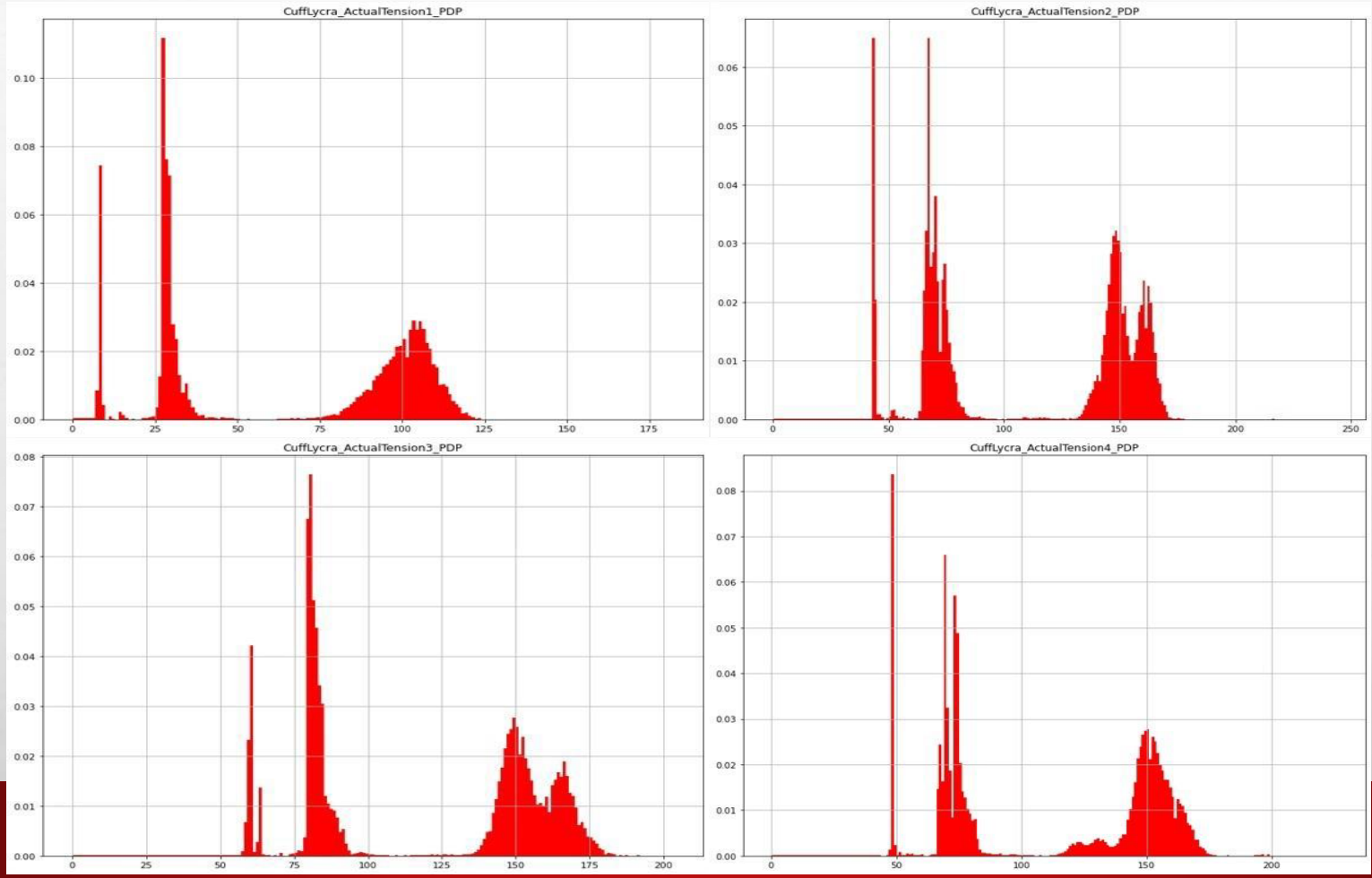
SENSOR VALUES SEPARATELY

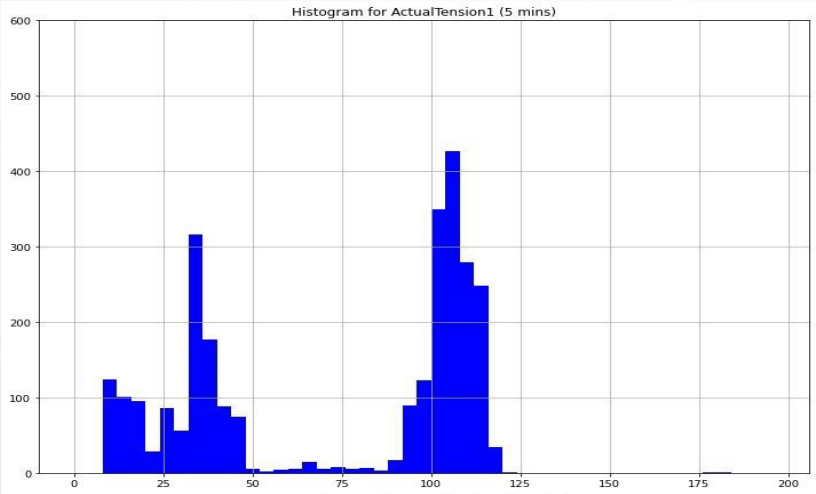


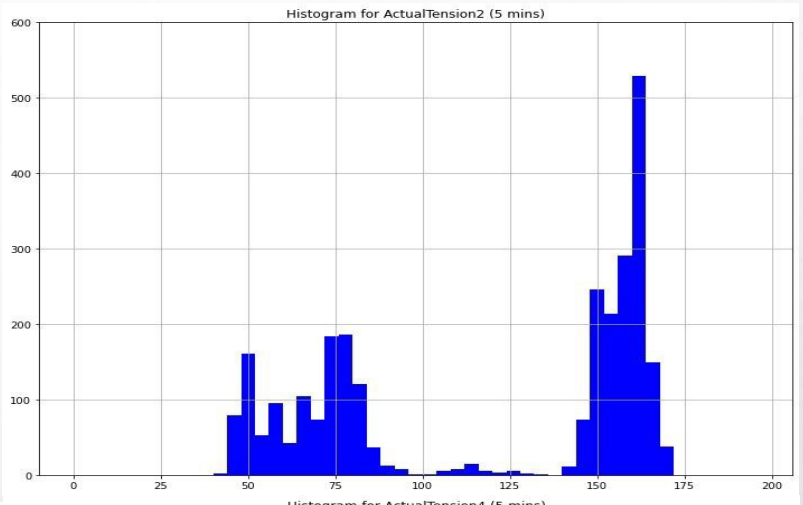
Sample Breakout

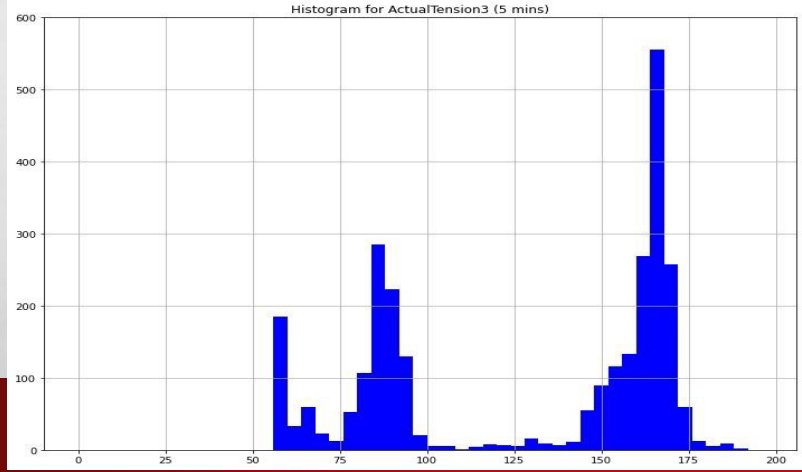
BREAKOUT DURATIONS

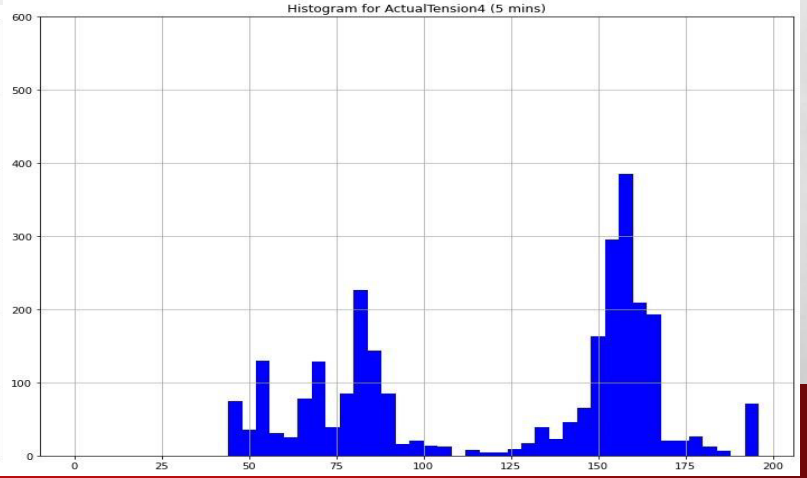


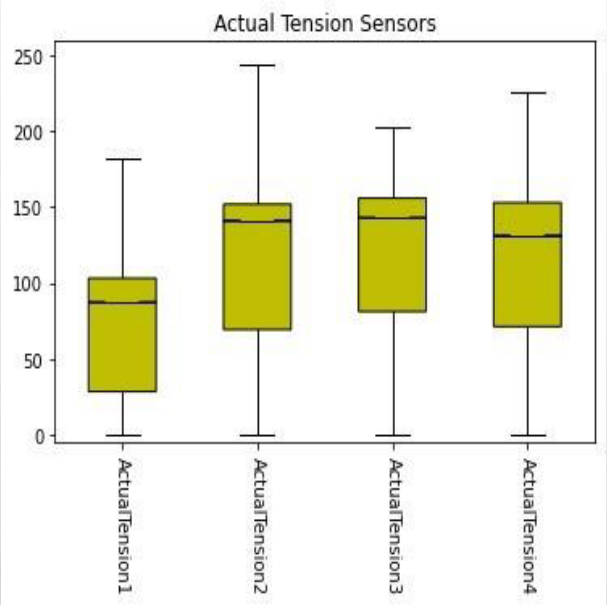

CuffLycra_ActualTension1_PDP
CuffLycra_ActualTension2_PDP
CuffLycra_ActualTension3_PDP
CuffLycra_ActualTension4_PDP

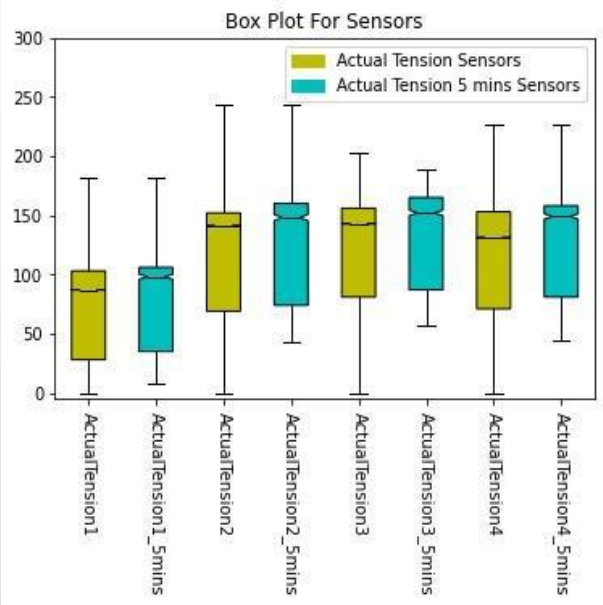

Histogram for ActualTension1 (5 mins)
600
500
400
300
200
100
0
0
25
50
75
100
125
150
175
200

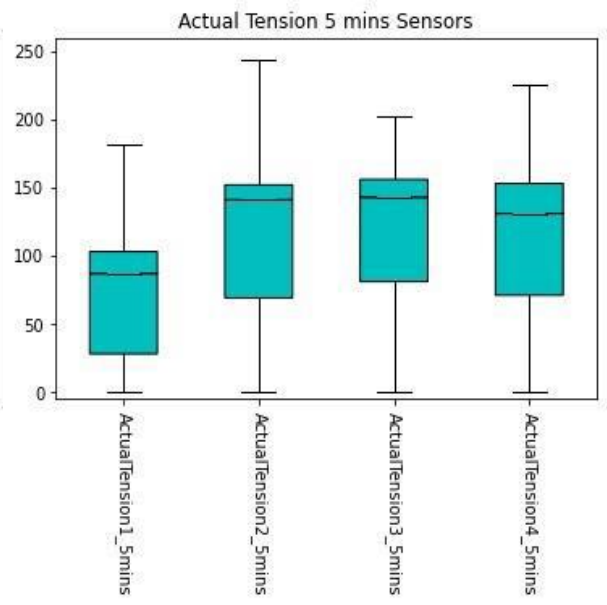

Histogram for ActualTension2 (5 mins)
600
500
400
300
200
100
0
0
25
50
75
100
125
150
175
200

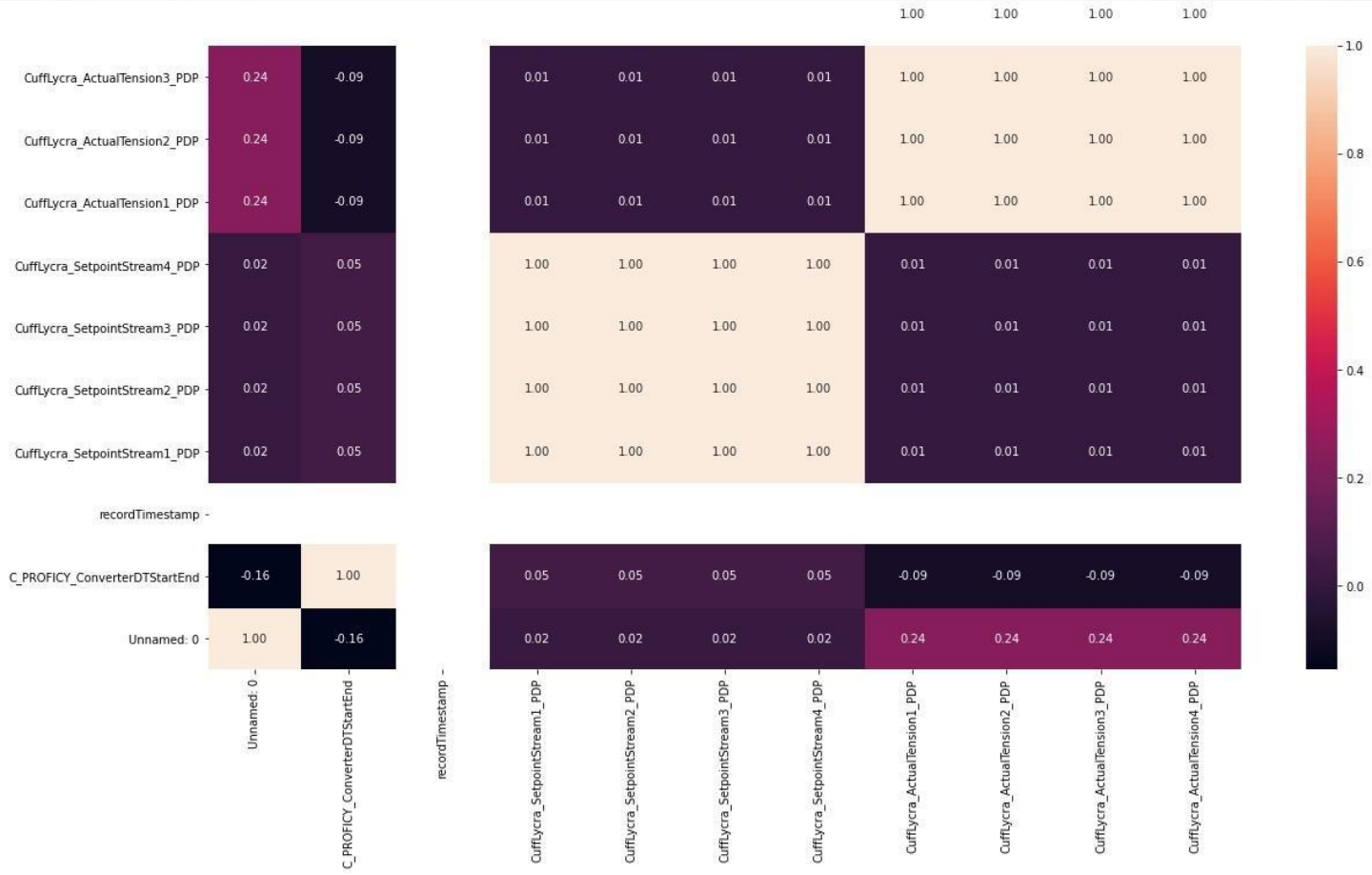

Histogram for ActualTension3 (5 mins)
600
500
400
300
200
100
0
0
25
50
75
100
125
150
175
200


Histogram for ActualTension4 (5 mins)
600
500
400
300
200
100
0
0
25
50
75
100
125
150
175
200


Actual Tension Sensors
250
200
150
100
50
0
ActualTension1
ActualTension2
ActualTension3
ActualTension4


Box Plot For Sensors
Actual Tension Sensors
Actual Tension 5 mins Sensors
300
250
200
150
100
50
0
ActualTension1
ActualTension1_5mins
ActualTension2
ActualTension2_5mins
ActualTension3
ActualTension3_5mins
ActualTension4
ActualTension4_5mins


Actual Tension 5 mins Sensors
250
200
150
100
50
0
ActualTension1_5mins
ActualTension2_5mins
ActualTension3_5mins
ActualTension4_5mins


CuffLycra_ActualTension3_PDP
CuffLycra_ActualTension2_PDP
CuffLycra_ActualTension1_PDP
CuffLycra_SetpointStream4_PDP
CuffLycra_SetpointStream3_PDP
CuffLycra_SetpointStream2_PDP
CuffLycra_SetpointStream1_PDP
recordTimestamp
C_PROFICY_ConverterDTStartEnd
Unnamed: 0
1.0
0.8
0.6
0.4
0.2
0.0

Our final data after pre-processing

df - DataFrame

Index	'nnamed:	_Converte	cordTimestam	sensor1	sensor2	sensor3	sensor4	timeLeft
104	14526	0	1605350596	0.717742	0.564315	0.784615	0.588496	262113
105	14527	0	1605350601	0.685484	0.564315	0.784615	0.584071	262108
106	14528	0	1605350606	0.741935	0.568465	0.748718	0.59292	262103
107	14529	0	1605350611	0.741935	0.568465	0.748718	0.59292	262098
108	14530	0	1605350616	0.685484	0.589212	0.753846	0.60177	262093
109	14531	0	1605350621	0.685484	0.589212	0.753846	0.60177	262088
110	14532	0	1605350626	0.741935	0.560166	0.774359	0.566372	262083
111	14533	0	1605350631	0.677419	0.572614	0.764103	0.59292	262078
112	14534	0	1605350636	0.741935	0.564315	0.753846	0.575221	262073
113	14535	0	1605350641	0.741935	0.564315	0.753846	0.575221	262068
114	14536	0	1605350646	0.717742	0.564315	0.758974	0.59292	262063
115	14537	0	1605350651	0.677419	0.572614	0.784615	0.579646	262058
116	14538	0	1605350656	0.677419	0.572614	0.769231	0.584071	262053
117	14539	0	1605350661	0.75	0.560166	0.758974	0.579646	262048
118	14540	0	1605350666	0.701613	0.560166	0.753846	0.561947	262043

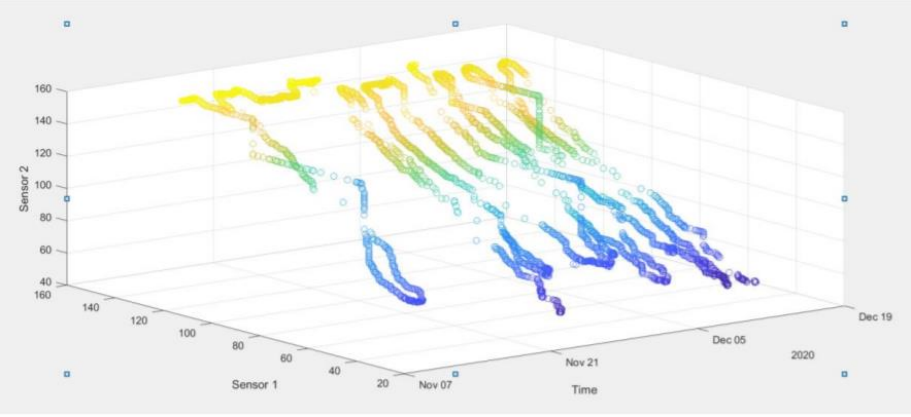
Format Resize Background color Column min/max

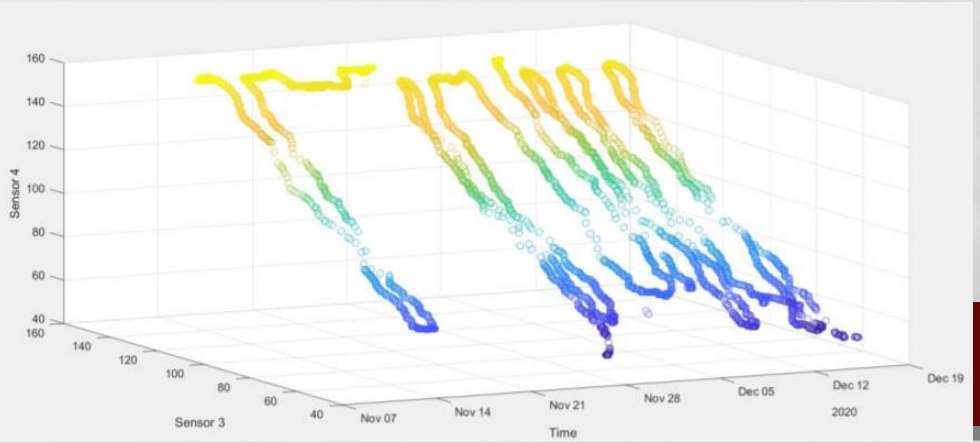
*sensor values are normalized

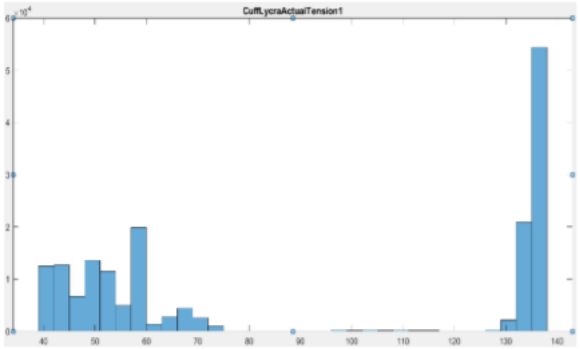
*timeLeft columns created for the time left until the next elastic breakout

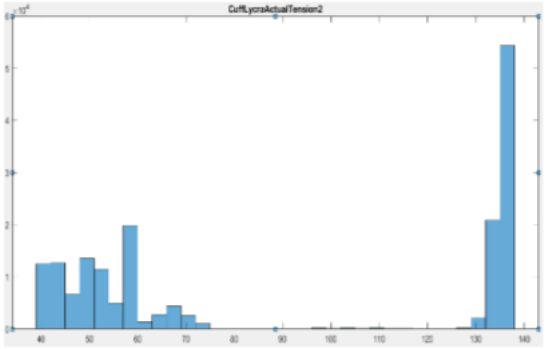
*some corrupted data was cleansed

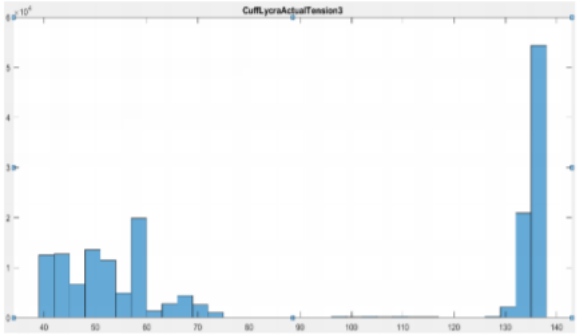
Materialization EEE

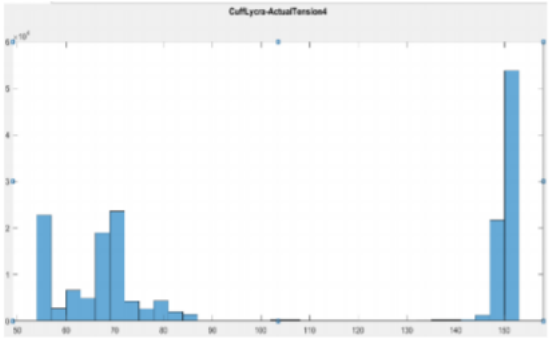




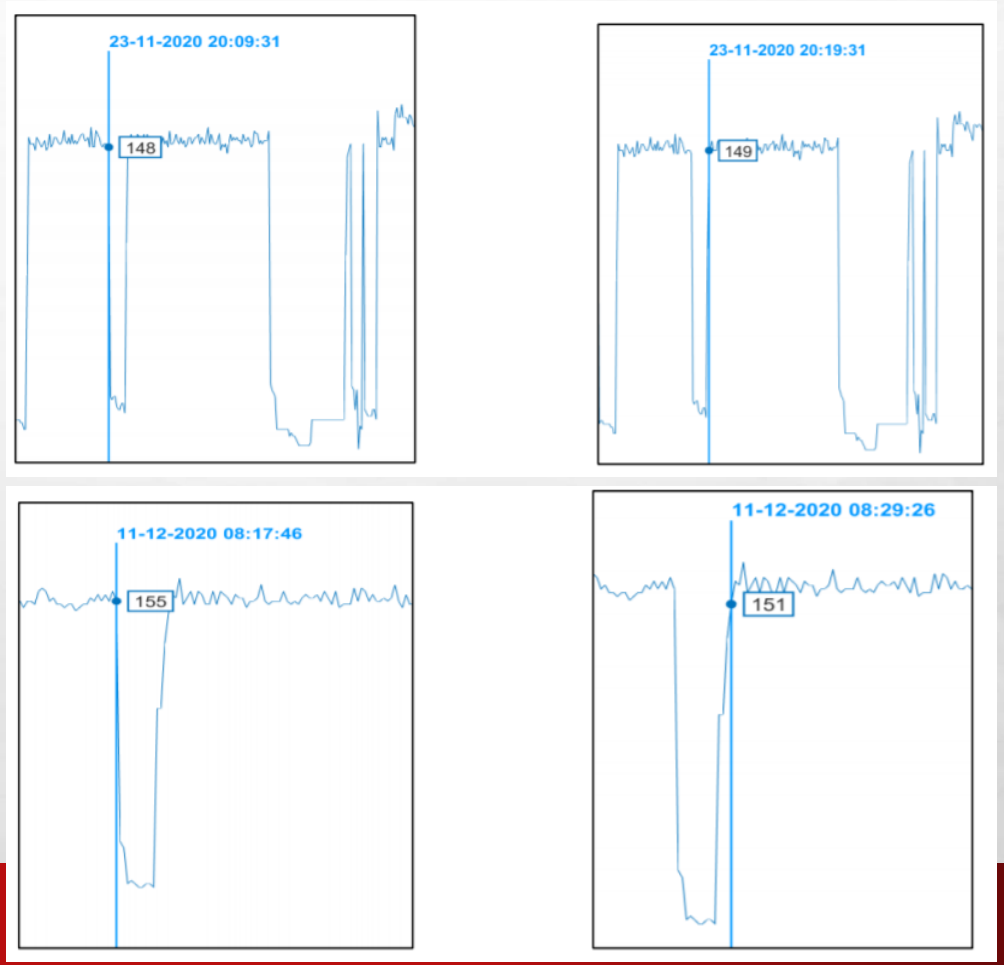

CuffLycraActualTension1


CuffLycraActualTension2


CuffLycraActualTension3


CuffLycra-ActualTension4

- Elastic breakout stops.
- Duration 10-15 minutes.
- Sudden drop.



Pre-Implementation SEN

Cross-Validation

- Pipelines
- GridSearchCV

```
# Pipeline and Parameters - MLP Regression

pipe_neural = Pipeline([('scl', StandardScaler()),
                        ('clf', MLPRegressor(max_iter=2000))])

param_neural = {'clf__alpha': [0.001, 0.01, 0.1, 1, 10, 100],
                'clf__hidden_layer_sizes': [(5),(10,10),(7,7,7)],
                'clf__solver': ['lbfgs'],
                'clf__activation': ['relu', 'tanh'],
                'clf__learning_rate' : ['constant', 'invscaling']}
```

```
grid_obj = GridSearchCV(estimator=pipeline,
                        param_grid=parameters,
                        cv=3,
                        scoring='r2',
                        verbose=2,
                        n_jobs=1,
                        refit=True)
grid_obj.fit(x_train, y_train)
```

Models:

1- **Ridge Regression**
2- Linear Regression
3- Lasso Regression
4- Polynomial Regression
5- MLP Regression
6- **KNN Regression**
7- Random Forest Regression
8- **Decision Tree Regression**
9- **XGBoost Regression**
10-**LightGBM Regression**

R-Squared: **0.18** ----> **0.78**

Materialization SEN

DECİSİON TREE & RANDOM FOREST REGRESSİON

Libraries

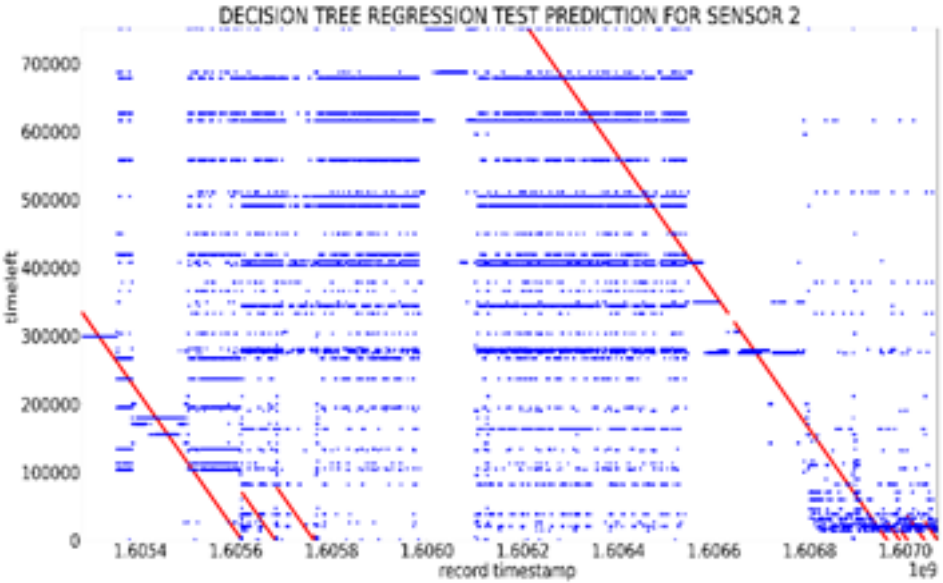
- pandas & numpy
- matplotlib.pyplot
- from sklearn.model_selection, train_test_split
- from sklearn.tree, DecisionTreeRegressor
- from sklearn.ensemble, RandomForestRegressor
- from sklearn, metrics and from sklearn.metrics r2_score
- & mean_absolute_percentage_error

General Information

- Train set data fraction as 0.1
- Test size is 0.1
- Four parameters considered:
  max_dept & random_state & max_leaf_nodes & n_estimators

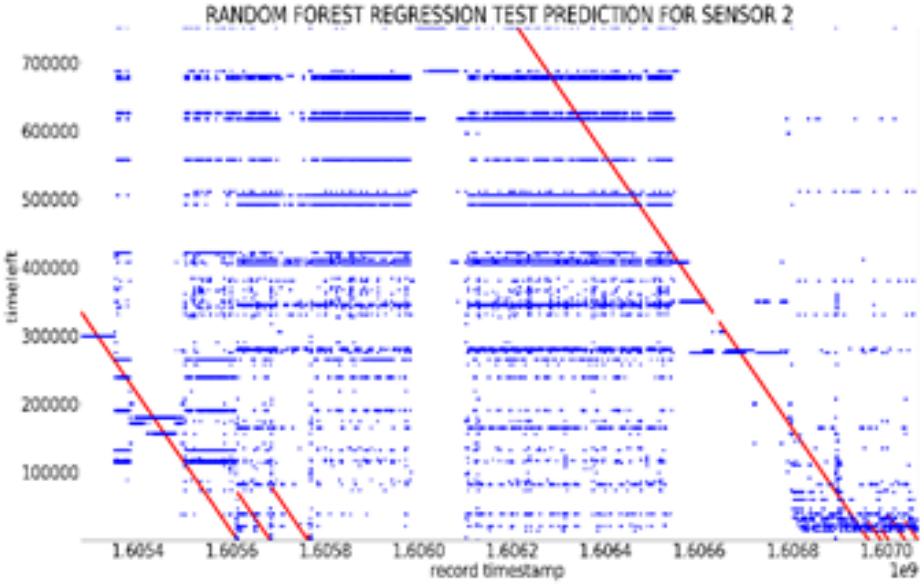
- Four error scores: MAE, MSE, RMSE, R2, MAPE
- Four results: TP, FP, TN, FN

Decision Tree



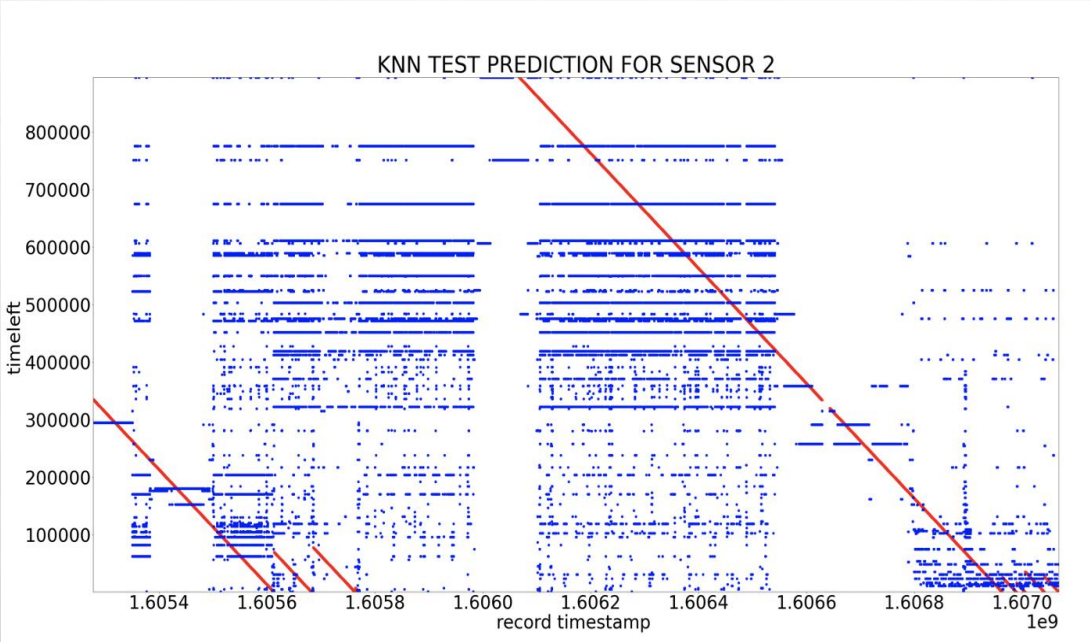
RSE	37%
TP	0.14%
FP	0.01%
TN	97.85%
FN	2.00%

Random Forest



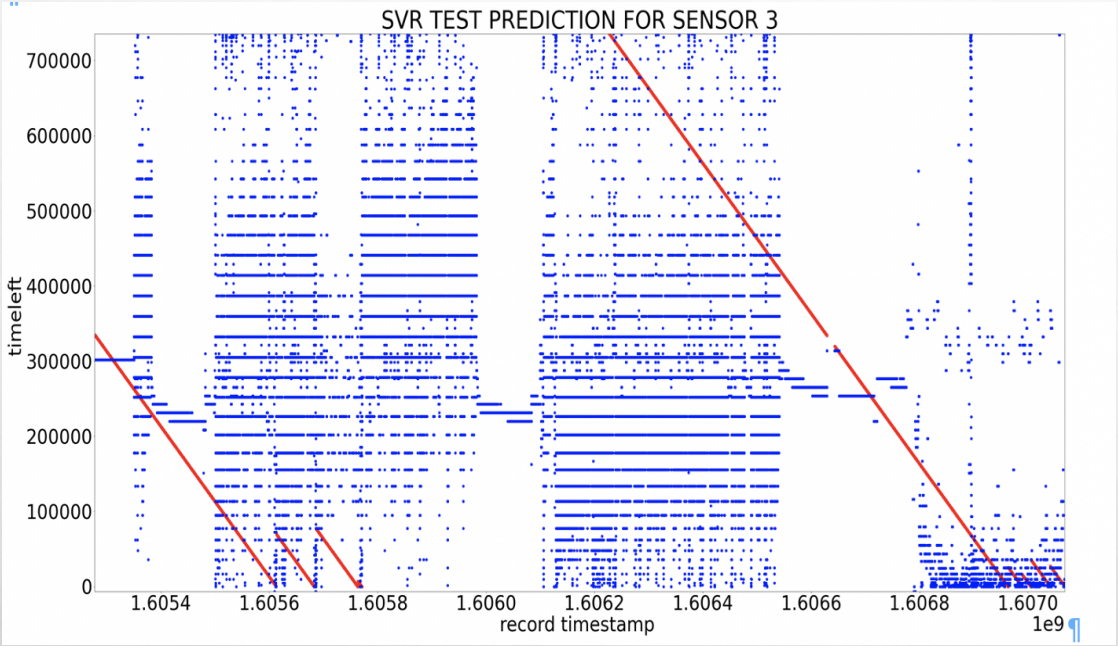
RSE	37%
TP	0.14%
FP	0.01%
TN	97.86%
FN	2.00%

KNN



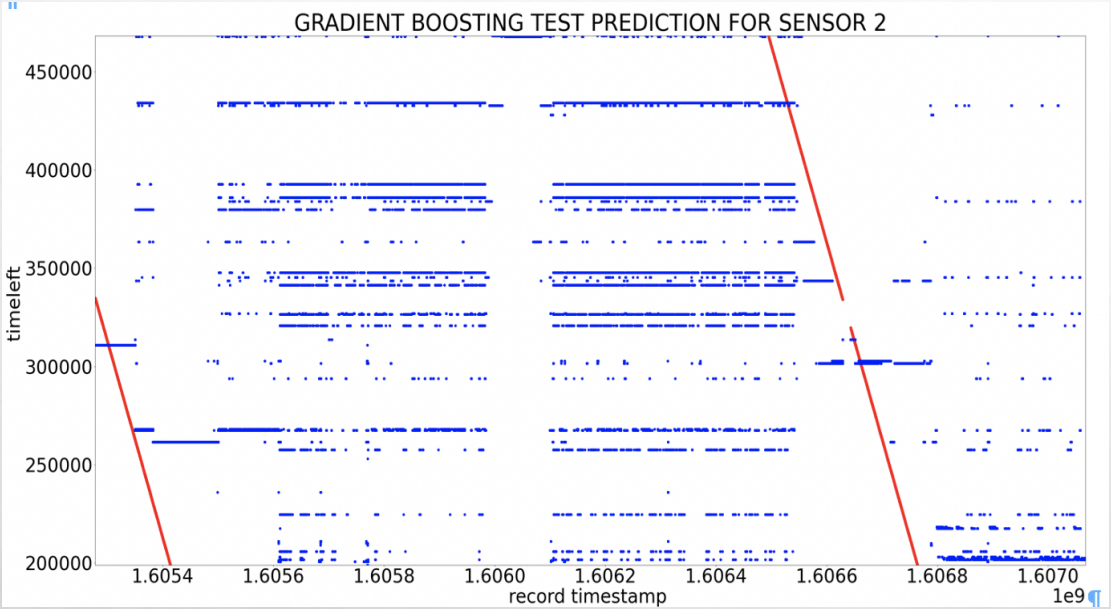
RSE	37%
TP	0.28%
FP	0.04%
TN	97.82%
FN	1.86%

SVR



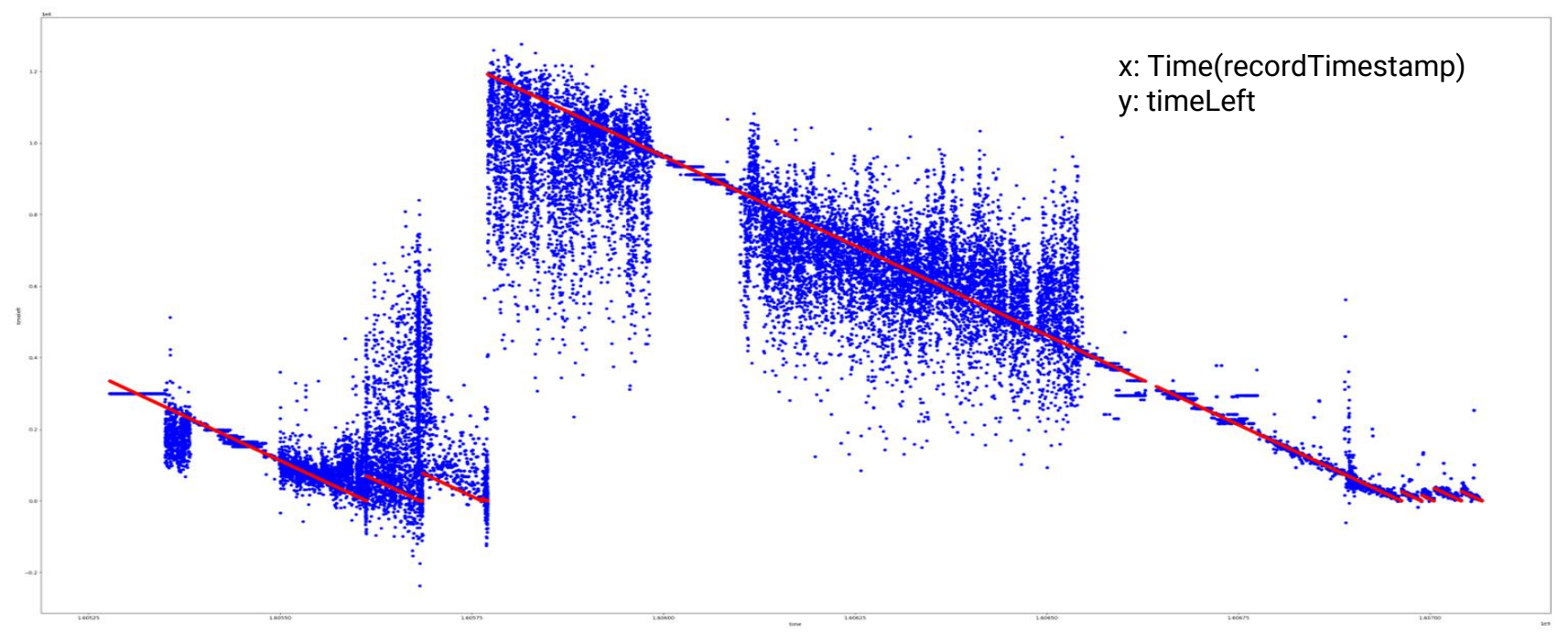
RSE	-0.06%
TP	0.31%
FP	3.67%
TN	94.20%
FN	1.82%

GRADIENT BOOSTING



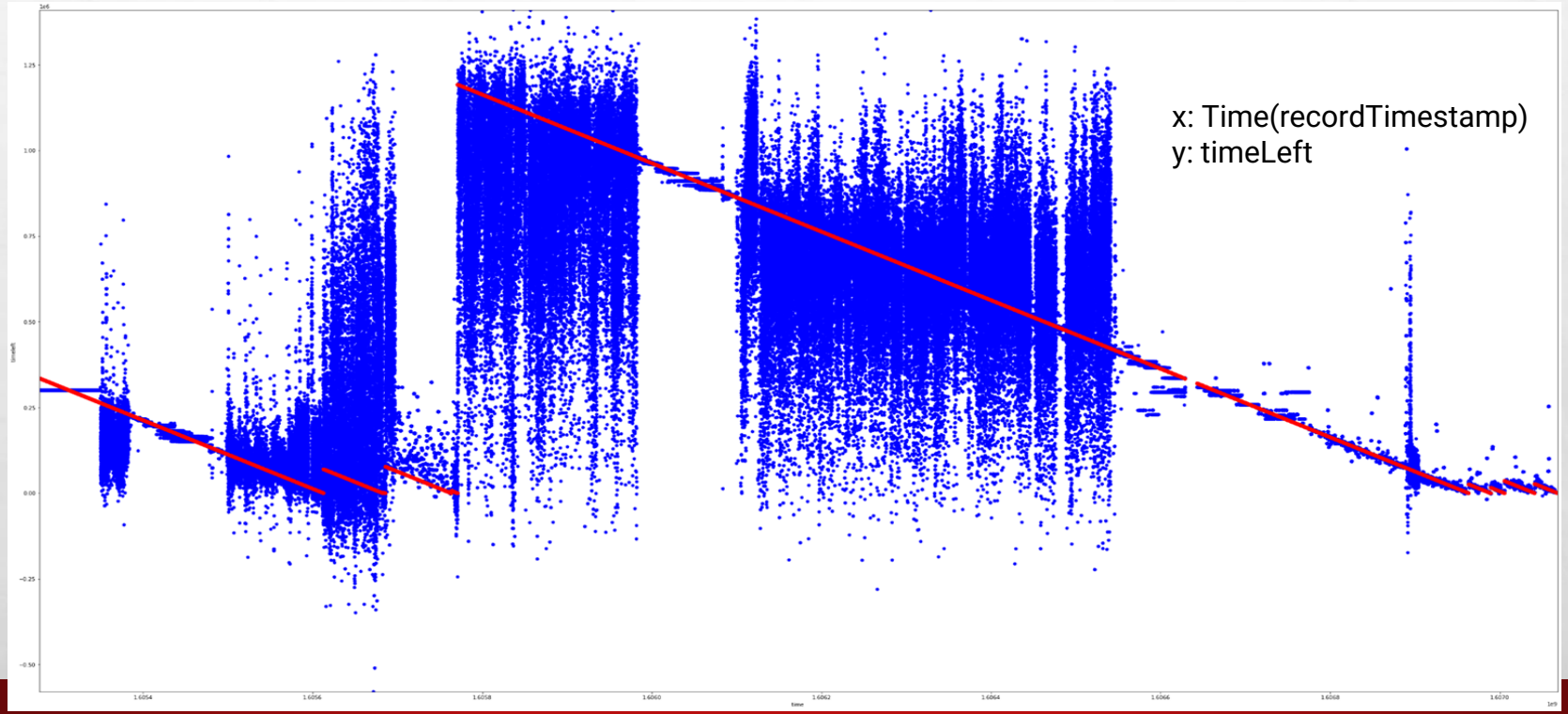
RSE	16%
TP	0%
FP	0%
TN	97.87%
FN	2.13%

LightGBM TRAIN SET



LightGBM

TEST SET



Highest score sensor error

Algorithm Name	Sensor Name	R2E	RMSE	MSE	MAE	MAPE
KNN	2	0,37	293366,2	8,61E+10	203844	16*10^17
SVR	3	-0,06	381630,7	1,46E+11	286255	17*10^17
DECISION TREE	2	0,37	294573,9	8,68E+10	207185	15*10^17
RANDOM FOREST	2	0,37	294469,7	8,67E+10	207145	15*10^17
GRADIENT BOOSTING	2	0,16	338516,1	1,15E+11	264541	21*10^17
XGBOOST	2	0,395	294463,0	8,67E+10	207107	15*10^17
MLP	2	0,346	329597,0	1,09E+11	253119	23*10^17
LIGHTGBM	1,2,3,4	0,7819	172771,8	2,99E+10	90249	2,13

Algorithm Name	TP	FP	TN	FN	TP + TN	FP +FN
KNN	0,28	0,04	97,82	1,86	98,10	1,90
SVR	0,31	3,67	94,20	1,82	94,51	5,49
DECISION TREE	0,14	0,01	97,85	2,00	97,99	2,01
RANDOM FOREST	0,14	0,01	97,86	2,00	98,00	2,00
GRADIENT BOOSTING	0,00	0,00	97,87	2,13	97,87	2,13
XGBOOST	0,26	0,04	97,83	1,87	98,09	1,91
MLP	0,00	0,00	97,87	2,13	97,87	2,13
LIGHTGBM	1,39	0,98	97,17	0,74	98,56	1,73

*This values are for timeLeft prediction before 3600 seconds

FILTERING

Finding valid ranges for sensors

sensor1				sensor2			
GENERAL		WORKING		GENERAL		WORKING	
0.225806	28733	1.000000	11771	0.290456	18548	0.614108	15944
0.233871	27093	0.830645	11214	0.614108	16084	0.609959	15893
0.217742	23856	0.806452	10942	0.609959	15925	0.618257	14862
1.000000	14469	0.822581	10914	0.618257	14916	0.605809	14453
0.830645	11368	0.790323	10706	0.605809	14481	0.622407	13742
sensor3				sensor4			
GENERAL		WORKING		GENERAL		WORKING	
0.410256	21547	0.764103	14519	0.323009	29556	0.659292	13257
0.420513	18432	0.769231	13521	0.327434	25220	0.654867	13094
0.425641	17137	0.758974	13372	1.000000	14418	0.663717	12815
0.430769	14998	0.753846	12823	0.659292	13309	1.000000	11720
0.764103	14583	0.779487	12237	0.654867	13106	0.650442	11608

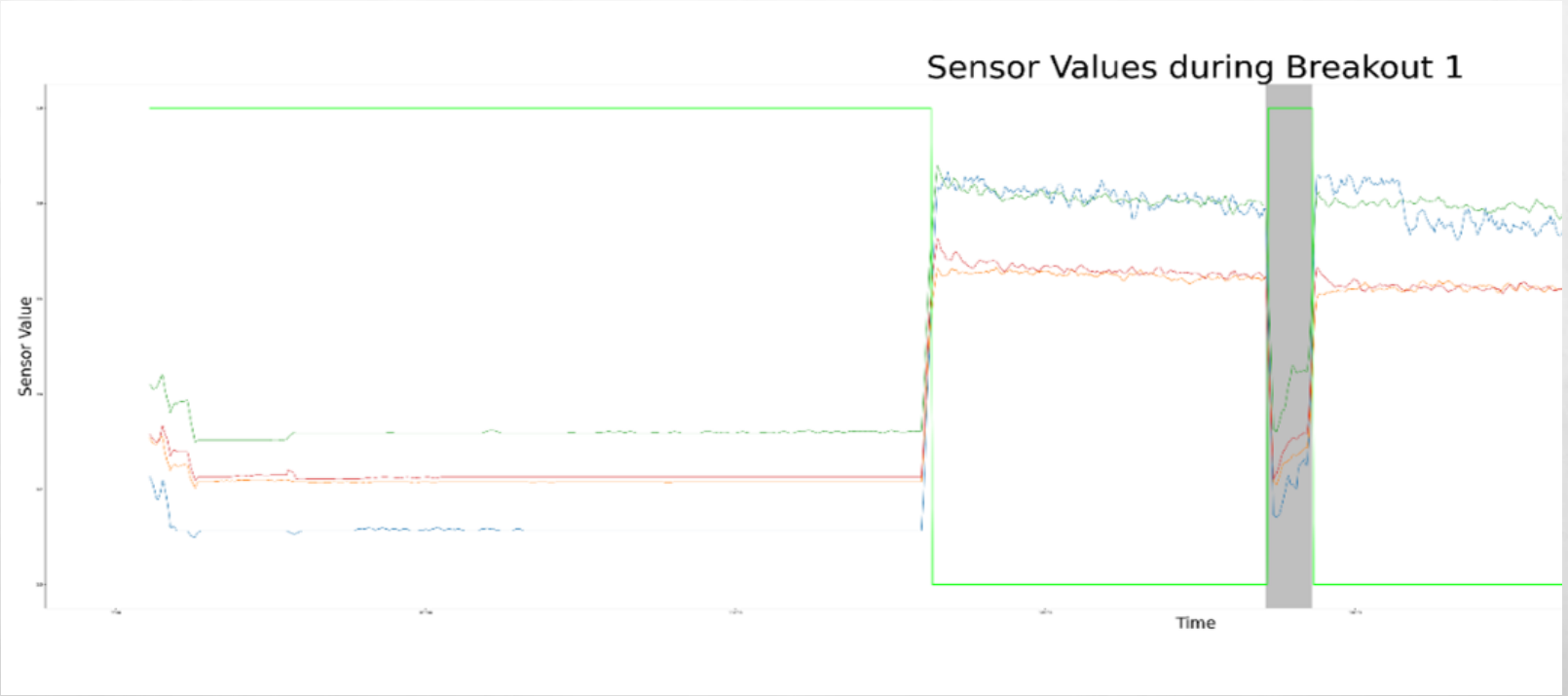
C_PROFICY_ConverterDTStartEnd'	
0	222265 (working)
1	129984
2	1718
3	519
4	222

Average While Working
s1 0,81
s2 0,61
s3 0,76
s4 0,65

Filtering - Some Values

```
s1 = 0.81
s2 = 0.61
s3 = 0.76
s4 = 0.65
for percentage in [0.1,0.15,0.2]:
    for index, row in df_filter.iterrows():
        if(s1*(1-percentage)<row['sensor1']<s1*(1+percentage) and
        s2*(1-percentage)<row['sensor2']<s2*(1+percentage) and
        s3*(1-percentage)<row['sensor3']<s3*(1+percentage) and
        s4*(1-percentage)<row['sensor4']<s4*(1+percentage)):
            df_filter.loc[index, 'filterWarning'] = False
        else:
            df_filter.loc[index, 'filterWarning'] = True
    print(f"{percentage} freedom")
    print("Machine Working")
    print(df_filter.loc[df['C_PROFICY_ConverterDTStartEnd'] == 0]['filterWarning'].value_counts())
    print("Machine NOT Working")
    print(df_filter.loc[df['C_PROFICY_ConverterDTStartEnd'] != 0]['filterWarning'].value_counts())
    print("=================================================")
```

Filtering - Visualization



Filtering Catches

0,2 Freedom	Breakouts	timeLeft (seconds)
	1	5
	2	2250
	3	10
	4	NO WARNING
	5	4743
	6	2435
	7	10
	8	NO WARNING
	9	15

Filtering Accuracy

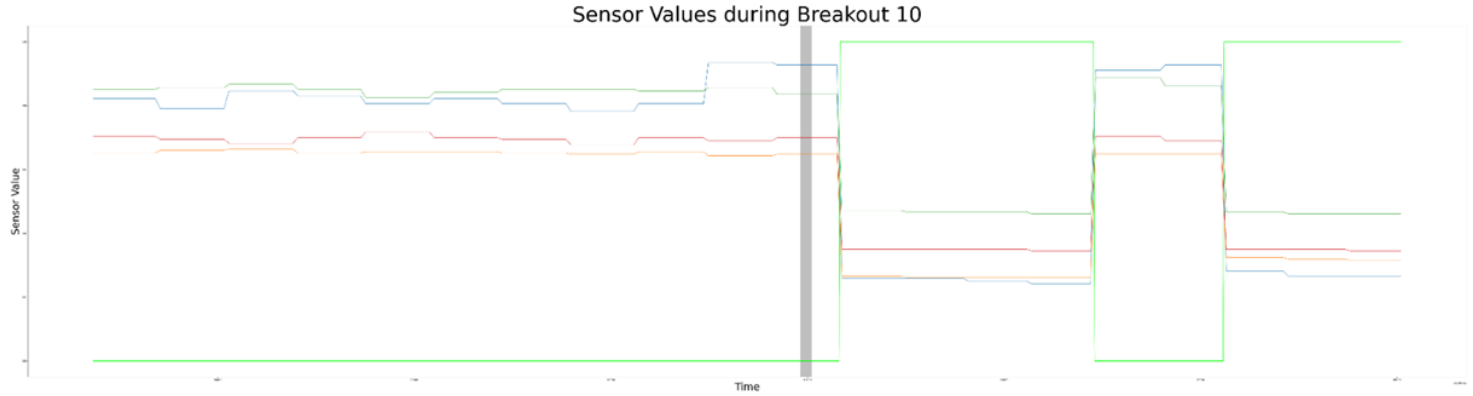
True Positive:1704 **0,48%**
False Positive:150485 **42,38%**
True Negative:200355 **56,42%**
False Negative:2546 **0,72%**

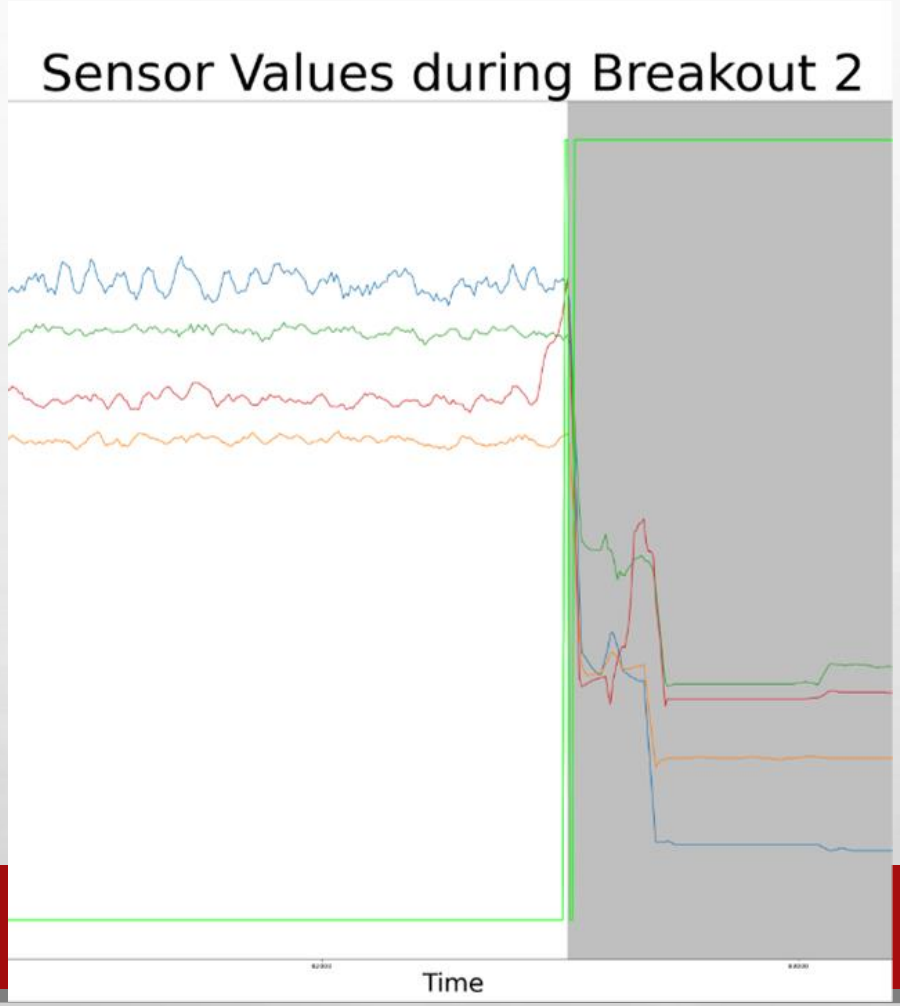
```
def getTPFPTNFN(y_true, y_pred):
    TP, FP, TN, FN = 0, 0, 0, 0
    for s_true, s_pred in zip (y_true, y_pred):
        if s_true <= 1800:
            if s_pred == True:
                TP += 1
            else:
                FN += 1
        else:
            if s_pred == False:
                TN += 1
            else:
                FP += 1

    print(f"True Positive:{TP}\nFalse Positive:{FP
    return TP, FP, TN, FN
```

These values are generated for timeLeft column with a constraint of 1800 seconds

Tons of false positive?


Sensor Values during Breakout 6
Sensor Value
Time


Sensor Values during Breakout 10
Sensor Value
Time


Sensor Values during Breakout 2
Time

Recommendations

Data about the breakouts should be organized, labeled more precisely.
Data errors such as high outlier rows should be avoided.
GMT difference between data and breakout records should be handled.
Some breakouts are recorded but there is no breakout in the data. This should be handled.

For training:
While training only for elastic breakouts, only elastic breakouts and normal state data should be trained. Since there are many breakouts algorithms tend to learn them too.

While modeling, sensor values should be used together in group of four.

Recommendations (cont)

XGBoost requires much more care in setting up than **Random Forest**, you can apply RF without that much of care and still end up with decent performance.

MLP algorithm is working so well with complex non-linear problems (similar to our problem). However, handling the computations was time consuming.

SVM solves complex problems with a convenient kernel solution function. However, it could be hard to understand the output because of the problems caused by the weight of variables (weight of variables are not constant).

Decision Tree model is very intuitive and easy to explain to stakeholders. On the other hand, it's inadequate for applying regression and predicting continuous values.

LightGBM: There were no necessary recommendations about framework itself, but the data should be cleaned and worked in detail generally.

Summary & Conclusion

- In conclusion we have created some models and those models should be developed. Some models showed higher performance will be more focused.
- The perspective of accuracy will be changed in real life rather than mathematical accuracy.
- Models will be tuned for the highest accuracy.
- Models can be divided into two for less left to breakout and much left to breakout.
- Data will be investigated more for the breakouts with higher accuracy.
- Instead of little fractions will use larger data to train.
- Write an algorithm that runs the model and simulates the data as it was in real time.
- Further research on tuning will be made.