An-Najah National University

Department of Artificial Intelligence

Final Project

Data Mining and Analysis

Submitted By: Aladdin Husni Odeh

Submitted To: Dr. Anas Toma

Contents

Table of figures:

1. Introduction

1.1. Objectives

In this report, we are going to analyze a dataset that shows the strength of a set of buildings, and it shows different spatial features that represent some metrics of the buildings on different positions and sides. The goal is to analyze and understand the relationship between these features on the different positions to build a model that can predict the actual output of the strength.

The actual strength of the buildings wasn't generated based on these features; it was measured using an expensive device, and the features were measured using a reasonably priced device. So, the goal here is trying to fit these features into a model that can replace the need for the expensive device.

1.2. Dataset overview

This is spatial data with a set of features (F1…F5), where each set of features represents a specific position on one side of the building.

The building has two sides, named G1 and G2, and each side has seven positions. There are two datasets with the same structure, and each of them represents a set of 36 different buildings.

The dataset obtained from an unknown resource has 36 records with the following features:

Variable Name	Role	Type	Description
ID	ID	Integer	From 1 to 36
Output Measured Value	Label	Integer	The strength of the building that we are aiming to predict.
Position	Feature	Categorical	The positions of the building from 1 to 7
D{x} G{y} F{z}	Feature	Integer	The main features of the dataset, where each combination of [x = Dataset number], [y = Group ID], and [Z = Feature number], represent a specific feature on a specific side at a specific position in the building

2. Data Preprocessing and Preparing

2.1. Data cleaning

The dataset is a very small one with only 36 buildings, so the initial steps was to observe it from the excel file directly. At the beginning, a small restructure has been conducted on the excel file to make it easier for python to load it in a form of data frame. This only includes preparing headers and renaming them with a format of Dx Gy Fz as shown in the image below:

	A	B	C	D	E	F	G	H
1	ID	Output Measured Value	Position	D1 G1 F1	D1 G1 F2	D1 G1 F3	D1 G1 F4	D1 G1 F5
2			1	24.85501	84.13925	23.8306	23.57176	1.531475
3			2	27.15659	52.1842	26.841	25.93047	3.56871
4			3	29.57864	45.10564	29.19554	24.9112	5.404616
5	1	8.3	4	26.07265	48.68284	24.99298	21.45438	4.923152
6			5	16.14923	66.64713	19.92993	20.11763	5.131219
7			6	14.88673	57.812	17.17827	15.95823	2.279877
8			7	16.6264	73.58484	17.86797	14.72317	3.186924
9			1	16.16932	63.39901	16.69973	20.67246	2.158096
10			2	15.87909	66.58972	16.16727	19.38002	2.761579
11			3	15.18531	80.41343	16.33961	19.14008	1.758195
12	2	8.68	4	14.55824	66.30258	15.20158	18.89529	2.674262
13			5	15.39873	64.77441	15.08598	19.11876	3.067592
14			6	20.93369	73.29805	19.3893	20.73887	1.003171
15			7	20.54933	68.12781	20.03993	22.1257	2.628585
16			1	17.93428	73.85178	16.79595	20.88575	0.950602
17			2	18.92172	83.49246	17.56046	21.69961	2.694277
18			3	17.28816	79.83968	16.80738	21.93547	2.395656
19	3	10.83	4	15.93081	74.46685	15.77129	20.94336	1.350689
20			5	18.34884	86.26861	18.58388	21.24209	2.120961
21			6	20.60764	96.55631	21.89577	22.36062	1.815434
22			7	22.81495	83.27465	21.97139	24.76776	2.004176

	I	J	K	L	M	N	O	P
1	D1 G2 F1	D1 G2 F2	D1 G2 F3	D1 G2 F4	D1 G2 F5	D2 G1 F1	D2 G1 F2	D2 G1 F3
2	22.43399	94.84443	26.2126	20.98678	2.067586	34.112	44.97622	24.1065
3	23.18599	106.9746	29.49597	22.60616	2.002584	38.62159	47.46208	25.9995
4	21.13815	111.087	27.88892	20.9351	2.536556	39.07217	49.26732	25.2507
5	18.58767	114.6899	21.04436	21.37583	5.015616	33.25973	50.68517	20.8406
6	19.05001	93.59177	17.06553	24.4271	2.59637	28.41966	44.36182	19.9844
7	15.65384	84.1202	15.65045	19.03427	3.769469	25.54367	46.98648	17.3556
8	14.06457	87.75506	13.66539	14.37028	3.639428	24.01689	49.56642	15.4388
9	20.25814	68.55585	20.73532	21.41214	3.140666	26.74959	39.52349	20.5664
10	18.665	77.13583	19.50243	18.8823	3.246029	25.618	38.39444	19.9612
11	19.58165	78.10972	20.13737	18.65518	2.895852	25.72127	42.65132	18.8735
12	19.98356	75.7209	19.92477	19.24261	3.005394	24.45156	39.30778	18.806
13	15.81561	71.23519	16.26104	18.03752	3.734387	25.02045	38.59473	19.4079
14	11.48917	65.48701	12.41408	15.20945	2.905167	30.30414	43.62489	21.9243
15	14.33105	70.68661	14.30829	16.93907	1.530356	29.91012	41.91068	22.1726
16	17.0473	65.29826	17.31546	21.18155	1.92155	26.58303	39.96562	20.361
17	17.90525	71.48596	18.05292	21.37447	1.606743	27.85878	39.71054	21.3308
18	17.03895	76.40328	17.58564	22.93866	2.681769	27.85647	37.51609	22.0134
19	17.43322	66.20949	18.47551	24.67518	2.662222	26.75088	38.15507	21.0085
20	17.62904	67.71284	19.20693	23.99745	1.795516	28.70354	40.53688	21.7547
21	18.30047	65.84942	18.97637	22.00734	2.982224	31.45575	45.38446	22.056
22	16.68755	77.46877	17.94574	19.80976	2.608421	33.12891	41.77757	24.6602

	Q	R	S	T	U	V	W
1	D2 G1 F4	D2 G1 F5	D2 G2 F1	D2 G2 F2	D2 G2 F3	D2 G2 F4	D2 G2 F5
2	24.0865	1.5315	33.37192	51.03957	20.9434	25.8995	2.0676
3	28.3358	3.5687	36.27009	52.09718	22.2476	28.5754	2.0026
4	29.3228	5.4046	35.12866	52.76409	21.2008	27.8947	2.5366
5	25.4488	4.9232	30.43268	44.88811	21.2663	21.1834	5.0156
6	19.5441	5.1312	30.75547	35.01132	25.1	17.5826	2.5964
7	18.6028	2.2799	24.94874	40.22407	18.8303	15.9264	3.7695
8	18.119	3.1869	21.17693	44.62926	14.8467	14.6558	3.6394
9	16.9678	2.1581	29.57069	45.59346	20.5749	21.0056	3.1407
10	15.8179	2.7616	28.21324	47.70601	18.8596	20.7308	3.246
11	17.3863	1.7582	28.44138	45.91905	19.6831	20.3249	2.8959
12	15.3968	2.6743	28.60204	46.12603	19.7136	20.5041	3.0054
13	15.4902	3.0676	24.83464	42.2206	18.1825	16.4988	3.7344
14	20.8964	1.0032	19.98612	39.08812	15.348	12.4677	2.9052
15	19.9018	2.6286	22.83939	40.96998	17.2062	14.9413	1.5304
16	17.0641	0.9506	27.65552	38.68452	21.5357	17.2438	1.9216
17	17.7158	2.6943	28.13574	41.01573	21.1946	18.4344	1.6067
18	16.9013	2.3957	28.87328	40.17517	21.966	18.5464	2.6818
19	16.5054	1.3507	30.65439	37.49314	24.2301	18.5878	2.6622
20	18.6045	2.121	30.63003	39.21527	23.6906	19.3321	1.7955
21	22.354	1.8154	28.74712	41.33447	21.4688	18.8837	2.9822
22	22.0314	2.0042	27.01953	42.77754	19.7396	18.2647	2.6084

Figure 1: Restructured dataset

The data had only 1 negative value which was converted to positive since it was clearly a mistake.

2.2. Introducing binary and multi classification features

Based on the description of the data, there are no guarantees for any relationship between the features and the label, so 2 new features were introduced for classification experimenting a long side with the regression experimentations.

a- **'Strength_2' feature**: This feature splits the output into 2 categories 'Weak' and 'Standard'. A threshold of 6 was chosen based on an assumption that most probably the data should have more acceptable building compared to defected ones.

b- **'Strength_3' feature**: This feature splits the output into 3 categories 'Weak', 'Standard', and 'Great'. Same threshold used to split Weak from other classes, and a threshold of 12.5 was chosen to split 'Great' from the rest.

	ID	Output Measured Value	Strength_2	Strength_3	Position	D1 G1 F1	D1 G1 F2	D1 G1 F3	D1 G1 F4	D1 G1 F5
0	1.0	8.3	Standard	Standard	1	24.855010	84.139247	23.830600	23.571756	1.531475
1	1.0	8.3	Standard	Standard	2	27.156592	52.184199	26.840996	25.930472	3.568710
2	1.0	8.3	Standard	Standard	3	29.578644	45.105642	29.195542	24.911200	5.404616
3	1.0	8.3	Standard	Standard	4	26.072645	48.682841	24.992980	21.454382	4.923152
4	1.0	8.3	Standard	Standard	5	16.149233	66.647127	19.929927	20.117635	5.131219

Figure 2: Dataset1 Group1 features with classification features.

2.3. Initial Assessment

The first part of the assessment was to plot a histogram to show the distribution of all features across both data sets as follows:

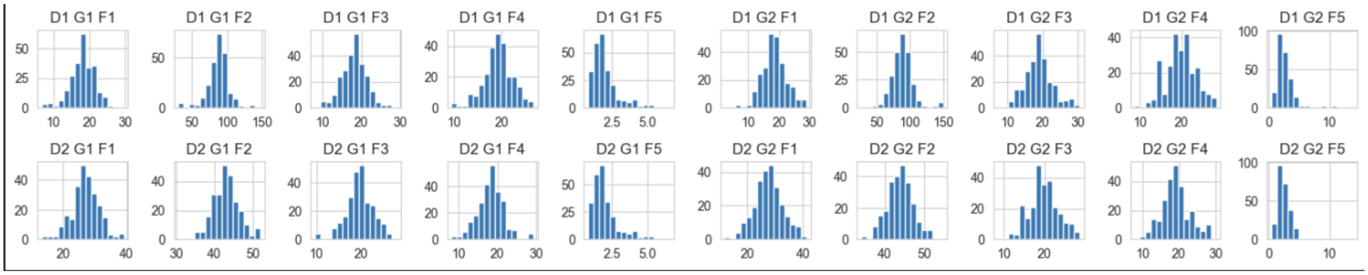


Figure 3: Features histogram in both data sets

In the first row, we can see all the 10 features (2 group x 5) and in the second row we can see the distribution of the same features in the second data set.

- The histogram shows a very similar distribution between both data sets which is a good start to start experimenting on them separately.
- The first 4 features have a normal distribution compared to F5 which is skewed to the left.

The second part of the assessment was to show scatter plots between each feature in each data set with the ‘Output Measured Value’. The aim of this analysis to understand if there is maybe any direct linear relationship with the output.

Figure 4: Scatter plot between features and Output

The results indicate the following:

1- There is no sign of any linear relationship between any of the features with the output which suggests a need of feature engineering to extract complex relationships that could potentially fit the data in a model.
2- F5 in both datasets is identical.
3- The feature in both datasets looks very similar with small differences, mainly some shifts and scaling changes which implies a need for a normalization.

The third step of the initial assessment was plotting a boxplot to understand the ranges of the dataset and to detect any outliers:

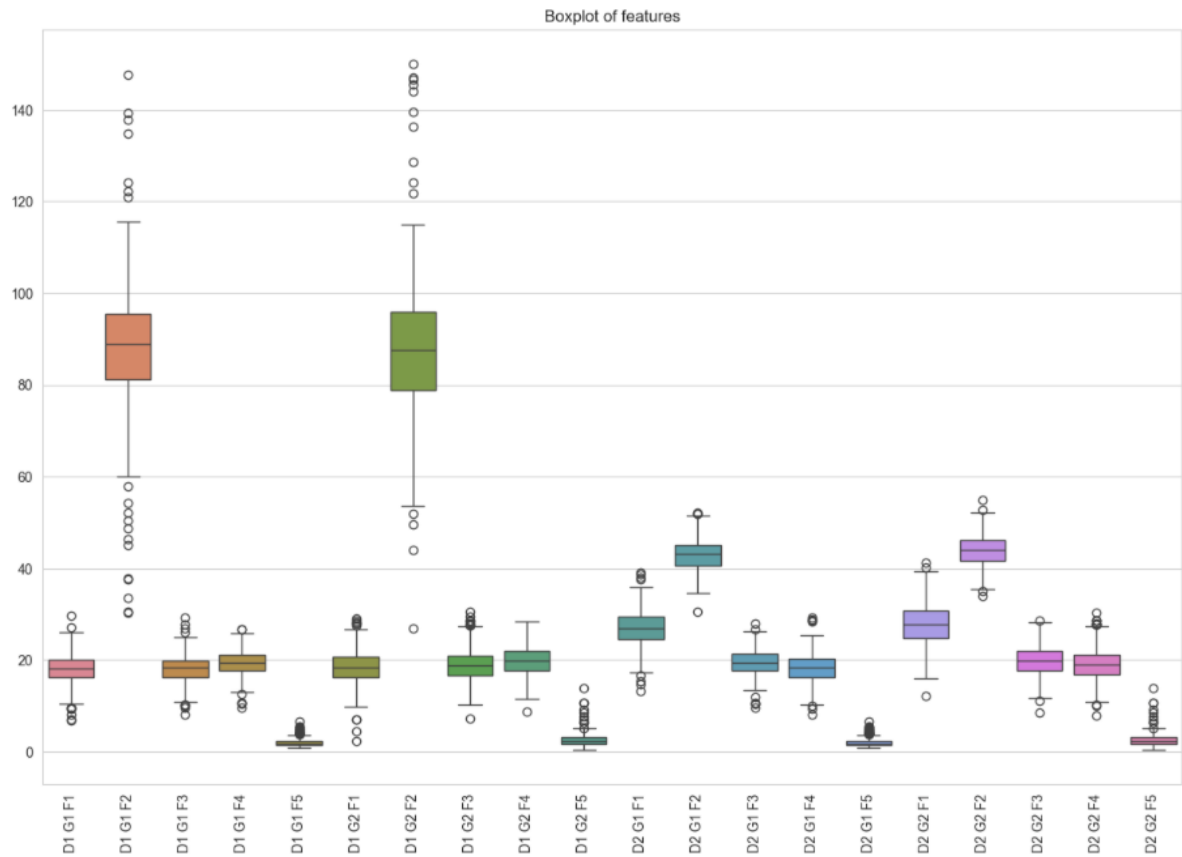


Figure 5: Boxplot for all features in the 2 datasets

The image shows various insights:

1- The range of F2 in the first dataset is significantly higher compared to all other features which again implies the need of some kind of normalization.
2- There are multiple outliers which could be useful in feature engineering to introduce new features related to outliers, but since the data sets are very small, outliers might not be very useful.

The last assessment was to draw a correlation matrix to see in a numeric readable way how much these features are correlated to the output:

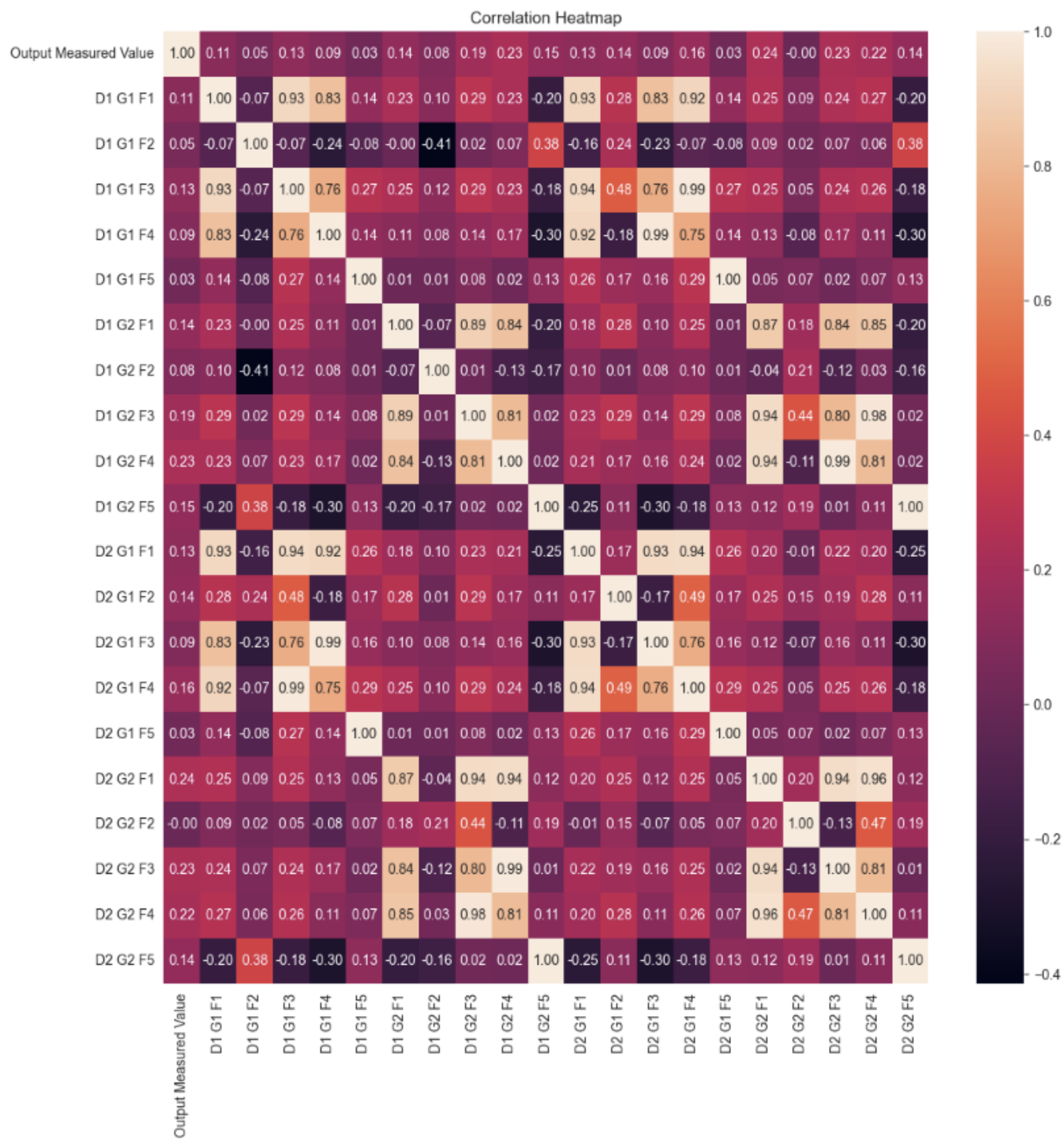


Figure 6: Correlation matrix between features

The correlation matrix shows a low correlation between all the features and the Output value. It also shows a high correlation between F1, F3 and F4 which.

It also shows high correlation between the same feature in different data sets.

2.4. Preparing multiple data frames for experimenting

The first step in data preparing was to normalize all values in all features as the initial assessment suggests unifying the scale and the shape of the data to avoid any possible issues while experimenting models.

Since the original data frame contains 2 different datasets in a form of spatial data. It was crucial to make various splits to make it ready for different explorations and experimenting. And the following data frames has been introduced initially:

1- ‘df1’ and ‘df2’ where each data frame represents a different data set from the 2 data sets available.
2- ‘df1g1’, ‘df1g2’, ‘df2g1’, and ‘df2g2’ where each data frame represents one side of the buildings for each data set.
3- ‘flattened_df1’ and ‘flattened_df1’ to convert the data into a simple 36 rows suitable for model trainings. These data frames were created by changing the feature format to include position, for example, ‘D1 G1 F1 P1’ which will represent F1 in position 1 in the first group of a building in the first data set.
4- ‘flattened_df1g1’ and ‘flattened_df1g1’

	ID	Output Measured Value	Strength_2	Strength_3	D1 G1 F1 P1	D1 G1 F1 P2	D1 G1 F1 P3	D1 G1 F1 P4	D1 G1 F1 P5	D1 G1 F1 P6	...	D1 G2 F4 P5	D1 G2 F4 P6	D1 G2 F4 P7	D1 G2 F5 P1	D1 G2 F5 P2	D1 G2 F5 P3	D1 G2 F5 P4	D1 G2 F5 P5	D1 G2 F5 P6	D1 G2 F5 P7
0	1.0	8.30	Standard	Standard	0.793402	0.894067	1.000000	0.846658	0.412638	0.357420	...	0.799299	0.525732	0.289139	0.124061	0.119244	0.158815	0.342534	0.163248	0.250184	0.240547
1	2.0	8.68	Standard	Standard	0.413516	0.400822	0.370479	0.343052	0.379813	0.621895	...	0.475169	0.331708	0.419448	0.203585	0.211393	0.185442	0.193560	0.247584	0.186132	0.084248
2	3.0	10.83	Standard	Standard	0.490710	0.533898	0.462451	0.403085	0.508842	0.607635	...	0.777503	0.676549	0.565071	0.113238	0.089909	0.169577	0.168128	0.103898	0.191843	0.164141
3	4.0	14.26	Standard	Great	0.605390	0.548463	0.535641	0.511731	0.432102	0.442521	...	0.755138	0.753249	0.591208	0.098212	0.136095	0.141034	0.118873	0.141431	0.238712	0.149820
4	5.0	13.17	Standard	Great	0.277146	0.310052	0.475830	0.643505	0.764163	0.768902	...	0.685484	0.578266	0.315869	0.131804	0.110324	0.135918	0.142337	0.224774	0.256621	0.144457

5 rows × 74 columns

	ID	Output Measured Value	Strength_2	Strength_3	D1 G1 F1 P1	D1 G1 F1 P2	D1 G1 F1 P3	D1 G1 F1 P4	D1 G1 F1 P5	D1 G1 F1 P6	...	D1 G1 F4 P5	D1 G1 F4 P6	D1 G1 F4 P7	D1 G1 F5 P1	D1 G1 F5 P2	D1 G1 F5 P3	D1 G1 F5 P4	D1 G1 F5 P5	D1 G1 F5 P6	D1 G1 F5 P7
0	1.0	8.30	Standard	Standard	0.793402	0.894067	1.000000	0.846658	0.412638	0.357420	...	0.614874	0.373309	0.301581	0.128595	0.479431	0.795595	0.712682	0.748513	0.257479	0.413683
1	2.0	8.68	Standard	Standard	0.413516	0.400822	0.370479	0.343052	0.379813	0.621895	...	0.556863	0.650954	0.731497	0.236507	0.340434	0.167639	0.325397	0.393133	0.037615	0.317530
2	3.0	10.83	Standard	Standard	0.490710	0.533898	0.462451	0.403085	0.508842	0.607635	...	0.680179	0.745140	0.884939	0.028562	0.328843	0.277417	0.097462	0.230112	0.177496	0.210000
3	4.0	14.26	Standard	Great	0.605390	0.548463	0.535641	0.511731	0.432102	0.442521	...	0.443215	0.475250	0.527621	0.166067	0.251307	0.232672	0.094050	0.216864	0.244507	0.103832
4	5.0	13.17	Standard	Great	0.277146	0.310052	0.475830	0.643505	0.764163	0.768902	...	0.773964	0.740325	0.663805	0.067080	0.057837	0.265536	0.324943	0.179663	0.181633	0.148741

5 rows × 39 columns

	ID	Output Measured Value	Strength_2	Strength_3	D1 G2 F1 P1	D1 G2 F1 P2	D1 G2 F1 P3	D1 G2 F1 P4	D1 G2 F1 P5	D1 G2 F1 P6	...	D1 G2 F4 P5	D1 G2 F4 P6	D1 G2 F4 P7	D1 G2 F5 P1	D1 G2 F5 P2	D1 G2 F5 P3	D1 G2 F5 P4	D1 G2 F5 P5	D1 G2 F5 P6	D1 G2 F5 P7
0	1.0	8.30	Standard	Standard	0.751515	0.779498	0.703294	0.608386	0.625590	0.499212	...	0.799299	0.525732	0.289139	0.124061	0.119244	0.158815	0.342534	0.163248	0.250184	0.240547
1	2.0	8.68	Standard	Standard	0.670547	0.611263	0.645373	0.660329	0.505232	0.344237	...	0.475169	0.331708	0.419448	0.203585	0.211393	0.185442	0.193560	0.247584	0.186132	0.084248
2	3.0	10.83	Standard	Standard	0.551065	0.582991	0.550755	0.565426	0.572713	0.597698	...	0.777503	0.676549	0.565071	0.113238	0.089909	0.169577	0.168128	0.103898	0.191843	0.164141
3	4.0	14.26	Standard	Great	0.624161	0.714242	0.770394	0.779153	0.779467	0.781980	...	0.755138	0.753249	0.591208	0.098212	0.136095	0.141034	0.118873	0.141431	0.238712	0.149820
4	5.0	13.17	Standard	Great	0.882642	0.938934	0.955381	0.861441	0.736587	0.576671	...	0.685484	0.578266	0.315869	0.131804	0.110324	0.135918	0.142337	0.224774	0.256621	0.144457

5 rows × 39 columns

Figure 7: Flattened normalized data

3. Data analysis

3.1. Position analysis

3.1.1. Mean and Std aggregation.

A simple way to tackle the spatial data instead of flattening it, is aggregating the features on the positions, by calculating the mean, standard deviation for every feature in every building across all positions.

A new data frame with 36 rows was created with 40 features (10 x 2 (data sets) x 2 (mean and std) for each feature). Then the correlation was calculated between these new features and the Output.

```
Output Measured Value    1.000000
D2 G2 F1 mean            0.376837
D2 G2 F3 mean            0.372496
D1 G2 F4 mean            0.363388
D2 G2 F4 mean            0.342215
D1 G2 F1 std             0.311229
D1 G2 F3 mean            0.310664
D1 G2 F3 std             0.307222
D2 G2 F4 std             0.290881
D2 G1 F4 mean            0.262590
D1 G1 F4 std             0.243388
D1 G2 F1 mean            0.239459
D2 G2 F1 std             0.234143
D2 G1 F1 std             0.221904
D1 G1 F3 mean            0.219015
D2 G1 F3 std             0.216539
D2 G1 F1 mean            0.214862
D2 G1 F2 mean            0.214445
D2 G1 F4 std             0.201963
D1 G1 F1 mean            0.191070
```

Figure 8: Correlation between aggregated features and the Output

The image above shows a better correlation for the aggregated features compared to each feature alone which suggest a relationship between different positions that impacts the overall output, and such relationship should be considered in model training.

3.1.2. Position differences, cumulative summation and multiplication.

Instead of extracting relationships that describe all positions together we could focus on relationships between the adjacent positions, like finding the difference in feature values between each 2 positions (P2 – P1, P3 – P2 … P7 – P6). We could also find the cumulative summation and multiplication for each position which could highlight some hidden insights.

For this purpose, a new data frame was created ‘df_with_diff_cum’ and the correlation between new features and the Output was as follows:

```
Output Measured Value    1.000000
D2 G2 F3_cumprod         0.365606
D1 G2 F4_cumprod         0.365120
D2 G2 F1_cumprod         0.335921
D2 G2 F4_cumprod         0.281699
D1 G2 F1_cumprod         0.281003
D1 G2 F3_cumprod         0.258043
D2 G2 F1_cumsum          0.218497
D2 G2 F3_cumsum          0.215232
D1 G2 F4_cumsum          0.212498
D2 G2 F4_cumsum          0.209268
D1 G2 F3_cumsum          0.195443
D1 G2 F1_cumsum          0.139065
D1 G2 F5_cumsum          0.101996
D2 G2 F5_cumsum          0.101848
D1 G1 F3_diff            0.088426
D2 G1 F4_diff            0.085283
D1 G1 F4_diff            0.079958
D2 G1 F1_diff            0.076903
D1 G1 F1_diff            0.068449
```

Figure 9: Correlation between adjacent positions and output

The result shows a similar correlation to the mean and std aggregation with better correlation on the cumulative operations which also suggests the spatial data relationship with output.

3.1.3. Distribution of Features by Position.

To better understand the ranges of each feature across all position, a boxplot for each feature in each position was plotted as following:

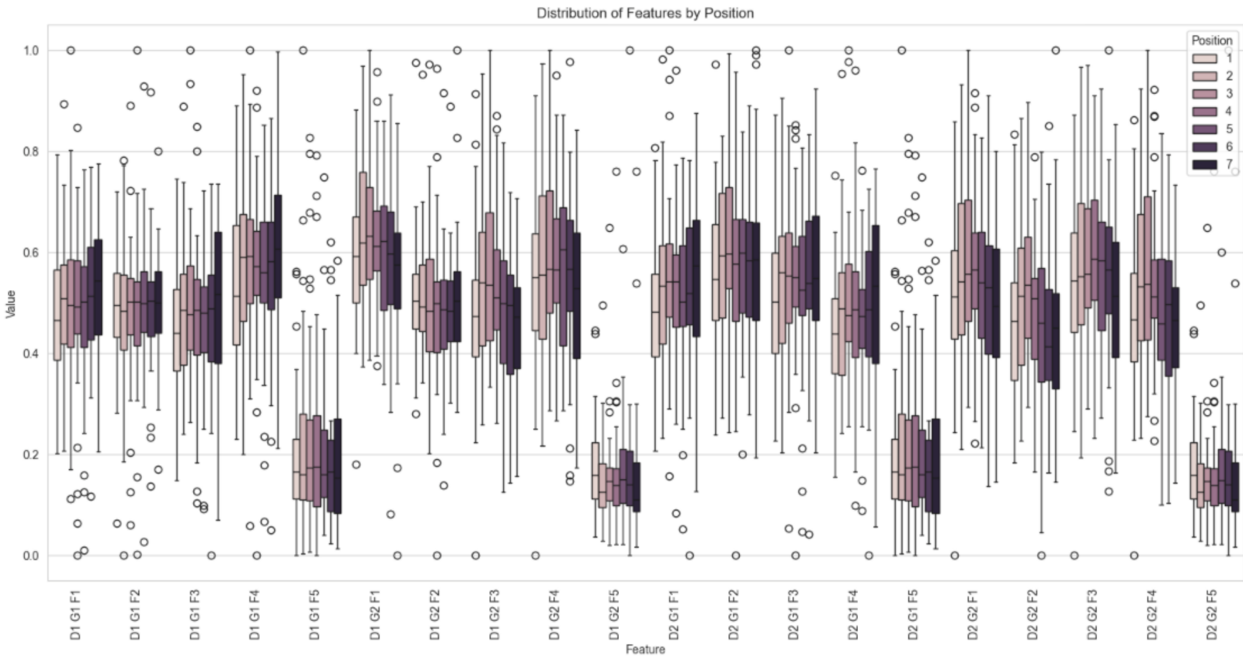


Figure 10: Boxplot of features per positions

The results show a similar distribution of data across all positions for each feature with small differences.

It also shows that the upper bound of values for the first group in the 7th position is higher than other features. Other than that, it shows that feature 5 most values are less than 0.3 with many outliers which might indicate some importance of these outliers for this specific feature in determining the Output.

3.1.4. Analyze similar buildings.

Since there are no strong relationships found, and since the data suggests a complex non-linear relationship between different features and positions, taking a sample of buildings with similar output values and analyze it could be useful. This approach could help us finding some kind of patterns that we could build on.

The image below shows a line chart with the values of each feature across each position for each one of the 3 building with similar output (~6):

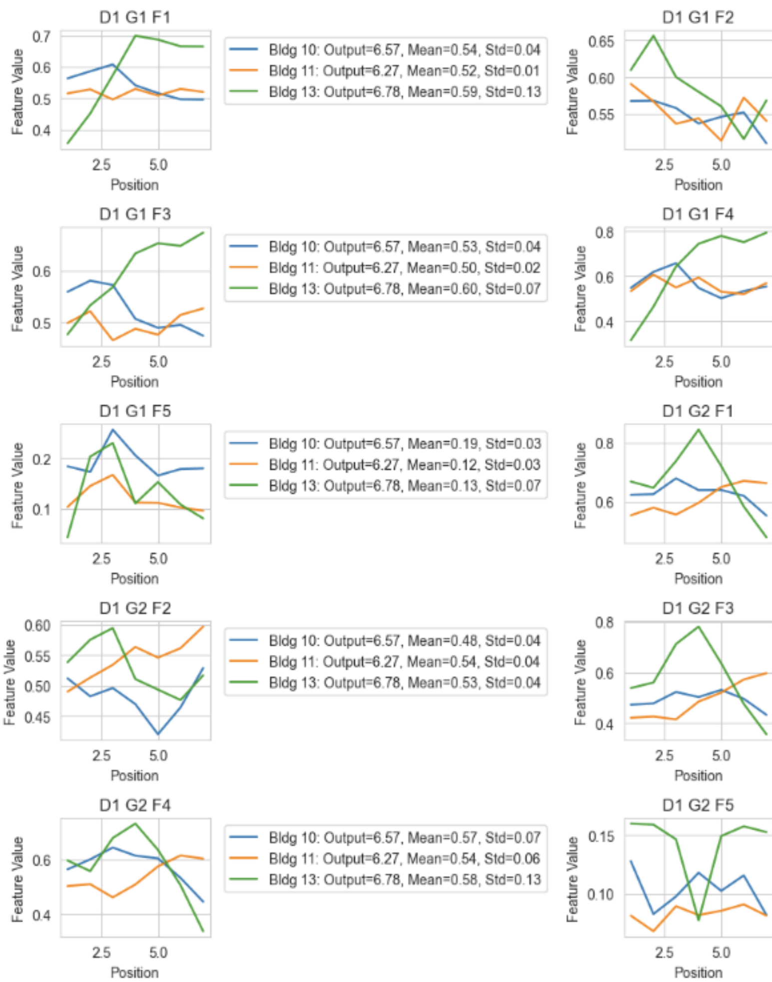


Figure 11: Similar building analysis

The lines look random and looking at line charts alone doesn't provide useful information. On the other hand, looking at the mean of these 3 buildings shows a similar value which indicates that the mean can play a good role in classifying or predicting the output.

3.2. Flattened data Analysis.

3.2.1. Flattened features analysis.

Until now, we didn't try to find relationships between flatten features with a format that includes the position ($D_{1-2}\ G_{1-2}\ F_{1-5}\ P_{1-7}$). The results of finding the correlation between the flattened data and the Output was as following:

Output Measured Value	1.000000
D1 G2 F4 P2	0.583738
D1 G2 F4 P1	0.521016
D1 G2 F1 P2	0.518539
D1 G2 F3 P2	0.472921
D1 G2 F4 P3	0.463253
D1 G2 F3 P3	0.460274
D1 G2 F1 P3	0.451629
D1 G2 F1 P1	0.443421
D1 G2 F3 P1	0.430019
D1 G2 F3 P4	0.404705
D1 G1 F3 P5	0.319539
D1 G2 F1 P4	0.314052
D1 G2 F4 P4	0.304644
D1 G1 F3 P6	0.285469
D1 G1 F1 P6	0.283825
D1 G1 F4 P7	0.258445
D1 G1 F1 P7	0.252741
D1 G1 F4 P6	0.247603
D1 G1 F3 P4	0.237240
D1 G1 F3 P7	0.224868
D1 G1 F5 P7	0.222519
D1 G2 F5 P2	0.215780
D1 G1 F1 P5	0.209780
D1 G1 F4 P5	0.194159
D1 G2 F5 P6	0.188206
D1 G2 F5 P7	0.172877
D1 G2 F5 P5	0.172305
D1 G1 F1 P4	0.165261
D1 G2 F5 P4	0.163796

Figure 12: Flattened feature v.s. Output

The results show the best correlation so far between any feature and the output with multiple values above 0.4. This suggests focusing on handling the data in a flatten way.

3.2.2. Flattened features product analysis.

Since the flattened features showed some improvements in the correlation with the output, it might be beneficial trying to extract new features from the flattened ones by multiplying them with each other. The results of finding the correlation between the multiplied flattened data and the Output was as following:

Output Measured Value	1.000000
D1 G2 F4 P2_x_D1 G2 F4 P2	0.629426
D1 G2 F4 P1_x_D1 G2 F4 P2	0.625030
D1 G2 F4 P1_x_D1 G2 F4 P3	0.620162
D1 G2 F1 P3_x_D1 G2 F4 P1	0.611539
D1 G2 F1 P2_x_D1 G2 F4 P2	0.592872
D1 G2 F3 P3_x_D1 G2 F4 P1	0.590751
D1 G2 F1 P1_x_D1 G2 F4 P2	0.589191
D1 G2 F1 P3_x_D1 G2 F4 P2	0.588999
D1 G2 F4 P2_x_D1 G2 F4 P3	0.585645
D1 G2 F1 P2_x_D1 G2 F4 P1	0.580040
D1 G2 F4 P1_x_D1 G2 F4 P4	0.576605
D1 G2 F3 P3_x_D1 G2 F4 P2	0.576280
D1 G2 F1 P4_x_D1 G2 F4 P2	0.575741
D1 G2 F3 P4_x_D1 G2 F4 P1	0.573443
D1 G2 F3 P4_x_D1 G2 F4 P2	0.571305
D1 G2 F3 P2_x_D1 G2 F4 P2	0.571261
D1 G2 F2 P1_x_D1 G2 F4 P1	0.570076
D1 G2 F3 P1_x_D1 G2 F4 P3	0.569584
D1 G2 F1 P4_x_D1 G2 F4 P1	0.567946
D1 G1 F1 P7_x_D1 G2 F4 P2	0.567084
D1 G2 F1 P1_x_D1 G2 F4 P3	0.566106
D1 G1 F3 P5_x_D1 G2 F4 P1	0.562634
D1 G2 F3 P1_x_D1 G2 F4 P2	0.560405
D1 G2 F3 P2_x_D1 G2 F4 P1	0.556246
D1 G2 F4 P1_x_D1 G2 F4 P1	0.555898
D1 G1 F4 P7_x_D1 G2 F4 P2	0.550646
D1 G1 F1 P6_x_D1 G2 F4 P2	0.549157
D1 G2 F4 P2_x_D1 G2 F4 P4	0.547990
D1 G1 F3 P5_x_D1 G2 F4 P2	0.546959

Figure 13: Multiplication of flattened features

A significant improvement in the correlation were observed with ~30 features with a correlation higher than 0.55. Transforming the flattened data to this form could have potential in fitting a good model.

3.3. Clustering analysis.

Clustering could be useful in group the data to groups with something in common which could reveal some hidden patterns, and which could be useful in features engineering by extracting features based on clusters.

I tried to cluster the data into 2 clusters as shown in the image below:

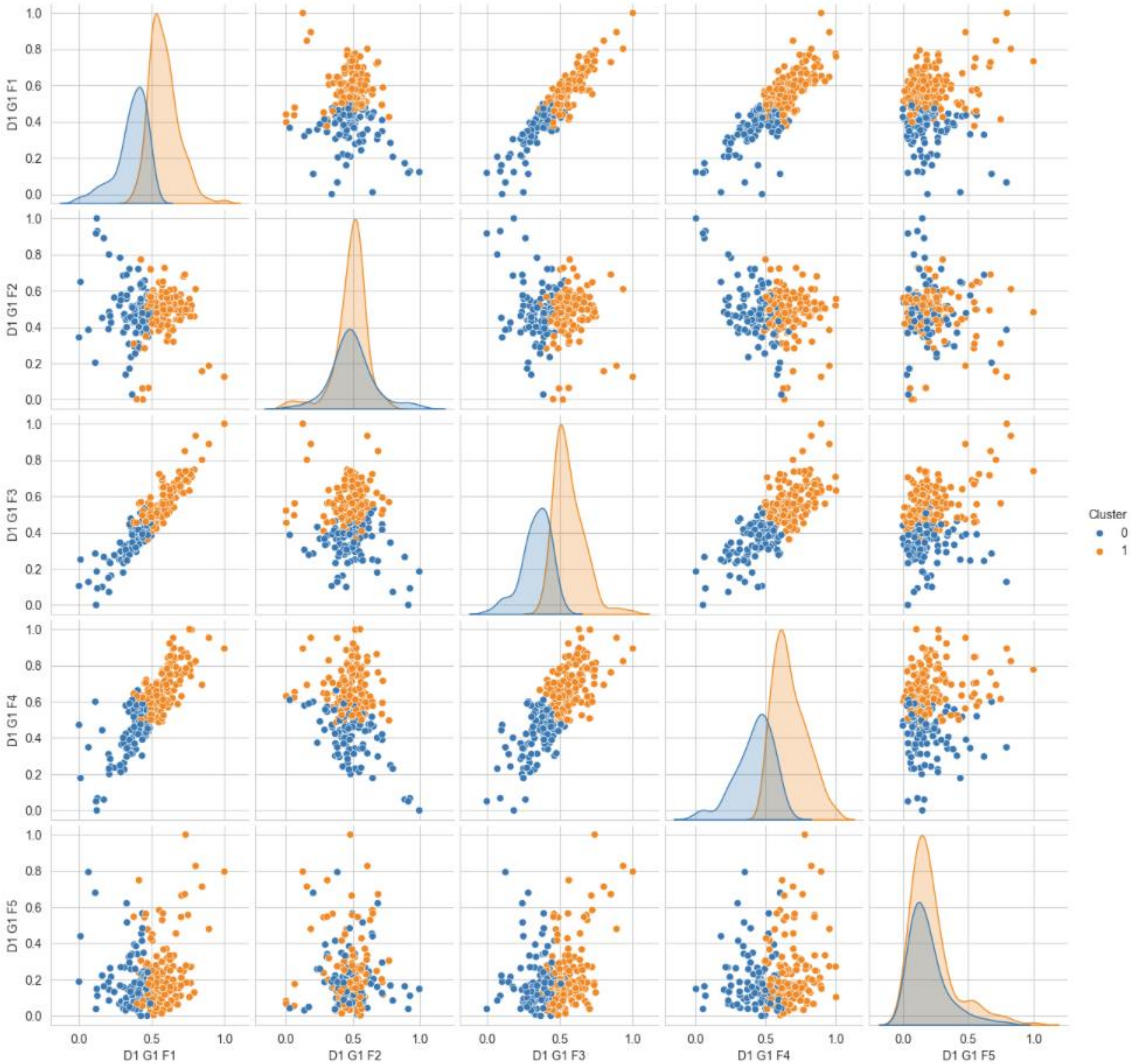


Figure 14: Clustering analysis for D1G1

The results shows that the data KMeans was able to split F1, F3, F4 and F5 into 2 clusters with similar number of elements in each cluster which suggests that we could use features such as cluster index and distance to cluster centroid as feature in the model training.

This also suggests that we should try training the model with and without F2 to see if it has negative impact since it consistently showed some randomness.

4. Model training experiments

4.1. Training Methodology

The data sets provided are very small which represents a set of features for only 36 buildings. The classic training and testing splits techniques could be problematic since a range of 10-30% test data sets only means 2-10 samples which will eventually lead to a not reliable results especially for classification experiments and it won't cover various sample from various strengths. So, instead of that, a cross-validation was used in all experiments with 5 folds to ensure the model generalize well on all subsets of data and to ensure the results are reliable as much as possible.

Since the data analysis showed what looks like a complex non-linear relationship, doing a regression a lone might not work. That's why 2 new columns were introduced to categorize outputs into binary classification and multi-class classification with the categories.

The experiments used 3 different models for experiments:

1- Linear regression model: It was decided to start with a straightforward approach with a basic model that will try to fit the features linearly which give us a starting point for more exploration.
2- Random forest regressor: Random Forest knows to its good ability of generalizing model to avoid overfitting and the randomness in building its subtrees and it's split could be beneficial in finding non-linear relationship. [1]
3- Random forest classifier: For the same reasons as random forest regression, the random forest classifier will be mainly used to classify the data into 2 or 3 categories. Another main reason for choosing random forest was based on a personal preference since I have most experience with this model. [1]

To get the most out of each model, a hyperparameters tunning was conducting using hundreds of parameters options trying to boost the accuracy as much as possible. The small size of the data and the high computational resources available made it viable to explore with as many options as possible.

The evaluation metrics used where RMSE (Root mean square error), R^2 for regression and average accuracy a cross all folds for classification in addition to ROC and AUC scores.

PCA will also be used trying to reduce data dimensionality.

4.2. Experimentation

4.2.1. Linear regression and Random Forest for flattened data and for aggregated data

The first few experiments targeted the aggregated positions data, the flattened data, and the flattened product data.

The experiments targeted both datasets using linear regression, random forest regression and random forest classifier.

Data used	Model type	Best parameters	Accuracy
Aggregated df1	Linear regression	-	RMSE = 9.67 R^2 = -3.86
	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 200}	RMSE = 4.78 R^2 = - 0.18
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100}	64.3%
	RF classifier (3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	50%
Aggregated df2	Linear regression	-	RMSE = 12.26 R^2 = -6.81
	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 50}	RMSE = 4.79 R^2 = - 0.195
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	69.3%
	RF classifier 3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	49.6%
Flattened df1	Linear regression	-	RMSE = 10.47 R^2 = -4.69
	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 50}	RMSE = 4.41 R^2 = - 0.011
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}	74.6%
	RF classifier (3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}	63.9%
Flattened df2	Linear regression	-	RMSE = 14.38 R^2 = - 9.74
	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 50}	RMSE = 4.38 R^2 = 0.002
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}	68.9%
	RF classifier (3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	69.3%
Flattened product df1	Linear regression	-	RMSE = 14.34 R^2 = -9.69
	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 5, 'n_estimators': 200}	RMSE = 4.2 R^2 = 0.08
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	77.5%
	RF classifier (3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	72.1%

Figure 15: Linear regression and Random Forest for flattened data and for aggregated data

The linear regression showed a very bad results with a very high RMSE ranges ~10-14 and negative R-suqared for all data frames used. These RMSE values with the original output ranges between 2-18 suggests a high failure in fitting the data in this linear model.

On the other hand random forest regressor showed a significatly better results acrossed different dataframes with RMSE around 4 and R-squared around zero. Even though, its way better than linear regression, it's still far away from a resonable error rate for such small range of outputs.

Finally the binary and multi-class classifers showed a good start with a highest accuracy of 72.1% for multi-class and 77.5% for binary class using the flattened product dataframe.

The classification showed the worst results for the the aggregated positions data compated to the flattened positions which also suggests to focus the upcoming expirments on the flattened data.

Both data sets showed similar results which make it reasonable to focus on one dataset for the upcoming expirments then after obtaining the desired results, the same approach can be applied to the other data set to reach conclusions.

4.2.2. Flattened data with outliers-related features.

For this experiment the flattened dataset 1 were used and an extra 4 features were added:

1- Number of upper-bound outliers for each building.
2- Number of lower-bound outliers for each building.
3- Whether the building has upper-bound outliers.
4- Whether the building has lower-bound outliers.

Data used	Model type	Best parameters	Accuracy
Flattened df1 with extra outliers related features	RF regressor	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 200}	RMSE = 4.38 R^2 = 0
	RF binary classifier	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100}	71.8%
	RF classifier (3 categories)	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	69.3%

The results didn't show any improvements compared to the previous experiments, which might indicate that the measured outliers are not actual outliers, and the main reason they were identified as outliers is the small size of the data.

4.2.3. Trying all combinations of positions.

In this expirment, the data will be flattened and all the combinations of positions will be explored. This expirment aims to find which combination of positions could lead to better relationships between the features and the output.

The combinations includes: (P1)(P2)...(P7)(P1 P2)...(P6 P7) .... (P1 P2 P3 P4 P5 P6 P7). The number of combinations is 127 ($2^7 - 1$).

```
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 80.4%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 75.0%
#### Combination: ('P1', 'P2', 'P4', 'P5', 'P6') #####
```

Figure 16: All positions combinations expirment

The best results for the 127 experiments were for positions ('P1', 'P2', 'P4', 'P5', 'P6') with multi-class accuracy of 75% and binary classification accuracy of 80.4% which are the best results obtained so far. This highlight the importance of the correlation between diffirent positions.

4.2.4. Trying clustering realted features

In this expirment, the flattened data will be clustered using different number of clusters, for each setting, the cluster index will be used as a feature, in addition to features distance from each cluster centroid.

The experiment ran on different number of clusters ranged from 2 to 12.

```
#### Num of clusters = 3 #####
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 71.8%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 66.8%
#### Num of clusters = 3 #####
```

Figure 17: Clustering experiment

The best obtained result was using 3 clusters with accuracy of 72% for binary classification and an accuracy of 67% for 3 categories classification.

The clustering didn't improve the results, but it could be beneficial if merged with other feature engineering techniques.

4.2.5. Trying PCA to reduce dimensionality.

The flattened data has a relatively large number of features compared to the number of samples. In this experiment, the dimensionality of flattened data was reduced using PCA with different number of PCA components (6, 12, 18, 24, 30, and 36). 36 is largest number of components that could be used because of the number of samples.

```
#### Num of PCA components = 36 #####
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 69.6%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 68.9%
#### Num of PCA components = 36 #####
```

Figure 18: PCA experiment

The best results came from 36 components which the largest possible options to use. The results had lower accuracy compared to the flattened data results on its own. This indicates that PCA won't be very helpful in this kind of data set because of the low number of samples which limits the number of components that could be used.

4.2.6. Splitting the output using weighted sum for features.

In this experiment, instead of flattening the data and getting only 36 samples, the output will be splitted between the 7 positions for each building using a weighted sum of features. Meaning that the position with the highest sum of features values will get the highest proportion of the output.

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
RMSE Linear regression: 4.0165495053469185
R2 Score Linear regression: 0.1614382655143295
RMSE Random Forest regressor: 4.514503023326877
R2 Score Random forest regressor: -0.05937248473776613
```

Figure 19: Splitting the output using weighted sum.

This experiment gave the best regression results so far with an RMSE = 4 and R-Squared – 0.16. But still the results are not great considering the range of output of values.

4.2.6. Using 14 positions with weight correlation with output.

In this experiment, the group 2 features will be transformed to form another 7 positions which will result in 14 positions per building, in addition to that, each feature in each position correlation with the output will decide how the output will be distributed over the 14 positions.

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
RMSE Linear regression: 4.336656265477407
R2 Score Linear regression: 0.022450421555503475
RMSE Random Forest regressor: 4.591492713931705
R2 Score Random forest regressor: 0.13881352180703888
```

Figure 20: Using 14 positions with weight correlation with output.

The results showed similar values compared to the previous experiment with a slight advantage to the previous one.

The last 2 experiments showed the best linear regression results so far.

4.2.7. Using deep learning

Another approach used was creating a multi-layer neural network with many hidden layers that could potentially capture non-linear patterns in the data. For this purpose, 6 hidden layers were used with 16 neurons in each with Relu as an activation function.

This experiment tried both regression, binary, and multi-class classification using the same model architecture with evaluation based on 5 number of folds and averaging the results.

- Avg RMSE: 5.195316390218844
- Avg R2 Score: -0.6948699653837845
- Avg Binary Classification Accuracy: 64%
- Avg Multi-Class Classification Accuracy: 50.3%

The results didn't show any improvments which suggests that the data were overfit on the training phase. And there are no poteniatl solution for this with this small number of data which makes deep learning a bad expirmentation choice.

4.2.8. Using rolling window

One of the great features provided by pandas is the rolling features, where you can take a slice of data (In this case, a slice of positions) and generate some aggregations on that window such as summation, mean, min, max … etc. [2]

In this experiment, 5 windows size were tested (2 to 6) with rolling on sum, mean, median, variance, and skewness.

```
######################### Number of items in window 2 #########################
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 74.6%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 69.6%
######################### Number of items in window 2 #########################


######################### Number of items in window 3 #########################
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 74.6%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 72.5%
######################### Number of items in window 3 #########################


######################### Number of items in window 4 #########################
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 77.5%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 75.4%
######################### Number of items in window 4 #########################


######################### Number of items in window 5 #########################
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 80.4%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 75.4%
######################### Number of items in window 5 #########################


######################### Number of items in window 6 #########################
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Cross-Validated RandomForestClassifierModel Accuracy for 2 classes: 77.5%
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Cross-Validated RandomForestClassifierModel Accuracy for 3 classes: 75.4%
######################### Number of items in window 6 #########################
```

Figure 21: Rolling windows.

This approach gave a good result on number of windows = 5, with an accuracy of 77.5% for multi-class and 80.4% for binary classification. Merging this approach with other approaches could enhance the overall results.

4.2.9. Hybrid approach between clustering and rolling windows.

For the final experiment, a different combination of previous experiments was merged aiming to getting the best of each feature engineering techniques. The final experiment was a combination of clustering approach and rolling window.

The flattened data 1, flattened data 2, flattened data 1 group 1, flattened data 1 group 2, flattened data 2 group 1, flattened data 2 group 2 where each tested using multiple configurations including 8 number of clusters options, 6 window sizes, and 6 different PCA options with a seventh option without PCA, which give us 334 experiments for each data frame.

All data frames were tested using binary classification and 3-classes classification and the accuracy was calculated using cross-validation with 5 number of folds to ensure the reliability of the test results.

```python
def buildModelDataFrame(df, n_clusters, DIndex, GIndex, window_size, n_components):
    df_temp = df.copy()

    df_adj_avg_cluster = df_temp[['ID', 'Output Measured Value', 'Strength_2', 'Strength_3']].copy()
    feature_columns = [col for col in df_temp.columns if 'F' in col] if GIndex == 0 else [col for col in df_temp.columns if f'G{1 if GIndex == 2 else 2}' not in col and 'F' in col]

    # Apply K-Means clustering on the selected features
    kmeans = KMeans(n_clusters, random_state=42)
    df_temp['Cluster'] = kmeans.fit_predict(df_temp[feature_columns])
    df_adj_avg_cluster['Cluster'] = df_temp['Cluster']

    # Calculate distances to each centroid
    centroids = kmeans.cluster_centers_
    for i in range(n_clusters):
        df_adj_avg_cluster[f'Distance_to_Centroid_{i}'] = np.linalg.norm(df_temp[feature_columns] - centroids[i], axis=1)


    for k in range(1, 6):
        for g in range(1, 3 if GIndex == 0 else 2):
            feature_cols = [f'D{DIndex} G{g if GIndex == 0 else GIndex} F{k} P{p}' for p in range(1, 8)]
            feature_df = df_temp[feature_cols]

            # Calculate various rolling statistics
            rolling_mean = feature_df.rolling(window=window_size, min_periods=1, axis=1).mean()
            rolling_sum = feature_df.rolling(window=window_size, min_periods=1, axis=1).sum()
            rolling_median = feature_df.rolling(window=window_size, min_periods=1, axis=1).median()
            rolling_var = feature_df.rolling(window=window_size, min_periods=1, axis=1).var()
            rolling_skew = feature_df.rolling(window=window_size, min_periods=1, axis=1).skew()

            # Adding the new columns to the DataFrame
            stats_suffixes = ['mean', 'sum','median', 'var', 'skew']
            for stat, suffix in zip([rolling_mean, rolling_sum,rolling_median,rolling_var,rolling_skew], stats_suffixes):
                new_col_names = [f'{col}_rolling_{suffix}' for col in feature_cols]
                df_adj_avg_cluster[new_col_names] = stat

    if n_components != None and n_components < 42: # Max is 36 - last try is without PCA
        excluded_cols = ['ID', 'Output Measured Value', 'Strength_2', 'Strength_3']
        df_excluded = df_adj_avg_cluster[excluded_cols]
        df_for_pca = df_adj_avg_cluster.drop(columns=excluded_cols)
        pca = PCA(n_components)  # for example, using 5 components
        df_pca_transformed = pd.DataFrame(pca.fit_transform(df_for_pca.fillna(0)))
        df_adj_avg_cluster = pd.concat([df_excluded, df_pca_transformed], axis=1)


    return df_adj_avg_cluster
```

Figure 22: Code snippet from the final experiment

This experiment gave the best results with a cross-validation accuracy of 89% for binary classification and a cross-validation accuracy of 81% for multi-class classification.

5. Results analysis

The last experiment gave the best accuracy for the flattened dataset 1 and the second-best accuracy was from the second group in the second dataset.

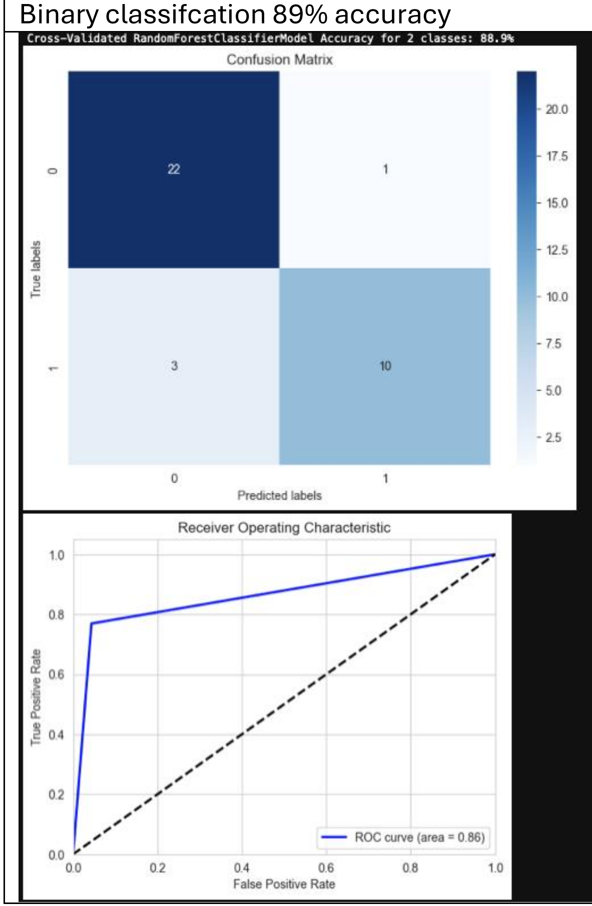
The best results for all used data frames can be summarized as the following:

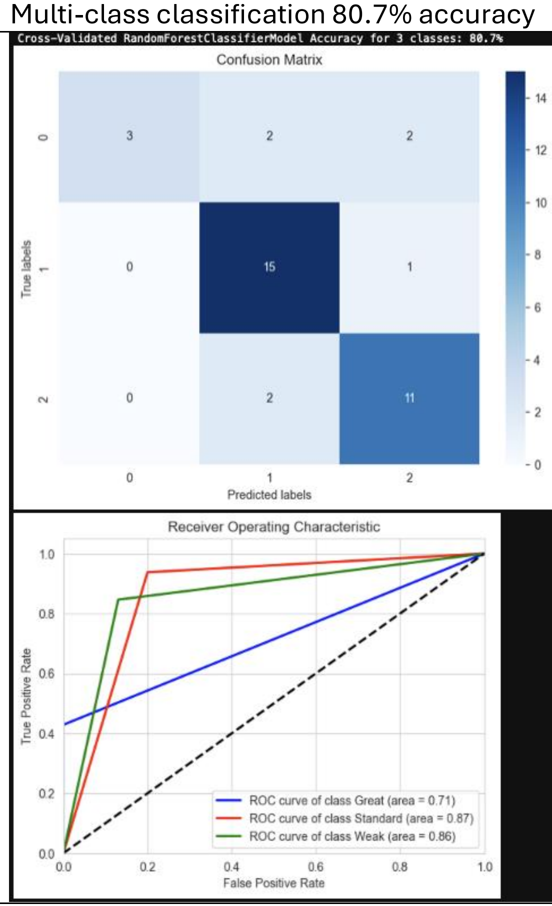
Data set	Classification	Configuration (PCA wasn't use in any)	Accuracy
Flattened_df1	Binary	Number of clusters = 6, Number of window items = 4	89%
	Three categories	Number of clusters = 5, Number of window items = 4	80.7%
Flattened_df1g1	Binary	Number of clusters = 3, Number of window items = 4	69%
	Three categories	Number of clusters = 8, Number of window items = 4	64%
Flattened_df1g2	Binary	Number of clusters = 4, Number of window item = 7	86.1%
	Three categories	Number of clusters = 4, Number of window items = 3	78%
Flattened_df2	Binary	Number of clusters = 3, Number of window items = 6	86.1%
	Three categories	Number of clusters = 4, Number of window items = 3	75%
Flattened_df2g1	Binary	Number of clusters = 6, Number of window items = 4	66.4%
	Three categories	Number of clusters = 5, Number of window items = 2	53%
Flattened_df2g2	Binary	Number of clusters = 11, Number of window items = 5	89%
	Three categories	Number of clusters = 6, Number of window items = 3	78%

Figure 23: Hybrid approach results

5.1. Flattened data set 1 analysis

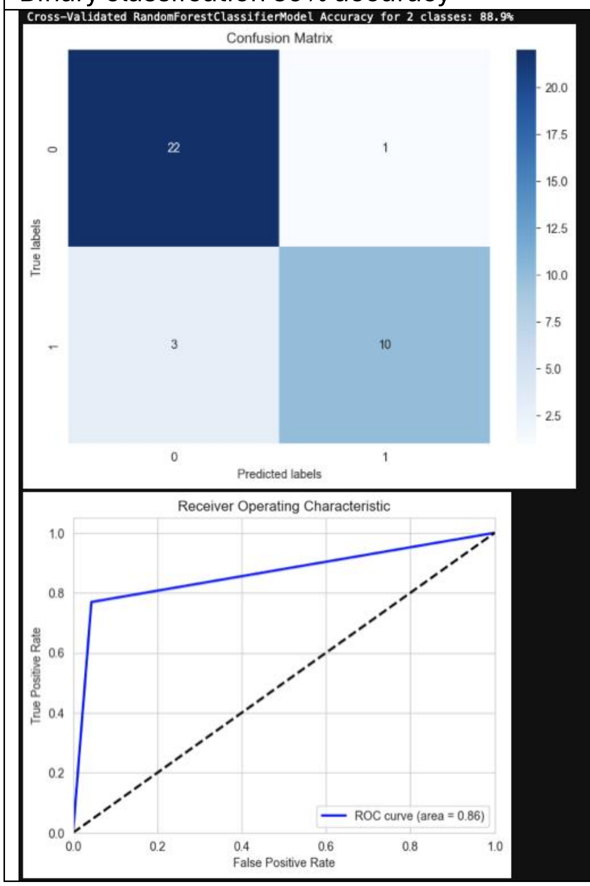
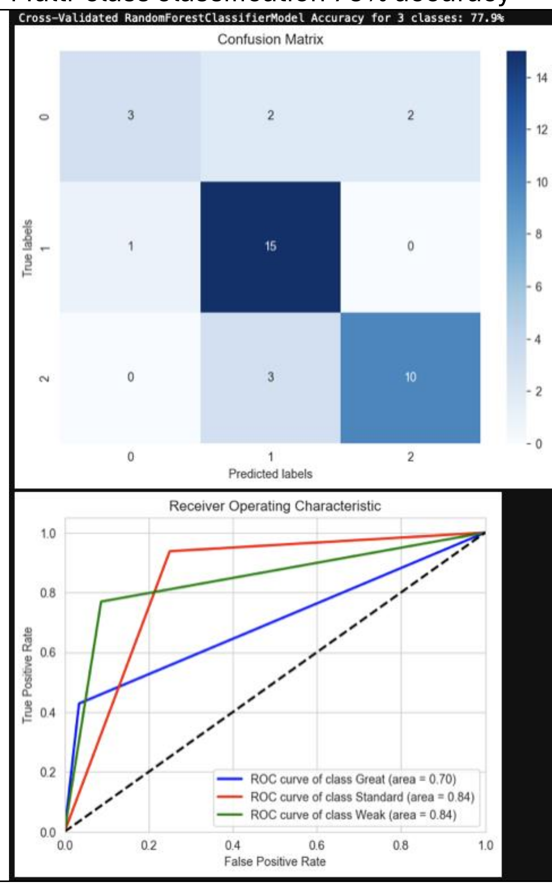
Binary classifcation 89% accuracy	Multi-class classification 80.7% accuracy





1- For binary classifaction: The results shows an 89% overall accuracy with 1 out of 23 as false positives and 3 out of 13 as false negatives.
2- For multi-class: The resutls shows an 80.7% overall accuracy with 2 out 13 was not predicted correctly for “weak class” which give us a better result compared to the binary classifciation if the main goal is to always detect the weak buildings.
3- For the multi class the “great” category gave the worst results with only 3 out of 7 were correctly identified.

5.2. Flattened data set 2 group 2 analysis

Binary classifcation 89% accuracy	Multi-class classification 78% accuracy
	

1- For binary classifaction: The results are identical to the flattened data set 1
2- For multi-class: The resutls shows an 78% overall accuracy with 3 out 13 was not predicted correctly for “weak class” which give us the same result compared to binary which suggests binary classifier if we decided to use this data frame is better.

5.3. Recommendations

If identifying the weak building is top priortiy, then using all the values in dataset 1 in it's flattened form is recommended since it resulted in the best accuracy to identify weak buildings.

If the overall accuracy is the most important, then using the binary classifier in the dataset 2 group 2 is the best option since it will give the highest accuracy and it will get rid off the need of taking mesaure from one side of the building.

There is a room for deeper analysis and model improvements if we gain more information about the data from the data owner and a domain expert. The conducted analysis was mostly blind with multiple expirements hoping for a good fitting for data.

6. Conclusion

In the report, we analyzed spatial data using various ways trying to find patterns and relationships between the different features, this included using different kind of plots like histograms, scatter plots, box plots and correlation matrix heatmap. We also restructered the original data using multiple ways like flattening it, aggregating it and taking different subsets.

Multiple expirements has been conducted based on different feature engineering techinques to find the hidden relationships between the features and the positions. Each expirment conducted on different models, different parameters, and different forms of data. We explored some advanced techinques like rolling window which helped in improving the overall model performance. We ended up with a hybrid approach that combine two techniques together which are clustering and rolling window aggregations to build the best model.

The analysis and expirments showed a poor performance of regression which lead us to focus on classification which resutled in a good accuracy of 89% usign binary classification and 80.7% using three categories classification. Building the model using the first dataset in it's flatten form gave the best results, and we got a similar results using only features from the second side of the building of the second data set. The multiclass classfication gave the best results for separting the weak category from other with an accuracy of 85.4% (One vs All).

It's crucial to get more information about the data for a future improvments to achieve a more targeted analysis that could eventually lead into a strong fitting of data.

7. References

[1] G. L. Team, "Random forest Algorithm in Machine learning: An Overview," 13 6 2023. [Online]. Available: https://www.mygreatlearning.com/blog/random-forest-algorithm.

[2] "Introduction to Pandas rolling," Educba, 7 6 2023. [Online]. Available: https://www.educba.com/pandas-rolling/.