AI/ML for 5G-Energy Consumption Modelling by ITU AI/ML in 5G Challenge

Hamdi Razi

Engineering Student at Ecole Polytechnique De Tunisie
Carthage University

PLAN

- Introduction
- Data Preprocessing
- Feature Engineering
- Results
- Conclusion

Introduction

- In the real world, there's no specific equation to measure the energy consumption of a 5G base station based on various configurations.

- Configurations include frequency, bandwidth, transmission power, power-saving modes, and load.

- Our goal: Measure energy consumption and find the best configuration combinations for energy reduction.

Data Preprocessing

The data provided is divided into four datasets. The training and test data are derived by merging CLdata and BSinfo based on the base station name and cell name.

	BS	CellName	RUType	Mode	Frequency	Bandwidth	Antennas	TXpower
0	B_0	Cell0	Type1	Mode2	365.0	20	4	6.875934
1	B_1	Cell0	Type2	Mode2	532.0	20	4	6.875934
2	B_2	Cell0	Type1	Mode2	365.0	20	4	6.875934
3	B_3	Cell0	Type2	Mode2	532.0	20	4	6.875934
4	B_4	Cell0	Type2	Mode2	532.0	20	4	6.875934

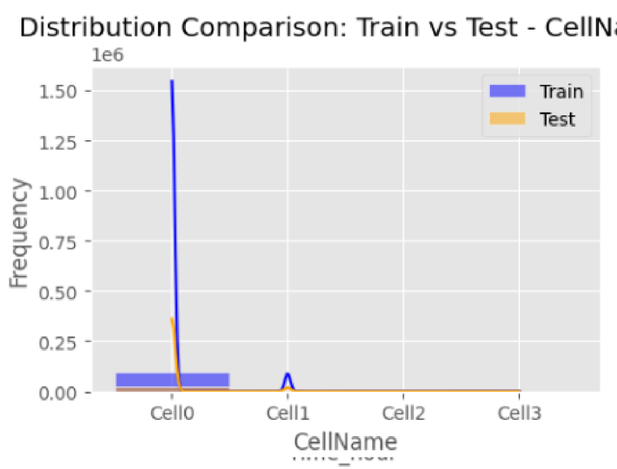
Base station information

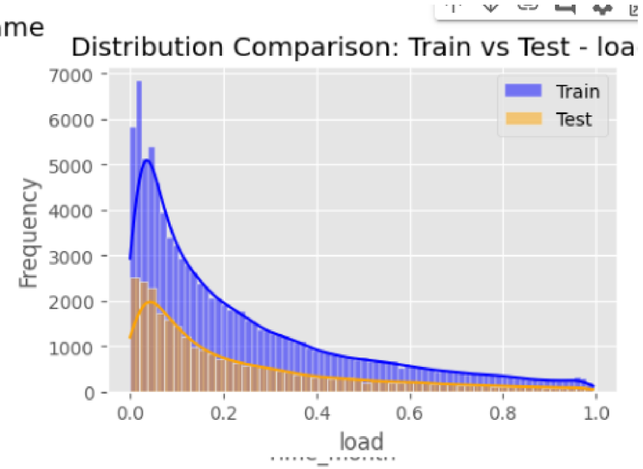
	Time	BS	CellName	load	ESMode1	ESMode2	ESMode3	ESMode5	ESMode6	ID
0	1/1/2023 1:00	B_0	Cell0	0.487936	0.0	0.0	0.0	0.0	0.0	1/1/2023 1:00_B_0
1	1/1/2023 2:00	B_0	Cell0	0.344468	0.0	0.0	0.0	0.0	0.0	1/1/2023 2:00_B_0
2	1/1/2023 3:00	B_0	Cell0	0.193766	0.0	0.0	0.0	0.0	0.0	1/1/2023 3:00_B_0
3	1/1/2023 4:00	B_0	Cell0	0.222383	0.0	0.0	0.0	0.0	0.0	1/1/2023 4:00_B_0
4	1/1/2023 5:00	B_0	Cell0	0.175436	0.0	0.0	0.0	0.0	0.0	1/1/2023 5:00_B_0

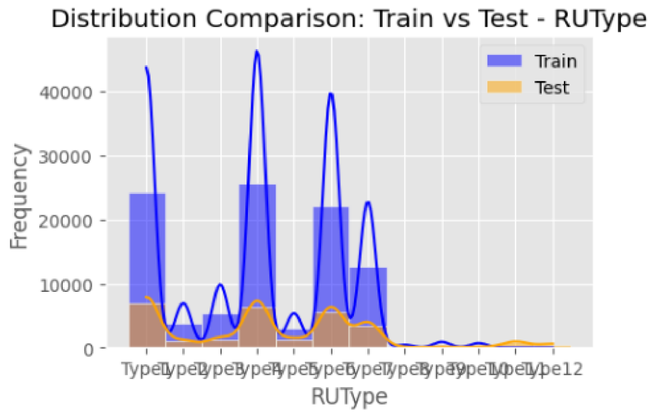
Cell information

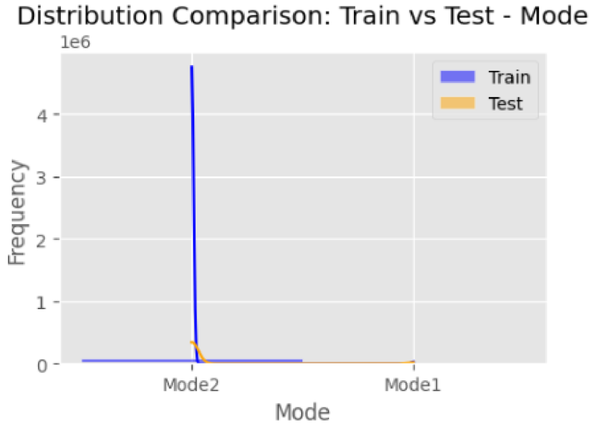
There are no NaN values, so there is no need for cleaning.

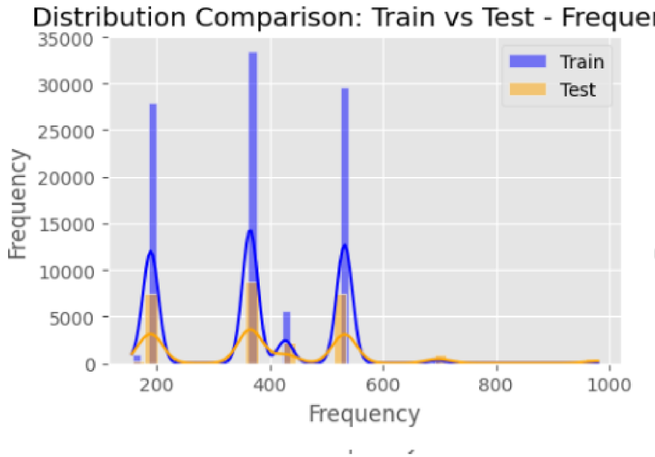
The main challenge of this competition is the large imbalance in the provided dataset,This imbalance is evident in the distribution of the training and test sets.

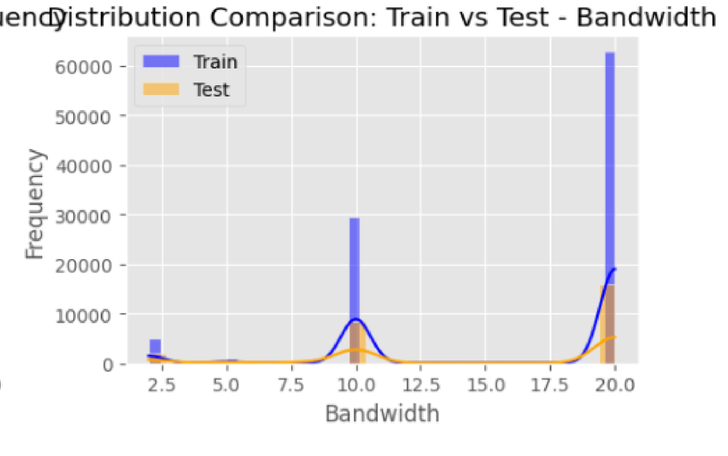


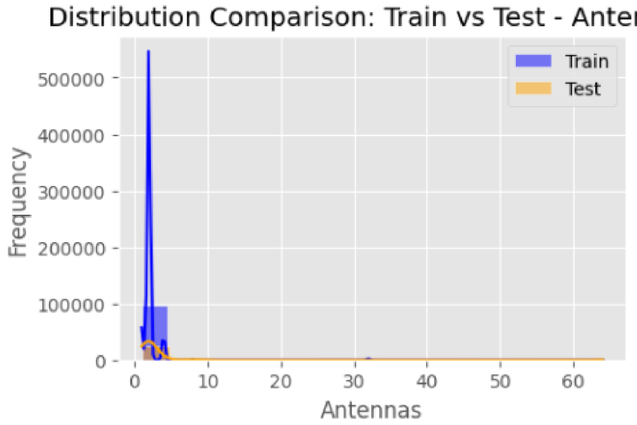


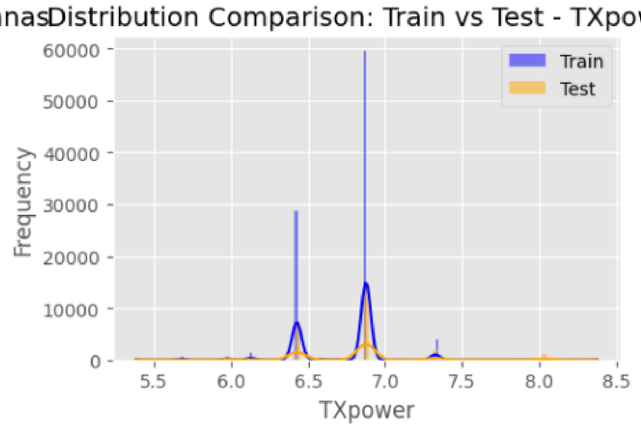








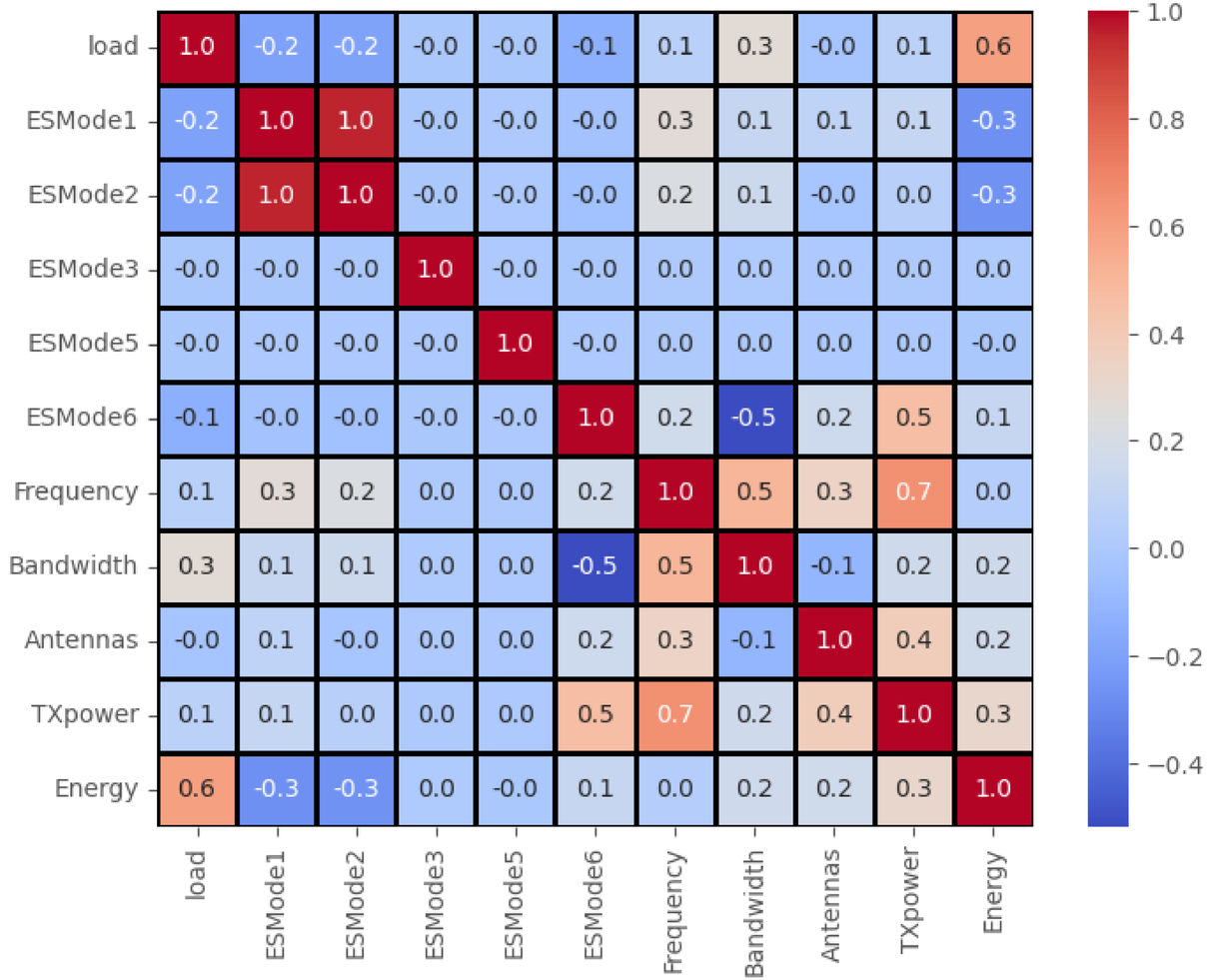




Most of the features in our data are noisy and unbalanced. Therefore, there will be a need to create some valuable features to improve accuracy and generalization.

Feature Engineering

From this correlation heatmap, we can see that 'load' is the most correlated feature to energy. However, the main problem is that most features are noisy, as I mentioned in the previous section. Therefore, we need to generate new features from the existing ones to extract valuable patterns that will help us accurately predict energy consumption.



Feature combination

```
data['lo_tx']=data['TXpower']/data['Antennas']
data['lo_fr']=data['load']*data['Frequency']
data['combination']=data['Frequency']/data['Frequency'].max()+data['TXpower']/data['TXpower'].max()
-data['Antennas']/data['Antennas'].max()
```

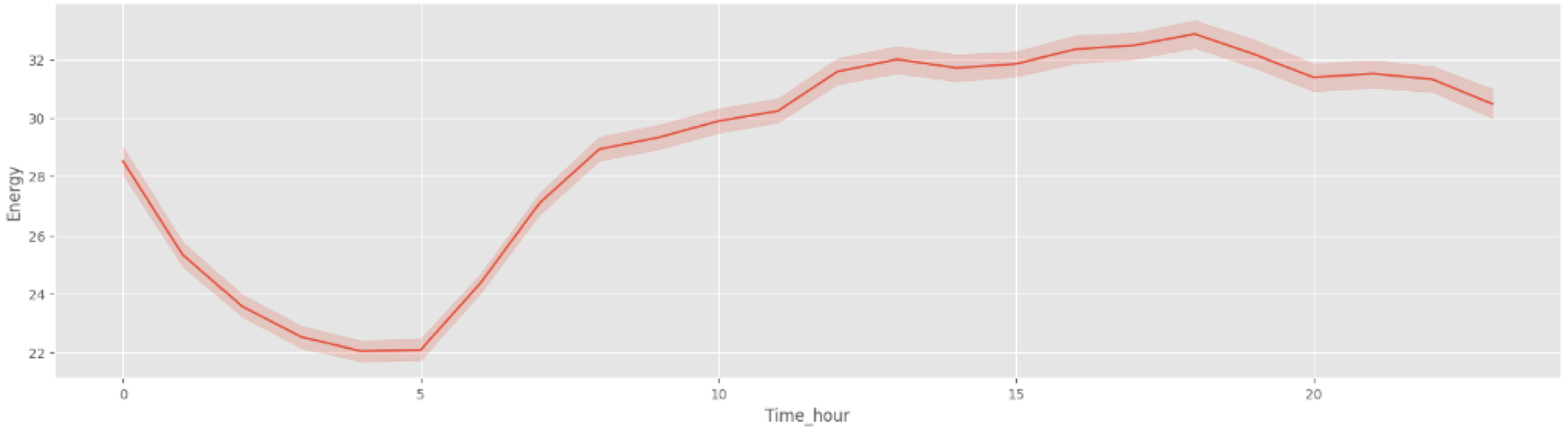
The generated features are valuable and less noisy, especially the "load * frequency" features, which are the most important features later.

Temporal features

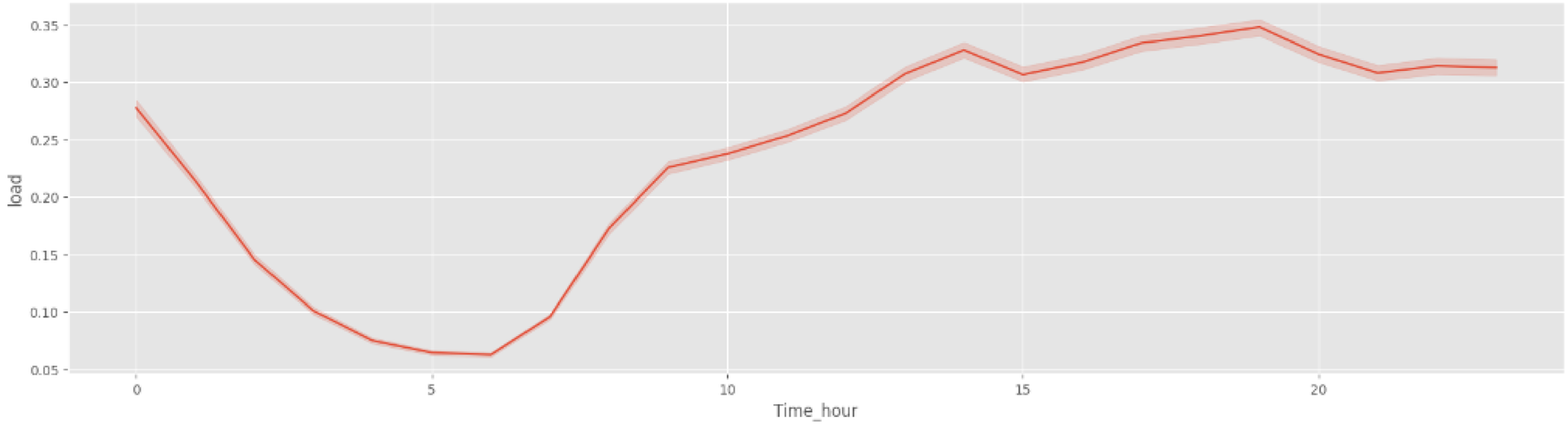
Base station energy consumption differs from hour to hour. For example, consumption during the early hours of the day is lower than during the midday hours. Additionally, consumption during the middle of the week varies from consumption during the weekend. To account for these differences, I create temporal features, such as the hour of the day and the day of the week, and similarly for the month.

These kinds of features are not related to any future leakage; they are just like encoding the hours of the day and the day of the week.

These features are one of the key concepts for predicting energy.

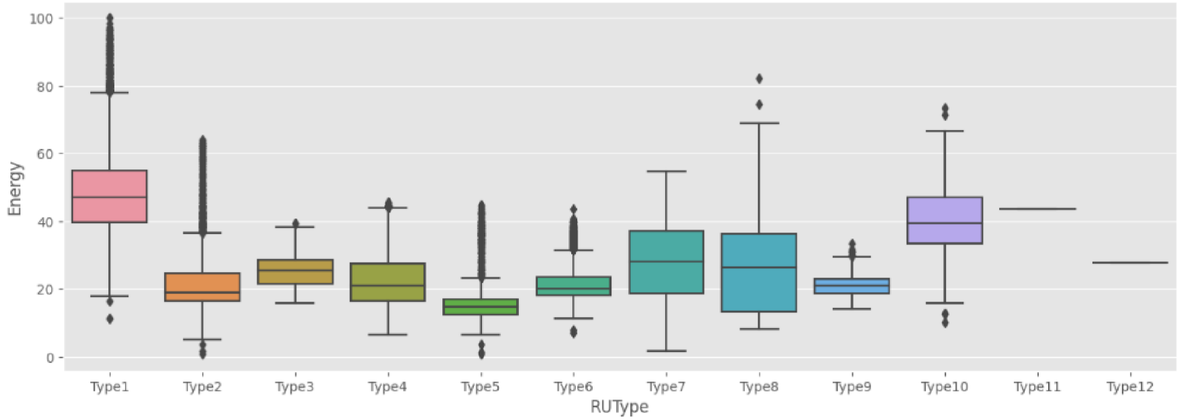


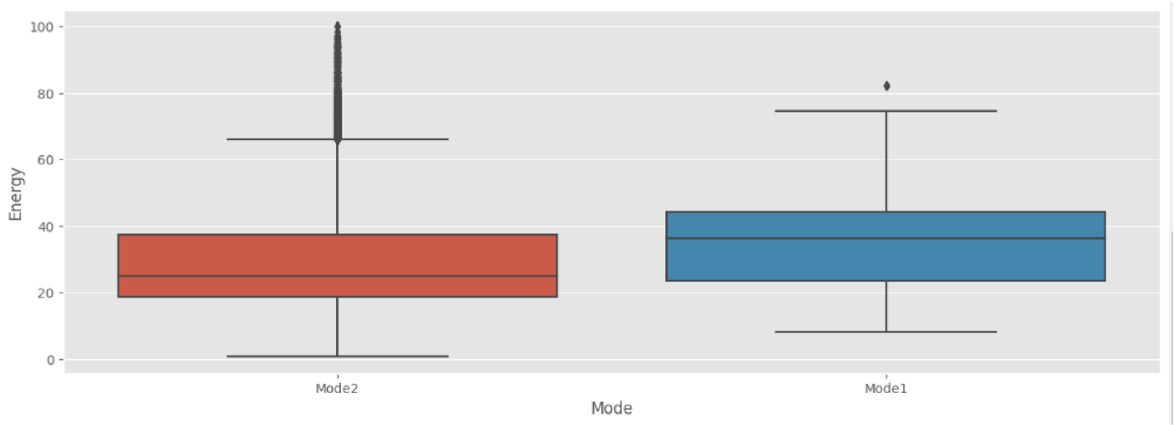
As we can see here the energy values get higher after the first 10 hours of the day also for the load of cells.



Categorical Encoding

Given that the modes of transmission and the radio unit types of the base stations have a significant impact on energy consumption, I performed a simple label encoding for the Mode feature and used one-hot encoding for the RUType feature.





Statistical features

This method is well-suited to our challenge, which involves the imbalance in the dataset. By utilizing aggregation and quantiles (calculating the 10th, 25th, 75th, and 90th percentiles), it helps extract meaningful information from our raw data while also reducing the imbalance.

The main process involves taking categorical features, namely BS, RUType, Time_hour, and Time_day. For each of these features, we group the data by each class within that specific feature and generate aggregation and quantile features with respect to other numerical features.

```python
def Agg(df,cols1,cols2) :
        for col1 in cols1 :
            for col2 in cols2 :
                df[f"{col1}_{col2}_mean"] = df.groupby(col1)[col2].transform('mean')
                df[f"{col1}_{col2}_std"] = df.groupby(col1)[col2].transform('std')
                df[f"{col1}_{col2}_max"] = df.groupby(col1)[col2].transform('max')
                df[f"{col1}_{col2}_min"] = df.groupby(col1)[col2].transform('min')
                df[f"{col1}_{col2}_median"] = df.groupby(col1)[col2].transform('median')
        return df
def AddQuantiles(df,cols1,cols2) :
        for col1 in cols1 :
            for col2 in cols2 :
                q75 = dict(df.groupby(col1)[col2].quantile(0.75))
                q25 = dict(df.groupby(col1)[col2].quantile(0.25))
                q90 = dict(df.groupby(col1)[col2].quantile(0.90))
                q10 = dict(df.groupby(col1)[col2].quantile(0.10))
                df[f"{col1}_{col2}_q75"] = df[col1].map(q75)
                df[f"{col1}_{col2}_q25"] = df[col1].map(q25)
                df[f"{col1}_{col2}_q90"] = df[col1].map(q90)
                df[f"{col1}_{col2}_q10"] = df[col1].map(q10)
        return df
```

```python
data=Agg(data,['Time_hour'],['load','Frequency','Bandwidth','TXpower','lo_fr','lo_tx'])
data=Agg(data,['Time_day'],['load'])
data=Agg(data,['BS'],['load','Frequency','Antennas','Bandwidth','TXpower','lo_fr','lo_tx'])
data=Agg(data,['RUType'],['load','Frequency','Antennas','Bandwidth','TXpower','lo_fr','lo_tx'])
```

```python
data=AddQuantiles(data,['Time_hour'],['Frequency','Bandwidth','TXpower','lo_fr','lo_tx'])
data=AddQuantiles(data,['BS'],['load','Frequency','Antennas','Bandwidth','TXpower','lo_fr','lo_tx'])
data=AddQuantiles(data,['RUType'],['load','Frequency','Antennas','Bandwidth','TXpower','lo_fr','lo_tx'])
data=AddQuantiles(data,['Time_day'],['load'])
```

As you can see here, these are the functions for generating the aggregation features and the quantile features. And, as you can see, we calculate the aggregation and quantiles for load, Frequency, Bandwidth, TXpower, lo_fr, and lo_tx for each categorical feature, except for Time_day, where we calculate only for the load.

I made this kind of features with respect to all dataset , to clarify the use of Time dependent features, in my approach i did not select any specific features values and make it as feature , but groupedby categorical features and for each class this those ones i have selection a mesuarement for example for load or TXpower for BS_0 , this kind statistics and as you now in statistics large number of samples leads to better estimation for example for the mean or standard deviation . same thing for quantiles .

The total number of input features for the models is 207
These are some of the overall features to use.

```
'ESMode1', 'ESMode2', 'ESMode3', 'ESMode5', 'ESMode6', 'Time_day',
'Time_hour', 'Time_month', 'Mode', 'Antennas', 'TXpower', 'Energy',
'lo_tx', 'lo_fr', 'combination', 'Time_hour_load_mean',
'Time_hour_load_std', 'Time_hour_load_max', 'Time_hour_load_min',
'Time_hour_load_median', 'Time_hour_Frequency_mean',
'Time_hour_Frequency_std', 'Time_hour_Frequency_median',
'Time_hour_Bandwidth_mean', 'Time_hour_Bandwidth_std',
'Time_hour_Bandwidth_max', 'Time_hour_Bandwidth_min',
'Time_hour_Bandwidth_median', 'Time_hour_TXpower_mean',
'Time_hour_TXpower_std', 'Time_hour_TXpower_max',
'Time_hour_TXpower_median', 'Time_hour_lo_fr_mean',
'Time_hour_lo_fr_std', 'Time_hour_lo_fr_max', 'Time_hour_lo_fr_min',
'Time_hour_lo_fr_median', 'Time_hour_lo_tx_mean', 'Time_hour_lo_tx_std',
'Time_hour_lo_tx_max', 'Time_hour_lo_tx_min', 'Time_hour_lo_tx_median',
'Time_day_load_mean', 'Time_day_load_std', 'Time_day_load_max',
'Time_day_load_min', 'Time_day_load_median', 'BS_load_mean',
'BS_load_std', 'BS_load_max'],
```

Results

The best model that gives the WMAPE score is one lightgbm model with this set of hyperparameters :

```
# best set of parameters
best_params={'learning_rate': 0.050149327111138855,
 'num_leaves': 82,
 'reg_alpha': 2.5639389256690515,
 'reg_lambda': 0.12523931338344882,
 'n_estimators': 3760}
```

I did not use too many parameters that could make the model too complex and lead to overfitting. Instead, I simply used reg_alpha and reg_lambda to prevent overfitting.

This Model gives me the private score : 0.070974957

Conclusion

The use of statistical methods for generating features, in addition to time-extracted features and combined features, results in a total of 207 features. These features enhance the performance of the LightGBM model with its optimal set of parameters, yielding a score of 0.070974957. This achievement is obtained with a single LightGBM model, and I believe that using a custom k-fold cross-validation approach with good data splitting will likely yield even more accurate results.

Thank you for your attention