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

Razi Hamdi

Ecole Polytechnique De Tunisie, University of Carthage , Tunisia

razihamdi4@gmail.com,

Abstract -

5G, offers many new services and benefits, but it's using more energy than the previous 4G. This is because 5G needs more cell towers(Base station's), processes more data, and has extra antennas.

The amount of energy a base station uses depends on many things like how it's built, its settings, how busy it is, and whether it's using energy-saving features. Machine learning can be used to predict the energy consumption of a specific base station based on its configurations.The challenge of predicting energy consumption is the huge imbalance in the provided data, so we cannot use that raw data. In this competition, I used different machine learning techniques, such as various feature engineering methods and feature selection. Additionally, I used a LightGBM model with tuned parameters to achieve good generalization, resulting in a score of 0.070974957 WMAPE.

Keywords -

Energy consumption; Machine learning; Generalization

1 Introduction

In the real world, there is no specific equation for measuring the energy consumption of a 5G base station with respect to its configurations, such as frequency, bandwidth, transmission power, power-saving modes, and load.

In addition to mesuring the energy consumption of a base station, we are also seeking to find the best combination of configurations to reduce this consumption.
So Machine learning approaches are the best solution for generalizing predictions of energy consumption, but there is a significant challenge, which is the large imbalance in the provided dataset.

2 Data Preprocessing

2.1 Data merging

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. There are no NaN values, so there is no need for cleaning. But, as mentioned in the introduction, there is a significant imbalance in the dataset. This imbalance is evident in the distribution of the training and test sets in the notebook. 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.

2.2 Feature engineering

Machine learning models cannot capture certain patterns without human help, and feature engineering is an essential step in a machine learning project.
Feature engineering involves creating new features from existing ones ,It provides a better generalization for machine learning models and offers an effective way to improve scores.
In this project, I utilized multiple feature engineering methods to capture valuable patterns for energy prediction.

2.2.1 Feature combination

In this project[1], I discovered a strong relationship between the load and the frequency of cells. Both of these factors are noisy features when considered separately. Therefore, I created a new feature that represents the product of these two variables, which turned out to be the most important feature in my approach. Additionally, I generated another feature, which is the division of transmission power over the number of antennas. I also created count features for cell names and radio unit types for each base station. These last features helped extract valuable patterns in the energy consumption value intervals.

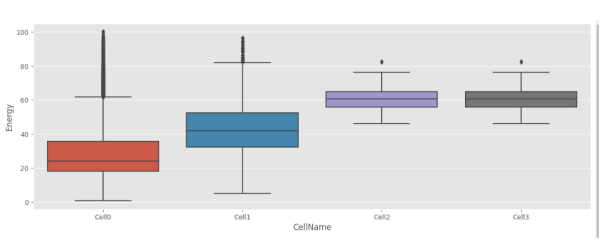

Figure 1. Box-plot of Energy with respect to CellNames

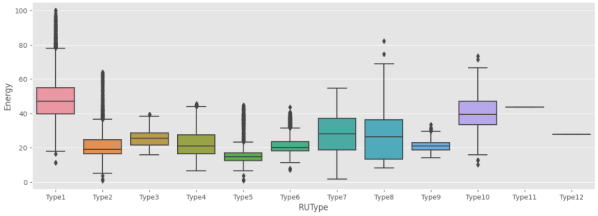

Figure 2. Box-plot of Energy with respect to RU-Types

2.2.2 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.

	Time	Time_day	Time_hour	Time_month
0	2023-01-01 01:00:00	1	1	1
1	2023-01-01 02:00:00	1	2	1
2	2023-01-01 03:00:00	1	3	1
3	2023-01-01 04:00:00	1	4	1
4	2023-01-01 05:00:00	1	5	1
...	...	...	...	...
125570	2023-01-02 19:00:00	2	19	1
125571	2023-01-02 20:00:00	2	20	1
125572	2023-01-02 21:00:00	2	21	1
125573	2023-01-02 22:00:00	2	22	1
125574	2023-01-02 23:00:00	2	23	1

Figure 3. Temporal Features

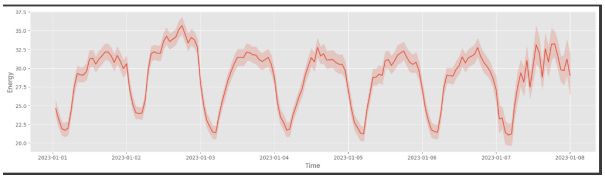

Figure 4. line-plot of Energy with respect to Time

2.2.3 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.

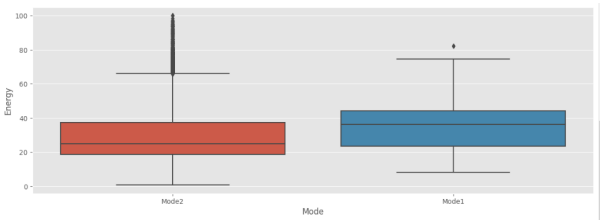

Figure 5. Box-plot of Energy with respect to Mode

2.2.4 Statistical features

The approach that helped me reach the top five in the leaderboard is using statistical techniques like aggregation and quantiles. This is crucial because our dataset has a big imbalance, which can lead to poor results. Using these techniques to create new features is the best way to improve our model's performance for several reasons:

- Aggregated features can help models generalize better by providing a more stable representation of the data.Especially in the case of unbalanced data, beside capturing essential information about the data distribution, central tendency, and variability, allowing the models to understand underlying patterns and relationships more effectively and make them less sensitive to outliers and extreme values.

- Quantiles provide information about the spread and distribution of data. So by including quantile features, the model can better generalize to unseen data points because it has learned the underlying structure of the data distribution.Also using quantiles makes the model more robust to outliers which is especially important with non-normally distributed data like the project dataset beside making more stable predictions.

I have written two functions for generating aggregations and quantiles.

The Agg function takes the dataset as an argument, along with two lists named cols1 and cols2. It iterates through each column col1 in cols1 and each column col2 in cols2. For each combination of col1andcol2, it calculates various aggregate statistics using the groupby

```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
```

Figure 6. script for aggregations and quantiles

method. These statistics include the mean, standard deviation, maximum, minimum, and median. It then adds the generated feature, such as col1-col2-max to the dataset.

Same thing for AddQauantiles , it takes three arguments which are the dataset , cols1 and cols2 and for each combination of col1 and col2 it calculates specif quantiles (10th,25th,75th and 90th percentiles) using the quatile method on the grouped data.The calculated quantile values are then mapped to the original dataset , creating new features similar to the Agg but with name like col1-col2-q75 or col1-col2-q10.

```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'])



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'])
```

Figure 7. statistical features generation

The figure below shows the statistical features generated from numeric features grouped by categorical features. The most valuable features among those are the ones grouped by base stations.

2.3 Features selection

As we mentioned from the beginning, most of the base features are noisy and unbalanced. In my approach, I dropped these features: ESMode4, Frequency, Bandwidth, and load. Additionally, I removed other statistically noisy features, including Time-hour-Frequency-min, Time-hour-Frequency-max, Time-hour-TXpower-min, RUType-load-min, RUType-load-max, and RUType-Antennas-mean.

3 Modeling

My approach is based on using gradient boosting algorithms such as LightGBM and CatBoost. In this section, I will present the hyperparameters tuning technique, along with the comparison of results, and finally, the best-selected model.

3.1 Hyperparameters tuning

The goal of hyperparameter tuning is to find the combination of hyperparameters that results in the best model performance, such as higher accuracy, lower error, or faster convergence.
For this tuning, I used Optuna[2] , which is an open-source Python library for hyperparameter optimization. It provides a framework for automating the process of searching for the best hyperparameters for machine learning models. The power of optuna that you define the search space for hyperparameters, specifying the ranges or distributions from which Optuna will sample values during the optimization process.

```python
# best set of parameters
best_params={'learning_rate': 0.05014932711138855,
 'num_leaves': 82,
 'reg_alpha': 2.5639389256690515,
 'reg_lambda': 0.12523931338344882,
 'n_estimators': 3760}
```

Figure 8. best set of parameters for lightgbm

```python
cat_params={'iterations': 61666,
            'depth': 5,
            'learning_rate': 0.054404170700668457,
            'l2_leaf_reg': 0.101929807646990755,
            'bagging_temperature': 0.7805657381624727,
            'random_strength': 0.7982157709406001,
            'border_count': 197,
             'loss_function':'MAE',
            'task_type':'GPU',
              }
```

Figure 9. best set of parameters for catboost

3.2 results and selection

In this part I will present to results of predictions from the boosting algorithms with their best parameters and the selected model .

	Public	Private
LightGBM	0.071636682	0.070974957
CatBoost	0.089422424	0.088889418
Weighted Sum Between the Two Models	0.071764620	0.071051690

Table 1. Results

As you can see, LightGBM provides the best score, indicating the best generalization.

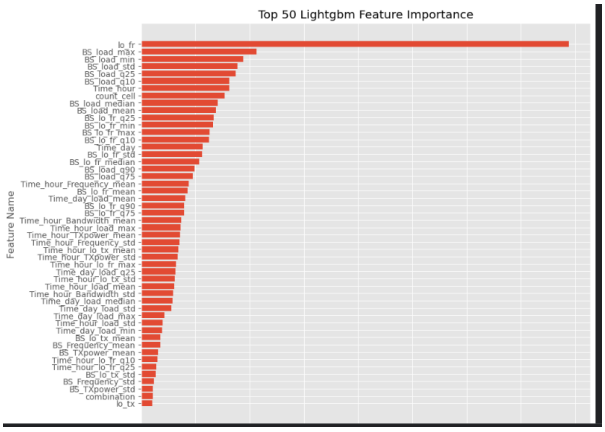

Figure 10. Top 50 features lighgbm

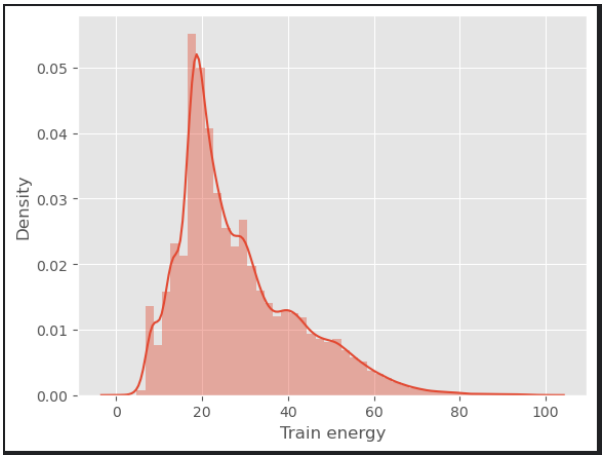

Figure 11. Train energy distribution

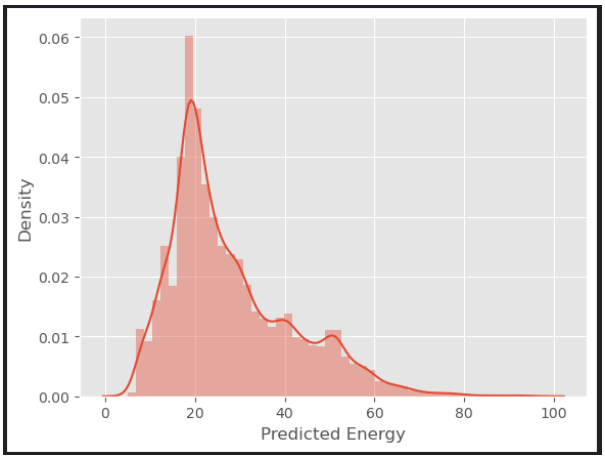

Figure 12. Predicted energy distribution

4 Future work

For future work, I will attempt to discover new relations between energy and features while also focusing on improving the models for better generalization. Additionally, I will explore the use of deep neural networks to capture more complex relationships, aiming for accurate predictions.

5 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.

Acknowledgement

Special thanks to the ITU team for organizing the competition, and to the Zindi team for hosting this competition.

References

[1] Zindi. ITU. Ai/ml for 5g-energy consumption modelling by itu ai/ml in 5g challenge. https://zindi.africa/competitions/aiml-for-5g-energy-consumption-modelling, 2023.

[2] Optuna. open-source python library for hyperparameter optimization. https://optuna.org/, 2023.