Spatio-Temporal Beam-Level Traffic Forecasting

Forecasting traffic throughput volumes within communication channels

Stephen Kolesh

Koleshjr@gmail.com

Abstract - This paper addresses the challenge of ***spatio-temporal beam-level traffic forecasting*** in communication networks, aiming to predict downlink throughput volume (DLThpVol) with high accuracy. The task involves analyzing multivariate time series data, including additional factors such as Physical Resource Block (PRB) utilization and user count, to capture the complex interactions that drive traffic patterns. The goal is to develop machine learning models that can effectively model both spatial and temporal dependencies in the data, surpassing baseline forecasting models.

Using advanced machine learning techniques, I leverage high-resolution traffic data with hourly granularity to accurately forecast traffic volume at the beam level. This research demonstrates the effectiveness of such models in predicting traffic across different base stations (spatial) and time periods (temporal), leading to more efficient network resource management. The findings suggest that improved traffic forecasting can reduce network congestion, conserve energy, and enhance user experience. This study also contributes to the *AI for Good* initiative by the International Telecommunication Union (ITU), which seeks to advance global development goals through the application of AI.

1. Introduction

The rapid expansion of communication networks, coupled with the increasing demand for data traffic, presents a significant challenge for efficient traffic management in modern network operations. Accurate forecasting of traffic throughput volumes is essential for optimizing network performance, resource allocation, and ensuring user satisfaction. in particular, beam-level technology, which employs multiple beams per cell, allows for finer granularity in service provision. As a

result, predicting traffic at the beam level has become crucial for managing network congestion and improving overall efficiency.

Traditional time series forecasting models such as ARIMA (Shumway & Stoffer) and Holt-Winters (Chatfield, 1978) have been commonly used to predict network traffic but are often limited when applied to complex, high resolution data. To address these challenges, this paper leverages ***Gradient Boosting Decision Trees (GBDT)*** specifically models like **CatBoost** (Dorogush, Ershov, & Gulin, 2018) and **LightGBM** (Ke, et al.) to predict downlink throughput volume (DLThpVol) at the beam level. GBDT models have demonstrated significant success in capturing multivariate time series dependencies, making them well-suited for this task. My approach utilizes features such as Physical Resource Block (PRB) utilization, user count, and temporal variables to enhance the accuracy of predictions.

The primary task involves predicting traffic volumes both one week ahead and six weeks ahead, requiring a model that can generalize effectively across these different forecasting horizons. Given the need for robust performance across varied timeframes, my model is designed to be adaptable to the two distinct scenarios. The evaluation metric used to assess model performance is ***Mean Absolute Error (MAE)***, chosen for its interpretability and its ability to provide a straightforward measure of prediction accuracy.

In this paper, I contribute to the field by developing GBDT-based models that surpass baseline forecasts and demonstrate their applicability to high-resolution beam level traffic data. My findings reveal that these models can significantly improve traffic management, reduce congestion, and promote energy efficiency in communication networks.

2. Datasets

The datasets provided for this study include detailed network performance metrics across multiple base stations and beams, captured at hourly intervals over a five-week period. The data is critical for modeling and forecasting downlink throughput volume (DLThpVol) at the beam level. Each dataset corresponds to a specific network performance metric and

offers multivariate time series data to support the development of robust forecasting models.

The challenge involves forecasting DLThpVol for 2,880 beams across 30 base stations, each of which is divided into 3 cells, with each cell containing 32 beams. The data is recorded at an hourly resolution, resulting in a rich dataset that captures the fine-grained dynamics of traffic flow within the network.

The four datasets used in this study are as follows:

- **traffic_DLThpVol.csv**: This dataset contains the target variable, representing the downlink throughput volume (DLThpVol). It is the primary focus of the forecasting task.
- **traffic_DLThpTime.csv**: This dataset records the throughput time, providing additional time-related features that are essential for capturing temporal patterns in network traffic.
- **traffic_MR_number.csv**: This dataset provides the number of users connected to each beam, offering insights into how user density affects network traffic at different times.
- **traffic_DLPRB.csv**: This dataset captures the Physical Resource Block (PRB) utilization, which reflects the allocation of network resources and can be a critical factor influencing throughput volume.

These datasets together form the basis for developing a multivariate time series forecasting model, designed to predict future values of DLThpVol based on historical data. The high-resolution, beam level data provided in these datasets allows for a detailed analysis of both spatial and temporal dependencies, crucial for accurate forecasting of network traffic in communication networks.

The objective of the task is to predict the DLThpVol for two distinct time horizons: one week ahead and six weeks ahead. This requires the model to generalize across varying timeframes while capturing the intricate patterns present in the dataset.

2.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is an essential step in understanding the underlying patterns and relationships within

the data, which can inform the development of accurate forecasting models. The high resolution, beam-level datasets provided offer a rich source of multivariate time series data, encompassing throughput volume (DLThpVol), throughput time, Physical Resource Block (PRB) utilization, and user count.

In this section, I explore key trends, distributions, and correlations within the data to uncover potential insights for model design.

2.1.1 Distribution of Throughput Volume (DLThpVol)

Understanding the distribution of DLThpVol values across all beams is crucial for identifying the overall traffic trends in the network. We can clearly see that the data is highly right skewed and even applying sqrt transformation did not help that much.

Before sqrt transformation

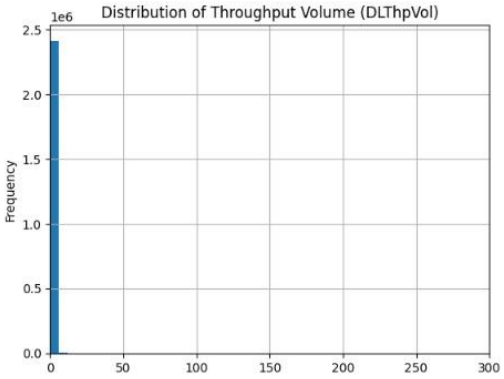

Figure 1: Distribution of Throughput Volume

After sqrt transformation

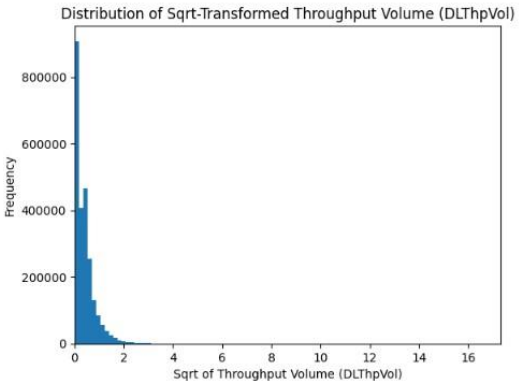

Figure 2: Distribution of throughput volume(sqrt transform)

2.1.2 Time Series Plot: Hourly Traffic Volume Trends

The variation in traffic volume over hours provides insight into temporal patterns that could impact forecasting accuracy. The following plot shows how DLThpVol changes throughout the week. The plot exhibits a clear cyclical pattern, with peaks and troughs repeating regularly over time, suggesting daily periodicity in traffic

volume. The peaks indicate higher throughput during specific hours, likely to correspond to peak network usage times, while the troughs correspond to lower usage periods

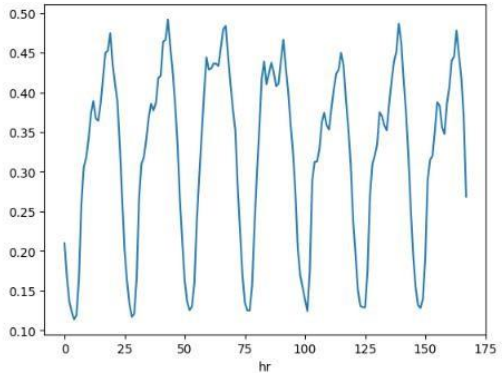

Figure 3: Hourly Traffic Volume Trends

2.1.3 Time Series Plot: Weekly Traffic Volume Trends

Analyzing how traffic varies across different weeks helps in assessing seasonal effects or long-term trends in the data.

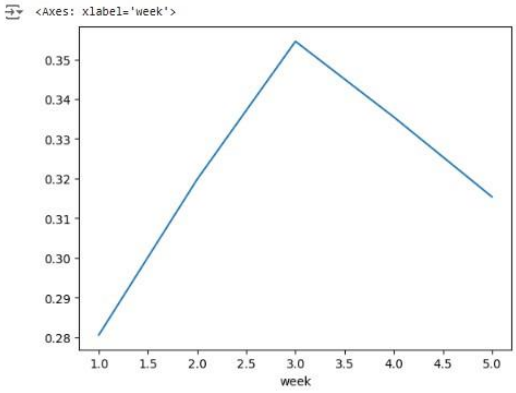

Figure 4: Weekly Traffic Volume Trends

2.1.4 Box Plot (Base Station vs. Throughput Volume)

The box plot below illustrates the distribution of throughput volume across different base stations. Significant variability and potential outliers are evident between base stations, suggesting differences in usage patterns.

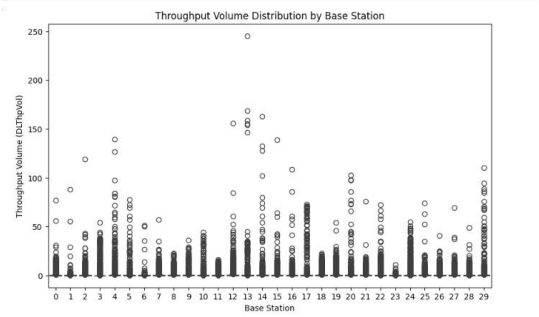

Figure 5: Throughput Volume Distribution by Base Station

2.1.5 Beam-level Traffic Volume Heatmap

A heatmap visualization of traffic volume across beams reveals how traffic is distributed spatially, allowing us to identify areas with consistently high or low traffic. From the below plot, some beams (e.g., around 10-15) consistently show higher traffic volumes, appearing as warmer colored horizontal bands across the week while other beams (e.g., at the bottom) show consistently low traffic, appearing as persistent blue bands.

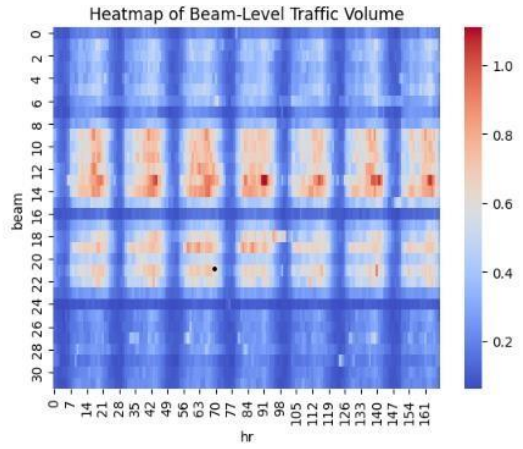

Figure 6: Heatmap of beam-level traffic volume

2.1.6 Scatter Plots

Two scatter plots were created to visualize the relationships between throughput volume and two other critical metrics: PRB utilization and user count.

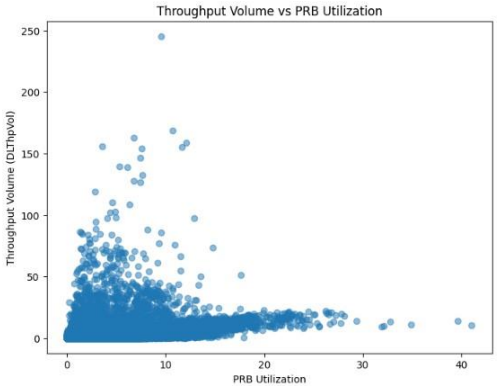

Figure 7: Throughput Volume vs PRB Utilization

The data points are concentrated at low PRB utilization values, with most values of throughput volume below 100, and a sparse distribution beyond 10 units of PRB utilization. There are very few instances where high PRB utilization leads to a throughput volume exceeding 100, suggesting that higher throughput does not necessarily correspond to increased PRB utilization in the majority of cases.

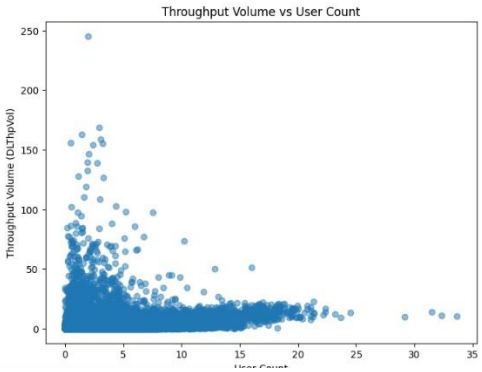

Figure 8: Throughput Volume vs User Count

Most of the points are clustered around lower user counts (0-5 users) with throughput volume below 100. However, there are occasional outliers where even with low user counts, the throughput volume spikes significantly (above 100). There is a relatively low density of data points with user counts above 10.

2.2 Feature Engineering

To enhance the predictive capabilities of the models, various feature engineering techniques were applied to extract temporal and group-based patterns from the data. The primary goal was to capture the relationships between historical values of key variables such as throughput

volume (my Target), PRB utilization, user count and throughput time. These features enable the model to learn from past trends and anticipate future values more effectively.

2.2.1 Rolling Features

Rolling features were generated to capture statistical summaries over specific time windows. This technique helps to aggregate recent historical data and provide a smooth temporal view of trends

- Rolling windows: For each group of base station, beam and cell type, rolling statistics (mean, median , standard deviation and quantiles) were computed over two window sizes: 168 hours(one week) and 336 hours(two weeks). These windows align with the temporal nature of the data, which is recorded hourly.
- Shift Period: A shift period of 168 hours (one week) was applied to prevent data leakage, ensuring that only past values are used to calculate the rolling statistics.
- Statistics: The statistics generated include
  - Mean: Average value of the window
  - Median: Middle value, providing a robust measure of central tendency
  - Standard Deviation (std): Measure of variability if dispersion
  - Quantile: The 25^{th} and 75^{th}
  - percentiles were computed to capture the range and distribution of values over the window
- This approach was applied to the following variables:
  - Target (throughput volume)
  - PRB Utilization (prb)
  - Throughput Time(time)

2.2.2 Expanding combination

Expanding features were created to capture cumulative statistics over time, providing insights into the long-term evolution of key variables.

- **Expanding Statistics**: For each group of base station, beam, and cell type, expanding mean and standard deviation were computed using

different shift periods (168, 336, and 504 hours).

- **Shift Periods**: These shift periods allow for the incorporation of weekly, bi-weekly, and tri-weekly cumulative statistics, which is crucial for capturing long-term temporal dependencies in the dataset.

Expanding features were generated for the following variables:

- Target (throughput volume)
- PRB Utilization (prb)
- User Count (mrno)
- Throughput Time (time)

2.2.3 Weekly Aggregation Features

In addition to rolling and expanding features, weekly mean aggregations were calculated to further summarize historical trends. Two additional features were created:

1. **Previous Weekly Mean**: The mean value of Target for the previous week was computed and mapped to the current week.
2. **Beam-Level Weekly Mean**: For each base_station, beam, and cell type, the mean Target value for the previous week was calculated and mapped accordingly.

These weekly features provide the model with information about general patterns from the previous week, giving it more contextual information.

2.2.4 Shifted Features

To capture temporal lags, shifted features were created for key variables. This helps the model learn from specific time lags and predict future values based on recurring patterns.

- **Shifted Features**: For each base_station, beam, cell_type, daily_hr, and day, shifted values of Target, mrno, and prb were created for the previous 1 to 4 hours within the same hour of the day.

This ensures that the model can learn temporal patterns at the granularity of individual hours, which is important for predicting future values at hourly intervals.

2.2.5 Target Encoding:

I also implemented a more sophisticated

feature engineering technique known as target encoding. Target encoding is a method that replaces categorical variables with a statistical measure of the target variable, effectively capturing the relationship between the categorical features and the target. This was done in a fold's manner.

- Group Columns Used:
    - ['base_station'],
    - ['hr'],
    - ['daily_hr'],
    - ['beam'],
    - ['base_station', 'beam'],
    - ['base_station', 'hr'],
    - ['base_station', 'daily_hr'],
    - ['base_station', 'cell_type'],
    - ['base_station', 'cell_type', 'daily_hr'],
    - ['base_station', 'cell_type', 'beam'],
    - ['base_station', 'cell_type', 'beam', 'daily_hr'],
    - ['base_station', 'cell_type','beam', 'day'],
    - ['base_station', 'beam', 'daily_hr'],
    - ['beam', 'daily_hr'],
    - ['beam', 'cell_type'],
    - ['beam', 'cell_type', 'daily_hr'],
- Target Columns Used:
    - 'Target'(throughput volume)
    - 'mrno' (user count)
    - 'prb' (physical resource block utilization)

The essence of performing target encoding in a fold's manner lies in its ability to mitigate overfitting while capturing valuable information about the relationship between categorical features and the target variable. By implementing target encoding within a cross-validation framework, I ensure that the encoding for each fold is based on out-of-fold data, preventing data leakage and providing a more robust estimation of the true relationship between features and target.

My implementation calculates various statistics ***(mean, standard deviation, skewness, minimum, maximum, and specific percentiles)*** of the target variable for each combination of group columns. This approach allows us to capture different aspects of the target distribution within each categorical group, potentially providing richer information to the model.

The fold-based approach involves:

- Encoding the test set using the entire training data to ensure consistent encoding across all folds.
- For each fold in the training set:
  - Encoding the validation data using statistics computed from the other folds.
  - Storing the encoded validation data for later concatenation.

This method ensures that the model is always trained and validated on data that has been encoded using independent samples, maintaining the integrity (prevent data leakage) of the cross-validation process and providing a more accurate estimate of model performance.

3. Cross Validation Strategy

Cross-validation is a critical technique used in machine learning to assess the generalization performance of a model and ensure that it performs well on unseen data. In this study, I employed **Stratified K-Fold Cross-Validation** to evaluate my model's robustness and reliability. The following outlines the approach used in my analysis:

1. **Stratified K-Fold Definition**:
   Stratified K-Fold CrossValidation is a variation of KFold Cross-Validation that maintains the distribution of target classes in each fold. This is particularly important when dealing with imbalanced datasets, as it ensures that each fold is representative of the overall dataset.
   In this case I stratified using 'base_station'
2. **Implementation**: I utilized the StratifiedKFold class from the sklearn.model_selection module, specifying the following parameters:
   - n_splits = 10: This indicates that the dataset is to be divided into 10 distinct folds. The model will be trained on 9 folds and validated on the remaining fold in each iteration.
   - shuffle=True: This parameter shuffles the data before splitting into batches, which helps ensure

that the model is evaluated on different subsets of data, reducing potential bias from the order of the data.

- random_state=42: Setting the random state ensures reproducibility of the results by controlling the shuffling process.

This approach allows us to validate the performance of the Gradient Boosting Decision Trees model reliably, ensuring that the findings of my traffic forecasting task are robust and applicable to real-world scenarios and generalize across all different base stations.

4. Modelling

In this section, I describe my modeling approach, which utilizes two powerful gradient boosting algorithms: LightGBM and CatBoost. I implemented two distinct feature selection approaches for each model, aiming to capture different aspects of the data and potentially improve overall predictive performance.

4.1 Feature Selection

My feature selection strategy was tailored to address two distinct prediction tasks:

- Predict for Week 6: This was the immediate week following the first 5 training weeks.
- Predict for Week 11: This had a 5-week gap from the last training week.

I employed two feature selection approaches to optimally handle these different prediction scenarios:

4.1.1 Approach 1 (for Week 11 prediction):

This approach used a subset of the original features, excluding shifted, previous, and expanding features. It also omitted percentile-based target encoding features. The rationale behind this selection was to focus on more stable, long-term predictive features that would remain relevant despite the 5-week gap. By excluding time dependent features like shifted and previous values, I aimed to create a model more robust to the time gap and less prone to overfitting on short-term patterns.

4.1.2 Approach 2 (for Week 6 prediction):

This approach utilizes all selected columns, including shifted, previous, and expanding features, as well as all target encoding features. The inclusion of these time-dependent features was appropriate for Week 6's prediction as they could capture recent trends and patterns that were likely to persist into the immediate next week. This richer feature set allowed the model to leverage short-term dynamics in the data.

By using these two distinct feature sets, I aimed to optimize my predictions for both the short-term (Week 6) and longer term (Week 11) forecasting tasks. This dual approach acknowledges the different challenges posed by immediate versus gap predictions and attempts to leverage the most appropriate information for each task.

The use of different feature subsets also allowed us to compare the performance of models trained on more stable, long-term features versus those trained on a fuller feature set including short-term indicators.

4.2 LightGBM Model

LightGBM, a gradient boosting framework that uses tree-based learning algorithms, was my first choice model

4.2.1 Parameter Selection

I used different sets of hyperparameters for each approach:

- **Approach 1 Parameters**

```
params = {
'learning_rate': 0.02053584056236377,
 'num_leaves': 254,
'max_depth': 10,
'feature_fraction': 0.6696720940035359,
'bagging_fraction': 0.7228804381516678,
 'bagging_freq': 8,
'min_child_samples': 100,
'lambda_l1': 2.4850576136378315e-06,
'lambda_l2': 1.694355465523766e-08,
'n_estimators': 1000
}
```

- **Approach 2 parameters**

```
params= {
'learning_rate': 0.08304613332693336,
'num_leaves': 151,
```

'max_depth': 9,

'feature_fraction': 0.7095128621678289,

'bagging_fraction':0.9361601754529758
'bagging_freq': 1,

'min_child_samples': 22,

'lambda_l1':6.421772454121583e0,

'lambda_l2': 0.0034348407467305964,

'n_estimators': 5000

}

4.2.2 Model Training and Evaluation

I trained the LightGBM models using a 10-fold cross-validation strategy. For each fold, I:

1. Split the data into training and validation sets.
2. Applied a square root transformation to the target variable.
3. Trained the model on the training set and evaluated it on the validation set.
4. Made predictions on the validation set and inverse transformed them.
5. Calculated the Mean Absolute Error (MAE) for the fold.
6. Made predictions on the test set and stored them in an array for averaging later

4.2.3 Results

- ***Average MAE for Approach 1:*** 0.19712383813106563
- ***Average MAE for Approach 2:*** 0.1912539406395293
- ***Public Leader Board Score:*** 0.192539542

4.2.4 Feature Importance

Feature Importance Approach one

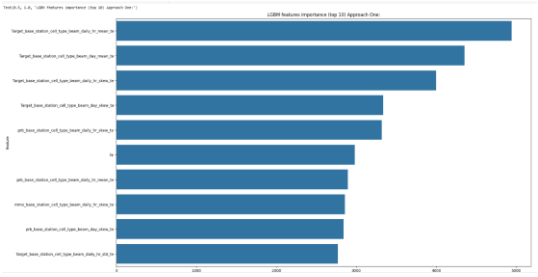

Figure 9: LightGBM Feature Importance approach one

Feature Importance Approach two

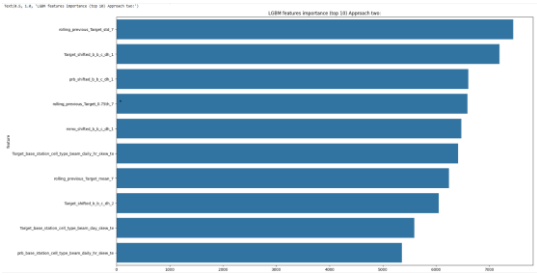

Figure 10: LightGBM Feature Importance approach two

4.3 Catboost Model

CatBoost, another gradient boosting library, was my second choice of model.

4.3.1 Parameter Selection

For catboost I used the same set of parameters for both approaches

```
cat_params = {
    'learning_rate': 0.020218465729343698,
    'depth': 9,
'l2_leaf_reg': 1.339103723284128e-06,

'random_strength':6.000809910512735e-07,
'bagging_temperature':0.38040823680407604,
'leaf_estimation_iterations': 7,
'iterations': 15000
}
```

4.3.2 Model Training and Evaluation

I trained the catboost models using a 10-fold cross-validation strategy. For each fold, I:

1. Split the data into training and validation sets.
2. Applied a square root transformation to the target variable.
3. Trained the model on the training set and evaluated it on the validation set.
4. Made predictions on the validation set and inverse transformed them.
5. Calculated the Mean Absolute Error (MAE) for the fold.
6. Made predictions on the test set and stored them in an array for averaging later

4.3.3 Results

- ***Average MAE for Approach 1:*** 0.1972058677455319
- ***Average MAE for Approach 2:*** 0.19177680810324244
- ***Public Leader Board Score:*** 0.191880897

4.3.4 Feature Importance

Feature Importance Approach One

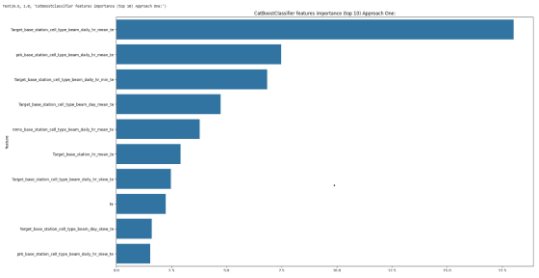

Figure 11: Catboost Feature Importance approach one

Feature Importance Approach Two

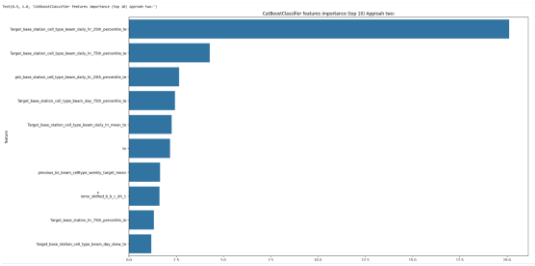

Figure 12: Catboost approach two feature importance

4.4 Final Submission

The final submission was a weighted ensemble of the catboost model with weight
0.6 and lightgbm model with weight 0.4. This was guided by the feature importances as each model was prioritizing a different set of features and the public leaderboard score.

4.4.1 Result

- .**Public Leader Board Score**
  0.191948815

5. Conclusion

This study focused on predicting cellular network metrics for two distinct time horizons: the immediate next week (Week 6) and a future week with a 5-week gap (Week 11). My approach leveraged advanced feature engineering techniques, particularly target encoding, and employed two powerful gradient boosting algorithms, LightGBM and CatBoost, each with two different feature selection strategies.

Key findings of my study include:

1. The effectiveness of target encoding in capturing complex relationships between categorical variables and the target metrics.
2. The importance of tailoring feature selection to the prediction time horizon, as evidenced by my dual approach strategy.
3. The robust performance of both LightGBM and CatBoost models, highlighting the suitability of gradient boosting methods for this type of prediction task.
4. The potential benefits of ensemble methods, combining predictions from different models and folds to enhance overall predictive accuracy.

My results demonstrate the feasibility of predicting cellular network metrics with reasonable accuracy for both short-term and longer-term horizons. The differences in performance between my two feature selection approaches provide insights into the temporal dynamics of the predictive

features and their relevance over different time scales.

6. Future Work and Analysis

While my current approach using Gradient Boosting Decision Trees (GBDT) models, specifically CatBoost and LightGBM, has shown promising results in predicting downlink throughput volume (DLThpVol) at the beam level, there are several avenues for future work that could further enhance the effectiveness and applicability of my models:

1. **Multi-Step Forecasting:** Develop and compare different strategies for multi-step forecasting, such as direct multi-step, recursive, or hybrid approaches. This could improve the model's ability to predict across different time horizons, from one week to six weeks ahead.

2. **Feature Engineering Optimization:** Building on the feature importance analysis already conducted, investigate more advanced feature engineering techniques. This could include creating more sophisticated lag features, rolling statistics, or domain-specific engineered features that capture network behavior patterns.

3. **Model Interpretability:** While GBDT models often outperform traditional time series models in accuracy, they can be less interpretable. Explore methods to enhance model interpretability, such as SHAP (SHapley Additive exPlanations) values, which could provide insights into how different features contribute to predictions across various time horizons.

By pursuing these directions, future work could not only improve the accuracy and reliability of beam-level traffic predictions but also provide deeper insights into network behavior and trends. This could ultimately lead to more efficient network management, improved user experience, and more informed decision-making in cellular network operations, particularly in the context of emerging technologies like 5G and beyond.

References

1. Shumway, R. H., Stoffer, D. S., Shumway, R. H., & Stoffer, D. S. (2017). ARIMA models. *Time series analysis and its applications: with R examples*, 75-163.
2. Chatfield, C. (1978). The Holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *27*(3), 264-279.
3. Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.
4. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.