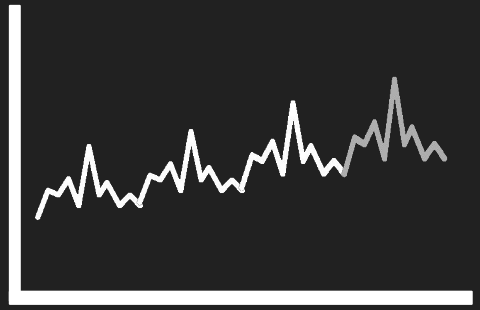
NOKIA

Network and Service Time Series Auto-Forecast



Team Members: Jacob Poston, William Easterby, Dillon Gentry, Daniel Lazcano, Shoyon Kermany

Faculty Advisor: Ari Arapostathis

TA Advisor: Venkata Suresh Rayudu

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

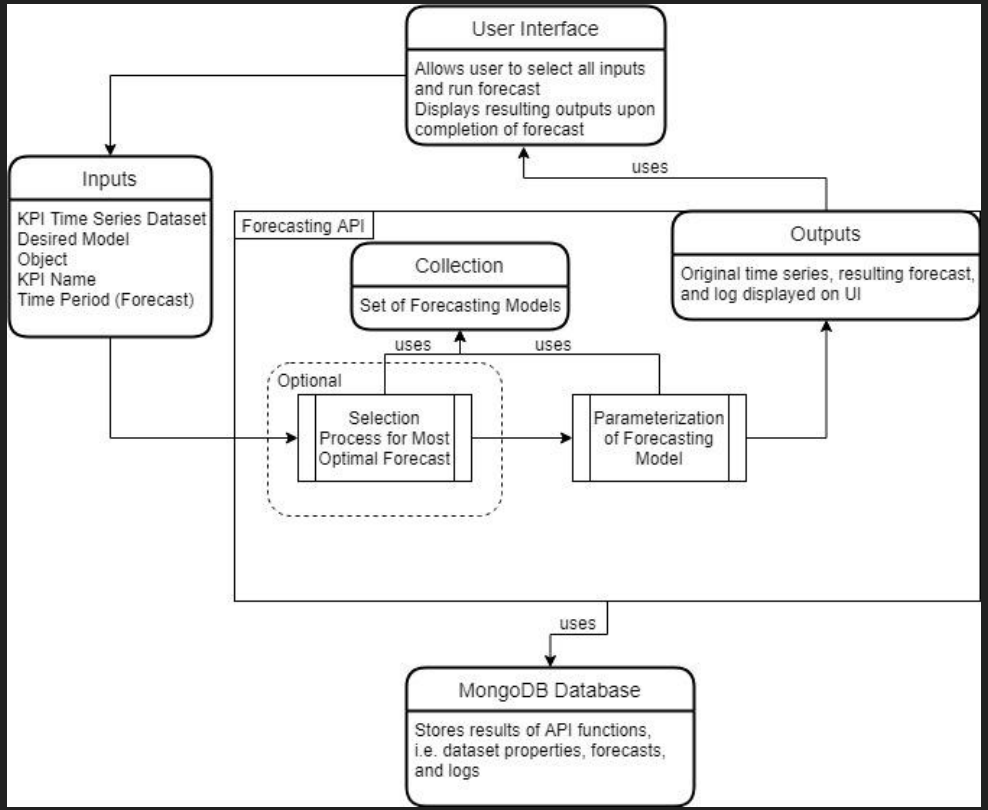
Problem Statement

In preparation for 5G, Nokia is interested in having the ability to use various time-series forecasting models to create accurate predictions of their data.

This data comes in the form of “KPI”s which are “Key Performance Indicators”. These are data points from sensors on Nokia towers.

We were tasked to include multiple forecasting models within an API that will automatically choose the best model for forecasting based on the data’s characteristics.

Block Diagram

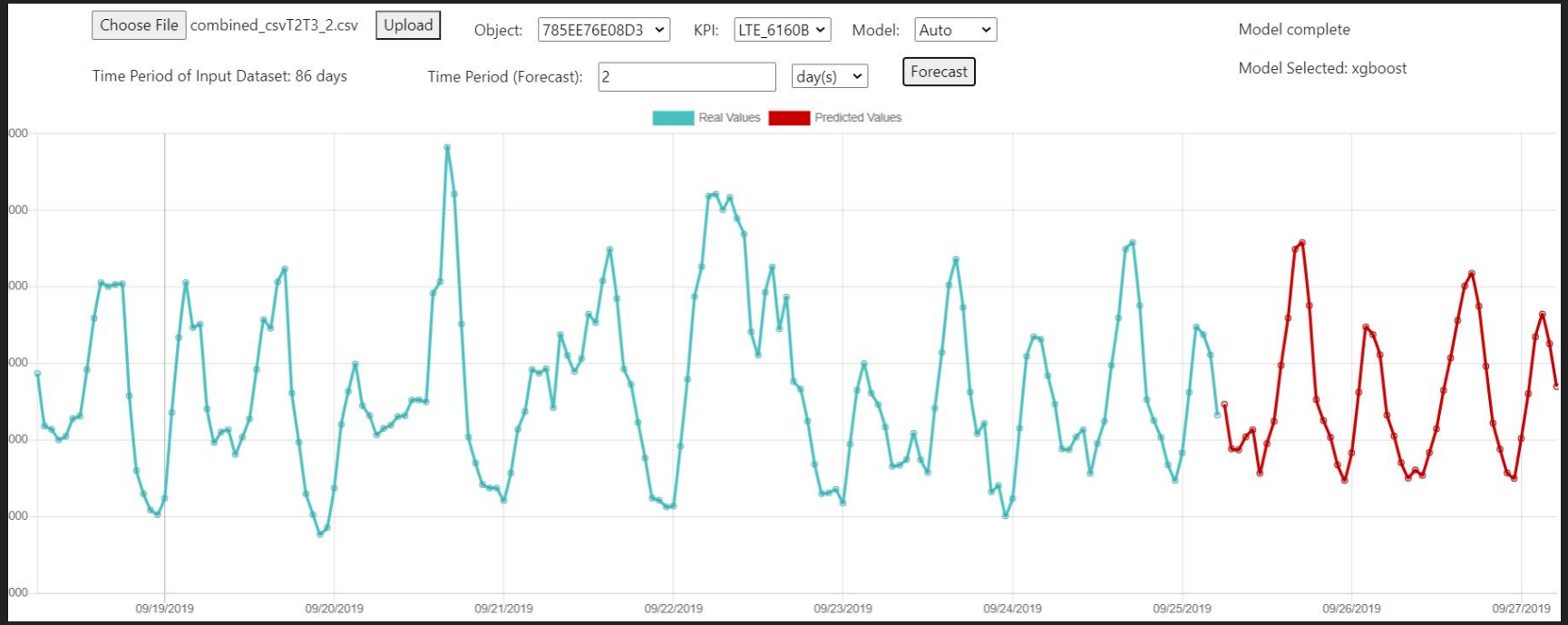


Overview

Frameworks we used in development:

We’ve implemented 3 separate models: SARIMA, XGBoost, and LSTM.

User Interface - Auto

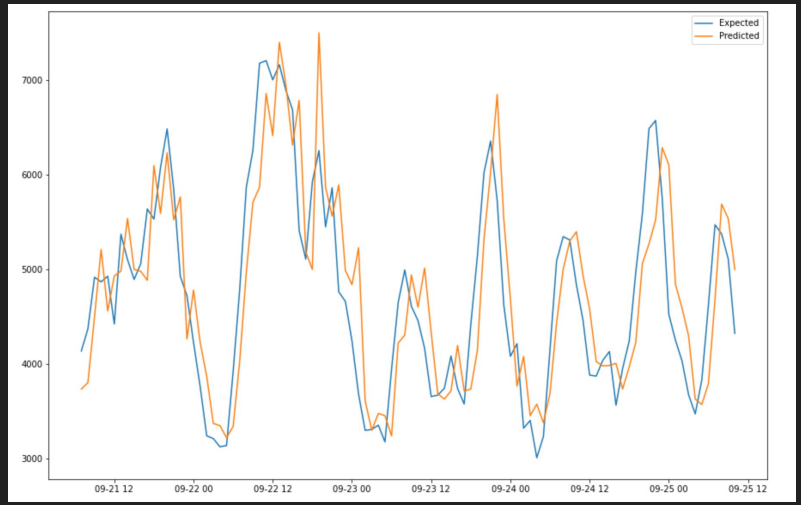


API

- Built on Flask and Flask-RESTful
- Handles forecasting and automation
- Uses MongoDB for persistent storage
- Available endpoints
  - Upload Dataset
  - Get Dataset Details
  - Get KPI from Dataset
  - Create (Start) New Forecast
  - Get Forecast
- Also includes logging for easy debugging

Forecasting/Automation

- Forecasting
  - Trained the various models on the entire dataset and forecasted future values
- Automation
  - Split the dataset into train (1 week) and test (24 hours) datasets
  - Train models on train set and predicted values for the test dataset
  - Chose model with minimum RMSE



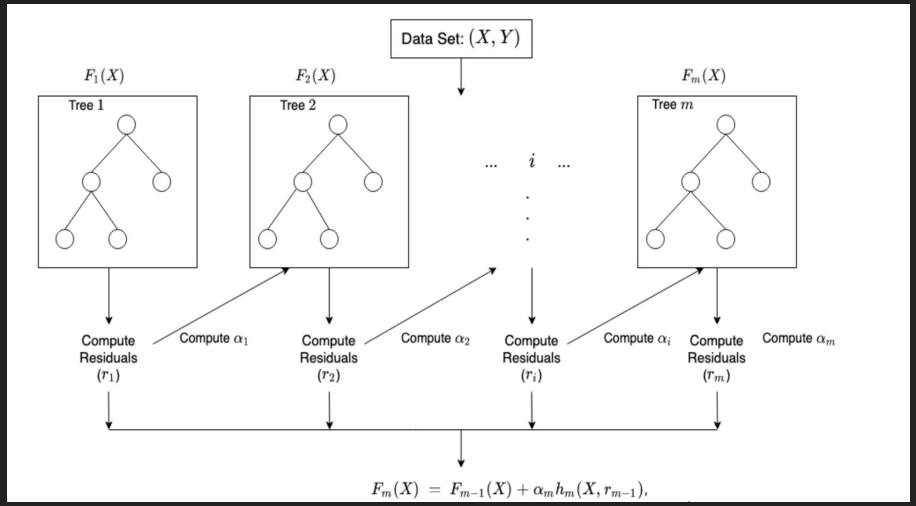
SARIMA Model

- Seasonal Autoregressive Integrated Moving Average
  - Seasonal because our KPIs have 24-hour patterns
  - Essentially just a linear regression model that uses time-series data lags and lagged forecast errors
  - Must make time-series stationary and non-season, which model handles
- More traditional model for time-series forecasting
- Used statsmodels Python package

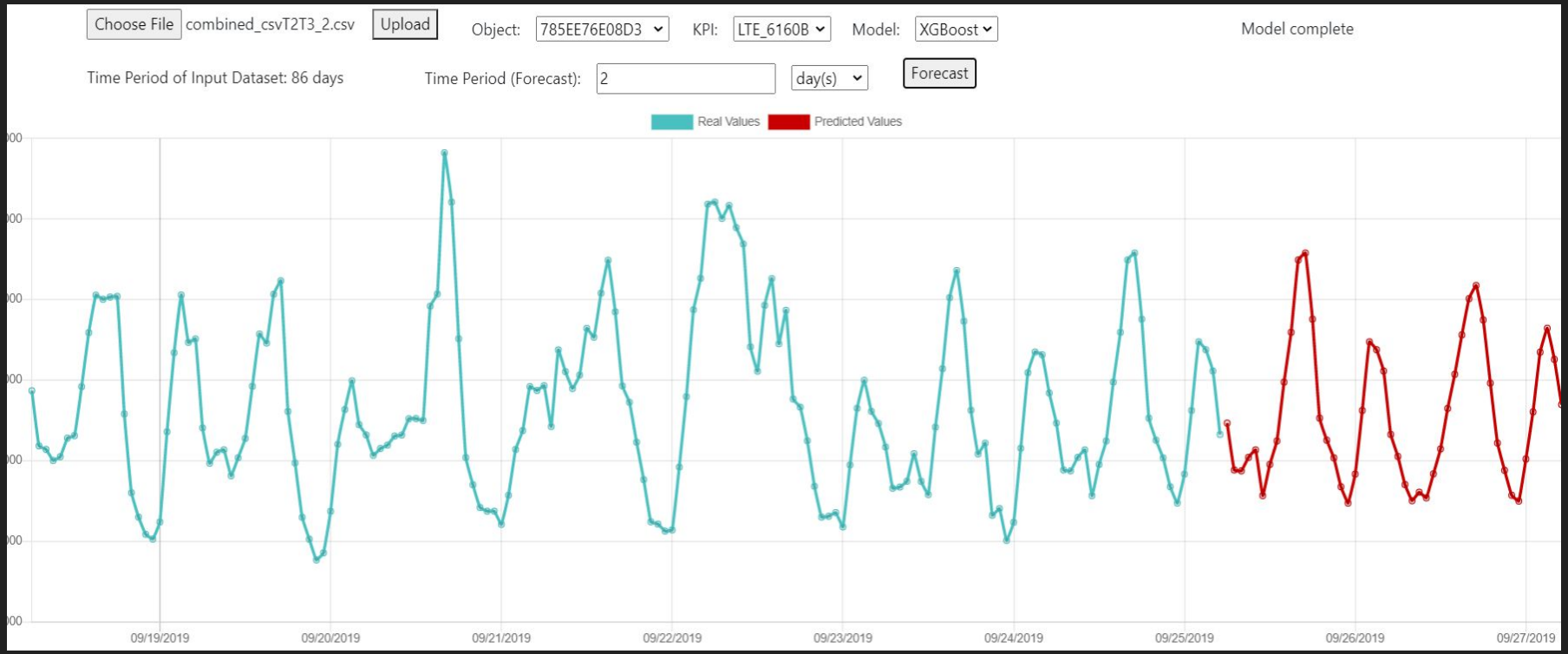
User Interface - SARIMA

XGBoost Model

- XGBoost stands for eXtreme Gradient Boosting
- Very fast and accurate compared to other boosting methods
- Supervised learning algorithm that predicts target variables by combining a set of weaker models
- Our project uses XGBRegressor which uses regression trees as weak learners

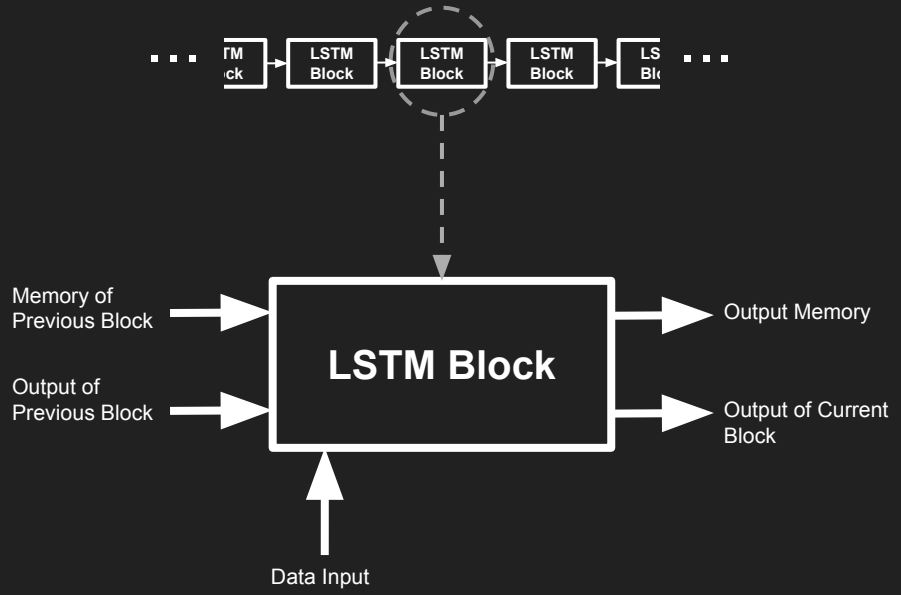


User Interface - XGBoost

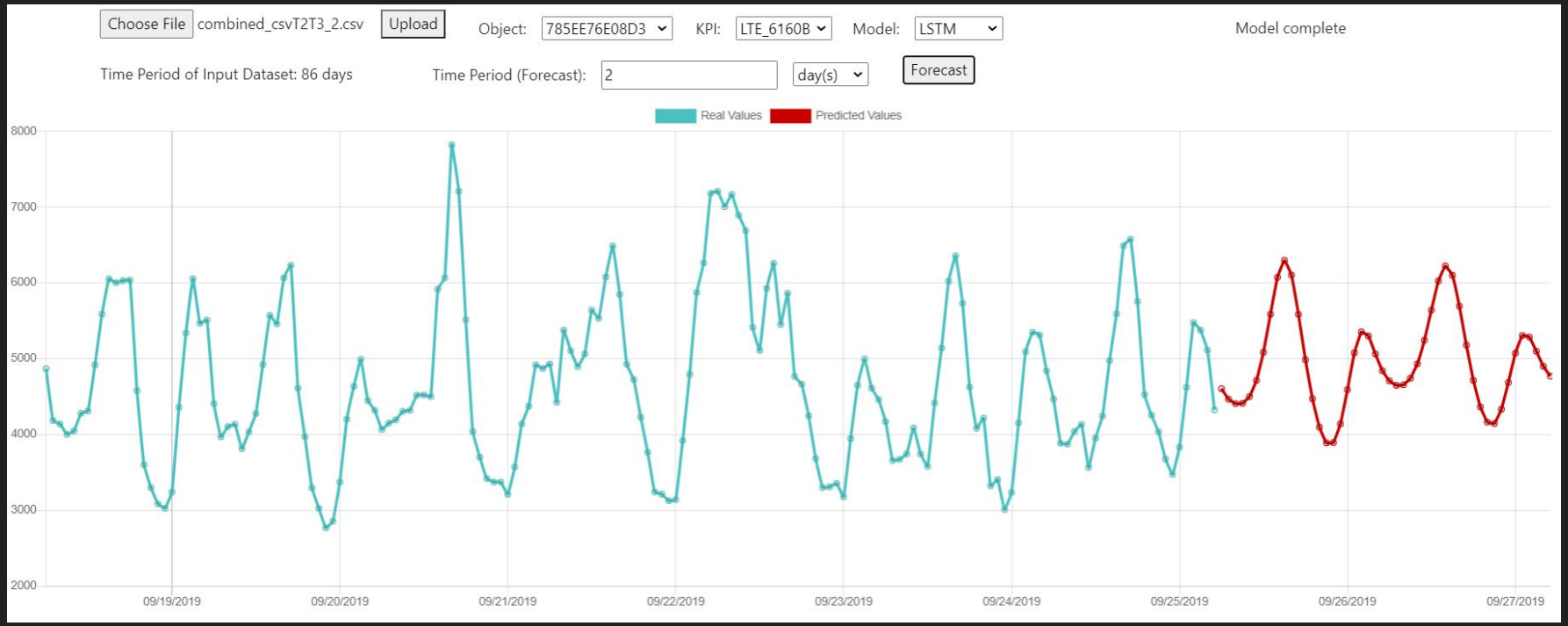


LSTM Model

- Long Short-Term Memory recurrent neural networks
  - Created to solve the issues of vanishing/exploding gradients
  - Main feature of model is memory
- Used Keras to implement LSTM model in Python



User Interface - LSTM



Evaluation - Root Mean Square Error

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

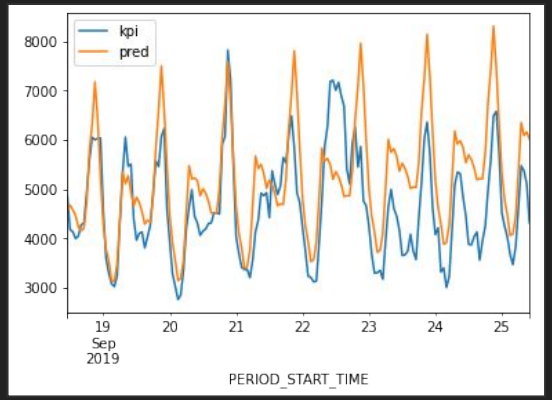
For each model, we selected 5 KPIs, and evaluated each across 1 week of predictions (N=168)

Our goal was to minimize the average RMSE of each model

Evaluation - Model Performances

SARIMA

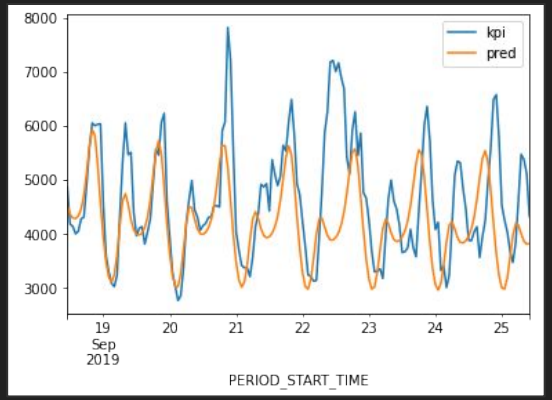
- Average RMSE: 962.78
- Best RMSE: 9.73
- Worst RMSE: 1097.04



RMSE: 962.78

LSTM

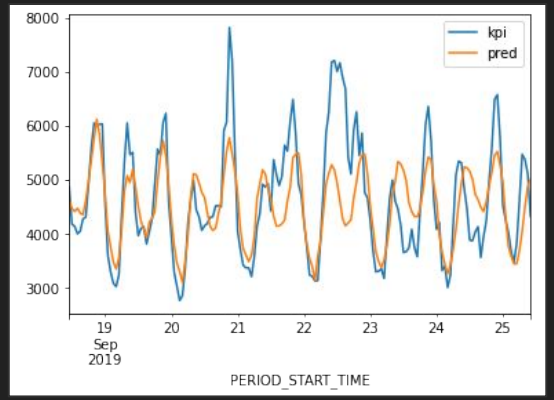
- Average RMSE: 934.73
- Best RMSE: 5.79
- Worst RMSE: 2038.81



RMSE: 1039.26

XGBoost

- Average RMSE: 765.24
- Best RMSE: 5.90
- Worst RMSE: 1708.72



RMSE:752.57

Recommendations

- Introducing more models to the selection process
  - Models that fit certain data types better than current models
- Faster and smarter auto model selection process
- Add new input parameters to add more customization to the predictions
- Add multiple KPI input format to UI

Conclusion

Delivered a product to be handed off to Nokia network engineers.

Learned about industry standard development practices.

Gained experience working remotely with a team and managing a technical project.

Gathered knowledge on time series forecasting models and machine learning.

Met almost all the project requirements

Questions?