**Introduction**

In recent years, controlling traffic loads for internet providers has become an integral part of their task because the Internet is an inseparable part of people’s lives. Consequently, predicting and monitoring the network's traffic load is essential for their plan to set up BTS and radio transmitter sources in more congested parts of the city. Also, it will help them make plans for energy consumption and they can have a schedule for when all BTS should be On or off. This is why data scientists and analysts gather to extract traffic consumption patterns and predict traffic consumption.

**Task 1:**

I use a Python script with Jupyter Notebook IDE to answer this part of the question. I Utilize the Pandas library to read all data in a data frame. The characteristics of my laptop in which the program is run are RAM: 12 GB, CPU:i7 6800, and GPU: 4 GB 960m on Windows 11 OS.

After reading all traffic data within 2 months for all the given cities to calculate the probability density function, we should add all the traffic that was consumed in an area. The result is shown in Figure 1. Loading datasets from my local Hardware in a data frame took about 5 minutes.

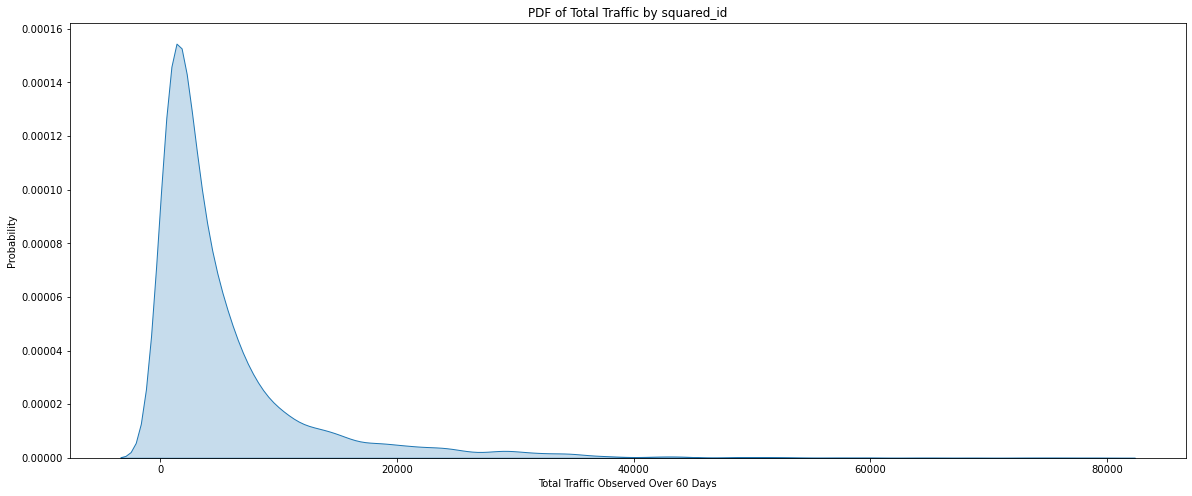


Figure 1 - probability density function of the traffic in the target city in the target city, computed over 10,000 samples that each represent the total two-month traffic in a geographical area

As it is obvious, the main traffic distribution in two months changes around 5000 call detail records (CDR). This means that most cities’ internet traffic within 2 months is about 5000, and only a few cities consume more than 20000 CDR. I also plotted the cumulative probability density function (CDF) to interpret density easily. The derivative at 5000 CDR is so High. This means that more than 80% of cities have 5000CDR internet traffic while the number of cities with more than 10000 CDR is so rare because the derivative becomes zero rapidly after 20000CDR.

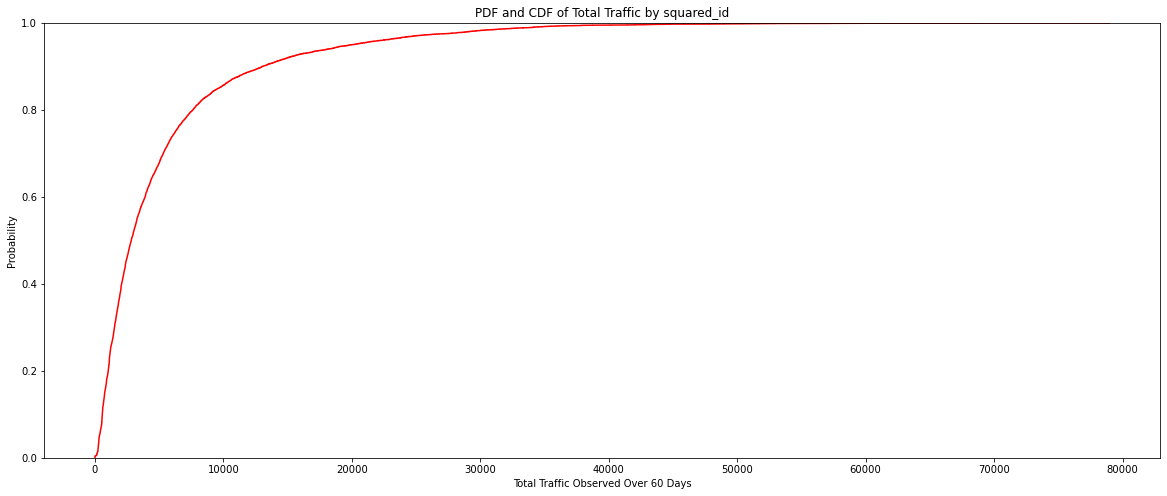


Figure 2- cumulative density function of the traffic in the target city in the target city, computed over 10,000 samples that each represent the total two-month traffic in a geographical area.

The figure of the time series of network traffic during the first two weeks of the area that has the highest total traffic is shown in Figure 3

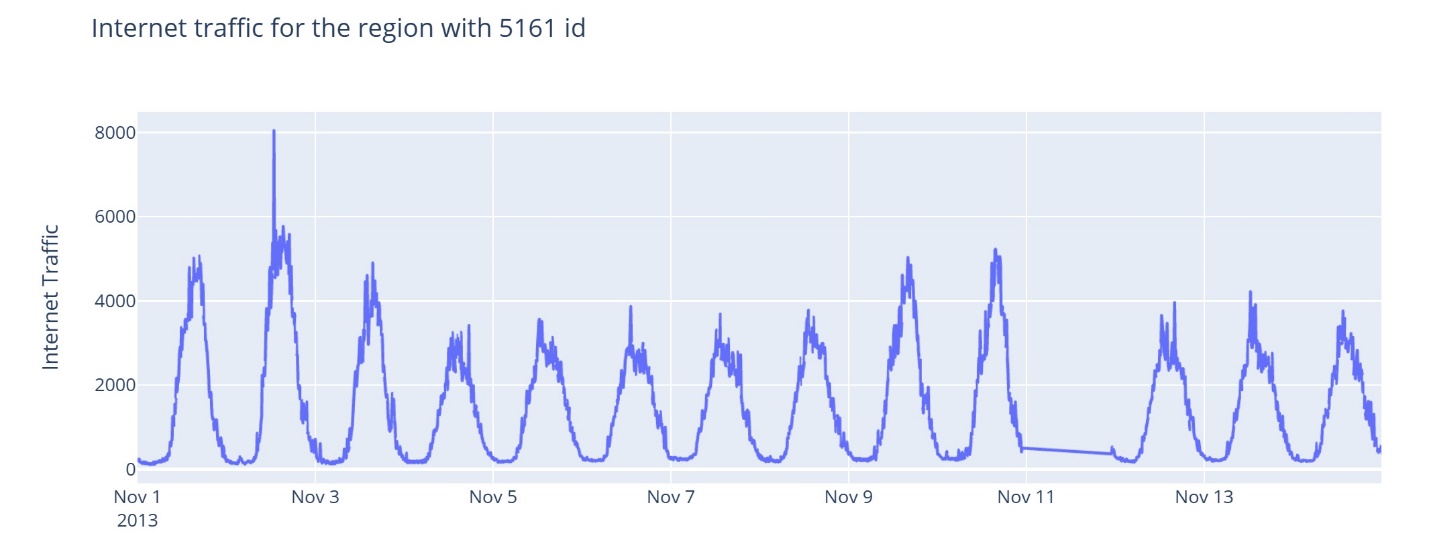


Figure 3 - Area with the highest Internet traffic

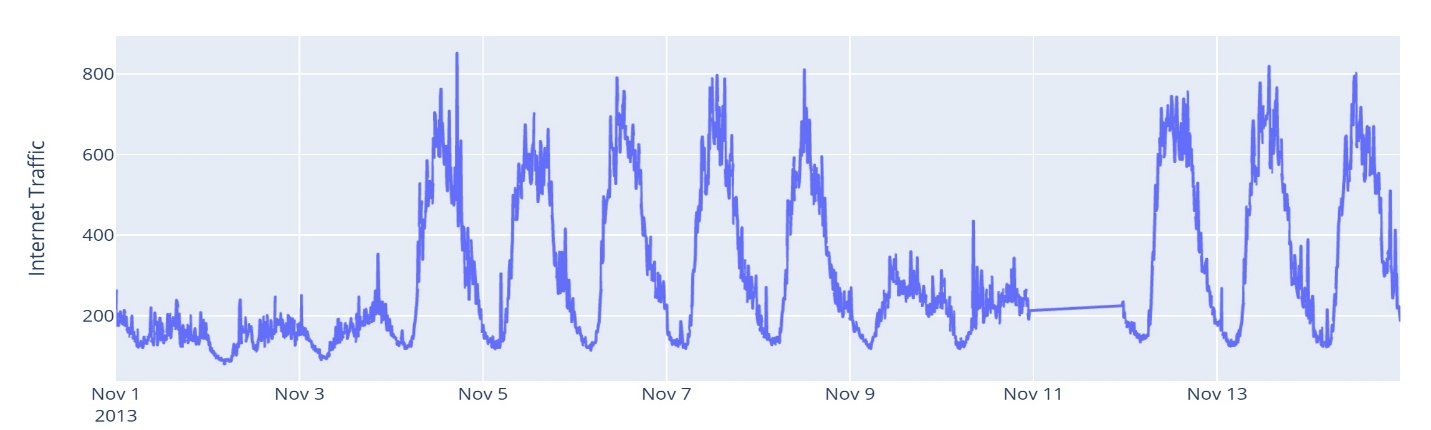


Figure 4 - internet traffic of the city with area\_id = 4159

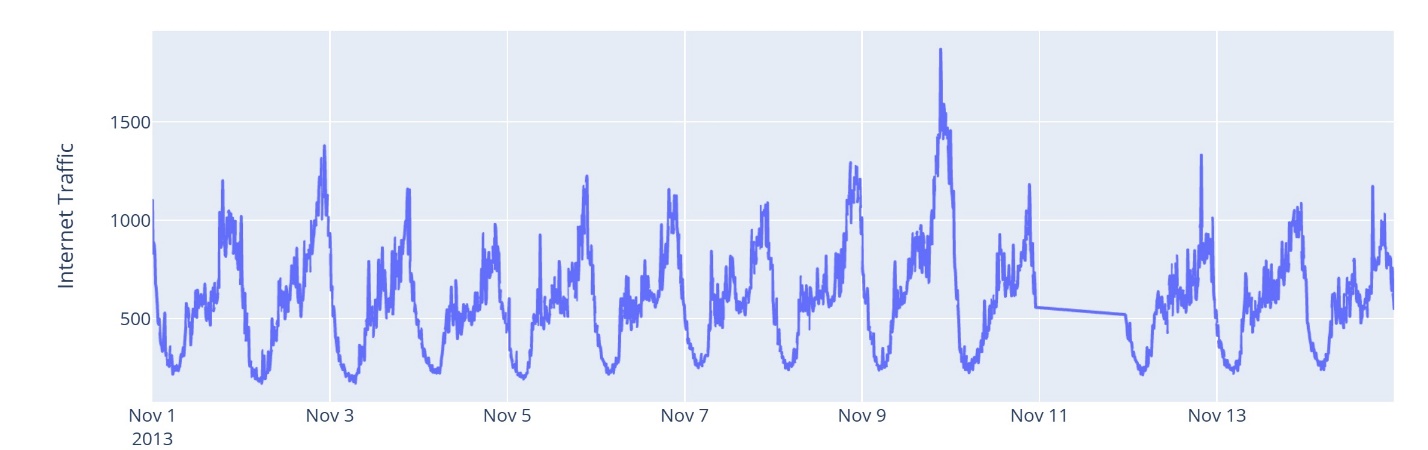


Figure 5- internet traffic of the city with area\_id = 4556.

According to the diagram, all areas peak between 12:00 and 16:00 on all days, and their lowest amount is after 22 o’clock because, at this time, people sleep. The 5161 area has the same pattern on all days during the first two weeks. The total traffic in the 4556 area is half of that for the 5161 area. These differences may relate to the larger population that lives in each city. 4556 ID has its peak consumption on Nov 10. Also, it keeps the pattern of internet traffic the same during the two weeks while the area with 4159 ID fluctuates significantly. This city experiences the lowest rate of internet traffic in the first four days of the month. Additionally, all the signals are stationary because the test statistic is less than the critical value, so we can say that the time series is stationary. Take area 5161 as an example; Test static is equal to -19.341, which is less than its critical value, equal to -3.42. So we can realize that the signal is stationary.

**Task 2:**

There are many machine learning and deep learning approaches to predicting time series, so I tried three of them, from classic machine learning to advanced top-tier approaches.

1. **The first and most straightforward approach is linear regression:**

Linear regression is widely used in practice and adapts naturally to complex forecasting tasks. Hence, linear regression will be the first model used in forecasting. linear regression algorithm learns how to make a weighted sum from its input features.

There are two kinds of features unique to time series[1]

* **Time-step Features:** features that can be derived directly from the time index. the most basic is the time dummy, which counts steps from beginning to end.
* **Lag Features:** Shifting the observations of the target series such that they appear to have occurred later produces lag features.

To determine the shift value, we can find the autocorrelation between the main signal and the lag ones. I have calculated it for lag 1 to 20, and the chart is plotted by utilizing the “statsmodels” library. Correlation indicates the strength and direction of a linear relationship between two variables. Autocorrelation(ACF) is the correlation between the values of a time series in successive periods[2].

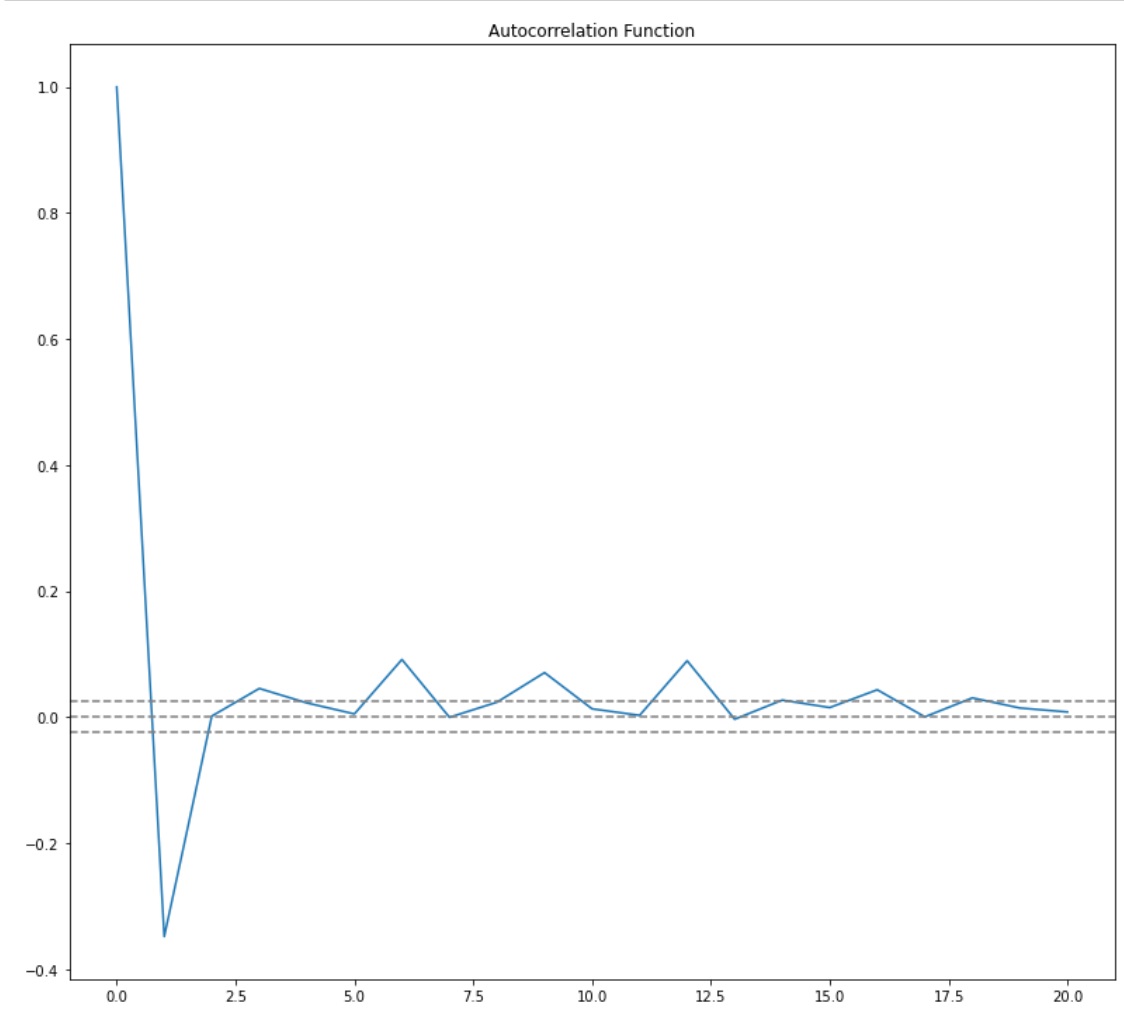


Figure 6- the Autocorrelation function (ACR) for the city with maximum total traffic for lag one to twenty.

We chose shift 3 because it was the first time ACF crossed my upper threshold. I resampled the traffic signal to 10-minute intervals and then gave them to a linear regression algorithm using sklearn library. I also want to add that I trained this simple model on my laptop which its characteristic had mentioned in task 1. In the evaluation part, I will compare the mean absolute error (MAE) and mean absolute percentage error (MAPE) with other algorithms. The following figure represents the real and predicted internet traffic signal by linear regression from December 16 to 22.

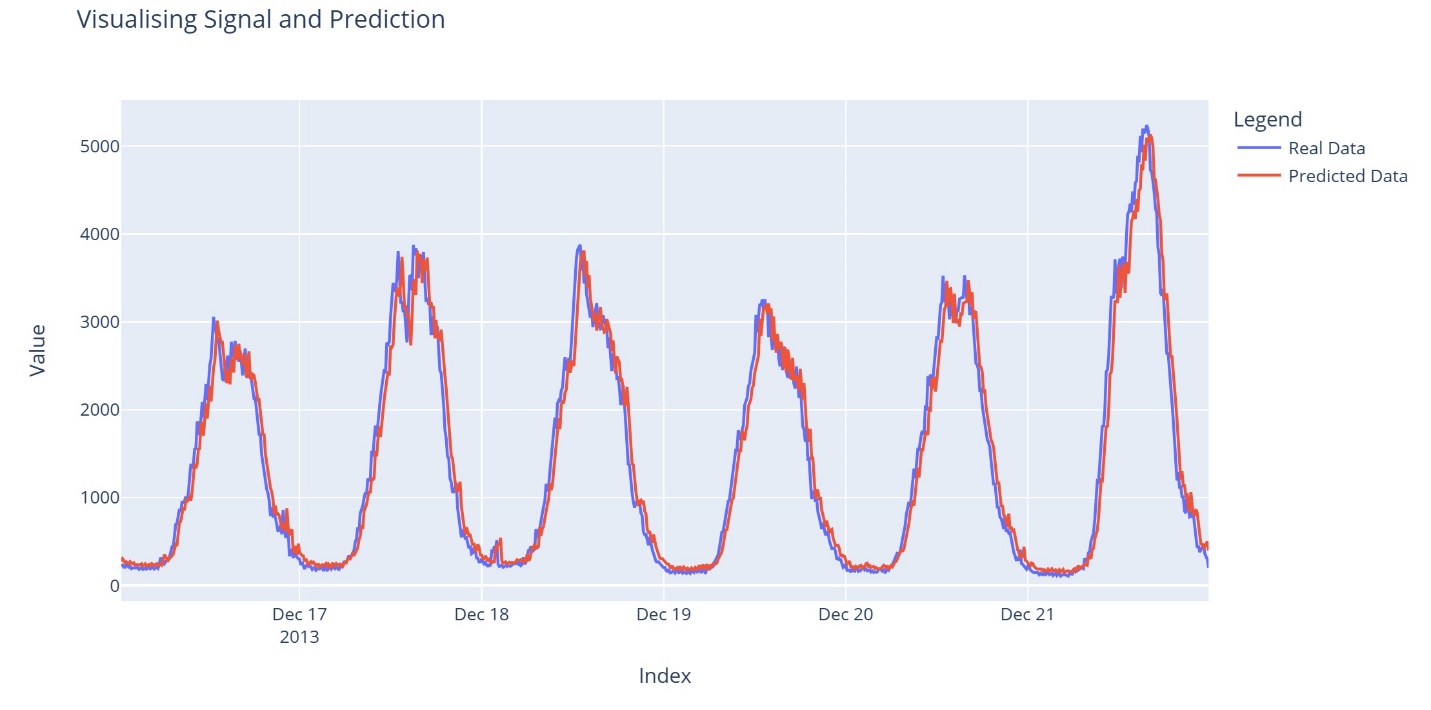


Figure 7-a

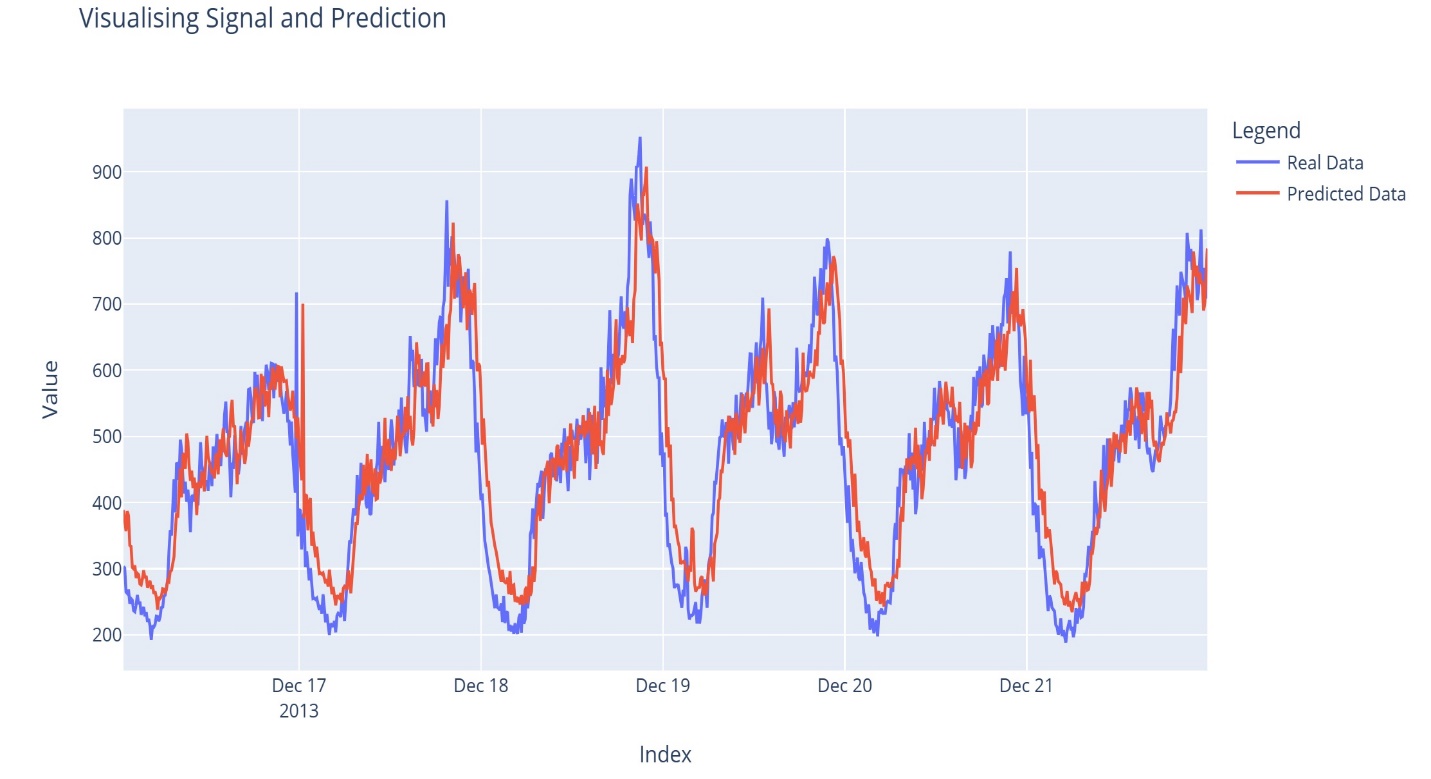


Figure 7-b

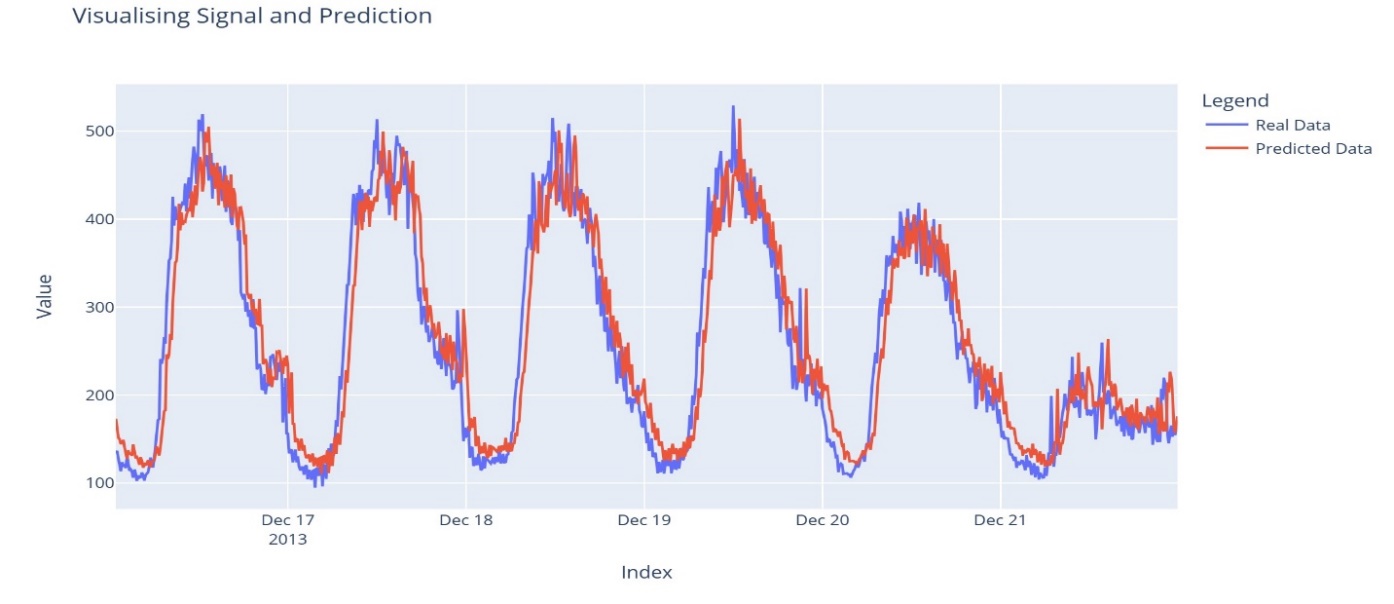


Figure 8-c – the real and predicted internet traffic depicted in a,b, and c are related to areas where its id is 5161, 4556, and 4159, respectively.

1. **Predicting the traffic signal with Chronos**:

Chronos is a simple yet effective framework for pre-trained probabilistic time series models[3]. Chronos tokenizes time series values using scaling and quantization into a fixed vocabulary and trains existing transformer-based language model architectures on these tokenized time series via the cross-entropy loss. I utilize a pre-trained Chronos model based on the T5 family, complemented by a synthetic dataset that we generated via Gaussian processes to improve generalization. As for training our target date traffic, we need a powerful GPU. I use Google Colab to train this part of my code with an A100 GPU. In this part, I resample the traffic signal every 3 hours because when I analyzed the signal in the frequency domain after getting FFT from the signal, I found that the main features of the signal are repeated every 3 hours[4].

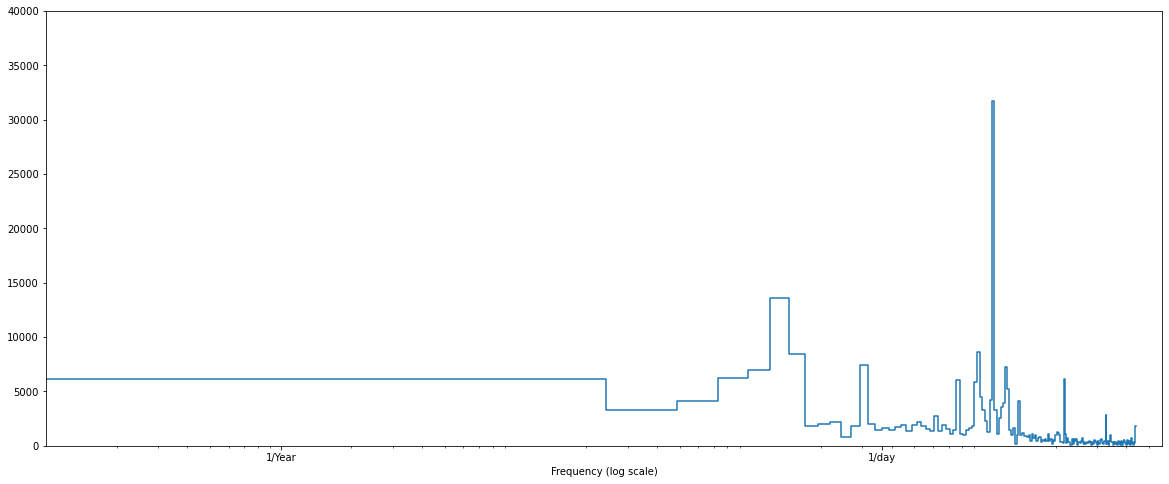


Figure 8-FFT of the signal to determine which frequencies are important by extracting features.

Then, due to the GPU memory limitation, I used 300 past steps from my series to predict 48 next steps. Each step equals 3 hours, so by 48 steps, we can find traffic for Dec 16 to 22. I have separated my data into training and test data frames. The training data frame contains internet traffic records from the 1st of November to the 15th Dec. To run the code, you should only install the Chronos library by the following command: “pip installs git+https://github.com/amazon-science/chronos-forecasting.git ” and utilize the cuda cores of your GPU. The code file for this section is stored in another Python script named task2.ipynb, while for other methods is stored in task1.ipynb.

The following result for predicting the traffic congestion for the target date is shown below.

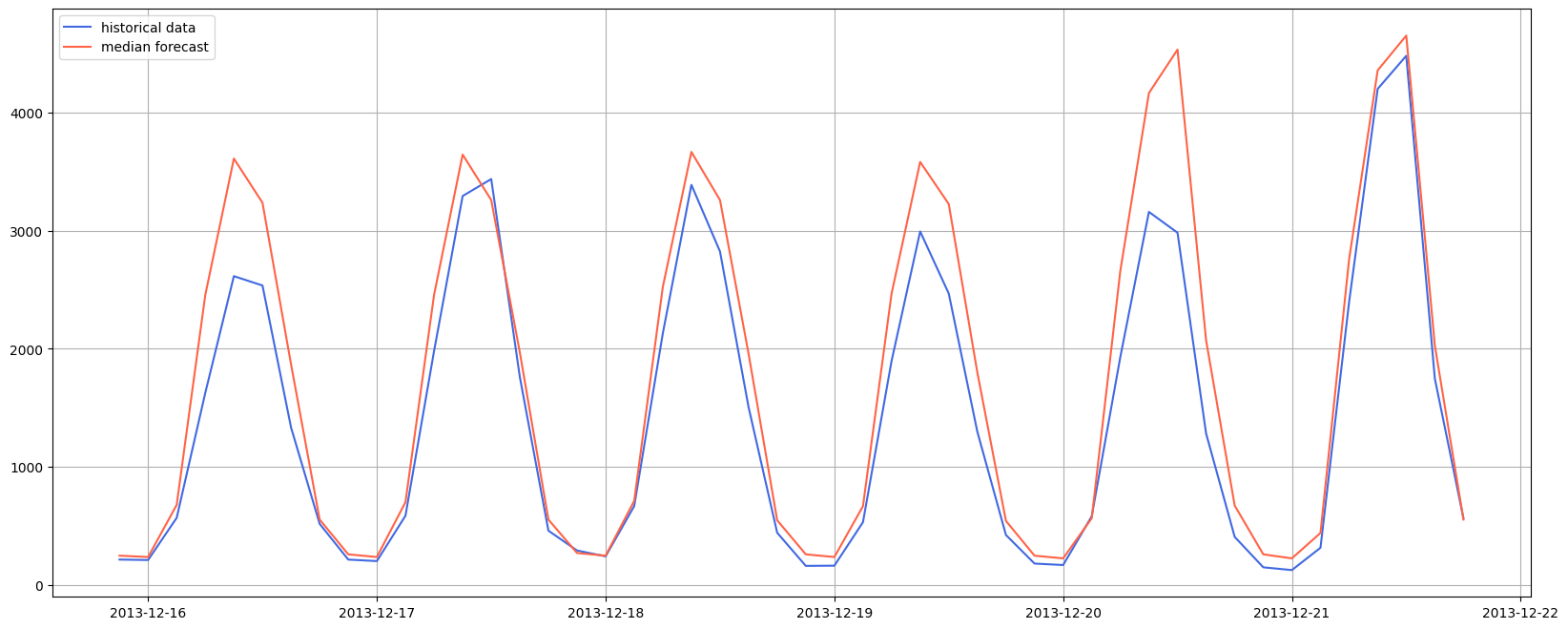


Figure 9-a

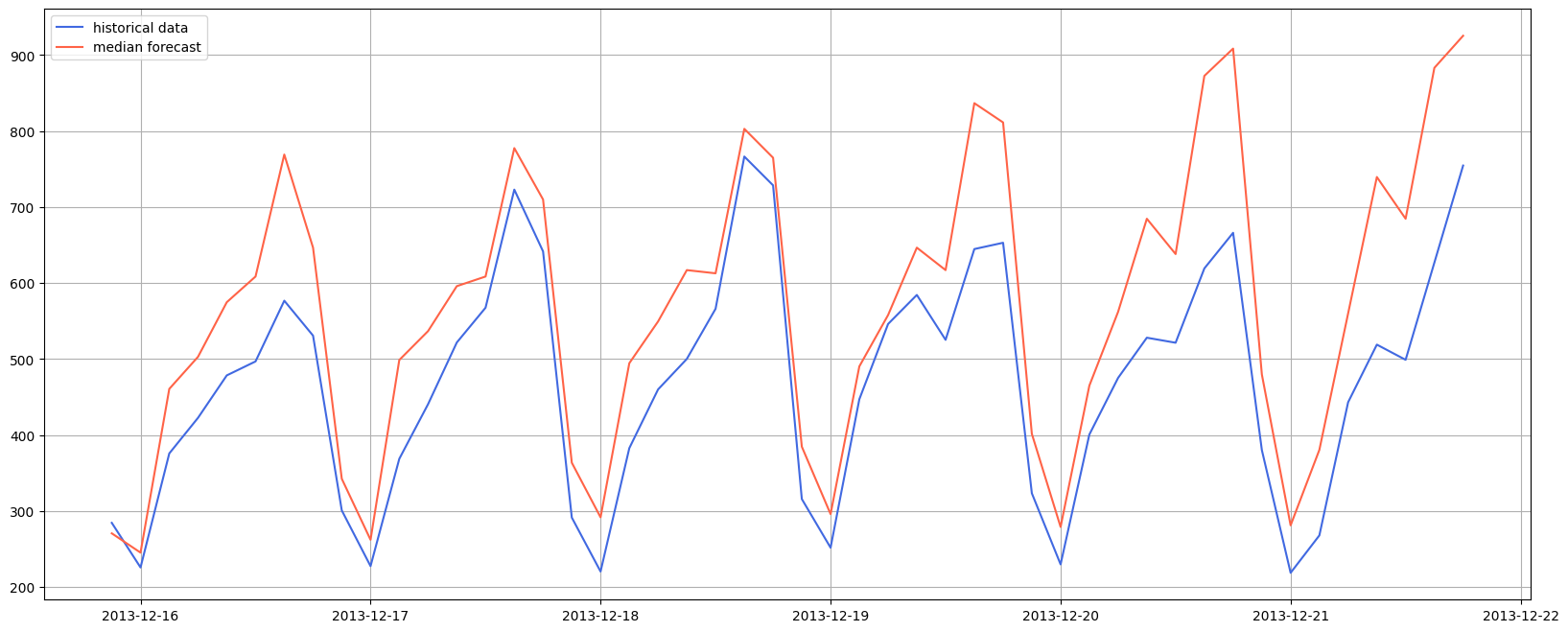


Figure 9-b

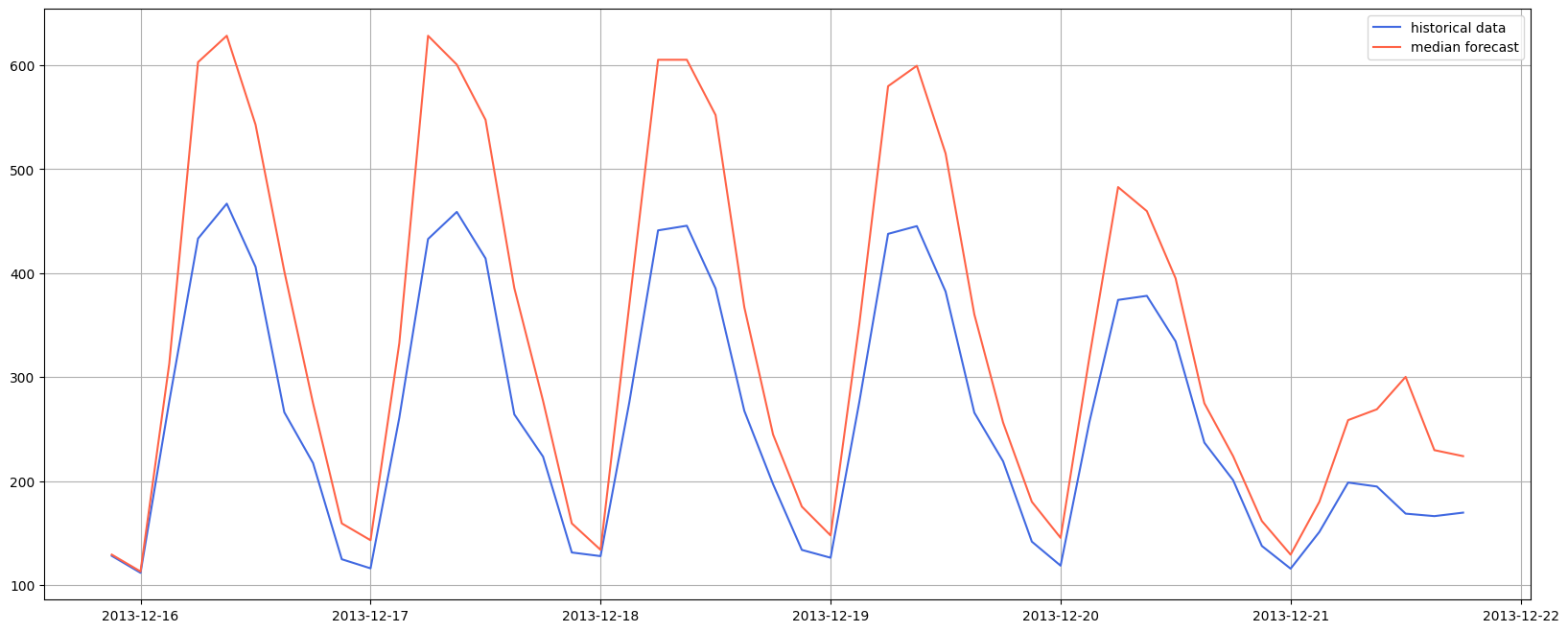


Figure 9-C – Figure 9 shows the result of predicted traffic for cities. Figure a,b, and c correspond to the city with areas 5161,4556, and 4159 respectively.

It is worth noting that this algorithm calculates three quantiles for prediction, which helps the internet provider make more obvious decisions. I have just plotted the second quantile to simplify the depiction.

In the future, I think it is valuable to fine-tune this model for our whole dataset. I believe the traffic of each city affects other cities. Not only does it increase the accuracy, but we can also consider other important features that significantly affect internet traffic, such as weather conditions.

1. **Deep learning approach- utilizing RNN to predict internet traffic for our data**:

I use the autoregressive model for this section. It is a way for the model to make one prediction at a time and feed the output back to the model[4].

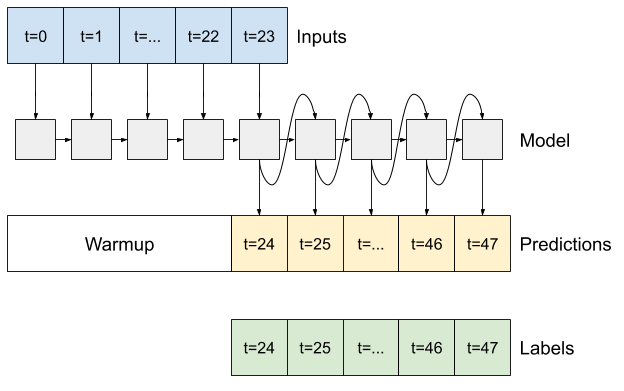


Figure 10 – figure shows that each model's output can be fed back into itself at each step and predictions can be made conditioned on the previous one.

As I mentioned in the previous method, I downsampled my data to every 3 hours. Then, I decided to train my model with 100 steps of the past and predict 48 steps of the future, which corresponds to 16 Dec to 22. The model is trained with my laptop, and its hardware characteristics are mentioned in the Task 1 section. Also, the code is available in Task1.ipynb.

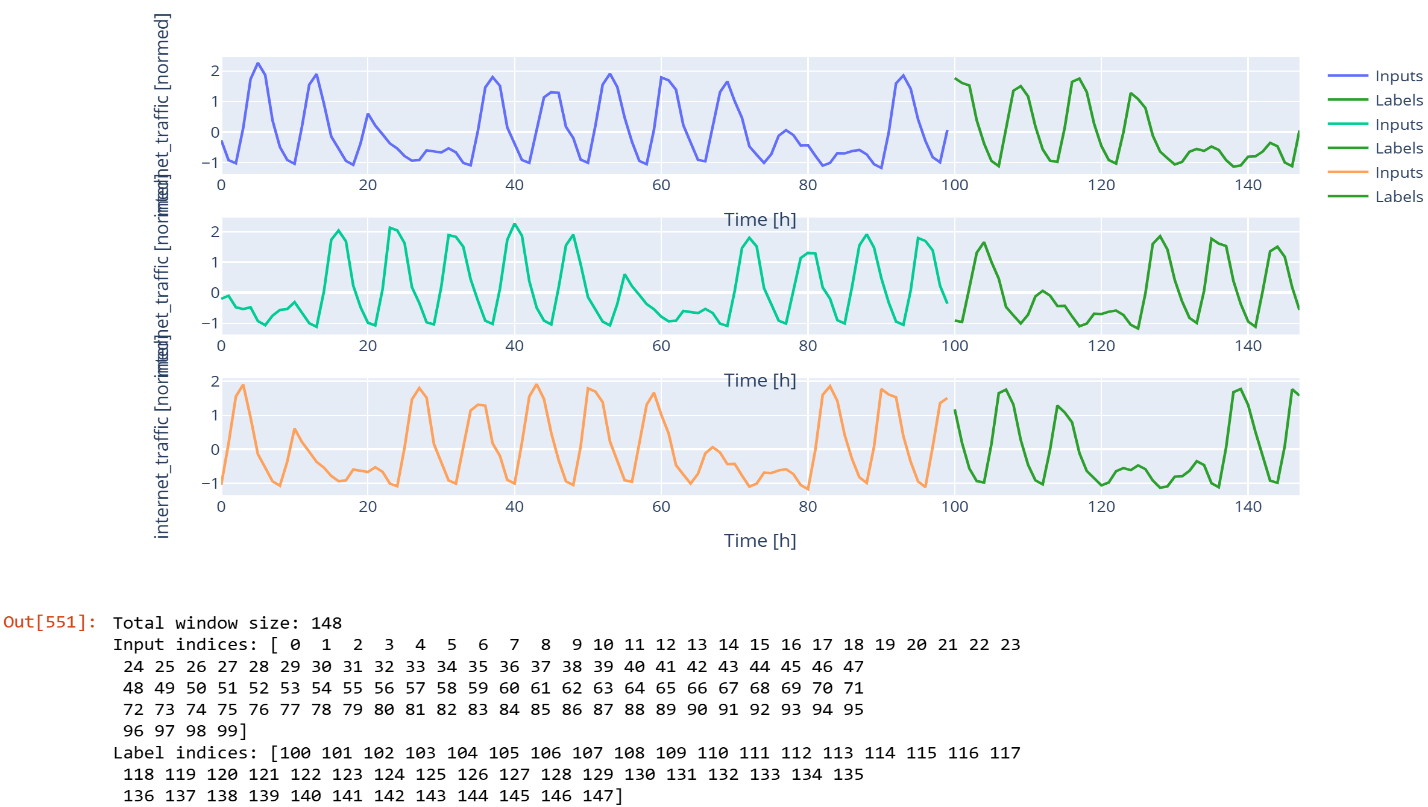


Figure 11- the window as an input to the model

The training process took approximately 10 minutes, which is reported in the last cell of Task1.py, 636.7965521812439 s.

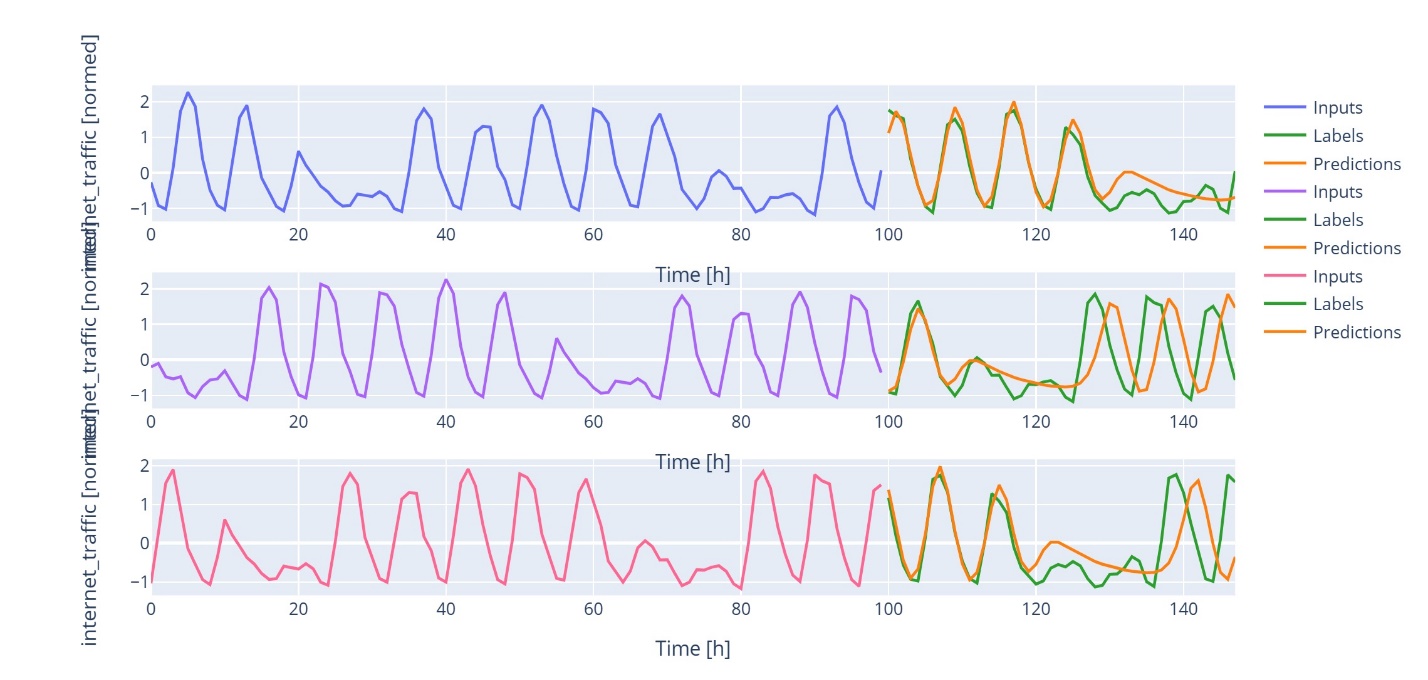


Figure 12-The result for predicting internet traffic for the city with id equal to 4159.

Result and Evaluation:

I like to build a table and compare the MAE and MAPE of each algorithm together

Table 1 – comparing the MAE and MAPE of the proposed algorithms.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Area\_id | Mean absolute error (MAE) for LR | Mean Absolute Percentage Error (MAPE) for LR | MAE for Chronos | MAPE for Chronos | MAE for LSTM | MAPE for LSTM |
| 5161(area with maximum traffic) | 138.2526 | 0.140 | 315.210 | 26.672 |  |  |
| 4556 | 52.90 | 0.1308 | 99.6458 | 21.57 |  |  |
| 4159 | 31.1273 | 0.1351 | 79.8089 | 28.2965 | 0.4113 | 136.1895 |

According to MAPE criteria, linear regression has the best performance. The time needed to run and learn the training dataset is less than one minute. I hadn’t down-sampled the dataset for this method, while in both the Chronos and LSTM, the datasets were resampled by 3-hour intervals. For the LSTM method to increase accuracy, the window's width can be expanded if appropriate hardware is available.

To Conclude, I think the best approach for predicting internet traffic is to deploy the Chronos model or other top-tier transformer-based model to predict the future value of internet traffic by considering other factors and features that are effective in the internet traffic.

**references**

[1] “GitHub - AmirHB98/TimeSeriesBeginner: Bunch of codes and materials for time series.” Accessed: Apr. 08, 2024. [Online]. Available: https://github.com/AmirHB98/TimeSeriesBeginner

[2] “ARIMA Model for Time Series Forecasting.” Accessed: Apr. 08, 2024. [Online]. Available: https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting

[3] A. F. Ansari *et al.*, “Chronos: Learning the Language of Time Series,” Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.07815

[4] “Time series forecasting  |  TensorFlow Core.” Accessed: Apr. 08, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/structured\_data/time\_series#single\_step\_models