IMDEA Assessment

MohammadErfan Jabbari

May 2023

1 Introduction

In this assessment, we aim to analyze the internet traffic in the city of Milan during November and December 2013. This analysis has been conducted as part of the IMDEA assignment.

The assignment is composed of two main parts. Initially, we are tasked to extract insights and general information about internet traffic in Milan. Following this, we will develop predictive models to forecast the city's internet traffic for the period spanning December 16th to December 22nd.

2 Hardware & Software setups

To do this assignment, I have been using my personal laptop. The config is as below:

- CPU: Intel Core i5 – 10300H (8 Cores @ 2.5GHz)
- Memory: 16GB
- GPU: GeForce GTX 1650Ti Mobile
- OS: Ubuntu 22.04
- Python: 3.8.16
- Pip: 23.1.2

3 Dataset

The dataset used for this assignment was compiled as part of the Telecom Italia Big Data Challenge. This comprehensive dataset encompasses diverse information sources, capturing SMS activity, call activity, and internet activity from the entire city of Milan and the Trentino province. The geographical area is divided into 10,000 equal sections (100x100 cells), with data reported in rows. Each row represents the aforementioned activities within a specific grid square (identified by a 'square_id') during a specific 10-minute time interval. The dataset comprises a text file for each day spanning from November 1, 2013, to December 31, 2013. Additionally, the dataset includes a geojson file describing the geographical location of each grid square, which will be utilized to visualize data on geospatial maps.

Because of the size of dataset (both hourly and 30 minutes resampled), I couldn't upload them on my zip file, you can fin both dataset in my google drive in this link. If you wanted to run the code, you can download and place the datasets in folder 'cleaned_dataset' and run the notebooks.

4 Task 0 (Loading & Cleaning the data!)

The first essential step in the data analysis was downloading and cleaning the dataset. The complete dataset, roughly 20GB in size, was downloaded from the Harvard Dataverse website.

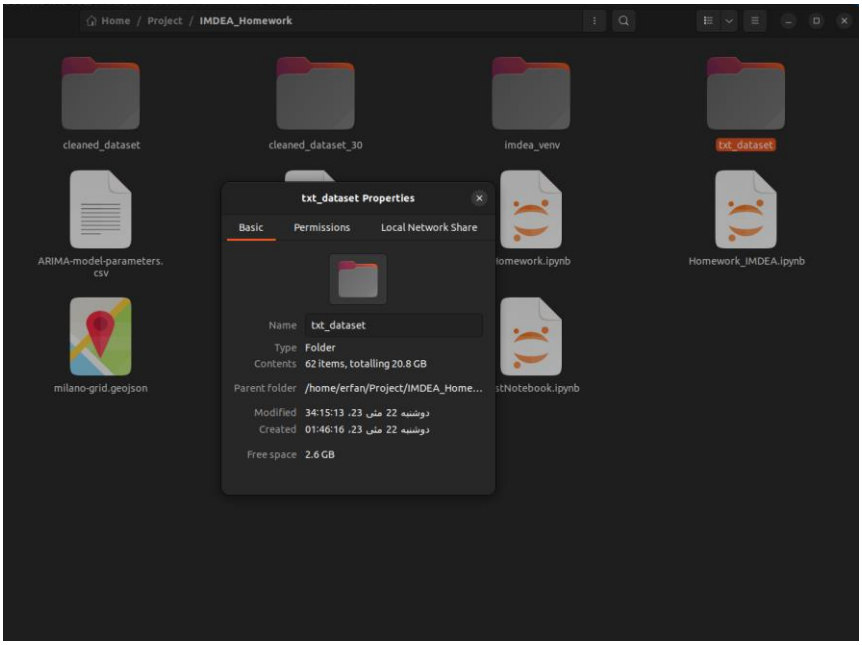

Figure 1

Upon download, the 'DataCleaning.ipynb' file was used to load and clean the dataset. This script was designed to locate a folder named 'txt_dataset' in the same folder with code, holding all the downloaded dataset files formatted like ('./txt_dataset/sms-call-internet-mi-2013-11-01.txt'). It then generates a list of available files in the txt_dataset folder, iterating over each file to load the data into a Pandas DataFrame. Only the 'square_id', 'time_interval', and 'internet_traffic' columns were kept for the analysis.

The 'time_interval' data, originally in timestamp format, was then converted to datetime format. To align with our analytical needs, the data was resampled from a 10-minute interval to an hourly interval. In this process, every six rows with the same 'square_id' were combined into one row. The resultant 'internet_traffic' field was the sum of the earlier values, and the 'time_interval' field signified the commencement of each one-hour interval.

There is also some code provided in 'DataCleaning.ipynb' which is commented, that lets you read the main complete dataset and generate the cleaned dataset win half-an-hour time intervals. To be clear, for the current report we used the hourly cleaned data. This is a picture of the cleaned data in hourly periods:

	square_id	time_interval	internet_traffic
0	1	2013-10-31 23:00:00	57.799009
1	1	2013-11-01 00:00:00	44.046899
2	1	2013-11-01 01:00:00	41.207149
3	1	2013-11-01 02:00:00	33.022070
4	1	2013-11-01 03:00:00	31.376930

Figure 2

Then In the main file that is ‘Homework_IMDEA.ipynb’ we have a section called ‘Load numeric data from dataset’, that we load all 60 files and concat them together into a single dataframe called data. Then by using the day-name method of the pandas library, we added a column to the data dataframe that showed the weekday name of each row.

A view of the data dataframe in ‘Homework_IMDEA.ipynb’ is like this:

	square_id	time_interval	internet_traffic	day_of_week
0	1	2013-10-31 23:00:00	57.799009	Thursday
1	1	2013-11-01 00:00:00	44.046899	Friday
2	1	2013-11-01 01:00:00	41.207149	Friday
3	1	2013-11-01 02:00:00	33.022070	Friday
4	1	2013-11-01 03:00:00	31.376930	Friday

Figure 3

In the following part we will use this data dataframe to do analysis, it is our go-to source for data.

The dataset also provided a geojson file that locates the geographical location of each square_id. In the loading section of ‘Homework_IMDEA.ipynb’ we also loaded this geojson file and plotted the actual squares on the map of the city of Milan. The output can be seen at the following image:

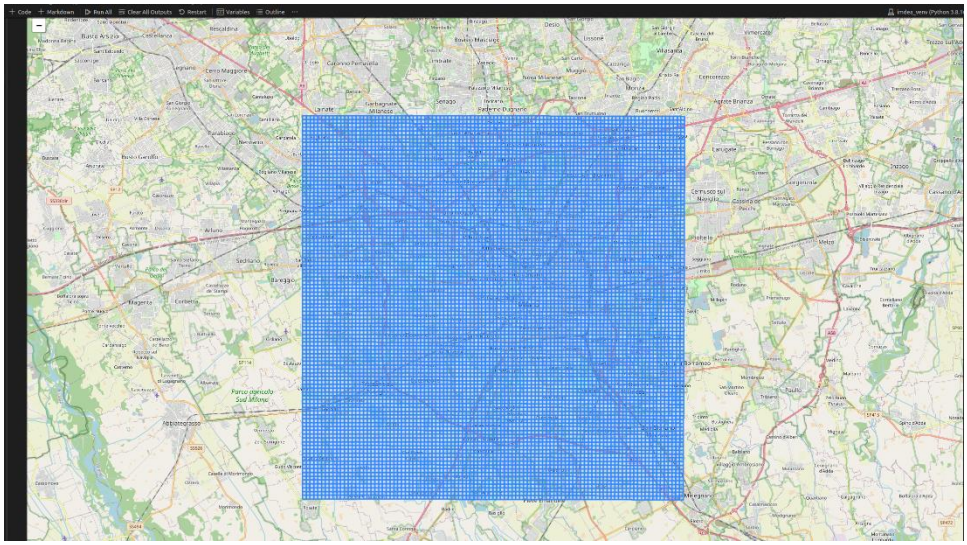

Figure 4

5 Task 1

For task 1 we were asked to do some simple data analysis on the dataset and provide any further insight we could. The code for this part can be found at 'Homework_IMDEA.ipynb'. The file is divided into two main parts for task 1 and task 2. In this section we are focused on task 1.

5.1 Provide PDF for total data

In this part we first were asked to plot the PDF for the total data of each 10,000 squares in the city. The code can be found in section "1.1" of the 'Homework_IMDEA.ipynb' file. We first grouped the data dataframe with respect to 'square_id' for the total of 2 months. So, we have a data frame with 10,000 rows that each row shows the total internet_traffic of each square for the whole 2-month period. Then we wanted to plot the PDF. For this purpose, we had two options first was the hist function of matplotlib and the second was the gaussian_kde function.

5.1.1 Histogram Analysis

The histogram is a common graphical representation of a statistical distribution of numerical data, where the range of output values is divided into small intervals (bins), and the data is distributed among these bins. The bins are usually specified as consecutive, non-overlapping intervals of a variable. So, the main question in this method is how to choose the K optimally. There are two famous methods to choose the bins for histogram that are Sturge's rule, Freedman-Diaconis rule.

Sturge's rule is a simple rule on deciding the number of classes or bins for discretizing the continuous data in a histogram. It suggests that the number of bins (k) can be calculated as follows:

$$\mathbf{k = 1 + \log_2(n)}$$

where n is the number of observations (size of the data). This rule is simple and easy to compute, but it might not be the best choice for a large dataset or a dataset that does not follow a normal distribution.

Applying Sturge's rule to our data, we calculated the suggested k is 14 and then we generate the histogram as shown in the notebook. The plot shows the frequency distribution of the total internet traffic across the different square_ids.

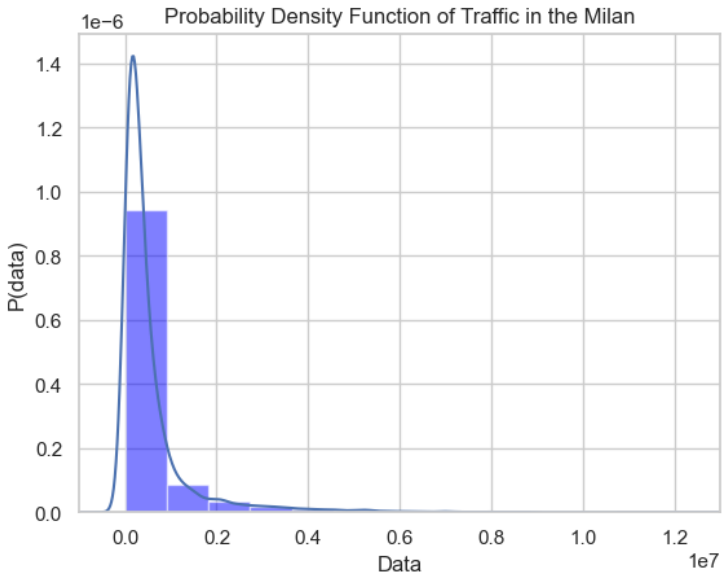

Figure 5

Unlike Sturge's rule, the Freedman-Diaconis rule considers the variability in the data (IQR). So, it's more reliable for a wider range of datasets, particularly for skewed distributions, and as you can see from the plot of Sturge's rule that our data is highly skewed and it's visually testable.

bin_width = 2 * (IQR) / (n ** (1/3))

k = (data_range) / (bin_width)

By applying the Freedman-Diaconis rule, we calculated the k to be 298 then we generate the histogram and observe how it compares to the one produced using Sturge's rule. The distribution should be similar, but the number of bins may vary due to the difference in the binning rules.

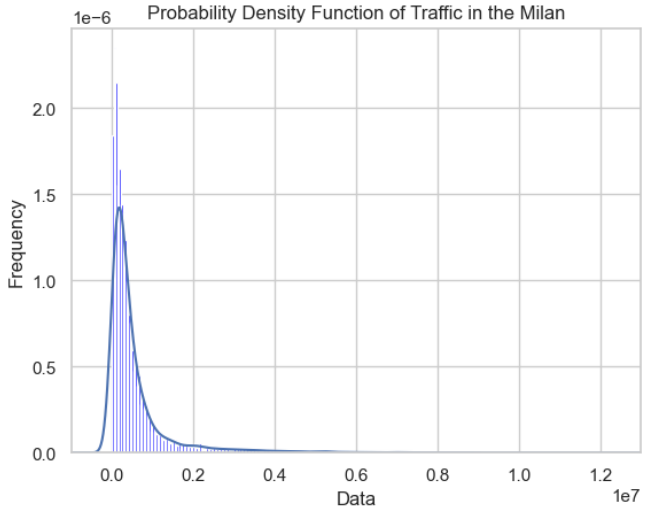

Figure 6

For continuous data, both rules can provide a good initial estimate for the number of bins. Freedman-Diaconis is preferred when dealing with skewed data or data with outliers, which is our case, as it considers the IQR of the data. So. Also, both rules give us a good estimation of the PDF of the internet traffic for the city of Milan for the period of 2 months, but the Freedman-Diaconis has captures more information from the data and thus is our preferred rule to be used with histogram method.

5.1.2 Gaussian KDE method

We can also plot the PDF of our data by using gaussian_kde method from SciPy library of Python. The gaussian_kde function estimates the Probability Density Function (PDF) of a random variable by using a Gaussian Kernel. It uses kernel density estimation, a non-parametric way to estimate the probability density function of a random variable.

The Gaussian kernel density estimator is defined as:

$$K(x) = 1/(\sqrt{2\pi}) * \exp(-x^2/2)$$

This is a smoother version of the histogram. Instead of calculating the number of points in each bin, it smoothes these points across a Gaussian distribution. This method doesn't require you to specify bins. Instead, it uses the bandwidth as the smoothing parameter, which controls the size of the neighborhoods around each data point. The Gaussian Kernel Density Estimator then fits a Gaussian distribution to each point in your dataset, and the PDF is the sum of these distributions.

Here we have also used the gaussian_kde function to plot the PDF function for us.

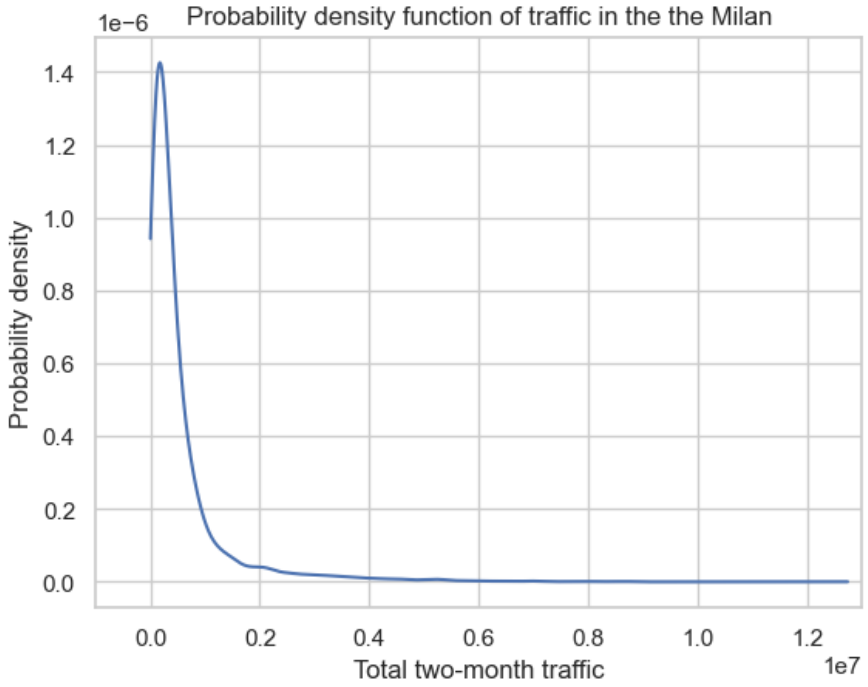

Figure 7

As you can see all three methods are pretty similar to each other most of the hourly traffic data is concentrate data below $2*10^7$ unit in each square. To further analyze this data let's look at this plot:

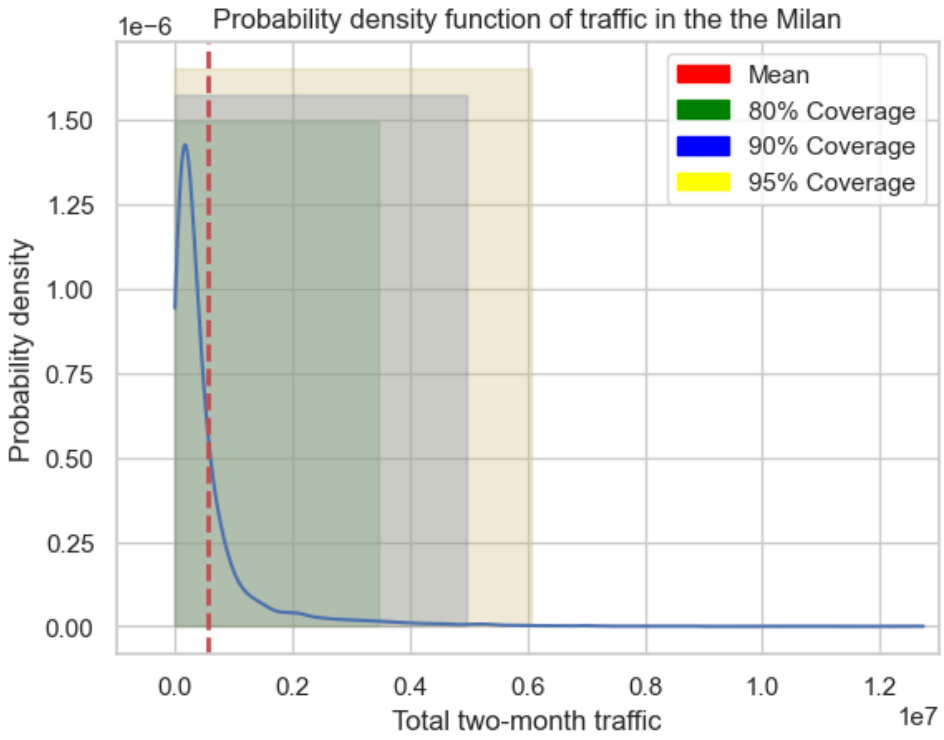

Figure 8

The presented plot clearly illustrates the distribution of internet traffic across different cells in the city of Milan over the period of two months. The mean traffic per cell is approximately 55,5289 units. Additionally, we found that 80% of the cells generated equal to or less than 34,700 units of internet traffic. This extends to 90% of cells accounting for equal to or less than 49,500 units and 95% of cells having an equal or lesser load of 60,000 units of internet traffic during the study period.

This data provides invaluable insights, particularly for administrative bodies, Internet Service Providers (ISPs), and Information and Communication Technology (ICT) operators, as it allows them to predict and manage network demand effectively. Understanding how internet traffic is distributed enables these entities to gauge the network capacity required to handle a certain percentage of requests.

For example, accommodating up to 80% of network requests would require a capacity to handle up to 34,700 units of traffic for the duration of 2 months. However, if the aim is to handle up to 95% of the requests, the capacity would need to be expanded to manage up to 60,000 units of traffic. Therefore, based on available budget and infrastructure limitations, strategic decisions can be made about whether to focus on maintaining current network capacity or planning for expansion.

Additionally, the analysis can help find geographical areas that may require more network support or are ripe for investment based on high internet traffic. These insights are important for ensuring efficient network management, improving user satisfaction (Quality of Service), and helping informed decision-making related to network expansion and infrastructure development.

Finally, this analysis can also form the basis for future predictive modeling, where trends in internet traffic can be analyzed over time to forecast future network capacity needs. This proactive approach can

ensure that the network infrastructure remains robust and capable of meeting user demands, thereby minimizing network congestion, reducing service disruptions, and enhancing overall network performance.

5.2 Comparison of three areas

In this part we were asked to compare the internet traffic for three areas, 1) area with the highest traffic during the whole two months, 2) area with square_id 4159 and 3) the area with square_id 4556.

So first we had to find the area with the most total internet traffic during the period of two months. This area is the area with square_id 5161 as you can find in the following figure:

So, let us start by looking at the plot of internet traffic for these three areas (5161, 4159, 4556):

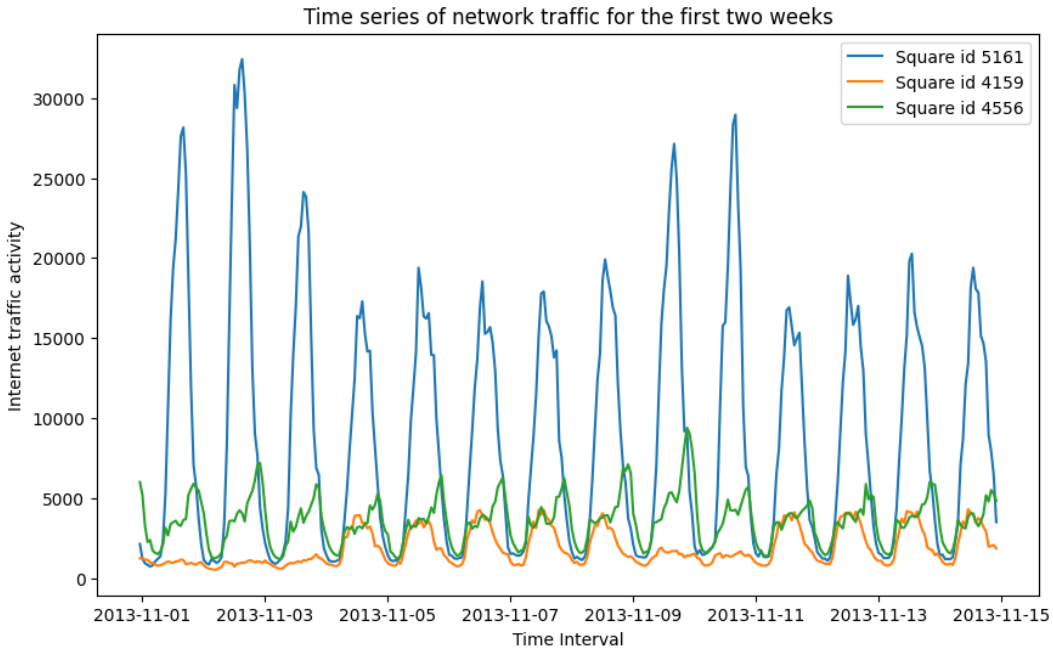

Figure 9

As is evident from the graph, there is a significant difference in the internet traffic load between these areas. A higher volume of internet traffic, such as in the area represented by square_id 5161, could suggest that this area is a business district or a hotspot for tourists, possibly because of historical attractions or shopping stores.

On the other hand, the internet traffic in the area with square_id 4556 peaks later in the day, suggesting a residential region where the residents' internet usage grows at the end of the day. This area might also be a zone such as a park or a riverside hangout spot, where people gather in the evenings with cafes and restaurants.

The area represented by square_id 4159 presents a different pattern with very low internet traffic over the weekends (11/01, 11/02, 11/10, and 11/11) and not much high peaks during workdays, so it may not be a historical area or a famous business zone. This could imply that this area might be a university zone, or a transitional area situated between business hubs and residential areas.

Now let’s pin the areas on the geospatial map and see if our predictions are correct or not?

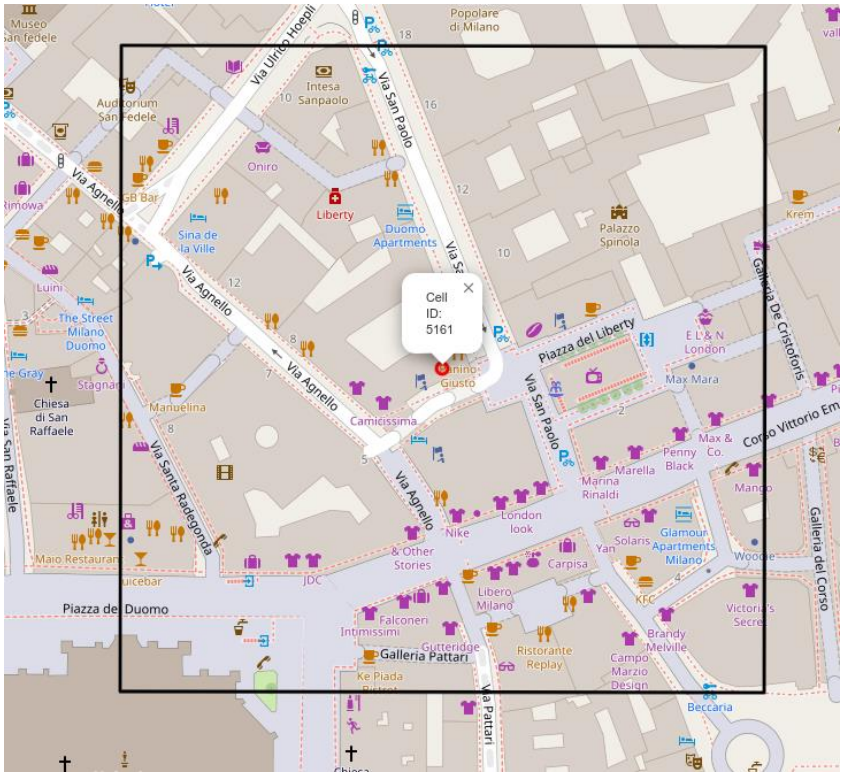

Figure 10

As expected, the area with square_id 5161 is located at the heart of Milan, close to well-known places such as the Palazzo Spinola and the Duomo di Milano. This central location, surrounded by shops, aligns with our initial prediction based on the high volume of internet traffic.

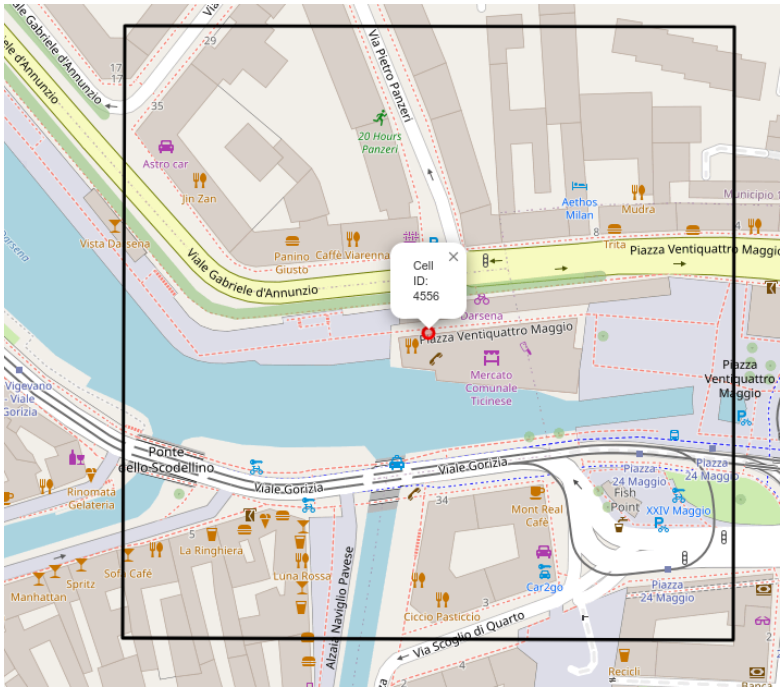

Figure 11

As we expected, this area is located near a well-known waterway known as "Darsena del Naviglio", a prominent nighttime destination in the middle of lots of cafes. Although it's a hotspot, it doesn't match the busy nature of area 5161. Thus, it's not surprising that its internet traffic reaches its peak later in the day and doesn't get as high as the areas characterized by business activities and tourism.

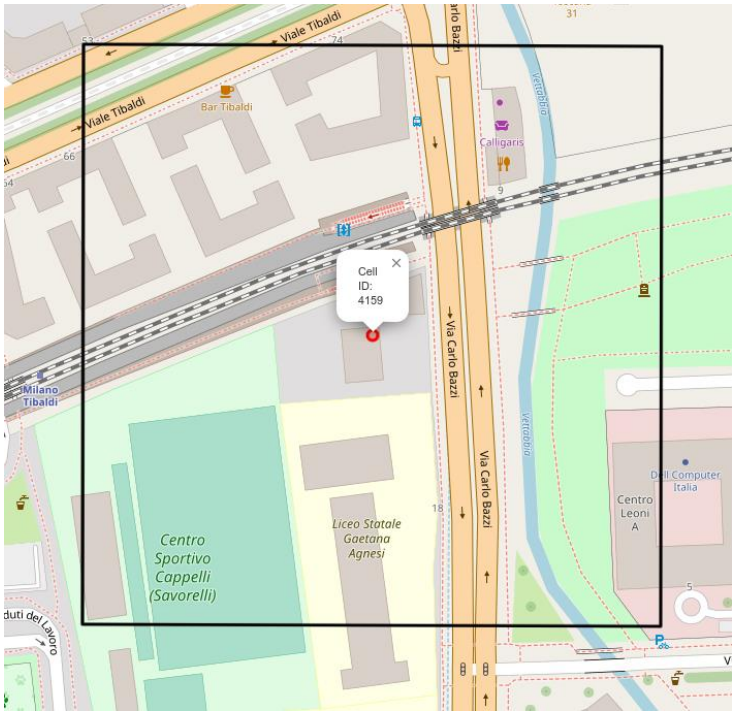

Figure 12

This area is near the Bocconi University and some of the facilities of the university are located inside this area. So, one of our predictions was true, that this area is near a university.

Here is another plot that shows the mean internet traffic for days of a week that was calculated from the first two-week data that we used in this task:

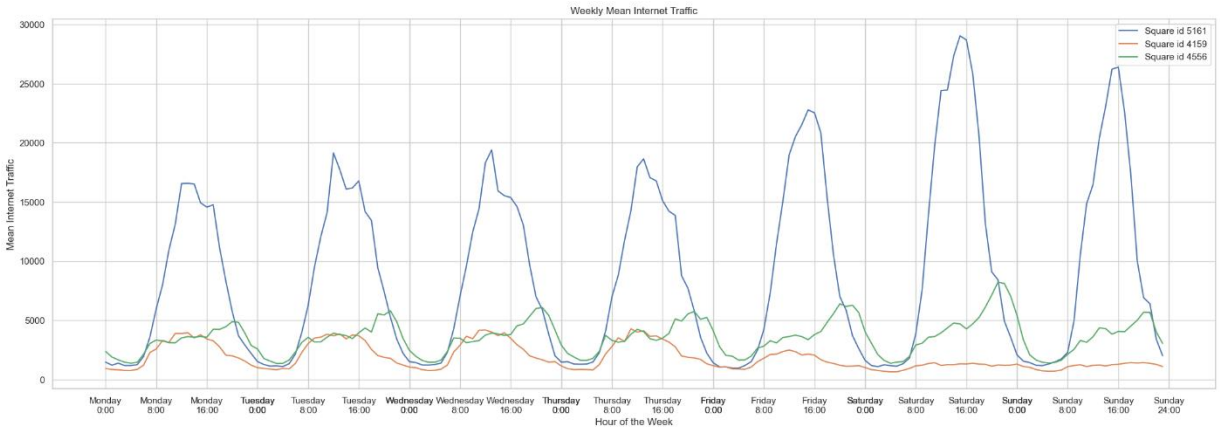

Figure 13

This plot is used to further demonstrate the hypothesis that we proposed above. As you can see the area 5161 usually hits its peak during between 11 AM to 1 PM that is the busiest hours also according to the google (you search the area, google shows you the relative busy hours of each day) that lots of tourists meet the historical hotspots we mentioned above. Also, here because of the higher resolution of the data we can see the peak for area 4556 happens between 8PM to 12 PM.

5.3 Further Analysis

In this part I will share with you, analysis that I found to be interesting about the dataset, alongside the above analysis that was asked in the homework.

5.3.1 Peak Traffic Analysis

In section 5.2 we only analyzed the data from three areas. Here we want to do analysis on the whole city, so its perspectives are much wider. The following plot shows the probability function that a cell will hit its peak at which hour of day.

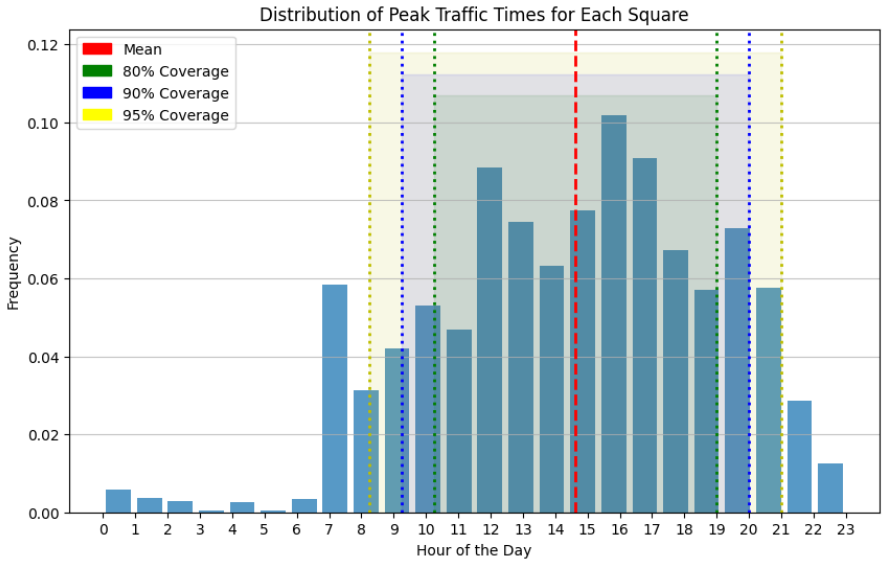

Figure 14

As illustrated in the analysis, it's more probable for an area to experience peak internet traffic around 7 AM rather than 8 or 9 AM, because during that time (7) a lot of people are going to work. Another peak is observed around 2 to 3 hours post-lunchtime (lunch time is usually about 12:30 to 14:30 according to google), which aligns with typical working hours and lifestyle patterns.

Drawing insights from Section 5.1, where we analyzed the Probability Density Function (PDF) for the entire city, combining these insight and data from section 5.1 can be interesting. Our data suggests that for any given cell, there is a high likelihood of peak internet traffic occurring between 11 AM to 7 PM.

By strategically leveraging these insights, network administrators can optimize their resources more efficiently. Instead of planning infrastructure (like Virtual Network Functions including routers, switches, or firewalls) to support peak traffic levels for all day long, they can focus on ensuring high-performance during the predicted peak hours of 11 AM to 7 PM. This approach could potentially manage the cell's traffic effectively for about 80 percent of the time.

Further tailoring of this strategy can be done based on the specific requirements of each area. For instance, a service provider may aim for 99 percent coverage in business districts to maintain a strong reputation among corporate clients, while a lower coverage of 80 percent might be deemed acceptable for residential areas. Our analysis enables providers to plan and allocate resources accurately according to these varying demands.

5.3.2 Correlation Analysis

The correlation analysis can give us insight into which areas internet traffic movement are like each other. For example, if areas with square_id 100 and 200 have a high correlation, it means that we predict when internet traffic of area 100 will increase, also, internet traffic of area 200 will increase.

We can have multiple kinds of correlation analysis between areas. Correlation between areas at the same time and lagged correlation that is the correlation between internet traffic of 100 (example) at current time with internet traffic of area 200 for 1 or 2 hours ago.
The first correlation helps us to predict which areas' internet traffic will rise and fall with each other (or act opposite in case of negative correlation). An example of usage for lagged correlation can be that we can see and predict for example which areas between business districts and residential areas experience a rise in internet traffic after businesses are close in the afternoon and we can predict the paths that most people take home, so we can plant our infrastructure near those popular paths and provide better service.

Let us start with the simple correlation analysis, the following is the plot of correlation matrix between areas:

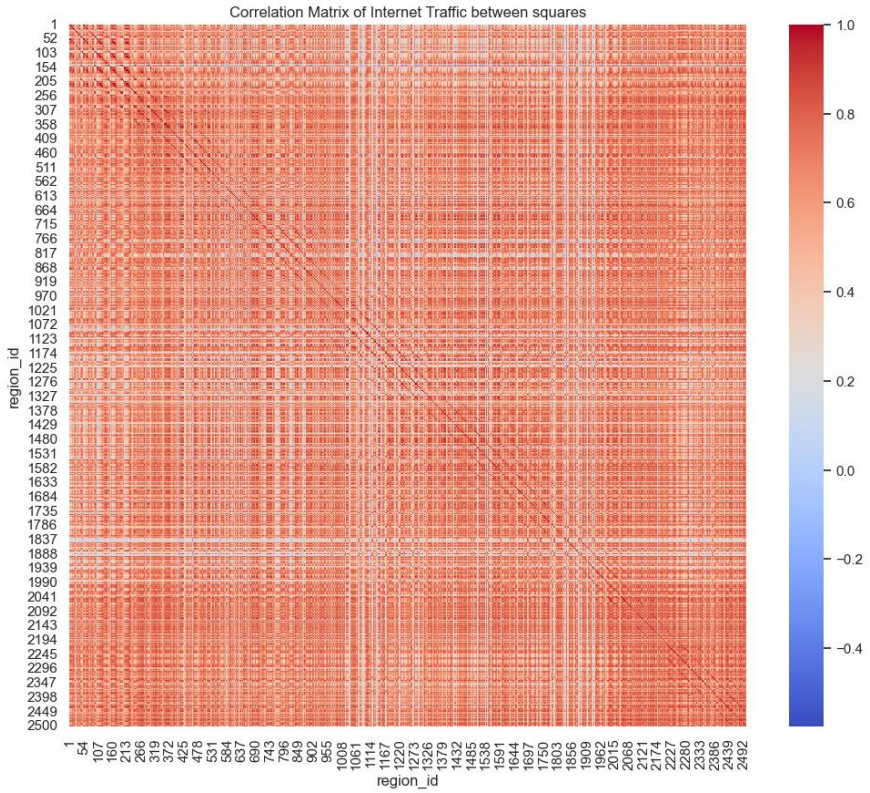

Figure 15

As you can see on the figure, the axis are regions, because we gathered every 4 squares into a region because we wanted the bigger picture for whole city and 100,000 squares didn't make up that much of information. So, we grouped each 4 squares into a single region (squares 1,2,101,102 to region 1). From the plot we can see that there is no valuable information to receive, just near each other areas are correlated as we expect. But what about grouping each 100 (10*10) squares into a region? The correlation matrix for that is like this:

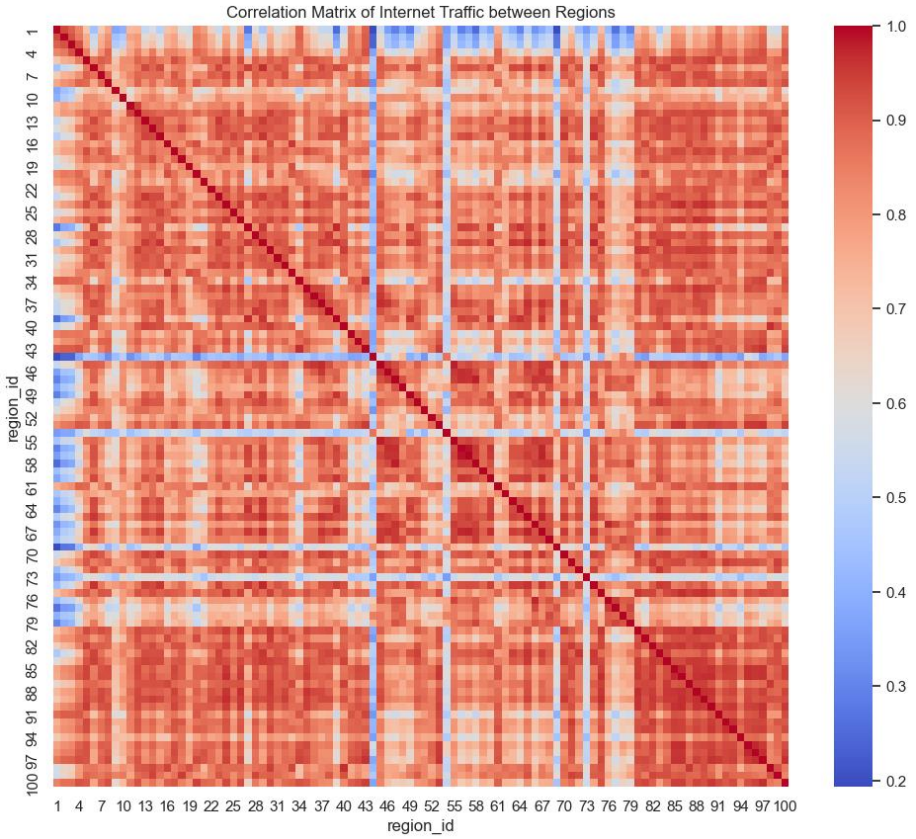

Figure 16

As we can see, there is a little bit more information here, we can see that areas are grouped into large squares that have good correlation (square that contains regions (79,100) on both axis). Again, not so much to digest here. Then I wanted to see if there was any correlation between traffic of each area and the traffic of other areas in the previous time step. The correlation matrix for this one looks like this:

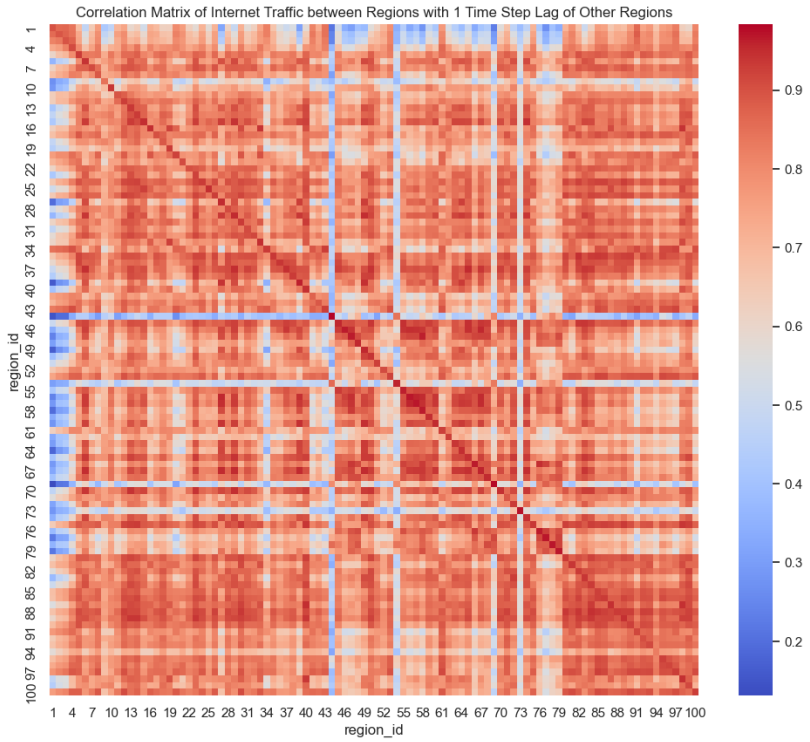

Figure 17

As we can see there is no significant difference between this correlation matrix and the previous one. So, we can say that for each area there is only a good(useful) correlation between traffic of that area and 20 to 50 nearby areas. So, a wise action would be for each area to see the incoming peak in traffic. We monitor 20 to 50 nearby areas alongside the area of interest. The reason I say there is no interesting data here is that what I had in my mind was to be able to see the correlation between traffic of business and touristic areas and the main pathways to residential areas, but I could not extract the information :(

5.3.3 Hotspot Analysis

In this part we are focused on 10 areas with the most traffic during the period of 2 months, that's why we called this one "Hotspot".

So, we first start by finding the areas with the most traffic during the period of 2 months. The list is like this from highest traffic to lower traffic: 5161, 5059, 5259, 5061, 5258, 5159, 6064, 4855, 4856, 5262. As we have found in section 5.2 the area with square_id 5161 is the area with the most traffic.

The following figure shows the box plot for mean internet traffic of these 10 areas for each day of the week:

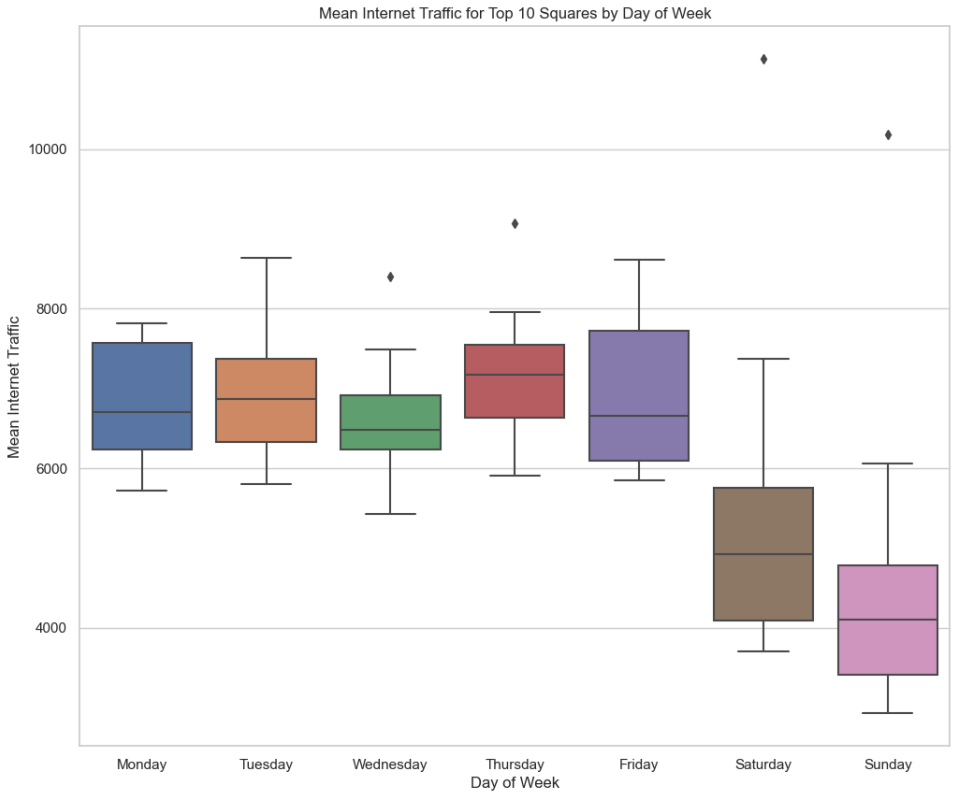

Figure 18

As you can see the traffic for each day is concentrated in an interval of 60,000 traffic units and 80,000 traffic units. There are also outliers for 4 days (Wednesday, Thursday, Saturday, and Sunday) that we predict are the traffic for area 5161 as it's the area with the most traffic. Also, we can see that the mean data of these areas decrease by about 40 to 45 percent during the weekends. Without any further data we can predict that most of these top 10 areas are business districts and there are few touristic places near them.

Now lest get the data for this plot with higher resolution. The following plot shows the swarm plot of mean traffic for each day of the week for each top 10 areas.

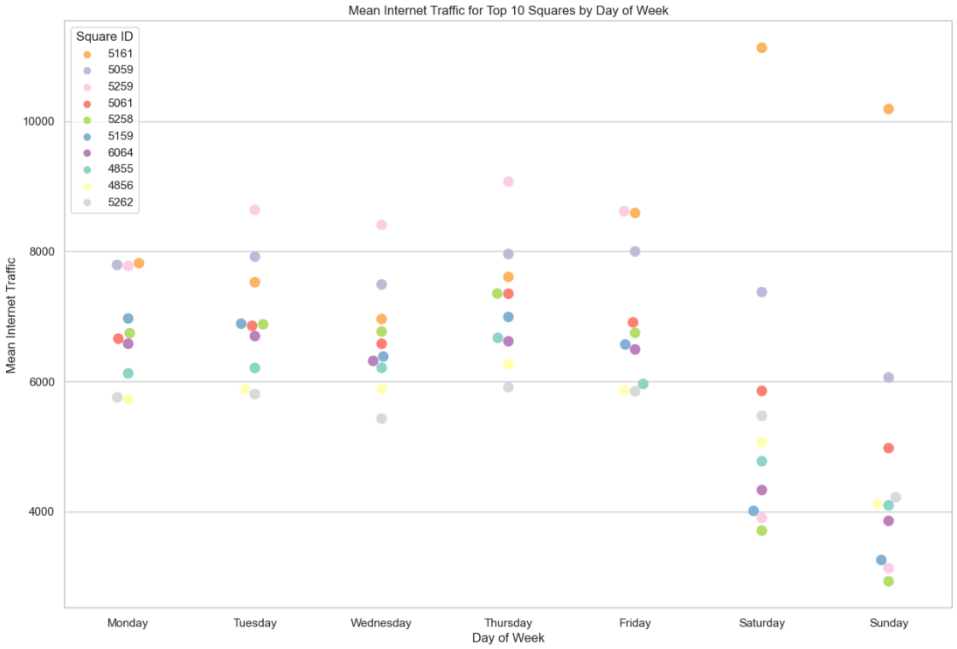

Figure 19

Interesting information here, earlier we predicted that the outlier point in the box plots are points of area 5161 that is the area with highest traffic. But surprisingly, during the workdays those points were associated with area 5259. This area is the area with the highest traffic during workdays and experiences about 50 percent reduction in internet traffic during weekends. That is characteristics of a business district as we suspected. Same goes for all other areas except for 5161 and 5069.

Now let's see these areas on map:

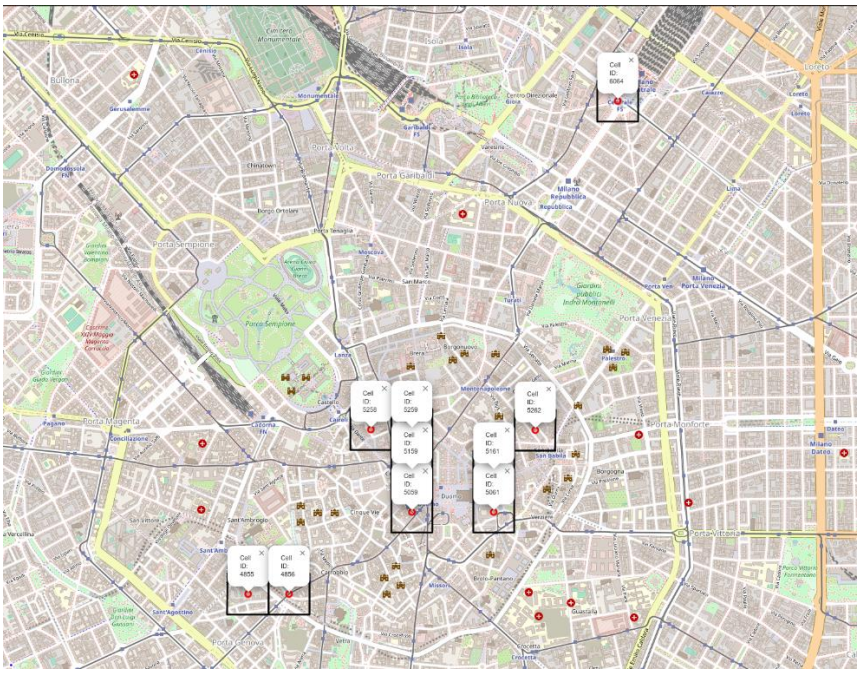

Figure 20

Interesting that 9 of these 10 areas are located relatively close to each other. If we zoom on the area 6064, we can see that this area does not have as many shops and historical landmarks. But if we search for the name of the area that is" Piazza Duca d'Aosta" we find out that this is one of the most populated and busiest areas of Milan where the central stations and business buildings are located.

From this analysis we can get interesting insights that for 9 of these 10 areas (all except 5161) the internet providers can lower the resources dedicated to providing service in those areas and schedule more on 5161 for weekends. That can lower their cost of service by a lot regarding the level of traffic those 9 areas generate during workdays.

6 Task2

In this section, we will develop methods to predict internet traffic for three specific areas: 6151, which had the most internet traffic during the two-month period under consideration, and two other areas with square IDs 4159 and 4556. Our goal is to predict the internet traffic in these areas from December 16th to December 22nd.

6.1 Analyzing the Internet Traffic for Three Areas

For this task, we will employ a method based on the ARIMA model. ARIMA is a famous and widely used method for forecasting time series data like weather forecasting, stock price forecasting and others. However, before we dive into the modeling process, let us get to know ARIMA and its variants.

6.1.1 ARIMA Model Explanation

The ARIMA model is a popular tool for analyzing and forecasting time series data. It is an acronym that stands for AutoRegressive Integrated Moving Average. The AR (Autoregressive) part of ARIMA refers to the use of past values in the time series to predict the future. This component is based on the principle of regression, where one variable is predicted based on its relationship with other variables. In the context of an AR model, the variable of interest is predicted based on its own past values. The basic idea behind an autoregressive model: we are using the variable's own history to predict its future.

In an AR model, we consider 'lags' of the variable. A lag is a fixed amount of passing time; one set of observations in the past is compared with another set of observations in the future. For example, if we wanted to predict today's temperature based on the temperature 3 days ago, we would be using a lag of 3 steps. So, when we say that a variable is 'regressed on its own lagged values', we mean that we are predicting the current value of the variable (for our problem, today's internet traffic) based on its values at some number of time periods in the past.

The I (Integrated) part indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of these act of differencing is to make the time series stationary (we will get to this concept later).

The MA (Moving Average) component of ARIMA models is about predicting based on errors made in past predictions. It's called a 'moving average' model because it uses the average of past errors to improve the prediction for the next time step.

In forecasting, an 'error' is the difference between the actual value and our prediction for a specific time step. For example, if we predicted that yesterday's temperature would be 20 degrees Celsius and it was actually 22 degrees, the error would be 2 degrees.

What the MA part does is it looks at these past errors and treats them as useful information. It assumes that the errors aren't just random noise but are part of a pattern that could help make better future predictions.

When we say the 'regression error is actually a linear combination of error terms', we're saying that the error for any given time step can be represented as a weighted sum of previous errors. 'Linear combination' here means that we're adding together previous errors, but each error is multiplied by a certain weight before it's added.

For example, we might find that yesterday's error gives us useful information for predicting today's temperature, but the error two days ago doesn't. The MA model would give a larger weight to yesterday's error than the one from two days ago.

So, in simpler terms, the MA component of ARIMA models allows us to use a weighted average of past errors to improve our current prediction.

6.1.2 Variations of ARIMA

There are several variations of the ARIMA model, such as SARIMA (Seasonal ARIMA), SARIMAX (Seasonal ARIMA with exogenous variables), and others. These variations offer more flexibility and can be more suitable for specific types of time series data.

SARIMA: In addition to the parameters of ARIMA, SARIMA models also include additional parameters to specifically model the seasonal component of the time series. This makes SARIMA a good choice for time series with clear seasonal patterns.

SARIMAX: This is an extension of SARIMA that also includes external or exogenous variables. It is useful when the time series is believed to be influenced by factors other than past values of the same series.

Next, we'll analyze our data and determine how best to apply these models for accurate traffic prediction. We'll choose the most suitable version of ARIMA, considering the specific characteristics of the internet traffic data for each of the three areas.

6.1.3 Stationarity

Stationarity is a key concept in the context of ARIMA models and most time series prediction models. When we say that a time series is stationary, we mean that its statistical properties do not change over time. This includes things like its mean (average), variance (spread), and autocorrelation (how correlated a variable is with a lagged version of itself). Why does stationarity matter? It matters because if a time series has consistent statistical properties over time, it becomes easier to make reliable predictions. Imagine trying to predict the weather if the climate were wildly inconsistent from year to year – then we wouldn't have weather forecasters every morning!

There are two main components that can make a time series non-stationary: trend and seasonality. A trend exists when there's a long-term increase or decrease in the data. It doesn't have to be linear. On the other hand, seasonality occurs when there are patterns that repeat at regular intervals, like the rise of tourists in the city of Milan during summer every year.

To use an ARIMA model, we need our time series to be stationary. This often means we need to remove any trend or seasonality from the data before proceeding with the model. There are several methods to test for stationarity. One popular method is the Dickey-Fuller Test.

The Dickey-Fuller Test is a statistical test that can help us to find out if our time series is stationary. Without getting too much into the math, the test checks if the data is significantly different from a linear trend. If the test statistic is less than the critical value, we can reject the null hypothesis (that the time series is non-stationary) and say that the series is stationary.

Simply put, Dickey-Fuller Test is a tool that helps us confirm if our data has consistent statistical properties over time. If it does, it becomes more likely that our ARIMA model will make reliable predictions. If not, we might need to take additional steps, such as differencing or transformation, to make our data stationary before we use it in an ARIMA model.

Now we are going to predict stationarily for the three mentioned areas. We start with area 6151 that was the area with the most internet traffic during the period of two months.

We can use a method called 'decompose' on the data of this method to decompose the seasonality, trend and what is left after removing seasonality and trend from data that is called residual. The following figure shows the plot of original data alongside the mean, std also shows the result of Dickey-Fuller Test result for area 5161:

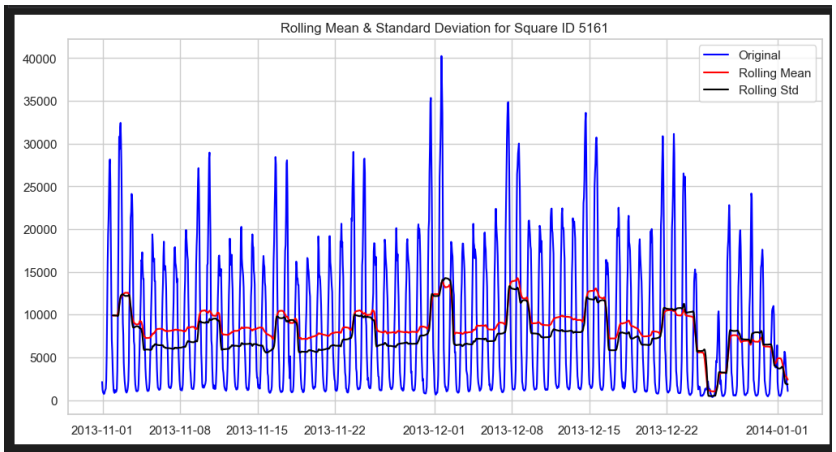

Figure 21

As we can see there the data during workdays and separately during weekends have a steady mean and std. Now let's look at the Dickey-Fuller test result. As we can see the test statistic is less than the 1% critical value and we can say by confidence of more than 99% that our time series is stationary. The following plot is the residual part of the internet traffic data:

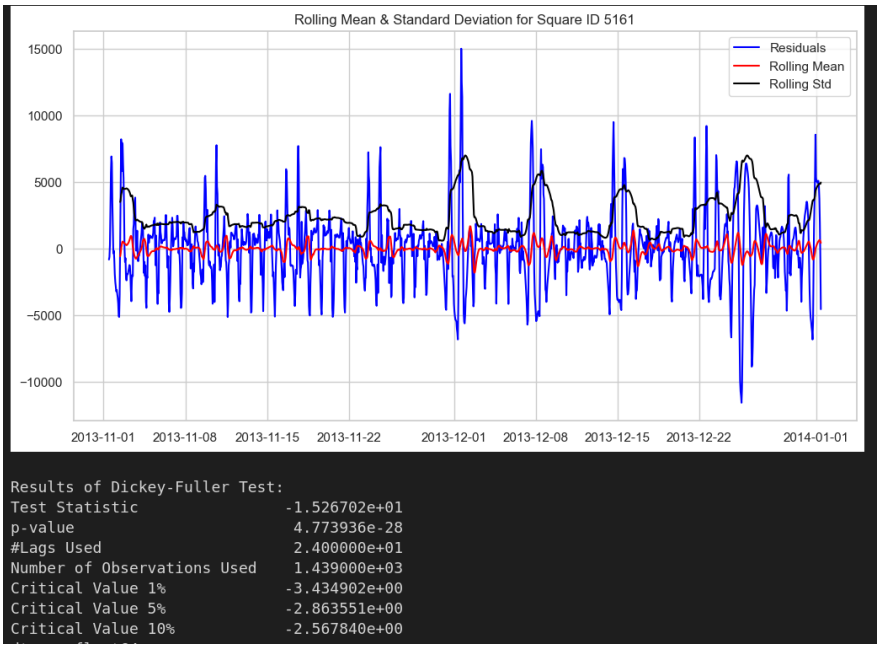

Figure 22

As you can see the residual data has a much more stable mean. If we also check the Dickey-Fuller test the test statistics go from about –3.9 to –15.3, that suggest much more stationarity of the data after we removed the seasonality and trend from it.

Transformation is a method applied to make the data more 'manageable' for analysis. The raw data we have from the real world might not always be suitable for direct analysis. For instance, the data might have a trend or seasonality which could make it non-stationary and unsuitable for certain statistical models like ARIMA, which require the data to be stationary. In such cases, transformations can be used to make the data stationary. The following plot show the Transformations can take on many forms such as log-transformation, differencing, or even a combination of transformations. Here we used the log-transformation to see the difference in the result of our tests. After transformation, we plotted these two figures:

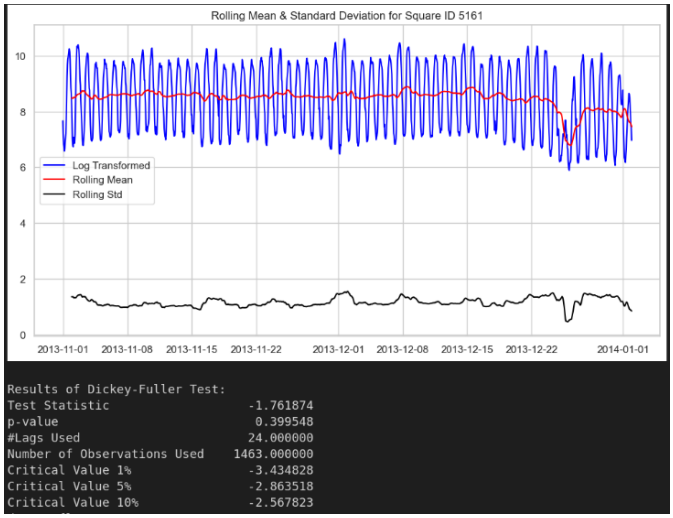

Figure 23

As you can see the test statistics went up after taking log10 and plotting the internet traffic for area 5161. We didn't expect that, but my guess is that it's because of the seasonality of the data. Because if you see the following figure that shows the plot and test results for the residual part of the log data, we can see that the test statistics are good here. But it was still worse than the previous raw data. So, in our case the log transformation didn't work as we expected.

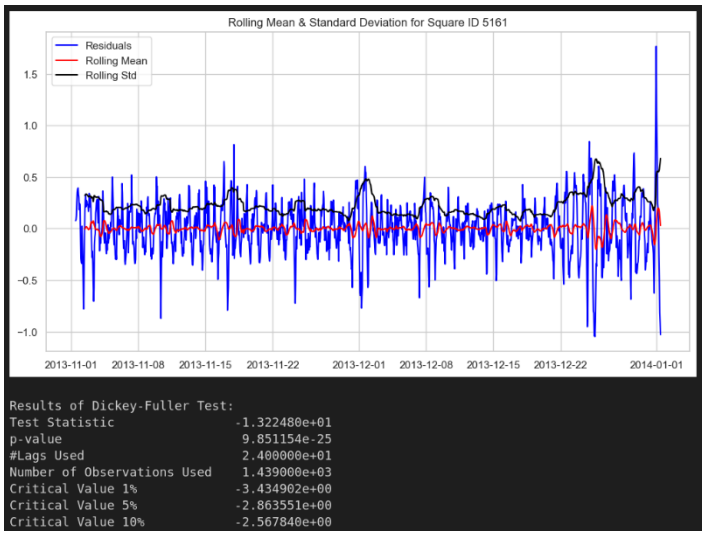

Figure 24

Now, let's take a look at the plots and figures of the other 2 areas in focus. The following figure is the plot of original data for area 4159 with its mean and std:

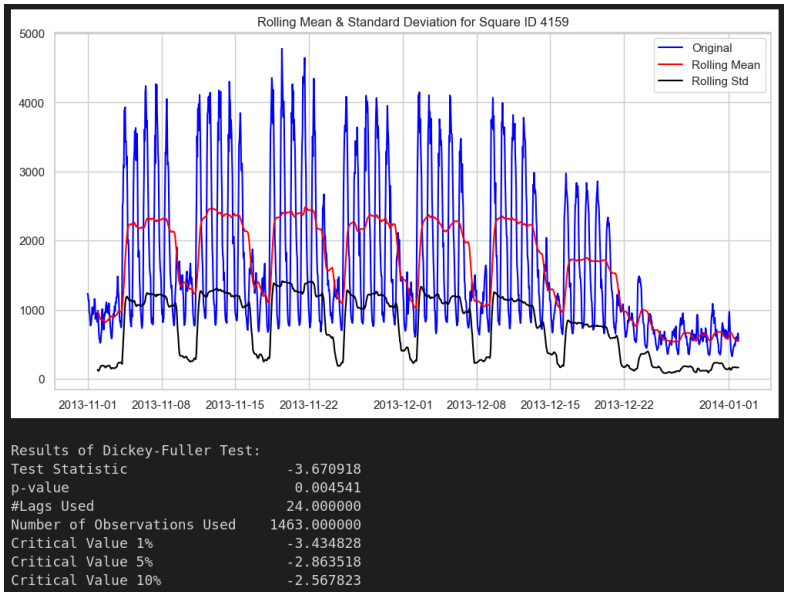

Figure 25

As we can see the test statistics are lower than the 1% critical value and we can say that time series of internet traffic at area of 4159 is a stationary time series. The difference between the max and min of

mean of the traffic is a larger percentage of the max of internet traffic in this area than the same ratio for area 5161. The following is the same plot for the residual part of the internet traffic for area 4159:

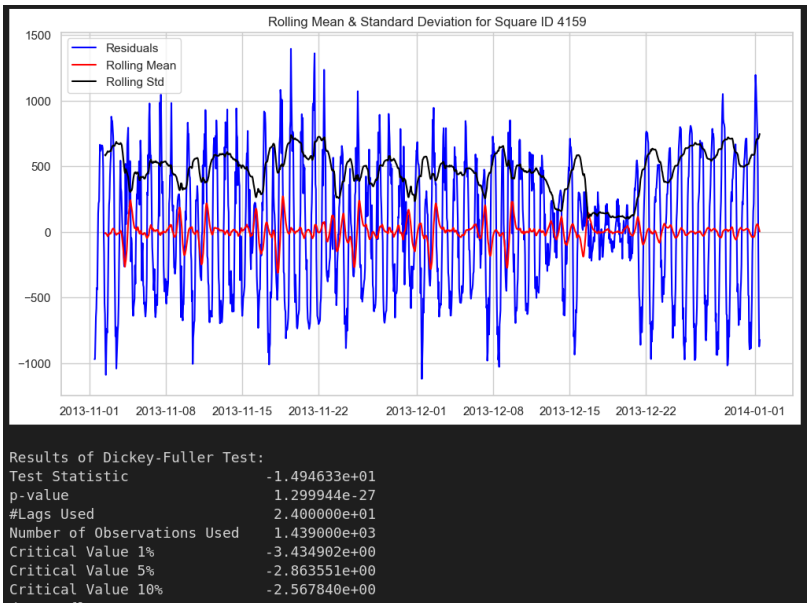

Figure 26

The test stats for residual part of data in area 4159 also shows that the internet traffic time series in this area is a stationary time series with great confidence.

If we take log of the data and then perform the test, the result is as shown in the following plot:

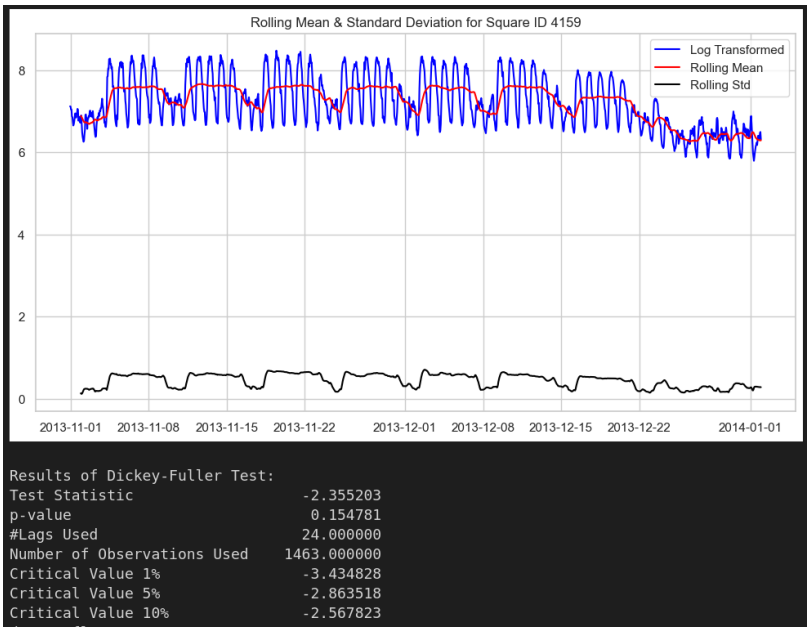

Figure 27

As you can see from the results of the test, after logging the data is less likely to be stationary.

And the test result and plot for the residual part of the logged data is like this:

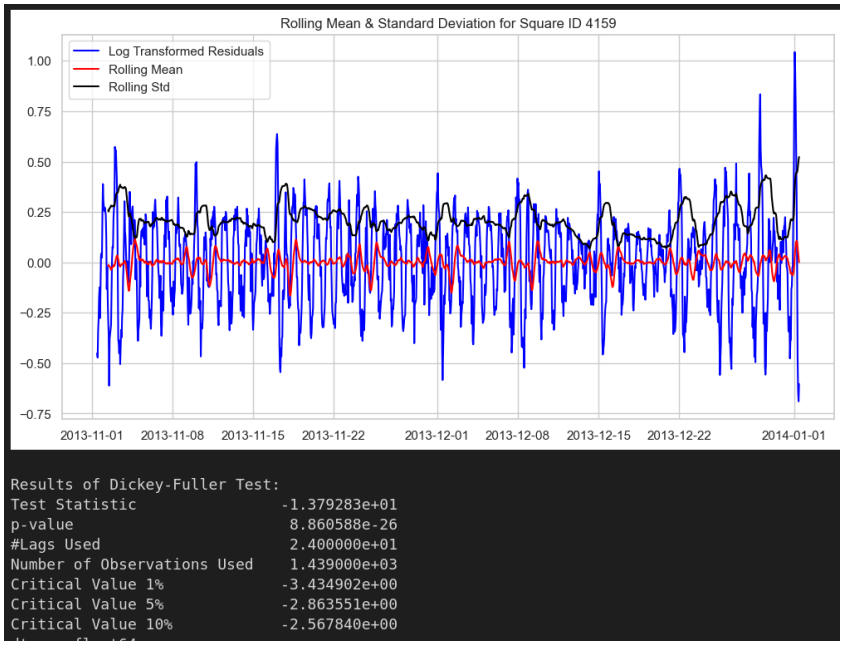

Figure 28

So just like the previous area we won't be using the logged data for the ARIMA model later.

Now let's look at the last area, which is 4556. The following plot is the raw time series of the area 4556, and the test results of that:

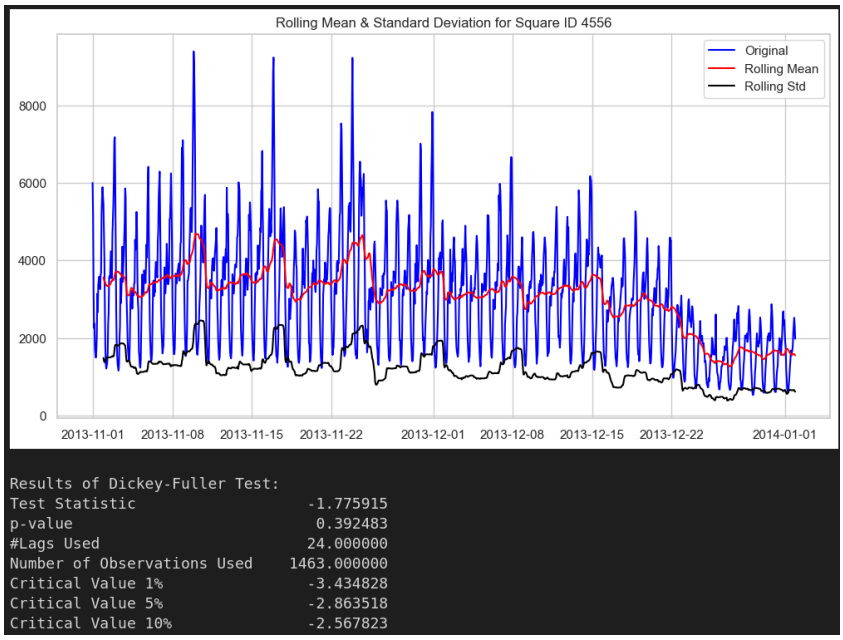

Figure 29

As we can see, the result is not pretty good, according to the Dickey-Fuller test the data is less than 90 percent stationary.
Now let's look at the plot and test result for the residual part of this traffic in this area:

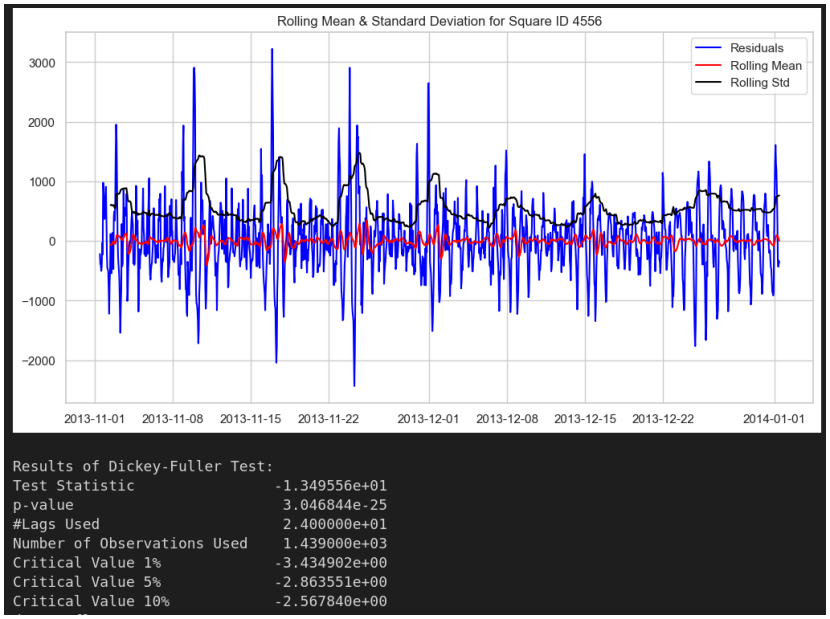

Figure 30

As you can see the residual part of the data is considered stationary with good confidence. So, we can suggest that most of the non-stationarity of the raw data was because of the seasonality and trend, so we can expect that the SARIMA model that considers seasonality of data would still work good on this area.

Now for the last plots of this sub-section let’s take a look at the taking a log of the raw data in area 4556 does help or not:

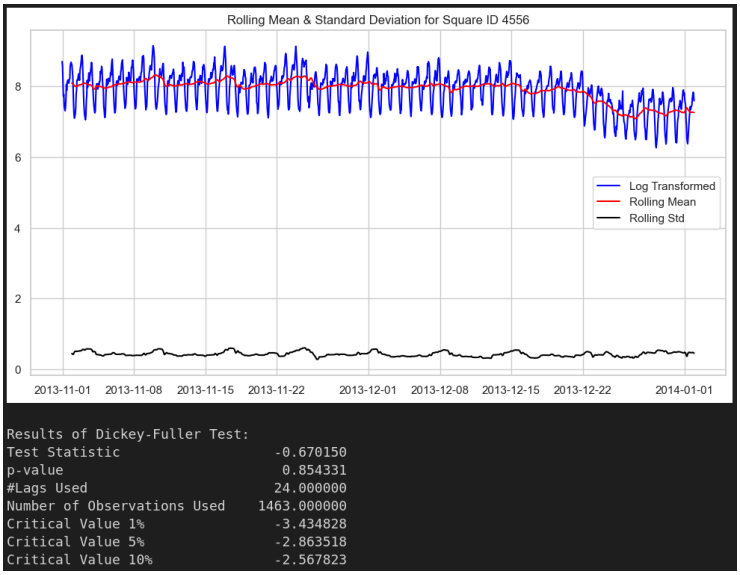

Figure 31

As you can see from the test result, time series became worse than the already non-stationary timeseries before taking log of it. The following is the time series of residual part of data:

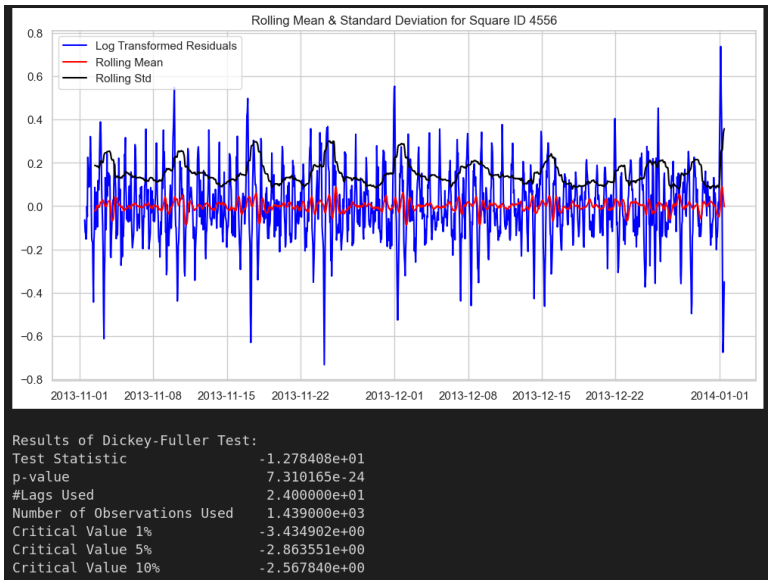

Figure 32

We can see that residual part of data is still a stationary time series with good probability.

Now that we have seen the figures from all three areas, we can say that we are not going to use the logged data for the ARIMA model.

6.1.4 Seasonality and Trend for timeseries

For the last section before we predict the time series, let's take a look at the seasonality and trend for the internet traffic time series of these three areas.

For area 5161 the plot is like this:

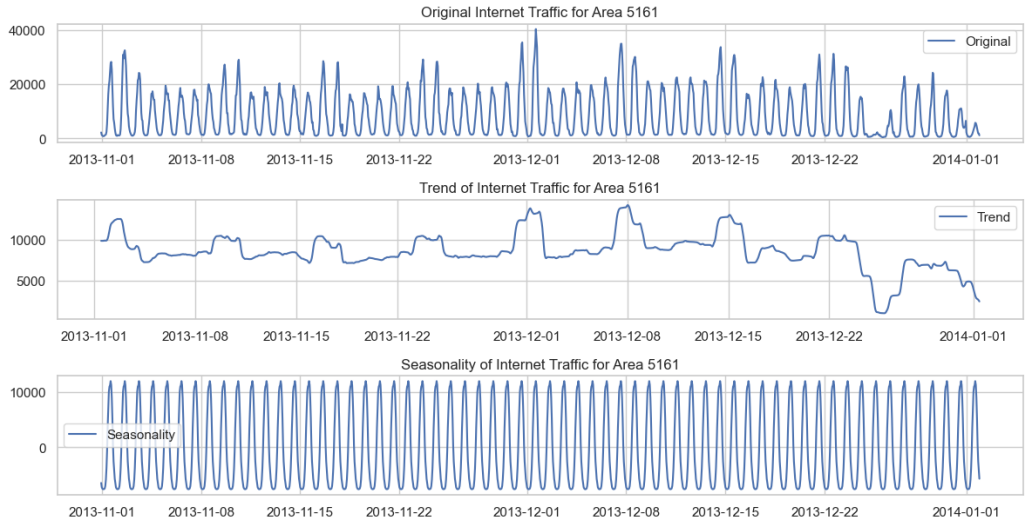

Figure 33

As you can see, and it's visually obvious from the original data, there is a very perfect seasonality in our time series for this area. Also, we can see that the traffic hits a peak each week during weekends in a positive way.

The plot for area 4159 is like this:

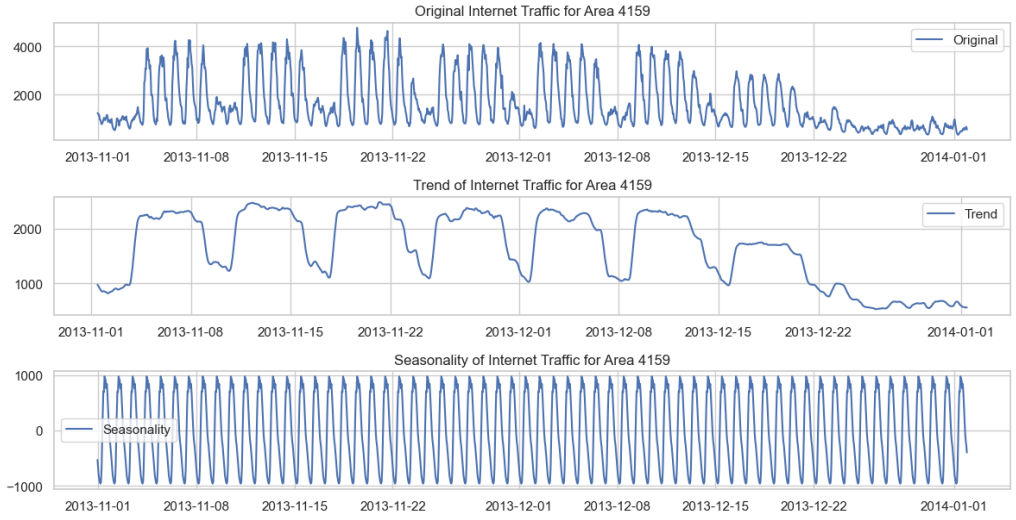

Figure 34

Just like previous areas the seasonality is obvious, and we expect that because internet traffic time series is a recurring phenomenon. People's general usage of the internet obeys predictable rules and patterns.

And last, the plot for area 4556 is like this:

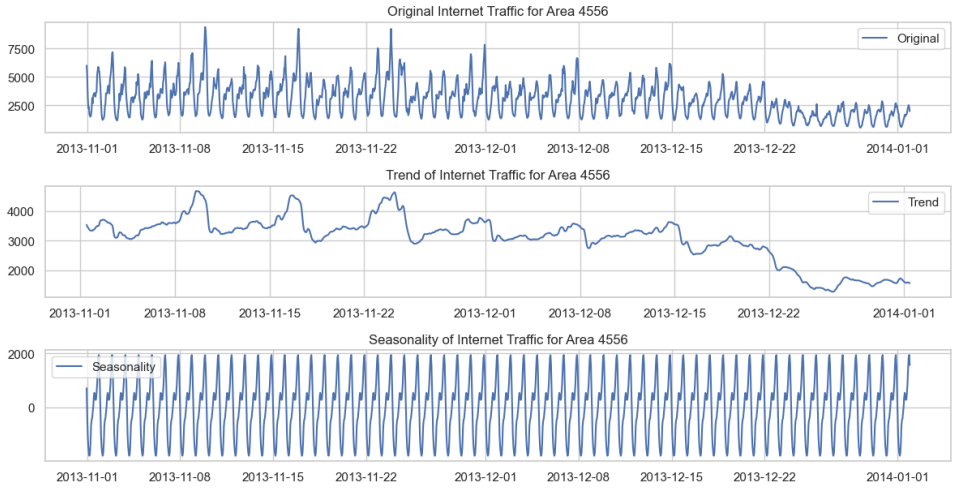

Figure 35

Just like the two other plots the seasonality is obvious, and the trend is recurring, and an interesting point is when we approach the end of year the trend is negative, and we see that the general mean of the time series starts to decrease. That can be the reason for the worse Dickey-Fuller test result in the previous section.

6.2 Traffic Prediction

In this part we are going to predict the internet traffic for three areas (5161, 4159, 4556) for the period of December 16 to 22. We will use two methods, the first is an ARIMA based method and the second is an LSTM method to predict internet traffic.

6.2.1 SARIMA Method

As we have seen in the previous section, the internet traffic time series for mentioned areas are both stationary (for 5161 and 4159 with high confidence and for 4556 with lower confidence). So, we can use ARIMA methods. But we also saw that our time series had very perfect and precise seasonality for all the duration of dataset. Having that in mind, we choose the SARIMAX model for this prediction.

SARIMAX stands for Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors. It extends the ARIMA model to include seasonality and the possibility of including additional variables (the 'exogenous' part). Like ARIMA, SARIMAX models capture a suite of different standard temporal structures in time series data. What makes it special is its ability to handle seasonality. The 'S' in SARIMAX stands for 'seasonal', which means the model is capable of learning and predicting based on seasonality in time series data. For example, if we expect the internet traffic to be higher during weekends compared to weekdays, this would be a form of weekly seasonality which SARIMAX can incorporate into its predictions.

Because selecting perfect parameters for any ARIMA model is pretty hard, we use auto_arima method from ‘pmdarima’ library that receives multiple parameters and an objective function. Auto ARIMA is a methodology applied in automated time series forecasting. The idea behind auto ARIMA is to use a systematic approach to arrive at the optimal parameters for the ARIMA or SARIMA model. This involves

a combination of grid search and sophisticated algorithms to find the model that minimizes a certain metric, for our case, the Akaike Information Criterion (AIC). The advantage of this approach is that it saves the analyst from having to manually specify multiple different combinations of parameters to find the best fit.

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are essential tools in time series analysis. They help us understand how each data point in a series is influenced by its predecessors. ACF is a measure of the correlation between the time series with a lagged version of itself, while PACF reveals the correlation of the residuals (i.e., the amount left after subtracting the previous value estimated by the AR model) with the previous actual values of the series. We use the ACF and PACF to help us set intervals and boundaries for auto_arima method to get to the optimal value in less time.

We expect the SARIMAX model to capture not only the general trends (e.g., increasing or decreasing internet usage) but also the specific weekly patterns (e.g., higher usage on weekends or specific times of day).

We used the auto_arima method once and then saved the parameters in a csv file named 'ARIMA-model-parameters.csv' to save time running the process and plotting for later.

6.2.1.1 Timeseries Prediction for area 5161

Let us start with area 5161. The ACF and PACF plots for this area are like this:

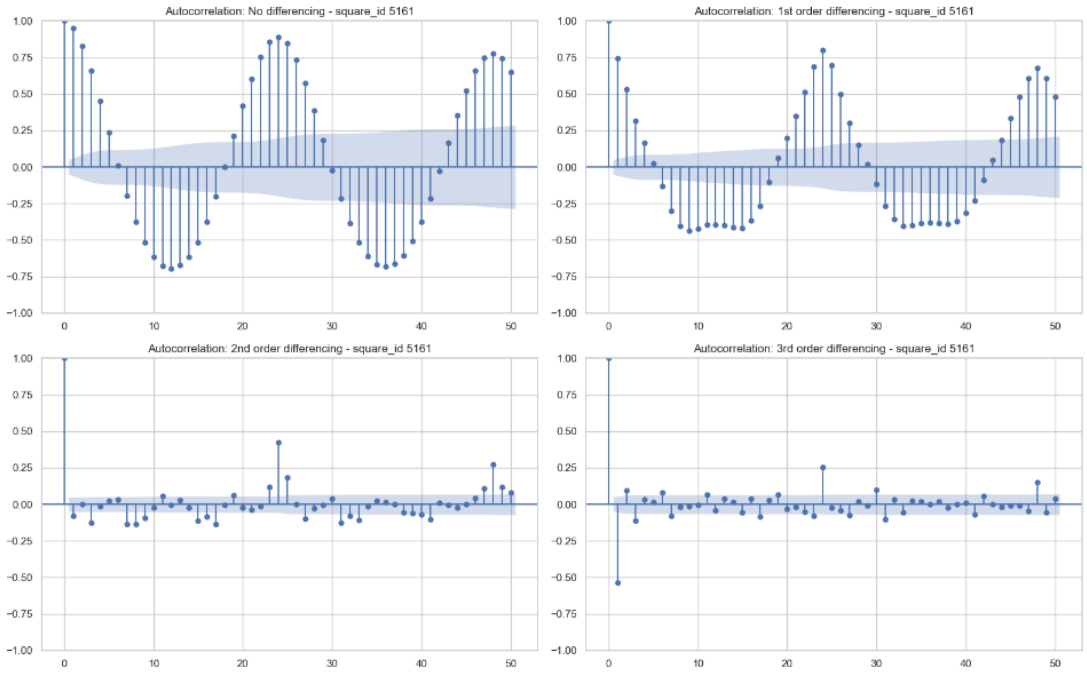

Figure 36

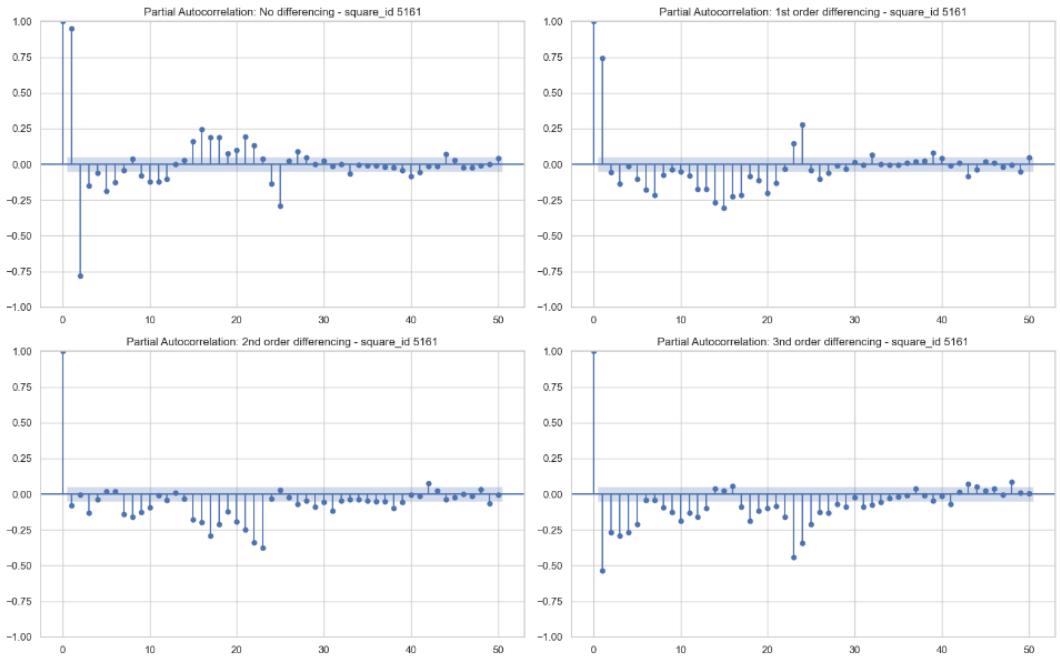

Figure 37

What we are trying to identify in these plots are the levels of differencing where most of the data points fall within the blue shaded region, which represents noise. When the majority of points fall in this area, it suggests that the series has been made stationary and is suitable for forecasting.

For the Autocorrelation Function (ACF) plot, we see that this occurs at the 2nd level of differencing. This means that the time series becomes stationary when we take the difference between each data point and the two steps back data.

For the Partial Autocorrelation Function (PACF) plot, stationarity is achieved at the 1st level of differencing, meaning that subtracting each data point from the previous one helps to make the series stationary.

In both ACF and PACF plots, the first significant data point outside of the blue noise area is at lag 24. This is expected and aligns with our earlier observations. As we have identified daily and weekly seasonality in our data, this lag of 24 hours indicates the daily cycle in the time series. It shows that the traffic patterns today are influenced by the traffic patterns exactly one day (or 24 hours) ago, reinforcing the presence of daily seasonality in our data.

The auto_arima function has provided us with two sets of parameters: the order and the seasonal_order.

- order: (1, 0, 1): These parameters represent the (p, d, q) of the ARIMA part of the model:
    - p (AR terms) = 1: This suggests that the model uses 1 lagged value of the series in the forecasting equation. In other words, it uses the value of the series from one period ago to forecast the current value.

    - d (differencing) = 0: This means the model did not require any differencing to make the series stationary, indicating that the original series is already stationary. (As we have witnessed in previous section that timeseries of area 5161 is stationary)
    - q (MA terms) = 1: This suggests that the model uses the error of the first lagged forecast in the equation. In other words, it uses the difference between the predicted value and actual value from one period ago to improve the current forecast. (As seen in PACF plot)
- seasonal_order: (1, 0, 2, 24): These parameters represent the (P, D, Q, S) of the seasonal component of the model:
    - P (Seasonal AR terms) = 1: This indicates that the model uses 1 lagged value of the seasonal difference in the equation. This is the seasonally adjusted equivalent of 'p' from the order parameter.
    - D (Seasonal differencing) = 0: This means that no seasonal differencing was needed. This indicates that the seasonality in the series does not need to be adjusted to make the series stationary.
    - Q (Seasonal MA terms) = 2: This suggests that the model uses the errors of the first two lagged seasonal forecasts in the equation. This is the seasonally adjusted equivalent of 'q' from the order parameter.
    - S (seasonal length) = 24: This tells us the length of the seasonal cycle. In this case, the cycle is 24 periods (in our hours), which aligns with our understanding of the data having a daily seasonality. (As we know our data is resampled to hourly data and internet traffic has a daily -24 hour- seasonality)

Now if we provide these parameters to SARIMAX method and plot the result it is like this:

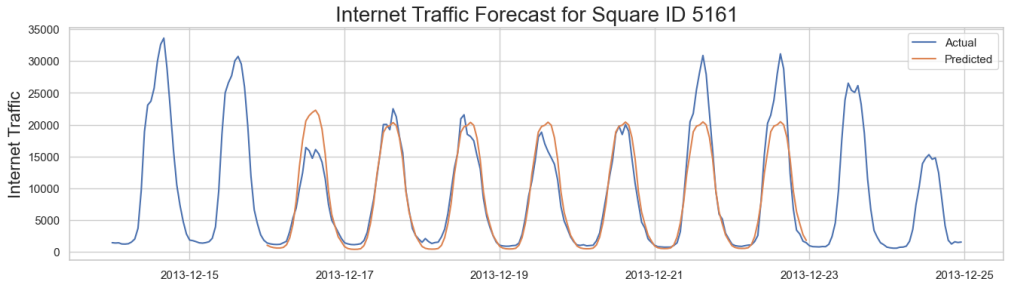

Figure 38

As you can see our algorithm could predict the lower peak during for most of the week but couldn't keep up the good prediction for after the multiple closed days on 22 of December when the traffic spikes again. The optimization metrics we have are AIC, MAE and MAPE:

```
AIC for sarima (Square ID 5161): 18644.39681149649
MAE for sarima (Square ID 5161): 1643.965144298266
MAPE for sarima (Square ID 5161): 0.26055735896589194%
```

Figure 39

The AIC is around 18650, MAE is 1643 and the MAPE is about 26 percent. That means our algorithm predicted correctly almost 74% percent of the timesteps that is not the best, but according to the amount of data available and the simplicity of the model, it's a pretty good result.

6.2.1.2 Timeseries Prediction for area 4159

Now let's look at area 4159. The ACF and PACF plots for this area are like this:

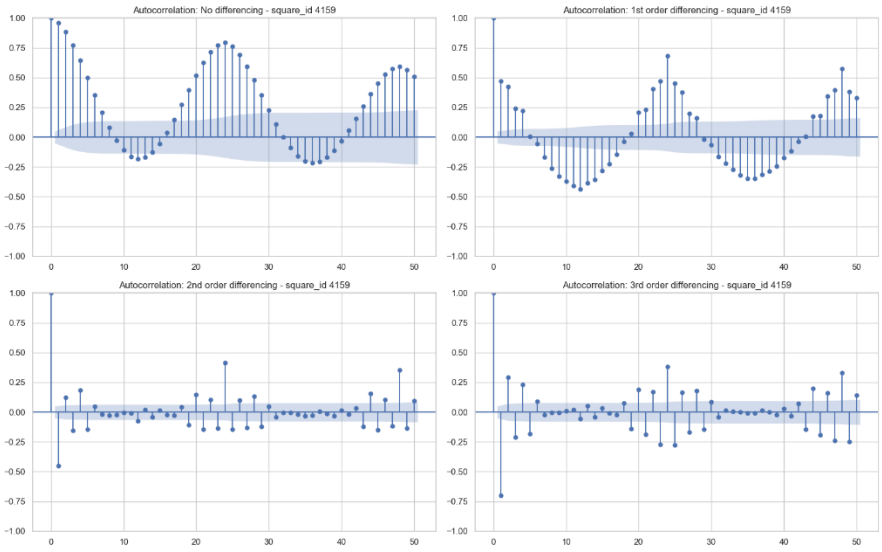

Figure 40

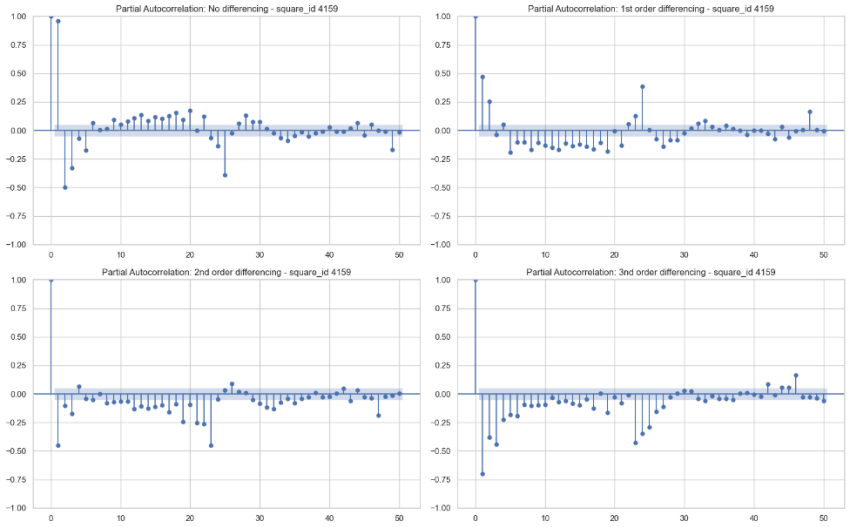

Figure 41

For the Autocorrelation Function (ACF) plot, we see that most point bars are at noise level at the 2nd level of differencing. Because if you look closely at both the 2nd and 3nd difference is quite like each other and in these situations, we choose the least level of differencing possible. Also, we can see that our time series didn't quit becoming stationary even after 2nd differencing. It will affect the accuracy of our prediction later.

For the Partial Autocorrelation Function (PACF) plot, stationarity is achieved at the 1st level of differencing.

Just like area 5161, in this area (4159) we can see that again in both ACF and PACF plots, the first significant data point outside of the blue noise area is at lag 24.

The auto_arima function has provided us with two sets of parameters: the order and the seasonal_order.

- order: (2, 0, 0): These parameters represent the (p, d, q) of the ARIMA part of the model:
    - p (AR terms) = 2: This suggests that the model uses 2 lagged values of the series in the forecasting equation.
    - d (differencing) = 0: This means the model did not require any differencing to make the series stationary, indicating that the original series is already stationary. (As we have seen in the previous section that the time series of this area is stationary)
    - q (MA terms) = 0: This indicates that the model does not use the error of any lagged forecasts in the equation. This suggests that the recent forecast errors did not improve the forecast in this case.
- seasonal_order: (1, 0, 1, 24): These parameters represent the (P, D, Q, S) of the seasonal component of the model:
    - P (Seasonal AR terms) = 1: This indicates that the model uses 1 lagged value of the seasonal difference in the equation. This is the seasonally adjusted equivalent of 'p' from the order parameter.
    - D (Seasonal differencing) = 0: This means that no seasonal differencing was needed. This indicates that the seasonality in the series does not need to be adjusted to make the series stationary.
    - Q (Seasonal MA terms) = 1: This suggests that the model uses the error of the first lagged seasonal forecast in the equation. This is the seasonally adjusted equivalent of 'q' from the order parameter.
    - S (seasonal length) = 24: This tells us the length of the seasonal cycle. In this case, the cycle is 24 periods (or hours), which aligns with our understanding of the data having a daily seasonality. (As we know our data is resampled to hourly data and internet traffic has a daily -24 hour- seasonality)
    -

Now if we provide these parameters to SARIMAX method and plot the result it is like this:

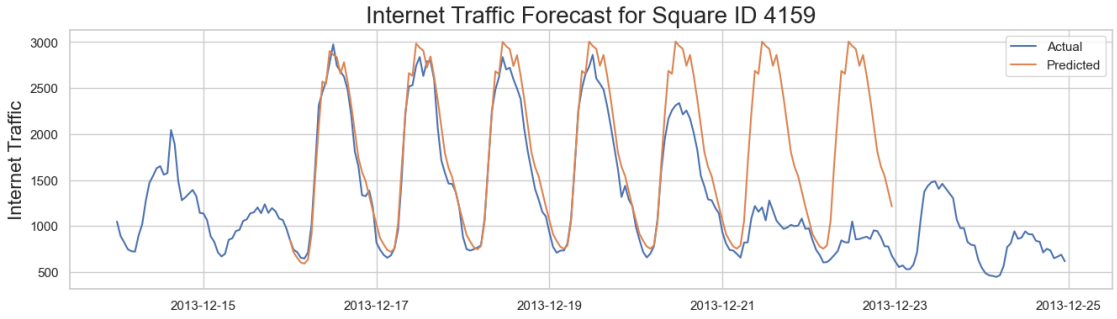

Figure 42

As we can see from the figure our algorithm predicted the workdays pretty good but couldn't keep up with weekends. The optimization metrics result is as follows:

```
AIC for sarima (Square ID 4159): 14779.145998317468
MAE for sarima (Square ID 4159): 398.82219597572856
MAPE for sarima (Square ID 4159): 0.37350299774428247%
```

Figure 43

So, the MAPE is 37% that tells us, our model was correct about 63% of the time. That is not too good.

6.2.1.3 Timeseries Prediction for area 4556

Now let's look at our last area which is 4556. The ACF and PACF plots for this area are like this:

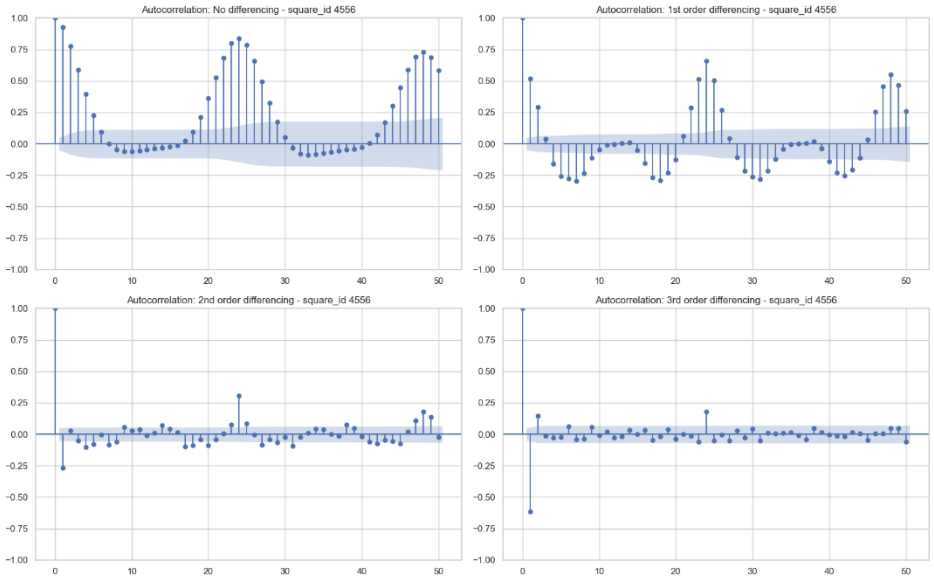

Figure 44

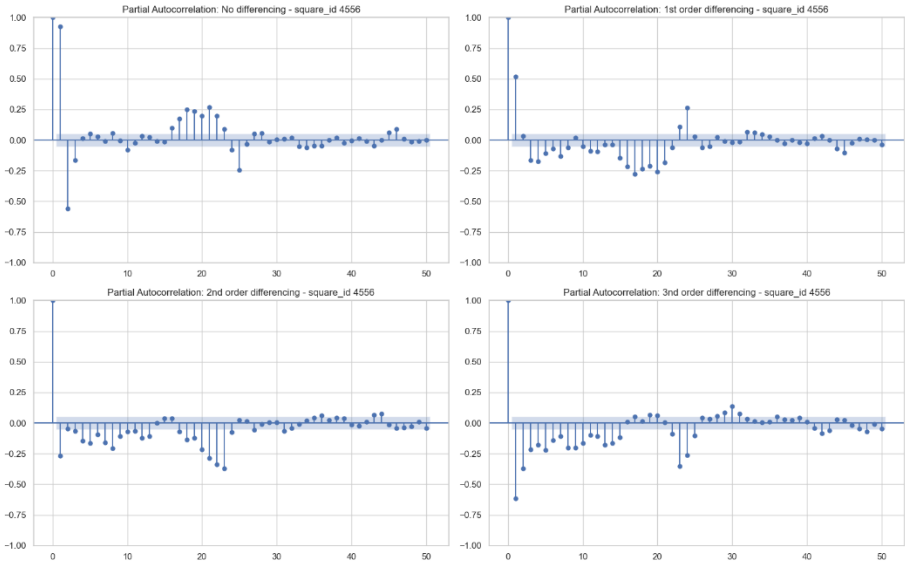

Figure 45

For the Autocorrelation Function (ACF) plot, we see that most point bars are at noise level at the 2nd level of differencing.

For the Partial Autocorrelation Function (PACF) plot, stationarity is achieved at the 1st level of differencing. And there is no noticeable difference between 1nd, 2nd and 3nd differencing levels.

Just like the previous section, in this area (4556) we can see that again in both ACF and PACF plots, the first significant data point outside of the blue noise area is at lag 24.

The auto_arima function has provided us with two sets of parameters: the order and the seasonal_order.

- The auto_arima function has provided us with two sets of parameters: the order and the seasonal_order.
    - order: (1, 0, 0): These parameters represent the (p, d, q) of the ARIMA part of the model:
    - p (AR terms) = 1: This suggests that the model uses 1 lagged value of the series in the forecasting equation.
    - d (differencing) = 0: This means the model did not require any differencing to make the series stationary, indicating that the original series is already stationary. (As we have seen in the previous section that the time series of this area is stationary)
    - q (MA terms) = 0: This indicates that the model does not use the error of any lagged forecasts in the equation. This suggests that the recent forecast errors did not improve the forecast in this case.
- seasonal_order: (1, 0, 1, 24): These parameters represent the (P, D, Q, S) of the seasonal component of the model:
    - P (Seasonal AR terms) = 1: This indicates that the model uses 1 lagged value of the seasonal difference in the equation. This is the seasonally adjusted equivalent of 'p' from the order parameter.
    - D (Seasonal differencing) = 0: This means that no seasonal differencing was needed. This indicates that the seasonality in the series does not need to be adjusted to make the series stationary.
    - Q (Seasonal MA terms) = 1: This suggests that the model uses the error of the first lagged seasonal forecast in the equation. This is the seasonally adjusted equivalent of 'q' from the order parameter.
    - S (seasonal length) = 24: This tells us the length of the seasonal cycle. In this case, the cycle is 24 periods (or hours), which aligns with our understanding of the data having a daily seasonality. (As we know our data is resampled to hourly data and internet traffic has a daily -24 hour- seasonality)

Now if we provide these parameters to SARIMAX method and plot the result it is like this:

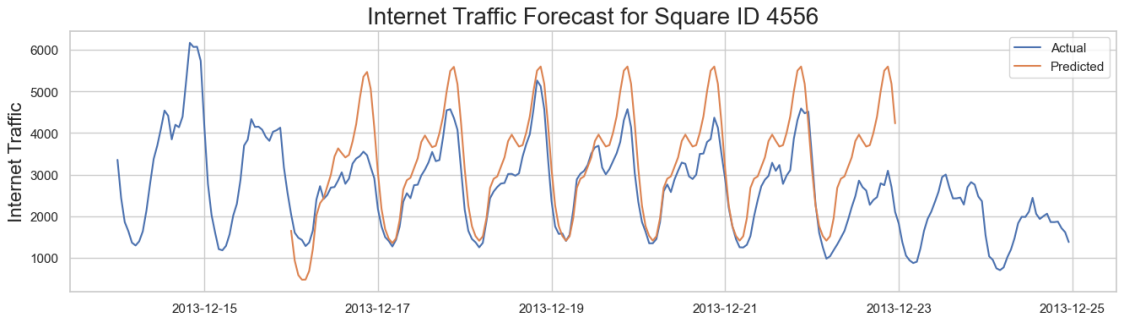

Figure 46

As we can see from the figure our algorithm predicted the workdays pretty good but couldn't keep up with weekends again, also we have an offset in most of the time steps from the actual data.

The optimization metrics result is as follows:

```
AIC for sarima (Square ID 4556): 16044.244314854159
MAE for sarima (Square ID 4556): 681.6256174917886
MAPE for sarima (Square ID 4556): 0.2645130968534066%
```

Figure 47

So, the MAPE is 26% that tells us, our model was correct about 75% of the time.

6.2.1.4 How to improve SARIMA

In total I think SARIMA did well on this task. About 75% of accuracy for a model that can be trained and set up to work that easy is really good. But if you take a closer look at the plots, we can see the predicted traffic for the first day of prediction time, was repeated for the other 5 days. I think that because of the reason that ARIMA can have only 1 seasonality parameter. For example, if we could set two seasonality parameters and we set one for 24 that shows the daily seasonality and then set the other one to 168 that shows the weekly seasonality the results would be much better. For example, we may be able to predict the decrease in traffic during weekends for areas 4556 and 4159. But I couldn't find an option on the library used or for SARIMA models to do that (work with two seasonality), I tried to develop something by myself, but I ended up receiving not satisfactory results.

6.2.2 LSTM method

During my research on forecasting timeseries data. I came across a method called LSTM. LSTM stands for Long Short-Term Memory. It's a type of recurrent neural network (RNN), a class of neural networks that excel in learning from sequential data. However, traditional RNNs struggle to learn from sequences that have long-term dependencies due to an issue called the vanishing gradient problem. Here is where LSTM comes in handy; it is specifically designed to overcome this issue.

In simple terms, LSTM networks have a kind of memory mechanism, allowing them to remember or forget information over time. This 'memory' is controlled by structures in the network called gates.

LSTMs are particularly effective for time series forecasting, which is our task. Because they can remember information for long periods, they're able to leverage past information (e.g., traffic patterns from previous hours or days) to predict future trends (e.g., future internet traffic). In our case, this ability can help us predict internet traffic based on historical data.

But LSTMs have some requirements to do well:

- Data Preprocessing: LSTMs expect input data to be in a specific format, usually a 3D array. The three dimensions of this array are the samples (individual instances of data), time steps (the sequential steps within each instance), and features (the variables we measure at each time step that is internet traffic). We must structure our time series data accordingly.
- Scale of Data: LSTMs, like many other neural networks, perform best when the input data is scaled, typically between 0 and 1 (normalization). This helps the network learn more efficiently.
- Choosing the Right Parameters: Parameters such as the number of layers in the network, the number of neurons in each layer, the type of optimizer, the learning rate, and many others can significantly impact the performance of the LSTM. It can take some trial and error to find the best parameters for your specific problem.
- Sufficient Data: LSTM networks usually require a large amount of data to learn effectively. The more high-quality, relevant data we can feed the model, the better its predictions are likely to be.
- Sequence Length: The choice of sequence length or the amount of previous time steps to use as input (also called the 'window size') is crucial. If the window size is too short, the model might miss valuable information. If it's too long, the model may struggle to learn from the data.

Now let’s see if we can get this method right. The predicted traffic for area 5161 by LSTM is like this:

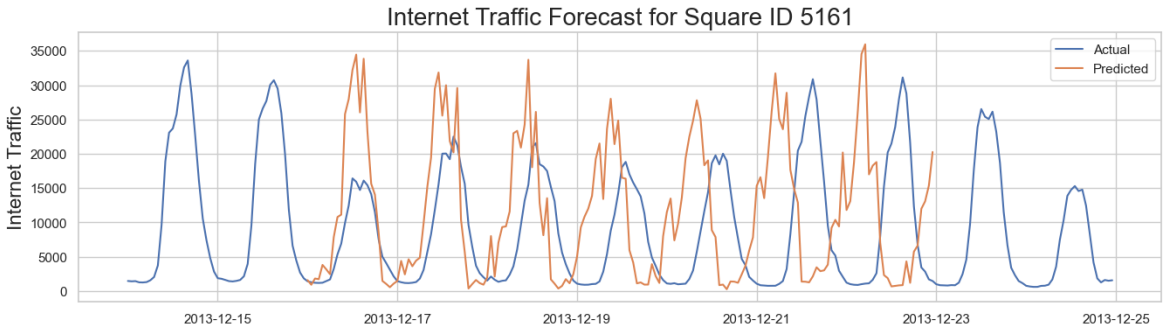

Figure 48

AS you can the prediction is very bad, but at least could get the shape somehow right. We have changed the number of layers and number of neurons in each layer, but we couldn’t get a better result. And the MAPE is 420%. Yes 420% that not a typo. As I read the blogs for this problem, it seems that either it’s because there are prediction data with value of zero or the amount of error is that high that the function returned more than 100 percent. From the plot we can say that the first reason seems more rational here. But generally, LSTM does not appear to be a good option for the data we have.

The plot for area 4159 is like this:

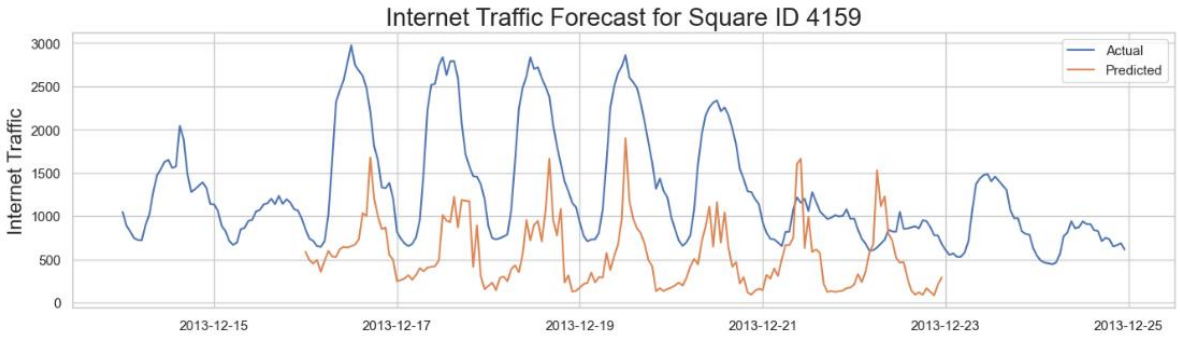

Figure 49

Again, for workdays the shape is not the worst but for the weekend the prediction is literally the worst thing we can produce. And the MAPE is about 63%. This MAPE supports our hypothesis for the previous plot that the enormous number of MAPE for 5161 is because of zero values in the predicted timeseries.

And at last, the plot for 4556 is like this:

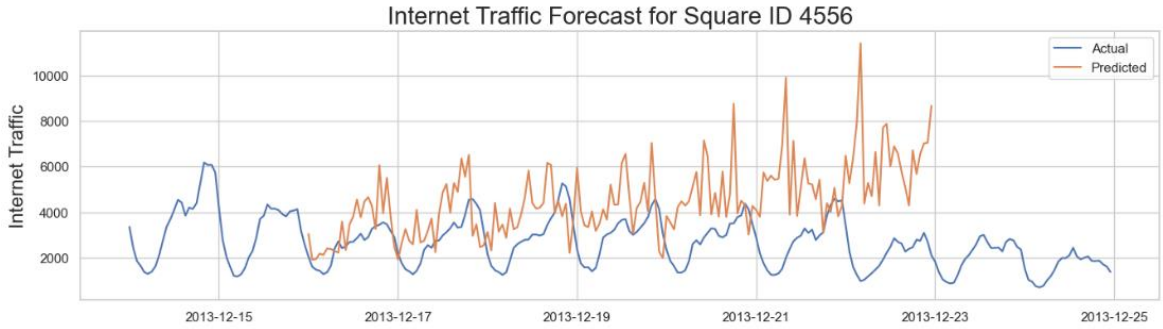

Figure 50

As we have seen for the other two areas the predictions are not good. Bu at least the general shape of the prediction for December 16 to 20 is somehow likely to actual data. The MAPE for this area is 97% that is not good at all.

I think that the LSTM didn't work because of the amount of data we had. Maybe If I used the main dataset that the timesteps were 10 munities we would have get a better outcome but now I don't have the hardware power and capacity to run that huge 10 GB data set and run the LSTM of 5-layer LSTM with 200 neurons.