IMDEA Assessment

Harsh Dinesh Mehta

Contents

Introduction:

This study will examine internet traffic in Milan during November and December 2013. This analysis was completed as part of the IMDEA assignment.

The assignment consists of two major elements. First, we are entrusted with extracting insights and broad information about internet traffic in Milan. Next, we will create predicted models for the city's internet traffic from December 16th to December 22nd.

For this assignment we will be using google colab with the following specifications:

CPU: x86_64

CPU Count: 8

Total Memory (RAM): 50.99 GB

Operating System: Linux 6.1.85+

Python Version: 3.10.12

Pip Version: pip 24.1.2

Data preprocessing:

The dataset used for this project was created as part of Telecom Italia's Big Data Challenge. This dataset captures SMS, phone, and internet activities across Milan and Trentino province.

The geographic area is Divided into 10,000 equal components (100x100 cells), with data shown in rows. Each row shows actions within a grid square (identified by a 'square_id') within a 10-minute time span. The dataset includes text files for each day from November 1, 2013, to December 31, 2013. The collection contains a geojson file that describes each grid square's position, allowing for visualization on geospatial maps.

We used dataverse API to import data into my google drive and then convert the .txt files into a combined CSV file with the column header named: "sms_in", "sms_out", "call_in", "call_out", square_id, time_interval, country_code, internet_traffic.

For the given tasks in the assignment, we only need square_id, time_interval, internet_traffic so we can drop rest of the columns. The 'time_interval' data, originally in timestamp format, was then converted to datetime format. To align with our analytical needs. The dataset also provided a geojson file that locates the geographical location of each square_id. 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:

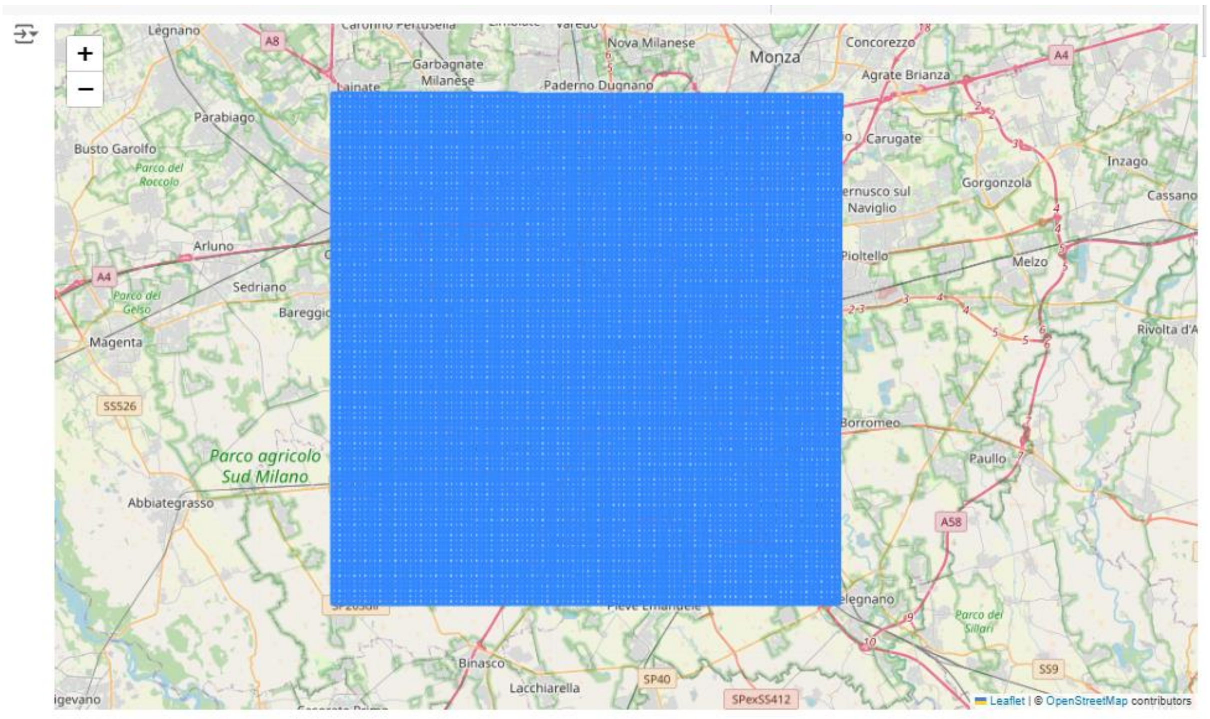


Figure 1

Task 1

We will first use the data to represent the distribution of internet traffic across the city by grouping the dataset based on square_id while summarizing the total amount of traffic per grid cell over the analysis period. Using this grouped data, we continued exploring how to plot the PDF, which will be able to visually understand how traffic is distributed among Milan's grid cells.

Histogram Analysis

Histograms are an effective way of displaying the internet-traffic distribution if the data needs to be binned. In this exercise, we used two of the most applied binning schemes:

Sturges' Rule

This rule suggests a smaller number of bins, calculated based on the size of the dataset, and hence gave a simpler view of traffic distribution. But the drawback of this method is that it may fail to provide finer detail in skewed data such as this one. It suggests that the number of bins (k) can be calculated as by k = 1 + log2(n)

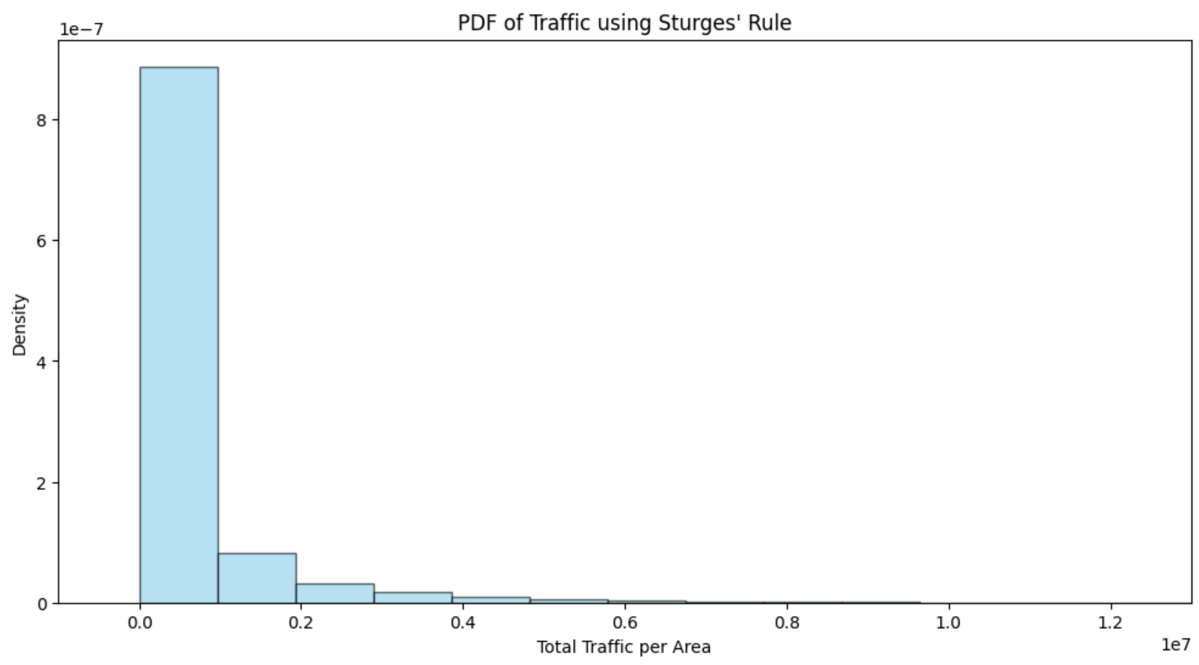


Figure 2

The histogram is skewed to the extreme left since most areas show very low total traffic, bunched up on the left side of the plot (near zero), where density decreases sharply with increased traffic. This means that a large proportion of Milan's grid cells experience low traffic, while some spots display much higher usage.

Freedman-Diaconis Rule

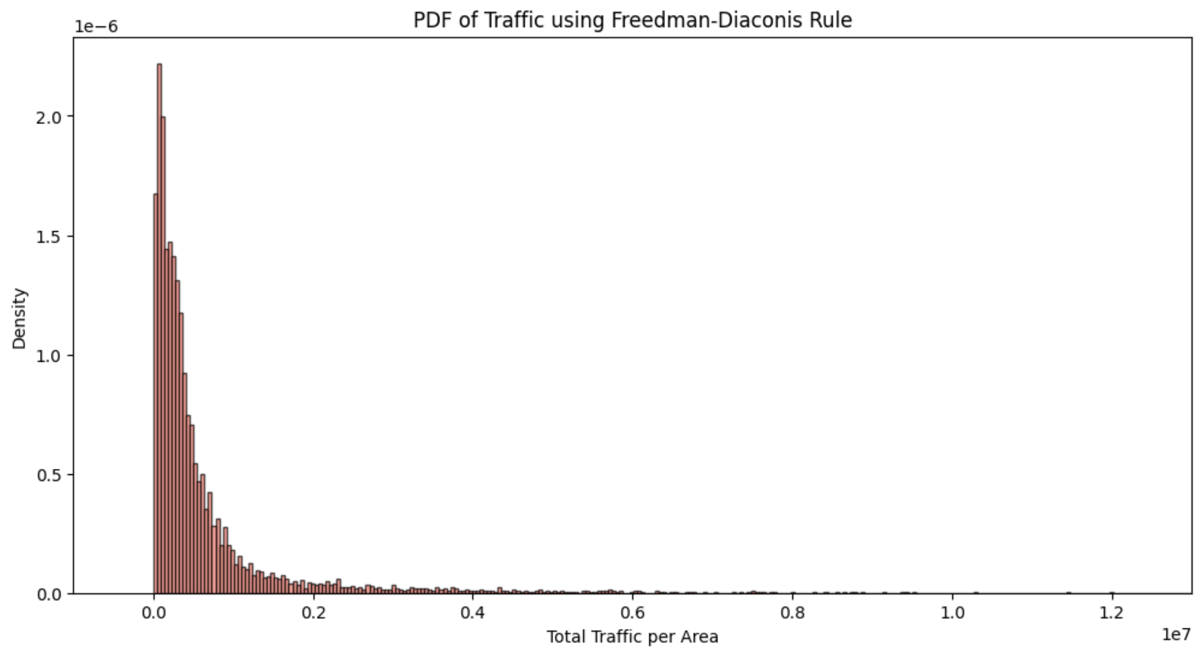


Figure 3

This method deals with the variation of data, using the interquartile range-IQR. This generated a much larger number of bins than Sturges' rule did and showed much more finely the distribution of the traffic across cells. This rule is good to be used when dealing with skewed data and the cases with the significantly variable traffic within Milan's grid cells it generates the best fit. the number of bins (k) can be calculated as by bin_width = 2 * (IQR) / (n ** (1/3)) k = (data_range) / (bin_width)

This histogram comparison showed that most grid cells were in the low-to-moderate traffic class, while others were tremendously high, indicating a right-skewed distribution.

Both rules appropriately quantify the PDF of internet traffic in Milan for two months. However, the Freedman-Diaconis rule captures more information from data because it produces finer granularity bins, thus a more detailed pattern of distribution. Though both methods are useful, we have settled on the Freedman-Diaconis method as the best method to be used to represent the histogram of internet traffic in this analysis.

Gaussian Kernel Density Estimation Method

To smoothen out the traffic distribution, we used a non-parametric technique called Kernel Density Estimation. KDE produces a continuous density function, and the choice of Gaussian KDE overlays Gaussian curves at every point in the data set, generating a smooth probability density curve. Thus, the visualization of traffic distribution was made without predetermined bins while the overall distribution and density peaks were captured effectively. The Gaussian kernel density estimator is defined as: K(x) = 1/(sqrt(2*pi)) * exp(-x^2/2)

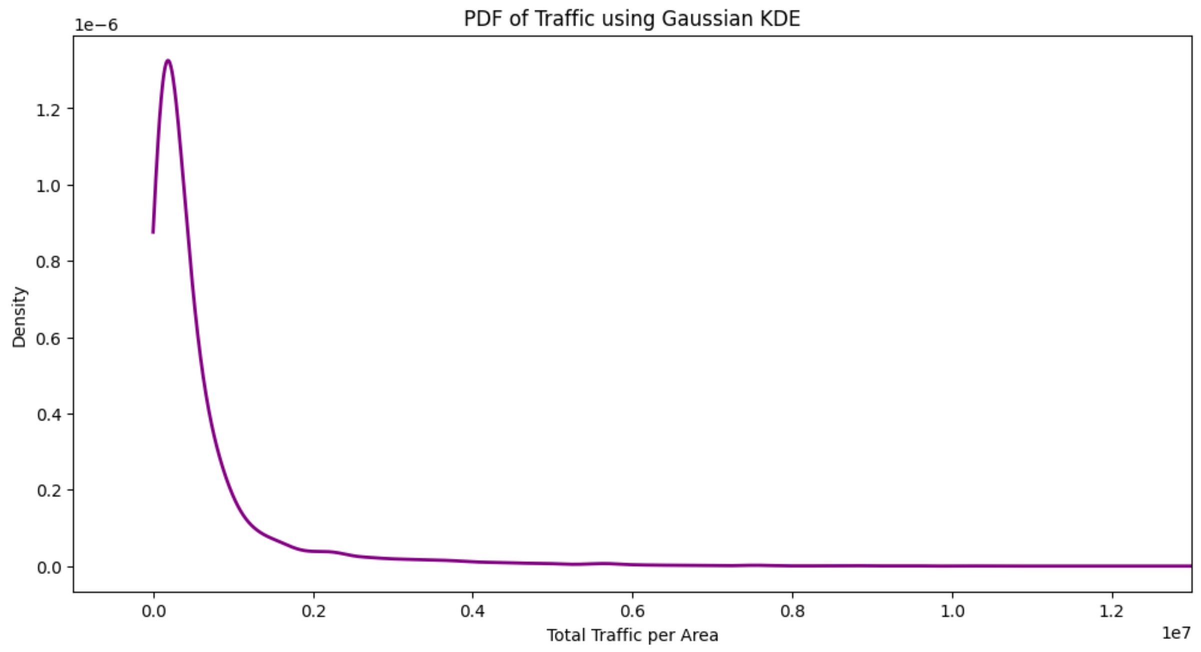


Figure 4

Inferences from Traffic Distribution Analysis

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 55000 units of internet traffic. This extends to 90% of cells accounting for equal to or less than 65000 units and 95% of cells having an equal or lesser load of 80,000 units of internet traffic during the study period.

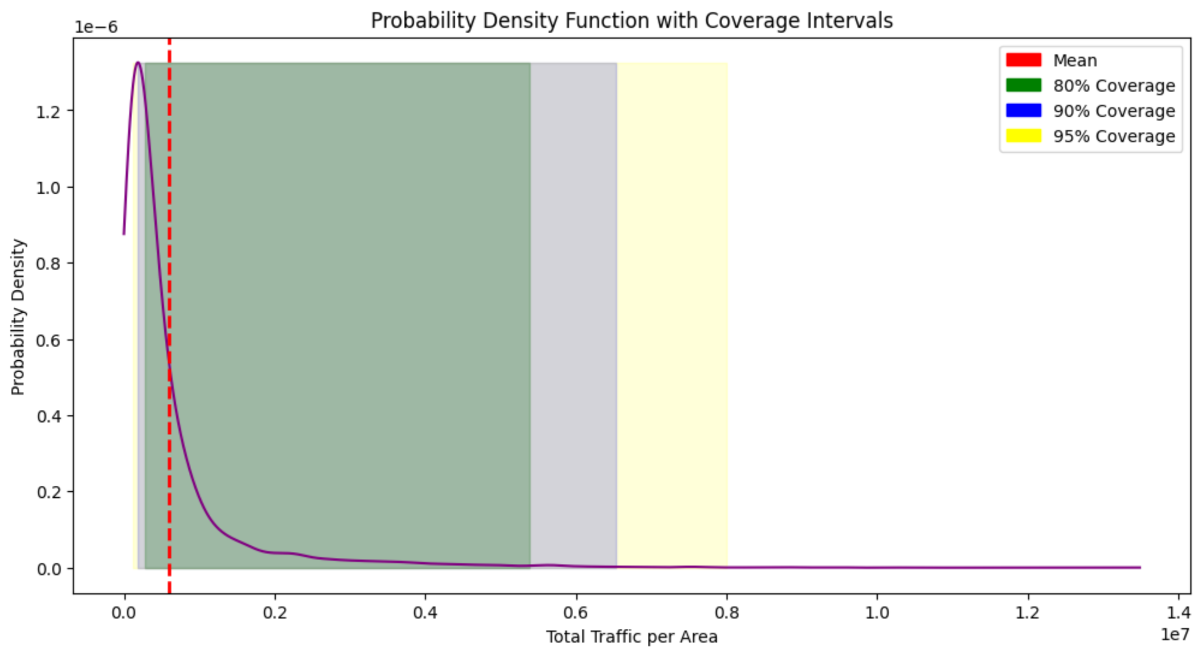


Figure 5

All these findings can be very helpful for network vendors, city planners, and the city administrators themselves. Traffic concentration identification allows them to optimize the resources from the network, prioritize their investment in infrastructure, and have efficient internet service management across Milan. Further, this identification of the congested areas supports proactive infrastructure planning like increasing network capacity for the peak demands at easily identified areas for minimizing the service disruption in such major hubs.

Comparison of three areas

At this analytical stage, we will compare internet traffic across the three areas that were given in Milan as follows:

The area with the greatest overall internet traffic for the two-month period.

The area with square_id 4159.

The area with square_id 4556.

First, we needed to identify which of these regions experienced the highest total volume of internet traffic over this period of time; in our case, this is square_id 5161. Most probably it is a high traffic region, or maybe because it is commercial or tourist-based activity that of course will be reflected in its total volume of traffic.

Then, we investigated and visualized the internet traffic flow over time for all three of these regions. We can plot each of those to see how traffic changes day-to-day in that area, in terms of peaks and even general trends, then compare the most congested area with the other two we chose (4159 and 4556), and by this comparison, we would be able to know through the traffic density and patterns of all those different areas of Milan what could be the reasons behind more and less internet usage in each one of them.

Now, let's have a look at the plot of internet traffic for square_ids 5161, 4159, and 4556 in the figure.

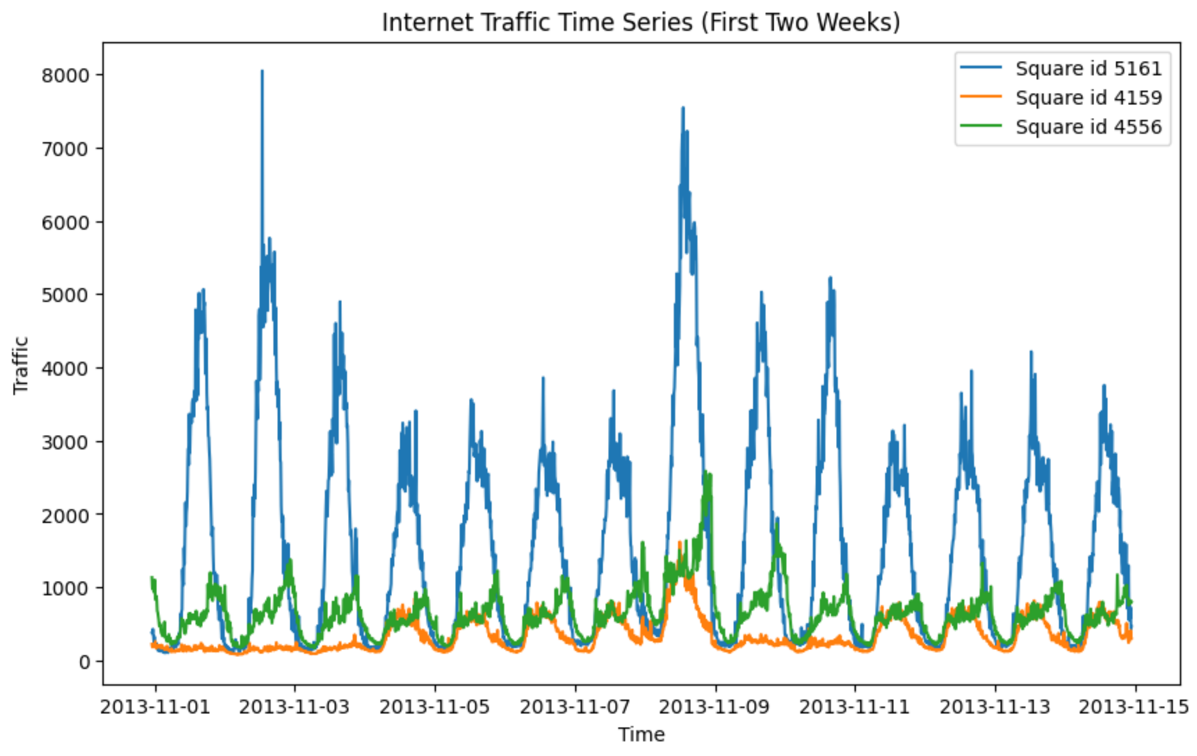


Figure 6

Now let’s geographically plot and research about the given 3 id’s

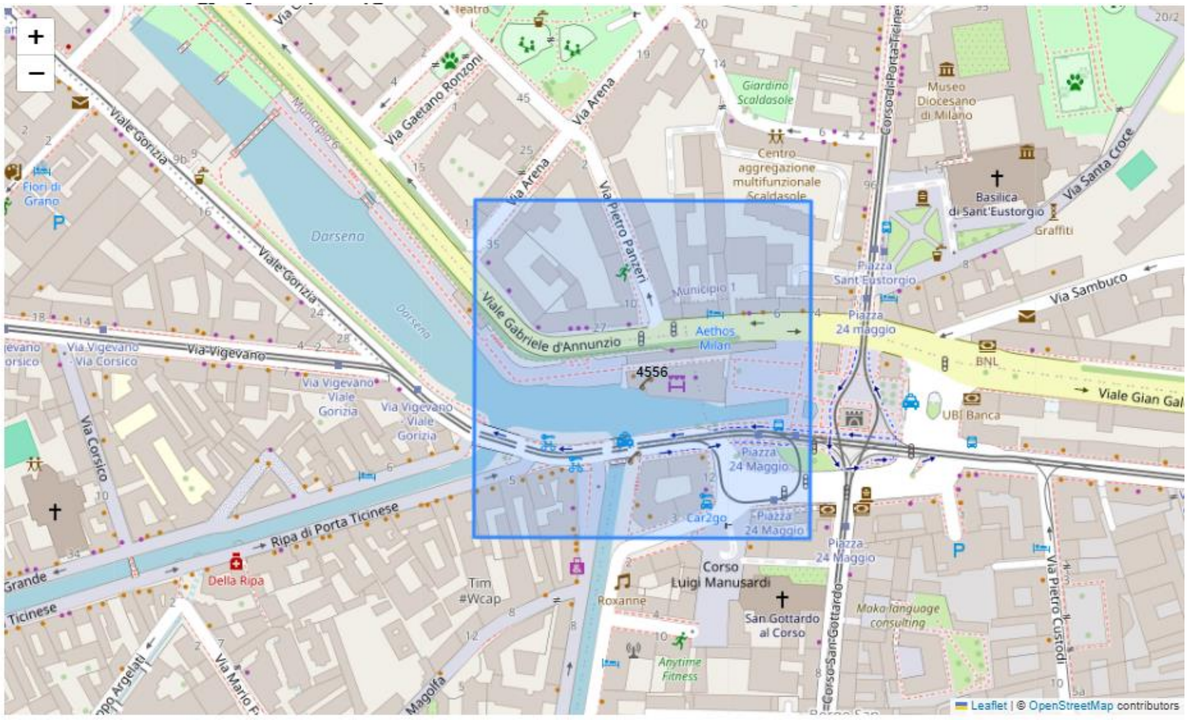

4556
Darsena
Viale Gabriele d'Annunzio
Piazza 24 Maggio
Leaflet | © OpenStreetMap contributors

Figure 7

```
centroid = filtered_gdf.geometry.unary_union.centroid
```

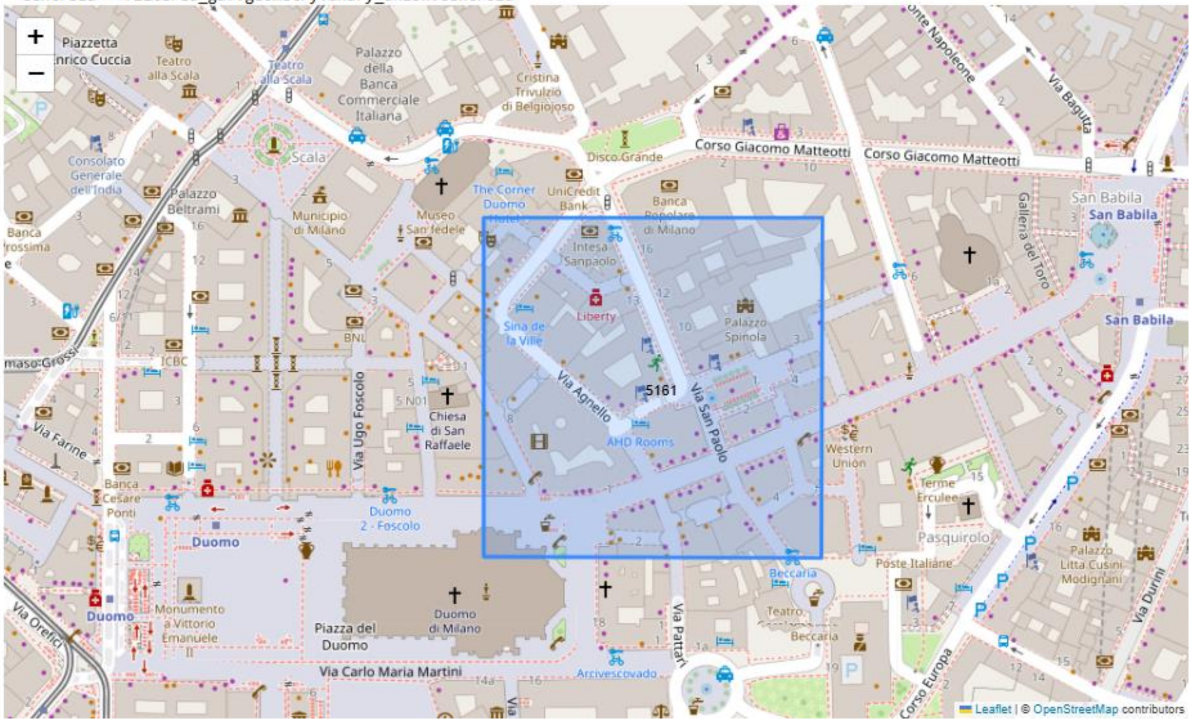


Figure 8

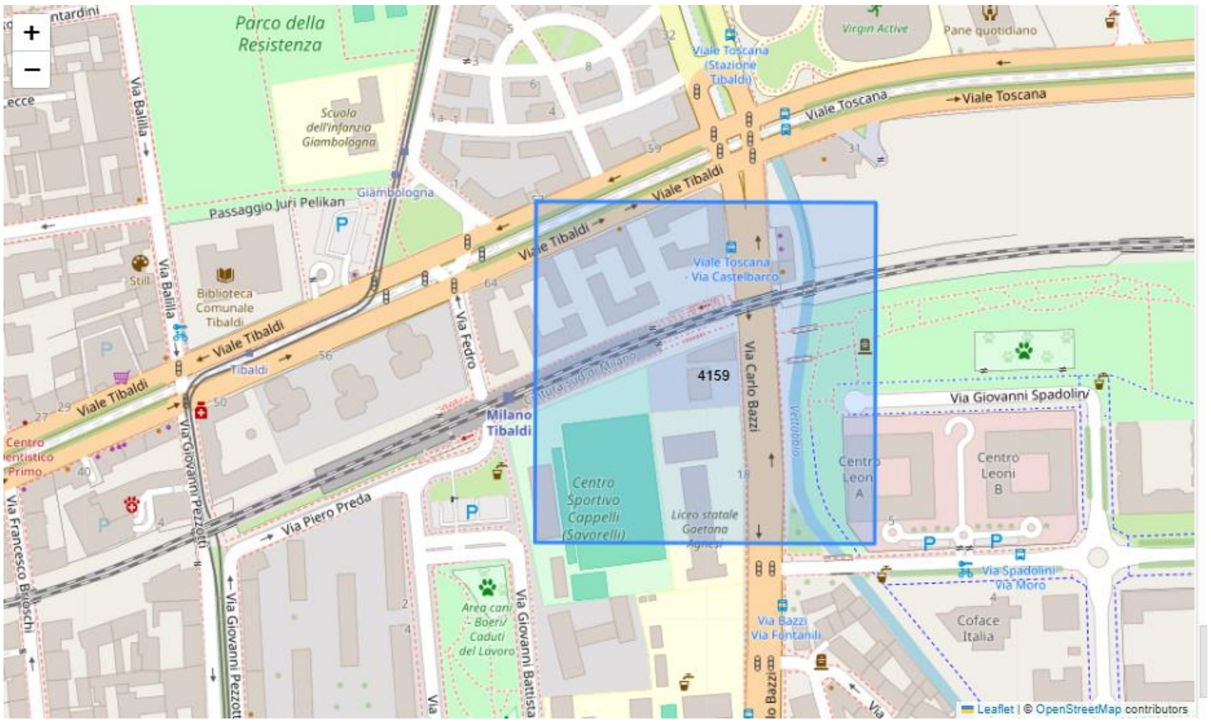


Figure 9

As the map shows the id 5161 is just near piazza Duomo which is the central cathedral area of an Italian city which explains the highest amount of internet traffic due to high foot fall. While the square grid 4159 consists of an area near a university sports facility which also depicts the low traffic on certain days for this area.

Now let's plot the heat map showcasing the internet traffic volume on the city.

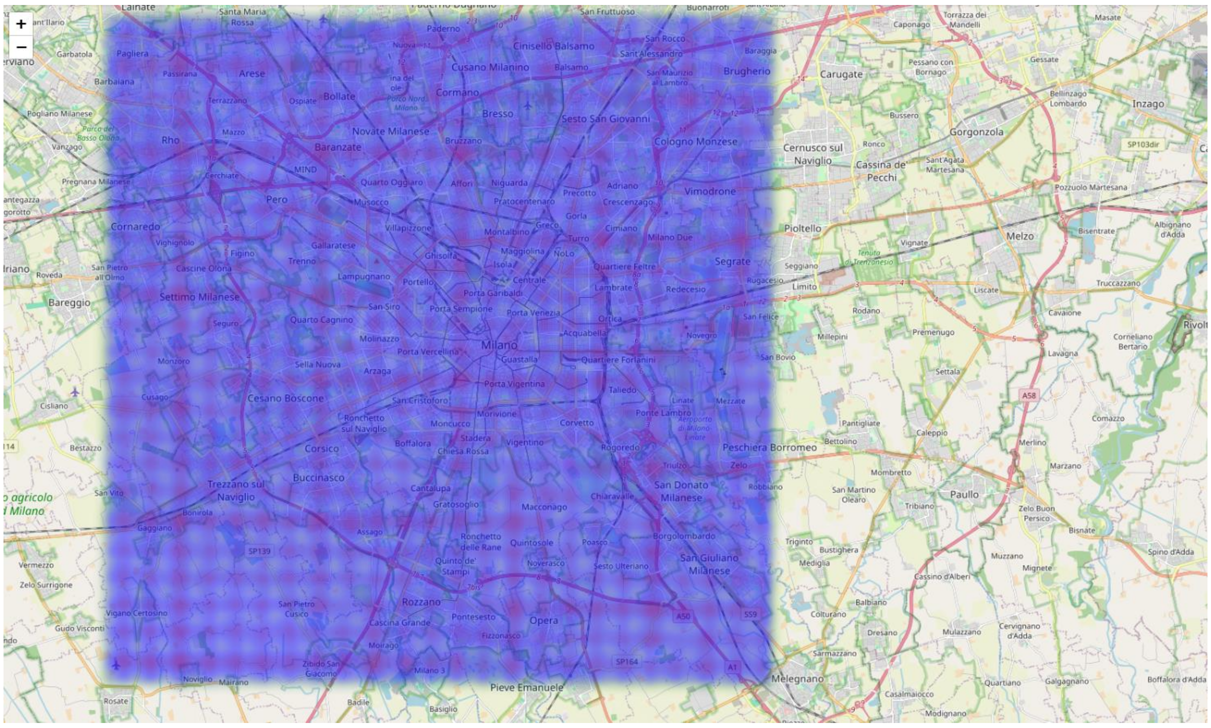


Figure 10

The densest red concentration can be seen in the centre of Milan and, thus, may indicate the peak level of internet activity. This is intuitive, as it would be expected that activity would always be at a peak within an urban centre, given the high density of populations, commercial centres, and the number of devices connected. Following outwards, traffic diminishes regarding the farthest suburbs and rural areas, indicated by both more blue and fewer red regions.

Let's analyse the top 10 square ids with the most traffic. The top 10 blocks are:

Table 1

```
Top 10 areas with the highest total traffic:
      square_id  internet_traffic
5160       5161      1.348949e+07
5058       5059      1.197819e+07
5258       5259      1.144482e+07
5060       5061      1.030696e+07
5257       5258      9.498652e+06
5158       5159      9.437481e+06
6063       6064      9.399213e+06
4854       4855      9.170492e+06
4855       4856      8.866177e+06
5261       5262      8.765110e+06
```

Let us plot these areas on the map

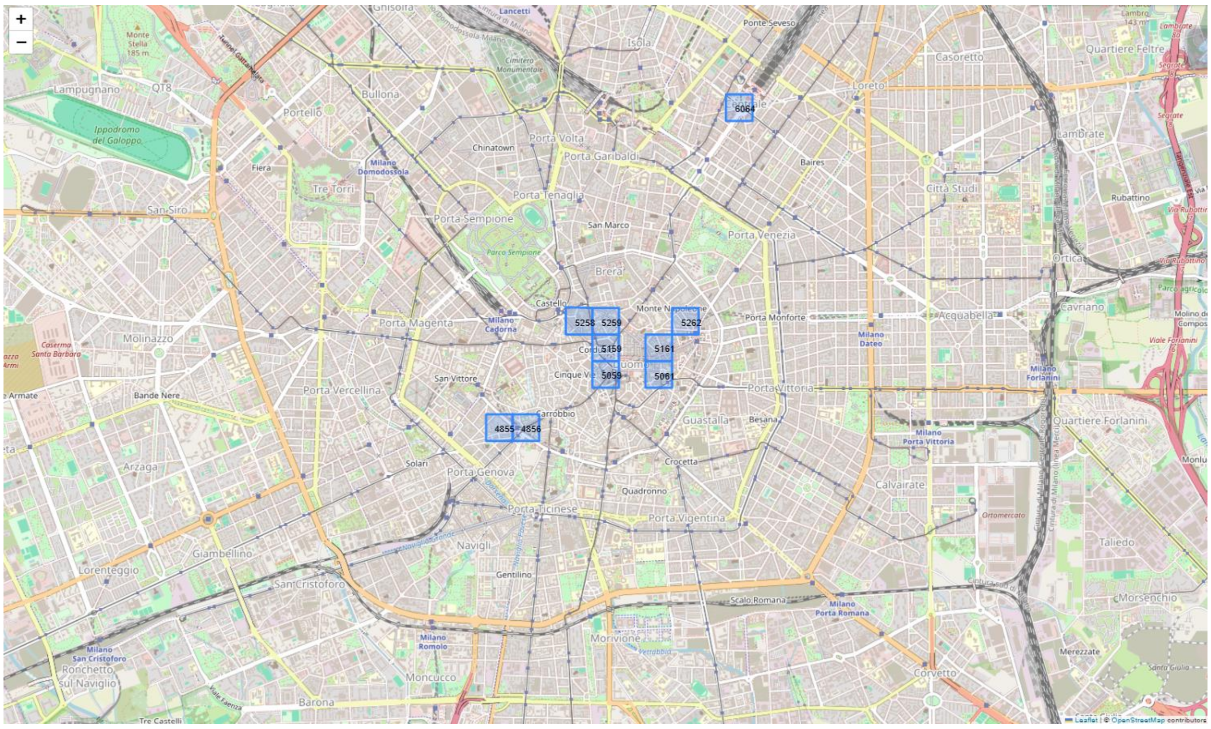


Figure 11

As expected, most of the high traffic areas are located in the centre of the city also ID 6064 which is not a centre area represents the Milan central railway station.

Now let us analyse the lowest traffic area and plot it on the map:

10 areas with the lowest total traffic:

Table 2

```
10 areas with the lowest total traffic:
      square_id  internet_traffic
5238       5239        226.486230
5338       5339        736.752410
2800       2801       2938.110593
5237       5238       3297.373102
111         112       3770.238921
1206       1207       5825.234425
5337       5338       7696.129817
2900       2901      12668.880545
5309       5310      13826.428724
5209       5210      13857.355342
```

As expected, this areas with the lowest traffic are in the outskirts of Milan with a very low population density

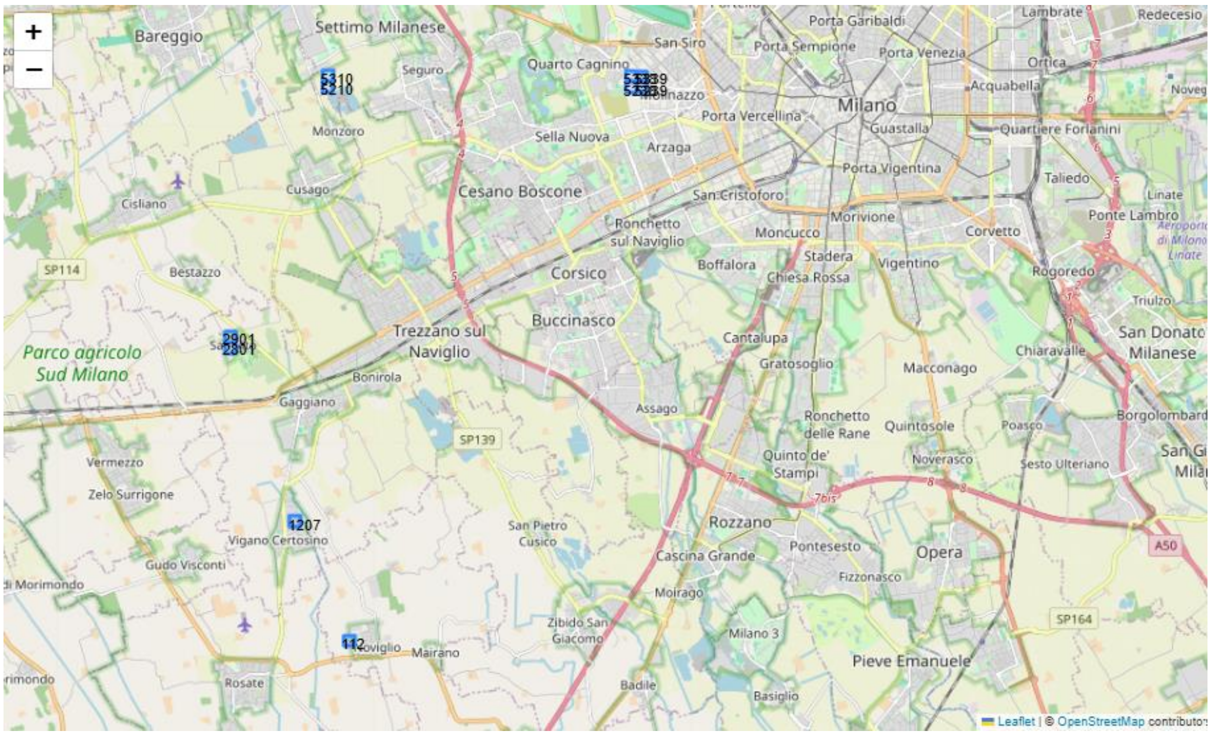


Figure 12

Seasonal analysis

Let us now focus on the traffic throughout the day while comparing the results on weekends and weekdays.

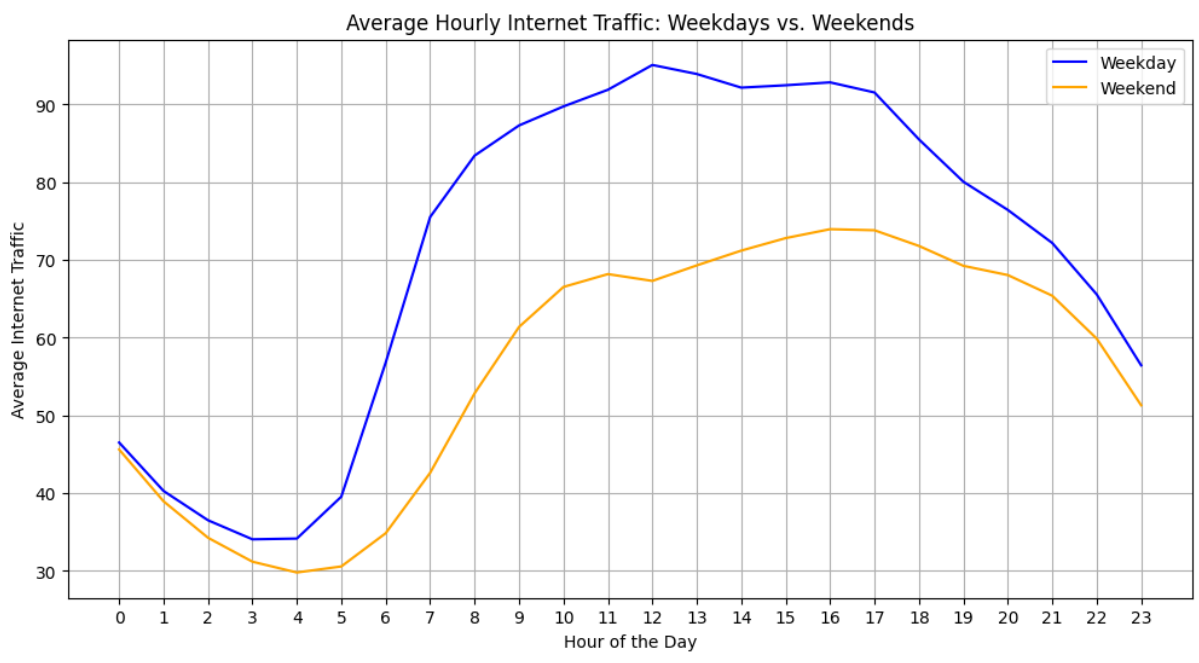


Figure 13

As the graph suggests in weekends the traffic is generally less then weekdays. With the peak usage in afternoon on weekdays and in evenings on weekends.

Let us analyse if this trend continues during the Christmas week it is 24th of December to 31 December

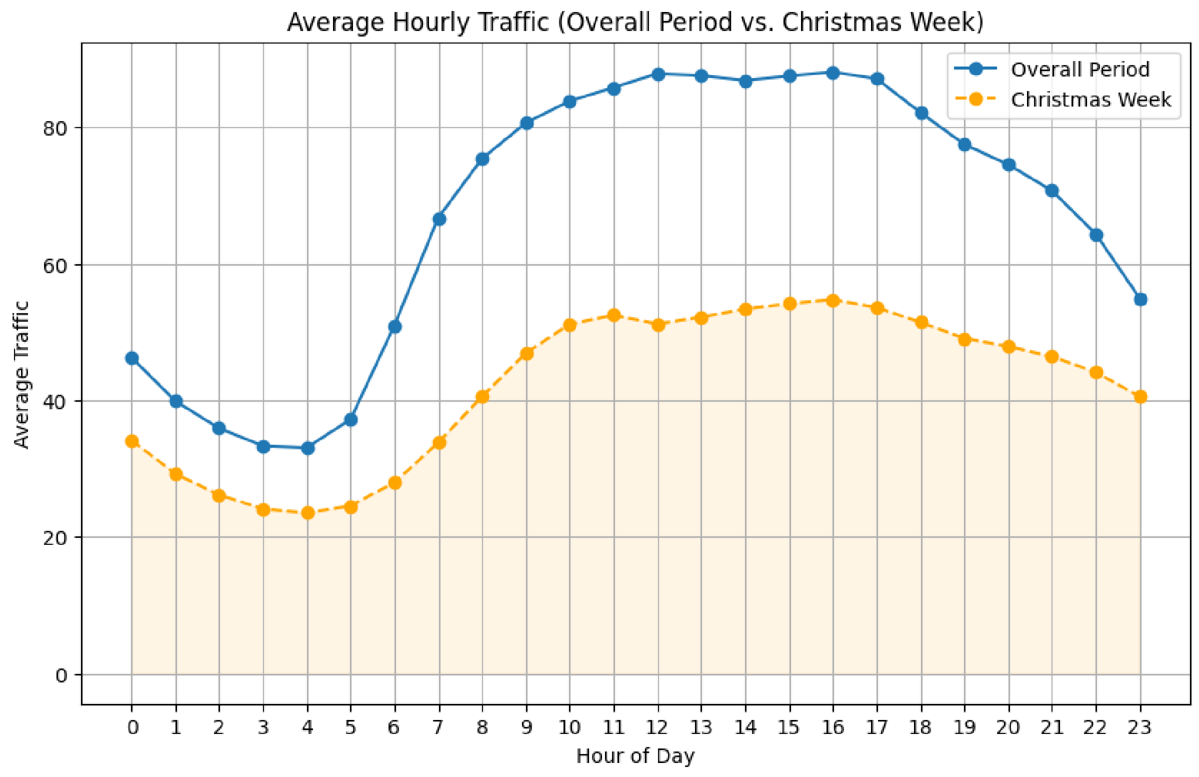


Figure 14

This graph shows Christmas week has low traffic rather compared to the regular days. This task provides insights into the distribution of traffic throughout the city, enabling the organization to plan resource allocation effectively and determine optimal times for maintenance.

Task 2:

In this part, we will build algorithms to estimate internet traffic for three specific areas: 5161, which had the maximum traffic over the two-month period, and 4159 and 4556 (square IDs). Our objective is to forecast internet traffic in these locations from December 16th to 22nd.

ARIMA model

The ARIMA (Autoregressive Integrated Moving Average) model is one of the most popular methods in time series analysis and forecasting. In short, it connects three main components together:

AR (Autoregressive): This component models the time series as a function of its past values. Alternatively, AR models utilize "lags" of the series. A lag is simply a prior observation in the data. Using traffic data from the last 3 days as a lag, we can use this information to predict today's internet traffic. This "regression" of the variable on its past values allows the model to capture trends or patterns in the series.

Integrated: In ARIMA, the data is differenced to be made stationary. That is, the mean, variance, and autocorrelation of the series should be equal across time. Differencing introduces stationarity in a time series that otherwise would be nonstationary-that is, that has trends or seasonality-it being very hard to forecast.

MA (Moving Average): The MA component employs past forecasting errors in order to update an expectation further. Error is the difference of actual with the predicted values at certain times and is believed to hold significant information in ARIMA. A moving average model calculates a weighted sum of past errors to assist the forecasting that is being done. For instance, if the model has regularly underpredicted the traffic on some of the days, it will modify its prediction accordingly in subsequent days.

Stationarity Analysis

To analyse the stationarity of a time series, we typically start by plotting the data to visually inspect for trends, seasonality, and other patterns. Visual inspection can provide valuable insights into the behaviour of the time series and help us determine if it appears to be stationary or not.

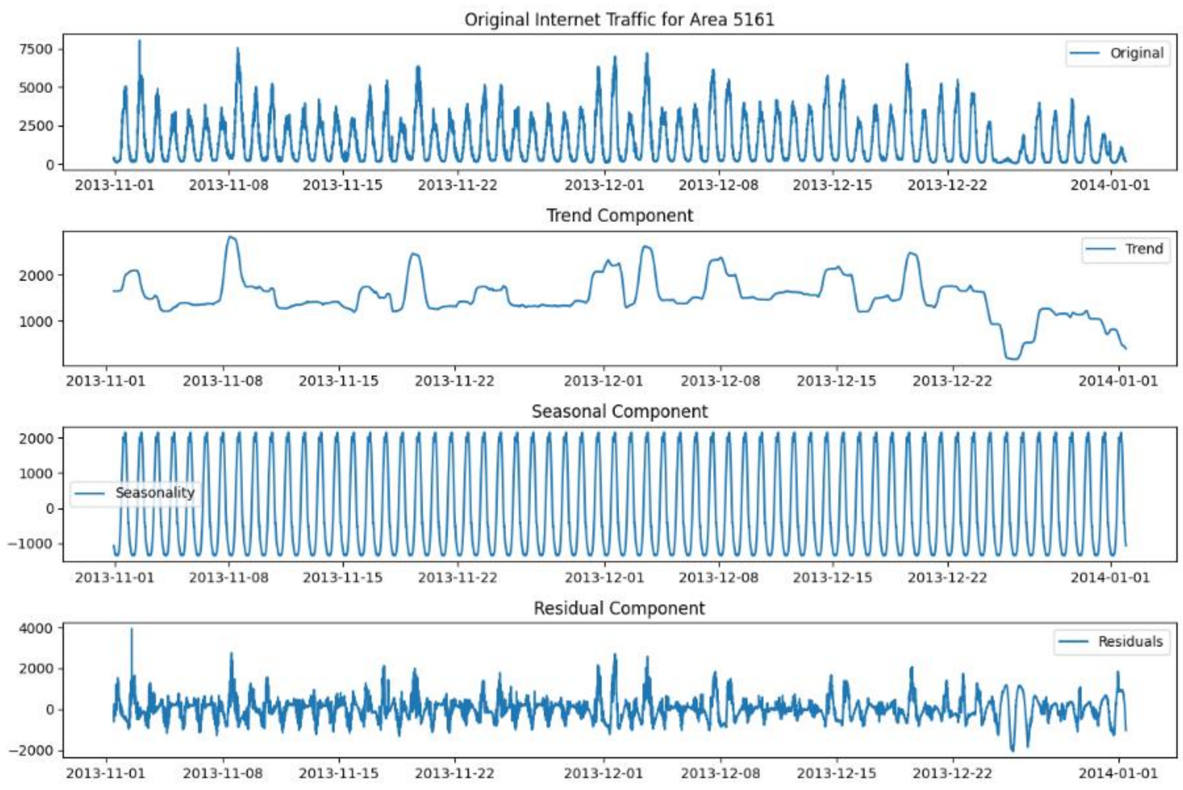


Figure 15

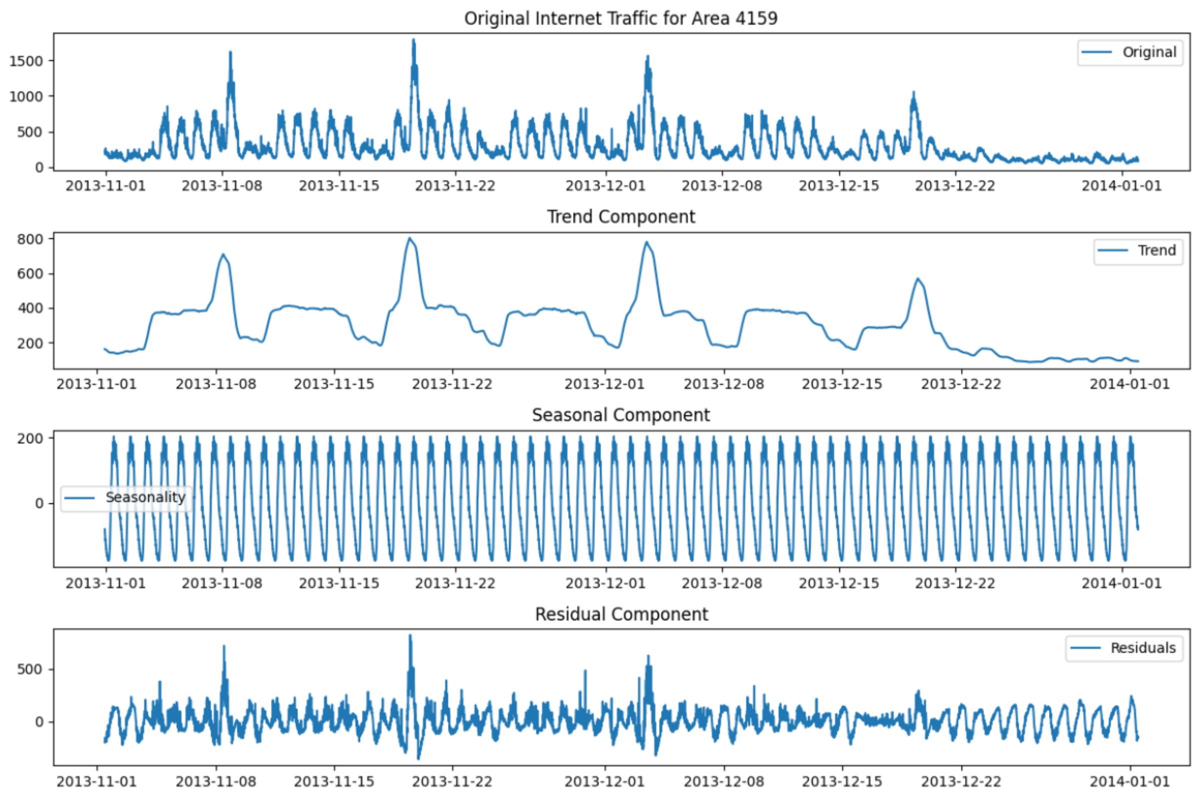


Figure 16

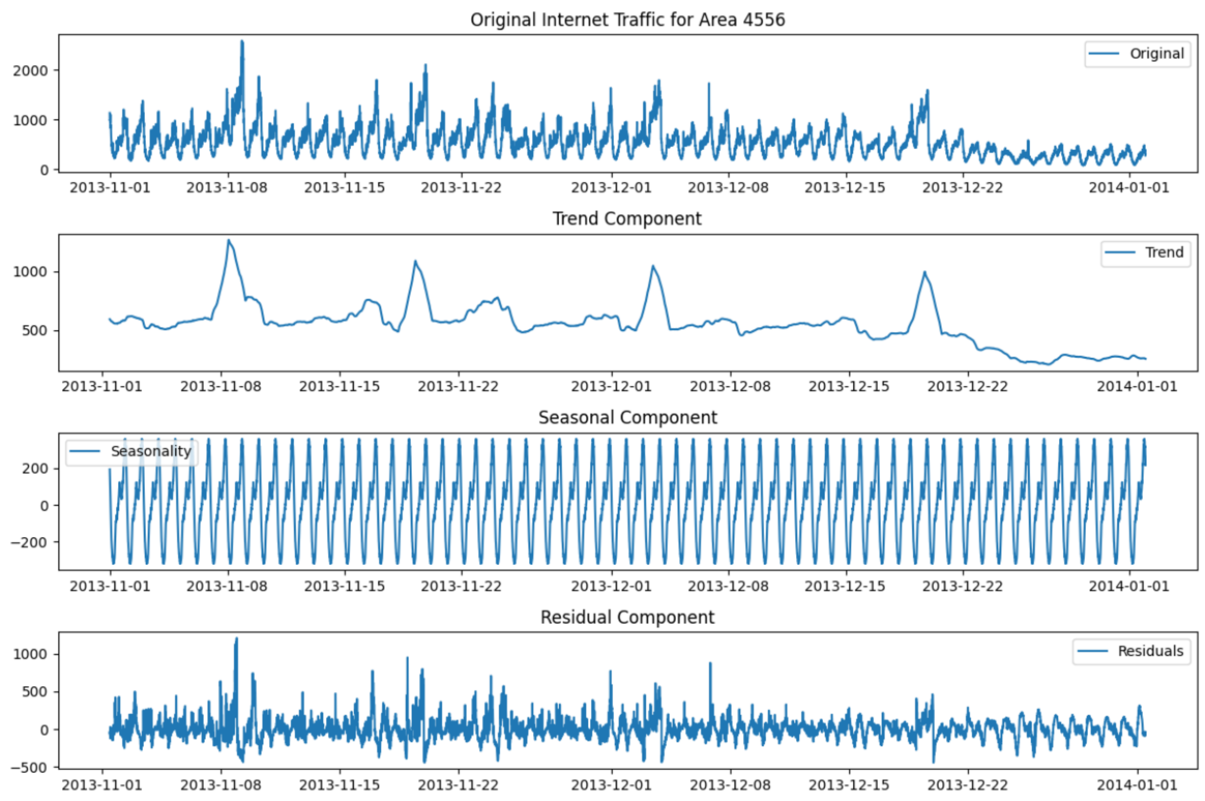
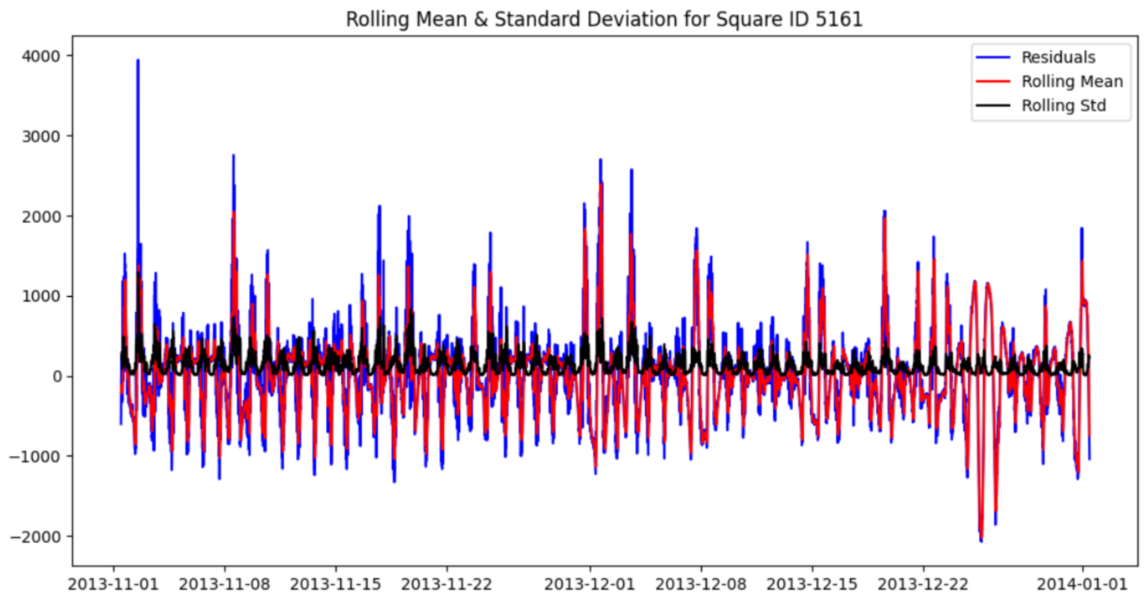


Figure 17

the decomposition suggests that internet traffic in the given areas exhibits a combination of a long-term upward trend and significant seasonal variations. The residual component indicates that there is still some unexplained variability in the data, which could be due to various factors.

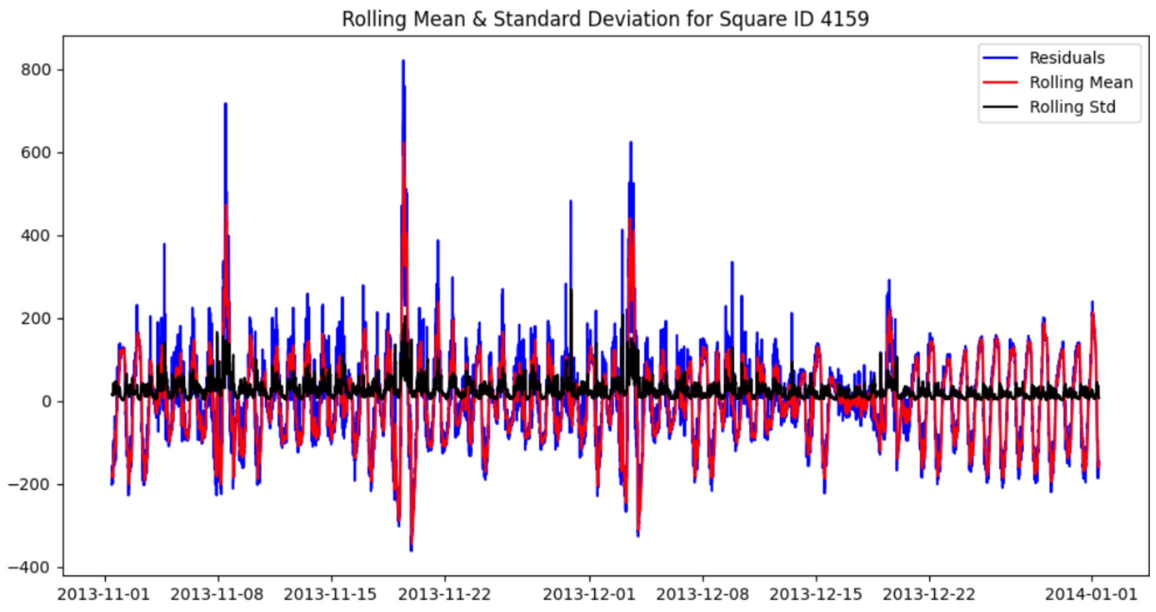
The seasonal pattern appears to be quite strong, suggesting that it is important to account for this when analysing the data.

We ran the Dickey-Fuller test on each area to check if the Internet traffic time series was stationary; this is a pre-condition for efficient modelling. The Dickey-Fuller test results showed higher confidence for areas 5161 and 4159 but only moderate confidence for the area 4556. Stationarity has occurred after trending out the seasonal. SARIMA, which models seasonal trends, will apply.



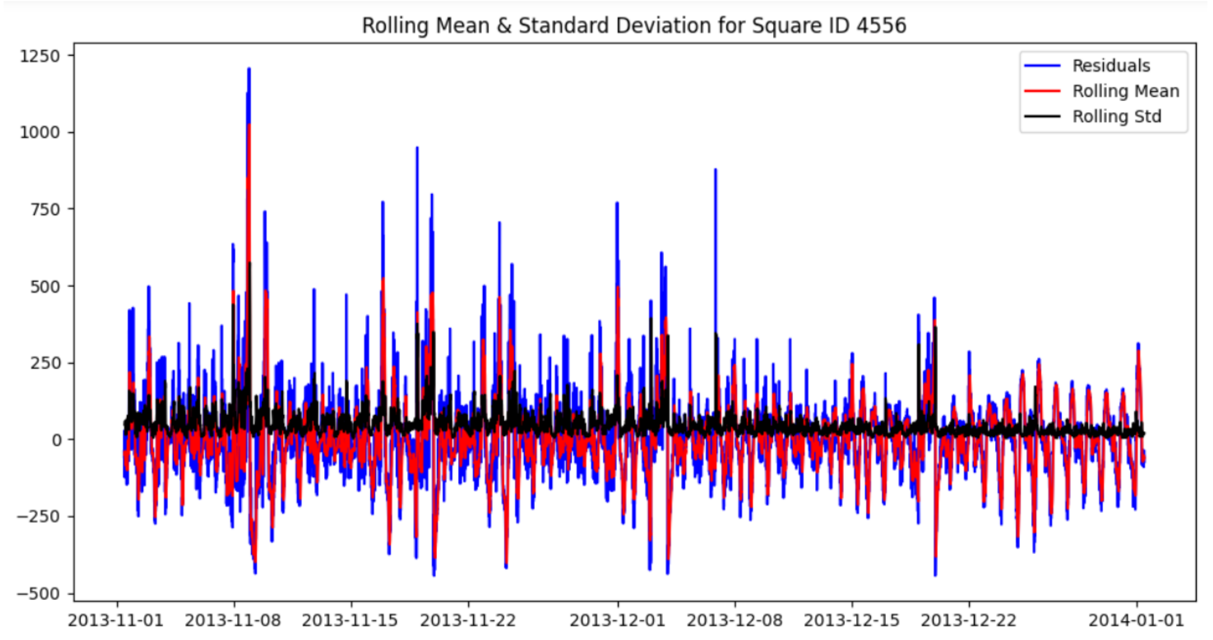
```
Results of Dickey-Fuller Test for Square ID 5161:
Test Statistic                -1.821704e+01
p-value                        2.384934e-30
#Lags Used                     3.700000e+01
Number of Observations Used    8.746000e+03
Critical Value 1%             -3.431098e+00
Critical Value 5%             -2.861871e+00
Critical Value 10%            -2.566946e+00
dtype: float64
```

Figure 18



```
Results of Dickey-Fuller Test for Square ID 4159:
Test Statistic                  -18.974011
p-value                           0.000000
#Lags Used                       37.000000
Number of Observations Used    8746.000000
Critical Value 1%                -3.431098
Critical Value 5%                -2.861871
Critical Value 10%               -2.566946
dtype: float64
```

Figure 19



```
Results of Dickey-Fuller Test for Square ID 4556:
Test Statistic                 -1.696755e+01
p-value                         9.210237e-30
#Lags Used                      3.700000e+01
Number of Observations Used     8.746000e+03
Critical Value 1%              -3.431098e+00
Critical Value 5%              -2.861871e+00
Critical Value 10%             -2.566946e+00
dtype: float64
```

Figure 20

SARIMA model

We found our time series to be very clear with consistent seasonal patterns all along. So we used a SARIMAX model for prediction.

SARIMAX is short for Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors. It's essentially an extension of the ARIMA model but includes seasonality and can embrace other exogenous regressors as well (the "exogenous" part). The SARIMAX model captures a lot of common structures found in time series data such as the ARIMA model; however, its strength lies in its ability to handle seasonality. The "S" in SARIMAX is for "seasonal," which is that the model can remember and forecast with the help of seasonality from information of a time series. For instance, if we predict the Internet to be more congested on weekends than during the weekdays, this has been a sort of seasonality created at the weekly level, which SARIMAX is going to effectively use in its models to predict in consequence.

Determining (P, D, Q, s)

First plot the ACF and PACF plots for each square ID:

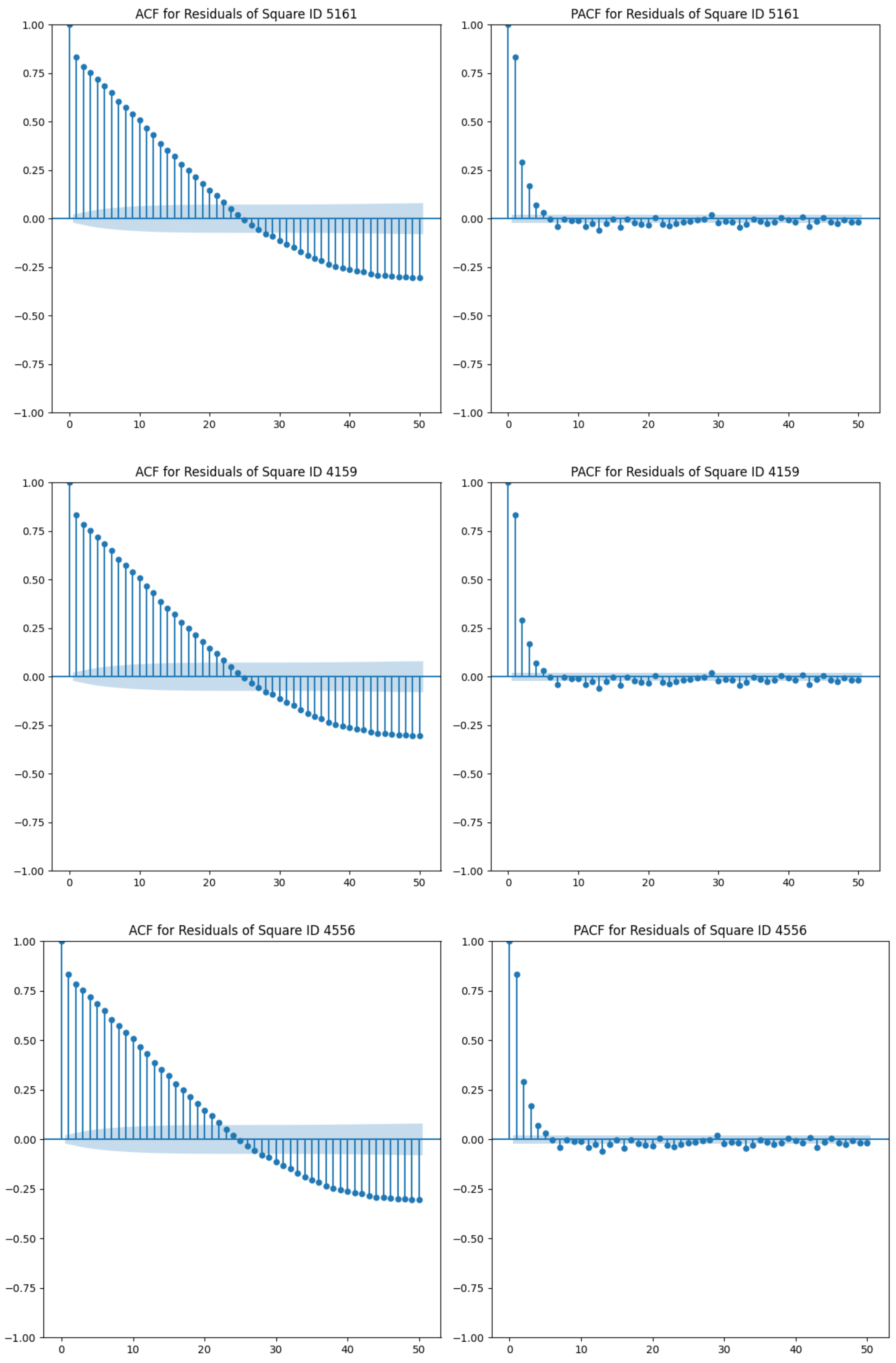

ACF for Residuals of Square ID 5161
PACF for Residuals of Square ID 5161
ACF for Residuals of Square ID 4159
PACF for Residuals of Square ID 4159
ACF for Residuals of Square ID 4556
PACF for Residuals of Square ID 4556
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50

Figure 21

The plots of ACF and PACF give a direction to the selection of the order of the P, and Q terms. Peaks in the ACF and PACF are where the number of lags will be included.

PACF plot Identify the lag at which the first major spike appears. That lag determines p, the order of the AR component. all plots for the three ID's point to a major spike at lag 1, so p=1.

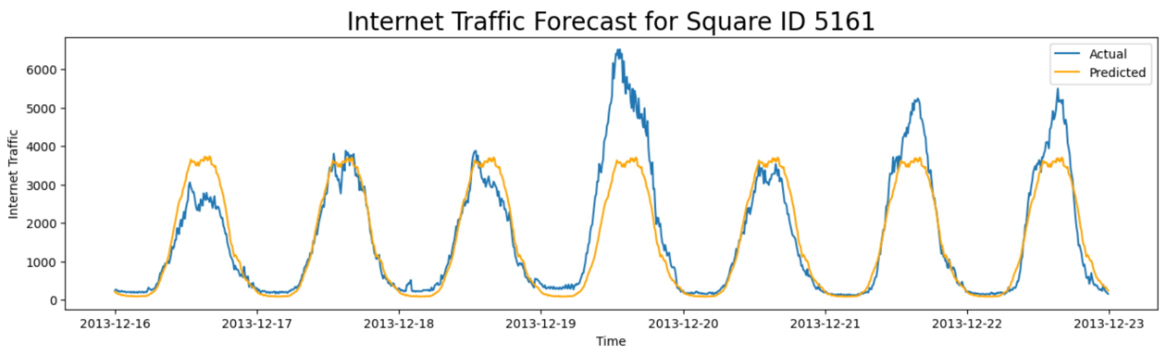
And the ACF plot Identify the lag at which the first significant spike occurs. This lag indicates the order of the MA component (q). The plots for all three ID's suggest that there's a significant spike at lag 1, suggesting q=1.

Dickey-Fuller test, which was already nonstationary in both seasonal and non-seasonal components. Thus, terms of difference were included for that reason with D=1.

The seasonal period is set at 144 since data follows the daily seasonality pattern. Observations are taken every 10 minutes. So, times 144×10 minutes will cover 24 hours, capturing the daily internet traffic cycle.

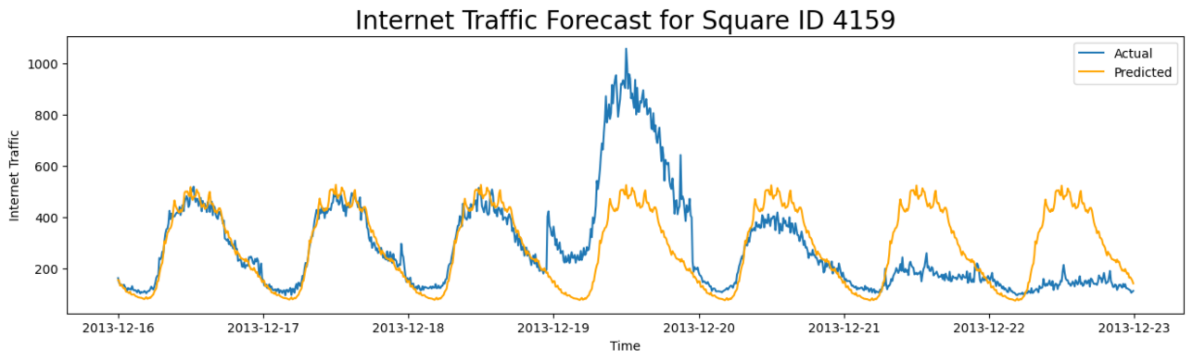
Model implementation and results

For each area, predictions for the one-week period were generated and evaluated using Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).



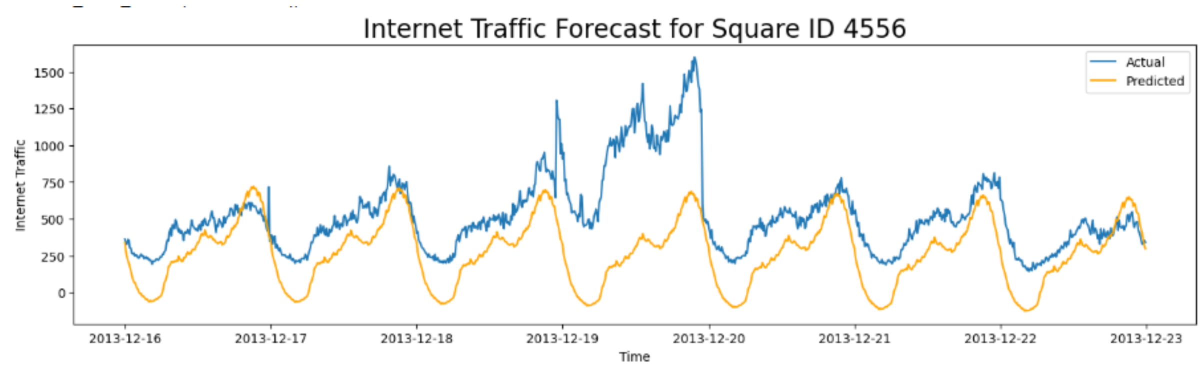
```
SARIMA MAE for Square ID 5161: 383.62895293370326
SARIMA MAPE for Square ID 5161: 30.43%
```

Figure 22



```
SARIMA MAE for Square ID 4159: 102.44356106362547
SARIMA MAPE for Square ID 4159: 43.15%
```

Figure 23



```
SARIMA MAE for Square ID 4556: 265.5754344612696
SARIMA MAPE for Square ID 4556: 56.92%
```

Figure 24

K-Nearest Neighbours (KNN)

Besides SARIMA, we have implemented the KNN regression model, providing a way to predict internet traffic across the specific areas of Milan. KNN is a non-parametric technique that uses similar past instances within a specified "neighbourhood" to predict future values, which can especially be helpful in describing localised patterns in the data without any assumptions made with respect to the underlying distribution.

To implement KNN, we pre-processed our data by converting every observation into feature vectors. These vectors carry time-based attributes like hour and day of the week to capture cyclical patterns and take advantage of temporal trends in traffic data. Normalization is applied to ensure that all features contribute equally to distance calculation such that the accuracy of the model may be enhanced.

The most important hyperparameter of KNN is K-number of neighbours. We tested a range of Ks to find the optimal with cross-validation on a validation set that minimizes prediction errors on a validation dataset. We have also tried various distance metrics: Euclidean distance and Manhattan distance, to observe which, one would suit the characteristics of our dataset better.

```
Optimal K for Square ID 5161: 7
MAE: 107.52301039662636
MAPE: 9.832140303714231%
```

```
<ipython-input-6-d130c873073b>:30: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future vers
ion, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `se
r.iloc[pos]`
  y.append(data[i])  # Target: the next time step
```

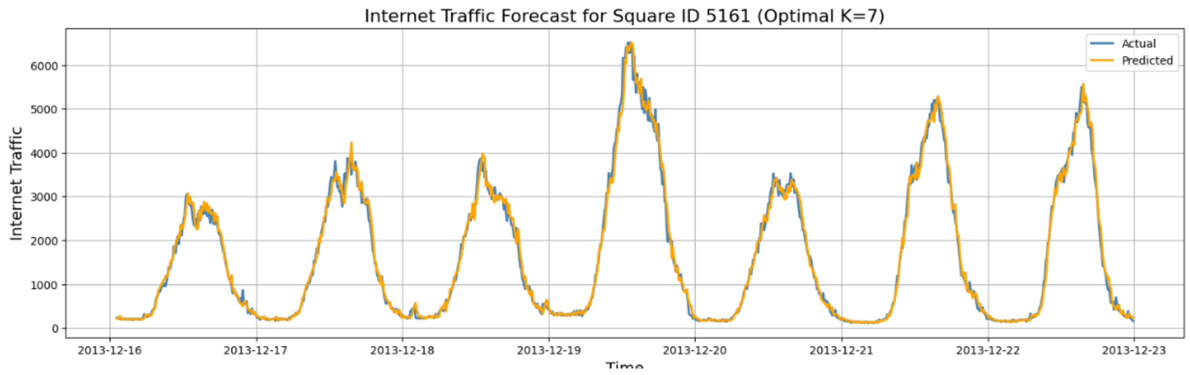


Figure 25

```
Optimal K for Square ID 4159: 4
MAE: 22.66091631282308
MAPE: 8.220850405127985%
```

```
<ipython-input-6-d130c873073b>:30: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future vers
ion, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `se
r.iloc[pos]`
  y.append(data[i])  # Target: the next time step
```

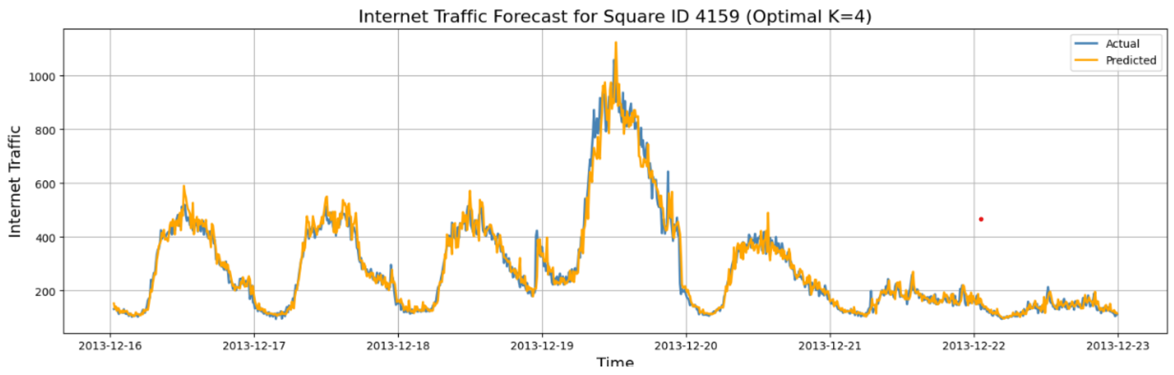


Figure 26

```
Optimal K for Square ID 4556: 8
MAE: 41.10965258712518
MAPE: 8.298329884780486%
```

```
<ipython-input-6-d130c873073b>:30: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future vers
ion, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `se
r.iloc[pos]`
  y.append(data[i])  # Target: the next time step
```

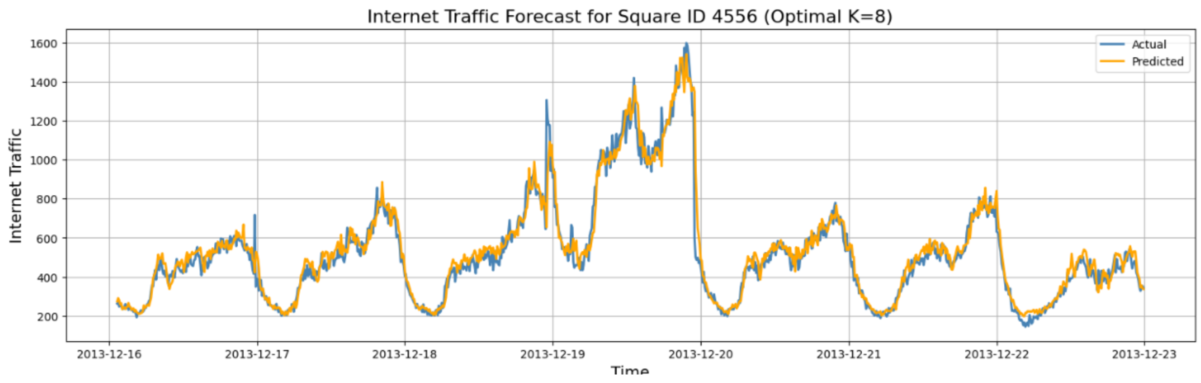


Figure 27

For time series with strong seasonal patterns, SARIMA outperformed; for those with traffic trendable based on nearby observations, as in recent traffic levels predicting well, going forward, KNN proved effective.

It really put forward the strength of KNN as a flexible, instance-based learner that could self-adjust to the conditions of traffic flow without the need to tune models heavily. That is another promising area for future work through combining KNN with SARIMA-a robust hybrid model capturing seasonal and localized variations.