Fayoum University
Engineering of Faculty
Electrical Engineering Department

B. Eng. Final Year Project

Mobile Network Traffic Forecasting

By:

Alaa Muhammad Sedeeq

Tarek Muhammad Abdullah

Abdullah Muhammad Amin

Ibrahim Sayed Muhammad

Supervised By:

Dr\Tarek Said

Fayoum University

Engineering of Faculty

Electrical Engineering Department

Sponsored by:

B. Eng. Final Year Project

Mobile Network Traffic Forecasting

By:

Alaa Muhammad Sedeeq

Tarek Muhammad Abdallah

Abdullah Muhammad Amin

Ibrahim Sayed Muhammad

Supervised By:

Dr\Tarek Said

Date of examination

July-2022

DEDICATION

We dedicate this to the All-Powerful God for His Supremacy and to the Redeemer of Our Souls, as well as to our parents who have given us much that we cannot adequately acknowledge or thank them for and who have helped us in every way to work toward this moment because they have faith in us and our goals.

ACKNOWLEDGMENT

This project would not have been possible without the support of many people who were truly helpful and supportive. Many thanks to Dr. Tarek Said, the professor at Fayoum university who accepted the challenge with us, and Dr. Kareem Abdullah, Head of AI and Analytics team at Ericsson MENA who was a great mentor, and helped us in every step in this project, our advisers: Eng. Mariam El Sobky, Eng. Mayar Hefny from Ericsson, who assisted in clearing up the issue, Dr. Fatma Mazen The professor at Fayoum Engineering faculty to be truly supportive.

We would also like to express our gratitude to the university of Fayoum College of Engineering for all of its assistance and the many new things we have learned over the past five years.

We have to mention our thanks to Eng. Salsabel Adel Form Nile university, Eng. Tarek Nasr from Nokia, and also Eng. Abdullah Ashmawi from Dua corporate to their advices and great effort with us.

And for whom began the journey with us, Eng. Ibrahim El Diftar the Head of AI at Ericsson to give us the opportunity to work with Ericsson Experts.

Finally, we would want to express our gratitude to our parents and the many friends who supported us throughout this drawn-out procedure. It has been a wonderful opportunity to get a lot of real-world experience and understanding about how to plan and analyze projects. We want to express our gratitude to everyone who makes it for students like us. Our sincere gratitude goes out in particular to the doctors for their efforts in supplying us with the relevant information and paving the way for the students to implement all educational sessions in real-time project creation and analysis. We would like to convey our sincere gratitude to the supervisor of our graduation project for his tolerance and leadership during the semester.

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in *Electrical Engineering* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Alaa Muhammad Sedeeq

Registration No.: 4822

Date: Sun, 17 July 2022.

Signed: Abdullah Muhammad Amin

Registration No.: 4821

Date: Sun, 17 July 2022.

Signed: Ibrahim Sayed Muhammad

Registration No.: 4803

Date: Sun, 17 July 2022.

Signed: Tarek Muhammad Abdullah

Registration No.: 4816

Date: Sun, 17 July 2022.

ABSTRACT

Predictive analysis on mobile network traffic is becoming a fundamental part for the next generation cellular network. Proactively knowing the user demands, allows the system for an optimal resource allocation. Due to the dynamic nature of mobile users, mobile communication systems must adapt to the temporally and spatially changing mobile network traffic in order to deliver high-quality service. Since these changes are not entirely random, it is possible to anticipate the future status of network traffic by extracting the predictable component and patterns from the observed network traffic. Such prediction can be used for a number of proactive network management techniques, such as load balancing, coordinated beam management, and beam activation and deactivation. In this project, we design a system for the traffic forecasting using many machines learning, deep learning, and statistical algorithms, with accuracy reaches 91% for 7 days forecasts.

Table of Contents:

LIST OF Tables

LIST OF FIGURES

Figures of Chapter 1:

Figures of chapter 2:

Figures for chapter 3:

List of Abbreviations

Abbreviation	Definition
LTE	Long-Term Evolution
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
SC- FDMA	Carrier Frequency Division Multiple Access
FDD	Frequency Division Duplex
TDD	Time Division Duplex
BH	Busy Hour
QoS	Quality Of Service
RLB	Radio link budge
MAPL	Maximum Allowable Path Loss
SINR	Signal to Interference plus Noise Ratio
TxPower	Transmitter Power
H_{BS}	Indicates the height of the base station antenna
H_{SS}	Indicates the height of the terminal antenna
T site	Actual throughput
T cell	Cell throughput

Q	System load
NBR	No of Resource Block
CCH	Control Channel Ratio
VoIP	Voice Over Internet Protocol
FTP	File Transfer Protocol
SON	Self-Organizing network
T	Trend Signal
C	Cyclic Signal
R	Random Signal
ARCH	Autoregressive Conditional Heteroskedasticity
VAR	Vector Autoregressive
LSTM	Long short-term memory
GAIN	Generative Adversarial Imputation Networks
Conv-GAIN	Convolutional Generative Adversarial Imputation Networks
FBProphet	Facebook Prophet
STL	Seasonal and Trend decomposition using Loess
SARIMA	Seasonal Autoregressive Integrated Moving Average
CNN	Convolutional Neural Networks
CDRs	Call Detailed Records

RBS	Radio Base Station
SMS	Short Message Service
SEATS	Seasonal Extraction in ARIMA Time Series
ARIMA	Autoregressive Integrated Moving Average
COV	Covariance
ACF	AutoCovariance Function
PACF	Partial AutoCovariance Function
KPSS	Kwiatkowski–Phillips–Schmidt–Shin
H0	The Null Hypothesis
H1	The Alternative Hypothesis
ADF	Augmented Dickey–Fuller test
CSS	Cross-Site Scripting
DDOS	Distributed Denial-Of-Service
KPIs	Key performance indicators
MAE	Mean Absolute Error
PCA	Principal Component Analysis
MF	Matrix Factorization
RNN	Recurrent Neural Network
MCAR	Missing Completely at Random

MAR	Missing At Random
MNAR	Missing Not at Random
LVO	Last valid observation
NVO	Next valid observation
KNN	K-Nearest Neighbor
MICE	Multiple Imputation Using Chained Equations
Var	Variable
GAN	Generative Adversarial Networks
D	Discriminator
G	Generator
ETS	Exponential Smoothing
MLP	Multilayer Perceptron
ConvLSTM	Convolutional LSTM
MAPE	Mean Absolute Percentage Error
AR	Autoregressive
MA	Moving Average
SVR	Support Vector Machine
LR	Linear Regression

Chapter One: LTE Planning Overview

1|LTE Planning Overview

1.1 Introduction:

4G LTE networks. Then, we outline the planning steps of LTE planning. UMTS Long Term Evolution (LTE) is one of the choices for next generation broadband wireless networks. LTE network are aiming at several target. LTE supports data rates of over 100Mbps in the downlink and more than 50 Mbps in the uplink. LTE reduces packet latency by reducing number of nodes in network. It uses OFDM – based technology resulting in higher spectral efficiency and increased capacity. By improving data rates. LTE reduce latency and offers a rich multimedia user experience. LTE uses Orthogonal Frequency Division Multiple Access (OFDMA) technology for downlink transmission. It uses Single Carrier Frequency Division Multiple Access (SC- FDMA) technology for uplink transmission. LTE supports both TDD (Time Division Duplex) and FDD (Frequency Division Duplex) modes operation.

Radio network planning is a very necessary step for a wireless communication technology. The LTE radio network planning work just like other cellular technologies, initial stage plans is normally guided by various industries and vendors at their own discretion. They aren't likely to disclose their advancements and findings. That makes the job even more challenging. For any new cellular technology there are hundreds of its RF parameters, they are need tuning process with a view to find out optimum value. But this process is time consuming and very costly. So, by some extensive simulation methods cost can be greatly minimized. All these aim at proper radio network planning of LTE.

1.2 Radio Network Planning Process:

Network planning is a complicated process consisting of several phases. The final target for the network planning process is to define the network design, which is then built as a cellular network. The network design can be an extension of the existing LTE network or a new network to be launched. Environmental factors also greatly affect network planning.

1.3 Network planning process steps

The radio network planning process is divided into five main steps, where four steps are prelaunch and the last one comes after the network has been launched. The flowchart for the network planning process is shown in Figure 1. The five main steps are: preplanning, planning, detailed planning, acceptance and optimization.

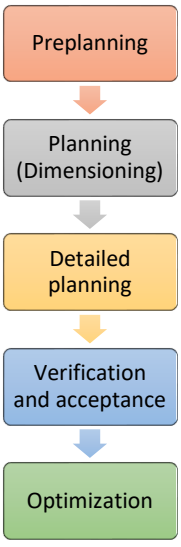


Figure 1 1 Network planning process steps

1.3.1 Preplanning:

The preplanning phase covers the assignments and preparation before the actual network planning is started. As in any other business it is an advantage to be aware of the current market situation and competitors. The network planning criteria are agreed with the customer. And collecting sufficient information and data about the area to be planned, for example, this is some information that we obtained from some sites and it is related to the city of Milan

Figure 1 2 Milan geography

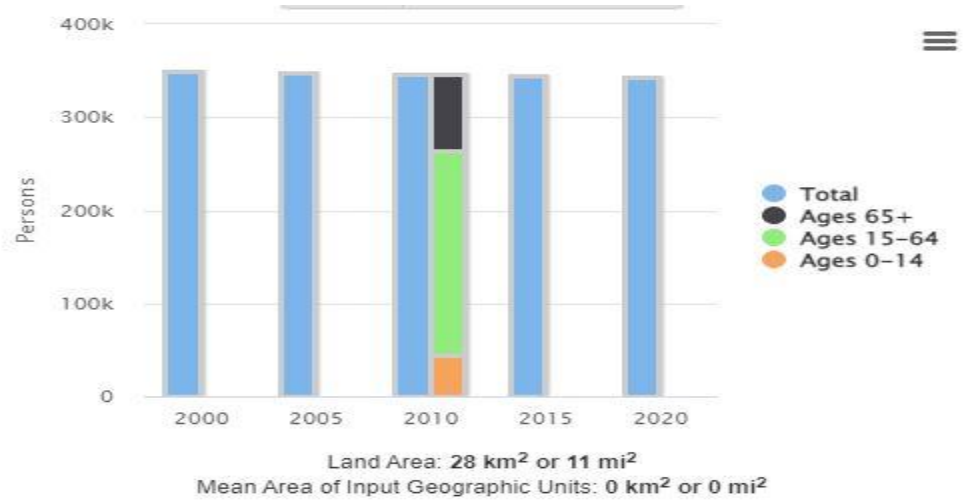


Figure 1 3 Milan Population density

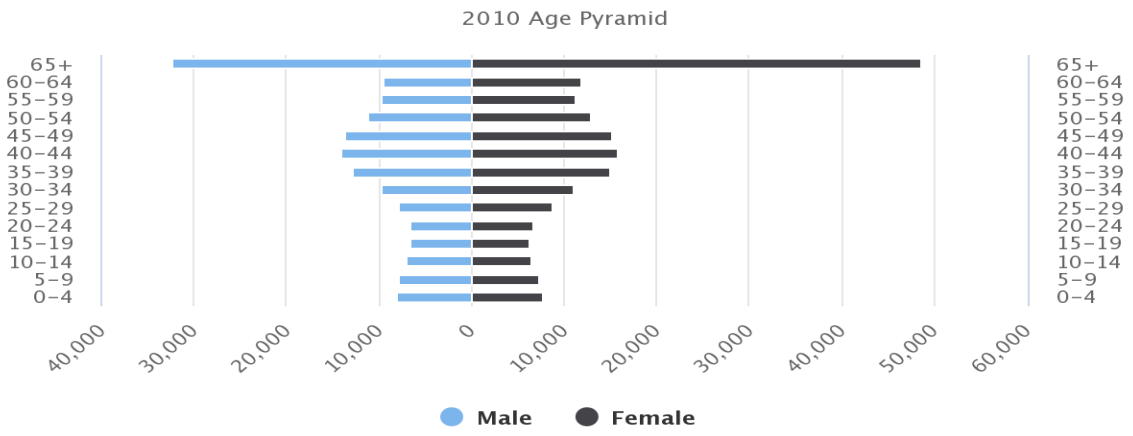


Figure 1 4 Milan Age pyramid

1.3.2 LTE Access Network Dimensioning:

The target of the LTE access network dimensioning is to estimate the required site density and site configurations for the area of interest. Initial LTE access network planning activities include radio link budget and coverage analysis, cell capacity estimation, estimation of the amount of eNB. The calculation of the sites number based on the coverage and the capacity.

If we have N sites from capacity planning and M sites from coverage planning which will be used, compare between N coverage and M capacity and choose the larger one take N capacity (output capacity dimension) and consider it as input to the coverage dimension algorithm. Then change your assumption (such as site initial configuration) Take output number of sites from coverage dimension (M coverage) and enter it to the capacity dimension algorithm as an input

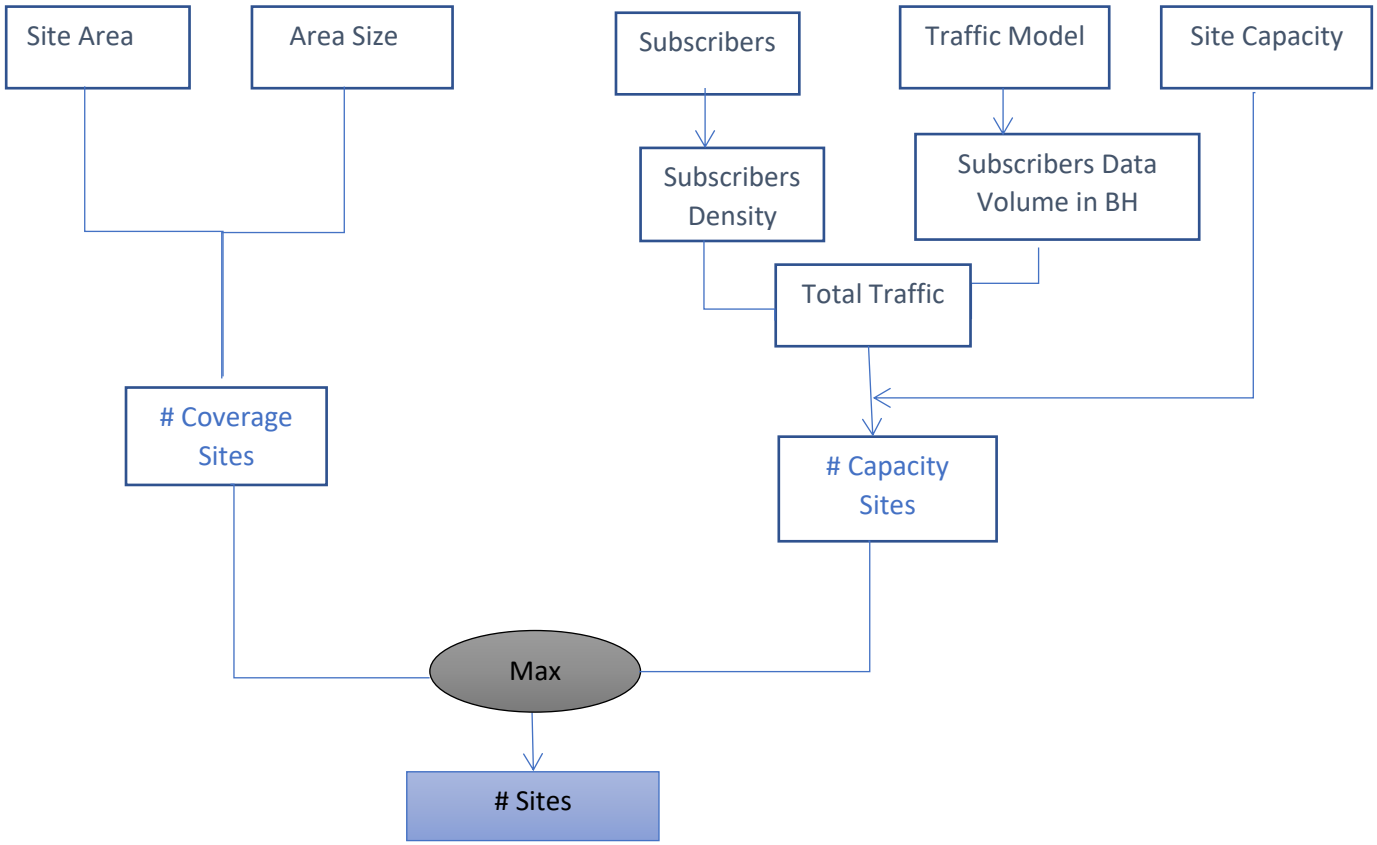


Figure 1 5 LTE Dimensioning

- Inputs of LTE Dimensioning

One of the basic objectives of this work is to clearly differentiate between LTE dimensioning inputs and outputs. This section discusses all the LTE dimensioning inputs used in the development of methods and models for LTE dimensioning. LTE dimension inputs can be broadly divided into three categories: Traffic, coverage and capacity-related inputs.

Traffic related inputs include average cell throughput, number of subscribers and demand traffic for each user in BH (Busy Hour). These parameters are the customer requirements to provide a certain level of service to its users. These inputs directly translate into (QoS) parameters. Besides cell edge performance criterion is used in the dimensioning tool to determine the cell radius and thus the site count. Three methods are employed to determine the cell edge. These include user defined maximum throughput at the cell edge, maximum coverage with respect to lowest MCS (giving the minimum possible site count) and predefined cell radius. Radio link budge (RLB) is of central importance to coverage planning

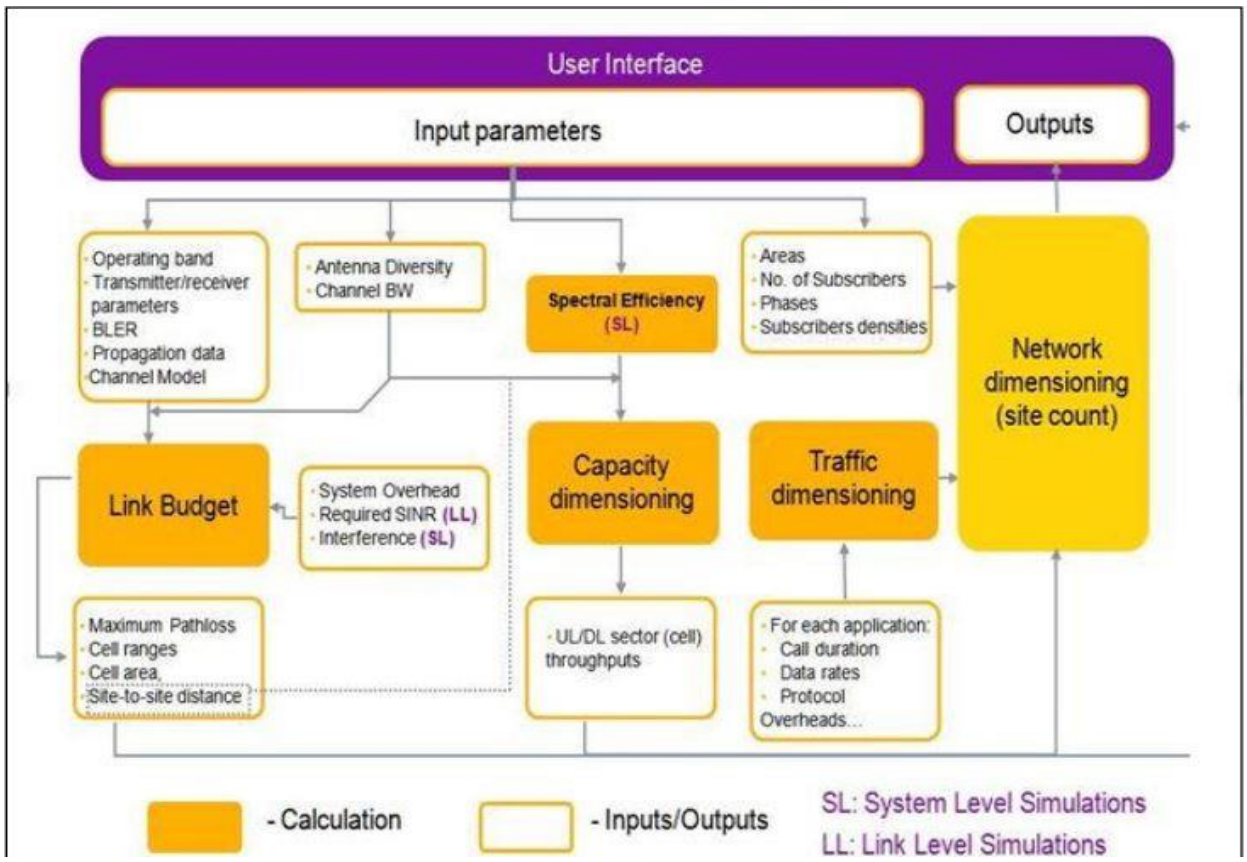


Figure 1 6 LTE Dimensioning 2

in LTE.

Radio link budget (RLB) inputs include transmitter power, transmitter and receiver antenna systems, configuration antennas used, conventional system gains and losses, Cell loading that effect the value of interference margin and propagation models. LTE can operate in conventional frequency bands of 900, 1800 and 2100 MHz as well as extended band of 2600 MHz, these bands are considered in the inputs. Additionally, channel types and geographical information is needed to start the coverage dimensioning exercise. Geographical input information consists of area type information (Urban, Rural, etc) and it related parameters (penetration loss, shadowing margin, etc) and sizes of each area type to be covered.

- Outputs of LTE Dimensioning

Outputs of the dimensioning phase are used to estimate the feasibility and cost of the network. These outputs are further used in detailed network planning and can be utilized for future work on LTE core network planning. Dimensioned LTE network can help out LTE core network team to plan a suitable network design and to determine the number of backhaul links required in the starting phase of the network

Cell size is the main output of LTE dimensioning exercise. Two values of cell radii are obtained, one from coverage evaluation and second from capacity evaluation. The maximum of the two numbers is taken as the final output. Cell radius is then used to determine the number of sites.

1.3.2.1 LTE Coverage Dimensioning Process:

the main goal of coverage planning is to estimate the coverage distance of an eNB with parameter settings based on actual cell edge coverage requirements in

order to meet network size requirements, in other words we will get three parameters from coverage planning

- Inter site distance (D)

$$D = 1.5 \times R \qquad (1.1)$$

1.5 factor because the overlapping between sites

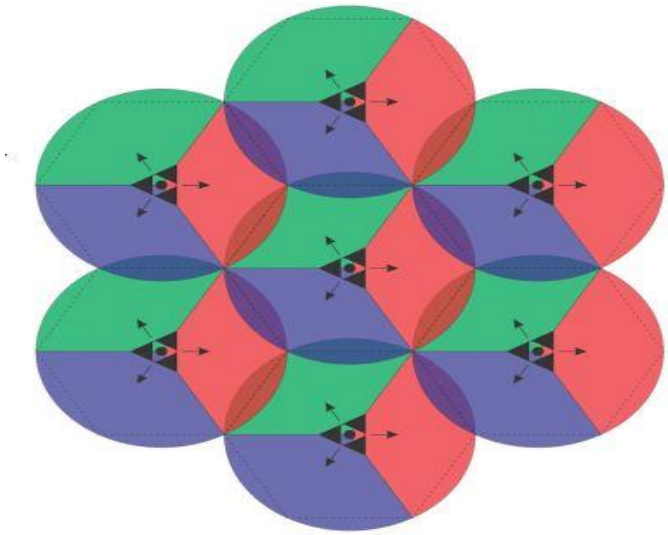


Figure 1 7 LTE Cell Overlapping

- *Number of sites N:*

$$N = \frac{Total\ Area}{Areaofsinglesite(Asite)} \qquad (1.2)$$

A site value depends on the antenna type used in this site

- At used Directive Antenna, it is :

$$A\ site\ = \frac{9\sqrt{3}}{8} \times R^2 = 1.948\ \times R^2 \qquad (1.3)$$

- At used Omnidirectional Antenna, it is

$$A\ site\ = \frac{3\sqrt{3}}{2} \times R^2 = 2.598\ \times R^2 \ (1.4)$$

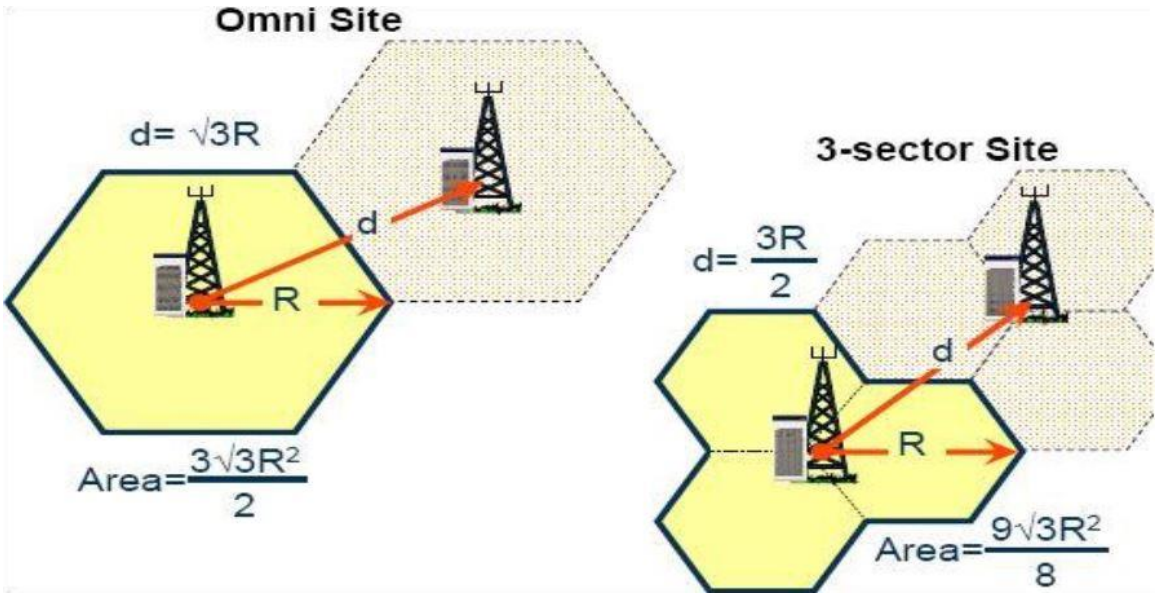


Figure 1 8 Cell antenna Types

- *For* site radius (R):
  To calculate site radius, we will pass through this step

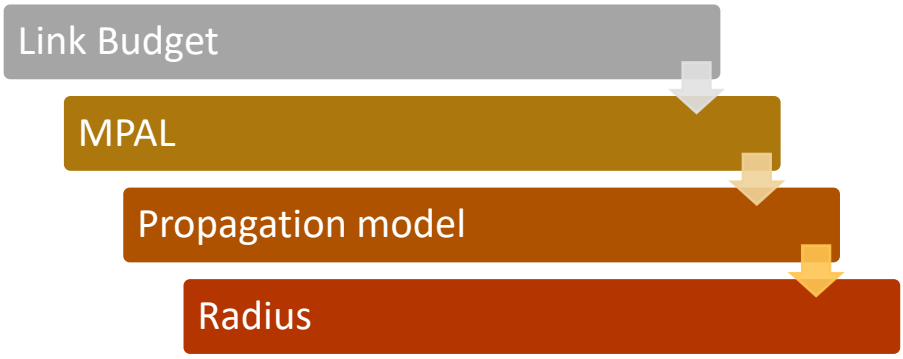


Figure 1 9 calculating cell Radius

1.3.2.1.1 Radio Link Budget:

Coverage planning consists of evaluation of DL and UL radio link budgets. The maximum allowable path loss (MAPL) is calculated based on service throughput defined by the cell edge user that required SINR level at the receiver. The minimum of the maximum path losses in UL and DL directions is ***converted into cell radius, by using a propagation model,*** such as Okumura–Hata. The cell range gives the number of base station sites required to cover the target geographical area. The link budget calculation can also be used to compare the relative coverage of the different systems. The propagation model used will be the COST-231 Hata model.

In general, there are two main reasons for establishing the RF link budget for LTE network:

- To establish the system designs for all gains and losses that may occur.
- To calculate the Maximum Allowed Path Loss (MAPL).

Link budget uses various parameters. Some of them are cited below:

I. Transmitting end:

- Cell edge user throughput
- Transmitting power
- Transmitting antenna gain
- Cable loss
- TMA insertion loss
- Body loss

II. Receiving end:

- UE Noise figure
- Thermal Noise
- Required SINR
- Receiver sensitivity

The Figure in below show LTE UL Budget Procedure

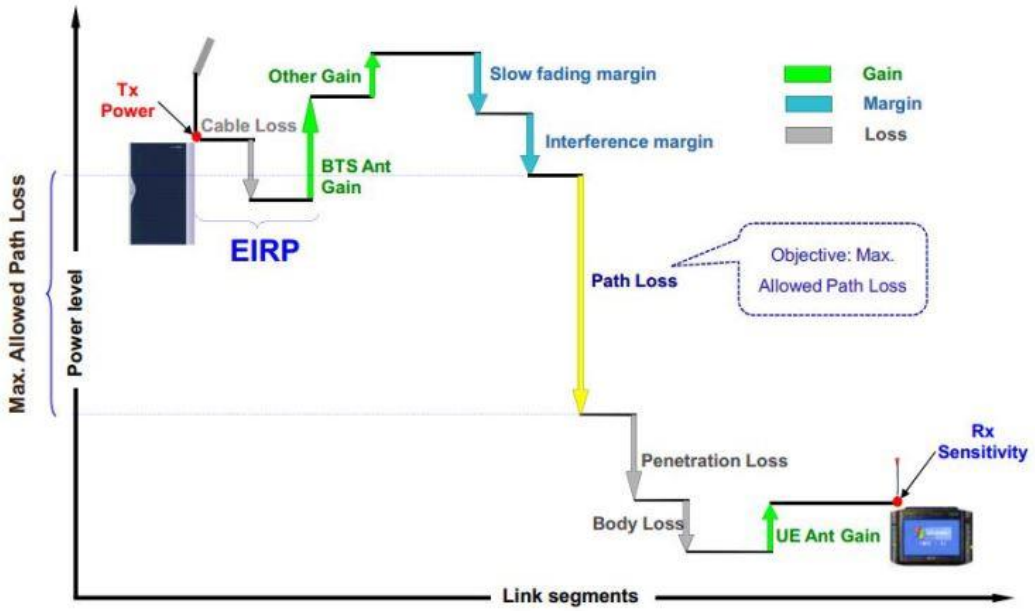


Figure 1 10 LTE UL Budget Procedure

And this Figure show LTE DL Budget Procedure:

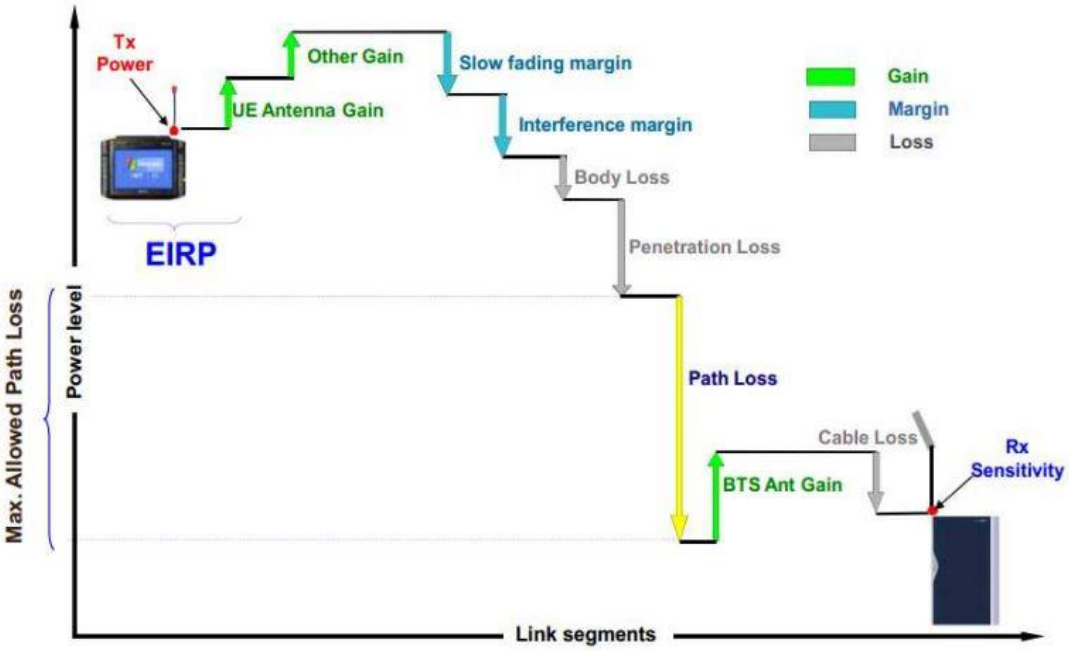


Figure 1 11 LTE DL Budget Procedure

Maximum Allowable Path Loss (MAPL): It is the maximum cell range to be estimated with a suitable propagation model which provides the number of base station sites required to cover the targeted area. Its equation is given by:

$$\mathbf{MAPL = TxPower + TxGains - TxLosses - RxSensitivity + RxGains - RxLosses}$$

We can write it in this formula

$$\mathbf{MAPL = TxPower - Losses(feeber + connectors + jumpers) + Tx\ ant\ Gains - Losses\ margins + Rx\ ant\ gains + RxSensitivity}$$

- **Transmitter Power (**TxPower**):** It includes the base station affecting downlink budget and terminal sides which affects the uplink budget
  eNB has two TxPower types 43dBm or 46dBm
- **Losses "feeder, connectors, jumpers"**

Figure 1 12 Feeders

Figure 1 14 Connectors

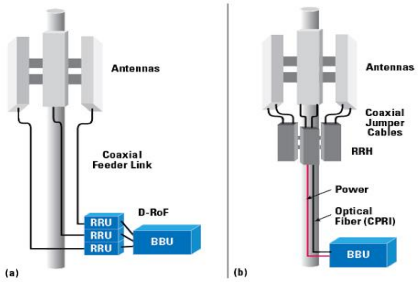


Figure 1 13 Jumpers

$$Feeder\ length = tower\ height + 5m \quad (1.5)$$

$$Feeder\ losses = Loss\left({}^{dB}/_{100}\right) \times Feeder\ length \quad (1.)$$

$$Jumper\ Losses = 0.5\ inch\ feeder \times \#\ jumpers\ length \quad (1.7)$$

$$Connectors\ Losses = \#\ jumpers\ \times 0.05\ dB \quad (1.8)$$

we can get $Loss(^{dB}/_{100})$ from this table:

Table 1 Feeder Losses

Feeder type	Loss (dB/100m)		
	2600 MHz	2100 MHz	900 MHz
½ inch	11	10.8	9
7/8 inch	6.3	6	4.9
1.25 inch	4.6	4.5	3.2
1.625 inch	3.8	3.5	2.6

- **Tx Ant gains**

Antenna Parameters:

- Gain & height: Choose proper antenna gain and height for specific frequency, morphology and coverage requirement.
- Beamwidth: related with sectorization. Select horizon beamwidth of 65 degree for 3 sectors scenario

- **Losses margins**

$$\text{Lossesmargins} = \text{penetration}_{\text{loss}} + \text{Body}_{\text{loss}} + \text{Slowfading}_{\text{loss}} + \text{Fastfading}_{\text{loss}} + \text{IM}_{UL} \quad (1.6)$$

- Penetration loss: Penetration loss indicates the fading of radio signals from an indoor terminal to a base station due to obstruction by a building.

Table 2 Penetration loss for environments

Clutter type	Losses
Dense Urban	20~25
Urban	18~20
Sub-urban	12~15
Rural	8~10
Car	6~8

- Body loss: Body loss indicates the loss generated due to signal blocking and absorption when a terminal antenna is close to the body.

Table 3 Body Losses

Loss Type	Losses (dB)
Voice calls	3
Video calls or Data	0

- 3

low Fading Margin**:** Shadow fading indicates the fading brought by

Table 4 Slow feeding margins for different environments

bstruction due to a building or a natural feature. Shadow fading changes

Environment	Coverage Probability			
	98%	95%	90%	85%
Rural / Sub-urban	5.5	2.9	0.5	-1.2
Urban	8.1	4.9	1.8	0.2
Dense Urban	10.6	6.7	3.1	0.6

ly, thus the name “slow fading”. Statistics repeatedly show that the median levels of received signals follow log-normal distribution with the time and location at a certain distance. Fading caused by location mainly from obstruction far exceeds fading caused by time. Therefore, the major concern for shadow fading is those caused by location changes.

- Interference Margin**:** Interference margin accounts for the increase in the terminal noise level caused by the interference from other users and it indicates the degradation of system receive performance caused by internal interference in the system due to system traffic. In fact, due to the frequency division nature of LTE, there is also a close correlation between actual traffic load and interference margin experienced by the network.
  IM can be defined by this relation:

$$IM = \frac{S/N}{S/(I_{own} + I_{other} + N)} \qquad (1.10)$$

- **Receiver Sensitivity:** It indicates the minimum signal strength required for decoding by the eNB or UE if there is no interference. Its formula is:

$$\mathbf{RxSensitivity} = \mathbf{Noise\ figure} + \boldsymbol{SINR} + \boldsymbol{Thermal\ Noise} \qquad (\mathbf{1.11})$$

Where:

- SINR = Signal-to-Interference Noise Ratio

- Noise figure is the ratio of SINR at the input end to the SINR at the output end of the receiver and is used to measure the performance of the receiver.
- And **Thermal Noise:** It is noise due to heat. It is given by:

$$\boldsymbol{Thermal\ Noise = KBT} \qquad (\mathbf{1.12})$$

Where:

- K = Boltzmann constant (1.38 x 10-23 J/K)
- T = Absolute temperature at 290K
- B = Channel bandwidth, which is 20 MHz

Finally, we can be calculating the MAPL value and go to next step

1.3.2.1.2 Propagation models:

The radio propagation model plays a key role in the link budget. The coverage radius of a base station is obtained based on the maximum propagation loss

Table 5 Propagation models usages

allowance in the link budget

Model	Frequency MHZ	Recommended use
Cost-231 Hata	150—2000	$0.02 < d < 5$ km, UMTS, GSM1800, LTE
Erceg-Greenstein	1900—6000	$0.1 < d < 8$ km, Fixed WiMAX
IMT	800-2800	Indoor office, vehicular, outdoor to indoor
ITU-526	30—1000	Fixed receivers
ITU-529	300-1500	$1 < d < 100$ km, GSM900, CDMA2000, LTE
Okumura-Hata	150—2200	$1 < d < 20$ km, GSM900, CDMA2000, LTE
WLL	30—10000	Fixed receivers, Microwave Links, WiMAX

These models are based on the frequency band, type of deployment area (urban, rural, suburban, etc.), and type of application. This **Table** lists the most widely used propagation models in current cellular systems. Most of these models are a fusion of empirical formulas extracted from field measurements and some statistical prediction models

Will be used one of the listed models, It is **Cost231-Hata Model**

- **Cost231-Hata Model**

Four parameters are used for estimation of the propagation loss by Hata's well-known model: frequency f, distance d, base station antenna height and the height of the mobile antenna

- Frequency band: 1500 MHz to 2000 MHz
- Base station height: 30 meters to 200 meters.
- Terminal antenna height: 1 meter to 10 meters
- Distance between the transmitter and receiver: 1 km to 20 km

The Cost231-Hata model can be expressed by the following formula:

$$Total = L - a(H_{ss}) + C_m \quad (1.9)$$

$$L = 46.3 + 33.9 \times \log(f) - \log(H_{Bs}) + (44.9 - 6.55 \times \log(H_{BS})) \times \log(d) \quad (1.10)$$

Where:

f indicates the working frequency of the system. The unit is MHz

H_{BS} indicates the height of the base station antenna. The unit is m.

H_{SS} indicates the height of the terminal antenna. The unit in m.

d indicates the distance between the terminal and the base station. The unit in km.

$a(H_{SS})$ indicates the terminal gain function. This function is related to the antenna height and working frequency of the terminal and the environment.

The value of C_m depends on the terrain type.

1.3.2.2 LTE Capacity Dimensioning:

With a rough estimation of the cell size and sites count, verification of coverage analysis is carried out for the required capacity. It is verified whether with the given sites density, the system can carry the specified load or new sites have to be added. Theoretical capacity of the network is limited by the number of eNBs installed in the network. Cell capacity in LTE is impacted by several factors, which includes interference level, packet scheduler implementation and supported In LTE, the main indicator of capacity is the SINR distribution in the cell. The SINR distribution can be directly mapped to the system capacity (data rate). The capacity

based on the number of sites is compared with the result of the coverage and the larger of the two numbers is selected as

$$No\ of\ Site\ = \frac{Total\ area\ traffic}{Site\ traffic\ (T\ site)} \quad (1.13)$$

$$T\ site\ = T\ cell\ \times (No\ of\ cells\ per\ site) \times Q \quad (1.14)$$

where:

- T site: actual throughput
- T cell: cell throughput
- Q: System load

$$T\ cell = 2000 \times NBR \times \left(\#\ {}^{RE}/_{RB} \right) \times (Modulation\ order \times coding\ rate) \times (1 - CCH) \quad (1.15)$$

Where:

- NBR: No of Resource Block
- CCH: control channel ratio

$$Total\ Traffic\ = No\ of\ active\ subscribers\ \times traffic\ per\ subscriber \quad (1.16)$$

1.3.2.2.1 *Subscriber traffic:*

Traffic forecast should be done by analyzing the offered Busy Hour traffic per subscriber for different services in each area. The traffic model defines an application mix consisting of 5 services (VoIP, Video, Streaming, Web browsing & FTP).

At the end the required sites number for a specific area should be chosen to be the maximum number of sites obtained from coverage and capacity dimensioning calculations to satisfy the demand traffic requirements

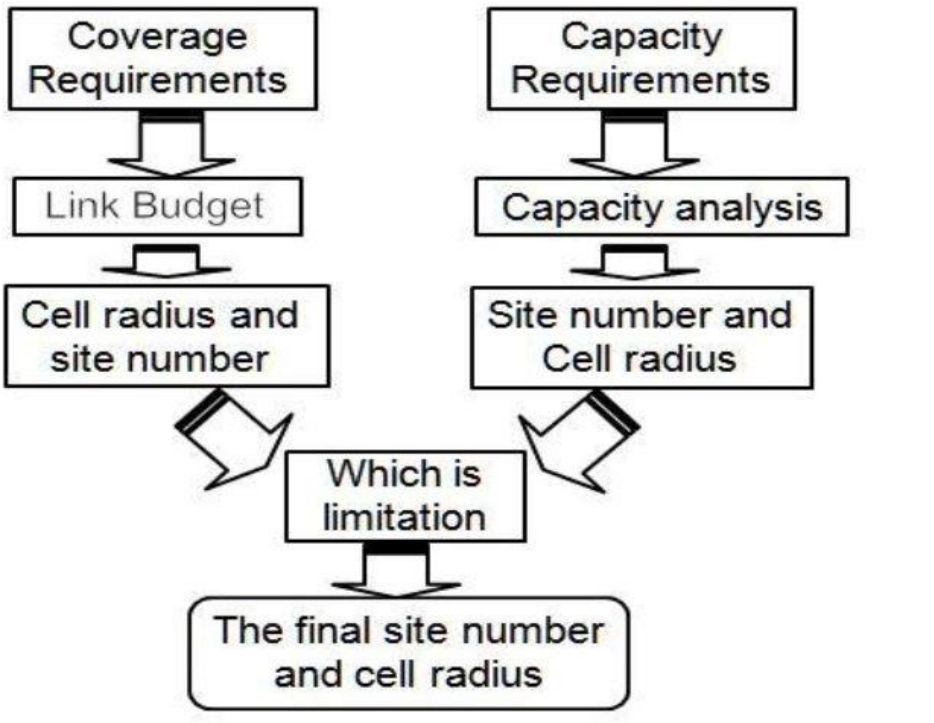


Figure 1 15 # of sites procedures

1.3.3 Detailed planning

After the planning phase has finished and the site location and configurations are known, detailed planning can be started. The detailed planning phase includes frequency, adjacency and parameter planning.

1.3.4 Verification and acceptance

In addition to fine-tuning a search is made for possible mistakes that might have occurred during the installation. Prelaunch optimization is high level optimization but does not go into detail. Network optimization continues after the launch at a more detailed level.

1.3.5 Optimization

As we know that optimization is a continuous process. All available information about the network and its status is required as input for the optimization. Some necessary components like statistical figures, alarms and traffic have to be monitored carefully.

Chapter Two: Time Series Analysis

2|Time Series analysis overview

2.1 Introduction:

Forecasting demand in any industry is a very important step that can be used in a large variety of applications, from resources planning to process optimization and predictive maintenance. All of these applications help the company to take some proactive actions that can be done to maximize benefits of the company's resources.

The telecom industry is not an exception and making accurate forecasts whether for demand - which means traffic in our case - or even the Key Performance indicators (KPI's) help the telecom players to adjust their efforts to maximize network performance.

When making forecasts about the traffic, we can use these forecasts for network planning, network optimization and predictive maintenance which can be used further to be used as a prerequisite step for the Self organizing network (SON).

In this chapter we'll describe some important features that we must consider when describing and modeling a time series.

2.2 Time series features:

Time Series data is a sequence of measurements that represents observations for variables or some variables; more often these measurements are made at a regular time interval.

This can be anything, maybe a sensor measurement, sales of a specific product or as in our case traffic of a mobile network in a specific region.

Any time series can be described by using some features that can represent all aspects of the time series very well. These features are as follow:

2.2.1 Trend: (T)

Trend feature is the measurement of the average direction of the data, whether it was as increase or decrease, so to describe a trend of time series data is that we have to observe on average the series tends to increase or decrease, if it increases so it has a positive trend, if it decreases so it has a negative trend.

And as examples of trend are in the following use cases:

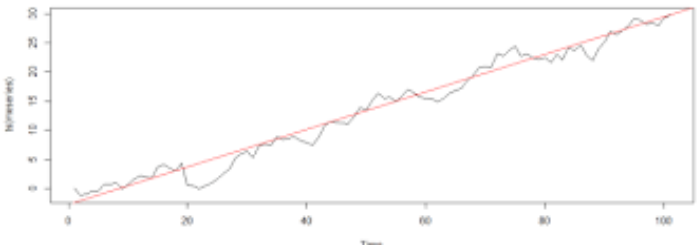


Figure 2 1 Upward trend

In this series we can observe an increasing direction in the long term that is represented with the red line, so this series has a positive (Upward) trend.

On the other hand, by the same approach we can observe this series goes down with time, so this series has a negative (Downward) trend, as the value of the variable tends to decrease in general.

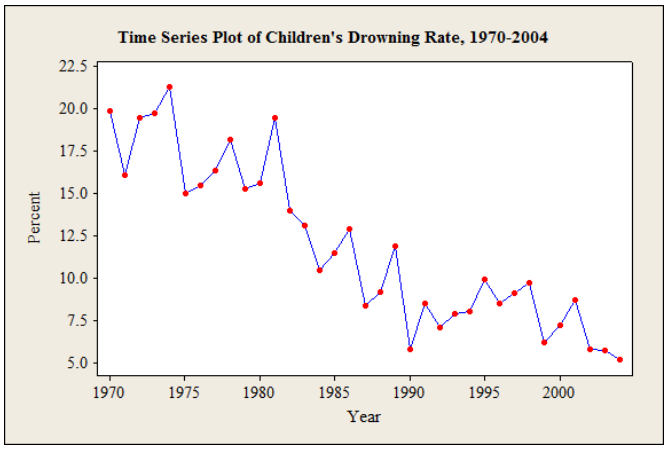


Figure 2 2 Downward trend example

There is another type of trend which is horizontal or stationary, in which the series does not tend to increase or decrease.

We can also classify trend with another classification paradigm, according to its consistency, if the trend is constant whether it was positive or negative, it's called a constant trend, if it's not constant e.g., the series is upward in some point and downward in another time then this is a variable trend.

2.2.2 Periodic part:

2.2.2.1 Seasonality: (S)

Seasonality means if there is a regular repeating ups and downs in the series related to time intervals, this interval can be minute, hour, day, week quarter or even a year or any other period of time that the series is repeating itself regularly.

As an example of the seasonality, we can recall temperature during the year, in summer the temperature goes up, and then till the winter it goes down and this pattern is being repeated each year, so this is a yearly seasonality.

2.2.2.2 Cyclic part: (C)

Something is repeated on a time not related to the seasonal part, and usually it may be a thing that is repeated with intervals more than one year.

2.2.3 Random part: (R)

This is the complete random part of the series, and in most cases, this is the part which causes most of the errors in models.

2.3 Time series models:

Any time series can be modeled with combination of the previous characteristics, but there are two main models in addressing time series-related problems:

2.3.1 Additive Model:

This is the model where all components are added together, this model is optimal representation of the series where the trend is not stationary, and this model can be represented with the following equation:

$$Y = T + S + C + R \qquad (2.1)$$

2.3.2 Multiplicative model:

In this model we can represent the series as all parts multiplied together, and this is an optimal approach when addressing a stationary trend problem, where the trend is nearly constant and has no increasing or decreasing effect:

$$Y = T * S * C * R \qquad (2.2)$$

Although these are the most common models, but we can use a combination of these two approaches to address the problem.

2.4 Time series characteristics:

2.4.1 Autocovariance:

Autocovariance measures the linear dependency between two points on the same series observed at different times, and can be observed from the equation:

$$\gamma(s,t) = E[(x - \mu_s)(x - \mu_t)] \qquad (2.3)$$

Where E is the mean function and can be found by the following equation:

$$\mu_{xt} = E(x_t) = \int_{-\infty}^{\infty} x\, f_t(x) dx \qquad (2.4)$$

Where x_t is the value of the series x at time t, and it represents the expected value where no confusion exists?

Very smooth series has a large covariance and the choppy-series has nearly zero covariance.

2.4.2 Autocorrelation function: (ACF)

Measures the linear predictability of series at time t say x_t using only the value x_s and can be represented by the equation:

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(s,t)}} \qquad (2.5)$$

Where $-1 \leq \rho(s,t) \leq 1$.

2.4.3 partial autocorrelation function: (PACF):

$$\phi_{hh} = corr(x_h - x_h^{h-1}, x_0 - x_0^{h-1}), h \geq 2 \qquad (2.6)$$

The PACF, ϕ_{hh}, is the correlation between x_t and $x_t - h$ with the linear dependence of $\{x_{t-1}, \ldots, x_{t-(h-1)}\}$, on each, removed. If the process x_t is Gaussian, then $\phi_{hh} = corr(x_t, x_{t-h}|x_{t-1}, \ldots, x_{t-(h-1)})$. That is, ϕ_{hh} is the correlation coefficient between x_t and x_{t-h} in the bivariate distribution of (x_t, x_{t-h}) conditional on $x_{t-1}, \ldots, x_{t-(h-1)}$.

2.4.4 Stationarity:

The preceding definitions of the mean and autocovariance functions are completely general. Although we have not made any special assumptions about the behavior of

the time series, many of the preceding examples have hinted that a sort of regularity may exist over time in the behavior of a time series. We introduce the notion of regularity using a concept called stationarity, so we have the following conditions of stationarity:

2.4.4.1 Strictly stationary series:

Time series is one for which the probabilistic behavior of a collection of data:

$$\{x_1, x_2, x_3.x_t\}$$

Is identical to that of the time shifted set:

$$\{x_[1+h], x_[2+h], x_[3+h]..X_[t+h]\}$$

That is,

$$P\{xt1 \leq c1, ..., xtk \leq ck\} = P\{xt1 + h \leq c1, ..., xtk + h \leq ck\} \qquad (2.7)$$

If a time series is strictly stationary, then all of the multivariate distribution functions for subsets of variables must agree with their counterparts in the shifted set for all values of the shift parameter h.

2.4.4.2 Weakly stationary:

For the series to be weakly stationary must have the following properties:

i) the mean value function, μ_t, defined in (2.4) is constant and does not depend on time t.

ii) the covariance function, $\gamma(s,t)$, defined in (2.5) depends on s and t only through their difference $|s - t|$.

This formula expresses that the mean and auto-covariance of X are time-independent and that its variance is finite. Weak stationarity relaxes the strong stationarity property by only constraining the first moment and auto-covariance function of X instead of its whole distribution. Thus, strong stationarity implies

weak stationarity. Figure 1 shows an example for weak stationarity, mean stationarity, variance stationarity and mean and variance non-stationarity. Weak stationarity is also called second-order stationarity. First-order stationarity refers to the case where only the mean is constant. Weak stationarity relaxes the strong stationarity property by only constraining the first moment and autocovariance function of X instead of its whole distribution. Thus, strong stationarity implies weak stationarity. Figure 1 shows an example for weak stationarity, mean stationarity, variance stationarity and mean and variance nonstationary. Weak stationarity is also called second-order stationarity. First-order stationarity refers to the case where only the mean is constant.

And examples:

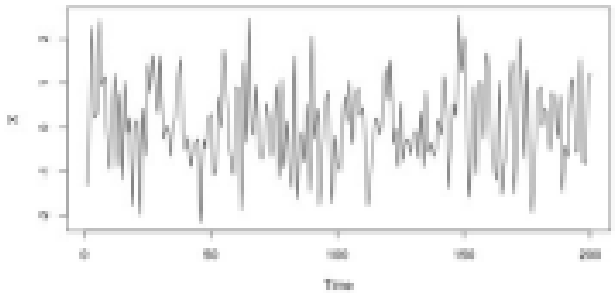


Figure 2 3 Weak stationary

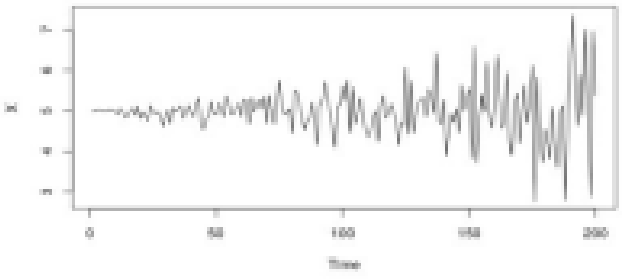


Figure 2 4 Mean stationarity and variance nonstationary

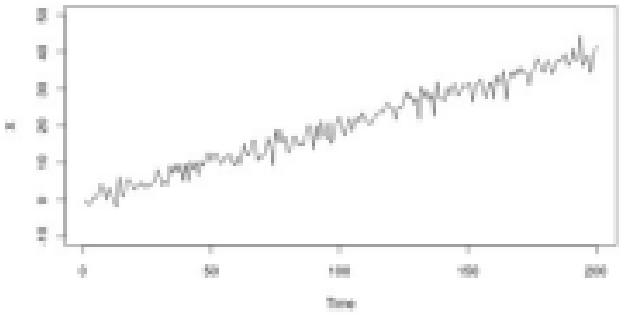


Figure 2 5 Mean nonstationary and variance stationarity

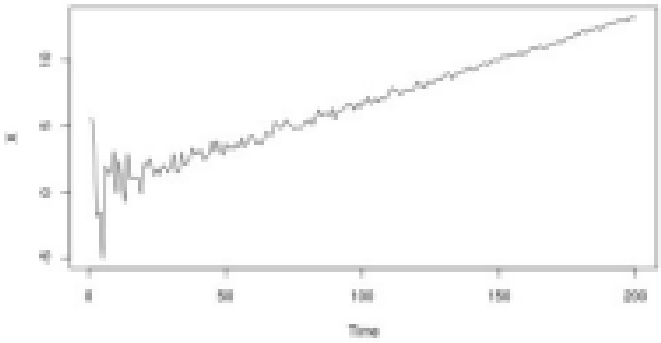


Figure 2 6 Mean and variance no stationarities

2.5 Forecasting Models:

There are several methods to make forecasts, we will explore some of them:

2.5.1 ARIMA Models:

Classical regression is often insufficient for explaining all of the interesting dynamics of a time series. the introduction of correlation as a phenomenon that may be generated through lagged linear relations leads to proposing the autoregressive (AR) and autoregressive moving average (ARMA) models. Adding nonstationary models to the mix leads to the autoregressive integrated moving average (ARIMA) model popularized in the landmark work by Box and Jenkins (1970). The Box–Jenkins method for identifying a plausible ARIMA model is given in this chapter along with techniques for parameter estimation and forecasting for these models.

2.5.1.1 Autoregressive Moving Average Models:

Autoregressive models are based on the idea that the current value of the series, xt, can be explained as a function of p past values, $x_[t1], x_[t2], ..., x_[tp]$, where p determines the number of steps into the past needed to forecast the current value.

An autoregressive model of order p, abbreviated AR(p), is of the form:

$$x_t = \phi_1 x_{t1} + \phi_2 x_{t-2}. + \phi_p x_{t-p} + \omega_t \quad (2.8)$$

where xt is stationary, φ1, φ2,...,φp are constants (φp = 0). Unless otherwise stated.

If the mean, μ, of x_t is not zero, replace x_t by $x_{t-\mu}$ in (2.7), i.e.,

$$x_{t-\mu} = \phi_1(x_{t-1-\mu}) + \phi_2(x_{t-2-\mu}) + +\phi_p(x_{t-p-\mu}) + \omega_t \quad (2.9)$$

So we can present the Auto-regressive operator as follows:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B_2 - -\phi_p B_p \quad (2.10)$$

We initiate the investigation of AR models by considering the first-order model, AR(1), given by $xt = xt - 1 + wt$ Iterating backwards k times, we get

$$x_t = \phi x_{t-1} + \omega_t = \phi(\phi x_{t-2} + \omega_{t-1}) + \omega_t = \phi_2 x_{t-2} + \phi\omega_{t-1} + w_t = \phi_k x_{t-k} + \sum_{j=0}^{k-1} \phi_j \omega_{t-j} \quad (2.11)$$

This method suggests that, by continuing to iterate backwards, and provided that |φ| < 1 and xt is stationary, we can represent an AR (1) model as a linear process given by:

$$x_t = \sum_{j=0}^{\infty} \phi_j \omega_{t-j} \qquad (2.12)$$

Moving average model of order q, or MA(q) model, is defined to be:

$$x_t = \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + + \theta_q \omega_{t-q} \qquad (2.13)$$

where there are q lags in the moving average and θ1, θ2,...,θq (θq = 0) are parameters. The noise *wt* is assumed to be Gaussian white noise.

We may also write the MA(q) process in the equivalent form:

$$x_t = \theta(B)\omega_t \qquad (2.14)$$

where:

$$\theta(B) = 1 + \theta_1 B + \theta_2 B_2 + + \theta_q B_q \qquad (2.15)$$

A time series $\{x_t; t = 0, 1, 2, ...\}$ is ARMA(p, q) if it is stationary and

$$x_t = \phi_1 x_{t-1} + + \phi_p x_{t-p} + \omega_t + \theta_1 \omega_{t-1} + + \theta_q \omega_{t-q} \qquad (2.16)$$

with $\phi_p = 0, \theta_q = 0, and \sigma_\omega^2 > 0$.

We set $\alpha = \mu(1 - \phi_1 - - \phi_p)$ and write the model as

$$x_t = \alpha + \phi_1 x_{t-1} + + \phi_p x_{t-p} + \omega_t + \theta_1 \omega_{t-1} + + \theta_q \omega_{t-q} \qquad (2.17)$$

2.5.1.2 Seasonal models:

The past models deal with time series with no seasonality on it, but in the existence

of the seasonality we need some models that can deal with seasonal components in the models, so we represent the seasonal ARIMA models.

As we discussed further seasonality is a repeated pattern that is repeated periodically in some certain time intervals.

In a seasonal ARIMA (S-ARIMA) model, seasonal AR and MA terms predict

x_t using data values and errors at times with lags that are multiples of S (the span of the seasonality).

2.5.1.2.1 Differencing

Almost by definition, it may be necessary to examine different data when we have seasonality. Seasonality usually causes the series to be non-stationary because the average values at some particular times within the seasonal span (months, for example) may be different from the average values at other times. For instance, our sales of cooling fans will always be higher in the summer months.

- Seasonal differencing

Seasonal differencing is defined as a difference between a value and a value with lag that is a multiple of S.

- Non-seasonal differencing

If trend is present in the data, we may also need non-seasonal differencing. Often (not always) a first difference (non-seasonal) will "detrend" the data. That is, we use $(1 - B)x_t = x_t - x_{t-1}$ in the presence of trends.

- Differencing for Trend and Seasonality

When both trend and seasonality are present, we may need to apply both a non-seasonal first difference and a seasonal difference.

2.5.1.2.2 S-ARIMA Models:

The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model. One shorthand notation for the model is:

$$ARIMA(p,d,q) \times (P,D,Q)S \qquad (2.18)$$

with p = non-seasonal AR order, d = non-seasonal differencing, q = non-seasonal MA order, P = seasonal AR order, D = seasonal differencing, Q = seasonal MA order, and S = time span of repeating seasonal pattern.

2.5.1.3 ARCH:

An ARCH (autoregressive conditionally heteroscedastic) model is a model for the variance of a time series. ARCH models are used to describe a changing, possibly volatile variance. Although an ARCH model could possibly be used to describe a gradually increasing variance over time, most often it is used in situations in which there may be short periods of increased variation. (Gradually increasing variance connected to a gradually increasing mean level might be better handled by transforming the variable).

- The ARCH (1) Variance Model

Suppose that we are modeling the variance of a series y_t . The ARCH (1) model for the variance of model *yt* is that conditional on y_{t-1} ,

the variance at time t is:

$$Var(y_t|y_{t-1}) = \sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 \qquad (2.19)$$

- Generalizations:

An ARCH(m) process is one for which the variance at time t is conditional on observations at the previous *m* times, and the relationship is:

$$Var(y_t|y_{t-1}....y_{t-m}) = \sigma_t^2 = \alpha_0 + \alpha_1 y_{t-2}^2 + ... + \alpha_m y_{t-m}^2 \qquad (2.20)$$

2.5.1.4 Vector Autoregressive (VAR) models:

VAR models (vector autoregressive models) are used for multivariate time series. The structure is that each variable is a linear function of past lags of itself and past lags of the other variables.

As an example suppose that we measure three different time series variables, denoted by $x_{t,1}, x_{t,2}, and x_{t,3}$ The vector autoregressive model of order 1, denoted as VAR(1), is as follows:

$$x_{t,1} = \alpha_1 + \phi_{11} x_{t-1,1} + \phi_{12} x_{t-1,2} + \phi_{13} x_{t-1,3} + \omega_t \qquad (2.21)$$

$$x_{t,2} = \alpha_2 + \phi_{21} x_{t-2,1} + \phi_{22} x_{t-2,2} + \phi_{23} x_{t-2,3} + \omega_t \qquad (2.22)$$

$$x_{t,3} = \alpha_3 + \phi_{31} x_{t-3,1} + \phi_{32} x_{t-3,2} + \phi_{33} x_{t-3,3} + \omega_t \qquad (2.23)$$

Each variable is a linear function of the lag 1 values for all variables in the set.

2.6 Time Series decomposition;

This step is very important for any analysis in time series. We separate each part of the time series to be able to make any analysis on it and to know the main properties for the series.

This can be done through some steps as follows:

- Estimating the trend: this can be done by several approaches like:
  - Smoothing approaches like moving average.
  - Model trend with a regression equation.
- De-trend the series, by subtracting the effect of the trend form the overall

series if we use additive model, and by dividing on the trend if we use the multiplicative model.

- Extract the seasonal components: by extracting the effect of seasonal factor and eliminating its effect.
- Finally, we estimate the residuals or the random part of the series by:
  - $Randompart = Series - Trend - Seasonalpart$

    or
  - $$RandomPart = \frac{Series}{SeasonalPart \times Trend}$$

The random component could be analyzed for such things as the mean location, or mean squared size (variance), or possibly even for whether the component is actually random or might be modeled with an ARIMA model.

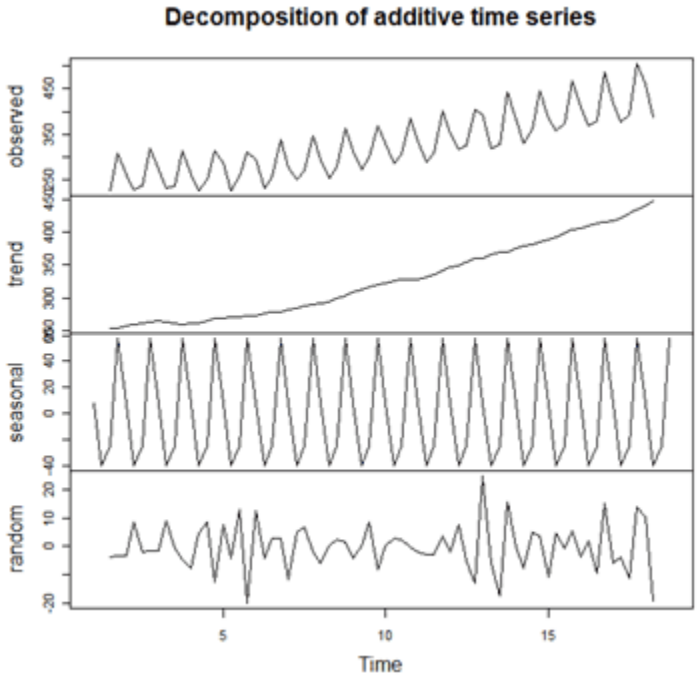


Figure 2 7 time series components

2.7 Smoothing Time Series:

Smoothing is usually done to help us better see patterns, trends for example, in

time series. Generally smooth out the irregular roughness to see a clearer signal. For seasonal data, we might smooth out the seasonality so that we can identify the trend. Smoothing doesn't provide us with a model, but it can be a good first step in describing various components of the series, as an example:

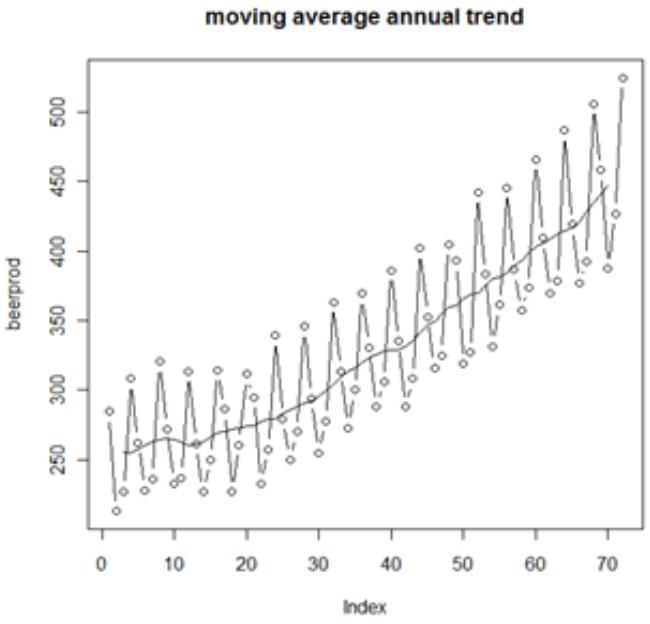


Figure 2 8 detrending with moving average

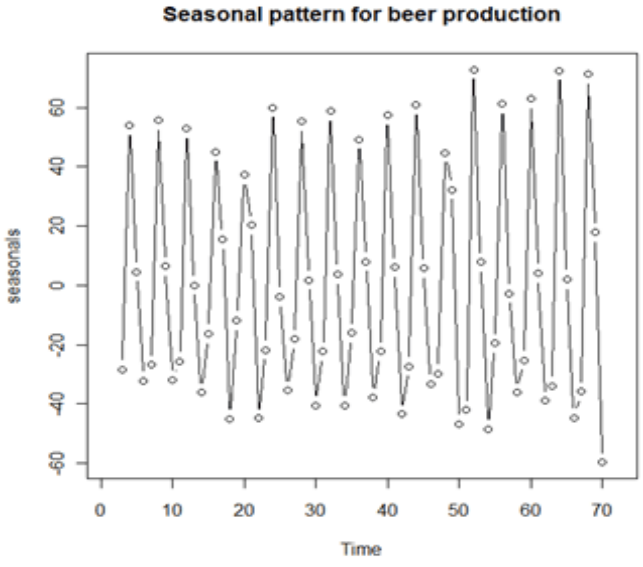


Figure 2 9 Seasonal component

Chapter Three: Milan Traffic Forecasting

3| Milan Traffic forecasting

3.1 Problem statement:

In the recent years, mobile network loads has grown exponentially due to the increasing demand and the variety of the services that can a mobile network operator can provide, which caused a huge demand for predicting the volume of the traffic, and this is a very challenging task as network traffic faces high nonlinearities and complex patterns, in this project we propose a several models to predict the temporal aspects of the mobile network traffic by dealing with each area as a separated entry, the modeling process provides a wide variety of alternatives throughout the whole modeling form anomaly detection which includes using Tokey's box, isolation forest and LSTM, and missing data imputations which includes using GAINs and ConvGains and also Seasonal interpolation, and finally the forecasting models which includes using FBProphet model that has a great performance and accuracy and also other statistical methods like STL and SARIMA and finally deep learning methods like CNN.

3.2 Dataset Understanding:

Our Dataset is from Telefonica operator in Italy, and it represents the traffic loads across 2 months from November 1st, 2013 to January 1st, 2014, this data is a spatio-temporal data that contains temporal measurements taken from a given area, so the spatial distribution irregularity is aggregated in square grids, this allows

comparisons between different areas and eases the geographical management of the data.

The data was taken from Two Italian cities, Milan and Tarantino, the area of Milan is divided into 100X100 grid squares and Tarantino is divided into 117X98 grid square, as in the following figure, every grid square is 235 X 235 meters.

This data measures the interaction between the humans and the mobile network through measuring the Call Detailed Records (CDRs) over a certain area in a given time interval, and the mechanism of taking these CDRs is as follows: A new CDR is generated every time a user engages with the network entity, here is the Radio Base Station (RBS), this created CDR is recording the time of interaction between the user and RBS.

An imaginary image about the data is in the following figure:

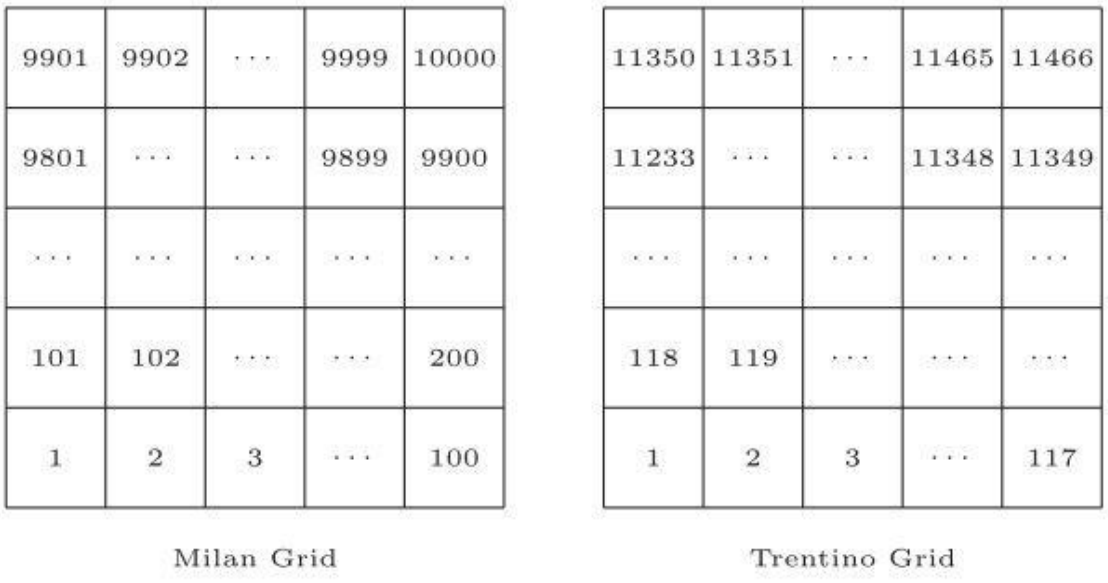


Figure 3. 1 Milan and Trentino grid systems

So, the data is about a measurement every 10 minutes for two months over 10000 grid squares for Milan and 11466 grid squares for Trentino.

So, for every grid square we have 6*24*60 = 8640 measurement for every grid.

3.2.1 Dataset description:

This dataset serves as a measure of the level of interaction between the users and the mobile phone network.

Square id: identification string of a given square of Milan/Trentino GRID;

Time Interval: start interval time expressed in milliseconds. The end interval time can be obtained by adding 600,000 milliseconds (10 min) to this value;

SMS-in activity: activity proportional to the amount of received SMSs inside a given square id and during a given Time interval. The SMSs are sent from the nation identified by the Country code;

SMS-out activity: activity proportional to the amount of sent SMSs inside a given square id during a given Time interval. The SMSs are received in the nation identified by the Country code;

Call-in activity: activity proportional to the amount of received calls inside the square id during a given Time interval. The calls are issued from the nation identified by the Country code;

Call-out activity: activity proportional to the amount of issued calls inside a given square id during a given Time interval. The calls are received in the nation identified by the Country code;

Internet traffic activity: number of CDRs generated inside a given square id during a given Time interval. The Internet traffic is initiated from the nation identified by the Country code;

Country code: the phone country code of the nation.

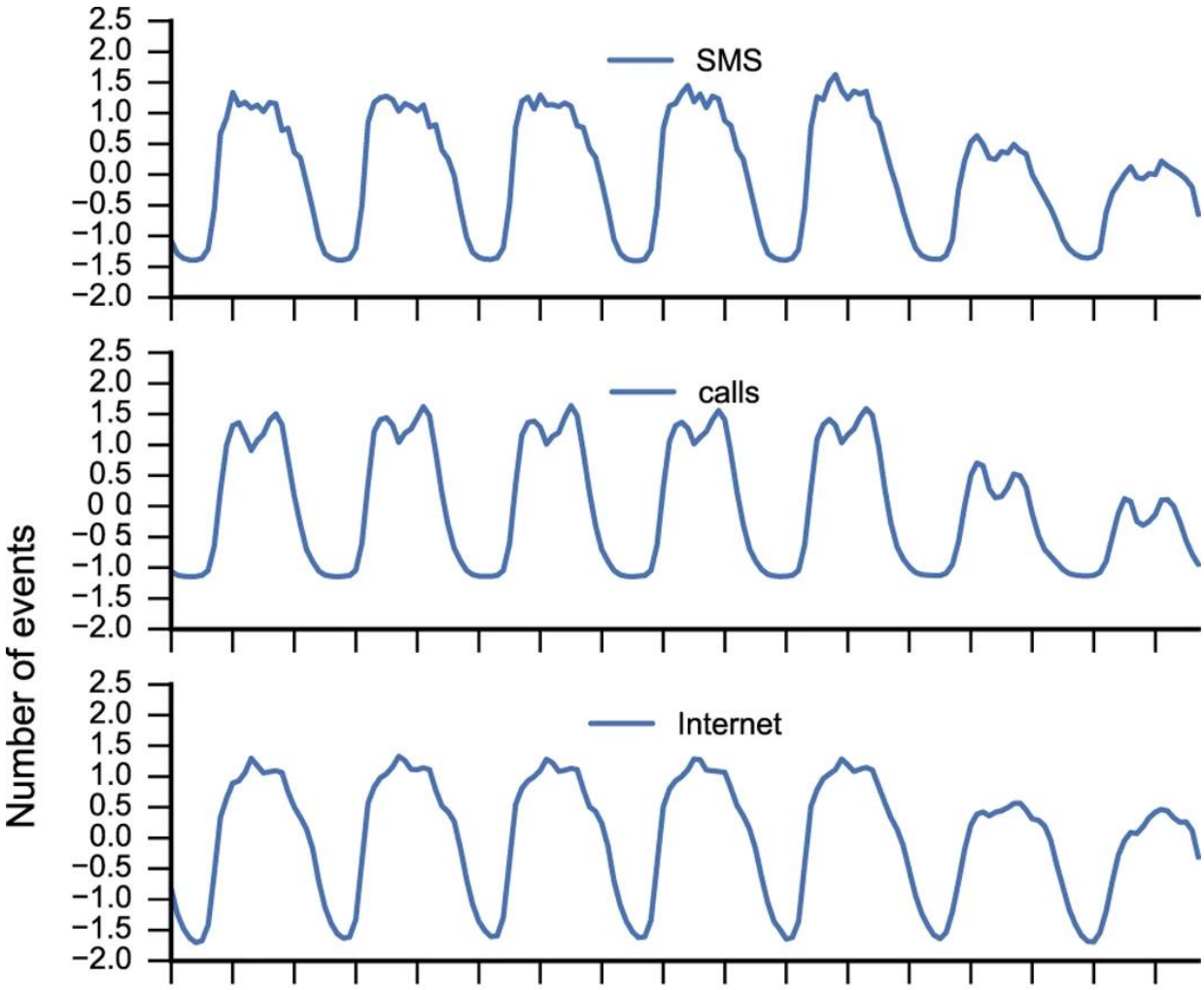


Figure 3. 2 Line plot for the dataset main feature

3.3 Data Preprocessing:

After knowing our dataset, we need to explore it further and clean it to be appropriate for forecasting step, in this stage we did some steps in cleaning.

3.3.1 Data Cleaning and exploring:

3.3.1.1 Roaming:

Roaming is the process in which a user from an operator X uses the infrastructure of the operator Y to connect with the network and uses different services, so in this case, the user makes a load in the Y network infrastructure but the operator who benefits from this user is the operator X.

And we can find the roaming mainly in two shapes:

First, a user from a certain country is traveling to another one, so when landing on the new country he needs any services form the mobile network so goes roaming till he buys a new SIM card from an operator in the new country. We can find this kind of roaming in airports and in touristic places.

Second form of roaming happens when a user of an operator in an area goes to another area inside the country but his operator does not have coverage there, so he will use any other available network.

Here in our analysis when we got the data, we want to see the effect of the roaming traffic in our data, to decide if we will proceed with it or ignore its effect if it's negligible.

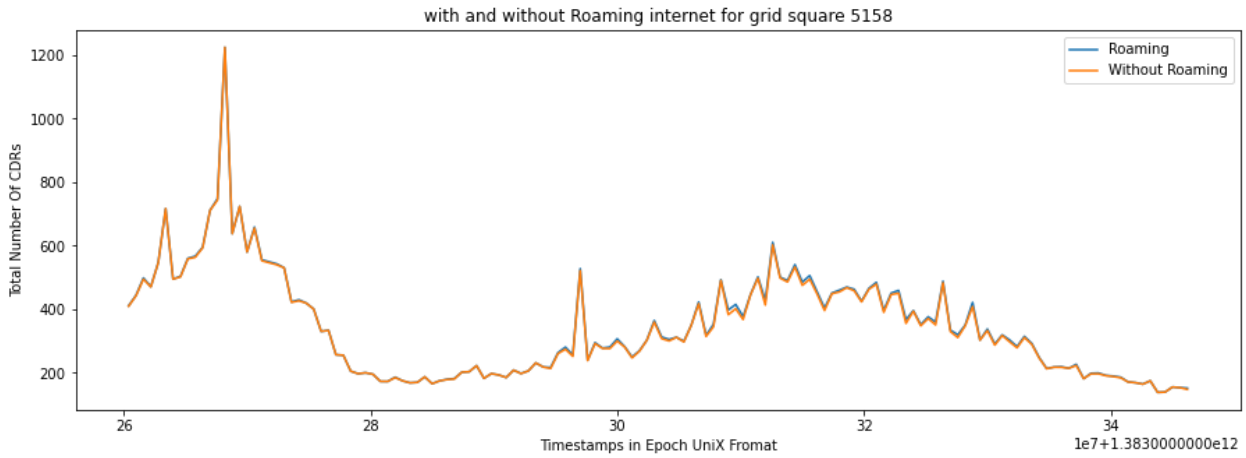


Figure 3. 3

When we draw the traffic of the internet with and without roaming for randomly selected cells, we have found that it has no effect on the data so we can ignore the effect of roaming data to make our decisions easier.

3.3.1.2 Call Details Requests: (CDRs)

Our dataset measures the CDRs not the actual traffic, this is come from Telefonica itself to hide the true numbers by multiplying the CRDs with some constant k.

But the problem is how the call detailed ratio is being calculated, The Call Detail Records (CDRs) are provided by the Semantics and Knowledge Innovation Lab (SKIL) of Telecom Italia. Every time a user engages in a telecommunication interaction, a Radio Base Station (RBS) is assigned by the operator and delivers the communication through the network. Then, a new CDR is created recording the time of the interaction and the RBS which handled it. From the RBS it is possible to obtain an indication of the user's geographical location, thanks to the coverage maps Cmap which associates each RBS to the portion of territory which it serves.

In order to spatially aggregate the CDRs inside the grid, each interaction is associated with the coverage area v of the RBS which handled it. Hence, the number of records $si(t)$ in a grid square i at time t is computed as follows:

There are many types of CDRs and Telecom Italia has recorded the following activities:

Received SMS a CDR is generated each time a user receives an SMS

Sent SMS a CDR is generated each time a user sends an SMS

Incoming Call a CDR is generated each time a user receives a call

Outgoing Call a CDR is generated each time a user issues a call

Internet a CDR is generated each time a user starts an Internet connection or ends an Internet connection. During the same connection a CDR is generated if the connection lasts for more than 15 min or the user transferred more than 5 MB.

The shared datasets were created combining all this anonymous information, with a temporal aggregation of time slots of ten minutes. The number of records in the datasets $S'i(t)$ follows the rule:

$$S(t) = Si(t)k \qquad (3.1)$$

where k is a constant defined by Telecom Italia, which hides the true number of calls, SMS and connections.

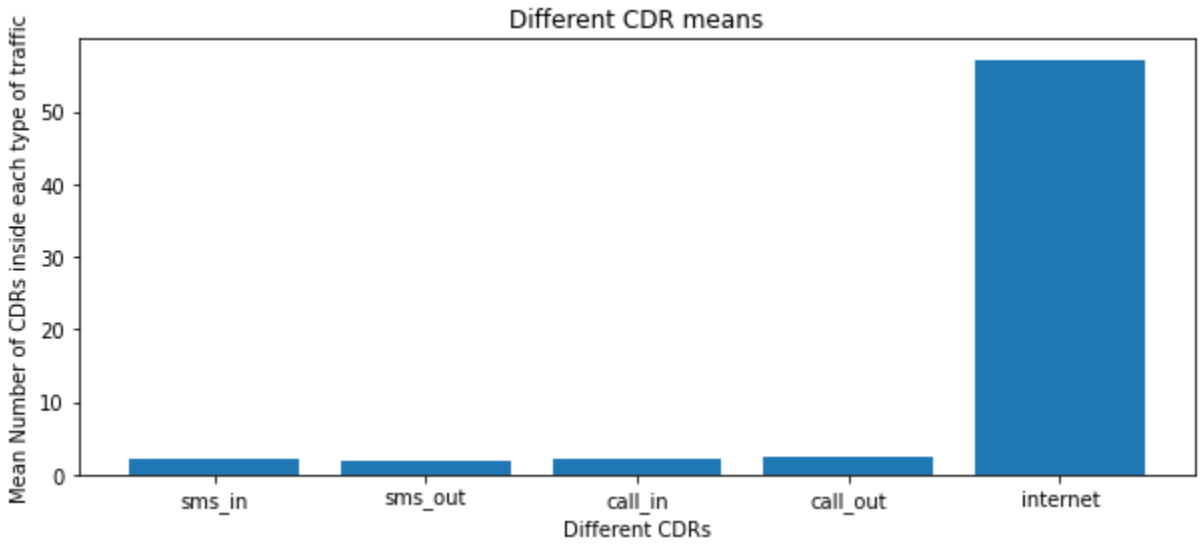


Figure 3. 4

In our analysis we want to address the dominant CDRs and as we go o time, we may see that the internet is the most important thing for the operator to provide a good service, so we have to validate this:

As we can see the mean internet CDRs is very large, so in our analysis we want to make sure we are forecasting the most thing that makes load on the network, in our

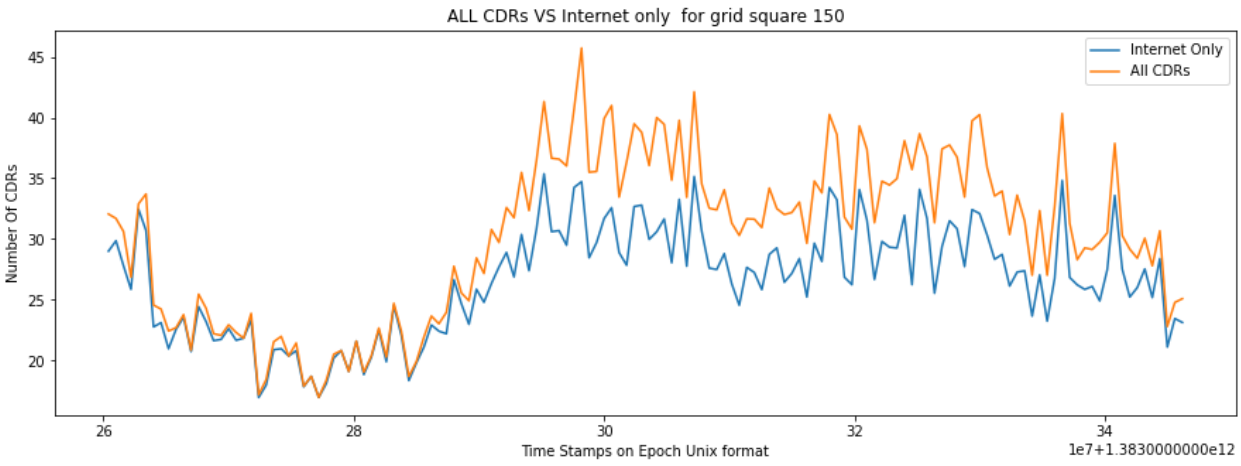


Figure 3. 5

case here we may think it's internet, but we want to validate more.

Form the previous graph we can assume that the internet not only is the dominant CDR but also it represents the data fluctuations very well, so we will decide to proceed with internet CDRs only.

3.3.1.3 Resampling:

Due to the missing measurements, we have found two types of missing measurements, the first one is that there are timestamps of the measurements but the value is Nan.

The second type is that neither the measurement or the timestamp is in the data.

So to make sure that every time stamp is there we resampled the data and added more Nans.

3.3.1.4 Choosing Subsets:

As we mentioned we consider the main grid squares that face more loads, as these places tend to have more importance for the operators to make sure it's doing well as it is the most beneficial place for them.

And also due to our limits in computational power, so we decided to choose the most loaded 9 contagious cells, so to know this we did the following graph:

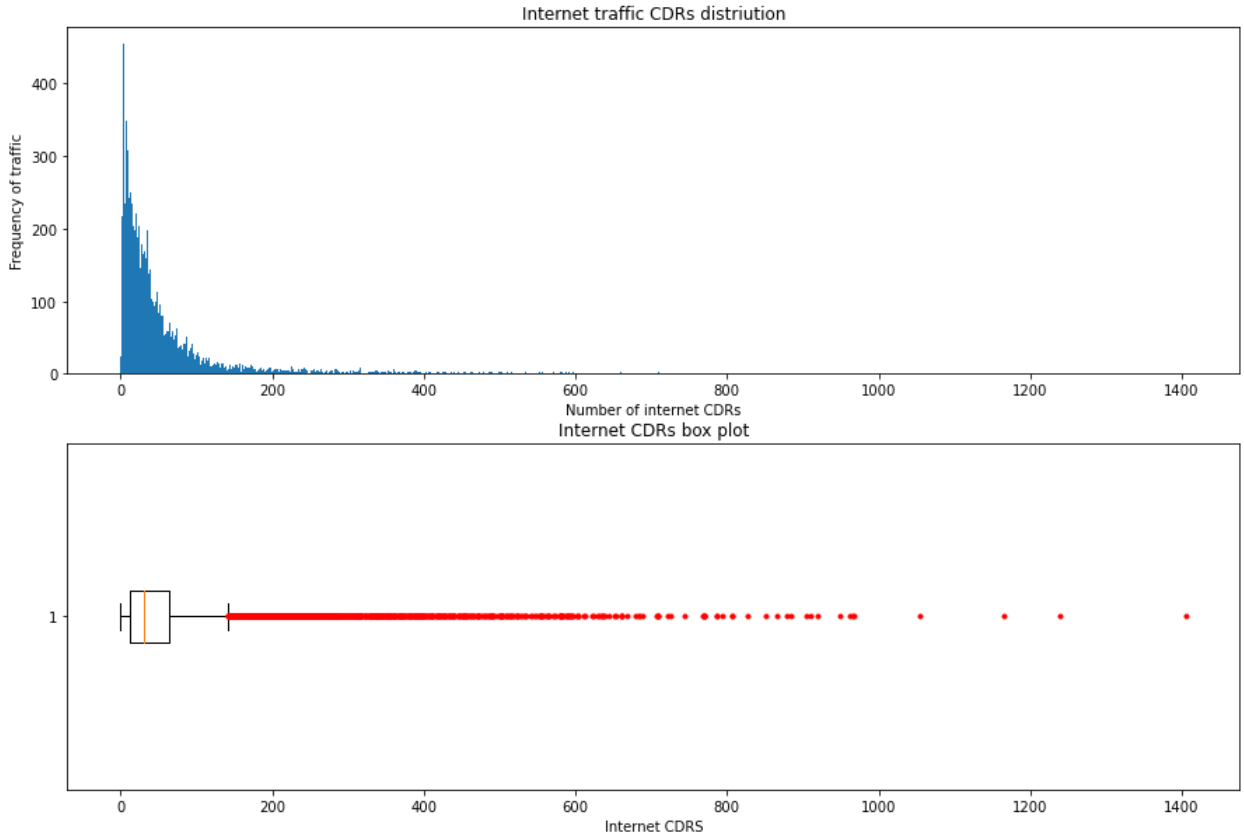


Figure 3. 6

As we can see this graph shows us that the average cell exhibits between 100 to 200 CDRs on average, but the most loaded cells may have more CDRs so we chose the grids of ids [5056, 5057, 5058, 5156, 5157, 5158, 5256, 5257, 5258] which we found they are the most interesting ones.

3.3.1.5 Excluding last 10 days:

As we mentioned before, the data is collected from 1st of November to the 1st of January so the last 10 days in Europe are the holidays of the new year, so these 10 days exhibit different loads and characteristics that we need a yearly data to model, which is not available for us.

For that reason, we have decided to exclude these last 10 days to unbiased the results of the modeling process.

3.3.1.6 Decomposition:

Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category. We have three types of time series patterns: trend, seasonality and cycles (noise).

When we decompose a time series into components, we usually combine the trend and cycle into a single trend-cycle component.

Thus, we think of a time series as comprising three components: a trend-cycle component, a seasonal component, and a remainder component (containing anything else in the time series).

Often this is done to help improve understanding of the time series, but it can also be used to improve forecast accuracy. And the aim of time series decomposition is to decompose a non-stationary time series X = {X1, X2, ...} = into non-stationary effects (the deterministic components) and a remaining component (the stochastic constituent).

Autocorrelation is the key concept that allows future prediction in time series. Informally, autocorrelation is a measure that reflects the degree of dependence between the values of a time series over successive time intervals.

Otherwise stated, a high autocorrelation means that values are highly dependent on their lagged values.

So, by using any model form Chapter 2 we need to see the main components of the

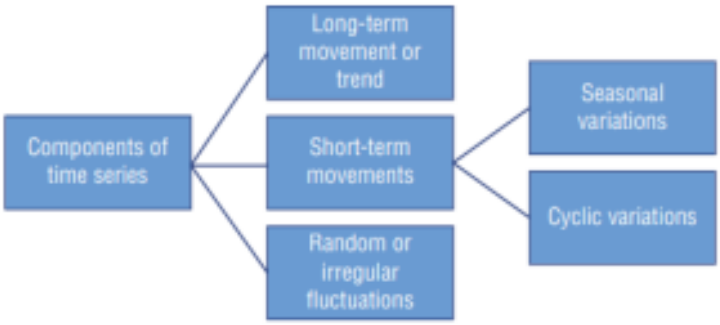


Figure 3. 7

time series.

3.3.1.6.1 Classical decomposition

- Additive Decomposition:

    - $Yt = St + Tt + Rt$, where Yt is the data, St is the seasonal component, Tt is the trend-cycle component, and Rt is the remainder component, all at period t.
    - The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series.

- Multiplicative Decomposition:

- $Yt = St \times Tt \times Rt$, where Yt is the data, St is the seasonal component, Tt is the trend-cycle component, and Rt is the remainder component, all at period t.
- Multiplicative decompositions are common with economic time series.
- When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate.

3.3.1.6.2 SEATS decomposition:

Seasonal Extraction in ARIMA Time Series, it's an ARIMA model-based approach.

In principle, it aims to derive the components with statistical models.

3.3.1.6.2 STL decomposition

STL is a versatile and robust method for decomposing time series. STL is an acronym for "Seasonal and Trend decomposition using Loess", while Loess is a method for estimating nonlinear relationships. The STL method was developed by R. B. Cleveland, Cleveland, McRae, & Trepanning.

STL has several advantages over the classical, SEATS decomposition methods:

• Unlike SEATS, STL will handle any type of seasonality, not only monthly and quarterly data.

• The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.

• The smoothness of the trend-cycle can also be controlled by the user.

• It can be robust to outliers (i.e., the user can specify a robust decomposition), so that occasional unusual observations will not affect the estimates of the trend-cycle and seasonal components. They will, however, affect the remaining component.

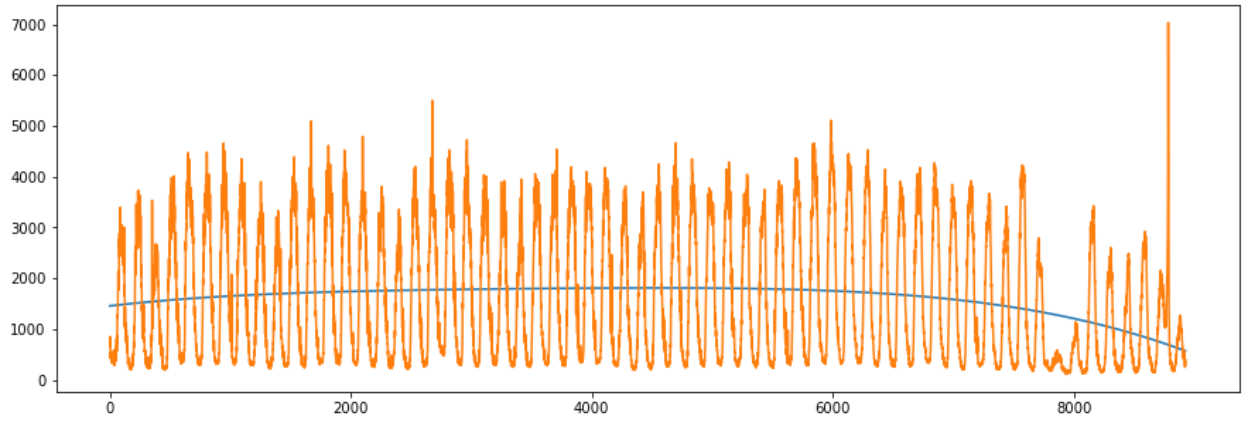


Figure 3. 8 Trend Component

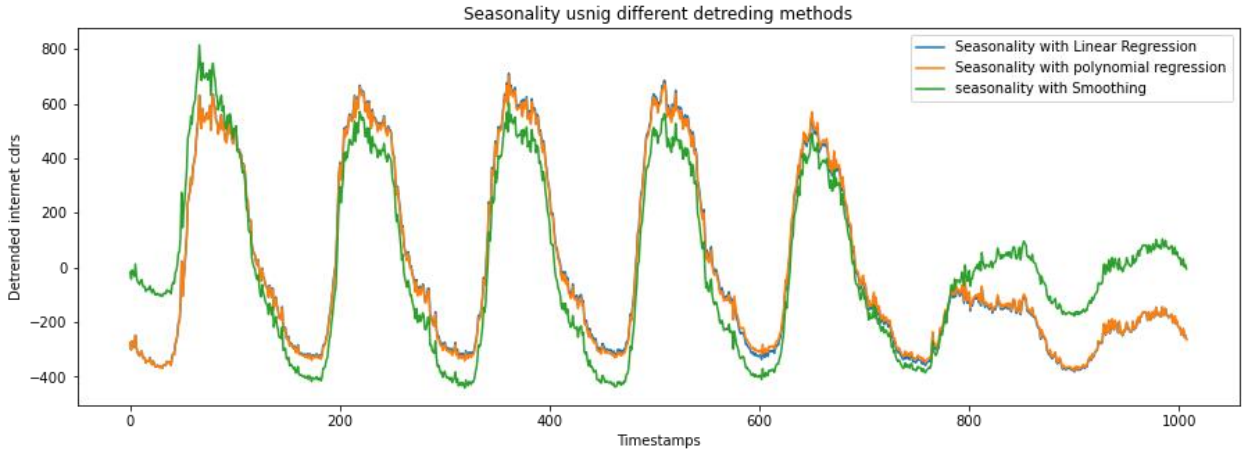


Figure 3. 9 Seasonality component

3.3.1.7 Stationarity:

A stochastic process X is stationary if its statistical properties (e.g., mean, covariance, etc.) are time-independent. It also may by either strong stationarity or weak stationarity as we discussed in chapter 2.

3.3.1.7.1 Testing for the Stationarity

Testing the stationarity for a time series is of great importance for its modeling.

To assess the stationarity of a stochastic process X based on a realization, we can use either graphical methods or statistical tests.

3.3.1.7.1.1 Graphical Methods

I. Analysis of time series plot: Stationarity can be visually assessed from time series plots on which prominent seasonality, trend and changes in variances are investigated.

II. Plotting rolling statistics: Rolling statistics graphs plot means and variances computed within sliding time windows. Such graphs can provide insights on stationarity.

Indeed, if these graphs exhibit time-invariance, then the studied time series will probably be stationary.

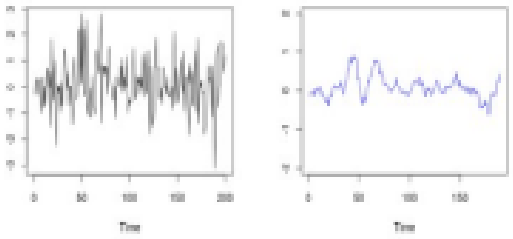


Figure 3. 10

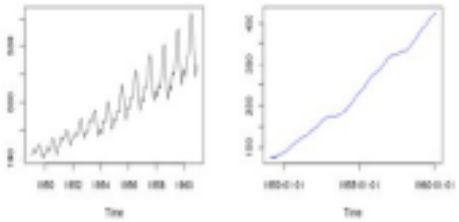


Figure 3. 11

Figure 3.10, 3.11 Rolling mean method applied to examples of stationary and nonstationary time series. Window size equals 12. (a) White noise process N (0, 1) where N stands for Gaussian law (left), rolling mean graph (right). (b) USA monthly airline numbers of passengers (left), rolling mean graph (right).

III. Correlogram vs. covariogram: In time series analysis, a correlogram and a covariogram are respectively the graphs of the autocorrelation function (acf) and of the scaled autocovariance function.

These functions are defined as:

$$\rho(h) = \frac{Cov(Xt, Xt+h)}{\sqrt{Var(Xt)\,Var(Xt+h)}}\,(autocorrelation\,function), \tag{3.2}$$

$$\rho(h) = \frac{Cov(Xt, Xt+h)}{Var(Xt)}\,(autocorrelation\,function) \tag{3.3}$$

3.3.1.7.1.2 Statistical Tests

Here is a short reminder about statistical tests. A statistical test is used to decide between two hypotheses, the null hypothesis H0 and the alternative hypothesis H1, where H0 is the hypothesis considered true if no evidence is brought against it.

- Dickey-Fuller test (KPSS):
    - Null Hypothesis: Time Series is stationary. It gives a time-dependent trend.
    - Alternate Hypothesis: Time Series is non-stationary. In another term, the series doesn't depend on time.

ADF or t Statistic < critical values: Accept the null hypothesis. Time series is stationary.

ADF or t Statistic > critical values: Failed to reject the null hypothesis. The time series is non-stationary.

3.3.2 Anomaly Detection

3.3.2.1 Introduction:

In data analysis, anomaly detection (also referred to as outlier detection and sometimes as novelty detection) is generally understood to be the identification of rare items, events or observations which deviate significantly from the majority of the data and do not conform to a well-defined notion of normal behavior. Such examples may arouse suspicions of being generated by a different mechanism, or appear inconsistent with the remainder of that set of data.

Figure 3. 12 Wild image highlighting anomaly data points

Anomaly detection finds application in many domains including cyber security, medicine, machine vision, statistics, neuroscience, law enforcement and financial

fraud to name only a few. Anomalies were initially searched for clear rejection or omission from the data to aid statistical analysis, for example to compute the mean or standard deviation. They were also removed to better predictions from models such as linear regression, and more recently their removal aids the performance of machine learning algorithms. However, in many applications anomalies themselves are of interest and are the observations most desirous in the entire data set, which need to be identified and separated from noise or irrelevant outliers.

3.3.2.2 What Are Anomalies?

Anomalies can be classified generally in several ways:

- Network anomalies: Anomalies in network behavior deviate from what is normal, standard, or expected. To detect network anomalies, network owners must have a concept of expected or normal behavior. Detection of anomalies in network behavior demands the continuous monitoring of a network for unexpected trends or events.

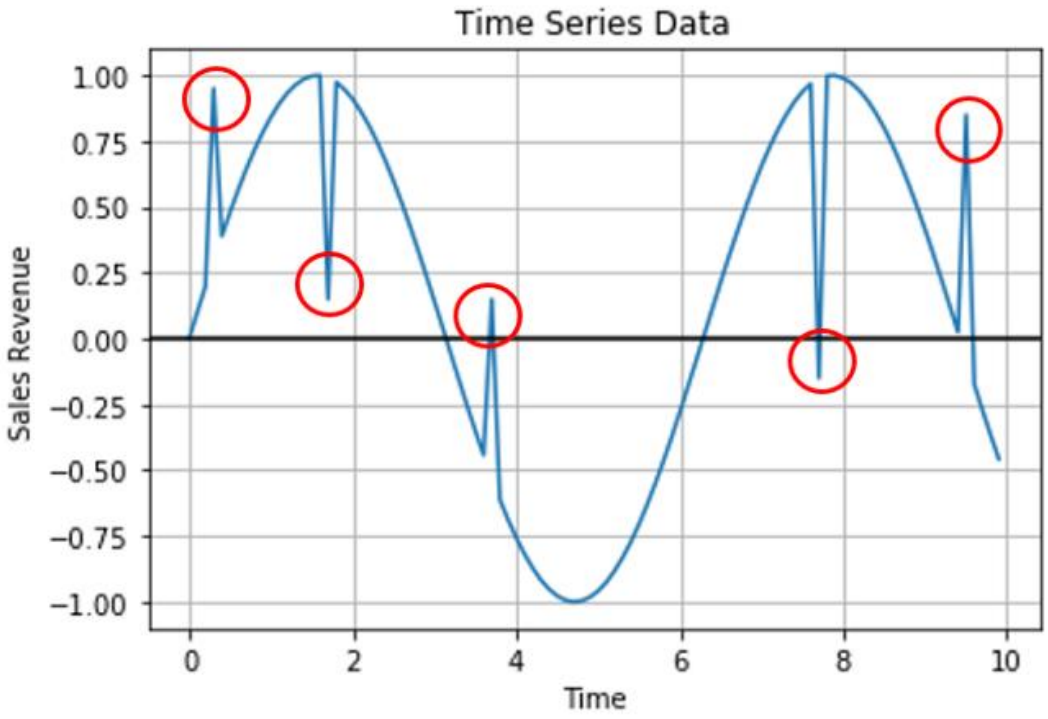


Figure 3. 13 Time Series plot highlighting anomaly data points

- Application performance anomalies: These are simply anomalies detected by end-to-end application performance monitoring. These systems observe application function, collecting data on all problems, including supporting infrastructure and app dependencies. When anomalies are detected, rate limiting is triggered and admins are notified about the source of the issue with the problematic data.
- Web application security anomalies: These include any other anomalous or suspicious web application behavior that might impact security such as CSS attacks or DDOS attacks.

Detection of each type of anomaly relies on ongoing, automated monitoring to create a picture of normal network or application behavior. This type of monitoring

might focus on point anomalies/global outliers, contextual anomalies, and/or collective anomalies; the context of the network, the performance of the application, or the web application security is more important to the goal of the anomaly detection system.

Anomaly detection and novelty detection or noise removal are similar, but distinct. Novelty detection identifies patterns in data that were previously unobserved so users can determine whether they are anomalous. Noise removal is the process of removing noise or unneeded observations from a signal that is otherwise meaningful.

To track monitoring KPIs such as bounce rate and churn rate, time series data anomaly detection systems must first develop a baseline for normal behavior. This enables the system to track seasonality and cyclical behavior patterns within key datasets.

3.3.2.3 Why Anomaly Detection Is Important?

It is critical for network admins to be able to identify and react to changing operational conditions. Any nuances in the operational conditions of data centers or cloud applications can signal unacceptable levels of business risk. On the other hand, some divergences may point to positive growth.

Therefore, anomaly detection is central to extracting essential business insights and maintaining core operations. Consider these patterns—all of which demand the ability to discern between normal and abnormal behavior precisely and correctly:

An online retail business must predict which discounts, events, or new products may trigger boosts in sales which will increase demand on their web servers.

An IT security team must prevent hacking and needs to detect abnormal login patterns and user behaviors.

A cloud provider has to allot traffic and services and has to assess changes to infrastructure in light of existing patterns in traffic and past resource failures.

An evidence-based, well-constructed behavioral model can not only represent data behavior, but also help users identify outliers and engage in meaningful predictive analysis. Static alerts and thresholds are not enough, because of the overwhelming scale of the operational parameters, and because it's too easy to miss anomalies in false positives or negatives.

To address these kinds of operational constraints, newer systems use smart algorithms for identifying outliers in seasonal time series data and accurately forecasting periodic data patterns.

3.3.2.4 Anomaly Detection Use Cases:

Anomaly detection has wide applications across industries. Below are some of the popular use cases:

- **Banking.** Finding abnormally high deposits. Every account holder generally has certain patterns of depositing money into their account. If there is an outlier to this pattern the bank needs to be able to detect and analyze it, e.g. for money laundering.
- **Finance.** Finding the pattern of fraudulent purchases. Every person generally has certain patterns of purchases which they make. If there is an outlier to this pattern the bank needs to detect it in order to analyze it for potential fraud.

- **Healthcare.** Detecting fraudulent insurance claims and payments.
- **Manufacturing.** Abnormal machine behavior can be monitored for cost control. Many companies continuously monitor the input and output parameters of the machines they own. It is a well-known fact that before failure a machine shows abnormal behaviors in terms of these input or output parameters. A machine needs to be constantly monitored for anomalous behavior from the perspective of preventive maintenance.
- **Networking.** Detecting intrusion into networks. Any network exposed to the outside world faces this threat. Intrusions can be detected early on using monitoring for anomalous activity in the network.

3.3.2.5 Why perform anomaly detection?

Anomaly detection has the following benefits:

- It helps with fraud detection. Anomaly detection will identify unusual and suspicious events in time series. Anomaly detection models are applied in banks and other financial institutions to detect fraud. The anomaly detection model will identify suspicious activities and transactions.
- It detects sudden spikes and drops in the time series dataset. Anomaly detection will explain the sudden spikes and drops in the dataset. We will also be able to gain valuable insights from the dataset. The sudden spikes and drops may lead to inconsistent results during forecasting.
- It identifies noise in the dataset. Noise is unwanted and erroneous data points. Noisy data has meaningless information that may corrupt the time series model in training. Removing the noise will ensure we have a high-quality dataset.

- It helps in identifying failures/malfunctioning in applications and devices. The model identifies unexpected changes in time-series data that record devices and applications' performance. The unexpected changes may be associated with device failure or malfunctioning.
- It helps in network intrusion and network anomaly detection. Models that monitor the network traffic will detect sudden changes in network traffic. These sudden changes may be due to cyber-attacks and other unauthorized access.

So, Anomaly detection is the process of discover the event or the points which are unexpected at this position of the dataset or deviates from the normal pattern of the dataset.

So, the detection of those points very important; because it give us an early step to make the emergency movements to control that un usual change.

3.3.2.6 Anomaly Detection Techniques

In searching data for anomalies that are relatively rare, it is inevitable that the user will encounter relatively high levels of noise that could be similar to abnormal behavior. This is because the line between abnormal and normal behavior is typically imprecise, and may change often as malicious attackers adapt their strategies.

Furthermore, because many data patterns are based on time and seasonality, there is additional baked-in complexity to anomaly detection techniques. The need to

break down multiple trends over time, for example, demands more sophisticated methods to identify actual changes in seasonality versus noise or anomalous data.

For all of these reasons, there are various anomaly detection techniques. Depending on the circumstances, one might be better than others for a particular user or data set. A generative approach creates a model based solely on examples of normal data from training and then evaluates each test case to see how well it fits the model. In contrast, a discriminative approach attempts to distinguish between normal and abnormal data classes. Both kinds of data are used to train systems in discriminative approaches.

We used many techniques to reach best one to apply it on our way of the project and those the best ones:

3.3.2.6.1 Tukey's box plot Method:

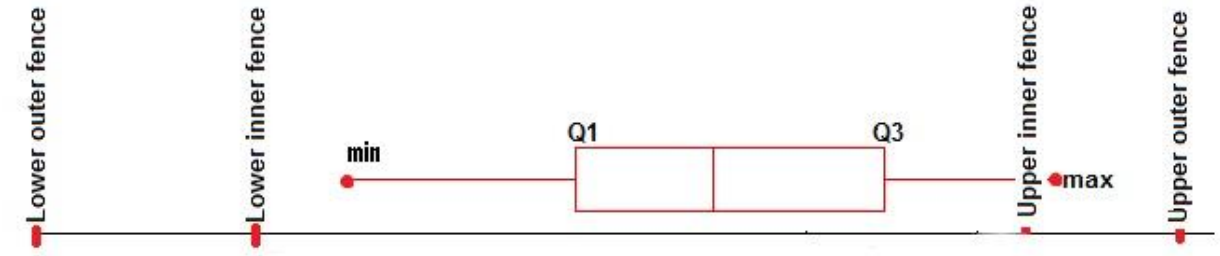


Figure 3. 14 Example of a box plot including the inner and outer fences and minimum and maximum observations

Tukey's box plot is the statistical method of anomaly detection in this method we depend on the pox plot to determine if the point is outlier or not and not only that it gives us the ability to decide if this outlier is possible or probable outlier point; by calculate the following parameters:

25th percentile: Q_1

75th percentile: Q_3

interquartile range: $IQR = Q_3 - Q_1$ (3.4)

Lower inner fence: $Q_1 - (1.5 * IQR)$ (3.5)

Upper inner fence: $Q_3 + (1.5 * IQR)$ (3.6)

Lower outer fence: $Q_1 - (3 * IQR)$ (3.7)

Upper outer fence: $Q_3 + (3 * IQR)$ (3.8)

Then decide if the point between the inner fence and outer fence it considered as a possible outlier point. And if the point lies outside the outer fence, it will be considered as probable outlier.

The following two graphs indicate what we said. on a random grid of the whole dataset. The red points are the anomalies points of this grid.

The other two graphs indicate the anomalies points on the whole dataset.

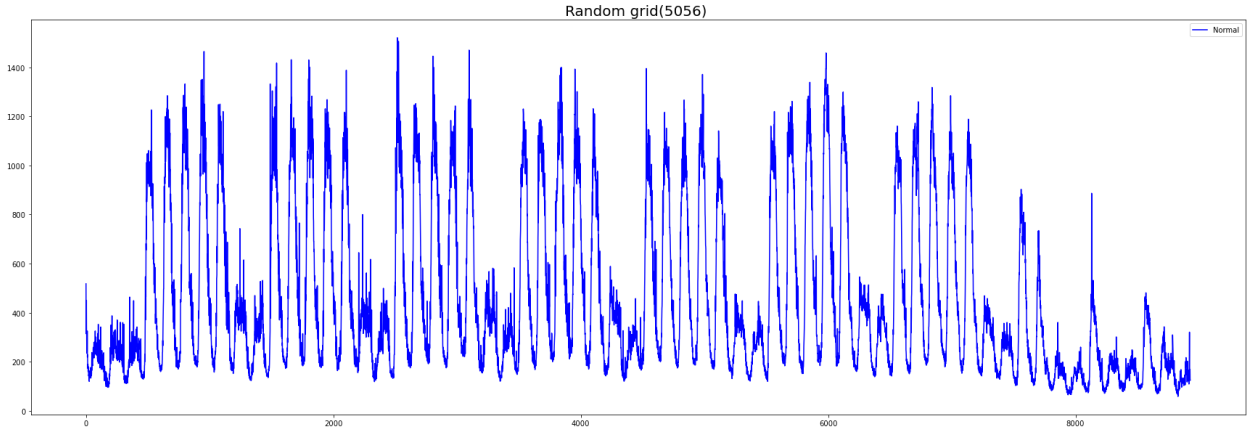


Figure 3. 15 Random grid (5056)

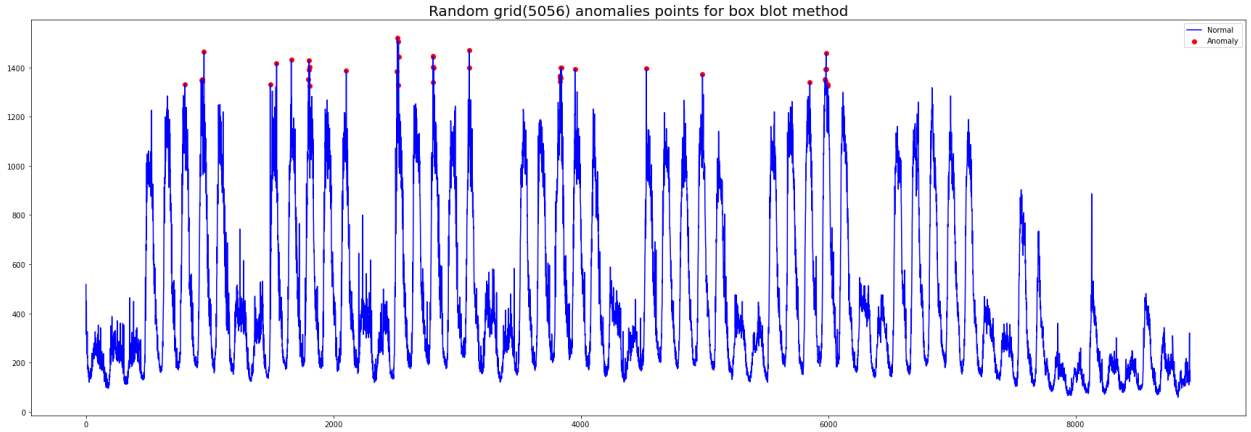


Figure 3. 16 Random grid anomalies points for box plot method

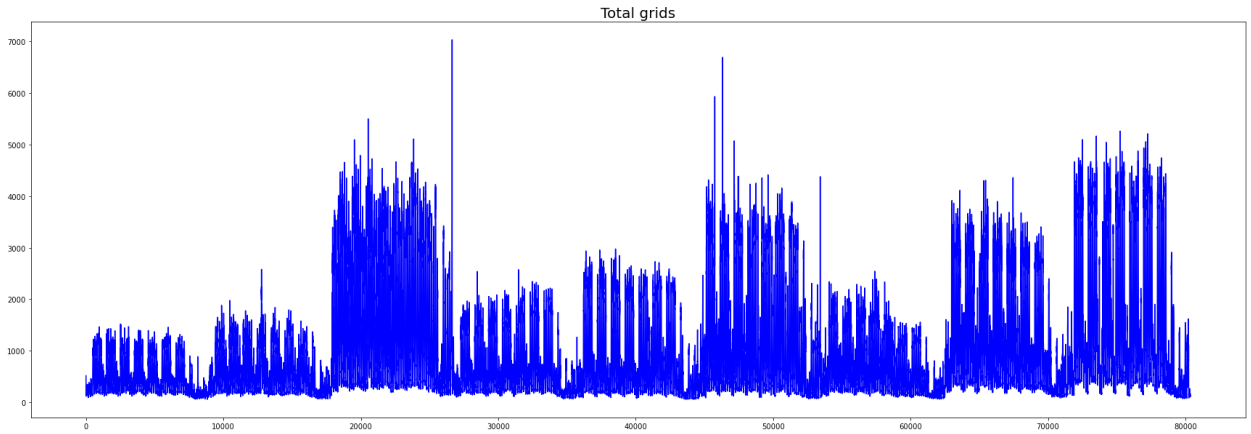


Figure 3. 17 Total 9 grids

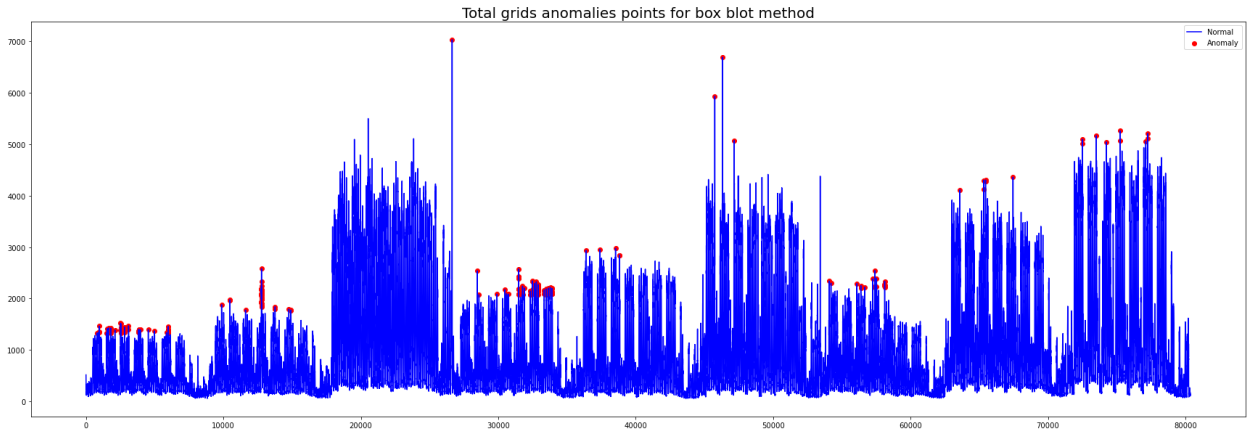


Figure 3. 18 Total grids anomalies points for box plot method

After using the box plot method as we shown before we will indicate the cons and pros of this method.

Pros:

- Quick representation and very fast method
- Easy to implement

Cons:

- The number of points is so small (215 anomalies points from total dataset points 80253).
- Not efficient

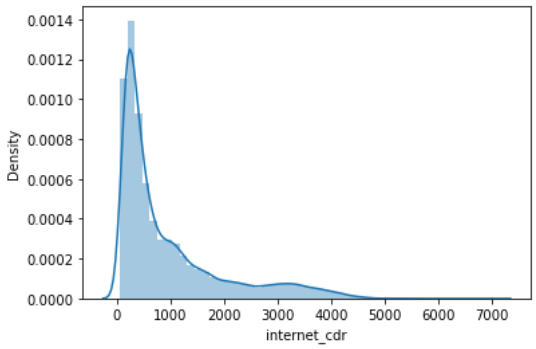


Figure 3. 19 distribution plot

- If the data is skewed it will miss some anomalies like our data.

3.3.2.6.2 Isolation Forest Method:

In this method we will depend on the detection using some Machine Learning algorithms. One of those algorithms is the Isolation Forest method.

Isolation Forest build using the decision trees which depend on the points that go deeper into the tree are not anomalies and points which go short distance have big probability to be anomalies, and it is unsupervised learning model which used without labeled data.

How the Isolation Forest Algorithm Works

The Isolation Forest Algorithm takes advantage of the following properties of anomalous samples (often referred to as outliers):

- **Fewness** — anomalous samples are a minority and there will only be a few of them in any dataset.
- **Different** — anomalous samples have values/attributes that are very different from those of normal samples.

These two properties make it easier to isolate anomalous samples from the rest of the data in comparison to normal points.

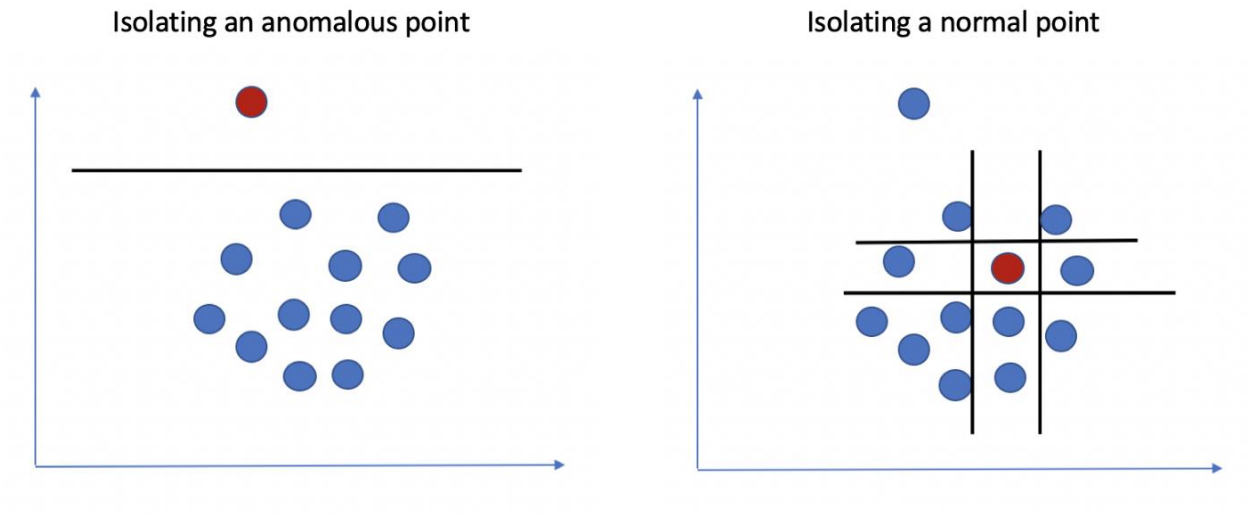


Figure 3. 20 Isolating an anomalous point versus a normal point

The Algorithm

The algorithm goes by selecting a sample of the dataset then branch it on the binary tress by setting a threshold if the sample we selected is less than this threshold it will be in the left branch and if it not it will be in the right branch. This process repeated until we every point in the dataset is isolated.

Given a sample of data points *X*, the Isolation Forest algorithm builds an Isolation Tree *T*, using the following steps.

1. Randomly select an attribute *q* and a split value *p*.
2. Divide *X* into two subsets by using the rule $q < p$. The subsets will correspond to a left subtree and a right subtree in *T*.
3. Repeat steps 1–2 recursively until either the current node has only one sample or all the values at the current node have the same values.

The algorithm then repeats steps 1–3 multiple times to create several Isolation Trees, producing an Isolation Forest. Based on how Isolation Trees are produced and the properties of anomalous points, we can say that most anomalous points will be located closer to the root of the tree since they are easier to isolate when compared to normal points.

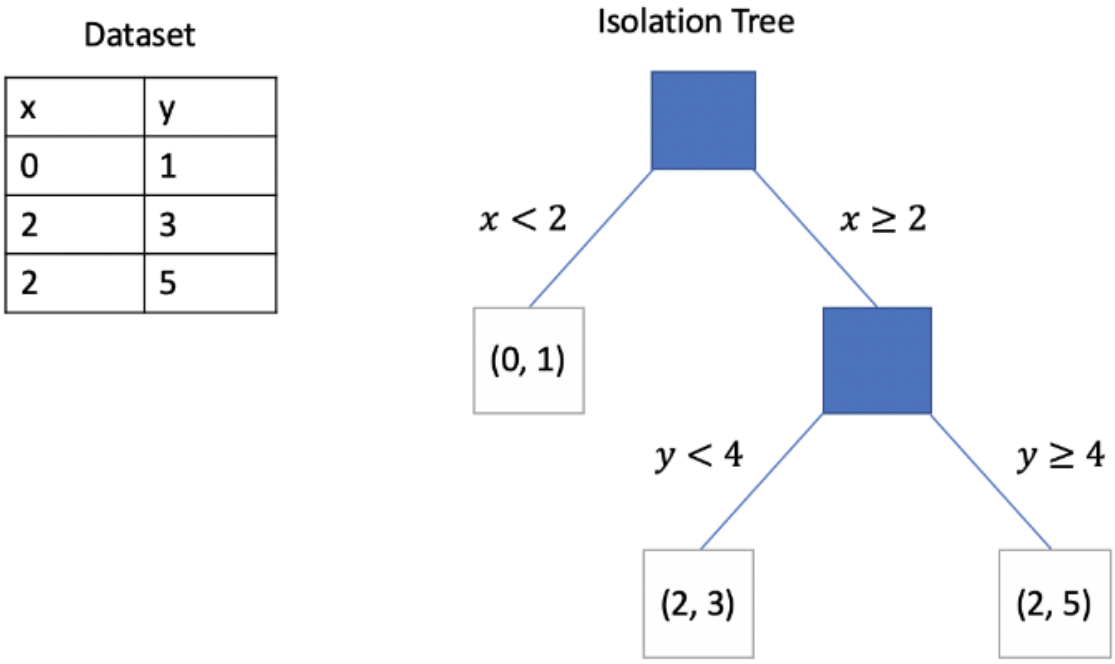


Figure 3. 21 An example of an isolation tree created from a small dataset

Once we have an Isolation Forest (a collection of Isolation Trees) the algorithm uses the following anomaly score given a data point x and a sample size of m:

$$S(x,m) = 2^{\frac{-E(h(x))}{c(m)}} \qquad (3.9)$$

In the equation above, h(x) represents the path length of the data point x in a given Isolation Tree. The expression E(h(x)) represents the expected or "average" value of this path length across all the Isolation Trees. The expression c(m) represents

the average value of h(x) given a sample size of m and is defined using the following equation.

$$c(m) = \begin{cases} 2H(m-1) - \dfrac{2(m-1)}{n} & , for\ m > 2 \\ 1 & , for\ m = 2 \\ 0 & , othewise \end{cases} \qquad (3.10)$$

The equation above is derived from the fact that an Isolation Tree has the same structure as a binary search tree. The termination of a node in an Isolation Tree is similar to an unsuccessful search in a binary search tree as far as the path length is concerned. Once the anomaly score *S(x, m)* is computed for a given point, we can detect anomalies using the following criteria:

1. If *S (x, m)* is close to 1 then *x* is very likely to be an anomaly.
2. If *S (x, m)* is less than 0.5 then *x* is likely a normal point.
3. If *S (x, m)* is close to 0.5 for all of the points in the dataset then the data likely does not contain any anomalies.

After building the algorithm we reached to the following output:

The following two graphs indicate what we said. on a random grid of the whole dataset. The red points are the anomalies points of this grid.

The other two graphs indicate the anomalies points on the whole dataset.

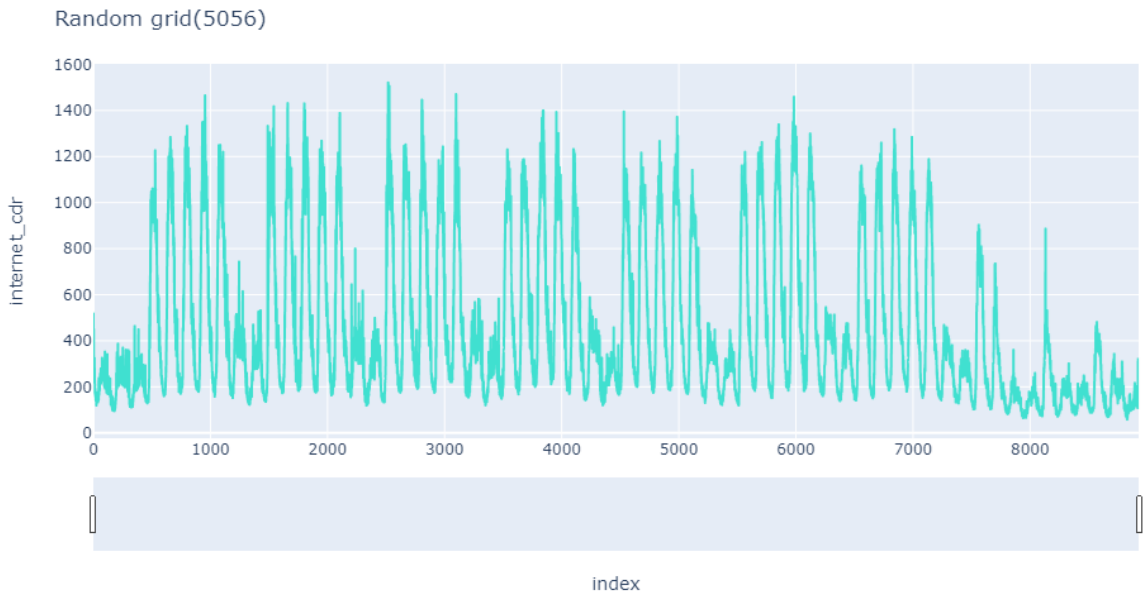


Figure 3. 22 random grid (5056)

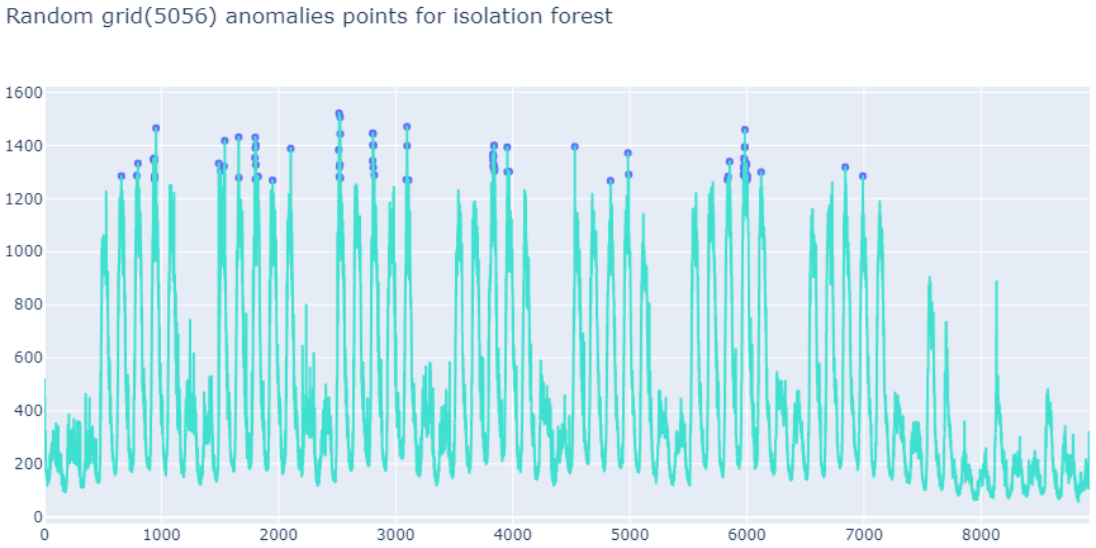


Figure 3. 23 Random grid anomalies points for isolation forest method

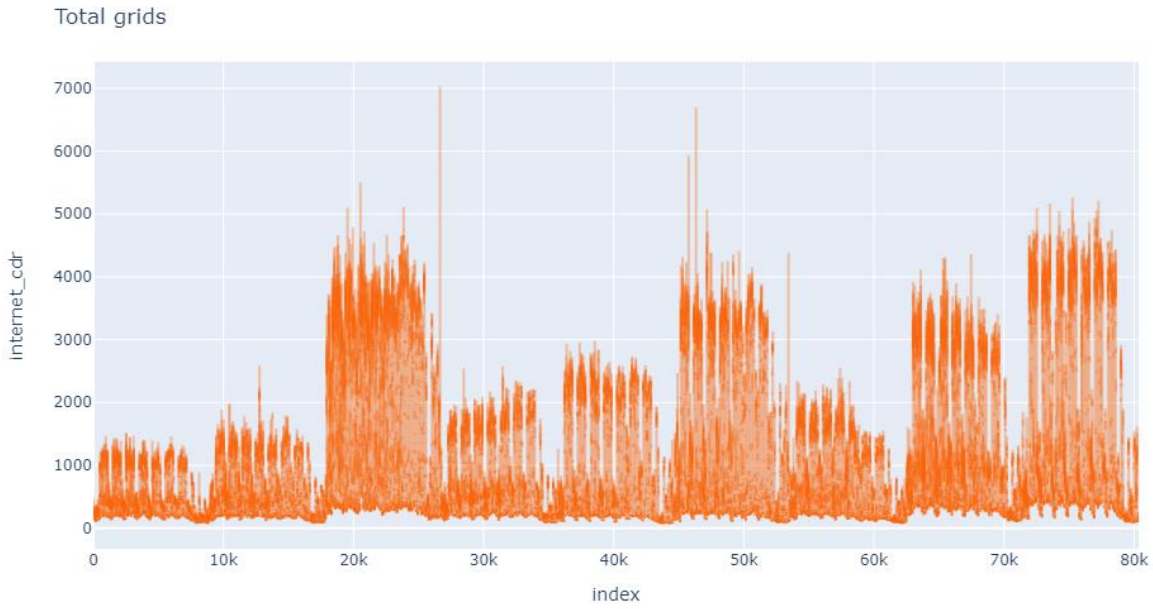


Figure 3. 24 Total 9 grids

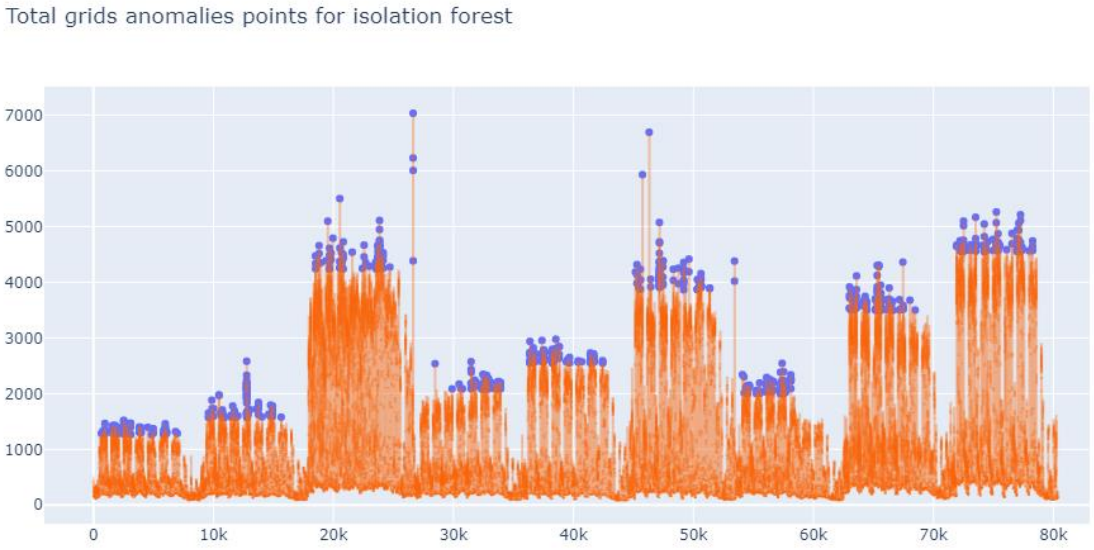


Figure 3. 25 Total grids anomalies points for isolation forest method

By looking on the previous graphs and results of the isolation forest method we will indicate the advantages and disadvantages of it.

Pros:

- We could use any number of the features which will make the model more accurate.
- detect anomalies faster
- require less memory compared to other algorithms
- It gives bigger number of the anomaly's points (1258 anomalies points from total dataset points 80253) than the box plot method.

Cons:

- There is biasing for the mode due to the brunching process.
- The greater number of features is good for the model but it will affect badly on the performance and the speed of the model so we should use our feature carefully.

3.3.2.6.3 LSTM Autoencoders Method:

In this method we will depend on the detection using the **forecasting** by **Deep Learning** algorithms. In the forecasting methods we depend on predict the next point with the addition of some noise and make comparison of this point and the true point at this timestamp by finding the difference between the two points then add threshold finally find the anomalies by compare the difference of the two points with this threshold (we used the Mean absolute error MAE).

Autoencoders: are type of self-supervised learning model which are a neural network that learn from the input data. We use autoencoder because the Principal Component Analysis (PCA), which we used in the previous method we depend on the linear algebra to do the models, but by using autoencoders we depended on the non-linear transformation like by use the activation functions; those non-linearity gives us the ability to go deep in the number of the neural network layers.

Autoencoder is a neural network designed to learn an identity function in an unsupervised way to reconstruct the original input while compressing the data in the process so as to discover a more efficient and compressed representation. The idea was originated in the 1980s, and later promoted by the seminal paper by Hinton & Salakhutdinov, 2006.

It consists of two networks:

- *Encoder* network: It translates the original high-dimension input into the latent low-dimensional code. The input size is larger than the output size.
- *Decoder* network: The decoder network recovers the data from the code, likely with larger and larger output layers.

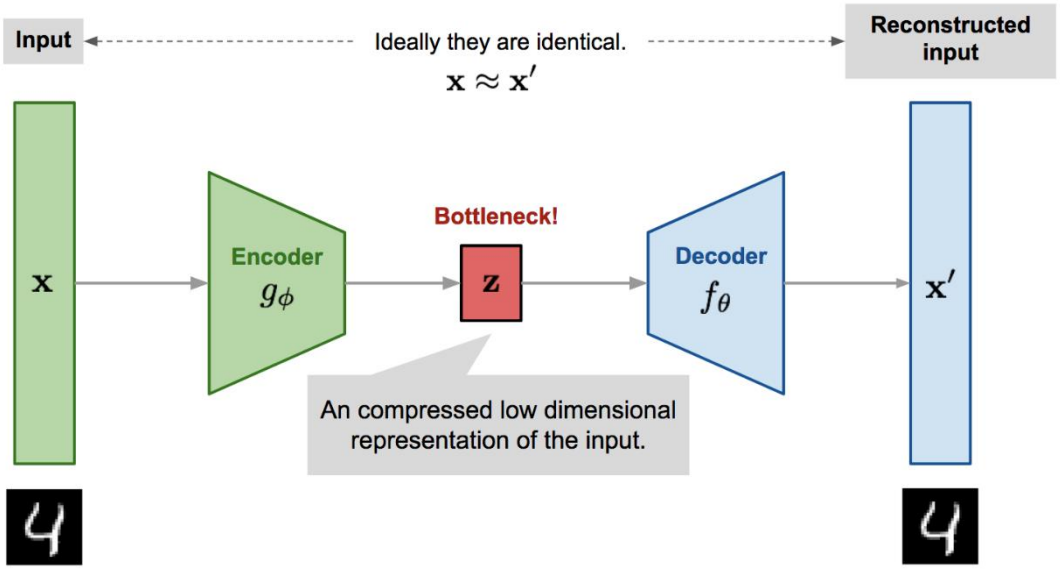


Figure 3. 26 Illustration of autoencoder model architecture.

The encoder network essentially accomplishes the dimensionality reduction, just like how we would use Principal Component Analysis (PCA) or Matrix Factorization (MF) for. In addition, the autoencoder is explicitly optimized for the data reconstruction from the code. A good intermediate representation not only can capture latent variables, but also benefits a full decompression process.

The model contains an encoder function $g(.)$ parameterized by ϕ and a decoder function f(.) parameterized by θ. The low-dimensional code learned for input x in the bottleneck layer is $z = g\phi(x)$ and the reconstructed input is $x' = f_\theta(g_\phi(x))$.

The parameters (θ, ϕ) are learned together to output a reconstructed data sample same as the original input, $x \approx f_\theta(g_\phi(x))$, or in other words, to learn an identity function. There are various metrics to quantify the difference between two vectors, such as cross entropy when the activation function is sigmoid, or as simple as MSE loss:

$$L_{AE}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - f_\theta(g_\phi(x^{(i)})))^2 \qquad (3.11)$$

Long Short-Term Memory (LSTM) is a type of artificial recurrent neural network (RNN). which are designed to handle sequential data, with the previous step's output being fed as the current step's input.

Let's say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly, RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they cannot remember long term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

LSTM Architecture

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.

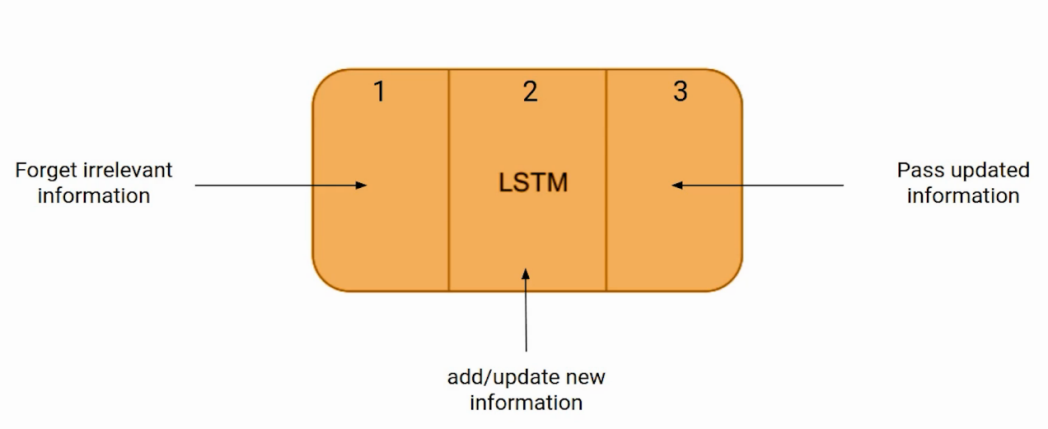


Figure 3. 27 the internal functioning of the LSTM network

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

These three parts of an LSTM cell are known as gates. The first part is called **Forget gate,** the second part is known as **the Input gate** and the last one is **the Output gate**.

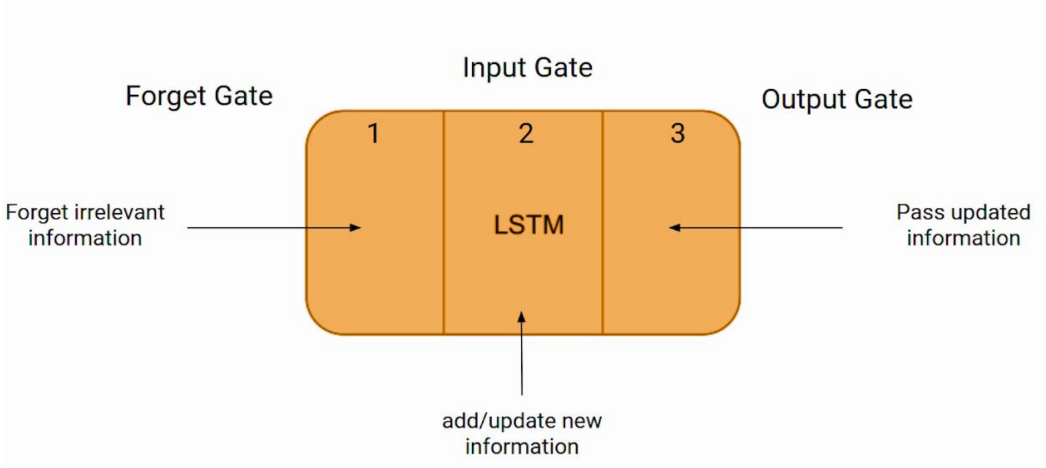


Figure 3. 28 the internal functioning of the LSTM network

Just like a simple RNN, an LSTM also has a hidden state where $H(t-1)$ represents the hidden state of the previous timestamp and H_t is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by $C(t-1)$ and $C(t)$ for previous and current timestamp respectively.

Here the hidden state is known as short term memory and the cell state is known as long term memory. Refer to the following image.

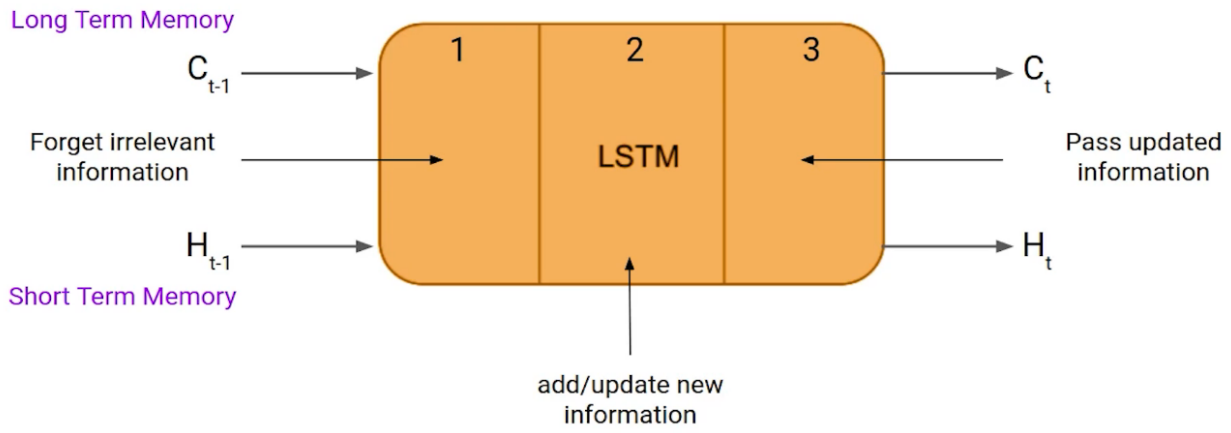
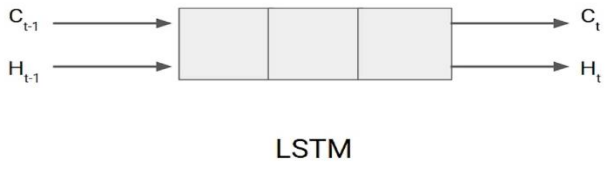


Figure 3. 29 the internal functioning of the LSTM network

It is interesting to note that the cell state carries the information along with all the timestamps.



Bob is a nice person. Dan on the other hand is evil.

Figure 3. 30 the internal functioning of the LSTM network

Let's take an example to understand how LSTM works. Here we have two sentences separated by a full stop. The first sentence is "Bob is a nice person" and the second sentence is "Dan, on the Other hand, is evil". It is very clear; in the first

sentence we are talking about Bob and as soon as we encounter the full stop(.) we started talking about Dan.

As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Bob. Now our subject is Dan. Here, the Forget gate of the network allows it to forget about it. Let's understand the roles played by these gates in LSTM architecture.

Forget Gate

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for forget gate.

$$f_t = \sigma\left(x_t * U_f + H_{t-1} * W_f\right) \qquad (3.12)$$

Let's try to understand the equation, here

- x_t: input to the current timestamp.
- U_f: weight associated with the input
- H_{t-1}: The hidden state of the previous timestamp
- W_f : It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make f_t a number between 0 and 1. This $f_t$is later multiplied with the cell state of the previous timestamp as shown below.

$$C_{t-1} * f_t = 0 \qquad if\ f_t = 0\ (forget\ everything) \quad (3.13)$$

$$C_{t-1} * f_t = C_{t-1} \quad if\ f_t = 1\ (forget\ nothing) \qquad (3.14)$$

If $f_t$is 0 then the network will forget everything and if the value of f_t is 1 it will forget nothing. Let's get back to our example, The first sentence was talking about Bob and after a full stop, the network will encounter Dan, in an ideal case the network should forget about Bob.

Input Gate

Let's take another example

"Bob knows swimming. He told me over the phone that he had served the navy for four long years."

So, in both these sentences, we are talking about Bob. However, both give different kinds of information about Bob. In the first sentence, we get the information that he knows swimming. Whereas the second sentence tells he uses the phone and served in the navy for four years.

Now just think about it, based on the context given in the first sentence, which information of the second sentence is critical. First, he used the phone to tell or he served in the navy. In this context, it doesn't matter whether he used the phone or any other medium of communication to pass on the information. The fact that he was in the navy is important information and this is something we want our model to remember. This is the task of the Input gate.

Input gate is used to quantify the importance of the new information carried by the input. Here is the equation of the input gate:

$$i_t = \sigma\left(x_t * U_i + H_{t-1} * W_i\right) \qquad (3.15)$$

Here,

- x_t: Input at the current timestamp t
- U_i: weight matrix of input
- H_{t-1}: A hidden state at the previous timestamp
- W_i : Weight matrix of input associated with hidden state

Again, we have applied sigmoid function over i_t. As a result, the value of i at timestamp t will be between 0 and 1.

New information

$$N_t = \tanh(x_t * U_c + H_{t-1} * W_c) \qquad (3.16)$$

Now the new information that needed to be passed to the cell state is a function of a hidden state at the previous timestamp $t-1$ and input x at timestamp t. The activation function here is $tanh$. Due to the $tanh$ function, the value of new information will between -1 and 1. If the value is of N_t is negative the information

is subtracted from the cell state and if the value is positive the information is added to the cell state at the current timestamp.

However, the N_t won't be added directly to the cell state. Here comes the updated equation

$$C_t = f_t * C_{t-1} + i_t * N_t \qquad (3.17)$$

Here, C_{t-1} is the cell state at the current timestamp and others are the values we have calculated previously.

Output Gate

Now consider this sentence

"Bob single-handedly fought the enemy and died for his country. For his contributions, brave_____ ."

During this task, we have to complete the second sentence. Now, the minute we see the word brave, we know that we are talking about a person. In the sentence only Bob is brave, we cannot say the enemy is brave or the country is brave. So based on the current expectation we have to give a relevant word to fill in the blank. That word is our output and this is the function of our Output gate.

Here is the equation of the Output gate, which is pretty similar to the two previous gates.

$$O_t = \sigma\left(x_t * U_o + H_{t-1} * W_o\right) \qquad (3.18)$$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use O_t and $tanh$ of the updated cell state. As shown below.

$$H_t = O_t * \tanh(c) \qquad (3.19)$$

It turns out that the hidden state is a function of long-term memory (C_t) and the current output. If you need to take the output of the current timestamp just apply the SoftMax activation on hidden state Ht.

Here the token with the maximum score in the output is the prediction.

This is the More intuitive diagram of the LSTM network.

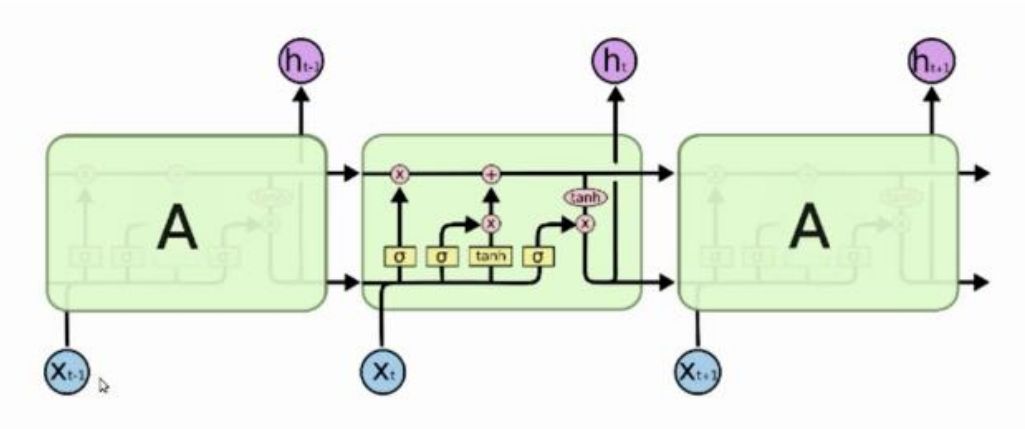


Figure 3. 31 The repeating module in an LSTM contains four interacting layers.

We apply some dimensionality reduction on our dataset by use encoder to make the dimension small then use the decoder to get it back and that minimize the reconstruction loss. In fact, that will make us lose some information but it gives us the ability to know the main pattern of the information and thought that we could define any information out hits pattern under sone threshold will be outlier.

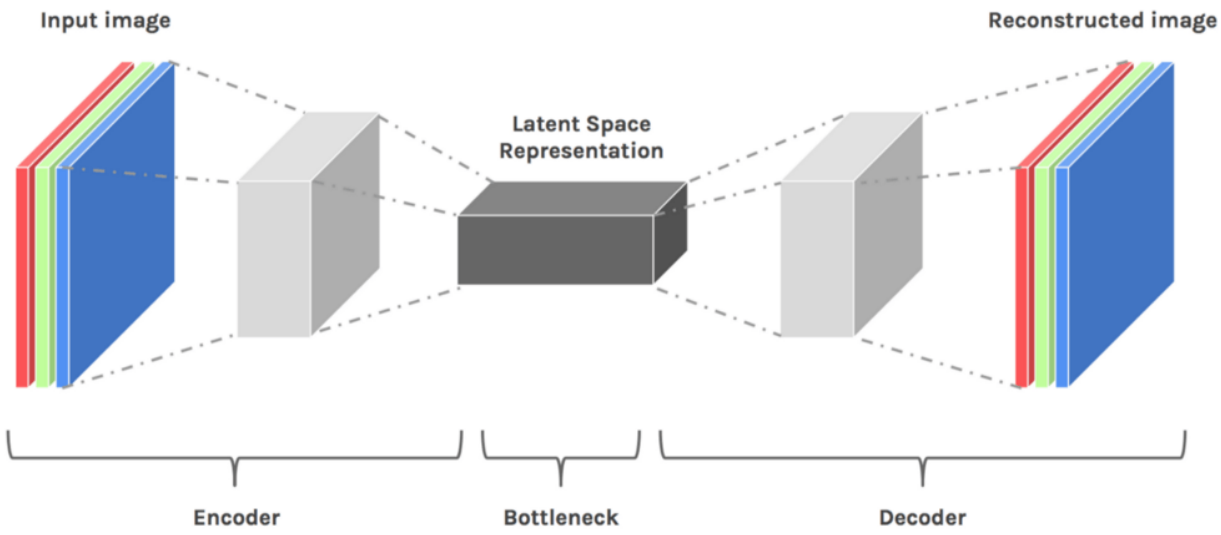


Figure 3. 32 Dimensionality Reduction diagram

After applying the model, we get the following results:

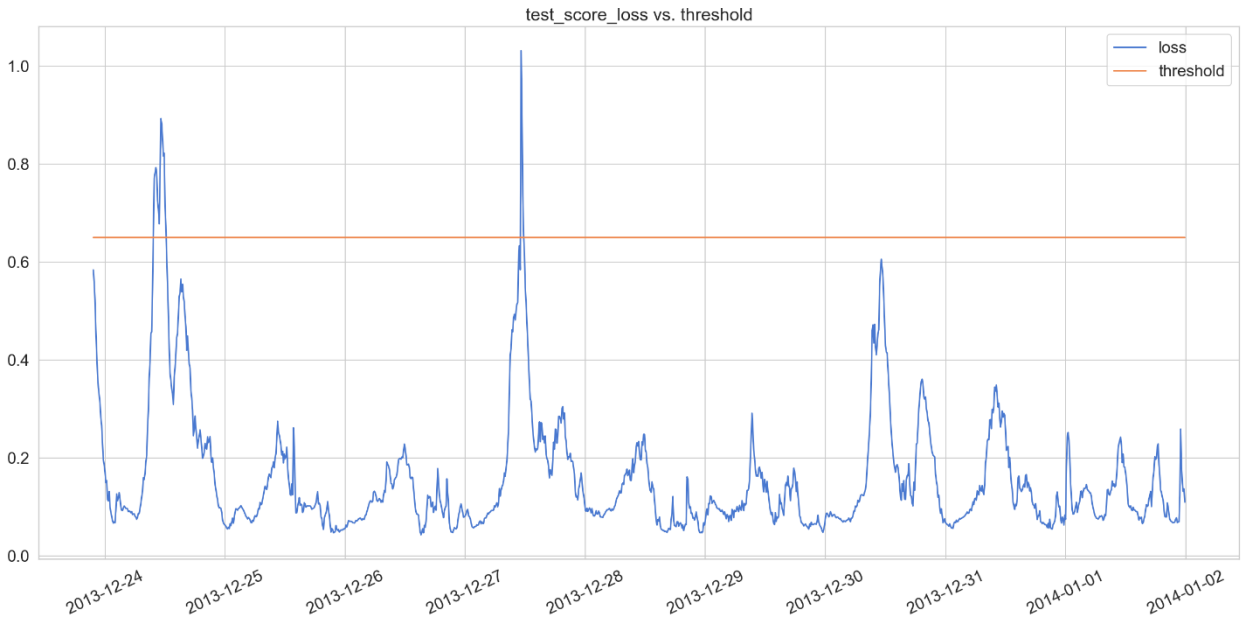


Figure 3. 33 test score loss vs. threshold

To determine the cutoff point we use the Mean Absolute Error (MAE). We use the MAE because it so sensitive toward outliers. MAE find the mean absolute error

between the actual value y and predicted value $\hat{y}$ of the dataset then find the threshold like the following equation:

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} |y - \hat{y}_i| \qquad (3.20)$$

When we apply the threshold to the predicted values which will give us the anomalies at the points which corresponding to the locations of the signal which above the threshold line the previous graph, we get the following graph for the anomalies.

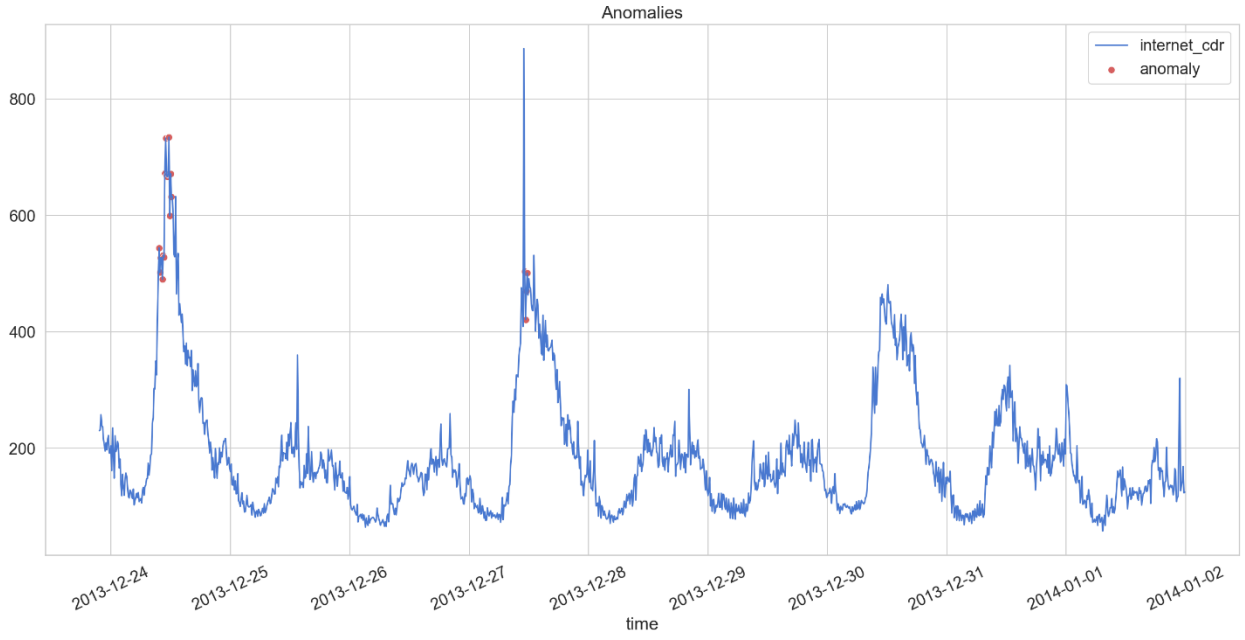


Figure 3. 34 predicted values vs. the anomalies for the LSTM Autoencoders method

Pros:

- Autoencoder could easily deals with data which have high dimensions.
- As the good use of the activation function, it gives big ability to deal with complex datasets.

Cons:

- Need big data to train the deep learning models.
- High cost and low speed when deal with those big datasets.

3.3.2.6.4 Seasonal-Trend Decomposition Method:

Now we will go to the final method which is decomposition. Signal decomposition aims to analysis our signal to its main three components Seasonal, trend and the residual (S, T, R). Seasonal is the signal component which contain the most rapidly pattern which occurs regular every cerin time. Trend contain the general shape of the data over the whole dataset and finally the residual is the rest of the signal after extract the seasonal and trend of it, it is in somehow a random part over the signal which indicate it.

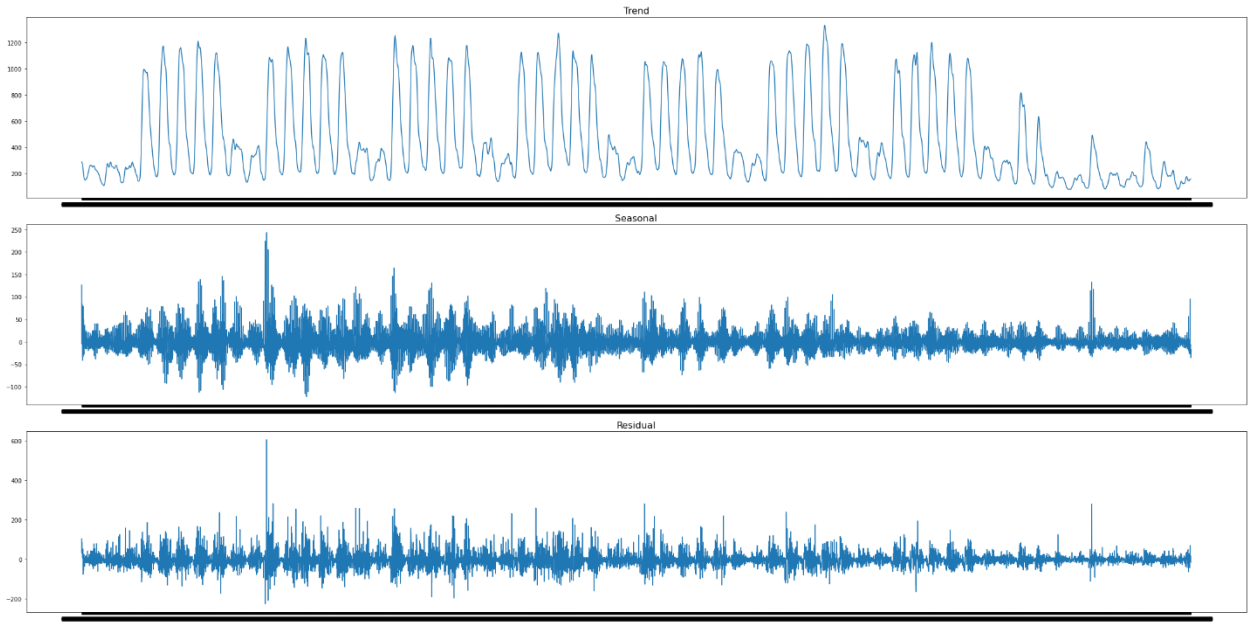


Figure 3. 35 random grid 5056 decomposition three parts (S, T, R)

To make the residual more obvious to us we will plot the signal with and without the residual component as the following:

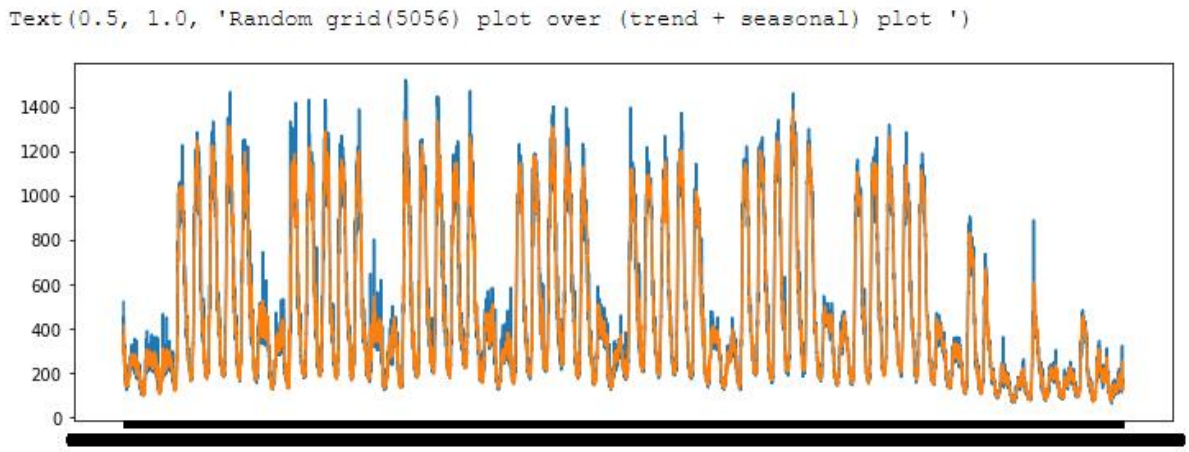


Figure 3. 36 signal with and without the residual component

The residual will be our focus here, we will first analysis the signal to its main three component and take the residual to work on it.

We will apply the model by define the threshold which depend on the he confidence interval, then apply it for the residual then decide if this point is an anomaly or not.

We will define certain two limits lower limit and the upper limit on the dataset points. First is the lower limit which define through this equation:

$$Lower = \mu - 3 * S \qquad (3.21)$$

Second is the upper limit which define through this equation:

$$Upper = \mu + 3 * S \qquad (3.22)$$

After applying those constrain on the residual component, we will have the ability to define if this point is anomaly or not.

By input every grid of the nine grids on this condition we will the following outputs one for rand grid (5056) and after combine the all nine grids we will have the other graph.

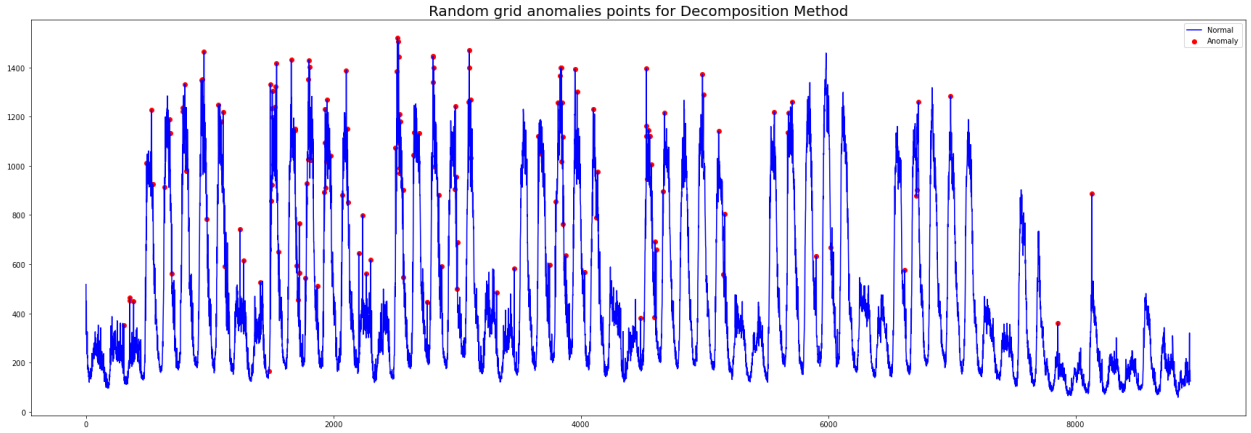


Figure 3. 37 Random grid (5056) anomalies points for Decomposition method

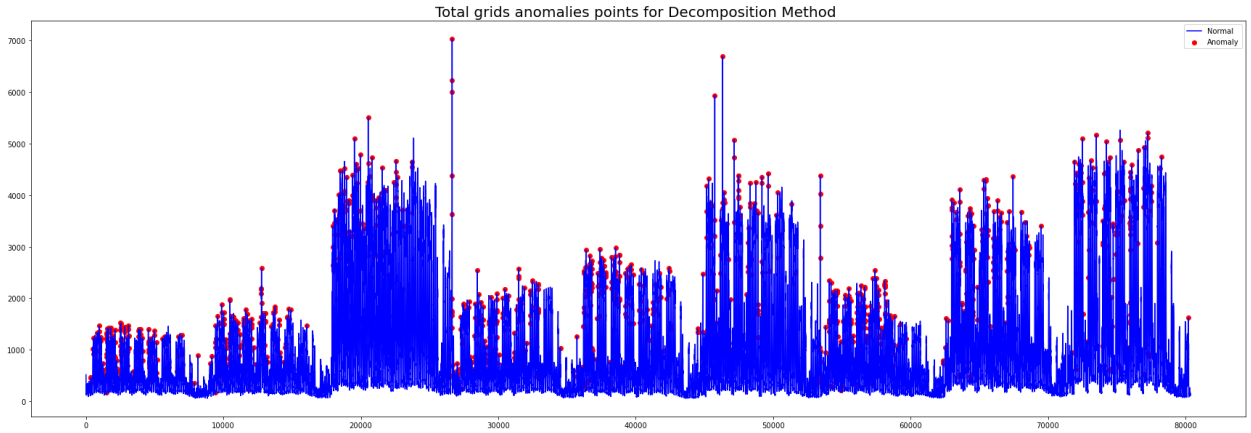


Figure 3. 38 Total grids anomalies points for Decomposition method

Let us give our intuition of the results of this method by showing its cons and pros:

Pros:

- So simple method didn’t need to the forecasting to define the anomalies.
- Robust which could work under any type of the datasets.

Cons:

- If there is a suddenly change made by us on the process, we work we should study the process individually after and before this change happen; as we using the confidence interval to determine the threshold.

3.3.2.7 What is next?

After determining the anomalies points what we should do about them? In the most common application like Microsoft anomaly detection, they have some web application to send some emails to the concerned persons about the sudden and didn't expected change, then they take the right decision to solve this issue.

But in our scope here we concerned with if this point is outlier or point of interest.

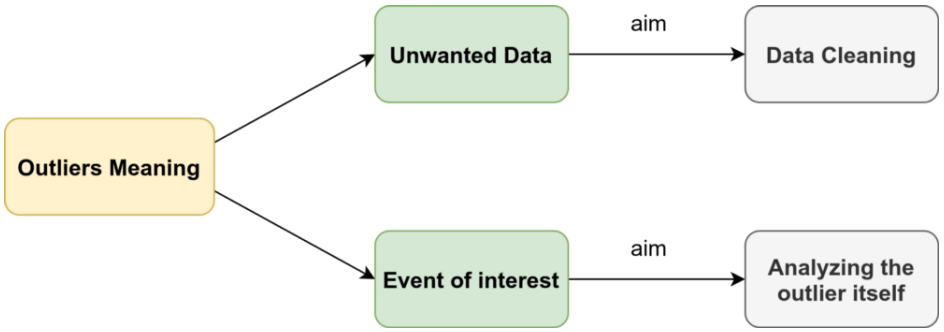


Figure 3. 39 Outliers Road

So, if the point we detected as outlier is a data that we did not need we will do on it some cleaning data processing. But in the other hand some time did not mean if there is some unusual event that we didn't need it. Some event most be important to us because might this event will happen in the future so by studying it will make us have the ability to avoid this sudden change in the future by handle it by some control flow process.

Here we after studying those points we will make all anomalies point as a nan value so will handle it in the coming part, **Missing value imputation**.

3.3.3 Missing data imputation:

Time series forecasting has become an important aspect of data analysis and has many real-world applications. However, undesirable missing values are often encountered, which may adversely affect many forecasting tasks. The needs of data completeness of the observation data for the uses of advanced analysis becomes important to be solved.

The missing data existence an affect the modeling in several ways, one of them is that most models rely on having a past measurements to forecast future ones, also another one is that it can prevent us to study the seriates' different behaviors, i.e., Autocorrelation function (ACF) needs the past values to be determined, partial autocorrelation function (PACF) needs the same thing.

So, the completeness of the data not only affects the forecasting models, but also prevents us from knowing the true behavior of the series.

And in our dataset, we have some missing data ratios as in the following figure:

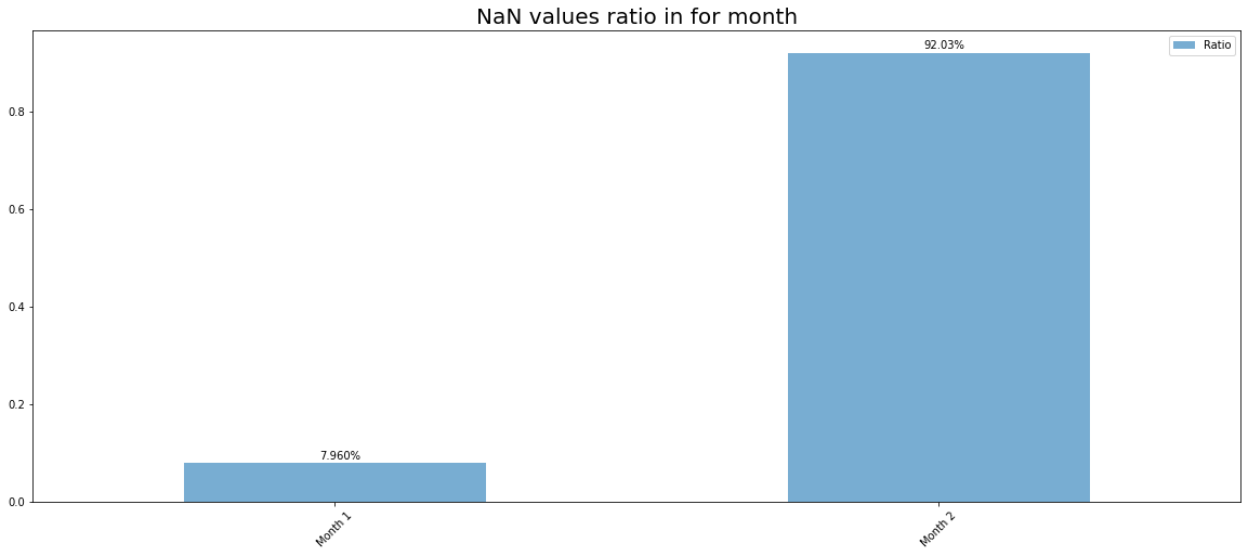


Figure 3. 40 Missing data ratios in first and second month

As we can see, the first month of the data has only around 8% of the missing data ratio, and the most of the missing data are in the second month so we need to go deeply behind this missing data.

3.3.3.1 Causes of missingness:

There is a broad distinction between two types of missing data: *intentional* and *unintentional* missing data. Intentional missing data are planned by the data collector. For example, the data of a unit can be missing because the unit was excluded from the sample. Another form of intentional missing data is the use of different versions of the same instrument for different subgroups, an approach known as matrix sampling. See Gonzalez and Eltinge (2007) or Graham (2012 Section 4) for an overview. Also, missing data that occur because of the routing in a questionnaire are intentional, as well as data (e.g., survival times) that are censored data at some time because the event (e.g., death) has not yet taken place. A related term in a multilevel context is systematically missing data. This term refers to variables that are missing for all individuals in a cluster because the variable was not measured in that cluster. (Resche-Rigon and White 2018)

Though often foreseen, unintentional missing data are unplanned and not under the control of the data collector. Examples are: the respondent skipped an item, there was an error in the data transmission causing data to be missing, some of the objects dropped out before the study could be completed resulting in partially complete data, and the respondent was sampled but refused to cooperate. A related term in a multilevel context is sporadically missing data. This term is used for variables with missing values for some but not all individuals in a cluster.

Another important distinction is *item nonresponse* versus *unit nonresponse*. Item nonresponse refers to the situation in which the respondent skipped one or more items in the survey. Unit nonresponse occurs if the respondent refused to participate, so all outcome data are missing for this respondent. Historically, the methods for item and unit nonresponse have been rather different, with unit nonresponse primarily addressed by weighting methods, and item nonresponse primarily addressed by edit and imputation techniques.

3.3.3.2 Types of missing data:

Before we review a number of simple fixes for the missing data in Section 1.3 let us take a short look at the terms MCAR, MAR and MNAR. Rubin (1976) classified missing data problems into three categories. In his theory every data

point has some likelihood of being missing. The process that governs these probabilities is called the missing data mechanism or response mechanism. The model for the process is called the missing data model or response model.

If the probability of being missing is the same for all cases, then the data are said to be missing completely at random (MCAR). This effectively implies that causes of the missing data are unrelated to the data. We may consequently ignore many of the complexities that arise because data are missing, apart from the obvious loss of information. An example of MCAR is a weighing scale that ran out of batteries. Some of the data will be missing simply because of bad luck. Another example is when we take a random sample of a population, where each member has the same chance of being included in the sample. The (unobserved) data of members in the population that were not included in the sample are MCAR. While convenient, MCAR is often unrealistic for the data at hand.

If the probability of being missing is the same only within groups defined by the observed data, then the data are missing at random (MAR). MAR is a much broader class than MCAR. For example, when placed on a soft surface, a weighing scale may produce more missing values than when placed on a hard surface. Such data are thus not MCAR. If, however, we know the surface type and if we can assume MCAR within the type of surface, then the data are MAR. Another example of MAR is when we take a sample from a population, where the probability to be included depends on some known property. MAR is more general and more realistic than MCAR. Modern missing data methods generally start from the MAR assumption.

If neither MCAR nor MAR holds, then we speak of missing not at random (MNAR). In the literature one can also find the term NMAR (not missing at random) for the same concept. MNAR means that the probability of being missing varies for reasons that are unknown to us. For example, the weighing scale mechanism may wear out over time, producing more missing data as time progresses, but we may fail to note this. If the heavier objects are measured later in time, then we obtain a distribution of the measurements that will be distorted. MNAR includes the possibility that the scale produces more missing values for the

heavier objects (as above), a situation that might be difficult to recognize and handle. An example of MNAR in public opinion research occurs if those with weaker opinions respond less often. MNAR is the most complex case. Strategies to handle MNAR are to find more data about the causes for the missingness, or to perform what-if analyses to see how sensitive the results are under various scenarios.

Rubin's distinction is important for understanding why some methods will work, and others not. His theory lays down the conditions under which a missing data method can provide valid statistical inferences. Most simple fixes only work under the restrictive and often unrealistic MCAR assumption. If MCAR is implausible, such methods can provide biased estimates.

So, we can conclude these by the following:

- Missing Completely at Random (MCAR):
  When there is no relationship between the missing data and already observed data, so the probability of missing data is completely random and has no dependence on the observed data:

  $$P(Missing|CompleteData) = P(Missing) \quad (3.23)$$ https://www.codecogs.com/eqnedit.php?latex=%20P(Missing%20%7C%20CompleteData)%20%3D%20P(Missing)%20 - 0

- Missing at Random (MAR):
  A variable is missing at random if the probability of missingness depends only on the available information:

  $$P(Missing|\ CompleteData) = P(Missing\ |\ Observed) \quad (3.24)$$ https://www.codecogs.com/eqnedit.php?latex=%20P(Missing%20%7C%20CompleteData)%20%3D%20P(Missing%20%7C%20observed)%20 - 0

- Missing not at Random (MNAR):
  The probability of missingness, in this case, depends on the variable itself.

Time series models work with complete data, and therefore they require the missing data to be replaced with meaningful values before any analysis. At a high

level, missing values in time series are handled in two ways, either dropping them or replacing them. However, dropping missing values can be an inappropriate solution due to the time order of the data and the correlation between observations in adjacent periods.

3.3.3.3 Missing data imputation techniques:

We can divide all available approaches which can be used to handle missing data in time series into:

3.3.3.3.1 Conventional methods:

In these methods, we replacing the missing data with a well know value that is a characteristic of the series itself or we simply ignore the missing values and these methods are:

3.3.3.3.1.1 Mean Imputation:

We replace the missing data with the mean value of the series.

3.3.3.3.1.2 Median Imputation:

We replace the missing data with the median value of the series, and the median of the value is the value that divides the data equally into two parts, and it's more realistic than the mean as it's immune to outliers of the series.

3.3.3.3.1.3 Mode Imputation:

We replace the missing data with the most repetitive value in the series, but this method is not realistic in numerical values, as it's difficult to have the same identically real observed data, but we may have some very near values.

3.3.3.3.1.4 Deletion or ignorance:

By deleting the missing entries, but in time series this can be a very destructive approach as the sequence is important in time series.

And these techniques are not good enough to handle missing values as those methods can cause bias to the data. They do not work appropriately if the time series has seasonality and trend components. This is because the seasonality and trend components are not considered while imputing the missing data. Therefore, they can only work better if the observed time series has no seasonality or trend component. If the time series has these components, we need more sophisticated techniques.

And after we have applied these methods, we got the following errors with increasing the missing data ratio:

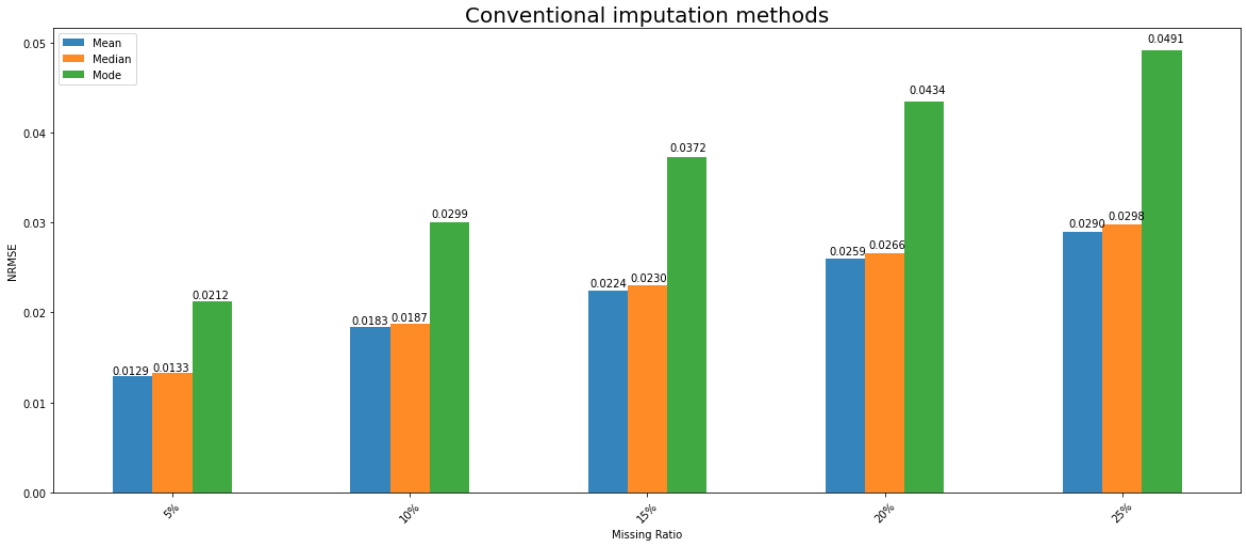


Figure 3. 41 Conventional imputation methods

As we can see the mode as expected is the worst thing to impute with as it rarely has repeated numerical values.

The mean and median have near error rates in different missing ratios but we need more accurate algorithms to have better accuracy in imputation.

3.3.3.3.2 Imputation procedures:

In these techniques we make some simple procedures to replace missing data, in these procedures we rely on the relationship between measurements in the nature of time series, as it represents the sequences.

3.3.3.3.2.1 Last valid observation:

We replace the missing data with the last valid measurement.

3.3.3.3.2.2 Next valid observation:

We replace the missing data with the next valid measurement.

3.3.3.3.2.3 Simple interpolation:

We use simple linear interpolation between the last and the next valid observations

3.3.3.3.2.4 Seasonal interpolation:

As we have seasonality, this can be used to make effective imputation, as the seasonality means that there is some sort of periodicity in measurements, and as we can see, our data is very seasonal.

So, by making interpolations between the next and last valid observations in the same timestamp over all the series we can make a good imputation.

And after we applied these techniques we have seen a great enhancements in the imputation accuracy:

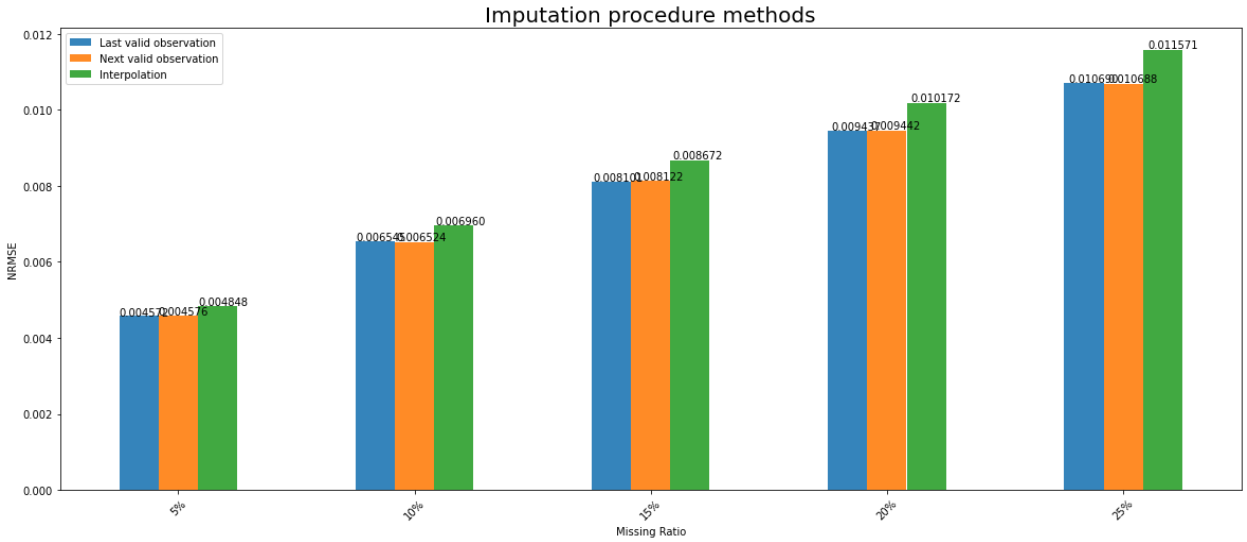


Figure 3. 42 Imputation procedure methods

As a comparison with the previous techniques with errors more than 2% we have seen that all of these methods have errors less than 1% NRMSE errors with different missing data ratios.

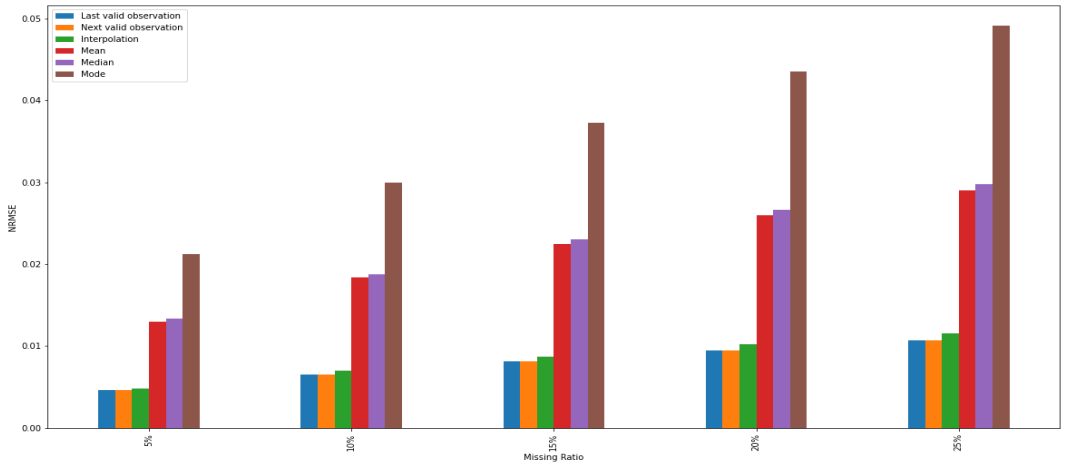


Figure 3. 43 Comparison between imputation procedures and conventional methods

3.3.3.3.3 Learnable methods:

These methods are more intelligent to know the relationship between the observations to impute the values with the best options.

3.3.3.3.3.1 K-Nearest Neighbors: (KNN)

Algorithm, where we apply the K-nearest neighbor algorithm to impute the missing values, which uses the nearest K points to the missed value to impute, as in the figure.

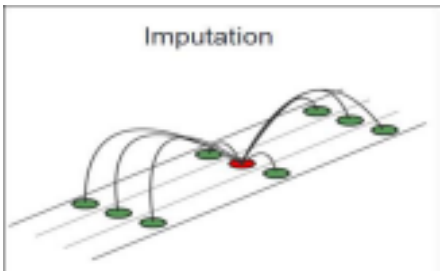


Figure 3. 44 KNN Imputation

3.3.3.3.3.2 MICE (Multiple imputation using chained equations):

It's a robust, informative method of dealing with missing data in datasets. The procedure imputes missing data in a dataset through an iterative series of predictive models. In each iteration, each specified variable in the dataset is imputed using the other variables in the dataset. These iterations should be run until it appears that convergence has been met.

The chained equation process can be broken down into four general steps:

- Step 1: A simple imputation, such as imputing the mean, is performed for every missing value in the dataset. These mean imputations can be thought of as “place holders”.
- Step 2: The “place holder” means imputations for one variable (“var”) are set back to missing.
- Step 3: The observed values from the variable “var” in Step 2 are regressed on the other variables in the imputation model, which may or may not consist of all of the variables in the dataset. In other words, “var” is the dependent variable in a regression model and all the other variables are independent variables in the regression model. These regression models operate under the same assumptions that one would make when performing linear, logistic, or Poisson regression models outside of the context of imputing missing data.
- Step 4: The missing values for “var” are then replaced with predictions (imputations) from the regression model. When “var” is subsequently used as an independent variable in the regression models for other variables, both the observed and these imputed values will be used.
- Step 5: Steps 2–4 are then repeated for each variable that has missing data. The cycling through each of the variables constitutes one iteration or “cycle.” At the end of one cycle all of the missing values have been replaced with predictions from regressions that reflect the relationships observed in the data.
- Step 6: Steps 2–4 are repeated for a number of cycles, with the imputations being updated at each cycle.

3.3.3.3.3.3 Generative Imputations Adversarial networks: (GAIN)

Which makes use of the well-known Generative Adversarial Nets (GAN) framework. So that, it’s called GAIN. The generator (G) observes some components of a real data vector, imputes the missing components conditioned on what is actually observed, and outputs a completed vector. The discriminator (D) then takes a completed vector and attempts to determine which components were actually observed and which were imputed. To ensure that D forces G to learn the desired distribution, we provide D with some additional information in the form of a hint vector. The hint reveals to D partial information about the missingness of the original sample, which is used by D to focus its

attention on the imputation quality of particular components. This hint ensures that G does in fact learn to generate according to the true data distribution. We tested our method on various datasets and found that GAIN significantly outperforms state-of-the-art imputation methods.

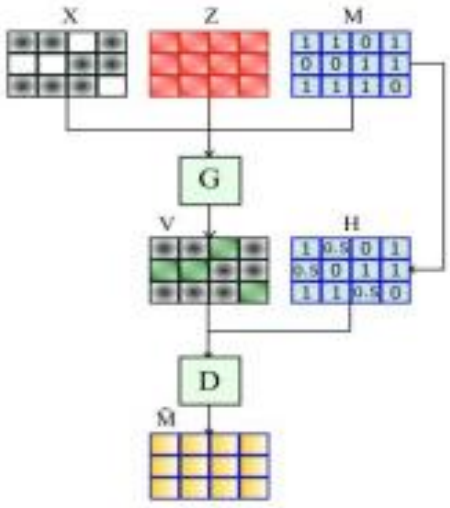


Figure 3. 45 GAIN

Summary of the GAIN procedures:

- Suppose that:
  - X = (X1, ..., Xd) is a random variable which is the data vector with distribution P(X).
  - M = (M1, ..., Md) is a random variable taking values in {0, 1}d, for each i ∈ {1, ..., d}, M is the mask vector.
  - X i˜ = Xi ∪ {∗} where ∗ is simply a point not in any Xi , representing an unobserved value

    $$.X\,I` = \begin{cases} XI, if\ Mi = 0 \\ ?\ otherwise \end{cases} \quad (3.25)$$
  - So that M indicates which components of X are observed. Note that we can recover M from X˜.

    In the imputation setting, n i.i.d. copies of X˜ are realized, denoted x1˜ , ..., xn˜ and we define the dataset , where mi is simply the recovered realization of M corresponding to x˜ i . We need to impute the unobserved values in each x˜i.

- **The generator, G:**

Takes (realizations of) X˜, M and a noise variable, Z, as input and outputs V, a vector of imputations. Then we define the random variables V, Xˆ ∈ X by :

- $V = G(X, M_i, (1 - M) \odot Z)$ (3.26)
- $V = M \odot +(1 - M) \odot V$ (3.27)

- **The Discriminator, D:**
  Will be used as an adversary to train G. However, unlike in a standard GAN where the output of the generator is either completely real or completely fake, in this setting the output is composed of some components that are real and some that are fake.
  Rather than identifying that an entire vector is real or fake, the discriminator attempts to distinguish which components are real (observed) or fake (imputed) - this amounts
  to predicting the mask vector, m.

GAIN	G:	Dense	[125]
		Dense	[125]
		Dense	[125]
		Dense	[125]
	D:	Dense	[125]
		Dense	[125]
		Dense	[125]
		Dense	[125]

Figure 3. 46 GAIN Implementation

And we choose to apply the following architecture to the network:

3.3.3.3.3.4 Convolutional Generative Imputations Adversarial networks: (ConvGAIN)

Also, a GAN-based technique, but the Generative Adversarial Imputation Nets (GAIN) performance is improved by applying convolutional neural networks instead of full layers to better capture the correlation of data.

The proposed structure:

The Generator and the Discriminator have the same architecture, each consists of two convolution layers (32, 64), right after each convolution layer a pooling layer is used to reduce the dimension. The filter size, 3×3 and number of channels for

each convolution layer is denoted in the following figure. The filter size 2 ×2 for the pooling layer is also described. After convolutional and pooling layers, the number of outputs is decreased gradually through two dense layers in the end of the neural network models so the models do not experience a dramatic change of

Conv-GAIN	G:	Convolution	[3,3,32]	[6, 125, 32]
		Pooling	[2,2]	[3, 63, 32]
		Convolution	[3,3,64]	[3, 63, 64]
		Pooling	[2,2]	[2, 32, 64]
		Data flatten		[4096]
		Dense		[1024]
		Dense		[375]
	D:	Convolution	[3,3,32]	[6, 125, 32]
		Pooling	[2,2]	[3, 63, 32]
		Convolution	[3,3,64]	[3, 63, 64]
		Pooling	[2,2]	[2, 32, 64]
		Data flatten		[4096]
		Dense		[1024]
		Dense		[375]
		Dense		[375]

Figure 3. 47 ConvGain Implementation

parameter size.

And after applying these methods we have found the following results:

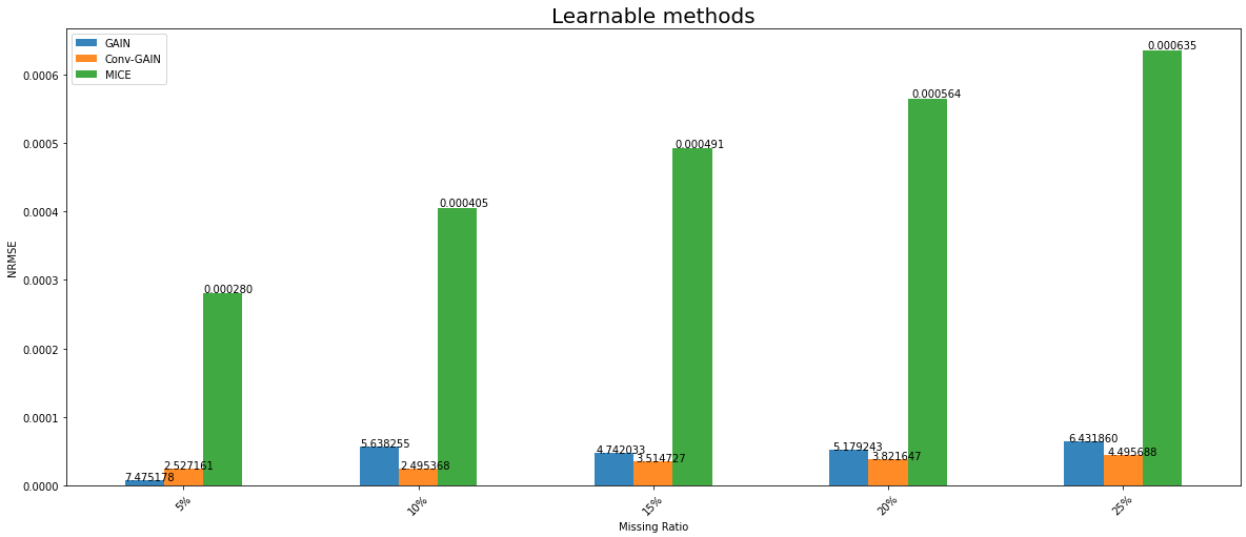


Figure 3. 48 Learnable methods imputation

As we can observe from here, these methods are more accurate than the previous ones, so these techniques are very effective as the NRMSE error is less than 0.06% which is very efficient.

3.3.3.4 Imputations Evaluation procedures:

In the previous sections we were evaluating the performance of each algorithm by using the following procedure:

- Add synthesized NaN values with ratios (5%, 10%, 15%, 20%, and 25%).
- Apply all methods for all grids with all of these ratios.
- Evaluate the performance of each method for all ratios using the NRMSE evaluation metrics.
- Compare different methods and decide which one to use.

3.3.3.5 imputations conclusion:

Missing values can be treated in various ways depending on the problem caused by the missing values. The method of missing values can affect the result of the analysis or data mining process. Estimation technique is probably the best option of missing values handling, since to accomplish certain work the complete dataset is required and some dataset have dependent variables which is impossible to delete the missing values as it can disrupt the data itself. Although the recent methods' performance is good, the better methods to cut out the time complexity or demanding computational resources required by certain methods.

And the results are as follows:

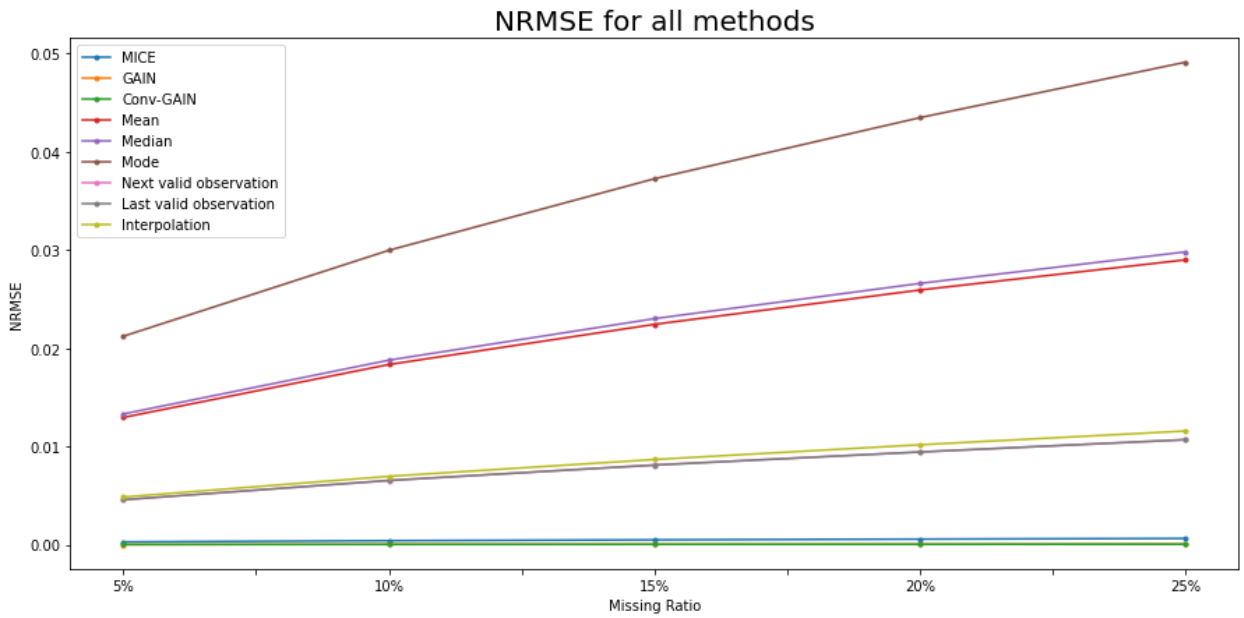


Figure 3. 49 NRMSE for All imputation Methods

We can see that, compared with other methods, the GAIN and ConvGAIN are the superstars in their accuracies and also immune to missing data ratio, which is good.

After we are sure about the data, as we went through different steps as in the following figure:

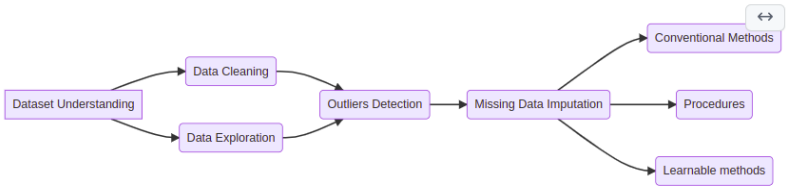


Figure 3. 50 Preprocessing steps

Next step is to make forecasts.

3.4 Forecasting:

We will do forecasting as in the following figure:

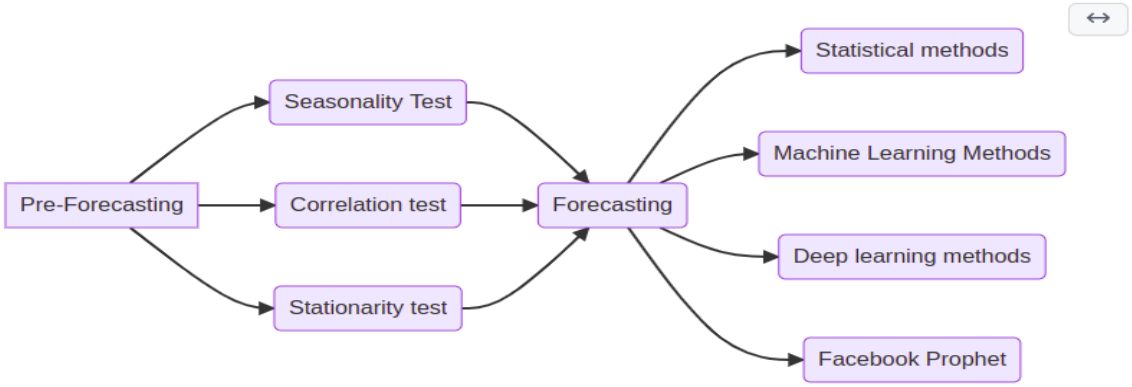


Figure 3. 51 Forecasting steps

Time series forecasting is a subdomain of time series analysis in which the historical measurements are modeled to describe the underlying characteristics of the observable and extrapolated into the future. For a few decades, the domain of time series analysis has been dominated by statistical methodology. Forecasting can be done using various approaches. The forecasting strategies we consider are as follows:

- Four statistical methods (ARIMA, ETS, VAR, and Theta).
- Five machine learning methods (KNN, SVR, Lasso, ElasticNet, and Linear Regression)
- Various deep learning approaches (MLP, CNN, ConvLSTM).
- Facebook prophet.

We conduct the forecasting test on three horizons (1, 2, 3, 7 days). And we are going to apply STL decomposition for each method so that we can make a good comparison and see how decomposition affects the forecasting accuracy in each approach.

We are going to use two error metrics, MAPE and NRMSE for each horizon for each algorithm with and without STL decomposition.

We can summarize what we are going to use as following:

3.4.1 Statistical methods:

Use statistical modeling to forecast the future values. It has many approaches also:

3.4.1.1 Common Approaches:

Box–Jenkins Approaches: One of the most important and generally used models is the AutoRegressive Integrated Moving Average (ARIMA), which can be quickly built thanks to the Box–Jenkins methodology. The ARIMA family has excellent flexibility for presenting varying time series. Nevertheless, it has limits due to its assumption of linearity of the time series. Another dominating and widely used statistical method is the Exponential Smoothing (ETS) method, which was proposed in the 1950s [4–6]. It is often considered as an alternative to the ARIMA models. While the ARIMA family develops a model where the prediction is a weighted linear sum of recent past observations or lags, forecasts produced by ETS are weighted averages of past observations, with the weights decaying exponentially as the observations get older.

These approaches are:

- ARIMA.
- SARIMA.
- Trend, Seasonal, Residual Decompositions:
    - Seasonal Extraction in ARIMA Time Series (SEATS).
    - Seasonal and Trend decomposition using Loess (STL).
    - Exponential smoothing:
        - Single Exponential Smoothing, or SES, for univariate data without trend or seasonality
        - Double Exponential Smoothing for univariate data with support for trends.
        - Triple Exponential Smoothing, or Holt-Winters Exponential Smoothing, with support for both trends and seasonality. (TES)
    - Autoregressive Models (AR).
    - Moving Average Models (MA).

3.4.2 Machine learning methods:

Leverage the machine learning algorithms to forecast the future values. We are going to use:

- KNN
- SVR.
- LR.
- ElasticNet.
- Lasso

3.4.3 Deep Learning Methods:

Leverage the machine learning algorithms to forecast the future values. We are going to use:

- MLP.
- CNN.
- LSTM.

Notes before proceeding in our project, If time-series has:

1. No trend, no seasonality and is stationary, we can use:
    - Moving Average
    - Simple exponential smoothing
2. Trend only, we can use:
    - Double exponential smoothing (TS must be stationary)
    - ARIMA
3. Trend and seasonality, we can use:
    - Triple exponential smoothing (TS must be stationary)
    - SARIMA
4. Trend, seasonality and holiday effects, we can use:
    - FbProphet
5. Effect of external variables, we can use:
    - FbProphet
    - ML models
    - DL models
        1. MLP
        2. CNN
        3. LSTM
        4. Conv-LSTM
6. But SARIMA and smoothing methods have some issues:
    - Good for one step or short term forecasts, but not for long term forecasts
    - Works for univariate data with linear relationships
    - Focus on temporal dependence only. It does not consider events like holidays, promotions

3.5 FB-Prophet model:

This model has been developed by the R&D team of Meta (Previously Facebook) to make forecasts inside Facebook especially for the demand on their services, and the main two ideas as we will discuss in this part, first, is that this model is dealing with each part of the time series separately, these parts includes Trend, Seasonality as we discussed in chapter 2 and the new added part is the holidays effect.

Second idea is centered about scaling the forecasting process and by scaling we have several aspects here, the first one is that large number of people is making forecasts, possibly without training on time series analysis as the most of them may be subject matter experts only, the second scale here is the variety of problems that can use this model to address, the third type here is that with large number of forecasts we need an automated way to evaluate the forecasts and this the last type of scales.

3.5.1 Forecasting model:

We now describe a time series forecasting model designed to handle the common features of business time series as described in the next figure. Importantly, it is also designed to have intuitive parameters that can be adjusted without knowing the details of the underlying model.

This is necessary for the analyst to effectively tune the model.

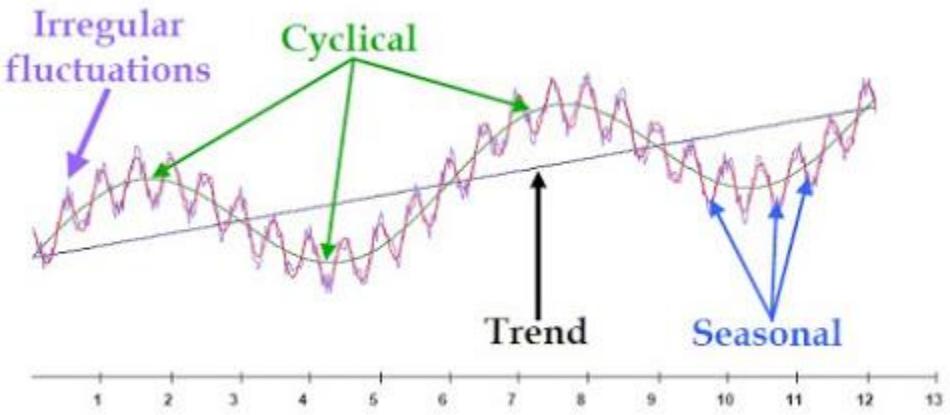


Figure 3. 52 Time series components

The main feature of Facebook prophet model is that is uses a decomposable time series model with three main components: Seasonality, Trend and holidays, fore example the additive models inside it is described as:

$$Y(t) = G(t) + S(t) + H(t) + \epsilon_t \quad (3.28)$$

And the multiplicative model is:

$$Y(t) = G(t) \times S(t) \times H(t) \times \epsilon_t \quad (3.29)$$

Where $G(t)$ is the trend model, which is long term changes and this function models the non-periodic part in the time series, $S(t)$ is the seasonality function, and it represents the periodic changes in the time series, $H(t)$ represents the effect of holidays that occur during measurement session which may occur in an irregular basis and finally the ϵ_t which represents the idiosyncratic changes which are not accommodated by the model.

3.5.2 Modeling approach:

In fact, in this model we are framing the time series problem as a curve fitting problem, which is inherently different from time series models that explicitly account for the temporal dependence structure in the data. While we give up some important inferential advantages of using a generative model such as an ARIMA, this formulation provides a number of practical advantages:

- Flexibility: We can easily accommodate seasonality with multiple periods and let the analyst make different assumptions about trends.
- Unlike with ARIMA models, the measurements do not need to be regularly spaced, and we do not need to interpolate missing values e.g. from removing outliers.
- Fitting is very fast, allowing the analyst to interactively explore many model specifications.
- The forecasting model has easily interpretable parameters that can be changed by the analyst to impose assumptions on the forecast. Moreover, analysts typically do have experience with regression and are easily able to extend the model to include new components.

3.5.3 Models of components:

As we mentioned the prophet model represents each component individually, trend and seasonality models are presented separately from seasonality and holidays models.

3.5.3.1 Trend Models:

Facebook scientists have modeled the trend with two different models according to the nature of the series, and these two models are as follows:

3.5.3.1.1 Non-linear, saturating growth:

For growth forecasting, the main consideration is how the population has grown and how it is expected to grow in the next times, and often the growth is similar to the population growth in natural ecosystem where the growth saturates at a certain carrying capacity, i.e. when the fakebook users are equal to all internet users, this sort is modeled using the logistic function as follows:

$$G(t) = \frac{C}{1 + exp\big(-k(t-m)\big)} \qquad (3.30)$$

Where, C is the carrying capacity, k is the growth rate and m is the offset parameter.

From equation 3.30 we can observe that we assume carrying capacity is constant, but in real life it's not always the case, so the first modification here is to change the carrying capacity to be a function in time $C(t)$ that its value changes with respect to time.

So the equation (3.30) becomes:

$$G(t) = \frac{C(t)}{1 + exp\big(-k(t-m)\big)} \qquad (3.31)$$

Second modification here is the growth rate, growth rate is constant in the equation, but this rate may change in some times, so the model must be able to deal with hs varying rate, to do so we make some changes here.

We incorporate trend changes in the growth model by explicitly defining changepoints where the growth rate is allowed to change. Suppose there are S changepoints at times sj,

$j = 1, \ldots, S$. We define a vector of rate adjustments $\delta \in \Re^2$, where δ_j is the change in rate that occurs at time s_j. The rate at any time t is then the base rate k, plus all of the adjustments up to that point: $k + \sum_{j:t>s_j} \delta_j$: This is represented more cleanly by defining a vector $a(t) \in \{0,1\}^S$ such that the rate at time t is then $k + a(t)^t\delta$. When the rate k is adjusted, the offset parameter m must also be adjusted to connect the endpoints of the segments. The correct adjustment at changepoint j is easily computed as:

$$\gamma_j = \left(s_j - m - \sum_{i<j} \gamma_j\right)\left(1 - \frac{k + \sum_{i<j} \delta_i}{k + \sum_{i\leq j} \delta_i}\right) \qquad (3.32)$$

The piecewise logistic growth model is then:

$$G(t) = \frac{C(t)}{1 + exp\left(-(k + a(t)^t\delta)\left(t - (m + a(t)^t\gamma)\right)\right)} \qquad (3.33)$$

An important set of parameters in our model is C(t), or the expected capacities of the system at any point in time. Analysts often have insight into market sizes and can set these accordingly. There may also be external data sources that can provide carrying capacities, such as population forecasts from the World Bank. The logistic growth model presented here is a special case of generalized logistic growth curves, which is only a single type of sigmoid curve. Extensions of this trend model to other families of curves are straightforward.

3.5.3.1.2 Linear trend with changepoints:

For forecasting problems that do not exhibit saturating growth, a piecewise constant rate of growth provides a parsimonious and often useful model. Here the trend model is:

$$G(t) = (k + a(t)^t\delta)t \; + (m + a(t)^t\gamma) \qquad (3.34)$$

whereas before k is the growth rate, δ has the rate adjustments, m is the offset parameter, and $_j$ is set to $-s_j\delta_j$ to make the function continuous.

3.5.3.1.3 Automatic Changepoint Selection:

The changepoints s_j could be specified by the analyst using known dates of product launches and other growth-altering events, or may be automatically selected given a set of candidates. Automatic selection can be done quite naturally with the formulation in the previous functions by putting a sparse prior on δ.

We often specify a large number of changepoints (e.g., one per month for a several year history) and use the prior δj ∼ Laplace(0; τ). The parameter τ directly controls the flexibility of the model in altering its rate. Importantly, a sparse prior on the adjustments δ has no impact on the primary growth rate k, so as τ goes to 0 the fit reduces to standard (not-piecewise) logistic or linear growth.

3.5.3.1.4 Trend Forecast Uncertainty:

When the model is extrapolated past the history to make a forecast, the trend will have a constant rate. We estimate the uncertainty in the forecast trend by extending the generative model forward. The generative model for the trend is that there are S changepoints over a history of T points, each of which has a rate change δj ∼ Laplace(0; τ). We simulate future rate changes that emulate those of the past by replacing τ with a variance inferred from data.

We thus measure uncertainty in the forecast trend by assuming that the future will see the same average frequency and magnitude of rate changes that were seen in the history. Once λ has been inferred from the data, we use this generative model to simulate possible future trends and use the simulated trends to compute uncertainty intervals.

3.5.3.2 Seasonality:

Business time series often have multi-period seasonality as a result of the human behaviors they represent. For instance, a 5-day work week can produce effects on a time series that repeat each week, while vacation schedules and school breaks can produce effects that repeat each year. To fit and forecast these effects we must specify seasonality models that are periodic functions of t.

The model relies on Fourier series to provide flexibility to model periodic changes, so if we let P is the regular period is the series is expected to has, so the approximation of seasonal effects will be:

$$S(t) = \sum_{n=1}^{N} \left(a_n \cos \frac{2\pi nt}{P} + b_n \sin \frac{2\pi nt}{P} \right) \quad (3.35)$$

A standard Fourier series. Fitting seasonality requires estimating the 2N parameters $= [a_1; b1; \ldots; a_N; b_N]^t$. This is done by constructing a matrix of seasonality vectors for each value, for example with yearly seasonality and N = 10:

$$X(t) = \sum_{n=1}^{10} \left(a_n \cos \frac{2\pi nt}{365.25} + b_n \sin \frac{2\pi nt}{365.25} \right) \quad (3.36)$$

So, the seasonal component then is:

$$S(t) = X(t)\beta \quad (3.37)$$

In our generative model we take β ~ Normal(0; σ2) to impose a smoothing prior on the seasonality.

3.5.3.3 Holidays and Events:

Mobile traffic as a business time series has the same properties, as the traffic is affected by the seasons of holidays, and also the extraordinary events that happens, so it's important to get into the problem from that perspective also.

Holidays and events provide large, somewhat predictable shocks to many business time series and often do not follow a periodic pattern, so their effects are not well modeled by a smooth cycle. For instance, Thanksgiving in the United States occurs on the fourth Thursday in November. The Super Bowl, one of the largest televised events in the US, occurs on a Sunday in January or February that is difficult to declare programmatically. Many countries around the world have major holidays that follow the lunar calendar. The impact of a particular holiday on the time series is often similar year after year, so it is important to incorporate it into the forecast.

Incorporating the list of holidays into the model is made straightforward by assuming that the effects of holidays are independent. For each holiday i, let Di be

the set of past and future dates for that holiday. We add an indicator function representing whether time t is during holiday i, and assign each holiday a parameter κi which is the corresponding change in the forecast. This is done in a similar way as seasonality by generating a matrix of regressors:

$$Z(t) = [1(t \in D_1), \dots \dots, 1(t \in D_t)] \qquad (3.38)$$

And taking:

$$h(t) = Z(t)k \qquad (3.39)$$

As with seasonality, we assume $k\ Normal(0, \sigma^2)$.

3.5.4 Model Fitting:

We have the following setting:

```
def init_prophet():
    """
    A  function to initialize the prophet model with best parameters
    and added some custom seasonalities
    """
    #initiating the model
    model = pt.Prophet(changepoint_range=0.9, changepoint_prior_scale=0.005,
                       n_changepoints = 50, weekly_seasonality= 100,
                       daily_seasonality= 100,
                       seasonality_mode = "additive",
                       holidays  = milan_holidays)
    # Adding country holidays
    model.add_country_holidays(country_name= "IT")
    # Custom regressors for times in day according to active time
    model.add_regressor("hour")
    model.add_regressor("day")
    model.add_regressor("dayofweek")
    return model
```

We can see that we use the prophet model with the following parameter:

- Changepoint_range : this parameter controls the spread of change point on the past data, so by assigning the value of 0.9 we make the changepoints spread over 90% percent of the past data.
- Changepoint_prior_scale: This parameter controls the effect of the changepoint if it happens, so by assigning 0.005 we found this value is the optimum for our use case.
- n_changepoints : This parameter controls the number of change points, we found that 50 change points is the training data will be very sufficient.
- Weekly_seasonality: this controls the "N" parameter in equation (3.8) for weekly seasonality, and we found 100 is the best.
- Daily_seasonality: this controls the "N" parameter in equation (3.8) for daily seasonality, and we found 100 is the best.
- Seasonality_mode: here we found that our data may exhibit some variable trends on daily and weekly data seasons, so we used the additive model as we discussed in chapter 2.
- Holidays: this parameter used to enter the holidays of a specific city, so we entered the dates of Milan Holidays.

We have noticed that this model behaves the best among other models after tuning hyperparameters, and also among other algorithms this algorithm is the super star here with accuracies more than 90% on the week forecasting.

3.5.5 Results:

And this is a sample from FB-prophet forecasts:

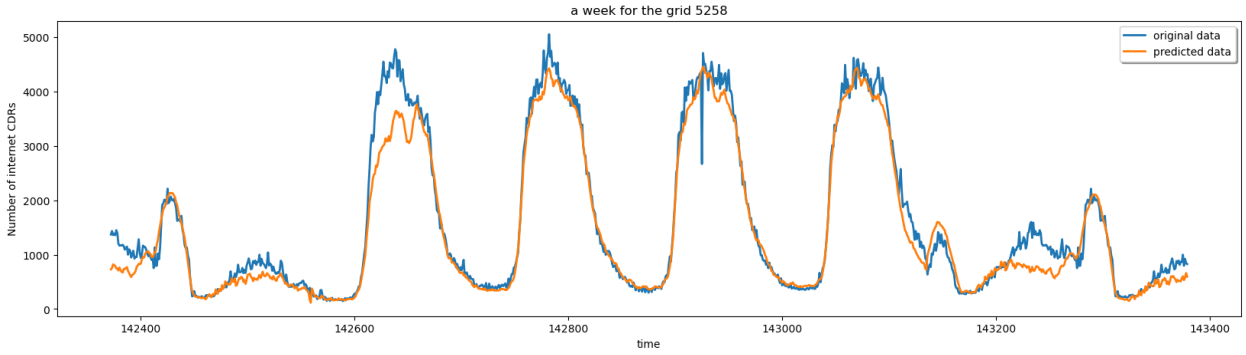


Figure 3. 53 week forecasts for grid 5258

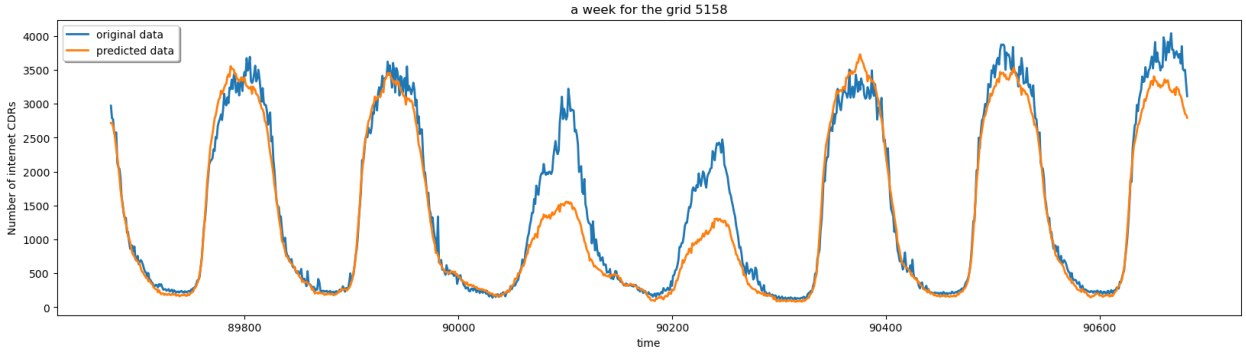


Figure 3. 54 week forecasts for grid 5158

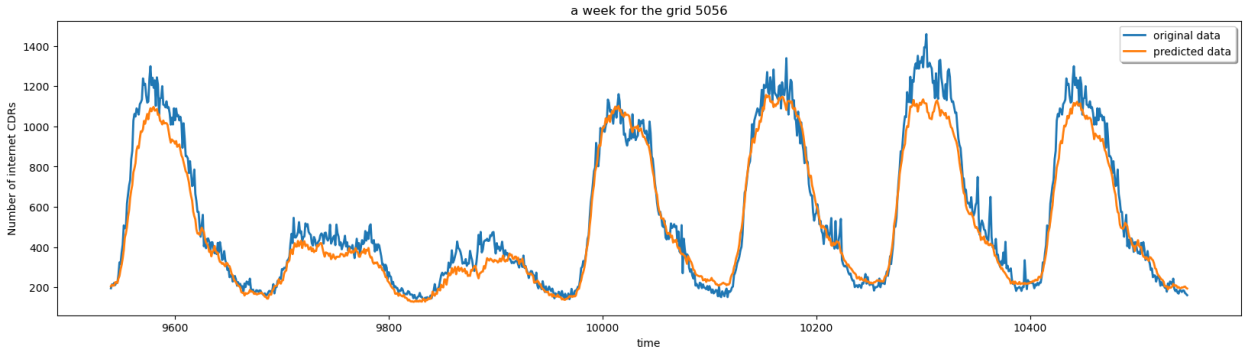


Figure 3. 55 Week Forecasts for grid 5158

This image represents the forecasting accuracy for the prophet model, these are randomly chosen weeks for randomly chosen grids, and as we can see that this results in a near-real forecasts, and the main error sources are weekends where the model miss forecast the series as the traffic is different from the ordinary, but in most cases the model is identical to the ground truth data.

And if we want to show up the whole forecasts, we have the following images:

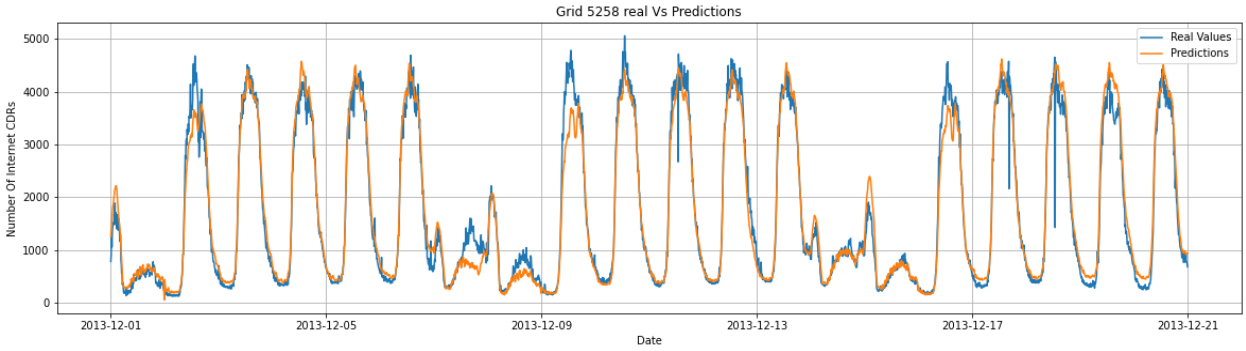


Figure 3. 56 All 3 weeks forecasts for Grod 5258

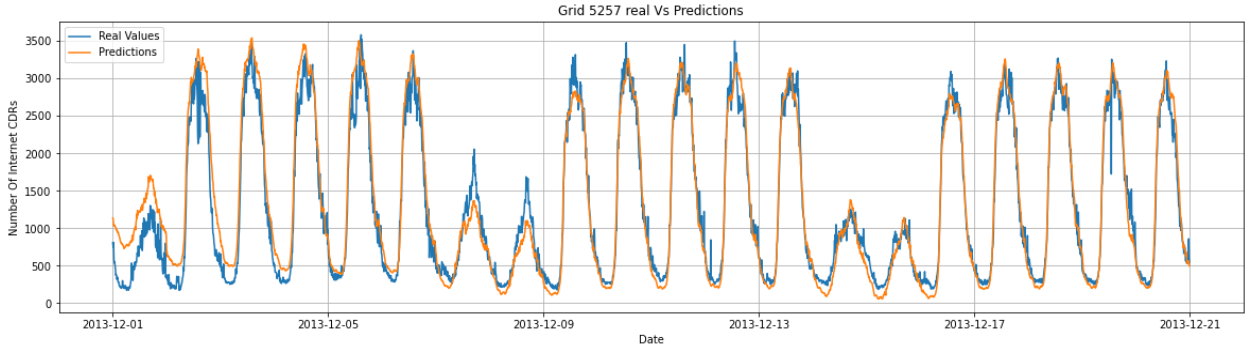


Figure 3. 57 Grid 5275 Real Vs Forecasts

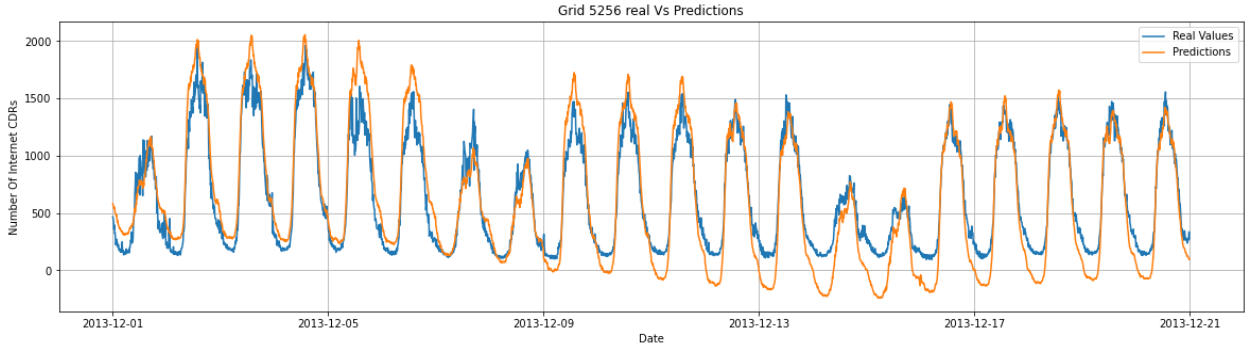


Figure 3. 58 Grid 5256 Real Vs Forecasts

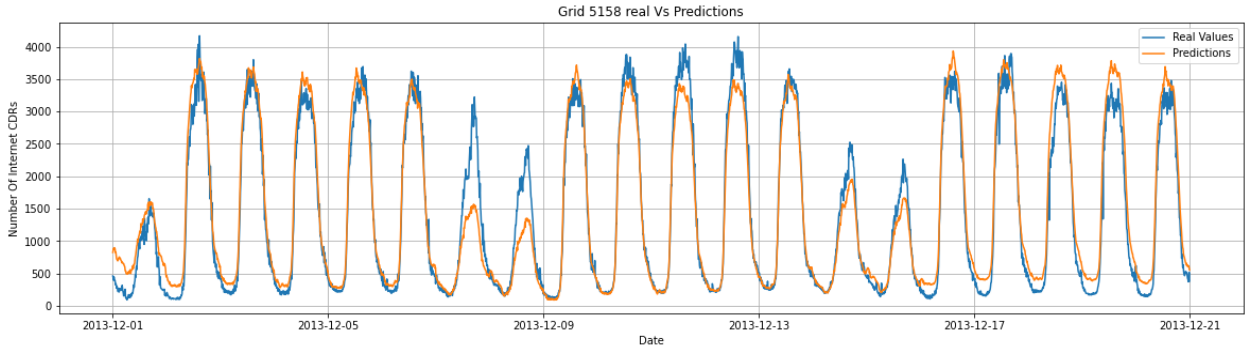


Figure 3. 59 Grid 5158 Real Vs Forecasts

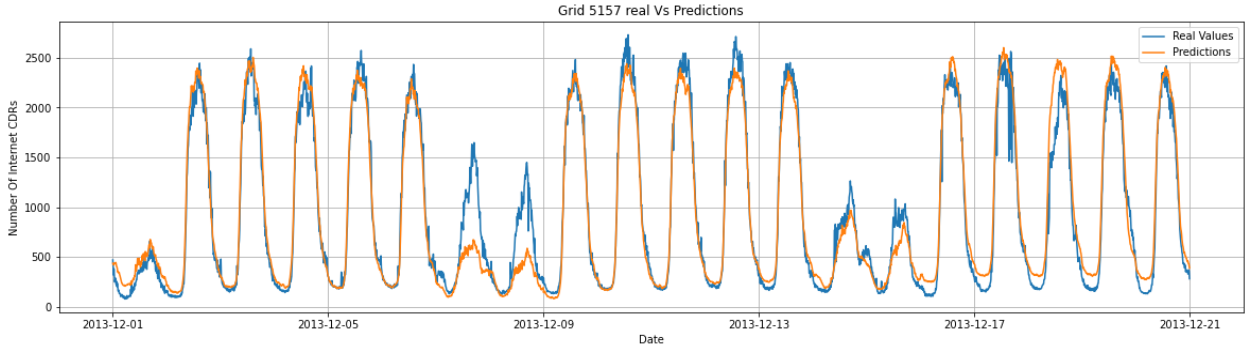


Figure 3. 60 Grid 5157 Real Vs Forecasts

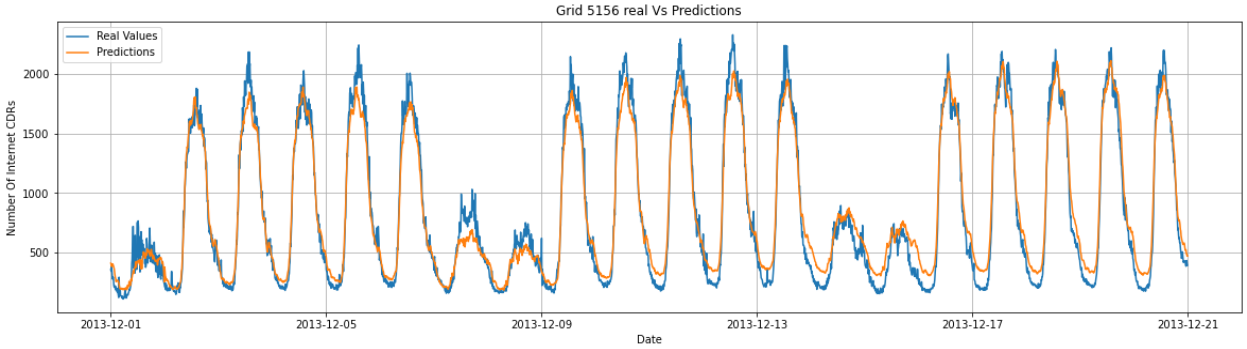


Figure 3. 61 Grid 5156 Real Vs Forecasts

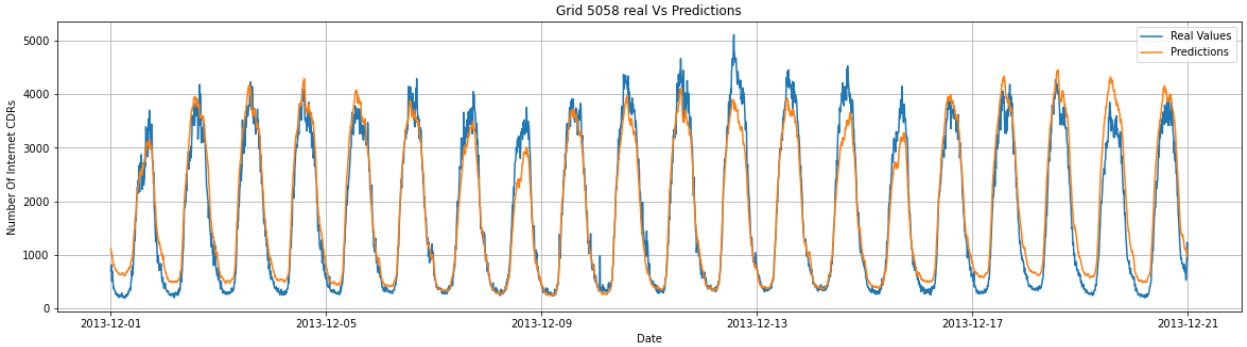


Figure 3. 62 Grid 5058 Real Vs Forecasts

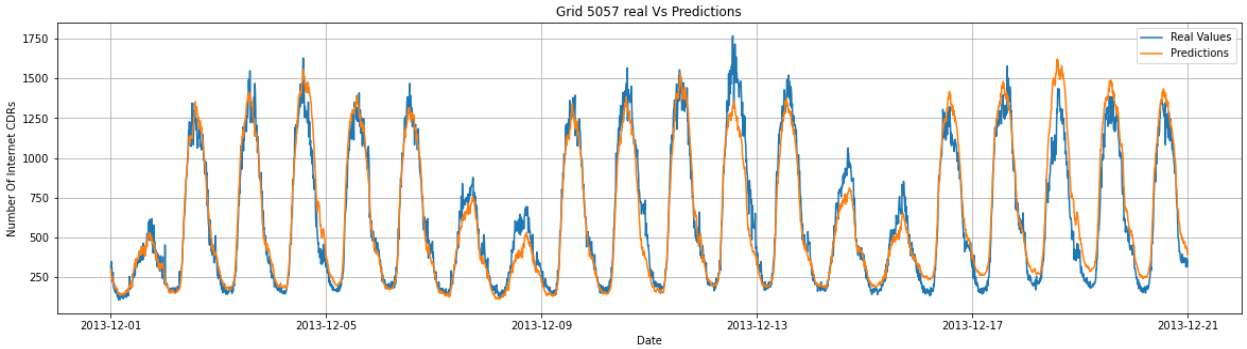


Figure 3. 63 Grid 5057 Real Vs Forecasts

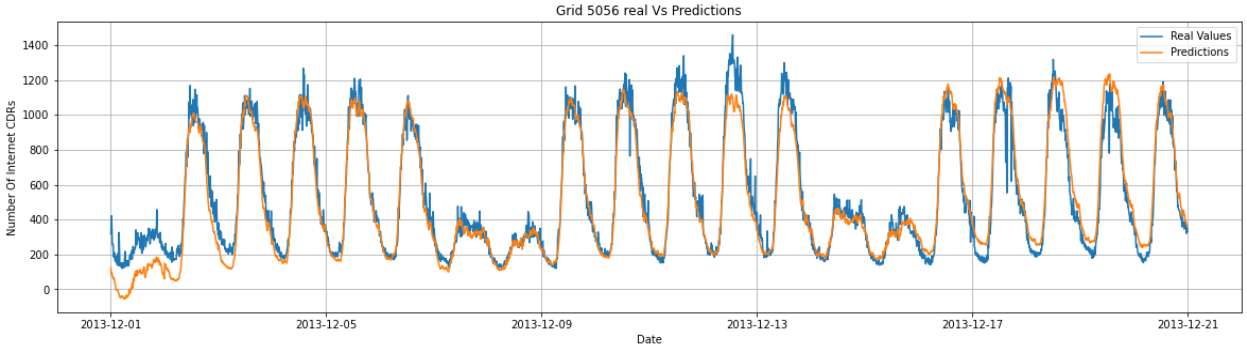


Figure 3. 64 Grid 5056 Real Vs Forecasts

So these are the forecasts for the 3 weeks, but we did forecasts for 1, 2, 3 days and a week and the Normalized root mean square error (NRMSE) and the Mean Absolute Percentage Error (MAPE) are as follows:

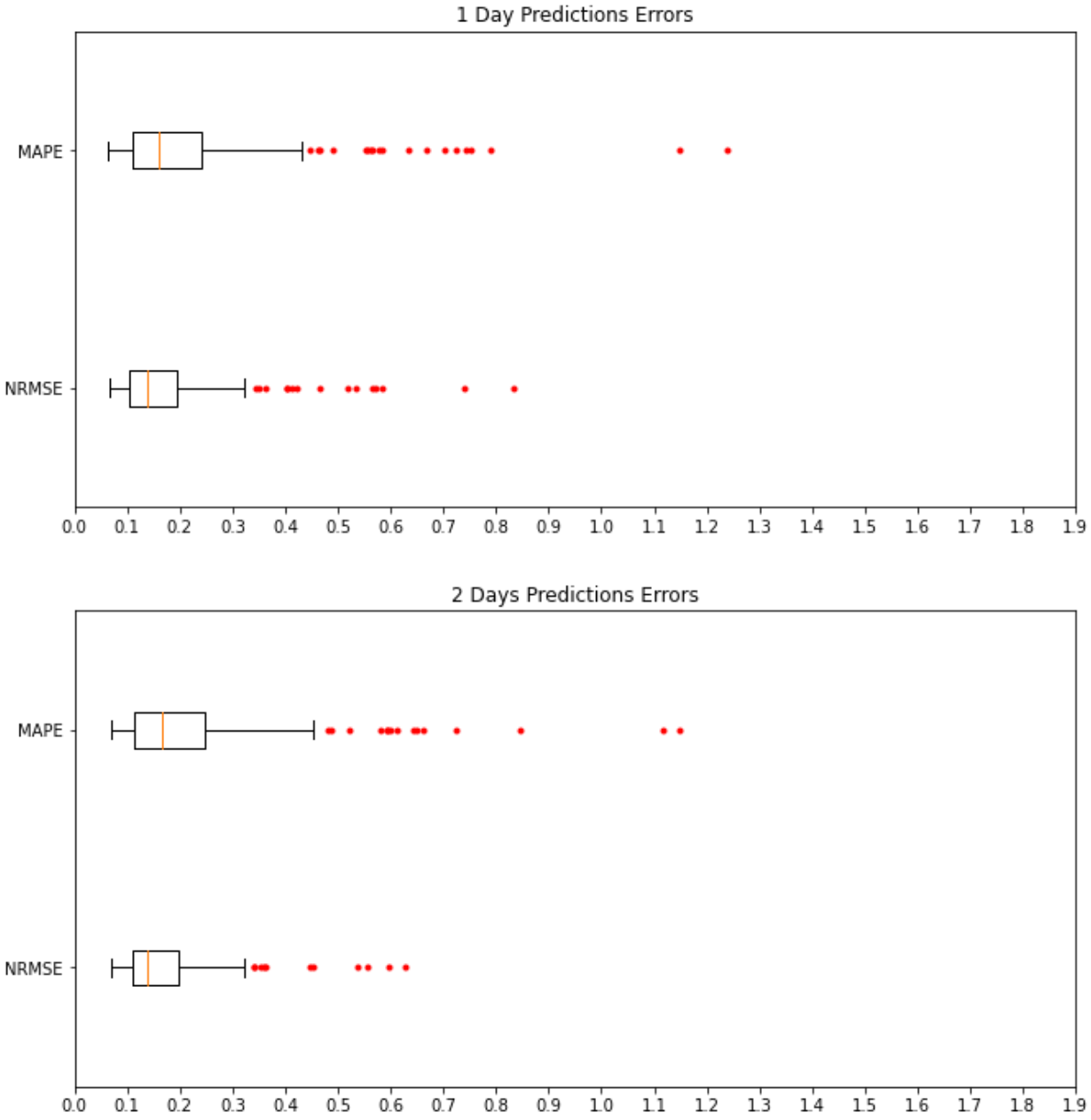


Figure 3. 65 NRMSE and MAPE box plot for 1 and 2 days

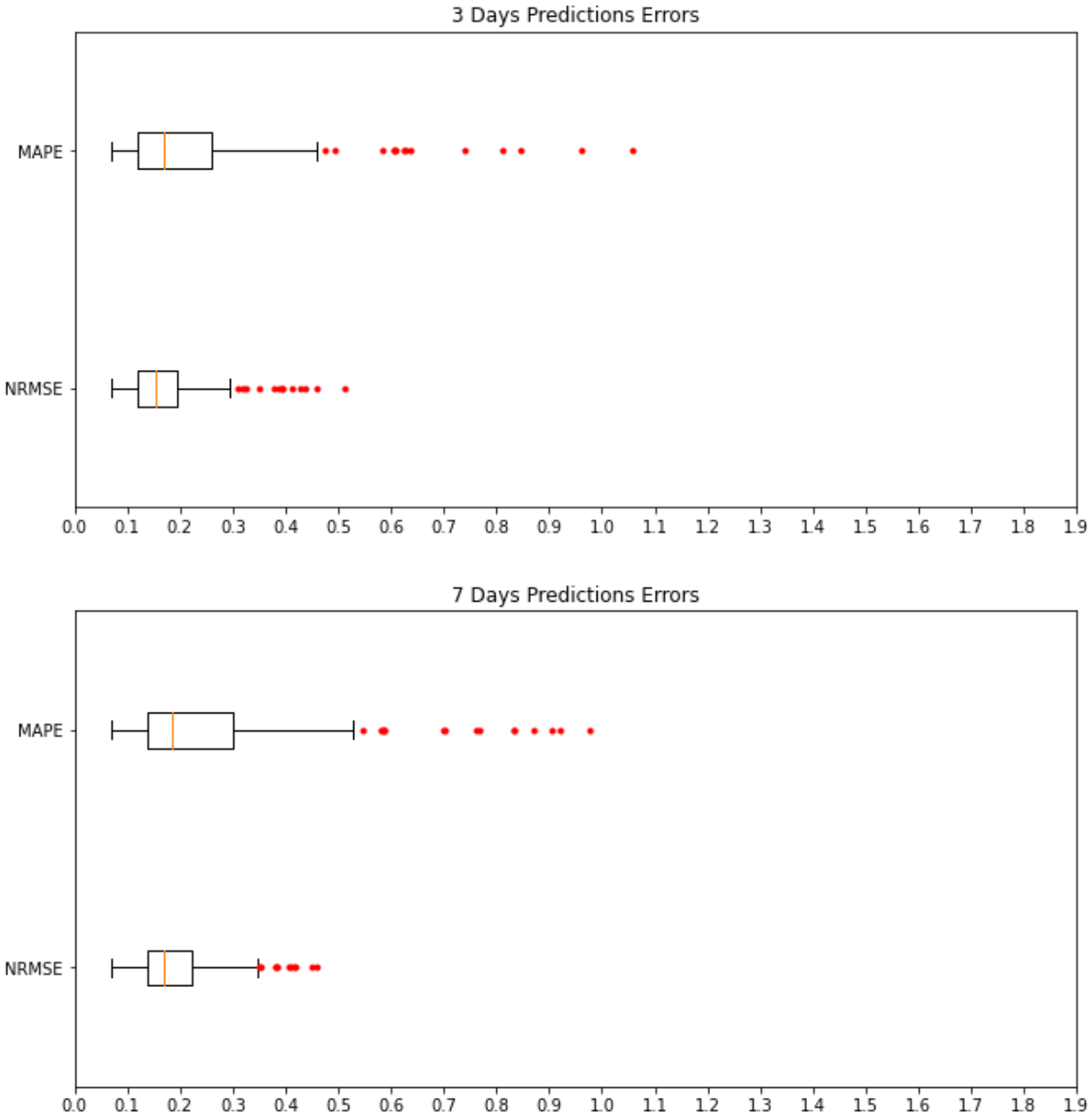


Figure 3. 66 NRMSE and MAPE box plot for 3 and 7 days

So for week forecasts Errors are ranging from (7% to 35%) for NRMSE and form (7% to 50%) for MAPE, and these large errors are most concentrated on weekends as we saw in the last figures.

3.6 Forecasting Results:

We will use the sliding window technique with shifting window with window size of 30 days (4320 steps) and forecasting for 1,2,3,7 Days (144, 288, 432, 1008 steps), and measuring the accuracy of our models by two metrics: Normalized root mean square error (NRMSE) and mean absolute percentage error (MAPE).

The results for the Statistical methods:

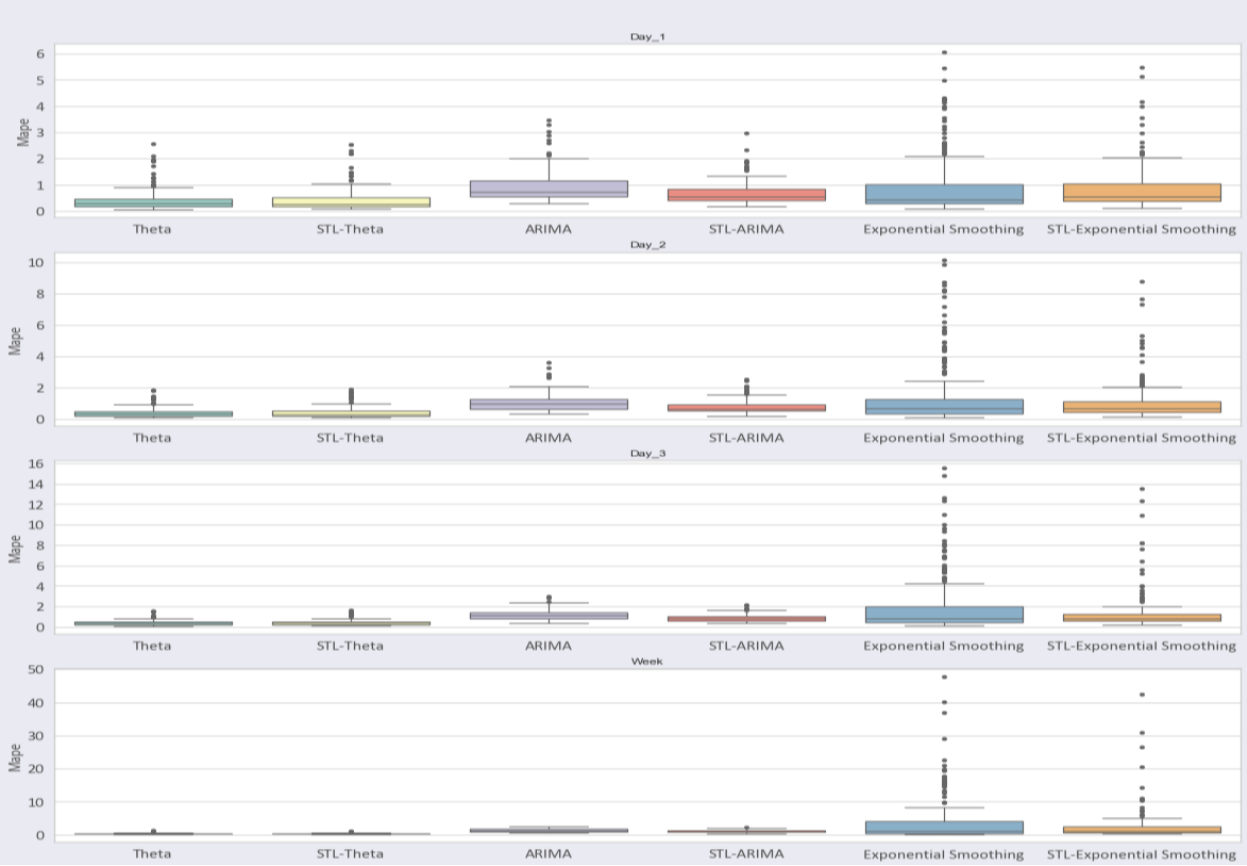


Figure 3. 67 Forecasting results for the Statistical methods

From here, we can observe that the best Statistical model is Theta model.

Results for the machine learning methods:

Figure 3. 68 Forecasting results for the machine learning methods

From these results, the best model was KNN.

Results For Deep learning Models:

Figure 3. 69 Forecasting Results for Deep learning Models

From these results we can observe that is the MLP

Final Comparison:

Table 6 Model forecast accuracies

	1-Day		2-Day		3-Day		Week	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
Knn	0.3541 ± 0.391	0.4739 ± 0.531	0.3184 ± 0.336	0.4129 ± 0.412	0.3035 ± 0.329	0.3981 ± 0.411	0.3373 ± 0.357	0.4391 ± 0.438
Prophet	0.22 ± 0	0.17 ± 0.12	0.22 ± 0.17	0.17 ± 0.1	0.23 ± 0.17	0.17 ± 0.08	0.26 ± 0.18	0.19 ± 0.08
Theta	0.4024 ± 0.387	0.5464 ± 0.457	0.398 ± 0.3	0.4898 ± 0.261	0.3999 ± 0.245	0.4819 ± 0.2	0.3919 ± 0.177	0.4649 ± 0.15
Stl-Knn	0.4097 ± 0.48	0.5291 ± 0.588	0.3801 ± 0.446	0.4784 ± 0.501	0.3631 ± 0.443	0.4618 ± 0.503	0.4192 ± 0.482	0.53 ± 0.542
Stl-Theta	0.4267 ± 0.443	0.5054 ± 0.541	0.4121 ± 0.333	0.4283 ± 0.302	0.41 ± 0.264	0.4107 ± 0.205	0.4248 ± 0.176	0.4099 ± 0.098
Stl-Lasso	0.4182 ± 0.374	0.5135 ± 0.417	0.3926 ± 0.344	0.47 ± 0.333	0.3819 ± 0.342	0.4624 ± 0.337	0.4296 ± 0.366	0.5112 ± 0.356
Stl-Elasticnet	0.4182 ± 0.374	0.5135 ± 0.417	0.3926 ± 0.344	0.47 ± 0.333	0.3819 ± 0.342	0.4624 ± 0.337	0.4296 ± 0.366	0.5112 ± 0.356
Stl-Linearregression	0.4182 ± 0.374	0.5135 ± 0.417	0.3926 ± 0.344	0.47 ± 0.333	0.3819 ± 0.342	0.4624 ± 0.337	0.4296 ± 0.366	0.5112 ± 0.356
Stl-Svr	0.4395 ± 0.441	0.5164 ± 0.486	0.4159 ± 0.407	0.4728 ± 0.396	0.403 ± 0.402	0.4617 ± 0.397	0.453 ± 0.431	0.519 ± 0.419
Svr	0.5362 ± 0.219	0.6775 ± 0.213	0.5331 ± 0.222	0.681 ± 0.214	0.5363 ± 0.221	0.6865 ± 0.213	0.5399 ± 0.22	0.6907 ± 0.216
Linearregression	0.5464 ± 0.425	0.5128 ± 0.408	0.5208 ± 0.402	0.4676 ± 0.315	0.5112 ± 0.405	0.4619 ± 0.319	0.5624 ± 0.43	0.503 ± 0.334
Lasso	0.5468 ± 0.426	0.5128 ± 0.408	0.5211 ± 0.402	0.4676 ± 0.315	0.5116 ± 0.405	0.4619 ± 0.319	0.563 ± 0.43	0.5031 ± 0.334
Elasticnet	0.5468 ± 0.426	0.5128 ± 0.408	0.5211 ± 0.402	0.4676 ± 0.315	0.5116 ± 0.405	0.4619 ± 0.319	0.563 ± 0.43	0.5031 ± 0.334
Stl-Arima	0.6899 ± 0.42	0.5691 ± 0.392	0.7874 ± 0.412	0.5718 ± 0.287	0.844 ± 0.361	0.5829 ± 0.192	1.0285 ± 0.38	0.6355 ± 0.134
Stl-Var	0.8095 ± 0.268	0.7496 ± 0.153	1.0258 ± 0.351	0.7793 ± 0.101	1.0405 ± 0.319	0.7965 ± 0.092	1.3004 ± 0.369	0.7902 ± 0.078
Var	0.7167 ± 0.223	0.7647 ± 0.164	1.0265 ± 0.351	0.7797 ± 0.101	1.0405 ± 0.319	0.7965 ± 0.092	1.3004 ± 0.369	0.7902 ± 0.078
Arima	0.9439 ± 0.587	0.6416 ± 0.366	1.0734 ± 0.549	0.636 ± 0.208	1.1513 ± 0.496	0.6562 ± 0.162	1.3163 ± 0.44	0.7177 ± 0.12
Stl-Exponential Smoothing	0.8588 ± 0.841	0.8464 ± 0.826	1.0956 ± 1.276	0.8705 ± 0.803	1.4116 ± 1.978	0.9872 ± 1.114	2.8172 ± 5.486	1.5936 ± 2.514
Exponential Smoothing	0.9425 ± 1.115	0.9896 ± 1.075	1.5305 ± 2.083	1.1958 ± 1.358	2.0712 ± 2.936	1.4326 ± 1.681	4.8174 ± 8.092	2.6994 ± 3.916

From All of these Model we can observe that Facebook prophet model is the best one to deal with mobile network traffic, and also it has the ability to generalize well with long forecasts.

3.6 Main Contribution:

- Full Pipeline for the time series forecasting, starting from the data understanding to processing to forecasting using different models.
- High Accurate Missing data imputation and outliers detection methods
- Long Forecast: 7 Days, compared to past literature which was 10 hours only.
- Accurate long forecasts: We have reached an accuracy of 91% in the 7 days forecasts, which outperforms the past literature.

3.7 Conclusion and final Remarks:

We have used a dataset of Milan and Terenteno which was taken to represent the interaction between the users and the mobile network, this dataset is representing the interaction in the form of CDRs, CDRs has some forms like SMS, call and internet CDRs, this is a spatio-temporal data that represents the spatial and temporal data of the traffic in certain area for a given period of time, which starts from the 1st of November 2014 and ends in 1st of January 2015.

And by using most accurate methods for preprocessing parts especially in outliers detection and missing data imputations, we have got accurate long term forecasts which have reached 7 consecutive days of forecast with an accuracy reaches 91% which is sufficient for network to take proactive actions and to use in network planning.

Chapter Four: Future Work

4| Future work

The other main thing after knowing the traffic and making an accurate forecast, we need to know the nature of the traffic, and by nature here we mean the environment from which the traffic was initiated.

This will help the operators to determine the best way to plan any area, and decide the best spectrum can be allocated for this area according to the dominant environment, and also to decide whether this site need more enhancements or not according to it's KPIs.

And we have to say that this stage is under development and we have a notable progress on it and is expected to be finished soon.

And by relating the KPLs with environment classification we can address a variety of problems which can be more forward step towards a real tool that help telecom operators to make an efficient spectrum planning.

References:

[1] J. Huang, J. Tan, Y. Liang, Wireless big data: transforming heterogeneous networks to smart networks. J. Commun. Inf. Netw. 2(1), 19–32 (2017)

[2] X. Wu, X. Zhu, G. Wu, Data mining with big data. IEEE Trans. Knowl. Data Eng. 26(1), 97–107 (2014)

[3] Y. Lecun, Y. Bengio, G. Hinton, Deep learning. Nature 521(7553), 436 444 (2015)

[4] M. Jordan, T. Mitchell, and Machine learning: trends, perspectives, and prospects. Science 349(6245), 255–260 (2015)

[5] Ericsson Mobility Report and Data Forecasts, Mobile data traffic outlook (2021)

[6] Qingtian Zeng, Qiang Sun, Geng Chen 1, Hua Duan, Attention based multi-component spatiotemporal cross-domain neural network model for wireless cellular network traffic prediction, EURASIP Journal on Advances in Signal Processing 2021:46 (2021)

[7] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, Franz J. Király, sktime: A Unified Interface for Machine Learning with Time Series.

[8] Dr. N.D Lewis, DEEP TIME SERIES FORECASTING With PYTHON: An Intuitive Introduction to Deep Learning for Applied Time Series Modeling.

[9] Jason Brownlee, Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future

[10] Jinsung Yoon, James Jordon, Mihaela van der Schaar, GAIN: Missing Data Imputation using Generative Adversarial Nets.

[11] Ehsan Adeli, Jize Zhang, Alexandros , A. Taflanidis, Convolutional generative adversarial imputation networks for spatio-temporal missing data in storm surge simulations.

[12] Irfan Pratama, Adhistya Erna Permanasari, Igi Ardiyanto, Rini Indrayani, A Review of Missing Values Handling Methods on Time-Series Data.

[13] Barlacchi, G., De Nadai, M., Larcher, R. et al. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. Sci Data 2, 150055 (2015). https://doi.org/10.1038/sdata.2015.55.

[14] R. J. A. Little and D. B. Rubin, Statistical Analysis with Missing Data, 2nd Ed., Hoboken, NJ: Wiley, 2002.15. H. Demirhan, Z. Renwick, Missing value imputation for short to mid-term horizontal solar irradiance data. Appl Energy, vol. 225, pp. 998–1012, (2018). 10.1016/j.apenergy.2018.05.054.

[15] Shin-Fu Wu, Chia-Yung Chang, and Shie-Jue Lee, Time Series Forecasting with Missing Values

[16] Jinsung Yoon, James Jordon, Mihaela van der Schaar, GAIN: Missing Data Imputation using Generative Adversarial Nets.

[17] Ehsan Adeli, Jize Zhang, Alexandros , A. Taflanidis, Convolutional generative adversarial imputation networks for spatio-temporal missing data in storm surge simulations.

[18] Irfan Pratama1, Adhistya Erna Permanasari, Igi Ardiyanto, Rini Indrayani, A Review of Missing Values Handling Methods on Time-Series Data.

[19] Stef van Buuren TNO Karin Groothuis-Oudshoorn, MICE: Multivariate Imputation by Chained Equations.

[20] Time-Series Anomaly Detection Service at Microsoft Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou∗ Tony Xing, Mao Yang, Jie Tong, Qi Zhang Microsoft Beijing, China.

[21] Iglewicz B, Hoaglin DC (1993) How to detect and handle outliers. ASQC Quality Press.

[22] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407 (2019).

[23] https://blogs.sap.com/2020/12/21/anomaly-detection-in-time-series-using-seasonal-decomposition-in-python-machine-learning-client-for-sap-hana/

[24] https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-1-4ece5098b755

[25] Chaoyun Zhang, Paul Patras, Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks.

[26] Sean J. Taylor , Benjamin Letham, Forecasting at Scale.

[27] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, Qi Zhang, Time-Series Anomaly Detection Service at Microsoft.

[28] Time-Series Anomaly Detection Service at Microsoft Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou∗ Tony Xing, Mao Yang, Jie Tong, Qi Zhang Microsoft Beijing, China.

[29] Iglewicz B, Hoaglin DC (1993) How to detect and handle outliers. ASQC Quality Press.

[30] https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-1-4ece5098b755
[31] https://blogs.sap.com/2020/12/21/anomaly-detection-in-time-series-using-seasonal-decomposition-in-python-machine-learning-client-for-sap-hana/
[32] https://neptune.ai/blog/anomaly-detection-in-time-series
[33] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407 (2019).
[34] https://avinetworks.com/glossary/anomaly-detection/
[35] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/
[36] https://lilianweng.github.io/posts/2018-08-12-vae/

[37] https://mirror-medium.com/?m=https%3A%2F%2Ftowardsdatascience.com%2Fhow-to-perform-anomaly-detection-with-the-isolation-forest-algorithm-e8c8372520bc
[38] https://www.section.io/engineering-education/anomaly-detection-model-on-time-series-data-using-isolation-forest/

[39] Geoffrey E. Hinton, and Ruslan R. Salakhutdinov. “Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.
[40] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/