RESTAURANT ITEMS PREDICTION SYSTEM
4th year ACADEMIC PROJECT

PRODUCT NAME: ChefTech

THEME: Data Science

Project Period: Second Semester 2019

Date of Completion: May 23, 2019

Project Group: GoData

Supervisor(s): Mr. Sifi Sami

Group Members:

- Allaoui Hayfa
- Ben Othmen Ons
- Chandoul Wejdene
- Ghdiri Oussama
- Masmoudi Skander
- Sabbagh Wiem

Year of 2018-2019

ABSTRACT

Each food item is linked to respective resources, country for example, and as each product is made for a specific population following many factors that are dependent or not by time. These changes in inventory are kept track of through predicting these items on a daily basis which in result will allow us to predict the revenue

Business owners nowadays opt for data science to help them with their common problem which is PROFIT, so based on historical data of business, predictive models of machine learning can estimate future offers, needs and wins.

The project also proposes to keep track of each and every menu item. At a specific time period (typically the end of the week). The project also makes smart predictions on required inventory for the following week based upon the predicted climate and possible occasions or events that may influence near future sales. At the end of the week, the software takes into account all threshold levels, predictions, and other factors to generate an order form, which after being verified by the manager is sent out to the vendors.

ACKNOWLEDGEMENT

We take this opportunity to thank all those people who were involved in one way or the other in making the process successful.

We express our deep gratitude to Mitakus Analytics, for the time invested and letting me use their data as a case study for our project. We would also like to thank Mr.Sifi Sami, Mr Selmi Heny, and Miss Trabelsi Dorra, for the immense advice and guidelines passed on to us throughout the entire period.

Table of Contents

CHAPTER ONE : PROJECT CONTEXT

1.1 Background

Our proposed project is a real time prediction of menu items and revenue for a restaurant business.

A menu items and revenue prediction system is a system that enables people and companies to know how much stock they must have at a given time and how to keep track of it.

Most restaurant owners are faced with this problem and have to spend extra capital so as to be able to keep track of their stock.

Forecasting item quantities can help restaurants and canteen owners make the best operations to maximize the revenue.

With an accurate item forecasting model, owners can prepare suitable amount of ingredients that exactly satisfy future visitors.

Based on this project, not only business managers will profit, also the environment.

While food loss happens mainly at the production stage due to insufficient skills, natural calamities, lack of proper infrastructure and poor practices, food waste occurs when edible food is intentionally discarded by restaurants and others after they fail to plan their meals properly and store food till it spoils or goes past the expiry date. At times, food waste can also happen due to oversupply in markets.

We propose a solution to those issues by developing a system that keeps track of inventory in restaurants and updates it according to daily sales. Each food item is linked to its respective supplies and as each product is sold the ingredients utilized in making that product are also utilized. These changes in inventory are kept track of through utilizing a database.

Restaurants also tend to reject a lot of food because it doesn’t conform to their quality and aesthetic standards. For example, in fancy restaurants, poor quality products should not be used otherwise it will disinterest customers of a certain wealth.

Food wastage is a growing problem in Germany. Approximately eleven million tons of food is thrown away per year. Per capita, that means that each German wastes around 55 kilograms of food every year which is a huge number that should be taken care of considering the fact that Germany holds 82.85 million inhabitant.

We propose to create a system that will help restaurants to keep track of each and every product supply. At a specific time period (typically the end of the week); if the inventory is below the threshold level, order forms to the specific suppliers are generated in order to restock the required items for the next week.

The project also makes smart analysis on the type of food most requested and the time or period it is most requested. This helps managers to know their most popular foods and when their demand is high.

Creating this project will not be sufficient if we do not focus on the consumer that will be our target. The German population is our target in our case.

We should know everything about this population. How often they eat outside, their diet, their preferences, their lifestyle, their holidays, their religion, their region and surely, we will consider seasonality and other public events.

Such project will make a healthy balanced environment for our clients and the consumers.

The data that will be used for training our methods is provided by Mitakus.

The data provided include variables such as revenue, item description, amount, price…. all of which are known to have an impact on the sales of a restaurant or a canteen.

1.2 Statement of problem

In Germany a percentage of 65 to 70 of people recognize themselves as Christians, 29% of which as Catholics. There is also a Muslim minority of 4.4%. A number as high as 36% do not identify themselves as having any religion or belong to another than Christianity or Muslim.

Germans drink a lot of beer, eat sausages and make bread of different taste.

Wurst, which means sausages in German, are an important part of the German cuisine, which are made of pork, beef or veal and flavored differently.

The German cuisine is also rich with different types of delicious traditional foods, as well as vegetarian and vegan. In Germany you will always have a lot of choices on what to eat and drink.

1.2.1 Advantages & Disadvantages of an Inventory Management System:

As a business owner, you can choose to rely on traditional hand counting methods or institute some form of computerized inventory control. Both approaches have their advantages and disadvantages, and it is important to weigh the relative advantages carefully.

1.2.2 Speed and Efficiency

A prediction management system makes everything from inputting information to taking revenue much easier. Doing a hand count of revenue can take long, but with a prediction management system, the same process can be done in a matter of hours.

1.2.3 Timely Data

When dealing with a manual system, the data is only as accurate and up to date as the last hand count. With prediction management system, the management team can pull a forecast and instantly see how many units are on the floor, how many have sold and which products are selling the fastest and the ones that will be sold in future.

1.3 Business Objectives

To minimize capital investment in inventory by eliminating excessive stocks. Restaurant owners will be able to know the type of food their customers enjoy and the type that is not popular to the customers. This will enable them to make the necessary changes to avoid incurring losses.

Maximize profit. Restaurant owners will be able to know the most popular foods by their customers and what time the customers prefer the food. This will assist them to make a menu that will attract more customers hence increasing their profits.

So, our business objectives are mainly focused on optimizing the supply chain, and if we succeed in doing that, we can guarantee that, in terms of financial performance, our client will notice a fast and positive improvement in revenue growth and net income.

We basically want to synchronize supply with demand:

- Optimizing food stock: The right number of good products in the right place
- Reduce risk of stock shortage: Ensure availability of all necessary ingredients considering season constraints
- Greater understanding of consumer behavior: Their most/least favorite dishes (most/least purchased)
    - Considering holidays in Germany
    - According to population per region (Average age, Social status, Revenues, Socio-Professional Category …)
    - According to typical dishes per region
    - Considering seasonal dishes
    - Considering clients' constraints (Allergies, Vegetarians, …)

1.4 Research questions

How will the system solve the problems faced by restaurants in stock management?

Will the system work effectively to solve the stated problems?

How will the system increase profits to the restaurant?

1.5 Justification

The system will help improve profits and efficiency for restaurants. Restaurants owners and managers will be able to easily regulate their stock without any delays in supply. This will ensure that the restaurant always has ready stock because of pre-predicted items and quantities.

The system will help restaurant to save time. The restaurants will be able to save a lot of time that calculating profits, supplies utilized and amount needed to purchase new supplies.

This time will be spent doing other things that are beneficial to the restaurants.

The system will help restaurants to reduce unnecessary expenses. Capital spent on people and resources used to deal with stock calculations and tracking will no longer be necessary. The system will be able to do this on its own hence enabling restaurants to save money.

It avoids shortage of supplies and duplicate ordering.

Our project will be considered a success if:

- A good estimation of the items' quantities and costs are provided.
- A fancy menu for the client according to its preferences and constraints is offered
- An accurate estimation of the real time revenue is provided.
- Optimize food stock with a better accuracy.
- Detect risks of food shortage and food waste.

1.6 Data Science Goals

- Predict choice of menu items, their cycles, their costs and prices according to multiple constraints.
- Identify food trends in real time
- Predict the revenue

1.7 Conclusion

The aim of this chapter was to have an inbound understanding of what a prediction system is and its impact on restaurants on a technological scale.

In a specific point of view our system will have its impacts on the administration, the organization, the clients or customers. Most of its impacts as we so in the article are mostly positive as it has helped many organizations, restaurants included, to maximize their profits, avoid losses due to factors such as late delivery of stock, expiry of stock, overstocking, understocking, customer dissatisfaction due to lack of commodities etc.

Were just but a few points raised in the area. We were able to realize that our system will tremendously improve restaurants leading to higher profits and better stock arrangement.

With all the content covered in this article we can come to a conclusion that our prediction system has made businesses have an easier time when it comes to their stock. It can also be seen that this has brought nothing but positive factors to the industry.

CHAPTER TWO: DATA UNDERSTANDING

2.1 Introduction

It is important to understand that data exploration and discovery is both an art and a science. There is the science of digging into data and coaxing answers from it, and there is the art of knowing where to look and working with colleagues.

2.2 Data Exploration

In this project, we have 8 initial datasets one for each year from 2011 until 2018 for two canteens in Lehel and Giesing.

We have 24 attributes and more than 23 Millions records of the activities of those canteens

- ❖ Variable description:

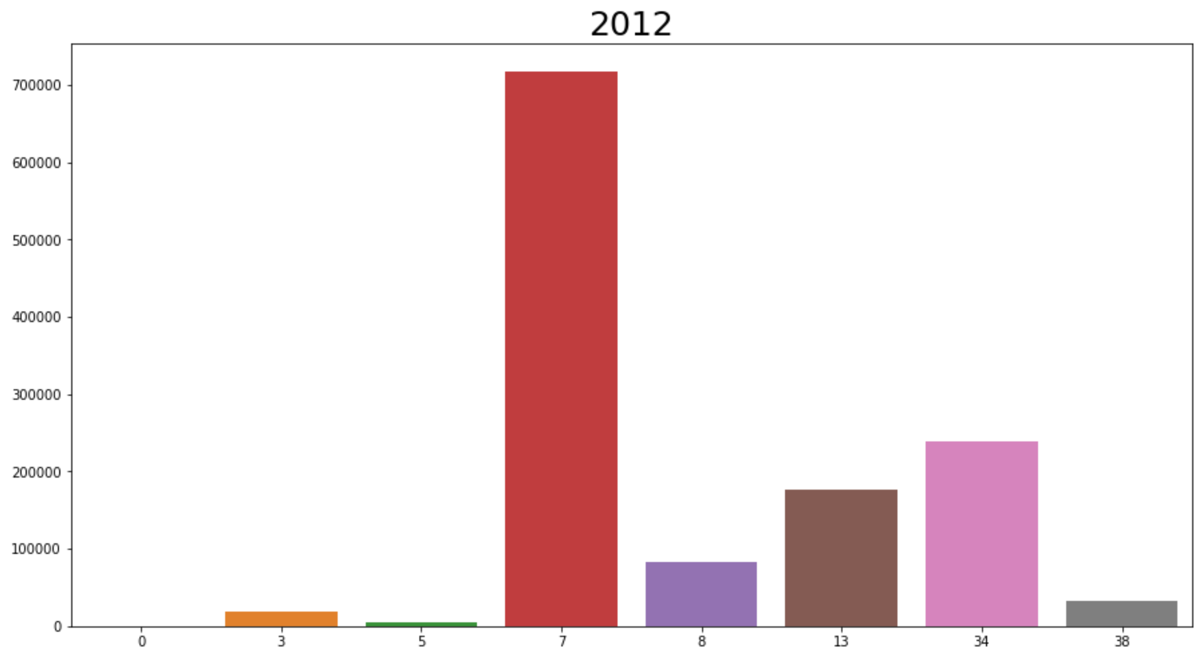
.

- Date: The date of the transactions under the format "Day.Month.Year".
- Time: The time of the transactions under the format "Hour:Minute".
- Cash point: Numerical discret attribute contains the id of the cash point.

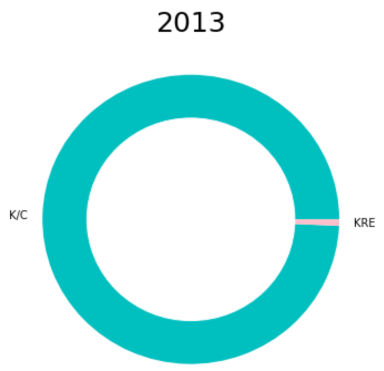
  1, 2 : first location:Lehel

  11, 12, 13 ,14 : second location: Giesing
- Receipt Number: Numerical discret attribute contains the receipt number of the transaction which must be unique but can repeated if it have many purshases at the same time.

- Cashier: Numerical discret attribute contains the id of the cashier.

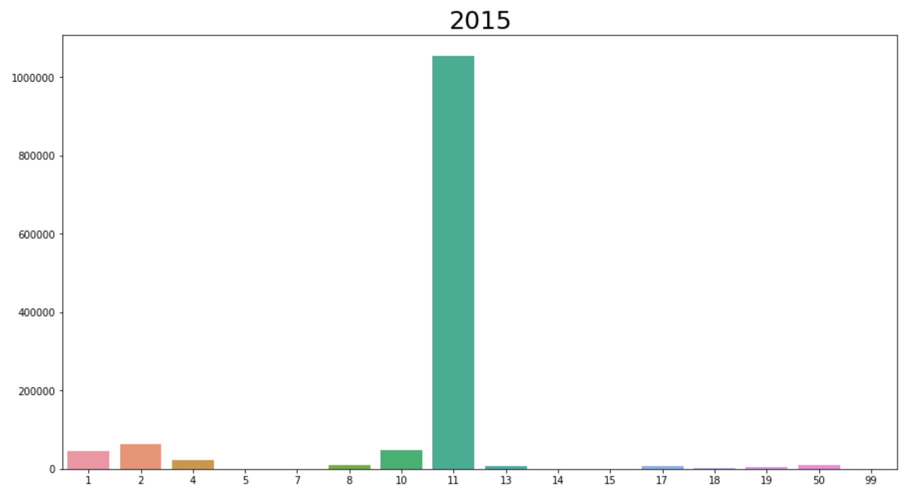


- Payment Type: Categorical nominal attribute descripes the type of payment where it could be by cash or by card.

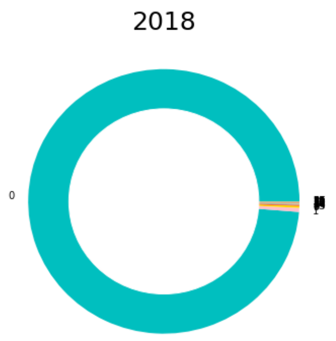


- CardID: Numerical discret attribute contains the id of client's card of course if it's paid by card.

- Firm: Numerical discret attribute contains the firm's id.

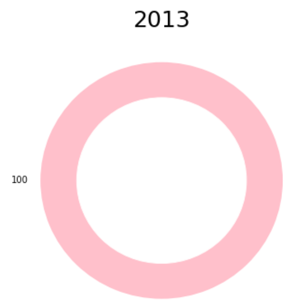


- Department: Numerical discret attribute contains the departments's id.

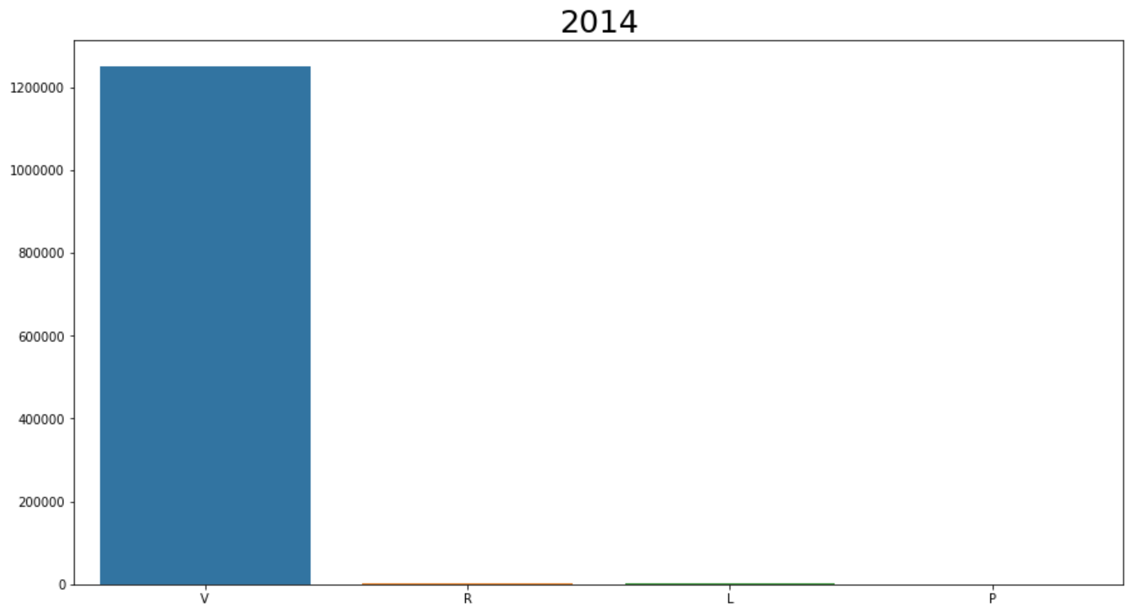


- Cost Unit: Numerical discret attribute contains the cost unit.

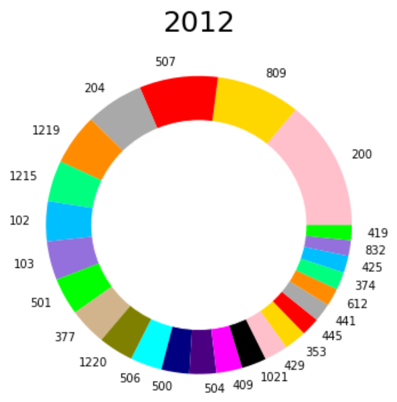
- SubventionLevel: Numerical continuous attribute defines the percentage of the sybvention.



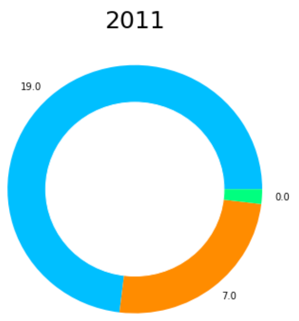
- PositionType: Categorical nominal attribute defines the type of the transaction.



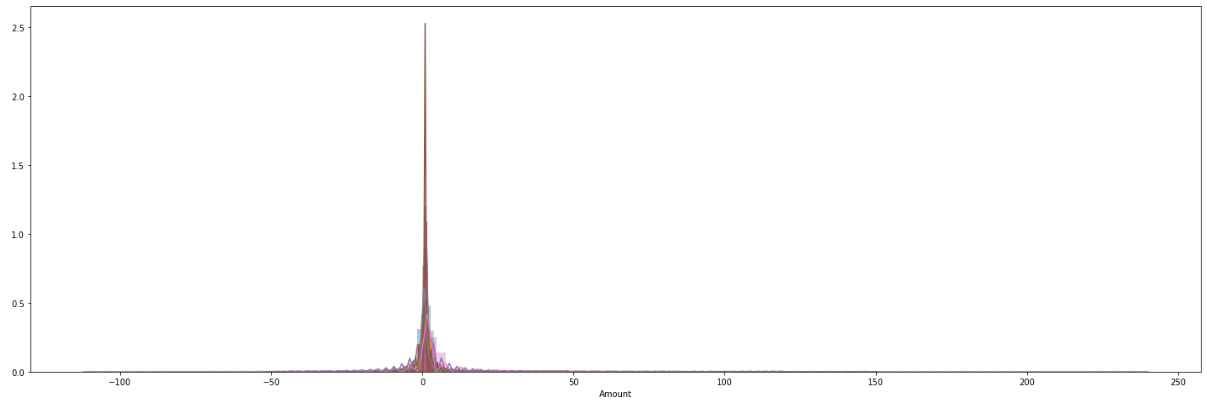
- ItemNumber: Numerical discret attribute contains the item's id.



- Taxes: Numerical continuous attribute describes the amount of the tax.



- Amount: Numerical discret attribute defines the quantity of the items.

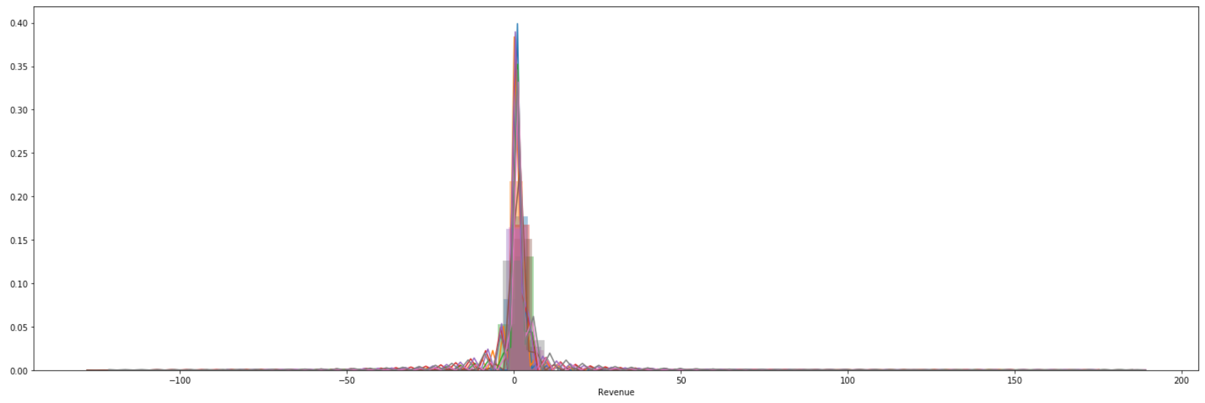


Mean:0.9959062898603541

Median:1.0

Standard Deviation:0.31645895801996005

- Revenue: Numerical continuous attribute of the price of the item.

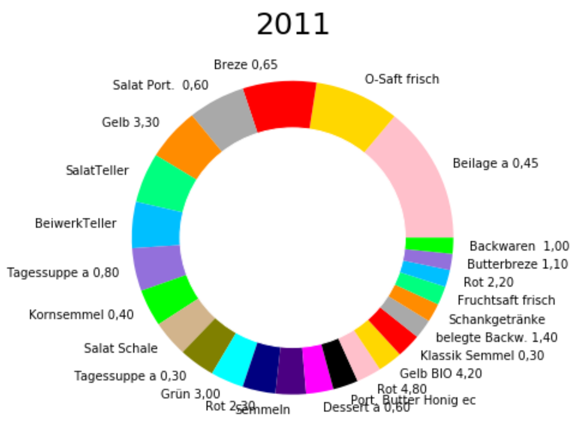


Mean:1.459942435748651

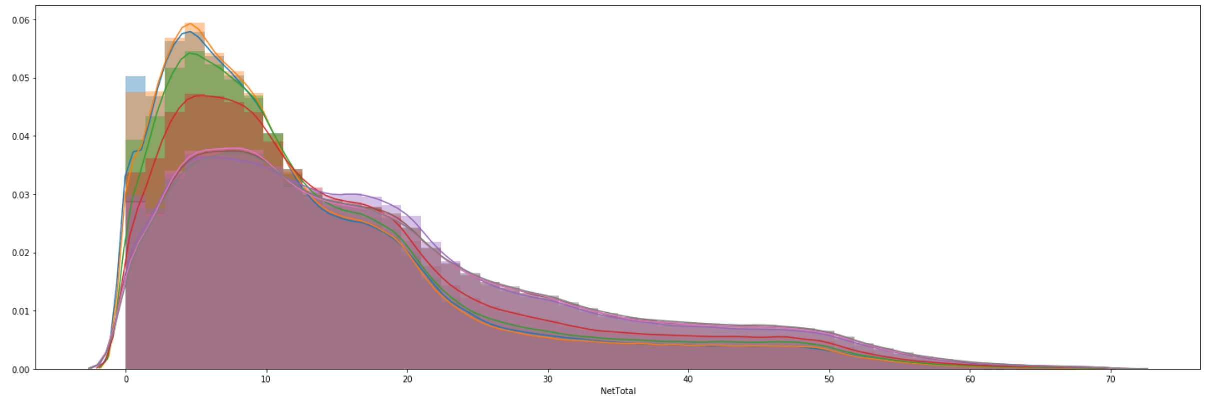
Median:1.0

Standard Deviation:3.340030961719746

- ItemDescription: Categorical nominal attribute descripes the item mainly it's name.



- NetTotal: Numerical continuous attribute contains the sum of the price in the reciept.

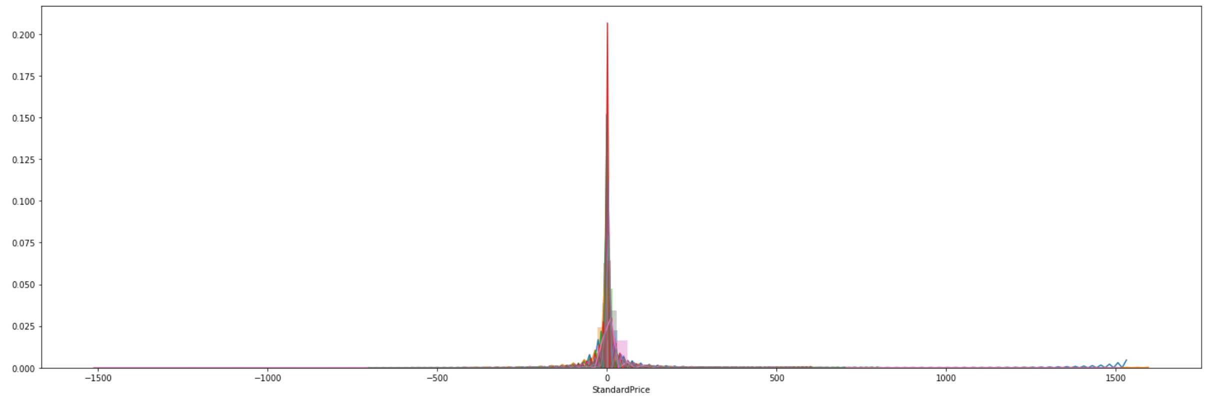


Mean:257.13497314851753

Median:15.63

Standard Deviation:1522.7393072745479

- StandardPrice: Numerical continuous attribute of the price of the item.

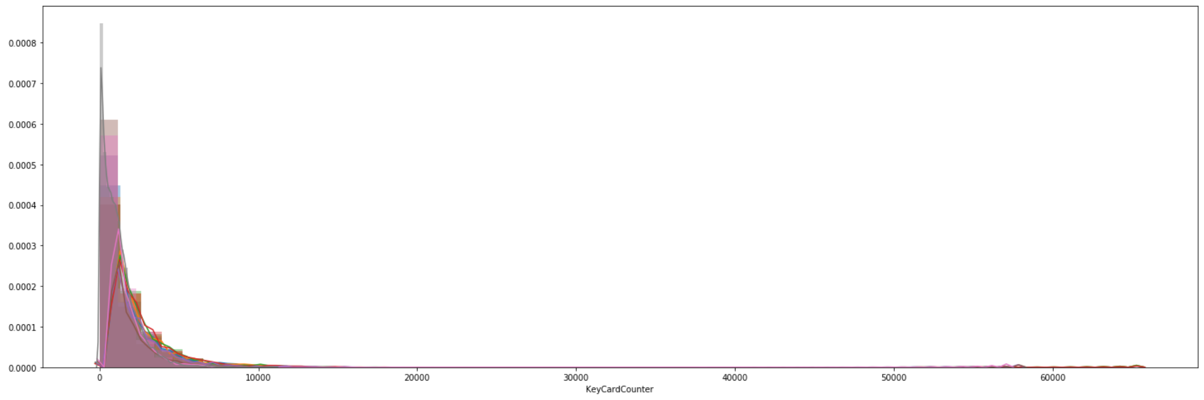


Mean:0.8697674935895499

Median:0.95

Standard Deviation:5.92479401923214

- KeyCardCounter: Numerical discret attribute describes the total of purchases by a client or card holder.

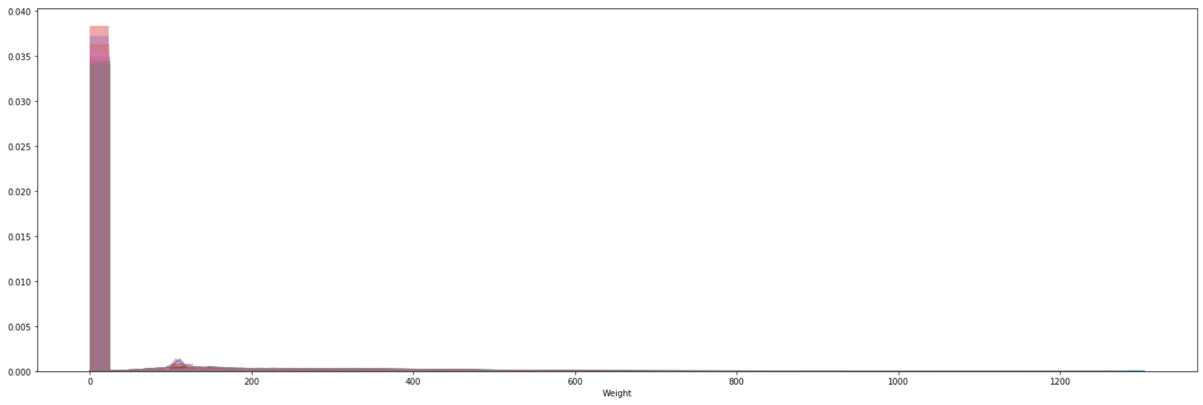


Mean:1179.0985907035301

Median:913.0

Standard Deviation:1097.1659294972333

- Weight: Numerical continuous attribute of the weight of some items.

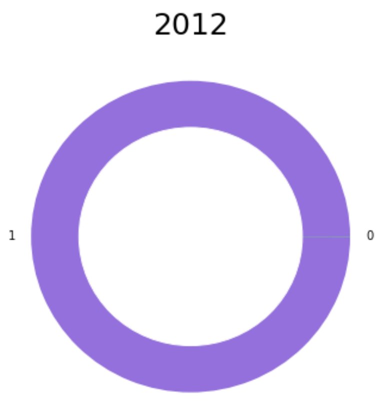


Mean:25.641302809833594

Median:0.0

Standard Deviation:90.99198942048679

- CurrencyFlag: Numerical discret attribute describes the currency used.



2.3 Conclusion

In this chapter we were able to look at the history of inventory management data of Mitakus, seeing that it dates back to 2011.

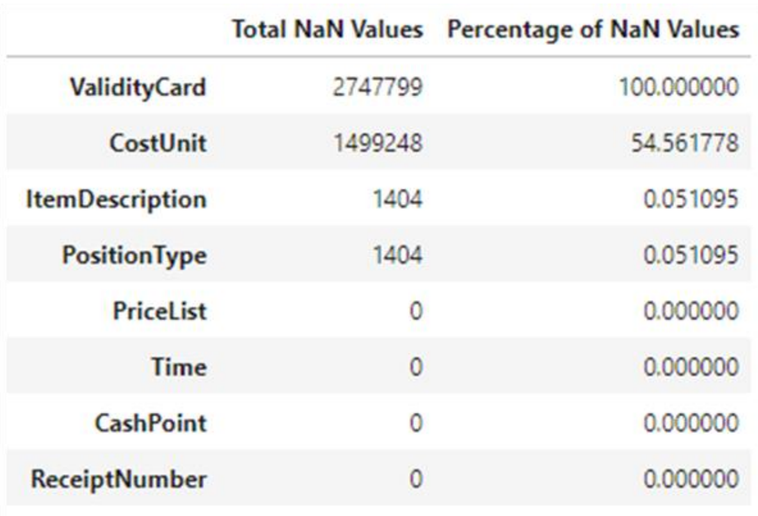
Such immense labeled data will be significatively important to make our predictions which will assure the project's study, economical, legal, operational and technical feasibility that will enable

CHAPTER THREE: DATA PREPARATION

3.1 Data Cleaning

3.1.1 Missing Data

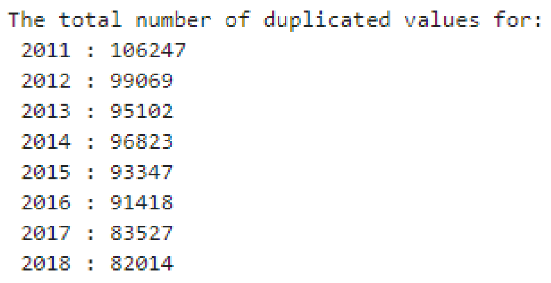
After showing the percentage of missing data we notice that all the values of 'ValidityCard' and around 50% of values of 'CostUnit' are missing so we will ignore them since they are not really useful in our context. While, for the variables 'ItemDescription' and 'PositionType' we will take care of them later.



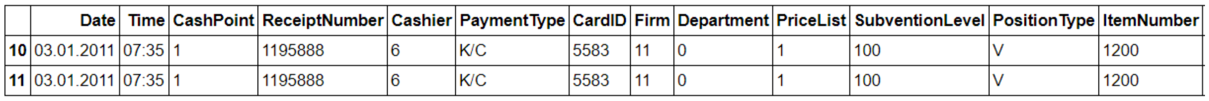
	Total NaN Values	Percentage of NaN Values
ValidityCard	2747799	100.000000
CostUnit	1499248	54.561778
ItemDescription	1404	0.051095
PositionType	1404	0.051095
PriceList	0	0.000000
Time	0	0.000000
CashPoint	0	0.000000
ReceiptNumber	0	0.000000

3.1.2 Duplicated Data

We will get rid of all the duplicated values (of course we will keep one as the 'drop.duplicates' keeps the first sample)



```
The total number of duplicated values for:
 2011 : 106247
 2012 : 99069
 2013 : 95102
 2014 : 96823
 2015 : 93347
 2016 : 91418
 2017 : 83527
 2018 : 82014
```



	Date	Time	CashPoint	ReceiptNumber	Cashier	PaymentType	CardID	Firm	Department	PriceList	SubventionLevel	PositionType	ItemNumber
10	03.01.2011	07:35	1	1195888	6	K/C	5583	11	0	1	100	V	1200
11	03.01.2011	07:35	1	1195888	6	K/C	5583	11	0	1	100	V	1200

This is a sample of a duplicated records, if it was a different purchase on the same receipt the net total should be the sum of all the previous and the current purchases.

3.1.3 Item Description cleaning

We wrote a script, that gets form every excel file the menu for that week and for each day the right name and the wrong name of dishs.

According to the file name will extract the year, week and place and then we will change the dish names to match the menu.

We have noticed that some of dish names haven't changed and that's because some of the wrong names doesn't appear in the menu, also some cells in the menu are with empty colors (for example 8th week 2016 - Lehel) And we have some missing files (for example 2nd week 2016 - Lehel).

We have used the 3rd sheet of each excel file but we have noticed that some values are correct on the 1st sheet but are wrong on the 3rd which makes some dishs unfilled (for example week 10 - Lehel).

We have noticed some similarities between Lehel and Giesing on the menu for the same weeks so we can use these similarities to fill the untreated wrong names.

3.1.4 Check anomality

• **DATE**: We are going to convert the 'Date' to datetime type so that we can detect the day,

month and year for every transaction. For every data set we will accept only a valid date of each

year.

• **Net Total** According to the distribution we have some outliers we are going to

eliminate them.

• **Standard Price** We have some outliers we are going to eliminate them such as the Standard price must be positive.

• **Cash Points** We are only interested the transactions from Cashpoints 1 and 2 for Lehel and Cashpoints 11, 12, 13 and 14 for Giesing.

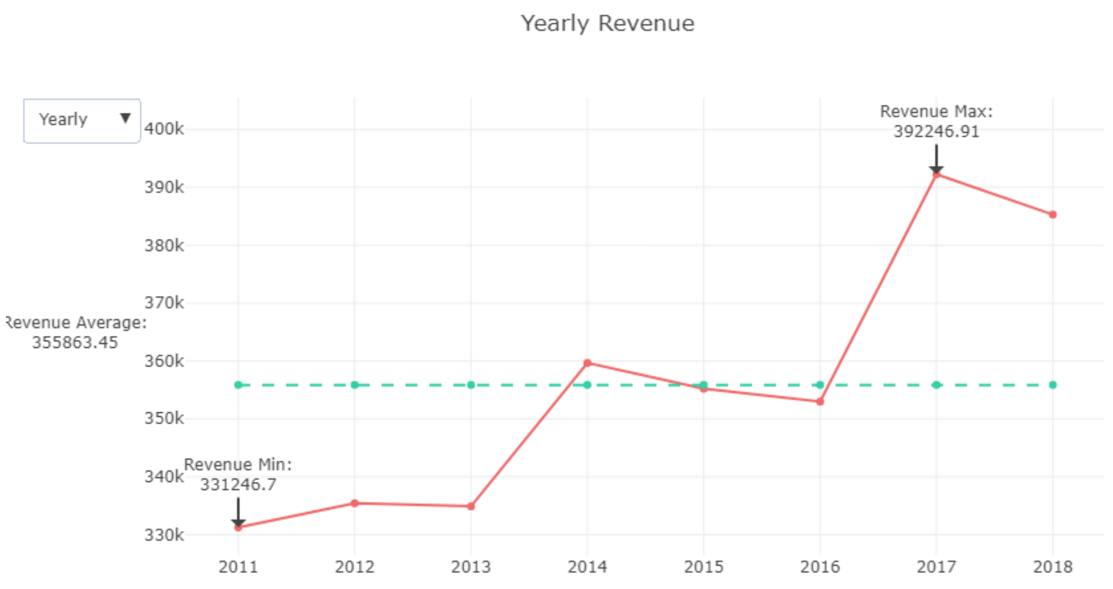
• **Key Card Counter** We have some outliers we are going to eliminate, for example it's hard that a client makes 40k purchase with an average of 13 purchases daily that is why we will eliminate them.

• **Weight** Since not all the items have the attribute weight the distribution is higher than zero we wanted to see the global weight of the weights of the measurable items. According to the distribution it's highly distributed around 80 g.

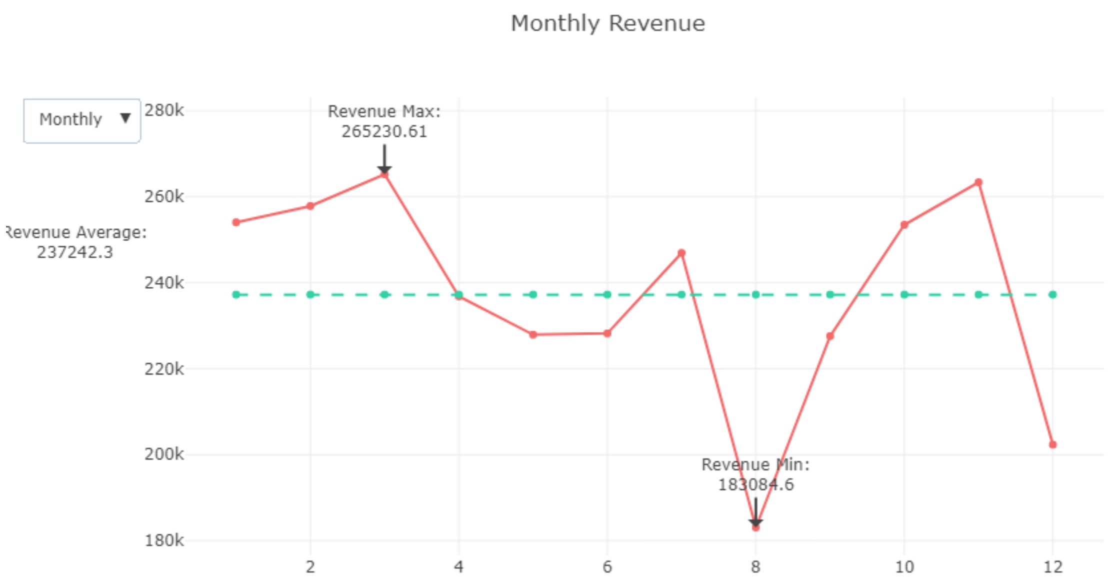
• **Currency Flag** Always equals 1 which stands for euro so we are going to ignore it.

3.2 Data Visualization

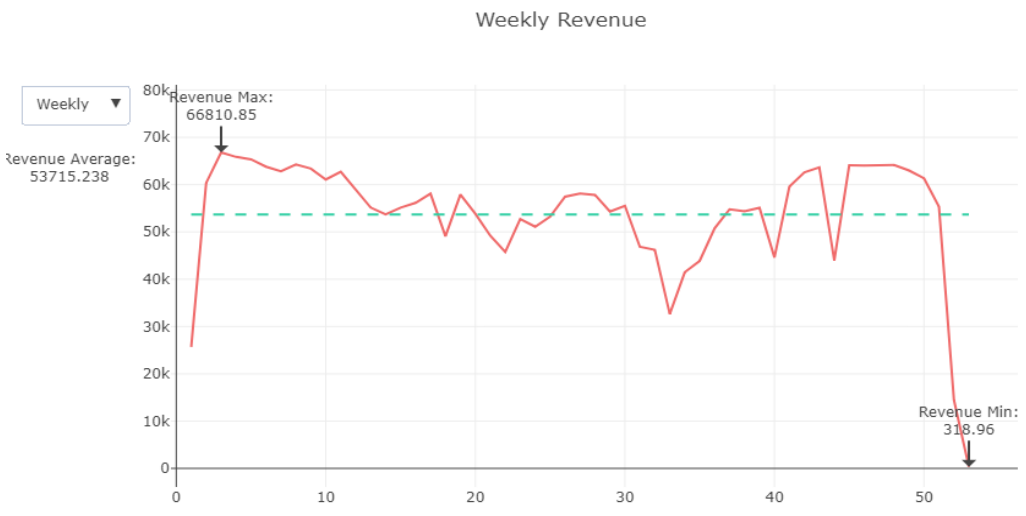
3.2.1 Revenue



Yearly : We can notice that the steady growth of the revenue from 2011 until 2013, the revenue range was between 1.72M and 1.77M, but the change started between 2013 and 2014 we can see a clear improvement and a strong progress as the revenue went from 1.77M to almost 1.86M, in just one year, the growth of the revenue is almost the double of its growth in the last 3 years And maybe this is due to the introduction of new plats or the improvement of their marketing strategies or the growth of the number of employees in the area. In 2016 the revenue reaches its maximum value which is 1.91 M but we can clearly see how the revenue dropped between 2016 and 2017 but hopefully in 2018 the revenue shows a slight improvement.



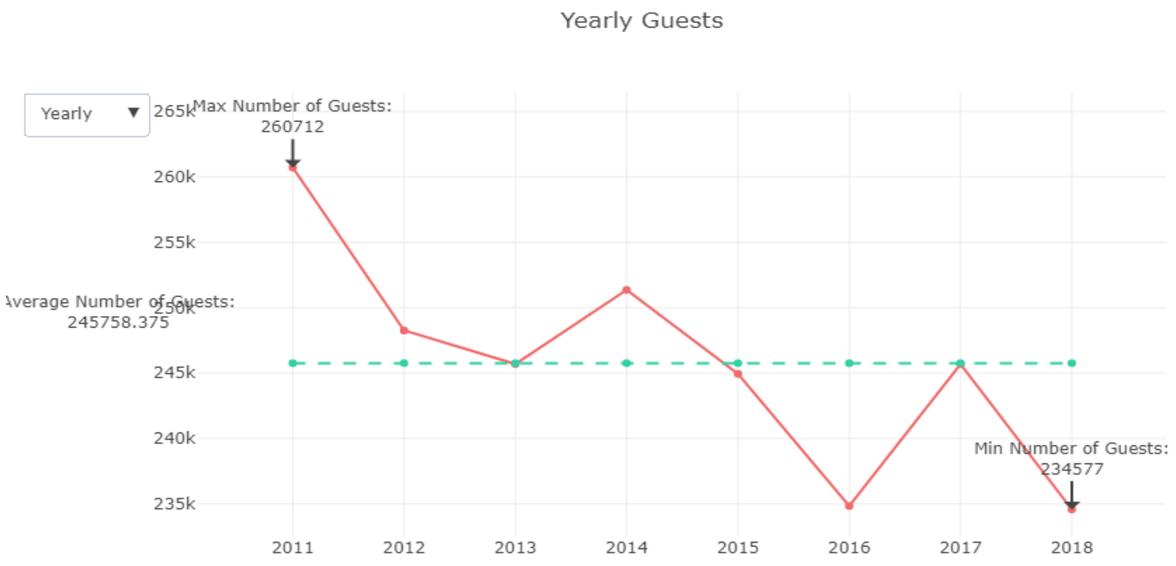
Monthly : the revenue reaches its maximum(1.35M) in march and maybe it is related to the weather it's the spring and usually employees/workers/interns tend to go take their lunch out, However the minimum value was detected in December and it's kind of obvious because December means the "year-end holidays" and many companies close during this period so the number of clients will decrease and automatically the revenue will act the same.

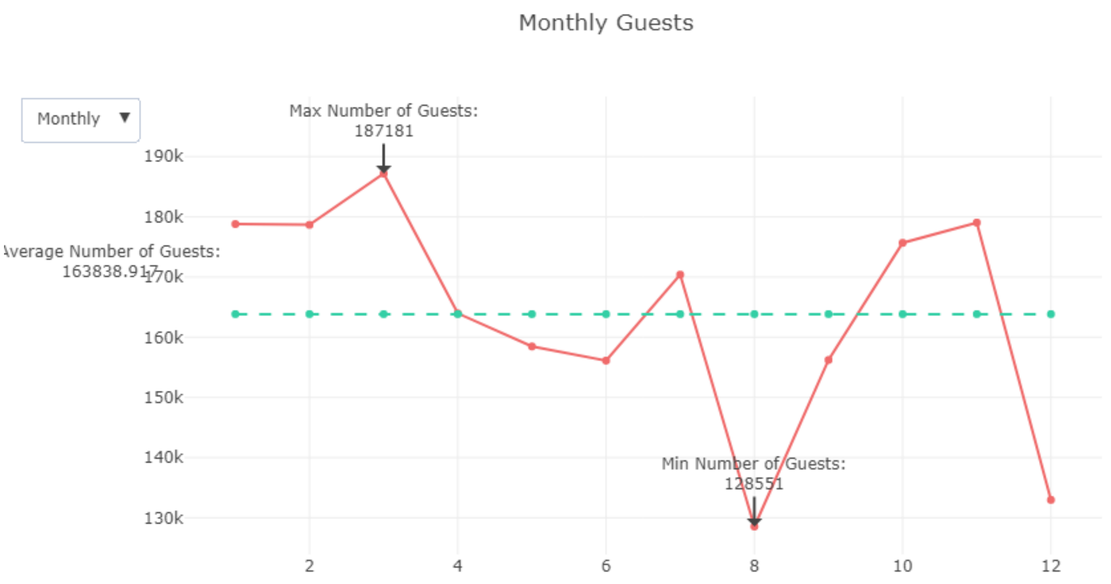


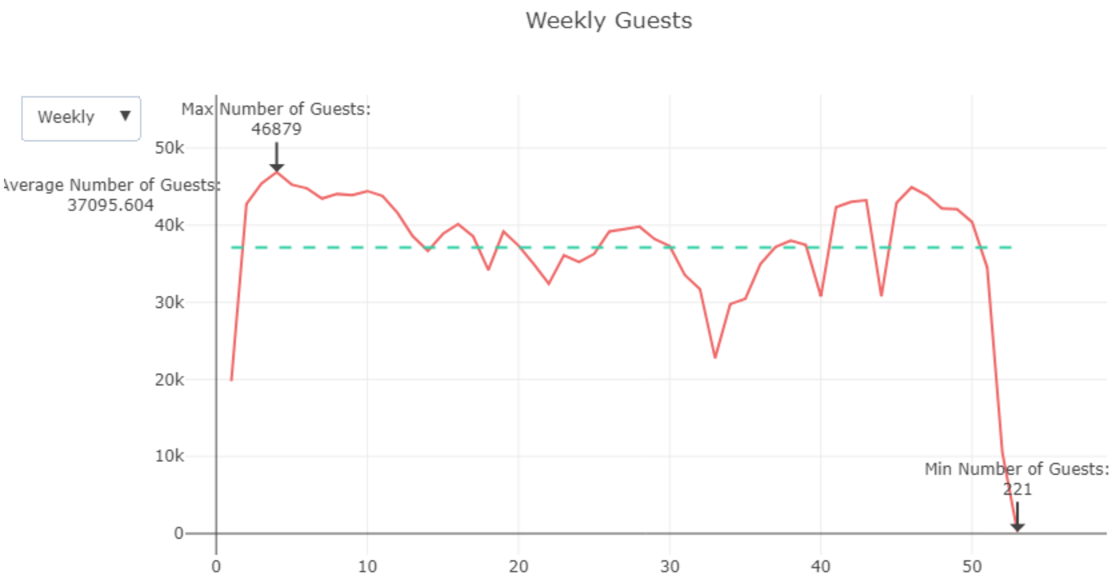
Weekly: the revenue reaches its maximum(340.7K) in the third week of the year, and as we have already seen the minimum value was detected in December and here we have the same results we can see that the minimum value (10.54k) is in the last week of the year.

⇨ Geising revenue is much bigger than Lehel

3.2.2 Number of Guests

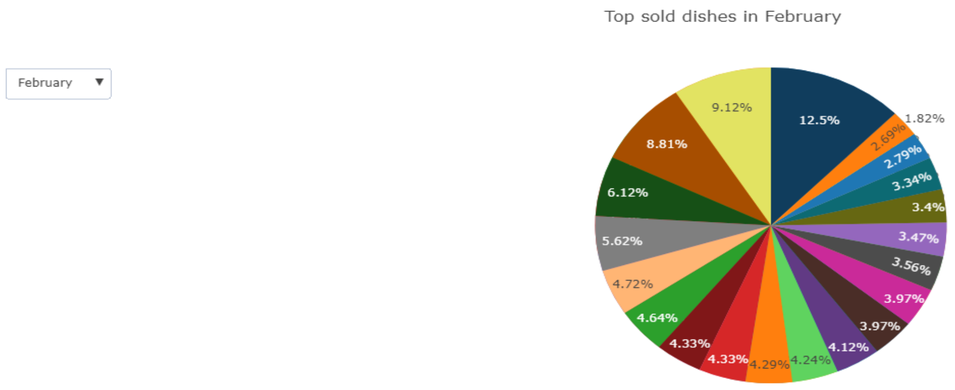


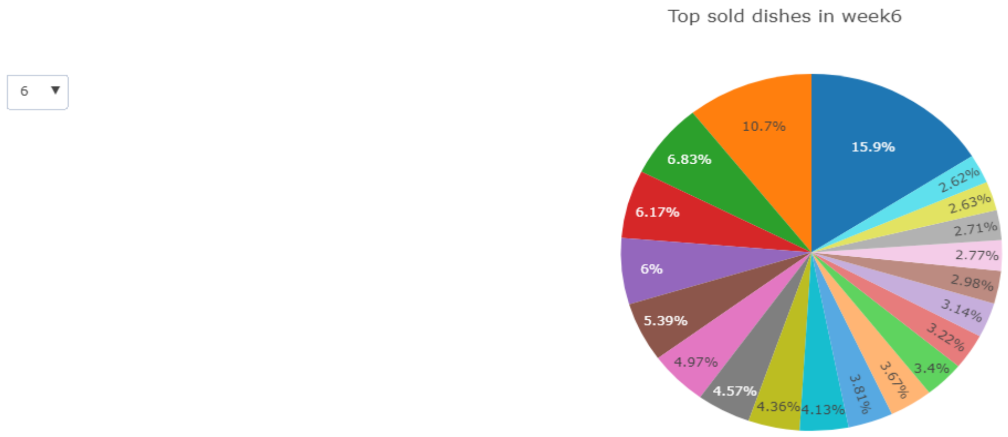


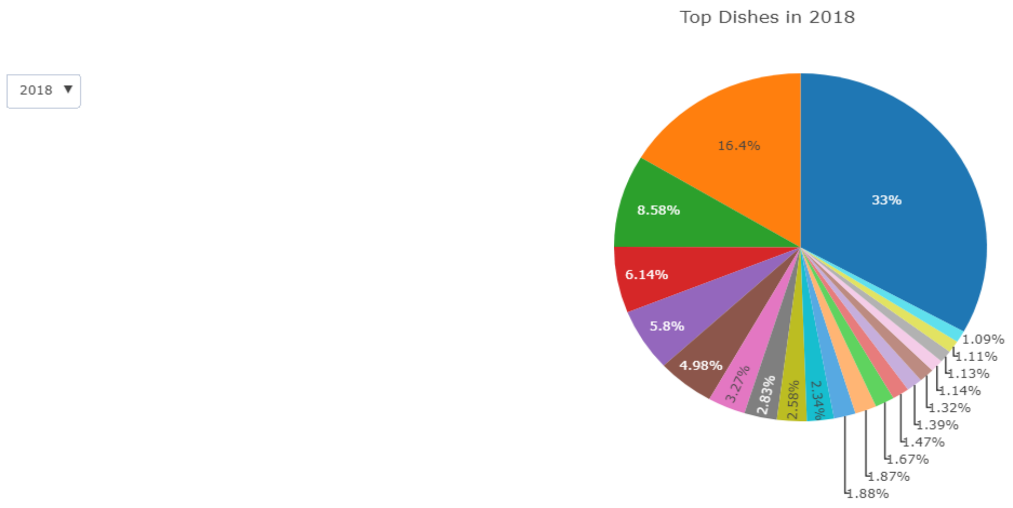


The number of guests and revenue are highly correlated, and it's totally normal if we have a high number of guests the revenue will automatically increase. The Revenue and the number of guests have the same behaviour as the revenue. We notice a shortage of the numbre in clients in teh period of holidays.

3.2.3 Dishes


February ▼
Top sold dishes in February
12.5%
1.82%
2.69%
2.79%
3.34%
3.4%
3.47%
3.56%
3.97%
3.97%
4.12%
4.24%
4.29%
4.33%
4.33%
4.64%
4.72%
5.62%
6.12%
8.81%
9.12%


6 ▼
Top sold dishes in week6
15.9%
2.62%
2.63%
2.71%
2.77%
2.98%
3.14%
3.22%
3.4%
3.67%
3.81%
4.13%
4.36%
4.57%
4.97%
5.39%
6%
6.17%
6.83%
10.7%


2018 ▼
Top Dishes in 2018
33%
1.09%
1.11%
1.13%
1.14%
1.32%
1.39%
1.47%
1.67%
1.87%
1.88%
2.34%
2.58%
2.83%
3.27%
4.98%
5.8%
6.14%
8.58%
16.4%

In 2011 gelb 3.30 was the top dish withb 17.8% , then Grun 3 (10.9%) also Rot (10.9%) on the other side Bio dishes like Rot BIO 2.90 and Gelb Bio 2.8 were on the bottom with 1.11% and 1.67% After that in 2012 Gelb Bio 2.8 made an epic evolution to be the top dish with 12.6% along side with Grun (12.6%). Rot 2.30 pourcentage went from the top thrid in 2011 to only 3.75% in 2012 and Rot 2.2 was the last with 1.82%. in 2013 there was no remarquable change. from 2014 to 2016 the top 3 dishes were the same , grun 2.45 then Rot 2.80 and after that Gelb 2.80 in 2017 grun 1.99 was on the top with 34.8% which is a record pourcentage in the last 6 years and then Rot 3.2 with 10.8% and also Gelb 3.80 with 7.32% on the third place , Rot 2.3 was the last dish with only 1.2% of the dishes in 2018 Grun 1.99 was on the top again with 33.0% and on the second place Rot 3.60 with 16.4%. on the bottom we can find Gelb Bio 4.20 with only 1.09%.

CHAPTER FOUR : FEATURE ENGINEERING AND FEATURE SELECTION

4.1 Feature engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

❖ **Basic Features**

When we look at a date time stamp, a number of features, or pieces of information are immediately obvious:

- ✓ Year
- ✓ Month
- ✓ Day
- ✓ Day of week
- ✓ Week of year
- ✓ Hour of day

Month and day of week can be quite useful in understanding periodicity or seasonality of transactions. We may find that some actions are more probable on certain days of the week, or somethings happen around the same month every year.

❖ **Adding Part of the day feature**

One of the questions frequently asked was about a content recommender and what is the right time to display what type of recipe.
We used this time of day map to understand when people see breakfast recipe, kids lunch box recipes, appetizers etc.
This feature was extracted from clickstream to enrich the data and give additional insight into what types of recipes to show at what time.

❖ Seasonality

Season was also a good predictor to understand which recipes are timeless and which are more seasonal.

❖ Holidays

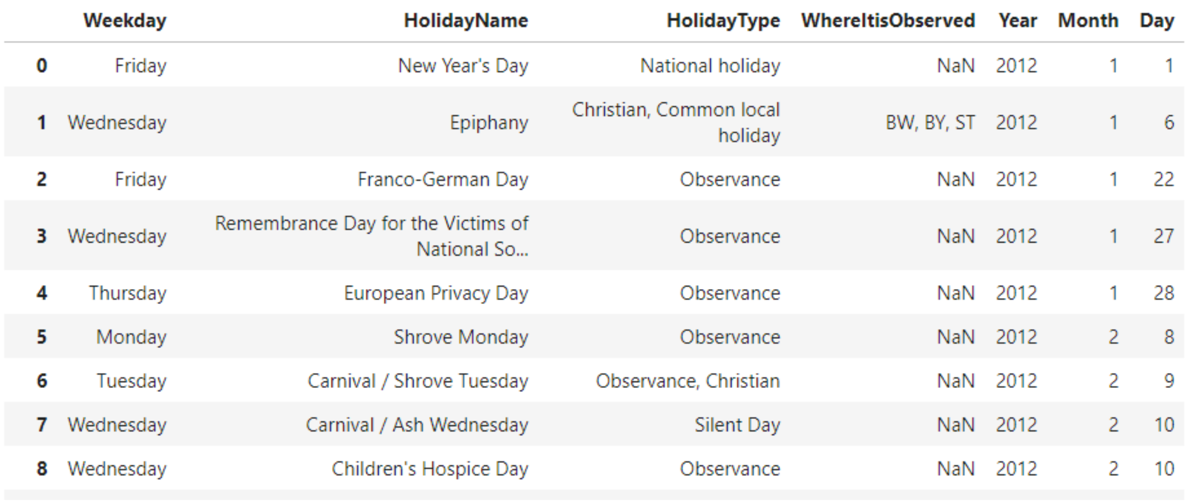
Retail is a very seasonal business. Often, people are buying for an occasion or near an occasion. Intuitively, people may be purchasing for Valentine's day, Thanksgiving due to all the great sales, Christmas etc. To understand which customers are more driven by these special occurrences, a set of features need to be created that measures distances to these occurrences. Following steps are required to make this work:

- Pull a list of holidays/occurrences from a data source/API for a given geography.
- Create a pandas data frame of these.
- Merge the holidays dataset with our dataset.

❖ Pull a list of holidays and create a DataFrame

There are a number of public sources like Wikipedia or sites that provides an API like timeanddate.com. For the purpose of this article, knowing that all purchases are from Germany. Further, instead of using API, lets scrape this data from scratch

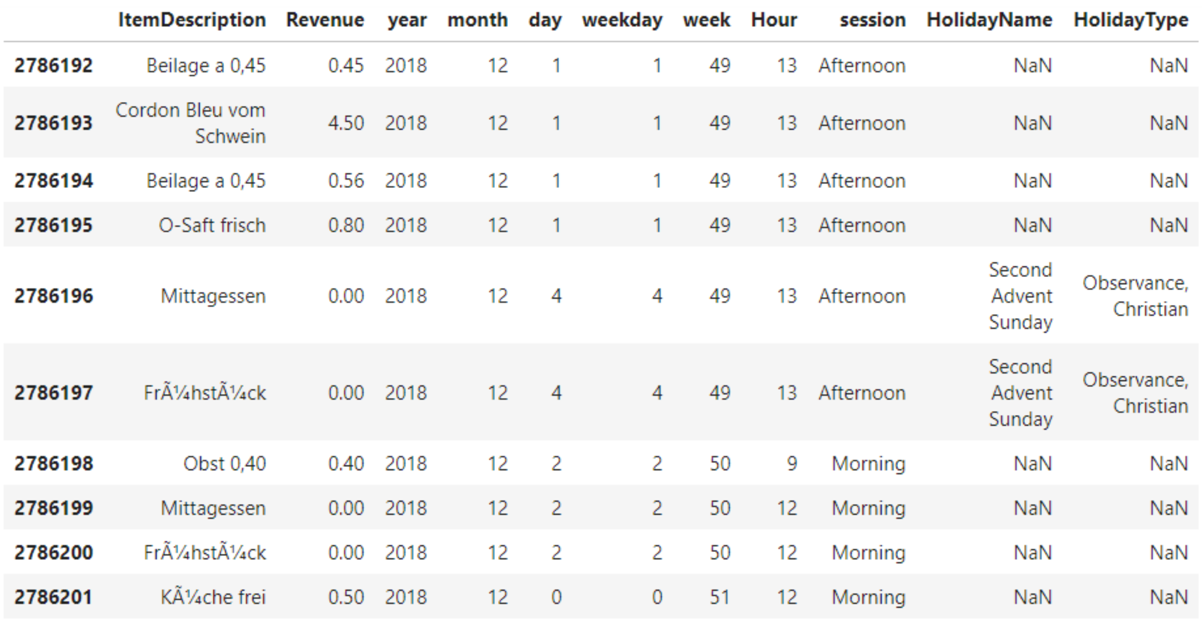
We will use Beautiful Soup to extract the data and create a list of dictionary objects that can be loaded into pandas DataFrame



	Weekday	HolidayName	HolidayType	WhereItisObserved	Year	Month	Day
0	Friday	New Year's Day	National holiday	NaN	2012	1	1
1	Wednesday	Epiphany	Christian, Common local holiday	BW, BY, ST	2012	1	6
2	Friday	Franco-German Day	Observance	NaN	2012	1	22
3	Wednesday	Remembrance Day for the Victims of National So...	Observance	NaN	2012	1	27
4	Thursday	European Privacy Day	Observance	NaN	2012	1	28
5	Monday	Shrove Monday	Observance	NaN	2012	2	8
6	Tuesday	Carnival / Shrove Tuesday	Observance, Christian	NaN	2012	2	9
7	Wednesday	Carnival / Ash Wednesday	Silent Day	NaN	2012	2	10
8	Wednesday	Children's Hospice Day	Observance	NaN	2012	2	10

❖ **Create the holiday features**

Logic for creating these features is to take each row in our data, compare the date of purchase to every row in the holidays frame created.



	ItemDescription	Revenue	year	month	day	weekday	week	Hour	session	HolidayName	HolidayType
2786192	Beilage a 0,45	0.45	2018	12	1	1	49	13	Afternoon	NaN	NaN
2786193	Cordon Bleu vom Schwein	4.50	2018	12	1	1	49	13	Afternoon	NaN	NaN
2786194	Beilage a 0,45	0.56	2018	12	1	1	49	13	Afternoon	NaN	NaN
2786195	O-Saft frisch	0.80	2018	12	1	1	49	13	Afternoon	NaN	NaN
2786196	Mittagessen	0.00	2018	12	4	4	49	13	Afternoon	Second Advent Sunday	Observance, Christian
2786197	FrÃ¼hstÃ¼ck	0.00	2018	12	4	4	49	13	Afternoon	Second Advent Sunday	Observance, Christian
2786198	Obst 0,40	0.40	2018	12	2	2	50	9	Morning	NaN	NaN
2786199	Mittagessen	0.00	2018	12	2	2	50	12	Morning	NaN	NaN
2786200	FrÃ¼hstÃ¼ck	0.00	2018	12	2	2	50	12	Morning	NaN	NaN
2786201	KÃ¼che frei	0.50	2018	12	0	0	51	12	Morning	NaN	NaN

4.2 Feature Selection

The model needs features, and all we have is a 1-dimentional time series. What features can we extract?

- Lags of time series
- Window statistics:
  - Max/min value of series in a window
  - Average/median value in a window
  - Window variance
  - etc.
- Date and time features:
  - Minute of an hour, hour of a day, day of the week, and so on
  - Is this day a holiday? Maybe there is a special event? Represent that as a Boolean feature
- Target encoding
- Forecasts from other models (note that we can lose the speed of prediction this way)

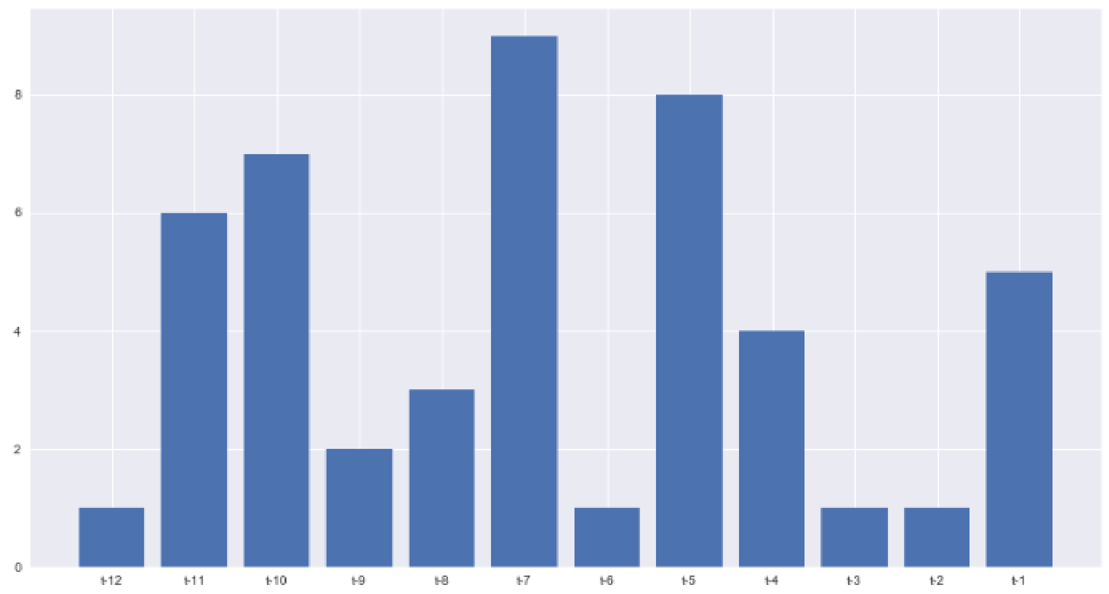
Lags of time series

Shifting the series n steps back, we get a feature column where the current value of time series is aligned with its value at time $t{-}n$. If we make a 1 lag shift and train a model on that feature, the model will be able to forecast 1 step ahead from having observed the current state of the series. Increasing the lag, say, up to 6, will allow the model to make predictions 6 steps ahead; however, it will use data observed 6 steps back. If something fundamentally changes the series during that unobserved period, the model will not catch these changes and will return forecasts with a large error. Therefore, during the initial lag selection, one has to find a balance between the optimal prediction quality and the length of the forecasting horizon.

Feature selection of Lag variables

We used feature selection to automatically identify and select those input features that are most predictive. A popular method for feature selection is called Recursive Feature Selection (RFE). RFE works by creating predictive models, weighting features, and pruning those with the smallest weights, then repeating the process until a desired number of features are left. The example below uses RFE with a random forest predictive model and sets the desired number of input features to 4.

```
Selected Features:
t-12
t-6
t-3
t-2
```



Bar graph showing the feature selection rank

CHAPTER FIVE: MODELING

5.1 Quality metrics

Before we begin forecasting, let's understand how to measure the quality of our predictions and take a look at the most commonly used metrics.

R squared: coefficient of determination (in econometrics, this can be interpreted as the percentage of variance explained by the model), $]-\infty,1]$

Mean Absolute Error: this is an interpretable metric because it has the same unit of measurement as the initial series, $[0,+\infty[$

Median Absolute Error: again, an interpretable metric that is particularly interesting because it is robust to outliers, $[0,+\infty[$

Mean Squared Error: the most commonly used metric that gives a higher penalty to large errors and vice versa, $[0,+\infty[$

Mean Squared Logarithmic Error: practically, this is the same as MSE, but we take the logarithm of the series. As a result, we give more weight to small mistakes as well. This is usually used when the data has exponential trends, $[0,+\infty[$

Mean Absolute Percentage Error: this is the same as MAE but is computed as a percentage, which is very convenient when you want to explain the quality of the model to management, $[0,+\infty[$

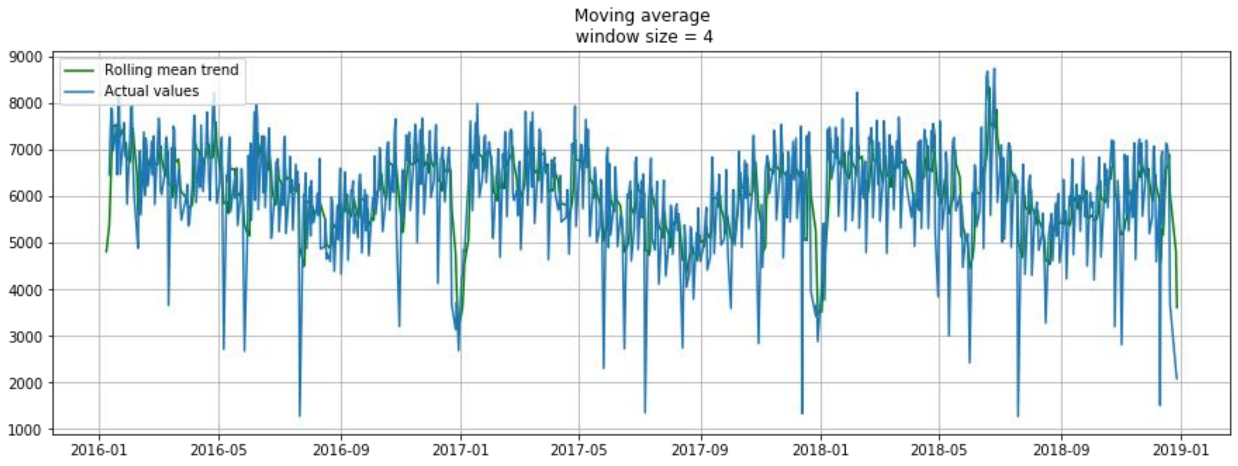
5.2 Time Series Approach

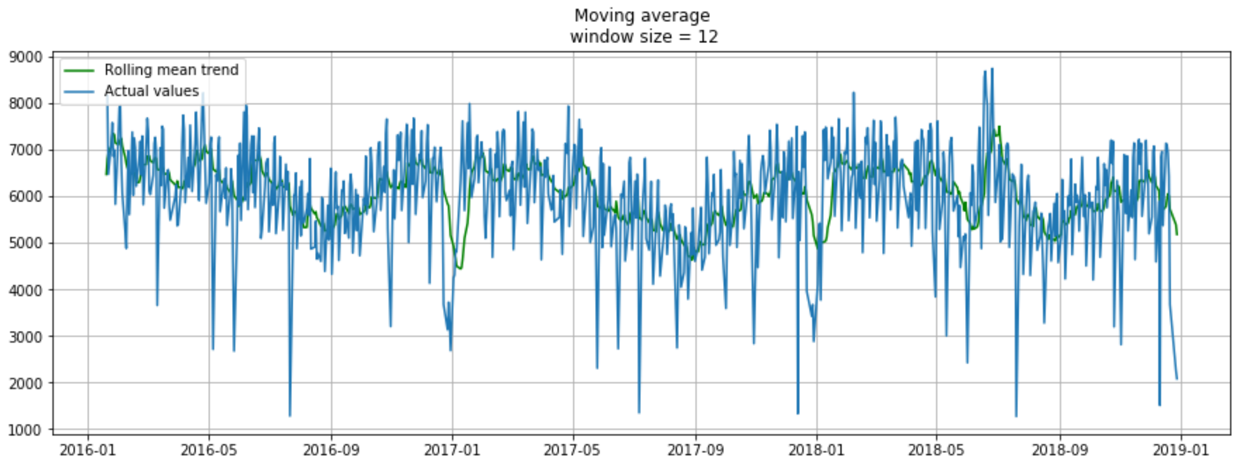
Our data is organized by relatively deterministic dates, that for each we have a perfect sum for the revenue and count for each menu item.

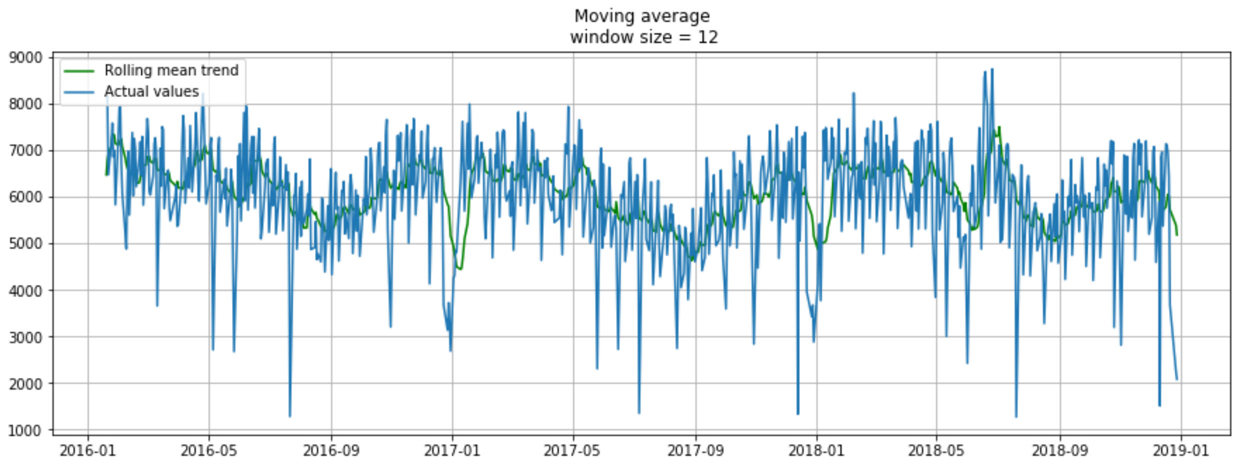
Our goal is to predict the revenue for each day and number of items for each item daily as well.

5.2.1 Exponential smoothing, Holt-Winters model

Let's start with a naive hypothesis: "tomorrow will be the same as today". However, instead of a model like $\hat{y}(t)=y(t-1)$ (which is actually a great baseline for any time series prediction problems and sometimes is impossible to beat), we will assume that the future value of our variable depends on the average of its k previous values. Therefore, we will use the moving average.

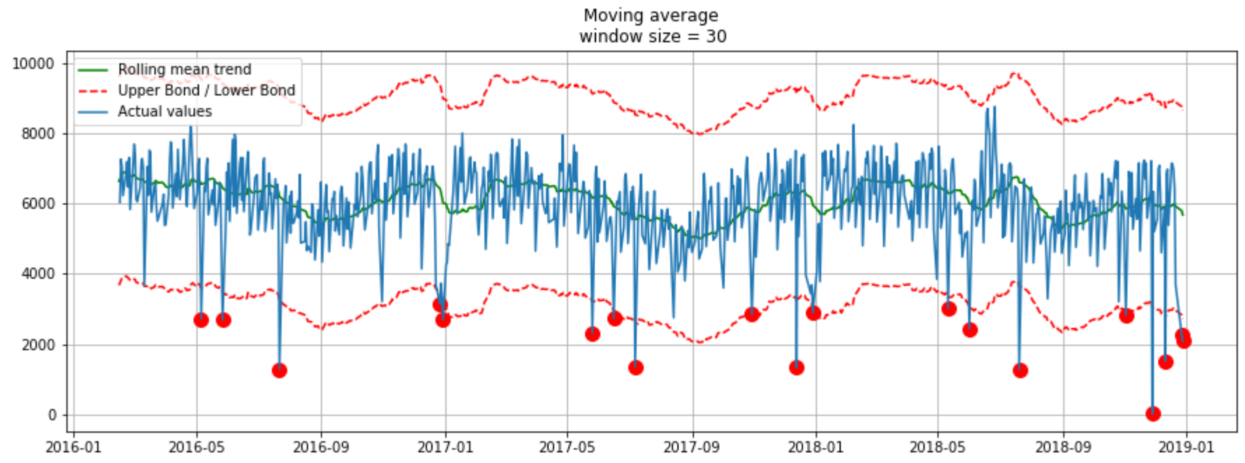






When we applied monthly smoothing on daily data, we could clearly see the dynamics of revenue watched. during the first and the 9th months we observe low values for the revenue as for the rest months of the years it is practically stable

We can also plot confidence intervals for our smoothed values



This was not as great! Here, we can see the downside of our simple approach -- it did not capture the monthly seasonality in our data and marked almost many anomalies. If we want to avoid false positives, we should consider more complex models

Weighted average is a simple modification to the moving average. The weights sum up to 1 with larger weights assigned to more recent observations.

$\hat{y}t=\sum n=1,k\ (\omega n y t+1-n)$

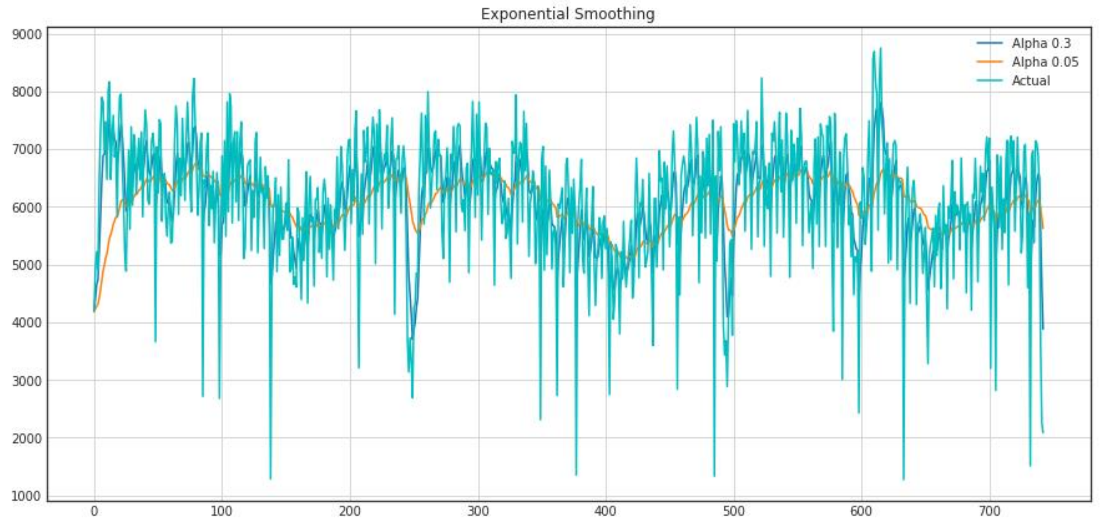
Exponential smoothing

Now, let's see what happens if, instead of weighting the last kk values of the time series, we start weighting all available observations while exponentially decreasing the weights as we move further back in time. There exists a formula for "*exponential smoothing"* that will help us with this:

$$\hat{y}t=\alpha\cdot yt+(1-\alpha)\cdot\hat{y}t-1$$

Here the model value is a weighted average between the current true value and the previous model values. The αα weight is called a smoothing factor. It defines how quickly we will "forget" the last available true observation. The smaller αα is, the more influence the previous observations have and the smoother the series is.

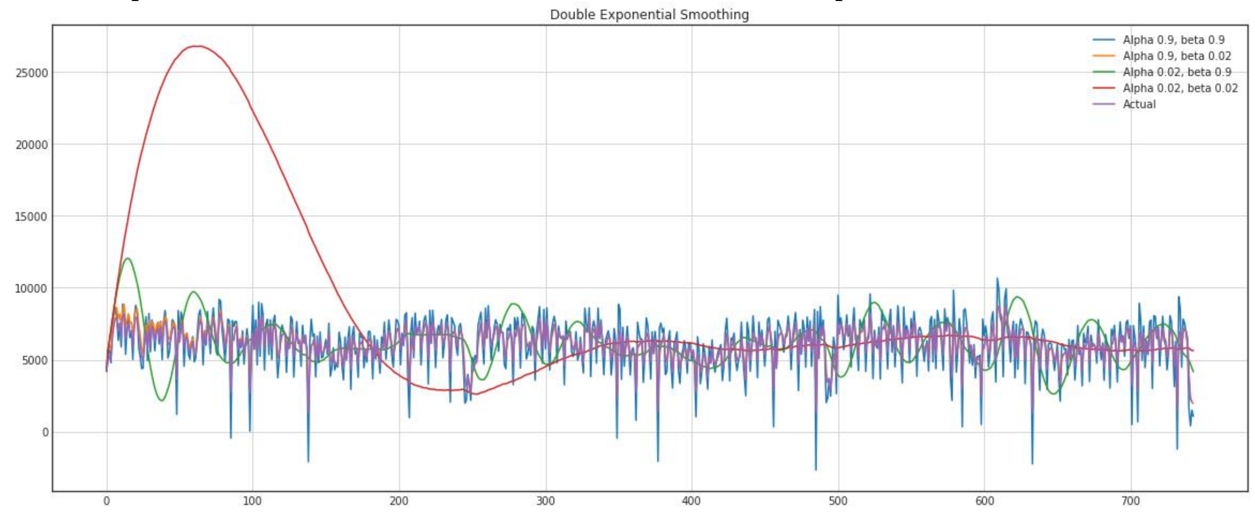
Exponentiality is hidden in the recursiveness of the function -- we multiply by (1−α) each time, which already contains a multiplication by (1−α) of previous model values.



Double exponential smoothing
Up to now, the methods that we've used have been for a single future point prediction (with some nice smoothing). But it is also not enough. Let's extend exponential smoothing so that we can predict two future points (of course, we will also include more smoothing).
Series decomposition will help us -- we obtain two components: intercept (i.e. level) ℓℓ and slope (i.e. trend) bb. We have learnt to predict intercept (or expected series value) with our previous methods; now, we will apply the same exponential smoothing to the trend by assuming that the future direction of the time series changes depends on the previous weighted changes. As a result, we get the following set of functions:

$$\ell x=\alpha yx+(1-\alpha)(\ell x-1+bx-1)$$
$$bx=\beta(\ell x-\ell x-1)+(1-\beta)bx-1$$
$$\hat{y}\,x+1=\ell x+bx$$

The first one describes the intercept, which, as before, depends on the current value of the series. The second term is now split into previous values of the level and of the trend. The second function describes the trend, which depends on the level changes at the current step and on the previous value of the trend. In this case, the ββ coefficient is a weight for exponential smoothing. The final prediction is the sum of the model values of the intercept and trend.



Now we have to tune two parameters: α and β. The former is responsible for the series smoothing around the trend, the latter for the smoothing of the trend itself. The larger the values, the more weight the most recent observations will have and the less smoothed the model series will be. Certain combinations of the parameters may produce strange results, especially if set manually. We'll look into choosing parameters automatically in a bit; before that, let's discuss triple exponential smoothing.

Triple exponential smoothing a.k.a. Holt-Winters

We've looked at exponential smoothing and double exponential smoothing. This time, we're going into triple exponential smoothing.
As you could have guessed, the idea is to add a third component - seasonality. This means that we should not use this method if our time series is not expected to have seasonality. Seasonal components in the model will explain repeated variations around intercept and trend, and it will be specified by the length of the season, in other words by the period after which the variations

repeat. For each observation in the season, there is a separate component; for example, if the length of the season is 30 days (a monthly seasonality), we will have 12 seasonal components, one for each month of the year.

With this, let's write out a new system of equations:

$$\ell x=\alpha(yx-sx-L)+(1-\alpha)(\ell x-1+bx-1)$$

$$bx=\beta(\ell x-\ell x-1)+(1-\beta)bx-1$$

$$sx=\gamma(yx-\ell x)+(1-\gamma)sx-L$$

$$\hat{y}x+m=\ell x+mbx+sx-L+1+(m-1)modL$$

The intercept now depends on the current value of the series minus any corresponding seasonal component. Trend remains unchanged, and the seasonal component depends on the current value of the series minus the intercept and on the previous value of the component. Take into account that the component is smoothed through all the available seasons; for example, if we have a January component, then it will only be averaged with other Januarys.

Now that we have the seasonal component, we can predict not just one or two steps ahead but an arbitrary m future steps ahead, which is very encouraging.

Below is the code for a triple exponential smoothing model, which is also known by the last names of its creators, Charles Holt and his student Peter Winters. Additionally, the Brutlag method was included in the model to produce confidence intervals:

$$\hat{y}max(\mathrm{x})=\ell x-1+bx-1+sx-T+m\cdot dt-T$$

$$\hat{y}min(x)=\ell x-1+bx-1+sx-T-m\cdot dt-T$$

$$dt=\gamma|yt-\hat{y}t|+(1-\gamma)dt-T,$$

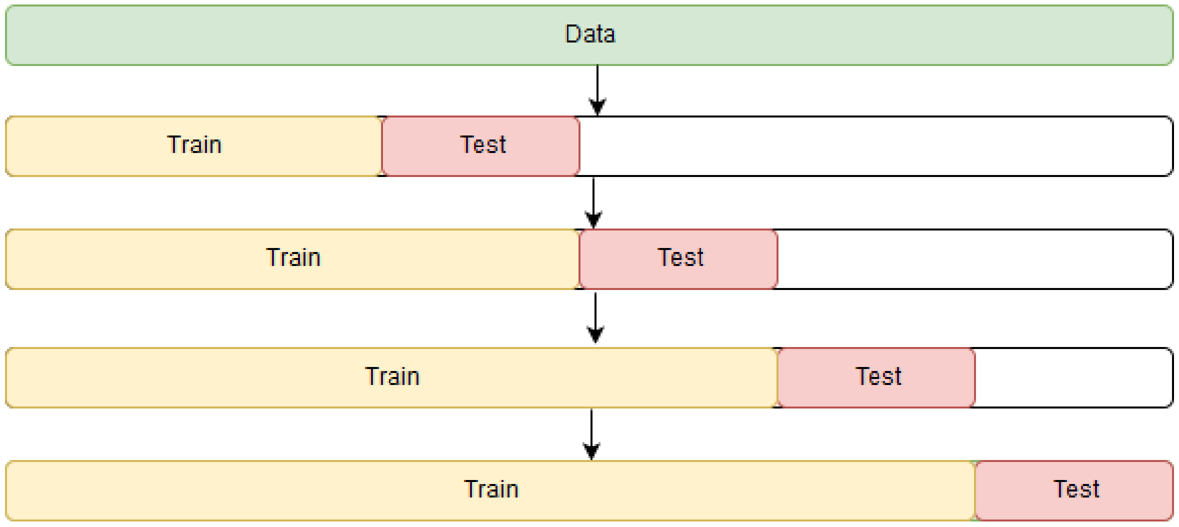
where T is the length of the season, d is the predicted deviation. Other parameters were taken from triple exponential smoothing.

Time-series cross validation, parameters selection

Before we start building a model, let's first discuss how to estimate model parameters automatically. There is nothing unusual here; as always, we have to choose a loss function suitable for the task that will tell us how closely the model approximates the data. Then, using cross-validation, we will evaluate our chosen loss function for the given model parameters, calculate the gradient, adjust the model parameters, and so on, eventually descending to the global minimum.

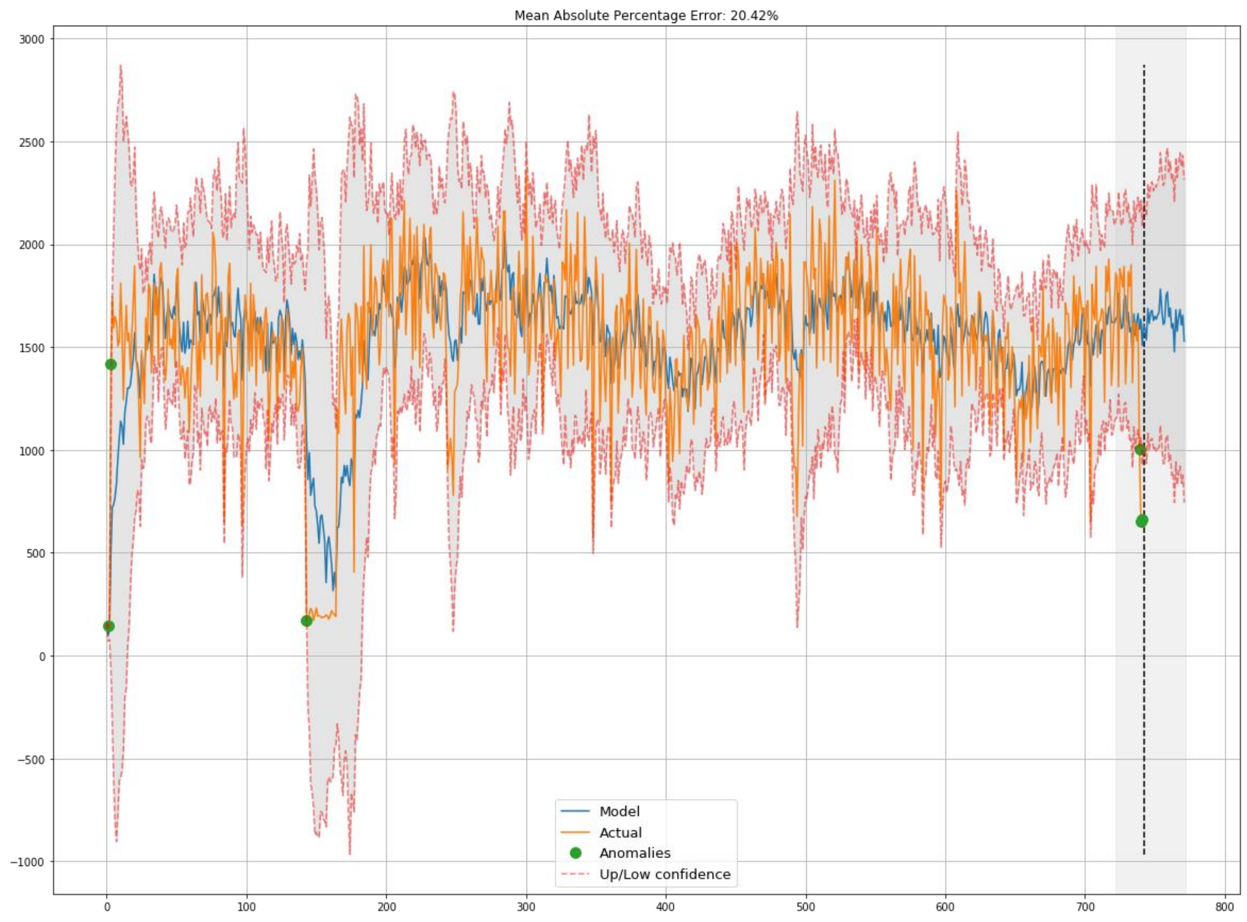
You may be asking how to do cross-validation for time series because time series have this temporal structure and one cannot randomly mix values in a fold while preserving this structure. With randomization, all time dependencies between observations will be lost. This is why we will have to use a trickier approach in optimizing the model parameters.

The idea is rather simple -- we train our model on a small segment of the time series from the beginning until some t, make predictions for the next $t+n$ steps, and calculate an error. Then, we expand our training sample to $t+n$ value, make predictions from $t+n$ until $t+2*n$, and continue moving our test segment of the time series until we hit the last available observation. As a result, we have as many folds as n will fit between the initial training sample and the last observation.



Now, knowing how to set up cross-validation, we can find the optimal parameters for the Holt-Winters model. Recall that we have monthly seasonality, hence the slen=30 parameter.

In the Holt-Winters model, as well as in the other models of exponential smoothing, there's a constraint on how large the smoothing parameters can be, each of them ranging from 0 to 1. Therefore, in order to minimize our loss function, we have to choose an algorithm that supports constraints on model parameters. In our case, we will use the truncated Newton conjugate gradient.



If we look at the model deviations, we can clearly see that the model reacts quite sharply to changes in the structure of the series but then quickly returns the deviation to the normal values, essentially "forgetting" the past.

5.2.2 Stationarity, unit root

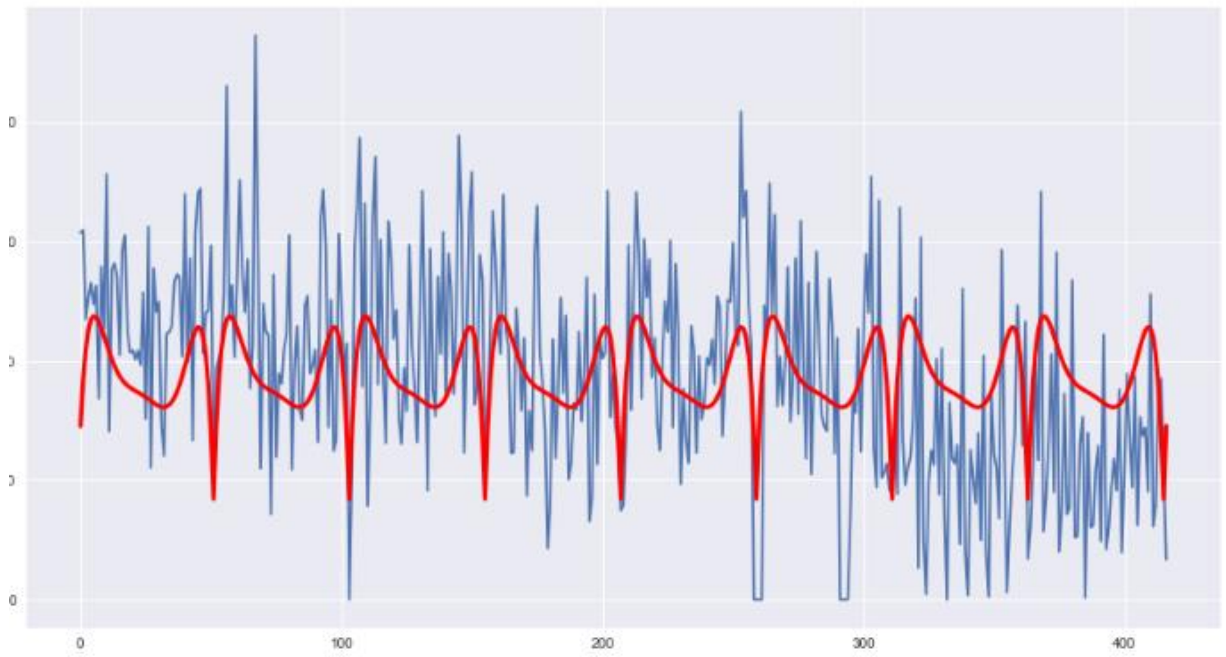
Before we start modeling, we should mention such an important property of time series: **stationarity**

If a process is stationary, that means it does not change its statistical properties over time, namely its mean and variance. (The constancy of variance is called **homoscedasticity**)

The covariance function does not depend on time; it should only depend on the distance between observations.

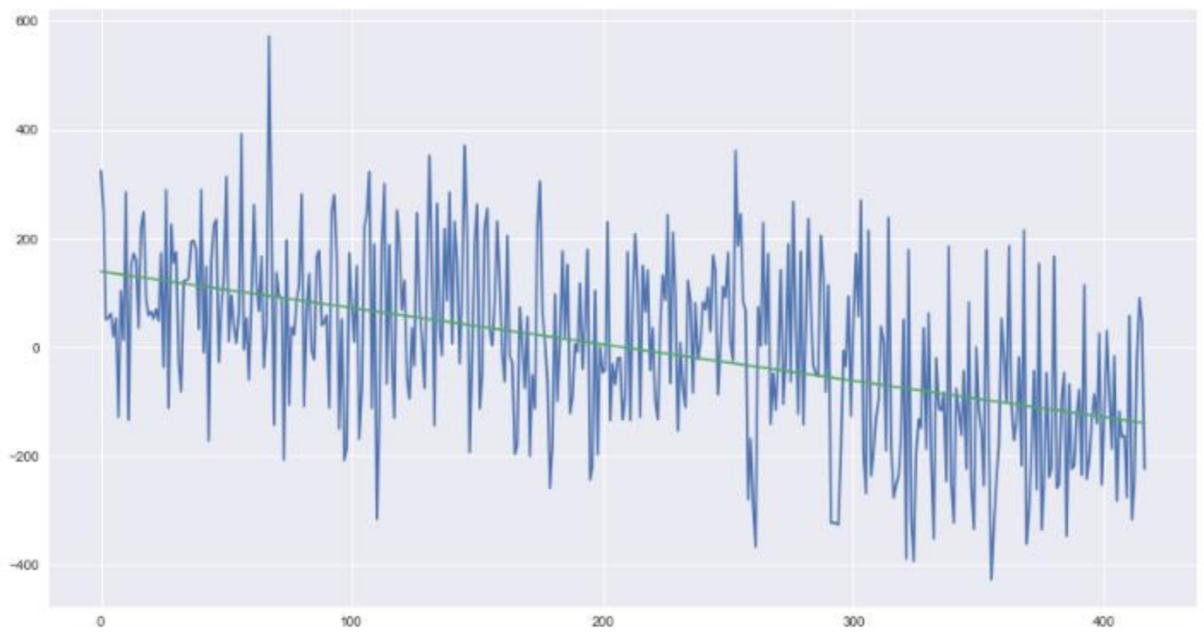
Why is stationarity so important? Because it is easy to make predictions on a stationary series since we can assume that the future statistical properties will not be different from those currently observed. Most of the time-series models, in one way or the other, try to predict those properties (mean or variance, for example). Future predictions would be wrong if the original series were not stationary. Unfortunately, most of the time series that we see outside of textbooks are non-stationary, but we can (and should) change this.

So, in order to combat non-stationarity, we have to eliminate seasonality and trends of our time series. We can model the seasonal component directly, then subtract it from the observations. The seasonal component in a given time series is a sine wave over a generally fixed period and amplitude. This can be approximated easily using a curve-fitting method. The curve can then be used as a new input for modeling with supervised learning algorithms, or subtracted from observations to create a seasonally adjusted series. Let's start off by fitting a curve to dataset. The NumPy library provides the polyfit() function that can be used to fit a polynomial of a chosen order to a dataset. Ideally, we want the simplest curve that describes the seasonality of the dataset. For consistent sine wave-like seasonality, a 5th order or 6th order polynomial will be sufficient.



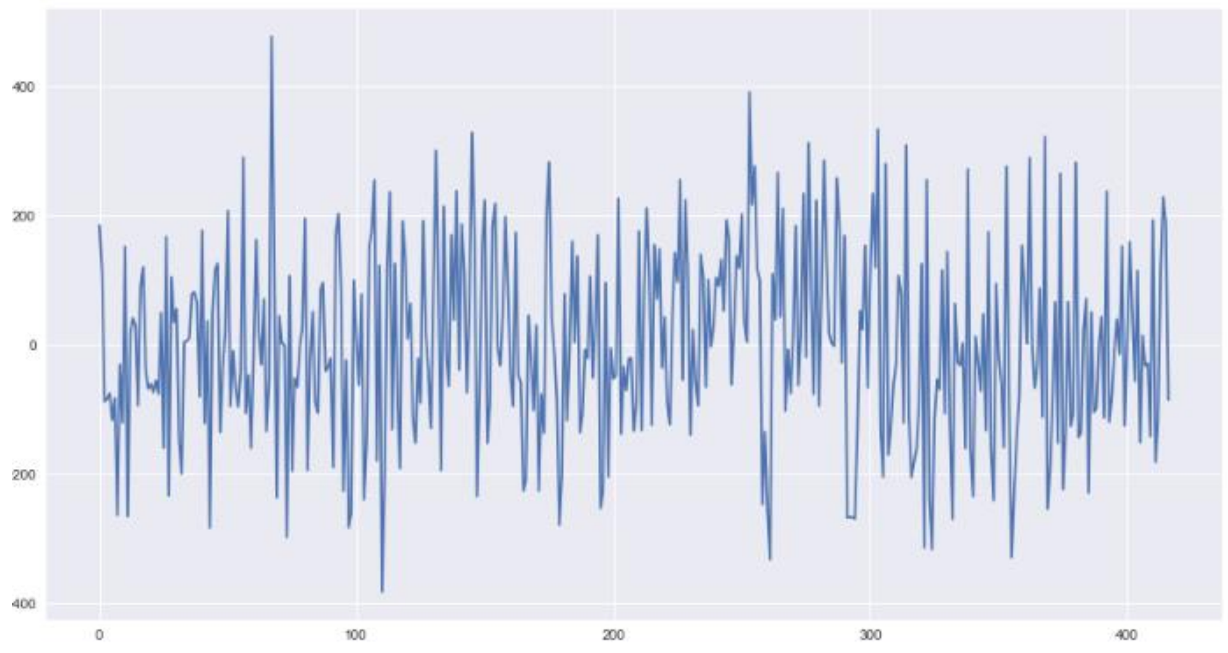
Curve-fitting method

After that, we detrended our time series. We can clearly see that the trend is linear so used a linear model "linear Regression" to detrend our time series. The linear model is fitting to the integer-indexed observations then we plot the trend line (green) over the original dataset (blue).



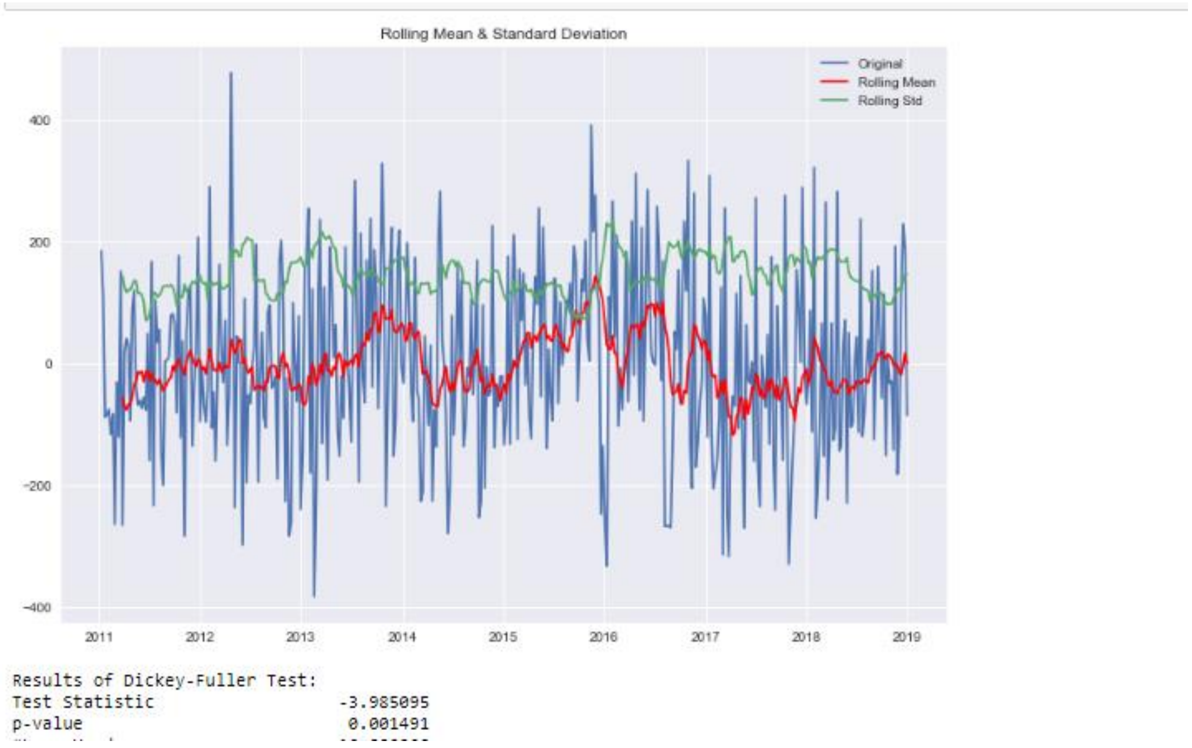
Revenue dataset plot with trend

Next, the trend is subtracted from the original dataset and the resulting detrended dataset is plotted.



Revenue dataset plot without trend

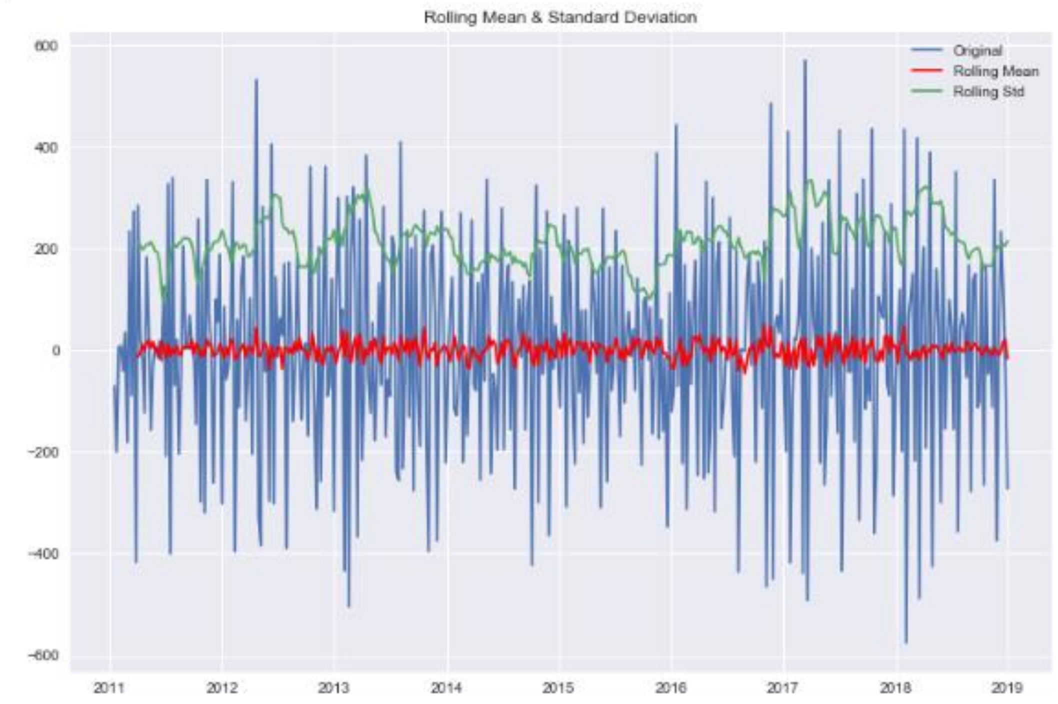
To verify the stationarity of our time series we used **Dickey-Fuller test.** The main idea behind the Dickey-Fuller test for stationarity of time series (testing the presence of a unit root). The null hypothesis of the test is that the time series is non-stationary, which was rejected after eliminate the seasonality and trend of the time series.



```
Results of Dickey-Fuller Test:
Test Statistic                -3.985095
p-value                        0.001491
```

Result of Dickey-Fuller test

We can see that p-value <= 0.5 but we can ameliorate the results by differencing our time serie. Specifically, a new series is constructed where the value at the current time step is calculated as the difference between the original observation and the observation at the previous time step. This has the effect of removing a trend from a time series dataset. The Dickey-Fuller confirm that our time series was improved.



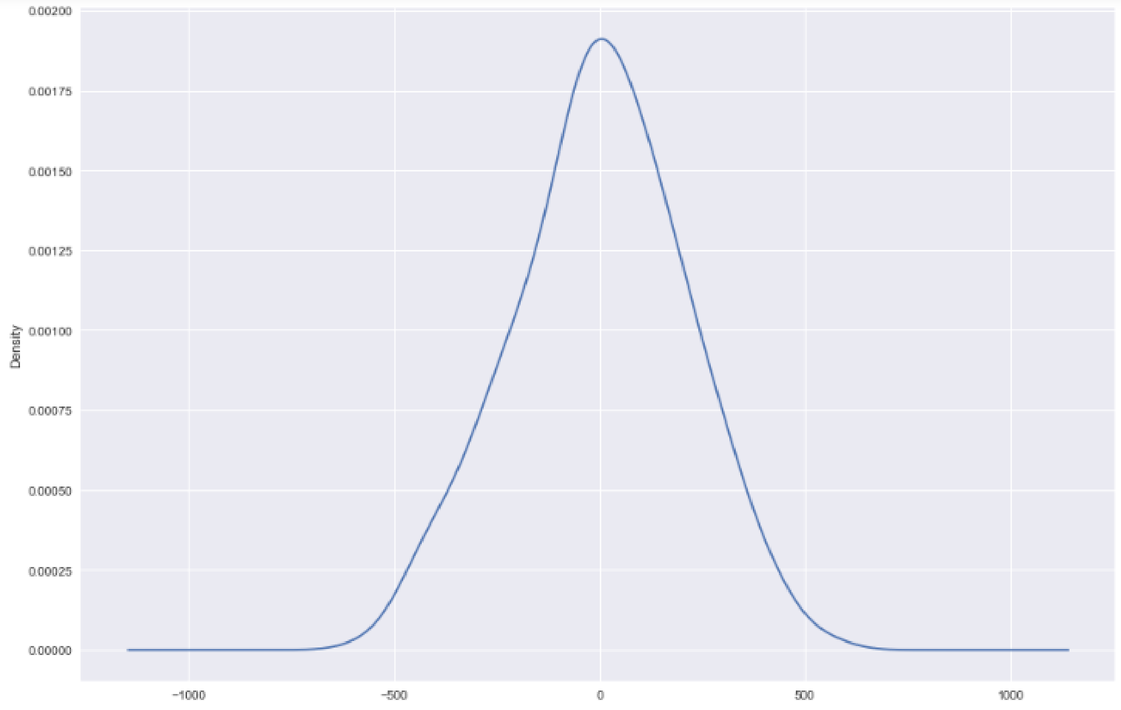
```
Results of Dickey-Fuller Test:
Test Statistic                -8.252396e+00
p-value                        5.347397e-13
```

Result of Dickey-Fuller test after differencing

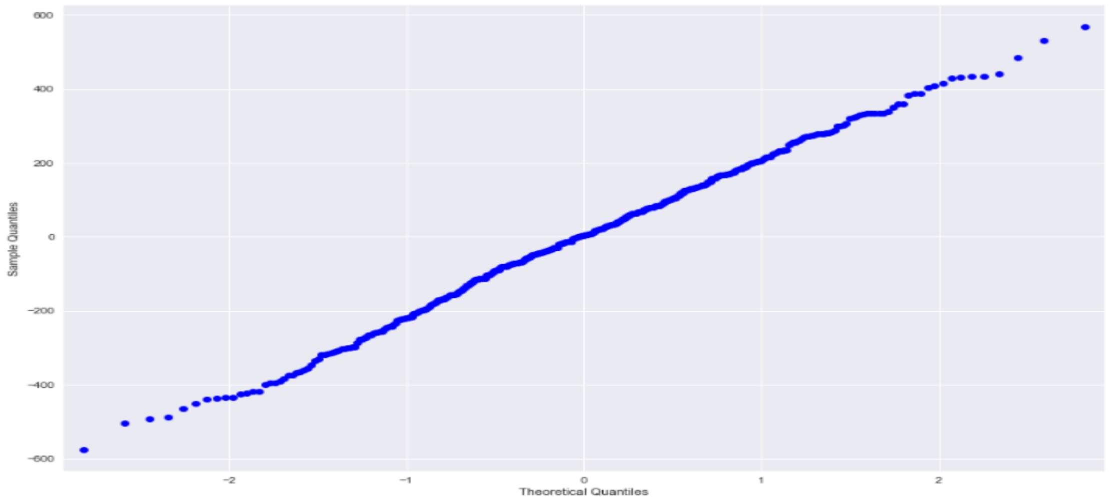
We notice that the p-value decreased to 5.3e-13 which is a very satisfied result.

5.2.3 Verification of the normal distribution and QQ PLOT:

Another important visualization is of the distribution of observations themselves. This means a plot of the values without the temporal ordering. Some linear time series forecasting methods assume a well-behaved distribution of observations (i.e. a bell curve or normal distribution). This can be explicitly checked using tools like statistical hypothesis tests. But plots can provide a useful first check of the distribution of observations both on raw observations and after any type of data transform has been performed.



We notice that the distribution follows a gaussian form but the QQ plot is a much better visualization of our data, providing us with more certainty about the normality. From QQ plot we can be more assured our data is normal,



Perfect! Our series now looks like something indescribable, oscillating around zero. The Dickey-Fuller test indicates that it is stationary, and the distribution is normal. We can finally start modeling!

5.2.4 SARIMA intuition and model building

We will explain this model by building up letter by letter. $SARIMA(p,d,q)(P,D,Q,s)$, Seasonal Autoregression Moving Average model:

$AR(p)$ - autoregression model i.e. regression of the time series onto itself. The basic assumption is that the current series values depend on its previous values with some lag (or several lags). The maximum lag in the model is referred to as p . To determine the initial p , you need to look at the PACF plot and find the biggest significant lag after which most other lags become insignificant.

$MA(q)$ - moving average model. Without going into too much detail, this models the error of the time series, again with the assumption that the current error depends on the previous with some lag, which is referred to as q . The initial value can be found on the ACF plot with the same logic as before.

Let's combine our first 4 letters:

$AR(p)+MA(q)=ARMA(p,q)$

What we have here is the Autoregressive–moving-average model! If the series is stationary, it can be approximated with these 4 letters. Let's continue.

$I(d)$ - order of integration. This is simply the number of nonseasonal differences needed to make the series stationary. In our case, it's just 1 because we used first differences.

Adding this letter to the four gives us the $ARIMA$ model which can handle non-stationary data with the help of nonseasonal differences. Great, one more letter to go!

$S(s)$ - this is responsible for seasonality and equals the season period length of the series

With this, we have three parameters: (P,D,Q)

P - order of autoregression for the seasonal component of the model, which can be derived from PACF. But you need to look at the number of significant lags, which are the multiples of the season period length. For example, if the period equals 24 and we see the 24-th and 48-th lags are significant in the PACF, that means the initial P should be 2.

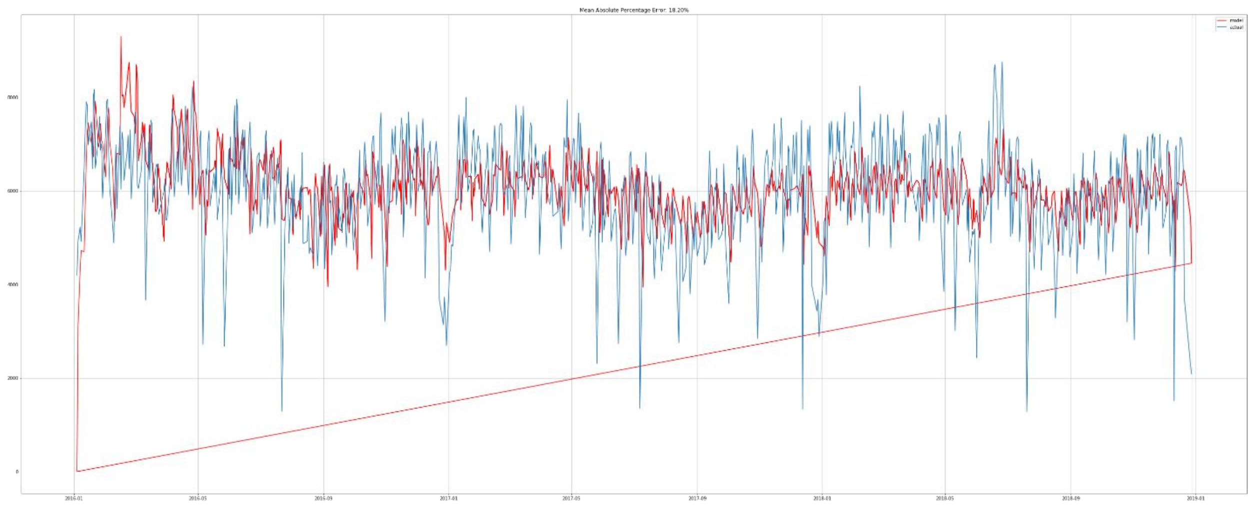
Q - similar logic using the ACF plot instead.

D - order of seasonal integration. This can be equal to 1 or 0, depending on whether seasonal differences were applied or not.

Now that we know how to set the initial parameters, let's have a look at the final plot once again and set the parameters:

- pp is most probably 2 since it is the last significant lag on the PACF, after which, most others are not significant.

- dd equals 1 because we had first differences
- qq should be somewhere around 2 as well as seen on the ACF
- PP might be 2, since 30-th and 60-th lags are somewhat significant on the PACF
- DD again equals 1 because we performed seasonal differentiation
- QQ is probably 1. The 30-th lag on ACF is significant while the 60-th is not.



In the end, we got very adequate predictions. Our model was wrong by 19.38% on average, which is not bad at all. However, the overall costs of preparing data, making the serie stationary, and selecting parameters might not be worth this accuracy.

5.3 XgBoost

5.3.1 Introduction¶

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Jerome Friedman (24 Feb 1999) .

XGBoost is an implementation of gradient boosted decision trees (also known as GBDT, GBM) designed for speed and performance. It is used for supervised learning problems, where we use the training data (with multiple features) *xi* to predict a target variable *yi*.
But how well does XGBoost perform when used to predict future values of a time-series?

5.3.2 Ensemble learning

XGBoost is ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

Let’s briefly discuss bagging before taking a more detailed look at the concept of boosting.

5.3.2.1 Bagging

While decision trees are one of the most easily interpretable models, they exhibit highly variable behavior. Consider a single training dataset that we randomly split into two parts. Now, let’s use each part to train a decision tree in order to obtain two models.

When we fit both these models, they would yield different results. Decision trees are said to be associated with high variance due to this behavior. Bagging or boosting aggregation helps to reduce the variance in any learner. Several decision trees which are generated in parallel, form the base learners of bagging technique. Data sampled with replacement is fed to these learners for training. The final prediction is the averaged output from all the learners.

5.3.2.2 Boosting

In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.

The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Such small trees, which are not very deep, are highly interpretable. Parameters like the number of trees or iterations, the rate at which the gradient boosting learns, and the depth of the tree, could be optimally selected through validation techniques like k-fold cross validation. Having a large number of trees might lead to overfitting. So, it is necessary to carefully choose the stopping criteria for boosting.

Concept

- An initial model F0 is defined to predict the target variable y. This model will be associated with a residual (y – F0)
- A new model h1 is fit to the residuals from the previous step
- Now, F0 and h1 are combined to give F1, the boosted version of F0. The mean squared error from F1 will be lower than that from F0
- To improve the performance of F1, we could model after the residuals of F1 and create a new model F2.
- This can be done for ‘n’ iterations, until residuals have been minimized as much as possible

5.3.3 How Gradient Boosting works

Gradient boosting involves three elements:

- A loss function to be optimized.
- A weak learner to make predictions.
- An additive model to add weak learners to minimize the loss function.

5.3.3.1 Loss Function

The loss function used depends on the type of problem being solved.

It must be differentiable, but many standard loss functions are supported and you can define your own.

For example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

5.3.3.2 Weak Learner

Decision trees are used as the weak learner in gradient boosting.

Specifically, regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.

This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

5.3.3.3 Additive Model

Trees are added one at a time, and existing trees in the model are not changed.

A gradient descent procedure is used to minimize the loss when adding trees.

Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss). Generally, this approach is called functional gradient descent or gradient descent with functions.

5.4 LSTM

Unlike other machine learning algorithms, long short-term memory recurrent neural networks are capable of automatically learning features from sequence data, support multiple-variate data, and can output a variable length sequences that can be used for multi-step forecasting.

Recurrent neural networks, or RNNs, are specifically designed to work, learn, and predict sequence data. A recurrent neural network is a neural network where the output of the network from one-time step is provided as an input in the subsequent time step. This allows the model to make a decision as to what to predict based on both the input for the current time step and direct knowledge of what was output in the prior time step.

Perhaps the most successful and widely used RNN is the long short-term memory network, or LSTM for short. It is successful because it overcomes the challenges involved in training a recurrent neural network, resulting in stable models. In addition to harnessing the recurrent connection of the outputs from the prior time step, LSTMs also have an internal memory that operates like a local variable, allowing them to accumulate state over the input sequence.

LSTMs offer a number of benefits when it comes to multi-step time series forecasting; they are:

- Native Support for Sequences. LSTMs are a type of recurrent network, and as such are designed to take sequence data as input, unlike other models where lag observations must be presented as input features.
- Multivariate Inputs. LSTMs directly support multiple parallel input sequences for multivariate inputs, unlike other models where multivariate inputs are presented in a flat structure.
- Vector Output. Like other neural networks, LSTMs are able to map input data directly to an output vector that may represent multiple output time steps.

We start off by developing a simple LSTM model that reads in a sequence of days of total daily amount of item and predicts a vector output of the next standard week of daily amount of item

For that, we then need to iterate over the time steps and divide the data into overlapping windows; each iteration moves along one-time step and predicts the subsequent seven days.

Now, we can define and fit the LSTM model on the training data.

This multi-step time series forecasting problem is an autoregression. That means it is likely best modeled where that the next seven days is some function of observations at prior time steps. This and the relatively small amount of data means that a small model is required.

We develop a model with a single hidden LSTM layer with 200 units. The number of units in the hidden layer is unrelated to the number of time steps in the input sequences. The LSTM layer is followed by a fully connected layer with 200 nodes that will interpret the features learned by the LSTM layer.

We use the efficient Adam implementation of stochastic gradient descent and fit the model for 70 epochs with a batch size of 16.
The small batch size and the stochastic nature of the algorithm means that the same model will learn a slightly different mapping of inputs to outputs each time it is trained. This means results may vary when the model is evaluated.

In order to predict the next standard week, we need to retrieve the last days of observations. As with the training data, we must first flatten the history data to remove the weekly structure so that we end up with eight parallel time series.

Next, we retrieve the last seven days of daily items consumed. We parameterize this as we did for the training data so that the number of prior days used as input by the model can be modified in the future.

We then make a prediction using the fit model and the input data and retrieve the vector of seven days of output.

Finally, an output layer will directly predict a vector with seven elements, one for each day in the output sequence.

We use the mean squared error loss function as it is a good match for our chosen error metric of RMSE.

5.5 Best model selection

Model	**R^2**
SARIMA	0.81 – 0.86
XGBoost	0.79 - 0.93
LSTM	0.63 – 0.8

CHAPTER SIX: DEPLOYMENT

6.1 Introduction

Now that we have selected the model from which we can get the best results, it is time to build a client application that exploits every result we reached so far in our predictions.

It is necessary for our application to have certain requirements in order to be built.

6.2 Application requirements

6.2.1 Access and navigability

The system must be easy to use by both managers and chefs such that they do not need to read an extensive number of manuals.

The system must be quickly accessible by both managers and chefs.

The system must be intuitive and simple in the way it displays all items the restaurant has to offer.

The menus of the system must be easily navigable by the users with buttons that are easy to understand.

The system must provide a password enabled login to the user to avoid any foreign entity changing the data in the system. The system should provide the user updates on completion of requested processes and if the requested processes fail, it should provide the user the reason for the failure.

The system should not update the data in any database for any failed processes.

6.2.2 Performance

The system must not lag.

The calculations performed by the system must comply according to the norms set by the user and should not vary unless explicitly changed by the user.

6.2.3 Supportability

The software is designed such that it works even on systems having the minimum configuration. The system is adaptable even if additional plugins or modules are added at a later point.

The data can be exported to the manager so as to make the system more portable.

6.2.4 Interfacing

The system must offer an easy and simple way of viewing current and future predictions by selecting the dates and the items or the revenue.

The system must be able to display all menu items with their predicted count values unless the user wants to only view the top ones.

6.3 Application tools

In order to respond to our application requirements, we went through searching for the best way to implement both our work on predictions and client's needs for a more adaptive tool.

In the sections below we will discuss what we have chosen and why.

6.3.1 Python Web Frameworks

The web frameworks were chosen from Web Frameworks

for Python. There are two sets of Python web frameworks, full-stack and non-full-stack frameworks, one of each was chosen. A full-stack framework is a framework that contains all you need to develop a web application. A non-full-stack

framework is a framework which does not contain all packages to develop a complete web application. Choosing either of these two categories is a matter of taste,

full-stack frameworks are ready out of the box while non full-stack needs additional

packages, but non-full-stack frameworks have the advantage that the developer

can choose his preferred packages for the job. First the Django web framework

was chosen, which is a full-stack framework. Django was chosen because it is one

of the most popular web frameworks. The second choice was the Flask micro web

framework, one of the most popular non-full-stack frameworks

6.3.2 Flask

Flask is a micro web framework. Flask is highly customizable as you can choose

your own architecture of your web application. Also, it is possible for instance to

make your own form validation or use a form validation package that one wants.

Flask comes with the following features:

• Development server

• Unit test support

• REST support

The power of Flask is that the developer is able to customize everything.

Why Flask?

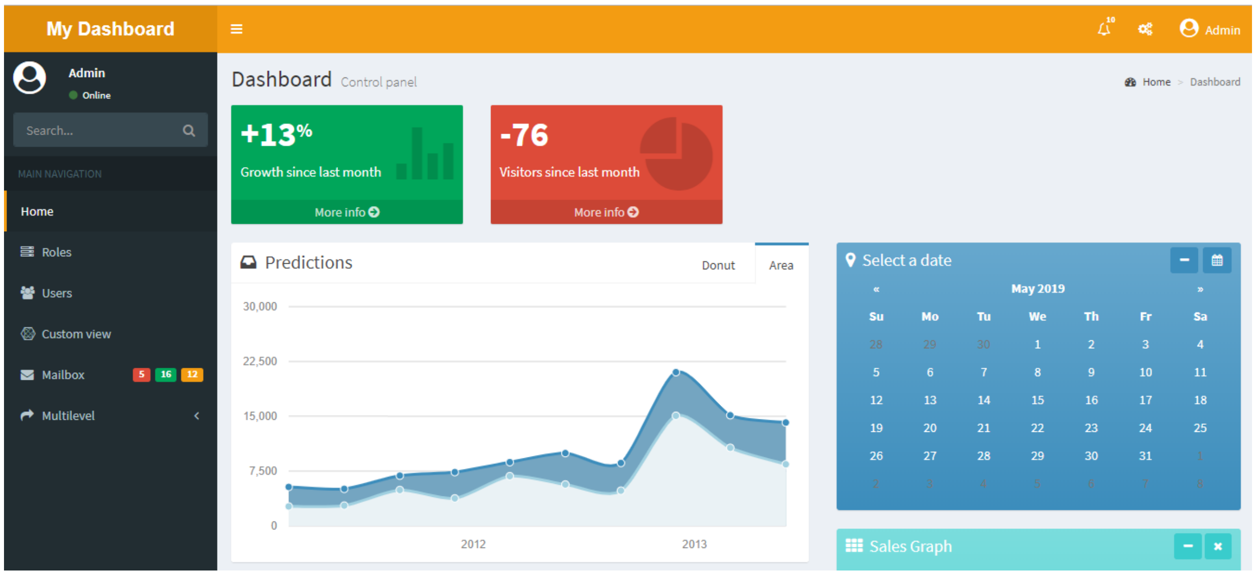
This section argues why the Flask web framework was chosen for this project.

The two key factors were its simplicity and our previous experience with the framework. These factors will be described shortly in the following.

Simplicity: The simplicity of Flask enables us to focus on the important task at hand and not on how to develop a web application. This means rapid development of web applications used as examples and for testing the tool during the project.

Previous experience: The project group has previous experience using Flask. This again, means that we can develop examples faster, without having to spend time getting acquainted with the framework.

6.3.3 Results



This interface will be accessible by the manager, assistants or the chefs of the restaurant.

Clear, easy to use, handy for any level of users and understandable for business handlers.

The user can select a date from the date picker in blue, the page will automatically reloads giving us results of the revenue and items for the date selected, the dashboard also informs us with statistical results like growth and number of visitors since last month.

Such application is accessible wherever the user is, since it is a web application. The product does not have to be installed nor downloaded from any resources.

CHAPTER SEVEN: Conclusions & Future Work

7.1 Conclusion

Successful management of inventory levels is one of the key factors in the financial success of every manufacturer, wholesaler or retailer. As the primary component of working capital for most of these businesses, inventory affects the flow of money through the business as much as it does the flow of materials and merchandise through the company's operations. Predictive inventory management helps ensure that the business can meet the ever-changing needs of a diverse customer base while minimizing costs and end-of-life discounting, and maximizing profitability. An optimum level of inventory, which enables the business to free up working capital, can give a business the flexibility to take advantage of capital-intensive opportunities when they present themselves.

Our AI-powered application for predictive inventory management can help companies determine the correct inventory levels for products and materials at virtually all stages along the supply chain. Our solution can help restaurants or canteens balance the trade-offs between carrying costs and customer service levels—and between production volume and capital reserves.

7.2 Future Work

As with all business processes, the search for improvement in inventory management never ends. But predictive analytics has been shown to help forward-looking businesses gain an edge that can keep them one step ahead of the competition.

We propose a very thorough application that handles predictions and online restaurant management such as purchasing ingredients online, stock survey programs, suggesting best suppliers, theft detection by employees.….

With this step we would definitely revolutionize the gastronomy sector as it will be the best solution in the market today.