Final Project
Taxis RCP S.A.S.

Documentation

Team #48

Jeferson Andrés Castaño Tafur
Juan Felipe Chaves
Sandra Guerrero
Cristian Danilo Romero Orjuela
Rodrigo Alejandro Quintana Romero

November 2020

Contents

1. Introduction

Business Problem

Taxi services are an essential part of urban transportation. This service is distinguished by its reliability and flexibility due to its 24/7 availability, point-to-point service, exhaustive coverage, and security perception. The growth of urban areas has made taxi services become increasingly more popular. It is estimated that 47 million trips are completed worldwide every day[1] (that is all Colombia travelling in taxi on a daily basis). Rough estimates of the market size of the taxi industry are around USD 32.5 billion yearly revenue in 2020 in the United States alone[2]. Other massive markets include China, London, and Tokyo.

Nevertheless, traditional taxi systems suffer from inefficiencies due to unbalanced supply-demand dynamics. Passengers experience long waiting times for a taxi to be available, affecting the perceived quality of the service and the customer's satisfaction. Taxi drivers roam vacant for extended periods of time, resulting in increased costs and contributing even more to environment pollution.

A supply-demand matching approach should seek for actions to coordinate taxi supply according with expected demand, given that taxi companies and regulators are only able to control supply. A thorough understanding of demand dynamics is crucial, and accurate immediate demand prediction can help taxi companies to allocate their fleet in a more efficient way. There are, however, major challenges to understand taxi demand, including:

- Human behavior
- Complex spatial-temporal structure
- Changeful real-time dynamics
- Exogenous predictable variables (e.g. socioeconomics, demographics)
- Exogenous unpredictable variables (e.g. traffic, weather)
- Substitute services (e.g. ride sharing like Uber, scooter and electric bike rentals)

In the particular business case studied in this project, where we evaluate taxi requests and cancelled requests, more than 60% of cancelled requests are label to the reason "there were no available vehicles to attend the request", followed by more than 25% of "I do not need the service anymore".

In the era of big data, large amounts of information can be collected in real-time from mobile devices, either from passengers or from drivers, including pick-up location, drop-off location, date-time, number of passengers, weather data, traffic data, vehicle's data, and passenger's personal data. This information provides the data necessary to create a smart transportation system that can coordinate taxi supply at a large-scale and in real-time through policies like dynamic pricing incentives for drivers.

Traditional forecasting methods, like time series analysis, are not suited for heterogenous massive data (e.g. traffic data) and at best can predict demand on a daily basis, whereas modern machine learning techniques, like deep learning, are able to exploit this data for the means of traffic and demand prediction in real-time.

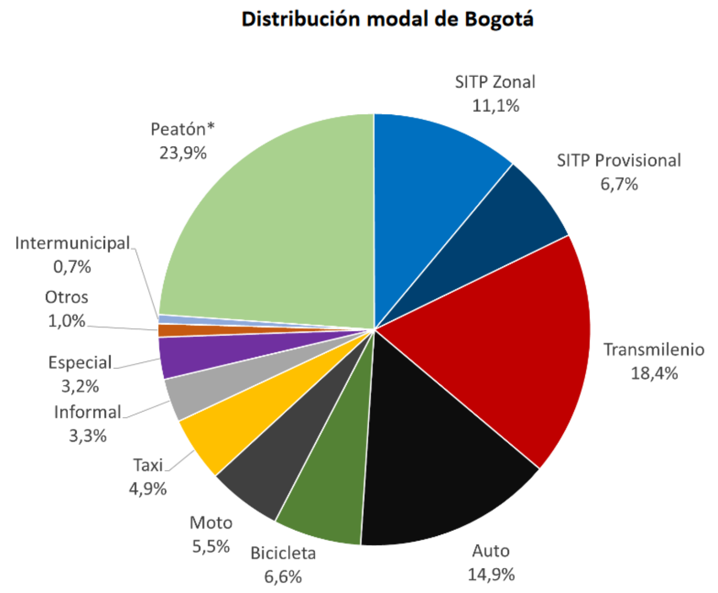
[1] https://infinitecab.com/
[2] https://www.ibisworld.com/united-states/market-research-reports/taxi-limousine-services-industry/

Some of the current techniques applied recently to this problem are the following. Chen et. al. (2020) designed a framework using taxi and Uber demand data to build two deep models, containing a specific feature fusion method, to predict demand in New York City. Kalacou et. al. presents an exploratory taxi fleet service analysis and compares two forecast models, ARIMA and Artificial Neural Network (ANN), aimed at predicting the spatiotemporal variation of short-term taxi demand in Lisbon. Khryashchev et. al. tested a predictive algorithm such as LSTM (deep learning) versus an ARIMA (time series) to predict taxi and Uber demand in New York City. Chen et. al. proposed three methods, random forest model (RFM), ridge regression model (RRM), and combination forecasting model (CFM), to predict the taxi demand in hotspots. Sun et. al. proposes a multi-task learning model containing three parallel LSTM layers to co-predict taxi pick-up and drop-off demands. Wu et. al. utilizes a joint guidance residual network JG-Net for city-scale taxi demand prediction. Che et. al. study the taxi travel demand by constructing dynamic networks based on taxi trajectory data.

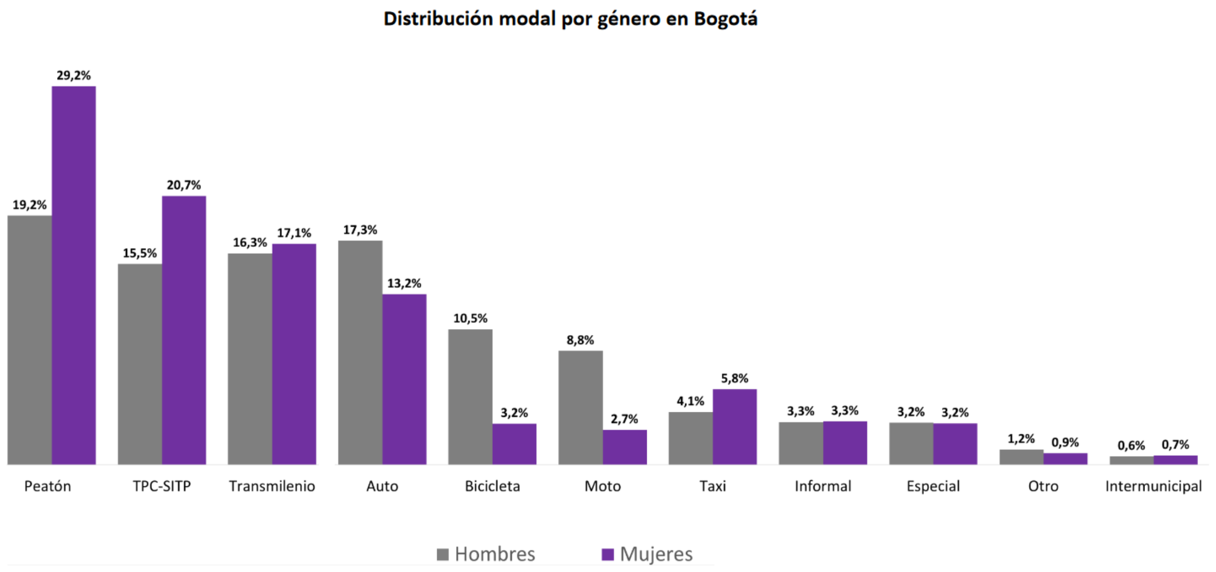
Informed driving is increasingly becoming a key feature for sustainability of transportation systems. The data deluge is providing new opportunities for discovering knowledge and real-time decision making. Intelligent transportation systems for taxi supply is the future for this industry.

Business Context

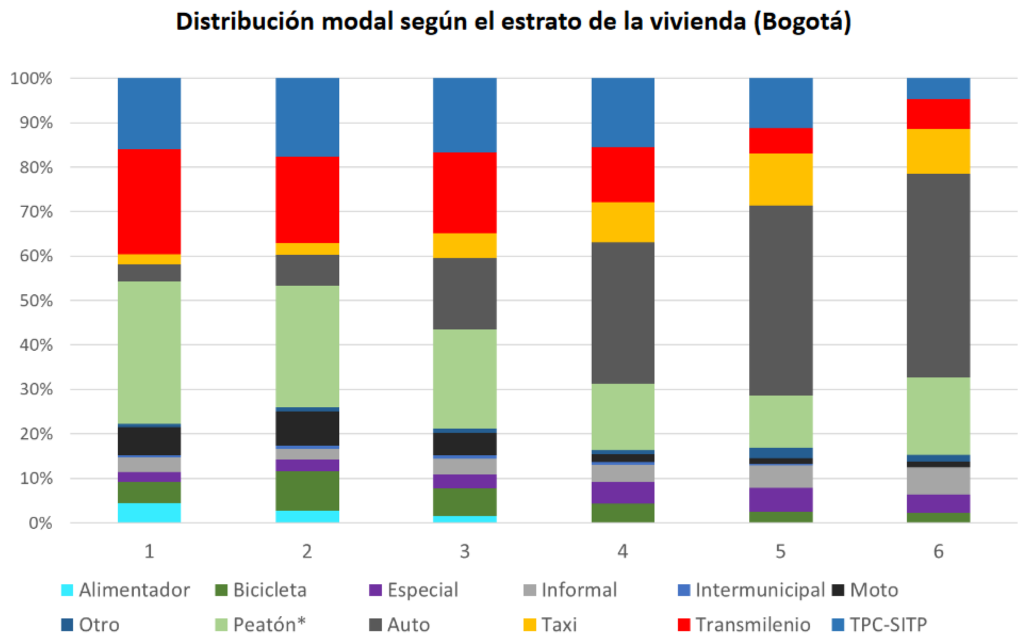
In Colombia, the most complete and updated information about urban transportation comes from a survey conducted by Bogota's city hall in 2019. Although it provides only local data, it could still be indicative of trends and patterns.



Around 4.9% of commutes is made in taxi. Out of 13,359,728 daily trips made by Bogota 's metropolitan area habitants, this is equivalent to the whole population of Soacha commuting in taxi.



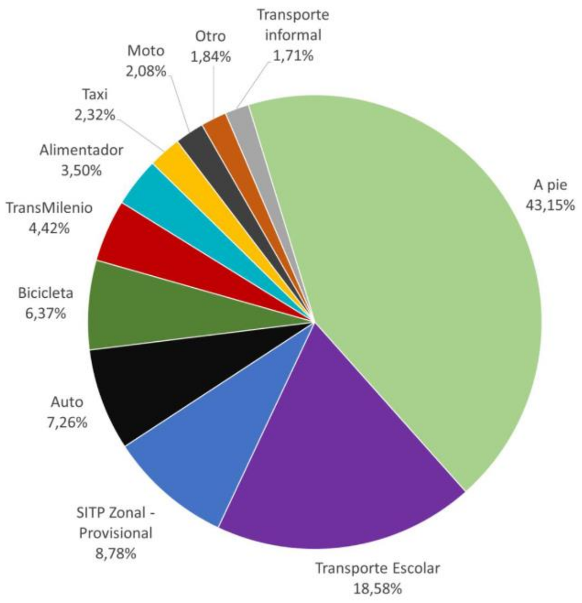
There is a difference, however, between women, accounting for 5.8% of their trips in taxi, and men, where only 4.1% of their trips are made in taxi.

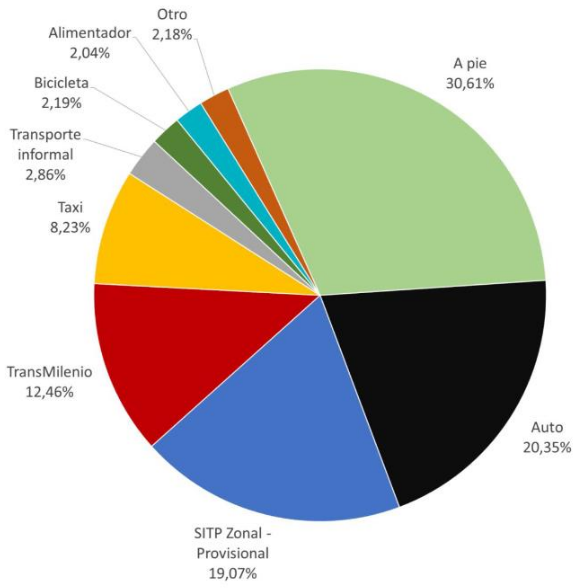


Stratums #5, #6, and #4 made the most use of taxi services (in percentage terms).

Niños, niñas y adolescentes

Personas mayores

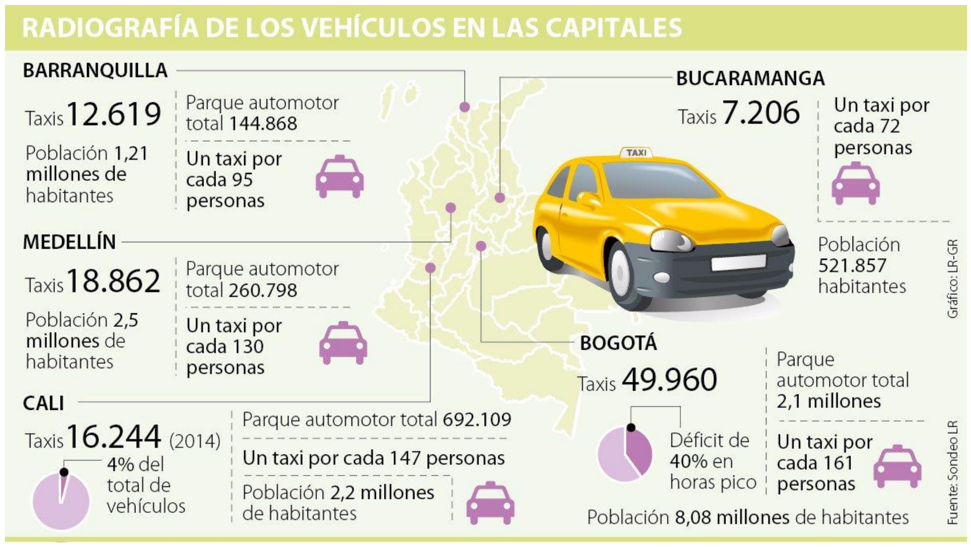




Underage people tend to use taxi less, only 2.32%, than elder people, up to 8.23%.

Overall, this simple overview highlights stylized fact that describe taxi demand in Bogota. In particular, how it is especially demanded by women over men, elders over youngsters, and high stratum people vs low stratum.

Additional information of the taxi market in Colombia is obtained from journal La Republica.


RADIOGRAFÍA DE LOS VEHÍCULOS EN LAS CAPITALES

BARRANQUILLA
Taxis 12.619
Población 1,21 millones de habitantes
Parque automotor total 144.868
Un taxi por cada 95 personas

MEDELLÍN
Taxis 18.862
Población 2,5 millones de habitantes
Parque automotor total 260.798
Un taxi por cada 130 personas

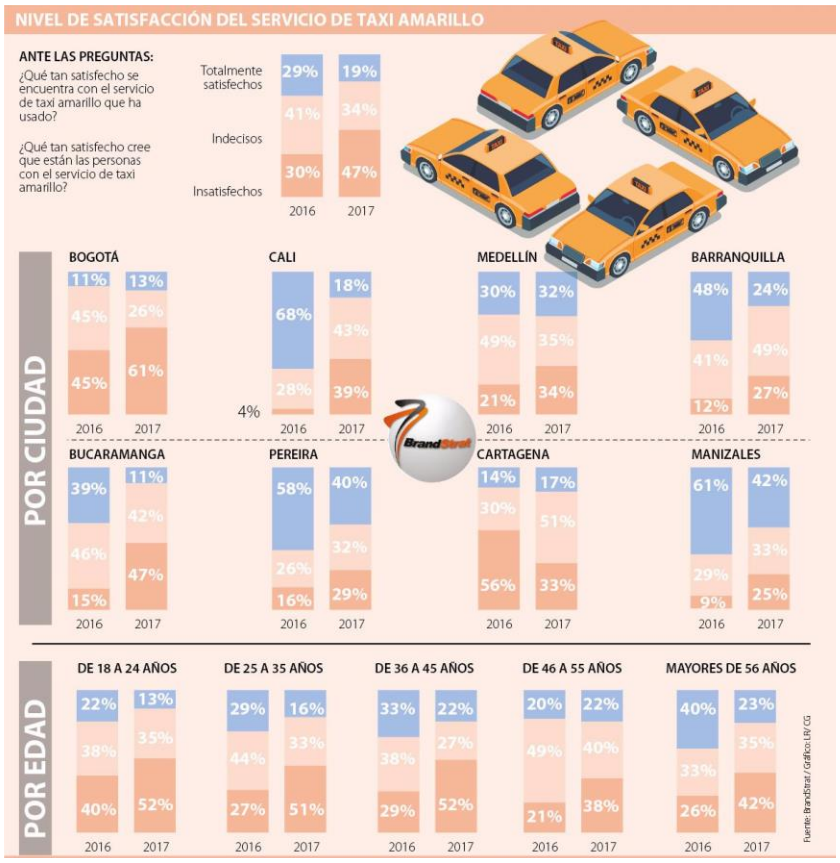
CALI
Taxis 16.244 (2014)
4% del total de vehículos
Parque automotor total 692.109
Un taxi por cada 147 personas
Población 2,2 millones de habitantes

BUCARAMANGA
Taxis 7.206
Un taxi por cada 72 personas
Población 521.857 habitantes

BOGOTÁ
Taxis 49.960
Déficit de 40% en horas pico
Parque automotor total 2,1 millones
Un taxi por cada 161 personas
Población 8,08 millones de habitantes

Gráfico: LR-GR
Fuente: Sondeo LR

On the supply side, major cities have between 72-161 people per taxi as of 2017.



General unsatisfaction with taxi services is high among major cities, but smaller in minor ones.

Predicting Taxicab requests using 2.4 million requests.

Thanks to Taxis Rcp Pasto. for data about these taxicab requests. We endeavored to delve into this gold mine using 3.5 years of Pasto taxi trip data - around 2.4 million records - from February 2017 to August 2020.

This project's primary objective is to predict the taxicab requests throughout Taxis Rcp Pasto as it changes from day to day and hour to hour. So, given a specific date and time, can we predict the number of requests to a reasonably high accuracy? A secondary objective is also to get a customer segmentation to identify potential customers for marketing campaigns.

2. Data Exploration

In general, the dataset provided by the company has 27 tables organized in CSV files. Some tables have relevant information about request volume, and other tables contain complementary information about the company operation.

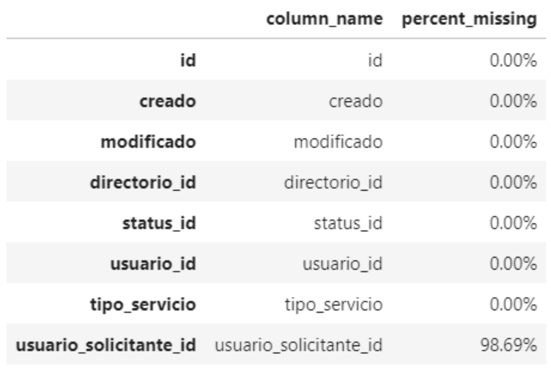
The following is the description of some useful tables:

- **File 1: "mastermind_solicitud.csv"** It offers detailed information about more than 2.4M taxicab service requests.
  This table contains the most crucial information about the demand for taxicab services via requests. It identifies all requests made to the company, including request Date Time, address, service type, and client (if available). With this data, demand behavior in the past can be analyzed and modeled to produce forecasts.
- **File 2: "mastermind_solicitudcancelada.csv"** This table contains useful information about the cancelation of requests. It identifies all cancelations made to the company, including some details like request identification and motive for cancellation. With this data, it could be possible to analyze the potential harmful factors that impact the client's perception of the company and ultimately affect the demand for their services, such as delays in the car's arrival or driver's unavailability to cover up the requested service.
- **File 3: "administracion_barrio.csv"** Contains neighborhood names for the city of Pasto.
- **File 4: "administración_direccion.csv"** This table contains all the addresses to which have requested a radio taxi service. The information about the address and neighborhood of the request can be useful for geolocation analysis. It contains latitude and longitude information. However, this data is not reliable, so it is suggested not to use these columns.
- **File 5: "adminsitracion_directorio.csv"** This table contains information on the people who have requested a taxi service from the company. This information can be useful to create profiles of users and link their requests.
- **File 6: "administracion_afiliacion.csv"** This table contains information on taxi drivers' affiliation with the company. It can be useful to check the number of taxi drivers affiliated with the company.

Identifying mistakes and missing data.

We will inspect the missing values of the tables.

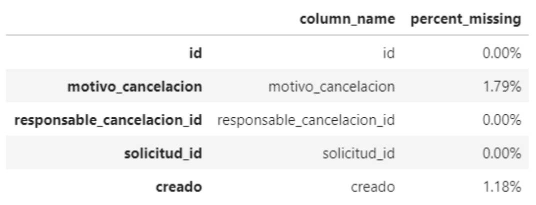
➢ Administración Solicitud



	column_name	percent_missing
id	id	0.00%
creado	creado	0.00%
modificado	modificado	0.00%
directorio_id	directorio_id	0.00%
status_id	status_id	0.00%
usuario_id	usuario_id	0.00%
tipo_servicio	tipo_servicio	0.00%
usuario_solicitante_id	usuario_solicitante_id	98.69%

As we can see, “usuario_solicitante_id” has 98.7% missing values.
We can handle missing values by dropping the column.
However, the “usuario_solicitante_id” referencing the operator of the company, who records the request.

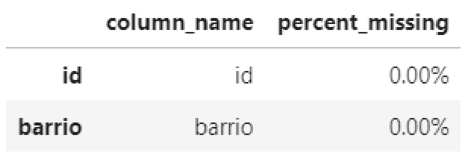
➢ Mastermind Solicitud Cancelada



	column_name	percent_missing
id	id	0.00%
motivo_cancelacion	motivo_cancelacion	1.79%
responsable_cancelacion_id	responsable_cancelacion_id	0.00%
solicitud_id	solicitud_id	0.00%
creado	creado	1.18%

This table does not have significant missing values. But we will remove records with "creado" missing.

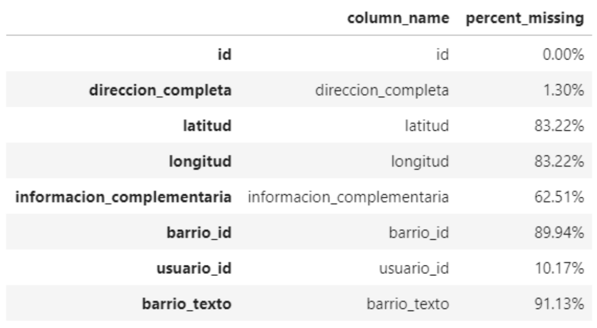
➢ Administración Barrio



	column_name	percent_missing
id	id	0.00%
barrio	barrio	0.00%

The "barrio" dataset does not have missing data.

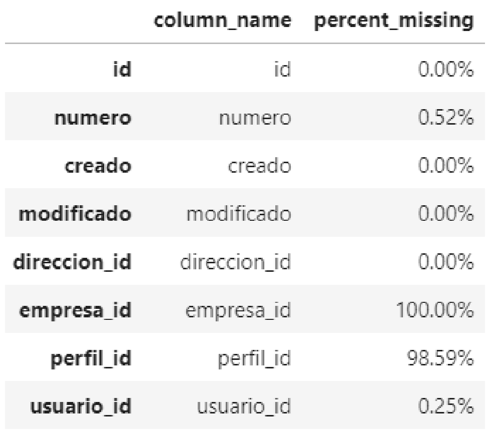
➢ Administración Dirección



	column_name	percent_missing
id	id	0.00%
direccion_completa	direccion_completa	1.30%
latitud	latitud	83.22%
longitud	longitud	83.22%
informacion_complementaria	informacion_complementaria	62.51%
barrio_id	barrio_id	89.94%
usuario_id	usuario_id	10.17%
barrio_texto	barrio_texto	91.13%

The “direccion_completa”,”informacion_complementaria” ,”barrio_id”, “usuario_id” and “barrio_texto” columns have 1.3%, 62.5%, 89.9%, 10.2% and 91.1% missind data, respectively.
There are a high number of missing values for the “barrio_id” column, so it is a challenge to have the address for each user, and do not count on the neighborhood associated.

- **Administración directorio**



	column_name	percent_missing
id	id	0.00%
numero	numero	0.52%
creado	creado	0.00%
modificado	modificado	0.00%
direccion_id	direccion_id	0.00%
empresa_id	empresa_id	100.00%
perfil_id	perfil_id	98.59%
usuario_id	usuario_id	0.25%

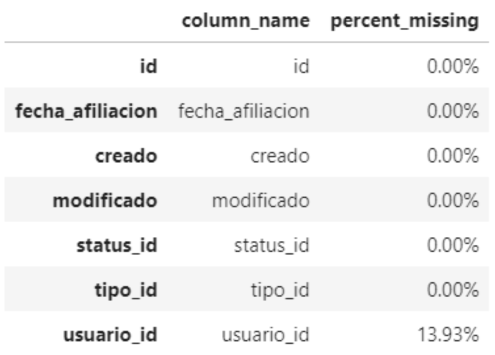
The variable “empresa_id” is not essential. This variable is directly related to the company to which the vehicle belongs.

The variable “perfil_id” is a foreign key to the “administracion_perfil” table which contains the birth date and gender of users, so it is a pity that we do not have this information.

The variable “usuario_id” referencing the operator of the company, who records the request.

We will remove these variables.

- **Administración Afiliación**

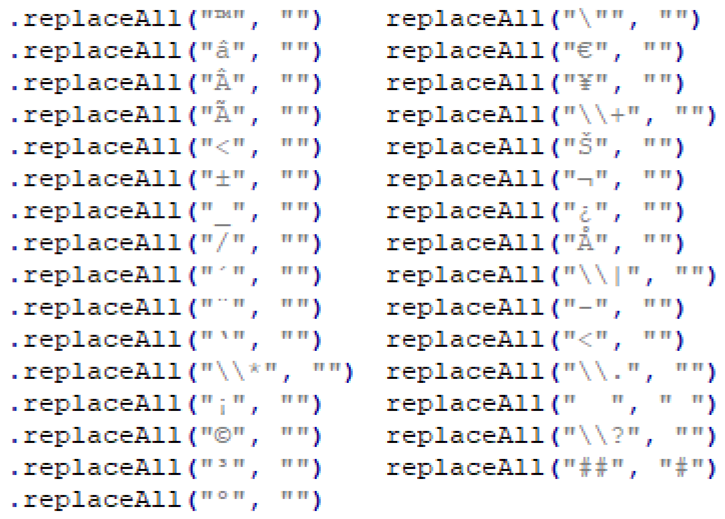


	column_name	percent_missing
id	id	0.00%
fecha_afiliacion	fecha_afiliacion	0.00%
creado	creado	0.00%
modificado	modificado	0.00%
status_id	status_id	0.00%
tipo_id	tipo_id	0.00%
usuario_id	usuario_id	13.93%

The "Afiliación" dataset does not have missing data, except “usuario_id” variable.

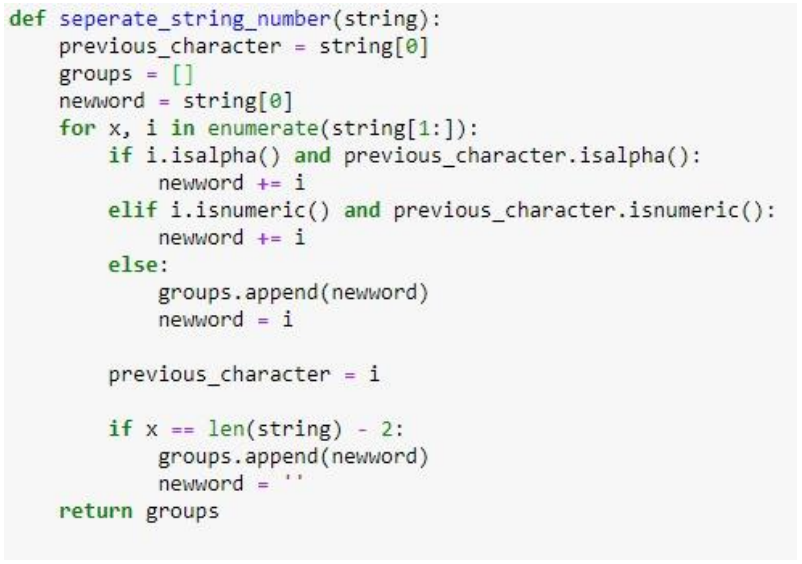
Data cleaning

We executed several lines of code in order to replace the characters-letters-numbers contained in the addresses that hindered the use of the database in the API, achieving to eliminate more than 35,000 characters in different positions of the database.



```
.replaceAll("™", "")    replaceAll("\"", "")
.replaceAll("â", "")    replaceAll("€", "")
.replaceAll("Â", "")    replaceAll("¥", "")
.replaceAll("Ã", "")    replaceAll("\\+", "")
.replaceAll("<", "")    replaceAll("Š", "")
.replaceAll("±", "")    replaceAll("¬", "")
.replaceAll("_", "")    replaceAll("¿", "")
.replaceAll("/", "")    replaceAll("Å", "")
.replaceAll("´", "")    replaceAll("\\|", "")
.replaceAll("¨", "")    replaceAll("–", "")
.replaceAll("`", "")    replaceAll("<", "")
.replaceAll("\\*", "")  replaceAll("\\.", "")
.replaceAll("¡", "")    replaceAll("  ", " ")
.replaceAll("©", "")    replaceAll("\\?", "")
.replaceAll("³", "")    replaceAll("##", "#")
.replaceAll("º", "")
```

Additionally, remove words that have 5 characters or less, since the API could not recognize it. First, we counted the total words of each of the addresses and for those who have a single address, we found that generally were not useful for the API, since the characters were pasted. Then, we use the function seperate_string_number, which is to separate the characters of the numbers and thus are able to save much of the data we have.



```python
def seperate_string_number(string):
    previous_character = string[0]
    groups = []
    newword = string[0]
    for x, i in enumerate(string[1:]):
        if i.isalpha() and previous_character.isalpha():
            newword += i
        elif i.isnumeric() and previous_character.isnumeric():
            newword += i
        else:
            groups.append(newword)
            newword = i

        previous_character = i

        if x == len(string) - 2:
            groups.append(newword)
            newword = ''
    return groups
```

Additionally, the composition of many addresses was complex to handle because of the lack of certain characters or the little coherence, many of these were eliminated or manipulated manually.

Data Analysis

We will utilize simple plots to explore relationships between some variables and the volume of requests and identify any taxicab usage patterns and some crucial days.

First, we are going to start with some univariate distribution to understand the data better.

➢ Mastermind Solicitud

tipo_servicio	Percentage
SC	99.91%
SEA	0.01%
SEI	0.00%
SEU	0.07%

It is evident that "SC", which we know from the company that states for "Servicio Corriente" (ordinary service), takes most of the requests made to the company.

➢ Mastermind Solicitud Cancelada

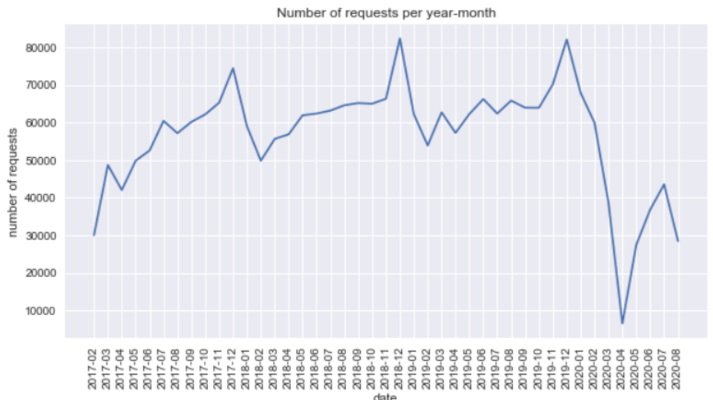
The following are the top 10 most frequent reasons for canceling a request.

motivo_cancelacion	Percentage
No se encontraton vehículos que atiendan la solicitud	63.51%
Ya no necesito el servicio	26.87%
La búsqueda de taxis tarda mucho	2.89%
Cliente abordó otro taxi	1.68%
El taxi tardó mucho	1.61%
Tomé otro taxi	0.63%
Dirección del cliente incorrecta	0.16%
ya no necesitan el taxi	0.14%
Motivo personal	0.12%
El conductor tiene una mala actitud	0.07%

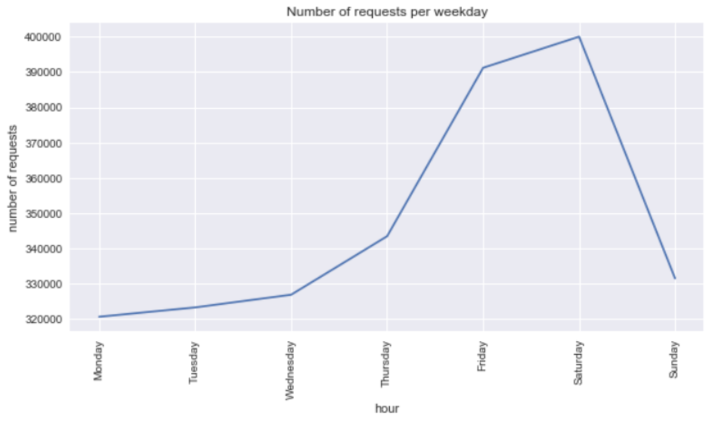
From this table, it is evident there is a problem in the supply causing many cancellations. Reason 1 "No vehicles were found to attend the request" account for more than 60% of the cases, so it seems the primary reason for cancelations is due to the company. Reason 2, "I do not need the service anymore" accounts for almost 30%. However, this reason can be due to the client getting another service.

Understand the key variables of interest

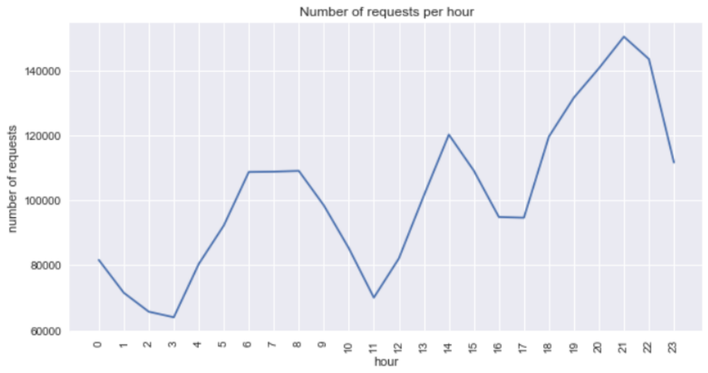
The following graphs show the trend of requests for taxicabs.



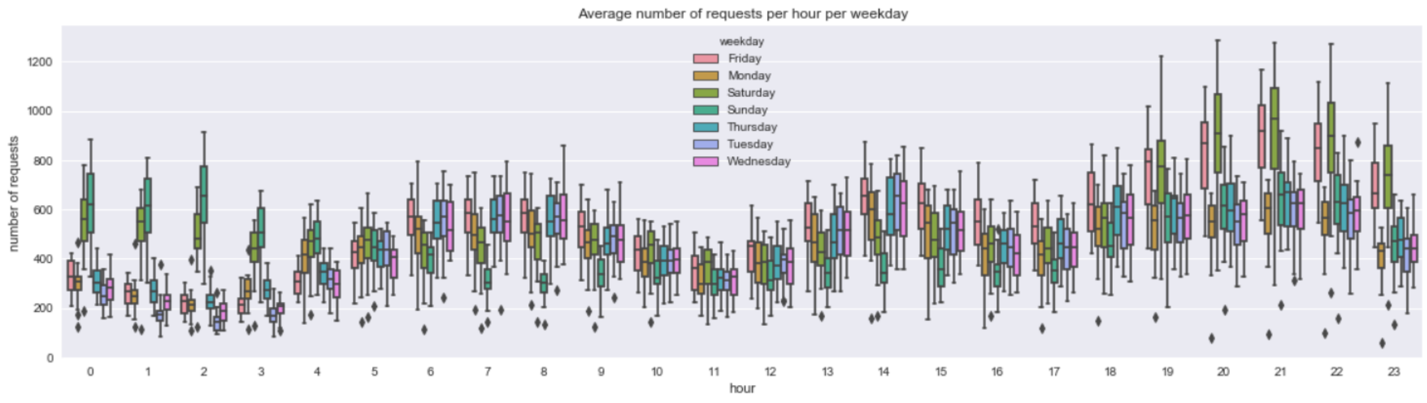
The variable "creado" spans the range from 06-Feb-2017 to 19-aug-2020. That is around 3.5 years of data. We can observe a seasonal behavior, such that for every year, February is the month with the least requests and grows steadily to December, after which falls to next February. That is true for 2017-2019 but is different for 2020, where requests kept falling beyond February to a low in April. This last fact is explained by the bio-security measures taken by the government starting March/2020.



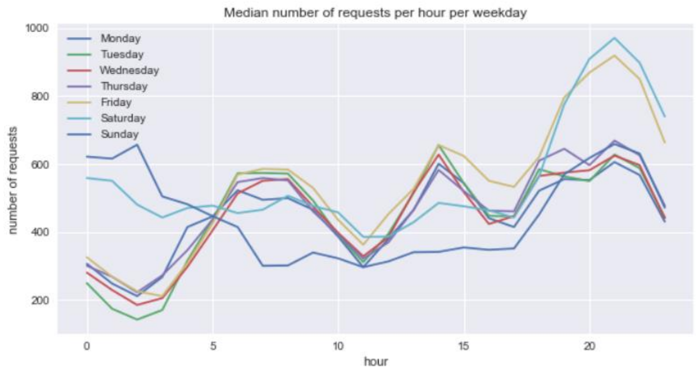
We can observe a seasonal behavior, such that Mondays have the least requests, following a growth all the way to Saturday, after which falls sharply to next Monday. We may conclude that Fridays and Saturdays have the most demand for taxicabs, way above other days of the week.



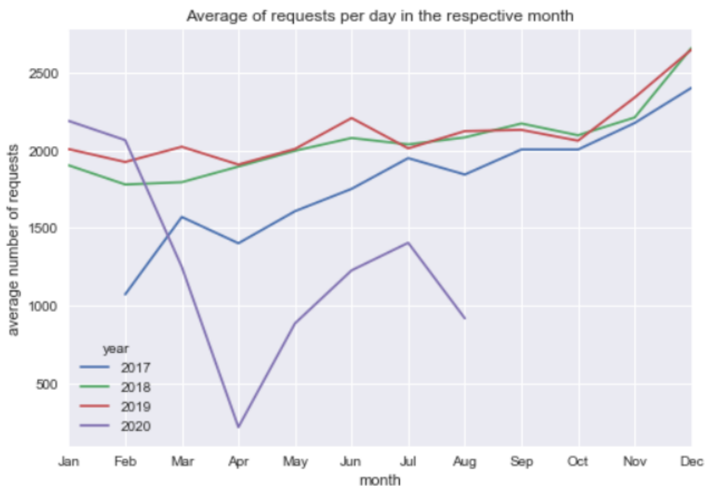
We observe a highly seasonal behavior. This behavior is explained by factors such as rush hours between 6-8 am and 6-9 pm, and lunchtime between 1-2 pm. It can also be noticed that the hardest rush hour is the evening one. However, this can be biased by Fridays and Saturdays.



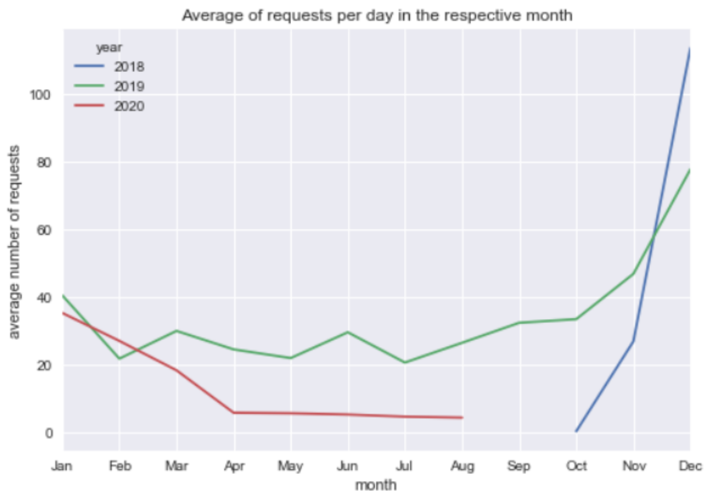
This is very interesting. In the early morning hours, there are two days that behave differently from the rest, which is Saturday and Sundays. This can be explained due to the fact that these days are the non-working days for most people. However, for the rest of the day, they seem to have lower demand than working days. A similar trend is observed for late-night hours, where the two different days are Friday and Saturday, the party days.



This graph shows a similar tendency for almost every day, except for the weekends.



We can see that the number of day-to-day requests on average is less for February, even though we do not have info in January for the year 2017, and 2020 is atypical due to the pandemic. Years 2018 and 2019 have similar behavior. In contrast, 2017 shows an average of requests significantly less, particularly for the first semester of this year. The year 2020 started well, but due to the pandemic, the year is atypical. Maybe it would be convenient to take on count the information since July 2017 for the model.



We have info on cancelations since Oct 2018, and we can see that the higher number of cancelations happens in December. In 2019 we observed that the number of cancelations is between 20 and 40 day-to-day requests on average.

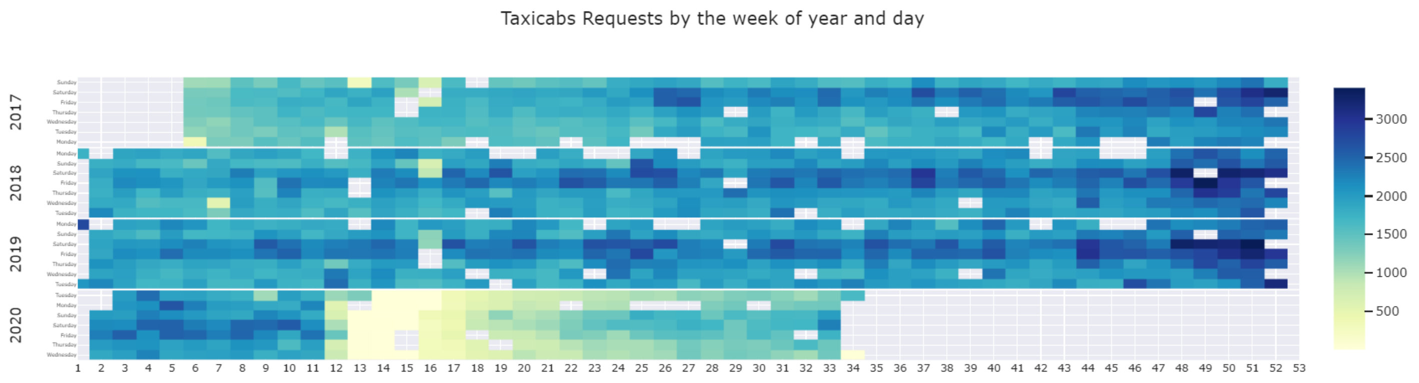
3. Feature Engineering

We expect that some holidays, no-car days, and other days like “El Carnaval de Negros y Blancos,” have a behavioral different from the other days, maybe a higher number of requests. We collected data from the “holidays_co” a package in python to get the holidays in Colombia to validate this hypothesis. We collected the ‘no-car days’ from Alcaldia municipal de pasto.
Día sin carro y sin moto.
Día del taxista: 7 de Mayo
Carnaval de Negros y Blancos.

	N días	Mean	Median
Viernes Santo	4	846.75	1063.5
Día de San José	4	1174.5	1204
Jueves Santo	4	1006	1294
Sagrado Corazón de Jesús	3	1360.67	1311
Año Nuevo	3	1373.33	1324
San Pedro y San Pablo	4	1350.5	1336
Ascensión del Señor	4	1308	1344
Día de la Raza	3	1336.67	1359
Corphus Christi	4	1303	1361
Todos los Santos	3	1446.67	1414
Independencia de Cartagena	3	1653	1417
Día del Trabajo	4	1366.5	1470
La Asunción de la Virgen	4	1487.5	1505
Batalla de Boyacá	4	1558.25	1550
Día taxista	4	1463	1581
Día de la Independencia	4	1788.5	1743
Carnaval	16	2342.81	2331
Día de los Reyes Magos	3	2151	2408
Día de la Inmaculada Concepción	3	2579	2476
Día de Navidad	3	2466.67	2507
Día sin carro	7	2834.86	2982

We can see that days on the Carnaval, día de los reyes magos, día de la inmaculada concepción y el día sin carro have a higher number of requests on average and median by day, and the other days have a lower number of request, specially "Viernes y Jueves Santo".

The following graph shows the number of requests by the week of year and weekday.



We can observe that Friday and Saturday have many requests for almost all the weeks each year, except 2020. There are some days in yellow (lower number of requests), and we should investigate why.
It is evident the low number of requests for weeks 12 and 19.due to the pandemic in 2020.
Finally, days like Friday and Saturday and the higher request days mentioned above have different behavior from the other days.

Other helpful variables.

Unlike more vanilla time series forecasting tasks -where one uses a single series to predict the series' future, this structure allows for transfer learning between series. For example, it may be that a group of customers have similar requests patterns, customers one-timer, or recurrent customers. Thus, it would make sense for a model to learn shared patterns to provide more robust forecasts.

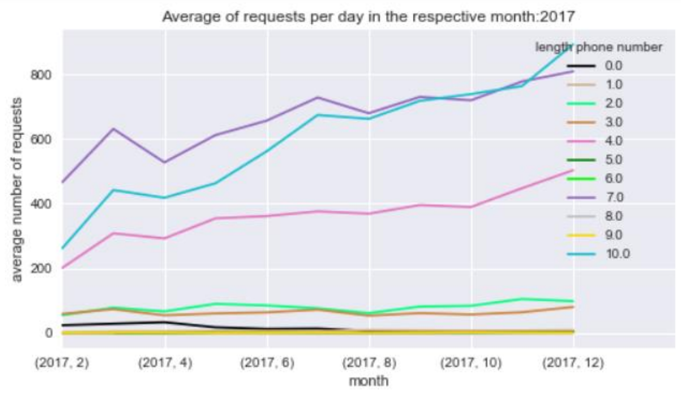
Based on the above, we constructed some variables that could help group customers with a similar underlying structure before diving into modeling. This way, we could similarly pre-process series within groups and train a model for each cluster, which specializes in learning this underlying structure.
We will use an approach of clustering to time series using Time Warping Distance Metric.

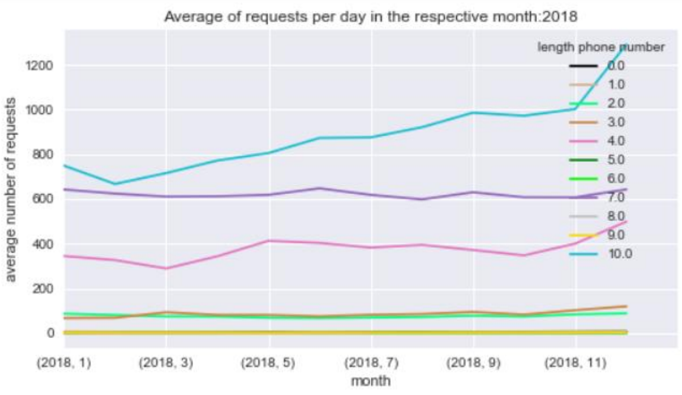
Phone type:

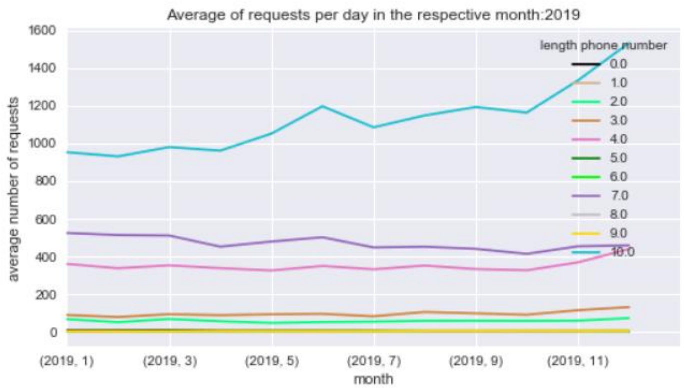
We identify the length of phone numbers and classify them into "mobile," "landline," and "others." The following is the distribution for the "phone type":

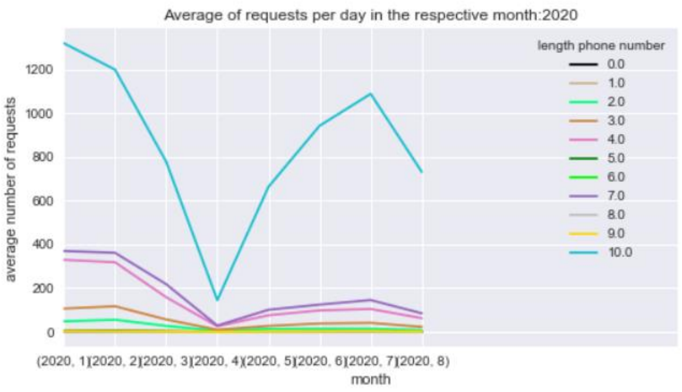
phone_type	**N**	**%**
mobile	1,146,155	47%
landline	665,286	27%
len_4	423,027	17%
other	194,494	8%
missing	7,889	0%
Total	**2,436,851**	

We can see most are mobile and landline. The following graphs show the behavior of requests by each phone length:





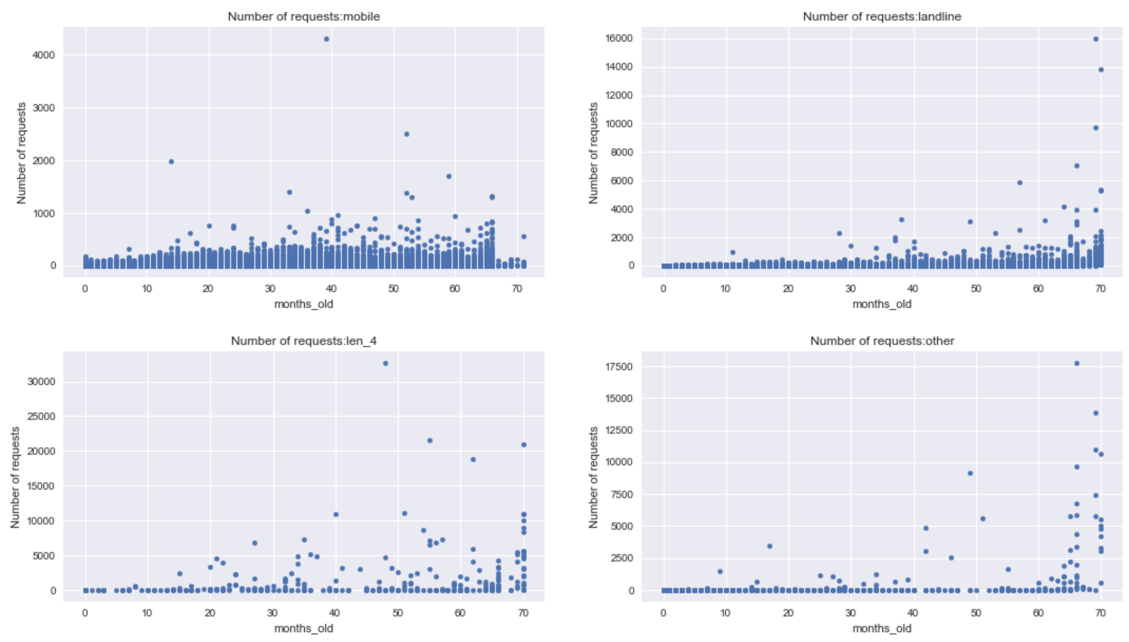




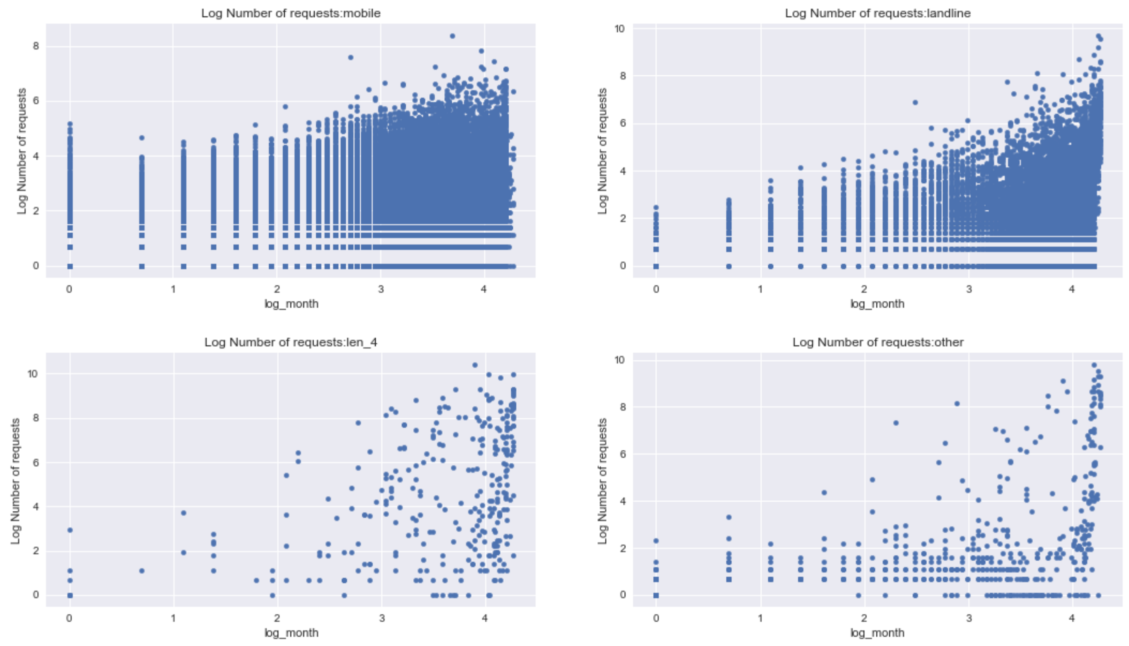
We can see that mobile (phone number length 10) has a higher request on average per day, and the requests from landline (length phone number 7) have around 200-400 requests on average per day.
There are many requests from phone numbers with four digits, even though we do not know length four what means.
On the other hand, requests from numbers with different sizes seem to be an inconsistency of the data, so we will remove them.

Months old:

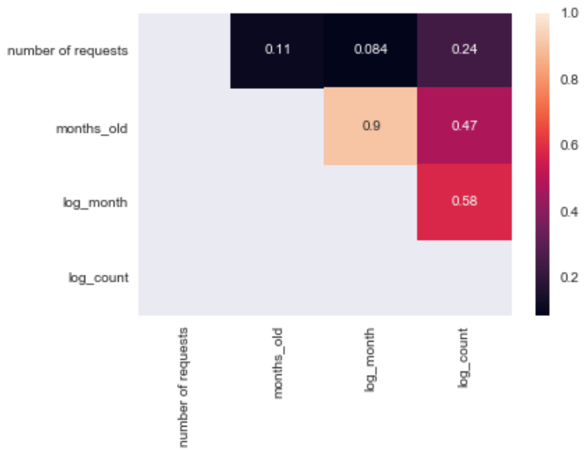
We calculate the maximum difference between the date of creation of the phone number in the "directorio" data frame and the date of requests, and we expect that customers with high "months_old" have more requests, and we can see that, but the relationship seems week. Additionally, we can see a few clients with extreme values of requests from the following graph:



If we plot the logarithm for both variables, we can see that the relationship seems strong, and the extreme values seem to disappear.



We will calculate the correlation between “months_old” and the number of taxicab's requests.



We can see that the correlation is higher when we use the logarithm of the variables.

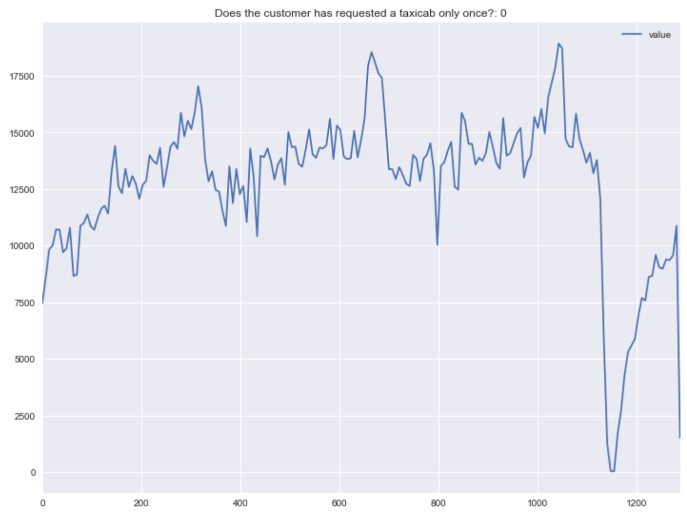
Trend variables:

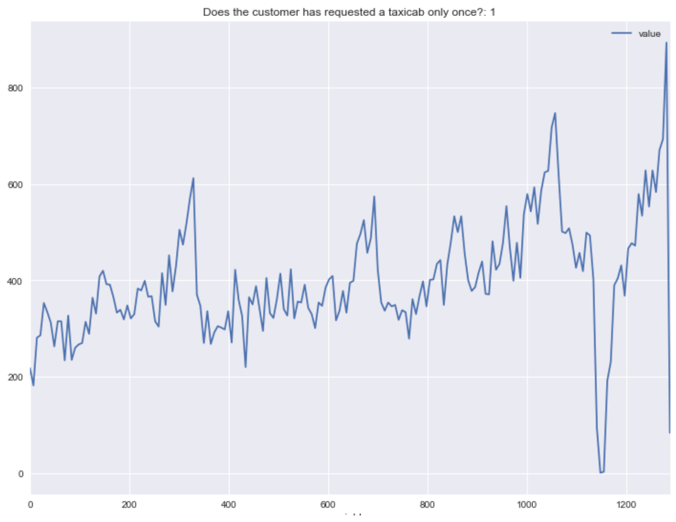
We calculated some variables to identify some trends from customer historic requests. We calculate variables like mean, maximum, minimum, standard deviation, and slope of each customer's historic requests on daily, monthly, and quarterly windows. The following table shows the correlations between these variables and the requests:

variable	correlation
slope_q	-0.352
slope_m	-0.122
mean_rq_d	0.501
min_rq_q	0.622
max_rq_d	0.656
std_rq_m	0.864
std_rq_q	0.891
max_rq_m	0.905
max_rq_q	0.936
mean_rq_m	0.969
mean_rq_q	0.983

Flag one-timer.

We identify customers one-timer, which represents 50% of the total customers. The following graph shows the differences between the series from one-timer customers (flag = 1) and recurring customers (flag = 0).





We have 1290 days of history, and the graph shows the requests binned by week, which will lead the number of requests per time unit to be less volatile.

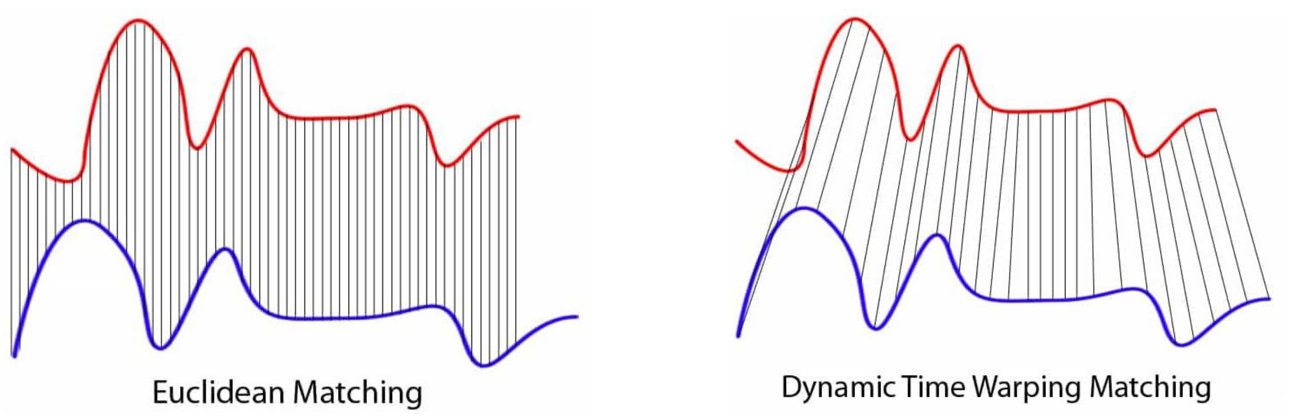
The number of requests for the total one-timer customers ranges from 200 to 600, and recurring customers range from 10,000 to roughly 17,000, except for the pandemic period.

4. Modeling and Prediction

Time series clustering for forecasting preparation

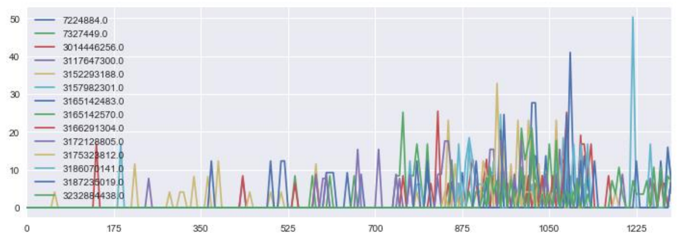
The most common approach to time series clustering is to flatten the time series into a table, with a column for each time index (or aggregation of the series), and directly apply standard clustering algorithms like k-means with Euclidian distance. Intuitively, the distance measures used in standard clustering algorithms, such as Euclidean distance, are often not appropriate to time series. A better approach is to replace the default distance measure with a metric for comparing time series, such as Dynamic Time Warping (DTW).

DTW compares each element in series "X" with each element in series "Y". The comparison, distance between "X" and "Y", $d(x_i, y_j)$, is just the simple subtraction $x_i — y_j$.
Then for each x_i in X, DTW selects the nearest point in Y for distance calculation. The following graph illustrates the Euclidean distance and DTW for time series:

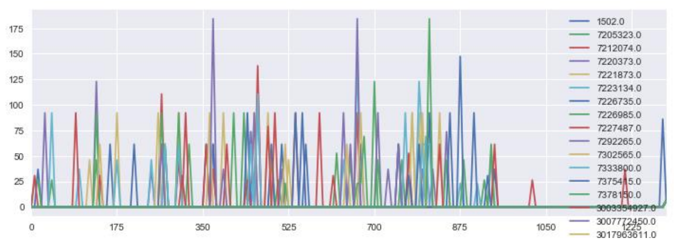


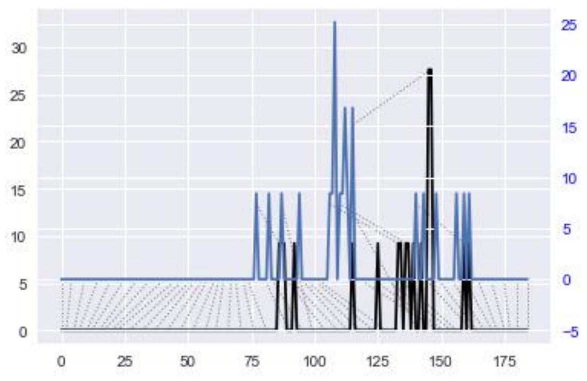
For this problem, we are sample 200 customers as an example and calculate clusters for this sample. The following graphs show some clusters resulting:

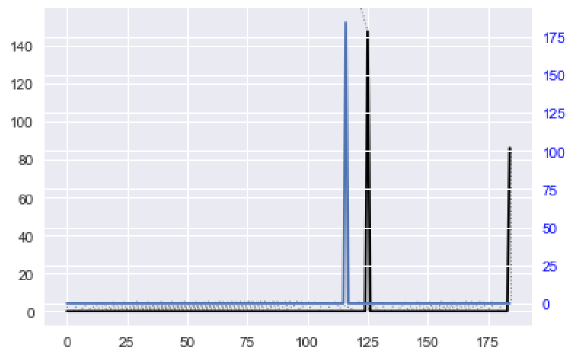
Cluster 1



Cluster 2







We can see that customers in cluster 1 are more recurring than customers in cluster 2.

Time Series Forecasting with ARIMA -R.

We use R since it is a more robust software in diverse methodologies

Different techniques were implemented in the statistical package R to make the model, under a time series analysis of the number of applications. To start the decomposition of the series, by means of an arima model, with the auto arima command and an exponential smoothing model was made.

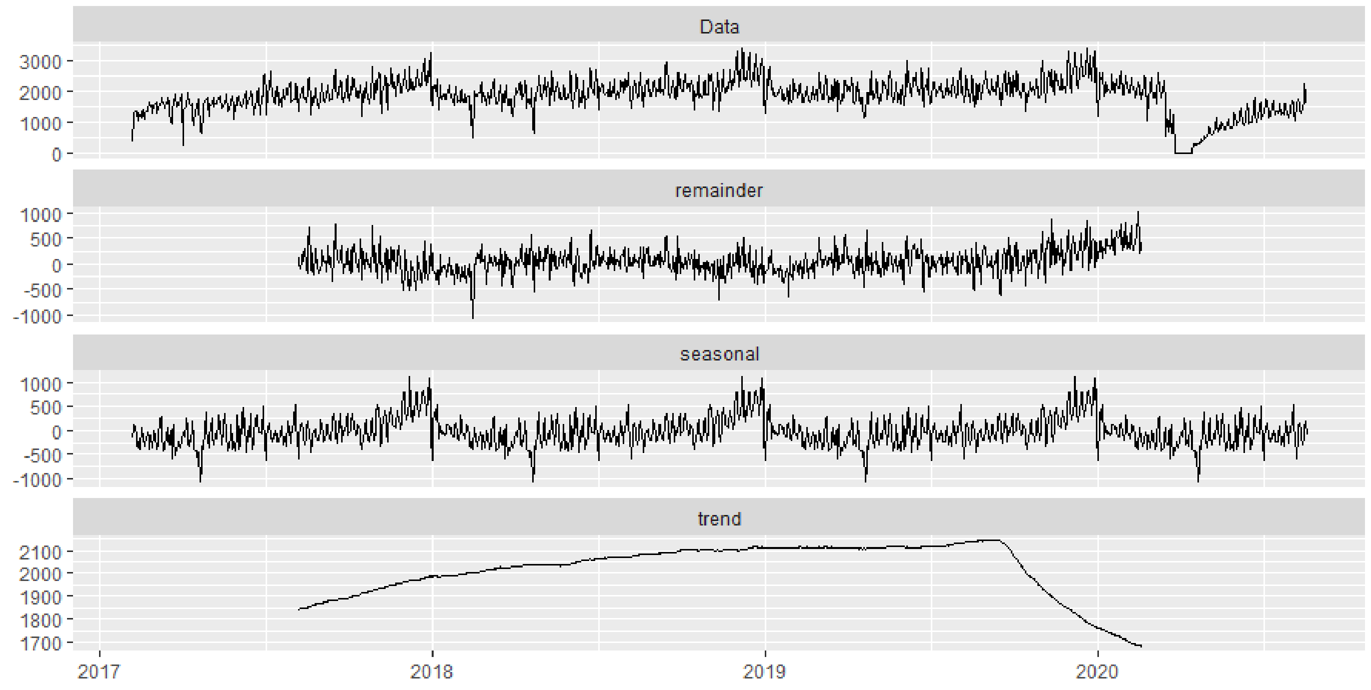
```
fit<-decompose(series2, type="additive")
autoplot(fit)
auto.arima(series2)
acf(diff(series2,1))
pacf(diff(series2,1))

fit=Arima(series2,order=c(5,1,1),include.mean=TRUE)
plot(forecast(fit,h=30))
tsdiag(fit)
```

In addition, a Decomposition was made in order to improve the base to be able to work it more comfortably, this Decomposition was worked in python.

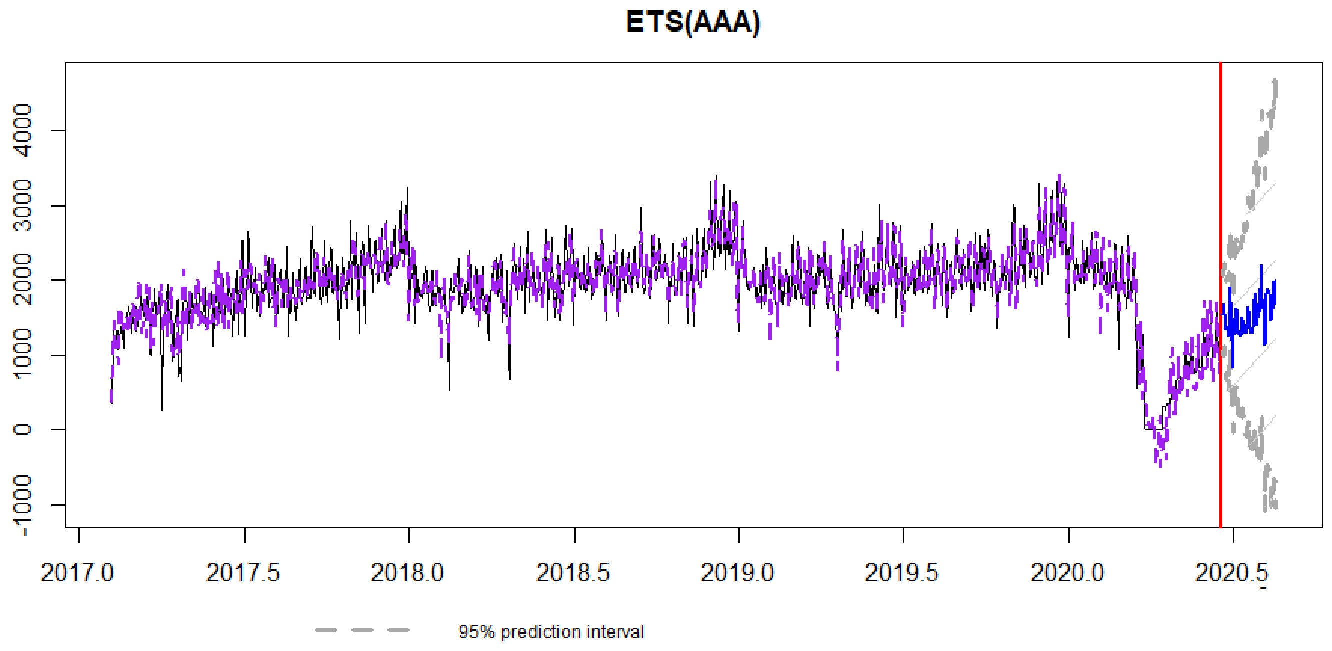
We are reading the database and we condition it to get the result per year.

```
series <- ts(Serie_taxis$count, frequency=365, start=c(2017, 37))
plot(series)

series2 <- na_kalman(series,model="auto.arima")
plot(series2)
```

The decomposition of the series showed us:



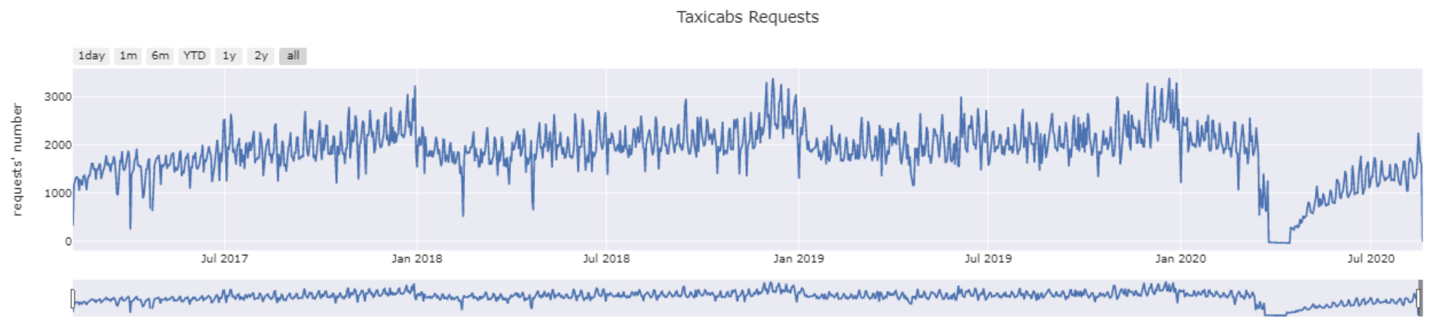
1)The First: We show the trend over time, especially that sharp decrease is due to the current covid pandemic
2)Display the residues
3)The third shows us a pattern of every 7 days, showing the weekly cyclical behavior that allows us to infer the presence of some trend.
4)The last one shows us the trend showing once again the abrupt decrease in the number of requests due to the current pandemic

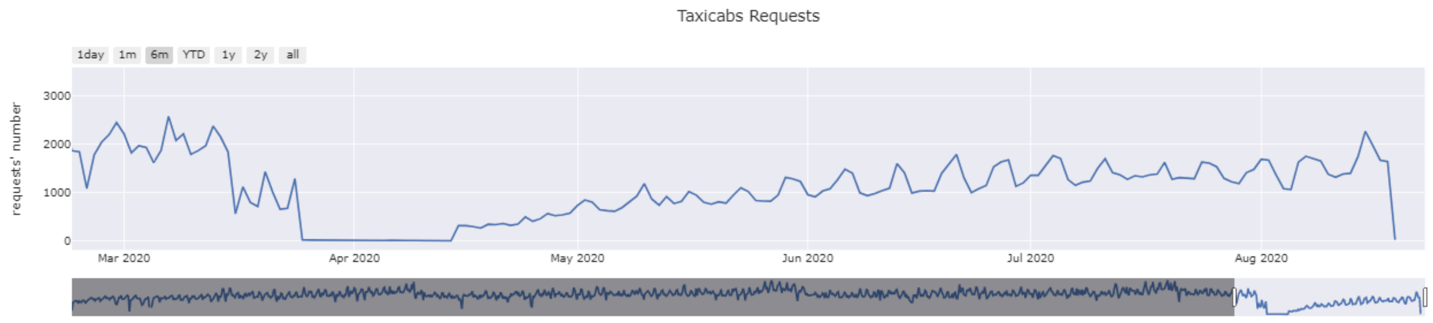


The best model with information criteria is an arima 5.1.1, for which we expressed a simple and partial correlation distribution, showing that every 7 periods is significant, that is, a seasonality each period, this with a confidence level of 95%.

Only R was used for the statistical analysis of the time series, but for reasons of interactivity design with the website it was decided to use Python.

Time Series Forecasting with Prophet – Python.

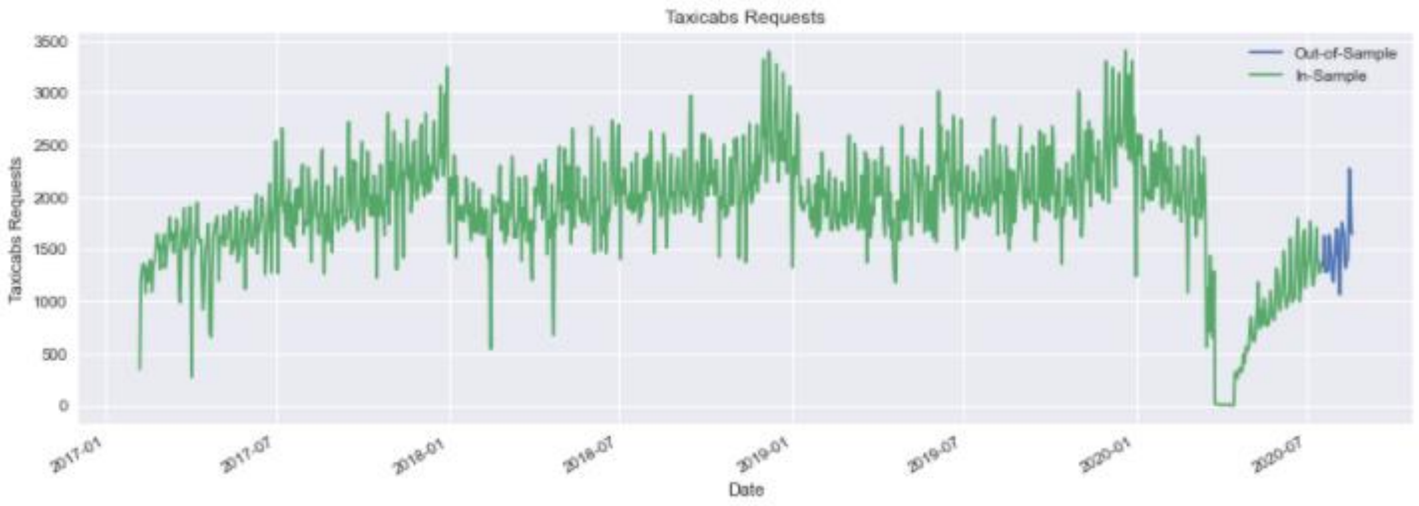




The first graph shows the series of taxicab requests for all time in the observation window, and the second, the last six months of available data. We can see that the request's number is approximately 2,000 daily before the pandemic. There are very few requests in the covid lockdown period, and post covid, the taxicab requests have been recovering. The last point (August 19, 2020) seems to do not have the data ultimately.

Make an In-Sample and Out-of-Sample data frames.

We will cut off the data after July 2020 to use it as our validation set. We will train on the earlier data. We choose this cut off because we want to capture the pandemic's effect and some values post-pandemic that shows the recovering of taxicab requests.



Simple Prophet Model

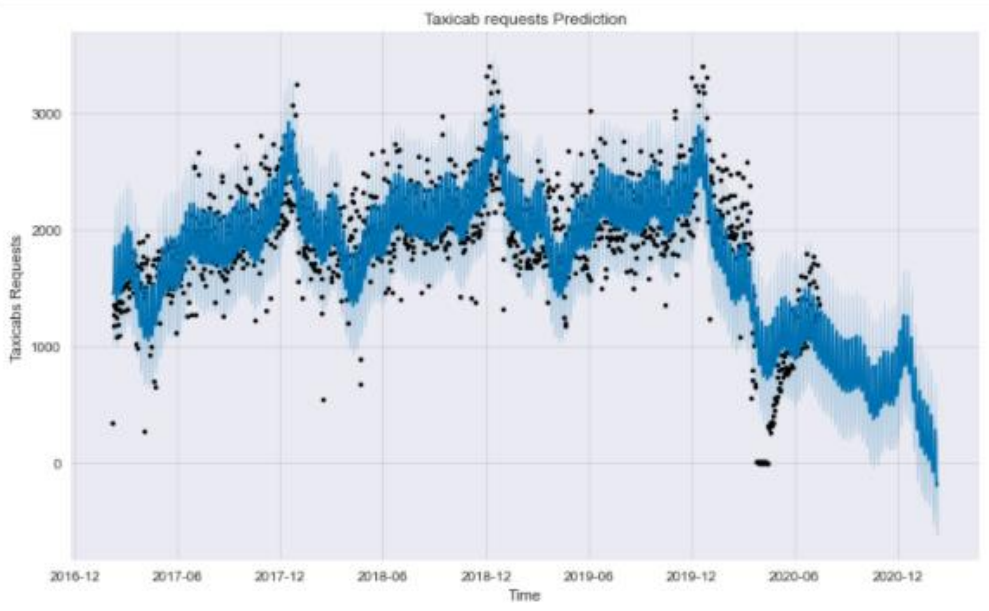
We use Prophet an open-source time-series forecasting library made available by Facebook's core data science team. They use a decomposable time series model with three main model components: trend, seasonality, and holidays. There are combined in the following equation:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t.$$

Here $g(t)$ is the trend function that models non-periodic changes in the value of the time series, $s(t)$ represents periodic changes (e.g., weekly and yearly seasonality), and $h(t)$ represents the effects of holidays on potentially irregular schedules over one or more days. The error describes any idiosyncratic changes that are not accommodated by the model; later, they make the parametric assumption that is normally distributed.

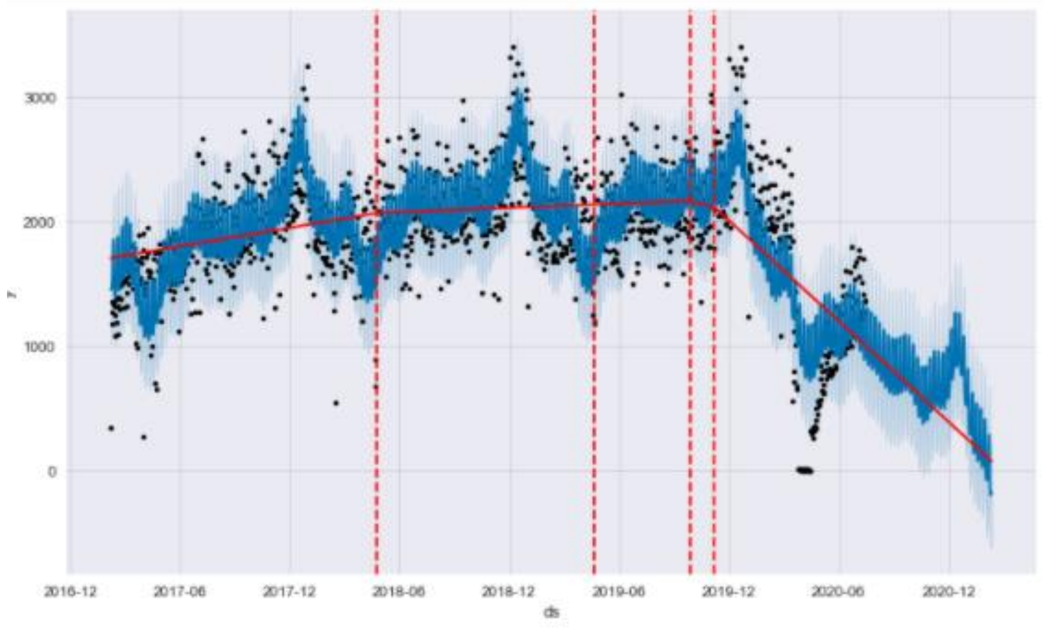
The first model we are training is a simple model that makes future predictions for 210 days. We plot the forecast by Prophet as follows:

```
# Setup and train simple model.
model1 = fbprophet.Prophet()

model1.fit(data_inSample)

# Predict
future_requests= model1.make_future_dataframe(periods=210, freq='D')
forecast_rq = model1.predict(future_requests)

# Predict on out sample with model
rq_os_fcst = model1.predict(df=data_outSample)

# Plot
fig = model1.plot(forecast_rq)
```



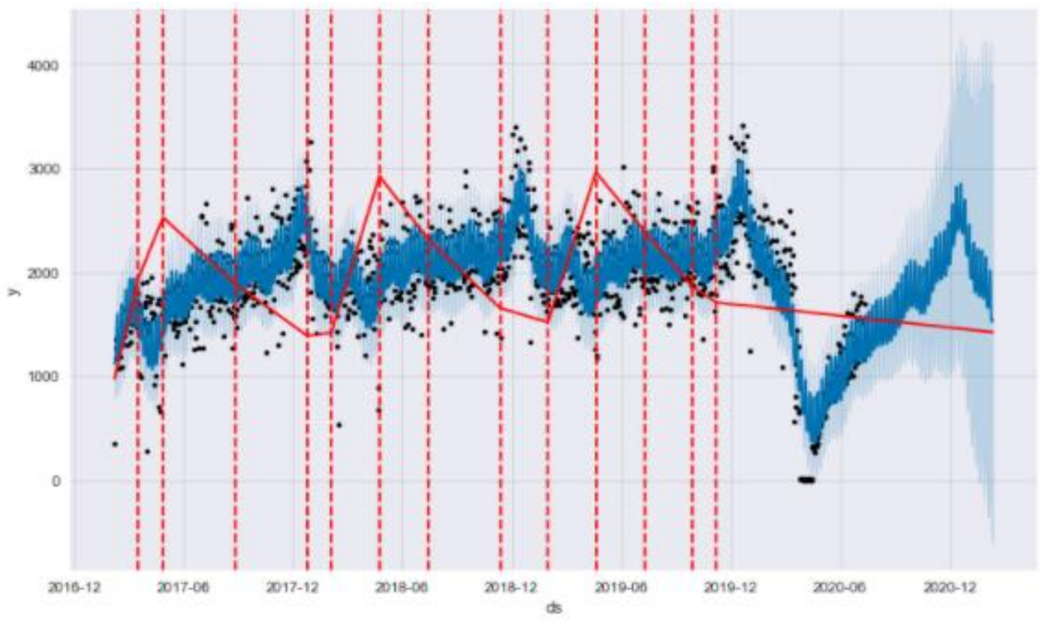
The black points represent the actual values, and the blue line represents the predicted values. We can see that the future predictions have a downward trend, unlike for the real values in the out-of-sample set, where it shows an upward trend.

Additionally, we can see that the trend of this series changes abruptly for some periods. To handle these points, Prophet attempts to detect these changes automatically using a Laplacian or double exponential prior. By default, the shift points only fit the 1st 80% of the time series, allowing sufficient runway for the actual forecast. In the taxicab requests data, note a sharp dip in April 2020 due to the COVID outbreak. Let us display the change points detected by Prophet:



Visually it appears that the general trend is correct, but it is being underfitted. For example, the upward trend in the period post covid is not detected. We can use the parameter "changepoint_prior_scale", which is set to 0.05 by default, to adjust the trend change. Increasing its value would make the trend more flexible and reduce underfitting at the risk of overfitting. We set it to 0.5, as suggested by the Prophet Documentation Guide. With this parameter, we fit a second model with the following predictions:

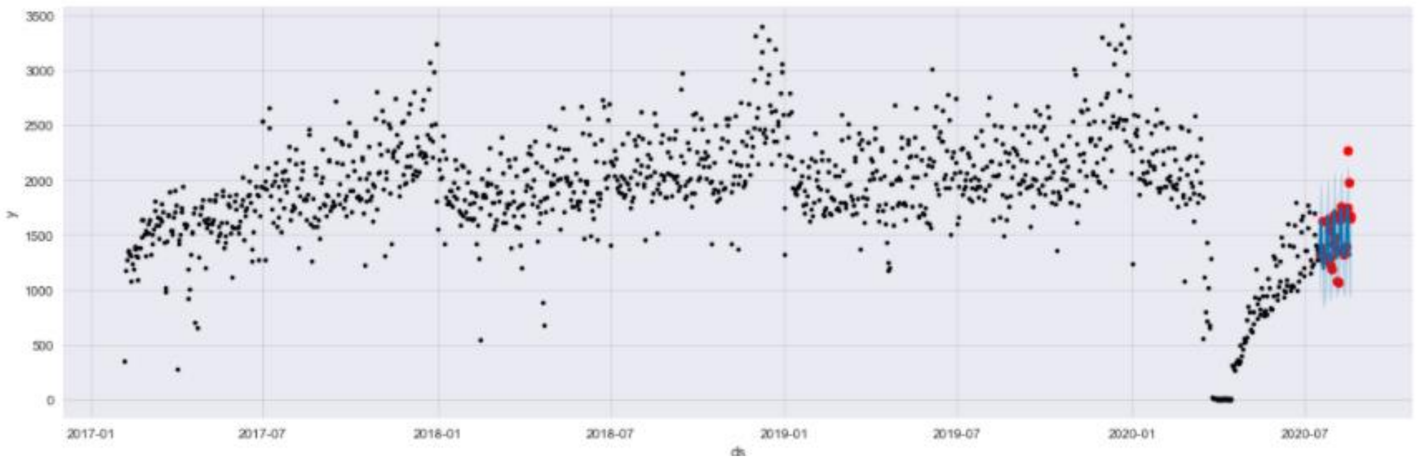
```
# Setup and train simple model.
model2 = fbprophet.Prophet(changepoint_prior_scale=0.5)

model2.fit(data_inSample)

# Predict
future_requests= model2.make_future_dataframe(periods=210, freq='D')
forecast_rq = model2.predict(future_requests)

# Predict on out sample with model
rq_os_fcst = model2.predict(df=data_outSample)

# Plot
fig = model2.plot(forecast_rq)
a=add_changepoints_to_plot(fig.gca(),model2,forecast_rq)
```



We can see that the upward trend post-covid is fixed with this new model, and the covid period is detected better than the first model.

Compare Forecast to Actuals

We plot the forecast vs. actual values in the out-of-sample, as shown in the following graph:



The red points represent the actual values in the out-of-sample.

Error Metrics

We calculate the mean square error (MSE), the mean absolute error (MAE), and mean absolute percent error, and we get the following values:

RMSE error	49,031.98
MAE error	161.42
MAPE error	10.41

Cross Validation

We can perform cross-validation to measure forecast error. Cut off points are selected, and we train the model with data up to that point. We can then compare the prediction vs. actual data over a specified time horizon.

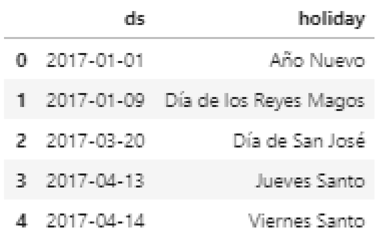
```
# Setup and train simple model.

from fbprophet.diagnostics import cross_validation
df_cv = cross_validation(model2, initial='1000 days',
                         period='90 days', horizon = '180 days')

from fbprophet.diagnostics import performance_metrics
from fbprophet.plot import plot_cross_validation_metric

df_p = performance_metrics(df_cv)
df_p.head()
```

horizon	mse	rmse	mae	mape
18 days	120,314.76	346.86	278.25	0.1201
19 days	119,793.20	346.11	276.76	0.1199
20 days	120,577.73	347.24	280.66	0.1218
21 days	122,626.29	350.18	284.06	0.1221
22 days	106,706.79	326.66	259.63	0.1122

Adding Holidays

Next, we will see if adding holiday indicators will help the accuracy of the model. "Prophet" comes with a Holiday Effects parameter that can be provided to the model before training.

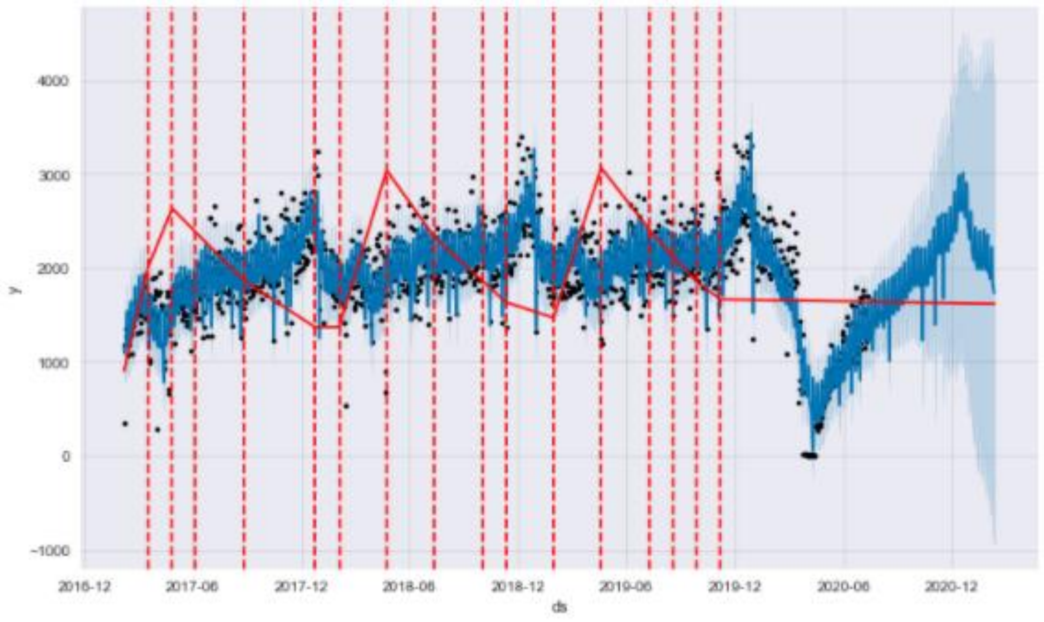
We add the holiday data frame that we collect in the featuring engineering:



	ds	holiday
0	2017-01-01	Año Nuevo
1	2017-01-09	Día de los Reyes Magos
2	2017-03-20	Día de San José
3	2017-04-13	Jueves Santo
4	2017-04-14	Viernes Santo

The code for the model:

```
# Setup and train model with holidays.
model_with_holidays = fbprophet.Prophet(changepoint_prior_scale=0.5,
                                        holidays=holiday_df)
model_with_holidays.fit(data_inSample)

# Predict with holidays
future_requests= model_with_holidays.make_future_dataframe(periods=210, freq='D')
forecast_rqHds = model_with_holidays.predict(future_requests)

# Predict on out sample with model
rq_os_fcst_with_hols = model_with_holidays.predict(df=data_outSample)

# Plot
fig = model_with_holidays.plot(forecast_rqHds)
a=add_changepoints_to_plot(fig.gca(),model_with_holidays,forecast_rqHds)
```



We can see some peaks for the ends of December each year and some low points for some holiday Mondays.

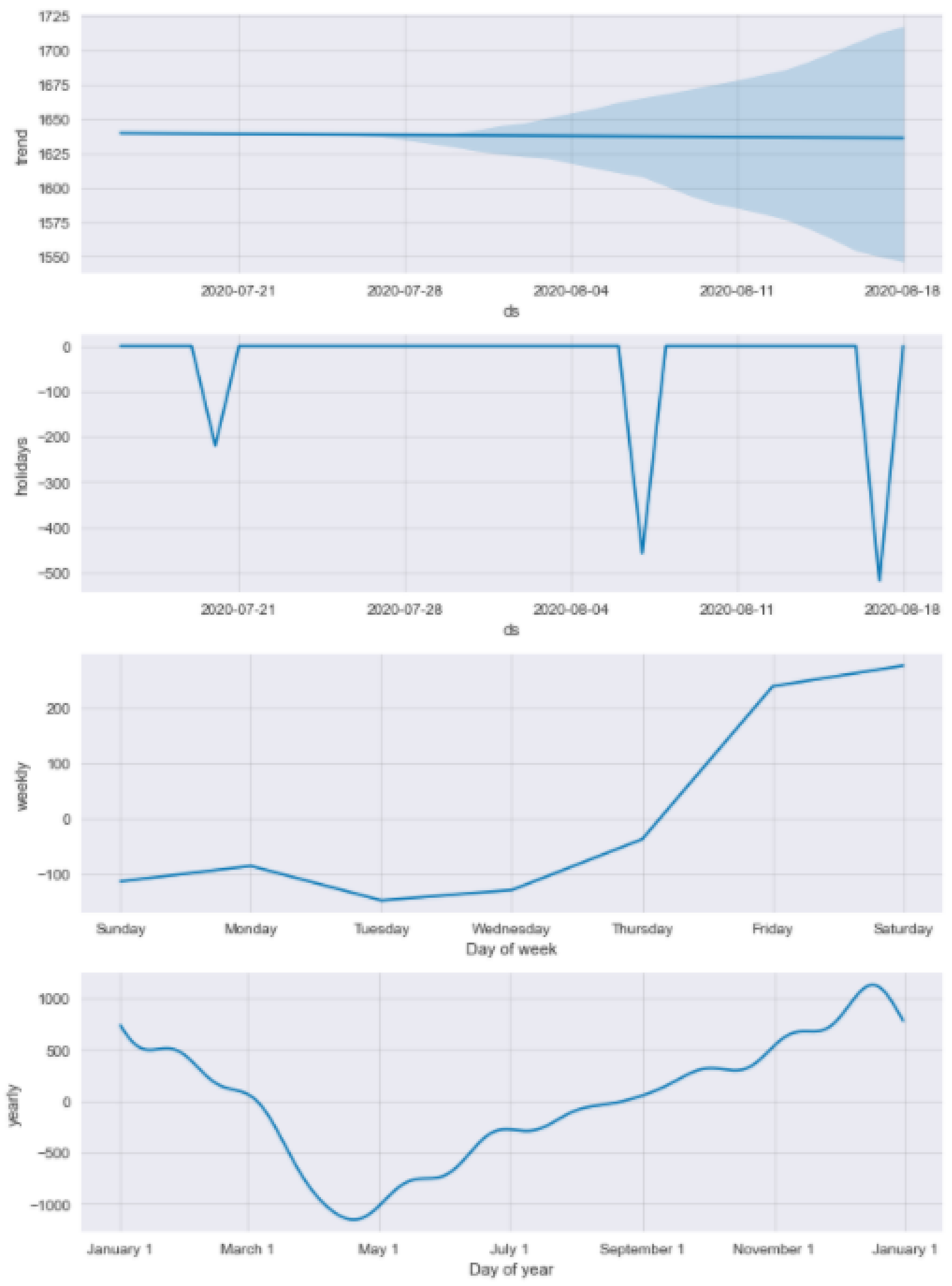
Error Metrics with Holidays Added

RMSE error	59,902.20
MAE error	200.24
MAPE error	13.41

The model's errors with holidays are slightly higher than the previous errors, maybe because we have little information for the out-of-sample, without holidays. However, seems this model best fits due to the correction of the forecast for the holidays.

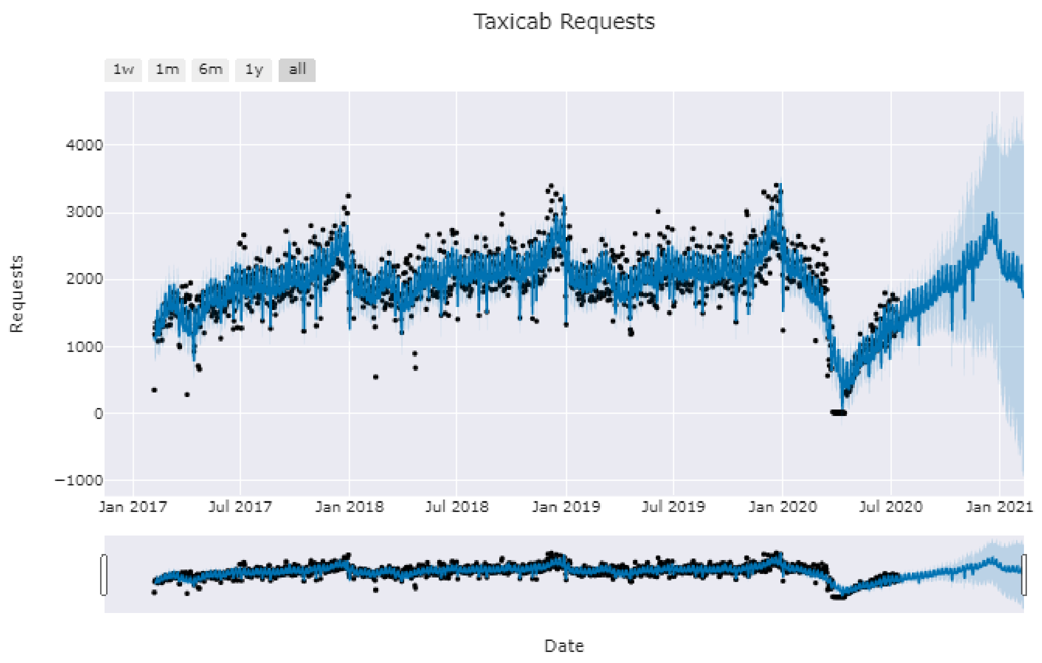
Plot Model Components

Using the "plot_components" function, we can display the components of the model:



We observe the low points correspond to holiday Mondays (July 20, August 7, and 17). The yearly seasonality for the Decembers, and as we expect the weekly trend, is upward on with a peak on Friday and Saturday.

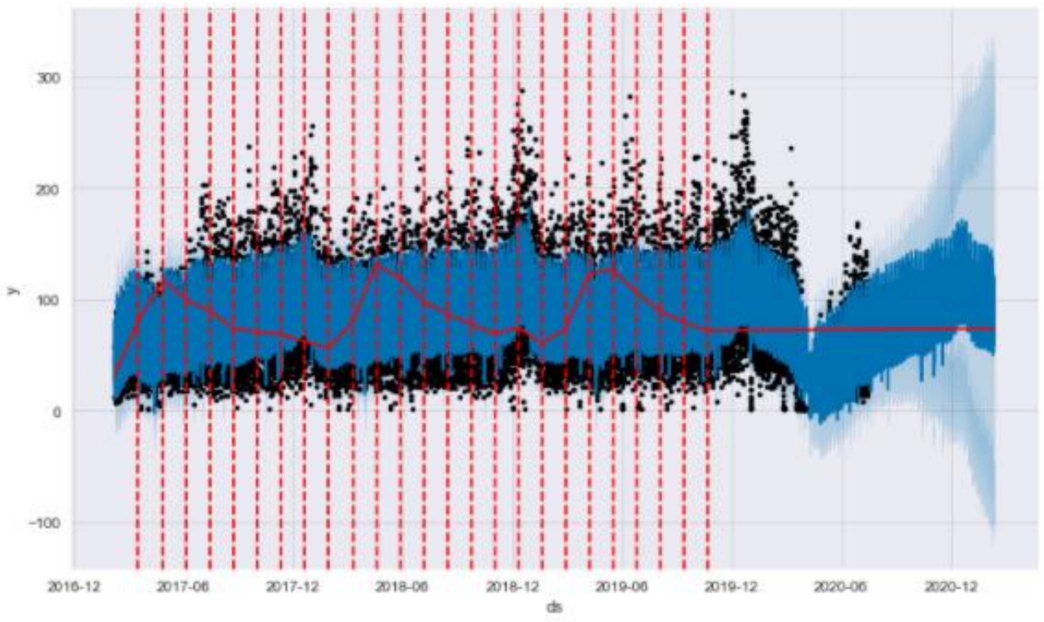
The following graph show the actual and predicted values with the "model_with_holidays."



Prophet Model to forecast hourly data.

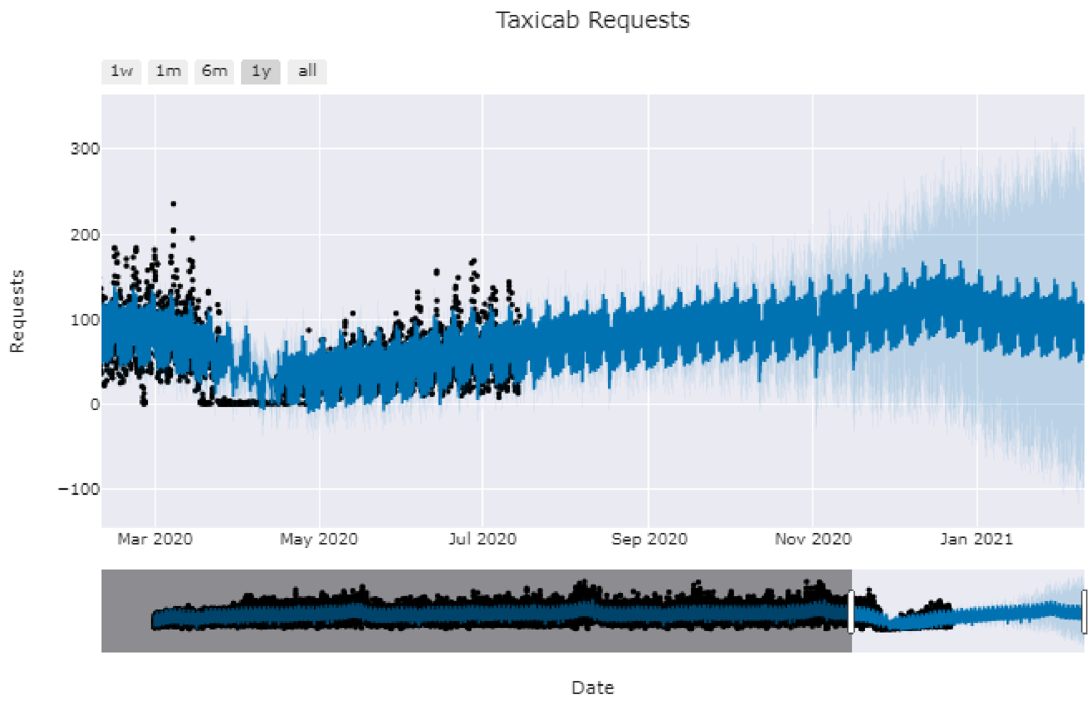
We fit a model as the "model_with_holidays" with hourly data. We predict the values for 5,400 future points with an hourly freq.

The following graph shows the actual and forecast values for in-sample and out-of-sample data frames.



We can see that the model captures the pandemic period, and the trend post-covid is upward. Additionally, we observe low points for some holidays. The following are the errors and the actual and predicted values for the out-of-sample and future window:

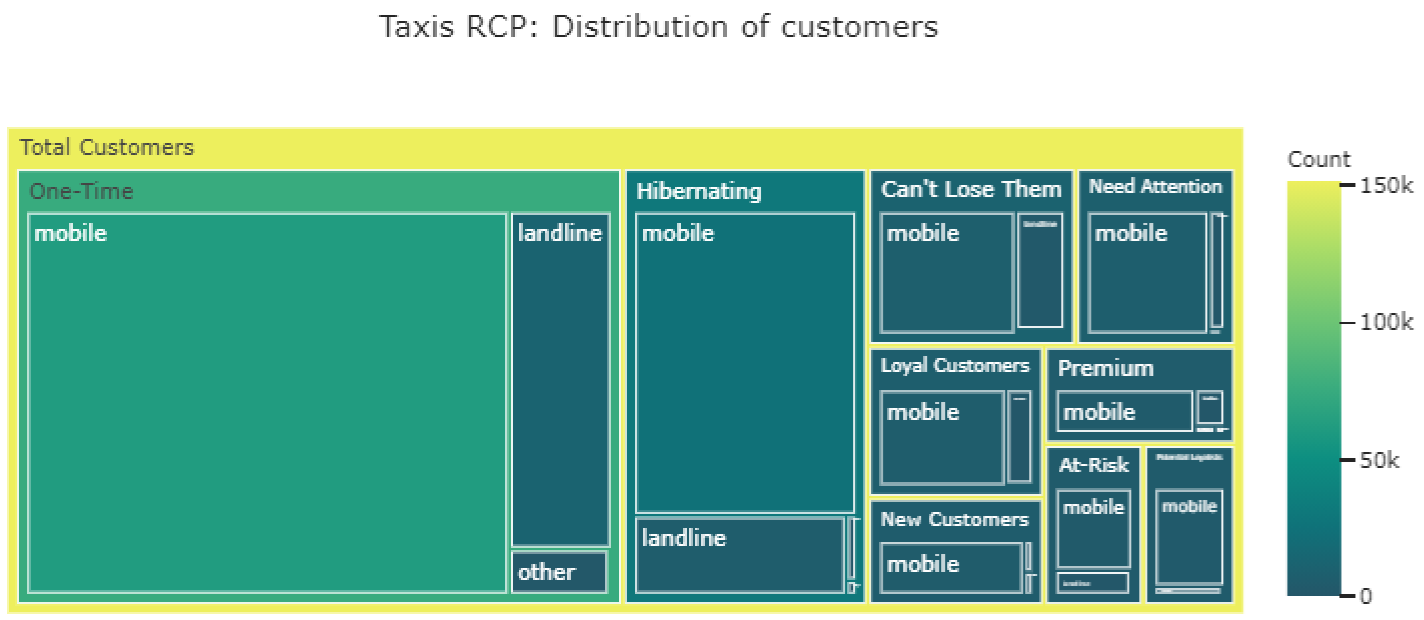
RMSE error	523.23
MAE error	18.23
MAPE error	50.82



5 Know your customer

This section describes a client segmentation based on customer activity's freshness and the customers' total number of requests. We have calculated some variables: Recency: Time since last activity, number of total requests by customer, average monthly requests, and months seniority. With these variables, we calculate a score for each customer in a range of 111 – 444, where 111 represents customers with recent activity and most frequent activity, and 444 represents customers with last activity more than 12 months ago and a low-frequency activity.

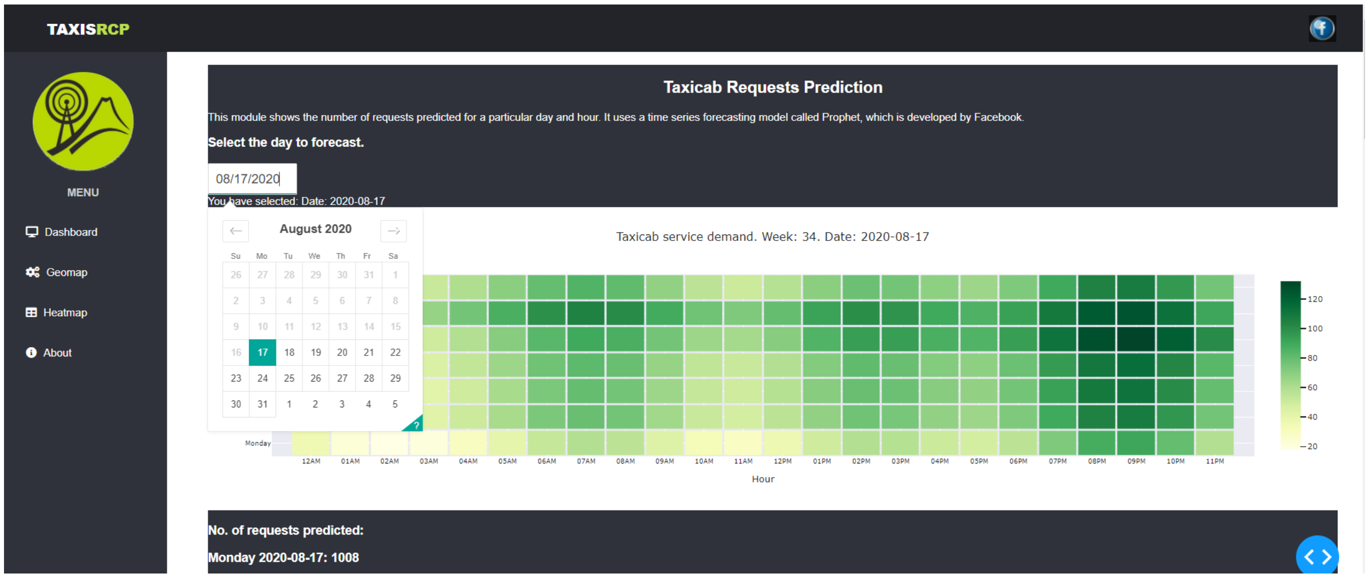
The following graph shows the distribution for each segment:



- **One-time:** *50%* are one-time service customer.
- **Who are their best customers?** *4%* are Premium: Recently have requested a service, and they are the most often customers.
  5% of your customers are Loyal.
- **Who has the potential to become valuable customers?** *3%* are Potential Loyalist, they are recent customers with average frequency.
- **New Customers:** Start building relationships with these customers by providing onboarding support. 4% are new customers.
- **At-Risk Customers** *3%* of your customers are at risk to churn. They request a service often but have not required a service recently.
- **Cannot Lose Them** They are similar to Premium customers but haven't been requesting a service recently. *7%* of your customers are in this segment.
- **Need Attention**: *5%* of your customers you need incentivizing to use your service.
- **Hibernating**: *20%* of your customers contribute to your churn rate.

6 Front End

The application has a forecast module where we can select a particular day to get the forecast for the chosen day, and the number of requests predicted by day and hour for the week of the day picked:



Taxicab Requests Prediction

This module shows the number of requests predicted for a particular day and hour. It uses a time series forecasting model called Prophet, which is developed by Facebook.

Select the day to forecast.

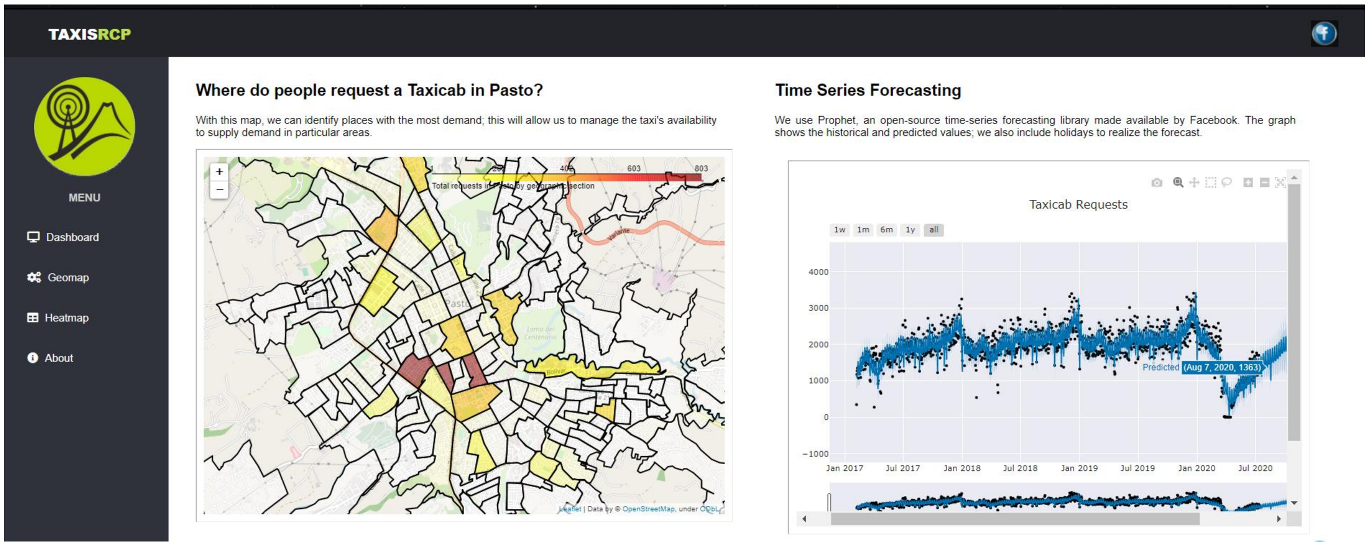
08/17/2020

You have selected: Date: 2020-08-17

No. of requests predicted:

Monday 2020-08-17: 1008

In the second part, we can see the map with the density of historic requests in Pasto city to identify places with the most demand; this will allow us to manage the taxi's availability to supply demand in particular areas. Additionally, we can see a graph that shows historical and predicted values.



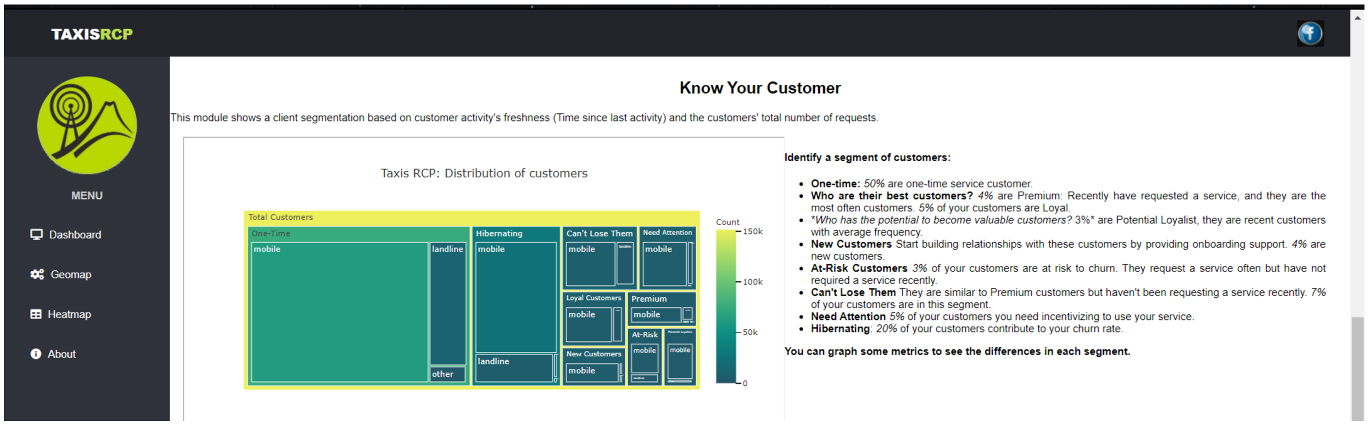
Where do people request a Taxicab in Pasto?

With this map, we can identify places with the most demand; this will allow us to manage the taxi's availability to supply demand in particular areas.

Time Series Forecasting

We use Prophet, an open-source time-series forecasting library made available by Facebook. The graph shows the historical and predicted values; we also include holidays to realize the forecast.

Finally, we can see a module "know the customer", this allows to see a segment customers and description about each segment.



Know Your Customer

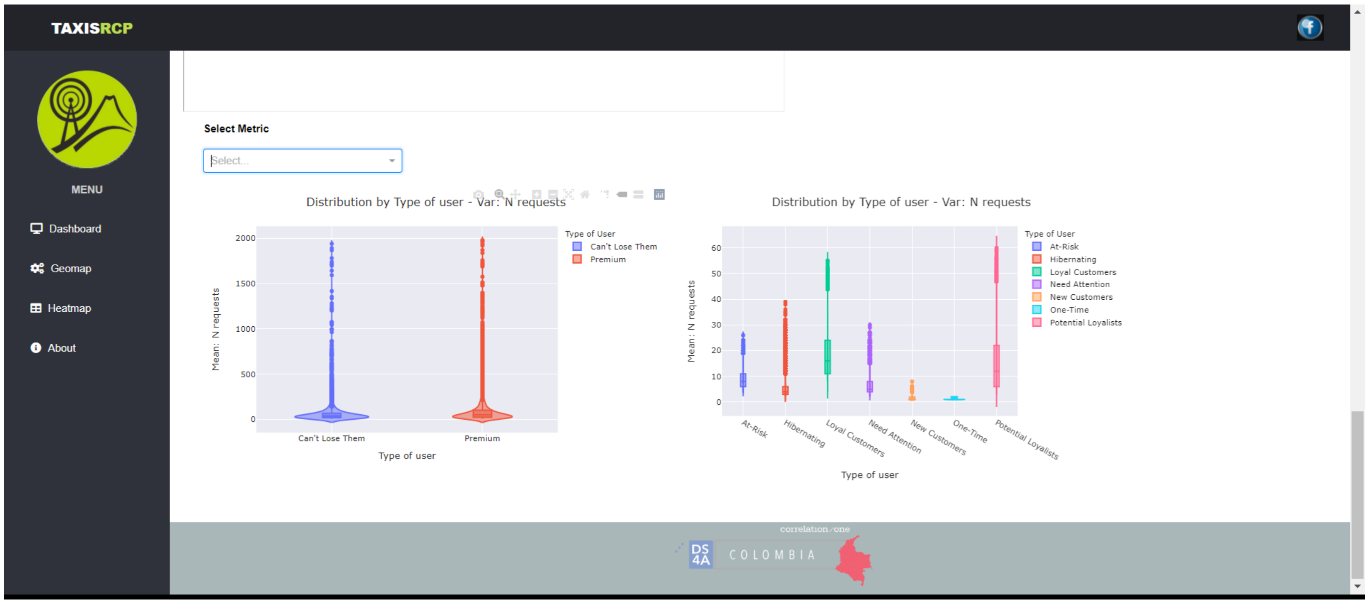
This module shows a client segmentation based on customer activity's freshness (Time since last activity) and the customers' total number of requests.

Identify a segment of customers:

- **One-time:** *50%* are one-time service customer.
- **Who are their best customers?** *4%* are Premium: Recently have requested a service, and they are the most often customers. *5%* of your customers are Loyal.
- **Who has the potential to become valuable customers?* 3%* are Potential Loyalist, they are recent customers with average frequency.
- **New Customers** Start building relationships with these customers by providing onboarding support. *4%* are new customers.
- **At-Risk Customers** *3%* of your customers are at risk to churn. They request a service often but have not required a service recently.
- **Can't Lose Them** They are similar to Premium customers but haven't been requesting a service recently. *7%* of your customers are in this segment.
- **Need Attention** *5%* of your customers you need incentivizing to use your service.
- **Hibernating**: *20%* of your customers contribute to your churn rate.

You can graph some metrics to see the differences in each segment.

You can also graph some metrics like Recency: Time since last activity, number of total requests by customer, average monthly requests, and months seniority for each segment.



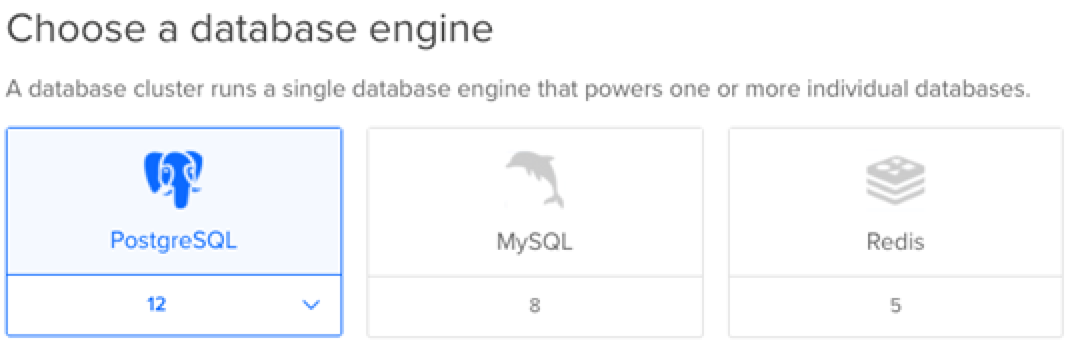
You can explore our application at the following link:

[Pasto City Taxi.]()

7 Relational Database

We create PostgreSQL Database Cluster at service Managed Databases of DigitalOcean.

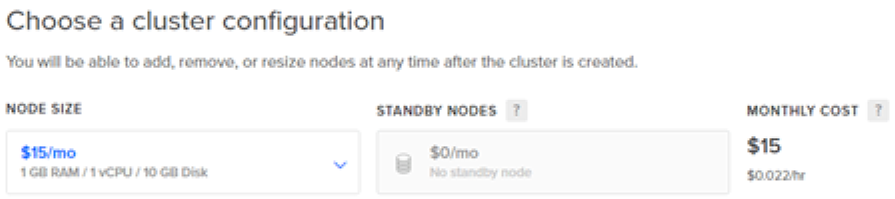
1. In the choose a database engine section, we chose PostgreSQL version 11.



Choose a database engine

A database cluster runs a single database engine that powers one or more individual databases.

2. In the choose a cluster configuration section, we specify the number and size of the database nodes.

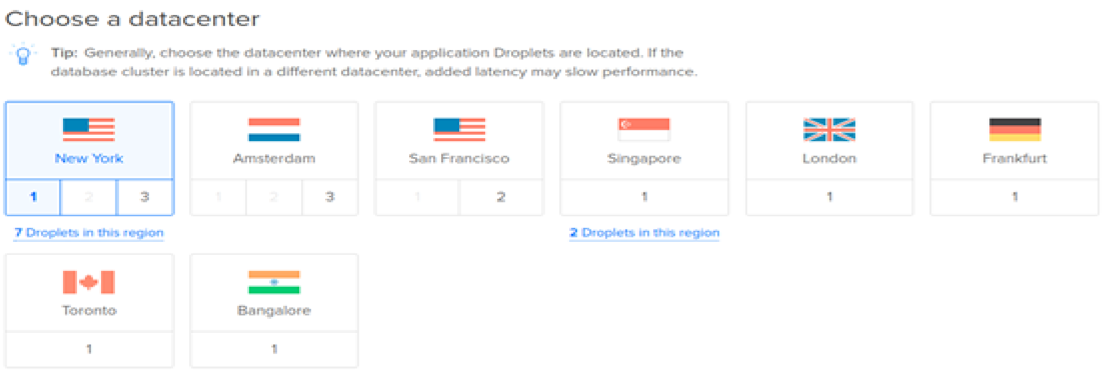


Choose a cluster configuration

You will be able to add, remove, or resize nodes at any time after the cluster is created.

NODE SIZE	STANDBY NODES ?	MONTHLY COST ?
$15/mo 1 GB RAM / 1 vCPU / 10 GB Disk	$0/mo No standby node	$15 $0.022/hr

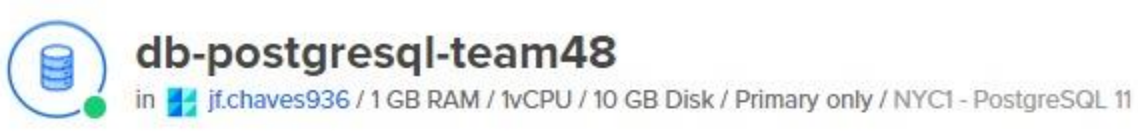
3. In the Choose a datacenter section, we select New York like the region for the database cluster. In this region we also have the Droplets server.



Choose a datacenter

Tip: Generally, choose the datacenter where your application Droplets are located. If the database cluster is located in a different datacenter, added latency may slow performance.

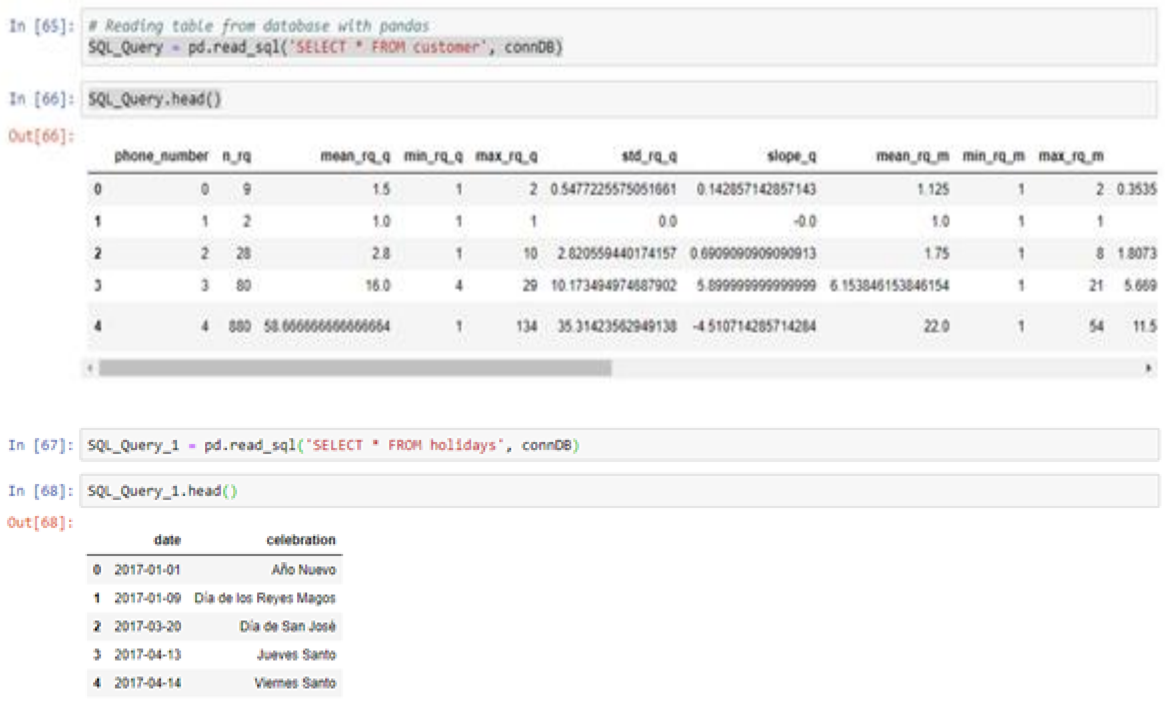
4. In the last section, Finalize and Create, we chose the name for the cluster, the project to add it to, and any tags we use.



db-postgresql-team48

in jf.chaves936 / 1 GB RAM / 1vCPU / 10 GB Disk / Primary only / NYC1 - PostgreSQL 11

5. We configure the connection details, such as username, password, host, port and database.
6. To import the databases "customer" and "holidays", we use the psycopg2 package.



```
In [65]: # Reading table from database with pandas
         SQL_Query = pd.read_sql('SELECT * FROM customer', connDB)

In [66]: SQL_Query.head()

Out[66]:
```

	phone_number	n_rq	mean_rq_q	min_rq_q	max_rq_q	std_rq_q	slope_q	mean_rq_m	min_rq_m	max_rq_m	
0	0	9	1.5	1	2	0.5477225575051661	0.142857142857143	1.125	1	2	0.3535
1	1	2	1.0	1	1	0.0	-0.0	1.0	1	1	
2	2	28	2.8	1	10	2.820559440174157	0.6909090909090913	1.75	1	8	1.8073
3	3	80	16.0	4	29	10.173494974687902	5.899999999999999	6.153846153846154	1	21	5.669
4	4	880	58.666666666666664	1	134	35.31423562949138	-4.510714285714284	22.0	1	54	11.5

```
In [67]: SQL_Query_1 = pd.read_sql('SELECT * FROM holidays', connDB)

In [68]: SQL_Query_1.head()

Out[68]:
```

	date	celebration
0	2017-01-01	Año Nuevo
1	2017-01-09	Día de los Reyes Magos
2	2017-03-20	Día de San José
3	2017-04-13	Jueves Santo
4	2017-04-14	Viernes Santo

Bibliography

Che, F., Jia, J., Zhang, H., & Zhang, L. (2020). Revealing urban traffic demand by constructing dynamic networks with taxi trajectory data. *IEEE Access*.

Chen, C., Huang, H., Xiang, C., & Zhao, J. (2020). Unifying Uber and taxi data via deep models for taxi passenger demand prediction. *Personal and Ubiquitous Computing*.

Chen, H., Li, Y., Liu, Z., & Zhang, Q. (2020). Taxi Demand Prediction Based on a Combination Forecasting Model in Hotspots. *Journal of advanced transportation*.

Kalakou, S., Martins, A., Moura, F., & Rodrigues, P. (2020). Spatiotemporal Variation of Taxi Demand. *Transportation Research Procedia*.

Khryashchev, D., & Thanh Huy, V. (2019). Predicting Taxi and Uber Demand in Cities: Approaching the Limit of Predictability. *IEEE Transactions on Knowledge and Data Engineering*.

Sun, L., Wang, X., Zhang, C., & Zhu, F. (2020). Taxi Demand Prediction Using Parallel Multi-Task Learning Model. *IEEE Transactions on Intelligent Transportation Systems*.

Wu, C., Wu, H., Xiang, L., & Yan, J. (2020). CITY-SCALE TAXI DEMAND PREDICTION USING MULTISOURCE URBAN GEOSPATIAL DATA. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Wu, Z. (2020). A novel dynamically adjusted regressor chain for taxi demand prediction. *2020 International Joint Conference on Neural Networks*.