KSCHOOL

JUNE 2022

PREDICTION AND ANALYSIS OF BICIMAD SERVICE DEMAND

Trabajo de Fin de Máster
Máster en Data Science

AUTOR:
Paula Martín Moreno

Content

1. Introduction

Bicimad is the public electric bicycle system of the city of Madrid, managed by the EMT of Madrid. With Bicimad people can rent a bike from one location and return it in a different station.

One of the main problems of these businesses of bike sharing is being able to predict the bike demand on any day. Because having excess of bikes result in wastage of resource, and also underestimating the demand result in lower revenues (missing immediate customers because there are no bikes available and missing long-term customers due to dissatisfied clients. So being able to predict the demand of the service is critical for the efficient functioning of the business.

The goal of this project is to provide an interactive platform that helps Bicimad in their decision making in the short term: on the one hand, predicting the demand for the next days help them to identify peak hours of demand and prepared them to restructure bicycles by zone according to predictions, and on the other hand, understanding the patterns in demand, both globally and by areas of Madrid, to understand which are the days/months/times of lower demand, where the business can dedicate to improve the service (maintenance of the bikes, improvement of the stations...) and the days/months/times with the highest demand to be sure that the demand that will occur will be covered and that the bikes are available in the areas where they are needed.

It also helps the decision making in the long-term, to identify the areas with the highest demand where to expand supply, and to assist in the expansion of service to areas of Madrid that are close to the areas with the highest demand and to make decisions to close stations that are not profitable (low demand).

We have information from 2019 to June 2021, so the development of the project will consist of predicting the demand that will occur from July 2021 onwards.

In order to put the application into production it would be necessary to run the models with all the information available up to the current day every 24 hours, in order to have the new forecasts for the next 14 days.

The github repository where the project is structured in three folders:

1. Data: in it are all the files that are needed to run the notebooks (either raw data or files that have been obtained from these during the development of the notebooks).

2. Notebooks: the project has been done with python, in jupyter notebook, with a total of 4 notebooks.
    - 1. Exploring and preparing the data: analysis of the raw data and processing and cleaning to prepare it for the project.
    - 2. Exploratory data analysis: using the dataset obtained in the first notebook to analyze the data and obtain relevant information to be used in the models.
    - 3. Models for global demand: machine learning models are developed to predict the demand for bicycles at an aggregate level.

- 4. Models demand per zone: machine learning models to predict the demand of the service by area of Madrid.

3. Frontend: a frontend platform has been developed with Streamlit, everything needed for the frontend is in this folder (code with the application, files used, library requirements).

In order to run the project, you first need to download all the datasets available in the folder of Data and save them in the same folder as the jupyter notebooks. Some datasets are downloaded via a google drive link, due to the size of these files

2. Data collection and description

The data used for the project is obtained from the open source portal data of the community of Madrid: https://opendata.emtmadrid.es/Home

We obtained a dataset about the use of the electric bicycle service and a dataset with the stations location and situation by day and hour.

2.1. Usage of the service

The files contain anonymized information about the use of electric bicycle service, each month is in one file and we need to merge all the files to obtain the final dataset (this is performed in the first notebook). All the files are located in the folder data, there is a link to google drive from which can be downloaded.

The columns of the dataframe are:

__id: identificator of the movement

user_day_code: user code. For a same date, all the movement of the sale user have the same code.

idunplug_station: number of the origin station

idunplug_base: number of origin base

idplug_station: number of destination station

idplug_base: number of destionation base

unplug_hourTime: time the bicycle is picked up

travel_time: time in seconds of the ride

track: detail of the ride (GeoJSON)

user_type: type of user: (0: not defined, 1: anual user, 2: ocasional user, 3: worker of bicimad)

ageRange: age range of the user: (0: not defined, 1: between 0 and 16 years, 2: between 17 and 18 years, 3: between 19 and 26 years, 4: between 27 and 40 years, 5: between 41 and 65 years, 6: more than 65 years

zip_code: postal code of the user

We are interested in the columns: __id, used to count the number of rides per hour, idunplug_station: used to identify the station where the bike is taken, unplug_hourTime, this column provides the hour and day of the pickup, user:type: we are only interested in the annual users and occasional users to predict the demand, not taking into account the rides made by workers of bicimad.

We also filtered the initial dataset, to remove the trips with less than 2 minutes of duration and trips of more than 4 hours, considering that these trips are wrong in the table due to errors in the data collection or technical failures of the bicycles.

2.2. Stations

We are interested in predicting the demand of bicycles per area. We use this dataframe to obtain information about the stations. The file is .json, id located in the 'links to drive' file, and has the following columns:

id: code of the base station

latitude: latitude of the station in WGS84 format

Longitude: lenght of the station in WGS84 format

name: name of the station

light: ocupation of the station (0=low, 1=medium, 2=high, 3=inactive)

number: logical designation of the base station

actívate: active station (0=Non active, 1=active)

no_available: station availability (0=available, 1=not available)

total_bases: number of bases of the station

dock_bikes: number of docked bicycles

free_bases: number of free bases

reservations_count: number of active reservations

From this dataset we are interested in obtaining the name and coordinates of the stations, the address and the total bases available.

2.3. Holidays

In this file we have information on public holidays in the Community of Madrid from 2013 to 2022. The columns are:

day: day in date format

day_week: day of the week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)

holiday: weekday / holiday / Sunday holiday

type of holiday:national holiday, local holiday of the Community, Community of Madrid holiday

festivity: name of the festivity

From this dataset we are interested in knowing the dates of festivities in Madrid, so we can study the impact of festivities in demand.

2.4. Transformations and final datasets for models

From the raw data presented in sections 2.1, 2.2. and 2.3. we have obtained two dataframes we will use in the following notebooks: movements_grouped.csv and rides_per_station.csv.

In the first file we have the information of the demand per hour at an aggregated level, that we will use in the notebook 3, and in the second file we have the same information but grouped by station, to use it in the notebook 4.

For the second file, we need to obtain the postal code of the stations, since we are going to segment the area according to the postal code, for this we have used the python geopy library. In this way, for each station, from the latitude and longitude we obtain the address, from where the zip code is extracted for each station.

3. Methodology: machine learning techniques used, statistical methodologies

The figure 1 below shows the methodology used during the development of the project. Each step will be explained in detail throughout this section.

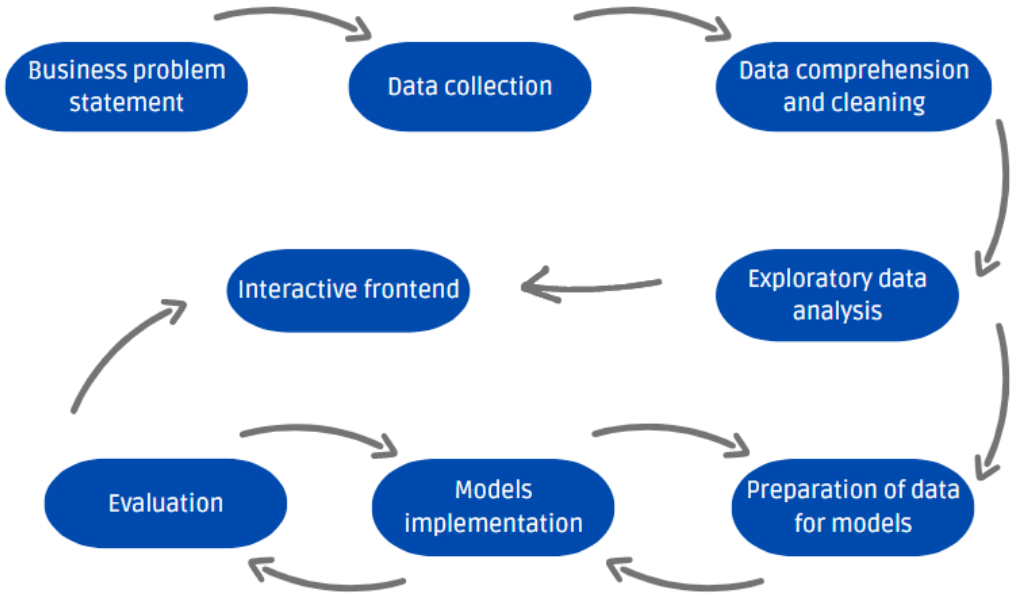

Figure 1: Methodology followed in the project

The first step was to define the business problem to be solved, then collect the necessary data and proceed to the cleaning and understanding of the data to keep the relevant information.

Before starting with the development of the models to predict the demand, a preliminary analysis of the data (EDA) is performed, both of the evolution of the demand, to analyze the time series and identify patterns in the demand, and of the available Bicimad stations, since the objective is to predict the demand by zones.

Subsequently, machine learning models have been developed to predict the demand for the next 14 days, using different models that will be explained later.

Finally, an interactive frontend platform has been developed, where the results obtained can be observed and whose objective is to help Bicimad in its decision making and demand planning.

3.1. Exploratory Data Analysis (EDA).

This section presents the most relevant results of the exploratory analysis.

3.1.1. EDA demand

First, as can be seen in the figure 2 below, demand patterns vary differently depending on whether the day is a weekend (blue color) or a weekday (red color).

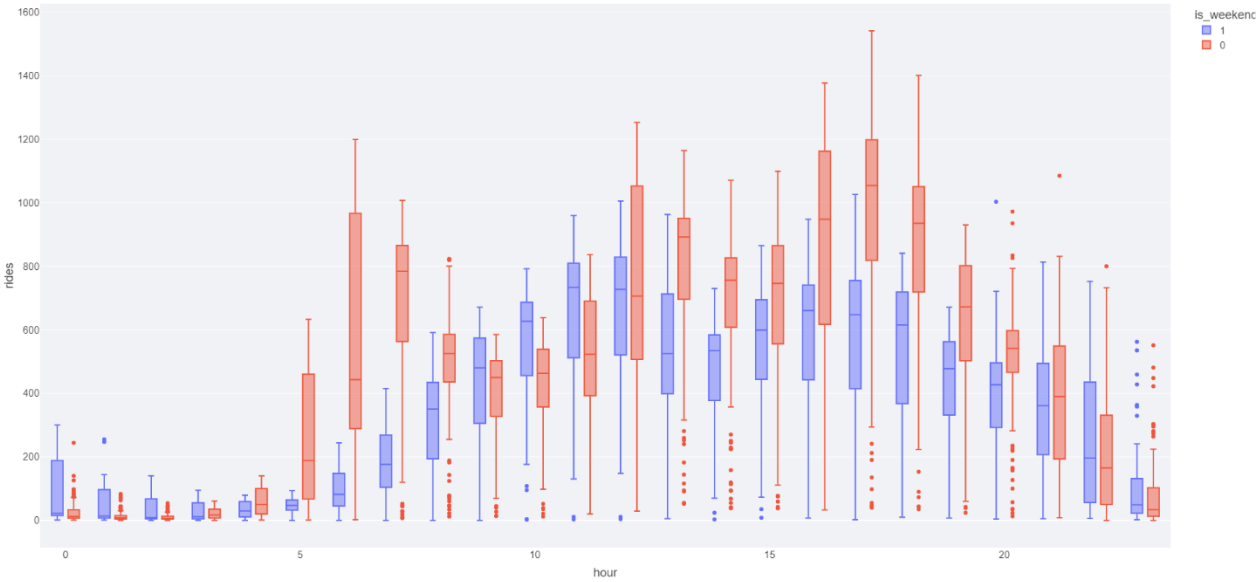

Figure 2: Distribution of the demand per hour in weekdays and weekends

The average demand is higher on weekdays than on weekends, and on weekdays the hours with the highest demand are between 16:00 and 18:00, and there is also a high demand between 7:00 and 8:00 a.m. and 12:00 a.m. While on weekends, hourly demand does not vary as markedly as on weekdays, the hours of highest demand are between 11 and 12 am, and there is a higher demand between 00 and 04 am (compared to the demand at the same time during the week).

These patterns indicate that the service is mostly used on weekdays, with users using it for commuting to and from work (peak demand coincides with the beginning and end of the workday).

There are two patterns of demand clearly differentiated. Red shows the demand per hour on weekdays, as can be seen, the hours with the highest demand are between 6 and 7 a.m. and between 4 and 5 p.m., and there is also a high demand at 12 and 18 o'clock.

On the other hand, blue represents the demand by hours on weekends, which stands out because it has a higher demand between 12 and 3 a.m., which can be explained because the service is used when the subway is closed.

As for the analysis of average demand by month, the months with the highest demand are September and October, followed by May and June. This indicates that demand is affected by the weather, with higher demand in the months with better weather, except for July and August, which coincide with the vacation period.

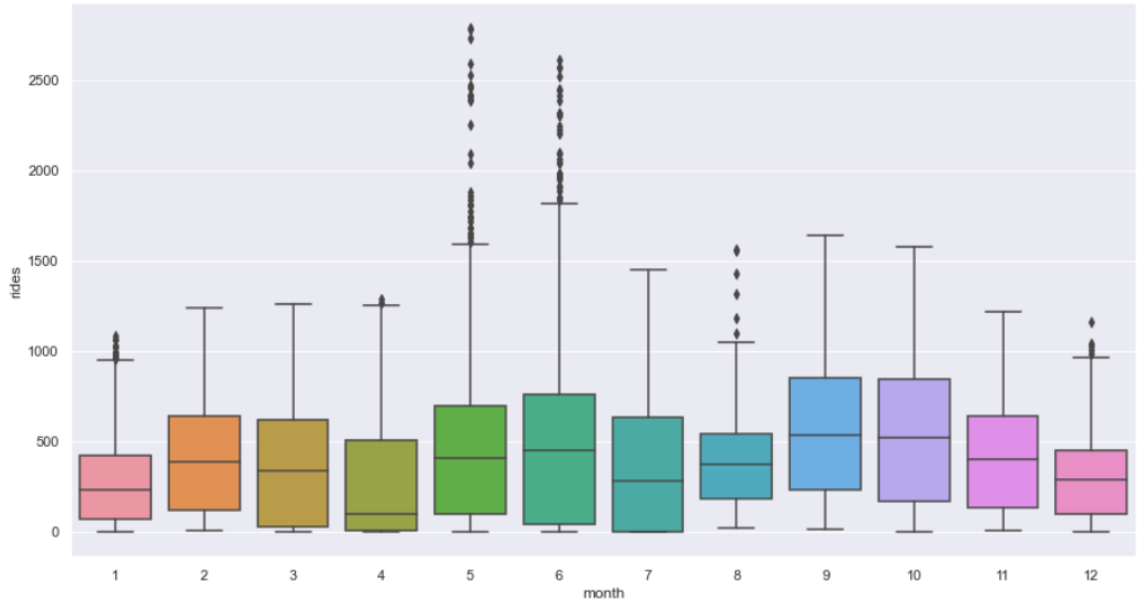

Figure 3: Distribution of the demand per months

In January, May, June, August and December there are a lot of outliers, this can be explained with the changes in the weather in the months of December and January, because of variable distribution across the day.

The outliers in May and April can be explained by the covid-19 situation.

From the correlation matrix represented in figure 4, we can see the correlation coefficients between the variables with our variable of interest (rides).

The variables that are more correlated with our target variable are the lag_1 (demand one day before at the same hour) and lag_7 (demand 7 days before at the same hour). There are also highly correlated the rest of the lag variables, and to a lesser extent the time and month. The variable of interest (rides) has no correlation with the variables is holiday (indicating whether the day is a holiday or not), year, day and is_weekday (indicating whether the day is weekend or not).

We will take this into account to choose exogenous variables to be used as predictors in the models.

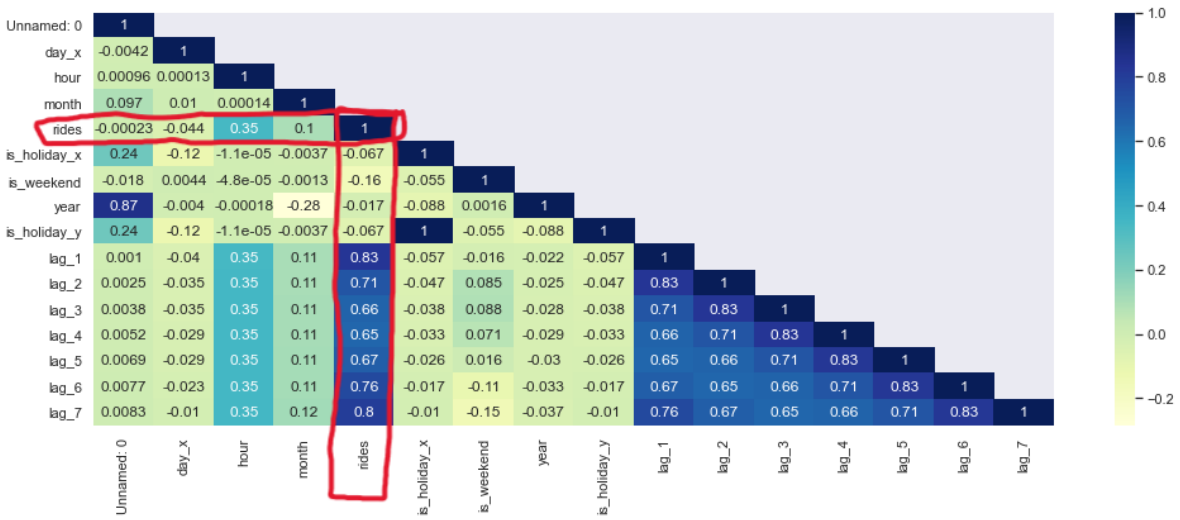

Figure 4: Correlation matrix of the demand (rides) variable

3.1.2. EDA stations

We are interested in predicting the demand of the service per zone of Madrid, to identify the different zones we divided the stations per postal code.

From the analysis we identified that there are 264 stations of the service of Bicimad, located in 29 postal codes.

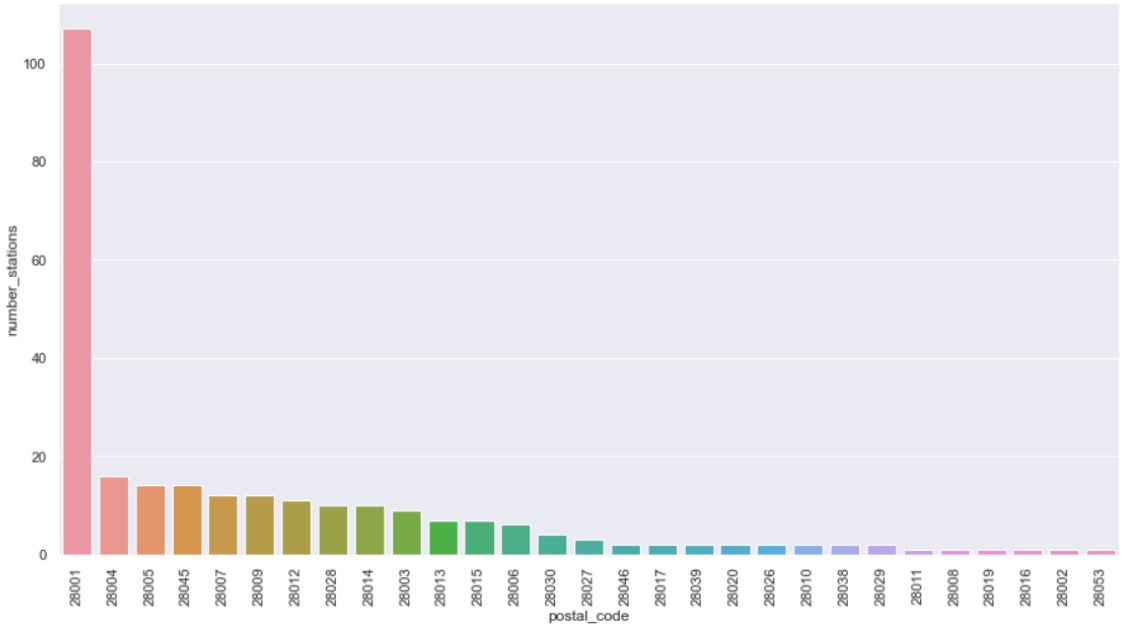

Figure 5: Number of stations per postal code of Madrid

As can be seen in figure 5, the 63% of stations of Bicimad are located in the postal code 28001, which belongs to the center of Madrid. In order to develop the prediction models, this area will be segmented into different zones, so that the prediction is of a more specific area, for this purpose a cluster analysis will be performed, which will be explained later.

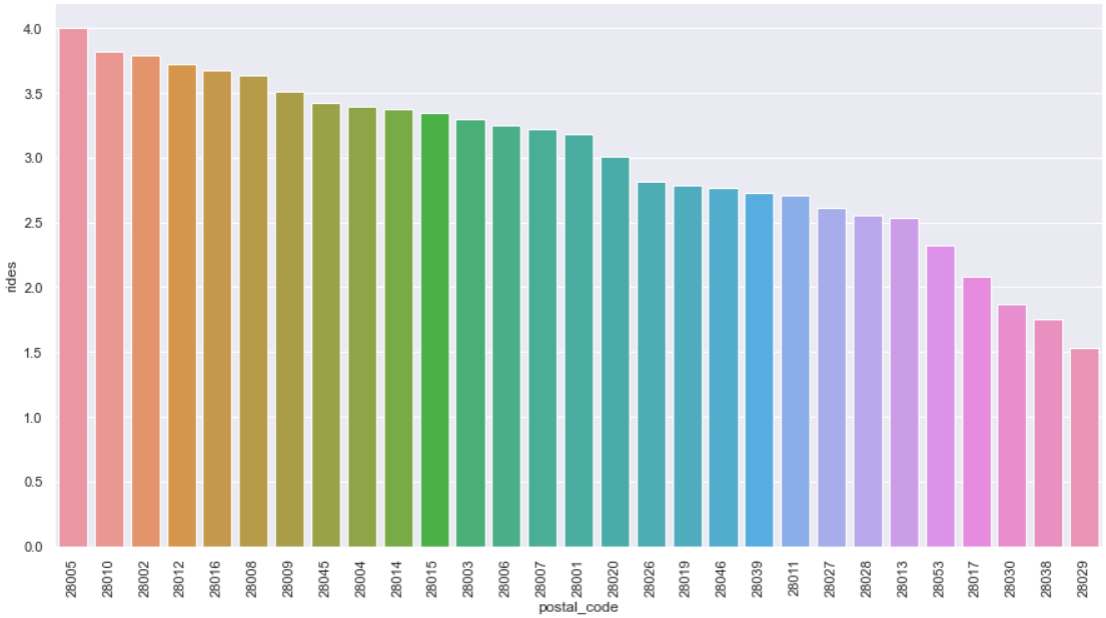

Figure 6: average demand per postal code

Figure 6 shows that the higher average demand occurs in the postal code 28005, followed by the postal code 28010 and 28002. While the lower average demand occurs in the postal code 28029.

3.2. Modeling

3.2.1. Definition of the time series problem

Demand forecasting is the process of using predictive analysis of historical data to estimate the future demand. To model the demand of the service of Bicimad we used time series forecasting, that consists in predict the future by analyzing the trends in the past, so we use historical data to predict future values.

There are 4 main components that define a Time series:

1. Trend: upward and downward movement of the demand over a period of time.

2. Seasonality: seasonal variances of the data.

3. Residuals: noise - random variations.

4. Cyclicity: the behavior is repeated after an interval of time.

To understand if we are facing a time series problem and understand the properties of the time series, we analyze the autocorrelation plot. The autocorrelation plot (ACP) represents the correlation between two observations at different points. Past values influence in the current values if correlations are present.

This plot helps us uncover hidden patterns in our data and select the correct forecasting methods, it is also useful to identify the seasonality

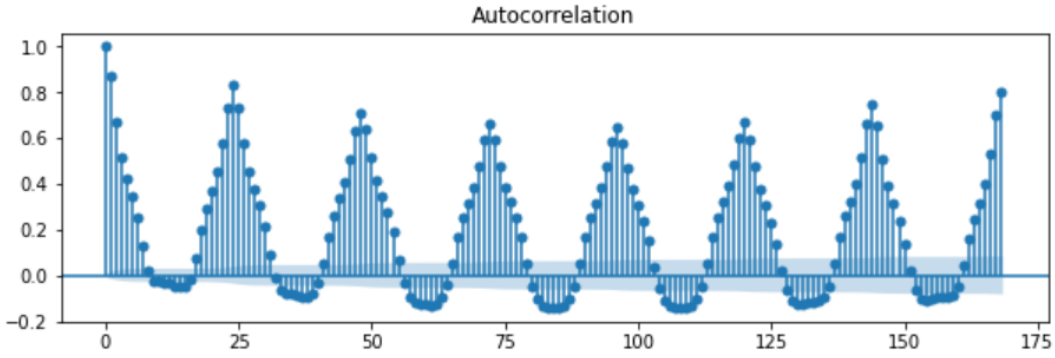

Figure 7: Autocorrelation plot of the variable 'rides' with 168 lags

We have the hourly data, so each lag represents one hour, in the figure 7 there are represented 168 lags, so, 7 days.

From the figure 7 we can infer that the current value of the time series is highly correlated with the lag 24 (demand at the same time one day before) and lag 164 (demand at the same time 7 days before). There is also high correlation with demand each 24 hours, this indicates that the hour is a quite important feature to predict the demand.

Another thing we must test in a time series analysis is the stationarity of the data. When forecasting or predicting the future, most time series models assume that each point is independent of one another, and stationarity in time series proves that the observations are independent.

To test stationarity, we performed the Dickey-Fuller test: this test is used to try the null hypothesis: time-series data is non-stationary. It calculates the p-value and compare it with a threshold value of 0.05. If the p-value is lower than this level, data is stationary.

```
Test statistic          -8.521028e+00
p-value                  4.488707e-12
Number of lags           4.700000e+01
Number observations      2.184100e+04
Critical value (1%)     -3.959185e+00
Critical value (5%)     -3.410691e+00
Critical value (10%)    -3.127168e+00
dtype: float64
```

Figure 8: Results of the Dickey-Fuller test

From the results in figure 8 we can conclude that the time series is stationary.

3.2.2. Feature engineering

We added lag values, that we used as predictors in the machine learning models. We consider initially the following lag variables

- Rides_lag_7: demand the same hour 7 days before
- Rides_lag_1: demand the same hour 1 day before
- Rides_lag_2: demand the same hour 2 days before
- Rides _lag:1_month: demand the same hour 1 day before

In the figure 4 we observed that the variable of the demand (rides) is highly correlated with the lag values.

The variable we want to predict is highly correlated with the lags 7,1,2, 1 month and with the hour of the day. So we will use some of this variables as predictors.

3.2.3. Evaluation of the models

To forecast the demand of bicycles we tried different models, and we measured the error in different ways. A forecast error is the difference between an observed value and its forecast. We can measure forecast accuracy in different ways, for this project we used the following metrics:

- MSE “minimum squared error”: this metric is obtained by calculating the square of each forecast error and taking the average of the result. A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite. A smaller MSE is preferred because it indicates that the data points are dispersed closely around its mean. It reflects the centralized distribution of your data values, the fact that it is not skewed, and, that it has fewer errors (errors measured by the dispersion of the data points from its mean).

- MAE “mean absolute error”: is a way of measuring the accuracy in forecasting, is obtained as the mean of the absolute value of the difference between the forecasted value and the actual value. This metric is useful because is easy of understand and compute.

- RMSE “root mean square error”: this metric gives higher importance to the most significant errors. Is calculated as the square root of the mean square of the square of the errors This metric is widely used despite being more difficult to interpret.

We also represented the demand and the predictions in a plot to view the differences in a plot.

We are working with time series data, which means that the order of the data cannot be altered, so we cannot resort to cross-validation to evaluate the models. However, we will resort to the process of backtesting, which consists of evaluating the performance of a predictive model by applying it retrospectively on historical data. It is therefore a validation strategy that allows the predictive capacity of a model to be quantified.

In the context of time-series forecasting, the notion of backtesting refers to the process of assessing the accuracy of a forecasting method using existing historical data. The process is typically iterative and repeated over multiple dates present in the historical data. Backtesting is used to estimate the expected future accuracy of a forecasting method, which is useful to assess which forecasting model should be considered as most accurate.

The backtesting process starts by selecting a list of threshold dates within a time span covered by the historical data. In the figure 9, the thresholds are noted T1, T2, T3 and T4.

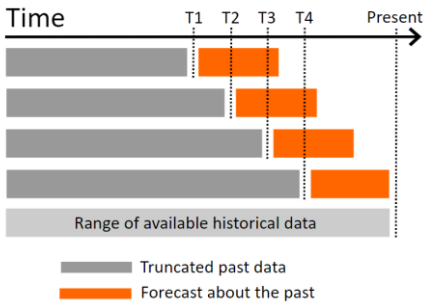

Figure 9: Backtesting process for evaluating models

For each threshold, the historical data is truncated at the threshold, the forecasting model is trained and applied on the truncated data, the forecasts are compared with the original untruncated data. Finally, an average forecast error is established over all the thresholds. This averaged error can be interpreted as an estimation of the error that will be associated with the model when producing true forecasts (about the future).

3.2.4. Models used

Machine learning forecasting proved to be the most effective in capturing the patterns in the sequence of both structured and unstructured data and its further time series analysis forecasting. Machine learning is useful to short/mid-term planning.

For the machine learning models we used as predictors the lag values of the time series, and the hour, month and weekday.

3.2.4.1. Naïve model

We created a naïve model that serve us as a baseline to compare it with the machine learning models. In this model we assume that the demand is the same as the demand at the same hour 7 days before (we have demonstrated that this is the lag that has the highest autocorrelation with the demand variable).

3.2.4.2. ARIMA

ARIMA stands for Autoregressive Integrated Moving Average Model. This model explains a given time series based on its own past values (its own lags and the lagged forecast errors). ARIMA Models are specified by three order parameters: (p, d, q):

- p is the order of the AR term: a regression model that utilizes the dependent relationship between a current observation and observations over a previous period. An auto regressive (AR(p)) component refers to the use of past values in the regression equation for the time series. Refers to the number of lags to be used as predictors

- q is the order of the MA term: a model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average component depicts the error of the model as a combination of previous error terms. The order q represents the number of terms to be included in the model.

- d is the number of differencing required to make the time series stationary: uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the

subtraction of the current values of a series with its previous values d number of times. d is the minimun number of differenciating needed to make the series stationary.

3.2.4.3. Random Forest

Random Forest is an ensemble of decision trees algorithms that can be used for classification and regression predictive modeling.

Random forests build a bunch of decision trees independently. Each decision tree is a simple predictor, but their results are aggregated into a single result. Since a random forest is an ensemble of decision trees, it has a lower variance than other algorithms and can produce better results.

To implement the random forest we use the scikitlearn library. To develop the model we use 'One hot encoding' to encode the categorical variables of the model and use them in the models (weekday, month, hour).

3.2.4.4. Gradient Boosting

Models based on gradient boosting have demonstrated in the last years to be competitives in time series forecasting. They are an ensemble of decision trees where new trees fix errors of the previous trees of the model.

Their main advantages compared to other forecasting models (such as ARIMA) are easiness of including exogenous variables, incorporating non-linear behaviors in the model, high scalability, useful with large volumes of data.

Gradient boosting is useful because of the little training time and little expertise for parameter tuning.

To implement the models we used the scikitlearn library, with the following regressors:

- XGBoost (Extreme Gradient Boosting): uses a presorted algorithm and histogram-based algorithm for comuting the best splits.

- CatBoost: builds symmetric (balanced) trees, unlike XGBoost and LightGBM. In every step, leaves from the previous tree are split using the same condition.

- LightGBM: uses a novel technique of Gradient-base One-Side Sample to filter the data instances for finding a split value

3.2.5. Procedure followed in the machine learning models

For the different models we tried we followed the same procedure, summarized in figure 10. First, we prepared the dataset to obtain the predictors we want to use for each model, then we divided the dataset in train (to train the models) and test (to test the results of the model), using the last 14 days data as the test set, for each model we perform a grid search to find the best combination of hyperparameters. Then we fit the model with the train set and predict the forecast for the next 14 days. Once we have the predictions, we compare them with the test set, calculating the metrics and plotting both curves.

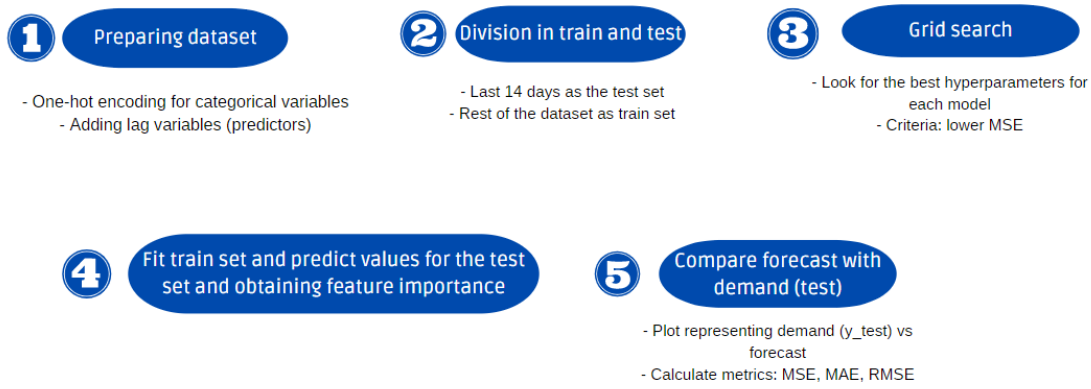

Figure 10: Procedure followed for each model

Once we've made this procedure for each model, we compare the results obtained and select the model with the best metrics (lower errors and higher proximity of demand and forecast curves).

4. Summary of main results

The previous section explained the models and metrics used to develop this project. This section will show the main results obtained, both for the global demand and for the demand by zones.

4.1. Prediction of global demand

In this section we are going to present the main results obtained in the notebook “3. Models for global demand”. To represent the results, we plotted the demand (variable rides in the test set) and the predictions (obtained by the different models tested). And we expose the metrics obtained.

4.1.1. Naïve model

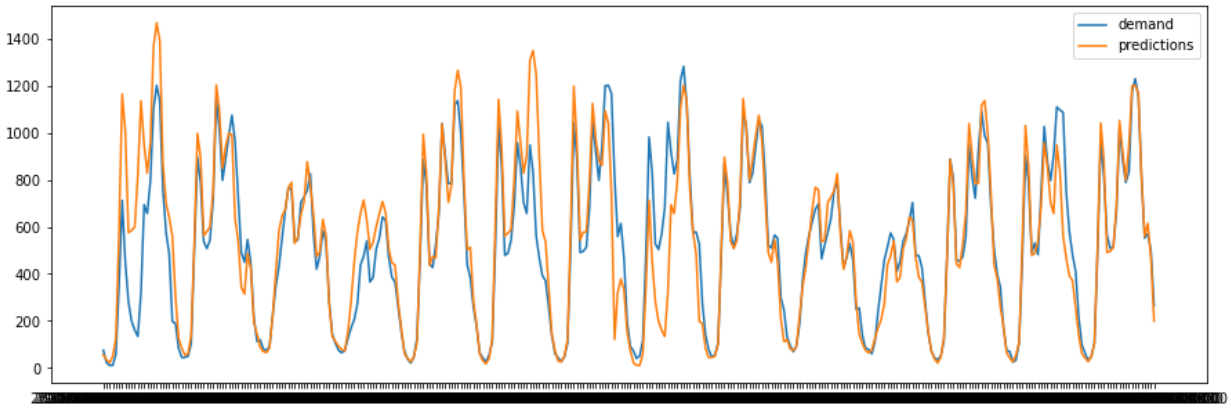

Figure 11: Predictions and demand of the naive model

Table 1: Metrics for the naive model

Metric	*Value*
MSE	24746.33
MAE	92.88
RMSE	157.3

This is the model we are going to use to compare it with the models we are going to build, which must improve this metrics.

As we can see in the figure 11 and table 1 the results of the naïve model are good, which means that the variable: rides_lag_7 is quite important to predict the demand of the service.

4.1.2. ARIMA model

To determine the coefficients of the ARIMA model we followed the following criteria:

- d=0: differentiating is not necessary since data is stationary (as we proved with the Dickey-Fuller test)
- p = 24: to estimate p we use the autocorrelation plot and find the most significant lag in the partial autocorrelation plot. The lag with the higher autocorrelation is lag 24, as can be seen in the figure 12.

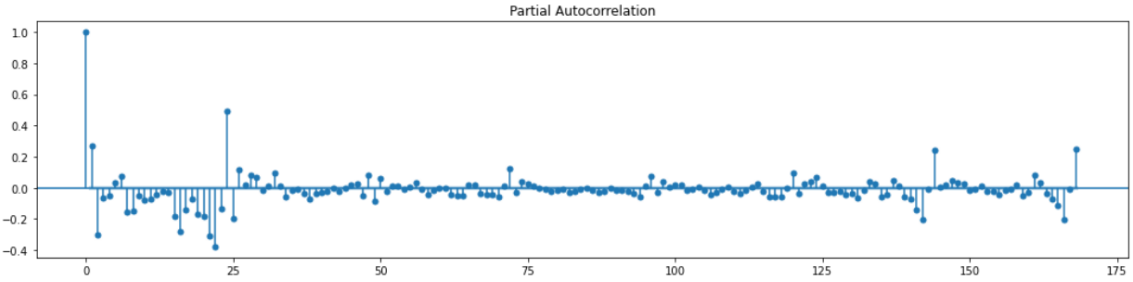

Figure 12: Partial autocorrelation graph for 168 lags

- q= 6: to estimate the q we use the autocorrelation plot, taking into account the number of lags crossing the threshold. We select the maximum lag value over the blue band (confidence interval), from the figure 13 we select the lag 7.

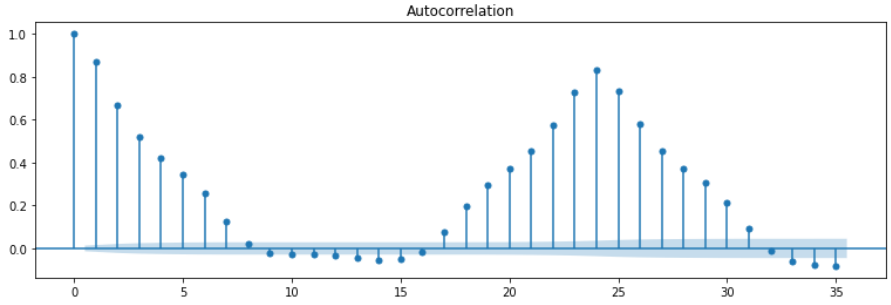

Figure 13: Autocorrelation plot used to estimate the parameter q

From the summary of the results, that is shown in the figure 14, the terms with higher significance of the models are ar.L24 (autorregresive lag 24 hours) and ma.L1.rides and ma.L2 rides. All the lags with the p-value above 0.05 are not significant for the model

	coef	std err	z	P>|z|	[0.025	0.975]
const	396.6481	25.417	15.606	0.000	346.832	446.464
ar.L1.rides	0.0473	0.008	6.215	0.000	0.032	0.062
ar.L2.rides	0.0020	0.007	0.270	0.787	-0.012	0.016
ar.L3.rides	0.0058	0.007	0.790	0.430	-0.009	0.020
ar.L4.rides	0.0320	0.007	4.454	0.000	0.018	0.046
ar.L5.rides	0.0333	0.007	4.911	0.000	0.020	0.047
ar.L6.rides	0.0455	0.007	6.213	0.000	0.031	0.060
ar.L7.rides	-0.0154	0.007	-2.067	0.039	-0.030	-0.001
ar.L8.rides	-0.0044	0.007	-0.587	0.557	-0.019	0.010
ar.L9.rides	-0.0501	0.005	-9.206	0.000	-0.061	-0.039
ar.L10.rides	0.0280	0.005	5.440	0.000	0.018	0.038
ar.L11.rides	0.0143	0.005	2.849	0.004	0.004	0.024
ar.L12.rides	0.0113	0.005	2.221	0.026	0.001	0.021
ar.L13.rides	-0.0051	0.005	-1.012	0.312	-0.015	0.005
ar.L14.rides	-0.0081	0.005	-1.595	0.111	-0.018	0.002
ar.L15.rides	0.0004	0.005	0.087	0.931	-0.009	0.010
ar.L16.rides	-0.0348	0.005	-7.017	0.000	-0.045	-0.025
ar.L17.rides	-0.0454	0.005	-9.204	0.000	-0.055	-0.036
ar.L18.rides	0.0142	0.005	2.857	0.004	0.004	0.024
ar.L19.rides	0.0222	0.005	4.458	0.000	0.012	0.032
ar.L20.rides	0.0320	0.005	6.405	0.000	0.022	0.042
ar.L21.rides	0.0199	0.005	3.953	0.000	0.010	0.030
ar.L22.rides	0.0127	0.005	2.520	0.012	0.003	0.023
ar.L23.rides	0.0367	0.005	7.355	0.000	0.027	0.046
ar.L24.rides	0.6996	0.005	139.862	0.000	0.690	0.709
ma.L1.rides	1.0460	0.011	98.455	0.000	1.025	1.067
ma.L2.rides	0.6951	0.014	49.129	0.000	0.667	0.723
ma.L3.rides	0.4332	0.015	29.341	0.000	0.404	0.462
ma.L4.rides	0.2338	0.014	16.840	0.000	0.207	0.261
ma.L5.rides	0.0850	0.014	6.174	0.000	0.058	0.112
ma.L6.rides	0.0898	0.012	7.233	0.000	0.065	0.114
ma.L7.rides	0.1358	0.011	12.761	0.000	0.115	0.157

Roots

Figure 14:Summary of the results of the ARIMA model

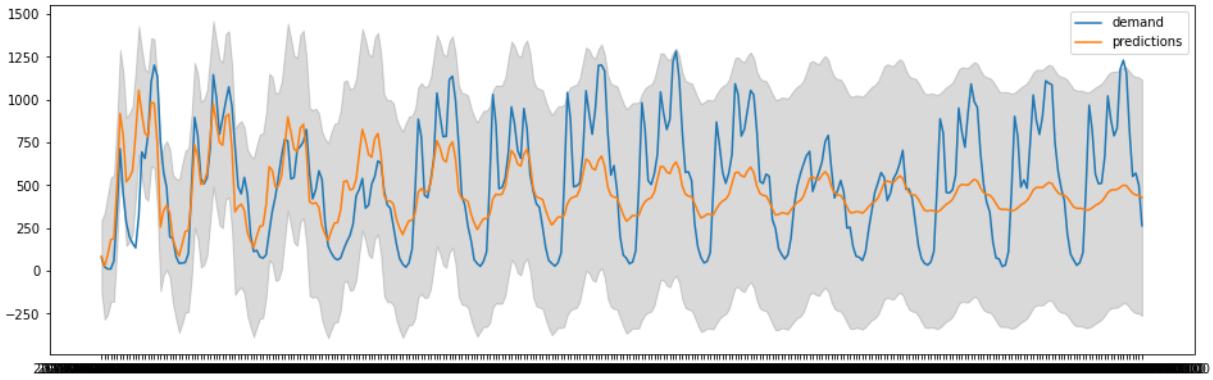

Figure 15: Predictions and demand for the ARIMA model

Table 2: Metrics for the ARIMA model

Metric	*Value*
MSE	71041.26
MAE	213.57
RMSE	266.53

The metrics are worse than the naive model that we have developed before, moreover, in the figure it can be observed that for values close to the initial date the model adapts better than as time progresses, where it is observed that the predictions are increasingly farther away from the real demand and tend towards the average.

4.1.3. Random Forest

- Random Forest without exogenous variables (only lag values)

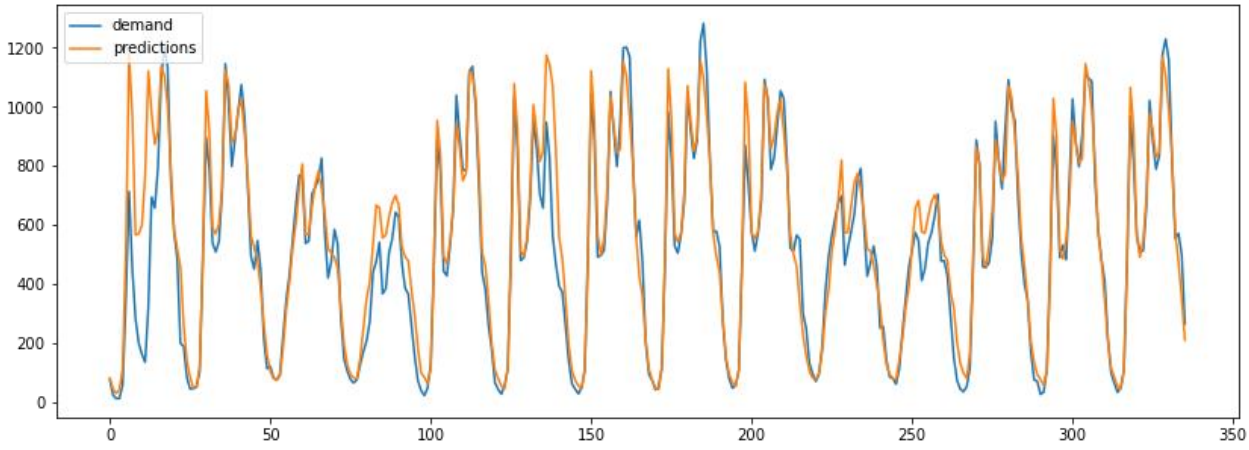

Figure 16: Demand and predictions for the Random Forest model

Table 3: Metrics of the Random Forest model

Metric	*Value*
MSE	13062.95
MAE	69.37
RMSE	114.29

With the Random Forest model we improved the model in a 47% in terms of MSE, in a 25,3% in terms of MAE and in a 27,3% in terms of RMSE.

In the figure 17 we can see the most important features that the model used to make the predictions. As expected, after analyzing the characteristics of the time series, the most important lag variables in the model are lag_1: 1 hour earlier, lag_168: 7 days earlier and lag 24: 24 hours earlier.

:

	feature	importance
0	lag_1	0.651705
167	lag_168	0.161130
23	lag_24	0.101823
166	lag_167	0.008942
22	lag_23	0.005712
143	lag_144	0.005050
25	lag_26	0.003253
24	lag_25	0.002870
95	lag_96	0.002618
1	lag_2	0.002270

Figure 17 : feature importance of lag values for Random Forest model

- Random Forest with exogenous variables

In the previous model we only consider the lag variables, now we added exogenous variables, we performed a one-hot-encoding for each one:

- Weekday
- Hour
- Month

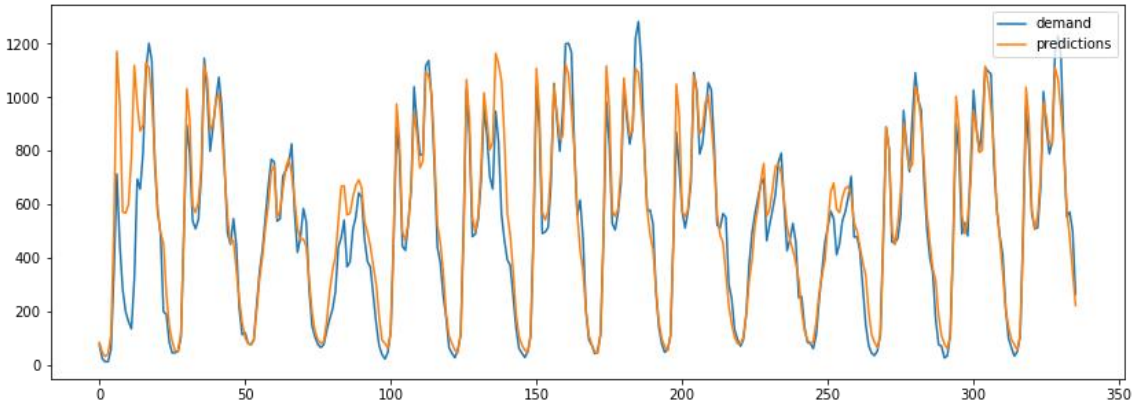

Figure 18: Predictions and demand Random Forest with exogenous variables

Table 4: Metrics for Random Forest with exogenous variables model

Metric	*Value*
MSE	13124.97
MAE	69.81
RMSE	114.56

The results of the model adding the exogenous variables is somewhat worse than the model that only considered the lag variables as predictors, although the results are quite similar. The MSE is 6,8% lower compared to the other Random Forest model.

In the figure 19 are represented the most important features considered in the model. As can be seen, the principals are the same as in the previous model, where no exogenous variables (month, day of the week, time) were considered, which means that the lag variables have a strong influence in predicting demand.

	feature	importance
0	lag_1	0.632706
167	lag_168	0.157104
23	lag_24	0.098341
166	lag_167	0.008294
22	lag_23	0.006080
143	lag_144	0.005179
25	lag_26	0.003603
24	lag_25	0.003275
1	lag_2	0.002782
95	lag_96	0.002705

Figure 19: Feature importance Random Forest with exogenous variables

4.1.4. XGBoost

- XGBoost without exogenous variables (only lag values)

We performed a grid search to find the best combination of hyperparameters (n_estimators, max_depth and learning_rate).

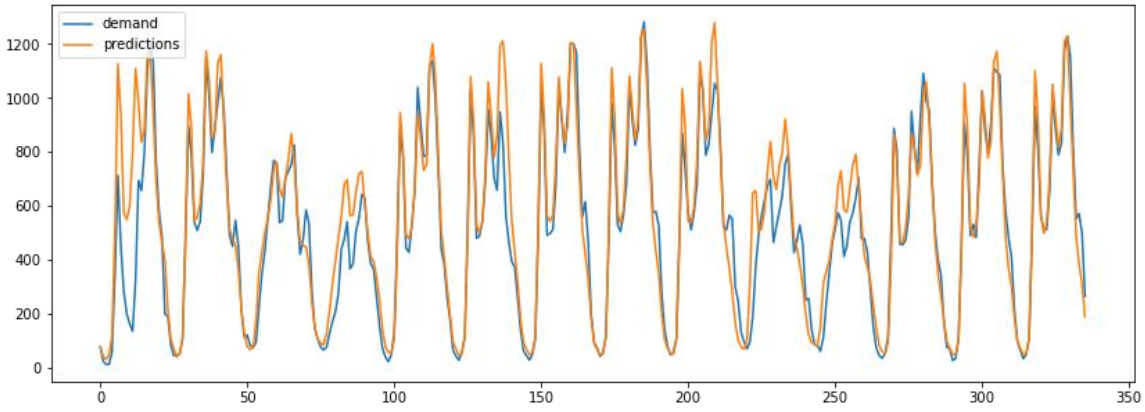

Figure 20: Demand and predictions for XGBoost model

Table 5: Metrics for XGBoost model

Metric	Value
MSE	15484.63
MAE	78.61
RMSE	124.43

Figure 21 shows the most important features used by the model to make the predictions, the order of the first ones is the same as the random forest, but the importance is lower.

	feature	importance
0	lag_1	0.164177
167	lag_168	0.159586
23	lag_24	0.064628
166	lag_167	0.018919
1	lag_2	0.017908
95	lag_96	0.015311
143	lag_144	0.014229
55	lag_56	0.013418
102	lag_103	0.012202
22	lag_23	0.011435

Figure 21: Feature importance for XGBoost model

- XGBoost with exogenous variables (month, weekday, hour)

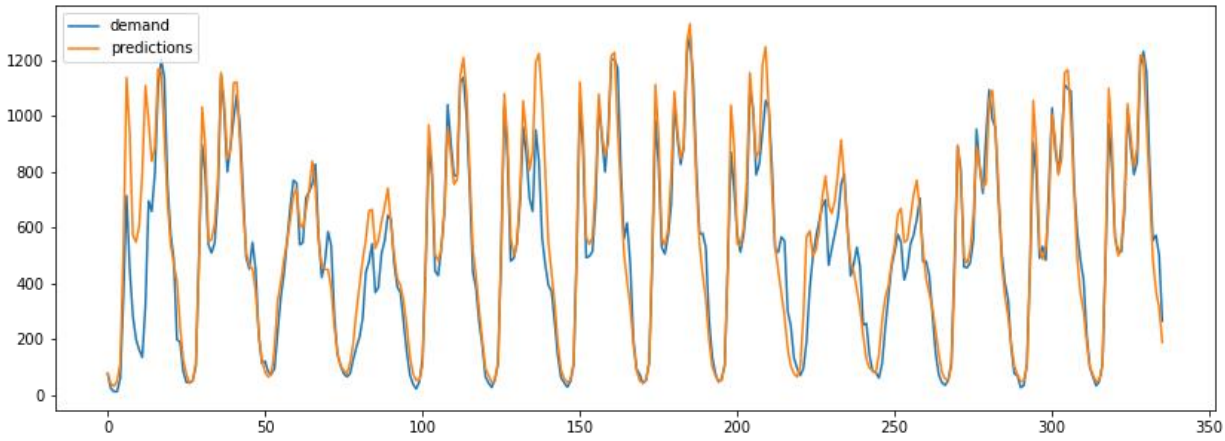

Figure 22: Demand and predictions for XGBoost model

Table 6: metrics for XGBoost model

Metric	Value
MSE	14538.14
MAE	75.4
RMSE	120.57

The result of the model adding exogenous variables is 6% better in terms of MSE than the same model without exogenous variables. The most important features of the model are the lag values (same as the rest of the models), but we can observe that the variable hour_18 is in the top 10 feature importance, although the importance is low (1.6% compared to the 15.1% of the first feature).

	feature	importance
0	lag_1	0.151749
167	lag_168	0.147553
23	lag_24	0.057247
1	lag_2	0.016630
166	lag_167	0.016345
193	hour_18	0.016090
95	lag_96	0.014800
143	lag_144	0.013351
97	lag_98	0.011967
22	lag_23	0.011152

Figure 23: Feature importance XGBoost with exogenous variables

4.1.5. Light GBM

- Light GBM without exogenous variables (only lag variables)

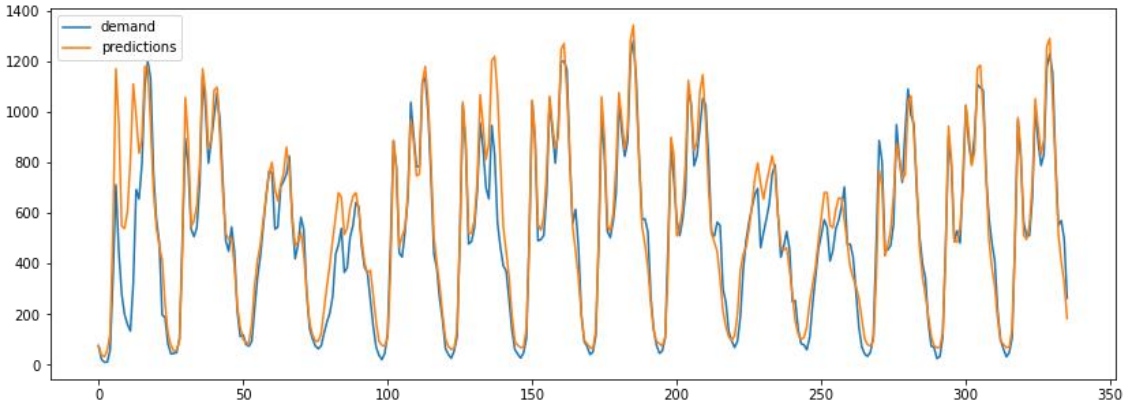

Figure 24: Demand and predictions for LightGBM model

Table 7: Metrics for Light GBM model

Metric	*Value*
MSE	13366.18
MAE	72.42
RMSE	115.61

The metrics of the Light GBM model are slightly better than the XGBoost model. The most important features are the same as the rest of the models.

	feature	importance
0	lag_1	2611
23	lag_24	1082
167	lag_168	718
24	lag_25	578
22	lag_23	525
5	lag_6	482
1	lag_2	477
25	lag_26	390
166	lag_167	377
9	lag_10	374

Figure 25: Feature importance for Light GBM model

- Light GBM with exogenous variables (weekday, month and hour)

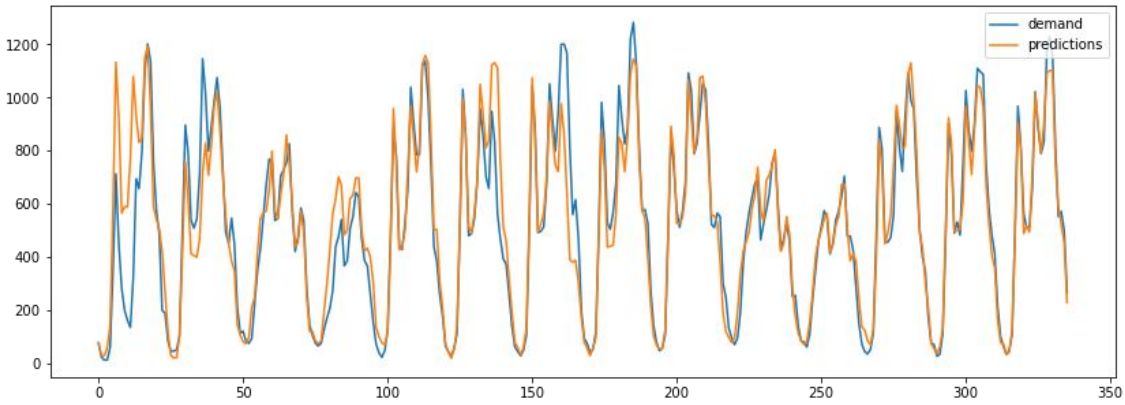

Figure 26: demand and predictions Light GBM with exogenous variables

Table 8: metrics for Light GBM with exogenous variables

Metric	*Value*
MSE	12110.64
MAE	61.34
RMSE	110.04

The results are better than the same model without exogenous variables. The MSE is 9% better than the same model without exogenous variables. This model gives importance to the day of the week to forecast the demand, as can be shown in figure 27.

	feature	importance
0	lag_24	3978
1	lag_168	3512
3	weekday_Monday	871
4	weekday_Saturday	518
5	weekday_Sunday	504
2	weekday_Friday	477
37	month_5	357
7	weekday_Tuesday	258
44	month_12	238
16	hour_7	213

Figure 27: feature importance for Light GBM model with exogenous variables

4.1.6. CatBoost

- CatBoost model with exogenous variables

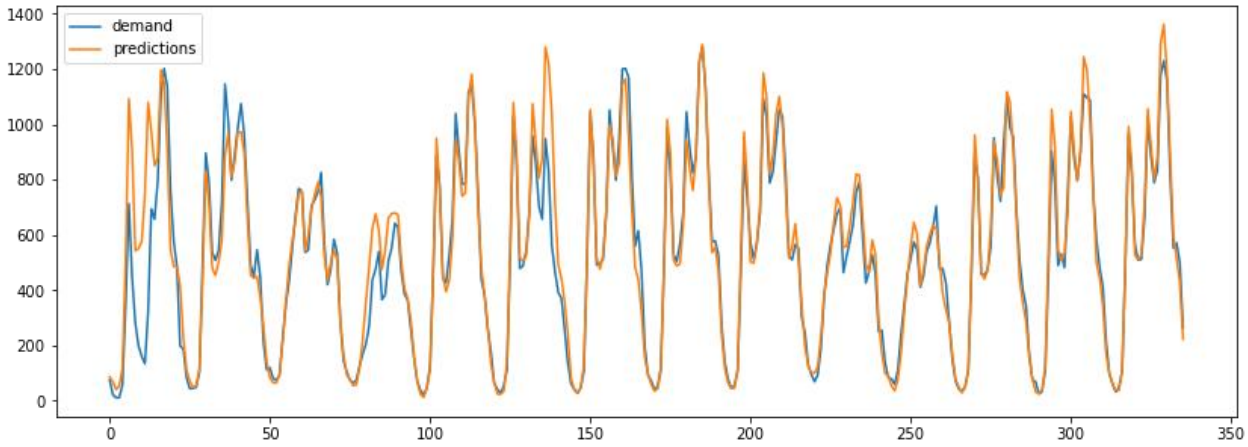

Figure 28: demand and predictions of Catboost model

Table 9: metrics for Catboost model

Metric	*Value*
MSE	11075.27
MAE	59.81
RMSE	105.23

The results of the CatBoost model are better than the other models used, the most important features used as predictors are the lag variables (lag_1, lag_24 and lag_168).

	feature	importance
0	lag_1	34.058630
23	lag_24	9.917872
167	lag_168	7.503889
166	lag_167	3.730534
22	lag_23	2.946521
143	lag_144	2.741277
5	lag_6	2.111039
25	lag_26	1.708847
6	lag_7	1.351299
24	lag_25	1.234548

Figure 29: Feature importance for Catboost model

4.1.7. Comparison of models

Figure 30 shows the results of all the models used to predict global demand, ordered from lowest to highest MSE.

The models with which the best results have been obtained are the CatBoost and the Light GBM, including exogenous variables, these are gradient boosting models. Gradient boosting models have proved to be very competitive to forecast demand.

The model that obtained the worst results is the ARIMA model.

	model	MSE	MAE	RMSE
12	Catboost exog variables	11075.272690	59.816788	105.239121
11	Light GBM exog variables	12110.648835	61.348503	110.048393
3	Random Forest grid search	13062.952660	69.379588	114.293275
5	Random Forest with exog. variables grid search	13124.972068	69.811174	114.564270
9	Light GBM grid search	13366.185942	72.424363	115.612222
4	Random forest with exog. variables	13951.752894	70.143006	118.117538
7	XGBoost with exog	14538.145517	75.407824	120.574232
10	Light GBM exog variables	15368.099382	75.656189	123.968139
6	XGBoost	15484.633669	78.618286	124.437268
2	Random Forest	16884.489419	86.475893	129.940330
8	Light GBM	23815.884251	111.858306	154.323959
0	naive	24746.330357	92.889881	157.309664
1	ARIMA (24,0,6)	71041.269014	213.579555	266.535681

Figure 30: comparison of all models trained

After analyzing the results obtained in the different models, a selection has been made of those models that offer the best results, which will be those that will be shown in the frontend. In order to obtain the predictions, the model has been retrained using all the available data.

As can be seen in the figure, the predictions of the four models compared are quite similar.

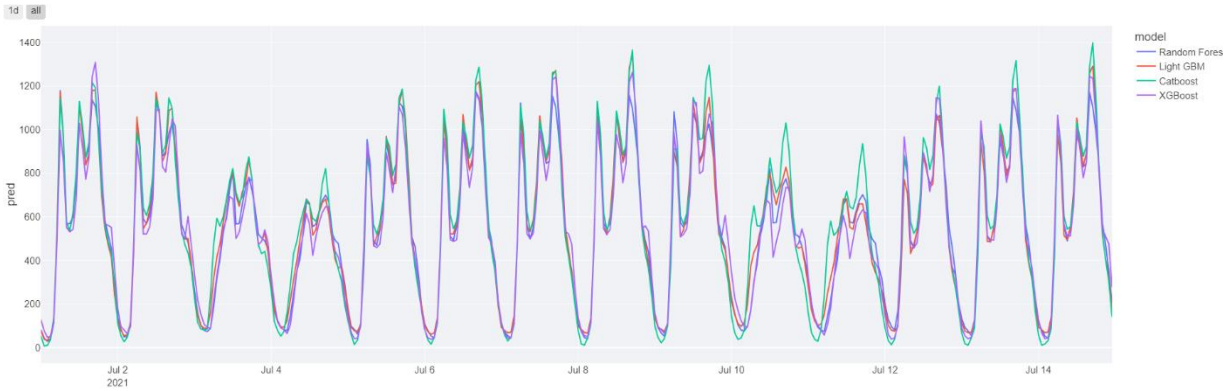

Figure 31: Comparison of predictions of models with best results

Models including exogenous variables (hour, weekday and month) proved to have better results than the same models that only considered lag values.

4.2. Models for zone

4.2.1. Modelling the zones

To model the demand per zones, we will first analyze how we divide into zones the whole area covered by Bicimad services. In the figure 5 we observed that the postal 28001 has a lot of stations, so the first step is to perform a cluster analysis in this zone, so the number of stations per zone is equally distributed.

We use **KMeans** , this unsupervised algorithm follows a simple procedure of classifying a given data set into a number of clusters, defined by the letter "k," which is fixed beforehand. The clusters are then positioned as points and all observations or data points are associated with the nearest cluster, computed, adjusted and then the process starts overusing the new adjustments until a desired result is reached. We are interested in the group the stations by proximity in terms on coordinates (latitude and longitude).

As we can see in the figure 32 we divided the stations in the zone 28001 in 10 regions by proximity, each one represented by one color in the figure.

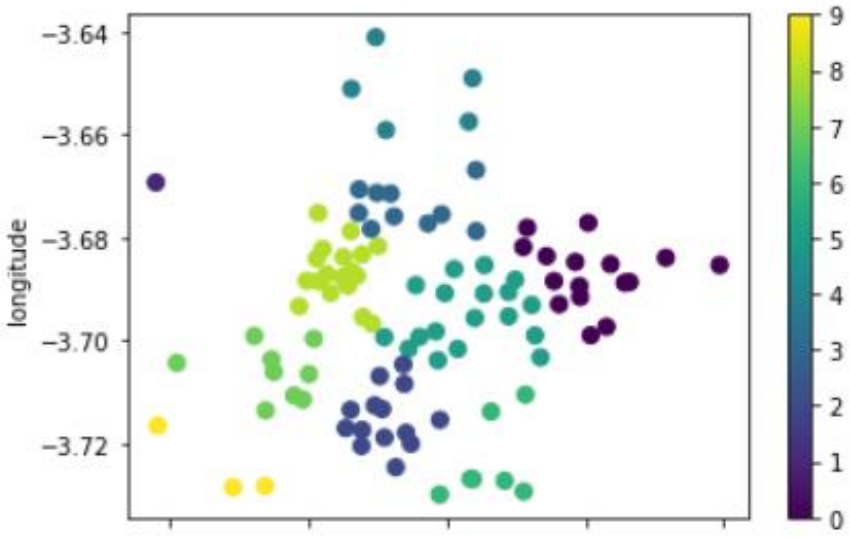

Figure 32: Clusterization of the postal code 28001

To identify these zones in which the stations located in the postal code 28001 have been divided, we named them: 28001_0, 28001_1,..., 28001_9. In the figure 33 are represented the number of stations per zone (considering the zones we are going to use in the project), as we can see, compared with the figure 5, the distribution of stations per postal code is more equal.

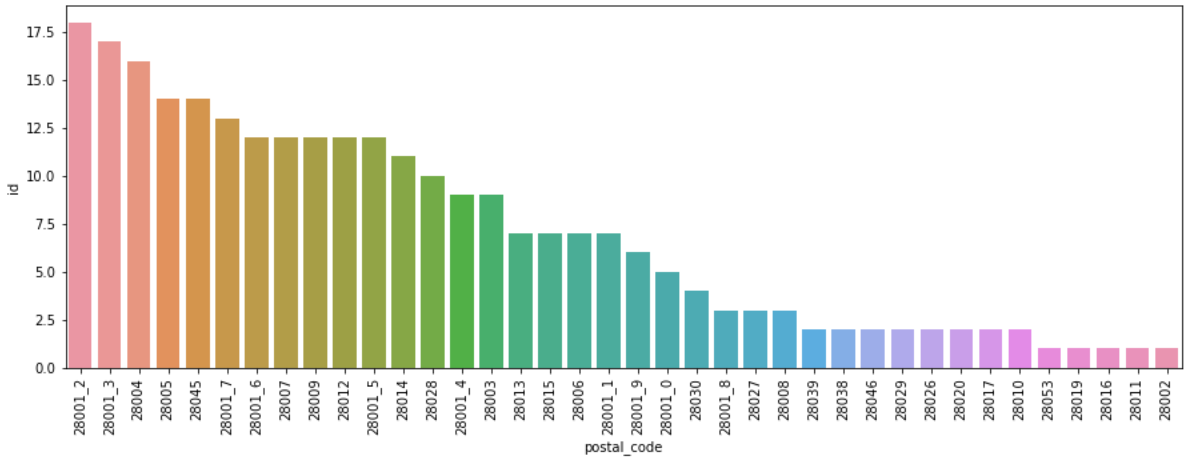

Figure 33:Stations per postal code after cluster

4.2.2. Machine learning models per zone

We want to make a model that predicts the demand of each zone (we divided the area of the service in 38 zones), and we perform the process shown in figure 34 for each zone. In the notebook “4. Models demand per zone” the results of all the models can be seen.

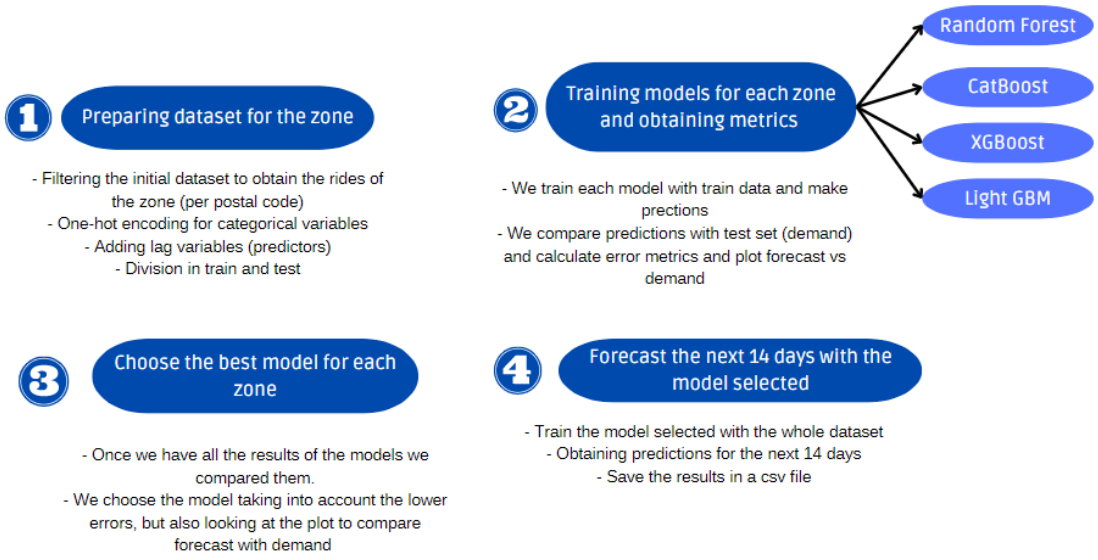

Figure 34: Process to obtain predictions of demand per zone

Predicting demand by zone is useful to the business because it allows it to identify which zones have the highest demand and which have the lowest demand. The service area has been divided into 38 zones, and quite diverse results have been obtained.

In the zones where there is a higher demand, such as, for example, the zone 28001_3 results are better than in those areas, such as 28001_4, where the hourly demand is lower and therefore the results and models are worse adapted and worse results are obtained (not in absolute terms, but looking at the graphs you can see the difference).

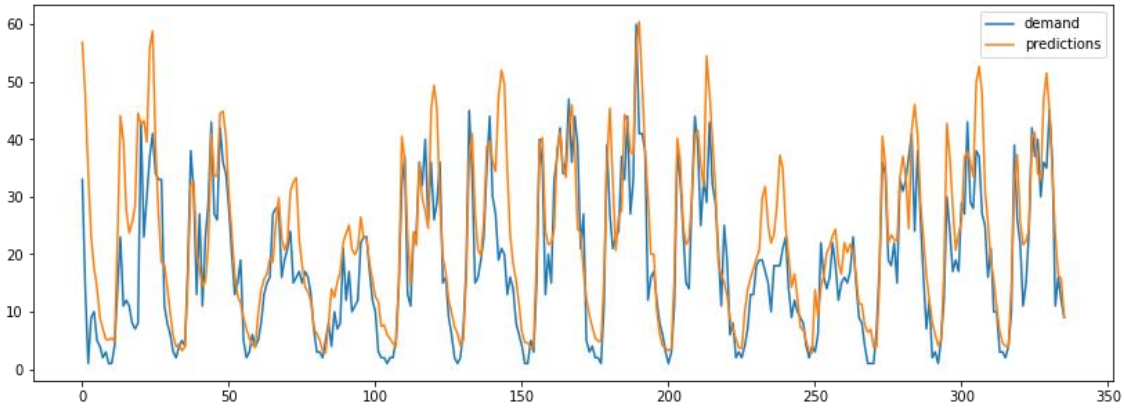

Figure 35: demand - forecast for zone 28001_3

Table 10: metrics for model 28001_3

Metric	*Value*
MSE	74.82
MAE	6.21
RMSE	8.65

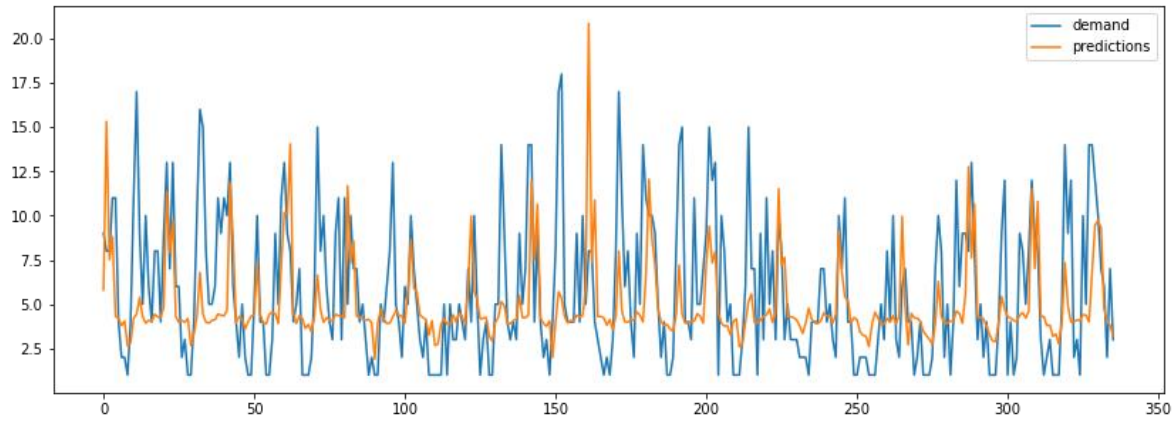

Figure 36: demand and forecast for 28001_4

Table 11: metrics for model 28001_4

Metric	*Value*
MSE	13.13
MAE	2.67
RMSE	3.62

For each zone, the best machine learning model has been selected (looking at the graphs and the error results), in the figure 37 a summary of the models selected for each zone is shown.

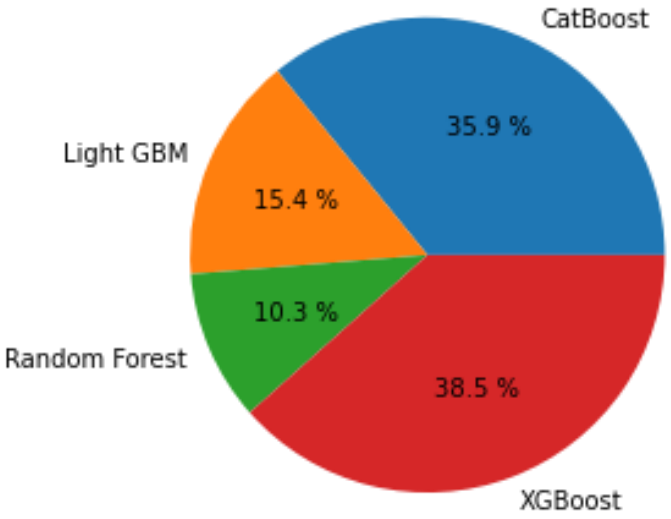

Figure 37: distribution of models selected in demand per zones

The best results have been obtained with XGBoost followed by CatBoost, indicating that gradient boosting models give good results for time series prediction.

5. Conclusions

In this section we will present the main conclusions obtained after the elaboration of this project.

First, after analyzing the time series and its correlation factors with the different variables analyzed, as well as after analyzing the feature importance after performing the machine learning models, it is concluded that the demand for bicycles is highly related to itself in the past (lag variables), especially with lag 1 (demand one hour before), lag 24 (demand 24 hours before) and lag 168 (demand one week before).

The models used for global demand forecasting have much better metrics than those obtained for forecasting demand per zones. However, if one looks at the graphs comparing forecast and demand, the curves fit better in the global model than in the partial models. This is because the absolute demand numbers are much higher in the global demand than in the partial demand (up to 1500 in the case of the global demand).

The results are better for the global demand than for the demand by zones because we have more information for the total demand (higher number of demand per hour), while for some zones the demand is very low in general and in many occasions it is 0 for some hours, so the models are less adjusted to the real demand.

To train an ARIMA model you need to have much more technical knowledge of time series to get good results.

Machine learning models, especially gradient boosting models (XGBoost, LightGBM and CatBoost) have proven to obtain the really great for predicting demand. These models (gradient boosting and also Random Forest) are easier to train than traditional models used for time series forecasting (such as ARIMA models), the time it takes to optimize the model in the grid search and in the fit and predict is much less than in the other models, being able to obtain the predictions in a much faster way.

Moreover, these models are characterized by the easiness of including exogenous variables in addition to the autoregressive ones, they allow incorporating non-linear behaviors in the model and prove high scalability with large volumes of data.

6. User manual for the front end

The frontend has been developed in Strimlit (https://share.streamlit.io/paulamartinm/bicimad_prevision_tfm/main/Frontend/streamlit_app.py)

At the top of the page is the menu (number 1 in figure 32), where you can switch between the two top views:

- Prediction of demand: here you can see the demand predictions for the next 14 days or 24 hours (it starts predicting from the next hour from which you have historical data, in the case of the application, from July 2021). In addition, more information is provided and will be explained below.
- Dashboard demand: here you can view data of interest on historical information (demand graphs according to time of day or day of the week, differentiating between weekdays and weekends).

Both pages have different filtering options, all of them are located in the bar on the left side of the screen. (Number 2 in figure 32)

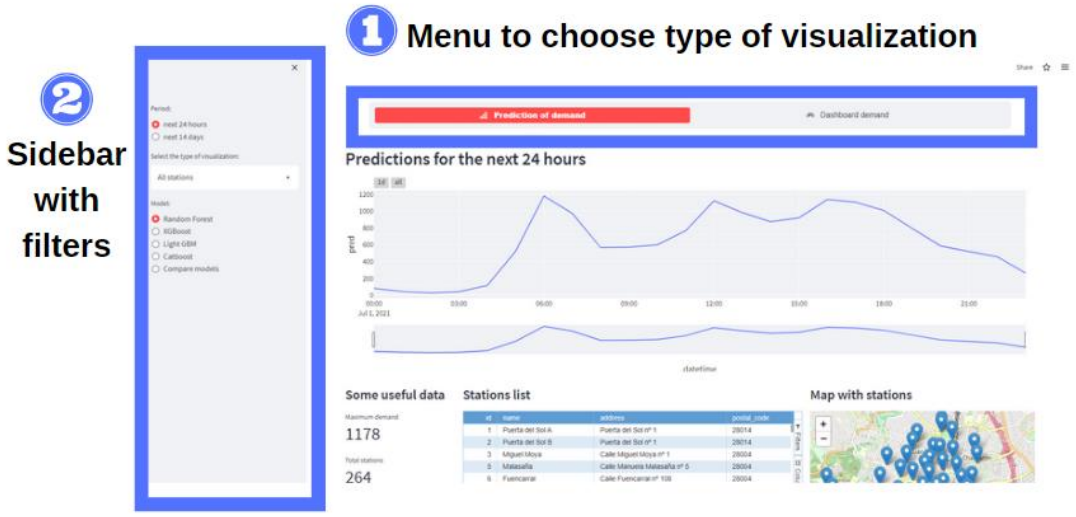

Figure 38: Frontend – menú and filtering

6.1. Frontend: prediction of demand

At the top is a graph showing the demand forecasts for the next 24h or for the next 14 days (as selected in the period selector).

In the lower part, some important KPIs are shown:

- Maximum demand

- Total number of stations

- Total number of docks

All information is filtered according to the selectors on the left (period, zone, model used).

In addition, information about the stations is displayed at the bottom. A table with the name, location and number of docks of the stations, and also a map with these.

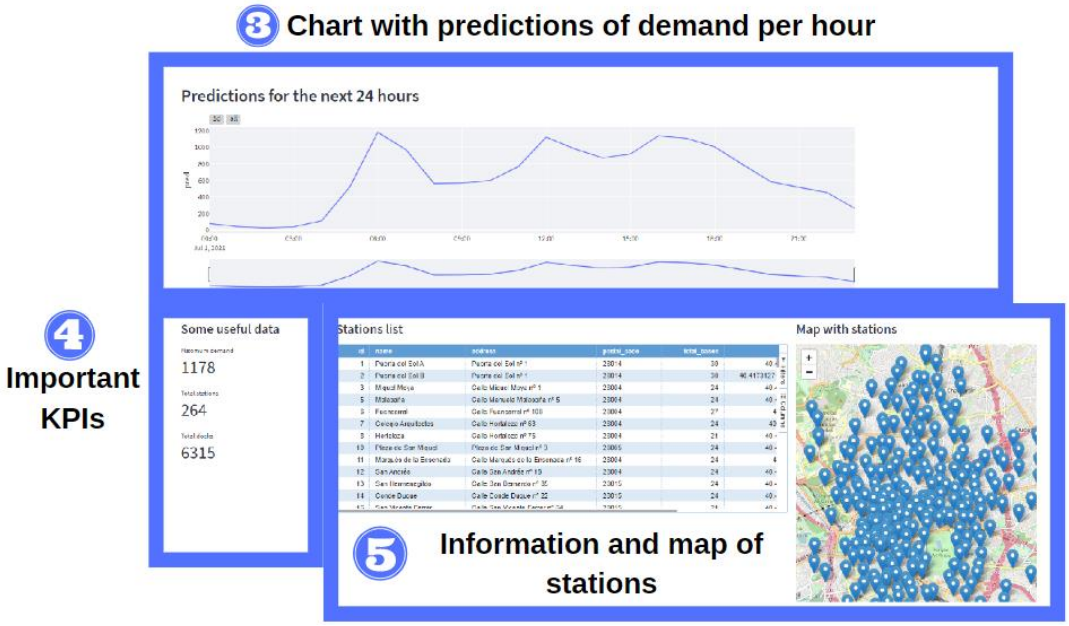

Figure 39: Frontend - prediction of demand I

Options of filters:

- Period:
  - 24 hours: filters the predictions and KPIs for the next 24 hours
  - 14 days: filters the predictions and KPIs for the next 14 days

- Type of visualization:
  - All stations: shows predictions of aggregated demand of all the Bicimad service.
  - Demand per zones: shows the predictions for each of the zones for which demand has been modeled

Then depending on the type of visualization (demand per zones / all stations) there are different filters available:

Demand per zones:

- Postal code: zones available to see predictions (to see the list of stations included in each postal code you can consult the stations list)

- Comparison of all zones or individual zones:
  - Individual zone: Here you see the information zone by zone (using the postal code filter).
  - Comparison all zones: here the visualization changes, the upper graph shows the predictions for all zones (one line per zone) and the map of the stations is replaced with a graph showing the models used in the zone predictions. As can be seen in figure 34.

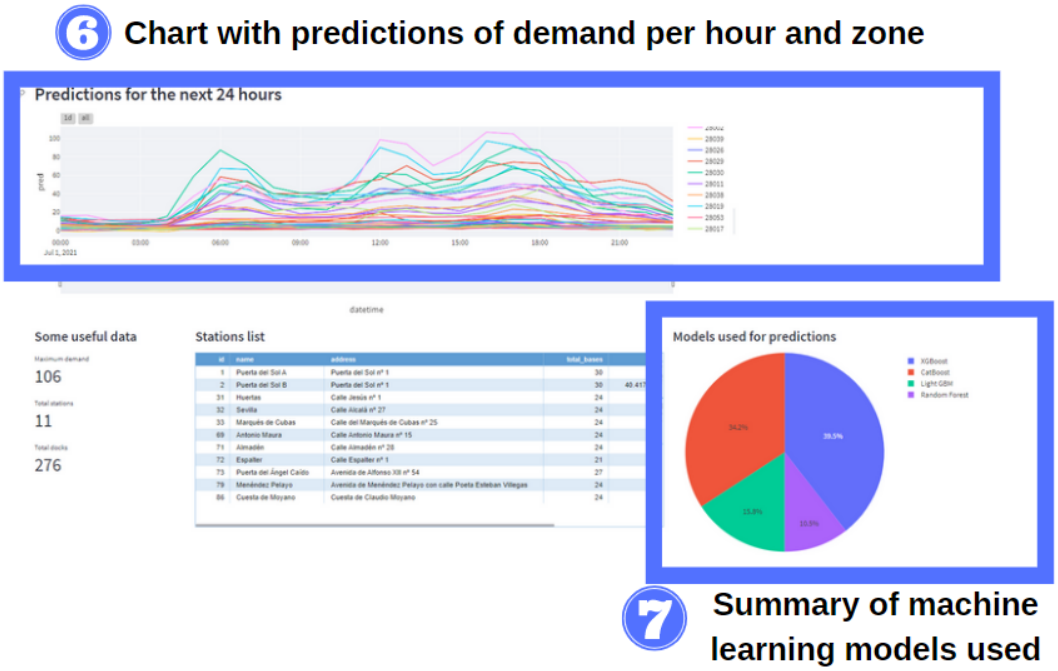

Figure 40: Frontend - prediction demand II

All stations:

- Model: here you can filter the predictions for the different models developed in the project. There is the option of 'compare models', when selected, the visualization changes (see figure 35).
  The graph shows the comparison between the predictions obtained in each model and below is a table where the metrics of the different models used can be compared.

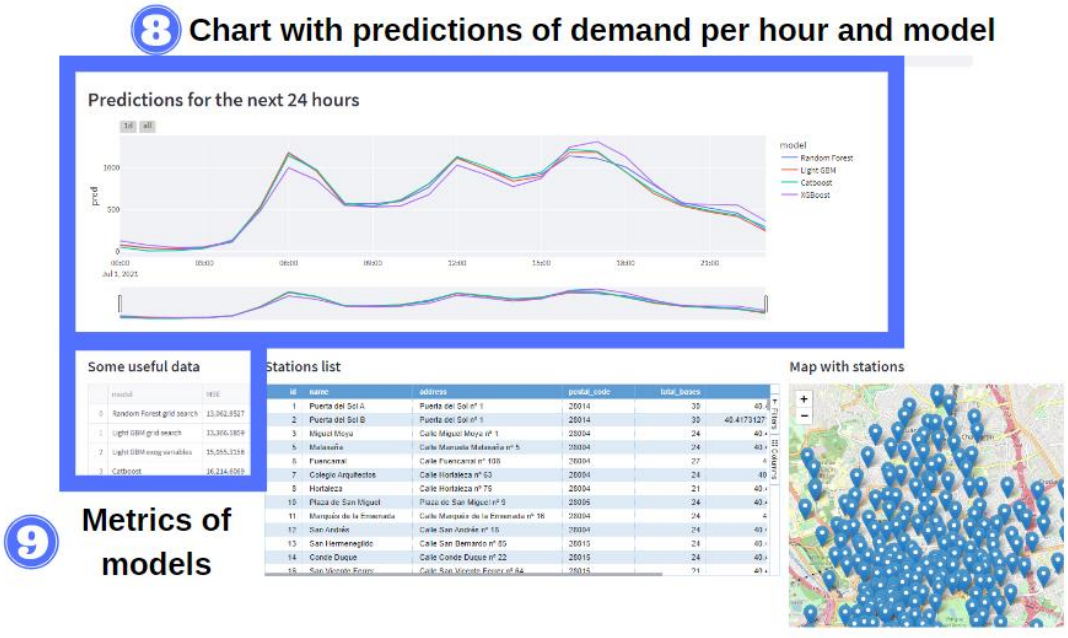

Figure 41: Frontend - prediction demand III

6.2. Dashboard demand

At the top of the screen, 4 KPIs related to Bicimad's historical demand data are displayed.

- Average demand per hour: with the variation compared to the previous year.

- Peak demand hour

- Highest demand day of the week

- Number of docks available

Below is a graph with the distribution of demand, distinguishing between weekdays (Monday to Friday) and weekends (Saturday and Sunday). This graph has several display options: by hour of day or by day of the week and as a boxplot or lineplot (with the average demand per period).

The lower part shows the evolution of the time series in the selected year.

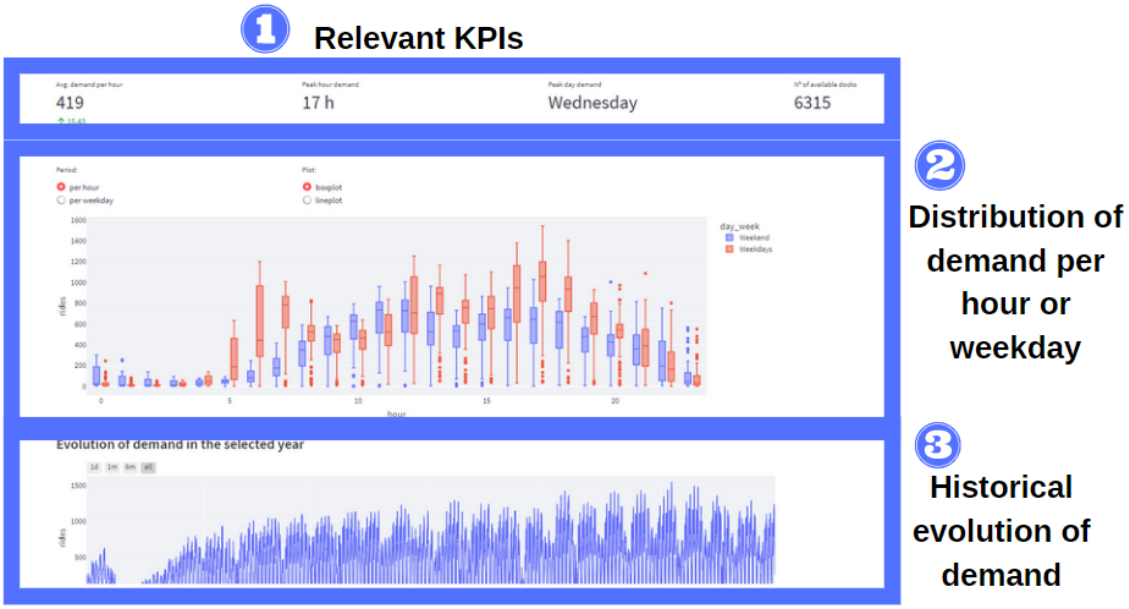

Figure 42: Frontend - dashboard demand I

All the information can be filtered by the filters located in the sidebar:

Options of filters:

- Year
- Month
- Type of visualization:
  - All stations: aggregated demand
  - Demand per zones: if selected, a postal code selector is displayed to select the zone (postal code).

- Bottom information:
    - Evolution demand: if selected, the historical evolution of demand is displayed at the bottom of the page.
    - Stations information: if selected the bottom of the page shows information about the stations of Bicimad (list and map).