People Traffic Forecasting

Ramiro Arribas Garayoa
in collaboration with:

June 7, 2019

Contents

1 Introduction

This project is the product of my collaboration with iMotion analytics[1], a company that offers services of customer analysis, assisting their clients in making informed decissions and optimize their business strategies. They are willing to explore new lines of services and proposed me to try to find a predictor of people traffic using the data they already gather.

So that became the **purpose of this project:** to establish the foundations of a generic predictor of people traffic for a given time window, compatible with their current systems, using supervised learning algorithms.

As per the company specifications, there is a special interest in reaching high accuracies for the next immediate days. During the development, the aim has been to be able to answer the following questions:

- Is it preferable to keep the original (high) frequency or would it be better to downsample the data to a daily frequency to get better results?
- Do the external data and possible engineered features add relevant information to the models or just introduce noise resulting in a poorer performance?
- Which of the models to be tested yields better results?
- Do the assumptions made for one time series holds for the others?

The following are a list of services and software tools I have used to develop the project:

Python has been the programming language used for exploring and cleaning data and for the creation and analysis of prediction models.

The linux command line and Github for version control.

Texmaker to write the memorandum.

Because of privacy reasons the original data is not available on the repository, just the external weather data used, provided by Meteocat: http://www.meteocat.com. All the python notebooks used are stored on the different folders of the repository, available for review.

[1]For more information, check their website: http://www.imotionanalytics.com/en/

2 Data

The raw data provided by the company consists of three time series of different length, belonging to three public health buildings located in different places of the Barcelona province: 'Guinardó' and 'Gòtic' city quarters and 'Vilanova i la Geltrú', a town at 50 km distance south-west of Barcelona. Concretely, the name of the buildings (which we will use when referring to its pertinent time series) are 'CAP Maragall', 'CAP Raval Nord' and 'CAP Jaume I'.

Figure 1: Map of the three CAP locations.

Each time series contains 48 observations per day of the aggregated total amount of **entries** per time stamp (at a frequency of 30 minutes). Other features are the **time stamp** in Unix time stamp format (UTC time zone), **SitedId**, **total entries per genre**, **total outs** (exit traffic) and **total accesses** (as a sum of entries plus exits).

The weather data from Meteocat was provided on demand, as they limit the data available on their website. Its granularity is the same than the company data, and the original fields of their files are:

EMA: code of the meteorological station
DATA: time stamp in local time
T: average temperature
TX: max temperature
TN: min tempreature
PPT: precipitation

3 Exploratory analysis and data cleaning

Before starting the exploratory analysis, there is a notebook created to adjust the meteorological data to the UTC time zone. That was a previous step needed to merge the data from both sources coherently.

The data of each site has been analyzed separately. The purpose of the exploratory process has been to check the consistency of each time series (gaps have been found on the Maragall and Jaume I time series), studying the presence of outliers and extracting information to be used in the forecasting models. At the end of this phase, different csv files of each location have been prepared for two different scenarios: original frequency forecasting and daily-grouped frequency forecasting.

Bearing in mind that the purpose of the company is to come up with a generic predictor, there's not much room for domain specific features. Therefore, just two binary features have been created from patterns detected on the exploratory analysis: **Weekday / Weekend** and **Open / Closed** (to define the schedule of each building; not applicable on the daily grouped data context). From the Maragall dataset, another feature has been created, given the existence of relevant **outliers**: specifically, the outliers detected belong to two relevant dates of the recent political situation in Catalonia. One is November 11th of 2017, when a demonstration took place claiming for the liberation of the so-called 'imprisoned politicians'; the other is September 11th of 2018, date of celebration of the national festivity in Catalonia. The feature has been created for possible analytical purposes since it would not be possible to use it for the forecasting process (a future date should be flagged as outlier beforehand, which for certain domains may not make sense).

When analyzing the data, the autocorrelation plot of each time series has shown the same peculiarity: there are weekly cycles of strongly correlated data (see **figure 2**). Therefore, it has been decided to use lags as features of the forecasting models, including all the records within one week (7 day * 48 observations = 336 lags on the original frequency data, and 7 lags on the down-sampled data).

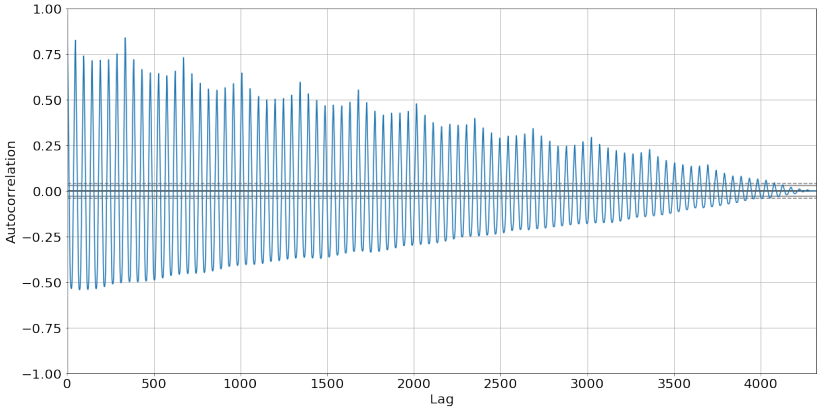

Figure 2: Raval Nord autocorrelation plot.

When grouping the data by day, the traffic data has been summed, whilst for the meteorological data the average has been calculated. Finally, different csv files have been prepared for each location and frequency.

4 Modeling

Since the Maragall time series is the longest of the three (spanning more than two years), it has been used as a reference to test all the models. Because of calendar reasons, it has not been possible to study the time series of the other locations. The supervised learning algorithms used have been linear regression, auto ARIMA and random forest.

When forecasting time series data, if the purpose is to forecast unknown data, no future information can be included in the model. This obvious statement has important implications: using lags of future observations as input (independent) features will introduce a systematic bias on the predictions that will produce a better performance. That would be erroneous because we cannot know for certain which values those future lags would take. An approach to build an honest predictor would be to predict new data and along the way, use it to create features for the new predictions. The only future inputs allowed are the features extracted from external data (weather, schedules...), since that information would be previously collected and transformed to be fitted into the model.

Some custom functions have been coded in order to enhance the process of results extraction and analysis:

· **train_dev_test_split:** a function that implements the splitting of time series data (without randomizing) in train, dev and test.

· **get_future_predictions:** a generic function that allows the user to get results for a future period of unknown data. Among other parameters, it accepts a data frame of 'external data' as a source from which to extract data of different nature (e.g. weather data, or engineered features). It makes predictions, includes them as new observations and create new lags as features to forecast the next prediction.

· **ARIMA_future_preds:** similar to the generic 'get_future_predictions' function, but adapted to the ARIMA context (where exogenous data can be excluded). In this function the model isn't refit in each iteration.

· **get_scoring_measures:** a function to build a data frame with the RMSE and $\bar{R}^2$ scorings of results extracted from different scenarios.

The plan for each machine learning algorithm used have been to produce and analyze results for different scenarios in order to test different hypothesis:

- Predict known future or predict unknown future: for comparative reasons, results have been extracted both from lags of known observations (introducing future information in the forecast) and from lags created along the forecasting process.

- Use all the available data for the predictions or use just lags: since high frequency data is at hand, maybe the lags might have all the information the regressor needs to make accurate predictions and adding external information would be a source of noise.

- High frequency vs. daily frequency: since the objective is to forecast people traffic for whole days, maybe downsampling the data grouping by day prior to fitting it into the predictive models is better than keeping the original frequency.

In general terms, for each algorithm tested there is a batch of results for eight different scenarios. These are:

Table 1: Analyzed scenarios.

	High frequency	**Daily frequency**
Known future	All columns Just lags	All columns Just lags
Unknown future	All columns Just lags	All columns Just lags

Besides the above, the computational cost for each scenario has been measured as a way to check the time each model needs to process the data vs. the scoring it reaches. Two notebooks have been prepared for each algorithm used, one for results extraction and other for results analysis. For all models the same data splitting approach have been used (train, dev and test).

The performance scoring measures used have been Root Mean Square Error and R^2 adjusted (or $\bar{R}^2$). The RMSE has two main characteristics worth mentioning: it punishes large errors and it's in the same units as the predictions, which is a plus for interpretation.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{T}(\hat{y}_t - y_t)^2}{T}} \tag{1}$$

Where $\hat{y}_t$ are the predicted values for times t of a regression's dependent variable y_t for T different predictions.

$\bar{R}^2$ is a measure that helps to get an idea of the amount of variance explained by the model, and its main difference compared to R^2 is that it adjusts for the number of terms in the model.

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{2}$$

Where $R^2 = 1 - FVU$ (or the *Fraction of Variance Unexplained*), p is the total number of explanatory variables in the model (not including the constant term), and n is the sample size. Note that $\bar{R}^2$ may be negative when the analyzed model fits the data poorly.

4.1 Algorithms

In this section, the most relevant ideas extracted from the results of each algorithm will be discussed. For an in-depth analysis of the models, refer to the notebooks.

Linear Regression

The linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The algorithm of the scikit-learn python library has been the one chosen for this project. The results extraction notebook is straightforward, aiming to get results for all scenarios of the whole test portion of data:

Table 2: Linear regression performance scoring results (original frequency data).

	$RMSE$	$\bar{R}^2$
PKF All columns	25.9837	0.9504
PKF Lags	26.1267	0.9499
PUF All columns	71.1761	0.6280
PUF Lags	84.7839	0.4725

As seen on the table above, when predicting known future (PKF), including or not external data doesn't make a big difference. However, when predicting unknown future (PUF), including external data dramatically improves the forecasts (both in terms of RMSE and $\bar{R}^2$). The model chosen for further analysis has been the one that **predicts unknown future using all columns**. When predicting a few days into the future, scorings have been calculated for time windows of different span, and the best results are shown when taking into account the first 7 days.

Comparisons of the two frequencies have been made, grouping the high frequency forecasts by day. When analyzing the results of using high frequency data vs. daily grouped data, the RMSE for the whole test portion is much better when using high frequency:

Table 3: Comparison of Linear Regression performance scoring for both frequencies.

	$RMSE$
48 obsv./day	1555.4155
1 obs./day	2551.8389

When estimating the RMSE for the first 14 days it can be observed that even though in general terms the high frequency curve has smaller values, the daily curve has a better scoring on some days:

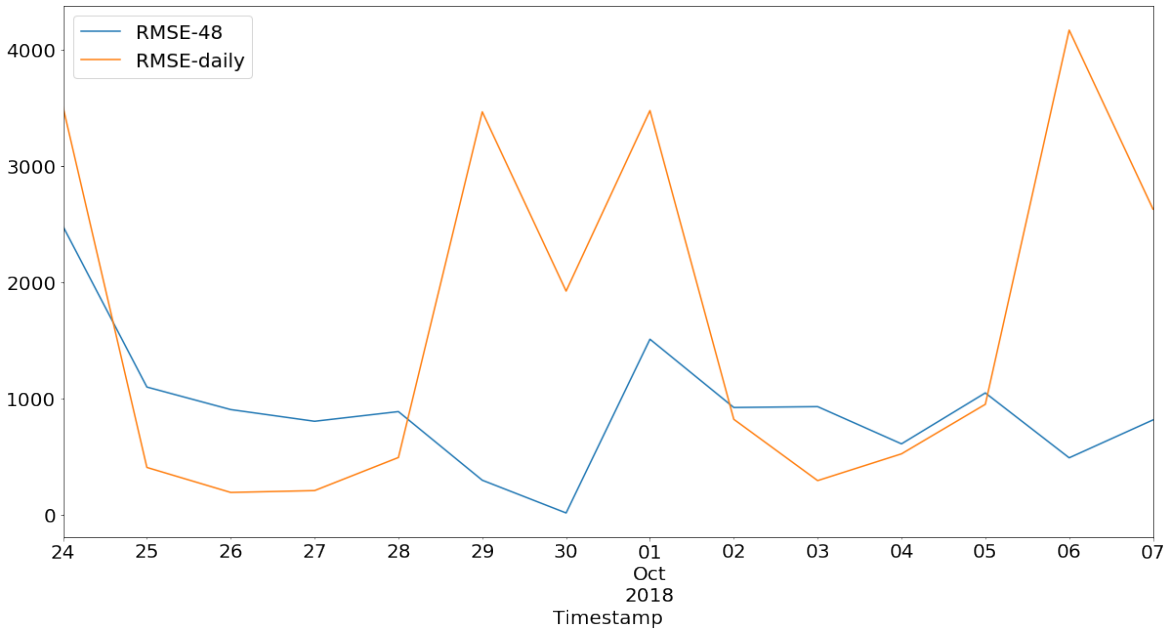

Figure 3: Linear regression RMSE curves for both frequencies of the first 14 days of predictions.

The calculation cost is much higher when predicting high frequency data (more than 2 hours for the whole test portion) than daily data (which takes seconds).

ARIMA

The term ARIMA stands for *Autoregressive Integrated Moving Average*, and is a class of model that captures a suite of different standard temporal structures in time series data. The ARIMA algorithm used for this project has been the one offered by the pyramid ARIMA python package[1], that allows for an automatic adjustment of parameters. The ARIMA models can be fitted in two ways: just using the target variable or fitting the model using the target feature and exogenous data. When using exogenous data, constant values must be avoided.

It has not been possible to extract results for the scenario that predicts unknown future using all columns. The reason is that a runtime error arose during its computation. On the other hand, besides the rest of scenarios an extra iteration of results has been extracted without using exogenous data.

[1]For more information, check the documentation at: https://www.alkaline-ml.com/pmdarima/

The results obtained using ARIMA show the expected behaviour, with a much smaller RMSE when using observation's lags, and better scorings using external data for unknown future predictions:

Table 4: ARIMA performance scoring results (original frequency data).

	$RMSE$	$\bar{R}^2$
PKF All columns	28.5617	0.9401
PKF Lags	28.6010	0.9400
PUF Lags	103.2243	0.2180
No exogenous data	169.7499	

The model that doesn't include exogenous data shows an almost constant oscillation between two values, so judging by that and by the scoring results it doesn't seem a good candidate:

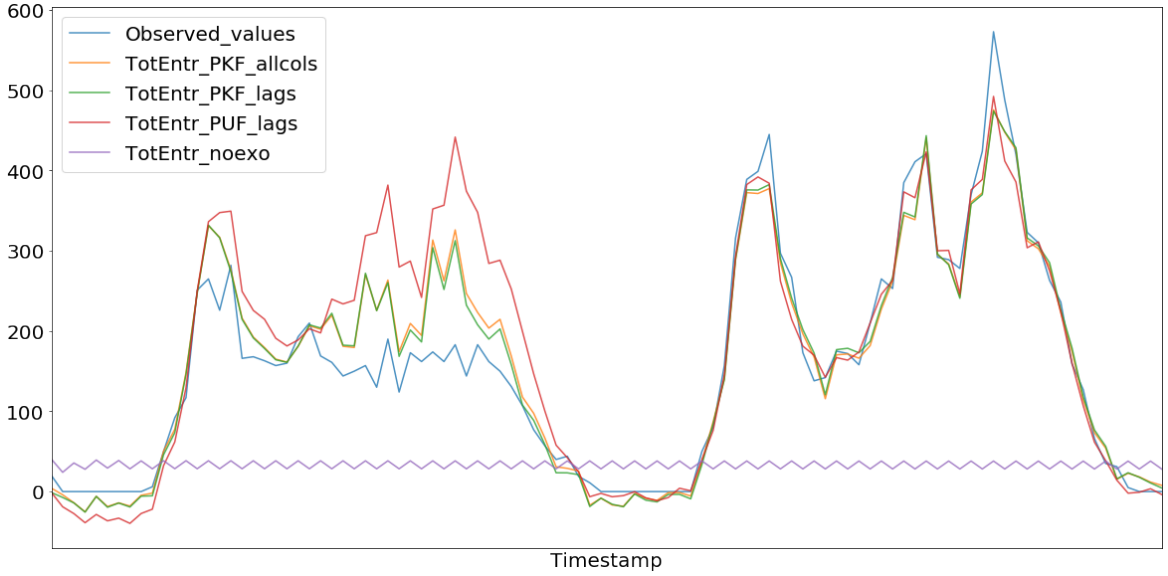

Figure 4: Prediction curves for the first 100 observations of each high frequency scenario.

The model chosen for further analysis has been the one that **predicts unknown future using lags**. The residuals show a fairly strong bias towards negative values, and there's a positive correlation of residuals against fitted values. For the first few predicted days, the time window that yields the best results is the one that includes 12 days. When comparing both frequencies for the overall test data just using lags, the ARIMA algorithm shows a better RMSE performance using the daily frequency than the high frequency.

Table 5: Comparison of ARIMA performance scoring for both frequencies.

	RMSE
48 obsv./day	3386.0450
1 obs./day	2984.8807

If we shift the focus to the first days, results differ: for spans of different size, high frequency is better, but when considering specific future days, the daily frequency is more accurate.

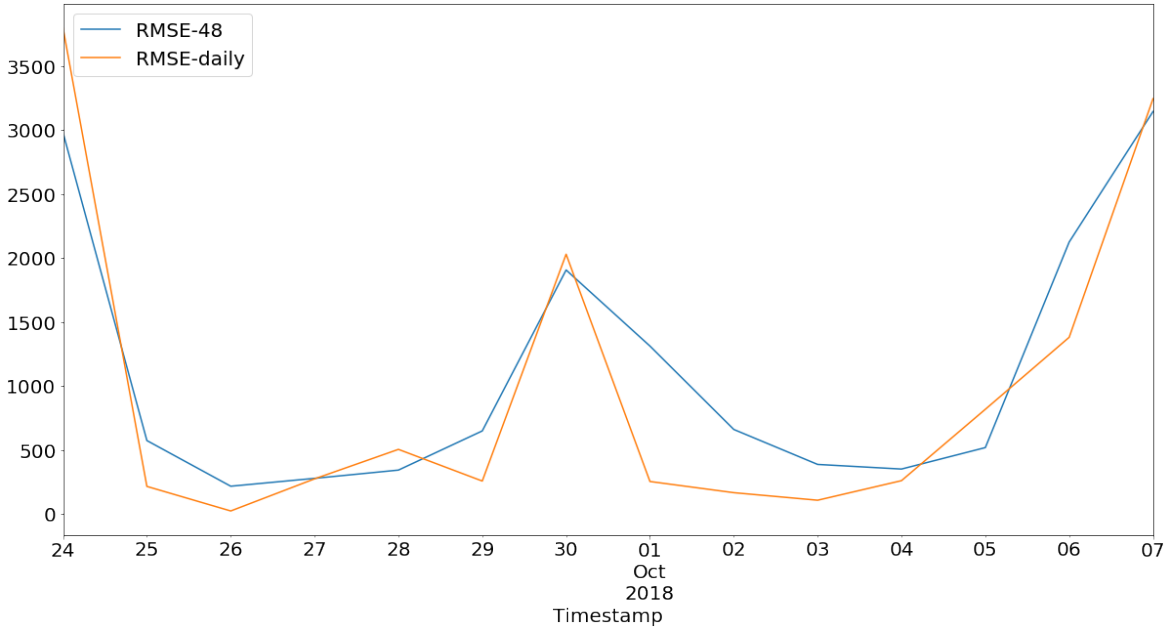

Figure 5: ARIMA RMSE curves for both frequencies of the first 14 days of predictions.

When considering the computational cost, the most costly algorithm is the high frequency model that just uses lags whilst all the others take almost no time to run:

Table 6: ARIMA runtime (in hours) for each scenario.

	48 obs./day	**1 obs./day**
PKF All columns	0.2250	0.000004
PKF Lags	0.2210	0.000005
PUF All columns		0.0012
PUF Lags	3.6753	0.0009
No exogenous data	0.0002	0.000007

If the aim would be to find good results for the first days, the daily resolution model using just lags would be the chosen one (as it is much faster to train).

Random Forest

Random forests are an ensemble learning method for classification and regression tasks that operates by constructing a multitude of decision trees at a training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The algorithm of the scikit-learn python library has been the one chosen for this project. Prior to the results extraction, there has been a process of model tuning to find the optimal parameters to use in the present context. There's a variety of hyperparameters to tweak, and even though at the beginning the aim was to try different values for six of them, the scope was quickly reduced (the prime reason being runtime, as some combinations of parameters took up to 95 minutes to run). Finally, the number of estimatiors and the maximum depth have been the two hyperparameters tuned.

The results extraction process has led to a modification of the original *get future predictions* function to set the refitting process as optional, since each iteration of getting predictions plus refitting the model took more than 6 minutes. Bearing in mind the size of the test portion of data (7932 rows), that meant a computational cost of more than a month. Therefore, just results of the first 10 days (480 observations) have been extracted in that fashion, and results for all the test portion have been estimated without refitting the model in each iteration:

Table 7: Random Forest performance scoring results (original frequency data).

	$RMSE$	$\bar{R}^2$
PKF All columns	21.7028	0.9654
PKF Lags	21.7274	0.9654
PUF All columns	162.3448	-0.9353
PUF Lags	75.5406	0.5812

Table 8: Random Forest performance scoring results for the first 10 days.

	$RMSE$		$\bar{R}^2$	
	No refit	**Refit**	**No refit**	**Refit**
PKF All columns	23.0795	23.0795	0.9124	0.9124
PKF Lags	22.8848	22.8848	0.9163	0.9163
PUF All columns	209.2612	47.7116	-6.1991	0.6258
PUF Lags	48.7061	48.4944	0.6209	0.6242

The results of the above tables show two significant points: the first one is that when predicting unknown future for the whole test portion (Table 7), using just lags improves the RMSE results by a factor bigger than x2 compared to the use of all columns. The second is that when focusing on just the first 10 days (Table 8), there's almost no

difference on the performance scorings of the models that just use lags either refitting each iteration or not. It will be discussed later, but this could mean huge savings in computational costs. The results of the first 10 days (extracted refitting each iteration) have been more deeply studied, its residuals showing a bias towards negative values. The time window that shows the smaller RMSE is the one that spans for 10 days.

When studying the daily grouped data for predictions of unknown future, again just using lags yields better scorings than using all columns.

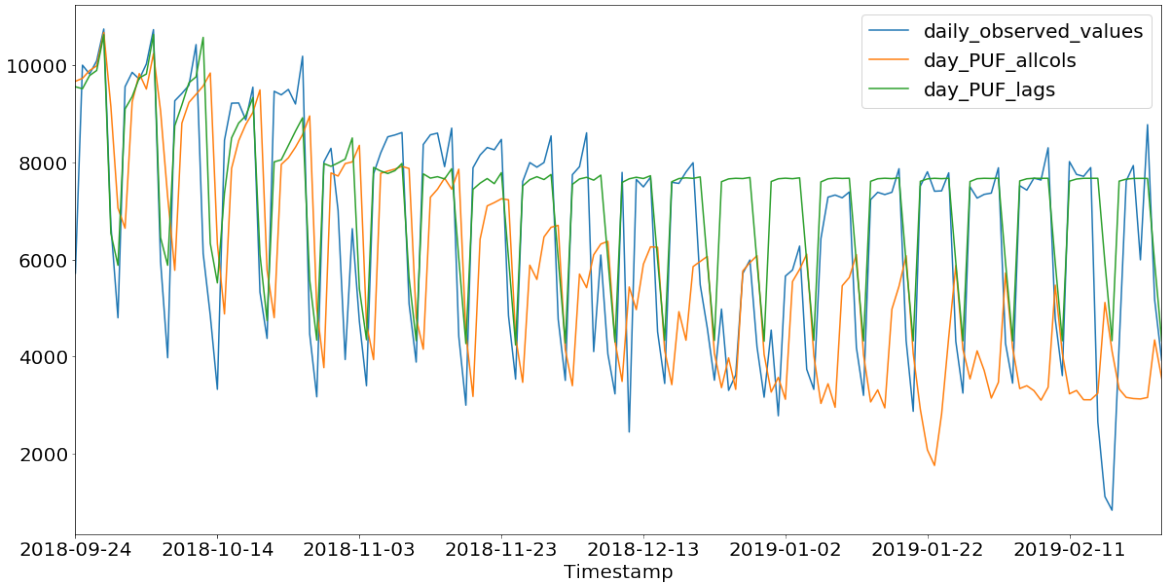

Figure 6: Random Forest daily grouped data predictions for unknown future.

In **figure 6** it can be observed that after the first two months of predictions, the model fitted just with lags loses its ability to capture the ups and downs shown by the observations, falling in a repetitive pattern over time. When comparing both frequencies for the overall test data just using lags, the Random Forest algorithm shows a better RMSE performance using the daily frequency than the high frequency.

Table 9: Comparison of Random Forest performance scoring for both frequencies.

	RMSE
48 obsv./day	2782.7830
1 obs./day	1525.6177

The results lead towards the same conclusion when comparing the performance scorings of time windows of different span and RMSE of future specific days:

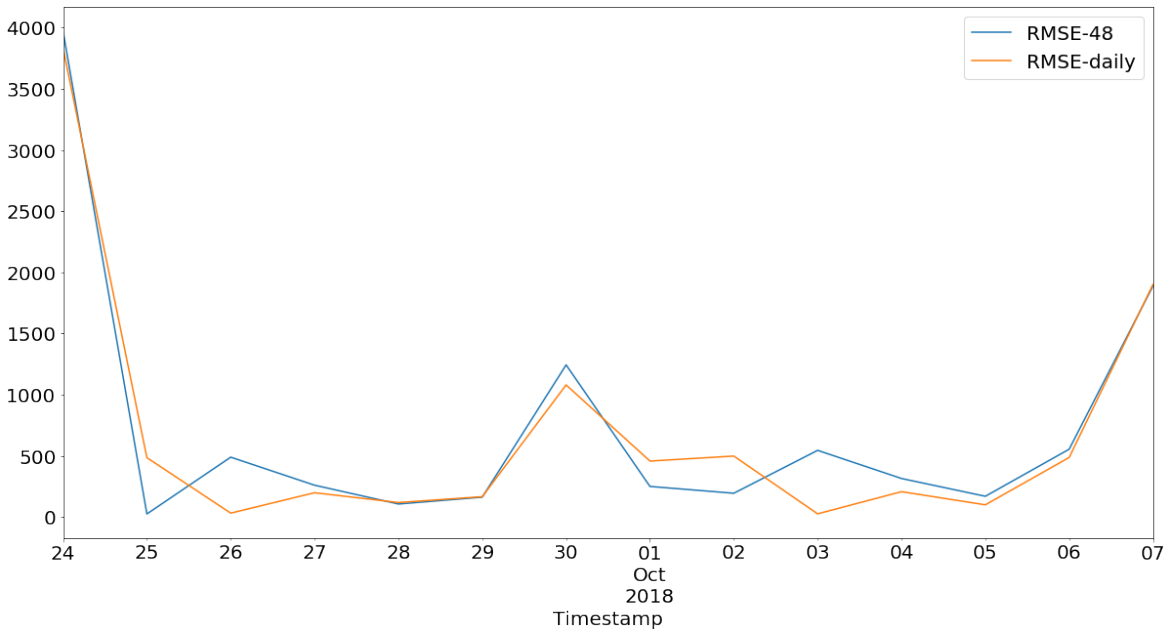

Figure 7: Random Forest RMSE curves for both frequencies of the first 14 days of predictions.

As previously stated, better performance scoring predictions using low frequency data mean a most significant gain in computational effort:

Table 10: Random Forest runtime (in hours) for each scenario.

	48 obs./day	**1 obs./day**
PKF All columns	0.1033	0.0001
PKF Lags	0.1074	0.0001
PUF All columns	1.5493	0.0188
PUF Lags	1.3247	0.0188
PUF All columns (10 days)	50.3885	
PUF Lags (10 days)	50.5996	

Even though refitting the model in each iteration improves the performance scorings[1], the difference is minimal (even more when considering the runtime required). The conclusion, then, is that the model that consumes daily data using just lags would be the chosen one.

[1]See Table 8.

5 Conclusions

The optimal scenarios selected for each algorithm have been:

- **Linear regression:** high frequency data using all columns.
- **ARIMA:** low frequency data using lags.
- **Random Forest:** low frequency data using lags.

When comparing the RMSE of each model for the first 15 days, the Random Forest algorithm seems to be the one that performs better:

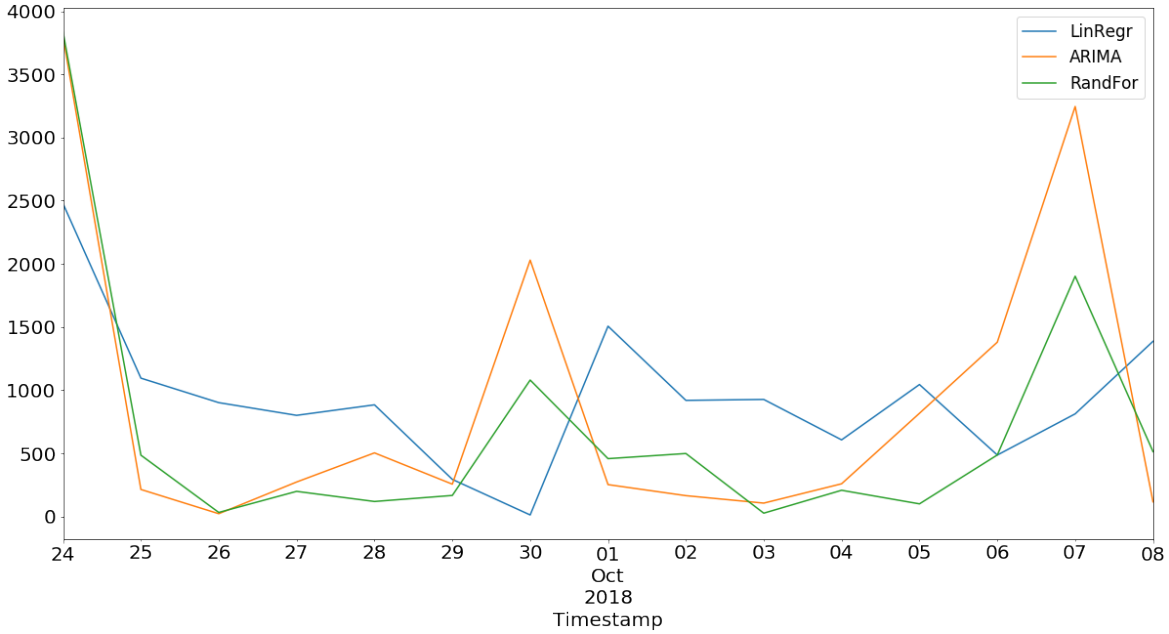

Figure 8: All algorithms RMSE curves for the first 15 days of predictions.

The first day of predictions shows a strong deviation from the observation in all algorithms. If we check the RMSE for the next days discarding the first one, the results are conclusive:

Table 11: RMSE of all algorithms for the next 2 to 15 days

	RMSE
Linear Regression	917.0444
ARIMA	1129.1207
Random Forest	660.9301

Random Forest is the algorithm that yields the most accurate predictions. It has been demonstrated that there's not a general rule to apply for all algorithms; each of

them works better with a precise frequency and benefits from the use of external data in a different way.

As previously stated, due to calendar reasons it has not been possible to test the different algorithms and scenarios for the data of each location. It would be interesting to do so in the future, because the Raval Nord and Jaume I data spans for just a few months and the conclusions drawn from their study might be different than the ones extracted from the Maragall exploration. It would also be desirable to present the results as an interval (with a defined confidence) instead of a unique value. That feature is not available in all the algorithms used.

The analysis of the residuals (for the whole test portion) shows the same traits in all the algorithms used:

- Their variance seem to increase over time.
- When analyzing their distribution, the percentile 50% is always negative.
- There's a positive correlation when plotting the residuals against their fitted values.

Those features might be hints of possible ways to improve the ongoing project. Another enhancing strategy could be to use *power transforms*, with methods like the Box-Cox transform[1]. Data transforms are intended to remove noise and improve the signal in time series forecasting.

All the analysis have been conducted following the rule of not using domain specific features for the predictions. In the future, if the service is finally implemented, it would be interesting to tailor the predictive models to each domain/client because it would potentially enhance the accuracy. The current project has been developed using data from public health buildings, which has a specific time structure and strong week cycles. The same principles might not be applicable for other domains and therefore, special care should be taken when making assumptions for different customers.

6 Bibliography

Athanasopoulos, G., & Hyndman, R.J., *Forecasting: Principles and Practice (2nd edition*, 2018.

Brownlee, J., *Introduction to Time Series Forecasting with Python*, 2018.

Fulcher, B. D., *Feature-based time-series analysis*, 2017.

Mélard, G., *Forecasting Daily and High-frequency Data*, 2013.

[1] https://en.wikipedia.org/wiki/Power_transform