**Recruit Restaurant Visitor Forecasting**

*Author(s):*

Ayala Shvarzman

Arnon Kleinman

# Introduction

The running of a successful and profitable restaurant is not easy to say the least...

Even if you hired the right cooks and staff, the food is delicious, your location is prime, and your service level is excellent, you could still go under.

Many successful restaurants go belly up, it's a difficult to succeed in this business, and you need to save your money wherever you can.

We decided to make a predictive model that will help cut expenses and save money.

Our focus is the on number of customers a restaurant should expect each day, and by so to save a lot of money on buying excess raw materials that will be dumped at the end of the day, or by reducing the use of unnecessary labor, and by planning campaigns in advanced to attract customers on weaker days.

This paper examines the number of customers restaurants in japan have each day.

We were provided with data from two sites, "Hot Pepper Gourmet" (a restaurant review service) and "AirREGI" (a restaurant point of sales service).

"Hot Pepper Gourmet" (hpg): similar to Yelp, here users can search restaurants and also make a reservation online.

"AirREGI / Restaurant Board" (air): similar to Square, a reservation control and cash register system

# Methodology (Project design)

## Data

The data includes number of visitors per restaurant per day, number of reservations through the two sites, information about the restaurants, location, area, what kind of food it serves, we were also provided with information about the date, is it a holiday or not.

We added weather data from the web, temp', wind speed, precipitation, cloud cover, etc..

Since we couldn't contact "AirREGI" and "Hot Pepper Gourmet", we had no way of validating their data, The weather data is from Japan Meteorological Agency and we couldn't find other accurate and detailed weather data to make validation tests on the data.

**Clear outcome variable definition:**

the y variable, which we wish to predict, is defined as the future restaurant visitors totals on a given date. We will provide, as a result, a table of three rows: restaurant id, date and number of visitors.

A source for bias that exists in our data: random day (not holidays or reoccurring) that a specific restaurant suddenly has a very large amount of visitors, also holidays are a source of bias in general. That is why we have a special categorical variable tagging a date as a holiday or not.

**Variable Engineering:**

**Original data**

The project will be based on a time frame beginning of 2016 starching to middle 2017:

The training period is from 01/01/2016 until 31/03/2017.

The test period is from 01/04/2017 until 22/04/2017.

Our raw training data consists of about ~252,000 rows of data, and our key is a combination of date and restaurant id. About 28,000 rows included reservation data of some sort. Because so much reservation data is missing, we decided to divide our data to two different sets, meaning two different flat tables. One includes reservation data , one don't.

Data from the competition csv files have been combined in SQL to one csv file that contains the fields:

(air\_store\_id, air\_genre\_name, air\_area\_name, latitude, longitude, visit\_date, visitors, reservations, daily\_reservations, weekday, holiday\_flag).

The weather data was from 1663 weather stations that collected data between 01/01/2016-31/05/2017, each weather station had fields of longitude and latitude.

We added the weather data by using kmeans to create 10 area clusters and giving each restaurant weather data based on its location.

(specified in our data retrieval protocol),

We analyzed the data and several of our initial columns were divided using one hot encoding to categorical variables (will be discussed above at the variable engineering section).

**One hot encoding**

In addition, we also split categorical variables into binary indicator variables, as one of our models (GLMNet) could not handle categorical data. These variables include:

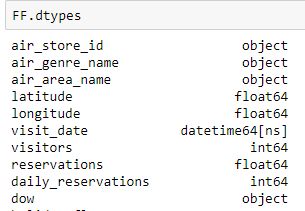
Weekday, split to 7 categories, day of month split to 12 categories, genre split to 14 types of restaurant genres, and finally the region (location of the restaurant) split to 14 different genres.

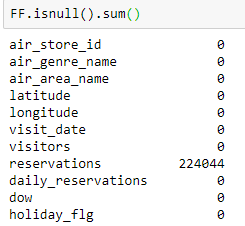
We will further explain in detail our one hot encoding process on the next chapter (feature and variable selection).

The Inclusion criteria, is the restaurants that exist in the "air\_stor\_info" table.

EDA.

First, we used ".dtypes" to check that each field is defined correctly, then we checked for missing data, not only Nulls and Nans, but also checking that a field doesn't contain data that doesn't belong to it, for instance, that a numeric field doesn't contain text which will not show on a regular .isnull() test.

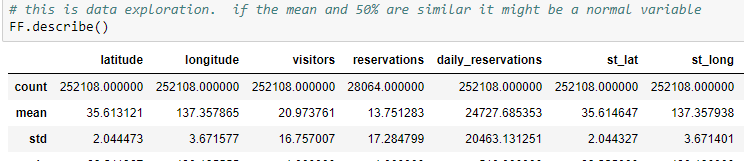




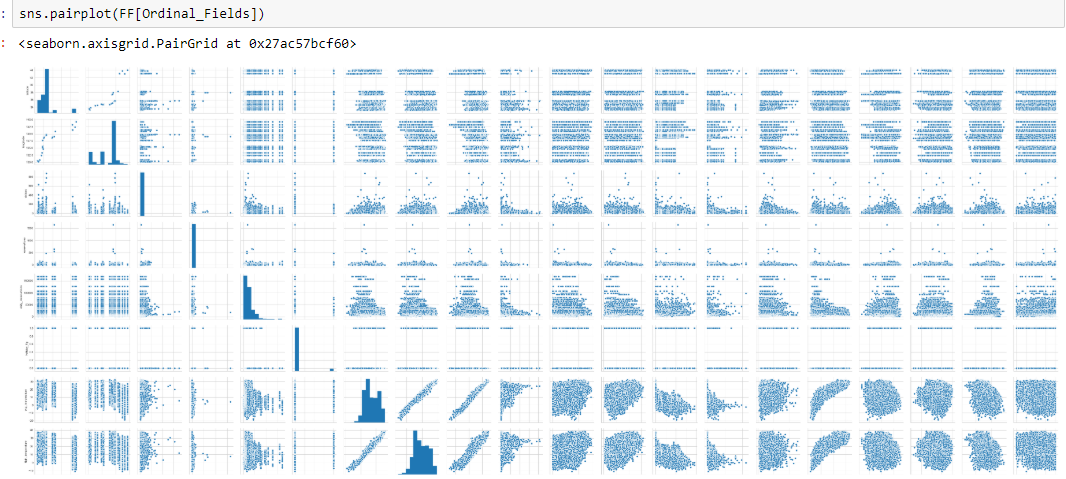
We had no missing data part for "Reservations" field, which had close to 90% missing data.

All the fields that were supposed to be continuous were.

Then we used ".describe()" to further explore the data, to get some feel of the data, trying to find fields the might distribute normally.



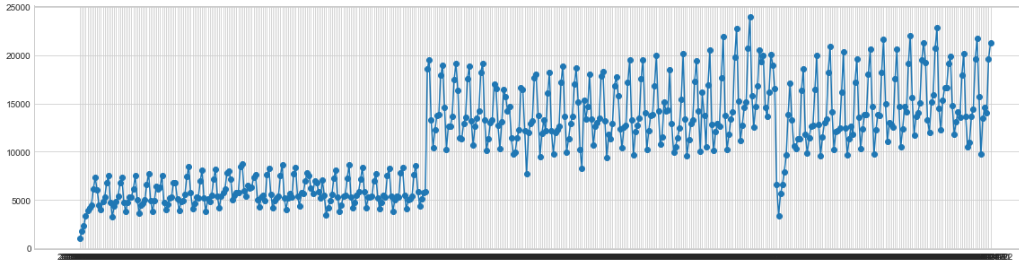
We used "pairplot" to visually look at correlations between fields and see fields distribution.



While checking for missing data, we found a lot of missing data in the "reservations" field, close to 90% missing, if it wasn't such an important field we would've let it go, instead we decided to divide our project into two models, one only for the records that contained reservations data and another without reservations data.

We continued to analyze the data.

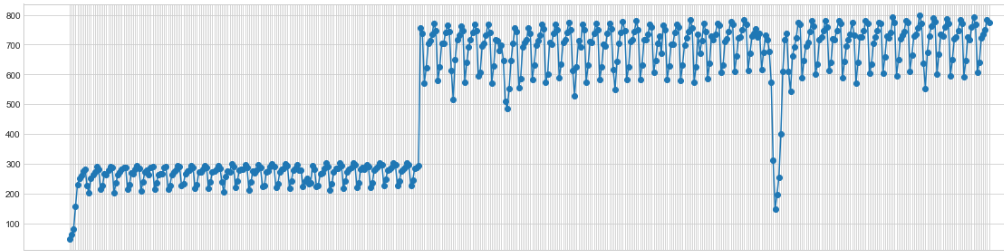
**Number of visitors each day**



At first glance, it appears as if somewhere during mid 2016 there was a jump in visitors number.

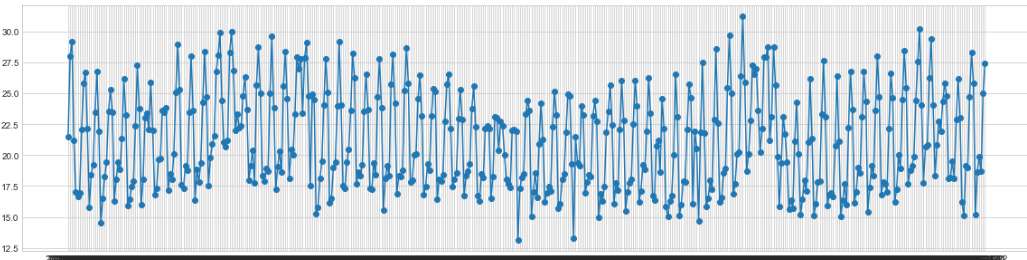
A second graph helps explain the sudden change.

**Number of different stores visited each day**

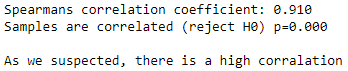


By looking at the two graphs above, we suspect that there's high correlation between 'number of different stores' and 'number of visitors'

**Average number of visitors each day**

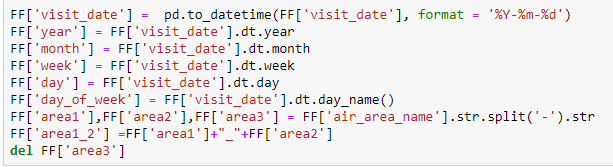


We also used spearman to calculate the correlation.

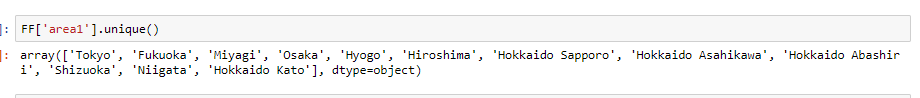


We created new fields to further explore the data.

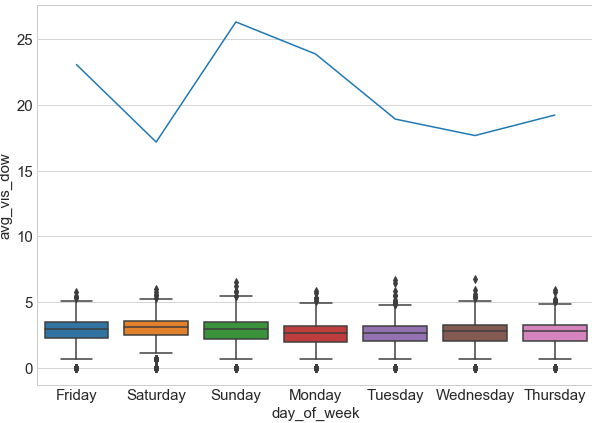
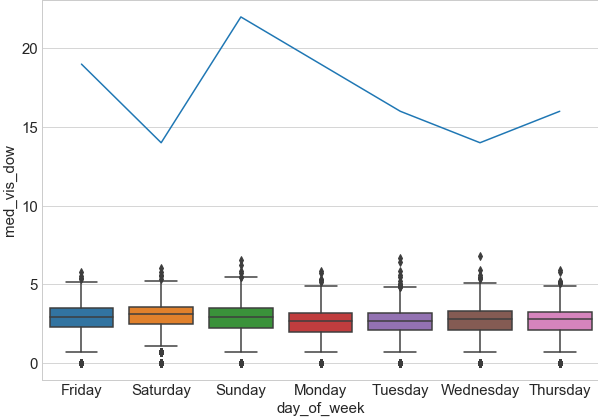
We broke visit date into (year,month,day,weekday), and air area name into (area1,area2,area1\_2).



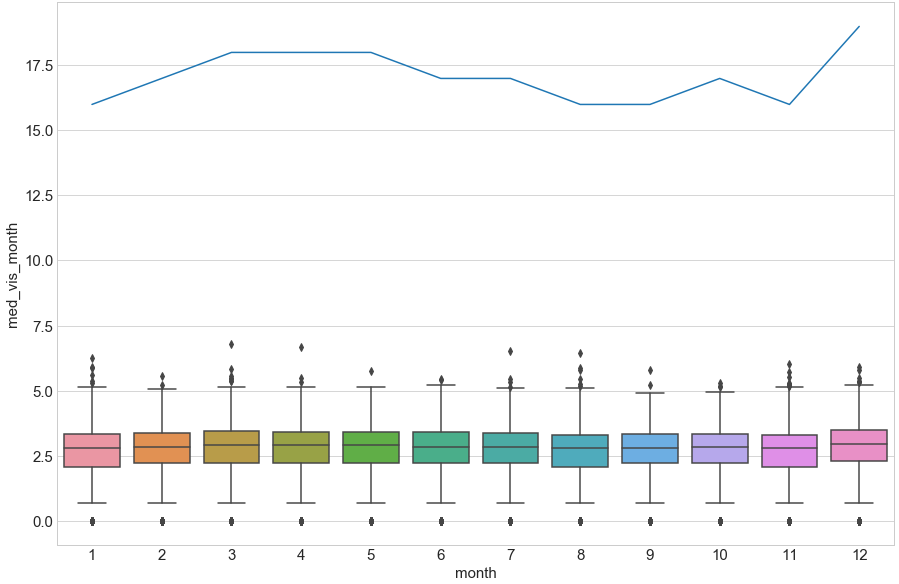
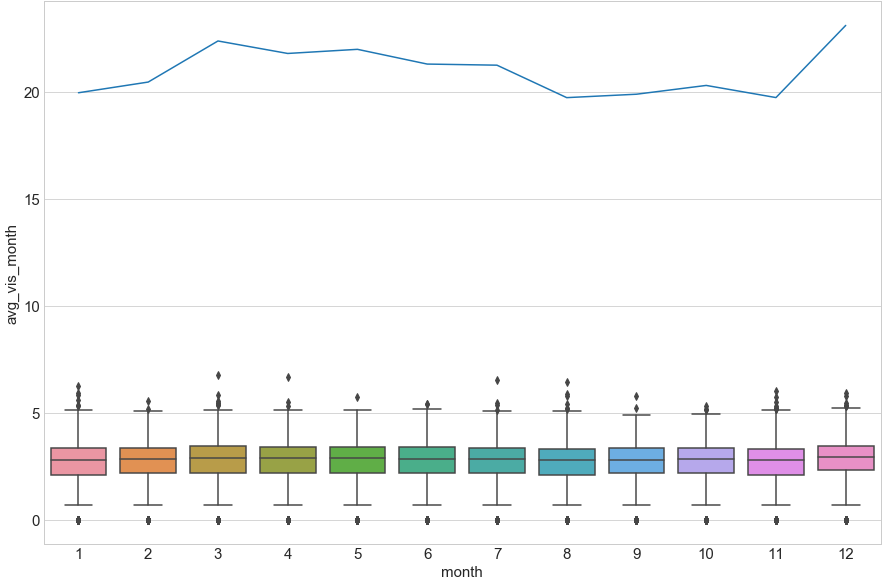
Dividing the region column and focusing on the first part, the county:



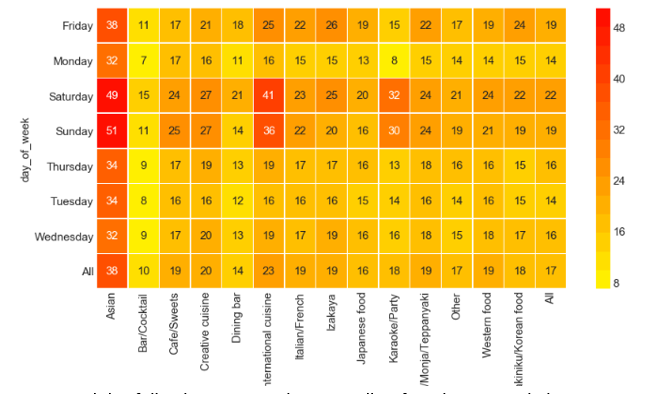
Visitors: Average-Weekday Median-Weekday



Visitors: Average-Month Median-Month

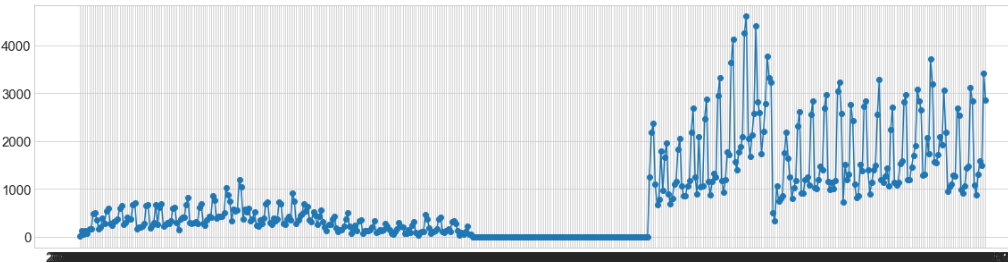
 

The following was a try we did to put in a table the median number of visitors as a function of genre and day of week, in order to study the traffic changes over these parameters:

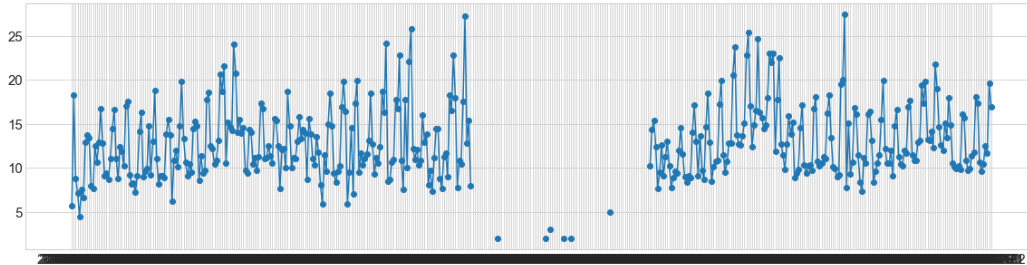


Exploring the "reservations" field.

**Total number of reservations each day**



**Average number of reservations each day**

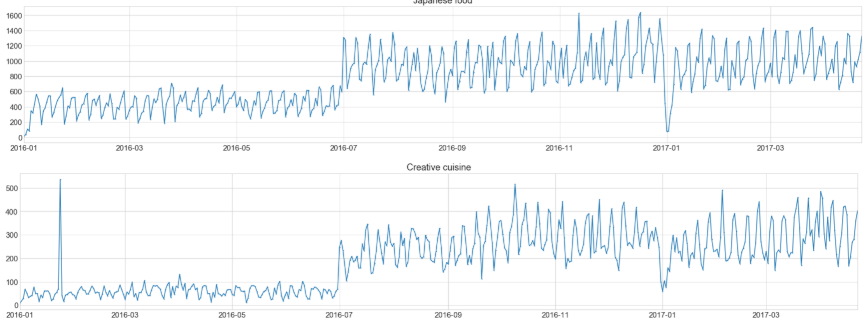


There appears to be a range of dates without any data at all.

After deliberation we concluded that there was probably a problem with the reservation control system, and no data was accumulated during that time.

By dividing into two models we solve that problem.

**Graph of number of visitors per food genre by date**

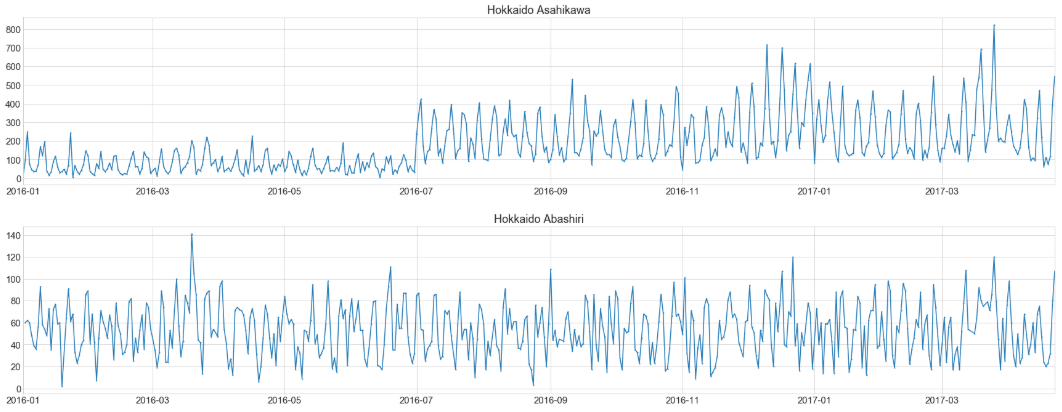


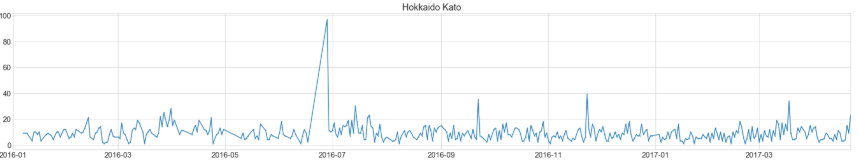


\*Showing only an example.

Some food genres got into the system later on 2016.

**Graph of number of visitors per prefecture by date**





\*Showing only an example.

We made all sort of cutting and dicing fields for visitors data,

Average and Median of all the combinations below:

Average Daily Visitors

Median Daily Visitors

Average Visitors per ['day\_of\_week']

Average Visitors per ['air\_genre\_name']

Average Visitors per ['area1']

Median Visitors per ['day\_of\_week']

Median Visitors per ['air\_genre\_name']

Median Visitors per ['area1']

Average Visitors per ['day\_of\_week','holiday']

Average Visitors per ['air\_genre\_name','holiday']

Average Visitors per ['area1','holiday']

Median Visitors per ['day\_of\_week','holiday']

Median Visitors per ['air\_genre\_name','holiday']

Median Visitors per ['area1','holiday']

Average Visitors per ['day\_of\_week','area1','holiday']

Average Visitors per ['day\_of\_week','area1','air\_genre\_name','holiday']

Average Visitors per ['day\_of\_week','area1']

Average Visitors per ['day\_of\_week','area1','air\_genre\_name']

Median Visitors per ['day\_of\_week','area1','holiday']

Median Visitors per ['day\_of\_week','area1','air\_genre\_name','holiday']

Median Visitors per ['day\_of\_week','area1']

Median Visitors per ['day\_of\_week','area1','air\_genre\_name']

Average Visitors per ['air\_store\_id']

Median Visitors per ['air\_store\_id']

Average Visitors per ['air\_store\_id','day\_of\_week']

Median Visitors per ['air\_store\_id','day\_of\_week']

We've also made "one hot encoding" of the category fields:

'day\_of\_week'

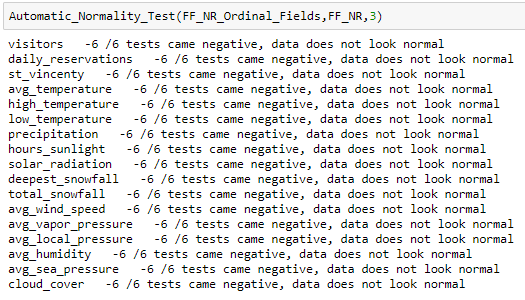
'air\_genre\_name'  
 'area1' and 'area1\_2'

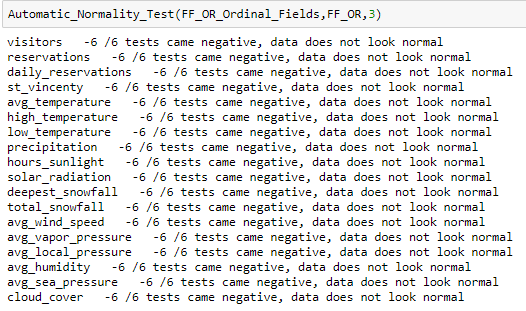
Now we split into two models, one only for records with reservations data and another without reservations data.

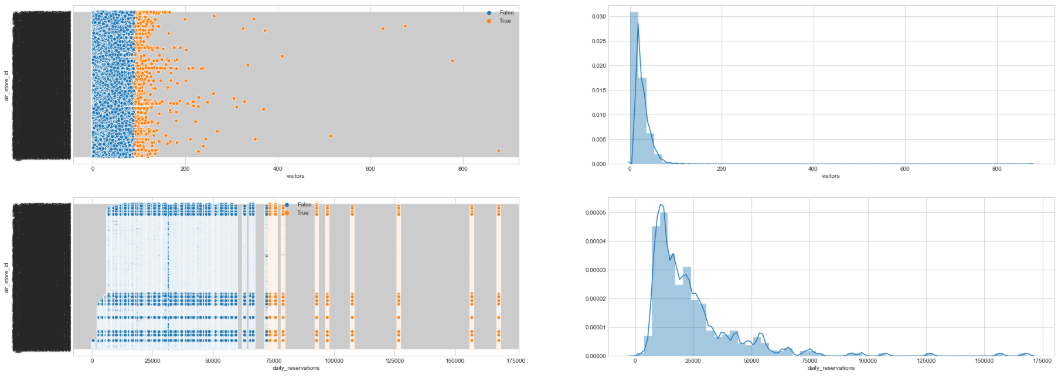
From the pair plot it appeared that "avg\_humidity" and "avg\_sea\_pressure" distribute normaly, but using Z-score on them gave a high number of outliers.

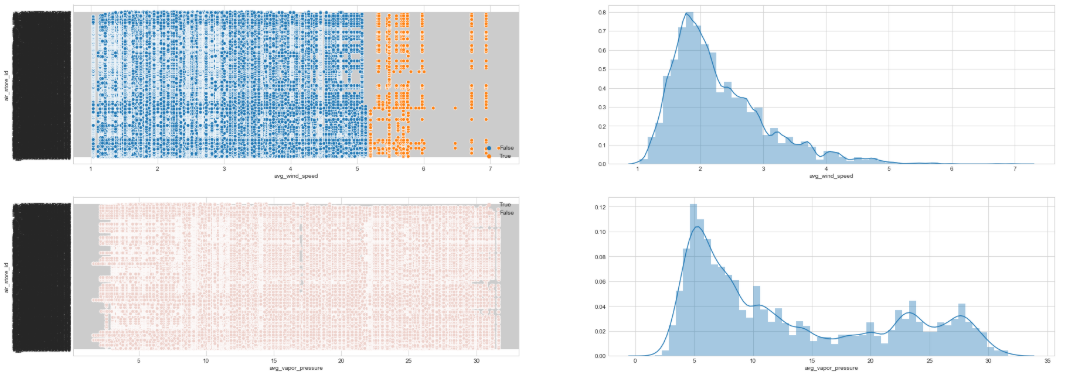
So, we used statistical normality tests.

In both models, none of the fields had normal distribution.



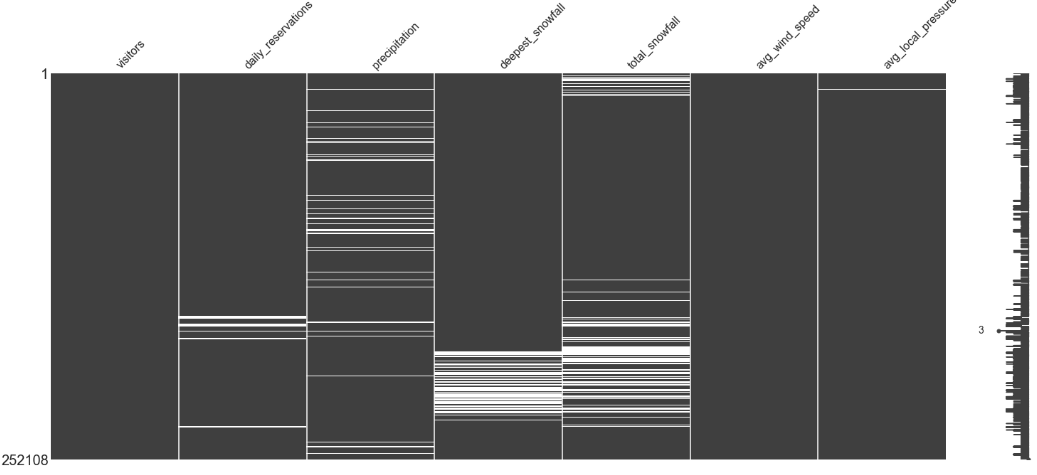
   
We used InterQuartileRate to find Univariate outliers and we used scatterplot and distplot to dicide the right cutoff.

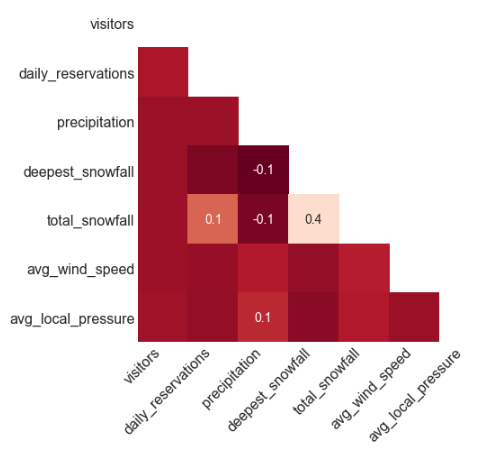


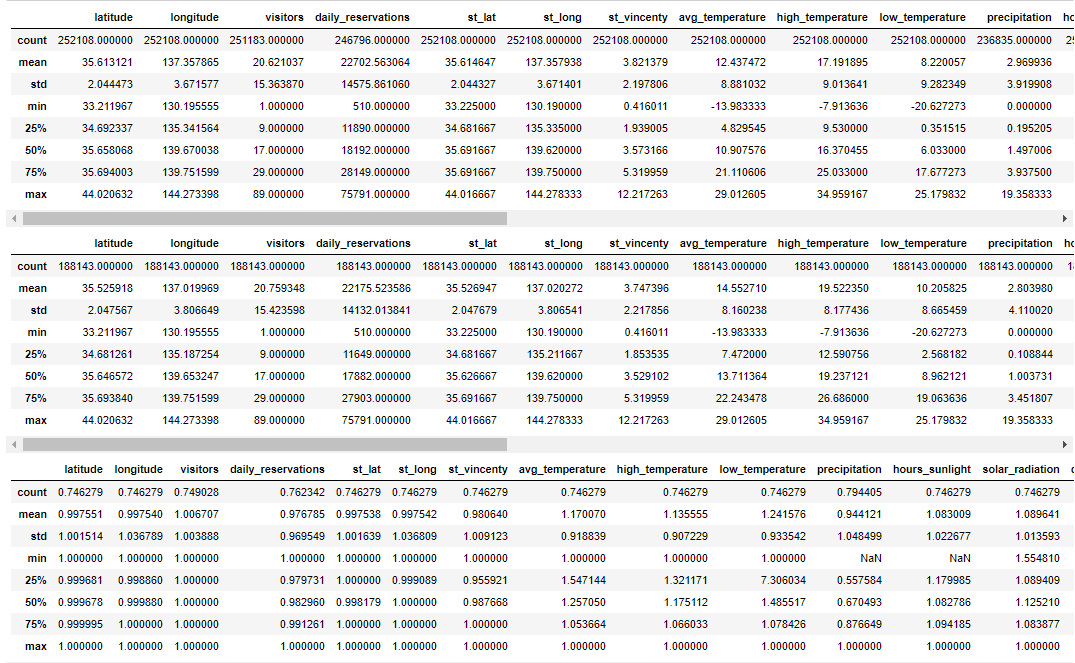


\*Showing only an example.

After detecting the outliers we replaced them with Nan's and used Missingno and Heatmap to see their distribution and Correlation in the data.

We also used Describe (after dropping the rows with missing) in order to see the affect on the data.

  
There are three describe tables, (1) before the rows with missing have been removed, (2) after the rows with missing have been removed, and (3) a division of one in the other, to show the change in percentage.



Some fields had a change of more then 10% so we decided not to drop the rows and imputate the missing data.

We choose to use Knn algorithm to impute the missing data.

This process has been done on both models.

Then we went on to do multivariate outliers detection, we choose to use dbscan.

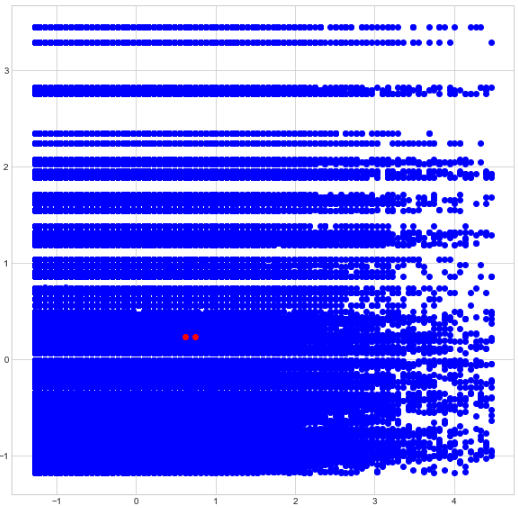
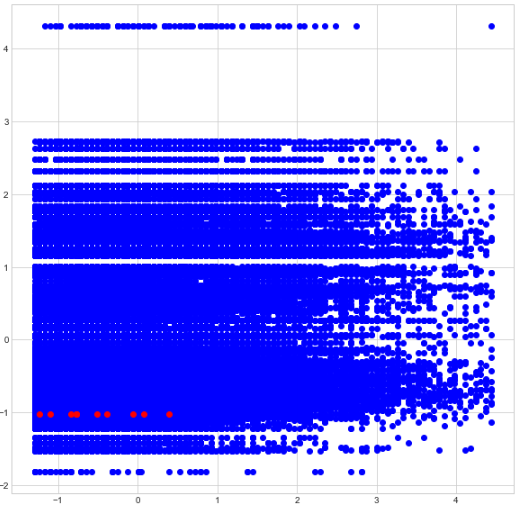
On the first model we had 252,108 rows and 16 columns, and we didn't had sufficient computing power to run all the data at once.

So we divided the data to four groups (A,B,C,D) and run combined permutations:

AB, AC, AD, BC, BD, CD. Each group overlaps 3 times.

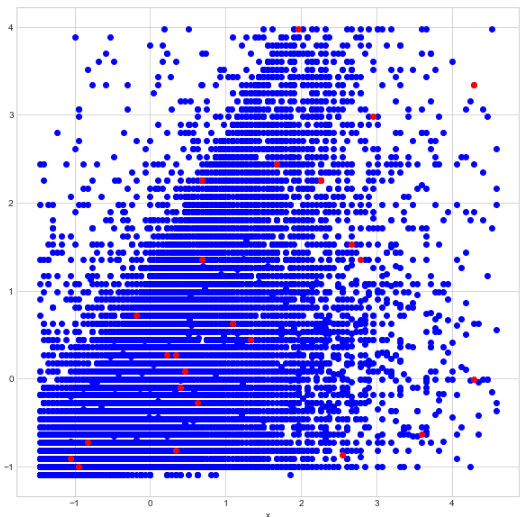
Only rows that were identified as outliers on all 3 check were designated as outliers.

(before treating univariate outliers, this check found 47 outlier rows, but after it found only 2).

On the second model, there were only 28,064 rows and 17 columns so we could run the whole data.

It found 24 outlier rows (it was expected, because of the small amount of rows).



We removed the outlier rows found and continued to feature selection.

(no point in checking the effect on the data of removing 2 out of 252,108 rows)

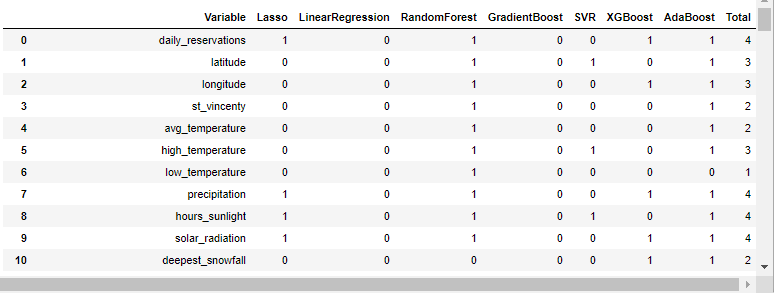
(forgot to check the effect on the data of removing 24 out of 28,064 rows,

and due to lack of time didn't go back and reevaluated the data)

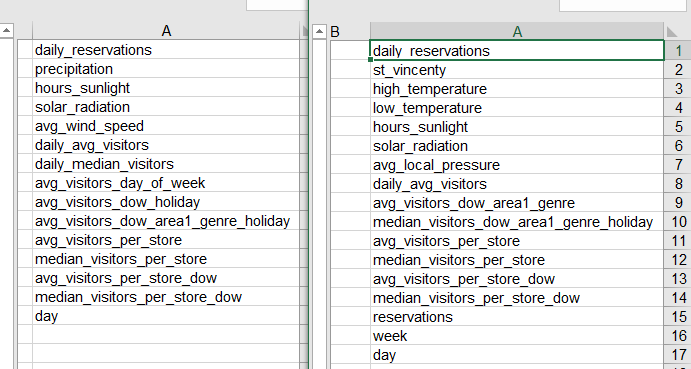
Feature Selection.

We run 7 Feature Selection models:

* LASSO (data was standardized before)
* Linear Regression (data was standardized before)
* Random Forest
* Gradient Boost Regressor
* SVM Regressor
* XGBoost Regressor
* AdaBoost Regressor



These are the variables that were selected by the models:



Link to data retrieval protocol.

## Models

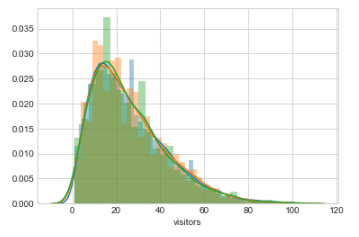
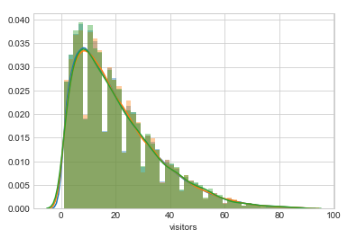
**Train Dev Test split.**

We split our training data into 3 groups using randomized split.

* Train 70%
* Dev 20%
* Test 10%

We are going to use tbl1 to check that the data is divided correctly and well balanced, by creating a new column that assigns each row the name of its and test with tbl1 to see if any field p value is below 0.05.

We also used sns.distplot to see the data distributions.



**Predictive Models.**

Our target optimization metric is the Root Mean Squared Logarithmic Error.

The RMSLE is calculated as:

Where:

**n** - is the number of observations

**pi** - is our predicted count

**ai** - is the actual count

**log(x)** - is the natural logarithm of **x**

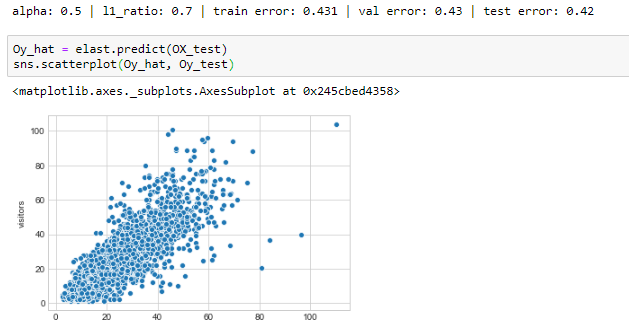
We seek to identify the models that result in predictions which minimize this error.

We will use regression models to predict our outcome.

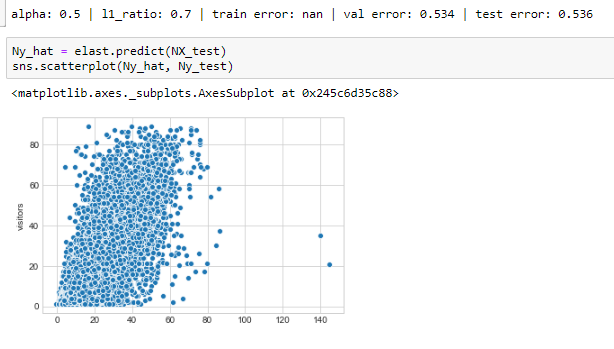
We choose to try ElasticNet – L1/L2 regularization and SVM Regression.

We will use our best model.

ElasticNet best parameters for reservations only data:

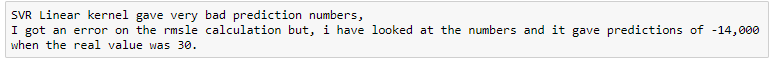


ElasticNet best parameters for no reservations data:



SVR, tried to find which kernel gives the best results:

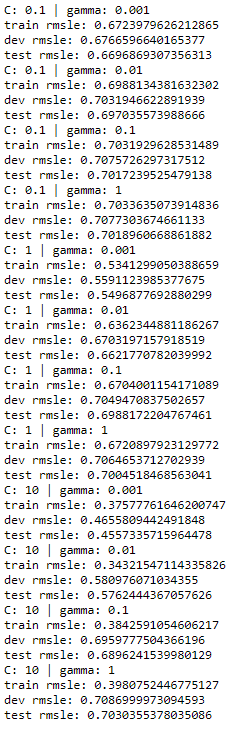
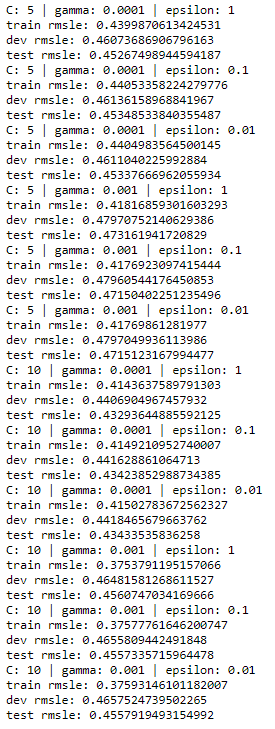
Tested "rbf", "linear" and "polynomial".





So, only "rbf" gave us a viable error, so we choose it for parameter tuning.

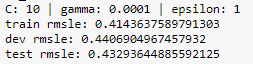
We looped through "C", "gamma" and "epsilon" hyperparameters on both reservations only data model and no reservations data model.



We found that the best parameters for both are:

C = 10, gamma = 0.0001, epsilon = 1

reservations only No reservations



We choose SVR because it gave better results.

Run the model on the data that we wanted to predict and got:



**Analysis on the final results:**

* We tried to avoid over fitting, and we guess we did a bad job at that.
* Due to lack of time We didn't check more predictive models.

**Conclusion**

Honestly, I'm not sure what to write…

I've learned a lot during the development of this project but not enough.

It is unfortunate that we didn't have more practice during the course, most of the things we did for the first time was in the project, and we were left with ourselves and the internet for answers, under the pressure of limited time.

I have still a lot of questions about certain stages of the process,

For instance, if I did the feature selection correctly, I'm also not sure I did more good then bad with the outliers treatment, but I guess there is only one way to learn, you have to try and see the results.

Another thing that I know I didn't do the right way is the part I split the multivariate outliers detection using dbscan, I should have divided the data by columns and not by rows.

I am going to try and better this model, and each time I'll make a change that will better\worsen the results, I'll learn.

Appendix:

Data retrieval protocol.

