Extending prediction of parking occupancy to untracked city areas using GIS metadata
- attempt of Proof of Concept -

20th February 2017

1 Introduction and Goal

The purpose of the following is to show the technical realization of the thesis on a small scale. More precisely, a practical step-by-step process will be presented, by which prediction models for areas with untracked parking spots are built. First, the data is clustered around current occupancy rates. For every resulting cluster a model will be built with the occupancy rate in 1 hour as target variable. A model will then be applied to data entries that have been associated to a nearby cluster. A short evaluation will follow.

We start off by describing the setup, the available data and services used (section 2). Next, the preparation of the data is shown (section 3), before using a Machine Learning service to build the models (section 4). The results are discusses in section 5 and the document ends with a short conclusion (section 6).

2 Setup

The data used here comes from the SFpark project in San Francisco[1]. The project involved installing sensors in different parts of the city to collect the parking occupancy data. The data spans roughly from 2011 to 2013, although in practice some sensors were out of order for some intervals. In addition, other data was collected on site, i.e. parking price, employment, weather, gas price, etc. The raw data is freely available to download and is spatially organised per blocks, i.e. the segment of a street between the intersections of two consecutive streets (see figure 1).

Time-wise the data is structured per full hour, i.e. as integers from 0 to 23. For this experiment only a fraction of the whole available data is used, more precisely only the **blocks**, **times of measurement** and **occupancy rates**. This however is enough in order to detect patterns of occupancy-time correlations, e.g. blocks that have a peak in the morning and evening, blocks that are peaking only in the evening and weekends, etc.

The platform on which the experiment was run is the Machine Learning service `http://bigml.com`. The site initially allows processing datasets of up to 16MB for free, by running up to 2 parallel tasks in the cloud. Soon after starting to use it, I received a free update for crunching 1GB datasets for 1 month. The ML methods available are Classification, Regression, Cluster Analysis, Anomaly Detection and Associative Discovery.

Uploading data sources can be done through the service itself or from another cloud storing service, in case of larger files. I made an Amazon S3 account and loaded my data sources there first. Then I provided the URL to the source to load it in BigML (see figure 2).

Figure 1: Blocks in a city; a block has more faces for each street; a street block is the spacial unit for the SFpark data; image source[2]

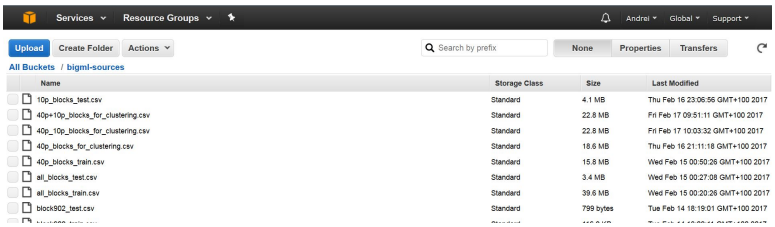

Figure 2: My Amazon S3 account with directory and file layout

3 Data Preprocessing

The dataset to be loaded on the BigML platform will contain blocks, times and occupancy rates. Since we are developing a prediction, for simplicity select prediction in 1 hour. Hence, the following pieces of information need to be extracted from SFpark data:

- block id
- date and time
- occupancy (current)
- occupancy in 1 hour

It turns out that the first two points are readily available in the SFpark parking occupancy dataset. The occupancy values are, however, indirectly represented by **Total occupied seconds**, **Total vacant seconds** and **Total unknown seconds** (see figure 3).

BLOCK_ID	STREET_NAME	BLOCK_NUM	STREET_BLOCK	AREA_TYPE	PM_DISTRICT_NAME	RATE	START_TIME_DT	TOTAL_TIME	TOTAL_OCCUPIED_TIME	TOTAL_VACANT_TIME	TOTAL_UNKNOWN_TIME	OP_TIME
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 00:00	169200	18927	150273	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 01:00	169200	19681	149519	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 02:00	169200	21784	147416	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 03:00	169200	22940	146260	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 04:00	169200	21600	147600	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 05:00	169200	21600	147600	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 06:00	169200	21610	147590	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 07:00	169200	23504	145696	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore		01.04.2011 08:00	169200	47368	121832	0	0
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore	2	01.04.2011 09:00	169200	57859	111341	0	162000
902	CALIFORNIA ST	24	California and Steiner Lot	Pilot	Fillmore	2	01.04.2011 10:00	169200	71471	97729	0	162000

Figure 3: Extract of the SFpark occupancy spreadsheet; selected colums are interesting here

Each of these values is the sum in seconds of all the sensors on the respective block that reported car occupation, vacancy or unknown status. The latter basically means that the respective sensor was out of order at that time. Following the SFpark evaluation guide, to calculate the occupancy rate, the unknown time will not be included in the total time at the denominator:

$$\text{Occupancy Rate} = \frac{\text{Total occupied}}{\text{Total occupied} + \text{Total vacant}}$$

As far as time is concerned, we want to detect patterns with regard to time of day, day of week and perhaps even calendar week. Therefore these values are extracted from the provided date+time stamp. A snapshot of the final dataset to be uploaded in BigML looks as follows. It contains about 1,048k lines and is 51MB large (see figure 4).

	A	B	C	D	E	F	G
1	BLOCK_ID	START_TIME_DT	CALENDAR_WEEK	DAY_OF_WEEK	HOUR	OCCUPANCY	OCCUPANCY_1H
2	902	01.04.2011 00:00	14	6	0	11.1861702	11.63179669
3	10202	01.04.2011 00:00	14	6	0	100	100
4	10302	01.04.2011 00:00	14	6	0	67.4444444	74.86574074
5	10403	01.04.2011 00:00	14	6	0	84.187037	62.42037037
6	10702	01.04.2011 00:00	14	6	0	66.6666667	61.96296296
7	32608	01.04.2011 00:00	14	6	0	21.7892157	19.32679739
8	32800	01.04.2011 00:00	14	6	0	55.7769097	44.12934028
9	33002	01.04.2011 00:00	14	6	0	60.0661376	
10	33004	01.04.2011 00:00	14	6	0	36.1239712	27.67798354
11	33005	01.04.2011 00:00	14	6	0	42.470679	
12	33103	01.04.2011 00:00	14	6	0	8.82352941	8.843137255
13	33104	01.04.2011 00:00	14	6	0	24.5570437	34.19097222
14	33105	01.04.2011 00:00	14	6	0	93.5483871	93.36536117
15	46404	01.04.2011 00:00	14	6	0	55.0432099	
16	902	01.04.2011 01:00	14	6	1	11.6317967	12.87470449
17	10202	01.04.2011 01:00	14	6	1	100	100
18	10302	01.04.2011 01:00	14	6	1	74.8657407	83.33333333

Figure 4: Extract of the CSV file that is ready for processing; notice that some occupancy in 1h values are not available due to temporary not functioning sensors

4 Building the models

In the first step, a CSV file is loaded (possibly via Amazon S3) in BigML under **Sources**. At this point, one can configure the datatypes of the fields. Blocks is Categorial, Date+Time stamp is Datetime, while all the others are numerical (see figure 5).

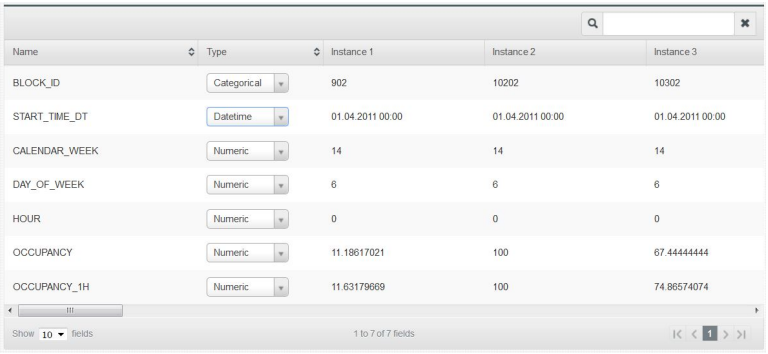

Figure 5: Setting datatypes for the fields in the Sources area in BigML

Next step is the creation of a proper dataset, that BigML internally uses for further operations. In our case it is identical to the source (see figure 6).

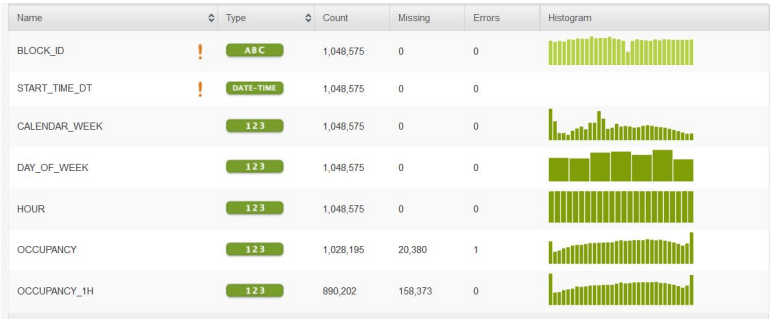

Figure 6: The view on the dataset in BigML

From the dataset one can start a K-means clustering with 8 clusters. The fields to be taken in account for clustering are **calendar week**, **day of week**, **hour** and **occupancy**. Missing values will be filled with a default value. The result is displayed as follows (see figure 7).

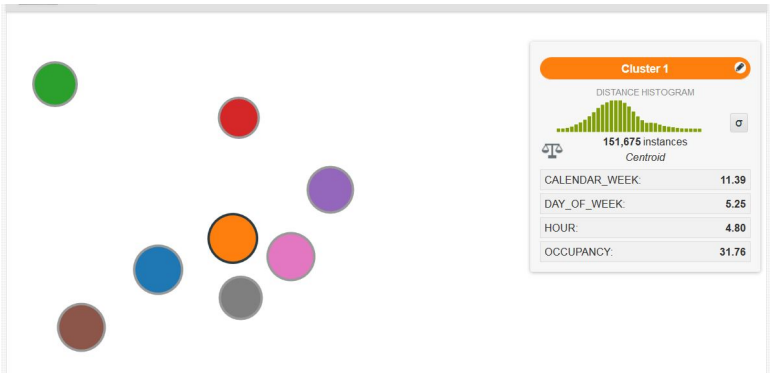

Figure 7: Clusters visualisation in BigML

The assumption here is that nearby clusters have a similar time-occupancy pattern. The following steps will build upon this assumption. In case the assumption is false, the results should reveal this fact.

A prediction model is built for the center cluster (in the figure above, Cluster 1). The target variable is **Occupancy 1H**. BigML uses decision trees for this operation (see figure 8).

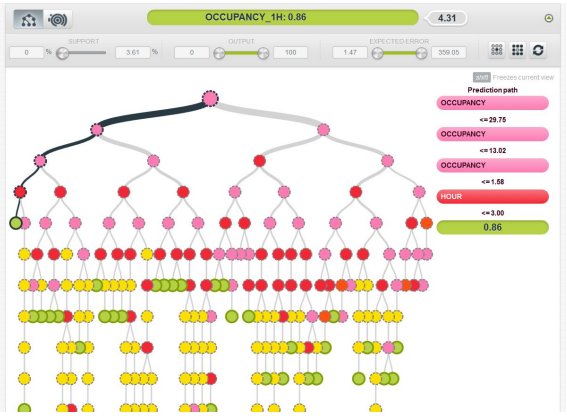

Figure 8: Visualisation of a prediciton model as a decision tree in BigML

This model is used for prediction on the other datasets represented by the surrounding clusters. Their **occupancy in 1h** values are ignored during the process and will be used afterwards to calculate the prediction accuracy.

5 Evaluation

The accuracy is calculated as the average absolute difference between the measured **occupancy in 1h** and the predicted one:

$$\text{Model Prediction Error} = \frac{\sum_{\text{record}} |\text{Ocuppancy1h}_{\text{real}} - \text{Occupancy1h}_{\text{predicted}}|}{\text{number of records}}$$

Here is the error calculated for each cluster, as described above:

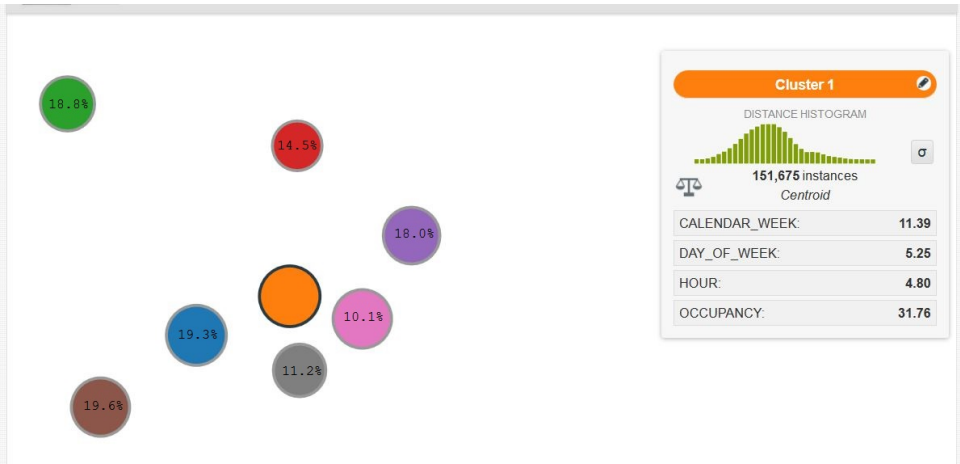

Figure 9: Annotated cluster image with the aveage absolute error for each cluster dataset apart from the center cluster

In looking at this result, there is a slight tendency to be seen: the further away a cluster is from the center cluster, the larger the error when applying the model of its central cluster on its dataset. I would have wished that this tendency is clearer, however, by adding other attributes to the datasets, e.g. parking price, traffic, weather and, of course, GIS metadata, such a tendency may become clearer.

I also run another comparison. I created a dataset with 90% of the data, created a prediction model using the same Decision Tree method and having **occupancy in 1h** as target variable. I then applied it to the rest 10% of the data. The prediction error in this case was 9.3%. Note that the data is split time-wise, so that blocks will be present in both training and test datasets.

Despite the general model delivering a better result, I believe that by enriching the datasets with other attributes, the smaller clusters will build more fitted models that will improve on the error of the whole model.

6 Conclusion

By using clustering as a pre-processing step, a similarity function is internally computed between the future prediction models. This is equivalent to the similarity measure described in the thesis idea from a previous document.

This experiment has shown that such an approach delivers decent results when the models contain just basic attributes. By adding more attributes, the clustered models will be more accurate and I believe that their preciseness will surpass the one of a general prediction model.

References

[1] *SFpark project*. `http://sfpark.org`. Accessed: 2017-01-30.

[2] *Japanese addresses: No street names. Block numbers.* `http://sivers.org/jadr`. Accessed: 2017-02-19.