IT'S ALL ABOUT LOCATION, LOCATION, LOCATION.

Introduction

The aim of this paper is to assess the attractiveness of a location to open a restaurant using machine learning algorithms. I believe these models can be of particular interest for a screening approach, either before opening or buying an existing place.

The data use to assess this attractiveness are publicly available. I adopted a naïve approach where I only tell the model if whether or not restaurants are present, but not how many. The reason is that a lot of locations do not have restaurant and some have a lot (up to 156). This inequal distribution raises issues for some regressions. Surprisingly, the number of false negatives was pretty low for the area with more than 3 restaurants. While focusing on the Chicago (IL) urban area, the data are available for the whole United-States.

Data source

In order to find these possible factors, three publicly available sources are combined: Census data, Safegraph, and Foursquare data.

Safegraph

The Chicago area (the Cook county) is divided into almost 4000 blocks by the Census bureau. We use these blocks as units to characterize each area. The Safegraph (public) data give us two types of information:

- The GPS coordinates of each block. It is a polygon with coordinates for each of the edges.
- The visits pattern among these, for each hour and each day. I do not have much information on the reliability of these information. More precisely I cast doubt on the representativeness of the users tracked by this service. Nonetheless I found this kind of data, put publicly on Kaggle, interesting and decided to use it.

Census data

From the Census data to get information about income, age, occupation, population size, and some geographic information, for each census block group (CBG). The Census data is nine years old (the next one is in the coming year) but has a lot of local information. Some CBG are discarded since they are empty and/or in water area (to reach a total number of 3991 CBG).

Foursquare API

I retrieve all restaurants in the Cook county from the Foursquare API. I got around 11'700 restaurants, with their locations and specialties. According to Tripadvisor, there are 8190 restaurants in Chicago thus I am pretty confident in the exhaustivity of my list but I know some restaurants of my list to be closed for years. This was not the only issue with Foursquare. The API does not return all entries of the database matching the query. Despite these issues, the Foursquare API was mandatory for this essay. Two methods have been uses to retrieve an exhaustive restaurants list: 'search' and 'explore'.

"Search for venues" approach

This method is supposed to return the list of venues within a given area ("browse"). First, a box encompassing each CBG has been defined using the package Shapely. Then I called the list of venues in the category "food" (4d4b7105d754a06374d81259) in each of these boxes. If the query got 50 entries (the upper limit of the API), I divided the box into 4 and run a query for each of these. Then I merged all the results and kept only the unique entries, defined by the venue's id.

As a robustness check, I also tried the category of Japanese restaurant (4bf58dd8d48988d111941735) using the same code. I got the following surprise: for some boxes, more *Japanese restaurants* were returned than *restaurants*! Thus, I combined a second approach to extend my list of restaurants.

"Venue Recommendations" approach

This method returns a list of venues near a given location. Using the center of each CBG, I retrieve the list of venues in the section "food" within a radius of 500 meters. Again, if the query got more than 50 entries (the upper limit of the API), I run again the query but with a smaller radius.

The first approach returns 25'021 food related entries (e.g. restaurants and bakeries). The second around 12000 venues. Combined, I got 25'602 unique entries. Given the improvement from the second approach, I used the venue recommendations approach once more and reach a total number of 26'163 food related entries. Then I kept only the entries corresponding to a restaurant's category and within the Cook county area. It leads to 11'712 unique entries. To assign each restaurant to a CBG, I use Shapely to check if the restaurant's location is within a CBG's Polygon.

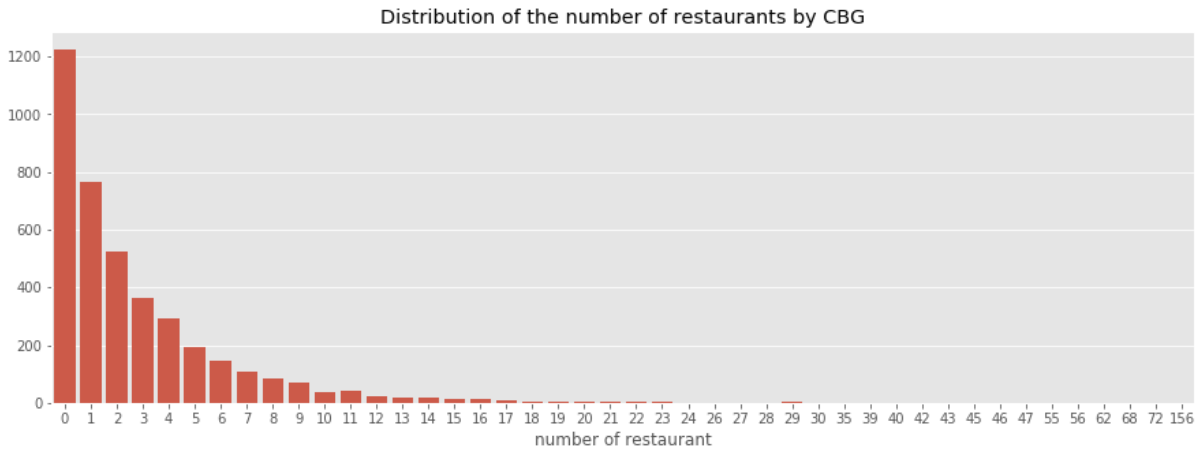
This part as been run on IBM Watson Studio for two reasons. This part is time consuming because of the Foursquare API limitation of 5000 calls per hours. And I could save the results on the IBM COS service.

Data cleaning

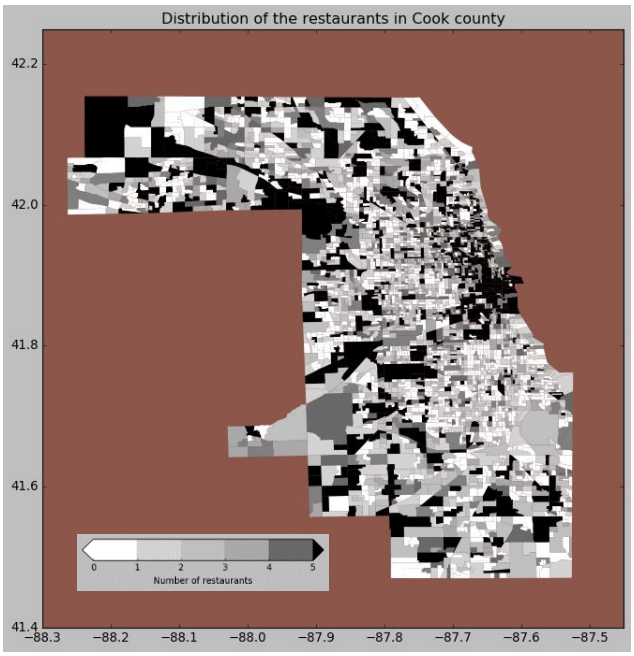
The data for a few CBGs were missing. I first compute the neighbor CBGs of each one (using Shapely.touches). Having this dictionary at hand, the missing values were replaced by the mean of the neighbors' value.

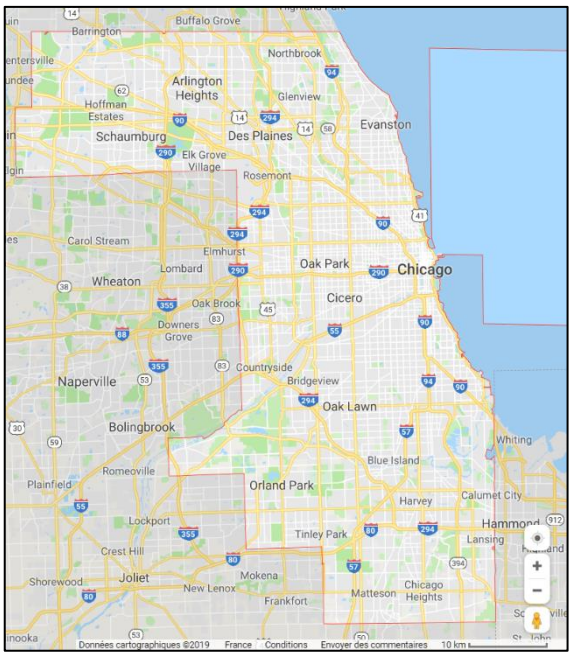
I also encountered one CBG within another, and I simply merged these two. The computational speed dominated the cost of dealing with multipolygons just for one block.

Description of the variables



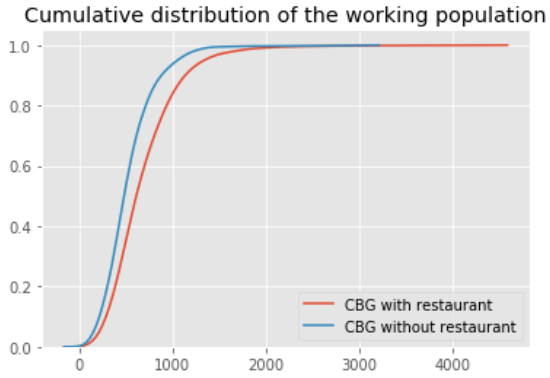
There are a lot of CBG without any restaurant. Two aspects must be taken into account: some blocks have no restaurants because of their characteristics and other because nobody has opened one in these yet. This aspect will be further discussed in the Analysis section. The map below emphasizes the irregular distribution of the restaurant within the Cook county. The white areas have no restaurant while the black have more than 5.

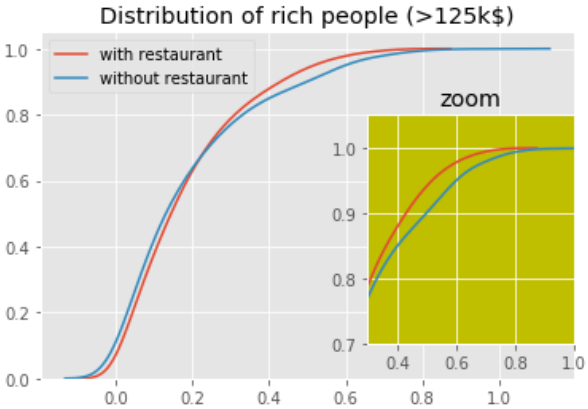




Distribution analysis

On the right, I plot the distribution of the working population, distinguishing between the blocks with and without restaurants. One can immediately observe that the restaurants are in areas with more active people. Here, the two curves do not cross, we can speak of [first order dominance]. Please note that an inactive person can be unemployed, too young, or too old to work.

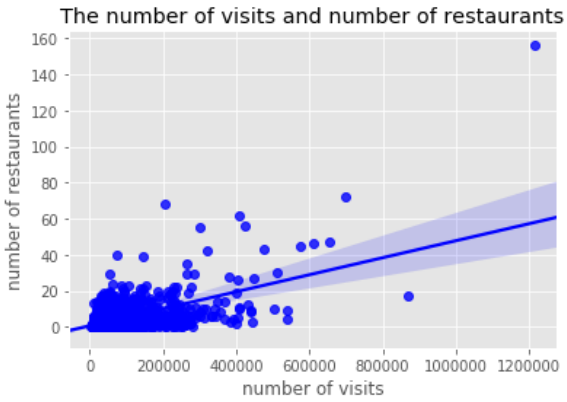




Also, if we look at the distribution of rich people (defined as the number of people earning more than 75% of the Cook county population, i.e. 125’000$), we see that richer areas tend to have no restaurants. This can be explained by the real estate price market or homeowner associations rules.

The number of visits

The use of the Safegraph data confirms the positive correlation between the number of visit and the number of restaurants. Of course, one cannot infer any causation here as each variable depends on the other.



Factor analysis – visits pattern

The number of visits is highly correlated across hours and days. In order to get uncorrelated features from this information, I did a factor analysis. The second advantage of this technique is data reduction.

The technique used is **principal factor analysis**. Some may ask why *not principal component analysis*. "In principal components analysis we assume that all variability in an item should be used in the analysis, while in principal factors analysis we only use the variability in an item that it has in common with the other items. […] In most cases, these two methods usually yield very similar results. However, principal components analysis is often preferred as a method for data reduction, while principal factors analysis is often preferred when the goal of the analysis is to detect structure."[1] The latter is precisely our purpose. I used Stata (in Python, using IPyStata) for this task, for personal reasons.

I reduced the 24+7 variables to 5 factors. The factor 1 capture most of the common features. Factor 4 captures the WE pattern. Factor 5 captures more the off-work visits patterns.

Uniqueness is the percentage of variance for the variable that is not explained by the common factors. The 5 factors reduce the uniqueness of each variable below 0.02, which in line with the underlying PFA hypothesis that the uniqueness is 0.

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Uniqueness
_0h	0.9334	0.3451	0.0495	0.0167	0.0642	0.0028
_1h	0.8953	0.4355	0.0015	0.0432	0.0738	0.0015
_2h	0.8788	0.4652	-0.0344	0.0569	0.0606	0.0032
_3h	0.8891	0.4515	-0.0135	0.0391	0.0074	0.0039
_4h	0.9139	0.3825	0.0517	0.0059	-0.1004	0.0057
_5h	0.9584	0.1633	0.1279	-0.0020	-0.1800	0.0061
_6h	0.9756	-0.0047	0.1152	0.0326	-0.1735	0.0037
_7h	0.9748	-0.1044	0.0142	0.1078	-0.0912	0.0188
_8h	0.9743	-0.0980	-0.1507	0.0939	-0.0134	0.0094
_9h	0.9741	-0.0655	-0.2034	0.0342	-0.0039	0.0043
_10h	0.9730	-0.0590	-0.2128	0.0043	-0.0012	0.0045
_11h	0.9697	-0.0644	-0.2274	-0.0158	0.0089	0.0034
_12h	0.9680	-0.0736	-0.2315	-0.0314	0.0126	0.0028
_13h	0.9730	-0.0787	-0.2068	-0.0335	0.0042	0.0033
_14h	0.9786	-0.0942	-0.1703	-0.0324	-0.0112	0.0033
_15h	0.9804	-0.1167	-0.1418	-0.0206	-0.0067	0.0047
_16h	0.9812	-0.1346	-0.1158	-0.0006	0.0084	0.0057
_17h	0.9811	-0.1476	-0.0590	-0.0102	0.0155	0.0118
_18h	0.9825	-0.1216	0.0259	-0.1058	-0.0088	0.0079
_19h	0.9850	-0.0365	0.0506	-0.1493	-0.0124	0.0035
_20h	0.9849	0.0244	0.0520	-0.1500	-0.0129	0.0040
_21h	0.9857	0.0797	0.0726	-0.1116	0.0099	0.0043
_22h	0.9794	0.1470	0.0811	-0.0470	0.0488	0.0080
_23h	0.9639	0.2302	0.0820	-0.0046	0.0624	0.0073
Monday	0.9678	-0.1967	0.1201	0.0715	0.0266	0.0044
Tuesday	0.9655	-0.2085	0.1206	0.0808	0.0351	0.0021
Wednesday	0.9735	-0.1863	0.0820	0.0767	0.0301	0.0041
Thursday	0.9746	-0.1894	0.0671	0.0764	0.0332	0.0028
Friday	0.9668	-0.2013	0.1401	0.0470	0.0327	0.0019
Saturday	0.9297	-0.2127	0.2716	-0.0319	0.0668	0.0112
Sunday	0.9382	-0.1862	0.2679	-0.0219	0.0263	0.0123

[1] Source: http://www.statsoft.com/Textbook/Principal-Components-Factor-Analysis

Predictive modeling

I create a dummy variable to define the CBG with any restaurant. I applied 3 different techniques to forecast the presence of restaurants: support vector machines (SVM), random forest, and logistic.

The 3991 locations are split in two: a test set (40%) and a training set (60%). Because of the imbalance between the locations, I use StratifiedShuffleSplit, which returns stratified splits, i.e. which creates splits by preserving the same percentage for each target class as in the complete set. The features are standardized using StandardScaler. It removes the mean and scales to unit variance. The parameters used for each technique are in the code.

The models evaluation

		Predicted	
		No restaurant	Some restaurant
Observed	No restaurant	True negative	False positive
	Some restaurant	False negative	True positive

The evaluation of these models will be done through the recall and the precision score. They are defined as follow:

$$recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

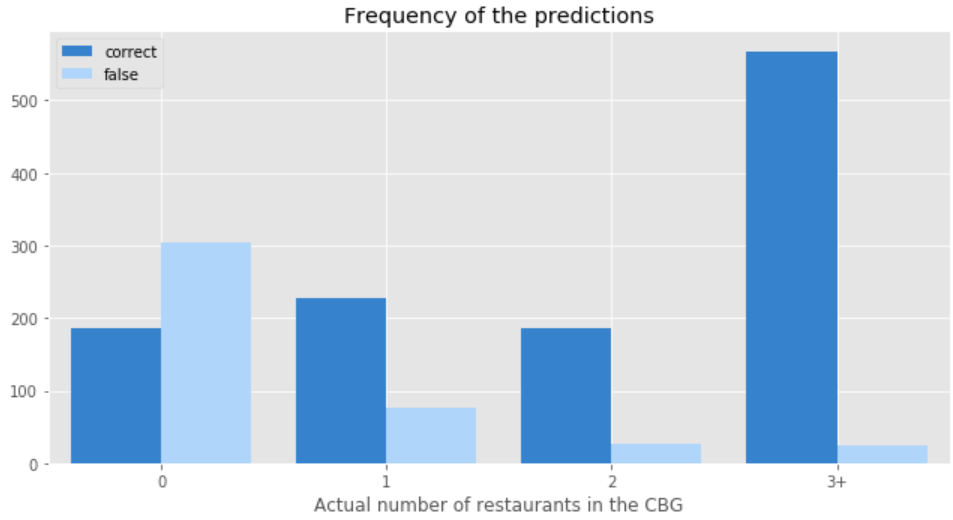
$$precision = \frac{TP}{TP + FP}$$

The false negative error results in an opportunity cost: the model invites not to open a restaurant in this location. Not opening a restaurant where it would be profitable is a potential loss. On the other hand, a false positive can lead to a concrete loss since we would open a restaurant in a non-attractive location. Nonetheless, **a false positive can also be a true opportunity reveals by the model**: we observe no restaurant in this location but it could be profitable to open one there. Thus, I will look at the recall score for the location with more than 3 restaurants. A high recall score for this subset shows the ability of the model to recognize the attractive location. A model with a high recall score tends to predict more positive outcomes. Adopting a conservative approach, I will use a model with a high recall score to predict the location where one should NOT open a restaurant.

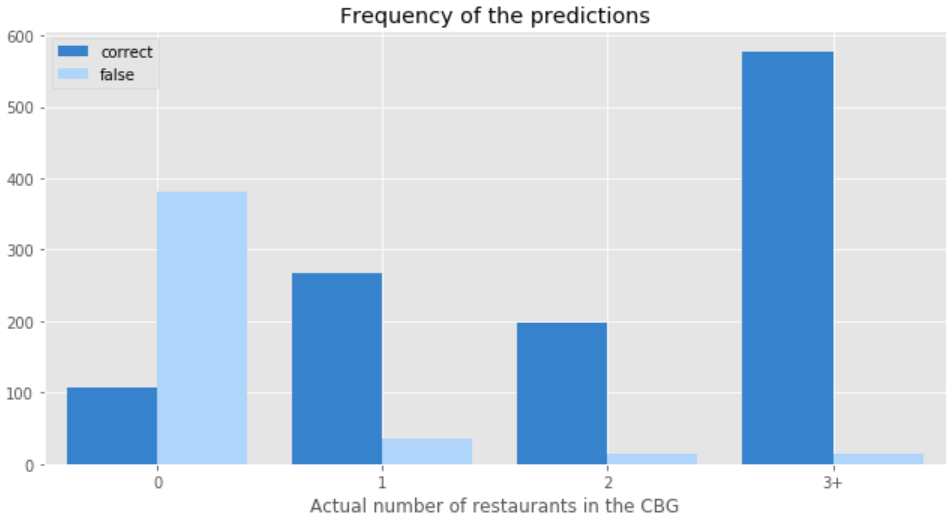
It is worth to emphasize that **the model does not have the actual number of restaurants**, but only if there are any. I was surprise to see that the recall score was so high for the location with many restaurants (more than 3). Once I get the predicted dummy variable (none or any restaurant), I look at the actual number of restaurants in this location.

Support vector machines

This model tends to predict more positive outcome. The following graph shows the number of correct and incorrect prediction for each actual number of restaurants in the test set. Its overall recall score is around 93% and increases to more than 97% for the CBG with more than 3 restaurants.

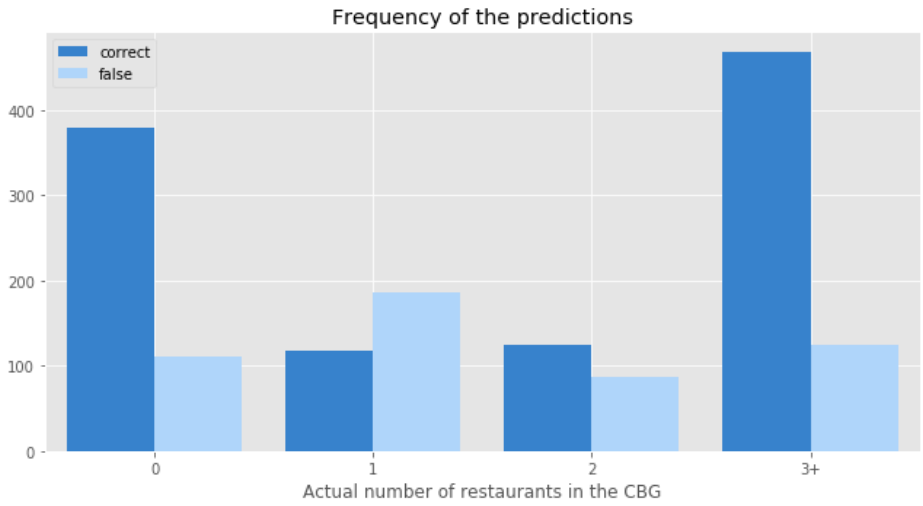


Random forest



The random forest is more conservative (predicts more negative outcomes) but kept a nice recall score for the CBG with more than 3 restaurants (95%).

Logistic model



This model is the more conservative. It predicts a lot of negative outcomes, even for the popular CBG. The recall score for the CBG with more than 3 restaurants is of 79%. On the other hand, its precision score is much higher than the other two.

Conclusion

The following score are for the same random state. The results are pretty similar across random trials.

Score	SVM	Random Forest	Logistic
Precision	.731	.762	.865
Recall	.941	.885	.641
Recall when $N \geq 3$	.975	.957	.791

As said in the introduction, a higher precision score will avoid a loss but may also discourage investment in a profitable location. On the other hand, a high recall score can highlight a too optimistic result. My recommendation would be:

- To use the SVM model to see if the location is 'bad' (negative predicted outcome)
- To use the Logistic model to see if the location is 'good' (positive predicted outcome)

The random forest is more an in-between model for this particular problem and may be used to confront the predictions of the other two.

Two concluding remarks should be made. This approach could be improved by using the visit pattern across the location, with a Markov process. It does not take into account other factors like the ratings and price range of the restaurant. For example, an open restaurant with a bad reputation highlight an attractive location (like a touristic site).