Capstone Project - The Battle of Neighborhoods

1 Business problem

For opening a new restaurant, there are some key factors determine whether the business is finally successful or not. How to get some advanced advice for an investor who is planning to open a new restaurant will be expected strongly. The inventor want to know the success indicator if he/she is planning to open a restaurant in a certain venue and compare the indicator score among different venue, then a better address can be found out for their final success. Hence a predication system will be very helpful to help restaurant investors to avoid the risk of investment because the wrong venue selection.

2 Data Exploration

To implement this system, I will explore Foursquare's geographic location data and pertinent social network data, to train a machine learning model (classification).

First of all, a good restaurant must own plenty of customer, means it can't be too far away from neighborhoods where popularity gathered, so that neighborhood is a good starting point. As the story happened in NYC, I got this location data for free on the web, here is one link to the dataset: https://geo.nyu.edu/catalog/nyu_2451_34572

Tab-1: Neighborhood data for NY city. (sampled first 5 rows)

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

Then moved to the true target data of our study, namely: the "restaurant". Through Foursquare *"search"* API, I got all nearby restaurants for each of the above neighborhoods, to construct a full id list of all restaurants who sit near those 306 New York neighborhoods. Following restful API will

download all the venue IDs whose category is Restaurant or Coffee: https://api.foursquare.com/v2/venues/search?&query=Restaurant,Coffee

Tab2: restaurant scanned through neighborhoods in NY city:

	v_id	v_name	v_dist	v_cat
0	4db03c875da32cf2df4509f4	Big Daddy's Caribbean Taste Restaurant	1008	Caribbean Restaurant
1	4c66e0068e9120a15929d964	Kaieteur Restaurant & Bakery	1011	Caribbean Restaurant
2	4c994113a004a1cdc3393e6e	Bay 241 Restaurant & Lounge	792	Caribbean Restaurant
3	4be5f0eacf200f47d1fa133c	McDonald's	904	Fast Food Restaurant
4	508af256e4b0578944c87392	Cooler Runnings Jamaican Restaurant Inc	479	Caribbean Restaurant

Here the basic restaurant information listed including:

- v_id: the venue id of the restaurant, I use this ID to approach more information around the restaurant
- v_name: the name of the restaurant
- v_dist: the distance from neighborhood to nearby restaurant
- v_cat: the restaurant's major category

On this table, the restaurant ID is the key to catch more detailed restaurant information in next step. The distance from neighborhood to restaurant could also be useful, as I believe, the closer a restaurant to neighborhood, the more convenient for residents to visit the restaurant.

However, the above dataset is not what I finally want, because for each restaurant, there must be multiple nearby neighborhoods, so there must be duplicated lines for a single restaurant, I need to transpose it and merge same restaurants by "*groupby*" operation. Then I got one additional transposed feature: the average distance from a restaurant to all its nearby neighborhoods.

Here is the new data set:

Tab 3. Basic restaurant data:

	v_name	v_cat	avg_dist_2_neighborhood	cnt_near_neighborhood	restaurant_id
0	Cooler Runnings Jamaican Restaurant Inc	Caribbean Restaurant	479.0	1	508af256e4b0578944c87392
1	McDonald's	Fast Food Restaurant	904.0	1	4be5f0eacf200f47d1fa133c

Form above data, you can see two new columns named **"avg_dist_2_neighborhood"**, and **"cnt_near_neighborhood"**. The useless column "v_dist" has been dropped off, and duplicated restaurants are also grouped in a single line in the dataframe.

Then, more features will be added as following:

- Latitude of the restaurant
- Longitude of the restaurant
- Average rating of clustered restaurants where this restaurant is the centroid
- How many recommended popular venues nearby this restaurant
- The label(or the true target of regression) of the restaurant
- The menu items(food) offered from this restaurant

To get these basic restaurant information, such as latitude, longitude and rating, I need to invoke the following Foursquare API “venue”:

https://api.foursquare.com/v2/venues/{ restaurant id }

https://api.foursquare.com/v2/venues/explore/ll{restaurant Latitude, Longitude}

https://api.foursquare.com/v2/venues/{ recommend venue id }

https://api.foursquare.com/v2/venues/{restaurant id}/menu

After all the iterative (or even cascaded) Foursquare APIs invocations, I got the feature list looks like this :

Tab 4: detailed restaurant data set

v_name	v_cat	avg_dist_2_neighborhood	cnt_near_neighborhood	restaurant_id	la
0	Cooler Runnings Jamaican Restaurant Inc	Caribbean Restaurant	479.0	1	508af256e4b05
1	McDonald's	Fast Food Restaurant	904.0	1	4be5f0eacf200
2	241 St Cafe & Restaurant	American Restaurant	1019.0	1	4c010e75cf3aa
3	Ripe Kitchen & Bar	Caribbean Restaurant	798.0	1	4d375ce799fe8
4	Townhouse Restaurant	Restaurant	218.0	1	4be2b79d660ec

Define a dictionary as below:

attribute Name	Data description	Data Type	Potential Contribution
v_name	Restaurant name	String	n/a
v_cat	Category	Foursquare Category	certain Category maybe special popular, more easily catch eyes, will be encoded
avg_dist_2_neighborhood	Average distance from this restaurant to all its nearby neighborhoods	float	the lower means distance closer to potential clients
cnt_near_neighborhood	how many nearby neighborhoods are close to this restaurant, for example with 1 km	int	the more means more potential clients
avg_rate	Average rating of nearby popular sites(such as food, drinks, coffee, shops, arts, outdoors, etc.)	float	the higher, the more possibility for stable client source
nearby_rec	the total number of popular sites recommended from Foursquare which centriod by the restaurant	int	the higher, the more possibility for stable client source
menu	special food offered	list of food items	will do clustering , turn it to group(clustering) label before being used for classification model
rating	**the restaurant rating (how good the restaurant is)**	**float/classification**	**the label or target value(y)**

3 Methodology

3.1 Model Selection

I make the model to give a prediction as simply “bad or good” classification by turning "rating" to binary classification. In that way, I need to label restaurant data set by rule: take the 75% value 7.75 as a threshold, if rating larger than 7.75, then label it as “Good”(1); or else label it as “not good”(0).

v_name	rating	label
Cooler Runnings Jamaican Restaurant Inc	6.5	0
McDonald's	6.5	0
241 St Cafe & Restaurant	6.6	0
Ripe Kitchen & Bar	8.7	1
Townhouse Restaurant	5.6	0

3.2 handling dummy variable

The last step is to working on the dummy variable(menu group and category). For menu feature, I won't consider the influence for a while, so I will drop menu and will use the panda's method 'get_dummies' to assign numerical values to category.

3.3 Classification model and result

The classification models was much more straightforward, with the location and food input features, I need to predict whether these choices , combine together, will bring up one success(1) or failed(0) business. Here I have all the data for model training, I will divide them into training set and validation set:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

Build the model:

LR = LogisticRegression(C=0.1, solver='liblinear',class_weight={1:0.65,0:0.35}).fit(X_train,y_train)

3.4 Evaluation

Now from a comprehensive angle, look at accuracy of classifier through confusion matrix. I created a confusion matrix here:

The classifier correctly predicted 19 of 22 as 0, so, it has done a good job in predicting the higher risk of negative result. Meanwhile 2 of 3 good rating prediction incorrectly to risky, this is bit high, Now

that avoiding risk is our major goal, it is acceptable.