**Coursera - IBM Data Science Capstone Project**

**Brian Lee**

[BrianTrewLee@gmail.com](mailto:BrianTrewLee@gmail.com)  
LinkedIn: <https://www.linkedin.com/in/brian-lee-data>  
Github code:<https://github.com/Brian-Lee/Coursera_Capstone/blob/master/brians_capstone_project.ipynb>

**1. Introduction**

If I were to open a Starbucks, where should I locate it? Clearly, the location is one of the most important business decisions for this venture. Is there a science to locating a Starbucks? Can we create a machine learning model to predict where a Starbucks is likely to be located? I'd like to find out.

If a reliable model can be made, it could be used in the process of opening a store. It could be used as a final sanity check, or at the beginning stage, to select a promising location. If a trustworthy model predicts with a high degree of confidence that a Starbucks should be located in an area, but there is not one there yet, perhaps that is an opportunity.

Any person interested in opening a Starbucks should be interested in these results. This includes myself, other potential franchisees, and the Starbucks corporation itself, which operates many of its own stores. The Starbucks competition might also be interested as they could possibly gain competitive insights. I also believe others might be interested in this procedure, as it might be applied to predicting the location of other entities, based on the same sort of data.

The main purpose of this project is to prove the concept of predicting Starbuck's locations in general. I suspect it may work better or worse depending on the locations chosen for training and prediction, the radius size, and the specific features used in the model. I may vary those factors in order to prove the concept, which could then be applied carefully to a particular geographic area of interest at a later date.

**2. Data**

Foursquare ([https://foursquare.com](https://foursquare.com/)) is a company that crowdsources location data, tying latitude-longitude coordinates (and other things) with public venues, including many businesses, such as Starbucks locations. After generating a corpus of geographic coordinates, the features for the Starbucks location classifier will come from JSON venue data returned by the Fourquare API ‘explore’ endpoint. I will use venue category names (such as 'bar', 'Chinese restaurant', 'coffee shop') as features to classify an area as either an area with a Starbucks, or an area without one. I will use the venue name (e.g. 'Starbucks') to determine whether or not the venue is a Starbucks. An area containing a Starbucks contains at least one venue named 'Starbucks' (case insensitive) within the radius supplied to the Foursquare API's explore' endpoint.

Once I have labelled data, I will use sklearn.model\_selection.train\_test\_split to generate training and testing sets. I will also generate a separate validation set. Then I will use Python scikit-learn classifiers to generate predictive models.

An area will be a circular area with a given radius. I used a radius of 300m for this project. Within that radius, the Foursquare API will return venues. The larger the radius, the more venues that could be anticipated, and the higher likelihood of a Starbucks, but a larger radius is also less useful for business use.

The quality of the Foursquare venue data is good, but not perfect. I noticed venues returned as "SUBWAY" separately from other venues returned as "Subway." I cleaned that by altering the case of the venue names after received by the API. I also noticed that the crowdsourced latitude-longitude coordinates for Starbucks locations often include duplicative results, slightly offset. For instance, there could be a cluster of tightly packed coordinates returned as Starbucks locations, giving the impression of many Starbucks next to each other. I suspect this actually reflects different geographic coordinates representing a single Starbucks venue. Additionally, there are areas (for instance in my validation area in Illinois) where no venues are reported within the radius. In some cases, it can be seen that some venues (such as churches) are in fact located nearby.

The areas chosen for training/testing and validation data contain a mix of areas with Starbuck's and those without. I selected a geographic rectangle in San Francisco for training/testing. With a reasonably small radius, most points will have true labels of 0 (or not near a Starbucks.) In order to balance the classes for better binary classification performance, I added a large number of additional geographic points to the training set. These additional coordinates come from the Kaggle dataset (<https://www.kaggle.com/starbucks/store-locations>) containing the latitude and longitude coordinates of 25,600 worldwide Starbucks locations. The idea was that these added points should have true labels of 1 (near a Starbucks), and most do. However, I discovered that they do not all. To that end, the best balancing would be a ratio of about 2:1. The fact that these added points don't all appear to have labels of 1 is distressing. Perhaps that is a sign that something is wrong, or perhaps it is due to different data sources not agreeing on exactly where a current Starbucks venue is located. In any event, my training data is comprised of a grid of points inside a rectangle, combined with an equal number of points throughout California.

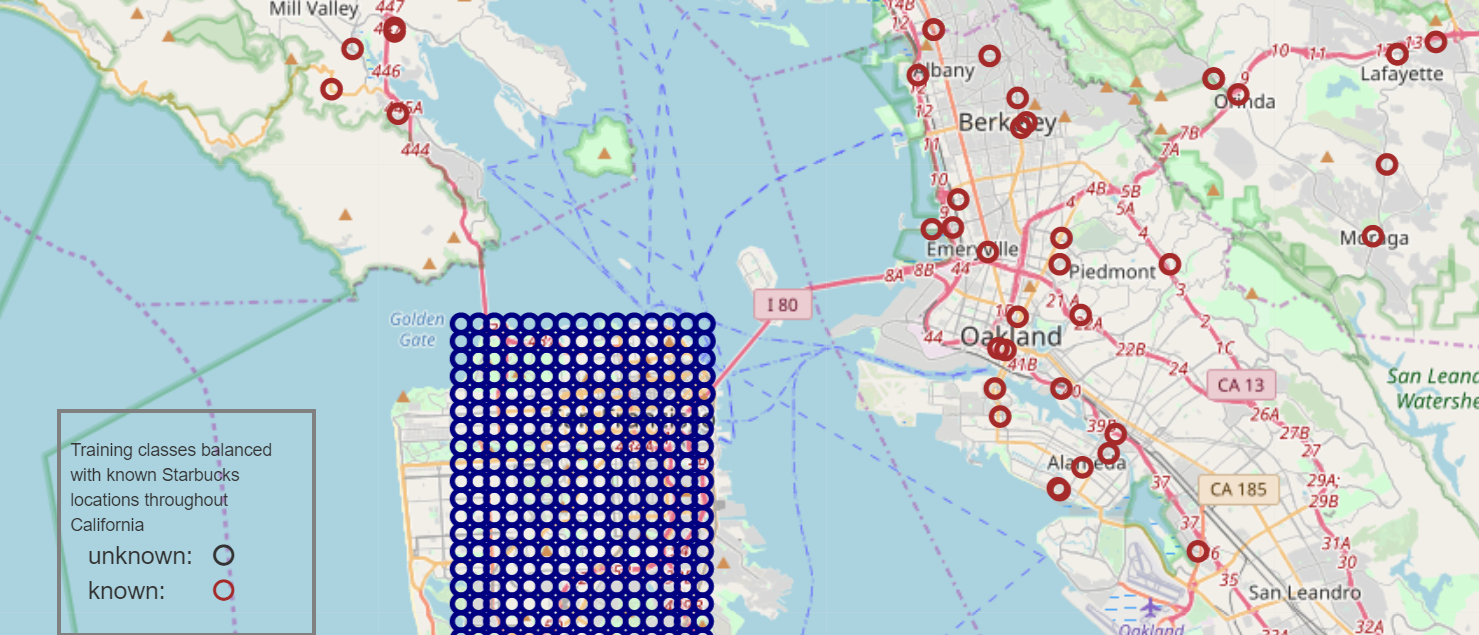
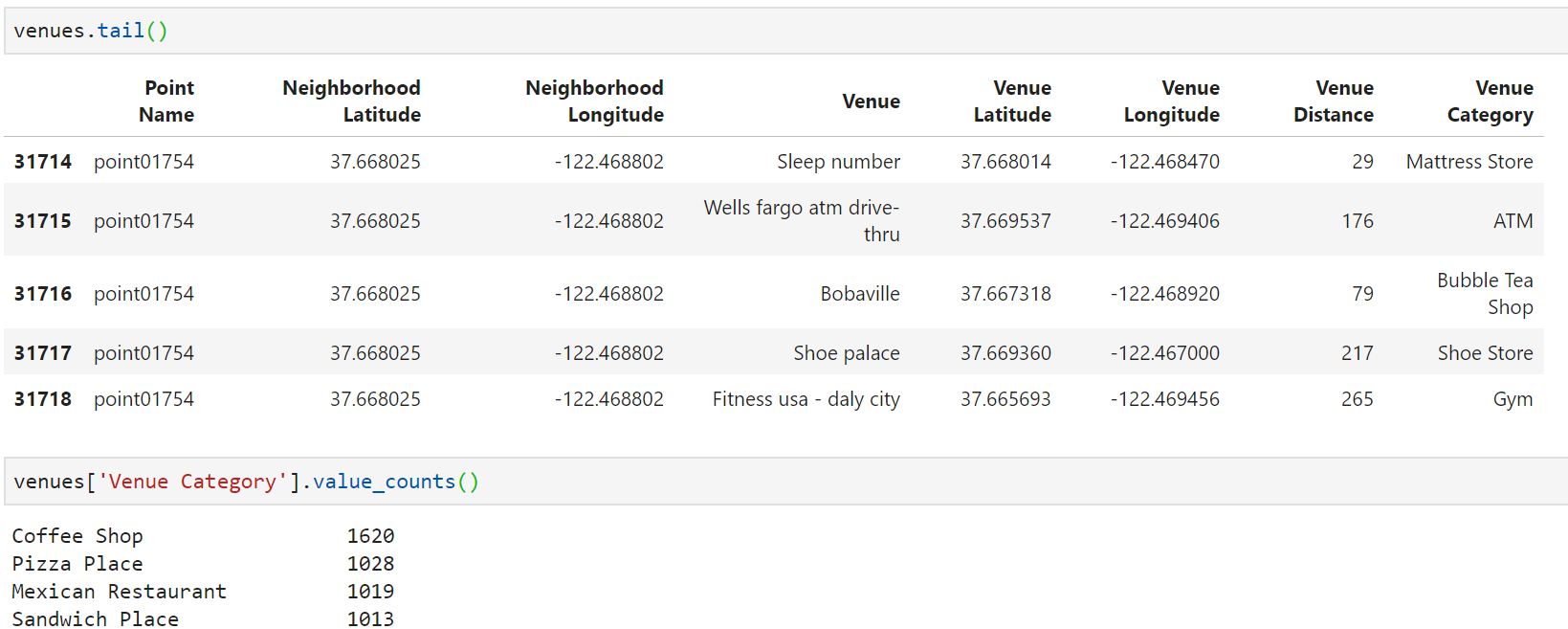
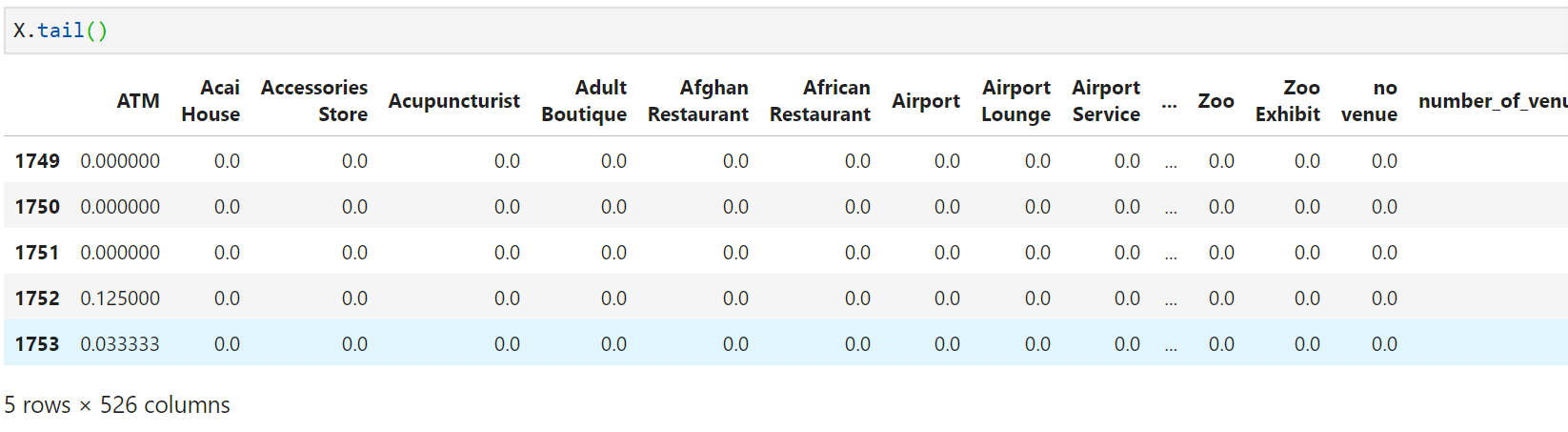
****

Figure 1: Zoomed map showing some of the training/testing data points, with navy circles representing grid coverage coordinates, and brown circles representing coordinates intended to balance the resulting classes.

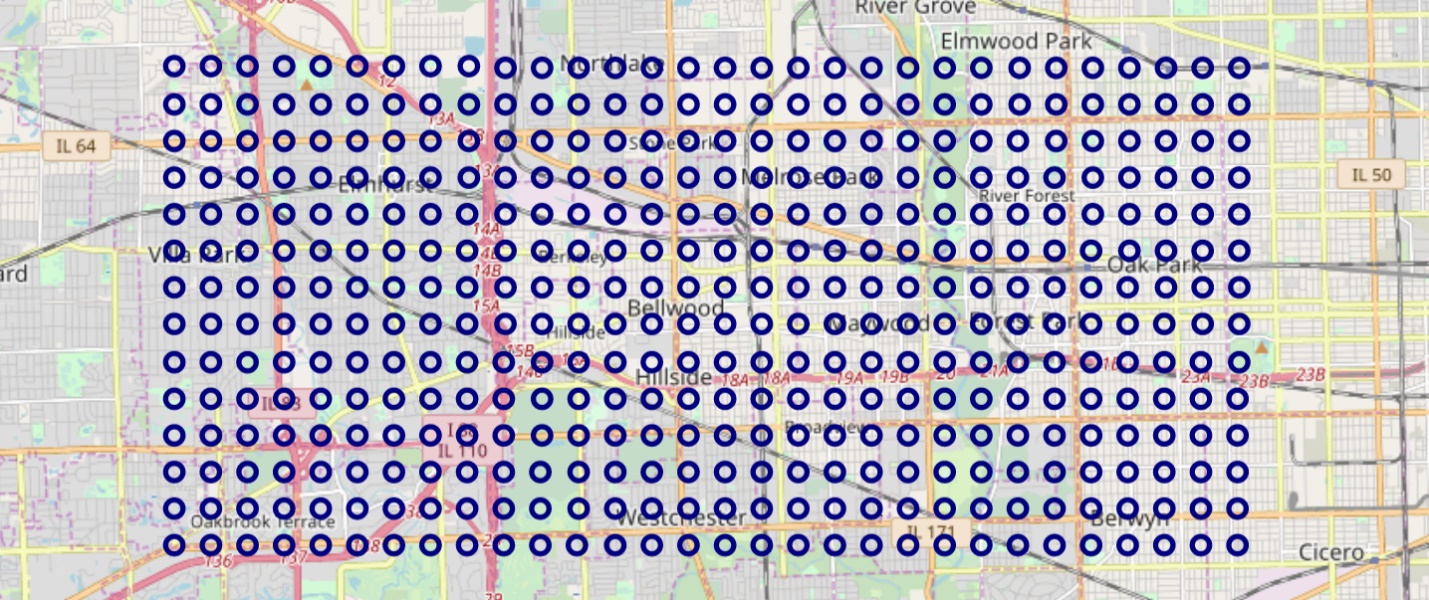
The foursquare API returns the names of venues as well as the names of the categories of the venues. Here is an example of the resulting pandas DataFrame:



In order to keep my feature-space down, I used the category names, but not the venue names. The category names were one-hot encoded and normalized. Multiple rows for a given coordinate set were combined, resulting in a feature DataFrame such as the following:



I wanted to validate the results on a completely separate geographic area, with a naturally occurring distribution of classes. I chose a rectangular area in Illinois, near Chicago, with a decent number of Starbucks locations according to Google Maps.



3. Methodology

I ran some random forests and support vector machines on the full set of features, initially getting training set accuracies as high as 1, and test-set accuracies as high as 96% or so. Ultimately, I decided that I should remove the venue entries that represent actual Starbucks locations, even though they were deidentified, with only the venue category name (“coffee shop”) in the training data. When I removed those data rows, my performance dropped a fair bit, but I believe these results are more transferrable to a real-world situation.

When I got the venue category names I used to create onehot features for the validation set, I discovered that I had a slightly different set of features. There were a few (3) venue category types that did not occur in my training/testing set, and a few dozen (~30) that occurred in the validation data that did not occur in the training/testing data. I could have retrained my model with a superset of features that would satisfy both regimes, but I chose instead to reformat my validation features to fit my trained model. I added three dummy columns to the validation data, and threw out 30 columns, resulting in a feature set of 452. I felt this approach was a good approach to test the generalizability and durability of the trained model. Ultimately, I increased the size of the training area, resulting in over 500 features. I also added a few engineered features, which tend to rise to the top of the feature importances. Some of the important engineered features are sums of other features, for instance, “rssj”, which increments for each feature with “restaurant”, “store”, “shop”, or “joint” (e.g. “Burger Joint”) in its name. This is an attempt to aggregate a host of similar type venues into a single feature.

I checked for strong correlations (> 0.7) between features, as that can degrade training of some models. I removed those features first and re-ran my models. I then used recursive feature elimination to try to ascertain the best number of features to use.

I tried different hyperparameters for Random Forests (max-depth from 30-20, number of estimators from 1-100). Ultimately, I used 100 estimators (probably overkill) and a max-depth of 6. Max-depths over 15 showed signs of overfitting, as the training scores were much higher than the testing and validation scores. I attempted to gain insight into the classifier by visualizing a model with just one estimator and a depth of only 4. I ran Random Forests with a wide range of features, from over 500 to just 1.

I tried several kernels for SVM: Radial Basis Function, Linear, Polynomial, and Sigmoid. I ran the Support Vector Machines with 500 features, and one extra promising SVM with a Radial Basis Function with just 50 features.

For feature reduction, I first used the approach of removing highly correlated (>0.7) features. After that, I gradually and recursively removed the features with the lowest feature importances. When I thought I had an insight, I created a few features to exploit that, such as a feature that just represents the total number of venues in an area.

In the hopes of gaining insights, I plotted my data on maps, and superimposed Foursquare results for “Starbucks” searches. My data includes rectangular areas of points which can be classified as either “0: not near Starbucks” or “1: near Starbucks” Also, irregularly spaced points drawn from the Kaggle starbucks/store-locations.csv file, which we could guess will have labels of 1, but in practice only some do. Those points can be classified with either label. Finally, I added locations of Starbucks returned from a Foursquare search. These points are not classified, but are shown in red to inspire understanding. We would like to see our green circles (class 1 points) clustered closely with red circles. Away from red circle markers, we’d like to see mostly blue circle markers. We can zoom in or out on the Folium map in order to try to identify venues, venue clusters, geographic features etc. that we would expect to influence the likelihood of a Starbucks location in the area.

I ran the Random Forests with a wide range of features, from all the features (over 500) to just 1. I ran the Support Vector

4. Results

I got the best performance from a Support Vector Machine trained on 500 features. The validation accuracy is suspiciously high at 0.95:

-----SVM (with rbf kernel)-----

SVM (with rbf kernel) Train accuracy = 0.8132573057733429

SVM (with rbf kernel) Train ROC AUC = 0.8980169971671388

SVM (with rbf kernel) Test accuracy = 0.8091168091168092

SVM (with rbf kernel) Test ROC AUC = 0.8797108512020793

SVM (with rbf kernel) Validation accuracy = 0.95

SVM (with rbf kernel) Validation ROC AUC = 0.8595297768230099

SVM (with rbf kernel) Average of Test and Validation accuracy: 0.8795584045584046

SVM (with rbf kernel) Average of Test and Validation ROC AUC score: 0.8696203140125446

I also got good performance from an SVM using a polynomial kernel:

Support Vector Machine - SVM (with poly kernel) created

-----SVM (with poly kernel)-----

SVM (with poly kernel) Train accuracy = 0.8980755523877405

SVM (with poly kernel) Train ROC AUC = 0.9534549119862137

SVM (with poly kernel) Test accuracy = 0.8005698005698005

SVM (with poly kernel) Test ROC AUC = 0.8557667316439246

SVM (with poly kernel) Validation accuracy = 0.9380952380952381

SVM (with poly kernel) Validation ROC AUC = 0.7728845924334646

SVM (with poly kernel) Average of Test and Validation accuracy: 0.8693325193325193

SVM (with poly kernel) Average of Test and Validation ROC AUC score: 0.8143256620386946

The Random Forest with all 526 features did well:

-----RF\_all\_features\_1-----

RF\_all\_features\_1 Train accuracy = 0.8503207412687099

RF\_all\_features\_1 Train ROC AUC = 0.9405871379160383

RF\_all\_features\_1 Test accuracy = 0.811965811965812

RF\_all\_features\_1 Test ROC AUC = 0.8927550357374918

RF\_all\_features\_1 Validation accuracy = 0.8725981620718463

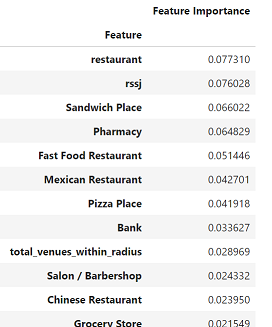
RF\_all\_features\_1 Validation ROC AUC = 0.8725981620718463

RF\_all\_features\_1 Average of Test and Validation accuracy: 0.8422819870188292

RF\_all\_features\_1 Average of Test and Validation ROC AUC score: 0.882676598904669

RF\_all\_features\_1 Num Features: 526

Engineered features “restaurant” (combining such features as “American Restaurant” and “Chinese Restaurant”, “rssj” (restaurants, shops, stores and “joints”) feature more prominently than “total\_venues\_within\_radius”.



Random Forests do well with very few features:

-----Random Forest #13-----

Random Forest #13 Train accuracy = 0.7947255880256593

Random Forest #13 Train ROC AUC = 0.8866215386866416

Random Forest #13 Test accuracy = 0.7492877492877493

Random Forest #13 Test ROC AUC = 0.8414879792072775

Random Forest #13 Validation accuracy = 0.8768946174961213

Random Forest #13 Validation ROC AUC = 0.8768946174961213

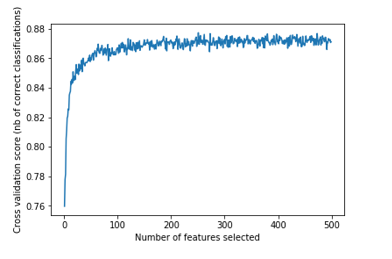
Random Forest #13 Average of Test and Validation accuracy: 0.8130911833919353

Random Forest #13 Average of Test and Validation ROC AUC score: 0.8591912983516994

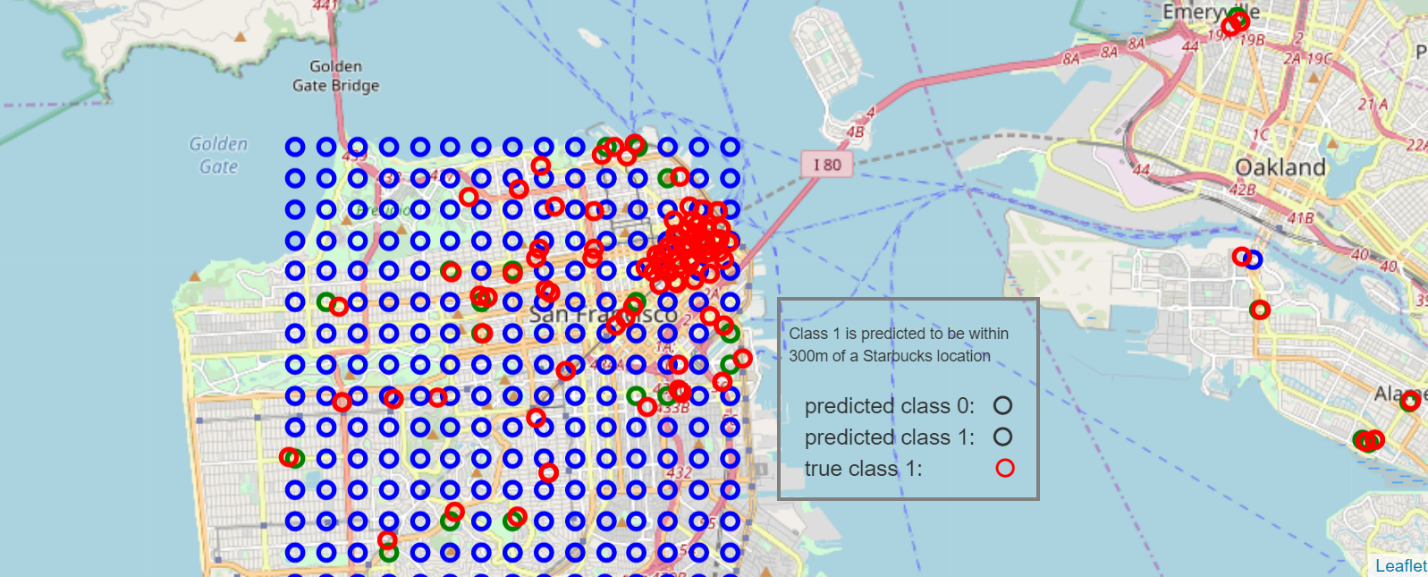
Random Forest #13 Num Features: 3



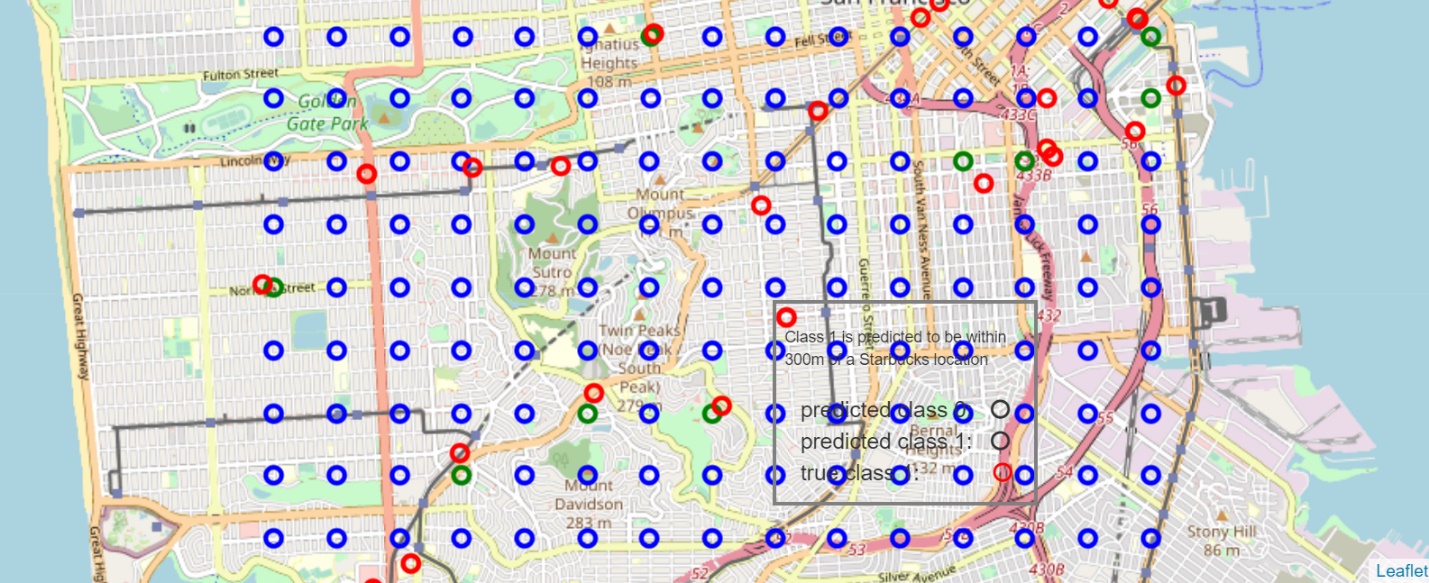
The Random Forests do marginally better with more features:



Here is a zoomed map of the training/testing geographic area showing class 0 pedictions (blue circles), class 1 predictions (green circles) and coordinates returned by the Foursquare API search endpoint for the query “Starbucks” (red circles):



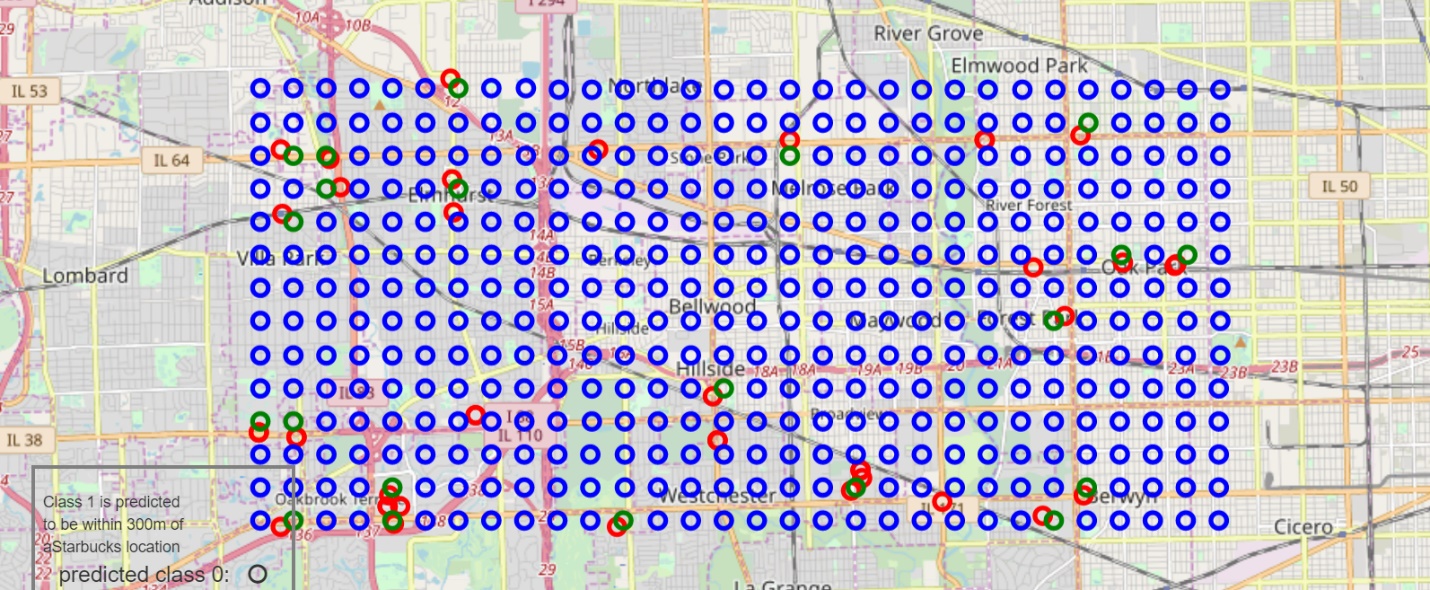
Here is a closer zoom-in:



Here is a single correctly predicted Starbucks location at the corner of Spruce and California Street zoomed in the max amount:



Here is a zoomed map of the validation geographic area showing class 0 pedictions (blue circles), class 1 predictions (green circles) and coordinates returned by the Foursquare API search endpoint for the query “Starbucks” (red circles):



5. Discussion

Removing highly correlated features did not seem to help the Random Forest models very much. In fact, removing even on the order of 500 features only helped marginally. I would generally prefer the far simpler models, even at the expense of a bit of accuracy.

I would like to analyze the performance of polynomial and RBF SVMs with more variations, especially the addition and subtraction of features. I would like to apply Principle Component Analysis (PCA) for feature reduction. I would also like to train on something like 20-100 times as much training data, and see how well the models work on a wide variety of new geographic areas. In particular, it would be nice to see how well these models predict qualitatively different regions, for example, rural areas.

It might be worth altering the radius of 300m as well.

6. Conclusion

The dominant features appear to be closely related to general business density, however, there seems to be an emphasis on food-type venues. It is worth noting , though, that venue categories such as restaurants, fast food, and sandwich places (Subway is the second most common venue after Starbucks) make up a disproportionate quantity of business venues in general.

At a certain point, I was worried that this entire exercise would boil down to the simplistic proposition, “Starbucks are located where there are a lot of businesses.” I was pleased to discover that my engineered feature “total\_venues\_within\_radius” had a lower importance than 8 (or so depending on the exact model) other features, including “Bank” and “Pharmacy.” Also, “Chinese Restaurant” and “Clothing Store” have higher incidences than “Bank” or “Pharmacy”, and yet have little predictive power for Starbucks locations. Indeed, even the venue category “Coffee Shop” has a higher incidence than “Bank” or “Pharmacy”, but much lower feature importance. It should be noted that this category only includes venues other than Starbucks. When I ran the models with the Starbucks venue rows included but deidentified as Starbucks, the feature “Coffee Shop” became the top feature, and the accuracy went up to about 96%.