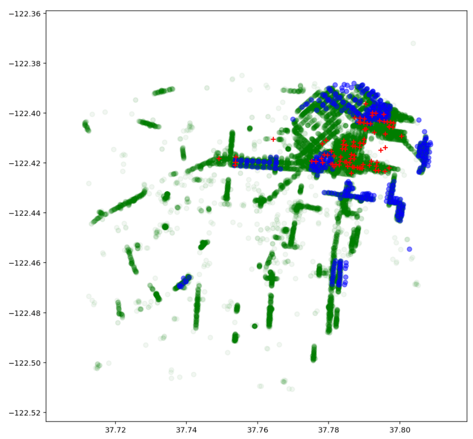
Parking in SF : Kaggle Challenge

Team Name: Chengcheng & Tim Lee
Members: Tim Lee (Kaggle: TimLee), Chengcheng Xu (Kaggle ids: chengchengx)



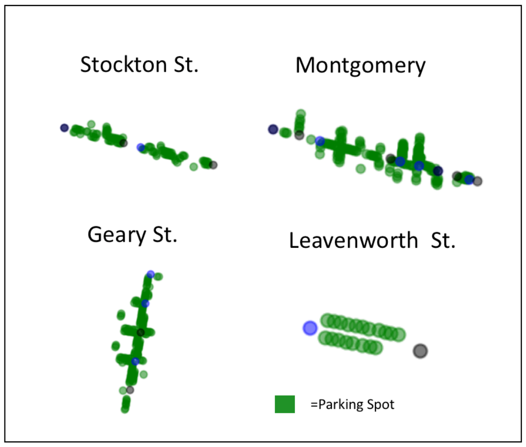


Figure: (left) Overlap of SF data coverage, (right) sample parking spots on select streets

Abstract

With increasing interest in autonomous driving technology, large amounts of money and research are being focused on technology to understand navigating roads, streets and neighborhoods. One of the related areas of research is a statistical approach for predicting parking spaces. While autonomous cars will theoretically be in constant motion, there is a growing need to understand vehicle migration between neighborhoods, and available parking spots for the future robot cars. While driving is a real-time recognition problem, parking has its own share of difficulties, with longer forecasting windows, road work conditions, such as special laws about curbs..

Using manually collected surveys, parking meter records, and magnetic sensor parking data, we attempted to build a parking spot predictive model. We had two goals: first to build a machine learning model to predict parking spaces by street for a particular day and hour, and second to understand the parking behavior of different streets. After using the client-supplied datasets and some additional external geolocation datasets, we successfully applied gradient boosted models to the data to achieve a precision score of 0.52, and a recall score of 0.63, with an overall F0.5 score of 0.61.

Section A: Description of Dataset:

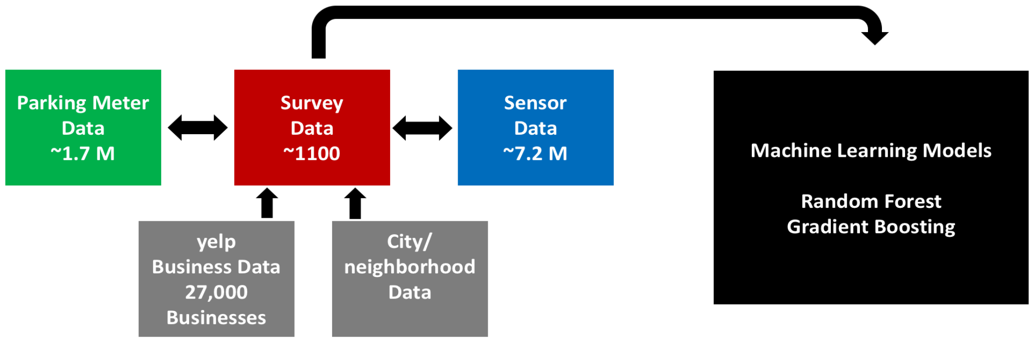


Fig1: Datasets used

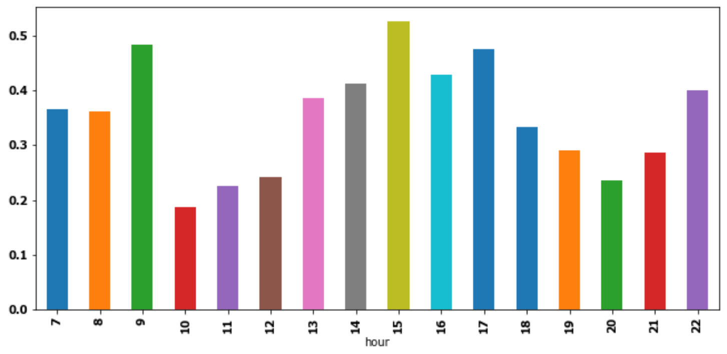
Our client partner provided 3 datasets covering downtown metro San Francisco (SF). The first was 1,100 street survey observations which was collected manually over select streets during the period of Jan-Mar 2014. The data has the street block information, the date and time information, and the number of parking spots manually counted. The second dataset was 7.8 million parking sensor information collected from magnetic pucks embedded under the street at select locations. The embedded sensor data has street name, block number, and records amount of occupied time vs. vacant time. The final third dataset was 1.8 million rows of paid parking meter information, with GPS coordinates and date times. This data represents times when a parking meter was paid. Of the 3 datasets, the smaller 1,100 row survey dataset is considered the training set, and the Kaggle's evaluation set came from a similar survey source.

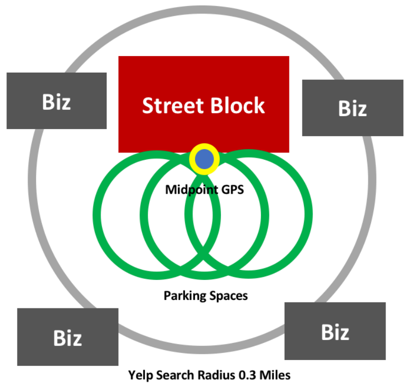
To feed our model insight about the city at large, additional data was sourced from google maps api, San Francisco city stats and the yelp restaurant api. The google maps api was used to assist with GPS to address conversion (and vice versa), and understand overlapping data coverage. In SF, there are 10+ sub-districts or neighborhoods, and city-data.com records were used to understand area, population makeup. Finally, the yelp api was used to pull high-level business information around street blocks in question. Restaurant IDs, categories, review counts, and ratings were all collected within a 0.3 mile radius of the middle of the street blocks.

Section B: Feature Engineering & Approach

The training dataset has approximately 90 unique "blocks" consisting of a street between two intersections. Unfortunately the survey dataset had a number of inconsistencies in the intersection information. A row might be encoded as "mission street from intersection of mission street to B street" which creates a location problem because most streets do not have a self-intersection. Upon further research, these are streets that had small dead-end alleys. Fixing this involved manual labor of looking up the streets on google map, assuming a 1 block length, and picking the closest replacement intersection.

Between the 3 client datasets, the overlapping coverage was very different. The parking data covered more than half of the survey data, but there was much less overlap in the sensor data with the survey data. Figure1 at the introduction shows the coverage of the data provided. The green color is parking data, the blue data is sensor data, and the yellow/red data is survey data. Mapping the GPS parking data onto street blocks was non-trivial, ensuring that spots were actually on a particular block required some mathematical estimations.

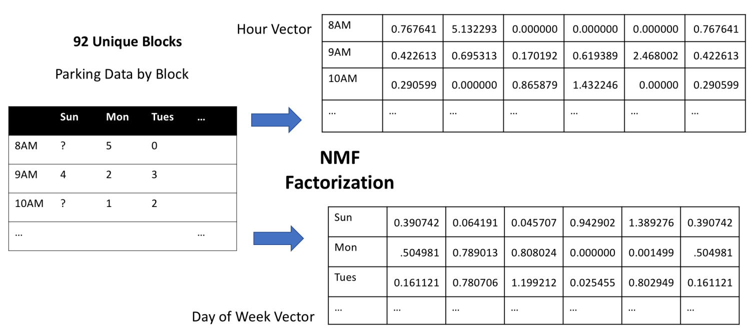




Day of Week and Hour Vectors

Some of the key features were the street, and time of day by hour and day of week (shown in the bar chart above). There are fewer parking spaces at the beginning of business hours, (hour 10) and also around dinner time (hour 18 through hour 21). Similarly, different streets exhibited different parking behavior. A main street like van ness is much different than a smaller street such as residential redwood. How can a machine learning model understand blocks on a smaller scale? Unfortunately due to the small size of the hand-labeled survey data, there are many block-hour-day combinations that are unaccounted for. As a result, we turned to the parking data. The parking data was gathered by street block and analyzed by hour and day of week. Then using a non-negative matrix factorization technique we created two vector sets. One for hour of the day, and one for day of the week, we chose the dimension around half the field's cardinality. This was applied for every unique 91 blocks.

dow	0	1	2	3	4	5	6
hour							
8	3.0	1.0	1.0	1.0	3.0	2.0	NaN
9	8.0	9.0	6.0	5.0	4.0	3.0	NaN
10	5.0	3.0	3.0	2.0	5.0	7.0	NaN
11	4.0	3.0	3.0	3.0	6.0	8.0	NaN
12	8.0	2.0	2.0	2.0	3.0	1.0	NaN
13	4.0	7.0	8.0	2.0	6.0	9.0	NaN
14	3.0	6.0	2.0	1.0	4.0	5.0	NaN
15	6.0	8.0	5.0	12.0	7.0	6.0	2.0
16	5.0	15.0	7.0	8.0	6.0	4.0	NaN
17	6.0	5.0	1.0	4.0	8.0	5.0	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	1.0



	Sun	Mon	Tues	...
8AM	?	5	0	
9AM	4	2	3	
10AM	?	1	2	
...				...

8AM	0.767641	5.132293	0.000000	0.000000	0.000000	0.767641
9AM	0.422613	0.695313	0.170192	0.619389	2.468002	0.422613
10AM	0.290599	0.000000	0.865879	1.432246	0.00000	0.290599
...	...	...	...	...	...	...

Sun	0.390742	0.064191	0.045707	0.942902	1.389276	0.390742
Mon	.504981	0.789013	0.808024	0.000000	0.001499	.504981
Tues	0.161121	0.780706	1.199212	0.025455	0.802949	0.161121
...	...	...	...	...	...	...

Machine Learning Methods

After adding all the additional features to the model, we totaled roughly 300+ features of counts, percentages, and vector floating point. For the ML modeling, we selected tree ensembles such as Random Forest and Gradient boosted tree. The advantage of tree ensembles is they don't require feature normalization, can stably handle collinearity between fields, and still be effective at smaller dataset sizes. For the Random Forest and gradient boosted models, cross validation was used to score, but overfitting remained an issue due to the small data size. As a result, additional parameters were tuned to combat overfitting. The tree depth was limited in level, the features were sampled at every tree, and also at every level.

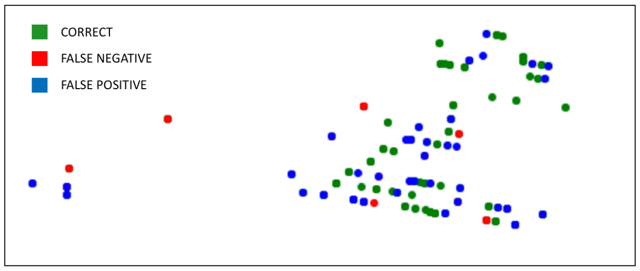
From running the actual models, the random forest approaches often generates unstable classifications, such that precision, recall, and F0.5 will vary dramatically for re-runs of the same model without any modifications to parameters or data. XGBoost (gradient boosted trees) was found to be more stable once small amounts of L1 and L2 regularization were added to the model. The scores would still vary, but much smaller in comparison to the random forest results.

Precision, Recall and F0.5 of 3-fold cross validation:

Model	Precision	Recall	F0.5
RF baseline	0.48	0.30	0.41
XGboost baseline	0.50	0.27	0.39
XGboost_Yelp	0.51	0.35	0.46
XGboost_Yelp + Parking	0.51	0.34	0.45

Non cross validation scores:

Model	Precision	Recall	F0.5
RF baseline	0.09	0.26	0.189
XGboost baseline	0.162	0.301	0.26
XGboost_Yelp	0.59	0.52	0.61
XGboost_Yelp + Parking	0.52	0.63	0.61



XGBoost - Feature Importance (with Yelp + Parking Vectors):

min	0.071115	hourvec8	0.022783	hourvec4	0.017738
day	0.054353	hourvec1	0.022457	hourvec5	0.017413
hour	0.041497	hourvec9	0.02083	Clean_To	0.015948
week	0.035313	hourvec3	0.020667	dowvec10	0.015622
dow	0.034988	dowvec0	0.018226	morn_night	0.015622

XGBoost – Neighborhoods with most errors:

Our top neighborhoods that our model couldn't guess are listed below. Civic center and tenderloin had the most errors, with many of the other errors coming from smaller areas such as Financial District, Fillmore district (historical area). These neighborhoods had erratic traffic data, or sparse data between both the parking and survey datasets.

High Error in Predictions	
tenderloin	13
civiccenter	11
missiondistrict	11
nobhill	10
pacificheights	7
Fillmoredistrict.	6

Conclusions / Next Steps

From working through some of the models, XGBoost has the most promising potential. It remains stable, works well with the huge number of features that we provided it. From the initial studies the most promising features are the time-based ones, with hour, day of week, morning vs. night showing high on the feature importance. While most of the time was spent combining the data and figuring out logical ways inform the machine learning model, there's a number of areas show promise:

- NMF Vector Imputation: Not all the blocks had perfect hour vectors and dow vectors. There were some combinations that were missing, and were imputed global vectors. A better imputation process would theoretically improve the vectors, and improve the score since many of the attributes show high feature importance.
- More specific use of the vast Yelp data: Initially all businesses were pulled, and their categories added with count statistics. Current features have number of 1000+ review restaurants, or number of restaurants with 5 star ratings. Knowing that time-related information is much more powerful, having business operating hours, or classifying food into breakfast, lunch, dinner, or bars could theoretically boost the predictive power.
- More feature transformation of time-based metrics: Additional time-based features such as "hours before next holiday", or change in number of parking spots (is it a high-fluctuation area) could be explored with more time.
- Make specific models for difficult neighborhoods: The current model performs decently. But based on high-error neighborhoods, making a separate classifier and ensembling might help the model focus on these heavy-downtown areas.

E. Responsibilities

Chengcheng - model selection, tuning, feature selection, and exploration
Tim - supplementary data identification, scraping, combining and feature creation

Code Link:

https://github.com/USF-ML2/final-project-chengcheng-tim