Parking in San Francisco
Advanced Machine Learning

Tyler White, Chuan Xu

March 5, 2018

Abstract

In this project, we worked with parking datasets given to us from Parknav. We used an ensemble of random forest models and gradient boosting models to determine where parking would be available. We found ensembling many models together would give an improvement to our accuracy.

We also found cross validation to be a much better approach to validating our predictions than one carefully constructed set. After moving to cross validation, the results of the test set were much more in line with our validation set.

1 Team Members

Our team includes two members, Tyler White and Chuan Xu, whose kaggle ids are tylervw and carriexu24, respectively. Our code is posted on `https://github.com/USF-ML2/final-project-carrie-tyler`.

2 What data?

The datasets we got from Parknav for this project contain SF Sensor Data, SF Parking Records, and Parking Spot Availability Data.

SF Sensor Data is data collected by parking sensors in San Francisco, ranging from April 2011 through July 2013. Parking Records are location and time information of parking, as measured by parking meter activity. The dates range from March 2015 through September 2017.

Parking Spot Availability Data provides segmented street occupancy information based on date and time. The length of the street is also provided. This dataset is split into train set and test set, which have 1100 and 726 records, respectively. In the train set, we have the information of whether any spot is available as well as the real number of spots available. Our goal is using this information to predict whether we could find any parking spot in the given area in the test set.

One problem is that the date range doesn't match very well in the train and test set. We only have a couple of months of data from Jan 2014 to Mar 2014 to train our model. However, we need to predict the parking availability from Mar 2014 to Nov 2016. Therefore, we are very likely unable to observe trends over time that might exist.

We also check the location overlap between different datasets. We find that although the location of the train set and the test set are perfectly matched, there is less overlap between the sensor and parking data. The two datasets have a very different format, so it is not obvious how to combine these together. In figure 1, we show the locations of the various sources of data.

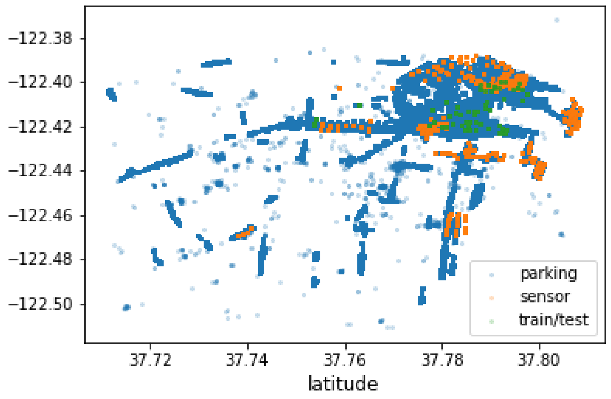

Figure 1: Scatter Plot of Location Overlap of Datasets

3 Techniques Overview

To get more and better features, we tried many methods. First of all, we added some common time features like `day of week` and `hour of day`. We also checked if the day was a holiday or not using Python library `holidays`.

Second, we applied regularized mean encoding and quantile encodings. We encoded the number of real spots for each of the categories in Street, From, To, hour, and day of week. For each category we used quantile encoding on we found the 0%, 25%, 50%, 75%, and 100% quantiles. Unfortunately, due to the large number of outliers using too many of these features overfit the data. So, we made many models using varying combinations of the mean and quantile encodings. Some combinations of these features worked well and others did not.

In figure 2, you can see a box plot of the number of `Real.Spots` vs `hour`.

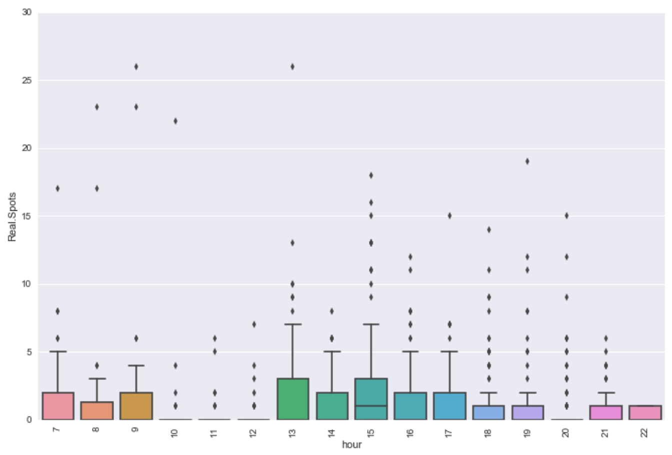

Figure 2: Box Plot of the Number of Real.Spots vs Hour

Third, we get the latitude and longitude of the street intersections thanks to the package `geocoder`. These features help improve our model performance a lot. We also tried mapping information from sensor data and parking data back with our train and test set, but there was a limit for further improvement of our models.

In figure 3, you can see a box plot of the number of `Real.Spots` vs `street`.

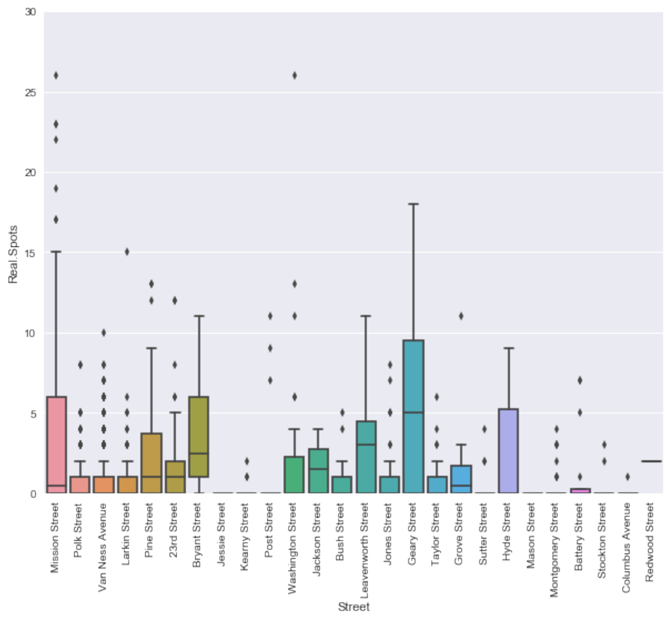

Figure 3: Box Plot of the Number of Real.Spots vs Street

We mainly used two machine learning methods for this project, random forest and XGBoost. Overall, XGBoost gave us better prediction results but random forest helped us to get better insights of the data. At the end we took our most successful models and performed a weighted vote. Each model was given a vote with a weight equal to our Kaggle score for that. If a single observation had more than half the total weight then it predicted a parking space.

4 Experimental results

We found that it was often very easy to fit to any given validation set. So one very distributionaly similar validation set could have a completely different score then the test. We had two ways of trying to make sure that we did not overfit. First, ordered the data by the time. We split them by the first 70% and the last 30%, as well as an 75/25 and 80/20 split. We found that often we would get a very high score on just one of the splits but do quite poorly on all other splits. We used this for a while at the beginning.

We later moved to cross validation. Due to the short period of time we had data collected in the same manner as the test, we could not determine any overall trends. Given this fact, we found that cross validation produce reliable validation scores to the test. After moving to cross validation the results of the test set were much more in line with our validation set.

5 Conclusions

Using gradient boosting and random forests can overfit to a specific validation set fairly easily. This can be combated by using several splits when validating. We found cross validation to be a much better approach to validating our predictions than one carefully constructed set. We also saw that ensembling many models together gave an improvement to our score.

6 Team responsibilities

Chuan performed most of the exploratory data analysis. She also worked on combining the sensor data in a meaningful manner. She created the random forest models. Tyler determined the best way to create a accurate validation score against the test set. He created the gradient boosting models. Both Chuan and Tyler worked on the final report.