Advanced Machine Learning: Parking Spots Availability Prediction

Kunal Kotian, Yu Tian

Abstract

This paper describes the data exploration and modeling conducted on the problem of availability of on-street parking in San Francisco. It describes how we cleaned the data provided by Parknav, used external information to augment our understanding of the data, merged two separate datasets, and generated several features to help our models learn better. We also provide a description of the machine learning approaches adopted to predict whether a particular street segment has parking spots available.

1. The Problem and the Data

Parknav has developed an app that shows the likelihood of on-street parking is available at a given time. Parknav uses data collected by public sources, combined with its own manually collected labeled data (counting the number of parking spots available on a street) to build models to predict the probability of street parking being available. The task assigned to USF's MSAN students was along similar lines - can we build a predictive model using data on parking meters and the limited number of manually labeled observations? For this project, Parknav provided 4 data files to USF, namely, a labeled training dataset, a test dataset, and 2 unlabeled auxiliary datasets from SFMTA.

1.1 Main Training Data

In the labeled training data, each record specified the street the validator (possibly Parknav employees or subcontractor) was driving on, the last perpendicular street he/she crossed, and the next perpendicular street they were heading towards. Knowing these 3 values allows us to uniquely identify a street segment between 2 intersections. The training data also provided the observation (label) of the number of available parking spots and its binary version (parking available/unavailable). A timestamp was also provided for each observation. Additionally, use of data from external sources was encouraged.

A key challenge in this project was the relatively small size of the labeled training data - 1100 observations. Finally, the training data was mildly imbalanced with 63.5% of observations being negative labels (parking unavailable).

1.2 Auxiliary Data

Data on the transactions of smart parking meters, collected by SFMTA, was provided. Parking sensor data (also collected by SFMTA) containing information on the triggering of parking sensors was also provided.

1.3 Data Cleaning Challenges

The training and testing data had incorrectly recorded "From" and "To" streets, constituting nearly 30% of the training data. These records were manually reviewed and corrected, often based on educated guesses.

The street intersections available in the training data were geocoded into their GPS coordinates using Google Maps API. It was later found that there were some inconsistencies in the way Google Maps API returned the latitude and longitude for certain intersections; these cases were found through a review of the training and parking records data and were corrected manually.

2. Feature Engineering

2.1 DateTime Based Features

For prediction purpose, we parse DateTime to year, month, week, day, the day of the week, the day of the year, whether is the month end, whether is the month start, whether is the quarter end, whether is the quarter start, whether is the year-end, and whether is the year start. Later, it was seen that there was very little overlap in the months covered by training and test data; hence, we removed all month-related features.

2.2 Parking Records

Geocoding the intersections in the training data allowed merging the parking records dataset with training data and allowed the use of parking records for feature generation. Every meter in the parking records data (as identified by its GPS coordinates) was assigned to a street present in the training dataset, based on the meter's distance from every street in the training data. This allowed computing an estimate of the total number of smart meters (and, by proxy, the number of parking spots) present on every street. The coordinates of all smart meters in San Francisco were clustered into 20 clusters and each meter was assigned to a cluster. These clusters were later used for calculating aggregate statistics for feature engineering.

2.3 Rotation of GPS Coordinates

Most training (and hence, test as well) street segments are sandwiched between Mission Street and U.S. Highway 101. A large proportion of streets in these areas are in a grid that is aligned with the direction of U.S. Highway 101. U.S. Highway 101 has an angle of ~11.2 degrees with the north-south direction as it passes through north-east San Francisco. Thus, rotating the entire map clockwise around a point in the middle of the training data by an angle of 11.2 degrees would have the effect of aligning most of the streets with the north-south and east-west directions. Such alignment makes it easier for a tree-based classifier to identify useful

splits on the GPS coordinates. Hence, we added new features recording the rotated coordinates of street intersections.

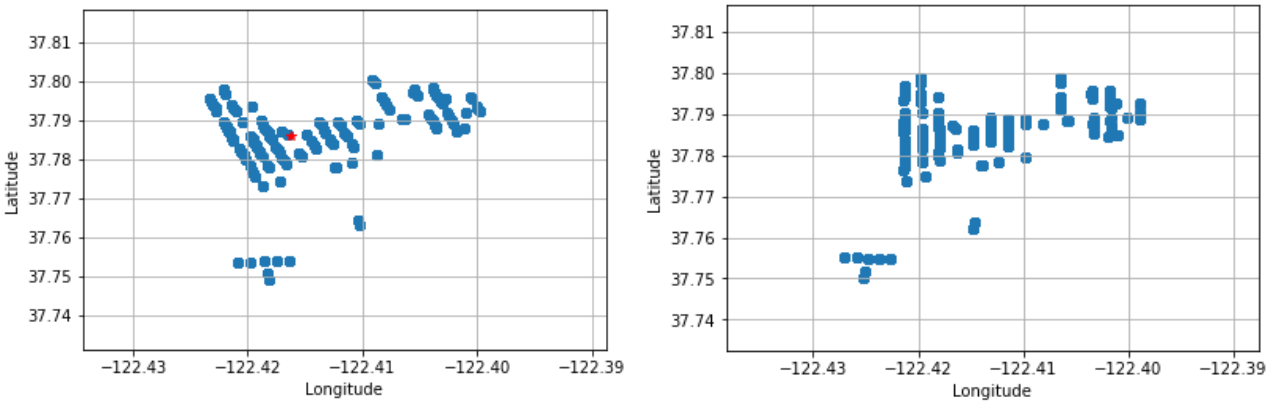

Figure: The GPS and rotated coordinates of the street intersections in the training data

2.4 Parking District

SFMTA divides San Francisco into "parking districts". We manually added parking management district ID and area type, obtained from DataSF's SFMTA Parking Management Districts map to the training data. This allowed us to add mean encoded features based on the parking district.

2.5 Mean and Target Encoding

We employed extensive mean encoding to generate features based on the mean of street length and number of meters group by parking cluster, parking management district id for "from" address, and parking management district id for "to" address, and several other variables.

2.6 Data Preparation Prior to Modeling

For variables that are not of numeric type, a transformation needs to be applied to change them to categorical variables; one-hot encoding was used for categorical variables in this case. The resulting train and test data includes 400+ variables. As the data is imbalanced, we would also need to evaluate whether there is a need to under-sample or over-sample one of the classes for modeling purposes.

3. Machine Learning Methods

The objective here was binary classification and the column "any_spot" was used as the target variable that the model predicts. As the data is mildly imbalanced, with 63.5% of training data belonging to the negative class, we under-sampled the records belonging to the majority class (negative label).

We used 5-fold cross-validation to tune the model and obtained the optimal hyperparameters. In 5-fold cross-validation, the training data is randomly split into 5 folds, with the scoring parameter for cross validation being F0.5 score.

Logistic regression, XGBoost, and Random Forest were used for modeling. For each method, the best estimators from each fold were recorded and an average score was used as a reflection of the goodness of model.

4. Results

- The ensemble of best logistic regression models based on 5-fold cross validation resulted in an average F0.5 score of 0.6929, with precision 0.6935 and recall 0.6932.
- The ensemble of best XGBoost models based on 5-fold cross validation had average F0.5 score of 0.69, precision 0.70, and recall 0.70 on the validation set.
- The best random forest model resulted in an F0.5 score of 0.72, and had precision 0.72 and recall 0.72 on the validation set.

5. Conclusions and Lessons Learned

<u>Starting with simple models and available data</u>

- We spent a lot of effort early on trying to figure out ways to merge parking records and sensors datasets. Instead, we should have started building simple models using only the training data.

<u>Importance of background research</u>

- We found that it is often very easy to find a lot of additional information on the provided data by simply searching for the column names and file names.
- We found a treasure trove of information on DataSF.com related to parking meters and sensors. In particular, we were able to locate a publically available analysis of the parking sensors data published by SFMTA that included key observations such as the fact that some of the sensors were faulty and their data had to be ignored.

<u>Benefits of a well-designed workflow</u>

- At the start of the this project, we spent quite some effort on devising a workflow strategy that we would follow for the duration of the project. This involved setting up an Amazon S3 bucket for easily sharing data between team members, fixing the project directory structure, and commiting code to our github repository in regular intervals.

Appendix

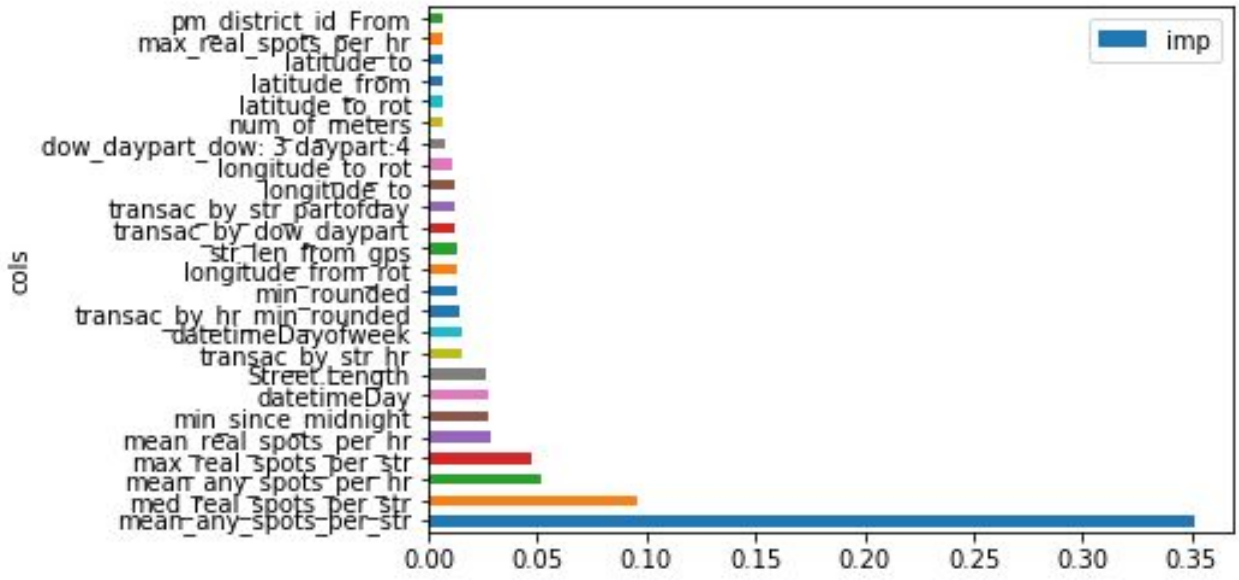

Figure: Feature importances from the random forest model

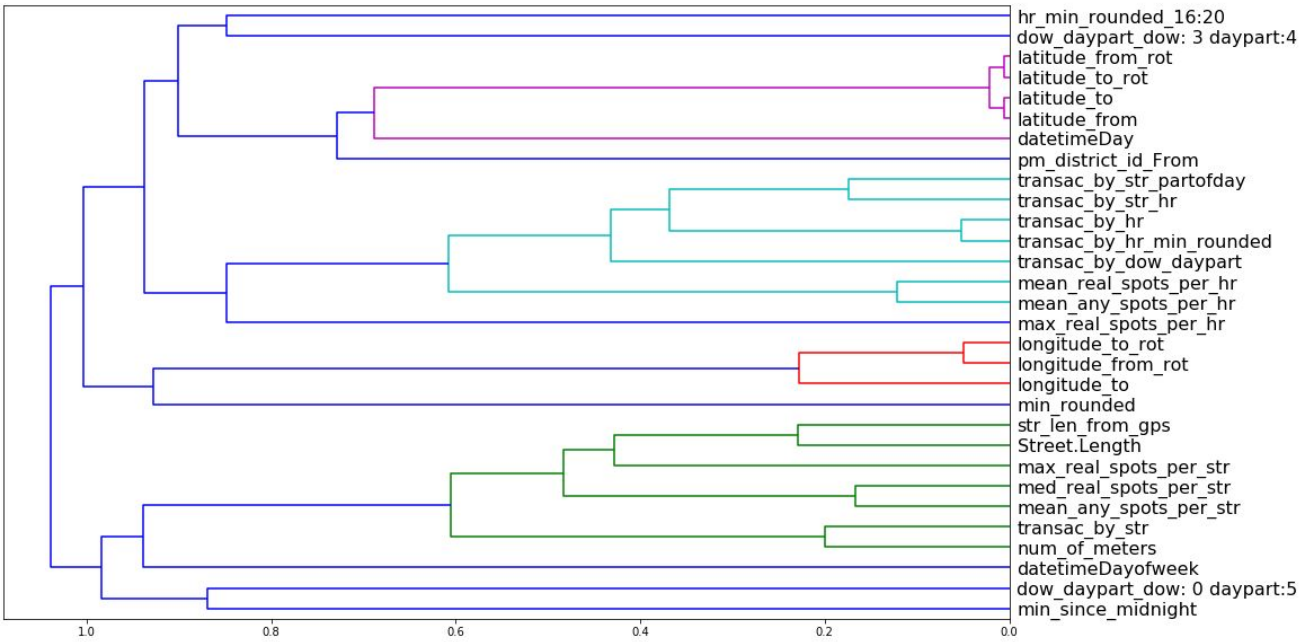

Figure: A dendogram of one of the classifiers in the random forest model