Abstract
San Francisco is one of the most congested cities in United States, and people always have trouble finding a parking spot, especially in downtown area. This has resulted in more and more people choosing to live outside San Francisco and commute using public transport such as BART. In this project, we aim to predict the availability of parking spots in certain street segments of San Francisco. Also, we are interested to study the factors that are influencing availability of parking.

Team Members

Team Member	Kaggle id
Nishan Madawanarachchi	nishan89
Qian Li	lqian5

Dataset
Data for this project was provided by ParkNav. This included three datasets.

1. Survey data ('train-parking.csv')
   A row in this dataset tells you how many parking spots are vacant for a given time of the day in a particular street segment in San Francisco. Target variable for this exercise is 'any_spot', which indicate whether parking was available (1) or not (0). This dataset is relatively imbalanced with 63.5% of all the instances being parking not available.

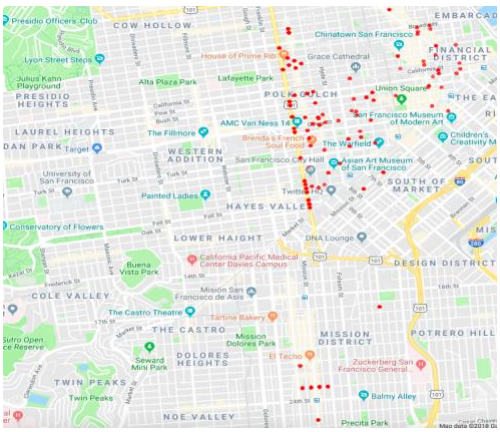
Sample rows of the dataset

Street	From	To	Date	Time	Real.Spots	Street.Length	any_spot
Mission Street	25th Street	26th Street	1/7/2014	16:19	4	179.132970	1
Polk Street	Ellis Street	Olive Street	1/18/2014	20:42	0	52.740210	0

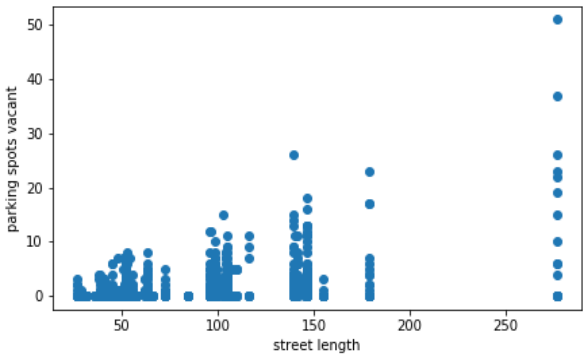
For e.g. there are 4 vacant parking spots on 7/1/2014 at 16:19 hours in the segment of the Mission street between 25th street and 26th street.

This dataset has 1100 rows and 7 columns. There are 96 unique street segments of interest, i.e. unique combinations of Street, From and To columns. Time period of the dataset is from 7/1/2014 to 3/28/2014 and time frame of interest has been 07:00 - 23:00 hours.

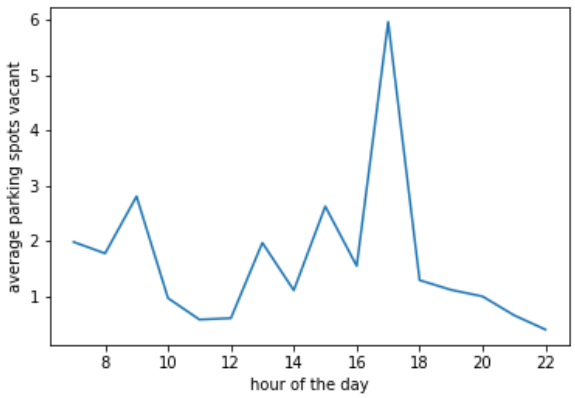
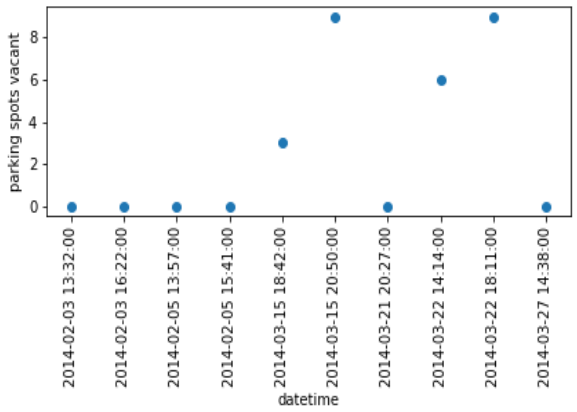
Geographic distribution of data collection points (i.e. mid points of street segments marked in red)



Change in number of parking spots available with respect to street length



Dataset only contain certain street segments within San Francisco (as seen in image on right). Number of observations for a street segment varies between 2 and 29. As we would expect there is a strong correlation between street length and parking spots vacant.

Change in parking spots available with hour of the day	**Rows for a typical street segment (e.g. segment of Hyde street between Golden Gate Avenue and McAllister Street**
	

Plot indicate that it is relatively hard to find vacant places between 10am-12pm and 9pm-10pm. And we also want to highlight from the plot on right, that we only have parking information for certain datetimes.

2. Test data ('test-no-label-parking.csv')
   This dataset contain test data we are using to make the final prediction. Test dataset contains 726 rows and 5 columns. It has all the 96 unique combinations of Street, From and To. Time period for the test data is 3/28/2014 to 4/11/2016.

3. Sensor Data("ParkingSensorData.csv")
   This dataset contains information on occupancy of parking spots extracted by sensors installed in certain parking spots in San Francisco. About 5% of the data provided here overlap with 29 out of 96 unique street segment combinations.

4. Parking payment data('parking-records.csv')
   This dataset provides information on location of a parking meter and the datetime a payment is made.

5. External data
   We extracted past weather information, specifically temperature and precipitation data in San Francisco.

Techniques Overview

Evaluation metrics: F0.5, log loss
Our target variable is any_spot, which is either 0 or 1, so this is a binary classification problem. F0.5 is the final evaluation metric in the competition. In addition, we used log loss to give consistent performance during modeling particularly due to fact that, dataset is small and unbalanced.

Methodology: Random Forest, XGBoost, Logistic Regression
Three classification algorithms were used initially, and later we put more focus on Random Forest and XGBoost. Random Forest performed better compared to XGBoost in terms the evaluation metrics (probably due to dataset being small). However Random Forest tend to produce unstable results, so both Random Forest and XGBoost were used interchangeably.

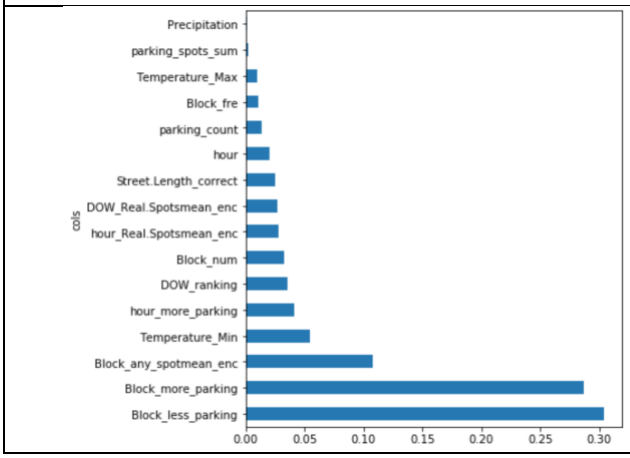
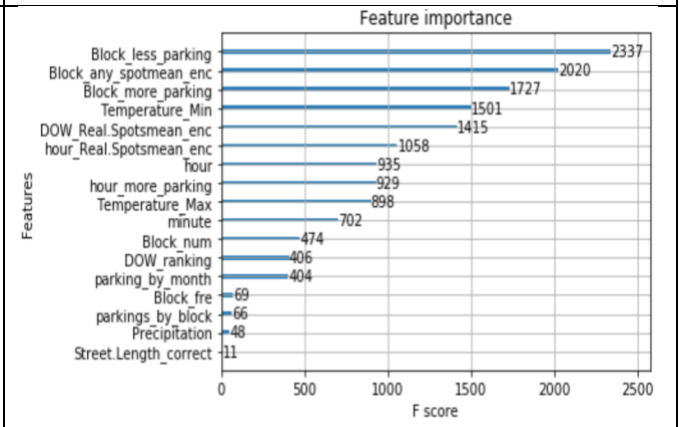
Train and validation set:
We have 1100 rows of training data and 726 rows of test data, in a ratio of roughly 1.5 to 1, and test data has more recent dates from train. We decided that a ratio of 1.5 to 1 is too large for train and validation split. So, we split training data by datetime to train and validation in percentage of 80% and 20%.

Feature engineering:
We extracted new features from the training data to increase the predictive power of our models. And some feature columns from weather data were kept as it is.

Techniques used to generate new features:

- Datetime information extraction: obtain day of the week, hour, month using datetime column
- String concatenation: “Street”, “From”, “To” are concatenated to create unique “Block” combinations
- Mean encoding: numerical feature encoding on categorical features (not only limiting to the target variable)
- Geo encoding: attach latitude and longitude to street segments. Done by taking the average of latitude and longitude of two junctions corresponding to a street segment

Features are created and modeled using random forest and XGBoost at first to identify importance of our features.

Feature importance with Random Forest	**Feature importance with XGBoost**
	

From this we were able to identify most important features and remove less important ones. However, we only got rid of features if the act of removing features does not result in significant reduction in predictive power.

By repetitive experimenting on train and validation set, we limit down features to the following:

Selected Features	
Column	**Description**
Hour	Hour of the day
minute	Minutes for a given hour of the day
Street.Length_correct	Since there were few mismatches in street lengths, we have corrected it by taking a median of street length after grouping by street
Block_num	Refers to a unique Street-From-To combination. We use this information to identify each small segment of streets.
parkings_by_block	This feature is extracted from Parking Record data, and it represents the count of record for each block. The location of block is interpreted from latitude and longitude.

parking_by_month	Number of parking spots estimated per month from Sensor data
Temperature_Max	Daily maximum temperature from weather data.
Temperature_Min	Daily minimum temperature from weather data.
Precipitation	Daily precipitation from weather data.
hour_more_parking	We grouped any_spot by hour and identified hours that have a relatively higher chance of finding an empty parking spot.
DOW_ranking	We grouped any_spot by day of week and ordered day of week by mean of any_spot.
Block_more_parking	We grouped any_spot by block and identified blocks that have a relatively higher chance of finding an empty parking spot.
Block_less_parking	We grouped any_spot by block and identified blocks that have a relatively lower chance of finding an empty parking spot.
'Block_any_spotmean_enc'	Mean encoding of any_spot on Block
DOW_Real.Spotsmean_enc	Mean encoding of Real.Spots on day of week
hour_Real.Spotsmean_enc	Mean encoding of Real.Spots on hour
Block_fre	The frequency of block showing up in the training data.

We have also included details in appendix for an analysis done with sensor data, which did not really help to improve the predictions.

Experimental Results

Hyper parameter Tuning

Hyper parameters are carefully chosen in each model. For Random Forest, cross validation and grid search were used to produce an optimal combination. However, manual interventions were required to be relative conservative (i.e. to not to overfit validation data) when picking the best hyper parameters. This meant sometimes not using the hyper parameters which gives the best validation score. We chose the combination that won't differentiate log loss on train prediction and validation prediction. For XGBoost, since it is more stable, we perform linear search to help select hyper parameters. Similar to Random Forest, we try to keep the hyper parameter values as conservative as possible.

Example score

Below is one set of scores when experimenting with different set of features and model hyper parameters. The XGBoost model, for instance, achieves a F0.5 score of 0.559 in public leaderboard.

	Precision (Validation Set)	Recall (Validation Set)	F0.5 (Validation Set)	F0.5 (public leaderboard)
Random Forest	0.567	0.567	0.522	0.563
XGBoost	0.5	0.478	0.495	0.559

Stacking

We did stacking on previous submission files to improve prediction performance. The stacking methods used were majority vote on top 3 Kaggle submissions and averaging predicted probabilities of top 2 Kaggle submissions.

Conclusions

Overall it was a challenging exercise due to the nature of training and test datasets i.e. dataset being smaller and lot of dynamics hidden in between sparse nature of the records. Out of the external data we extracted we only used weather data in the final model. And we felt had we used more external data we could have captured some of these dynamics better.

Key lessons learned

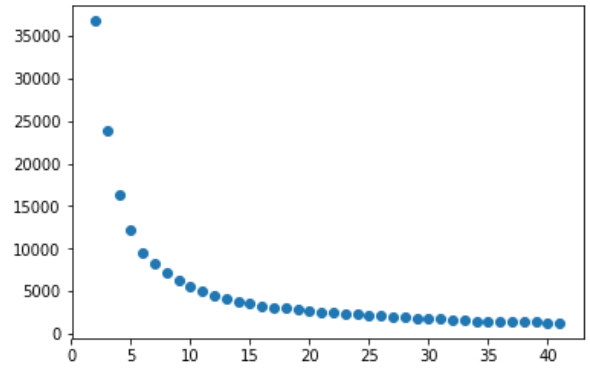
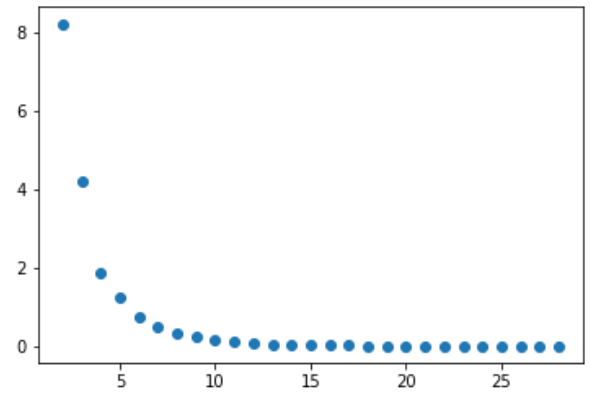
- Since it is a smaller dataset creating more and more features organically does not improve the performance of the model after some point and it in fact results in poor model performance.
- Also due to dataset being smaller, our predictions on the validation set were not stable, as the same combination of hyper parameters would result in different prediction performance. In order to counter this, we used cross validation, smaller learning rates with Xgboost and conservative selection of hyper parameters.
- With feature engineering steps like mean encoding, we had to make sure outliers are taken care of. Otherwise, means calculated will be skewed. We saw couple of outliers in street lengths and real spots in the training data and handled for them.
- When using sensor data, vacant time of the parking space may over represent the availability of the space. So, it was important to calibrate for this using availability of the streets in general from the training data.
- The dataset provided here is somewhat imbalanced. We tried to handle for this by oversampling. However, since we used mean encoding we had be careful to do mean encoding before oversampling to retain the dynamics of the dataset.

Appendix

Summary of analysis of sensor data

As mentioned previously, only 5% of the sensor data provided overlap with 29 out of 96 unique street segment combinations. However, sensor data is rich with 425,635 rows corresponding to the 29 streets. So, we thought of exploiting this relatively larger dataset for prediction rather than the original training set.

To do this first hurdle was to represent all rows in training data in a way that is compatible with sensor data.

Change in cost with different K value of KNN algorithm for rows in training data	**Change in cost with different K value of KNN algorithm for 29 streets of interest in training data**
	

We tried to find similar rows in the training dataset using day of the week, hour of the day and real parking spots using KNN algorithm. Plot in the left shows how the cost gets reduced with different values for number of clusters (K). We thought K=15 is good a value. So, this means each row in the training dataset can be put into 15 clusters.

Then in training data, we calculated the mean cluster value for a given street segment. So, each street segment will have it own mean cluster value. Afterwards, we wanted to find the similar streets based on the mean cluster values. For this we performed KNN on the mean cluster values to find similar streets. Plot on the right shows the results of this experiment. We decided that K=10 is a good value for this. So now we can represent each street segment in the training, testing, and sensor data with an id from 0-9. Basically, we are able to find 10 types similar street segments.

After this is done we removed Street, From, To columns in training and test datasets and Block information in sensor dataset and used new KNN id to represent each row (Now both datasets are compatible). Then we did feature engineering, used our original training data as validation dataset and applied XGBoost algorithm to predict on the test data. However, we were only able to reach a validation f0.5 score of 0.46 in the training dataset and this translated in to a very low score in Kaggle.