Data Science at Scale Capstone Project Report – Predict Detroit Building Abandonment

07/13/2016

The project asks you to predict building abandonment ("blight") based on a number of public datasets, including criminal incidents, permit violations, and more. This project grew out of a roundtable discussion with Socrata and city analytics leaders in Detroit, Kansas City, New Orleans, Boston, and more --- these cities shared a common interest in being proactive about identifying buildings at risk for abandonment and taking steps to prevent the harms that arise as a result.

1. Data Processing

There are four files to use in this project:

- detroit-blight-violations.csv : Each record is a blight violation incident.
- detroit-demolition-permits.tsv: Each record represents a permit for a demolition.
- detroit-311.csv: Each record represents a 311 call, typically a complaint
- detroit-crime.csv: Each record represents a criminial incident.

For each incident type, the location is included as a latitude, longitude.

Each file also has an Address variable, for example in the Violation file, the address variable is "ViolationAddress" containing the text format address such as "5035 balfour".

To aggregation all the violation, permit, 311 call, crime files together, I need to merge the 4 files together. I tried to merge the 4 file using lat and lon, but the lat and lon variables all have missing cases; but the address variables are almost all complete on the 4 files. So I decided to aggregate the file using the Address variables first.

First, I concatenated the 4 files vertically to get a long table with 454538 rows. The 4 different address variables were put into the same variable called "addr". Three are 185113 unique Addresses (185113 unique Building IDs) among the 454538 rows. I sorted the file using Addr, Lat, Lon to fuzzy match of the similar addresses into one unique Building Address (Building ID). I deleted the stop words such as 'michigan', 'detroit', standardized the words as "avenue"="ave", "north"="n". Then I used python NLTK edit distance to fuzzy match the text addresses. If for two addresses, the text edit distance is less than 5 and the street number is the same, I give them the same Building ID. For example, at first addresses 'balfour st' and 'balfour stt' have different Building ID, they will have the same Building ID after collapsing. Through this fuzzy match, I cleaned the 185113 Building IDs into 165190 Building IDs. I didn't group the original table by Building ID, the table still have 454538 rows.

Second, for cases with Lat and Lon missing, I imputed the missing Lat as Lat Median 42.4, imputed the missing Lon as the Lon Median -83.10. Then I checked the cases with imputed Lat and Lon, if the Lat and

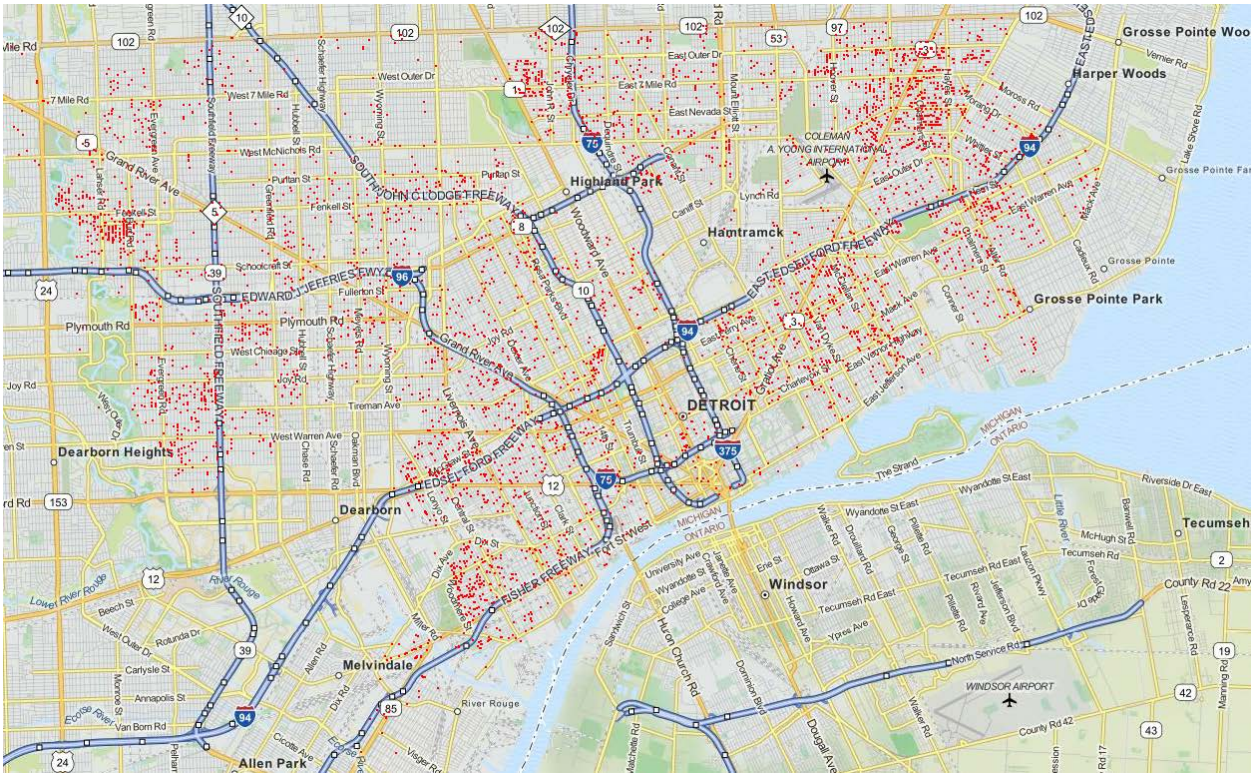
Lon are imputed and the length of the Address is less than 10, for example Lat=42.4, Lon=-83.10, Addr='balfour' with the address lengh equal to 7, I say the both the Lat/Lon and address information are not good enough, and I will records like this. There is only 1 record falling into this situation. I deleted 1 record out of 454538 records.

Third, I deleted the non-detroit outlier cases. I only keep records with Lat between (42, 43) and Lon between (-84,-82). 5 cases fell into this category. 5 more cases were deleted out of 454537 records, leaving 454532 records.

Fourth, I further collapsed the Building Address (Building ID) using Lat and Lon. At this time, the Building ID was defined purely based on the Addr variable. If the length of the text Addr variable is too short: less than 5, such as "bal", "ave", I assume that the text address is not reliable. There are 19189 records with 6864 unique addresses labeled as too short. For these buildings, I rounded the Lat Lon to 4 digits and sorted the cases using Lat Lon Addr (comparing with fuzzy match sorted by Addr Lat Lon, the sort order is different). Then for these cases, if the Lat Lon are the same, but the Building IDs are different, I collapsed them into the same Building ID. After this cleaning, I got 6832 unique Building IDs out of the 19189 short text addr records (originally there are 6864 unique Building IDs for these short addr cases), 32 Building IDs were collapsed using Lat Lon.

After all these cleaning, I got 165158 unique Building IDs in this concatenated table with 454532 rows. Below are final map of the Permit data.

Permits:



2. Modeling

As I explained initially, I concatenated the 4 files vertically to a long list, and I have a unique case ID to link all the records back to all the 4 individual files. After the address data cleaning and get the final Building ID variable, I merged the other variable from Permit, Crime, Violation and Call 911 back to the address list using the case ID. So the table looks like under the same Building ID, I could have 4 rows with 4 different Case IDs containing information from the 4 files. Next I need to group the long list by Building ID and sum up the information to the Building Level.

There are 7133 rows in the Permit file. The frequency of the variable BLD_PERMIT_TYPE is as below:

DISM 5859; Dismantle 1274

Both DISM and Dismantle should mean blight. I assume all the 7133 records in the Permit file are Blighted Buildings. I grouped the full address list by Building ID and use the Maximum Y as Y across cases under the same Building ID as the building Y. 6339 among the 165158 buildings are labeled as abandoned. Then I randomly selected 6352 (4%) of the left 158819 buildings as Non-Blight cases to make the model input file have similar numbers of Blighted and Not-Blighted.

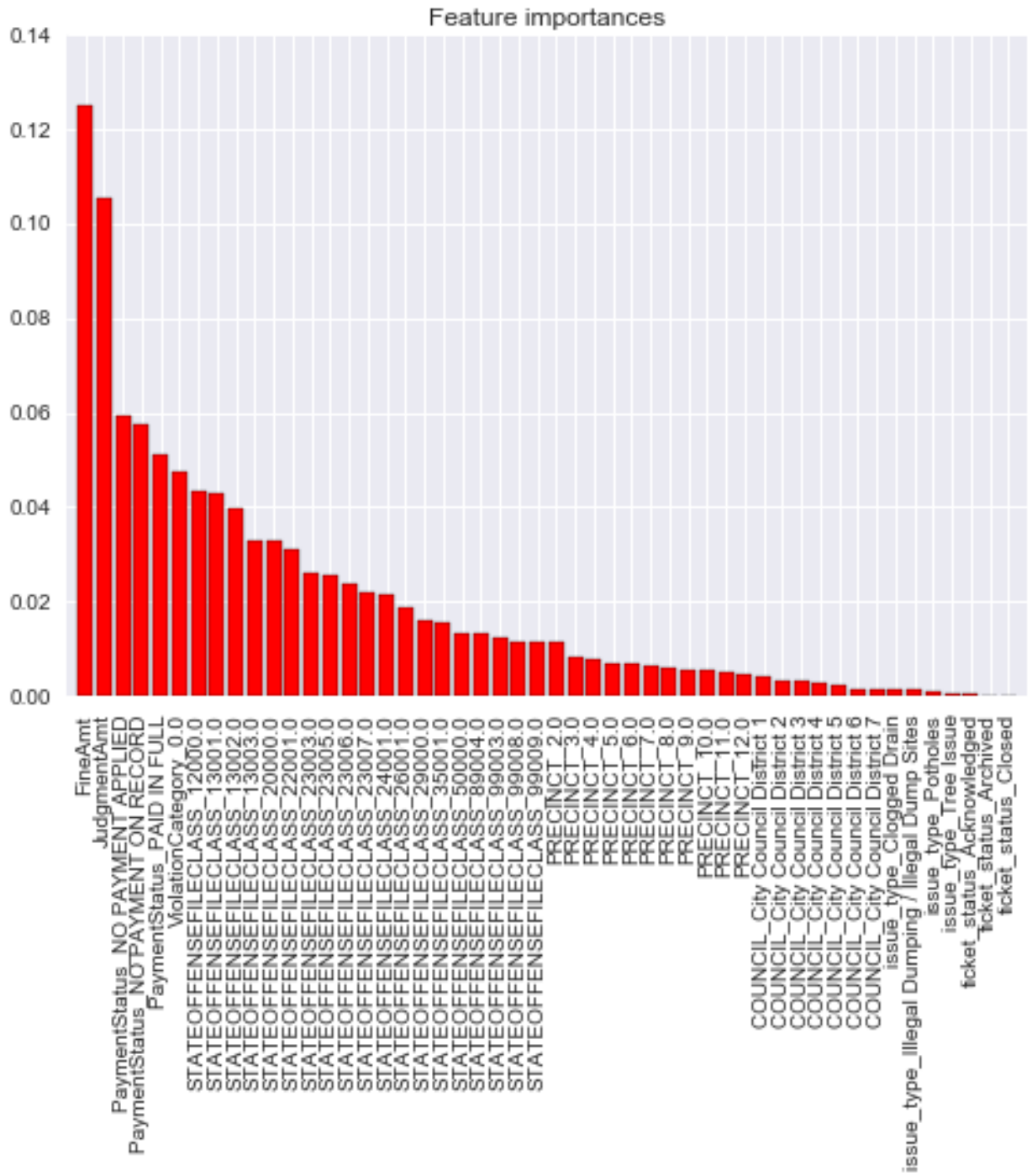
Y

1 6339; 0 6352

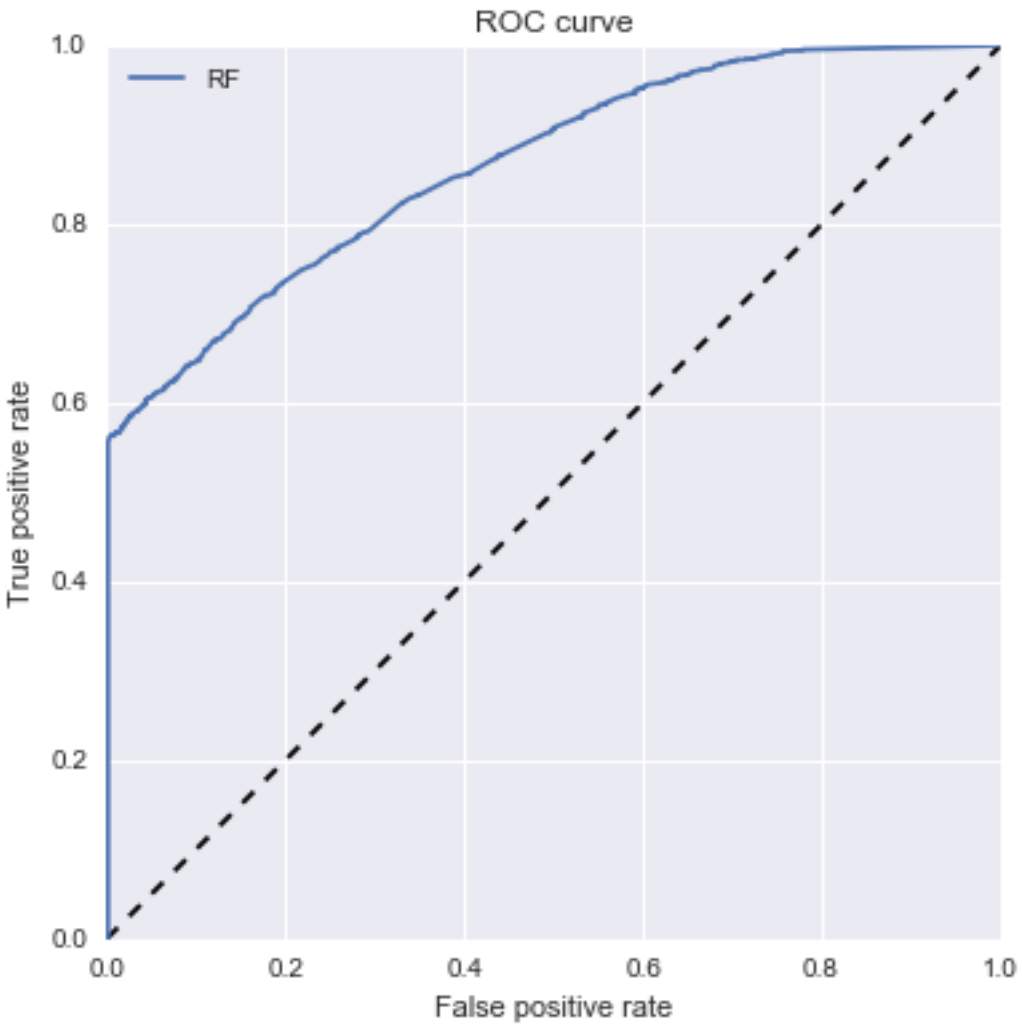
Then I considered the three continuous variables FineAmt, JudgmentAmt,rating and the categorical variables PaymentStatus, ViolationCategory, STATEOFFENSEFILECLASS, PRECINCT, COUNCIL, issue_type, ticket_status as predictors. I excluded redundant variables and categorical variables with too many levels. I use the value 0 to fill the missing cases of the 3 continuous variables FineAmt, JudgmentAmt,rating. Eventually I got 168 variables (165 dummy + 3 continuous).

I grouped the long list of addresses by Building ID and sum the predictors as the value for that Building. For example, sum of the FineAmt across one building ID is the FineAmt for the Building. Sum of the dummy variables for example ViolationCategory_0 is the Count of the Violations in the Category 0 for that Building. I selected the same 6339 abandoned buildings and the 6352 randomly sampled not abandoned buildings as the target variable Y.

All the 165 dummy variables and 3 continuous variable were used in a random forest model. I split the dataset with 12691 unique buildings into Train and Test with 7:3 ratios. The training X matrix has 8,883 buildings; and the test X matrix has 3,808 buildings. After running the random forest model, I ranked the 168 variables by their importance.



I made prediction using the random forest model on the test data. The model accuracy on the Test data is 77.21%. The ROC Score of the Test data is 86.52%.



3. Summary

- I established a list of 165158 unique buildings by collapsing the address, lat, lon variables. 6369 of the buildings are labeled as abandoned based on the Permit file. I randomly selected 6352 out of the left 158819 buildings (4%) as not abandoned. My complete model input file has 12691 buildings (abandoned + no abandoned) in total.
- I used 3 continuous variables and 165 dummy variables created from categorical variables as model predictors.
- I used 70% of the data as Training data to build the model, and 30% of the data as test data. My training file has 8,883 buildings; and the test file has 3,808 buildings.
- Using the random forest model, I got the accuracy of 77.21% and Roc score of 86.52% to predict the Test data.