Predicting Blight in Detroit

David García Miguel
15/05/2016

The problem

In this project we are presented with an urban decay problem in detroit in which we should be able to use open data to create a model that predicts blight in order to avoid this issue in the future or address it promptly. We are initially presented with four datasets:

- detroit-blight-violations
  This is the main dataset and consist of the registered blight violations in detroit with some annotations for the fines imposed.
- detroit-demolition-permits
  This dataset shows the demolitions allowed with the geo localization data presented
- detroit-311
  This dataset provides a set of calls from non-emergency situations that affect the condition of the neighbourhood like broken lambs, illegal burnings, noise complaints.
- detroit-crime
  A geolocalized crime data set that could be used to train the final prediction model.

Reproducibility

The code of this project can be found on github[1]. It has been done as a Spark Notebook. Spark notebook is similar to Jupyter/IPython notebooks but it only works with scala language and it is tailored to work with Spark. It comes with a few plotting features too. You can find raw scala code there too.

Tools

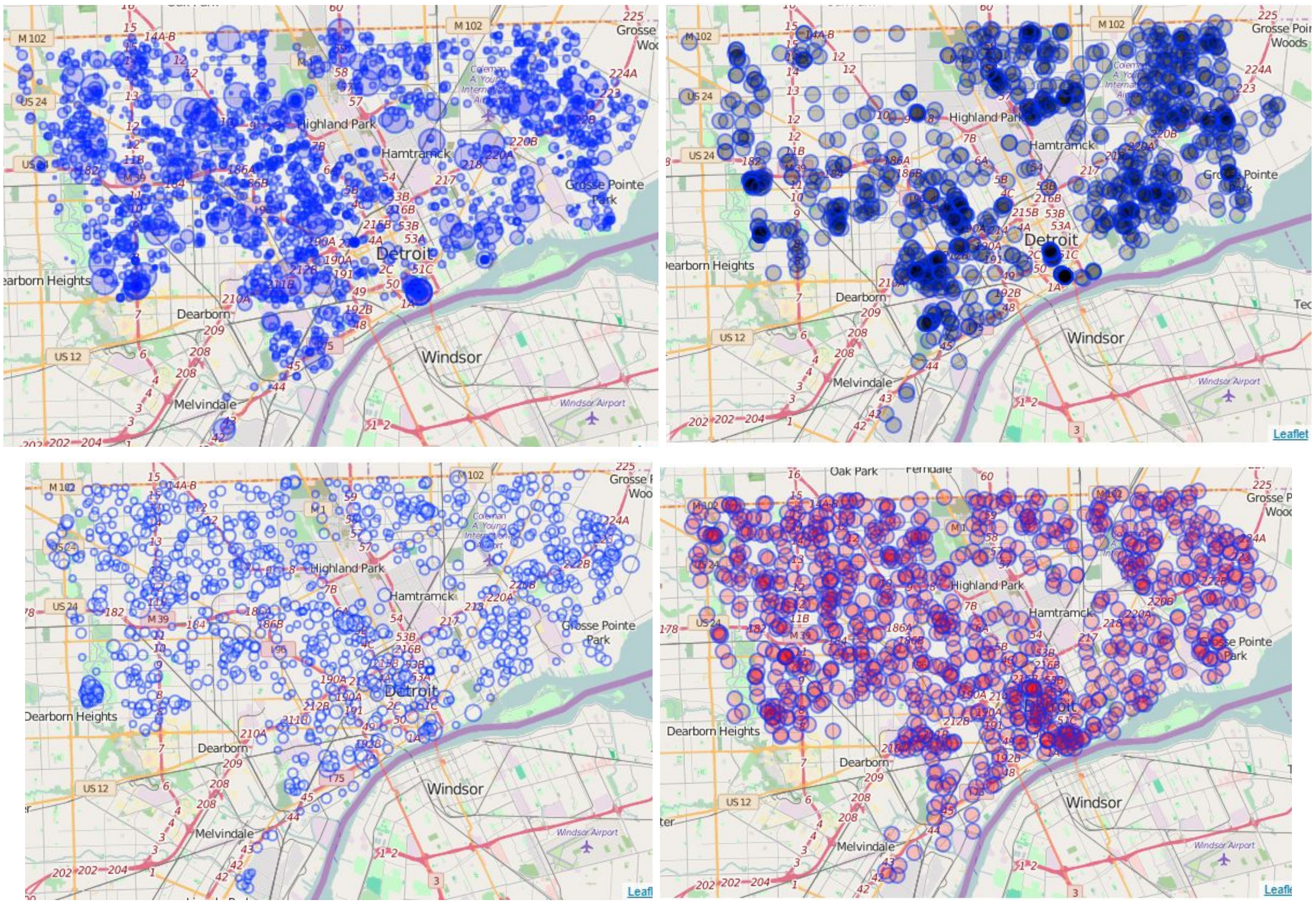
As this course states this should be "Data science at scale" (or at least scalable) so I tried to stick to this subject, and not use the regular python with scikit learn and R tools.

In this project I've used Apache Spark and MLlib to solve this problem. Apache spark is similar to Hadoop MapReduce but it presents a generalized computing model of Hadoop Mapreduce, allowing the user to define a DAG (Directed acyclic graph) in which to compute the different distributed steps. It also works as an in memory computing platform (as well as disk) so it is up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark can be used with several languages, such as python, java, R or scala. Scala language has been used to write the computing jobs for Spark because it offers the best compatibility with the tool (as Spark itself is written in scala)

Geolocalized data

In order to have a better understanding of the data I present here a map plot of all of our datasets



```
Blight violations → (left top)     → blue fill
Demolitions       → (right top)    → black fill
311 calls         → (left bottom)  → white fill
Crimes      → (right bottom)→ red fill
```

Building definitions

Defining building in this problem is not a trivial task and a lot of options arise. I could have tried to find a list of building out of the demolition permits because that is probably the most reliable source of buildings but I'd probably be lacking a lot of buildings to fit the rest of the data.

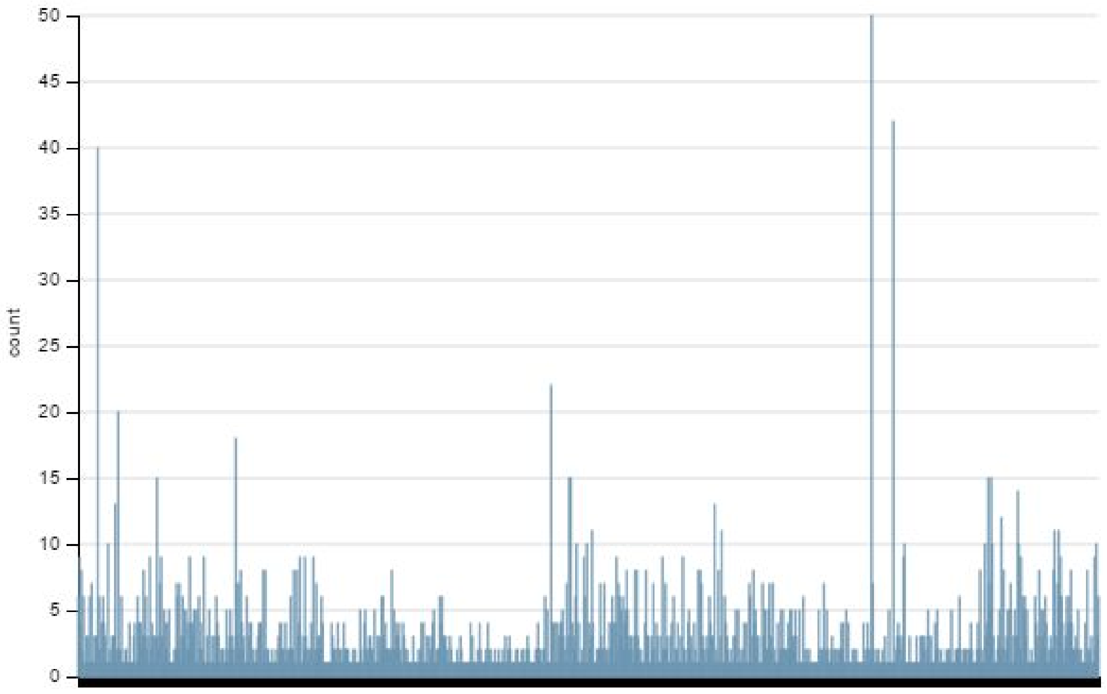
I've finally opted to use a grid-like approach to fit all the geo localization data into a "cell".
I was concerned about clusters around the edges of cells not being detected (because those evidences fall into different cells) so I've developed the following approach to avoid that to some extent.

1. I've defined **2 grids** using cells of "0.0001" degrees each side.

2. These 2 grids have the same cell size but the second one is overlapped onto the first one. They work like this:
3. Assume that our cell definition has “1” degree side to make it simpler
4. We get the following coordinates c = (40.734, 2.34)
5. **The first grid** will make those coordinates be in a cell that starts at (40.0, 2.0) to (41.0, 3.0)
6. **The second grid is overlapped** with the first one by half the cell side so the same test coordinate would fall into a cell (40.5, 2.0) to (41.5, 3.0)
7. At this time we have our initial coordinates “c“ placed into 2 different grids, now I **compute the center of those 2 resulting cells**.
8. Finally I **compute the distance** from our coordinates c t**o the center of those cells** and **select** the cell that is **closer** to the coordinate points c.

The methodology used for defining buildings in this project is pretty flexible and the cell size can be changed at any time, even independently by latitude and longitude so it is easy to iterate using different cells sizes to see which one fits the data better.

Using the method described above a with a cell size of “0.0001” degree by side I end up with 93489 buildings in the blight dataset with an average of 3.29 blight violations per building.
For reference this is the amount of blight violations per building for a sample of the dataset.



Count of blight violations by building (sample)

With this building discretization I account for 4875 building demolitions (out of ~6000 permits), 16908 buildings in the 311 calls dataset and 10547 buildings with 2 or more crimes.

Labeling

For the labels and since I’ve got a demolition dataset I’ll be discretizing this dataset with the same building definition technique and at this point I’m going to assume the resulting building area/cell as blighted.

Features

At this point I’ve got a method to identify buildings for each dataset so I can give each record a building id in which I’ll aggregate features from every dataset. I’ve tried to get a glimpse of each dataset into the features so I’ve decided to use the same building definition in each dataset and get these extra features.

- Blights
  - Building latitude
  - Building longitude
  - Counts of blight violations in the building area
  - Sum of accumulated “FineAmt” per building
  - Average “FineAmt” per building
  - Min “FineAmt” per building
  - Max “FineAmt” per building
- Calls
  - Counts of 311 calls in the building area
  - Average rating of the calls in the building area
  - Min rating of the calls in the building area
  - Max rating of the calls in the building area
- Crimes
  - Counts of crime incidents in the building area

Model selection

At first I’ve tried using a simple classifier (logistic regression) to iterate faster with the feature engineering and to test the pipeline with these new tools. Then I’ve used random forest algorithm because it scales really well in distributed environments and it is a nice match for a classification problem.
As requested I’m using 5 fold cross validation with both of these models. I’m using 80% of the dataset for training and the remaining 20% to test the accuracy of the model.

Results

After joining all the datasets features I had ~95k rows. Every row contains data about all of the datasets and this particular building lacks data from a given dataset it is not discarded, the missing data is filled with zeros

Training using a **logistic regression** models showed an accuracy of **67.62%** and the models was really simplistic because it was close to a trivial classification, it predicted almost every instance as a non blighted building.

Using a **Random forest** (10 trees) has showed and accuracy of **81.81%** and trained in less than 3 minutes even using a single machine (remember this solution is scalable because it is build on Spark)

Future work

- Given more time to solve this problem time related data should be used to train this model.
- On the other hand I came up with a limited set of blighted houses compared to non-blighted ones, so next time I should try to find more insights to identify more blighted buildings to train the model.
- Using better building definitions could improve the accuracy of the model. Maybe using detroit open data to extract the actual buildings instead of using a grid approach could help grouping the data.
- At this point I've only used data regarding the near proximity of a building but it could be fine to use other features that include further events

References and links

[1] https://github.com/NoxWings/Coursera-capstone