Using google map API grabs features to implement robbery prediction (May 2019)

Hsuan Yu Liu 823327369

**Abstract—** **I wonder know that relation between a feature of location between crime risk. Therefore, I use a crime data and street lights from SD city database, google map API, and Yolo to collect my data object. I focus on the robberies which happened in years 2010 to 2012, 626 events, and random picked 600 locations which don’t have robberies in 2010 to 2012. After that, I choice how many cars, how many windows, how many street lights, and distances to the closest SDPD, convenience store, gas station, and lamp to be features. Last but not least, I choice k-nearest neighbor and AdaBoost to train models and predict result.**

# INTRODUCTION

The area where has high probability of crime occurred might have similar features. For example, the place where doesn’t have street lights and people rarely walked along is easier to be robbed. Usually, robbers might try to target a victim who doesn’t close to a police station and might not dare to rob around polices. Moreover, robbers would try to reduce a chance to be seen when they rob, so distances to stores, convenience store or gas station, which open 24 hours might affect a probability of robbery. According to above assumptions, I used google map api, “nearbysearch”, “distancematrix” and “streetview”, to get street view images and distance to stores. Using dataset of crime gets information of locations where robberies happened.

The tools which I introduce into, not only documents and google map API, but also the YOLO object detection. It a detector which can assist to detect labels in an image. I used Yolo to detect features which aren’t provided by documents. For instance, how many cars or how many windows around a location. In this research, I used 237 images to train YOLO.

After data collected from 626 coordinates of robberies and 600 coordinates of safeties, I used K nearest neighbor and Gaussian Naive Bayes to train and predict. If the accuracy is good enough, then I am able to predict a probability of robbery of an unseen location.

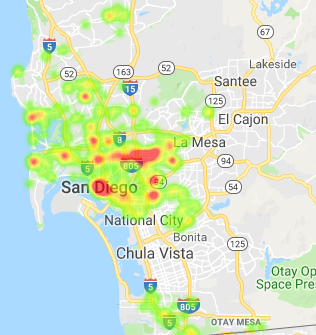
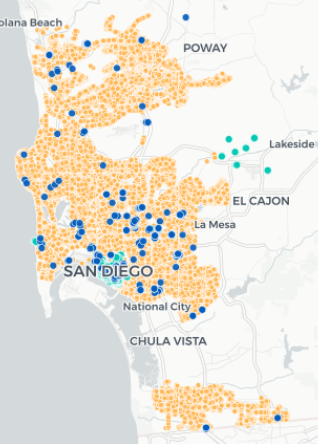
 

Figure A histogram of crime incident

Figure 2 A scatter plot of street lights

# Task Description

Before started to collect and clean data, I had to create data, lights and window, for YOLO. Then, trained a YOLO model for detecting lights and windows.

At first, I collected a dataset which contains features and labels. I choice windows, car, lights, distance to gas station, distance to SDPD, distance to the closest lights, and distance to convenience store to be features. Secondly, I parsed and extracted the information from the documents to build a dataset for training. Thirdly, use a coordinate in the dataset to get street views and distance to the closest those places. Fourthly, used the YOLO model to count number of target objects in street view and assign in the dataset. Finally, separated dataset into 2 parts and calculate accuracy of different classifier.

# Major challenges and solutions

## Challenges

The major challenge is how to detect object in a street view. It’s very difficult to detect an object because an image has much more attributes than a regular dataset. Moreover, lights or windows cannot use few attributes to describe and there are a lot of different styles of light. Therefore, it requires a complicated method to train a model to reach the object. The worse thing is that training a model by using images needs powerful CPU and GPU or it would take a long time.

## Solution

A popular object detection calls YOLO (You Only Look Once). It’s a detector which can assist me to detect an object. I just need to provide enough images and enough detail of labels. After created data, insert into the YOLO function, and it will start to train a YOLO model. My data only has 237 images, but it took 8 hours with i5-7200U CPU without GPU. However, the confidence of custom YOLO is low. Therefore, I use a document from San Diego government to find how many lights around a location. Although, my Yolo cannot supply good detection of lights, there is a good pretrain model which is good at detecting cars. It resolves a part of find cars.

# Experiments

## Data processing

Collect data from database: At first, I found a document of crime in 2007 to 2013 from San Diego regional data library. It contains different type of crimes and each of incidents has year, a distance to the closest lamp, and a coordinate. Also, there is a document which contains 61K street lights locations, from Data San Diego. According to those two files, I focused on the robbery which happened in 2010 to 2012, because a place where incidents happened in the past might have different street views. For example, the area might open a new convenience store or place a new street light in 2011 but the incident happened in 2009. Then the data collected by documents or google api don’t correspond to the real situation. After that, I followed the robbery incidents map in San Diego city to randomly pick 600 safe points to be safe locations data.

Collect data from Google API: Secondly, I used google api to grab street view for each coordinate of safe and crime. Every coordinate got 4 images for different heading. Then I could get 4 images around a coordinate. These images would be saved for later steps which detect features in images.

Figure street view Figure 4 street view

Figure street view Figure 6 street view

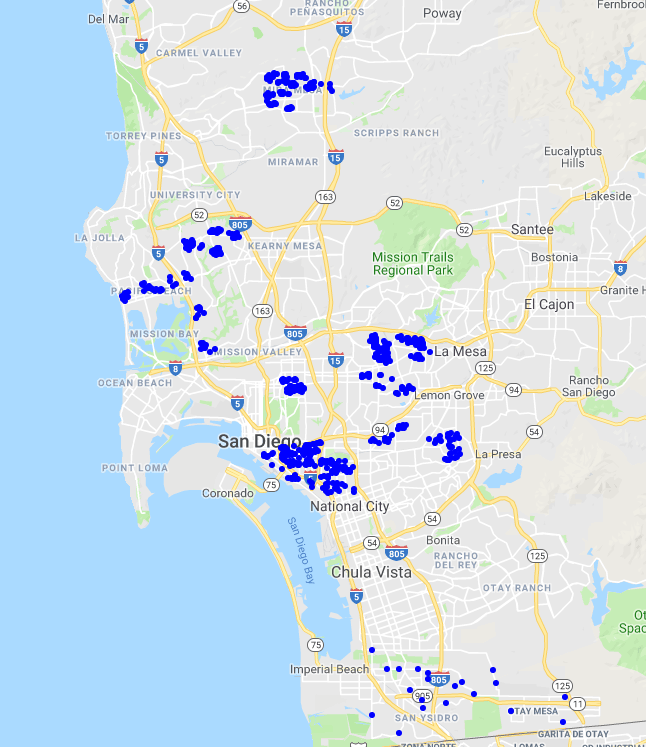
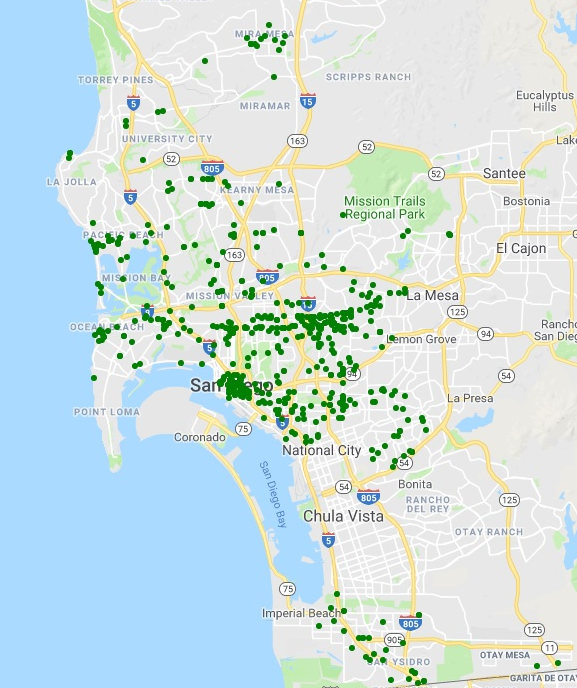
 

Figure Safe map Figure 8 Robbery map

Then using “nearbysearch” to get coordinates of SDPD, gas station, or convenience store could find the closest one. The google had returned a json data, so I would read data and extracted the closest for calculating distance. After extracting a coordinate, insert two points into “distancematrix” can get a distance between two points.

Detect objects in street views by YOLO: Third step, it’s the most complicatedly part, since I want to detect a feature an image. The way is introduced by me is YOLO. It’s a source which uses a pretrain data to train a label which I want to detect. For every image I had to select an area and give it a label name at beginning. Then I run the training function to train my model by those data which I created. I used 237 images with features of street light and window and took 8 hours to train my model. The model can detect most of street light and windows in an image. Also, the pre-trained model can detect cars, so I didn’t train my model to detect cars. After completely trained my model, use the model to scan every street to get a result. In training step, I didn’t use the result about street lights, since number of lights have already been assigned by using the document. It saves time to check the result about street lights. For example, in the Figure 9, there are two cars and one window. in the Figure 10, there are four cars.

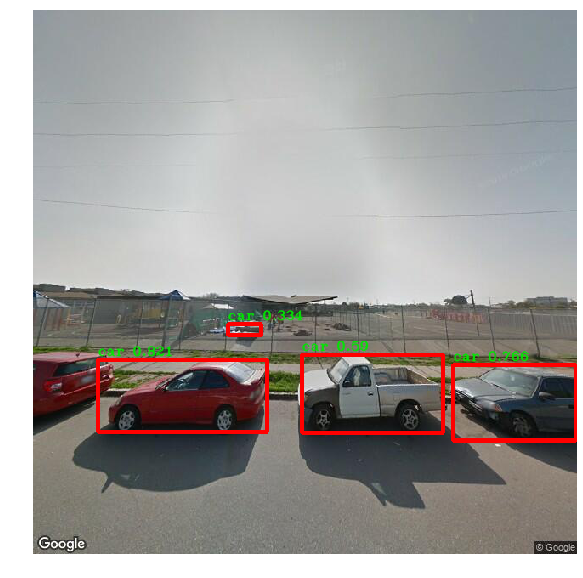
 

Figure Objects in the image Figure 10 Objects in the image

## Methods

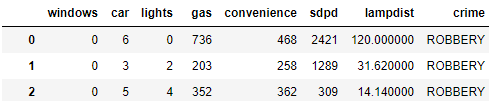
## Grab images and Label detection: 1. Google API (streetview, nearbysearch, distancematrix), 2. YOLO (Neural network).

Classification: 1. K nearest neighbor, 2. AdaBoost, 3. Normalization

## Major results

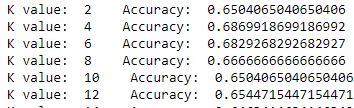
The dataset which contains columns, windows, car, lights, gas, convenience, sdpd, lampdist and crime. The X values are windows, car, lights, gas, convenience, sdpd and lampdist. The Y value is crime. Also, there are two datasets, without normalizing and normalized.

Without normalizing



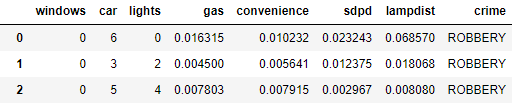
K nearest neighbor (Without normalizing):

Tried different K and found K = 4 has highest accuracy. It’s 68.69%



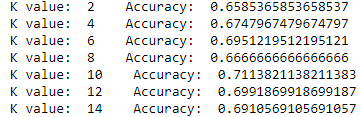
AdaBoost: Using AdaBoost with decision tree, I get an accuracy 80.8%

With normalizing



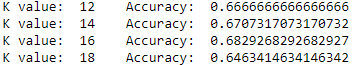
K nearest neighbor (Normalized):

Tried different K and found K = 10 has highest accuracy. It’s 71.13%



K nearest neighbor (Normalized without light/window):

Tried different K and found K = 16 has highest accuracy. It’s 68.29%



## Confusion Matrix (X: Predict, Y: Actual)

No Normalize, K = 4 After Normalized, K = 10

|  |  |  |
| --- | --- | --- |
|  | Safe | Crime |
| Safe | 80 | 33 |
| Crime | 44 | 89 |

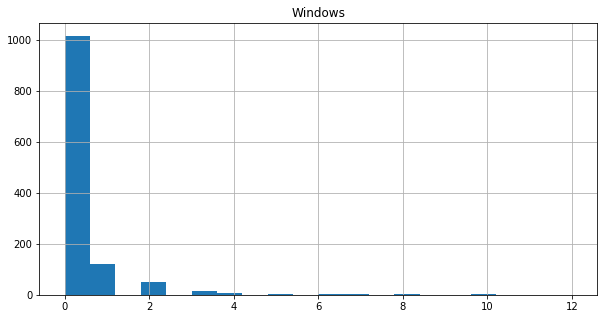
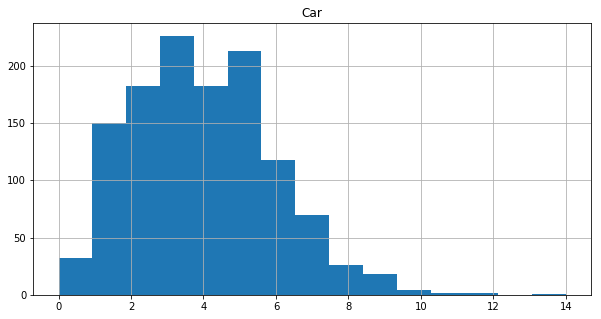
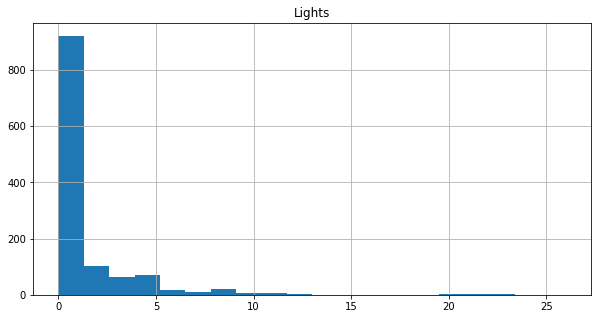
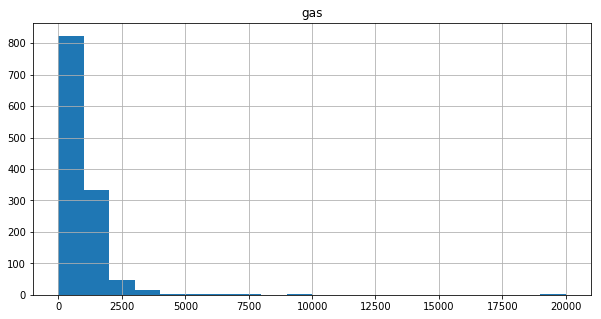
|  |  |  |
| --- | --- | --- |
|  | Safe | Crime |
| Safe | 71 | 42 |
| Crime | 29 | 104 |

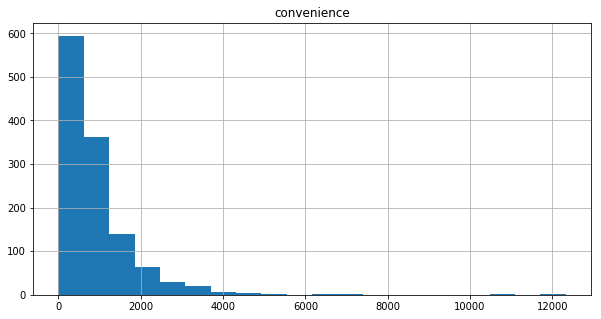
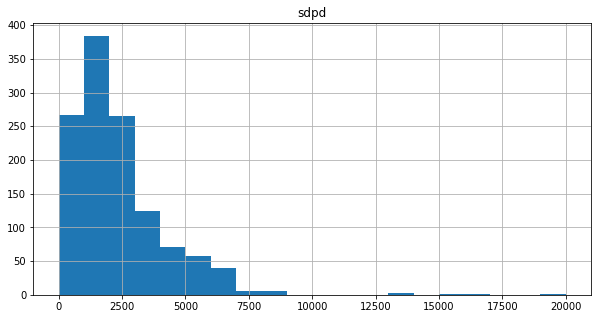
After Normalized without window/light, K = 16

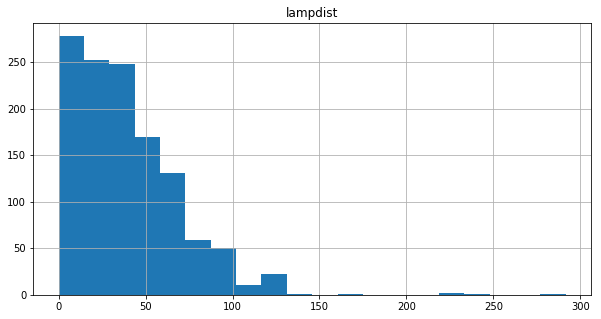
|  |  |  |
| --- | --- | --- |
|  | Safe | Crime |
| Safe | 75 | 38 |
| Crime | 40 | 93 |

## Analysis

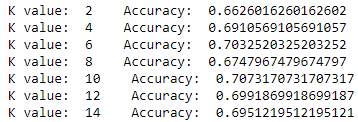
With histogram of each features, I found that most of windows are zero by histogram.



Consequently, I ignore the windows and lights features respectively, and the results are similar to the original one. It means windows and lights in my dataset don’t contribute to model training.



Further, that is because values of distances are larger than numbers, so the distances influence much more than numbers. Therefore, it should normalize those columns which record distance. Using the data dealt with normalization, the accuracy improved.

# Conclusion and future works

The result shows that the highest accuracy is only 71%. It isn’t good enough to support the assumption and the prediction. Additionally, my YOLO detection doesn’t provide high confidence in detecting windows or lights. It causes I cannot get enough information about windows and lights. I need more images and different sides of windows to train my model. The YOLO pretrain model demonstrates it can have very high accuracy to detect objects. If I have various data, then my YOLO will detect accurately. The more complete, the higher accuracy. In this project, I only considered 7 attributes. Some attributes might useful. Such as, width of roads, width of sidewalk, traffic light, or an area of location. If I can get those data, I can include not only robbery but also motor vehicle theft, or theft/larceny.

References

Database resource:

1. City of San Diego, “Streetlight point locations”, City of San Diego Open Portal, year 2017-05-03
2. City of San Diego, “City streetlights get smart”, City of San Diego Open Portal, year 2017-10-09
3. eric@clarinova.com, “incidents-100k”, San Diego regional data library, year August 7, 2013
4. Trieu, “darkflow with YOLO”, GitHub link: <https://github.com/thtrieu/darkflow>, year 2018-05-24
5. Google, “Street View Static API”, Google Maps Platform, link: <https://developers.google.com/> maps/documentation/streetview/intro
6. Google, “Places API”, Google Maps Platform, link: https://developers.google.com/places/web-service/search
7. Google, “Distance Matrix API”, Google Maps Platform, link: <https://developers.goo> gle.com/maps/documentation/distance-matrix/start