Machine Learning Engineer Nanodegree

Capstone Project

Ken Adachi
September 12th, 2019

I. Definition

Project Overview

In this project, I want to apply Machine Learning technique for the purpose of crime prevention.

There are some academic papers or articles on this domain such as:

Learning, Predicting and Planning against Crime:Demonstration Based on Real Urban Crime
Data https://pdfs.semanticscholar.org/cacd/e031e470af4fe835bf50f14eb4c265e0f2a6.pdf

USING MACHINE LEARNING ALGORITHMS TO ANALYZE CRIME
DATA https://www.researchgate.net/publication/275220711_Using_Machine_Learning_Algorithms_to_Analyze_Crime_Data

AI for Crime Prevention and Detection – 5 Current
Applications https://emerj.com/ai-sector-overviews/ai-crime-prevention-5-current-applications/

I want to take Crime Opportunity Theory as a basis for this project.This theory suggests that the occurence of a crime depens not only on the presense of the

motivated offender but also on the conditions of the environment in which that offender is situated.

Reference: Community Safety Maps for Children in Japan: An Analysis from a Situational Crime Prevention
Perspective https://link.springer.com/article/10.1007/s11417-011-9113-z

I except that I can find a pattern in where actual crime happened. So I want to analyze the histrocial crime incident data with the information on where it occured.

This might lead to prevent crime to happen in my neighborhood and proctec children in my community.

I'm going to use "Police Department Incident Reports: Historical 2003 to May 2018" data set provided by San Francisco City Government.

https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry

This dataset includes police incident reports filed by officers and by individuals through self-service online reporting for non-emergency cases through 2003 to 2018.The dataset has attributes such as when the incident reports filed (Date, time) and detail location of the incidents(latitude, longitude).

Problem Statement

In order to avoid to be involved in encountering crime, I want to develop a solution to predict if the crime occur in the present location. There are several points to consider:

- Location

Crime Opportunity theory suggests that the "view" of the location is important for the ones who try to commit a crime to decide to do so.For example, if there is a street with many tall trees which makes it hard for everyone to be seen, there is a higher chance of the crime to be occured.

So if I can specify the "location" as a street level granurality , then that would precise. Or if I can use street view image of the location leveraging map data such as google street view, then the result would be interesting.

However, given limitted resources and time, I'm thinking of specifying location as neigborhood in a way such as "the crime actually happend in within a few miles radius from the current point".

- Types of Crime

I would not classify what types of crimes occured or will occure. The main purpose of this project is to identify the location that makes criminals to think the place is a good opportunity for them to commit a crime. So, I just want to focus on analyizing the crime occured in that place or not.

- Timing of the crime

It is expected that the crime might occur on specific timing. For example, crime targeted for the children will occur more when children got home from school.

In sum, I want to make the solution to accept the location and time and then predict if the crime will occur or not.

Metrics

This is a classification problem. However, after conducting data exploration, I realised that the label dataset is not closely balanced. That's wahy I would not use F measures as metrics and instead I'm going to use log loss. Please see the detail on Section II. Analysis.

II. Analysis

Data Exploration

There are 33 features in this dataset with about 2 million data rows. There is no description on the original dataset about the feature ranging from 'SF Find Neighborhoods' to ':@computed_region_2dwj_jsy4'. So I will omit those features and focus on following featuers:

```
crime_df =
pd.read_csv('Police_Department_Incident_Reports__Historical_2003_to_May_2018.
csv')
crime_df.columns.values
array(['IncidntNum', 'Category', 'Descript', 'DayOfWeek', 'Date', 'Time',
       'PdDistrict', 'Resolution', 'Address', 'X', 'Y', 'Location', 'PdId',
       'SF Find Neighborhoods', 'Current Police Districts',
       'Current Supervisor Districts', 'Analysis Neighborhoods',
       ':@computed_region_yftq_j783', ':@computed_region_p5aj_wyqh',
       ':@computed_region_rxqg_mtj9', ':@computed_region_bh8s_q3mv',
       ':@computed_region_fyvs_ahh9', ':@computed_region_9dfj_4gjx',
       ':@computed_region_n4xg_c4py', ':@computed_region_4isq_27mq',
       ':@computed_region_fcz8_est8', ':@computed_region_pigm_ib2e',
       ':@computed_region_9jxd_iqea', ':@computed_region_6pnf_4xz7',
       ':@computed_region_6ezc_tdp2', ':@computed_region_h4ep_8xdi',
       ':@computed_region_nqbw_i6c3', ':@computed_region_2dwj_jsy4'],
dtype=object)
crime_df.shape
(2215024, 33)
```

There are 33 features in this dataset with about 2 million data rows. There is no description on the original dataset about the feature ranging from 'SF Find Neighborhoods' to ':@computed_region_2dwj_jsy4'. So I will omit those features and focus on following featuers:

'IncidntNum': Unique key value on each incident.

'Category':VEHICLE THEFT,NON-CRIMINAL etc.

'Descript':STOLEN MOTORCYCLE,PAROLE VIOLATION etc

'DayOfWeek':'Monday', 'Tuesday'...

'Date':DD/MM/YYYY

'Time':HH:mm

'PdDistrict':SOUTHERN,MISSION etc

'Resolution':ARREST,BOOKED etc

'Address':Street name of crime such as Block of TEHAMA ST

'X':Longitude

'Y':Latitude

'Location':Concat of X and Y

'PdId':Unique Identifier for use in update and insert operations

Category data is not well balanced.

```
crime_df['Category'].value_counts()
LARCENY/THEFT                 480448
OTHER OFFENSES                309358
NON-CRIMINAL                  238323
ASSAULT                       194694
VEHICLE THEFT                 126602
DRUG/NARCOTIC                 119628
VANDALISM                     116059
WARRANTS                      101379
BURGLARY                       91543
SUSPICIOUS OCC                 80444
MISSING PERSON                 64961
ROBBERY                        55867
FRAUD                          41542
SECONDARY CODES                25831
FORGERY/COUNTERFEITING         23050
WEAPON LAWS                    22234
TRESPASS                       19449
PROSTITUTION                   16701
STOLEN PROPERTY                11891
SEX OFFENSES, FORCIBLE         11742
```

```
DISORDERLY CONDUCT             10040
DRUNKENNESS                     9826
RECOVERED VEHICLE               8716
DRIVING UNDER THE INFLUENCE     5672
KIDNAPPING                      5346
RUNAWAY                         4440
LIQUOR LAWS                     4083
ARSON                           3931
EMBEZZLEMENT                    2988
LOITERING                       2430
SUICIDE                         1292
FAMILY OFFENSES                 1183
BAD CHECKS                       925
BRIBERY                          813
EXTORTION                        741
SEX OFFENSES, NON FORCIBLE       431
GAMBLING                         348
PORNOGRAPHY/OBSCENE MAT           59
TREA                              14
```

Exploratory Visualization

- Location Let's visualize the occurence of the crime by the location.

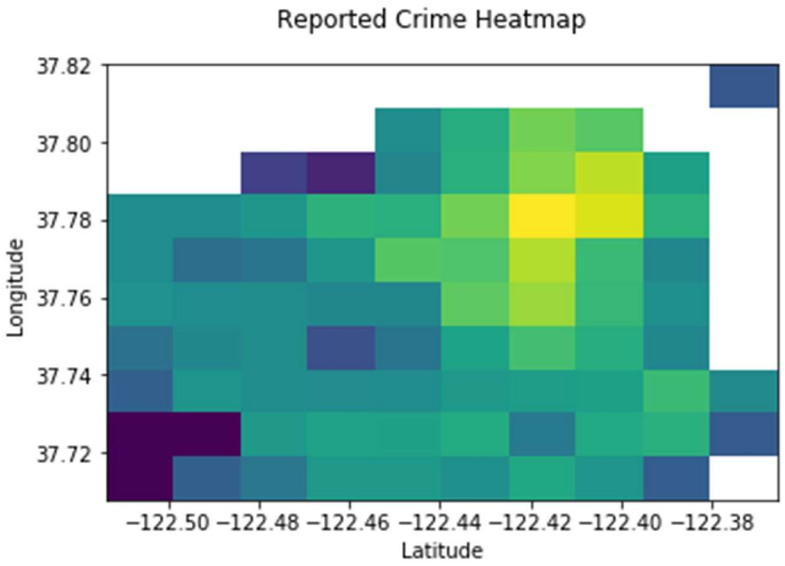
```
rounding_factor = 4

from matplotlib.colors import LogNorm
x = np.round(crime_df['X'].head(10000),rounding_factor)
y = np.round(crime_df['Y'].head(10000),rounding_factor)
fig = plt.figure()
plt.suptitle('Reported Crime Heatmap')
plt.xlabel('Latitude')
plt.ylabel('Longitude')
H, xedges, yedges, img = plt.hist2d(x, y, norm=LogNorm())
extent = [yedges[0], yedges[-1], xedges[0], xedges[-1]]
```

```
plt.show()
```

Fig1 shows the heatmap. I can see that there are certain locations that no crimes occur.

Fig1

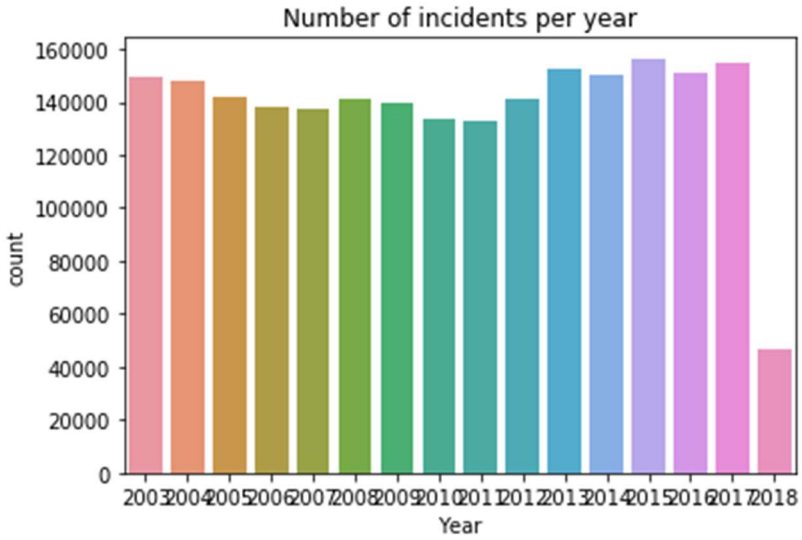


- Occurence by Year

```
crime_df['Year'] = [int(dte.split("/")[2]) for dte in crime_df['Date']]
sns.countplot(x='Year',data=crime_df)
```

Fig2 shows the case number by Year. There are less cases in 2018 compared to another year. I assume this is because the dataset is still under updated when I get this one.

Fig2

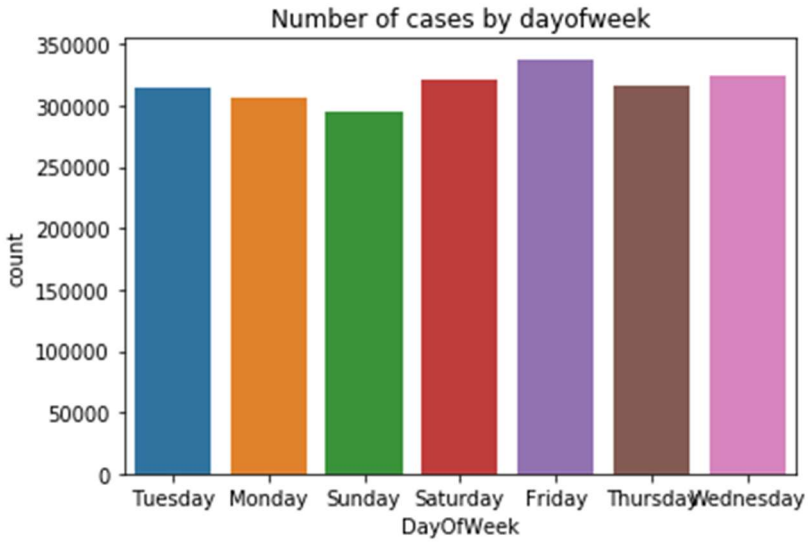


- Occurence by dayofweek

```
sns.countplot(x='DayOfWeek',data=crime_df)
plt.title('Number of cases by dayofweek')
```

Fig3 shows the case number by day of work.It looks like much more cases are recorded on Weekend.

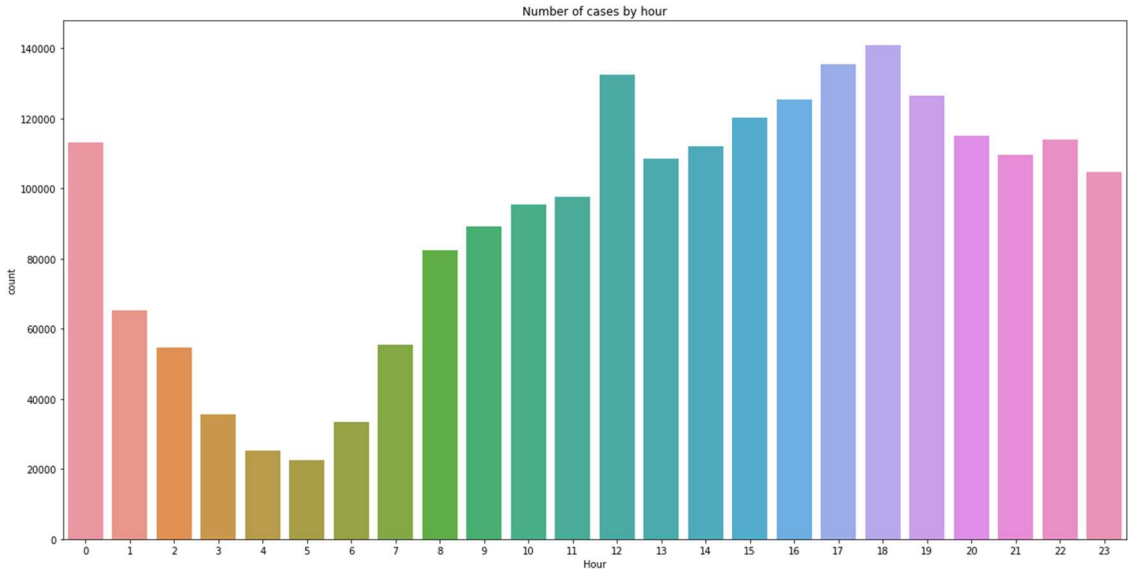
Fig3



- Occurence by Hour

Fig4 shows the case number by hour. It shows the incident happens more on daytime thourgh midnight.

Fig4



Algorithms and Techniques

I want to build prediction model leveraging machine learning techniques.

In this case, I want to apply surpervised learning becuase the labeled data is available.

Based on sklearn algorithm cheat-sheet, I'm going to use classification methods to build the model.

https://scikit-learn.org/stable/tutorial/machine_learning_map/

I'm going to use ensemble model leveraging XGBoost.

Benchmark

I will use very simple model such as logistic regression as a benchmark model.

III. Methodology

Based on the result of EDA above, I will use DayOfWeek, Time, Location(Longitute,Latitude) as predicators.

Data Preprocessing

- Year There are over 2 million records in a dataset. This is too many for my machine does not enough power to learn all those datasets. My interest is on the scenary of the place of the crime. Landscape changes with the time passes due to redevelopment etc. So I will focus on the latest data and use the data from 2015 to 2018.

```
crime_2015_2018 = crime_df[crime_df.Year > 2014]
crime_2015_2018 = crime_2015_2018[crime_df.Year < 2019]
crime_2015_2018.shape
(508850, 34)
```

- Location I try to scale the longitude and latitude by using standard scaler.

```
xy_scaler = preprocessing.StandardScaler()

xy_scaler.fit(crime_df[["X","Y"]])

crime_df[["X","Y"]]=xy_scaler.transform(crime_df[["X","Y"]])
```

- Category Data set has a text field of 'Category'.This is a non-numerical feature, so I'm going to encode this to numerical value using pandas get_dummies() or LabelEncoder.

```
le_crime = preprocessing.LabelEncoder()
crime = le_crime.fit_transform(crime_df.Category)
crime_df['Category'] = crime
```

- DayOfWeek This feature too is a text field.I encode this as well.

```
crime_df['DayOfWeek'] = le_crime.fit_transform(crime_df.DayOfWeek)
```

Implementation

I build the initial model with 'DayOfWeek', 'Time', 'X', 'Y' as predicting features and 'Category' as label.

I split the data into training and testing for cross_validation.

```
training, testing =
cross_validation.train_test_split(crime_2015_2017,test_size = 0.2,
random_state=0)
#training = training[['Category', 'DayOfWeek', 'Date', 'Time', 'X', 'Y']]
training = training[['Category', 'DayOfWeek',  'Time', 'X', 'Y']]
# Rename X,Y to Longitude, Latitude
training.columns = ['Category', 'DayOfWeek',  'Time', 'Longitude',
'Latitude']
label = training['Category'].astype('category')

testlabel = testing['Category'].astype('category')

del training['Category']

del testing['Category']
```

Firstly build model by logistic regression for benchmarking.

```
lr = LogisticRegression()
lr.fit(training,label)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
lrpredicted = lr.predict_proba(testing)
log_loss(testlabel,lrpredicted)
2.4779045272356286
```

Next , I build initial model with XGBClassifier.

```
xgb_model = XGBClassifier()
xgb_model.fit(training,label)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='multi:softprob', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
predicted = xgb_model.predict_proba(testing)
log_loss(testlabel,predicted)
2.3437577077398806
```

Log loss of XGB is only 5% better than benchmark.

Refinement

I tried to search better set of hyperparameters using GridSearchCV. For example, I tried different sets of parameters like following:

```
params={
       'max_depth':[4,10,13],
       'min_child_weight':list(range(1,3,1)),
       'gamma':[0.5,1,2]
}
grid_search = GridSearchCV(xgb_model,
                           param_grid = param_distNn,
                           neg_log_loss': make_scorer(log_loss,
labels=crime_classes, greater_is_better=False,needs_proba=True)},
                 n_jobs=1,
                 refit='neg_log_loss',
                           verbose=10)
```

After doing several time of search, I optimized parameters and final model was following:

```
xgb_model = XGBClassifier(
                    max_depth = 3,
```

```
                    min_child_weight=1,

                    gamma = 1,
)
```

IV. Results

Model Evaluation and Validation

The final model is chosen after conducting parameter tuning. Following is a final XGBoost classifier definition.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='multi:softprob', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
```

To verify this model, validation test was conducted by spliting the datasets into test datasets.

```
xgb_model.fit(training,label)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='multi:softprob', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
predicted = xgb_model.predict_proba(testing)
log_loss(testlabel,predicted)
2.3498548077681436
```

The metrics of log_loss only improves the by 0.1% from initial model. So it's not a great improvement.

Justification

Using XGBoost model, I built the prediction model. To clarify the performance of this model, I visualized how the training is conducted in Fig5.

V. Conclusion

Free-Form Visualization

As stated above, final model performance can be assessed by the log loss value. Final model above is only 5% better than the bench mark which was not satisfactionally.

Fig5 shows how the training went. As the number of training improves, log loss value is declined, which shows that it successfully is trained and validated for test datasets as well.

Fig5

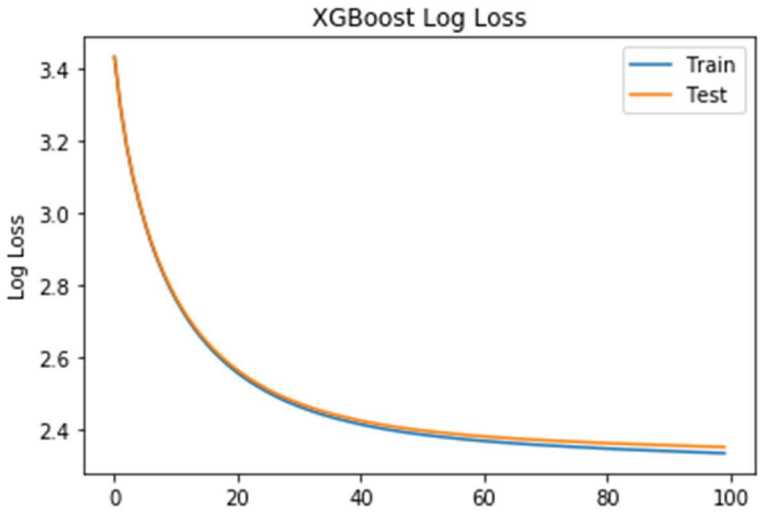
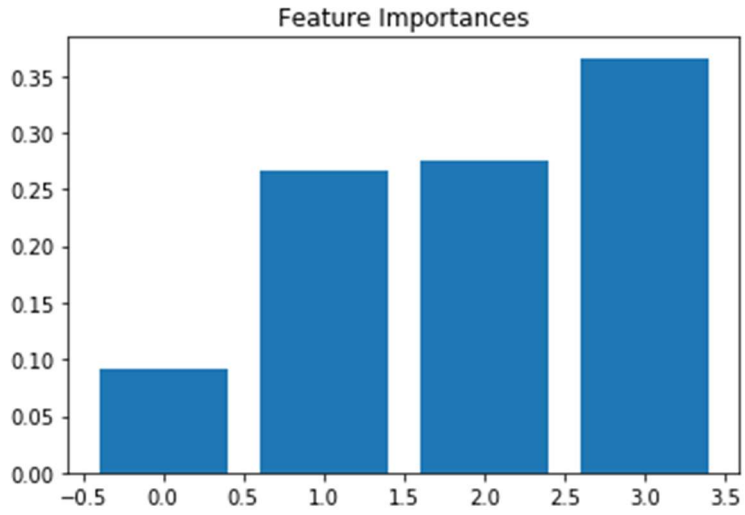


Fig6 shows the importance of each feature in this model. Time,Locations has much more importance than dayofweek.

Fig6



Reflection

The difficult part of this project was hyperparameter tuning. I decided to use GridSearchCV. The challenge was the time it took to optimize the parameters. It took approximately 15min to 20min for each fits. So I could not devote much time to conduct many attempts thus got a very limited result of the tuning.

I should have spent more time on changing parameters of GridSearchCV to reduce the time of tuning.

Improvement

There is a discussion in kaggle competeition to create features based on longitude and latitude. https://www.kaggle.com/c/sf-crime/discussion/18853#latest-413648 I'm not quite sure what this means actually and hesitated to adopt the idea in the timeframce of this project.

But I want to look into above thread and do more research as well as do more parameter tuning to improve the model performance.