San Francisco Crime Classification

Submitted By:

Shyam Kumar V N (MS2018017)
Bosco Sebstian (PH2018007)
Arun Kumar S (PH2018005)

Machine Learning
Gen 511

December 6, 2018

Contents

0.1 Introduction 2
0.2 Objective 2
0.3 Our Dataset 2
0.3.1 Features 2
0.3.2 Preprocessing 3
0.3.3 Feature Enrichment 3
0.4 Software and technologies 3
0.5 Visualizing San Francisco Crime Data 4
0.6 Preliminary Findings 14
0.7 Model Selection / Methods 14
0.7.1 Confusion Matrix 14
0.7.2 Logistic Regression 16
0.7.3 Principal Component Analysis 16
0.7.4 XGBClassifier 16
0.7.5 Bayesian Classifier 16
0.7.6 Random Forests 16
0.7.7 Naive Bayes 17
0.8 Results 17
0.9 Conclusions 18
0.10 References 18

0.1 Introduction

There is a huge amount of crimes reported every day in the city of San Francisco. The San Francisco Police Department (SFPD) has captured all of these reports within their systems and has made the data publicly available in a Kaggle competition. The challenge is to "predict the category of crimes that occurred" based on information the information in their reports. This competition was taken as baseline with modified data set provided by IIITB ML Team.

0.2 Objective

Our goal is to predict the probability that a crime belongs to certain category based on its time and location. Because the label we want to predict is categorical, this task is a classification problem. Further, the label we are predicting has more than two labels. Hence, this is a multi-class classification problem.

0.3 Our Dataset

Our data-set is a provided by IIITB ML team and is available at https://www.kaggle.com/c/IIITB-ML-Project-sfo-crime-classification/data in Kaggle, which has information about 868k crimes that took place in San Francisco city over a span of nearly twelve years.

0.3.1 Features

Every entry in our training data set is about a particular crime, and contains the following information:

- Date and timestamp of the incident.
- Day of the week that the crime occurred.
- Name of the Police Department District.
- Address: the approximate street address of the crime incident.
- Latitude.
- Longitude.
- Category: category of the crime incident. This is the target variable.
- Description: a brief note describing any pertinent details of the crime. (This was not used as a feature in our classifiers.)
- Resolution: whether the crime was resolved (with the perpetrator being, say, arrested or booked) or not. (This was also not used as a feature in our classifiers.)

TABLE : Few sample rows from our data-set

Dates	Category	Descript	DayOfWeek	PdDistrict
13-05-2015 23:53	WARRANTS	WARRANTS	Wednesday	NORTHERN
13-05-2015 23:53	OFFENSES	TRAFFIC VIOLATION	Wednesday	NORTHERN
13-05-2015 23:33	OFFENSES	TRAFFIC VIOLATION	Wednesday	NORTHERN
13-05-2015 23:30	LARCENY	GRAND THEFT	Wednesday	NORTHERN

0.3.2 Preprocessing

Before implementing machine learning algorithms on our data, we went through a series of preprocessing steps with our classification task in mind. These included:

- Dropping features such as Resolution, Description: The resolution and description of a crime are only known once the crime has occurred, and have limited significance in a practical, real-world scenario where one is trying to predict what kind of crime has occurred, and so, these were omitted.
- The days of the week, police categories and crime categories were indexed and replaced by number.
- The time-stamp contained the year, date and time of occurrence of each crime. This was decomposed into five features: Year (2003-2015), Month (1-12), Date (1-31), Hour (0-23) and Minute (0-59).

0.3.3 Feature Enrichment

As we plunged into solving our classification problem, we felt that our feature set was not adequate enough in terms of the information it contained to predict crime.

- So we first added Time Based Features. The train.data date fields were split into Morning, Noon, Evening and Night.
- Further we split the data into Monsoon categories like Fall, Winter, Spring and Summer based on month. This was done for both Train and Test data set available to us.
- We tried our Training model without adding Address features and were ranked initially about 14 in the leader board score. We tried to identify further features and noticed that we could explore more on Address field available to us. We extracted features from Address Field including Street-No, Intersection and Address separately for both Train and Test Data.
- We started training our data by reading the csv file and dropping Description, Resolution and Category features. We did encoder.transform based on the labels and called our time and address features and obtained Final Training Features and in a similar way for Test Features.

0.4 Software and technologies

The software used to carried out this project was Python version 3.7 with important libraries being Pandas, NumPy , sklearn and matplotlib .

- Pandas, is an open source library in Python that provides a highly optimized manipulation and operations in data structures. Form this library, we mainly used the DataFrame structure for creating and manipulating both training and testing data-sets as well as for creating the submission file.

- NumPy, is a fundamental package in Python for N-dimensional array object, along with a large library of high-level mathematical functions to operate on these arrays. We used this library in order to efficiently create and manage arrays such as the grid cells, the probabilities of the crimes in the cells and the submission file.

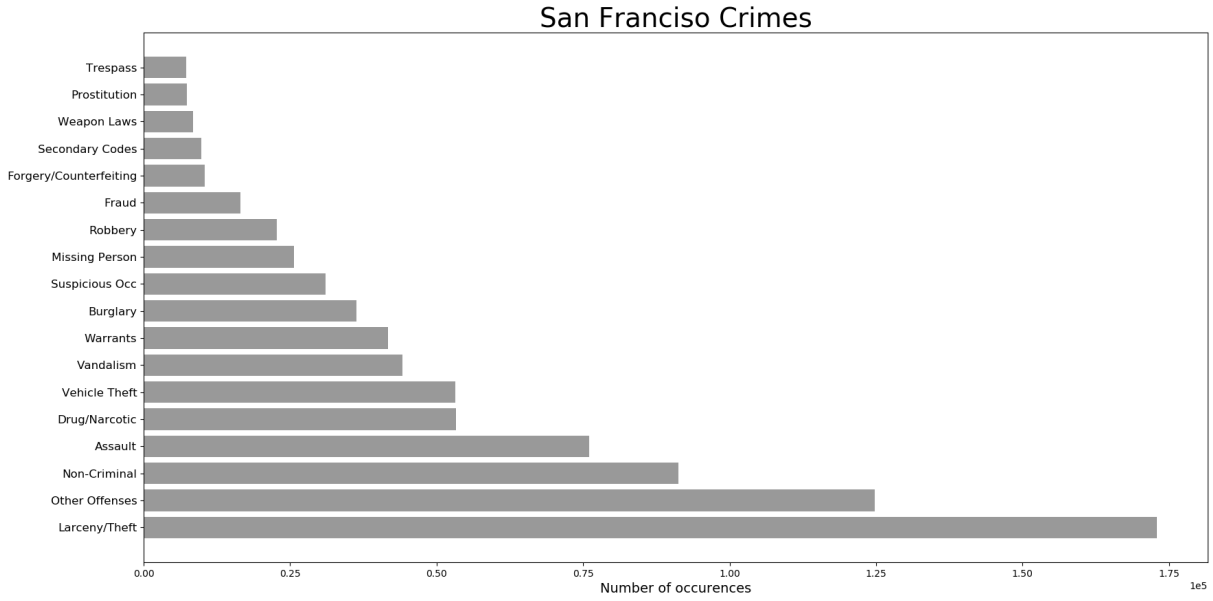
- sklearn (Scikit-Learn), is an open source machine learning library for Python. It does not only contains various classification, regression and clustering algorithms but also preprocessing techniques such as standardization, scaling, normalization, binarization, encoding and others,

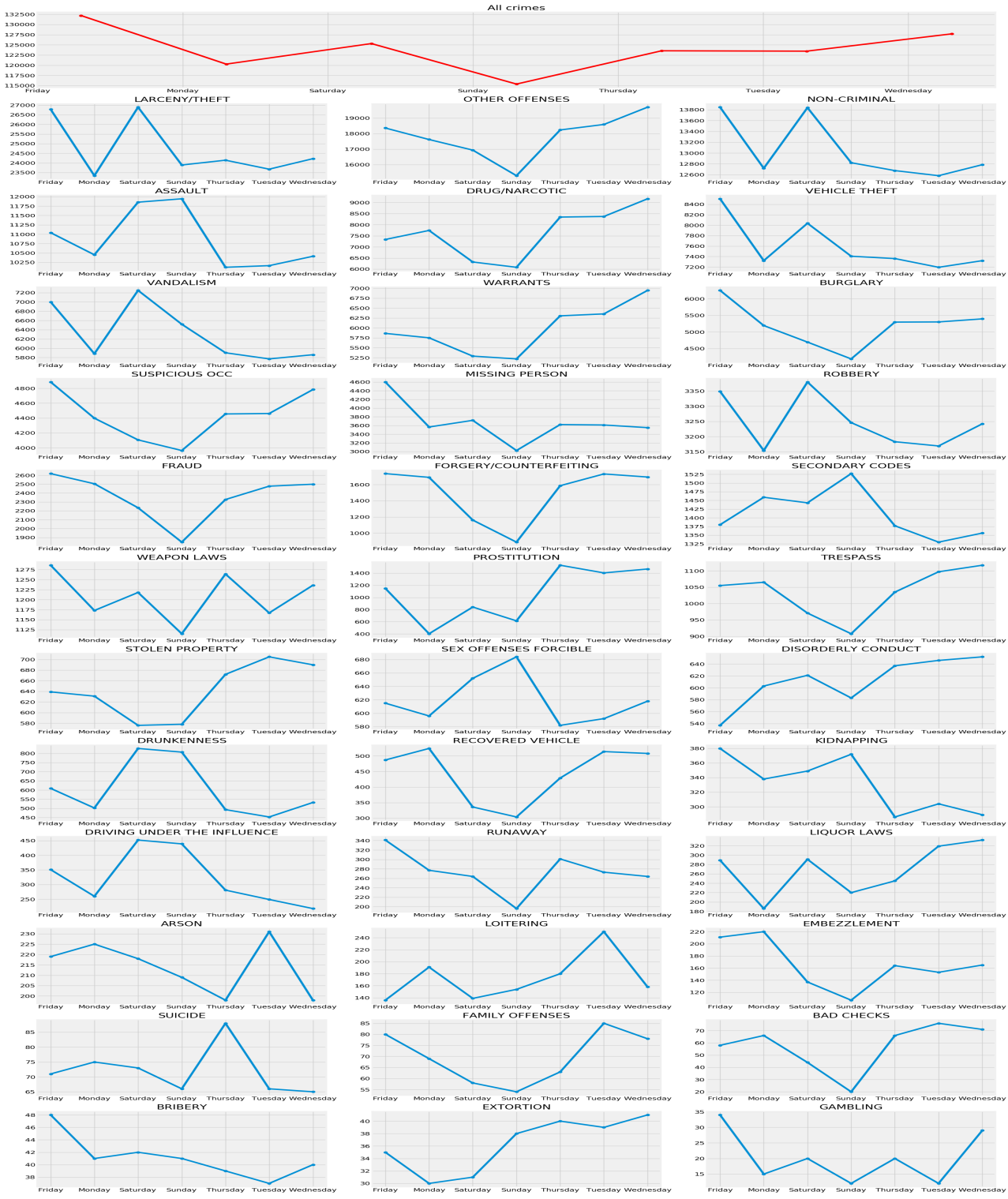
- matplotlib, is a python 2D plotting library and its numerical mathematics extension NumPy.

We used this library for implementing scaling, PCA, Logistic regression, Naive Bayes Classifier, Random forest and log-loss.

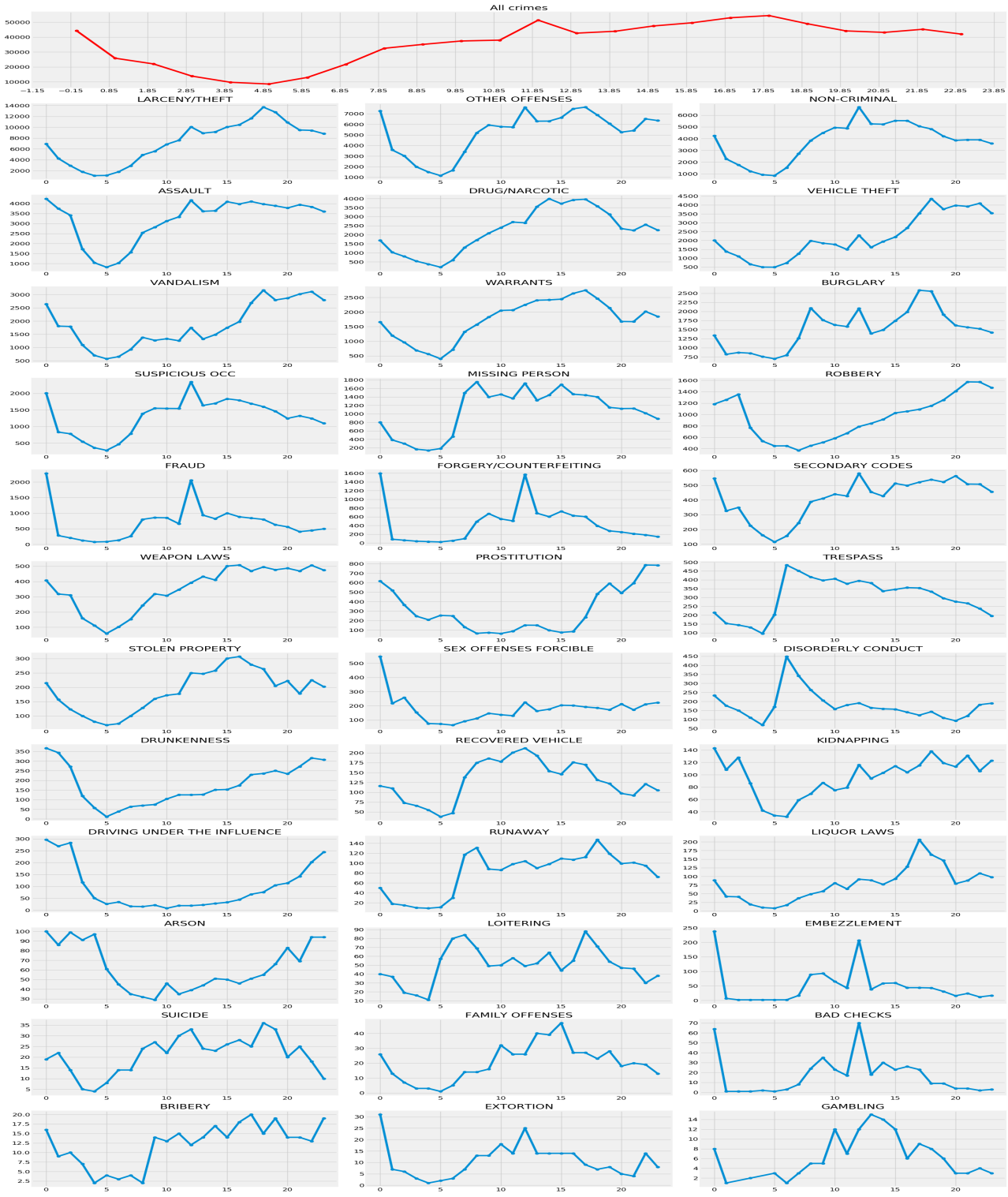
0.5 Visualizing San Francisco Crime Data

Which crime was most prominent ? Below graph clearly explains us that Larceny/Theft, Other offenses and non- criminal are the top 3 most criminal incidents, which will help us in making feature selection decisions.

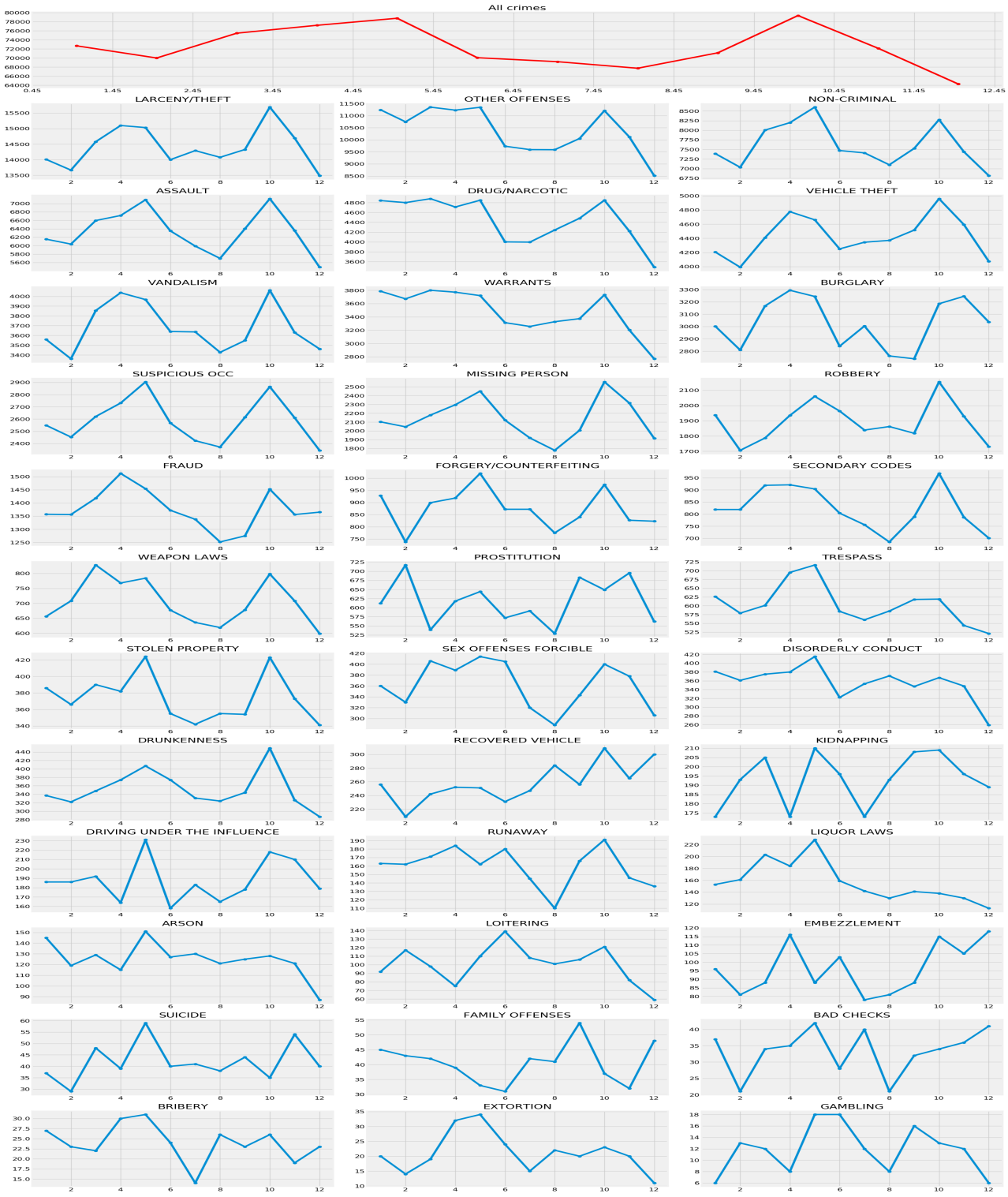




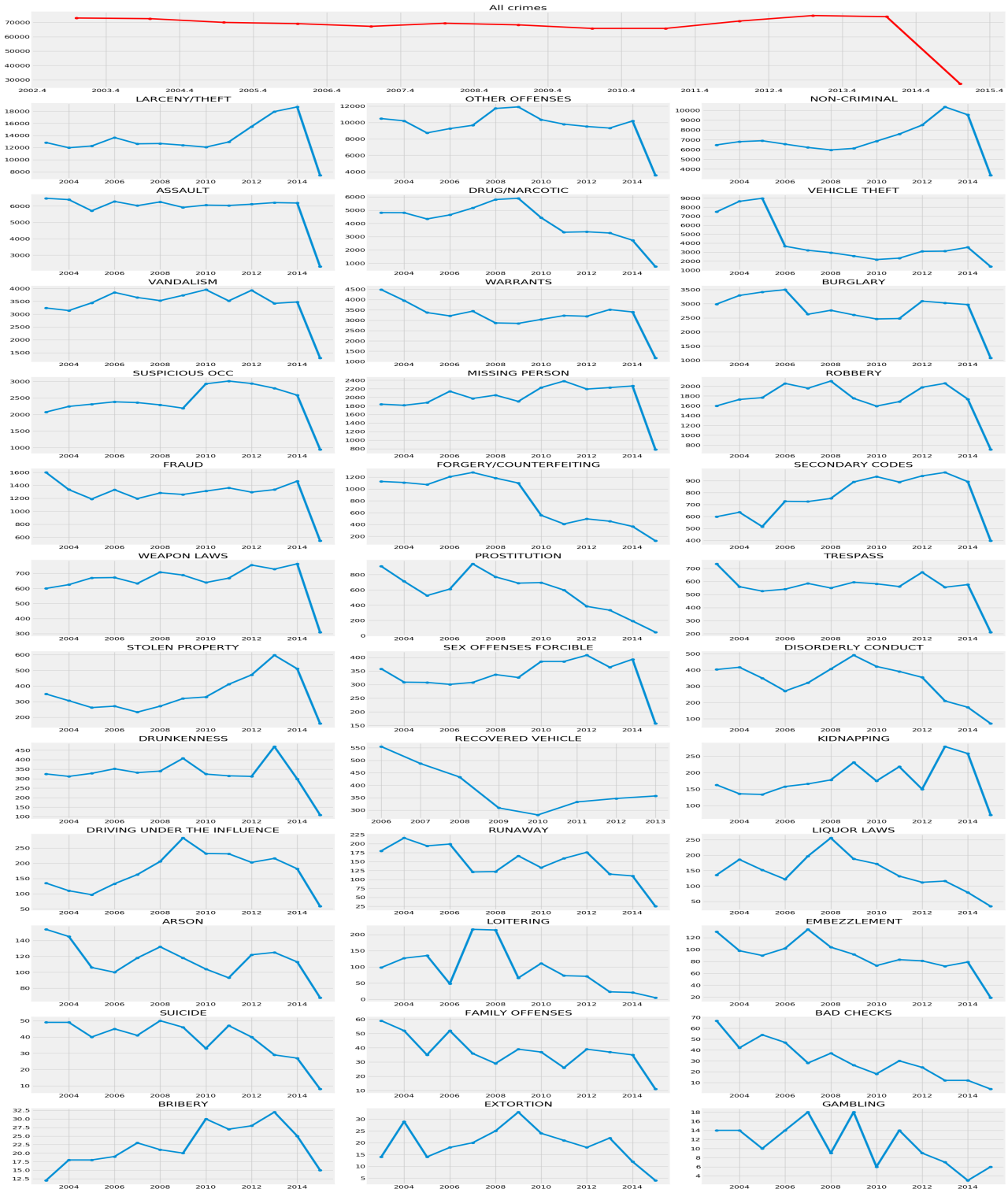
When did the crime was most prominent ? From the chart its pretty evident that the crimes were less on weekends and high on Friday and Wednesday.



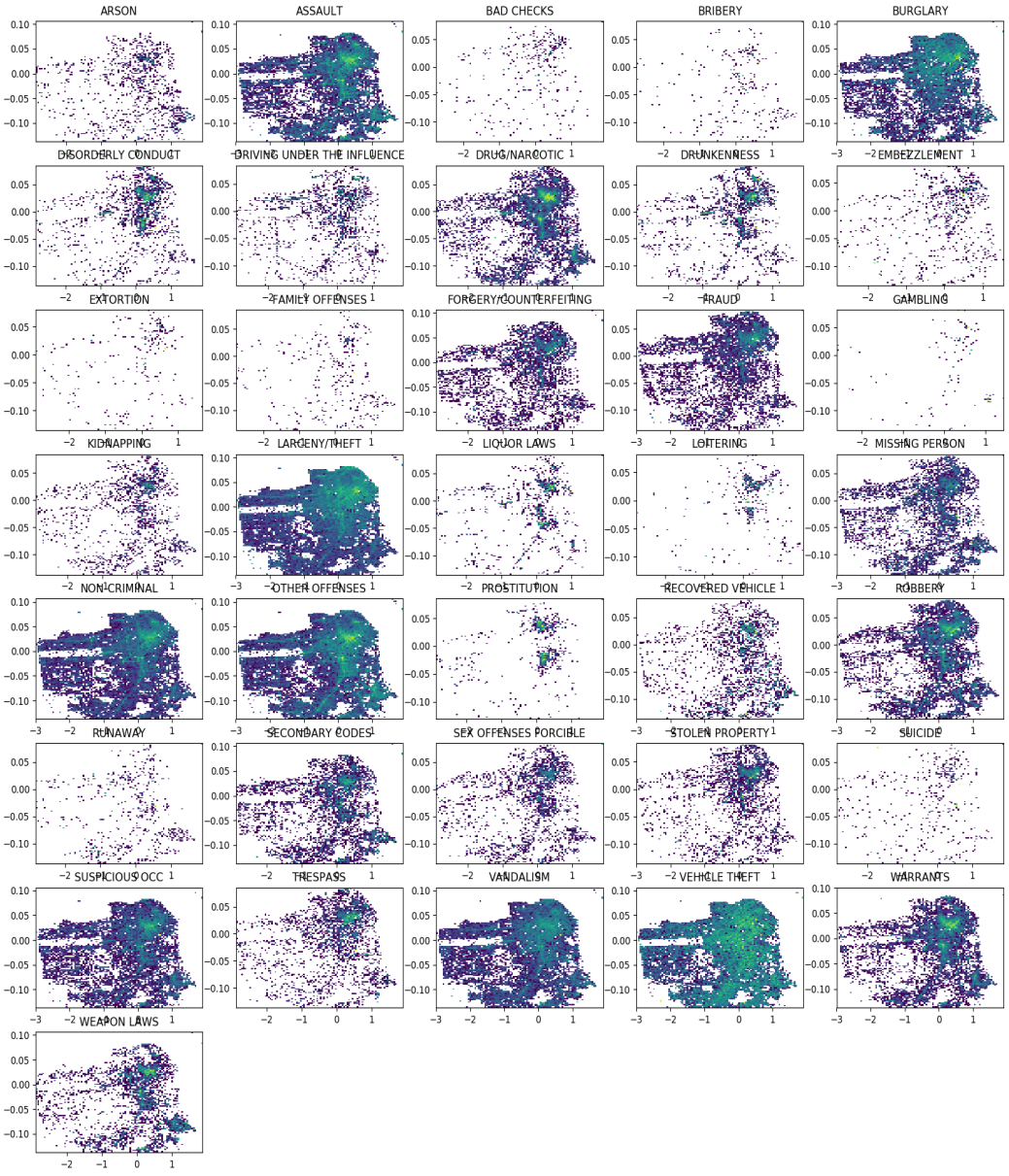
Graph explains the crime chart based on every hour



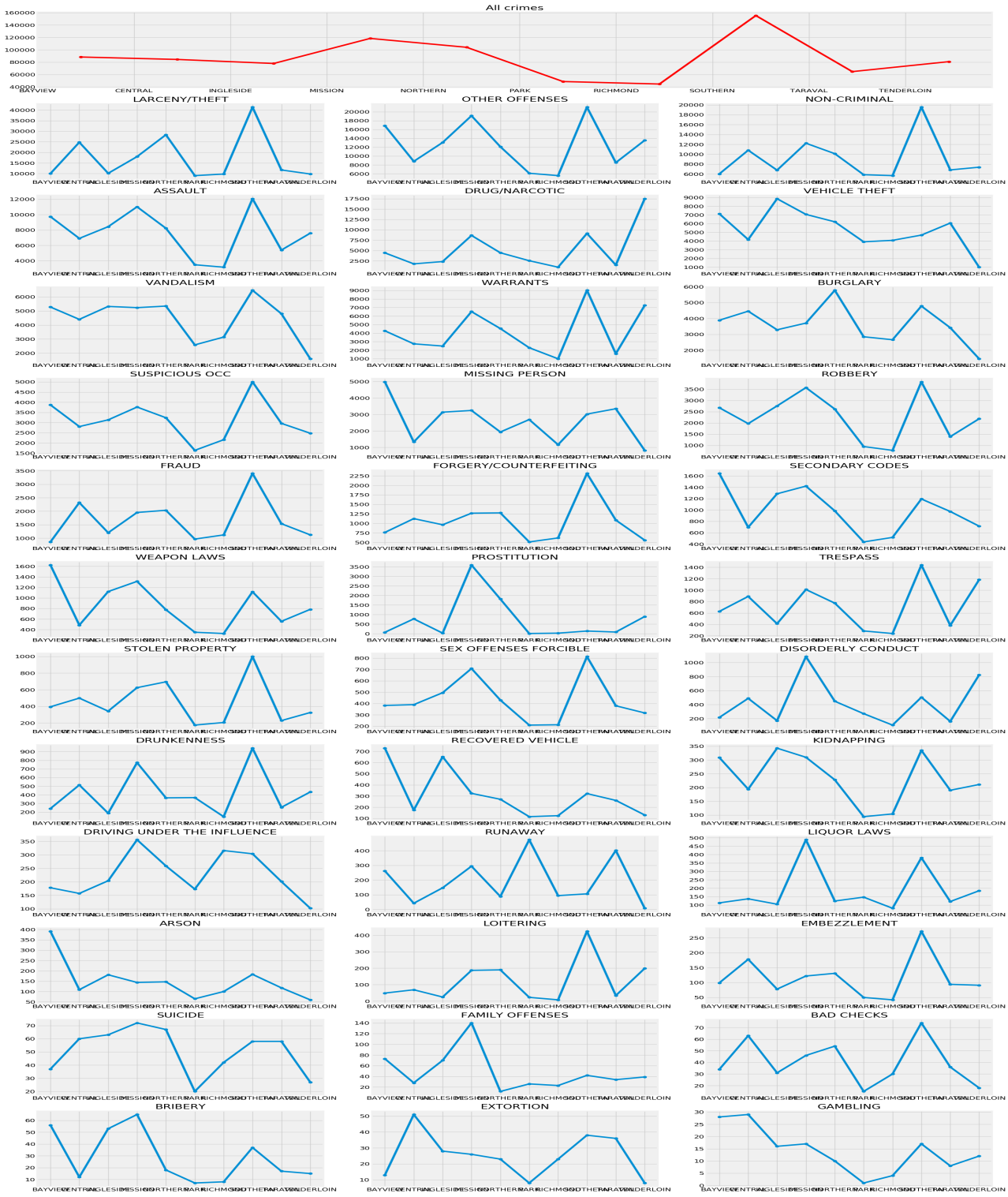
Graph explains the crime chart based on every Month



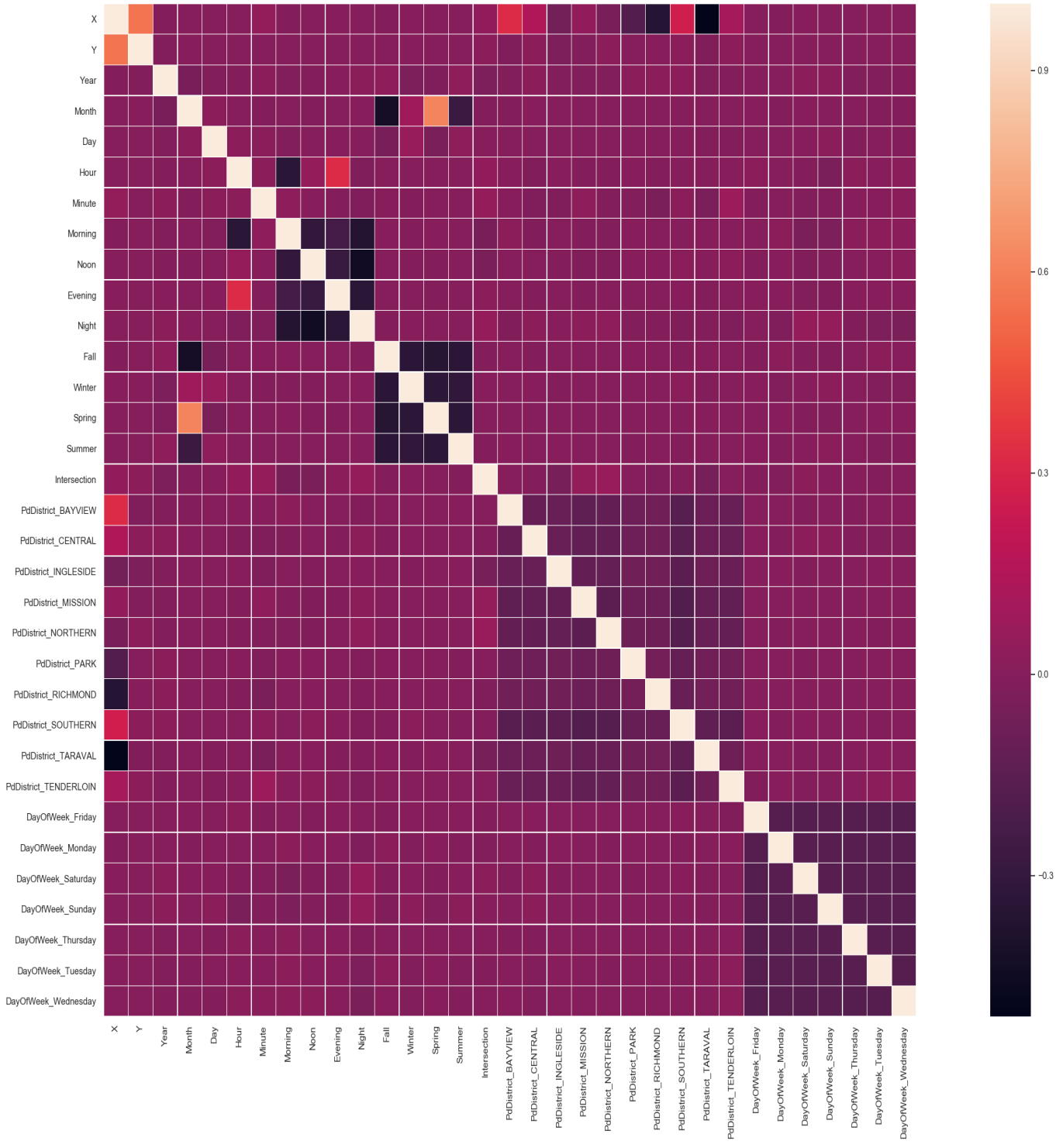
Graph explains the crime chart based on every Year



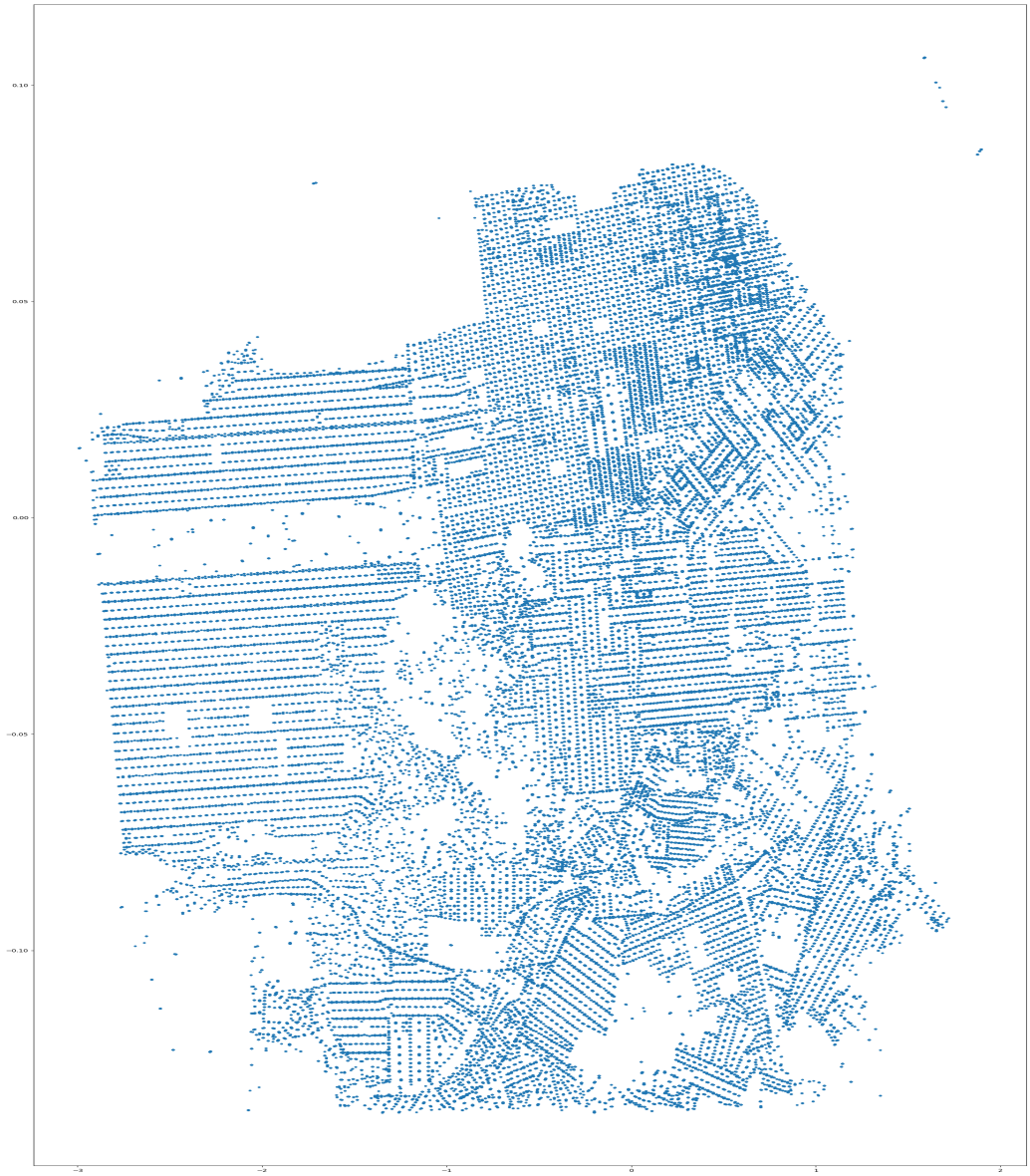
Graph explains the crime chart based on Location



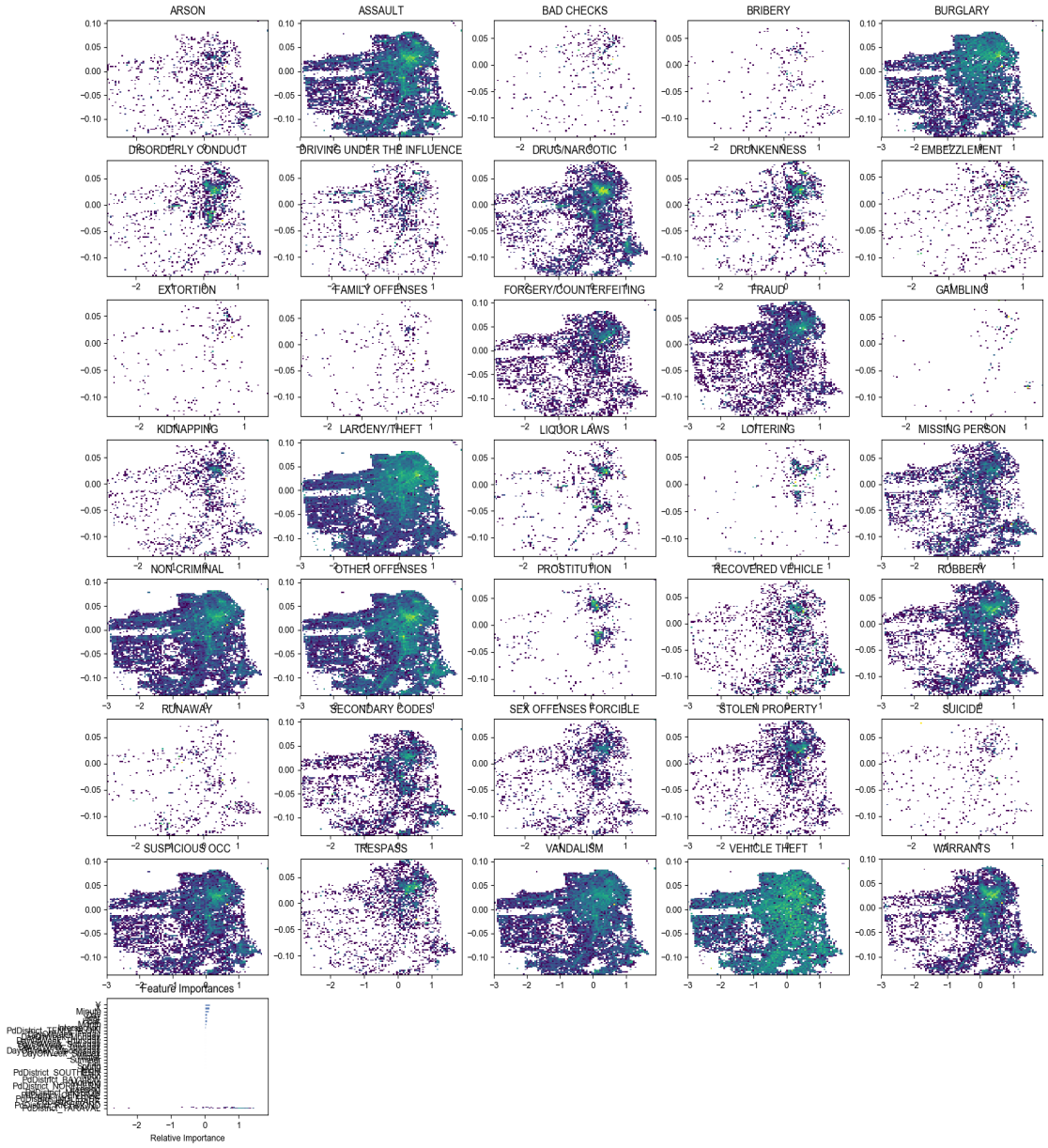
Graph explains the crime chart based on PdDistrict



Graph explains the Feature Correlation



The SF Police Department has recently divided the city into 10 districts in order to manage incidents more efficiently. Heres the district map. SFO Location Map



RF Variable Importance

0.6 Preliminary Findings

As a start, we explored the data to figure out some of the peculiarities of this problem. And our preliminary findings then shaped the approaches we decided to take.

1. After plotting the data based on their location coordinates, we realized that about 67 points had latitudes of 90 degrees, meaning they took place at the North Pole. These points were heavy outliers and were excluded them from our training set.

2. The data was heavily skewed in terms of the frequency of each category. Only 19 of the 39 classes had proportions of the training set in amounts above .5 percent. Since we were seeking to minimize the log-loss, we decided our efforts focus overwhelmingly on doing well on distinguishing between these major classes, since errors on the remaining classes would contribute little to the error.

3. Many of our intuitive guesses were correct. For instance, certain classes of crime were much more frequent on weekends (and vice versa). Many crimes had different frequencies depending on the time of day. As such, we found it appropriate, when designing our models, to use feature mappings that take these intricacies into account. Namely, we mapped the time stamps into a much higher dimensional feature space, i.e.month, weekday, time of day, etc.

4. We did a lit review of findings based on this data set. Of the thousands of posts about it, most involved finding interesting ways of visualizing the data, and divulged little information about approaches to the actual classification problem.

5. The training set contained over 8 million reports, but each report had a relatively small amount of information. We decided to disregard most of the textual columns such as PdDistrict and Descript and Resolution (as asked). Thus we were left with Dates, the coordinates, address and the labels. This problem then became an issue of predicting between a large amount of classes on a large set of data points, based on a relatively small amount of features.

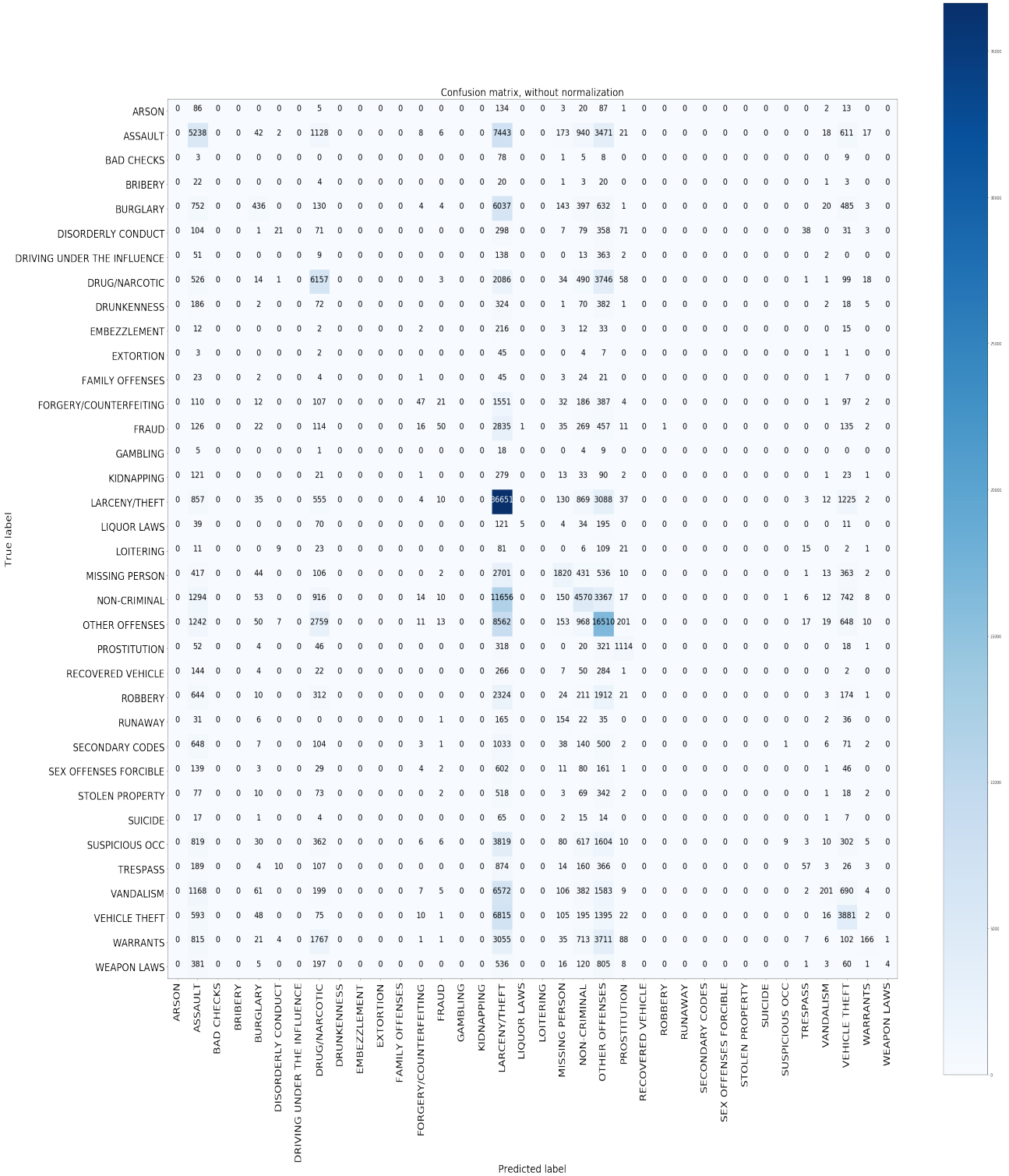
6. Given that we had plenty of data with a small amount of features, we figured the most difficult part of this problem would, in fact, be in finding the most appropriate model for this application.

0.7 Model Selection / Methods

In order to increase the classification accuracy, given we did lot of Pre-processing, Data Cleaning and Feature Engineering we decided to try multiple classification algorithms. In this section, we will explore different classification models, and compare their accuracy. We did model optimization after we had finished validations towards different feature combinations and had settled one combination which yielded the best result.

0.7.1 Confusion Matrix

Below is the Confusion Matrix. Since the data is heavily skewed we couldn't make much out of the Confusion Matrix. We can do better here for sure and an area of improvement in our project.


Confusion matrix, without normalization
True label
Predicted label
ARSON
ASSAULT
BAD CHECKS
BRIBERY
BURGLARY
DISORDERLY CONDUCT
DRIVING UNDER THE INFLUENCE
DRUG/NARCOTIC
DRUNKENNESS
EMBEZZLEMENT
EXTORTION
FAMILY OFFENSES
FORGERY/COUNTERFEITING
FRAUD
GAMBLING
KIDNAPPING
LARCENY/THEFT
LIQUOR LAWS
LOITERING
MISSING PERSON
NON-CRIMINAL
OTHER OFFENSES
PROSTITUTION
RECOVERED VEHICLE
ROBBERY
RUNAWAY
SECONDARY CODES
SEX OFFENSES FORCIBLE
STOLEN PROPERTY
SUICIDE
SUSPICIOUS OCC
TRESPASS
VANDALISM
VEHICLE THEFT
WARRANTS
WEAPON LAWS

0.7.2 Logistic Regression

For logistic regression, we changed the value of the regularizer, computed log loss separately, and located the sweat point between under-fitting and over-fitting. Though the score we got was 2.89756 which was not enough for us to lead the Kaggle competition.

0.7.3 Principal Component Analysis

In order to increase the classification accuracy, and avoid over-fitting, we used PCA to reduce the dimensionality. We noticed that the PCA performance was not giving us the desired log-loss and we were scoring around 3.01891.

0.7.4 XGBClassifier

For XGB, parameters we have tuned are n_estimators and tried to tune against Random Forest. n_estimators is the number of trees to build before taking the maximum voting or averages of predictions. The model is under-fitting when that value is too small and over-fitting when its too large. When comparing with Random Forest for the tree tuning we got a score 3.45685 that's very low with models in our hand.

0.7.5 Bayesian Classifier

This classifier didnt prove to be the best for this set of features. It sets a baseline of 2.84923294 on the validation set, which was much lesser than the Random Forest results we got through tuning.

0.7.6 Random Forests

Random Forests is a very popular ensemble learning method which builds a number of classifiers on the training data and combines all their outputs to make the best predictions on the test data. Thus, the Random Forests algorithm is a variance minimizing algorithm that uses randomness when making split decision to help avoid over-fitting on the training data. A random forests classifier is an ensemble classifier, which aggregates a family of classifiers h(x—1), h(x—2), ..h(x—k). Each member of the family, h(x—), is a classification tree and k is the number of trees chosen from a model random vector. Also, each k is a randomly chosen parameter vector. If D(x, y) denotes the training data-set, each classification tree in the ensemble is built using a different subset Dk (x, y) D(x, y) of the training data-set. Thus, h(x—k) is the kth classification tree which uses a subset of features xk x to build a classification model. Each tree then works like regular decision trees: it partitions the data based on the value of a particular feature (which is selected randomly from the subset), until the data is fully partitioned, or the maximum allowed depth is reached. The final output y is obtained by aggregating the results.

For random forests algorithms, the parameters to tune are the number of trees and the maximum depth of each tree. In order to pick optimum values for these, we tested the algorithm on the data for different combinations of the parameters. Finally, we picked the set of parameters that gave not only the overall highest accuracy but also the highest precision and recall values for crime classes. From this, the maximum accuracy obtained for number of trees = 700 and min-samplessplit depth = 30. Thus, random forests worked fairly well on this problem and we could reach 8th position in the Leader-board.

TABLE : Random Forest Training Study

RF Parameter	Log-loss
$n_estimators = 170$ min_samples_split=70	2.36233
$n_estimators = 250$ min_samples_split=50	2.3550
$n_estimators = 300$ min_samples_split=20	2.3731
$n_estimators = 400$ min_samples_split=35	2.3484

For random forest, parameters we tuned are n_estimators and max_depth. N_estimators is the number of trees to build before taking the maximum voting or averages of predictions. The model is under-fitting when that value is too small and over-fitting when its too large. Max_depth is a limitation of how deep the decision trees are built. It can prevent the model from over-fitting but the model also suffers from under-fitting when the value is too small. Again we tried different values and located a sweat point. We could get the best result with n_estimators=700 and min_samples_split=30. This was achieved only with a higher CPU and 64GB RAM machine as the OOM error were persistent throughout our tests while increasing the tree depth.

0.7.7 Naive Bayes

For Naive Bayes, because the other models outperformed it by too much in the feature validation part and the options of optimization towards this model was very limited, we didnt optimized it at last.

0.8 Results

The submission refers to, regardless of any differences among features, "predicting" the probability of any case belonging to each category to be the categorys frequency in the training data. After submitting the results to the Kaggle competition, our best classifier placed at 7 out of 33 (as of Nov 26, 2018), with a log-loss of 2.07211306800713361. This is among the Top 10% of the leader-board standings.

The result of baselines for Naive Bayes, logistic regression and random forests are shown in TABLE Below. The minimum log-loss comes from the Random Forest model.

TABLE : Min Log-loss

Results for Baseline(Log-loss)	Feature Naive Bayes Logistic	XGBoost	Random Forest
2.616885	2.503459929437911	2.3404000772523994	2.0721130680071336

0.9 Conclusions

Further improvements may be achieved by splitting the features more finely. For example, we notice that the longitude and the latitude are not completely independent as other features do. To modify this we can pair the longitude and the latitude, and perform locally weighted regression in two dimension of location. These may be the directions in which we could try more in the future. In this submission, we explored a wide spectrum of possible classifiers that might be a good fit for solving the San Francisco Crime Classification problem. We achieved a log-loss metric that was higher than most of the published solutions, with subtle feature engineering, and classifier parameter tuning we could train a better model.

0.10 References

Below shared google drive has all our report (LaTeX format), pickle(model) files.

https://drive.google.com/open?id=1juzKE1GfbdcxVrfyTzaHQU3xeyFFE2xb