SAN FRANCISCO CRIME CLASSIFICATION

Tanu Bordia(IMT2016002)
Gautami Gupta(IMT2016069)
Ayush Saraswat(IMT2016123)

Machine Learning Project

International Institute of Technology - Bangalore

I. Problem Statement

From 1934 to 1963, San Francisco was infamous for housing some of the world's most notorious criminals on the inescapable island of Alcatraz.

Today, the city is known more for its tech scene than its criminal past. But, with rising wealth inequality, housing shortages, and a proliferation of expensive digital toys riding BART to work, there is no scarcity of crime in the city by the bay.

From Sunset to SOMA, and Marina to Excelsior, this competition's dataset provides nearly 12 years of crime reports from across all of San Francisco's neighborhoods. Given time and location, you must predict the category of crime that occurred.

II. Data Set

The San Francisco Crime Classification dataset contains the following set of features:

- Dates—timestamp of the crime incident
- Category—category of the crime incident (Column to be predicted)
- Descript—detailed description of the crime incident
- Day of week—the day of the week
- PdDestrict—name of Police Department District
- Resolution—how the crime incident was resolved
- Address—Approximate street address of the crime incident
- X—longitude
- Y—latitude

III. Label/Category

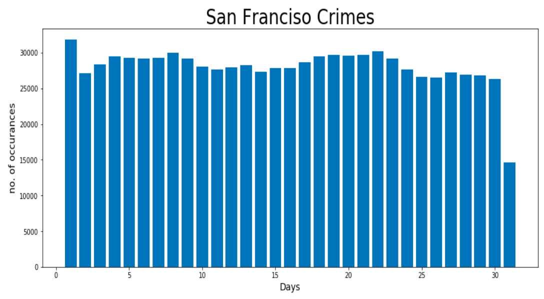
The San Francisco Crime Classification dataset contains the following set of Categories:

- Arson
- Assault
- Bad Checks
- Bribery
- Burglary
- Disorderly Conduct
- Driving under the influence
- Drug/Narcotic
- Drunkenness
- Embezzlement
- Extortion
- Family offences
- Forgery/Counterfeiting
- Fraud
- Gambling
- Kidnapping
- Larceny/Theft
- Liquor Laws
- Loitering
- Missing Person
- Non-Criminal
- Other Offenses
- Prostitution
- Recovered Vehicle
- Robbery
- Runaway
- Secondary Codes
- Sex Offences Forcible
- Stolen Property
- Suicide
- Suspicious occ
- Trespass
- Vandalism
- Vehicle Theft
- Warrants
- Weapon Laws

Data Analysis and Preprocessing: The Resolution column and Description column were supposed to be dropped because they are direclty related. Now, we will explore the features one by one.

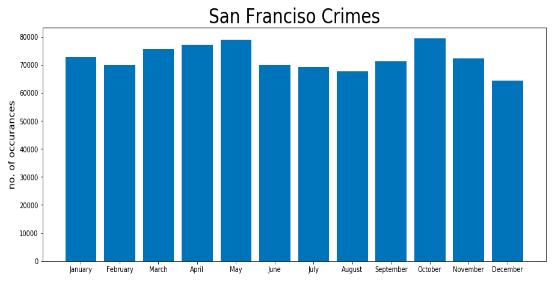
IV. Date

A. Day of the Month



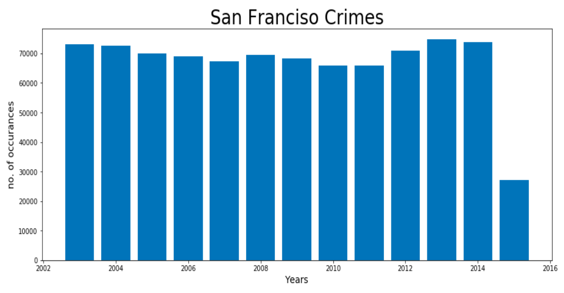
Given the graph above, we can see that the day of the month hardly has any impact of frequency of crimes. Thus, We decided to discard this as one of the features used in training our model.

B. Month



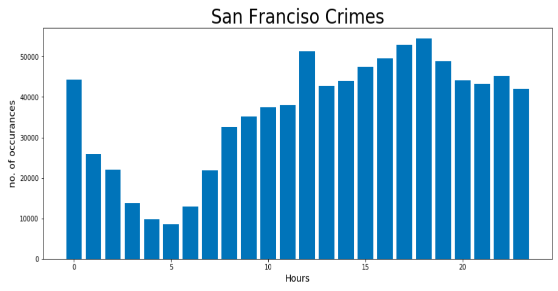
The frequency of crimes varied substantially every month, so we took month as one of the features while training.

C. Year



Though it looks like the crime rate has been steady, except for the sudden drop in 2015(because of the lack of data), the year feature helped improve the model.

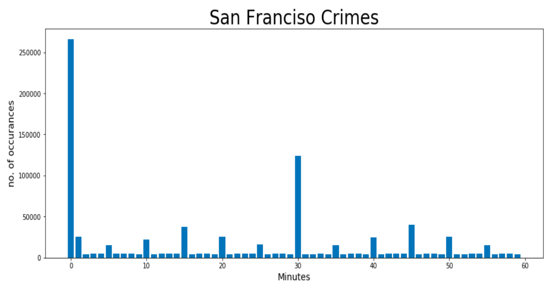
D. Hour



Hour feature was one of the best features in our training data set. As we can see, the crime rate per hour is comparatively low from 1 A.M. to 7 A.M. And the crime rate varies every hour considerably. We extracted another feature:

- High_LowHour:We divided the date set into the hours of low and high crime rate. Low hours were from 1 A.M. to 7 A.M. and the rest were High hours. We didnot use this feature as it did not have much impact.

E. Minutes



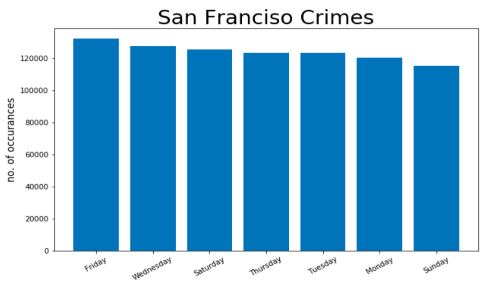
Our first approach was to discard minutes, but the plot suggested otherwise. For the same we decided to preprocess minutes and narrowed the range down to 30 mins by subtracting 30 from minutes ranging from 31 to 59. i.e.

$$minutes = \begin{cases} minutes & \text{if minutes<=30} \\ minutes - 30 & \text{if minutes>30} \end{cases} \tag{1}$$

F. Seconds

The timestamp for every crime in the dataset had seconds=00 so we discarded seconds as one of the features.

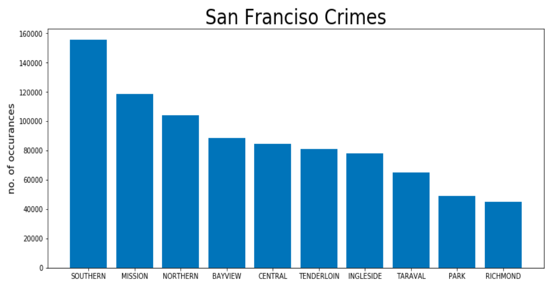
V. DayOfWeek



We included this feature in the model as the crime rate varied considerably with the days.
We also extracted another feature:

- is_Weekend:This checked if the the day of the week was a weekend or not.

VI. PdDistrict



Crime rate per district varies considerably through districts, thus this was an important feature while model training.

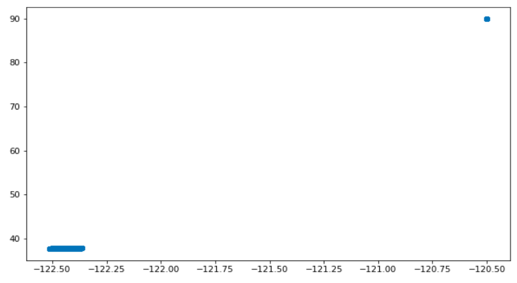
VII. Address

There were 23,201 unique addresses in the given data, thus using the whole column made no sense. But we extracted some features from this column:

- is_crossroad : This feature checked if the given address was a crossroad or not. This was essentially evaluated by checking if there was any '/' present in the address or not.
  We used this feature in model training.
- is_AV : This feature checked if the given crime happened at an avenue or not. This was evaluated by checking if there was 'AV' present in the address string or not.
  We used this feature in model training.
- is_block : This feature checked if crime happened at a block or not, this was evaluated y checking the presence of "Block" in the address string.
  We tried putting this feature in our model but it didn't help, rather decreased our score, so we discarded it.
- Streets We divided the address where we found '/' or 'of'. If it was a crossroad, we got 2 streets, so we put them in alphabetical order. This divided the address in 2 parts:
  - Street1 : The first part of the string, it was either some block or a street.
  - Street2: The second part of the string, it was either some street or Avenue.
  - StreetNo. : This returned the the block number presented in address. If there was no block, we called it junction.

  We didn't use Street1 or Street2 because it led to overfitting since the number of distinct values for them 2000,1500 respectively, were quite high and the model became too specific to these two features

VIII. X and Y



X denotes latitude and Y denotes longitutde. The scatter plot above shows that there are outliers in the given data. While pre-processing out data, we removed them.
We extracted some more feature out of X and Y, essentially to get a spatial information. We got the idea while we were studying the open kernels already given. The link to the same is: Feature Engineering of lat-long (x-y) helps
With this we extracted the given features:

1) Three variants of rotated Cartesian coordinates (rotated by 30, 45, 60 degree each):
   - X_1 and Y_1: Rotated co-ordinates by 30°.
   - rotX_2 and rotY_2: Rotated co-ordinates by 45° .
   - rotX_3 and rotY_3: Rotated co-ordinates by 60°.
2) radial_r : $\sqrt{X^2 + Y^2}$

IX. Training

In this chapter, we'll discuss about the models we implemented.

A. Logistic Regression

Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.
We set the parameter of the same as:

- random_state = 1711
- max_iter = 200
- verbose = 1
- n_jobs = -1
- solver = 'sag'
- multi_class = 'multinomial'

This gave us a logloss metric of 2.40

B. Bernoulli Naive Bayes Algorithm

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.
In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies.
This gave us a logloss metric of 2.51, not that useful for the given dataset.

C. Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

For the given dataset, Random Forest Classifier failed terribly and gave us a logloss metric of 8.2495

D. XGBoost Classifier

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.
We set the following as parameters for the same:

- random_state = 1711
- learning_rate = 0.375
- max_depth=7
- n_estimators = 10
- gamma = 0
- reg_alpha = 0.1
- objective = multi:softprob
- booster = gbtree
- silent = True
- subsample = 0.8
- colsample_bytree = 0.8
- multi_class = 'multinomial'
- max_delta_step = 1
- n_jobs = -1

This gave us the best result of 2.295 logloss metric.
Kaggle Score and Future Work

X. Kaggle Score

After submitting our files, we got a Kaggle Score of 2.25. We stood 11^{th} out of 21 teams that participated.

XI. Future Work

We think we could have improved our score if we applied more feature engineering on X , Y and Address. Also, we would be able to score better if we had used neural networks.

XII. Conclusion

In this paper, We explored a wide spectrum of possible classiers that might be a good fit for solving the San Francisco Crime Classication problem. We achieved a log-loss metric that was higher than most of the published solutions, with subtle feature engineering, and classifier parameter tuning.