CRIME ANALYSIS USING PYTHON

A PROJECT REPORT

Submitted by

MOHD FAIZ(151)

PRAGYA PORWAL(157)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

University Institute of Engineering and Technology

C. S. J. M. University, Kanpur

December 2019

University Institute of Engineering and Technology

C. S. J. M. University, Kanpur

BONAFIDE CERTIFICATE

Certified that this project report **Crime Analysis**

is the bonafide work of Mohd Faiz and Pragya Porwal who carried out the project work under my supervision.

SIGNATURE

Dr.Rashi Agarwal

(SUPERVISOR)

Head Of Department

Information Technology Department

UIET, CSJMU, Kanpur

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, Dr. Rashi Agarwal, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project. We deeply express our sincere thanks to our Head of Department for encouraging and allowing us to present the project on the topic “Crime Analysis using Python“ at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree. We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement through out our career. Last but not the least we express our thanks to our friends for their cooperation and support.

MOHD FAIZ(151)

PRAGYA PORWAL(157)

ABSTRACT

Data Analysis And Crime Prediction In San Francisco by Mohd Faiz and Pragya Porwal

Crime has been prevalent in our society for a very long time and it continues to be so even today. The San Francisco Police Department has continued to register numerous such crime cases daily and has released this data to the public as a part of the open data initiative. In this paper, Data analysis is used on this dataset and a tool that predicts crime in San Francisco is provided. The focus of the project is to perform an in-depth analysis of the major types of crimes that occurred in the city, observe the trend over the years, and determine how various attributes, such as seasons, contribute to specific crimes. Furthermore, the proposed model is described that builds on the results of the performed predictive analytics, by identifying the attributes that directly affect the prediction. More specifically, the model predicts the type of crime that will occur in each district of the city. After preprocessing the dataset, the problem reduced to a multi-class classification problem. K-nearest neighbor is used. Lastly, our results are experimentally evaluated and compared against previous work. The proposed model finds applications in resource allocation of law enforcement in a Smart City.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

Introduction

The concept of a smart city has been derived as one of the means to improve the lives of the people living within the city by taking smart initiatives in a variety of domains like urban development, safety, energy and so on . One of the factors that determine the quality of life in the city is the crime rate therein. Although there could be a lot of technological advancement in the city but the basic requirement of citizens' safety still remains.

Crime continues to be a threat to us and our society and demands serious consideration if we hope to reduce the onset or the repercussions caused by it. Hundreds of crimes are recorded daily by the data officers working alongside the law enforcement authorities throughout the United States. Many cities in the United States have signed the Open Data initiative, thereby making this crime data, among other types of data, accessible to the general public. The intention behind this initiative is increasing the citizens' participation in decision making by utilizing this data to uncover interesting and useful facts.

The city of San Francisco is one amongst the many to have joined this Open Data movement. The data scientists and engineers working alongside the San Francisco Police Department (SFPD) have recorded over 100,000 crime cases in the form on police complaints they have received. With the help of this historical data, many patterns can be uncovered. This would help us predict the crimes that may happen in the future and thereby help the city police better safeguard the population of the city. Motivation

1.1 Motivation

The motivation behind taking up this topic for the research is that every aware citizen in today's modern world wants to live in a safe environment and neighborhood. However it is a known fact that crime in some form, exists in our society. Although we cannot control what goes on around us, we can definitely try to take a few steps to aid the government and police authorities in trying to control it. The SFPD has made the Police Complaints data from the year 2003 to 2015 available to the general public. Hence, taking inspiration from the facts stated above, we decided to process this data provided and analyze it to identify the trends in crime over the years as well as make an attempt to predict the crimes in the future.

1.2 Problem Formulation

The problem being tackled in this project can be best explained in two distinct parts:

1. Performing exploratory analysis of the data to mine patterns in crime:
    - The first step in determining the safety within different areas in the city is

analyzing the spread and impact of the crime.

- We utilize this provided crime dataset by the SFPD and perform exploratory analysis on it, to observe existing patterns in the crime throughout the city of San Francisco.

2. Building a prediction model to predict the type of crime that can take place in the city, in the test dataset:

- After observing the patterns of crime from the historical data as explained previous, the next thing is to predict the crimes that can occur in the test dataset.
- Our goal is to build a prediction model that treats this problem as a multi-class classification problem, by classifying the unseen data into one of the crime categories (classes) thereby predicting the crime that can occur.

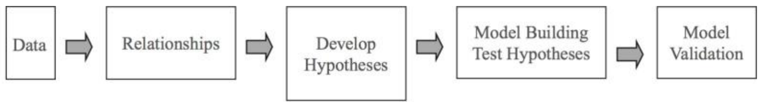
CHAPTER 2

Definitions and Techniques

In this chapter we will go over the details of some concepts and techniques which will be discussed and implemented throughout this project.

2.1Predictive Analytics

Predictive analytics is the technique of analyzing the past or historical data in order to predict the future outcome. It is different from data mining. As explained in the Figure predictive analysis starts by capturing relationships between the different variables in the data. After that, hypothesis is developed based on these results. Following this, based on the outcome of the previous steps, a model is built in order to test this hypothesis



-

Figure 1: Data mining and predictive analytics.

There are numerous advantages of using Predictive Analytics in general. For example, an organization can study it's internal data to identify trends in profit, so that they can adopt the necessary steps to possibly replicate that in the future. It is also a useful technique for professionals in the marketing industry as it can help decide which campaign successfully generated revenue and business. For the purpose of this research, Predictive Analytics is helpful for the following reasons:

- It will help us identify the progression in crime throughout the years
- It will help us closely observe the variables having highest correlation with the predictor or target variable
- Visualizing the data can even help map potential outliers, which can then be effectively handled during data preprocessing
- The analysis can bring up some interesting facts from the past which might prove to be useful to the SFPD while planning their patrol or strategies against crime

2.2 Classification Techniques

Classification techniques are used to segregate the data into one or more categories also known as class labels. The goal of classification is to create a certain set of rules that will either make a binary decision, or predict which of the multiple classes should the data be classified into. Classification can mainly be divided into two types:

1. Binary Classification

   In binary classification, The goal is to classify the element into one of the two categories specified, say x and not x. To determine the efficiency of the Binary Classifier, we pass a set of inputs to the classifier and examine the output. There are 4 possible results - True Positive, True Negative, False Positive and False Negative. Generally, 'Accuracy' is used to determine the efficiency of a binary classifier.

2. Multiclass Classification

   Multiclass Classification involves classifying the data into more than two categories. The most common types of Multiclass Classifiers are:

   - Pigeonhole Classifier: In a pigeonhole classifier every item is classified into only one of the many categories. Hence, for a given item, there can be only one output category assigned to it.
   - Combination Classifier: This type of classifier can place an item into more than one output categories. Hence, unlike a pigeonhole classifier, this type of classifier does not assign a unique category to each input.
   - Fuzzy Classifier: These classifiers not only assign an input to more than one categories but also assign a degree to each category. This means that every input belongs to every category by a certain degree. Hence the output is an N-dimensional vector, where N is the number of categories.

Below we will look at the classification techniques that have been used in this project.

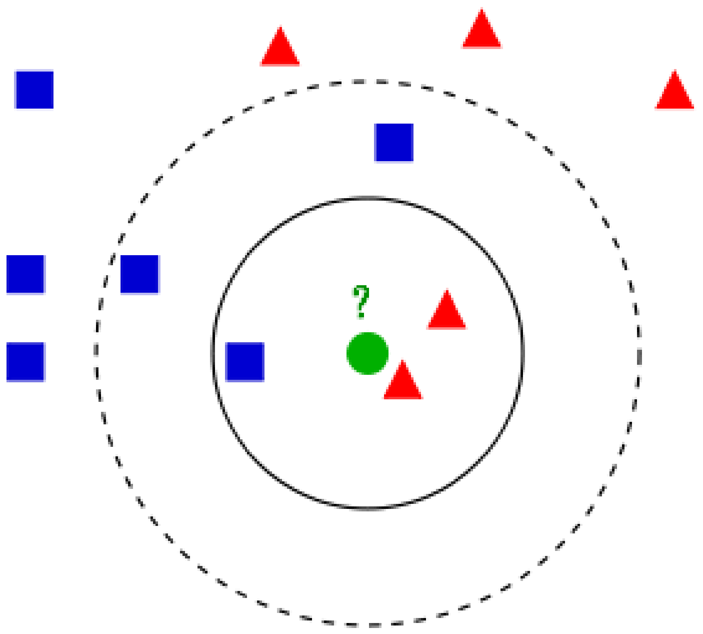
1. K- Nearest Neighbor

According to the K-Nearest Neighbor (KNN) algorithm, data is classified into one of the many categories by taking a majority vote of its neighbors. The label is assigned depending on the most common of the categories among its neighbors. In other words, we identify the neighbors closest to the new point we wish to classify and based on these neighbors, we predict the label of this new point. The number of neighbors to consider can be a user-defined constant K as in the case of K-nearest neighbors or it can be based on the density of points in a certain radius specified.

The distance metric used can be Euclidean or Manhattan if the target variable is a continuous variable or Hamming distance if the target variable is a categorical variable

The Figure3illustrates a simple KNN classifier. Here, the green circle in question would be categorized as a red triangle if we consider k=3, i.e. if we consider its 3 nearest neighbors. If k=5, then it would be categorized as a blue square.

Figure 3: KNN Classifier Examle



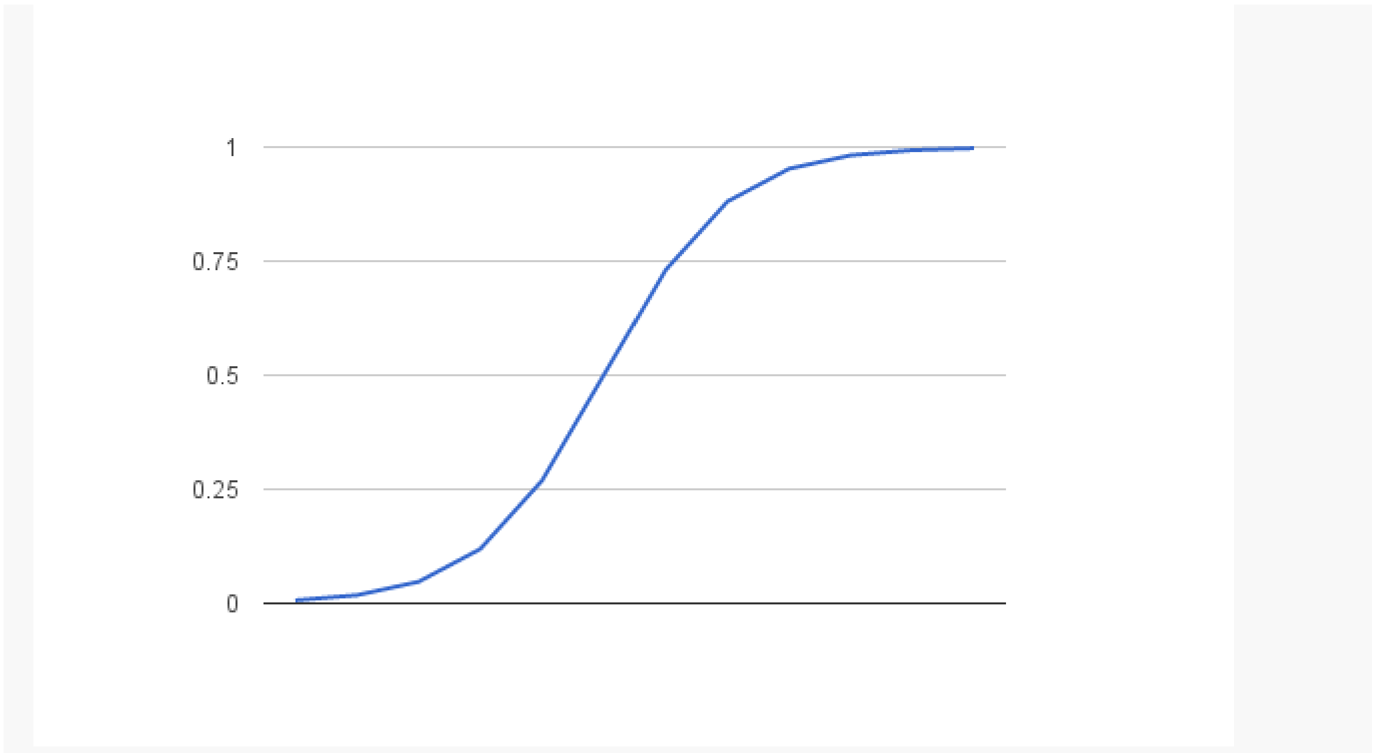
2. Logistic Regression

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1. 1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.



Logistic Function

Now that we know what the logistic function is, let's see how it is used in logistic regression.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

3.Random Forest

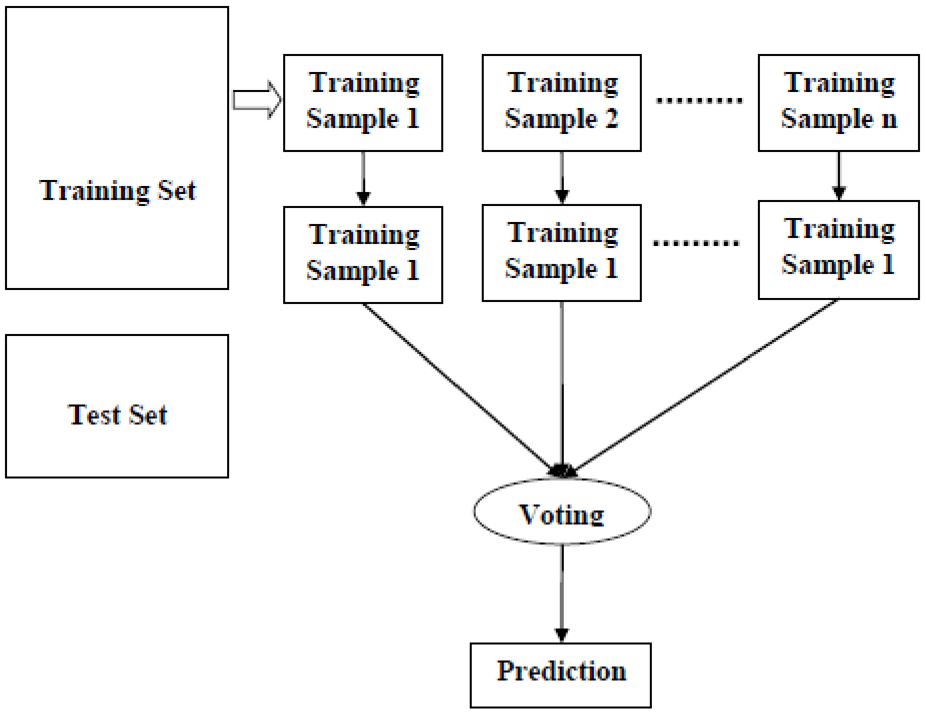
Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps −

- **Step 1** − First, start with the selection of random samples from a given dataset.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.
- **Step 4** − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −



4.Naïve Bayes

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as:

P(h|d) = (P(d|h) * P(h)) / P(d)

Where

- **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.
- **P(d|h)** is the probability of data d given that the hypothesis h was true.
- **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- **P(d)** is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

5.XGboost Ensemble

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now

XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications. As a result, there is a strong community of data scientists contributing to the XGBoost open source projects with ~350 contributors and ~3,600 commits on GitHub. The algorithm differentiates itself in the following ways:

1. A wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.

2. Portability: Runs smoothly on Windows, Linux, and OS X.

3. Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.

4. Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

6.Gradient descent

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

Think of a large bowl like what you would eat cereal out of or store fruit in. This bowl is a plot of the cost function (f). A random position on the surface of the bowl is the cost of the current values of the coefficients (cost).

The bottom of the bowl is the cost of the best set of coefficients, the minimum of the function.

The goal is to continue to try different values for the coefficients, evaluate their cost and select new coefficients that have a slightly better (lower) cost.

Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost.

CHAPTER 3

Related Work

Over the years, there have been a lot of studies involving the use of predictive analytics to observe patterns in crime. Some of these techniques are more complex than others and involve the use of more than one datasets. Most of the datasets used in these researches are taken from the Open Data initiative supported by the government. In this section we will study the various techniques used by different authors which will help answer questions such as: *What is the role of analytics in crime prediction?, What techniques are used for data preprocessing?* and *What are the classification techniques which have proved to be most efficient?*

3.1 Temporal and Spectral Analysis

A lot of research in the area of crime analysis and prediction revolves around the analysis of spatial and temporal data. The reason for this is fairly obvious as we are dealing with geographical data spread over the span of many years.

The authors of have studied the fluctuation of crime throughout the year to see if there exists a pattern with seasons. In their research, they have used the crime data from three different Canadian cities, focusing on property related crimes. Ac- cording to their first hypothesis, the peaks in crime during certain time intervals can be distinctly observed in case of cities where the seasons are more distinct. Their sec- ond hypothesis is that certain types of crimes will be more frequent in certain seasons because of their nature. They were able to validate their hypothesis using Ordinary Least Squares (OLS) Regression for Vancouver and Negative Binomial Regression for Ottawa. Since their research focused on crime seasonality, quadratic relationship in the data was predicted. Crime peaks were observed in the Summer months as compared to Winter.

In a similar study, the authors of have analyzed the crime data of two US cities - Denver,CO and Los Angeles,CA and provide a comparison of the statistical analysis of the crimes in these cities. Their approach aims on finding relationships between various criminal entities as this would help in identifying crime hotspots. To increase the efficiency of prediction, various preprocessing techniques like dimen- sionality reduction and missing value handling were implemented. In the analysis, they compared the percentage of crime occurrence in both cities as opposed to the count of crimes. Certain common patterns were observed in both the cities such as the fact that Sunday had the lowest rate of crime in both the cities. Also, important derivations like the safest and the most notorious district, were noted. Decision Tree classifier and Naive Bayes classifier were used.

L. Venturini *et al.* in their paper have discovered spatio-temporal patterns in crime using spectral analysis. The goal is to observe seasonal patterns in crime and verifying if these patterns exist for all the categories of crime or if the patterns change with the type of crime. The temporal analysis thus performed highlights that the patterns not only change with month but also with the type of crime. Hence, the authors of rightly stress the fact that models built upon this data would need to account for this variation. They have used the Lomb-Scargle periodogram to highlight the seasonality of the crime as it deals better with uneven or missing data. The AstroML Python package was used to achieve this. In their paper they have described in detail how every category of crime performs when the algorithm is applied to the data. Further, the authors suggest that researchers should focus on the monthly and weekly crime patterns

3.2 Prediction using Clustering and Classification techniques

The authors have described a method to predict the type of crime which can occur based on the given location and time. Apart from using the data from the Portland Police Bureau (PPB), they have also included data such as ethnicity of the population, census data and so on, from other public sources to increase the accuracy of their results. Further, they have made sure that the data is balanced to avoid getting skewed results. The machine learning techniques that are applied are Support Vector Machine (SVM), Random Forest, Gradient Boosting Machines, and Neural Networks . Before applying the machine learning techniques to predict the category of the crime, they have applied various preprocessing techniques such as data transformation, discretization, cleaning and reduction. Due to the large volume of data, the authors have sampled the data to less than 20,000 rows. They used two datasets to perform their experiments - one was with the demographic information used without alterations and in the second case, they used this data to predict the missing values in the original dataset. In the first case, ensemble techniques like as Random Forest or Gradient Boosting worked best, while in the second case, SVM and Neural Networks showed promising results.

Since a smart city should give importance to the safety of their citizens, the authors have designed a strategy to construct a network of clusters which can assign police patrol duties, based on the informational entropy. The idea is to find patrol locations within the city, such that the entropy is maximized. The reason for the need to maximize the entropy is that the entropy in this case is mapped to the variation in the clusters, i.e. more entropy means more cluster coverage . The dataset used for the research is the Los Angeles County GIS Data. The data has around 42 different crime categories. Taking the help of a domain expert, the authorshave assigned weights to these crimes based on the importance of the crime. Also the geocode for each record is taken into consideration and the records that do not have a geocode are skipped. Because the authors are trying to maximize the entropy in this case, consider the equation

$$H_{c1} = -p(c_1)p(c_1) \tag{3}$$

The probability p(c1) is defined as the ratio of weight of the centroid of the crime to the weight of the system, plus the ratio of the quickest path between two centroids, to the quickest path in the whole system.

The authors have taken a unique approach towards crime classification where unstructured crime reports are classified into one of the many categories of crime using textual analysis and classification. For achieving this, the data from vari- ous sources including but not limited to the databases which stores information about traffic, criminal warrants of New Jersey (NJ) and criminal records from NJ Criminal History was combined and preprocessed. As a part of the preprocessing activity, all the stop words, punctuations, case IDs, phone numbers and so on were removed from the data. Following this, document indexing is performed on the data to convert the text into its concise representation. In order to identify the topics or specific incident types from the concise representation, the authors used Latent Semantic Analysis (LSA). Next, similarity between these topics was identified using the Topic Model- ing technique where the closer the score is to 1, the more similar it is to the topic which was followed by Text Categorization. The classification methods used in this research were Support Vector Machines (SVM), Random Forests, Neural Networks, MAXENT (Maximum Entropy Classifier), and SLDA (Scaled Linear Discriminant analysis). However, the authors observed that SVM performed consistently better of them all.

3.3 Hotspot Detection

A crime hotspot is an area where the occurrence of crime is high as compared to the other locations. Many researchers have taken an interest in determining crime hotspots from the given dataset. The authors of mainly discuss two approaches for detecting hotspots - circular and linear. The authors also discuss the fundamentals of Spatial Scan Statistics which is a useful tool for hotspot detection. The results on the Chicago crime dataset are also discussed in detail using both the approaches.

CHAPTER 4

Design and Implementation

The fundamental goal of the project is to build a model such that it can predict the crime category that is more likely to surface given a certain set of characteristics like the time, location, month and so on. We will also take the help of statistical and graphical analysis to help determine which attributes contribute to the overall improvement.

4.1 Overview of the dataset

The data used in this research project is the San Francisco crime dataset made available by the San Francisco Police Department on the SF Open Data website, which is a part of the open data initiative. The dataset consists of the following attributes:.

- Descript: It is a text field. Contains a brief description about the crime. This field provides slightly more information than the *Category* field but is still quite limited.
- DayOfWeek: It is a text field. Specifies the day of the week when the crime occurred. It takes on one of the values from: *Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday*
- Date: It is a Date-Time field. Specifies the exact date of the crime.
- PdDistrict: It is a text field. Specifies the police district the crime occurred in. San Francisco has been divided in 10 police districts. It takes on one of the values from: *Southern, Tenderloin, Mission, Central, Northern, Bayview, Richmond, Taraval, Ingleside, Park*
- Resolution: It is a text field. Specifies the resolution for the crime. It takes one of these values: *Arrested, Booked, None*
- Address: It is a text field. Gives the street address of the crime.
- X: It is a geographic field. It gives the longitudinal coordinates of the crime.
- Y: It is a geographic field. It gives the latitudinal coordinates of the crime.
- Location: It is a location field. It is in the form of a pair of coordinates, i.e. (X, Y)
- Category: It is a text field. Specifies the category of the crime. Originally, there

are 39 distinct values (such as *Assault, Larceny/Theft, Prostitution, etc.*) in this field. It is also the dependent variable we will try to predict for the test set.

There are about 878049 rows in the dataset and the size of the dataset is approximately 121 MB. It contains data from the year 2003 to (january) 2015.

4.2 Data Preprocessing

The data made available by the San Francisco Police Department is mostly complete with no Null values. However, there are a few outliers which have been handled. This will be explained in detail in the subsequent sections. The dataset provided a lot of potential to extract more meaningful information from the existing columns. Hence, a few columns have been removed or transformed to improve the score of the resulting prediction.

4.2.1 Techniques used for preprocessing

The original dataset is modified to create a new dataset where new columns are added, existing columns are transformed and outliers are handled. This gives much better results than the existing data. The decision to add or transform columns has been taken by studying the graphical analysis which has been performed on the data prior to building a model.

4.2.1.1 Data Cleaning

Data Cleaning is taking care of incorrect or missing data. Although the dataset does not contain Null values or missing values.

There are 39 distinct categories in the dataset. However, some of the categories are very similar to each other. For example, when the Category column contains values or keywords like: *INDECENT EXPOSURE* or *OBSCENE* or *DISORDERLY CONDUCT* then return *PORNOGRAPHY/OBSCENE MAT.*

4.2.1.2 Data Transformation

Data transformation is one of the most important data preprocessing techniques. Usually, the data is originally present in the form that makes more sense if it is transformed. In this case, the main transformations performed are as follows:

1. **Feature Extraction**:
   There a lot of features like Address, Time, Date, X and Y which can be transformed into new features that hold more meaning as compared to the existing ones. Hence, all of these features have been used to generate new features and some of these old features have been eliminated.

Time* to *Hour : The Time feature is in the Timestamp format. It would be interesting to observe patterns in crime by the hour. Hence the Hour field is extracted from the Time field.

Date* to *Day, Year and Month: The Date field is a very important one for prediction. Using this single field, we are able to extract four features. Spark provides inbuilt methods to extract the Day, Month and Year from the Date and hence our script makes use of the same.

4.3 Software and Technologies Used

This project is implemented in Python (version 3.6.4). The other libraries used throughout the project are described below:

Pandas: Pandas is an open source library that provides tools for data mining and analysis using Python. It is mainly used in this project to prepare the data for consumption by specific machine learning algorithms.

NumPy: NumPy is a Python library that can handle multidimensional data and perform scientific and mathematical operations on the same. NumPy was used in this project as an accessory to the Pandas library to perform some basic mathematical operations.

scikit-learn: scikit-learn is an open-source Python machine learning library which provides numerous classification, regression and clustering algorithms. This library was used in this project to perform the actual task of model building and prediction. It provides a variety of evaluation metrics to validate the performance of the model, which makes it a valuable tool.

CHAPTER 5

Experimental Results

5.1Results of Graphical Analysis

The sum of the distinct categories of crime in the dataset, is illustrated in the figure1.

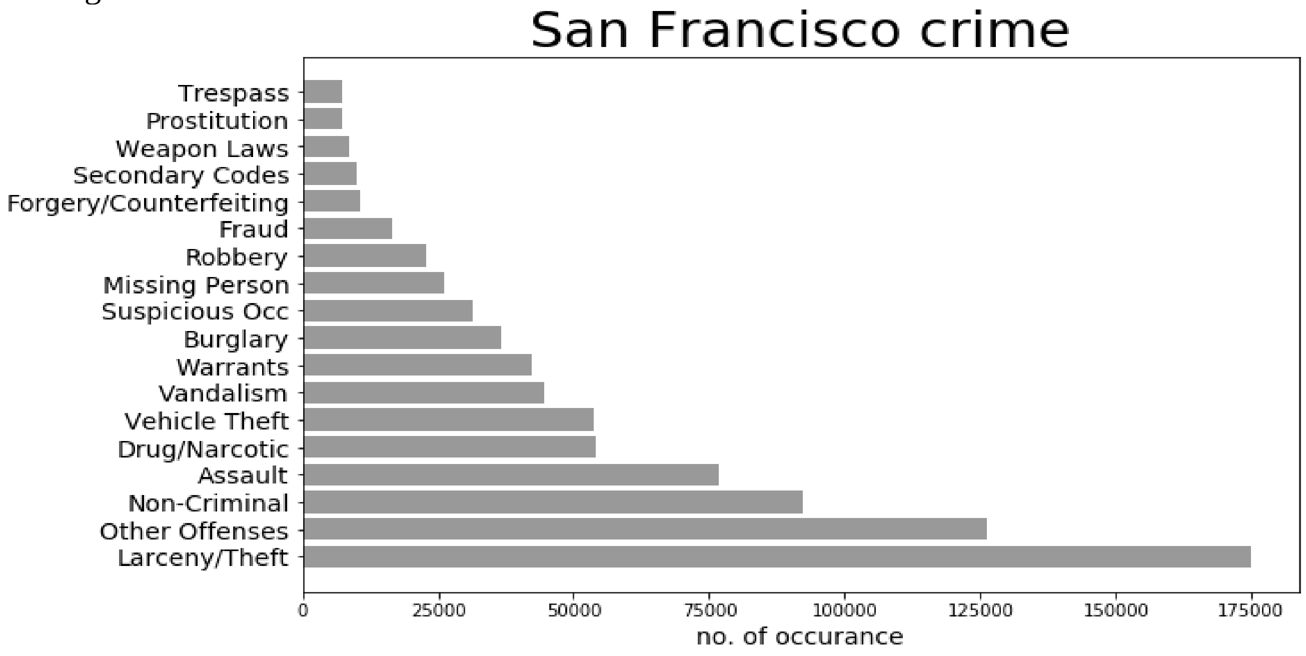


Figure 1: Count of Distinct Categories in the Dataset

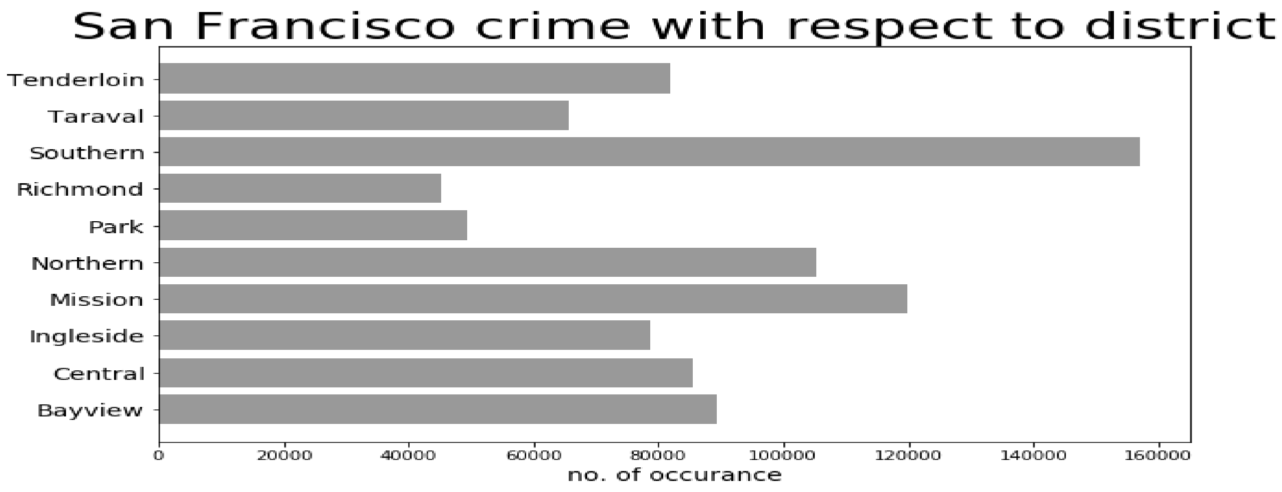


Figure 2: Total Crime per District

Figure2 shows the trend of the crime over the years in various districts of San Francisco. These are the Police Districts and each of those include many other city districts. Here, you can see that the crime in *SOUTHERN*, *MISSION* and *NORTH- ERN*, *TENDERLOIN* and *BAYVIEW* is highest.

If you look at Figure3, you can see that there is a clear pattern in crime and the hour of the day. Generally, crime rate is low in the morning hours from around 1:00 AM to 6:30 AM and it rises to its peak in the morning rush hours, i.e. from 11 AM to 12:00 PM and is generally high at night. However, it would be really interesting

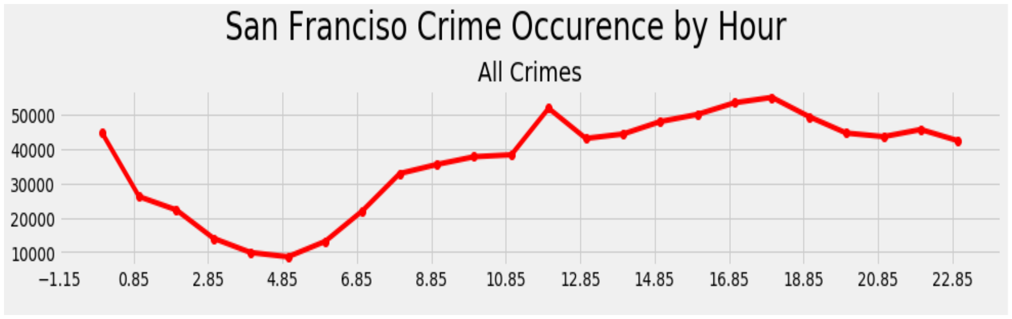


Figure 3: Rate of overall crime every Hour

to see if this pattern is followed by all the different types of crime. For this, we plot graphs for the top six crimes that we found interesting.

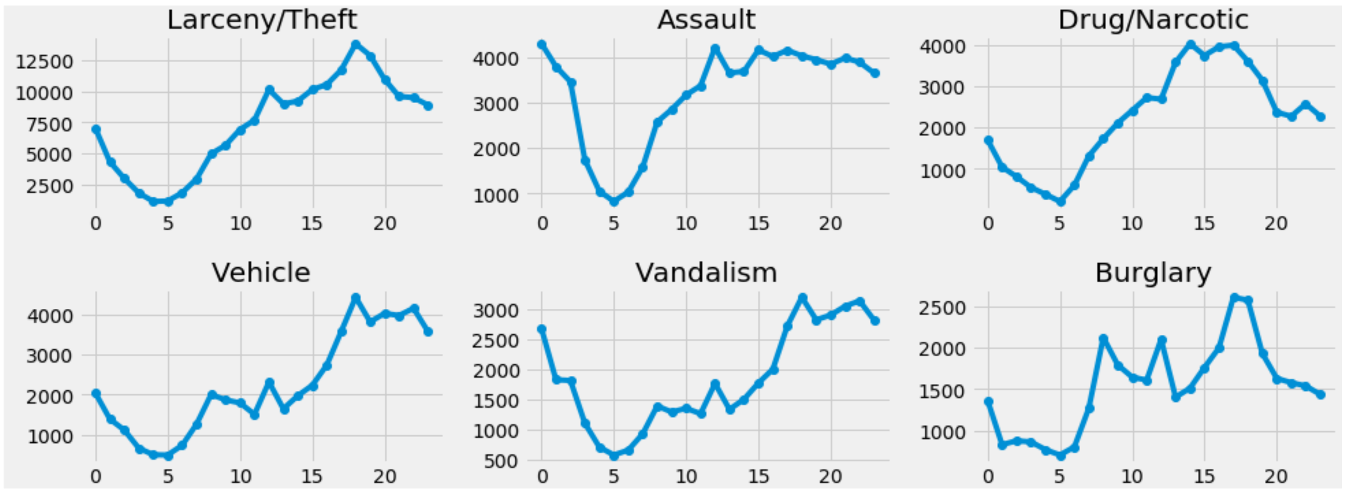


Figure 4: Rate of top 6 cri every Hour

These graph shows the increment and decrement in crime with respect to tome. We can see the crime rate rises to its peak in the evening rush hours and increases continuously until the morning.

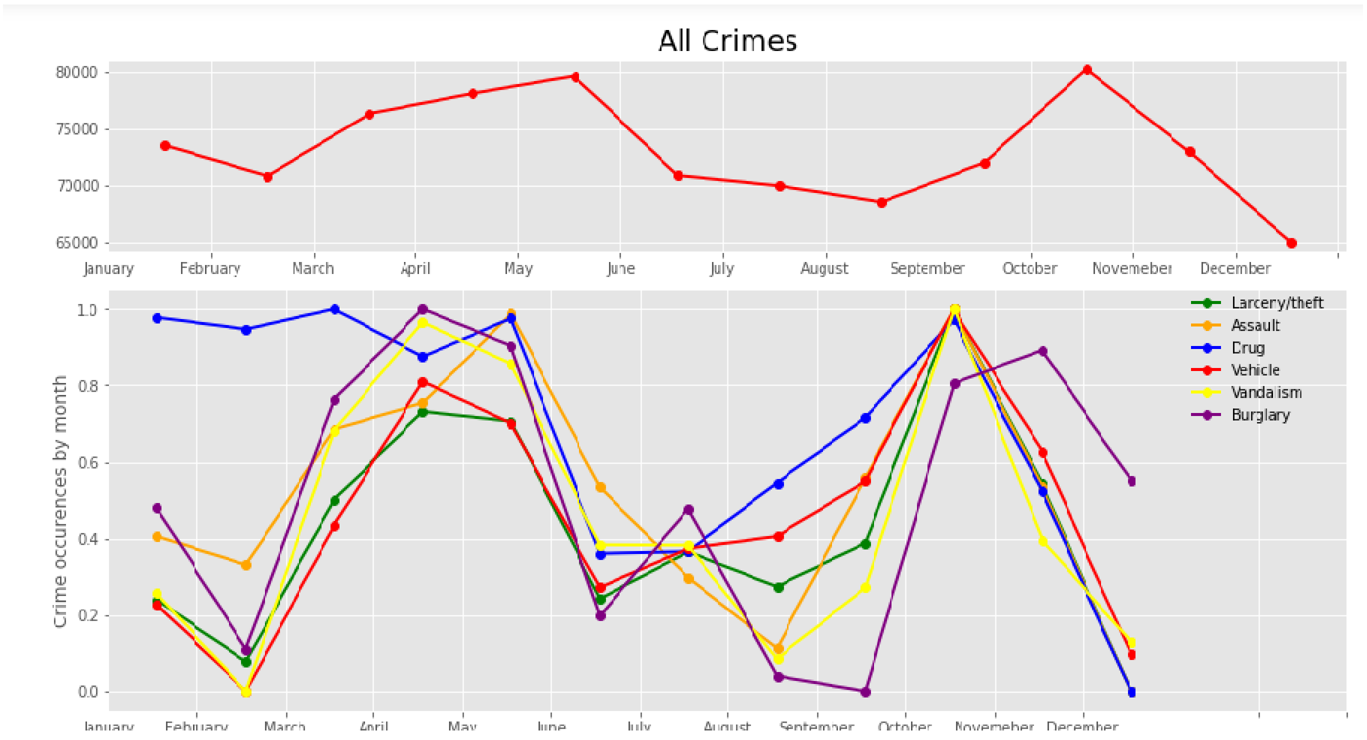


Figure5: crime occurance by month

Next we plot the graph Figure 5 of crime rate according to the month you can see that the crime rate is reach it peaks in the month October and November then its decreases to minimum.

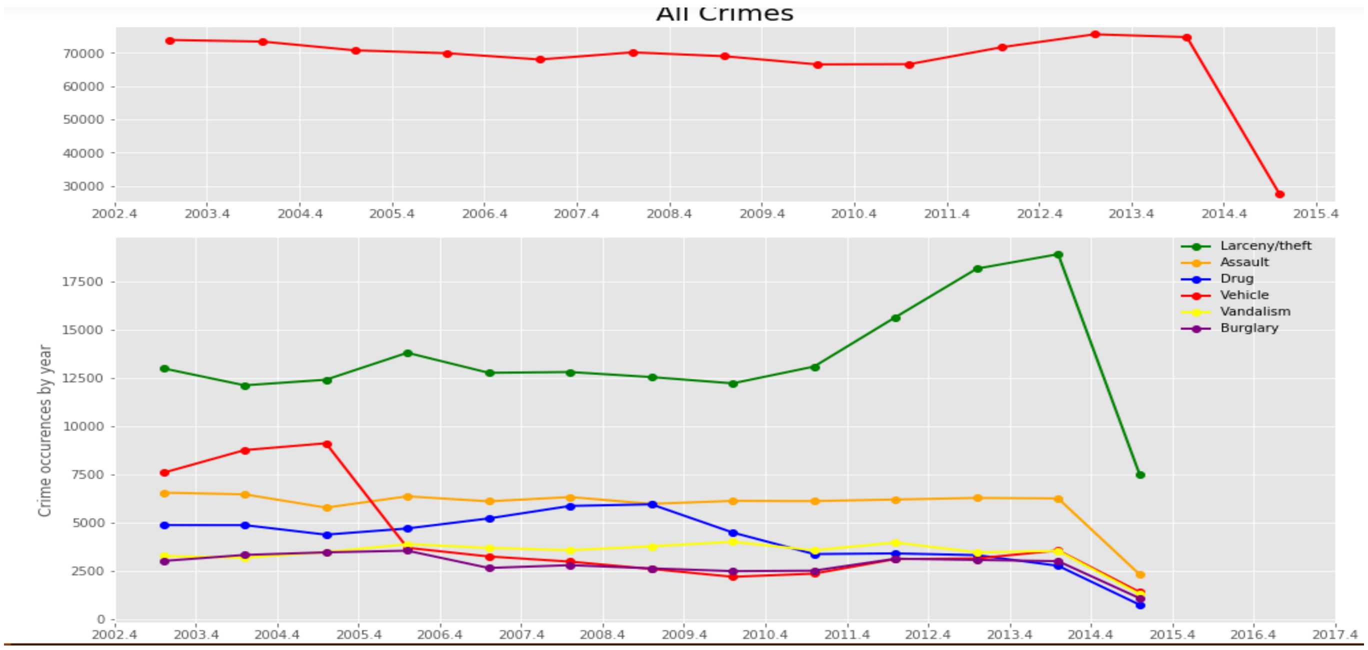
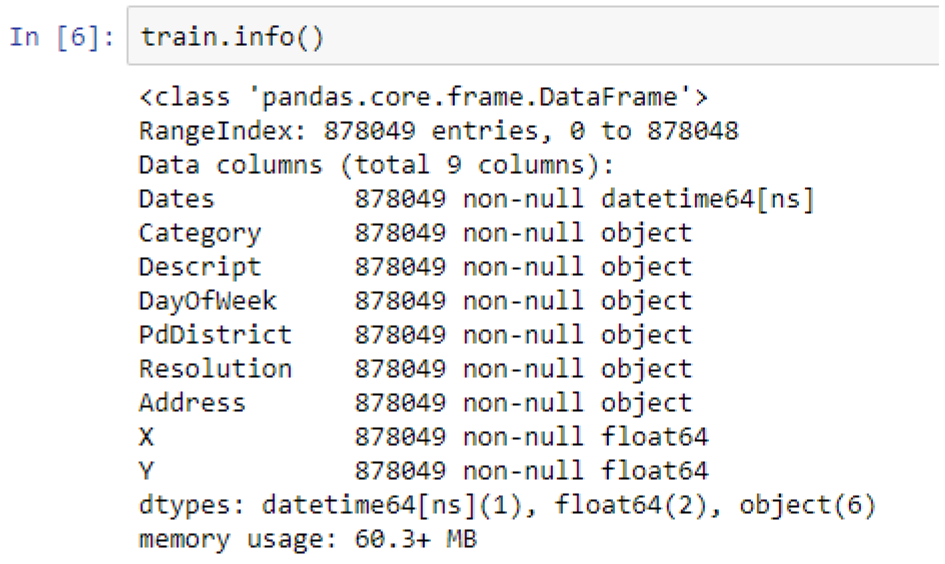


Figure6: Crime occurance per Year

In figure6 we plot the crime rate per year we can see that the crime rate is slightly decreased from 2004 to 2016 but after that it increases slightly and then after 2011 it increases very rapidly and reached the maximum peak.

5.2what data we have:



```
In [6]: train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 878049 entries, 0 to 878048
Data columns (total 9 columns):
Dates          878049 non-null datetime64[ns]
Category       878049 non-null object
Descript       878049 non-null object
DayOfWeek      878049 non-null object
PdDistrict     878049 non-null object
Resolution     878049 non-null object
Address        878049 non-null object
X              878049 non-null float64
Y              878049 non-null float64
dtypes: datetime64[ns](1), float64(2), object(6)
memory usage: 60.3+ MB
```

This dataset suffers from **imbalanced classes** (**TREA** has 6 occurrences while **LARCENY/THEFT** has 1,749,000 occurrences)

- There are a couple ways to deal with imbalanced classes, such as:
  - Changing performance metric (Do not use accuracy, use a confusion matrix, precision, recall, F1 score, ROC curves)
  - Resample dataset (Oversample under-represented classes, and undersample over-represented classes)
  - Try different ML algorithms that can handle imbalanced classes
    - Decision Trees (Random Forests/XGBoost) often perform well on imbalanced classes (due to splitting rules)

Things we learn thus far

- 878,049 instances in training set (or recorded crime instances in SF)
- 9 columns (8 potential features + 1 label (Category))
- Data types:

    - 2 columns with float values
    - 7 objects
- There are no null (NaN) values

5.3Data preprocessing

5.3.1Data Cleaning

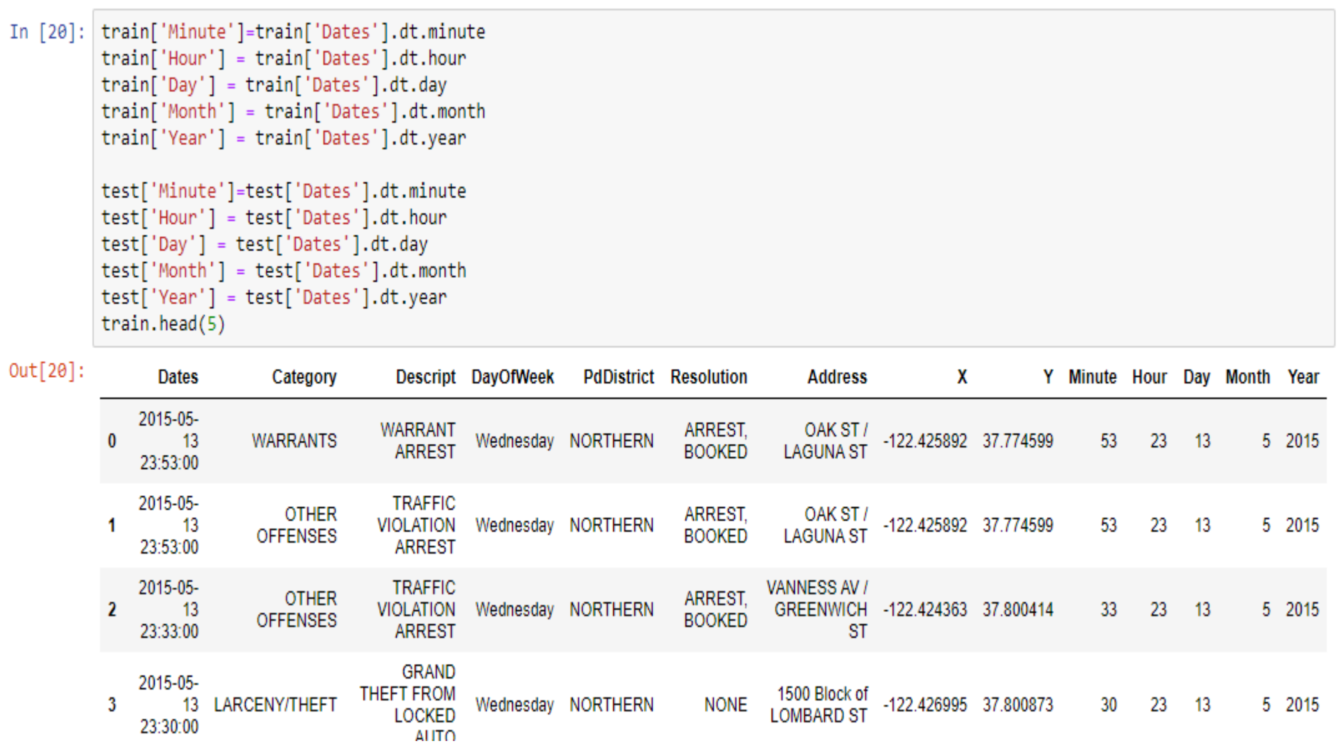
we notice that there are 108 rows with incorrect coordinates, and they seem to be the exact same two coordinates (90, -120.5). There are many ways to handle this. We need to do data imputation, which can be done several ways. For now, I will randomly sample from a normal distribution with the range of a standard deviation from the mean. However, I could use a linear regression model to predict the latitude and longitude values (based on other variables such as PD district?) and use that to impute the bad / inconsistent data points.

5.3.2Feature Engineering

- Let's create some new features from the data that exists in the current feature space
- There are a couple categories of features:
    - Temporal features
    - Spatial features

5.3.2.1Temporal Features

We want to have a column for Time, so we must parse through the 'Dates' feature to create the 'Time' feature



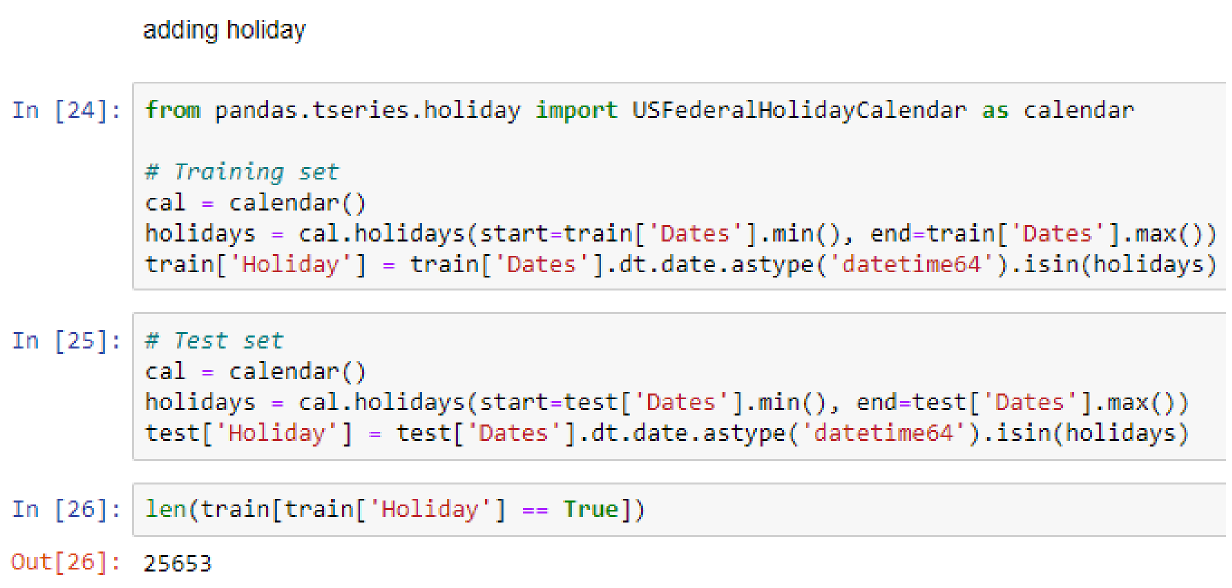
```
In [20]: train['Minute']=train['Dates'].dt.minute
         train['Hour'] = train['Dates'].dt.hour
         train['Day'] = train['Dates'].dt.day
         train['Month'] = train['Dates'].dt.month
         train['Year'] = train['Dates'].dt.year

         test['Minute']=test['Dates'].dt.minute
         test['Hour'] = test['Dates'].dt.hour
         test['Day'] = test['Dates'].dt.day
         test['Month'] = test['Dates'].dt.month
         test['Year'] = test['Dates'].dt.year
         train.head(5)
```

Out[20]:

	Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y	Minute	Hour	Day	Month	Year
0	2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599	53	23	13	5	2015
1	2015-05-13 23:53:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599	53	23	13	5	2015
2	2015-05-13 23:33:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	VANNESS AV / GREENWICH ST	-122.424363	37.800414	33	23	13	5	2015
3	2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1500 Block of LOMBARD ST	-122.426995	37.800873	30	23	13	5	2015

5.3.2.2Holiday Feature

- Certain crimes may be more apparent on holidays



adding holiday

```
In [24]: from pandas.tseries.holiday import USFederalHolidayCalendar as calendar

         # Training set
         cal = calendar()
         holidays = cal.holidays(start=train['Dates'].min(), end=train['Dates'].max())
         train['Holiday'] = train['Dates'].dt.date.astype('datetime64').isin(holidays)
```

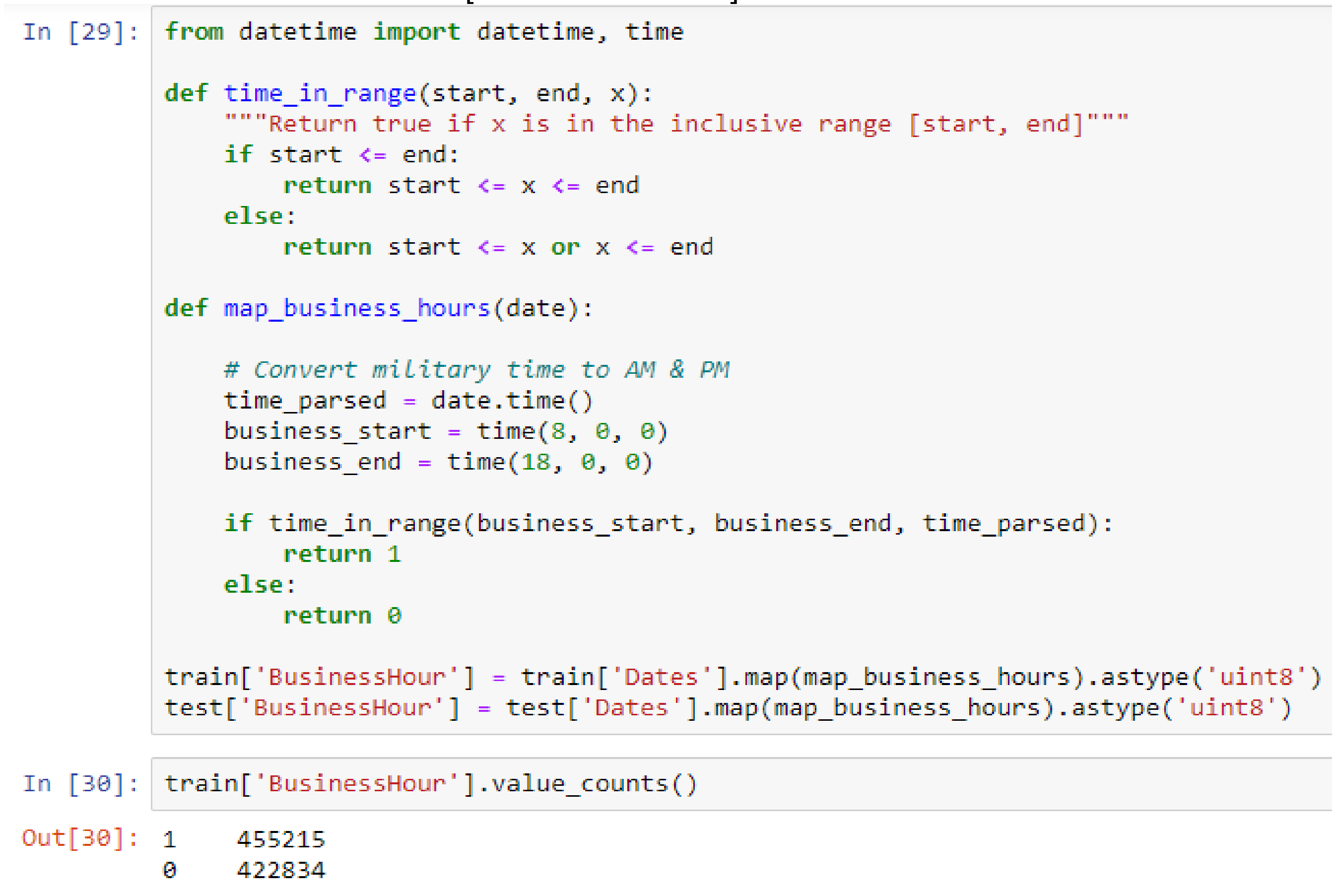
```
In [25]: # Test set
         cal = calendar()
         holidays = cal.holidays(start=test['Dates'].min(), end=test['Dates'].max())
         test['Holiday'] = test['Dates'].dt.date.astype('datetime64').isin(holidays)
```

```
In [26]: len(train[train['Holiday'] == True])

Out[26]: 25653
```

5.3.2.3Business Hours Feature

There should be an effect of business hours on the type of crime committed
Let's create a binary feature where:

- 1 is typical business hours [8:00AM - 6:00PM]
- 0 is not business hours [6:01PM - 7:59 AM]



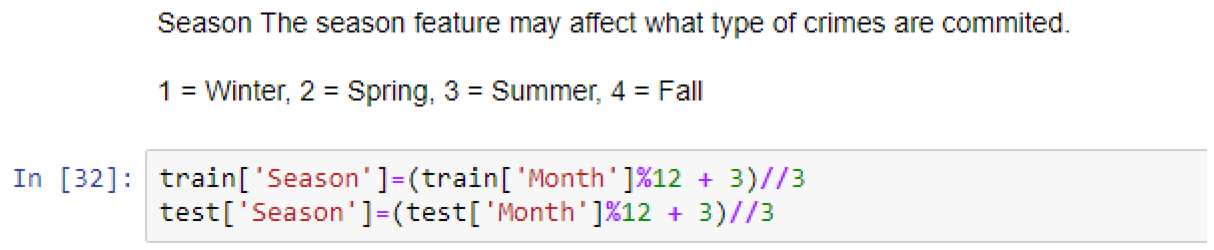
```
In [29]: from datetime import datetime, time

         def time_in_range(start, end, x):
             """Return true if x is in the inclusive range [start, end]"""
             if start <= end:
                 return start <= x <= end
             else:
                 return start <= x or x <= end

         def map_business_hours(date):

             # Convert military time to AM & PM
             time_parsed = date.time()
             business_start = time(8, 0, 0)
             business_end = time(18, 0, 0)

             if time_in_range(business_start, business_end, time_parsed):
                 return 1
             else:
                 return 0

         train['BusinessHour'] = train['Dates'].map(map_business_hours).astype('uint8')
         test['BusinessHour'] = test['Dates'].map(map_business_hours).astype('uint8')
```

```
In [30]: train['BusinessHour'].value_counts()

Out[30]: 1    455215
         0    422834
```

5.3.2.4Season

The season feature may affect what type of crimes are committed.

- 1 = Winter, 2 = Spring, 3 = Summer, 4 = Fall



Season The season feature may affect what type of crimes are commited.

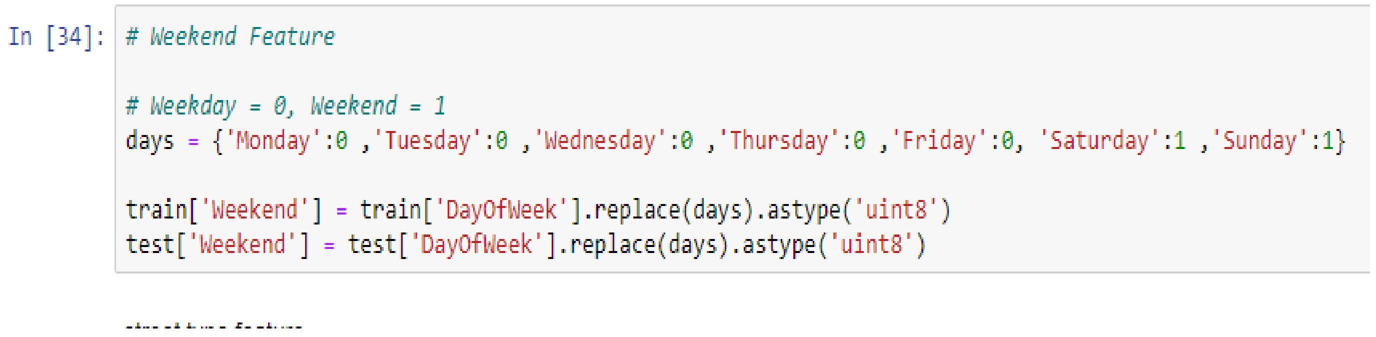
1 = Winter, 2 = Spring, 3 = Summer, 4 = Fall

```
In [32]: train['Season']=(train['Month']%12 + 3)//3
         test['Season']=(test['Month']%12 + 3)//3
```

5.3.2.5Weekend

Weekends may have effect on what types of crimes are committed

- Weekday = 0, Weekend =1



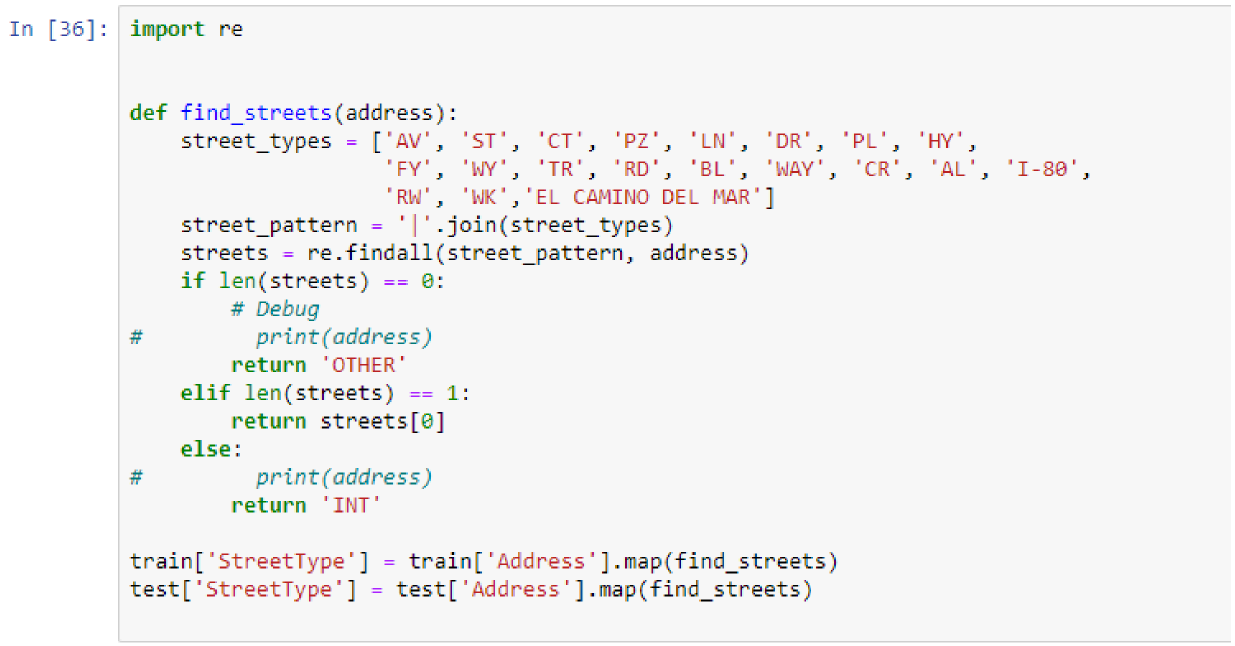
```
In [34]: # Weekend Feature

         # Weekday = 0, Weekend = 1
         days = {'Monday':0 ,'Tuesday':0 ,'Wednesday':0 ,'Thursday':0 ,'Friday':0, 'Saturday':1 ,'Sunday':1}

         train['Weekend'] = train['DayOfWeek'].replace(days).astype('uint8')
         test['Weekend'] = test['DayOfWeek'].replace(days).astype('uint8')
```

5.3.3Spatial Features

5.3.3.1Street Type

The street type can have an effect on what type of crime is committed, so we want to extract the street type from the 'Address' feature.

We have avenues, streets, ways, boulevards, highways, courts, walks, plazas, and differet number of intersections of roads/streets



```
In [36]: import re


         def find_streets(address):
             street_types = ['AV', 'ST', 'CT', 'PZ', 'LN', 'DR', 'PL', 'HY',
                             'FY', 'WY', 'TR', 'RD', 'BL', 'WAY', 'CR', 'AL', 'I-80',
                             'RW', 'WK','EL CAMINO DEL MAR']
             street_pattern = '|'.join(street_types)
             streets = re.findall(street_pattern, address)
             if len(streets) == 0:
                 # Debug
         #         print(address)
                 return 'OTHER'
             elif len(streets) == 1:
                 return streets[0]
             else:
         #         print(address)
                 return 'INT'

         train['StreetType'] = train['Address'].map(find_streets)
         test['StreetType'] = test['Address'].map(find_streets)
```

5.3.3.2Block Number Feature

Let's explore the block number from address

Block number has ordinal data type (order matters), and has spatial significance

It seems all the block numbers are in intervals of 100

How to categorize

- Addresses that do not have a block number will be categorized as 0
- Addresses with block number will be divided by 100, and added by 1 for mapping (0 is saved for addresses with no block number)

85 unique block numbers (including 1 where there is no block number)

Block number features

```
In [ ]: def find_block_number(address):
            block_num_pattern = '[0-9]+\s[Block]'
            block_num = re.search(block_num_pattern, address)
            if block_num:
        #         print(address)
                num_pattern = '[0-9]+'
                block_no_pos = re.search(num_pattern, address)
                # Get integer of found regular expression
                block_no = int(block_no_pos.group())
                # Convert block number by dividing by 100 and adding 1 (0 = addresses with no block)
                block_map = (block_no // 100) + 1
        #         print(block_map)
                return block_map
            else:
        #         print(address)
                #
                return 0


        train['BlockNo'] = train['Address'].map(find_block_number)
        test['BlockNo'] = test['Address'].map(find_block_number)
```

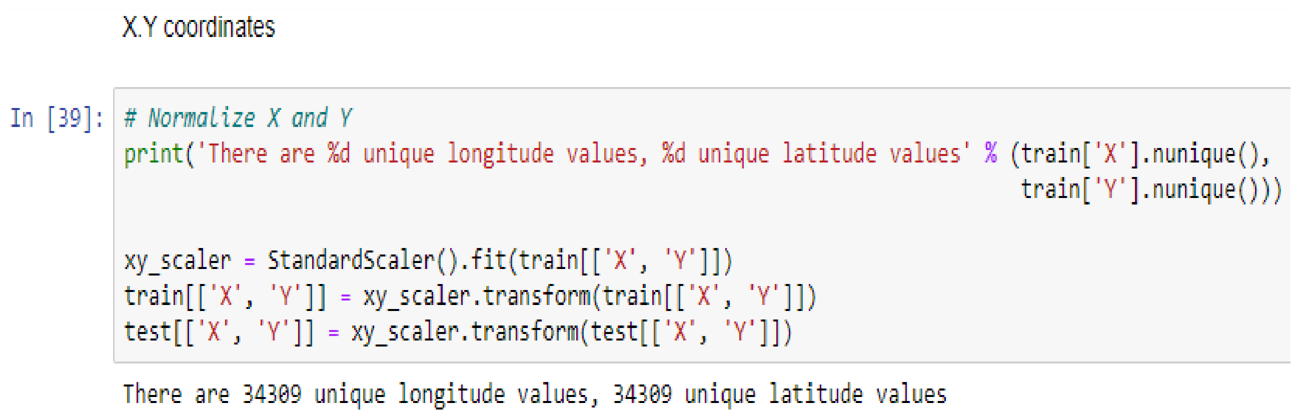
5.3.3.3X, Y Coordinates

Normalize and scale the X and Y coordinates

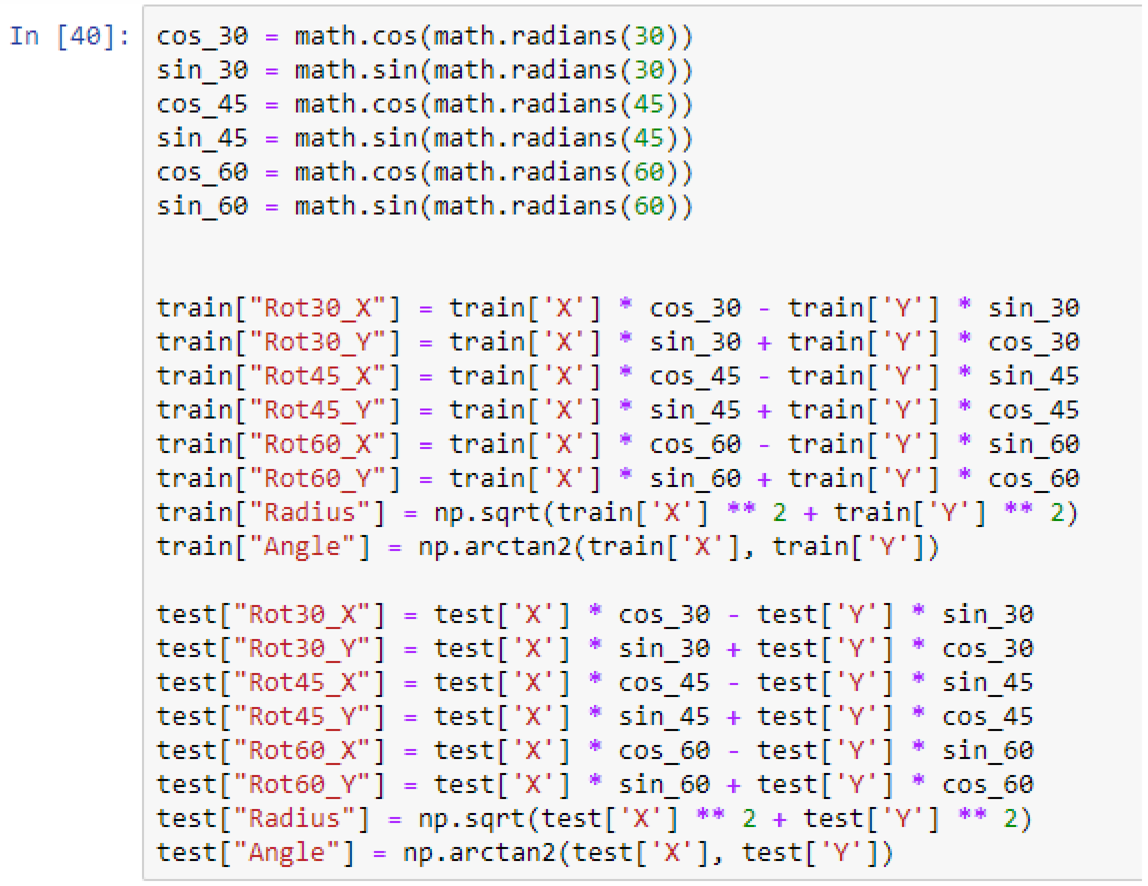
we use **K-Means clustering** to create a new feature for the longitude and latitude by grouping clusters of points based on Euclidean distances.
X = longitude, Y = latitude

we also extract more spatial features from the X, Y coordinates by transforming them from the cartesian space to the polar space

1. three variants of rotated Cartesian coordinates (rotated by 30, 45, 60 degree each)
2. Polar coordinates (i.e. the 'r' and the angle 'theta')
3. The approach makes some intuitive sense i.e. that having such features should help in extracting some more spatial information (than relying on the current x-y alone)


X.Y coordinates

```
In [39]: # Normalize X and Y
         print('There are %d unique longitude values, %d unique latitude values' % (train['X'].nunique(),
                                                                                    train['Y'].nunique()))


         xy_scaler = StandardScaler().fit(train[['X', 'Y']])
         train[['X', 'Y']] = xy_scaler.transform(train[['X', 'Y']])
         test[['X', 'Y']] = xy_scaler.transform(test[['X', 'Y']])

There are 34309 unique longitude values, 34309 unique latitude values
```

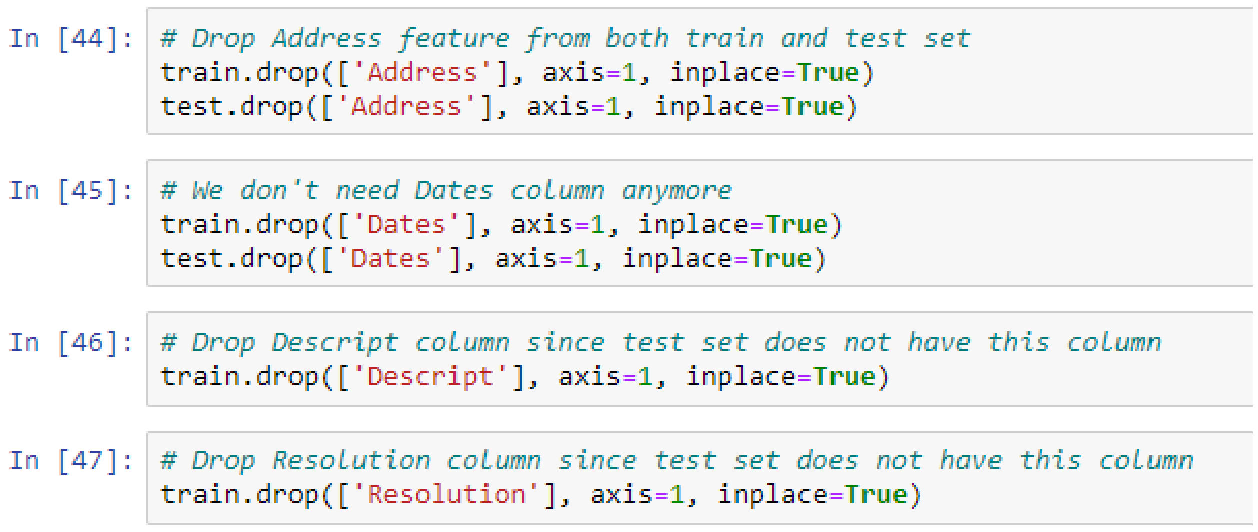


```
In [40]: cos_30 = math.cos(math.radians(30))
         sin_30 = math.sin(math.radians(30))
         cos_45 = math.cos(math.radians(45))
         sin_45 = math.sin(math.radians(45))
         cos_60 = math.cos(math.radians(60))
         sin_60 = math.sin(math.radians(60))


         train["Rot30_X"] = train['X'] * cos_30 - train['Y'] * sin_30
         train["Rot30_Y"] = train['X'] * sin_30 + train['Y'] * cos_30
         train["Rot45_X"] = train['X'] * cos_45 - train['Y'] * sin_45
         train["Rot45_Y"] = train['X'] * sin_45 + train['Y'] * cos_45
         train["Rot60_X"] = train['X'] * cos_60 - train['Y'] * sin_60
         train["Rot60_Y"] = train['X'] * sin_60 + train['Y'] * cos_60
         train["Radius"] = np.sqrt(train['X'] ** 2 + train['Y'] ** 2)
         train["Angle"] = np.arctan2(train['X'], train['Y'])

         test["Rot30_X"] = test['X'] * cos_30 - test['Y'] * sin_30
         test["Rot30_Y"] = test['X'] * sin_30 + test['Y'] * cos_30
         test["Rot45_X"] = test['X'] * cos_45 - test['Y'] * sin_45
         test["Rot45_Y"] = test['X'] * sin_45 + test['Y'] * cos_45
         test["Rot60_X"] = test['X'] * cos_60 - test['Y'] * sin_60
         test["Rot60_Y"] = test['X'] * sin_60 + test['Y'] * cos_60
         test["Radius"] = np.sqrt(test['X'] ** 2 + test['Y'] ** 2)
         test["Angle"] = np.arctan2(test['X'], test['Y'])
```

5.3.4 Drop Features

- We have already extracted all the necessary features from the `Address` attribute, so drop
- We don't need `Resolution` or `Descript` features since it is not included in the training data

Drop features



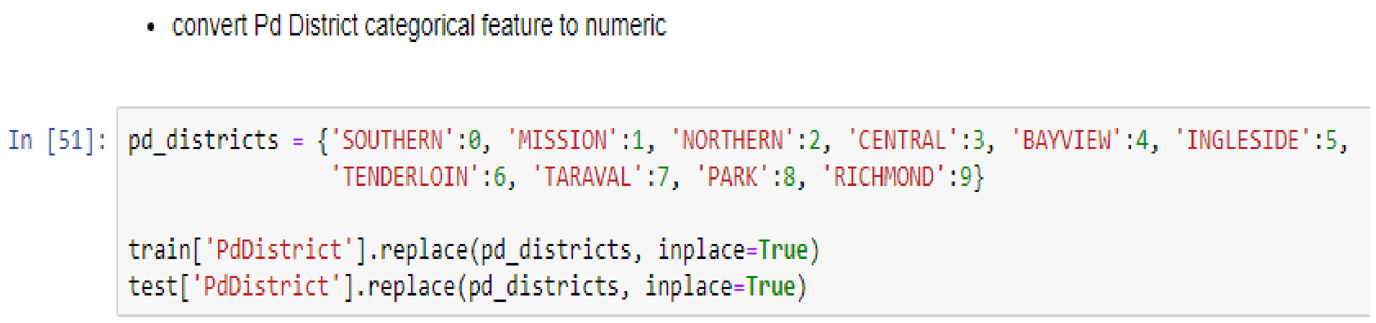
```
In [44]: # Drop Address feature from both train and test set
         train.drop(['Address'], axis=1, inplace=True)
         test.drop(['Address'], axis=1, inplace=True)

In [45]: # We don't need Dates column anymore
         train.drop(['Dates'], axis=1, inplace=True)
         test.drop(['Dates'], axis=1, inplace=True)

In [46]: # Drop Descript column since test set does not have this column
         train.drop(['Descript'], axis=1, inplace=True)

In [47]: # Drop Resolution column since test set does not have this column
         train.drop(['Resolution'], axis=1, inplace=True)
```

5.3.5Feature Encoding

Convert categorical data to numeric data

5.3.5.1Pd Districts

convert Pd District categorical feature to numeric



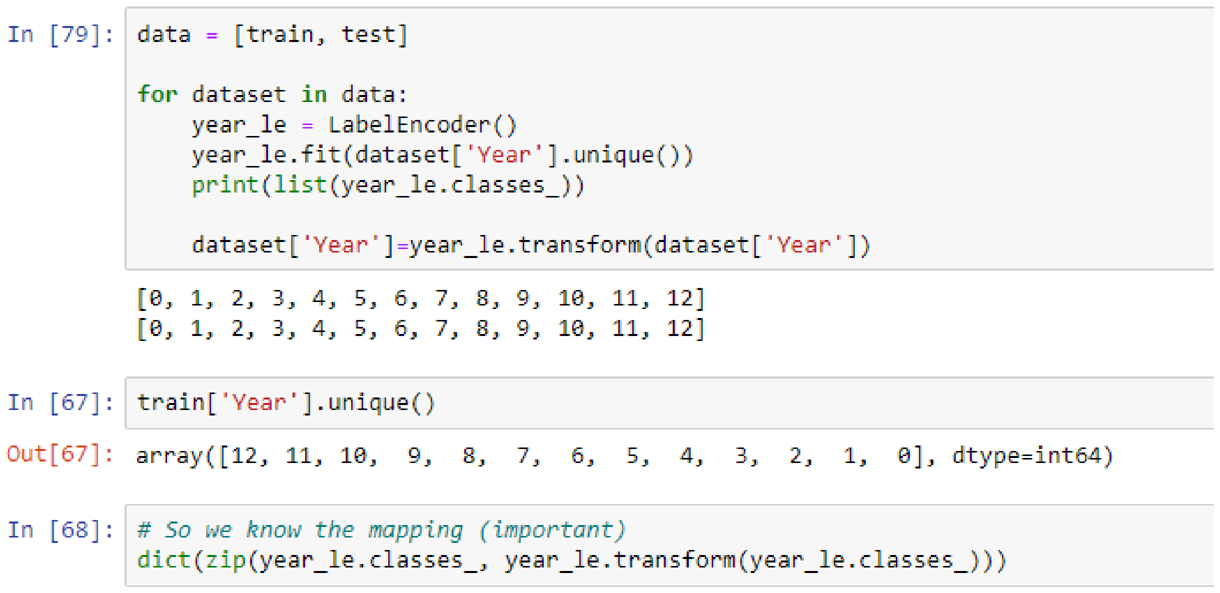
- convert Pd District categorical feature to numeric

```
In [51]: pd_districts = {'SOUTHERN':0, 'MISSION':1, 'NORTHERN':2, 'CENTRAL':3, 'BAYVIEW':4, 'INGLESIDE':5,
                         'TENDERLOIN':6, 'TARAVAL':7, 'PARK':8, 'RICHMOND':9}

         train['PdDistrict'].replace(pd_districts, inplace=True)
         test['PdDistrict'].replace(pd_districts, inplace=True)
```

5.3.5.2**Year**

Year is an **ordinal** variable, so let's keep that ordering and mapping

convert Year categorical feature to numeric

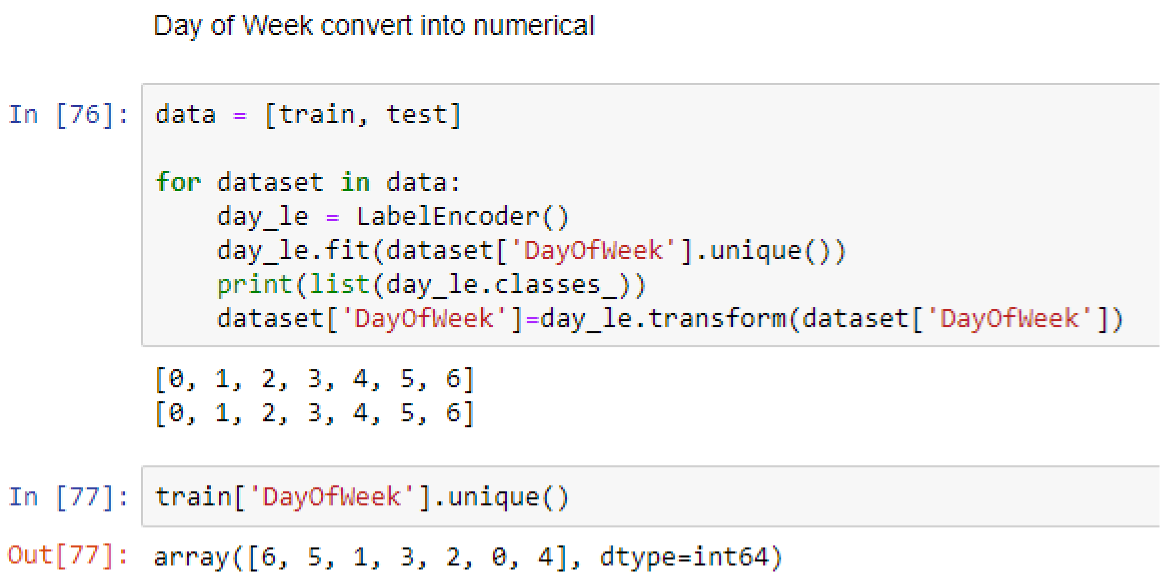


```
In [79]: data = [train, test]

         for dataset in data:
             year_le = LabelEncoder()
             year_le.fit(dataset['Year'].unique())
             print(list(year_le.classes_))

             dataset['Year']=year_le.transform(dataset['Year'])

         [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
         [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

In [67]: train['Year'].unique()

Out[67]: array([12, 11, 10,  9,  8,  7,  6,  5,  4,  3,  2,  1,  0], dtype=int64)

In [68]: # So we know the mapping (important)
         dict(zip(year_le.classes_, year_le.transform(year_le.classes_)))
```

5.3.5.3DayOfWeek

we are going to use sklearn's LabelEncoder to encode the categorical data to numeric

Day of week is considered a categorical and nominal variable



Day of Week convert into numerical

```
In [76]: data = [train, test]

         for dataset in data:
             day_le = LabelEncoder()
             day_le.fit(dataset['DayOfWeek'].unique())
             print(list(day_le.classes_))
             dataset['DayOfWeek']=day_le.transform(dataset['DayOfWeek'])

         [0, 1, 2, 3, 4, 5, 6]
         [0, 1, 2, 3, 4, 5, 6]

In [77]: train['DayOfWeek'].unique()

Out[77]: array([6, 5, 1, 3, 2, 0, 4], dtype=int64)
```

We also apply label encoder on Holiday, category, street type

5.3 Building Machine Learning Models

Baseline Models

Let's train a couple models on a stratified sample of the training data
Evaluate on a hold out set to get baseline results for each model to determine what model to use

- Models:
  - Stochastic Gradient Descent (with elastic net regularization)
  - Gaussian Naive Bayes
  - K Nearest Neighbors
  - Logistic Regression (with L1 regularization)
  - Random Forest
  - XGBoost

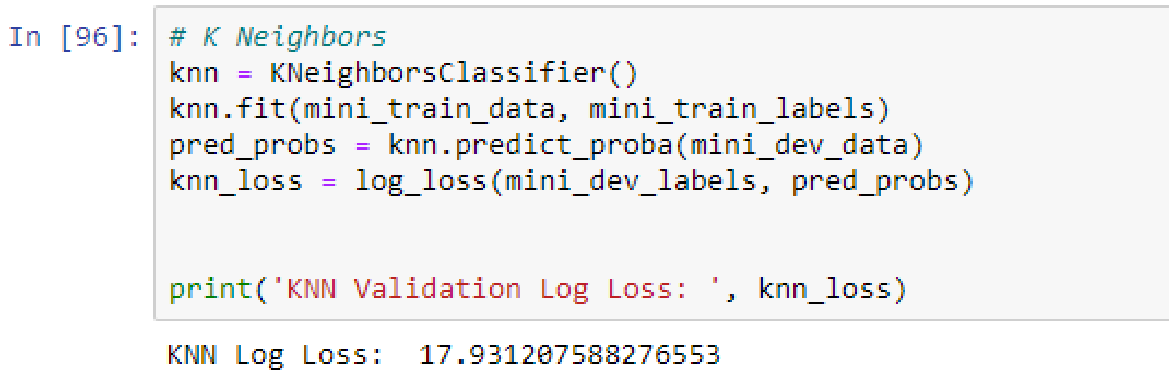
-Almost all the default scikit-learn ML algorithm hyperparameters exhibit bad performance

  - Researched online & read literature to determine some more ideal default hyperparameters

Couple things to note:

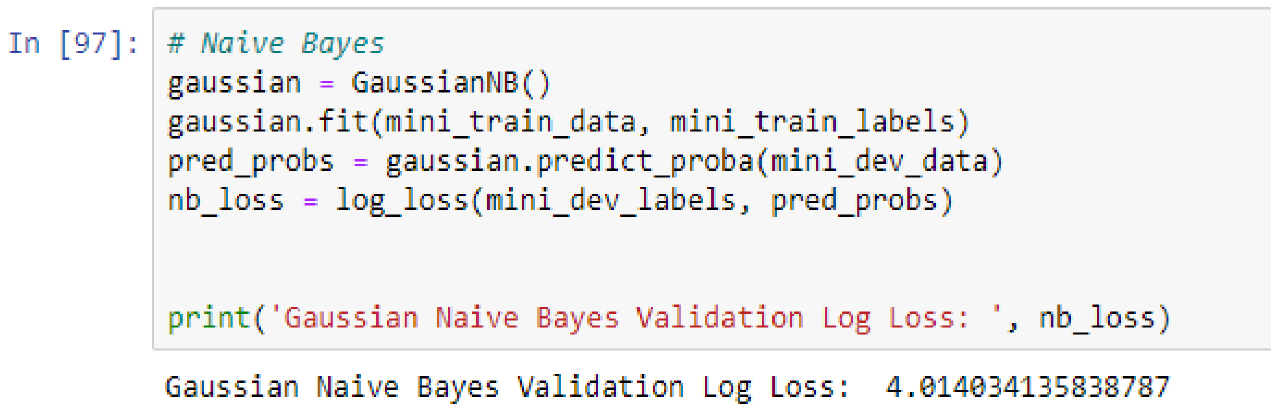
-Decision tree models including Ensemble methods (Random Forest & XGBoost) can handle categorical variables without one-hot encoding them.
-Linear models (SGD & Logistic Regression) cannot handle categorical features & need features to be OHE before training
Always OneHotEncode before you split data up to training/dev/test so that all features & classes will be represented

K-neighbors



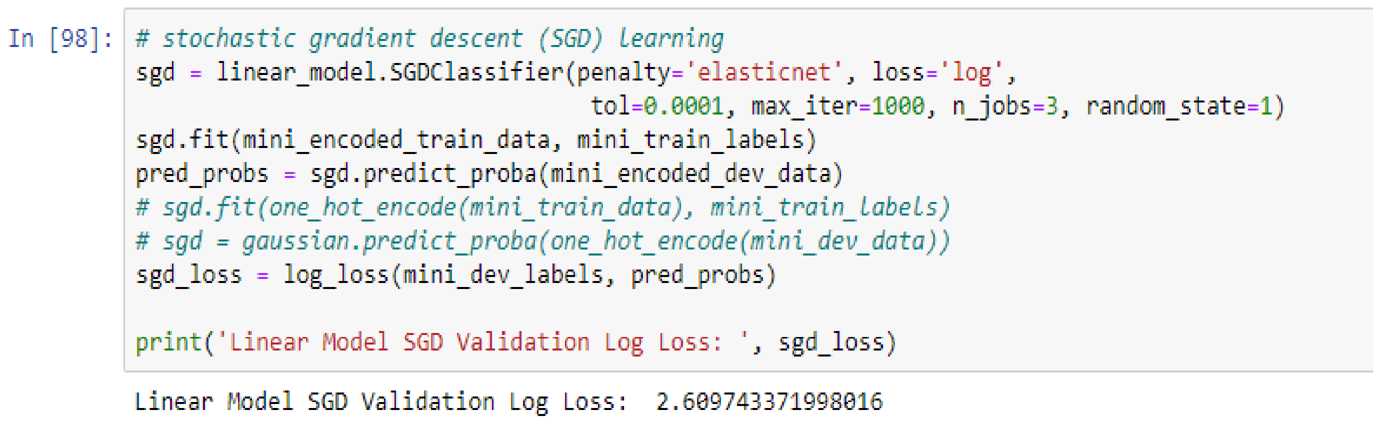
```
In [96]: # K Neighbors
         knn = KNeighborsClassifier()
         knn.fit(mini_train_data, mini_train_labels)
         pred_probs = knn.predict_proba(mini_dev_data)
         knn_loss = log_loss(mini_dev_labels, pred_probs)


         print('KNN Validation Log Loss: ', knn_loss)

         KNN Log Loss:  17.931207588276553
```

Naïve Bayes



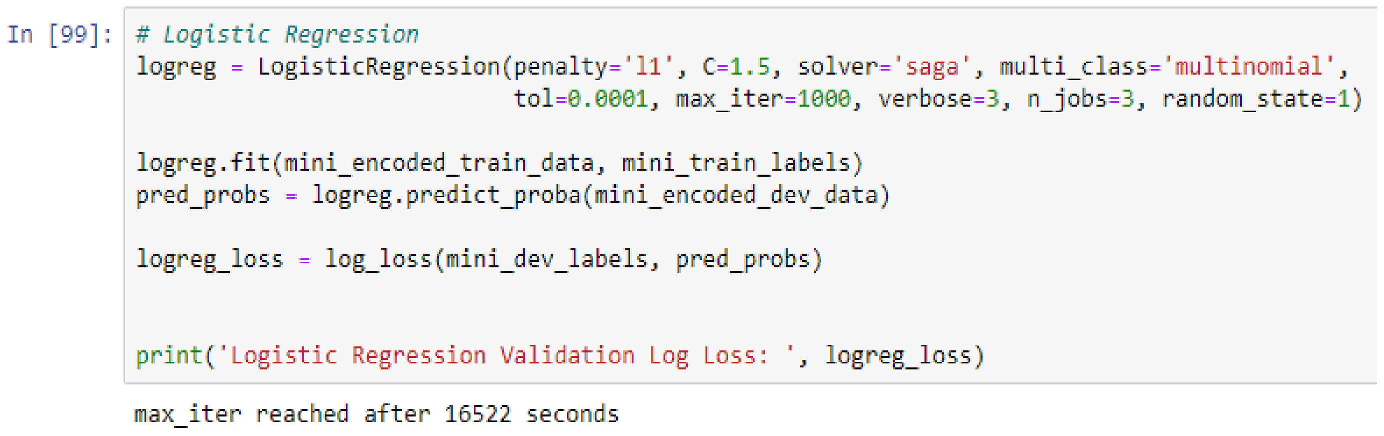
```
In [97]: # Naive Bayes
         gaussian = GaussianNB()
         gaussian.fit(mini_train_data, mini_train_labels)
         pred_probs = gaussian.predict_proba(mini_dev_data)
         nb_loss = log_loss(mini_dev_labels, pred_probs)


         print('Gaussian Naive Bayes Validation Log Loss: ', nb_loss)

         Gaussian Naive Bayes Validation Log Loss:  4.014034135838787
```

stochastic gradient descent (SGD) learning



```
In [98]: # stochastic gradient descent (SGD) learning
         sgd = linear_model.SGDClassifier(penalty='elasticnet', loss='log',
                                           tol=0.0001, max_iter=1000, n_jobs=3, random_state=1)
         sgd.fit(mini_encoded_train_data, mini_train_labels)
         pred_probs = sgd.predict_proba(mini_encoded_dev_data)
         # sgd.fit(one_hot_encode(mini_train_data), mini_train_labels)
         # sgd = gaussian.predict_proba(one_hot_encode(mini_dev_data))
         sgd_loss = log_loss(mini_dev_labels, pred_probs)

         print('Linear Model SGD Validation Log Loss: ', sgd_loss)

         Linear Model SGD Validation Log Loss:  2.609743371998016
```

Logistic Regression



```
In [99]: # Logistic Regression
         logreg = LogisticRegression(penalty='l1', C=1.5, solver='saga', multi_class='multinomial',
                                     tol=0.0001, max_iter=1000, verbose=3, n_jobs=3, random_state=1)

         logreg.fit(mini_encoded_train_data, mini_train_labels)
         pred_probs = logreg.predict_proba(mini_encoded_dev_data)

         logreg_loss = log_loss(mini_dev_labels, pred_probs)


         print('Logistic Regression Validation Log Loss: ', logreg_loss)

         max_iter reached after 16522 seconds
```

Random Forest Ensemble

```
In [100]: # Random Forest Ensemble
          random_forest = RandomForestClassifier(n_estimators=500, max_depth=15, max_features='sqrt',
                                                 min_samples_leaf=5, min_samples_split=25,
                                                 random_state=1, verbose=1, n_jobs=2)


          random_forest.fit(mini_train_data, mini_train_labels)
          pred_probs = random_forest.predict_proba(mini_dev_data)

          rf_loss = log_loss(mini_dev_labels, pred_probs)

          print('Random Forest Validation Log Loss: ', rf_loss)
```

```
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:   29.2s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:  2.1min
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:  4.6min
[Parallel(n_jobs=2)]: Done 500 out of 500 | elapsed:  5.1min finished
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    5.4s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:   22.0s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:   49.8s
[Parallel(n_jobs=2)]: Done 500 out of 500 | elapsed:   55.9s finished
```

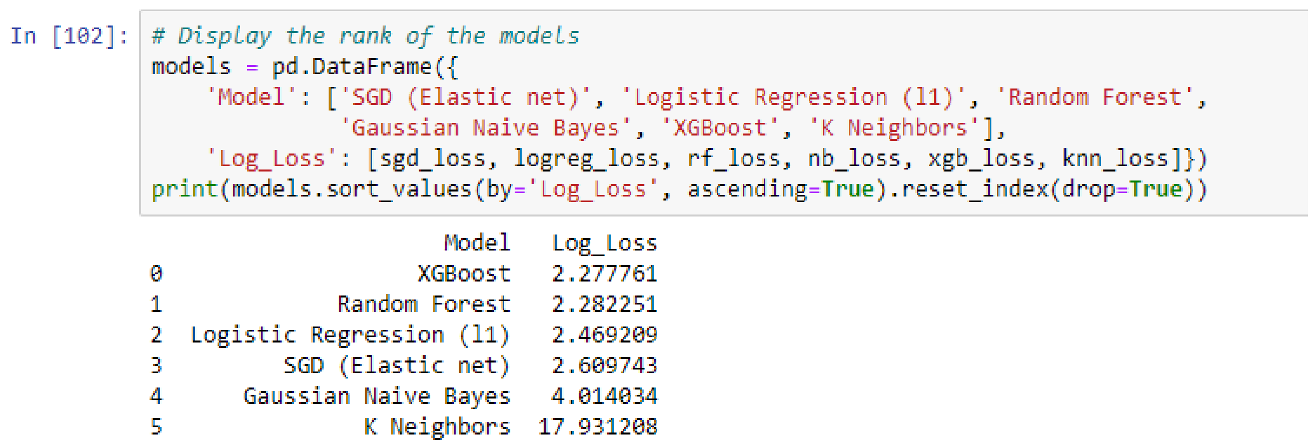
```
Random Forest Validation Log Loss:  2.2822514377263325
```

XGBoost Ensemble

```
In [101]: # XGBoost Ensemble
          # xgb = XGBClassifier(n_estimators=100, verbose=3, n_jobs=2, random_state=1)
          xgb = XGBClassifier(n_estimators=500, objective="multi:softprob",
                              verbose=3, n_jobs=3, random_state=1)

          xgb.fit(mini_encoded_train_data, mini_train_labels)
          pred_probs = xgb.predict_proba(mini_encoded_dev_data)

          xgb_loss = log_loss(mini_dev_labels, pred_probs)

          print('XGBoost Validation Log Loss: ', xgb_loss)
```

```
XGBoost Validation Log Loss:  2.2777614366851147
```

The result of the models are:



```
In [102]: # Display the rank of the models
          models = pd.DataFrame({
              'Model': ['SGD (Elastic net)', 'Logistic Regression (l1)', 'Random Forest',
                        'Gaussian Naive Bayes', 'XGBoost', 'K Neighbors'],
              'Log_Loss': [sgd_loss, logreg_loss, rf_loss, nb_loss, xgb_loss, knn_loss]})
          print(models.sort_values(by='Log_Loss', ascending=True).reset_index(drop=True))

                        Model   Log_Loss
          0               XGBoost   2.277761
          1         Random Forest   2.282251
          2  Logistic Regression (l1)   2.469209
          3     SGD (Elastic net)   2.609743
          4  Gaussian Naive Bayes   4.014034
          5           K Neighbors  17.931208
```

5.4Model Selection

Although Logistic Regression with L1 regularization seems promising, our dataset has a mixture of categorical and numerical features that have very different statistics (mean, variance), thus not very linear. In addition, with any linear model, this would require **one hot encoding** that would greatly increase the feature space (some categorical features such as `BlockNumber` have many levels/values).

- Logistic Regression is a generalized linear model, and can theoretically only solve problems where the classes are linearly separable & features are linear.
- In practice, if we do more feature engineering and convert the non-linear features to linear features, we could increase the performance of LR

Ensemble methods have been historically and theoretically powerful in handling datasets with very different features (numerical & categorical features). In addition, ensemble methods are effective in solving non-linear problems. So, we will select between Random Forest & XGBoost as the final model.

We select XGboost which have the lowest **logloss value which is 2.27** and have higher accuracy.

```
In [129]: # It seems running time scales quadratically with the number of classes
          xgb = XGBClassifier(
              n_estimators=86,
              objective="multi:softprob",
              learning_rate=0.1858621466840661,
              colsample_bylevel=1.0,
              colsample_bytree=1.0,
              gamma=0.49999999999999994,
              max_delta_step=0,
              max_depth=50,
              min_child_weight=5,
              reg_alpha=1.0,
              reg_lambda=60.121460571845695,
              scale_pos_weight=1e-06,
              subsample=1.0,
              random_state=1,
              n_jobs=4,
              silent=False)


          xgb.fit(X_train, Y_train)

          Y_test_pred = xgb.predict_proba(X_test)
```

CHAPTER 6

Conclusion

This project has taught us a lot about data science and has given us hands-on experience with working with data and completing an end-to-end data science project. we've had a lot of fun visualizing, analyzing, and experimenting with the data to gain more insight. This is just the beginning of our journey into data science, and we are very excited to see what the future holds in terms of new and interesting data science problems and datasets.

- **What we learned**:
    - There are more efficient ways to label or integer encode features
        - Will use sklearn's LabelEncoder, OneHotEncoder, & MultiLabelBinarizer next time
    - Instead of just blindly training models, research more about ways to optimize the hyperparameters efficiently
        - Spent too many AWS EC2 hours with `GridSearchCV`, when we should have used *Bayesian Optimization* for efficient hyperparameter tuning
        - Do more research on the domain of the problem, certain core ML algorithms, and data processing techniques
- **What's next?**
    - AutoML with `tpot` or `auto-sklearn`
        - automate the hyperparameter tuning and model selection with AutoML packages
    - Problem Redirection (Classification ---> Regression)
        - Instead of predicting category of crime, predict X & Y coordinates (longitude & latitude) continuous values given same spatial and temporal features as well as category of crime
        - **Use case:** Dynamically concentrate police on certain serious categories of crime to prevent crimes from happening beforehand

LIST OF REFERENCES

.1.DataSF, “Open government,”https://www.data.gov/open-gov/, accessed 2015-04-12.

2.City and C. of San Francisco, “Police Department Incidents,”https://data.sfgov.org/Public-Safety/Police-Department-Incidents/tmnf-yvry/, 2017, [Online; accessed 2017-09-12].

3.B. Kolo, *Binary and Multiclass Classification*.Lulu. com, 2011.

4.D. Jurafsky and J. H. Martin, “Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recogni- tion,” pp. 1–1024, 2009.

5.A. Liaw, M. Wiener, *et al.*, “Classification and regression by randomforest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.

6.F. Livingston, “Implementation of breiman’s random forest machine learning algorithm,” *ECE591Q Machine Learning Journal Paper*, 2005.

7.E. Andrew B. Collier, “Making Sense of Logarithmic Loss,”http://www.exegetic.biz/blog/2015/12/making-sense-logarithmic-loss/, 2015, [Online; accessed 2018-19-04].

8.T. Almanie, R. Mirza, and E. Lor, “Crime prediction based on crime types and using spatial and temporal criminal hotspots,” *arXiv preprint arXiv:1508.02050*, 2015.

9.Y.A bouelnaga,“San francisco crime classification,” *arXiv preprint arXiv:1607.03626*, 2016.

10.G. H. Larios, “Case study report: San francisco crime classification,”https://gabrielahrlr.github.io/personal-website/docs/SF_crimes.pdf, 2016, [Online; accessed 2018-24-04].