**Boston Crime Analysis and Prediction**

**1. Summary**

Open security is imperative to general wellbeing and bliss, and a city's safety can be an urgent factor in choosing where to study, work and live. It is reported that the city violent crime rate for Boston in 2016 was higher than the national violent crime rate average by 12.27% [1]. As International students pursuing master's degree in Boston, to study well while living here, we would like to know how safe the city is, which kind of and how frequently crime is going on in our neighborhood.

Along these lines, the objective of this project is to study the distribution base on the crime incidents of the recent five years to help address the crime and find a pattern or prediction if possible. By providing a model to determine the most criminal hotspots and discover the sort, area and time of committed crimes, we need to raise individuals' mindfulness regarding the perilous areas in certain timeframes. On the other hand, police forces can utilize this result to increment the accuracy of crime prediction in order to have police resources allocations with higher efficiency.

The data set about crime incident reports are provided by the Boston Police Department, which containing records focused on capturing the type of incident as well as when and where it occurred. Records time range covers from 2015-06-15 to 2019-11-20. The data set has 440606 rows, consisting of 7 numerical variables (offense\_code, reporting\_area, year, month, hour, lat, long) and 10 categorical variables (incident\_number, offense\_code\_group, offense\_description, district, shooting, occurred\_on\_date, day\_of\_week, ucr\_part, street, location). Among them, offense\_code\_group, reporting\_area, ucr\_part, incident\_number and location are not what we are going to focus on, so we removed them before the analysis.

In this project, large dataset has been reviewed and information such as if with shooting, time, location and the type of crimes have been extracted and plot to help people discover the crime pattern in Boston city. Then we used three basic classifier models, Random Forest Tree, Naïve Bayesian and Support Vector Machine, to anticipate potential crime types based on time-related and location-related features. After we trained our model, we compared each model’s performance by different matrices due to imbalanced data. However, we found that the performance was not satisfied, so we came up with another dataset which included more features maybe would be better for prediction. Models execution were compared based on cross-validation accuracy and the prediction outcome was reported.

**2. Methods**

We strongly believe that finding relationships between crime elements could exceptionally help in foreseeing potential dangerous hotspots at a specific time in the future. Therefore, our proposed approach aimed to focus on three primary components of crime data, which are the type of crime, the occurrence time and the crime location. We attempted to extract all conceivable interesting frequent patterns based on the crime factors. At that point, we applied some classification methods in order to predict potential crime types in a specific location within a particular time.

In this section, we explain how we prepared our datasets. From that point onward, we provide how we analyzed the data using some statistical analysis. Then we introduce how we constructed our prediction models to accomplish our motivation.

2.1 Data preprocessing

2.1.1 Data Cleaning

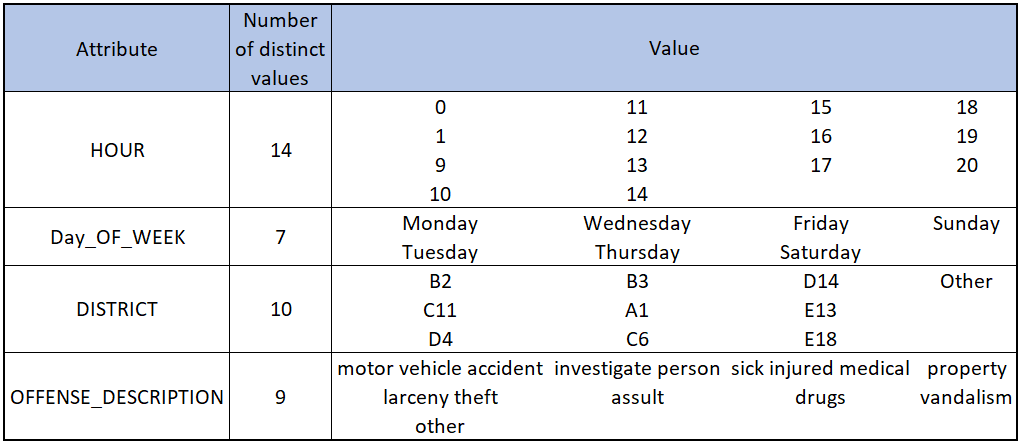
There are some missing values in some attributes. Among them, SHOOTING, Lat, Long, and STREET have the most missing values. There are 394617 cases (89.6%) where SHOOTING has missing values; 17256 cases (3.9%) where SHOOTING, Lat, Long have missing value together; 12297 cases (2.8%) where Lat and Long have missing value together; 9755 cases (2.2%) where STREET, Lat, Long, and SHOOTING have missing value together. However, we found that the 4 attributes containing most missing values are not our key attributes. The SHOOTING column only contains 1 and NA before 2019-09-29. It also only contains 1 and 0 after that date. Consequently, we could assume to have 0 replace all NA values in the SHOOTING column. Then the proportion of missing values in this data set is acceptable. All other key attributes like year, month, hour, lat, long, district, day\_of\_week were completed with tidy values in the datasets.

2.1.2 Data Transformation

In the dataset, four attributes are generated by occurred\_on\_date attribute: year, month, hour and day\_of\_week. Prior to the data transformation, we had 5 diverse distinct values for the year attribute, 12 types for the month attribute, 24 types for the hour attribute and 7 types for the day\_of\_week attribute. For the offense\_description attribute, we have 84 different distinct values. For the district attribute, we have 13 type of values. We realized that it is necessary to reduce the diversity of these attributes’ values. Along these lines, we applied data transformation to these attributes by mapping their values to fall inside littler gatherings. Our objective was to get more frequent patterns and to increase the model accuracy.

We characterize the numbers of cases less than 5% of the overall 440606 cases (≈20000) as less frequent categories for that attribute. For the offense\_description attribute, we firstly binned similar minor offense categories into major offense categories, then only kept the most frequent major categories and binned less frequent ones into the category "Other". Therefore, we have 8 major offense categories for analysis and modeling. For the hour attribute, we mapped its less frequent ones into the categorical value "1". For the district attributes, we mapped its less incessant ones into the category “Other”. Table 1 illustrates the resulted of attributes after data preprocessing.

Table 1. Resulted attributes after data preprocessing



2.2 Data Analysis

As an initial step to analyze and get the big view of our data, we conducted statistical analysis on the attribute values of our datasets. We started with generating a R script to calculate frequencies of distinct values for every attribute. Then we created a variety of graphs to give us better understanding of our data. Each graph came up with the aggregated number of crime occurrences regarding a particular aspect.

2.3 Model building

In order to extract frequent patterns from Boston crimes, we applied the Support Vector Machine,Naïve Bayesian classifier and Random Forest classifier to build different multi-class classification models for the dataset. The motivation behind the classifiers is to predict the potential crime type in a specific location within a particular time in the future. We aimed to examine every model then choose the model that gives the best accuracy in prediction. In this section, we provide a short depiction of how we address this prediction task sequentially.

At first, we applied the machine learning algorithms mentioned before to build 5 multi-class classification models on the processed dataset with 38 features. For Naïve Bayesian classifiers, we considered Gaussian Naïve Bayes, Bernoulli Naïve Bayes and Multinomial Naïve Bayes to explore how do they perform on our dataset under different assumptions. However, we did not acquire a desirable performance on the models we created on both the training set and test set. Chances are that we under fitted our dataset. In other words, we made our models too sample to capture the underlying pattern. Therefore, we added more features in the next step. Considering the computational cost and training efficiency, we only trained a multi-class classification Random Forest on new dataset with 1212 features. Nevertheless, the performance is not satisfied, either. Since a high accuracy on such a 9 classes classification task is challenging and we found out that the prediction of type “Other” crime category is more accurate than others, we decided to build a one-vs-others binary classification model to predict whether the crime is an “Other” type or not. Disappointingly, we still did not get acceptable results. To optimize our model, we tended to Principal Component Analysis to see if we could gain an improvement by dimension reduction techniques. During the visualization and analysis of our data after applying PCA, we concluded the true reason why our models did not perform well in this case.

**3 Results**

3.1 Time related analysis

Figure 1-4 provide a statistical comparison between with or without crime incidents in different time intervals. In Boston, the crime with shooting and the crime without shooting are positively correlated. The hour of 17, the month of august, the Friday and the year of 2017 have the maximum number of crimes in their respective comparison scope. On the other hand, the hour of 5, the month of February, the Sunday and the year of 2015 have the least number of crimes in Boston. (comment: the data of 2015 and 2019 do not cover the whole years.)

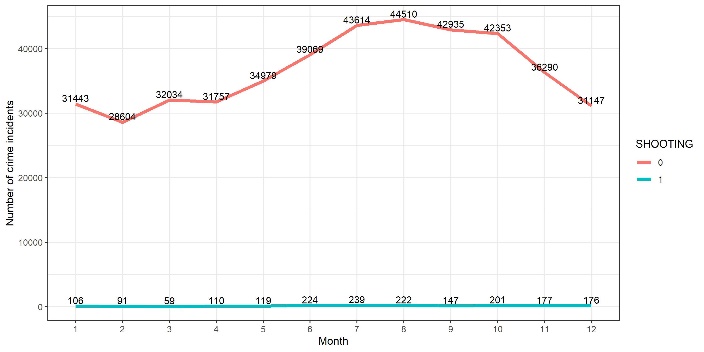
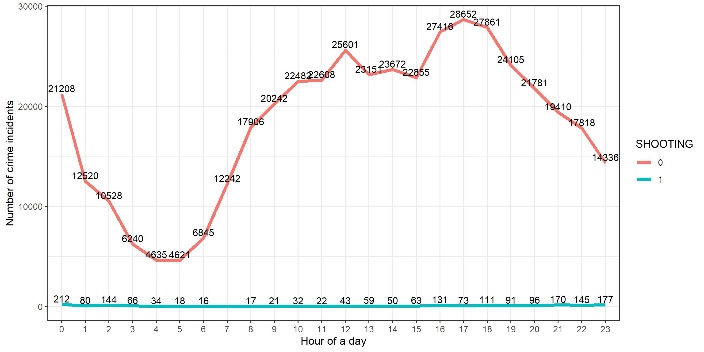


Figure 1. Crimes over the 24 hours based on if shooting Figure 2. Crimes over the 12 months based on if shooting

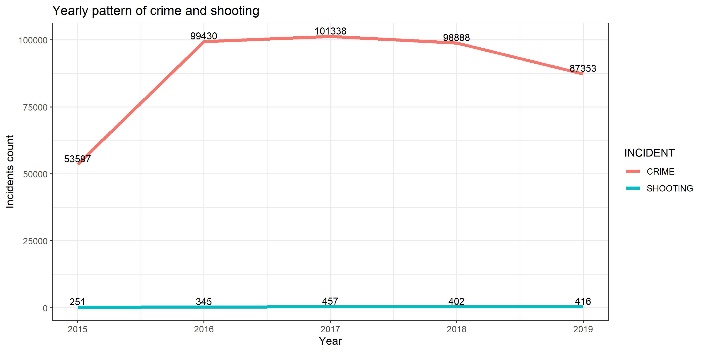
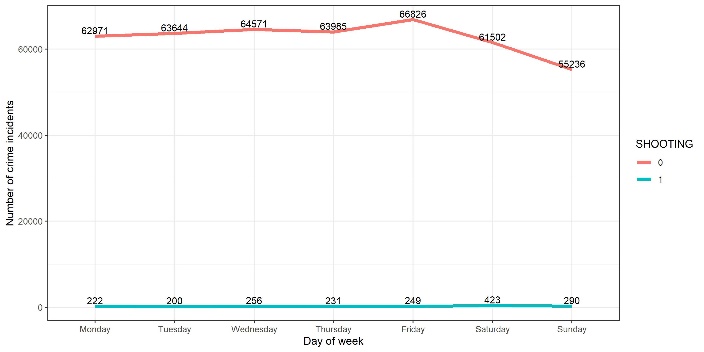


Figure 3. Crimes over the weekdays based on if shooting Figure 4. Crimes over the 5 years based on if shooting

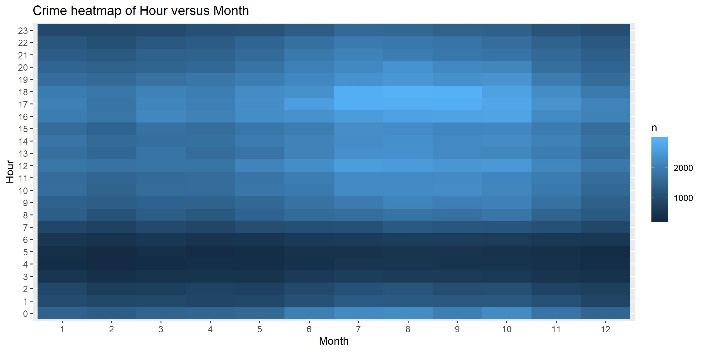
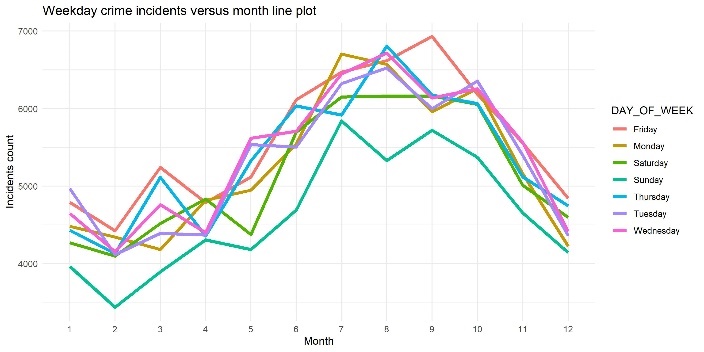


Figure 5. Crimes over the weekdays in 12 months Figure 6. Crimes over the 24 hours in 12 months

Figure 5-6 shows the relationship of crimes and 2 time-related attributes. We can see the Friday has higher crime frequency and reach its peak in September, while the Thursday has lower crime frequency and touch the trough in February. From July to September, the 17-19 time slots have relative higher crime frequency, while the hour between 3-6 am have lower crime frequency over the 12 months. Figure 7 shows the hour of 0 am in the weekend have larger number of crimes than workdays. Also, the hourly crime patterns are almost same in both weekend and workdays: low crime frequency in the early morning and high crime frequency in the later afternoon.

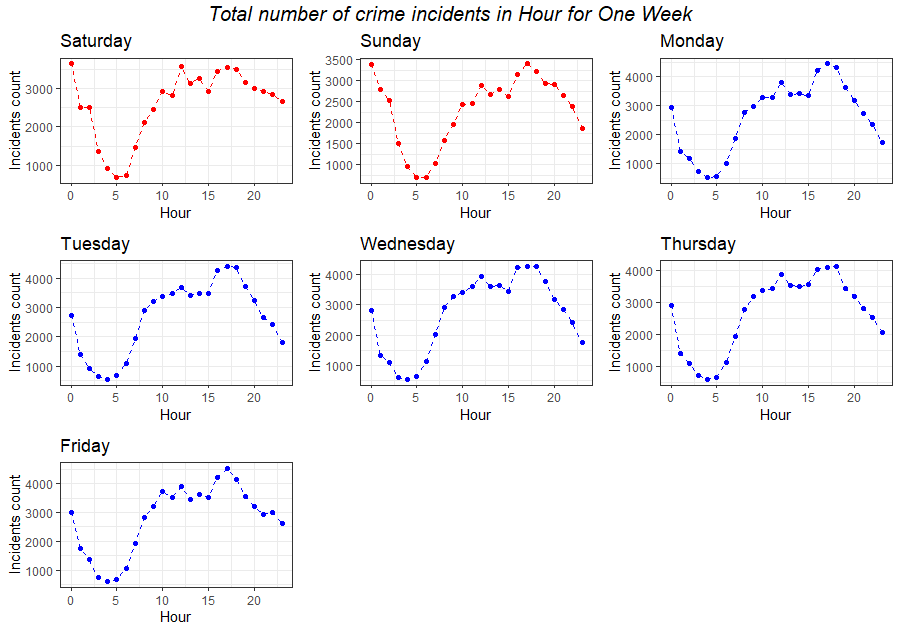
 

Figure 7. Crimes over 24 hours in 7 weekdays Figure 8. Crimes description word cloud

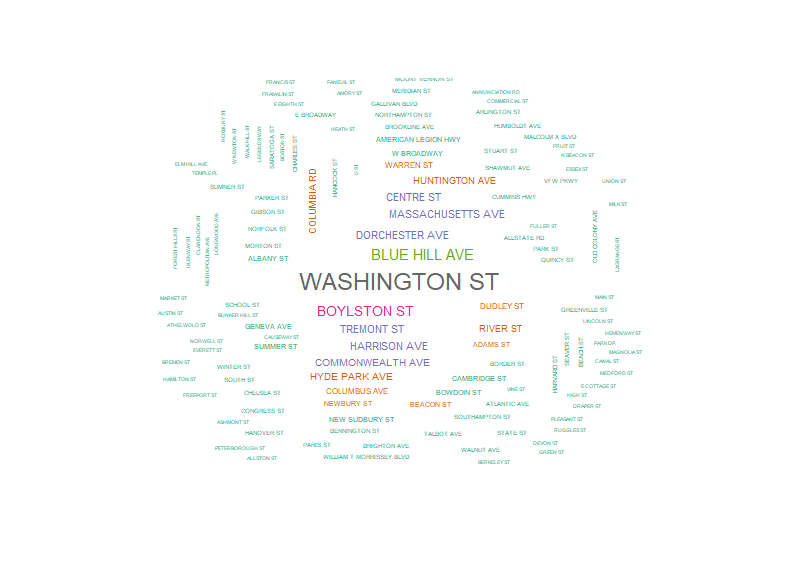
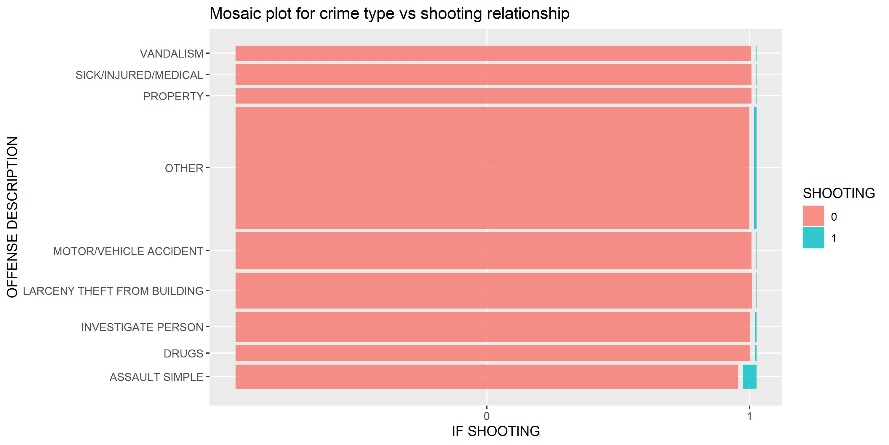


Figure 9. Mosaic plot for crime type and shooting Figure 10. Street name of crimes word cloud

3.2 Crime type analysis

In figure 8 wordcloud, the greater the term is, the larger percentage of the contributing word is in leading to crimes in Boston. The top3 contributing words are ‘property’, ‘motor vehicle’ and ‘person’. Figure 9 shows the proportion of crime frequencies based on the combination of SHOOTING and crime type. The heights of the boxes are proportional to the percentage of crime type. The crime type contains most of crime incidents is “OTHER”, then is “MOTOR VEHICLE ACCIDENT” and “LARCENY THEFT FROM BUILDING”. The widths of the boxes are proportional to the percentage of with or without SHOOTING. The “ASSAULT” with highest proportion of SHOOTING, then is the “OTHER” type.

3.3 Location related analysis

Figure 10 represent the most dangerous streets in Boston. Washington St. appears to be the most perilous street in Boston, however there are Washington streets in multiple districts, so that is a limitation of this word cloud. Figure 11-12 shows hot spots of crime locations. In figure 11, Downtown crossing, Back bay and the junction of South Boston, South end and Roxbury are dangerous. In figure 12, each crime type has higher frequencies on the north of district Downtown and South Boston. The most secure area in Boston is the junction of Jamaica Plain, Roxbury, Mattapan and Hyde Park where has very low residential rate (park, zoo, cemeteries area). The crime type of “VANDALISM” has lower frequencies and is evenly distributed in the Boston city and the crime type of “DRUGS” and “PROPERTY” distribute over a smaller zone.

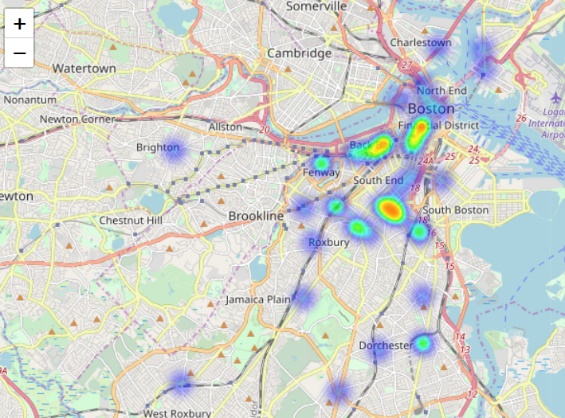
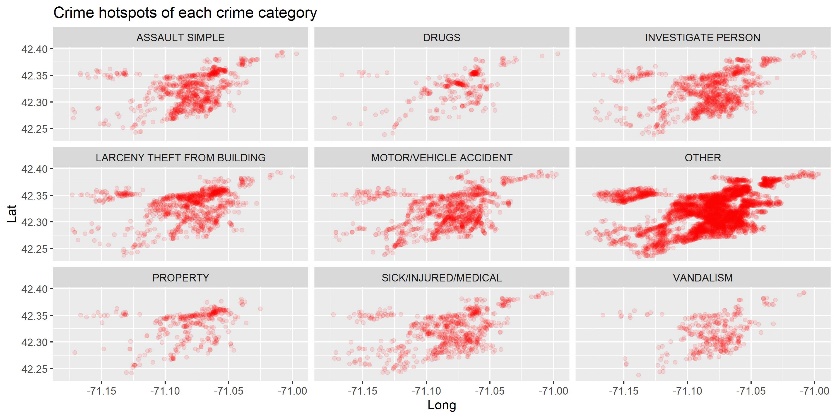
 

Figure 11. Hot spots of top 50 crime locations Figure 12. Hot spots of each crime type in Boston

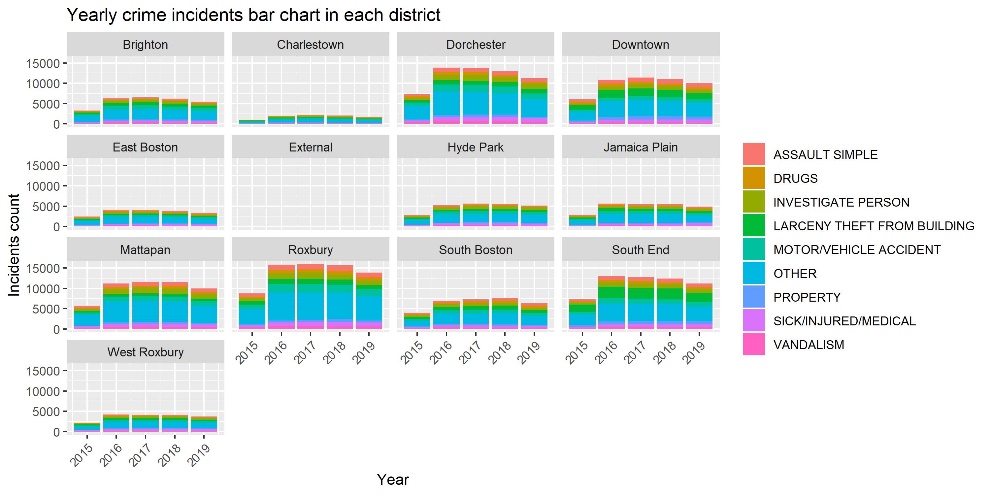


Figure 13. Yearly crimes pattern of each crime type in each district

Figure 13 shows the yearly crime trends of different crime type in each district. (comment: the data of 2015 and 2019 do not cover the whole years.) The crime trend in each district has small difference between each other. We can likewise observe various proportions of crime types based on district. The commercial district “Downtown” and “South End” have more “Larceny theft” crimes, while the residential district “Roxbury” and “Dorchester” have higher “Motor vehicle accident.

3.4 Model evaluation

In this section, we evaluated each of the constructed models regarding different aspects and explained how we did the modeling procedure step by step. Due to our goal is multi-classification so we should pay more attention to imbalanced label data. The figure 14 shows the class frequency of each crime type.

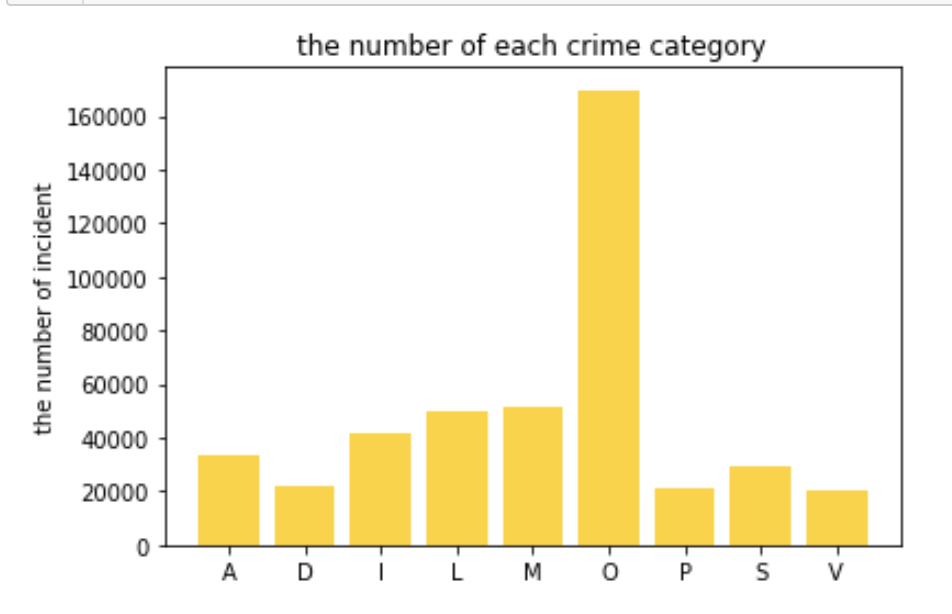


Figure 14. Class frequency of each crime type

As we could see from this distribution plot, the number of incidents in each crime category is not evenly distributed, and the O type is the one with the most number. To address the imbalanced issue, we utilized stratified cross validation strategy during model training to mimic the actual distribution and make sure that we do not introduce bias during train-test split.

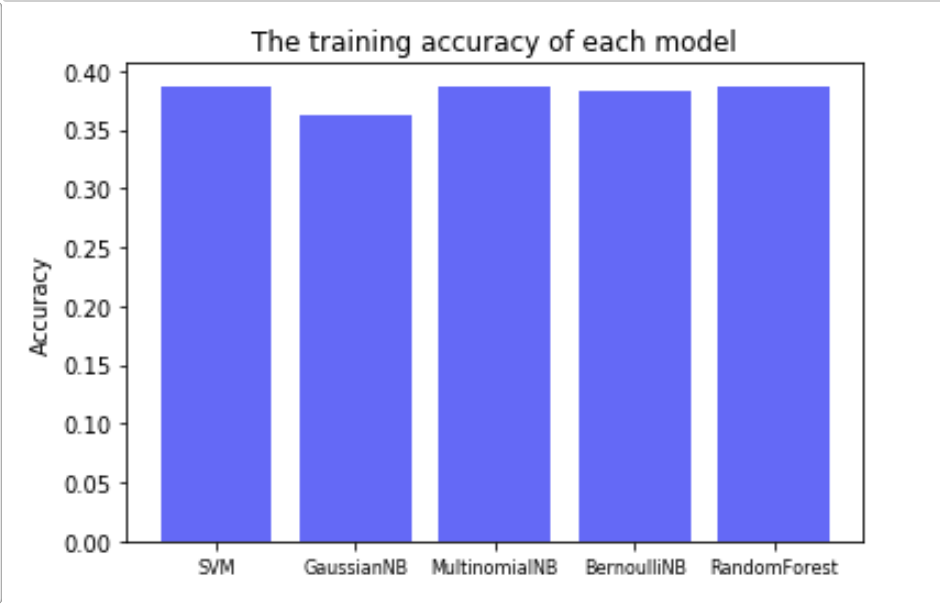
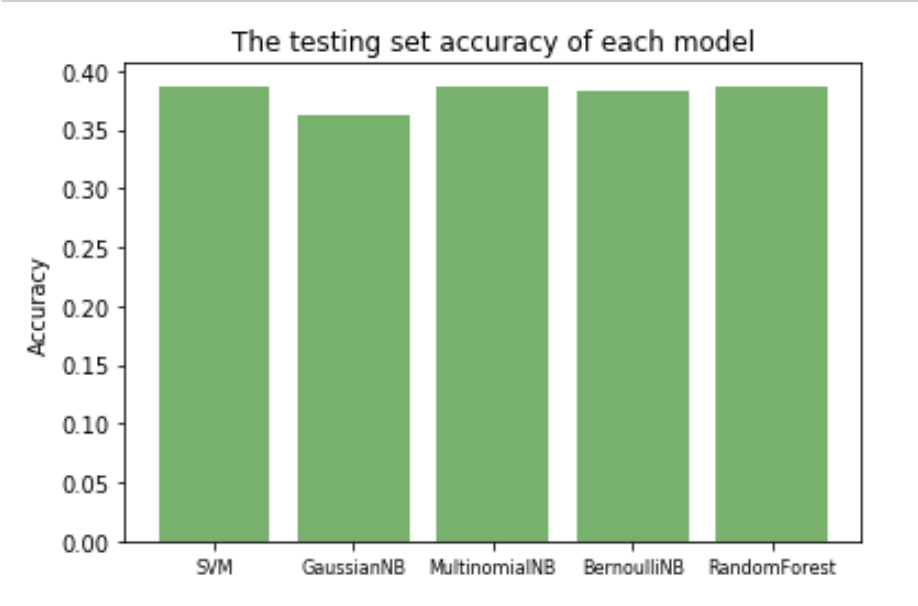
 

Figure 17. Comparison of models’ accuracy

3.4.1 Multi-class classification with 38 features

As mentioned before, we built 5 different models and applied the 5-fold stratified cross validation strategy on both models then compared the prediction accuracy. They are SVM, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, Multinomial Naïve Bayes and Random Forest classifiers.

According to figure 17, Multinomial Naïve Bayesian classifier achieves the best accuracy of 39.5% in crime prediction compared with other Naïve Bayesian classifiers. On the other hand, random forest tree classifier reports less prediction accuracy with 38.9%. In any case, the two classifiers have the same performance in terms of their running time. However, the accuracy of each model in both the training set and testing set is only around 40%. Based on the observed results, it is likely that we under fitted our dataset. Hence, we decided to put more complexity into our classification model in the next step.

3.4.2 Multi-class classification with 1212 features

To increase the complexity of our model, we tend to add more features and process them to make the model training more easily. As we mentioned before, the feature SHOOTING is the one with a large fraction of missing values. To make our model learn this specific trend, we create a new feature called IS\_SHOOTING\_NA. Also, we reprocessed the features with geographic information. Since we have a lot of categorical variables, we applied two strategies to handle different situations. For features with small number of categories, we create dummy variables. As For features with large number of categories, we only create dummy variables with frequent category. In other words, we label infrequent values as uncommon to avoid a vast number of dummy variables we might encountered. After the reprocessing of our features, we acquire a new dataset with 1212 features. Considering the computational cost and training efficiency, we only trained a multi-class classification Random Forest on this new dataset.

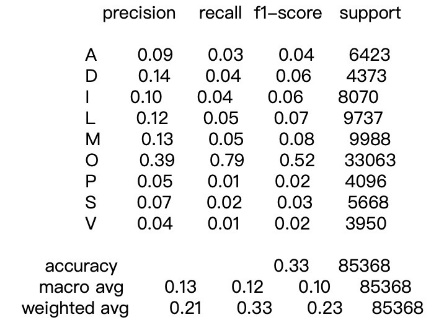
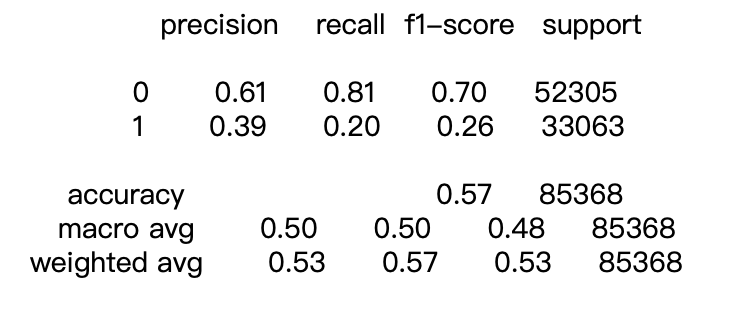
 

Figure 18. Result of multi-class classification (left) and O vs Others binary classification (right)

3.4.3 O vs Others binary classification

In this part, we reprocessed the target variable. Labeled the observation as ‘1’ when it is an O type crime and ‘0’ when it is other crime category. Then we trained our dataset with 1212 features on a Random Forest classifier. Disappointingly, based on the result of figure 18, we still did not get acceptable results. Usually, for a binary classification task, an accuracy higher than 80% is ideal. To optimize our model, we tended to Principal Component Analysis to see if we could gain an improvement by dimension reduction techniques.

3.4.4 Principal Component Analysis

In this section, we applied Principal Component Analysis to reduce the dimension of our new dataset and extract the top 3 principal components regarding the variance explained. Then we visualized our dataset in the 3D principal components dimension and analyze the pattern for each class. As we could see from figure 19, features in our dataset do not separate each class well and all the classes are just joined together in several specific space. In other words, it is hard for the models we build to find a perfect decision hyperplane to split each class accurately. The reason for this is that the features we have do not possess an ideal modelablity and enough information to accomplish such a challenging 9 class classification task. And that is also the main reason that we do not gain a desirable performance on all our models.

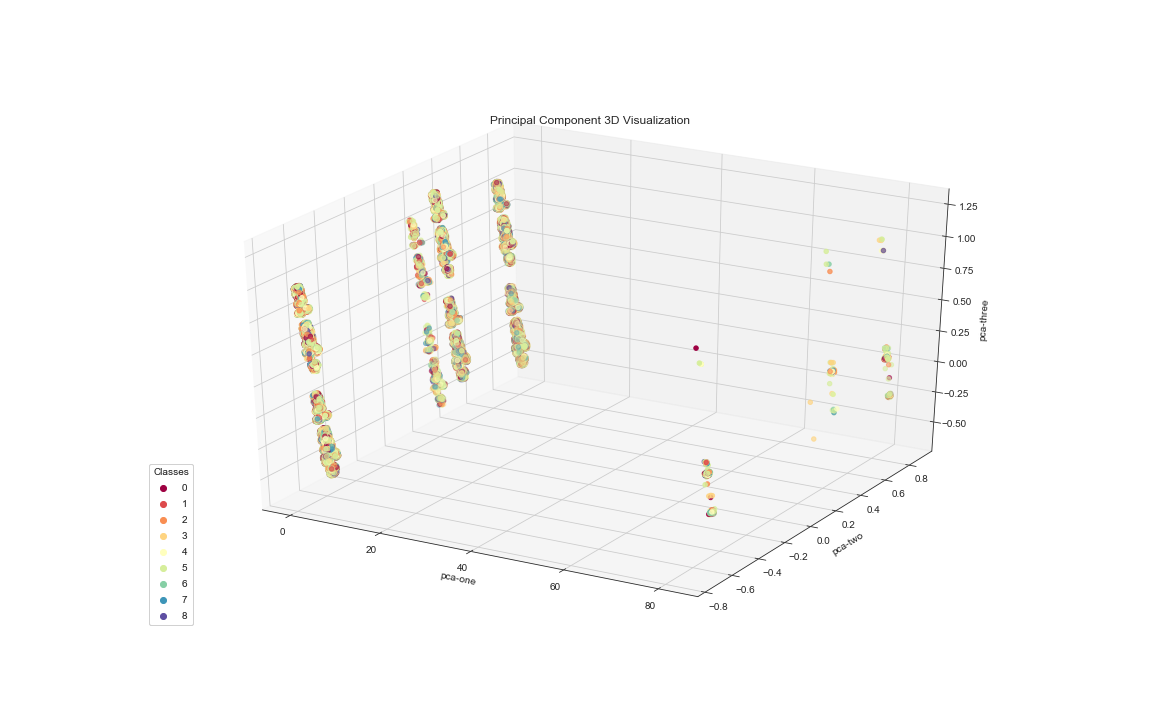


Figure 19. Principal component 3D visualization

**4 Discussion**

We generated numerous graphs and discovered interesting statistics that demonstrated the baseline to understand Boston crimes datasets. From that point onward, we applied Support Vector Machine, Naïve Bayesian and Random Forest. From that point onward, we applied Random Forest Tree and Naïve Bayesian classifiers to help to predict future crimes in a specific location within a particular time. We achieved 39% of prediction accuracy. We aimed to additionally comprehend our models’ discoveries and to capture the factors that might affect the safety of neighborhoods. As a future extension of our work, we intend to apply more classification models to increase crime prediction accuracy and to enhance the overall performance. It is also a helpful extension for our study to consider the demographic’s information and the income information for neighborhoods in order to see if there are relationships between neighborhoods population/income level and their crime rate. Besides, we want to study other crimes datasets from new cities. Finally, we hope this study of crimes can help law enforcements and keep our community more secure for everybody.

**5 Statement of contributions**

Data processing: Minjie Xu, Surui Yang

Data visualization: Minjie Xu

Models building and evaluation: Surui Yang, Jieyu Sheng

Slides and report summarization and improving: Huilian Jiang, Yuchen Tian

**6 References**

[1] “Boston Crime Rate Report (Massachusetts).” CityRating.com: Find the Best Places to Live in the USA,

<https://www.cityrating.com/crime-statistics/massachusetts/boston.html>

[2] Dataset resource: <https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-newsystem>

**7 Appendix**

7.1 R script

|  |
| --- |
| library(tidyverse)  crime<-read.csv("tmpwwunr6r7.csv",na="")  # names(crime)  # drop useless columns OFFENSE\_CODE\_GROUP, REPORTING\_AREA, OCCURRED\_ON\_DATE, UCR\_PART,Location  crime<-crime%>%  select(-OFFENSE\_CODE\_GROUP, -REPORTING\_AREA, -UCR\_PART, -Location)  # clean incorrect lat and long data  crime <- crime %>% mutate(Lat = replace(Lat, as.integer(Lat)==-1, NA),  Long=replace(Long, as.integer(Long)==-1, NA))  glimpse(crime)  library(naniar)  gg\_miss\_upset(crime)  # clear OFFENSE\_DESCRIPTION column  crime$OFFENSE\_DESCRIPTION<-iconv(crime$OFFENSE\_DESCRIPTION,from="UTF-8",to="ASCII//TRANSLIT")  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'M/V', 'MOTOR/VEHICLE')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'D/W', 'DANGEROUS WEAPON')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'VAL', 'VALIDATION')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'VIOL.-', 'VIOLATION-')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'VIOL. OF', 'VIOLATION OF')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'B&E', 'BREAKING AND ENTERING')  crime$OFFENSE\_DESCRIPTION<-str\_replace(crime$OFFENSE\_DESCRIPTION, 'A&B', 'ASSAULT AND BATTERY')  code\_name<-crime%>%  filter(!is.na(OFFENSE\_DESCRIPTION))%>%  group\_by(OFFENSE\_CODE)%>%  summarise(OFFENSE\_DESCRIPTION2=names(which.max(table(OFFENSE\_DESCRIPTION))))  crime<-crime%>%  left\_join(code\_name,by="OFFENSE\_CODE")%>%  select(-OFFENSE\_DESCRIPTION)  # Word cloud for offense description  library(tidytext)  library(wordcloud)  tidy\_desc<-crime%>%  unnest\_tokens(word, OFFENSE\_DESCRIPTION2)%>%  count(word, sort=TRUE)%>%  filter(!word %in% stop\_words$word)  wordcloud(words = tidy\_desc$word, freq = tidy\_desc$n, min.freq = 1,  max.words=100, random.order=FALSE, rot.per=0.15,  colors=brewer.pal(8, "Dark2"),scale=c(3,0.2))  # bigram analysis for offense description  library(igraph)  library(ggraph)  crime\_graph<-crime%>%  unnest\_tokens(bigram, OFFENSE\_DESCRIPTION2, token = "ngrams", n = 2)%>%  select(bigram)%>%  separate(bigram, c("word1", "word2"), sep = " ")%>%  filter(!word1 %in% stop\_words$word,!is.na(word1)) %>%  filter(!word2 %in% stop\_words$word,!is.na(word2))%>%  count(word1, word2, sort=TRUE) %>%  filter(n > 1500) %>%  graph\_from\_data\_frame()  ggraph(crime\_graph,layout="igraph",algorithm="kk") +  geom\_edge\_link() +  geom\_node\_point() +  geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)  ggsave("bigram.jpeg", width=10, height=7)  # Bin firstly OFFENSE\_DESCRIPTION2 based on names.  # Then we will only keep the most frequent categories and bin less frequent (<5%\*440606~20000) ones into "Other"  crime<-crime%>%  mutate(OFFENSE\_DESCRIPTION2=str\_replace(OFFENSE\_DESCRIPTION2,"\\ - ", "-"),  OFFENSE\_DESCRIPTION2=str\_replace(OFFENSE\_DESCRIPTION2,"\\-.\*", ""),  OFFENSE\_DESCRIPTION2=str\_replace(OFFENSE\_DESCRIPTION2,"\\(.\*", ""),  OFFENSE\_DESCRIPTION2=str\_trim(OFFENSE\_DESCRIPTION2),  sub\_name=str\_sub(OFFENSE\_DESCRIPTION2, start = 1L, end = 4L))  crime\_name\_abb<-crime%>%  group\_by(sub\_name)%>% summarise(OFFENSE\_DESCRIPTION3=names(which.max(table(OFFENSE\_DESCRIPTION2))),count=n())%>%  arrange(desc(count))%>%  mutate(OFFENSE\_bin=ifelse(count>20000,OFFENSE\_DESCRIPTION3,"OTHER"),  OFFENSE\_CATEGORY=str\_sub(OFFENSE\_bin, start = 1L, end = 1L))  crime\_name\_abb2<-crime\_name\_abb%>%  group\_by(OFFENSE\_bin,OFFENSE\_CATEGORY)%>%  summarise(n=sum(count))  ggplot(crime\_name\_abb2,aes(x=reorder(OFFENSE\_bin,n),y=n))+  geom\_col(fill="darkred")+  geom\_label(aes(label = n), color = "black", hjust = 1) +  coord\_flip()+  labs(x = "Offense Description", y='Number of crime incidents')  ggsave("crime\_type\_bar\_plot.jpeg", width=10, height=5)  # simplify OFFENSE\_DESCRIPTION for the original crime data frame  crime<-crime%>%  left\_join(crime\_name\_abb,by="sub\_name")%>%  select(-OFFENSE\_DESCRIPTION2,-OFFENSE\_DESCRIPTION3,-sub\_name,-count,-OFFENSE\_bin)  # visualization for DISTRICT  dist\_table<-tibble(DISTRICT=c("A1","A15","A7","B2","B3","C11","C6","D14","D4","E13","E18","E5","External"),  DIST\_NAME=c("Downtown","Charlestown","East Boston","Roxbury","Mattapan",  "Dorchester","South Boston","Brighton","South End","Jamaica Plain",  "Hyde Park","West Roxbury","External"))  crime%>%  filter(!is.na(DISTRICT))%>%  left\_join(dist\_table,by="DISTRICT")%>%  count(DIST\_NAME,sort=TRUE)%>%  ggplot(aes(x=reorder(DIST\_NAME,n),y=n))+  geom\_col(fill="darkred")+  geom\_label(aes(label = n), color = "black", hjust = 1) +  coord\_flip()+  labs(x = "District Name", y='Number of crime incidents')  ggsave("crime\_district\_bar\_plot.jpeg", width=10, height=5)  ## Should bin West Roxbury, East Boston, Charlestown, External into Other, do it later  # Word cloud for street  street<-as.data.frame(table(crime$STREET))  colnames(street) <- c("Street\_Name", "Count")  street<-street[street$Street\_Name!="",]  wordcloud(street$Street\_Name, street$Count, min.freq = 50,  max.words=150, random.order=FALSE, rot.per=0.15,  colors=brewer.pal(8, "Dark2"),scale=c(1.5,.3))  # Visualization based on Long and Lat data  library(leaflet) # interactive mapping  library(leaflet.extras) #extra mapping for leaflet  crime\_location<-crime %>%  filter(!is.na(Long),!is.na(Lat))%>%  select(Long,Lat)  crime\_location %>%  leaflet() %>%  setView(lng = -71.0705, lat = 42.33306, zoom = 11) %>%  addTiles( ) %>%  addHeatmap(lng =crime\_location$Long, lat =crime\_location$Lat, max = 2, radius = 12)  # hot spots for crime location top 50  crime\_location2<-crime\_location %>%  count(Long,Lat,sort=TRUE)%>%  top\_n(50)  crime\_location2 %>%  leaflet() %>%  setView(lng = -71.0745, lat = 42.3275, zoom = 12) %>%  addTiles( ) %>%  addHeatmap(lng =crime\_location2$Long, lat =crime\_location2$Lat, max = 2, radius = 12)  # Visualization Long and Lat data hot spots based on offense category  crime %>%  filter(!is.na(Long),!is.na(Lat),!is.na(OFFENSE\_CATEGORY))%>%  left\_join(crime\_name\_abb2,by="OFFENSE\_CATEGORY")%>%  count(OFFENSE\_bin,Long,Lat,sort=TRUE)%>%  top\_n(10000)%>%  ggplot()+  geom\_point(aes(x=Long,y=Lat),alpha=0.1,color="red")+  facet\_wrap(~ OFFENSE\_bin)+  labs(title="Crime hotspots of each crime category")  ggsave("Crime\_hotspots\_each\_crime\_category.jpeg", width=10, height=5)  # Visualization for date-time features  crime<-crime%>%  mutate(Date=str\_sub(OCCURRED\_ON\_DATE, start = 1L, end = 10L))  # Daily crime incidents line  crime%>%  count(Date,sort=TRUE)%>%  ggplot(aes(x = Date, y = n, group=1))+  geom\_line(color="#E7B800", size = 1)+  theme\_minimal()+  labs(x = "Date", y='Number of crime incidents in that day',title='Daily crime incidents line plot')  # clean shooting column  crime<-crime%>%mutate(SHOOTING =as.character(SHOOTING),  SHOOTING=str\_replace(SHOOTING, 'Y', '1'))  crime%>%  count(Date,SHOOTING)%>%  ggplot( )+  geom\_line(aes(x=Date,y=n,group=SHOOTING,color=SHOOTING),size=1)+  theme\_minimal()+  labs(x = "Date", y="Incidents count",title='Daily crime incidents line plot')  # clean shooting column  # fill na with 0  crime<-crime%>%mutate(SHOOTING =replace\_na(SHOOTING,"0"))  crime%>%  count(Date,SHOOTING)%>%  ggplot( )+  geom\_line(aes(x=Date,y=n,group=SHOOTING,color=SHOOTING),size=1)+  theme\_minimal()+  labs(x = "Date", y="Incidents count",title='Daily crime incidents line plot')  ggsave("Daily\_crime\_incidents\_line\_plot.jpeg", width=10, height=5)  # Visualize yealy shooting & crime pattern  crime\_year<-crime%>%  count(YEAR)  crime%>%  filter(SHOOTING=="1")%>%  count(YEAR)%>%  left\_join(crime\_year,by="YEAR")%>%  mutate(SHOOTING=n.x,CRIME=n.y)%>%  gather(key="INCIDENT",value="n",SHOOTING,CRIME)%>%  ggplot(aes(x=YEAR,y=n))+  geom\_line(aes(color = INCIDENT),size=1.5) +  geom\_text(aes(label = n),vjust=-0.3, color="black", size=3.5)+  labs(title = "Yearly pattern of crime and shooting", x = "Year", y = "Incidents count")+  theme\_bw()  ggsave("Yearly pattern of crime and shooting.jpeg", width=10, height=5)  crime%>%  count(YEAR,SHOOTING)%>%  left\_join(crime\_year,by="YEAR")%>%  mutate(proportion=n.x/n.y)%>%  ggplot(aes(x=YEAR,y=proportion,fill=SHOOTING,label = paste0(round(proportion\*100,1),"%")))+  geom\_col( )+  geom\_text(position = position\_stack(vjust = 0.5),color="black", size=3.5)+  coord\_flip()+  labs(x = "Year", y='SHOOTING Proportion')  ggsave("Yearly pattern of crime and shooting proportion.jpeg", width=10, height=5)  # yearly stacked barchart  crime%>%  filter(!is.na(YEAR),!is.na(DISTRICT),!is.na(OFFENSE\_CATEGORY))%>%  count(YEAR,DISTRICT,OFFENSE\_CATEGORY)%>%  left\_join(dist\_table,by="DISTRICT")%>%  left\_join(crime\_name\_abb2,by="OFFENSE\_CATEGORY")%>%  ggplot( ) +  geom\_col(aes(x=YEAR,y=n.x,fill=OFFENSE\_bin),position="stack")+  facet\_wrap(~ DIST\_NAME)+  theme(legend.title = element\_blank(),axis.text.x = element\_text(angle = 45, hjust = 1))+  labs(x = "Year", y="Incidents count",title="Yearly crime incidents bar chart in each district")  ggsave("Yearly crime incidents bar chart in each district.jpeg", width=10, height=5)  # Visualize monthly pattern for crime  crime%>%  count(MONTH,DAY\_OF\_WEEK)%>%  ggplot( )+  geom\_line(aes(x=as.factor(MONTH),y=n,  group=DAY\_OF\_WEEK,color=DAY\_OF\_WEEK),size=1.5)+  theme\_minimal()+  labs(x = "Month", y="Incidents count",title='Weekday crime incidents versus month line plot')  ggsave("Weekday crime incidents versus month line plot.jpeg", width=10, height=5)  # Visualize monthly pattern for SHOOTING  crime%>%  filter(SHOOTING=="1")%>%  count(MONTH,DAY\_OF\_WEEK)%>%  ggplot( )+  geom\_line(aes(x=factor(DAY\_OF\_WEEK,weekdays(min(as.Date(crime$Date))+0:6)),y=n,  group=as.factor(MONTH),color=as.factor(MONTH)),size=1.5)+  guides(color=guide\_legend(title="Month"))+  theme\_minimal()+  labs(x = "Day of week", y="Incidents count",title='Monthly shooting incidents versus weekday line plot')  ggsave("Monthly shooting incidents versus weekday line plot.jpeg", width=10, height=5)  # heatmap to show month and hour crime relationship.  crime %>%  count(MONTH,HOUR)%>%  ggplot()+  geom\_tile(aes(x = as.factor(MONTH),y=as.factor(HOUR), fill=n), na.rm=TRUE)+  labs(x = "Month", y='Hour',title='Crime heatmap of Hour versus Month')  ggsave("Crime heatmap of hour versus Month.jpeg", width=10, height=5)  # Visualization of Shooting as a function of month  crime%>%  count(MONTH,SHOOTING)%>%  ggplot()+  geom\_line(aes(x=as.factor(MONTH),y=n,group=SHOOTING,color=SHOOTING),size=1.5)+  geom\_text(aes(x=as.factor(MONTH),y=n,label = n),vjust=-0.3, color="black", size=3.5)+  labs(x = "Month", y="Number of crime incidents")+  theme\_bw()  ggsave("Monthly pattern of if shooting crime.jpeg", width=10, height=5)  # Visualize weekly pattern for crime  crime %>%  count(DAY\_OF\_WEEK,sort=TRUE)%>%  ggplot(aes(x = reorder(DAY\_OF\_WEEK,n), y = n)) +  geom\_col(fill = "darkred") +  geom\_label(aes(label = n), color = "black", hjust = 1) +  coord\_flip() +  labs(title = "Weekday pattern of crime", x = "Day of week", y = "Incidents count")+  theme\_bw()  ggsave("Weekday pattern of crime.jpeg", width=10, height=5)  # Visualization of Shooting as a function of weekday  crime%>%  count(DAY\_OF\_WEEK,SHOOTING)%>%  ggplot(aes(x=factor(DAY\_OF\_WEEK,weekdays(min(as.Date(crime$Date))+0:6)),y=n))+  geom\_line(aes(color=SHOOTING,group=SHOOTING),size=1.5)+  geom\_text(aes(label = n),vjust=-0.3, color="black", size=3.5)+  labs(x = "Day of week", y="Number of crime incidents")+  theme\_bw()  ggsave("Weekday pattern of if shooting crime.jpeg", width=10, height=5)  # crime incidents as a function of Hours in a week  library(grid)  library(gridExtra)  Sun <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Sunday")  Mon <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Monday")  Tue <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Tuesday")  Wed <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Wednesday")  Thu <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Thursday")  Fri <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Friday")  Sat <- crime %>%  count(HOUR, DAY\_OF\_WEEK) %>%  filter(DAY\_OF\_WEEK=="Saturday")  m1 <- Sun %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="red") +  geom\_point(color="red") +  labs(title = "Sunday", x = "Hour", y = "Incidents count") +  theme\_bw()  m2 <- Mon %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="blue") +  geom\_point(color="blue") +  labs(title = "Monday", x = "Hour", y = "Incidents count") +  theme\_bw()  m3 <- Tue %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="blue") +  geom\_point(color="blue") +  labs(title = "Tuesday", x = "Hour", y = "Incidents count") +  theme\_bw()  m4 <- Wed %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="blue") +  geom\_point(color="blue") +  labs(title = "Wednesday", x = "Hour", y = "Incidents count") +  theme\_bw()  m5 <- Thu %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="blue") +  geom\_point(color="blue") +  labs(title = "Thursday", x = "Hour", y = "Incidents count") +  theme\_bw()  m6 <- Fri %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="blue") +  geom\_point(color="blue") +  labs(title = "Friday", x = "Hour", y = "Incidents count") +  theme\_bw()  m7 <- Sat %>% ggplot(aes(x = HOUR, y = n)) +  geom\_line(lty = 2,color="red") +  geom\_point(color="red") +  labs(title = "Saturday", x = "Hour", y = "Incidents count") +  theme\_bw()  grid.arrange(m7,m1,m2,m3,m4,m5,m6, top = textGrob("Total number of crime incidents in Hour for One Week", gp = gpar(fontsize = 15,font = 3)))  # Visualization of Shooting as a function of hour  crime%>%  count(HOUR,SHOOTING)%>%  ggplot()+  geom\_line(aes(x=as.factor(HOUR),y=n,color=SHOOTING,group=SHOOTING),size=1.5)+  geom\_text(aes(x=as.factor(HOUR),y=n,label = n),vjust=-0.3, color="black", size=3.5)+  labs(x = "Hour of a day", y="Number of crime incidents")+  theme\_bw()  ggsave("Hourly pattern of if shooting crime.jpeg", width=10, height=5)  # mosaic plot to show offense type and shooting relationship.  library(ggmosaic)  crime %>%  left\_join(crime\_name\_abb2,by="OFFENSE\_CATEGORY")%>%  ggplot()+  geom\_mosaic(aes(x = product(SHOOTING,OFFENSE\_bin), fill=SHOOTING), na.rm=TRUE) +  coord\_flip() +  labs(x ="OFFENSE DESCRIPTION" , y="IF SHOOTING",title='Mosaic plot for crime type vs shooting relationship')  ggsave("Mosaic plot for crime type vs shooting relationship.jpeg", width=10, height=5)  # Visualization of Shooting as a function of district  crime%>%  filter(!is.na(DISTRICT))%>%  count(DISTRICT,SHOOTING)%>%  left\_join(dist\_table,by="DISTRICT")%>%  ggplot()+  geom\_line(aes(x=DIST\_NAME,y=n,color=SHOOTING,group=SHOOTING),size=1.5)+  geom\_text(aes(x=DIST\_NAME,y=n,label = n),vjust=-0.3, color="black", size=3.5)+  labs(x = "District", y="Number of crime incidents")+  theme(axis.text.x = element\_text(angle = 90, hjust = 1))  ggsave("District distribution of if shooting crime.jpeg", width=10, height=5)  ## bin West Roxbury, East Boston, Charlestown, External into Other  crime\_district<-crime%>%  filter(!is.na(DISTRICT))%>%  left\_join(dist\_table,by="DISTRICT")%>%  count(DISTRICT,DIST\_NAME,sort=TRUE)%>%  mutate(DISTRICT2=ifelse(n>20000,DIST\_NAME,"Other"))  ## bin 21,22,23,1,2,3,4,5,6,7,8 into 0 for the column HOUR  crime<-crime%>%  mutate(HOUR=ifelse(HOUR %in% c(21,22,23,1,2,3,4,5,6,7,8),1,HOUR))  # Preparation for models  crime<-crime%>%  left\_join(crime\_district,by="DISTRICT")%>%  select(DISTRICT2,OFFENSE\_CATEGORY,MONTH,DAY\_OF\_WEEK,HOUR)%>%  filter(!is.na(DISTRICT2),!is.na(OFFENSE\_CATEGORY),!is.na(MONTH),!is.na(DAY\_OF\_WEEK),  !is.na(HOUR))  # creating dummy variables  crime\_y<-crime[,"OFFENSE\_CATEGORY"]  crime\_x<-crime[,-2]  crime\_x<-mutate\_all(crime\_x,as.character)  library(caret)  dummies\_model <- dummyVars(" ~ .", data=crime\_x,fullRank=TRUE)  crime\_x <- data.frame(predict(dummies\_model, newdata = crime\_x))  crime<-cbind(crime\_x,as.character(crime\_y))  names(crime)[length(crime)]<-"OFFENSE\_CATEGORY"  write.csv(crime,"clean\_crime.csv", row.names = FALSE) |

7.2 Python script

|  |
| --- |
| # Multi-class classification with 38 features  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.metrics import accuracy\_score  from skmultilearn.problem\_transform import BinaryRelevance  from skmultilearn.problem\_transform import ClassifierChain  from sklearn.naive\_bayes import GaussianNB  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt; plt.rcdefaults()  from sklearn.metrics import precision\_recall\_curve  from sklearn.metrics import average\_precision\_score  from sklearn.preprocessing import label\_binarize  from itertools import cycle  from sklearn.svm import LinearSVC  from sklearn.multiclass import OneVsRestClassifier  def preprecess(root):  data = pd.read\_csv(root)  df = pd.DataFrame(data = data)  dictionary = {"A":0,"D":1,"I":2,"L":3,"M":4,"O":5,"P":6,"S":7,"V":8}  X\_df = df[df.columns[0:38]]  Y\_df = df[["OFFENSE\_CATEGORY"]]  y = list()  x = []  #prepare for y  for index, rows in Y\_df.iterrows():  off = rows["OFFENSE\_CATEGORY"]  label = dictionary[off]  y.append(label)      #prepare for x  name = list(X\_df.columns)  for col in name:  a = X\_df[col].tolist()  x.append(a)  X = np.array(x)  XX = np.reshape(X, (len(X[0]), len(X)))  return XX, np.asarray(y)  root = "/Users/jieyusheng/Rstudio/project/data1.csv"  x, y = preprecess(root)  print(len(x))  X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2)  # explore the classes to check whether imbalanced class or not  dictionary = {"A":0,"D":1,"I":2,"L":3,"M":4,"O":5,"P":6,"S":7,"V":8}  a = set(y)  b = {}  for i in a:  b.update({i:0})    for i in range(len(y)):  if y[i] in b.keys():  b[y[i]] += 1  objects = list(dictionary.keys())  y\_pos = np.arange(len(objects))  performance = list(b.values())  plt.bar(y\_pos, performance, align='center', color="#ffce00", alpha=0.9)  plt.xticks(y\_pos, objects)  plt.ylabel('the number of incident')  plt.title('the number of each crime category')  plt.show()  plt.savefig('cal1.jpg')  def acc(obs, predict):  y\_pred = list(predict)  y\_obs = obs.tolist()  count = 0  for i in range(len(obs)):  if (y\_obs[i] == y\_pred[i]):  count += 1  return count/len(obs)  #model part: NB, SVM and Randomforest  from sklearn.naive\_bayes import MultinomialNB  from sklearn.naive\_bayes import BernoulliNB  from sklearn.ensemble import RandomForestClassifier  from sklearn.datasets import make\_classification  #svm  svm\_clf = OneVsRestClassifier(LinearSVC(random\_state=0))  svm\_test\_score = svm\_clf.fit(X\_train, y\_train).predict(X\_test)  svm\_acc = acc(y\_test, svm\_test\_score)  # Gussian Naive Beyes  gaussianNB\_clf = ClassifierChain(GaussianNB())  gaussianNB\_clf.fit(X\_train, y\_train)  gaussian\_test\_score = gaussianNB\_clf.predict(X\_test)  gaussian\_acc = acc(y\_test, gaussian\_test\_score)  #Multinomial Naive Beyes  multinomialNB\_clf = MultinomialNB()  multinomialNB\_clf.fit(X\_train, y\_train)  multinomialNB\_test\_score = multinomialNB\_clf.predict(X\_test)  multinomialNB\_acc = acc(y\_test, multinomialNB\_test\_score)  #Bernoulli Naive Beyes  bernoulliNB\_clf= BernoulliNB()  bernoulliNB\_clf.fit(X\_train, y\_train)  bernoulliNB\_test\_score = bernoulliNB\_clf.predict(X\_test)  bernoulliNB\_acc = acc(y\_test, bernoulliNB\_test\_score)  #random forest  rf\_clf = RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',  max\_depth=2, max\_features='auto', max\_leaf\_nodes=None,  min\_impurity\_decrease=0.0, min\_impurity\_split=None,  min\_samples\_leaf=1, min\_samples\_split=2,  min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None,  oob\_score=False, random\_state=0, verbose=0, warm\_start=False)  rf\_clf.fit(X\_train, y\_train)  rf\_test\_score = rf\_clf.predict(X\_test)  rf\_acc = acc(y\_test, rf\_test\_score)  #svm\_acc2 = acc(y\_test, svm\_train\_score)  # Gussian Naive Beyes  gaussian\_train\_score = gaussianNB\_clf.predict(X\_train)  gaussian\_acc2 = acc(y\_test, gaussian\_train\_score)  #Multinomial Naive Beyes  multinomialNB\_train\_score = multinomialNB\_clf.predict(X\_train)  multinomialNB\_acc2 = acc(y\_train, multinomialNB\_train\_score)  #Bernoulli Naive Beyes  bernoulliNB\_train\_score = bernoulliNB\_clf.predict(X\_train)  bernoulliNB\_acc2 = acc(y\_train, bernoulliNB\_train\_score)  #random forest  rf\_clf.fit(X\_train, y\_train)  rf\_train\_score = rf\_clf.predict(X\_train)  rf\_acc2 = acc(y\_train, rf\_train\_score)  #training accuracy  objects = ("SVM", "GaussianNB", "MultinomialNB", "BernoulliNB", "RandomForest")  y\_pos = np.arange(len(objects))  performance = (svm\_acc2, gaussian\_acc2, multinomialNB\_acc2, bernoulliNB\_acc2,rf\_acc2)  plt.bar(y\_pos, performance, align='center', color="b", alpha=0.6)  plt.xticks(y\_pos, objects, size="8")  plt.ylabel('Accuracy')  plt.title('The training accuracy of each model')  plt.show()  svm\_clf = OneVsRestClassifier(LinearSVC(random\_state=0))  svm\_train\_score = svm\_clf.fit(X\_train, y\_train).predict(X\_train)  svm\_acc2 = acc(y\_train, svm\_train\_score)  #test accuracy  objects = ("SVM", "GaussianNB", "MultinomialNB", "BernoulliNB", "RandomForest")  y\_pos = np.arange(len(objects))  performance = (svm\_acc, gaussian\_acc, multinomialNB\_acc, bernoulliNB\_acc,rf\_acc)  plt.bar(y\_pos, performance, align='center', color="g", alpha=0.6)  plt.xticks(y\_pos, objects, size="8")  plt.ylabel('Accuracy')  plt.title('The testing set accuracy of each model')  plt.show()  ## Multi-class classification with 1212 features  import pandas as pd  import numpy as np  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score  from sklearn.model\_selection import StratifiedKFold  from sklearn.model\_selection import cross\_val\_score  from sklearn.multiclass import OneVsRestClassifier  from sklearn.metrics import recall\_score  from sklearn.metrics import classification\_report  from sklearn.multiclass import OneVsRestClassifier  import matplotlib.pyplot as plt  from mpl\_toolkits.mplot3d import Axes3D  df = pd.read\_csv('tmpwwunr6r7.csv')  df.head()  df = df.drop(['INCIDENT\_NUMBER'], axis=1)  df = df.drop(['YEAR'], axis=1)  df = df.drop(['Location'], axis=1)  df = df.drop(['OFFENSE\_DESCRIPTION'], axis=1)  df = df.drop(['OFFENSE\_CODE'], axis=1)  df = df.drop(['OCCURRED\_ON\_DATE'], axis=1)  df['ST\_IS\_NA'] = ['1' if df['SHOOTING'][i] != df['SHOOTING'][i] else '0' for i in range(len(df)) ]  df['MONTH'] = df['MONTH'].apply(str)  df['HOUR'] = df['HOUR'].apply(str)  df['SHOOTING'] = [1 if df['SHOOTING'][i] =='Y' else 0 for i in range(len(df)) ]  df['Lat'].fillna(df['Lat'].mean(), inplace=True)  df['Long'].fillna(df['Long'].mean(), inplace=True)  df = df[df['OFFENSE\_CODE\_GROUP']== df['OFFENSE\_CODE\_GROUP'] ]  # categorial variable process  #DISTRICT  DISTRICT = pd.get\_dummies(df.DISTRICT,drop\_first=True)  #REPORTING\_AREA  counts\_REPORTING\_AREA = df.REPORTING\_AREA.value\_counts()  pro\_REPORTING\_AREA = counts\_REPORTING\_AREA/counts\_REPORTING\_AREA.sum()  threshold\_REPORTING\_AREA = 1.0/counts\_REPORTING\_AREA.count()  repl\_REPORTING\_AREA = pro\_REPORTING\_AREA[pro\_REPORTING\_AREA <= threshold\_REPORTING\_AREA].index  REPORTING\_AREA = pd.get\_dummies(df.REPORTING\_AREA.replace(repl\_REPORTING\_AREA, 'Uncommon\_REPORTING\_AREA'),drop\_first=True)  #MONTH  MONTH = pd.get\_dummies(df.MONTH,drop\_first=True)  #DAY\_OF\_WEEK  DAY\_OF\_WEEK = pd.get\_dummies(df.DAY\_OF\_WEEK,drop\_first=True)  #HOUR  HOUR = pd.get\_dummies(df.HOUR ,drop\_first=True)  #UCR\_PART  #UCR\_PART = pd.get\_dummies(df.UCR\_PART,drop\_first=True)  #STREET  counts\_STREET = df.STREET.value\_counts()  pro\_STREET = counts\_STREET/counts\_STREET.sum()  threshold\_STREET = 1.0/counts\_STREET.count()  repl\_STREET = pro\_STREET[pro\_STREET <= threshold\_STREET].index  STREET = pd.get\_dummies(df.STREET.replace(repl\_STREET, 'Uncommon\_STREET'),drop\_first=True)  df\_processed =pd.concat([DISTRICT, REPORTING\_AREA, MONTH,DAY\_OF\_WEEK, HOUR, UCR\_PART, STREET], axis=1, join='inner')  df\_processed.head()  df\_processed = pd.concat([df[['SHOOTING','Lat', 'Long', 'ST\_IS\_NA']],df\_processed],axis=1, join='inner')  df\_processed.head()  data = pd.read\_csv('clean\_crime\_2.csv')  y = data['OFFENSE\_CATEGORY'][df.index]  from sklearn.metrics import classification\_report  from sklearn import preprocessing  le = preprocessing.LabelEncoder()  le.fit(y)  y = le.transform(y)  X = df\_processed  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y,test\_size=0.2, random\_state=28)  #Random Forest  clf2 = RandomForestClassifier(n\_estimators=100,random\_state=28,verbose=2)  clf2.fit(X\_train,y\_train)  predicted = clf2.predict(X\_test)  target\_names = list(le.classes\_)  print(classification\_report(y\_test, predicted, target\_names=target\_names))  # O vs Others binary classification  data['OFFENSE\_CATEGORY'] = [1 if data['OFFENSE\_CATEGORY'] [i] == 'O' else 0 for i in range(len(data)) ]  y\_= data['OFFENSE\_CATEGORY'][df.index]  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_, stratify=y,test\_size=0.2, random\_state=28)  clf3 = RandomForestClassifier(n\_estimators=100,random\_state=28,verbose=2).fit(X\_train, y\_train)  predicted = clf3.predict(X\_test)  print(classification\_report(y\_test, predicted))  # PCA  from sklearn.decomposition import PCA  pca = PCA(n\_components=3)  pca.fit(X)  print(pca.explained\_variance\_ratio\_)  result=pd.DataFrame(pca.transform(X), columns=['PCA%i' % i for i in range(3)], index=X.index)  data = pd.read\_csv('clean\_crime\_2.csv')  y\_pca = data['OFFENSE\_CATEGORY'][df.index]  le = preprocessing.LabelEncoder()  le.fit(y\_pca)  y\_pca = pd.DataFrame(le.transform(y\_pca))  y\_pca.index = df.index  result.head()  result\_pca= pd.concat([result,y\_pca], axis=1, join='inner')  result\_pca.columns =["PCA1","PCA2","PCA3","OFFENSE\_CATEGORY"]  result\_pca  ax = plt.figure(figsize=(16,10)).gca(projection='3d')  scatter = ax.scatter(  xs=result\_pca["PCA1"],  ys=result\_pca["PCA2"],  zs=result\_pca["PCA3"],  c=result\_pca["OFFENSE\_CATEGORY"],  cmap=plt.cm.Spectral  )  ax.set\_xlabel('pca-one')  ax.set\_ylabel('pca-two')  ax.set\_zlabel('pca-three')  legend1 = ax.legend(\*scatter.legend\_elements(),  loc="lower left", title="Classes")  ax.add\_artist(legend1)  plt.title('Principal Component 3D Visualization')  plt.savefig('PCA\_3D.png') |