Crime Prediction

Master's in information technology

Capstone Project

Student's Name: Venkata Avinash Paturu

Faculty Sponsor: Dr. Soon Ae Chun

December 2018

Table of Contents

ABSTRACT

This project investigates the feasibility of using machine learning techniques, specifically neural networks, to make prediction on criminal behavior based on the history of the arrest bookings. The experiment will have to handle imbalanced data frequencies. To combat the challenge, data augmentation and weighted loss function is being developed to extract information from the minority classes. For this project, I have focused on how neural networks can be advantageous in classification of crime prediction.

The specific kind of neural network that has been used in the project is a deep fully connected neural network. Fully connected neural networks are suitable for problems where domain knowledge is limited and many to many relations between features are important. As this report shows, machine learning techniques could definitely be of use for classification of criminal behavior, and we recommend exploring the discussed data augmentation and modeling methods more thoroughly to improve on the results and find new patterns.

1. INTRODUCTION

Future crime prediction forecasts crimes with two important goals. First, to take preventive measures and intelligently allocate law enforcement resources. Second, to assist the criminal justice system to make decisions about individuals. Which will be used for intervention, reform and rehabilitation of criminals while preventing crime from happening the problem we deal with in this work is of great importance for law enforcement agencies. Currently, law enforcement agencies statistically model spatio-temporal patterns of criminal incidents. These statistical models are used to study causality of crimes and predict future criminal incidents.

There have been a number of previous works addressing crime prediction. In Iqbal et al. [2] the authors used Naive Bayes and Decision Trees classifiers to predict the crime category for different states in the USA with 83.95% accuracy. In [4] the authors apply Twitter-specific linguistic analysis and statistical topic modeling to automatically identify discussion topics across a major city and improves the crime prediction performance over a kernel density estimation. [3] introduced a spatio-temporal model, to discover underlying factors related to crimes and predict future incidents. The model can fully utilize many different types of data, such as spatial, temporal, geographic, and demographic data, to make predictions. The notable difference between our work and above previous works is that we predict the crime and its type at the level of the individual, while above works are made at the level of a population. For instance, predicting how likely a crime will occur in a certain area. Moreover, we can predict the development of the seriousness of future crimes for individuals, which was not considered in previous work.

We apply the machine learning method DNN on the personal criminal charge history to accurately predict the crime reoccurrences. Moreover, we study the trend of the criminal charge level developed over the time of every single person, and in a varying number of years. More precisely our goal is to predict (a) whether a person will commit a crime in future. (b) The trend of one person's crime level. We try to reach these two goals by answering the following two questions, given one's criminal charge history and personal info (including gender, age, the level before year N):

1. Will this person obtain a new charge in the near future years?

2. Which level L (L ∈ 1,2,3, seriousness indicator of a crime) of the charge will this person obtain if she/he will obtain a charge?

This project is structured as follows, Part 2 describes relevant previous work in crime prediction. Part 3 focuses on definition of the research problem, the data at our disposal, engineered features, potential algorithms that can be used and our approaches. Part 4 and Part 5 discusses the experimental steps and our findings. Part 6 presents conclusions and scope of future work.

2. RELATED WORK

Researchers has devoted attention to studying crime hotspots based on both people and place centric perspective. Majority of the works are concentrated on exploring algorithms such as Decision trees, Random forests, SVM and Bayesian classifiers.

Bogomolov et al. [3] proposed an approach where they combined crime event data with the London borough profile and smart steps data, Smart steps data have a variety of demographic information such as average footfall and estimated gender, age, home/work/visitor group splits computed every hour for a 3-week period based on mobile phone usage in the area. Their proposed approach lies in using aggregated and anonymized human behavioral data derived from mobile network activity to tackle the crime prediction problem where they predict weather an area in the city will be a crime hotspot or not. They have experimented with Breiman's Random forests classifier yielding 70% accuracy and their findings support the hypothesis that aggregated human behavioral data captured from the mobile network infrastructure, in combination with basic demographic information, can be used to predict crime.

Iqbal et al. [2] used dataset prepared using real data from socio-economic data from 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR to predict the crime category for different states of USA. For the experiments they have used a manual feature selection to select 12 features from large selection oof 128 features and compared naïve bayes and decision trees for prediction. They have used WEKA software for this research. This is a multiclass classification with 3 labels Low, Medium and High. Both the classifiers used for this experiment have some pros and cons. Naïve Bayesian requires short training time, fast evaluation and is more suitable for real world problems. If talking about solving complex classification problems, then Naïve Bayesian is not a recommended choice. In order to handle complex classification prob- lems, Decision Tree is a better choice. It can produce reasonable and interpretable classification trees, which can be used for making critical decisions. However, it does not work well on all datasets. With an accuracy of 83% compared to 70% the results manifest that decision tree performed better than naïve bayes.

These research works make use of prior crime occurrences to classify hotspots, but they have a limited real time use on time sensitive crime prediction. After the proliferation of social media, it makes much more sense to use such data to track and predict crime hot spots which are time sensitive. Wang et al. [4] worked on twitter data and Kernel Density Estimation for crime prediction, for example, predicted crimes using GPS-tagged tweets. He hypothesized a pattern that determines that crime-related tweets dramatically increase in areas surrounding the locations and times of crime incidents. He analyzed the topics of tweets by using a latent Dirichlet allocation topic model. He then trained the prediction model and obtained successful results. Although a collection of tweets is free, obtaining a collection of historical tweets is either impossible or would require financial expenditures.

Broken windows theory states that visible signs of crime, anti-social behavior, and civil disorder create an urban environment that encourages further crime and disorder, including serious crimes. In order to take advantage of environmental context information, it needs to be fused with other spatial and temporal information to generate effective features. Kang et al. [1] focused on developing a deep learning model to learn from such extensive multi model data. Where they employed Alex net to learn from crime location images collected from google street view and fused them with demographics, weather and crime incidence information using deep neural networks. Their findings support their hypothesis that using environmental context information improves crime hotspots prediction.

Table 1 methods and results from previous research

	Data (# of records)	Decision Tree	Random Forests	SVM	KDE	Naive Bayes	ANN
Kang et al. [1]	Location images+ Demographics+ Crime incident data			67.1	66.33		84.25
Bogomolov et al. [3]	Crime incident data+ Location based tweets		69.54				
Iqbal et al. [2]	US state crime incident data+ Census data	83.9				70.8	

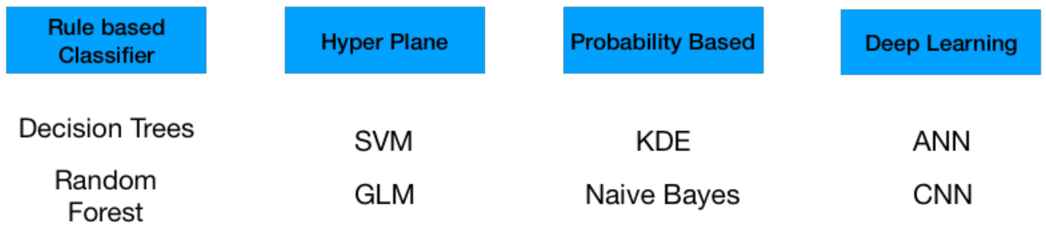


Figure 1Technologies used in the research papers

Despite the above contributions, most existing methods focus on hotspot classification to make effective use of law enforcement patrols. These approaches do not consider the rehabilitation and intervention in troubled cities to support recurring offenders in leading a path away from crime, which can be problematic. Thus, we use an alternative method to provide a solution to State Department of Police wherein they can examine a criminal history and focus on criminals who may be a threat to society, which will be used for intervention, reform and rehabilitation of criminals while preventing crime from happening, i.e. encountering the criminals at an early stage to avoid them to commit more severe crimes.

3. DATASET

3.1 Overview

The problem we are trying to solve is to accurately predict the crime reoccurrences based on the personal criminal charge history. Data collection is critical for the accurate prediction of crime reoccurrences. The dataset at our disposal is the booking history of criminals from 1997 to 2017. The raw data has 5 features per record with a unique PersonID to link multiple records together Table2. There are 16,841 unique people with 63,133 records of arrest history. There are 42 unique crime types with 3 levels of seriousness. We plan to create a model which predicts the possible crime level in the next 5 years. In order to create the labels for supervised learning, we used 5-year windows to look ahead for each crime record and pick the most serious level (e.g. level 1, 2, 3 or 0 for NoCrime) as its label for the predicted crime. The problem with this dataset is that each unique individual has his own timeline of sequence of crimes. Some people have days between 2 successive arrests and some have years. We need to find as effective method to create feature vectors for analysis while keeping the information intact.

In this section, we explore the dataset to provide the statistics of important features, and then preprocess the data to maximize the information utilization.

Table 2 Features in the Raw data Records (1997-2017)

Feature	Description
PersonID	A unique alphanumeric identification sequence
NCIC Crime Code (See table 4 and 5)	The National Crime Information Center (NCIC) is the Federal Government's central database for tracking crime-related information. These codes help determine the nature of offense (type and seriousness).
Gender	Gender of the person
Age	Age at the time of booking
Race	Race of the person. Has 5 levels
Booking Date	Date of the Booking

3.2 Exploratory Data Analytics

The collected data may contain information which does not need for crime occurrence. To solve this, it is necessary to explore data to select meaningful data with statistical significance related to crime occurrences. The basic statistics on the numerical attributes reveals that the worst criminal has a history of 56 bookings in his timeline. The oldest criminal in the data is 86 and the youngest is just 17 years of age. While NCIC_Level mean of 2.45 indicates most crimes are on level3. The mean year value 2008.2 indicates that there are more crimes committed in the window 2007-17 than in 1997-07.

Table 3 Basic Statistics on numerical attributes of the raw data

	Number of Booking	NCIC Level	Age	Year of booking
Counts (# of records)	63133	[1-3]	[17-86]	[1997-2017]
Mean (per person)	3.0955	2.451	32.2	2008
STD	4.05	0.817	11.1	3.863
Min	1	1	17.0	1997
25%	1	2	23.0	2005
50%	2	3	30.0	2008
75%	3	3	40.0	2011
Max	56	3	86.0	2017

Gender distribution shows that there are 89.2% records are from male offenders. About 66% of crimes are level 3, and level2 is the least of the three with 12%.

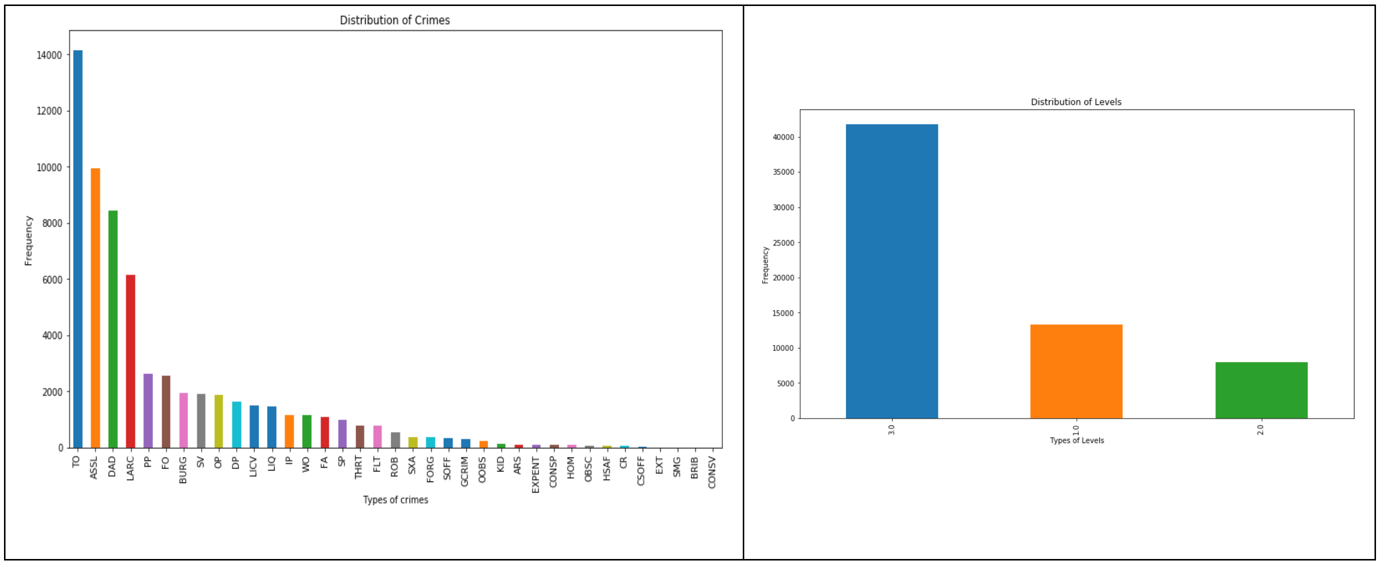


Figure 2 Crime Distribution

Distribution of crime and level show's that Traffic offence (TO), Assault (ASSL), Dangerous Drugs (DAD), Larceny (LARC) are the most frequent crimes compared to others.

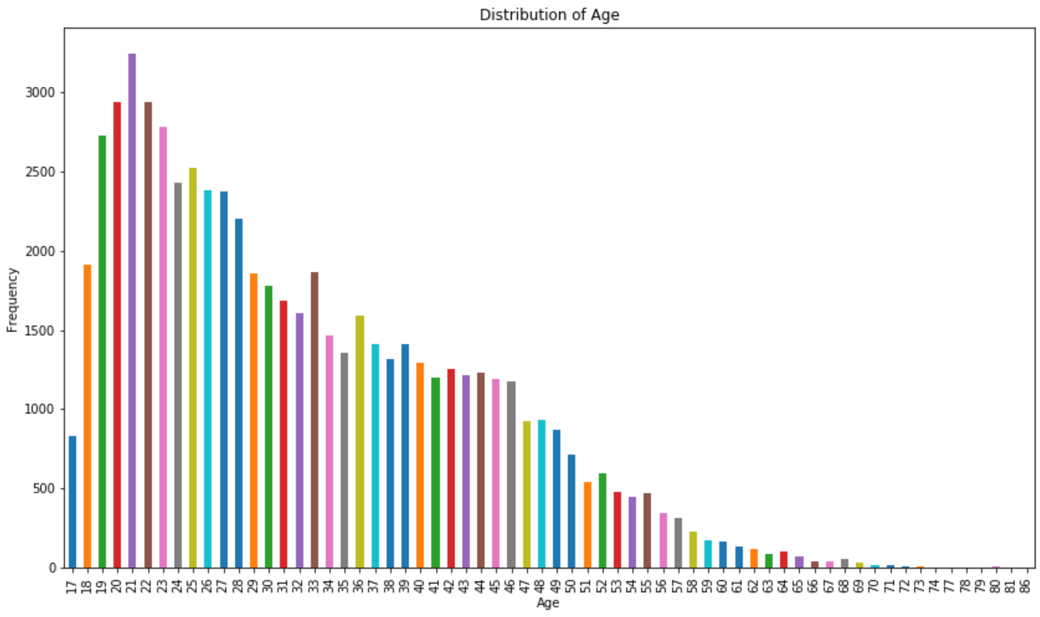


Figure 3 Age Distribution

The observed distribution of ages follows the expected assumption that young people are more frequent offenders.

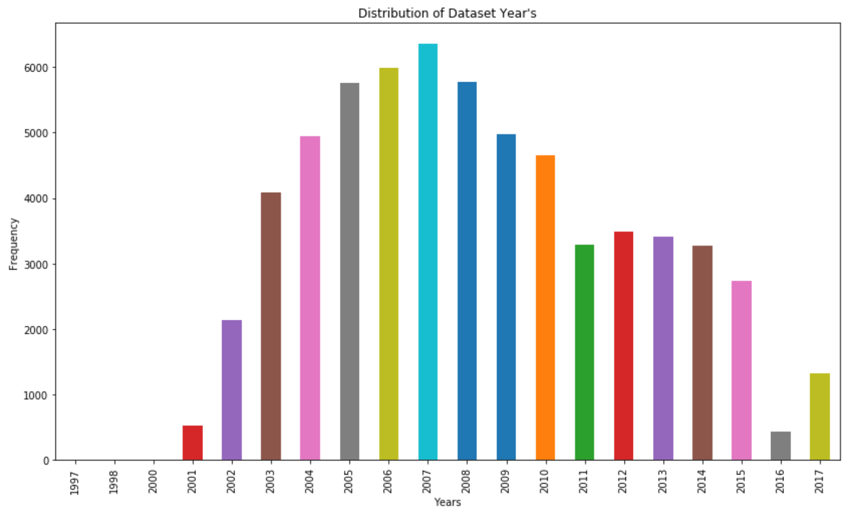


Figure 4 Booking Year Distribution

There is a significant increase in crimes around mid to late 2000’s

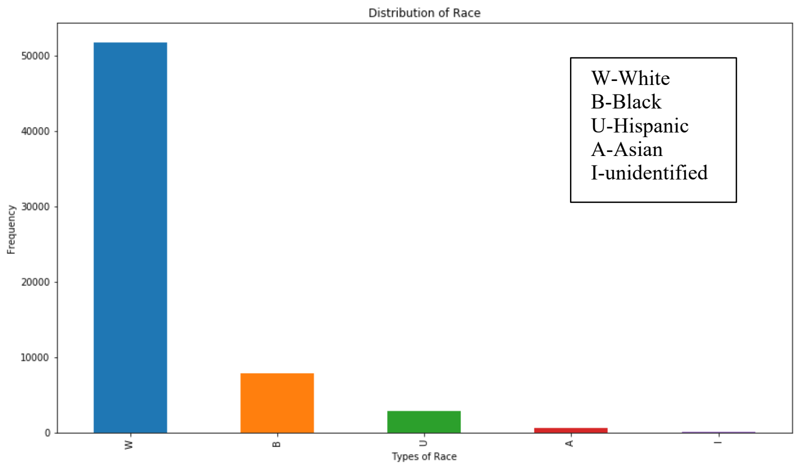


Figure 5 Race Distribution

There is significant pattern with race – White

The observations from the histograms and the statistical inferences gives us enough evidence to further investigate the data. However to learn commonalities of crime progression between criminals, we have built a network plot which might give us evidence to build a prediction model to detect crime reoccurrences.

<u>3.3 Correlation and Progression Analysis</u>

The ordinary correlation analysis does not reveal the directionality of the crime progression. To find out the crime progression which may shed some lights on the prediction of crimes that occur, we used a graph-based progression analyses, one with pairwise progression between two crimes, and the other the temporal progression from each crime type.

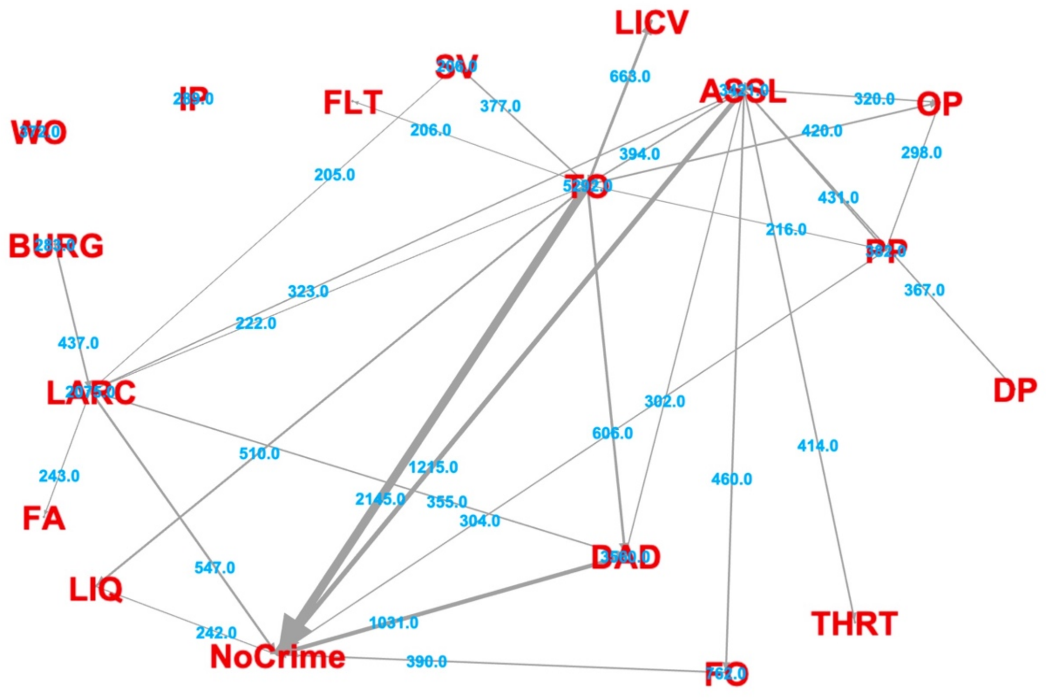


Figure 6 Crime Network with pairwise frequency and direction

The above network plot gives us pair wise progression frequencies of the crimes. The thicker arrows represent higher frequencies. Traffic offence to NoCrime which indicated that the traffic offenders do not necessarily commit further crimes. Public peace → Assault and Assault → NoCrime are some of the popular crime progression. The people whose first crime is Public peace has higher chance of committing assault as second and NoCrime as third.
In order to find out the progression stemming from each crime, we also analyzed the dataset with a rooted tree that shows the transition from a crime c1 to c2, from c2 to c3, etc. In each transition, we also provide the frequency of records(people) involved in the transition between two crimes and the average age of these people. See figure 7 to see the graph analysis.

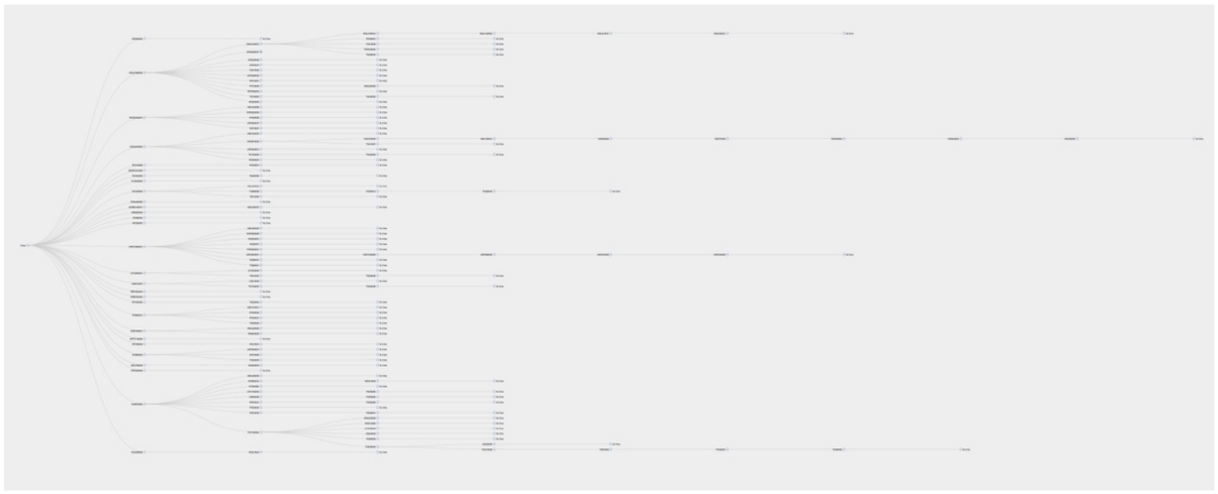


Figure 7 Crime progression tree

This tree-based crime transition analysis helps us to understand how offenders move from one crime to another over time and the average of criminals involved in the progression

As seen in Fig 8, each crime in the network plot is represented by its acronym and the two numbers next to acronym represents the frequency of crime at that level and average age of the people who committed the crime ex. ASSL (47) # (31)-assault with 47 instances nada vg age of 32. By taking only crimes with more than 20 children, the network plot has been thresholded by most frequent crimes progressions. People in the dataset have many common traits in crime progression, we need to find evidence of such patterns to build a prediction model for crime.

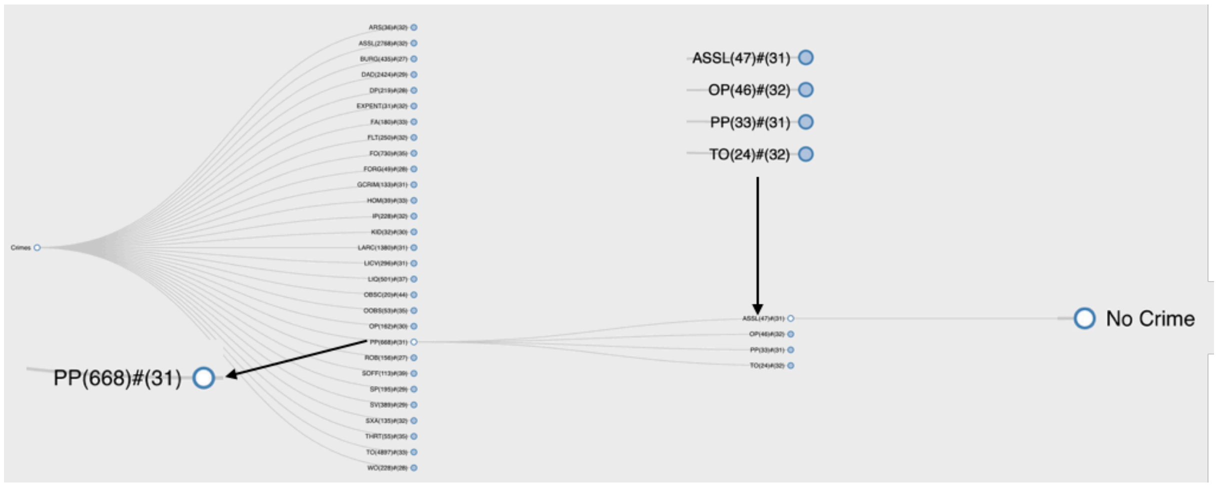


Figure 8 Public peace Crime Progression

We learned that Public Peace Disturbance to Assault and Assault to NoCrime has higher pair wise frequency. After plotting the unique progression paths, the above network graph shows us that there are 668/16841 people had first crime as Public Peace disturbance, out of these 668 people 47 moved to Assault as second crime. None of the 47 comitted any other crime further.

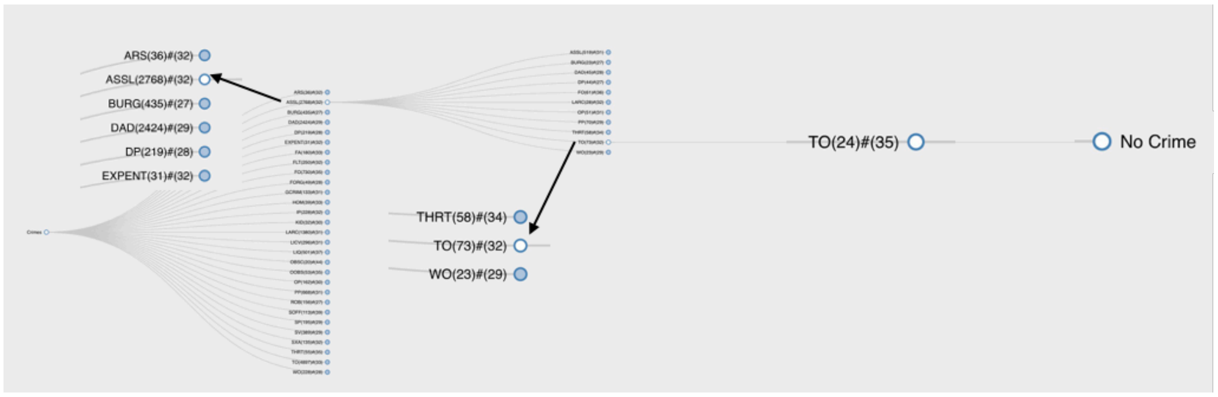


Figure 9 Assault Crime Progression (Assault → Traffic offense → Traffic offense)

If we considered the people who committed assault as the first crime, there are 2768/16841 such cases with an average age of 32 years. The most frequent second crime of these 2768 people is assault with 519 instances with average age of 31. And second most frequent crime is Traffic offence with 73 instances and average age of 32.of these 73 people 24 committed traffic offence again and stopped there.

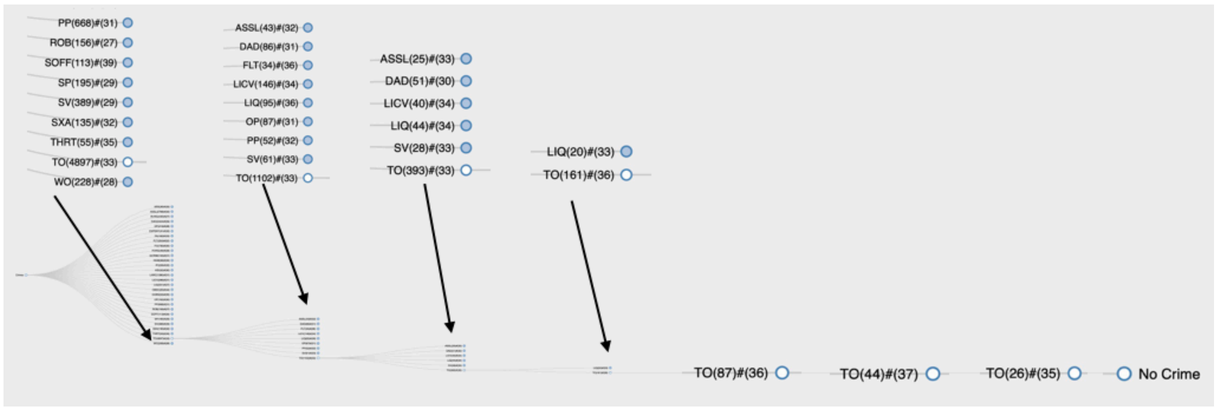


Figure 10 Traffic Offence Crime Progression

The most frequent first crime in dataset is **Traffic Offence** with a frequency of 4897 and average age of 33. And of these people second most frequent crime **Traffic Offence** with an average age of 33 and frequency of 1102/4897. The trend continues with **Traffic Offence** again as their 3rd, 4th, 5th,6th,7th. This trend of frequent criminals ended up with no crime at 8th progression.

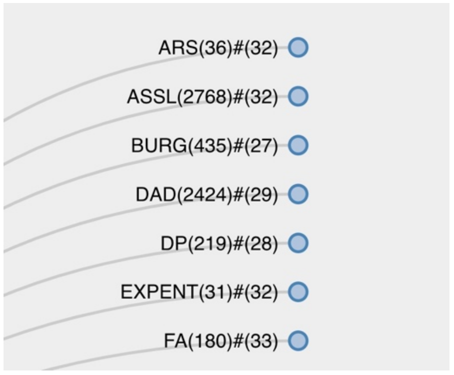


Figure 11 Few first offence stats

This graph strengthens our assumption that there are reoccurring patterns in criminal behavior. Age plays important role in type of the crime, the people who committed burglary or drug offences has significantly less average age compared to other crime types. With all these findings, it is apparent that causal relationships can be established on the persons criminal history and his future offences. In the following section, we make a ML prediction model to be able to within a time window, e.g. five years.

4. PREDICTION MODELING USING DEEP LEARNING OR DEEP NEURAL NETWORK:

The unpublished previous work on this data used a schema that treated each level occurrence and no crime occurrence as a binary label Yes/No in total they created 4 labels, 3 for level 1,2,3

and Any_Level(Yes/No). And applied the machine learning method SVM on the personal criminal charge history to create 3 different models for the next 5-year prediction.
The major concern with this work is that they considered levels to be independent of each other. By that I mean it does not guarantee that a person who got classified 'NO' in Level 1 classification commits no crime in the next 5 years, he still has a probability of committing level 2 or level 3. Rather than treating the classification as binary, our idea is to treat it as multiclass classification which takes NoCrime Level 1,2,3 together and utilize neural networks capability to exploit the learning.

We compare our model's performance with the results from previous work done on this data. In this section, we describe the structure and learning method of the prediction model. We used a Deep Neural Network to build the prediction model. Neural network model offers the flexibility to build a small or large network based on the size of the data. The basic foundational unit of a neural network is the *neuron*, which is actually conceptually quite simple.

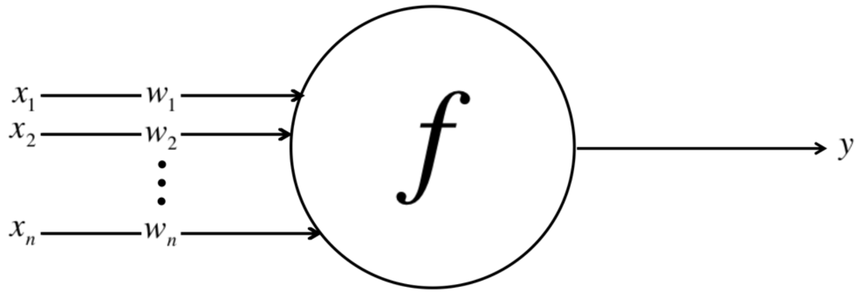
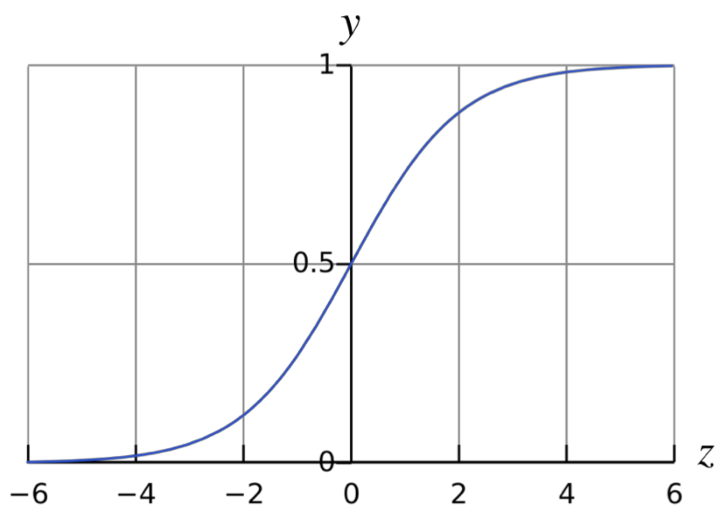


Figure 12 Schematic for a neuron in a neural net

Each neuron has a set of inputs, each of which is given a specific weight. The neuron computes some function on these weighted inputs. A linear neuron takes a linear combination of the weighted inputs. A sigmoidal neuron does something a little more complicated It feeds the weighted sum of the inputs into the *logistic function*. The logistic function returns a value between 0 and 1. When the weighted sum is very negative, the return value is very close to 0. When the weighted sum is very large and positive, the return value is very close to 1. Whatever function the neuron uses, the value it computes is transmitted to other neurons as its output. In practice, sigmoidal neurons are used much more often than linear neurons because they enable much more versatile learning algorithms compared to linear neurons.



$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$y = \frac{1}{1 + e^{-z}}$$

Figure 13 The function of a sigmoidal neuron

The ANN is constructed by having a large number of these neurons working in parallel and connecting some neurons to others through weighted connections, creating a weighted and directed network of different layers. To make this structure easier to visualize, I've included a simple example of a neural net below. We let w(k)i,j be the weight of the link connecting the *ith* neuron in the *kth* layer with the *jth* neuron in the *k+1*st layer. It is by adjusting these weighted connections and the internal activations of the neurons the ANN can be improved or trained. Usually the network cycle through a set of training data sufficiently many times, until the weights have been adjusted enough to produce the desired output.

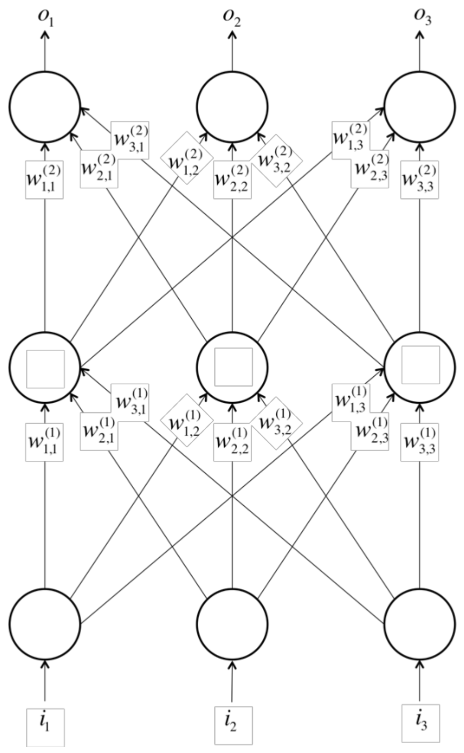


Figure 14 a neural net with 3 layers and 3 neurons per layer

A common way of learning for a neural network is the process of back propagation. This is a method of dynamic adjustment based on providing feedback to the network, initiated from a difference between the desired output and the current output. The weights of the interconnected neurons are adjusted depending on the degree they contribute to the error and this process is repeated in cycles until the network achieves a desired accuracy of classification.

An important part of the design of the network model, is choosing the loss function. The loss function represents the price paid for inaccuracy in predictions made by the neural network. By minimizing the loss function during the training process the error of the network also will be minimized. For classification problems, one of the most popular choices for loss function is the softmax cross entropy functions, defined as:

$$\mathrm{Hy}'\ (\mathrm{y}) = -\sum_{i} \mathrm{y}'\mathrm{i}\log(\mathrm{yi})$$

where y is the predicted probability function and y' is the true distribution.

Gradient descent is a common optimization method for minimizing the loss function in machine learning algorithms. The idea is to follow the steepest descent of a function F(X), which will be proportional to the negative gradient, −∇F(X) to find minima of the function. This is illustrated in Figure 16 where the red line shows the direction in each iteration step for the gradient descent algorithm, towards the minimum in the center.

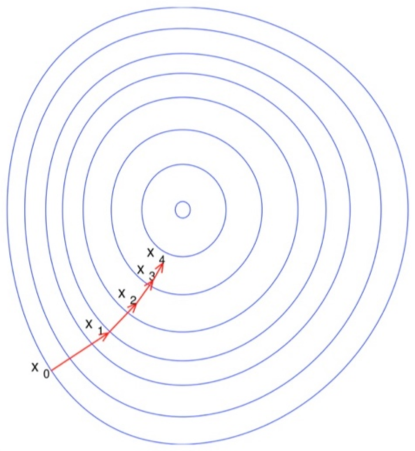


Figure 15 Illustration of the gradient descent method

4.1 TensorFlow

TensorFlow (TF) is an open source API developed by Google mainly for Machine Learning and Deep Learning, but it is also applicable for other numerical computations. Google uses TF in some of its commercial products, such as their speech recognition API and Gmail, but it is also used for research. The framework can be used both as backend, with C++, and frontend with Python.

One of the advantages of using TF are the many built in functions. We are using the high-level API Estimators, which can be customized if needed. For this project, we have built our own estimator with a custom model. This model is utilizing the TF built in functions *tf.layers* and *tf.losses*, defining the structure and loss function of our neural network.

4.2 Loss function

I have used the softmax cross entropy function in the *tf.losses* module. The parameters available for the loss function include weights, scope, reduction and smoothing. Weights can be used to bias the network toward a certain class, e.g. to combat overfitting. The optimization method is also a parameter for the network.

4.3 Layers

The layers in the *tf.layers* module include the standard Fully connected layers convolution. Other commonly used layers are *dropout* layers. We work with the *dropout* layers and Fully connected layers for this project. Dropout layers and L2 regularization are used to prevent overfitting, where the user provides a rate between 0 and 1 at which the model will randomly drop neurons. And we set a beta value for regularization to penalize the largest weight so the model would stop overfitting.

4.4 Other parameters

The batch size of the network is the number of data points passed through the network in each propagation. After one batch is processed, the weights are updated.

The learning rate is a measure of how much the model is inclined to abandon the belief that a certain feature is most predominant, and instead choose another feature. Both too high and too low learning rates can make the training process longer.

Parameters for the training process, such as the number of steps for the training, are also available for user modification.

In order to construct a supervised learning problem using deep learning we need to preprocess the data to maximize data utilization and also create labels

4.5 Pre-processing: Feature Engineering

Due to confidential nature of data, the area and the personal information is not revealed. The initial raw data consist of 6 features which consist of all the charge history of each person in the collected 20 years. When a person commits a crime, we have record of booking, each of these bookings can hold multiple counts of charges (NCIC Crime Code) depending on the type of the crimes committed in the given booking.

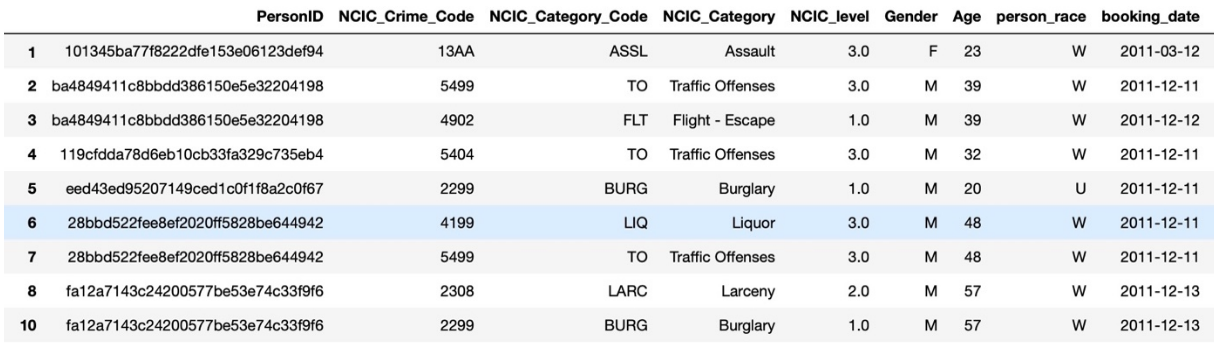
The NCIC crime code can be expanded in to NCIC Charge category, Crime Charge and ICE criminal offense Level. Table4 represents a compilation of criminal offense codes that the Department of Justice has determined to be violent or drug-related or nonviolent and not drug related [8]. With the help of opensource FBI documents, the "NCIC Crime Codes" were expanded in to three new features NCIC Category Code, NCIC Category and NCIC Level. See Table 5 for description of these three features.

Table 4 NCIC Code description

Offense Code Categories	**Violent & Drug Related Description of Crime**
09XX	Homicide crimes
10XX	Kidnapping crimes (kidnapping, abduction, false imprisonment)
11XX	Sexual Assault crimes (rape, sexual assault including sodomy, statutory rape)
12XX	Robbery (forcible purse snatching, carjacking (armed), aggravated assault, simple assault, intimidation (including stalking))
13XX	Assault (aggravated, simple, intimidation)
16XX	Threat – Terrorism
21XX	Extortion crimes (threat of injury to person only)
35XX	Drug crimes (manufacturing, distributing and sale of drugs, possession and smuggling of drugs)
52XX	Weapons crimes (only including the threat to bomb and the threat to burn)

Table 5 Expanded features description.

Features	Description
NCIC Category	Descriptive crime name derived from NCIC Code (e.g Traffic Offense, Larceny, Dangerous Drugs, etc.)
NCIC Category Code	An Acronym of the type of crime based on NCIC Code (e.g. TO, LARC, DAD etc.)
NCIC Level	Seriousness of the crime. Has 3 levels L1, L2, L3. (L1 is the worst crimes such as Homicide, while L3 includes Traffic Offense.)



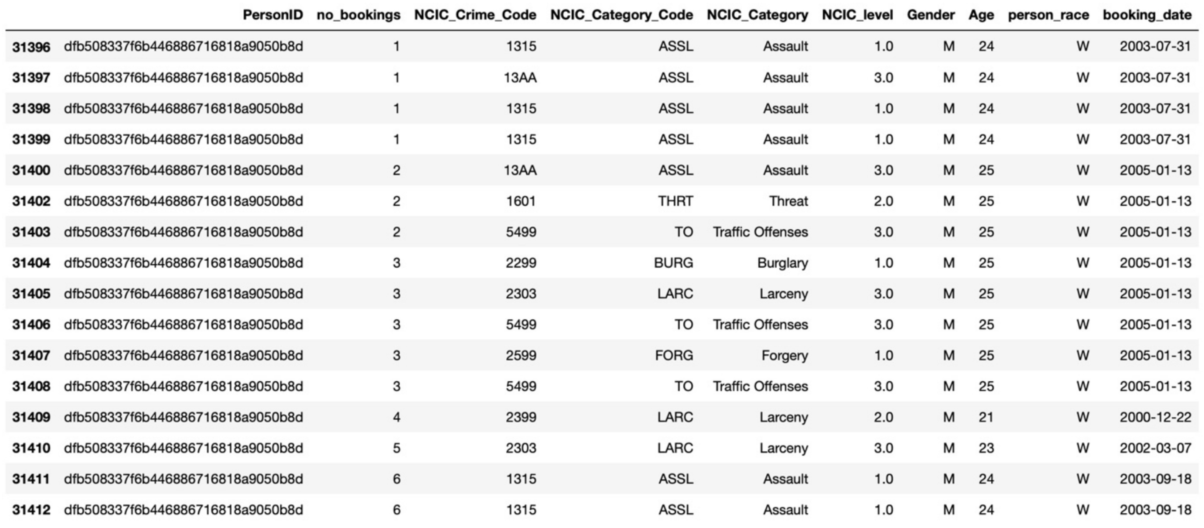
	PersonID	NCIC_Crime_Code	NCIC_Category_Code	NCIC_Category	NCIC_level	Gender	Age	person_race	booking_date
1	101345ba77f8222dfe153e06123def94	13AA	ASSL	Assault	3.0	F	23	W	2011-03-12
2	ba4849411c8bbdd386150e5e32204198	5499	TO	Traffic Offenses	3.0	M	39	W	2011-12-11
3	ba4849411c8bbdd386150e5e32204198	4902	FLT	Flight - Escape	1.0	M	39	W	2011-12-12
4	119cfdda78d6eb10cb33fa329c735eb4	5404	TO	Traffic Offenses	3.0	M	32	W	2011-12-11
5	eed43ed95207149ced1c0f1f8a2c0f67	2299	BURG	Burglary	1.0	M	20	U	2011-12-11
6	28bbd522fee8ef2020ff5828be644942	4199	LIQ	Liquor	3.0	M	48	W	2011-12-11
7	28bbd522fee8ef2020ff5828be644942	5499	TO	Traffic Offenses	3.0	M	48	W	2011-12-11
8	fa12a7143c24200577be53e74c33f9f6	2308	LARC	Larceny	2.0	M	57	W	2011-12-13
10	fa12a7143c24200577be53e74c33f9f6	2299	BURG	Burglary	1.0	M	57	W	2011-12-13

Figure 16 Dataset with expanded features.

Fig.16 is the data after expansion of NCIC code. It consists of date of the booking, the charges information, levels of the charges, and demographics such as persons race, gender and age.

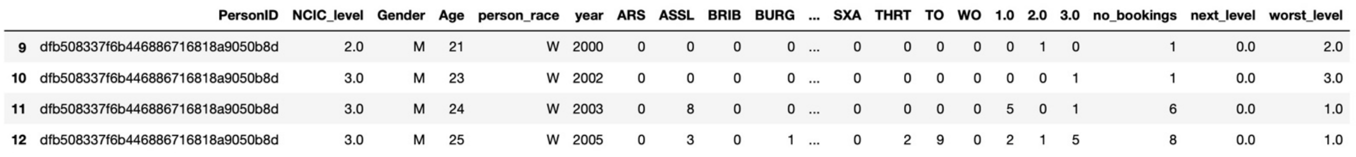
The classical supervised learning problem is to construct a classifier that can correctly predict the classes of new objects given training examples of old objects. The dataset consists of a grainy charge history of each person. Some of these charges are separated by days and some are separated by years. In order to create labels for the next given window of years, data aggregation is done on a person level for each year. Namely, each person's information in a year is aggregated by adding up crime type-related features. New features total no of bookings per year, total no of types of levels per year, most serious offence per year are created Fig17.

Fig. 17(a)



	PersonID	no_bookings	NCIC_Crime_Code	NCIC_Category_Code	NCIC_Category	NCIC_level	Gender	Age	person_race	booking_date
31396	dfb508337f6b446886716818a9050b8d	1	1315	ASSL	Assault	1.0	M	24	W	2003-07-31
31397	dfb508337f6b446886716818a9050b8d	1	13AA	ASSL	Assault	3.0	M	24	W	2003-07-31
31398	dfb508337f6b446886716818a9050b8d	1	1315	ASSL	Assault	1.0	M	24	W	2003-07-31
31399	dfb508337f6b446886716818a9050b8d	1	1315	ASSL	Assault	1.0	M	24	W	2003-07-31
31400	dfb508337f6b446886716818a9050b8d	2	13AA	ASSL	Assault	3.0	M	25	W	2005-01-13
31402	dfb508337f6b446886716818a9050b8d	2	1601	THRT	Threat	2.0	M	25	W	2005-01-13
31403	dfb508337f6b446886716818a9050b8d	2	5499	TO	Traffic Offenses	3.0	M	25	W	2005-01-13
31404	dfb508337f6b446886716818a9050b8d	3	2299	BURG	Burglary	1.0	M	25	W	2005-01-13
31405	dfb508337f6b446886716818a9050b8d	3	2303	LARC	Larceny	3.0	M	25	W	2005-01-13
31406	dfb508337f6b446886716818a9050b8d	3	5499	TO	Traffic Offenses	3.0	M	25	W	2005-01-13
31407	dfb508337f6b446886716818a9050b8d	3	2599	FORG	Forgery	1.0	M	25	W	2005-01-13
31408	dfb508337f6b446886716818a9050b8d	3	5499	TO	Traffic Offenses	3.0	M	25	W	2005-01-13
31409	dfb508337f6b446886716818a9050b8d	4	2399	LARC	Larceny	2.0	M	21	W	2000-12-22
31410	dfb508337f6b446886716818a9050b8d	5	2303	LARC	Larceny	3.0	M	23	W	2002-03-07
31411	dfb508337f6b446886716818a9050b8d	6	1315	ASSL	Assault	1.0	M	24	W	2003-09-18
31412	dfb508337f6b446886716818a9050b8d	6	1315	ASSL	Assault	1.0	M	24	W	2003-09-18

Fig. 17(b)



	PersonID	NCIC_level	Gender	Age	person_race	year	ARS	ASSL	BRIB	BURG	...	SXA	THRT	TO	WO	1.0	2.0	3.0	no_bookings	next_level	worst_level
9	dfb508337f6b446886716818a9050b8d	2.0	M	21	W	2000	0	0	0	0	...	0	0	0	0	0	1	0	1	0.0	2.0
10	dfb508337f6b446886716818a9050b8d	3.0	M	23	W	2002	0	0	0	0	...	0	0	0	0	0	0	1	1	0.0	3.0
11	dfb508337f6b446886716818a9050b8d	3.0	M	24	W	2003	0	8	0	0	...	0	0	0	0	5	0	1	6	0.0	1.0
12	dfb508337f6b446886716818a9050b8d	3.0	M	25	W	2005	0	3	0	1	...	0	2	9	0	2	1	5	8	0.0	1.0

Figure 17 (a) Raw data of a person & (b)Aggregated data of the person per year

The aggregated data per person per year has 24404 records in it. From this it is straight forward to create expanding pooling and windowed data.

<u>4.6 Data representation</u>

We have a data ranging from year 1997 to 2017. The dataset is highly imbalanced 42.2% of the population has just one crime in their record, 60% of the population has less than or equal to 2 crimes. As we have to create labels, we need to look 5 years into the feature and take the worst crime level as our label. i.e we filter the data at 2012 and look at the next 5 years and create labels No_crime or Level1, Level 2 or Level3. After doing there is a huge imbalance in created labels. In order to solve the imbalance - data augmentation, over sampling and under sampling should be performed. As a first step I used expanded pooling to augment the data and create more minority classes. We are using an expanding window with a stride length of 1 year starting from first booking year to last year for 5-year future label. i.e 2012 of each person to create aggregate feature vectors at every year along with label for next 5 years.

Let’s consider the person Fig18 who has two records first in Year1 and second in Year7. We are going to start at Year1 and take all the features as our X1-in this case its crime1. Look ahead next five years and create label NoCrime. Now we move to Year2 and take the aggregated features of Year1 and Year2 as our feature vector X2 and look ahead to Year3-Year7 and create label crime2. In the same manner we continue until year 7 to create pooled feature vectors at each year.

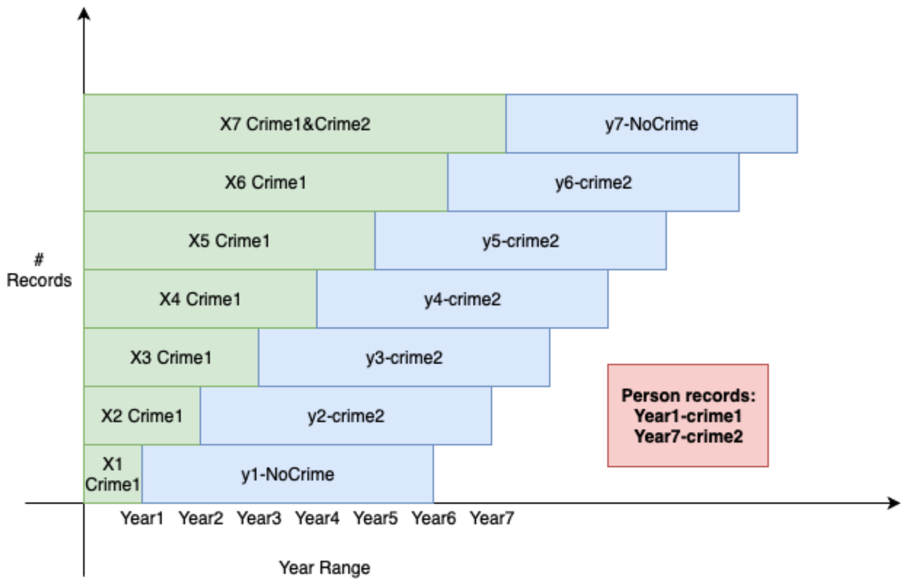
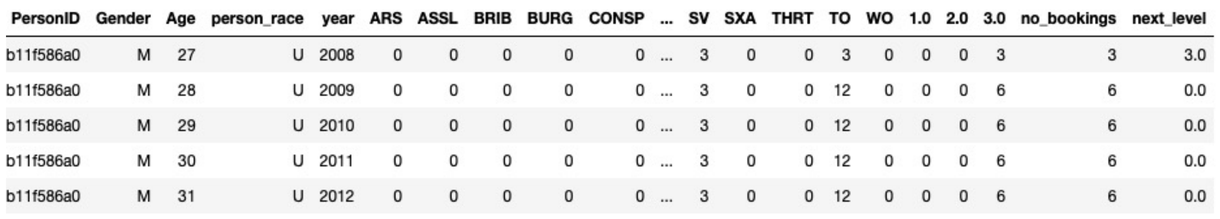


Figure 18 Data pooling with n-year lookahead window



PersonID	Gender	Age	person_race	year	ARS	ASSL	BRIB	BURG	CONSP	...	SV	SXA	THRT	TO	WO	1.0	2.0	3.0	no_bookings	next_level
b11f586a0	M	27	U	2008	0	0	0	0	0	...	3	0	0	3	0	0	0	3	3	3.0
b11f586a0	M	28	U	2009	0	0	0	0	0	...	3	0	0	12	0	0	0	6	6	0.0
b11f586a0	M	29	U	2010	0	0	0	0	0	...	3	0	0	12	0	0	0	6	6	0.0
b11f586a0	M	30	U	2011	0	0	0	0	0	...	3	0	0	12	0	0	0	6	6	0.0
b11f586a0	M	31	U	2012	0	0	0	0	0	...	3	0	0	12	0	0	0	6	6	0.0

Figure 19 Pooled dataset for each year from a person’s criminal booking history

If a person has more than one booking in the dataset, this feature augmentation method helps us to create more positive and negative records. By using random under sampling of majority class and SMOTE over sampling of minority class we can create balanced data set for training. The other method to use a window to select the latest aggregated records.

4.7 Data Selection

4.7.1 Dynamic Window Data Model

We set latest records length to a desire number n and use this to filter the records from the dataset.

For the analysis we chose to use 3 filters and no filter on the record selection. Although the number of minority classes are increased in number from the raw data, we use SMOTE over sampling for further over sampling of minority classes.

In the instance of data selection where we didn't use filter for selection of data the class imbalance is very large as we created more majority classes compared to minority classes. So we used random majority under sampling to make the data balanced.

Table 6 Dynamic Window Data Model sampling specifications

Dynamic Window Data Model Filter Specification	Training Sample Size							
	Before Sampling				After Sampling			
	NoCrime	Level 1	Level2	Level3	NoCrime	Level1	Level2	Level3
2 Latest Years - SMOTE	17401	823	294	1098	17401	3000	3000	3000
3 Latest Years- SMOTE	13584	1269	452	1675	13584	4000	4000	4000
4 Latest Years- SMOTE	31017	1714	613	2204	31017	6000	6000	6000
No Filter – under sampled majority class	52828	4312	1532	4560	10404	4312	1532	4560

The pooled data contains early records information which does not need in building the predictive model. To take advantage of the pooled data we are going to use a window of years to select the latest aggregated records. By doing this we can increase the number of records in training sample. This method helps in increasing the records to an optimal level while reducing redundancies.

Table 7 Features used in Dynamic Window Data Model

Features Used	**Description**
No of Bookings	Total number of bookings up to the pooled year.
No of Level 1 crime?	Total number of level 1 bookings up to the pooled year.
No of Level2	Total number of level 2 bookings up to the pooled year.
No of Level3	Total number of level 3 bookings up to the pooled year.
Age	Age of the person at the current pooled year.
Booking Race	Race of the person
NCIC Category - 42 feature columns	An Acronym of the type of crime based on NCIC Code. It has a total of 42 levels with level information in the cell.

4.7.2 Fixed Window Data Model

The previous work in the literature lacks the temporal aspect and studies were based only on social-economic features. We tried to build model by incorporating features designed to capture temporal aspect of the crimes. In this model we are going to use last rows of aggregated features and label pair X_n & Y_n of each person records from the previously discussed pooled data. This way we are using one record per person. In order to capture the temporal aspect, we built 3 new features Timespan, Time since last crime, Avg age.

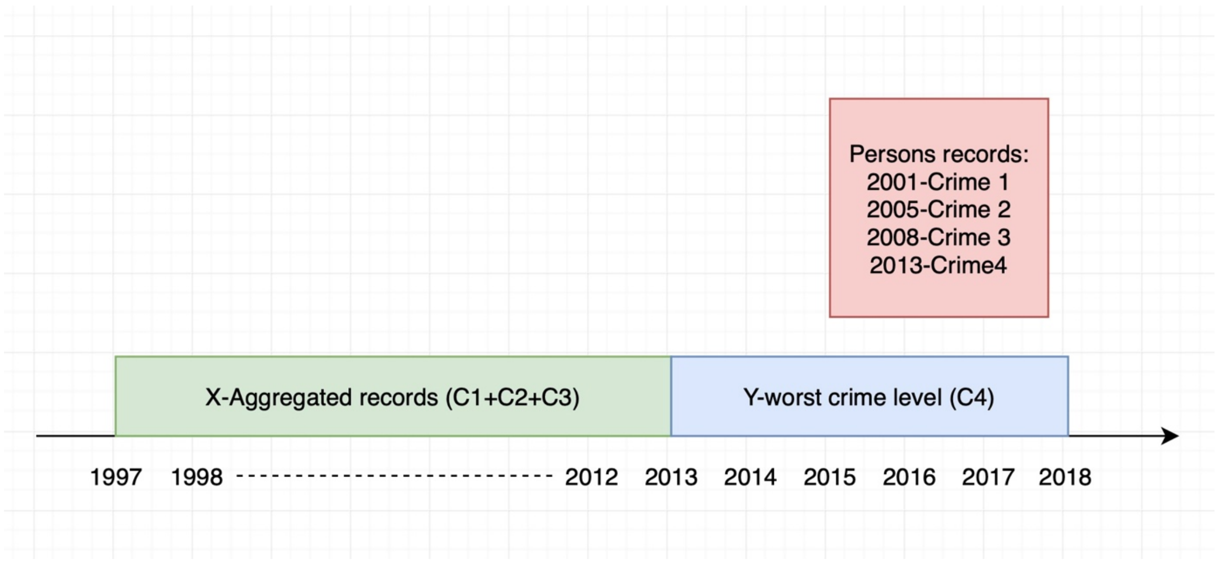


Figure 20 Aggregated representation of a person's criminal history with fixed window data model

Table 8 Fixed Window Data Model

Features Used	Description
No of Bookings	Total number of bookings in the history of the person.
No Level 1	Total number of level 1 bookings in the history of the person.
No Level2	Total number of level 2 bookings in the history of the person.
No Level3	Total number of level 3 bookings in the history of the person.
Timespan	Average days between 2 consecutive crimes
Avg age	Avg age of the person throughout the records
Time since last crime	Total number of days lapsed from last crime
Booking Race	Race of the person
Min Age	Age from the persons first record
Max Age	Age of the person from 2012
NCIC Category 42 feature columns	An Acronym of the type of crime based on NCIC Code. It has a total of 42 levels with level information in the cell.

4.8 Network architecture

We chose a relatively simple structure of our NN, with four hidden fully connected layers. The output layer has 4 neurons condensing the individual weights of the neurons in the network into network predictions Fig21. The problem with the architecture is that the model learning is based on loss function, when we have a imbalanced dataset and as the optimizer priority is to reduce loss value. The model prioritizes learning on majority class. And miss classifies minority classes.

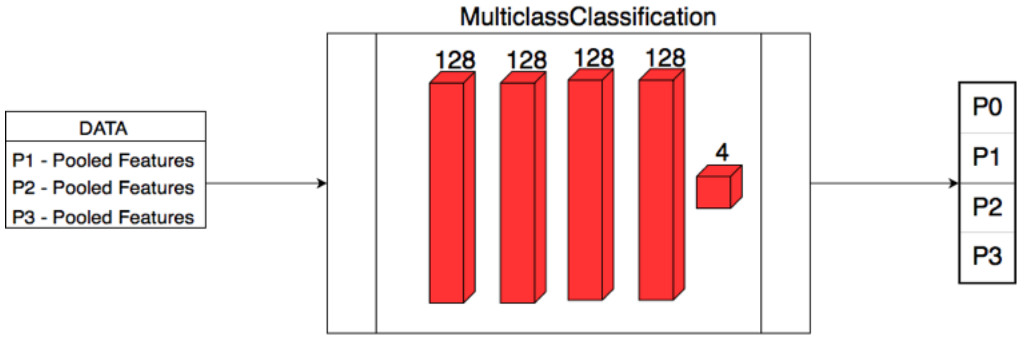


Figure 21 Neural Network model pipeline

To address the class imbalance problem, we developed a weighted loss function Eq1, Eq2 and Fig22. If each outcome's actual probability is *pi* but model is estimating probability as *qi.* In this case, each event will occur with the probability of *pi* but surprisal will be given by *qi* in its formula. Now, weighted average surprisal, in this case, is nothing but cross entropy(ce) and it could be scribbled as

$$ce = \sum_{0}^{n} \mathrm{p}_i \log\left(1/\mathrm{q}_i\right)$$

Eq1: Cross entropy

$$custom\ ce = (1-w)\sum_{0}^{n} \mathrm{pb}_i \log\left(\frac{1}{\mathrm{qb}_i}\right) + w\sum_{0}^{n} \mathrm{pm}_i \log\left(\frac{1}{\mathrm{qm}_i}\right)$$

Eq2: Weighted cross entropy

We treat the prediction as two different problems. The class bias is high on label 0, so we created custom binary labels, whenever the person is positive on any of the label 1,2or3 we take it as a binary label 1 and do the prediction, look at table8 and table9. We perform a hierarchical classification where we just take the positive cases from the binary classification and further classify them in to the Labels 1,2or3. We separate the cross-entropy loss on these two predictions pb_i actual binary probability, qb_i is model estimated probability and pm_i is actual multiclass probability, qm_i is model estimated probability. We assign weight (1-w) and (w) to them to control the loss from each of these classifications, weight parameter which can be adjusted between 0 and 1 to prioritize learning from binary or multiclass. This removes the skewed class balance on multiclass prediction.

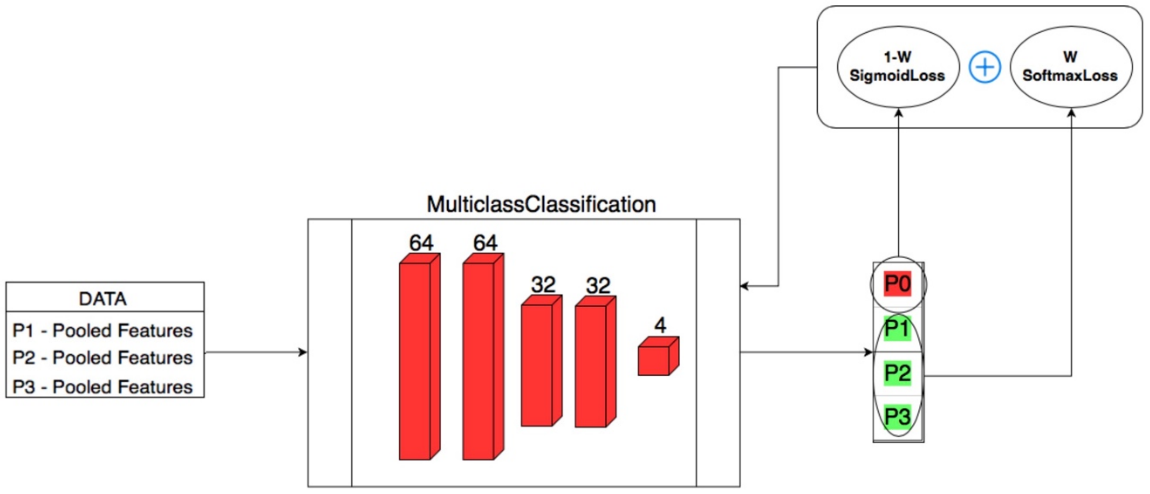


Figure 22 Neural Network model pipeline with weighted loss.

Table 9 Label transformation for regular prediction model

#	Ground Truth Label	One Hot Encoded Label Vector			
		No_crime	Level 1	Level2	Level 3
Person1	3	0	0	0	1
Person2	0	0	0	0	0
Person3	1	0	1	0	0
Person4	2	0	0	1	0
Person5	0	0	0	0	0

Table 10 Label transformation for the custom loss model

#	Ground Truth Label	Binary Label (crime= Yes or No)	Crime = Yes		
			Level 1	Level 2	Level3
Person1	3	1	0	0	1
Person2	0	0	0	0	0
Person3	1	1	1	0	0
Person4	2	1	0	1	0
Person5	0	0	0	0	0

<u>4.9 Transforming the training data set into a Supervised Deep Learning Problem:</u>

A custom function is called which takes in two arguments. The dataset, number of outputs. Then this function creates a 2-dimensional array which is the input array X where the first dimension is instances, second is features and output array Y. The input data of the network is given in batches. In order for the network to use the data for training, it has to be reshaped into tensor format. In our case, we have linear feature vector with length of 49 and varies depending on the data model. The resulting tensor passed through the network changes form during the process, but the initial shape is [batch_size,49]. Our choice of batch size is 2000 for this model, as the network usually trains faster for smaller batch sizes.
There is no one right answer to how to choose the network parameters. A common method when finding good parameters is to simply try different values and compare the results. The parameters used in the model that we evaluated are listed in Tables. Comments on the choices and thoughts of improvement are listed in the discussion section of this report.

The loss function used in our NN is softmax cross entropy, a common choice for NN’s. We are using weighted modification parameters to control learning, will discuss possible improvements regarding the loss function in the discussion section of the report. We settled for a learning rate of 0.005, which was sufficient in our case. For the optimization method, we used gradient descent.
The filter size for record selection (number of records per person) in the training process was a parameter that we investigated when we evaluated the accuracy of the network. The results are presented in the network performance section of the report.

5. EXPERIMENTAL RESULTS

Here we present some results for the model trained to predict the 5-year labels. In these series of graphs, the loss, binary accuracy and multi class accuracy of the network is plotted over its training period. The datasets used for this analysis are selected based on varying window lengths as discussed in Dynamic Window Data Model and used all the pooled records of minority class and randomly under sampled majority class. In the graph we are showing the accuracy and loss for the network, both when it is evaluating data from a test dataset and when it is evaluating on its own training dataset. The discrepancy between the two curves could be a result of imbalance dataset. Which is when a model has learned to classifying data from major class in training dataset but is not as good in the general case as it over fit to certain parameters.

Table 11 Results from the SVM Binary Model (with Fixed window)

Model	**Predicting level**	**Config, T(in years)**	**Accuracy**	**F1 Score**
Model 1	Level1	5	92.5%	0.51
Model 2	Level2	5	91.5%	0.37
Model 3	Level3	5	97.6%	0.87

Table 12 Results from the dynamic window model with DNN Learning.

Data Selection Techniques (pool model with 5 year lookahead window)	**Binary Train Accuracy** (crime= Yes or No)	**Multi Train Accuracy (crime = Yes then L1, L2, L3)**	**F1-score Train**	**Binary Test Accuracy** (crime= Yes or No)	**Multi Test Accuracy (crime = Yes then L1, L2,L3)**	**F1-score Test**
2 Latest Years - SMOTE	0.840	0.828	0.73	0.822	0.800	0.34
3 Latest Years-SMOTE	0.731	0.714	0.65	0.743	0.705	0.32
4 Latest Years-SMOTE	0.748	0.737	0.60	0.808	0.777	0.34
No Filter – under sampled majority class	0.693	0.617	0.531	0.734	0.694	0.355

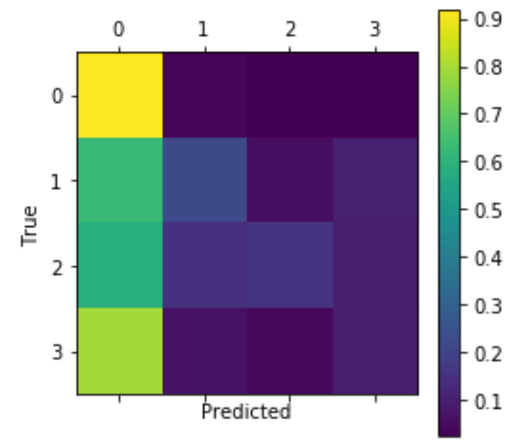
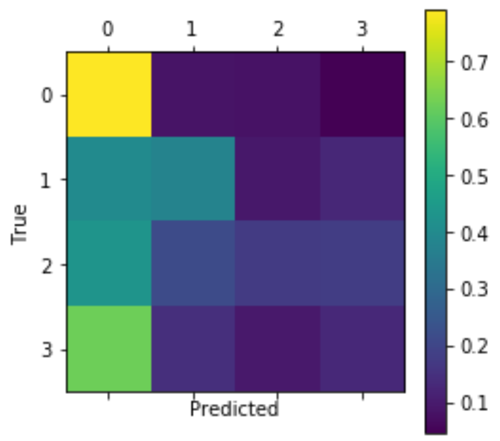
Table 13 Optimal parameters of NN model

Input Data Shape	[Batch Size, 49]
Batch size	2000 samples
Number of layers	5 layers (4 Hidden, 1 output layer)
Layers configuration	128, 64, 64, 64, 4
Weight of loss function (W=Minority weight) (1-W=Majority weight)	0.8
L2-Regulirization Beta	0.005
Learning rate	0.005

Table 14 Sample Sizes used for training NN

Dynamic Window Data Model Filter Specification	Training Sample Size							
	Before Sampling				After Sampling			
	NoCrime	Level1	Level2	Level3	NoCrime	Level1	Level2	Level3
2 Latest Years - SMOTE	17401	823	294	1098	17401	3000	3000	3000
3 Latest Years-SMOTE	13584	1269	452	1675	13584	4000	4000	4000
4 Latest Years-SMOTE	31017	1714	613	2204	31017	6000	6000	6000
No Filter – under sampled majority class	52828	4312	1532	4560	10404	4312	1532	4560

Table 15 Confusion Matrix for NN training with different window data models (with 2 and 3 year crime history data)

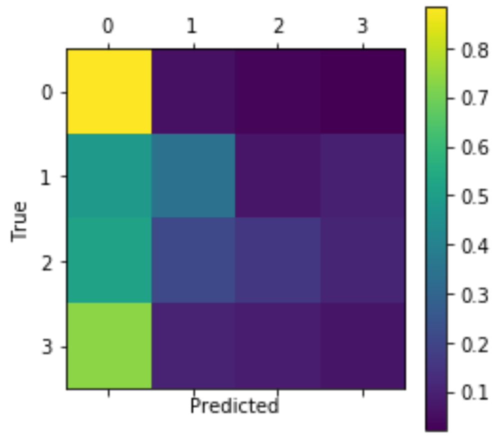
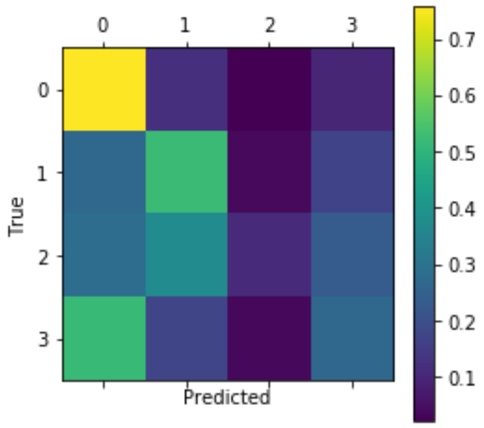
2 Latest Years - SMOTE	3 Latest Years - SMOTE
Confusion Matrix Test	Confusion Matrix Test
	

12592	474	336	311
594	208	52	97
180	46	48	30
984	83	52	119

10850	1146	1112	605
381	361	87	122
131	65	53	55
777	180	118	163

Although the results show that there are a lot of miss classification on the binary and multiclass labels, we can see that the model can still classify the minority classes well, and the mis classifications could be due to the pooling method we used, where we create lot of majority classes that are similar to minority classes on many features.

Table 16 Confusion Matrix for NN training with different window data models (with 4 and all years crime history data)

4 Latest Years - SMOTE	No window–under sampled majority class
Confusion Matrix Test	Confusion Matrix Test
	

12131	836	463	283
462	329	68	92
158	64	49	33
908	130	113	87

10393	1693	279	1348
255	498	38	160
86	114	33	71
643	217	46	332

The results of the four models shows us that when we are using 2 latest years the accuracy results are higher while F1-Score being the least. Looking at the No Filter model confusion matrix compared to other three, we see that the accuracy is lower but the f1 score is higher. As records are aggregated year wise and as we have more records per person, we captured the time dependencies of crime progression.

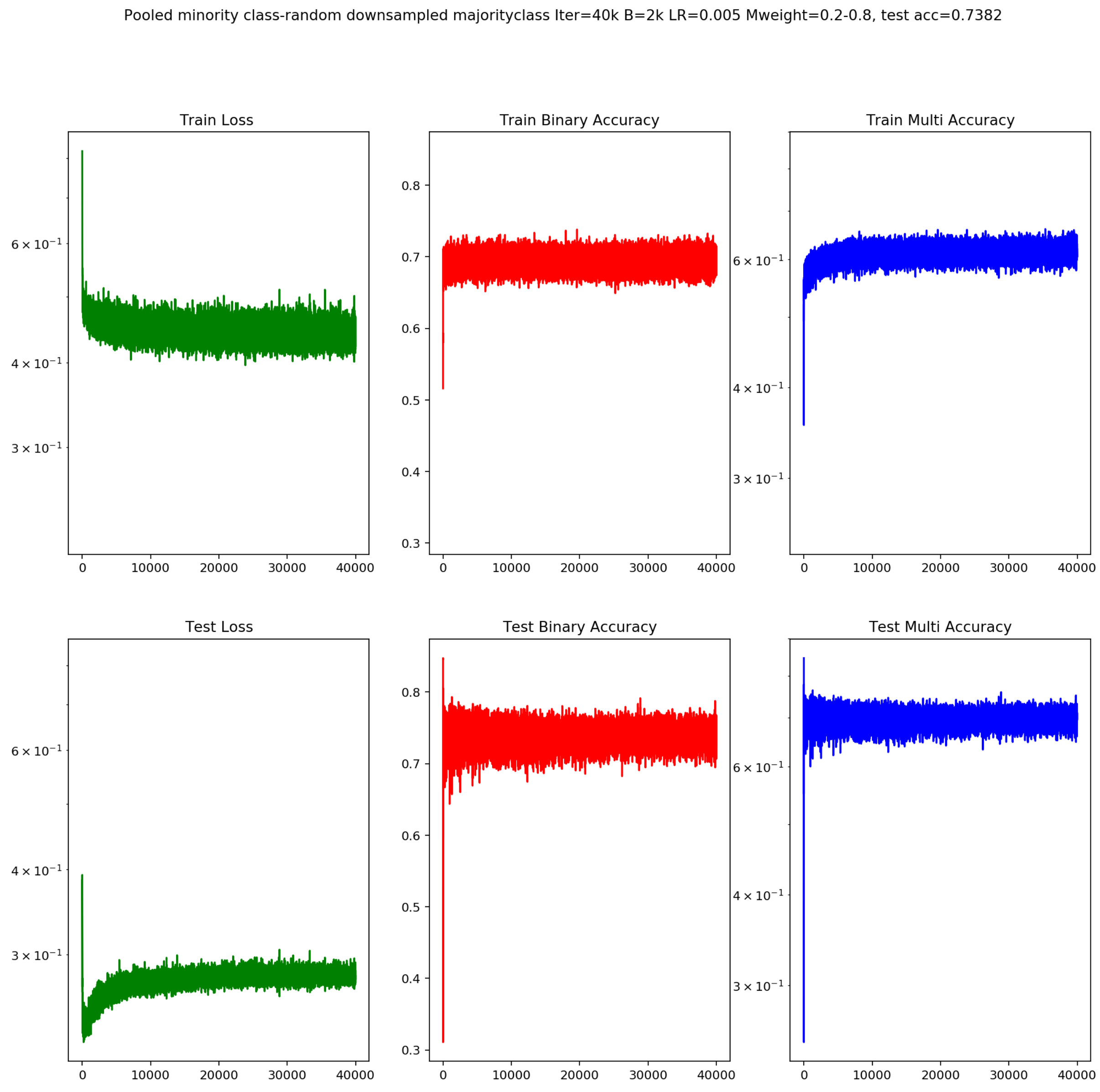


Figure 23 Accuracy and Loss for training and testing NN models

Although the train and test loss curves show that there is small amount of overfitting, but the learning didn't get effected as there is no significant directional change in the curve. The Binary and multi accuracy on both train and test also follow similar structure without any significant jumps or dips.

Here I present results for the model trained to predict the 5-year labels based on the Fixed Window Data Model. In the graph we are showing the accuracy and loss for the network, both when it is evaluating data from a test dataset and when it is evaluating on its own training dataset. In the first graph, the accuracy of the network is plotted over its training period. The dataset used for this specific model contain 10024-person records, each of these feature vectors represent aggregated history of the individual. In the graph we are showing the accuracy for the network, both when it is evaluating data from a test dataset and when it is evaluating on its own training dataset. There different between the train loss curve and test loss curve is due to overfitting on train set, But as both test accuracies don't show any significant deviations w.r.t loss, this over fitting can be ignored.

The Temporal features may have helped to gain significant improvement classification accuracy in binary classification.

Table 17 Optimal parameters of NN model

Input Data Shape	[BatchSize, 57]
Batch size	2000 samples
Number of layers	5 layers (4 Hidden, 1 output layer)
Layers configuration	64, 64, 32, 32, 4
Weight of loss function (W=Minority weight) (1-W=Majority weight)	0.8
L2-Regulirization Beta	0.005
Learning rate	0.0005

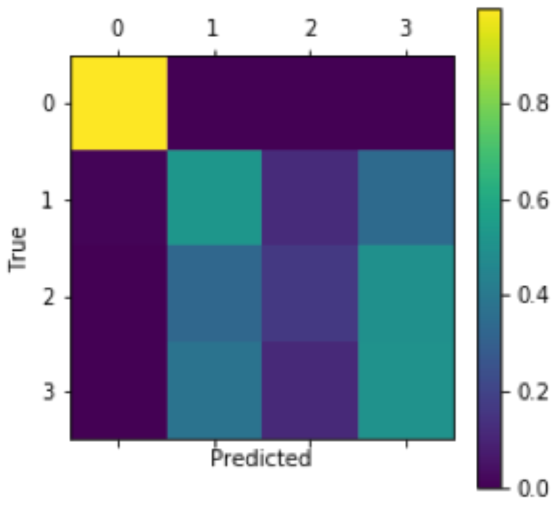
Table 18 Sample Sizes used for training NN

Filter Specification	Training Sample Size							
	Before Sampling				After Sampling			
	NoCrime	Level1	Level2	Level3	NoCrime	Level1	Level2	Level3
Aggregated features at 2012	8740	359	121	526	8740	402	300	582

Table 19 Results from the fixed window model with DNN Learning.

Fixed Window Data Model	**Binary Train Accuracy**	**Multi Train Accuracy**	**F1-score Train**	**Binary Test Accuracy**	**Multi Test Accuracy**	**F1-score Test**
Aggregate records until 2012 - SMOTE	0.999	0.992	0.955	0.997	0.940	0.543

Table 20 Confusion Matrix for NN fixed window data model

Fixed Window Data Model
Confusion Matrix Test


2080	0	0	5
1	47	11	31
0	10	5	15
0	50	15	67

The results of this model are significantly better compared to the. But this model performance cannot be generalized to all the years as it is based on the data aggregated until 2012. In order to build model which can generalize well to any year, further research in to the features and type of neural network architecture is needed.

6. CONCLUSIONS & FUTURE WORK:

Within the scope of this project, we presented a prediction learning model with Dynamic Window based data modeling to answer   whether a specific person will commit a new crime in the future years within n window time. Specifically, we compared different window sizes of considering historical crime data, looking back 2 years, 3 years, 4 years, and all possible years of crime history (1 year to n-5 years). For multi-class crime predictions, we found the data pooling method of looking at all the possible historical years per person (type 4) performs the best.

With Fixed Window Data Model, we were able to classify the binary classification (crime occurs or not) with 99% accuracy. However, the F1 scores using both the data models were around 0.55 as highest. Due to the unbalanced nature of the data we proposed a weighted cross entropy loss setup, this helped us to bias the network towards the minority classes (Level 1,2,3) and boosted learning for better classification. Through experimentation we also iteratively found the optimal network structure and parameters for better learning as listed in results.

As noted in the result section, we experienced mis-classifications with our network. To overcome this, we could start with a biased network toward the less represented classes, by changing the weight parameters. We used an iterative method to find the optimal weight, but in the future, an automated method can be developed to choose the optimal weight parameters which yields best F1-score, by parameterizing the cost function.

A likely cause of the problem with mis-classifications may stem from an unbalanced class distribution; A few different improvements could be done using the pooled data. An RNN network can be developed where each crime record at time t is input for predicting a crime at time t+1. The network training may help the class imbalance issues. In addition, we can employ a combination of under and oversampling with SMOTE with this RNN method to balance the training dataset.

We would recommend further investigations into the use of different types of neural networks which takes the time dependent nature of data in to account for the crime project, as we believe that it has great potentials in many aspects.

For the future development our recommendations include.

- Improving the training data by several means: using better preprocessing to get the person level timelines to an even scale to fit the RNN model, simulate data by over and under sampling for better balance.
- Customizing the loss function further to automate the optimal weight parameter picking.

APPENDICES

Appendix A – (Python Code)

```python
''''Python code to implement Crime Model'''
import pandas as pd
import tensorflow as tf
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
# from sklearn.cross_validation import train_test_split
from sklearn.metrics import f1_score
#import visualizations2 as vis
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
le = preprocessing.LabelEncoder()
import collections
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from imblearn.over_sampling import SMOTE

from sklearn.metrics import precision_recall_fscore_support
F1=precision_recall_fscore_support
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
import matplotlib.pyplot as plt




NUM_ITERS=40000
DISPLAY_STEP=1
BATCH=2000
beta=0.005
Mweight=tf.constant(0.8,dtype=tf.float32)
learning_rate = 0.005
window=3

class CrimeProject(object):

    def __init__(self):
        # pool the data
        print('Loading data')
        csv_file = 'Charges_df.csv'
        self.data = pd.read_csv(csv_file,index_col=0)
        self.data=self.data.dropna(how='any')

        print('converting date time')
        # convert datetime
        self.data['booking_date'] =  pd.to_datetime(self.data['booking_date'])
        self.data['year'] = self.data['booking_date'].dt.year
        self.data=(self.data.sort_values('booking_date')).reset_index(drop=True)

        # generate dummy data
        print('generating dummy data')
        dummy_cri=pd.get_dummies(self.data['NCIC_Category_Code'])
        dummy_level=pd.get_dummies(self.data['NCIC_level'])
        dummy_df=pd.concat([dummy_cri, dummy_level], axis=1)

        # merge and drop columns
        print('Merging and droping the columns')
        self.data=pd.concat([self.data, dummy_df], axis=1)
        self.data=self.data.drop(['NCIC_Category_Code','no_bookings','NCIC_Crime_Cod
e','NCIC_Category','booking_date'],axis=1)

        # split to test and train
        print('Splitting test and train')
        value_counts = self.data.PersonID.value_counts(dropna=True, sort=True)
        ids = value_counts.rename_axis('ID').reset_index(name='counts')


        # get the Id's for criminals
        cri1=ids.loc[ids['counts'] <= 2]
        cri2=ids.loc[ids['counts'] > 2]
        train1=cri1.sample(frac=0.8,random_state=200)
        test1=cri1.drop(train1.index)

        train2=cri2.sample(frac=0.8,random_state=200)
        test2=cri2.drop(train2.index)

        train=pd.concat([train1,train2 ])
        test=pd.concat([test1,test2 ])
        train=train['ID'].unique()
        test=test['ID'].unique()

        f_test=self.preprocess(self.data,test)
        f_train=self.preprocess(self.data,train)

        print('generating CSV')
        f_test.to_csv("f_test_5year.csv", index=False)
        f_train.to_csv("f_train_5year.csv", index=False)

        f_test = pd.read_csv("f_test_5year.csv")
        f_train = pd.read_csv("f_train_5year.csv")

        lable_col="next_level"

        X_train,y_train=self.train_test(f_train,lable_col,window)
        X_test,y_test=self.train_test(f_test,lable_col,window,test=True)

        train_lab=self.lab(y_train)
        test_lab=self.lab(y_test)

                x_train=X_train.values
                x_test=X_test.values
                self.createModel(x_train,y_train,x_test,y_test,train_lab,test_lab)

            def loadData(self):
                pass

            def train_test(self,df1,lable_col,window,test=False):
                if test:
                    df=df1.copy()
                    dataR=df.pop('person_race')
                    dataG=df.pop('Gender')
                    dummy_R = pd.get_dummies(dataR)
                    dummy_G = pd.get_dummies(dataG)
                    df=pd.concat([df, dummy_R],axis=1)
                    df=pd.concat([df, dummy_G],axis=1)
                    y=df[lable_col]
                    x=df.drop(["PersonID","NCIC_level","next_level",'year'],axis=1)
                    x=x.reset_index(drop=True)
                    y=y.reset_index(drop=True)
                else:

                    df=df1.copy()

                    dataR=df.pop('person_race')
                    dataG=df.pop('Gender')
                    dummy_R = pd.get_dummies(dataR)
                    dummy_G = pd.get_dummies(dataG)
                    df=pd.concat([df, dummy_R],axis=1)
                    df=pd.concat([df, dummy_G],axis=1)
                    df=df.reset_index(drop=True)


                    number_records_fraud = len(df[df.next_level >=1])
                    fraud_indices = np.array(df[df.next_level >=1].index)

                    split= number_records_fraud*1


                    # Picking the indices of the normal classes
                    normal_indices = df[df.next_level ==0].index

                    # Out of the indices we picked, randomly select "x" number (number_records_fraud)
                    random_normal_indices = np.random.choice(normal_indices, split, replace = False)
                    random_normal_indices = np.array(random_normal_indices)

                    # Appending the 2 indices
                    under_sample_indices = np.concatenate([fraud_indices,random_normal_indices])

                    # Under sample dataset
                    under_sample_data = df.iloc[under_sample_indices,:]


                    y = under_sample_data[lable_col]
                    x=under_sample_data.drop(["PersonID","NCIC_level","next_level",'year'],axis=1)

                    x=x.reset_index(drop=True)
                    y=y.reset_index(drop=True)

                return x,y

            def plot_confusion_matrix(self,cls_pred,cls_true,normalize=False):

                cm = confusion_matrix(y_true=cls_true,
                                      y_pred=cls_pred)
                print(cm)
```

```

        if normalize:
            cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        else:
            print('Confusion matrix, without normalization')
            print(cm)
        plt.matshow(cm)
        plt.colorbar()
        tick_marks = np.arange(4)
        plt.xticks(tick_marks, range(4))
        plt.yticks(tick_marks, range(4))
        plt.xlabel('Predicted')
        plt.ylabel('True')

        plt.show()


    def smo(self,X_train, y_train):
        sm = SMOTE(ratio={0.0: 31017, 1.0: 6000, 2.0: 6000, 3.0: 6000},random_state=12,k_neighbors=2,m_neighbors=4)
        X_train, y_train = sm.fit_sample(X_train, y_train)
        return X_train, y_train

    def one_hot_encode(self,np_array,i):
        return (np.arange(i) == np_array[:,None]).astype(np.float32)

    def preprocess(self,data,idl):
        x=idl
        df33 = pd.DataFrame()
        for n in x:
            y=data.loc[data['PersonID'] == n]

            uniq=y['year'].unique()
            mrk=[2013,2014,2015,2016,2017]
            if len(uniq)==1 and any(x in mrk for x in uniq):
                df=y
                nb=len(df)
                ex=(df.loc[:, ['PersonID','NCIC_level','Gender','Age','person_race','year']]).tail(1)

                df1=df.copy()
                df1=df1.drop(columns=['NCIC_level','Gender','Age','person_race','year'])
                XF=df1.groupby(["PersonID"],as_index=False).sum()
                XF.drop(columns=['PersonID'],inplace=True)
                ex.reset_index(drop=True, inplace=True)
                XF.reset_index(drop=True, inplace=True)
                doo=pd.concat([ex, XF], axis=1)
                doo['no_bookings']=1
                doo['next_level']=0.0
                df33=df33.append(doo,ignore_index=True)

            else:
                mx=y['year'].max()
                mi=y['year'].min()
                age=y['Age'].min()

                for p in range(mi,2013):

                    df = y[(y['year'] <= p)]
                    nb=len(df)
                    ex=(df.loc[:, ['PersonID','NCIC_level','Gender','Age','person_race','year']]).tail(1)
                    df1=df.copy()
                    df1=df1.drop(columns=['NCIC_level','Gender','Age','person_race','year'])
                    XF=df1.groupby(["PersonID"],as_index=False).sum()
                    XF.drop(columns=['PersonID'],inplace=True)
                    ex.reset_index(drop=True, inplace=True)
                    XF.reset_index(drop=True, inplace=True)
                    doo=pd.concat([ex, XF], axis=1)
                    jj=[p+1,p+2,p+3,p+4,p+5]
                    boo = any(x in jj for x in y['year'].unique())

                    if boo==True:
                        bb=y[(y['year'] >= p+1)&(y['year'] <= p+5)]
                        label=bb['NCIC_level'].min()
                        doo['no_bookings']=nb
                        doo.at[0, 'Age'] =  age
                        doo['next_level']=label
                        doo['year']=p
                        df33=df33.append(doo,ignore_index=True)
                        age+=1
                    else:

                        label=0.0
                        doo['no_bookings']=nb
                        doo.at[0, 'Age'] =  age
                        doo['next_level']=label
                        doo['year']=p
                        df33=df33.append(doo,ignore_index=True)
                        age+=1


        return df33

    def lab(self,y):
        z=y.copy()
        z[z > 0] = 1
        y=self.one_hot_encode(y,4)
        print(z.shape)
        print(y.shape)
        a=y[:, [1,2,3]]
        b=np.column_stack((z,a))
        return b


    def customLoss(self,true,pred):

        Btru = tf.slice(true, [0, 0], [-1, 1])
        print(Btru,'Shape of Btrue')
        Bpred = tf.slice(pred, [0, 0], [-1, 1])

        Mtru = tf.slice(true, [0, 1], [-1, 3])
        print(Mtru,'Shape of Mtru')
        Mpred = tf.slice(pred, [0, 1], [-1, 3])
        print(Mpred,'Shape of Mpred')
```

```python

        cross_entropyB=tf.nn.sigmoid_cross_entropy_with_logits(logits=Bpred, labels=Btru)+beta*tf.nn.l2_loss(W1) +beta*tf.nn.l2_loss(b1) +beta*tf.nn.l2_loss(W2) +beta*tf.nn.l2_loss(b2) +beta*tf.nn.l2_loss(W3) +beta*tf.nn.l2_loss(b3) +beta*tf.nn.l2_loss(W4) +beta*tf.nn.l2_loss(b4) +beta*tf.nn.l2_loss(W5) +beta*tf.nn.l2_loss(b5)

        cross_entropyM = tf.nn.softmax_cross_entropy_with_logits_v2(logits=Mpred, labels=Mtru)+beta*tf.nn.l2_loss(W1) +beta*tf.nn.l2_loss(b1) +beta*tf.nn.l2_loss(W2) +beta*tf.nn.l2_loss(b2) +beta*tf.nn.l2_loss(W3) +beta*tf.nn.l2_loss(b3) +beta*tf.nn.l2_loss(W4) +beta*tf.nn.l2_loss(b4) +beta*tf.nn.l2_loss(W5) +beta*tf.nn.l2_loss(b5)
        print(cross_entropyM,'Shape of CE')

        TBtru=tf.reshape(Btru,[-1,])
        print(TBtru,'Shape of TBtrue')
        cross_entropyM=tf.multiply(TBtru,cross_entropyM)
        print(cross_entropyM,'Shape of CE After')
        loss= tf.multiply((1.0-Mweight),tf.reduce_mean(cross_entropyB))+tf.multiply(Mweight,tf.reduce_mean(cross_entropyM))

        return loss


    def customaccu(self,true,pred):

        Btru = tf.slice(true, [0, 0], [-1, 1])
        Bpred = tf.slice(pred, [0, 0], [-1, 1])

        Mtru = tf.slice(true, [0, 1], [-1, 3])
        Mpred = tf.slice(pred, [0, 1], [-1, 3])


        predicted = tf.nn.sigmoid(Bpred)
        print(predicted)
        Bipred = tf.round(predicted)
        print(Bipred)
        correct_pred = tf.equal(Bipred, Btru)
        print(correct_pred)
        accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
        print(accuracy)
        print(Mpred, 'shape of mpred')
        mulpred = tf.cast(tf.argmax(Mpred,1), tf.float32)
        l=tf.constant(1.0,dtype=tf.float32)
        mulpred =mulpred+l
        TBpred=tf.reshape(Bipred,[-1,])
        mulpred = tf.multiply(TBpred,mulpred)


        print(mulpred, 'shape of mulpred')
        Mtrue=tf.cast(tf.argmax(Mtru,1), tf.float32)
        Mtrue=Mtrue+l
        TBtru=tf.reshape(Btru,[-1,])
        Mtrue = tf.multiply(TBtru,Mtrue)
        correct_prediction = tf.equal(mulpred, Mtrue)
        print(correct_prediction)

        accuracy1 = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

```


                return accuracy,accuracy1,Bipred,mulpred


        def logmax(self,pred):

                Bpred = tf.slice(pred, [0, 0], [-1, 1])

                Mpred = tf.slice(pred, [0, 1], [-1, 3])


                predicted = tf.nn.sigmoid(Bpred)
                Bipred = tf.round(predicted)

                mulpred = tf.cast(tf.argmax(tf.nn.softmax(Mpred),1), tf.float32)
                l=tf.constant(1.0,dtype=tf.float32)
                mulpredi=mulpred+l

                return Bipred,mulpredi

        def createModel(self,x_train,y_train,x_test,y_test,train_lab,test_lab):


                tf.set_random_seed(0)

                X = tf.placeholder(tf.float32, [None, 49])
                # correct answers will go here
                Y_ = tf.placeholder(tf.float32, [None, 4])


                # layers sizes
                L1 = 128
                L2 = 64
                L3 = 64
                L4 = 64
                L5 = 4
                # L6 = 16
                # L7 = 2
                # L8 = 16
                # L9 = 8
                # L10 = 2



                # weights - initialized with random values from normal distribution m
ean=0, stddev=0.1
                # output of one layer is input for the next
                W1 = tf.Variable(tf.truncated_normal([49, L1], stddev=0.1))
                b1 = tf.Variable(tf.zeros([L1]))

                W2 = tf.Variable(tf.truncated_normal([L1, L2], stddev=0.1))
                b2 = tf.Variable(tf.zeros([L2]))

                W3 = tf.Variable(tf.truncated_normal([L2, L3], stddev=0.1))
                b3 = tf.Variable(tf.zeros([L3]))

                W4 = tf.Variable(tf.truncated_normal([L3, L4], stddev=0.1))
                b4 = tf.Variable(tf.zeros([L4]))

                W5 = tf.Variable(tf.truncated_normal([L4, L5], stddev=0.1))
                b5 = tf.Variable(tf.zeros([L5]))






                # -1 in the shape definition means compute automatically the size of this dimension
                XX = tf.reshape(X, [-1, 49])

                # Define model
                Y1 = tf.nn.relu(tf.matmul(XX, W1) + b1)
                Y2 = tf.nn.relu(tf.matmul(Y1, W2) + b2)
                Y3 = tf.nn.relu(tf.matmul(Y2, W3) + b3)
                Y4 = tf.nn.relu(tf.matmul(Y3, W4) + b4)


                Ylogits = tf.matmul(Y4, W5) + b5



                cross_entropyx = self.customLoss(Y_,Ylogits)


                accuracy1,_,_,_=self.customaccu(Y_,Ylogits)
                _,accuracy2,_,_=self.customaccu(Y_,Ylogits)

                _,Multi=self.logmax(Ylogits)

                Binary,_=self.logmax(Ylogits)


                train_step = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropyx)



                # Initializing the variables
                init = tf.global_variables_initializer()
                init_l = tf.local_variables_initializer()





                train_losses = list()
                train_Binary_acc = list()
                train_Multi_acc = list()
                test_losses = list()
                test_Binary_acc = list()
                test_Multi_acc = list()
                train_F1 = list()
                test_F1 = list()


                # saver = tf.train.Saver()



                sess = tf.Session()
                # Launch the graph
                with tf.Session() as sess:
                    sess.run(init)
                    sess.run(init_l)



                    for i in tqdm(range(1,NUM_ITERS+2)):
                        idx = np.random.choice(len(x_train), BATCH, replace=True)
                        X_batch = x_train[idx, :]

                        y_batch = train_lab[idx]


                        if i%DISPLAY_STEP ==0:

                            n=i/DISPLAY_STEP
                            print(n)

                            acc_Binary_trn,acc_mul_trn, loss_trn = sess.run([accuracy1,accuracy2, cross_entropyx], feed_dict={X: X_batch, Y_: y_batch})

                            acc_Binary_tst,acc_mul_tst, loss_tst = sess.run([accuracy1,accuracy2, cross_entropyx], feed_dict={X: x_test, Y_: test_lab})


                            print("#{} Trn Binary acc={}  Trn Multi acc={}  Trn loss={} ,Tst Binary acc={} Tst Multi acc={}  Tst loss={}".format(i,acc_Binary_trn,acc_mul_trn,loss_trn,acc_Binary_tst,acc_mul_tst,loss_tst))

                            train_losses.append(loss_trn)
                            train_Binary_acc.append(acc_Binary_trn)
                            train_Multi_acc.append(acc_mul_trn)

                            test_losses.append(loss_tst)
                            test_Binary_acc.append(acc_Binary_tst)
                            test_Multi_acc.append(acc_mul_tst)


                        sess.run(train_step, feed_dict={X: X_batch, Y_: y_batch})
                        print(learning_rate)


                    pred_train_Binary = Binary.eval(feed_dict={X: x_train}, session=sess)
                    pred_train_Binary =np.ravel(pred_train_Binary[ : , 0])
                    pred_train_Multi = Multi.eval(feed_dict={X: x_train}, session=sess)
                    print(pred_train_Binary.shape)
                    print(pred_train_Multi.shape)

                    pred_test_Binary = Binary.eval(feed_dict={X: x_test}, session=sess)
                    pred_test_Binary=np.ravel(pred_test_Binary[ : , 0])
                    pred_test_Multi = Multi.eval(feed_dict={X: x_test}, session=sess)
```

```

            print ("accuracy_Binary_train", sess.run(accuracy1, feed_dict={X: x_train, Y_: train_lab}))
            print ("accuracy_Multi_train", sess.run(accuracy2, feed_dict={X: x_train, Y_: train_lab}))

            truelab=y_train

            Predlab=np.multiply(np.array(pred_train_Binary),np.array(pred_train_Multi))

            f1_train=f1_score(truelab,Predlab, average='macro')
            print ("F1-Score_train", f1_train)
            plot_confusion_matrix(Predlab,truelab,normalize=True)


            print ("accuracy_Binary_test", sess.run(accuracy1, feed_dict={X: x_test, Y_: test_lab}))
            print ("accuracy_Multi_test", sess.run(accuracy2, feed_dict={X: x_test, Y_: test_lab}))

            truelab1=y_test

            Predlab1=np.multiply(np.array(pred_test_Binary),np.array(pred_test_Multi))

            f1_test=f1_score(truelab1,Predlab1,average='macro')
            print ("F1-Score", f1_test)
            self.plot_confusion_matrix(Predlab1,truelab1,normalize=True)



    def train(self):
        pass

    def predict(self):
        pass

if __name__=='__main__':
    c=CrimeProject()
```

<u>REFERENCES</u>

[1] Kang, H. and Kang, H. (2017). Prediction of crime occurrence from multi-modal data using deep learning. *PLOS ONE*, 12(4), p.e0176244.

[2] Iqbal R, Azmi Murad MA, Mustapha A, Hassany Shariat Panahy P, KhanahmadliraviN. An experimental study of classification algorithms for crime prediction. Indian Jour- nal of Science and Technology. 2013; 6(3):4219–25

[3] Bogomolov A, Lepri B, Staiano J, Oliver N, Pianesi F, Pentland A (2014) Once upon a crime: towards crime prediction from demographics and mobile data. In: Proceedings of the 16th international conference on multimodal interaction. ACM, New York, pp 427-434

[4] X. Wang, M. S. Gerber, and D. E. Brown. Auto- matic crime prediction using events extracted from twitter posts. In Social Computing, Behavioral- Cultural Modeling and Prediction, pages 231–238. Springer, 2012.

[5] TensorFlow main website, https://www.tensorflow.org/

[6] TensorFlow Layers guide, https://www.tensorflow.org/get_started/mnist/beginners

[7] TensorFlow Losses, https://www.tensorflow.org/ versions/r1.5/api_docs/python/tf/losses

[8] USA Public Policy website https://secure.ssa.gov/poms.nsf/lnx/0202613900