CRIME COUNT FORECASTING AND ARREST PREDICTION

A thesis submitted to

Veermata Jijabai Technological Institute

in fulfilment for the award of the degree of Bachelor of Technology in

Information Technology

by

Mahesh Gond, Sandesh Nagrale, Shrirang Pinjarkar, Vasu Mhashakhetri

Under the guidance of

Prof. Sowmiya Raksha Naik

Department of Computer Science and

Information Technology

DEPARTMENT OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

May 2019

Declaration

I declare that,

1. The work contained in this report has been done by me under the guidance of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
4. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

May 2019

Mahesh Gond, Sandesh Nagrale, Shrirang Pinjarkar,
Vasu Mhashakhetri

Abstract

Names:**Mahesh Gond, Sandesh Nagrale, Shrirang Pinjarkar, Vasu Mhashakhetri**

Roll No:**151080005, 151080006, 151080008, 151080010**

Degree for which submitted: **B. Tech**

Department: **Computer Science and Information Technology**

Thesis title: **CRIME COUNT FORECASTING AND ARREST PREDICTION**

Thesis supervisor: **Prof. Sowmiya Raksha Naik**

Month and year of thesis submission: **May 2019**

Crime is one of the most predominant and alarming aspects in our society and its prevention is a vital task as it is threatening public safety and disrupting the economy. To be better prepared to respond to criminal activity, it is important to understand patterns in crime. In our project, we analyze crime data from the city of Chicago, scraped from publicly available website of Chicago Police. This research aims at developing a model to predict future crime count using spatio-temporal data. The Proposed model uses the recurrent neural network for forecasting crime count, and also uses random forest to predict arrests. Long short-term memory (LSTM) networks are variants of RNNs which are designed to cope with the gradient vanishing problem. They are well-suited to learn from experience to predict time series when there are time lags of unknown size and bound between important events. We use the historical data to build a deep recurrent predictive model (STNN) that can detect the spatio-temporal patterns and predict the crime counts in the near future by using deep recurrent neural network architecture.

Acknowledgements

We would like to express or deep gratitude to Prof. Sowmiya Raksha Naik for her patient guidance, enthusiastic encouragement and useful critiques of this research work. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment. Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment. Finally, we wish to thank our parents for their support and encouragement throughout this study.

The Authors.

Contents

List of Figures

List of Tables

Abbreviations

FEA	**F**inite **E**lement **A**nalysis
FEM	**F**inite **E**lement **M**ethod
LVDT	**L**inear **V**ariable **D**ifferential **T**ransformer
RC	**R**einforced **C**oncrete

Symbols

D^{el}	elasticity tensor
σ	stress tensor
ε	strain tensor

Chapter 1

Introduction

1.1 Aim

Crime forecasting is the latest technology that can be used to forecast future crimes, most vulnerable locations and decide prevention efforts. Our aim is to precisely forecast high risk crime regions in the city of Chicago using spatio-temporal neural network methods.

1.2 Motivation

Crimes are a common social problem affecting the culture and the economic growth of society. Because of increasing rate of crimes, law enforcement agencies are continuing to demand advanced GIS and new data mining approaches to improve crime analytics and for better development of their communities. Crime Forecasting is one of the most helpful strategies for the police which enables to identify high crime risk place and police department would be able to utilize their resources efficiently to reduce and prevent criminal activity. Human behavior has a huge scope for unpredictability but there's a certain structure that can be investigated even in the most chaotic scenarios. Using repetitive statistics, data collected from hot spots of crime can help analyze the intricacies of crime prevention. The crime rates can be significantly reduced by the real-time crime

forecasting which are helpful in saving lives that is the most valuable thing. Proper analysis of previous crime data helps in predicting the crimes and thus supports in reducing the crime rate. The analysis process includes the study of crime reports and identifying the emerging patterns, series, and trends as quickly as possible. This analysis helps in preparing statistics, queries, and maps on demand. It also helps to see if a crime fits in a certain known pattern or a new pattern is necessary. Each city or area have certain level of crime risk which depends on various factors like income level of residents, poverty ratio, illiterate ratio, crowdy place, secluded area, etc. These factors help in identifying hotspot and determine the model which will be used to predict future criminal activity. But risk at a location changes over time. So, we need to take care of selecting model which will be adaptive to those changing environments. Due to more advance technology, a significant amount of data with spatial and temporal information has been collected. The main challenge is to develop a method to handle this large dataset and develop a model that can accurately predict future hot spots. There are many regressions models the can be used to develop model to predict hotspot. Random Forest, Naïve Bayes, 2-Layer Neural Network can be used to forecast crime. But these algorithms are static dependent on data. Because after some time, there is possibility that certain area become less active crime area. So the model should remember the changes that occurs during time and use the present criminal activity to create hotspot. This can be done by using recurrent neural network. Recurrent neural networks (RNN), which have been successfully employed for word embedding and formal languages generation, are capable of capturing long-distance dependencies. However, due to the gradient vanishing problem, the gradient (error signal) decreases exponentially while the front layers train very slowly, the effect is not ideal. Long short-term memory (LSTM) networks are variants of RNNs which are designed to cope with the gradient vanishing problem. They are well-suited to learn from experience to predict time series when there are time lags of unknown size and bound between important events. However, for a space-time series, which is a sequence taken at a successive equally spaced planes in time and each plane represents objects defined in a geometric space.

Chapter 2

Literature Review

2.1 Crime Hot Spot Forecasting: A Recurrent Model with Spatial and Temporal Information [13]

Model propose is the Spatio-Temporal neural network (STNN) to precisely forecast crime hot spots with embedding spatial information. They evaluate the model using call-for-service data provided by the Portland, Oregon Police Bureau (PPB) for a 5-year period from March 2012 through the end of December 2016. They have shown that STNN model outperforms a number of classical machine learning approaches and some alternative neural network architectures.

2.1.1 Methodology

Analysis of the data-set released by the NIJ for Portland found that the ranking of the hottest cells was very resistant to change. The two horizontal axes show the time index and cell index (one row for each unique cell), so if one was to look down at Figure 3 from above, each point would be a single cell-interval. The vertical axis shows a tiered measure of crime activity and reports whether or not that cell-interval was hot (0) or not (1) by the actual number of crimes in the given threshold. The cells were indexed by their ranking in the first interval (the cell with the highest number of crimes was given index 0, the next

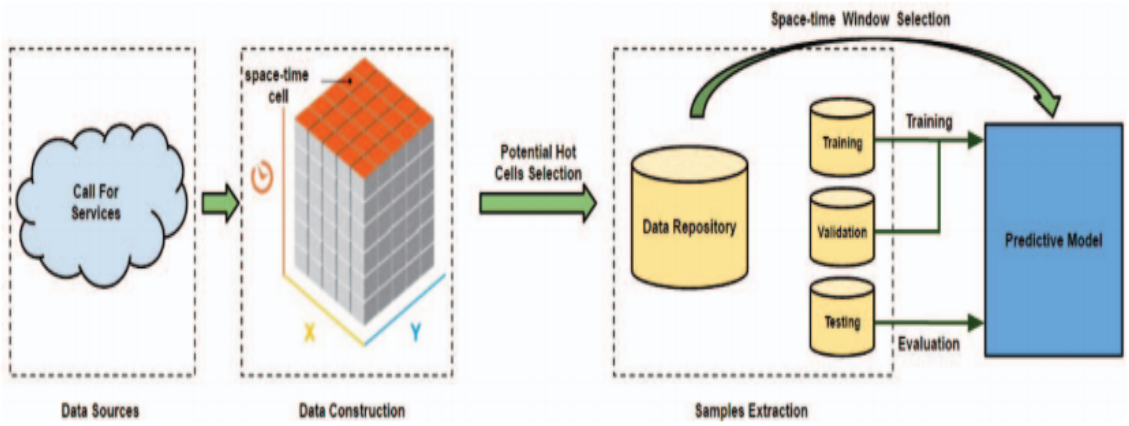


FIGURE 2.1: The workflow of the study

1, and so on), which is why we see that the predicted hot spots are clustered in those cells with low indices - the cells that were the hottest in the first interval are among the hottest for the entire 5-year period. Cells that were outside the highest 400 cells by number of crimes were almost never hot spots. Because of this, we focus the training of our model on those areas that are hot spots or border on hot spots. To build a precise space-time series analysis model, both temporal and spatial influence should be considered. In each cell, both its own crime count and that of the cells around it are essential factors for predicting the crime level of the cell, so it is necessary to include all of that information in our model. Therefore the historical spacial influence can be considered as the accumulation of the short-term influences of recent, nearby events. Since influence of a certain event will decrease with the extension of time and space, the historical spacial influence at a cell s can be limited to only those events that have influence that can "reach" that cell. This model uses a traditional looping RNN structure, where the network contains a hidden layer that uses the output of the previous time step as an additional input to the analysis of the current time-step. Layers use a tanh activation neuron as the nonlinearity to produce output.

2.2 Feature Engineering for Crime Hotspot Detection [5]

Analyzing characteristics of the urban environment in these two cities, deploying a machine learning model to detect categories and hotspots of criminal activities. Proposed

an extensive set of spatio-temporal & urban features which can significantly improve the accuracy of machine learning models for these tasks.

2.2.1 Methodology

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to machine learning predictive models, resulting in improved model accuracy on unseen data. We propose a set of Urban Features (UF) which are correlated with criminal activity, increasing the detection performance of machine learning algorithms for hotspot of crimes, extending current work in this area. An Urban Feature is an individual measurable property of the built (or static) environment of an urban area. It expresses one characteristic of a neighborhood like the street network, land usage or type of buildings. Urban Features describe different aspects of the urban space and can be numeric or binary in nature. Binary features describe for example the presence or absence of an attribute. In contrast to the Urban Features which are static and do not change over time, we also propose a series of Spatio-Temporal Features (STF) which are rather dynamic. Their measurements change over time in dependence of the criminal activities.It first discretize the urban space into cells using the kmeans clustering algorithm.After that a Random Forest Classifier learns from one set of cells and then predicts on the rest of the cells.

2.3 CRIMECAST: A Crime Prediction and Strategy Direction Service [1]

The aim of this paper is to introduce CRIMECAST which is a crime prediction and strategy direction service which attempts to predict probable future crimes by simulating probabilistic model implementation and Artificial Neural Network.

2.3.1 Methodology

Calculating Probability of Crime Occurrence

Probability Factor: Then we calculate a probability factor. Expressed as- Probability of Occurrence of Crime i in Location s: We can determine this by following equation - Probability of Occurrence,

$$O.P(i,s) = O(i,s)/\sum O(i,s) \tag{2.1}$$

Where, m = Number of domains in the given dataset In percentage, Probability of Occurrence = O.P(i, s) * 100%

Comparison between mathematical implementation and Artificial Neural Network implementation shows that ANN implementation gives us more precise prediction than mathematical implementation.

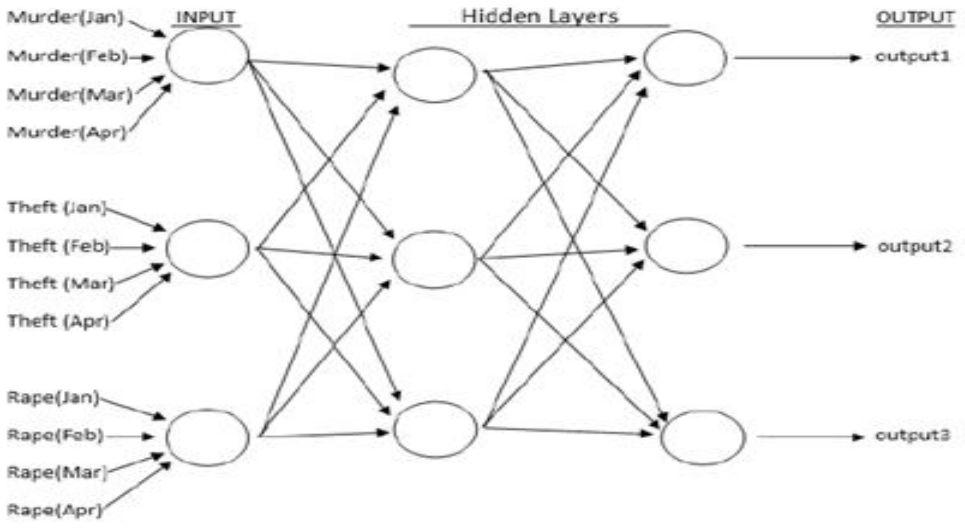


FIGURE 2.2: CRIMECAST ANN

2.4 Complementing GIS with Cluster Analysis in Assessing Property Crime in Katsina State, Nigeria [12]

One of the fundamental techniques to combat criminal activities is the better understanding of the dynamics of crime. Techniques are needed to categorise geographical areas according to their similarities in criminal activities that would facilitate the understanding of why and where crimes take place.

2.4.1 Methodology

ArcView GIS 3.2a, software was utilised for spatial analysis so as to map out the category of crimes enumerated earlier. Cluster Analysis :- Given a data matrix containing multivariate measurements on a large number of individuals (or objects), the objective is to build subgroups for which the observations or objects within each cluster are similar, but the clusters are dissimilar to each other according to some appropriate criterion. Cluster analysis can be divided into two fundamental steps:

1. Choice of a proximity measure:

1. Choice of a proximity measure:
2. Choice Of group

Starting point of cluster is given by X(p*n) with n measurements and p variables. Proximity matrix can be given by [h] The Euclidian distance dij cases, i and j with variable

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}$$

value xi (xi1,xi2,xi3.....) and xj(xj1,xj2,xj3...) is given by

$$\sqrt{\sum_{k=1}^{p}(x_{ik}+x_{jk})^2} \tag{2.2}$$

2.5 Using Linear Regression to Forecast Future Trends in Crime of Bangladesh [8]

linear regression model is used to forecast future crime trends of Bangladesh. The real dataset of crime is collected from the website of Bangladesh police. Then the linear regression model is trained on this dataset. After training the model, crime forecasting is done for dacoit, robbery, murder,women & child repression, kidnapping, burglary, theft and others for different region of Bangladesh.

2.5.1 Methodology

The associated task for the dataset used in this paper is regression. Therefore, liner regression model is used to forecast the status of crime. The linear regression model is simple and provides enough description of how the input affects the output. It predicts a variable Y (target variable) as a linear function of another variable X (input variable/features), given m training examples of the form

$$(x1, y1), (x2, y2), \ldots, (xn, yn) \quad where: \quad xi \ \epsilon \ X \ and \ yi \ \epsilon \ Y \tag{2.3}$$

2.6 Geospatial-Temporal Analysis and Classification of Criminal Data in Manila [7]

This study focused on the geospatial-temporal analysis and implementation of classification algorithms to predict criminal activities in the respective districts of Manila. Specifically, it aims to achieve the following:

1. To identify spatio-temporal hot spots using an actual dataset of crimes

2. To implement different classification algorithms to predict possible types of crimes

2.6.1 Methodology

Data preprocessing is imperative in data mining. This process cleans and transforms the raw crime data gathered from MPD into its appropriate representation for crime analysis. The crime incidents were graphically presented using heat map and hot spots which indicate the density of these crimes in a particular location. Geospatial-temporal analysis involves mapping crime location and time while overlaying other demographic information to predict criminal activities. In order to perform geospatial analysis in the dataset, ArcGIS 10, was used in point mapping the coordinates of the location and to produce the spatio-temporal heat maps using Kernel Density Estimation (KDE) for points features. ArcGIS is a desktop based GIS tool which can be used for hotspot analysis. KDE was used to visualize the density of crimes across three temporal perspectives: time of the day, day of the week, and quarters of the year.
Classification models are used to predict categorical class labels. In this paper, five algorithms for classification were implemented to predict the possible crime locations along with other variables that contribute to the occurrence of crime.

2.7 Draw backs

[1][8][7][5]Main aims for all of these methods include clustering and analysis only. Clustering methods includes k means clustering , Db-scan clustering , hierarchical clustering methods. They are typical well in grouping the similar data and patterns but randomness in crime data set is high. And as we know that clustering algorithm computes distance between each of the points and finds similarity on the basis of distance only that may be Euclidean or Manhattan. Whenever one trains data set with model it will run from scratch and calculate distance between points and cluster according to similarity in distance. Now each time computing of distance for millions of points is not feasible from users perspective if he is using some low end devices and time complexity of each models is in seconds and

in minutes for some which will lead to inconvenience. ArcGIS is used in some papers. Apart from computational Drawbacks these methods are just forming clusters from data set, which means user will get just visualization of patterns for crime but not real time forecasting, when will new crime happen? What will be next crime count for the current location? What will be arrest prediction ? Will criminal get arrested? [12]Some methods are using Web based methods for only visualizing and provide interface where user enters conditions regarding location details, crime details and gets visualization graphs and maps only. Which is again not forecasting any detailed analysis for current locations. Paper no [3] Time series data is used to predict the results. They used DTW for tracking the patterns and create hot spots.

DTW:-

$$Given\ \ two\ \ time\ \ series\ \ , X = x1, x2, ..., xi, ...,\ \ xn\ \ and\ \ Y = y1, y2, ..., yj, ..., ym, \tag{2.4}$$

DTW aligns the two series so that their difference is minimized. But problem with this approach is weights of the give matrix are large so computation for each pattern. Useful Where the dimensions do not have equal weightages. But drawbacks include Don't give exact location based prediction Computation for each time will be higher. Paper [] uses count based method for forecasting crime trends in Bangladesh. It counts for each of the type of crime.Its main advantage is it will give exact probability for the current location. Drawbacks include they are using just time and count not location based features like ward, beat, and longitude latitude. Regression models accuracy also less. Crime type based features can also be included.

Chapter 3

Fundamental Concepts

3.1 Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

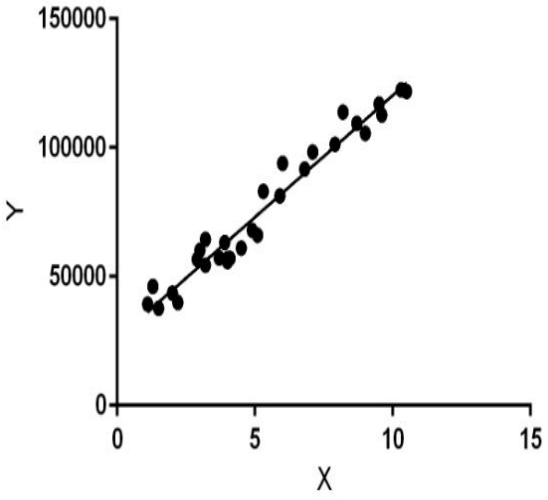


Figure 3.1: Linear Regression Plot

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.
Hypothesis function for Linear Regression :

$$Y_i = \theta_0 + \theta_1 X_i \tag{3.1}$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)
When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_0 and θ_1.
θ_0: coefficient of x
θ_1: coefficient of y
Once we find the best θ_0 and θ_1 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.
Cost Function (J):
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_0 and θ_1 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$J = \frac{1}{n} \sum_{i=1}^{n} \left(pred_i - y_i \right)^2 \tag{3.2}$$

3.2 Bernoulli naive Bayes

Bernoulli Naive Bayes is for binary features only. Similarly, multinomial naive Bayes treats features as event probabilities.In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model,

this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies. If is a boolean expressing the occurrence or absence of the ith term from the vocabulary, then the likelihood of a document given a class is given by

$$p(x|\, C_k) = \Pi_{i=1}^{n}\, p_{ki}^{xi}\big(1 - p_{ki}\big)^{(1-xi)} \tag{3.3}$$

3.3 Random Forest

Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random.The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.
One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:

random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier. Fortunately, you don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also deal with Regression tasks by using the Random Forest regressor. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

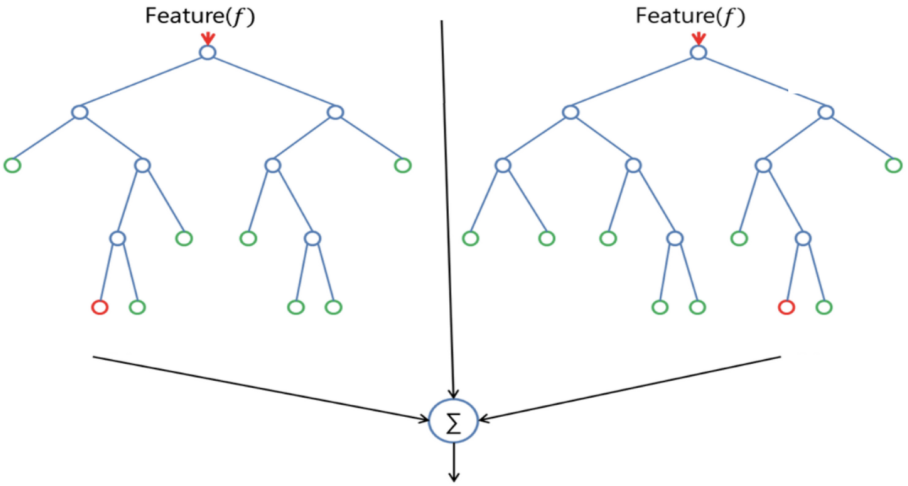


FIGURE 3.2: Random Forest

You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does). Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction.

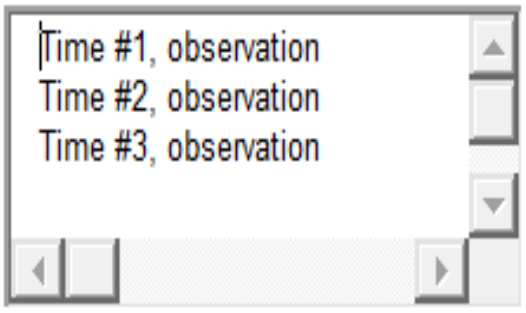
3.4 Time Series Forecasting

Time series forecasting is an important area of machine learning that is often neglected. It is important because there are so many prediction problems that involve a time component. These problems are neglected because it is this time component that makes time series problems more difficult to handle.

3.4.1 Time Series

A normal machine learning dataset is a collection of observations. ime does play a role in normal machine learning datasets. Predictions are made for new data when the actual

outcome may not be known until some future date. The future is being predicted, but all prior observations are almost always treated equally. Perhaps with some very minor temporal dynamics to overcome the idea of "concept drift" such as only using the last year of observations rather than all data available. A time series dataset is different. Time series adds an explicit order dependence between observations: a time dimension. This additional dimension is both a constraint and a structure that provides a source of additional information. For example:



3.4.2 Describing vs. Predicting

In descriptive modeling, or time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, relation to external factors, and the like. In contrast, time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series

3.4.3 Time Series Analysis

Time series analysis involves developing models that best capture or describe an observed time series in order to understand the underlying causes. This field of study seeks the "why" behind a time series dataset. This often involves making assumptions about the form of the data and decomposing the time series into constitution components. The quality of a descriptive model is determined by how well it describes all available data and the interpretation it provides to better inform the problem domain.

3.4.4 Time Series Prediction

Making predictions about the future is called extrapolation in the classical statistical handling of time series data. More modern fields focus on the topic and refer to it as time series forecasting. Forecasting involves taking models fit on historical data and using them to predict future observations. Descriptive models can borrow for the future (i.e. to smooth or remove noise), they only seek to best describe the data. An important distinction in forecasting is that the future is completely unavailable and must only be estimated from what has already happened. Types :

1. Univariate Time Series Forecasting
2. Multivariate Time series forecasting
3. Multistep Time series forecasting

3.5 PCA Principle Component Analysis

It is a technique to reduce the dimension of the feature space by feature extraction. For example, if we have 10 variables, in feature extraction, we create new independent variables by combining the old ten variables. By creating new variables it might seem as if more dimensions are introduced, but we select only a few variables from the newly created variables in the order of importance. Then the number of those selected variables is less than what we started with and that's how we reduce the dimensionality.There are multiple ways to look at PCA

PCA The Statistical Perspective

We have all these dataset features and values , but some of them are redundant and it makes it hard to visualize. However, we can't just drop some of them randomly. That's where dimensionality reduction plays a role to reduce the number of variables without losing too much information.Another problem is that it's hard to interpret those values, because they are based on different ranges and scales. Variables such as the test scores above measured on different scales or on a common scale with widely differing ranges are often standardized to refer to them with the same standard. **Standardization :-**

It includes most commonly used Z-score method to normalize the data. It uses mean, standard deviations like statistical metrics. The formula for Z-score is

$$Z = \frac{(X - \mu)}{\sigma} \tag{3.4}$$

Covariance/ correlation :-

In order to see the relation between all those variables, covariance is used. Covariance tells you about linear relationships between two variables: how two variables were closely related. Covariance formula is given as follows:

$$cov_{x,y} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \tag{3.5}$$

Again one have to follow following 2 steps to get to the final new Dimensions.

1. transformation of the data set : define new variables to replace the existing ones
2. selection of the data set : measure how well the new variables represent the original variables.

3.6 Long Short Term Memory

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way as shown above. In the above diagram, each line carries an entire vector, from the output of one node to the inputs

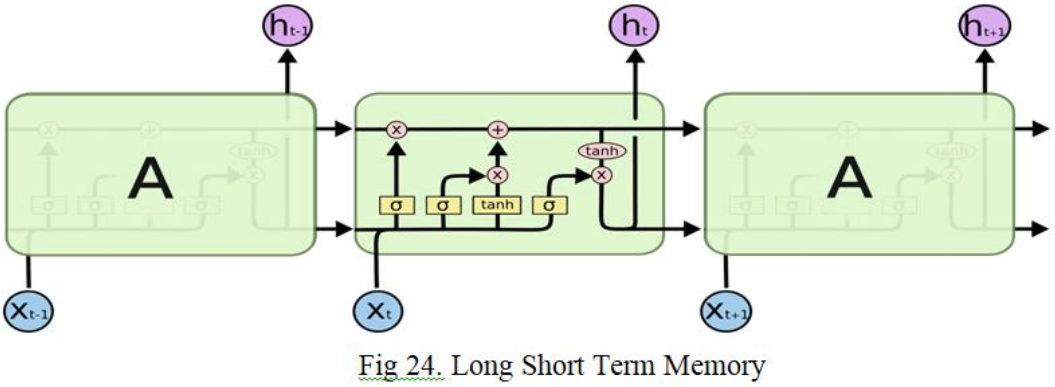


FIGURE 3.3: Long Short Term Memory

of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

3.7 ARIMA

A popular and widely used statistical method for time series forecasting is the ARIMA model. In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

An ARIMA model is a class of statistical models for analyzing and forecasting time series data.It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

- AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
- I: Integrated. The use of differencing of raw observations (.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used. The parameters of the ARIMA model are defined as follows:

- p: The number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations are differences, also called the degree of differencing.
- q: The size of the moving average window, also called the order of moving average.

When two out of the three terms are zeros, the model may be referred to based on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1). A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model. ARIMA models can be estimated following the Box–Jenkins approach.

Chapter 4

Proposed Model

4.1 Machine Architecture

Machine architecture for Crime prediction and analysis is given below. **Each steps sig-**

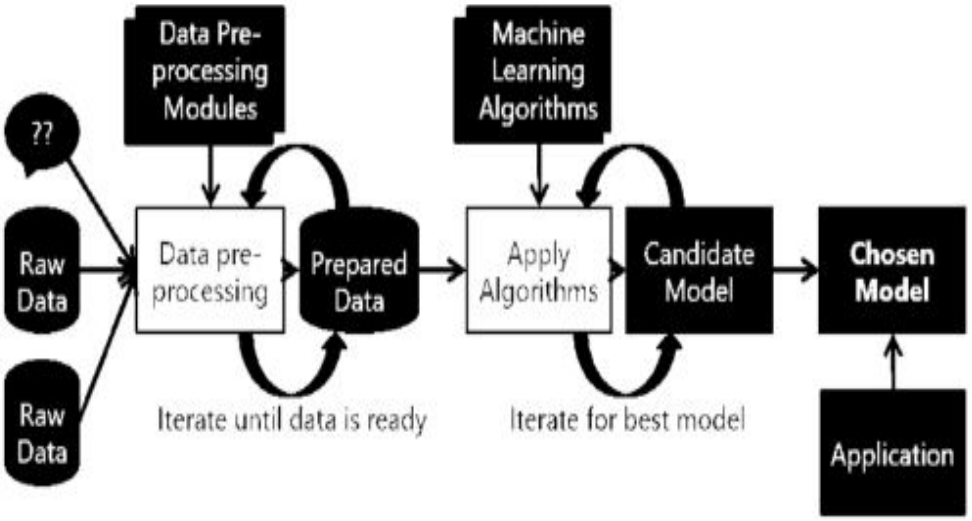


FIGURE 4.1: Machine Architecture

nificance and its working is given as follows:-

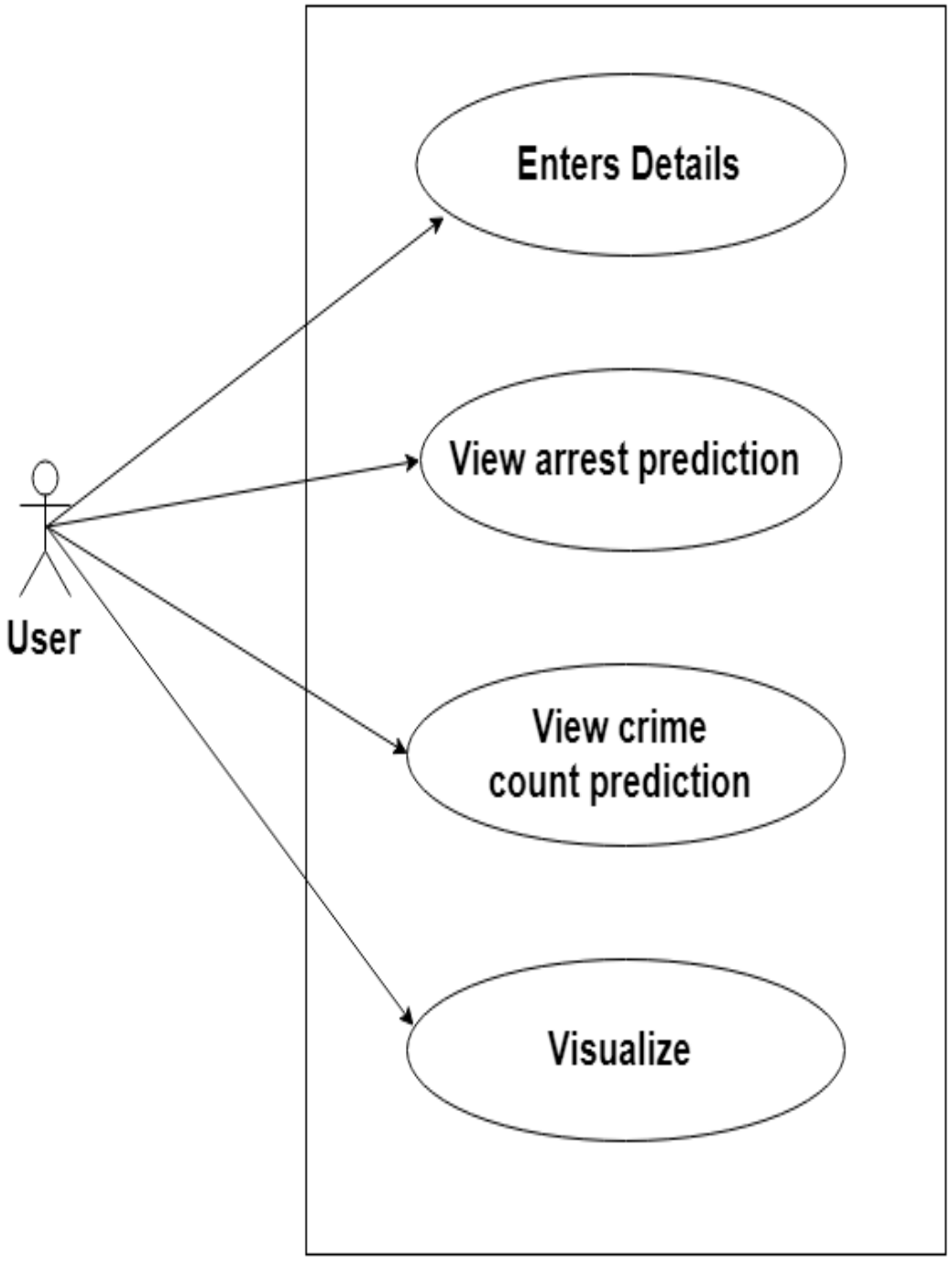


FIGURE 4.2: Use Case Diagram

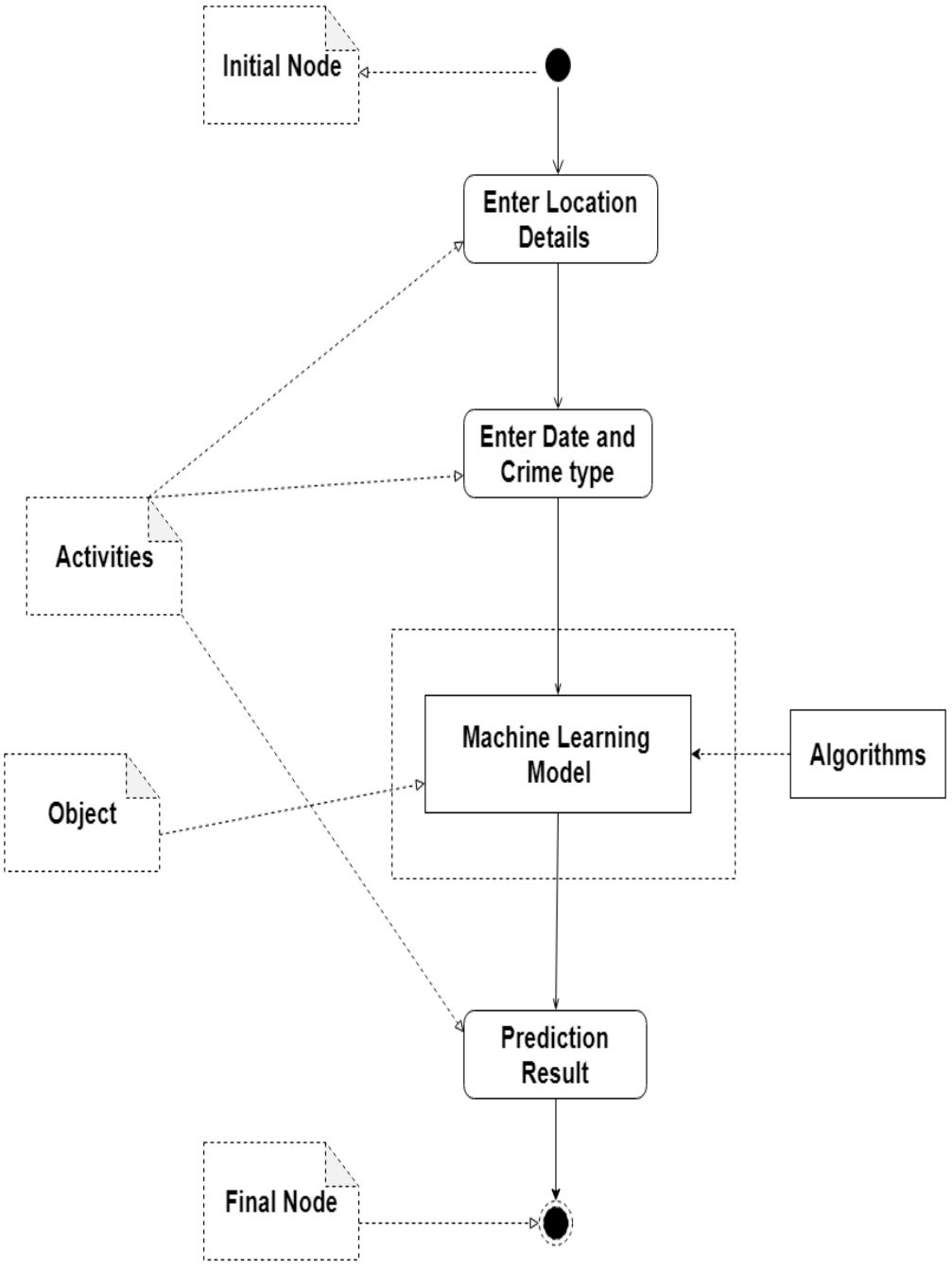


FIGURE 4.3: Activity Diagram

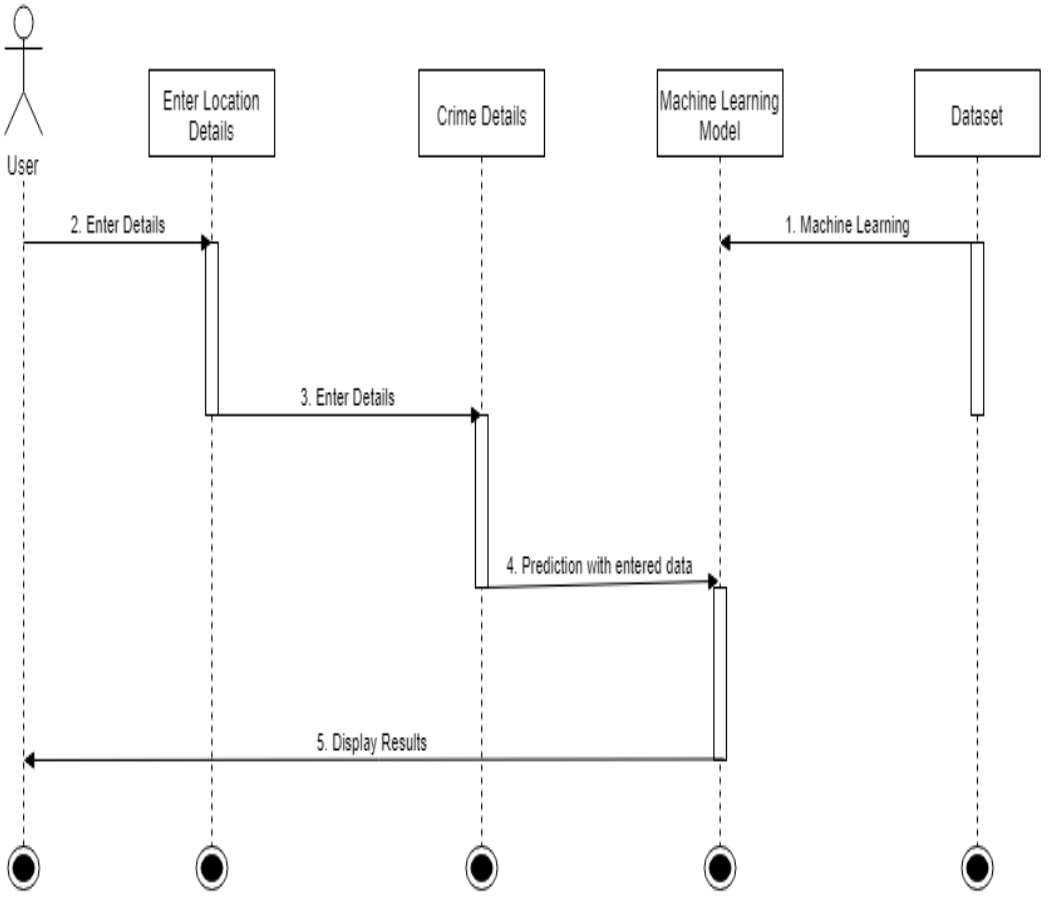


FIGURE 4.4: Sequence Diagram

4.2 Data Set

[h] The very first step is raw Data collection. At the first iteration we proposed for Indian cities but for India data is not available in longitude, latitude and time format. Data.gov.in provides data for crime but not according to the exact location and time rather it just gives number of crimes happened. Also other attributes like population, poverty ratio, and literacy ratio are not available accordingly. So we decided to perform this on Chicago data set which is available on Chicago Crime portal. The best thing about this data it gets updated in every week.

4.2.1 Features

There are 27 features in the data set.Each one with its description is as follows:-

1. ID :- Unique identifier for the record.
2. Date:-Date when the incident occurred. this is sometimes a best estimate.
3. Block :- The partially redacted address where the incident occurred, placing it on the same block as the actual address.
4. Primary :- Type The primary description of the IUCR code.
5. Description The secondary description of the IUCR code, a subcategory of the primary description.
6. Arrest :- Indicates whether an arrest was made.
7. Domestic :- Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.
8. Beat :- Indicates the beat where the incident occurred. A beat is the smallest police geographic area – each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts. See the beats at
9. District :- Indicates the police district where the incident occurred.
10. Ward :- The ward (City Council district) where the incident occurred.
11. Community Area :- Indicates the community area where the incident occurred. Chicago has 77 community areas. See the community areas
12. Latitude :- The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.
13. Longitude :- The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.

4.3 Preprocessing

4.3.1 Deleting rows

This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values.

4.3.2 Adding columns

Creating new column containing only date.

4.3.3 Extracting Time series Data

Creating a new dataset containing daily crime count in each beat from 01-01-2001 to 28-11-2018.

4.3.3.1 Time Series Data into Supervised Learning problem

Machine learning methods like deep learning can be used for time series forecasting. Before machine learning can be used, time series forecasting problems must be re-framed as supervised learning problems. From a sequence to pairs of input and output sequences.

4.3.3.2 The series_to_supervised() Function

We can use the shift() function in Pandas to automatically create new framings of time series problems given the desired length of input and output sequences. This would be a useful tool as it would allow us to explore different framings of a time series problem with machine learning algorithms to see which might result in better performing models. We define a new Python function named series_to_supervised() that takes a univariate or multivariate time series and frames it as a supervised learning dataset. The function takes four arguments:

- **data:** Sequence of observations as a list or 2D NumPy array. Required.
- **n_in:** Number of lag observations as input (X). Values may be between [1..len(data)] Optional. Defaults to 1.
- **n_out:** Number of observations as output (y). Values may be between [0..len(data)-1]. Optional. Defaults to 1.
- **dropnan:** Boolean whether or not to drop rows with NaN values. Optional. Defaults to True.

The function returns a single value:

- **return:** Pandas DataFrame of series framed for supervised learning.

4.3.3.3 Multi-Step or Sequence Forecasting

A different type of forecasting problem is using past observations to forecast a sequence of future observations. This may be called sequence forecasting or multi-step forecasting. We can frame a time series for sequence forecasting by specifying another argument. For example, we could frame a forecast problem with an input sequence of 10 past crime counts in a beat to forecast 2 future observations as follows:

```
data = series_to_supervised(values, 10, 2)
```

4.4 Apply Models

Our proposed system contains two predictions one for Arrest and one for crime count.proposed Machine learning models for each are as follows

4.4.1 Arrest Prediction

The pre-processed data set contains column which suggests weather arrest was done or not. Its data type is binary i.e. in true or false. We have converted the arrest column into

0 and 1 .Our proposed model for the Arrest prediction will predict weather the arrest will happen or not.
Data features as input selected for the module are:

1. Ward Number
2. Community area
3. Primary type
4. Beat number
5. Month
6. Day
7. hour
8. Location Description
9. Domestic

And arrest as output.
As the aim of the module is to predict the arrest, prediction based supervised models are used.

4.4.1.1 Linear regression

Linear regression takes the input data frame from pre-processed dataset. To implement the linear regression we have used linear model from sklearn library.
As it is one of the common problem in analysis of complex data comes from a large number of variables, which requires a large amount of memory and computation power. And as per the dataset it contains 60 million values and 9 input features so time and space complexities will be higher. PCA comes into picture to solve this problem of complexity. PCA(Principle component analysis) is the tool to select some of the features from the current data set on the basis of statistical matrices. After compressing dataset by using PCA we load our data set to the object formed by regression model and fit it with test

size of 20% and random state of 30. The outuput is stored in new array and compared with the values 0.5. If value is greater than 0.5 it will be tagged as true and if less then false. Then we predict the result and calculate the accuracy for it using confusion matrix. The accuracy achieved by LinearRegression is 80.7834%.

4.4.1.2 Bernoulli Naive Bayes

Accuracy for Linear regression is above 80% but to check if we can achieve some better we choose Bernoulli Naive Bayes theorem. Naive Bayes is preffered over others because it uses probability of arrest done and not done so indirectly it will be efficient to predict on the basis past probabilities. Bernoulli's naïve Bayes gives the ability to predict the probability for binary values.
Again input to the model is same as regression. Again Module is fit with sklearn object and accuracy is calculated using confusion matrix. Accuracy for Bernoulli again is 76.0478%

4.4.1.3 Random Forest

To achieve more accuracy, Bernoulli has less accuracy than Linear regression, we tried Random forest. We used Sklearn for the inbuilt libraries for Random Forest. We set obb (Out-of-bag) error for Random forest in list 50 to 100. n-estimators is the number of trees to be used in the forest and we set it to i which is iterator of for loop. Max-features on the other hand gives max features to be used at a time. Its has accuracy 87.7705%

4.4.2 Count Prediction

Different types of crimes can occur in different part of Chicago city and it is hard to know which areas will have more number of crimes and which will have less or not at all on a particular day. Beat indicates the beat where the incident occurred. A beat is the smallest police geographic area – each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts. By predicting the number of crimes in a beat the police can monitor the area efficiently and can patrol the area depending upon the

number of crimes that are going to happen in that area . For this we propose two models based on time series forecasting. One is ARIMA model and the other one is LSTM neural networks.

4.4.2.1 ARIMA

The statsmodels library provides the capability to fit an ARIMA model. An ARIMA model can be created using the statsmodels library as follows:

1. the model by calling ARIMA() and passing in the p, d, and q parameters.
2. The model is prepared on the training data by calling the fit() function.
3. Predictions can be made by calling the predict() function and specifying the index of the time or times to be predicted.

We split the dataset in train and test set. Train set has 66% of the total observation and train has 44% observations. We use the train set to fit the model, and generate a prediction for each element on the test set. First, we fit an ARIMA(5,1,0) model. This sets the lag value to 5 for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 0. When fitting the model, a lot of debug information is provided about the fit of the linear regression model. We can turn this off by setting the disp argument to 0. We use the predict() function on the ARIMAResults object to make prediction of the count of number of crimes in a particular ‘ Beat ’. It accepts the index of the time steps to make predictions as arguments. These indexes are relative to the start of the training dataset used to make predictions. If we used 100 observations in the training dataset to fit the model, then the index of the next time step for making a prediction would be specified to the prediction function as start=101, end=101. This would return an array with one element containing the prediction. The output shows the result based on the given parameters i.e., (5,1,0). The parameters need to be selected which best fit the model on our time series dataset. We use the train set again to fit an ARIMA(5,2,0) model. We prefer the forecasted values to be in the original scale, in case we performed any differencing (d¿0 when configuring the model). This can

be specified by setting the typargument to the value 'levels': type='levels'. Alternately, we can avoid all of these specifications by using the forecast() function, which performs a one-step forecast using the model. A rolling forecast is required given the dependence on observations in prior time steps for differencing and the AR model. Its is performed by re-creating the ARIMA model after each new observation is received. We manually keep track of all observations in a list called history that is seeded with the training data and to which new observations are appended each iteration. Running the model prints the prediction and expected value each iteration. We also calculate a final mean squared error score (MSE) for the predictions, providing a point of comparison for other ARIMA configurations. A line plot is created showing the expected values (blue) compared to the rolling forecast predictions (red). We can see the values show some trend and are in the correct scale.

4.4.2.2 LSTM

From the original dataset, we have created time series dataset which describes daily number of crimes happened in each beat from 2001 to 2018.

Data Preparation and Model Evaluation

From the available Multi-stage forecast strategies we use Multiple Output Forecast Strategy.

Multiple Output Forecast Strategy Direct-Recursive Hybrid Multi-step Forecast Strategies. The direct and recursive strategies can be combined to offer the benefits of both methods.For example, a separate model can be constructed for each time step to be predicted, but each model may use the predictions made by models at prior time steps as input values.We can see how this might work for predicting the crime counts for the next two days, where two models are used, but the output from the first model is used as an input for the second model.

```
prediction(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n))
prediction(t+2) = model2(prediction(t+1), obs(t-1), ..., obs(t-n))
```

Prepare Data:

Before we can use it to train an LSTM, the data must be prepared so that it can easily feed

to neural network. Specifically, two additional changes are required for multistep output forecasting:

1. Stationary. The data shows an increasing trend that must be removed by differencing.
2. Scale. The scale of the data must be reduced to values between -1 and 1, the activation function of the LSTM units.

We can introduce a function to make the data stationary called difference(). This will transform the series of values into a series of differences, a simpler representation to work with.

Transform data to be stationary:
For the first step ,we will create a function to make the data stationary called difference(). This will transform the series of values into a series of differences, a simpler representation to work with.
Rescale values to -1, 1 :
We can use the MinMaxScaler from the sklearn library to scale the data.
Transform into supervised learning problem X, y :
The next step is to transform the data from a series into a supervised learning problem. That is to go from a list of numbers to a list of input and output patterns. We will achieve this using a pre-prepared function called series_to_supervised().The function series_to_supervised() converts time series dataset into supervised learning problem. For a given beat, we will be required to make a next two days forecast based on previous crime counts for the last 2 years. That is given historical observations $\big($t-1, t-2, ... t-n$\big)$ forecast t, t+1.

- n_lags = 365*2 = 730 (Last 730 observations)
- n_seq = 2 (Forecasting observation)
- n_test = 10 (No. of test observation)

Split into train and test sets:

We will be predicting two days forecast for the last 10 days timestep. Fit LSTM Network. Next, we need to fit an LSTM network model to the training data. This first requires that the training dataset be transformed from a 2D array [samples, features] to a 3D array [samples, timesteps, features]. We will fix time steps at 1, so this change is straightforward. Samples will be equal 5799 and no of features is equal to $365 * 2 = 730$ i.e. last 2 years observations of crime counts.

Next, we need to design an LSTM network. We will use a simple structure i.e. 1 LSTM unit with (30 neurons, dropout = 0.2, stateful = True), then a hidden layer of LSTM with (60 neurons, stateful = True) and last an output layer with linear activation and 2 output values. The network will use a mean squared error loss function and the efficient ADAM optimization algorithm. The LSTM is stateful and the network will be fit for 50 epochs.The same batch size (i.e.1) will be used for training and prediction, and we require predictions to be made at each time step of the test dataset. This means that a batch size of 1 must be used. We will put all of this together in a function called fit_lstm(). The function takes a number of key parameters that can be used to tune the network later and the function returns a fit LSTM model ready for forecasting and history of model during training.

Make LSTM Forecasts:

The next step is to use the fit LSTM network to make forecasts. A single forecast can be made with the fit LSTM network by calling model.predict(). Again, the data must be formatted into a 3D array with the format [samples, timesteps, features]. We will put this into a function called forecast_lstm(). Now we want to predict for the last 10 days timestep. We will iterate through test data and store its prediction in forecast variable. We will put this in make_forecasts() function. It will give next two days crime counts for the last 10 days timestep. Invert Transforms After the forecasts have been made, we need to invert the transforms to return the values back into the original scale as we have converted them into supervised learning dataset. This is needed so that we can calculate error scores and create plots of forecast and actually happened crime counts. We can invert the scale of the forecasts directly using the MinMaxScaler object. We can invert the differencing by

adding the value of the last observation to the first forecasted value, then propagating the value down the forecast. We will put this in a function name inverse_difference() that takes the last observed value prior to the forecast and the forecast as arguments and returns the inverted forecast. We will create an inverse_transform() function that works through each forecast, first inverting the scale and then inverting the differences, returning forecasts to their original scale. To calculate RMSE, we will create a function evaluate_forecasts which will take test and forecaste crime counts and calculate RMSE.

Chapter 5

Implementation

5.1 Implementation

5.2 Linear Regression

```
from sklearn import linear_model
#Importing PCA library
from sklearn.decomposition import PCA
pca = PCA()
reg = linear_model.LinearRegression()
Data = Data.dropna(axis = 0, how = 'any')
Y = Data['Arrest']
Data.drop(['Arrest'], inplace = True, axis = 1)
#Using PCA to optimize fit for each algorithm
pca.fit(Data[0:28])
PCAData = pca.fit_transform(Data)
X = PCAData
print(pca.explained_variance_ratio_)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 30)
reg.fit(X_train,Y_train)
Reg_Expected_Y = reg.predict(X_test)
for i in range(len(Reg_Expected_Y)):
    if (Reg_Expected_Y[i] >= 0.5):
        Reg_Expected_Y[i] = True
    else:
```

```
            Reg_Expected_Y[i] = False
```

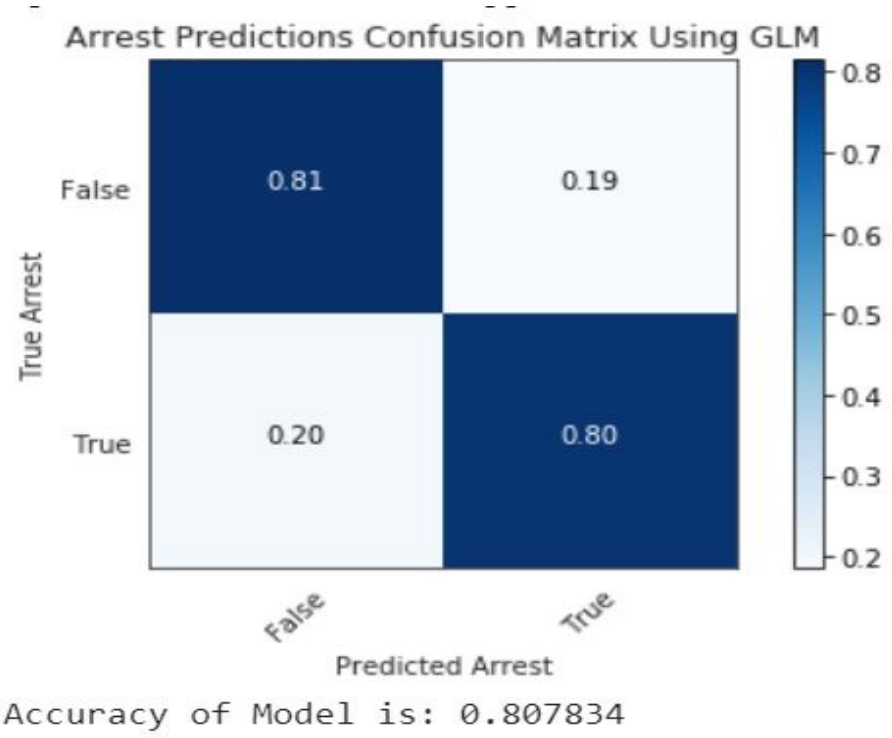


FIGURE 5.1: Confusion Matrix for Regression

5.3 Bernoulli Naive Bayes

```
from sklearn.naive_bayes import BernoulliNB
BNB = BernoulliNB()
BNB.fit(X_train, Y_train)
BNB_Expected_Y = BNB.predict(X_test)
cm = confusion_matrix(Y_test, BNB_Expected_Y)
plt.figure()
plot_confusion_matrix(cm, normalize = True, classes = Y.unique(),title = 'Arrest Predictions Co
plt.show()
print("Accuracy of Model is: %f"%accuracy_score(Y_test, BNB_Expected_Y))
```

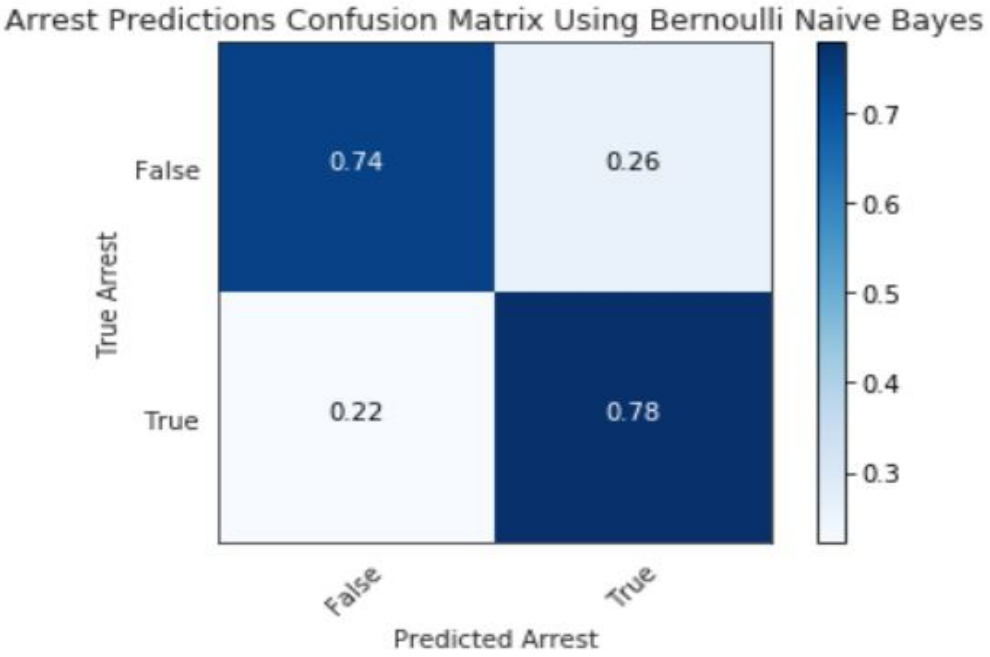


FIGURE 5.2: Confusion Matrix for Bernoullis Naive Bayes

5.4 Random Forest

```
from sklearn.ensemble import RandomForestClassifier
OOB_Err = list(range(50,100))
for i in range(50,100):
    rfc = RandomForestClassifier(n_estimators = i, oob_score = True, n_jobs = -1)
    rfc.fit(X_train,Y_train)
    OOB_Err[i-50] = 1 - rfc.oob_score_
rforest = RandomForestClassifier(n_estimators = 90, n_jobs = -1)
rforest.fit(X_train, Y_train)
RF_Expected_Y = rforest.predict(X_test)
#Very accurate compared to previous project and much faster
#Running for cumulative crimes led to a 4-hour wait and an accuracy below 50%
cm = confusion_matrix(Y_test, RF_Expected_Y)
plt.figure()
plot_confusion_matrix(cm, normalize = True, classes =  Y.unique(),title = 'Arrest Predictions C
plt.show()
print("Accuracy of Model is: %f"%accuracy_score(Y_test, RF_Expected_Y))
```

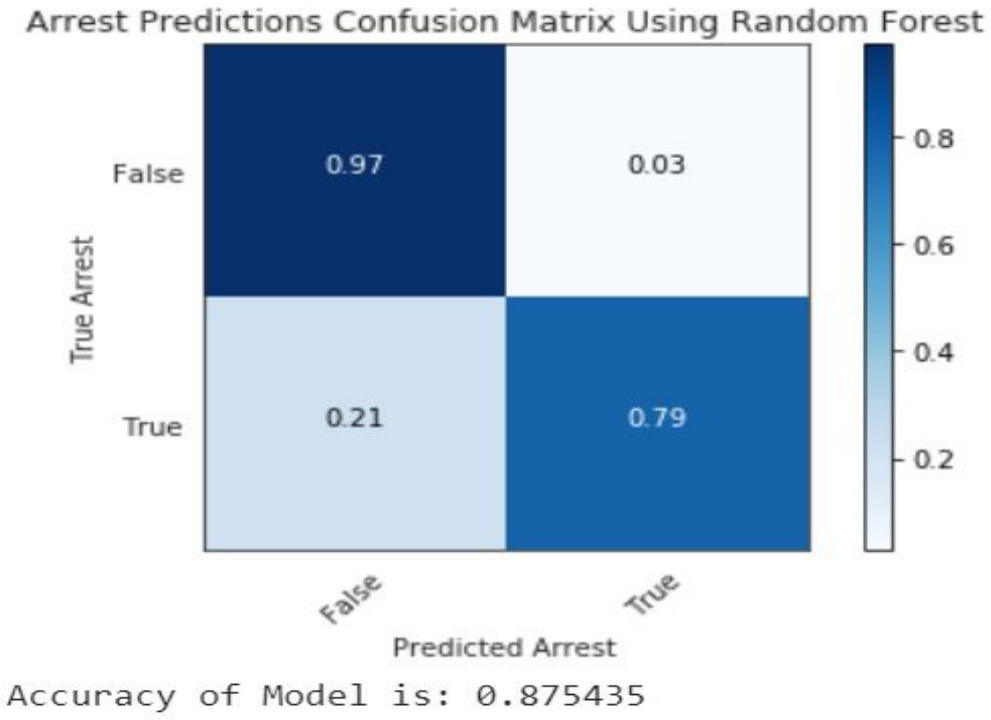


FIGURE 5.3: Confusion Matrix for Random Forest

5.4.1 Mapping Of Incidents Count

```
   %%time

chicago_map_crime = folium.Map(location=[41.895140898, -87.624255632],
                        zoom_start=13,
                        tiles="CartoDB dark_matter")

for i in range(500):
    lat = CR_index['LocationCoord'].iloc[i][0]
    long = CR_index['LocationCoord'].iloc[i][1]
    radius = CR_index['ValueCount'].iloc[i] / 45

    if CR_index['ValueCount'].iloc[i] > 1000:
        color = "#FF4500"
    else:
        color = "#008080"

    popup_text = """Latitude : {}<br>
                Longitude : {}<br>
                Criminal Incidents : {}<br>"""
```

```
popup_text = popup_text.format(lat,
                               long,
                               CR_index['ValueCount'].iloc[i]
                               )
folium.CircleMarker(location = [lat, long], popup= popup_text,radius = radius, color = color,
```

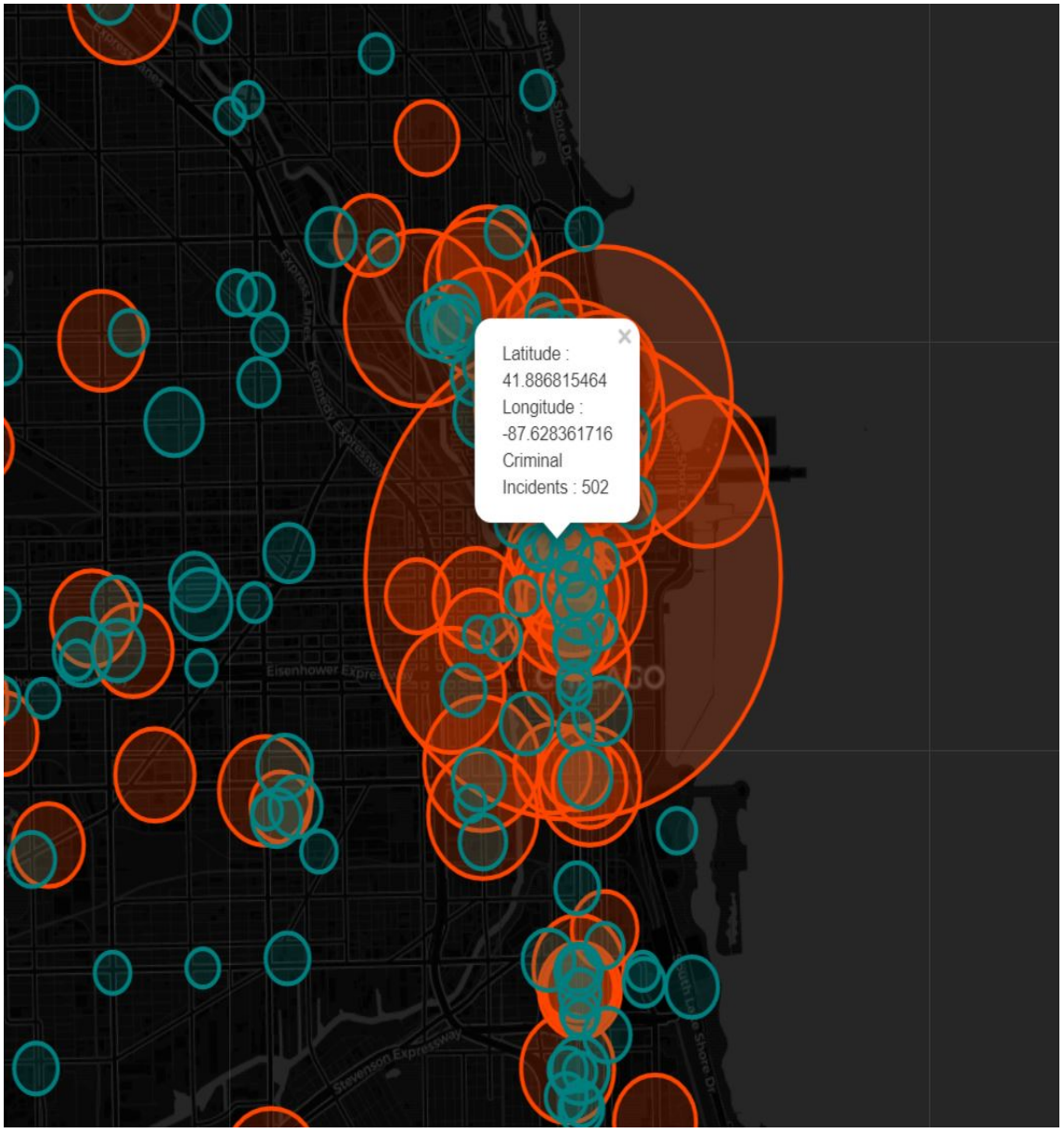


FIGURE 5.4: Heat Map

5.4.2 Arrest Mapping

```
popup_text = """Community Index : {}<br
```

```
                Arrest : {}<br>
                Location Description : {}<br>"""
for i in range(len(new_locations)):
 lat = new_locations.iloc[i][0]
 long = new_locations.iloc[i][1]
 popup_text = """Community Index : {}<br>
                Arrest : {}<br>
                Location Description : {}<br>"""
 popup_text = popup_text.format(new_locations.index[i],
                                new_locations.iloc[i][-1],
                                new_locations.iloc[i][-2]
                                )
 folium.CircleMarker(location = [lat, long], popup= popup_text, fill = True).add_to(chicago_map
 chicago_map
```

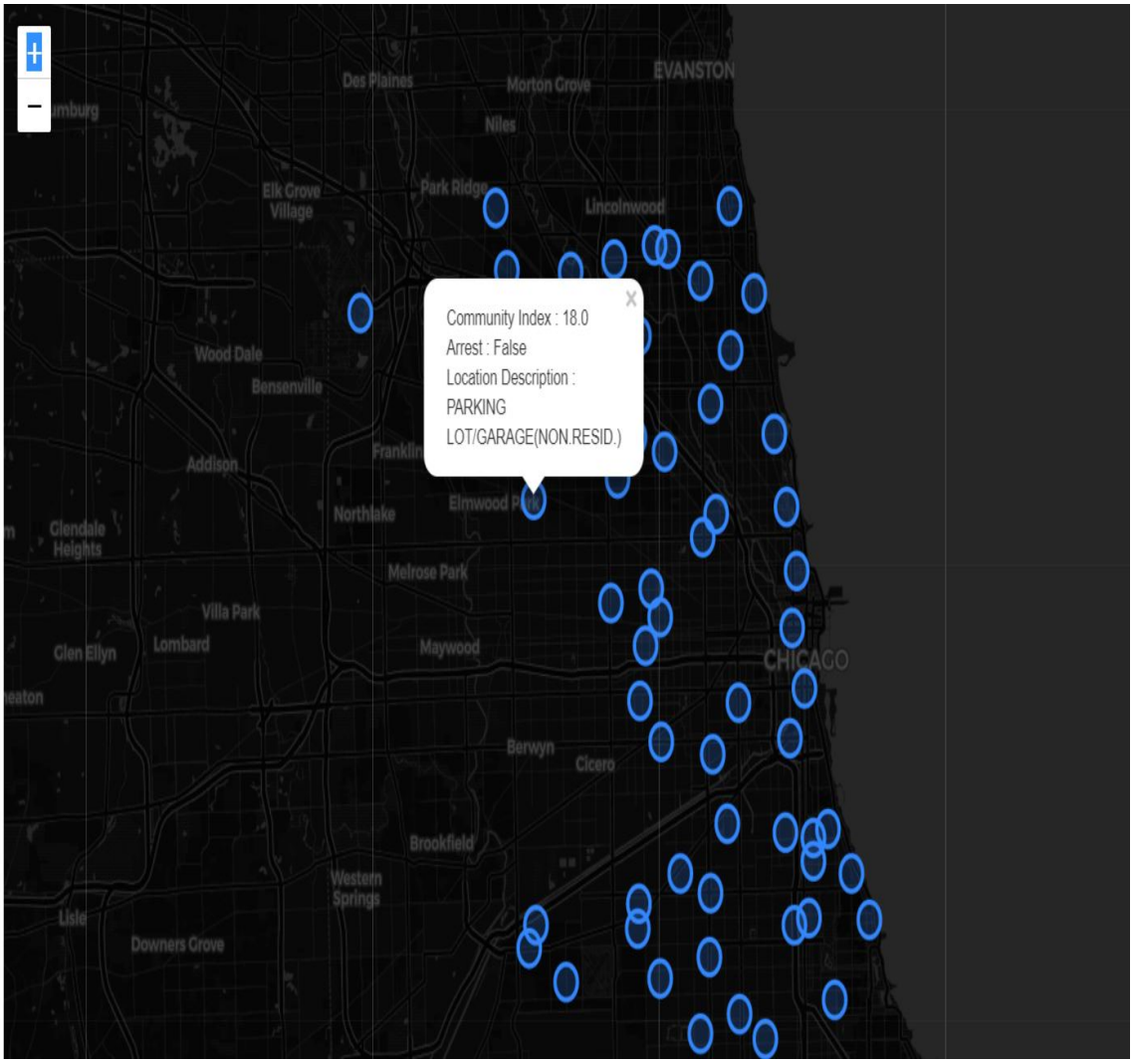


FIGURE 5.5: Heat map of Arrest Prediction

5.5 ARIMA Model

```
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
X = df.loc[:,'111'].values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(5,2,2))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
       history.append(obs)
       print('i=%d predicted=%f, expected=%f' % (t,yhat, obs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

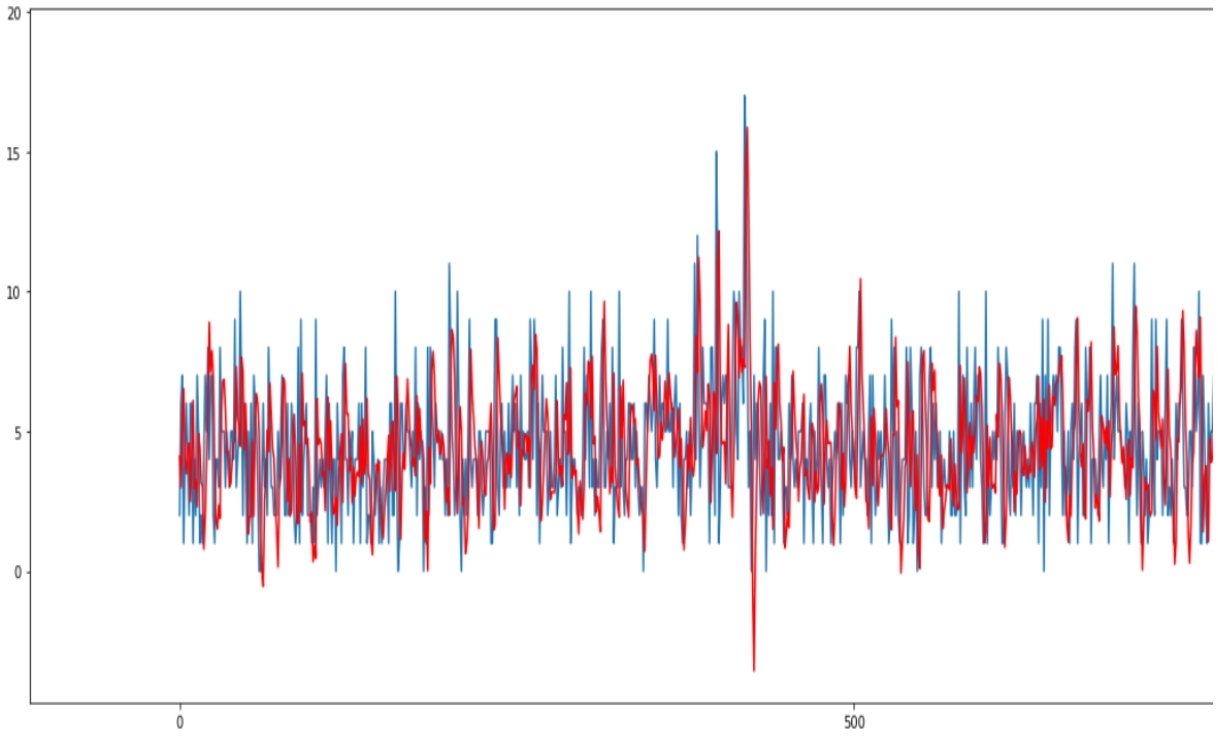


FIGURE 5.6: Arima plot for test vs prediction

```
i=163 predicted=2.376662, expected=1.000000
i=164 predicted=1.136130, expected=6.000000
i=165 predicted=3.523043, expected=6.000000
i=166 predicted=4.214624, expected=4.000000
i=167 predicted=3.636795, expected=4.000000
i=168 predicted=5.517319, expected=6.000000
i=169 predicted=6.870541, expected=5.000000
i=170 predicted=5.778373, expected=5.000000
i=171 predicted=4.813437, expected=4.000000
i=172 predicted=4.423632, expected=3.000000
```

FIGURE 5.7: Arima Predicted vs Actual

5.6 LSTM

```
 def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):

X, y = train[:, 0:n_lag], train[:, n_lag:]
X = X.reshape(X.shape[0], 1, X.shape[1])
print(X.shape)
print(X)
print(y.shape)
print(y)
model = Sequential()
model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.shape[1], X.shape[2]), stateful=True, dr
model.add(LSTM(n_neurons*2, return_sequences=False, stateful=True))
model.add(Dense(y.shape[1]))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])
model.summary()
history = model.fit(X, y, epochs=nb_epoch, batch_size=n_batch, verbose=2, shuffle=False,validati
return history,model
```

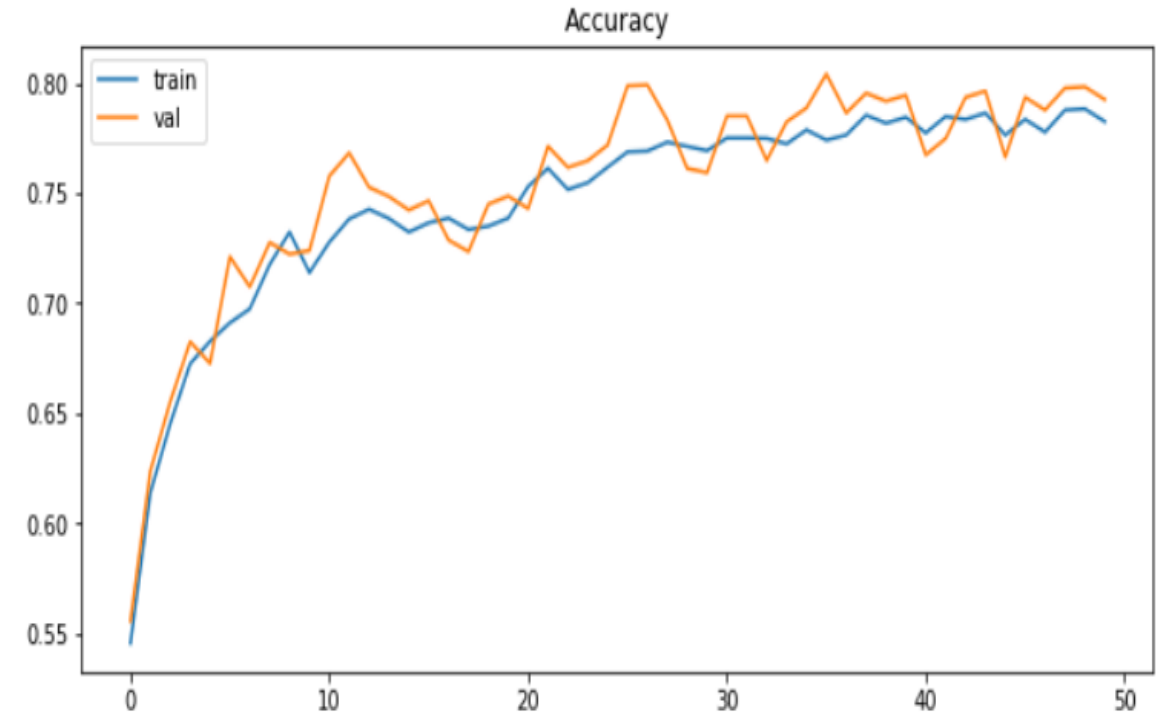


FIGURE 5.8: Lstm Output

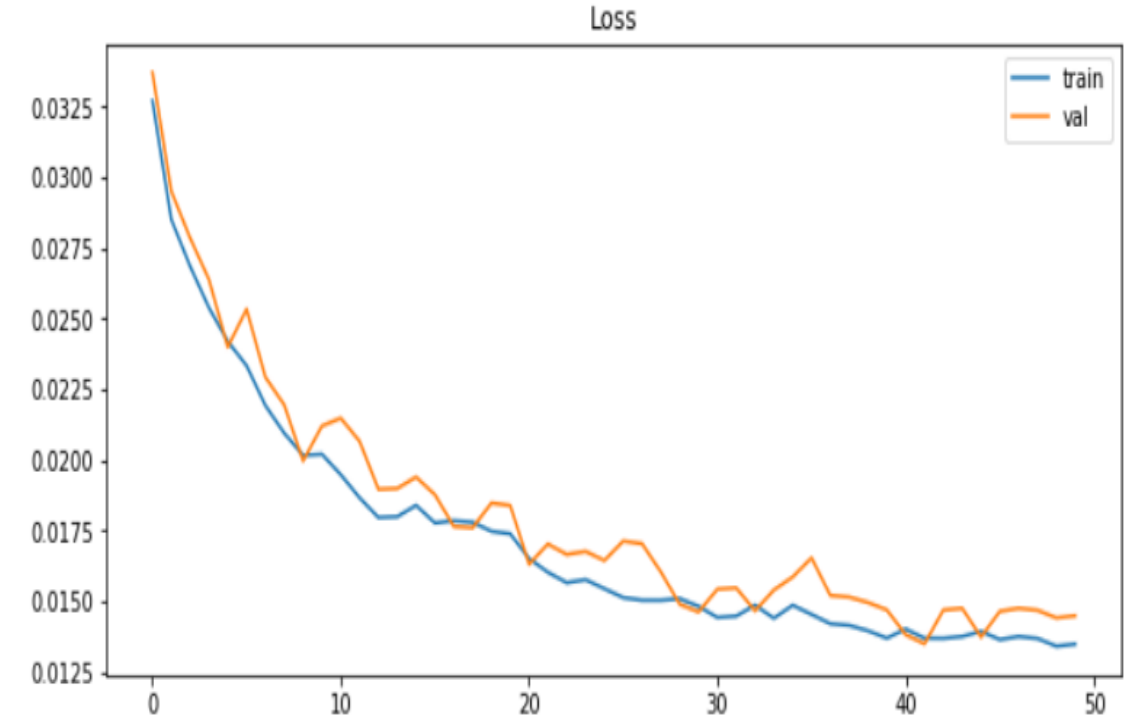


FIGURE 5.9: Loss plot for train and values

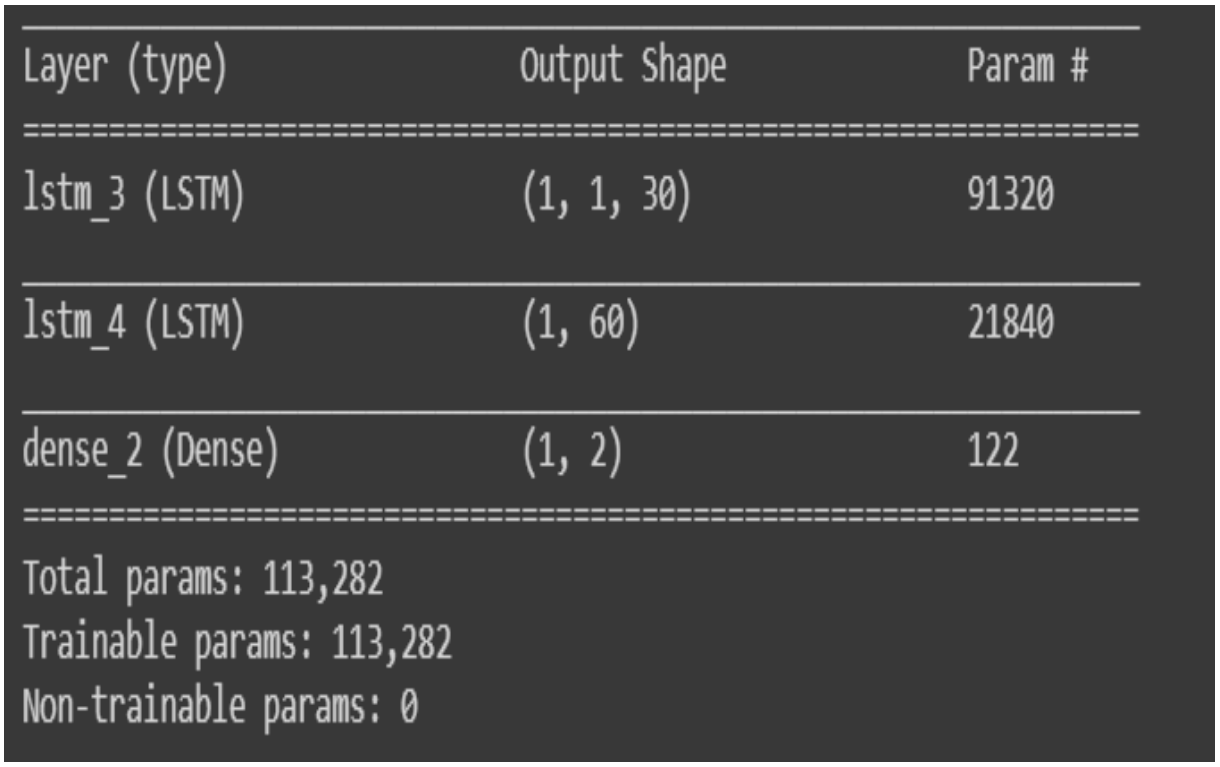


Figure 5.10: Lstm Output

Chapter 6

Conclusion

- Crime detection is the dynamic and emerging research field in the real world environment which aims to prevent the crime rates.
- Data Mining and Machine learning plays a vital role in law enforcement agencies in crime analysis in terms of crime detection and prevention.
- The Long Short-Term Memory network can learn and forecast long sequences. A benefit of LSTMs in addition to learning long sequences is that they can learn to make a one-shot multi-step forecast which is useful for time series forecasting.
- It has been observed that 'Location Description', etc features influence in determining the arrest.
- Linear regression and Bernoullis Naive Bayes models have lesser accuracy but use less computations. Random forest preditcs better arrests but uses lots of computing power.
- ARIMA uses statistical regression modelling to conver a time-series data into stationary data which can be used to predict number of crimes.
- Deep learning models are used to forecast the count and predict weather arrest will happen or not.

Bibliography

[1] 4. Nafiz Mahmud, Khalid Ibn Zinnah, Y. A. R. N. A. C. C. (2016). :a crime prediction and decision service.

[2] B.Chandra, Manish Gupta, M. P. G. (2013). : A multivariate time series clustering approach for crime trends prediction.

[3] Guiyun Zhou, Jiayuan Lin, W. Z. (2016). A web-based geographical information system for crime mapping and decision support.

[4] José Florencio de Queiroz Neto, Emanuele Santos, C. A. V. (2016). : Mskde - using marching squares to quickly make high quality crime hotspot maps.

[5] Julio Borges, Daniel Ziehr, M. B. (2015). : Feature engineering for crime hotspot detection.

[6] Khushboo Sukhija, Shailendra Narayan Singh, J. K. (2014). : Spatial visualization approach for detecting criminal hotspots: An analysis of total cognizable crimes in the state of haryana.

[7] Maria Jeseca C. Baculo, Charlie S. Marzan, R. d. D. B. C. R. (2016). : : Geospatial-temporal analysis and classification of criminal data in manila.

[8] Md. Abdul Awal, Jakaria Rabbi, S. I. H. and Hashem:, M. M. A. (2016). Using linear regression to forecast future trends in crime of bangladesh.

[9] Shoaib KhalidI3, liechen Wangl, M. S. X. N. (2016). : Spatio-temporal analysis of the street crime hotspots in faisalabad city of pakistan.

[10] S.Sivaranjani, Dr.S.Sivakumari, A. (2014). : Crime prediction and forecasting in tamilnadu using clustering approaches.

[11] Xiang Zhang, Zhiang Hu, R. L. and Zheng, Z. (2018). :detecting and mapping crime hot spots based on improved attribute oriented induce clustering.

[12] Yelwa, S. A. (2012). : Complementing gis with cluster analysis in assessing property crime in katsina state, nigeria.

[13] Yong Zhuang, Matthew Almeida, M. M. and Ding, W. (2017). : Crime hot spot forecasting: A recurrent model with spatial and temporal information.

[14] Zhanhong Wang, Jianping Wu, B. Y. (2013). :analyzing spatio-temporal distribution of crime hot-spots and their related factors in shanghai, china.