**MACHINE LEARNING BASED CRIME ANALYSIS**

**USING SVM**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO.** |
| **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **ABSTRACT** | ⅷ  ⅸ  ⅻ  xiii |
| **CHAPTER 1: INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM DISADVANTAGES   4. PROPOSED SYSTEM   5. LITERATURE SURVEY | **1-5**  1  1  1  1  2 |
| **CHAPTER 2: PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4 GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM | **6-7**  6  6  6  6  6  6  7 |
| **CHAPTER 3: REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS | **8-9**  8  8  9 |
| **CHAPTER 4: SYSTEM DESIGN**  4.1 general  4.2 UML DIAGRAMS  4.2.1 USE CASE diagram  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 COMPONENT DIAGRAM  4.2.5 DEPLOYMENT DIAGRAM  4.2.6 SEQUENCE DIAGRAM  4.2.7 COLLABORATION DIAGRAM  4.2.8 STATE DIAGRAM  4.2.9 ACTIVITY DIAGRAM  4.2.10 DATA FLOW DIAGRAM  4.3 SYSTEM ARCHITECTURE | **10-21**  10  10  10  11  12  13  14  15  16  17  18  19-20  21 |
| **CHAPTER 5: DEVELOPMENT TOOLS**  5.1 PYTHON  5.1.1 HISTORY OF PYTHON  5.1.2 IMPORTANCE OF PYTHON  5.1.3 FEATURES OF PYTHON  5.1.4 LIBRARIES USED IN PYTHON | **22-24**  22  22  22  23  24 |
| **CHAPTER 6: IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION | **25-32**  25  25 |
| **CHAPTER 7: SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS | **33-35**  33  33 |
| **CHAPTER 8: SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING  8.3.1 UNIT TESTING  8.3.2 FUNCTIONAL TEST  8.3.3 SYSTEM TEST  8.3.4 PERFORMANCE TEST  8.3.5 INTEGRATION TESTING  8.3.6 ACCEPTANCE TESTING  8.3.7 BUILD THE TEST PLAN | **36-38**  36  36  36  36  37  37  37  37  38  38 |
| **CHAPTER 9:**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 FUTURE ENHANCEMENTS | **39**  39 |
| **CHAPTER 10:**  10.1 CONCLUSION  **REFERENCES** | **40**  40  **41** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1 | Use Case Diagram | 10 |
| 4.2 | Class Diagram | 11 |
| 4.3 | Object diagram | 12 |
| 4.4 | Component diagram | 13 |
| 4.5 | Deployment diagram | 14 |
| 4.6 | Sequence diagram | 15 |
| 4.7 | Collaboration Diagram | 16 |
| 4.8 | State Diagram | 17 |
| 4.9 | Activity Diagram | 18 |
| 4.10 | Data Flow Diagram | 19-20 |
| 4.11 | System Architecture | 21 |
| 5.1 | Python Libraries | 24 |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class A  Class B  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single class. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | Database |
| 2. | SVM | Support Vector Machine |
| 3. | KNN | K Nearest Neighbor |
| 4. | NCRB | National Crime Records Bureau |
| 5. | Numpy | Numerical Python |
| 6. | RTA | Road Traffic Accidents |

**ABSTRACT**

This project investigates machine-learning-based crime prediction. Crime analysis is a systematic way of detecting and investigating patterns and trends in crime. In this project, we use various clustering approaches of data mining to analyze the crime data of Telangana. The crime data is extracted from National Crime Records Bureau (NCRB) of India. It consists of crime information about 15 years. SVM is used to cluster crime activities based on some predefined cases and the results of these clustering are compared to find the best suitable clustering algorithm for crime detection. The K-Nearest Neighbor (KNN) classification is used for crime prediction. The performance of each clustering algorithms is evaluated using the metrics such as precision, recall and F-measure, and the results are compared. This work helps the law enforcement agencies to predict and detect crimes in Telangana with improved accuracy and thus reduces the crime rate.

**CHAPTERS**

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Crime is a socio-economical problem affecting life quality and economic growth. Previous researches in crime prediction have found that factors like education, poverty, employment, and climate affect the crime rate. Vancouver is one of the most populous, ethnically-diverse, and multi-cultural urban cities in Canada. The overall crime rate in Vancouver dropped 1.5% in 2017, but high vehicle break-ins and theft is still an issue. Recently, the Vancouver Police Department (VPD) introduced a crime predictive model to predict crimes related to property break-ins and, once implemented, the city of Vancouver witnessed a 27% drop in residential break-ins.

* 1. **OBJECTIVE**

1. The primary objective of this work is to create a prediction model that can accurately predict crime.
2. In our research, two classification algorithms, K-Nearest Neighbor (KNN) and boosted decision tree, were implemented to analyze the VPD crime dataset compiled between 2003 and 2018 with more than 560,000 records.
   1. **EXISTING SYSTEM**
3. Recently, machine learning has been applied in self-driving cars, speech recognition, web search, and an improved understanding of the human genome.
4. It has also made predicting crime based on referenced data feasible.
5. Classification has been used in many domains including weather forecasting, medical care, finances and banking, homeland security, and business intelligence.
6. In the first approach, each neighborhood and crime category were given a unique number when a certain crime happens in a certain neighborhood.
7. In the second approach, the neighborhood and the day of the week during which the crime was committed were given a binary number and marked as 1 when the crime happened on that day in that neighborhood, and 0 otherwise.
   1. **PROPOSED SYSTEM**
   2. **LITERATURE SURVEY**
      1. **ONCE UPON A CRIME: TOWARDS CRIME PREDICTION FROM DEMOGRAPHICS AND MOBILE DATA**

**Author:** Andrey Bogomolov, Bruno Lepri, Jacopo Staiano.

**Year:** 2014

**Description:**

In this paper, we present a novel approach to predict crime in a geographic space from multiple data sources, in particular mobile phone and demographic data. The main contribution of the proposed approach lies in using aggregated and anonymized human behavioral data derived from mobile network activity to tackle the crime prediction problem. While previous research efforts have used either background historical knowledge or offenders' profiling, our 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. In our experimental results with real crime data from London we obtain an accuracy of almost 70% when predicting whether a specific area in the city will be a crime hotspot or not. Moreover, we provide a discussion of the implications of our findings for data-driven crime analysis.

* + 1. **CRIME IN RELATION TO URBAN DESIGN CASE STUDY: THE GREATER CAIRO REGION**

**Author****:** [Heba Adel](https://www.sciencedirect.com/science/article/pii/S2090447915001379#!) , [Mohamed Salheen ,](https://www.sciencedirect.com/science/article/pii/S2090447915001379#!)  [Randa A.Mahmoud](https://www.sciencedirect.com/science/article/pii/S2090447915001379#!) .

**Year:** 2016

**Description:**

Crime is a part of any social system and known to human communities since its origins. It differs from community to another, even within one community it doesn’t occur equally in all places and nor by the same way. It is also concentrated in some places more than others, sometimes increases, sometimes decreases, etc. Previous researches have proved that crime rate has significant correlation with different social factors: education levels, poverty rates and lack of social organization, while others have drawn the attention to its relation with the built environment. They proposed that crime occurs in places where both opportunities and criminals are available. The role of this paper is to identify urban circumstances related to crime occurrence within the Greater Cairo Region, and to propose different ways to reduce these crimes. Consecutively, agglomeration’s main districts were scrutinized according to social analysis, street-network pattern and land-use.

* + 1. **CRIME DATA MINING: A GENERAL FRAMEWORK AND SOME EXAMPLES**

**Author:** H. Chen; W. Chung; J.J. Xu; G. Wang.

**Year:** 2004

**Description:**

A major challenge facing all law-enforcement and intelligence-gathering organizations is accurately and efficiently analyzing the growing volumes of crime data. Detecting cybercrime can likewise be difficult because busy network traffic and frequent online transactions generate large amounts of data, only a small portion of which relates to illegal activities. Data mining is a powerful tool that enables criminal investigators who may lack extensive training as data analysts to explore large databases quickly and efficiently. We present a general framework for crime data mining that draws on experience gained with the Cop link project, which researchers at the University of Arizona have been conducting in collaboration with the Tucson and Phoenix police departments since 1997.

* + 1. **MINING ROAD TRAFFIC ACCIDENT DATA TO IMPROVE SAFETY: ROLE OF ROAD-RELATED FACTORS ON ACCIDENT SEVERITY IN ETHIOPIA**

**Author:** Tibebe Beshah, Shawndra Hill.

**Year:** 2010

**Description:**

Road traffic accidents (RTAs) are a major public health concern, resulting in an estimated 1.2 million deaths and 50 million injuries worldwide each year. In the developing world, RTAs are among the leading cause of death and injury; Ethiopia in particular experiences the highest rate of such accidents. Thus, methods to reduce accident severity are of great interest to traffic agencies and the public at large. In this work, we applied data mining technologies to link recorded road characteristics to accident severity in Ethiopia, and developed a set of rules that could be used by the Ethiopian Traffic Agency to improve safety.

* + 1. **AREA-SPECIFIC CRIME PREDICTION MODELS**

**Author:** Mohammad Al Boni; Matthew S. Gerber

**Year:** 2016

**Description:**

The convergence of public data and statistical modeling has created opportunities for public safety officials to prioritize the deployment of scarce resources on the basis of predicted crime patterns. Current crime prediction methods are trained using observed crime and information describing various criminogenic factors. Researchers have favored global models (e.g., of entire cities) due to a lack of observations at finer resolutions (e.g., ZIP codes). These global models and their assumptions are at odds with evidence that the relationship between crime and criminogenic factors is not homogeneous across space. In response to this gap, we present area-specific crime prediction models based on hierarchical and multi-task statistical learning. Our models mitigate sparseness by sharing information across ZIP codes, yet they retain the advantages of localized models in addressing non-homogeneous crime patterns. Out-of-sample testing on real crime data indicates predictive advantages over multiple state-of-the-art global models.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

This paper investigates machine-learning-based crime prediction. In this work, Vancouver crime data for the last 15 years is analyzed using two different data-processing approaches.

**2.2 METHODOLOGIES**

* + 1. **MODULES NAME:**
* **DATA SOURCE:**

The original datasets were obtained from the open data catalog of the city of Vancouver. There are two datasets used for this project: crime and neighborhood. The crime dataset has been collected by the VPD since 2003 and is updated every Sunday morning. It provides information on the type of crime committed and the time and location of the offence. The neighborhood dataset contains the boundaries for the city’s 22 local areas in the Geographic Information System (GIS).

* **PREPROCESSING:**

The original dataset needs to be preprocessed to fill the empty cells, delete unnecessary columns, and add several relevant features to the original and preprocessed datasets.

* **STATISTICAL ANALYSIS:**

The distribution of the crime dataset described in based on year, month, and day. In Vancouver, the average number of crime incidents is around 31624 per year, 2720 per month, and 90 per day. The dataset tends to show a normal distribution as the time intervals lengthen. However, the graph of each day has an abnormal max value of 650 incidents, which is suspected as an outlier - and turns out to indicate the Stanley- Cup riot on June 15, 2011.

* **CLASSIFICATION:**

After statistical analysis we classify the prediction values of the crime rate in Vancouver. Machine Learning predictive models KNN and boosted decision tree were used to obtain crime-prediction accuracy between 39% to 44%. The accuracy, complexity, and training time of algorithms were slightly different for different approaches and algorithms. Finally, we get the prediction of the Vancouver crime rate.

* 1. **TECHNIQUE USED OR ALGORITHM USED**

**Support Vector Machine**

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e., the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

Vehicle detection module detects the presence of vehicle by using inductive sensors in which metal wire loop is placed beneath the road. When a vehicle crosses the loop, there is change in induced current which detects presence of vehicle. As a result, the DSP is interrupted and it triggers the IR camera to capture the image.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Operating System : Windows 7/8/10
* Platform : Spyder3
* Programming Language : Python, HTML
* Front End : Spyder3

**CHAPTER 4**

**DESIGN ENGINEERING**

* 1. **GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

* 1. **UML Diagrams**
     1. **USE CASE DIAGRAM**

**Fig 4.1 Use Case Diagram**

**EXPLANATION**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists system as an actor. Each will play a certain role to achieve the concept.

* + 1. **CLASS DIAGRAM**



**Fig 4.2 Class Diagram**

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

* + 1. **OBJECT DIAGRAM**



**Fig 4.3 Object Diagram**

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

* + 1. **COMPONENT DIAGRAM**



**Fig 4.4 Component Diagram**

**EXPLANATION**

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.

* + 1. **DEPLOYMENT DIAGRAM**



**Fig 4.5 Deployment Diagram**

**EXPLANATION:**

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

* + 1. **SEQUENCE DIAGRAM**

 **Fig 4.6 Sequence Diagram**

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

* + 1. **COLLABORATION DIAGRAM**



**Fig 4.7 Collaboration Diagram**

**EXPLANATION**

A collaboration diagram shows the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram. The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

* + 1. **STATE DIAGRAM**



**Fig 4.8 State Diagram**

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.9 ACTIVITY DIAGRAM**



**Fig 4.9 Activity Diagram**

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

* + 1. **DATA FLOW**

LEVEL0:

DATABASE

PREPROCESSING

DATA ANALYSIS

CLASSIFICATION

LEVEL1:

Preprocessing

Data analysis

Classification, types of crimes

Testing data,

Training data

Crime prediction

Integration,

cleaning

**Fig 4.10 Data Flow Diagram**

**EXPLANATION**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

* 1. **SYSTEM ARCHITECHTURE**

Preprocessing

Trend Analysis

SVM

TYPES OF CRIMES

Statistical Analysis

DATABASE

LOCATION

PERDICTION

**Fig 4.11 System Architecture**

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 PYTHON**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.1.1 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**5.1.2 IMPORTANCE OF PYTHON**

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.1.3 FEATURES OF PYTHON

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.1.4 LIBRARIES USED IN PYTHON**

* NumPy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as data frames.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



**Fig 5.1 Python Libraries**

**CHAPTER 6**

**IMPLEMENTATION**

* 1. **GENERAL**
     1. **IMPLEMENTATION:**

# -\*- coding: utf-8 -\*-

import NumPy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g., pd.read\_csv)

import MatPlotlib.pyplot as plt

from subprocess import check\_output

crimes1 = pd.read\_csv('input/crime/42\_District\_wise\_crimes\_committed\_against\_women\_2001\_2012.csv')

crimes2 = pd. read\_csv('input/crime/42\_District\_wise\_crimes\_committed\_against\_women\_2013.csv')

crimes = pd.concat([crimes1, crimes2], ignore\_index=False, axis=0)

# Rename the STATE/UT column to STATE

crimes.rename (columns={'STATE/UT':'STATE'}, inplace=True)

# delete data sets post concat

del crimes1

del crimes2

print('Dataset is ready....')

# know the shape of dataset

crimes.shape

# collect the state names in a list and print

states = crimes.STATE.unique()

print(states)

# do some data cleansing on state names

for i in range(0, len(states)):

states[i] = states[i].lower()

for j in range(0, len(states)):

if states[j] == 'a & n islands':

states[j] = 'a&n islands'

if states[j] == 'd & n haveli':

states[j] = 'd&n haveli'

print(states)

# remove duplicate state names from the list

states = np.unique(states).tolist()

print(states)

# convert the state names to lower

crimes['STATE'] = crimes['STATE'].str.lower()

crimes.head(3)

# filter out the Total crimes for each State & UT

crimes\_total = crimes[crimes['DISTRICT'] == 'TOTAL']

# drop DISTRCT Column as we do not intend to use at this point

crimes\_total.drop('DISTRICT', axis=1, inplace=True)

# filter out the Total crimes for each State & UT for the year 2001

crimes\_total\_2001 = crimes\_total[crimes\_total['Year'] == 2001]

crimes\_total\_2001.drop('Year', axis=1, inplace=True)

# Data of Rape crime committed in the year 2001 per state

x = crimes\_total\_2001['STATE'].values

y = crimes\_total\_2001['Rape'].values

# plot the bar graph

fig, ax = plt.subplots()

crime\_rape = crimes\_total\_2001['STATE'].values

y\_pos = np.arange(len(crime\_rape))

performance = crimes\_total\_2001['Rape'].values

ax.barh(y\_pos, performance, align='center',color='green', ecolor='black')

ax.set\_yticks(y\_pos)

ax.set\_yticklabels(crime\_rape)

ax.invert\_yaxis() # labels read top-to-bottom

ax.set\_xlabel('Rapes')

ax.set\_title('RAPE VS STATE')

fig.set\_size\_inches(20, 18, forward=True)

plt.show()

# Any results you write to the current directory are saved as output.

# creating a new data set

crimes\_total\_women1 = pd.read\_csv('input/crime/42\_District\_wise\_crimes\_committed\_against\_women\_2001\_2012.csv')

crimes\_total\_women2 = pd.read\_csv('input/crime/42\_District\_wise\_crimes\_committed\_against\_women\_2013.csv')

crimes\_total\_women = pd.concat([crimes\_total\_women1,crimes\_total\_women2], ignore\_index=False, axis=0)

crimes\_total\_women.rename(columns={'STATE/UT':'STATE'}, inplace=True)

del crimes\_total\_women1

del crimes\_total\_women2

# calculating total crimes of all kinds in each state from 2001 to 2013

crimes\_total\_women = crimes\_total\_women[crimes\_total\_women['DISTRICT'] == 'TOTAL'] crimes\_total\_women.drop('DISTRICT', axis=1, inplace=True)

crimes\_total\_women['Total Crimes']= crimes\_total\_women.iloc[:, -9:-1].sum(axis=1)

crimes\_total\_women = crimes\_total\_women.groupby(['STATE'])['Total Crimes'].sum()

# plot graph of crimes committed on women since 2001-2013 in each state/ UT

fig1, ax1 = plt.subplots()

states = crimes\_total\_women.index.tolist()

y\_pos = np.arange(len(states))

performance = crimes\_total\_women.tolist()

ax1.barh(y\_pos, performance, align='center',color='green', ecolor='black')

ax1.set\_yticks(y\_pos)

ax1.set\_yticklabels(states)

ax1.invert\_yaxis() # labels read top-to-bottom

ax1.set\_xlabel('All Crimes Aganist Women')

ax1.set\_title('Crime VS STATE')

fig1.set\_size\_inches(20, 18, forward=True)

plt.show()

#Import dependencies

import numpy as np

import pandas as pd

from urllib.error import HTTPError

from urllib.parse import quote

from urllib.parse import urlencode

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

crime\_df = pd.read\_csv('input/2001\_2012/42\_District\_wise\_crimes\_committed\_against\_women\_2001\_2012.csv')

print(crime\_df.columns)

crime\_df = crime\_df[['STATE/UT', 'DISTRICT','Year','Rape']]

print(crime\_df.head())

crime\_df = crime\_df.loc[crime\_df['STATE/UT'] == "ANDHRA PRADESH"]

crime\_df = crime\_df.loc[crime\_df['DISTRICT'] == "EAST GODAVARI"]

#ANDHRA PRADESH

#SECUNDERABAD RLY.

X = crime\_df.Year.values.reshape(-1, 1)

y = crime\_df.Rape.values.reshape(-1, 1)

print(y)

print("Shape: ", X.shape, y.shape)

plt.scatter(X, y)

plt.show()

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X, y)

print('Weight coefficients: ', model.coef\_)

print('y-axis intercept: ', model.intercept\_)

x\_min = np.array([[X.min()]])

x\_max = np.array([[X.max()]])

y\_min = model.predict(x\_min)

y\_max = model.predict(x\_max)

plt.scatter(X, y, c='blue')

plt.plot([x\_min[0], x\_max[0]], [y\_min[0], y\_max[0]], c='red')

plt.ylabel('Crime Count for 510-Auto Stolen')

plt.xlabel('Month')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20)

from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')

svclassifier.fit(X\_train, y\_train)

y\_pred = svclassifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

**CHAPTER 7**

**SNAPSHOTS**

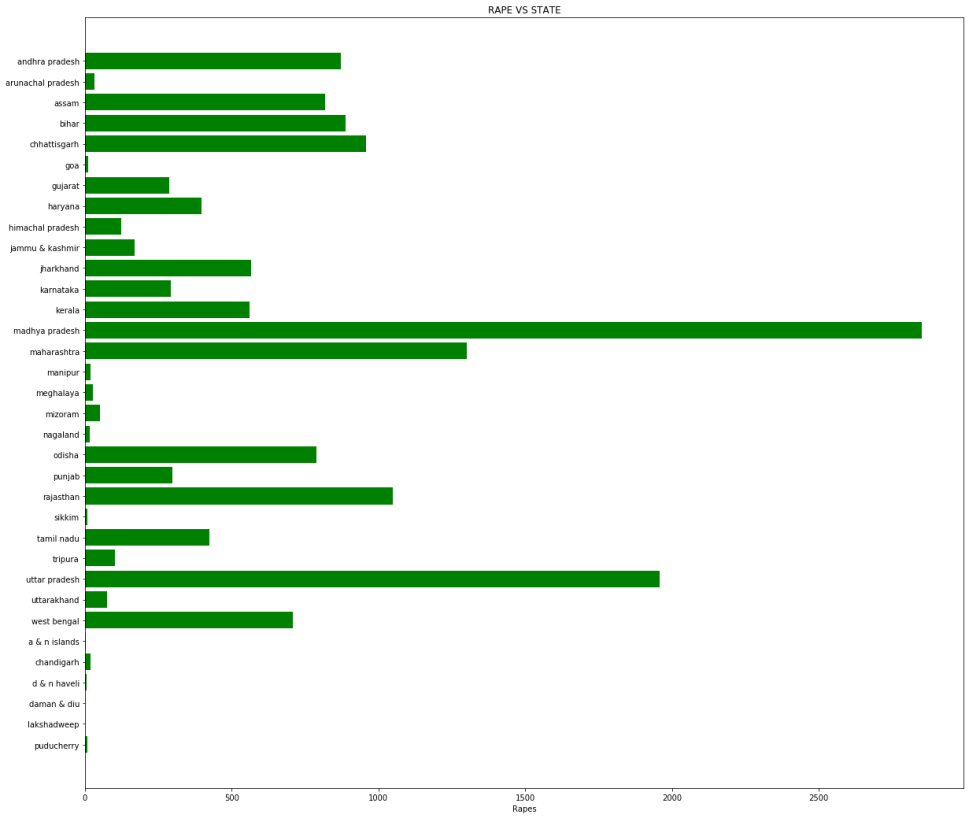
* 1. **GENERAL**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

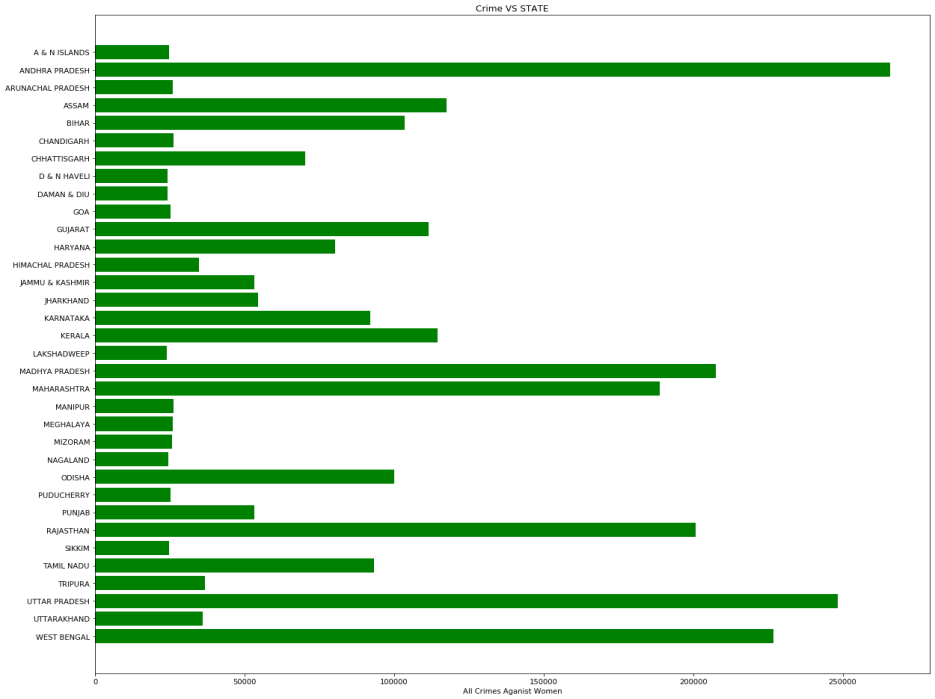
* 1. **SNAPSHOTS**

****

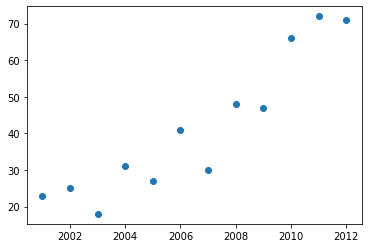
**Figure : 7.2.1**

****

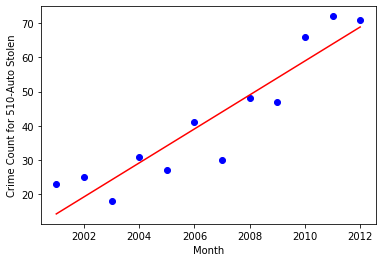
**Figure : 7.2.2**

****

**Figure : 7.2.3**

****

**Figure : 7.2.4**

****

**Figure : 7.2.5**

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional Test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**9.1 FUTURE ENHANCEMENT**

Crime prediction is a law enforcement technique that uses data and statistical analysis for the identification of crimes most likely to occur in the future. This field has been subject to continued research in many parts of the world.

**CHAPTER 10**

**10.1 CONCLUSION**

In this research, Telangana crime data for the last 15 years was used in two different dataset approaches. Machine Learning predictive models KNN and boosted decision tree were used to obtain crime-prediction accuracy between 70 to 80%. The accuracy, complexity, and training time of algorithms were slightly different for different approaches and algorithms. The prediction accuracy can be improved by tuning both the algorithm and the data for specific applications. Although this model has low accuracy as a prediction model, it provides a preliminary framework for further analyses.

**REFERENCES**

[1] A. Bogomolov, B. Lepri, J. Staiano, N. Oliver, F. Pianesi, and A. Pentland, "Once upon a crime: towards crime prediction from demographics and mobile data," Proc. of the 16th Intl. Conf. on Multimodal Interaction, pp. 427-434, 2014.

[2] H. Adel, M. Salheen, and R. Mahmoud, "Crime in relation to urban design. Case study: the greater Cairo region," Ain Shams Eng. J., vol. 7, no. 3, pp. 925-938, 2016.

[3] "Overall crime rate in Vancouver went down in 2017, VPD says," CBC News, Feb. 15, 2018. [Online] Available: https://www.cbc.ca/news/canada/british-columbia/crime-rate-vancouver2017-1.4537831. [Accessed: 09- Aug- 2018].

[4] J. Kerr, "Vancouver police go high tech to predict and prevent crime before it happens," Vancouver Courier, July 23, 2017. [Online] Available: https://www.vancourier.com/news/vancouver-police-go-high-tech-topredict-and-prevent-crime-before-it-happens-1.21295288. [Accessed: 09- Aug- 2018]

[5] J. Han, Data mining: concepts and techniques, Morgan Kaufmann, 2012.

[6] R. Iqbal, M. A. A. Murad, A. Mustapha, P. H. Shariat Panahy, and N. Khanahmadliravi, "An experimental study of classification algorithms for crime prediction," Indian J. of Sci. and Technol., vol. 6, no. 3, pp. 4219- 4225, Mar. 2013.

[7] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau, "Crime data mining: a general framework and some examples," IEEE Computer, vol. 37, no. 4, pp. 50-56, Apr. 2004.

[8] T. Beshah and S. Hill, "Mining road traffic accident data to improve safety: role of road-related factors on accident severity in Ethiopia," Proc. of Artificial Intell. for Develop. (AID 2010), pp. 14-19, 2010.

[9] M. Al Boni and M. S. Gerber, "Area-specific crime prediction models," 15th IEEE Intl. Conf. on Mach. Learn. and Appl., Anaheim, CA, USA, Dec. 2016.