Crime Prediction Using K-Nearest Neighboring Algorithm

Guide: Prof.Mangala S.Biradar
*Priyanka Khedkar, **Akshay Kurumkar, ***Ankita Patil, ****Vaishnavi Rasal

SHREE RAMCHANDRA COLLEGE OF ENGINEERING LONIKAND, WAGHOLI, PUNE 412216
[12345] *Department of Computer Engineering, Maharashtra, India*

ABSTRACT

In this era of recent times, crime has become an evident way *of making people and society under trouble. An increasing crime factor leads to an imbalance in the constituency of a country. In order to analyze and have a response ahead this type of criminal activity, it is necessary to understand the crime patterns. This study imposes one such crime pattern analysis by using crime data obtained from Kaggle open source which in turn is used for the prediction of most recently occurring crimes. The major aspect of this project is to estimate which type of crime contributes the most along with time period and location where it has happened. Some machine learning algorithms such as Naive Bayes is implied in this work in order to classify among various crime patterns and the accuracy achieved was comparatively high when compared to precomposed works.*

Keywords: - Crime*, Analyze, Crime patterns, Kaggle, Estimate*

1.1 INTRODUCTION

1.1.1 Project Scope The investigation of crime patterns requires a thorough understanding of all aspects of criminology as well as the ability to recognize trends. To regulate some of these criminal activities, the government will need to devote a significant amount of time and effort to incorporating technology. Numerous studies were conducted by researchers to aid in the analysis of crime patterns and their relationships in a specific location. Machine learning methods are used in this proposed one to uncover matching crime patterns through categorization of temporal and spatial data.

1.1.2 User Classes and characteristics In this system, the user must first login and register. If registration is successful, the user can use the SVM algorithm to classified and predicts ,crime data set. Domains of machine learning and artificial intelligence

1.1.3 Assumptions and Dependencies Using Python language Input as Textual data Dependencies: Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances. Python is a general-purpose programming language, so it can be used for many things. Python is used for web development, AI, machine learning, operating systems, mobile application development, and video games. Python is a relatively easy programming language to learn and follows an organized structure. Python is a general purpose and high level programming language. You can use Python for developing desktop GUI applications, websites and web applications. The simple syntax rules of the programming language further makes it easier for you to keep the code base readable and application maintainable.

2.1 FUNCTIONAL REQUIREMENTS

2.1 System Feature (Functional Requirement) In order to find a solution which can be used as a part of the Reg SOC system, it is necessary to allow its integration with other modules. Research on anomaly-based intrusion detection systems is the most often carried out on the preexisting data sets or in laboratory environments in which simplification concerning infrastructure, data collection or services have been applied. Due to legal and technical limitations, our solution will detect threats through the analysis of Net Flow data and headers from network protocols. In addition, in the real environment it is not possible to obtain labeled teaching and validation datasets, which forces the introduction of adaptation mechanisms already at the deployment stage. In our approach we will try to prepare a suitably scaled model on the basis of the available datasets and then to adjust it in the following steps to the existing network. The models prepared and tuned this way will later become reference models during the implementation of the anomaly detection module in the subsequent networks.

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.3.1 User Interfaces When interacting with user interfaces, do users always get what they expect? For each user interface element in thousands of Desktop App, we extracted the desktop application they invoke as well as the text shown on their screen. This association allows us to detect outliers: User interface elements whose text, context or icon suggests one action, but which actually are tied to other actions.

3.1.2 Hardware Interfaces and software interfaces

Malware is a serious threat to network-connected embedded systems, as evidenced by the continued and rapid growth of such devices, commonly referred to as the Internet of Things. Their ubiquitous use in critical applications require robust protection to ensure user safety and privacy. That protection must be applied to all system aspects, extending beyond protecting the network and external interfaces. Anomaly detection is one of the last lines of defence against malware, in which data-driven approaches that require the least domain knowledge are popular. However, embedded systems, particularly edge devices, face several challenges in applying data-driven anomaly detection, including unpredictability of malware, limited tolerance to long data collection windows, and limited computing/energy resources. In this article, we utilize sub component timing information of software execution, including intrinsic software execution, instruction cache misses, and data cache misses as features, to detect anomalies based on ranges, multi-dimensional Euclidean distance, and classification at run time. Detection methods based on lumped timing range are also evaluated and compared.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 Performance Requirements

In order to meet stringent performance requirements, system administrators must effectively detect undesirable performance behaviours, identify potential root causes, and take adequate corrective measures. The problem of uncovering and understanding performance anomalies and their causes (bottlenecks) in different system and application domains is well studied. In order to assess progress, research trends, and identify open challenges, we have reviewed major contributions in the area and present our findings in this survey. Our approach provides an overview of anomaly detection and bottleneck identification research as it relates to the performance of computing systems. By identifying fundamental elements of the problem, we are able to categorize existing solutions based on multiple factors such as the detection goals, nature of applications and systems, system observability, and detection methods.

3.2.2 Safety Requirements a cumbersome task and practically infeasible in many applications. Therefore, an automated monitoring system is of both fundamental and practical interest.an intelligent solution that uses live camera images to detect workers who breach safety rules by not wearing high-visibility vests. The proposed solution is formulated in the form of an anomaly detection algorithm developed in the random finite set (RFS) framework.

3.2.3 Security Requirements Mob detection is the process of finding outliers in a given dataset. Outliers are the data objects that stand out amongst other objects in the dataset and do not conform to the normal behavior in a dataset. Anomaly detection is a data science application that combines multiple data science tasks like classification, regression, and clustering. The target variable to be predicted is whether a transaction is an outlier or not. Since clustering tasks identify outliers as a cluster, distance-based and density-based clustering techniques can be used in anomaly detection tasks.

3.2.4 Software Quality Attributes software has many qualities attribute that are given below:
Adaptability: This software is adaptable by all users.
Availability: This software is freely available to all users. The availability of the software is easy for everyone.
Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
Reliability: The performance of the software is better which will increase the reliability of the Software.
User Friendliness: Since the software is a GUI application; the output generated is much user friendly in its behavior.
Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.
Security: Users are authenticated using many security phases so reliablesecu- rity is provided.

4.1 SYSTEM REQUIREMENTS

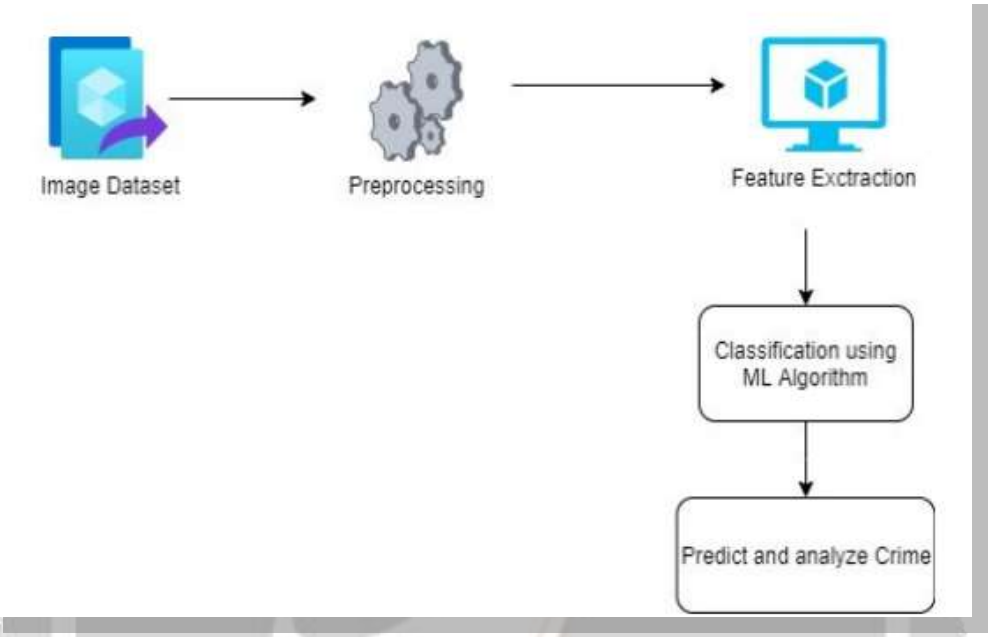
4.1.1 Database Requirements the Database Requirements involve the use of a lot of information, some of which will be needed several times and the most appropriate form of storage of this data is in a database. This will allow data to be saved from input to the Database Requirements and retrieved to be used by the Database Requirements. An important aspect of this project is the use of the Time Control System. In this section several databases are reviewed for their suitability to this project.

3.5.2 Software Requirements
RAM : 8 GB
Processor : Intel i5 Processor
IDE : Spyder Coding
Language : Python Version 3.8
Operating System : Windows 10

3.5.3 Hardware Requirements
Speed : 1.1 GHz
Hard Disk : 400GB
Key Board : Standard Windows Keyboard

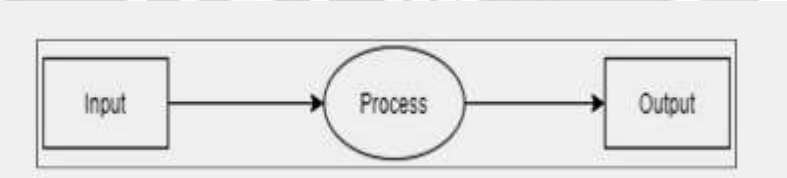
5. SYSTEM DESIGN

Figure 5.1: system Architecture



5.2 DATA FLOW DIAGRAM

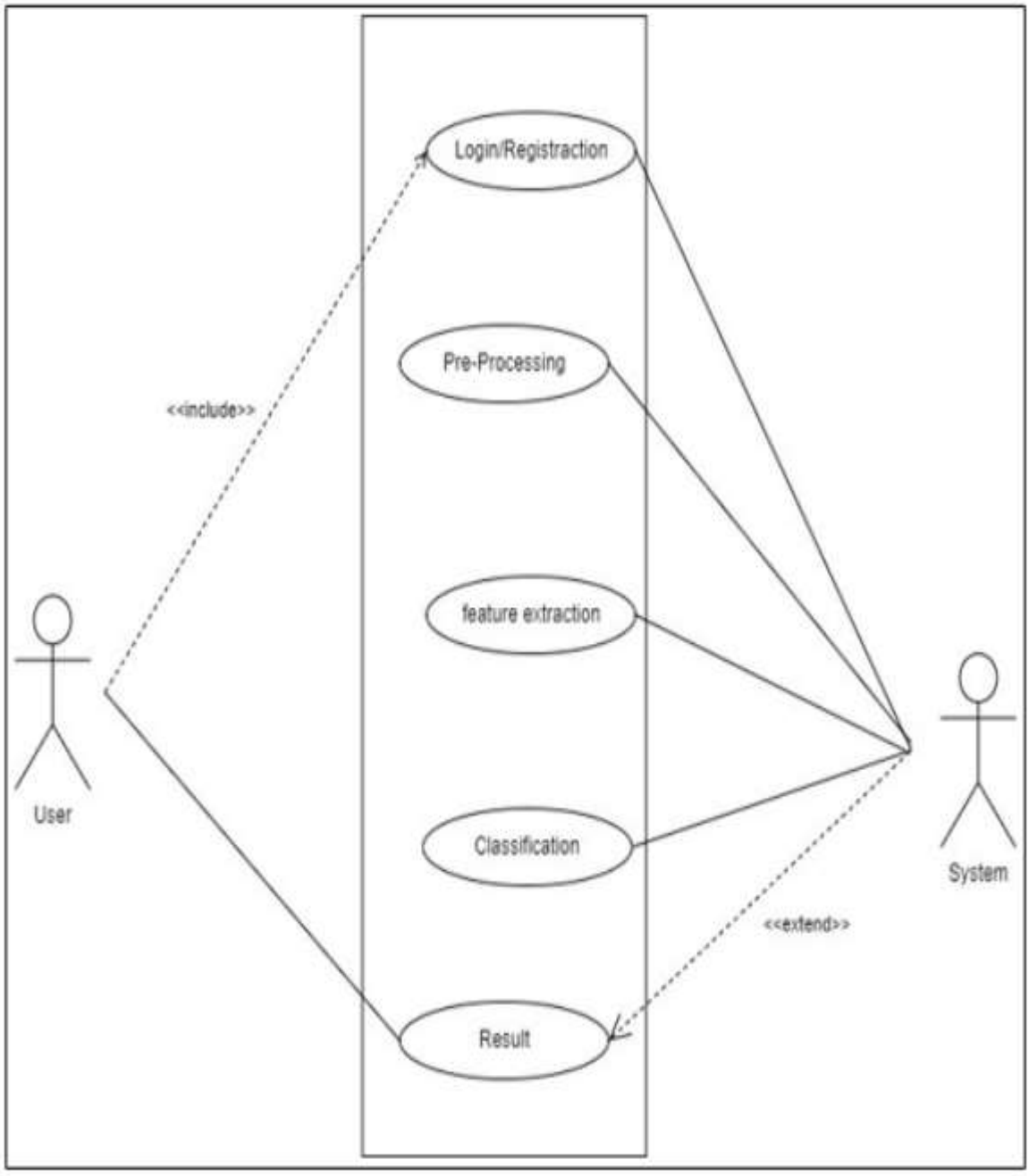
In Data Flow Diagram, we Show that flow of data in our system in DFD0 we show that base DFD in which rectangle present input as well as output and circle show our system, In DFD1 we show actual input and actual output of system input of our system is text or image and output is rumor detected like wise in DFD 2 we present operation of user as well as admin.



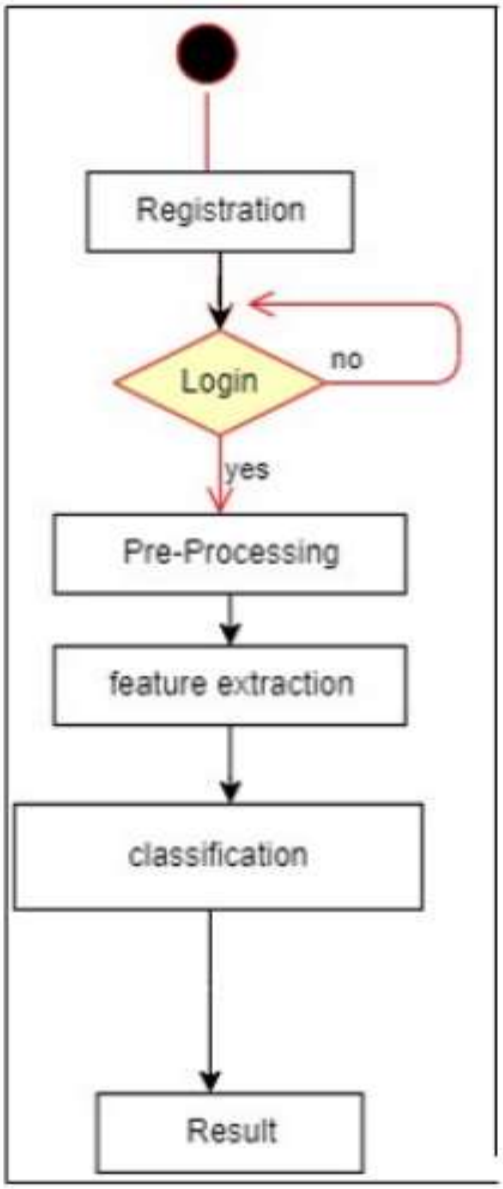
5.3 UML DIAGRAMS

Unified Modeling Language is a standard language for writing software blue prints. The UML may be used to visualize, specify, construct and document the artifacts of a software intensive system. UML is process independent, although optimally it should be used in process that is use case driven, architecture-centric, iterative, and incremental. The Number of UML Diagram is available.

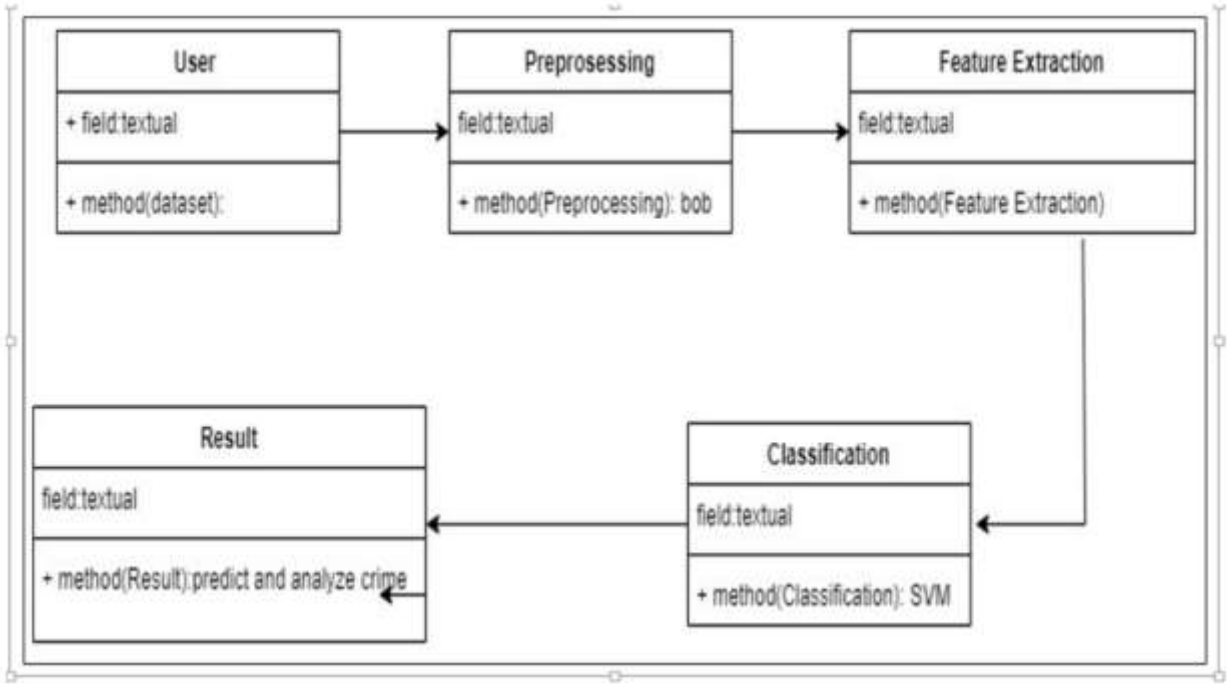
UML Diagram-



Activity Diagram-


Registration
Login
no
yes
Pre-Processing
feature extraction
classification
Result

Class Diagram-



6.1 CONCLUSION

A systematic approach to identifying crime is crime analysis and prediction. This system can predict and visualise crime-prone areas by predicting regions with a high probability of crime occurrence. We can extract previously unknown, useful information from unstructured data using the concept of data mining.

6.2 FUTURE SCOPE

Our future research direction will focus on:
1) Training the model on a larger data set from the deep web to perform a comparative analysis of information embedded in the surface web (Twitter) and darknet forums using SVM .
2) Collecting and annotating more data from surface web deep web which will be made publicly available.

7.1 REFERENCES

1. Shiju Sathyadevan M.S, Surya Gangadharan: "Crime Analysis and Prediction Using Data Mining," in NetworksSoft Computing(ICNSC), (2014) First International Conference.
2. H. Benjamin Fredrick David1, A. Suruliandi: "Survey on crime analysis and prediction using data mining techniques," Department of Computer Science and Engineering, Manonmaniam Sundaranar University, India. Ictact journal on soft computing, April (2017).
3. Jesia Quader Yuki, Md. Mahfil Quader Sakib, Zaisha Zamal, Khan Mohammad Habibullah, Amit Kumar Das: "Predicting Crime Using Time and Location Data" (2019).
4. Peng Chen, Justin Kurland: "Modus Operandi: Time, Place, A Simple Apriori Algorithm Experiment for Crime Pattern Detection" (2018). 9th International Conference on IISA.