# Empirical Analysis for Crime Prediction and Forecasting Using Machine Learning and Deep Learning Techniques

**A PROJECT REPORT**

***Submitted by***

**ASHWIN KUMAR M**

***In partial fulfillment for the award of the degree***

***Of***

**BACHELOR OF TECHNOLOGY**

**IN**

INFORMATION TECHNOLOGY

**KINGSTON ENGINEERING COLLEGE, VELLORE**

**ANNA UNIVERSITY: CHENNAI 600 025**

MAY 2023

## BONAFIDE CERTIFICATE

Certified that this project report “Empirical Analysis for Crime Prediction and Forecasting Techniques”is the bonafide “ASHWIN KUMAR M(511320205004),KISHOREKUMAR J(511320205017),RAMPRASAD R(511320205020),SHAMMIKUMAR T(511320205025)”who carried out the project work under my supervision during the academic year 2022-2023.

| | |
|---|---|
| ……………………… | ……………………… |
| **SIGNATURE** | **SIGNATURE** |
| **Mrs. M. MENAKA, M.E.,(Ph.D)** | **Mrs. S.SARAH.M.,** |
| **HEAD OF THE DEPARTMENT** | **ASSISTANT PROFESSOR** |
| **DEPARTMENT OF IT,** | **DEPARTMENT OF IT,** |
| **KINGSTON ENGINEERING COLLEGE,** | **KINGSTONENGINEERINGCOLLEGE** |
| **CHITOOR MAIN ROAD,** | **CHITOOR MAIN ROAD,** |
| **VELLORE-632059** | **VELLORE-632059** |

# ACKNOWLEDGEMENT

My sincere thanks and performed sense of gratitude goes to the respected Chairman **Mr.D.M.KathirAnand, MBA (USA)** for all his effort in educating me in a premier institution.

We like to express our gratitude to our Principal **U.V.ARIVAZHAGU**

We extend our heartfelt and thanks to the Head of the Department, Mrs. **M.Menaka, M.E., (Ph.D), Department of Information Technology** for her guidance and advice all through the project and motivated us throughout the project.

We convey our sincere and in-depth gratitude to our internal guide **Ms. .,**for her valuable guidance throughout the duration of this project.

We would also like to thank all our faculty members, friends and my batch member for the support they extended during the course of this project.

We are personally indebted to a number of persons that a complete acknowledgement would be encyclopedic. We love to record our deepest gratitude to the Almighty Lord and our family.

We thank our family members and our guide for being a part throughout the project and motivated us to complete the project successfully and God who made us to be healthy and made us to complete without any difficulties on the way and this brought us the great success in this project.

# ABSTRACT

Crime and violation are the threat to justice and meant to be controlled. Accurate crime prediction and future forecasting trends can assist to enhance metropolitan safety computationally. The limited ability of humans to process complex information from big data hinders the early and accurate prediction and forecasting of crime. The accurate estimation of the crime rate, types and hot spots from past patterns creates many computational challenges and opportunities. Despite considerable research efforts, yet there is a need to have a better predictive algorithm, which direct police patrols toward criminal activities. Previous studies are lacking to achieve crime forecasting and prediction accuracy based on learning models. Therefore, this study applied different machine learning algorithms, namely, the logistic regression, support vector machine (SVM), Naïve Bayes, k-nearest neighbors (KNN), decision tree, multilayer perceptron (MLP), random forest, and eXtreme Gradient Boosting (XGBoost), and time series analysis by long-short term memory (LSTM) and autoregressive integrated moving average (ARIMA) model to better fit the crime data. The performance of LSTM for time series analysis was reasonably adequate in order of magnitude of root mean square error (RMSE) and mean absolute error (MAE), on both data sets. Exploratory data analysis predicts more than 35 crime types and suggests a yearly decline in Chicago crime rate, and a slight increase in Los Angeles crime rate; with fewer crimes occurred in February as compared to other months. The overall crime rate in Chicago will continue to increase moderately in the future, with a probable decline in future years. The Los Angeles crime rate and crimes sharply declined, as suggested by the ARIMA model. Moreover, crime forecasting results were further identified in the main regions for both cities. Overall, these results provide early identification of crime, hot spots with higher crime rate, and future trends with improved predictive accuracy than with other methods and are useful for directing police practice and strategies.

## LIST OF ABBREVATIONS

| **ABBREVATION** | **DESCRIPTION** |
|---|---|
| **LSTM** | Long Short Term Memory |
| **KNN** | K-nearest neighbors algorithm |
| **SVM** | Support Vector Machine |
| **RCNN** | Region based Convolutional Neural Network |

| | |
|---|---|
| | |
| | |

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

Criminality is a negative phenomenon, which occurs worldwide in both developed and underdeveloped countries. The criminal activities can severely strike the economy as well as affect the quality of life and well-being of residents, thus leading towards social and societal issues [1]. The crimes and criminal acts can incur costs to both the public and private sectors [2]. Public safety is a considerable factor for secure environments when people travel or move to new places [3]. In reality, different kinds of crimes may be associated with distinct consequences [4]. Overall, crimes take place due to various circumstances including specific motives, human nature and behavior, critical situations and poverty [5]. Furthermore, multiple factors such as unemployment, gender inequality, high population density, child labor, and illiteracy, can cause an increase in violent crimes [6]. The growing and populated cities also have a strong correlation with higher crime rates associated with multiple types of environments such as commercial buildings and municipal housing areas [7]. A socially sustainable community heavily relies on minimizing crime so that people can live peacefully and actively, while corrupt societies cannot prosper both socially and economically in the absence of peace. Consequently, analyzing the crime reports and statistics are essential to improve the safety and security of humanity while maintaining sustainable development.

### 1.1.1AIM OF THE PROJECT

### 1.1.2 SCOPE OF THE PROJECT

## 1.2EXISTING SYSTEM

### 1.2.1 SYSTEM DESCRIPTION

- Some type of news such as various bad events from natural phenomenal or climate are unpredictable. When the unexpected events happen, there are also fake news that are broadcasted that creates confusion due to the nature of the events
- Very few people know the real fact of the event while the most people believe the forwarded news from their credible friends or relatives. These are difficult to detect whether to believe or not when they receive the news information
- Crime prediction and criminal identification are the major problems to the police department as there are tremendous amount of crime data that exist. There is a need of technology through which the case solving could be faster

### 1.2.2 DRAWBACK

- Criminal activities can severely strike the economy as well as affect the quality of life and well-being of residents.

- Time consuming
- Low accuracy
- The overall system will not expect the time wherein the crime is occurring

## PROPOSED SYSTEM

Here it provides the result as crime rate in some specific location along with the red zones and highly occurring crimes in that area. There is an additional feature which removes the fake crime reported by machine learning

This work helps the law enforcement agencies to predict and detect crimes in improved accuracy and thus reduces the crime rat

• Creating a website that helps the police department to analyse and predict the crime rate of a particular area using Machine Learning technique.

• Help people travelling to different place to understand the crime trends of a particular area.

• Enable the common people to file a complaint through online portal as well as track their complaint.

### 1.3.1 SYSTEM DESCRIPTION

### 1.3.2 ADVANTAGES

- The objective would be to train a model for prediction. The training would be done using the training data set which will be validated using the test dataset.

Building the model will be done using better algorithm depending upon the accuracy to predict the crime rate

- Visualization of dataset is done to analyse the crimes which may have occurred in the country. This work helps the law enforcement agencies to predict and detect crimes through machine learning and data science can make the work easier and faster.
- The aim of this project is to make crime prediction using the features present in the dataset. The dataset is extracted from the official sites. With the help of machine learning algorithm, using python as core we can predict the highest of crime action which will occur in a particular area or district.
- We had a high accuracy in this model prediction methodology. In this algorithm for a data mining approach to help predict the crimes patterns and fast up the process of solving crime.
- The results are perfect and accurate using this technology

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 TITLE Deep Learning for Real-Time Crime Forecasting and Its Ternarization

AUTHOR: **:** Bao Wang, Penghang Yin

YEAR : 2019

DESCRIPTION: Real-time crime forecasting is important. However, accurate prediction of when and where the next crime will happen is difficult. No known physical model provides a reasonable approximation to such a complex system. Historical crime data are sparse in both space and time and the signal of interests is weak. In this work, the authors first present a proper representation of crime data. The authors then adapt the spatial temporal residual network on the well represented data to predict the distribution of crime in Los Angeles at the scale of hours in neighborhood-sized parcels. These experiments as well as comparisons with several existing approaches to prediction demonstrate the superiority of the proposed model in terms of accuracy. Finally, the authors present a ternarization technique to address the resource consumption issue for its deployment in real world. This work is an extension of our short conference proceeding paper [Wang, B., Zhang, D., Zhang, D. H., et al., Deep learning for real time Crime forecasting, 2017, arXiv: 1707.03340].

## 2.2 TITLE : Comparison of Machine Learning Algorithms for Predicting Crime Hotspots

AUTHOR: Xu Zhang, Lin Liu

YEAR : 2020

DESCRIPTION: Crime prediction is of great significance to the formulation of policing strategies and the implementation of crime prevention and control. Machine learning is the current mainstream prediction method. However, few studies have systematically compared different machine learning methods for crime prediction.

This paper takes the historical data of public property crime from 2015 to 2018 from a section of a large coastal city in the southeast of China as research data to assess the predictive power between several machine learning algorithms. Results based on the historical crime data alone suggest that the LSTM model outperformed KNN, random forest, support vector machine, naive Bayes, and convolutional neural networks. In addition, the built environment data of points of interests (POIs) and urban road network density are input into LSTM model as covariates. It is found that the model with built environment covariates has better prediction effect compared with the original model that is based on historical crime data alone. Therefore, future crime prediction should take advantage of both historical crime data and covariates associated with criminological theories. Not all machine learning algorithms are equally effective in crime prediction.

## 2.3 TITLE : Suggesting a Hybrid Approach: Mobile Apps with Big Data Analysis to Report and Prevent Crimes

AUTHOR: Abdi Fidow, Ahmed Hassan

YEAR : 2019

DESCRIPTION: Conventional crime prediction techniques rely on location-specific historical crime data. Yet relying on historical crime data alone has deficiencies, as such data is limited in scope and often fails to capture the full complexity of crimes. This chapter proposes a novel approach to employ mobile applications with big data analysis for crime reporting and prevention using aggregate data from multiple sources, the Hybrid Smart Crime Reporting App (HIVICRA). It is an infographic intelligent crime-reporting analysis application that incorporates crime data sourced from local police, social media and crowdsourcing, including sentiment analysis of Twitter streams in conjunction with historical police crime datasets. An evaluation of the approach suggests that by combining sentiment analysis with smart crime reporting applications, it is possible to improve the forecasting of crime.

## 2.4 TITLE : Spatiotemporal Patterns and Driving Factors on Crime Changing During Black Lives Matter Protests

AUTHOR: Zhiran Zhang [1,2,3], Dexuan Sha [2,4]

YEAR : 2020

DESCRIPTION: The death of George Floyd has brought a new wave of 2020 Black Lives Matter (BLM) protests into U.S. cities. Protests happened in a few cities accompanied by reports of violence over the first few days. The protests appear to be related to rising crime. This study uses newly collected crime data in 50 U.S. cities/counties to explore the spatiotemporal crime changes under BLM protests and to estimate the driving factors of burglary induced by the BLM protest. Four spatial and statistic models were used, including the Average Nearest Neighbor (ANN), Hotspot Analysis, Least Absolute Shrinkage, and Selection Operator (LASSO), and Binary Logistic Regression. The results show that (1) crime, especially burglary, has risen sharply in a few cities/counties, yet heterogeneity exists across cities/counties; (2) the volume and spatial distribution of certain crime types changed under BLM protest, the activity of burglary clustered in certain regions during protests period; (3) education, race, demographic, and crime rate in 2019 are related with burglary changes during BLM protests. The findings from this study can provide valuable information for ensuring the capabilities of the police and governmental agencies to deal with the evolving crisis.

## 2.5 TITLE : LASSO-based feature selection and naïve Bayes classifier for crime prediction and its type

AUTHOR: Gnaneswara Rao Nitta, B. Yogeshwara Rao

YEAR : 2019

DESCRIPTION: For centuries, crime has been viewed as random because it is based on human behavior; even now, it incorporates an excessive number of factors for current machine learning models to forecast accurately. In this work, we tend to discuss the early crime prediction results from a model developed using the data from the Chicago crime dataset. In any case, with a superior execution future crime is to anticipated accurately, it is a testing assignment as a result of the increase in several crimes in present days. Therefore, the crime foreseeing method is foremost, and it identifies the future crimes and number of crimes are degraded. In this paper, we built up a model to anticipate future crime occurrences at a future time and also predict which type of crime may be happening in a given area. First, we analyze how certain crime features like given a date, time and some geologically important relevant features such as latitude and longitude. Second, we discuss several analytics techniques we used to find meaning in our data, such as LASSO feature selection analysis, classification models like naïve Bayes and SVM. Finally, we select the best model for foreseeing crime type and seriousness of the crime for giving different features.

## 2.6 Title: Spatio-temporal crime predictions in smart cities: A data-driven approach and experiments

**Author:** Charles E. Catlett, Eugenio Cesario

**Year: 2019**

**Description:** Steadily increasing urbanization is causing significant economic and social transformations in urban areas, posing several challenges related to city management and services. In particular, in cities with higher crime rates, effectively providing for public safety is an increasingly complex undertaking. To handle this complexity, new technologies are enabling police departments to access growing volumes of crime-related data that can be analyzed to understand patterns and trends. These technologies have potentially to increase the efficient deployment of police resources within a given territory and ultimately support more effective crime prevention. This paper presents a predictive approach based on spatial analysis and auto-regressive models to automatically detect high-risk crime regions in urban areas and to reliably forecast crime trends in each region. The algorithm result is a spatio-temporal crime forecasting model, composed of a set of crime-dense regions with associated crime predictors, each one representing a predictive model for estimating the number of crimes likely to occur in its associated region. The experimental evaluation was performed on two real-world datasets collected in the cities of Chicago and New York City. This evaluation shows that the proposed approach achieves good accuracy in spatial and temporal crime forecasting over rolling time horizons

Title: Early prediction of circulatory failure in the intensive care unit using machine learning

**Author:**Stephanie L. Hyland, Martin Faltys

**Year: 2020**

**Description:** Intensive-care clinicians are presented with large quantities of measurements from multiple monitoring systems. The limited ability of humans to process complex information hinders early recognition of patient deterioration, and high numbers of monitoring alarms lead to alarm fatigue. We used machine learning to develop an early-warning system that integrates measurements from multiple organ systems using a high-resolution database with 240 patient-years of data. It predicts 90% of circulatory-failure events in the test set, with 82% identified more than 2 h in advance, resulting in an area under the receiver operating characteristic curve of 0.94 and an area under the precision-recall curve of 0.63. On average, the system raises 0.05 alarms per patient and hour. The model was externally validated in an independent patient cohort. Our model provides early identification of patients at risk for circulatory failure with a much lower false-alarm rate than conventional threshold-based systems.

Title: Deep learning for time series classification: a review

**Author:** Hassan Ismail Fawaz, Germain Forestier

**Year: 2019**

**Description:** Time Series Classification (TSC) is an important and challenging problem in data mining. With the increase of time series data availability, hundreds of TSC algorithms have been proposed. Among these methods, only a few have considered Deep Neural Networks (DNNs) to perform this task. This is surprising as deep learning has seen very successful applications in the last years. DNNs have indeed revolutionized the field of computer vision especially with the advent of novel deeper architectures such as Residual and Convolutional Neural Networks. Apart from images, sequential data such as text and audio can also be processed with DNNs to reach state-of-the-art performance for document classification and speech recognition. In this article, we study the current state-of-the-art performance of deep learning algorithms for TSC by presenting an empirical study of the most recent DNN architectures for TSC. We give an overview of the most successful deep learning applications in various time series domains under a unified taxonomy of DNNs for TSC. We also provide an open source deep learning framework to the TSC community where we implemented each of the compared approaches and evaluated them on a univariate TSC benchmark (the UCR/UEA archive) and 12 multivariate time series datasets. By training 8730 deep learning models on 97 time series datasets, we propose the most exhaustive study of DNNs for TSC to date.

Title: Non-Stationary Model for Crime Rate Inference Using Modern Urban Data

**Author:** Hongjian Wang, Huaxiu Yao

**Year: 2019**

**Description:** Crime is one of the most important social problems in the country, affecting public safety, children development, and adult socioeconomic status. Understanding what factors cause higher crime rate is critical for policy makers in their efforts to reduce crime and increase citizens' life quality. We tackle a fundamental problem in our paper: crime rate inference at the neighborhood level. Traditional approaches have used demographics and geographical influences to estimate crime rates in a region. With the fast development of positioning technology and prevalence of mobile devices, a large amount of modern urban data have been collected and such big data can provide new perspectives for understanding crime. In this paper, we use large-scale Point-Of-Interest data and taxi flow data in the city of Chicago, IL in the USA. We observe significantly improved performance in crime rate inference compared to using traditional features. Such an improvement is consistent over multiple years. We also show that these new features are significant in the feature importance analysis. The correlations between crime and various observed features are not constant over the whole city. In order to address this geospatial non-stationary property, we further employ the geographically weighted regression on top of negative binomial model (GWNBR). Experiments have shown that GWNBR outperforms the negative binomial model.

Title: An intelligent hybridization of ARIMA with machine learning models for time series forecasting

**Author:** Domingos S. de O. Santos Júnior [a b], João F.L. de Oliveira [b]

**Year: 2019**

**Description:** The development of accurate forecasting systems can be challenging in real-world applications. The modeling of real-world time series is a particularly difficult task because they are generally composed of linear and nonlinear patterns that are combined in some form. Several hybrid systems that combine linear and nonlinear techniques have obtained relevant results in terms of accuracy in comparison with single models. However, the best combination function of the forecasting of the linear and nonlinear patterns is unknown, which makes this modeling an open question. This work proposes a hybrid system that searches for a suitable function to combine the forecasts of linear and nonlinear models. Thus, the proposed system performs: (i) linear modeling of the time series; (ii) nonlinear modeling of the error series; and (iii) a data-driven combination that searches for: (iii.a) the most suitable function, between linear and nonlinear formalisms, and (iii.b) the number of forecasts of models (i) and (ii) that maximizes the performance of the combination. Two versions of the hybrid system are evaluated. In both versions, the ARIMA model is used in step (i) and two nonlinear intelligent models – Multi-Layer Perceptron (MLP) and Support Vector Regression (SVR) – are used in steps (ii) and (iii), alternately. Experimental simulations with six real-world complex time series that are well-known in the literature are evaluated using a set of popular performance metrics. Our results show that the proposed hybrid system attains superior performance when compared with single and hybrid models in the literature.

.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 HARDWARE REQUIREMENTS:

|  |  |  |
|---|---|---|
| ❖ **System** | **:** | Pentium IV 2.4 GHz. |
| ❖ **Hard Disk** | **:** | 40 GB. |
| ❖ **Floppy Drive** | **:** | 1.44 Mb. |
| ❖ **Monitor** | : | 14’ Colour Monitor. |
| ❖ **Mouse** | **:** | Optical Mouse. |
| ❖ **Ram** | **:** | 512 Mb. |

## 3.2 SOFTWARE REQUIREMENTS:

|  |  |  |
|---|---|---|
| ❖ **Operating system** | **:** | Windows 7 Ultimate. |

- **Coding Language** : Python.
- **Front-End** : HTML, CSS.

## 3.3METHODOLOGY

## 5. MODULES

The implementation of this project can be done in the following steps:

**Data Collection:**

Crime dataset from kaggle is used in CSV format.

**Data Pre-processing:**

This is the first step in Machine Learning. We need to clean the data and remove the unnecessary noises, null data and missing values.

**Feature Selection:**

The attributes used for feature selection are Block, Location, District, Community area, X coordinate, Y Coordinate, Latitude, Longitude, Hour and month.

**Building and Training Model:**

The dataset is divided into pair of xtrain, ytrainand xtest, ytest.

**Prediction:**

After the model is build using the above process, prediction is done using model. Predict (xtest).The accuracy is calculated using accuracy _score imported from metrics –metrics. Accuracy_ score (ytest, predicted).

**Visualization:**

To analyse the Crime dataset in the Graphical Representation such as pie chart and bar graph.

More than the written results the pictorial view would be easier to understand and analyse.

## 6. ALGORITHM

### Deep Convolutional Neural Network

### What are Deep Convolutional Neural Networks?

Deep learning is a machine learning technique used to build artificial intelligence (AI) systems. It is based on the idea of artificial neural networks (ANN), designed to perform complex analysis of large amounts of data by passing it through multiple layers of neurons.

There is a wide variety of deep neural networks (DNN). Deep convolutional neural networks (CNN or DCNN) are the type most commonly used to identify patterns in images and video. DCNNs have evolved from traditional artificial neural networks, using a three-dimensional neural pattern inspired by the visual cortex of animals.

Deep convolutional neural networks are mainly focused on applications like object detection, image classification, recommendation systems, and are also sometimes used for natural language processing.

### Architecture of the deep CNN algorithm

CNN is a type of machine learning that is used in various fields, especially in image and sound recognition. Deep CNNs imitate the connectivity patterns of neurons in the animal visual cortex. CNNs consist of 1 or more convolutional layer, a pooling layer, and a fully connected layer. Every convolutional layer responds to stimuli only in a restricted region of the visual field known as the receptive field. This structure is distinguished from conventional image classification algorithms and other deep learning algorithms, since CNN can learn the type of filter that is hand-crafted in conventional algorithms.

This study used 16 convolutional layers and 3 fully connected dense layers; this network is illustrated in Figure 1. Each convolutional layer was designed with a kernel size of 3×3 pixels, the same padding, and a rectified linear unit activation function. The maximum pooling layers were designed with strides of 2×2 pixels. After extracting the feature quantities of images using convolutional layers, we used the maximum pooling layers to reduce the

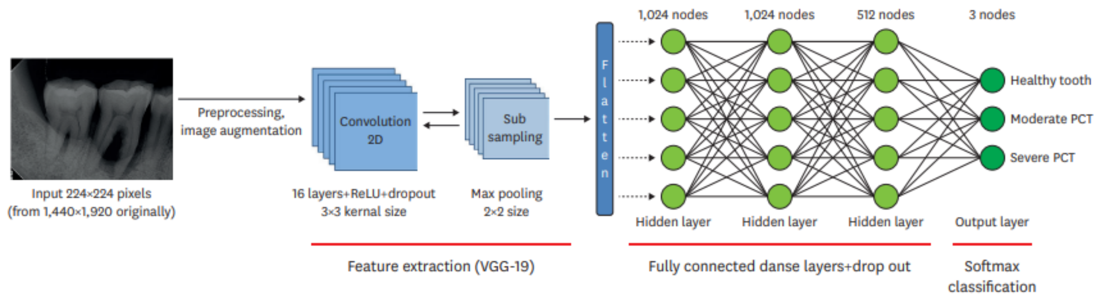


Figure 1. Overall architecture of the deep CNN model. The dataset for the PCT images (224×224 pixels) is labeled as the input. Each of the convolutional layers is followed by a ReLU activation function, dropout, maximum pooling layers, and 3 fully connected layers with 1,024, 1,024, and 512 nodes, respectively. The final output layer performs 3 classifications using the Softmax function. CNN: convolutional neural network, PCT: periodontally compromised tooth, ReLU: rectified linear unit.

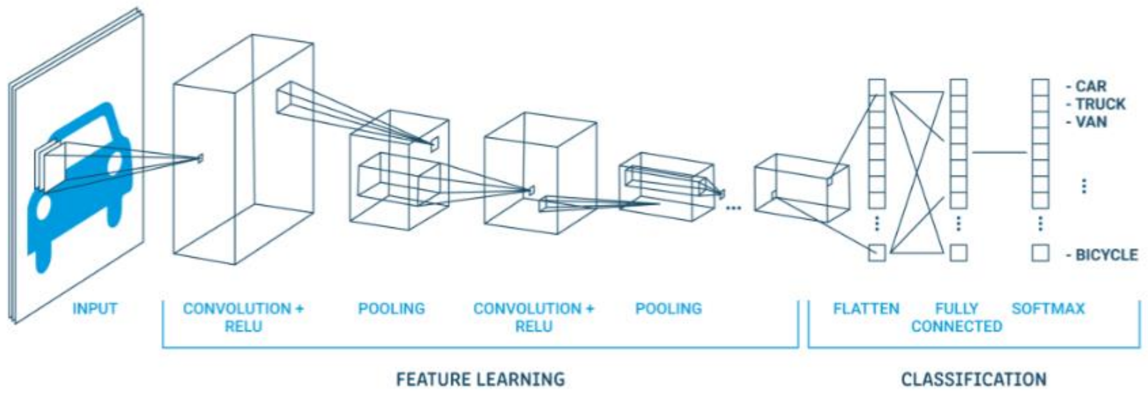
position sensitivity problem and to allow for more generic recognition capability. Next, 3 fully connected deep hidden layers with 1,024, 1,024, and 512 nodes, respectively, were connected to remove spatial information and to statistically determine the key classification of PCT [21]. Dropout, which is a typical method of regularization (rescaling the deep CNN weights to a more effective range), was set to 0.5, and the final output layer was classified in terms of PCT using the Softmax classifier [22]. A total of 500 epochs were used and training network weights were learned using the Adam algorithm (learning rate=0.0001), a stochastic gradient descent method [23]. To produce better diagnosis and prediction of PCT, after 20 epochs of this training phase, fine-tuning was performed in order to optimize the weights and to improve the results by adjusting the hyperparameters of layers [24,25].

## Deep Convolutional Neural Networks Explained

The strength of DCNNs is in their layering. A DCNN uses a three-dimensional neural network to process the Red, Green, and Blue elements of the image at the same time. This considerably reduces the number of artificial neurons required to process an image, compared to traditional feed forward neural networks.

Deep convolutional neural networks receive images as an input and use them to train a classifier. The network employs a special mathematical operation called a "convolution" instead of matrix multiplication.

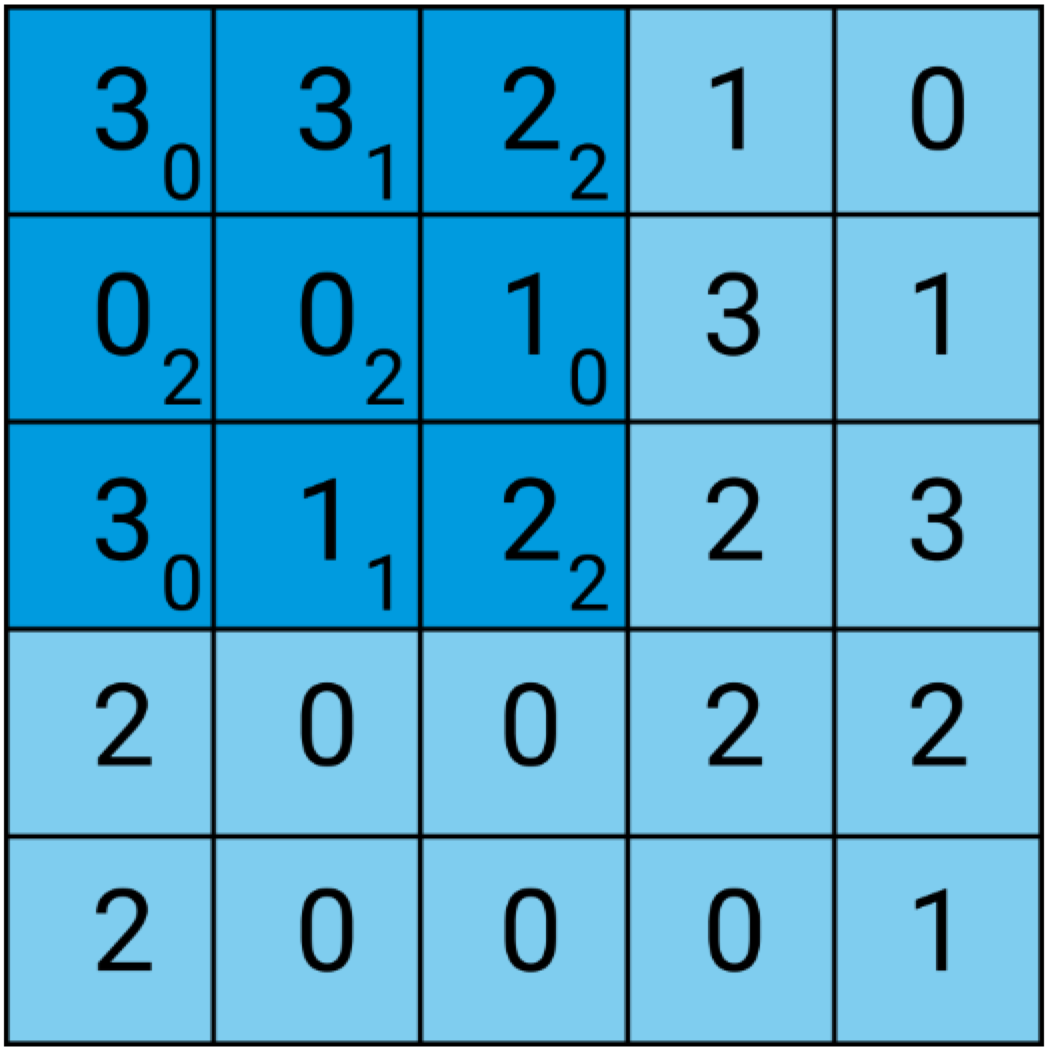
The architecture of a convolutional network typically consists of four types of layers: convolution, pooling, activation, and fully connected.

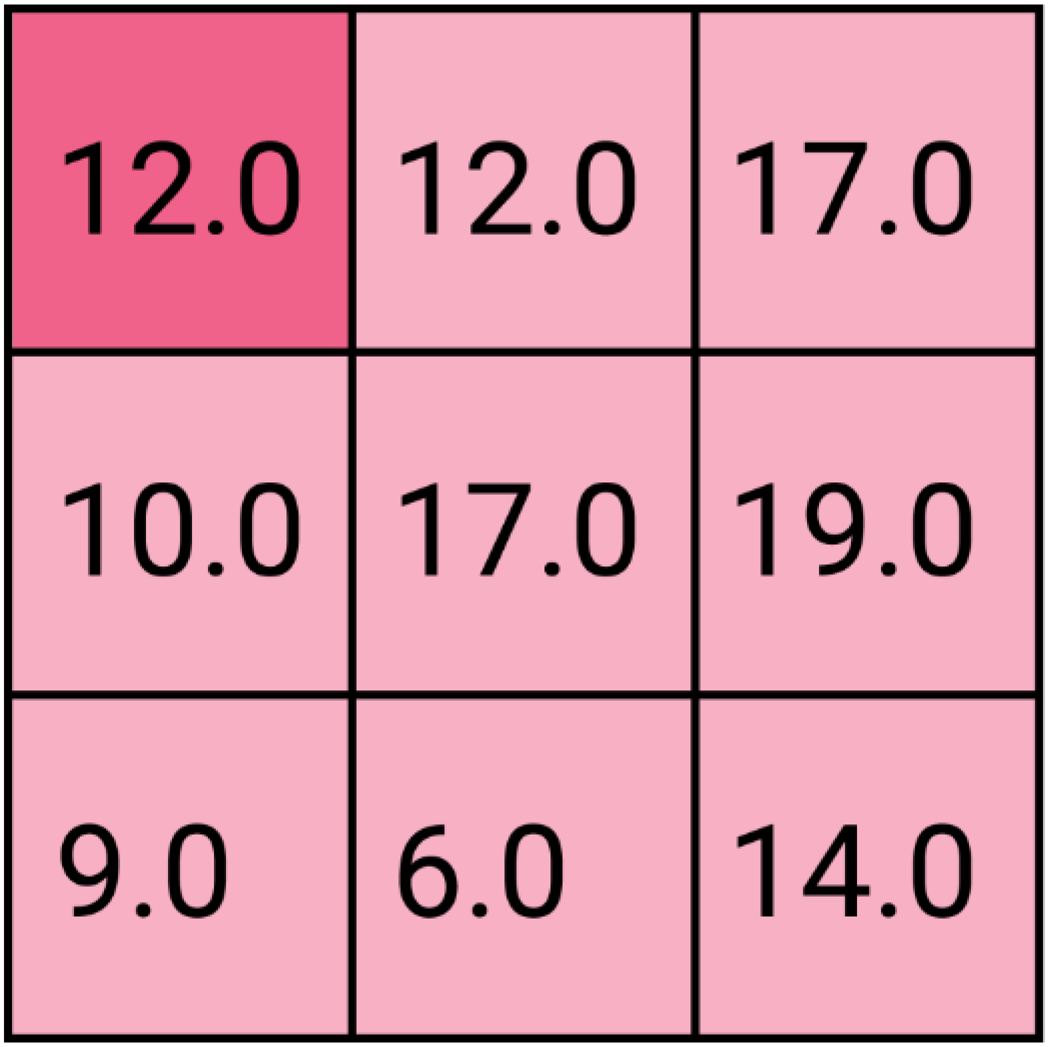


## Convolutional Layer

Applies a convolution filter to the image to detect features of the image. Here is how this process works:

- **A convolution**—takes a set of weights and multiplies them with inputs from the neural network.
- **Kernels or filters**—during the multiplication process, a kernel (applied for 2D arrays of weights) or a filter (applied for 3D structures) passes over an image multiple times. To cover the entire image, the filter is applied from right to left and from top to bottom.
- **Dot or scalar product**—a mathematical process performed during the convolution. Each filter multiplies the weights with different input values. The total inputs are summed, providing a unique value for each filter position.


$3_0$ $3_1$ $2_2$ 1 0
$0_2$ $0_2$ $1_0$ 3 1
$3_0$ $1_1$ $2_2$ 2 3
2 0 0 2 2
2 0 0 0 1



| | | |
|---|---|---|
| 12.0 | 12.0 | 17.0 |
| 10.0 | 17.0 | 19.0 |
| 9.0 | 6.0 | 14.0 |

**ReLU Activation Layer**

The convolution maps are passed through a nonlinear activation layer, such as Rectified Linear Unit (ReLu), which replaces negative numbers of the filtered images with zeros.

**Pooling Layer**

The pooling layers gradually reduce the size of the image, keeping only the most important information. For example, for each group of 4 pixels, the pixel having the maximum value is retained (this is called max pooling), or only the average is retained (average pooling).

Pooling layers help control overfitting by reducing the number of calculations and parameters in the network.

After several iterations of convolution and pooling layers (in some deep convolutional neural network architectures this may happen thousands of times), at the end of the network there is a traditional multi-layer perceptron or “fully connected” neural network.

**Fully Connected Layer**

In many CNN architectures, there are multiple fully connected layers, with activation and pooling layers in between them. Fully connected layers receive an input vector containing the flattened pixels of the image, which have been filtered, corrected and reduced by convolution and pooling layers. The softmax function is applied at the end to the outputs of the fully connected layers, giving the probability of a class the image belongs to – for example, is it a car, a boat or an airplane.

**Related content: read our guide to deep learning for computer vision.**

What are the Types of Deep Convolutional Neural Networks?

Below are five deep convolutional neural network architectures commonly used to perform object detection and image classification?

**R-CNN**

Region-based Convolutional Neural Network (R-CNN), is a network capable of accurately extracting objects to be identified in the image. However, it is very slow in the scanning phase and in the identification of regions.

The poor performance of this architecture is due to its use of the selective search algorithm, which extracts approximately 2000 regions of the starting image.

Afterwards it executes N CNNs on top of each region, whose outputs are fed to a support vector machine (SVM) to classify the region.

**Fast R-CNN**

Fast R-CNN is a simplified R-CNN architecture, which can also identify regions of interest in an image but runs a lot faster. It improves performance by extracting features before it identifies regions of interest. It uses only one CNN for the entire image, instead of 2000 CNN networks on each superimposed region. Instead of the SVM which is computationally intensive, a softmax function returns the identification probability. The downside is that Fast R-CNN has lower accuracy than R-CNN in terms recognition of the bounding boxes of objects in the image.

**GoogleNet (2014)**

GoogleNet, also called Inception v1, is a large-scale CNN architecture which won the ImageNet Challenge in 2014. It achieved an error rate of less than 7%, close to the level of human performance. The architecture consists of a 22-layer deep CNN based on small convolutions, called "inceptions", batch normalization, and other techniques to decrease the number of parameters from tens of millions in previous architectures to four million.

**VGGNet (2014)**

A deep convolutional neural network architecture with 16 convolutional layers. It uses 3x3 convolutions, and trained on 4 GPUs for more than two weeks to achieve its performance. The downside of VGGNet is that unlike GoogleNet, it has 138 million parameters, making it difficult to run in the inference stage.

**ResNet (2015)**

The Residual Neural Network (ResNet) is a CNN with up to 152 layers. ResNet uses "gated units", to skip some convolutional layers. Like GoogleNet, it uses heavy batch normalization. ResNet uses an innovative design which lets it run many more convolutional layers without increasing complexity. It participated in the ImageNet Challenge 2015, achieving an impressive error rate of 3.57%, while beating human-level performance on the trained dataset.

Business Applications of Convolutional Neural Networks

**Image Classification**

Deep convolutional neural networks are the state of the art mechanism for classifying images. For example, they are used to:

- **Tag images**—an image tag is a word or combination of words that describes an image and makes it easier to find. Google, Facebook and Amazon use this technology. Labeling includes identifying objects and even analyzing the sentiment of the image.

- **Visual search**—matching the input image with an available database. Visual search analyzes the image and searches for an existing image with the identified information. For example, Google search uses this technique to find different sizes or colors of the same product.

- **Recommendation engines**—using CNN image recognition to provide product recommendations, for example in websites like Amazon. The engine analyzes user preferences and returns products whose images match previous products they viewed or bought, for example, a red dress or red shoes with red lipstick.

**Medical Image Analysis**

CNN classification on medical images is more accurate than the human eye and can detect abnormalities in X-ray or MRI images. Such systems can analyze sequences of images (for examples, tests taken over a long period of time) and identify subtle differences that human analysts might miss. This also makes it possible to perform predictive analysis.

Classification models for medical images are trained on large public health databases. The resulting models can be used on patient test results, to identify medical conditions and automatically generate a prognosis.

**Optical Character Recognition**

Optical character recognition (OCR) is used to identify symbols such as text or numbers in images. Traditionally OCR was performed using statistical or early machine learning techniques, but today many OCR engines use deep convolutional neural networks.

OCR powered by CNNs can be used to improve search within rich media content, and identify text in written documents, even those with poor quality or hard to recognize handwriting. This is especially important in the banking and insurance industries. Another application of deep learning OCR is for automated signature recognition.

Deep Convolutional Neural Networks with Run:AI

Run:AI automates resource management and orchestration for machine learning infrastructure. With Run:AI, you can automatically run as many compute intensive experiments as needed.

Here are some of the capabilities you gain when using Run:AI:

- **Advanced visibility**—create an efficient pipeline of resource sharing by pooling GPU compute resources.
- **No more bottlenecks**—you can set up guaranteed quotas of GPU resources, to avoid bottlenecks and optimize billing.
- **A higher level of control**—Run:AI enables you to dynamically change resource allocation, ensuring each job gets the resources it needs at any given time.

Run:AI simplifies machine learning infrastructure pipelines, helping data scientists accelerate their productivity and the quality of their models.

Learn more about the Run:AI GPU virtualization platform.

### DCNN Advantage

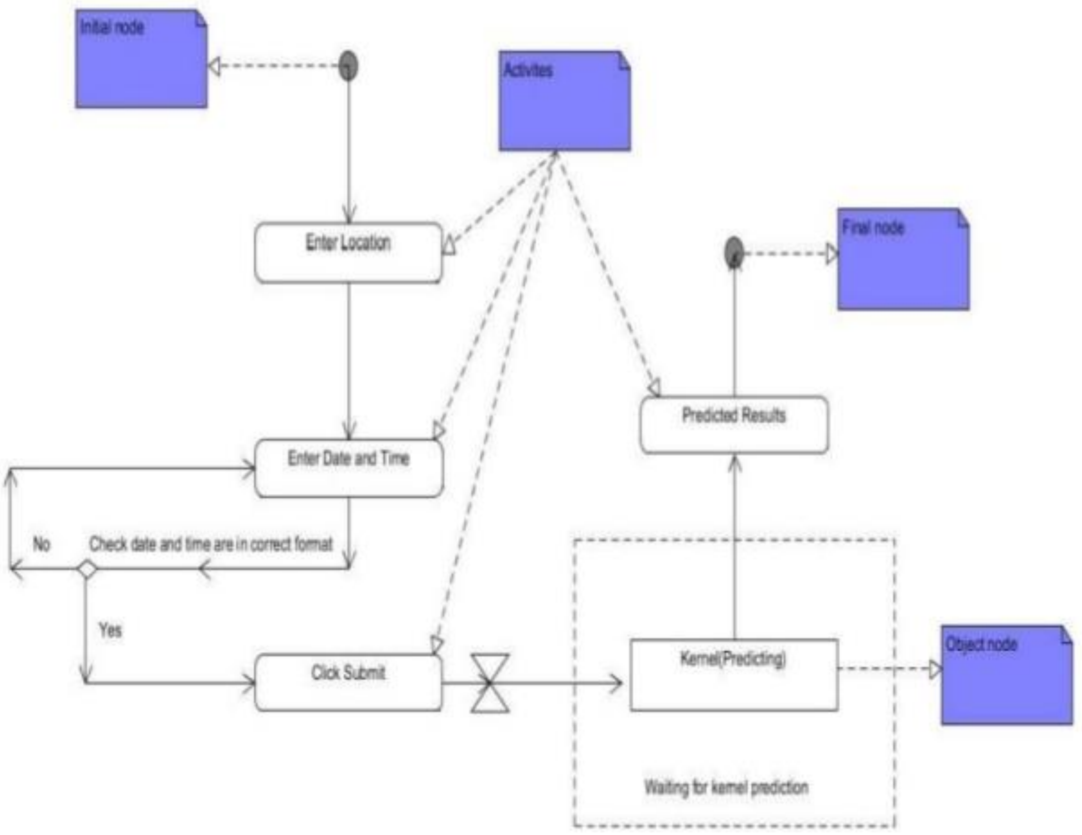
The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself. CNN is also computationally efficient.

### DCNN Disadvantage

Some of the disadvantages of CNNs: include the fact that a lot of training data is needed for the CNN to be effective and that they fail to encode the position and orientation of objects. They fail to encode the position and orientation of objects. They have a hard time classifying images with different positions.

## 7. SYSTEM DESIGN

## 7.1 Architecture Diagram



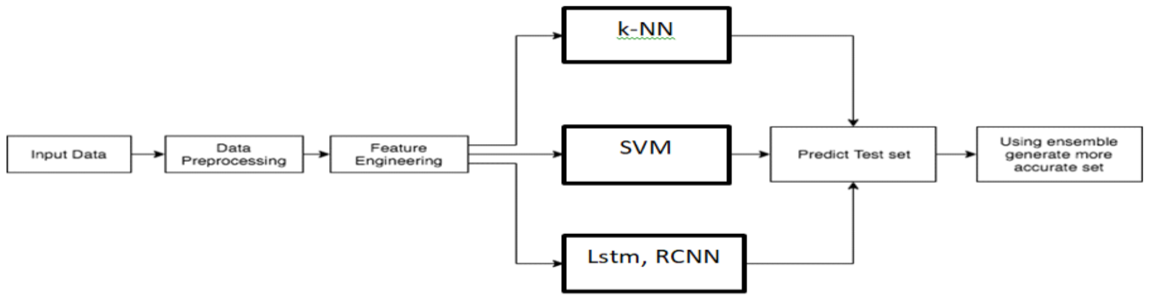


Fig 1: Architecture Diagram

## UML diagrams

# 8. SYSTEM TESTING

## PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming

Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., `x = 2`, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address.

The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, `x = 2`, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., `x = 2`; `y = 2`; `z = 2` result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, Python has array index and array slicing expressions on lists, denoted as

a[key],

a[start:stop]

or

a[start:stop:step].

Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

**List comprehensions vs. for-loops**

**Conditional expressions vs. if blocks**

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ...causes a syntax error in Python.

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

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.

- **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.

**Python's features include −**

- **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 ofUnix.

**Scalable** − Python provides a better structure and support for large programs than shellscripting. 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 asOOP.

 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 typechecking.

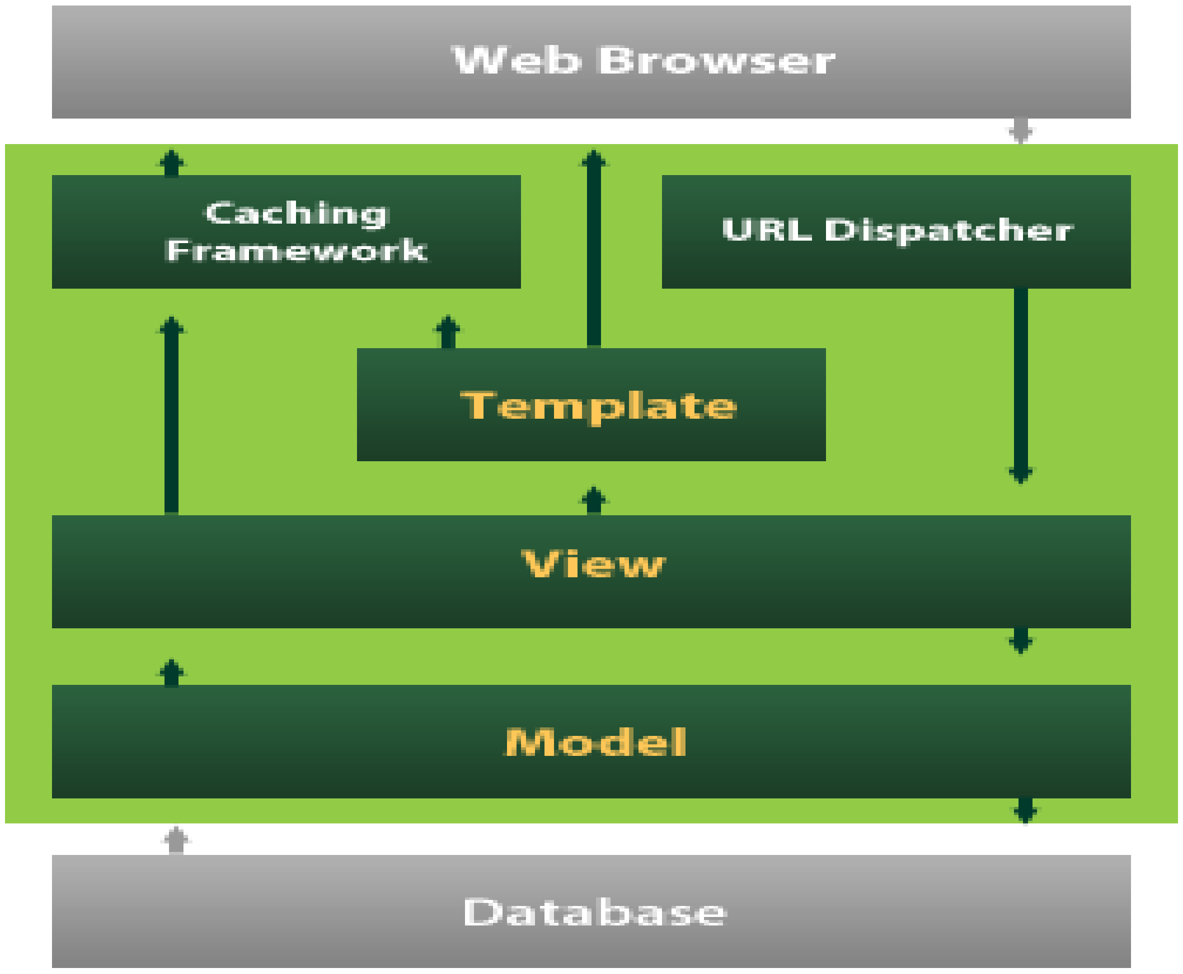
 It supports automatic garbagecollection.

 It can be easily integrated with C, C++, COM, ActiveX, CORBA, andJava.

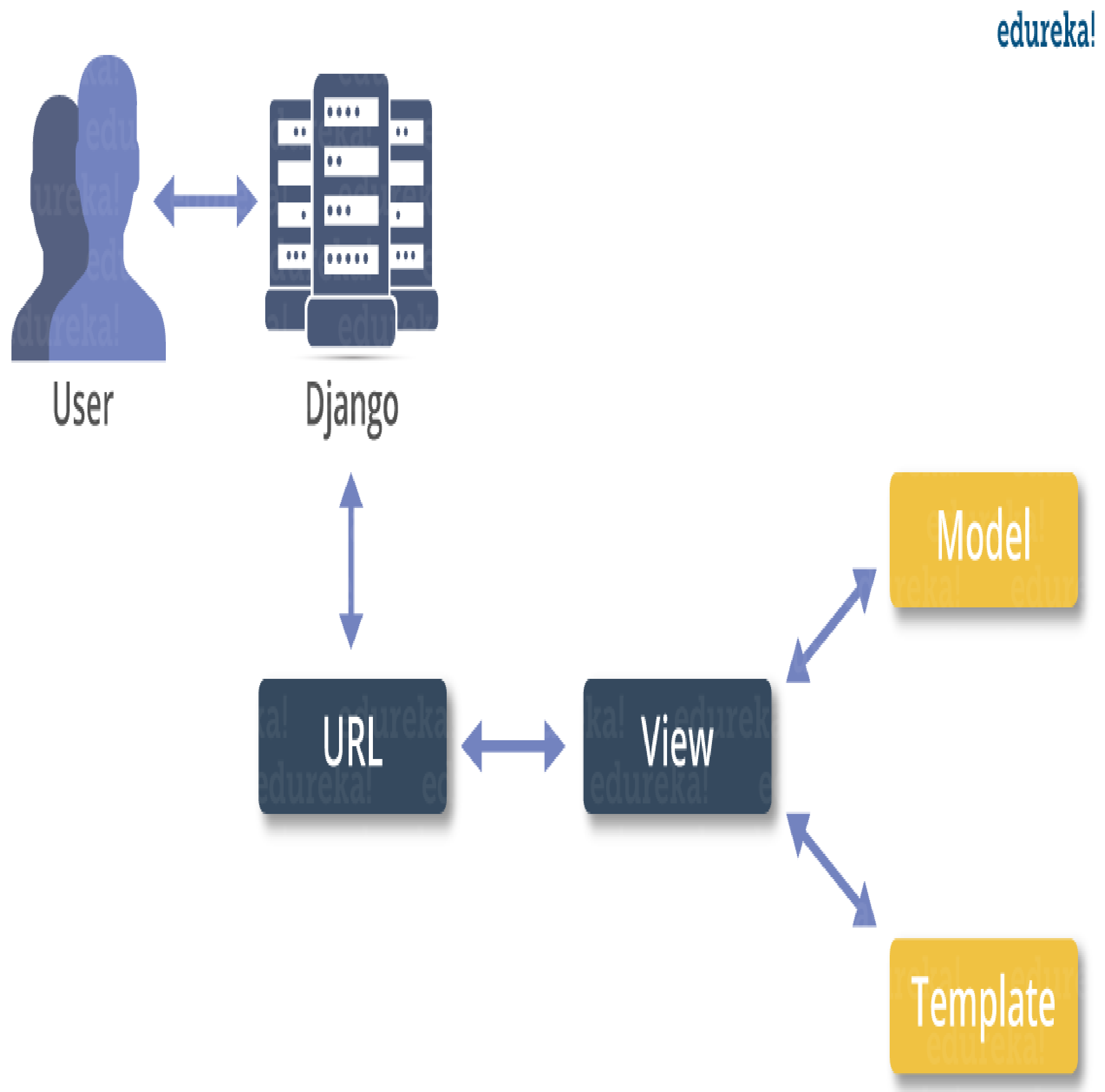
**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



**Python – The New Generation Language**

Python is a widely used general-purpose, high level programming language. It was initially designed by

Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for an emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

**Features**

• Interpreted

In Python there is no separate compilation and execution steps like C/C++. It directly run the program

from the source code. Internally, Python converts the source code into an intermediate form called byte codes which is then translated into native language of specific computer to run it.

• Platform Independent

Python programs can be developed and executed on the multiple operating system platform. Python can

be used on Linux, Windows, Macintosh, Solaris and many more.

• Multi- Paradigm

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspectoriented programming.

• Simple

Python is a very simple language. It is a very easy to learn as it is closer to English language. In python more emphasis is on the solution to the problem rather than the syntax.

• Rich Library Support

Python standard library is very vast. It can help to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML,

WAV files, cryptography, GUI and many more.

• Free and Open Source

Firstly, Python is freely available. Secondly, it is open-source. This means that its source code is available to the public. We can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open Source Software). As the Python community, we’re all headed toward one goal- an ever-bettering Python

**Types of Machine Learning**

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. Broadly Machine Learning can be categorized into four categories .I.

**Supervised Learning**

**Unsupervised Learning**

**Reinforcement Learning IV.**

Semi-supervised Learning Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly

**Supervised Learning**

Supervised Learning is a type of learning in which we are given a data set and we already know what arecorrect output should look like, having the idea that there is a relationship between the input and output.Basically, it is learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.Supervised learning problems are categorized

**Unsupervised Learning**

Unsupervised Learning is a type of learning that allows us to approach problems with little or no idea what our problem should look like. We can derive the structure by clustering the data based on a relationship among the variables in data. With unsupervised learning there is no feedback based on prediction result. Basically, it is a type of self-organized learning that helps in finding previously unknown patterns in data set without pre-existing label.

**Reinforcement Learning**

Reinforcement learning is a learning method that interacts with its environment by producing actions anddiscovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristicsof reinforcement learning. This method allows machines and software agents to automatically determine theidealbehavior within a specific context in order to maximize its performance. Simple reward feedback isrequired for the agent to learn which action is best.

## Semi-Supervised Learning

Semi-supervised learning fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount ofunlabeled data. The systems that use this method are able to considerably improve learning accuracy.Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require

## Objectives

Main objectives of training were to learn:

How to determine and measure program complexity,

- ✓ Python Programming
- ✓ ML Library Scikit, Numpy , Matplotlib, Pandas , Theano , TensorFlow
- ✓ Statistical Math for the Algorithms.
- ✓ Learning to solve statistics and mathematical concepts.
- ✓ Supervised and Unsupervised Learning
- ✓ Classification and Regression

## ML Algorithms

## Machine Learning Programming and Use Case

## Advantages of Machine Learning

Every coin has two faces, each face has its own property and features. It’s time to uncover the faces of ML.

A very powerful tool that holds the potential to revolutionize the way things work.

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation) -

With ML, we don’t need to babysit our project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software. they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement -

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say we need to make a weather forecast model. As the amount of data, we have keeps growing, our algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data -Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications -

We could be an e-seller or a healthcare provider and make ML work for us. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Applications of Machine Learning**

**Applications of Machine Learning include:**

**Web Search Engine**:

One of the reasons why search engines like google, bingetc work so well is because the system has learnt how to rank pages through a complex learning algorithm.

**Photo tagging Applications:**

Be it face book or any other photo tagging application, the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.

**Spam Detector:**

Our mail agent like Gmail or Hotmail does a lot of hard work for us in classifying the mails and moving the spam mails to spam folder. This is again achieved by a spam classifier running in the back end of mail application.

**Database Mining for growth of automation:**

Typical applications include Web-click data for betterUX, Medical records for better automation in healthcare, biological data and many more.

**Applications that cannot be programmed:**

There are some tasks that cannot be programmed as the computers we use are not modelled that way. Examples include Autonomous Driving, Recognition tasks from unordered data (Face Recognition/ Handwriting Recognition), Natural language Processing, computer Vision etc.

**Understanding Human Learning:**

This is the closest we have understood and mimicked the human brain. It is the start of a new revolution, The real AI. Now, after a brief insight lets come to a more formal definition of Machine Learning

## REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

## REQUIREMENT SPECIFICATION

### Functional Requirements

- Graphical User interface with the User.

### Software Requirements

For developing the application the following are the Software Requirements:

1. Python
2. Django

3. MySql
4. MySqlclient
5. WampServer 2.4

**Operating Systems supported**

1. Windows 7
2. Windows XP
3. Windows 8

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

- Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

## SYSTEM STUDY

### FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

- ♦ **ECONOMICAL FEASIBILITY**
- ♦ **TECHNICAL FEASIBILITY**
- ♦ **SOCIAL FEASIBILITY**

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## SYSTEM TEST

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.

### TYPES OF TESTS

#### Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**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. |

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**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.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

## Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

## Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## 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.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 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.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 9. CONCLUSION

Crimes are serious threats to human society, safety, and sustainable development and are thus meant to be controlled. Investigation authorities often demand computational predictions and predictive systems that improve crime analytics to further enhance the safety and security of cities and help to prevent crimes. Herein, we achieved an improved predictive accuracy for crimes by implementing different machine learning algorithms on Chicago and Los Angeles crime datasets. Among the different algorithms, XGBoost achieves the maximum accuracy on Chicago datasets and KNN achieves the maximum accuracy on Los Angeles. Data preprocessing was followed by splitting the dataset into training and testing sets, and later the performance parameters were examined. This study further applied a deep learning architecture for time series analysis through LSTM, by which the Chicago crime count had intense variations compared with Los Angeles, as shown by the RMSE and MAE. Also, the exploratory data analysis exhibited extensive visualizations regarding crime particulars, including crime rates in different periods from daily to yearly trends, crime types, and high-intensity areas based on historical patterns. Moreover, the implementation of an ARIMA model to predict the five-year trends regarding the crime rate and hot spots having high crime density suggest moderate variations for Chicago and a decline for Los Angles.

## 10. FUTURE SCOPE

For future work, this study will be expanded by using satellite imagery data, and the implementation of different learning techniques with corresponding visual data for different crime datasets.

**/Appendix**

**Visualization Libraries**

```
import matplotlib

import matplotlib.pyplot as plt

import seaborn as sns
```

```
#Preprocessing Libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import precision_score, recall_score, confusion_matrix,
classification_report, accuracy_score, f1_score

# ML Libraries

from sklearn.ensemble import RandomForestClassifier,VotingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural_network import MLPClassifier

# Evaluation Metricsl

from yellowbrick.classifier import ClassificationReport

from sklearn import metrics

from sklearn.svm import SVC

from google.colab import files

uploaded = files.upload()

df = pd.concat([pd.read_csv('data1.csv',error_bad_lines=False)],ignore_index
True)

df=pd.concat([pd.read_csv('data2.csv',error_bad_lines=False)],
ignore_index=True)
```

```
df = pd.concat([df, pd.read_csv('data3.csv', error_bad_lines=False)], ignore_index=True)

df = pd.concat([df, pd.read_csv('data4.csv', error_bad_lines=False)], ignore_index=True)

df.head()

<ipython-input-3-130af21cb0c5>:1: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_line

 df = pd.concat([pd.read_csv('data1.csv', error_bad_lines=False)],ignore_index=True)

<ipython-input-3-130af21cb0c5>:2: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_line

 df = pd.concat([pd.read_csv('data2.csv', error_bad_lines=False)], ignore_index=True)

<ipython-input-3-130af21cb0c5>:3: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_line

 df = pd.concat([df, pd.read_csv('data3.csv', error_bad_lines=False)], ignore_index=True)

<ipython-input-3-130af21cb0c5>:4: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_line
```

**df = pd.concat([df, pd.read_csv('data4.csv', error_bad_lines=False)], ignore_index=True)**

**Unnamed:**

**0**

**ID Case**

**Number Date Block IUCR Primary**

**Type Description Location**

**Description Arrest ... Ward Community**

**Area**

**FBI**

**Code**

**X**

**Coordinate**

**0 0 4673626 HM274058 04-02-**

**2006 13:00**

**055XX N**

**MANGO**

**AVE**

**2825 OTHER**

**OFFENSE**

**HARASSMENT BY**

**TELEPHONE RESIDENCE False ... 45.0 11.0 26 1136872.0**

**1 1 4673627 HM202199**

**02/26/2006**

**01:40:48**

**PM**

**065XX S**

**RHODES**

**AVE**

**2017 NARCOTICS MANU/DELIVER:CRACK SIDEWALK True ... 20.0 42.0 18 1181027.0**

**2 2 4673628 HM113861 01-08-**

**2006 23:16**

**013XX E**

**69TH ST 051A ASSAULT**

**AGGRAVATED:**

**HANDGUN OTHER False ... 5.0 69.0 04A 1186023.0**

**3 4 4673629 HM274049 04-05-**

```
2006 18:45

061XX W

NEWPORT

AVE

460 BATTERY SIMPLE

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 175975 entries, 0 to 175974

Data columns (total 23 columns):

 # Column Non-Null Count Dtype

--- ------ -------------- -----

 0 Unnamed: 0 175975 non-null int64

 1 ID 175975 non-null int64

 2 Case Number 175975 non-null object

 3 Date 175975 non-null object

 4 Block 175975 non-null object

 5 IUCR 175975 non-null object

 6 Primary Type 175975 non-null object

7 Description 175975 non-null object
```

```
 8 Location Description 175877 non-null object
 9 Arrest 175975 non-null bool
 10 Domestic 175975 non-null bool
 11 Beat 175975 non-null int64
 12 District 175974 non-null float64
 13 Ward 175973 non-null float64
 14 Community Area 175866 non-null float64
 15 FBI Code 175975 non-null object
 16 X Coordinate 155141 non-null float64
 17 Y Coordinate 155141 non-null float64
 18 Year 175975 non-null int64
 19 Updated On 175975 non-null object
 20 Latitude 155141 non-null float64
 21 Longitude 155141 non-null float64
 22 Location 155141 non-null object
dtypes: bool(2), float64(7), int64(4), object(10)
memory usage: 28.5+ MB
# Preprocessing
# Remove NaN Value (As Dataset is huge, the NaN row could be neglectable)
```

```
df = df.dropna()

# As the dataset is too huge is size, we would just subsampled a dataset for modelling as proof of concept

df = df.sample(n=100000)

# Remove irrelevant/not meaningfull attributes

df = df.drop(['Unnamed: 0'], axis=1)

df = df.drop(['ID'], axis=1)

df = df.drop(['Case Number'], axis=1)

df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 100000 entries, 4739 to 167319

Data columns (total 20 columns):

 # Column Non-Null Count Dtype

--- ------ -------------- -----

 0 Date 100000 non-null object

 1 Block 100000 non-null object

 2 IUCR 100000 non-null object

 3 Primary Type 100000 non-null object

 4 Description 100000 non-null object
 5 Location Description 100000 non-null object
 6 Arrest 100000 non-null bool
7 Domestic 100000 non-null bool
 8 Beat 100000 non-null int64
 9 District 100000 non-null float64
 10 Ward 100000 non-null float64
 11 Community Area 100000 non-null float64
 12 FBI Code 100000 non-null object
 13 X Coordinate 100000 non-null float64
 14 Y Coordinate 100000 non-null float64
 15 Year 100000 non-null int64
 16 Updated On 100000 non-null object
 17 Latitude 100000 non-null float64
 18 Longitude 100000 non-null float64
 19 Location 100000 non-null object
dtypes: bool(2), float64(7), int64(2), object(9)
memory usage: 14.7+ MB
```

```
# Splitting the Date to Day, Month, Year, Hour, Minute, Second
df['date2'] = pd.to_datetime(df['Date'])
```

```
df['Year'] = df['date2'].dt.year

df['Month'] = df['date2'].dt.month

df['Day'] = df['date2'].dt.day

df['Hour'] = df['date2'].dt.hour

df['Minute'] = df['date2'].dt.minute

df['Second'] = df['date2'].dt.second

df = df.drop(['Date'], axis=1)

df = df.drop(['date2'], axis=1)

df = df.drop(['Updated On'], axis=1)

df.head()
```

```
Block IUCR Primary

Type Description Location Description Arrest Domestic Beat District Ward ...

Y

Coordinate Year Latitu

4739 063XX N

HOYNE AVE 810 THEFT OVER $500 PARKING

LOT/GARAGE(NON.RESID.) False False 2413 24.0 50.0 ... 1942168.0 2006
41.9969

14717
```

**017XX W**

**JUNEWAY**

**TER**

**1350 CRIMINAL**

**TRESPASS**

**TO STATE**

**SUP LAND**

**CHA**

**HALLWAY/STAIRWELL/ELEVATOR True False 2422 24.0 49.0 ... 1951493.0 2006 42.0225**

**33881**

**026XX S**

**CHRISTIANA**

**AVE**

**810 THEFT OVER $500 STREET False False 1032 10.0 22.0 ... 1886346.0 2006 41.8439**

**25928 051XX S**

**RACINE AVE 1811 NARCOTICS**

**POSS:**

**CANNABIS**

**30GMS OR**

**LESS**

**STREET True False 933 9.0 16.0 ... 1870802.0 2006 41.8009**

**134031**

**032XX W**

**WASHINGTON**

**BLVD**

**1153 DECEPTIVE**

**PRACTICE**

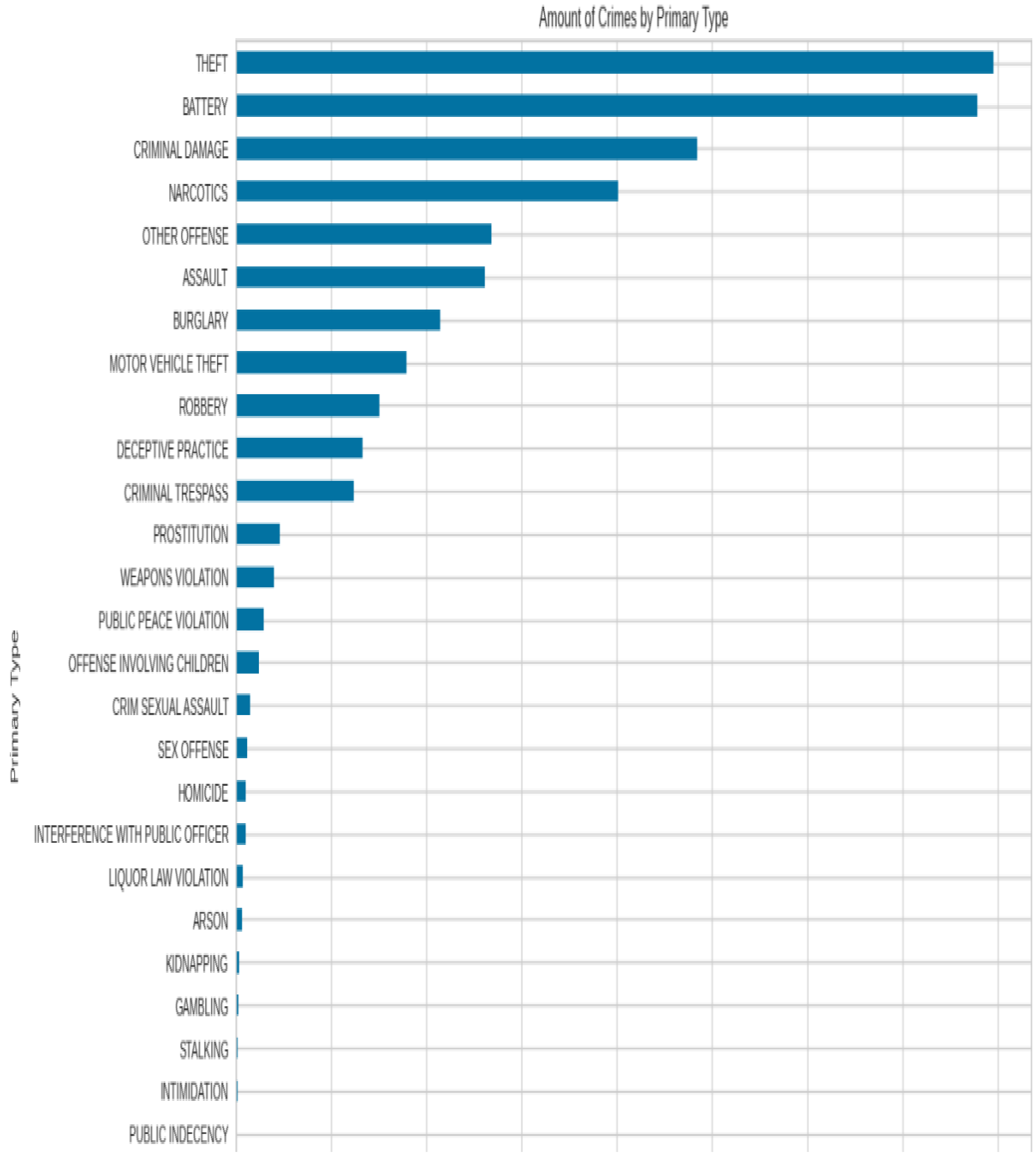
**FINANCIAL**

**IDENTITY**

**THEFT**

**OVER $ 300**

**RESIDENCE False False 1123 11.0 28.0 ... 1900496**



**.0 2015 41.8827**

**5 rows × 23 columns**

**# Convert Categorical Attributes to Numerical**

```
df['Block'] = pd.factorize(df["Block"])[0]

df['IUCR'] = pd.factorize(df["IUCR"])[0]

df['Description'] = pd.factorize(df["Description"])[0]

df['Location Description'] = pd.factorize(df["Location Description"])[0]

df['FBI Code'] = pd.factorize(df["FBI Code"])[0]

df['Location'] = pd.factorize(df["Location"])[0]

Target = 'Primary Type'

print('Target: ', Target)

Target: Primary Type

# Plot Bar Chart visualize Primary Types

plt.figure(figsize=(14,10))

plt.title('Amount of Crimes by Primary Type')

plt.ylabel('Crime Type')

plt.xlabel('Amount of Crimes')

df.groupby([df['Primary
Type']]).size().sort_values(ascending=True).plot(kind='barh')

plt.show()
```

```
# At previous plot, we could see that the classes is quite imbalance

# Therefore, we are going to group several less occured Crime Type into
'Others' to reduce the Target Class amount

# First, we sum up the amount of Crime Type happened and select the last 13
classes

all_classes = df.groupby(['Primary Type'])['Block'].size().reset_index()

all_classes['Amt'] = all_classes['Block']

all_classes = all_classes.drop(['Block'], axis=1)

all_classes = all_classes.sort_values(['Amt'], ascending=[False])

unwanted_classes = all_classes.tail(13)

unwanted_classes

Primary Type Amt

12 INTERFERENCE WITH PUBLIC OFFICER 254

15 LIQUOR LAW VIOLATION 187

0 ARSON 163

14 KIDNAPPING 95
```

**9 GAMBLING 70**

**28 STALKING 48**

**13 INTIMIDATION 46**

**24 PUBLIC INDECENCY 5**

**20 OBSCENITY 4**

**19 NON-CRIMINAL 2**

**18 NON - CRIMINAL 1**

**11 HUMAN TRAFFICKING 1**

**# After4 that, CONCEALED CARR we replaced itY LICENSE VIOLA with label 'OTHERS' TION 1**

**df.loc[df['Primary Type'].isin(unwanted_classes['Primary Type']), 'Primary Type'] = 'OTHERS'**

**# Plot Bar Chart visualize Primary Types**

**plt.figure(figsize=(14,10))**

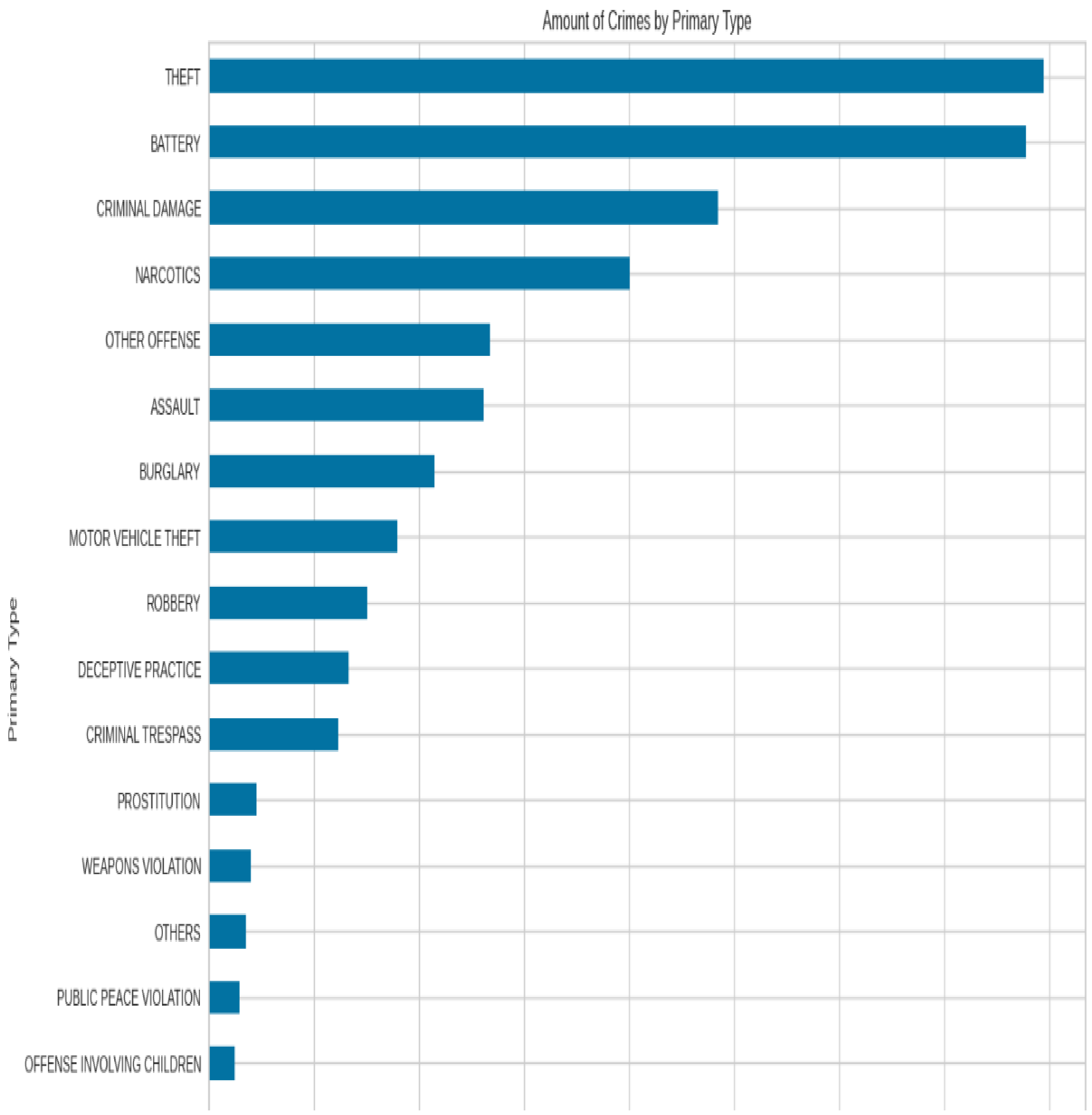
**plt.title('Amount of Crimes by Primary Type')**

**plt.ylabel('Crime Type')**

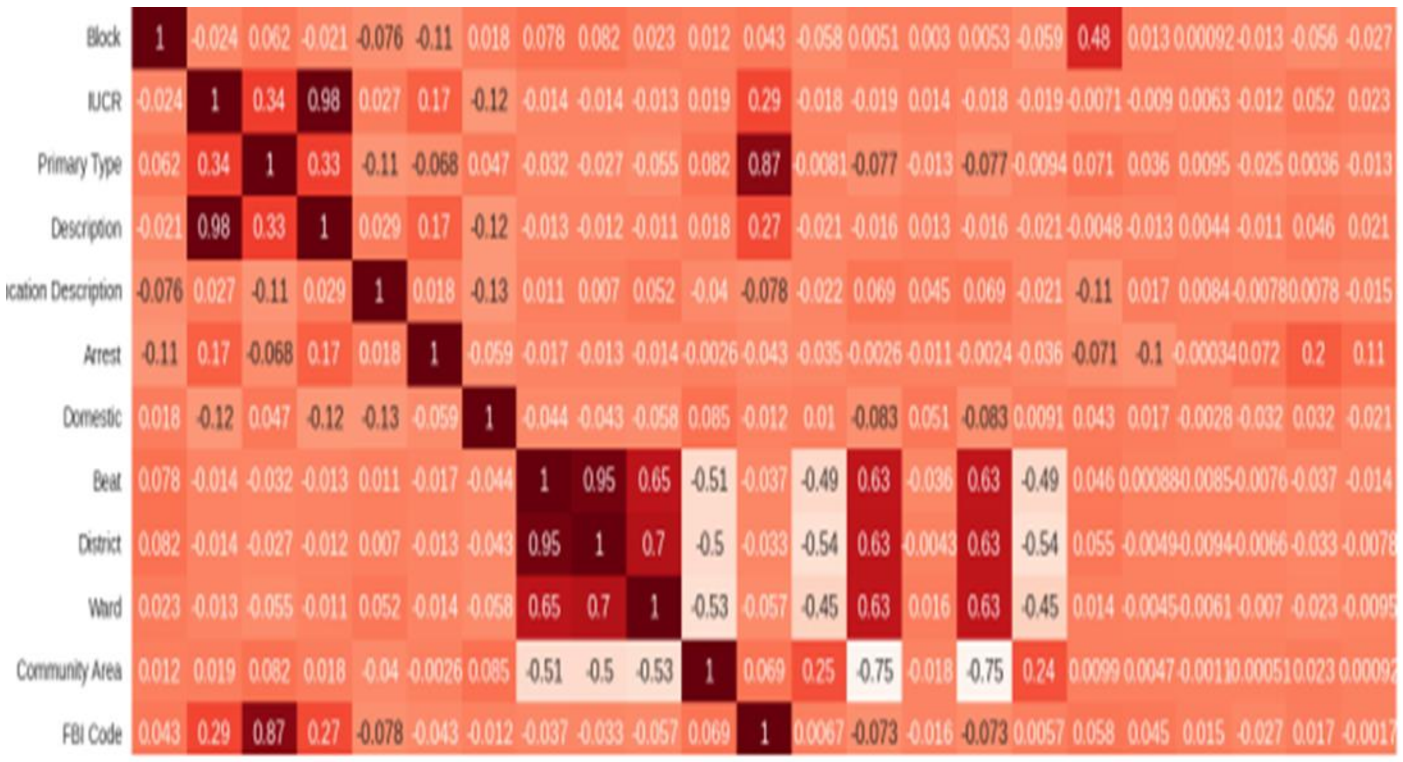
**plt.xlabel('Amount of Crimes')**

**df.groupby([df['Primary Type']]).size().sort_values(ascending=True).plot(kind='barh')**

**plt.show()**


Amount of Crimes by Primary Type
Primary Type
THEFT
BATTERY
CRIMINAL DAMAGE
NARCOTICS
OTHER OFFENSE
ASSAULT
BURGLARY
MOTOR VEHICLE THEFT
ROBBERY
DECEPTIVE PRACTICE
CRIMINAL TRESPASS
PROSTITUTION
WEAPONS VIOLATION
OTHERS
PUBLIC PEACE VIOLATION
OFFENSE INVOLVING CHILDREN

```
# Now we are left with 14 Class as our predictive class

Classes = df['Primary Type'].unique()

Classes

array(['THEFT', 'CRIMINAL TRESPASS', 'NARCOTICS', 'DECEPTIVE PRACTICE',

 'BATTERY', 'BURGLARY', 'PUBLIC PEACE VIOLATION', 'ROBBERY',

 'OTHER OFFENSE', 'CRIMINAL DAMAGE', 'SEX OFFENSE', 'ASSAULT',

 'CRIM SEXUAL ASSAULT', 'MOTOR VEHICLE THEFT', 'OTHERS',

 'OFFENSE INVOLVING CHILDREN', 'WEAPONS VIOLATION', 'PROSTITUTION',

 'HOMICIDE'], dtype=object)

#Encode target labels into categorical variables:

df['Primary Type'] = pd.factorize(df["Primary Type"])[0]

df['Primary Type'].unique()

array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,

 17, 18])

# Feature Selection using Filter Method
# Split Dataframe to target class and features

X_fs = df.drop(['Primary Type'], axis=1)

Y_fs = df['Primary Type']

#Using Pearson Correlation

plt.figure(figsize=(20,10))

cor = df.corr()

sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)

plt.show()
```



**Further Elaboration of Correlation**

**The correlation coefficient has values between -1 to 1**

**A value closer to 0 implies weaker correlation (exact 0 implying no correlation)**

**A value closer to 1 implies stronger positive correlation**

**A value closer to -1 implies stronger negative correlation**

```
#Correlation with output variable

cor_target = abs(cor['Primary Type'])

#Selecting highly correlated features

relevant_features = cor_target[cor_target>0.2]

relevant_features

IUCR 0.340309

Primary Type 1.000000

Description 0.333508

FBI Code 0.873328

Name: Primary Type, dtype: float64

# At Current Point, the attributes is select manually based on Feature Selection
Part.

Features = ["IUCR", "Description", "FBI Code"]

print('Full Features: ', Features)
```

```
Full Features: ['IUCR', 'Description', 'FBI Code']

#Split dataset to Training Set & Test Set

x, y = train_test_split(df,

  test_size = 0.2,

train_size = 0.8,

random_state= 3)

x1 = x[Features] #Features to train

x2 = x[Target] #Target Class to train

y1 = y[Features] #Features to test

y2 = y[Target] #Target Class to test

print('Feature Set Used : ', Features)

print('Target Class : ', Target)

print('Training Set Size : ', x.shape)

print('Test Set Size : ', y.shape)

Feature Set Used : ['IUCR', 'Description', 'FBI Code']

Target Class : Primary Type

Training Set Size : (80000, 23)

Test Set Size : (20000, 23)
```

**Machine Learning Modelling**

```
from sklearn.svm import SVC

from sklearn import metrics

svc=SVC() #Default hyperparameters

#svc.fit(X_train,y_train)

# Create Model with configuration

rf_model = RandomForestClassifier(n_estimators=70, # Number of trees

min_samples_split = 30,

bootstrap = True,

max_depth = 50,

min_samples_leaf = 25)

# Model Training

rf_model.fit(X=x1,

y=x2)

# Prediction

result = rf_model.predict(y[Features])

# Model Evaluation

ac_sc = accuracy_score(y2, result)

rc_sc = recall_score(y2, result, average="weighted")

pr_sc = precision_score(y2, result, average="weighted")
```

```
f1_sc = f1_score(y2, result, average='micro')

confusion_m = confusion_matrix(y2, result)

print("========== SVM Results ==========")

print("Accuracy : ", ac_sc)

print("Recall : ", rc_sc)

print("Precision : ", pr_sc)

print("F1 Score : ", f1_sc)

print("Confusion Matrix: ")

print(confusion_m)
```

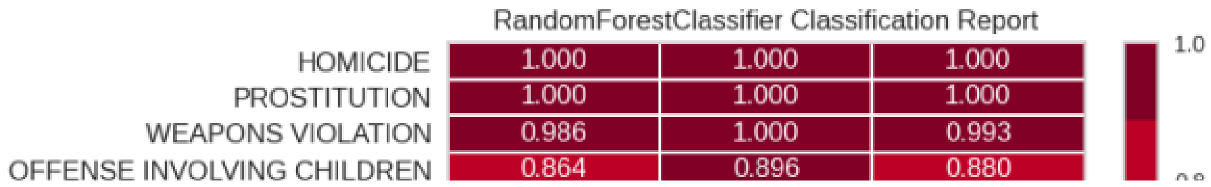
```
========== SVM Results ==========

Accuracy : 0.99485

Recall : 0.99485

Precision : 0.9949102531584498

F1 Score : 0.99485

Confusion Matrix:

[[3914 0 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 594 0 0 0 0 0 0 3 0 0 0 0 0

 0 0 0 0 0]
```

```
[ 0 0 2001 0 0 0 0 0 0 0 0 0 0 0

0 4 0 0 0]

[ 0 0 0 701 0 0 0 0 0 0 0 0 0 0

5 0 0 0 0]

[ 0 0 0 1 3913 0 0 0 0 0 0 0 0 0

0 0 0 0 0]

[ 0 0 0 0 0 1026 0 0 0 0 0 0 0 0

0 0 0 0 0]

[ 0 0 0 0 0 0 159 0 6 0 0 0 0 0

1 0 0 0 0]

[ 0 0 0 0 0 0 0 762 0 0 0 0 0 0

0 0 0 0 0]

[ 0 0 0 0 0 0 11 9 1339 0 2 0 0 0

3 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 2429 0 0 0 0

0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 59 0 0 0

0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 1251 0 0
```

```
 0 0 0 0 0]
 [ 0 0 0 0 2 0 0 0 0 0 0 6 73 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 947
 0 0 0 0 0]
 [ 0 2 3 0 0 0 12 0 0 0 4 7 0 0
 130 11 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 2 0 0 0
# Classification Report
# Instantiate the classification model and visualizer
target_names = Classes
visualizer = ClassificationReport(rf_model, classes=target_names)
visualizer.fit(X=x1, y=x2) # Fit the training data to the visualizer
visualizer.score(y1, y2) # Evaluate the model on the test data
print('================= Classification Report =================')
print('')
print(classification_report(y2, result, target_names=target_names))
g = visualizer.poof() # Draw/show/poof the data
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with

 warnings.warn(

================= Classification Report ==================

 precision recall f1-score support

 THEFT 1.00 1.00 1.00 3914

 CRIMINAL TRESPASS 1.00 0.99 1.00 597

 NARCOTICS 1.00 1.00 1.00 2005

 DECEPTIVE PRACTICE 1.00 0.99 1.00 706

 BATTERY 1.00 1.00 1.00 3914

 BURGLARY 1.00 1.00 1.00 1026

 PUBLIC PEACE VIOLATION 0.87 0.96 0.91 166

 ROBBERY 0.99 1.00 0.99 762

 OTHER OFFENSE 0.99 0.98 0.99 1364

 CRIMINAL DAMAGE 1.00 1.00 1.00 2429

 SEX OFFENSE 0.88 1.00 0.94 59

 ASSAULT 0.99 1.00 0.99 1251

 CRIM SEXUAL ASSAULT 1.00 0.90 0.95 81

 MOTOR VEHICLE THEFT 1.00 1.00 1.00 947

 OTHERS 0.90 0.77 0.83 169

OFFENSE INVOLVING CHILDREN 0.86 0.90 0.88 106

 WEAPONS VIOLATION 0.99 1.00 0.99 205

 PROSTITUTION 1.00 1.00 1.00 238
```

```
HOMICIDE 1.00 1.00 1.00 61

accuracy 0.99 20000

macro avg 0.97 0.97 0.97 20000

weighted avg 0.99 0.99 0.99 20000
```



RandomForestClassifier Classification Report

| | | | |
|---|---|---|---|
| HOMICIDE | 1.000 | 1.000 | 1.000 |
| PROSTITUTION | 1.000 | 1.000 | 1.000 |
| WEAPONS VIOLATION | 0.986 | 1.000 | 0.993 |
| OFFENSE INVOLVING CHILDREN | 0.864 | 0.896 | 0.880 |

```
# Neural Network(Convolution Neural Network)

# Create Model with configuration

nn_model = MLPClassifier(solver='adam',

alpha=1e-5,

hidden_layer_sizes=(40,),

random_state=1,

max_iter=1000

)

# Model Training

nn_model.fit(X=x1,

y=x2)

# Prediction

result = nn_model.predict(y[Features])

# Model Evaluation
```

```
ac_sc = accuracy_score(y2, result)
rc_sc = recall_score(y2, result, average="weighted")
pr_sc = precision_score(y2, result, average="weighted")
f1_sc = f1_score(y2, result, average='micro')
confusion_m = confusion_matrix(y2, result)
print("========== RCNN Neural Network Results ==========")
print("Accuracy : ", ac_sc)
print("Recall : ", rc_sc)
print("Precision : ", pr_sc)
print("F1 Score : ", f1_sc)
print("Confusion Matrix: ")
print(confusion_m)
========== RCNN Neural Network Results ==========
Accuracy : 0.96945
Recall : 0.96945
Precision : 0.9715802635814953
F1 Score : 0.96945
Confusion Matrix:
[[3914 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0]
 [ 0 523 0 0 0 0 0 0 74 0 0 0 0 0
 0 0 0 0 0]
 [ 0 0 1895 0 0 0 0 0 107 0 0 0 0 0
 3 0 0 0 0]
 [ 0 0 0 679 0 0 0 0 0 0 0 0 0 18
 0 0 4 5 0]
 [ 0 0 0 0 3885 0 0 26 0 0 0 0 0 0
 3 0 0 0 0]
 [ 0 0 0 0 0 1026 0 0 0 0 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 2 0 103 0 43 0 0 0 0 0
 18 0 0 0 0]
 [ 0 0 0 0 14 0 0 748 0 0 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 1355 0 0 0 0 0
 1 8 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 2427 0 2 0 0
0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 39 0 0 0
 20 0 0 0 0]
 [ 0 0 0 3 0 0 0 0 0 0 0 1219 0 11
 12 0 6 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 8 51 0
 22 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 944
 3 0 0 0 0]
```

```
 [ 0 0 0 0 18 10 0 0 11 8 0 0 0 11
 63 48 0 0 0]
 [ 0 0 0 0 0 0 0 0 29 0 0 1 0 0
 24 37 15 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 7
 0 9 189 0 0]
 [ 0 0 0 5 0 0 0 0 0 0 0 0 0 0
 0 0 2 231 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 61]]
```

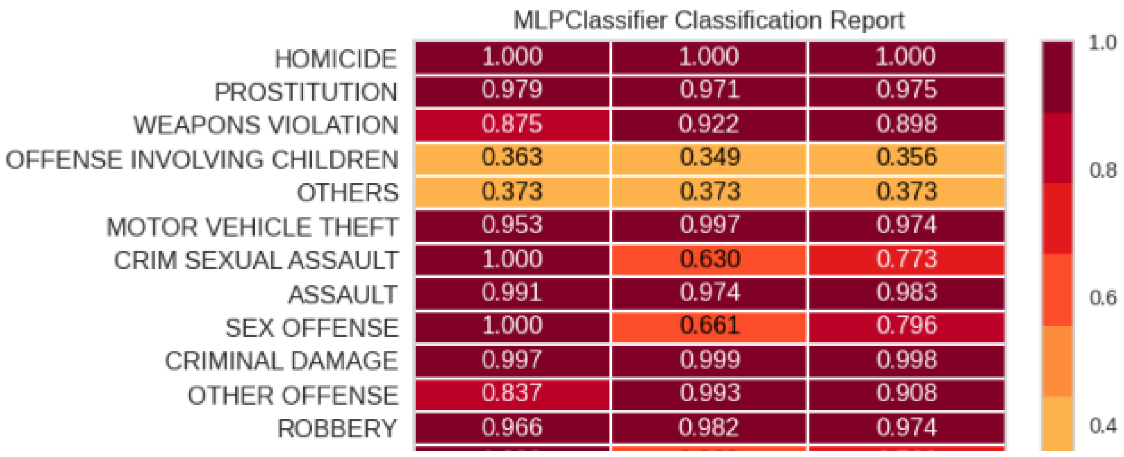
```
# Classification Report
# Instantiate the classification model and visualizer
target_names = Classes
visualizer = ClassificationReport(nn_model, classes=target_names)
visualizer.fit(X=x1, y=x2) # Fit the training data to the visualizer
visualizer.score(y1, y2) # Evaluate the model on the test data
print('================= Classification Report =================')
print('')
print(classification_report(y2, result, target_names=target_names))
g = visualizer.poof() # Draw/show/poof the data
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MLPClassifier was fitted with featur
```

```
 warnings.warn(
================= Classification Report ==================
 precision recall f1-score support
 THEFT 1.00 1.00 1.00 3914
 CRIMINAL TRESPASS 1.00 0.88 0.93 597
 NARCOTICS 1.00 0.95 0.97 2005
 DECEPTIVE PRACTICE 0.99 0.96 0.97 706
 BATTERY 0.99 0.99 0.99 3914
 BURGLARY 0.99 1.00 1.00 1026
 PUBLIC PEACE VIOLATION 1.00 0.62 0.77 166
 ROBBERY 0.97 0.98 0.97 762
 OTHER OFFENSE 0.84 0.99 0.91 1364
 CRIMINAL DAMAGE 1.00 1.00 1.00 2429
 SEX OFFENSE 1.00 0.66 0.80 59
 ASSAULT 0.99 0.97 0.98 1251
 CRIM SEXUAL ASSAULT 1.00 0.63 0.77 81
 MOTOR VEHICLE THEFT 0.95 1.00 0.97 947
 OTHERS 0.37 0.37 0.37 169
OFFENSE INVOLVING CHILDREN 0.36 0.35 0.36 106
 WEAPONS VIOLATION 0.88 0.92 0.90 205
 PROSTITUTION 0.98 0.97 0.97 238
 HOMICIDE 1.00 1.00 1.00 61
 accuracy 0.97 20000
```

macro avg 0.91 0.86 0.88 20000

weighted avg 0.97 0.97 0.97 20000

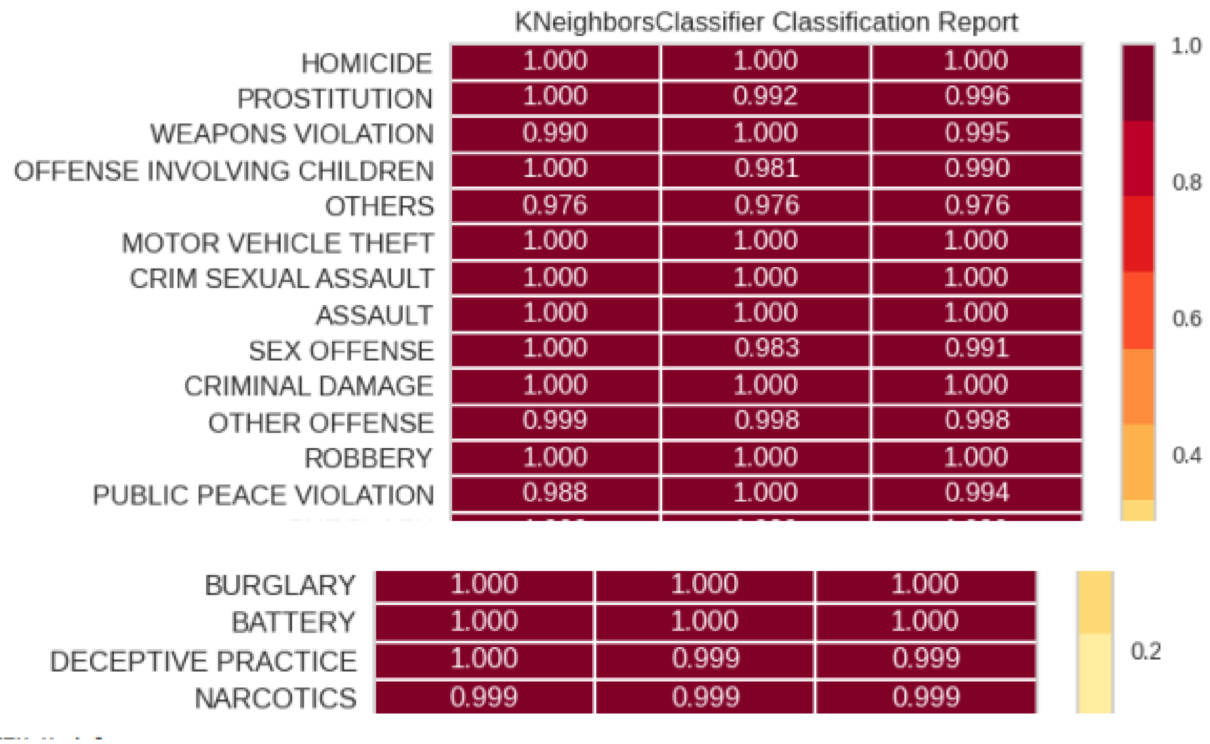


MLPClassifier Classification Report

| | | | |
|---|---|---|---|
| HOMICIDE | 1.000 | 1.000 | 1.000 |
| PROSTITUTION | 0.979 | 0.971 | 0.975 |
| WEAPONS VIOLATION | 0.875 | 0.922 | 0.898 |
| OFFENSE INVOLVING CHILDREN | 0.363 | 0.349 | 0.356 |
| OTHERS | 0.373 | 0.373 | 0.373 |
| MOTOR VEHICLE THEFT | 0.953 | 0.997 | 0.974 |
| CRIM SEXUAL ASSAULT | 1.000 | 0.630 | 0.773 |
| ASSAULT | 0.991 | 0.974 | 0.983 |
| SEX OFFENSE | 1.000 | 0.661 | 0.796 |
| CRIMINAL DAMAGE | 0.997 | 0.999 | 0.998 |
| OTHER OFFENSE | 0.837 | 0.993 | 0.908 |
| ROBBERY | 0.966 | 0.982 | 0.974 |

```
# K-Nearest Neighbors

# Create Model with configuration

knn_model = KNeighborsClassifier(n_neighbors=3)

# Model Training

knn_model.fit(X=x1,

y=x2)

# Prediction

result = knn_model.predict(y[Features])

# Model Evaluation

ac_sc = accuracy_score(y2, result)

rc_sc = recall_score(y2, result, average="weighted")

pr_sc = precision_score(y2, result, average="weighted")

f1_sc = f1_score(y2, result, average='micro')
```

```
confusion_m = confusion_matrix(y2, result)

print("========== K-Nearest Neighbors Results ==========")

print("Accuracy : ", ac_sc)

print("Recall : ", rc_sc)

print("Precision : ", pr_sc)

print("F1 Score : ", f1_sc)

print("Confusion Matrix: ")

print(confusion_m)

========== K-Nearest Neighbors Results ==========

Accuracy : 0.9992

Recall : 0.9992

Precision : 0.9992021914153906

F1 Score : 0.9992

Confusion Matrix:

[[3914 0 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 597 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 2003 0 0 0 0 0 1 0 0 0 0 0

 1 0 0 0 0]

 [ 0 0 0 705 0 0 0 0 1 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 0 0 3913 0 0 0 0 1 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 1026 0 0 0 0 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 166 0 0 0 0 0 0 0
 0 0 0 0 0]

[ 0 0 0 0 0 0 0 762 0 0 0 0 0 0
 0 0 0 0 0]
 [ 1 0 1 0 0 0 0 0 1361 0 0 0 0 0
 1 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 2429 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 1 0 0 0 0 0 58 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 1251 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 81 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 947
 0 0 0 0 0]
 [ 0 0 2 0 0 0 0 0 0 0 0 0 0 0
 165 0 2 0 0]
 [ 0 0 0 0 0 0 2 0 0 0 0 0 0 0
 0 104 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 205 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 2 0 0 236 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 61]]
# Classification Report
# Instantiate the classification model and visualizer
target_names = Classes
visualizer = ClassificationReport(knn_model, classes=target_names)
visualizer.fit(X=x1, y=x2) # Fit the training data to the visualizer
visualizer.score(y1, y2) # Evaluate the model on the test data
print('================= Classification Report =================')
print('')
print(classification_report(y2, result, target_names=target_names))
g = visualizer.poof() # Draw/show/poof the data
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with fe
 warnings.warn(
================= Classification Report =================
 precision recall f1-score support
 THEFT 1.00 1.00 1.00 3914
 CRIMINAL TRESPASS 1.00 1.00 1.00 597
 NARCOTICS 1.00 1.00 1.00 2005
 DECEPTIVE PRACTICE 1.00 1.00 1.00 706
 BATTERY 1.00 1.00 1.00 3914
 BURGLARY 1.00 1.00 1.00 1026
 PUBLIC PEACE VIOLATION 0.99 1.00 0.99 166
 ROBBERY 1.00 1.00 1.00 762
 OTHER OFFENSE 1.00 1.00 1.00 1364
 CRIMINAL DAMAGE 1.00 1.00 1.00 2429
 SEX OFFENSE 1.00 0.98 0.99 59
 ASSAULT 1.00 1.00 1.00 1251
 CRIM SEXUAL ASSAULT 1.00 1.00 1.00 81
 MOTOR VEHICLE THEFT 1.00 1.00 1.00 947
 OTHERS 0.98 0.98 0.98 169
OFFENSE INVOLVING CHILDREN 1.00 0.98 0.99 106
 WEAPONS VIOLATION 0.99 1.00 1.00 205
 PROSTITUTION 1.00 0.99 1.00 238
 HOMICIDE 1.00 1.00 1.00 61
 accuracy 1.00 20000
 macro avg 1.00 1.00 1.00 20000
 weighted avg 1.00 1.00 1.00 20000
```



| | | | |
|---|---|---|---|
| HOMICIDE | 1.000 | 1.000 | 1.000 |
| PROSTITUTION | 1.000 | 0.992 | 0.996 |
| WEAPONS VIOLATION | 0.990 | 1.000 | 0.995 |
| OFFENSE INVOLVING CHILDREN | 1.000 | 0.981 | 0.990 |
| OTHERS | 0.976 | 0.976 | 0.976 |
| MOTOR VEHICLE THEFT | 1.000 | 1.000 | 1.000 |
| CRIM SEXUAL ASSAULT | 1.000 | 1.000 | 1.000 |
| ASSAULT | 1.000 | 1.000 | 1.000 |
| SEX OFFENSE | 1.000 | 0.983 | 0.991 |
| CRIMINAL DAMAGE | 1.000 | 1.000 | 1.000 |
| OTHER OFFENSE | 0.999 | 0.998 | 0.998 |
| ROBBERY | 1.000 | 1.000 | 1.000 |
| PUBLIC PEACE VIOLATION | 0.988 | 1.000 | 0.994 |
| BURGLARY | 1.000 | 1.000 | 1.000 |
| BATTERY | 1.000 | 1.000 | 1.000 |
| DECEPTIVE PRACTICE | 1.000 | 0.999 | 0.999 |
| NARCOTICS | 0.999 | 0.999 | 0.999 |

```
# Ensemble LSTM Model

import tensorflow as tf

from keras.layers import Dense, BatchNormalization, Dropout, LSTM, Bidirectional

from keras.models import Sequential

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import regularizers

from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report, accuracy_score, f1_score

from keras import callbacks

from tensorflow.keras.callbacks import EarlyStopping

# Combine 3 Models to create an Ensemble Model
# Define and compile model
from tensorflow import keras
model = keras.Sequential()
model.add(Dense(28 , input_shape=(56,) , activation="relu" ,
name="Hidden_Layer_1"))
model.add(Dense(10 , activation="relu" , name="Hidden_Layer_2"))
model.add(Dense(1 , activation="sigmoid" , name="Output_Layer"))
opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile( optimizer=opt, loss="binary_crossentropy", metrics=['accuracy'])
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type) Output Shape Param #
=================================================================
 Hidden_Layer_1 (Dense) (None, 28) 1596


 Hidden_Layer_2 (Dense) (None, 10) 290
 Output_Layer (Dense) (None, 1) 11
=================================================================
Total params: 1,897
Trainable params: 1,897
Non-trainable params: 0
```

```
# Create Model with configuration

eclf1 = VotingClassifier(estimators=[('knn', knn_model), ('rf', rf_model), ('nn',
nn_model)],

weights=[1,1,1],

flatten_transform=True)

eclf1 = eclf1.fit(X=x1, y=x2)

# Prediction

result = eclf1.predict(y[Features])

# Model Evaluation

ac_sc = accuracy_score(y2, result)

rc_sc = recall_score(y2, result, average="weighted")

pr_sc = precision_score(y2, result, average="weighted")

f1_sc = f1_score(y2, result, average='micro')

confusion_m = confusion_matrix(y2, result)

print("============= LSTM Results =============")

print("Accuracy : ", ac_sc)

print("Recall : ", rc_sc)

print("Precision : ", pr_sc)

print("F1 Score : ", f1_sc)

print("Confusion Matrix: ")

print(confusion_m)
```

```
============= LSTM Results =============

Accuracy : 0.9971
```

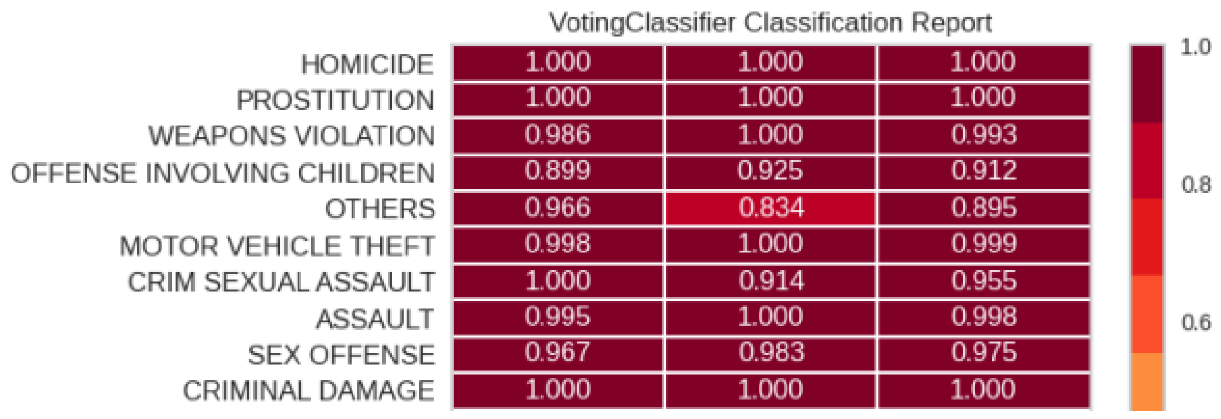
```
Recall : 0.9971

Precision : 0.9970885706244451

F1 Score : 0.9971

Confusion Matrix:

[[3914 0 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 594 0 0 0 0 0 0 3 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 2004 0 0 0 0 0 0 0 0 0 0 0

 1 0 0 0 0]

 [ 0 0 0 706 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 0 0 3914 0 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 0 0 0 1026 0 0 0 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 0 0 0 0 160 0 6 0 0 0 0 0

 0 0 0 0 0]

 [ 0 0 0 0 0 0 0 762 0 0 0 0 0 0

 0 0 0 0 0]

[ 1 0 1 0 0 0 0 0 1360 0 0 0 0 0

 2 0 0 0 0]

 [ 0 0 0 0 0 0 0 0 0 2429 0 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 1 0 0 0 0 0 58 0 0 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 1251 0 0
 0 0 0 0 0]
 [ 0 0 0 0 2 0 0 0 0 0 0 5 74 0
 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 947
 0 0 0 0 0]
 [ 0 0 5 0 0 10 0 0 0 0 0 0 0 2
 141 11 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 2 1 0 0
 2 98 3 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 205 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 238 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 61]]
```

```
# Classification Report
# Instantiate the classification model and visualizer
target_names = Classes
visualizer = ClassificationReport(eclf1, classes=target_names)
```

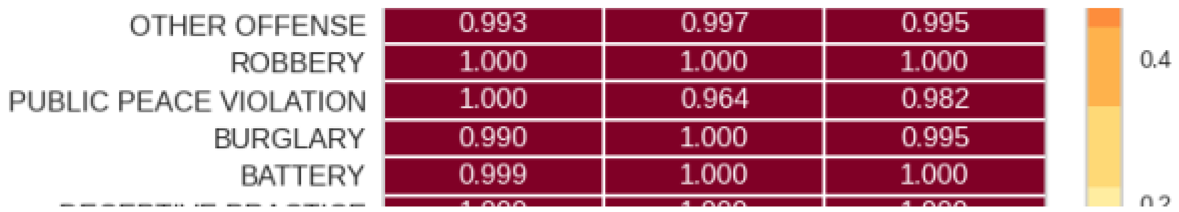
```
visualizer.fit(X=x1, y=x2) # Fit the training data to the visualizer

visualizer.score(y1, y2) # Evaluate the model on the test data

print('================= Classification Report =================')

print('')

print(classification_report(y2, result, target_names=target_names))

g = visualizer.poof() # Draw/show/poof the data
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with fe

 warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with

 warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MLPClassifier was fitted with feature n

 warnings.warn(

================= Classification Report =================

 precision recall f1-score support

 THEFT 1.00 1.00 1.00 3914

 CRIMINAL TRESPASS 1.00 0.99 1.00 597

 NARCOTICS 1.00 1.00 1.00 2005

 DECEPTIVE PRACTICE 1.00 1.00 1.00 706

 BATTERY 1.00 1.00 1.00 3914

 BURGLARY 0.99 1.00 1.00 1026

 PUBLIC PEACE VIOLATION 1.00 0.96 0.98 166
```

```
ROBBERY 1.00 1.00 1.00 762

OTHER OFFENSE 0.99 1.00 1.00 1364

CRIMINAL DAMAGE 1.00 1.00 1.00 2429

SEX OFFENSE 0.97 0.98 0.97 59

ASSAULT 1.00 1.00 1.00 1251

CRIM SEXUAL ASSAULT 1.00 0.91 0.95 81

MOTOR VEHICLE THEFT 1.00 1.00 1.00 947

OTHERS 0.97 0.83 0.90 169

OFFENSE INVOLVING CHILDREN 0.90 0.92 0.91 106

WEAPONS VIOLATION 0.99 1.00 0.99 205

PROSTITUTION 1.00 1.00 1.00 238

HOMICIDE 1.00 1.00 1.00 61

accuracy 1.00 20000

macro avg 0.99 0.98 0.98 20000

weighted avg 1.00 1.00 1.00 20000
```



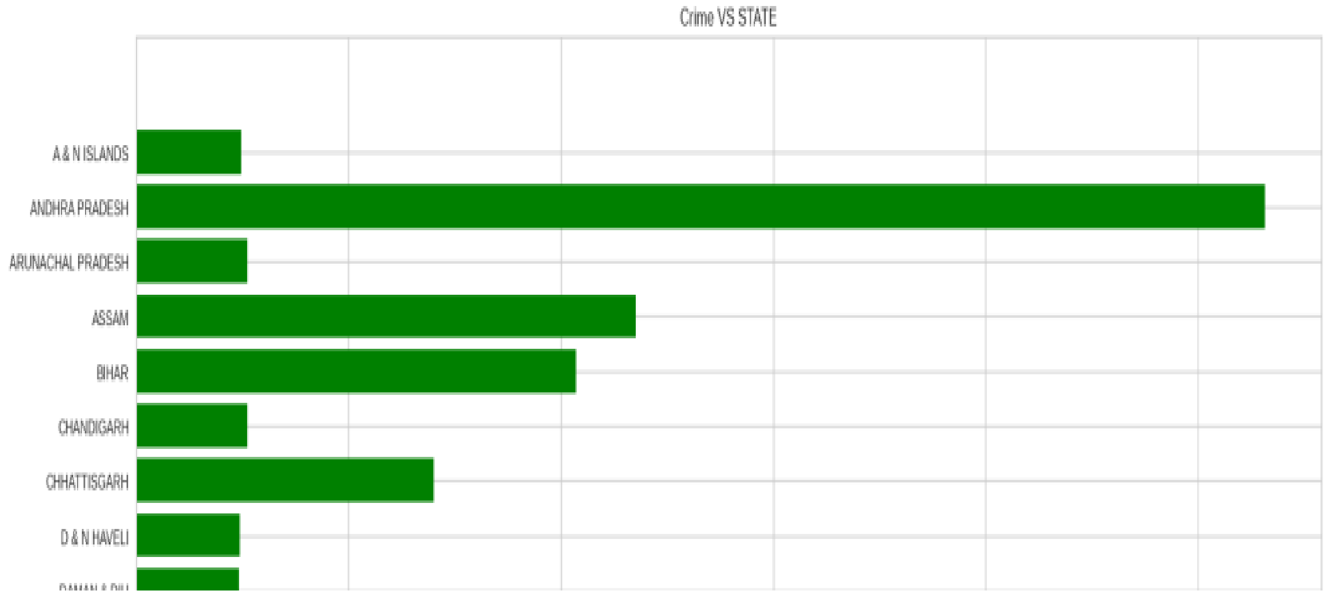
| | | | |
|---|---|---|---|
| HOMICIDE | 1.000 | 1.000 | 1.000 |
| PROSTITUTION | 1.000 | 1.000 | 1.000 |
| WEAPONS VIOLATION | 0.986 | 1.000 | 0.993 |
| OFFENSE INVOLVING CHILDREN | 0.899 | 0.925 | 0.912 |
| OTHERS | 0.966 | 0.834 | 0.895 |
| MOTOR VEHICLE THEFT | 0.998 | 1.000 | 0.999 |
| CRIM SEXUAL ASSAULT | 1.000 | 0.914 | 0.955 |
| ASSAULT | 0.995 | 1.000 | 0.998 |
| SEX OFFENSE | 0.967 | 0.983 | 0.975 |
| CRIMINAL DAMAGE | 1.000 | 1.000 | 1.000 |

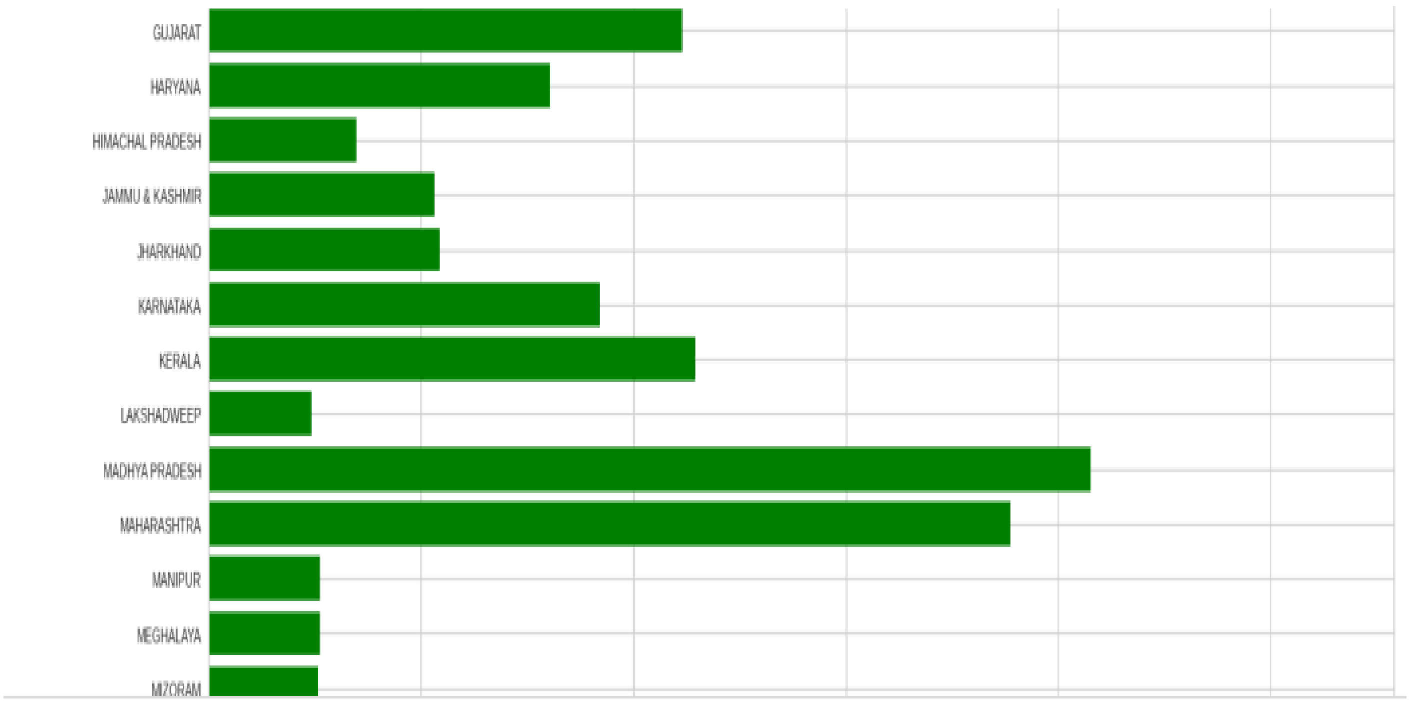


```
from google.colab import files

uploaded = files.upload()

import numpy as np

crimes_total_women1 = pd.read_csv('District2001_2020.csv')

crimes_total_women2= pd.read_csv('District_wise_crimes2022.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('Overall districtwise Crimerate')

ax1.set_title('Crime VS STATE')

fig1.set_size_inches(20, 18, forward=True)

plt.show()
```

```
<ipython-input-35-c1f36bf0c25e>:15: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
fut

 crimes_total_women['Total Crimes']= crimes_total_women.iloc[:, -9:-
1].sum(axis=1)
```


Crime VS STATE
A & N ISLANDS
ANDHRA PRADESH
ARUNACHAL PRADESH
ASSAM
BIHAR
CHANDIGARH
CHHATTISGARH
D & N HAVELI


GUJARAT
HARYANA
HIMACHAL PRADESH
JAMMU & KASHMIR
JHARKHAND
KARNATAKA
KERALA
LAKSHADWEEP
MADHYA PRADESH
MAHARASHTRA
MANIPUR
MEGHALAYA
MIZORAM

## 11. REFERENCE

[1] G. Mohler, ‘‘Marked point process hotspot maps for homicide and gun crime prediction in Chicago,’’ Int. J. Forecasting, vol. 30, no. 3, pp. 491–497, Jul. 2014.

[2] A. Iriberri and G. Leroy, ‘‘Natural language processing and e-government: Extracting reusable crime report information,’’ in Proc. IEEE Int. Conf. Inf. Reuse Integr., Las Vegas, IL, USA, Aug. 2007, pp. 221–226.

[3] V. Pinheiro, V. Furtado, T. Pequeno, and D. Nogueira, ‘‘Natural language processing based on semantic inferentialism for extracting crime information from text,’’ in Proc. IEEE Int. Conf. Intell. Secur. Informat., Vancouver, BC, Canada, May 2010, pp. 19–24.

[4] B. Wang, P. Yin, A. L. Bertozzi, P. J. Brantingham, S. J. Osher, and J. Xin, ‘‘Deep learning for real-time crime forecasting and its ternarization,’’ Chin. Ann. Math., B, vol. 40, no. 6, pp. 949–966, Nov. 2019.

[5] S. Chackravarthy, S. Schmitt, and L. Yang, ‘‘Intelligent crime anomaly detection in smart cities using deep learning,’’ in Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC), Philadelphia, PA, USA, Oct. 2018, pp. 399–404.

[6] H.-W. Kang and H.-B. Kang, ‘‘Prediction of crime occurrence from multimodal data using deep learning,’’ PLoS ONE, vol. 12, no. 4, Apr. 2017, Art. no. e0176244.

[7] A. Fidow, M. Hassan, M. Imran, X. Cheng, C. Petridis, and C. Sule, ‘‘Suggesting a hybrid approach mobile apps with big data analysis to report and prevent crimes,’’ in Social Media Strategy in Policing (Security Informatics and Law Enforcement), B. Akhgar, P. S. Bayeri, and G. Leventakis, Eds. Cham, Switzerland: Springer, 2019, pp. 177–195.

[8] P. J. Brantingham, M. Valasik, and G. O. Mohler, ‘‘Does predictive policing lead to biased arrests? Results from a randomized controlled trial,’’ Statist. Public Policy, vol. 5, no. 1, pp. 1–6, Jan. 2018.

[9] A. Nasridinov and Y.-H. Park, ‘‘A study on performance evaluation of machine learning algorithms for crime dataset,’’ Adv. Sci. Technol. Lett., vol. 90, pp. 90–92, Dec. 2014.

[10] A. Stec and D. Klabjan, ‘‘Forecasting crime with deep learning,’’ 2018, arXiv:1806.01486. [Online]. Available: http://arxiv.org/abs/1806.01486

[11] J. Fitterer, T. A. Nelson, and F. Nathoo, ‘‘Predictive crime mapping,’’ Police Pract. Res., vol. 16, no. 2, pp. 121–135, Mar. 2015.

[12] A. Najjar, S. Kaneko, and Y. Miyanaga, ‘‘Crime mapping from satellite imagery via deep learning,’’ 2018, arXiv: 1812.06764. [Online]. Available: http://arxiv.org/abs/1812.06764

[13] H. Wang, D. Kifer, C. Graif, and Z. Li, ‘‘Crime rate inference with big data,’’ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, Aug. 2016, pp. 635–644.

[14] X. Zhang, L. Liu, L. Xiao, and J. Ji, ‘‘Comparison of machine learning algorithms for predicting crime hotspots,’’ IEEE Access, vol. 8, pp. 181302–181310, 2020.

[15] G. R. Nitta, B. Y. Rao, T. Sravani, N. Ramakrishiah, and M. BalaAnand, ‘‘LASSO-based feature selection and Naïve Bayes classifier for crime prediction and its type,’’ Service Oriented Comput. Appl., vol. 13, no. 3, pp. 187–197, Sep. 2019.