**School of Computer Science and Engineering**

# CRIME ANALYSIS AND PREDICTION

**CSE3021 – Social and Information Networks - a Project Report**

**Submitted to:**

**Dr Punitha.K**

**Submitted by:**

**AADHITYA SWARNESH (18BCE1087)**

**SUMIT AJMERA (18BCE1223)**

In partial fulfilment for the award of the degree of

**Bachelor of Technology**

in

**Computer Science and Engineering**

*June 2021*

# TABLE OF CONTENTS

## TABLE OF FIGURES

## I. ABSTRACT

“**Injustice Anywhere is a threat to Justice Everywhere**” -- Following this saying, we notice that crimes exist all around the world, and are a never ending issue to humanity. As humans evolve crimes do as well and it falls on us to keep it under check. Machine Learning has swept the world and has shown what computers are capable of with the right data. In this paper, we aim to apply Machine Learning Algorithms to improve the field of crime analysis and prediction. We have analysed the top rated machine learning algorithms on being able to predict crimes to a good scale of accuracy. This paper can be divided into two sections, the first part deals with the analysis of a decade's worth of crime data from all around the country of India. Here we will discuss the top categories of crimes, and also how they are distributed across the states and years. In the second part, we divert our focus on one particular city and show how Machine Learning can help in predicting crimes. For this we consider a few scenarios including but not limited to predicting the future crimes that might take place in a particular locality, the seriousness of the crime based on Intelligence bureau standards, etc. We have applied a total count of 7 ML algorithms consisting of Decision Tree, Bernoulli and Gaussian Naive Bayes, Light GBM, Random Forest Classifiers. We have compared the results obtained across these models and the model trained with the Light GBM model has so far produced better results among the lot.

## II. INTRODUCTION

Crime is a never-ending feature for mankind, crimes by definition would continue to occur as long as there exist at least two individuals who see things in different perspectives and have very less or no control over their actions. These traits exist in individuals in a society, it cannot be eradicated, but losing hope is not an option in this case. It rests upon us to figure out ways to fight crime, and eradicate it from the roots. One important aspect that is needed to achieve this is coordination between agencies, to share data between themselves. Because jurisdictional issues lead to the rise of crimes which in turn causes loss of stability in the region. Data is the most important factor here that should be paving the way towards the future from now. Ever since the advent of computers, we have been digitizing all records, and the same goes for crime records as well. This curated list of records is what should be used as the data and fed into specially designed machines. We have been going along with the human way of trying to reduce crimes, though they have been effective so far, the crimes these days are getting more in quantity and also technical. This calls for better strategies, and I believe computers could help us here. This is where the term **Machine Learning** comes into play.

As the whole world turns to Machine Learning to evolve their tasks, we believe it's time for police  to turn towards this as well. ML has fundamentally changed our notion of what machines are capable of, we believe that this is the correct time to harness this power. The advent of computers have turned all portions of our life digital, which means more data for the machines to learn upon. This change which has been happening for the past decade, having us collect huge quantities of data which when fed into a Machine Learning Algorithm can learn the traits, figure out the trends, and thereby suggest loopholes in our system or help us in figuring out better techniques to predict future crimes and thereby allowing us to concentrate on hot spots of crimes which in turn might drastically reduce the crime rates helping humanity evolve and have a better future.

We have in this paper dealt with the  strategies where we can employ Machine Learning to help predict future trends on a broader perspective. We believe that based on a set of previous data gathered through the years for a small locality, we might form a specialized model for making predictions for say the seriousness of a crime that might take place in the near future, or the locality where the chances of the crime rates to hike is high. These might be due to any number of reasons in the real world, but these ML models just learn trends from the data collected to predict these.

The remainder of the paper has been organized as follows. In section **III**, we cover how feasible the whole project is by saying the devices that might be required, the manpower that might be required, etc. Section **IV** explains the dataset used for the analysis and also for making the predictions, some of their features and traits. Section **V** details the architecture proposed in this paper detailing the flow of working of our ML based prediction model. Section **VI** details the modules covered in our project, here we will be discussing both the country wide analysis of crime stats, and also about the much localized and detailed approach in making crime stat predictions for the near future. Section **VII** details the ML Algorithms used in the architecture, and also about their specialized usage in other fields, and how each plays a role in our model. Section **VIII** explains the risk involved in the project and also on deploying it in a real world scenario. Section **IX** details the tools and softwares used in developing and implementing our work, whose results have been elaborated upon in section **X**. We conclude our report in section **XI** followed by quoting the referred upon papers and articles in section **XII**.

## III. FEASIBILITY STUDY

A Feasibility study is absolutely necessary for any project, this is where we detail the hurdles that we have faced during the development of this project and also the ones we will have to face during its production stage. This project uses **Machine Learning** at its core, and so it is necessary for the model to be trained on an ample quantity of data before it is expected to reach a proper level of accuracy. So the **need for crime data is high**, so far only a handful of regions have satisfied these conditions, two of which are **Boston** and **Chicago**. So it was difficult to find the right dataset to build the model upon. When ML is concerned, it raises another question of how complex the training process will be, this naturally depends on the model used. As we have used a few standard models, and because the dataset was small when compared to the real world, our training time was within an hour. But this is far from the training time we could get when the model is trained upon terabytes of data. So a **sophisticated piece of hardware is necessary** for these models to train upon. With today's advances in **cloud computing**, we can solve this by hiring them to host our models. They provide high class hardware at relatively lower costs.

When ML is concerned, there arises a question about if this is an application that is to be run on **edge devices**, but this application is more of something that has to remain behind the curtains and provide only valuable results to the end user, and so there is no need for edge devices here. Predicting crimes is a **relatively new field**, and so there are not a lot of models trained here beforehand, and so customizing the models for this particular scenario by altering the parameters was required to produce good results. At the same time the models trained over here **do not cover all the parameters** that might exist in the real world, and are the reason why us humans are good at tracing and catching criminals. So we are in need of better representations of real world scenarios to generate a more representative dataset. So we call for a community to rise up and take upon this as a challenge to innovate customized algorithms in this field.

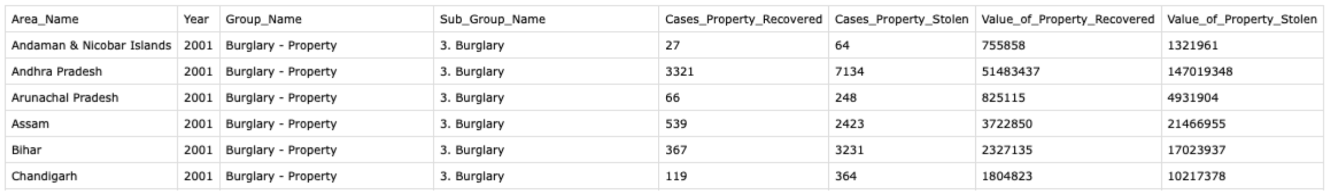
There also arises a need to **train the current set of police forces** to better equip the knowledge that these models produce, and use this as an external advice in performing their duties. We also have to address the most important issue of the **shortage of silicon chips** that has affected the world we live in. We humans are desperately searching for alternatives for silicon chips as the core component in our computers. This hurdle needs to be passed in order for ML to thrive upon and for our project to be applied in real life.

## IV. DATASET DESCRIPTION

We have used a total of three datasets in this project. As mentioned earlier in this report, this project has been divided into two modules, whose details have been covered in a later section. For now, we will elaborate upon the datasets used here in these modules.

The first module is regarding the analysis of crimes in the whole country of India. For this module, we use the "Crimes in India" dataset obtained from Kaggle. This is a repository of nearly 75 CSV files each of which lists out a particular category of crime across the states and years. Each file has data going back for nearly a decade on almost all the states. We have used this dataset mainly for performing Exploratory Data Analysis. We choose a few of these files by choosing the most important crime categories from the lot, and performing analysis on them. The dataset is available in the following link :

Dataset : **https://www.kaggle.com/rajanand/crime-in-india**

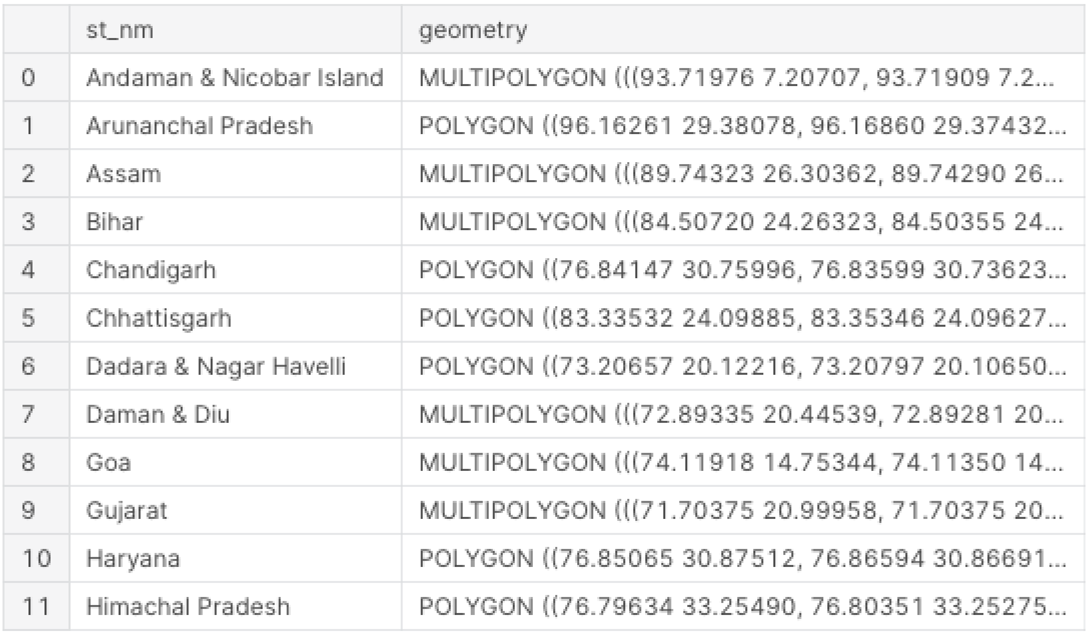


| Area_Name | Year | Group_Name | Sub_Group_Name | Cases_Property_Recovered | Cases_Property_Stolen | Value_of_Property_Recovered | Value_of_Property_Stolen |
|---|---|---|---|---|---|---|---|
| Andaman & Nicobar Islands | 2001 | Burglary - Property | 3. Burglary | 27 | 64 | 755858 | 1321961 |
| Andhra Pradesh | 2001 | Burglary - Property | 3. Burglary | 3321 | 7134 | 51483437 | 147019348 |
| Arunachal Pradesh | 2001 | Burglary - Property | 3. Burglary | 66 | 248 | 825115 | 4931904 |
| Assam | 2001 | Burglary - Property | 3. Burglary | 539 | 2423 | 3722850 | 21466955 |
| Bihar | 2001 | Burglary - Property | 3. Burglary | 367 | 3231 | 2327135 | 17023937 |
| Chandigarh | 2001 | Burglary - Property | 3. Burglary | 119 | 364 | 1804823 | 10217378 |

Figure 1 - A sample of the Crimes in India Dataset listing the property theft across states and years

While performing data analysis using these data files, we have also created plots for better visualizations. For plotting these data on the map of India, we have used the "India GIS data" available through kaggle. It details the shape of the country and also the shape of every state listed as a polygon of many dimensions. These polygons can be used to plot the map of India and also to mark the territory of the states in India accurately. The dataset can be obtained through the following link :

Dataset : **https://www.kaggle.com/nehaprabhavalkar/india-gis-data**

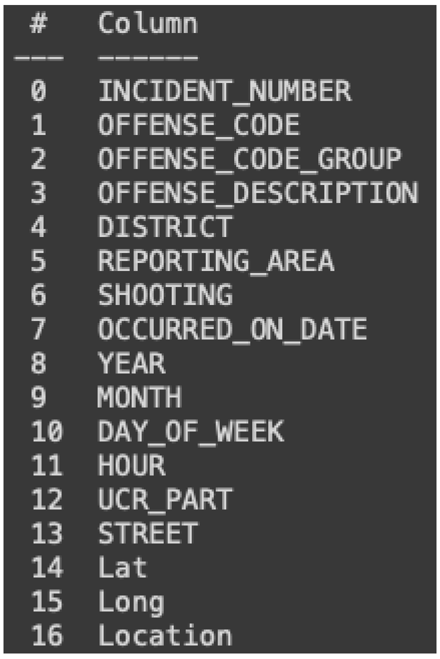


| | st_nm | geometry |
|---|---|---|
| 0 | Andaman & Nicobar Island | MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2... |
| 1 | Arunanchal Pradesh | POLYGON ((96.16261 29.38078, 96.16860 29.37432... |
| 2 | Assam | MULTIPOLYGON (((89.74323 26.30362, 89.74290 26... |
| 3 | Bihar | MULTIPOLYGON (((84.50720 24.26323, 84.50355 24... |
| 4 | Chandigarh | POLYGON ((76.84147 30.75996, 76.83599 30.73623... |
| 5 | Chhattisgarh | POLYGON ((83.33532 24.09885, 83.35346 24.09627... |
| 6 | Dadara & Nagar Havelli | POLYGON ((73.20657 20.12216, 73.20797 20.10650... |
| 7 | Daman & Diu | MULTIPOLYGON (((72.89335 20.44539, 72.89281 20... |
| 8 | Goa | MULTIPOLYGON (((74.11918 14.75344, 74.11350 14... |
| 9 | Gujarat | MULTIPOLYGON (((71.70375 20.99958, 71.70375 20... |
| 10 | Haryana | POLYGON ((76.85065 30.87512, 76.86594 30.86691... |
| 11 | Himachal Pradesh | POLYGON ((76.79634 33.25490, 76.80351 33.25275... |

Figure 2 - A sample of the India GIS Dataset listing the states and their territorial shapes

As for our second module which depends on the data for a much localized region, we have chosen the "**Boston Crime Data**". We have chosen this dataset mainly because of the extensive set of attributes that it lists which serve well with our goals. There are very few regions like Boston and Chicago that offer such extensive data like the time of occurrence of the crime and also the accurate latitude and longitude coordinates labelling the place of crime occurrence. When feeding data for a ML model, the more detailed the data is, the better will be the predictions given by the model. So this amount of detail is absolutely necessary to reach our goals. The dataset can be obtained through the following link :

Dataset : **https://www.kaggle.com/ankkur13/boston-crime-data**



```
 #   Column
---  ------
 0   INCIDENT_NUMBER
 1   OFFENSE_CODE
 2   OFFENSE_CODE_GROUP
 3   OFFENSE_DESCRIPTION
 4   DISTRICT
 5   REPORTING_AREA
 6   SHOOTING
 7   OCCURRED_ON_DATE
 8   YEAR
 9   MONTH
 10  DAY_OF_WEEK
 11  HOUR
 12  UCR_PART
 13  STREET
 14  Lat
 15  Long
 16  Location
```

Figure 3 - A sample list of attributes available in the Boston Crime Dataset

## V. PROPOSED ARCHITECTURE

Our proposal for the architecture covers an extensive three stage process backboning on the choice of ML models in the second stage. The entire architecture has been crafted as shown in the diagram below :

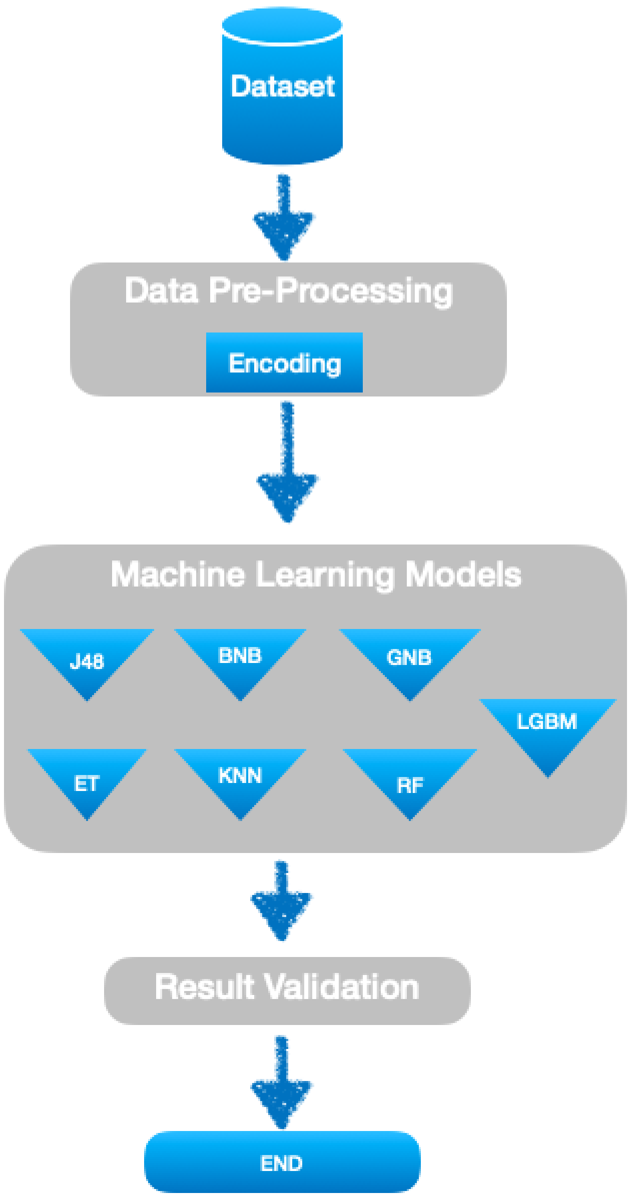


Figure 4 - Architecture Diagram of proposed system

Let us now break up this architecture and look into its parts in more detail :

The first part of this architecture deals with the flow of data into the preprocessing step. This is where we understand the data, the nuances of the dataset we have in our hand, and try to understand the circumstances in which this data has been collected. All this is necessary in order to ascertain facts that might help us better comprehend this data. We use this knowledge to come up with better notations for the data in hand so that the ML algorithms do not misunderstand variations in data for patterns which simply don't exist, or don't make any sense.

The second part of the architecture is covering the Machine Learning portion of the architecture which is also the core portion of our project. We have used a curated set of 7 ML algorithms which have been successful in many fields and have applied them here to analyse how they serve against this dataset. The results obtained from these algorithms have been stored, analysed and compared upon each other to decide upon the best candidate among the lot. This has been discussed in detail further down in this report.

The third part of this architecture details the Result validation portion which is also called as a testing phase or validation phase in Machine Learning terms. This is where the ML model trained upon the data is let to face an unseen set of data, and the model is analysed upon how well it works on these new sets. This serves as a factor in learning by probabilistic measure of how well it will serve on unseen data sets available in the real world.

Once we pass through all these stages of the architecture, we end up with a trained model with a good score capable of making good decisions on unseen data. We can then deploy this model on the real world and see the fruits of this work.

## VI. LIST OF MODULES

We will build our modular representation on the backs of the Architecture diagram drawn in the previous section. The list of modules will be based on the section of that architecture, and we shall take a look at those modules one by one. But before doing so, we will also cover the two sections of this project and branch the modules into the working parts of these sections.

The first section of this project deals with the Crime Stats of the country of India. As specified before this dataset is a combination of nearly 70 files each being a category of crime listing the crime stats for all the states and arranged by the year of crime and its state or region. We take in many scenarios or categories of crimes and derive conclusions based on them. To name a few, we observe the Human rights violations by the police, automotive and property theft, Juvenile crimes, anti corruption, firearm discharge cases. We have detailed the sub categories of these cases, their trends across the states in India, and also how the stats have been performing across the years. We believe that we can gain a general idea of  the crimes on a larger scale and can be used to test many hypotheses of the police of why there might be these varying trends in the crime stats.

The second section deals with the prediction of crimes in a much smaller region, here we perform a much detailed analysis, and also create a model which can make predictions with a good accuracy scores. In order to achieve this we go through a five stage process -- preprocessing, training, testing, validation, and result approval. We will go over each stage to cover more details on how this has been achieved. Here we perform predictions on three scenarios which we deem might serve as a beginning for future research. They are -- prediction the Federal UCR denoting the seriousness of the crime, the category of the crime itself, or the location where it might occur. All these have been modelled such that they work for predicting crimes that might occur in the future.

The First module is the **Pre-processing** of the dataset. The Boston Crime dataset has many parameters, and so we have used many techniques and intuition based mappings to convert this raw data into something understandable and relatable to the real world. As said before the data has to be converted such that we derive patterns designed on how the world works.One such step is to convert certain categorical data into multiple columns called as **One Hot Encoding**, which reduces the amount of correlation between the values of the same columns and prevents the ML algorithms from considering them to be patterns. We also need some general knowledge of the region of the crime occurrence in a few cases. For example, in the current case, we try to map the date and time of the crime to form a new parameter listing if it occurred during the daytime or in the night. This requires basic knowledge of the Daylight savings which alters the definition of day and night for a few months time. In some cases where we have many unique values the advantages of the One hot encoding do not outweigh its shortcomings in terms of the extra space incurred. In these cases, we just use **Labeled encodings** for these parameters to reduce complexity. This has been followed for many parameters in the dataset.

The next module is the **Training Phase**. In this phase, we use the **7 ML algorithms**, and train these models on the training dataset available to create a model. This is usually the longest stage because the models run through all the records of the dataset while it learns from its previous knowledge and the current record to correct itself to become better. Once the training is finished it would be able to give the closest best approximation for all.

The next module is the **testing and validation** stage. The whole dataset after preprocessing is split into a training set on which the above training process goes on, and a testing set  which is unseen by the model and is used to validate the model on unseen data which would give a true measure on how well the models will scale onto the real world. Once tested, the results are checked for overfitting, and in case an **overfitting** is observed the training process is repeated with a change in the parameters. This is a cyclic process and is repeated until the cases of over and **underfitting** are taken care of and we arrive at an optimal stage with a working set of models and their parameters.

The last module is the **Result analysis**, after all the models have been trained and validated, we are now left with the best possible model versions of the algorithm for this scenario. So we will now compare its results based on a few standard metrics like the **accuracy score**, **precision**, **recall**, **confusion matrix** which is a count of the rates of True positives, true negatives, false positives and false negatives, and the **F2 score** which is a combination of the precision and recall and gives a more balanced score with them as parameters.

# VII. ALGORITHMS

The Algorithms Section of our project comprises the ML backbone where we have experimented with various ML algorithms modelling and testing them in this scenario. We have used the Classification class of Machine Learning here in our problem, as we believe that we can model this into classifying crimes into certain categories like them belonging to a particular regional location, occurring during the day or night, belonging to a particular class of felonies, or any other categories that we deem fit in a scenario.

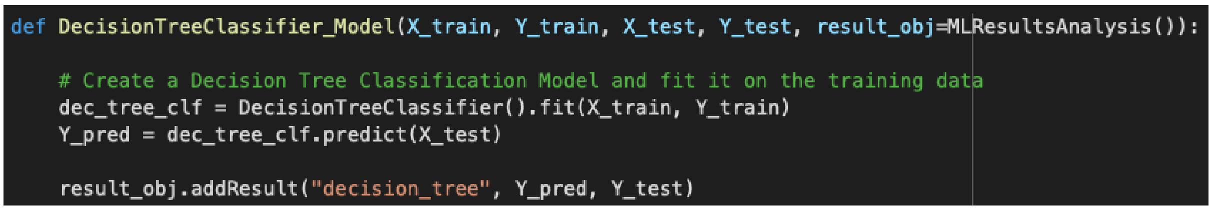
The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observations into a number of classes or groups. The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data. Some of the best examples of this are the spam classifiers in our mailboxes which classifies the mail to be spam or not. More specifically here we use a Multi-class Classifier algorithm, which refers to those classification tasks that have more than two class labels. Multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes. The number of class labels may be very large on some problems like in this current case.

We have listed down these Classification based ML algorithms along with their structural and the implementation details of how these algorithms have been written in the code repository. Each model has been implemented with taking in the train and test set and an object which it uses to store the results, which is later used for displaying and for comparisons between the models.

## 1. DECISION TREE CLASSIFIER --

The Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning. Decision Tree algorithms belong to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).



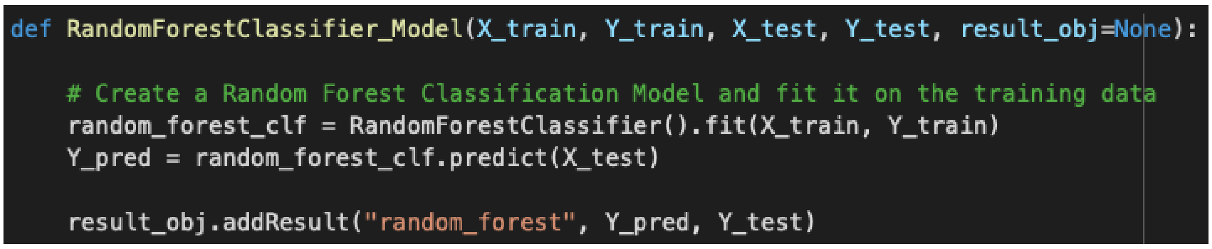
```
def DecisionTreeClassifier_Model(X_train, Y_train, X_test, Y_test, result_obj=MLResultsAnalysis()):

    # Create a Decision Tree Classification Model and fit it on the training data
    dec_tree_clf = DecisionTreeClassifier().fit(X_train, Y_train)
    Y_pred = dec_tree_clf.predict(X_test)

    result_obj.addResult("decision_tree", Y_pred, Y_test)
```

Figure 5 - Decision Tree Model

## 2. RANDOM FORESTS CLASSIFIER --

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Random Forest Classifiers facilitate the reduction in the over-fitting of the model and these classifiers are more accurate than the decision trees in several cases. Random forests exhibit real-time prediction but that is slow in nature. They are also difficult to implement and have a complex algorithm.

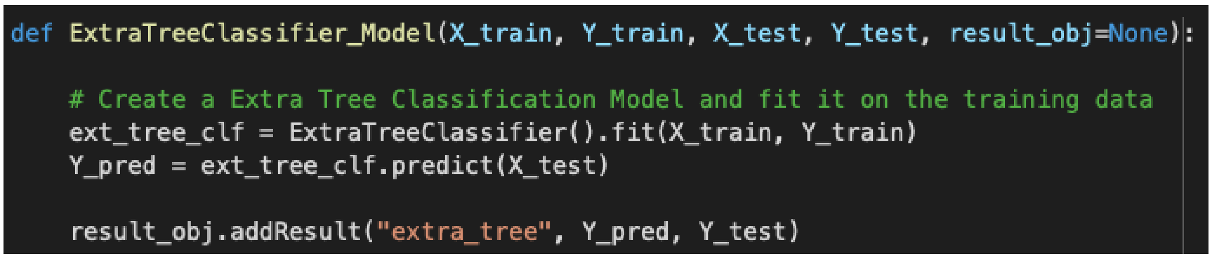


```
def RandomForestClassifier_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a Random Forest Classification Model and fit it on the training data
    random_forest_clf = RandomForestClassifier().fit(X_train, Y_train)
    Y_pred = random_forest_clf.predict(X_test)

    result_obj.addResult("random_forest", Y_pred, Y_test)
```

Figure 6 - Random Forest Classifier Model

## 3. EXTRA TREE CLASSIFIER --

These are also called **Extremely Randomized Trees**, and are a variation of the Random Forest Classifier explained above. To introduce more variation into the ensemble, we will change how we build trees. Each decision stump will be built with the following criteria: All the data available in the training set is used to build each stump. To form the root node or any node, the best split is determined by searching in a subset of randomly selected features of size sqrt(number of features). The split of each selected feature is chosen at random. The maximum depth of the decision stump is one. Since splits are chosen at random for each feature in the Extra Trees Classifier, it's less computationally expensive than a Random Forest.

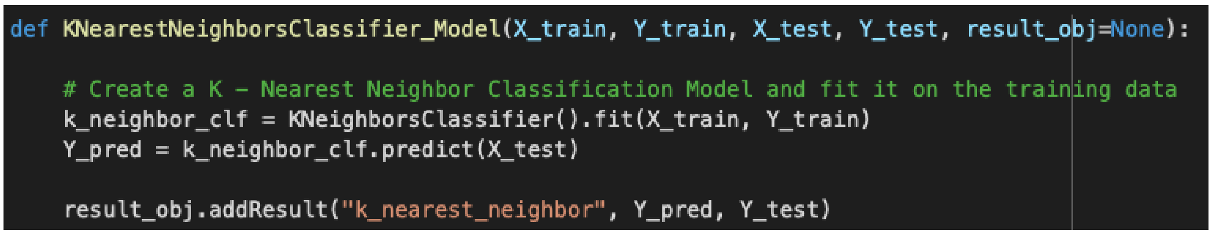


```
def ExtraTreeClassifier_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a Extra Tree Classification Model and fit it on the training data
    ext_tree_clf = ExtraTreeClassifier().fit(X_train, Y_train)
    Y_pred = ext_tree_clf.predict(X_test)

    result_obj.addResult("extra_tree", Y_pred, Y_test)
```

Figure 7 - Extra Tree Classifier Model

## 4. K NEAREST NEIGHBOR CLASSIFIER --

KNNs belong to the supervised learning domain and have several applications in pattern recognition, data mining, and intrusion detection. These KNNs are used in real-life scenarios where non-parametric algorithms are required. These algorithms do not make any assumptions about how the data is distributed. When we are given prior data, the KNN classifies the coordinates into groups that are identified by a specific attribute.

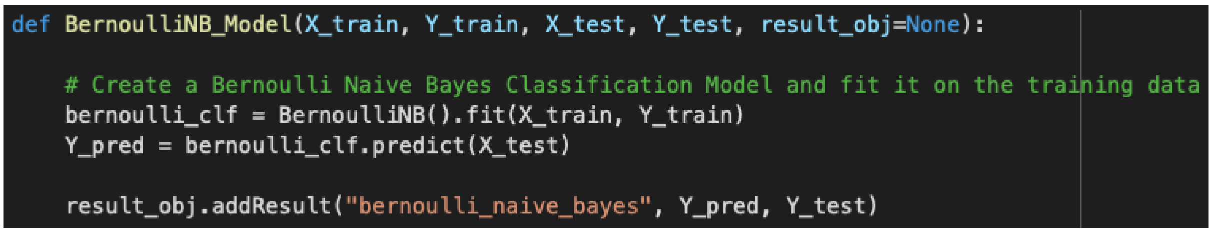


```
def KNearestNeighborsClassifier_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a K - Nearest Neighbor Classification Model and fit it on the training data
    k_neighbor_clf = KNeighborsClassifier().fit(X_train, Y_train)
    Y_pred = k_neighbor_clf.predict(X_test)

    result_obj.addResult("k_nearest_neighbor", Y_pred, Y_test)
```

Figure 8 - K Nearest Neighbor Classifier Model

## 5. BERNOULLI NAIVE BAYES CLASSIFIER --

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not. This is mainly used for text analysis, more appropriately in text classification with the 'bag of words' model.



```
def BernoulliNB_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a Bernoulli Naive Bayes Classification Model and fit it on the training data
    bernoulli_clf = BernoulliNB().fit(X_train, Y_train)
    Y_pred = bernoulli_clf.predict(X_test)

    result_obj.addResult("bernoulli_naive_bayes", Y_pred, Y_test)
```

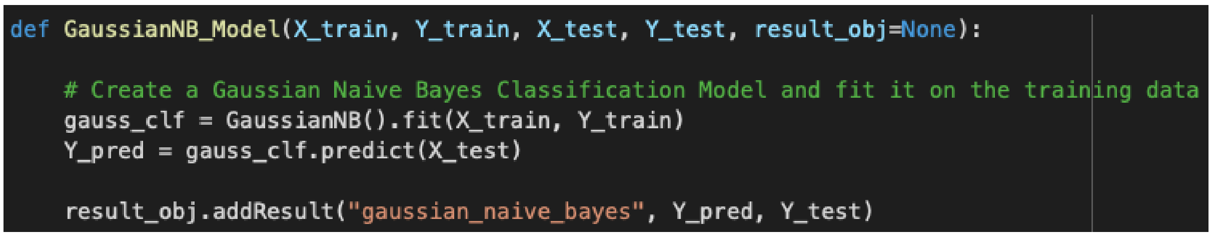
Figure 9 - Bernoulli Naive Bayes Classifier Model

## 6. GAUSSIAN NAIVE BAYES CLASSIFIER --

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution. Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution. An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all that is needed to define such a distribution.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Figure 10 - Gaussian Naive Bayes Formula



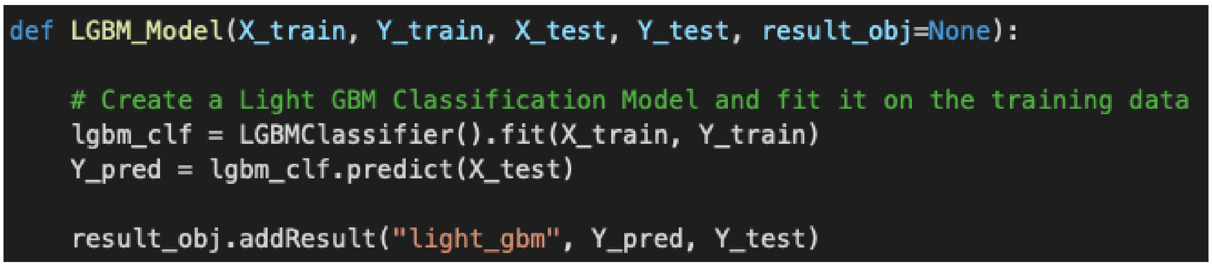
```python
def GaussianNB_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a Gaussian Naive Bayes Classification Model and fit it on the training data
    gauss_clf = GaussianNB().fit(X_train, Y_train)
    Y_pred = gauss_clf.predict(X_test)

    result_obj.addResult("gaussian_naive_bayes", Y_pred, Y_test)
```

Figure 11 - Gaussian Naive Bayes Classifier Model

## 7. LIGHT GBM CLASSIFIER --

LightGBM is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduce memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks, which form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.

LightGBM splits the tree leaf-wise as opposed to other boosting algorithms that grow tree level-wise. It chooses the leaf with maximum delta loss to grow. Since the leaf is fixed, the leaf-wise algorithm has lower loss compared to the level-wise algorithm. Leaf-wise tree growth might increase the complexity of the model and may lead to overfitting in small datasets.



```python
def LGBM_Model(X_train, Y_train, X_test, Y_test, result_obj=None):

    # Create a Light GBM Classification Model and fit it on the training data
    lgbm_clf = LGBMClassifier().fit(X_train, Y_train)
    Y_pred = lgbm_clf.predict(X_test)

    result_obj.addResult("light_gbm", Y_pred, Y_test)
```

Figure 12 - Light GBM Classifier Model

## VIII. RISK ANALYSIS

There are many risks associated with the implementation and bringing a project live into the real world, and this one is no exception. We have to address a lot of concerns that might raise in a lot of fields like data collection regarding the maintenance of the privacy of the individuals involved in these reports, and security of this data so that it does not fall into any unauthorized hands. So we are in a need for a standardized strategy for the collection, storage, maintenance and disposal of the data involved in this project.

We will also be defining the **Minimum Viable Product(MVP),** which says "What's the minimum we need for the capability to actually be useful?". The role of the data scientist in this process is then to convert business requirements into quantitative ML system requirements/metrics.The metrics selection process should also enable the team to determine what kinds of ML solutions might be appropriate. This process requires the data scientist to straddle two roles: representing stakeholder interests and guiding engineering efforts. Metrics that assess performance of an ML component go beyond an accuracy constraint, to account for qualities such as asymmetric costs of false positives vs. false negatives, model size and memory requirements, training and inference time, and model interpretability, most of which have been covered in the results section below. Getting the requirements wrong is a risk in software engineering, but its likelihood is elevated in ML engineering.

Engineering an ML system consumes too much time and too many resources. A consequential difference between traditional software systems and ML systems is that ML systems are dependent on data. Acquiring data is expensive, and large tech companies that can stomach the cost will often employ sizable teams to do nothing but expertly label data. After large data sets are acquired, they are technically challenging and expensive to maintain, and they demand intense computing resources. These factors raise staffing costs by imposing a need for data engineering, math, and statistics skills in addition to software engineering. ML systems are also usually more expensive to debug and maintain than traditional software systems. One reason is that ML systems are prone to normal software bugs, but also flaws like over- or under-fitting, difficulty training due to exploding or vanishing gradients, or learning something that satisfies an optimization metric but violates the engineer's intent. These flaws cannot always be found with a debugger, and diagnosing them requires data science knowledge.

ML Research programming has generally three challenges:  Progress can be slow and expensive due to experimentation iterations required for understanding the capabilities of an ML model, progress can be hard to predict which makes budgeting hard, and the next stage would often depend on the results of the current stage which makes planning uncertain. We also might get into unintended troubles when adding softwares with these models, like vulnerability to adversarial examples, interactions between ML models and software systems, feedback loops which can propagate biases in training data, and others. It may be possible to mitigate these risks to an extent by vetting the training data, adding features to the ML system that guide safe behavior, doing additional quality control and testing prior to release, and carefully maintaining a live model. But these additional costs should be taken into consideration. These can be avoided by routinely updating the data, assessing the data for bias, imbalance, and representativeness, and retrain the models. The price paid is a reduced iteration speed. Discrepancies between development and production environments (not just the datasets) between two versions of the training data and between the mission and model's objective function can lead to undesirable results and expensive debugging.

We have discussed a few key areas where we must concentrate upon while deploying this model into the real world. But there are many other risks especially in a project such as ours which would deal with bad actors at all times, we need to make sure that the system is tamper proof such that the models are not tampered with to satisfy any private goals, such a case would only breed catastrophe over the whole systems, such securities can be implemented by using secure hashing and encryption techniques for storing the model and its weights securely such that it is inaccessible and thus is not controllable by any individual without proper authority.

## IX. IMPLEMENTATION

We have used Google Colab environment for all our experiments. Since colab is already installed with basic python and it's corresponding libraries that ease up our work. The main libraries we used for our implementation are - Numpy, Pandas and SKlearn. We have used Matplotlib for the figures wherever any visualization was needed to support our work. We also used the Geopanda library that facilitates us to work with maps and their partitions. It is an external package and needs to be installed in the colab environment.

As one said the best way to work around any data analytics work is to first know the data you are working with. So, we first took an Indian dataset for crime and did a thorough analysis over the dataset with visualizations through many aspects. We will in the next subsection below describe some of the graphs that we plotted.

### VISUALIZATION

We mostly did our visualization on the indian dataset to set a theory regarding the crimes and their relation with time and geographical location of the person. We took a dataset of the crime and juvenile prison and plotted them across many related parameters. All the plots were plotted using either geopanda or matplotlib.

1. **HUMAN RIGHT VIOLATION**

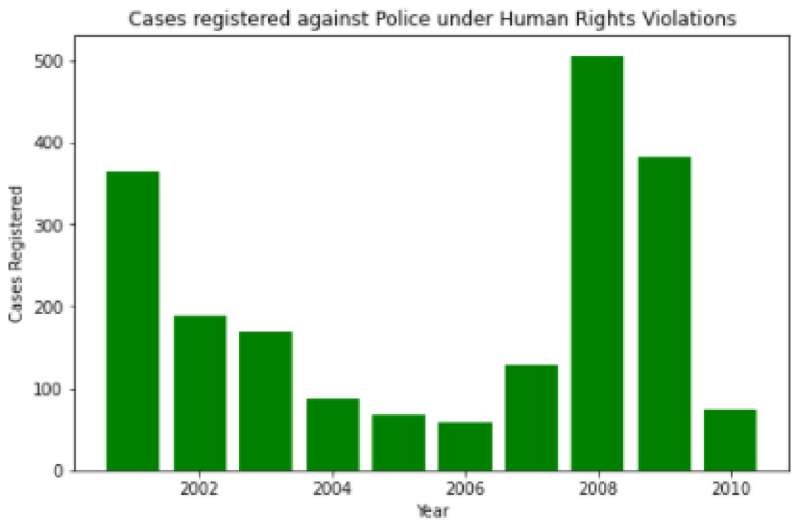


Figure 13 - Plot for Cases registered against Police under Human Rights Violation

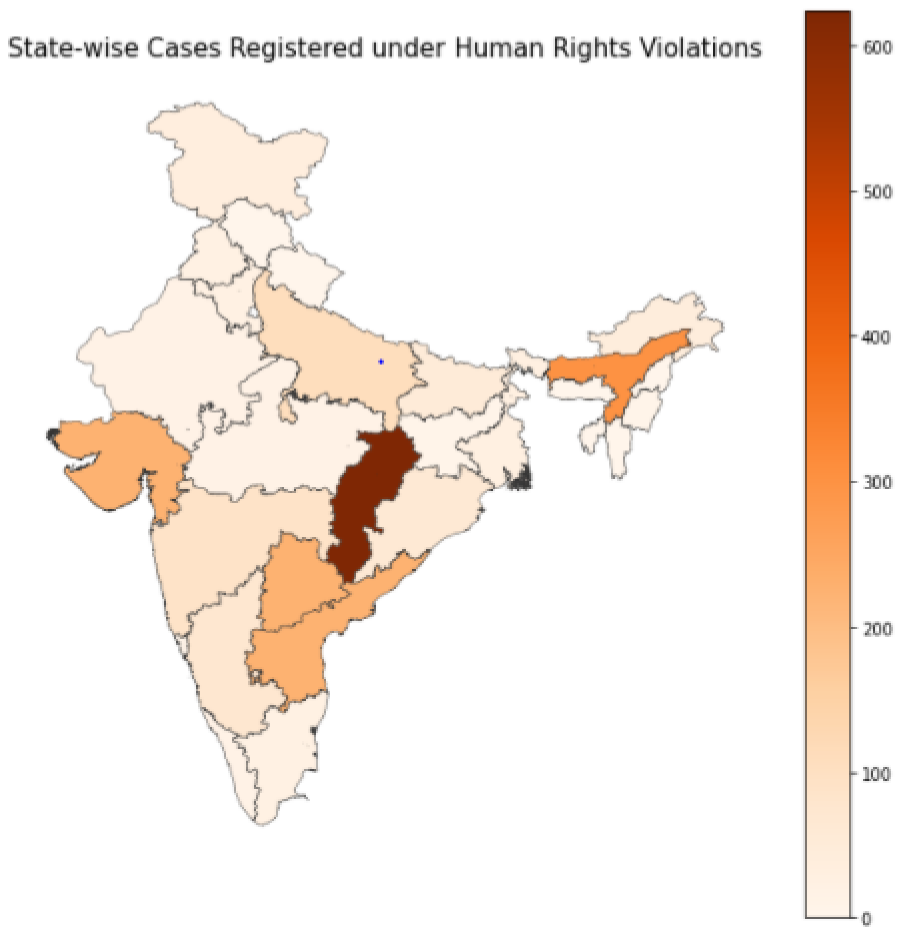


Figure 14 - State wise cases registered against Police under Human Rights Violation

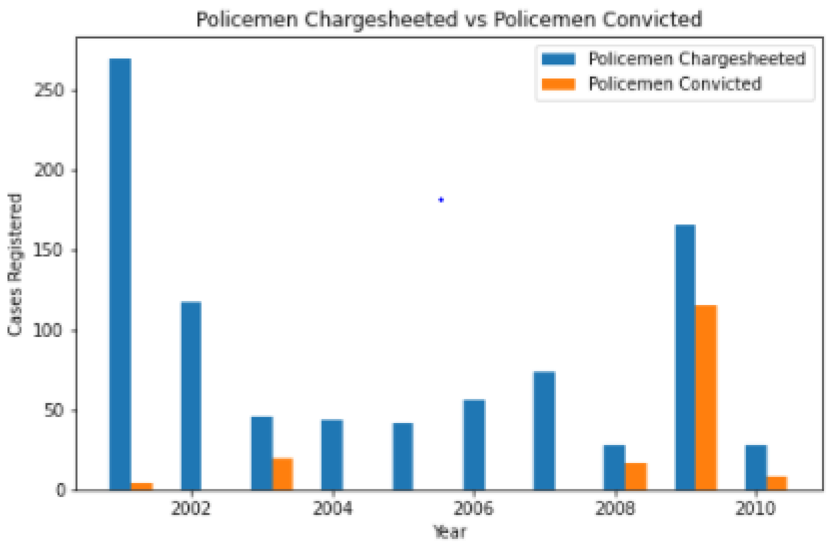


Figure 15 - Poliecemen Chargesheeted vs Convicted

## 2. VEHICLE STOLEN CASES

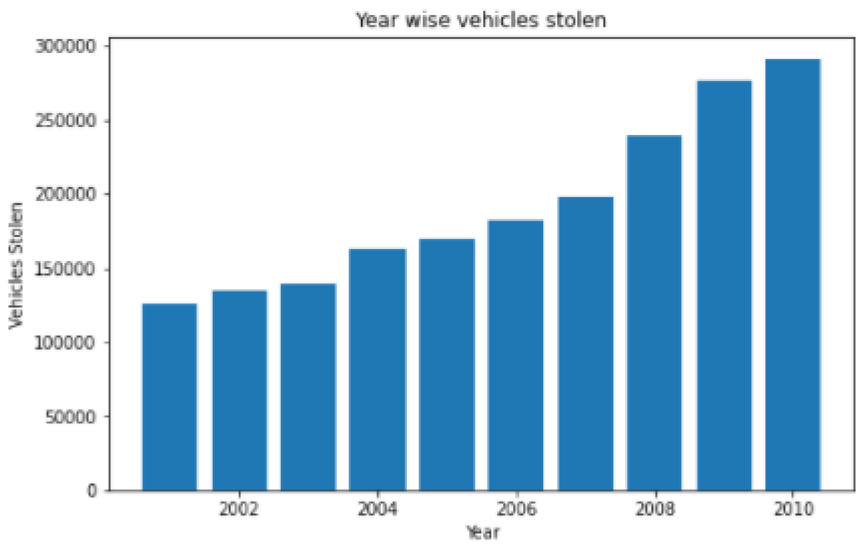


Figure 16 - Year Wise Vehicle stolen

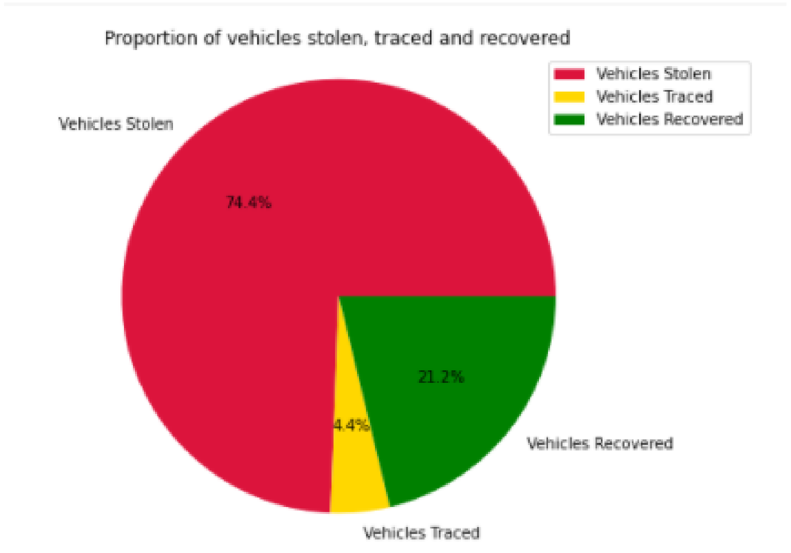


Figure 17 - Proportion of vehicles stolen

The Figure17 shows a Pie chart showing data about the report from the cases related to vehicle stealing. We can see that out of the 74.4% stolen vehicle we have only 4.4% vehicle traces and 21.2% of the vehicle recovered. This data shows that the recovery rate after a vehicle goes missing is very low.

## 3. STOLEN PROPERTY CASES

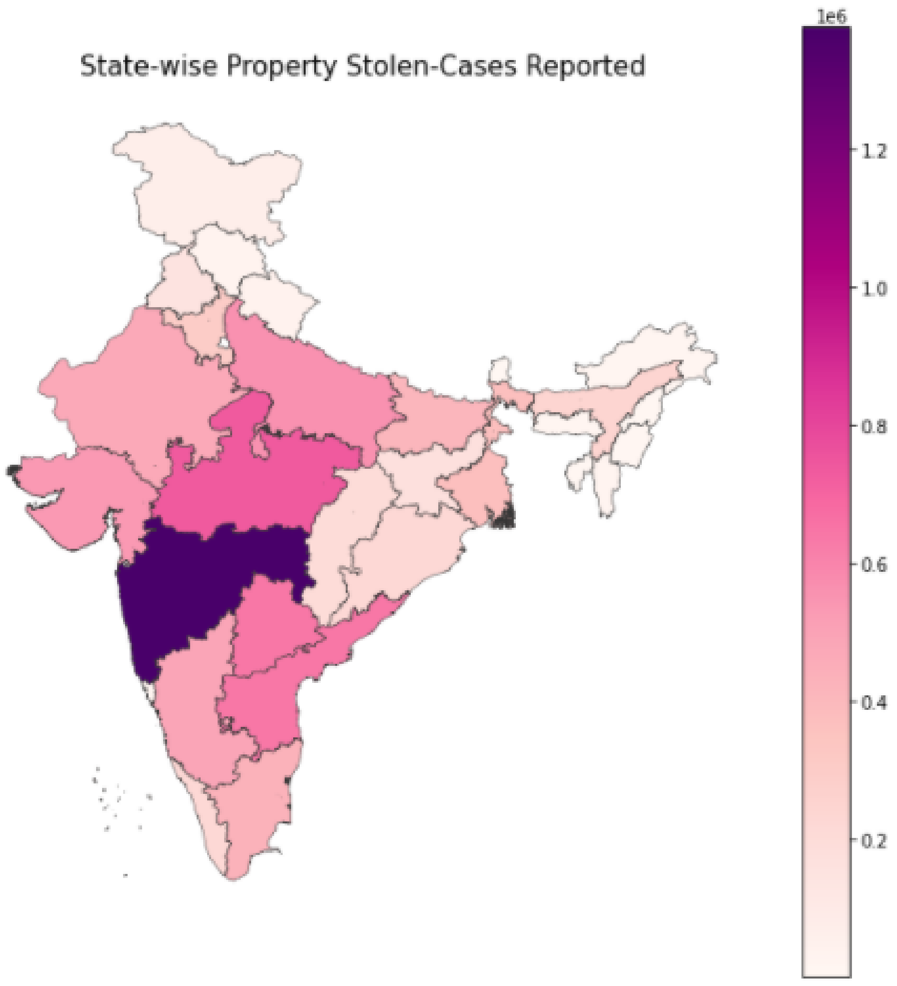


Figure 18 - State wise Property Theft

The map in Figure 18 is a heatmap representation of the state wise property stolen cases reported. It is plotted with the help of geopandas. From the figure we can conclude that the cases are quite higher in the states of Maharashtra and Madhya Pradesh. Also we can see from the figure that the cases are quite low in north indian states as well as the eastern part of the country.

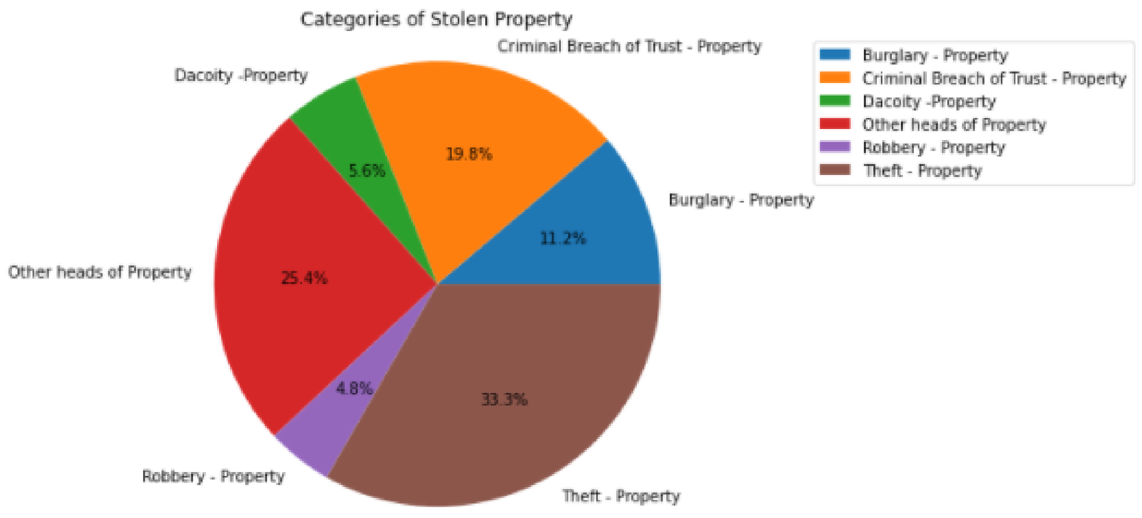


Figure 19 - Categories of Stolen Property

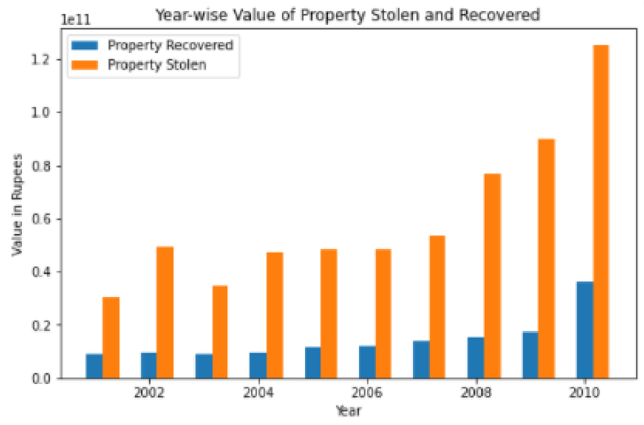


Figure 20 - Year wise Property theft stolen vs recovered

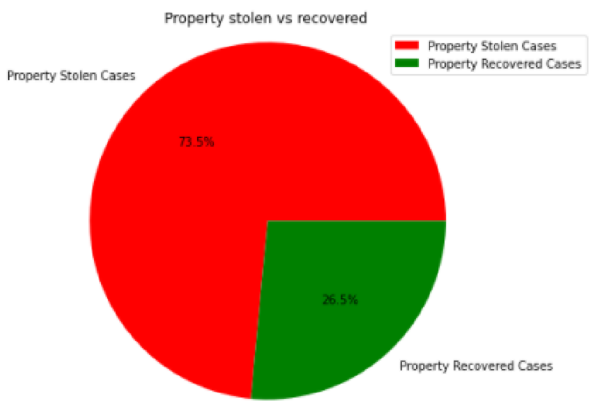


Figure 21 - Property

The Pie Chart in Figure shows that there are a high number of property stolen cases that are unsolved.

## 4. JUVENILE ARREST DATASET

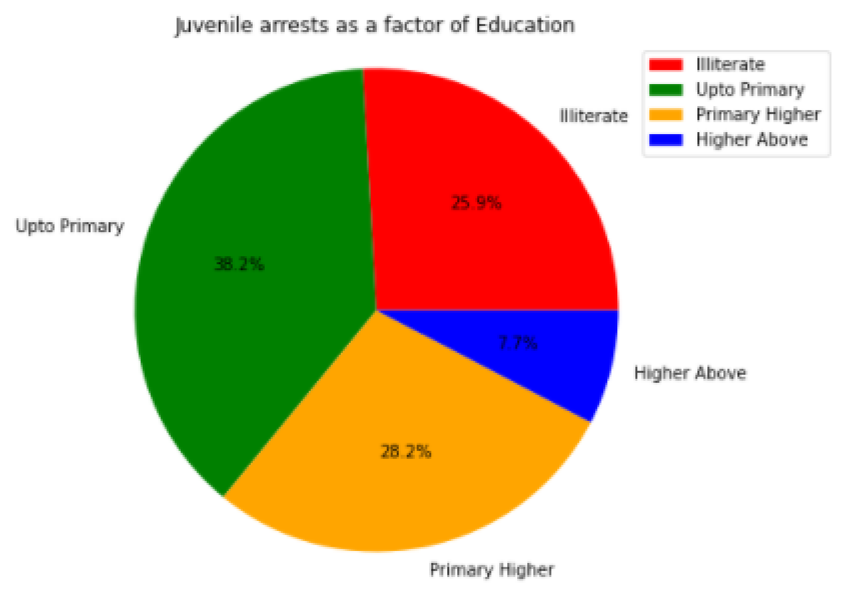


Figure 22 Juvenile arrests as a factor of education

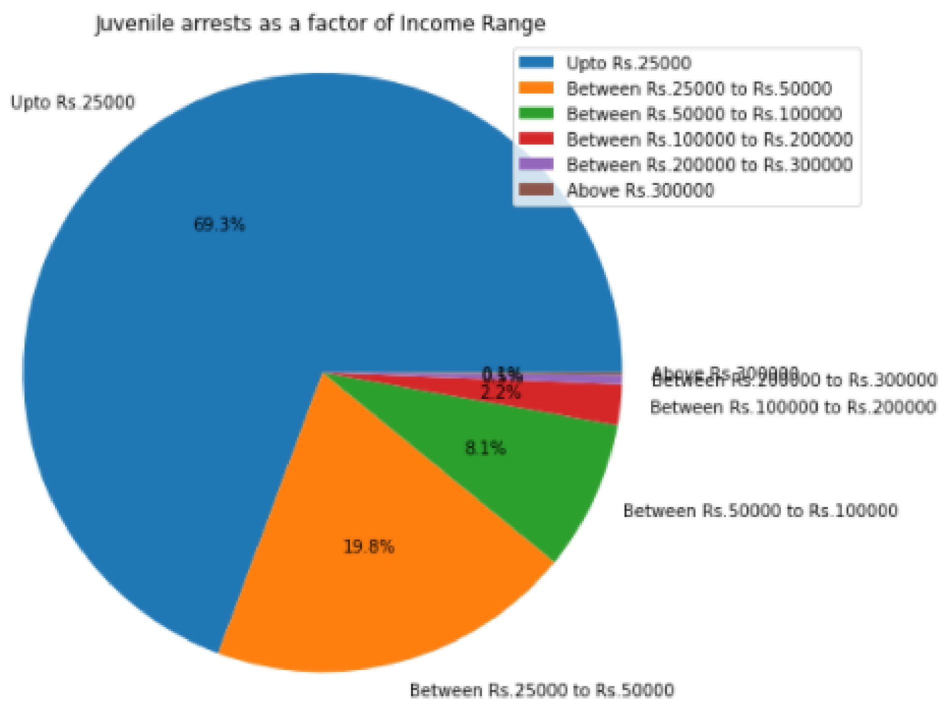


Figure 23 Juvenile arrests as a factor of Income Range

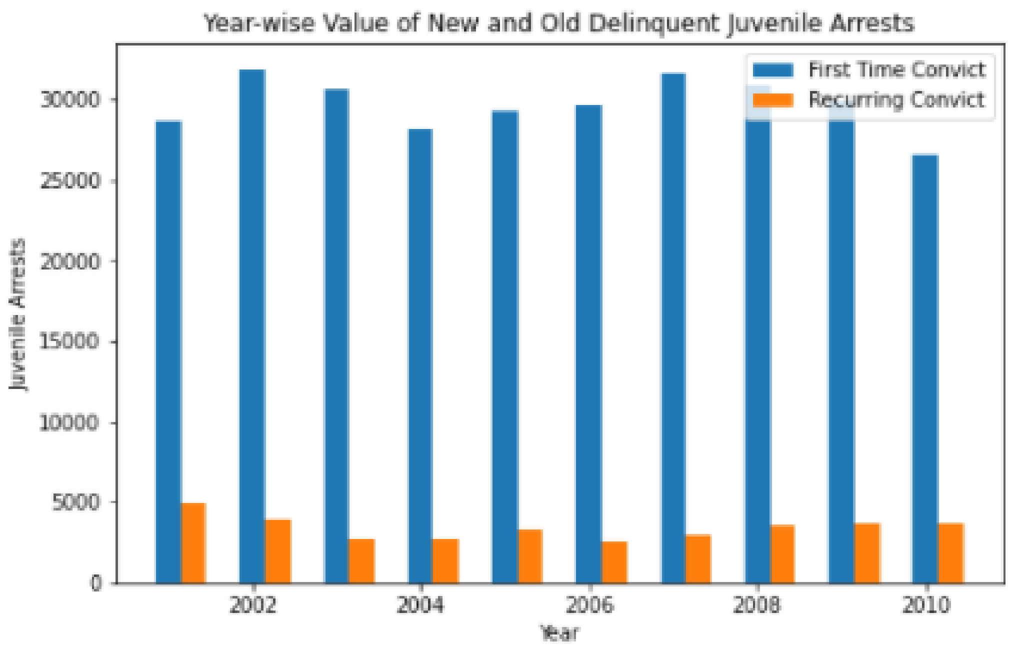
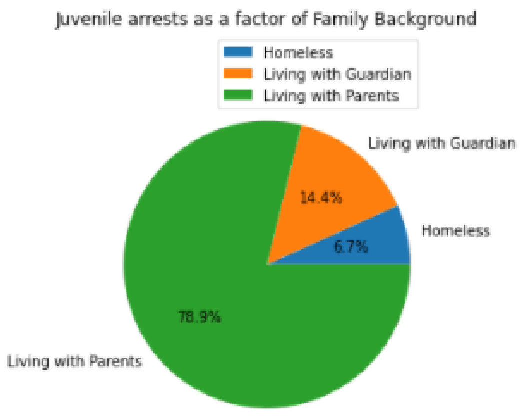


Figure 24 Year wise Juvenile Data



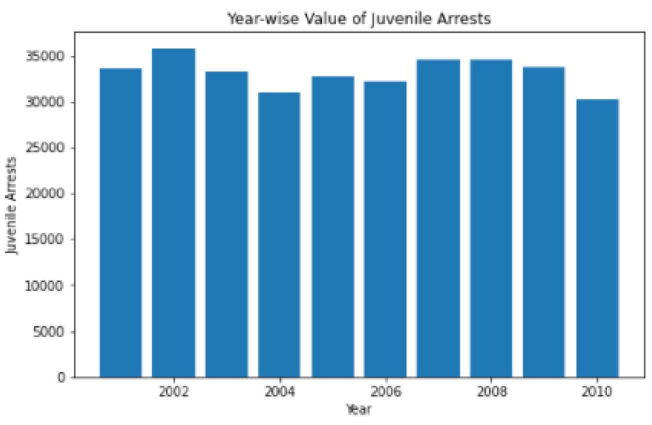
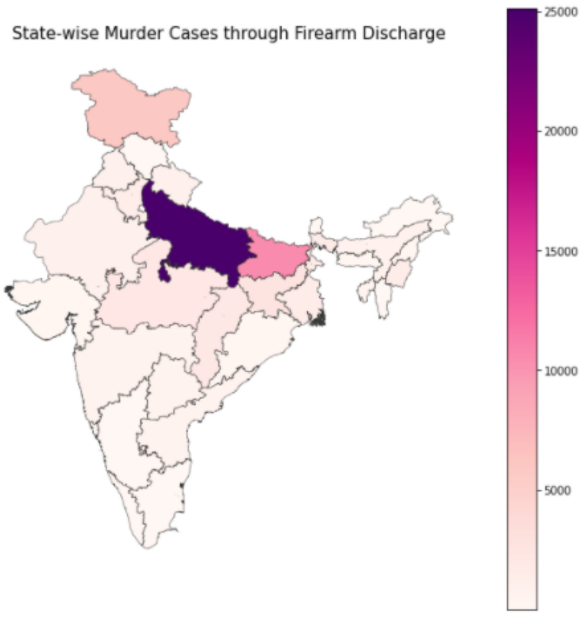


Figure 25 Juvenile as family background Figure 26 Yearly juvenile data

We can see from the dataset that juvenile arrest is an important factor that leads to future crimes. Also from Figure 25 we can see that 79% of the juvenile arrests were children who lived with parents which quite contradicts the general perception of homeless children to be convicted. Figure 26 tells us that the rate of juvenile crimes is quite consistent despite just a slight dip in 2010. Figure 23 tells that the income need is the main reason behind the crime by any children below 18.

## 5. FIREARM USAGE IN CRIME CASES



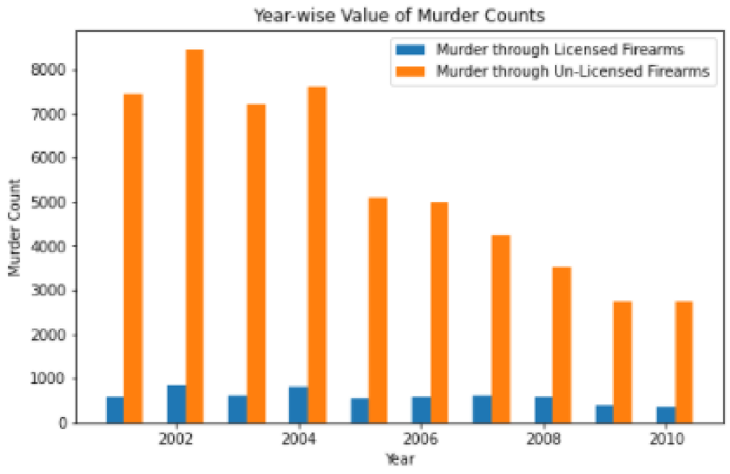


Figure 27 Statewise firearms data Figure 28 Year wise murder data

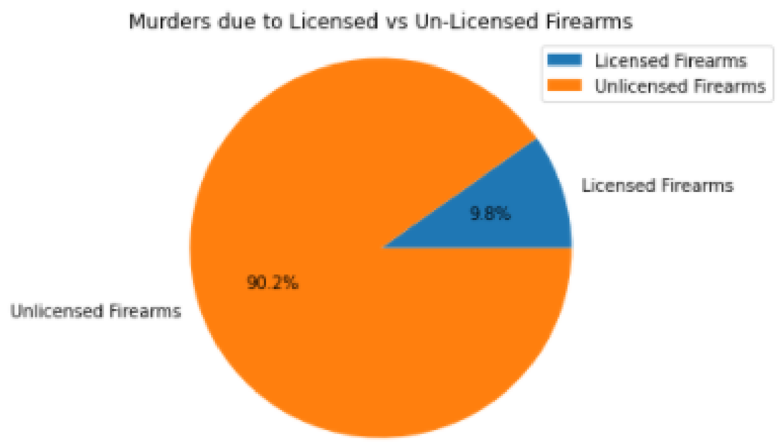
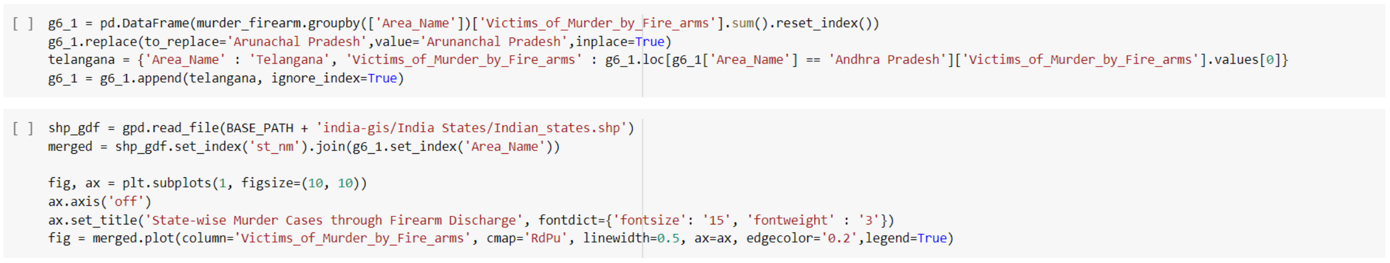


Figure 29 Murders due to licensed vs unlicensed firearms

Figure 27 - 29 describe how the Firearms usages are related to any crime cases. From Figure 27 it is well understood that the firearms are quite easily available in the state of Uttar Pradesh hence it is widely used in a crime case. Figure 29 tells us that most of the firearms used in a murder case are unlicensed. The number of murders through firearms though have come down significantly in recent years which may be the cause of stricter rules and regulations imposed on usage and availability of firearms.

## SAMPLE CODE FOR VISUALIZATION -

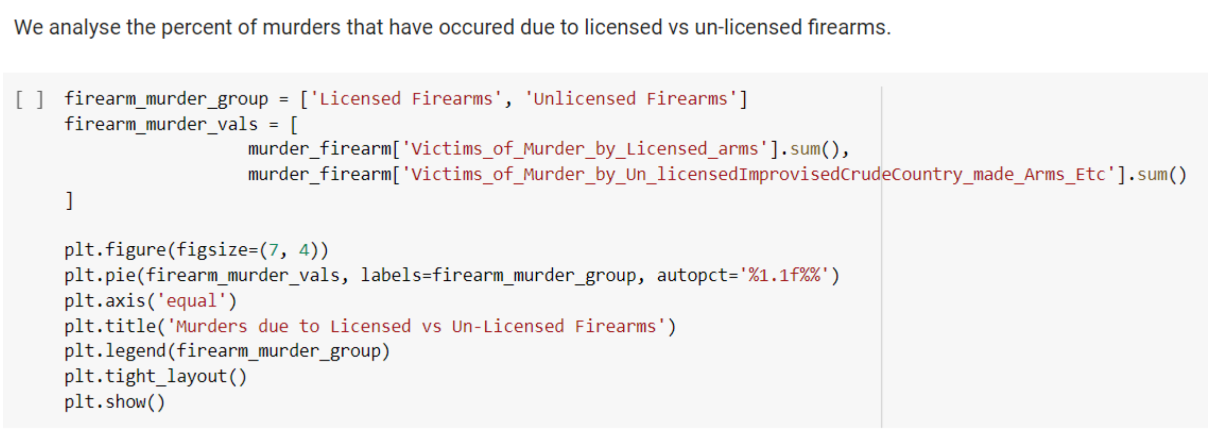
In this section, we will detail the code that allowed us to do the above interactive analysis.



```
[ ]  g6_1 = pd.DataFrame(murder_firearm.groupby(['Area_Name'])['Victims_of_Murder_by_Fire_arms'].sum().reset_index())
     g6_1.replace(to_replace='Arunachal Pradesh',value='Arunanchal Pradesh',inplace=True)
     telangana = {'Area_Name' : 'Telangana', 'Victims_of_Murder_by_Fire_arms' : g6_1.loc[g6_1['Area_Name'] == 'Andhra Pradesh']['Victims_of_Murder_by_Fire_arms'].values[0]}
     g6_1 = g6_1.append(telangana, ignore_index=True)
```

```
[ ]  shp_gdf = gpd.read_file(BASE_PATH + 'india-gis/India States/Indian_states.shp')
     merged = shp_gdf.set_index('st_nm').join(g6_1.set_index('Area_Name'))

     fig, ax = plt.subplots(1, figsize=(10, 10))
     ax.axis('off')
     ax.set_title('State-wise Murder Cases through Firearm Discharge', fontdict={'fontsize': '15', 'fontweight' : '3'})
     fig = merged.plot(column='Victims_of_Murder_by_Fire_arms', cmap='RdPu', linewidth=0.5, ax=ax, edgecolor='0.2',legend=True)
```

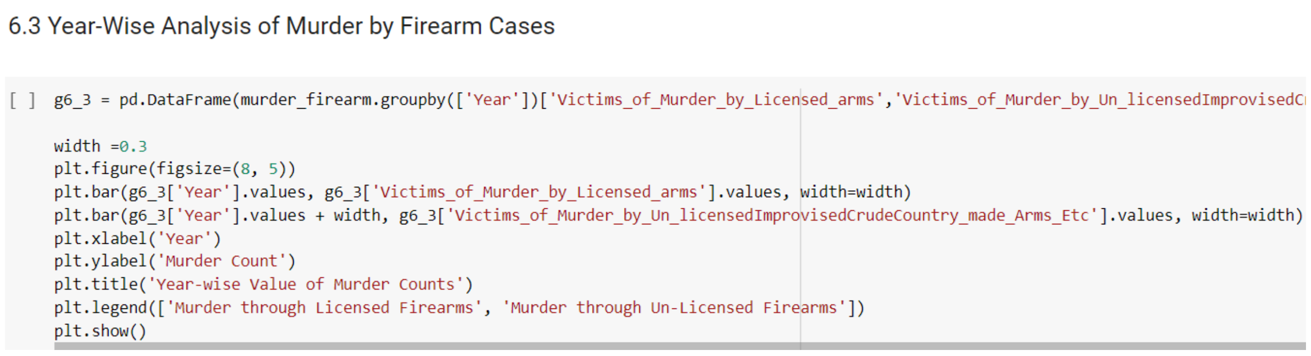
Figure 30 - Visualization code for statewise murder through fire discharge



We analyse the percent of murders that have occured due to licensed vs un-licensed firearms.

```
[ ]  firearm_murder_group = ['Licensed Firearms', 'Unlicensed Firearms']
     firearm_murder_vals = [
                     murder_firearm['Victims_of_Murder_by_Licensed_arms'].sum(),
                     murder_firearm['Victims_of_Murder_by_Un_licensedImprovisedCrudeCountry_made_Arms_Etc'].sum()
     ]

     plt.figure(figsize=(7, 4))
     plt.pie(firearm_murder_vals, labels=firearm_murder_group, autopct='%1.1f%%')
     plt.axis('equal')
     plt.title('Murders due to Licensed vs Un-Licensed Firearms')
     plt.legend(firearm_murder_group)
     plt.tight_layout()
     plt.show()
```

Figure 31 - Visualization code for pie chart for firearms cases



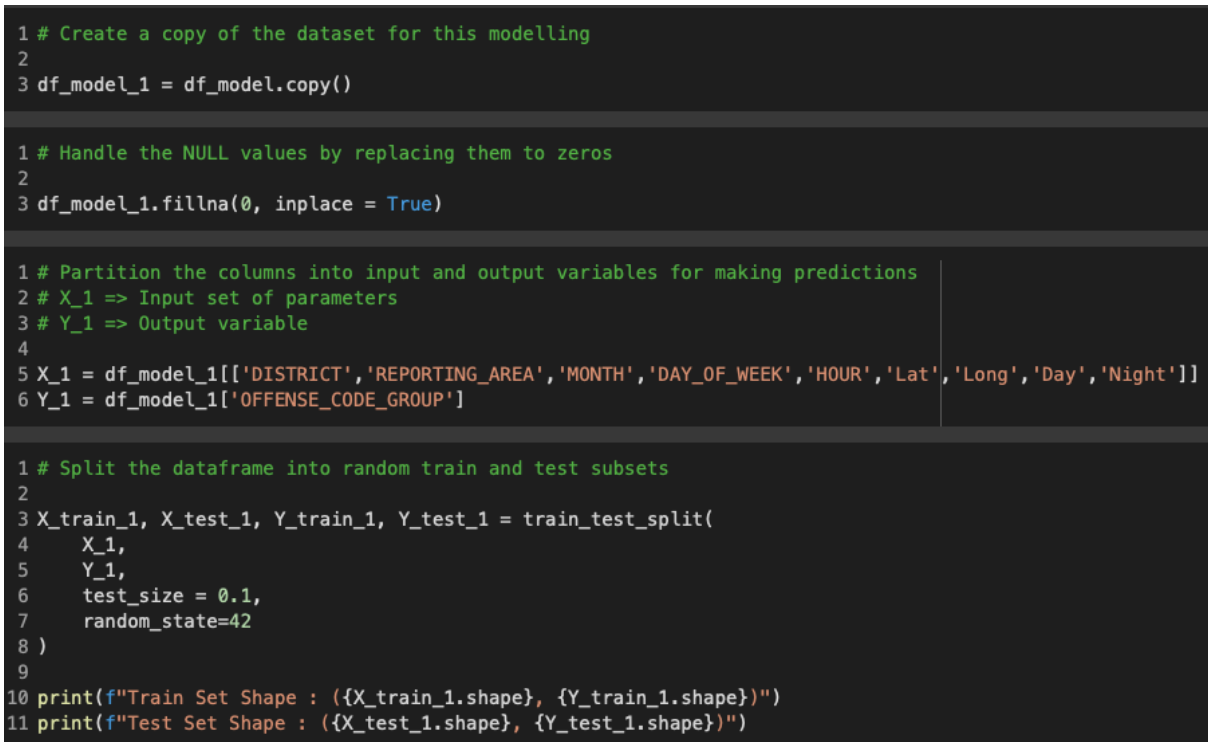
6.3 Year-Wise Analysis of Murder by Firearm Cases

```
[ ]  g6_3 = pd.DataFrame(murder_firearm.groupby(['Year'])['Victims_of_Murder_by_Licensed_arms','Victims_of_Murder_by_Un_licensedImprovisedC

     width =0.3
     plt.figure(figsize=(8, 5))
     plt.bar(g6_3['Year'].values, g6_3['Victims_of_Murder_by_Licensed_arms'].values, width=width)
     plt.bar(g6_3['Year'].values + width, g6_3['Victims_of_Murder_by_Un_licensedImprovisedCrudeCountry_made_Arms_Etc'].values, width=width)
     plt.xlabel('Year')
     plt.ylabel('Murder Count')
     plt.title('Year-wise Value of Murder Counts')
     plt.legend(['Murder through Licensed Firearms', 'Murder through Un-Licensed Firearms'])
     plt.show()
```

Figure 32 - Visualization code for Year-Wise Murder with Firearms.

## CODE FOR ML MODELLING -

In this section, we will deal with the code which is used to call upon the ML algorithms discussed earlier and see how we have programmed this procedure for one scenario. Here the "df_model" is the dataset after pre-processing steps.



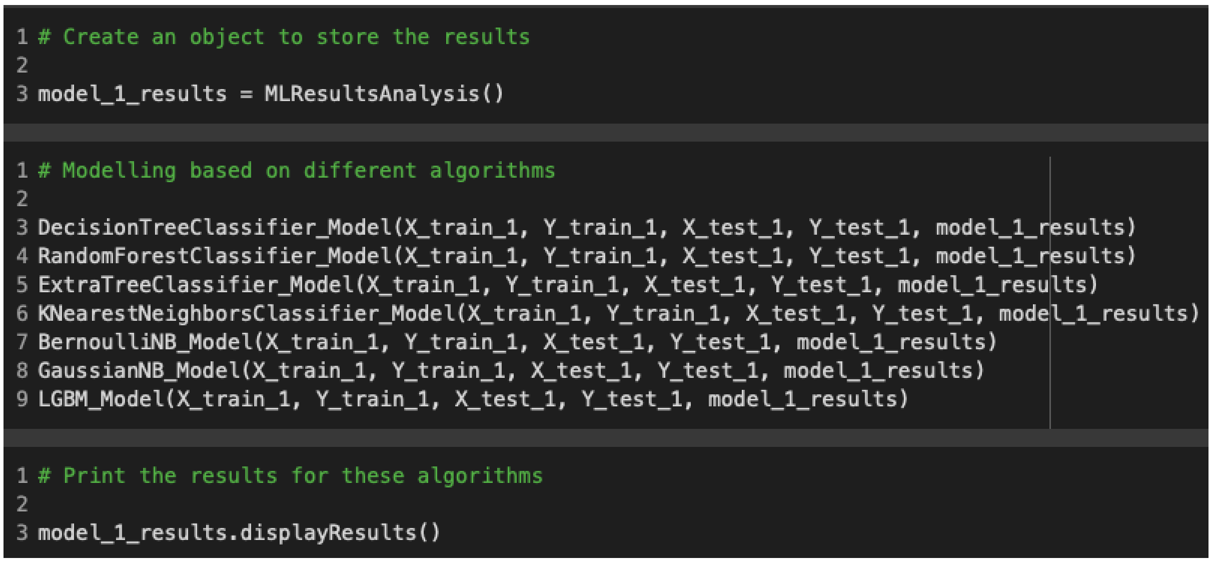
```python
# Create a copy of the dataset for this modelling

df_model_1 = df_model.copy()
```

```python
# Handle the NULL values by replacing them to zeros

df_model_1.fillna(0, inplace = True)
```

```python
# Partition the columns into input and output variables for making predictions
# X_1 => Input set of parameters
# Y_1 => Output variable

X_1 = df_model_1[['DISTRICT','REPORTING_AREA','MONTH','DAY_OF_WEEK','HOUR','Lat','Long','Day','Night']]
Y_1 = df_model_1['OFFENSE_CODE_GROUP']
```

```python
# Split the dataframe into random train and test subsets

X_train_1, X_test_1, Y_train_1, Y_test_1 = train_test_split(
    X_1,
    Y_1,
    test_size = 0.1,
    random_state=42
)

print(f"Train Set Shape : ({X_train_1.shape}, {Y_train_1.shape})")
print(f"Test Set Shape : ({X_test_1.shape}, {Y_test_1.shape})")
```

Figure 33 (a) - Stage 1 of predicting the offence categories



```python
# Create an object to store the results

model_1_results = MLResultsAnalysis()
```

```python
# Modelling based on different algorithms

DecisionTreeClassifier_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
RandomForestClassifier_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
ExtraTreeClassifier_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
KNearestNeighborsClassifier_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
BernoulliNB_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
GaussianNB_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
LGBM_Model(X_train_1, Y_train_1, X_test_1, Y_test_1, model_1_results)
```

```python
# Print the results for these algorithms

model_1_results.displayResults()
```

Figure 33 (b) - Stage 2 of predicting the offence categories

This code uses a copy of the model to create models, store and display the results.

## COMPARISON OF RESULTS -

We have used the confusion matrix to compare the different performance of the machine learning models that we are using.The confusion matrix provides us a matrix/table as output and describes the performance of the model.It is also known as the error matrix.The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like the table below.

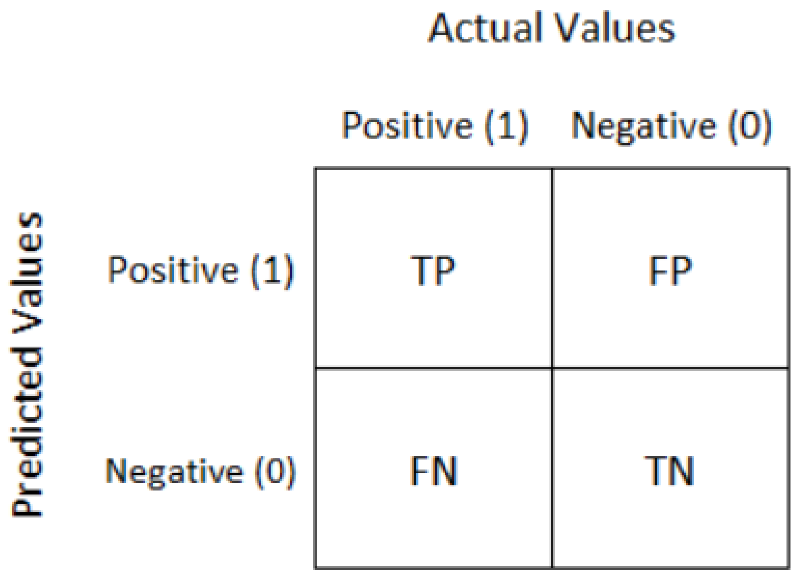


Figure 34 - Confusion matrix

**True Positives (TP)** - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if the actual class value indicates that this passenger survived and the predicted class tells you the same thing.

**True Negatives (TN)** - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. If the actual class says this passenger did not survive and the predicted class tells you the same thing.

False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

**False Positives (FP)** – When actual class is no and predicted class is yes. E.g. if the actual class says this passenger did not survive but the predicted class tells you that this passenger will survive.

**False Negatives (FN)** – When actual class is yes but predicted class in no. E.g. if the actual class value indicates that this passenger survived and the predicted class tells you that passenger will die.

Using the above confusion matrix we have derived the following performance measuring techniques-

**Accuracy** - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positives and false negatives are almost the same. Therefore, you have to look at other parameters to evaluate the performance of your model.

$$\textbf{\textit{Accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 35 - Accuracy Formula

**Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answers is of all passengers that are labeled as survived, how many actually survived? High precision relates to the low false positive rate.

$$Precision = \frac{TP}{TP + FP}$$

Figure 36 - Precision formula

**Recall** (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label?

$$Recall = \frac{TP}{TP + FN}$$

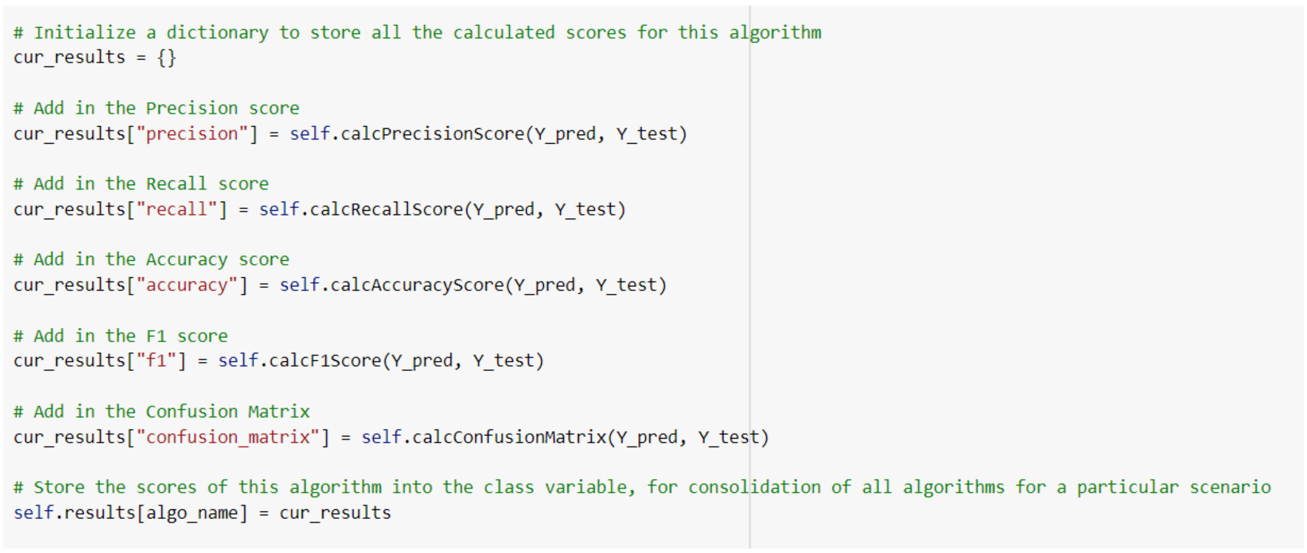
Figure 37 - Recall formula

**F1 score** - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Figure 38 - F1 score formula

All these parameters are called performance metrics, we have made a custom class and used SKlearn pre made metrics measurement functions to calculate the different performance metrics. The code implementation of the metrics can be seen in figure 38. There is a unique significance of each of the metrics on it's own. The results for the running of each of the classification algorithms for these performance metrics is shown in the results section below this section.



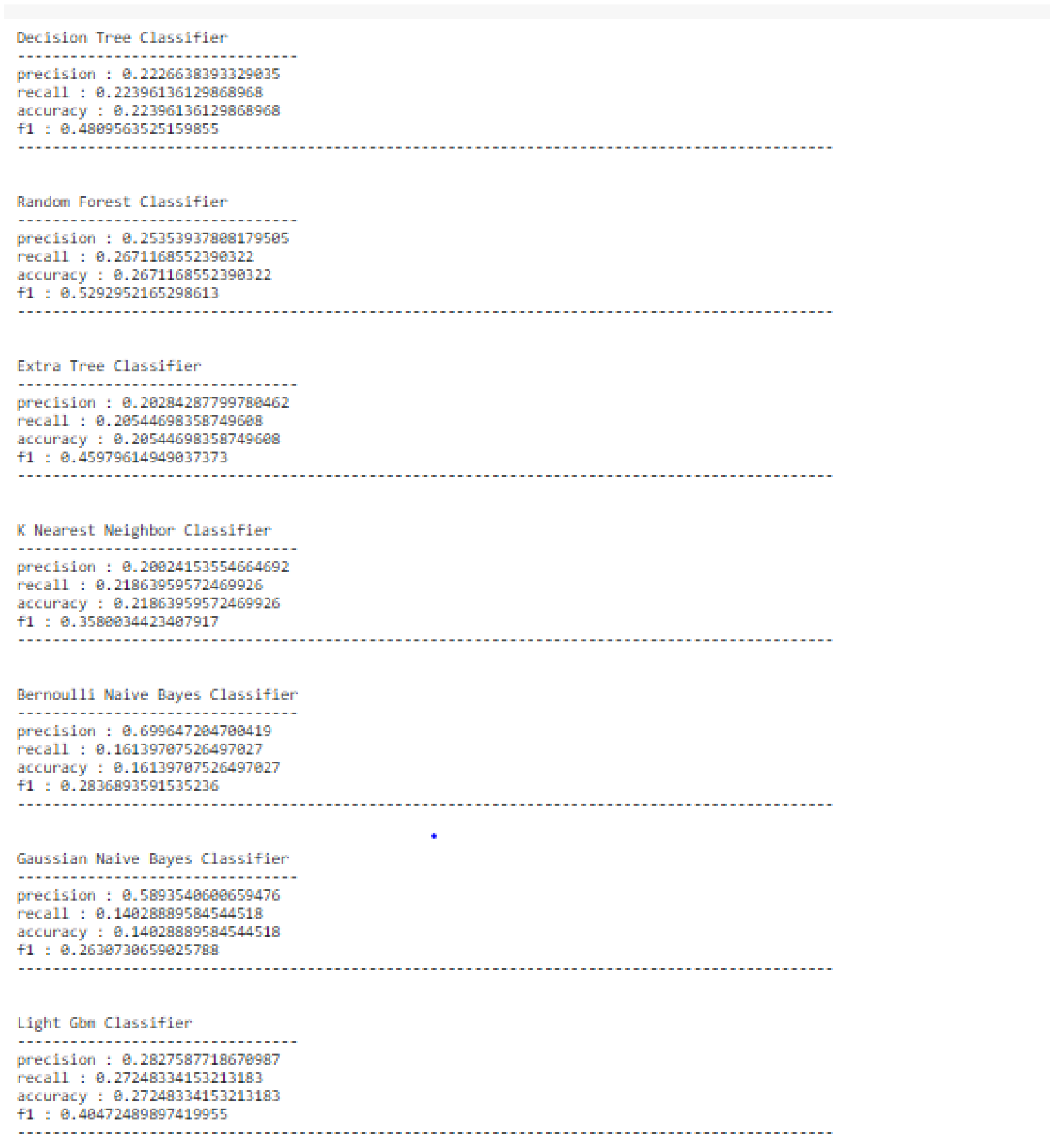
```
# Initialize a dictionary to store all the calculated scores for this algorithm
cur_results = {}

# Add in the Precision score
cur_results["precision"] = self.calcPrecisionScore(Y_pred, Y_test)

# Add in the Recall score
cur_results["recall"] = self.calcRecallScore(Y_pred, Y_test)

# Add in the Accuracy score
cur_results["accuracy"] = self.calcAccuracyScore(Y_pred, Y_test)

# Add in the F1 score
cur_results["f1"] = self.calcF1Score(Y_pred, Y_test)

# Add in the Confusion Matrix
cur_results["confusion_matrix"] = self.calcConfusionMatrix(Y_pred, Y_test)

# Store the scores of this algorithm into the class variable, for consolidation of all algorithms for a particular scenario
self.results[algo_name] = cur_results
```

Figure 39 - Code implementation for Calculating performance metrics.

## X. RESULTS

We ran our code implementation for three different scenarios for having a better prediction capability and thorough understanding of how different parameters affect the results of the machine learning algorithms. The seven machine learning algorithms used are namely - Decision Tree, Bernoulli and Gaussian Naive Bayes, Light GBM, Random Forest Classifiers, Linear SVM and Extra Tree Classifier.

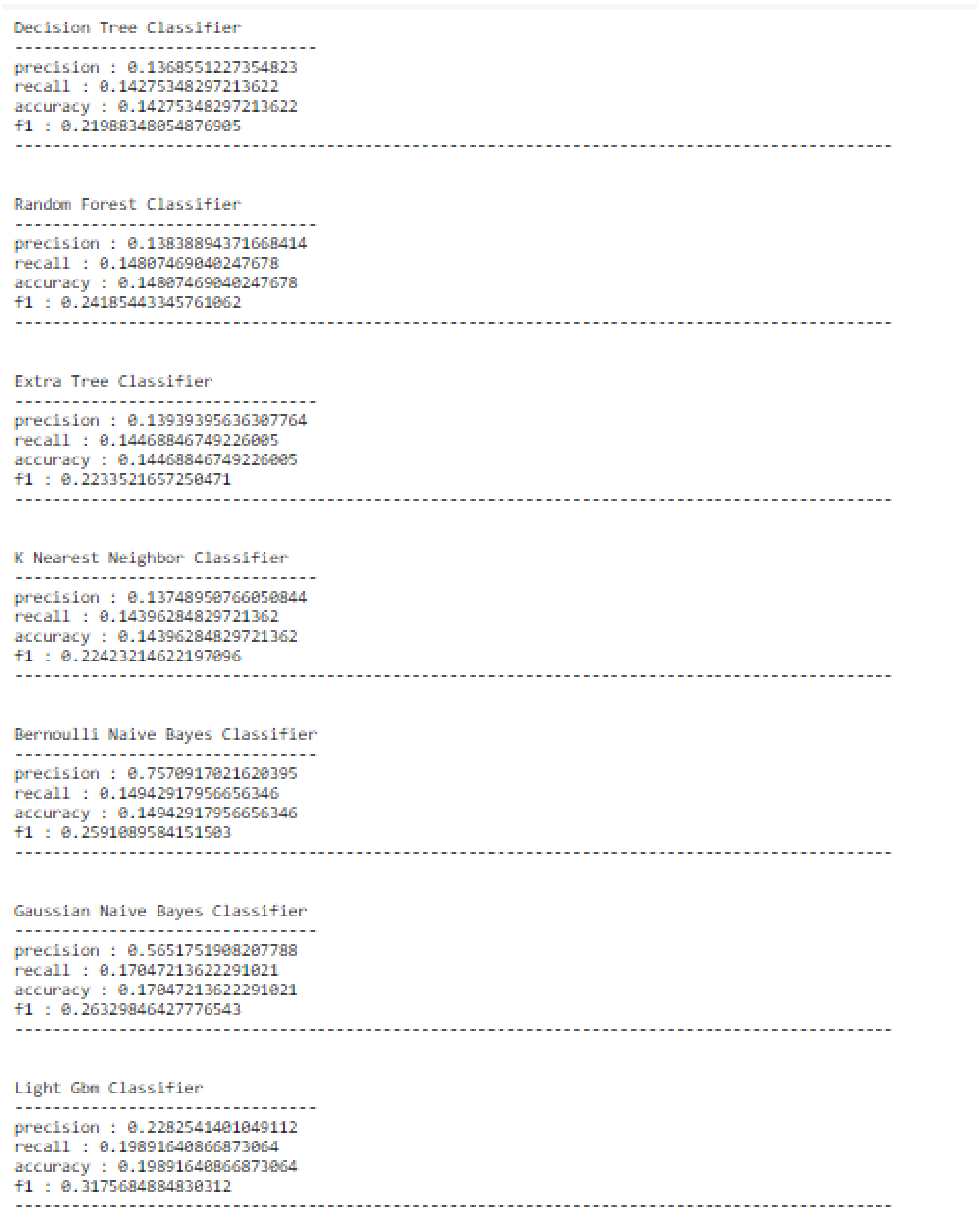
In the first section, we explored the possibility of being able to predict the "Offence Group" of the crime that might happen in the future based on parameters like `date`, `time`, `location`, etc.



```
Decision Tree Classifier
--------------------------------
precision : 0.2226638393329035
recall : 0.22396136129868968
accuracy : 0.22396136129868968
f1 : 0.4809563525159855
---------------------------------------------------------------------------------------------


Random Forest Classifier
--------------------------------
precision : 0.25353937808179505
recall : 0.2671168552390322
accuracy : 0.2671168552390322
f1 : 0.5292952165298613
---------------------------------------------------------------------------------------------


Extra Tree Classifier
--------------------------------
precision : 0.20284287799780462
recall : 0.20544698358749608
accuracy : 0.20544698358749608
f1 : 0.45979614949037373
---------------------------------------------------------------------------------------------


K Nearest Neighbor Classifier
--------------------------------
precision : 0.20024153554664692
recall : 0.21863959572469926
accuracy : 0.21863959572469926
f1 : 0.3580034423407917
---------------------------------------------------------------------------------------------


Bernoulli Naive Bayes Classifier
--------------------------------
precision : 0.699647204700419
recall : 0.16139707526497027
accuracy : 0.16139707526497027
f1 : 0.2836893591535236
---------------------------------------------------------------------------------------------


Gaussian Naive Bayes Classifier
--------------------------------
precision : 0.5893540600659476
recall : 0.14028889584544518
accuracy : 0.14028889584544518
f1 : 0.2630730659025788
---------------------------------------------------------------------------------------------


Light Gbm Classifier
--------------------------------
precision : 0.2827587718670987
recall : 0.27248334153213183
accuracy : 0.27248334153213183
f1 : 0.40472489897419955
---------------------------------------------------------------------------------------------
```

Figure 40 - Metric Scores for all models in Scenario : 1

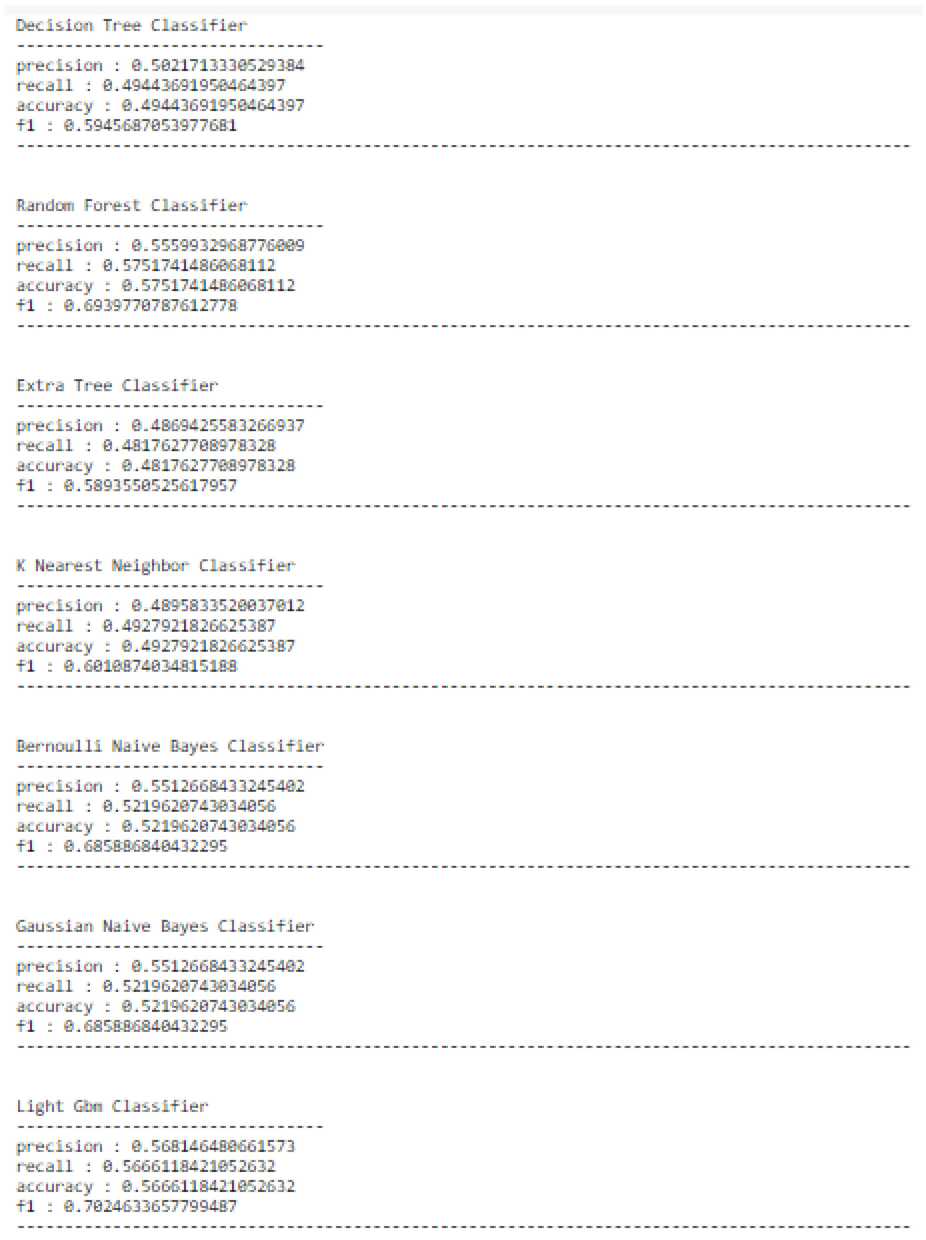
For second section results we have used the date and the offence type to predict where or in which district a particular type of crime can occur. This is done to check how a temporal location helps in preventing the crimes in future.



```
Decision Tree Classifier
--------------------------------
precision : 0.1368551227354823
recall : 0.14275348297213622
accuracy : 0.14275348297213622
f1 : 0.21988348054876905
--------------------------------------------------------------------------------------------


Random Forest Classifier
--------------------------------
precision : 0.13838894371668414
recall : 0.14807469040247678
accuracy : 0.14807469040247678
f1 : 0.24185443345761062
--------------------------------------------------------------------------------------------


Extra Tree Classifier
--------------------------------
precision : 0.13939395636307764
recall : 0.14468846749226005
accuracy : 0.14468846749226005
f1 : 0.2233521657250471
--------------------------------------------------------------------------------------------


K Nearest Neighbor Classifier
--------------------------------
precision : 0.13748950766050844
recall : 0.14396284829721362
accuracy : 0.14396284829721362
f1 : 0.22423214622197096
--------------------------------------------------------------------------------------------


Bernoulli Naive Bayes Classifier
--------------------------------
precision : 0.7570917021620395
recall : 0.14942917956656346
accuracy : 0.14942917956656346
f1 : 0.2591089584151503
--------------------------------------------------------------------------------------------


Gaussian Naive Bayes Classifier
--------------------------------
precision : 0.5651751908207788
recall : 0.17047213622291021
accuracy : 0.17047213622291021
f1 : 0.26329846427776543
--------------------------------------------------------------------------------------------


Light Gbm Classifier
--------------------------------
precision : 0.2282541401049112
recall : 0.19891640866873064
accuracy : 0.19891640866873064
f1 : 0.3175684884830312
--------------------------------------------------------------------------------------------
```

Figure 41 - Metric  Scores for all models in Scenario : 2

For the third section the results We used another type. This is based on the UCR parameter which stands for Uniform Crime Reporting. This section deals with the prediction of the seriousness of the crime that might occur at a particular time and location. The UCR_PART has categories like part 1-3, with one being the most serious types of crimes.



```
Decision Tree Classifier
--------------------------------
precision : 0.5021713330529384
recall : 0.49443691950464397
accuracy : 0.49443691950464397
f1 : 0.5945687053977681
--------------------------------------------------------------------------------------------

Random Forest Classifier
--------------------------------
precision : 0.5559932968776009
recall : 0.5751741486068112
accuracy : 0.5751741486068112
f1 : 0.6939770787612778
--------------------------------------------------------------------------------------------

Extra Tree Classifier
--------------------------------
precision : 0.4869425583266937
recall : 0.4817627708978328
accuracy : 0.4817627708978328
f1 : 0.5893550525617957
--------------------------------------------------------------------------------------------

K Nearest Neighbor Classifier
--------------------------------
precision : 0.4895833520037012
recall : 0.4927921826625387
accuracy : 0.4927921826625387
f1 : 0.6010874034815188
--------------------------------------------------------------------------------------------

Bernoulli Naive Bayes Classifier
--------------------------------
precision : 0.5512668433245402
recall : 0.5219620743034056
accuracy : 0.5219620743034056
f1 : 0.685886840432295
--------------------------------------------------------------------------------------------

Gaussian Naive Bayes Classifier
--------------------------------
precision : 0.5512668433245402
recall : 0.5219620743034056
accuracy : 0.5219620743034056
f1 : 0.685886840432295
--------------------------------------------------------------------------------------------

Light Gbm Classifier
--------------------------------
precision : 0.568146480661573
recall : 0.5666118421052632
accuracy : 0.5666118421052632
f1 : 0.7024633657799487
--------------------------------------------------------------------------------------------
```

Figure 42 - Metric  Scores for all models in Scenario : 3

We can see that in figure 39, figure 40 and figure 41, the Light Gbm model produced the better results in all the three cases by far than any of the other models. Also we can see that the UCR part of the dataset when trained alone gave the best results which can lead to the conclusion that none of the date or place can alone predict the rate of the crime on its own they need to have all the data for better prediction about the future crimes.

## XI. CONCLUSION

In this present work, we have analysed the crime stats for a large region like India on a broader scale, and have performed **exploratory data analysis** on this data of crimes committed through the years. While performing this, we have reached to a conclusion that more detailed data is required for us to begin reaching for solid conclusions in the field of crime prediction analysis. This led us in search for a much detailed dataset and we found the Boston Crime Dataset which had enough data to build a foundation to substantiate our cause in this field. We have done exactly the same in this paper, and proved the possibilities of crime predictions using ML algorithms.

We have in this paper taken up 7 Machine Learning Algorithms, and have applied them to a total of 3 scenarios to see how they perform in varying conditions. We have also concluded by the results that the LightGBM model has secured the best metric scores among the lot in these scenarios. Thus it can be concluded for now that this algorithm can be used, but the fight against crime does not stop at this level of accuracy.

Advances in Machine Learning did not stop when this level of accuracy was attained, it went on to nurture many other fields like Neural Networks, etc. There are many ways in which we can improve upon this foundation laid here, like by using Genetic Algorithms as a feature selector which models the algorithm to match closely to the real world conditions as it considers the natural evolution as a core component in its process. There are other portions of the ML pipeline which when tweaked would deliver better results.

There are also many other aspects of this project which can be worked upon in the future like concentrating on ways to improve the data collection strategies, to build a better more customized ML pipeline for this scenario, recommending better implementation strategies for this to work hand in hand with the police departments, better security of the whole system to prevent unauthorised access, recommending the governments for more detailed dataset formats which has more information, but does not compromise on the privacy of individuals, etc.

## XII. REFERENCES

1. S. Sathyadevan, M. S. Devan and S. S. Gangadharan, "Crime analysis and prediction using data mining," 2014 First International Conference on Networks & Soft Computing (ICNSC2014), 2014, pp. 406-412, doi: 10.1109/CNSC.2014.6906719.
2. Malathi, A. and Baboo, S.S., 2011. An enhanced algorithm to predict a future crime using data mining.
3. Wang, T., Rudin, C., Wagner, D. and Sevieri, R., 2013, June. Detecting patterns of crime with series finder. In Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence.
4. O. Llaha, "Crime Analysis and Prediction using Machine Learning," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020, pp. 496-501, doi: 10.23919/MIPRO48935.2020.9245120.
5. Hussain, K.Z., Durairaj, M. and Farzana, G.R.J., 2012, March. Criminal behavior analysis by using data mining techniques. In IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM-2012) (pp. 656-658). IEEE.
6. Keyvanpour, M.R., Javideh, M. and Ebrahimi, M.R., 2011. Detecting and investigating crime by means of data mining: a general crime matching framework. Procedia Computer Science, 3, pp.872-880.
7. Pratibha, A. Gahalot, Uprant, S. Dhiman and L. Chouhan, "Crime Prediction and Analysis," 2nd International Conference on Data, Engineering and Applications (IDEA), 2020, pp. 1-6, doi: 10.1109/IDEA49133.2020.9170731.
8. Yu, C.H., Ward, M.W., Morabito, M. and Ding, W., 2011, December. Crime forecasting using data mining techniques. In 2011 IEEE 11th international conference on data mining workshops (pp. 779-786). IEEE.
9. Thongsatapornwatana, U., 2016, January. A survey of data mining techniques for analyzing crime patterns. In 2016 Second Asian Conference on Defence Technology (ACDT)(pp. 123-128). IEEE.
10. Vadivel, A. and Shaila, S.G., 2016. Event pattern analysis and prediction at sentence level using Neuro-fuzzy model for crime event detection. Pattern Analysis and Applications, 19(3), pp.679-698.

11. Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F. and Pentland, A., 2014, November. Once upon a crime: towards crime prediction from demographics and mobile data. In Proceedings of the 16th international conference on multimodal interaction (pp. 427-434).
12. S. Kim, P. Joshi, P. S. Kalsi and P. Taheri, "Crime Analysis Through Machine Learning," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2018, pp. 415-420, doi: 10.1109/IEMCON.2018.8614828.
13. Grover, V., Adderley, R. and Bramer, M., 2006, December. Review of current crime prediction techniques. In International Conference on Innovative Techniques and Applications of Artificial Intelligence (pp. 233-237). Springer, London.
14. Marchant, R., Haan, S., Clancey, G. and Cripps, S., 2018. Applying machine learning to criminology: semi-parametric spatial-demographic Bayesian regression. Security Informatics, 7(1), pp.1-19.
15. McClendon, L. and Meghanathan, N., 2015. Using machine learning algorithms to analyze crime data. Machine Learning and Applications: An International Journal (MLAIJ), 2(1), pp.1-12.
16. Tayebi, M.A., Gla, U. and Brantingham, P.L., 2015, May. Learning where to inspect: Location learning for crime prediction. In 2015 IEEE International Conference on Intelligence and Security Informatics (ISI) (pp. 25-30). IEEE.
17. Sivaranjani, S., Sivakumari, S. and Aasha, M., 2016, October. Crime prediction and forecasting in Tamilnadu using clustering approaches. In 2016 International Conference on Emerging Technological Trends (ICETT) (pp. 1-6). IEEE.