CRIME DATA ANALYSIS USING MACHINE LEARNING

ABSTRACT

Recognizing the patterns of a criminal activity of a place is paramount in order to prevent it. Law enforcement agencies can work effectively and respond faster if they have better knowledge about crime patterns in different geological points of a city. The aim is to use machine learning with the help of python to classify a criminal incident by type ,depending on its occurrence at a given time and location. The survey is done using crime records of India.

INTRODUCTION

Criminal activities are present in every region of the world affecting quality of life and socio-economical development. As such, it is a major concern of many governments who are using different advanced technology to tackle such issues. Crime Analysis, a sub branch of criminology, studies the behavioral pattern of criminal activities and tries to identify the indicators of such events. Machine learning agents work with data and employ different techniques to find patterns in data making it very useful for predictive analysis. Law enforcement agencies use different patrolling strategies based on the information they get to keep an area secure. A machine learning agent can learn and analyze the pattern of occurrence of a crime based on the reports of previous criminal activities and can find hotspots based on time, type or any other factor. This technique is known as classification and it allows to predict nominal class labels. Classification has been used on many different domains such as financial market, business intelligence, healthcare, weather forecasting etc. In this research, a dataset from San-Francisco Open Data[8] is used which contains the reported criminal activities in the neighborhoods of the city San Francisco for a duration of 12 years. I used different classification techniques like Decision Tree, Naive Bayesian, Logistic Regression, k-Nearest Neighbor, Ensemble Methods to find hotspots of criminal activities based on the time of day. Results of different algorithms have been compared and most the effective approach has also been documented.

SYSTEM ANALYSIS

EXISTING SYSTEM

The dataset used for this is real and authentic. The dataset is acquired from UCI machine learning repository website. The title of the dataset is 'Crime and Communities'. It is prepared using real data from socio-economic data from 1990 US Census, law enforcement data from the 1990 US LEMAS survey and crime data from the 1995 FBI UCR. This dataset contains a total number of 147 attributes and 2216 instances.

PROPOSED SYSTEM

The purpose of this paper is to evaluate data mining methods and their performances that can be used for analyzing the collected data about the past crimes. I identified the most appropriate data mining methods to analyze the collected data from sources specialized in crime prevention by comparing them theoretically and practically. Three algorithms I used to analysis the crime data and I got good accuracy for those three algorithms.

LITERATURE SUREVY:

TITLE:

A KNN undersampling approach for data balancing. Journal of Intelligent Learning Systems and Applications

AUTHOR:

Beckmann, M., Ebecken, N. F., & de Lima, B. S. P. (2015)

CONTENT:

In supervised learning, the imbalanced number of instances among the classes in a dataset can make the algorithms to classify one instance from the minority class as one from the majority class. With the aim to solve this problem, the KNN algorithm provides a basis to other balancing methods. These balancing methods are revisited in this work, and a new and simple approach of KNN undersampling is proposed. The experiments demonstrated that the KNN undersampling method outperformed other sampling methods. The proposed method also outperformed the results of other studies, and indicates that the simplicity of KNN can be used as a base for efficient algorithms in machine learning and knowledge discovery.

TITLE:

Crime analysis and prediction using data mining

AUTHOR:

Sathyadevan, S., & Gangadharan, S. (2014, August)

CONTENT:

Crime analysis and prevention is a systematic approach for identifying and analyzing patterns and trends in crime. Our system can predict regions which have high probability for crime occurrence and can visualize crime prone areas. With the increasing advent of computerized systems, crime data analysts can help the Law enforcement officers to speed up the process of solving crimes. Using the concept of data mining we can extract previously unknown, useful information from an unstructured data. Here we have an approach between computer science and criminal justice to develop a data mining procedure that can help solve crimes faster. Instead of focusing on causes of crime occurrence like criminal background of offender, political enmity etc we are focusing mainly on crime factors of each day.

INPUT AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

SYSTEM SPECIFICATION:

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.

SOFTWARE REQUIREMENTS:

- **Operating system** : Windows 7 Ultimate.
- **Coding Language** : Python.
- **Front-End** : Python.
- **Designing** : Html,css,javascript.
- **Data Base** : MySQL.

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**

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.

MODULES

1. **Decision Tree Classifier:** Decision tree classification model forms a tree structure from dataset. Decision tree is built by dividing a dataset into smaller pieces. At each step in the algorithm, a decision tree node is splitted into two or more branches until it reaches leaf nodes. Leaf nodes indicates the class labels or result. At each step, decision tree chooses a feature that best splits the data with the help of two functions: Gini Impurity and Information Gain. Gini Impurity measures the probability of classifying a random sample incorrectly if the label is picked randomly according to the distribution in a branch.
2. **Gaussian Naive Bayes:** Gaussian Naive Bayes is a supervised classifier that uses naive assumption that there is no dependency between two features. This classifier is implemented by applying Bayesian Theorem.
3. **Logistic Regression:** Logistic regression uses linear boundaries to classify data into different categories. Logistic regression can work on both binary and multiclass problems. For multiclass dataset, one vs the rest scheme is used. In this method, logistic regression trains separate binary classifiers for each class. Meaning, each class is classified against all other classes, by assuming that all other classes is one category.
4. **K-Nearest Neighbor:** Nearest Neighbors method is used in both supervised and unsupervised learning. While testing with new data, KNN looks at k data points in training dataset which are closest to the test data point. k indicates the number of neighbors voting to classify a datapoint. The distance can be measured with various metrics. Euclidean distance is the most common choice.

5. **Ensemble Methods:** Ensemble learning is a method of combining multiple learning algorithm together to achieve better performance over a single algorithm. Ensemble methods can be divided into two categories: averaging methods and boosting methods.
   In this paper, two ensemble methods are used: Random Forest, which follows the principle of averaging method and Adaboost which is a boosting model.
   a) **Random Forest:** In this ensemble model several decision trees are built using samples drawn with replacement from the training set. The splitting of each node of a tree is not based on the best 19 split of all features, rather the best split among a random set of features.
   b) **Adaboost:** Adaboost or Adaptive Boosting is a boosting algorithm. Adaboost combines several weak learners to produce a stronger model. The final output is obtained from the weighted sum of the weak models. As it is a sequential process, in each step a weak learner is changed in favor of misclassified data points in previous classifiers.

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.

Software environment

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.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!"); However in Python version 2.4.3, this produces the following result −

Hello, Python!

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable

$./test.py
```

This produces the following result −

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambdayield

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
```

```
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
    'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python

# First comment
```

print "Hello, Python!" # second comment

This produces the following result −

Hello, Python!

You can type a comment on the same line after a statement or expression −

name = "Madisetti" # This is again comment

You can comment multiple lines as follows −

This is a comment.

This is a comment, too.

This is a comment, too.

I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline

comment.

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5];

list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5);

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara

dict['Age']:  7
```

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python


dict = {['Name']: 'Zara', 'Age': 7}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python

tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python
```

```
tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

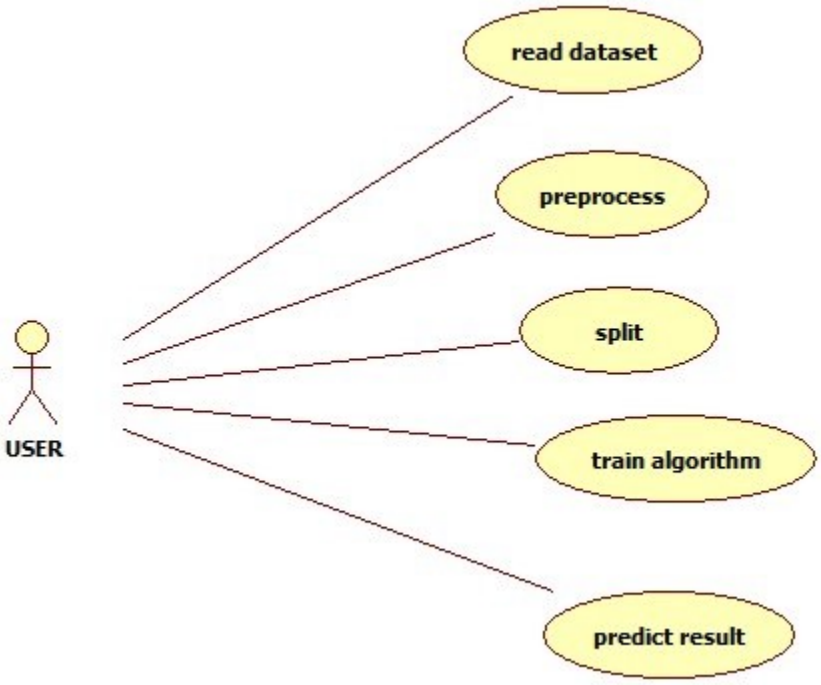
```
('physics', 'chemistry', 1997, 2000)

After deleting tup :

Traceback (most recent call last):

  File "test.py", line 9, in <module>

    print tup;

NameError: name 'tup' is not defined
```

UMLDIAGRAMS

USE CASE DIAGRAM:

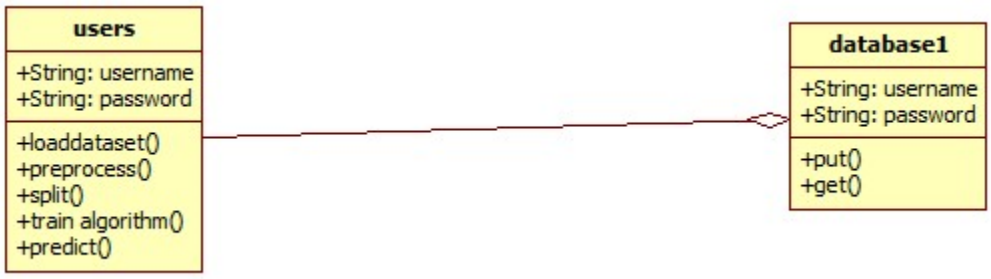
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in

the system can be depicted.



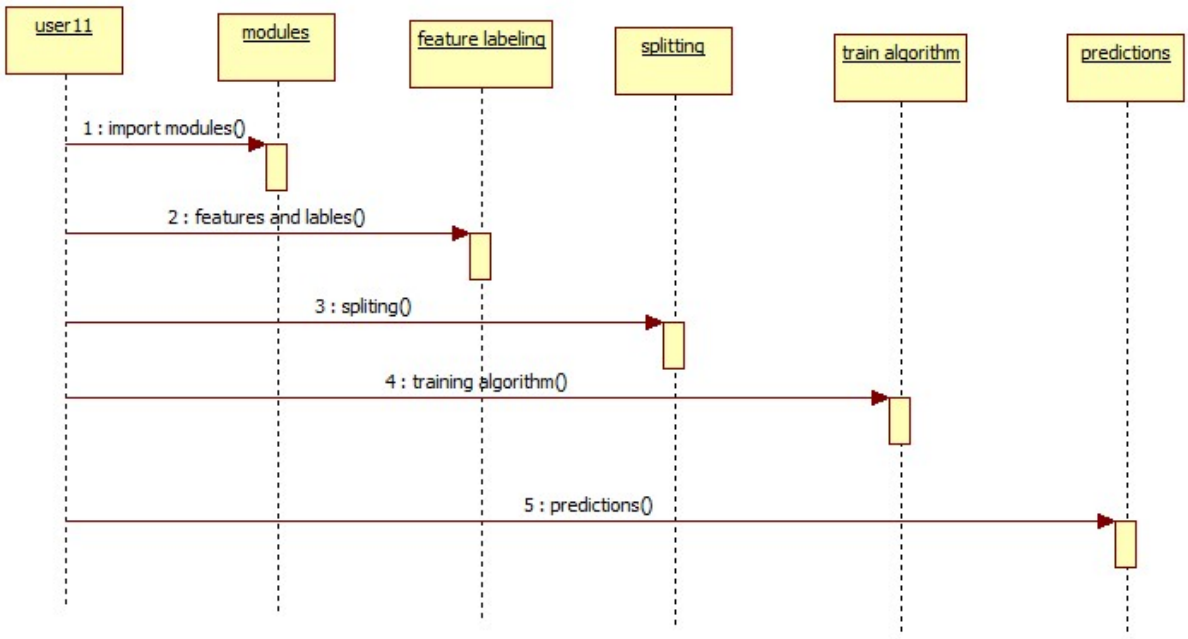
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



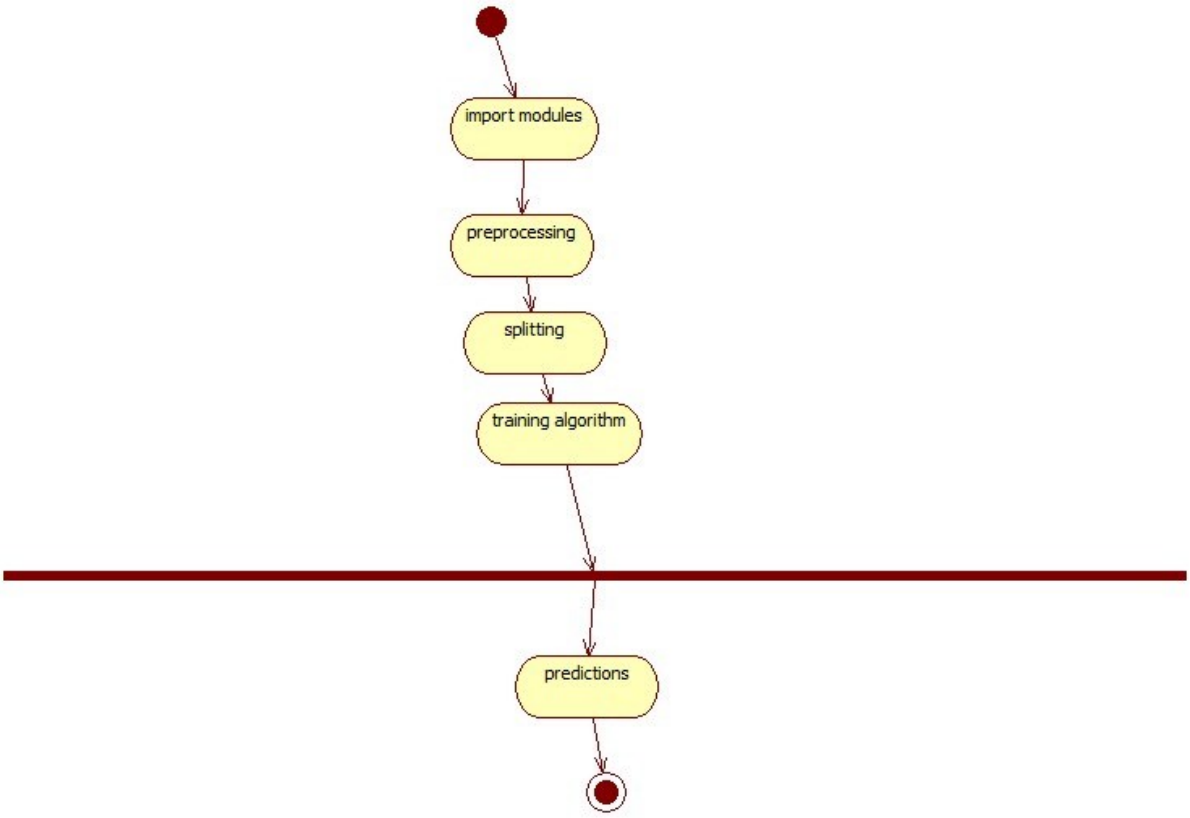
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



ACTIVITY DIAGRAM:

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


import modules
preprocessing
splitting
training algorithm
predictions

IMPLEMENTATION:

```
#-----------------------------------------------#

#1) IMPORT LIBRARIES

#Computation and Structuring:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
#Modeling:
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
#Testing:
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,precision_score,recall_score
```

```
#-----------------------------------------------#

#2) DATA IMPORT AND PRE-PROCESSING

#import full data set

df = pd.read_csv('MCI_2014_to_2018.csv',sep=',')

#list of relevant columns for model

col_list = ['occurrenceyear',
'occurrencemonth','occurrenceday','occurrencedayofyear','occurrencedayofweek','occurrencehour','MCI', 'Division', 'Hood_ID','premisetype']

#dataframe created from list of relevant columns

df2 = df[col_list]

df2 = df2[df2['occurrenceyear'] > 2013] #drop "stale" crimes, where occurence is before 2014. Since data set is filtered based on reported date, we're ignoring these old crimes.

#Factorize dependent variable column:

crime_var = pd.factorize(df2['MCI']) #codes the list of crimes to a int64 variable
```

```
df2['MCI'] = crime_var[0]

definition_list_MCI = crime_var[1] #create an index reference so we know which crimes are coded to which factors


#factorize independent variables:


#factorize premisetype:


premise_var = pd.factorize(df2['premisetype'])

df2['premisetype'] = premise_var[0]

definition_list_premise = premise_var[1]


#factorize occurenceyear:


year_var = pd.factorize(df2['occurrenceyear'])

df2['occurrenceyear'] = year_var[0]

definition_list_year = year_var[1]


#factorize occurencemonth:

month_var = pd.factorize(df2['occurrencemonth'])

df2['occurrencemonth'] = month_var[0]
```

```
definition_list_month = month_var[1]


#factorize occurenceday:

day_var = pd.factorize(df2['occurrenceday'])

df2['occurenceday'] = day_var[0]

definition_list_day = day_var[1]

#factorize occurencedayofweek:

dayweek_var = pd.factorize(df2['occurrencedayofweek'])

df2['occurrencedayofweek'] = dayweek_var[0]

definition_list_day = dayweek_var[1]


#factorize division:

division_var = pd.factorize(df2['Division'])

df2['Division'] = division_var[0]

definition_list_division = division_var[1]

#factorize HOOD_ID:

hood_var = pd.factorize(df2['Hood_ID'])

df2['Hood_ID'] = hood_var[0]

definition_list_hood = hood_var[1]


#factorize occurencehour:
```

```
hour_var = pd.factorize(df2['occurrencehour'])

df2['occurrencehour'] = hour_var[0]

definition_list_hour = hour_var[1]

#factorize occurencedayofyear:

dayyear_var = pd.factorize(df2['occurrencedayofyear'])

df2['occurrencedayofyear'] = dayyear_var[0]

definition_list_dayyear = dayyear_var[1]



#set X and Y:

X = df2.drop(['MCI'],axis=1).values #sets x and converts to an array

#print(X.head())

y = df2['MCI'].values #sets y and converts to an array

#split the data into train and test sets for numeric encoded dataset:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 21)

#need to OneHotEncode all the X variables for input into the classification model:

binary_encoder = OneHotEncoder(sparse=False)

encoded_X = binary_encoder.fit_transform(X)

X_train_OH, X_test_OH, y_train_OH, y_test_OH = train_test_split(encoded_X, y, test_size =
0.25, random_state = 21)

#-----------------------------------------------#

#3) MODELING AND TESTING:
```

```
#Numeric Encoded Model w/ SKLEARN:

classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 42)

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test) # Predicting the Test set results

print(accuracy_score(y_test, y_pred)) #accuracy at 0.63

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test,y_pred, target_names=definition_list_MCI))

#theft over is pulling down results. Pretty good on Assault (largest sample size) and break and enter

#One Hot Encoded Model w/ SKLEARN:

classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 42)

classifier.fit(X_train_OH, y_train_OH)

y_pred_OH = classifier.predict(X_test_OH) # Predicting the Test set results

print(accuracy_score(y_test_OH, y_pred_OH)) #modest improvement to 0.648

print(confusion_matrix(y_test_OH, y_pred_OH))

print(classification_report(y_test_OH,y_pred_OH, target_names=definition_list_MCI)) #modest improvement

#Balanced Class Weight doesn't make a big difference for results:

classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 42, class_weight='balanced')

classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)

print('RandomForestClassifier Algorithm Accuracy')

rf_ac=accuracy_score(y_test, y_pred) #accuracy at 0.63

print(rf_ac)

rf_precision = precision_score(y_test, y_pred,average='macro') * 100

print(rf_precision)

rf_recall = recall_score(y_test, y_pred,average='macro') * 100

print(rf_recall)

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test,y_pred, target_names=definition_list_MCI))

#-------------------------------------------------#

seed = 7

cart = DecisionTreeClassifier()

num_trees = 100

model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)

model.fit(X_train_OH, y_train_OH)

y_pred_OH = model.predict(X_test_OH) # Predicting the Test set results

print('BaggingClassifier Algorithm Accuracy')

bc_ac=accuracy_score(y_test_OH, y_pred_OH) #modest improvement to 0.648

bc_ac=accuracy_score(y_test_OH, y_pred_OH) #modest improvement to 0.648

print(bc_ac)
```

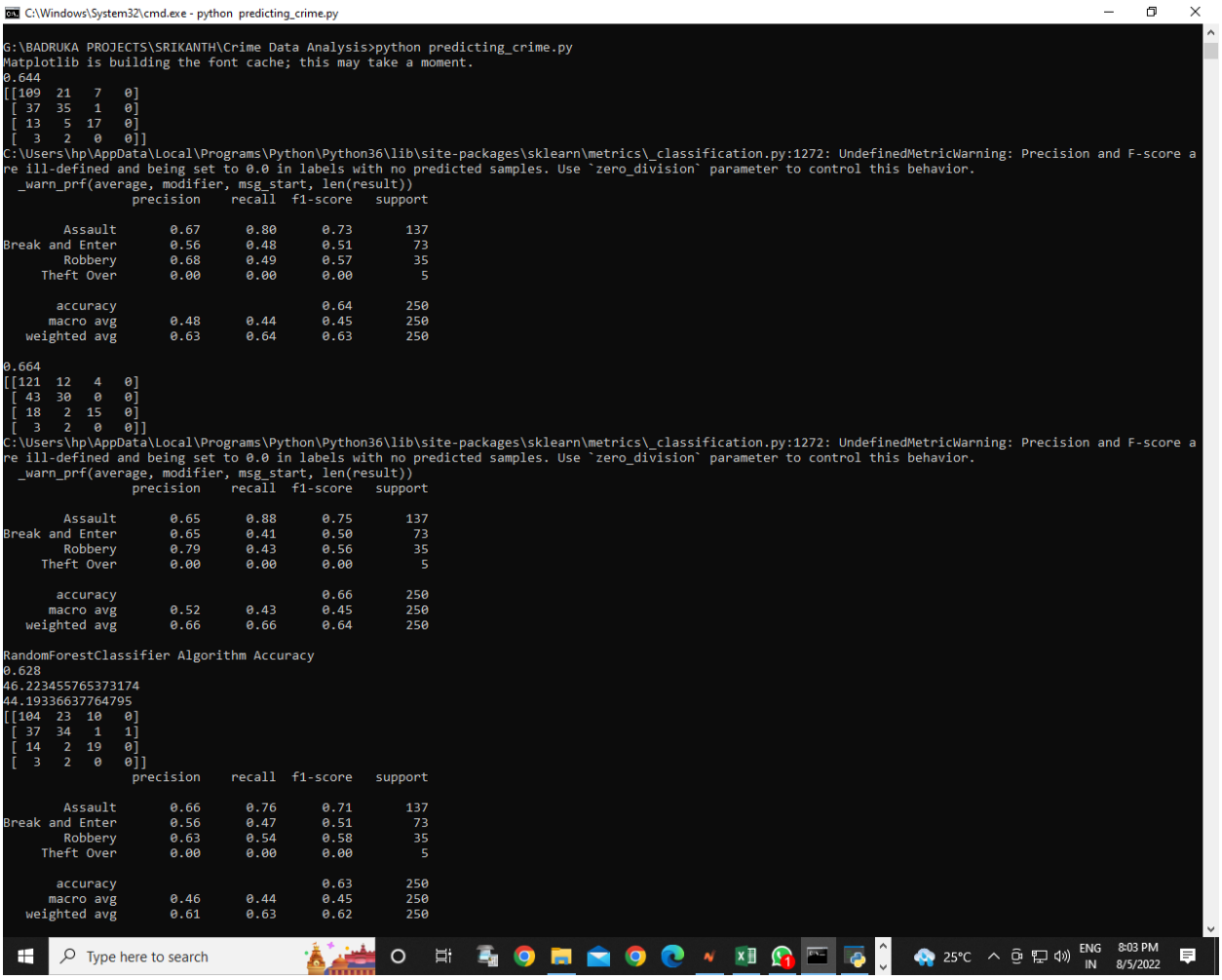
```
bc_precision = precision_score(y_test_OH, y_pred_OH,average='macro') * 100
print(bc_precision)
bc_recall = recall_score(y_test_OH, y_pred_OH,average='macro') * 100
print(bc_recall)
print(confusion_matrix(y_test_OH, y_pred_OH))
print(classification_report(y_test_OH,y_pred_OH, target_names=definition_list_MCI))
#gradientboost performs poorly relative to randomforest


grad_class = GradientBoostingClassifier(learning_rate=0.1,n_estimators = 10, random_state = 42)
grad_class.fit(X_train_OH, y_train_OH)
y_pred_OH = grad_class.predict(X_test_OH) # Predicting the Test set results
print('GradientBoostingClassifier Algorithm Accuracy')
gbc_ac=accuracy_score(y_test_OH, y_pred_OH) #modest improvement to 0.648
print(gbc_ac)
gbc_precision = precision_score(y_test_OH, y_pred_OH,average='macro') * 100
print(gbc_precision)
gbc_recall = recall_score(y_test_OH, y_pred_OH,average='macro') * 100
print(gbc_recall)
```

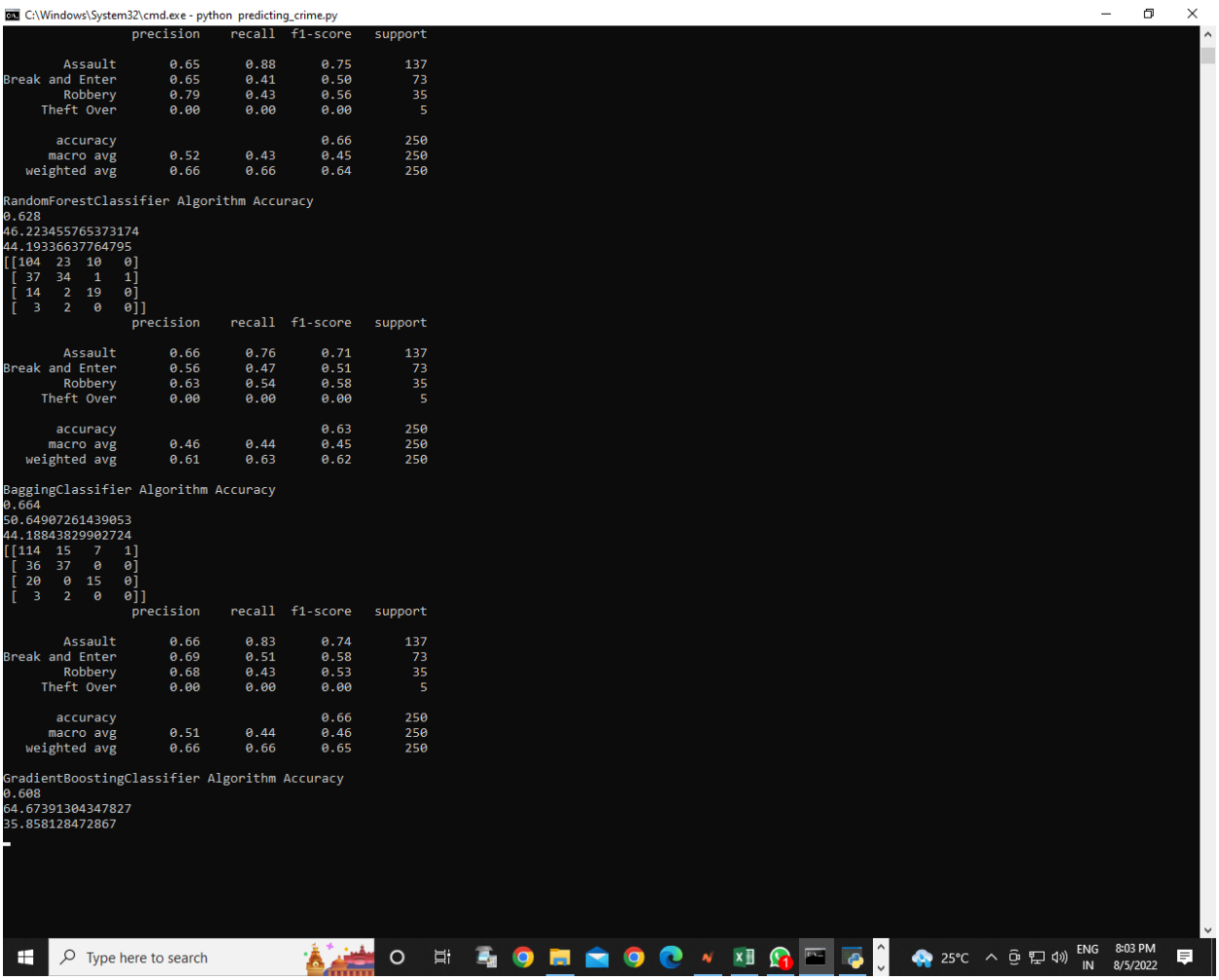
```
height = [rf_ac,bc_ac,gbc_ac]

bars = ('RFC', 'BC','GBC')

y_pos = np.arange(len(bars))

plt.bar(y_pos, height)

plt.xticks(y_pos, bars)

plt.title("Accuracy comparison")

plt.show()


height1 = [rf_precision,bc_precision,gbc_precision]

bars1 = ('RFC', 'BC','GBC')

y_pos = np.arange(len(bars))

plt.bar(y_pos, height1)

plt.xticks(y_pos, bars1)

plt.title("Precision comparison")

plt.show()


height2 = [rf_recall,bc_recall,gbc_recall]

bars2 = ('RFC', 'BC','GBC')

y_pos = np.arange(len(bars))

plt.bar(y_pos, height2)

plt.xticks(y_pos, bars2)
```

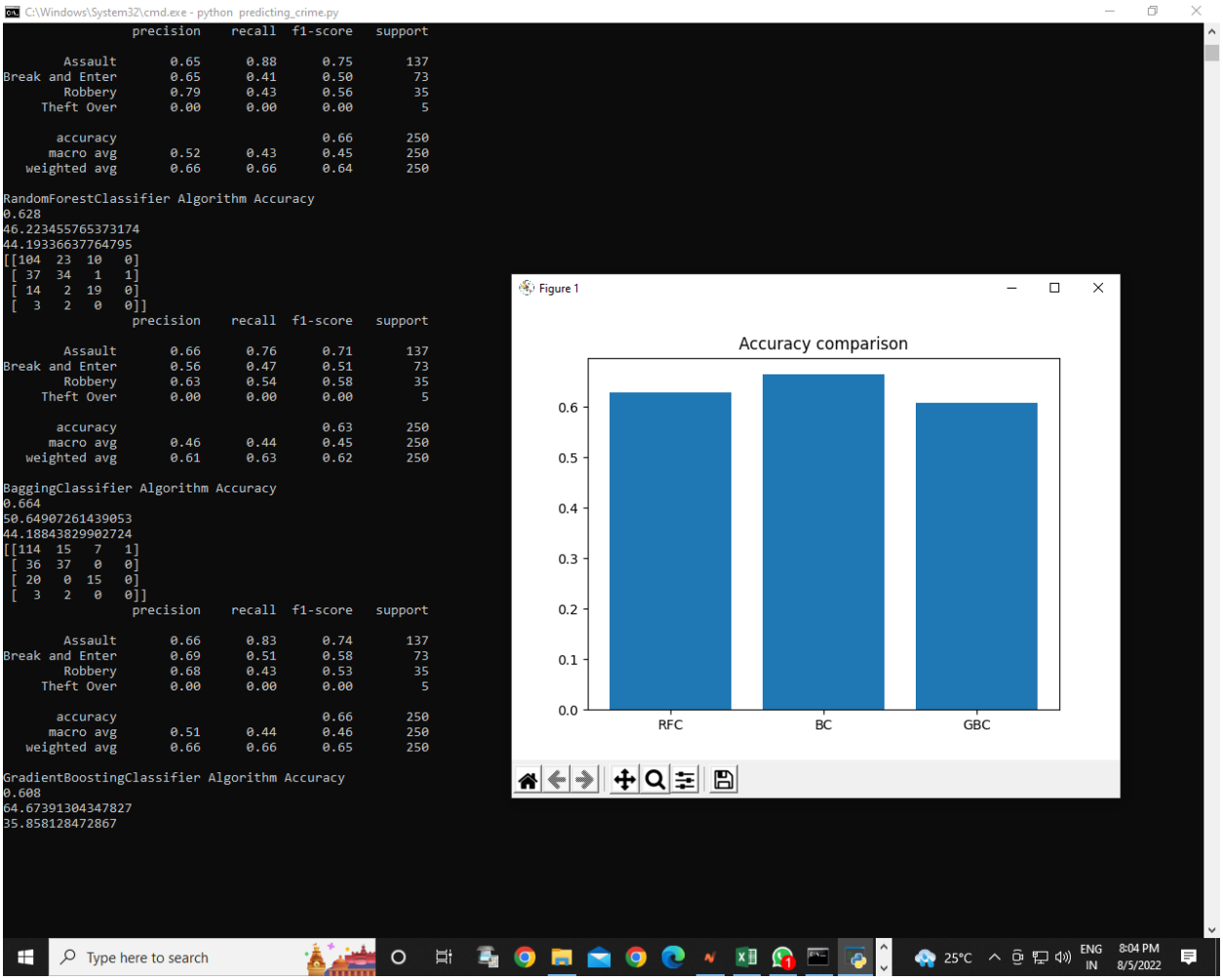
plt.title("Recall comparison")

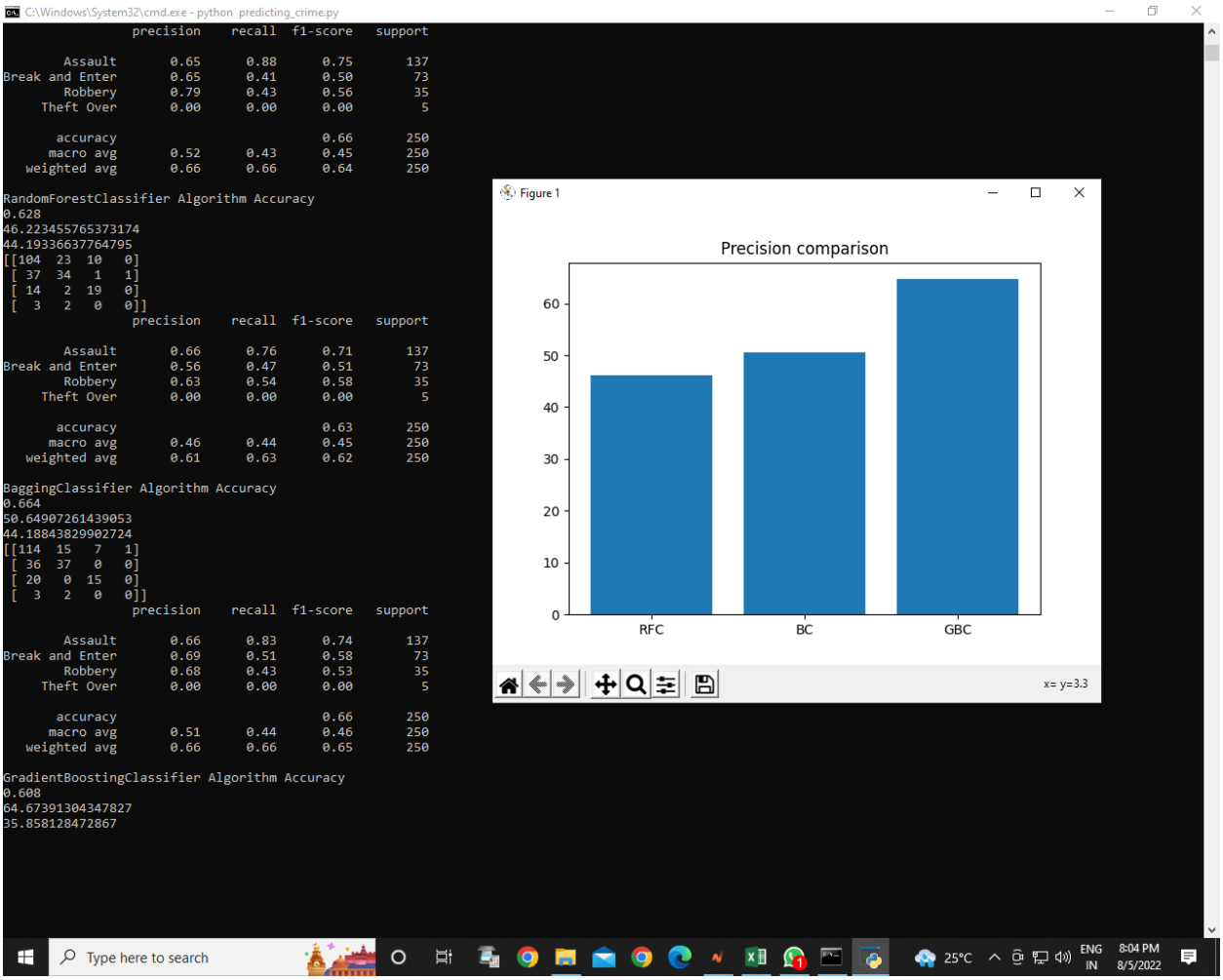
plt.show()

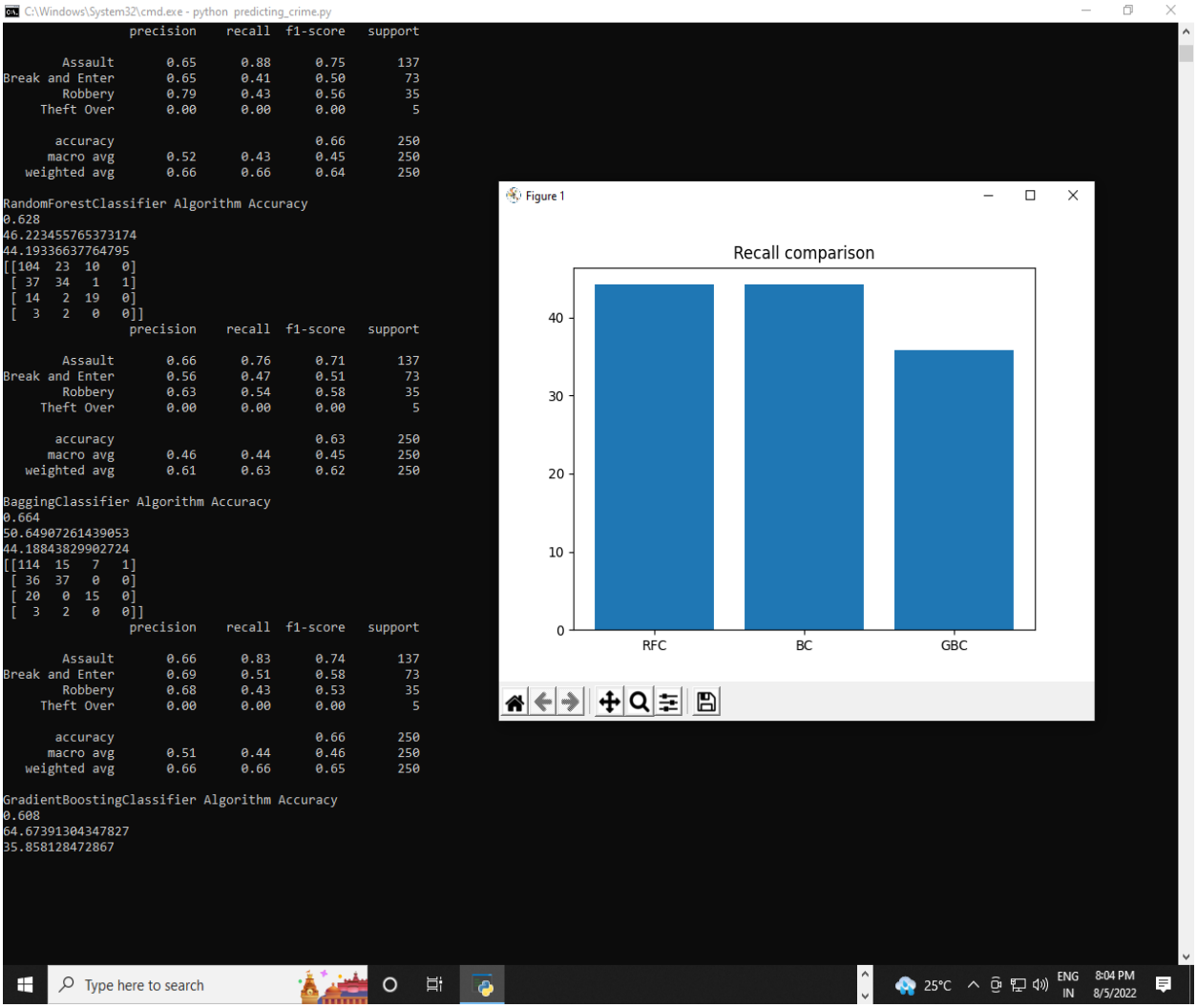
OUTPUT RESULTS

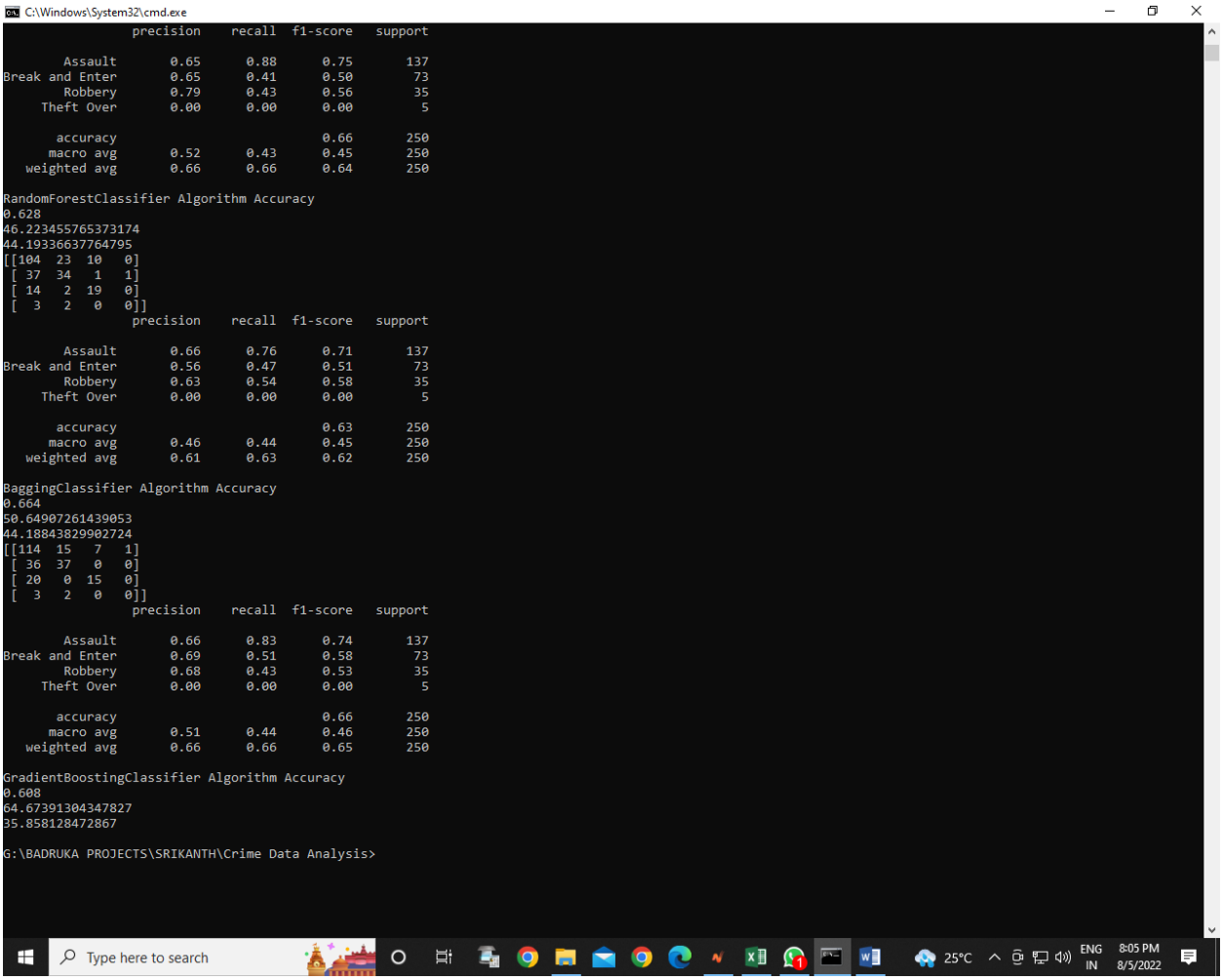



C:\Windows\System32\cmd.exe - python predicting_crime.py
precision recall f1-score support
Assault 0.65 0.88 0.75 137
Break and Enter 0.65 0.41 0.50 73
Robbery 0.79 0.43 0.56 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.52 0.43 0.45 250
weighted avg 0.66 0.66 0.64 250
RandomForestClassifier Algorithm Accuracy
0.628
46.223455765373174
44.19336637764795
[[104 23 10 0]
[37 34 1 1]
[14 2 19 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.76 0.71 137
Break and Enter 0.56 0.47 0.51 73
Robbery 0.63 0.54 0.58 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.63 250
macro avg 0.46 0.44 0.45 250
weighted avg 0.61 0.63 0.62 250
BaggingClassifier Algorithm Accuracy
0.664
50.64907261439053
44.18843829902724
[[114 15 7 1]
[36 37 0 0]
[20 0 15 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.83 0.74 137
Break and Enter 0.69 0.51 0.58 73
Robbery 0.68 0.43 0.53 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.51 0.44 0.46 250
weighted avg 0.66 0.66 0.65 250
GradientBoostingClassifier Algorithm Accuracy
0.608
64.67391304347827
35.858128472867
Type here to search
25°C
ENG
IN
8:03 PM
8/5/2022


C:\Windows\System32\cmd.exe - python predicting_crime.py
precision recall f1-score support
Assault 0.65 0.88 0.75 137
Break and Enter 0.65 0.41 0.50 73
Robbery 0.79 0.43 0.56 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.52 0.43 0.45 250
weighted avg 0.66 0.66 0.64 250
RandomForestClassifier Algorithm Accuracy
0.628
46.223455765373174
44.19336637764795
[[104 23 10 0]
[37 34 1 1]
[14 2 19 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.76 0.71 137
Break and Enter 0.56 0.47 0.51 73
Robbery 0.63 0.54 0.58 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.63 250
macro avg 0.46 0.44 0.45 250
weighted avg 0.61 0.63 0.62 250
BaggingClassifier Algorithm Accuracy
0.664
50.64907261439053
44.18843829902724
[[114 15 7 1]
[36 37 0 0]
[20 0 15 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.83 0.74 137
Break and Enter 0.69 0.51 0.58 73
Robbery 0.68 0.43 0.53 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.51 0.44 0.46 250
weighted avg 0.66 0.66 0.65 250
GradientBoostingClassifier Algorithm Accuracy
0.608
64.67391304347827
35.858128472867
Figure 1
Accuracy comparison
0.0
0.1
0.2
0.3
0.4
0.5
0.6
RFC
BC
GBC
Type here to search
25°C
ENG
IN
8:04 PM
8/5/2022


C:\Windows\System32\cmd.exe - python predicting_crime.py
precision recall f1-score support
Assault 0.65 0.88 0.75 137
Break and Enter 0.65 0.41 0.50 73
Robbery 0.79 0.43 0.56 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.52 0.43 0.45 250
weighted avg 0.66 0.66 0.64 250
RandomForestClassifier Algorithm Accuracy
0.628
46.223455765373174
44.19336637764795
[[104 23 10 0]
[37 34 1 1]
[14 2 19 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.76 0.71 137
Break and Enter 0.56 0.47 0.51 73
Robbery 0.63 0.54 0.58 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.63 250
macro avg 0.46 0.44 0.45 250
weighted avg 0.61 0.63 0.62 250
BaggingClassifier Algorithm Accuracy
0.664
50.64907261439053
44.18843829902724
[[114 15 7 1]
[36 37 0 0]
[20 0 15 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.83 0.74 137
Break and Enter 0.69 0.51 0.58 73
Robbery 0.68 0.43 0.53 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.51 0.44 0.46 250
weighted avg 0.66 0.66 0.65 250
GradientBoostingClassifier Algorithm Accuracy
0.608
64.67391304347827
35.858128472867
Figure 1
Precision comparison
60
50
40
30
20
10
0
RFC
BC
GBC
x= y=3.3
Type here to search
25°C
ENG
IN
8:04 PM
8/5/2022


C:\Windows\System32\cmd.exe - python predicting_crime.py
precision recall f1-score support
Assault 0.65 0.88 0.75 137
Break and Enter 0.65 0.41 0.50 73
Robbery 0.79 0.43 0.56 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.52 0.43 0.45 250
weighted avg 0.66 0.66 0.64 250
RandomForestClassifier Algorithm Accuracy
0.628
46.223455765373174
44.19336637764795
[[104 23 10 0]
[37 34 1 1]
[14 2 19 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.76 0.71 137
Break and Enter 0.56 0.47 0.51 73
Robbery 0.63 0.54 0.58 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.63 250
macro avg 0.46 0.44 0.45 250
weighted avg 0.61 0.63 0.62 250
BaggingClassifier Algorithm Accuracy
0.664
50.64907261439053
44.18843829902724
[[114 15 7 1]
[36 37 0 0]
[20 0 15 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.83 0.74 137
Break and Enter 0.69 0.51 0.58 73
Robbery 0.68 0.43 0.53 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.51 0.44 0.46 250
weighted avg 0.66 0.66 0.65 250
GradientBoostingClassifier Algorithm Accuracy
0.608
64.67391304347827
35.858128472867
Figure 1
Recall comparison
40
30
20
10
0
RFC
BC
GBC
Type here to search
25°C
ENG
IN
8:04 PM
8/5/2022


C:\Windows\System32\cmd.exe
precision recall f1-score support
Assault 0.65 0.88 0.75 137
Break and Enter 0.65 0.41 0.50 73
Robbery 0.79 0.43 0.56 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.52 0.43 0.45 250
weighted avg 0.66 0.66 0.64 250
RandomForestClassifier Algorithm Accuracy
0.628
46.223455765373174
44.19336637764795
[[104 23 10 0]
[37 34 1 1]
[14 2 19 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.76 0.71 137
Break and Enter 0.56 0.47 0.51 73
Robbery 0.63 0.54 0.58 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.63 250
macro avg 0.46 0.44 0.45 250
weighted avg 0.61 0.63 0.62 250
BaggingClassifier Algorithm Accuracy
0.664
50.64907261439053
44.18843829902724
[[114 15 7 1]
[36 37 0 0]
[20 0 15 0]
[3 2 0 0]]
precision recall f1-score support
Assault 0.66 0.83 0.74 137
Break and Enter 0.69 0.51 0.58 73
Robbery 0.68 0.43 0.53 35
Theft Over 0.00 0.00 0.00 5
accuracy 0.66 250
macro avg 0.51 0.44 0.46 250
weighted avg 0.66 0.66 0.65 250
GradientBoostingClassifier Algorithm Accuracy
0.608
64.67391304347827
35.858128472867
G:\BADRUKA PROJECTS\SRIKANTH\Crime Data Analysis>
Type here to search
25°C
ENG
IN
8:05 PM
8/5/2022

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.

CONCLUSION

Throughout the research it has been evident that basic details of a criminal activities in an area contains indicators that can be used by machine learning agents to classify a criminal activity given a location and date. Even though the learning agent suffers from imbalanced categories of the dataset, it was able to overcome the difficulty by oversampling and under sampling the dataset. Through the experiments, it can be seen the imbalanced dataset was benefitted by using ENN under sampling. Using the under sampled data, Adaboost decision tree successfully classified criminal activities based on the time and location. With a accuracy of 81.93%, it was able to outperform other machine learning algorithms. Imbalanced classes are one of the main hurdles to achieve a better result. Though the machine learning agent was able to predictive model out of simply crime data, a demographic dataset would probably help to further improve the result and solidify it.

BIBLIOGRAPHY

1. Beckmann, M., Ebecken, N. F., & de Lima, B. S. P. (2015). A KNN undersampling approach for data balancing. Journal of Intelligent Learning Systems and Applications, 7(4), 104.

2. Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F., & 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). ACM.

3. Braithwaite J. Crime, Shame and Reintegration. Ambridge: Cambridge University Press, 1989.

4. Sathyadevan, S., & Gangadharan, S. (2014, August). Crime analysis and prediction using data mining. In Networks & Soft Computing (ICNSC), 2014 First International Conference on (pp. 406-412). IEEE.

5. Nath, S. V. (2006, December). Crime pattern detection using data mining. In Web intelligence and intelligent agent technology workshops, 2006. wi-iat 2006 workshops. 2006 ieee/wic/acm international conference on (pp. 41-44). IEEE.

6. Zhao, X., & Tang, J. (2017, November). Exploring Transfer Learning for Crime Prediction. In Data Mining Workshops (ICDMW), 2017 IEEE International Conference on (pp. 1158-1159). IEEE.

7. Al Boni, M., & Gerber, M. S. (2016, December). AreaSpecific Crime Prediction Models. In Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on (pp. 671-676). IEEE.

8. Tayebi, M. A., Gla, U., & Brantingham, P. L. (2015, May). Learning where to inspect: location learning for crime prediction. In Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on (pp. 25-30). IEEE.