Crime Prediction Application

by

Sohan Biswas

(Registration Number: 18370049)

Project report Submitted in fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF ENGINEERING AND TECHNOLOGY

PONDICHERRY UNIVERSITY PUDUCHERRY-605014 MAY 2018

BONAFIDE CERTIFICATE

This is to certify that this project work entitled **"Crime Prediction"** is the bonafide record of work done by **Mr. Sohan Biswas (Reg. No.: 18370049)** at Pondicherry University in the fulfillment for the degree of Master of Science in Computer Science, Department of Computer Science, School of Engineering and Technology, Pondicherry University.

This work has not been submitted elsewhere for the award of any other degree to the best our knowledge.

INTERNAL GUIDE

Dr. V. Uma

Assistant Professor

Department of Computer Science Engineering and Technology
Pondicherry University.
Pondicherry-605014.

HEAD OF THE DEPARTMENT

Dr. T. Chithralekha

Professor and Head

Department of Computer Science School of School of Engineering and Technology
Pondicherry University.
Pondicherry-605014.

Submitted for the Viva-Voce Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I have immense pleasure in expressing our sincerest and deepest sense of gratitude towards our guide Dr. R. Sunitha and Dr. V. Uma for the assistance, valuable guidance and co-operation in carrying out this project successfully. We have developed this project with the help of Faculty members of our institute and we are extremely grateful to all of them. We also take this opportunity to thank Head of the Department Dr. T. Chithralekha, for providing the required facilities in completing this project. We are greatly thankful to our parents, friends and faculty members for their motivation, guidance and help whenever needed.

Abstract

To be better prepared to respond to criminal activity, it is important to understand patterns in crime. In our project, we analyze crime data from the city of San Francisco, scraped from publicly available website of datasf.org>opendata. It is publicly available dataset from the police department.

At the outset, the task is to predict which category of crime is most likely to occur given a time and place in San Francisco.
The use of AI and machine learning to detect crime via sound or cameras currently exists, is proven to work, and expected to continue to expand.

The use of AI/ML in predicting crimes or an individual's likelihood for committing a crime has promise but is still more of an unknown. The biggest challenge will probably be "proving" to politicians that it works. When a system is designed to stop something from happening, it is difficult to prove the negative. Companies that are directly involved in providing governments with AI tools to monitor areas or predict crime will likely benefit from a positive feedback loop. Improvements in crime prevention technology will likely spur increased total spending on this technology.

We also attempt to make our classification task more meaningful by merging multiple classes into larger classes. Finally, we report and reflect on our results with different classifiers, and dwell on avenues for future work.

TABLE OF CONTENTS

Chapter-1
Introduction

Many important questions in public safety and protection relate to crime, and a better understanding of crime is beneficial in multiple ways: it can lead to targeted and sensitive practices by law enforcement authorities to mitigate crime, and more concerted efforts by citizens and authorities to create healthy neighborhood environments.

With the advent of the Big Data era and the availability of fast, efficient algorithms for data analysis, understanding patterns in crime from data is an active and growing field of research.

The inputs to our algorithms are time (hour, day, month, and year), place (latitude and longitude), and categories of crime:

The output is the class of crime that is likely to have occurred. We try out multiple classification algorithms, such as Numeric Encoded Model, One Hot Encoded Model w/ SKLEARN, Balanced Class Weight and gradient boost to perform poorly relative to random forest classifier.
We also perform multiple classification tasks – we first try to predict which of the classes of crimes are likely to have occurred, and later try to differentiate between violent and non-violent crimes.

1.1 Rationale

With the rapid urbanization and development of big cities and towns, the graph of crimes is also on the increase. This phenomenal rise in offences and crime in cities is a matter of great concern and alarm to all of us.

There are robberies, murders, rapes and what not. The frequent and repeated thefts, burglaries, robberies, murders, killings, rapes, shoplifting, pick pocketing, drug- abuse, illegal trafficking,

smuggling, theft of vehicles etc., have made the common citizens to have sleepless nights and restless days.

They feel very insecure and vulnerable in the presence of anti-social and evil elements. The criminals have been operating in an organized way and sometimes even have nationwide and international connections and links.

1.2 Goal

Much of the current work is focused in two major directions:

- Predicting surges and hotspots of crime, and
- Understanding patterns of criminal behavior that could help in solving criminal investigations.

1.3 Objective

The objective of our work is to:

- Predicting crime before it takes place.
- Predicting hotspots of crime.
- Understanding crime pattern.
- Classify crime based on location.
- Analysis of crime in SF.
-

1.4 Methodology

1.4.1 Machine learning

The term machine learning refers to the automated detection of meaningful patterns in data. In the past couple of decades, it has become a common tool in almost any task that requires information extraction from large data sets. We are surrounded by a machine learning based technology: search engines learn how to bring us the best results (while placing portable ads), anti-spam software learns to filter our email messages, and credit card transactions are secured by a software that learns how to detect frauds. Digital cameras learn to detect faces and

intelligent personal assistance applications on smart-phones learn to recognize voice commands. Cars are equipped with accident prevention systems that are built using machine learning algorithms.

Machine learning is also widely used in scientific applications such as bioinformatics, medicine, and astronomy. One common feature of all of these applications is that, in contrast to more traditional uses of computers, in these cases, due to the complexity of the patterns that need to be detected, a human programmer cannot provide an explicit, fine detailed specification of how such tasks should be executed. Taking example from intelligent beings, many of our skills are acquired or reined through learning from our experience (rather than following explicit instructions given to us). Machine learning tools are concerned with endowing programs with the ability to learn and adapt

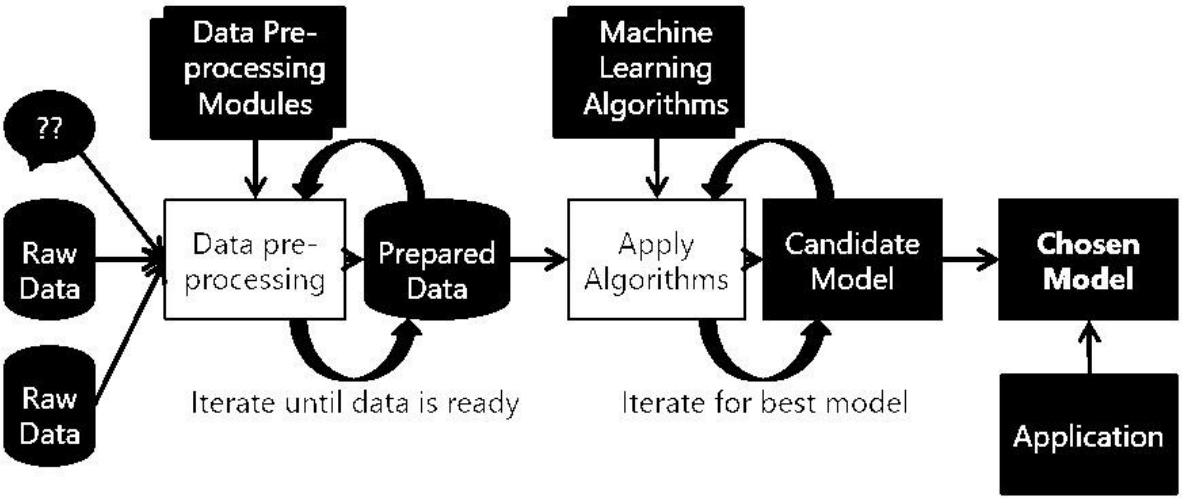


Fig 1.1-Machine learning process

1.4.2 Our Dataset

From 1934 to 1963, San Francisco was infamous for housing some of the world's most notorious criminals on the inescapable island of Alcatraz.

Today, the city is known more for its tech scene than its criminal past. But, with rising wealth inequality, housing shortages, and a proliferation of expensive digital toys riding BART to work, there is no scarcity of crime in the city by the bay.

From all areas in the area, this dataset provides nearly 15 years of crime reports from across all of San Francisco's neighborhoods. Given time and location, you must predict the category of crime that occurred.

This dataset is brought to you by SF OpenData (https://datasf.org/opendata/), the central clearinghouse for data published by the City and County of San Francisco.

This dataset contains incidents derived from SFPD Crime Incident Reporting system. The data ranges from 1/1/2003 to May of 2018.

2003-01-07 07:52:00	WARRANTS	WARRANT ARREST	Tuesday	SOUTHERN	ARREST, BOOKED	5TH ST / SHIPLEY ST	-122.402843	37.779829
2003-01-07 04:49:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Tuesday	TENDERLOIN	ARREST, BOOKED	CYRIL MAGNIN STORTH ST / EDDY ST	-122.408495	37.784452
2003-01-07 03:52:00	WARRANTS	WARRANT ARREST	Tuesday	NORTHERN	ARREST, BOOKED	OFARRELL ST / LARKIN ST	-122.417904	37.785167
2003-01-07 03:34:00	WARRANTS	WARRANT ARREST	Tuesday	NORTHERN	ARREST, BOOKED	DIVISADERO ST / LOMBARD ST	-122.442650	37.798999
2003-01-07 01:22:00	WARRANTS	WARRANT ARREST	Tuesday	SOUTHERN	ARREST, BOOKED	900 Block of MARKET ST	-122.409537	37.782691
2003-01-06 23:30:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	BAYVIEW	ARREST, BOOKED	REVERE AV / INGALLS ST	-122.384557	37.728487
2003-01-06 23:14:00	WARRANTS	WARRANT ARREST	Monday	CENTRAL	ARREST, BOOKED	BUSH ST / HYDE ST	-122.417019	37.789110
2003-01-06 22:45:00	WARRANTS	WARRANT ARREST	Monday	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421
2003-01-06 22:45:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421
2003-01-06 22:19:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	NORTHERN	ARREST, BOOKED	GEARY ST / POLK ST	-122.419740	37.785893
2003-01-06 21:54:00	WARRANTS	ENROUTE TO OUTSIDE JURISDICTION	Monday	NORTHERN	ARREST, BOOKED	SUTTER ST / POLK ST	-122.420120	37.787757

Data fields present:

- Dates - timestamp of the crime incident
- Category - category of the crime incident (only in train.csv). This is the target variable you are going to predict.
- Descript - detailed description of the crime incident (only in train.csv)
- DayOfWeek - the day of the week
- PdDistrict - name of the Police Department District
- Resolution - how the crime incident was resolved (only in train.csv)
- Address - the approximate street address of the crime incident
- X - Longitude
- Y – Latitude
- Others which are not so important for us, that includes updates, and other details.

1.4.3 Preprocessing

Before implementing machine learning algorithms on our data, we went through a series of preprocessing steps with our classification task in mind. These included:

- Dropping features such address, station, Complainant address, Accused found guilty or not.
- Dropping features such as Resolution, Description and Address: The resolution and description of a crime are only known once the crime has occurred, and have limited significance in a practical, real-world scenario where one is trying to predict what kind of crime has occurred, and so, these were omitted. The address was dropped because we had information about the latitude and longitude, and, in that context, the address did not add much marginal value.
- The timestamp contained the year, date and time of occurrence of each crime. This was decomposed into five features: Year (2018), Month (1-12), Date (1-31), Hour (023) and Minute (0-59).

Following these preprocessing steps, we ran some out-of-the box learning algorithms as a part of our initial exploratory steps

1.4.4 Methodology

After the preprocessing described in the previous sections, we had three different classifications problems to solve, which we proceeded to attack with an assortment of classification algorithms. The following are the algorithms which we are using:

- KNN (K- Nearest neighbors)
- Decision Tree
- Random Forests
- LSTM
- FACEBOOK PROPHET

1.6.1 Market potential

The use of AI and machine learning to detect crime via sound or cameras currently exists, is proven to work, and expected to continue to expand.

The use of AI/ML in predicting crimes or an individual's likelihood for committing a crime has promise but is still more of an unknown. The biggest challenge will probably be "proving" to politicians that it works. When a system is designed to stop something from happening, it is difficult to prove the negative. Companies that are directly involved in providing governments with AI tools to monitor areas or predict crime will likely benefit from a positive feedback loop. Improvements in crime prevention technology will likely spur increased total spending on this technology.

Possible avenues through which to extend this work include time-series modeling of the data to understand temporal correlations in it, which can then be used to predict surges in different categories of crime. It would also be interesting to explore relationships between surges in different categories of crimes.

For Example: it could be the case that two or more classes of crimes surge and sink together, which would be an interesting relationship to uncover. Other areas to work on include implementing a more accurate multi-class classifier, and exploring better ways to visualize our results.

1.6.2 Innovativeness

The idea behind this project is that crimes are relatively predictable; it just requires being able to sort through a massive volume of data to find patterns that are useful to law enforcement. This kind of data analysis was technologically impossible a few decades ago, but the hope is that recent developments in machine learning are up to the task.

1.6.3 Usefulness

Public safety and protection relate to crime, and a better understanding of crime is beneficial in multiple ways: it can lead to targeted and sensitive practices by law enforcement authorities to mitigate crime, and more concerted efforts by citizens and authorities to create healthy neighborhood environments. With the advent of the Big Data era and the availability of fast, efficient algorithms for data analysis, understanding patterns in crime from data is an active and growing field of research.

Chapter-2
Requirement Engineering

2.1 Functional Requirement

The functional requirements describe the core functionality of the project.

2.1.1 Platform Requirement:

- Python was used as the language on which the entire project is done.
- Anaconda with TensorFlow GPU enabled since that was very fast.
- Google Colab was also used because of the mobility and the fast hardware that they provide.

2.2 Non-Functional Requirement

Non function requirements are those requirements of the system which are not directly concerned with specific functional delivered by the system. They may be related to emergent properties such as reliability, extendibility, usability.

- To provide prediction of crime.
- To provide maximum accuracy.
- Provide visualized analysis.
- Ease of use.
- Availability
- Reliability
- Maintainability

Chapter-3
Analysis and Design

3.4 System architecture

The system architectural design is the design process for identifying the subsystems making up the system and framework for subsystem control and communication. The goal of the architectural design is to establish the overall structure of software system.

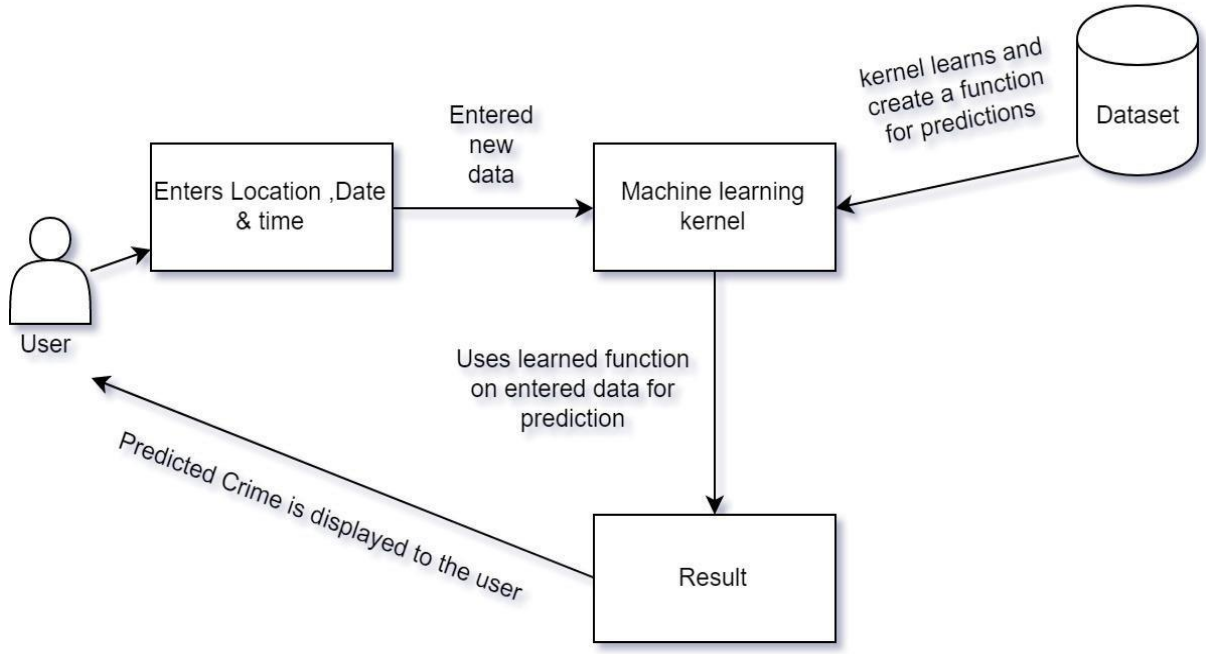


Fig 3.4-System architecture

Chapter-4
Construction

4.1 Implementation

The implementation of the project is done with the help of python language. To be particular, for the purpose of machine learning Google Colab and Anaconda are used.

Anaconda is an alternative and one of several Python distributions. Anaconda is a new distribution of the Python. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages. Anaconda is used for scientific computing, data science, statistical analysis, and machine learning. But all these can be very easily used in Google colab along with device portability and only internet connection is required.

On Python technology, I found out Colab to be easier. Since it helps with the following problems automatically:

- Installing Python on multiple platforms.
- Separating out different environments.
- Dealing with not having correct privileges.
- Getting up and running with specific packages and libraries.

This data was scraped from the publicly available data from SF police website which had been made by people in police station of different areas. Implementation of the idea started from the Indore city itself so as to limit an area for the prediction and making it less complex. The data was sorted and converted into a new format of timestamp, longitude, latitude, which was the input that machine would be taking so as to predict the crime rate in particular location or city.

The entries were done just to make the machine learn what all it has to do with the data and what actually the output is being demanded. As soon as the machine learnt the algorithms and the process, accuracy of different algorithms was measured & the algorithm with the most accuracy is used for the prediction kernel i.e. Random forest and other algorithms that I mentioned earlier and splits dataset based on it. It is done to reach a stage where we have homogenous subsets that are giving predictions with utmost surety.

4.4.1 Hardware Details (For Google Colab)

- Operating system: Any OS.
- System architecture: Any Architecture with an Internet Connection.
- CPU: Intel Core 2 Quad CPU Q6600 @ 2.40 GHz or greater.
- RAM: 2 GB or greater

4.4.2 Hardware Details (For Anaconda)

- Operating system: Any OS.
- System architecture: Any architecture with an Internet Connection.
- CPU: Intel® Core (TM) i7-8700k @ 3.70 GHz can be over clocked.
- GPU: NVIDIA GTX 1050 Ti
- RAM: 12 GB or greater

4.5 Testing

The development of this project involves a series of production activities were opportunities for injection of human fallibilities are enormous.

Error may begin to occur at very inspection of the process where the objective may be enormously or imperfectly specified as well as in lateral design and development stage. Because of human inability to perform and communicate with perfection, software development quality assurance activities.

The testing is a crucial element of this project's quality assurances and represents ultimate review of specification, design and coding.

4.5.1 White box testing

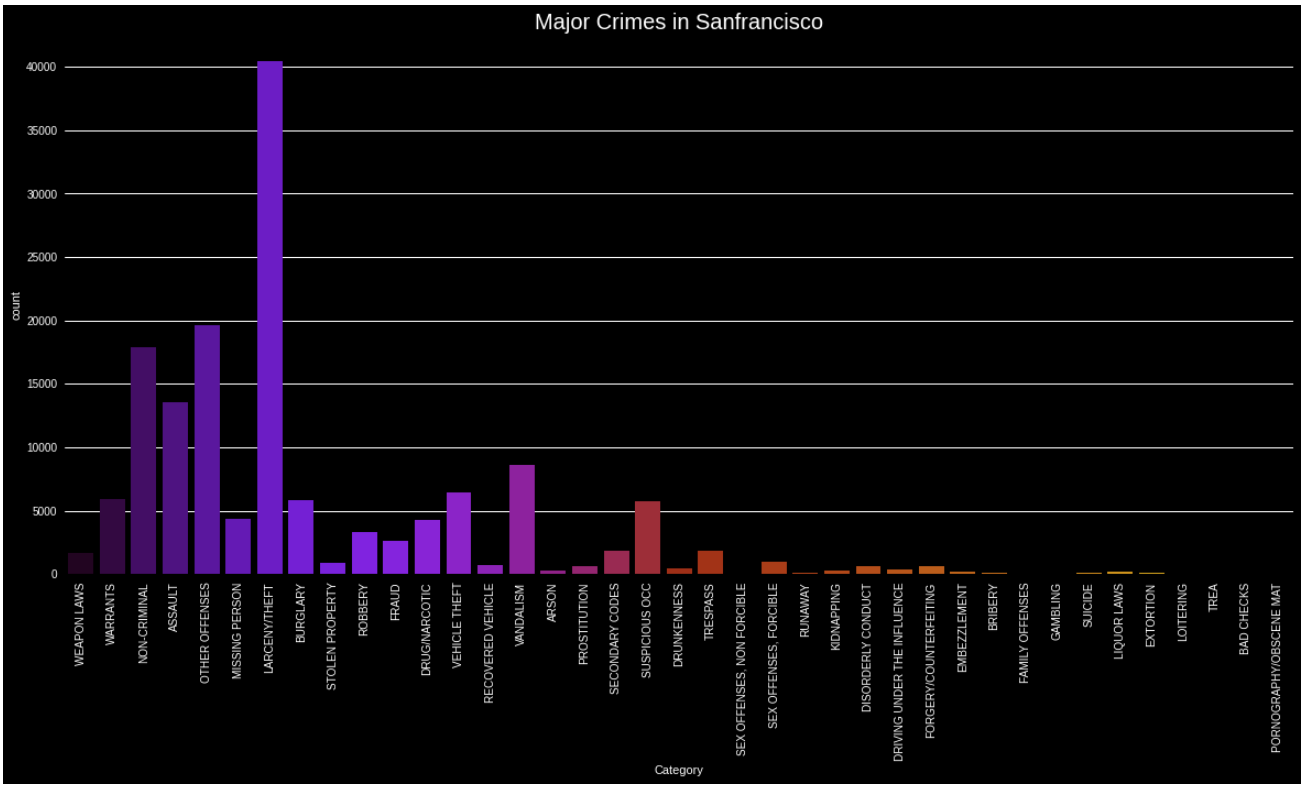
It focuses on the program control structure. Here all statement in the project have been executed at least once during testing and all logical condition have been exercised.

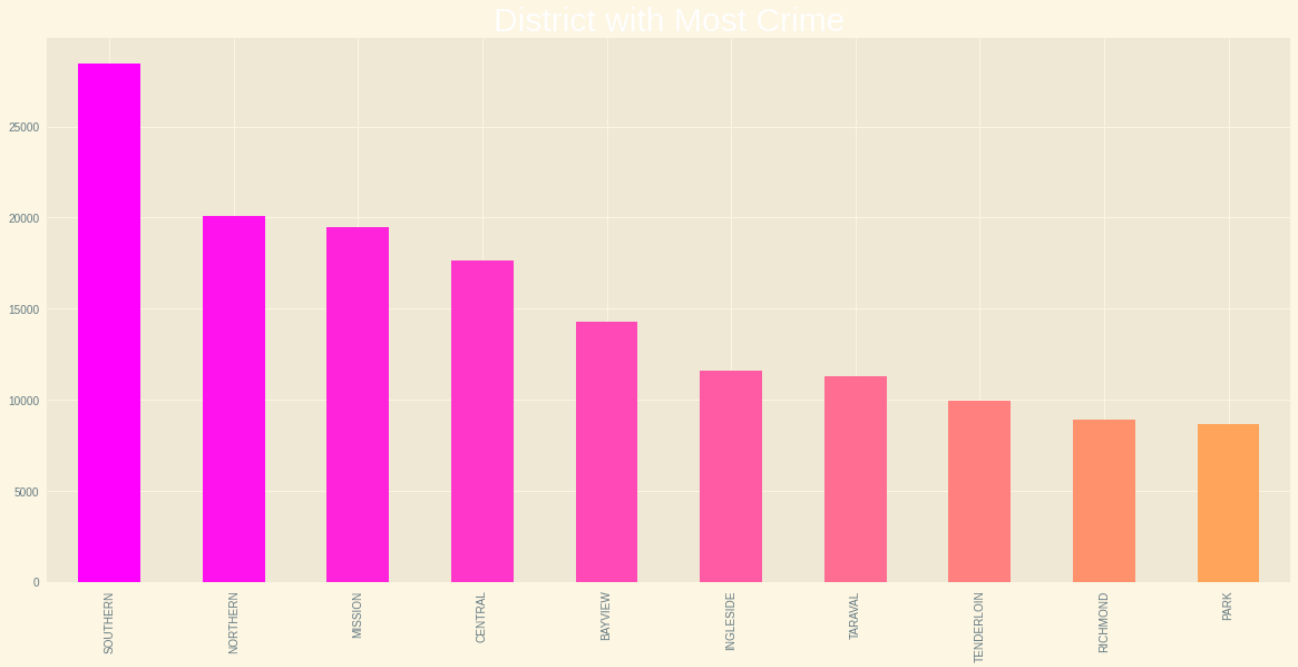
4.5.2 Black box testing

This is designed to uncover the error in functional requirements without regard to the internal working of the project. This testing focuses on the information domain of the project, deriving test case by partitioning the input and output domain of programming – A manner that provides through test coverage.

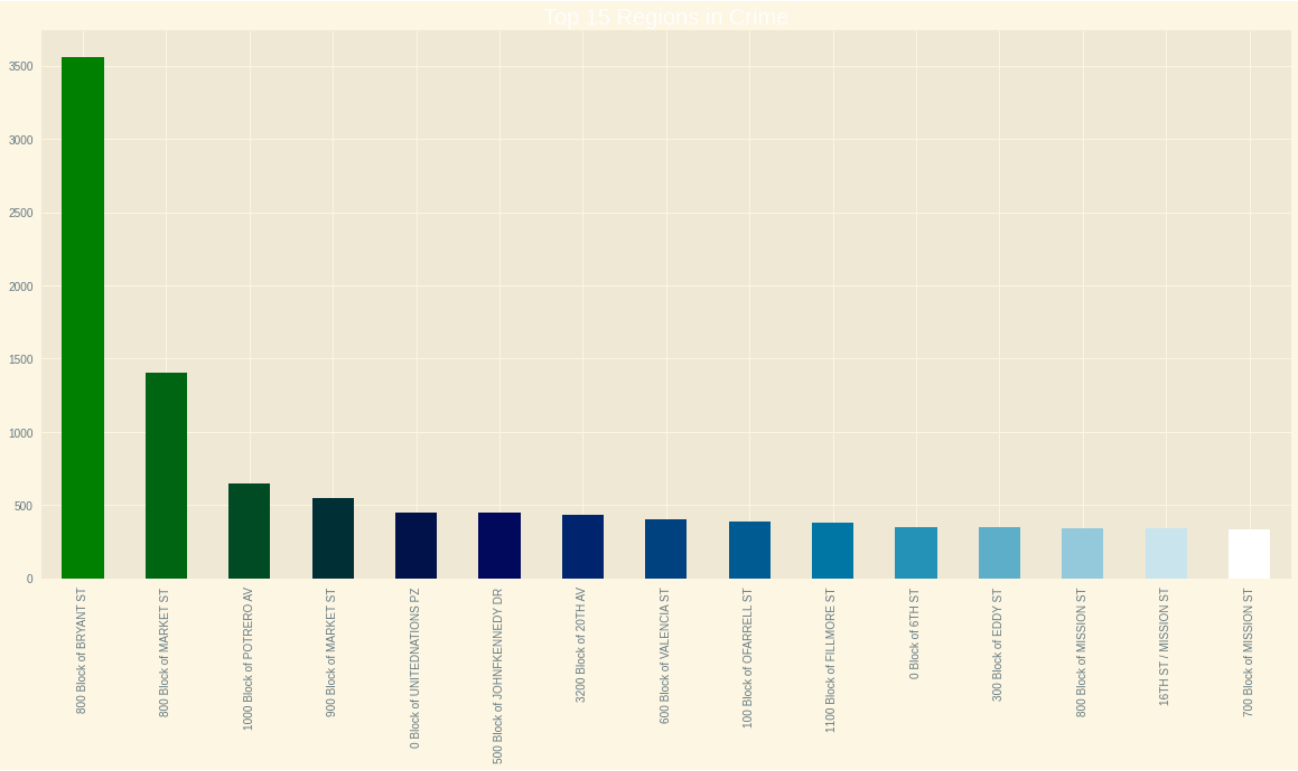
Things that I have done are directly implemented in Google Colab or Anaconda are provided to my guide.

The initial output is given below along with the output of each column listed immediately below:

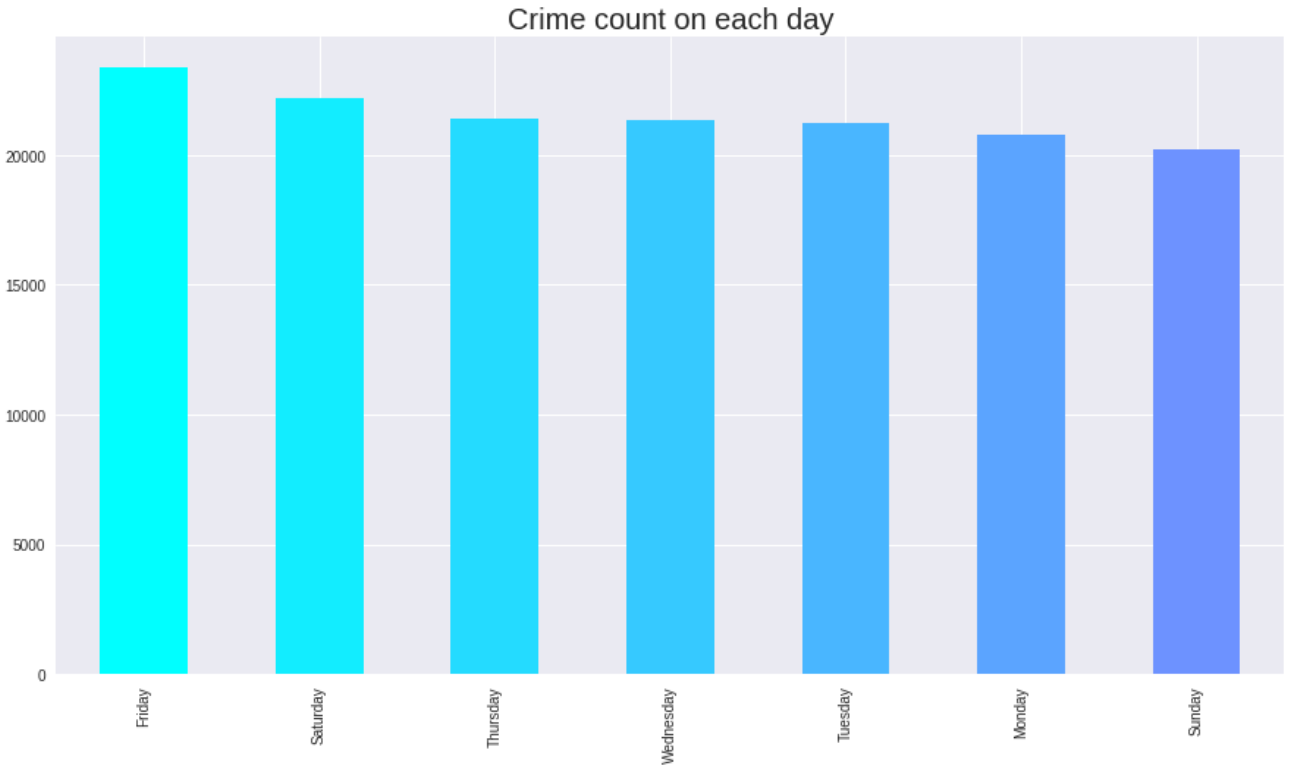


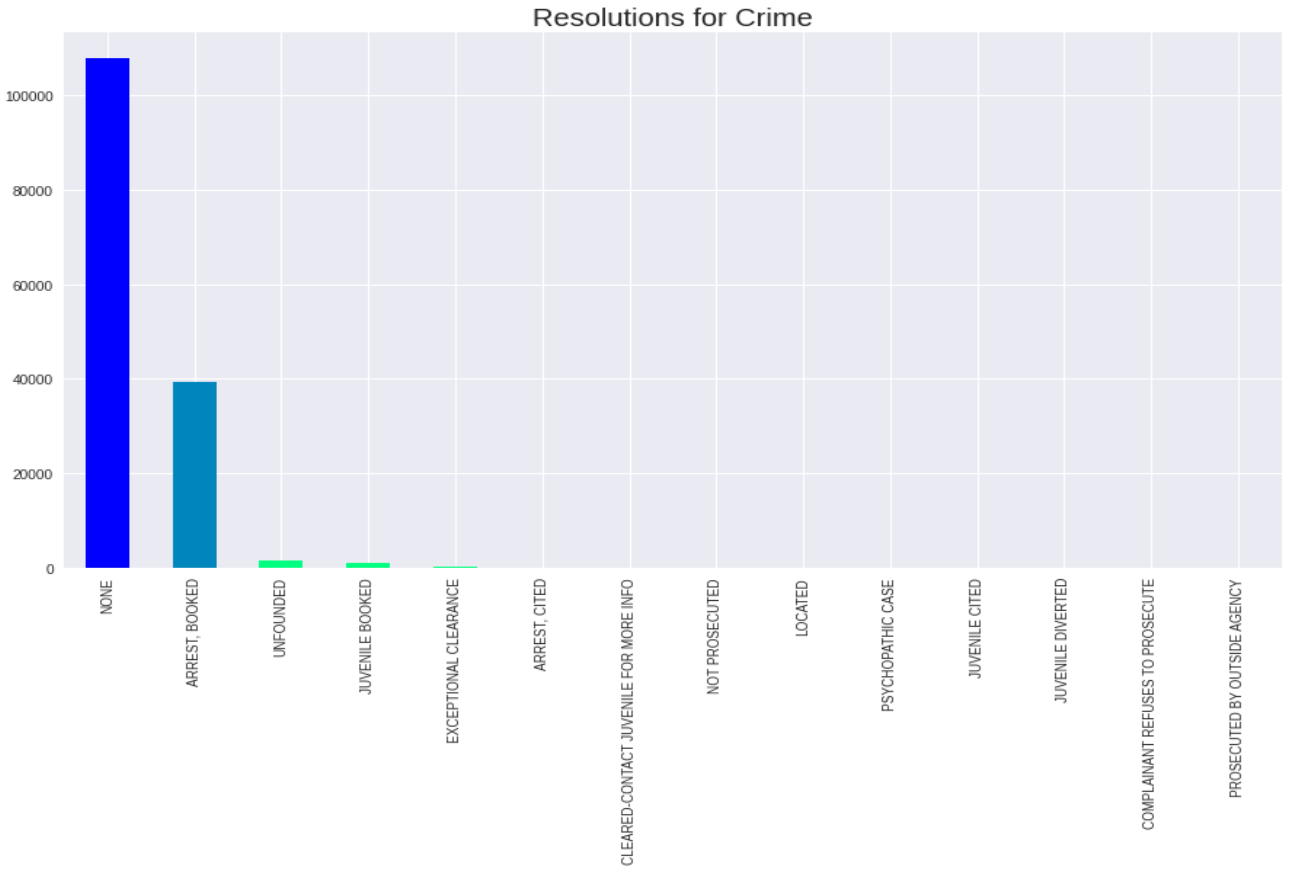


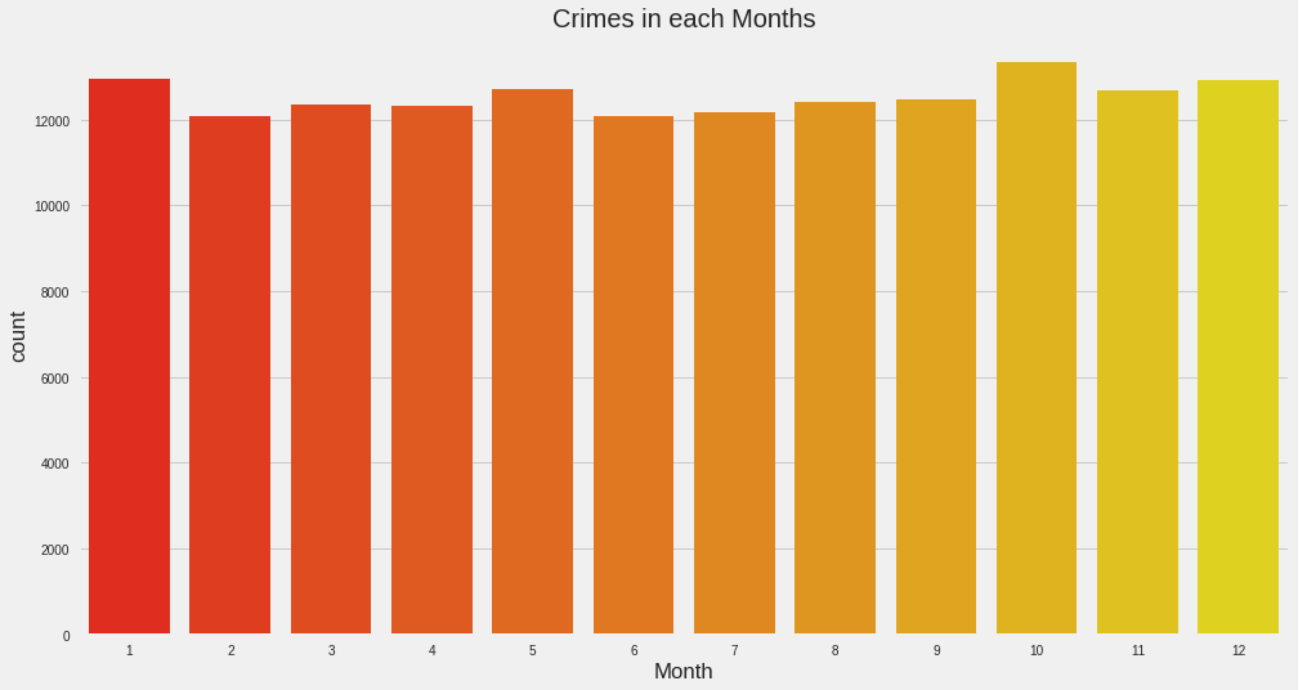
Crime Count vs Neighborhoods

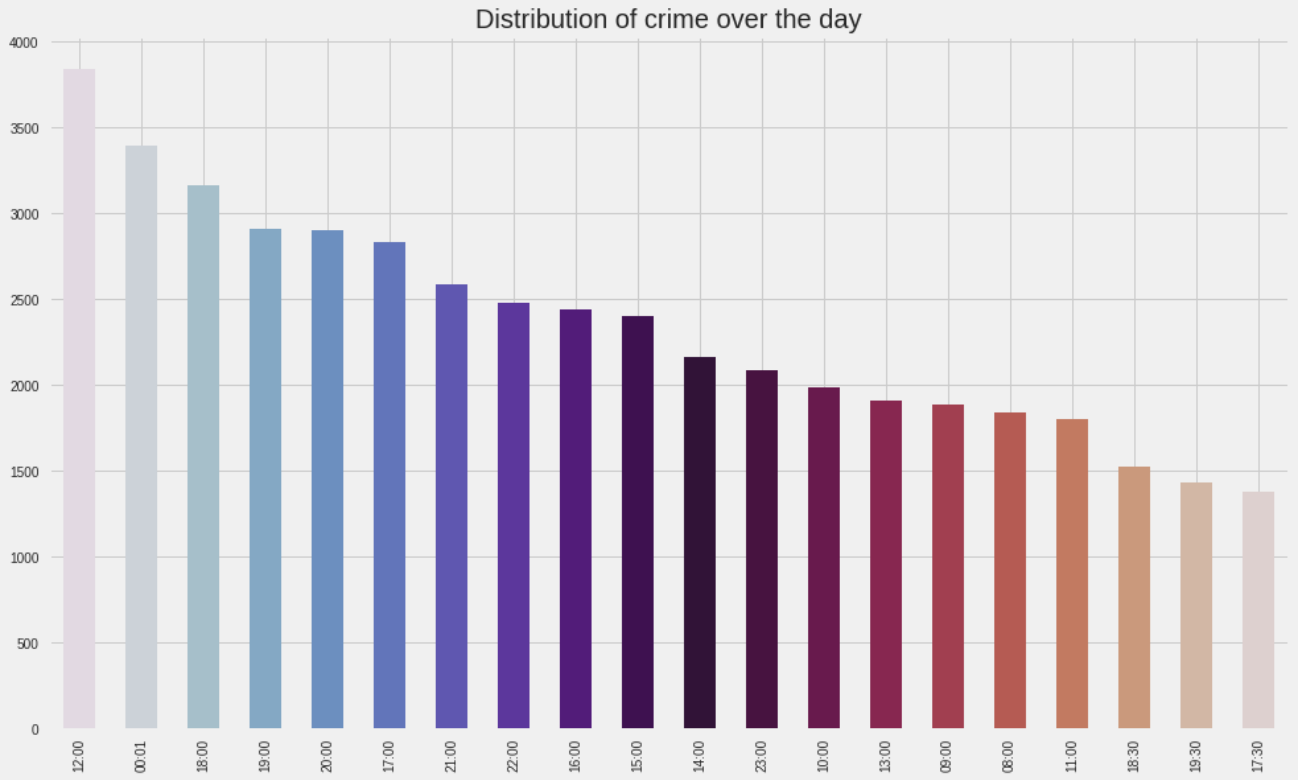


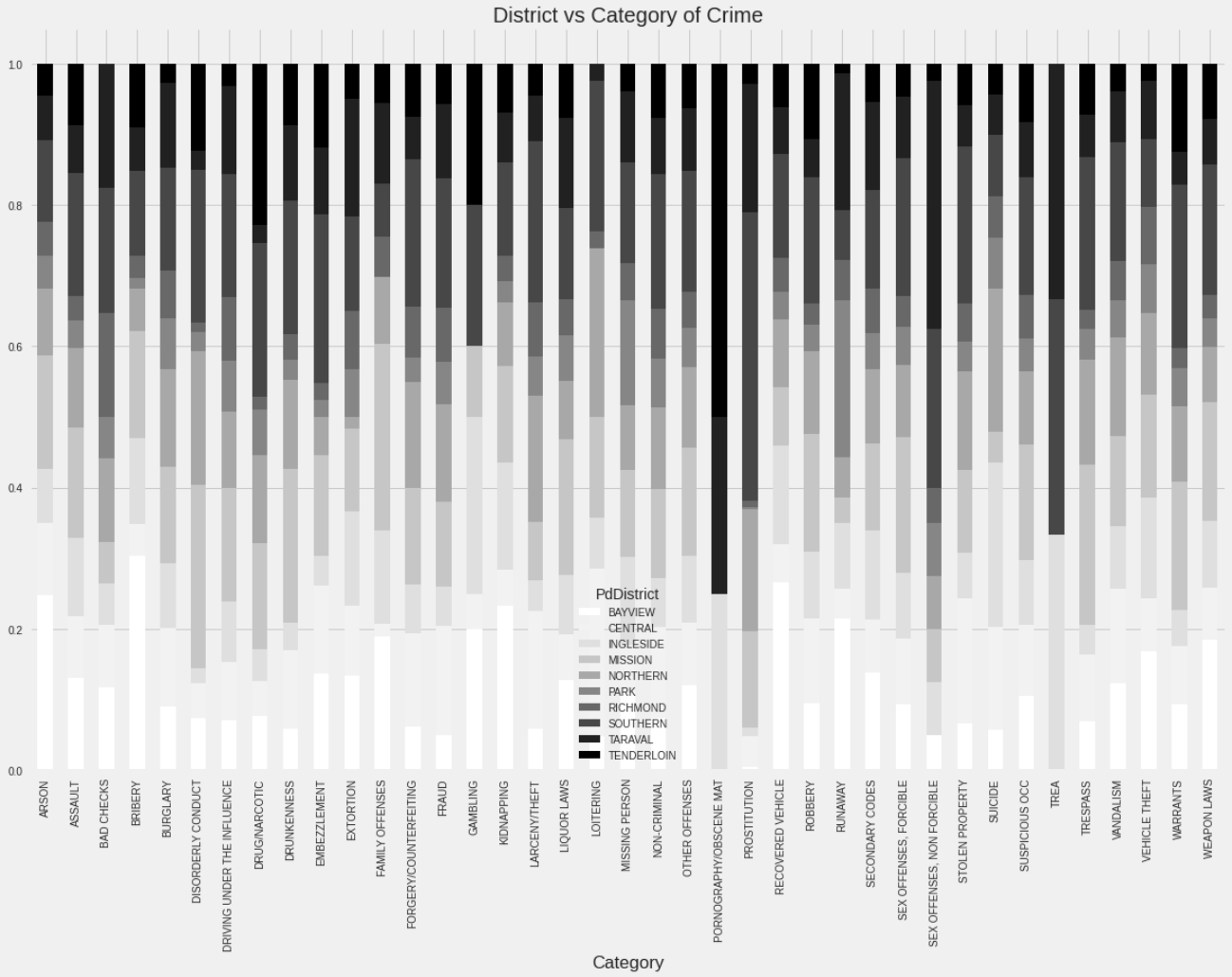
Crime Count vs Major Address Provided

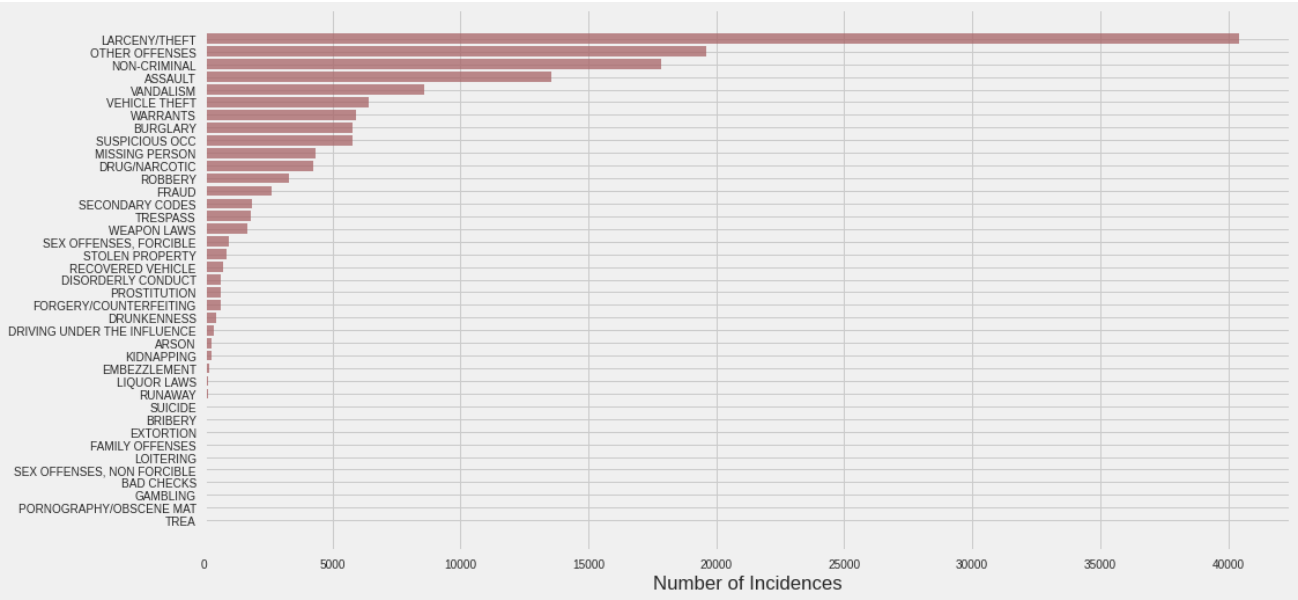

Crime count on each day
20000
15000
10000
5000
0
Friday
Saturday
Thursday
Wednesday
Tuesday
Monday
Sunday

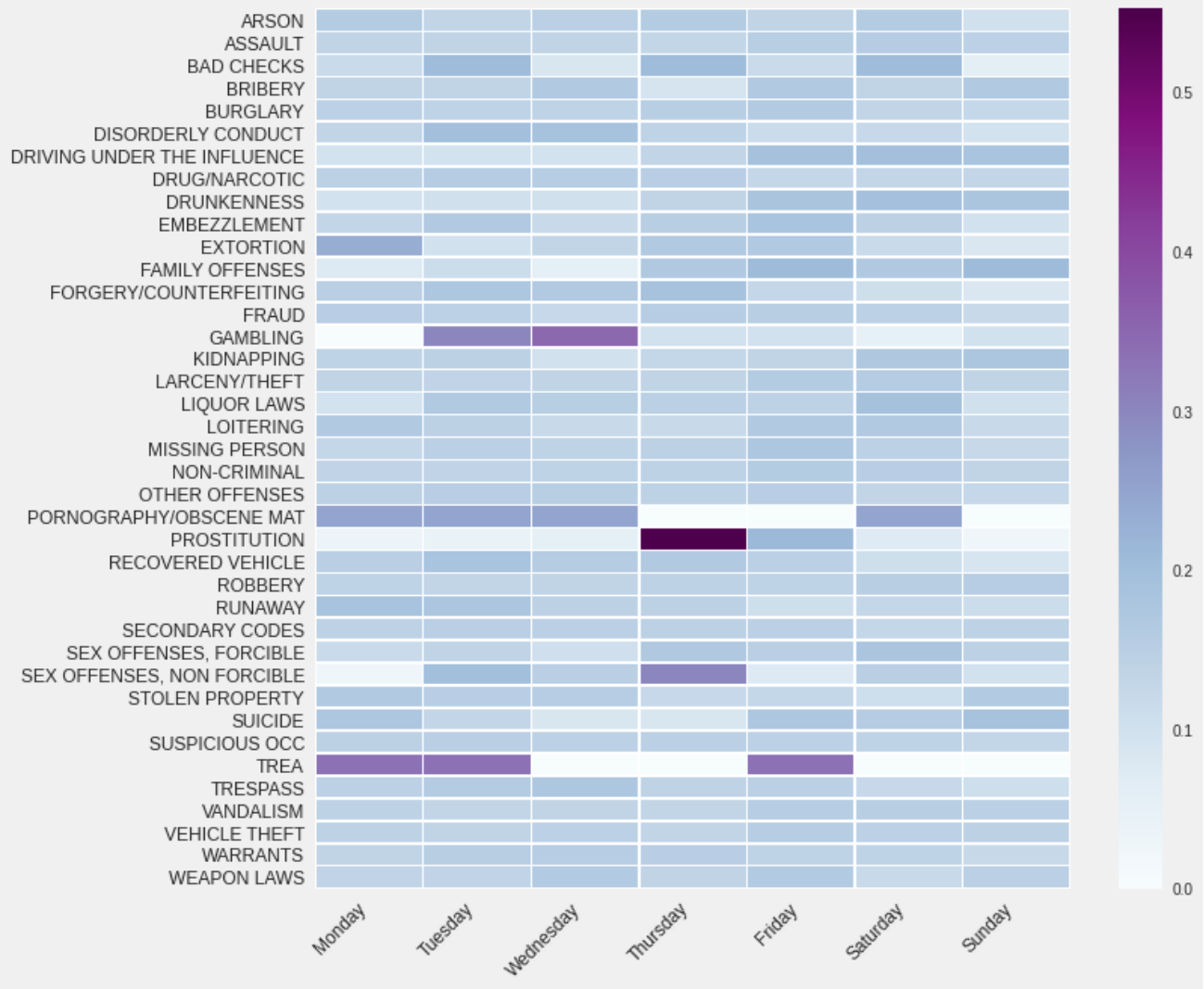

Resolutions for Crime
100000
80000
60000
40000
20000
0
NONE
ARREST, BOOKED
UNFOUNDED
JUVENILE BOOKED
EXCEPTIONAL CLEARANCE
ARREST, CITED
CLEARED-CONTACT JUVENILE FOR MORE INFO
NOT PROSECUTED
LOCATED
PSYCHOPATHIC CASE
JUVENILE CITED
JUVENILE DIVERTED
COMPLAINANT REFUSES TO PROSECUTE
PROSECUTED BY OUTSIDE AGENCY

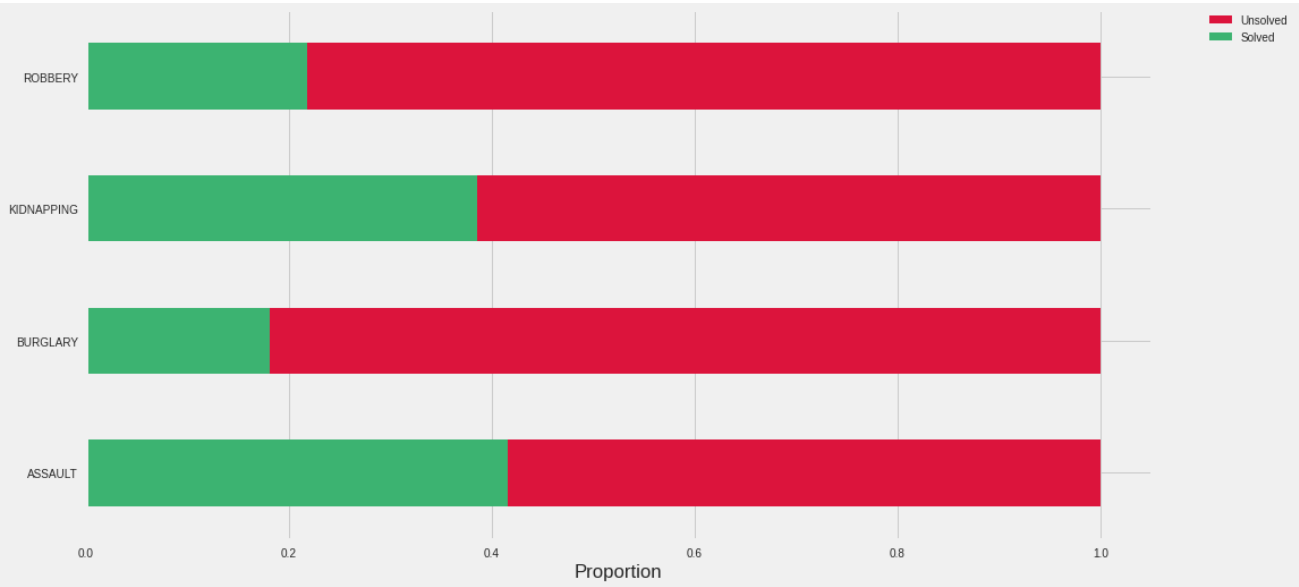

Crimes in each Months
count
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7
8
9
10
11
12
Month


Distribution of crime over the day
0
500
1000
1500
2000
2500
3000
3500
4000
12:00
00:01
18:00
19:00
20:00
17:00
21:00
22:00
16:00
15:00
14:00
23:00
10:00
13:00
09:00
08:00
11:00
18:30
19:30
17:30


District vs Category of Crime
1.0
0.8
0.6
0.4
0.2
0.0
PdDistrict
BAYVIEW
CENTRAL
INGLESIDE
MISSION
NORTHERN
PARK
RICHMOND
SOUTHERN
TARAVAL
TENDERLOIN
ARSON
ASSAULT
BAD CHECKS
BRIBERY
BURGLARY
DISORDERLY CONDUCT
DRIVING UNDER THE INFLUENCE
DRUG/NARCOTIC
DRUNKENNESS
EMBEZZLEMENT
EXTORTION
FAMILY OFFENSES
FORGERY/COUNTERFEITING
FRAUD
GAMBLING
KIDNAPPING
LARCENY/THEFT
LIQUOR LAWS
LOITERING
MISSING PERSON
NON-CRIMINAL
OTHER OFFENSES
PORNOGRAPHY/OBSCENE MAT
PROSTITUTION
RECOVERED VEHICLE
ROBBERY
RUNAWAY
SECONDARY CODES
SEX OFFENSES, FORCIBLE
SEX OFFENSES, NON FORCIBLE
STOLEN PROPERTY
SUICIDE
SUSPICIOUS OCC
TREA
TRESPASS
VANDALISM
VEHICLE THEFT
WARRANTS
WEAPON LAWS
Category


LARCENY/THEFT
OTHER OFFENSES
NON-CRIMINAL
ASSAULT
VANDALISM
VEHICLE THEFT
WARRANTS
BURGLARY
SUSPICIOUS OCC
MISSING PERSON
DRUG/NARCOTIC
ROBBERY
FRAUD
SECONDARY CODES
TRESPASS
WEAPON LAWS
SEX OFFENSES, FORCIBLE
STOLEN PROPERTY
RECOVERED VEHICLE
DISORDERLY CONDUCT
PROSTITUTION
FORGERY/COUNTERFEITING
DRUNKENNESS
DRIVING UNDER THE INFLUENCE
ARSON
KIDNAPPING
EMBEZZLEMENT
LIQUOR LAWS
RUNAWAY
SUICIDE
BRIBERY
EXTORTION
FAMILY OFFENSES
LOITERING
SEX OFFENSES, NON FORCIBLE
BAD CHECKS
GAMBLING
PORNOGRAPHY/OBSCENE MAT
TREA
0
5000
10000
15000
20000
25000
30000
35000
40000
Number of Incidences


ARSON
ASSAULT
BAD CHECKS
BRIBERY
BURGLARY
DISORDERLY CONDUCT
DRIVING UNDER THE INFLUENCE
DRUG/NARCOTIC
DRUNKENNESS
EMBEZZLEMENT
EXTORTION
FAMILY OFFENSES
FORGERY/COUNTERFEITING
FRAUD
GAMBLING
KIDNAPPING
LARCENY/THEFT
LIQUOR LAWS
LOITERING
MISSING PERSON
NON-CRIMINAL
OTHER OFFENSES
PORNOGRAPHY/OBSCENE MAT
PROSTITUTION
RECOVERED VEHICLE
ROBBERY
RUNAWAY
SECONDARY CODES
SEX OFFENSES, FORCIBLE
SEX OFFENSES, NON FORCIBLE
STOLEN PROPERTY
SUICIDE
SUSPICIOUS OCC
TREA
TRESPASS
VANDALISM
VEHICLE THEFT
WARRANTS
WEAPON LAWS
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
0.5
0.4
0.3
0.2
0.1
0.0


Unsolved
Solved
ROBBERY
KIDNAPPING
BURGLARY
ASSAULT
0.0
0.2
0.4
0.6
0.8
1.0
Proportion

Chapter: 5

Finding better solutions for this task are:

For the purpose of proper implementation and functioning several Algorithms and techniques were used. Following are the algorithms used:

5.1 KNN (K-Nearest neighbors)

A powerful classification algorithm used in pattern recognition K nearest neighbors stores all available cases and classifies new cases based on a similarity measure (e.g. distance function). One of the top data mining algorithms used today. A non-parametric lazy learning algorithm (An Instance based Learning method).

KNN: Classification Approach

- An object (a new instance) is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its K nearest neighbors (measured by distance function).

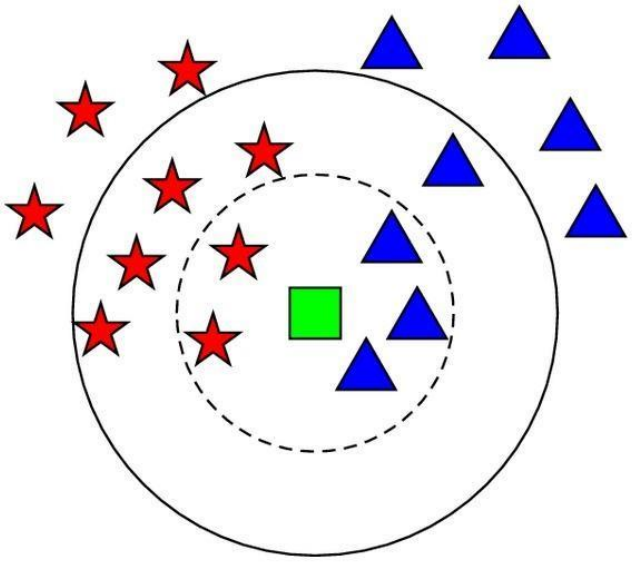


Fig 5.1.1 Principle diagram of KNN

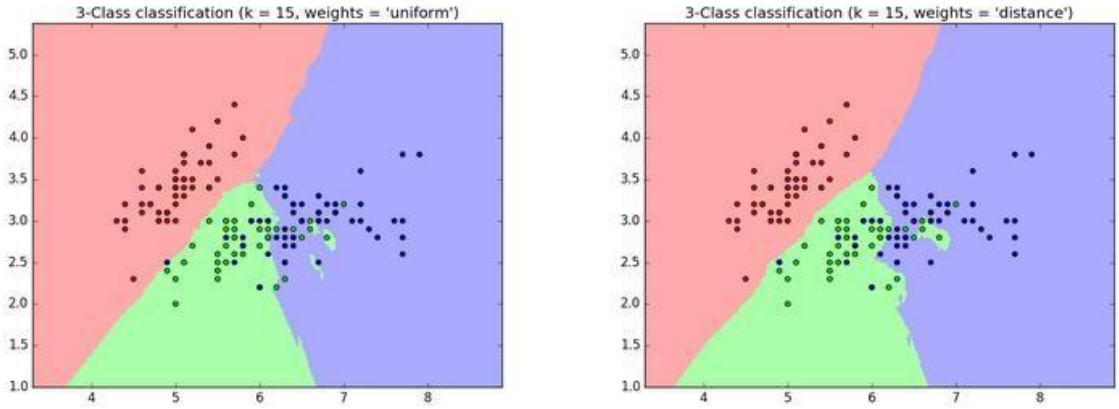


Fig 5.1.2 Shows graphical representation of KNN

Some frequently used distance functions.

Camberra :

$$d(x,y)=\sum_{i=1}^{m}\frac{|x_i-y_i|}{|x_i+y_i|} \quad (2)$$

Minkowsky :

$$d(x,y)=\left(\sum_{i=1}^{m}|x_i-y_i|^r\right)^{1/r} \quad (3)$$

Chebychev :

$$d(x,y)=\max_{i=1}^{m}|x_i-y_i| \quad (4)$$

Euclidean :

$$d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2} \quad (5)$$

Manhattan / city - block :

$$d(x,y)=\sum_{i=1}^{m}|x_i-y_i| \quad (6)$$

Fig 5.1.3 Distance functions

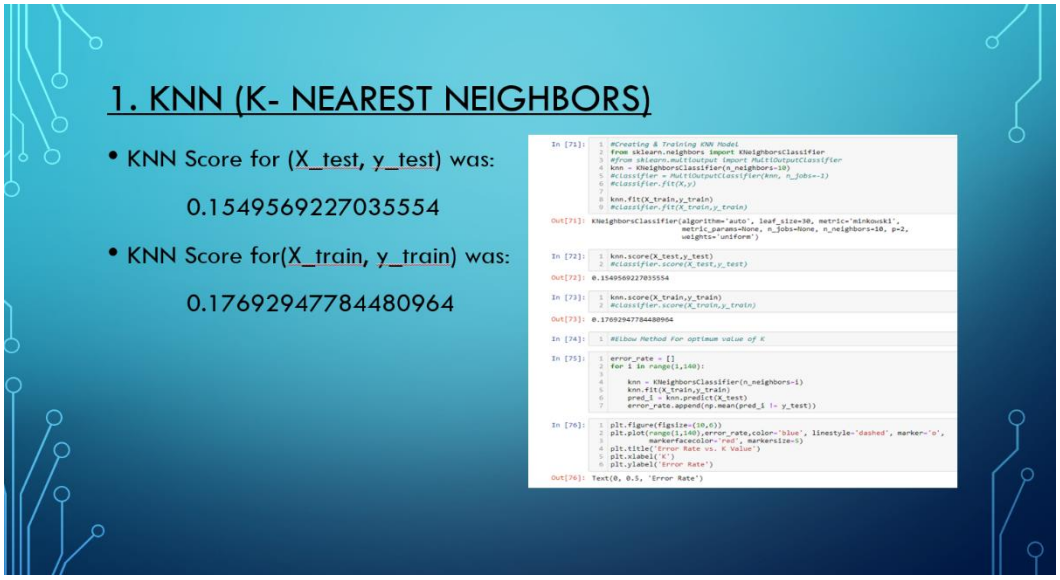


Fig 5.1.4 KNN score:

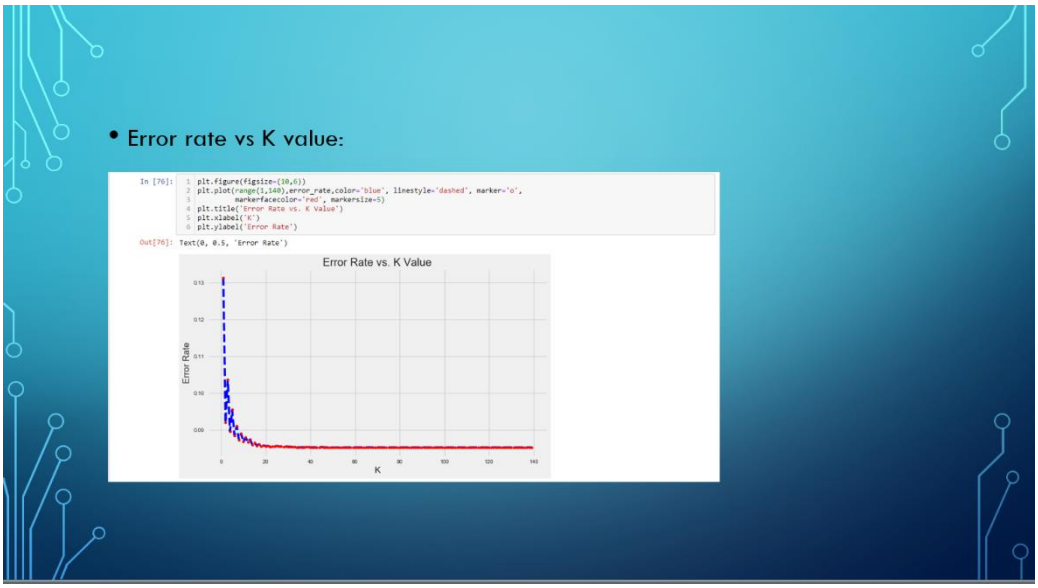


Fig 5.1.5 ERROR RATE VS K VALUE:

5.2 Decision Tree

As the name says all about it, it is a tree which helps us by assisting us in decision-making. Used for both classification and regression, it is a very basic and important predictive learning algorithm.

- It is different from others because it works intuitively i.e., taking decisions one-by-one.
- Non-parametric: Fast and efficient.

It consists of nodes which have parent-child relationships:

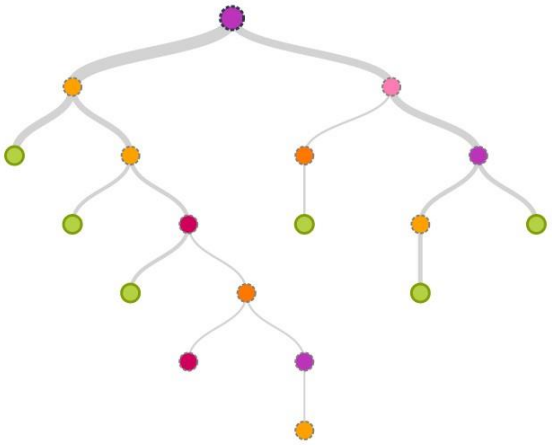


Fig 5.2.1 Decision tree

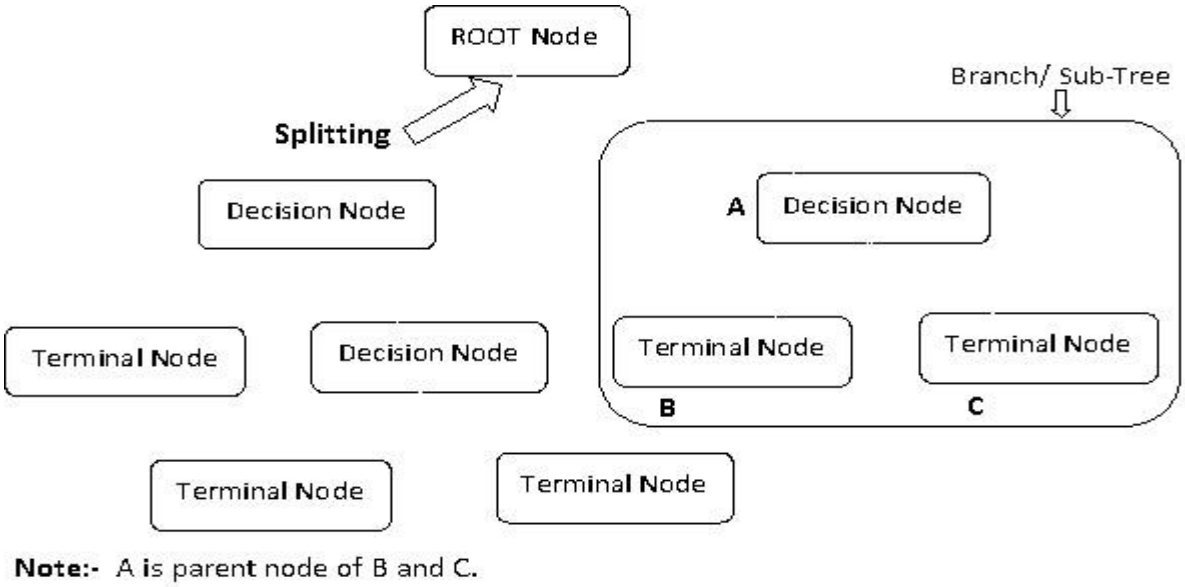


Fig 5.2.2 Decision Tree example

Decision tree considers the most important variable using some fancy criterion and splits dataset based on it. It is done to reach a stage where we have **homogenous subsets** that are giving predictions with utmost surety.

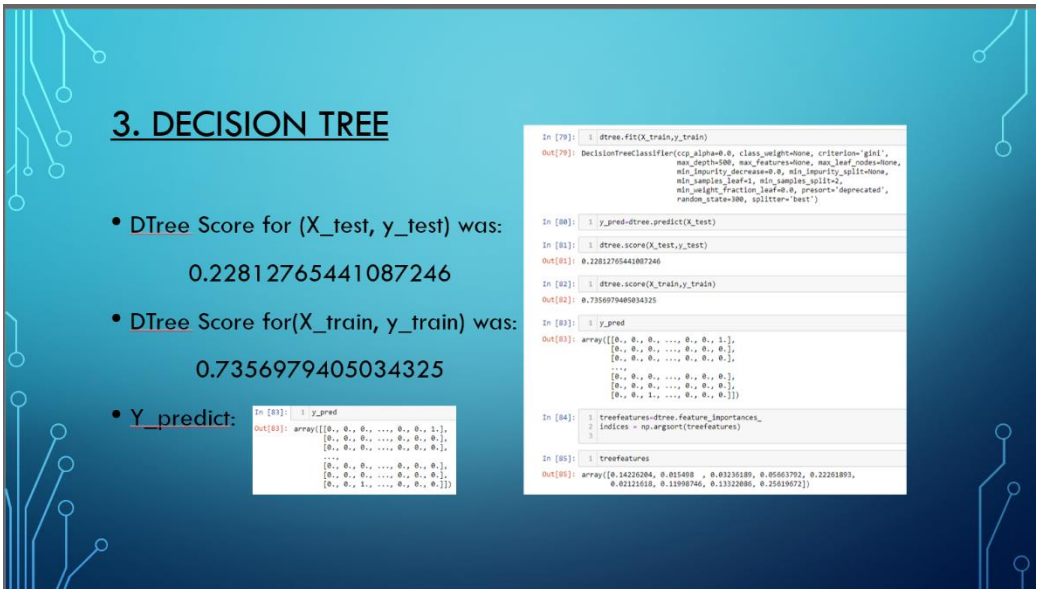


Fig 5.2.3 Decision tree score and the Y predict.

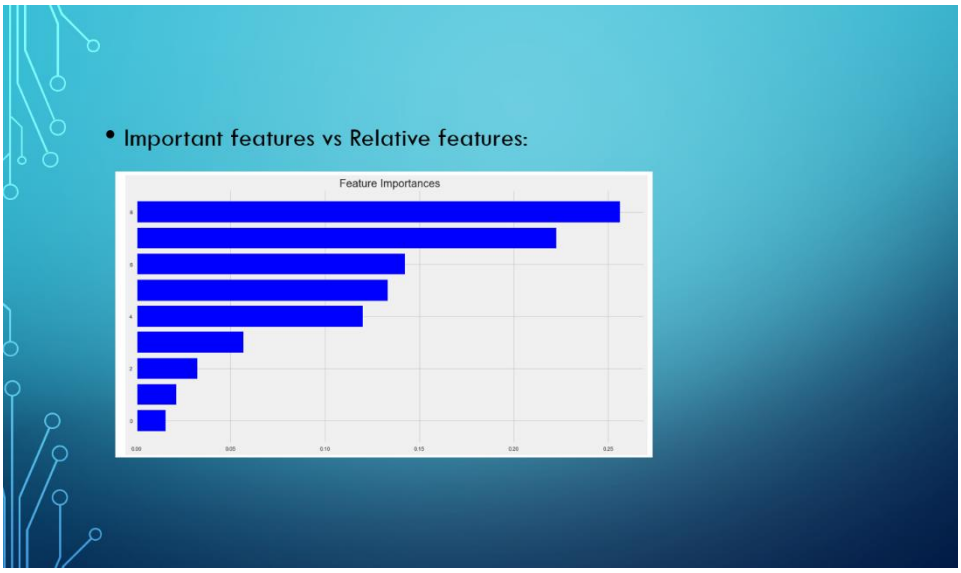


Fig 5.2.4 Relative features vs Important features.

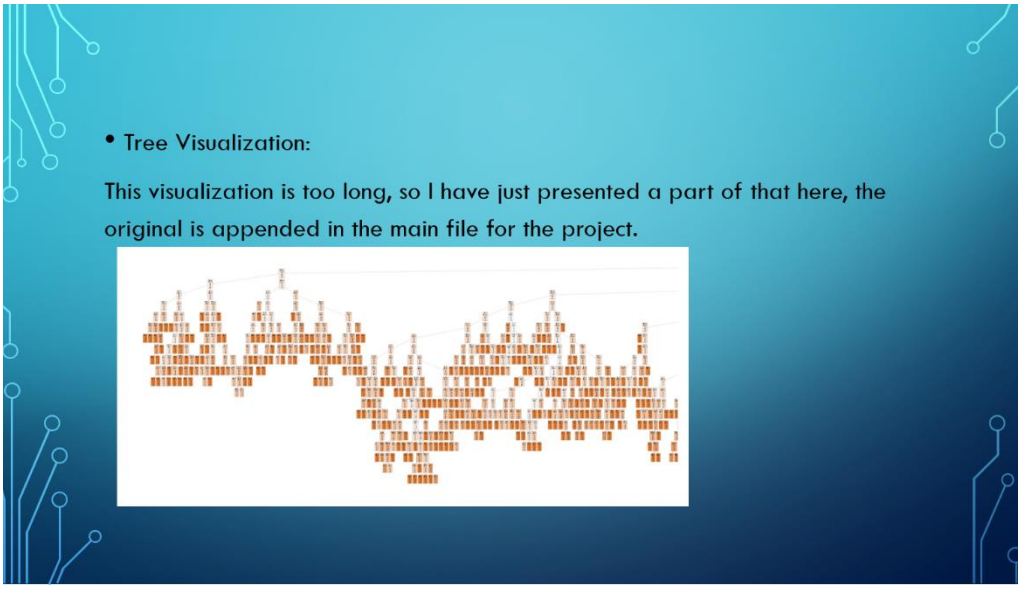


Fig 5.2.5 The Tree visualization.

5.3 Random forest

Random Forests is a very popular ensemble learning method which builds a number of classifiers on the training data and combines all their outputs to make the best predictions on the test data.

Thus, the Random Forests algorithm is a variance minimizing algorithm that uses randomness when making split decision to help avoid overfitting on the training data.

A random forests classifier is an ensemble classifier, which aggregates a family of classifiers h(x|θ1),h(x|θ2),..h(x|θk). Each member of the family, h(x|θ), is a classification tree and k is the number of trees chosen from a model random vector.

Also, each θk is a randomly chosen parameter vector. If D(x,y) denotes the training dataset, each classification tree in the ensemble is built using a different subset Dθk(x,y) ⊂ D(x,y) of the training dataset.

Thus, h(x|θk) is the kth classification tree which uses a subset of features xθk ⊂ x to build a classification model. Each tree then works like regular decision trees: it partitions the data based on the value of a particular feature (which is selected randomly from the subset), until the data is fully partitioned, or the maximum allowed depth is reached. The final output y is obtained by aggregating the results thus:

$$y = \text{argmax}_{p \in \{h(x_1)..h(x_k)\}} \{ \sum_{j=1}^{\kappa} (I(h(x|\theta_j) = p)) \}$$

where I denotes the indicator function.

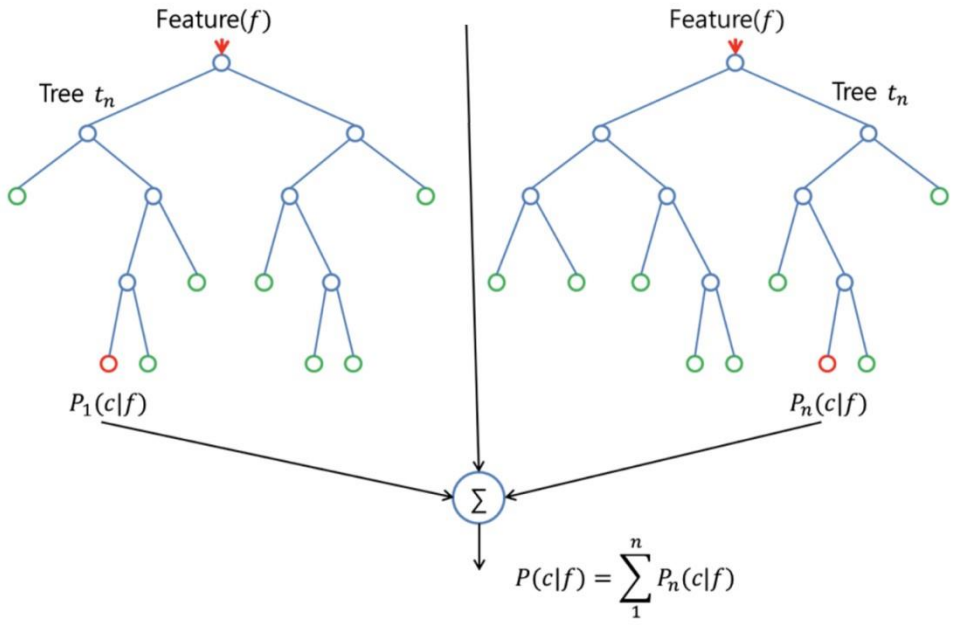


Fig 5.3.1 Random Forest Example

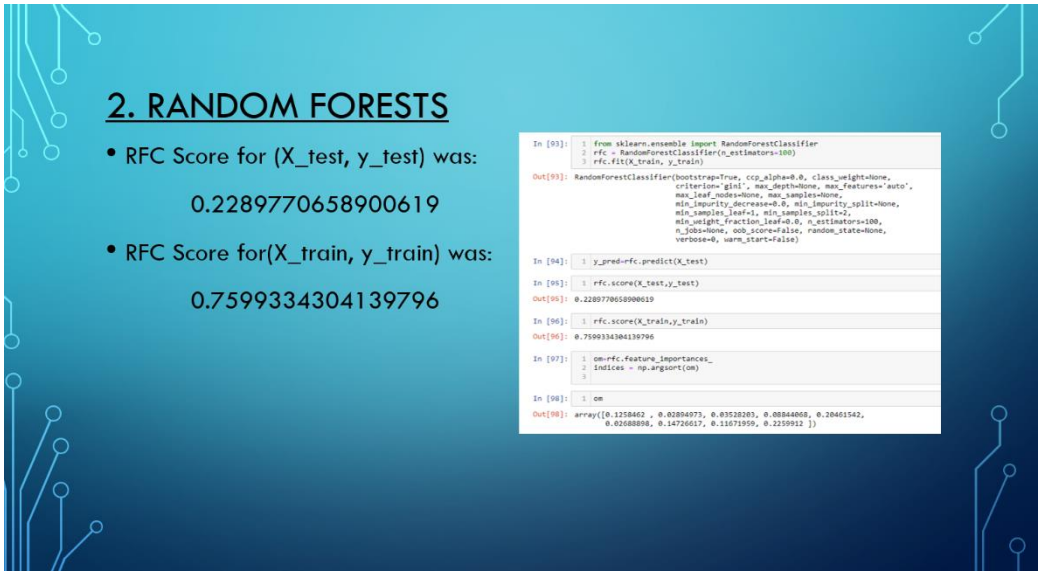


Fig 5.3.2 The output score of Random Forest

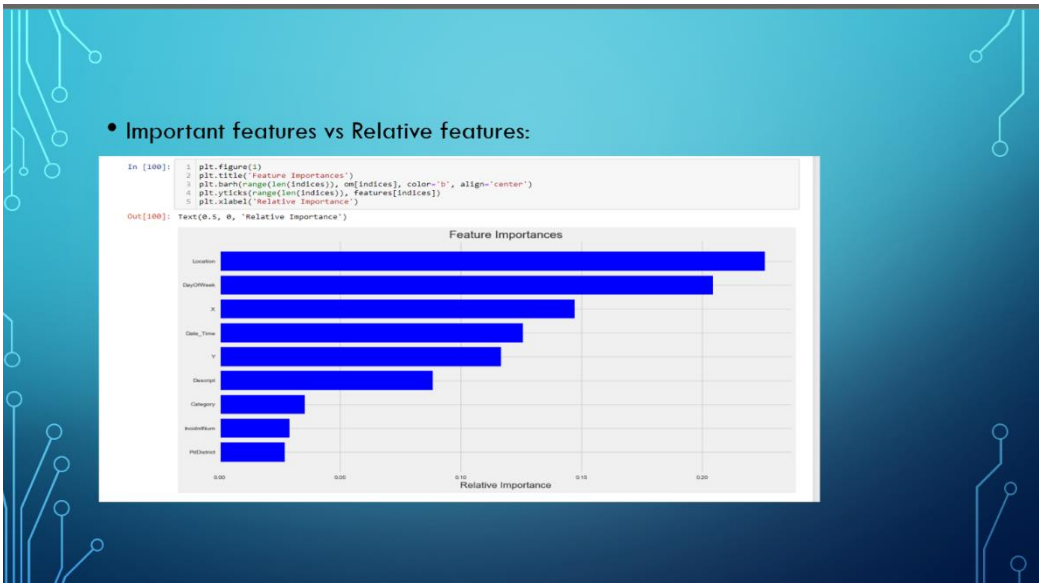
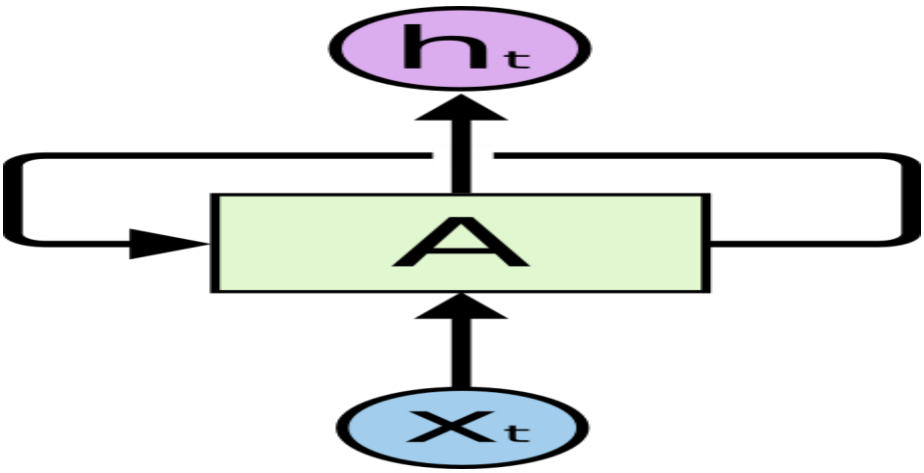


Fig 5.3.3 Important features vs Relative features.

5.4 Long Short-Term Memory

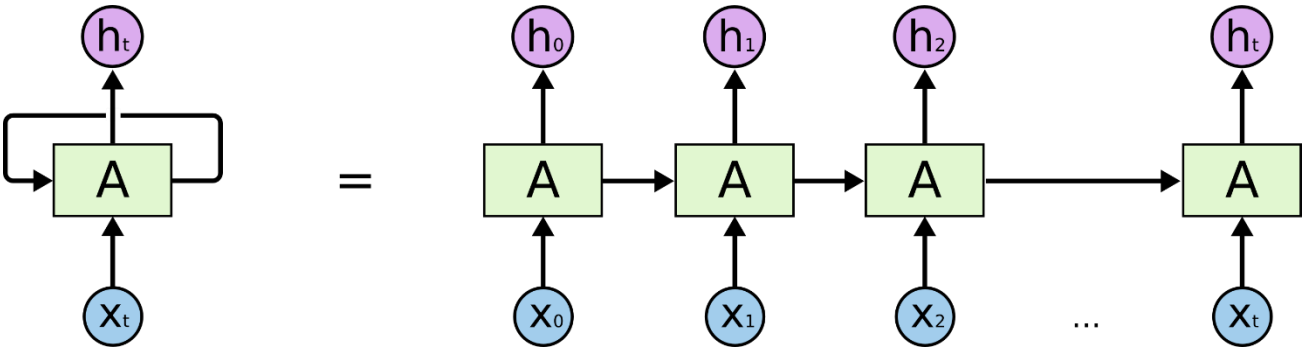
Recurrent Neural Networks

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



Recurrent Neural Networks have loops.

In the above diagram, a chunk of neural network, AA, looks at some input xtxt and outputs a value htht. A loop allows information to be passed from one step of the network to the next.
These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:



An unrolled recurrent neural network.

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data.

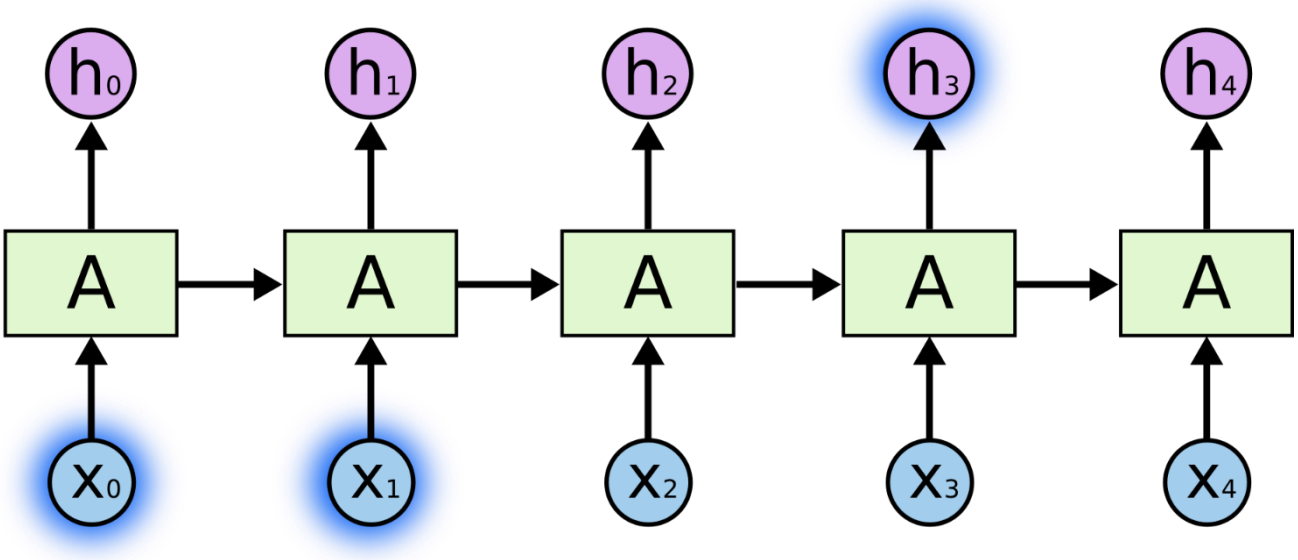
And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on. I'll leave discussion of the amazing feats one can achieve with RNNs to Andrej Karpathy's excellent blog post, The Unreasonable Effectiveness of Recurrent Neural Networks. But they really are pretty amazing.

Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with them. It's these LSTMs that this essay will explore.

The Problem of Long-Term Dependencies

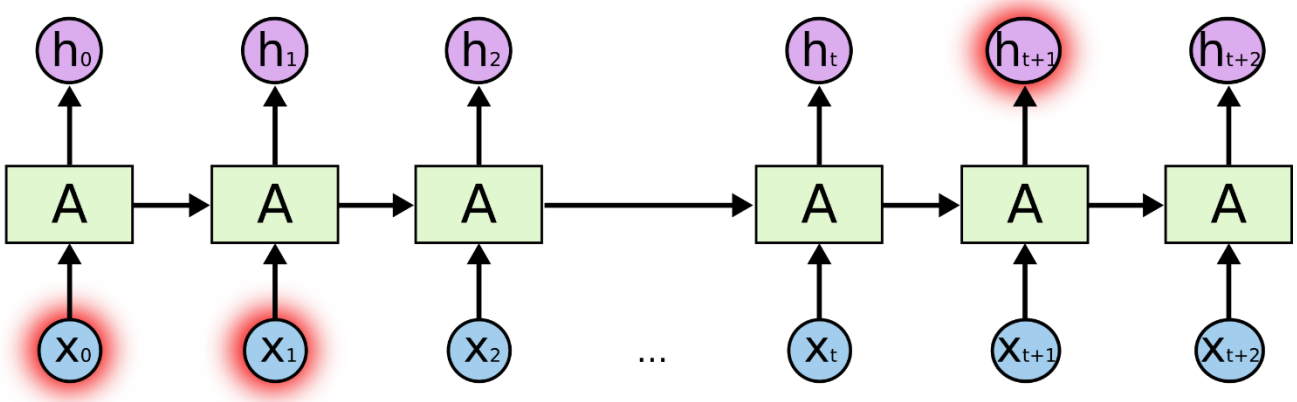
One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.



But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent *French*." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.



In theory, RNNs are absolutely capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them. The problem was explored in depth by Hochreiter (1991) [German] and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

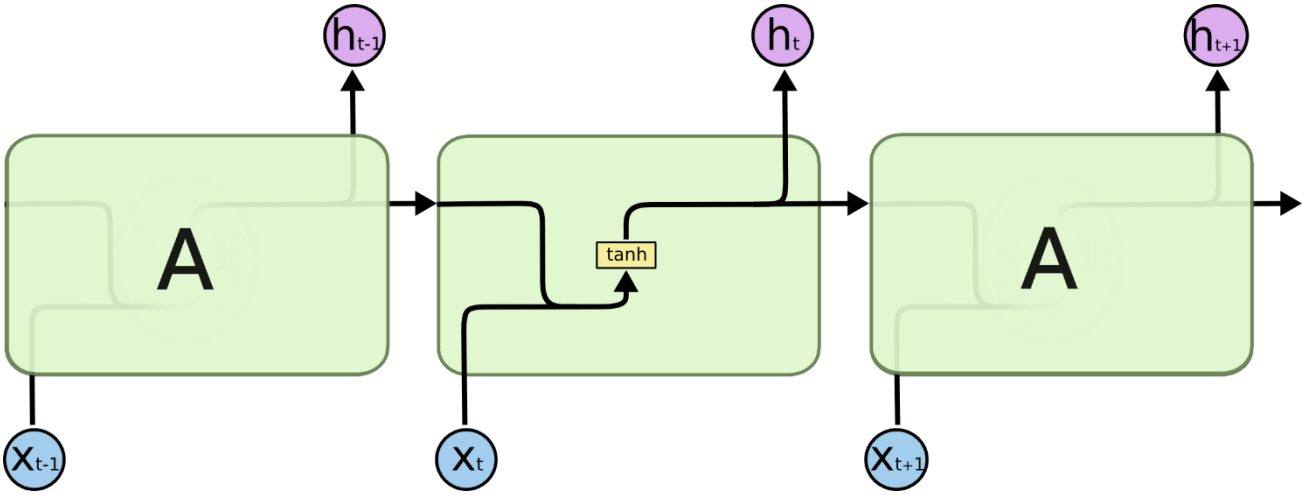
Thankfully, LSTMs don't have this problem!

LSTM Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

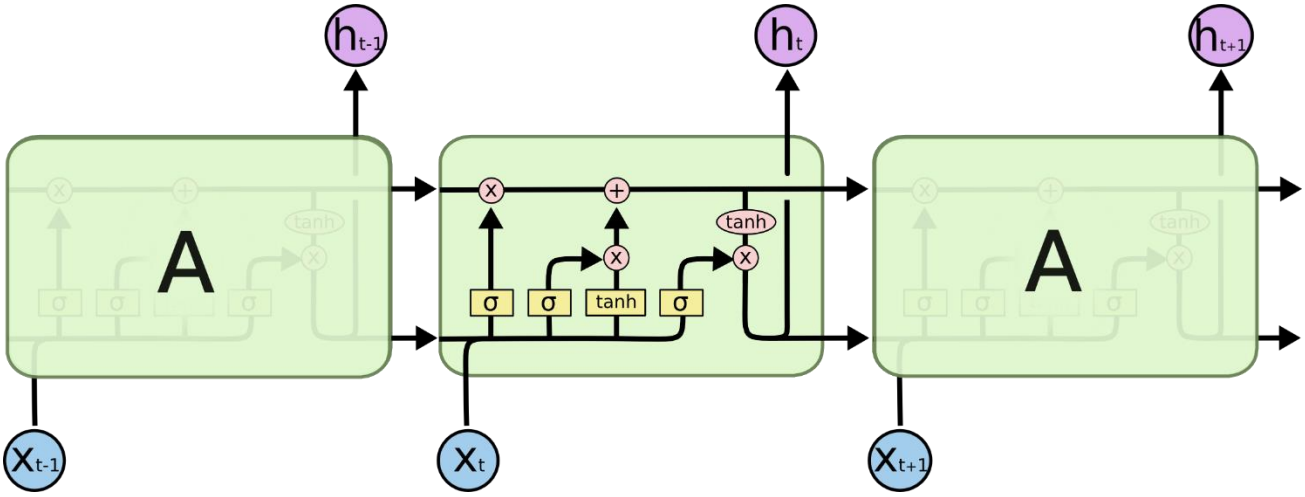
LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



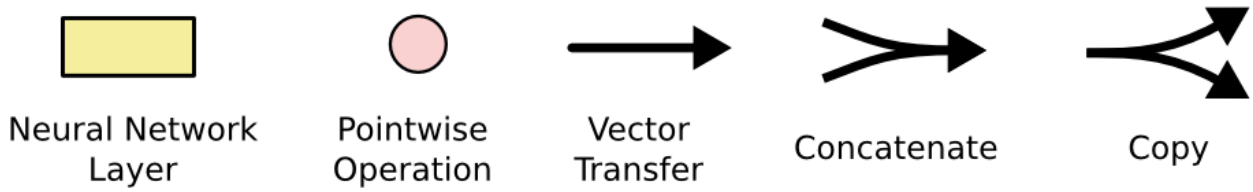
The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in an LSTM contains four interacting layers.

Don’t worry about the details of what’s going on. We’ll walk through the LSTM diagram step by step later. For now, let’s just try to get comfortable with the notation we’ll be using.

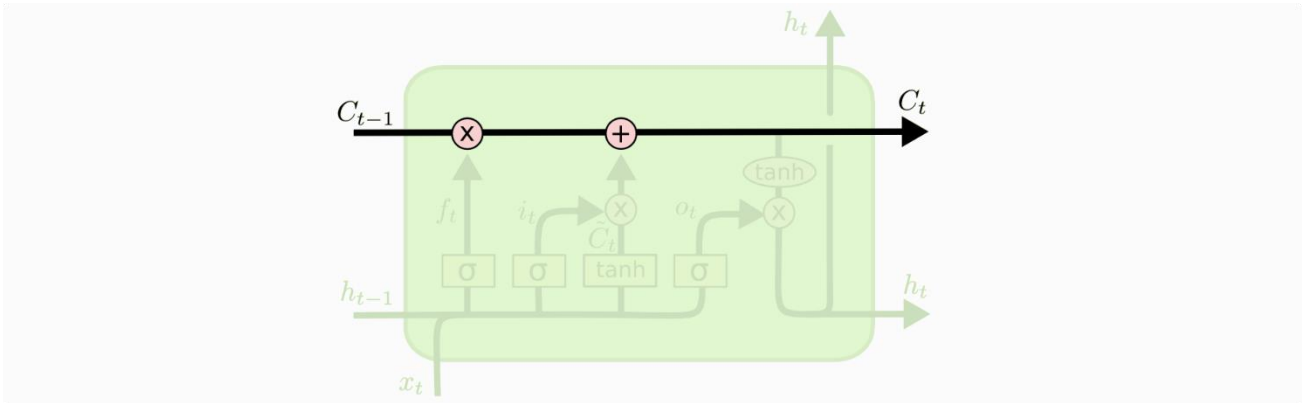


In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

The Core Idea Behind LSTMs

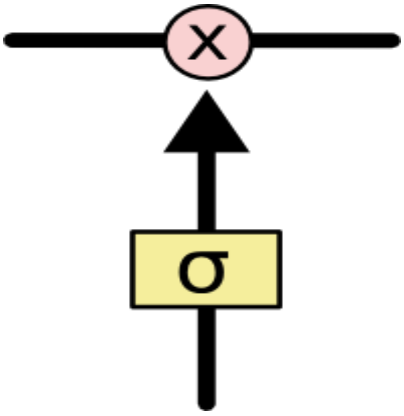
The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.



The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.

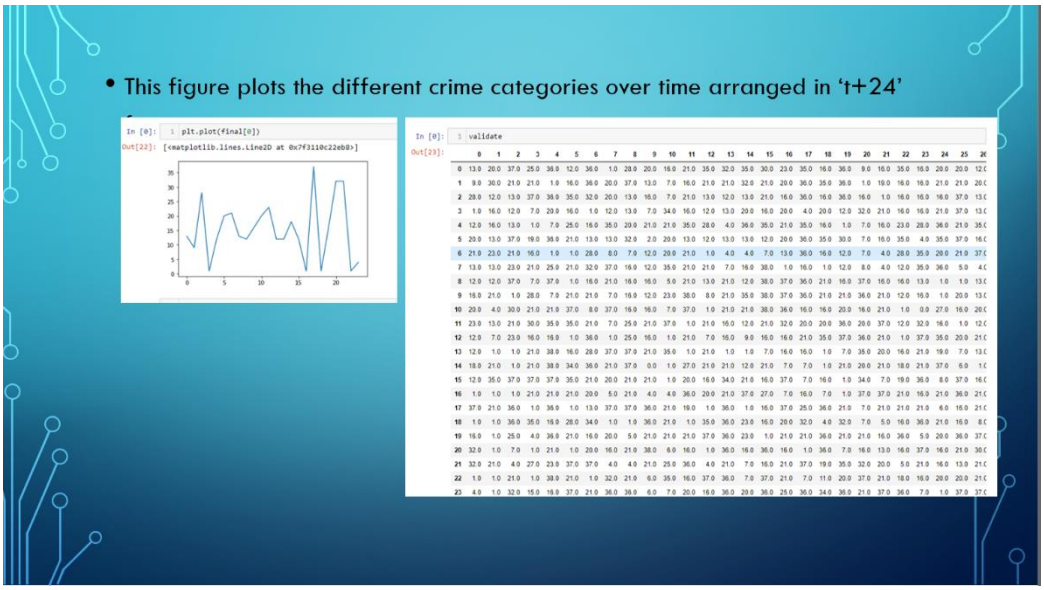


Fig 5.4.1 This figure represents the different crime categories over time arranged in ‘t+24’ format.

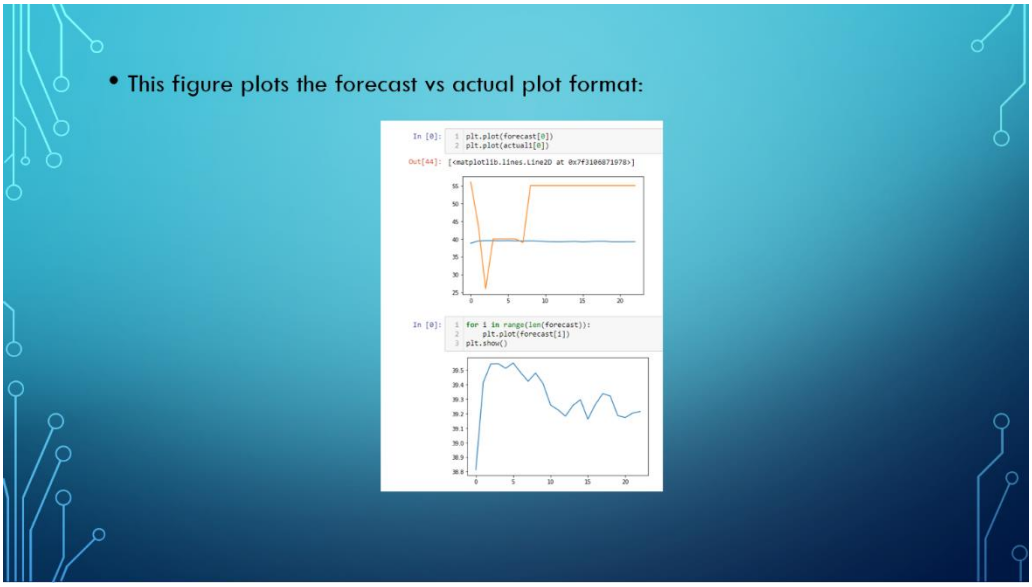


Fig 5.4.2 This figure plots the forecast vs actual plot format.

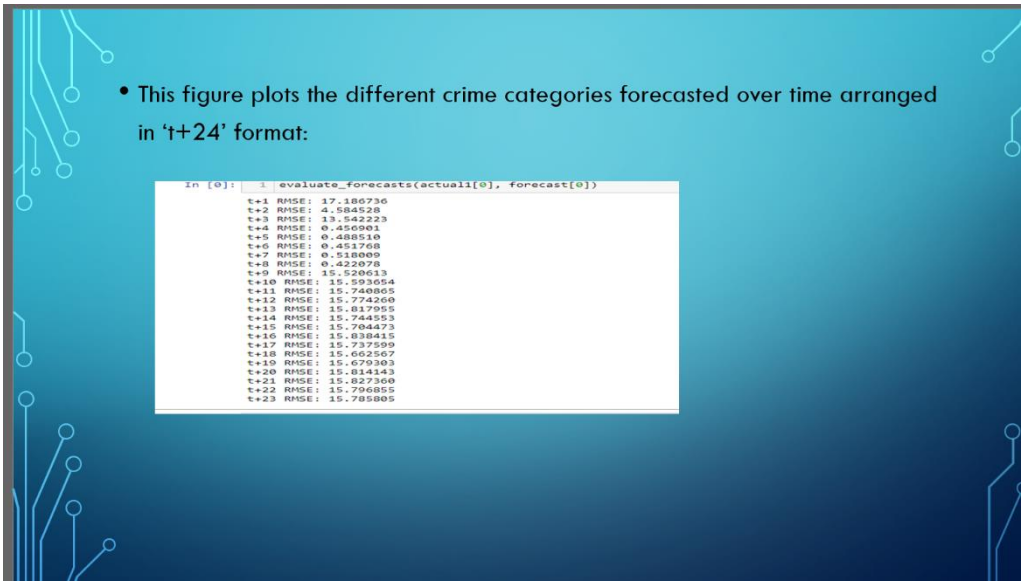


Fig 5.4.3 Forecasted crime categories with time arranged in 't+24' format

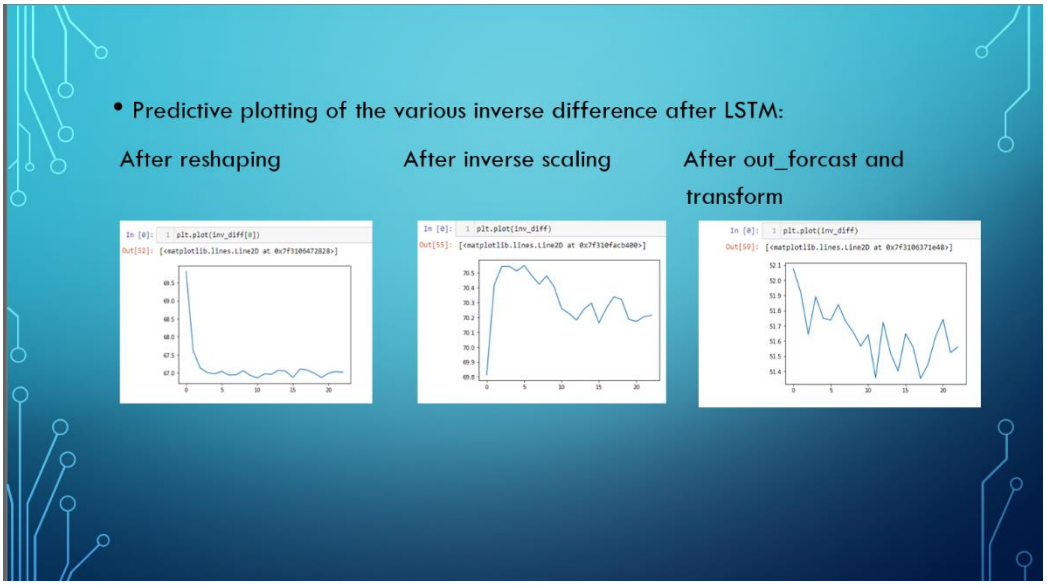


Fig 5.4.4 Predictive plotting after LSTM

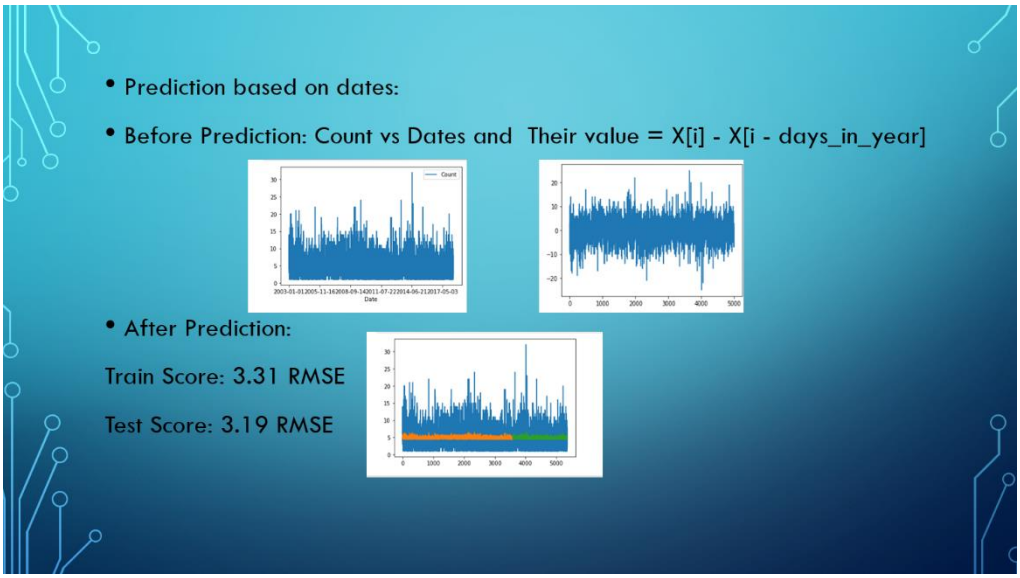


Fig 5.4.5 All these are prediction with RMSE Values.

5.5 Prophet: Automatic Forecasting

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.
Prophet is open source software released by Facebook's Core Data Science team. It is available for download on CRAN and PyPI.

Accurate and fast.

Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. We've found it to perform better than any other approach in the majority of cases. We fit models in Stan so that we can get forecasts in just a few seconds.

Fully automatic.

Get a reasonable forecast on messy data with no manual effort. Prophet is robust to outliers, missing data, and dramatic changes in your time series.

Tunable forecasts.

The Prophet procedure includes many possibilities for users to tweak and adjust forecasts. You can use human-interpretable parameters to improve your forecast by adding your domain knowledge.

Available in R or Python.

We've implemented the Prophet procedure in R and Python, but they share the same underlying Stan code for fitting. Use whatever language you're comfortable with to get forecasts.

Done some necessary tweaks for its implementation since it only takes in values in “ds” and “y”:

```
import numpy as np
dfN['Date'] = pd.to_datetime(dfN['Date'])
print(np.min(dfN['Date'] ),np.max(dfN['Date'] ))
dfN = dfN.sort_values('Date', ascending=True)
dfN = dfN.dropna(how='any')

dfN.head()
```

```
df=pd.DataFrame()
df = dfN[['Date']].copy()
dfN["Count"] = ""
df['Count'] = df.groupby('Date')['Date'].transform('count')
df.drop_duplicates(subset=None, keep='first', inplace=True)
df.columns = ['ds', 'y']
```

We fit the model by instantiating a new `Prophet` object. Any settings to the forecasting procedure are passed into the constructor. Then you call its `fit` method and pass in the historical dataframe.

```
m = Prophet()
m.fit(df)
```

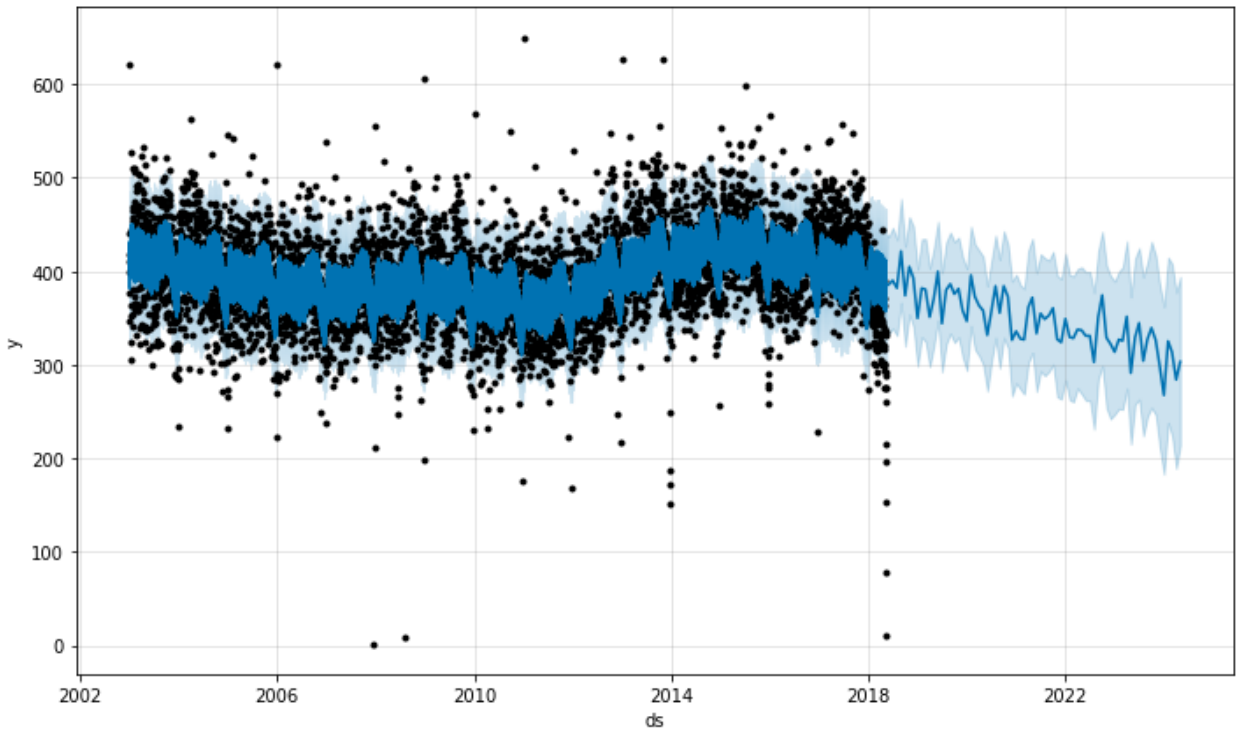
Predictions are then made on a dataframe with a column `ds` containing the dates for which a prediction is to be made. You can get a suitable dataframe that extends into the future a specified number of days using the helper method `Prophet.make_future_dataframe`.

```
future = m.make_future_dataframe(periods=12 * 6, freq='M')
future.tail()
```

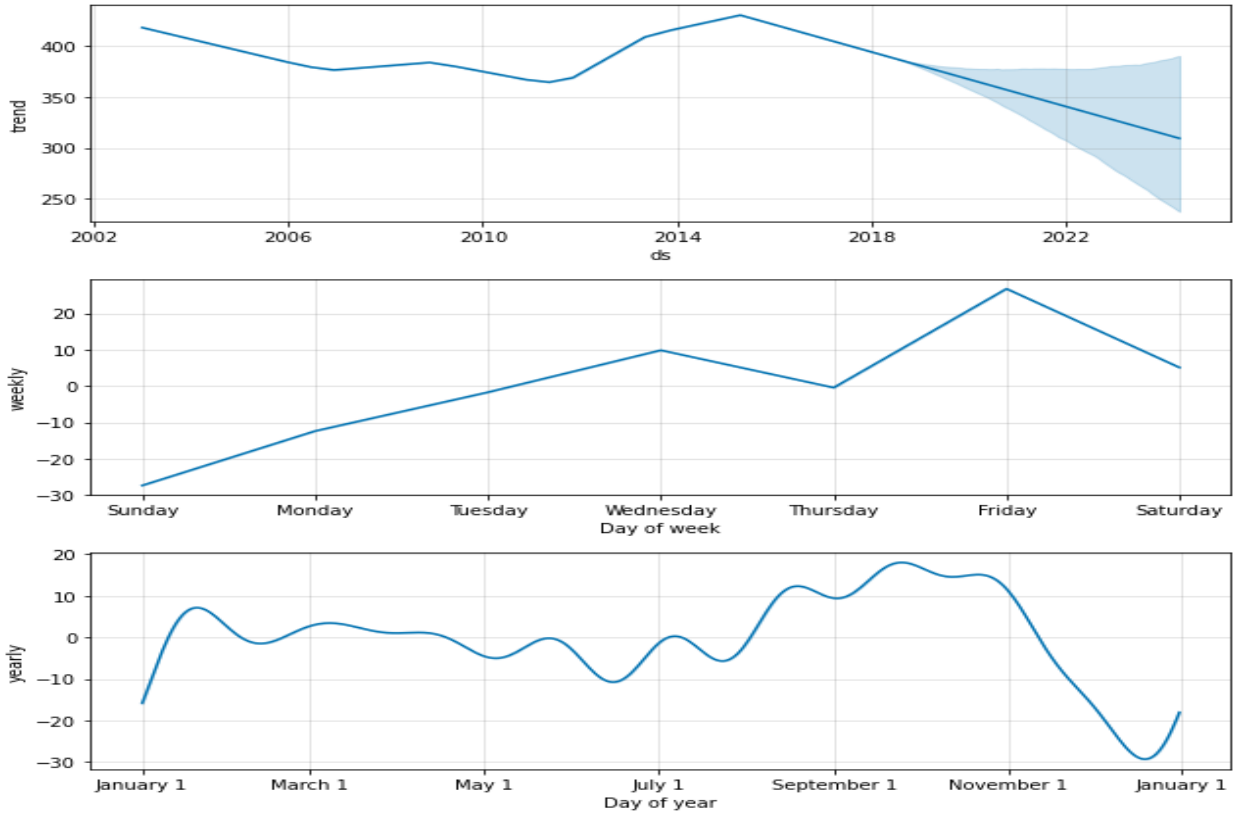
Then we start the prediction parts:

Initial predictions plot:

```
fig1 = m.plot(forecast)
```



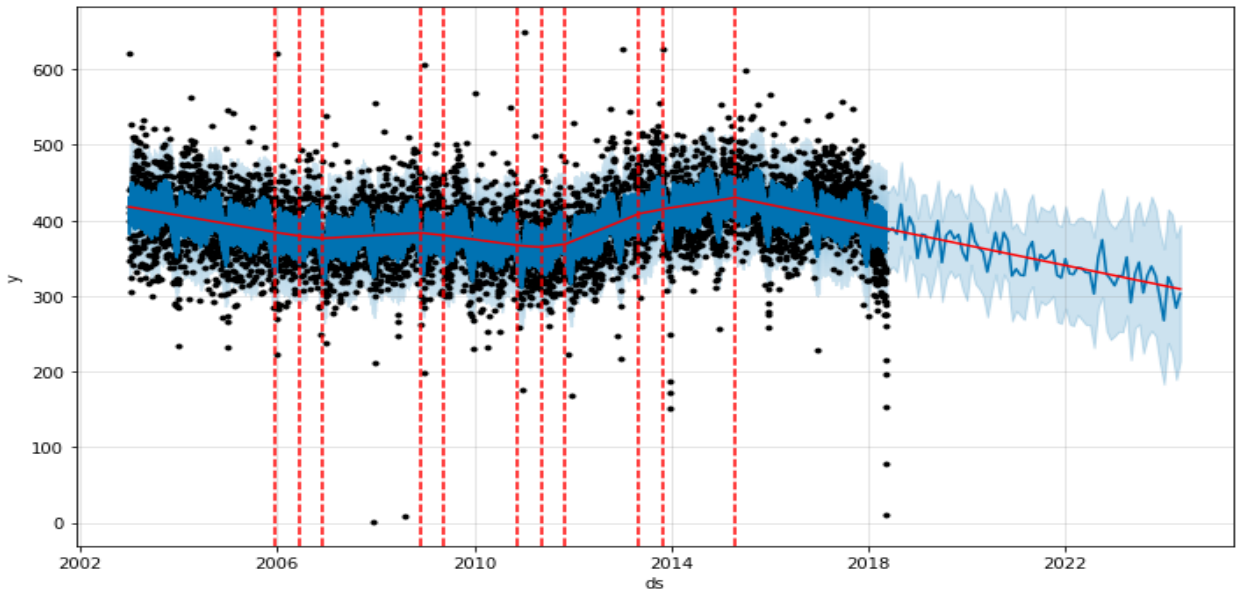

```
fig2 = m.plot_components(forecast)
```



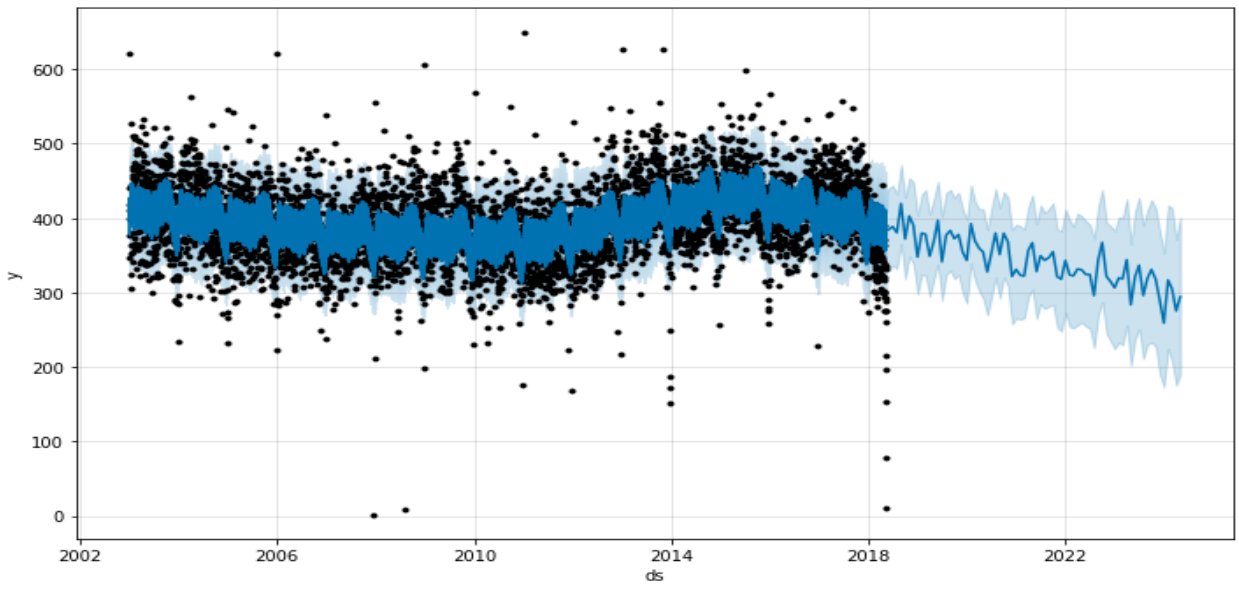
This just shows the trends for the dataset.

Here what we call change points occur that helps the forecasting for the next couple years based on the dataset. These change points are crucial to what is essentially the turning point of trends.

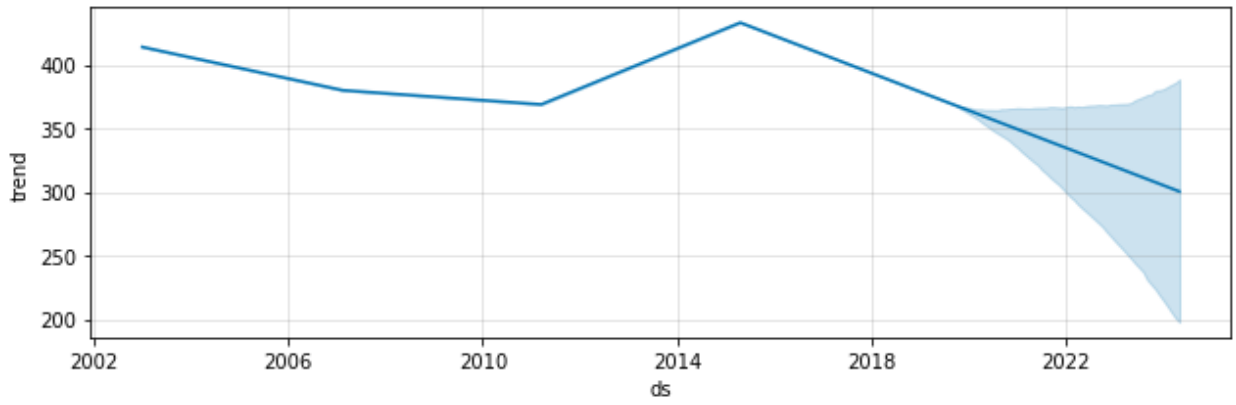
```
fig = m.plot(forecast)
a = add_changepoints_to_plot(fig.gca(), m, forecast)
```

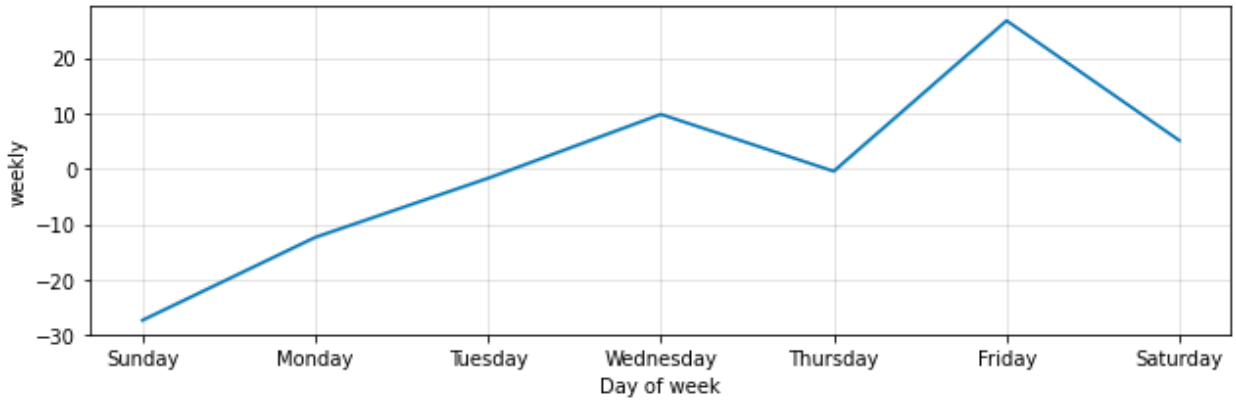


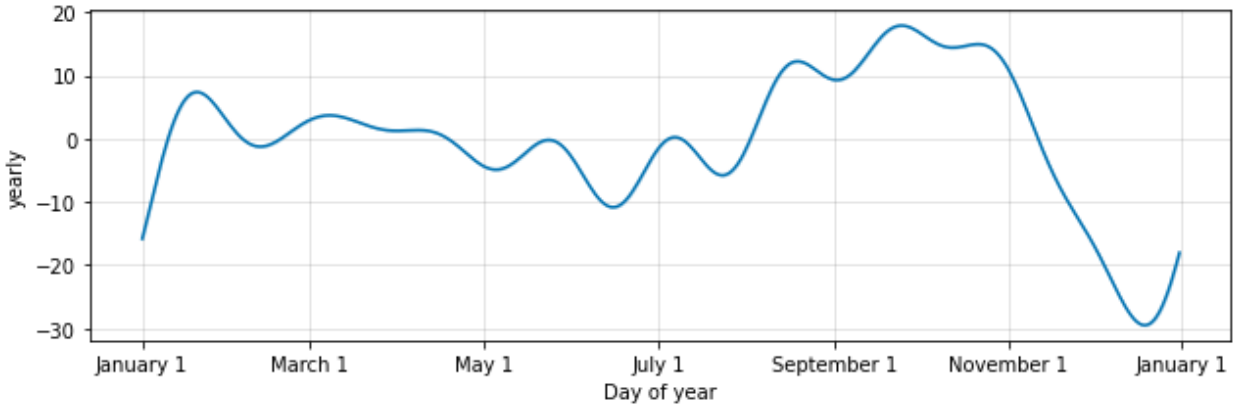

```
m = Prophet(n_changepoints=3).fit(df)
future = m.make_future_dataframe(periods=12 * 6, freq='M')
forecast = m.predict(future)
fig = m.plot(m.predict(future))
```




```
fig2 = m.plot_components(forecast)
```







After this a cross validation is performed, that take in days and according to them periods are made over the horizon.

```
df_cv = cross_validation(m, initial='730 days', period='180 days', horizon =
'365 days')
df_cv.head()
```

Then comes another crucial point in seasonality.

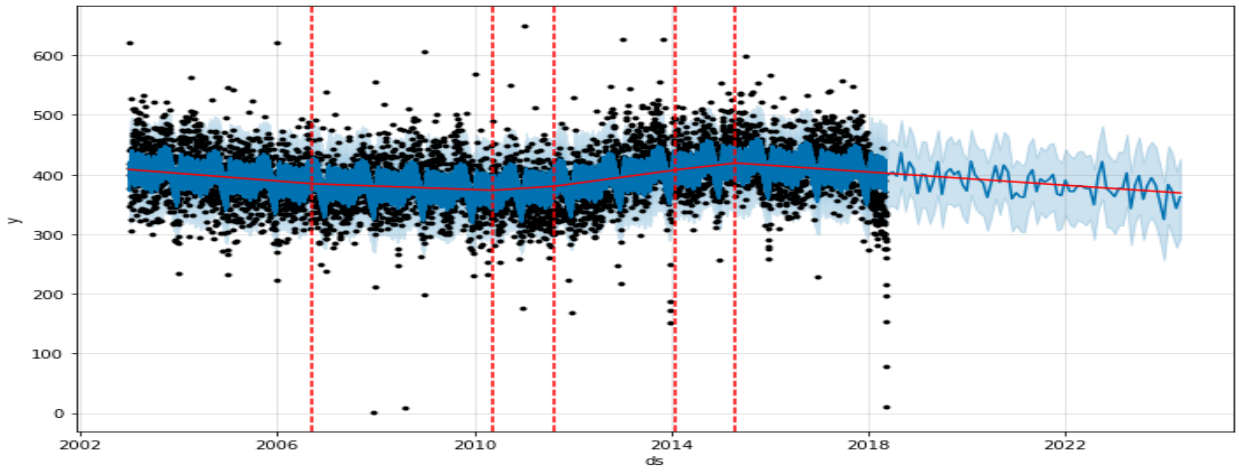
```
changepoint_prior_scales = [.005, .05, .5, 2]
n_changepoints = [8, 10, 15, 20, 25]

rmse = []
for changepoint_prior_scale in changepoint_prior_scales:
    for n_changepoint in n_changepoints:
```

```
        print('Changepoint Prior Scale:', changepoint_prior_scale)
        print('Number Changepoints:', n_changepoint)
        m = Prophet(changepoint_prior_scale=changepoint_prior_scale, n_change
points=n_changepoint).fit(df)
        future = m.make_future_dataframe(periods=12 * 6, freq='M')
        forecast = m.predict(future)
        fig = m.plot(forecast)
        a = add_changepoints_to_plot(fig.gca(), m, forecast)
        plt.show()
        df_cv = cross_validation(m, initial='1095 days', period='180 days', h
orizon = '365 days')
        df_p = performance_metrics(df_cv)
        rmse.append((df_p['rmse'].mean(), {'changepoint_prior_scale': changep
oint_prior_scale, 'n_changepoint': n_changepoint}))
        fig = plot_cross_validation_metric(df_cv, metric='mape')
        plt.show()
```

After this a series of predictive outputs were given that helps change and better shape the output for the prediction with each change points. If any problems occur, the algorithms shift directly to Newtons rule and that makes it that much robust in implementation and use.

There were total 25 change points that were encountered, and the prediction changed accordingly, a simple comparison between the 1st and the 25th will be very clear.

1st one:



The final 25th

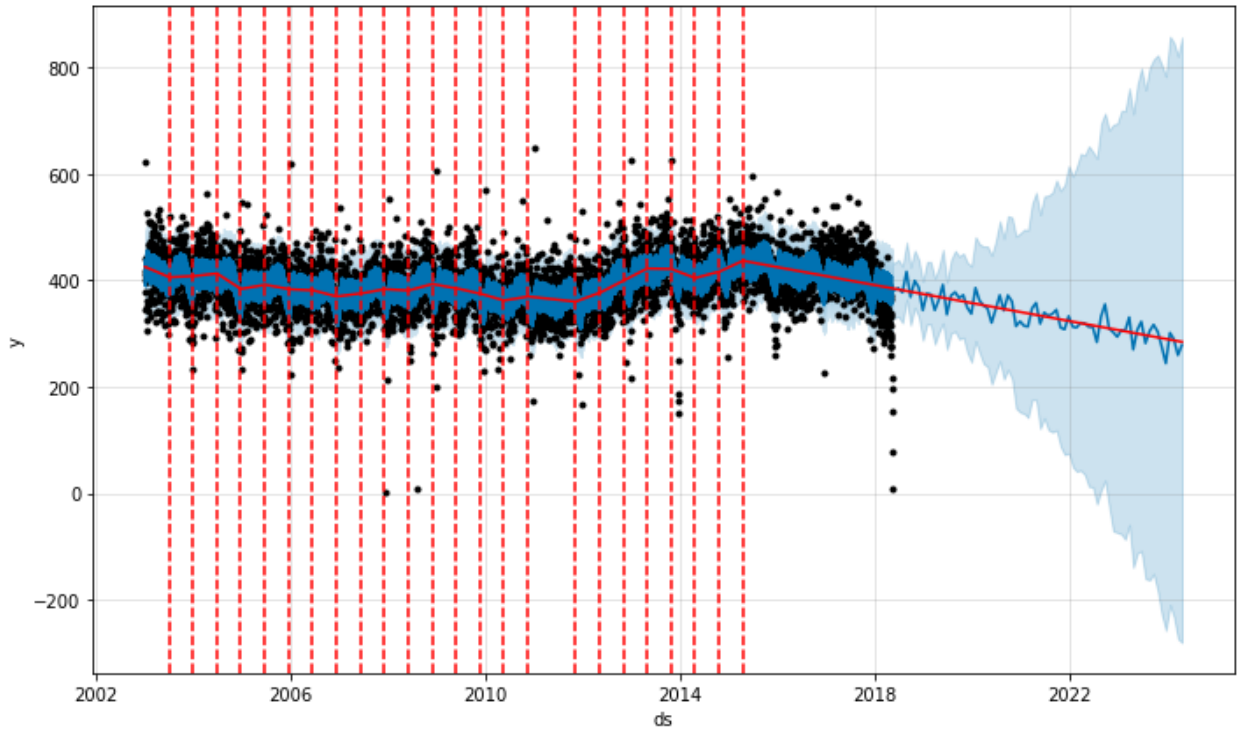


Fig 5.5.1 Prediction from prophet after every change points

As we can see, the output changed drastically with more accuracy.

```
[(46.49255482933897, {'changepoint_prior_scale': 0.005, 'n_changepoint': 8}),
 (46.4180779955481, {'changepoint_prior_scale': 0.005, 'n_changepoint': 10}),
 (46.46242303698752, {'changepoint_prior_scale': 0.005, 'n_changepoint': 15}),
 (46.473582349036235, {'changepoint_prior_scale': 0.005, 'n_changepoint': 20}),
 (46.538844014135464, {'changepoint_prior_scale': 0.005, 'n_changepoint': 25}),
 (45.88091202369968, {'changepoint_prior_scale': 0.05, 'n_changepoint': 8}),
 (45.9891669719476, {'changepoint_prior_scale': 0.05, 'n_changepoint': 10}),
 (45.87986193812367, {'changepoint_prior_scale': 0.05, 'n_changepoint': 15}),
 (45.94872354661919, {'changepoint_prior_scale': 0.05, 'n_changepoint': 20}),
 (45.910395581787945, {'changepoint_prior_scale': 0.05, 'n_changepoint': 25}),
 (45.572863861265155, {'changepoint_prior_scale': 0.5, 'n_changepoint': 8}),
 (45.945104401828594, {'changepoint_prior_scale': 0.5, 'n_changepoint': 10}),
 (45.70404174689712, {'changepoint_prior_scale': 0.5, 'n_changepoint': 15}),
 (45.7882771477 4089, {'changepoint_prior_scale': 0.5, 'n_changepoint': 20}),
 (45.74255233335355, {'changepoint_prior_scale': 0.5, 'n_changepoint': 25}),
 (45.75997242564671, {'changepoint_prior_scale': 2, 'n_changepoint': 8}),
 (46.879860936572726, {'changepoint_prior_scale': 2, 'n_changepoint': 10}),
 (47.18447920001171, {'changepoint_prior_scale': 2, 'n_changepoint': 15}),
 (46.606201814129776, {'changepoint_prior_scale': 2, 'n_changepoint': 20}),
 (47.13075613837792, {'changepoint_prior_scale': 2, 'n_changepoint': 25})]
```

Here is the most important one of them all and all the more reason to use this algorithm:

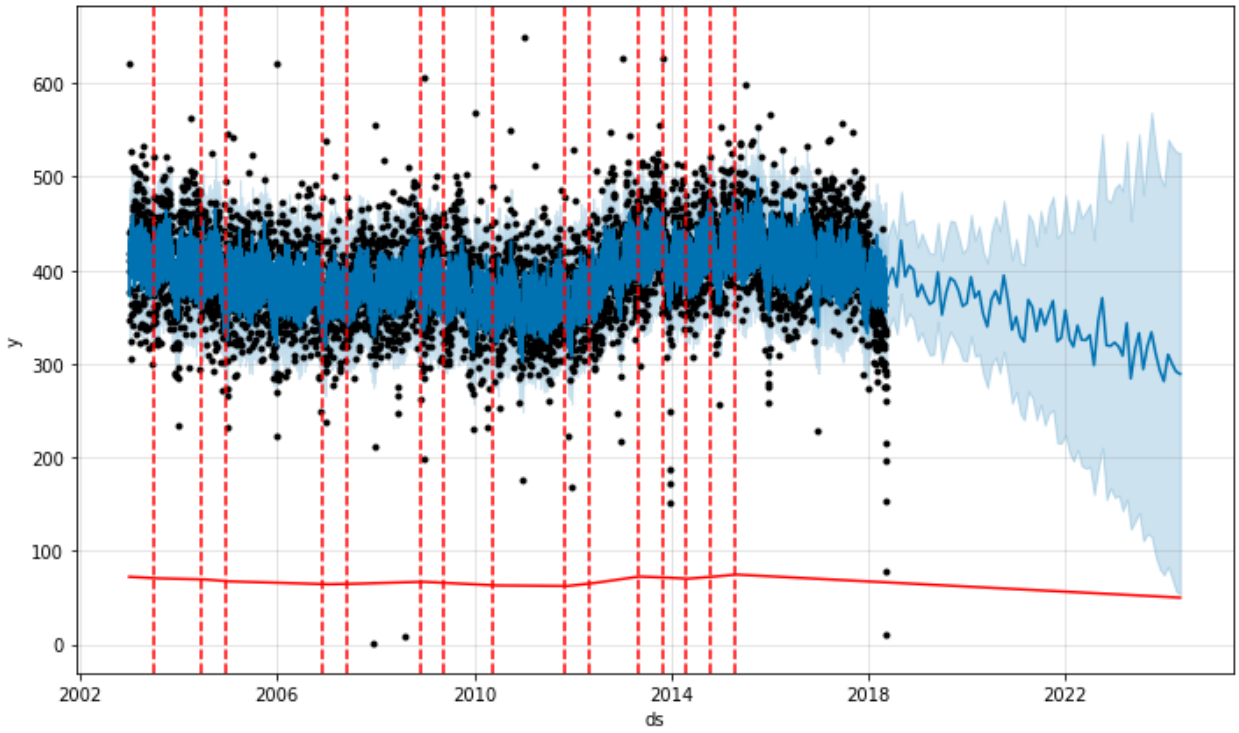
This algorithm takes into consideration of facts that may play crucial roles in seasonality; including holidays etc.

```
#dataframe of annual US Public Holidays over training and forecasting periods


ny = pd.DataFrame({'holiday': "New Year's Day", 'ds' : pd.to_datetime(['2016-01-01', '2017-01-01'])})
mlk = pd.DataFrame({'holiday': 'Birthday of Martin Luther King, Jr.', 'ds' : pd.to_datetime(['2016-01-18', '2017-01-16'])})
wash = pd.DataFrame({'holiday': "Washington's Birthday", 'ds' : pd.to_datetime(['2016-02-15', '2017-02-20'])})
mem = pd.DataFrame({'holiday': 'Memorial Day', 'ds' : pd.to_datetime(['2016-05-30', '2017-05-29'])})
ind = pd.DataFrame({'holiday': 'Independence Day', 'ds' : pd.to_datetime(['2015-07-04', '2016-07-04', '2017-07-04'])})
lab = pd.DataFrame({'holiday': 'Labor Day', 'ds' : pd.to_datetime(['2015-09-07', '2016-09-05', '2017-09-04'])})
col = pd.DataFrame({'holiday': 'Columbus Day', 'ds' : pd.to_datetime(['2015-10-12', '2016-10-10', '2017-10-09'])})
vet = pd.DataFrame({'holiday': "Veteran's Day", 'ds' : pd.to_datetime(['2015-11-11', '2016-11-11', '2017-11-11'])})
thanks = pd.DataFrame({'holiday': 'Thanksgiving Day', 'ds' : pd.to_datetime(['2015-11-26', '2016-11-24'])})
christ = pd.DataFrame({'holiday': 'Christmas', 'ds' : pd.to_datetime(['2015-12-25', '2016-12-25'])})

holidays = pd.concat([ny, mlk, wash, mem, ind, lab, col, vet, thanks, christ])
```

```
prophet = Prophet(growth='linear',
                  yearly_seasonality=True,
                  weekly_seasonality=True,
                  daily_seasonality=True,
#                   holidays=holidays,
                  seasonality_mode='multiplicative',
                  seasonality_prior_scale=10,
                  holidays_prior_scale=10,
                  changepoint_prior_scale=.05,
                  mcmc_samples=0
                  ).add_seasonality(name='quarterly',
                                    period=365.25 / 4,fourier_order=15
                  )
prophet.fit(df)
future = prophet.make_future_dataframe(periods=12 * 6, freq='M')
forecast = prophet.predict(future)
```

```
fig = prophet.plot(forecast)
a = add_changepoints_to_plot(fig.gca(), prophet, forecast)
plt.show()
df_cv = cross_validation(prophet, initial='1095 days', period='180 days', hor
izon = '365 days')
df_p = performance_metrics(df_cv)
fig = plot_cross_validation_metric(df_cv, metric='mape')
plt.show()
```

The final output for this entire part:



Chapter 6:

Plotting provided location before prediction as heat maps:

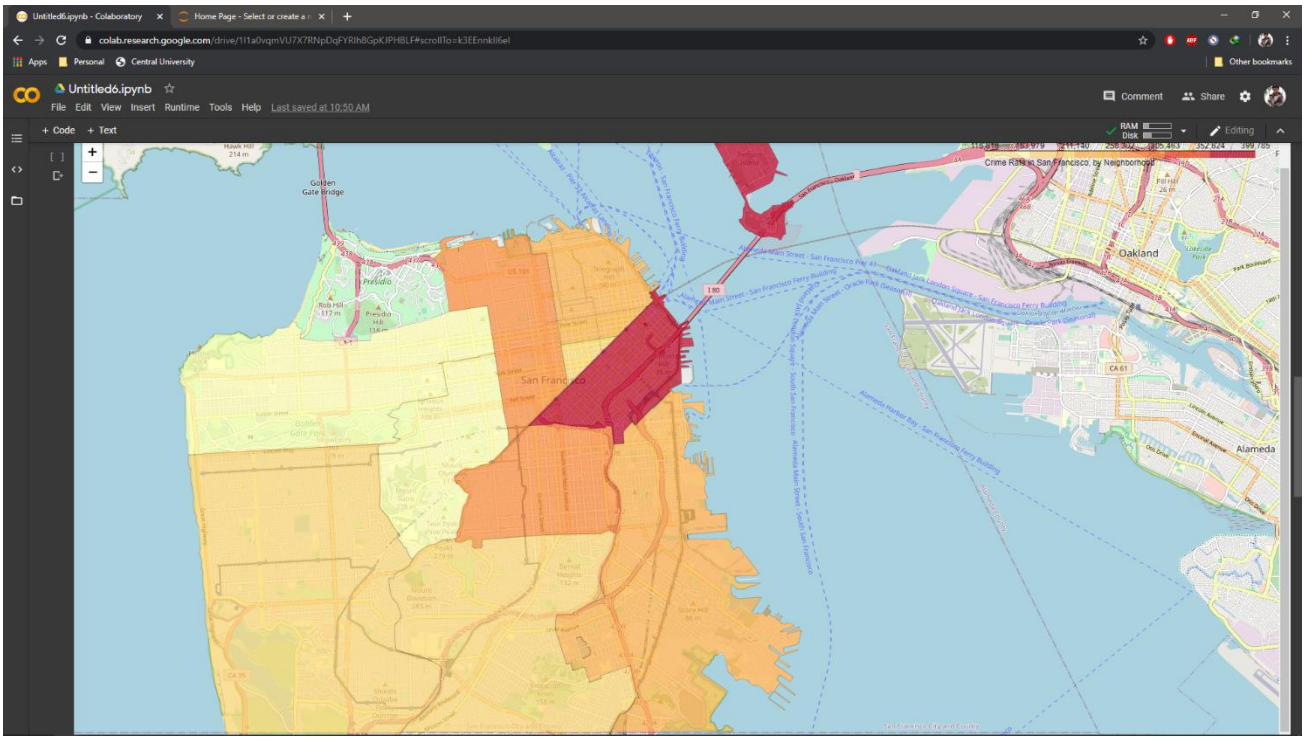


Fig 6.1 This provides the entire heatmap based on total number of crimes vs its count.

I tried using prophet by changing its parameter for location prediction by simultaneously solving the latitude and longitude for the desired prediction.

- This prediction is for only 1 month since the dataset ended.

Period:

5/15/2018 10:31:00 AM

to

6/14/2018 10:30:00 AM

- There are also various spots in the map but since this one was clustered too much, it was shown.

Then I simply plot them on the map using folium and its heatmap feature.

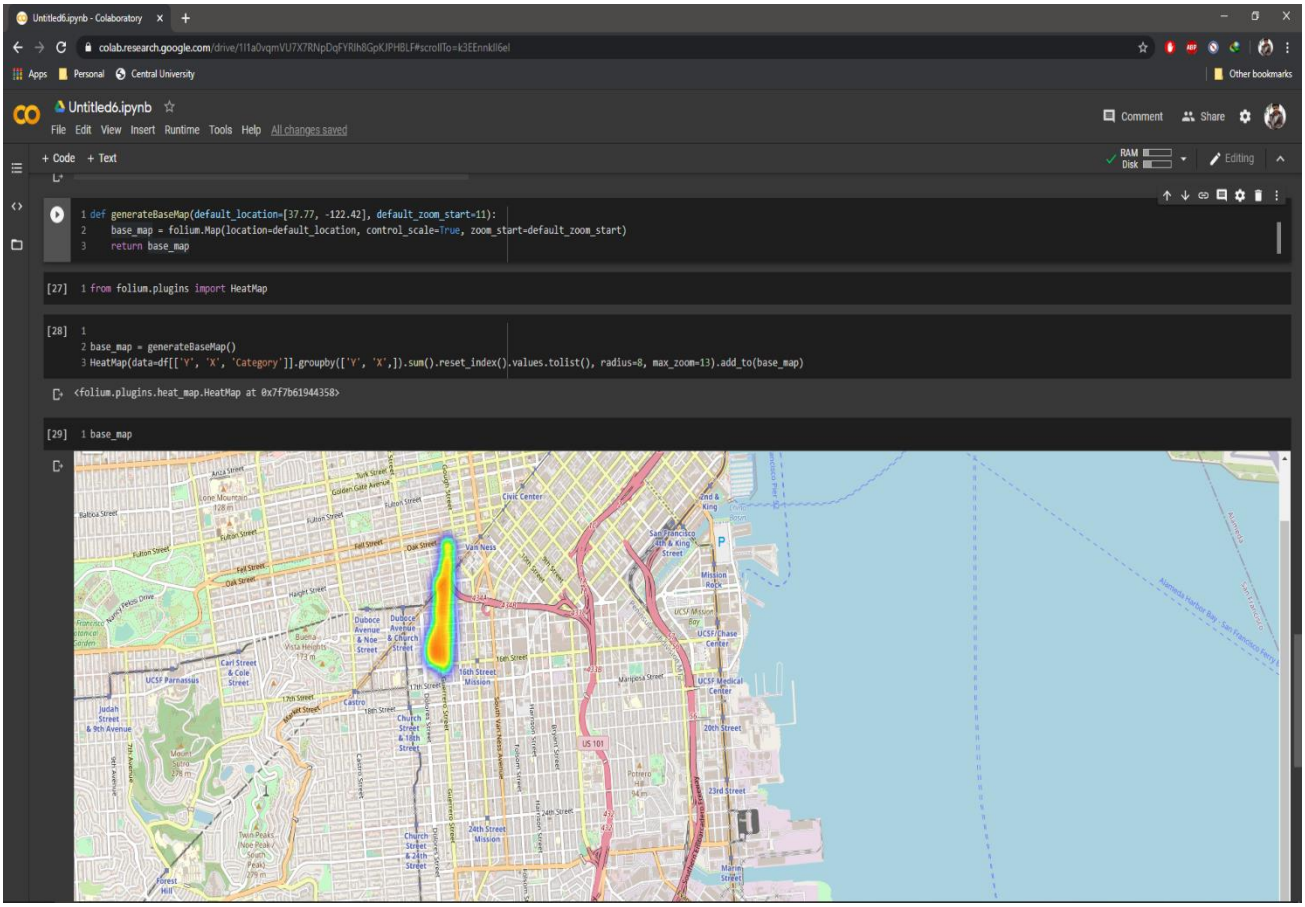


Fig 6.2 The immediate next month plotting example

Chapter-7
Conclusion and future scope

7.1 Conclusion

The initial problem of classifying different crime categories was a challenging multi-class classification problem, and there was not enough predictability in our initial data-set to obtain very high accuracy on it. We found that a more meaningful approach was to collapse the crime categories into fewer, larger groups, in order to find structure in the data. We got high accuracy and precision on Prediction. However, the Violent/Non-violent crime classification did not yield remarkable results with the same classifiers – this was a significantly harder classification problem. Thus, collapsing crime categories is not an obvious task and requires careful choice and consideration.

Possible avenues through which to extend this work include time-series modeling of the data to understand temporal correlations in it, which can then be used to predict surges in different categories of crime. It would also be interesting to explore relationships between surges in different categories of crimes – for example, it could be the case that two or more classes of crimes surge and sink together, which would be an interesting relationship to uncover. Other areas to work on include implementing a more accurate multi-class classifier, and exploring better ways to visualize our results.

7.2 Future Scope

The goal of any society shouldn't be to just catch criminals but to prevent crimes from happening in the first place

- **Predicting Future Crime Spots with greater accuracy:** By using historical data and observing where recent crimes took place we can predict where future crimes will likely happen. For example, a rash of burglaries in one area could correlate with more burglaries in surrounding areas in the near future. System highlights possible hotspots on a map the police should consider patrolling more heavily

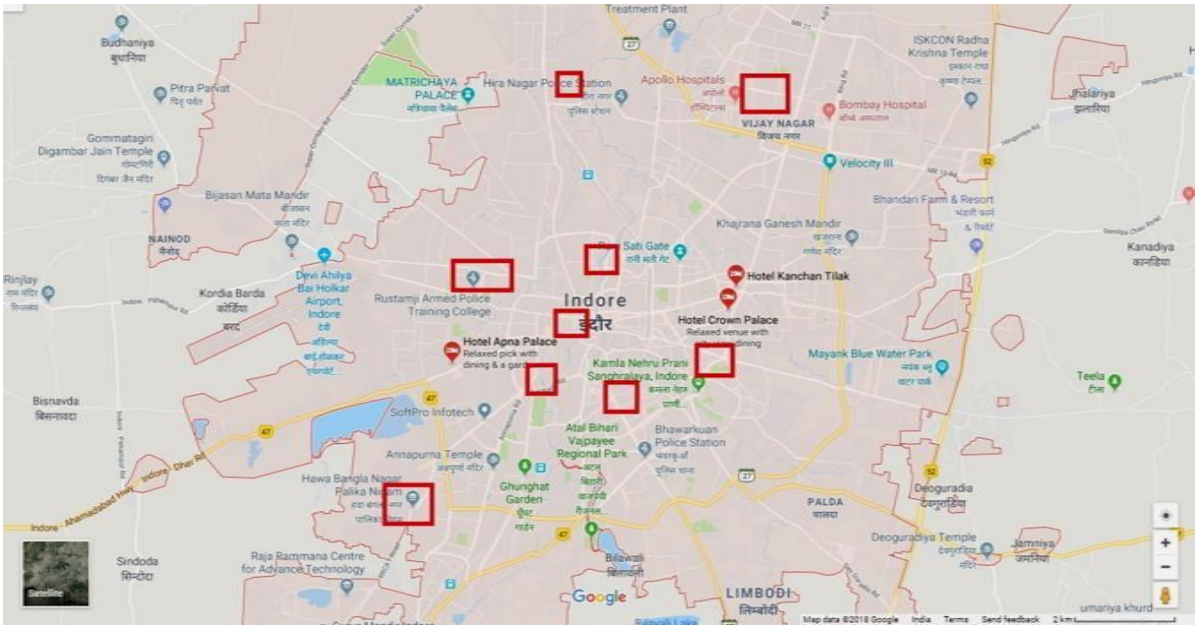


Fig 6.1 Predicting Surges with much greater accuracy and understanding.

- **Predicting Who Will Commit a Crime**: Using Face Recognition to predict if a individual will commit a crime before it happens. The system will detect if there are any suspicious changes in their behavior or unusual movements. For example, if an individual seems to be walking back and forth in a certain area over and over indicating they might be a pickpocket or casing the area for a future crime. It will also track individual over time.

- **Pretrial Release and Parole:** After being charged with a crime, most individuals are released until they actually stand trial. In the past deciding who should be released pretrial

or what an individual's bail should be set at is mainly now done by judges using their best judgment. In just a few minutes, judges had to attempt to determine if someone is a flight risk, a serious danger to society, or at risk to harm a witness if released. It is an imperfect system open to bias. The media organization's analysis indicated the system might indirectly contain a strong racial bias. They found, "That black defendants who did not recidivate over a two-year period were nearly twice as likely to be misclassified as higher risk compared to their white counterparts (45 percent vs. 23 percent)." The report raises the question of whether better AI/ML can eventually produce more accurate predictions or if it would reinforce existing problems. Any system will be based off of real-world data, but if the real-world data is generated by biased police officers, it can make the AI/ML biased.

7.3 Expected Outcome

The idea behind this project is that crimes are relatively predictable; it just requires being able to sort through a massive volume of data to find patterns that are useful to law enforcement. This kind of data analysis was technologically impossible a few decades ago, but the hope is that recent developments in machine learning are up to the task.

The use of AI and machine learning to detect crime via sound or cameras currently exists, is proven to work, and expected to continue to expand. The use of AI/ML in predicting crimes or an individual's likelihood for committing a crime has promise but is still more of an unknown. The biggest challenge will probably be "proving" to politicians that it works. When a system is designed to stop something from happening, it is difficult to prove the negative.

Companies that are directly involved in providing governments with AI tools to monitor areas or predict crime will likely benefit from a positive feedback loop. Improvements in crime prevention technology will likely spur increased total spending on this technology.

Possible avenues through which to extend this work include time-series modeling of the data to understand temporal correlations in it, which can then be used to predict surges in different categories of crime. It would also be interesting to explore relationships between surges in different categories of crimes – for example, it could be the case that two or more classes of crimes surge and sink together, which would be an interesting relationship to uncover. Other

areas to work on include implementing a more accurate multi-class classifier, and exploring better ways to visualize our results.

Thank You.