# ROAD ACCIDENT PREDICTION WITH SEVERITY LEVEL EXPLORATION

***A Major Project Report***

*submitted in partial fulfillment of the*
*requirements of VIII-Semester for the degree*
*of*
*Bachelor of Technology*
*in*
**COMPUTER SCIENCE & ENGINEERING**

***by***

**K. Raj Kamal (Roll No. 16115030)**

**G. Jayanth Kumar (Roll No. 16115022)**

**K. Gautham Kumar (Roll No. 16115029)**

Under the guidance of

**Dr. Preeti Chandrakar**

Assistant Professor

NIT-RAIPUR


**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**NATIONAL INSTITUTE OF TECHNOLOGY**
**RAIPUR, CG (INDIA)**

MAY 2020

# DECLARATION

We hereby declare that the work described in this thesis, entitled **"ROAD ACCIDENT PREDICTION WITH SEVERITY LEVEL EXPLORATION"** which is being submitted by us in partial fulfilment for the VIII-Semester of the degree of Bachelor of Technology in the Department of **Computer Science and Engineering** to the National Institute of Technology Raipur is the result of investigations carried out by us under the guidance of **Dr. Preeti Chandrakar**.

The work is original and has not been submitted for any Degree/Diploma of this or any other Institute/university.

Signature

Name of the Candidate

K. Raj Kamal
Roll No: 16115030

G. Jayanth Kumar
Roll No:16115022

K. Gautham
Roll No: 16115029

Place: Raipur,

Date: 07-05-2020.

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**NATIONAL INSTITUTE OF TECHNOLOGY**

## *CERTIFICATE*

This is to certify that the project entitled **"ROAD ACCIDENT PREDICTION WITH SEVERITY LEVEL EXPLORATION",** that is being submitted by **K. Raj Kamal (Roll No. 16115030), G. Jayanth (Roll No. 16115022), K. Gautham (Roll No. 16115029)** in partial fulfilment for VIII-Semester of the degree of **Bachelor of Technology in Computer Science & Engineering** to National Institute of Technology Raipur is a record of bonafide work carried out by them under my guidance and supervision.

The matter presented in this project document has not been submitted by them for the award of any other degree elsewhere.

*Signature of Supervisor*
**Dr. Preeti Chandrakar**
**Assistant Professor**
**Department of Computer Science & Engineering**
**National Institute of Technology, Raipur (CG.)**

*Project Coordinator*
**Dr. K. Jairam Naik**
**Department of Computer Science & Engineering**
**National Institute of Technology, Raipur (CG.)**

*H.O.D*
**Dr. Pradeep Singh**
**Department of C.S.E**
**NIT Raipur (CG.)**

# ACKNOWLEDGEMENT

We would like to acknowledge my college **National Institute of Technology, Raipur** for providing a holistic environment that nurtures creativity and research-based activities.

We express our sincere thanks to **Dr. Preeti Chandrakar, Assistant Professor CSE Department**, **NIT Raipur,** the guide of the project for guiding and correcting throughout the process with attention and care. She has frequently suggested our creative ideas and guided through the major hurdles that occurred during the duration of the project.

We would also thank **Dr. Pradeep Singh, Head of the department** and all the faculty members without whom this project would be a distant reality. We also extend my heartfelt thanks to our family and friends who supported us.

K. Raj Kamal
Roll No.: 16115030

G. Jayanth Kumar
Roll No: 16115022

K. Gautham
Roll No: 16115029

# ABSTRACT

Reducing traffic accidents is an important public safety challenge, therefore, accident analysis and prediction has been a topic of much research over the past few decades. Using small-scale datasets with limited coverage, being dependent on an extensive set of data, and being not applicable for real-time purposes are the important shortcomings of the existing studies.

To address these challenges, we use a new dataset labeled as "US Accidents" published in the paper titled "A Countrywide Traffic Accident Dataset". This dataset with 3 million records can be used as a solution for real-time traffic accident prediction using easy-to-obtain, but sparse data. Using this dataset accompanied with Machine Learning models we want to design better and reliable solutions for this problem.

We started with exploration of the dataset, which exposed that the dataset is completely imbalanced. So as a phase one, we've generated various versions of the dataset, to tackle this problem. Finally we ended with generating synthetic data using SMOTE - NC(Synthetic Minority Oversampling Technique for Nominal and continuous). Finally we were able to generate a final dataset which has 10,000 samples for each class.

Once the dataset is ready, we employ various machine learning models, for the generation of the predictions. Among all, boosting algorithms performed the best, and eXtreme Gradient Boosting (XGBoost) performed the best with 93.4% testing accuracy. Once the model is finalized, we have designed a graphical user interface, through which even a normal user can interact with the trained model. Through this GUI, we can give inputs through an HTML form and get the output prediction including the severity level of the accident, confidence score of the prediction along with a customized comment for the current prediction.

# CONTENT


| DESCRIPTION | PAGE NO. |
|---|---|
| DECLARATION | 2 |
| CERTIFICATE | 3 |
| ACKNOWLEDGEMENT | 4 |
| ABSTRACT | 5 |
| CONTENTS | 6 |
| LIST OF FIGURES | 8 |
| LIST OF TABLES | 9 |
| 1. INTRODUCTION | 10 |
| 1.1 Overview | 11 |
| 1.2 Objectives and Importance of the Project | 11 |
| 1.3 Scope and Motivation | 12 |
| 1.4 Organization of the Project Report | 13 |
| 2. LITERATURE REVIEW | 15 |
| 2.1 Overview | 16 |
| 2.2 Summary of Literature | 16 |
| 3. DATASETS USED | 18 |
| 3.1 Dataset Description | 19 |
| 3.2 Versions of Dataset | 20 |
| 3.2.1 Version 0: Actual Dataset | 20 |
| 3.2.2 Version 1: Considering Single Country Info | 21 |
| 3.2.3 Version 2: Sampling Dataset on the Basis of Classes | 22 |
| 3.2.4 Version 3: Generating Synthetic Data for Minority Classes | 23 |
| 4. PROPOSED METHODOLOGY | 24 |
| 4.1 Architecture Flow Diagram | 25 |
| 4.2 Data Pre-processing | 27 |
| 4.2.1 Feature Selection | 27 |
| 4.2.2 Feature Generation | 28 |
| 4.2.3 Handling Missing Data | 29 |
| 4.2.3.1 For Categorical Data | 30 |

4.2.3.2 For Continuous Data 30
4.2.4 Handling Outliers 30
4.2.5 Label Encoding for Categorical, Target Variables 31
4.2.6 Generating Train, Test and Validation Splits 33
4.3 Model Used 34
4.3.1 Logistic Regression 34
4.3.2 KNN 35
4.3.3 Support Vector Machines 36
4.3.4 Random Forest Classification 38
4.3.5 Naïve Bayes Classification 40
4.3.6 Boosting Algorithms 41
4.4 Tools and Technologies Used 43
4.4.1 Libraries Used 43
4.4.2 Technologies Used 44
4.5 Graphical User Interface 45
5. RESULTS AND PERFORMANCE EVALUATION 48
5.1 Metrics for Performance Evaluation 49
5.2 Results of all Models on all versions of Dataset 50
5.2.1 Results on Version 1 of Dataset 50
5.2.2 Results on Version 2 of Dataset 51
5.2.3 Results on Version 3 of Dataset 53
6. CONCLUSION 55
7. FUTURE SCOPE 56
REFERENCES AND USEFUL LINKS 57

## LIST OF FIGURES


| Figure. No. | Title | Page No. |
|---|---|---|
| 3.1 | Attributes of dataset | 19 |
| 3.2 | Versions of dataset | 20 |
| 3.3 | Version-0 of dataset | 21 |
| 3.4 | Country-wise accident locations | *22* |
| 3.5 | Version-2 of dataset | 22 |
| 4.1 | Flow Diagram of our project | 25 |
| 4.2 | List of features | 27 |
| 4.3 | Feature Generation using DateTime Module | 28 |
| 4.4 | Missing values of columns | 29 |
| 4.5 | Columns with Categorical Values | 32 |
| 4.6 | Generation of labels for target variables | 32 |
| 4.7 | Splitting into training and testing sets | 33 |
| 4.8 | Logistic Regression | 35 |
| 4.9 | K-Nearest Neighbors | 36 |
| 4.10 | SVM Linear Decision boundary | 36 |
| 4.11 | Non-linear Separable Data | 37 |
| 4.12 | Radial Base Function | 38 |
| 4.13 | Building a decision tree | 39 |
| 4.14 | Gaussian Distribution | 40 |
| 4.15 | Gradient Boosting | 41 |
| 4.16 | Data Flow Control | 45 |
| 4.17 | Graphical User Interface | 46 |
| 4.18 | Output Display page | 46 |

## LIST OF FIGURES


| Figure. No. | Title | Page No. |
|---|---|---|
| 5.1 | Results on Version 1 of dataset | 51 |
| 5.2 | Results on Version 2 of dataset | 52 |
| 5.3 | Results on Version 3 of dataset | 53 |

# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW

Accidents are something that are unfortunate and not predictable which leads to health risks and property damage. Nowadays the accident rate has increased abruptly, taking the lives of more than 1.4 million people per year. And across 20-50 million people suffer non-fatal injuries. They occur commonly due to driver's fault, atmospheric and road conditions, improper traffic management system, etc. Due to the introduction of a large number of vehicles and its varieties, one of the causes of this is over speeding. The consequences due to this are death, injury, and property damage to victims and their families. Therefore, prediction and accident analysis is an important topic and is also a challenge for the safety measures of the public. Hence a tool that is able to predict the occurrence of an accident can save many lives. It may be difficult but it isn't impossible.

To improve road infrastructure safety management road designers, authorities, and safety executioners need some type of prediction models. This project helps us to design such a system that can predict the severity levels of an accident and the percentage of it's occurrence. The dataset used in this project consists of the cases of the USA records taken from traffic cameras, sensors placed in road networks. Each of these records consists of a variety of attributes like location, time, weather, period-of-day, etc. Due to a large collection of records, we propose a new sampling technique to make the dataset shorter and more efficient to use. This obtained data is trained using Machine learning models that produce accurate results according to the code. This process can be carried out by a user interface which shows the results obtained in a more elaborated manner (like percentage of prediction and its severity level). This proposed system can then be used to improvise the safety measures and make the driver aware of the danger and prevent catastrophes.

## 1.2 OBJECTIVES AND IMPORTANCE OF PROJECT

The main objective is that we wanted to create an interactive accident predictor system which can be easily accessible by everyone. To achieve this aim the best way was to create a website consisting of a trained predictor. It should be able to do the following:

- Allow the users to insert some attributes which help in predicting the level of severity of an accident at a potential location.
- It also should show the percentage of occurrence of an accident for those constraints.
- To analyze the previous records and that location and determine the confidence of the event based on other inputs.
- To make the traveler be aware of the danger.

By achieving these objectives, we can reduce the percentage of accidents per year, make people aware of road safety, reduce property losses and decrease the fatality rate. Hence the requirement of achieving these objectives is important to control traffic accidents. Due to this, travelers can be assured of traveling by a road network without the fear of dangers. It is important to have as many as possible tools for this cause. They should all constitute for becoming an optimum solution in overcoming losses.

## 1.3 SCOPE AND MOTIVATION

Road accidents are a very common problem which leads to loss in health and economy. Due to the continuous increase in population transport management is a challenge to the system. The standard methods are not effective enough for upcoming transport management problems. More intelligent, efficient, and economically feasible techniques are to be found. We need more knowledge from a variety of research fields for developing these systems. Different transport systems need help from different branches of science for finding different new practices. Due to an increased number of automobiles, it is important to take the necessary steps to manage traffic and to avoid the inevitable like accidents and losing lives. Once after the prediction of accident possibility at a road network, we can plan to avoid those routes. And vehicles can try to bypass those networks. In order to achieve this, prediction models are used and the travelers can be assured about the danger by traffic sensors and prediction methods.

The future of the safety of roads is unpredictable, and definitely not the same for all places in the world. Firstly we don't know how every country plans on maintaining road safety. And last few decades the number of accident cases have been increased drastically. A road safety system should focus on providing strategies that help in providing road networks that are death free and avoiding injuries. Also the vehicles with safety measures like airbags, etc

should be invented more to delay the inevitable. People should be made aware of new technologies and equipment to be safe on the road. Their support also plays an important role in maintaining road safety.

Hence a proper interface is required to predict the occurrence of an accident and save lives. Their occurrence is depending on many factors like weather and atmospheric conditions which are taken into account. Previous accident history helps us in understanding the relationship between these attributes and accidents. Due to these many factors, the handling of the dataset may be complex and may get an error in prediction.to provide a better solution we sample the dataset and synthesize the data which helps in obtaining a perfect and accurate prediction. Therefore a many million records can be evaluated by using this method with ease and can predict the dangers. Specific patterns and surrounding conditions can help us detect the upcoming dangers. Due to the development of new methods in the following years, we can hope to build a more efficient and accurate system. Maybe we can also predict sudden natural disasters beforehand with the new technological advancements. There can be a lot of developments in this field with the help of proper devotion and making new tools for these requirements.

## 1.4 ORGANIZATION OF THE PROJECT REPORT

The Overall project has been explained in 7 chapters. Each chapter has been considered as an individual component contributing to the overall project. A brief introduction of all the chapters is as follows:

In the first chapter we explained the importance, Objectives, Scope and Motivation of the project. In the second chapter we explained the literature used along with comparisons with various methods and models currently existing.

In the third chapter we introduced the dataset which we used to predict the results. We

performed various modifications on the existing dataset to make it balanced. We first sampled the original dataset, and then to make it balanced we generated synthetic data for minority classes.

In the fourth chapter we explained the Flowchart of our project pipeline. We specified all of these models in fourth chapter. We performed data pre-processing on dataset such as handling missing values, splitting into training and testing set etc. We used various machine learning models to predict the severity level. We have built a GUI (Graphical User Interface) for used to interact with trained model.

In next chapter we described the metrics we used to get the performance of the model such as accuracy, f1-score etc. We projected results of all versions of datasets applied on all models we discussed in chapter 4 of this project.

In next chapters we concluded the project and with stating the future work and other opportunities available to extend the proposed approach.

# CHAPTER-2

# LITERATURE OVERVIEW

## 2.1 OVERVIEW

The idea of road accident prediction is not a new one, the first of this kind of work was done by Mountain et.al in their work titled “Accident Prediction model for roads with minor junctions”. [1]

The dataset which we’ve used in this project is titled “US accidents” created by Moosavi et.al in their work titled “A Countrywide Traffic Accident Dataset” [2]. Since this is the first work that’s being done on this dataset we don’t have any references of results. The references for the machine learning models we’ve used are mentioned in the references section.

## 2.2 SUMMARY OF LITERATURE:

| S.No | Paper Title | Authors | Year of Publication | Comments |
|---|---|---|---|---|
| 1 | Accident Prediction model for roads with minor junctions | Mountain et.al | 1998 | This is the first work that’s done on road accident prediction |
| 2 | A Countrywide Traffic Accident Dataset | Moosavi et.al | 2019 | This paper presents the dataset we’ve used in this project |
| 3 | Accident prediction models for urban roads | Poul Greibe | 2003 | This paper describes some of the main findings from two separate studies on accident prediction models for urban junctions and urban road links |
| 4 | Prediction of Road Accident Severity Using the Ordered Probit Model | Rui Garrido et.al | 2014 | The ordered probit model is used to examine the contribution of several factors to the injury severity faced by motor-vehicle occupants involved in road accidents. |

| | | | | |
|---|---|---|---|---|
| 5 | Application of Adaptive Neuro-fuzzy Inference System for road accident prediction | Mehdi Hosseinpour et.al | 2013 | This present paper aims to develop an ANFIS technique for modelling traffic accidents as a function of road and roadside characteristics |
| 6 | Macrolevel Accident Prediction Models for Evaluating Safety of Urban Transportation Systems | Alireza Hadayeghi et.al | 2013 | A series of macrolevel prediction models that would estimate the number of accidents in planning zones in the city of Toronto, Ontario, Canada, as a function of zonal characteristics |
| 7 | An Artificial Neural Network Model for Road Accident Prediction: A Case Study of a Developing Country | Francisca Nonyelum Ogwueleka | 2014 | Artificial Neural Network (ANN) model for the analysis and prediction of accident rates in a developing country |
| 8 | The use of multilevel models for the prediction of road accident outcomes | Andrew P.Jones | 2003 | This article introduces the potential of a recently developed form of regression models, known as multilevel models, for quantifying the various influences on casualty outcomes |
| 9 | Accident prediction models with random corridor parameters | Karim EL-Basyouny | 2009 | The proposed models were estimated in a Full Bayes context via Markov Chain Monte Carlo (MCMC) simulation and were compared in terms of their goodness of fit and inference |

# CHAPTER - 3

# DATASETS USED

## 3.1 DATASET DESCRIPTION

The dataset that has been used for this project is the open-sourced “US accidents” dataset available in Kaggle. The dataset contains the accident information of several accidents that took place in the United States. By the time this project is being performed the dataset has 3 Million records, having the accident data from 49 States of the United States. And this accident information has been collected from February 2016 to December 2019. This information has been gathered using several data providers, including two APIs that provide streaming traffic incident data. These APIs broadcast the traffic data captured by a variety of entities including both government and private organizations, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks in the United States.

The data initially has 49 attributes, which include the attributes defining the weather condition, kind of road, presence of any railways, or airways near the vehicle and it even has a description of the current weather condition. It takes into consideration the current Temperature, Atmospheric Pressure, Humidity, Wind Direction, and many more. “Severity” is the output variable in this dataset, this indicates the severity level of the accident, on a scale of 0 - 3(means it has 4 output classes in total).

The below screenshot shows what all attributes we have in the original dataset.

```
Index(['ID', 'Source', 'TMC', 'Severity', 'Start_Time', 'End_Time',
       'Start_Lat', 'Start_Lng', 'End_Lat', 'End_Lng', 'Distance(mi)',
       'Description', 'Number', 'Street', 'Side', 'City', 'County', 'State',
       'Zipcode', 'Country', 'Timezone', 'Airport_Code', 'Weather_Timestamp',
       'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)', 'Pressure(in)',
       'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)',
       'Precipitation(in)', 'Weather_Condition', 'Amenity', 'Bump', 'Crossing',
       'Give_Way', 'Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station',
       'Stop', 'Traffic_Calming', 'Traffic_Signal', 'Turning_Loop',
       'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight'],
      dtype='object')
```

Fig 3.1 Attributes of dataset.

For the description of each attribute please refer to this website by dataset creators. It has a detailed description of all the attributes along with saying whether a field can be null or not.

By the time of doing this project, the actual dataset has approximately 3 Million(30 lakhs) records, but due to the huge size of the dataset, we're not able to use the entire dataset for modeling. So as an alternative we've generated various versions of the dataset, which are explained in the following sections.

## 3.2 VERSIONS OF DATASET :

The actual dataset is further modified into various versions of datasets as mentioned below

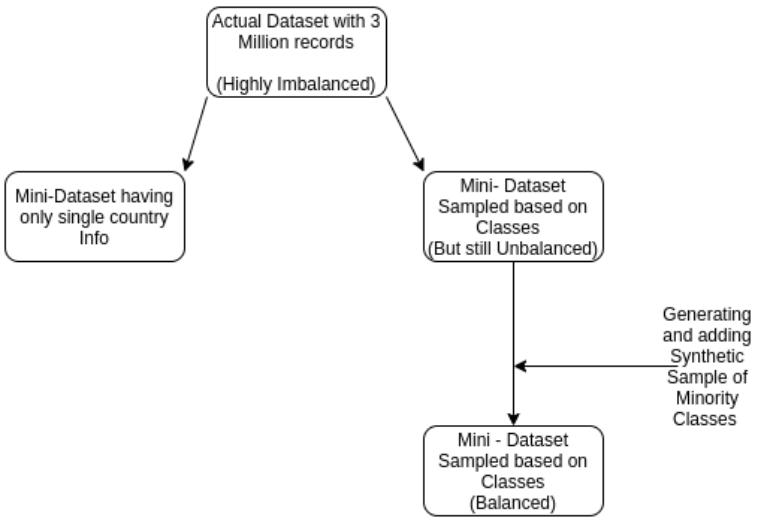



Figure 3.2 Versions of dataset.

### 3.2.1 VERSION 0: ACTUAL DATASET

The actual dataset has 3 million records, and the class distribution is shown below:

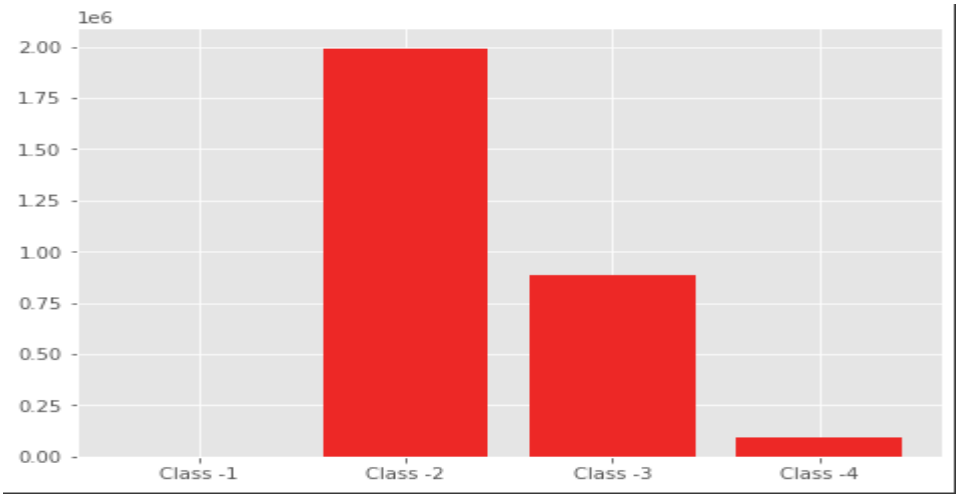



Figure 3.3 Verison-0.

It can be seen from the above class-wise sample distribution, the sample of class -1 is not even comparable with the other classes. And we're also having a relatively very low number of samples for class -4. The number of samples from each class are shown below:

```
Counter({1: 968, 2: 1993410, 3: 887620, 4: 92337})
```

From the above numbers, it can be concluded that the actual dataset is completely imbalanced.

### 3.2.2 VERSION 1: CONSIDERING SINGLE COUNTRY INFO

Now as we have mentioned earlier, the actual dataset has accident information from various countries in the United States. Since we're limited by the computational capabilities available for this project, we've decided on generating a smaller version of the actual dataset, which will have accident information from only one country. We've arbitrarily chosen the country "Montgomery". So using this smaller version of the dataset we're trained various machine learning models and the results have been explained in the Experimental Results section below. Now this newly generated dataset has about 55, 000 data samples.

Different preprocessing techniques used on the sampled data has been explained in the preprocessing section explained below. Once the preprocessing part is completed on the sampled data then various machine learning algorithms have been applied and the results are explained in the Experimental results section.

The accident distribution in the chosen country, color-coded based on the city is plotted in the below map :

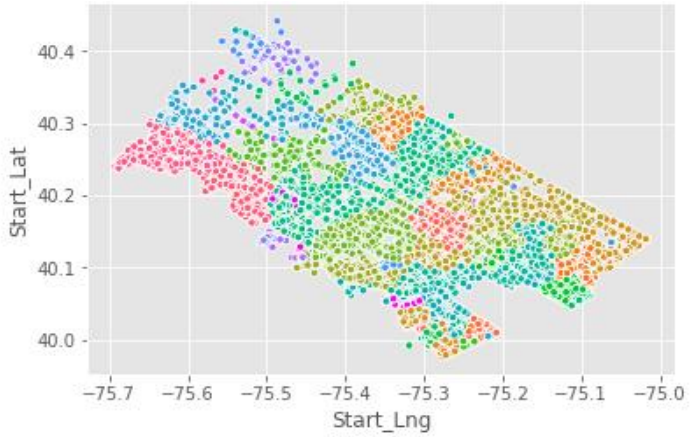



Fig 3.4 Country-wise Accident locations

### 3.2.3 VERSION 2 : SAMPLING DATASET ON THE BASIS OF CLASSES

Now since the major problem with the dataset is its enormous size, so we've tried to sample the dataset based on classes. Initially, there is a heavy imbalance in the original dataset, the num of the sample having Severity Level -1 were 968 and with Severity Level -2 were 1993410. So we've sampled 10,000 samples from each of the classes and generated the mini-version of the dataset having the following class distribution:

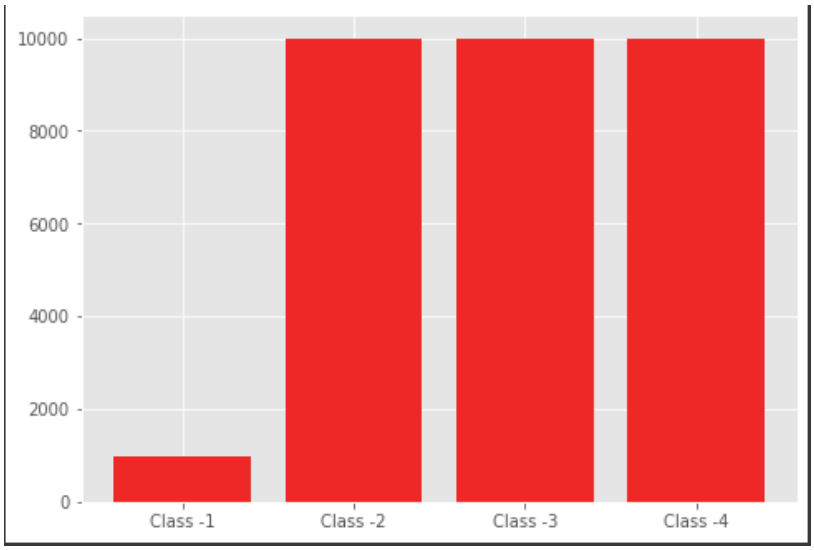



Fig 3.5 Version-2 of dataset.

It can be seen from the above distribution plot that even after sampling the dataset based on classes, the number of samples having Severity Level -1 are obviously still 968. So even after reducing the sample count of the other classes drastically, the resulting sample dataset is still heavily imbalanced. Even if we try to run the machine learning models on this imbalanced dataset, the model won't be able to perform well on the minority class(having Severity Level-1)

So just to make sure our assumptions are true, we've employed various machine learning algorithms on this sampled and imbalanced dataset. And the results clearly show that whichever model we've employed is not performing well in the minority class. The Experimental results sections show the related results.

### 3.2.4 VERSION 3: GENERATING SYNTHETIC DATA FOR MINORITY CLASSES

In the previous version, we've sampled the dataset based on classes but ended up still having an imbalance due to the very low number of samples having Severity Level -1. So as a solution in this version of the dataset we're gonna use some oversampling techniques for generating synthetic data for the minority classes.

We've used the Synthetic Minority Oversampling Technique -Nominal Continuous(SMOTE -NC) for generating the synthetic data samples. Since in the previous version of the dataset we've sampled 10,000 for each class, so we had to generate around 9,000 samples for the minority class to make a total of 10,000 samples having the output class Severity Level -1.

This Synthetic Minority Oversampling Technique -Nominal Continuous(SMOTE -NC) can handle the categorical features too. This algorithm works as follows:

1. Initially, it plots all the minority class data samples
2. Then it uses the K-nearest Neighbours algorithm for selecting the nearest k samples.
3. Then finally generates a new sample as a combination of these k samples.

So finally after using the SMOTE -NC for generating the synthetic samples, the final dataset is completely balanced having 10,000 samples from each class. Now there may be questions such as why haven't we generated some 100000 samples for the minority class making the dataset having more number of records, we have done so because from around 900 samples generating 100000 is not reliable and even if we do so the very nature(or to say distribution) of the minority class will be very diluted. So we had to choose a count through which this nature of the distribution for the minority class will still be significant even in the newly generated data samples. So we ended up generating only 9000 samples for class having Severity Level -1.

# CHAPTER-4

# PROPOSED METHODOLOGY

## 4.1 ARCHITECTURE FLOW DIAGRAM

The main objective of this project was to predict the occurrence the of the accident along with its severity. The dataset we've used for this project is the "US accidents" dataset available in the Kaggle. This dataset currently has 3 million records. It has a total of 49 attributes along with a multi class target variable. The target variable "Severity Level" ranges from 1 to 4. The below figure shows the overall architecture and data flow of the project.

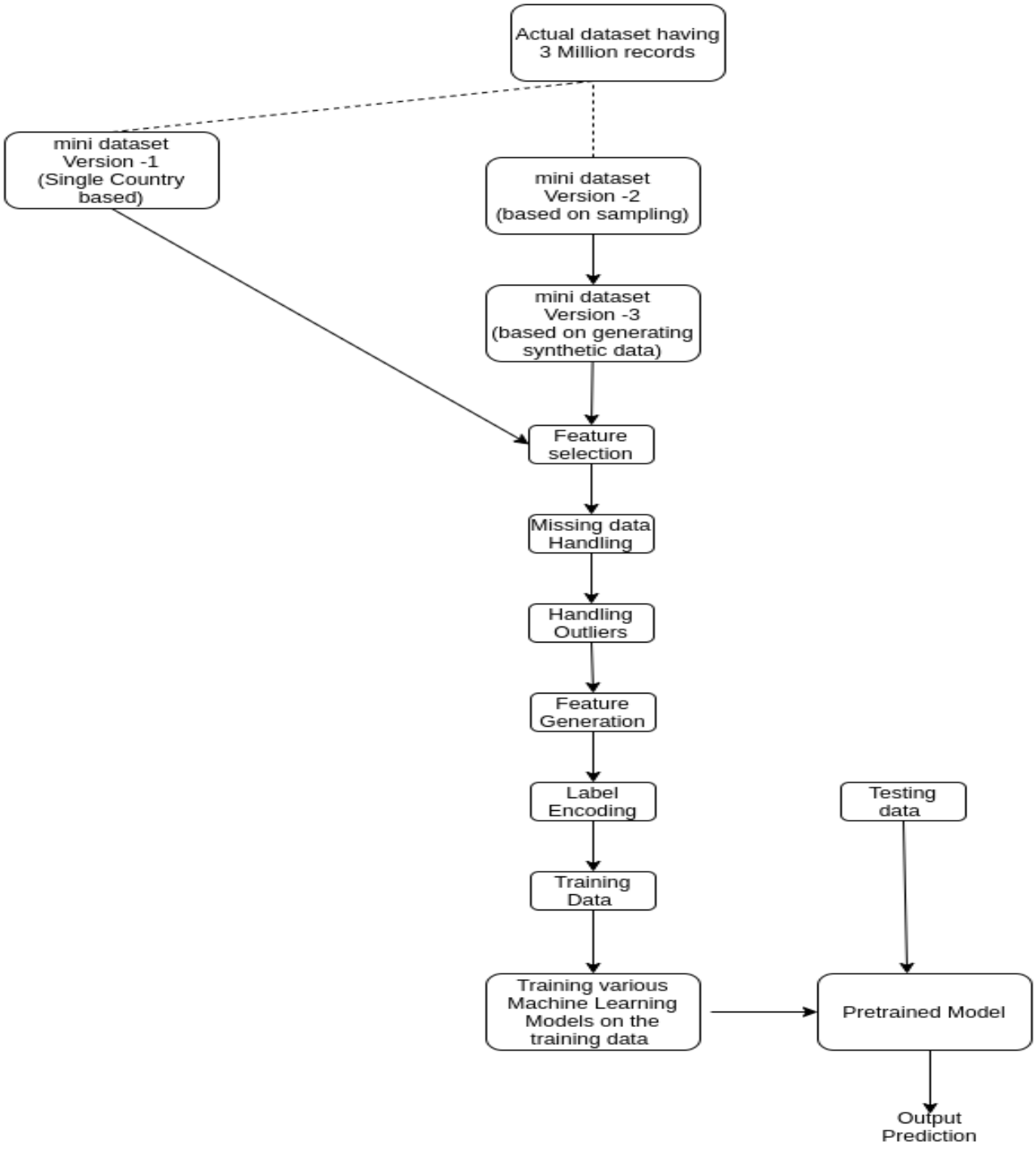



Fig 4.1 Flow Diagram of our project.

The above shown diagram shows the flow of the data in this project. It can be seen that due to the enormous size of the dataset, various mini versions of the dataset we generated. And on each of those versions various machine learning models we used and the associated results are explained in the experimental results section below.

The process initially starts with the data preprocessing phase, where various data transformation methods are used for the transformation of the raw data into modeling ready data. These preprocessing methods include, generating new features, handling missing data for both continuous and categorical attributes, handling the outliers, label encoding of the categorical columns and the target column, feature selection.

Now once the raw data undergoes various data transformation preprocessing methods and becomes modeling ready, then various machine learning models are employed such as Logistic Regression, Support Vector Machines, Boosting algorithms.

After training the models with the training data, then the models are validated using the testing data for their performance. The model which is performing well is chosen as the final model.

This project also has an Graphical User Interface (GUI) functionality, through which the user can give the inputs through an HTML form. These inputs are preprocessed in the same way the initial training was, after the preprocessing is completed, the final transformed data is fed into a pre-trained model which performed the best while testing. This model gives the prediction and severity level of the accident for the given input data. Once this prediction is generated, the output is shown to the user in a well-designed web page designed with HTML, CSS, Bootstrap etc. The generated output contains the predicted severity level of the accident, confidence score of the prediction and a comment which is the result of analyzing the output and says whether the risk is LOW, MODERATE, HIGH or VERY HIGH.

## 4.2 DATA PRE-PROCESSING:

Once the various versions of the dataset are prepared, then several preprocessing techniques are used for making the data modeling ready. These data preprocessing techniques have been explained in detail in the following sections:

### 4.2.1 FEATURE SELECTION:

The initial step in preprocessing the actual data is selecting the useful features for our task of accident prediction. So, keeping this in mind the following features have been selected:

```
feature_lst=[
             'TMC',
             'Severity',
             'Start_Lng','Start_Lat',
             'Side','City','County','State',
             'Timezone',
             'Temperature(F)','Humidity(%)','Pressure(in)', 'Visibility(mi)',
             'Wind_Direction',
             'Weather_Condition','Amenity','Bump',
             'Crossing','Give_Way','Junction','No_Exit','Railway',
             'Roundabout','Station','Stop','Traffic_Calming','Traffic_Signal',
             'Turning_Loop','Sunrise_Sunset','Hour','Weekday',
            ]
```

Fig 4.2 List of features

The features that are relevant for the task of accident prediction are only selected and the rest of the features have been removed from the dataset. A total of 31 features(29 available from the actual dataset and 2 additional features generated through the feature generation step explained in the below section) are being selected from the available 49 features.

The features such as End_Lat (ending latitude of the accident), End_Lng (Ending Longitude of the accident) and many removed features won't be available for us before the accident occurs, this is the reason why we have removed those features.

### 4.2.2 FEATURE GENERATION :

The next preprocessing step is feature generation, in this feature generation step apart from the 29 features that have been selected from the 49 actual features, we're gonna generate 2 more additional features to provide more insights and better modeling data for the machine learning models.

The Start time of an accident is given in the dataset but more information can be extracted from that start time feature such as, what hour of the day it is? Is it a weekend? And many more. For the time being we're gonna extract what hour of the day it is? And is it a weekend or not? Features through feature generation.

Once the feature are generated the final dataset has 31 features in total, 29 from the actual dataset and 2 features that are generated from the feature generation step. Now for the generation of these features we're gonna use the "DateTime" library in python. It has built-in methods for extracting the hour of the day and to extract whether it's a weekend or not.

Below is a code snippet that shows the generation of these new feature:

```
# Convert Start_Time and End_Time to datetypes
df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')

# Extract year, month, day, hour and weekday
df['Year']=df['Start_Time'].dt.year
df['Month']=df['Start_Time'].dt.strftime('%b')
df['Day']=df['Start_Time'].dt.day
df['Hour']=df['Start_Time'].dt.hour
df['Weekday']=df['Start_Time'].dt.strftime('%a')
```

Fig 4.3 Feature Generation using DateTime Module

From the above code snippet it can be seen how simply the feature generation can be done by using the "DateTime" module in Python.

## 4.2.3 HANDLING MISSING DATA :

The next step is to handle the missing data, now in the initia(or to say actual) version of the dataset we have a lot of missing values. We've written a Python script for checking the count of missing values and their presence in each selected column. And the output generated from that script is shown below:

There are 10 columns that have missing values out of 31 columns.

| | Zero Values | Missing(Null) Values | % of Total Missing(Null) Values | Total (Zero + Missing) Values | % Total (Zero + Missing) Values | Data Type |
|---|---|---|---|---|---|---|
| TMC | 0 | 728067 | 24.5 | 728067 | 24.5 | float16 |
| Weather_Condition | 0 | 65932 | 2.2 | 65932 | 2.2 | category |
| Visibility(mi) | 800 | 65691 | 2.2 | 66491 | 2.2 | float16 |
| Humidity(%) | 0 | 59173 | 2.0 | 59173 | 2.0 | float16 |
| Temperature(F) | 651 | 56063 | 1.9 | 56714 | 1.9 | float16 |
| Pressure(in) | 2 | 48142 | 1.6 | 48144 | 1.6 | float16 |
| Wind_Direction | 0 | 45101 | 1.5 | 45101 | 1.5 | category |
| Timezone | 0 | 3163 | 0.1 | 3163 | 0.1 | category |
| Sunrise_Sunset | 0 | 93 | 0.0 | 93 | 0.0 | category |
| City | 0 | 83 | 0.0 | 83 | 0.0 | category |

Fig 4.4 Missing values of columns

From the above image the output of our python script can be seen. From this output, the number of columns having the missing values out of the available columns along with the Statistics such as the percentage of the missing value, the percentage of the zero values and the percentage of the missing values for a columns compared to the actual size of the dataset can also be seen.

Now once this missing value data is available, we can use various techniques to fill(or to say Impute) these missing values. Now since the size of the dataset is very large and using complex imputing options such as using Classifiers / Regressors for predicting the missing categorical / continuous is computationally very expensive, and with the basic resources we've got for this project we can't employ such complex imputing techniques. But we've tried to write the script for such methods using the CATBoost algorithm for computing both categorical and continuous missing values, we were successful in writing the script but, due to the enormous size of the dataset we're not able to the run the script on the entire dataset. And if we opt for reducing the training data for the classifiers / regressors then the predicted value might not be

that accurate and the whole effort for training such a large model would go useless, so due to lack of options we've opted for low computationally expensive options as explained below.

But we are providing the written python script in the project code for using such complex models.

### 4.2.3.1 FOR CATEGORICAL DATA :

Now for imputing the categorical columns data, we have used the most frequent value(or to say category) in that column for imputing the missing values in that column.

This is the most accurate and computationally less expensive method most of the data engineers opt. And one more supporting factor for using this method is that, the columns having missing values in our dataset are having only 25% ar below missing values, that means the amount of missing values is very less than the amount of non-missing values, so imputing with the most common category should not affect the distribution of that columns much.

### 4.2.3.2 FOR CONTINUOUS DATA :

Now for imputing the continuous values we have a handful of options apart from the complex prediction models. Now to keep things simple, we've chosen to impute the continuous missing values with the mean of that column.

Now the above stated method is very cost effective and one more supporting factor for this method is that the continuous columns that are having the missing values are only having below 10% of the missing values compared to the actual size of the dataset. So imputing these missing values with the mean would merely affect the actual distribution of the column.

### 4.2.4 HANDLING OUTLIERS :

Now the next preprocessing step is to handle the outliers. Outliers are those data points in the dataset which are not having similar properties or to say similar distribution as the other data points in the datasets. They represent the extreme values in the dataset i.e min and max values.

Now to identify the outliers we can use the standard deviation of that column. We have assumed that the values which are five standard deviations away from the mean value can be considered as the extreme outliers. Now we have two options for handling the outliers, they are:

1. Dropping the outlier values
2. Imputing the outlier value with the extreme values(min and max)

The second option of imputing the outliers with the extreme values can be used in cases where the amount of training data we have is less, but since this isn't the case with us, imputing the outliers with the extreme value won't gain us anything, so we have opted for dropping the outliers.

The outliers were dropped using the following procedure:

1. Any value 5 standard deviations away from the mean is marked as an outlier
2. The marked outliers are replaced with NaN
3. Finally the NaN values have been dropped from the dataset.

### 4.2.5 LABEL ENCODING FOR CATEGORICAL AND TARGET VARIABLES :

The next step in data preprocessing is generation of the numeric labels for the categorical columns and the target variable. The columns having categories such as "MALE" and "FEMALE" can't be passed directly into the modeling phase, we have to replace these non-numeric values with numeric values so that the model can understand. So to do this we're gonna use the "LabelEncoder" object from the scikit-learn library.

The columns which are having such categorical values are :

```
['Side',
 'City',
 'County',
 'State',
 'Timezone',
 'Wind_Direction',
 'Weather_Condition',
 'Amenity',
 'Bump',
 'Crossing',
 'Give_Way',
 'Junction',
 'No_Exit',
 'Railway',
 'Roundabout',
 'Station',
 'Stop',
 'Traffic_Calming',
 'Traffic_Signal',
 'Turning_Loop',
 'Sunrise_Sunset',
 'Weekday']
```

Fig 4.5 Columns with Categorical values

Apart from the above shown categorical columns the target variable is also a categorical variable having 4 different classes, so we have to apply the label encoding for the target variable too.

The below code snippet shows the generation of labels for the categorical variables and once the labels are generated, the encoder objects are being stored in the pickle format so that we can encode the new data while generating predictions.

```
for col in data.columns:
  if str(data[col].dtype) == "category":
    label_encoder = preprocessing.LabelEncoder()
    label_encoder.fit(df_sel[col])
    data[col] = label_encoder.transform(data[col])
    with open(f"{col}_encoder.pkl", "wb") as f:
      pickle.dump(label_encoder, f)
```

Fig 4.6 Generation of labels for target variable

The same kind of code is used for generation of the labels for the target variable.

### 4.2.6 GENERATING TRAIN, TEST AND VALIDATION SPLITS :

Now once all the data transformation steps are completed we have to split the entire dataset into training, testing and validation sets. Now this splitting is necessary because in order to evaluate the performance of the model, we have to make the model to predict on unseen data during the training phase.

Now consider the entire dataset before the split as 100%, then we divided 60% for training, 20% for testing and 20% for validation. After the splitting is done, the machine learning models are trained on the training data, validated on the validation data and finally tested on the testing data.

Now for splitting the dataset we have used the scikit-learn library's "train_test_split" object method. And while splitting we have also followed the stratified splitting strategy, which makes sure that in each split(Train, Test and Validate) the number of samples of each class remains the same.

The below code snippet shows the stratified splitting example in Python:

```
# For Machine Learning learning
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Fig 4.7 Splitting into training and testing sets.

In the above shown image the entire dataset is being split into train and test sets having 80% and 20% of the data respectively. This example also shows the specification of the stratification of classes.

Now after once the data splitting is completed the splitted data is ready to be used in the modeling phase. The next section explains different models we've used in the modeling phase.

## 4.3 MODELS USED :

To predict the accident severity level we applied different models on various versions of the dataset. All the versions of the dataset are discussed in previous sections.In this section we are going to project the different models used.

The various models used in this project are :

Logistic Regression

KNN

Naive Bayes

Linear SVM

SVM Kernel

Decision Tree Classification

Random Forest Classification

### 4.3.1 LOGISTIC REGRESSION

It is a statistical model in which dependent variable is categorical. It is a linear classification model in which the decision boundary is linear.

Let x1 and x2 be two independent predictors of dependent variable y and we assume a linear relationship between independent and dependent variables with a,b,c as constants.

Let the classes be 0 and 1.

y = c + a*x1 + b*x2

Let p be the probability for y to be 1.

Sigmoid Function :

p(y=1) = 1/1+e-y

Therefore p = 1/1+e-y which means y = log(p/1-p) = c + a*x1 + b*x2

p=1/1+e-(c + a*x1 + b*x2)

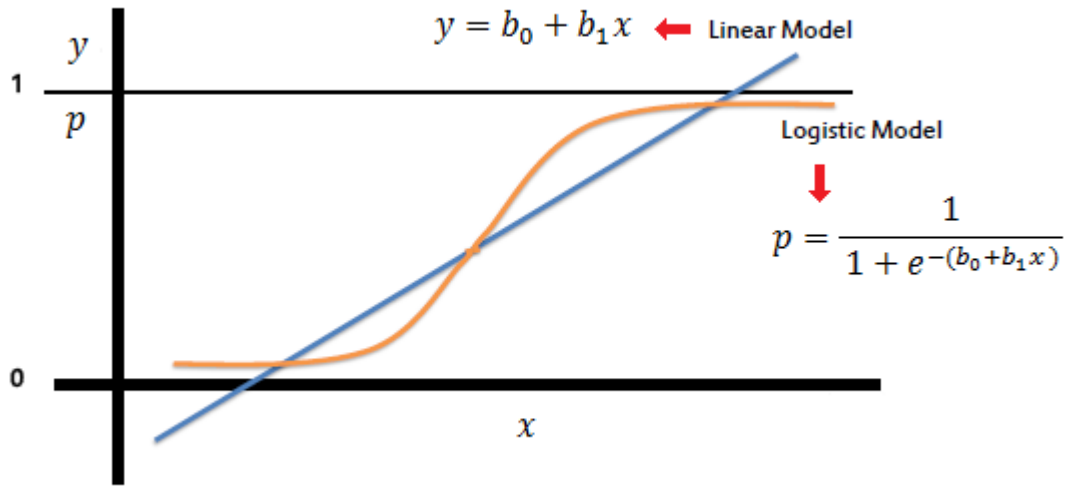



Fig 4.8 Logistic Regression

### 4.3.2 KNN

It is a classification algorithm which classifies new records on similarity measures like distance functions. The decision boundary to classify records in K-Nearest Neighbors algorithm is non-linear that is why K-Nearest Neighbors is non-linear classification model.

There are various metrics used to calculate the distance between two data points like Euclidean distance and Manhattan distance etc. In K-Nearest Neighbour algorithm the classifier fits the training dataset. To classify a random data point, K nearest neighbours to that datapoint are evaluated.Finally the model predicts the class of data point accordingly the majority vote among K nearest neighbours.The good value of K is chosen using heuristic values.The value of K in K-Nearest Neighbors algorithm is chosen wisely to improve accuracy.In the below diagram new data point is predicted to be in category 1 since majority of nearest neighbors(assuming K=5) belongs to category 1.

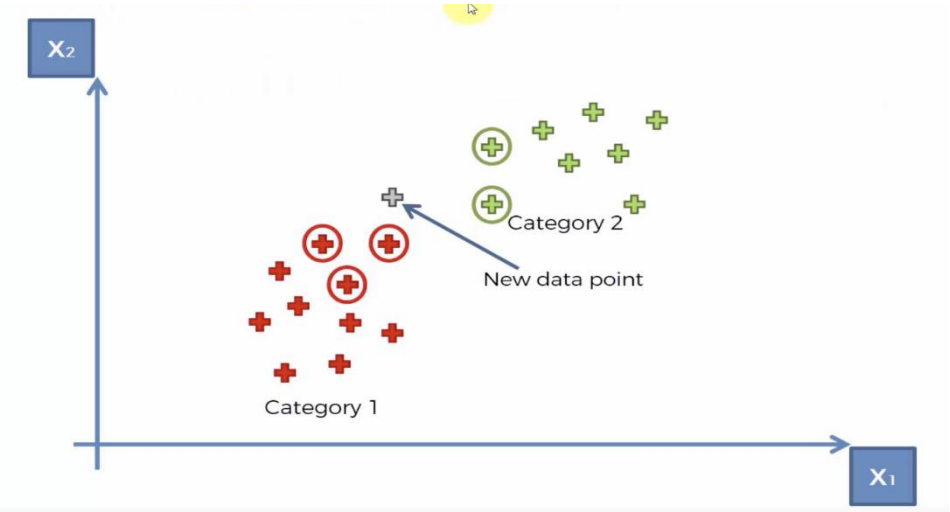



Fig 4.9 K-Nearest Neighbors

### 4.3.3 SUPPORT VECTOR MACHINES

The main objective of Support Vector Machine algorithm is to find a decision boundary that distinctly classifies all the data points. The decision boundary is dot for a one dimensional space(only one attribute), line for two dimensional space(two attributes) and a Hyperplane for N-dimensional space(considering N attributes).

Decision boundary is chosen wisely so that there is maximum margin.let us assume there are two classes of data points. The ultimate goal of the SVM algorithm is to find a hyperplane with maximum margin so that future data points are classified with maximum confidence.

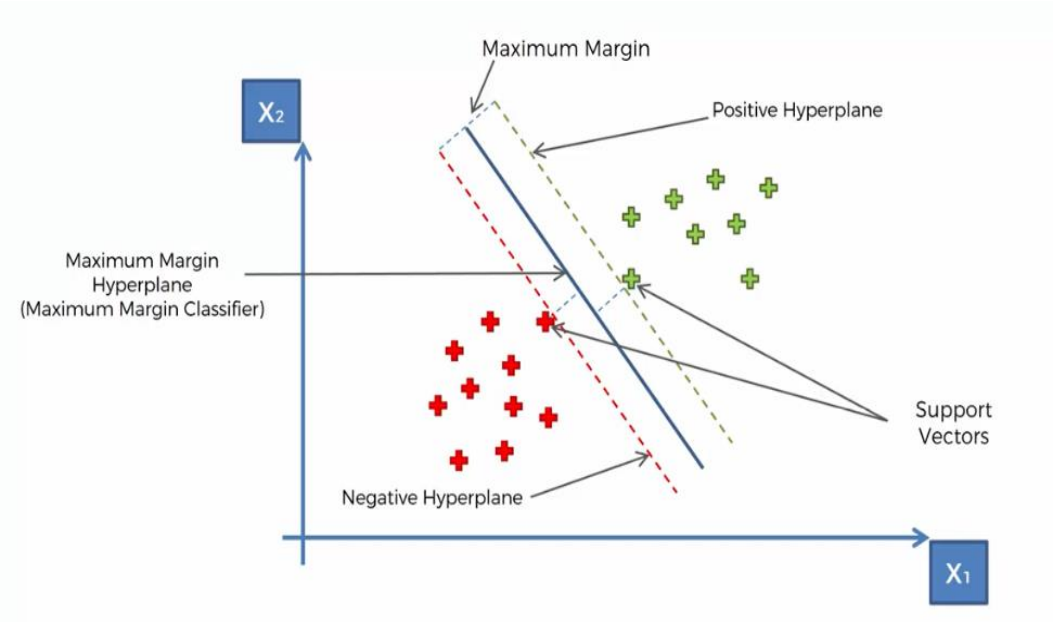



Fig 4.10 SVM Linear Decision boundary

Linear SVM is a machine learning algorithm in which the decision boundary is linear. It classifies the data points in N-Dimensional space linearly with linear hyperplane.
There are many cases in which data points cannot be linearly separable.Linear SVM algorithm fails to classify the data points when data points cannot be linearly separable.
In the above figure data points are of two classes and there are two independent variables. Data points are linearly separable and a hyperplane is chosen with maximum margin.Kernel SVM is used to deal with data points which are non linearly separable.

## GAUSSIAN RBF Kernal SVM :

As discussed above linear SVM is not helpful to deal with data points which cannot be linearly separable. There are many kernel functions to deal with these cases and find a hyperplane that classifies the data points by finding decision boundaries.
In the below figure data points cannot be linearly separable(cannot be separated by straight line)

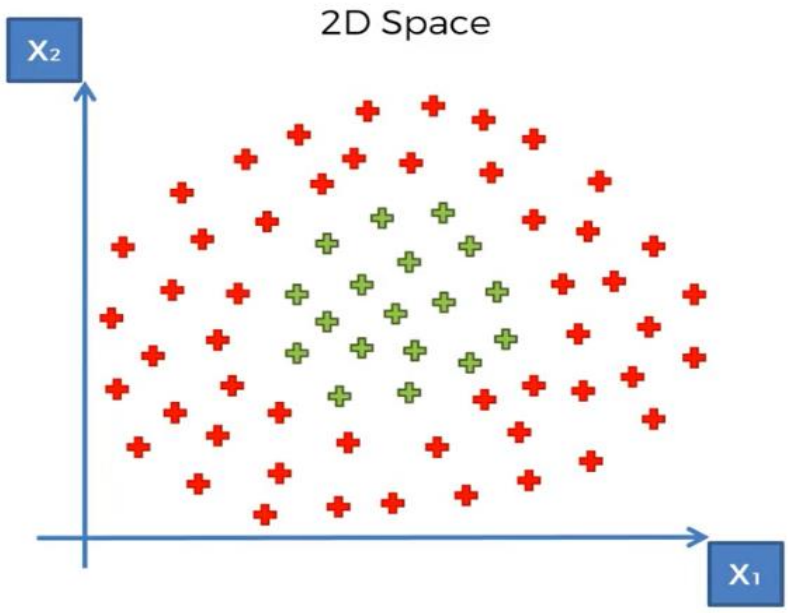



Fig 4.11 Non-linear separable data.

By using Gaussian RBF Kernel we are increasing the dimensionality of the space and decision boundary is drawn in extended dimension.After the data points are separated by hyperplane in higher dimension they are projected to actual dimension and new data points are hence classified by the decision boundary.

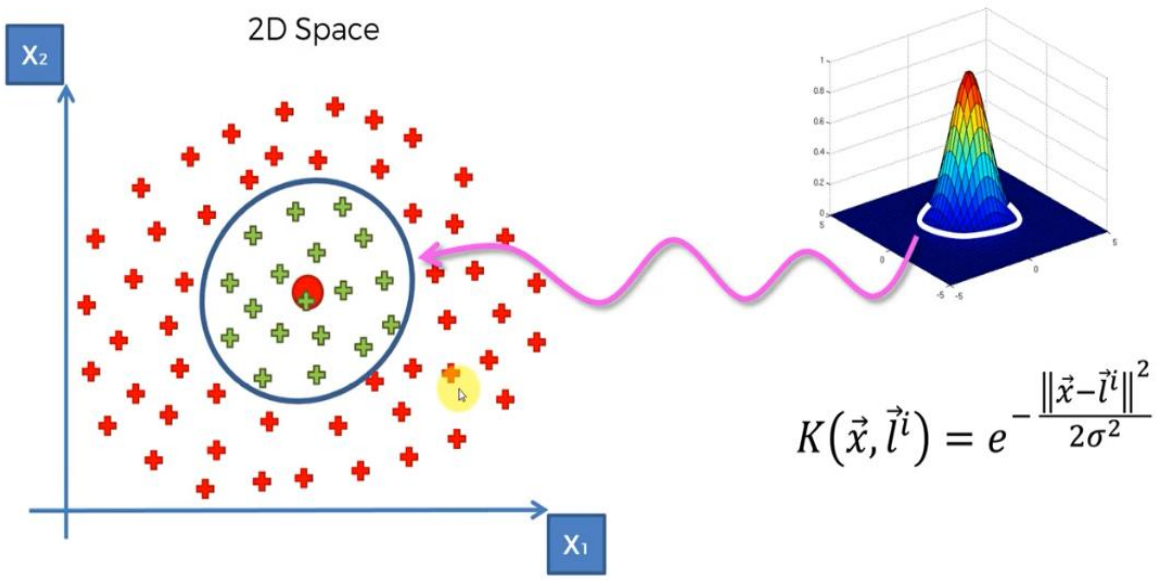



Fig 4.12 Radial Base Function

The equation in the above figure is called Radial Base Function.

L is the landmark vector chosen , x is any point in the plane.l and sigma are choosen wisely accordingly configuration of data points in space. After the RBF function is applied to data points in space a Gaussian curve is generated.All points in XY plane belongs to one class and remaining belongs to other class.

### 4.3.4 RANDOM FOREST CLASSIFICATION

Forest is the random group of decision trees. Decision Trees are the primitive units of Random Forest Classification.

A decision tree is built by splitting the node into subtrees which are subsets of the parent node. In each level of decision tree ,splitting is done in such a way to increase the homogeneity within the leaves. The last level of the decision tree consists of leaves,all the data points in each leaf have the same target variable.The best split or optimal split is to reduce the Information Entropy or Gini Index.

Ensemble learning is taking multiple machine learning algorithms and putting them together to create one bigger machine learning algorithm so that algorithm,so that final model is averaging all algorithms ensembled.Random forest in a kind of Ensemble learning where a lot of decision trees are put together to better predict target dependent variable.

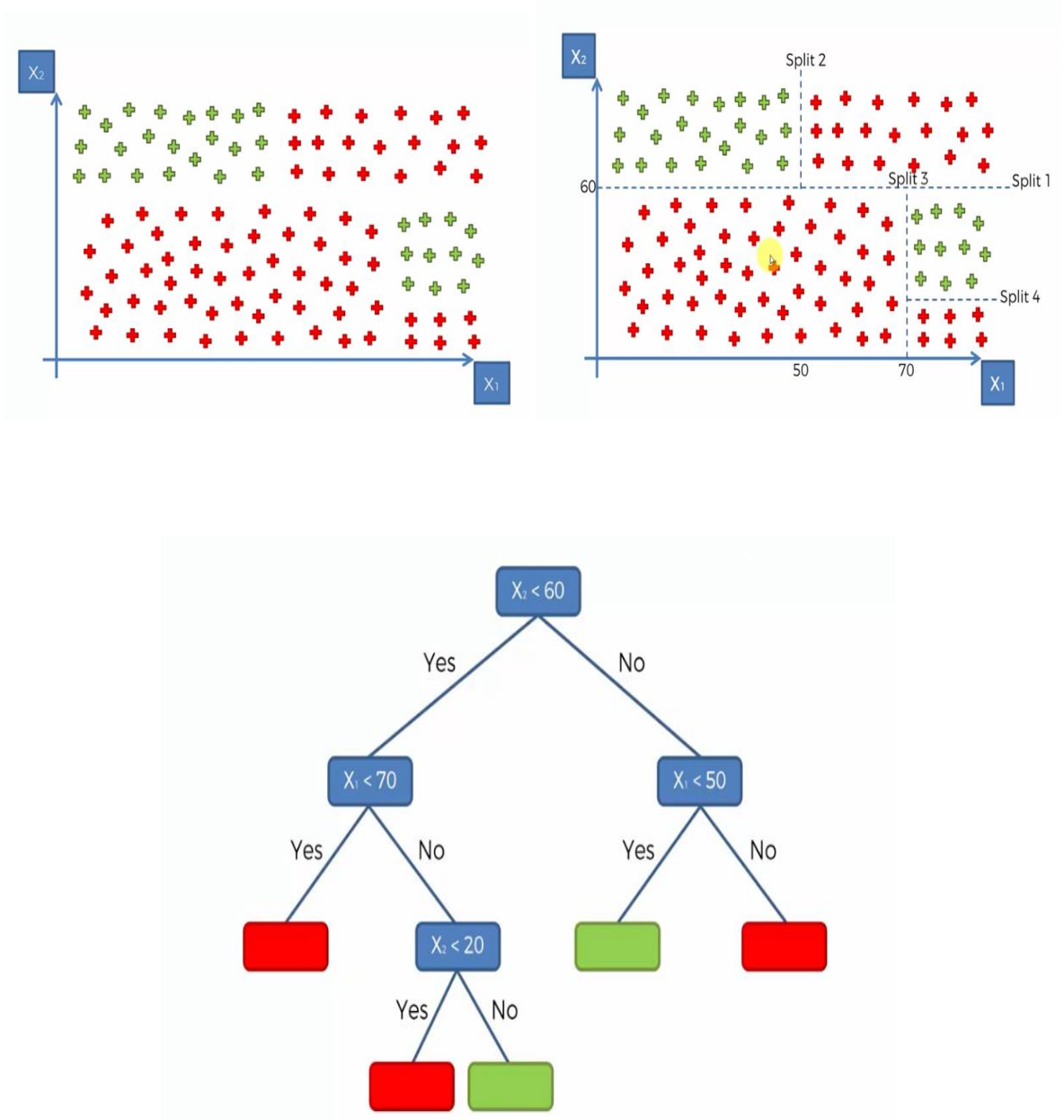



Fig 4.13 Building a decision tree

Random forest algorithm:

Step 1 : Pick random K data points from Training set.

Step 2 : Build the Decision Tree associated with these K data points.

Step 3 : Choose the number Ntree of trees you want to build and repeat Steps 1 & 2

Step 4 : For a new data point, make each of your Ntree trees predict the category to which the data points belong and assign the new data point to the category that wins the majority vote.

## 4.3.5 NAÏVE-BAYES CLASSIFICATION

Naive Bayes is a non linear classification model which works on the basis of Bayes theorem.

Bayes Theorem :

Let A and B are two events then Bayes theorem states that probability of occurence of B after A have occurred is

$$P(B/A) = (P(A/B) * P(B))/P(A)$$

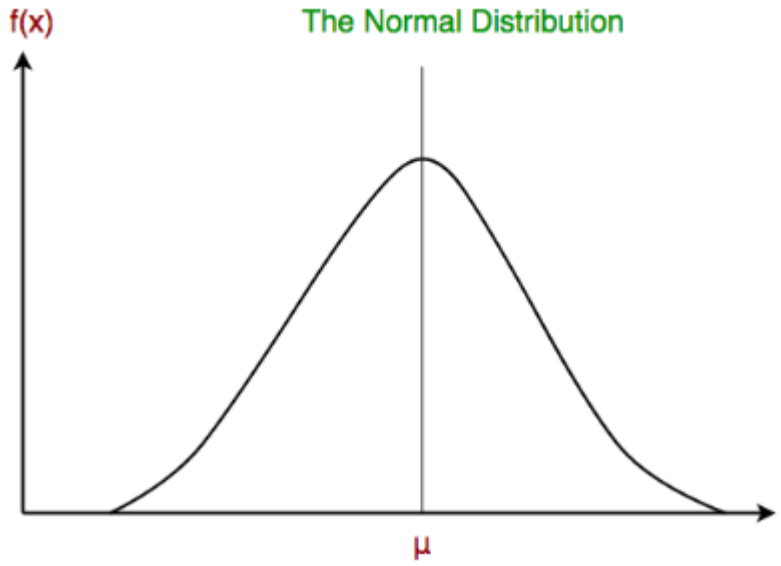
Naive Bayes Classification model assumes that features being classified are independent of each other.

Suppose two events A and B are independent of each other then

$$P(A,B) = P(A) * P(B)$$

Gaussian Naive Bayes Classifier :

The method discussed above only works good for discrete data. If the data is continuous then it must be distributed using Gaussian Distribution which is also called as Normal Distribution.




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

Fig 4.14 Gaussian distribution.

## 4.3.6 BOOSTING ALGORITHMS

Ensemble Learning is taking multiple machine learning algorithms and putting them together to create one bigger machine learning algorithm so that algorithm,so that the final model is averaging all algorithms ensembled.

Boosting is a technique which uses Ensemble Learning by ensembling weak learners to convert into a strong learner.

There are various boosting algorithms. They are :

Adaptive Boosting

Gradient Boosting

Extreme Gradient Boosting

Light GBM

CatBoost

## GRADIENT BOOSTING :

The weak learners in Gradient Boosting algorithms are decision trees. A decision tree is built by splitting the node into subtrees which are subsets of the parent node. It is good to use multiple decision trees rather than one tree to better predict a class of target variables. .All the weak learners are ensembled to convert into Strong learners.

In Gradient Boosting Technique the decision trees are built sequentially. Each new tree takes into account errors of previous trees. This is how decision trees are built sequentially in which each tree built on errors of previously built trees.

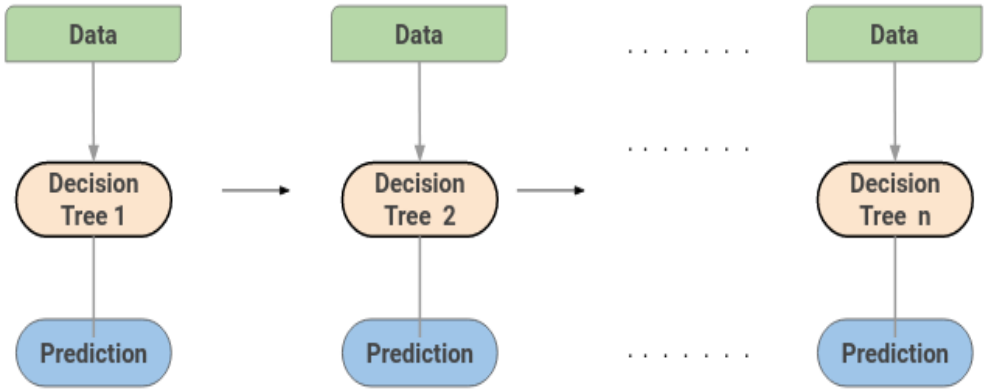



Fig 4.15 Gradient Boosting

## EXTREME GRADIENT BOOSTING :

XGBoost is a one of the popular machine learning boosting algorithms. It is the improvised version of Gradient Boosting technique. It is the same as Gradient Boosting in which decision trees are built sequentially in which the next tree takes care of errors of previous trees.

Extreme Gradient Boosting algorithm is faster than Gradient Boosting algorithm because of parallel processing at node level of decision tree.

This model can handle the missing values on its own.There are various regulation techniques which improve the model performance and reduce hurdles like overfitting etc included in XGBoost Machine.

## CatBoost :

CatBoost is a boosting algorithm which ensembles the weak learners and converts into strong learners. All the Boosting algorithms cannot handle categorical data or string type of data.They must be encoded into numbers using LabelEncoder and then later encoded with OneHotEncoder to handle categorical attributes in the data preprocessing step.

CatBoost in a machine learning boosting algorithm which can handle categorical attributes of data. So there is no need of encoding categorical data since CatBoost internally takes care of categorical data.

CatBoost is a widely used boosting algorithm since it works well with the default set of hyperparameters so that users need not to spend more time on tuning the hyperparameters.

Advantages of CatBoost :

1 . High Performance

2 . Handling Categorical Features automatically

3 . Robust

4 . Easy to use

5 . No need to work more on tuning the hyperparameters.

## 4.4 TOOLS AND TECHNOLOGIES USED

LANGUAGES USED **: Python**

### 4.4.1 LIBRARIES USED:

## SciKit-learn

The scikit learn is a Python library that comes with the implementations for almost all of the machine learning models. The Random Forest Classifier, Support Vector machines...etc are all being used from this package only. Some of the preprocessing techniques implementations are also.

## Pandas

Pandas is a python library that's very useful for the handling of the data. The data can be maintained in the form of pandas “DataFrames”, which are abstract structures similar to tables in relational databases.

## NumPy

NumPy s a package that is used for scientific computations. It can be used as a multi layered container. NumPy can easily generate data of any dimension and can work on it and it comes bundled with python. It consists of -

- Objects of N - dimensional array
- Complex methods
- Linear algebra, random number and Fourier transform functionalities.

Version of NumPy we used in this project is 1.17.1 which is compatible with the python version of 3.5 or newer.

**Matplotlib and Seaborn**

Both matplotlib and seaborn are python libraries which are very useful for visualization. Matplotlib is the base package to say, means it comes with all the properties to form the base visualization experience, and seaborn is kind of an extension package. Seaborn is used to add more customizations to the formed base graph using matplotlib.

**Pickle**

Pickle is a package used for storing the python objects. While storing the objects, it performs searlization where it converts everything into binary format, which can be stored on the disk. During the deserialization process the stored object on the disk is again converted to the python objects. This library is very useful for storing the trained machine learning models, and load them whenever they are necessary.

### 4.4.2 TECHNOLOGIES USED:

**Google Collaboratory:**

The Google Collaboratory or the google colab to be short, is a cloud based service provided by Google. Here we have access to virtual machines having 25GB of RAM long with good GPU support. Google colab gives such sessions that last for 10-12hrs. And the main thing is, this is all free of cost.

**Kaggle:**

Kaggle is a data science competition platform, various competitions are held throughout the year. Apart from hosting various competitions they also provide virtual machines which come along with max 16GB of RAM combined with a good GPU support. Kaggle provides these virtual machine sessions for 9hrs long. And one more useful feature of Kaggle is we can commit versions of our code, once we commit it will be run from the beginning and the output is also saved with version names, so we can refer to it later also. And these virtual machines are completely free of cost.

**Flask:**

FLASK is a python Framework used for designing websites. Flask comes with good in-built features such as dynamic templates, built-in server and many more.

**HTML and CSS:**

HyperText Mark-up language(HTML) and Cascading Style Sheet(CSS) are the basic building block for any web page. HTML is used to design the skeleton of the web page and CSS is used to give customization to that skeleton.

**Bootstrap:**

Bootstrap is used to give good visualization effects for the basic web page designed using HTML and CSS.

## 4.5 GRAPHICAL USER INTERFACE

Just designing the Machine Learning model isn't sufficient, we should be able to be able to take the input from the user and run the pre-trained machine Learning models on that given input and then finally display the generated output(or to say prediction) to the user.

We've used the Python Framework Flask, for creating this user interface. Flask is a very light yet powerful Framework which comes with very good features such as dynamic templates, built-in server support and many more.

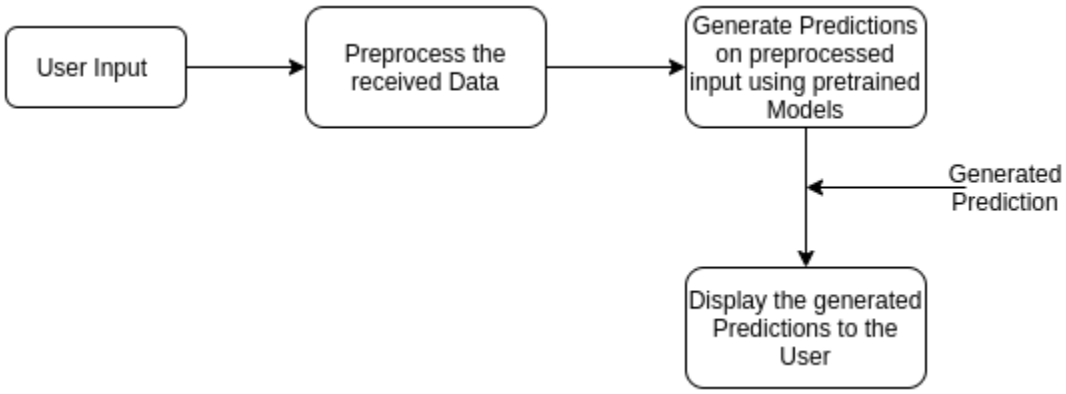



Fig 4.16 Data Flow Control

The above shown image shows the flow of data in the designed user interface. This initially starts with the user giving the input through a HTML form. Once the user presses the submit button the submitted data will come to the backend. In the backend, the data from the POST request is retrieved and the retrieved data will undergo certain data preprocessing steps, the same steps the data will go initially, these steps are explained in the above Preprocessing section.

Once the preprocessing of the retrieved data is completed then a pre-trained model is used and this preprocessed data is given as input. The pre-trained model takes the input and generates the output prediction. This prediction is passed on to the Frontend and displayed to the user through a nice design, designed with HTML, CSS and Bootstrap.

The below image shows the HTML form given to the use for providing the input data:

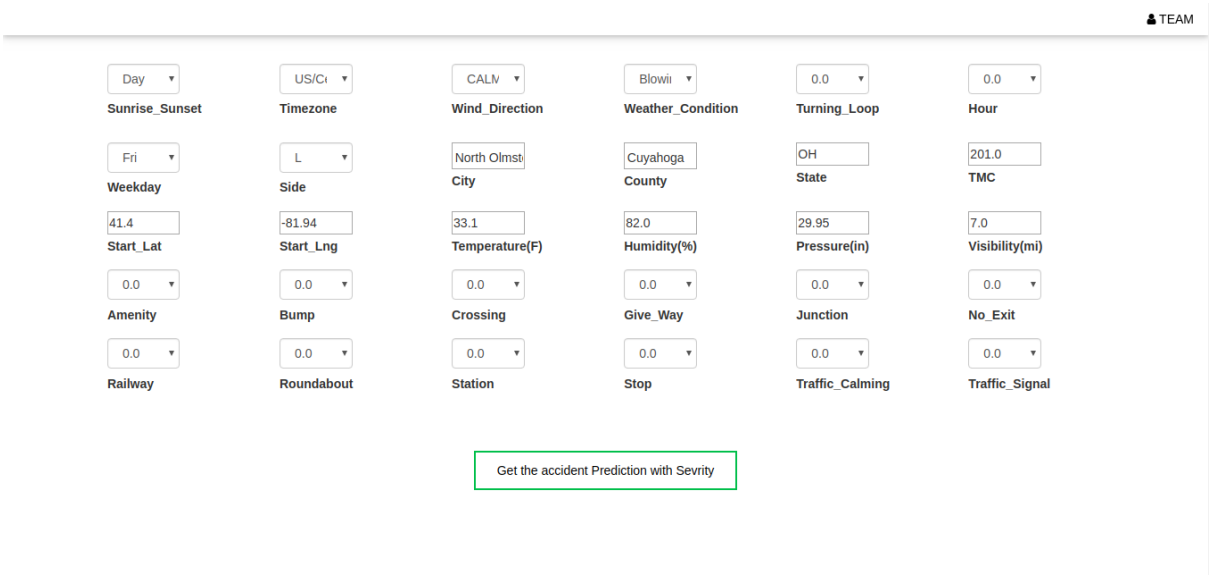



Fig 4.17 Graphical User Interface

Once all the inputs are filled then the user presses the submit button saying “Get the accident prediction with severity”. Once the user presses this button the given input data is passed to the backend system.

Now after once the user presses the submit button, the given input data is passed to the backend, where the preprocessing of the given input data is done and the final preprocessed data is passed on to a pre-trained machine learning model, and the output prediction generated from the model is passed to the front end to display the out to the user. The below image shows the output that is visible to the user once he submits the filled input form:

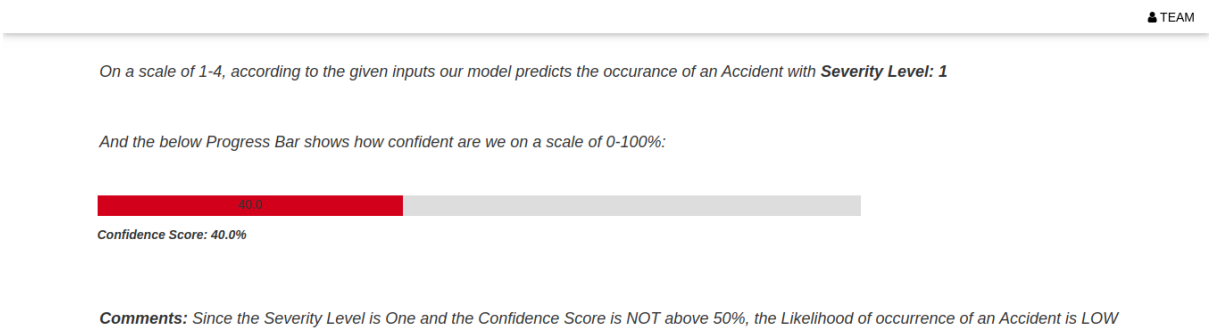



Fig 4.18 Output display page.

From the above image it can be seen that, along with providing the prediction for the severity level of the accident we're also providing with the confidence score. Additionally we're also providing a comment at the bottom, which analyzes the result and says to the point, what will be the probability of the occurrence of the accident and how severe it is, the probability might range from LOW, Moderate, HIGH to VERY HIGH.

Now when the comment includes that the probability of the accident is HIGH to VERY HIGH then it is considered as a serious issue.

# CHAPTER 5

# RESULTS AND PERFORMANCE EVALUATION

## 5.1 METRICS FOR PERFORMANCE EVALUATION

For evaluating the performance of various machine learning models we have used the Accuracy, Precision, Recall and F1 score as metrics. These metrics can be defined as:

Now for any prediction the following four cases are possible
True Positive(TP) – Output actually is Positive and the prediction is also Positive
True Negative(TN) – Output actually is Negative and the prediction is also Negative
False Positive(FP) – Output actually is Negative and the prediction is also Positive
False Negative(FN) – Output actually is Positive and the prediction is also Negative

Accuracy is the proportion of correct estimates to the total number of estimates. Accuracy is a very important performance evaluator and is used to evaluate many model's performance, accuracy is total number of predictions that are correct to the total number of predictions made by the model.
Precision means what fraction of predicted positives are actually correct and f1-score is HM(harmonic-mean) of recall which is nothing but sensitivity while calculating over confusion matrix and precision for both the models
Once these four classes are identified from the predictions, now the metrics can be defined as

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

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

$$\text{F1score} = \frac{2*\text{Precision}*\text{recall}}{\text{Precision+recall}}$$

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

Now after once these metrics are defined, now we can use these metrics for evaluating our machine learning models. Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive. Recall actually calculates

how many of the Actual Positives our model captures through labeling it as Positive (True Positive). And finally, the F1 score is a kind of function for precision and recall.

### 5.2 RESULTS OF ALL MODELS ON ALL VERSIONS OF DATASET

This section contains all the results obtained throughout this project. As initially stated, various versions of the dataset were generated from the actual dataset, and this section explains the results obtained by employing various machine learning models on each of those versions.

#### 5.2.1 RESULTS ON VERSION 1 OF DATASET:

Now initially, we'll demonstrate the results we've obtained by employing various machine learning models on the first version of the mini dataset i.e the dataset containing only a single country information. Once the dataset is ready a series of preprocessing steps are applied for data transformation. This modeling ready data is split into 80% training and 20% testing data. The performance of different models trained using this training data is:

[Logistic regression algorithm] accuracy_score: 0.849.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 3 |
| 1 | 0.86 | 0.97 | 0.91 | 8331 |
| 2 | 0.69 | 0.30 | 0.42 | 1832 |
| 3 | 0.00 | 0.00 | 0.00 | 15 |
| accuracy | | | 0.85 | 10181 |
| macro avg | 0.39 | 0.32 | 0.33 | 10181 |
| weighted avg | 0.83 | 0.85 | 0.82 | 10181 |

[Naive Bieas algorithm] accuracy_score: 0.468.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 566 |
| 1 | 0.51 | 0.99 | 0.67 | 4294 |
| 2 | 0.27 | 0.29 | 0.28 | 1719 |
| 3 | 0.60 | 0.00 | 0.00 | 3602 |
| accuracy | | | 0.47 | 10181 |
| macro avg | 0.35 | 0.32 | 0.24 | 10181 |
| weighted avg | 0.47 | 0.47 | 0.33 | 10181 |

[KNN algorithm] accuracy_score: 0.839.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 0 |
| 1 | 0.95 | 0.86 | 0.91 | 9211 |
| 2 | 0.32 | 0.61 | 0.42 | 970 |
| 3 | 0.00 | 0.00 | 0.00 | 0 |
| accuracy | | | 0.84 | 10181 |
| macro avg | 0.32 | 0.37 | 0.33 | 10181 |
| weighted avg | 0.89 | 0.84 | 0.86 | 10181 |

[Randon forest algorithm] accuracy_score: 0.909.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 3 |
| 1 | 0.92 | 0.97 | 0.95 | 8331 |
| 2 | 0.82 | 0.64 | 0.72 | 1832 |
| 3 | 1.00 | 0.13 | 0.24 | 15 |
| accuracy | | | 0.91 | 10181 |
| macro avg | 0.69 | 0.44 | 0.47 | 10181 |
| weighted avg | 0.91 | 0.91 | 0.90 | 10181 |

[Shocastic Gradient Descent algorithm] accuracy_score: 0.879.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 0 |
| 1 | 0.93 | 0.92 | 0.93 | 8462 |
| 2 | 0.63 | 0.68 | 0.65 | 1719 |
| 3 | 0.00 | 0.00 | 0.00 | 0 |
| accuracy | | | 0.88 | 10181 |
| macro avg | 0.39 | 0.40 | 0.40 | 10181 |
| weighted avg | 0.88 | 0.88 | 0.88 | 10181 |

[SVM algorithm] accuracy_score: 0.833.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 0 |
| 1 | 1.00 | 0.83 | 0.91 | 10000 |
| 2 | 0.09 | 0.90 | 0.16 | 181 |
| 3 | 0.00 | 0.00 | 0.00 | 0 |
| accuracy | | | 0.83 | 10181 |
| macro avg | 0.27 | 0.43 | 0.27 | 10181 |
| weighted avg | 0.98 | 0.83 | 0.89 | 10181 |

Fig 5.1 Results on Version 1 of dataset

From among all the above shown models, the Random Forest Classifier is performing the best with **90.9% testing accuracy.** But the problem with this version of the dataset is, even though the overall accuracy of the model is good enough, but the model is performing very poorly on the class having Severity Level 1, since it merely has 900 samples overall, and now since we're considering a single country data, the number of samples belonging to severity level 1 in that country are even very less, may be around 50 samples might be present. This is the reason why the model isn't performing well on the class severity level 1, since it has less data samples.

Since in the previous version of the dataset, the difference between the number of samples from different classes is very large, say for example in the previous version we've tried we had around 41,000 samples for severity level 2, 9136 samples for severity level 3, 74 samples for severity level 4 and 13 samples for severity level 1. The difference between the number of samples from each class is very significant during the training of the model, it won't be able to learn well from the classes which are having less number of samples.

### 5.2.2 RESULTS ON VERSION 2 OF DATASET:

So now in this version of the dataset we've reduced the number of samples from classes which are having very high numbers of samples. We've taken 10,000 samples from each of the class, this way the difference between the number of samples in each class has been significantly reduced. Now after generating this second version of the dataset we have used various machine learning models and the corresponding results are as follows:

[Logistic regression algorithm] accuracy_score: 0.662.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 10 |
| 1 | 0.66 | 0.54 | 0.60 | 2440 |
| 2 | 0.54 | 0.60 | 0.56 | 1800 |
| 3 | 0.85 | 0.87 | 0.86 | 1944 |
| accuracy | | | 0.66 | 6194 |
| macro avg | 0.51 | 0.50 | 0.51 | 6194 |
| weighted avg | 0.68 | 0.66 | 0.67 | 6194 |

[KNN algorithm] accuracy_score: 0.798.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.18 | 0.42 | 0.25 | 84 |
| 1 | 0.75 | 0.75 | 0.75 | 2012 |
| 2 | 0.80 | 0.75 | 0.77 | 2159 |
| 3 | 0.90 | 0.93 | 0.91 | 1939 |
| accuracy | | | 0.80 | 6194 |
| macro avg | 0.66 | 0.71 | 0.67 | 6194 |
| weighted avg | 0.81 | 0.80 | 0.80 | 6194 |

[Naive Bieas algorithm] accuracy_score: 0.607.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 8 |
| 1 | 0.03 | 0.55 | 0.06 | 105 |
| 2 | 0.96 | 0.48 | 0.64 | 4032 |
| 3 | 0.89 | 0.87 | 0.88 | 2049 |
| accuracy | | | 0.61 | 6194 |
| macro avg | 0.47 | 0.47 | 0.39 | 6194 |
| weighted avg | 0.92 | 0.61 | 0.71 | 6194 |

[Randon forest algorithm] accuracy_score: 0.899.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.32 | 0.84 | 0.46 | 74 |
| 1 | 0.86 | 0.90 | 0.88 | 1914 |
| 2 | 0.91 | 0.86 | 0.88 | 2132 |
| 3 | 0.98 | 0.95 | 0.96 | 2074 |
| accuracy | | | 0.90 | 6194 |
| macro avg | 0.77 | 0.88 | 0.80 | 6194 |
| weighted avg | 0.91 | 0.90 | 0.90 | 6194 |

[SVM algorithm] accuracy_score: 0.477.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 0 |
| 1 | 0.00 | 0.00 | 0.00 | 0 |
| 2 | 1.00 | 0.38 | 0.55 | 5219 |
| 3 | 0.48 | 0.98 | 0.64 | 975 |
| accuracy | | | 0.48 | 6194 |
| macro avg | 0.37 | 0.34 | 0.30 | 6194 |
| weighted avg | 0.92 | 0.48 | 0.57 | 6194 |

[Naive Bieas algorithm] accuracy_score: 0.706.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.70 | 0.30 | 0.42 | 450 |
| 1 | 0.33 | 0.86 | 0.47 | 763 |
| 2 | 0.93 | 0.59 | 0.72 | 3130 |
| 3 | 0.86 | 0.93 | 0.90 | 1851 |
| accuracy | | | 0.71 | 6194 |
| macro avg | 0.70 | 0.67 | 0.63 | 6194 |
| weighted avg | 0.82 | 0.71 | 0.72 | 6194 |

[Voting Classifier Algorithm] accuracy_score: 0.915.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.44 | 0.83 | 0.57 | 102 |
| 1 | 0.87 | 0.92 | 0.90 | 1897 |
| 2 | 0.94 | 0.87 | 0.90 | 2148 |
| 3 | 0.98 | 0.96 | 0.97 | 2047 |
| accuracy | | | 0.92 | 6194 |
| macro avg | 0.81 | 0.90 | 0.84 | 6194 |
| weighted avg | 0.92 | 0.92 | 0.92 | 6194 |

[XGBoost algorithm] accuracy_score: 0.912.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.45 | 0.76 | 0.57 | 116 |
| 1 | 0.87 | 0.92 | 0.89 | 1905 |
| 2 | 0.93 | 0.87 | 0.90 | 2134 |
| 3 | 0.98 | 0.96 | 0.97 | 2039 |
| accuracy | | | 0.91 | 6194 |
| macro avg | 0.81 | 0.88 | 0.83 | 6194 |
| weighted avg | 0.92 | 0.91 | 0.91 | 6194 |

Fig 5.2 Results on Version 2 of dataset

From among all the above used models the Voting classifier, which is a combination of the Random Forest and XGBoost classifier is performing the best with **91.5% testing accuracy.** And we can also observe that the precision recall and to say, the overall performance on the minority classes is increased very well when compared to the previous versions. So this is evident that by reducing the difference between the samples of different classes the overall performance can be improved.

### 5.2.3 RESULTS ON VERSION 3 OF DATASET :

Now as a step to further improve the performance, we're gonna use synthetic data, and make the dataset completely balanced, having 10,000 samples for each class. Now by running various machine learning models on this balanced dataset the corresponding resluts are as follows:

[Logistic regression algorithm] accuracy_score: 0.592.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.61 | 0.58 | 1805 |
| 1 | 0.70 | 0.52 | 0.60 | 2703 |
| 2 | 0.46 | 0.56 | 0.50 | 1647 |
| 3 | 0.66 | 0.71 | 0.68 | 1845 |
| accuracy | | | 0.59 | 8000 |
| macro avg | 0.59 | 0.60 | 0.59 | 8000 |
| weighted avg | 0.61 | 0.59 | 0.59 | 8000 |

[KNN algorithm] accuracy_score: 0.835.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.88 | 0.91 | 2153 |
| 1 | 0.74 | 0.77 | 0.75 | 1915 |
| 2 | 0.81 | 0.75 | 0.78 | 2141 |
| 3 | 0.85 | 0.95 | 0.90 | 1791 |
| accuracy | | | 0.83 | 8000 |
| macro avg | 0.83 | 0.84 | 0.83 | 8000 |
| weighted avg | 0.84 | 0.83 | 0.83 | 8000 |

[Naive Bieas algorithm] accuracy_score: 0.694.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.70 | 0.77 | 2420 |
| 1 | 0.30 | 0.78 | 0.44 | 779 |
| 2 | 0.92 | 0.56 | 0.69 | 3301 |
| 3 | 0.70 | 0.94 | 0.80 | 1500 |
| accuracy | | | 0.69 | 8000 |
| macro avg | 0.69 | 0.74 | 0.68 | 8000 |
| weighted avg | 0.80 | 0.69 | 0.71 | 8000 |

[Randon forest algorithm] accuracy_score: 0.919.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.96 | 0.96 | 1982 |
| 1 | 0.85 | 0.89 | 0.87 | 1913 |
| 2 | 0.91 | 0.86 | 0.88 | 2134 |
| 3 | 0.96 | 0.97 | 0.96 | 1971 |
| accuracy | | | 0.92 | 8000 |
| macro avg | 0.92 | 0.92 | 0.92 | 8000 |
| weighted avg | 0.92 | 0.92 | 0.92 | 8000 |

[SVM algorithm] accuracy_score: 0.379.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.29 | 0.45 | 6789 |
| 1 | 0.03 | 0.63 | 0.06 | 101 |
| 2 | 0.05 | 0.66 | 0.10 | 157 |
| 3 | 0.45 | 0.95 | 0.61 | 953 |
| accuracy | | | 0.38 | 8000 |
| macro avg | 0.38 | 0.63 | 0.30 | 8000 |
| weighted avg | 0.89 | 0.38 | 0.45 | 8000 |

[XGBoost algorithm] accuracy_score: 0.934.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.98 | 0.97 | 1971 |
| 1 | 0.87 | 0.91 | 0.89 | 1914 |
| 2 | 0.93 | 0.87 | 0.90 | 2124 |
| 3 | 0.97 | 0.98 | 0.97 | 1991 |
| accuracy | | | 0.93 | 8000 |
| macro avg | 0.93 | 0.94 | 0.93 | 8000 |
| weighted avg | 0.93 | 0.93 | 0.93 | 8000 |

[Voting Classifier Algorithm] accuracy_score: 0.932.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.97 | 0.97 | 1978 |
| 1 | 0.86 | 0.92 | 0.89 | 1888 |
| 2 | 0.93 | 0.87 | 0.90 | 2152 |
| 3 | 0.97 | 0.98 | 0.97 | 1982 |
| accuracy | | | 0.93 | 8000 |
| macro avg | 0.93 | 0.93 | 0.93 | 8000 |
| weighted avg | 0.93 | 0.93 | 0.93 | 8000 |

Fig 5.3 Results on Version 3 of dataset

From among all the models XGBoost classifier was performing the best with 93.4% testing accuracy. Apart from improving the overall accuracy, the class wise performance of the models has also been improved, now the models are performing equally well on all the classes. So this version of the dataset stands out to be the best version.

# 6.CONCLUSION

Every day a lot of people are losing their lives around the globe, due to road accidents. We can avoid these accidents if we can predict the occurence of the accidents beforehand and alert the nearby traffic police of this accident. To accomplish this we're proposing a project based on machine learning. The dataset we've used is the "US accidents" dataset, which is free available from Kaggle. This dataset has around 3 million records, and has a multi class target variable showing the severity level of the accident on a scale of 1-4. But due to the enormous size and the unbalanced classes we have in this dataset we've opted for the generation of various versions of this dataset. One the various versions of the dataset are generated, then a wide range of machine learning models have been employed for prediction of the accidents. We have designed the Graphical User Interface, in the form of a web page, so that even common users can interact with the designed model.

# 7.FUTURE SCOPE

The future scope of this project includes finding out the most effective ways of generating synthetic data in large amounts without deviating from the distribution of the actual data. Due to the short time constraint we weren't able to employ complex deep learning models, so employing various deep learning models would be a great concept to explore. If we can be able to generate more meaningful synthetic data in large numbers. By employing more complex deep learning models we can design much more reliable prediction models. In this project we've designed a web page to interact with the model, we can actually build an App and also design an API for the same. Through that API whenever there is high probability for the occurence of an accident then an alert message can be sent to the nearest traffic police along with vehicle number.

## REFERENCES AND USEFUL LINKS


[1] Mountain, Linda, Bachir Fawaz, and David Jarrett. "Accident prediction models for roads with minor junctions." *Accident Analysis & Prevention* 28.6 (1996): 695-707.

[2] Moosavi, S., Samavatian, M.H., Parthasarathy, S. and Ramnath, R., 2019. A Countrywide Traffic Accident Dataset. *arXiv preprint arXiv:1906.05409*.

[3] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.

[4] Marti A. Hearst. 1998. Support Vector Machines. IEEE Intelligent Systems 13, 4 (July 1998), 18–28. DOI:https://doi.org/10.1109/5254.708428

[5] Mountain, Linda, Bachir Fawaz, and David Jarrett. "Accident prediction models for roads with minor junctions." *Accident Analysis & Prevention* 28.6 (1996): 695-707.

[6] Jones, Andrew P., and Stig H. Jørgensen. "The use of multilevel models for the prediction of road accident outcomes." *Accident Analysis & Prevention* 35.1 (2003): 59-69.

[7] Ogwueleka, Francisca Nonyelum, et al. "An artificial neural network model for road accident prediction: a case study of a developing country." *Acta Polytechnica Hungarica* 11.5 (2014): 177-197.

[8] Hadayeghi, Alireza, Amer S. Shalaby, and Bhagwant Persaud. "Macrolevel accident prediction models for evaluating safety of urban transportation systems." *Transportation research record* 1840.1 (2003): 87-95.

[9] Garrido, Rui, et al. "Prediction of road accident severity using the ordered probit model." *Transportation Research Procedia* 3 (2014): 214-223.

[10] https://scikit-learn.org/stable/

[11] https://www.kaggle.com/

[12] https://colab.research.google.com/

[13] https://xgboost.readthedocs.io/en/latest/