Predicting Road Accidents and Risks Using Machine Learning

A Project Report

Submitted by:

Abhishek Dhankar (20030142002)

Anmol Bhimrajka (20030142007)

in partial fulfilment for the award of the degree

of

M.SC. COMPUTER APPLICATION

IN

COMPUTER STUDIES

at

SYMBIOSIS INSTITUTE OF COMPUTER STUDIES AND RESEARCH, PUNE, INDIA

AFFILIATED TO SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY) (INDIA)

NOVEMBER 2021

DECLARATION

I hereby declare that the project entitled **"Predicting Road Accidents and Risks Using Machine Learning"** submitted for the M.Sc. Computer Application degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Abhishek Dhankar (20030142002)

Anmol Bhimrajka (20030142007)

Signature of Student

Place: PUNE

Date: 10-NOVEMBER-2021

CERTIFICATE

This is to certify that the project titled "**Predicting Road Accidents and Risks Using Machine Learning**" is the bona fide work carried out by **Anmol Bhimrajka (20030142007) and Abhishek Dhankar (20030142002)** , students of M.Sc. Computer Application of Symbiosis Institute of Computer Studies and Research, Pune India, affiliated to Symbiosis International (Deemed University) during the academic year 2020-21, in partial fulfilment of the requirements for the award of the degree of **M.Sc. Computer Application.**

Dr. Priti Kulkarni

Signature of the Guide

Place: PUNE

Date: 10-NOVEMBER-2021

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

We would like to express our sincere gratitude and indebtedness to our project guide, **Dr. Priti Kulkarni**, who has supported us throughout our project with patience and knowledge.

We are also extremely thankful to our Course Coordinator **Dr. Priti Kulkarni** for valuable suggestions and interest throughout the course of this project.

Finally, we would like to take this opportunity to thank our families for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Abhishek Dhankar

20030142002

Anmol Bhimrajka

20030142007

ABSTRACT

Aside from health-related issues, in recent years, road accidents have become a global issue, ranked worldwide as the ninth leading cause of death. As stated by the World Health Organization, road traffic injuries killed an estimated 1.35 million people worldwide every year. That is, every 25 seconds, a person is killed. Allowing its citizens to die in traffic accidents is completely unacceptable and heart-breaking. This necessitates an examination of road accidents and the factors that influence them, as well as the development of a prediction model for road accident to reduce probability of occurrence.

This report was conducted to undergo a more in-depth analysis of Road accidents in order to determine the severity of accidents using machine learning approaches. we also identify the significant factors that have a clear impact on road accidents and make some beneficial recommendations on the issue. It also provides indicative results for government to take constructive measures to reduce accident impression and upgrade traffic safety, by acknowledging above mentioned key factors. Analysis can be done by, using classification algorithm Decision Tree, random forest and logistics regression to classify the severity of accidents into fatal, serious and Slight.

We used Python, Scikit-Learn, NumPy, Matplotlib, and other Machine Learning methods. The cloud platforms used were Google Collab. The OpenWeatherMap Api is used to obtain weather and light conditions at a certain time and place based on the user's location. We developed a web app that displays user input and output and sends a warning to the concerned authority so they can take preventative measures. On Google colab, the model is learned and tested.

This model will be useful in traffic planning and management, and it will help us reduce the number of road accidents in the future.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENT

Contents

CHAPTER 1

INTRODUCTION

As per the death figures reported by World Health Organization (WHO), the number of road deaths that occur each year around the world is worrying. Every year, in road crash 1.2 million individuals are killed, as well as around 50 million individuals are harmed. Every day, approximately 137,000 are wounded as well as 3,300 individuals lose their life. The widespread incidence of road collision has a direct relationship with property and human life threat, resulting in direct economic losses of 43 billion dollars.

Prediction of traffic accident is known to be a major exploration area of road safety. The structural attributes of the lane, flow of the traffic, features of driver, as well as the road environment all play a role in the incident of traffic crashes. Many researches on menacing areas/accident prone area detection, investigation of injury gravity caused by road crash, as well as examination of road crash period is taken into consideration so that frequencies of road crash as well as the attributes of road crashes can be forecast. Some research concentrates specifically upon the causes of injuries. Weather and road lighting conditions are also part of list of factors to be considered.

A model is created [9] of probability that linked notable accident forerunner to variation in accident capable. Crash prediction model has been built [1] using the method of logistic regression implemented using the idea of matched-case control. There is no clear method available for traffic cops to predict which areas are vulnerable to accidents at any given instance. A significant part is played by prediction of road crashes in the non-segregated designing as well as traffic administration, since several nonlinear factors, such as people, cars, roads, and environment, play a role in traffic accidents. The conventional method of linear analyses cannot disclose the true situation due to noise emissions and a lack of data, causing the prediction result to be unsatisfactory. The standard back-propagation matrix has few flaws, like a local minimal, several repetition, and sluggish training, among others. There are flaws in the conventional back propagation network. Its accuracy is 7.8% lower than that of the proposed model.

1.1 Objective

Machine learning algorithms are computer programmes (math and logic) that adapt to new data and enhance their performance. They are capable of processing a large number of classification parameters and recognizing useful patterns.

Every day, a huge number of incidents occur on every highway. So, how about developing a model that will aid in the prevention of accidents? We may be capable of preventing an accident by developing an interactive model that can anticipate accident zones along a driving route while taking various parameters into account.

We set a clear objective for ourselves before starting this project**: we wanted to build an interactive road accident predictor that everybody can use.** We agreed that the best way to do this was to use a platform to deploy a qualified predictor.

The main objective of the road accident prediction system is to analyse previously occurring incidents in the region, which will enable us to identify the most accident-prone areas and provide the immediate assistance that they need. To make predictions based on factors such as weather, traffic, road design, and so on.

1.2 Problem Definition

There are a number of issues with current systems for preventing incidents in the cities. Many institutes and government websites make the database which is publicly accessible. Using the best-suited algorithm, the collected data will be processed, combined, and grouped together based on various parameters.

Many factors may play a key role in a road accident, but some only occur mostly on roads or highways.

In the year 2018, approximately 4,67,044 traffic collisions were registered in the states as well as union territories, due to which 4,69,418 individuals were wounded as well as 1,51,417 individuals lost their lives [8]. Among these individuals 64.4% died due to rash-driving.

Table 1: In 2018, the classification of Roads had the most mishaps led to injury and death of many individuals.

Category of Roads	Length as on 31.3.17		Accidents		Persons killed		Persons injured	
	Kms	% age share in total	Number	% age share in total	Number	% age share in total	Number	% age share in total
National HighWays	1,14,158	1.94	1,40,843	30.16	54,046	35.69	1,40,622	29.96
State Highway	1,75,036	2.97	1,17,570	25.17	40,580	26.80	1,21,579	25.90
Other roads	56,08,477	95.10	2,08,631	44.67	56,791	37.51	2,07,217	44.14
Total	**58,97,671**	**100**	**4,67,044**	**100**	**1,51,417**	**100**	**4,69,418**	**100**

Many authorities, especially government agencies, are experiencing difficulties identifying the factors that lead to accidents on critical roads or highways. Measures to prevent accidents can include speed reduction, road widening, speed control, or the construction of a road divider, among other items.

All of the problems described above demonstrate the importance of researching road crash contributors, specifically on the intersection of roads, that has a high frequency of accidents annually. The analysing method would be Machine learning , which is a high-performance tool for designing prediction models.

This study would suggest an effective accident prediction model that any government agency can use to assist the authorities to provide protection measures as well as potential effort tackle the problem of safety, specifically for concerned authorities as well as road users in future.

1.3 Existing System

There is no clear viable method for the involved authorities to determine which areas are vulnerable to road crashes at an instance.

1.4 Proposed System

An app which is fuelled by machine learning that predicts the severity of accidents based on current conditions.

Which will be trained using approximately 1.6 million accident records from 2009 to 2019.

The goal of such a model is to be able to predict which conditions are more likely to result in an accident and thus take preventive measures.

A message or an alert will be sent to the concerned authority to take preventive measures based on the predicted severity.

1.5 Organization of Report

To provide a platform i.e. a web app for taking user input at a particular time and predict severity of an accident at a location beforehand and take precaution.

● Literature Survey discusses about the literature survey of this project which includes an insight into the core part of our project along with the technologies used.
● The System Architecture part deals with the design of our proposed system. The Implementation part deals with the implementation of our system which discusses about the algorithms used in building our system.

CHAPTER 2

LITERATURE SURVEY

A literature survey is an important part of the software development process because it displays the various studies and research done in the area of your interest, as well as substantive observations and theoretical and methodological contributions to a specific subject. It is one of the most critical part of the project because it directs you in the direction of your study and aids in the establishment of project goals. The goal is to show the audience the explored thoughts and methods for the particular topic, as well as their strengths and weaknesses.

Literature Review

The prediction of road crashes plays a significant part in the non-segregated designing as well as traffic administration, since several nonlinear factors, such as people, cars, roads, and environment, play a role in traffic accidents. The conventional method of linear analyses cannot disclose the true situation due to noise emissions and a lack of data, causing the prediction result to be unsatisfactory [4]. The road crashes are known to cause significant danger to individuals' life as well as the protection of the living beings [9]. Since conventional BP (back propagation) networks have flaws like local minimums, too many iterations, and slow preparation, choose the enhanced back propagation matrix to forecast proposed by Levenberg Marquardt [4]. An enhanced back propagation NN, that has been used to analyse traffic accident predictions was proposed as a part of recent development [4]. A recent model prediction for traffic collisions (TAP-CNN) was developed with construction of a state matrix for describing the CNN model as well as traffic scenario by incorporating attributes affecting road crashes like flow of traffic, temperature and visibility [9]. The precision of the above-mentioned model is examined using samples [9]. The results obtained indicate that the TAP-CNN model is highly efficient at predicting traffic accidents than the conventional neural network model. It serves as a point of reference for traffic accident forecasting. To boost predictive modelling accuracy, transformative algorithm for cross validation was proposed for determining ideal folds within data-set [2]. For generating folds at

random, techniques of cross validation like 10 fold cross validation are implemented [2]. For testing as well as validation of ML models, a fold is mix of training-data set and test-data set splits [2]. Every fold will give the model a certain level of accuracy [2]. We know that Random Forest works really well with structured data, an analysis of the Random Forest group of ensemble techniques is presented [7]. Ensemble in a traditional random forest induction procedure, a rigid count of randomized decision trees are taken. Development of random forest with incorporation of trees in a highly dependent manner in comparison to the traditional random forest induction algorithms [7]. The most researched problem in Machine Learning is supervised classification. Decision Tree algorithms are a common option among the many algorithms used in such tasks because they are both robust and efficient to build. The investigation is done to find effect of hyperparameter-tuning on decision trees [3]. There were four different tuning methods investigated. According to the World Health Organization's death figures, the number of road deaths that occur each year around the world is worrying. Every year, 1.2 million individuals lose their life in road crashes as well 50 million individuals are wounded. Every day, approximately 3,300 individuals lose their life as well as 137,000 are wounded. The widespread incidence of road collisions has a direct relationship with property as well as human life threat, resulting in direct economic losses of 43 billion dollars. Among the highly relevant examined topics in road safety is traffic collision prediction. The structural attributes of the lane, flow of the traffic, features of driver, as well as the road environment all play a role in the incident of traffic crashes [4-9]. Many researches on menacing areas/accident prone area detection [2], investigation of injury gravity caused by road crash [7], as well as examination of road crash period [3] is taken into consideration so that frequencies of road crash as well as the attributes of road crashes can be forecast. Some researches take into consideration the causes of injuries. Findings are resurfaced regarding the connection of accident rates on non-urban roads, structured attributes, forecasting using hierarchical tree-based regression [5]. A model is created [9] of probability that linked notable accident forerunner to variation in accident capable. Crash prediction model has been built [1] using the method of logistic regression implemented using the idea of matched-case control. Theory of deep-learning describes content, pictures, as well as audios, and is commonly used in the fields of speech, content and audios, with NN method as a very effective tool of deep-learning in road crash forecasting. Deep learning, in comparison

to conventional learning structures, can prepare complicated non linear phenomena with use of hierarchical characteristics as well as distributed representation [1].

2.1 Prediction Factors

The information was obtained from the government's website, www.data.gov.uk. Accident data were collected by UK police forces using the Stats19 form. The data includes all types of automobile crashes. The dataset's columns are all in numerical format.

On the www.data.gov.uk website, there is a supporting document that explains each numerical category in the incident's dataset.

Below is the table showing Prediction Factors:

Table 2. Prediction Factors

Week days in numbers	1 -> Sun, 2 -> Mon, 3-> Tue, 4 -> Wed, 5 -> Thur, 6-> Fri, 7 -> Sat.
Latitude and Longitude	
Visibility status	Night, day, street light present or not.
Weather status	Snow, wind, fog and rain.
Vehicle type	Pedal cycle, Motorcycle, Car
Road surface conditions	Wet, snow, ice, flood.
Speed limit	60, 70 mph
Output	Accident Severity -> 1 for Fatal, 2 for Serious and 3 for Slight

Below are the tables explaining category and meaning of weather and visibility classification attributes:

Table 3. Weather Status

Representation	**Label**

1	No high winds and status fine
2	No high winds and raining
3	No high winds and snow fall
4	High winds and status fine
5	High winds and raining
6	High winds and snow fall
7	Foggy or mist condition
8	Any other
9	Any unknown
-1	No data present or over the provided range

Table 4. Visibility status

Representation	**Label**
1	If daylight
4	Lights on and dark condition
5	Lights off and dark conditions
6	Lights not present and dark conditions
7	Unknown lighting and dark condition
-1	No data present or over the provided range

Below is the table for Road Surface conditions:

Table 5. Conditions of Road

Representation	**Labels**
1	Conditions are dry
2	Conditions are damp or wet
3	Snow fall condition
4	Icy or frosty condition
5	Condition of more than 3cm floods
6	Spilled diesel or oil

7	Muddy conditions
-1	No data present or over the provided range

Below is the table for Driver gender:

Table 6. Gender

Representation	**Labels**
1	For male
2	For female
3	For unknown
-1	No data present or over the provided range

Below is the table for Vehicle types:

Table 7. Vehicle Types

Representation	**Labels**
1	In case of pedal-cycle
2	In case of motorcycle of 50cc or below
3	In case of motorcycle between 125cc and 50cc
4	In case of motorcycle above 125cc and below 500cc
5	In case of motorcycle above 500 cc
8	In case of privately hired cars and taxies
9	In case of personal cars
10	In case of minibus with capacity (8-16 passenger)
11	In case of buses or coaches with capacity (17 or more passengers)
16	In case of Horse riding

17	In case of vehicles for agricultural purposes
18	In case of tram.
19	In case of vans or Goods vehicles 3.5 tonnes mgw* or below
20	In case of goods vehicles above 3.5 tonnes and below 7.5 tonnes
21	In case of goods vehicle with capacity of 7.5 tonnes mgw* and above
22	In case of Scooters
23	In case of electric motorcycle
90	In case of other Vehicles
97	In case of unknown cc motorcycles
98	In case of goods vehicles with unknown weight
-1	No data present or over the provided range

* maximum gross weight (mgw)

2.2 Validation

To build a ML model, training data-set is required specifically for techniques such as supervised learning which includes classification and regression algorithms. In training data-set the data is labelled i.e., we have the target value along with the input features and the more accurate the training data-set the better the model may perform.

If in the process of data-training large quantity of data is furnished, a more reliable and efficient regression as well as classification model can be formed. Forming the model is followed by the process of validation in which a trained-model is examined using a test data-set. The testing and training data-sets are part of the same data-set. For precise

forecasts under environment of production it is important to properly examine the model using sufficient test data-set.

Regrettably, less quantity of data frequently pushes machine learning expounders to divide the data-set into two subsets of data, called test-data as well as train-data. When we do a validate/train/test split on the data-set, we are tuning the hyperparameters prior to model testing. A common proportion for ensuring sufficient training data will be 80-10-10. Because of bias, the precision of validation as well as training of singular train-test split model would decrease. In such scenario, bias indicates that information points in a single train-test split may be aggregated in such a fashion that one group stays in the training set while the other remains in the test set. A circumstance like this leads to bias in the train-test split, which lowers the predicted accuracy of a model.

By limiting the training data, we potentially lose essential patterns or trends in the data set, that raises inaccuracy due to bias. Cross validation utilizes several train-test splits, cross validation with N folds do that the data is split into n subsets. Holdout technique is repeated n times, with one of the n subsets serving like test set or validation set and other n-1 subsets forming a training set each time.

This significantly help to reduces bias in the training model.

2.3 Random forest and Decision Tree

Decision-trees are structures resembling flow-chart in which every interior node constitutes an attribute test, every individual branch reflects the output of the test as well as every individual leaf node constitutes a class-mark.

The classification regulations are represented by paths form root to leaf.

There are three nodes in a decision tree:

Decision nodes – typically represented by squares

Chance nodes – typically represented by circles

End nodes – typically represented by triangles.

From a collection of sample groups of a data population, Machine Learning can be utilised to develop high-performing categorization systems. Creating a unique classifying approach by integrating an ensemble of classifiers, is an efficient way to deal with this type of problem. Since the early 1990s, studies have shown that certain combination concepts, like Bagging [2], Boosting [9], Random-Subspaces [7], and as per recent studies, Random-Forests [3] are specifically impactful. In order to maximise the ensemble's generalization result, aggregating classifiers optimally depends on the potential to account for synergy between individual classifiers. The diversity property is widely used to describe this ability. While there is no universally accepted definition of diversity [6,] it is widely acknowledged as one of the most critical characteristics for improving generalisation efficiency in a ensemble of classifiers [1]. It can be described as an ensemble's individual classifiers' potential to agree on better forecasting while differing on forecasting errors. Ensemble methods from Random-Forest class. Rigid value of randomised decision trees are included to build an ensemble in a traditional Random-Forest induction procedure.

There are two major drawbacks to this type of algorithm, one is the trees must be rigid in numbers and second is that randomization theory reduces the interpretability and interpretation capabilities of decision tree classifiers. This type of procedure, in which trees are added to the ensemble in a progressive manner, does not guarantee that all of those trees can perform together successfully in the same committee.

2.4 Decision Tree Hyperparameter Tuning

The most researched problem in Machine Learning is supervised classification. Because they are both durable and efficient to design, Decision Tree algorithms are a popular choice among the many algorithms employed in such jobs. Furthermore, there is the benefit of generating understandable models with ideal precision standards in a variety of fields of application. These algorithms, such as many ML techniques, have some hyperparameters whose values have a direct impact on the output of the induced models. Since there are so many possible values for these hyper-parameters, many studies have used methods that are optimized to search a better set of results to create classifiers with better forecasting efficiency.

The hyper-parameters of the J48 Decision Tree algorithm were tweaked using four different tuning techniques. A total of 102 heterogeneous datasets were used to investigate the tuning influence on the induced models. The experimental results indicate that, despite a low average increase across all datasets, the improvement is statistically significant in the majority of cases.

There are many classification algorithms from which to choose because supervised classification is one of the most common ML jobs. Inclusive methods based on Decision-Trees have been frequently employed [9]. Decision-Trees are often employed because of their intelligible presence, which mirrors human reasoning [4]. They are expressed as classifiers by rules placed in a tree. As per several authors, Decision-Trees are one of the most popular data mining algorithms among scholars and scientists, highlighting their importance in the ML domain [2], [7]. Distortion tolerance (absent values, unbalanced classes), cheap computing cost, and the capacity to manage duplicated attributes are all advantages of DT inclusive algorithms over many other machine learning techniques [4]. C4.5 algorithm of Quinlan [3] as well as Breiman's classification and regression Tree algorithm [6] are two possibly the best Decision-Tree induction techniques in the academia. Hyper parameters of the Machine Learning algorithm have a direct impact on the forecasting model performance induced. Resulting in, a strong selection of these values has been studied in ML for several years. HYPER-PARAMETER TUNING has a major impact on the predictive efficiency of machine learning algorithms [8]. Choosing an appropriate configuration for an ML algorithm's HPs is normally done by trial and error. It takes some time to search for better set of number or values manually, depending on how long the ML algorithm takes to learn. Further resulting in, recent Hyper-Parameter for Machine-Learning algorithm research focuses on improving hyper-parameter adusting method. The Hyper-Parameter technique is seen as a black-box optimization problem, with the objective function linked to the model's predictive output induced by the algorithm.

CHAPTER 3

METHODOLOGY

We will be developing a web app for our model. It will consist of four components:

Front-End: Users input for the prediction factors will be taken and sent to the backend server.

Back-End: The model will be deployed here and the input data will be fed into the Machine Learning model.

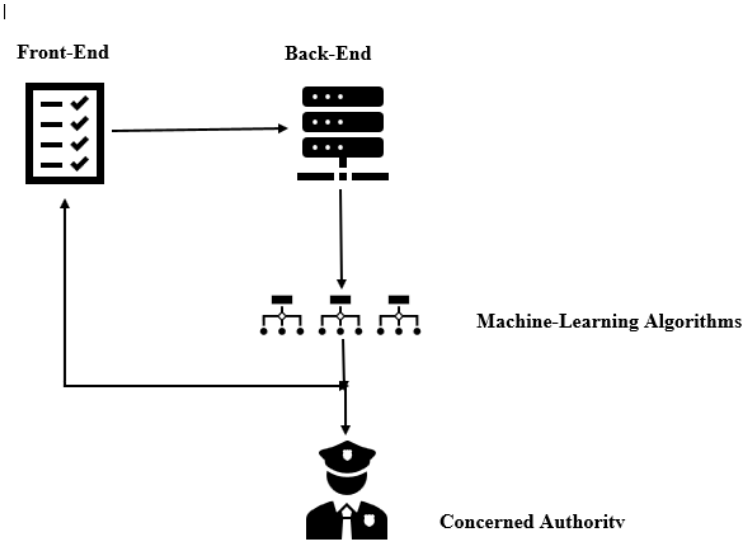
Machine Learning Model: We will be using a model that will give us the highest efficiency among decision tree, random forest, and logistic regression. After that, we can also perform hyperparameter tuning to tune our model for increased efficiency.

The model will run and predict the severity. The severity metrics would be 1= Fatal, 2= Serious, 3= Slight. The output will be sent back to the front-end and will be displayed to the user. The concerned authority will be informed about the severity and location of the user.

3.1 System Design

Describes the data flow in a diagrammatic representation.

Figure 1. System Model



3.2 Modules

1. **Front end:** We can use a Geolocation API that can take in the location of the user and through another API we can use the location and take the geographical conditions into consideration. User input will be taken for other parameters.
2. **Back end:** A server will be created and maintained. The input details will be fed to the model and severity will be predicted. The severity can be sent as a message or email to the concerned authorities to take preventive measures.
3. **Machine learning algorithm:** The basis of choosing an algorithm would depend on the accuracy that a model provides us. We would be trying classification algorithm decision tree, random forest and also logistic regression, parameter tuning would be performed when required to tune the model and increase the accuracy. Finally, the algorithm that gives us the highest accuracy in predicting the severity of the accident, will be chosen.

3.3 Technologies Used

3.3.1 Python

Python is a common high-level programming language for general-purpose applications. It was founded in 1991 by Guido van Rossum as well as developed by the Python Software Foundation. Its syntax makes it easier for programmers to code with lesser lines of code, and it was created with ease of code-readability in mind.

It is among many high level, interpreted language which can be used for a variety of tasks. Python has a dynamic style structure and memory management that is automated. Object-oriented, imperative, functional, and procedural programming paradigms are all supported. It also comes with a large standard library.

It is the world's most influential and fastest-growing programming language, with software developers, analysts, data scientists, and machine learning engineers all using it. It is used by sites like YouTube and Dropbox.

It supports OOP as well as functional and formal approaches. It gives high-level dynamic-runtime data types as well as facilitates dynamic type verification. It has a built-in trash collection system.

C, C++, COM, ActiveX, CORBA, and Java are all quickly integrated. To delimit blocks, Python uses whitespace indentation rather than curly brackets or keywords. Certain statements are followed by an increase in indentation, while the current block is ended by a decrease in indentation. As a result, the visual structure of the programme accurately represents the semantic structure of the programme.

3.3 NumPy

NumPy is the most important Python package for scientific computing. It has an object which is ndarray, tools for collaboration with various other languages, and handy fourier-transform, linear-algebra, as well as random-number abilities, among other things.

On top of its apparent scientific uses, NumPy could be used as a multi-dimensional store for generic-data. NumPy can now deal with a broad variety of databases with accuracy and convenience.

Since they are both interpreted, NumPy in Python provides functionality comparable to MATLAB, and they both empower users to construct quick programmes as long as the majority of operations are conducted on arrays/matrices instead of scalars. Matplotlib is a plotting software that mimics the plotting capabilities of MATLAB. NumPy is released under the BSD licence, which allows for easy reuse.

In the Python integration of the widely used computer-vision library OpenCV, NumPy arrays are often used to record and maintain data. Because visuals with various networks are easily stored as masking, slicing, 3-D arrays and indexing with other arrays are very effective ways to access specific pixels in visual-data.

3.3.3 Google Collab

Colaboratory (Collab) is Google cloud based Jupyter notebook ecosystem. Colaboratory started as part of Project Jupyter, but Google acquired over development over time.

The Python 2 and Python 3 kernels are supported by Colaboratory, but not the other Jupyter kernels like Julia and R. Jupyter-Project is a non-profit dedicated for "developing open-source applications, open-standards, and services for interactive computing in hundreds of programming languages." Fernando Pérez in 2014, derivated Jupyter-Project through IPython, and it now supports execution environments in over a dozen languages. Name Project-Jupyter is a play on Jupyter's three major languages that is python, R and Julia, and a tribute to Galileo's notebooks chronicling Jupiter's moon discoveries.

Jupyter-Notebook, Jupyter Hub, as well as Jupyterlab, the next-generation version of Jupyter Notebook, are all collaborative computing products developed and funded by Project Jupyter. Depending on the background, the word "notebook" may refer to a variety of things, including the Jupyter web application, Jupyter Python web server, or Jupyter document format. The Jupyter-Notebook record is a JSON file including an ordered sequence of output or input cells which can retain rich-media, mathematics, text, code and plots, as well as commonly ends in .ipynb. It's utilised to perform tasks that necessitate a large amount of resources.

3.3.4 Scikit Learn

Scikit-learn is a machine-learning library written in Python that may be downloaded. It is meant to interact with the Python numerical and scientific libraries NumPy as well as SciPy, and features support Density-Based Spatial Clustering, random-forest, vector-machines, k means, and gradient-boosting , among other classification, regression, and clustering methods.

Built on NumPy, SciPy, and matplotlib Open source, commercially usable - BSD licence. Data mining as well as data processing solutions that are both non-complex and effective Everyone has accessibility to it, and it can be used in a variety of situations. Matplotlib, Scipy as well as numerical-python were used to create this.

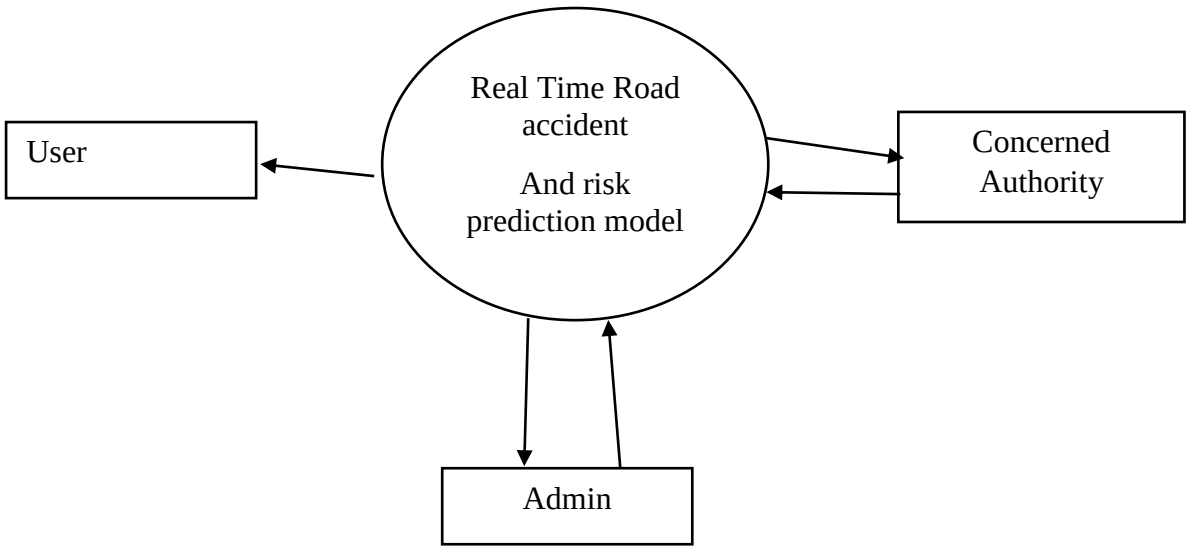
3.4 Diagrammatic Representation

3.4.1 Data Flow Diagram

A data flow diagram (DFD) has the ability to show flow of data/information within a process/system. It uses preset symbols such arrows, circles as well as rectangles, and short text labels, to depict data pathways, I/O, as well as storage sites among every target/destination. The flow-charts of data can be multi-level process overview, in-depth diagrams that go deeper into how data is processed and can also be simple hard-drawn overviews. They can also be utilized to investigate an existing system or to design a new one. A DFD, like the greatest diagrams as well as charts, can graphically represents stuff that are hard to express in words, and it can be utilised by both non-technical as well as technical people, ranging from CEO's to programmers. Due to which presently the DFDs are very well-known. While they are helpful for presenting systems and software data flow, as well as they are not very suitable in today's world for real time, database oriented, dynamic system as well as software in today's world.

3.4.1.1 DFD Level 0

Figure 2. DFD Level 0

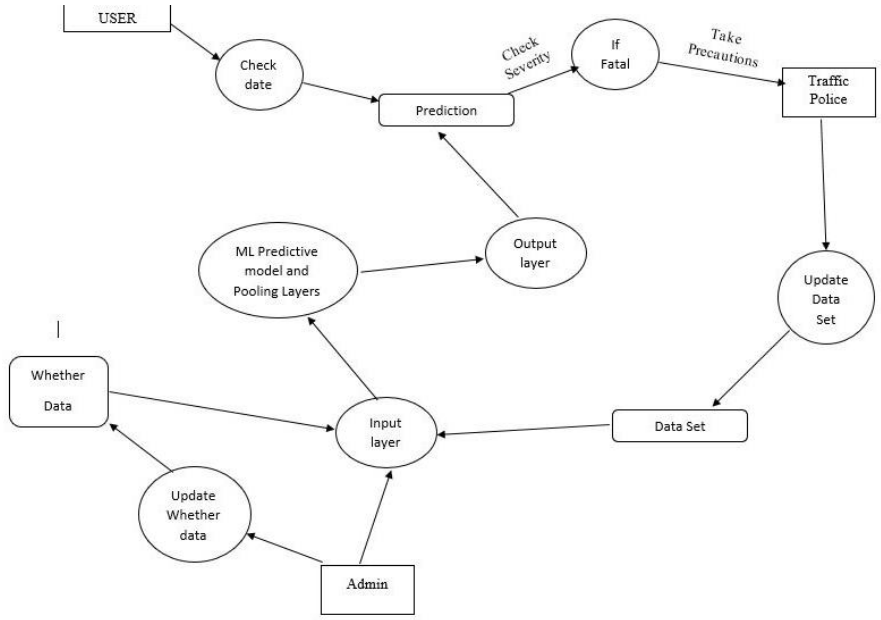


- Admin: is responsible for building the ML model and maintaining it.
- User : the person who views the output.

- Traffic police : they take respective action according to the output predicted by the ML model.

3.4.1.2 DFD level 1

Figure 3. DFD level 1



- The ML model is further divided into 3 layer
    - Input layer
    - Convolution layer or Pooling layer
    - Output layer
- If the output predicted is severity FATAL , which means that there is high probability for an accident to occur, so an alert is send to the traffic police to take respective action.

3.4.2 UML Diagrams

The Unified Modeling Language (UML) is a visual representation language for the architecture, design, and implementation of complex software systems. It is defined by the object management group. The building of a model lies at the heart of object-oriented problem solving. The model abstracts the key features of the underlying

problem from its typically complex real-world context. The scope of the project UML is a model-based language for describing, visualising, building, and documenting artefacts.

UML provides the following diagrams to represent the software process:

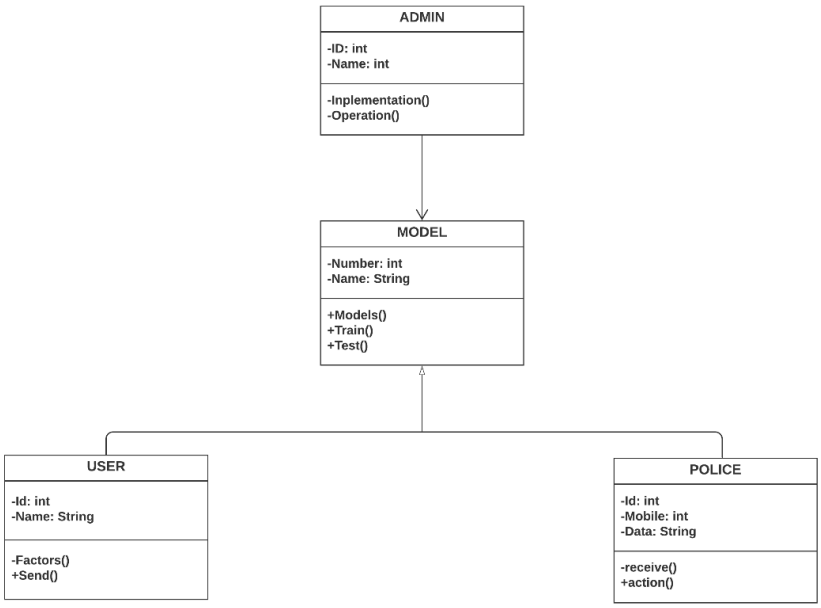
- Class Diagram
- Use Case diagram
- Sequence diagram

3.4.2.1 Class Diagram

A class diagram is a type of structural analysis diagram in the UML that represents the system's structure by presenting the software's classes, properties, actions (or methods), and interactions among objects.

It's a type of static structural graph that represents how a system's classes, characteristics, actions, as well as object connections are connected. It's utilised both for the broad modelling technique of the app's design as well as the detailed modelling of the designs' conversion in to the programming code. Class diagrams could also be used to depict data. The classes in a class diagram represent both of the application's primary characteristics and the classes which will be coded.

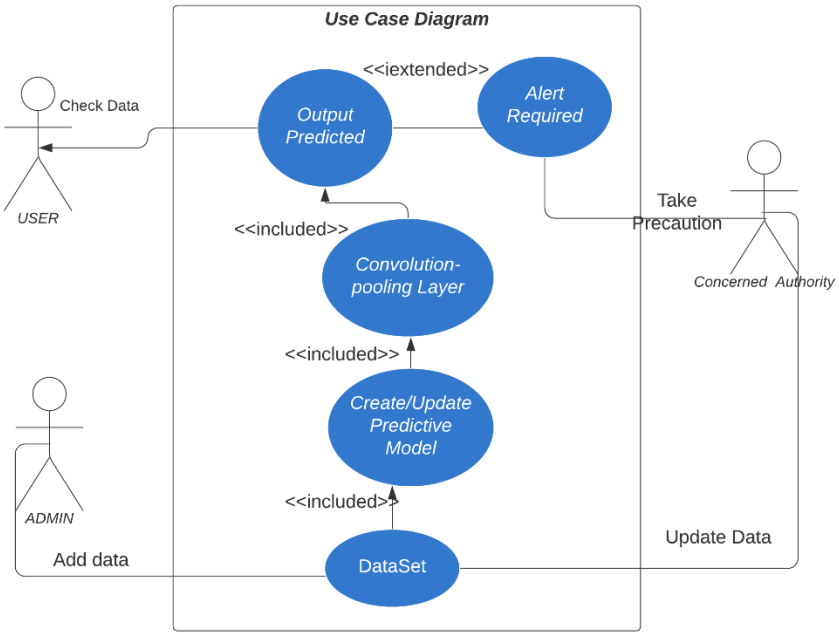
Figure 4. Class Diagram



3.4.2.2 Use Case Diagram

A use case diagram displays a user's involvement with the system at its most basic level by displaying the relationship between the user and the different use cases in which the user is involved. A use case diagram may be used to show the many types of users as well as the numerous use cases for a system, and it is typically accompanied by additional diagrams. The use cases can be represented by circle or ellipse.

Figure 5. Use-Case Diagram

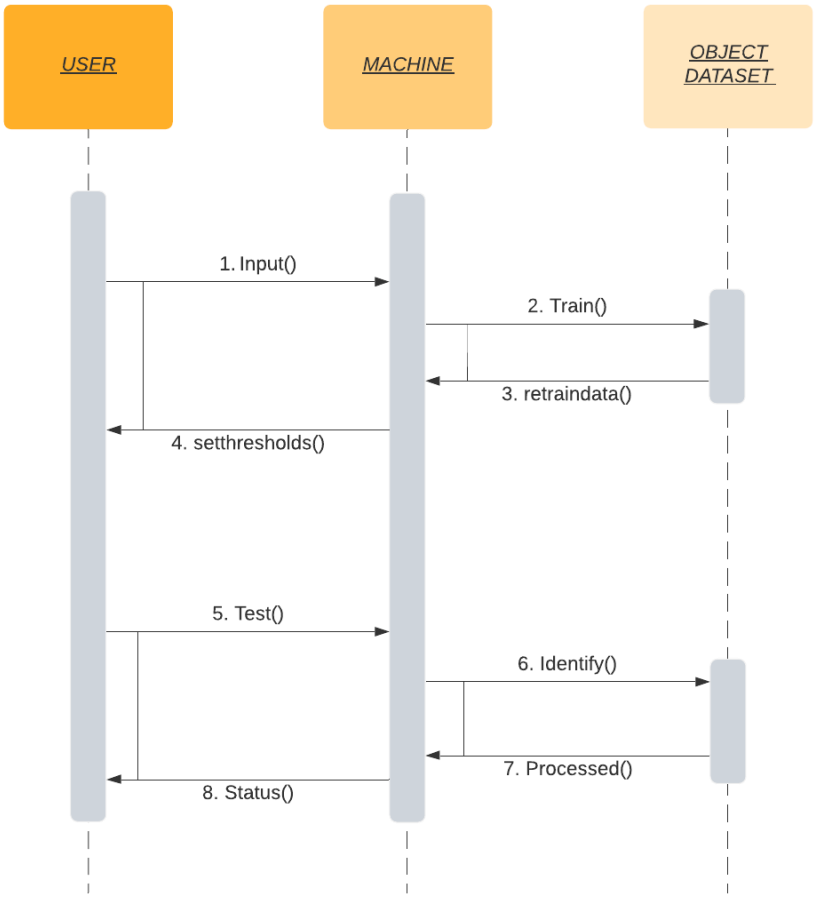


3.4.2.3 Sequence Diagram

Sequence-Diagram represents item interactions in serial order. It illustrates the classes and objects corresponding to a scenario, and the series of sent messages between them for carrying out the functionality with respect to the scenario. Sequence diagrams are frequently associated with use case realisations in the system's logical view. Sequence diagrams are sometimes known as event diagrams/event scenarios. Various procedures or items that exist at the same time are depicted as parallel vertical-lines, and the messages transferred between them are depicted as horizontal-arrows, in

manner in which they occur, in sequence diagram. This enables for graphical specification of simple runtime scenarios.

Figure 6. Sequence Diagram



3.5 Implementation of Proposed System

There are four important steps:

1. Pre-processing
2. Training
3. Testing
4. Web App Integration

3.5.1 Data Importing

To analyse the data, we need to import three files. This information is divided into three files: accidents, casualties, and vehicles. However, we have one more file that contains generic data on traffic counts from 2000 to 2015. For the machine learning aspect, we can use generic traffic statistics data.

- Importing of packages needed is done.
- CSV files Accidents.csv Casualties.csv Vehicles.csv
- Using pandas to import data into data frame
- accident.head() views top 5 rows of data frame

```
import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
# from mpl_toolkits.basemap import Basemap
from sklearn.model_selection import TimeSeriesSplit
plt.style.use('ggplot')
%config InlineBackend.figure_format = 'retina'
import warnings
warnings.filterwarnings('ignore')
```

Fig 3.1 Importing

```
accidents = pd.read_csv('accidents.csv',index_col='Accident_Index')
vehicles= pd.read_csv('vehicles.csv', error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
casualties = pd.read_csv('casualties.csv', error_bad_lines=False, index_col='Accident_Index', warn_bad_lines=False)
print('Loaded')
```

```
Loaded
```

Fig 3.2 Importing

```
print("accidents")
print("size=",accidents.size)
print(accidents.shape)
accidents.head()
```

```
accidents
size= 55200243
(1780653, 31)
```

Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles	Number_of_Casu
200501BS00001	525680.0	178240.0	-0.191170	51.489096	1	2	1	
200501BS00002	524170.0	181650.0	-0.211708	51.520075	1	3	1	
200501BS00003	524520.0	182240.0	-0.206458	51.525301	1	3	2	
200501BS00004	526900.0	177530.0	-0.173862	51.482442	1	3	1	
200501BS00005	528060.0	179040.0	-0.156618	51.495752	1	3	1	

Fig 3.3 Importing

3.5.2 Pre-processing of Data

Data Cleaning

We find noisy, irrelevant data here. We may also see which aspects are more essential by visualising them.

Identifying Missing Values

There are two types of missing values in this dataset: '-1' and 'Nan'. We'll look into each column that has a total of missing values. Because the dataset is large enough to do analysis, we will not be imputing any mean or median values.

```
accidents.drop(['Location_Easting_OSGR', 'Location_Northing_OSGR','LSOA_of_Accident_Location','

#combining two columns
accidents['Date_time'] = accidents['Date'] +' '+ accidents['Time']

for col in accidents.columns:
    accidents = (accidents[accidents[col]!=-1])
    #print(col ,' ' , x)
for col in casualties.columns:
    casualties = (casualties[casualties[col]!=-1])

accidents['Date_time'] = pd.to_datetime(accidents.Date_time)
accidents.drop(['Date','Time'],axis =1 , inplace=True)
accidents.dropna(inplace=True)
```

Using join method to combine accidents and vehicles files as they have the same primary key Accident_Index.

```
accidents = accidents.join(vehicles, how='outer')
```

Fig 3.4 join

Data Visualization

The first thing we can do is to find out about accidents time to get intuition and some driver's age who are involved in the accident.

- We can find out the number of accidents on the days of a week.
- We can find out about the accidents number using hours of the day.
- Finding out about the age of driver can tell us more about the accidents.

Accidents on Day of Week

The number of accidents on specific days of the week can be determined. As can be seen in this statistic from 2005 to 2015, Thursday has the largest number of accidents. We must keep in mind that the number of accidents may vary depending on the quantity of traffic on any given day.

```
plt.figure(figsize=(12,6))
accidents.Date_time.dt.dayofweek.hist(bins=7,rwidth=0.55,alpha=0.5, color= 'orange')
plt.title('Accidents on the day of a week' , fontsize= 30)
plt.grid(False)
plt.ylabel('Accident count' , fontsize = 20)
plt.xlabel('0 - Sunday ,  1 - Monday  ,2 - Tuesday , 3 - Wednesday , 4 - Thursday , 5 - Friday , 6 - Saturday' , fontsize = 13)
```

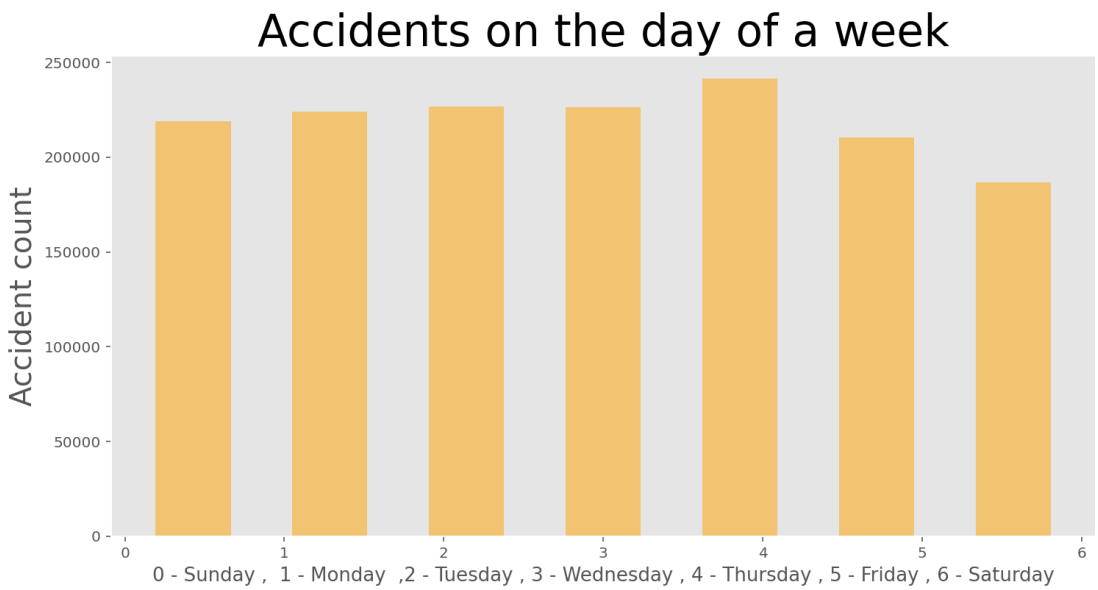


Fig 3.5 Accidents

Time of Accident

We discovered that the majority of the accidents occurred after noon. We can infer that the most traffic is flowing during this time of day, as people are leaving work.

```
plt.figure(figsize=(12,6))
accidents.Date_time.dt.hour.hist(rwidth=0.75,alpha =0.50, color= 'orange')
plt.title('Time of the day/night',fontsize= 30)
plt.grid(False)
plt.xlabel('Time 0-23 hours' , fontsize = 20)
plt.ylabel('Accident count' , fontsize = 15)
```

```
Text(0, 0.5, 'Accident count')
```

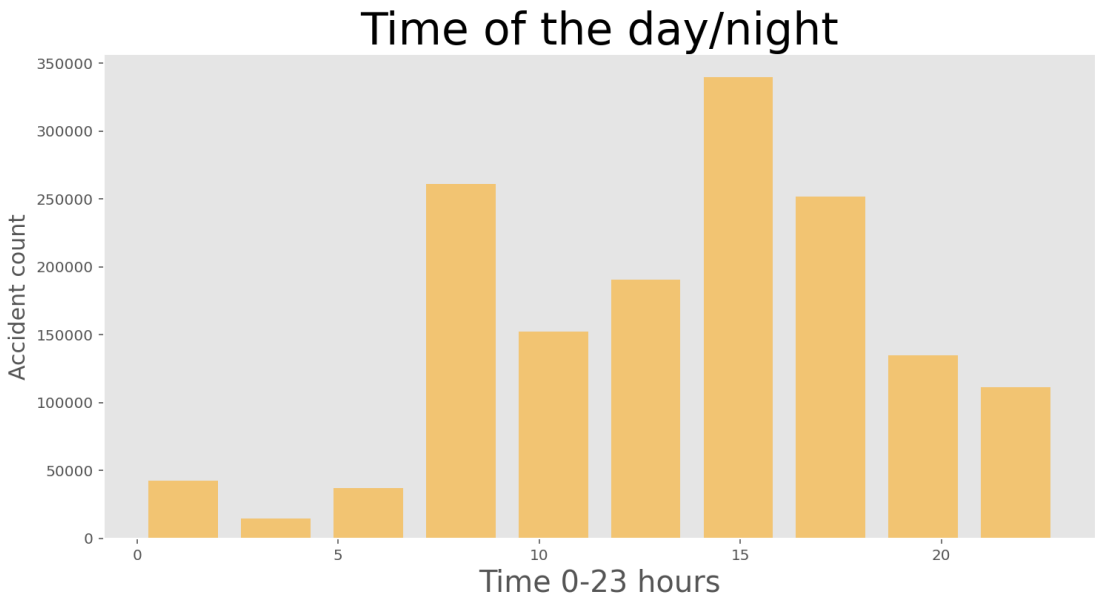


Fig 3.6 Time

Age Band of Casualties

The age bands in this dataset are divided into 11 distinct codes. We'll make the labels and give them to the plot as xticks to get a sense of how the bins are represented.

```
objects = ['0','0-5','6-10','11-15','16-20','21-25','26-35', '36-45', '46-55','56-65','66-75','75+']

plt.figure(figsize=(12,6))
casualties.Age_Band_of_Casualty.hist(bins = 11,alpha=0.5,rwidth=0.90, color= 'red',)
plt.title('Age of people involved in the accidents', fontsize = 25)
plt.grid(False)
y_pos = np.arange(len(objects))
plt.xticks(y_pos , objects)
plt.ylabel('Accident count' , fontsize = 15)
plt.xlabel('Age of Drivers', fontsize = 15,)
```

```
Text(0.5, 0, 'Age of Drivers')
```

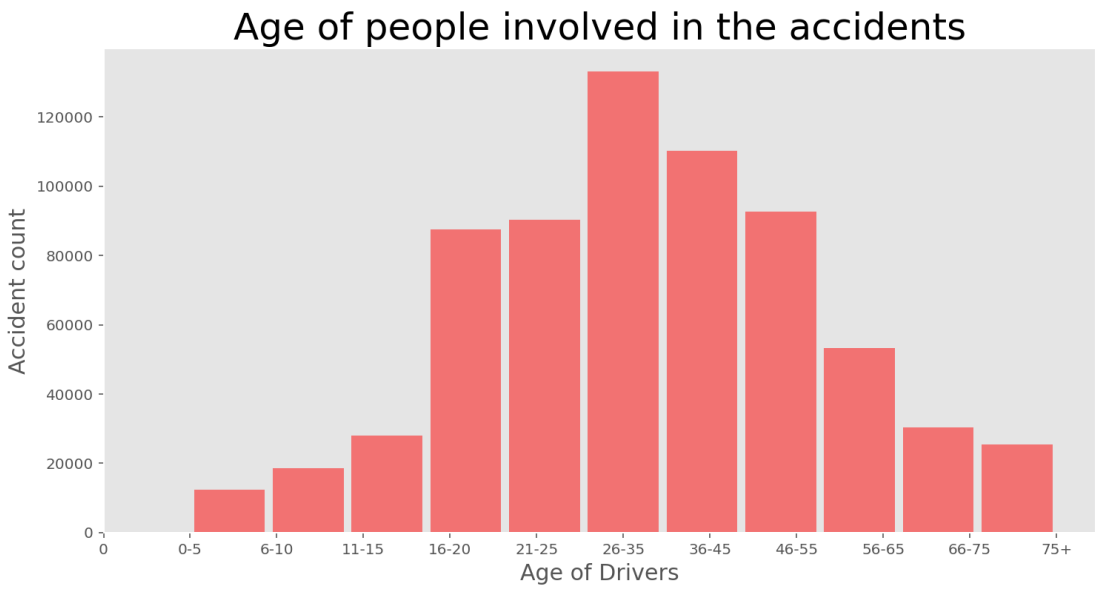


Fig 3.7 Age

This is a great aspect of the dataset. The majority of the drivers engaged in the accident are between the ages of 225 and 35. However, we don't know how many drivers aged 25 to 35 are on the road in comparison to other ages. Intuitively, I believe that the

number of drivers between the ages of 25 and 35 is higher than the number of drivers of other ages.

Co-relation between variables

Because our dataset is made up of numeric values. We can determine if there is a correlation between columns. As can be seen, there aren't many strong relationships between any of the variables. Only one major positive association exists between speed limit and urban or rural area.

```
corr =  accidents.corr()
plt.subplots(figsize=(20,9))
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3d00f8ad0>
```

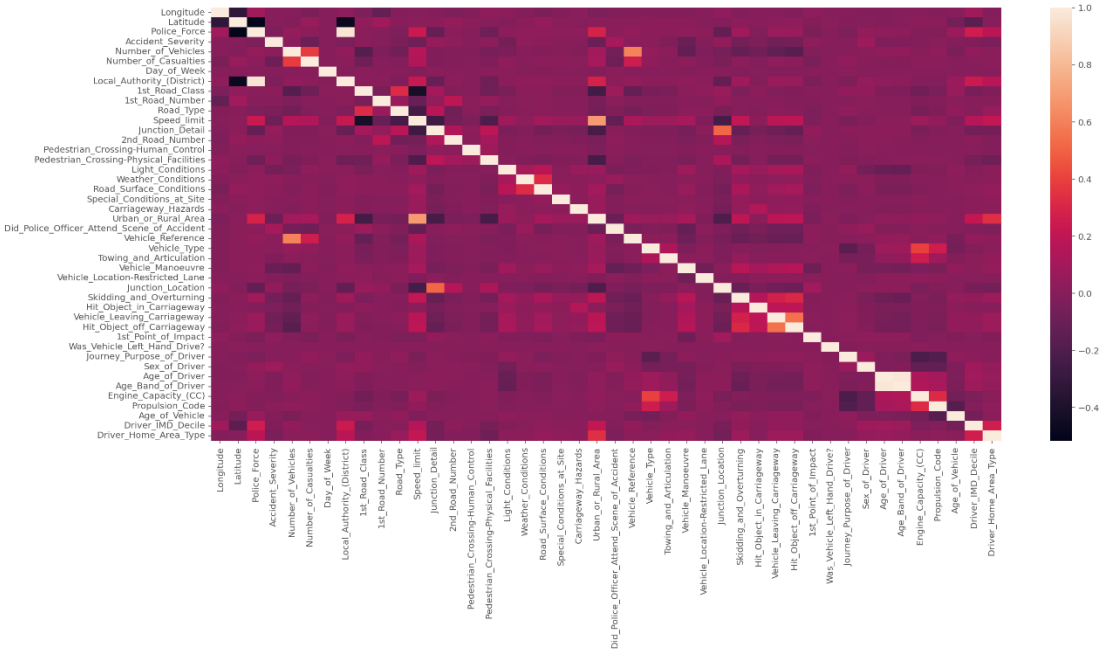


Fig 3.8 Correlation

Speed of Cars

```
speed_zone_accidents = accidents.loc[accidents['Speed_limit'].isin(['20' ,'30' ,'40' ,'50' ,'60' ,'70'])]
speed  = speed_zone_accidents.Speed_limit.value_counts()

explode = (0.0, 0.0, 0.0 , 0.0 ,0.0,0.0)
plt.figure(figsize=(10,8))
plt.pie(speed.values,  labels=None, autopct='%.1f',pctdistance=0.8, labeldistance=1.9 ,explode = explode, shadow=False,

plt.axis('equal')
plt.legend(speed.index, bbox_to_anchor=(1,0.7), loc="center right", fontsize=15, bbox_transform=plt.gcf().transFigure)
plt.figtext(.5,.9,'Accidents percentage in Speed Zone', fontsize=25, ha='center')
plt.show()
```

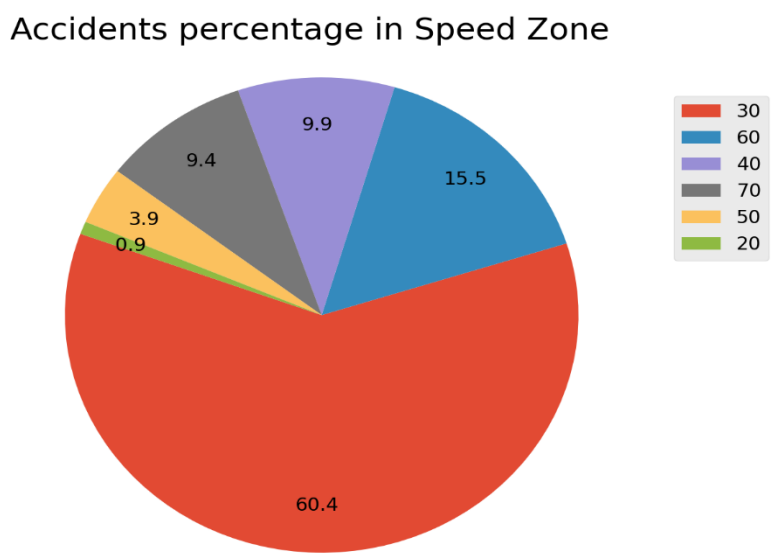


Fig 3.9 Speed

The majority of the collisions occurred on a route with a 30 mph speed limit. On highways and main roads, we expected to see more accidents. Stop signs, changing lanes, and driving into a parking lot are all possible causes of accidents.

Plotting accidents Location on Google Maps

Classifying locations based on severity

```
# accidents_2014 = accidents[accidents.Date_time.dt.year ==2014]
accidents01 = accidents[accidents.Accident_Severity == 1]
accidents02 = accidents[accidents.Accident_Severity == 2]
accidents03 = accidents[accidents.Accident_Severity == 3]
```

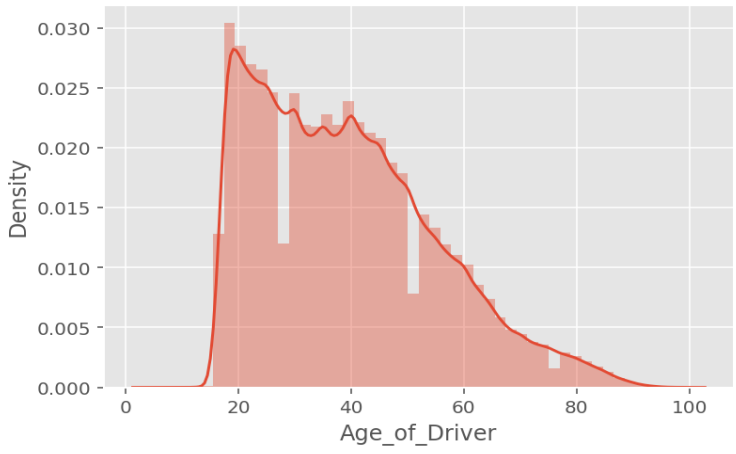
```
import gmaps
from ipywidgets.embed import embed_minimal_html
gmaps.configure(api_key="AIzaSyCto2LQ-KQh7sYQpzjaYNNa-1V2i5-ASF0")

fig = gmaps.figure(center=(53.0, 1.0), zoom_level=6)
heatmap_layer = gmaps.heatmap_layer(accidents01[["Latitude", "Longitude"]],
                                    max_intensity=30,point_radius=5)
heatmap_layer = gmaps.heatmap_layer(accidents02[["Latitude", "Longitude"]],
                                    max_intensity=5,point_radius=3)
heatmap_layer = gmaps.heatmap_layer(accidents03[["Latitude", "Longitude"]],
                                    max_intensity=1,point_radius=1)
fig = gmaps.figure()
fig.add_layer(heatmap_layer)
fig
```

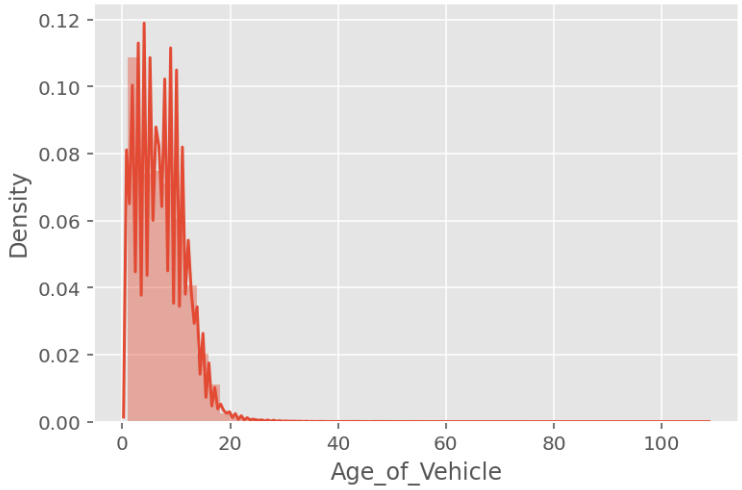
Fig 4.0 Heatmap

Normalize the Data

We will standardise a few columns so that our machine learning algorithms are not severely impacted. The driver's age ranges from 18 to 88 in the dataset, which we can normalise. Also, vehicle age ranges from 0 to 100, which can bias the efficiency of your machine learning model, so we'll normalise this prediction as well.

Before Normalization





After Normalization

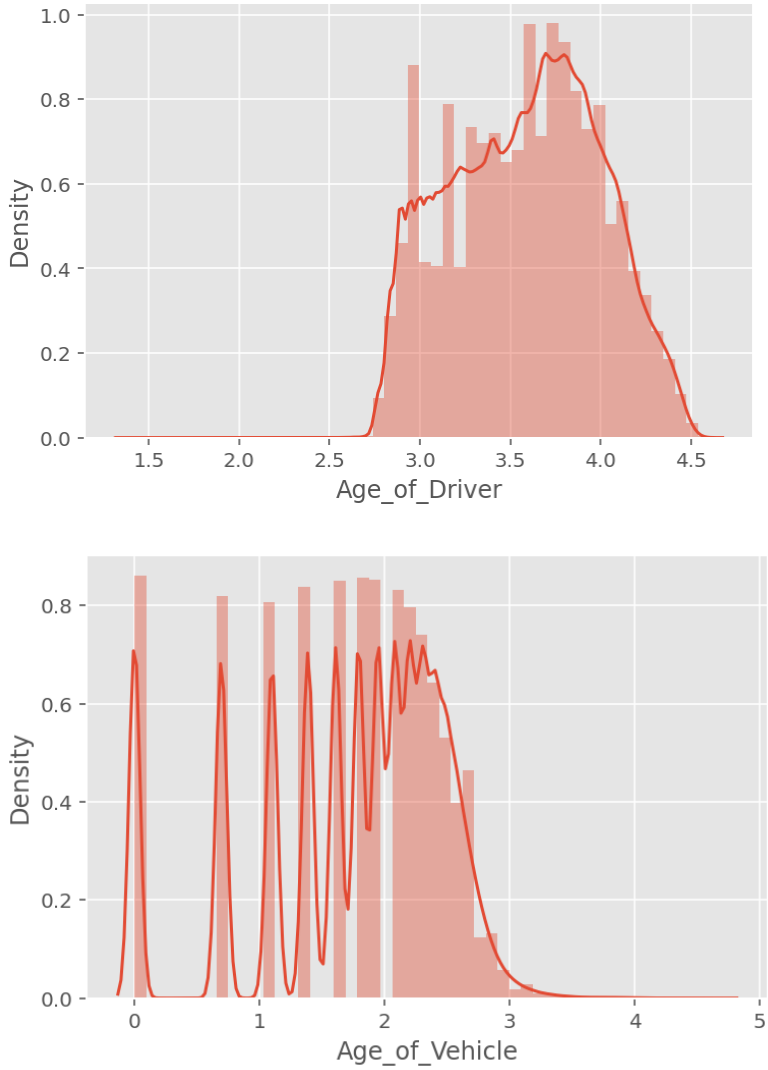


Fig 4.1 Normalization

3.5.3 Machine Learning

We'll look at several columns to see if we can forecast the severity of the Accident. We can provide some recommendations to law enforcement for looking into this and being prepared for the future once we can estimate the severity of the accident.

Following packages are being imported.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import log_loss
```

Fig 4.2 Packages

Splitting the data into training and test data

X is the input data and Y is the class label.

20% of the data is for testing and 80% for training.

```
accident_ml = accidents.drop('Accident_Severity' ,axis=1)
accident_ml = accident_ml[['Did_Police_Officer_Attend_Scene_of_Accident' , 'Age_of_Driver' ,'Vehicle_Type', 'Age_of_Vehicle','Eng
                           , 'Light_Conditions', 'Sex_of_Driver' ,'Speed_limit']]

accident_ml.head()

# Split the data into a training and test set.
X_train, X_test, y_train, y_test = train_test_split(accident_ml.values,
                                                    accidents['Accident_Severity'].values,test_size=0.20, random_state=99)
```

Algorithms and Techniques

Algorithms implemented with accuracy and confusion matrix

Logistic Regression

Accuracy 86.23

	precision	recall	f1-score	support
1	0.000000	0.000000	0.000000	4111
2	0.000000	0.000000	0.000000	38151
3	0.862323	0.999928	0.926042	264697
accuracy			0.862258	306959
macro avg	0.287441	0.333309	0.308681	306959
weighted avg	0.743599	0.862258	0.798545	306959

Predicted	1	3	All
Actual			
1	0	4111	4111
2	4	38147	38151
3	19	264678	264697
All	23	306936	306959

Fig 4.3 Accuracy: Logistics Regression

Decision Tree

Accuracy 75.3

	precision	recall	f1-score	support
1	0.036994	0.045974	0.040998	4111
2	0.159336	0.187728	0.172371	38151
3	0.871153	0.845495	0.858132	264697
accuracy			0.753035	306959
macro avg	0.355827	0.359732	0.357167	306959
weighted avg	0.771512	0.753035	0.761957	306959

Predicted	1	2	3	All
Actual				
1	189	890	3032	4111
2	920	7162	30069	38151
3	4000	36897	223800	264697
All	5109	44949	256901	306959

Fig 4.4 Accuracy: Decision Tree

Random Forest

Accuracy 84.58

	precision	recall	f1-score	support
1	0.062745	0.007784	0.013850	4111
2	0.230028	0.056303	0.090463	38151
3	0.866515	0.972625	0.916509	264697
accuracy			0.845817	306959
macro avg	0.386429	0.345571	0.340274	306959
weighted avg	0.776643	0.845817	0.801753	306959

Predicted	1	2	3	All
Actual				
1	32	317	3762	4111
2	105	2148	35898	38151
3	373	6873	257451	264697
All	510	9338	297111	306959

Fig 4.5 Accuracy: Random Forest

Hyper parameters tuning for Logistic Regression

Accuracy 86.23

	precision	recall	f1-score	support
1	0.000000	0.000000	0.000000	4111
2	0.000000	0.000000	0.000000	38151
3	0.862318	0.999977	0.926060	264697
accuracy			0.862301	306959
macro avg	0.287439	0.333326	0.308687	306959
weighted avg	0.743594	0.862301	0.798560	306959

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	6	264691	264697
All	6	306953	306959

Fig 4.6 Accuracy: Logistic Regression Hyper parameter

Hyper parameters tuning for Decision Tree

```
Accuracy 85.69
              precision    recall  f1-score   support

           1   0.063830  0.000730  0.001443      4111
           2   0.316813  0.043564  0.076595     38151
           3   0.866438  0.987450  0.922995    264697

    accuracy                       0.856922    306959
   macro avg   0.415694  0.343914  0.333678    306959
weighted avg   0.787378  0.856922  0.805456    306959
```

Predicted	1	2	3	All
Actual				
1	3	290	3818	4111
2	16	1662	36473	38151
3	28	3294	261375	264697
All	47	5246	301666	306959

Fig 4.4 Accuracy: Decision Tree Hyper parameter

Table of Accuracy

Algorithm	Accuracy
Decision tree	75.3
Random Forest	84.58
Logistics Regression	86.23
Decision Tree Hyper parameter tuning	85.69
Logistics Hyper parameter tuning	86.23

3.6 System Requirements

1. Windows 10.
2. Language: Python, javascript, HTML
3. IDE: Jupyter Notebook, Sublime Text, FLASK framework

3.6.1 Browsers

1. Chrome
2. Internet Explorer
3. Firefox
4. Safari
5. Edge

3.6.2 Hardware Requirements

1. System: Intel Xeon(4 vCpu) or i7
2. Hard Disk: 20 GB
3. Monitor: Virtual Machine: Standard D4s v3
4. Ram: 8 GB

CHAPTER 4

RESULT AND DISCUSSIONS

A web page has been developed for our model. It has been hosted on local machine and it can we easily accessed through this http://localhost/

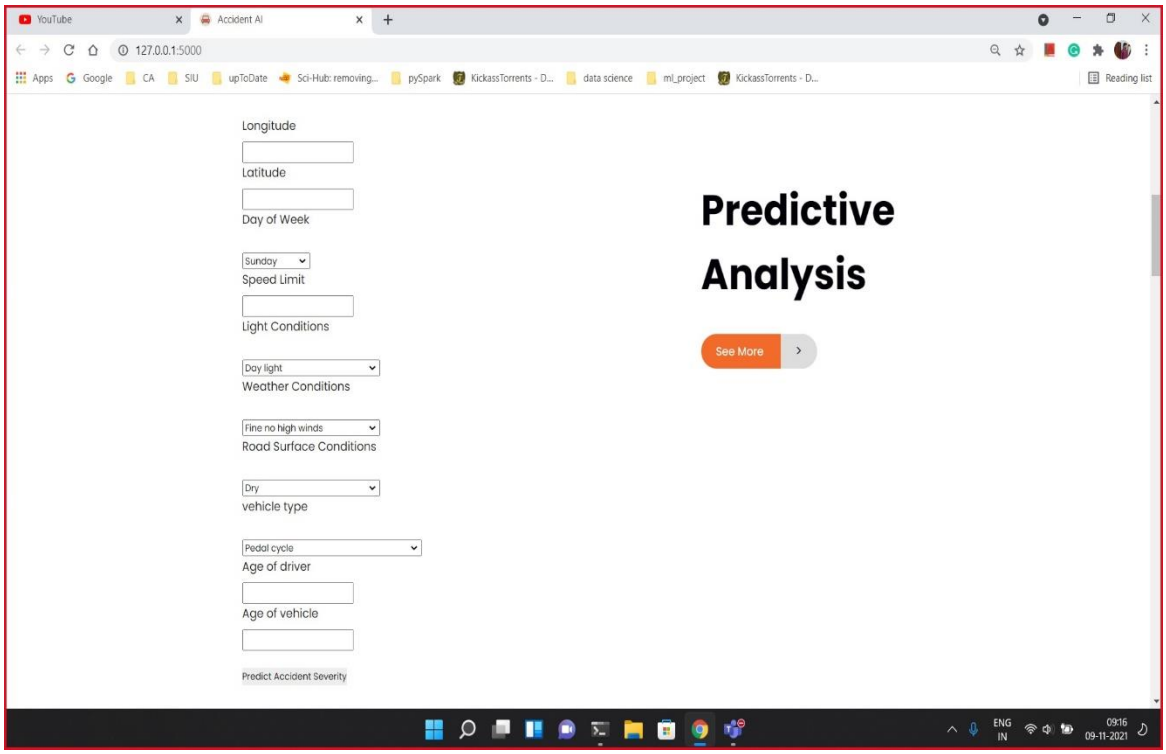
The Front-End which is the home page takes input for the prediction factors

- Location, latitude, Days of week, Speed limit, Light condition, Weather condition, Road surface condition, Vehicle type, Age of driver, Age of Vehicle is entered by user explicitly.

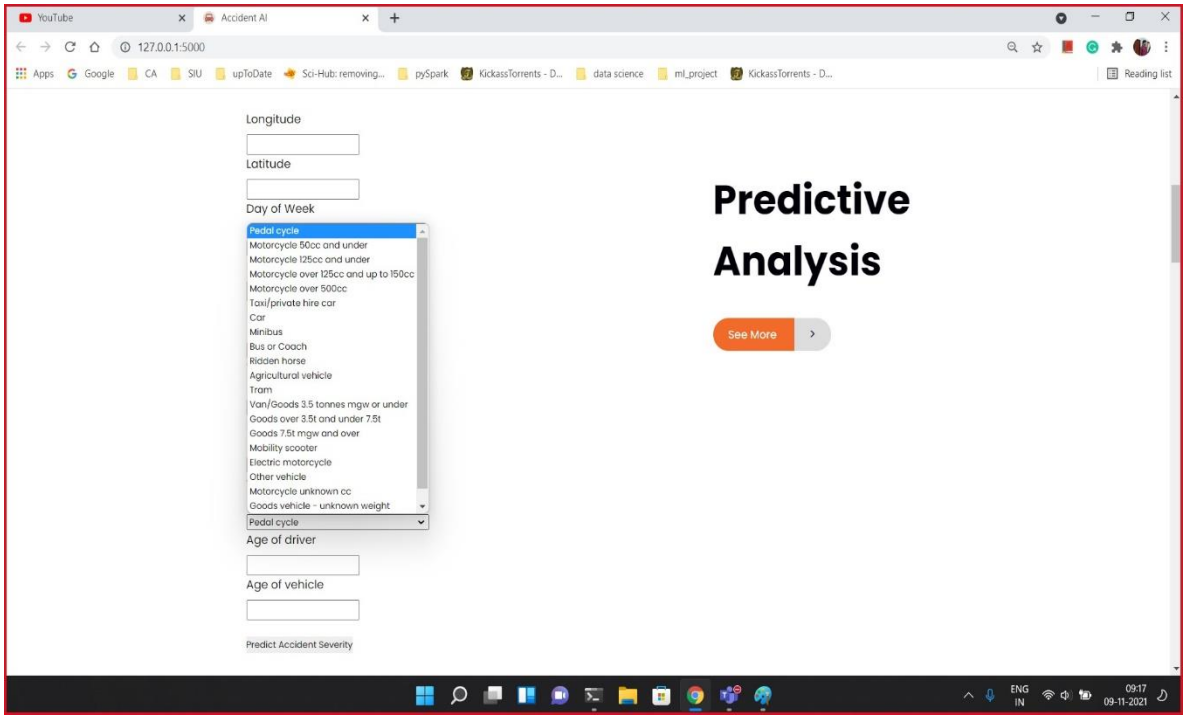
The back-end is where the model is deployed. The front-end input data is passed into the Machine Learning model. As our model, we selected the Random Forest algorithm, which had the greatest accuracy of 84.58 percent. The model is performed and the severity is predicted. The severity levels are 1 (fatal), 2 (serious), and 3 (slight).

The user sees the output, which is transmitted back to the front-end.

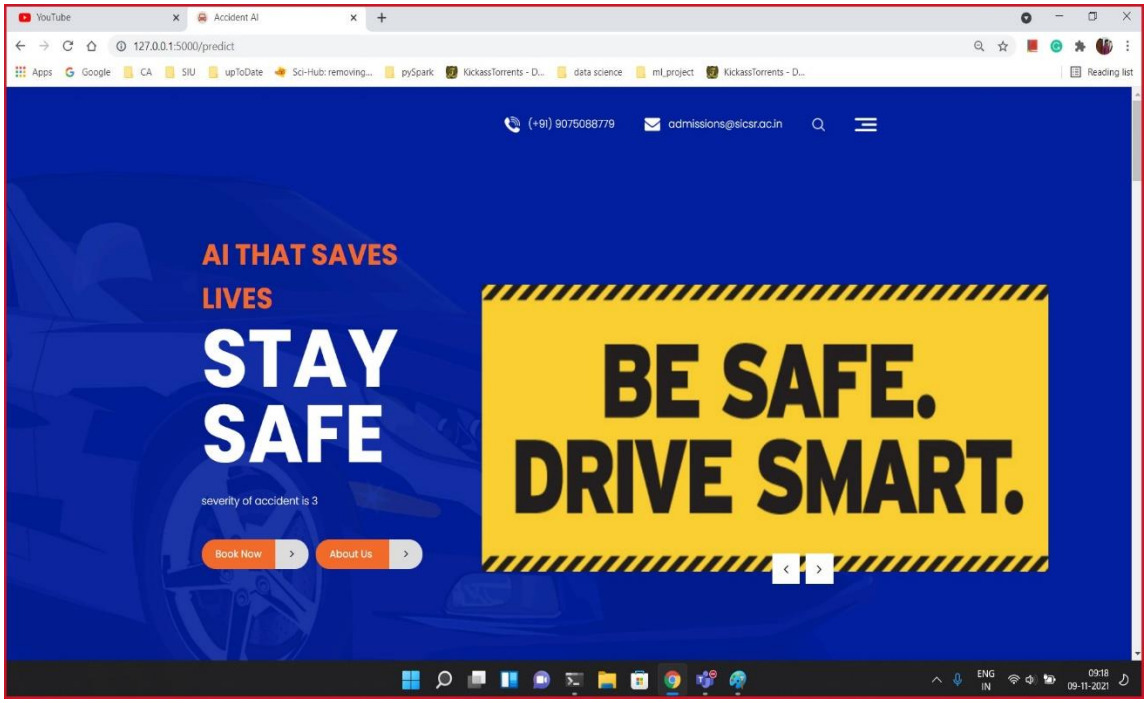
The police receive an SMS with the location coordinates and the severity of the accident so that they can take preventative steps at the scene.

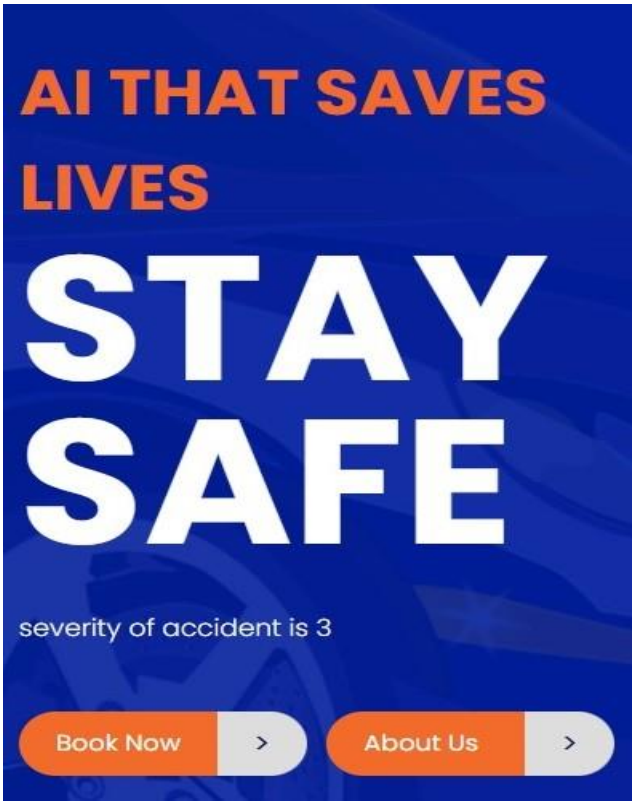


The above figure shows the home page of the web app.

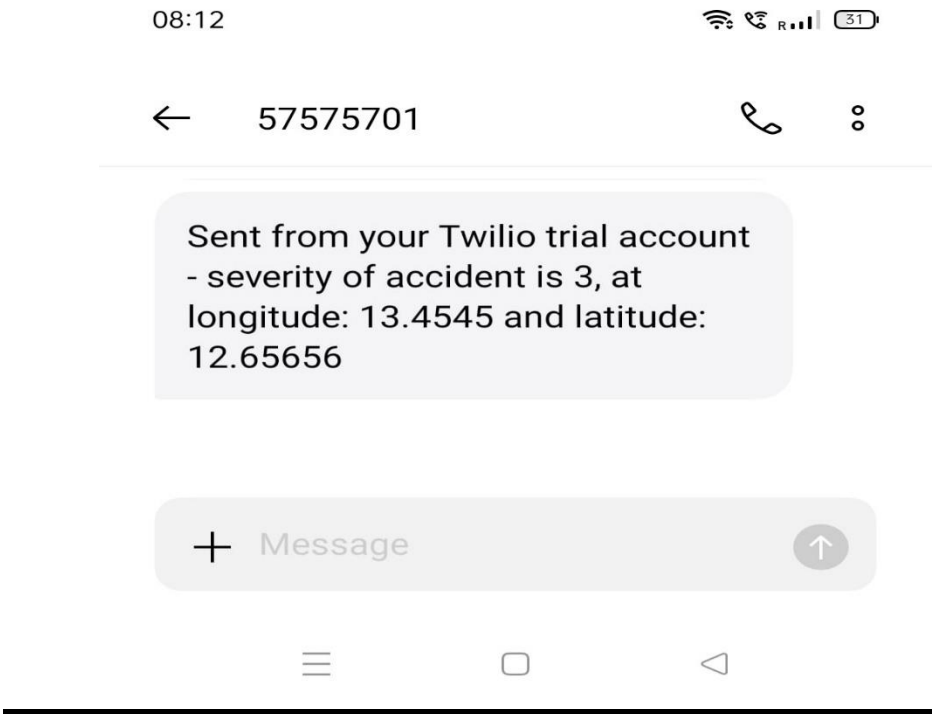


The input for parameters taken from users is shown in the diagram. The vehicle type, gender, and speed limit are among them.





When a user clicks Predict, the data are transmitted to the backend, where it is fed into Random Forest, our selected machine learning algorithm. The output is projected based on the severity of the event: 1- Fatal, 2- Severe, 3-Sligh



An SMS is sent to the police with location details and severity

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusions

The goal of this project is to predict the severity of an accident at a particular location with the help of Machine Learning classification algorithm. Machine learning algorithm will help us analyse the data and find a pattern within the data to predict the severity of accident with high accuracy. The best outcomes would be registered model which will be built using a ML classification algorithm. We will be testing decision-tree, random-forest, as well as logistic regression and algorithm which provides the best outcomes with respect to accuracy of severity prediction will be chosen. We'll put random forest, decision tree, and logistic regression to the test, and the method that produces the greatest results in terms of severity prediction accuracy will be picked.

5.2 Future Work

With increased resources, police can receive continuous predictions and alerts for each area at regular intervals, allowing them to take pre-emptive measures. The web app may be integrated with Google Maps, which the cops can track in real time. A completely functional web app for user and police interaction can be made available for real-time use. It can be used for Indian states or cities if the Indian government provides accurate accident data.

REFERENCES

[1] Abdel-Aty, Mohamed, Nizam Uddin, and Anurag Pande. 2005. "Split Models For Predicting Multivehicle Crashes During High-Speed And Low-Speed Operating Conditions On Freeways". Transportation Research Record: Journal Of The Transportation Research Board 1908 (1): 51-58. doi:10.1177/0361198105190800107.

[2] Bernard, Simon, Laurent Heutte, and Sebastien Adam. 2009. "On The Selection Of Decision Trees In Random Forests". 2009 International Joint Conference On Neural Networks. doi:10.1109/ijcnn.2009.5178693.

[3] Fu, Huilin, and Yucai Zhou. 2011. "The Traffic Accident Prediction Based On Neural Network". 2011 Second International Conference On Digital Manufacturing & Automation. doi:10.1109/icdma.2011.331.

[4] Gunasegaran, Thineswaran, and Yu-N Cheah. 2017. "Evolutionary Cross Validation". 2017 8Th International Conference On Information Technology (ICIT). doi:10.1109/icitech.2017.8079960.

[5] Karlaftis, Matthew G, and Ioannis Golias. 2002. "Effects Of Road Geometry And Traffic Volumes On Rural Roadway Accident Rates". Accident Analysis & Prevention 34 (3): 357-365. doi:10.1016/s0001-4575(01)00033-1.

[6] Lin, Lei, Qian Wang, and Adel W. Sadek. 2014. "Data Mining And Complex Network Algorithms For Traffic Accident Analysis". Transportation Research Record: Journal Of The Transportation Research Board 2460 (1): 128-136. doi:10.3141/2460-14.

[7] Mantovani, Rafael G., Tomas Horvath, Ricardo Cerri, Joaquin Vanschoren, and Andre C.P.L.F. de Carvalho. 2016. "Hyper-Parameter Tuning Of A Decision Tree Induction Algorithm". 2016 5Th Brazilian Conference On Intelligent Systems (BRACIS). doi:10.1109/bracis.2016.018.

[8] Ministry of Road Transport and Highways. 2018. "Road Accidents In India". Government of India. https://morth.nic.in/sites/default/files/Road_Accidednt.pdf.

[9] Wenqi, Lu, Luo Dongyu, and Yan Menghua. 2017. "A Model Of Traffic Accident Prediction Based On Convolutional Neural Network". 2017 2Nd IEEE International Conference On Intelligent Transportation Engineering (ICITE). doi:10.1109/icite.2017.8056908.

APPENDIX

Importing Data Set

Importing Data and cleaning

- We import three files to perform analysis on this data. This data is consist of three files that are accidents, casualities and vehicles. However, we have one more file which is general information about the traffic count for year 2000 to 2015. We can use general traffic information data for machine learning part.

```
import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
#from mpl_toolkits.basemap import Basemap
from sklearn.model_selection import TimeSeriesSplit
plt.style.use('ggplot')
%config InlineBackend.figure_format = 'retina'
import warnings
warnings.filterwarnings('ignore')
```

```
accidents = pd.read_csv('Accidents0515.csv',index_col='Accident_Index')
casualties=pd.read_csv('Casualties0515.csv' , error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
vehicles=pd.read_csv('Vehicles0515.csv', error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
#general_info = pd.read_csv('ukTrafficAADF.csv')
```

Checking Imported Data

```
print("accidents")
print("size=",accidents.size)
print(accidents.shape)
accidents.head()
```

```
accidents
size= 55200243
(1780653, 31)
```

Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles	Number_of_Casu
200501BS00001	525680.0	178240.0	-0.191170	51.489096	1	2	1	
200501BS00002	524170.0	181650.0	-0.211708	51.520075	1	3	1	
200501BS00003	524520.0	182240.0	-0.206458	51.525301	1	3	2	
200501BS00004	526900.0	177530.0	-0.173862	51.482442	1	3	1	
200501BS00005	528060.0	179040.0	-0.156618	51.495752	1	3	1	

```
print("vehicles")
print("size=",vehicles.size)
print(vehicles.shape)
vehicles.head()
```

```
vehicles
size= 63092925
(3004425, 21)
```

Accident_Index	Vehicle_Reference	Vehicle_Type	Towing_and_Articulation	Vehicle_Manoeuvre	Vehicle_Location-Restricted_Lane	Junction_Location	Skidding_and_Overturning
200501BS00001	1	9	0	18	0	0	0
200501BS00002	1	11	0	4	0	3	0
200501BS00003	1	11	0	17	0	0	0
200501BS00003	2	9	0	2	0	0	0
200501BS00004	1	9	0	18	0	0	0

```
print("casualties")
print("size=",casualties.size)
print(casualties.shape)
casualties.head()
```

```
casualties
size= 31034080
(2216720, 14)
```

Accident_Index	Vehicle_Reference	Casualty_Reference	Casualty_Class	Sex_of_Casualty	Age_of_Casualty	Age_Band_of_Casualty	Casualty_Severity	Pedestriar
200501BS00001	1	1	3	1	37	7	2	
200501BS00002	1	1	2	1	37	7	3	
200501BS00003	2	1	1	1	62	9	3	
200501BS00004	1	1	3	1	30	6	3	
200501BS00005	1	1	1	1	49	8	3	

Joining Of Tables

```
accidents = accidents.join(vehicles, how='outer')
print("done joining")
print(accidents.shape)
```

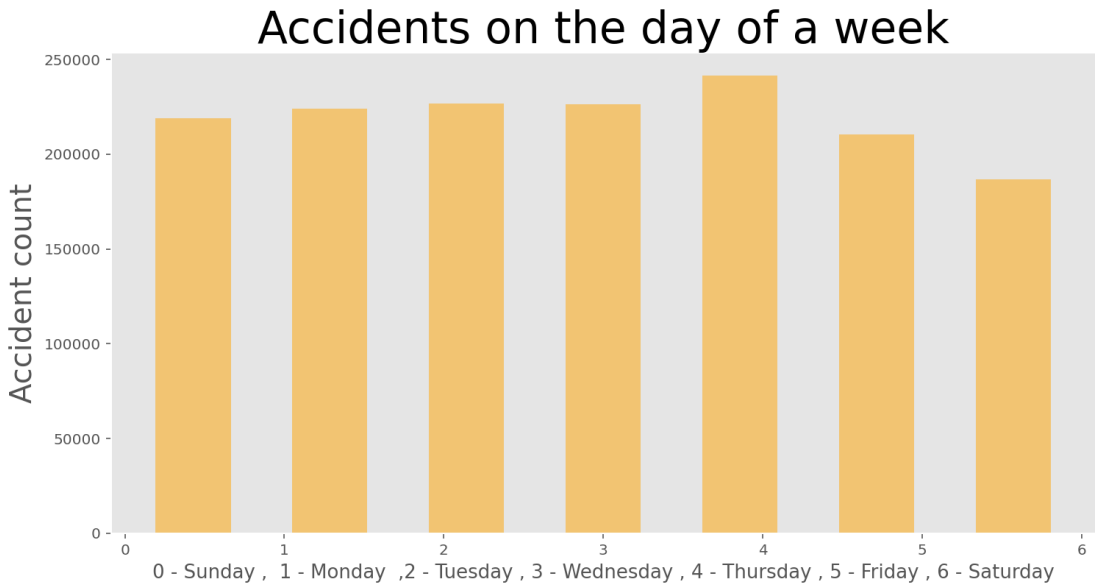
```
done joining
(3144481, 52)
```

Identifying Missing Values

```
accidents.drop(['Location_Easting_OSGR', 'Location_Northing_OSGR','LSOA_of_Accident_Location','

#combining two columns
accidents['Date_time'] = accidents['Date'] +' '+ accidents['Time']

for col in accidents.columns:
    accidents = (accidents[accidents[col]!=-1])
    #print(col ,' ' , x)
for col in casualties.columns:
    casualties = (casualties[casualties[col]!=-1])

accidents['Date_time'] = pd.to_datetime(accidents.Date_time)
accidents.drop(['Date','Time'],axis =1 , inplace=True)
accidents.dropna(inplace=True)
```

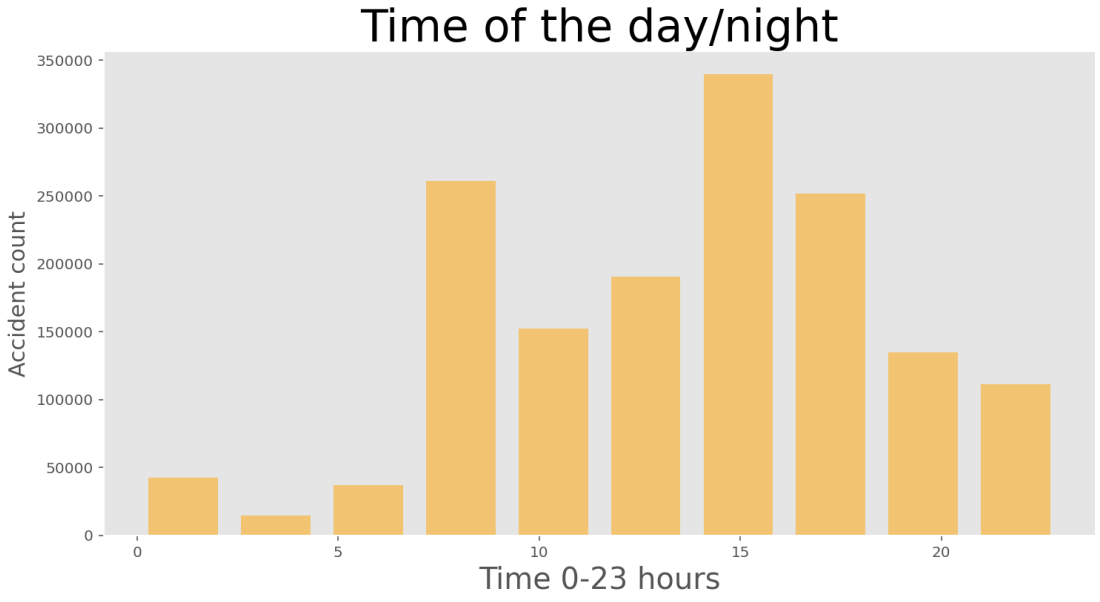
Data Visualization

```
plt.figure(figsize=(12,6))
accidents.Date_time.dt.dayofweek.hist(bins=7,rwidth=0.55,alpha=0.5, color= 'orange')
plt.title('Accidents on the day of a week' , fontsize= 30)
plt.grid(False)
plt.ylabel('Accident count' , fontsize = 20)
plt.xlabel('0 - Sunday ,  1 - Monday  ,2 - Tuesday , 3 - Wednesday , 4 - Thursday , 5 - Friday , 6 - Saturday' , fontsize = 13)
```



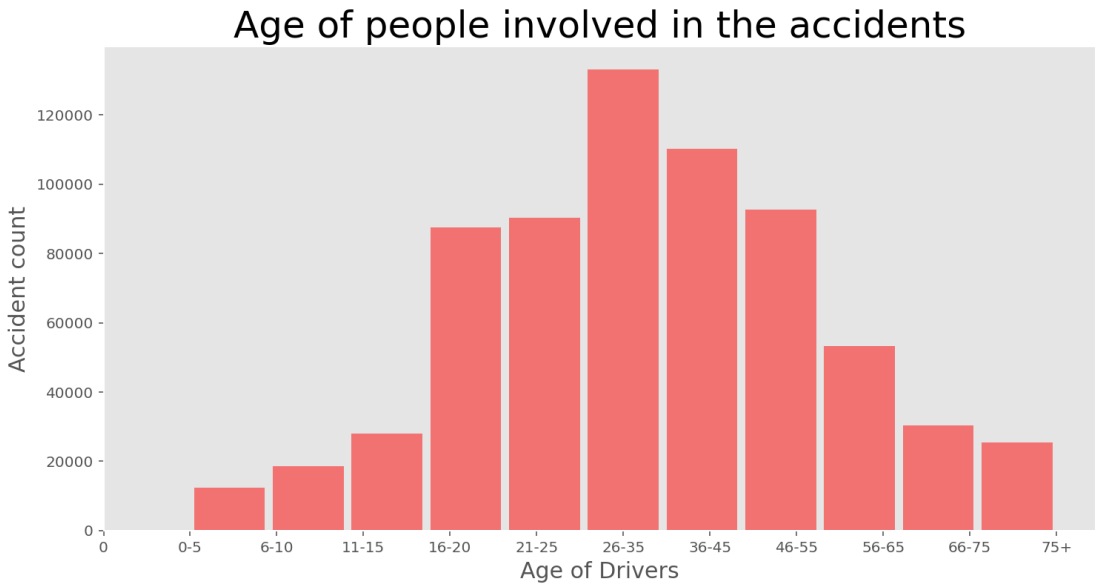

```
plt.figure(figsize=(12,6))
accidents.Date_time.dt.hour.hist(rwidth=0.75,alpha =0.50, color= 'orange')
plt.title('Time of the day/night',fontsize= 30)
plt.grid(False)
plt.xlabel('Time 0-23 hours' , fontsize = 20)
plt.ylabel('Accident count' , fontsize = 15)
```

```
Text(0, 0.5, 'Accident count')
```

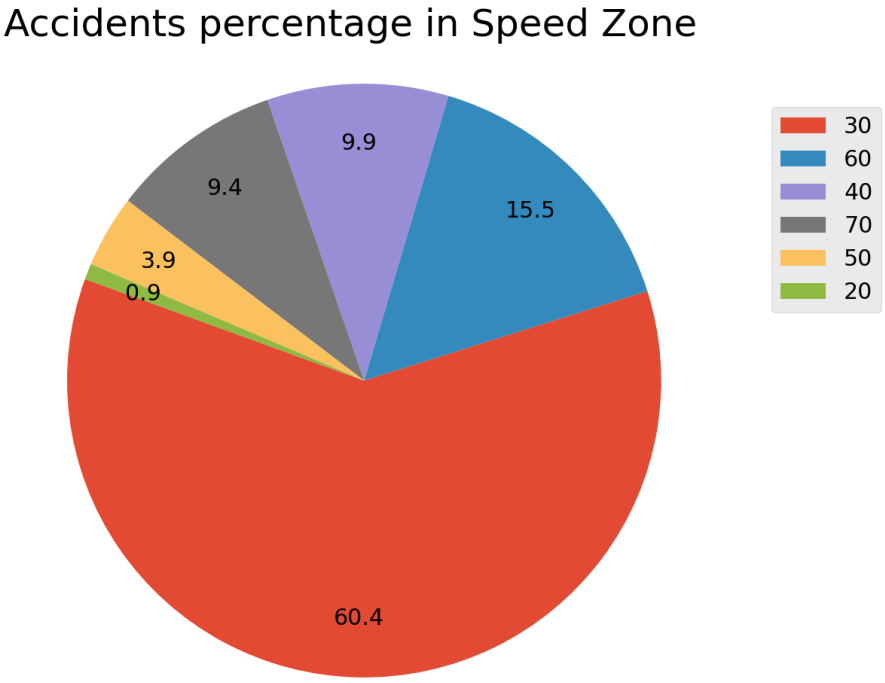


```
objects = ['0','0-5','6-10','11-15','16-20','21-25','26-35', '36-45', '46-55','56-65','66-75','75+']

plt.figure(figsize=(12,6))
casualties.Age_Band_of_Casualty.hist(bins = 11,alpha=0.5,rwidth=0.90, color= 'red',)
plt.title('Age of people involved in the accidents', fontsize = 25)
plt.grid(False)
y_pos = np.arange(len(objects))
plt.xticks(y_pos , objects)
plt.ylabel('Accident count' , fontsize = 15)
plt.xlabel('Age of Drivers', fontsize = 15,)
```

```
Text(0.5, 0, 'Age of Drivers')
```

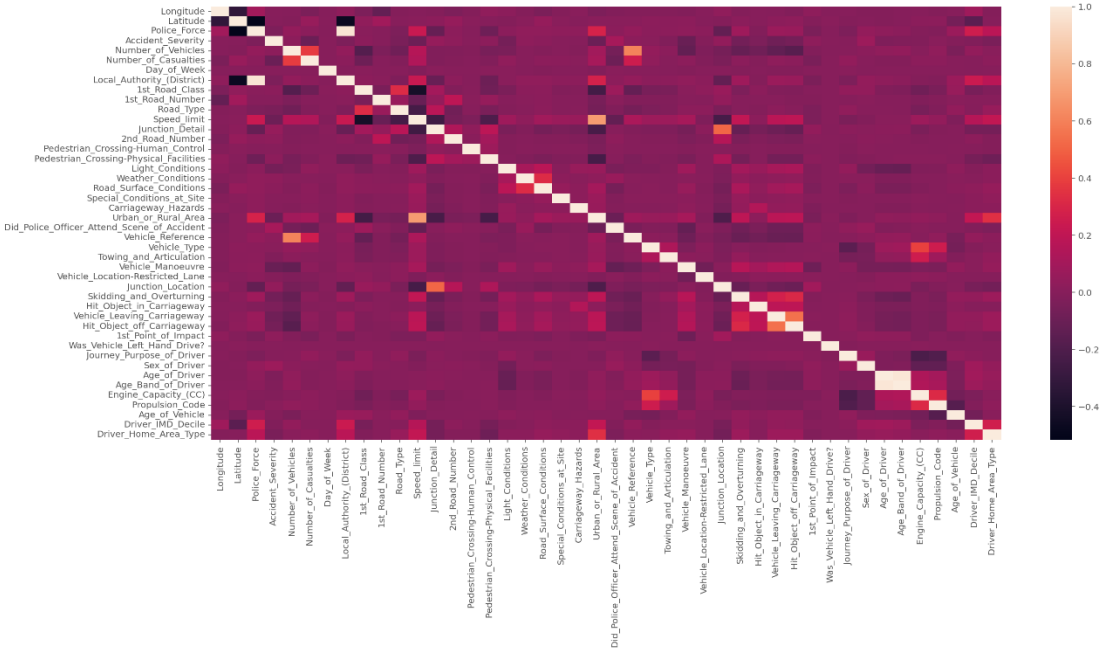



```
speed_zone_accidents = accidents.loc[accidents['Speed_limit'].isin(['20' ,'30' ,'40' ,'50' ,'60' ,'70'])]
speed  = speed_zone_accidents.Speed_limit.value_counts()

explode = (0.0, 0.0, 0.0 , 0.0 ,0.0,0.0)
plt.figure(figsize=(10,8))
plt.pie(speed.values,  labels=None, autopct='%.1f',pctdistance=0.8, labeldistance=1.9 ,explode = explode, shadow=False,

plt.axis('equal')
plt.legend(speed.index, bbox_to_anchor=(1,0.7), loc="center right", fontsize=15, bbox_transform=plt.gcf().transFigure)
plt.figtext(.5,.9,'Accidents percentage in Speed Zone', fontsize=25, ha='center')
plt.show()
```




```
corr =  accidents.corr()
plt.subplots(figsize=(20,9))
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3d00f8ad0>
```



Plotting accidents Location on Google Maps

```
import gmaps
from ipywidgets.embed import embed_minimal_html
gmaps.configure(api_key="AIzaSyCto2LQ-KQh7sYQpzjaYNNa-1V2i5-ASF0")

fig = gmaps.figure(center=(53.0, 1.0), zoom_level=6)
heatmap_layer = gmaps.heatmap_layer(accidents01[["Latitude", "Longitude"]],
                                    max_intensity=30,point_radius=5)
heatmap_layer = gmaps.heatmap_layer(accidents02[["Latitude", "Longitude"]],
                                    max_intensity=5,point_radius=3)
heatmap_layer = gmaps.heatmap_layer(accidents03[["Latitude", "Longitude"]],
                                    max_intensity=1,point_radius=1)
fig = gmaps.figure()
fig.add_layer(heatmap_layer)
fig
```

Machine Learning

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import log_loss
```

Normalize the Data

```
sns.distplot(accidents['Age_of_Driver']);
fig = plt.figure()
sns.distplot(accidents['Age_of_Vehicle']);
fig = plt.figure()
```

```
accidents['Age_of_Driver'] = np.log(accidents['Age_of_Driver'])
accidents['Age_of_Vehicle'] = np.log(accidents['Age_of_Vehicle'])
sns.distplot(accidents['Age_of_Driver']);
fig = plt.figure()
sns.distplot(accidents['Age_of_Vehicle']);
fig = plt.figure()
```

Splitting the data into training data and test data

```
accident_ml = accidents.drop('Accident_Severity' ,axis=1)
accident_ml = accident_ml[['Did_Police_Officer_Attend_Scene_of_Accident' , 'Age_of_Driver' ,'Vehicle_Type', 'Age_of_Vehicle','Eng
                           , 'Light_Conditions', 'Sex_of_Driver' ,'Speed_limit']]

accident_ml.head()

# Split the data into a training and test set.
X_train, X_test, y_train, y_test = train_test_split(accident_ml.values,
                                        accidents['Accident_Severity'].values,test_size=0.20, random_state=99)
```

Logistic Regression

```
lr = LogisticRegression()
# Fit the model on the trainng data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred)
print("Accuracy", round(accuracy_score(y_pred, y_test)*100,2))
print(sk_report)
pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
```

```
Accuracy 86.23
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      4111
           2   0.000000  0.000000  0.000000     38151
           3   0.862323  0.999928  0.926042    264697

    accuracy                       0.862258    306959
   macro avg   0.287441  0.333309  0.308681    306959
weighted avg   0.743599  0.862258  0.798545    306959
```

Predicted	1	3	All
Actual			
1	0	4111	4111
2	4	38147	38151
3	19	264678	264697
All	23	306936	306959

Decision Tree

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree1 = round(decision_tree.score(X_test, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
```

```
Accuracy 75.3
              precision    recall  f1-score   support

           1   0.036994  0.045974  0.040998      4111
           2   0.159336  0.187728  0.172371     38151
           3   0.871153  0.845495  0.858132    264697

    accuracy                       0.753035    306959
   macro avg   0.355827  0.359732  0.357167    306959
weighted avg   0.771512  0.753035  0.761957    306959
```

Predicted	1	2	3	All
Actual				
1	189	890	3032	4111
2	920	7162	30069	38151
3	4000	36897	223800	264697
All	5109	44949	256901	306959

Random Forest

```
random_forest = RandomForestClassifier(n_estimators=200)
random_forest.fit(X_train,y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_test, y_test)
acc_random_forest1 = round(random_forest.score(X_test, y_test) * 100, 2)

sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy" , acc_random_forest1)
print(sk_report)
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
```

```
Accuracy 84.58
              precision    recall  f1-score   support

           1   0.062745  0.007784  0.013850      4111
           2   0.230028  0.056303  0.090463     38151
           3   0.866515  0.972625  0.916509    264697

    accuracy                       0.845817    306959
   macro avg   0.386429  0.345571  0.340274    306959
weighted avg   0.776643  0.845817  0.801753    306959
```

Predicted	1	2	3	All
Actual				
1	32	317	3762	4111
2	105	2148	35898	38151
3	373	6873	257451	264697
All	510	9338	297111	306959

Hyperparameters tuning for the models

```
decision_tree = DecisionTreeClassifier(min_samples_leaf=12, max_features=4)
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree1 = round(decision_tree.score(X_test, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
pd.crosstab(y_test, Y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
```

```
Accuracy 85.69
              precision    recall  f1-score   support

           1   0.063830  0.000730  0.001443      4111
           2   0.316813  0.043564  0.076595     38151
           3   0.866438  0.987450  0.922995    264697

    accuracy                       0.856922    306959
   macro avg   0.415694  0.343914  0.333678    306959
weighted avg   0.787378  0.856922  0.805456    306959
```

Predicted	1	2	3	All
Actual				
1	3	290	3818	4111
2	16	1662	36473	38151
3	28	3294	261375	264697
All	47	5246	301666	306959

```
from sklearn.linear_model import LogisticRegressionCV
lr = LogisticRegressionCV(cv=3, random_state=0,multi_class='multinomial')
# Fit the model on the trainng data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred)
print("Accuracy", round(accuracy_score(y_pred, y_test)*100,2))
print(sk_report)
pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
```

```
Accuracy 86.23
              precision    recall  f1-score   support

           1   0.000000  0.000000  0.000000      4111
           2   0.000000  0.000000  0.000000     38151
           3   0.862318  0.999977  0.926060    264697

    accuracy                       0.862301    306959
   macro avg   0.287439  0.333326  0.308687    306959
weighted avg   0.743594  0.862301  0.798560    306959
```

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	6	264691	264697
All	6	306953	306959