CHAPTER 1

INTRODUCTION

According to the death statistics released by the World Health Organization, the number of traffic accidents occurring annually in the world is alarming. The traffic accidents killed 1.2 million people each year and 50 million people were injured. Approximate 3,300 people were killed and 137,000 people were injured every day. Direct economic losses of 43 billion dollar, the frequent occurrence of traffic accidents directly threaten human life and property safety.

Road accident prediction is one of the most important research area in traffic safety. The occurrence of road traffic accidents is mainly affected by geometric characteristics of road, traffic flow, characteristics of drivers and environment of road. Many studies have been conducted to predict accident frequencies and analyze the characteristics of traffic accidents, including studies on hazardous location/hot spot identification, accident injury-severities analysis, and accident duration analysis. Some studies focus on mechanism of accidents. Other factors include weather and light conditions of the road.

Lee et al [1] developed a probabilistic model relating significant crash precursors to changes in crash potential. Abdel [10] built a previous crash prediction model with the matched case-control logistic regression technique. No specific approach available for the traffic police to predict which area is accident prone at a specific time. The traffic accident prediction play an important role in the integrated planning and management of traffic, the reason which with much randomness about the traffic accident include some nonlinear elements, such as people, car, road, climate and so on. The traditional way of linear analyses can not reveal the really situation since the noise pollution and amount of data are too little, cause the result of prediction can not satisfactory. *B*ecause of the traditional BP network have some defects, such as local minimum, too many iterations, training too slow and so on. The traditional Back propagation network has defects. It has a 7.8% lower accuracy than the proposed model.

1.1 Objective

Machine Learning algorithms can process large number of classification parameters and are able to obtain useful patterns. It can process huge amounts of data efficiently and can be scalable. In computer science and related fields, artificial neural networks are computational models that simulate the central nervous system of the animal (especially the brain), allowing the machine to learn and identify information like the human brain.

1.2 Problem Definition

With the exponentially increasing number of vehicles, road safety is a matter of huge concern.
Road accidents kill 1.2 million people every year.
Road crashes cost $518 billion globally, costing individual countries from 1-2% of their economy.
In 2017, there have been 2367 accidents with injuries reported in Hyderabad alone.
Steps are being taken to combat this issue but they have been ineffective.

1.3 Existing System

No specific approach available for the traffic police to predict which area is accident prone at a specific time.
The traditional Back propagation network has defects. It has a 17% lower accuracy than the proposed model.
We propose the use of a machine learning technique. Machine learning has the ability to model complex non-linear phenomenon.

1.4 Proposed System

An ML powered web app which predicts accidents severity based on the current conditions.
It is trained with 1.6 million accident records over 2005-2015. More data means greater accuracy.
The purpose of such a model is to be able to predict which conditions will be more prone to accidents, and therefore take preventive measures.
We will even try to locate more precisely future accidents in order to provide faster care and precaution service.

According to the predicted severity, a message will be sent to the traffic police to take preventive measures.

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.
● The Result section displays our results and discussions through a series of screenshots. The final part talks about the conclusions and the future scope of our project.

CHAPTER 2

LITERATURE SURVEY

A literature survey in a software development process is a most significant part as it shows the various analyses and research made in the field of your interest including substantive findings, as well as theoretical and methodological contributions to a particular topic. It is the most important part of the report as it gives you a direction in the area of your research; it helps in setting up the goals for the analysis. The purpose is to convey to the reader what knowledge and ideas have been established on a topic, and what their strengths and weaknesses are.

Table 1: Literature Review

S. No.	Title	Author	Year	Objectives
1	A Model of Traffic Accident Prediction Based on Convolutional Neural Network	Lu Wenqi Luo Dongyu Yan Menghua	2017	to predict the traffic accident severity by using convolution neural Network.
2	The Traffic Accident Prediction Based on Neural Network	Fu Huilin, Zhou Yucai	2017	Traditional way of linear analyses can not reveal the really situation the result of prediction is not satisfactory. Compares traditional BP network with its proposed solution.

3	Evolutionary Cross Validation	Thineswaran Gunasegaran Yu-N Cheah	2017	This paper proposes an evolutionary cross validation algorithm for identifying optimal folds in a dataset to improve predictive modeling accuracy
4	On the Selection of Decision Trees in Random Forests	Simon Bernard, Laurent Heutte and Sebastien Adam	2017	This paper presents a study on on the Random Forest (RF) family of ensemble methods.
5	Hyper-parameter Tuning of a Decision Tree Induction Algorithm	Rafael G.Mantovan,, Ricardo Cerri,Joaquin Vanschoren	2016	This paper investigates how sensitive decision trees are to a hyper-parameter optimization process. Four different tuning techniques were explored..

According to the death statistics released by the World Health Organization, the number of traffic accidents occurring annually in the world is alarming. The traffic accidents killed 1.2 million people each year and 50 million people were injured. Approximate 3,300 people were killed and 137,000 people were injured every day. Direct economic losses of 43 billion dollar, the frequent occurrence of traffic accidents directly threaten human life and property safety. Road traffic accident prediction is one of the important research contents of traffic safety. The occurrence of road traffic accidents is mainly affected by geometric characteristics of road, traffic flow, characteristics of drivers and environment of road [1-2]. Many studies have been conducted to predict accident frequencies and analyze the characteristics of traffic accidents, including studies on hazardous location/hot spot identification [3], accident injury-severities analysis [4], and accident duration analysis [5].Some studies focus on mechanism of accidents. Karlaftis et al [6] used hierarchical tree-based regression revisits the relationship between rural road geometric

characteristics, accident rates and their prediction. Lee et al [1] develop a probabilistic model relating significant crash precursors to changes in crash potential. Abdel [10] built a previous crash prediction model with the matched case-control logistic regression technique.In recent years, the deep learning as a new machine learning method began to be highly concerned by researchers and business people. The deep learning theory explains the text, images and sounds, which is widely used in the field of text, image and speech recognition [9], and neural network technology as a highly efficient deep learning technique has been widely used in traffic accident prediction. Compared with the traditional learning structure, deep learning has ability to model complex non-linear phenomenon using distributed and hierarchical feature representation [10].

2.1 PREDICTION FACTORS

The data comes from government website www.data.gov.uk. UK police forces collect the accidents data using the form called Stats19. The data consists of all kind of vehicle collisions from 2005 to 2015. Every column of the dataset is in numerical format. A supporting document to understand each numerical category in accidents dataset is provided on the www.data.gov.uk website

Table 1. *Prediction Factors.*

Day_of_Week :Numeric: 1 for Sunday, 2 for Monday, and so on.
Latitude and Longitude
Light_Conditions : Day, night, street lights or not.
Weather_Conditions: Wind, rain, snow, fog.
Vehicle Type: Pedal cycle, Motorcycle, Car
Road_Surface_Conditions :Wet, snow, ice, flood.

Speed Limit : 60 mph , 70 mph
Output
Accident Severity : 1 = Fatal, 2 = Serious, 3 = Slight

CATEGORY AND MEANING OF WEATHER AND LIGHT CLASSIFICATION FACTORS

Table 3. *Weather Conditions*

code	label
1	Fine no high winds
2	Raining no high winds
3	Snowing no high winds
4	Fine + high winds
5	Raining + high winds
6	Snowing + high winds
7	Fog or mist
8	Other
9	Unknown
-1	Data missing or out of range

Table 4. *Light Conditions*

code	label
1	Daylight
4	Darkness - lights lit
5	Darkness - lights unlit
6	Darkness - no lighting
7	Darkness - lighting unknown
-1	Data missing or out of range

Road Surface Conditions and Gender of Driver and Vehicle Type

Table 5. *Road Conditions*

code	label
1	Dry
2	Wet or damp
3	Snow
4	Frost or ice
5	Flood over 3cm. deep
6	Oil or diesel
7	Mud
-1	Data missing or out of range

Table 6. *Gender*

code	label
1	Male
2	Female
3	Not known
-1	Data missing

Table 7. *Vehicle Type*

code	label
1	Pedal cycle
2	Motorcycle 50cc and under
3	Motorcycle 125cc and under
4	Motorcycle over 125cc and up to 500cc
5	Motorcycle over 500cc
8	Taxi/Private hire car
9	Car
10	Minibus (8 - 16 passenger seats)
11	Bus or coach (17 or more pass seats)
16	Ridden horse
17	Agricultural vehicle
18	Tram
19	Van / Goods 3.5 tonnes mgw or under
20	Goods over 3.5t. and under 7.5t
21	Goods 7.5 tonnes mgw and over
22	Mobility scooter
23	Electric motorcycle
90	Other vehicle
97	Motorcycle - unknown cc
98	Goods vehicle - unknown weight
-1	Data missing or out of range

Table 8. *Day of The Week*

code	label
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

2.2 Validation

Machine learning, especially supervised learning techniques such as classification and regression require training data to build a model. Training data consists of labelled data, i.e. datasets that are complete with the target value together with input feature vectors. A good classification or regression model can be built if significant amount of training data is supplied during the training process. This is followed by the validation process where test data is fed into the trained model to evaluate its predictive accuracy. It is important to test the model properly with enough test data so that the model would yield accurate predictions in the production environment.

Unfortunately, scarcity of data often prompts machine learning practitioners to split the dataset in hand into two subsets, namely training data and test data. These subsets emerge from splitting the original dataset according to a certain ratio such as 80:20 or 60:40, with the bigger proportion making up the training data subset. Training and validating a model using a single train-test split (a.k.a. holdout method) would not yield significant predictive accuracy due to bias. Bias in this case means that in a single train-test split, data points could be clustered in such a way that one cluster gets stuck in the training set and another cluster gets stuck in the test set. Such a situation leads to bias in the train-test split, thus adversely affecting the predictive accuracy of a model.

Therefore, it is important to utilize several unique splits of training and test data to build an accurate model. Cross validation utilizes several train-test splits, a.k.a folds and this technique enables the machine learning model to be trained with less bias because all different clusters of data points get to be chosen as training data in different folds. This helps to reduce bias in training the model.

2.3 Decision Trees and Random Forests

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules**.**

A decision tree consists of three types of nodes:

Decision nodes – typically represented by squares

Chance nodes – typically represented by circles

End nodes – typically represented by triangles

One purpose of Machine Learning is to design high performance classification systems from a set of representative samples of a population of data. An efficient way to tackle this kind of problematic is to combine an ensemble of individual classifiers to form a unique classification system, called Classifier Ensemble. This approach has been fed since the early 90's, by researches that have shown some combination principles to be particularly efficient, such as Boosting [1] (or Arcing [2]), Bagging [3], Random Subspaces [4], or more recently, Random Forests [5]. The efficiency in combining classifiers leans on the ability to take into account the complementarity between individual classifiers, in order to improve as much as possible the generalization performance of the ensemble. This ability is often defined through the diversity property. Although there is no agreed definition for diversity [6], this concept is usually recognized to be one of the most important characteristics for the improvement of the generalization performance in an ensemble of classifiers [7]. One can define it as the ability of the individual classifiers of an ensemble to agree mainly on good predictions and to disagree on prediction errors.

Random Forest (RF) family of ensemble methods. In a "classical" RF induction process a fixed number of randomized decision trees are inducted to form an ensemble. This kind of algorithm presents two main drawbacks: (i) the number of trees has to be fixed a priori (ii) the interpretability

and analysis capacities offered by decision tree classifiers are lost due to the randomization principle. This kind of process in which trees are independently added to the ensemble, offers no guarantee that all those trees will cooperate effectively in the same committee.

2.4 DECISION TREE HYPERPARAMETER TUNING

Supervised classification is the most studied task in Machine Learning. Among the many algorithms used in such task, Decision Tree algorithms are a popular choice, since they are robust and efficient to construct. Moreover, they have the advantage of producing comprehensible models and satisfactory accuracy levels in several application domains. Like most of the Machine Learning methods, these algorithms have some hyperparameters whose values directly affect the performance of the induced models. Due to the high number of possibilities for these hyper-parameter values, several studies use optimization techniques to find a good set of solutions in order to produce classifiers with good predictive performance.

Four different tuning techniques were explored to adjust J48 Decision Tree algorithm hyper-parameters. In total, experiments using 102 heterogeneous datasets analyzed the tuning effect on the induced models. The experimental results show that even presenting a low average improvement over all datasets, in most of the cases the improvement is statistically significant.

Supervised classification is one of the main Machine Learning (ML) tasks, and as a consequence, there is a large variety of classification algorithms available. Among them, Decision Tree (DT) induction algorithms have been popularly used [1]. As classifiers, DTs are represented by rules structured as a tree, being widely used especially due to its comprehensible nature which resembles the human reasoning [2]. Some authors stated that DTs also figure among the most used data mining algorithms by researchers and practitioners, which reinforces its importance in the ML area [3], [4]. DT induction algorithms have several advantages over many other ML algorithms, such as robustness to noise (missing values, imbalanced classes), low computational cost, and the ability to deal with redundant attributes [2]. There are many well-known DT induction algorithms in literature, such as Quinlan's C4.5 algorithm [5] and Breiman et al.'s Classification and Regression Tree (CART) [6]. The values chosen for the hyper-parameters (HPs) of ML algorithm directly

affect the predictive performance of the models induced by them. Thus, a good choice of these values has been the subject of study in ML for years.

HYPER-PARAMETER TUNING HP tuning can largely affect the predictive performance of ML algorithms [9]. Setting a suitable configuration for the HPs of a ML algorithm is usually performed by trial and error. Depending on the training time of the ML algorithm used, finding a good set of values manually can be very time consuming. As a result, recent works in HP for ML algorithms focus on the development of better HP tuning techniques [12], [20]. The HP process is usually treated as an optimization (blackbox) problem, whose objective function is associated with the predictive performance of the model induced by the algorithm.

CHAPTER 3

METHODOLOGY

We have developed a web app for our model. It consists of four components:

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

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

Machine Learning Model: We have used decision tree, random forest and logistic regression and also applied hyperparameter tuning to increase its efficiency. Random Forest algorithm showed the highest accuracy of 86.86% and hence chosen for our model. The model runs and predicts the severity. The severity metrics are 1= Fatal, 2= Serious, 3= Slight.

The output is sent back to the front-end and displayed to the user.

An sms containing the location coordinates and the severity of accident is sent to the police so that it can take preventive measures at the location.

3.1 System Design

Describes the data flow in a diagrammatic representation.

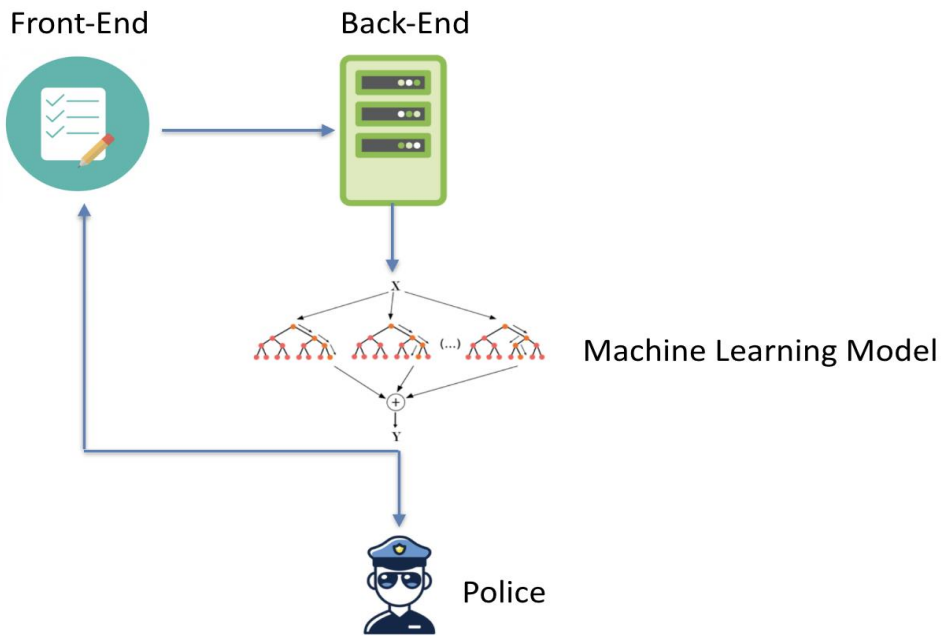


Figure 3.1 System model

3.2 Modules

1. **The Virtual Machine :** It has the trained and tested Machine learning algorithm implemented. The frontend and backend server are deployed on it.
2. **The front end (User) :** Geolocation Api takes the location of the user and sends it to the OpenWeatherMap Api which sends geographical conditions. User input is taken for other parameters like age, sex etc. User can view the heatmap of the accidents in the country.
3. **The back end (Admin):** The server is created and maintained. The input details are feeded to the model and severity is predicted. The severity can be sent as a message or email to the police to take preventive measures.
4. **Machine Learning Algorithm:** Classification Algorithms decision tree, random forest and logistic regression have been implemented. Hyperparameter tuning has been applied to find the best accuracy. Random forest has shown the highest accuracy with 86% and has been selected as the model for the web app.

3.3 Technologies Used

3.3.1 Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is an interpreted, high-level, general-purpose programming language. Python features a dynamic type system and automatic memory management. It supports multiple programming

paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library.

It is the world's fastest growing and most popular programming language used by software engineers, analysts, data scientists, and machine learning engineers alike. It is used by sites like YouTube and Dropbox.

It supports functional and structured programming methods as well as OOP.
It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic type checking. It supports automatic garbage collection.
It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java. Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure

3.3.2 Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image.

3.3.3 Google collab

Colaboratory (also known as Colab) is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive. Colaboratory started as a part of Project Jupyter, but the development was eventually taken over by Google[21]. As of September 2018, Colaboratory only supports the Python 2 and Python 3 kernels and does not support the other Jupyter kernels Julia and R.Project Jupyter is a nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". Spun-off from IPython in 2014 by Fernando Pérez, Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also an homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, Jupyter Hub, and Jupyterlab, the next-generation version of Jupyter Notebook.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension. It is used to run resource intensive tasks.

3.3.4 Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

3.3.5 Azure

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Microsoft lists over 600 Azure services,[4] of which some are covered below:

1) Compute

2) Mobile Services

3) Storage services

4) Data Management

5) Machine learning

3.3.6 Domain name and SSL Certificate

A domain name is your website name. A domain name is the address where Internet users can access your website. A domain name is used for finding and identifying computers on the Internet. Computers use IP addresses, which are a series of number.

Azure is used to buy a custom domain name. Binding of SSL certificate is done. It allows us to use the https communication. SSL Binding requires valid private certificate (.pfx) issued for the specific hostname.

3.3.7 API

Application Programming Interface (API) In basic terms, APIs just allow applications to communicate with one another and data to one another.

Apis used are:

1. The Geolocation API returns a location and accuracy radius based on information about cell towers and WiFi nodes that the mobile client can detect. This document describes the protocol used to send this data to the server and to return a response to the client. Communication is done over HTTPS using POST. Both request and response are formatted as JSON, and the content type of both is application/json.
2. Weather Api: Provided by OpenWeatherMap, you have access to current weather data, 5- and 16-day forecasts, UV Index, air pollution, weather conditions etc.
3. Sms Api: Provided by Text Local. Can be easily integrated with any application and can be used to start sending SMS in minutes.

3.3.8 SSH Client

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line login and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network in a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Windows. Windows 10 uses OpenSSH as its default SSH client.

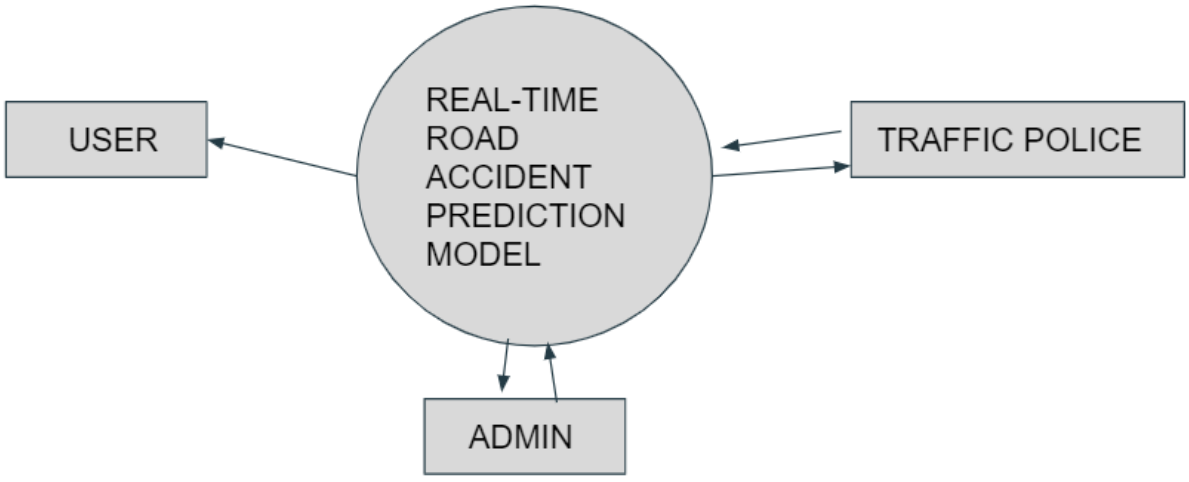
SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rlogin, rsh, and rexec protocols. Those protocols send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet, although files leaked by Edward Snowden indicate that the National Security Agency can sometimes decrypt SSH, allowing them to read the contents of SSH sessions.

3.4 Diagrammatic Representation

3.4.1 Data flow diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

3.4.1.1 DFD level 0



Frigure 4.1.1 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

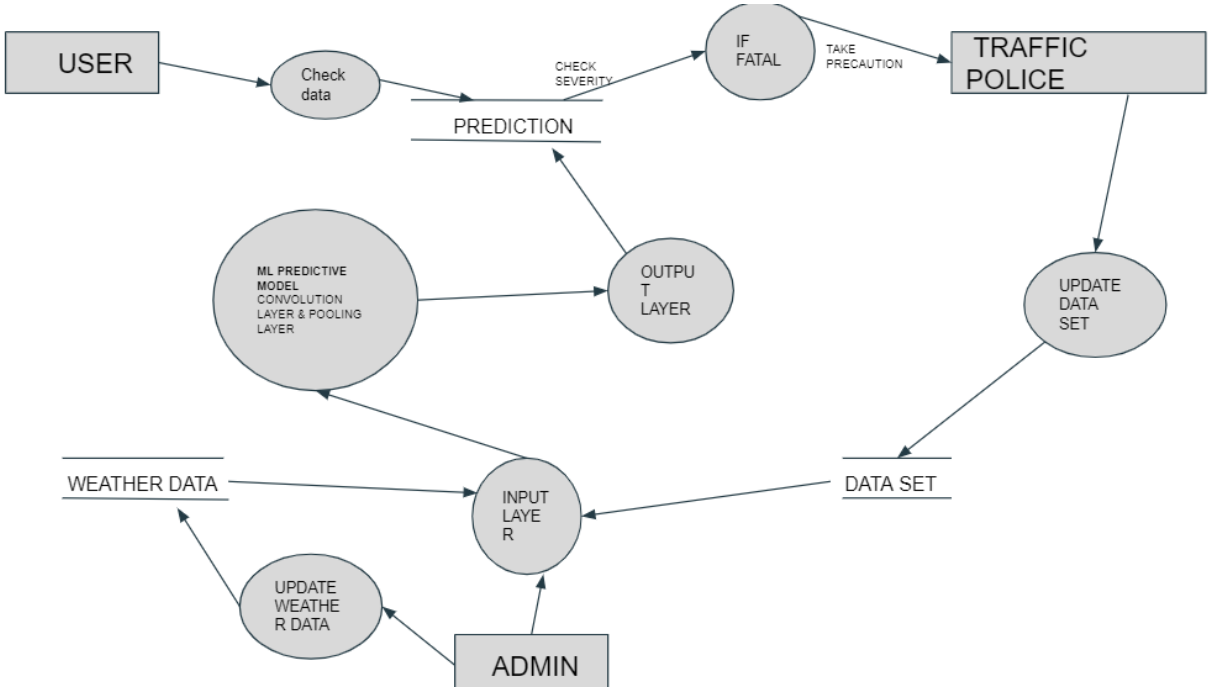


Figure 4.1.2 DFD level 1

- 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

UML is the international standard notation for object-oriented analysis and design. The object management group defines it. The heart of object-oriented problem solving is the construction of a model. The model abstracts the essential details of the underlying problem from its usually complicated real world. The scope UML is a language for specifying artifacts, visualizing artifacts, constructing artifacts and documenting artifacts. UML provides the following diagrams to represent the software process:

- Class Diagram
- Use Case diagram
- Sequence diagram

3.4.2.1 Class Diagram

class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

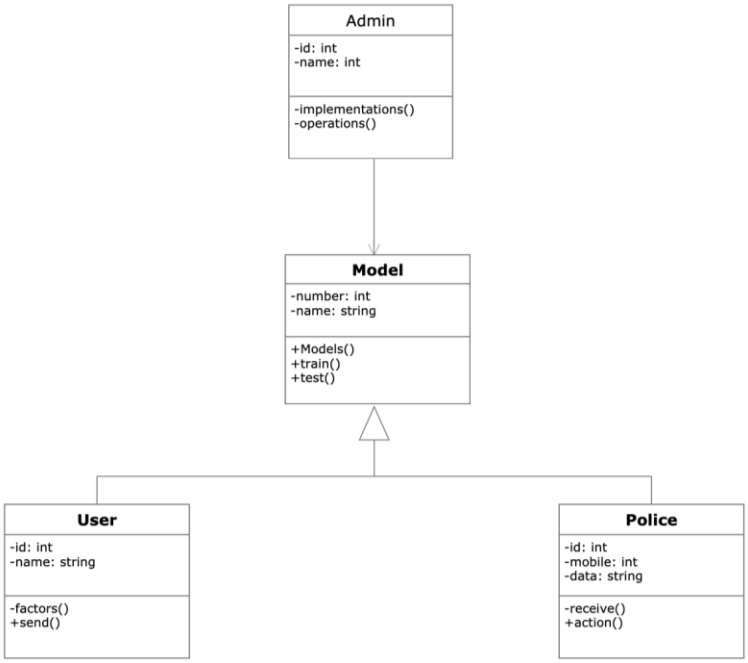


Figure 4.2.3 Class diagram

3.4.2.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

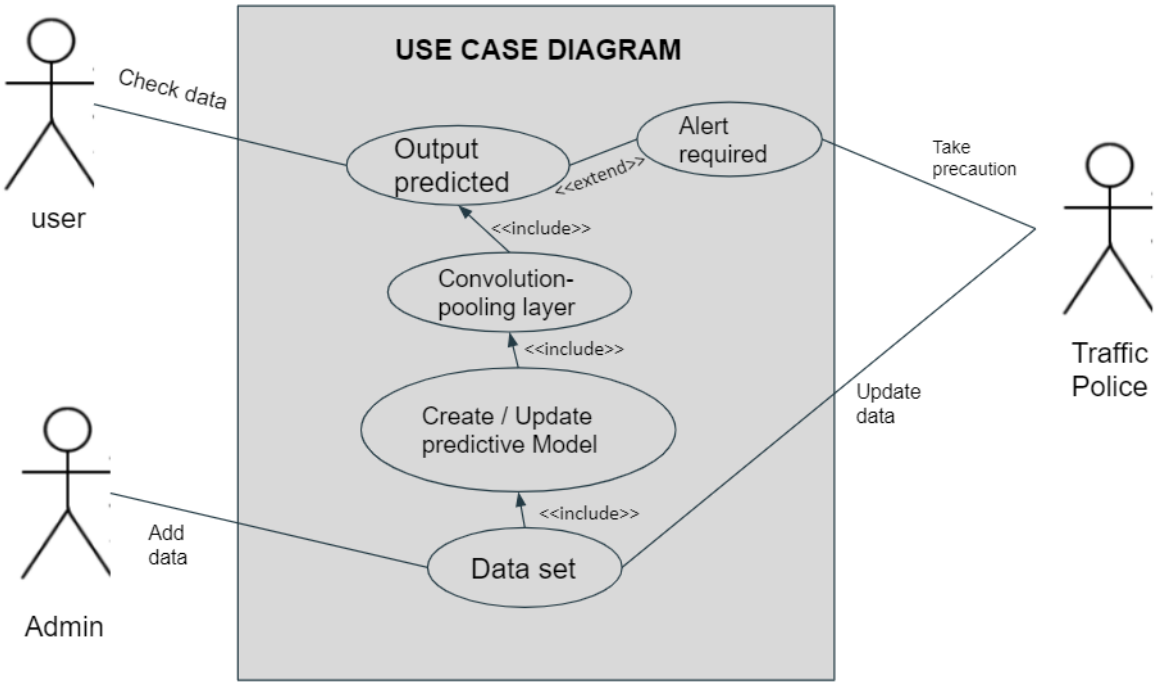


Figure 4.2.2 Usecase diagram

- In the system, there are two actors:user and play store.
- The user performs the tasks of searching for an application and viewing the result if an application is malicious.
- The play store downloads the required application and its comments.
- The other actions performed in the system are testing and sentiment analysis on the download application and comments.

3.4.2.3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

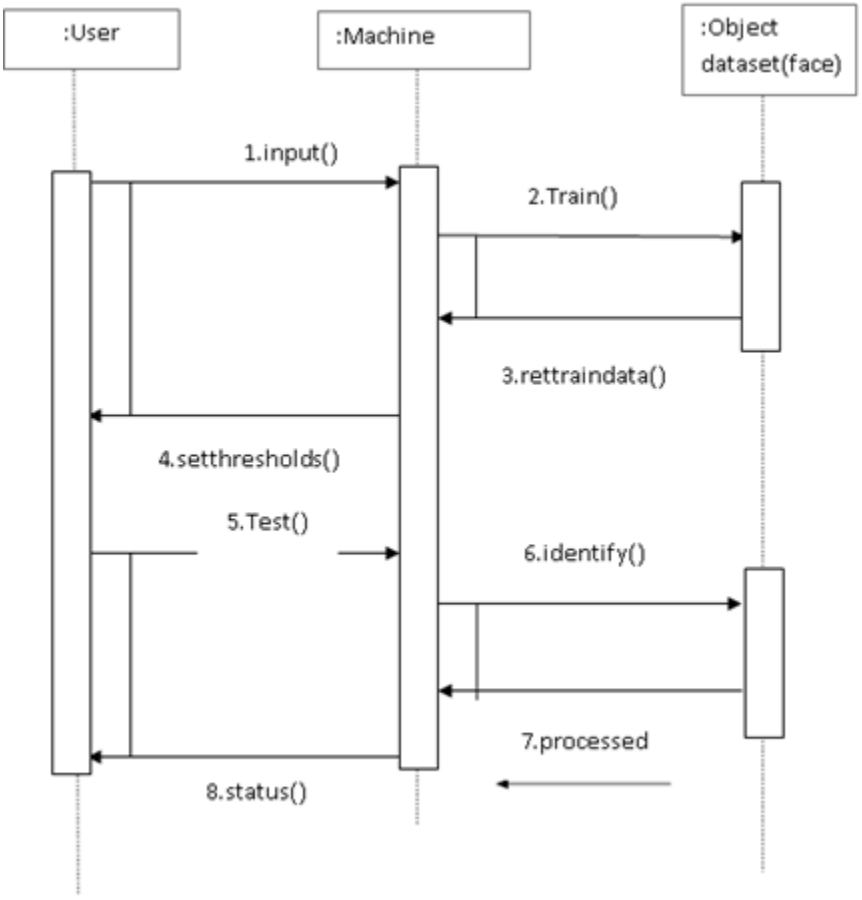


Figure 4.2.3 Sequence diagram

● The user represented by the object of ‘User’ class performs the first operation ‘searchApp’ in the system by sending a message to object of ‘Store’ class that represents the Google Play Store.This operation searches for an application in the Play Store.

● The object of ‘Store’ class then sends a message ‘download’ to an object of ‘Server’ class. This denotes that the required application has to be downloaded.

● The ‘Server’ class object sends the downloaded application to ‘Analysis’ class object and indicates it to perform sentiment analysis and testing on it. This is done through a message ‘test’.

● Finally, the analysis is completed by ‘Analysis’ object and returns the result to the user through ‘sendResult’.

3.5 Implementation of Proposed solution

There are four important steps:

1. Preprocessing
2. Training
3. Testing
4. Web App Integration

3.5.1 Data Importing

We import three files to perform analysis on this data. This data is consist of three files that are accidents, casualties 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.

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

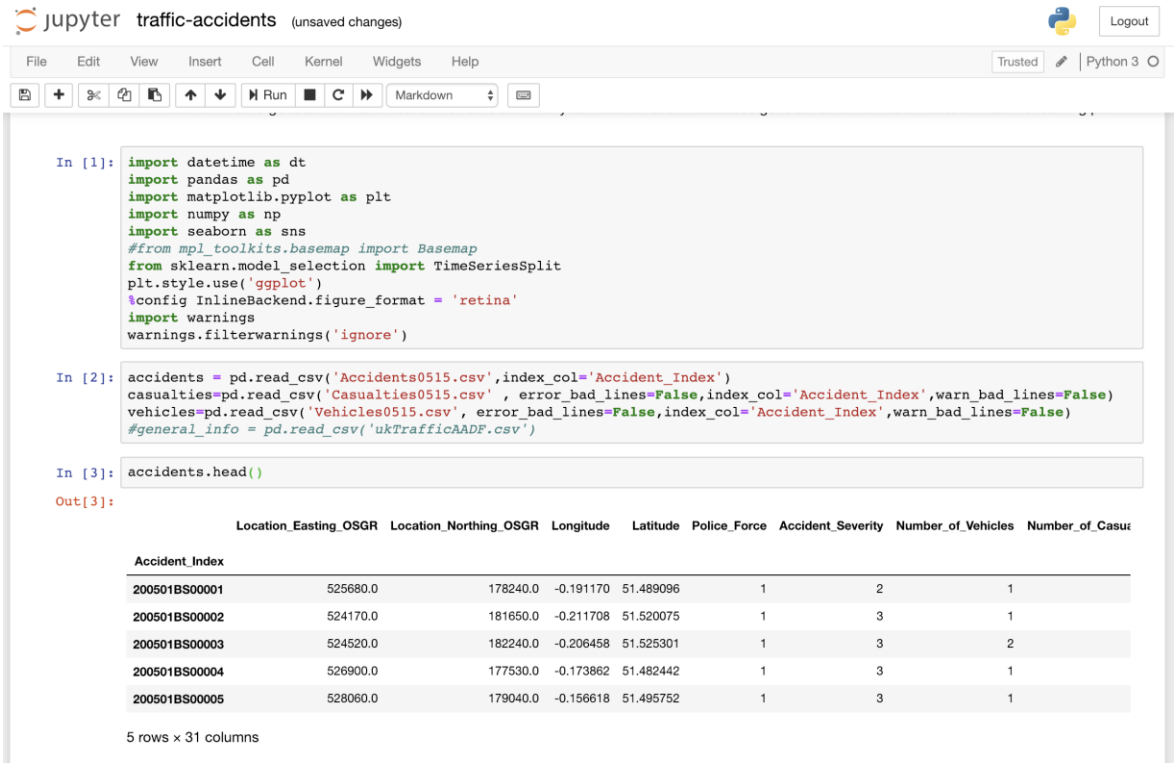


Fig 3.1 Importing

3.5.2 Preprocessing of Data

Data Cleaning

Her we identify noisy, irrelevant data. We also understand through visualization which factors are more important.

Identifying Missing Values

In this particular dataset, there are two types of missing values '-1' and 'Nan'. We will invesitigate each column with total missing values. We will not be imputing any mean or median value since the dataset is big enough to perform analysis.

```
In [5]: accidents.drop(['Location_Easting_OSGR', 'Location_Northing_OSGR','LSOA_of_Accident_Location',
                        'Junction_Control' ,'2nd_Road_Class'], axis=1, inplace=True)
        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.

```
In [4]: accidents = accidents.join(vehicles, how='outer')
```

Fig 3.2 Join

Data Visualization

The first thing we can do is to find out about accidents time to get intution 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

We can find out the number of accidents on the days of a week. As we can see that thursday has the highest amount of accidents in this dataset from 2005 to 2015. We have to keep in mind that accidents numbers could be depending on traffic amount on particular day.

```
In [6]: 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' , fontsi

Out[6]: Text(0.5,0,'0 - Sunday ,  1 - Monday  ,2 - Tuesday , 3 - Wednesday , 4 - Thursday , 5 - Friday , 6 - Saturday')
```

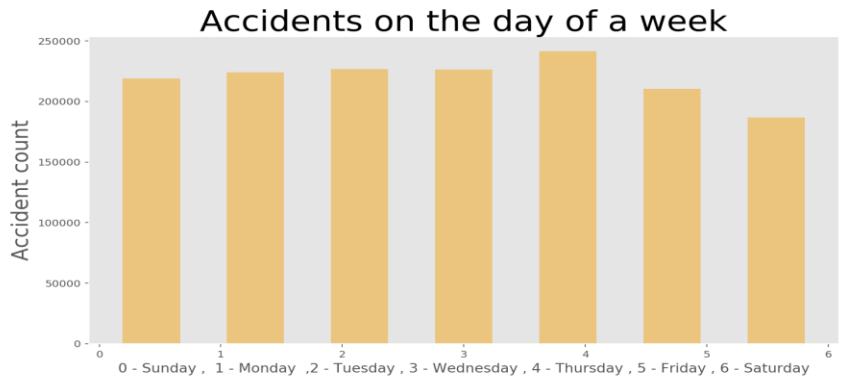


Fig 3.3 Accidents

Time of Accident

He we found out that the most of accidents happened around after noon. We can assume that this time of the day has the most traffic moving such as people leaving from work.

```
In [7]: 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)

Out[7]: Text(0,0.5,'Accident count')
```

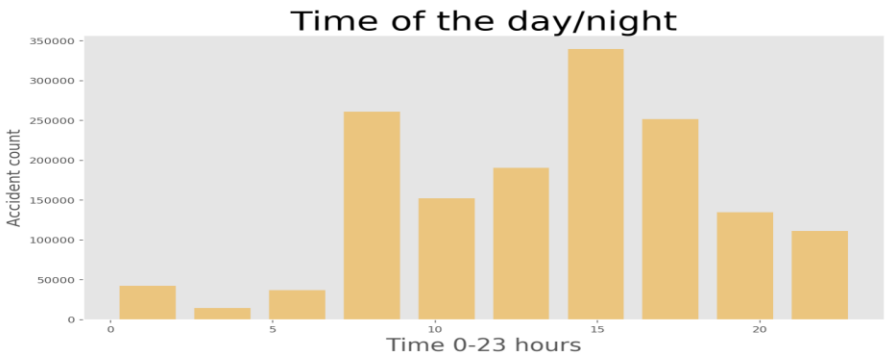


Fig 3.4 Time

Age Band of Casualties

In this dataset, age band is grouped in 11 different codes. We will create the labels and pass it to the plot as xticks so we can have idea about the bins representation

```
In [8]: 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,)

Out[8]: Text(0.5,0,'Age of Drivers')
```

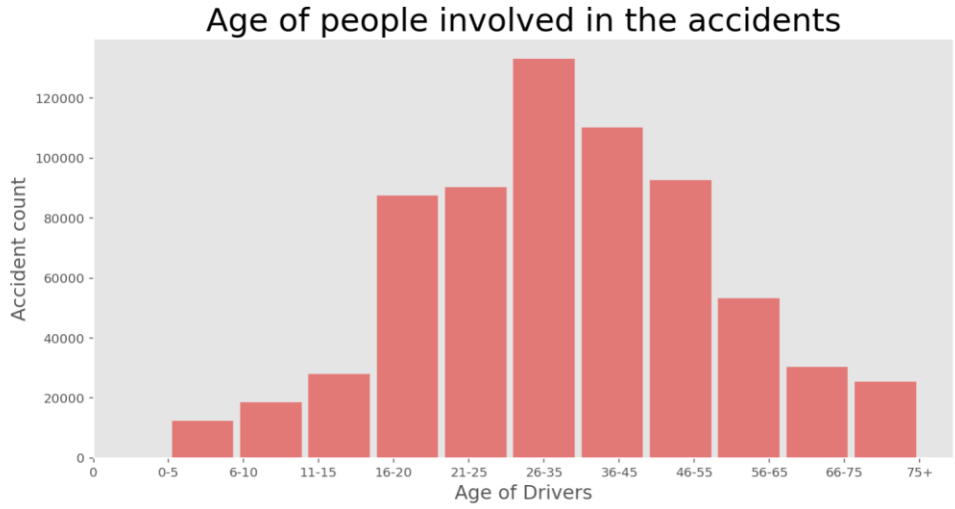


Fig 3.5 Age

This is very interesting fact about this dataset. Most of the drivers age is around 225 to 35 who are involved in the accident. However, we do not know the number of drivers with age 25 to 35 on the road compare to other ages. Intuitively, I would assume that the driver with age 25 to 35 are more in the number of drivers with different age.

Co-relation between variables

Since our dataset is in numeric values. We can find out correlation between columns.

As we see that there is not so much strong correlations between any variables. There is only one positive strong correlation between speed limit and Urban or Rural Area.

```
In [52]: corr =  accidents.corr()
         plt.subplots(figsize=(20,9))
         sns.heatmap(corr)

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3fde1c50>
```

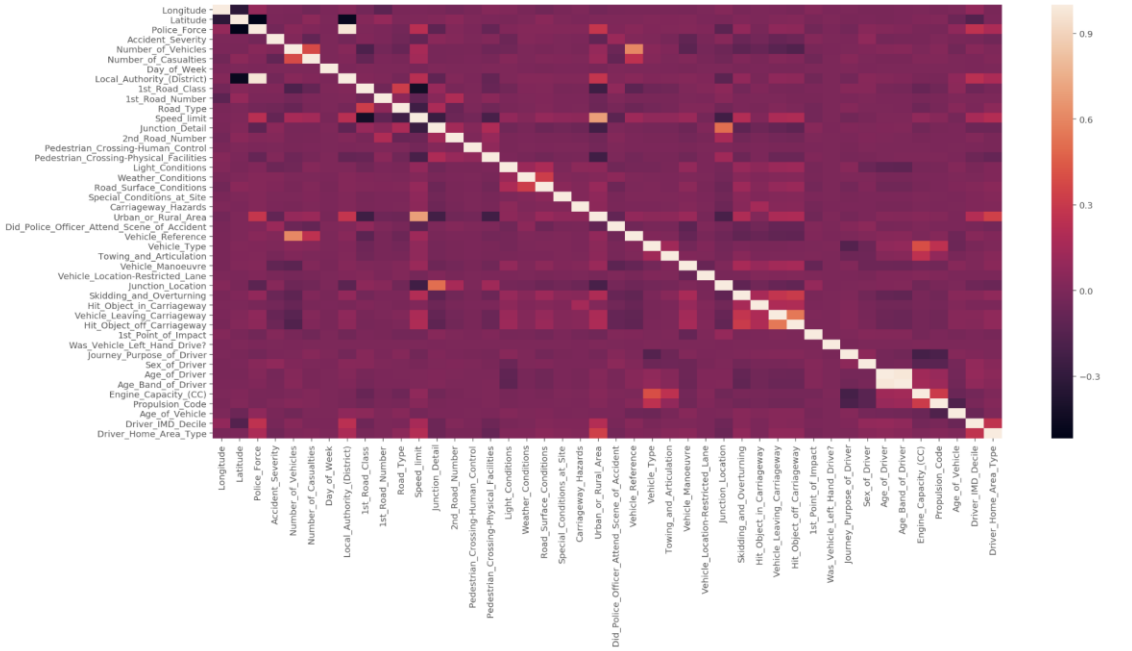


Fig 3.6 Correlation

Speed of Cars

```
In [41]: 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, startangle=160,textprops={'

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()
```

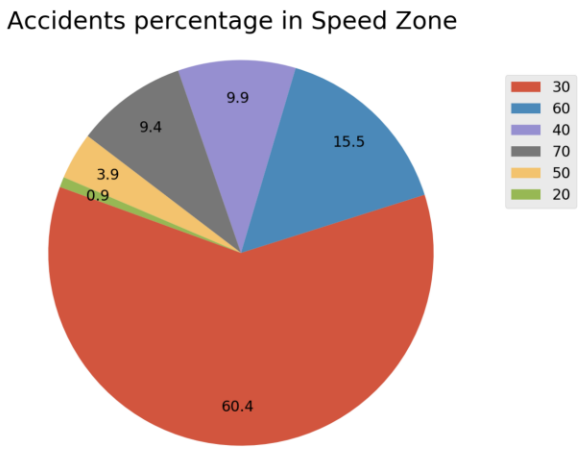


Fig 3.7 Speed

Most of the accidents occurred on the road where the speed limit is 30. We were expecting more accidents on highway or major roadways. Some of the accidents could be cause of stop sign, changing lanes or turning into parking lot etc.

Plotting accidents Location on Google Maps

Classifying locations based on severity

```
In [11]: accidents_2014 = accidents[accidents.Date_time.dt.year ==2014]
accidents_2014_01 = accidents_2014[accidents_2014.Accident_Severity == 1]
accidents_2014_02 = accidents_2014[accidents_2014.Accident_Severity == 2]
accidents_2014_03 = accidents_2014[accidents_2014.Accident_Severity == 3]
```

```
In [50]: !pip install gmaps
         !jupyter nbextension enable --py gmaps
         import gmaps
         gmaps.configure(api_key='AIzaSyD3t4mfJNy9NxxVKT4J_T47soKBgCRUTO4')

         fig = gmaps.figure(center=(53.0, 1.0), zoom_level=6)
         heatmap_layer = gmaps.heatmap_layer(accidents_2014_01[["Latitude", "Longitude"]],
                                             max_intensity=30,point_radius=8)
         heatmap_layer = gmaps.heatmap_layer(accidents_2014_02[["Latitude", "Longitude"]],
                                             max_intensity=5,point_radius=3)
         heatmap_layer = gmaps.heatmap_layer(accidents_2014_03[["Latitude", "Longitude"]],
                                             max_intensity=1,point_radius=1)

         fig = gmaps.figure()
         fig.add_layer(heatmap_layer)
         fig
```

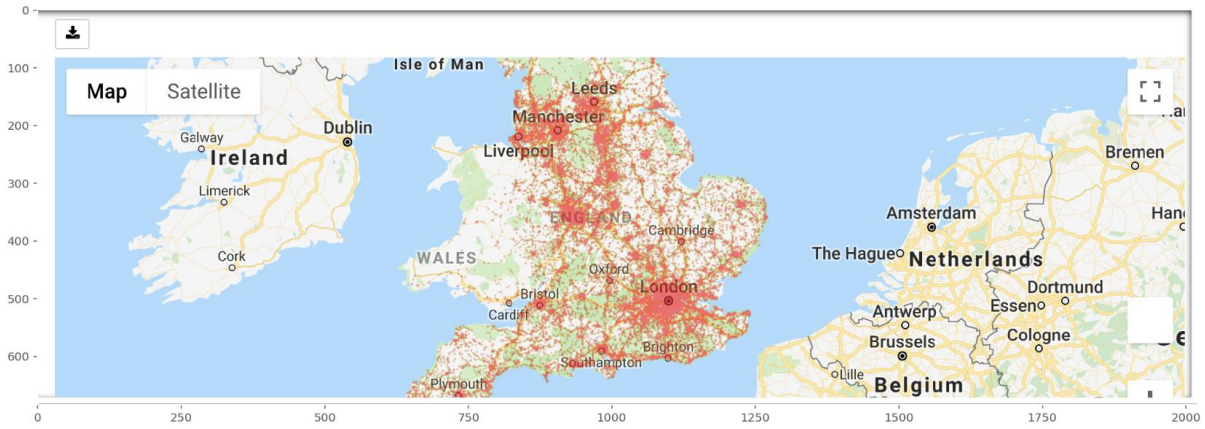
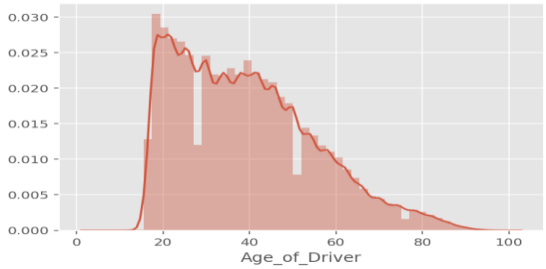
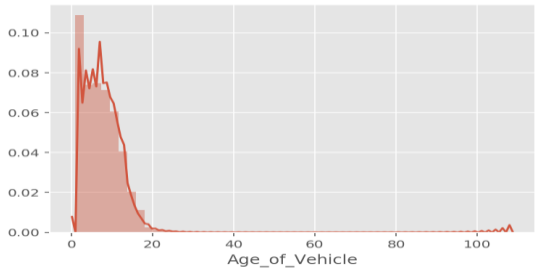


Fig 3.8 Heatmap

Normalize the Data

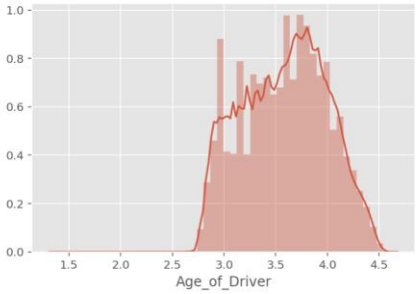
There are few columns that we will standardize, so it would not affect negatively on our machine learning algorithms. Age of driver is from 18 to 88 in the dataset and we can normalize it. Also, the age of vehicle is also from 0 to 100 and it can skew the performance of your machine learning algorithm and we will normalize this predictor too.

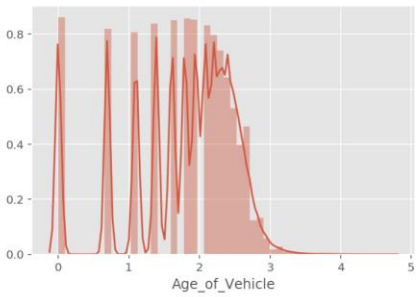




Before Normalization

```
In [ ]: 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()
```





After Normalization

Fig 3.9 Normalization

3.5.3 Machine Learning

We will be looking at different columns to figure out predicting about the accidents severity. After we can predict the accident severity, we can make some recommendation to law enforcement for looking into this and be prepared for the future.

Following packages are being imported.

```
In [136]: 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 4.0 Packages

Splitting the data into training and test data

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

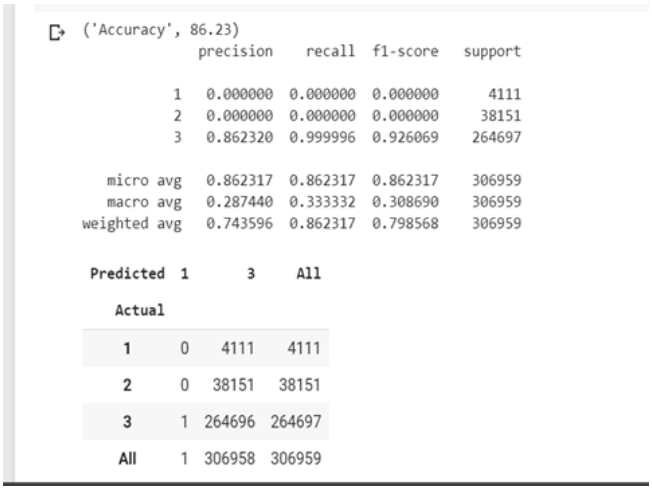
20% of the data is for testing and 80% for training.

```
In [ ]: 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_Veh
                                   , 'Light_Conditions', 'Sex_of_Driver' ,'Speed_limit']]

        # 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



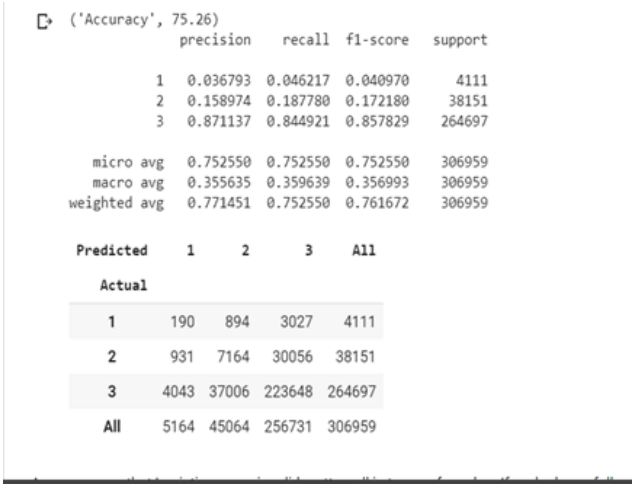
('Accuracy', 86.23)

	precision	recall	f1-score	support
1	0.000000	0.000000	0.000000	4111
2	0.000000	0.000000	0.000000	38151
3	0.862320	0.999996	0.926069	264697
micro avg	0.862317	0.862317	0.862317	306959
macro avg	0.287440	0.333332	0.308690	306959
weighted avg	0.743596	0.862317	0.798568	306959

Actual \ Predicted	1	3	All
1	0	4111	4111
2	0	38151	38151
3	1	264696	264697
All	1	306958	306959

Fig 4.1 Accuracy: Logistic Regression

Decision Tree



('Accuracy', 75.26)

	precision	recall	f1-score	support
1	0.036793	0.046217	0.040970	4111
2	0.158974	0.187780	0.172180	38151
3	0.871137	0.844921	0.857829	264697
micro avg	0.752550	0.752550	0.752550	306959
macro avg	0.355635	0.359639	0.356993	306959
weighted avg	0.771451	0.752550	0.761672	306959

Actual \ Predicted	1	2	3	All
1	190	894	3027	4111
2	931	7164	30056	38151
3	4043	37006	223648	264697
All	5164	45064	256731	306959

Fig 4.2 Accuracy: Decision Tree

Random Forest

```
('Accuracy', 86.86)
                precision    recall  f1-score   support

              1   0.031496  0.002928  0.005358      1366
              2   0.195915  0.040622  0.067291     20777
              3   0.884926  0.979143  0.929653    166321

      micro avg   0.868601  0.868601  0.868601    188464
      macro avg   0.370779  0.340898  0.334101    188464
   weighted avg   0.802781  0.868601  0.827884    188464

   done
```

Fig 4.31 Accuracy: Random Forest

Hyperparameters 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   0.862317  0.999974  0.926058    264697

     micro avg   0.862298  0.862298  0.862298    306959
     macro avg   0.287439  0.333325  0.308686    306959
  weighted avg   0.743594  0.862298  0.798558    306959
```

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	7	264690	264697
All	7	306952	306959

Fig 4.4 Accuracy: Logistic Regression Hyperparameter

Hyperparameters tuning for Decision Tree

```
('Accuracy', 85.71)
              precision    recall  f1-score   support

           1   0.071429  0.000730  0.001445      4111
           2   0.323387  0.045451  0.079700     38151
           3   0.866655  0.987333  0.923066    264697

   micro avg   0.857056  0.857056  0.857056    306959
   macro avg   0.420490  0.344504  0.334737    306959
weighted avg   0.788483  0.857056  0.805904    306959
```

Predicted	1	2	3	All
Actual				
1	3	301	3807	4111
2	13	1734	36404	38151
3	26	3327	261344	264697
All	42	5362	301555	306959

Fig 4.4 Accuracy: Decision Tree Hyperparameter

Table 9. *Accuracy Of Algorithms*

ALGORITHM	ACCURACY
TRAIN TEST- SCIKETLEARN CROSSVALIDATION	-
DECISION TREE	75.32
RANDOM FOREST	86.86
LOGISTIC REGRESSION	86.23
DECISION TREE HYPERPARAMETER TUNING	85.74
LOGISTIC REGRESSION WITH HYPERPARAMETER TUNING	86.23

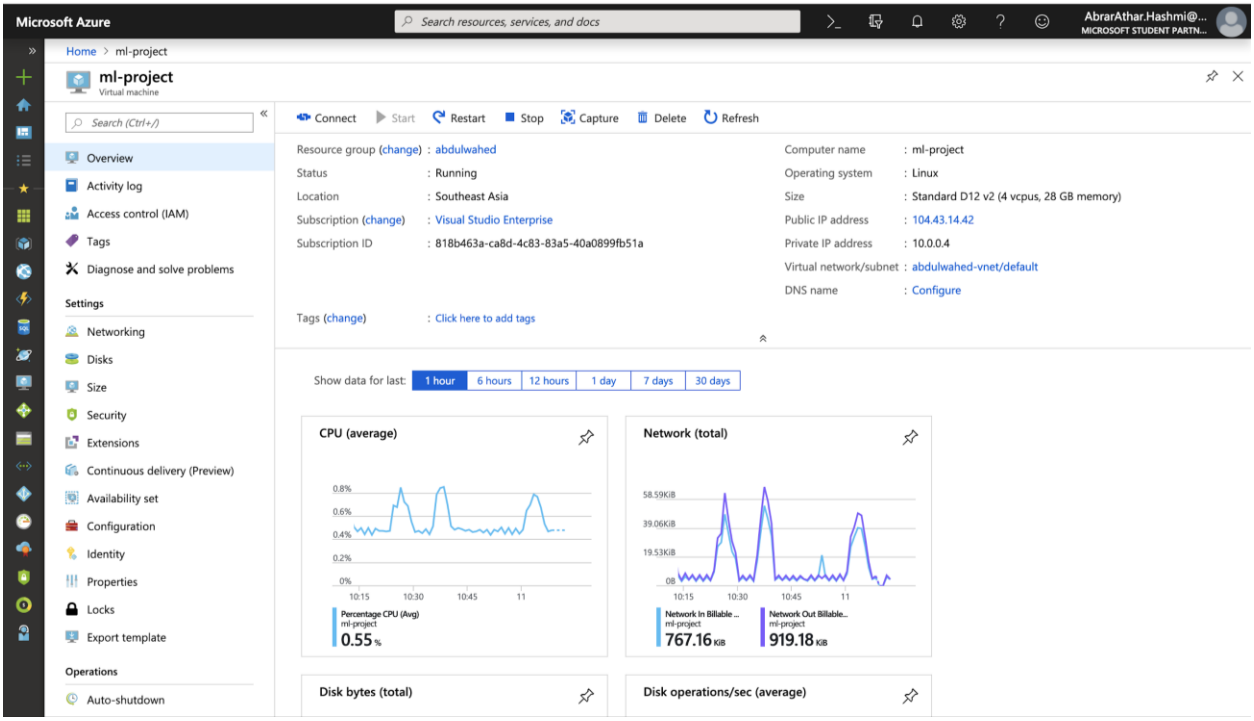


Fig 4.6 Azure VM page

Virtual Machine deployed on Azure.

We have chosen Random Forest as our model as it has the highest accuracy (86.86%).

Input taken from user is sent to the backend flask server which feeds the parameters to the ML model and returns the result. It also sends a message to the police to take preventive measures.

3.6 System Requirements

1. Windows XP, 7, 8, 10, Server 2003.
2. MacOS, iOS
3. Android
4. Windows Phone 8 and Windows 10 Mobile
5. Language Used: Python, JavaScript
6. IDE and Framework: Jupyter Notebook, Sublime, Flask
7. Cloud: Azure ML, Google Colab

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	: 16 GB

CHAPTER 4

RESULTS AND DISCUSSIONS

A web app has been developed for our model. It can be easily accessed through the custom domain name https://www.accidentprediction.com:4000.

The Front-End which is the home page takes input for the prediction factors in two ways.

1. Location and Latitude: This will be automatically taken from the browser using the GeoLocation Api and it is sent to the OpenWeatherMap Api. This api gives us the weather, road conditions, light conditions and day of the week which are implicitly updated in the backend.
2. User age, gender, vehicle type, vehicle age and engine capacity: This data is to be explicitly entered by the user.

The model is deployed in the back-end. The input data from the front-end is fed into the Machine Learning model. We have used Random Forest algorithm which showed the highest accuracy of 86.86% as our model. The model runs and predicts the severity. The severity metrics are 1= Fatal, 2= Serious, 3= Slight.

The output is sent back to the front-end and displayed to the user.

An SMS containing the location coordinates and the severity of accident is sent to the police so that it can take preventive measures at the location.

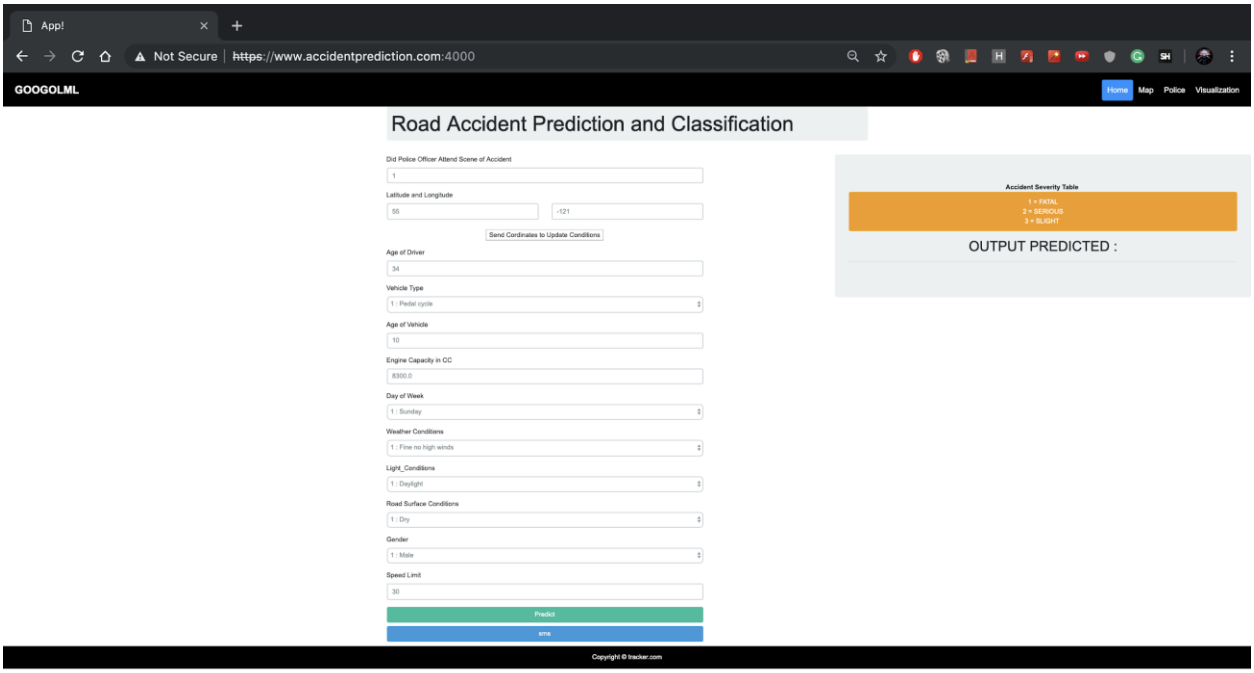


Figure 4.1 User page

The above figure 4.1 shows the home page of the web app. The web domain is secured with HTTPS wich has been obtained from the certificate authority for secure data transfer and to be able to use the Geolocation API. Displays the data owner login web page, which allows data owner to login and also to register, if the user does not have existing account.

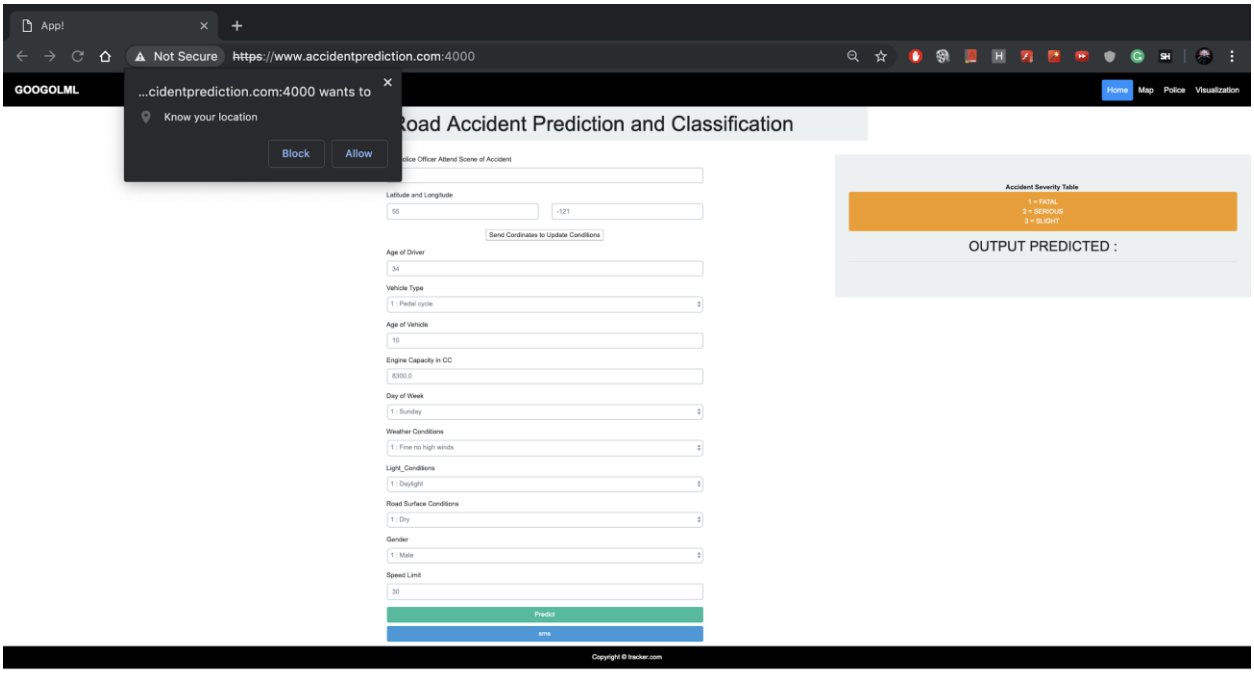


Figure 4.2 User Location by GPS

Figure 6.2.2 shows that when user clicks on update coordinates button, the web page requests the browser to take user coordinates. In the backend flask module, GeoLocation API is used to get location of the user. Ajax is used to update the latitude and longitude of the user in the web page.

The coordinates are sent to the OpenWeatherMap Api in the backend for the weather details. From the response we extract the details we require such as weather , road and light conditions.

Day of the week is updated with the getDate function of javascript.

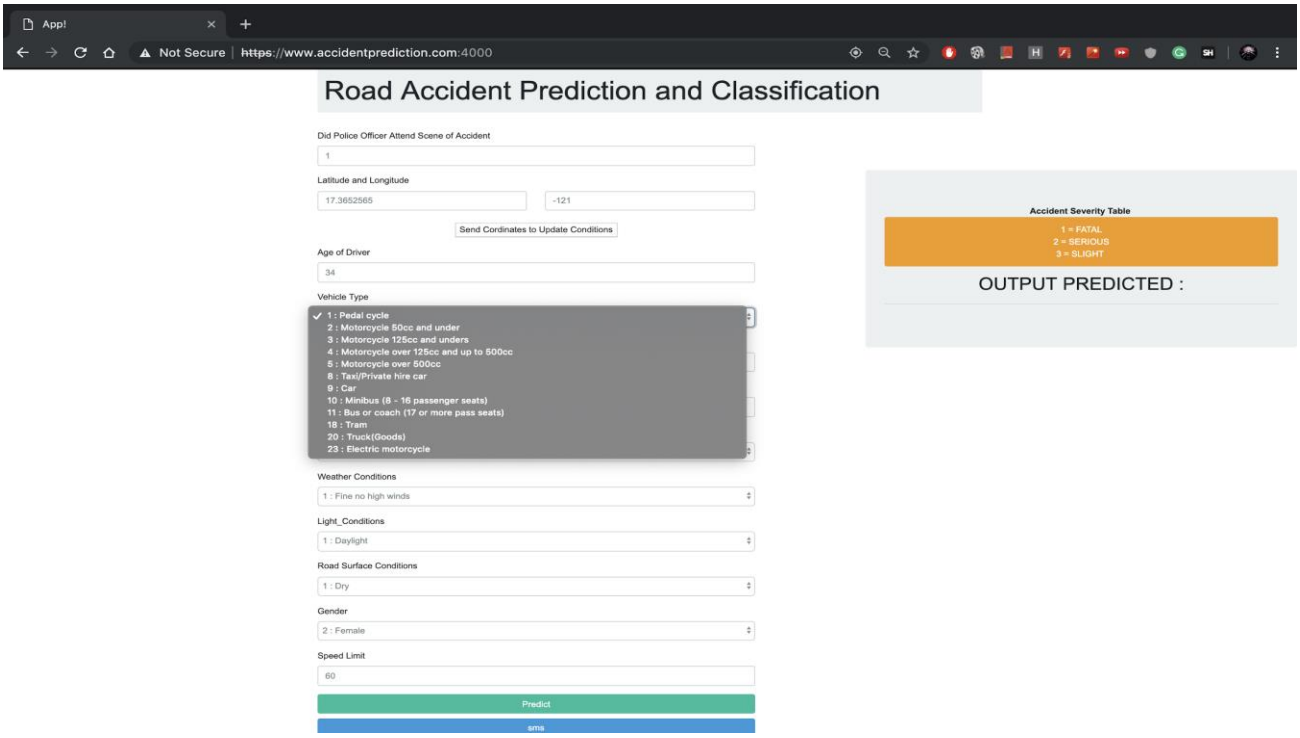


Figure 4.9 User input for other parameters

Figure 4.9 shows the input for parameters taken from users. These include the vehicle type, age gender and speed limit.

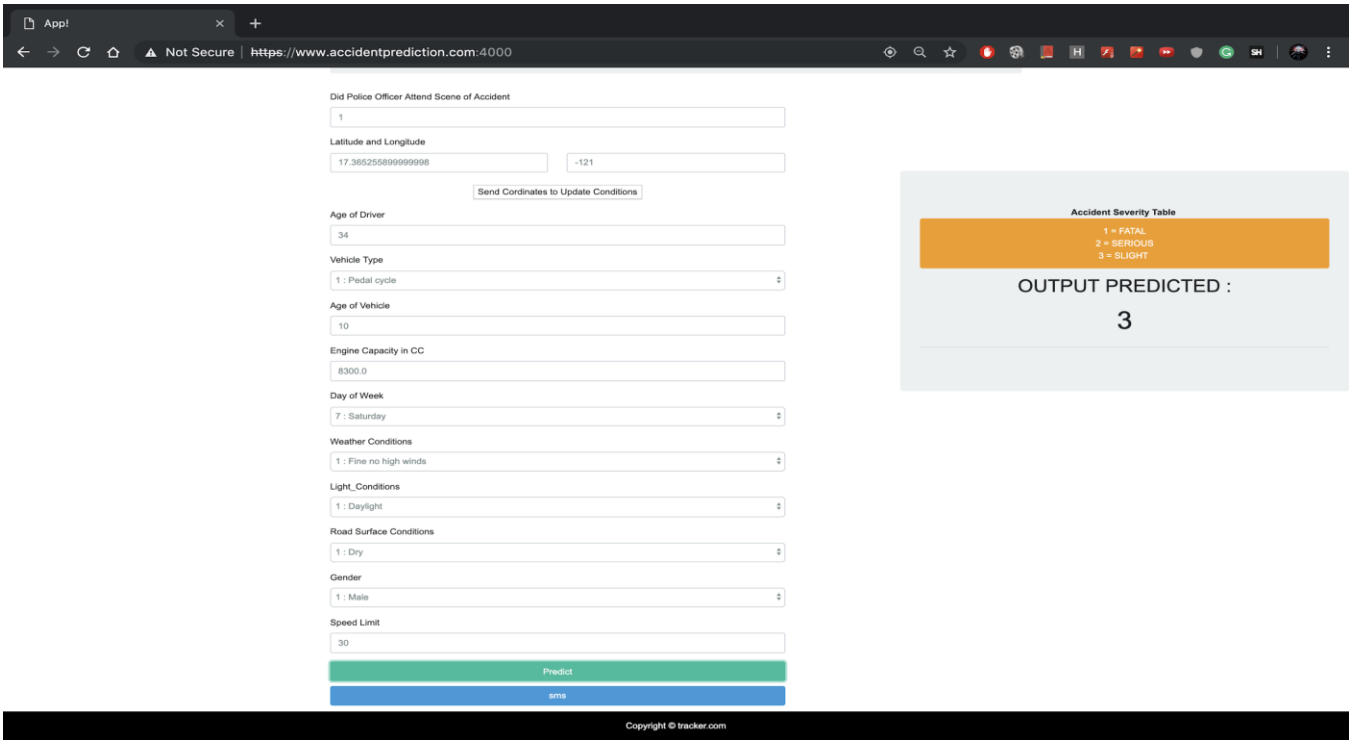


Figure 4.10 Output Predicted

Figure 4.10 shows all the data of the user. When the user clicks on Predict, that data is sent to the backend from where it is feeded into our chosen machine learning algorithm which is Random Forest. The output predicted is on the following basis of severity as 1- Fatal, 2- Severe, 3-Slight.

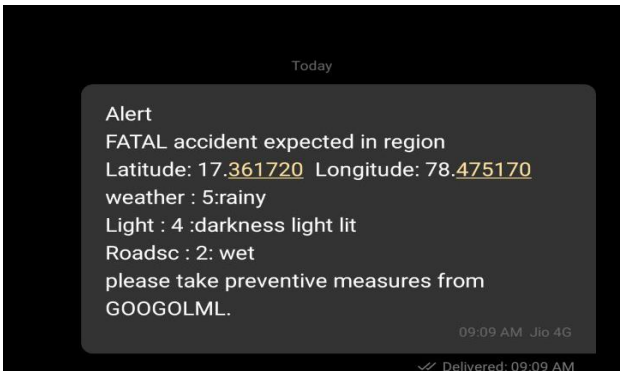


Figure 4.11 Click on sms button

In this Figure 4.11 an SMS is sent to the police with location details and severity. The TextLocal Api gives us 10 free messages to be sent every day.

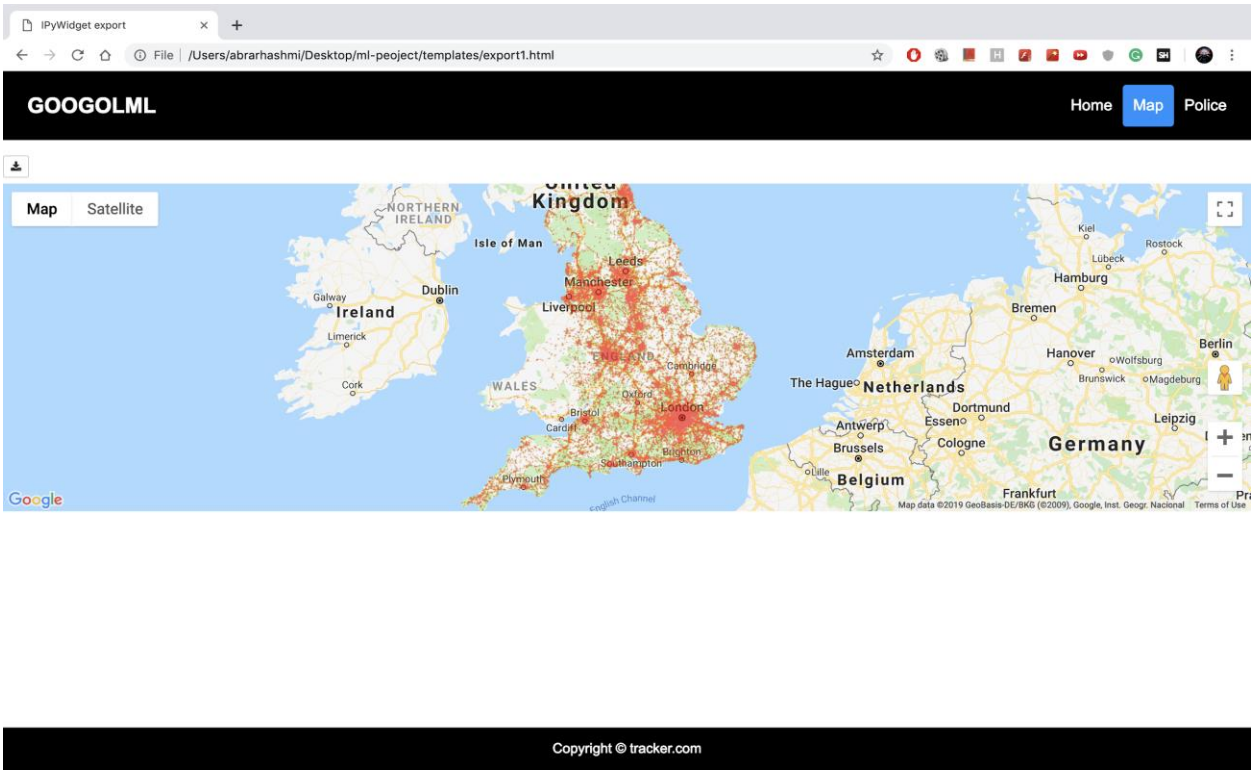


Figure 4.12 Map

In the Figure 4.12 This web page displays an interactive heat map for users. Darker points mean greater severity. The gmaps api is used to plot on google maps.

Figure 4.13 Visualization

The Figure 4.13 displays 4 images. It shows how different factors of the dataset affect the output.

For example in the image above about age of drivers, we can infer that the people in the ages 26-35 are more prone to have an accident. From such statistical data we were able to choose the factors from the dataset.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusions

This project aims at using Machine Learning classification techniques to predict severity of an accident at any particular location.

Machine Learning has enabled us to analyze meaningful data to provide solutions with a greater accuracy than with humans. We have built a model with a accuracy greater than 17% of the conventional system [1]. A web-based app using the most accurate algorithm has been developed which can be accessed through the domain name https://www.accidentprediction.com:4000.

This project can be used by governments to prevent accidents.

5.2 Future Work

With more resources, continuous prediction and alerts can be sent to the police for every location at regular intervals of time to take preventive measures. The web app can be incorporated with Google Maps which can be live tracked by the police. A fully-fledged web app for user and police interaction can be published for use in real-time. It can be used for Indian states or cities, if proper data of accidents is provided by the Indian Government.

REFERENCES

[1] Lu Wenqi, Luo Dongyu & Yan Menghua, "A Model of Traffic Accident Prediction" INSPEC Accession Number: 17239218 DOI: 10.1109/ICITE.2017.8056908

[2] Thineswaran Gunasegaran Yu-N Cheah, "Evolutionary Cross validation" INSPEC Accession Number: 17285520 DOI: 10.1109/ICITECH.2017.8079960

[3] Simon Bernard, Laurent Heutte and Sebastien Adam, "On the Selection of Decision Trees in Random Forests" INSPEC Accession Number: 10802866 DOI: 10.1109/IJCNN.2009.5178693

[4] Rafael G.Mantovan,, Ricardo Cerri, Joaquin Vanschoren, "Hyper-parameter Tuning of a Decision Tree Induction Algorithm" INSPEC Accession Number: 16651860 DOI: 10.1109/bracis.2016.018

[5] Fu Huilin, Zhou Yucai, "The Traffic Accident Prediction Based on Neural Network", 2011

[6] Lin, L., Wang, Q., Sadek, A.W., 2014. Data mining and complex networks algorithms for traffic accident analysis. In: Transportation Research Board 93rd Annual Meeting (No. 14-4172).

[7] Gunasegaran, T., & Cheah, Y.-N. (2017). Evolutionary cross validation. 2017 8th International Conference on Information Technology (ICIT). doi:10.1109/icitech.2017.8079960

[8] Bernard, S., Heutte, L., & Adam, S. (2009). On the selection of decision trees in Random Forests. 2009 International Joint Conference on Neural Networks. doi:10.1109/ijcnn.2009.5178693

[9] Matthew Bihis ; Sohini Roychowdhury, "A generalized flow for multi-class and binary classification tasks: An Azure ML approach"

[10] Abdel-Aty, M., N. Uddin, and A. Pande. Split Models for Predicting Multivehicle Collisions during High-Speed and Low-Speed Operating Conditions on Freeways. In Transportation Research Record: Journal of the Transportation Research Board, No. 1908, Transportation Research Board of the National Academies, Washington, D.C., 2005, pp. 51–58.

APPENDIX

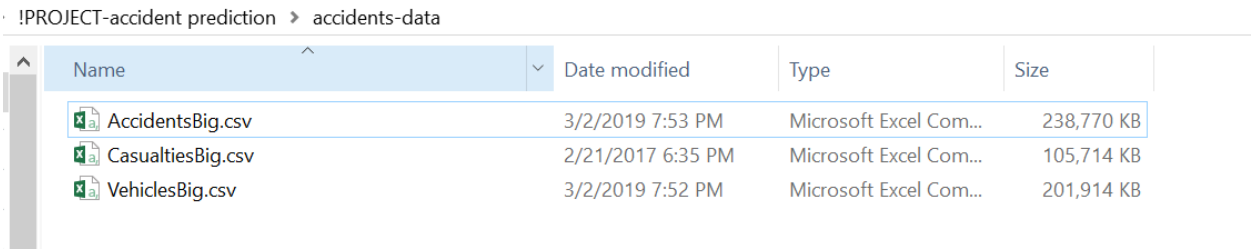
Importing Data Set

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.

In [2]:
```
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')
```

In [3]:
```
accidents = pd.read_csv('AccidentsBig.csv',index_col='Accident_Index')
casualties=pd.read_csv('CasualtiesBig.csv' , error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
vehicles=pd.read_csv('VehiclesBig.csv', error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
#general_info = pd.read_csv('ukTrafficAADF.csv')
```



Importing on cloud

```
[ ]  ## ---------METHOD 1 (FROM GITHUB)--------------------
     #!curl https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Accidents.csv
     # dataframe = pd.read_csv(!curl https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Accidents.csv)
     # dataframe.head()
     # import requests

     # url="https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Accidents.csv"
     # s=requests.get(url).content
     # c=pd.read_csv(s)
     #accidents = pd.read_csv('https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Accidents.csv',index_col='Accident_Index')
     #accidents = pd.read_csv('Accidents.csv',index_col='Accident_Index')
     #casualties= pd.read_csv('https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Casualties.csv' , error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
     #vehicles= pd.read_csv('https://raw.githubusercontent.com/abdulwahed786/final-yr-projectqA/master/Vehicles.csv', error_bad_lines=False,index_col='Accident_Index',warn_bad_lines=False)
     #general_info = pd.read_csv('ukTrafficAADF.csv')


     ## ---------METHOD 2 (FROM AZURE)--------------------
     # #importing data from azure workspace
     # from azureml import Workspace
     # ws = Workspace(
     #     workspace_id='f[redacted]',
     #     authorization_token='[redacted]',
     #     endpoint='https://studioapi.azureml.net'
     # )
     # ds = ws.datasets['Vehicles0515.csv']
     # frame = ds.to_dataframe()

     # from azureml import Workspace
     # ws = Workspace(
     #     workspace_id='[redacted]',
     #     authorization_token='[redacted]',
     #     endpoint='https://studioapi.azureml.net'
     # )
     # ds = ws.datasets['Accidents0515.csv']
     # accidents = ds.to_dataframe().set_index('Accident_Index')

     # dsc = ws.datasets['Casualties0515.csv']
     # casualties = dsc.to_dataframe()

     #accidents.set_index('Accident_Index')
     #accidents = pd.read_csv(frame,index_col='Accident_Index')
```

```
[ ]  # using python package TQDM to download dataset locally on colab
     # !pip install tqdm
     import requests
     import os
     from tqdm import tqdm
```

```
     Requirement already satisfied: tqdm in /usr/local/lib/python2.7/dist-packages (4.28.1)
```

In [3]:
```
import requests
import os
# !pip install tqdm
from tqdm import tqdm
# import pandas as pd
```

In [4]:
```
def download_dataset(file_url, name):
    r = requests.get(file_url, stream=True)

    with open(name, "wb") as file:
        for chunk in tqdm(r.iter_content(chunk_size=1024)):
            if chunk: file.write(chunk)

    print('Download complete.')
```

In [5]:
```
download_dataset("https://bitbucket.org/abdulwahed11314/accidents-data/raw/b7add9860d310171bca48bcaefeae37fe5157ac3/CasualtiesBig
download_dataset("https://bitbucket.org/abdulwahed11314/accidents-data/raw/b7add9860d310171bca48bcaefeae37fe5157ac3/AccidentsBig.
download_dataset("https://bitbucket.org/abdulwahed11314/accidents-data/raw/b7add9860d310171bca48bcaefeae37fe5157ac3/VehiclesBig.c
```

```
103368it [00:18, 5515.59it/s]
Download complete.
237031it [00:43, 5498.94it/s]
Download complete.
198729it [00:38, 5096.04it/s]
Download complete.
```

Checking Imported Data

In [4]:
```
accidents.head()
```

Out[4]:

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	

5 rows × 31 columns

```
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_Casualties	Date	Day_of_Week	...	Pedestrian_Crossing-Human_Control	Pedestrian_Crossi Physical_Facilit
200501BS00001	525680.0	178240.0	-0.191170	51.489096	1	2	1	1	04/01/2005	3	...	0	
200501BS00002	524170.0	181650.0	-0.211708	51.520075	1	3	1	1	05/01/2005	4	...	0	
200501BS00003	524520.0	182240.0	-0.206458	51.525301	1	3	2	1	06/01/2005	5	...	0	
200501BS00004	526900.0	177530.0	-0.173862	51.482442	1	3	1	1	07/01/2005	6	...	0	
200501BS00005	528060.0	179040.0	-0.156618	51.495752	1	3	1	1	10/01/2005	2	...	0	

5 rows × 31 columns

```
[ ] 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	Hit_Object_in_Carriageway	Vehicle_Leaving_Carriageway	Hit_Object_off_
200501BS00001	1	9	0	18	0	0	0	0	0	
200501BS00002	1	11	0	4	0	3	0	0	0	
200501BS00003	1	11	0	17	0	0	0	4	0	
200501BS00003	2	9	0	2	0	0	0	0	0	
200501BS00004	1	9	0	18	0	0	0	0	0	

5 rows × 21 columns

```
[ ] 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	Pedestrian_Location	Pedestrian_Movement	Car_Passenger	Bus_or_Coach_Passenger	P
200501BS00001	1	1	3	1	37	7	2	1	1	0	0	
200501BS00002	1	1	2	1	37	7	3	0	0	0	4	
200501BS00003	2	1	1	1	62	9	3	0	0	0	0	
200501BS00004	1	1	3	1	30	6	3	5	2	0	0	
200501BS00005	1	1	1	1	49	8	3	0	0	0	0	

Joining Of Tables

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

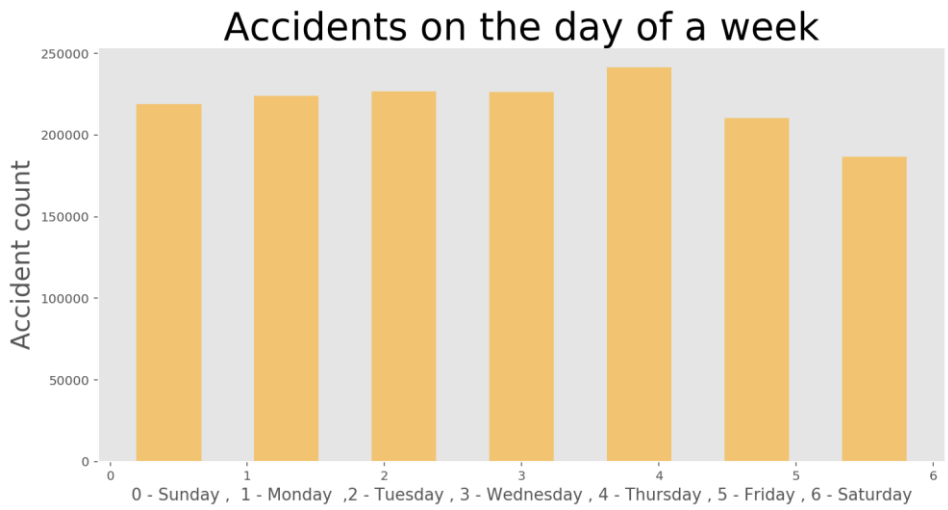
```
done joining
(3144481, 52)
```

Identifying Missing Values

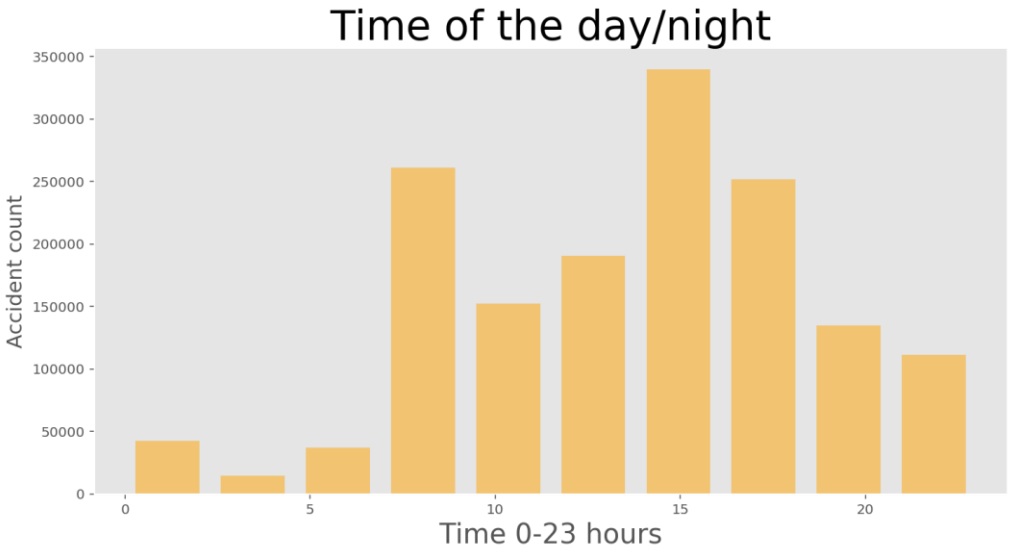
```
In [ ]: accidents.drop(['Location_Easting_OSGR', 'Location_Northing_OSGR','LSOA_of_Accident_Location',
                        'Junction_Control' ,'2nd_Road_Class'], axis=1, inplace=True)
        accidents['Date_time'] =  accidents['Date'] +' '+ accidents['Time']

        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

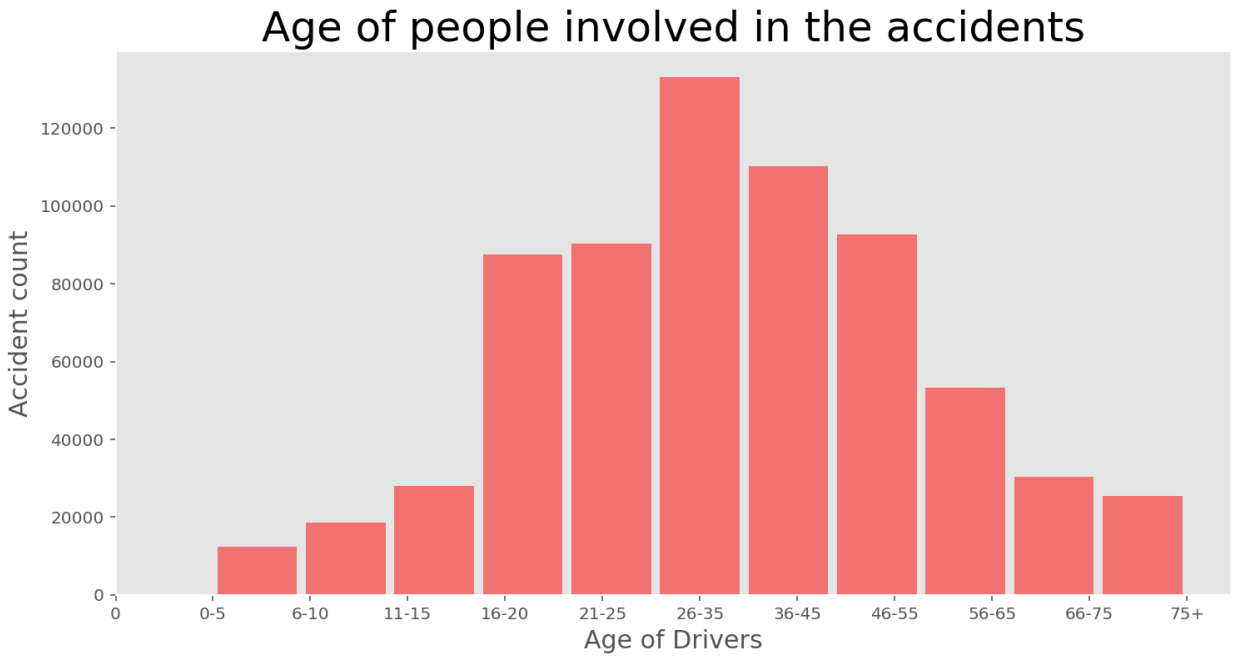
```
In [6]: plt.figure(figsize=(12,6))
        accidents.Date_time.dt.dayofweek.hist(bins=7,rwidth=0.55,alpha=0.5, color= 'orange')
        plt.title('Accidents on the day of a week' , fontsize= 30)
        plt.grid(False)
        plt.ylabel('Accident count' , fontsize = 20)
        plt.xlabel('0 - Sunday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



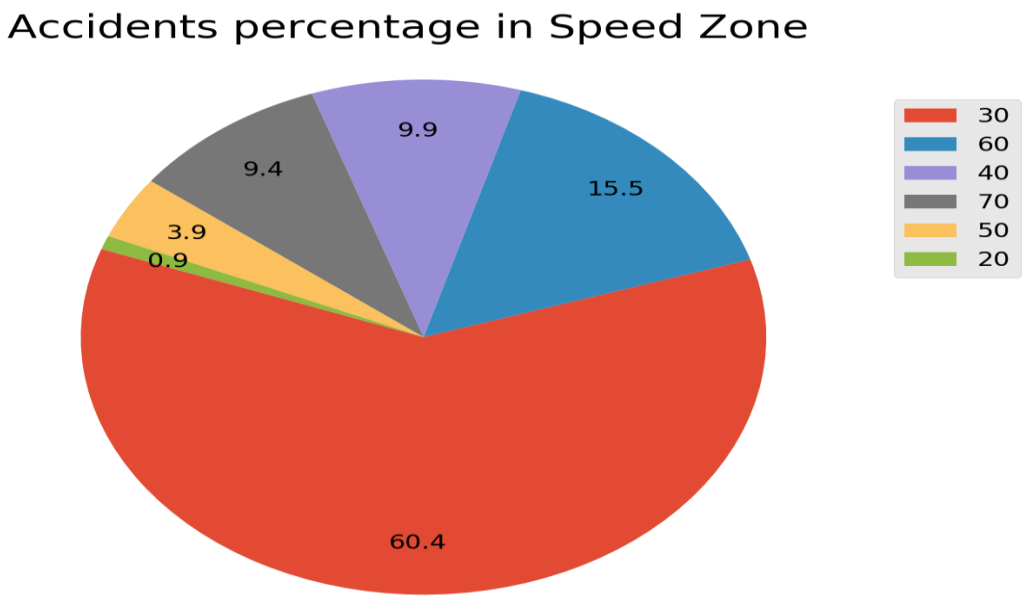
In [ ]:
```
plt.figure(figsize=(12,6))
accidents.Date_time.dt.hour.hist(rwidth=0.75,alpha =0.50, color= 'orange')
plt.title('Time of the day/night',fontsize= 30)
plt.grid(False)
plt.xlabel('Time 0-23 hours' , fontsize = 20)
plt.ylabel('Accident count' , fontsize = 15)
```



In [ ]:
```
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,)
```

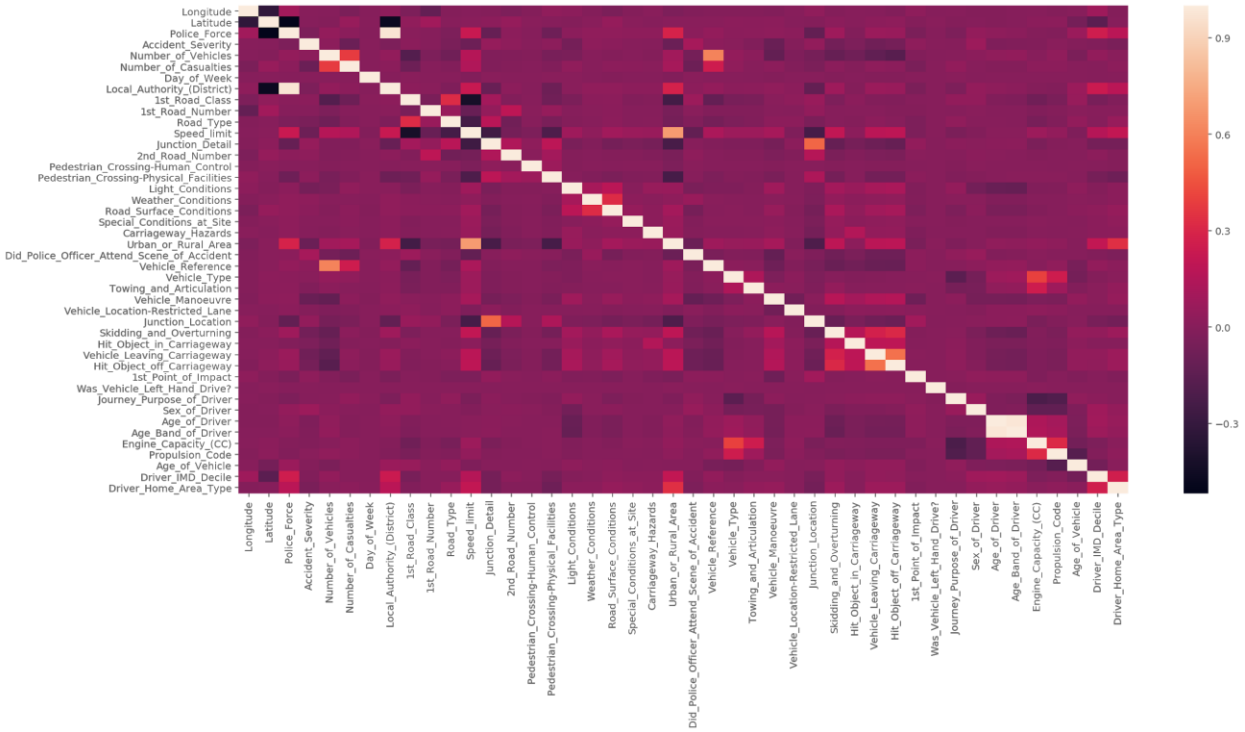


In [ ]:
```
speed_zone_accidents = accidents.loc[accidents['Speed_limit'].isin(['20' ,'30' ,'40' ,'50' ,'60' ,'70'])]
speed  = speed_zone_accidents.Speed_limit.value_counts()

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, startangle=160,textprops={'fontsize':

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()
```



Correlation between variables

```
In [ ]: corr =  accidents.corr()
        plt.subplots(figsize=(20,9))
        sns.heatmap(corr)
```



Plotting accidents Location on Google Maps

In [8]:
```
import gmaps
from ipywidgets.embed import embed_minimal_html
gmaps.configure(api_key='AIzaSyDFOjxJ23DfYRLTqEuNsgnqwP0E79Aybpk')

fig = gmaps.figure(center=(53.0, 1.0), zoom_level=6)
heatmap_layer = gmaps.heatmap_layer(accidents_2014_01[["Latitude", "Longitude"]],
                                    max_intensity=30,point_radius=5)
heatmap_layer = gmaps.heatmap_layer(accidents_2014_02[["Latitude", "Longitude"]],
                                    max_intensity=5,point_radius=3)
heatmap_layer = gmaps.heatmap_layer(accidents_2014_03[["Latitude", "Longitude"]],
                                    max_intensity=1,point_radius=1)
fig.add_layer(heatmap_layer)
fig
embed_minimal_html('export1.html', views=[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
print("done")
```

```
done
```

Normalize the Data

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

```
done
```

In [ ]:
```
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

```
In [ ]: 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']]

        accidents_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

```
[64] 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   0.862320  0.999996  0.926069    264697

   micro avg   0.862317  0.862317  0.862317    306959
   macro avg   0.287440  0.333332  0.308690    306959
weighted avg   0.743596  0.862317  0.798568    306959
```

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	1	264696	264697
All	1	306958	306959

Decision Tree

Decision Tree

```
[65] 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', 75.26)
              precision    recall  f1-score   support

           1   0.036793  0.046217  0.040970      4111
           2   0.158974  0.187780  0.172180     38151
           3   0.871137  0.844921  0.857829    264697

   micro avg   0.752550  0.752550  0.752550    306959
   macro avg   0.355635  0.359639  0.356993    306959
weighted avg   0.771451  0.752550  0.761672    306959
```

Predicted	1	2	3	All
Actual				
1	190	894	3027	4111
2	931	7164	30056	38151
3	4043	37006	223648	264697
All	5164	45064	256731	306959

Random Forest

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)

    print("done")
```

```
('Accuracy', 86.86)
              precision    recall  f1-score   support

           1   0.031496  0.002928  0.005358      1366
           2   0.195915  0.040622  0.067291     20777
           3   0.884926  0.979143  0.929653    166321

   micro avg   0.868601  0.868601  0.868601    188464
   macro avg   0.370779  0.340898  0.334101    188464
weighted avg   0.802781  0.868601  0.827884    188464

done
```

Hyperparameters tuning for the models

Hyperparameters tuning for the models

Logistic Regression with Hyperparameter tuning

```
[66] 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   0.862317  0.999974  0.926058    264697

   micro avg   0.862298  0.862298  0.862298    306959
   macro avg   0.287439  0.333325  0.308686    306959
weighted avg   0.743594  0.862298  0.798558    306959
```

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	7	264690	264697
All	7	306952	306959

Decision Tree hyperparameters tuning

All we are going to do is find the best values for mininum sample leaf and maximum features to get the best score.

```
[67] 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', 85.71)
              precision    recall  f1-score   support

           1   0.071429  0.000730  0.001445      4111
           2   0.323387  0.045451  0.079700     38151
           3   0.866655  0.987333  0.923066    264697

   micro avg   0.857056  0.857056  0.857056    306959
   macro avg   0.420490  0.344504  0.334737    306959
weighted avg   0.788483  0.857056  0.805904    306959
```

Predicted	1	2	3	All
Actual				
1	3	301	3807	4111
2	13	1734	36404	38151
3	26	3327	261344	264697
All	42	5362	301555	306959

Random Forest Hyperparameter tuning

First, we will see the default parameters of the random forest model before we tune the parameters.

```
[68] random_forest.get_params()
```

```
{'bootstrap': True,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 200,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

We will implement the grid search using sklearn library.

```
from sklearn.model_selection import RandomizedSearchCV
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [4, 5],
    'min_samples_leaf': [5, 10, 15],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300]
}
# Create a based model
random_f = RandomForestClassifier()
# Instantiate the grid search model
grid_search = RandomizedSearchCV(estimator = random_f, param_distributions = param_grid,
```

Connecting to GOOGLE DRIVE and saving the model

```
[ ] #connecting to GOOGLE DRIVE and saving the model
    !apt-get install -y -qq software-properties-common python-software-properties module-init-tools
    !add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
    !apt-get update -qq 2>&1 > /dev/null
    !apt-get -y install -qq google-drive-ocamlfuse fuse
    from google.colab import auth
    auth.authenticate_user()
    from oauth2client.client import GoogleCredentials
    creds = GoogleCredentials.get_application_default()
    import getpass
    !google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
    vcode = getpass.getpass()
    !echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
    !mkdir -p drive
    !google-drive-ocamlfuse drive


    print("done connecting to google drive")
```

```
E: Package 'python-software-properties' has no installation candidate
Selecting previously unselected package google-drive-ocamlfuse.
(Reading database ... 131323 files and directories currently installed.)
Preparing to unpack .../google-drive-ocamlfuse_0.7.1-0ubuntu3~ubuntu18.04.1_amd64.deb ...
Unpacking google-drive-ocamlfuse (0.7.1-0ubuntu3~ubuntu18.04.1) ...
Setting up google-drive-ocamlfuse (0.7.1-0ubuntu3~ubuntu18.04.1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Please, open the following URL in a web browser: https://accounts.google.com/o/oauth2/auth?client_id=[redacted].apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Ao
.........
Please, open the following URL in a web browser: https://accounts.google.com/o/oauth2/auth?client_id=[redacted].apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Ao
Please enter the verification code: Access token retrieved correctly.
done connecting to google drive
```

Loading the model

```
[ ]  from sklearn.externals import joblib
     modelfile="drive/litemodel.sav"
     joblib.dump(random_forest,modelfile)
```

```
['drive/litemodel.sav']
```

```
# load the model from drive
loaded_model= joblib.load(modelfile)
# result=loaded_model.score(X_test, y_test)
# print(result)
loaded_model
print("loaded model")
```

```
1.0
loaded model
```

Main.py Flask

```
from flask import Flask, render_template, request
import pandas as pd
from sklearn.externals import joblib
import numpy as np
import urllib.request
import urllib.parse

app = Flask(__name__)
model = joblib.load('litemodel.sav')

def sendSMS(apikey, numbers, sender, message):
    data =  urllib.parse.urlencode({'apikey': apikey, 'numbers': numbers,'message' : message, 'sender': sender})
    data = data.encode('utf-8')
    request = urllib.request.Request("https://api.textlocal.in/send/?")
    f = urllib.request.urlopen(request, data)
    fr = f.read()
    return(fr)
```

```
def cal(ip):
    input = dict(ip)
    Did_Police_Officer_Attend = input['Did_Police_Officer_Attend'][0]
    age_of_driver = input['age_of_driver'][0]
    vehicle_type = input['vehicle_type'][0]
    age_of_vehicle = input['age_of_vehicle'][0]
    engine_cc = input['engine_cc'][0]
    day = input['day'][0]
    weather = input['weather'][0]
    light = input['light'][0]
    roadsc = input['roadsc'][0]
    gender = input['gender'][0]
    speedl = input['speedl'][0]

    data = np.array([Did_Police_Officer_Attend, age_of_driver, vehicle_type, age_of_vehicle,
engine_cc, day, weather, roadsc, light, gender, speedl])

    print("logging",data)
    data = data.astype(float)
    data = data.reshape(1, -1)
    x = np.array([1, 3.73, 3, 0.69, 125, 4, 1, 1, 1, 1, 30]).reshape(1, -1)
    try: result = model.predict(data)
    except Exception as e: result = str(e)
    return str(result[0])

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/visual/', methods=['GET'])
```

```
def visual():
    return render_template('visual.html')

@app.route('/sms/', methods=['POST'])
def sms():
    res=cal(request.form)
    try:
        resp = sendSMS('UwYs16dD3zM-DKuzZKQYolAJkoba1j0BmRGompsNRs',
'9618205648', 'TXTLCL', 'Severe acceident')
        print (resp)
    except Exception as e: print(e)
    return res

@app.route('/', methods=['POST'])
def get():
    return cal(request.form)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, port=4000)
```