Accident Severity Prediction Model

Major Project Report

In partial fulfillment of the requirements for the award of the Degree of B.E in Computer Science and Engineering .

By

Kasturi Nischitha	160619733086
Pasham Akshatha Sai	160619733102
Tandra Hyde Celestia	160619733113

Under the Guidance of

Mrs. D. Radhika

Assistant professor, Department of Computer Science & Engineering

Department of Computer Science and Engineering

Stanley College of Engineering and Technology for Women

Chapel Road, Abids, Hyderabad- 500001

(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NAAC with 'A' Grade and NBA)

2023

Stanley College of Engineering and Technology for Women

Chapel Road, Abids, Hyderabad- 500001

(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NAAC with 'A' Grade and NBA)

CERTIFICATE

This is to verify that Major project report entitled **Accident Severity Prediction Model**

being submitted

BY

Kasturi Nischitha	160619733086
Pasham Akshatha Sai	160619733102
Tandra Hyde Celestia	160619733113

In partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science & Engineering to the Osmania University, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Guide

Mrs.D.Radhika

Assistant professor, Dept. of CSE

Head of the Department

Dr.Y.V.S Sai Pragathi

Ph.D Professor, Dept. of CSE

Project Coordinators

Dr.Y V S Sai Pragathi,

Professor & HOD Department of CSE

A.Tejaswi, Assistant professor Department of CSE

External Examiner

DECLARATION

We hereby declare that major project work entitled **Accident Severity Prediction Model** submitted to the Osmania University, Hyderabad, is a record of original work done by us. This project work is submitted in partial fulfillment of the requirements for the award of the degree of the B.E in Computer Science and Engineering.

Kasturi Nischitha 160619733086

Pasham Akshatha Sai 160619733102

Tandra Hyde Celestia 160619733113

ACKNOWLEDGMENT

The completion of any group project requires a collective contribution of its members and supportive mentors along with other faculty. It is here we would like to acknowledge their efforts and support. We consider ourselves fortunate enough to express gratitude and respect to all those who have aided us in execution.

Firstly, we would like to thank all the professors in the Department of Computer Science who have contributed to our cumulative knowledge and shaped our personalities without which the project wouldn't have seen the light of day.

We wish to express our sincere thanks to Sri Kodali Krishna Rao, Correspondent and Secretary of the college, for providing us with all necessary facilities.

We place on record, our sincere gratitude to Dr. Satya Prasad Lanka, Principal, for his constant encouragement.

We deeply express our sincere thanks to our Head of the Department, Dr.Y.V.S.Sai Pragathi, for encouraging and allowing us to present the Major Project on the topic "**Accident Severity Prediction Model"** at our department premises for the partial fulfillment of the requirements leading to the award of the B.E Degree.

We would specifically like to express our gratitude to our beloved mentor Mrs.D.Radhika for her support and guidance all along the course of this project. Her passionate encouragement had made us push hard to meet the deadline and expand our skillset. Without her, the project wouldn't have been a reality.

We would also like to thank Ms.A.Tejaswi for providing us with a conducive environment to work freely.

ABSTRACT.

With the exponentially increasing number of vehicles, road safety is a matter of huge concern. Road accidents kill 1.2 million people every year. In 2022, there have been 3967 accidents with injuries reported in Hyderabad alone. It causes loss of lives and economical damage, due to which is a serious concern which needs to be solved.

We used Machine Learning algorithms to predict the severity of an accident occurring at a particular location and time. Factors like speed limit, age, weather, vehicle type, light conditions and day of the week have been used as parameters for training the model. We have used the road accident data provided by the government of UK from 2011-2020. The dataset has 1.2 million records of which 80% is used to train the model and 20% to test it. We have chosen Random Forest for our Machine Learning model as it showed the highest accuracy of 86.86%. User data at a specific time will be used to predict the severity of a road accident at the given location. The severity metrics are 1= Fatal, 2= Serious, 3= Slight.

We used Machine Learning tools such as Python, Scikit-Learn, Numpy, Matplotlib etc. We have created a web app for user input and output display. The front end takes the input from the user and sends it to the backend where the Machine Learning model is deployed. The model will run the with the input data and predicts the severity of an accident occurring. We have bought a custom domain name for the web app so that it can be easily accessible by anyone. This model will play an important role in planning and management of traffic and would help us reduce a lot of road accidents in the future.

TABLE OF CONTENTS

5. CONCLUSION

REFERENCES

SOURCE CODE

List of Tables

List Of Figures.

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.

Accident-prone travel is becoming an important factor for governments now days because of their frequency, traffic accidents are a major cause of death globally, cutting short millions of lives per year. Therefore, a system that can predict the occurrence of traffic accidents or accident-prone areas can potentially save lives. Accidents don't arise in a purely stochastic manner; their occurrence is influenced by a multitude of factors such as drivers' physical conditions, car types, driving speed, traffic condition, road structure and weather. Studying historical accident records would help us understand the relationships between these factors and road accidents, which would in turn allow us to build an accident predictor. Using legacy methods to predict the risk was tough question because of the difficulty in prediction factors and methods. In order to mitigate the traditional way problems this topic proposes new method to predict the road accident severities. The method can distinguish every individual parameter like Age of driver, vehicle type, sex of driver, age of vehicle, speed limit etc. So, by treating these values specifically which will bring positive excitation and encourage every person to pay more attention to their behaviour, which has a significant influence on improvement of the whole society. However, for the past few decades, limited by computing methods, researchers can only use basic models to analyze risk factors, which is helpful but not accurate enough. Traditional models cannot fully utilize the potential information of data and factors. Therefore, we consider applying machine learning methods in risk prediction, which helps us improve the accuracy of our model and pursue further latent risk factors to perfect the model.

In this proposal we are trying to build a practical model to evaluate the accident severity and accident-prone areas based road accident data using machine learning methods. We select some representative features from behaviour data and build a high-accuracy model to predict the possibility of vehicle violations. The model can be applied to help predict the accident severity and accident-prone areas, and make more contribution to the improvement of transportation environment.

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 Statement:

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 2022, there have been 3967 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:

We built a ML powered web app which predicts accidents severity based on the current conditions. The model is trained with 1.6 million accident records over 20011-2020 which provides greater accuracy.

The purpose of this model is to be able to predict which conditions will be more prone to accidents, and therefore take preventive measures. Among all the available classification methods, random forest is used as it provides the highest accuracy and can also handle big data with numerous variables running into thousands. We tried to provide the best possible prediction using the crucial features.

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.

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. 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. Karlaftis et al used hierarchical tree-based regression revisits the relationship between rural road geometric characteristics, accident rates and their prediction. Lee et al develop a probabilistic model relating significant crash precursors to changes in crash potential. Abdel 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, 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 nonlinear phenomenon using distributed and hierarchical feature representation.

2.1 Prediction Factors:

Table 2. Prediction Factors.

Day_of_Week: Numeric: 1 for Sunday, 2 for Monday, and so on.
Light_Conditions: Day, night, street lights or not
Weather_Conditions: Wind, rain, snow, fog.
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. Day of week:

label
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

Table 4.

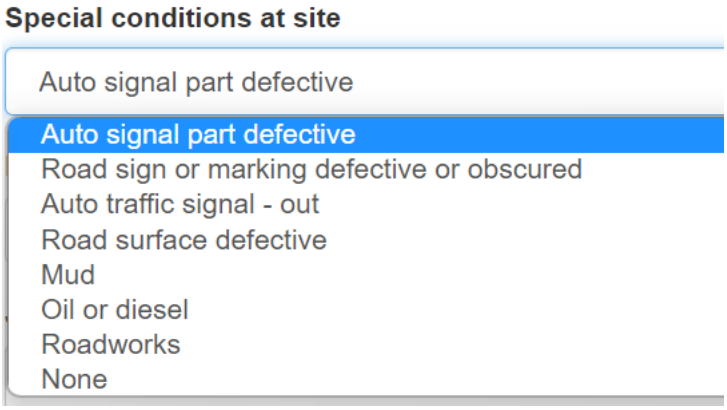


Table 5.

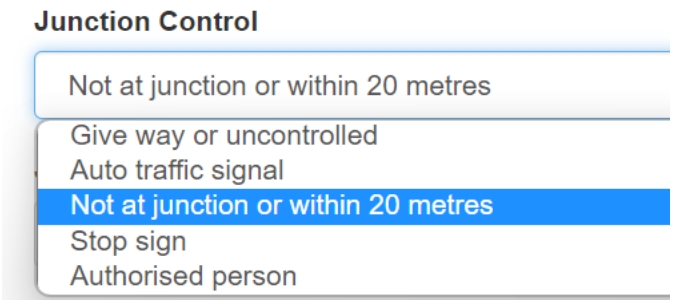


Table 6.

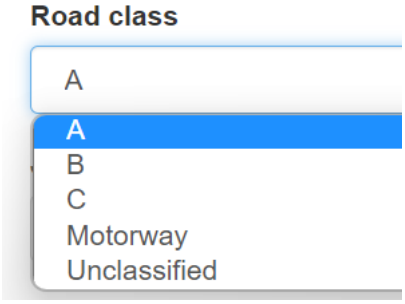


Table 7.

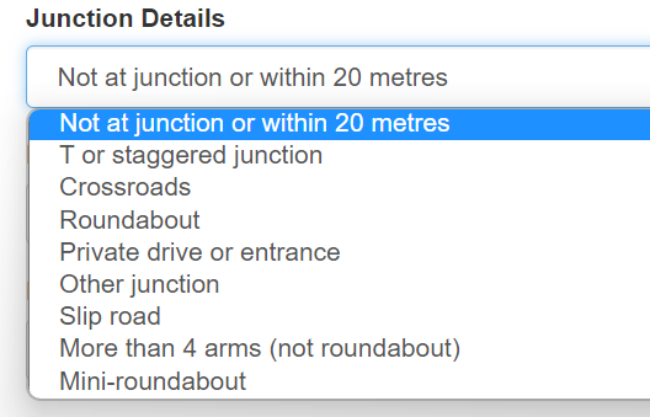


Table 8

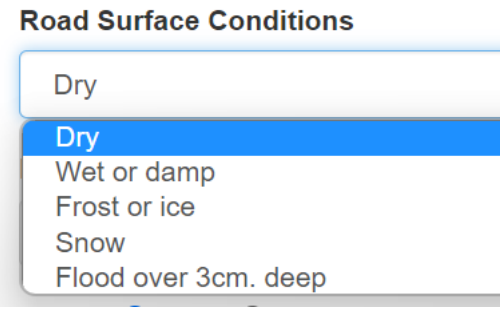


Table 9.

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 Random Forest:

A Random Forest Algorithm is a supervised machine learning algorithm which is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

Working of Random Forest Algorithm:

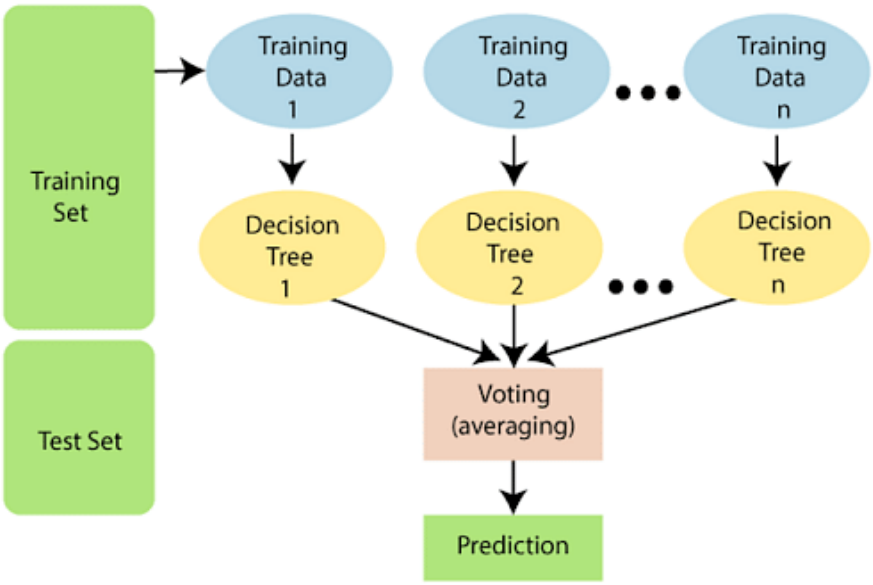


Figure 1: Working of Random Forest

The following steps explain the working Random Forest Algorithm:

Step 1: Select random samples from a given data or training set.

Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

Step 4: Finally, select the most voted prediction result as the final prediction result.

2.4 Advantage of using Random Forest Algorithm:

There are a lot of benefits to using Random Forest Algorithm, but one of the main advantages is that it reduces the risk of overfitting and the required training time. Additionally, it offers a high level of accuracy. Random Forest algorithm runs efficiently in large databases and produces highly accurate predictions by estimating missing data.

2.5 Evaluation Metrics:

Metrics are used to measure the performances of the proposed approach to predict road accident training set Namely: accuracy, specificity, precision, recall and F1 score.

Accuracy=TP+TN / TP+FP+FN+TN
Recall=TP / TP+FN
Specificity=TN / FP+TN
Precision=TP / TP+FP
F1Score=2∗(Precision∗Recall)/(Precision+Recall)
TP: it shows predictive is positive and it is normally true
TN: it implies predictive is Negative and it is normally True
FP: denotes predictive is positive and it is normally false
FN: represents predictive is negative and it is false.
Where TP implies true positive, TN denotes true negative, FP indicates false positive, and FN denotes false negative. in the actual study values are represented by true and false whereas predictive values denoted by positive and negative.

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.

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.

3.1 System Architecture:

Describes the data flow in a diagrammatic representation.

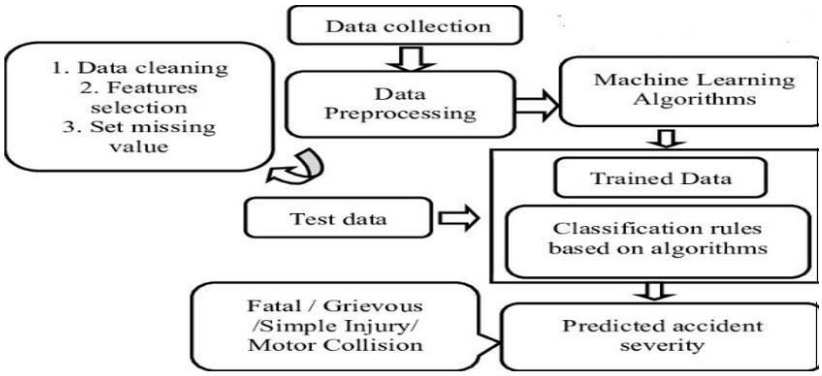


Figure 2: Architecture

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): User input is taken for other parameters like age, sex etc. Machine Shows the predicted accident Severity on the Screen

3. The back end: The input details are feeded to the model and severity is predicted.

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 multidimensional 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 threedimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image.

3.3.3 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

3.4 Implementation of Proposed solution:

There are four important steps:

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

3.4.1 Data Importing:

We import a file to perform analysis on this data. We can use general traffic information data for machine learning part.

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

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import metrics
        from sklearn.metrics import classification_report, confusion_matrix

        df = pd.read_csv('Accident_Information.csv', sep=',')
```

```
In [3]: accidents.head()
```

Out[3]:

Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles	Number_of_Casua
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

```
In [2]: encoding = {
        "Carriageway_Hazards": {"None": 0,
                                "Other object on road": 1,
                                "Any animal in carriageway (except ridden horse)": 1,
                                "Pedestrian in carriageway - not injured": 1,
                                "Previous accident": 1,
                                "Vehicle load on road": 1,
                                "Data missing or out of range": 0  }
        }
        df.replace(encoding, inplace=True)
        print(df['Carriageway_Hazards'].value_counts())
```

```
0    2010553
1      36703
Name: Carriageway_Hazards, dtype: int64
```

In [3]:
```
print(df['Light_Conditions'].value_counts())
encoding_light = {"Light_Conditions": {"Daylight": 0,
                                       "Darkness - lights lit": 1,
                                       "Darkness - no lighting": 1,
                                       "Darkness - lighting unknown": 1,
                                       "Darkness - lights unlit": 1,
                                       "Data missing or out of range": 0}}
df.replace(encoding_light, inplace=True)
print(df['Light_Conditions'].value_counts())
```

```
Daylight                          1496121
Darkness - lights lit              404144
Darkness - no lighting             112644
Darkness - lighting unknown         24362
Darkness - lights unlit              9971
Data missing or out of range           14
Name: Light_Conditions, dtype: int64
0    1496135
1     551121
Name: Light_Conditions, dtype: int64
```

In [4]:
```
print(df['Day_of_Week'].value_counts())
encoding_day_of_week = {"Day_of_Week": {"Saturday": 1,
                                        "Sunday": 1,
                                        "Monday": 0,
                                        "Tuesday": 0,
                                        "Wednesday": 0,
                                        "Thursday": 0,
                                        "Friday": 0}}
df.replace(encoding_day_of_week, inplace=True)
print(df['Day_of_Week'].value_counts())
```

```
Friday       335183
Wednesday    308580
Thursday     308240
Tuesday      306292
Monday       290482
Saturday     273152
Sunday       225327
Name: Day_of_Week, dtype: int64
0    1548777
1     498479
Name: Day_of_Week, dtype: int64
```

In [5]:
```
print(df['Special_Conditions_at_Site'].value_counts())
encoding_Special_Conditions_at_Site = {"Special_Conditions_at_Site": {"None": 0,
                                                                      "Roadworks": 1,
                                                                      "Oil or diesel": 1,
                                                                      "Mud": 1,
                                                                      "Road surface defective": 1,
                                                                      "Auto traffic signal - out": 1,
                                                                      "Road sign or marking defective or obscured": 1,
                                                                      "Auto signal part defective": 1,
                                                                      "Data missing or out of range": 0}}
df.replace(encoding_Special_Conditions_at_Site, inplace=True)
print(df['Special_Conditions_at_Site'].value_counts())
```

```
None                                          1995137
Roadworks                                       23525
Oil or diesel                                    6797
Mud                                              6363
Road surface defective                           4801
Auto traffic signal - out                        3855
Road sign or marking defective or obscured       2930
Data missing or out of range                     2835
Auto signal part defective                       1013
Name: Special_Conditions_at_Site, dtype: int64
0    1997972
1      49284
Name: Special_Conditions_at_Site, dtype: int64
```

In [6]:
```
encoding_1st_road_class = {"1st_Road_Class": {"A": 1, "A(M)": 1, "B": 2, "C": 3, "Motorway": 4, "Unclassified": 1}}
df.replace(encoding_1st_road_class, inplace=True)
df['1st_Road_Class'].value_counts()
```

Out[6]:
```
1    1536156
2     258076
3     174953
4      78071
Name: 1st_Road_Class, dtype: int64
```

In [7]:
```
#replacing 'Data missing or out of range' with most occured value 'Give way or uncontrolled'
df['Junction_Control'] = df['Junction_Control'].replace(['Data missing or out of range'], 'Give way or uncontrolled')
```

In [8]:
```
df['Junction_Control'].value_counts()
```

Out[8]:
```
Give way or uncontrolled              1742624
Auto traffic signal                    211335
Not at junction or within 20 metres     77304
Stop sign                               12333
Authorised person                        3660
Name: Junction_Control, dtype: int64
```

In [9]:
```
encoding_junction_detail = {"Junction_Control":
                            {"Give way or uncontrolled": 1,
                             "Auto traffic signal": 2,
                             "Not at junction or within 20 metres": 3,
                             "Stop sign": 4,
                             "Authorised person": 5,
                              }}
df.replace(encoding_junction_detail, inplace=True)
df['Junction_Control'].value_counts()
```

Out[9]:
```
1    1742624
2     211335
3      77304
4      12333
5       3660
Name: Junction_Control, dtype: int64
```

In [10]:
```
encoding_junction_detail = {"Junction_Detail":
                            {"Not at junction or within 20 metres": 1,
                             "T or staggered junction": 2,
                             "Crossroads": 3,
                             "Roundabout": 4,
                             "Private drive or entrance": 5,
                             "Other junction": 6,
                             "Slip road": 7,
                             "More than 4 arms (not roundabout)": 8,
                             "Mini-roundabout": 9,
                             "Data missing or out of range": 1 }}
df.replace(encoding_junction_detail, inplace=True)
df['Junction_Detail'].value_counts()
```

Out[10]:
```
1    827957
2    635349
3    196283
4    177214
5     72751
6     59692
7     30052
8     25551
9     22407
Name: Junction_Detail, dtype: int64
```

In [11]:
```
encoding_road_surface_cond = {"Road_Surface_Conditions":
                              {"Dry": 1,
                               "Wet or damp": 2,
                               "Frost or ice": 3,
                               "Snow": 4,
                               "Flood over 3cm. deep": 5,
                               "Data missing or out of range": 1 }}
df.replace(encoding_road_surface_cond, inplace=True)
df['Road_Surface_Conditions'].value_counts()
```

Out[11]:
```
1    1423360
2     568563
3      40321
4      12167
5       2845
Name: Road_Surface_Conditions, dtype: int64
```

In [12]:
```
encoding_road_type = {"Road_Type":
                              {"Single carriageway": 1,
                               "Dual carriageway": 2,
                               "Roundabout": 3,
                               "One way street": 4,
                               "Slip road": 5,
                               "Unknown": 0,
                               "Data missing or out of range": 1 }}
df.replace(encoding_road_type, inplace=True)
df['Road_Type'].value_counts()
```

Out[12]:
```
1    1527883
2     303407
3     136754
4      43258
5      21558
0      14396
Name: Road_Type, dtype: int64
```

In [13]:
```
encoding_urban_rural = {"Urban_or_Rural_Area":
                              {"Urban": 1,
                               "Rural": 2,
                               "Unallocated": 1 }}
df.replace(encoding_urban_rural, inplace=True)
df['Urban_or_Rural_Area'].value_counts()
```

Out[13]:
```
1    1322499
2     724757
Name: Urban_or_Rural_Area, dtype: int64
```

```
In [14]: encoding_weather = {"Weather_Conditions":
                             {"Fine no high winds": 1,
                              "Raining no high winds": 2,
                              "Raining + high winds": 3,
                              "Fine + high winds": 4,
                              "Snowing no high winds": 5,
                              "Fog or mist": 6,
                              "Snowing + high winds": 7,
                              "Unknown": 1,
                              "Other": 1,
                              "Data missing or out of range": 1 }}
         df.replace(encoding_weather, inplace=True)
         df['Weather_Conditions'].value_counts()

Out[14]: 1    1726874
         2     239281
         3      28343
         4      25816
         5      13387
         6      11068
         7       2487
         Name: Weather_Conditions, dtype: int64
```

3.4.2 Preprocessing of Data

Data Cleaning

Here 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 investigate 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.

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')
```

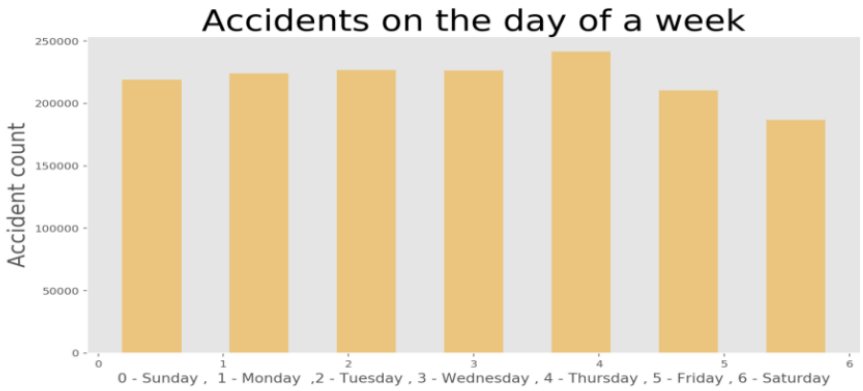


Figure 3: Accidents on the day of week

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')
```

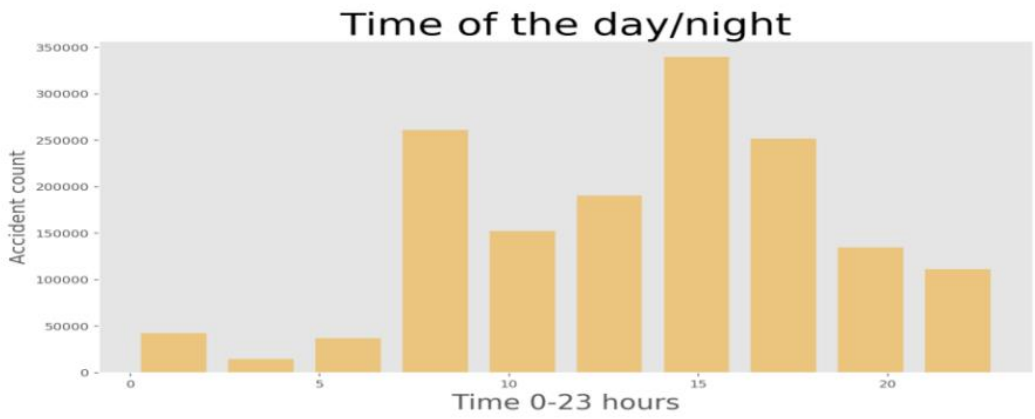


Figure 4: Time of the day/night Accident happened

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')
```

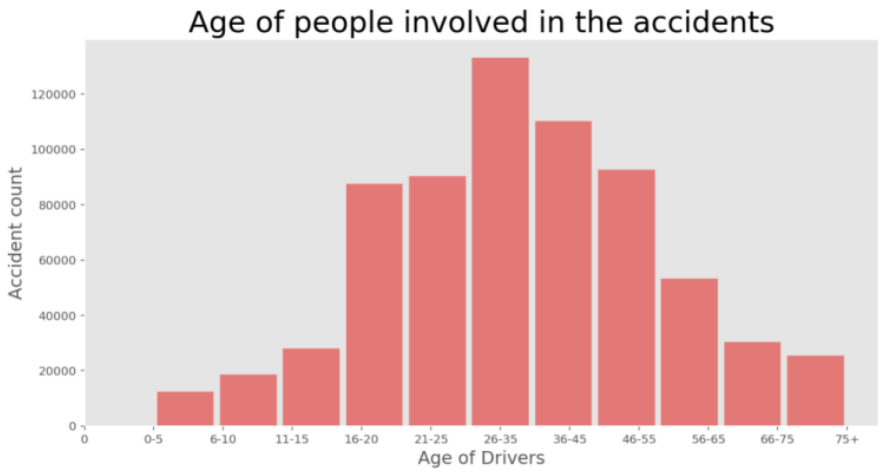


Figure 5: Age of People involved in the Accidents

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.

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()
```

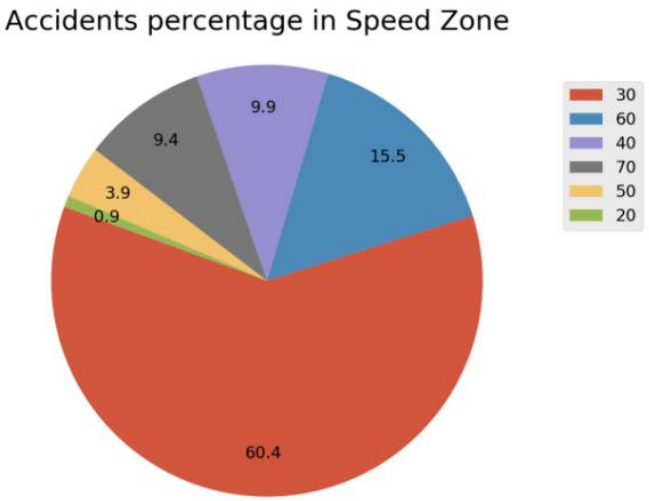


Figure 6: Speed of Cars where most Frequent Accidents Occurred

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.

Graphical Representation of Number of severity Data:

```
In [129]: sns.countplot("Accident_Severity",data=df_org)
Out[129]: <matplotlib.axes._subplots.AxesSubplot at 0x1118e4400>
```

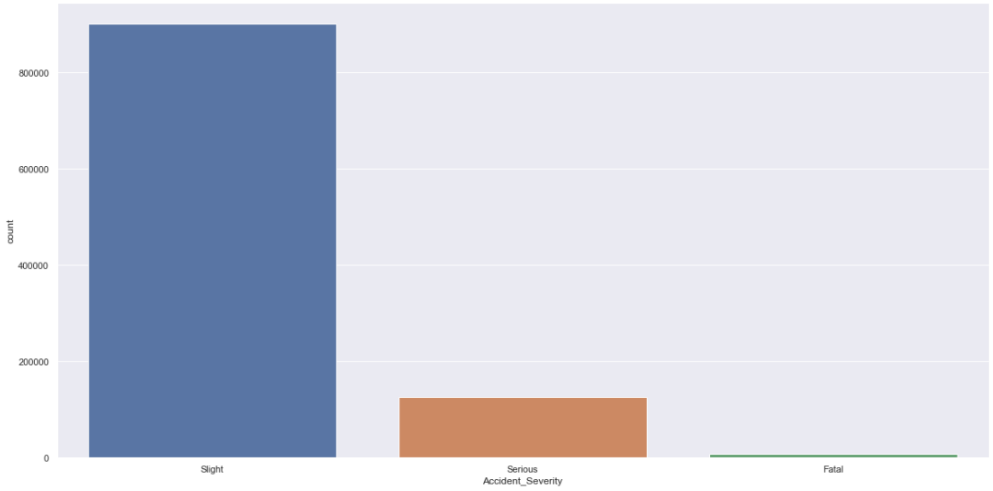


Figure 7: Graphical Representation of No. of Slight, Serious, Fatal Data present in the dataset

3.4.3 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
```

Figure 8: 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.

```
XY_Serious = XY[XY["Accident_Severity"]=="Serious"]
        XY_Slight = XY[XY["Accident_Severity"]=="Slight"]
        XY_Fatal = XY[XY["Accident_Severity"]=="Fatal"]

        X_Serious = XY_Serious[self.cols]
        Y_Serious = XY_Serious[['Accident_Severity']]
        X_Serious_train, X_Serious_test, Y_Serious_train, Y_Serious_test =
train_test_split(X_Serious, Y_Serious, test_size=0.25)

        X_Slight = XY_Slight[self.cols]
        Y_Slight = XY_Slight[['Accident_Severity']]
        X_Slight_train, X_Slight_test, Y_Slight_train, Y_Slight_test =
train_test_split(X_Slight, Y_Slight, test_size=0.25)

        X_Fatal = XY_Fatal[self.cols]
        Y_Fatal = XY_Fatal[['Accident_Severity']]
        X_Fatal_train, X_Fatal_test, Y_Fatal_train, Y_Fatal_test =
train_test_split(X_Fatal, Y_Fatal, test_size=0.25)
```

Figure 9: Training and Testing the Data

Algorithms and Techniques

Algorithms implemented with accuracy and confusion matrix

Random Forest: with Accuracy 0.6304064956019362

```
Accuracy: 0.6166137511060222
[[6150 4128]
 [3238 5697]]
              precision    recall  f1-score   support

     Serious       0.66      0.60      0.63     10278
      Slight       0.58      0.64      0.61      8935

    accuracy                           0.62     19213
   macro avg       0.62      0.62      0.62     19213
weighted avg       0.62      0.62      0.62     19213
```

Figure 10: Confusion Matrix of Random Forest

Logistic Regression: with Accuracy 0.6250975901733201

```
[[6759 3519]
 [3684 5251]]
              precision    recall  f1-score   support

     Serious       0.65      0.66      0.65     10278
      Slight       0.60      0.59      0.59      8935

    accuracy                           0.63     19213
   macro avg       0.62      0.62      0.62     19213
weighted avg       0.62      0.63      0.62     19213
```

Figure 11: Confusion Matrix of Logistic Regression

K-Neighbours Classifier: with Accuracy 0.5757559985426534

```
[[6422 3856]
 [4295 4640]]
              precision    recall  f1-score   support

     Serious       0.60      0.62      0.61     10278
      Slight       0.55      0.52      0.53      8935

    accuracy                           0.58     19213
   macro avg       0.57      0.57      0.57     19213
weighted avg       0.57      0.58      0.57     19213
```

Figure 12: Confusion Matrix of K-Neighbours Classifier

3.5 System Requirements

1. Language Used: Python, JavaScript
2. IDE and Framework: Jupyter Notebook, Sublime, Flask
3. Windows XP, 7, 8, 10

3.5.1 Browsers

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

3.5.2 Hardware Requirements

1. System : Intel®(4 vCpu) or i5
2. Hard Disk : 20 GB
3. Monitor : Virtual Machine: Standard D4s v3
4. Ram : 16 GB

4.RESULTS:

A web app has been developed for our model. The Front-End which is the home page takes input for the prediction factors that are 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.

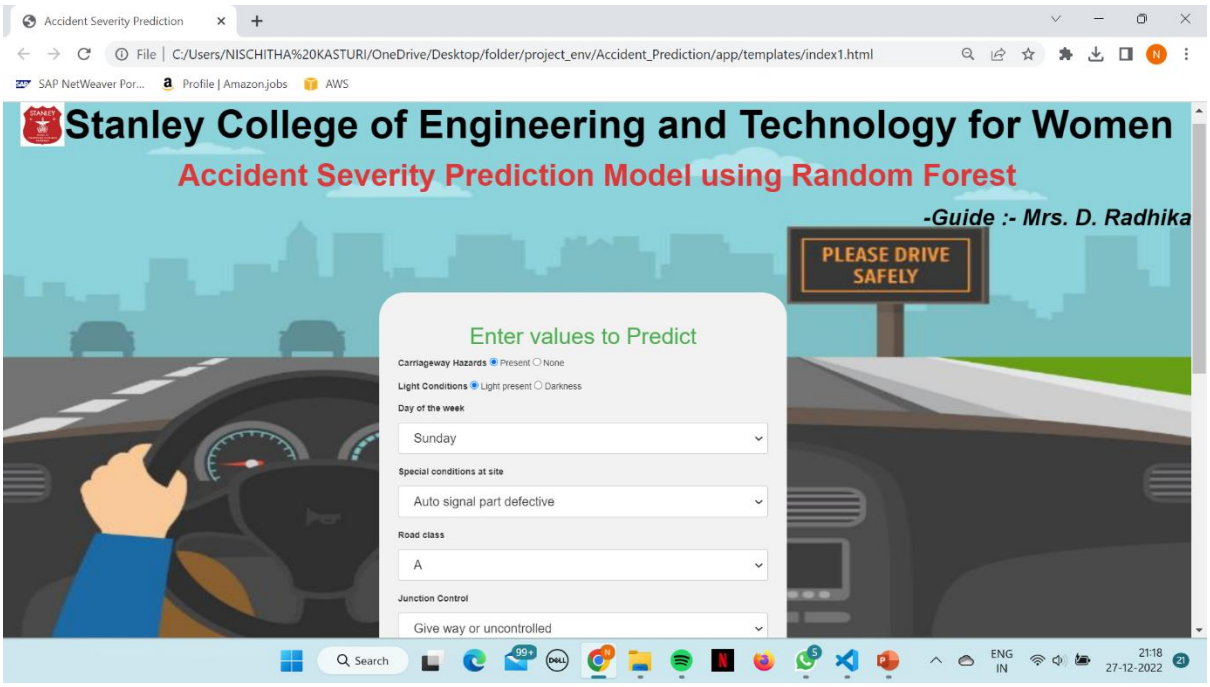


Figure 13: User Page 1

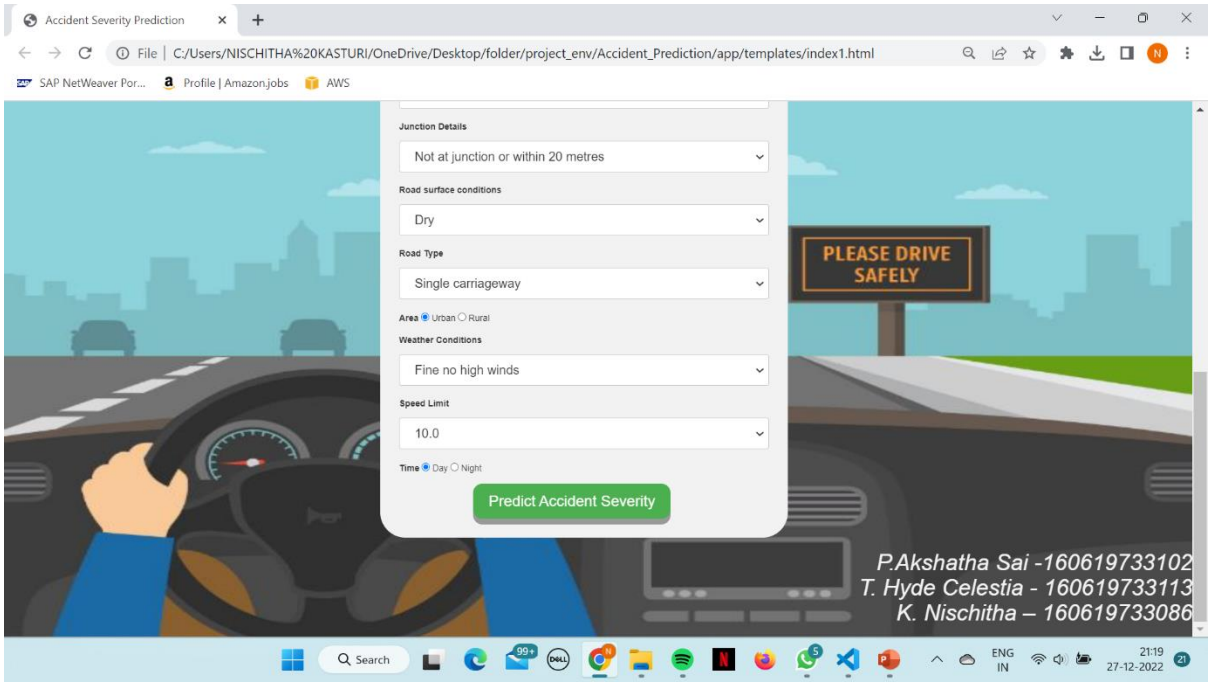


Figure 14: User Page 2

slight Level of Accidents Prediction:

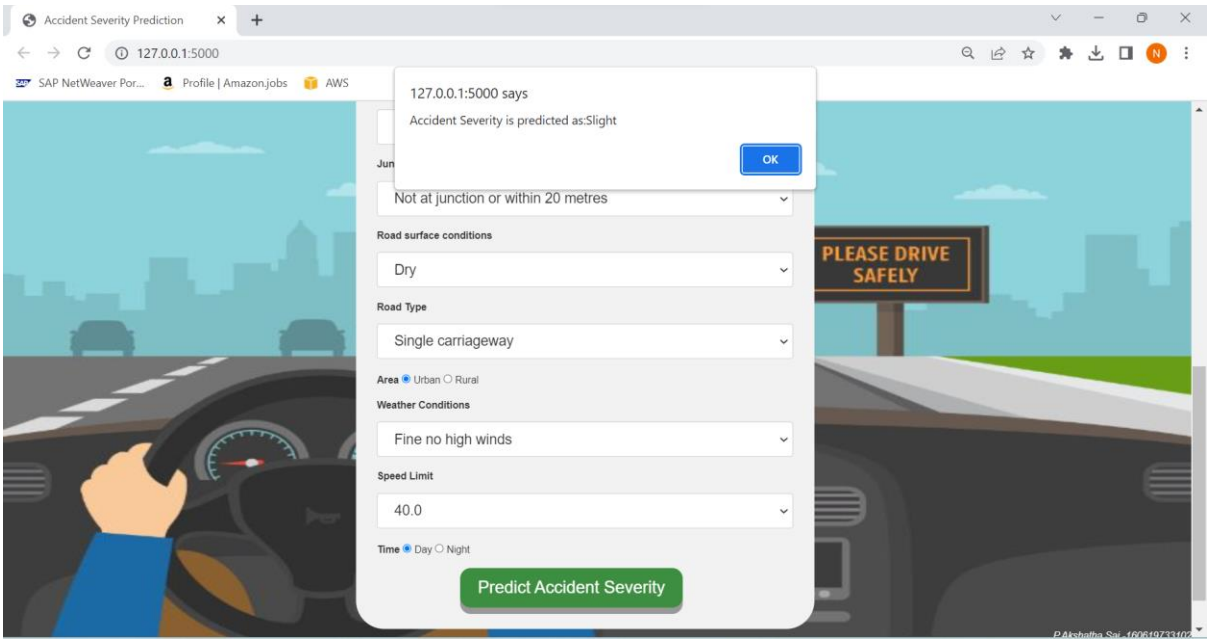


Figure 15: Output Predicted as Slight

Serious Level of Accidents Prediction:

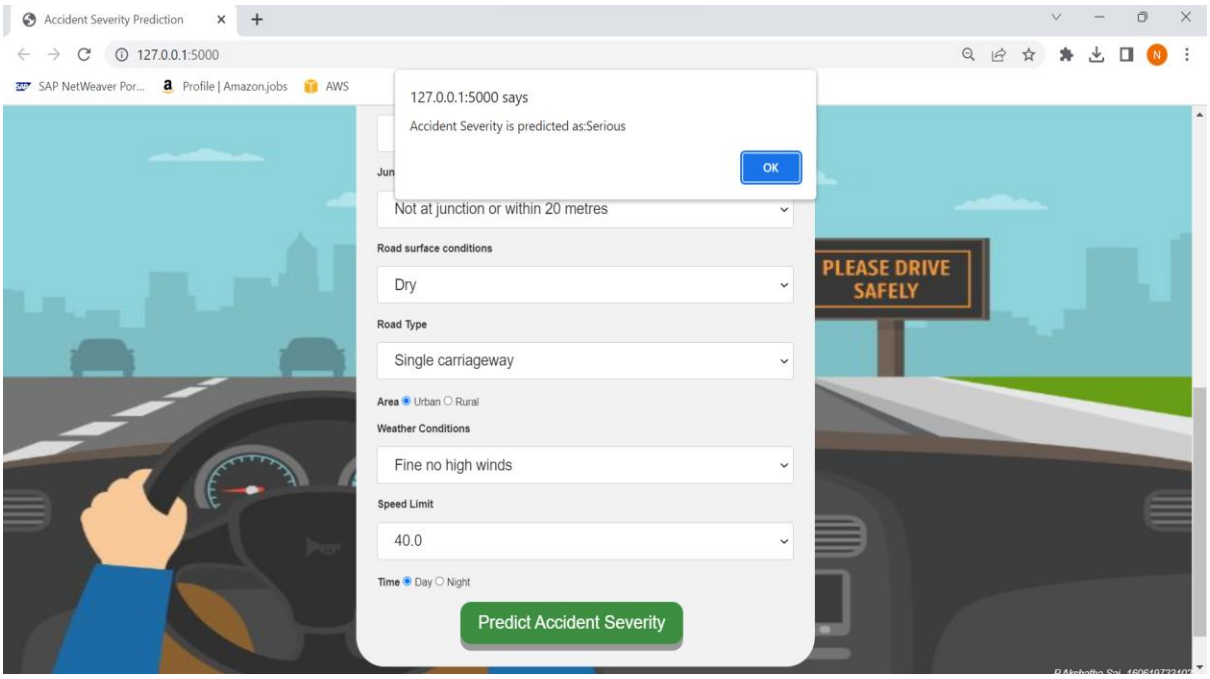


Figure 16: Output Predicted as Serious

Fatal Level of Accidents Prediction:

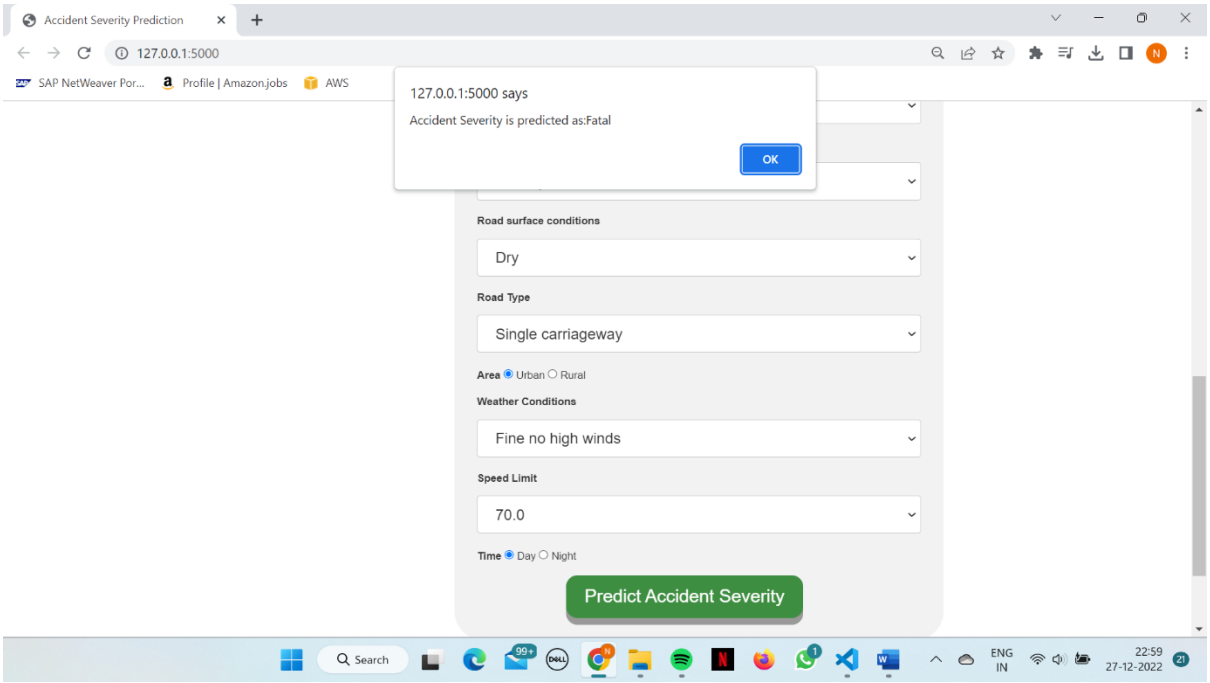
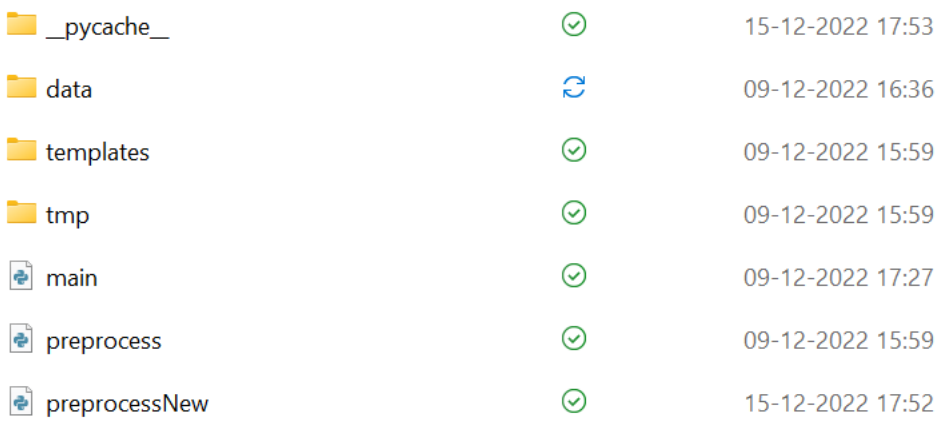


Figure 17: Output Predicted as Fatal

Arrangement of Required Files:



Executing Main.py in Visual Studious:

```
PS C:\Users\NISCHITHA KASTURI\OneDrive\Desktop\folder> ./project_env/Scripts/activate
(project_env) PS C:\Users\NISCHITHA KASTURI\OneDrive\Desktop\folder> cd project_env
(project_env) PS C:\Users\NISCHITHA KASTURI\OneDrive\Desktop\folder\project_env> cd Accident_Prediction
(project_env) PS C:\Users\NISCHITHA KASTURI\OneDrive\Desktop\folder\project_env\Accident_Prediction> cd ap
p
(project_env) PS C:\Users\NISCHITHA KASTURI\OneDrive\Desktop\folder\project_env\Accident_Prediction\app> p
ython main.py
```

Figure 18: Execution Steps in VS Code

Training the Model using Random Forest Algorithm:

```
def randomForestClassifier(self):
#class_weight = dict({2:1, 1:15, 0:50})
    X = self.df[self.cols]
    Y = self.df['Accident_Severity']
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
    rdf = RandomForestClassifier(bootstrap=True,
            class_weight="balanced_subsample",
            criterion='gini',
            max_depth=8, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0,
            min_samples_leaf=4, min_samples_split=10,
            min_weight_fraction_leaf=0.0, n_estimators=300,
            oob_score=False,
            random_state=35,
            verbose=0, warm_start=False)
    Y_pred = rdf.fit(X_train, Y_train).predict(X_test)
    self.printAnalysis(Y_test, Y_pred)
    return rdf
```

Figure 19: Random Forest Model

5.Conclusion

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

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.

Source Code:

Main.py

```
from flask import Flask, render_template
from flask import request, redirect, make_response
from flask_cors import CORS
import os
import os.path
import json
from preprocessNew import prediction
app = Flask(__name__, static_url_path = "", static_folder = "./temp")
CORS(app)

#initialize the data and model
obj = prediction()

@app.route("/")
def first_page():
   return render_template("index1.html")

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    print(data)
    val = obj.predictResult(data)
    r = {"result":val}
    resp = make_response(json.dumps(r))
    resp.status_code = 200
    resp.headers['Access-Control-Allow-Origin'] = '*'
    return resp

if __name__ == "__main__":
   app.run(host='0.0.0.0', debug=False, threaded=True)
```