Road Accident Severity Prediction Using Machine Learning

A project report submitted to ICT Academy of Kerala

in partial fulfillment of the requirements

for the certification of

CERTIFIED SPECIALIST

IN

DATA SCIENCE & ANALYTICS

Submitted by :

Arathi G
Geethu Vijayan
Medha P Sharma
Neeraj Padman U
Sebinuss S Pereira

ICT ACADEMY OF KERALA

THIRUVANANTHAPURAM, KERALA, INDIA
Nov 2022

List of Figures and Tables

Table of Contents

Abstract

Accidents in traffic lead to associated fatalities and economic losses every year worldwide and thus is an area of primary concern to society from loss prevention point of view.

Modeling accident severity prediction and improving the model are critical to the effective performance of road traffic systems for improved safety. In accident severity modeling, the input vectors are the characteristics of the accident, such as driver behavior and attributes of vehicle, highway and environment characteristics while the output vector is the corresponding class of accident severity.

This study performed a scientometric analysis of studies on traffic accidents in United Kingdom between 2017 and 2020.The main purpose of this study is to use different data mining techniques, namely, XGBoost, Random Forest (RF) and Decision Tree to compare their prediction capability, also to identify the significant variables that are strongly correlated with road accident severity, and distinguish the variables that have significant positive influence on prediction performance.

In this study, we are using F1 score as prediction performance evaluation measures to find the best integrated method which consists of the most effective prediction model.

Key words: XG Boost, Decision Tree, Random Forest Algorithm

1. Problem Definition

1.1 Overview

The work aims to study the dataset in various steps, which include:

1. Data Pre-Processing
2. Data visualization
3. Fixing the target column
4. Developing various machine learning models
5. Choosing the best model, fine tuning, and hosting the website.

The opening move is to obtain cleaned data from the original dataset. As the data contains a large number of null values and outliers, we have to do pre-processing of the data to fill in the missing values, remove the outliers, and perform principal component analysis for feature reduction as the data contains many columns that can adversely affect our model.

The next step is to plot different graphs so that we can get a better and easier understanding of the dataset. After data pre-processing, the cleaned data will be used for model creation. Our objective is to predict accident severity, so our target column will be "accident-severity," and we will be classifying the data into "slight injury," "serious injury," and "fatal injury" based on other dependent features. For this predictive modeling, we will try a logistic regression model, decision trees, and a random forest algorithm using our dataset. We will also note the changes in these models after min-max scaling and standard scaling. In order to evaluate these models, we will use the confusion matrix and accuracy score.

The work will be concluded by hosting a web page, where we can analyze the accident-severity by giving inputs which we have used to create the model.

2. Introduction

For the past many years, modeling the severity of motor vehicle crashes has been a significant area of research in traffic safety. It entails applying statistical tools to understand the variables that influence or are linked to crash severity and to forecast the severity of crashes with undetermined severity levels. There are various distinct categories used to quantify crash severity, including fatal, incapacitating, non-incapacitating, probable harm, and property damage alone. This work is focused on predicting the crash severity based on different parameters which are taken as input.

Traditionally, statistical models have been the most widely employed techniques for analyzing the severity of crash damages. The advantages of statistical models include the output of explicit formulas between dependent and explanatory variables and the ease of interpretation of the results. However, there are significant drawbacks to statistical models. For instance, a linear function is used to connect the dependent variable to the explanatory factors in statistical modeling technique, which makes assumptions about the data distribution. These presumptions might not always be accurate. Inaccurate parameter estimates will be produced if these assumptions are compromised.

We use machine learning approaches in our work to get beyond the shortcomings of statistical models. There are no presumptions about the relationships between variables in machine learning models.

Primarily, this work compares the performance of four machine learning models which are Random Forest, Decision tree algorithm and XGboost algorithm. Following that, the best algorithm in predicting the crash severity is selected based on the accuracy scores and is employed in developing the model. Finally, a web page will be hosted where a person can analyze the accident severity based on different parameter which can be given as inputs.

3. Literature Survey

Jian Zhang et al compared the predictive performance for crash injury severity between various machine learning and statistical models with distinct modeling logic. The predicting accuracy of each model on the training set and testing set was calculated and compared. Then the sensitivity analysis was conducted to infer the importance of explanatory variables on crash severity. The machine learning models in general produced better prediction performance for crash injury severity than did the statistical models. In particular, the RF and KNN were found to be the best models that had the highest overall predicting accuracy. Their best performance may be because they do not make any assumptions about the functional form of data and require fewer parameters to tune.

Their results suggest that though various machine learning methods can achieve a similar predicting accuracy on crash severity, sometimes they estimate variable importance on explanatory variables very differently. The probable reason is that different machine learning methods have distinct mechanisms on exploring the inherent features of data, leading to the different results on the estimation of parameter importance. More specifically, the logic of the KNN in prediction is to find the nearest/most similar data samples to make the inference on crash severity. While in the DT and RF, they aim at trying to find the best dividing structure and sequence to distinguish the crash severities; the RF prediction is made based on multiple rounds of voting of the DT models. The SVM maps the input vector into a high dimensional feature space and constructs an optimal separating hyperplane to separate the outcome into several groups, while maximizing the margin between the linear decision boundaries. As a consequence, due to the above differences, these machine learning models produced different inferences on variable importance.

Kanish Shah et al proposes the logistic regression, random forest and K-nearest neighbour algorithms which describes every aspect of model in detail by providing the evaluation metrics. They identify that the logistic regression has a higher level of accuracy (97%) followed by the random forest classifier algorithm (93%). The algorithm with the least accuracy among the three was K-nearest neighbour with the overall accuracy of 92%.

Md Adilur Rahim et al proposed a deep learning approach which was able to predict crash severity with improved performance. The performance of the deep learning model was evaluated using precision and recall, and compared with standard statistical learning models (SVM). The results showed that the proposed deep learning model outperformed the SVM model in classifying the fatal and injury crashes. An inference lawet was proposed to apply the deep learning framework for practical application.

Amirfarrokh Iranitalab et al study investigated the performance of classification methods for crash severity prediction, based on results of the application of these methods to real crash data. Among the methods experimented, KNN's best performance may be because this method does not make any assumptions about the functional form of data. Requirement of less parameters to tune and making less assumptions about the data may be one reason RF outperformed SVM. There are factors that enter a level of uncertainty and subjectivity in the results and conclusions of this study. For example, selection of type and kernel function in SVM. In other words, it is possible that different settings for the methods provide better results and change the draw conclusions.

4.Methodology

Three machine learning models were used in this investigation, each with a different classification algorithm. They are supervised learning models that approach modeling accident injury severity as a classification problem. Depending on the severity of the accidents, different categories are assigned to them. The labeled dataset is required to train the machine learning algorithms before they can predict injury severity given crash attributes. The given Dataset has been cleaned for null values and outliers in the pre-processing stage.

1. Random Forest algorithm

RF is considered an ensemble learning method for solving classification, regression and other tasks. This method generates many classifiers and aggregates their results. For a classification problem, the RF constructs a multitude of decision trees at the training time and outputs the class that is the mode of the classes. In RF, each node is split using the best among a subset of predictors randomly chosen at that node. Previous studies have reported that the RF performed well compared to many other classification models and suffer less overfitting issues. Two parameters need to be decided in the RF, which are the number of trees to grow and number of variables randomly sampled as candidates at each split.

2. Decision Tree Algorithm

Decision tree algorithm is a simple yet powerful technique for both classification and regression. It creates tree like model of decisions and their possible consequences. The tree is constructed by starting from the root node, which corresponds to the best feature to split the data based on, and then recursively partitioning the data at each internal node according to the values of the feature. The process stops when a certain stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf. The final nodes represent the predicted class or value for the input data. The advantage of the decision tree is that it is easy to interpret the result but it could easily overfit when the tree is too complex.

3. XGBoost

XGBoost (eXtreme Gradient Boosting) is an open-source library for training gradient boosting models in Python. It is an efficient and scalable implementation of gradient boosting. XGBoost is known for its speed, performance, and accuracy and is often used to train large-scale machine learning models.

Data Preparation

4.1 Libraries used

1. Time
2. Datetime
3. Math
4. Numpy
5. Pandas
6. Matplotlib
7. Seaborn
8. Plotly
9. Sklearn
10. Imblearn
11. sklearn

4.2 Data

The dataset was prepared from manual records of road traffic accidents that occurred in the UK. The dataset has information on 12316 accidents each with 32 features.

4.3 Exploratory data analysis

1. Data shape

Data shape gives us the number of rows and columns in the dataset. We can use the following command to get the shape of the data.

```
print("Shape of the data: {}".format(data.shape))
```

The output will be

```
Shape of the data: (12316, 32)
```

Which means there are 12316 rows and 32 columns.

2. Data columns

To get the columns in the dataset we use the command

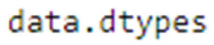
```
data.columns
```

We get the output as

```
Index(['Time', 'Day_of_week', 'Age_band_of_driver', 'Sex_of_driver',
       'Educational_level', 'Vehicle_driver_relation', 'Driving_experience',
       'Type_of_vehicle', 'Owner_of_vehicle', 'Service_year_of_vehicle',
       'Defect_of_vehicle', 'Area_accident_occured', 'Lanes_or_Medians',
       'Road_allignment', 'Types_of_Junction', 'Road_surface_type',
       'Road_surface_conditions', 'Light_conditions', 'Weather_conditions',
       'Type_of_collision', 'Number_of_vehicles_involved',
       'Number_of_casualties', 'Vehicle_movement', 'Casualty_class',
       'Sex_of_casualty', 'Age_band_of_casualty', 'Casualty_severity',
       'Work_of_casuality', 'Fitness_of_casuality', 'Pedestrian_movement',
       'Cause_of_accident', 'Accident_severity'],
      dtype='object')
```

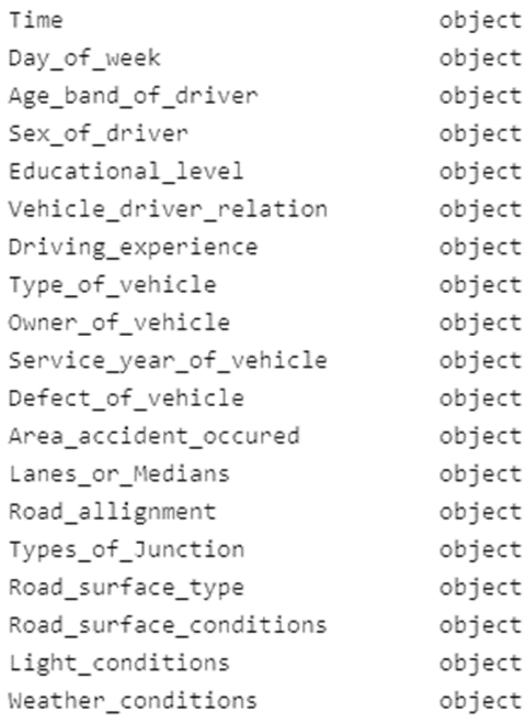
3. Data types

To identify the categorical and numerical data in the dataset we use the command as

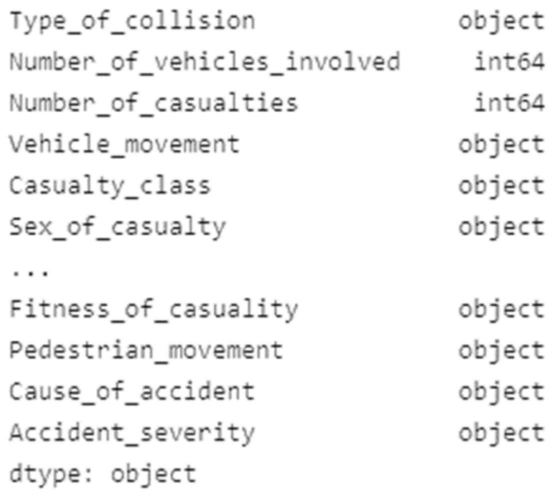


```
data.dtypes
```

The output will be,



```
Time                          object
Day_of_week                   object
Age_band_of_driver            object
Sex_of_driver                 object
Educational_level             object
Vehicle_driver_relation       object
Driving_experience            object
Type_of_vehicle               object
Owner_of_vehicle              object
Service_year_of_vehicle       object
Defect_of_vehicle             object
Area_accident_occured         object
Lanes_or_Medians              object
Road_allignment               object
Types_of_Junction             object
Road_surface_type             object
Road_surface_conditions       object
Light_conditions              object
Weather_conditions            object
```



```
Type_of_collision             object
Number_of_vehicles_involved    int64
Number_of_casualties           int64
Vehicle_movement              object
Casualty_class                object
Sex_of_casualty               object
...
Fitness_of_casuality          object
Pedestrian_movement           object
Cause_of_accident             object
Accident_severity             object
dtype: object
```

We can distinguish the categorical and numerical data from the above output.

```
Number of integer columns: 30
Number of object columns: 2
```

4. Duplicate data

There will be many duplicate values, duplicate rows and duplicate columns in the dataset. These duplicate values will affect the final output of the data and hence dealing with these data is important.

We can use the following code,

```
duplicate = data[data.duplicated() == True]
print("Number of duplicate rows: {}".format(len(duplicate)))
```

The output was,

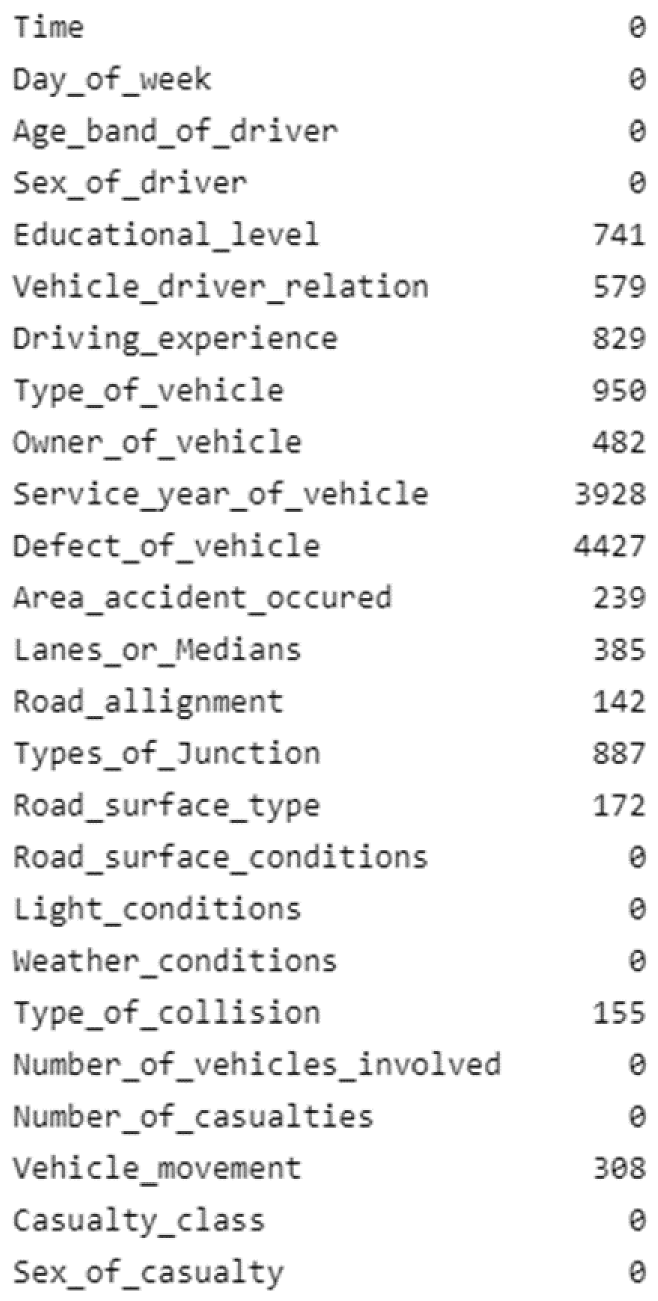
```
Number of duplicate rows: 0
```

5. Null value

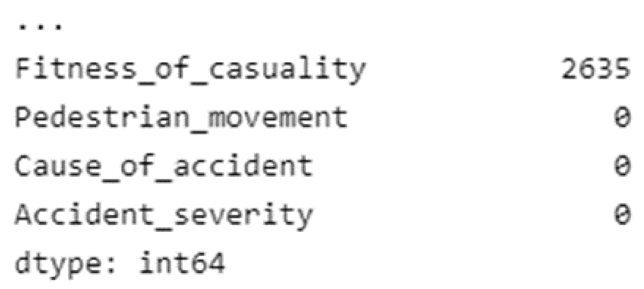
In a dataset, a null value is a placeholder for missing or null data. It is used to indicate that there is no data available for that particular element in the dataset. Null values are often represented by the special value None in Python.

```
data.isna().sum()
```

The output is as follows,



```
Time                            0
Day_of_week                     0
Age_band_of_driver              0
Sex_of_driver                   0
Educational_level             741
Vehicle_driver_relation       579
Driving_experience            829
Type_of_vehicle               950
Owner_of_vehicle              482
Service_year_of_vehicle      3928
Defect_of_vehicle            4427
Area_accident_occured         239
Lanes_or_Medians              385
Road_allignment               142
Types_of_Junction             887
Road_surface_type             172
Road_surface_conditions         0
Light_conditions                0
Weather_conditions              0
Type_of_collision             155
Number_of_vehicles_involved     0
Number_of_casualties            0
Vehicle_movement              308
Casualty_class                  0
Sex_of_casualty                 0
```



```
...
Fitness_of_casuality         2635
Pedestrian_movement             0
Cause_of_accident               0
Accident_severity               0
dtype: int64
```

After these processes

- Number of observations: 12316
- Number of columns: 32
- Number of integer columns: 2
- Number of object columns: 30
- Number of columns with missing values: 16

Also, Columns with missing values:

- Educational level
- Vehicle_driver_relation
- Driving_experience
- Type_of_vehicle
- Owner_of_vehicle
- Service_year_of_vehicle
- Defect_of_vehicle
- Area_accident_occured
- Lanes_or_Medians
- Road_allignment
- Types_of_Junction
- Road_surface_type
- Type_of_collision
- Vehicle_movement
- Work_of_casuality
- Fitness_of_casuality

4.4 Data visualization

For our work, we set the target variable as accident_severity.
Accident_severity is a categorical variable with three possible values.

1. Slight injury
2. Serious injury
3. Fatal injury

Comparing the frequencies of slight injury, serious injury and fatal injury with bar plot and donut plot.

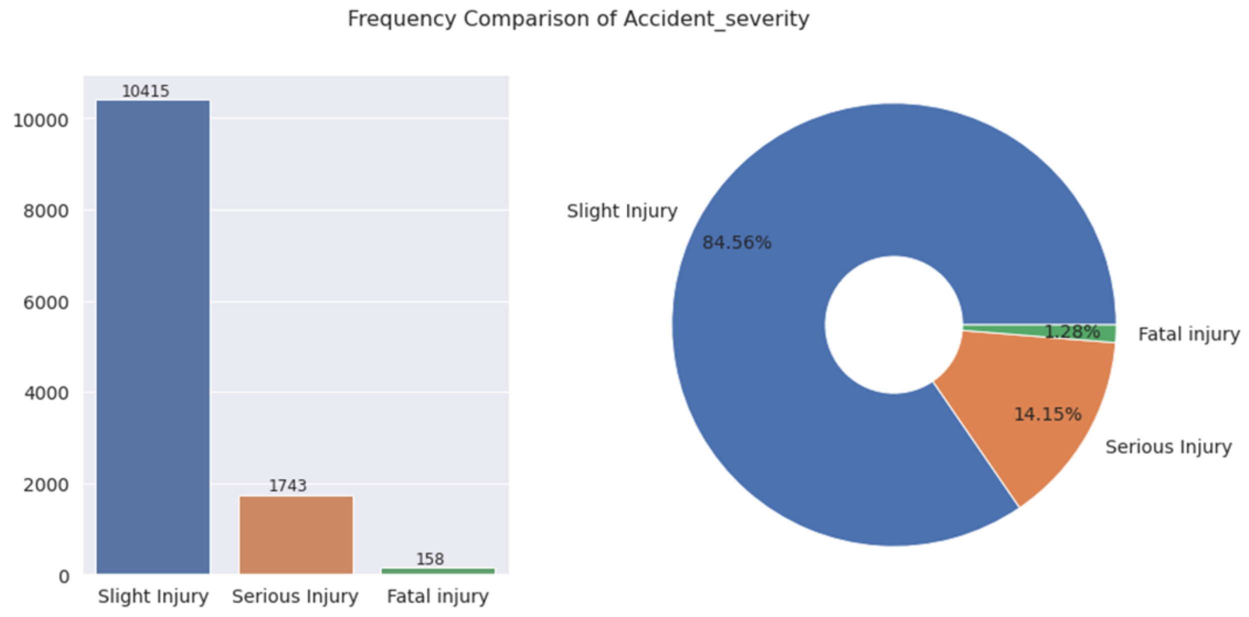


Fig.1.frequency comparison of Accident_severity

Plot to compare frequency distribution of features across target classes are as follows

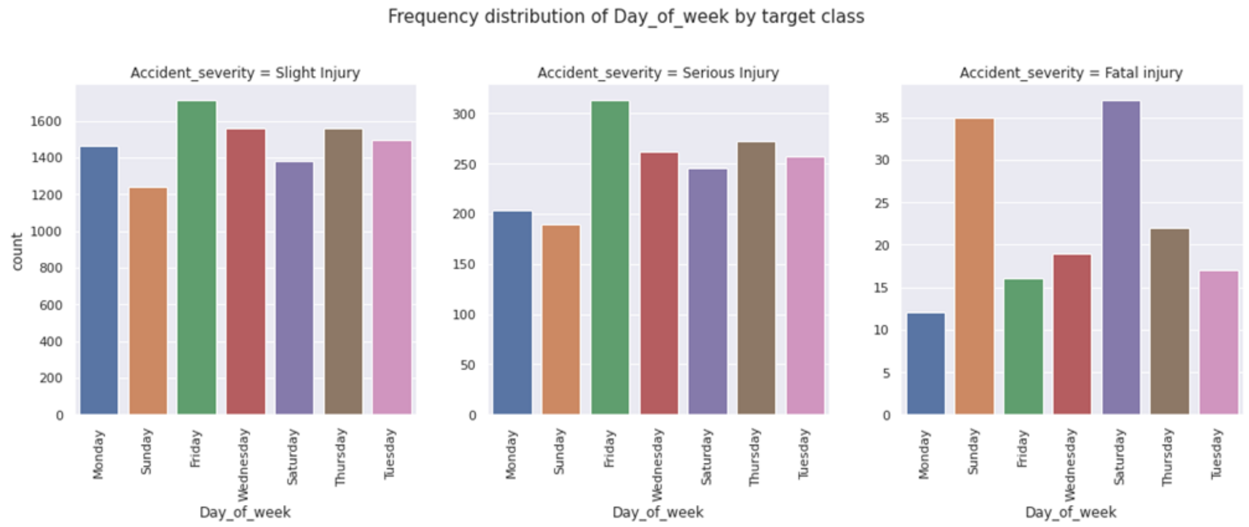


fig.2 frequency distribution of features across target classes(1)

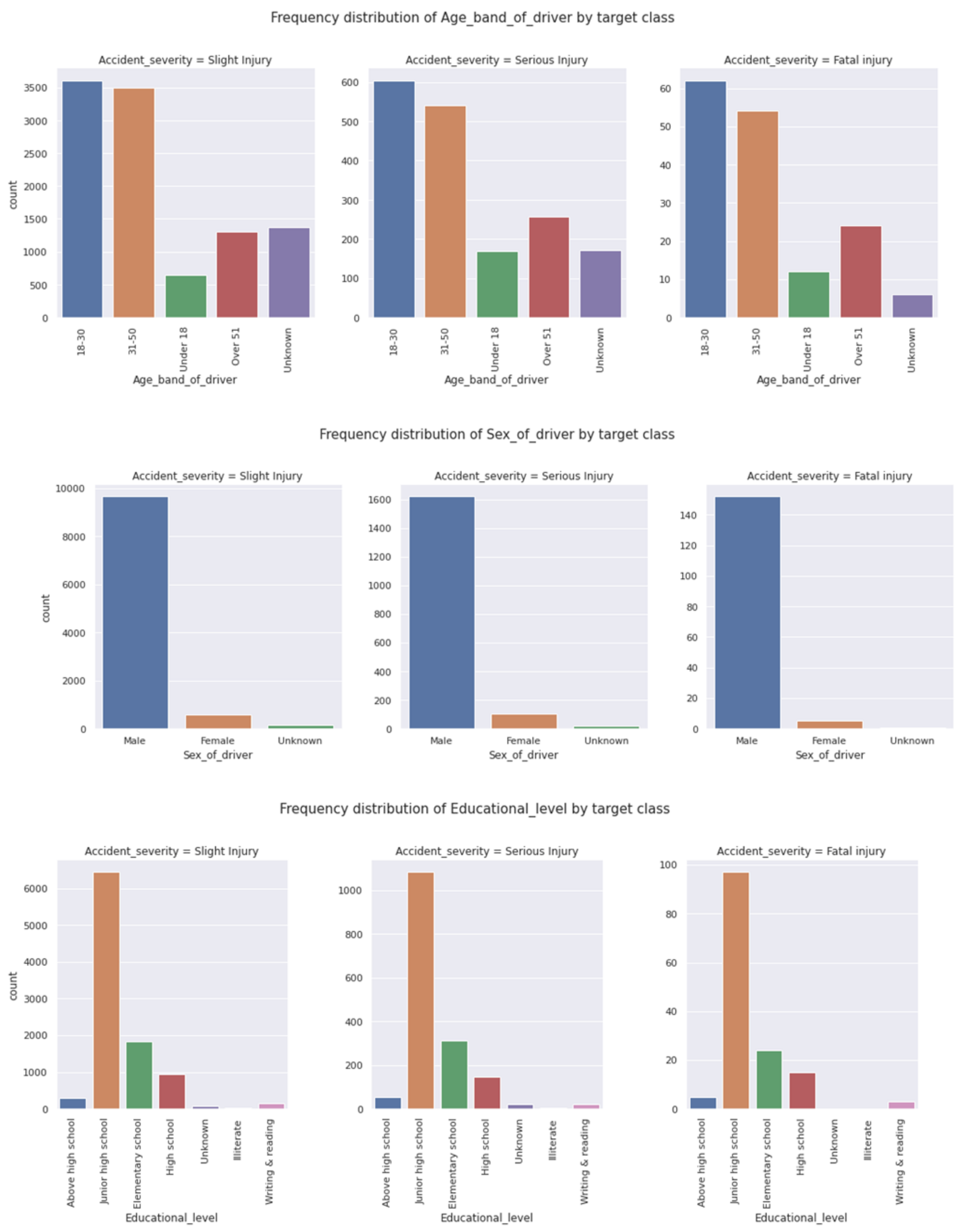


fig.2 frequency distribution of features across target classes(2)

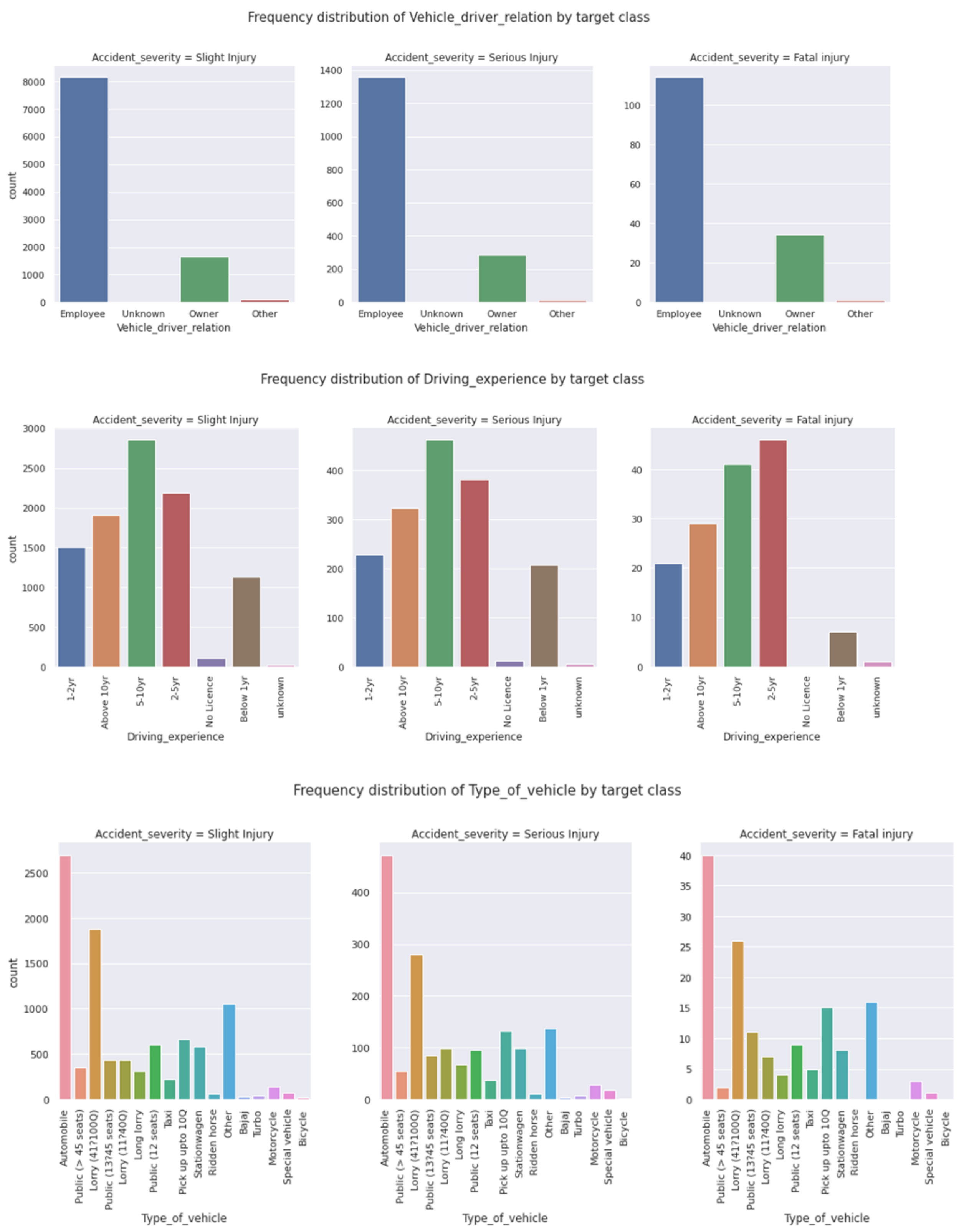


fig.2 frequency distribution of features across target classes(3)

4.5 Outlier detection

In a dataset, an outlier is a data point that is significantly different from the other data points in the dataset. Outliers can be caused by errors in data collection or recording, or they can be legitimate data points that are simply unusual.

To identify outliers in a dataset, we can use statistical techniques such as the mean and standard deviation to identify data points that are significantly different from the rest of the data. In Python, we can use the pandas library to easily calculate the mean and standard deviation of a dataset, and then use these values to identify outliers.

There are several methods for identifying outliers in a dataset in Python. Some common methods include:

1. Using descriptive statistics: We can use the mean and standard deviation of a dataset to identify outliers. For example, we can consider values that are more than three standard deviations away from the mean to be outliers.

2. Using box plots: Box plots are a graphical way of representing the distribution of a dataset. They can be used to identify outliers because they show the range of the dataset and any values that fall outside of the range can be considered outliers.

3. Using the interquartile range (IQR): The IQR is the range of the middle 50% of values in a dataset. We can use the IQR to identify outliers by considering values that are less than Q1 - 1.5*IQR or values that are greater than Q3 + 1.5*IQR to be outliers, where Q1 and Q3 are the first and third quartiles of the dataset, respectively.

4. Using the Z-score: The Z-score of a value is the number of standard deviations it is away from the mean. We can use the Z-score to identify outliers by considering values that have a Z-score greater than 3 or less than -3 to be outliers.

5. Using the DBSCAN algorithm: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that can be used to identify outliers in a dataset. It works by identifying clusters of points that are dense and labeling points that do not belong to any cluster as outliers.

6. Using the Isolation Forest algorithm: Isolation Forest is an algorithm that can be used to identify outliers in a dataset by isolating them in the tree structure created by the algorithm. It works by randomly selecting a feature and a split value and creating a decision tree. Points that are isolated in the tree structure are considered to be outliers.

In our work we use boxplots to detect outliers.

4.5.1 Boxplots

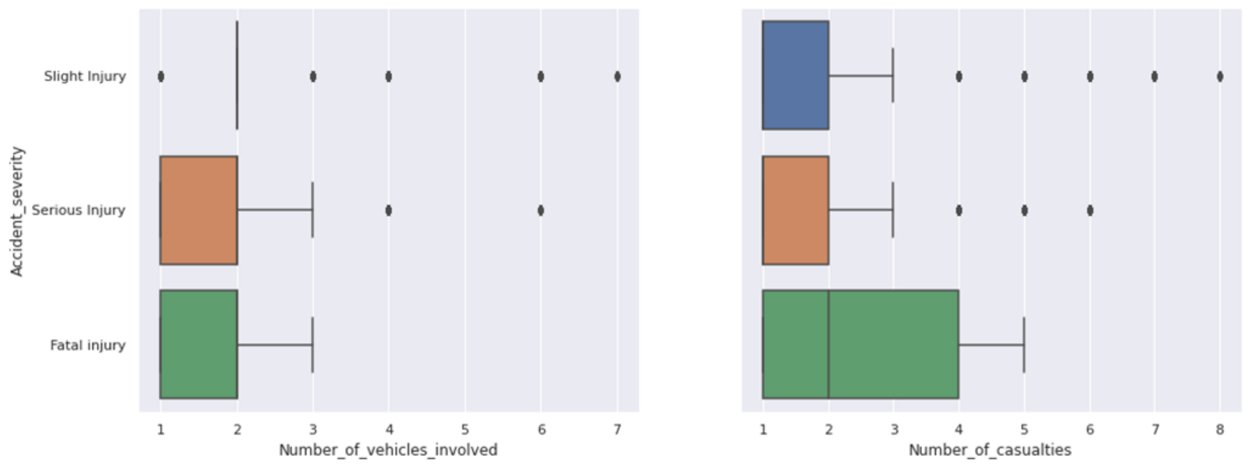


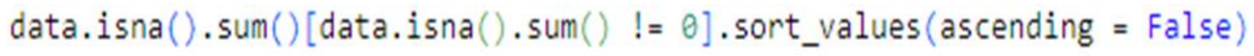
Fig.3 boxplots to identify outliers

Despite the appearances of the apparent outliers, we refrain from deleting or modifying them as it is evident from the range of the variables that these are most likely to be genuine values, containing relevant information about the corresponding variables.

5. Missing data imputation

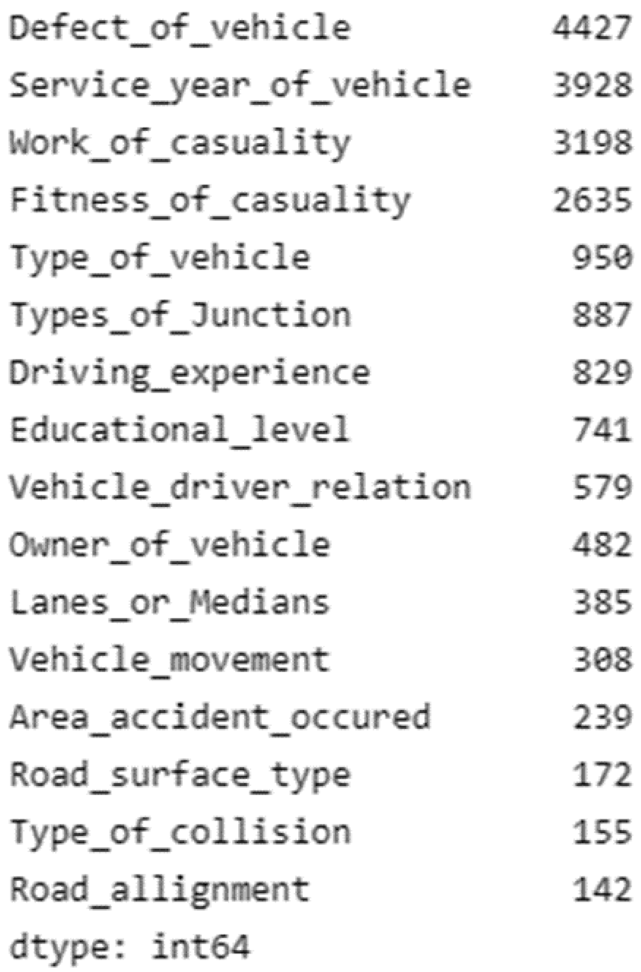
Columns with missing values are sorted according to descending order

we can use the following code



```
data.isna().sum()[data.isna().sum() != 0].sort_values(ascending = False)
```

The output is as follows

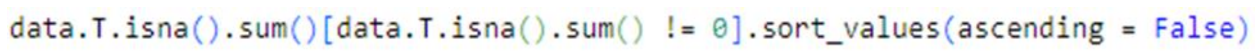


```
Defect_of_vehicle          4427
Service_year_of_vehicle    3928
Work_of_casuality          3198
Fitness_of_casuality       2635
Type_of_vehicle             950
Types_of_Junction           887
Driving_experience          829
Educational_level           741
Vehicle_driver_relation     579
Owner_of_vehicle            482
Lanes_or_Medians            385
Vehicle_movement            308
Area_accident_occured       239
Road_surface_type           172
Type_of_collision           155
Road_allignment             142
dtype: int64
```

16 columns out of 32 columns contain missing values. The features with missing values are categorical in nature.

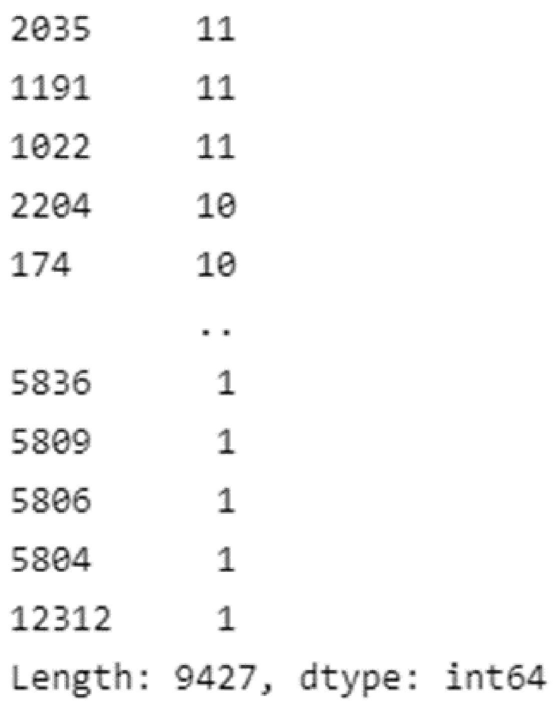
5.1 Rows with missing values

We can use the following code to sort out the rows with missing values



```
data.T.isna().sum()[data.T.isna().sum() != 0].sort_values(ascending = False)
```

The output will be as follows



```
2035     11
1191     11
1022     11
2204     10
174      10
         ..
5836      1
5809      1
5806      1
5804      1
12312     1
Length: 9427, dtype: int64
```

We can see that 9427 rows out of 12316 rows contain missing values.

5.2 Imputation

There are several imputation methods that we can use to fill in missing values in a dataset in Python. Some common methods include:

1. Mean imputation: This method involves replacing the missing value with the mean of the non-missing values for that feature.

2. Median imputation: This method involves replacing the missing value with the median of the non-missing values for that feature.

3. Mode imputation: This method involves replacing the missing value with the mode of the non-missing values for that feature.

4. K-nearest neighbor imputation: This method involves using the values of the k nearest neighbors of the missing value to impute the missing value.

5. Multiple imputation: This method involves creating multiple versions of the dataset with the missing values imputed using different methods or models, and then combining the results.

6. Interpolation: This method involves using a mathematical function to estimate the missing value based on the known values before and after it.

7. Extrapolation: This method involves using a mathematical function to estimate the missing value based on the known values after it.

8. Using machine learning models: We can also use machine learning models, such as linear regression or k-nearest neighbors, to impute missing values in a dataset.

9. Proportion based imputation is a method of imputing missing values in a dataset by replacing the missing value with the proportion of the non-missing values in that feature. To implement proportion based imputation in Python, we can use the SimpleImputer class from the sklearn.impute module.

6. Data encoding

Data encoding is the process of converting data from one format to another. In Python, data encoding is often used to convert categorical or text data into a numerical format that can be used by machine learning models. There are several methods for encoding data in Python, including:

1. One-hot encoding: This method involves creating a new binary column for each unique category in a categorical feature.
2. Label encoding: This method involves assigning a unique integer to each category in a categorical feature.
3. Binary encoding: This method involves encoding a categorical feature as a binary string
4. Count encoding: This method involves encoding a categorical feature as the count of how many times each category occurs in the dataset.
5. Target encoding: This method involves encoding a categorical feature based on the target variable.

To encode data in Python, we can use the LabelEncoder class from the sklearn.preprocessing module to perform label encoding, or can use the OneHotEncoder class to perform one-hot encoding.

An appropriate encoding scheme is given as follows:

Ordinal features: Manual encoding or Label encoding

Nominal features: One-hot encoding

However, the dataset contains a lot of nominal features. As a result, one-hot encoding produces too many columns, which eventually leads to a curse of dimensionality and loss of relevant information at the feature selection stage. For this reason, we resort to the following scheme:

Ordinal features: Manual encoding

Nominal features: Label encoding

7. Model Training

The following classifiers are experimented,

1. Decision tree
2. Random Forest algorithm
3. XGBooster

The confusion matrix obtained for each classifier are as follows.

7.1 Decision tree

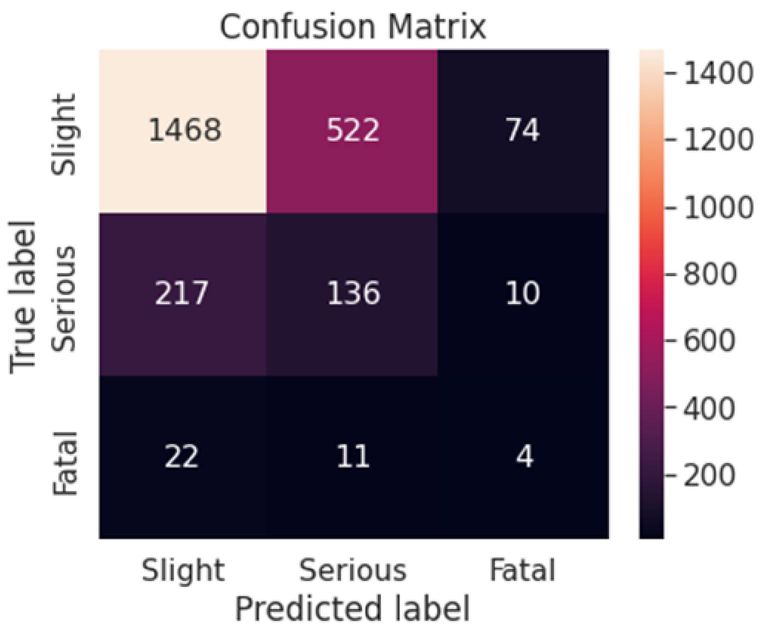


Fig.4 confusion matrix for Decision tree algorithm

7.2 Random Forest algorithm

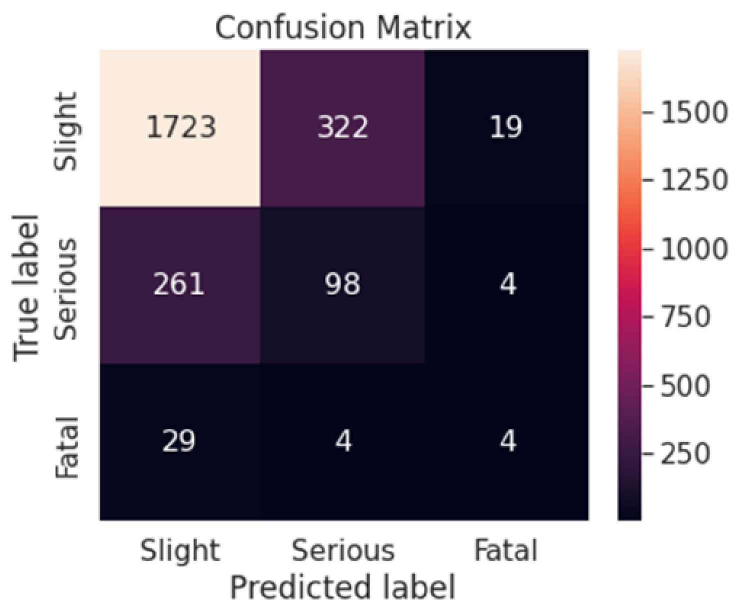


Fig.5 confusion matrix for Random Forest algorithm

7.3 XGBoost

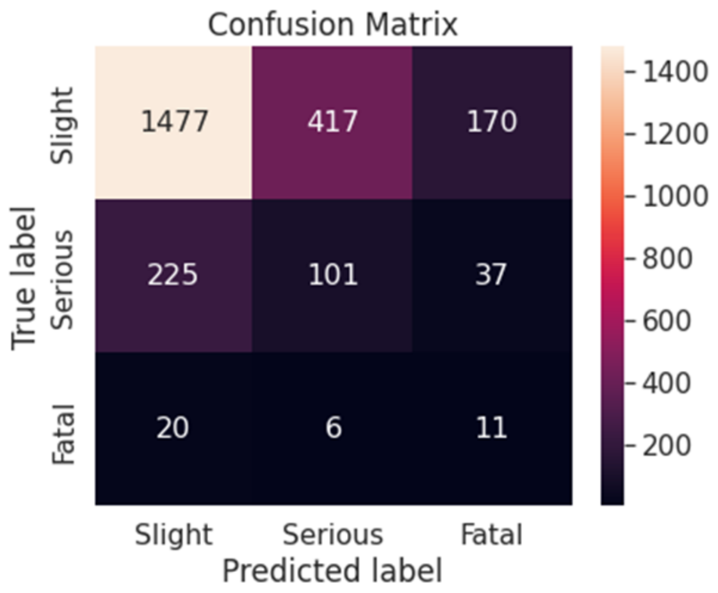


Fig.6 confusion matrix for XGBoost

Classifier	F1 Score
Decision tree	0.6919
Random forest algorithm	0.7465
XGBoost	0.7465

Table.1 Comparison of the performance of the classifiers

On training the model using several classification algorithms such as XGBoost, Random Forest, Decision Tree, the model trained with XGBoost gave good results.

- Used KFold with 5 splits cross validation with hyper-parameter tuning on XGBoost Classifier (baseline model) using GridSearchCV.

- We found that our baseline model (XgBoost Classifier) was overfitting the dataset. On investigation we found that the dataset was affected by Curse of Dimensionality. So, we reduced the dimensions and trained the model again.

- After retraining the model, we found that it was generalizing well with an accuracy of 74%.

- As per the problem statement we used F1 Score as the evaluation metric for the model.

8. Web deployment

8.1 Requirements

- joblib==1.0.1
- numpy==1.21.3
- pandas==1.2.4
- scikit_learn==1.1.0
- streamlit==1.9.2
- xgboost==1.5.1

Streamlit cloud helps to deploy apps in just one click.It’s an open source python library that allows you to build beautiful and interactive web based data applications quickly . Streamlit is particularly well-suited for creating data visualizations, machine learning tools and interactive dashboards and uses only minimal codes.

Step 1: Initializing project structure

- The. Streamlit folder for now, is used to save the Streamlit configuration file including the theme.

- The app.py contains the main code in python that is the soul of your Streamlit app, what it displays, and how it reacts to user input.

- The theme folder is used to gather any images or other media that is useful for the app.

- The data folder, as the name suggests, will accommodate the data, that is, the brain of your Streamlit app.

- The model.ipynb ,notebook where initial preprocessing done and contains machine learning model for our app.

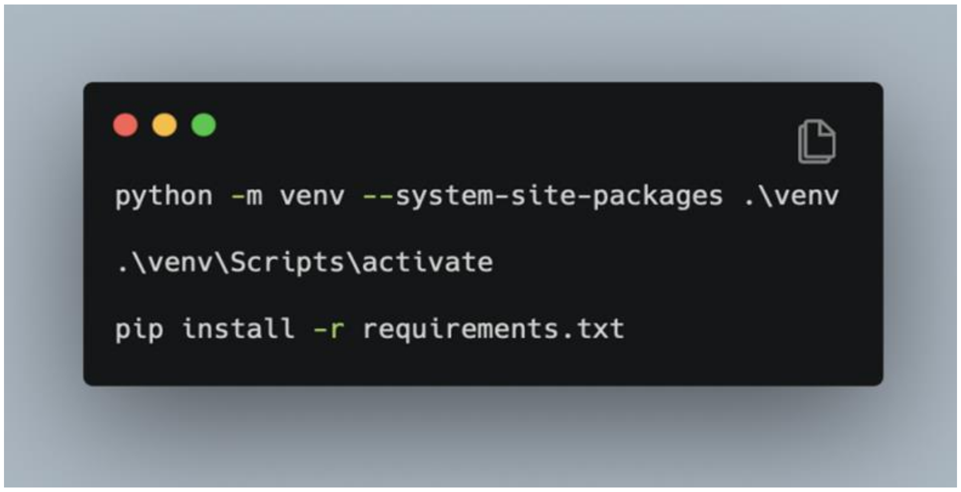
- The requirements.txt file contains any packages that the app depends on.

step 2: Install libraries

The first library we need is of course streamlit. Since the data we are using is a CSV file, it’s good to have pandas. We need to store all the required libraries by writing the script below in requirements.txt To ensure reproducibility and avoid future errors, the library version is also mentioned as follows.

```
pandas==1.2.3
streamlit==1.3.1
```

We need to install these libraries using pip command in the python environment of your choice. We need to create a new virtual environment dedicated to this project by running the code below in the terminal.



```
python -m venv --system-site-packages .\venv

.\venv\Scripts\activate

pip install -r requirements.txt
```

Step 3: Prepare Dataset

We can save the data to a csv file RTA_Dataset.csv in the folder dataset.

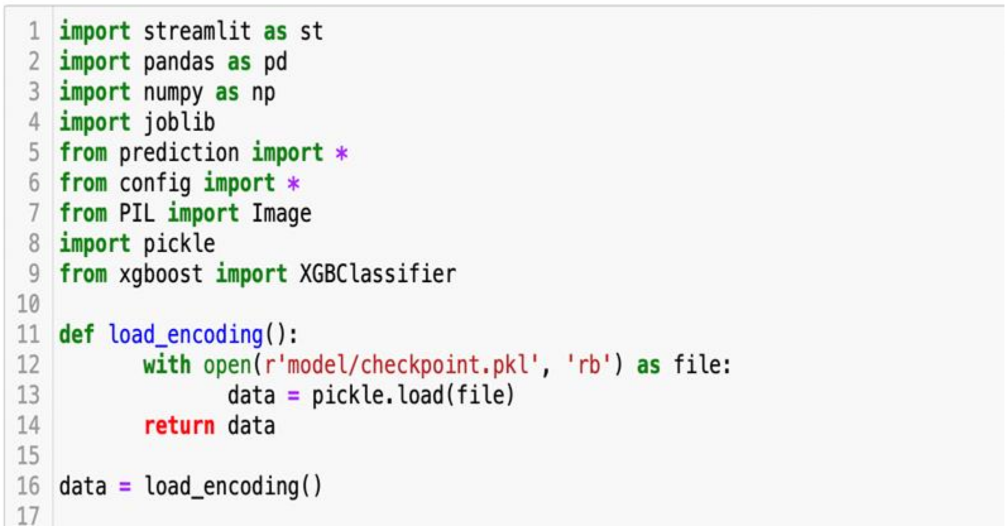
Step 4: Sketch for web app

We have to build the layout of the web app. We will separate the page into two parts:

1. Head: includes the title “Accident Severity Prediction Application”

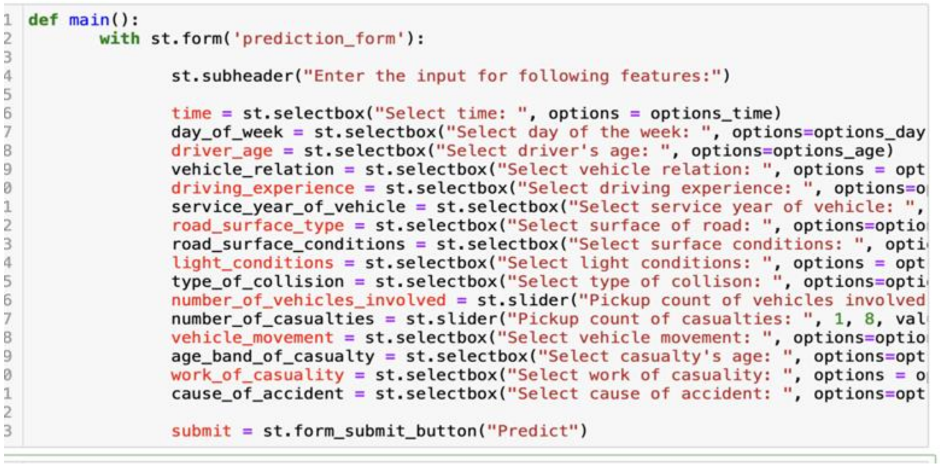
2. Body: includes region for the user input values and a submit button option.

Step 5: Build Main functions
We have to create a python function to read data from the dataset using pandas.

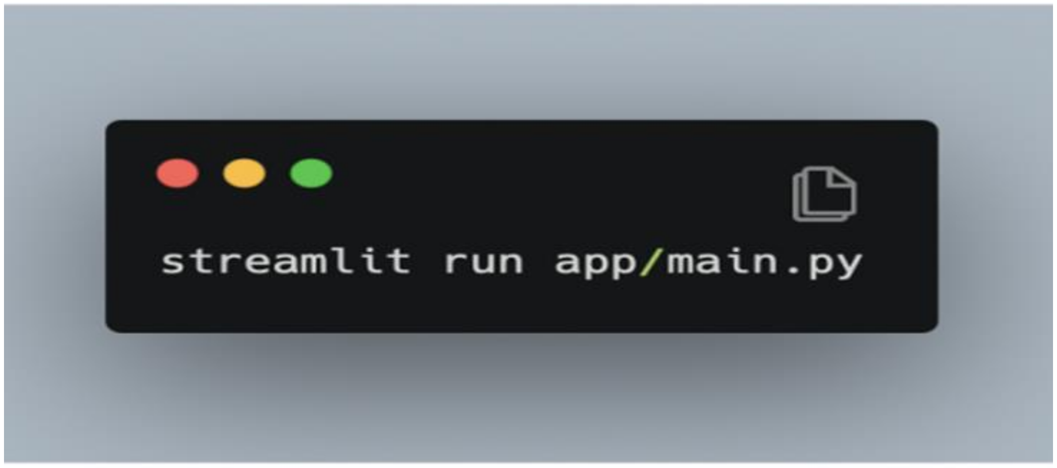

```python
import streamlit as st
import pandas as pd
import numpy as np
import joblib
from prediction import *
from config import *
from PIL import Image
import pickle
from xgboost import XGBClassifier

def load_encoding():
        with open(r'model/checkpoint.pkl', 'rb') as file:
                data = pickle.load(file)
        return data

data = load_encoding()

```

- We use st.markdown for the title and allows HTML format to force the title to be in the middle of the page.
- We use Image.open('doc/theme.png') to set a background theme

- Lastly, make a function for the body of the web app


```python
def main():
        with st.form('prediction_form'):

                st.subheader("Enter the input for following features:")

                time = st.selectbox("Select time: ", options = options_time)
                day_of_week = st.selectbox("Select day of the week: ", options=options_day
                driver_age = st.selectbox("Select driver's age: ", options=options_age)
                vehicle_relation = st.selectbox("Select vehicle relation: ", options = opt
                driving_experience = st.selectbox("Select driving experience: ", options=o
                service_year_of_vehicle = st.selectbox("Select service year of vehicle: ",
                road_surface_type = st.selectbox("Select surface of road: ", options=optio
                road_surface_conditions = st.selectbox("Select surface conditions: ", opti
                light_conditions = st.selectbox("Select light conditions: ", options = opt
                type_of_collision = st.selectbox("Select type of collison: ", options=opti
                number_of_vehicles_involved = st.slider("Pickup count of vehicles involved
                number_of_casualties = st.slider("Pickup count of casualties: ", 1, 8, val
                vehicle_movement = st.selectbox("Select vehicle movement: ", options=optio
                age_band_of_casualty = st.selectbox("Select casualty's age: ", options=opt
                work_of_casuality = st.selectbox("Select work of casuality: ", options = o
                cause_of_accident = st.selectbox("Select cause of accident: ", options=opt

                submit = st.form_submit_button("Predict")
```

- Now, we need to go to the terminal and type this code to run the Streamlit app locally to see how far we've gone.



```
streamlit run app/main.py
```

- The app will start in the default browser.

Step 6: Commit to GitHub

Streamlit cloud launches app directly from the GitHub repository. Hence app code and dependencies were uploaded to GitHub before deploying the app.

1. Go to github.com/new and enter your repository name.
2. Click the "Create repository" button at the bottom of the page.
3. Copy the repository URL.

Step 7: Deploying new app

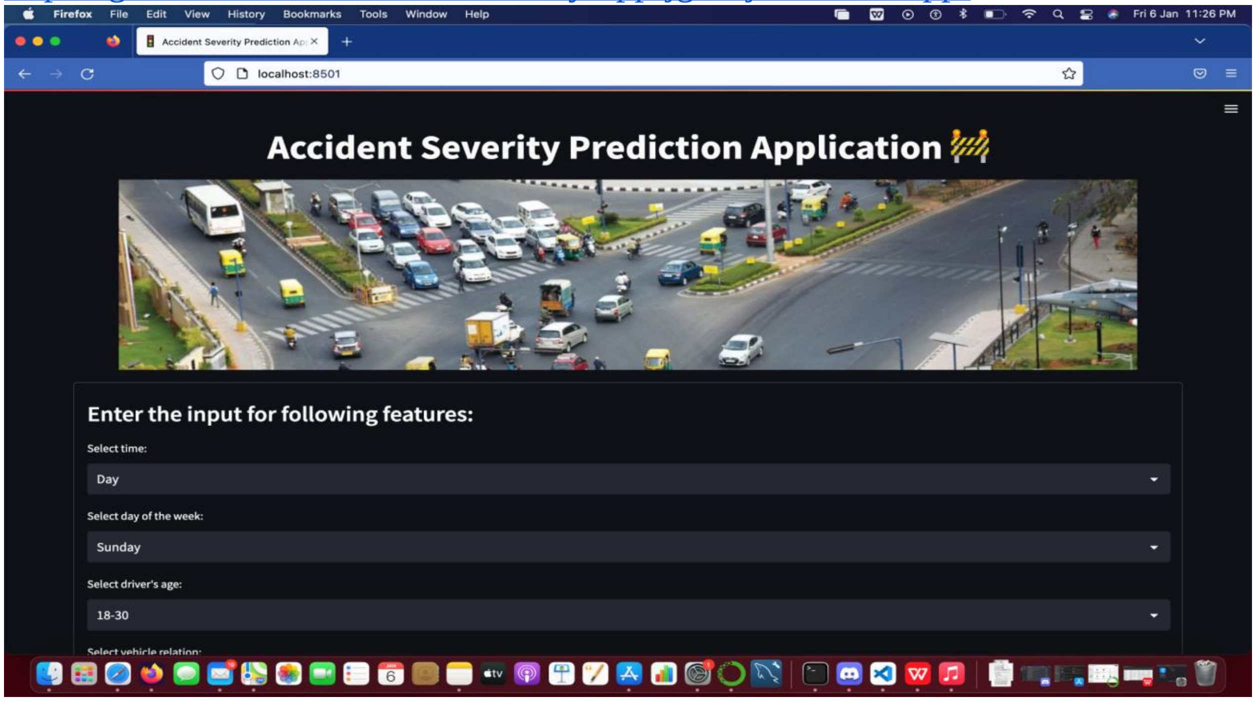
In share.streamlit.io click the "New app" button. Fill in the repository, branch, and main file path.
Most apps take only a couple of minutes to deploy.

Finally, we can obtain a unique subdomain URL (https://geethu-vishnu-accident-severity-app-jg2zaj.streamlit.app/)that we can share with others.

9. Result

App deployed:

https://geethu-vishnu-accident-severity-app-jg2zaj.streamlit.app/


Firefox File Edit View History Bookmarks Tools Window Help
Fri 6 Jan 11:26 PM
Accident Severity Prediction Ap
localhost:8501

Accident Severity Prediction Application

Enter the input for following features:

Select time:
Day

Select day of the week:
Sunday

Select driver's age:
18-30

Select vehicle relation:

Findings from this case study :

- Accidents : accidents occurred mostly on Fridays and least on Sundays.Most of the accidents occurred on 8 AM and between 3 PM to 6 PM (office and school hours) . 85% of the severity is slight injury.
- Driver : 35 % of the accident occurred to drivers in age band 18-30 and 33% of drivers in age band 31-50 . 94% of accidents occurred to male drivers . Accidents occurred mostly to persons having junior high school educational background
- Road : accidents occurred mostly on lanes having two-way (divided with broken lines road marking) . Road alignment were accidents occurred mostly was tangent road with flat terrain with Y shaped type of junction

10. Conclusion

This project presented a data analytic framework in which UK traffic accidents data was analyzed to establish a model for predicting injury severity. The project used publicly available data from 2017 to 2019 to build prediction models for injury severity level. The project has combined all attributes from the data sources to analyze 32 attributes and its relation with accident severity. We have observed that the application of EDA can assist in uncovering hidden patterns in the datasets and how important it is in data science. The work highlighted issues related to data quality and imbalanced data and applied techniques to tackle these issues. The work compared performance of different machine learning techniques such as Decision Tree, Random Forest and XGBoost. The XGBoost algorithm was shown to outperform other techniques with higher accuracy rate .

References

Jian Zhang1, Zhibin Li1, Ziyuan Pu2, Chengcheng Xu1, “Comparing Prediction Performance for Crash Injury Severity among Various Machine Learning and Statistical Methods **“,** School of Transportation, Southeast University, Nanjing, 211189 China Department of Civil and Environmental Engineering, University of Washington, Seattle, 98195, United States

Kanish Shah, Henil Patel, Devanshi Sanghvi, Manan Shah,” A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification”, Augmented Human Research (2020) 5:12

Md Adilur Rahim, Hany M. Hassan,” A deep learning-based traffic crash severity prediction framework”, Department of Civil and Environmental Engineering, Louisiana State University, Patrick Taylor Hall, Baton Rouge, LA, 70803, USA

Amirfarrokh Iranitalab, Aemal Khattak,” Comparison of four statistical and machine learning methods for crash severity prediction”, Department of Civil Engineering and Nebraska Transportation Center, University of Nebraska-Lincoln, 330P Prem S. Paul Research Center at Whittier School, Lincoln,

NE, 68583-0851, United States