Predictive Ananlysis for Accident Prevention and Response

**Abstract**

Traffic accidents have become severe risks as they are one of the causes of enormous deaths worldwide. Reducing the number of incidents is critical to saving lives and achieving sustainable cities and communities. Machine learning and data analysis techniques interpret the reasons for car accidents and propose solutions to minimize them. However, this needs to take the benefits of big data solutions as the size and velocity of traffic accident data are increasingly large and rapid. This paper explores road car accident data patterns and proposes a predictive model by investi-gating meaningful data features, such as accident severity, the number of casualties, and the number of vehicles. Therefore, a pre-processing model is designed to convert raw data using missing and meaningless feature removal, data attribute generalization, and outlier removal using interquartile. Four classification methods, including decision trees, random forest, multinomial logistic regression, and naïve Bayes, are used and evaluated to study the performance of road accident prediction. The results address acceptable levels of accuracy for car accident prediction except for naïve Bayes. The findings are discussed through a data-driven approach to understand the factors influencing road car accidents and highlight the key ones to propose accident prevention solutions. Finally, some strategies are provided to achieve healthy and community-friendly cities.

Keywords: machine learning; road car accident; prediction model; big data; sustainable community;data-driven approach; community-friendly

**Introduction :**

This is a multiclass classification project to classify the severity of road accidents into three categories. This project is based on real-world data, and the dataset is also highly imbalanced. There are three types of injuries in a target variable: minor, severe, and fatal.Road accidents are the major cause of unnatural deaths around the world. All governments work hard to raise awareness about the rules and regulations that must be followed when driving a vehicle on the road in order to reduce fatalities. Thus, it is necessary to have a mechanism that predicts the severity of such accidents and helps in reducing fatalities.

1.1 problem statement

The target feature is “Accident\_severity,” which is a multi-class variable. The task is to classify this variable based on the other 31 features step-by-step by going through each data science process and task. Our metric for evaluation will be your “F1 score.”

#### 1.2 Prerequisites

This is an **intermediate-level project** with an **imbalanced multiclass classification problem**. some of the prerequisites for this project.

* Understanding of classification machine learning algorithms is required.
* Knowledge of the Python programming language and python frameworks like Pandas, NumPy, Matplotlib, and Scikit-Learn, as well as the basics of streamlit library, is required.

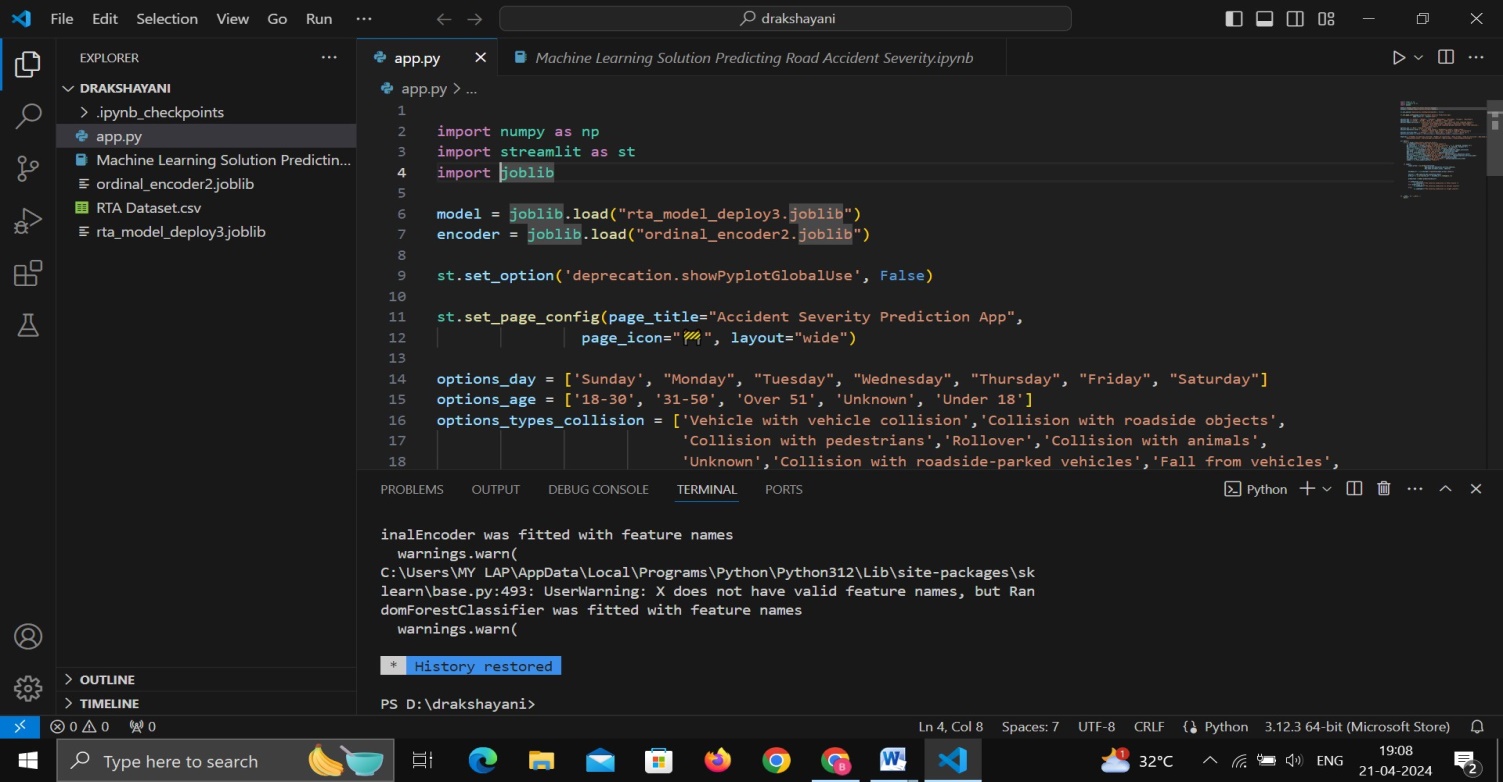
#### Dataset description :

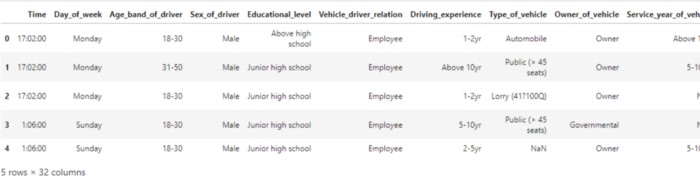
1. Time — time of the accident (In 24 hours format)
2. Day\_of\_week — A day when an accident occurred
3. Age\_band\_of\_driver —The age group of the driver
4. Sex\_of\_driver — Gender of driver
5. Educational\_level — Driver’s highest education level
6. Vehical\_driver\_relation — What’s the relation of a driver with the vehicle
7. Driving\_experience — How many years of driving experience the driver has
8. Type\_of\_vehicle — What’s the type of vehicle
9. Owner\_of\_vehicle — Who’s the owner of the vehicle
10. Service\_year\_of\_vehicle — The last service year of the vehicle
11. Defect\_of\_vehicle — Is there any defect on the vehicle or not?
12. Area\_accident\_occured — Locality of an accident site
13. Lanes\_or\_Medians — Are there any lanes or medians at the accident site?
14. Road\_allignment — Road alignment with the terrain of the land
15. Types\_of\_junction — Type of junction at the accident site
16. Road\_surface\_type — A surface type of road
17. Road\_surface\_conditions — What was the condition of the road surface?
18. Light\_conditions — Lighting conditions at the site
19. Weather\_conditions — Weather situation at the site of an accident
20. Type\_of\_collision — What is the type of collision
21. Number\_of\_vehicles\_involved — Total number of vehicles involved in an accident
22. Number\_of\_casualties — Total number of casualties in an accident
23. Vehicle\_movement — How the vehicle was moving before the accident occurred
24. Casualty\_class — A person who got killed during an accident
25. Sex\_of\_casualty — What the gender of a person who got killed
26. Age\_band\_of\_casualty — Age group of casualty
27. Casualty\_severtiy — How severely the casualty was injured
28. Work\_of\_casualty — What was the work of the casualty
29. Fitness\_of\_casualty — Fitness level of casualty
30. Pedestrain\_movement — Was there any pedestrian movement on the road?
31. Cause\_of-accident — What was the cause of an accident?
32. Accident\_severity — How severe an accident was? *(Target variable)*

So far, we have understood the problem statement and data descriptions. Now we will look at the end-to-end code implementations to solve predictive analytics problems and deploy machine learning solutions on the streamlit cloud.

## Analysing Datasets in Machine Learning Solution

Now that we have looked at the problem statement and data description, we will start loading the dataset into our development environment and start analyzing the data to find critical insights for the data preparation and modeling stages



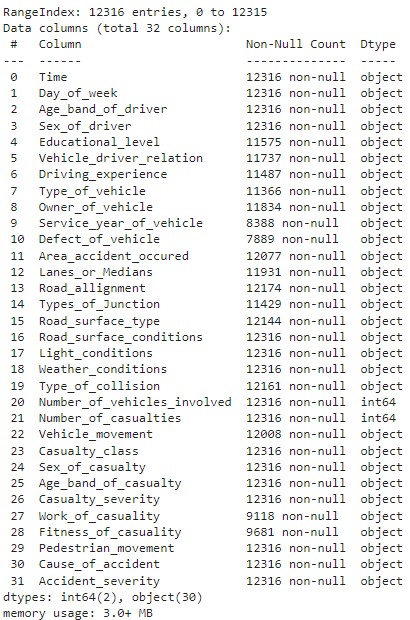


The output of the code

**. 2. Metadata of the dataset**

# print the dataset information

df.info()



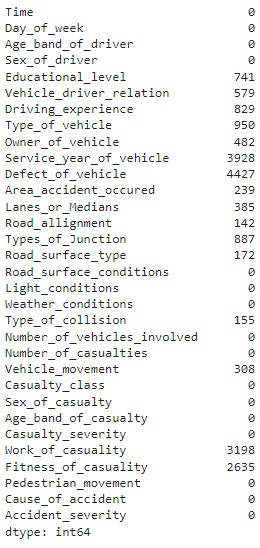
The output of the above code

The above method shows metadata information such as non-null values, datatypes of each column, the number of rows and columns present in the dataset, and the memory usage of the dataset.

**3. Find the number of missing values present in each column**

# Find the number of missing values present in each column

df.isnull().sum()



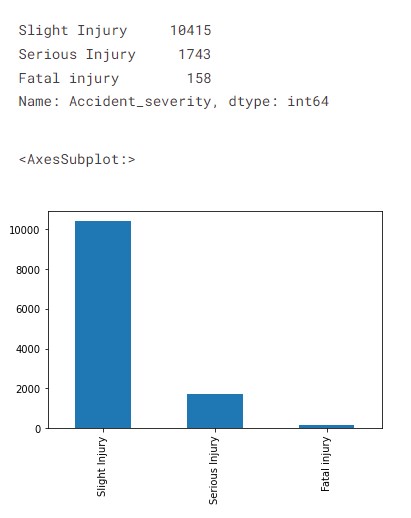
The method shows us how many missing values there are in each column. “Defect\_of\_Vehicle” shows the highest number of missing values, which is 4427 out of 12316 instances.

**4. Target variable classes distribution and visualization**

# target variable classes counts and bar plot

print(df['Accident\_severity'].value\_counts())

df['Accident\_severity'].value\_counts().plot(kind='bar')



As we saw earlier in the problem statement, target variable classes are highly imbalanced, and we will solve this problem during the data preparation stage to develop accurate and generalized machine learning models.

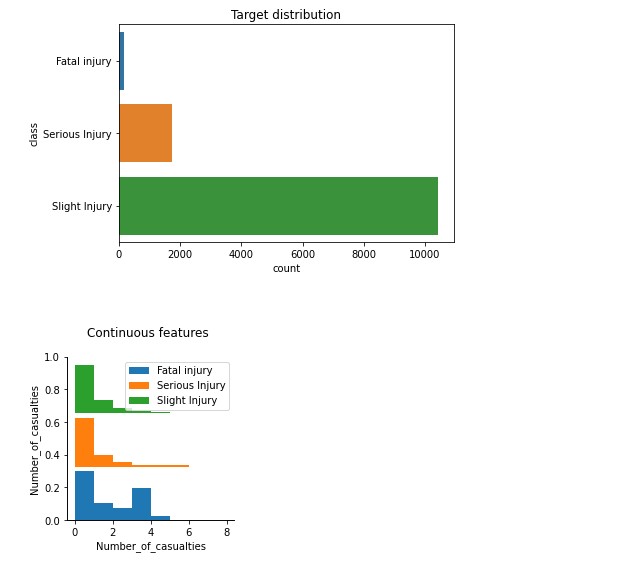
Auto data visualization using the ‘dabl’ library

#Visualizing dataset using dabl library

pip install dabl

import dabl

dabl.plot(df, target\_col='Accident\_severity')



Using just one line of code, we can visualize the relationships between input features and a target variable. From our analysis so far, we can derive the following insights:

* More the Number of casualties, the higher the chances of fatal injuries at the accident site
* More the vehicles involved higher the chances of Serious injury
* Light\_conditions being darkness can cause higher serious injury
* Data is highly imbalanced
* Features like area\_accident\_occured, Cause\_of\_accident, Day\_of\_week, type\_of\_junction seem to be essential features causing fatal injuries
* Road\_surface and road conditions do not affect severe or fatal accidents apparently

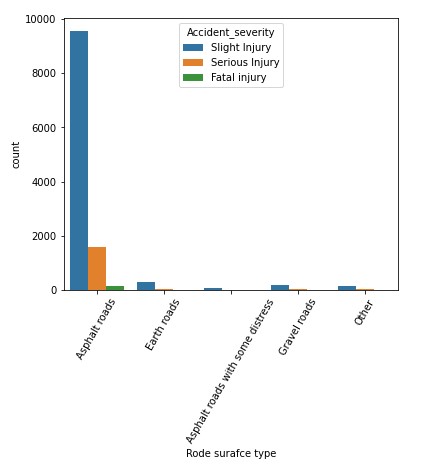
Association between the ‘road surface type’ column and target ‘accident severity’

# plot the bar plot of road\_surface\_type and accident severity feature

plt.figure(figsize=(6,5))

sns.countplot(x='Road\_surface\_type', hue='Accident\_severity', data=df)

plt.xlabel('Rode surafce type')



We can learn that most accidents happened on “asphalt roads” in our dataset, followed by “earth roads.” Here we can say that most fatal injuries occur on asphalt roads, so they might not be a significant variable to predict the target class.

Now, based on the above findings, we will preprocess the raw data for modeling and evaluation purposes.

## Data Preparation

We will start pre-processing the dataset by changing the “Time” column datatype to the “datetime” datatype. We will then extract the hour of the day feature to prepare the data for modeling.

# convert object type column into datetime datatype column

df['Time'] = pd.to\_datetime(df['Time'])

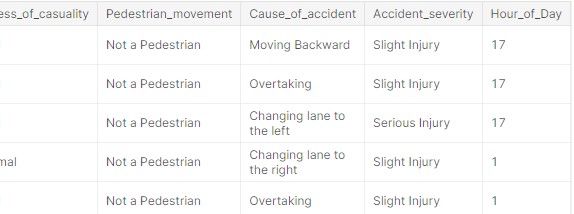
# Extrating 'Hour\_of\_Day' feature from the Time column

new\_df = df.copy()

new\_df['Hour\_of\_Day'] = new\_df['Time'].dt.hour

n\_df = new\_df.drop('Time', axis=1)

n\_df.head()



#### Missing value treatment

#### By this time, we would have a subset of features to be selected for further processing based on the insights gained from the previous stage. We will select a subset of features and then treat the missing values using the “fillna()” method of the Pandas library.

In our context, we will handle missing values by filling in “unknown” as a value, assuming missing values might not have been found during the investigation.

# NaN are missing because service info might not be available, we will fill as 'Unknowns'

feature\_df['Service\_year\_of\_vehicle'] = feature\_df['Service\_year\_of\_vehicle'].fillna('Unknown')

feature\_df['Types\_of\_Junction'] = feature\_df['Types\_of\_Junction'].fillna('Unknown')

feature\_df['Area\_accident\_occured'] = feature\_df['Area\_accident\_occured'].fillna('Unknown')

feature\_df['Driving\_experience'] = feature\_df['Driving\_experience'].fillna('unknown')

feature\_df['Type\_of\_vehicle'] = feature\_df['Type\_of\_vehicle'].fillna('Other')

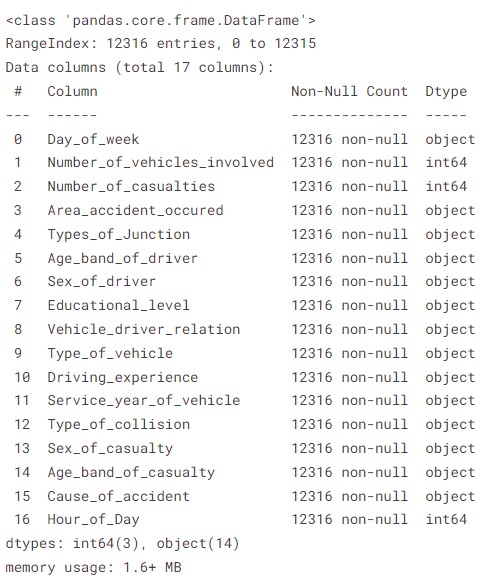
feature\_df['Vehicle\_driver\_relation'] = feature\_df['Vehicle\_driver\_relation'].fillna('Unknown')

feature\_df['Educational\_level'] = feature\_df['Educational\_level'].fillna('Unknown')

feature\_df['Type\_of\_collision'] = feature\_df['Type\_of\_collision'].fillna('Unknown')

# features information

feature\_df.info()



As we can see, our feature set now has zero null values compared to the previous raw dataset. Let’s move on to the other data pre-processing steps now.

#### One-Hot encoding using ‘get\_dummies()’ method

Pandas’ ‘get\_dummies()’ can be used to convert categorical columns into numerical features. let’s see the code snippet below:

# Categorical features to encode using one hot encoding

features = ['Day\_of\_week','Number\_of\_vehicles\_involved','Number\_of\_casualties','Area\_accident\_occured',

'Types\_of\_Junction','Age\_band\_of\_driver','Sex\_of\_driver','Educational\_level',

'Vehicle\_driver\_relation','Type\_of\_vehicle','Driving\_experience','Service\_year\_of\_vehicle','Type\_of\_collision',

'Sex\_of\_casualty','Age\_band\_of\_casualty','Cause\_of\_accident','Hour\_of\_Day']

# setting input features X and target y

X = feature\_df[features] # here features are selected from 'object' datatype

y = n\_df['Accident\_severity']

# we will use pandas get\_dummies method for on-hot encoding

encoded\_df = pd.get\_dummies(X, drop\_first=True)

encoded\_df.shape

-----------------------------------[Output]-----------------------------------

(12316, 106)

As a result of the encoding, we now have 106 columns in the encoded dataframe, which we will further preprocess to keep only significant features for modeling purposes.

#### Target encoding using ‘LabelEncoder()’ method

# import labelencoder from sklearn.preprocessing

from sklearn.preprocessing import LabelEncoder

# create labelencoder object

lb = LabelEncoder()

lb.fit(y)

y\_encoded = lb.transform(y)

print("Encoded labels:",lb.classes\_)

y\_en = pd.Series(y\_encoded)

-----------------------------------[Output]----------------------------------

Encoded labels: ['Fatal injury' 'Serious Injury' 'Slight Injury']

Now, we have “encoded\_df” as an encoded dataframe object and “y\_en” as an encoded target column to further select “Kbest” features and handle an imbalanced dataset.

#### Feature selection using the ‘Chi2’ Statistic

# feature seleciton method using chi2 for categorical output, categorical input

from sklearn.feature\_selection import SelectKBest, chi2

fs = SelectKBest(chi2, k=50)

X\_new = fs.fit\_transform(encoded\_df, y\_en)

# Take the selected features

cols = fs.get\_feature\_names\_out()

# convert selected features into dataframe

fs\_df = pd.DataFrame(X\_new, columns=cols)

We are selecting the top 50 features out of 106 features from the encoded dataframe and storing them in a new dataframe object called “fs\_df.” The “Chi2” statistic is used when the target feature is a categorical variable, and the “Pearson’s coefficient” is used when the target feature is a continuous variable. Now let’s upsample the dataset to balance categorical features.

## Machine Learning Modeling

Now, let’s develop a classification machine learning model using a random forest machine learning algorithm. We will import the Scikit-Learn library’s various classes to develop an ML model and evaluate it in the next step.

# import the necessary liabrary

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, classification\_report, f1\_score

# train and test split and building baseline model to predict target features

X\_trn, X\_tst, y\_trn, y\_tst = train\_test\_split(X\_n, y\_n, test\_size=0.2, random\_state=42)

# modelling using random forest baseline

rf = RandomForestClassifier(n\_estimators=800, max\_depth=20, random\_state=42)

rf.fit(X\_trn, y\_trn)

# predicting on test data

predics = rf.predict(X\_tst)

# train score

rf.score(X\_trn, y\_trn)

-----------------------------------[Output]----------------------------------

0.9416306609057449

Now, we have developed a random forest model with n\_estimators = 800 and max\_depth = 20. We will evaluate this model on test data to validate the results and make predictions based on new input data.

## Evaluating ML Model on Test Data

Let’s print the classification report on the test dataset.

# classification report on test dataset

classif\_re = classification\_report(y\_tst,predics)

print(classif\_re)

-----------------------------------[Output]----------------------------------

precision recall f1-score support

0 0.94 0.96 0.95 2085

1 0.84 0.83 0.84 2100

2 0.86 0.87 0.86 2064

accuracy 0.88 6249

macro avg 0.88 0.88 0.88 6249

weighted avg 0.88 0.88 0.88 6249

# f1\_score of the model

f1score = f1\_score(y\_tst,predics, average='weighted')

print(f1score)

-----------------------------------[Output]----------------------------------

0.8838187418909502

We can see that the model archives 88% accuracy on test data compared to 94% on the training dataset. Our model seems to perform well on the test dataset, and we are good to go with deploying the model on the streamlit cloud to make it accessible to end users.

## Deployment of the Dataset Using Streamlit

Now that we have analyzed the data and built a machine learning model with 88% of the f1\_score on the test dataset, we will go ahead with the ML pipeline to develop a web interface using the streamlit library. Creating machine learning-powered web applications using Streamlit is super easy, as you don’t have to know any front-end technologies. Moreover, Streamlit provides free cloud deployment with custom sub-domain names, which makes it a preferable option for deploying ML-powered applications.

Let’s start with saving the model object and selecting input features. We have selected 10 features for building web applications to infer the machine learning model from the web application. Apart from this, we will also save the ordinal encoder object to transform categorical inputs to their respective encodings.

# selecting 7 categorical features from the dataframe

import joblib

from sklearn.preprocessing import OrdinalEncoder

new\_fea\_df = feature\_df[['Type\_of\_collision','Age\_band\_of\_driver','Sex\_of\_driver',

'Educational\_level','Service\_year\_of\_vehicle','Day\_of\_week','Area\_accident\_occured']]

oencoder2 = OrdinalEncoder()

encoded\_df3 = pd.DataFrame(oencoder2.fit\_transform(new\_fea\_df))

encoded\_df3.columns = new\_fea\_df.columns

# save the ordinal encoder object for inference pipeline

joblib.dump(oencoder, "ordinal\_encoder2.joblib")

Now, we will combine seven categorical features with three numerical features to train the final model for inference.

# final dataframe to be trained for model inference

s\_final\_df = pd.concat([feature\_df[['Number\_of\_vehicles\_involved','Number\_of\_casualties','Hour\_of\_Day']],encoded\_df3], axis=1)

# train and test split and building baseline model to predict target features

X\_trn2, X\_tst2, y\_trn2, y\_tst2 = train\_test\_split(s\_final\_df, y\_en, test\_size=0.2, random\_state=42)

# modelling using random forest baseline

rf = RandomForestClassifier(n\_estimators=700, max\_depth=20, random\_state=42)

rf.fit(X\_trn2, y\_trn2)

# save the model object

joblib.dump(rf, "rta\_model\_deploy3.joblib", compress=9)

We will load below four files into our project repository:

1. **requirements.txt**
2. **app.py**
3. **ordinal\_encoder2.joblib**
4. **rta\_model\_deploy3.joblib**

**Load all the dependencies in requirements.txt**

pandas

numpy

streamlit

scikit-learn

joblib

shap

matplotlib

ipython

Pillow

Create an app.py file and write an inference pipeline with a form-based user interface to take inputs from the end-user to predict the severity of road accidents.

# import all the app dependencies

import pandas as pd

import numpy as np

import sklearn

import streamlit as st

import joblib

import shap

import matplotlib

from IPython import get\_ipython

from PIL import Image

# load the encoder and model object

model = joblib.load("rta\_model\_deploy3.joblib")

encoder = joblib.load("ordinal\_encoder2.joblib")

st.set\_option('deprecation.showPyplotGlobalUse', False)

# 1: serious injury, 2: Slight injury, 0: Fatal Injury

st.set\_page\_config(page\_title="Accident Severity Prediction App",

page\_icon="", layout="wide")

#creating option list for dropdown menu

options\_day = ['Sunday', "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

options\_age = ['18-30', '31-50', 'Over 51', 'Unknown', 'Under 18']

# number of vehical involved: range of 1 to 7

# number of casualties: range of 1 to 8

# hour of the day: range of 0 to 23

options\_types\_collision = ['Vehicle with vehicle collision','Collision with roadside objects',

'Collision with pedestrians','Rollover','Collision with animals',

'Unknown','Collision with roadside-parked vehicles','Fall from vehicles',

'Other','With Train']

options\_sex = ['Male','Female','Unknown']

options\_education\_level = ['Junior high school','Elementary school','High school',

'Unknown','Above high school','Writing & reading','Illiterate']

options\_services\_year = ['Unknown','2-5yrs','Above 10yr','5-10yrs','1-2yr','Below 1yr']

options\_acc\_area = ['Other', 'Office areas', 'Residential areas', ' Church areas',

' Industrial areas', 'School areas', ' Recreational areas',

' Outside rural areas', ' Hospital areas', ' Market areas',

'Rural village areas', 'Unknown', 'Rural village areasOffice areas',

'Recreational areas']

# features list

features = ['Number\_of\_vehicles\_involved','Number\_of\_casualties','Hour\_of\_Day','Type\_of\_collision','Age\_band\_of\_driver','Sex\_of\_driver',

'Educational\_level','Service\_year\_of\_vehicle','Day\_of\_week','Area\_accident\_occured']

Once, define all the inputs to be taken from the user we can define the ‘main()’ function to develop UI that will be rendered on the front end.

# Give a title to web app using html syntax

st.markdown("

**Accident Severity Prediction App**

", unsafe\_allow\_html=True)

# define a main() function to take inputs from user in form based approch

def main():

with st.form("road\_traffic\_severity\_form"):

st.subheader("Pleas enter the following inputs:")

No\_vehicles = st.slider("Number of vehicles involved:",1,7, value=0, format="%d")

No\_casualties = st.slider("Number of casualities:",1,8, value=0, format="%d")

Hour = st.slider("Hour of the day:", 0, 23, value=0, format="%d")

collision = st.selectbox("Type of collision:",options=options\_types\_collision)

Age\_band = st.selectbox("Driver age group?:", options=options\_age)

Sex = st.selectbox("Sex of the driver:", options=options\_sex)

Education = st.selectbox("Education of driver:",options=options\_education\_level)

service\_vehicle = st.selectbox("Service year of vehicle:", options=options\_services\_year)

Day\_week = st.selectbox("Day of the week:", options=options\_day)

Accident\_area = st.selectbox("Area of accident:", options=options\_acc\_area)

submit = st.form\_submit\_button("Predict")

# encode using ordinal encoder and predict

if submit:

input\_array = np.array([collision,

Age\_band,Sex,Education,service\_vehicle,

Day\_week,Accident\_area], ndmin=2)

encoded\_arr = list(encoder.transform(input\_array).ravel())

num\_arr = [No\_vehicles,No\_casualties,Hour]

pred\_arr = np.array(num\_arr + encoded\_arr).reshape(1,-1)

# predict the target from all the input features

prediction = model.predict(pred\_arr)

if prediction == 0:

st.write(f"The severity prediction is Fatal Injury")

elif prediction == 1:

st.write(f"The severity prediction is serious injury")

else:

st.write(f"The severity prediction is slight injury")

st.subheader("Explainable AI (XAI) to understand predictions")

# Explainable AI using shap library

shap.initjs()

shap\_values = shap.TreeExplainer(model).shap\_values(pred\_arr)

st.write(f"For prediction {prediction}")

shap.force\_plot(shap.TreeExplainer(model).expected\_value[0], shap\_values[0],

pred\_arr, feature\_names=features, matplotlib=True,show=False).savefig("pred\_force\_plot.jpg", bbox\_inches='tight')

img = Image.open("pred\_force\_plot.jpg")

# render the shap plot on front-end to explain predictions

st.image(img, caption='Model explanation using shap')

st.write("Developed By: Avi kumar Talaviya")

st.markdown("""Reach out to me on: [Twitter](https://twitter.com/avikumart\_) |

[Linkedin](https://www.linkedin.com/in/avi-kumar-talaviya-739153147/) |

[Kaggle](https://www.kaggle.com/avikumart)

""")

Finally, write down the project description and problem statement on the front end to showcase the project you have worked on clearly.

a,b,c = st.columns([0.2,0.6,0.2])

with b:

st.image("vllkyt19n98psusds8.jpg", use\_column\_width=True)

# description about the project and code files

st.subheader("Description:")

st.text("""This data set is collected from Addis Ababa Sub-city police departments for master's research work.

The data set was prepared from manual documents of road traffic accidents of the year 2017-20.

All the sensitive information has been excluded during data encoding and finally it has 32 features and 12316 instances of the accident.

Then it is preprocessed and for identification of major causes of the accident by analyzing it using different machine learning classification algorithms.

""")

st.markdown("Source of the dataset: [Click Here](https://www.narcis.nl/dataset/RecordID/oai%3Aeasy.dans.knaw.nl%3Aeasy-dataset%3A191591)")

st.subheader(" Problem Statement:")

st.text("""The target feature is Accident\_severity which is a multi-class variable.

The task is to classify this variable based on the other 31 features step-by-step by going through each day's task.

The metric for evaluation will be f1-score

""")

st.markdown("Please find GitHub repository link of project: [Click Here](https://github.com/avikumart/Road-Traffic-Severity-Classification-Project)")

# run the main function

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Conclusion**

In conclusion, the end-to-end data science and machine learning project successfully presented the potential for using data analysis and prediction to aid in the prevention of road accident fatalities. By thoroughly analyzing the provided data and training a machine learning model to predict the severity of potential accidents, the investigation agency can prioritize its efforts and resources toward the most high-risk situations. This project highlights the value of using data-driven approaches to address complex problems and the importance of continued investment in these initiatives. Let’s look at the key takeaways from this article.