Applied Capstone Project

Divya Ravindran

21-Sep-2020

1.Introduction

1.1 Background

Seattle Metropolitan Traffic Police have a huge dataset consisting of motor vehicle accident information. In the last 5 years they have handled a number of accident cases that could have been prevented. Hence, we are going to develop a model that will help prevent such incidents with a reasonable accuracy.

When driving to another city for work or to visit some friends, it maybe rainy and windy, and on the way, you come across a terrible traffic jam on the other side of the highway. Long lines of cars barely moving. As you keep driving, police car start appearing from afar shutting down the highway. Oh, it is an accident and there's a helicopter transporting the ones involved in the crash to the nearest hospital. They must be in critical condition for all of this to be happening. Now, wouldn't it be great if there is something in place that could warn you, given the weather and the road conditions about the possibility of you getting into a car accident and how severe it would be, so that you would drive more carefully or even change your travel if you are able to.

1.2 Problem

The problem statement here is to predict the severity of an accident given the data about the current weather conditions, road conditions and the location. When there is extensive data for these incidents, we might be able to come across a pattern that suggests a high probability for accidents.

1.3 Interest

The project stake holders are the Seattle City corporation for implementation of safety strategies and reduction of fatalities. Stakeholder groups includes state, federal and local government agencies, nongovernmental organisations and regional authorities. Car and life Insurance companies are benefited from the result to have a knowledge of the most recurring accident type and places.

Urgent and Emergency care get data-driven information from this project to plan schedules of emergency room and professional to save the life of injured person. Car manufacturers are another beneficiary of this project if they need the information of car crash and Airbag deployment.

2. Data Acquisition and Cleaning

2.1 Data Sources

The two data files that we need mainly are:

1. Data-collisions.csv
2. Seattle_zipcodes.csv

The first file consists of list of incidents. Each incident has an incident number, X co-ordinate, y co-ordinate, severity code, location address, severity description, road conditions, weather, light conditions and many more details. Click here for accessing the meta data for this file.

I scraped the second file from opendatasoft.com which is a publicly accessed site for US Demographics. It contains the geographic data zipcode, latitude and longitude for Seattle, Washington.

2.2 Data Cleaning

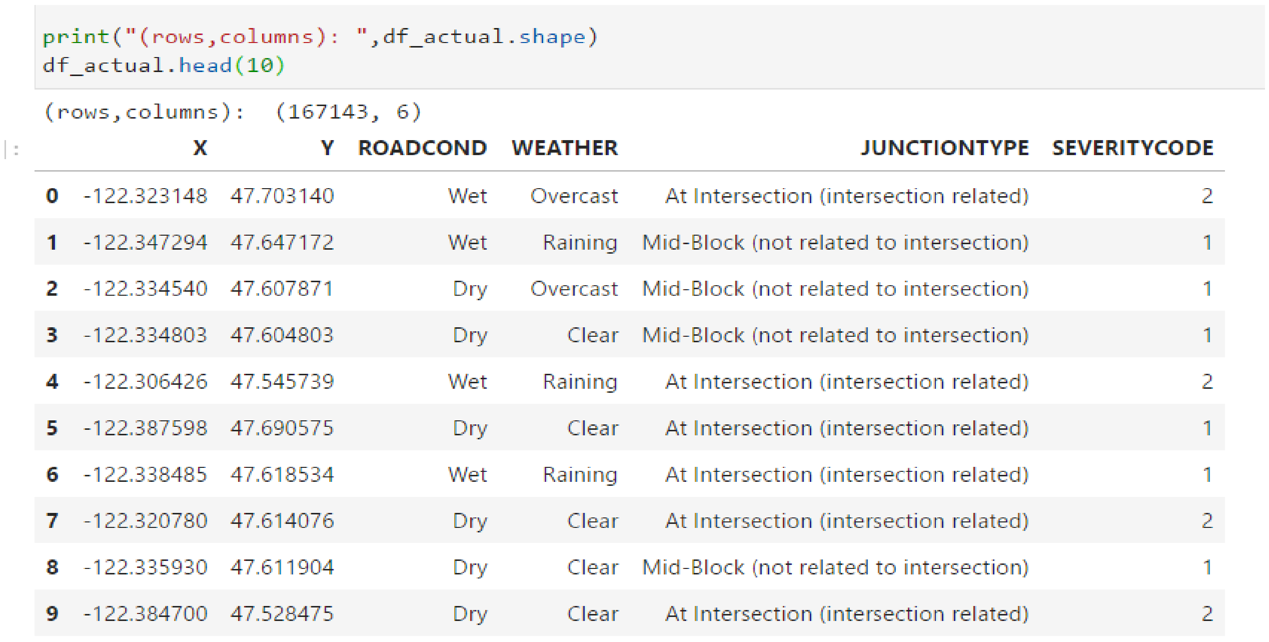
Data downloaded or scraped from multiple sources were combined into one table. There were a lot of missing values and na values in several rows. I decided to remove rows with missing and NA values and only use data from the remaining rows, that came up to approximately 18000 rows. There are several problems with the datasets. First, the latitude and longitude locations of the incidents were given which had to be converted to zipcodes and neighbourhoods. Secondly, the contributing factors like road conditions, weather conditions and junction types were all categorical variables that needed further conversions. In order to fix these issues, I had to remove rows with NA values in the columns, 'X' and 'Y'. 'Unknown' and 'Other' were the other unresolvable values in columns 'Road conditions', 'Junction type' and 'weather conditions' that had to be removed. After data cleaning I checked for outliers in the data. I found there were some extreme outliers, mostly caused by some types of small sample size problem.

The next data file that I scraped from opendatasoft had to be cleaned and checked for duplicates. Duplicate zipcodes needed to be resolved into one neighbourhood and zipcodes were set as index.

2.3 Feature Selection

After cleaning there were 167143 rows and 38 columns. Upon examining the data, it was clear that there were lots of redundancies. There were several columns to indicate the address, like ‘addrtype’ and ‘location’ but what we are really interested in is the latitude and longitude indication which is why they are definitely going inside our feature set. Between ‘Severity code’ and ‘severity desc’ it’s the ‘severity code’ that is easier to include in our feature set as number datatypes help to easily train the model. ‘person count’, ‘pedestrian count’ and ‘pedestrian cycle count’ are 3 redundant columns that we are going to omit, since they don’t really relate to accident severity.

The environmental conditions that contribute to accidents are ‘road conditions’ and ‘weather conditions’ that are categorical variables and would be a part of our feature set. Out of the redundant attributes ‘junction type’ and ‘address type’ we will be having ‘junction type’ as part of our feature set.



```
print("(rows,columns): ",df_actual.shape)
df_actual.head(10)
```

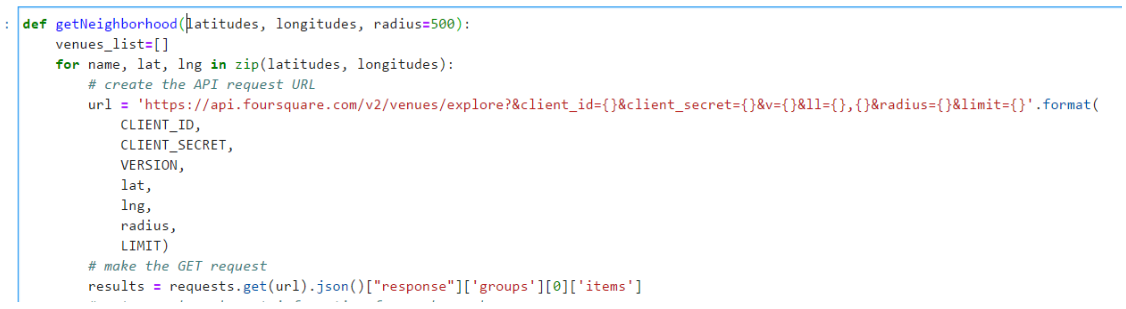
```
(rows,columns):  (167143, 6)
```

	X	Y	ROADCOND	WEATHER	JUNCTIONTYPE	SEVERITYCODE
0	-122.323148	47.703140	Wet	Overcast	At Intersection (intersection related)	2
1	-122.347294	47.647172	Wet	Raining	Mid-Block (not related to intersection)	1
2	-122.334540	47.607871	Dry	Overcast	Mid-Block (not related to intersection)	1
3	-122.334803	47.604803	Dry	Clear	Mid-Block (not related to intersection)	1
4	-122.306426	47.545739	Wet	Raining	At Intersection (intersection related)	2
5	-122.387598	47.690575	Dry	Clear	At Intersection (intersection related)	1
6	-122.338485	47.618534	Wet	Raining	At Intersection (intersection related)	1
7	-122.320780	47.614076	Dry	Clear	At Intersection (intersection related)	2
8	-122.335930	47.611904	Dry	Clear	Mid-Block (not related to intersection)	1
9	-122.384700	47.528475	Dry	Clear	At Intersection (intersection related)	2

2.3Neighbourhood segmentation using Foursquare API

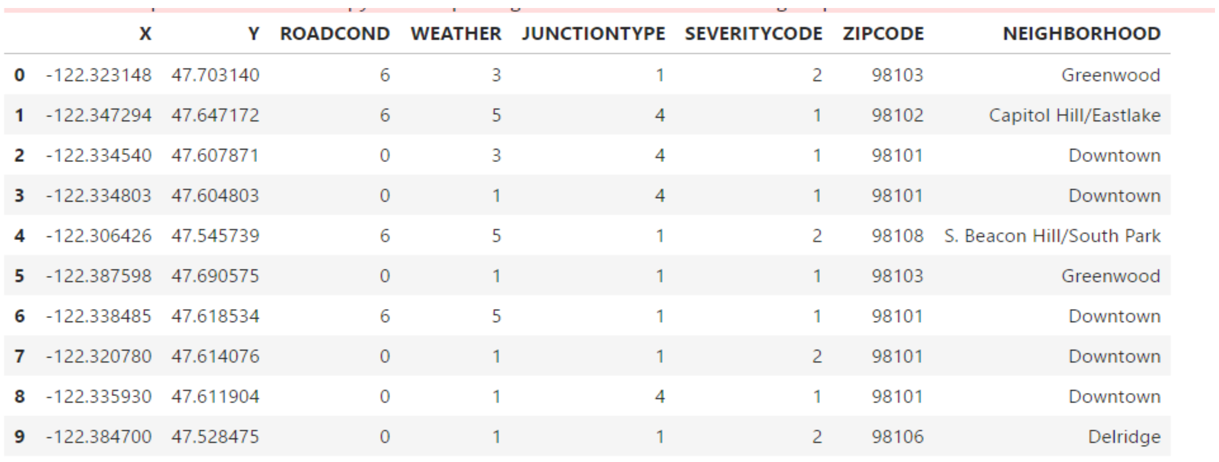
The FoursquareAPI has the complete demographic information required. We only try to obtain the neighbourhood names corresponding to each (x,y) location. Hence, we will be merging our feature set with their corresponding

Neighbourhoods and zipcodes. The Foursquare API can be used using the requests library in python combined with a secret client code and ID for authorization.



```
def getNeighborhood(latitudes, longitudes, radius=500):
    venues_list=[]
    for name, lat, lng in zip(latitudes, longitudes):
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)
        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']
```

And the final feature set looks as follows



	X	Y	ROADCOND	WEATHER	JUNCTIONTYPE	SEVERITYCODE	ZIPCODE	NEIGHBORHOOD
0	-122.323148	47.703140	6	3	1	2	98103	Greenwood
1	-122.347294	47.647172	6	5	4	1	98102	Capitol Hill/Eastlake
2	-122.334540	47.607871	0	3	4	1	98101	Downtown
3	-122.334803	47.604803	0	1	4	1	98101	Downtown
4	-122.306426	47.545739	6	5	1	2	98108	S. Beacon Hill/South Park
5	-122.387598	47.690575	0	1	1	1	98103	Greenwood
6	-122.338485	47.618534	6	5	1	1	98101	Downtown
7	-122.320780	47.614076	0	1	1	2	98101	Downtown
8	-122.335930	47.611904	0	1	4	1	98101	Downtown
9	-122.384700	47.528475	0	1	1	2	98106	Delridge

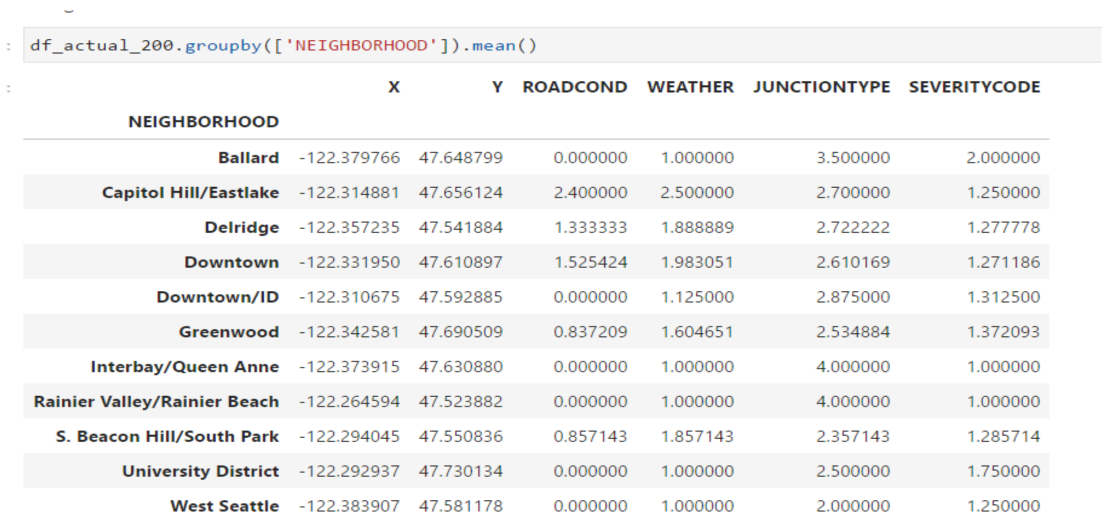
3.Exploratory Data Analysis

3.1 Calculation of Target Variable

The aim of this project is to classify the severity of the of the accident that corresponds to a single incident entry in the data set. A severity code of 1 implies property damage, while a severity code of 2 implies injury or fatality.

3.2 Relationship between Neighbourhood and severity

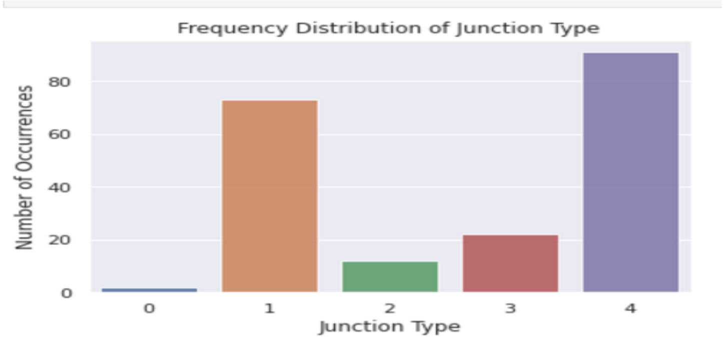
When the feature set is grouped by the neighbourhoods, we can see that the mean of the severity codes varies between 1 and 2. The maximum being 2 which means that all the accidents in that neighbourhoods have injuries or fatalities. Each neighbourhood does not have the exact co-ordinates and are approximated to the nearest centroid of the neighbourhood. Hence this is a classification itself (probably K-Nearest Neighbours) performed by FourSquare API.

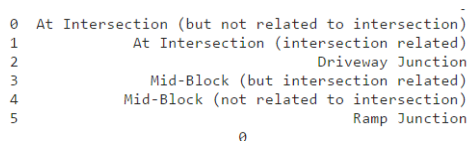


```
df_actual_200.groupby(['NEIGHBORHOOD']).mean()
```

NEIGHBORHOOD	X	Y	ROADCOND	WEATHER	JUNCTIONTYPE	SEVERITYCODE
Ballard	-122.379766	47.648799	0.000000	1.000000	3.500000	2.000000
Capitol Hill/Eastlake	-122.314881	47.656124	2.400000	2.500000	2.700000	1.250000
Delridge	-122.357235	47.541884	1.333333	1.888889	2.722222	1.277778
Downtown	-122.331950	47.610897	1.525424	1.983051	2.610169	1.271186
Downtown/ID	-122.310675	47.592885	0.000000	1.125000	2.875000	1.312500
Greenwood	-122.342581	47.690509	0.837209	1.604651	2.534884	1.372093
Interbay/Queen Anne	-122.373915	47.630880	0.000000	1.000000	4.000000	1.000000
Rainier Valley/Rainier Beach	-122.264594	47.523882	0.000000	1.000000	4.000000	1.000000
S. Beacon Hill/South Park	-122.294045	47.550836	0.857143	1.857143	2.357143	1.285714
University District	-122.292937	47.730134	0.000000	1.000000	2.500000	1.750000
West Seattle	-122.383907	47.581178	0.000000	1.000000	2.000000	1.250000

3.3 Junctiontype Vs. severity code





```
0  At Intersection (but not related to intersection)
1              At Intersection (intersection related)
2                                   Driveway Junction
3                 Mid-Block (but intersection related)
4               Mid-Block (not related to intersection)
5                                       Ramp Junction
                         0
```

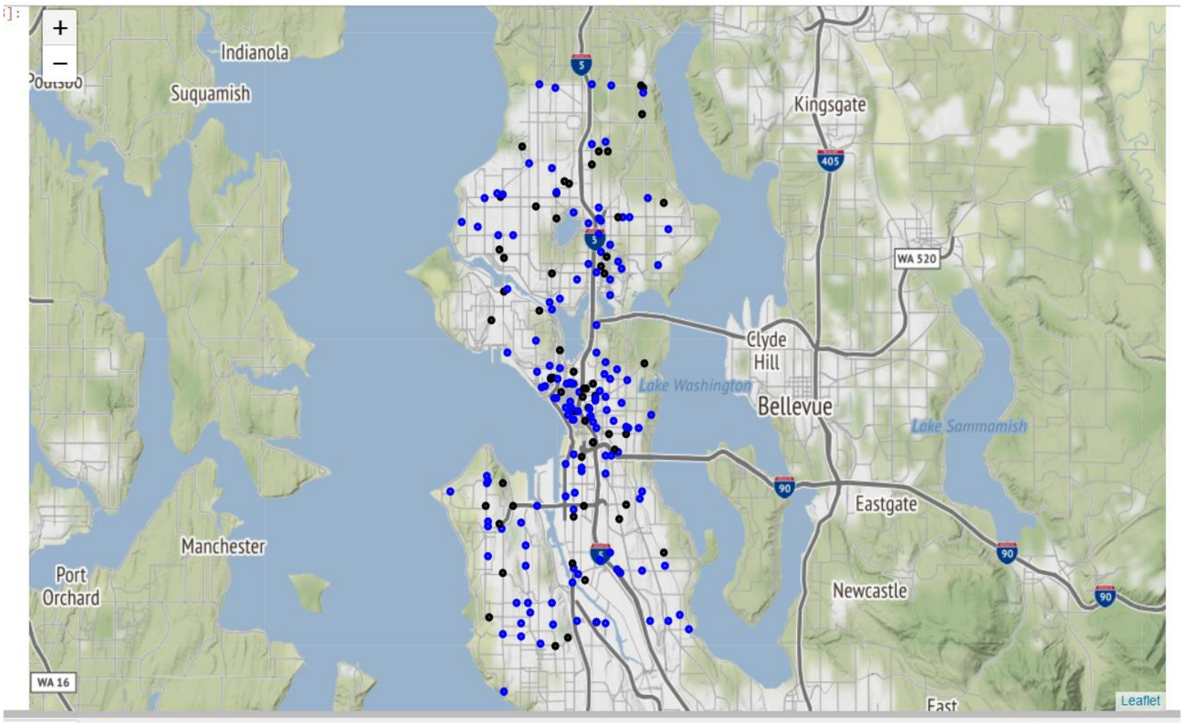
From the above illustration we can see that at ‘Mid-block (not related to intersection)’ junctiontype, we have had the maximum occurrences of accidental incidents. This proves that Junction type too influences the severity of an accident.

4.Predictive Modelling

Since severity code is a categorical value, I have gone with a classification model. Once the feature set has been fixed, we begin data modelling.

Just to get an idea of what we are dealing with, I plot the (x,y) coordinates from the dataset onto a map. The blue circles are severity code = 1 and the black circles are severity code = 2.

Submit Notebook ... Python



4.1Normalization / Train & Test split

Using preprocessing library, we normalize the data to avoid unbalanced data interfering with my predictive modelling.

```
X_data = df_actual_200[['ZIPCODE','ROADCOND','WEATHER','JUNCTIONTYPE']].values
y_data = df_actual_200['SEVERITYCODE']
```

```
X = preprocessing.StandardScaler().fit(X_data).transform(X_data.astype(float))
X[0:5]
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/sklearn/utils/validation.py:595: D
pe object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
array([[-0.2,  2. ,  0.8, -1.2],
       [-0.6,  2. ,  2.3,  1. ],
       [-0.9, -0.5,  0.8,  1. ],
       [-0.9, -0.5, -0.6,  1. ],
       [ 1.4,  2. ,  2.3, -1.2]])
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y_data, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (160, 4) (160,)
Test set: (40, 4) (40,)
```

We use the K-Nearest Neighbour Algorithm to classify the rows into severity code 1 and 2. We randomly set the number of neighbours, k=4. This supervised classification algorithm checks for ‘k’ closest neighbours

```
from sklearn.neighbors import KNeighborsClassifier
k = 4
#Train Model and Predict
knn_model = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
knn_model
```

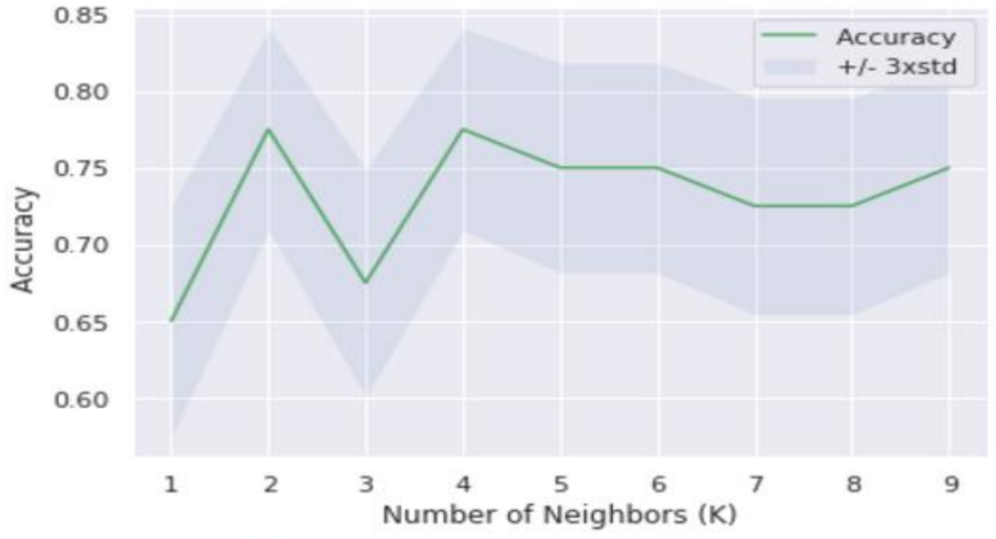
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=4, p=2,
           weights='uniform')
```

```
y_hat = knn_model.predict(X_test)
y_hat[0:5]
```

```
array([1, 1, 1, 2, 1])
```

```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, knn_model.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, y_hat))
```

```
Train set Accuracy:  0.7125
Test set Accuracy:  0.775
```



```
<Figure size 72x72 with 0 Axes>
```

From the above figure, we can see that the value of k=2 and 4 gives the best accuracy score

4.2Different Predictive Modelling results

I have applied the KNN classifier, SVM classifier, Decision Tree classifier and the Logistic Regression classifier to the dataset. The accuracy levels of the target variable differ slightly from model to model. The following table explains the various accuracy levels. The confusion matrix provided 28 true positives out of 40 rows which is a remarkable prediction rate.

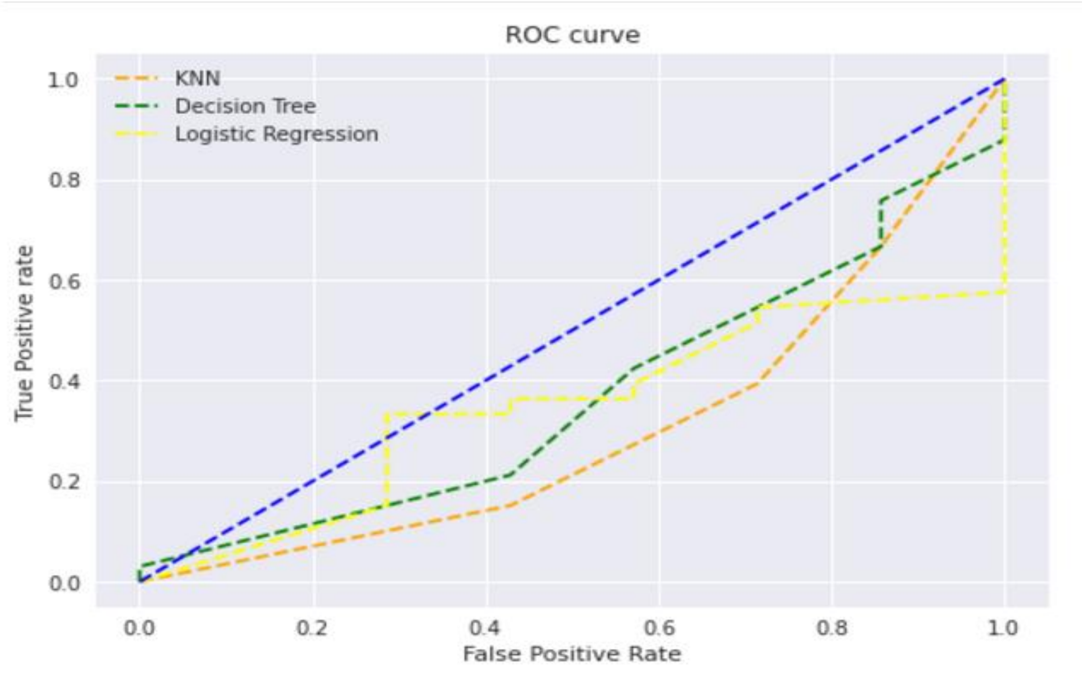
Metrics	**KNN**	**Decision Tree**	**Logistic Regression**	**Support Vector Mechanism**
Accuracy score	0.775	0.725	0.675	0.75

Jacquard_Index	0.775	0.725	0.675	0.75
F1_score	0.78	0.74	0.70	0.75
Cnf_matrix	[[28 5] [4 3]]	[[26 7] [4 3]]	[[25 8] [5 2]]	[[28 5] [5 2]]

Since the occurrence of accidents are closely related to the location where it happens, this scenario correlates closely with the KNN classification model which is found to be the best classifier for accident severity.

4.3 ROC Curve

I also evaluated the models using their ROC curves. In this particular problem, lower false positive rate is more important than higher true positive rate. In other words, it is more important to prevent accidents from occurring rather than to accurately predict its severity. In the ROC curves with low false-positive rate, the KNN model had slightly higher true positive rates than other models



5 Conclusion

In this study I was able to analyse the patterns of road accidents and their severity. With the right prediction system in place many of these accidents can be avoided with the right information in place. This system is highly beneficial for the Traffic police to avert accidents on a day-to-day basis.