Car Accident Severity

Introduction

Severe accidents not only lead to human loss of life but also heavy economic loss to the country. It also consumes a lot of time. At the top, any loss of human life is very dreadful. With the increasing no. of vehicles accidents are also increasing. If we see data there are approximately 1.35 million people who die each year as a result of road traffic crashes. Effect of this on their family members can't be explained. In some cases there are people who managed to survive but they have to suffer there whole life due to some permanent illness caused by the accident.

Now let's take the data of economic loss, the total cost of crashes was approximately $146.3 million. On an average traffic crash costs nations around 3% of their GNP (Gross National Product).

Risk factors for these accidents are mainly excessive speed, presence of alcohol, helmets not worn by users of two wheeled vehicles, roadside objects not crash protective.

Problem

Now wouldn’t it be great if somehow we came to know about the possibility of us getting into an accident. That’s exactly where we are going to work on this project.

This will help numerous people in saving their life. Not only this it would also cut the expenditure of crashes which may also help in the nation’s economy.

Data Description

The data I have used can be found at this link

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

The metadata can be found in the Github repository or at the link given below.

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Metadata.pdf

I have imported the data into a dataframe. The raw data contains 194673 rows and 38 columns/features. There were some rows which I didn’t find useful in prediction and thus they were dropped. Features like ‘REPORTNO’ which don’t have significant use were dropped Duplicate or highly similar columns/features were also dropped.

There were missing values in the row which accounted for approx 2% data and thus they were also dropped.

The main feature which we are going to predict is ‘SEVERITY CODE’ which consists only two classes ‘1’ and ‘2’

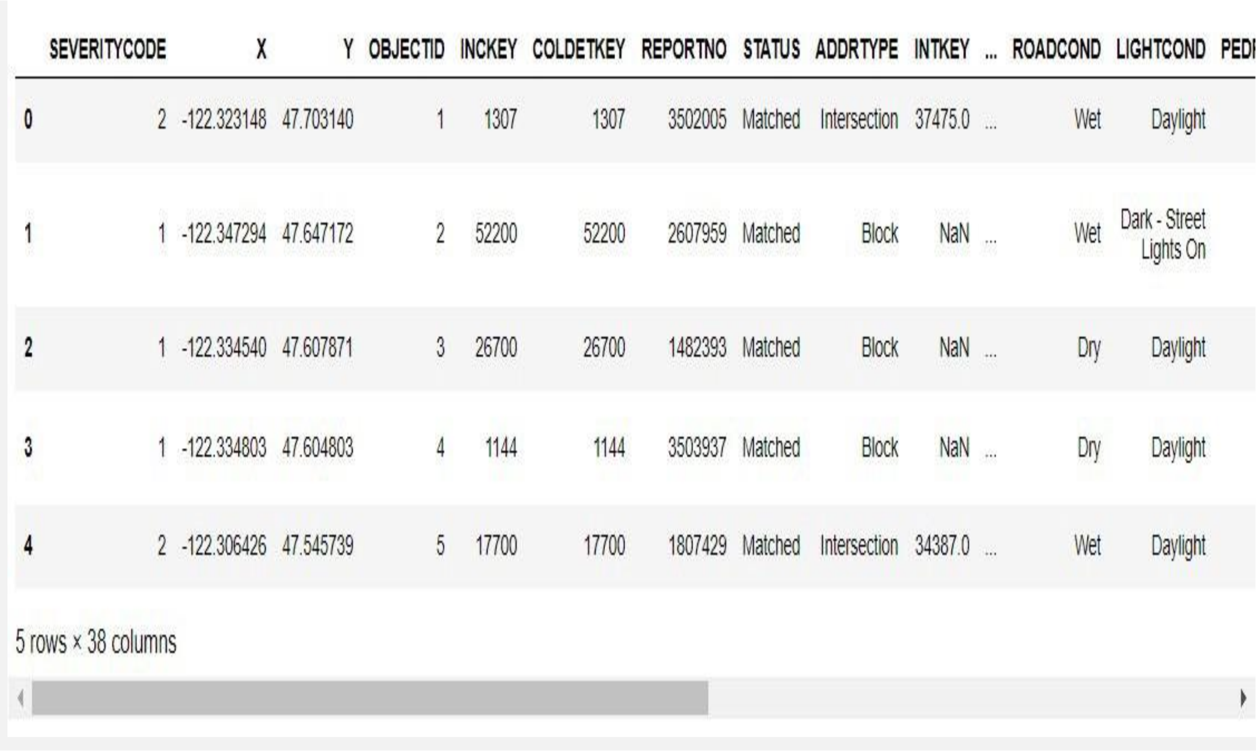
Here ,

- ‘1’represents property damage and

- ‘2’ represents injury

METHODOLOGY

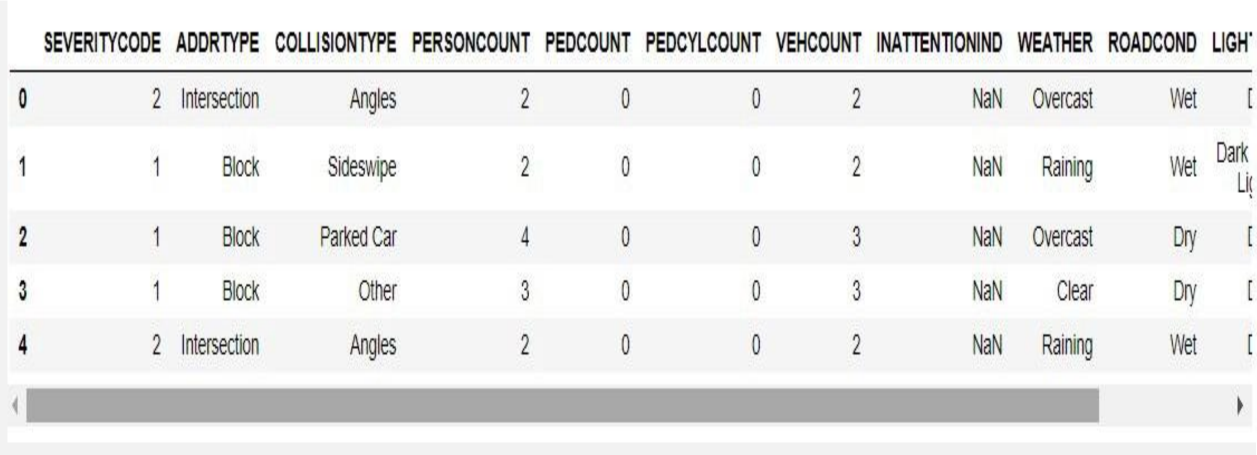
So starting with importing the dataset ,the raw data set looked like this.



	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDI
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight	
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On	
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight	
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight	
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight	

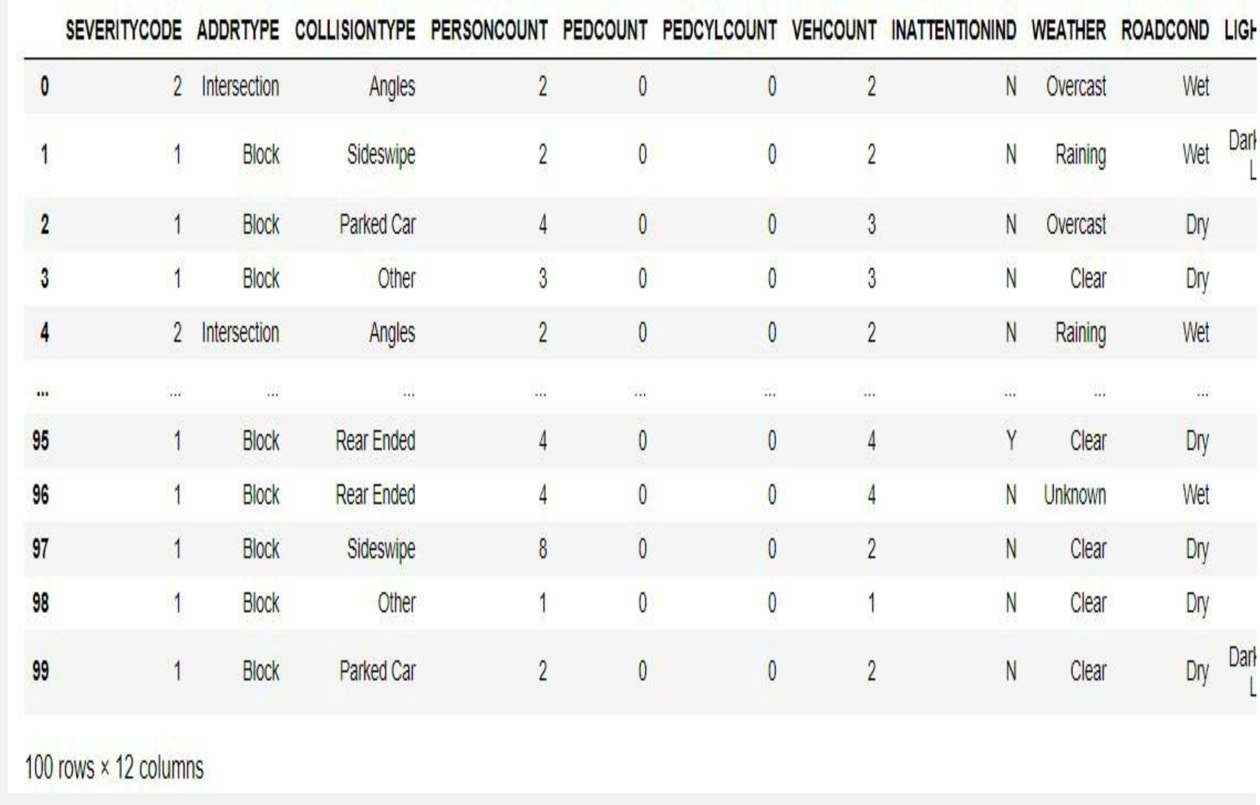
5 rows × 38 columns

And identifying which features I have to keep, some features were dropped.



	SEVERITYCODE	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND	WEATHER	ROADCOND	LIGH
0	2	Intersection	Angles	2	0	0	2	NaN	Overcast	Wet	[
1	1	Block	Sideswipe	2	0	0	2	NaN	Raining	Wet	Dark Li
2	1	Block	Parked Car	4	0	0	3	NaN	Overcast	Dry	[
3	1	Block	Other	3	0	0	3	NaN	Clear	Dry	[
4	2	Intersection	Angles	2	0	0	2	NaN	Raining	Wet	[

Some rows with ‘NaN’ value were dropped or they were replaced with ’N’ depending on the features.



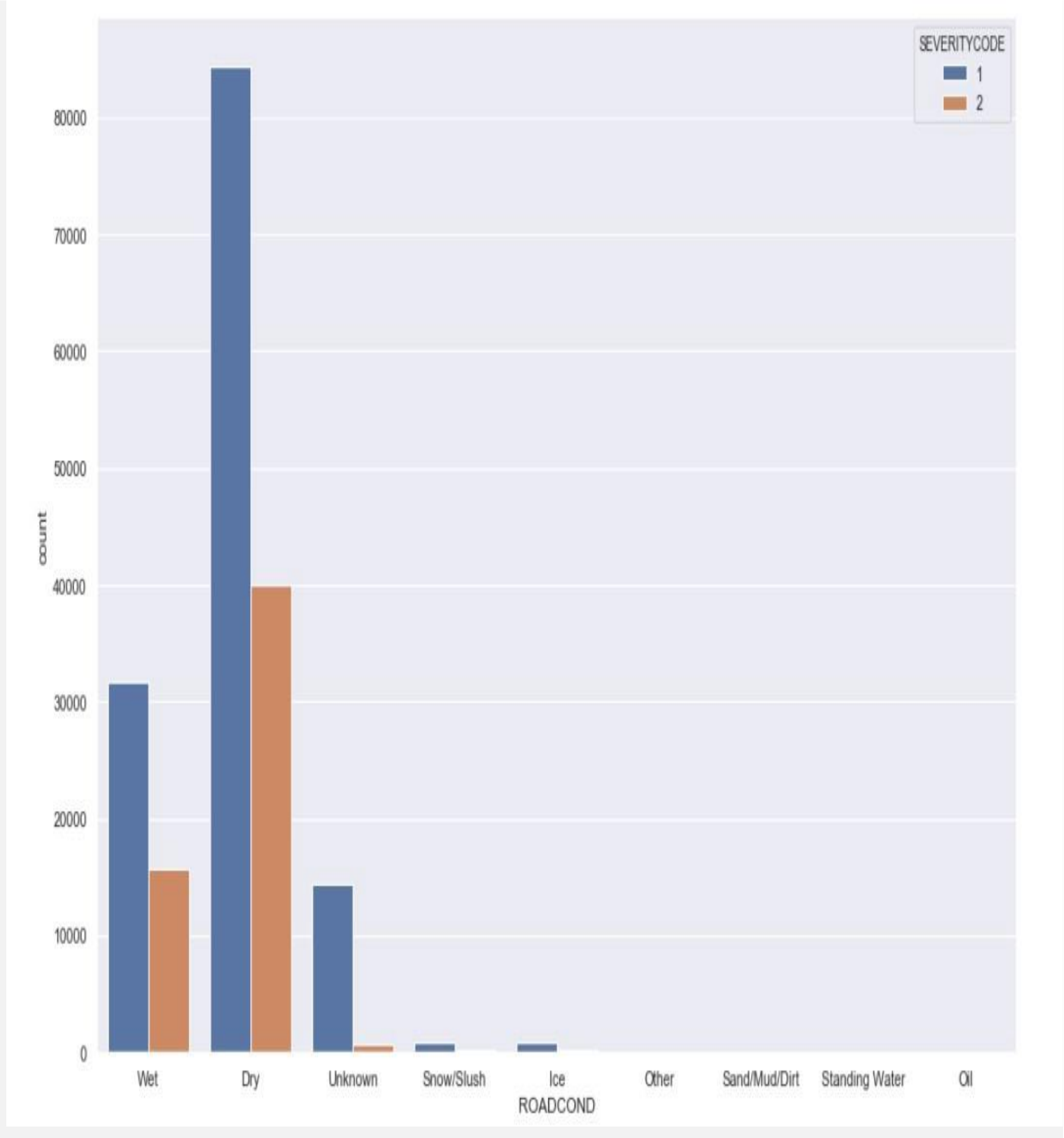
	SEVERITYCODE	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND	WEATHER	ROADCOND	LIGH
0	2	Intersection	Angles	2	0	0	2	N	Overcast	Wet	
1	1	Block	Sideswipe	2	0	0	2	N	Raining	Wet	Dark L
2	1	Block	Parked Car	4	0	0	3	N	Overcast	Dry	
3	1	Block	Other	3	0	0	3	N	Clear	Dry	
4	2	Intersection	Angles	2	0	0	2	N	Raining	Wet	
...	...	...	...	...	...	...	...	...	...	...	
95	1	Block	Rear Ended	4	0	0	4	Y	Clear	Dry	
96	1	Block	Rear Ended	4	0	0	4	N	Unknown	Wet	
97	1	Block	Sideswipe	8	0	0	2	N	Clear	Dry	
98	1	Block	Other	1	0	0	1	N	Clear	Dry	
99	1	Block	Parked Car	2	0	0	2	N	Clear	Dry	Dark L

100 rows × 12 columns

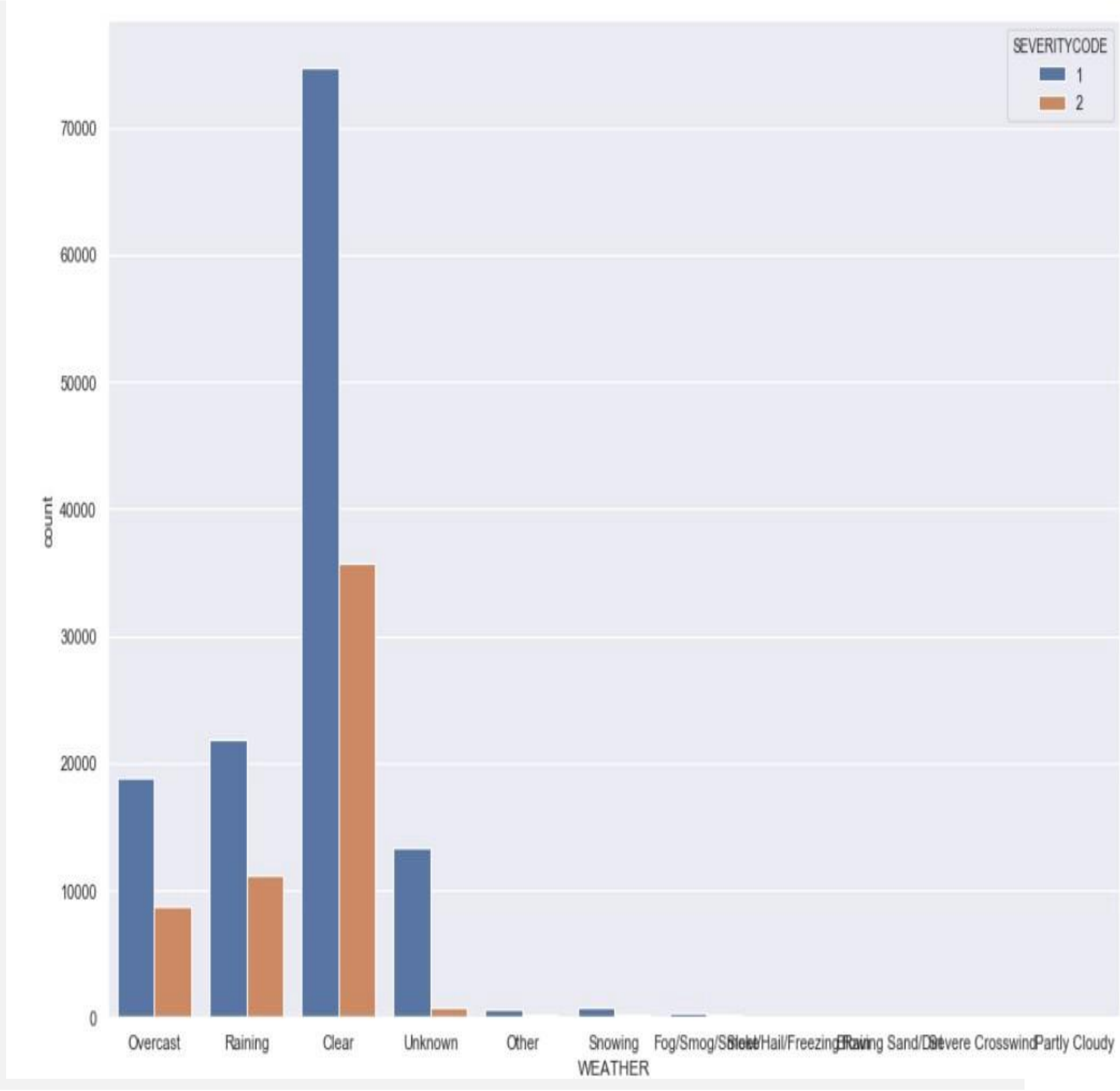
After this I checked the no. of classes in the ‘ROADCOND’ feature. Since there were many features which had many classes I found useful in encoding them.

The visualization of features which can have impact on **SEVERITYCODE** are shown below.

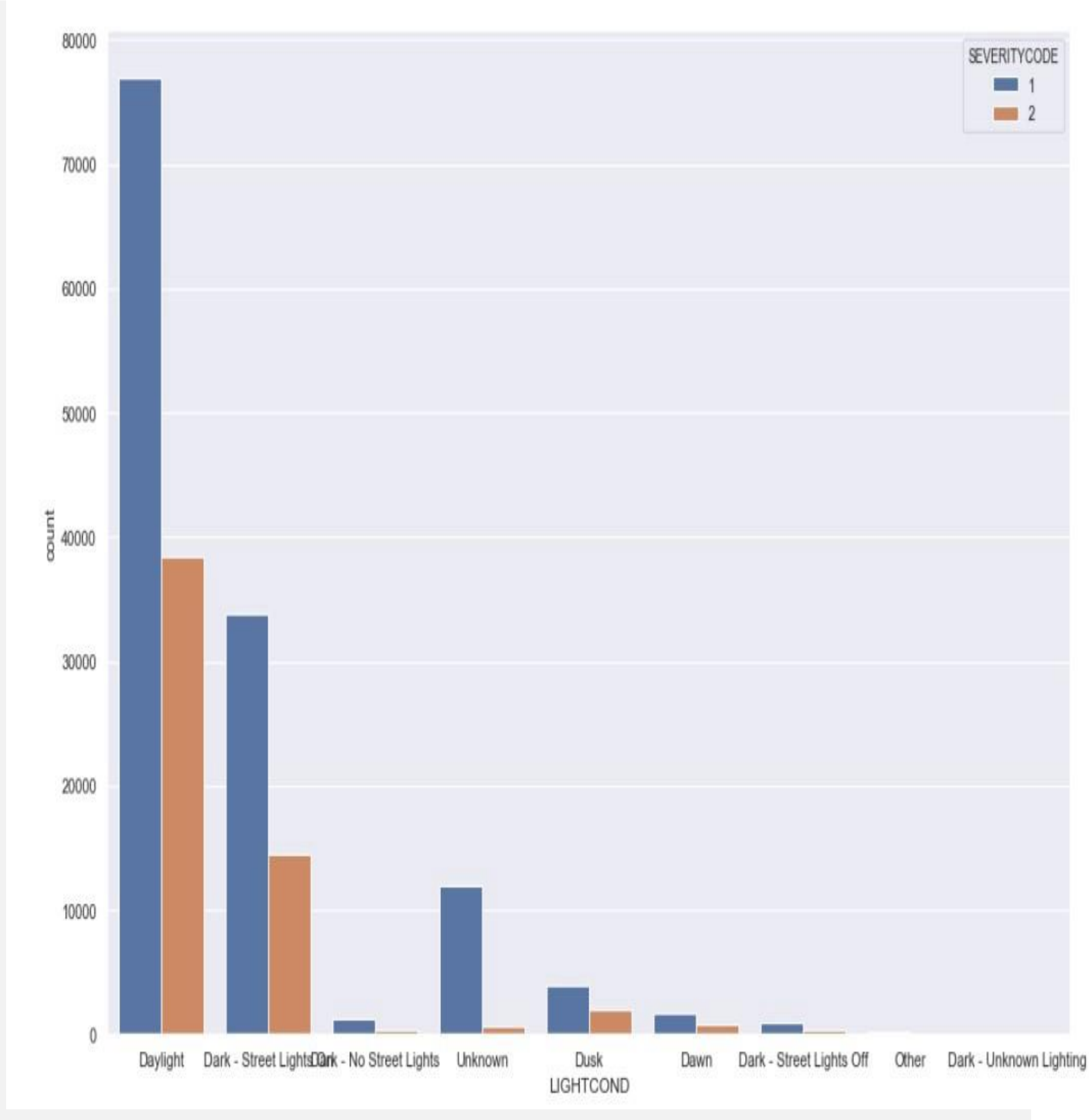
ROADCOND



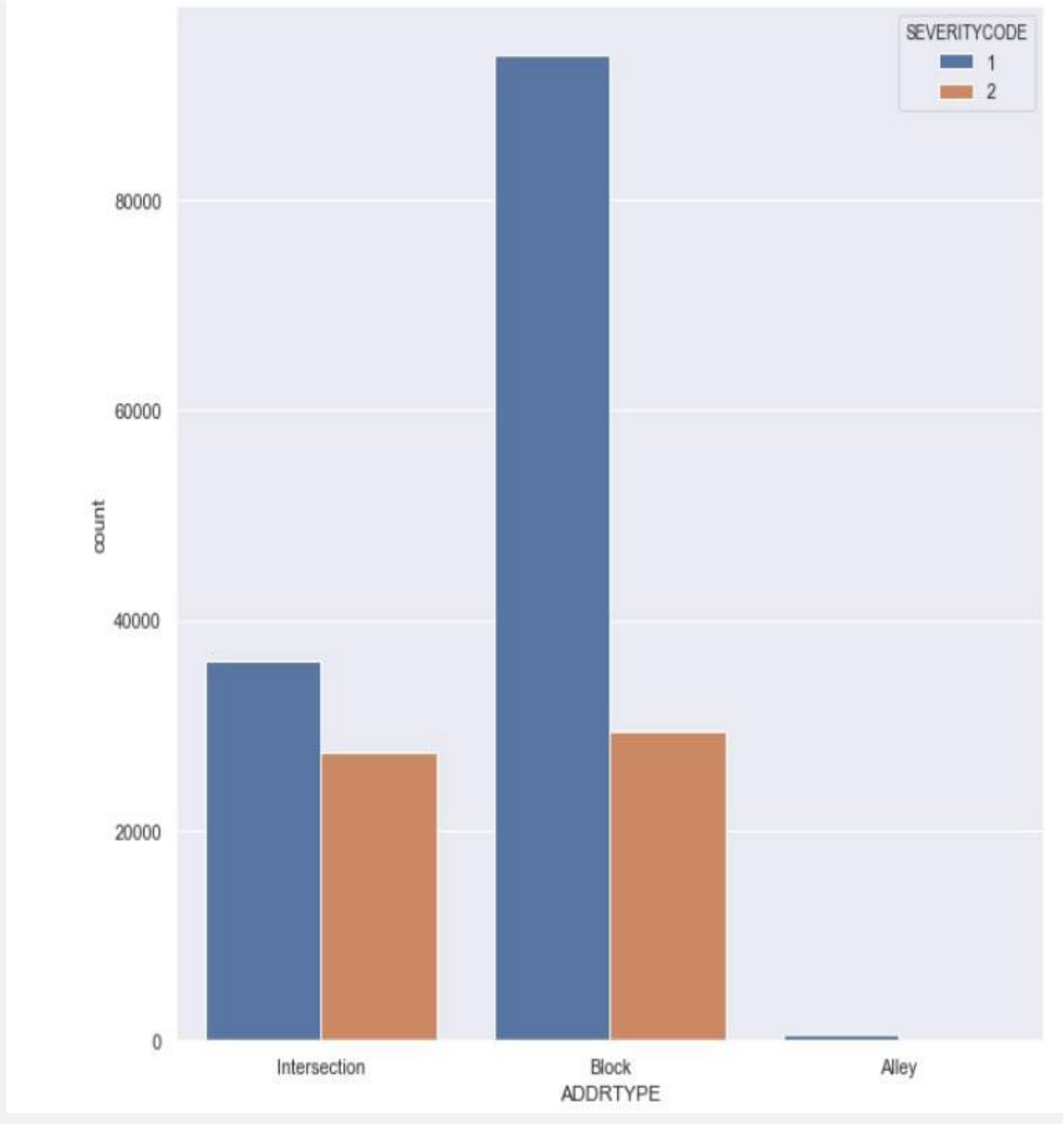
WEATHER


SEVERITYCODE
1
2
count
70000
60000
50000
40000
30000
20000
10000
0
Overcast
Raining
Clear
Unknown
Other
Snowing
Fog/Smog/Smoke
Sleet/Hail/Freezing Rain
Blowing Sand/Dirt
Severe Crosswind
Partly Cloudy
WEATHER

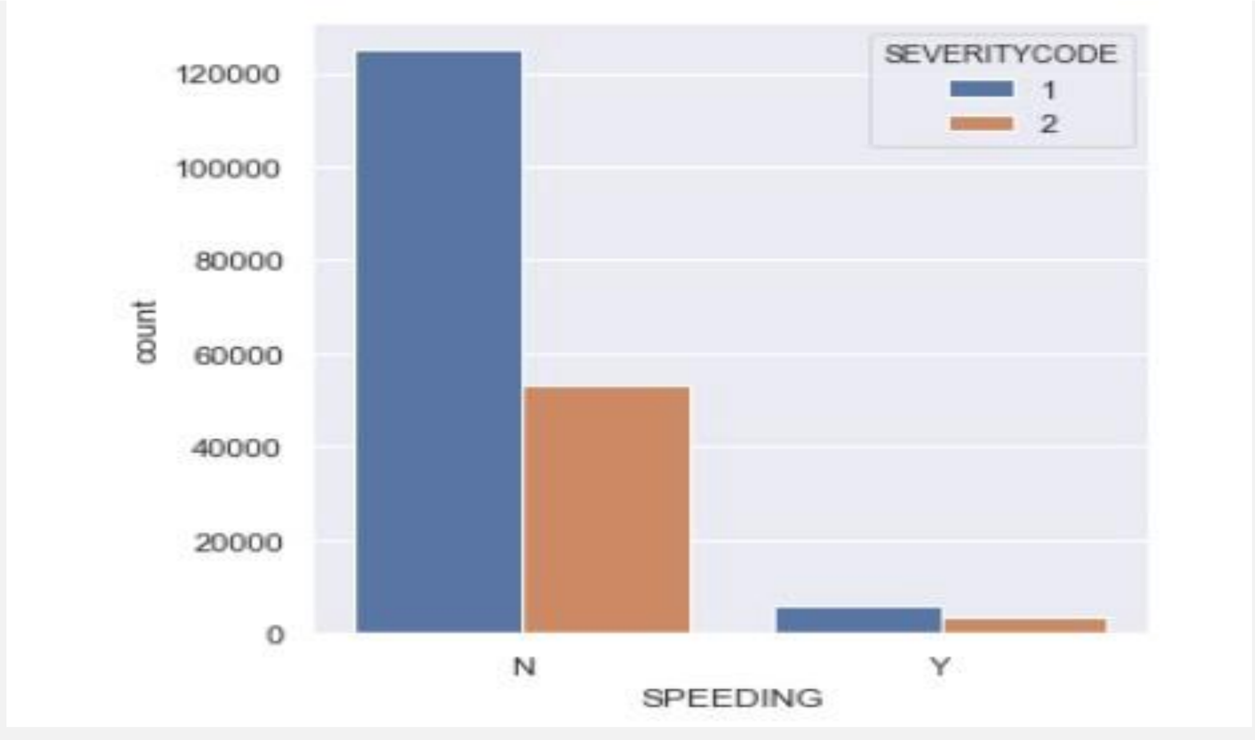
LIGHTCOND


SEVERITYCODE
1
2
80000
70000
60000
50000
40000
30000
20000
10000
0
count
Daylight
Dark - Street Lights On
Dark - No Street Lights
Unknown
Dusk
Dawn
Dark - Street Lights Off
Other
Dark - Unknown Lighting
LIGHTCOND

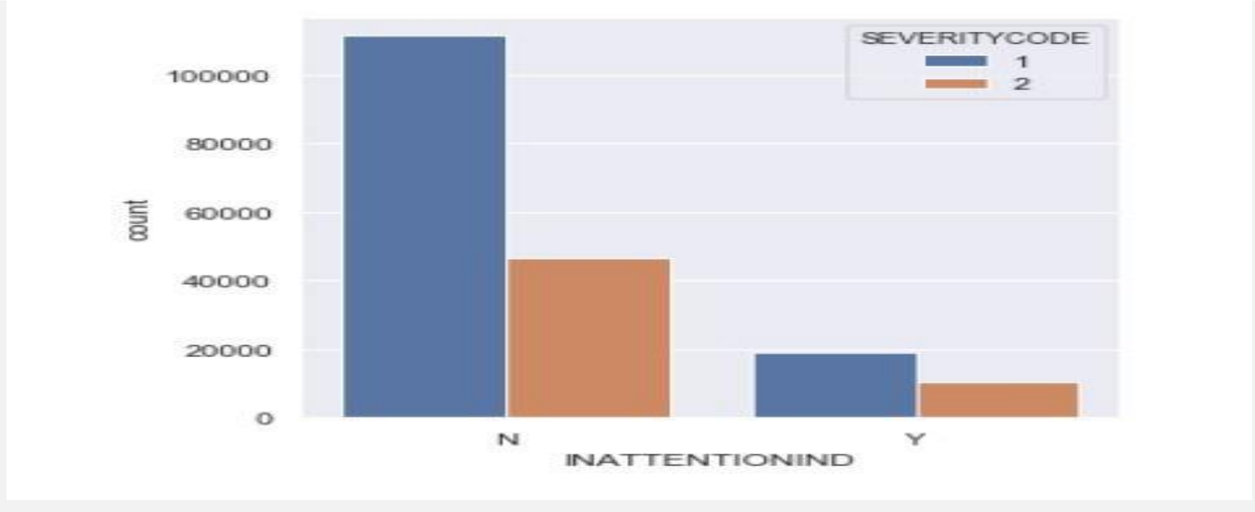
ADDRTYPE


SEVERITYCODE
1
2
80000
60000
40000
20000
0
count
Intersection
Block
Alley
ADDRTYPE

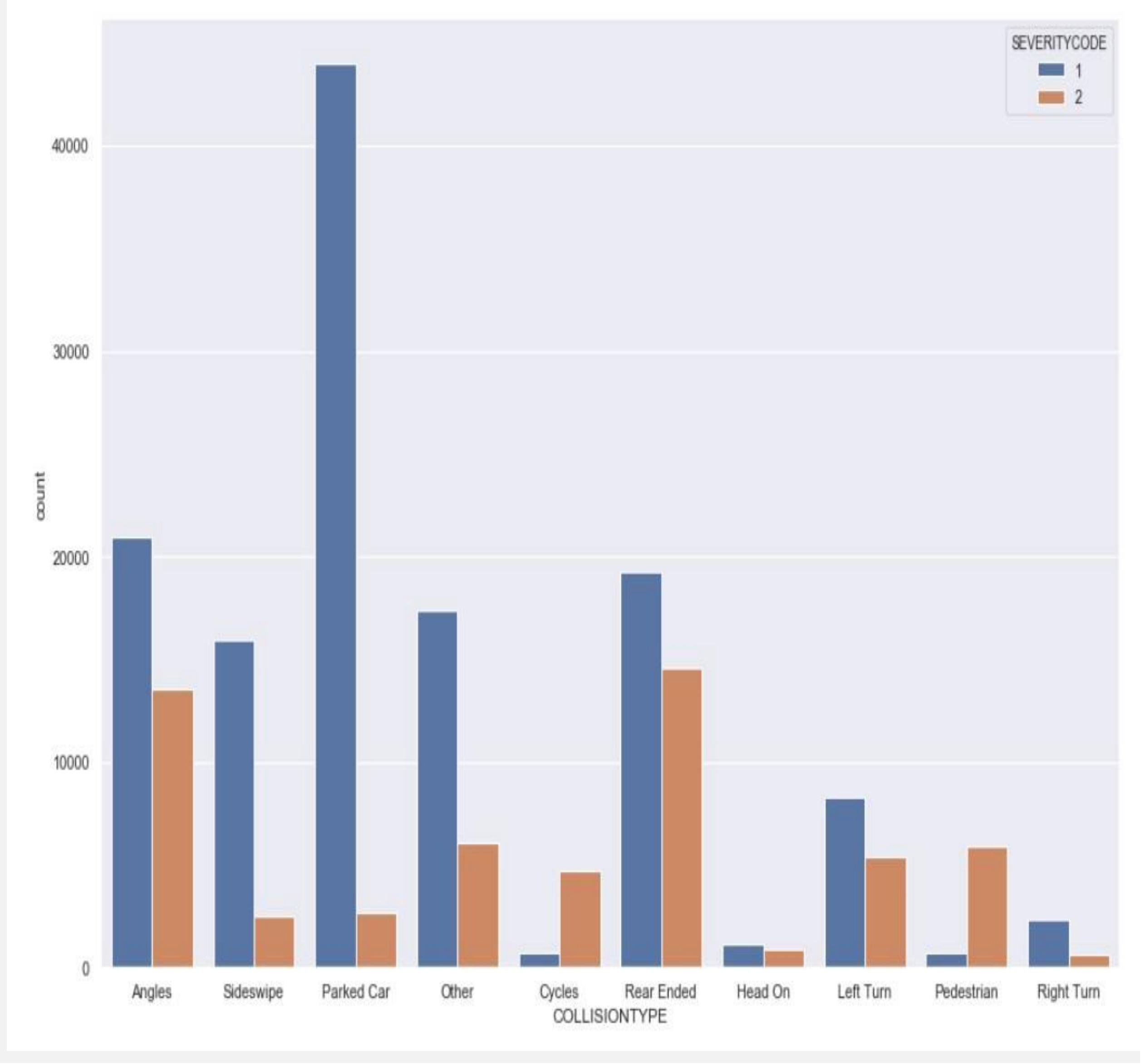
SPPEDING



INATTENTIONIND

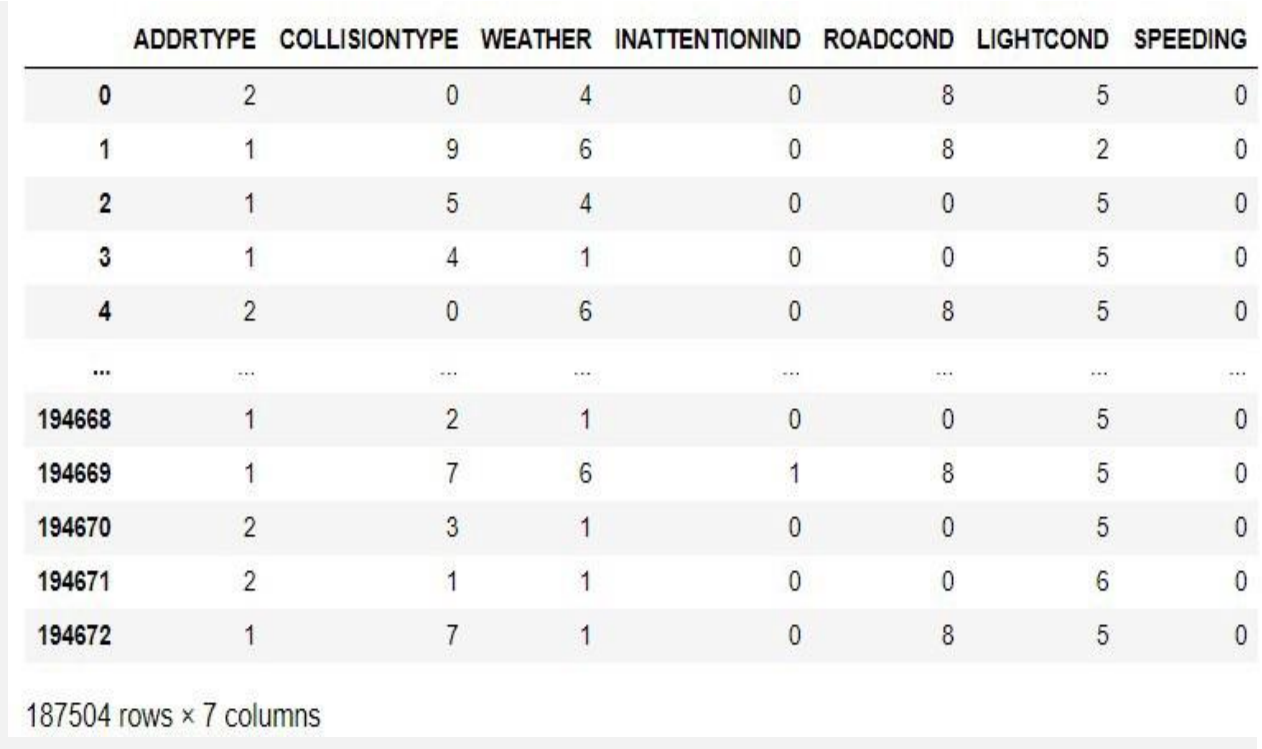


COLLISIONTYPE



For encoding I used Label Encoding on the features lsited below ‘ADDRTYPE’, ‘COLLISIONTYPE’, ‘WEATHER’, ‘INATTENTIONIND’, ‘ROADCOND’, ‘LIGHTCOND’, ‘SPEEDING’ .

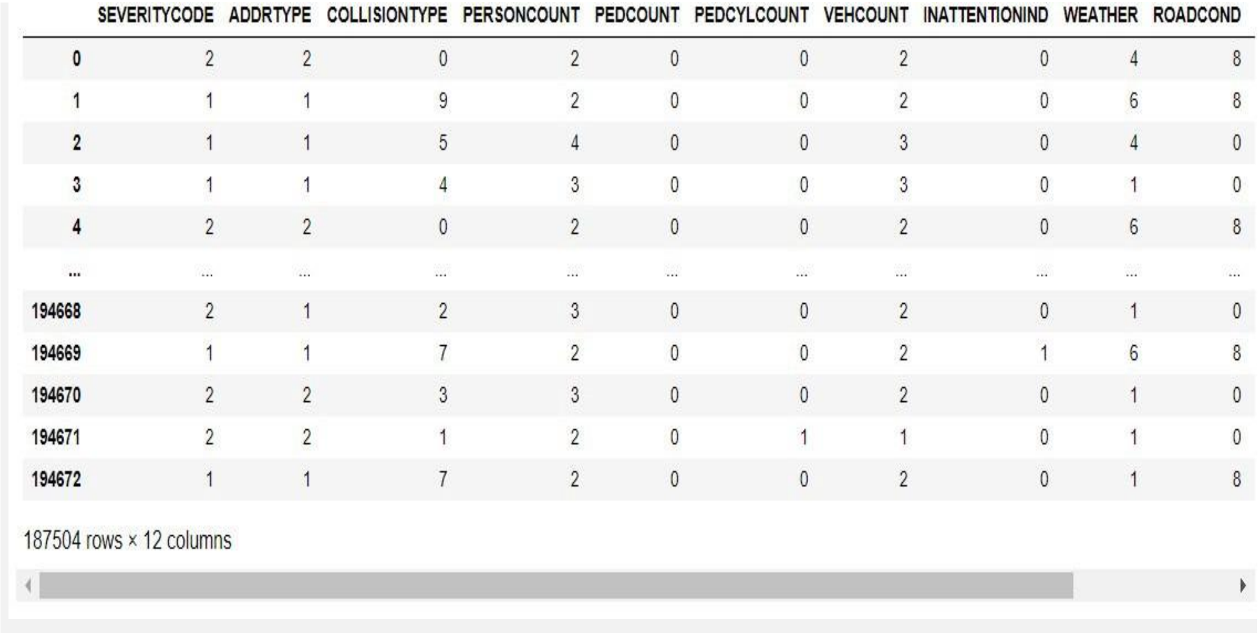
Label encoding is an approach which is used to convert the labels into numeric form so as to convert it into machine readable form.



	ADDRTYPE	COLLISIONTYPE	WEATHER	INATTENTIONIND	ROADCOND	LIGHTCOND	SPEEDING
0	2	0	4	0	8	5	0
1	1	9	6	0	8	2	0
2	1	5	4	0	0	5	0
3	1	4	1	0	0	5	0
4	2	0	6	0	8	5	0
...	...	...	...	...	...	...	...
194668	1	2	1	0	0	5	0
194669	1	7	6	1	8	5	0
194670	2	3	1	0	0	5	0
194671	2	1	1	0	0	6	0
194672	1	7	1	0	8	5	0

187504 rows × 7 columns

I replaced the old features value with the new encoded features and the data set looked like this.



	SEVERITYCODE	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND	WEATHER	ROADCOND
0	2	2	0	2	0	0	2	0	4	8
1	1	1	9	2	0	0	2	0	6	8
2	1	1	5	4	0	0	3	0	4	0
3	1	1	4	3	0	0	3	0	1	0
4	2	2	0	2	0	0	2	0	6	8
...	...	...	...	...	...	...	...	...	...	...
194668	2	1	2	3	0	0	2	0	1	0
194669	1	1	7	2	0	0	2	1	6	8
194670	2	2	3	3	0	0	2	0	1	0
194671	2	2	1	2	0	1	1	0	1	0
194672	1	1	7	2	0	0	2	0	1	8

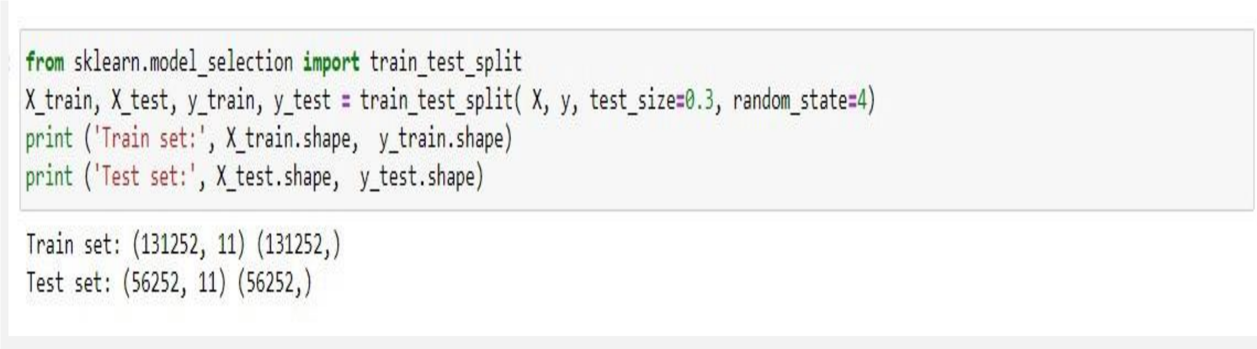
187504 rows × 12 columns

Now for modelling my dataset was ready.

Train Test Split

It is important that our models have a high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. So how can we improve out-of-sample accuracy? One way is to use an evaluation approach called Train/Test Split. Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, we will train with the training set and test with the testing set.

After normalizing dataset ,I trained the dataset, since there were sufficient rows and so that my model do not produce bias result I kept the test size at 30%.



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (131252, 11) (131252,)
Test set: (56252, 11) (56252,)
```

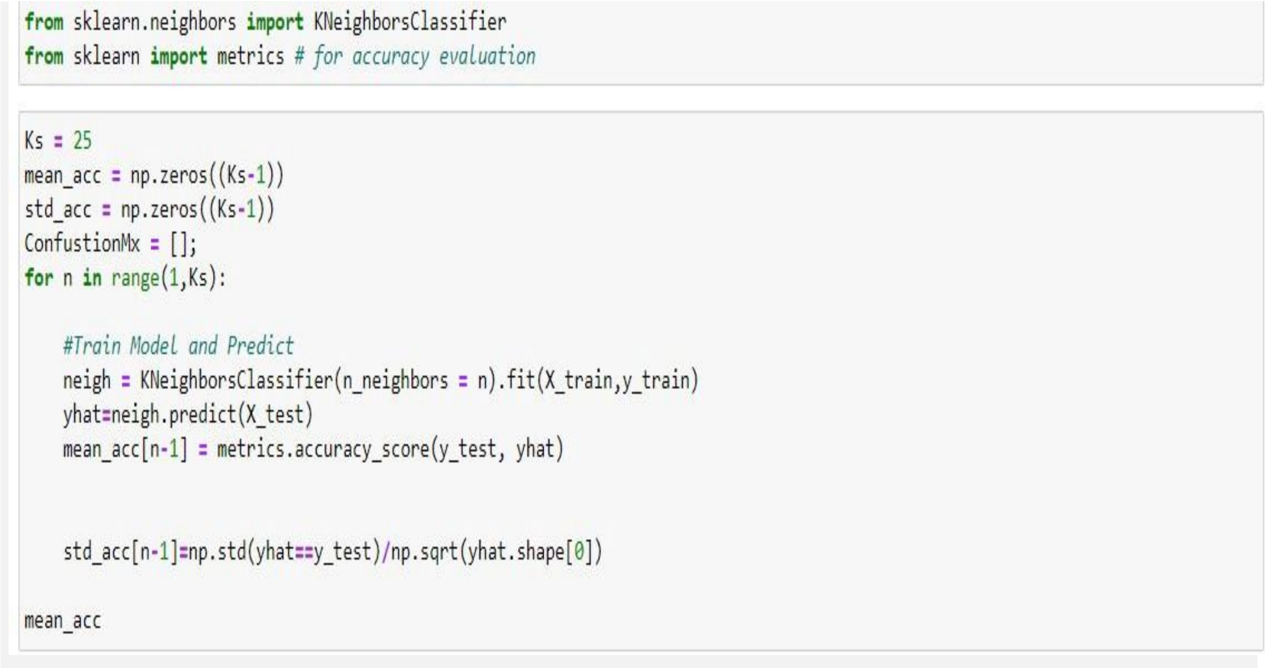
Modelling

For predictive modelling I used different types of classification models.

Let’s start with the models.

KNN

K-Nearest Neighbors is an algorithm for supervised learning. Where the data is ‘trained’ with data points corresponding to their classification. Once a point is to be predicted, it takes into account the ‘K’ nearest points to it to determine it’s classification.



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics # for accuracy evaluation
```

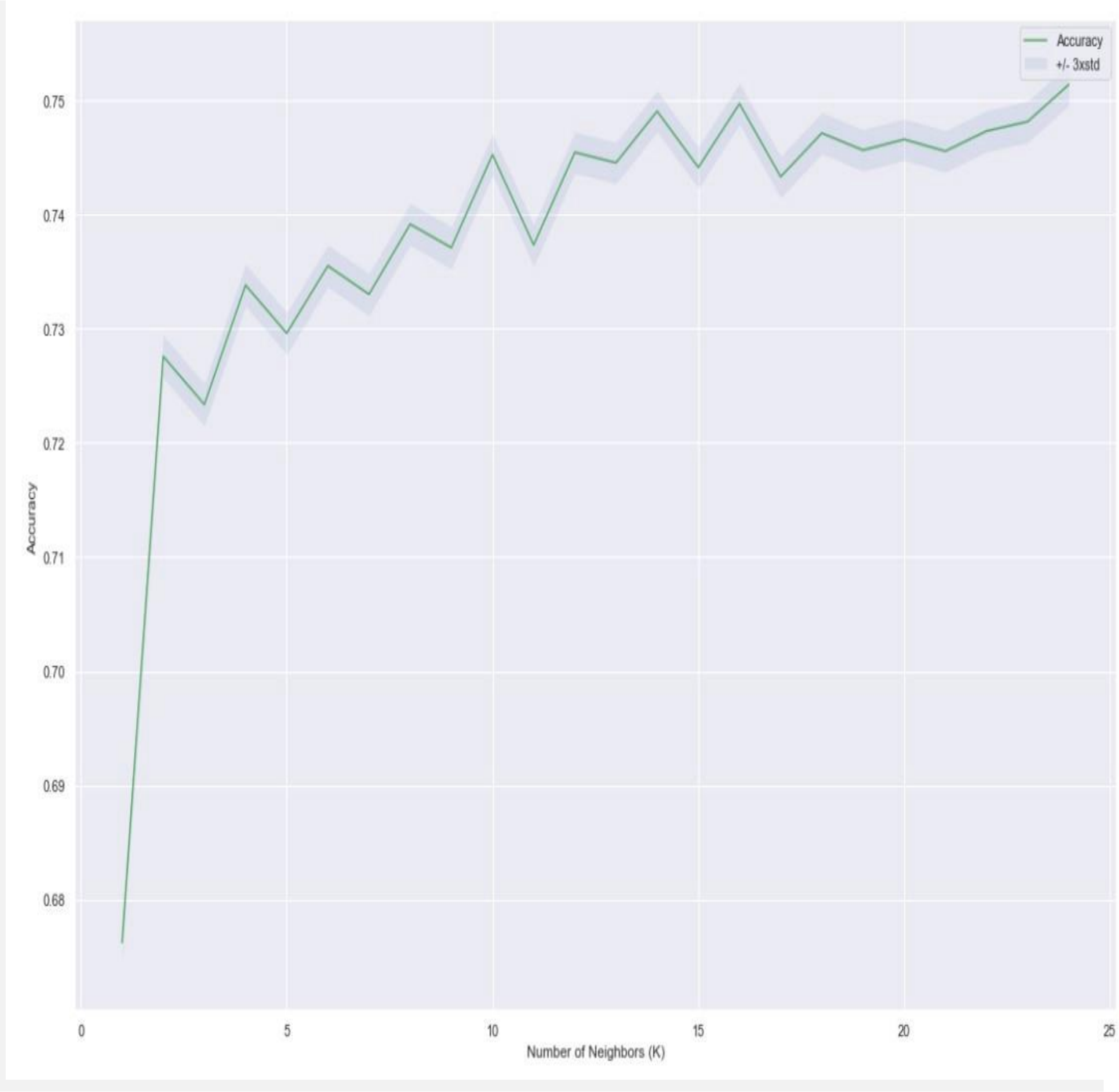
```
Ks = 25
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

For this model I have taken k as 24, I used for loop to find the highest accuracy between the values 1 to 25.

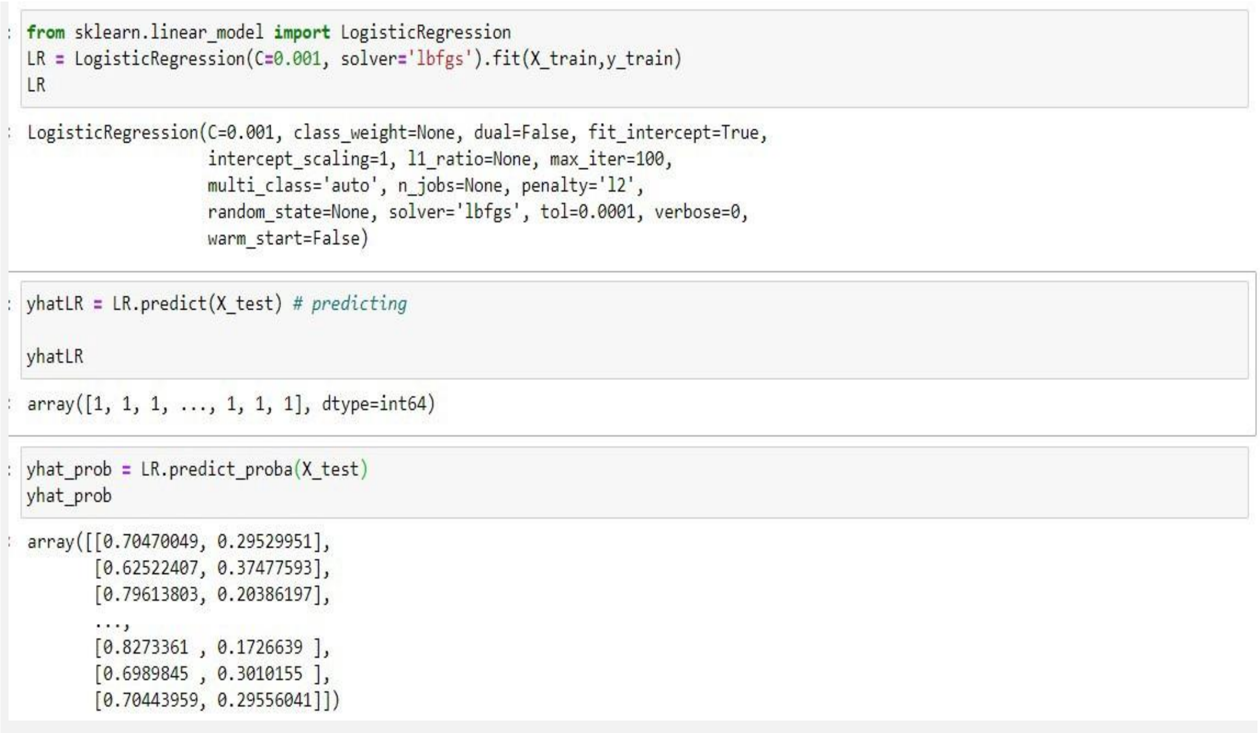


For predictions I have used f1_score, jaccard_score and mean_accuracy_score.

LOGISTIC REGRESSION

Logistic Regression is a variation of Linear Regression, useful when the observed dependent variable, **y**, is categorical. It produces a formula that predicts the probability of the class label as a function of the independent variables.

In addition to f1_score, mean_accuracy_score and jaccard_score , log_loss is also used in this.



```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.001, solver='lbfgs').fit(X_train,y_train)
LR
```

```
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
yhatLR = LR.predict(X_test) # predicting

yhatLR
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

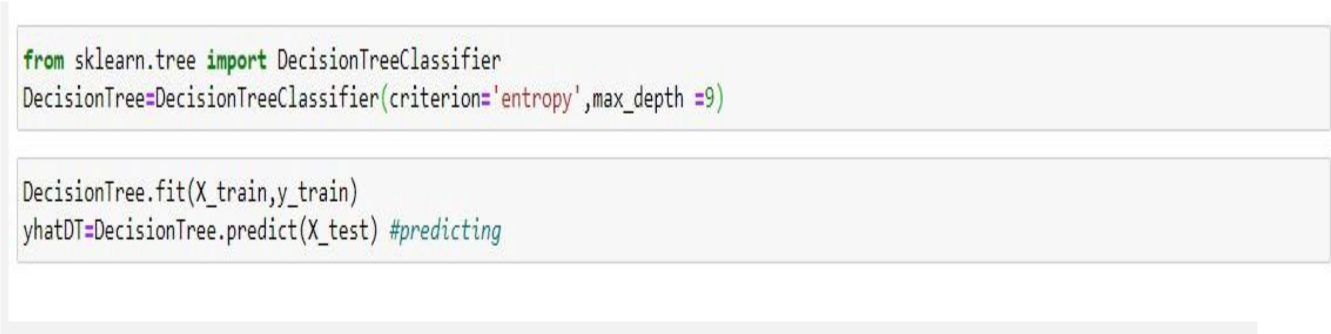
```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

```
array([[0.70470049, 0.29529951],
       [0.62522407, 0.37477593],
       [0.79613803, 0.20386197],
       ...,
       [0.8273361 , 0.1726639 ],
       [0.6989845 , 0.3010155 ],
       [0.70443959, 0.29556041]])
```

DECISION TREE

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter.

For predictions I have used f1_score, jaccard_score and mean_accuracy_score.



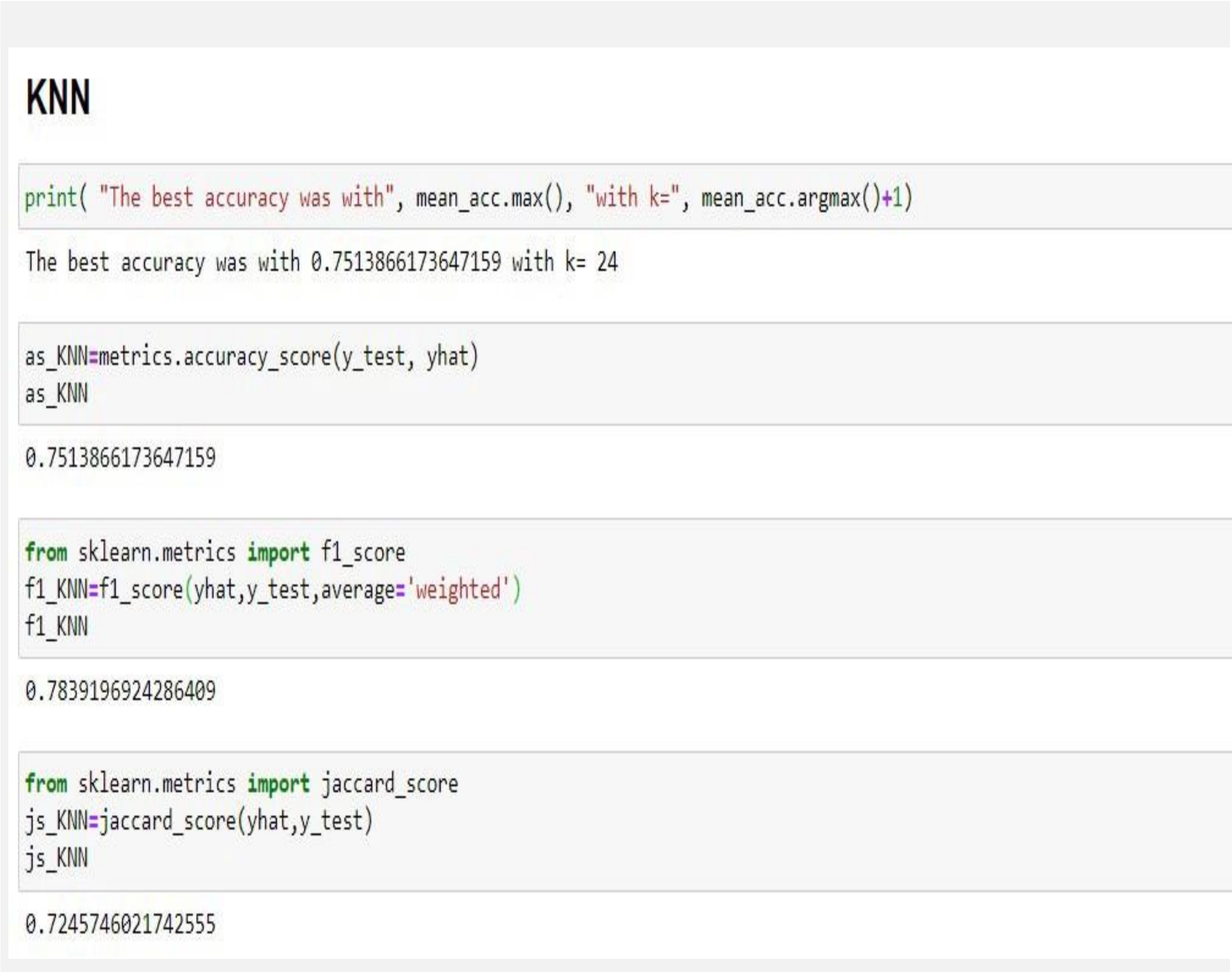
```
from sklearn.tree import DecisionTreeClassifier
DecisionTree=DecisionTreeClassifier(criterion='entropy',max_depth =9)
```

```
DecisionTree.fit(X_train,y_train)
yhatDT=DecisionTree.predict(X_test) #predicting
```

Evaluation

All the above models were evaluated through the different types of metrics score.

KNN



KNN

```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.7513866173647159 with k= 24
```

```
as_KNN=metrics.accuracy_score(y_test, yhat)
as_KNN
```

```
0.7513866173647159
```

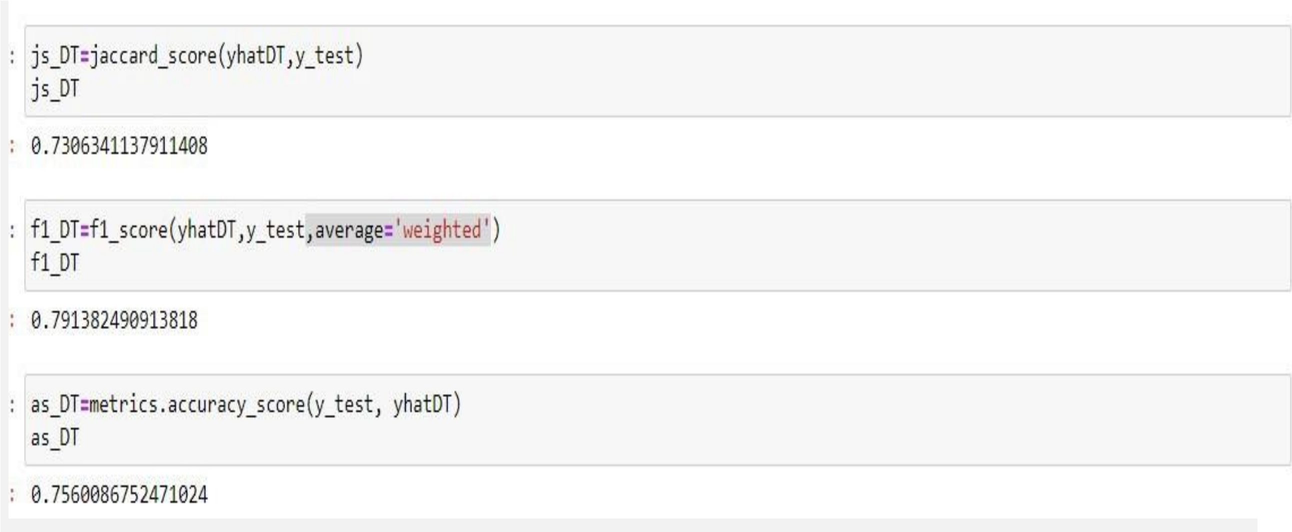
```
from sklearn.metrics import f1_score
f1_KNN=f1_score(yhat,y_test,average='weighted')
f1_KNN
```

```
0.7839196924286409
```

```
from sklearn.metrics import jaccard_score
js_KNN=jaccard_score(yhat,y_test)
js_KNN
```

```
0.7245746021742555
```

DECISION TREE



```
js_DT=jaccard_score(yhatDT,y_test)
js_DT
```

```
0.7306341137911408
```

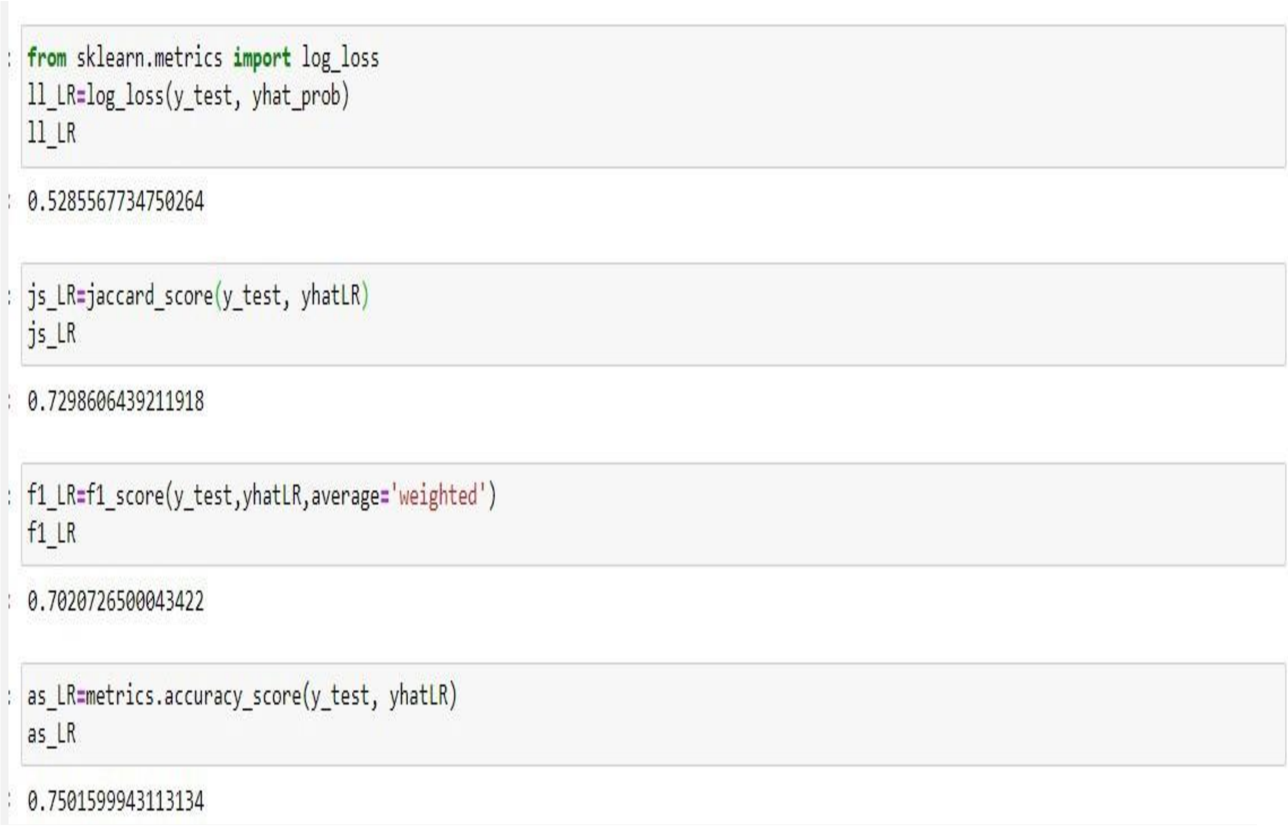
```
f1_DT=f1_score(yhatDT,y_test,average='weighted')
f1_DT
```

```
0.791382490913818
```

```
as_DT=metrics.accuracy_score(y_test, yhatDT)
as_DT
```

```
0.7560086752471024
```

LOGISTIC REGRESSION



```
from sklearn.metrics import log_loss
ll_LR=log_loss(y_test, yhat_prob)
ll_LR
```

```
0.5285567734750264
```

```
js_LR=jaccard_score(y_test, yhatLR)
js_LR
```

```
0.7298606439211918
```

```
f1_LR=f1_score(y_test,yhatLR,average='weighted')
f1_LR
```

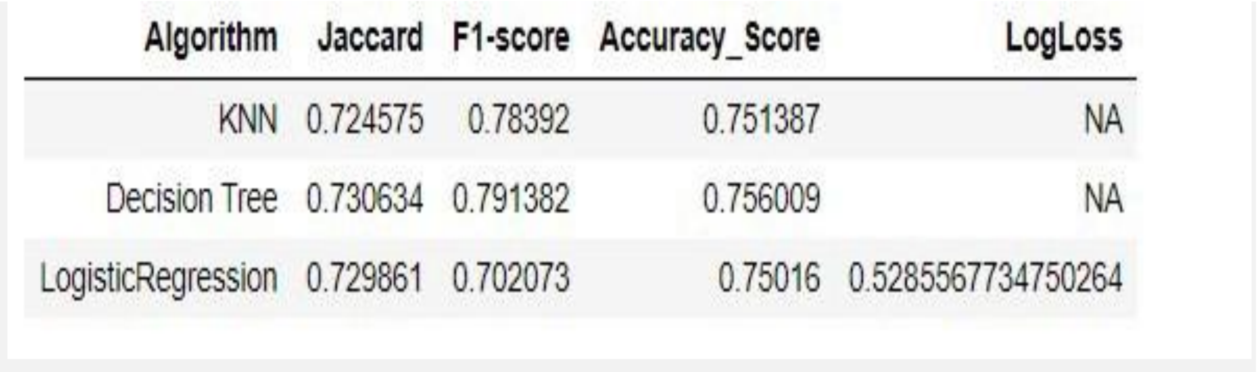
```
0.7020726500043422
```

```
as_LR=metrics.accuracy_score(y_test, yhatLR)
as_LR
```

```
0.7501599943113134
```

Results

Now, let’s see the score of all the models.



Algorithm	Jaccard	F1-score	Accuracy_Score	LogLoss
KNN	0.724575	0.78392	0.751387	NA
Decision Tree	0.730634	0.791382	0.756009	NA
LogisticRegression	0.729861	0.702073	0.75016	0.5285567734750264

Discussion

As I mentioned earlier lots of people are loosing their lives everyday and apart from this a huge amount of money is also spent on this.

At starting I performed data cleaning which is the most important and crucial section of any projects like this.

I used different types of classification models to predict.

On these models, I used different types of prediction scores which clears how much our models are effective. I ended up with showing the scores in an organized way.

Conclusion

As a result, the numbers of cars are increasing day-by-day. Most of the people prefer to use their own vehicle for their own comfort.

Government or Multinational service providers who are active in this field or wish to join this field, this type of data analysis can be very useful to them.

You can find the source code here:

https://github.com/sampurnalal/Coursera_Capstone/blob/main/Capstone_FINAL.ipynb

Thank you for reading, I hope you found it useful.