Predict the Severity of a Traffic Accident

IBM Data Science Professional Certificate

Monica

LinkedIn | GitHub

9 October 2020

Data Source

- Part of an example dataset in IBM Data Science Professional Certificate
- Can be downloaded from my GitHub repository https://github.com/monica110394/Coursera_Capstone

Variable Selection

- `'SEVERITYCODE', 'ADDRTYPE', 'COLLISIONTYPE', 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'INATTENTIONIND', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND', 'SPEEDING', 'ST_COLCODE', 'HITPARKEDCAR'`
- Target variable: `'SEVERITYCODE'`

Feature Engineering

- Null Values are detected
- Attributes are removed that have most of the values as null
- Records are dropped that had missing values
  - `Number of rows: 182895`
  - `Number of Columns: 15`
  - `'SEVERITYCODE', 'ADDRTYPE', 'COLLISIONTYPE', 'PERSONCOUNT','PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND', 'ST_COLCODE', 'HITPARKEDCAR'`

Label Encoding (categorical variables)

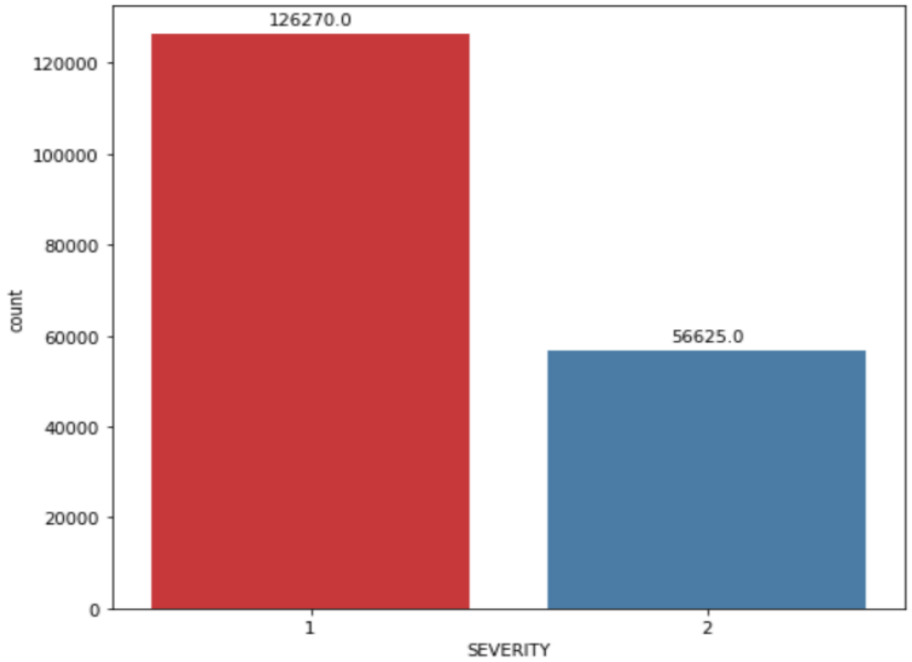
- 'ADDRTYPE'
- 'COLLISIONTYPE'
- 'JUNCTIONTYPE'
- 'SDOT_COLCODE'
- 'UNDERINFL'
- 'WEATHER'
- 'ROADCOND'
- 'LIGHTCOND'
- 'ST_COLCODE'
- 'HITPARKEDCAR'

Numerical variables

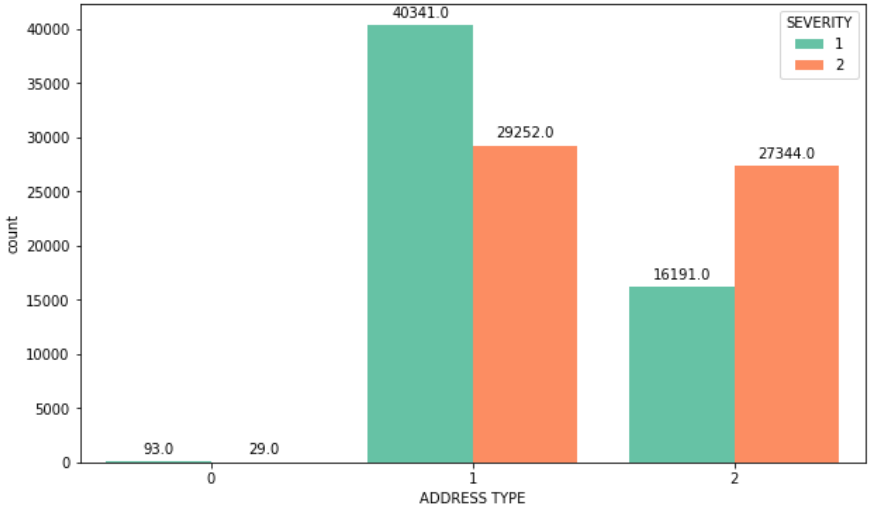
- 'PERSONCOUNT'
- 'PEDCOUNT'
- 'PEDCYLCOUNT'
- 'VEHCOUNT'

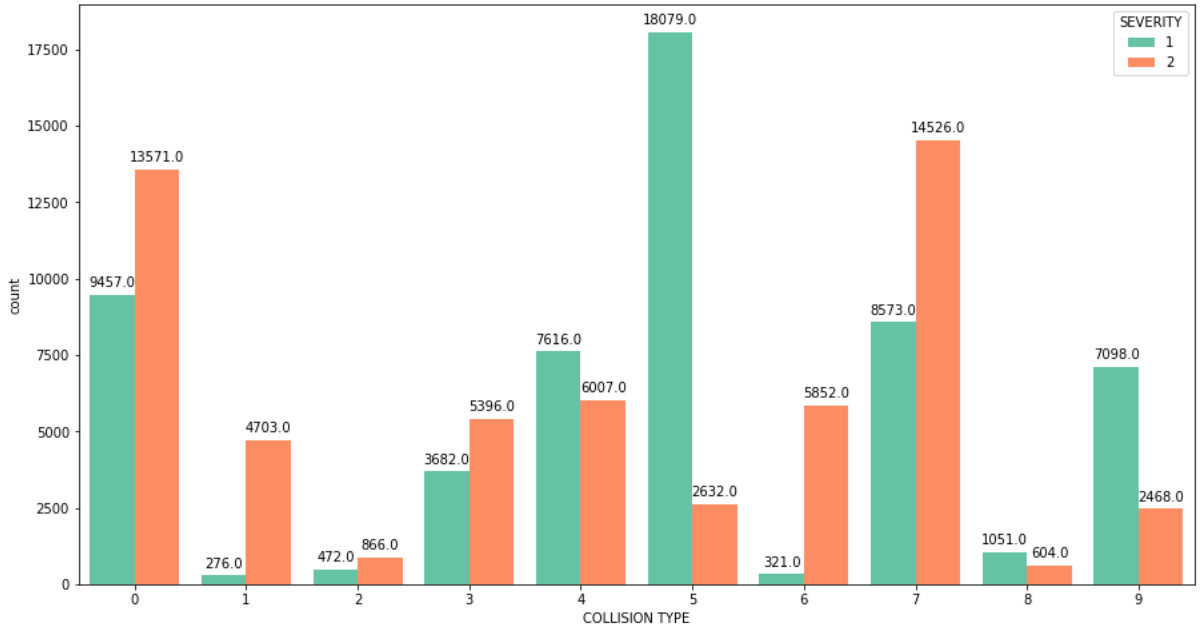
Balancing dataset:

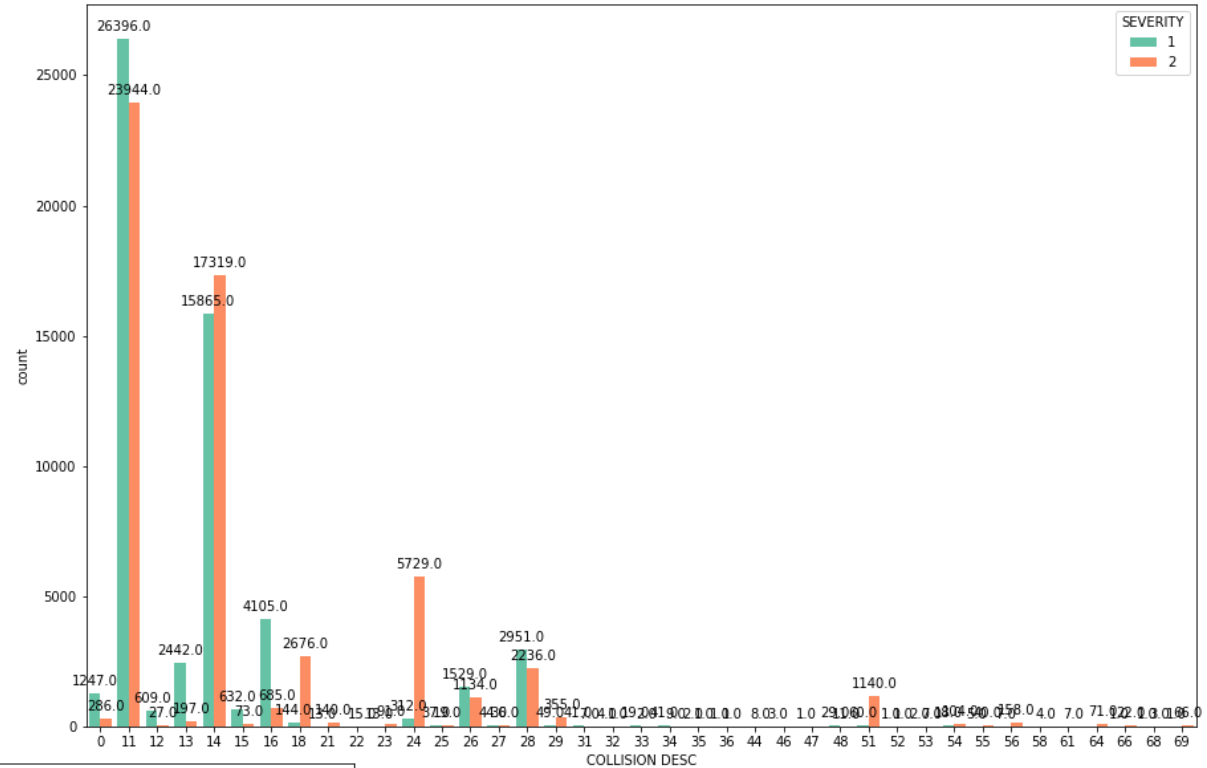
- Unbalanced dataset can cause the prediction to be skewed
- Dataset is balanced by down sampling the category that has greater number of samples
- In this case it is 'SEVERITYCODE'=1

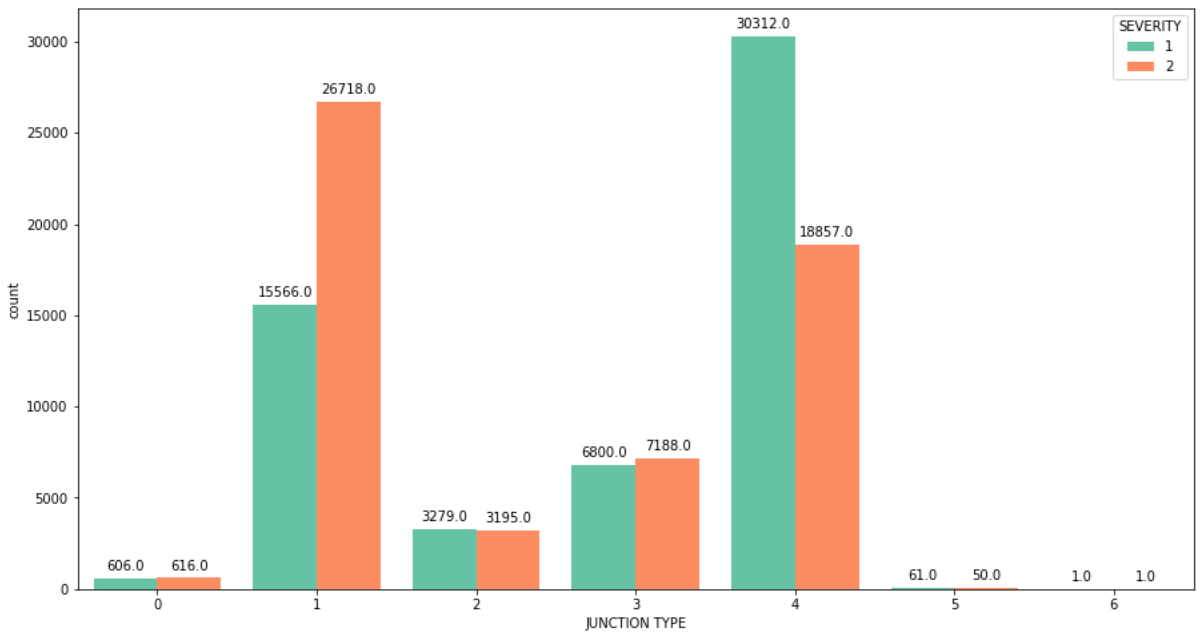


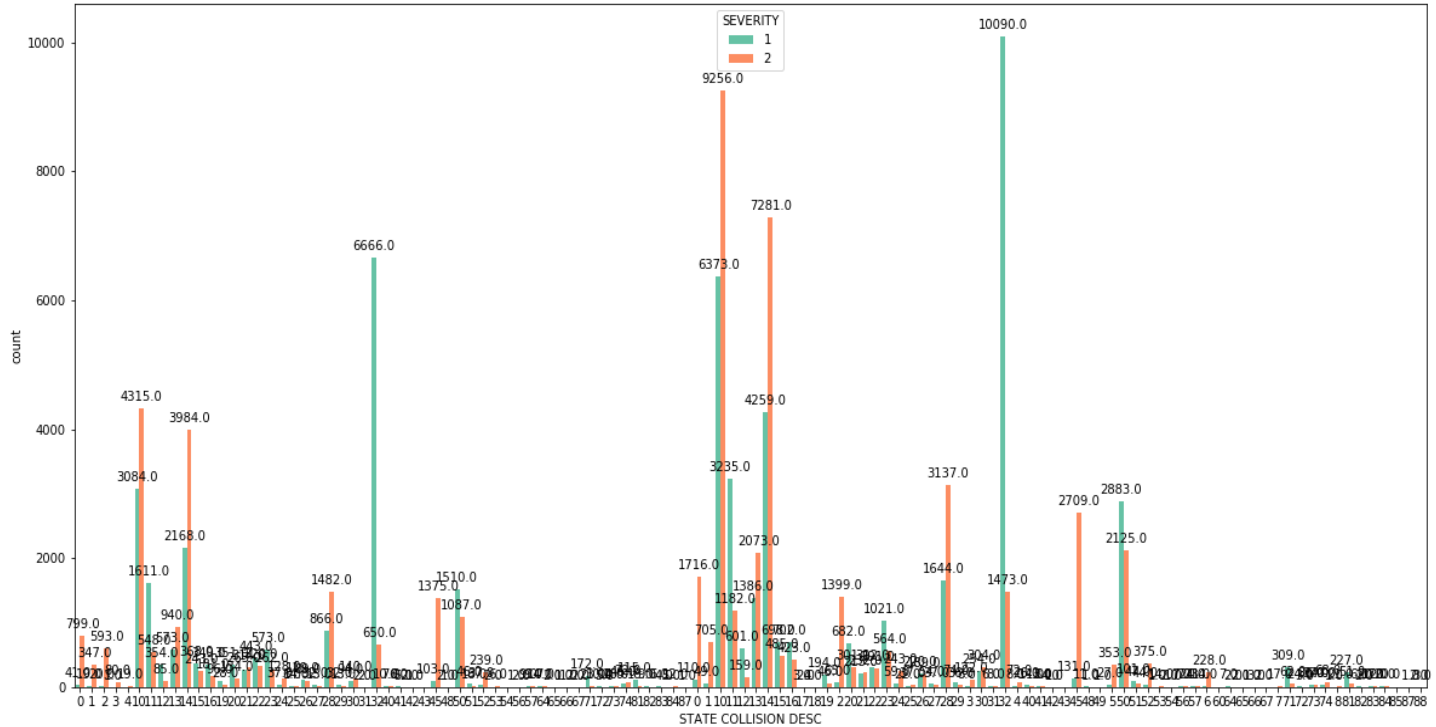
Exploratory Data Analysis (categorical variables)

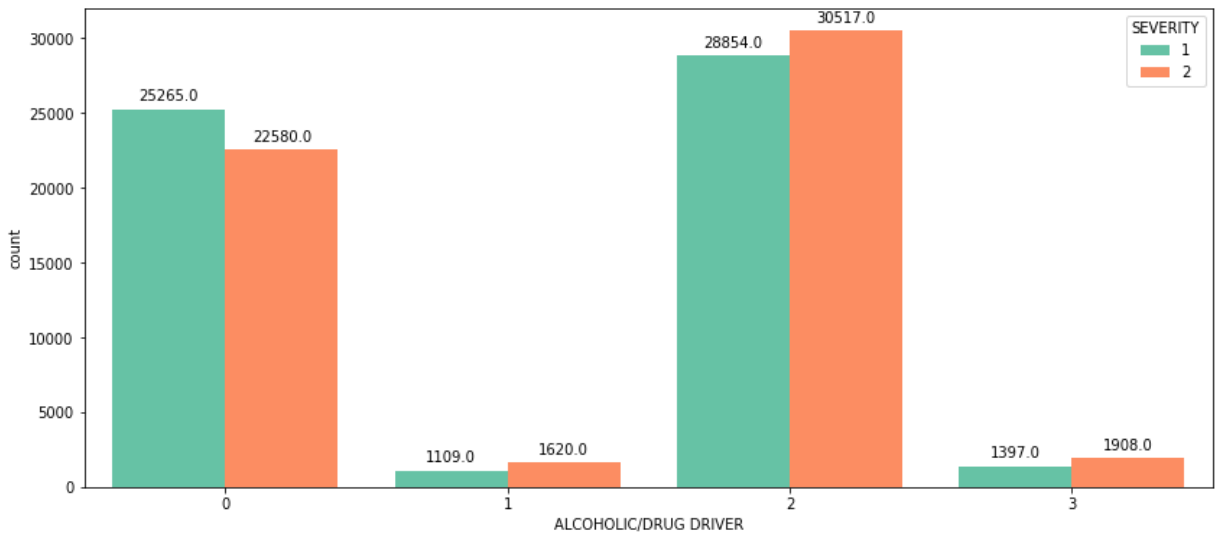


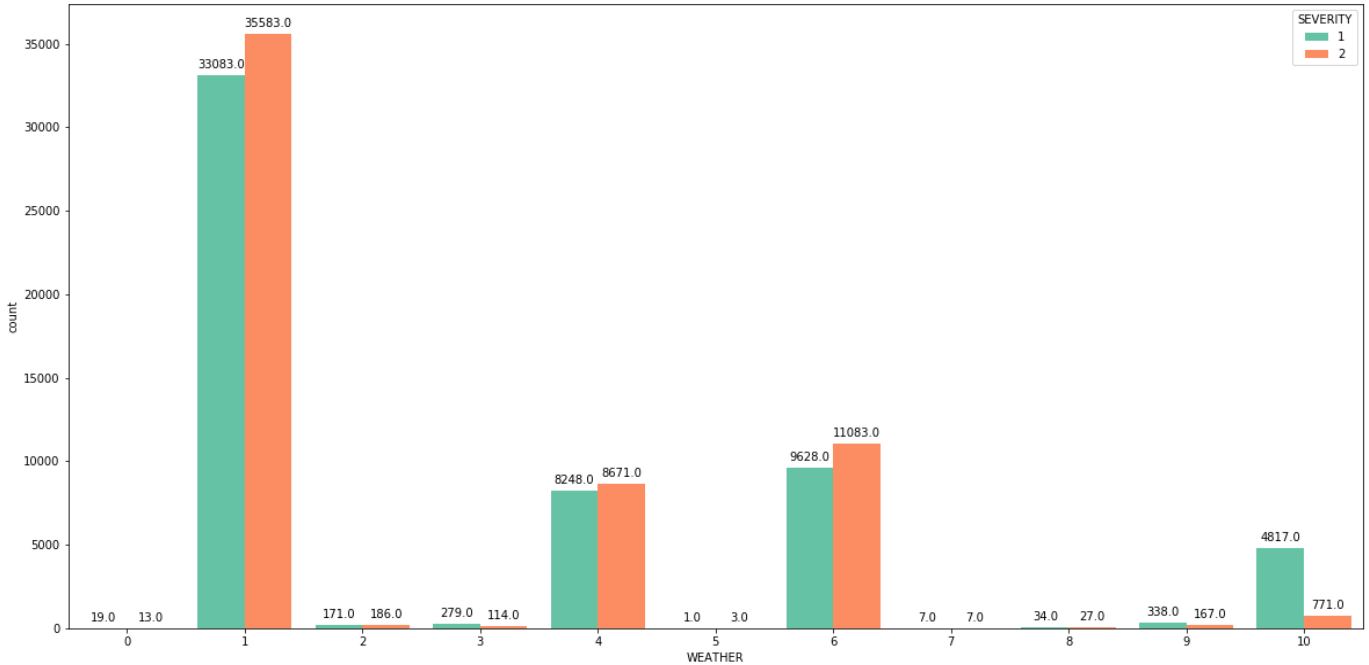


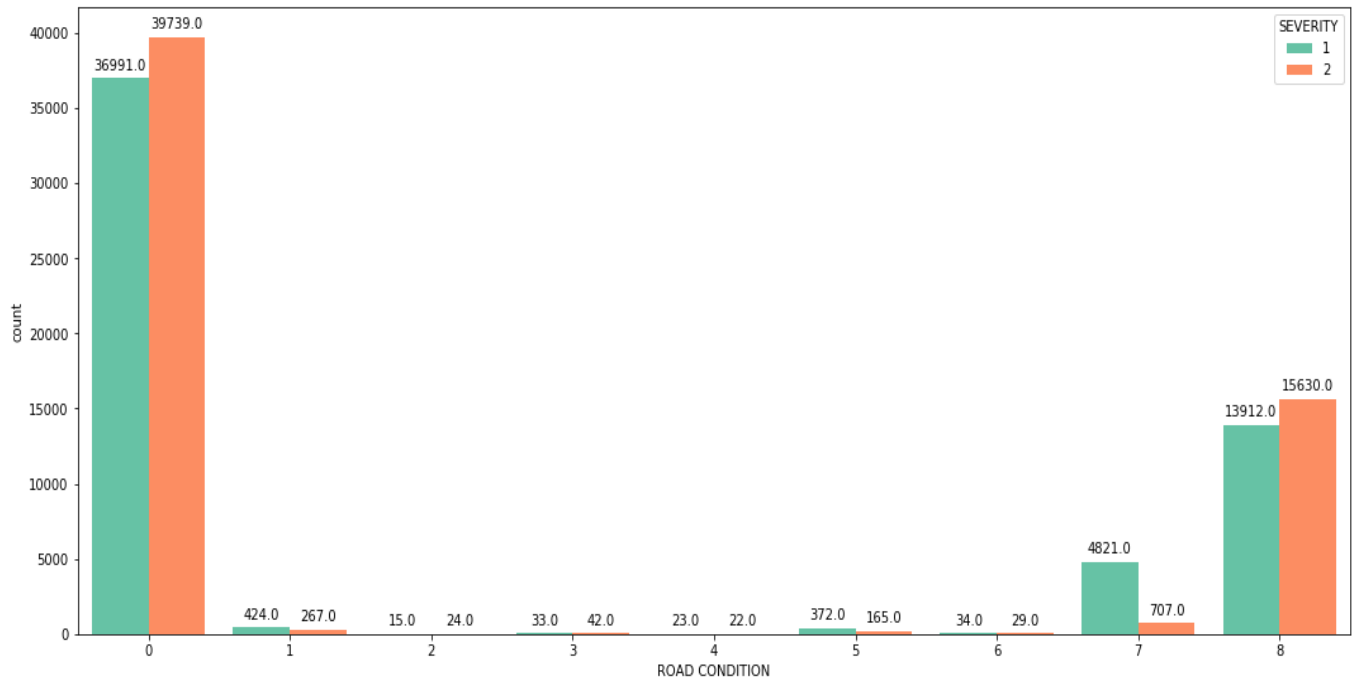

SEVERITY
1
2
count
COLLISION DESC
26396.0
23944.0
17319.0
15865.0
5729.0
4105.0
2951.0
2676.0
2442.0
2236.0
1529.0
1247.0
1140.0

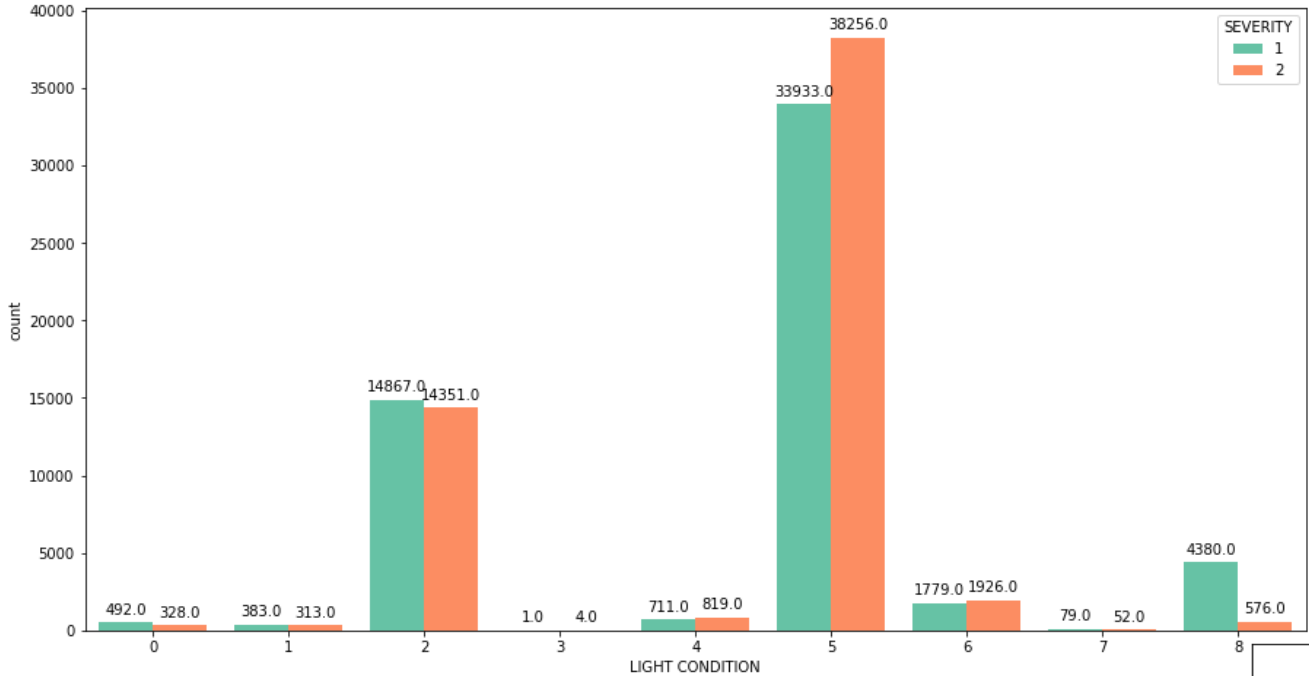

SEVERITY
1
2
count
JUNCTION TYPE
30312.0
26718.0
18857.0
15566.0
7188.0
6800.0
3279.0
3195.0
606.0
616.0
61.0
50.0
1.0
1.0

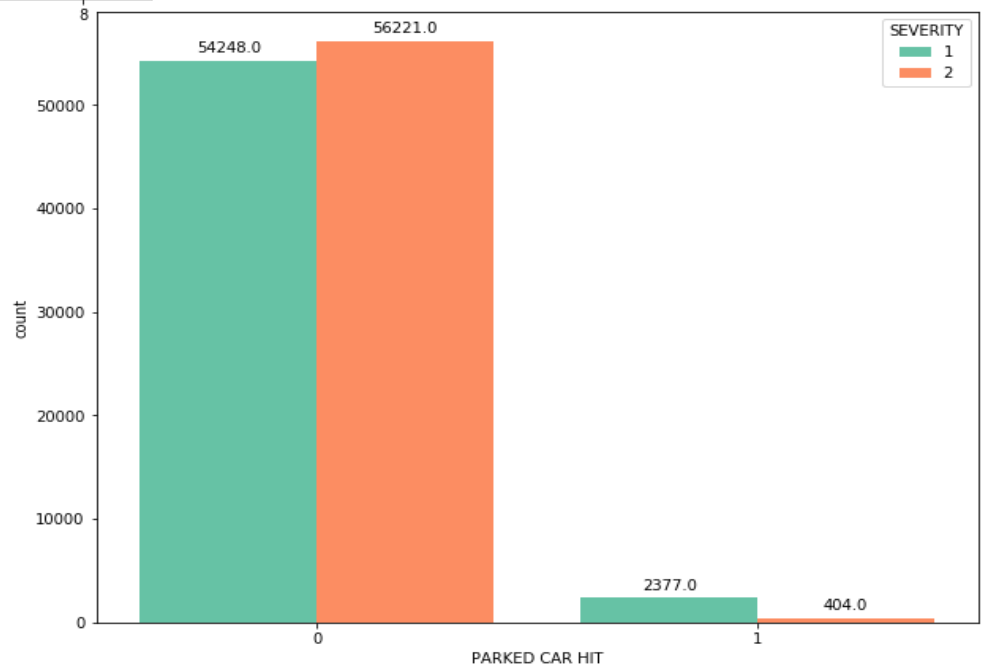

SEVERITY
1
2
count
STATE COLLISION DESC


SEVERITY
1
2
count
ALCOHOLIC/DRUG DRIVER
25265.0
22580.0
1109.0
1620.0
28854.0
30517.0
1397.0
1908.0

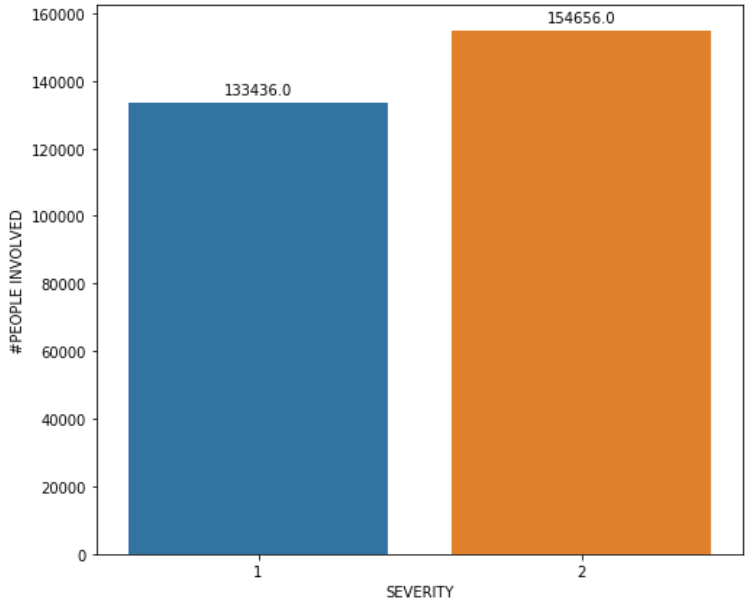

SEVERITY
1
2
count
WEATHER
0
1
2
3
4
5
6
7
8
9
10
19.0
13.0
33083.0
35583.0
171.0
186.0
279.0
114.0
8248.0
8671.0
1.0
3.0
9628.0
11083.0
7.0
7.0
34.0
27.0
338.0
167.0
4817.0
771.0

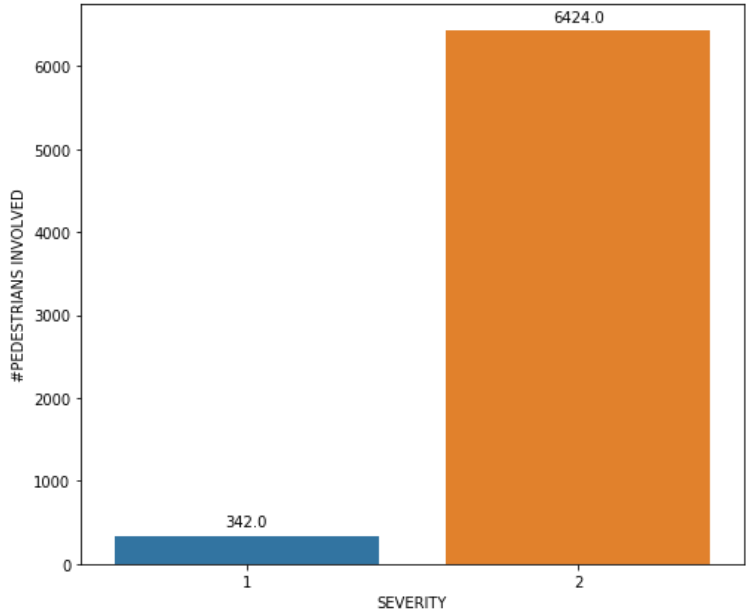

SEVERITY
1
2
count
ROAD CONDITION
0
1
2
3
4
5
6
7
8
36991.0
39739.0
424.0
267.0
15.0
24.0
33.0
42.0
23.0
22.0
372.0
165.0
34.0
29.0
4821.0
707.0
13912.0
15630.0

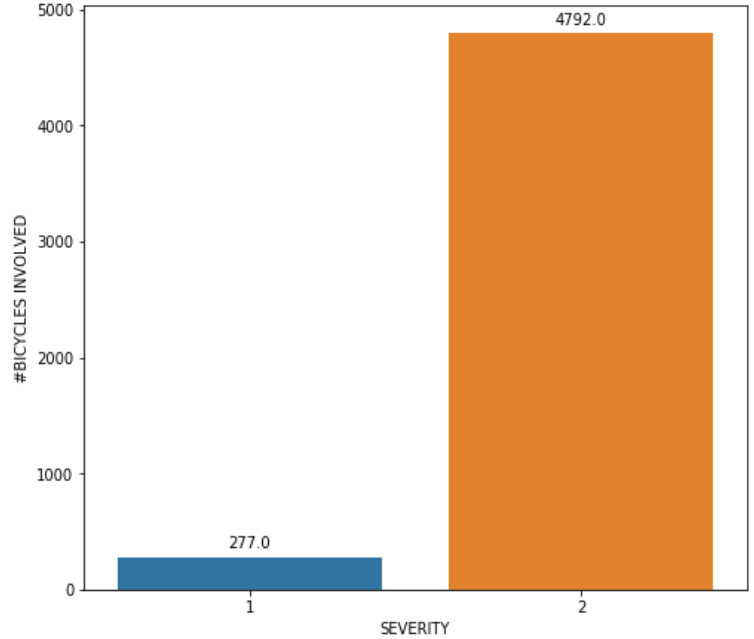

SEVERITY
1
2
count
LIGHT CONDITION
492.0
328.0
383.0
313.0
14867.0
14351.0
1.0
4.0
711.0
819.0
33933.0
38256.0
1779.0
1926.0
79.0
52.0
4380.0
576.0

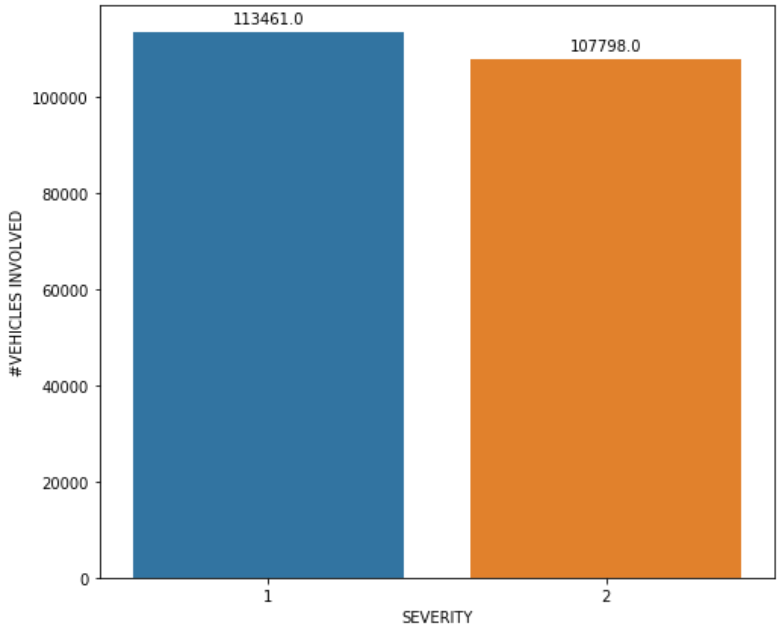

SEVERITY
1
2
count
PARKED CAR HIT
54248.0
56221.0
2377.0
404.0

Exploratory Data Analysis (numerical variables)









Data Preparation

- Normalize the feature set

```
from sklearn import preprocessing

X = preprocessing.StandardScaler().fit(features).transform(features)
```

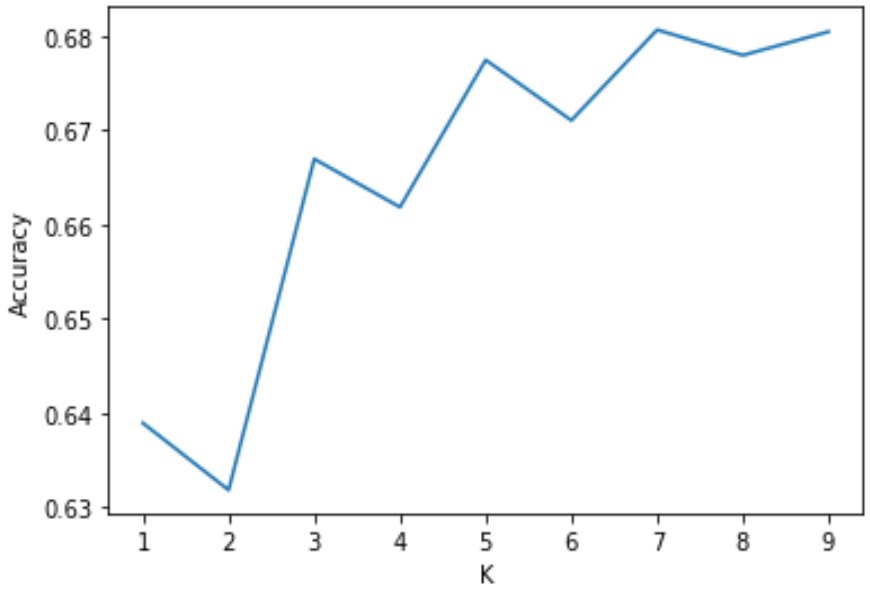
- Dataset split into train and test sets

```
Train set: (90600, 14) (90600,)
Test set: (22650, 14) (22650,)
```

K Nearest Neighbours

The best accuracy score achieved was 0.6806 with k=7

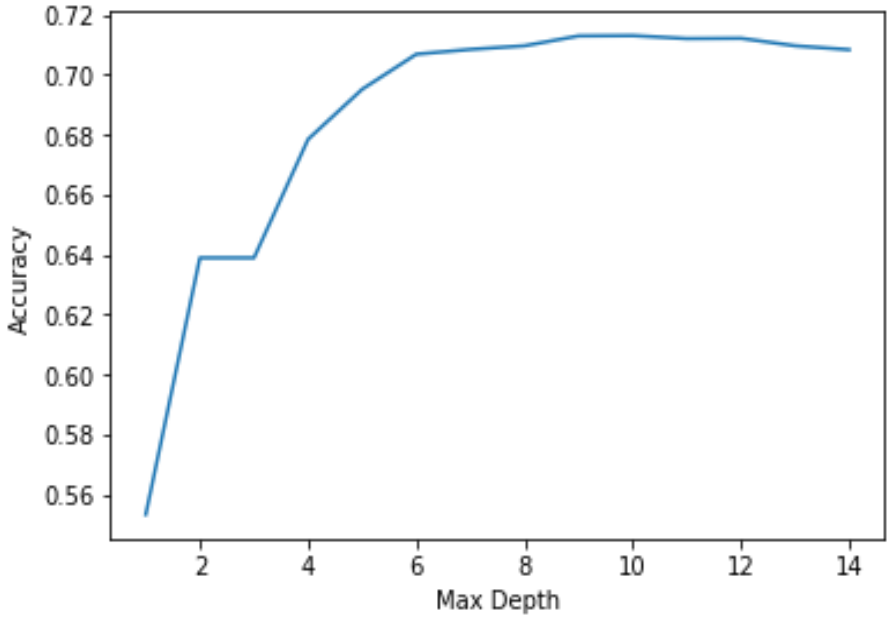
```
KNeighborsClassifier(algorithm='auto',
leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None,
n_neighbors=7, p=2,
weights='uniform')

Train set Accuracy:  0.7195
Test set Accuracy:  0.6806
```



Decision Tree

The best accuracy score achieved was 0.6806 with max_depth = 10

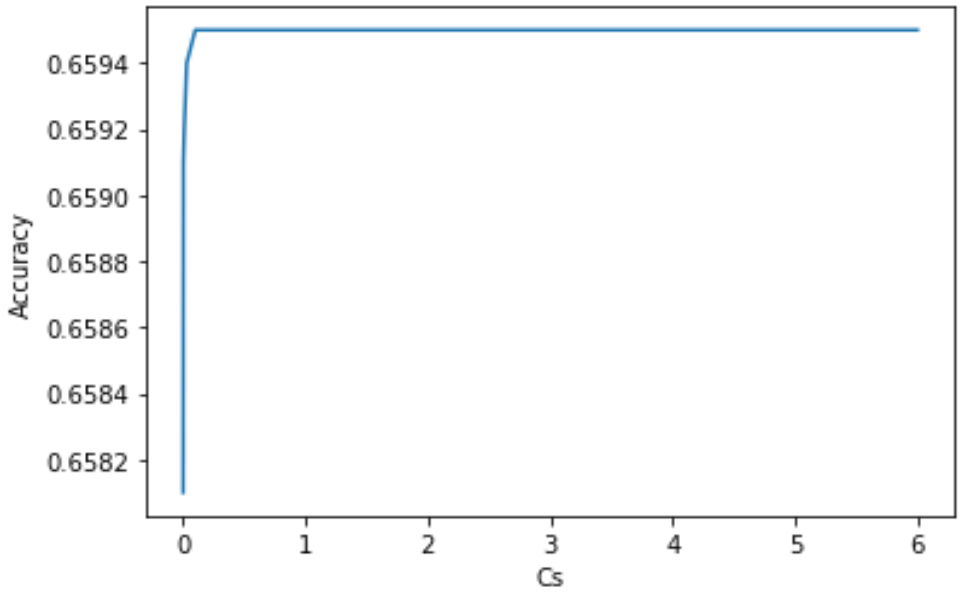
```
DecisionTreeClassifier(ccp_alpha=0.0,
class_weight=None, criterion='entropy',
max_depth=10, max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort='deprecated',
random_state=None, splitter='best')

Train set Accuracy:  0.7183
Test set Accuracy:  0.7128
```



Logistic Regression

The best accuracy score achieved was 0.6806 with C=0.1

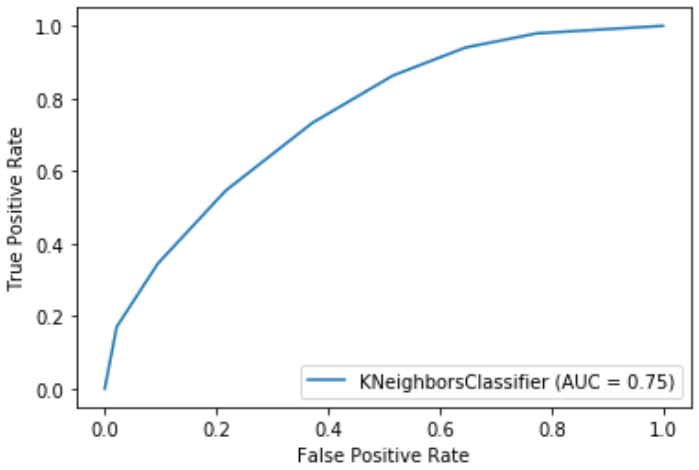
```
LogisticRegression(C=0.1, class_weight=None,
dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None,
max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='liblinear',
tol=0.0001, verbose=0,
warm_start=False)

Train set Accuracy:  0.6591
Test set Accuracy:  0.6595
```

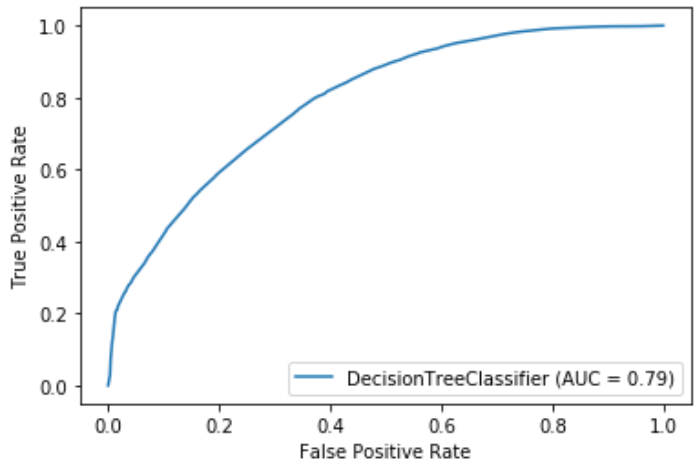


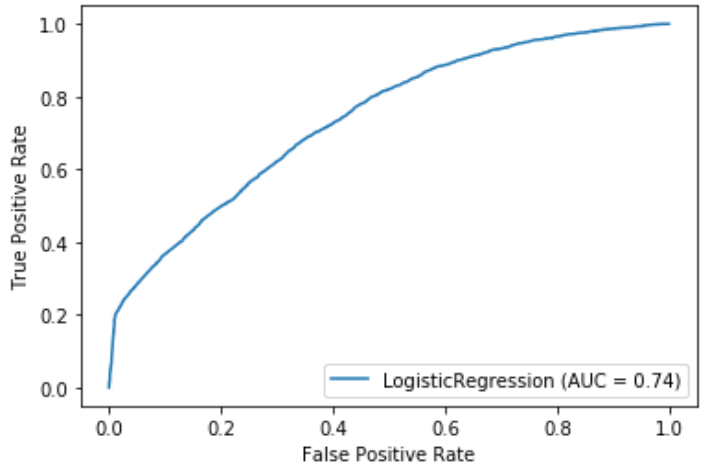
Results

- Jaccard Index
- F1 Score
- Precision Score
- Recall Score
- ROC

  In this problem, lower false positive rate is less important than higher true positive rate. In other words, it is more important to properly predict the high-severity accident properly, if there is room for doubt it is better to prevent.
- Log Loss (Logistic Regression only)







Algorithm	Jaccard	F1 Score	Precision	Recall	AUC (ROC)	Log Loss
KNN	0.6806	0.6797	0.6991	0.6284	0.75	NA
Decision Tree	0.7128	0.7104	0.7582	0.6207	0.79	NA
Logistic Regression	0.6595	0.6585	0.6417	0.7144	0.74	0.5899

Decision Tree is the best classifier out of the three with the best Jaccard Index 0.71, best F1 Score 0.75 and best AUC 0.79