PREDICTING ROAD ACCIDENT SEVERITY

New Ru Wee

3 September 2020

1. Introduction

1.1 Background

Road accidents is a major problem in the modern world – it is said more people are killed by road accidents than plane crashes. The objective of this analysis to investigate which are the factors driving the severity of a collisions and provide insights to the community on possible to reduce road accidents.

The dataset is taken from SDOT Traffic Management Division, Traffic Records Group from 2004 to 2020. The dataset has 38 columns consisting of 194673 observations without any duplicates. This dataset is a hybrid combination of numerical and categorical data types.

1.2 Problem

The first problem would be to identify which are the features which are driving the road accidents severity. Each column in the dataset represent a feature. In process of doing so, the features of importance are also being identified.

1.3 Interest

This analysis will be of interest to the authorities, insurance company and community when trying to reduce severe road accidents. Road accidents not only cause loss of lives but also cause injuries and monetary loss to individual, insurance companies and community.

2. Data Cleansing

The dataset has many columns, several of which are repetitive – for example some columns are represented twice, once with text description and another time with numerical representation, both of which carries the same information. Therefore, the first step taken was to select the feature of interest based on correlation from Exploratory Data Analysis (EDA) – this will be further discussed in the EDA section of this report.

On top of that, there are null values (indicated by “NaN” in pandas) in some of the records which needs to be removed as all machine learning algorithm in Python would not be able to work if there are null values in the dataset.

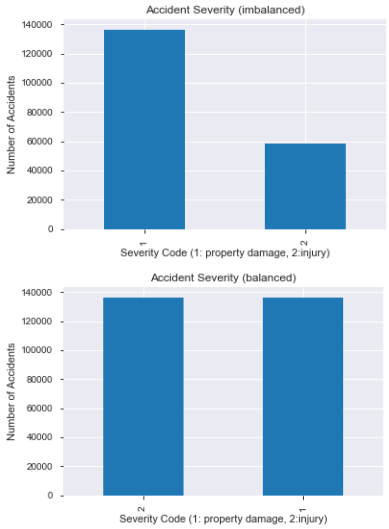
The dataset has many columns, several of which are repetitive – for example some columns are represented twice, once with text description and another time with numerical representation, both of which carries the same information. Therefore, the first step taken was to select the feature of interest based on correlation from Exploratory Data Analysis (EDA) – this will be further discussed in the EDA section of this report.

On top of that, there are null values (indicated by “NaN” in pandas) in some of the records which needs to be removed as all machine learning algorithm in Python would not be able to work if there are null values in the dataset.

In order to use the sklearn library in Python to run the machine learning methods, all the columns have to be converted to numpy array.

Categorical data has to be converted to numerical representation so that it could fit into the machine learning algorithm. This was achieved using LabelEncoder from sklearn library. Next the data has to be normalized using MinMaxScaler also from sklearn library.

As the dataset is imbalanced, we need to perform upsampling to deal with imbalanced dataset.

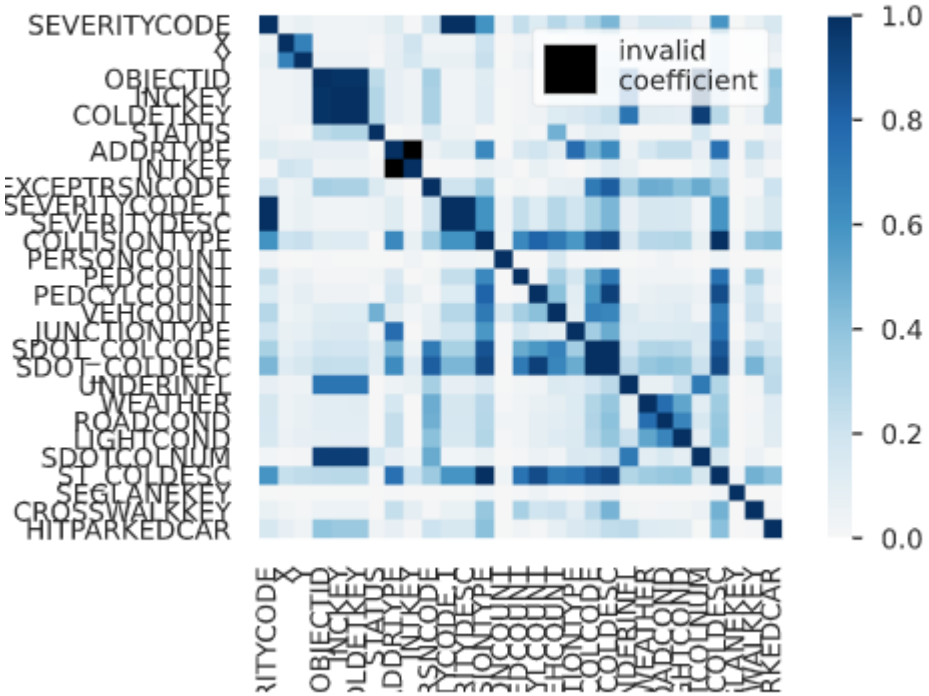


3. EDA

The EDA was first done using pandas dataframe’s describe function to get the Central Frequency of each feature. Next matplotlib’s graphing function was used to visualize the EDA separately for both numerical and categorical features.

Last but not least, I also run pandas profiling to get the EDA – this is where I got the correlation between the features. This is the link the profile report: https://ruwee1982.github.io/

Phik (φk) | Cramér's V (φc)




COLDETKEY
CROSSWALKKEY
INCKEY
INTKEY
OBJECTID
PEDCOUNT
PEDCYLCOUNT
PERSONCOUNT
SDOTCOLNUM
1e7
SDOT_COLCODE
SEGLANEKEY
SEVERITYCODE
SEVERITYCODE.1
VEHCOUNT
X
Y


SEVERITYCODE
SEVERITYCODE
X
Y
OBJECTID
INCKEY
COLDETKEY
INTKEY
SEVERITYCODE.1
PERSONCOUNT
PEDCOUNT
PEDCYLCOUNT
VEHCOUNT
SDOT_COLCODE
SDOTCOLNUM
1e7
SEGLANEKEY
CROSSWALKKEY

4. Classification

6 types of classification methods namely Logistic Regression, Decision Tree, KNN, Linear Discriminant Analysis, Gaussion Naïve Bayes and Support Vector Machine have been used to model the data. All the classification methods used are from sklearn library in Python.

Logistic Regression¶

In [125]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'.format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.65
Accuracy of Logistic regression classifier on test set: 0.65
```

Decision Tree¶

In [126]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 0.95
Accuracy of Decision Tree classifier on test set: 0.80
```

KNN¶

In [127]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.81
Accuracy of K-NN classifier on test set: 0.71
```

Linear Discriminant Analysis¶

In [128]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.65
Accuracy of LDA classifier on test set: 0.65
```

Gaussian Naive Bayes¶

In [129]:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
     .format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.64
Accuracy of GNB classifier on test set: 0.65
```

Support Vector Machine¶

In [130]:

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 0.67
Accuracy of SVM classifier on test set: 0.67
```

5. Conclusion

From the accuracy obtained, it seems like Decision Tree is the best model for this dataset.

6. Recommendation

-