CAR ACCIDENT SEVERITY REPORT : SEATTLE,WASHINGTON

Submitted by: Abdul Saboor

OCTOBER 1, 2020

IBM CAPSTONE PROJECT

1) Introduction

1.1) Background

Seattle is the largest city in the state of Washington , and is a hub to large two tech giants Microsoft and Amazon. Seattle accounts for nearly 3.4 million population, car accidents has become a major issue a lot these days due to increased car population

Nearly almost 1.25 million people die in road crashes each year. Car accidents are one of the leading causes of death. It took a toll of 518 billion USD on US government. According to Seattle Times, the city’s goal is to achieve zero fatalities and serious injuries by 2030.

1.2) Problem

The project aim is to reduce number of accidents by analyzing data that might contribute to the likelihood of potential car accidents. The factors which leads to car accidents can vary a lot , It includes people who are driving very fast due to effect of alcohol, other reasons include weather visibility or road conditions.

1.3) Stakeholders

This will be of huge interest to SDOT(Seattle Department of Transportation) who responsible for the maintenance of the city's transportation systems. Others interested could be car insurance companies , local government of Seattle ,so they can all play important role in decreasing no of accidents in Seattle

2) Data

2.1) Data Source

The data has been provided by SPD(Seattle Police Department) and recorded by Traffic Records Department.The data set has total observations(rows) of 194,673.The main purpose of this report is to predict the accident severity in Seattle, hence the severity code is as follows:

SEVERITY CODE	DESCRIPTION
3	Fatality
2b	Serious Injury
2	Injury
1	Prop Damage
0	Unknown

As the data contains null values and non-relevant columns it is important to clean the data.

2.2) Data Cleaning

As we can see there is a huge imbalance of feature selection 'SEVERITY CODE' which might give us inaccurate results. There is huge difference between first and second row as you can see

```
In [13]: df['SEVERITYCODE'].value_counts()

Out[13]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

Hence it is important to resample so we can have equal data to work on

```
In [22]: df_firstrow=df[df.SEVERITYCODE==1]
         df_secondrow=df[df.SEVERITYCODE==2]

         df_secondrow_sampling=resample(df_firstrow,replace=False,n_samples=58188,random_state=101)

         df_balanced=pd.concat([df_secondrow_sampling,df_secondrow])
         df_balanced.SEVERITYCODE.value_counts()

Out[22]: 2    58188
         1    58188
         Name: SEVERITYCODE, dtype: int64
```

3)Methodology

In this project we have used most significant feature variables like "WEATHER","ROADCOND" and "LIGHTCOND" to predict our target variable or outcome which is "SEVERITYCODE" in this case .Lets look at the table to get further understanding:

FEATURE VARIABLES	DESCRIPTION
WEATHER	Weather condition during the time of collision(wet,dry,clear)
ROADCOND	Road condition during the collision(Wet or Dry)
LIGHTCOND	Conditions of light during collision(bright or dark)

Hence I ran some analysis on features variables like their value count for example on 'LIGHTCOND' to understand the number of of accidents occurring due to different light conditions, same I did with 'ROADCOND' and 'WEATHER'

Therefore I ran some following analysis:

```
In [12]: predictor_df["ROADCOND"].value_counts()
```

```
Out[12]: Dry                124510
         Wet                 47474
         Unknown             15078
         Ice                  1209
         Snow/Slush           1004
         Other                 132
         Standing Water        115
         Sand/Mud/Dirt          75
         Oil                    64
         Name: ROADCOND, dtype: int64
```

```
In [11]: predictor_df["WEATHER"].value_counts()
```

```
Out[11]: Clear                       111135
         Raining                      33145
         Overcast                     27714
         Unknown                      15091
         Snowing                        907
         Other                          832
         Fog/Smog/Smoke                 569
         Sleet/Hail/Freezing Rain       113
         Blowing Sand/Dirt               56
         Severe Crosswind                25
         Partly Cloudy                    5
         Name: WEATHER, dtype: int64
```

```
In [14]: predictor_df["LIGHTCOND"].value_counts()
```

```
Out[14]: Daylight                    116137
         Dark - Street Lights On      48507
         Unknown                      13473
         Dusk                          5902
         Dawn                          2502
         Dark - No Street Lights       1537
         Dark - Street Lights Off      1199
         Other                          235
         Dark - Unknown Lighting         11
         Name: LIGHTCOND, dtype: int64
```

4)Modeling and Evaluation

Following machine learning models are applied Logistic Regression, K-Nearest Neighbor, Decision Tree. The reason we are not using SVM Support Vector Machine Model is because they are inaccurate for large data sets, Hence SVM works best for the data which filled with text and images.

Furthermore after preprocessing and scaling the data we applied machine learning models, I have used sckit library to build the model, then we evaluated the model and results were shown as follows:

KNN Model:

kNN

In [30]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

Ks=10
mean_acc=np.zeros((Ks-1))
std_acc=np.zeros((Ks-1))
ConfusionMatrix=[]
for n in range(1,Ks):
    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
```

Out[30]:
```
array([0.55200069, 0.54472546, 0.54234812, 0.5449546 , 0.55182883,
       0.5465013 , 0.5467018 , 0.5465013 , 0.54627216])
```

In [31]:
```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.552000687423023 with k= 1
```

KNN Model Evaluation:

knn evaluation

In [33]:
```
knn_yhat = neigh.predict(X_test)


jaccard_score(Y_test, knn_yhat)
```

Out[33]:
```
0.23779050185247558
```

In [34]:
```
f1_score(Y_test, knn_yhat, average='macro')
```

Out[34]:
```
0.5125097414762372
```

Decision Tree Model:

Decision Tree

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth = 7)

dt.fit(X_train,Y_train)
```

Out[36]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [38]:
```
dt_y_pred = dt.predict(X_test)
```

Decision Tree Model Evaluation:

Decision Tree Evaluation

In [39]:
```
jaccard_score(Y_test, dt_y_pred)
```

Out[39]:
```
0.2856941574300207
```

In [40]:
```
f1_score(Y_test, dt_y_pred, average='macro')
```

Out[40]:
```
0.5430741006902506
```

Logistic Regression:

Logistic Regression

In [41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=6, solver='liblinear').fit(X_train,Y_train)
```

In [42]:
```
LR_y_pred = LR.predict(X_test)
```

In [43]:
```
LR_y_prob = LR.predict_proba(X_test)
```

In [45]:
```
LR_y_prob = LR.predict_proba(X_test)
log_loss(Y_test, LR_y_prob)
```

Out[45]: 0.684679793585963

Logistic Regression Model Evaluation:

Logistic Regression Evaluation

In [46]:
```
jaccard_score(Y_test, LR_y_pred)
```

Out[46]: 0.277022568298799

In [47]:
```
f1_score(Y_test, LR_y_pred, average='macro')
```

Out[47]: 0.5155215511318116

Furthermore we found the accuracy of the model stated below:

```
In [48]: from sklearn.metrics import accuracy_score
         print("KNN Accuracy: ", accuracy_score(Y_test, knn_yhat))

         KNN Accuracy:  0.5462721622318334

In [50]: print("Decision Tree Accuracy: ", accuracy_score(Y_test, dt_y_pred))

         Decision Tree Accuracy:  0.5643743018359924

In [52]: print("Logistic Regression Accuracy: ", accuracy_score(Y_test, LR_y_pred))

         Logistic Regression Accuracy:  0.5292870850399565
```

5)Results

Machine Learning Model	Jaccard-Index	F1 Score	Accuracy
KNN	0.237	0.512	0.546
Decision Tree	0.285	0.543	0.564
Logistic Regression	0.277	0.515	0.529

6)Conclusion

Based on the results we can see Decision Tree is the best machine learning model.However these models could have done better if we have more balanced dataset for target variable, factors like precautionary measures when driving and etc.