IBM Data Science Capstone

Accident Severity Prediction

Introduction:

The data set covers the Seattle area accident history and statistics. Given the detrimental impact of accidents on both human and physical capital, it becomes imperative to understand this data set and put some statistics based computational efforts towards minimizing the incidence of these accidents.The underlying objective of using and understanding this data set is to use historical accident data and find out what conditions usually lead these accidents. This study should benefit any car manufacturer looking to develop alarms for drivers based on the findings that will emphasize the conditions under which accidents are most probable.

Data:

The data set consists of all collisions provided by the Seattle Police Department and recorded by Traffic Records.This includes all types of collisions. The time frame of the data collected is from 2004 to present. The target variable in this data set is the "severity" in terms of human fatality, traffic delay, property damage, or any other type of accident bad impact. The data broadly covers the time, place and conditions under which these collisions took place. The severity code is described as follows:

- 3 — Fatality
- 2b — Serious injury
- 2 — Injury
- 1 — Property damage
- 0 — Unknown

Methodology:

The process was divided into three broad steps: Data Loading and Cleaning, Data Pre-Processing for Model Building, Model Building and Evaluation.

Data Loading and Cleaning: In the first step, data was directly imported from the online database and read it into a pandas dataframe. This dataframe was then cleaned by removing unnecessary columns and dropping any rows that had missing data.

```
In [16]: df = pd.read_csv("https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv")

/opt/conda/envs/Python36/lib/python3.6/site-packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Columns (33) have mixed types. Specify dtype opt
  interactivity=interactivity, compiler=compiler, result=result)

In [17]: df.shape

Out[17]: (194673, 38)

In [18]: df.drop(columns =['OBJECTID', 'INCKEY','COLDETKEY','REPORTNO','STATUS','INTKEY','LOCATION','INCDATE','EXCEPTRSNCODE','EXCEPTRSNDESC','SDOT_COLCODE', 'INATTE
         'SDOTCOLNUM','SPEEDING','ST_COLCODE','ST_COLDESC','SEGLANEKEY','CROSSWALKKEY', 'SEVERITYCODE.1','SEVERITYDESC','PERSONCOUNT', 'PEDCOUNT','PEDCYLCOUNT','VEHC
         df.shape

Out[18]: (194673, 4)

In [19]: df.dropna(inplace = True)
         df.shape

Out[19]: (189337, 4)
```

At this point, the feature set included Weather, Road Condition and Light Condition and the target variable being the severity code.

Data Pre-Processing for Model Building: The data was pre-processed to be directly fed into the model. The **first step** involved encoding the categorical variables. In the **second step**, the data was balanced based on the value counts of the target variable. It showed that the data was mostly present for cases that resulted in a severity code of 1, which would have biased the model. In order to do, the data set which consisted of the severity code ==1 was down sampled to match the sample count of the data set which consisted of the severity code ==2. In the **third step**, the data was clearly dived into two different numpy arrays, one for the target variable, y and one for the feature set, X. In the **fourth step**, the data was standardized. Finally, in the **fifth step**, the data was split into training set and test set in order to feed into the various models

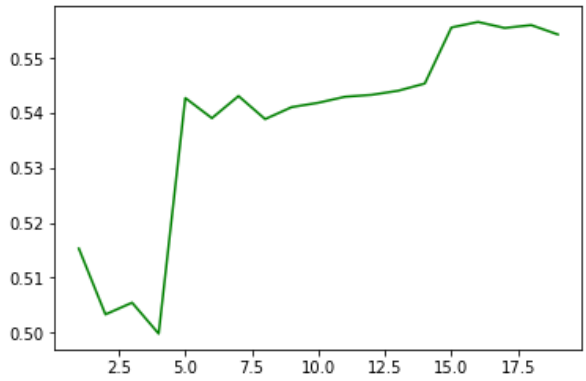
```
In [27]: #Train-Test Split
         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train , y_test = train_test_split(X,y, test_size =0.2, random_state = 11)
         print(x_train.shape)
         print(y_train.shape)
         print(x_test.shape)
         print(y_test.shape)

         (91283, 29)
         (91283,)
         (22821, 29)
         (22821,)
```

Model Building and Evaluation: In this step, four different ML models were employed to predict the severity code of the accidents based on the selected feature set.

A. KNN: An iterative approach was used to find the best 'k' value that maximized the mean accuracy score of the data.

```
In [32]: import matplotlib.pyplot as plt
         plt.plot(range(1,Ks),mean_acc,'g')
```

```
Out[32]: [<matplotlib.lines.Line2D at 0x7feaed8709b0>]
```




```
In [30]: print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.5565926120678323 with k= 16
```

B. Decision Tree Classifier:

```
In [33]: from sklearn.tree import DecisionTreeClassifier
         sev_tree = DecisionTreeClassifier(criterion ="entropy").fit(x_train, y_train)
         yhatDT = sev_tree.predict(x_test)
```

C. Support Vector Machine:

```
In [37]: from sklearn import svm
         sev_svm = svm.SVC().fit(x_train, y_train)
         yhatSVM = sev_svm.predict(x_test)
```

D. Logistic Regression:

```
In [34]: from sklearn.linear_model import LogisticRegression
         sev_lr = LogisticRegression().fit(x_train, y_train)
         yhatLR = sev_lr.predict(x_test)
```

Results:

Above described models were evaluated based on the Jaccard Similarity Score, F-1 score and log_loss score.

Jaccard Score:

In [39]:
```
#Jaccard_Score
from sklearn.metrics import jaccard_similarity_score
#KNN
print(jaccard_similarity_score(y_test, neigh16.predict(x_test)))
#Decision_Tree
print(jaccard_similarity_score(y_test, sev_tree.predict(x_test)))
#SVM
print(jaccard_similarity_score(y_test, sev_svm.predict(x_test)))
#Logistic_Regression
print(jaccard_similarity_score(y_test, sev_lr.predict(x_test)))
```

```
0.5565926120678323
0.559572323736909
0.5597476008939135
0.560930721703694
```

F-1 score

In [42]:
```
#F1 score
#KNN
print(metrics.f1_score(y_test, neigh16.predict(x_test),average='weighted'))
#Decision_Tree
print(metrics.f1_score(y_test, sev_tree.predict(x_test), average='weighted'))
#SVM
print(metrics.f1_score(y_test, sev_svm.predict(x_test),average='weighted'))
#Logistic_Regression
print(metrics.f1_score(y_test, sev_lr.predict(x_test),average='weighted'))
```

```
0.5207879568787507
0.5311844497581388
0.5305679928878598
0.530870947091857
```

Log-Loss

In [41]:
```
#LogLoss
#Logistic_Regression
from sklearn.metrics import log_loss
yhatLR_prob = sev_lr.predict_proba(x_test)
print(log_loss(y_test,yhatLR_prob))
```

```
0.6651707921766155
```

Discussion:

Based on the model evaluation, it was determined that the Logistic Regression is the most suitable model to predict the accident severity with a Log Loss score of 0.66. This was also expected because the dataset only consisted of two severity code outcomes, and based on that binary output result condition, a Logistic Regression is the best suited model. The model still needs further finetuning to get the best possible accuracy.

Conclusion:

It is evident that the road, weather and light conditions are extremely important while driving. This study basically shows a way to predict accident severity ideally using a logistic regression model to prevent or reduce the severity of accidents in the future based on historical data.