Accident Severity Prediction to reduce Road Traffic Accidents in Seattle, WA

IBM CAPSTONE PROJECT

Table Of Contents:

INTRODUCTION

- Every year the lives of approximately 1.35 million people are cut short as a result of a road traffic crash.

- 20 - 50 million fatal injuries

- Road traffic injuries cause considerable economic losses

- Study of influencing factors of traffic accidents is an important research direction in the field of traffic safety

Business Problem

- A model must be developed to predict the severity of an accident given the current weather, the road and visibility conditions

- The end user will be alerted to be more careful if the conditions are bad in real-time.

- Main objective of this project is to make a supervised prediction model that predicts the severity of an accident given certain circumstances (features).

Data

- Dataset originally provided by Seattle Department of Transportation(SDOT) Traffic Management Division, Traffic Records Group

- Our target variable will be '**SEVERITYCODE**'
  - It is used to measure the severity of an accident from 0 to 3 (including a "2b", as per the metadata.

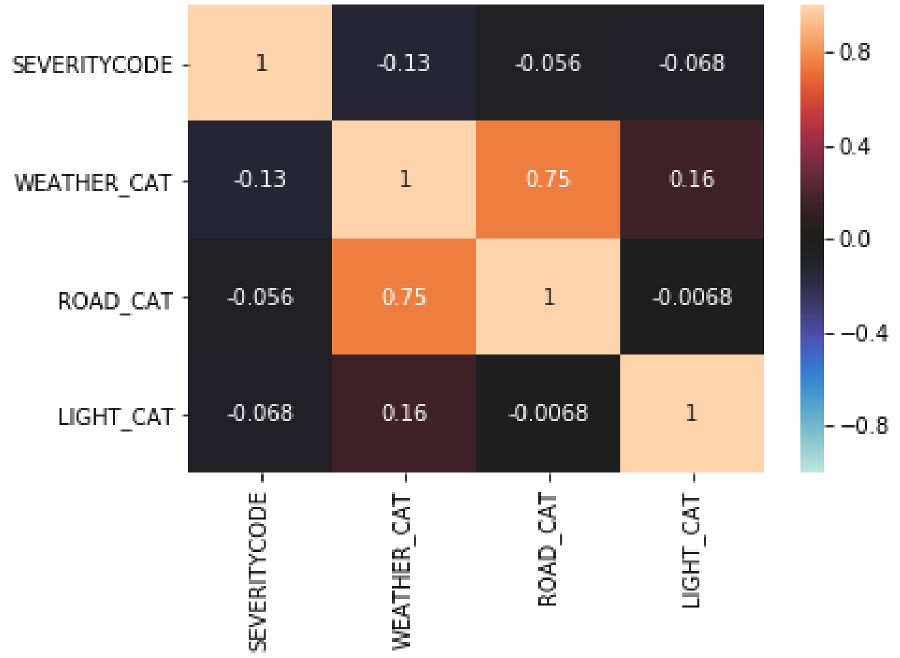
- Attributes used weigh the severity of an accident are '**WEATHER**', '**ROADCOND**' and '**LIGHTCOND**'.

- The entire dataset originally had 194,673 rows (Instances) and 38 columns (Features).

Methodology

- The pre-processed data analyzed using Exploratory Data Analysis and Inferential Statistical Analysis.
- Based on the inference, we shall proceed with the selection of Machine Learning Algorithm for our model.

```
In [256]: sns.heatmap(df_new.corr(), annot = True, vmin=-1, vmax=1, center= 0)
Out[256]: <matplotlib.axes._subplots.AxesSubplot at 0x7f86c4d92e48>
```



Machine Learning Algorithms & Evaluation

1. **K-Nearest Neighbor (KNN)** :

- Here we will be trying different values for k and get the result of the best k-value which will be used to predict the output

- KNN will help us predict the severity code of an outcome by finding the most similar data-point within k distance.

K Nearest Neighbor (KNN)

```
In [248]: # First, we are going to use the K Nearest Neighbor (KNN) Algorithm.

          #Train Model and Predict
          k=3
          kNN_model = KNN(n_neighbors=k).fit(X_train,y_train)
          kNN_model

Out[248]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                               weights='uniform')
```

```
In [204]: # Building the model again, using k=5
          k = 5
          #Train Model and Predict
          kNN_model = KNN(n_neighbors=k).fit(X_train,y_train)
          kNN_model

Out[204]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                               weights='uniform')
```

KNN Model Evaluation

```
In [205]: print("KNN Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat))
          print("KNN F1-score: %.2f" % f1_score(y_test, yhat, average='weighted') )

          KNN Jaccard index: 0.55
          KNN F1-score: 0.52
```

2. **Decision Tree** :

- A non-parametric supervised learning method used for classification and regression.
- Gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision.
- In our context, the decision tree observes all possible outcomes of different weather conditions.

Decision Tree

In [206]:
```
DT_model = DTC(criterion="entropy", max_depth = 4)
DT_model.fit(X_train,y_train)
DT_model
```

Out[206]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
            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=False, random_state=None,
            splitter='best')
```

In [207]:
```
yhat1 = DT_model.predict(X_test)
yhat1
```

Out[207]:
```
array([2., 2., 1., ..., 1., 1., 2.])
```

Decision Tree Model Evaluation

In [208]:
```
print("DT Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat1))
print("DT F1-score: %.2f" % f1_score(y_test, yhat1, average='weighted') )
```

```
DT Jaccard index: 0.56
DT F1-score: 0.53
```

3. <u>Logistic Regression</u> :

- Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

- Logistic regression is a go-to method for binary classification problems.

Logistic Regression

In [192]:
```
LR_model = LR(C=0.01,solver='liblinear').fit(X_train,y_train)
LR_model
```

Out[192]:
```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)
```

In [193]:
```
yhat2 = LR_model.predict(X_test)
yhat2
```

Out[193]:
```
array([2, 2, 2, ..., 2, 2, 2])
```

Logistic Regression Model Evaluation

In [194]:
```
yhat_prob = LR_model.predict_proba(X_test)
print("LR Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat2))
print("LR F1-score: %.2f" % f1_score(y_test, yhat2, average='weighted') )
print("LR LogLoss: %.2f" % log_loss(y_test, yhat_prob))
```

```
LR Jaccard index: 0.54
LR F1-score: 0.53
LR LogLoss: 0.68
```

Results

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.55	0.52	NA
Decision Tree	0.56	0.53	NA
LogisticRegression	0.54	0.53	0.68

Discussion

- Problems faced with dataset :
  - Mixed data types
  - Imbalanced data

- Removal of null values (rows), Label Encoding and Down-sampling were performed.

- Evaluation metrics used to test the accuracy of our models were Jaccard index, f-1 score and log_loss for logistic regression.

- Although KNN and Decision Tree seem to be ideal for this project, logistic regression made most sense because of its binary nature.

Future Scope

- We have just scratched the surface of this dataset with our use case.
- Scope for a vast variety of analytics and modeling that can be done with this dataset for various other use cases (for example, finding out the relation between various alleys/intersections with collision severity in order to improve the infrastructure).
- Optimizing the dataset and trying other algorithms will provide high scope for improvement of our model in the future.

Conclusion

- Our model could predict the accident severity with an accuracy of 54%.

- Accidents can be avoided if the end user is provided with real-time information on the road and lighting conditions and also regular updates on the weather using our application.

Questions???

Thank You!!!