IBM Data Science Capstone: Car Accident Severity

SEPTEMBER 2020

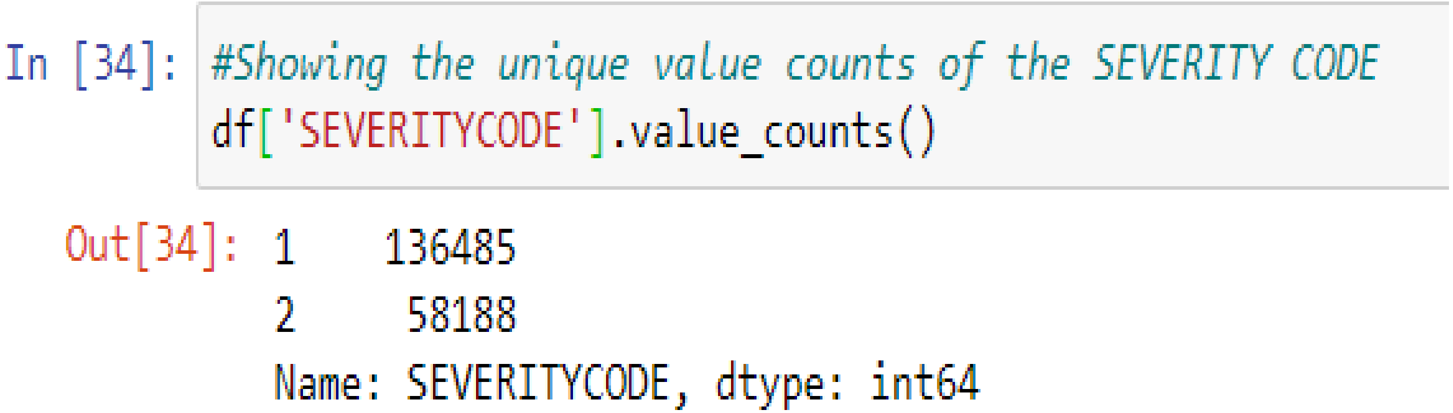
Business Understanding

In an effort to reduce the frequency of car collisions in a community, an algorithm must be developed to predict the severity of an accident given the current weather, road, and visibility conditions.

Data Understanding

Using the data provided by Coursera on Collisions, I will investigate the connection between the severity of car accidents and weather conditions. This data provides collisions from 2004 to the present in Seattle.

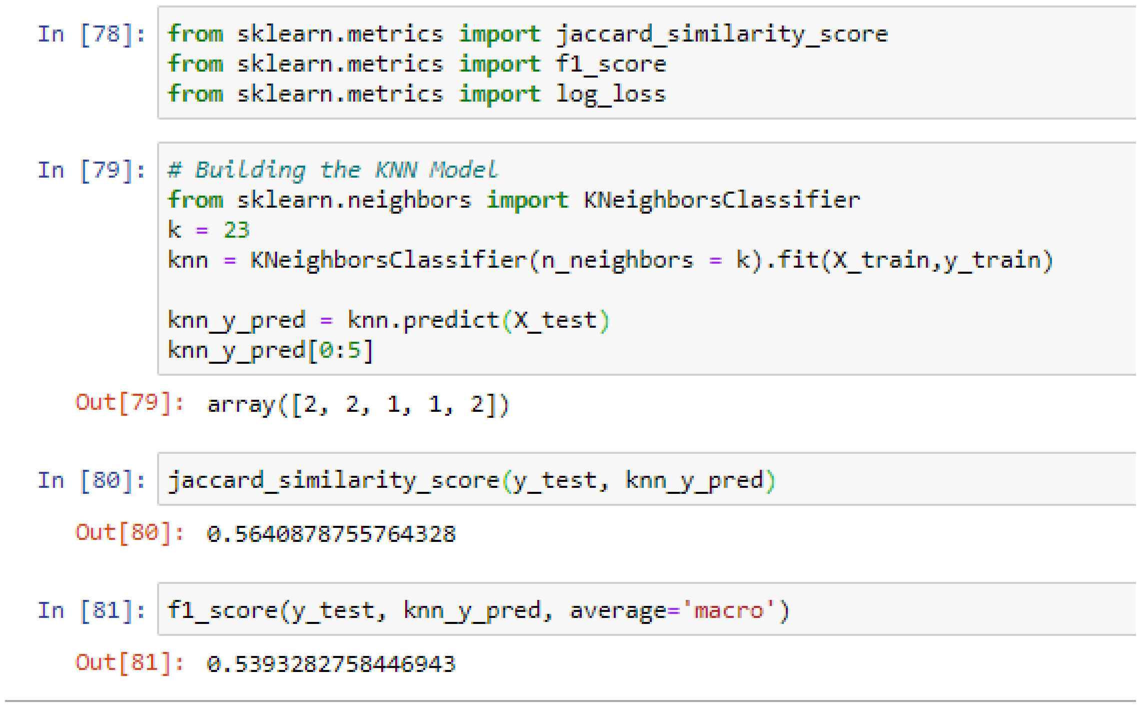
Data Preprocessing



```
In [34]: #Showing the unique value counts of the SEVERITY CODE
         df['SEVERITYCODE'].value_counts()

Out[34]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

To get a good understanding of the dataset, I have checked different values in the features. The results show, the target feature is imbalance, so we use a simple statistical technique to balance it.

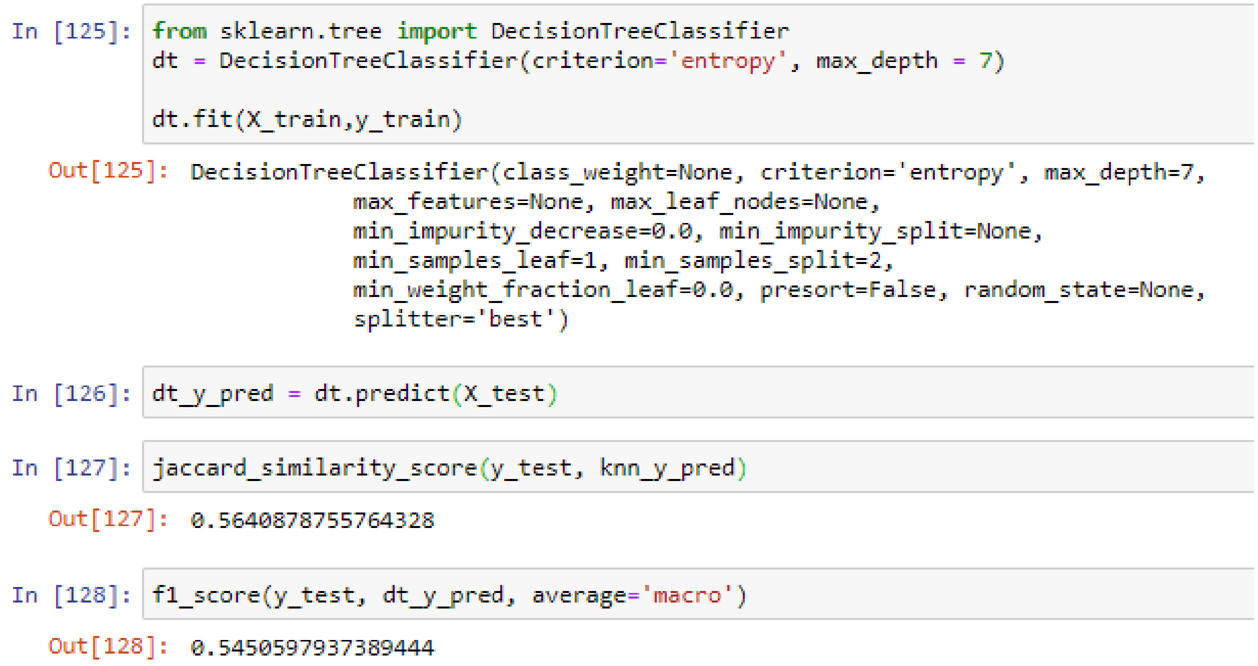
K-Nearest Neighbor (KNN)



```
In [78]: from sklearn.metrics import jaccard_similarity_score
         from sklearn.metrics import f1_score
         from sklearn.metrics import log_loss

In [79]: # Building the KNN Model
         from sklearn.neighbors import KNeighborsClassifier
         k = 23
         knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)

         knn_y_pred = knn.predict(X_test)
         knn_y_pred[0:5]

Out[79]: array([2, 2, 1, 1, 2])

In [80]: jaccard_similarity_score(y_test, knn_y_pred)

Out[80]: 0.5640878755764328

In [81]: f1_score(y_test, knn_y_pred, average='macro')

Out[81]: 0.5393282758446943
```

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

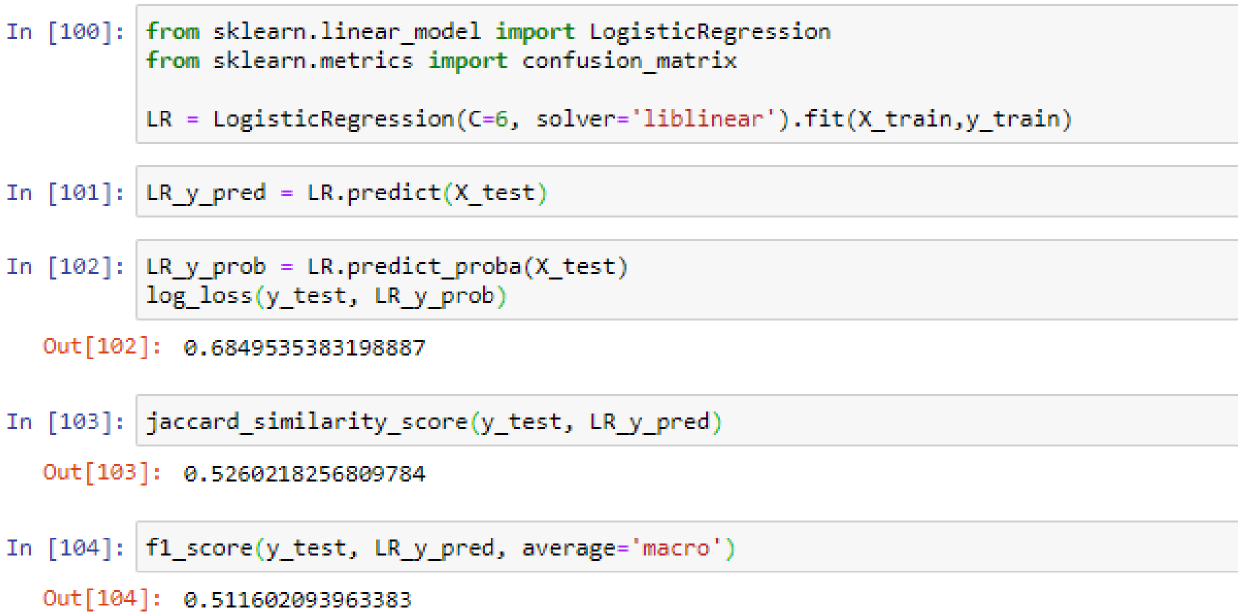
Decision Tree



```
In [125]: from sklearn.tree import DecisionTreeClassifier
          dt = DecisionTreeClassifier(criterion='entropy', max_depth = 7)

          dt.fit(X_train,y_train)

Out[125]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=7,
                      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 [126]: dt_y_pred = dt.predict(X_test)

In [127]: jaccard_similarity_score(y_test, knn_y_pred)

Out[127]: 0.5640878755764328

In [128]: f1_score(y_test, dt_y_pred, average='macro')

Out[128]: 0.5450597937389444
```

A decision tree model gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision. It context, the decision tree observes all possible outcomes of different weather conditions.

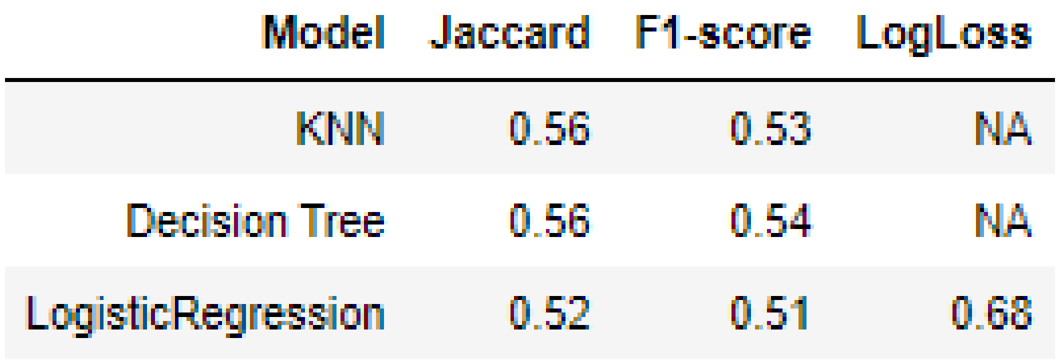
Logistic Regression



```
In [100]: from sklearn.linear_model import LogisticRegression
          from sklearn.metrics import confusion_matrix

          LR = LogisticRegression(C=6, solver='liblinear').fit(X_train,y_train)

In [101]: LR_y_pred = LR.predict(X_test)

In [102]: LR_y_prob = LR.predict_proba(X_test)
          log_loss(y_test, LR_y_prob)

Out[102]: 0.6849535383198887

In [103]: jaccard_similarity_score(y_test, LR_y_pred)

Out[103]: 0.5260218256809784

In [104]: f1_score(y_test, LR_y_pred, average='macro')

Out[104]: 0.511602093963383
```

Because our dataset only provides us with two severity code outcomes, our model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

Results and Evaluations



Model	Jaccard	F1-score	LogLoss
KNN	0.56	0.53	NA
Decision Tree	0.56	0.54	NA
LogisticRegression	0.52	0.51	0.68

Although the first two are ideal for this project, logistic regression made most sense because of its binary nature.

Conclusion

Based on the dataset provided for this capstone from weather, road, and light conditions pointing to certain classes, we can conclude that particular conditions have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2).

THANK YOU!