IBM Data Science Professional Certificate:

Applied Data Science Capstone

Predicting Severity of Car Accidents in Seattle, Washington, USA

Ron DiPaola

October 2020

1. Introduction / Business Understanding

1.1 Background

Motor vehicle accidents statistics in the United States are sobering with over 36k fatal crashes and 2.7 million people injured in 2018 alone[1]. Motor vehicle traffic crashes are the leading cause of death for youth (16-20) and young adults (21-24)[2] in the United States. The economic and societal costs are enormous, exceeding $836 billion US dollars in 2010. New methods to help reduce the number of accidents can go a long way in saving lives and the economic costs associated with these crashes.

1.2 Business Problem

Finding ways to reduce accidents has a vast array of interested stakeholders: government, police, first responders, hospitals, insurance companies, parents, drivers, passengers, users of public transportation, etc.

The city of Seattle, Washington, USA has a Department of Transportation (SDOT) which collects and publishes accident data. We will use this data to try to create a model to predict accident severity which in turn can be used by SDOT to help alert stakeholders that there are conditions present which are highly associated with collisions.

2. Data

2.1 Data Source

The data used in this model comes from the Seattle Department of Transportation's Open Data Portal:

https://data.seattle.gov/

Will we be using the collision data which can be found below but was provided by Coursera for use with this project.

https://data-seattlecitygis.opendata.arcgis.com/datasets/5b5c745e0f1f48e7a53acec63a0022ab_0

Metadata for use in understanding the attributes can be found below but has been provided by Coursera for use with this project:

https://www.seattle.gov/Documents/Departments/SDOT/GIS/Collisions_OD.pdf

[1] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812948
[2] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812951

2.2 Data Summary

The data set is of an adequate size having more than 194k rows (collisions) and 37 attributes for each accident and is recent, with entries from January 2014 to May of 2020. It also contains the value will be trying to predict in SEVERITYCODE

SEVERITYCODE	Text, 100	A code that corresponds to the severity of the collision: • **3**—fatality • **2b**—serious injury • **2**—injury • **1**—prop damage • **0**—unknown

2.3 Data Cleaning

While the data is large it will need to be cleaned up to be used in our model. Many of the collision records contain missing values, so rows missing values will be removed from the data set. Some of the attributes contain values which are not helpful such as “Other” and “Unknown”, these records will be removed as well.

Some of the attributes contain a word or combination of words, so we will convert them to category codes to use in the model. An example looks like this before categorizing for use in the model:

```
df_new["JUNCTIONTYPE"].value_counts()
```

```
Mid-Block (not related to intersection)              63190
At Intersection (intersection related)              56044
Mid-Block (but intersection related)                18197
Driveway Junction                                    6363
At Intersection (but not related to intersection)    1556
Ramp Junction                                         126
Unknown                                                 4
```

After the categorization, notice the “Unknown” records have been removed and now we have a numerical value instead of the words:

```
df_new["JUNCTIONTYPE_CATEGORY"].value_counts()
```

```
4    63190
1    56044
3    18197
2     6363
0     1556
5      126
```

Finally, the data set also has a bias towards one type of accident (Severity 1), so we will resample to reduce the bias giving us an equal amount of severity code records to run through our model.

Before resampling:

```
# rebalancing data, see counts of target variables
df_new['SEVERITYCODE'].value_counts()
```

```
1    96002
2    49474
```

After resampling:

```
balanced_df.SEVERITYCODE.value_counts()
```

```
2    49474
1    49474
```

2.4 Feature Selection

In examining the data set, the following features will be used to help predict the severity of an accident, basic correlation analysis was performed to see which attributes contribute to SEVERITYCODE.

These features were kept while all other data was discarded.

Field	Description
WEATHER	A description of the weather conditions during the time of the collision.
ROADCOND	The condition of the road during the collision.
LIGHTCOND	The light conditions during the collision
ADDRTYPE	Collision address type: • Alley • Block • Intersection
COLLISIONTYPE	Collision type
JUNCTIONTYPE	Category of junction at which collision took place
PEDCOUNT	The number of pedestrians involved in the collision.
PEDCYLCOUNT	The number of bicycles involved in the collision.

3. Methodology

3.1 Exploratory Data Analysis

After performing basic correlation methods and using reasoning that light, road and weather conditions would contribute to the severity, I looked at the values and counts for each of the attributes making sure they made sense and determining which still made sense to keep.

Example:

```
df_new["COLLISIONTYPE"].value_counts()
```

```
Angles        33668
Parked Car    32427
Rear Ended    32348
Sideswipe     17422
Left Turn     13320
Pedestrian     6257
Cycles         5191
Right Turn     2801
Head On        1938
```

3.2 Supervised Machine Learning

We are going to use four types of supervised machine learning where we teach the model by loading the model with knowledge so that we can have it predict future instances.

First, we take the cleaned dataset and split it into testing and training subsets (containing 30% and 70% of the samples) using the scikit learn “train_test_split” method.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=4)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (69263, 7) (69263,)
Test set: (29685, 7) (29685,)
```

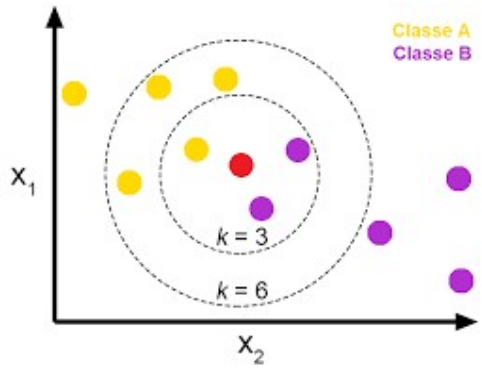
We separate the data to train the models based the training data and then take the model after it has been trained and run it against test data to see how it performed. Each of the models will use the same training and testing sets so we can evaluate their accuracy.

Four models were trained and evaluated:

K-Nearest Neighbors

The K-Nearest Neighbors algorithm is a classification algorithm which takes a bunch of labelled points and uses them to learn how to label other points. This algorithm classifies cases based on their similarity to other cases. In K-Nearest Neighbors, data points that are near each other are said to be neighbors. K-Nearest Neighbors is based on this paradigm.

A visualization of k-Nearest Neighbors:



For this project I ran the model looking for the best K up to a value of 50, because it runs in a reasonable amount of time on my machine, with more resources, I could bump the number up to 100 or 200 and see if there is a better result.

```
Ks = 50
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    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
```

```
array([0.62068152, 0.60588493, 0.66409384, 0.65630793, 0.66743065,
       0.64673565, 0.65897064, 0.65492602, 0.6606559 , 0.65438673,
       0.6638579 , 0.64451111, 0.6644983 , 0.65492602, 0.65772355,
       0.64430887, 0.68010381, 0.64542115, 0.67862078, 0.64538744,
       0.64926354, 0.64761199, 0.67346388, 0.67046412, 0.67043042,
       0.65064545, 0.66668914, 0.66571169, 0.6696889 , 0.6677677 ,
       0.67049783, 0.67002595, 0.66992484, 0.65445414, 0.65512825,
       0.65148809, 0.65529677, 0.65576865, 0.65570124, 0.65651016,
       0.65843136, 0.6561057 , 0.65988068, 0.65826283, 0.65961104,
       0.65816172, 0.65849877, 0.65785837, 0.65873471])
```

After looking for the best value of K, it turned out to be 17, which I then fed into the model:

```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.6801038120597256 with k= 17
```

```
#Re-run the model with the best K
print("Fitting with k="+str(mean_acc.argmax()+1))
kNN_model = KNeighborsClassifier(n_neighbors=mean_acc.argmax()+1).fit(X_train,Y_train)
kNN_yhat = kNN_model.predict(X_test)
kNN_model
```

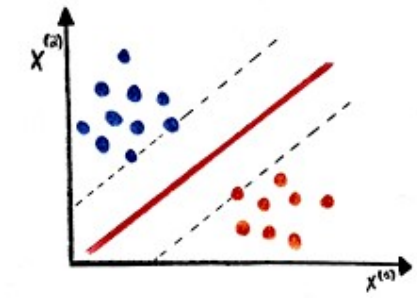
```
Fitting with k=17

KNeighborsClassifier(n_neighbors=17)
```

Support Vector Machine

A Support Vector Machine (SVM) is a supervised algorithm that can classify cases by finding a separator. SVM works by first mapping data to a high dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. Then, a separator is estimated for the data.

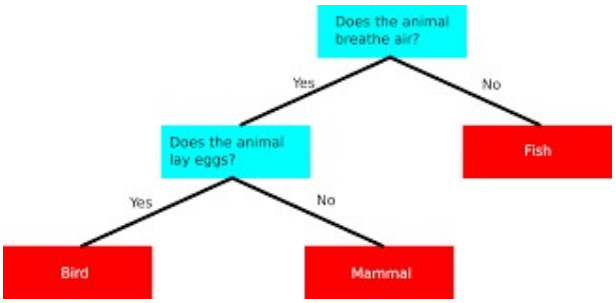
An example visualization of an SVM:



Decision Tree

Decision tree models identify the key features on which the data can be partitioned (and the thresholds at which to partition the data) in the hope of arriving, after some iterations, at "leaves" which contain only accidents belonging to one target variable value.

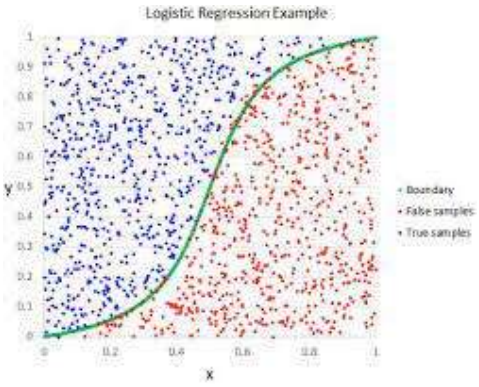
An example decision tree:



Logistic Regression

Logistic regression is a statistical and machine learning technique for classifying records of a dataset based on the values of the input fields. Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one. In linear regression, we might try to predict a continuous value of variables such as the price of a house, blood pressure of a patient, or fuel consumption of a car. But in logistic regression, we predict a variable which is binary such as yes/no, true/false, successful or not successful, pregnant/not pregnant, and so on, all of which can be coded as zero or one. I have included it here because the SEVERITYCODE we are trying to predict is binary because only two possible outcomes are possible. If the sample data had any additional entries such as “3 Fatality” or “2b Serious Injury” this would not be a good model to use.

A visualization of logistic regression:



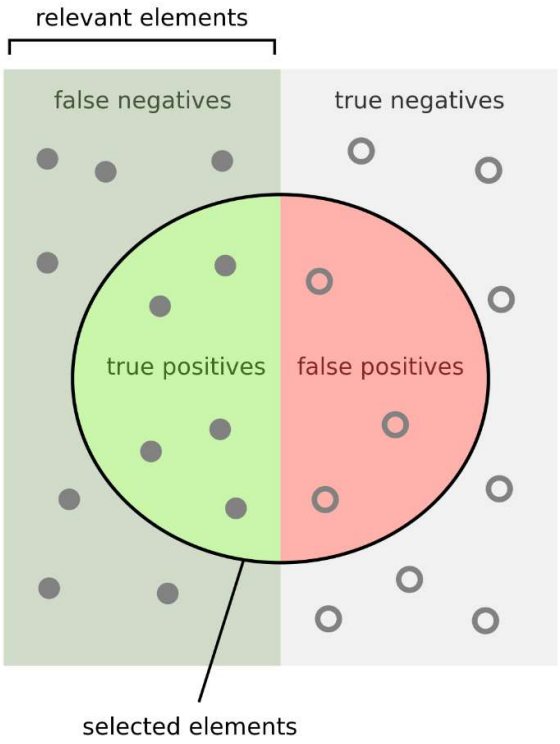
4. Results

4.1 Measures of accuracy for machine learning

Precision is a measure of the accuracy, provided that a class label has been predicted. It is defined by: precision = True Positive / (True Positive + False Positive) and Recall is the true positive rate. It is defined as: Recall = True Positive / (True Positive + False Negative). We can calculate the precision and recall of each class. Now we’re in the position to calculate the F1 scores for each label, based on the precision and recall of that label. The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (which represents perfect precision and recall) and its worst at 0.

The Jaccard index, also known as the Jaccard similarity coefficient, is the size of the overlap divided by the size of the union of two label sets.

F1 Score



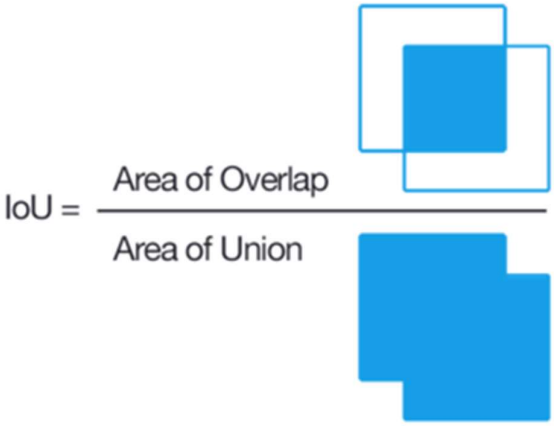
How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Jaccard Index



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

4.2 Machine Learning Results for Predicting Severity Code

After running each of the models I then tested for accuracy in the manner below, only logistic regression is shown here, see jupyter notebook for the rest.

```
#Model evaluation
print("Accuracy of Logistic Regression model:")
print("Train set Accuracy: ", metrics.accuracy_score(Y_train, LR_model.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(Y_test, LR_yhat))
print("Jaccard index: %.3f" % jaccard_score(Y_test, LR_yhat))
print("F1-score: %.3f" % f1_score(Y_test, LR_yhat, average='weighted') )
print("R2-score: %.3f" % r2_score(LR_yhat , Y_test) )
print(classification_report(Y_test, LR_yhat))
```

```
Accuracy of Logistic Regression model:
Train set Accuracy:  0.632701113727303
Test set Accuracy:  0.6277933196265463
Jaccard index: 0.470
F1-score: 0.627
R2-score: -0.496
              precision    recall  f1-score   support

           1       0.62      0.66      0.64     14766
           2       0.64      0.59      0.62     14903

    accuracy                           0.63     29669
   macro avg       0.63      0.63      0.63     29669
weighted avg       0.63      0.63      0.63     29669
```

Model	F1 Score	Jaccard Index
K Nearest Neighbor	0.677	0.472
Decision Tree	0.682	0.461
SVM	0.628	0.470
Logistic Regression	0.627	0.470

For prediction of accident severity given the data provided and using F1 and Jaccard Index as our benchmarks for evaluating the accuracy of the model, a decision tree slightly outperforms K-Nearest Neighbor for the best model for predicting the severity of accidents. None of the models is particularly great, but additional data could help bring the numbers up and make the predictions better.

5. Discussion

5.1 Observations

A little brainstorming or discussion with another interested party would help you to conclude that road, weather and light conditions all contribute to the severity of an automobile accident. However, more thought and analysis are needed to see how the junction type, address type, pedestrians and bicyclists involved all play a part in helping to determine that severity.

5.2 Recommendations

The data collected here is after the fact on accidents and is incomplete if you look at it from a different perspective. Weather is reported at the time of the accident, but what happened right before the accident weather-wise? Same with light conditions, was it sunny and a storm rolled in darkening the skies and bringing precipitation or had it been overcast all day? Knowing some of this information can help with a more accurate prediction.

6. Conclusion

6.1 Severity Can Be Predicted

We've shown that many of the variables collected by SDOT can help predict the severity of an accident with decent accuracy considering no two accidents are the same, from driver's ages to speed to make/model/year of a vehicle. Using the machine learning model we've created can help SDOT with their predictions.

6.2 What can be done with the model now that we have it?

SDOT can warn stakeholders via mobile text alerts, Twitter, mobile app alerts, electronic signage on the roads or publishing to news organizations. SDOT can also alter electronic speed limits if necessary, suggest an increased police presence in an area, and warn first responders and hospitals so they can be prepared when conditions favor a severe accident. This can help prepare all the stakeholders for what they may encounter when called to an accident scene or before an ambulance arrives at hospital.

6.3 The Future

Going forward I hope to see SDOT and other departments of transportation collecting even more information about accident conditions with the help of IOT (Internet of Things). IOT sensors can measure traffic density and speeds with remote sensors. These can then be fed into a revised model to help save lives and the economic costs to us all.