Accident Severity Machine Learning Models Based On Seattle Traffic Data

M. V. Tegethoff

September 2020

Table of Contents

Introduction

Accident severity is defined as the understanding if traffic accidents result in injury or property damage. Reducing the number of accidents that result in injury is a critical objective for communities, traffic engineers and the automotive industries. A proactive approach to reducing injuries relies on data science by understanding the data available documenting the severity of traffic accidents and creating machine learning models that can help predict the attributes that contribute to accident severity. In this project we will examine a data set for the city of Seattle and develop several machine learning model that predict accident severity.

Data

The data set used contains the city of Seattle collisions from 2006 to 2015 provided by SDOT Traffic Management Division, Traffic Records Group. The data set is available from Seattle Traffic Data Metadata and Seattle Traffic Dataset. Additional information is available in the 2015 Traffic Report by the City of Seattle.

We will use the data to create several machine learning models, including KNN, Decision Tree, SVM, Logistic Regression, and Random Forest that can help predict the attributes that contribute to accident severity in terms of Property Damage Only Collision and Injury Collision. We will need to critically evaluate the data and select attributes that will result on suitable accuracy for the models.

Methodology

The methodology used includes various steps including data analysis, attribute selection, data preparation and data balancing. Once the data set is balanced, several machine learning were used to create models including KNN, Decision Tree, SVM, Logistic Regression, and Random Forest and their accuracy estimated using various metrics.

Data Analysis and Attribute Selection

The first step is to analyze the data set, removing attributes that would not contribute to the model and in order to select the best attributes. Figure 1 shows the result of the Pair Plot analysis was used to examine the data.

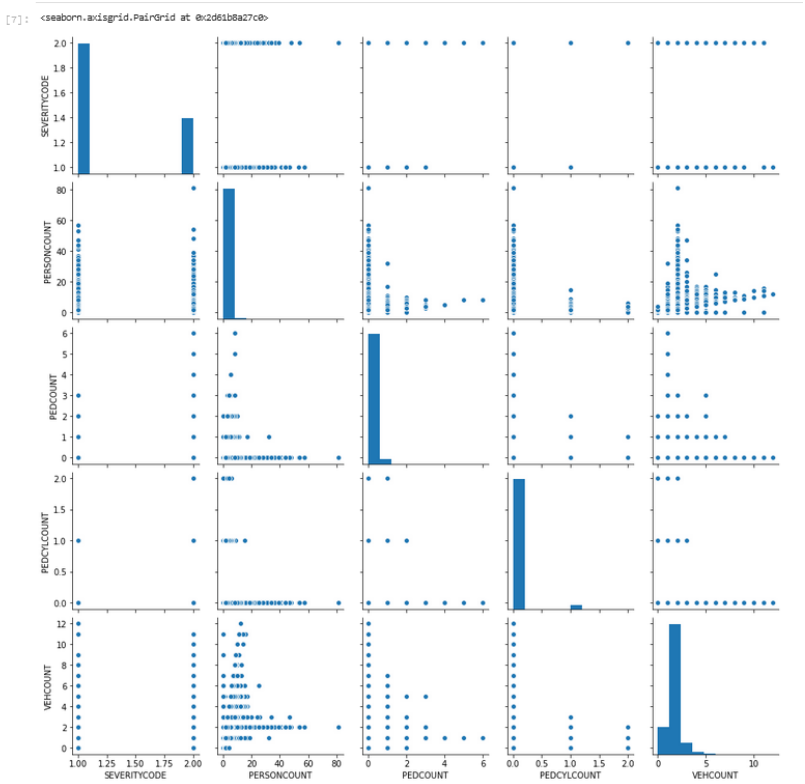


Figure 1 – Pair Plot Analysis

Further analysis was performed by examining individuals attribute. Figure 2 shows various histograms comparing attributes with collision severity. Note that clear versus raining weather, wet versus dry road condition, and day versus nigh light condition for have significant impact on severity and opposed to month where the impact is uniformly distributed in every month.

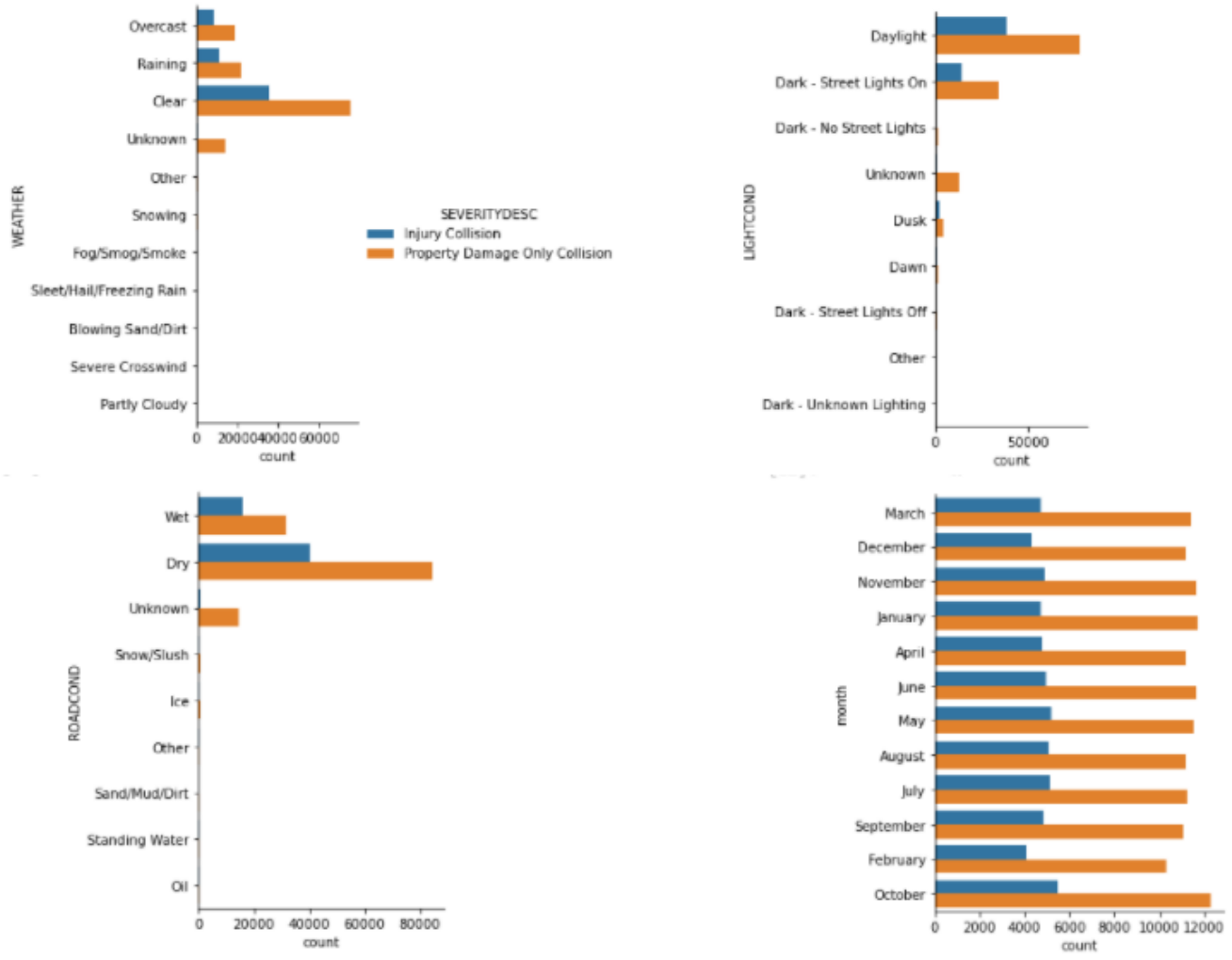


Figure 2 – Attribute Histogram Analysis

As a result of extensive analysis of all the attributes, weather, road condition and light condition were selected as the key attributes for the modeling due to their impact on severity. Further combining of these attributes will be performed to increase the model performance efficiency by combining multiple selections into categories. For example, all combinations of “Dark” will be combined in a single “Dark” attribute. However we note that the ration of injury and non-injury accidents seems to remain somewhat constant across different road, light and weather conditions. This will be further examines with a correlation matrix.

Data Preparation

The first step in preparing the data was to remove missing values. This removed 4,200 non injury entries and 1,136 injury entries. Figure 3 has the details of the resulting data frame.

```
[14]: df['SEVERITYCODE'].value_counts()

[14]: 1    136485
      2     58188
      Name: SEVERITYCODE, dtype: int64

[15]: df.dropna(inplace=True)

[16]: df['SEVERITYCODE'].value_counts()

[16]: 1    132285
      2     57052
      Name: SEVERITYCODE, dtype: int64
```

Figure 3 – NULL value removal

The datasets originally contains labels for several attributes that need to be expanded to additional categories with numerical values in order to be suitable to the models. The pd.get_dummies (df) approach was used to convert the object entries to numerical values and we further combined the weather into clear, rain and snow and light condition into light and dark. Figure 4 shows the resulting dataframe.

[18]:

	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	VEHCOUNT	clear	snow	rain	winter	spring	summer	fall	light	dark
0	2	2	0	2	1	0	0	0	1	0	0	1	0
1	1	2	0	2	0	0	1	1	0	0	0	0	1
2	1	4	0	3	1	0	0	0	0	0	1	1	0
3	1	3	0	3	1	0	0	0	1	0	0	1	0
4	2	2	0	2	0	0	1	1	0	0	0	1	0

Figure 4 – Data Frame after Attribute Selection

As a result of the analysis, weather, road condition and light condition were selected as the key attributes for the modeling. Figure 5 shows the correlation heatmap of the selected attributes. Correlation is useful to see if pairs of variables are related helping understand the data. Unfortunately the correlation matrix does not show strong correlation between any of the variables and the severity of the accident.

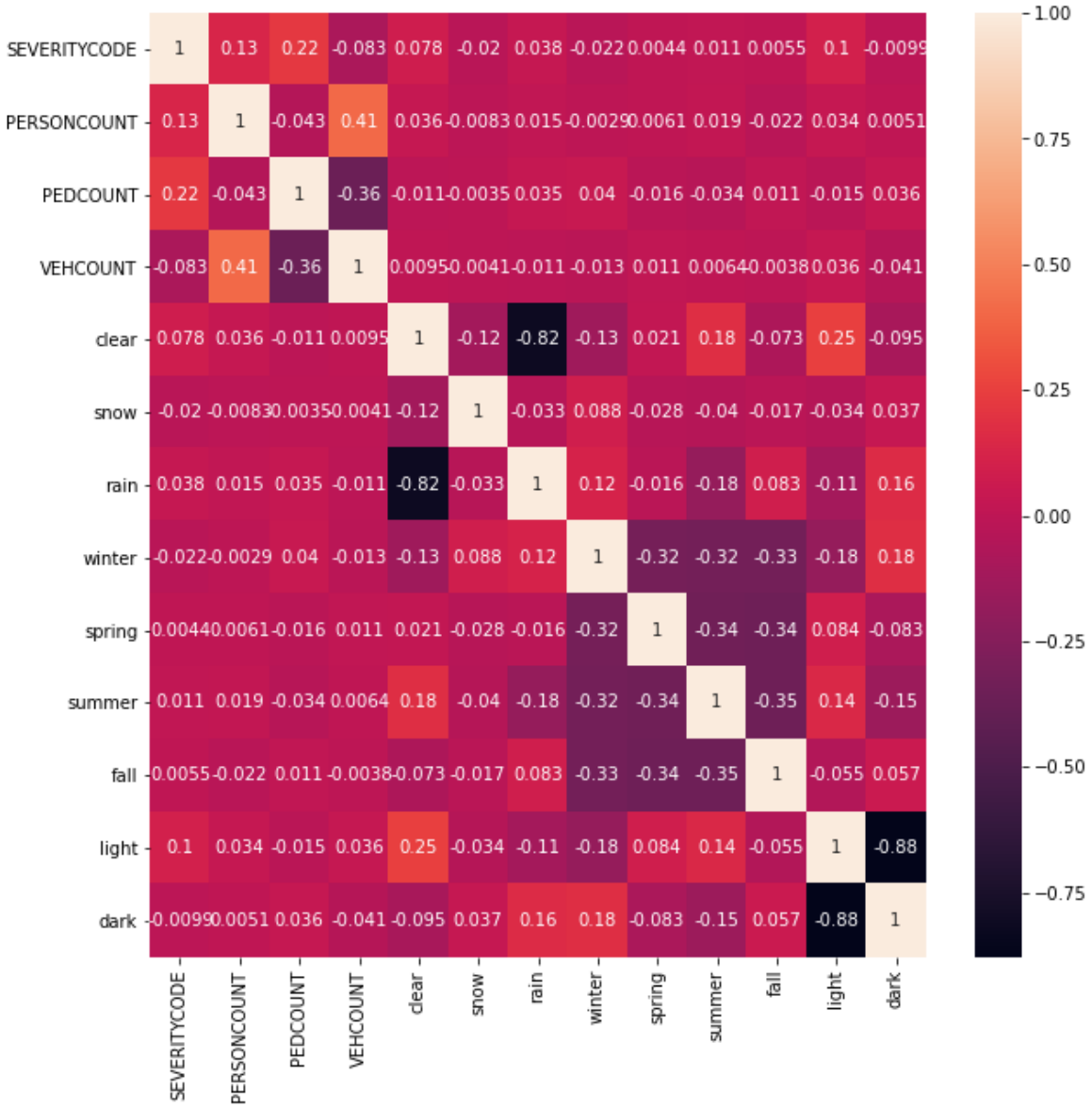


Figure 5 – Attribute Correlation Heatmap

Data Balancing and Transformation

The original dataset is unbalanced with 136,485 entries in the Property Damage Only Collision class and 58,188 entries in the Injury collision class. After removing Null values the dataset was balanced. Two approaches were tried, Downsample the majority class and Upsample minority class. After experimentation the decision was made to use Downsample the majority class with 57,052 for each class.

After balancing the data was transformed to make it look like standard normally distributed in a Gaussian manner with zero mean and unit variance. Figure 6 shows the resulting data.

```
[24]: X= preprocessing.StandardScaler().fit(X).transform(X)
      X[0:5]
```

```
[24]: array([[ 0.58216734, -0.06972237, -0.47600611,  0.69139792, -0.60686467],
             [-1.71771917, -0.06972237, -0.47600611, -1.4463451 , -0.60686467],
             [ 0.58216734, -0.06972237, -0.47600611,  0.69139792, -0.60686467],
             [ 0.58216734, -0.06972237, -0.47600611,  0.69139792, -0.60686467],
             [-1.71771917, 14.34259896, -0.47600611, -1.4463451 ,  1.64781384]])
```

Figure 6 – Transformed Data

Data Training and Testing

Finally the data was separated into training data and test data. Several splits were considered and the final value was chosen to be 25% for test data. Figure 7 shows the resulting training and test data.

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.25, random_state=7)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

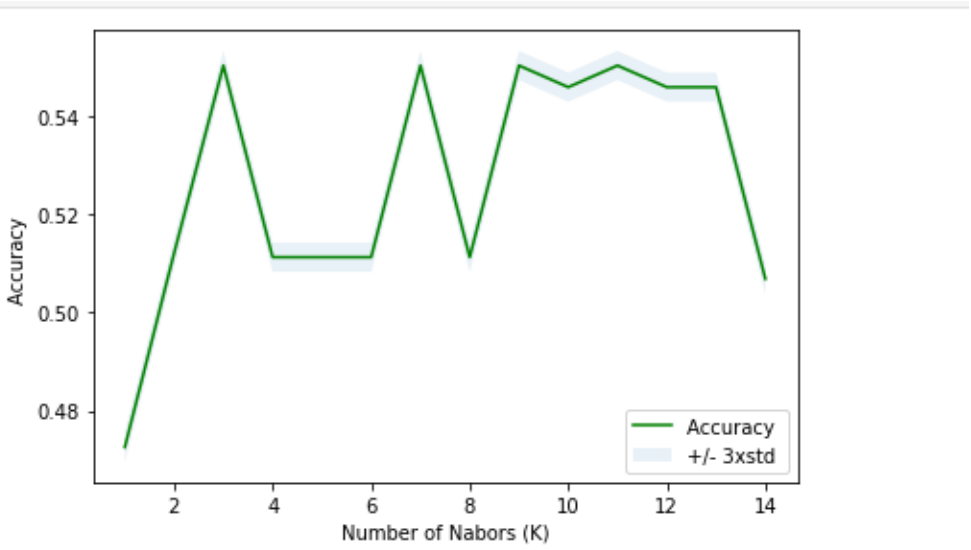
```
Train set: (85578, 5) (85578,)
Test set: (28526, 5) (28526,)
```

Figure 7 – Train and Test Data

Model Implementation

KNN Model

The K-nearest neighbors (KNN) is a supervised machine learning algorithm that can be used for classification tasks. KNN does not make assumptions about the underlying data. Figure 8 shows the result of the model versus multiple values of K. [1]




```
The best accuracy of KNN is  0.5503750964032812 , k= 3
KNN's Accuracy:  0.5503750964032812
```

Figure 8 – KNN Results

Decision Tree Model

Decision Tree is a supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Figure 9 shows the Decision Tree results. [1]

```
[1 1 2 2 1 1 1 1 1 2]
```

```
[32]: from sklearn import metrics
      import matplotlib.pyplot as plt

      Acc_dt = metrics.accuracy_score(y_test, predTree)
      print("DecisionTrees's Accuracy: ", Acc_dt)
```

```
DecisionTrees's Accuracy:  0.556895463787422
```

Figure 9 – Decision Tree Results

SVM Model

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection. SVM is effective in high dimensional spaces. Different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels. Figure 10 shows the Decision Tree results. [1]

```
from sklearn.metrics import f1_score
F1_svm =f1_score(y_test, yhat, average='weighted')
print("SVM F1 Score: ", F1_svm)
```

```
SVM F1 Score:  0.5301108876321696
```

Figure 10 – SVM Results

Logistic Regression Model

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables. Figure 11 shows the Logistic Regression results. [2]

```
LL_AC=metrics.accuracy_score(yhat,y_test)
print('The accuracy of the Logistic Regression is', LL_AC)
```

```
The accuracy of the Logistic Regression is 0.556895463787422
```

```
from sklearn.metrics import log_loss
LL_LR =log_loss(y_test, yhat_prob)
print("Logistic Log Loss score: ", LL_LR)
```

```
Logistic Log Loss score:  0.6691064961117598
```

Figure 11 – Logistic Regression Results

Random Forest Model

Random forests is a supervised learning algorithm used both for classification and regression. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. Figure 12 shows the Random forest results.[2]

```
[46]: Acc_RF = metrics.accuracy_score(y_test, forestpredict)
      print("Random Forest's Accuracy: ", Acc_RF)

      Random Forest's Accuracy:  0.556895463787422
```

Figure 12 – Random Forest Results

Results

Figure 13 shows a summary of the model results.

```
[56]: print("KNN's Accuracy: ", Acc_knn)
      print("DecisionTrees's Accuracy: ", Acc_dt)
      print("SVM F1 Score: ", F1_svm)
      print('Logistic Regression Accuracy', LL_AC)
      print("Random Forest's Accuracy: ", Acc_RF)
      print("Logistic Log Loss score: ", LL_LR)

      KNN's Accuracy:  0.5503750964032812
      DecisionTrees's Accuracy:  0.556895463787422
      SVM F1 Score:  0.5301108876321696
      Logistic Regression Accuracy 0.556895463787422
      Random Forest's Accuracy:  0.556895463787422
      Logistic Log Loss score:  0.6691064961117598

[ ]:
```

Figure 13 – Model Results

Discussion

One would expect that weather, light condition and road condition would have an impact on severity of accidents. However, the accuracy of the models we generated is less that we would want. That was expected based on what we saw in the correlation analysis.

In terms of comparing the different models, all models show similar accuracy around 0.55 except for Logistic regression.

Conclusion

In this project we used a data set for the city of Seattle to develop several machine learning model that predict accident severity. Accident severity is defined as the understanding if traffic accidents result in injury or property damage. Reducing the number of accidents that result in injury is a critical objective for communities, traffic engineers and the automotive industries. The methodology used included data analysis, attribute selection, data preparation and data balancing. KNN, Decision Tree, SVM, Logistic Regression, and Random Forest were developed to predict accident severity and their accuracy estimated using various metrics. The accuracy of the models we generated is less that we would want and the correlation between the available attributes and accident severity was not strong. In terms of comparing the different models, all models show similar accuracy around 0.55 except for Logistic regression that shows a better showing in terms of log loss accuracy.

References

[1] https://scikit-learn.org/stable/modules/svm.html

[2] https://www.datacamp.com/