COURSERA CAPSTONE PROJECT

Prediction on Severity of An Accident

Name: Chenyang Liu

Complete Date: 09/16/2020

ABSTRACT

1. INTRODUCTION

Driving is one of the most common part of life for many people. Although it is hard, if not impossible, to predict the happening of an accident, it is easier to predict the severity of an accident. Therefore, an accurate prediction would help drivers to make a decision when they know the level of severity of the accident happened in front of them. Having an accurate prediction is very important for a GPS service provider, such as Google Map, to inform the driver and to choose a suitable route for the driver based on the estimation of the severity.

To meet this demand, this project focuses on the prediction on the severity of an accident based on the data provided by SPD. To achieve a good result, the project chooses several advanced machine learning techniques to build the model so that the model will help us to answer the business question: based on the information we have, whether the accident is an injury (with severity code as 2) or a property damage (with severity code as 1)?

2. METHODOLOGY

To answer this classification question, there are three classification models will be considered in this project, which are Logistic Regression, K Nearest Neighbors, and Decision Tree. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In pattern recognition, the K-Nearest Neighbors algorithm (k-NN) is a non-parametric method proposed by Thomas Cover used for classification and

regression. In both cases, the input consists of the k closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, normalizing the training data can improve its accuracy dramatically. A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Since the data provided by SPD is an imbalanced data, to solve the problem, there are two sampling methods will be used: Upsampling and Downsampling. To keep as much information as we can, we will use upsampling so that we won't loss and information from the data. While considering the computability of the computer for some machine learning models, we will also use downsampling with Principal Component Analysis to reduce the dimensions and the number of records.

To evaluate the result, 3-fold cross validation will be used to give a relatively fair result. Besides, since predicting an accurate result for both categories are important for us, F1-score will be chosen as the key metric to evaluate the performance of each model. Moreover, to find the best parameters for some models, Grid Search will be used to get the most suitable parameter among the choices provided.

3. DATA EXPLORATION AND DATA PREPARATION

Originally, this data set includes 194673 records and 38 variables, which contains

categorical variables, such as ADDRTYOE, and numerical variables, such as ROADCOND. There are some problems in this data set which need to be fixed. These problems are: missing values, duplicate variables in different formats, conflict formats in the same variable and imbalanced data. Besides, to apply the data to the models, other steps like data transformation and resampling methods are also necessary. The details of each problem and the corresponding way to fix the problem are followed.

1) ***Removing Variables with a Large Number of Missing Value***

   As shown in ***Appendix 1***, there are 19 variables (out of 38) with missing values. There are 7 variables have a large number of missing values (at least 40%) out of the total 194673 records. These variables are: INTKEY, EXCEPTRSNCODE, EXCEPTRSNDESC, INATTENTIONIND, PEDROWNOTGRNT, PEEDING, and SDOTCOLNUM. Since variables with a large number of missing values can't provide us enough information, they will be removed from the data for the further use. As a result, all the remaining variables have missing values less than 3.5%.

2) ***Excluding Variables with Same Values in Different Formats***

   After reviewing the data set, there are 5 pairs of variables with the same values but in different format: SEVERITYCODE & SEVERITYDESC, INCDATE & INCDTTM, SDOT_COLCODE & SDOT_COLDESC, and ST_COLCODE & ST_COLDESC. Hence, only one variable of each pair is kept in the data set. Besides, SEVERITYCODE has existed for twice in this data set, so one of them will be removed.

3) *Deleting Unnecessary Variables*

Since the value of some variables are not relevant to the value of the severity that we want to predict, five variables are excluded: X, Y, LOCATION, SDOT_COLCODE, INCDATE and ST_COLCODE. Other variables, such as the dummy variable created for ADDRTYOE=Alley, are deleted based on t-tests, which will be explained later in the data analysis part.

4) *Change Values to the Same Format*

As shown below, the values of UNDERINFL used two formats: 1/0 and Y/N. According to the understanding of data, these two formats are indicating the same values. These values will be changed to use the same format - 1 for Y and 0 for N.

UNDERINFL	
0	80394
1	3995
N	100274
Y	5126

Picture 1 - Conflict Formats for Variable UNDERINFL

5) *Data Transformation*

To apply the values from the data to the model, there are some necessary transformation steps needed to be done before training the models. Transforming Binary Variable to 1/0 for variables - SEVERITYCODE, STATUS, and HITPARKEDCAR, is necessary since models like logistic regression can only compute numeric values. Similarly, categorical variables - ADDRTYPE, COLLISIONTYPE, JUNCTIONTYPE, WEATHER, ROADCOND, LIGHTCOND, that have more than two categories should be transformed to dummy variables from n categories to n dummy variables.

6) Removing Missing Values

As the result shown in the ***Appendix 1***, there are some missing values would still exist in the data set even we have removed those variables with a large proportion of missing data. This would impact the result that should be removed as well.

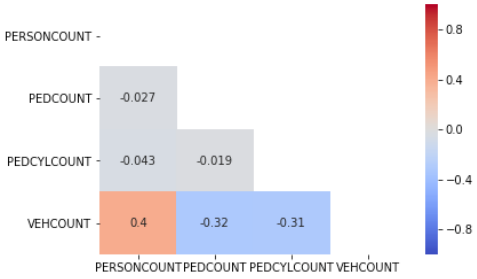
7) Imbalanced Data

In this dataset, there are 132630 records with severity code as 1 while there are only 57159 records with severity code as 2, about 30% of the total number. When the distribution of examples across the known classes is biased, it would be an unequal distribution of classes, which would impact the performance of the model. To keep as much information as possible, upsampling is used for logistic regression and decision tree. After upsampling, there are 265260 records used to train the models. Due to computability, downsampling is used for k-nearest neighbors. After downsampling, there are 114318 records used to train the models.

4. DATA ANALYSIS

1) Correlation for Numeric Data

As we can see from the correlation heat map below, the correlations of the numeric variables are very low, which means, there is no collinearity problem here.

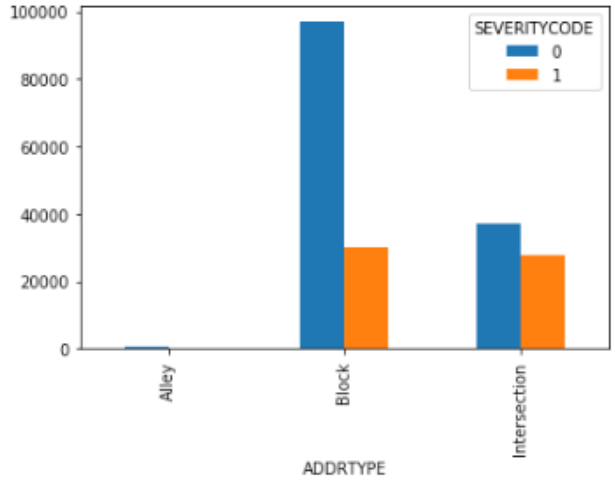


Picture 2 - Correlation Heat Map

2) *Bar Charts and ANOVA Tests for Categorical Data*

As shown in the bar chart and the result of the ANOVA tests, we can find that, in terms of severity, the impact of different address types has a significant difference.

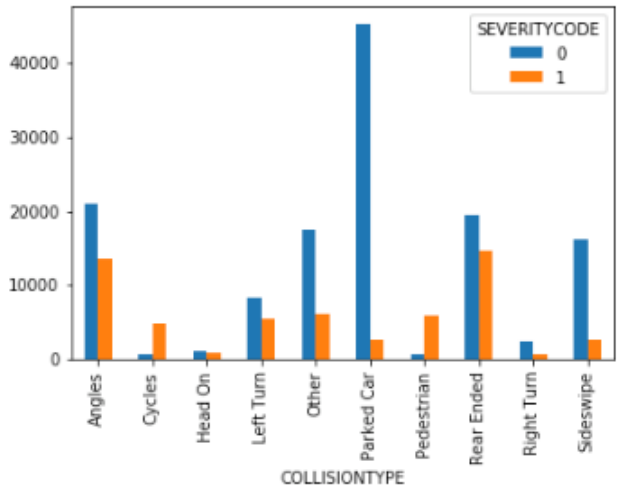
```
F_onewayResult(statistic=3530.8078377494403, pvalue=0.0)
```



Picture 3 - Number of Records of Severity Code 1&2 by ADDTYPE

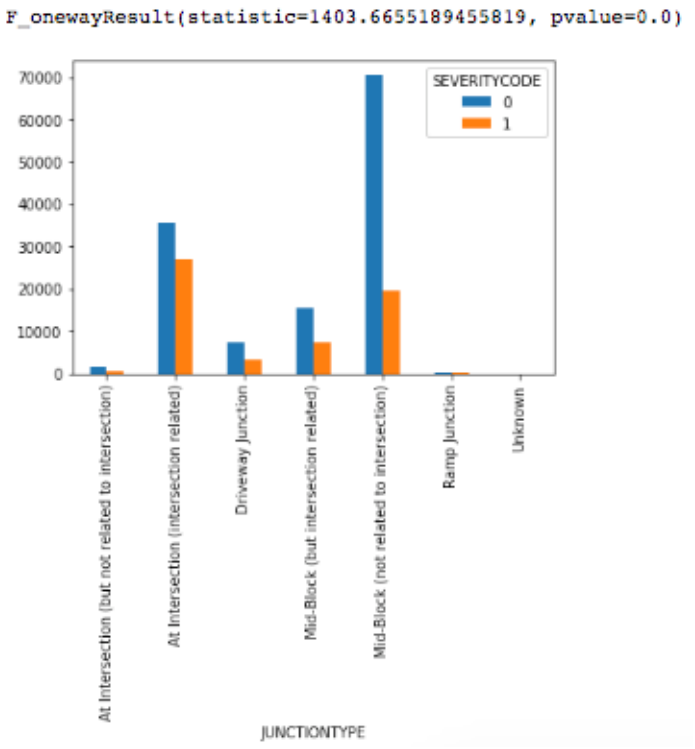
As shown in the bar chart and the result of the ANOVA tests, we can find that, in terms of severity, the impact of different collision types has a significant difference.

```
F_onewayResult(statistic=5369.001306448202, pvalue=0.0)
```



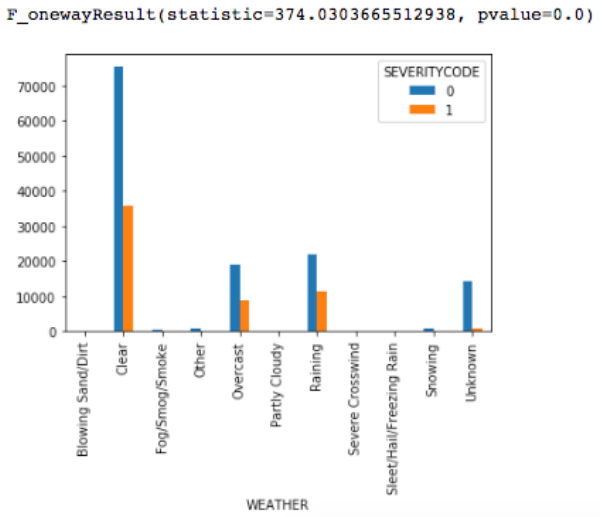
Picture 4 - Number of Records of Severity Code 1&2 by COLLISIONTYOE

As shown in the bar chart and the result of the ANOVA tests, we can find that, the impact of different categories of junction has a significant difference.



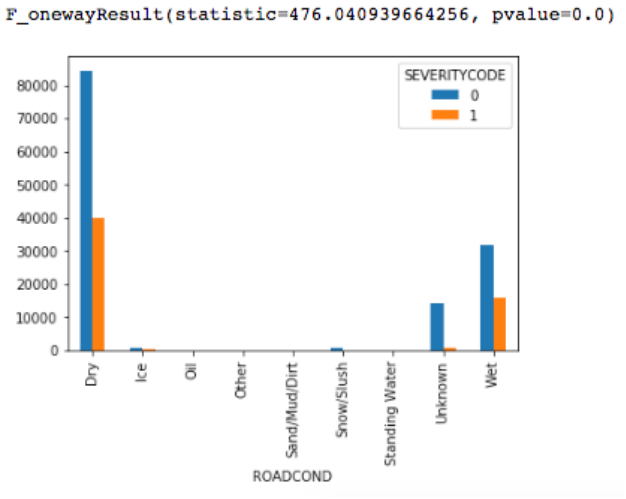
Picture 5 - Number of Records of Severity Code 1&2 by JUNCTIONTYPE

As shown in the bar chart and the result of the ANOVA tests, we can find that, in terms of severity, the impact of different weather types has a significant difference.



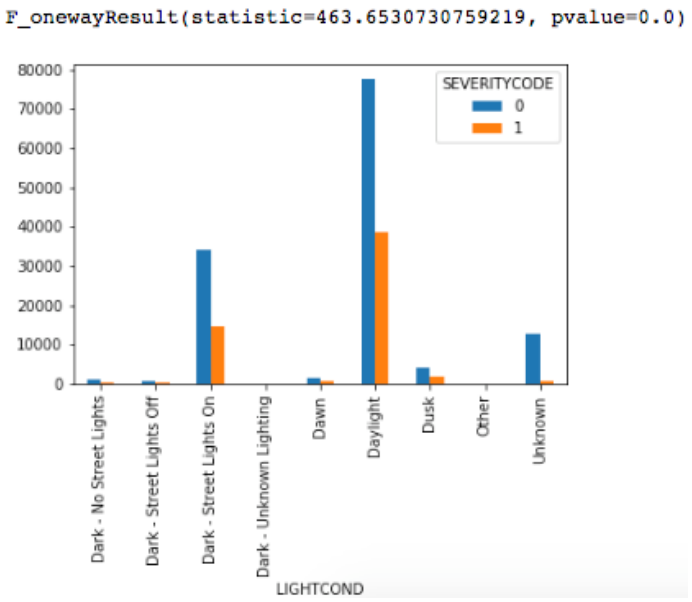
Picture 6 - Number of Records of Severity Code 1&2 by WEATHER

As shown in the bar chart and the result of the ANOVA tests, we can find that, in terms of severity, the impact of different conditions of the road during the collision has a significant difference.



Picture 7 - Number of Records of Severity Code 1&2 by ROADCOND

As shown in the bar chart and the result of the ANOVA tests, we can find that, in terms of severity, the impact of different conditions of the light during the collision has a significant difference.



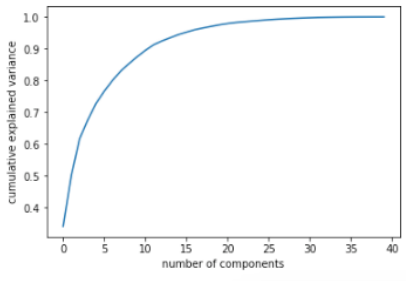
Picture 8 - Number of Records of Severity Code 1&2 by LIGHTCOND

3) *T-test*

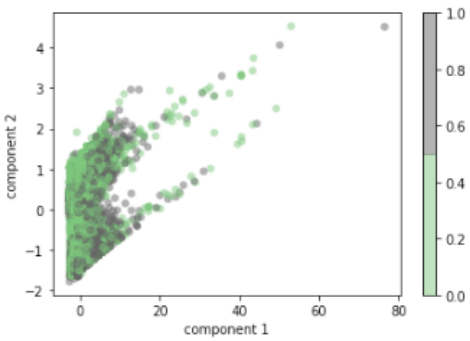
As an example of the result of t-test shown in ***Appendix 2***, we can notice that no matter which status it is recorded, statistically speaking, there is no significant difference on the value of severity code between the two groups of status. Similarity, we can notice that other variables, for example, left turn collision type, ramp junction and dawn weather.

4) *Principal Component Analysis*

As the result shown in ***Appendix 3***, there are three principal components that have a relatively high explained variance, which are chosen to use when building the models. However, we can also notice that, as shown in the plot below, with only 3 principal components, the cumulative explained variance is relatively low - around 70% of total. This shortage is also reflected in ***Picture 10***.



Picture 9 - Number of Principal Components vs. Cumulative Explained Variance



Picture 10 - Number of Principal Components vs. Cumulative Explained Variance

5. MODEL EVALUATION

1) Linear Regression

As the results of the 3-fold cross validation shown in ***Appendix 4***, with upsampling data and after using grid search and finding the best parameter value for C as 0.001, the average F1-score we got when using the testing data is 0.73. The three F1-scores we got when using the training data are: 0.724, 0.724, and 0.727. The F1-score we got when using the testing data is 0.702. Besides, the time to complete the model building and employment is relatively short.

2) K Nearest Neighbor

As the results of the 3-fold cross validation shown in ***Appendix 5***, with downsampling data and after using the best value of k as 11, the average F1-score we got when using the testing data is 0.68. The three F1-scores we got when using the training data are: 0.682, 0.677, and 0.683. The F1-score we got when using the testing data is 0.686. Besides, the time to complete the model building and employment is the longest.

3) Decision Tree

As the results of the 3-fold cross validation shown in ***Appendix 6***, with upsampling data and after using the best maximum of depth as 9, the average F1-score we got when using the testing data is 0.73. The three F1-scores we got when using the training data are: 0.724, 0.725, and 0.727. The F1-score we got when using the testing data is 0.705. Besides, the time to complete the model building and employment is the shortest.

6. CONCLUSION

Based on the values of F1-score, we can conclude that Decision Tree has a slightly better performance than Logistic Regression and a much better performance than K-Nearest Neighbor. As explained previously, this bad performance may be related to the variance explained by the principal components. Moreover, the time consuming of Decision Tree is also shorter than the other two models. Therefore, Decision Tree is the best model for the prediction out of these models.

7. RECOMMENDATION

Since Decision Tree has the best performance in terms of both the accuracy (evaluated by F1-score) and the computing time, it would be the recommended model. Besides, since all of the three models have a relatively good performance, reflected by the F1-scores (around 0.7), which is better than random guessing (which is 0.5). The Ensemble Learning should also be recommended for future usage, which uses multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. For example, if both logistic regression and decision tree predict 1 while k-nearest neighbors predicts 0, the prediction with ensemble learning would be 1 because the majority algorithms (2 out of 3) suggests 1 as the class.

APPENDICES

Appendix 1 - Missing Values (Part)

```
df.isnull().sum()
```

```
SEVERITYCODE           0
X                   5334
Y                   5334
OBJECTID               0
INCKEY                 0
COLDETKEY              0
REPORTNO               0
STATUS                 0
ADDRTYPE            1926
INTKEY            129603
LOCATION            2677
EXCEPTRSNCODE     109862
EXCEPTRSNDESC     189035
SEVERITYCODE.1         0
SEVERITYDESC           0
COLLISIONTYPE       4904
PERSONCOUNT            0
```

Appendix 2 - Result of T-test

```
results = rp.ttest(group1= df_trans['SEVERITYCODE'][df_trans.iloc[:,1] == 1], group1_name= "1",
                   group2= df_trans['SEVERITYCODE'][df_trans.iloc[:,1] == 0], group2_name= "0")
results[1].iloc[3,1]
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/scipy/stats/_distn_infrastructure.py:1920:
multiply
  lower_bound = self.a * scale + loc
/opt/conda/envs/Python36/lib/python3.6/site-packages/scipy/stats/_distn_infrastructure.py:1921:
multiply
  upper_bound = self.b * scale + loc
```

```
0.2555
```

Appendix 3 - Result of Principal Component Analysis

```
pca = PCA(n_components=3)
pca.fit(df_trans[df_trans.columns[1:42]])
print(pca.explained_variance_)
```

```
[1.91661579 0.89834652 0.64487807]
```

```
projected = pca.fit_transform(df_trans[df_trans.columns[1:42]])
print(df_trans[df_trans.columns[1:42]].shape)
print(projected.shape)
```

```
(189789, 40)
(189789, 3)
```

Appendix 4 - Linear Regression

```
lr = LogisticRegression(C=0.001,solver='liblinear').fit(X_train,y_train)
f1_score = cross_val_score(lr, X_train, y_train, cv=3, scoring = make_scorer(f1_score))
f1_score
```

```
array([0.72435956, 0.72452267, 0.7268931 ])
```

```
print("Logistic Regression Average Training F1-score: %0.2f (+/- %0.2f)" % (f1_score.mean()
```

```
Logistic Regression Average Training F1-score: 0.73 (+/- 0.00)
```

```
from sklearn.metrics import f1_score
y_LR = lr.predict(X_test)

print("Logistic Regression's Test F1-score ",f1_score(y_test, y_LR, average='weighted') )
```

```
Logistic Regression's Test F1-score  0.7024750994501149
```

Appendix 5 - K-Nearest Neighbors

```
from sklearn import metrics
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
from sklearn.model_selection import cross_val_score

k = mean_acc.argmax()+1
neigh = KNeighborsClassifier(n_neighbors=n).fit(X_train_pca,y_train_pca)

f1_score = cross_val_score(neigh, X_train_pca, y_train_pca, cv=3, scoring = make_scorer(f1_s
f1_score
```

```
array([0.6824057 , 0.67662548, 0.6825614 ])
```

```
print("K Nearest Neighbor Average Training F1-score: %0.2f (+/- %0.2f)" % (f1_score.mean(),
```

```
K Nearest Neighbor Average Training F1-score: 0.68 (+/- 0.01)
```

```
from sklearn.metrics import f1_score

y_KNN = neigh.predict(X_test_pca)

print("KNN's Test set F1-score ",f1_score(y_test_pca, y_KNN, average='weighted') )
```

```
KNN's Test set F1-score  0.6860351922013241
```

Appendix 6 - Decision Tree

```
Tree = DecisionTreeClassifier(criterion="entropy", max_depth = 9)
Tree.fit(X_train,y_train)

f1_score = cross_val_score(lr, X_train, y_train, cv=3, scoring = make_scorer(f1_score))
f1_score
```

```
array([0.72435956, 0.72452267, 0.7268931 ])
```

```
print("Decision Tree Average Training F1-score: %0.2f (+/- %0.2f)" % (f1_score.mean(), f
```

```
Decision Tree Average Training F1-score: 0.73 (+/- 0.00)
```

```
y_DT = tree.predict(X_test)

print("Decision Tree's Test F1-score ",f1_score(y_test, y_DT, average='weighted') )
```

```
Decision Tree's Test F1-score  0.7054549177447814
```