# Seattle Traffic Accident Severity Prediction

Guankai Zhai

August 20, 2020

## I. Introduction

1. Background

   According to the US Department of Motor Vehicles (DMV), about 6 million car accidents happen each year in the US. On an average day, over 90 people die due to traffic accidents, leaving hundreds of families behind them. Despite such disturbing facts, people's awareness of traffic safety is still insufficient. According to a survey, only 1 in 7 people will wire a seatbelt when he or she sits in a car, while seatbelts could reduce the risk of death by over 45%. Therefore, it is advantageous for people to realize the risk of traffic accidents and thus practice safer driving habits.
2. Problem

   This project aims to determine which factors affect the severity of car accidents and then build a model that could predict the severity of car accidents based on the determined factors.
3. Interest

   The information from this project would be beneficial to many parties including the DMV, police departments around the country, and most importantly, drivers and their passengers. The project would shed light on what combination of conditions are conducive to fatal car accidents, how the severity of car accidents depend on these conditions, etc. After knowing the relevant information for a given road on a given day, the police department could predict the possibility of fatal car accidents and therefore warn drivers to practice safe driving styles.

## II. Data

1. Sources

   The dataset that this project will use is provided by the International Business Machines (IBM) Corporation here.
2. Content

   The dataset has 194672 instances of traffic accidents that occurred in Seattle. It labels each collision as either of severity code 1 or 2, representing prop damage or injury. Complete metadata of this dataset could be found here.

## III. Methodology

1. Preprocessing

   Since the raw dataset contains far more instances of SEVERITYCODE 1 than of 2 (around 2.34 : 1), dataframe.sample() method was used to sample from SEVERITYCODE==1 instances an amount equal to the number of SEVERITYCODE==2 instances. In this way, the bias from unbalanced data was eliminated.

2. Exploratory Data Analysis (EDA)

   Although 37 features are provided in the dataset, certainly not all of them are directly correlated with our target. Thus, how can we select those that do? The Dataframe.groupby(feature_name)[].value_counts() method is used on each column to determine the ones correlated with accident severity. The INCDATE object was also converted to day-of-week so that we could discover whether people driving on a certain day of the week are more likely to experience a fatal accident. After the above steps, we have determined which features are relevant to the SEVERITYCODE and cold proceed to the next step.
3. One Hot Encoding
   Since many machine learning models could only take in inputs that are numerical, the categorical features in this data set must be converted. To achieve this, the Dataframe[feature_name].replace() method was utilized, and the dataframe.dtype attribute was also used to validate the post-processing dataset.
4. Feature Selection and Normalization
   After the above steps, we are finally ready to create the feature set. 14 columns of data were selected in the feature dataset, including weather, road condition, lighting condition, etc. The the dropna() method was called to drop any rows with NaN data, and the preprocessing.StandardScalar().fit().transform() function was also used to normalize the feature set so as to prevent any bias caused by the different scales of different features. In this way, we are ready for the model training.
5. Model Training and Testing
   I used the train_test_split() method to split the datasets into X_train, y_train, X_test, y_test. Then four machine learning classification models were imported, including K Nearest Neighbors, Decision Tree, SVM, and Logistic Regression. They were each trained with X_train and y_train and then tested with X_test and y_test to obtain their performance.

IV. Results and Discussions

Out of the four models, Decision Tree achieved the highest accuracy score of 0.702, although the differences between them were small. The Decision Tree model was also the one that took the least computing time, so its should be recommended for future deployment.

I learned an important lesson from this project: preparing the data instead of training and testing the models is the most time-consuming part. Although I learned this fact a few weeks ago, I didn’t fully realize this concept until now. The process of normalizing and EDA was really the most important part of this project and is also the foundation of my subsequent model training.

For the future, I suggest that the parameters of these models be further tuned so that better performance could be obtained. Also, after the model is deployed, newly-generated data should be further fed into the model so that its performance could be updated.