Coursera Capstone:
Collision Severity Prediction

IBM Data Science Professional Certificate

Kaming Yip

September 1, 2020

Table of Contents

1. Introduction……………………………………………………………………………………3
2. Data Description…………………………………………………………………………………4
3. Methodology…………………………………………………………………………………. 5
   3.1 Data Preprocessing…………………………………………………………………...5
      i. Missing Values……………………………………………………………………. 5
      ii. Feature Engineering………………………………………………………………7
   3.2 Exploratory Data Analysis…………………………………………………………8
      i. Maps of Accidents……………………………………………………………………8
      ii. Collisions in Different Types………………………………………………………10
      iii. Collision Time………………………………………………………………...11
      iv. Collision Size…………………………………………………………………12
   3.3 Data Wrangling……………………………………………………………………...14
   3.4 Modeling………………………………………………………………………….... 14
4. Results………………………………………………………………………………………. 17
5. Conclusion and Future Work……………………………………………………………19
References………………………………………………………………………………… 21

1. Introduction

According to the annual report released by the National Highway Traffic Safety Administration (NHTSA) (2020), motor vehicle crashes, as the number one safety problem in American transportation, count for 94 percent of transportation death and 99 percent of transportation injury. The lifetime economic cost of these crashes is over $150 billion annually, which is quite a huge amount from taxes. From the previous statistic, the public pays 13 percent of the cost of injuries treated in an emergency department; 26 percent of the cost of injuries requiring hospitalization; and 48 percent of the cost of injuries treated in a rehabilitation hospital. It is predicted that, with yearly increases in travel and no improvement over our current safety performance, fatalities and injuries could increase by 50 percent by 2020.

The challenge facing us is how we can prevent from more car accidents and improve highway safety by learning from past experience. In a scenario when a driver is heading to another city for work or visit some friends, if there was an accident happened on the way, the traffic would be terrible and everyone would be stuck in that slow lane. Cars already stuck in traffic have no choice but to wait no matter how long it takes to pass the accident scene; however, incoming cars or drivers would have no idea what happens at the front of the lane but to also join the lane. This could take couples of hours to completely clean the field and go back to normal.

However, if there is something in place that could warn the drivers, given the weather and the road conditions, the possibility of getting into a car accident and how severe it would be, it would be much better for the drivers to drive more carefully or even change their travel if possible. It can be imagined that, with these useful prompts, tens of thousands of drivers can be saved from the unexpected or undesirable accidents and better control their travel plans.

In this paper, we will be working on a machine learning model, which will predict the severity of an accident based on the attributes, such as weather, road, and light conditions, and provide suggestions for drivers based on the environment, based on a dataset provided by the IBM Data Science instructors.

2. Data Description

The dataset we will use in this paper is the shared data originally provided by Seattle Department of Transportation (SDOT) Traffic Management Division, Traffic Records Group (2020), and modified to particularly meet the project criteria. The entire dataset has 194,673 observations (rows) and 38 attributes (columns). Each row is a collision happened in Seattle recorded by Seattle Police Department (SPD) from January 2004 through May 2020. More detailed information of the dataset can be found in the metadata (Seattle Department of Transportation [SDOT], n.d.).

The labeled data (i.e. the output variable) of this project is the column named *SEVERITYCODE*, which corresponds to the severity of each collision. The description of each severity code is listed in the Table 1, as well as the number of appearance in the dataset.

Table 1: Description and Frequency of Severity Code.

Severity Code	Description	Count
3	Fatality	NA
2b	Serious Injury	NA
2	Injury	136,485
1	Prop Damage	58,188
0	Unknown	NA

It is worth noting that the labels are unbalanced, with 136,485 cases in a severity of 2 and 58,188 cases in a severity of 1. Balancing the data is essential and critical, otherwise, the model created by this original dataset will be a biased one. Remember, *garbage in, garbage out*. This problem will be handled in the Methodology section.

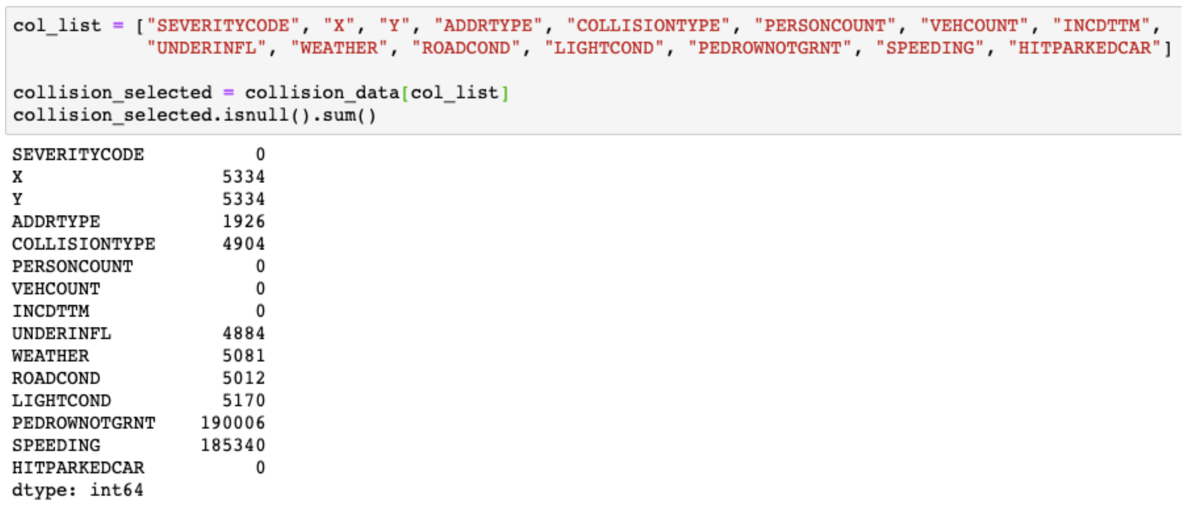
Moreover, the rest of the columns in the dataset are the attributes that some of them will be used to train the predictive model and try to predict the different accidents' severity. Among the columns, specific attention will be paid to those which have high correlation with the labeled data, for example location, weather condition, road condition, light condition, junction type, car speeding, number of people and vehicles involved in, and so forth. Feature engineering will also be applied in this project to improve the predictability of the model. Details on what and why features will be selected will be discussed in the Methodology section.

3. Methodology

3.1 Data Preprocessing

With all the features the dataset originally has, only the features that we will use in the following data wrangling, visualization, and analysis will be selected and remained, while the other features, which we assume will certainly not be used in this project, will be subtracted from the dataset.

Figure 1: Feature Selection



```
col_list = ["SEVERITYCODE", "X", "Y", "ADDRTYPE", "COLLISIONTYPE", "PERSONCOUNT", "VEHCOUNT", "INCDTTM",
            "UNDERINFL", "WEATHER", "ROADCOND", "LIGHTCOND", "PEDROWNOTGRNT", "SPEEDING", "HITPARKEDCAR"]

collision_selected = collision_data[col_list]
collision_selected.isnull().sum()
```

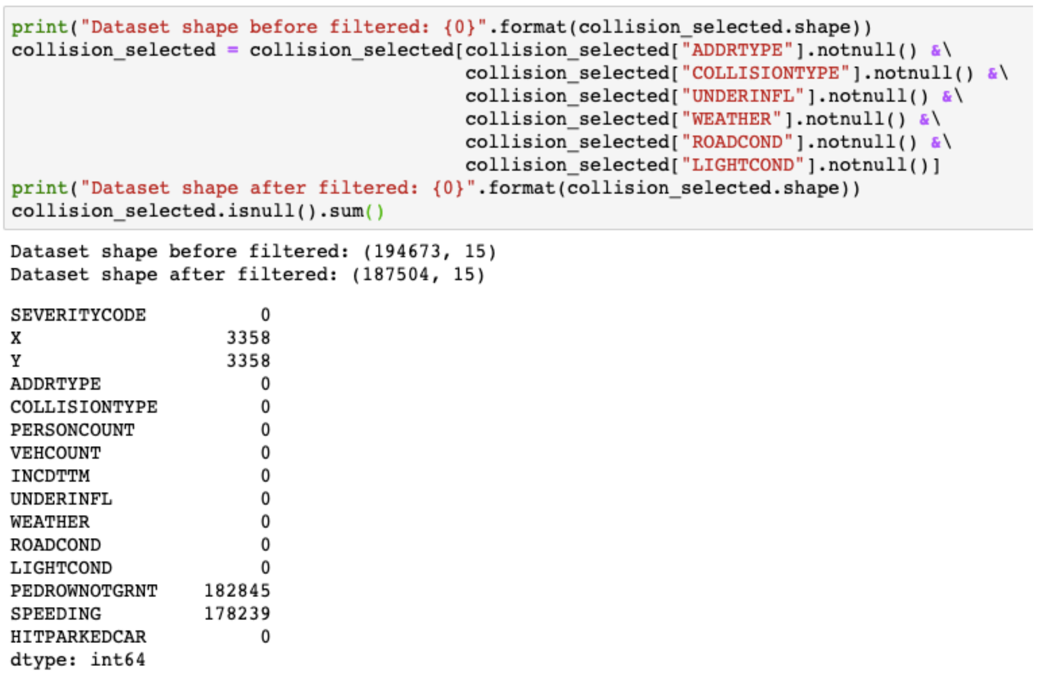
```
SEVERITYCODE          0
X                  5334
Y                  5334
ADDRTYPE           1926
COLLISIONTYPE      4904
PERSONCOUNT           0
VEHCOUNT              0
INCDTTM               0
UNDERINFL          4884
WEATHER            5081
ROADCOND           5012
LIGHTCOND          5170
PEDROWNOTGRNT    190006
SPEEDING         185340
HITPARKEDCAR          0
dtype: int64
```

i. Missing Values

However, the raw dataset is not perfect, containing missing values in some of the important features, for example *ADDRTYPE*, *COLLISIONTYPE*, and other condition description columns, that we would like to keep and apply in the model. The reason for the missing values might be human errors, interruptions in the data flow, privacy concerns, and so on. Whatever is the reason, in this project, we will implement two methods to handle this problem.

The first method is to remove rows that contain missing values for particular columns. Although this method would increase the risk of losing certain information contained in the records for the other variables, it is still better than removing the entire column to ensure the data analysis process. Plus, we have a relatively large dataset that cleaning off a minor part of "imperfect" records is still acceptable.

Figure 2: Handling Missing Values



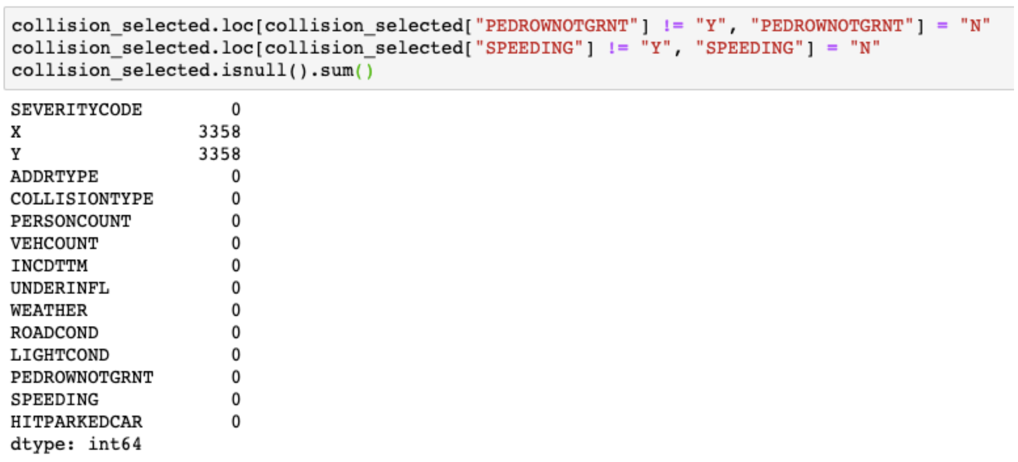
```
print("Dataset shape before filtered: {0}".format(collision_selected.shape))
collision_selected = collision_selected[collision_selected["ADDRTYPE"].notnull() &\
                                        collision_selected["COLLISIONTYPE"].notnull() &\
                                        collision_selected["UNDERINFL"].notnull() &\
                                        collision_selected["WEATHER"].notnull() &\
                                        collision_selected["ROADCOND"].notnull() &\
                                        collision_selected["LIGHTCOND"].notnull()]
print("Dataset shape after filtered: {0}".format(collision_selected.shape))
collision_selected.isnull().sum()
```

```
Dataset shape before filtered: (194673, 15)
Dataset shape after filtered: (187504, 15)

SEVERITYCODE          0
X                  3358
Y                  3358
ADDRTYPE              0
COLLISIONTYPE         0
PERSONCOUNT           0
VEHCOUNT              0
INCDTTM               0
UNDERINFL             0
WEATHER               0
ROADCOND              0
LIGHTCOND             0
PEDROWNOTGRNT    182845
SPEEDING         178239
HITPARKEDCAR          0
dtype: int64
```

However, in other cases, dropping rows or the entire column with missing values will lead to a large loss of data. In that cases, an alternative to omitting records with missing values we use in this project is to replace the missing value with an imputed value, called imputation. Imputation is applied in 2 columns in this project, named *PEDROWNOTGRNT* and *SPEEDING*. The reason why we are able to impute values into the missing value cells is that, after checking the existing values in those columns, we can find out that they both are columns which should contain Boolean values but only with a single value in the raw dataset. In this case, we can assume that those empty cells should be imputed with the corresponding value.

To be more specific, let's take column *SPEEDING* as an example. As described, the column should have *Y* or *N* to indicate whether or not speeding was a factor in the collision; however, the raw data contains 9,333 *Y*'s out of the 187,504 rows, leaving the remaining 178,171 cells empty, instead. Thus, in this case, we can reasonably impute *N*'s to those empty cells suggesting that those collisions were not due to inattention. Similar operations can be made to the other two aforementioned features.

Figure 3: Imputation Missing Values



```
collision_selected.loc[collision_selected["PEDROWNOTGRNT"] != "Y", "PEDROWNOTGRNT"] = "N"
collision_selected.loc[collision_selected["SPEEDING"] != "Y", "SPEEDING"] = "N"
collision_selected.isnull().sum()
```

```
SEVERITYCODE        0
X                3358
Y                3358
ADDRTYPE            0
COLLISIONTYPE       0
PERSONCOUNT         0
VEHCOUNT            0
INCDTTM             0
UNDERINFL           0
WEATHER             0
ROADCOND            0
LIGHTCOND           0
PEDROWNOTGRNT       0
SPEEDING            0
HITPARKEDCAR        0
dtype: int64
```

So far, we have finished handling the missing values in the selected dataset. Since column *X* and *Y* will only be used in data visualization, missing values in these two columns will not be a matter in this case.

ii. Feature Engineering

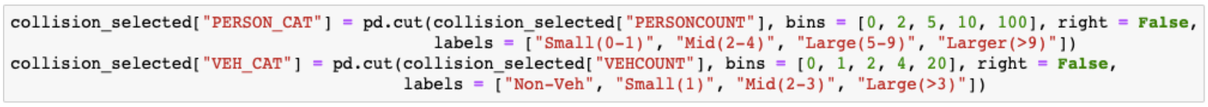
In order to better explore the raw dataset, feature engineering is utilized to transform raw date time data into more specific features that better represent the underlying patterns hidden in the original feature.

To be more specific, column *INCDTTM*, which records the date and time of the incident, is converted to date time format and extracted more detailed information to create new features to better illustrate the distribution of the incidents happened at different time.

Another technique we will use is called Feature Binning, which is a method of turning continuous variables into categorical values. The main motivation of binning is to make the model more robust and prevent overfitting, however, it has a cost to sacrifice information and make the data more regularized and generalized.

In the raw dataset, column *PERSONCOUNT* provides the total number of people involved in the collision. However, it is too specific and inclined to overfitting problem if the original format of the data is utilized in the model. So is column *VEHCOUNT*, which provides the number of vehicles involved in the collision. Thus, binning method is applied to these two numeric columns to be converted into categorical values. Notice that the trade-off between performance and overfitting is the key point of the binning process.

Figure 4: Feature Binning



```
collision_selected["PERSON_CAT"] = pd.cut(collision_selected["PERSONCOUNT"], bins = [0, 2, 5, 10, 100], right = False,
                                          labels = ["Small(0-1)", "Mid(2-4)", "Large(5-9)", "Larger(>9)"])
collision_selected["VEH_CAT"] = pd.cut(collision_selected["VEHCOUNT"], bins = [0, 1, 2, 4, 20], right = False,
                                       labels = ["Non-Veh", "Small(1)", "Mid(2-3)", "Large(>3)"])
```

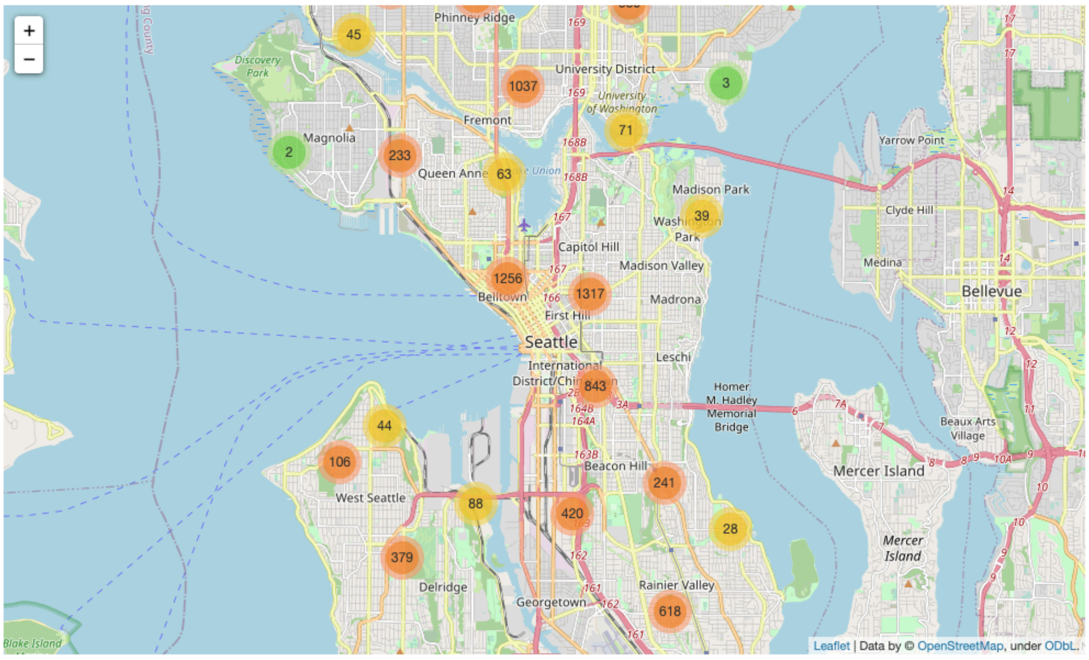
3.2 Exploratory Data Analysis

i. Maps of Accidents

The first question we might ask is where the accidents happened, are they geographically evenly distributed or are they clustered in some particular locations? To answer this question, we use library Folium to visualize geospatial data that has been manipulated on an interactive Leaflet map, which is represented as column *X* and *Y* in the dataset. Also, the severity of each accident is visualized as marker on the map.

Taking computational cost and storage consumption into consideration, we will only subsample 10,000 cases out of the dataset to get a general sense of the geographic distribution of the accidents.

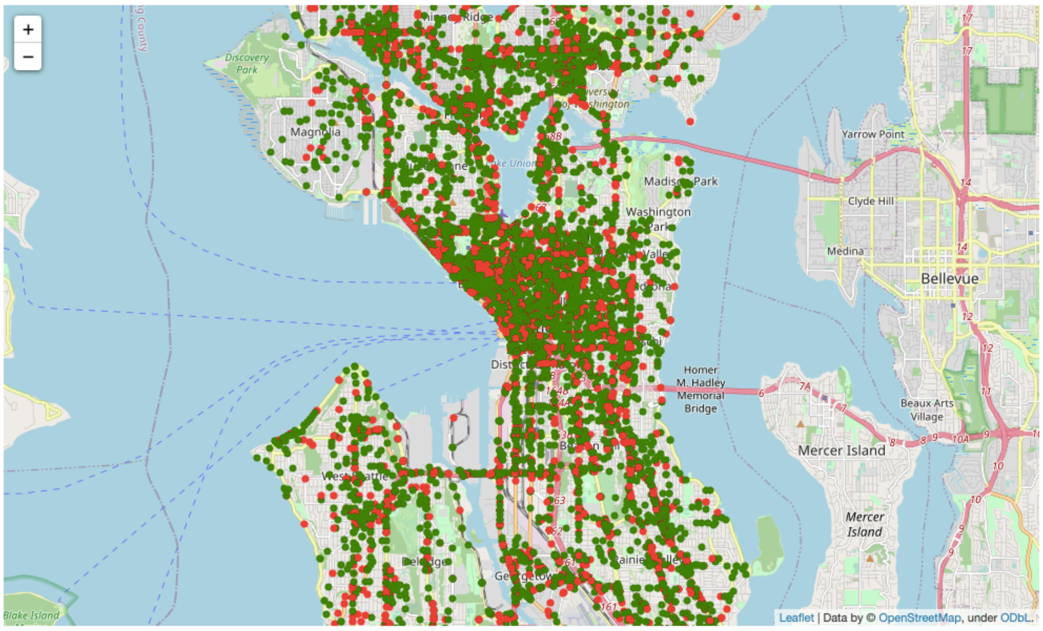
Figure 5: Accidents Distribution



As displayed in the map, it is clear that, with the busiest traffic within the area, downtown Seattle has the highest density of accidents, while some districts, such as Magnolia and Southwest, with the lowest density. This map does prove that, generally, the busier the district traffic is, the more accidents the area will have.

In the following map, the severities of the accidents are added with different colors. Green markers represent accidents with a severity of 1, which are described as Property Damage Only Collisions; red markers, on the other hand, represent accidents with a severity of 2, which are Injury Collisions.

Figure 6: Collision Severity Distribution



However, unlike the density map, there is no clear pattern exist between the severity of the accidents and the geographic location where they happened. This map suggests that, when we try to predict the severity of an accident, geographic location might not be an effective parameter that we would like to incorporate in our model.

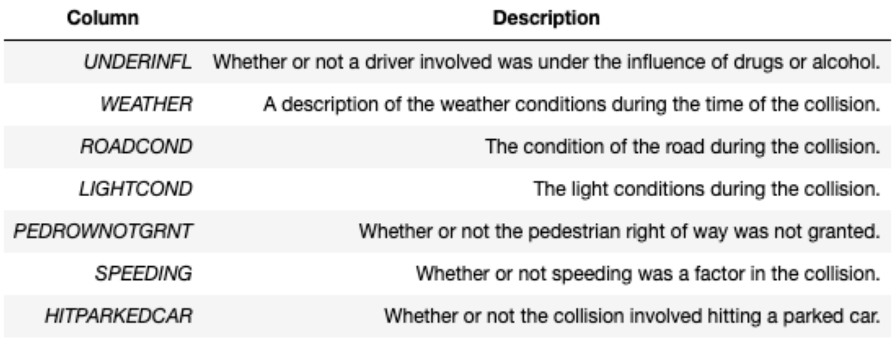
Even though the maps seem not to focus on the objectives of the project, they still provide a sense of the data in geographic way and remind us that the dataset comes from the real world, which we should keep in mind through the entire project when we are building our model.

ii. Collisions in Different Types

There are two variables, including ADDRTYPE and COLLISIONTYPE, in the dataset that categorize where and how the collisions happened. To discover the potential relationships between each of these parameters with the collision severity, we can visualize the distribution of the collision severity within each category.

Also, similar exploratory visualization can be applied to columns that describe the conditions when the collisions took place, including

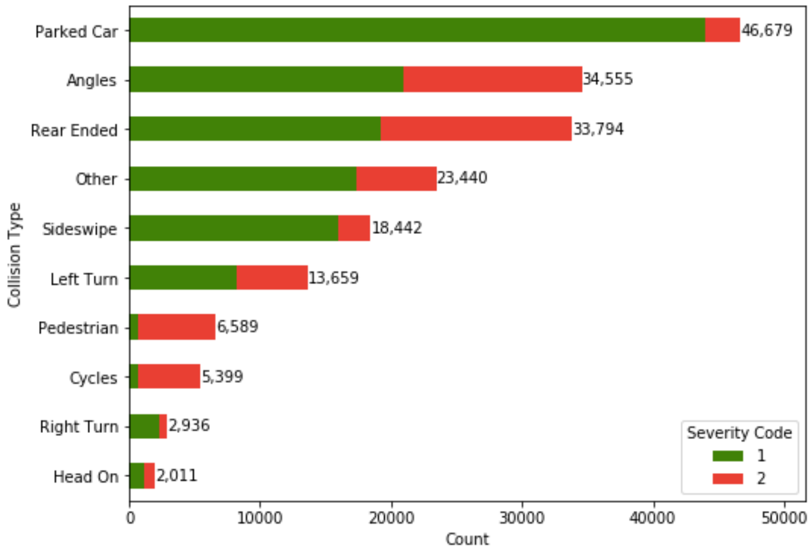
Table 2: Feature Description



Column	Description
UNDERINFL	Whether or not a driver involved was under the influence of drugs or alcohol.
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.
PEDROWNOTGRNT	Whether or not the pedestrian right of way was not granted.
SPEEDING	Whether or not speeding was a factor in the collision.
HITPARKEDCAR	Whether or not the collision involved hitting a parked car.

Take column *COLLISIONTYPE* as an example. The severity distribution within each collision type is displayed in the bar plot below. Interestingly, the graph indicates that, in some particular types, collisions are more likely to be severer than the other collision types. Especially, collisions which *Pedestrian* and *Cycles* are involved in have outstandingly high percentage, 89.8% and 87.6% respectively, of collisions with a severity of 2, while collisions involved with *Parked Car* are more likely to be mild, having only 5.7% of the collisions in this type with a severity of 2. This bar plot certainly suggests that there are some patterns between the different collision types and the collision severity, which makes the collision type as an effective input to the model.

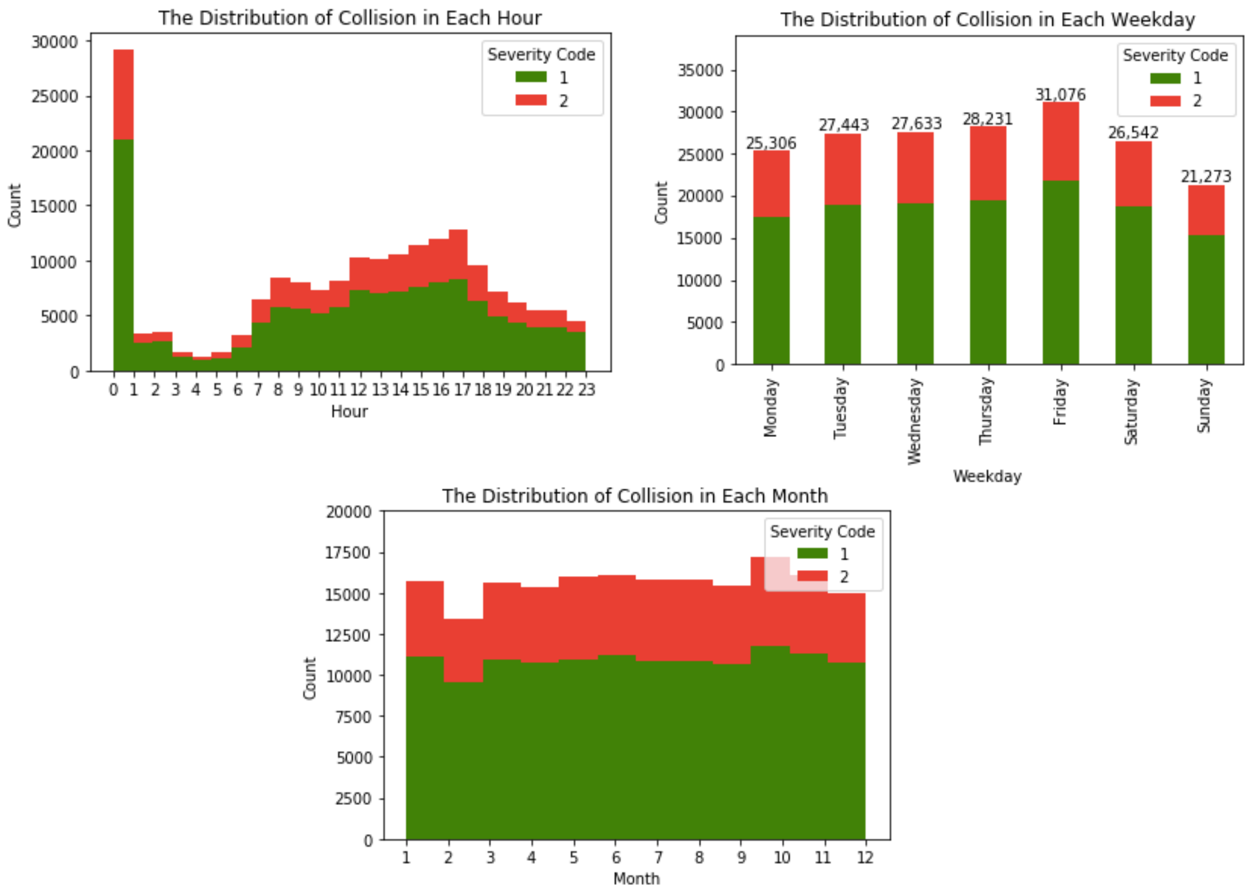
Figure 7: The Distribution of Collision Severity in Each Collision Type



iii. Collision Time

In the Feature Engineering process in Data Preparation, we have extracted more detailed collision time (i.e. Hour, Weekday, and Month) for each collision. Here, we would like to explore whether or not collision time was a factor in the collision.

Figure 8: Collision Time



As shown in the histogram, we can learn that there were more collisions took place in the afternoon and at midnight, remarkably. This does make sense since traffic is usually busier in the afternoon and driving in the dark can be riskier due to fatigue, compromised night vision, the glare of headlights from an oncoming vehicle, and so forth.

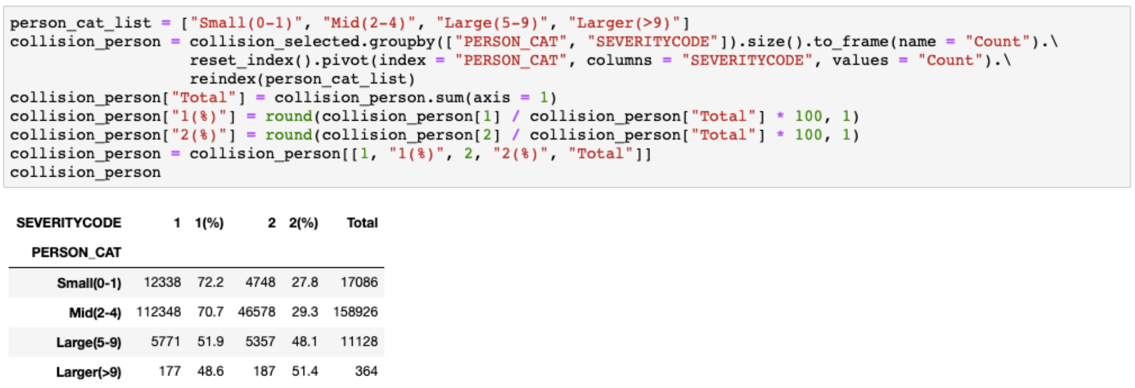
However, it seems that the severity of the collisions may not have much relationship with the time when the collision occurred in a day. Even at midnight, the distribution of collision severity seems no big difference against the collisions occurred during the day.

After checking the weekday and month distribution, unluckily, it seems that neither weekday nor month has sufficient impact on the collision severity. There is no significant difference in the collision severity between weekday and weekend, which is really surprising. Also, collisions and their severity distribute consistently through the entire year, suggesting that month is not our targeted input to the model. Therefore, we will not include any of these time factors into our model.

iv. Collision Size

In Data Preparation, we have created two columns, named *PERSON_CAT* and *VEH_CAT*, by feature binning method. They describe the size of people and vehicles, respectively, involved in the collision. Now, we would like to dig into the two columns and study their potential relationship with collision severity.

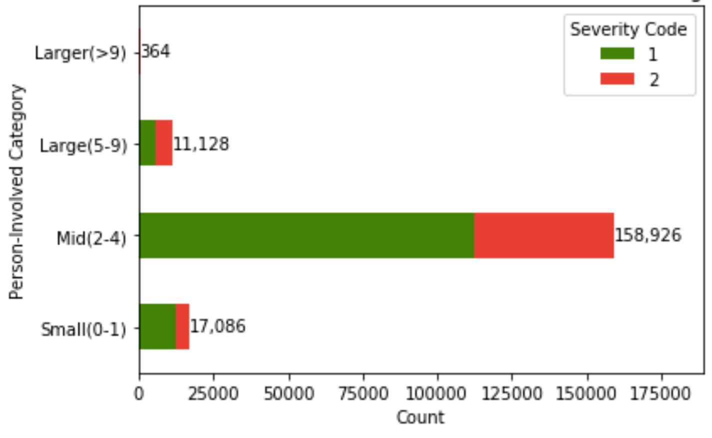
Figure 9: Person-involved Categories



```
person_cat_list = ["Small(0-1)", "Mid(2-4)", "Large(5-9)", "Larger(>9)"]
collision_person = collision_selected.groupby(["PERSON_CAT", "SEVERITYCODE"]).size().to_frame(name = "Count").\
                   reset_index().pivot(index = "PERSON_CAT", columns = "SEVERITYCODE", values = "Count").\
                   reindex(person_cat_list)
collision_person["Total"] = collision_person.sum(axis = 1)
collision_person["1(%)"] = round(collision_person[1] / collision_person["Total"] * 100, 1)
collision_person["2(%)"] = round(collision_person[2] / collision_person["Total"] * 100, 1)
collision_person = collision_person[[1, "1(%)", 2, "2(%)", "Total"]]
collision_person
```

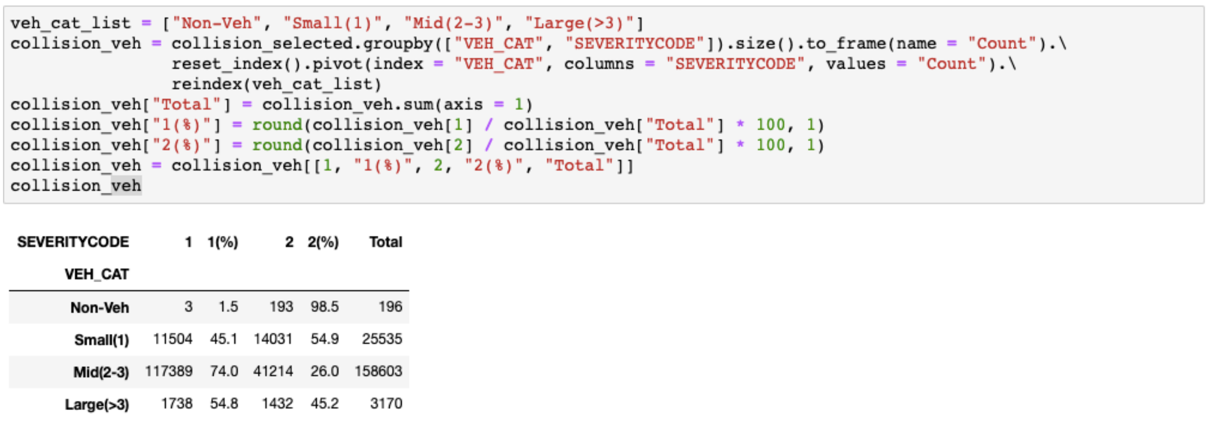
SEVERITYCODE	1	1(%)	2	2(%)	Total
PERSON_CAT					
Small(0-1)	12338	72.2	4748	27.8	17086
Mid(2-4)	112348	70.7	46578	29.3	158926
Large(5-9)	5771	51.9	5357	48.1	11128
Larger(>9)	177	48.6	187	51.4	364

Figure 10: The Distribution of Collision in Each Person-Involved Category



From the bar plot, luckily, we can find out that, the larger the size of people involved in the accident, the higher the probability of collision with a severity of 2. This does comply with common sense since the more people involved in the accident, the severer the situation could be. Also, the number of people could be a direct indicator for the size of the accident, and the larger the size of the accident, the more likely the accident is severe.

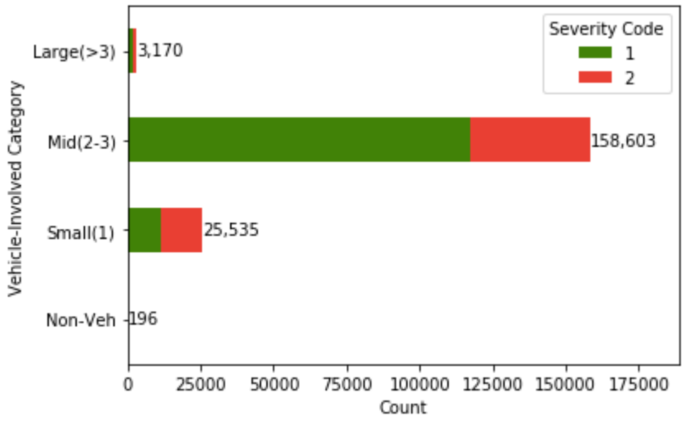
Figure 11: Vehicle-Involved Category



```
veh_cat_list = ["Non-Veh", "Small(1)", "Mid(2-3)", "Large(>3)"]
collision_veh = collision_selected.groupby(["VEH_CAT", "SEVERITYCODE"]).size().to_frame(name = "Count").\
                reset_index().pivot(index = "VEH_CAT", columns = "SEVERITYCODE", values = "Count").\
                reindex(veh_cat_list)
collision_veh["Total"] = collision_veh.sum(axis = 1)
collision_veh["1(%)"] = round(collision_veh[1] / collision_veh["Total"] * 100, 1)
collision_veh["2(%)"] = round(collision_veh[2] / collision_veh["Total"] * 100, 1)
collision_veh = collision_veh[[1, "1(%)", 2, "2(%)", "Total"]]
collision_veh
```

SEVERITYCODE	1	1(%)	2	2(%)	Total
VEH_CAT					
Non-Veh	3	1.5	193	98.5	196
Small(1)	11504	45.1	14031	54.9	25535
Mid(2-3)	117389	74.0	41214	26.0	158603
Large(>3)	1738	54.8	1432	45.2	3170

Figure 12: The Distribution of Collision in Each Vehicle-Involved Category



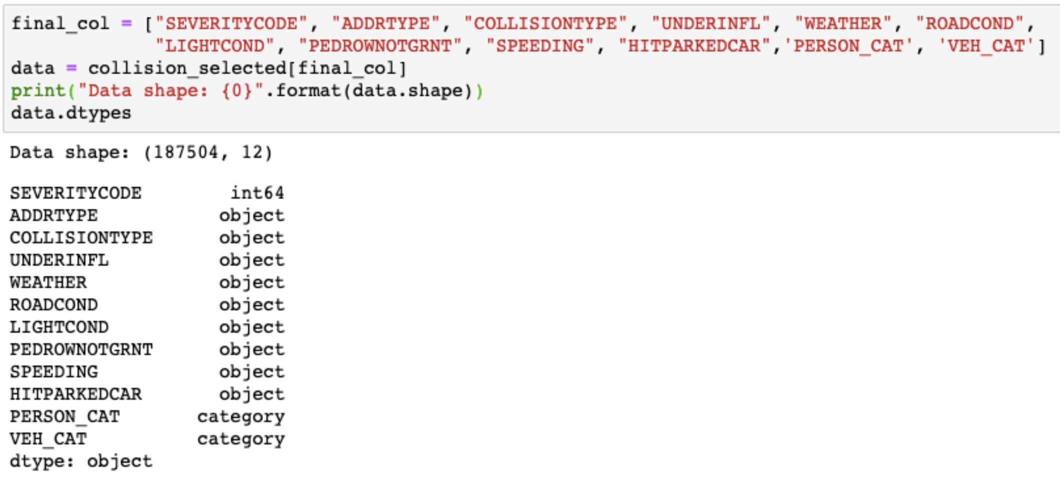
Interestingly, the collision severity neither consistently increase nor consistently decrease when the number of vehicles involved in the collision. Instead, when there is no vehicle involved in, collisions with a severity of 2 are overwhelming, indicating that those collisions, for whatever reason, usually have higher severity of injury. On the other hand, middle-sized collisions seem to have the lowest percentage (i.e. 26.0%) of higher-severity collision.

The collision severity seems to be more fluctuated and has certain relationship with the number of vehicles involved in the collision, which is slightly different from what we can learn from the person size of the collision.

3.3 Data Wrangling

After exploratory data analysis, we have found out that not all of the selected data can be used in modeling. It turns out that some variables have limited impact on predicting the output variable, *SEVERITYCODE*. Therefore, we will, again, filter out those inappropriate attributes and convert the data into a desired format.

Figure 13: Data Wrangling



```
final_col = ["SEVERITYCODE", "ADDRTYPE", "COLLISIONTYPE", "UNDERINFL", "WEATHER", "ROADCOND",
             "LIGHTCOND", "PEDROWNOTGRNT", "SPEEDING", "HITPARKEDCAR",'PERSON_CAT', 'VEH_CAT']
data = collision_selected[final_col]
print("Data shape: {0}".format(data.shape))
data.dtypes
```

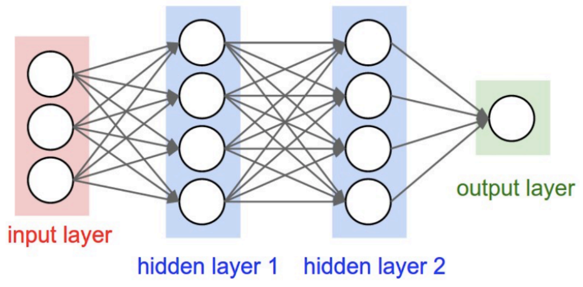
```
Data shape: (187504, 12)

SEVERITYCODE        int64
ADDRTYPE           object
COLLISIONTYPE      object
UNDERINFL          object
WEATHER            object
ROADCOND           object
LIGHTCOND          object
PEDROWNOTGRNT      object
SPEEDING           object
HITPARKEDCAR       object
PERSON_CAT       category
VEH_CAT          category
dtype: object
```

3.4 Modeling

As we can see from data wrangling results, the final dataset has 187504 entries and 12 attributes, which include 1 output variable and 11 categorical input variables. Since the input variables are all categorical, we do not need to normalize the data here.

Taking the size of predictors and records into consideration, multilayer feedforward Neural Networks, which is one of the most common applications of neural networks in data mining, is applied in this project.

Figure 14: Feedforward Neural Network Structure

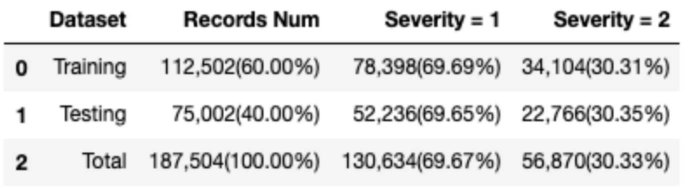


The graph shown above is a typical structure of the feedforward neural networks. In this structure, Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. The structure shown above is a 3-layer neural network, which is the same structure we will use to build our models, instead, we will have 11 attributes.

Neural Networks mimic the biological activity in the human brain, where neurons are interconnected and learn from experience. They can capture complex, or generally unknown, relationships between predictors and an outcome variable, which is often not possible with other predictive models.

Let's first split the dataset into training set and testing set and extract input and output variables separately. Notice that, since neural networks can only work with numerical data, we will convert the input variables into dummies.

Table 3: Splitting Data

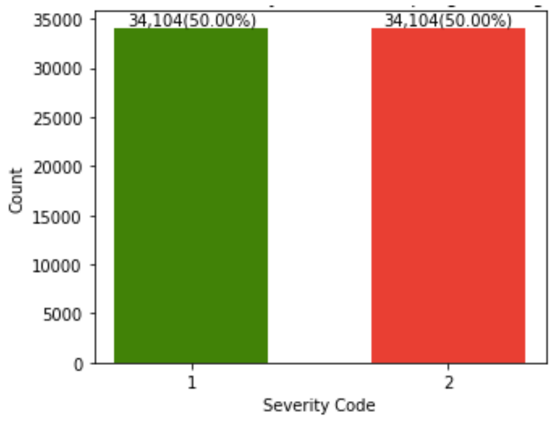


	Dataset	Records Num	Severity = 1	Severity = 2
0	Training	112,502(60.00%)	78,398(69.69%)	34,104(30.31%)
1	Testing	75,002(40.00%)	52,236(69.65%)	22,766(30.35%)
2	Total	187,504(100.00%)	130,634(69.67%)	56,870(30.33%)

As mentioned before, since the raw data has unbalanced labels, two sampling approaches will be applied to handle this problem. Notice that we will only balance the training dataset and remain the testing set unchanged to reflect the original data imbalance.

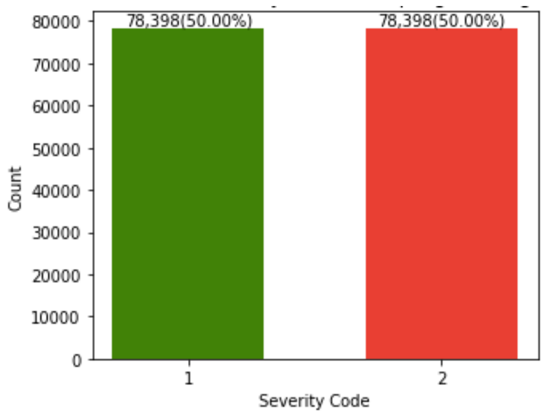
The first method is called undersampling approach, which is to delete instances from the over-represented class, which is severity of 1, and keep an equal balance between the majority and minority class. Here we apply the simplest undersampling method which involves randomly deleting examples from the majority class, referred as random undersampling.

Figure 15: The Distribution of Severity in Undersampling Training Dataset



The second method, named oversampling approach, on the other hand, duplicates examples in the minority class or synthesize new examples from the examples in the minority class, which is severity of 2, in order to reach an equal balance between the minority and majority class.

Figure 16: The Distribution of Severity in Oversampling Training Dataset



Here, we construct our neural network architecture for the undersampling training data - a 58-58-40-2 feedforward neural network. We have two hidden layers each with 58 and 40 nodes, respectively, applying ReLU activation function and initialized by HE uniform initialization, and a single unit output layer as in a typical binary classification task (that is, using a sigmoid as activation function and binary cross-entropy as loss). To train the model, we set the learning rate parameter of the Adam optimizer to 0.001. The batch size defines the number of samples that will be propagated per gradient update through the network and is set as 50. The network is then allowed to train for a total of 10 epochs, meaning that the model iterates over the entire provided data 10 times in an attempt to learn an underlying pattern.

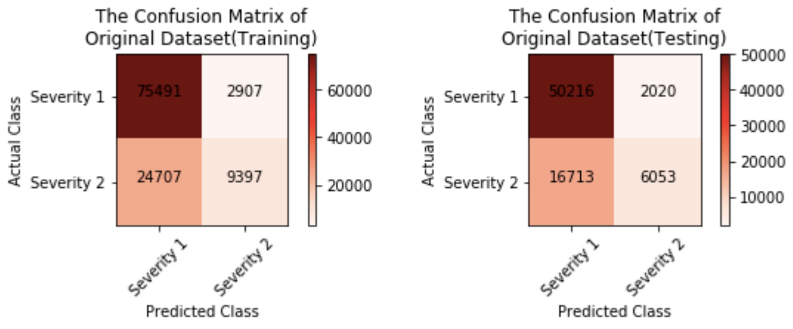
Figure 17: Neural Network Model Structure

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 58)                3422      
_________________________________________________________________
dense_2 (Dense)              (None, 40)                2360      
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 41        
=================================================================
Total params: 5,823
Trainable params: 5,823
Non-trainable params: 0
_________________________________________________________________
```

4. Results

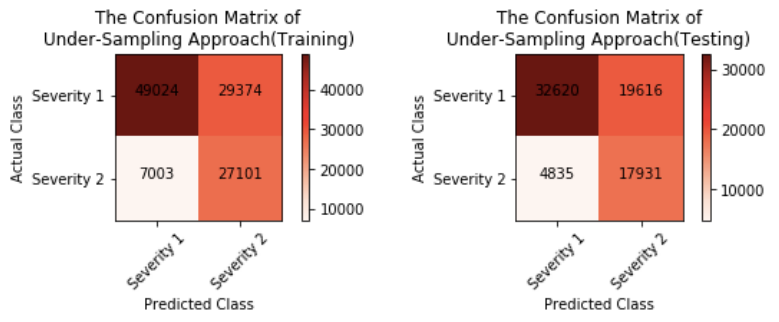
Key results are described in this section. We not only focus on the accuracy of each model, but also other metrics such as Recall, Precision, and F1 score. By visualizing the confusion matrix and summarizing the performance, we can get a clear sense of how the models perform against training and testing data.

Figure 18: Original Dataset Results



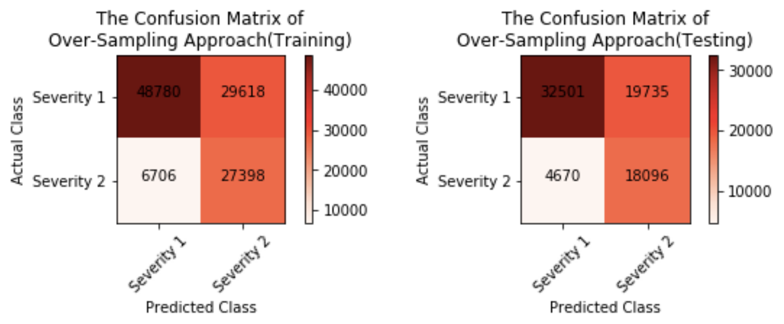
```
            Training Data    Testing Data
---------  ---------------  --------------
Accuracy             0.755            0.75
Recall               0.276           0.266
Precision            0.764            0.75
F1 Score             0.405           0.393
```

Figure 19: Undersampling Dataset Results



```
             Training Data    Testing Data
---------  ---------------  --------------
Accuracy             0.677           0.674
Recall               0.795           0.788
Precision             0.48           0.478
F1 Score             0.598           0.595
```

Figure 20: Oversampling Dataset Results



```
             Training Data    Testing Data
---------  ---------------  --------------
Accuracy             0.677           0.675
Recall               0.803           0.795
Precision            0.481           0.478
F1 Score             0.601           0.597
```

As demonstrated, the neural network model built by the original dataset has an Accuracy and Precision score of 75%, while Recall score and F1 score are too low to satisfy. Especially, the model overwhelmingly predicts collision with a severity of 1, which does make sense considering the imbalanced original data. This is exactly the main reason why we seek for the sampling approaches to avoid this problem at the first place.

On the other hand, the Accuracy of each sampling-approach model is around 67%, either against training data or testing data. Also, the Recall score and Precision score are both consistent with the two datasets, as well as the F1 score, which can be regarded that the models neither overfit nor underfit with the training data. Considering the size of the dataset, we are happy to see the consistency of model performance, even when it confronts with never-seen data.

When comparing the two sampling-approach models with the original model, although the accuracy of these two models is slightly lower than the model built by the original data, the F1 score is much better in these two models. F1 score is the weighted average of Precision and Recall; therefore, this score takes both false positive(FP) and false negative(FN) into account. Alone, neither Precision nor Recall score tells the whole story. We can have excellent Precision with terrible Recall score, like we have in the original model, or alternatively, terrible Precision with excellent Recall score. Instead, F1 score provides a way to express both concerns with a single score and it is widely used on imbalanced classification problems. Therefore, we still consider the sampling-approach models outperform the original model on the whole.

As for the two sampling approaches (i.e. undersampling and oversampling), we are surprised that models built by the different sampling datasets turn out to have similar performance in predicting the collision severity. This suggests that neural network models built in this project are not sensitive to the size of the training data.

However, what should be concerned in here is that, there are still a lot of collisions misclassified by both models, especially in collisions with a severity of 1. To be more specific, around 40% of the collisions with a severity of 1 in actual are predicted with a severity of 2. This problem seems like a backfire issue because of the balanced data through the sampling approaches.

5. Conclusion and Future Work

In this project, we addressed the problem of predicting collision severity based on the multiple attributes which describe the conditions when the collision occurred. Three models were built by different formats of data, including original, undersampling, and oversampling, in order to compare the influence of imbalanced labels in data mining and machine learning. Feedforward

Neural network architecture was applied to build the models and the performance of each model was compared and analyzed. Potential future work could focus on tuning hyper parameters in the neural network, with the purpose of improving the prediction performance of the models. In addition, it would be interesting to further explore the importance of each attribute included in this project and rebuilt the models by different sets of attributes. These results could help us better predict the collision severity based on the internal and external factors.

References

National Highway Traffic Safety Administration. (2020). People Saving People – On the Road to a Healthier Future. Retrieved from https://one.nhtsa.gov/nhtsa/whatis/planning/2020Report/2020report.html

Traffic Management Division, Traffic Records Group. (2020). Collisions [Data set]. Seattle Department of Transportation. https://opendata.arcgis.com/datasets/5b5c745e0f1f48e7a53acec63a0022ab_0.csv

Seattle Department of Transportation. (n.d.) Collisions-All Years. https://www.seattle.gov/Documents/Departments/SDOT/GIS/Collisions_OD.pdf