Coursera Data Science Project

Predict severity of vehicles accident in Seattle by

building machine Learning model

Irina Dzyu

September 2020

1. Introduction

1.1. Background

A traffic collision, also called a motor vehicle collision, car accident, or car crash, occurs when a vehicle collides with another vehicle, pedestrian, animal, road debris, or other stationary obstruction, such as a tree, pole or building. Traffic collisions often result in injury, disability, death, and property damage as well as financial costs for people and government (Wikipedia).

There are number of factors involved to the risk of accident, including vehicle condition, road design and environment, weather, human behavior and other destruction objects on a road. Million people worldwide died or sustained injuries from traffic collisions.

Importance of vehicle accidents study is obvious; prediction model will help reduce accident risks and support citizen avoid road traffic by changing or cancelling their trips as well as minimize financial cost from crash damages. Therefore, predictive model based on machine learning will have positive impact on daily life for both society and individuals involved.

1.2. Problem description

Cars accidents are the serious and significant challenge of this city. "Seattle's status as the country's fastest-growing city — adding 19,901 people since the 2016 report — is a backdrop to the traffic rundown," - Material from The Seattle Times. There is not a well-established mass-transit system in Seattle, most of the residents are drivers. Road congestion, collisions and traffic accident is equally high and is the major cause of death in Seattle. City is surrounded on three sides by water and is a fairly mountainous city. With all of its rain, hills, bridges, tunnels and narrow roads, this city is the perfect storm for car accidents. Combined all factors, car accidents are an hourly occurrence in Seattle.

Thus, the weather and the road conditions are two factors that can support to predict the possibility of getting into a car accident and provide recommendation to drive more careful or change travel plans to avoid car accidents and reduce accidence rate in the city.

2. Data Analysis

2.1. Data source

The project begins, by using Jupyter Notebook, web-based multi-functional interactive tool used for programming, visualization and coding

I will use the Notebook to build the code based on CVS file (Data-Collisions.csv), this Dataset includes all types of collisions provided by SDOT Traffic Management Division, Traffic Records Group. Collisions will display at the intersection or mid-block of a segment. Timeframe: 2004 to Present. All collisions provided by SPD and recorded by Traffic Records in Seattle. To download the data, will use !wget to download it from IBM Object Storage.

Metadata form (Collisions All Year.pdf) contains a description of all features. By reading it, it’s possible to identify features to evaluate source data.

2.2. Data cleaning

Data cleanup is needed to start analysis and data methodology. Followed changes to clean up data performed on Dataset:

a) Removal of Non-Relevant Features

This dataset contains 194,674 samples. Each with 34 features, including the target one which want to predict. However, not all these features will be useful or relevant towards our goal. The file Collisions All Year.pdf (Metadata form) – contains a description of features. By reading it, it’s possible to identify some features that can be disregarded from the outset.

The following features will be removed from Data analysis. Either because they aren't relevant to problem or create redundancy:

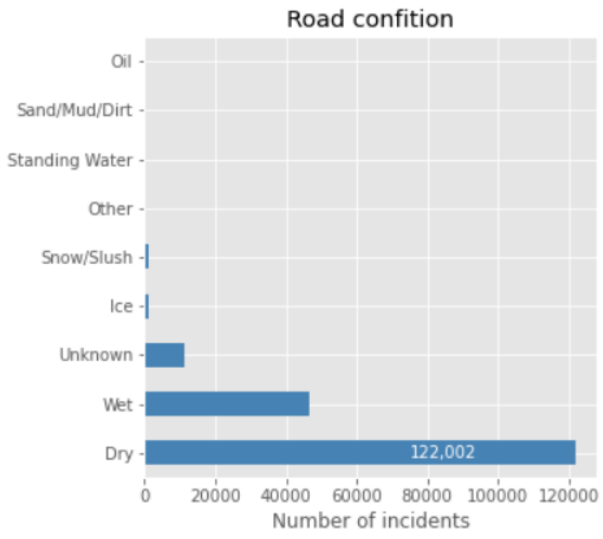
drop = ['X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPORTNO', 'STATUS', 'INTKEY', 'EXCEPTRSNCODE', 'EXCEPTRSNDESC', 'SEVERITYCODE.1', 'INCDATE', 'INCDTTM', 'INATTENTIONIND', 'UNDERINFL', 'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'] *b) Removal N/A values*

All rows contain “NA” values will be removed and this changes resulted 182,660 records. It was dropped 6.2% of the samples due to inconsistent values and we have dropped 21 features that weren't meaningful. The next step is to check a few numerical features.

```
[12]: #check how many null objects we have in the dataset as follows:
      new_df.isnull().sum()
```

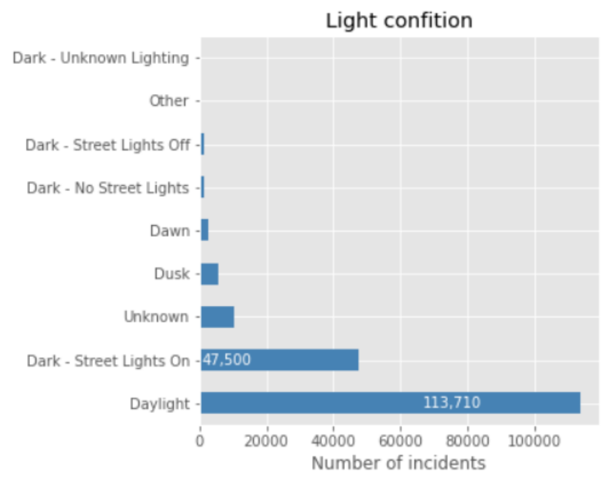
```
[12]: SEVERITYCODE      0
      ADDRTYPE          0
      ADDRESS           0
      SEVERITYDESC      0
      COLLISIONTYPE     0
      PERSONCOUNT       0
      PEDCOUNT          0
      PEDCYLCOUNT       0
      VEHCOUNT          0
      JUNCTIONTYPE      0
      SDOT_COLCODE      0
      SDOT_COLDESC      0
      WEATHER           0
      ROADCOND          0
      LIGHTCOND         0
      ST_COLCODE        0
      ST_COLDESC        0
      dtype: int64
```

For remain selected attributes 'ROADCOND','LIGHTCOND','WEATHER','COLLISIONTYPE' decided to remove samples including NA values, In this instance this as of no consequence. However, it could be necessary to consider removing such data to reduce the noise it might generate.



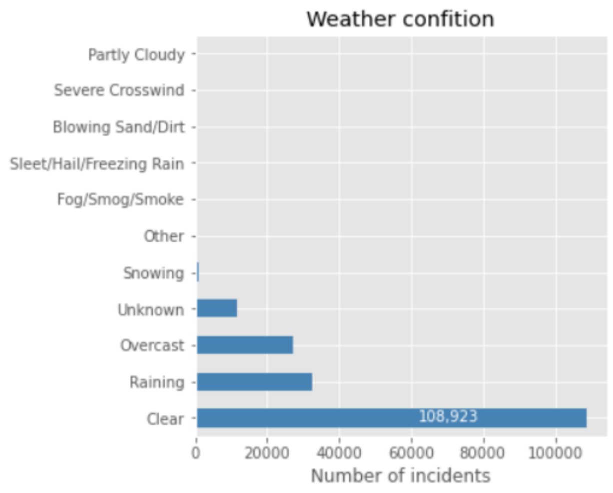
```
# review Roadcond data
new_df['ROADCOND'].value_counts()
```

```
Dry               122002
Wet                46671
Unknown            11479
Ice                 1176
Snow/Slush           977
Other                123
Standing Water       108
Sand/Mud/Dirt         64
Oil                   60
Name: ROADCOND, dtype: int64
```



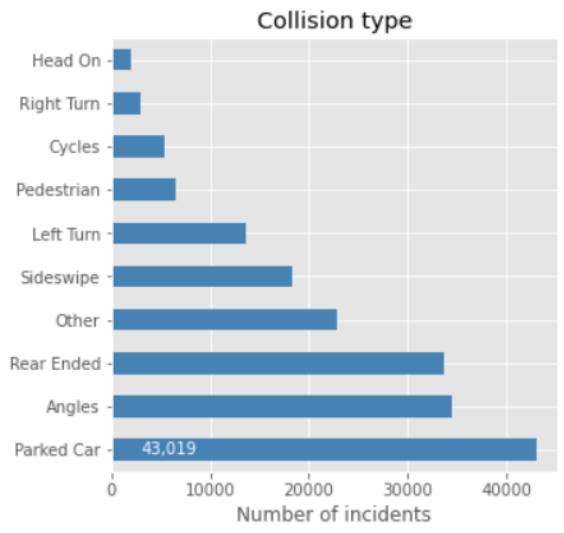
```
new_df['LIGHTCOND'].value_counts()
```

```
Daylight                    113710
Dark - Street Lights On      47500
Unknown                      10413
Dusk                          5768
Dawn                          2450
Dark - No Street Lights       1444
Dark - Street Lights Off      1154
Other                          210
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```



```
new_df['WEATHER'].value_counts()
```

```
Clear                       108923
Raining                      32616
Overcast                     27155
Unknown                      11597
Snowing                        881
Other                          742
Fog/Smog/Smoke                 555
Sleet/Hail/Freezing Rain       112
Blowing Sand/Dirt               49
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

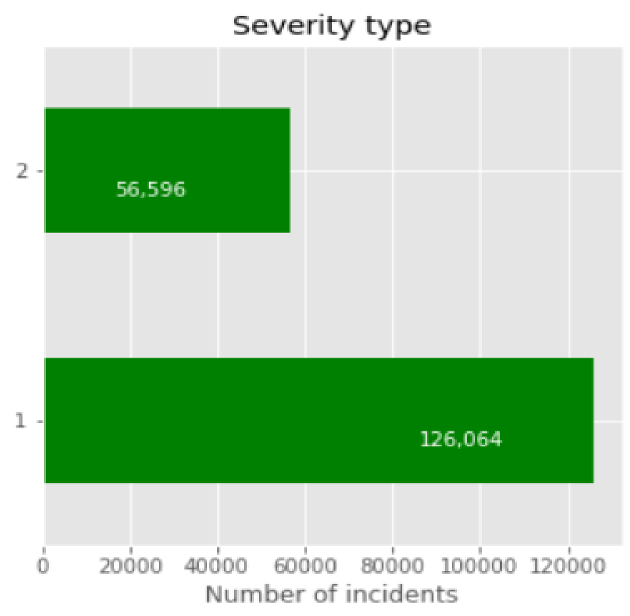


```
new_df['COLLISIONTYPE'].value_counts()
```

```
Parked Car    43019
Angles        34444
Rear Ended    33639
Other         22860
Sideswipe     18282
Left Turn     13637
Pedestrian     6497
Cycles         5358
Right Turn     2929
Head On        1995
Name: COLLISIONTYPE, dtype: int64
```

For each feature, we need to consider the cases with inconsistent and/or missing values. This results in either imputing the missing values, dropping a specific sample or even dropping a feature altogether –the details are available in the notebook loaded in on GitHub. After doing this for each categorical and numerical features, we end up dropping 21 of them that were irrelevant to our problem; or due to too many missing/inconsistent values, and with no clear way to impute them. Also, we have dropped some samples that have some inconsistent values for some feature that behave as expected otherwise.

In any case, the most important thing that we have learned from this step is that there is a huge imbalance of classes for our target variable:



Given that “1-prop damage” crashes are the most common ones and that as the severity of the crash increases, so does the importance of making a good prediction. 31% of cleaned sample data is 2-injury, and furthermore 2b – serious injury and 3-fatality are not presented in current dataset. This issues needs to be highlighted if we hope to produce a good operational model.

The model to be able to analyze and predict the severity of a crash regardless of when it happened, thus time features are excluded as well.

3. Predictive modeling

In machine learning classification is a supervised learning approach which can be thought of as a means of categorizing or classifying some unknown items into a discrete set of classes. Classification attempts to learn the relationship between a set of feature variables and a target variable of interest. The target attribute in classification is a categorical variable with discrete values. Given a set of training data points along with the target labels, classification determines the class label for an unlabeled test case.

The goal of a car accident predictor is to use existing car accident default data which has information about independent variables such as the condition of the road during the collision, the light conditions during the collision, a description of the weather conditions during the time of the collision to build a classifier, prevent trip or avoid future accident to the model, and then label it, i.e the data points as proper damage or injury. Or for example zero or one. This is how a classifier predicts an unlabeled test case.

Will use the training set to build an accurate model. Then use the test set to report the accuracy of the model I will use the following algorithm:

- K Nearest Neighbor(KNN)
- Decision Tree
- Logistic Regression

3.1. K Nearest Neighbor (KNN)

KNN will help predict the severity code of an outcome by finding the most similar to data point within k distance.

If collisions data can be used to predict type of severity, the Traffic Management Division can send Seattle citizens security notifications to reduce traffic and avoid car accidents on a road. This is a classification problem. That is, given the dataset with predefined labels, we need to build a model to be used to predict the class of a new or unknown case. The study focuses on using factors such as the condition of the road during the collision, the light conditions during the collision, a description of the weather conditions during the time of the collision to predict severity and collision type.

The target field called SEVERITYCODE has four possible values that correspond to the four severity type as follows: 1-proper damage, 2-injury, (2b – serious injury and 3-fatality, not presented in our dataset population). Our objective is to build a classifier.

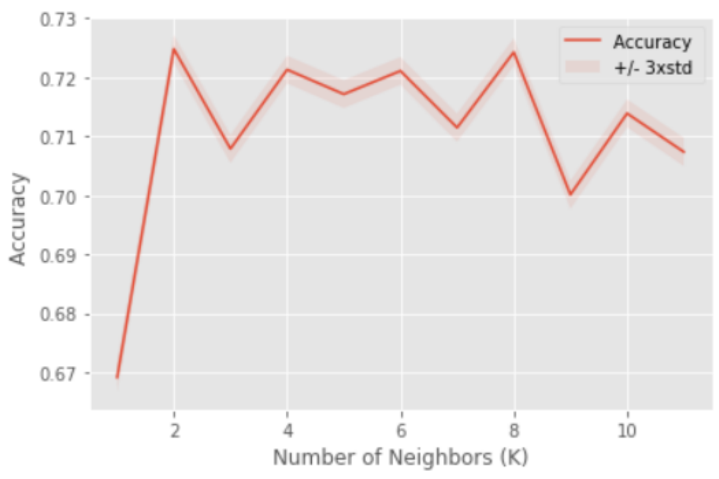
The K-Nearest Neighbors algorithm is a classification algorithm that takes a bunch of labeled points and uses them to learn how to label other points. This algorithm classifies cases based on their similarity to other cases. In K-Nearest Neighbors, data points that are near each other are said to be neighbors. K-Nearest Neighbors is based on this paradigm. Similar cases with the same class labels are near each other. Thus, the distance between two cases is a measure of their dissimilarity. There are different ways to calculate the similarity or conversely, the distance or dissimilarity of two data points.

Normalize Data - Data Standardization give data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on distance of cases - the details are available in the notebook loaded in on GitHub.

Out of Sample Accuracy is the percentage of correct predictions that the model makes on data that that the model has NOT been trained on. Doing a train and test on the same dataset will most likely have low out-of-sample accuracy, due to the likelihood of being over-fit.

It is important that our models have a high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. Use of an evaluation approach called Train/Test Split will improve out-of-sample accuracy.

Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set.



```
The best accuracy was with 0.7248165991459542 with k= 2
```

K in KNN, is the number of nearest neighbors to examine. In our study starts from Ks=12. The general solution is to reserve a part of your data for testing the accuracy of the model. Then chose k =1, use the training part for modeling, and calculate the accuracy of prediction using all samples in the test set. After multiple interactions decreasing/increasing the k, and final best accuracy is 0.72 with Ks=2 is the best for the model.

3.2. Decision Tree

A decision tree model gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision. It context, the decision tree observes all possible outcomes of different weather conditions.

The feature sets of this dataset are age, gender, blood pressure, and cholesterol of our group of patients and the target is the drug that each patient responded to. It is a sample of binary classifiers, and you can use the training part of the data set to build a decision tree and then use it to predict the class of an unknown patient. In essence, to come up with a decision on which drug to prescribe to a new patient. Let's see how a decision tree is built for this dataset. Decision trees are built by splitting the training set into distinct nodes, where one node contains all of or most of one category of the data. If we look at the diagram here, we can see that it's a patient's classifier. So as mentioned, we want to prescribe a drug to a new patient, but the decision to choose drug A or B will be influenced by the patient's situation. We start with age, which can be young, middle aged or senior. If the patient is middle aged, then we'll definitely go for drug B. On the other hand, if he has a young or a senior patient, will need more details to help us determine which drug to prescribe. The additional decision variables can be things such as cholesterol levels, gender or blood pressure.

We will be using train/test split on our decision tree. Let's import train_test_split from sklearn.cross_validation.

ne more change on data was applied, but did not drop any further samples.

As all features in this dataset are categorical, and Sklearn Decision Trees do not handle categorical variables. Need to convert these features to numerical values pandas.get_dummies() - Convert categorical variable into dummy/indicator variables.

Now will return 4 different parameters. We will name them: x_train, X_test, y_train, y_test. The train_test_split will need the parameters: X, y, test_size=0.3, and random_state=3. X and y are the arrays required before the split, the represents the ratio of the testing dataset, and the random_state ensures that we obtain the same splits. We will first create an instance of the DecisionTreeClassifier called drugTree.

Inside of the classifier, specify criterion="entropy" so we can see the information gain of each node. Next, we will fit the data with the training feature matrix X_train and training response vector y_train.

Accuracy classification score computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

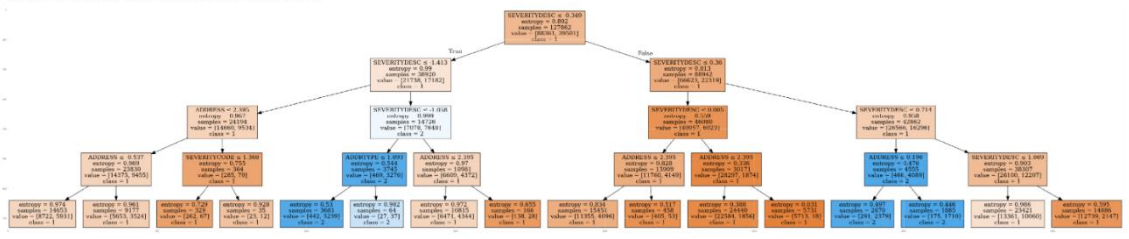
In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

Finally, installing packages for tree visualization.

!conda install -c conda-forge pydotplus -y &

!conda install -c conda-forge python-graphviz -y

```
<matplotlib.image.AxesImage at 0x7f60bbada780>
```



```
DecisionTrees's Accuracy:  0.7388955801306617
```

3.3. 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. It is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.

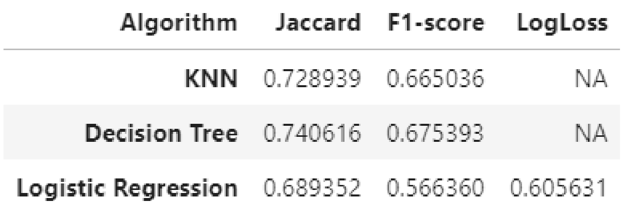
Logistic Regression is a variation of Linear Regression, useful when the observed dependent variable, y, is categorical. It produces a formula that predicts the probability of the class label as a function of the independent variables.

Because our dataset only provides us with two severity code outcomes (SEVERITYCODE), our model will only predict one of those two classes (1-proper damage, 2-injury / 2b – serious injury and 3-fatality, not presented in our dataset population). This makes our data binary, which is perfect to use with logistic regression.

3.4. Model evaluation

The result is reported as the accuracy evaluation of each classifier, using the following metrics when these are applicable:

- jaccard index - define jaccard as the size of the intersection divided by the size of the union of two label sets. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.
- F1 score - score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is a good way to show that a classifier has a good value for both recall and precision.
- log loss - In logistic regression, the output can be the probability of customer churn is yes (or equals to 1). This probability is a value between 0 and 1. Log loss (Logarithmic loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1.



Algorithm	Jaccard	F1-score	LogLoss
KNN	0.728939	0.665036	NA
Decision Tree	0.740616	0.675393	NA
Logistic Regression	0.689352	0.566360	0.605631

4. Conclusion & Future Directions

In this study, I analyzed the relationship between multiple factors and collision types. Based on historical data from SDOT Traffic Management Division conditions pointing to certain classes, i can conclude that particular condition of the road during the collision, the light conditions during the collision, a description of the weather conditions during the time of the collision have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2).

Once data was analyzed and cleaned, I built three classification models (KNN, Decision Tree and Logistic Regression) to predict how much possible accident. Although the first two are ideal for this project, logistic regression made most sense because of its binary nature.

Evaluation metrics used to test the accuracy of our models were jaccard index, f-1 score and log loss for logistic regression. Choosing different k, max depth and hypermeter C values helped to improve our accuracy to be the best possible.

These models can be very useful in helping SDOT Traffic Management Division to provide security notifications to reduce accident risks and support citizen avoid road traffic by changing or cancelling their trips as well as minimize financial cost from crash damages. Therefore, selected predictive models will have positive impact on daily life for both society and individuals involved.