A Project Report on

"Road Accident Analysis & Prediction of Accident Severity in Seattle"

Priyesh Saini

September 5th, 2020

1. Introduction

1.1 Background:

In the last two decades, most of the countries have witnessed a steep rise in road accidents. The two crucial factors that are responsible for it are population growth and large immigration to urban cities. In this project, we are going to take the case of the City of **Seattle.**

Seattle, a seaport city on the west coast of the US, is home to around 8 lac people. Washington State's largest city is also an address of some of the largest Tech Industries with Microsoft and Amazon headquartered in its metropolitan area. Residents of this city get around by various means of transportation such as a car, trolley, streetcar, public buses, bicycle, on foot and by the rail. With such busy streets, it is of no surprise that the city is witnessing road accidents every-day.

According to the recent 2019 Annual Traffic Collison report from the Washington State Department of Transportation (WSDOT), there were a total of 10,315 cases of crashes in Seattle alone. Out of which, 22 were fatal, 190 were serious injury collisions and 834 were minor injury collisions.

From the above stats, it is evident that the city needs some strict measures to counteract the current situation. Predicting the severity of crashes is a crucial constituent of reducing the consequences of accidents. This project is a major step to do the same.

1.2 Problems:

An Accident is a final consequence of the number of factors at play. There are plethora of factors such as Infrastructure (road condition, Sign Boards), weather conditions, vehicle condition, traffic behaviour (driver, pedestrian and passengers), characteristics of driver (age, attentiveness), location, time of the day, etc. Furthermore, some of those factors are more important in determining the severity than the others. Thus, it is evident that the analysis of determinant factors will help in revealing more patterns and knowledge that can be used in predicting severity, prevention and safety strategies of traffic accidents.

1.3 Interests:

This project and its conclusions are of paramount importance not only for the Seattle Department of Transportation (SDOT) but also to the general public so that they can take precautionary measures to avoid the accidents. The outcome of this study must be implemented practically for the sake of the safety of citizens and to provide the required treatment in case of any mishaps.

2. Data Wrangling / Data Pre-processing:

2.1 Data Sources:

I am using the dataset provided by the Seattle Police Department (SPD) and recorded by the Seattle Department of Transportation (SDOT)(made available to me by Coursera). It contains all the collision records and was updated weekly. This dataset contains a total of 1,94,673 records and 37 attributes. To get detailed the detailed information and the metadata of this dataset, you can contact SDOT Traffic Management Division, Traffic Records Group.

2.2 Data Cleaning & Features Selection:

There were a lot of problems with the existing dataset. To start with, there were a lot of missing values and they were represented in 2 forms (i.e. NaN values, string values with 'Unknown' Label). Example: ROADCOND attribute has 5012 missing values (NaN Type), but it also has 15078 values with 'Unknown' label.

```
In [9]: df.ROADCOND.value_counts().to_frame()
```

Out[9]:

	ROADCOND
Dry	124510
Wet	47474
Unknown	15078
Ice	1209
Snow/Slush	1004
Other	132
Standing Water	115
Sand/Mud/Dirt	75
Oil	64

```
In [10]: df.ROADCOND.isnull().sum()
Out[10]: 5012
```

Same was the case for all other attributes. So, the first step I took was to convert all the missing values into NaN type so that it will be easy for me while modelling the dataset. Also, I renamed 'X' and 'Y' attribute to 'LONGITUDE' and 'LATITUDE' respectively to make it more meaningful.

After fixing these problems, now it was time for Dimensionality Reduction. There were a lot of attributes which needs to be removed from data frame like,

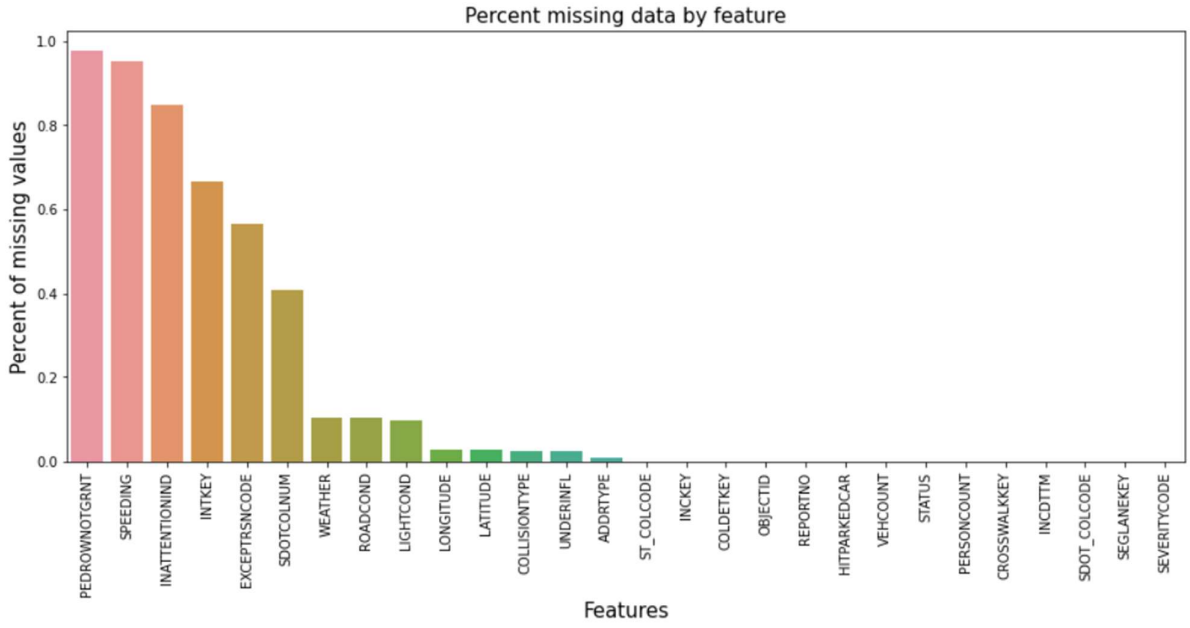
- Handling attributes which were giving the same information repeatedly.
- Handling attributes which have a significant amount of missing data.

Attribute like ‘LOCATION’ is redundant because this information is already available to us in the form of latitude and longitude. So ‘LOCATION’ needs to be dropped. Same was the case for the following attributes: SEVERITYCODE.1, SEVERITYDESC, PEDCOUNT, PEDCYLCOUNT, INCDATE, JUNCTIONTYPE, EXCEPTRSNDESC, SDOT_COLDESC. Hence all these attributes have been dropped.

If an attribute has a lot of missing data, it can create a bias in the model. In this project, I have neglected all those attributes where missing value is more than 40%. Following are the attributes that I dropped:

- PEDROWNOTGRNT, missing value: 97.6%
- SPEEDING, missing value: 95.2%
- INATTENTIONIND, missing value: 84.7%
- INTKEY, missing value: 66.5%
- EXCEPTRSNCODE, missing value: 56.43%
- SDOTCOLNUM, missing value: 43.7%

PEDROWNOTGRNT	190006	0.976026
SPEEDING	185340	0.952058
INATTENTIONIND	164868	0.846897
INTKEY	129603	0.665747
EXCEPTRSNCODE	109862	0.564341



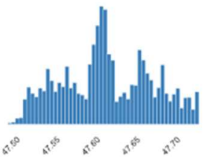
Now LATITUDE & LONGITUDE has 2.7 % missing value. Although I can use imputer to predict the missing values, it is very complicated. These missing values can’t be imputed by simple SK Learn Imputer. This is a geographical data and needs a geography-specific library to predict these missing values. But, the missing values % is very small, I decided to drop those missing values.

LATITUDE

Real number (ℝ≥0)

MISSING

Distinct	23839	Mean	47.61954252
Distinct (%)	12.6%	Minimum	47.49557292
Missing	5334	Maximum	47.73414158
Missing (%)	2.7%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	1.5 MiB

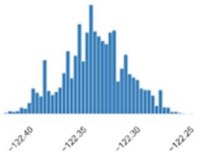


LONGITUDE

Real number (ℝ)

MISSING

Distinct	23563	Mean	-122.3305184
Distinct (%)	12.4%	Minimum	-122.4190911
Missing	5334	Maximum	-122.2389494
Missing (%)	2.7%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	1.5 MiB



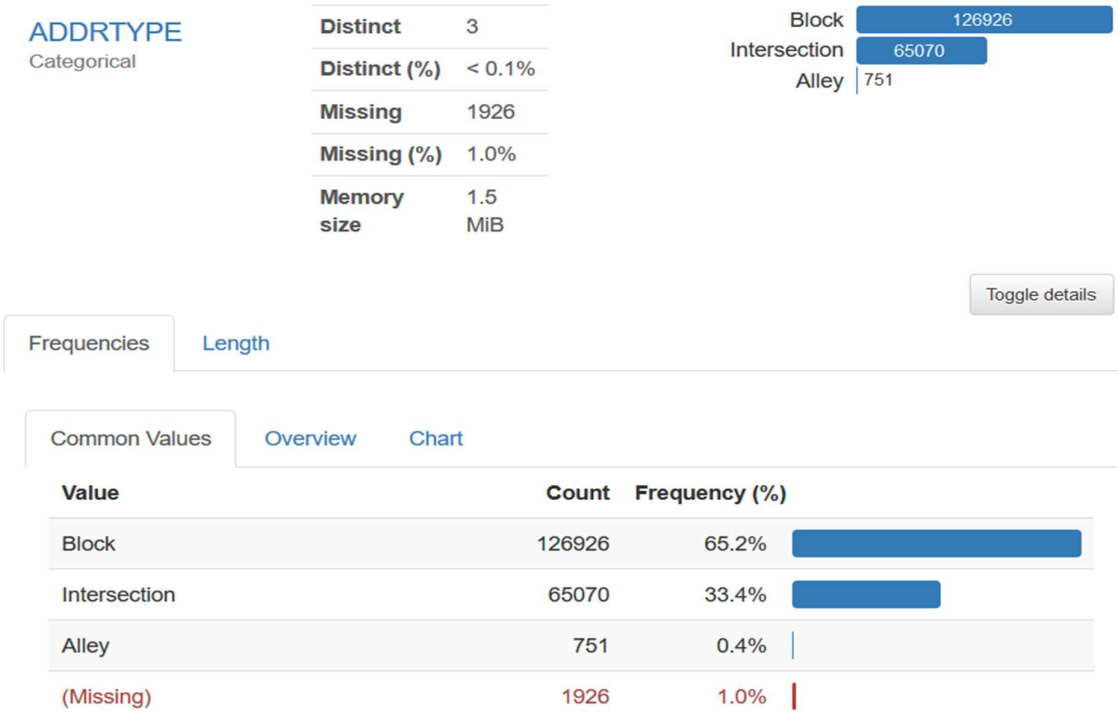
Now there are some attributes which have NO impact on our target variable (i.e. SEVERITYCODE). They are simply post-crash details recorded by the department. These attributes are not responsible to cause any crash and hence are not appropriate in predicting the severity of crashes. Such attributes are:

- OBJECT_ID
- INCKEY
- COLDETKEY
- REPORTNO
- STATUS
- SDOT_COLCODE
- ST_COLCODE
- SEGLANEKEY
- CROSSWALKKEY

I decided to drop all these attributes.

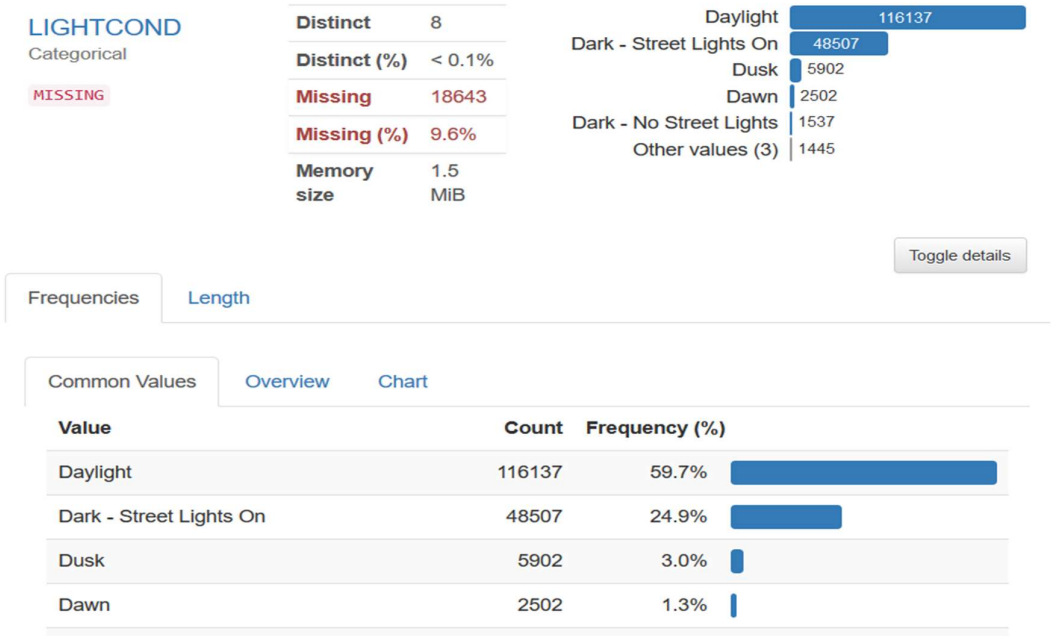
Handling Missing Values in remaining attributes

a. ADDRTYPE



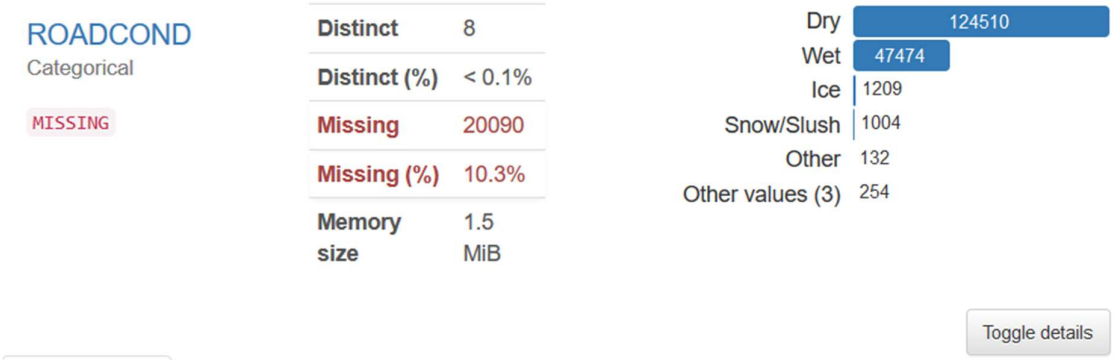
As 65% of crashes happened in Block, there is a high chance that the remaining 1% (missing values) also happened at Block. Therefore, I replaced NaN values by a label value 'Block'.

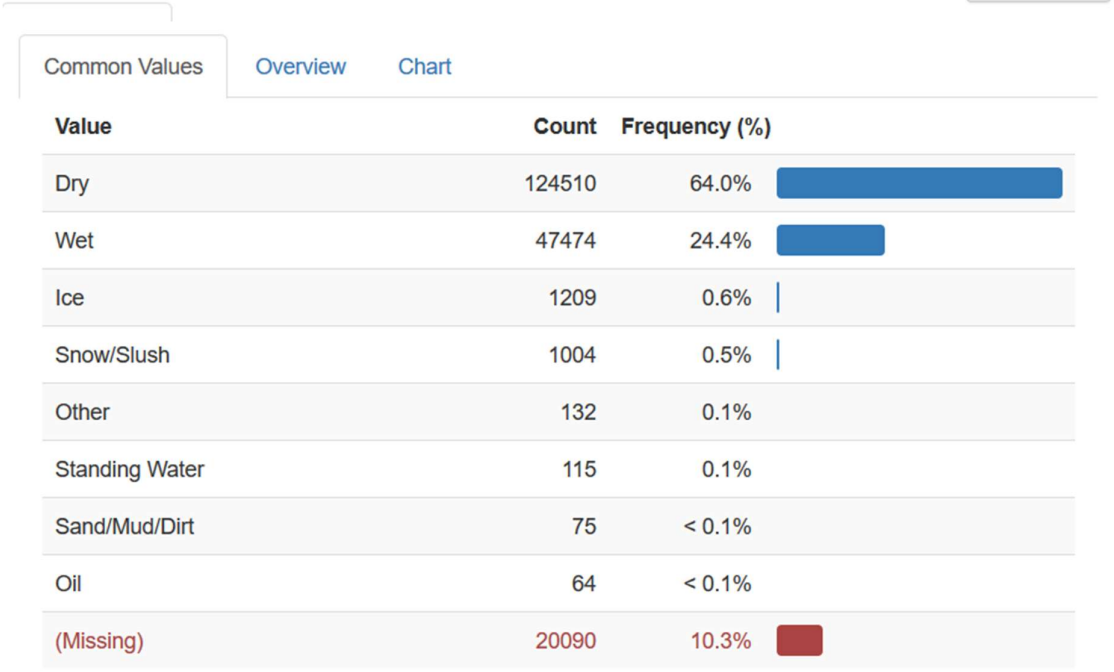
b. LIGHTCOND



Around 60% of accidents happened during daylight, the probability of those missing values being 'Daylight' is high. Hence, I replaced NaN values with the label "Daylight".

c. ROADCOND





Value	Count	Frequency (%)
Dry	124510	64.0%
Wet	47474	24.4%
Ice	1209	0.6%
Snow/Slush	1004	0.5%
Other	132	0.1%
Standing Water	115	0.1%
Sand/Mud/Dirt	75	< 0.1%
Oil	64	< 0.1%
(Missing)	20090	10.3%

In 64% of accidents, the condition of the road was 'Dry'. Hence, I decided to replace the NaN values with label "Dry".

d. WEATHER

Common Values | Overview | Chart

Value	Count	Frequency (%)
Clear	111135	57.1%
Raining	33145	17.0%
Overcast	27714	14.2%
Snowing	907	0.5%
Other	832	0.4%
Fog/Smog/Smoke	569	0.3%
Sleet/Hail/Freezing Rain	113	0.1%
Blowing Sand/Dirt	56	< 0.1%
Severe Crosswind	25	< 0.1%
Partly Cloudy	5	< 0.1%
(Missing)	20172	10.4%

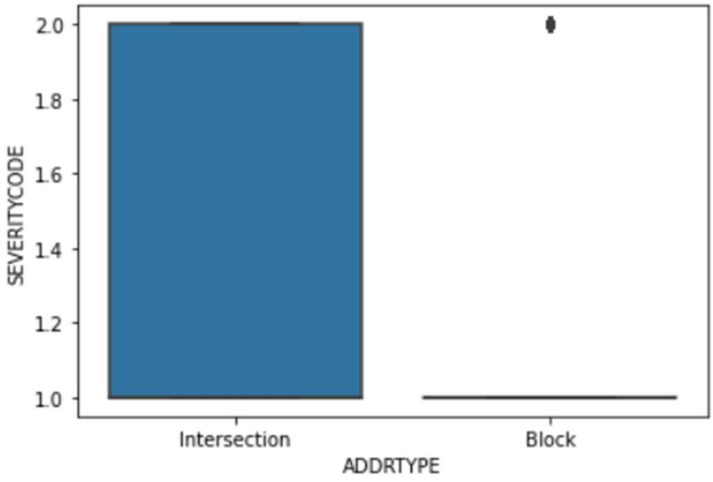
Around 57% of the cases happened during a clear day. Hence, I replaced NaN values with label ‘Clear’.

Methodology:

In this section we are going to discuss the relation of each input features with the target variable. Data Visualization is an important step in finding the meaningful insights from the data which otherwise is not possible in the tabular form of data. In this project, I have taken help of several libraries like Matplotlib, Seaborn, and Pandas profiling to visualise the data.

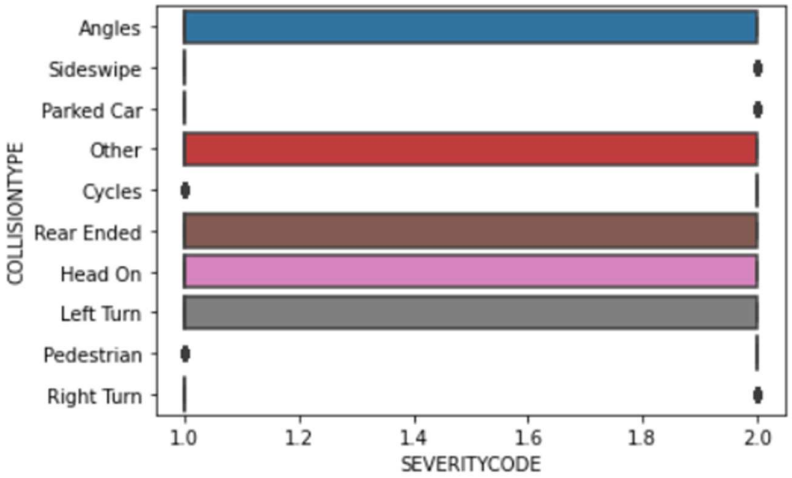
Let's start by analysing the relationship of each input attribute with the target variable.

1) **ADDRTYPE & SEVERITYCODE**



As we saw earlier that most crashes happened in 'Block' area (65.2%), but above boxplot makes it clear that almost all of those accidents were of severity level 1, except few outliers. Moreover, most of the accidents that happened in the 'Intersection' area belonged to severity level 2 which is more fatal. Also, there is no overlap between two variables. Hence it is a good predictor variable for our model.

2) **COLLISIONTYPE & SEVERITYCODE**



Above is a horizontal boxplot and they are useful when you have a greater number of categorical values to compare. It is evident from above figure that crashes were more severe (level 2) where type of collision was at 'Angles', 'Rear Ended', 'Head On', 'Left

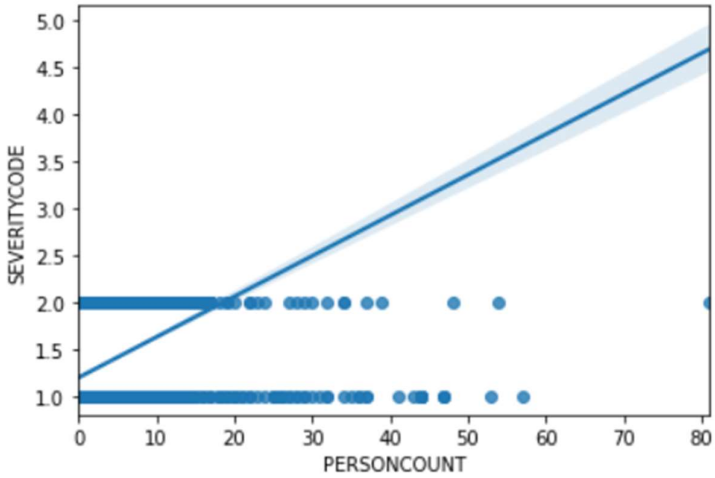
Turn' and 'Others'. Meanwhile, there were rare cases of severity 2 accidents when the collisions were of type 'Sideswipe', 'Parked Car', & 'Right Turn'. Similarly, there were collisions involving 'Cycle' and 'Pedestrian' that was of severity level 1. These are considered as Outliners.

To get more clear idea about the collision and severity level, I grouped the data frame by collision type as follows.

COLLISIONTYPE	SEVERITYCODE	LONGITUDE	LATITUDE	PERSONCOUNT	VEHCOUNT
Angles	48033	-4.215507e+06	1.641099e+06	93455	71540
Cycles	10090	-6.578954e+05	2.561449e+05	11444	5259
Head On	2859	-2.442923e+05	9.507742e+04	5450	4251
Left Turn	19006	-1.666626e+06	6.488239e+05	36837	27952
Other	28073	-2.725141e+06	1.060672e+06	40718	32467

It can be seen that collision at angles is proving to be more fatal with total number of 48033 cases with highest PERSONCOUNT of 93455.

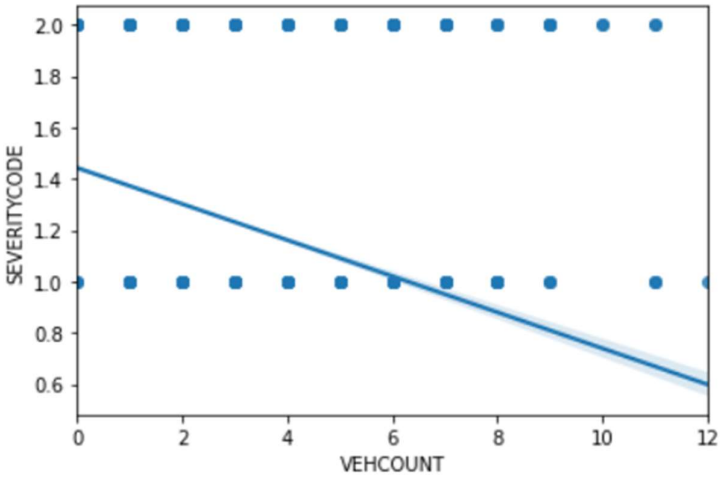
3) PERSONCOUNT & SEVERITYCODE



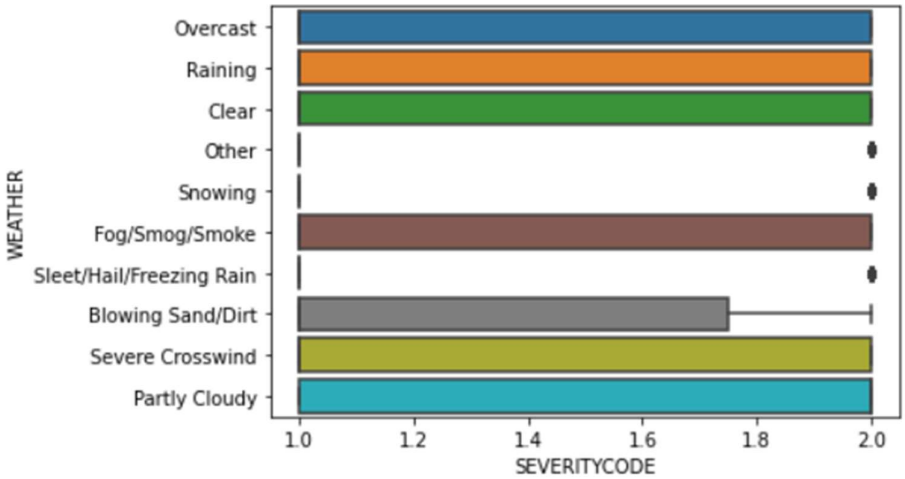
As expected, there is a direct linear relationship between the number of casualties and severity of crashes. Since the regression line have positive slope there exists a positive correlation between the two variables. Although, there is not much difference between the number of casualties in accidents of severity level 1 and 2.

4) VEHCOUNT & SEVERITYCODE

There is a negative correlation between two variables since slope of regression line is negative. Vehicle count is almost similar in both cases of severity. Interestingly the rate of increase in vehicle count is more in accidents of severity 1.



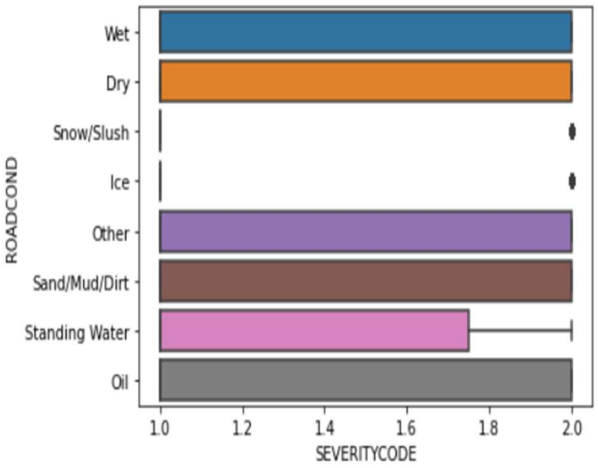
5) WEATHER & SEVERITYCODE

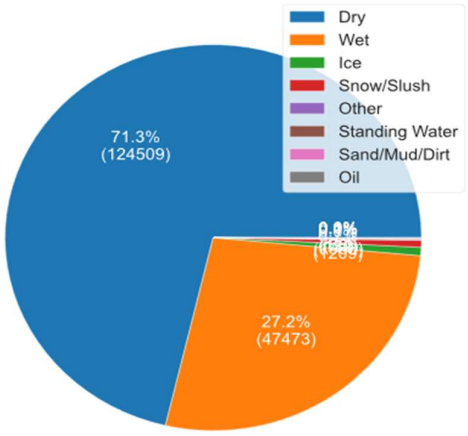


Above is the horizontal boxplot showing relation between WEATHER & SEVERITYCODE. It is evident that on weather conditions like Clear, Rainy, Overcast and Severe Crosswind, most of the crashes belongs to severity level 2. There are some extreme outliers for level 2 crashes for weather conditions like Snowing, Sleet/Hail/Freezing Rain and others. Otherwise, most of the accidents were not so severe for these climatic conditions.

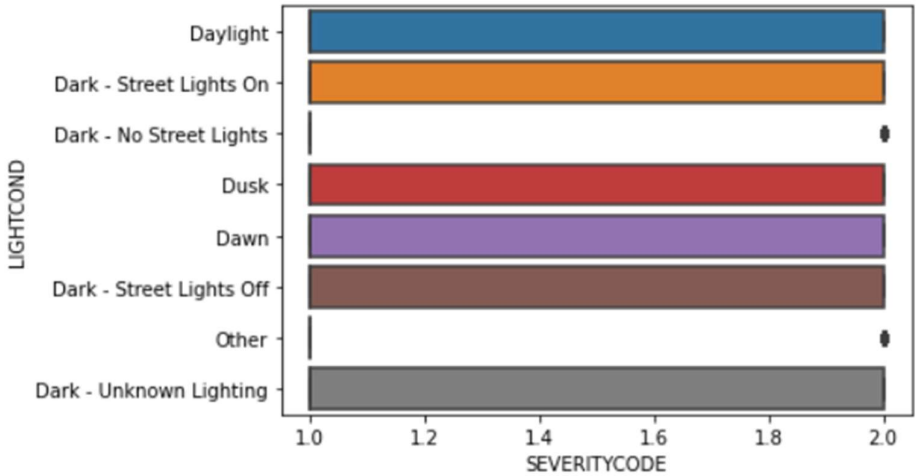
6) ROADCOND & SEVERITYCODE

As expected, accidents were more severe when the roads were wet, oily, covered with sand/mud/dirt and other situations. Surprisingly, severity was high even for Dry conditions. We also saw earlier that most of the accidents occurred during dry road conditions.



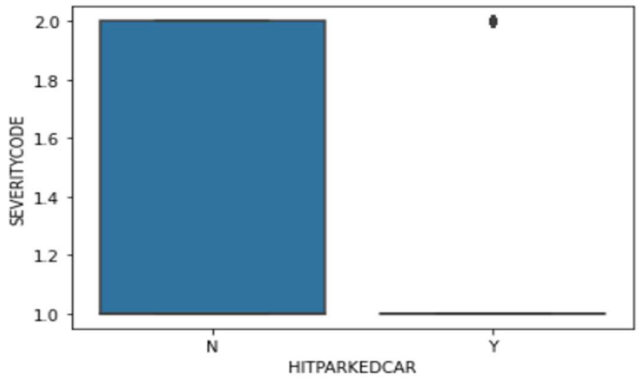


7) LIGHTCOND & SEVERITYCODE

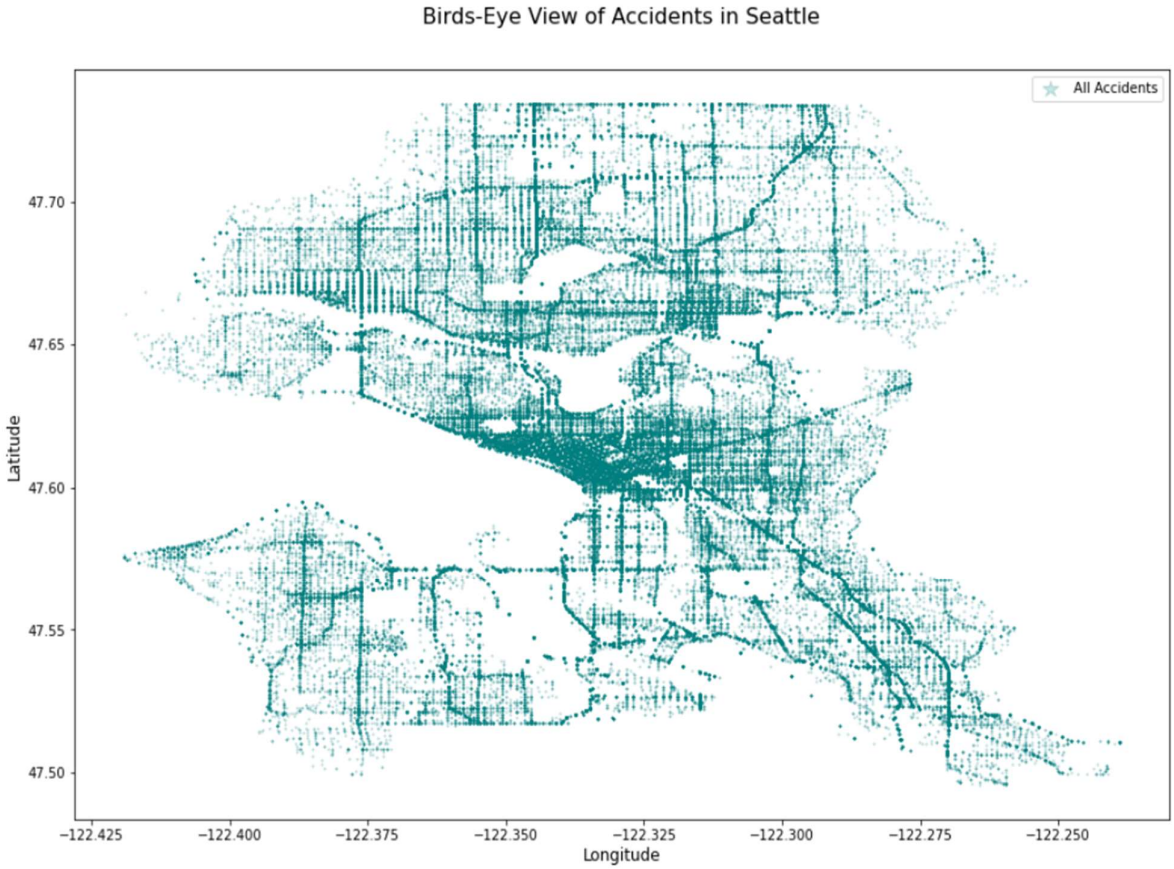


From above boxplot we can see that in the absence of proper lit environment, severity of accidents is high. For e.g. at conditions like Dusk, Dawn, Dark – Street lights off, Dark – Unknown Lightning the number of crashes are less but the probability of each crash being of high severity is quite high in these situations. Surprisingly, severe crashes also happened during Daylight furthermore, the number of accidents was also highest during daylight. There are some extreme outliers in situation like Dark – No Street Lights and others.

8) HITPARKEDCAR & SEVERITYCODE



9) Relation between Coordinates and Accidents



From the above picture it can be observed that the density of accident spots is more in latitude range 47.60 to 47.65 and longitude range of -122.350 to -122.300.

Exploratory Data Analysis (EDA)

After analysing the relation between each input feature and target variable we are ready to develop our model. But before proceeding to that we need to make some changes in our data. Since most of our input features are categorical, we need to transform them in to numeric values. Most of the libraries which offers machine learning algorithm works on numeric data so, transforming it is an important step.

Let's explore datatypes of each attribute.

```
df.dtypes
```

```
SEVERITYCODE       int64
LONGITUDE        float64
LATITUDE         float64
ADDRTYPE          object
COLLISIONTYPE     object
PERSONCOUNT        int64
VEHCOUNT           int64
INCDTTM           object
WEATHER           object
ROADCOND          object
LIGHTCOND         object
HITPARKEDCAR      object
dtype: object
```

Before transforming the data, first we need to covert the object datatype to string.

```
In [57]: # Converting the datatype from Object to String
         df['LIGHTCOND'] = df['LIGHTCOND'].astype("string")
         df['ROADCOND'] = df['ROADCOND'].astype("string")
         df['WEATHER'] = df['WEATHER'].astype("string")
         df['ADDRTYPE'] = df['ADDRTYPE'].astype("string")
         df['HITPARKEDCAR'] = df['HITPARKEDCAR'].astype("string")
         df['COLLISIONTYPE'] = df['COLLISIONTYPE'].astype("string")

In [58]: df.dtypes

Out[58]: SEVERITYCODE       int64
         LONGITUDE        float64
         LATITUDE         float64
         ADDRTYPE          string
         COLLISIONTYPE     string
         PERSONCOUNT        int64
         VEHCOUNT           int64
         INCDTTM           object
         WEATHER           string
         ROADCOND          string
         LIGHTCOND         string
         HITPARKEDCAR      string
         dtype: object
```

Now we are ready to transform the data. In this project we are using Label Encoder feature from Sklearn.preprocessing. The transformed data looks like in the figure given

below. Also note that I have separated all the input features and gave it label ‘X’ from the target variable which is labelled ‘y’.

```
# X is input features

X=df[['LATITUDE','LONGITUDE','ADDRTYPE','COLLISIONTYPE','PERSONCOUNT','VEHCOUNT','WEATHER','ROADCOND','LIGHTCOND','HITPA
X
```

	LATITUDE	LONGITUDE	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	WEATHER	ROADCOND	LIGHTCOND	HITPARKEDCAR
0	47.703140	-122.323148	1	0	2	2	4	7	5	0
1	47.647172	-122.347294	0	9	2	2	6	7	2	0
2	47.607871	-122.334540	0	5	4	3	4	0	5	0
3	47.604803	-122.334803	0	4	3	3	1	0	5	0
4	47.545739	-122.306426	1	0	2	2	6	7	5	0
...	...	...	...	...	...	...	...	...	...	...
184577	47.565408	-122.290826	0	2	3	2	1	0	5	0
184578	47.690924	-122.344526	0	7	2	2	6	7	5	0
184579	47.683047	-122.306689	1	3	3	2	1	0	5	0
184580	47.678734	-122.355317	1	1	2	1	1	0	6	0
184581	47.611017	-122.289360	0	7	2	2	1	7	5	0

184582 rows × 10 columns

```
y=df['SEVERITYCODE']
```

At last we will do normalization and standardization of the data. If the dataset contains the data of varied ranges it might affect the performance and accuracy of the model. Therefore, it is always best practice to do the normalization before fitting it into model. Here we are using StandardScaler().fit().transfrom() from Sklearn.

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

```
array([[ 1.48739777,  0.24501881,  1.38540755, -1.60088434, -0.34423279,
         0.04747219,  0.80212158,  1.71821177,  0.57629845, -0.19415094],
       [ 0.49174574, -0.55927011, -0.72180926,  1.6151249 , -0.34423279,
         0.04747219,  1.78209098,  1.71821177, -1.54999725, -0.19415094],
       [-0.20740912, -0.13443019, -0.72180926,  0.18578746,  1.12835405,
         1.84491063,  0.80212158, -0.58893996,  0.57629845, -0.19415094],
       [-0.26199758, -0.1432006 , -0.72180926, -0.1715469 ,  0.39206063,
         1.84491063, -0.66783251, -0.58893996,  0.57629845, -0.19415094],
       [-1.31271947,  0.80202434,  1.38540755, -1.60088434, -0.34423279,
         0.04747219,  1.78209098,  1.71821177,  0.57629845, -0.19415094]])
```

As a last step in our EDA, we are going to split our dataset into training and testing data. The purpose of any model is to predict the output for unknown input. If we use the whole dataset to train the model and then use the same data for accuracy evaluation (in-sample accuracy), that will not be the true test of our model. Out-of-Sample Accuracy is more crucial practically and it is evaluated using testing data.

```
: X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=1)
  print ('Train set:', X_train.shape,  y_train.shape)
  print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (147665, 10) (147665,)
Test set: (36917, 10) (36917,)
```

Model Development & Evaluation

At a fundamental level, there are two types of Supervised Machine Learning Algorithm.

- Regression Model -Used to predict the continuous values.
- Classification Model - Used to predict the categorical values.

Reason: Since our target variable (SEVERITYCODE) is a categorical variable i.e. it has two values '1' and '2', our problem falls to the category of classification model. We are going to use two machine learning algorithms to predict the value of target variable.

- Logistic Regression
- Decision Tree Model

Finally, we will find out-of-sample accuracy for both models using test data and compare the accuracy of both in the result section.

Logistic Regression:

Unlike Linear Regression which is used to predict the continuous value, Logistic regression is mainly used to handle classification problem.

Let's create a logical regression object.

```
# Fitting the model using Logistic Regression object, LR
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

```
LogisticRegression(C=0.01, solver='liblinear')
```

Predicting the target variable (yhat) using test data and its probability.

```
# Predicting the target variable using Test Data
yhat = LR.predict(X_test)
yhat
```

```
array([2, 1, 1, ..., 1, 1, 1], dtype=int64)
```

predict_proba returns estimates for all classes, ordered by the label of classes. So, the first column is the probability of class 1, P(Y=1|X), and second column is probability of class 0, P(Y=0|X):

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

```
array([[0.47016722, 0.52983278],
       [0.78106286, 0.21893714],
       [0.74269633, 0.25730367],
       ...,
       [0.92957679, 0.07042321],
       [0.59542303, 0.40457697],
       [0.75388773, 0.24611227]])
```

Decision Tree Model:

Let's create a Decision Tree Classifier Instance and predict the target variable using test data.

```
# Creating a Decision Tree Classifier Instance
severity = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
severity.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
# Predicting the target variable using Test Data
pred_severity = severity.predict(X_test)
pred_severity
```

```
array([2, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Results:

In multilabel classification, **accuracy classification score** is a function that computes subset accuracy. This function is equal to the jaccard_similarity_score function. Essentially, it calculates how closely the actual labels and predicted labels are matched in the test set. 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.

Another way of looking at accuracy of classifier is to look at confusion matrix.

Based on the count of each section, we can calculate precision and recall of each label:

- **Precision** is a measure of the accuracy provided that a class label has been predicted. It is defined by: precision = TP / (TP + FP)
- **Recall** is true positive rate. It is defined as: Recall = TP / (TP + FN)

So, we can calculate precision and recall of each class.

F1 score: Now we are in the position to calculate the F1 scores for each label based on the precision and recall of that label.

The F1 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 classifer has a good value for both recall and precision.

And finally, we can tell the average accuracy for this classifier is the average of the F1-score for both labels, which is 0.72 in our case.

Evaluation of Logistic Regression Model

```
print('Jaccard Smimilarity Score is: ',jaccard_score(y_test, yhat))
```

```
Jaccard Smimilarity Score is:  0.7008705052344105
```

```
# Creating a Confusion Matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
np.set_printoptions(precision=2)
```

```
print(classification_report(y_test, yhat))
```

```
              precision    recall  f1-score   support

           1       0.73      0.95      0.82     25828
           2       0.60      0.16      0.25     11089

    accuracy                           0.72     36917
   macro avg       0.66      0.56      0.54     36917
weighted avg       0.69      0.72      0.65     36917
```

```
# Calculating Log Loss
print('Log Loss is: ',log_loss(y_test, yhat_prob))
```

```
Log Loss is:  0.570318380953288
```

Evaluation of Decision Tree Model

```
: # Evaluating the model using F1 Score and Jaccard Similarity Score
  d=f1_score(y_test,pred_severity, average='weighted')
  print("F1 Score is: ",d)
  print("DecisionTree's Jaccard Smilarity Score Score is: ", jaccard_score(y_test, pred_severity))
```

```
F1 Score is:  0.7201209036124941
DecisionTree's Jaccard Smilarity Score Score is:  0.7214827752348831
```

The results we got above are very promising. The model accuracy is around 72% with high F1 and jaccard score.

Discussion:

With above accuracy, we are ready for prediction of severity of the accidents. This model can also help us in finding the key factors and situations that led to accidents. For example, we can find out answer to following crucial questions:

- Which areas in Seattle are more prone to severe collisions?
- What road and light conditions are responsible for accidents?
- Up to what extent weather can have impact on severity of crashes?

Answer to these questions will help the Transportation Department to take appropriate measures to counter act the underlying cause of crashes and minimize the human and property loss in the city.

Although, we able to get high accuracy, we can further increase it by applying more complicated algorithms to develop the model but that requires advanced computation power. In addition to that, structuring the dataset with the help of someone who have more domain specific knowledge will give a great boost to the accuracy of model. Moreover, it will be cheery on the cake if the sources of the dataset will record and update the data frequently. Large percentage of missing values was a great hindrance in our project and its accuracy. Although, we handled it, it would be even better if the dataset is more accurate with its records.

Conclusion:

We developed the Logistic Regression and Decision Tree models for prediction by identifying the key input features depending on their relation with and impact on target variable. In this project we tried to solve the given problem using two classification models. It is now ready to be deployed and predict the severity of accidents provided the sufficient input. I hope that the Seattle Department of Transportation will get sufficient help from this project to overcome the problem of fatal crashes. This project was a step in the direction of keeping the citizens safe from accidents so that they can return to their homes safe and sound where their family is waiting for them.