Capstone Project - Car Accident Severity

Introduction | Business Undertanding

The Seattle government is going to prevent avoidable car accidents by employing methods that alert drivers, health system, and police to remind them to be more careful in critical situations.

In order to reduce the frequency of car collisions in a community, an algorithim must be developed to predict the severity of an accident given the current weather, road and visibility conditions. When conditions are bad, this model will alert drivers to remind them to be more careful.In most cases, not paying enough attention during driving, abusing drugs and alcohol or driving at very high speed are the main causes of occurring accidents that can be prevented by enacting harsher regulations. Besides the aforementioned reasons, weather, visibility, or road conditions are the major uncontrollable factors that can be prevented by revealing hidden patterns in the data and announcing warning to the local government, police and drivers on the targeted roads.

The target audience of the project is local Seattle government, police, rescue groups, and last but not least, car insurance institutes. The model and its results are going to provide some advice for the target audience to make insightful decisions for reducing the number of accidents and injuries for the city.

Data Understanding

Our predictor or target variable will be 'SEVERITYCODE' because it is used measure the severity of an accident from 0 to 5 within the dataset. Attributes used to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'. The data was collected by the Seattle Police Department and Accident Traffic Records Department from 2004 to present.

The data consists of 37 independent variables and 194,673 rows. The dependent variable, “SEVERITYCODE”, contains numbers that correspond to different levels of severity caused by an accident from 0 to 4.

Severity codes are as follows:

1. **Property Damage Only Collision**
2. **Injury Collision**
   </b>

streaming_body_1 = client_3b4d19ae96ba43e3aa551fa97b61f7f9.get_object(Bucket='courseraproject-donotdelete-pr-rtsjyaxd5muifd', Key='Metadata.pdf')['Body'] if not hasattr(streaming_body_1, "iter"): streaming_body_1.iter = types.MethodType(iter, streaming_body_1)

Data processing

In the original form, this data is not fit for analysis. Data contain NULL value and also need to be cleaning, converting in to usable data
After analyzing the data set, I have decided to focus on only four features:

1. SEVERITYCODE

2. WEATHER

3. ROADCOND

4. LIGHTCOND

To get a good understanding of the dataset, let's start import and process our data!!!

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/IPython/core/interactive
shell.py:3020: DtypeWarning: Columns (33) have mixed types. Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STA
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matc
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matc
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matc
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matc
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matc

5 rows × 38 columns

Out[31]:

	SEVERITYCODE
1	136485
2	58188

As mentioned in "Data Understanding", Our target variable will be SEVERITYCODE .
Other attributes used to weigh the severity of an accident are
'ADDRTYPE', JUNCTIONTYPE', 'WEATHER', 'ROADCOND' and 'LIGHTCOND'
.and for exploratory data analysis I wanted to choose some of numerical variables such as (PERSONCOUNT, VEHCOUNT, PEDCOUNT, PEDCYLCOUNT)

Therefore considering below relevant features by dropping others to simplify the dataset.

1. **'X'**
2. **'Y'**
3. **'COLDETKEY'**
4. **'REPORTNO'**
5. **'INTKEY'**
6. **'LOCATION'**
7. **'EXCEPTRSNCODE'**
8. **'EXCEPTRSNDESC'**
9. **'SEVERITYCODE.1'**
10. **'SEVERITYDESC'**
11. **'INCDATE'**
12. **'INCDTTM'**
13. **'SDOT_COLCODE'**
14. **'SDOT_COLDESC'**
15. **'INATTENTIONIND'**
16. **'UNDERINFL'**
17. **'PEDROWNOTGRNT'**
18. **'SDOTCOLNUM',**
19. **'SPEEDING'**
20. **'ST_COLCODE'**
21. **'ST_COLDESC'**
22. **'SEGLANEKEY'**
23. **'CROSSWALKKEY'**
24. **'HITPARKEDCAR'**
25. **'PERSONCOUNT'**
26. **'PEDCOUNT'**
27. **'PEDCYLCOUNT'**
28. **'VEHCOUNT'**
29. **'OBJECTID'**
30. **'COLLISIONTYPE'**
31. **'STATUS'**

Out[4]:
```
(194673, 12)
```

Out[30]:

	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	SEVERITYDESC
0	2	2	0	0	2	Injury Collision
1	1	2	0	0	2	Property Damage Only Collision
2	1	4	0	0	3	Property Damage Only Collision
3	1	3	0	0	3	Property Damage Only Collision
4	2	2	0	0	2	Injury Collision

Data selected also contained Missing Value so we need to fulfil these cell

Out[5]:

```
ADDRTYPE        0
JUNCTIONTYPE    0
WEATHER         0
ROADCOND        0
dtype: int64
```

Then, I began choosing columns to use from the dataframe that I created. The columns that I chose were SEVERITYCODE, which assigns a crash a value of 1, which means no injury, and 2, indicating injury, COLLISIONTYPE, which describes the type of crash, WEATHER, which describes the weather at the time of crash, ROADCOND, which describes the condition of the road at the time of crash, LIGHTCOND, which describes the light conditions at the time of crash.

We can see there are only 2 types of accident list in this data are:

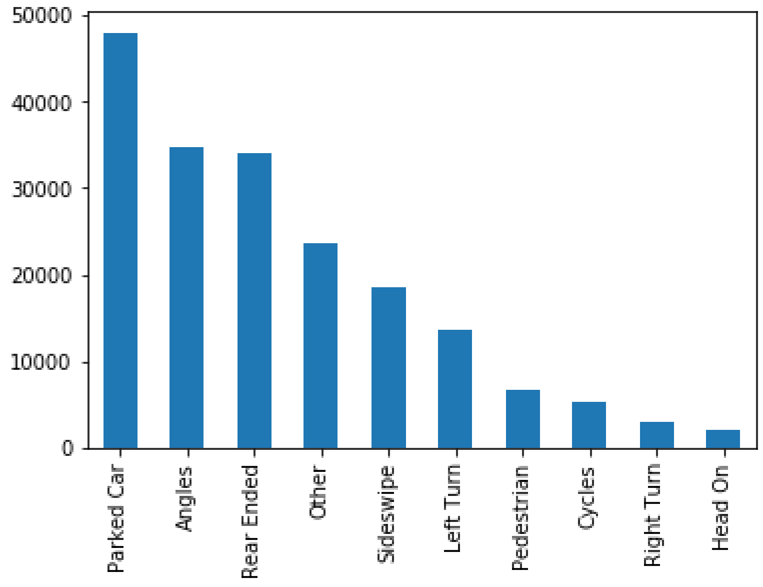
1. Property Damage Only Collision

2. Injurry Collisions

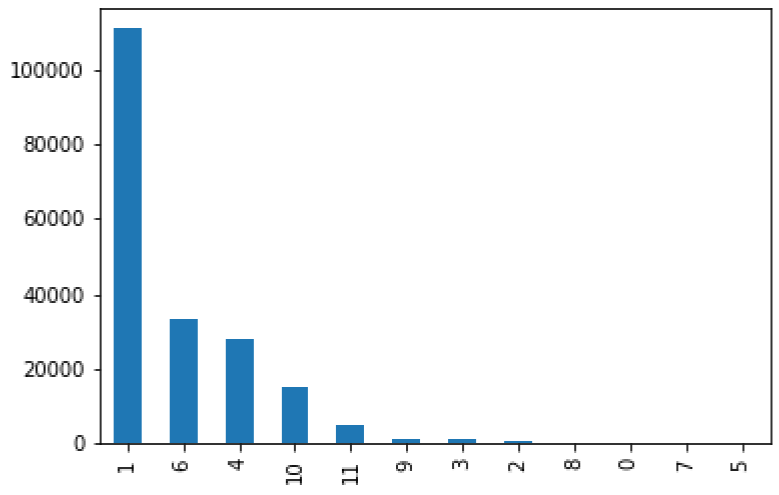
and the percentage of Injurry Collision to Property Damage Only Collision is around : ~42%

Graph to visualize selected Attributes & Corellation other attributes

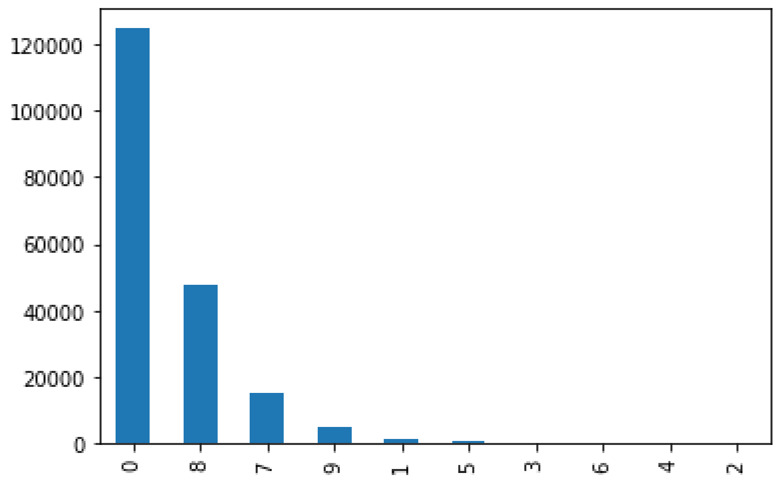
```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f108dd06f28>
```



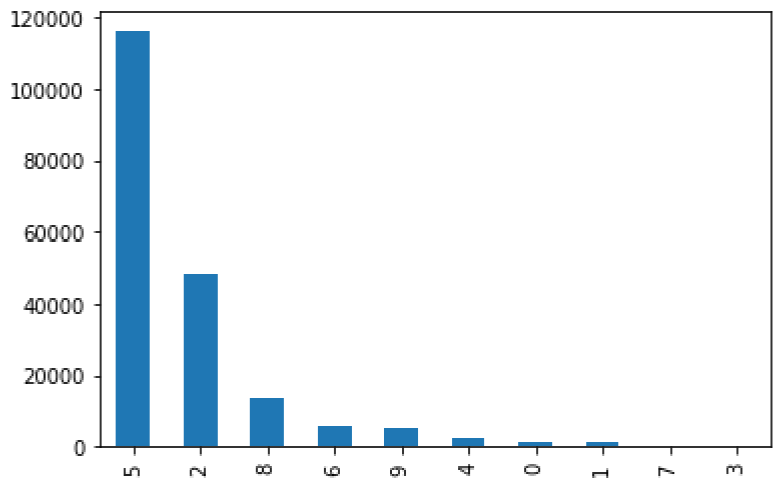
```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f10870385c0>
```



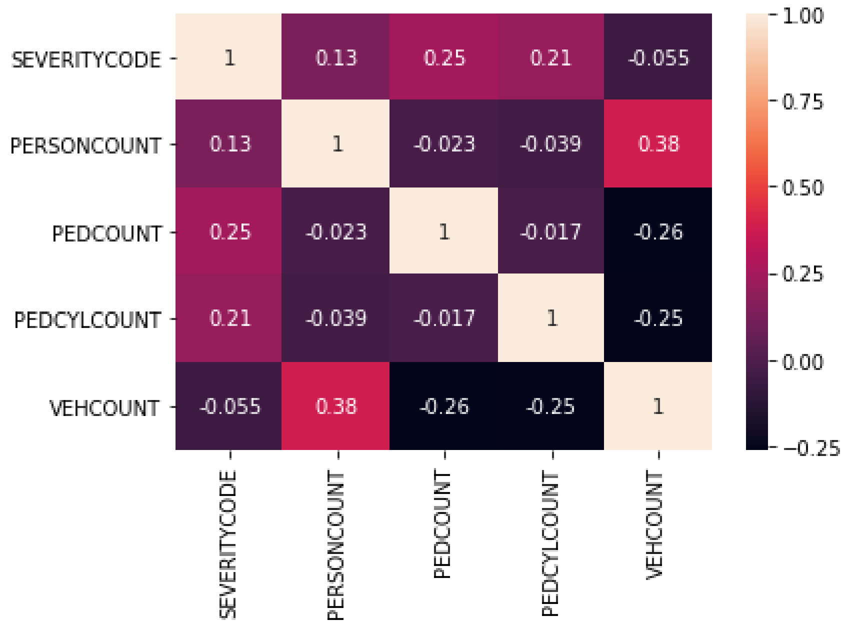
```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f108da7ddd8>
```



```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1086fc0c50>
```



```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1086edb588>
```



	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT
SEVERITYCODE	1	0.13	0.25	0.21	-0.055
PERSONCOUNT	0.13	1	-0.023	-0.039	0.38
PEDCOUNT	0.25	-0.023	1	-0.017	-0.26
PEDCYLCOUNT	0.21	-0.039	-0.017	1	-0.25
VEHCOUNT	-0.055	0.38	-0.26	-0.25	1

With the below correlation matrix, on numerical variables we can say that there is no much correlation between any of these variables

Methodology

For implementing the solution, I have used Github as a repository and running Jupyter Notebook to preprocess data and build Machine Learning models. Regarding coding, I have used Python and its popular packages such as Pandas, NumPy and Sklearn.
Once I have load data into Pandas Dataframe, used 'dtypes' attribute to check the feature names and their data types. Then I have selected the most important features to predict the severity of accidents in Seattle. Among all the features, the following features have the most influence in the accuracy of the predictions:

1. `"ADDRTYPE"`
2. `“JUNCTIONTYPE”,`
3. `“ROADCOND”,`
4. `“LIGHTCOND”`
5. `"WEATHER"`

</b>
So before started we have to standardizing these features

After convert data value to usable, we can now use this data in our analysis and ML models!

Out[29]:

	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	SEVERITYDESC
0	2	2	0	0	2	Injury Collision
1	1	2	0	0	2	Property Damage Only Collision
2	1	4	0	0	3	Property Damage Only Collision
3	1	3	0	0	3	Property Damage Only Collision
4	2	2	0	0	2	Injury Collision

I have employed three machine learning models:

1.K Nearest Neighbour (KNN)

2. Decision Tree

3. Logistic Regression

Let's get started!

K-Nearest Neighbor (KNN

KNN will help us predict the severity code of an outcome by finding the most similar to data point within k distance.
Here while implementing KNN algorithm, I have taken it for set of 15 values the model will be trained on training set of data, and then predicting the values based on test data set, Atlast the model accuracy will be calculated by comparing actual values with predicted values of test data set.

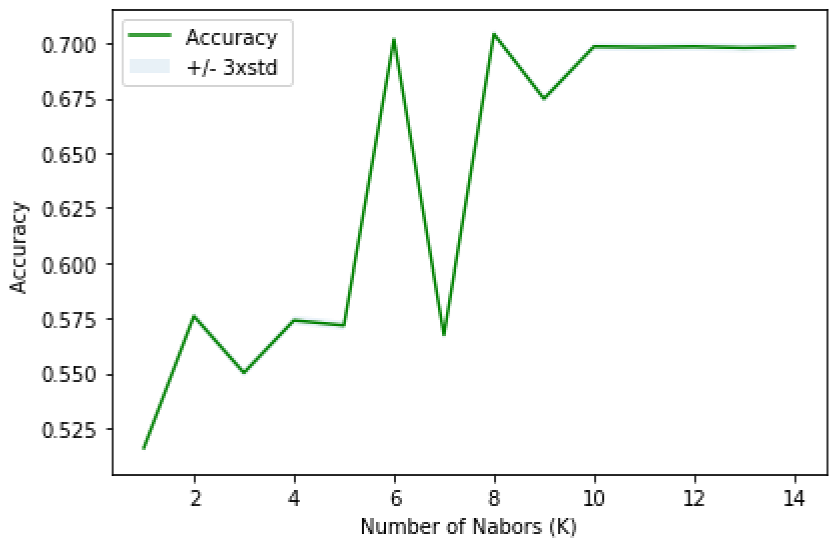
Out[7]:
```
array([[4, 8, 5],
       [6, 8, 2],
       [4, 0, 5],
       [1, 0, 5],
       [6, 8, 5]])
```

Out[8]:
```
array([2, 1, 1, 1, 2])
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validatio
n.py:595: DataConversionWarning: Data with input dtype int64 was converted to 
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validatio
n.py:595: DataConversionWarning: Data with input dtype int64 was converted to 
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

Out[9]:
```
array([[ 0.22981187,  1.39847224,  0.25900713],
       [ 0.87758556,  1.39847224, -1.36653782],
       [ 0.22981187, -0.73846749,  0.25900713],
       [-0.74184867, -0.73846749,  0.25900713],
       [ 0.87758556,  1.39847224,  0.25900713]])
```

```
Train set (136271, 3) (136271,)
Test set: (58402, 3) (58402,)
```



```
The best accuracy of KNN is  0.7042224581349954 , k= 8
```

Decision Tree

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

```
The best accuracy of DT is  0.6994109790760591 with a max depth of 6
```

Logistic Regression

Because our dataset only provides us with two severity code outcomes, our model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

```
Train set: (155738, 3) (155738,)
Test set: (38935, 3) (38935,)
```

```
Out[15]: LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                   tol=0.0001, verbose=0, warm_start=False)
```

```
Out[16]: array([[0.72788689, 0.27211311],
                [0.67683024, 0.32316976],
                [0.65535907, 0.34464093],
                ...,
                [0.67283962, 0.32716038],
                [0.74802079, 0.25197921],
                [0.72788689, 0.27211311]])
```

```
Out[17]: 0.5986563145341074
```

Result

Model Evaluation - Finding accuracy of the Model

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/svm/base.py:196: 
FutureWarning: The default value of gamma will change from 'auto' to 'scale' 
in version 0.22 to account better for unscaled features. Set gamma explicitly 
to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
KNN F1-score: 0.58
KNN Jaccard Score: 0.70
Decision Tree F1-score: 0.58
Decision Tree Jaccard Score: 0.70
SVM F1-score: 0.58
SVM Jaccard score: 0.70
LogLoss: : 0.60
LOG F1-score: 0.5779
LOG Jaccard score: 0.7011
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/metrics/classifi
cation.py:1143: UndefinedMetricWarning: F-score is ill-defined and being set 
to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/metrics/classifi
cation.py:1143: UndefinedMetricWarning: F-score is ill-defined and being set 
to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Once we analysed and cleaned the data, it was then fed through three ML models; K-Nearest Neighbour, Decision Tree and Logistic Regression. 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 logloss for logistic regression. Choosing different k values helped to improve our accuracy to be the best possible.

The final results of the model evaluations are summarized as below

	Algorithm	Jaccard	F1-score	Logloss
1	KNN	0.70	0.58	NA
2	Decision Tree	0.70	0.58	NA
2	Logistic Regression	0.7011	0.577	NA

Discussion and Conclusion

<u>I have got a decent accuracy value for all classification algorithms. So the best classifier of this problem are Decision Tree, Logistic and KNN based on their accuracy value.</u>

<u>By revealing hidden patterns in predicting severity in accidents based on the features Weather, Road and Light conditions, addresstype, junctiontype have significant impact on whether to travel or not which often result in injury and property damage.</u>

<u>Based on historical data from weather conditions pointing to certain classes, we can conclude that particular weather conditions have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2).</u>

<u>Thank you for reading!</u>