Predicting impact of driving conditions on accident severity

Vidur Channa

Introduction

- Traffic accidents result in severe fatalities across the world and are the leading cause of death for people aged 15-29. Weather patterns can have a dramatic impact on the safety of driving in specific regions, due to limited visibility and reduced grip from tyres.
- When people start planning their trips between places, they tend to check their route first and see estimated time to arrival, but often do not see any warnings for weather or increased accident risks on certain roads. Due to this, drivers might end up taking a dangerous route and encounter treacherous road surfaces and end up in an accident.
- The purpose of this project is to model these dangerous driving conditions and predict regions where accidents are most likely to be severe. This will be done to improve road safety by warning people and allowing them to postpone travel or choose safer routes to get to their destinations. The data obtained from this model will be delivered to the Seattle government and police as well, so they can remain on high alert in regions predicted to be high risk for severe crashes.

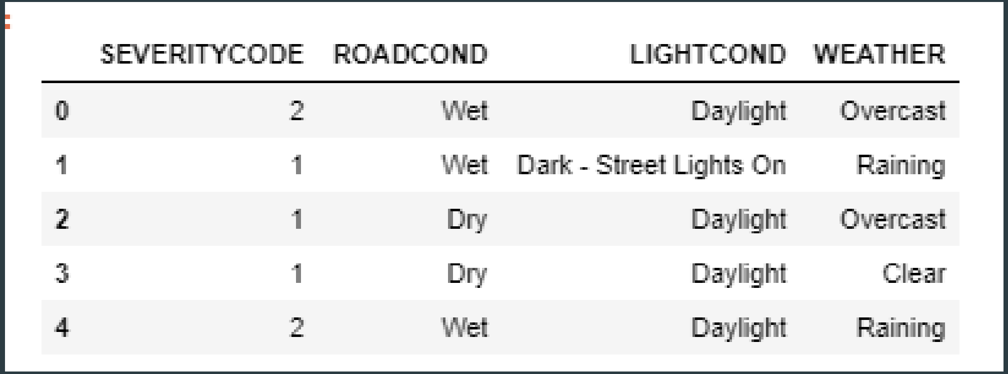
Data Description

- The Seattle SPOT Traffic Management Division has a dataset containing information regarding collisions from 2004 to 2020 with several key features surrounding severity and weather. The ‘SEVERITYCODE’ column is going to be the target variable which the model will predict to warn drivers of dangerous areas. The variable is effectively a binary class which classifies accidents into either ‘property damage’ or ‘injury collision’. There are classifiers for ‘severe injury’ and ‘fatality’, but no data for those metrics is present within the data set hence they cannot be modeled.

- The variables used to train the model will be ‘WEATHER’, ‘LIGHTCOND’ and ‘ROADCOND’ as they pertain directly to the conditions the drivers were in at the time of their accidents. The three variables are all TEXT data types and can hence be classified using frequency counts by processing the data.

Data Cleaning

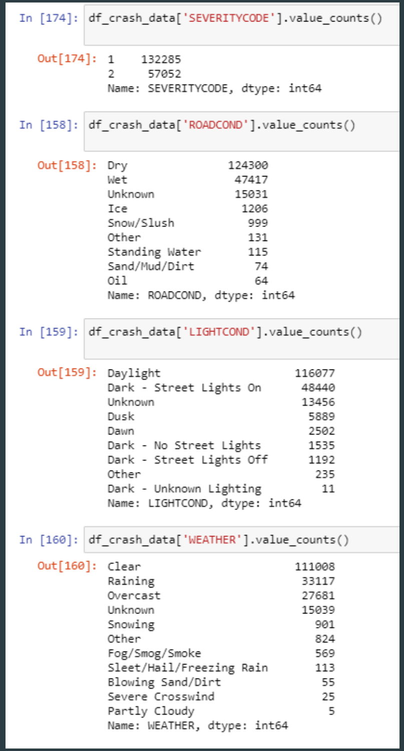
- The DataFrame with only relevant metrics was created and then flushed of null values
- Null values were removed as they only comprised a small percentage of the total dataset
- Replacing them with average values would not have worked as the value fields are all discrete and categorical



	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER
0	2	Wet	Daylight	Overcast
1	1	Wet	Dark - Street Lights On	Raining
2	1	Dry	Daylight	Overcast
3	1	Dry	Daylight	Clear
4	2	Wet	Daylight	Raining

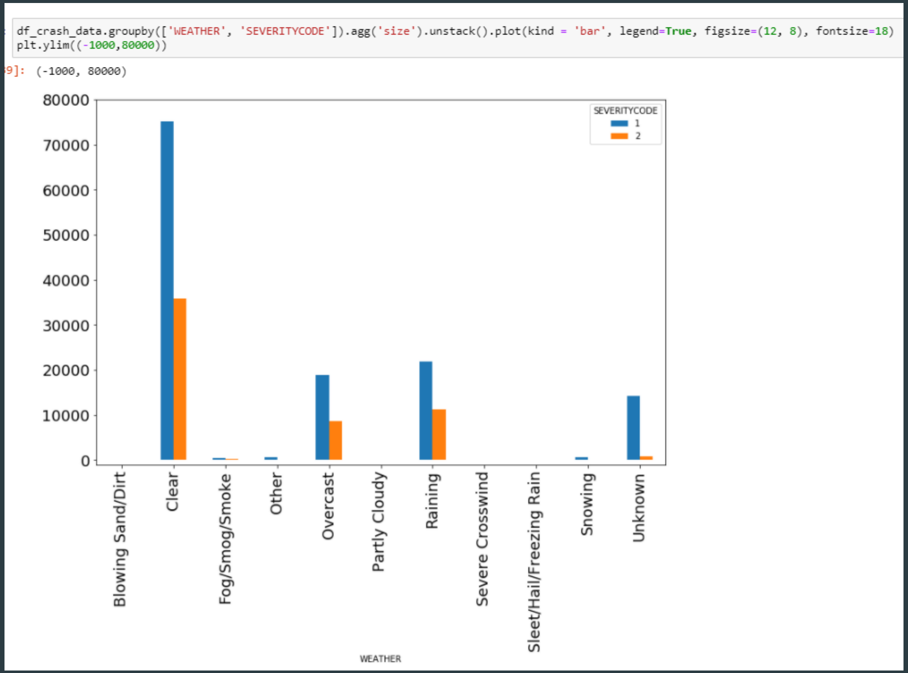
Data Exploration

- To analyze the data further, we must also gain an understanding of what the leading metrics are for each categorical variable
- This can be done by using a combination of basic statistics and some visualizations using matplotlib
- The base statistics are displayed below with clear leaders in each category:

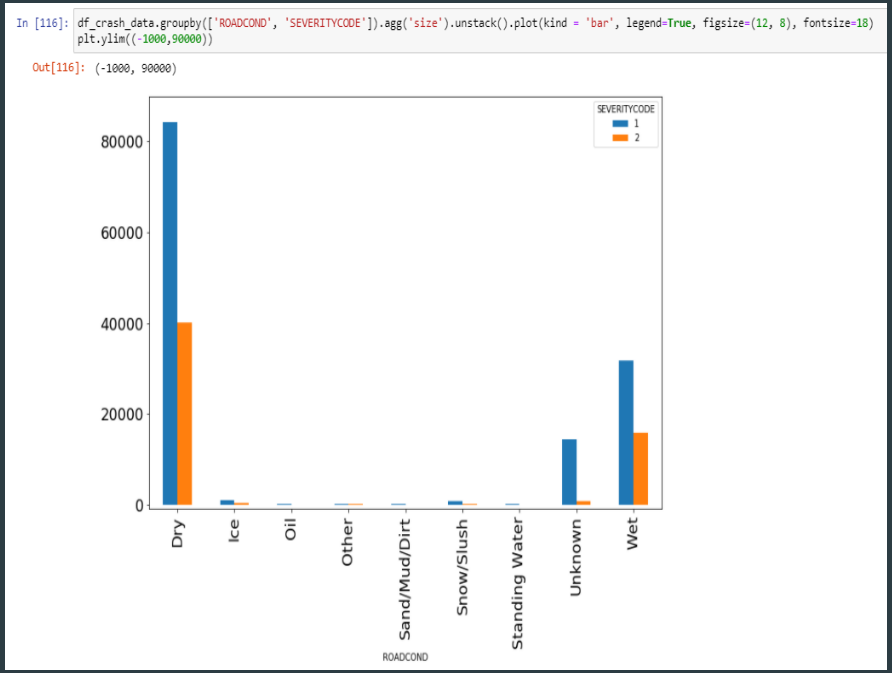


```
In [174]: df_crash_data['SEVERITYCODE'].value_counts()

Out[174]: 1    132285
          2     57052
          Name: SEVERITYCODE, dtype: int64

In [158]: df_crash_data['ROADCOND'].value_counts()

Out[158]: Dry               124300
          Wet                47417
          Unknown            15031
          Ice                 1206
          Snow/Slush           999
          Other                131
          Standing Water       115
          Sand/Mud/Dirt         74
          Oil                   64
          Name: ROADCOND, dtype: int64

In [159]: df_crash_data['LIGHTCOND'].value_counts()

Out[159]: Daylight                    116077
          Dark - Street Lights On      48440
          Unknown                      13456
          Dusk                          5889
          Dawn                          2502
          Dark - No Street Lights       1535
          Dark - Street Lights Off      1192
          Other                          235
          Dark - Unknown Lighting         11
          Name: LIGHTCOND, dtype: int64

In [160]: df_crash_data['WEATHER'].value_counts()

Out[160]: Clear                       111008
          Raining                      33117
          Overcast                     27681
          Unknown                      15039
          Snowing                        901
          Other                          824
          Fog/Smog/Smoke                 569
          Sleet/Hail/Freezing Rain       113
          Blowing Sand/Dirt               55
          Severe Crosswind                25
          Partly Cloudy                    5
          Name: WEATHER, dtype: int64
```

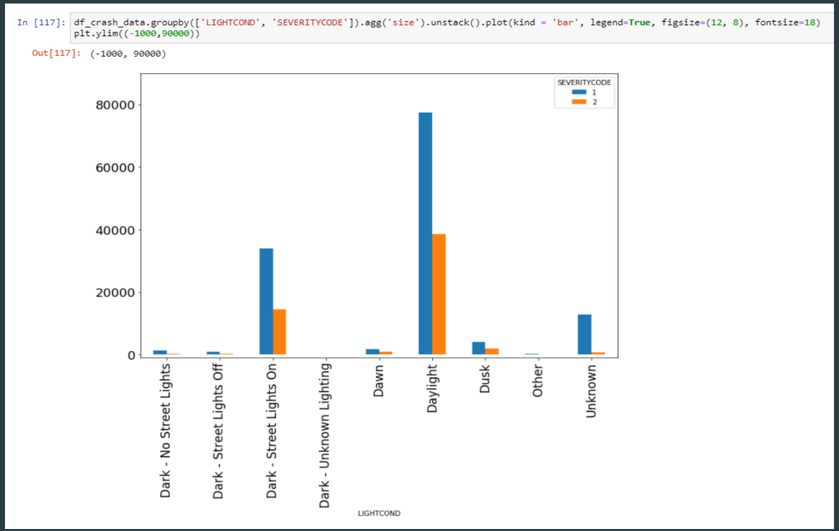
Data Exploration – Visualizations.



```
: df_crash_data.groupby(['WEATHER', 'SEVERITYCODE']).agg('size').unstack().plot(kind = 'bar', legend=True, figsize=(12, 8), fontsize=18)
  plt.ylim((-1000,80000))
9]: (-1000, 80000)
```



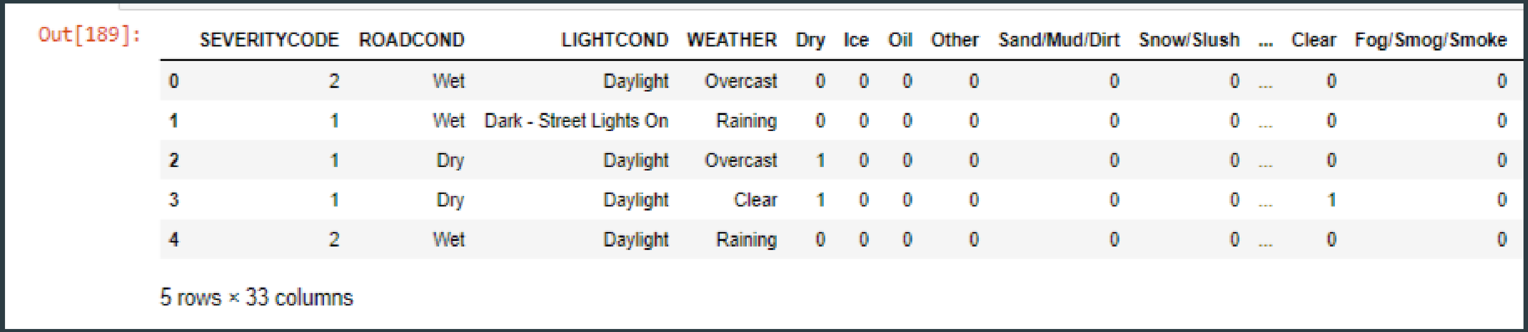
```
In [116]: df_crash_data.groupby(['ROADCOND', 'SEVERITYCODE']).agg('size').unstack().plot(kind = 'bar', legend=True, figsize=(12, 8), fontsize=18)
          plt.ylim((-1000,90000))
Out[116]: (-1000, 90000)
```



```
In [117]: df_crash_data.groupby(['LIGHTCOND', 'SEVERITYCODE']).agg('size').unstack().plot(kind = 'bar', legend=True, figsize=(12, 8), fontsize=18)
          plt.ylim((-1000,90000))
Out[117]: (-1000, 90000)
```

Data Wrangling

- After cleaning the dataset, the next step was to format it into a suitable DataFrame for conducting analysis and modelling on it
- This was done using the Pandas Dummies function and creating a table with many columns of binary variables to represent the categorical variables
- The final table for analysis looked like the following:



Out[189]:

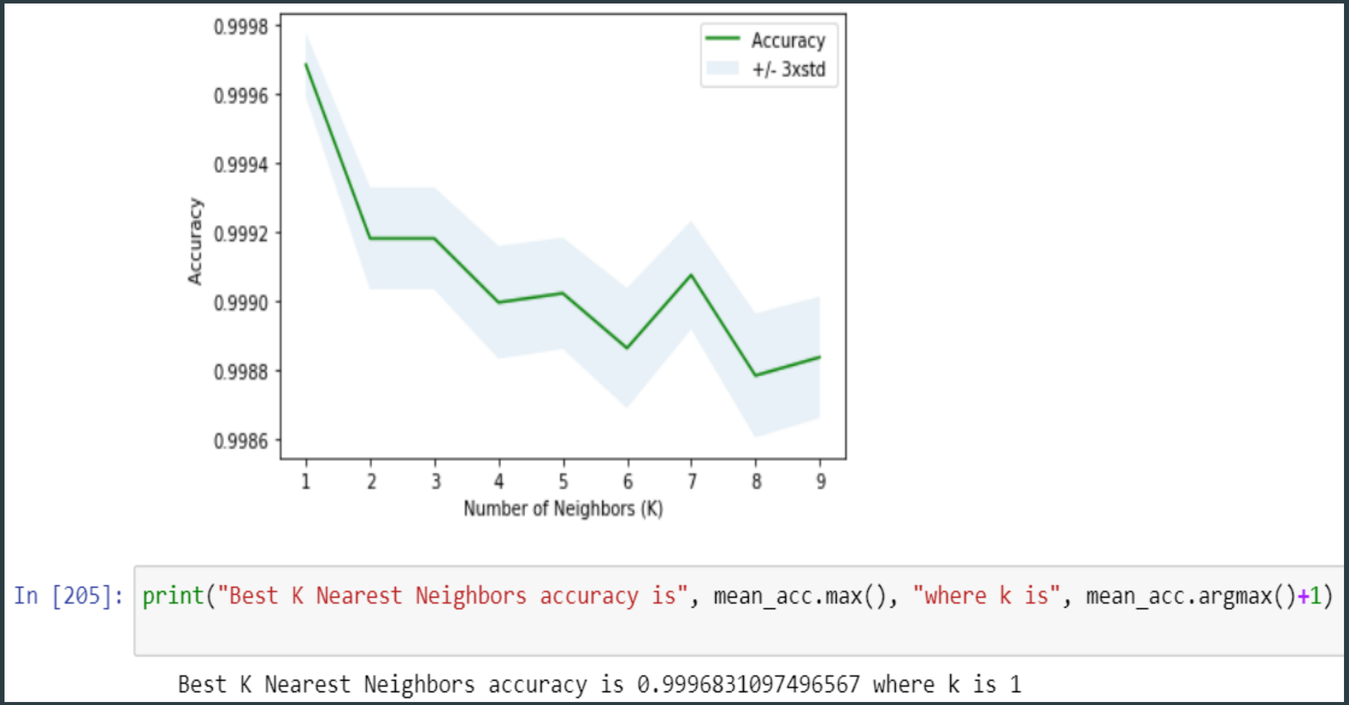
	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER	Dry	Ice	Oil	Other	Sand/Mud/Dirt	Snow/Slush	...	Clear	Fog/Smog/Smoke
0	2	Wet	Daylight	Overcast	0	0	0	0	0	0	...	0	0
1	1	Wet	Dark - Street Lights On	Raining	0	0	0	0	0	0	...	0	0
2	1	Dry	Daylight	Overcast	1	0	0	0	0	0	...	0	0
3	1	Dry	Daylight	Clear	1	0	0	0	0	0	...	1	0
4	2	Wet	Daylight	Raining	0	0	0	0	0	0	...	0	0

5 rows × 33 columns

- From this, a training and testing set was created for the modelling algorithms

K-Nearest Neighbors

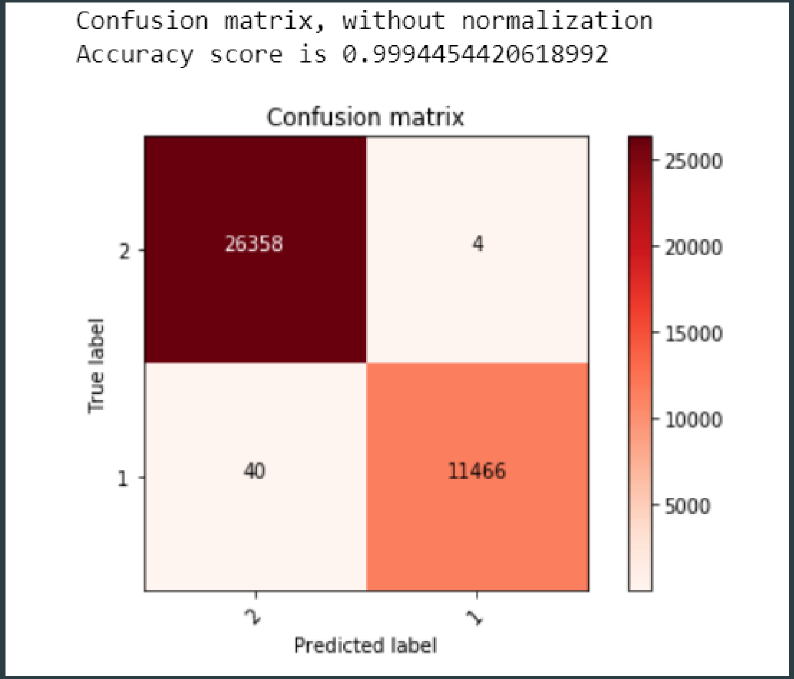
- The first algorithm I used was the KNN model, which classifies data points relative to the characteristics of nearby points. This is a slow process and iterating through various K values took some time, but resulted in a trained model with a highly accurate prediction.



```
In [205]: print("Best K Nearest Neighbors accuracy is", mean_acc.max(), "where k is", mean_acc.argmax()+1)

          Best K Nearest Neighbors accuracy is 0.9996831097496567 where k is 1
```

Support Vector Machine

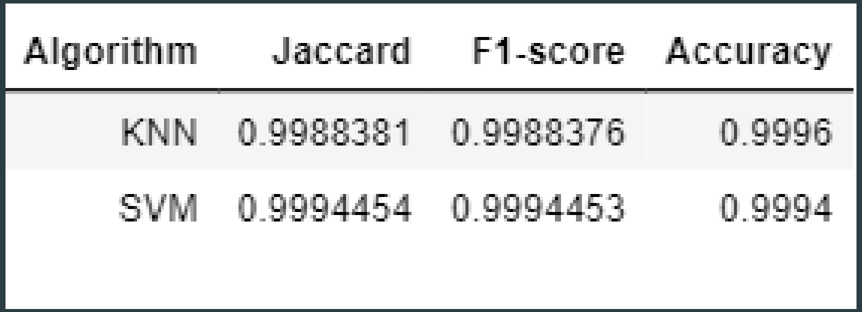
- The other algorithm used was an SVM which uses higher dimensional spaces to plot points and solves for hyperplanes which divide the different classifications into subspaces. This algorithm was much faster than the KNN albeit slightly less accurate than it. I visualised the output of this model through a confusion matrix to check for True Positives, True Negatives, False Positives and False Negatives.



```
Confusion matrix, without normalization
Accuracy score is 0.9994454420618992
```

Evaluation

- Both the models were accurate in predicting the labels of the testing data set and acting as a gauge for whether certain driving conditions would lead to an injury accident or just a property damage one. To further evaluate the models, I used the Jaccard Index and F1 score in combination with standard accuracy measures (R^2) and tabled the SVM against the KNN. This was done by importing the standard functions from the sklearn metrics toolkit.



Algorithm	Jaccard	F1-score	Accuracy
KNN	0.9988381	0.9988376	0.9996
SVM	0.9994454	0.9994453	0.9994

Conclusion

- The margin of error for each of these models is extremely low due to the large quantity of training data. Due to the extremely strong performance of the models, both would suffice as strong predictors for the Seattle Government to use to gauge the severity of accidents with given weather, road, and lighting conditions.
- However, performance is also a key metric which arises when deciding between models for usage in a larger scale setting. As the KNN algorithm took almost an hour to iterate through several variations of K and 15 minutes for a specific K, the time complexity of this algorithm is quite slow. On the other hand, the SVM only took 5 minutes to run the same dataset and came up with predictions that were in an error range of 6 significant figures of the KNN. The Jaccard and F1 score were extremely tight as well, with both algorithms sporting strong readings in both tests.
- Overall, based on the time performance and similar accuracy of the model, the **Support Vector Machine is the better algorithm** to model the impact of different driving conditions on the severity of an accident. The Seattle government can use this to deploy safety services and put up warning signs in regions where dangerous conditions are being exhibited and potentially lower the severity and number of accidents in the areas.