IBM Data Science Capstone Project v1.0

USA Accident – Traffic prediction Analysis

Prepared By: Vijayaragavan Narendran

1. Introduction

The aim of this project is predicting the traffic delay duration with given weather condition in United states. This model will be useful for who are new comers on the state for safe journey. A model to predict accident severity is built using these predictors as input vectors. Once the model is validated using a Machine Learning algorithm approach, we can be confident in predicting accident severity. In this proactive approach, the results of the analysis would be useful to various Entities like the Police and Insurance Companies. The goal is to reduce the fatalities and economic losses from accidents.

Approximately 1.35 million people die each year as a result of road traffic crashes. The 2030 Agenda for Sustainable Development has set an ambitious target of halving the global number of deaths and injuries from road traffic crashes by 2020. Road traffic crashes cost most countries 3% of their gross domestic product. More than half of all road traffic deaths are among vulnerable road users: pedestrians, cyclists, and motorcyclists. 93% of the world's fatalities on the roads occur in low- and middle-income countries, even though these countries have approximately 60% of the world's vehicles. Road traffic injuries are the leading cause of death for children and young adults aged 5-29 years.

1.1 Data Description

USA accident data has taken from https://www.kaggle.com/. This dataset contains several features and indicates that severity based on more features. Aim of this project is predict the severity with applicable features. These will be helpful and reduce the traffic delays. Severity data has 4 different values. 1-Minor, 2-Medium, 3-High,4-Very High.

2. Methodology

Dataset has 1.2gb size, and not taken all the data for prediction. Have filtered only 2019 data for prediction. Most of the data are having null values. Those null records were eliminated for better prediction.

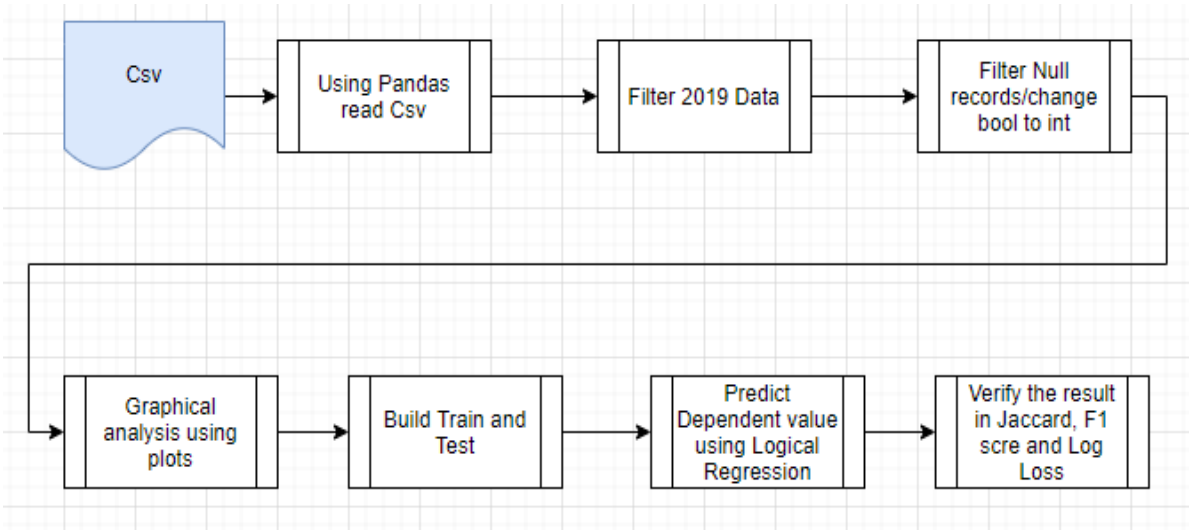
Some features ('Bump', 'Crossing', 'Junction', 'Roundabout', 'Stop') are having bool datatypes (True/False). True – data are replaced by 1 and False data are replaced by 0. Wind_Direction has more similar duplicate values hence replaced the values and keep unique and clear values, example (East to E, West to W, North to N, South to S). Start_date and End_date columns are string and converted as DateTime. “Wind Direction” and “weather condition” are having string values. Using pd dummies, those columns are converted as new separate rows with features which values are either 1 or 0. Used Logical Regression for getting prediction value.

Feature	Datatype	Required Feature for Prediction
ID	object	
Source	object	
TMC	float64	
Severity	int64	Y
Start_Time	object	Y

End_Time	object	Y
Start_Lat	float64	Y
Start_Lng	float64	Y
End_Lat	float64	Y
End_Lng	float64	Y
Distance(mi)	float64	
Description	object	
Number	float64	
Street	object	
Side	object	
City	object	Y
County	object	y
State	object	Y
Zipcode	object	
Country	object	
Timezone	object	
Airport_Code	object	
Weather_Timestamp	object	
Temperature(F)	float64	Y
Wind_Chill(F)	float64	Y
Humidity(%)	float64	Y
Pressure(in)	float64	Y
Visibility(mi)	float64	Y
Wind_Direction	object	Y
Wind_Speed(mph)	float64	Y
Precipitation(in)	float64	
Weather_Condition	object	Y
Amenity	bool	
Bump	bool	Y
Crossing	bool	Y
Give_Way	bool	
Junction	bool	Y
No_Exit	bool	
Railway	bool	Y
Roundabout	bool	Y
Station	bool	Y
Stop	bool	Y
Traffic_Calming	bool	
Traffic_Signal	bool	
Turning_Loop	bool	
Sunrise_Sunset	object	

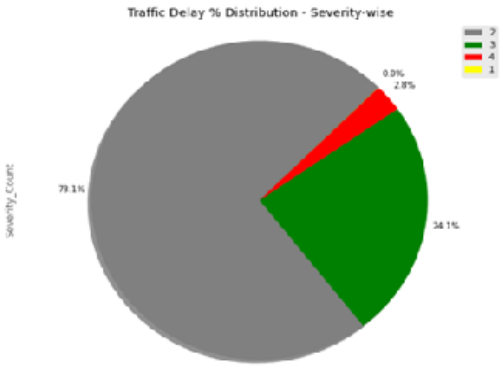
Civil_Twilight	object	
Nautical_Twilight	object	
Astronomical_Twilight	object	

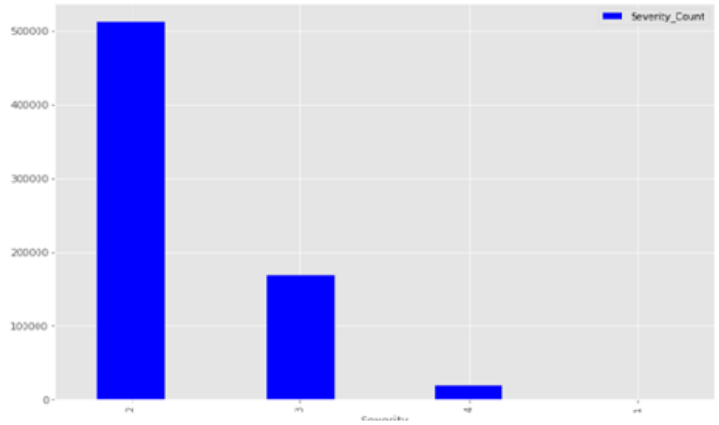
Below is the process flow diagram from csv to modelling



3. Data analysis:

3.1) Have to verify the severity count in DataFrame





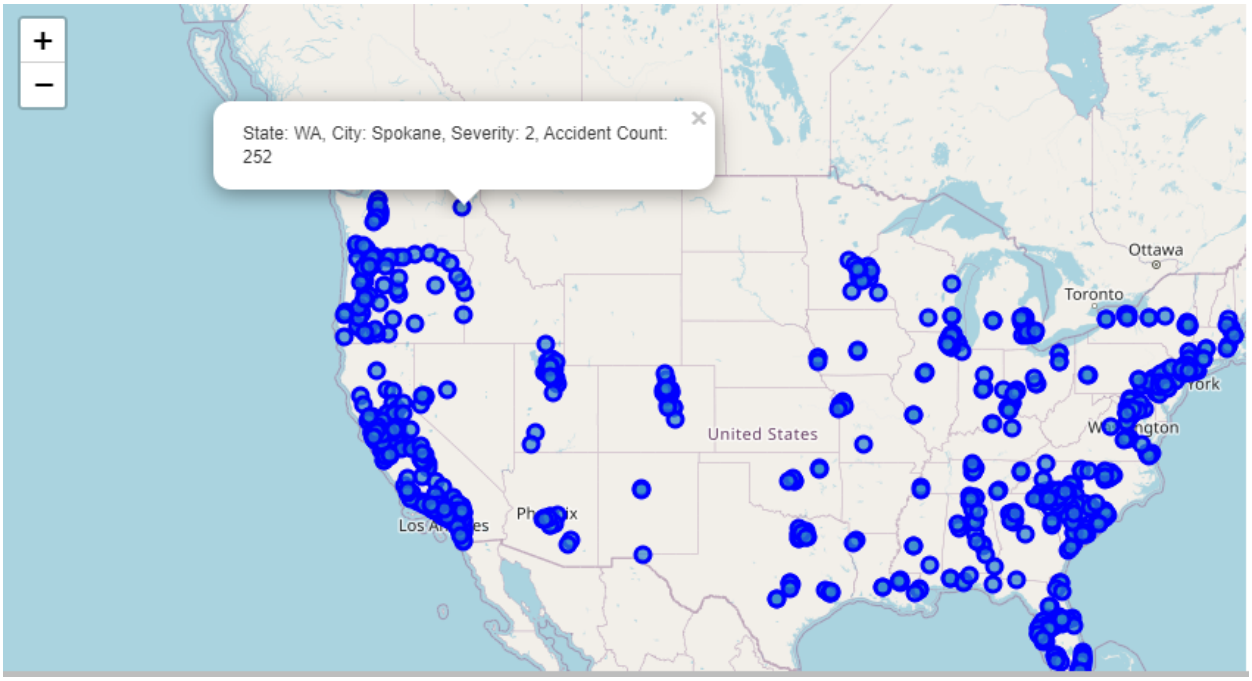
Severity	Severity_Count
2	511665
3	168942
4	19499
1	151

3.2) Have to verify Accident counts with state and city

```
accdf_city
```

	Severity	State	City	SeverityCount	Lat	Lan
5257	2	NC	Charlotte	15278	35.391026	-80.570168
9062	2	TX	Houston	12746	30.029045	-95.133965
1136	2	CA	Los Angeles	11891	34.156310	-118.123697
8943	2	TX	Austin	11716	30.513420	-97.554733
7304	2	OK	Oklahoma City	8226	35.623718	-97.264999
...	...	...	...	...	...	...

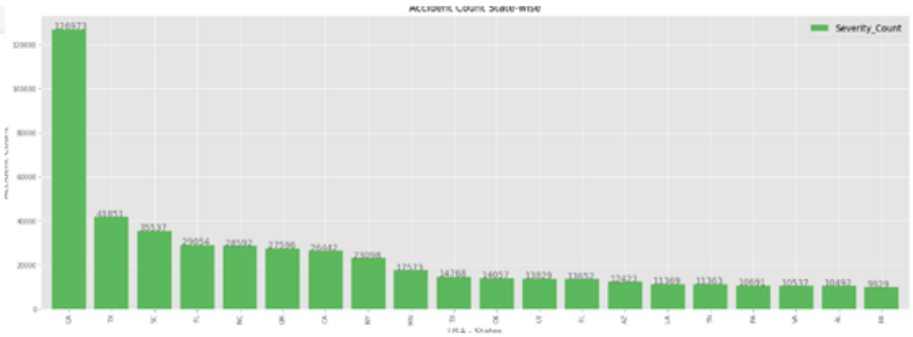
3.3)Use Folium map display the Accident counts in USA map by state, city wise



3.4)Below bar chart indicates Accident count in state wise

```
df_state_vis.head(10)
```

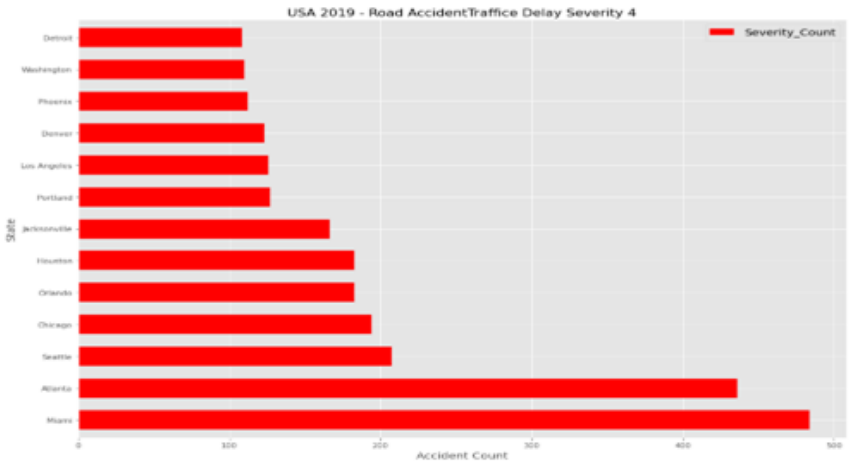
State	Severity_Count
CA	126973
TX	41851
SC	35537
FL	29054
NC	28592
OR	27596
CA	26442
NY	23098
MN	17573
TX	14768



3.5)Below Bar chart indicates only Severity4 to City wise

```
df_Sev4.head(10)
```

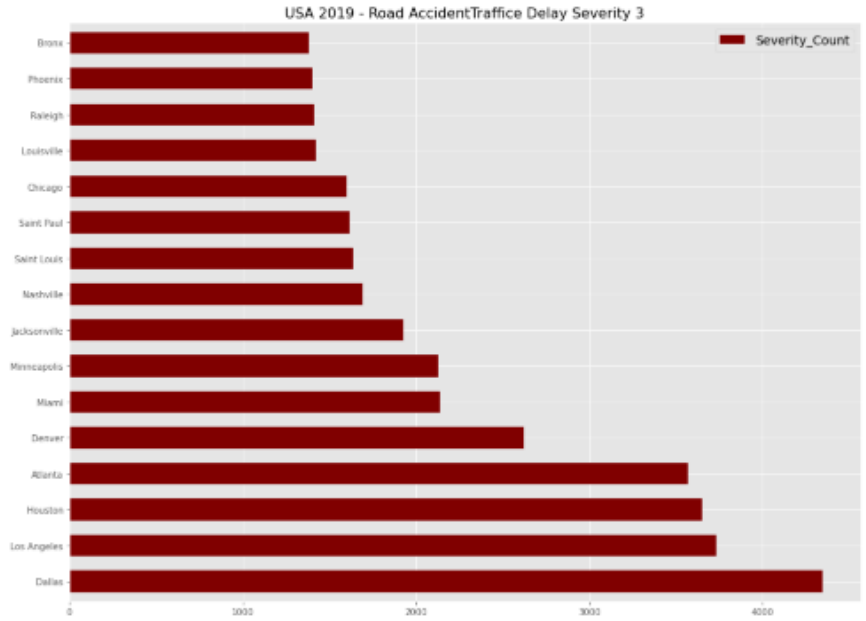
City	Severity_Count
Miami	484
Atlanta	436
Seattle	207
Chicago	194
Orlando	183
Houston	183
Jacksonville	166
Portland	127
Los Angeles	126
Denver	123



3.6)Below Bar chart indicates only Severity3 to City wise

```
df_Sev3.head(10)
```

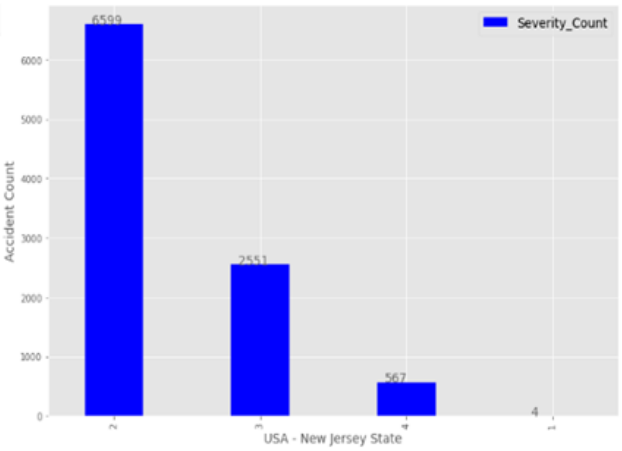
City	Severity_Count
Dallas	4348
Los Angeles	3734
Houston	3652
Atlanta	3569
Denver	2621
Miami	2138
Minneapolis	2130
Jacksonville	1927
Nashville	1694
Saint Louis	1636

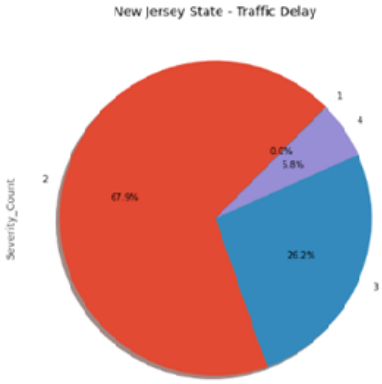


3.7)Below bar/pie chart shows Severity count for State: New Jersey count

```
df_nj_cnt
```

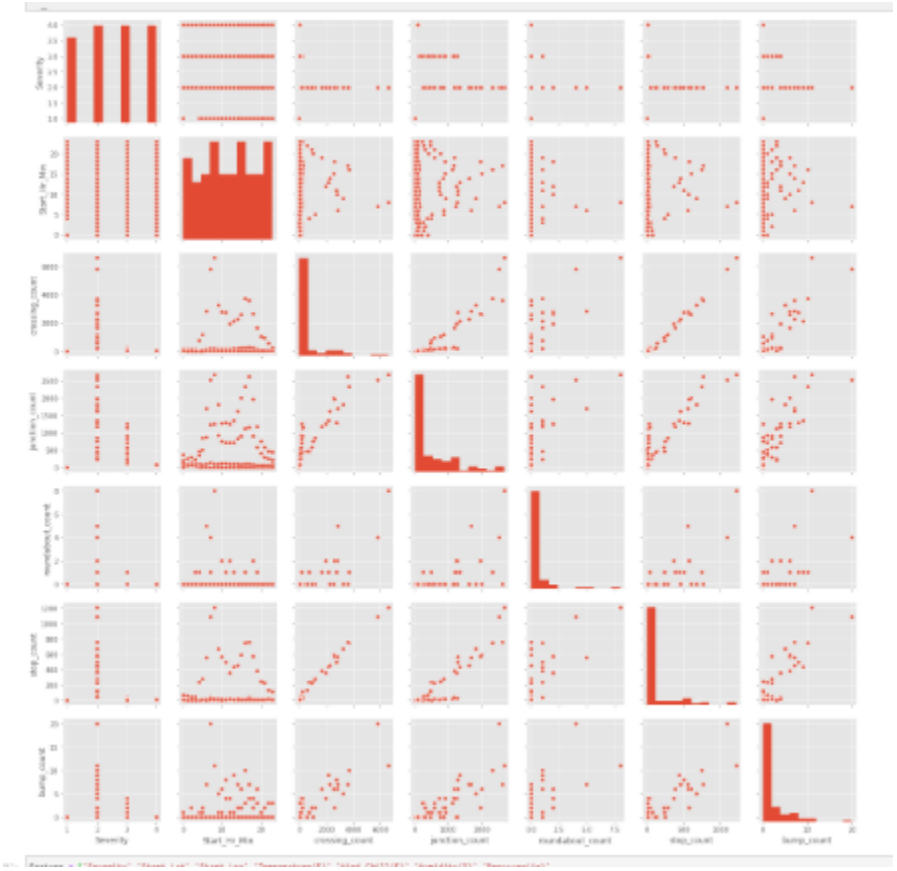
Severity	Severity_Count
2	6599
3	2551
4	567
1	4





3.8)Below plot indicates other column impact analysis to Severity

	Severity	Start_Hr_Min	crossing_count	junction_count	roundabout_count	stop_count	bump_count
0	1	0	0	0	0	0	0
1	1	4	1	0	0	0	0
2	1	5	0	0	0	0	0
3	1	6	2	0	0	0	0
4	1	7	5	0	0	0	0
...	...	...	...	...	...	...	...



4. Discussions:

Mentioned below plots are using in this project.

- Folium map
- Pie Chart
- Bar chart (vertical and horizontal)
- Pairplots

This project has below dependencies

- pandas
- numpy
- bs4
- matplotlib
- requests
- folium
- sklearn

After build the Train and Test data, prediction gives Severity 1 to 4.

5. **Conclusion**:

Built useful models to predict traffic delays and plan for travel in.

Accuracy of the models has room for improvement.

Below is the final score in Logical Regression model.

```
yhat = LR.predict(X_test)
yhat
```

```
array([2, 2, 3, ..., 2, 2, 2])
```

```
knn_Jaccard = jaccard_similarity_score(y_test, yhat)
print (knn_Jaccard)
knn_f1_score = f1_score(y_test, yhat, average='weighted')
print (knn_f1_score)
ll_log_loss = log_loss(y_test, yhat_prob)
print (ll_log_loss)
```

```
0.9720675177791106
0.9592660214774792
0.05748875901860401
```