Mass. traffic accidents and classification of traffic impact

Kachun Lee(Tim)

Dataset: U.S. traffic accidents (2016 - 2020)

- Original dataset contains data collected in real time on traffic accidents in the United States
- Cut down dataset to focus on MA and selected for certain features (more on next slide)
- Goals - Focus on a subset of data where we could do meaningful analysis given the dataset was so large: specifically discovering accident hotspots and studying the impact of features that may help predict traffic accidents and/or their level of severity
- Hopefully being aware of conditions that determine traffic accidents could increase awareness amongst drivers and help prevent future accidents or at least reduce their severity

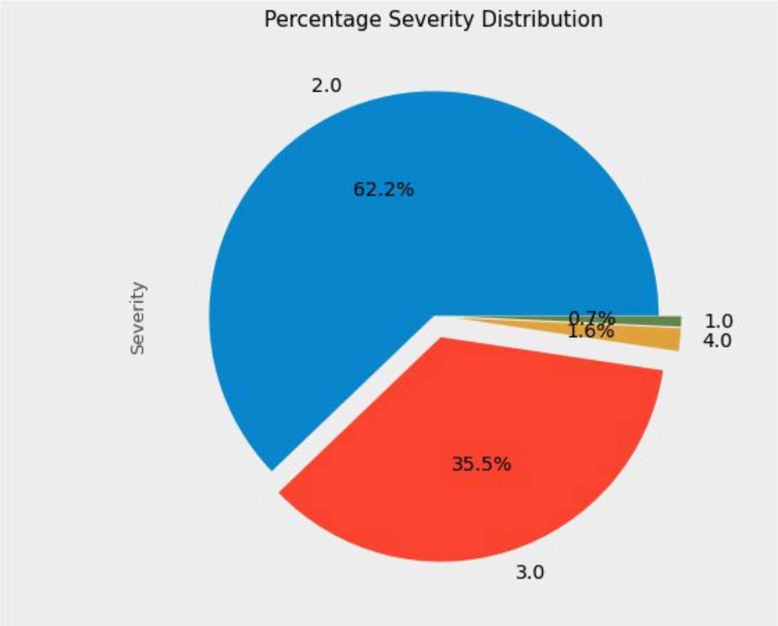
Resulting project data

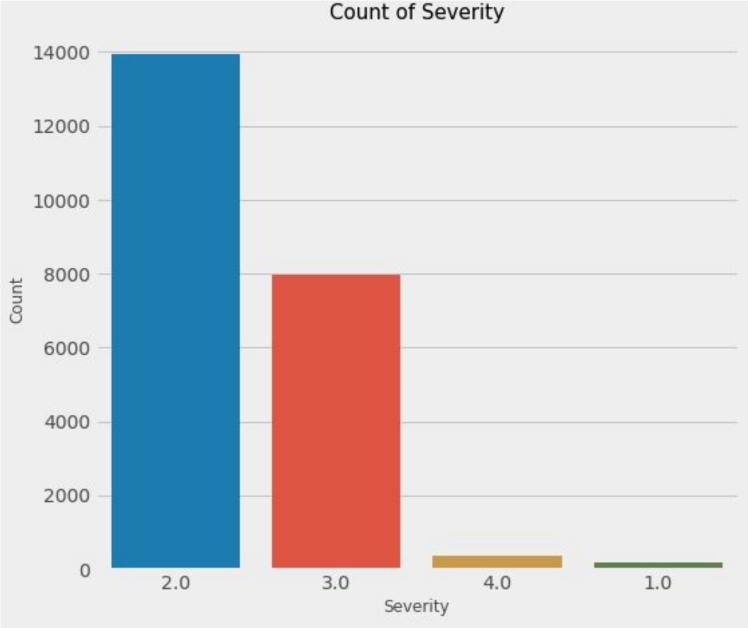
- Reduced dataset from ~2.7 million data points (rows) to ~22,000 for MA
- Narrowed it down to MA state only and
- Features kept: latitude and longitude, distance, city, county, temperature, time, day, severity etc. →

0	Severity	22456 non-null	float64
1	Start_Lng	22456 non-null	float64
2	Start_Lat	22456 non-null	float64
3	Distance(mi)	22456 non-null	float64
4	Side	22456 non-null	object
5	City	22456 non-null	object
6	County	22456 non-null	object
7	Timezone	22456 non-null	object
8	Temperature(F)	22456 non-null	float64
9	Humidity(%)	22456 non-null	float64
10	Pressure(in)	22456 non-null	float64
11	Visibility(mi)	22456 non-null	float64
12	Wind_Direction	22456 non-null	object
13	Weather_Condition	22456 non-null	object
14	Amenity	22456 non-null	object
15	Bump	22456 non-null	object
16	Crossing	22456 non-null	object
17	Give_Way	22456 non-null	object
18	Junction	22456 non-null	object
19	No_Exit	22456 non-null	object
20	Railway	22456 non-null	object
21	Roundabout	22456 non-null	object
22	Station	22456 non-null	object
23	Stop	22456 non-null	object
24	Traffic_Calming	22456 non-null	object
25	Traffic_Signal	22456 non-null	object
26	Turning_Loop	22456 non-null	object
27	Sunrise_Sunset	22456 non-null	object
28	Hour	22456 non-null	float64
29	Weekday	22456 non-null	object
30	Time_Duration(min)	22456 non-null	float64

EDA

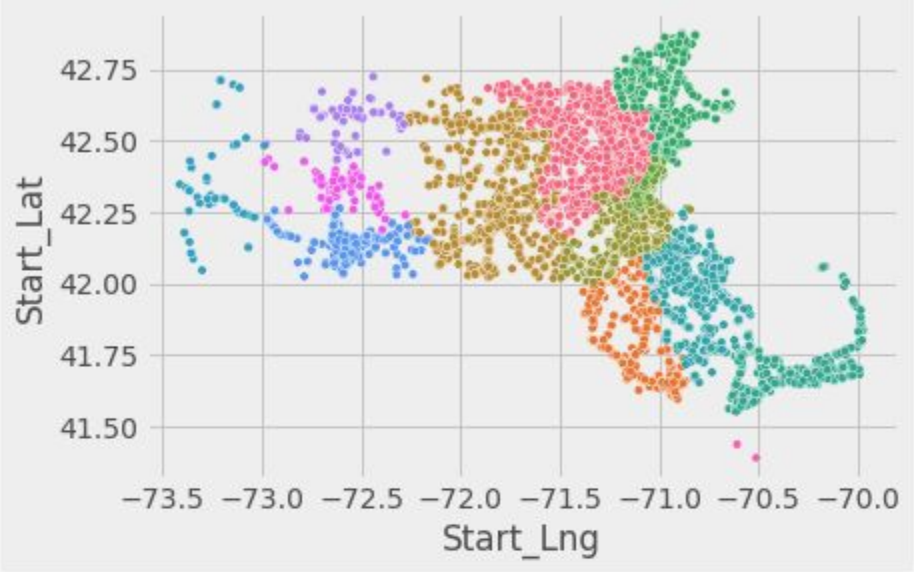
Severity (delay impact on traffic)





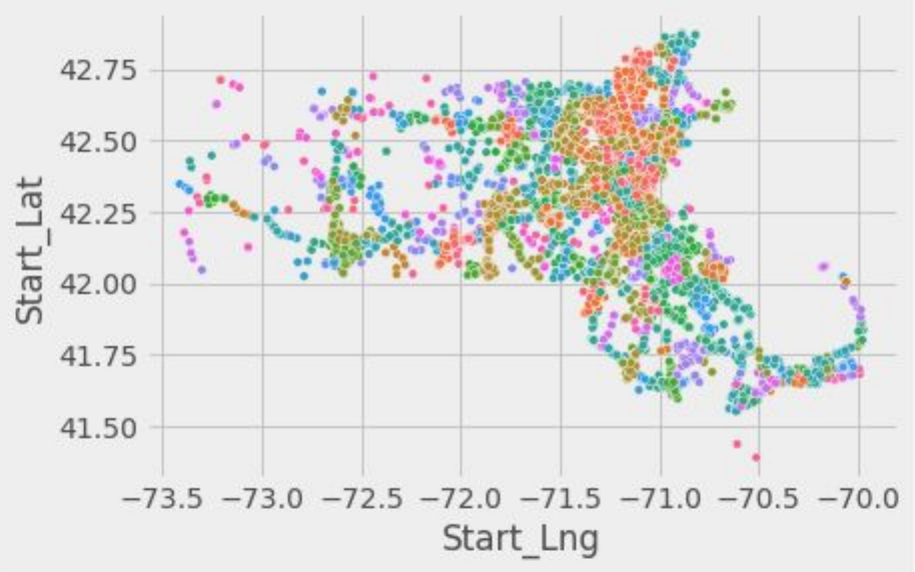
Accidents by county in Massachusetts

- Datapoints (accidents) by county
- Some significant clustering (major cities)
- Middlesex county (pink hue) - Cambridge, universities (MIT, Tufts...)
- Suffolk county (Boston, other major cities, a lot of people commuting for work, state st)



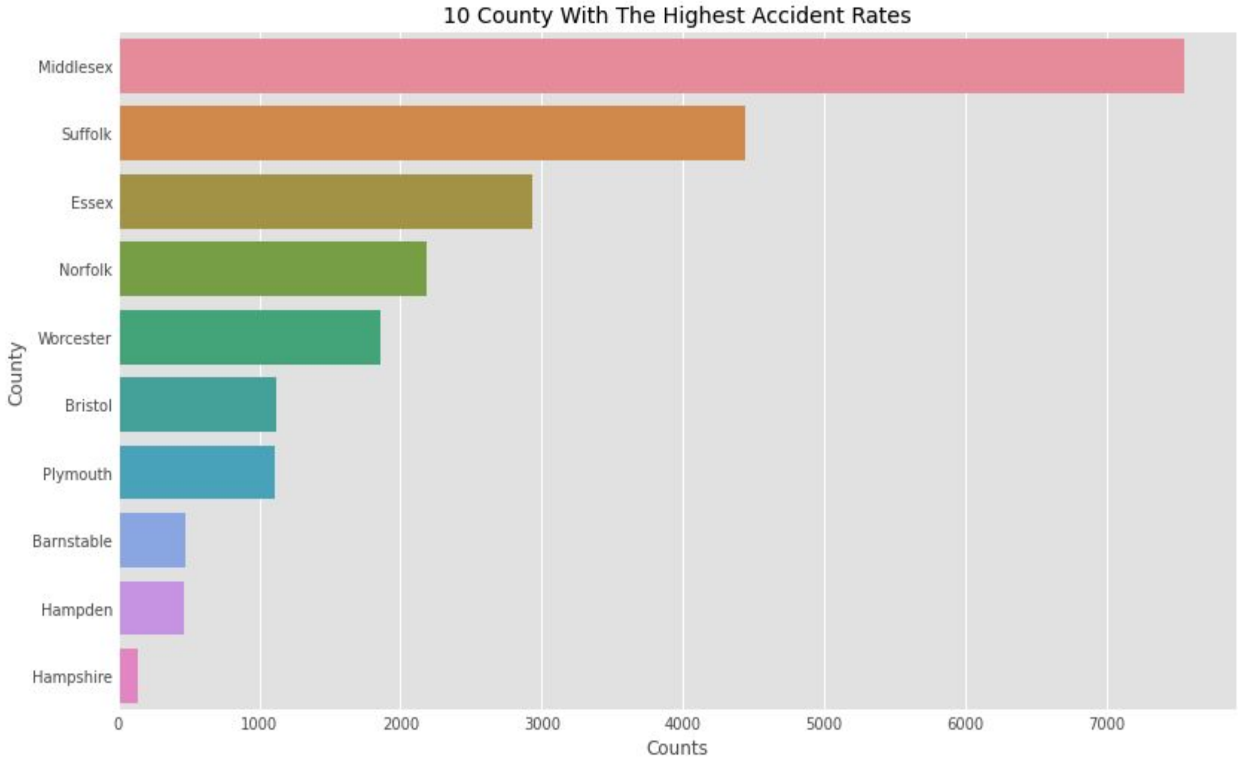
Accidents by city in Massachusetts

- Datapoints (accidents) by city
- Similar significant clustering near large cities (Boston, Cambridge, Brookline)
- Bristol and Plymouth get a little sparse
- Cape Cod (vacation spot)



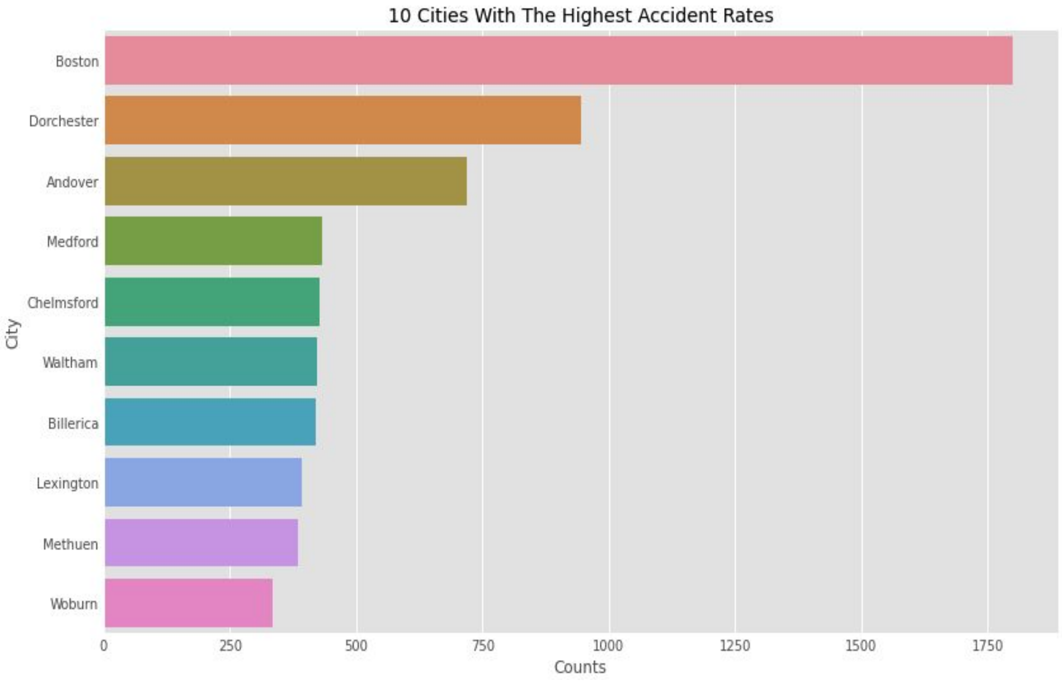
MA counties

- Accidents by county
- Top: Middlesex and Suffolk



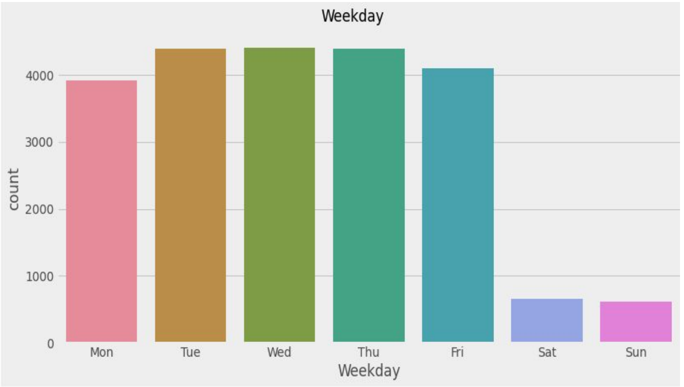
Highest accident rates

- Top 10 number of accidents per city
- Again, large disparity amongst top 10 cities, Boston has highest rate
- Over double the second highest which is Dorchester

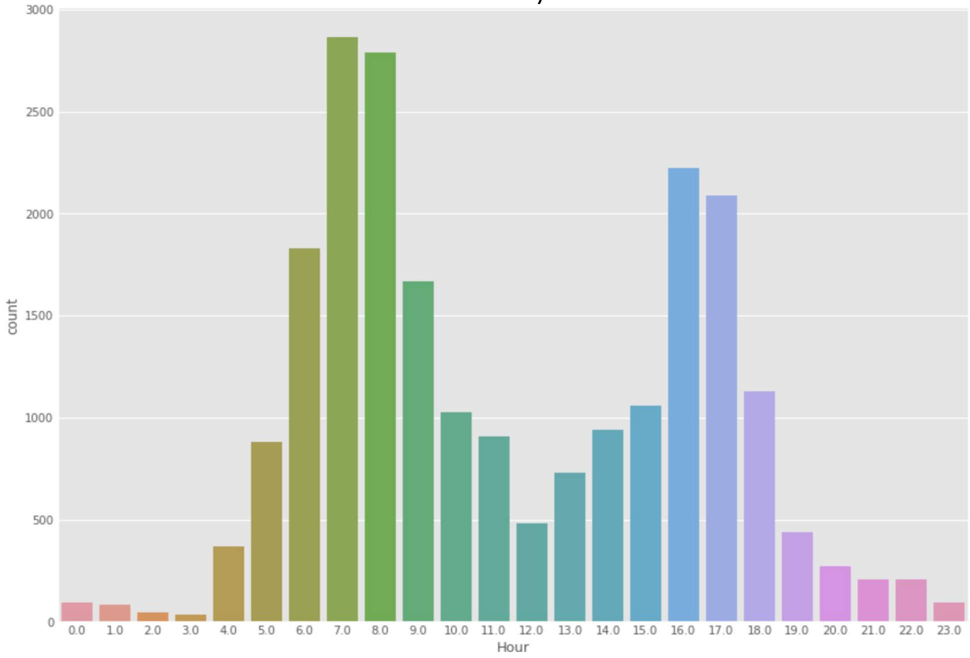


Weekday and time of day

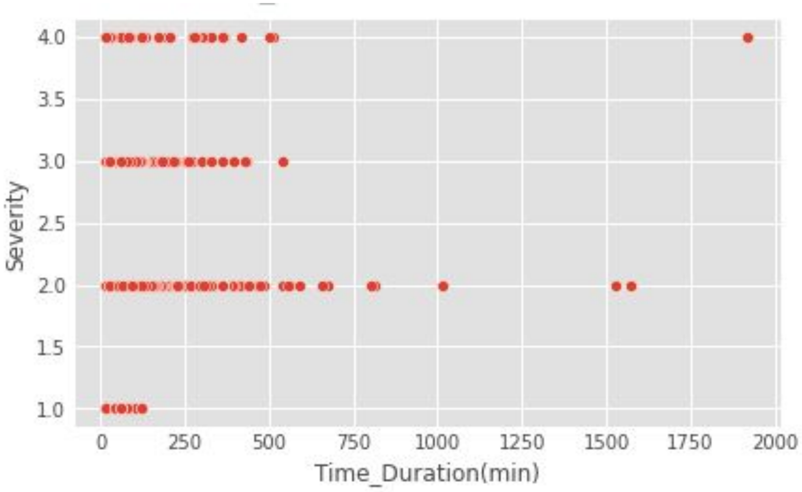
Most accidents occur during regular work week and rush hour traffic (ie. when people are likely driving to work or driving from work)

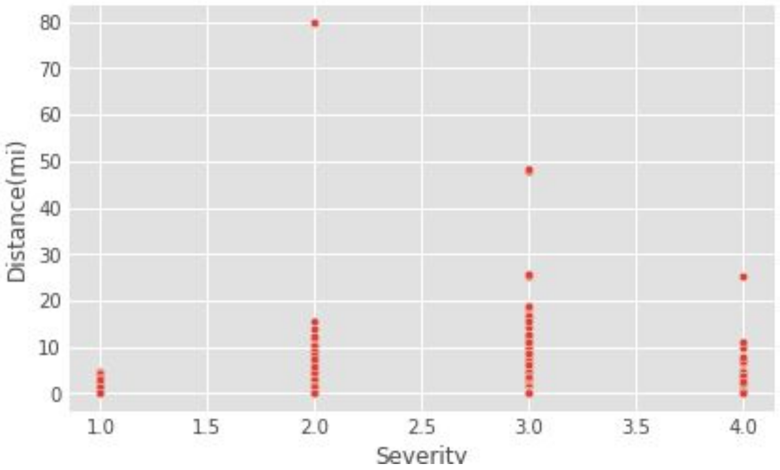


Time of day



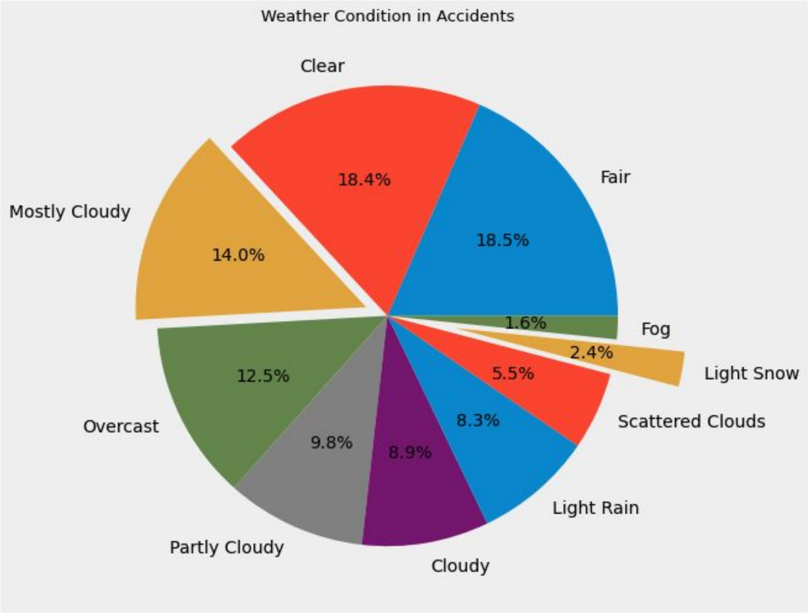
Relationships between features and Target





Weather Condition

- Fair and Clear is the most frequent situation
- Following is Mostly cloudy
- Overcast
- Partly Cloudy

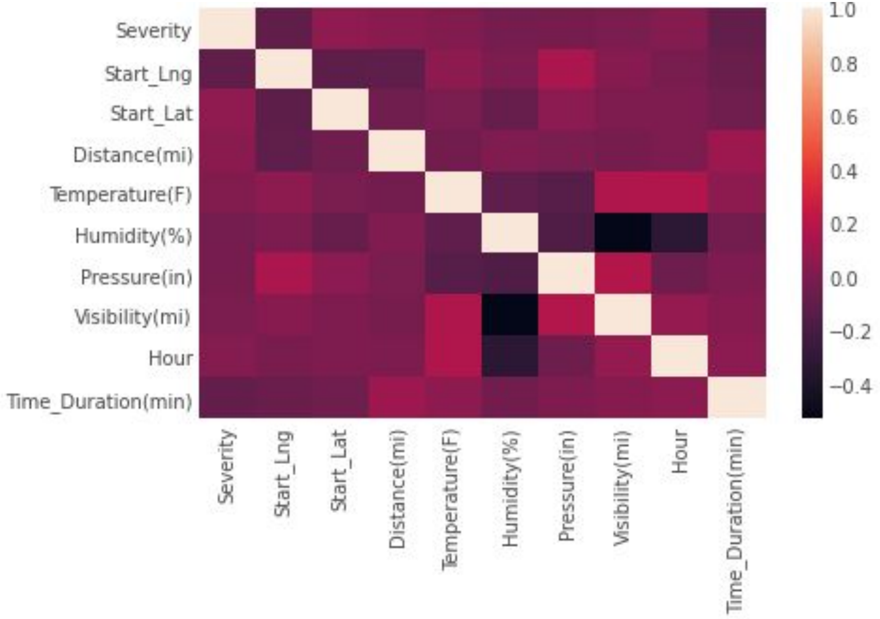


Statistics Summary table

	Severity	Start_Lng	Start_Lat	Distance(mi)	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Hour	Time_Duration(min)
count	22456.000000	22456.000000	22456.000000	22456.000000	22456.000000	22456.000000	22456.000000	22456.000000	22456.000000	22456.00000
mean	2.379943	-71.233870	42.345667	0.263861	52.406439	66.946206	29.939952	8.707195	11.365025	77.98811
std	0.531959	0.384438	0.240019	1.368770	18.881792	20.697541	0.318814	2.829148	4.811915	97.28543
min	1.000000	-73.412150	41.389977	0.000000	-13.000000	8.000000	27.790000	0.000000	0.000000	15.00000
25%	2.000000	-71.289574	42.238193	0.000000	37.900000	50.000000	29.780000	10.000000	7.000000	30.00000
50%	2.000000	-71.140488	42.357565	0.000000	52.000000	68.000000	29.960000	10.000000	10.000000	45.00000
75%	3.000000	-71.055998	42.516640	0.010000	68.000000	86.000000	30.130000	10.000000	16.000000	75.00000
max	4.000000	-69.973511	42.877491	79.946000	97.000000	100.000000	30.860000	10.500000	23.000000	1920.00000

Correlation heatmap

	Severity
Severity	1.000000
Start_Lng	-0.100034
Start_Lat	0.065549
Distance(mi)	0.045119
Temperature(F)	0.019396
Humidity(%)	-0.031258
Pressure(in)	-0.022908
Visibility(mi)	-0.007308
Hour	0.022871
Time_Duration(min)	-0.089494



Classification-Logistic Regression

Accuracy=0.743

Precision=0.75

Recall=0.74

```
[Logistic regression algorithm] accuracy_score: 0.743.

             precision    recall  f1-score   support

        1.0       0.57      0.12      0.20        34
        2.0       0.79      0.81      0.80      2924
        3.0       0.65      0.67      0.66      1672
        4.0       1.00      0.04      0.07        78

avg / total       0.75      0.74      0.74      4708
```

Grid Search CV

- Increase 2% accuracy

-From 69% to 71% accuracy

```
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)
```

```
Tuned Hyperparameters : {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy : 0.7194672131147541
```

use best parameter to fit the model

```
logreg = LogisticRegression(C = 0.1,
                            penalty = 'l2',
                            solver = 'newton-cg')
logreg.fit(X_train,y_train)
y_pred = logreg.predict(X_test)
print("Accuracy:",logreg.score(X_test, y_test))
```

```
Accuracy: 0.7106557377049181
```

Classification-KNN

- algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.
- mostly used to classifies a data point based on how its neighbours are classified.(Euclidean distance)
- K:the number of nearest neighbours to include in the majority of the voting process.

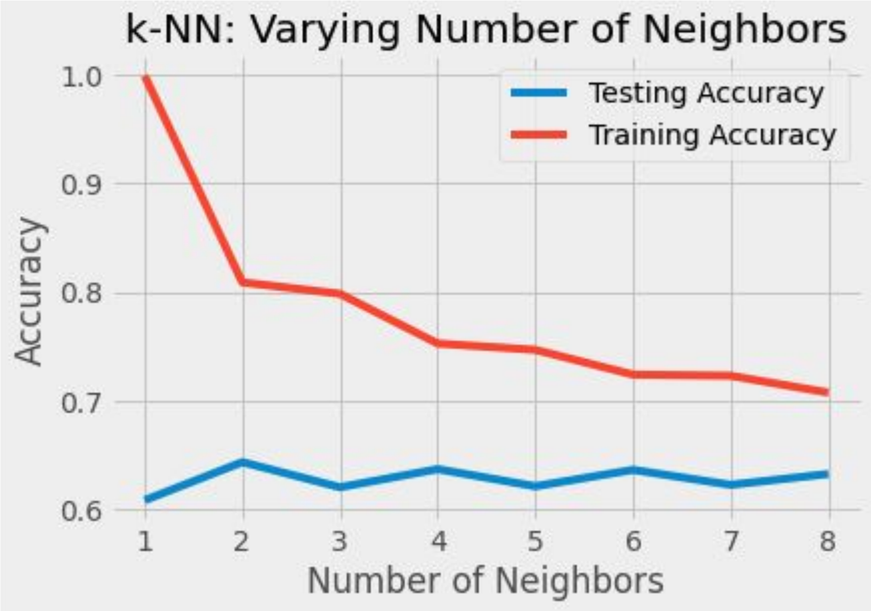
- Using K=5 at the beginning

[K-Nearest Neighbors (KNN)] knn.score: 0.624.

	precision	recall	f1-score	support
1.0	0.35	0.24	0.28	34
2.0	0.68	0.76	0.72	2924
3.0	0.50	0.41	0.45	1672
4.0	0.53	0.12	0.19	78
avg / total	0.61	0.62	0.61	4708

Classification-KNN

- Convergence from 5 Neighbors
- Best at 8 Neighbors



Classification-KNN

- Accuracy from 62.4% to 63.8%
- Precision stays the same
- Recall from 62% to 64%

[K-Nearest Neighbors (KNN)] knn.score: 0.638.

	precision	recall	f1-score	support
1.0	0.23	0.09	0.13	34
2.0	0.66	0.86	0.75	2924
3.0	0.53	0.28	0.37	1672
4.0	0.60	0.12	0.19	78
avg / total	0.61	0.64	0.60	4708

Classification-Decision Tree

- uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.
- Precision:0.76
- Recall:0.67

```
[Decision Tree -- entropy] accuracy_score: 0.684.
[Decision Tree -- gini] accuracy_score: 0.673.

             precision    recall  f1-score   support

        1.0       0.00      0.00      0.00        34
        2.0       0.90      0.56      0.69      2924
        3.0       0.53      0.91      0.67      1672
        4.0       0.64      0.18      0.28        78

avg / total       0.76      0.67      0.67      4708
```

Classification-Random forest

- establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees.

- A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision.
- number of estimators (50, 100, 250, 500)

- Precision:0.814

	precision	recall	f1-score	support
1.0	1.00	0.36	0.53	25
2.0	0.80	0.96	0.87	801
3.0	0.83	0.58	0.69	315
4.0	0.88	0.38	0.53	79
accuracy			0.81	1220
macro avg	0.88	0.57	0.66	1220
weighted avg	0.82	0.81	0.80	1220

Grid Search

Accuracy-0.7393

Not considering using Grid- search

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=4,
             param_grid={'bootstrap': [True, False],
                         'max_depth': [2, 4, 6, 8, 10],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 5, 10],
                         'n_estimators': [50, 100, 200, 300, 500, 800, 1000]},
             return_train_score=True, scoring='accuracy')
```

Random search parameter for random Forest

```
rf_random.best_params_
```

```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 50,
 'bootstrap': False}
```

```
best_model = RandomForestClassifier(n_estimators= 1000,
 min_samples_split= 2,
 min_samples_leaf= 1,
 max_features= 'auto',
 max_depth= 50,
 bootstrap= False)
best_model.fit(X_train,y_train)

y_pred=best_model.predict(X_test)

#  accuracy score
accuracy=accuracy_score(y_test, y_pred)
accuracy
```

```
0.8254098360655737
```

	precision	recall	f1-score	support
1.0	0.92	0.44	0.59	25
2.0	0.81	0.96	0.88	801
3.0	0.86	0.60	0.71	315
4.0	0.83	0.44	0.58	79
accuracy			0.82	1220
macro avg	0.86	0.61	0.69	1220
weighted avg	0.83	0.82	0.81	1220

Accuracy increased 1 %

Feature importance

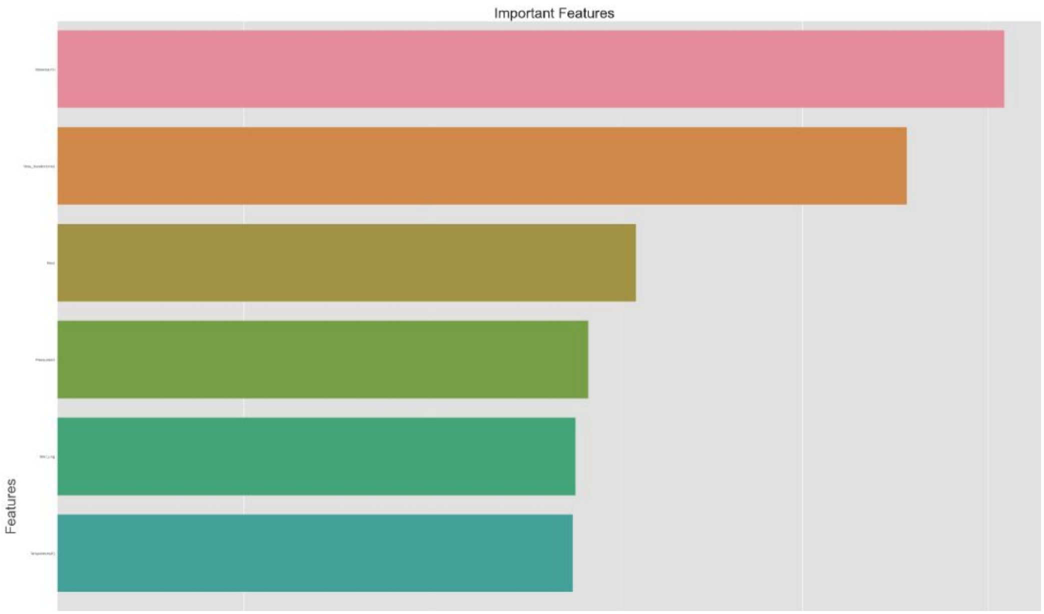
1.Distance

2:Hour

3:Pressure

4: Start_Lng

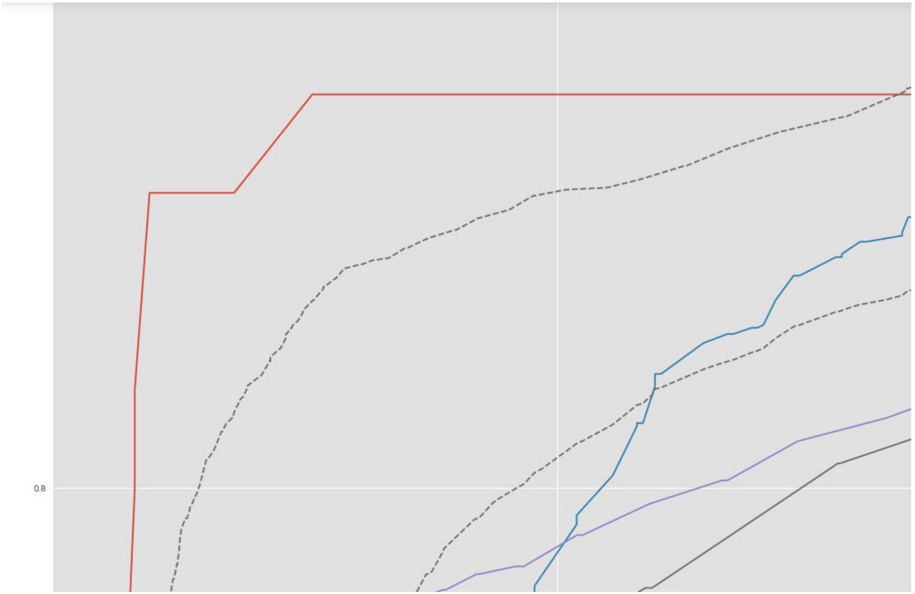
5:Temperature



Truncated ROC AUC curve

ROC of class 1 :0.96
ROC of class 2 :0.86
ROC of class 3 :0.85
ROC of class 4:0.85

AUC -0.88



XG boost after grid_search

```
The best hyperparameters are {'colsample_bytree': 0.3, 'reg_alpha': 0, 'reg_lambda': 0}
```

```
accuracy = accuracy_score(y_test, grid_predict)
accuracy
```

```
0.8065573770491803
```

What to improve?

Need to take a look on whether the model is under unbalance sampling on some minority class of accident

Try on deep-learning