GNR 652 Machine learning for Remote Sensing

Course Project

Road Accident Severity Prediction

Submitted to:
Prof. B. Banerjee

Submitted By:
Abhijendra Anand (213310010)
Amol Narang (213310001)
Shubham Gupta (213310005)
Swabhiman Dehariya (213310018)
Uddesh Sahu (213310004)

Problem Definition

The United States of America has recorded 2.8 millions accidents from 2016-2021. The dataset contains 47 features. These accidents are divided into 4 categories according to their severity: Severity level 1, 2, 3 and 4.

Develop a model to help decision makers to enhance the decision making process by predicting the severity of the accident.

Motivation behind selecting the Problem

- According to WHO, every year the lives of approximately 1.3 million people are cut short as a result of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability as a result of their injury.

- Road traffic injuries cause considerable economic losses to individuals, their families, and to nations as a whole.

- Traffic safety management can be improved by accurate prediction of road accidents, because the prominent influencing factors in high-risk road sections could be found out to provide beneficial suggestions for improving road safety.

- In this project, an attempt has been made to analyse and predict the severity of the accidents for the US so that the road authority can take relevant factors into account and make appropriate policies to reduce the number of accidents.

Knowing our Dataset

Following Columns are present in original dataset:

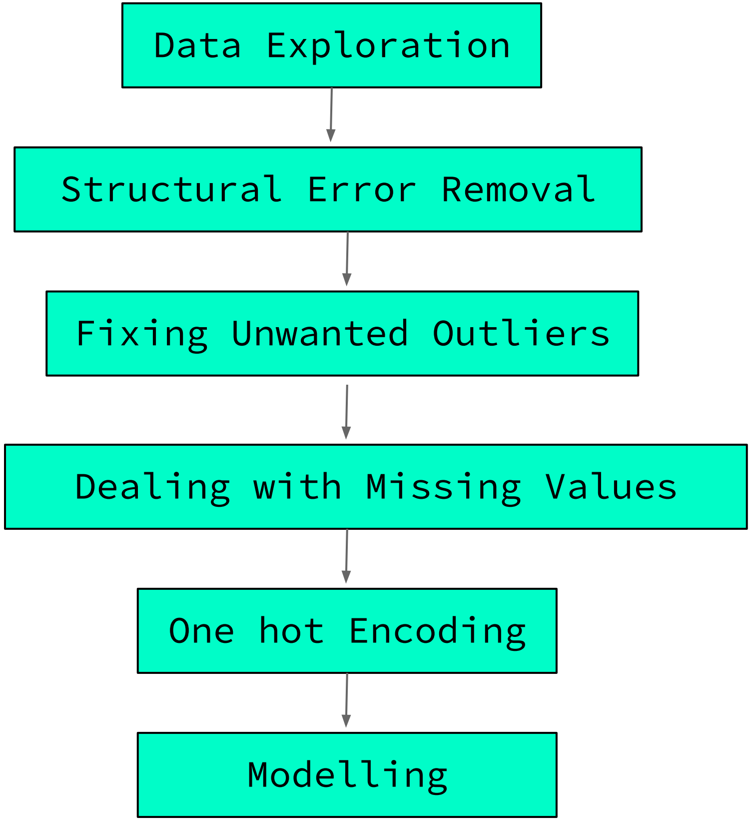
['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
'Astronomical_Twilight']

Sr. No.	Attribute	Description
1	ID	This is a unique identifier of the accident record.
2	Severity	Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).
3	Start_Time	Shows start time of the accident in local time zone.
4	End_Time	Shows end time of the accident in local time zone. End time here refers to when the impact of accident on traffic flow was dismissed.
5	Start_Lat	Shows latitude in GPS coordinate of the start point.
6	Start_Lng	Shows longitude in GPS coordinate of the start point.
7	End_Lat	Shows latitude in GPS coordinate of the end point.
8	End_Lng	Shows longitude in GPS coordinate of the end point.
9	Distance(mi)	The length of the road extent affected by the accident.
10	Description	Shows natural language description of the accident.
11	Number	Shows the street number in address field.
12	Street	Shows the street name in address field.
13	Side	Shows the relative side of the street (Right/Left) in address field.

Sr. No.	Attribute	Description
14	City	Shows the city in address field.
15	County	Shows the county in address field.
16	State	Shows the state in address field.
17	Zipcode	Shows the zipcode in address field.
18	Country	Shows the country in address field.
19	Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).
20	Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.
21	Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).
22	Temperature(F)	Shows the temperature (in Fahrenheit).
23	Wind_Chill(F)	Shows the wind chill (in Fahrenheit).
24	Humidity(%)	Shows the humidity (in percentage).
25	Pressure(in)	Shows the air pressure (in inches).
26	Visibility(mi)	Shows visibility (in miles).
27	Wind_Direction	Shows wind direction.

Sr. No.	Attribute	Description
28	Wind_Speed(mph)	Shows wind speed (in miles per hour).
29	Precipitation(in)	Shows precipitation amount in inches, if there is any.
30	Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)
31	Amenity	A PO annotation which indicates presence of amenity in a nearby location.
32	Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.
33	Crossing	A POI annotation which indicates presence of crossing in a nearby location.
34	Give_Way	A POI annotation which indicates presence of give_way in a nearby location.
35	Junction	A POI annotation which indicates presence of junction in a nearby location.
36	No_Exit	A POI annotation which indicates presence of no_exit in a nearby location.
37	Railway	A POI annotation which indicates presence of railway in a nearby location.
38	Roundabout	A POI annotation which indicates presence of roundabout in a nearby location.
39	Station	A POI annotation which indicates presence of station in a nearby location.
40	Stop	A POI annotation which indicates presence of stop in a nearby location.
41	Traffic_Calming	A POI annotation which indicates presence of traffic_calming in a nearby location.
45	Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.

Flow Chart of Solution



Data Exploration

Checking Data types of each columns

Checking number of unique values for each column

Dropping the columns which have only one unique value in all the records

Dropping the columns which are obviously not related to the Severity

Structural Error removal

Example of structural error = Wind_Direction column:

```
['SW', 'Calm', 'WSW', 'WNW', 'West', 'NNW', 'South', 'W', 'NW',
  'North', 'SSE', 'SSW', 'ESE', 'SE', nan, 'East', 'Variable', 'NNE',
  'NE', 'ENE', 'CALM', 'S', 'VAR', 'N', 'E']
```

Code to solve this issue:

```
df.Wind_Direction.replace(to_replace=['WNW','W','WSW'], value= 'West', inplace=True)
df.Wind_Direction.replace(to_replace=['ESE','E','ENE'], value= 'East', inplace=True)
df.Wind_Direction.replace(to_replace=['NNW','N','NNE'], value= 'North', inplace=True)
df.Wind_Direction.replace(to_replace=['SSW','S','SSE'], value= 'South', inplace=True)
df.Wind_Direction.replace(to_replace=['CALM'], value= 'Calm', inplace=True)
df.Wind_Direction.replace(to_replace=['VAR'], value= 'Variable', inplace=True)
```

Structural Error removal

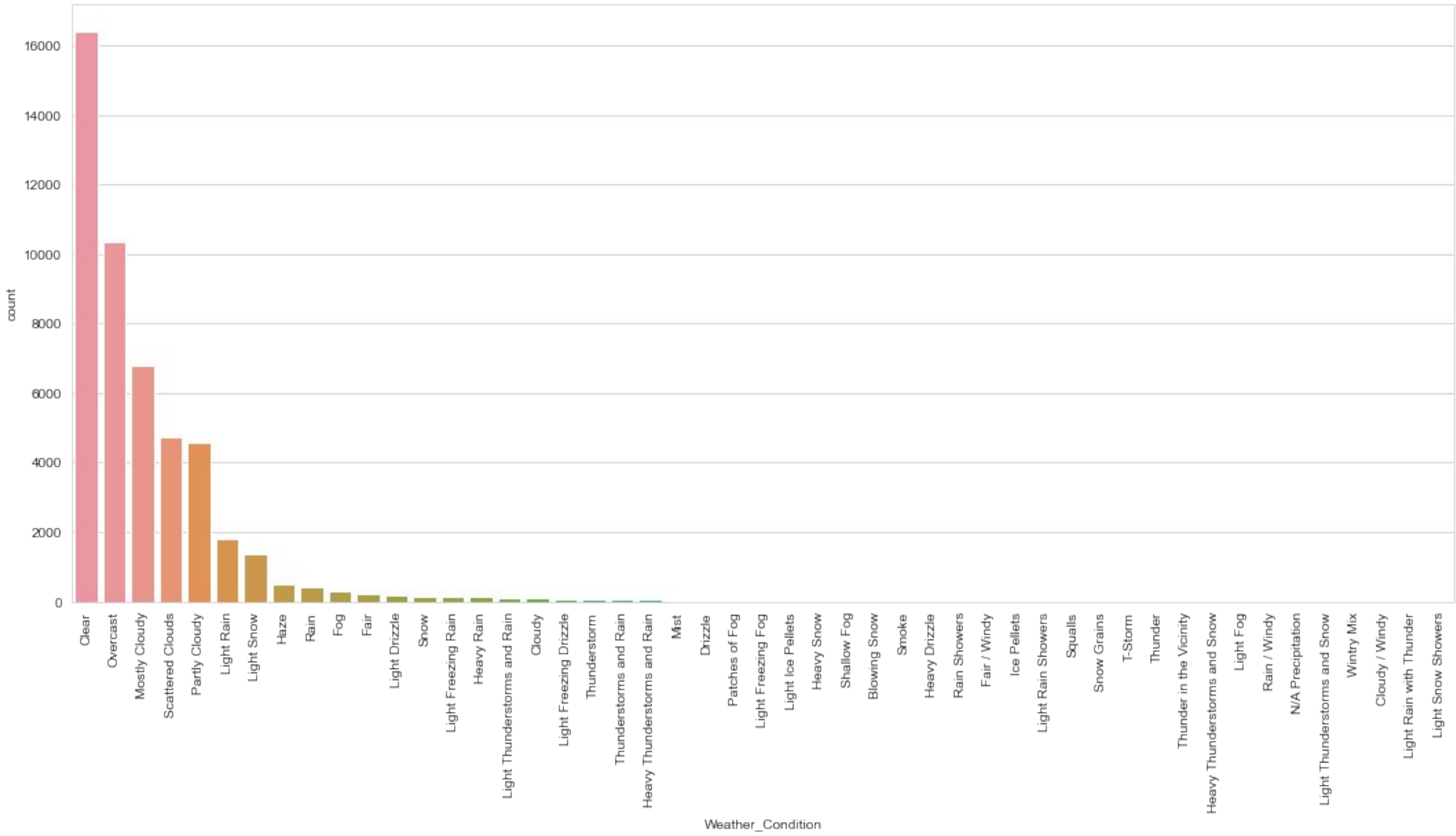
Example of structural error = Weather_condition Column:

```
['Light Rain', 'Overcast', 'Mostly Cloudy', 'Snow', 'Light Snow', 'Cloudy', nan, 'Scattered Clouds',
'Clear', 'Partly Cloudy', 'Light Freezing Drizzle', 'Light Drizzle', 'Haze', 'Rain', 'Heavy Rain', 'Fair',
'Drizzle', 'Fog', 'Thunderstorms and Rain', 'Patches of Fog', 'Light Thunderstorms and Rain', 'Mist',
'Rain Showers', 'Light Rain Showers', 'Heavy Drizzle', 'Smoke', 'Light Freezing Fog', 'Light Freezing
Rain', 'Blowing Snow','Heavy Thunderstorms and Rain', 'Heavy Snow', 'Snow Grains', 'Squalls', 'Light Fog',
'Shallow Fog', 'Thunderstorm', 'Light Ice Pellets', 'Thunder', 'Thunder in the Vicinity', 'Fair / Windy',
'Light Rain with Thunder','Heavy Thunderstorms and Snow', 'Light Snow Showers','Cloudy / Windy', 'Ice
Pellets', 'N/A Precipitation', 'Light Thunderstorms and Snow', 'T-Storm', 'Rain / Windy','Wintry Mix']
```

Solution:

If we plot a bar graph(mentioned in next slide) for all the categories present above, we will see that only few frequency have the high frequency, whereas most of the categories are rarely occurring. So, we combine the rarely occurring categories into one category and named it as ‘Other’.

```
a = df['Weather_Condition'].value_counts()
m = df['Weather_Condition'].isin(a.index[a < 1000])
df.loc[m, 'Weather_Condition'] = 'Other'
```



count
16000
14000
12000
10000
8000
6000
4000
2000
0
Clear
Overcast
Mostly Cloudy
Scattered Clouds
Partly Cloudy
Light Rain
Light Snow
Haze
Rain
Fog
Fair
Light Drizzle
Snow
Light Freezing Rain
Heavy Rain
Light Thunderstorms and Rain
Cloudy
Light Freezing Drizzle
Thunderstorm
Thunderstorms and Rain
Heavy Thunderstorms and Rain
Mist
Drizzle
Patches of Fog
Light Freezing Fog
Light Ice Pellets
Heavy Snow
Shallow Fog
Blowing Snow
Smoke
Heavy Drizzle
Rain Showers
Fair / Windy
Ice Pellets
Light Rain Showers
Squalls
Snow Grains
T-Storm
Thunder
Thunder in the Vicinity
Heavy Thunderstorms and Snow
Light Fog
Rain / Windy
N/A Precipitation
Light Thunderstorms and Snow
Wintry Mix
Cloudy / Windy
Light Rain with Thunder
Light Snow Showers
Weather_Condition

Fixing Unwanted Outliers

We can remove outliers using the following methods:

1. Z-score
2. Inter-quartile range

We have used Inter-quartile method to remove the outliers from the data. We created this function to detect the outliers:

```
def detect_outliers(df, col):
    Q1 = np.nanpercentile(df[col], 25, interpolation= 'midpoint')
    Q3 = np.nanpercentile(df[col], 75, interpolation= 'midpoint')
    IQR = Q3 - Q1
    upper_bound = Q3 + 1.5 * IQR
    lower_bound = Q1 - 1.5 * IQR

    ls = df.index[(df[col] < lower_bound) | (df[col] > upper_bound)]
    return ls
```

Fixing Unwanted Outliers

Keep detecting outlier for every column and store its index in a list using a loop(code is given below).

Once the loop terminates, we get all the records where outliers are present. Now, drop those records from the dataframe.

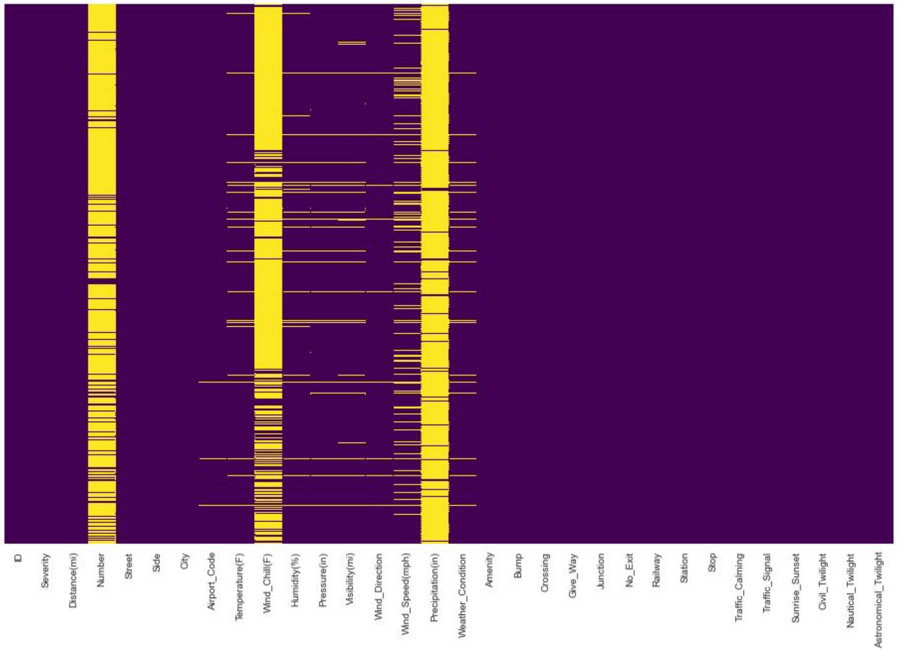
```
index_lst = []
for feature in features_lst:
    index_lst.extend(detect_outliers(df, feature))
```

Dealing with Missing Value

Seaborn library is used to visualize the missing values in the dataframe using a heatmap function

```
sns.heatmap(df_new.isnull(), yticklabels= False, cbar=False, cmap='viridis')
```

Output:



Yellow lines represent missing values here

Dealing with Missing Value

To get the exact values numerically, used the following code:

```
for col in df_new.columns:
    total_na = df_new[col].isna().sum()
    if total_na != 0:
        print(f'{col}: {total_na}')
```

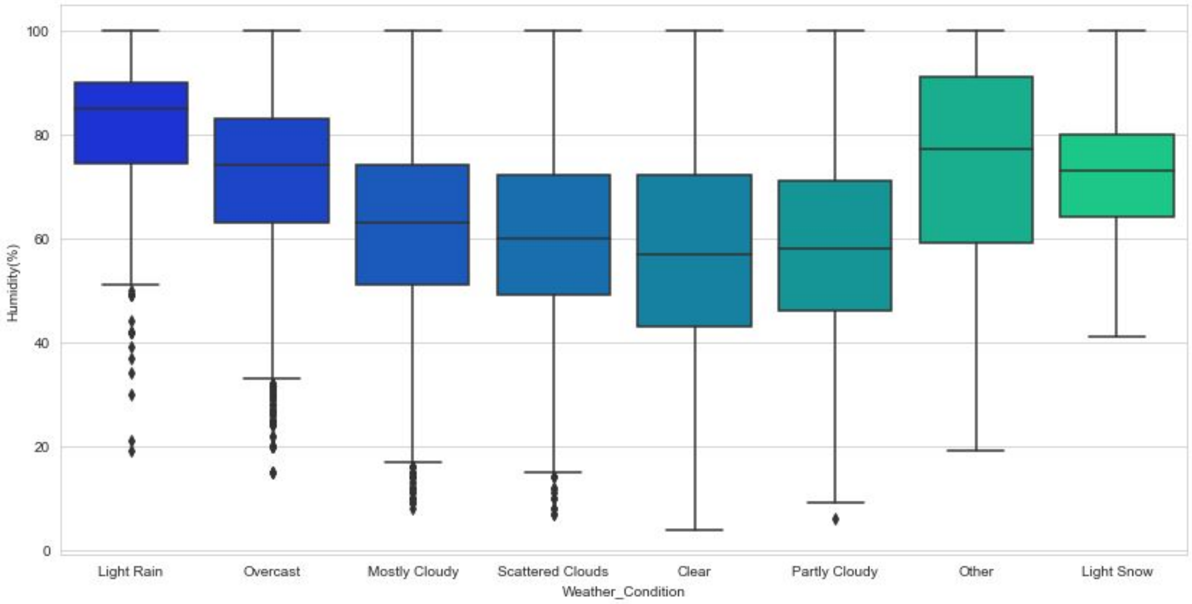
Output:

Number: 26790
Airport_Code: 40
Temperature(F): 1078
Wind_Chill(F): 26003
Humidity(%): 1102
Pressure(in): 1037
Visibility(mi): 1289
Wind_Direction: 403
Wind_Speed(mph): 4996
Precipitation(in): 30980
Weather_Condition: 1192

Imputing Missing Value

HUMIDITY COLUMN

To impute values in Humidity, we made a box plot of Humidity vs Weather_condition.



OBSERVATION from BOX PLOT:

Humidity is changing with respect to weather_conditions.

Ex. if there is Light_rain, humidity is high and vice versa.
If the sky is clear, then humidity is low.

Using this, we impute the missing values of Humidity column by using Weather_condition column

```
def impute_humidity(cols):
    Humidity = cols[0]
    Weather_Cond = cols[1]
    if pd.isnull(Humidity):
        if Weather_Cond == 'Light Rain':
            return 85.
        elif Weather_Cond == 'Overcast':
            return 75.
        elif Weather_Cond == 'Mostly Cloudy':
            return 64.
        elif Weather_Cond == 'Scattered Clouds':
            return 60.
        elif Weather_Cond == 'Partly Cloudy':
            return 58.
        elif Weather_Cond == 'Clear':
            return 56.
        elif Weather_Cond == 'Light Snow':
            return 72.
        else:
            return 77.
    else:
        return Humidity
```

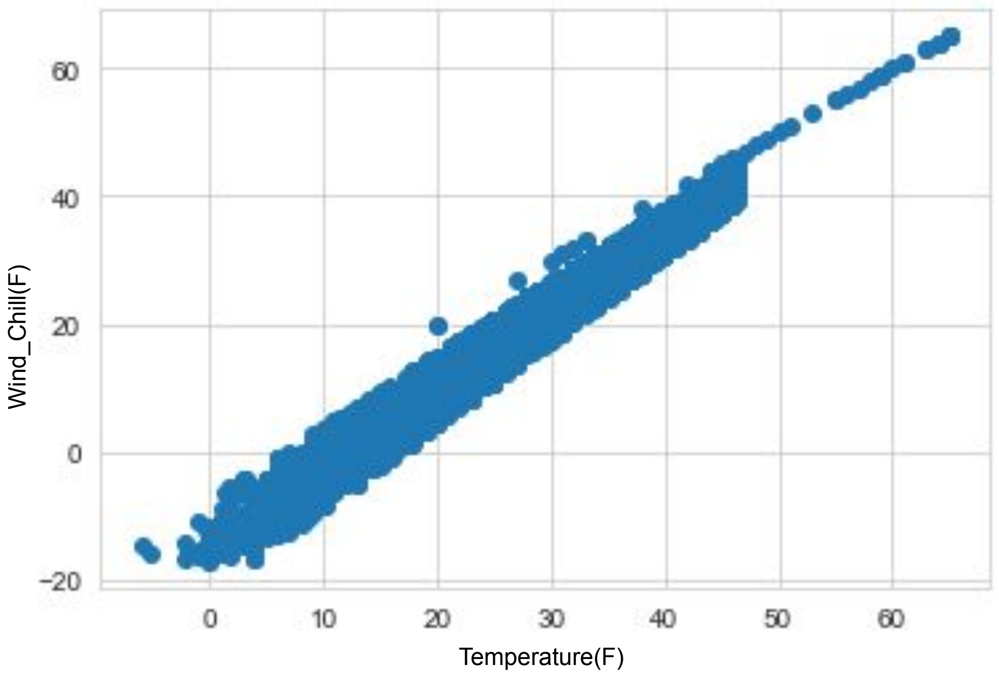
Imputing Missing Value

Wind_Chill(F) COLUMN

Upon finding the correlation between Wind_Chill(F) and Temperature(F) columns, it comes out to be 0.986486.

It means one column is redundant and can be dropped.

Hence, we dropped Wind_Chill(F) column.



Imputing Missing Value

Numbers and Precipitation(in) COLUMN

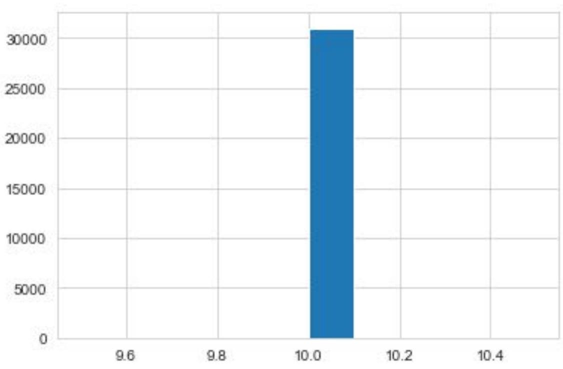
Upon checking the number of missing values in Numbers column, we get to know that 82.6% and 95.7% data is missing in Numbers and Precipitation(in) column respectively.

Hence, we dropped Numbers and Precipitation(in) column.

Imputing Missing Value

Visibility COLUMN

Histogram for visibility column



Description for visibility column

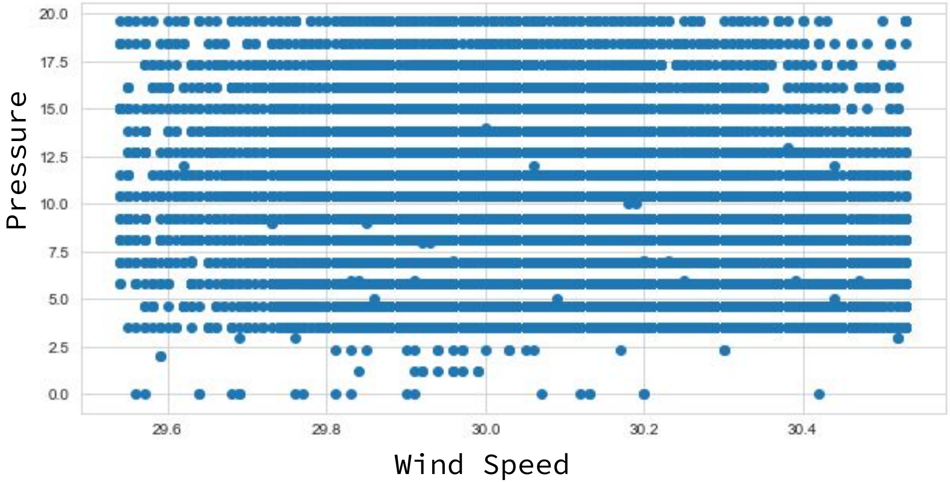
count	30972.0
mean	10.0
std	0.0
min	10.0
25%	10.0
50%	10.0
75%	10.0
max	10.0

Here, visibility is equal to 10 miles which is roughly 16 km, which is enough for a person to avoid the accident.
So, this column is of no use. Hence, we remove this column from dataframe.

Imputing Missing Value

Pressure COLUMN

Scatter plot for pressure vs wind_speed column




```
df_new['Pressure(in)'].corr(df_new['Wind_Speed(mph)'])
```

output

-0.12

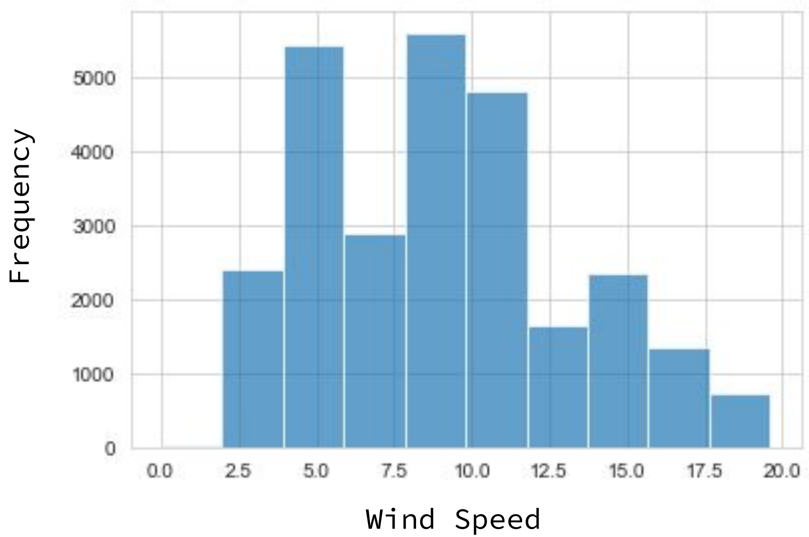
Here, visibility is equal to 10 miles which is roughly 16 km, which is enough for a person to avoid the accident. So, this column is of no use. Hence, we remove this column from dataframe.

Imputing Missing Value

Wind_Speed COLUMN

As we can see the Normal distribution in the histogram, it indicates that we can use the mean value to impute the missing values in this column

Histogram of wind_speed column




```
df_new['Wind_Speed(mph)'].fillna(df_new['Wind_Speed(mph)'].mean(), inplace = True)
```

Imputing Missing Value

Wind_Direction COLUMN

Since this is a categorical feature, we can not use mean to impute the missing values.

Instead, we use mode to impute the missing values of categroical data.

```
df_new['Wind_Direction'].fillna(df_new['Wind_Direction'].mode()[0], inplace = True)
```

One Hot Encoding

Street COLUMN

Street column has 6215 unique values. If we directly perform one-hot encoding then it will lead to too many features and we may end up with **Curse of Dimensionality** problem.

So, to get rid of this problem we do one hot encoding to the top ‘n’ occurring values. Remaining values will be assigned to ‘n + 1’th category and will be dropped eventually.

```
top_10_streets = [x for x in df_new.Street.value_counts().sort_values(ascending=False).head(10).index]


def one_hot_top_x(df, col, top_x_labels):
    for label in top_x_labels:
        df[col + '_' + label] = np.where(df_new[col]==label,1,0)
```

Similar process is followed for City, Airport_Code, Wind_Direction, Weather_conditon

Proposed solution & Modeling

We have used following ML models for predicting the accident severity and compared their performance.

- Decision Tree
- Random Forest
- Support vector machine
- Logistic Regression

Decision Tree classification accuracy

<u>Without Optimization</u>

Classification report -

	precision	recall	f1-score	support
2	0.79	0.77	0.78	4513
3	0.31	0.32	0.32	1275
4	0.35	0.39	0.37	440
accuracy			**0.65**	6228
macro avg	0.48	0.49	0.49	6228
weighted avg	0.66	0.65	0.65	6228

<u>With Optimization</u>

Classification report :

	precision	recall	f1-score	support
2	0.74	0.99	0.84	4513
3	0.64	0.02	0.04	1275
4	0.58	0.12	0.21	440
accuracy			**0.73**	6228
macro avg	0.65	0.38	0.36	6228
weighted avg	0.70	0.73	0.64	6228

Random forest classification accuracy

Accuracy = **75.59 %**

	precision	recall	f1-score	support
2	0.771134	0.960115	0.855310	4513
3	0.562353	0.187451	0.281176	1275
4	0.739130	0.309091	0.435897	440
accuracy			**0.755941**	6228
macro avg	0.690872	0.485552	0.524128	6228
weighted avg	0.726131	0.755941	0.708142	6228

SVM classification accuracy

Accuracy = **73 %**

	precision	recall	f1-score	support
2	0.74	0.98	0.84	4513
3	0.51	0.07	0.12	1275
4	0.59	0.13	0.22	440
accuracy			**0.73**	6228
macro avg	0.61	0.39	0.39	6228
weighted avg	0.68	0.73	0.65	6228

In [ ]:

Logistic regression classification accuracy

Accuracy **73.43 %**

	precision	recall	f1-score	support
2	0.745393	0.976955	0.845608	4513
3	0.530516	0.088627	0.151882	1275
4	0.510000	0.115909	0.188889	440
accuracy			**0.734265**	6228
macro avg	0.595303	0.393831	0.395460	6228
weighted avg	0.684773	0.734265	0.657192	6228

Conclusion

Accuracy of the following classifiers:

- Decision tree = 73%
- Random Forest = 75.59%
- SVM = 73&
- Logistic regression = 73.43%

After comparing accuracy of all the classifiers, we find that all the classifier have almost same accuracy but Random Forest has the highest accuracy of 75.59%

Contribution of each Group members

Data cleaning & processing = Abhijendra (213310010) & Uddesh (213310004)

Modelling = Amol (213310001), Shubham (213310005), Swabhiman (213310018)

All contributed equally for making presentation and helped each other in all stages of the project.

References

- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://www.geeksforgeeks.org/
- https://machinelearningmastery.com/https://machinelearningmastery.com/
- https://stackoverflow.com/

Thank You