CSE 4/587 Fall 2022
Project Phase #1

Team
Name: Raakhal Rapolu
UBIT Name: raakhalr
Name: Kajol
UBIT Name: kajol

Deliverables [50 marks]
1. Problem Statement: Form a title and problem statement that clearly state the problem and questions you are trying to answer. Additionally:
a. [5 marks] Discuss the background of the problem leading to your objectives. Why is it a significant problem?
b. [5 marks] Explain the potential of your project to contribute to your problem domain. Discuss why this contribution is crucial?

Title : Analysis of Car Accidents from 2016-2021

Problem Statement: We will do the analysis and look for primary contributing elements to accidents, and will try to infer the severity from them.

a) Driving around is an aspect of life in a country that runs on wheels. What would you least want to happen if you had just suited up and were heading out on your first interview? What is the one thing you must certainly avoid if your family has just embarked on a road trip to a national park you've always wanted to visit? The answer is: Car Crash/Accidents. First, we want to research and comprehend the nature of traffic accidents and how they have evolved through time. Where do mishaps occur and what are the primary reasons for car accidents. Whether we can gauge how serious an accident will be and stop it before it happens.

b) There are trends in the times, locations, and meteorological conditions when the majority of incidents happened. With the use of several classification machine learning algorithms, the seriousness of each accident may be anticipated rather correctly.

   The questions which will be answered and which will contribute to our problem are as follows:

   When do accidents typically occur?
   - Daylight vs darkness,

- In the middle of the week vs weekends
- Peak period

Where do accidents typically occur?

- County
- City
- Zip Code
- Street Side
- Location

What kind of weather conditions cause the most accidents?

- The worst weather for accidents
- The worst three weather conditions for accidents
- The severity of each accident's weather circumstances

2. Data Sources [5 marks]: Collect your data. Your data can come from multiple sources. For example, Medical, Bank, sports, health, Kaggle, Amazon reviews, Twitter, Youtube, Reddit, etc. This data has to be large enough for the data analysis to yield significance. At least 2000 rows.

Ans: We have taken the data from **Kaggle.** It is about **US Accidents (2016 - 2021).**
Link to the dataset : https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

Description of dataset: This dataset on car accidents in the USA includes data from all 49 states. Kaggle used several APIs that offer streaming traffic incident (or event) data were used to gather the accident data, which covered the period from February 2016 to December 2021. The US and state departments of transportation, law enforcement organizations, traffic cameras, and traffic sensors embedded in the road networks are only a few of the organizations whose traffic data is broadcast through these APIs. This dataset currently contains roughly 2.8 million accident records.

Info on dataset:
There are 47 columns, 2845342 rows and data types are as follows:

- Bool (13 columns)
- float64 (13 columns)
- Int 64 (1 column)
- Object (20)

```
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 773.4+ MB
```

3. Data Cleaning/Processing [10 Marks]: Your dataset has to be cleaned and properly processed. Please submit a report where you explain each processing/cleaning step properly. We expect to see comments and markup for this step. In order to get full marks you must clearly document 7 (10 for 587 students) distinct processing/cleaning operations.

Data Cleaning

1. Handling missing values

Here, we will remove any incorrect or missing values from the dataset.
Action on Side column:

```
accident_data["Side"].value_counts()
```

```
R    2353309
L     492032
N          1
Name: Side, dtype: int64
```

Here we observed that there is one row without side value as R/L therefore we can drop it.
After drop operation:

```
accident_data = accident_data.drop(accident_data[accident_data.Side == 'N'].index)
accident_data["Side"].value_counts()
```

```
R    2353309
L     492032
Name: Side, dtype: int64
```

Also dropped ID and Description column as they are not providing information on the accident which can be feeded into the models in the future.

```
accidents.drop(['ID', 'Description'], axis=1, inplace=True)
pd.set_option('display.max_columns', 500)
accidents.sample(3)
```

[4]:

	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	Street	Side	City	County
590722	2	2021-08-02 10:09:46	2021-08-02 12:00:47	37.932798	-75.547519	37.936624	-75.546019	0.277	7661.0	Lankford Hwy	R	Oak Hall	Accomack
99922	2	2016-05-19 18:09:24	2016-05-20 00:09:24	40.760280	-73.967510	40.760930	-73.967030	0.051	199.0	E 57th St	L	New York	New York
2728742	4	2017-11-11 01:56:23	2017-11-11 07:56:23	44.374658	-91.307101	44.374620	-91.310811	0.183	NaN	N36964 US Highway 53 121	R	Whitehall	Trempealeau

2. Handling data with a high cardinality

- Data can be combined or binned. By assembling them into coherent sets, the goal is to lower the number of unique values.
- As long as the groupings do not significantly affect model performance and business stakeholder explainability, this approach is excellent.

```
#Values of Weather_Condition
print(accidents.Weather_Condition.unique())
```

```
['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' 'Partly Cloudy / Windy'
 'Heavy T-Storm' 'Sand' 'Light Rain / Windy' 'Widespread Dust'
 'Mostly Cloudy / Windy' 'Blowing Dust / Windy' 'Blowing Dust'
 'Volcanic Ash' 'Freezing Rain / Windy' 'Small Hail' 'Wintry Mix / Windy'
 'Light Snow / Windy' 'Heavy Ice Pellets' 'Heavy Snow / Windy'
 'Heavy Rain / Windy' 'Heavy T-Storm / Windy' 'Fog / Windy' 'Dust Whirls'
 'Showers in the Vicinity' 'Funnel Cloud' 'Thunder / Windy' 'Snow / Windy'
 'Haze / Windy' 'Light Snow and Sleet' 'T-Storm / Windy'
 'Sand / Dust Whirlwinds' 'Light Snow with Thunder' 'Rain Shower'
 'Blowing Snow / Windy' 'Light Rain Shower' 'Snow and Sleet'
 'Drizzle and Fog' 'Light Sleet' 'Drizzle / Windy' 'Light Snow Shower'
 'Snow and Thunder / Windy' 'Light Sleet / Windy' 'Smoke / Windy'
 'Widespread Dust / Windy' 'Light Drizzle / Windy' 'Tornado'
 'Squalls / Windy' 'Hail' 'Blowing Snow Nearby' 'Partial Fog'
 'Sand / Windy' 'Thunder / Wintry Mix' 'Light Freezing Rain / Windy'
 'Duststorm' 'Light Snow and Sleet / Windy' 'Heavy Rain Shower / Windy'
 'Sand / Dust Whirlwinds / Windy' 'Light Rain Shower / Windy'
 'Thunder and Hail' 'Freezing Rain' 'Heavy Sleet' 'Sleet'
 'Freezing Drizzle' 'Snow and Sleet / Windy' 'Heavy Freezing Drizzle'
 'Heavy Freezing Rain' 'Blowing Sand' 'Thunder / Wintry Mix / Windy'
 'Mist / Windy' 'Sleet / Windy' 'Patches of Fog / Windy'
 'Sand / Dust Whirls Nearby' 'Heavy Rain Shower' 'Drifting Snow'
 'Heavy Blowing Snow' 'Low Drifting Snow' 'Light Blowing Snow'
 'Heavy Rain Showers' 'Light Haze' 'Heavy Thunderstorms with Small Hail'
 'Heavy Snow with Thunder' 'Thunder and Hail / Windy']
```

It would be better to have fewer distinct weather situations because, as we can see from analysis of the weather, there are many of them.

[6]:

```
accidents.loc[accidents["Weather_Condition"].str.contains("N/A Precipitation", na=False), "Weather_Con
accidents.loc[accidents["Weather_Condition"].str.contains("Snow|Sleet|Wintry", na=False), "Weather_Con
accidents.loc[accidents["Weather_Condition"].str.contains("Smoke|Volcanic Ash", na=False), "Weather_Co
accidents.loc[accidents["Weather_Condition"].str.contains("Cloud|Overcast", na=False), "Weather_Condit
accidents.loc[accidents["Weather_Condition"].str.contains("Sand|Dust", na=False), "Weather_Condition"]
accidents.loc[accidents["Weather_Condition"].str.contains("Wind|Squalls", na=False), "Weather_Conditio
accidents.loc[accidents["Weather_Condition"].str.contains("Mist|Haze|Fog", na=False), "Weather_Conditi
accidents.loc[accidents["Weather_Condition"].str.contains("Thunder|T-Storm", na=False), "Weather_Condi
accidents.loc[accidents["Weather_Condition"].str.contains("Rain|Drizzle|Shower", na=False), "Weather_C
accidents.loc[accidents["Weather_Condition"].str.contains("Hail|Pellets", na=False), "Weather_Conditio
accidents.loc[accidents["Weather_Condition"].str.contains("Fair", na=False), "Weather_Condition"] = "C

print('Unique values:',accidents["Weather_Condition"].unique())
```

```
Unique values: ['Rain' 'Cloudy' 'Snow' nan 'Clear' 'Fog' 'Thunderstorm' 'Smoke' 'Windy'
 'Hail' 'Sand' 'Tornado']
```

3. One-hot Encoding

- We converted bool values to 0s and 1s as will be easier to feed the models we make in phase2.
- One hot encoding improves predictions and classification accuracy of a model, is the crucial process of changing the categorical data variables to be fed to machine and deep learning algorithms.

```
[9]: accidents = accidents.replace([True, False], [1,0])
```

```
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                int64
 31  Bump                   int64
 32  Crossing               int64
 33  Give_Way               int64
 34  Junction               int64
 35  No_Exit                int64
 36  Railway                int64
 37  Roundabout             int64
 38  Station                int64
 39  Stop                   int64
 40  Traffic_Calming        int64
 41  Traffic_Signal         int64
 42  Turning_Loop           int64
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: float64(13), int64(14), object(20)
memory usage: 1020.3+ MB
```

nsole

No bool values after the operation.

4. Remove Duplicate Data

```
#Drop duplicates
print("Total Rows:", len(accidents.ID))
Duplicate_Data = accidents[accidents.duplicated()]
print("Duplicate Rows :{}".format(Duplicate_Data))
accidents.drop_duplicates(inplace=True)
print("After drop number of rows:", len(accidents.ID))
```

```
Total Rows: 2845342
Duplicate Rows :Empty DataFrame
Columns: [ID, Severity, Start_Time, End_Time, Start_Lat, Start_Lng, End_Lat, End_Lng, Distance(mi), Description, Number, Str
eet, 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, B
ump, Crossing, Give_Way, Junction, No_Exit, Railway, Roundabout, Station, Stop, Traffic_Calming, Traffic_Signal, Turning_Loo
p, Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight]
Index: []

[0 rows x 47 columns]
After drop number of rows: 2845342
```

We did not have any duplicates but this was fail safe cleaning as Duplicate data sets have the potential to contaminate the training data with the test data, or the other way around. This can make the model bias towards one feature.

5. NA value handling (Imputation by mean value)

- The observed data's mean is preserved when the mean is imputed. Therefore, even if all of the data are randomly missing, the mean estimate is still fair. That's advantageous. Additionally, we can retain our sample size at the full sample size by imputing the mean.
- Since filling them with zero doesn't make much sense, we populated the following columns with their average value.

```
accidents['Wind_Speed(mph)']=accidents['Wind_Speed(mph)'].fillna(accidents['Wind_Speed(mph)'].mean())
accidents['Humidity(%)']=accidents['Humidity(%)'].fillna(accidents['Humidity(%)'].mean())
accidents['Temperature(F)']=accidents['Temperature(F)'].fillna(accidents['Temperature(F)'].mean())
accidents['Pressure(in)']=accidents['Pressure(in)'].fillna(accidents['Pressure(in)'].mean())
accidents.dropna(inplace=True)
```

```
Total NA values: ID                                    0
```

6. NA value handling (Imputation by median value)

When the data is skewed, it is wise to think about replacing the missing values with the median value. Keep in mind that only numerical data can be used to impute missing data using the median value.
Because it lessens the impact of outliers, using the median to impute is more reliable.

```
accidents['Visibility(mi)']=accidents['Visibility(mi)'].fillna(accidents['Visibility(mi)'].median())
accidents['Wind_Chill(F)']=accidents['Wind_Chill(F)'].fillna(accidents['Wind_Chill(F)'].median())
accidents['Precipitation(in)']=accidents['Precipitation(in)'].fillna(accidents['Precipitation(in)'].median())
accidents.dropna(inplace=True)
```

+ Code + Markdown

7. Impute empty value with most occurring value (or mode)

The mode value, or most frequent value, of the full feature column is substituted for the missing values in mode imputation, a further approach. It is wise to think about utilizing mode values to replace missing values when the data is skewed. You might think about substituting values for data points like the city field with mode. It should be noted that both numerical and categorical data can be used to impute missing data using mode values.

```
accidents['City'] = accidents.groupby('State')['City'].transform(lambda data: data.fillna(data.value_counts().inde
```

+ Code + Markdown

8. Scaling of feature

We will scale and normalize the characteristics in this part.

- We normalized the values of the continuous features to enhance the performance of our models.

```
ss = MinMaxScaler()
attributes = ['Wind_Speed(mph)','Pressure(in)','Humidity(%)','Distance(mi)','Visibility(mi)','Temperat
accidents[attributes] = ss.fit_transform(accidents[attributes])
accidents.sample(3)
```

[50…

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Des
2173606	A-2173607	2	2020-06-15 13:34:50	2020-06-15 14:34:50	0.773152	0.889753	43.457640	-73.445210	0.000000	14 - A
1359844	A-1359845	2	2021-03-16 05:33:00.000000000	2021-03-16 12:44:30.000000000	0.076525	0.744794	26.432800	-81.777114	0.002835	St t fro
716033	A-716034	2	2021-11-19 06:01:30	2021-11-19 07:47:30	0.386777	0.110990	34.027192	-118.209601	0.014157	Slo o Si Fw

3 rows × 47 columns

9. Adding of feature

- This dataset included complex features, which are combinations of two or more features. By locating prospective information, extracting and developing more meaningful features will enhance model performance.
- We made the decision to divide the Start Time feature (so that we can feed it to the model later) into the following components: year, month, day, weekday, hour, and minute.

```
accidents['M'] = pd.to_datetime(accidents.Start_Time).dt.strftime('%m')
accidents['Y'] =  pd.to_datetime(accidents.Start_Time).dt.strftime('%Y')
accidents['D'] = pd.to_datetime(accidents.Start_Time).dt.strftime('%w')
accidents['H'] = pd.to_datetime(accidents.Start_Time).dt.strftime("%H")
accidents.sample(2)
```

[6]:

	Distance(mi)	Description	...	Traffic_Signal	Turning_Loop	Sunrise_Sunset	Civil_Twilight	Nautical_Twilight	Astronomical_Twilight	M	Y	D	H
3	5.936	Stationary traffic on Pacific Highway 1 S - I-...	...	False	False	Day	Day	Day	Day	05	2021	1	15
)	0.048	Incident on NW 21ST AVE near NW 52ND ST Drive ...	...	False	False	Day	Day	Day	Day	01	2021	5	17

10. Examine feature variance

- This part will examine each feature's variance in order to eliminate features with very low variances because they cannot aid in instance discrimination.

```
pd.set_option('display.max_columns', None)
accidents.describe()
accidents.round(2)
```

[52…

ort_Code	Weather_Timestamp	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Direction	Wind_Speed(mp
KOSU	2016-02-08 00:53:00	0.46	36.1	0.58	0.51	0.07	SW	0.
KFFO	2016-02-08 05:58:00	0.44	NaN	0.91	0.50	0.07	Calm	Na
KLUK	2016-02-08 05:53:00	0.44	NaN	0.97	0.50	0.07	Calm	Na
KAKR	2016-02-08 06:54:00	0.45	NaN	0.55	0.50	0.07	Calm	Na
KLUK	2016-02-08 07:53:00	0.44	29.8	0.93	0.50	0.07	WSW	0.
...	...	...	...	...	...	...	...	
KRAL	2019-08-23 17:53:00	0.61	86.0	0.39	0.49	0.07	W	0.
KMYF	2019-08-23 18:53:00	0.56	70.0	0.73	0.50	0.07	SW	0.
KSNA	2019-08-23 18:53:00	0.57	73.0	0.64	0.50	0.07	SSW	0.

- Precipitation and pressure don't need to be dropped despite the modest variance they have because their increments are often minimal.

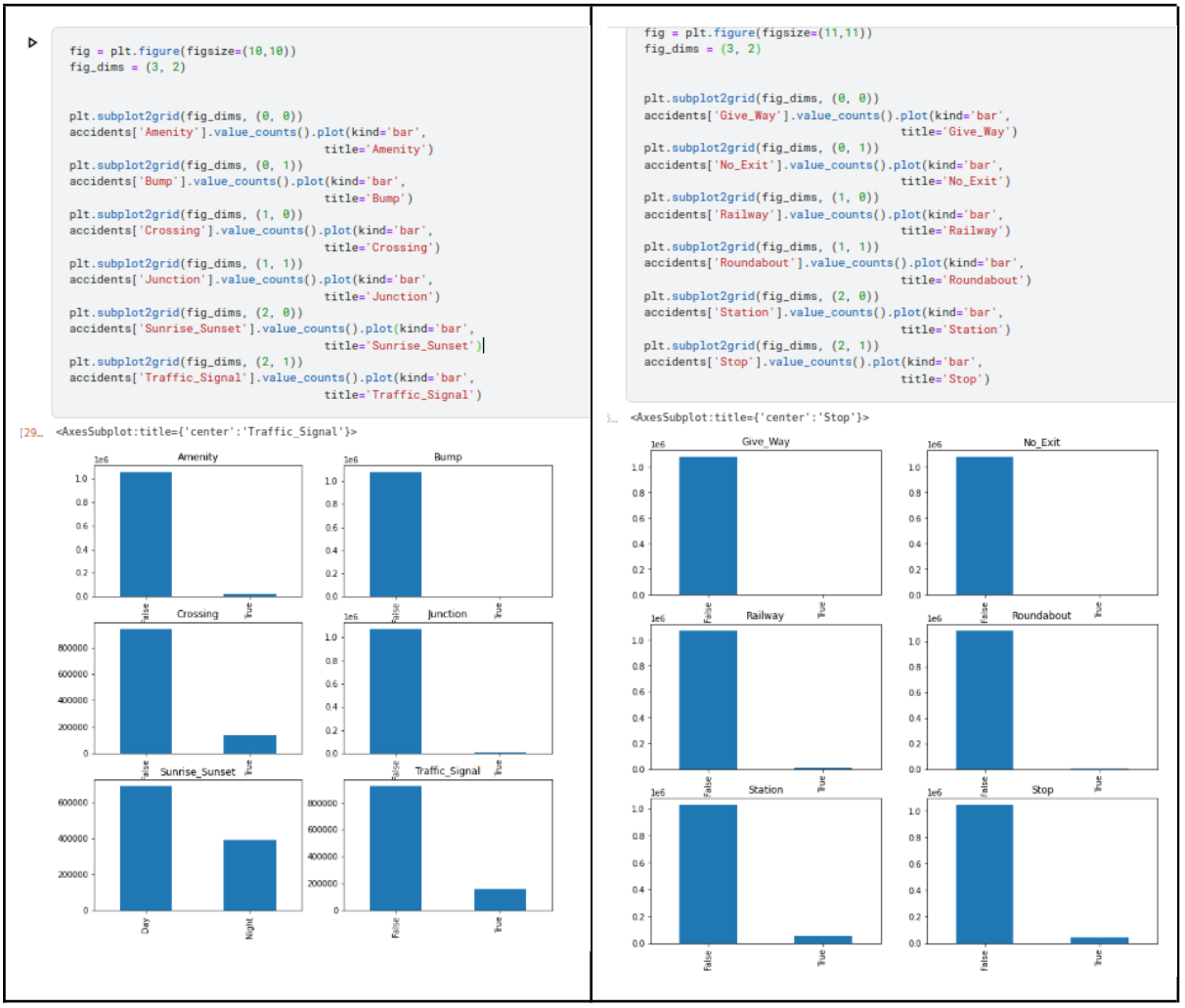
4. Exploratory Data Analysis (EDA) [25 Marks]: Perform exploratory data analysis as
defined in the NIST publication [2] and as originally described by John Tukey [4,5].
Record the outcomes and what you learned and how you will use this information. For example, in choosing features (columns) and dropping columns, and in short feature engineering. You need to demonstrate 7 (10 for 587 students) different, significant and relevant EDA operations and describe how you used these to process the data sets further to provision them for downstream modeling and analytics. Figures and tables should be included where relevant.

Ans:

Exploratory Data Analysis

1. Data Distribution

Analysing distribution of data, balanced or imbalance data and its attributes in boolean type of data



- The above bar plot is to illustrate the distribution of the boolean attribute,
- Here it is observed that for columns like ‘Amenity, Crossing, Sunrise_sunset, traffic_signal, Station and stop have some significant impact other rest other fields are significantly less compared with the whole data for 5 years.
- Here we can drop the attributes with highly imbalanced data, or we can also use the information by sub dividing into year wise and analysing the same distribution further.

2. Correlation between the attributes

Analysing the importance of attributes based on the correlation metric

```
import matplotlib.pyplot as plt

corr = accidents.corr()
corr.style.background_gradient(cmap='YlOrRd')
```

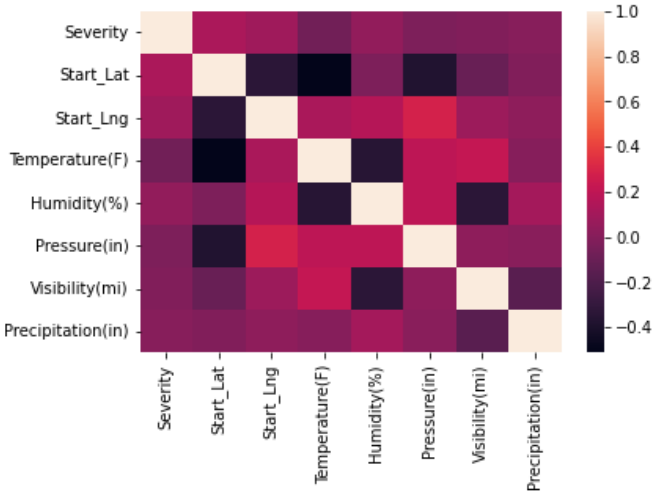
```
/opt/conda/lib/python3.7/site-packages/pandas/io/formats/style.py:2813: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
/opt/conda/lib/python3.7/site-packages/pandas/io/formats/style.py:2814: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
```

	Severity	Start_Lat	Start_Lng	Distance(mi)	Number	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Speed(mph)	Precipitation(in)	Amenity	Bump	Crossing	Give_Way	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal
Severity	1.000000	0.161870	0.106045	0.157531	-0.000812	-0.069488	-0.080722	0.034189	0.057275	-0.001228	0.035933	0.005289	-0.003061	-0.005477	-0.042683	0.004286	0.025837	-0.007018	-0.003091	-0.002375	-0.023957	-0.000402	-0.005317	-0.002052
Start_Lat	0.161870	1.000000	-0.325707	0.110992	0.098743	-0.287426	-0.277181	-0.026069	-0.165902	-0.045703	0.009667	-0.008930	0.005654	-0.002240	-0.089699	0.015639	0.028732	-0.018524	0.007983	-0.004686	-0.138413	0.035352	-0.004012	-0.001908
Start_Lng	0.106045	-0.325707	1.000000	-0.030439	-0.169348	0.067167	0.052543	0.109810	0.150879	0.030415	0.041200	0.017657	0.010605	-0.020844	0.054569	0.012004	-0.001508	0.002770	-0.018279	0.000860	0.053922	-0.077178	-0.015121	0.043335
Distance(mi)	0.157531	0.110992	-0.030439	1.000000	0.067449	-0.038827	-0.038074	0.009900	-0.032052	-0.010061	0.006221	0.000245	-0.021010	-0.004228	-0.052056	0.002310	0.005430	-0.005772	-0.006246	-0.001753	-0.033342	0.006793	-0.005149	-0.058872
Number	-0.000812	0.098743	-0.169348	0.067449	1.000000	-0.041328	-0.037169	0.012111	-0.024115	-0.014367	0.004928	-0.000733	-0.031943	-0.003680	-0.054011	-0.003823	-0.010874	-0.005198	-0.013399	-0.001358	-0.027944	-0.003965	-0.005029	-0.058123
Temperature(F)	-0.069488	-0.287426	0.067167	-0.038827	-0.041328	1.000000	0.964240	-0.356758	0.155972	0.207496	0.098716	-0.001445	0.008089	0.004222	0.049853	-0.005053	-0.010392	0.010592	0.001053	0.002235	0.045219	-0.005848	0.005945	0.021723
Wind_Chill(F)	-0.080722	-0.277181	0.052543	-0.038074	-0.037169	0.964240	1.000000	-0.323427	0.164171	0.199670	0.047615	-0.000469	0.008748	0.004488	0.045493	-0.005979	-0.012849	0.009979	0.001018	0.002205	0.044856	-0.004832	0.006232	0.011439
Humidity(%)	0.034189	-0.026069	0.109810	0.009900	0.012111	-0.356758	-0.323427	1.000000	0.167807	-0.340725	-0.180659	0.077261	-0.005815	-0.006913	-0.029309	0.000377	-0.000292	-0.009044	-0.000553	-0.000187	0.001870	-0.013911	-0.006833	-0.041066
Pressure(in)	0.057275	-0.165902	0.150879	-0.032052	-0.024115	0.155972	0.164171	0.167807	1.000000	0.033117	-0.034227	0.007389	0.013184	-0.007003	0.002801	-0.001095	0.015435	-0.004431	0.005084	0.000644	0.027227	-0.011604	-0.004858	-0.000145
Visibility(mi)	-0.001228	-0.045703	0.030415	-0.010061	-0.014367	0.207496	0.199670	-0.340725	0.033117	1.000000	0.044612	-0.117172	0.007676	0.001301	0.022434	0.000782	-0.001869	0.005345	0.001142	0.000319	0.015667	-0.001315	0.002196	0.015905
Wind_Speed(mph)	0.035933	0.009667	0.041200	0.006221	0.004928	0.098716	0.047615	-0.180659	-0.034227	0.044612	1.000000	0.023129	0.001751	-0.000423	0.019347	0.001538	0.007663	0.001748	0.000450	-0.000466	0.014217	-0.003025	-0.000108	0.025875
Precipitation(in)	0.005289	-0.008930	0.017657	0.000245	-0.000733	-0.001445	-0.000469	0.077261	0.007389	-0.117172	0.023129	1.000000	0.000057	-0.000714	0.003164	-0.001002	0.000166	0.000296	-0.000848	-0.000151	0.001995	-0.003218	-0.000267	0.000767
Amenity	-0.003061	0.005654	0.010605	-0.021010	-0.031943	0.008089	0.008748	-0.005815	0.013184	0.007676	0.001751	0.000057	1.000000	0.005596	0.114342	0.000403	-0.005284	0.012043	0.034036	-0.000497	0.119404	0.017325	0.011395	0.089732
Bump	-0.005477	-0.002240	-0.020844	-0.004228	-0.003680	0.004222	0.004488	-0.006913	-0.007003	0.001301	-0.000423	-0.000714	0.005596	1.000000	0.014576	0.000293	-0.000625	0.003775	0.003254	-0.000222	0.005102	0.022025	0.845122	-0.008197
Crossing	-0.042683	-0.089699	0.054569	-0.052056	-0.054011	0.049853	0.045493	-0.029309	0.002801	0.022434	0.019347	0.003164	0.114342	0.014576	1.000000	0.033026	-0.017232	0.044108	0.239614	-0.002950	0.130195	0.073264	0.021708	0.355996
Give_Way	0.004286	0.015639	0.012004	0.002310	-0.003823	-0.005053	-0.005979	0.000377	-0.001095	0.000782	0.001538	-0.001002	0.000403	0.000293	0.033026	1.000000	-0.001833	0.005706	0.004284	0.004214	-0.005193	0.047041	0.000475	0.029969
Junction	0.025837	0.028732	-0.001508	0.005430	-0.010874	-0.010392	-0.012849	-0.000292	0.015435	-0.001869	0.007663	0.000166	-0.005284	-0.000625	-0.017232	-0.001833	1.000000	-0.002095	0.000818	0.082391	-0.013099	-0.008160	-0.000817	-0.014149
No_Exit	-0.007018	-0.018524	0.002770	-0.005772	-0.005198	0.010592	0.009979	-0.009044	-0.004431	0.005345	0.001748	0.000296	0.012043	0.003775	0.044108	0.005706	-0.002095	1.000000	0.001118	-0.000421	0.012732	0.009908	0.002758	0.024270
Railway	-0.003091	0.007983	-0.018279	-0.006246	-0.013399	0.001053	0.001018	-0.000553	0.005084	0.001142	0.000450	-0.000848	0.034036	0.003254	0.239614	0.004284	0.000818	0.001118	1.000000	-0.000914	0.053182	0.003651	0.002093	0.052191
Roundabout	-0.002375	-0.004686	0.000860	-0.001753	-0.001358	0.002235	0.002205	-0.000187	0.000644	0.000319	-0.000466	-0.000151	-0.000497	-0.000222	-0.002950	0.004214	0.082391	-0.000421	-0.000914	1.000000	-0.000939	0.004779	0.003251	-0.003489
Station	-0.023957	-0.138413	0.053922	-0.033342	-0.027944	0.045219	0.044856	0.001870	0.027227	0.015667	0.014217	0.001995	0.119404	0.005102	0.130195	-0.005193	-0.013099	0.012732	0.053182	-0.000939	1.000000	0.009357	0.009542	0.093673
Stop	-0.000402	0.035352	-0.077178	0.006793	-0.003965	-0.005848	-0.004832	-0.013911	-0.011604	-0.001315	-0.003025	-0.003218	0.017325	0.022025	0.073264	0.047041	-0.008160	0.009908	0.003651	0.004779	0.009357	1.000000	0.020818	-0.063183
Traffic_Calming	-0.005317	-0.004012	-0.015121	-0.005149	-0.005029	0.005945	0.006232	-0.006833	-0.004858	0.002196	-0.000108	-0.000267	0.011395	0.845122	0.021708	0.000475	-0.000817	0.002758	0.002093	0.003251	0.009542	0.020818	1.000000	-0.001346
Traffic_Signal	-0.002052	-0.001908	0.043335	-0.058872	-0.058123	0.021723	0.011439	-0.041066	-0.000145	0.015905	0.025875	0.000767	0.089732	-0.008197	0.355996	0.029969	-0.014149	0.024270	0.052191	-0.003489	0.093673	-0.063183	-0.001346	1.000000

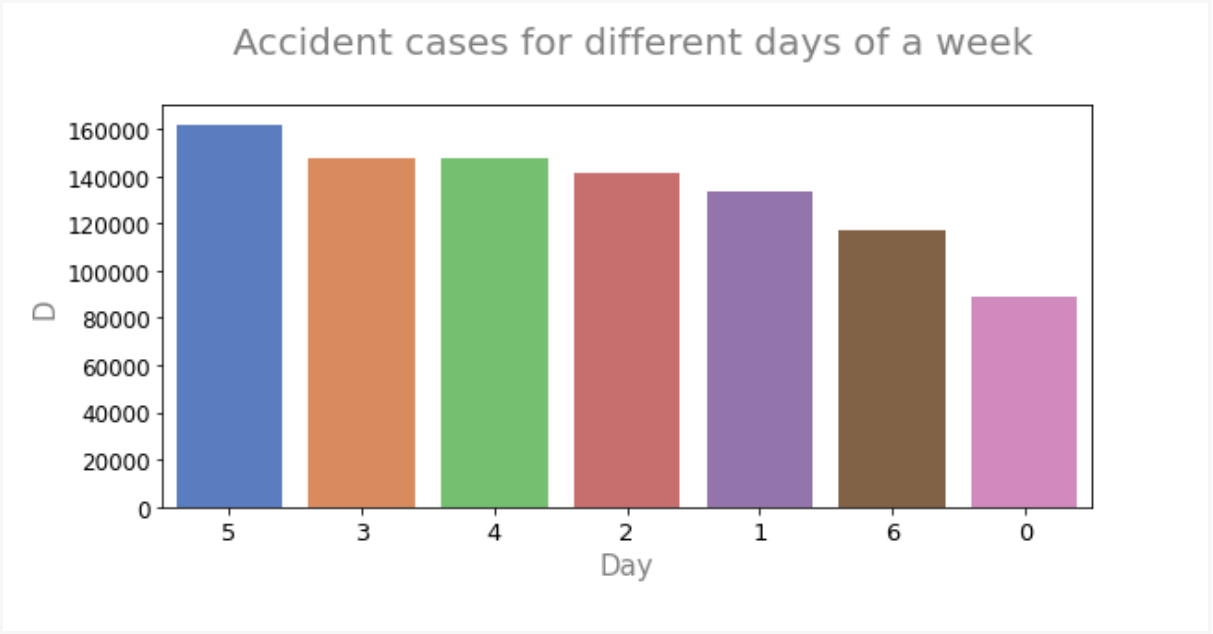
- The above image shows correlation between the attributes.
- Traffic signal and bump are highly correlated positively with a correlation of 0.849028
- Using this correlation matrix, we will be able to identify the dependency between the attributes and the factor of change of one attribute with respect to other attribute.

```
sns.heatmap(accidents[['Severity','Start_Lat','Start_Lng','T
```

```
<AxesSubplot:>
```



3 .Time analysis - weekday week ends



Sunday	17.24732031
Monday	15.7377416
Tuesday	15.73198798
Wednesday	15.09344301
Thursday	14.22038485
Friday	12.4667036
Saturday	9.50241865

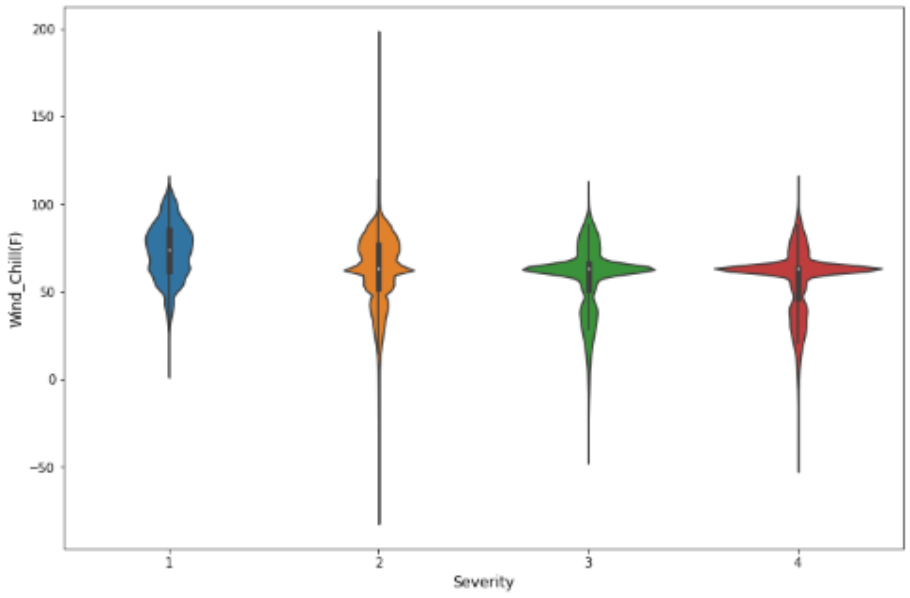
- From the above bar plot, it is observed that almost 160K accidents 17.24% in last 5 years happened on Friday followed by Wednesday and Thursday.
- Further narrowing it down to year wise and hour wise, we shall be able to understand during what time of the day on fridays most accidents happen and we can infer further from the analysis.
- This information can be used to segregate the data based on timeframe and gather the insights

4. Severity distribution Analysis

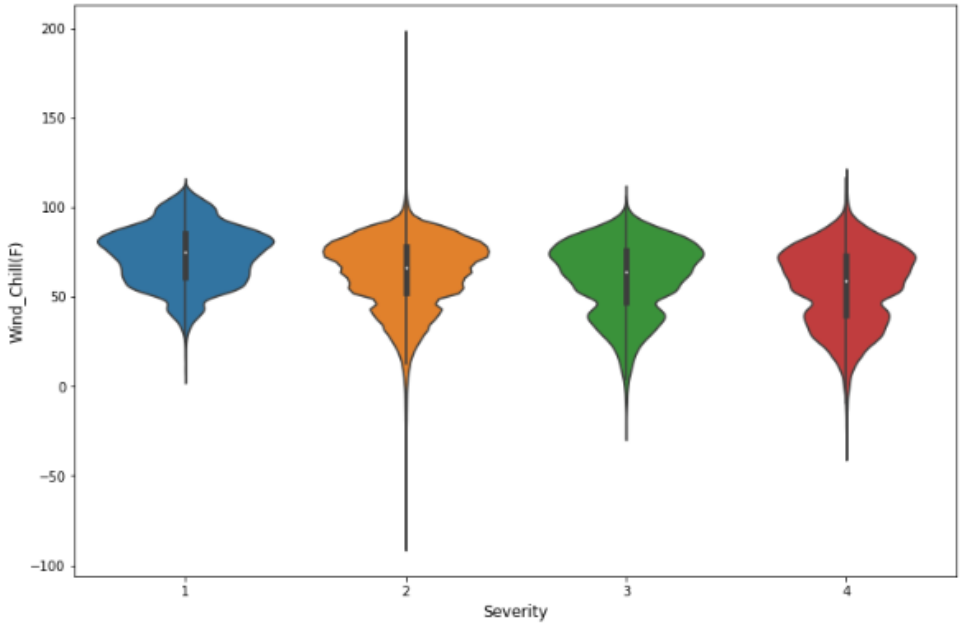
- Voilin plot is a kernel density plot which is a hybrid form of box plot, this is used to visualise the distribution of numerical data which shows the summary statistics and the density of each variable.
- Violin plot before performing the cleaning steps.

```
import seaborn as sns


plt.figure(figsize=(12,8))
sns.violinplot(x='Severity', y='Wind_Chill(F)', data=accidents)
plt.xlabel('Severity', fontsize=12)
plt.ylabel('Wind_Chill(F)', fontsize=12)
plt.show()
```



- Violin plot after performing data cleaning steps, change in data distribution is observed



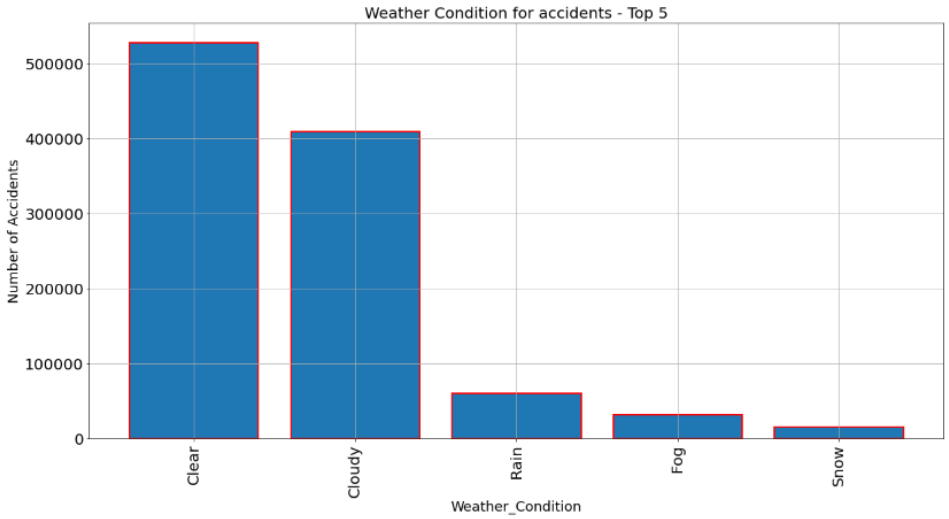
5. Weather Analysis

Analysis of Cause of accidents based on weather conditions

- From the above plot it is observed that 48 % accidents are happening in clear weather conditions
- 37% of the accidents are in cloudy weather condition followed by rain, fog and snow conditions

```
[28]: fig, ax=plt.subplots(figsize=(20,10))
accidents['Weather_Condition'].value_counts().sort_values(ascending=False).head(5).plot.bar(width=0.8,edgecolor='red',align='center',linewidth=2)
plt.xlabel('Weather_Condition',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=20)
plt.title('Weather Condition for accidents - Top 5',fontsize=20)
plt.grid()
plt.ioff()
```

```
[28… <matplotlib.pyplot._IoffContext at 0x7f1c560ba610>
```



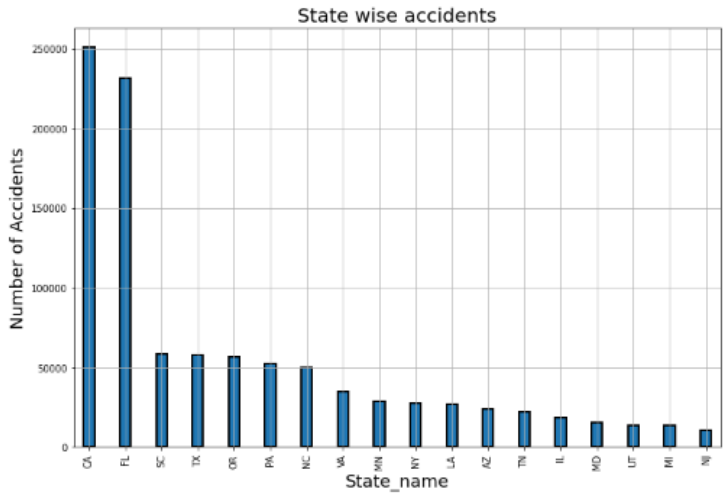
- Here, it is observed that weather condition has more importance, as it is distributed with respect to the number of accidents.
- This information is used further to do selective modelling, and use this time series data patterns.

6. Location Analysis

- The below plot shows the top 18 state wise accidents in last 5 years
- In California maximum number of accidents are seen with 23 % accidents
- Followed by Florida with 21% accidents during the last 5 years
- California is highly populated country, which is why we can see more accidents happening in that state

[31]:
```
plt.figure(figsize=(12,8))
accidents.State.value_counts().sort_values(ascending=False).head(18).plot.bar(width=0.3,edgecolor='k',align='center',linewidth=2)
plt.xlabel('State_name',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=20)
plt.title('State wise accidents',fontsize=20)
plt.grid()
plt.ioff()
```

[31]: `<matplotlib.pyplot._IoffContext at 0x7fd1cfaf1ld0>`



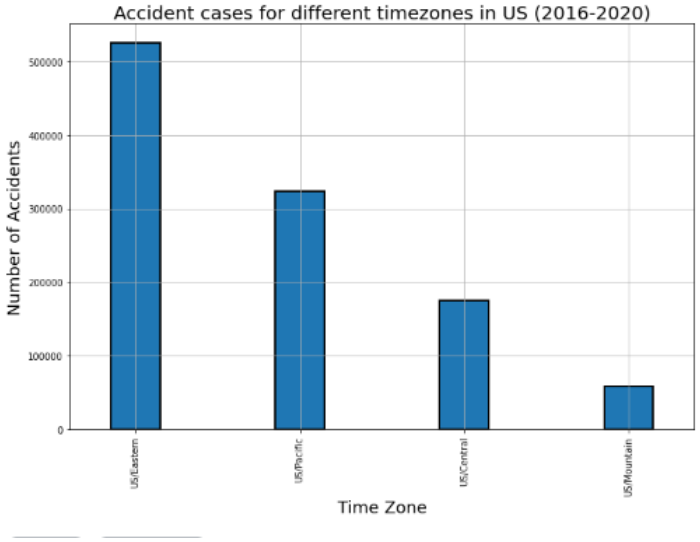
- This information is used to segregate the data state wise and make detailed analysis in which city more accidents are caused

7. Timezone Analysis

```
#df.Timezone.value_counts()

plt.figure(figsize=(12,8))
accidents.Timezone.value_counts().sort_values(ascending=False).head(18).plot.bar(width=0.3,edgecolor='k',align='center',linewidth=2)
plt.xlabel('Time Zone',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=30)
plt.title('Accident cases for different timezones in US (2016-2020)',fontsize=20)
plt.grid()
plt.ioff()
```

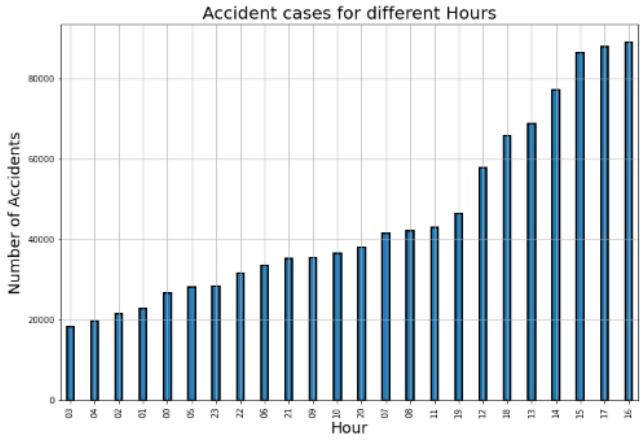
```
[40… <matplotlib.pyplot._IoffContext at 0x7fd1cfba0810>
```



- From the above visualisation, it is observed that most number of accident cases are reported in the US/Eastern timezone
- according to census data, largest population of the country resides in Eastern timezone.
- Number of accidents are then followed by US/Pacific, US/Central and US/Mountain

```
[51]:
plt.figure(figsize=(12,8))
accidents.H.value_counts().sort_values(ascending=True).head(24).plot.bar(width=0.3,edgecolor='k',align='center',linewidth=2)
plt.xlabel('Hour',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=30)
plt.title('Accident cases for different Hours',fontsize=20)
plt.grid()
plt.ioff()
```

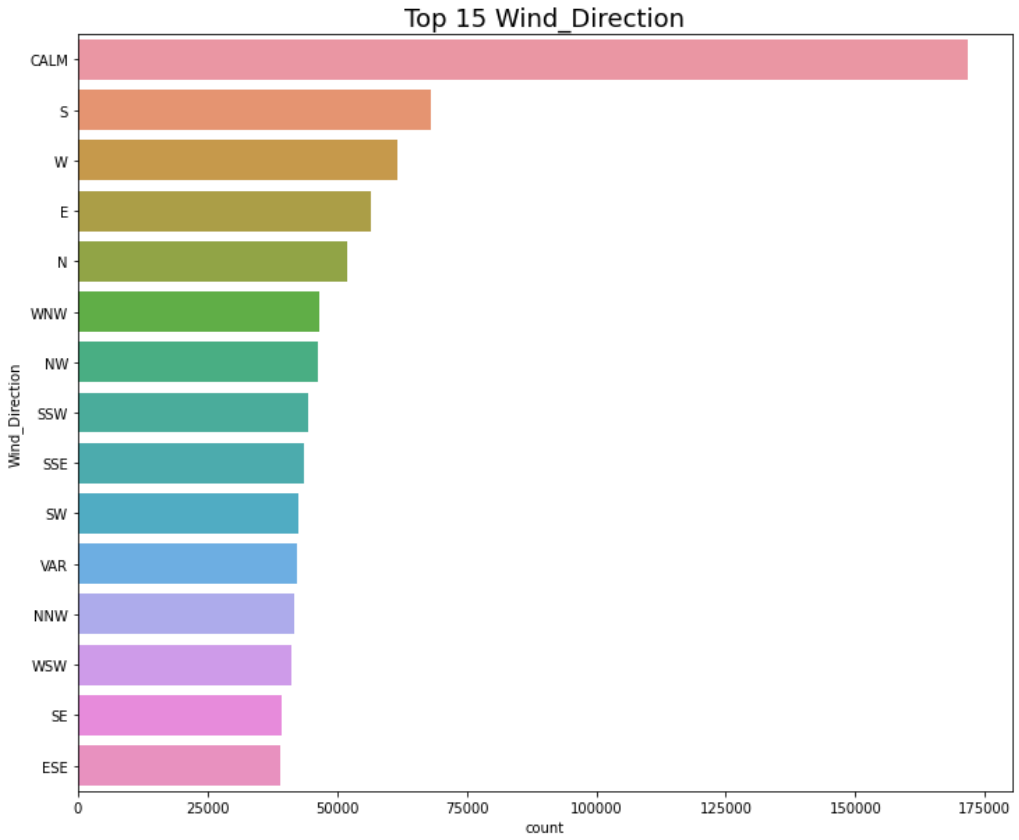
```
[51… <matplotlib.pyplot._IoffContext at 0x7fd1e26a7790>
```



- From the above bar plot, we can see that many accidents happening during the 13:00 - 17 Hrs

8. Wind Direction Analysis

```
fig = plt.figure(figsize = (12, 10))
sns.countplot(y='Wind_Direction', data=accidents, order=accidents['Wind_Direction'].value_counts()[
plt.show()
```

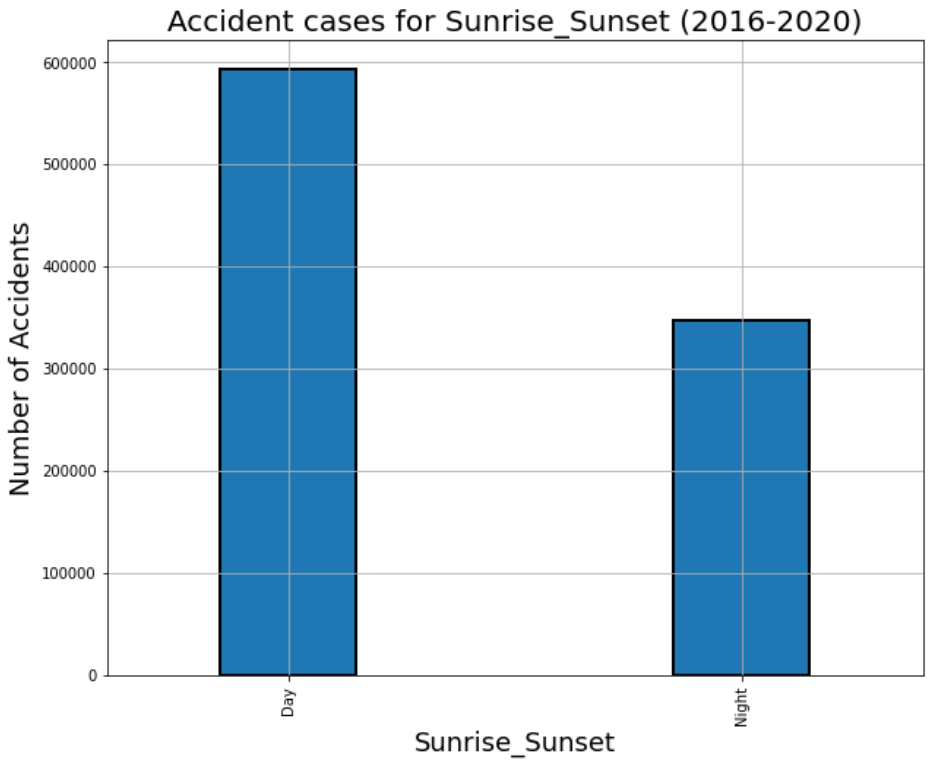


- From the above plot we can illustrate that maximum number of accidents with almost 175 cases are during calm condition and its seen maximum when the wind direction is South.
- The further cases are observed in wind direction West, East and North.
- This information is used in analysing how different weather condition trigger an accident.

9. Sunset Sunrise Analysis

[43]:
```
plt.figure(figsize=(10,8))
accidents.Sunrise_Sunset.value_counts().sort_values(ascending=False).head(5).plot.bar(widt
plt.xlabel('Sunrise_Sunset',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=30)
plt.title('Accident cases for Sunrise_Sunset (2016-2020)',fontsize=20)
plt.grid()
plt.ioff()
```

[43…
```
<matplotlib.pyplot._IoffContext at 0x7fda52bb3dd0>
```

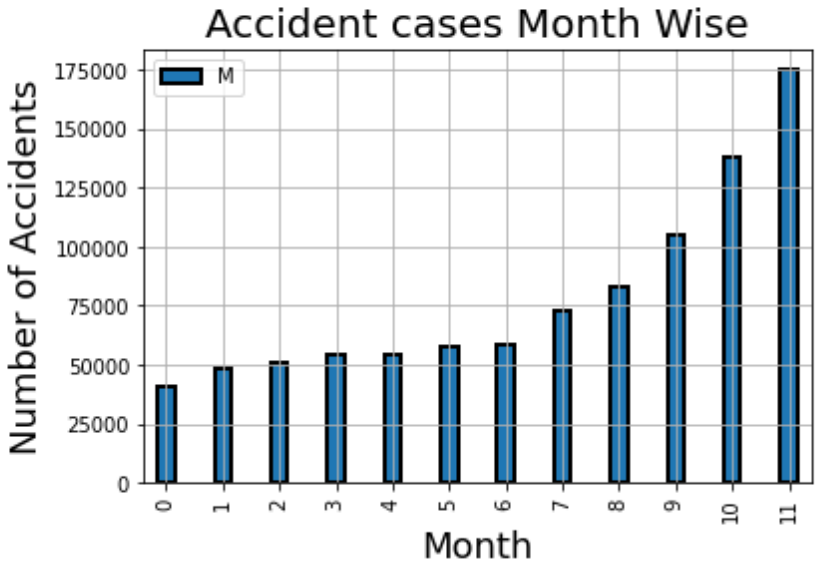


- From the above box plot, we can see that around 600k accidents are during the day time
- Around 350k cases are during the night time

10. Month vs Number of accidents

[88]:
```
#hour wise plot wrt accidents
plt.figure(figsize=(12,8))
accidents.M.value_counts().sort_values(ascending=True).head(24
plt.xlabel('Month',fontsize=18)
plt.ylabel('Number of Accidents',fontsize=18)
ax.tick_params(labelsize=30)
plt.title('Accident cases Month Wise',fontsize=20)
plt.grid()
plt.ioff()
```

[88…
```
<matplotlib.pyplot._IoffContext at 0x7fa6311f8190>
<Figure size 864x576 with 0 Axes>
```



- From the above bar plot we can see around 175k accidents with leading number of accidents in the month of December
- Following december, November and october has many accidents
- From the above we can infer that most of the accidents are at the year ending, which can lead to many weather conditions and road condition during the winter
- This information is used to regulate and analyse the data in a scalar level during different season and different months

References:

[1] C. O'Neill and R. Schutt. Doing Data Science., O'Reilly. 2013.
[2] NIST on EDA, https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm, last viewed February, 2021.
[4] John Tukey Biography, https://mathshistory.st-andrews.ac.uk/Biographies/Tukey/, last viewed 2021.
[5] J. Tukey,
http://www.ru.ac.bd/wp-content/uploads/sites/25/2019/03/102_05_01_Tukey-Explora
tory-Data-Analysis-1977.pdf. Last