Introduction: Business Problem

```
As an Auto Insurance company, it's critical to understand what could
be the factors that impact the severity of car accidents.  Finding
these factors would help the insurance company to price its products
properly.  In addition, based on this analysis, the insurance
company could also find risk control solutions to mitigate and
reduce the severity of car accidents.

In this project, we will study the car accident data and see the
severity level of car accidents has any relationship with various
weather patterns and factors.  We'll do regression analysis of
individual weather factors as well as multiple regression.  We'll
attempt to use Ridge Regression and Grid Search to improve the
models.
```

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

Dataset

The car accident dataset we use in this project covers 49 contiguous states of the United States of America for the period from Feb 2016 to June 2020. This dataset contains more than 3.5 million accident records with data captured by various entities such as the US and state departments of transportation, law enforcement agencies, etc. The dataset also contains weather conditions, day light conditions (like Sunrise, Twilight,etc.).

Credits/Acknowledgements:

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. “A Countrywide Traffic Accident Dataset.”, 2019.

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

In [2]:

```python
csv_path = '~/OneDrive/python_learning/Data/US_Accidents_June20.csv'
```

In [3]:
```python
car_accident = pd.read_csv(csv_path)
```

Data Cleanup

Here's the original table structure with 49 attributes: Here's the original table structure with 49 attributes:

#	Attribute	Description
1	ID	This is a unique identifier of the accident record.
2	Source	Indicates source of the accident report (i.e. the API which reported the accident.)
3	TMC	A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event.
4	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.)
5	Start_Time	Shows start time of the accident in local time zone.
6	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.
7	Start_Lat	Shows latitude in GPS coordinate of the start point.
8	Start_Lng	Shows longitude in GPS coordinate of the start point.
9	End_Lat	Shows latitude in GPS coordinate of the end point.
10	End_Lng	Shows longitude in GPS coordinate of the end point.
11	Distance(mi)	The length of the road extent affected by the accident.
12	Description	Shows natural language description of the accident.
13	Number	Shows the street number in address field.
14	Street	Shows the street name in address field.
15	Side	Shows the relative side of street (Right/Left) in address field.
16	City	Shows the city in address field.
17	County	Shows the county in address field.
18	State	Shows the State in address field.
19	Zipcode	Shows the zipcode in address field.
20	Country	Shows the country in address field.
21	Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).
22	Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.
23	Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).
24	Temperature(F)	Shows the temperature (in Fahrenheit).
25	Wind_Chill(F)	Shows the wind chill (in Fahrenheit).

26	Humidity(%)	Shows the humidity (in percentage).
27	Pressure(in)	Shows the air pressure (in inches).
28	Visibility(mi)	Shows visibility (in miles)
29	Wind_Direction	Shows wind direction
30	Wind_Speed(mph)	Shows wind speed (in miles per hour)
31	Precipitation(in)	Shows precipitation amount in inches, if there is any.
32	Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)
33	Amenity	A POI annotation which indicates presence of amenity in a nearby location.
34	Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.
35	Crossing	A POI annotation which indicates presence of presence of crossing in a nearby location.
36	Give_Way	A POI annotation which indicates presence of presence of give_way in a nearby location.
37	Junction	A POI annotation which indicates presence of presence of junction in a nearby location.
38	No_Exit	A POI annotation which indicates presence of presence of no exit in a nearby location.
39	Railway	A POI annotation which indicates presence of presence of railway in a nearby location.
40	Roundabout	A POI annotation which indicates presence of presence of roundabout in a nearby location.
41	Station	A POI annotation which indicates presence of presence of station in a nearby location.
42	Stop	A POI annotation which indicates presence of presence of stop in a nearby location.
43	Traffic_Calming	A POI annotation which indicates presence of presence of traffic calming in a nearby location.
44	Traffic_Signal	A POI annotation which indicates presence of presence of traffic signal in a nearby location.
45	Turning_Loop	A POI annotation which indicates presence of presence of turning loop in a nearby location.
46	Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.
47	Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.
48	Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.
49	Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.

In this project, we're focusing on the relationship, if any, between the severity of car accidents with weather conditions, POI elements and Period of Day. We'll need to clean up those data that are irrelevant, namely, Source, TMC, Start_Time, End_Time, Start_Lat, Start_Lng, End_Lat, End_Lng, Distance(m), Description, Number, Street, Side, City, County,, Zipcode, Country, Timezone, Airport_Code, Weather_Timestamp, Civil_Twilight, Nautical_Twilight, and Astronomical_Twilight.

Dropping unnecessary columns

In [4]:
```
car_accident.drop(columns=['Source','TMC','Start_Time','End_Time','Sta
                           'End_Lat','End_Lng', 'Distance(mi)','Descri
                           'County','Zipcode','Country','Timezone','Ai
                           'Civil_Twilight','Nautical_Twilight',
                           'Astronomical_Twilight'], inplace=True)
```

In [5]:
```
car_accident.columns
```

Out[5]:
```
Index(['ID', 'Severity', 'Side', 'State', 'Temperature(F)', 'Wind_Chi
ll(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Directi
on',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', '
Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railwa
y',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_S
ignal',
       'Turning_Loop', 'Sunrise_Sunset'],
      dtype='object')
```

In [6]:
```
#Finding missing data
missing_data = car_accident.isnull()
#count missing values in @ column
for column in missing_data.columns.values.tolist():
    print(column)
    print(missing_data[column].value_counts())
    print("")
```

```
False    1487859
Name: Precipitation(in), dtype: int64

Weather_Condition
False    3437597
True       76143
Name: Weather_Condition, dtype: int64

Amenity
False    3513740
Name: Amenity, dtype: int64

Bump
False    3513740
Name: Bump, dtype: int64

Crossing
False    3513740
Name: Crossing, dtype: int64
```

Our target variable, Severity has no missing value.

As we are interested in the weather factors that may affect Severity of an accident, we want to know if there are significant missing data. According to the above, we can see the following missing data:

Temperature(F): 65736 Wind_Chill(F): 1645484 Humidity(%): 69691 Pressure(in): 55884 Visibility(mi): 75861 Wind_Speed(mph): 454613 Precipitation(in): 1487859

We will replace these NaN by mean value

In [7]:
```
#Calculate the mean value for each of the above variables:
avg_temp = car_accident['Temperature(F)'].astype('float').mean(axis=0)
#Replace 'NaN' by mean value
car_accident['Temperature(F)'].replace(np.nan, avg_temp, inplace=True)
```

In [8]:
```
avg_windchill = car_accident['Wind_Chill(F)'].astype('float').mean(ax
car_accident['Wind_Chill(F)'].replace(np.nan, avg_windchill, inplace =
```

In [9]:
```
avg_hum = car_accident['Humidity(%)'].astype('float').mean(axis=0)
car_accident['Humidity(%)'].replace(np.nan, avg_hum, inplace=True)
```

In [10]:
```
avg_press = car_accident['Pressure(in)'].astype('float').mean(axis=0)
car_accident['Pressure(in)'].replace(np.nan, avg_press, inplace=True)
```

In [11]:
```
avg_vis =car_accident['Visibility(mi)'].astype('float').mean(axis=0)
car_accident['Visibility(mi)'].replace(np.nan, avg_vis, inplace=True)
```

In [12]:
```
avg_wspeed = car_accident['Wind_Speed(mph)'].astype('float').mean(axis
car_accident['Wind_Speed(mph)'].replace(np.nan, avg_wspeed, inplace=Tr
```

In [13]:
```
avg_prec =car_accident['Precipitation(in)'].astype('float').mean(axis=
car_accident['Precipitation(in)'].replace(np.nan, avg_prec, inplace=Tr
```

In [ ]:

In [ ]:

Overview

First, we want to take a look at the distribution of the car accidents by state and summarized them by Severity levels. By looking at the simple stacked chart, we can immediately spot that most of the 3.5 million records of car accidents were originated from the state of California.

Out of over 800,000 cases in California, majority of the cases are categorized as the 2nd and 3rd level of severity.

In [14]:
```
state_accident = car_accident.groupby(['State','Severity']).size().uns
state_accident.head()
```

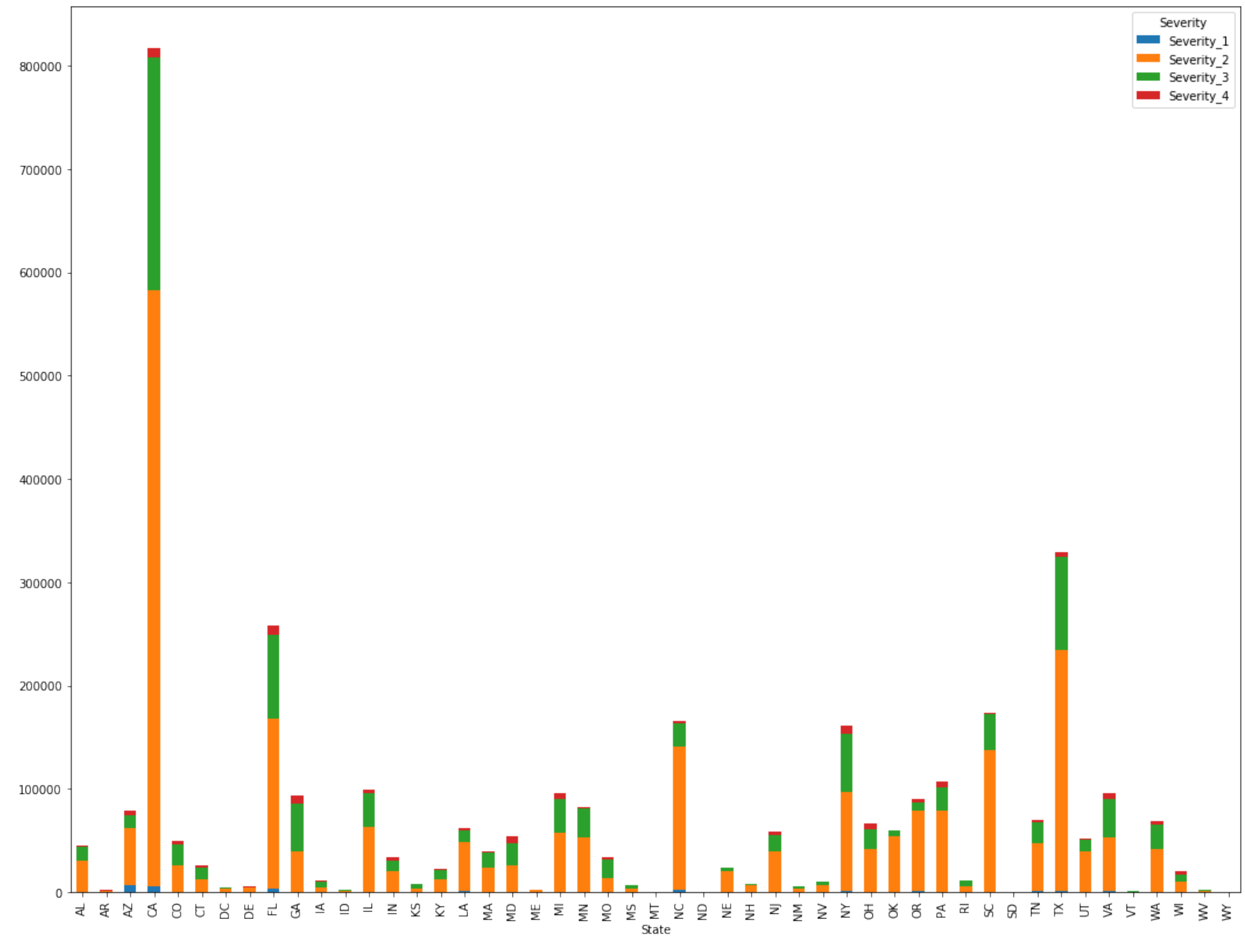
Out[14]:

Severity	Severity_1	Severity_2	Severity_3	Severity_4
State				
AL	133.0	30002.0	13890.0	600.0
AR	11.0	1011.0	488.0	502.0
AZ	6705.0	55091.0	13178.0	3612.0
CA	5801.0	576742.0	225820.0	8463.0
CO	519.0	25516.0	19888.0	3808.0

In [15]:
```
#plt.figure(figsize=(25,18))
state_accident.plot(kind='bar', stacked=True, figsize=(18,14))

#state_accident.plot(figsize=(18,14))
```

Out[15]: `<matplotlib.axes._subplots.AxesSubplot at 0x7ff21b550d30>`



In [16]:
```
!conda install -c conda-forge folium=0.5.0 --yes
import folium
print('Folium Installed')
```

```
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.

Folium Installed
```

In [17]:
```
us_geo =r'US_States.json'
us_map = folium.Map(location=[37, -102], zoom_start=5)
```

In [18]:
```
state_accident['Total']=state_accident.sum(axis=1) #adding a Total col

state_accident
```

```
state_accident
```

Out[18]:

Severity	Severity_1	Severity_2	Severity_3	Severity_4	Total
State					
AL	133.0	30002.0	13890.0	600.0	44625.0
AR	11.0	1011.0	488.0	502.0	2012.0
AZ	6705.0	55091.0	13178.0	3612.0	78586.0
CA	5801.0	576742.0	225820.0	8463.0	816826.0
CO	519.0	25516.0	19888.0	3808.0	49731.0
CT	22.0	12002.0	11632.0	2245.0	25901.0
DC	43.0	2991.0	1099.0	687.0	4820.0
DE	10.0	4288.0	629.0	812.0	5739.0
FL	3014.0	165506.0	80563.0	8919.0	258002.0
GA	406.0	38922.0	46837.0	7449.0	93614.0
IA	6.0	5088.0	5460.0	921.0	11475.0
ID	NaN	1598.0	207.0	243.0	2048.0
IL	265.0	63401.0	32652.0	3374.0	99692.0
IN	60.0	19928.0	10960.0	2804.0	33752.0
KS	5.0	3571.0	3947.0	416.0	7939.0
KY	40.0	11920.0	9745.0	848.0	22553.0
LA	1262.0	47099.0	11925.0	1229.0	61515.0
MA	183.0	23452.0	15019.0	390.0	39044.0
MD	305.0	26051.0	21359.0	5878.0	53593.0
ME	1.0	2065.0	75.0	102.0	2243.0
MI	57.0	57060.0	33542.0	5324.0	95983.0
MN	41.0	53538.0	27817.0	469.0	81865.0
MO	70.0	13868.0	18014.0	1691.0	33643.0
MS	5.0	3639.0	2636.0	305.0	6585.0
MT	NaN	264.0	141.0	107.0	512.0
NC	1807.0	139052.0	22047.0	3057.0	165963.0
ND	NaN	21.0	12.0	11.0	44.0
NE	38.0	20009.0	3637.0	287.0	23971.0
NH	4.0	6352.0	1444.0	184.0	7984.0
NJ	93.0	39160.0	16040.0	3766.0	59059.0

NM	48.0	3020.0	2093.0	362.0	5523.0
NV	3.0	7064.0	3202.0	455.0	10724.0
NY	727.0	96064.0	56889.0	7137.0	160817.0
OH	526.0	41120.0	18904.0	5590.0	66140.0
OK	72.0	53599.0	5892.0	440.0	60003.0
OR	1263.0	77747.0	8073.0	3051.0	90134.0
PA	219.0	78724.0	22124.0	5727.0	106794.0
RI	71.0	5567.0	6000.0	115.0	11753.0
SC	116.0	137371.0	34620.0	1170.0	173277.0
SD	NaN	17.0	8.0	36.0	61.0
TN	1453.0	45942.0	20808.0	1692.0	69895.0
TX	1070.0	233840.0	89667.0	4707.0	329284.0
UT	395.0	38562.0	11379.0	1349.0	51685.0
VA	1739.0	51639.0	37187.0	5510.0	96075.0
VT	1.0	486.0	146.0	69.0	702.0
WA	531.0	41732.0	23234.0	3048.0	68545.0
WI	33.0	10077.0	7302.0	2708.0	20120.0
WV	2.0	1388.0	504.0	487.0	2381.0
WY	NaN	133.0	178.0	197.0	508.0

In [19]:

```
# Adding the Full State Name column to match up with the JSON File
state_names=['Alabama','Arkansas','Arizona','California','Colorado','C
             'Delaware','Florida','Georgia','Iowa','Idaho','Illinois',
             'Kentucky','Louisiana','Massachusetts','Maryland','Maine'
             'Mississippi','Montana','North Carolina','North Dakota','
             'New Jersey','New Mexico','Nevada','New York','Ohio', 'Ok
             'Pennsylvania', 'Rhode Island', 'South Carolina', 'South
             'Texas', 'Utah','Virginia','Vermont','Washington','Wiscon

state_accident['StateName']=state_names
state_accident
```

Out[19]:

Severity	Severity_1	Severity_2	Severity_3	Severity_4	Total	StateName
State						
AL	133.0	30002.0	13890.0	600.0	44625.0	Alabama
AR	11.0	1011.0	488.0	502.0	2012.0	Arkansas
AZ	6705.0	55091.0	13178.0	3612.0	78586.0	Arizona

CA	5801.0	576742.0	225820.0	8463.0	816826.0	California
CO	519.0	25516.0	19888.0	3808.0	49731.0	Colorado
CT	22.0	12002.0	11632.0	2245.0	25901.0	Connecticut
DC	43.0	2991.0	1099.0	687.0	4820.0	District of Columbia
DE	10.0	4288.0	629.0	812.0	5739.0	Delaware
FL	3014.0	165506.0	80563.0	8919.0	258002.0	Florida
GA	406.0	38922.0	46837.0	7449.0	93614.0	Georgia
IA	6.0	5088.0	5460.0	921.0	11475.0	Iowa
ID	NaN	1598.0	207.0	243.0	2048.0	Idaho
IL	265.0	63401.0	32652.0	3374.0	99692.0	Illinois
IN	60.0	19928.0	10960.0	2804.0	33752.0	Indiana
KS	5.0	3571.0	3947.0	416.0	7939.0	Kansas
KY	40.0	11920.0	9745.0	848.0	22553.0	Kentucky
LA	1262.0	47099.0	11925.0	1229.0	61515.0	Louisiana
MA	183.0	23452.0	15019.0	390.0	39044.0	Massachusetts
MD	305.0	26051.0	21359.0	5878.0	53593.0	Maryland
ME	1.0	2065.0	75.0	102.0	2243.0	Maine
MI	57.0	57060.0	33542.0	5324.0	95983.0	Michigan
MN	41.0	53538.0	27817.0	469.0	81865.0	Minnesota
MO	70.0	13868.0	18014.0	1691.0	33643.0	Missouri
MS	5.0	3639.0	2636.0	305.0	6585.0	Mississippi
MT	NaN	264.0	141.0	107.0	512.0	Montana
NC	1807.0	139052.0	22047.0	3057.0	165963.0	North Carolina
ND	NaN	21.0	12.0	11.0	44.0	North Dakota
NE	38.0	20009.0	3637.0	287.0	23971.0	Nebraska
NH	4.0	6352.0	1444.0	184.0	7984.0	New Hampshire
NJ	93.0	39160.0	16040.0	3766.0	59059.0	New Jersey
NM	48.0	3020.0	2093.0	362.0	5523.0	New Mexico
NV	3.0	7064.0	3202.0	455.0	10724.0	Nevada
NY	727.0	96064.0	56889.0	7137.0	160817.0	New York
OH	526.0	41120.0	18904.0	5590.0	66140.0	Ohio
OK	72.0	53599.0	5892.0	440.0	60003.0	Oklahoma
OR	1263.0	77747.0	8073.0	3051.0	90134.0	Oregon
PA	219.0	78724.0	22124.0	5727.0	106794.0	Pennsylvania

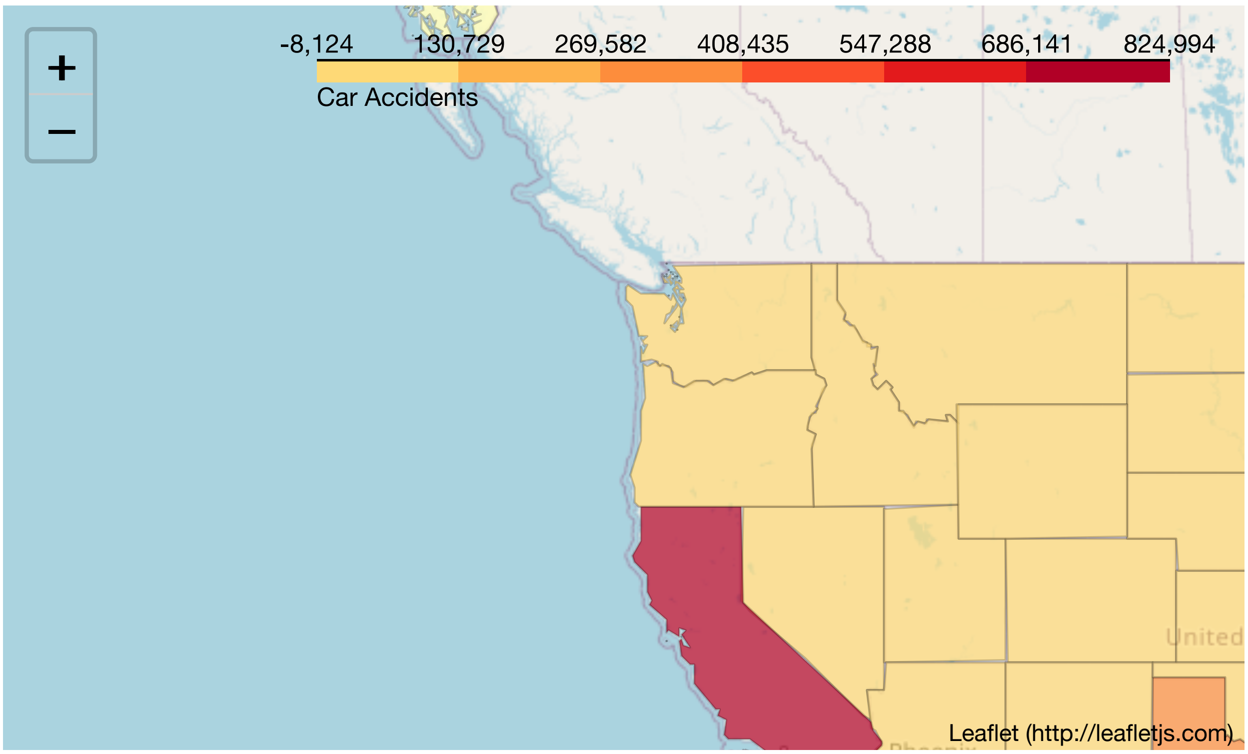
RI	71.0	5567.0	6000.0	115.0	11753.0	Rhode Island
SC	116.0	137371.0	34620.0	1170.0	173277.0	South Carolina
SD	NaN	17.0	8.0	36.0	61.0	South Dakota
TN	1453.0	45942.0	20808.0	1692.0	69895.0	Tennessee
TX	1070.0	233840.0	89667.0	4707.0	329284.0	Texas
UT	395.0	38562.0	11379.0	1349.0	51685.0	Utah
VA	1739.0	51639.0	37187.0	5510.0	96075.0	Virginia
VT	1.0	486.0	146.0	69.0	702.0	Vermont
WA	531.0	41732.0	23234.0	3048.0	68545.0	Washington
WI	33.0	10077.0	7302.0	2708.0	20120.0	Wisconsin
WV	2.0	1388.0	504.0	487.0	2381.0	West Virginia
WY	NaN	133.0	178.0	197.0	508.0	Wyoming

Mapping Accidents by State

In [86]:
```python
us_map = folium.Map(location=[37.09, -95.71], zoom_start=4, tiles='ope
us_map.choropleth(
    geo_data=us_geo,
    data=state_accident,
    columns=['StateName','Total'],
    key_on='feature.properties.NAME',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name= 'Car Accidents')

us_map
```

Out[86]:



Focusing on California Data

Having 818K over 3.5 million data points, California accident records are good representation of the US accidents data. By limiting the size of the dataset, it will improve the performance of the analysis without jeopardizing the quality of the analysis.

We first review the correlation between Wind Speed and Severity. Then, we look at the Linear Regression of multiple factors of Weather Conditions and Severity.

In [21]:
```python
Calif_accident = car_accident.loc[car_accident['State']=='CA']
Calif_accident.shape
```

Out[21]:
```
(816826, 27)
```

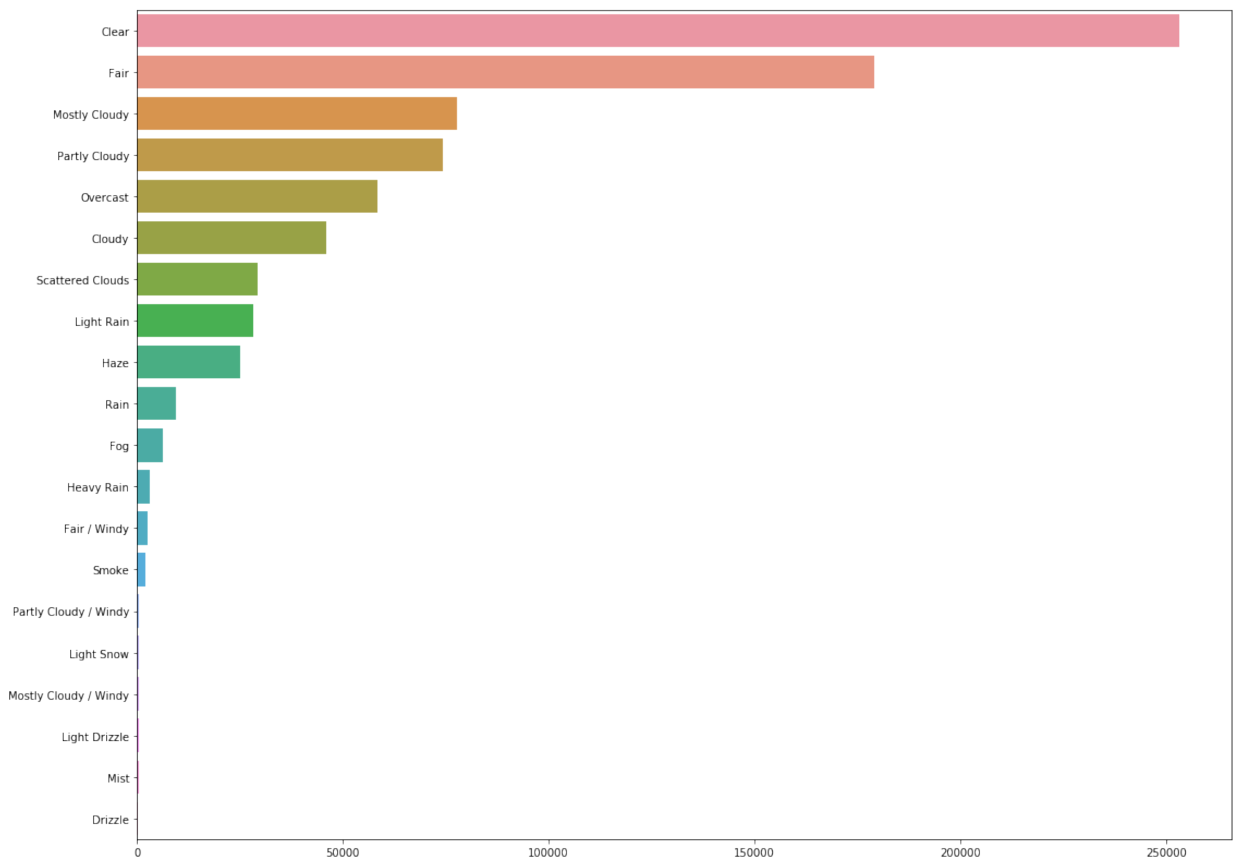
```
## Weather Condition <a name='Weather Condition'></a>

While one would guess that most of the accidents happen in a poor
weather condition, we find that majority of the accidents occurred
in clear weather condition and very small number of cases occurred
in snowing or even foggy conditions.
```

In [24]:
```
weather_pattern = Calif_accident.Weather_Condition.value_counts().heac

plt.figure(figsize =(18,14))
sns.barplot(weather_pattern.values, weather_pattern.index)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff26f67dfd0>



Wind Speed vs Severity

We want to review the relationship between the Severity level and wind speed. Applying Linear Regression analysis, we can see a very low score that is quite conclusive to say that there's no correlation between the Wind Speed and Severity.

In [22]:
```
wind_accident=Calif_accident[['Severity','Wind_Speed(mph)']].copy()
wind_accident.fillna(0,inplace=True)
wind_accident.head()
```

Out[22]:

	Severity	Wind_Speed(mph)
728	3	5.8
729	3	4.6
730	2	4.6
731	3	4.6
732	2	5.8

In [23]:
```
from sklearn.linear_model import LinearRegression
X = wind_accident[['Wind_Speed(mph)']]
Y = wind_accident[['Severity']]
lm = LinearRegression()
lm.fit(X,Y)
lm.score(X,Y)
```

Out[23]: 0.0017048815614110202

In [89]:
```
from scipy import stats
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Wind_Speed(m
```

Out[89]: (0.04129021144787945, 4.731173444384992e-305)

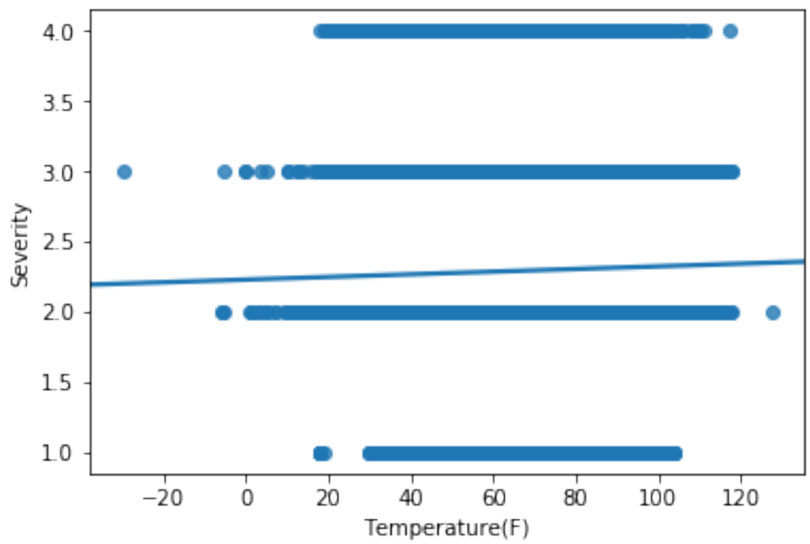
Individual Weather Factors vs Severity

```
## Temperature

From the correlation, we don't see a big correlation between
Temperature and Severity. The Perason Correlation Coefficient is:
.024 and p-value is: 5.990 e^-104.
```

In [25]:
```
#Looking at the correlation, if any, between individual weather factor
#severity of accidents
sns.regplot(x='Temperature(F)',y='Severity',data=Calif_accident)
plt.ylim()
```

Out[25]: (0.8461768805618731, 4.153823119438128)



In [91]:
```
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Temperature(
```

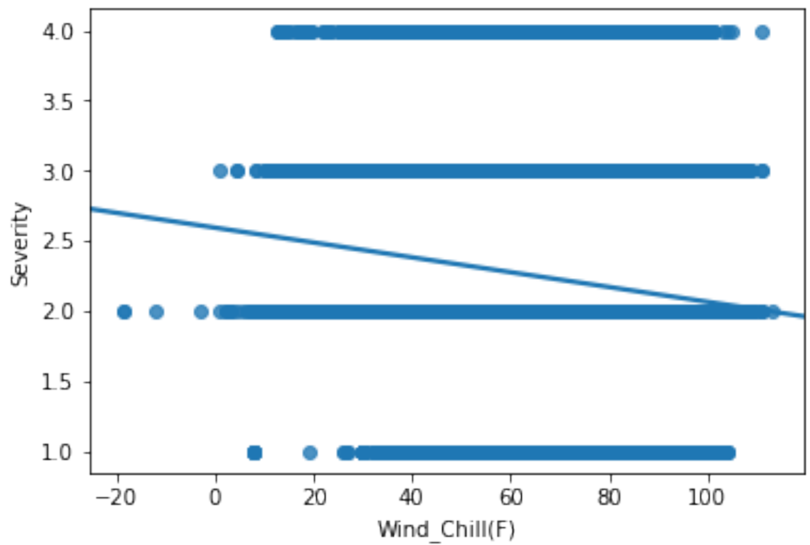
Out[91]: (0.02395218314186769, 5.989153121752934e-104)

Wind Chill

From the correlation, we don't see any correlation between Wind Chill and Severity. The Perason Correlation Coefficient is: -.108 and p-value is: 0.000.

In [26]:
```
sns.regplot(x='Wind_Chill(F)',y='Severity',data=Calif_accident)
plt.ylim()
```

Out[26]: (0.8461768805618731, 4.153823119438128)



In [93]:
```
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Wind_Chill(F
```

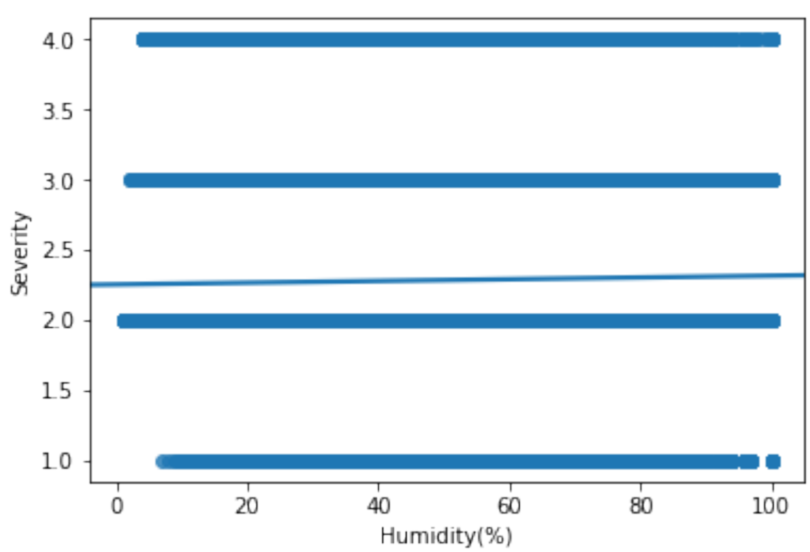
Out[93]: (-0.10849908577891573, 0.0)

Humidity

Humidity is another low correlation with Severity.The Perason Correlation Coefficient is: .030 and p-value is: 4.284 e^-157.

In [27]:
```
sns.regplot(x='Humidity(%)',y='Severity',data=Calif_accident)
plt.ylim()
```

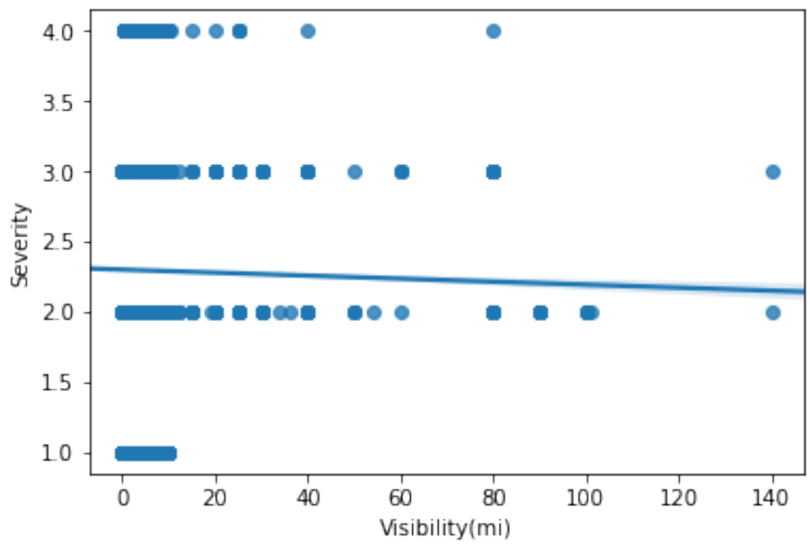
Out[27]: (0.8461768805618731, 4.153823119438128)



In [95]:
```
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Humidity(%)'
```
Out[95]:
```
(0.029539938798180492, 4.2842244391312885e-157)
```

Visibility
Visibility and Severity have very little correlation as well. The Perason Correlation Coefficient is: -0.005 and p-value is: 4.008 e^-06.

In [28]:
```
sns.regplot(x='Visibility(mi)',y='Severity',data=Calif_accident)
plt.ylim()
```
Out[28]:
```
(0.8461768805618731, 4.153823119438128)
```



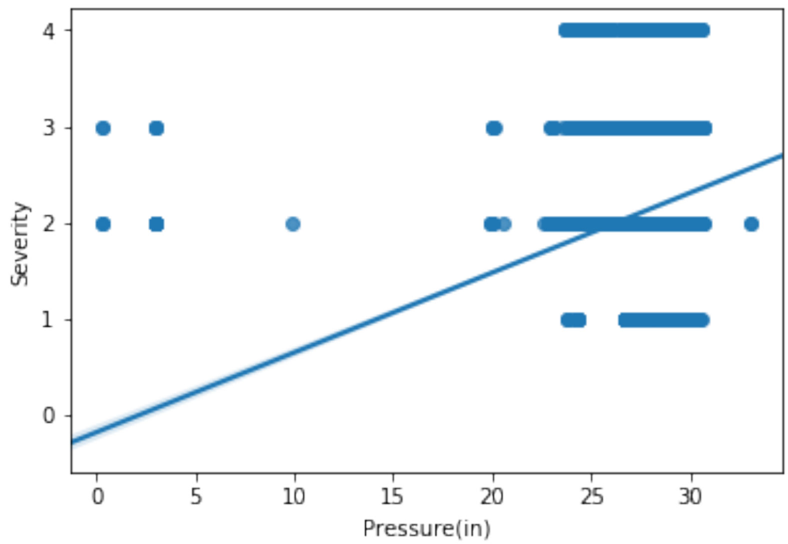
In [96]:
```
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Visibility(m
```
Out[96]:
```
(-0.005101826963224123, 4.007868861995451 5e-06)
```

Pressure

Air Pressure and Severity have very little correlation as well. The Perason Correlation Coefficient is: .095 and p-value is: 0.0.

In [29]:
```
sns.regplot(x='Pressure(in)',y='Severity',data=Calif_accident)
plt.ylim()
```

Out[29]:
```
(-0.5963764625461522, 4.222516135776606)
```



In [97]:
```
stats.pearsonr(Calif_accident['Severity'],Calif_accident['Pressure(in)
```

Out[97]:
```
(0.09525123614477869, 0.0)
```

Multiple Regression

We want to look further at the relationships between Severity of an accident (Target) and weather factors, namely, Temperature(F), Wind_Chill(F), Humidity(%), Pressure(in), Visibility(mi),Precipitation(in), and Wind_Speed(mph).

We develop a model using these variables as the predictor variables.

First: Training Data

In [30]:
```
y_data = Calif_accident['Severity']
x_data = Calif_accident[['Temperature(F)','Wind_Chill(F)','Humidity(%)
                'Wind_Speed(mph)']]
```

In [31]:
```
from sklearn.model_selection import train_test_split


x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, te


print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 163366
number of training samples: 653460
```

In [32]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train[['Temperature(F)','Wind_Chill(F)','Humidity(%)','Pressu
                'Visibility(mi)','Precipitation(in)',
                'Wind_Speed(mph)']], y_train)
```

Out[32]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [33]:
```
yhat_train = lr.predict(x_train[['Temperature(F)','Wind_Chill(F)',
                                 'Humidity(%)','Pressure(in)','Visibil
                                 'Precipitation(in)','Wind_Speed(mph)'
yhat_train[0:5]
```

Out[33]:
```
array([2.11865956, 2.27341099, 2.33734683, 2.17071951, 2.35067175])
```

In [34]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

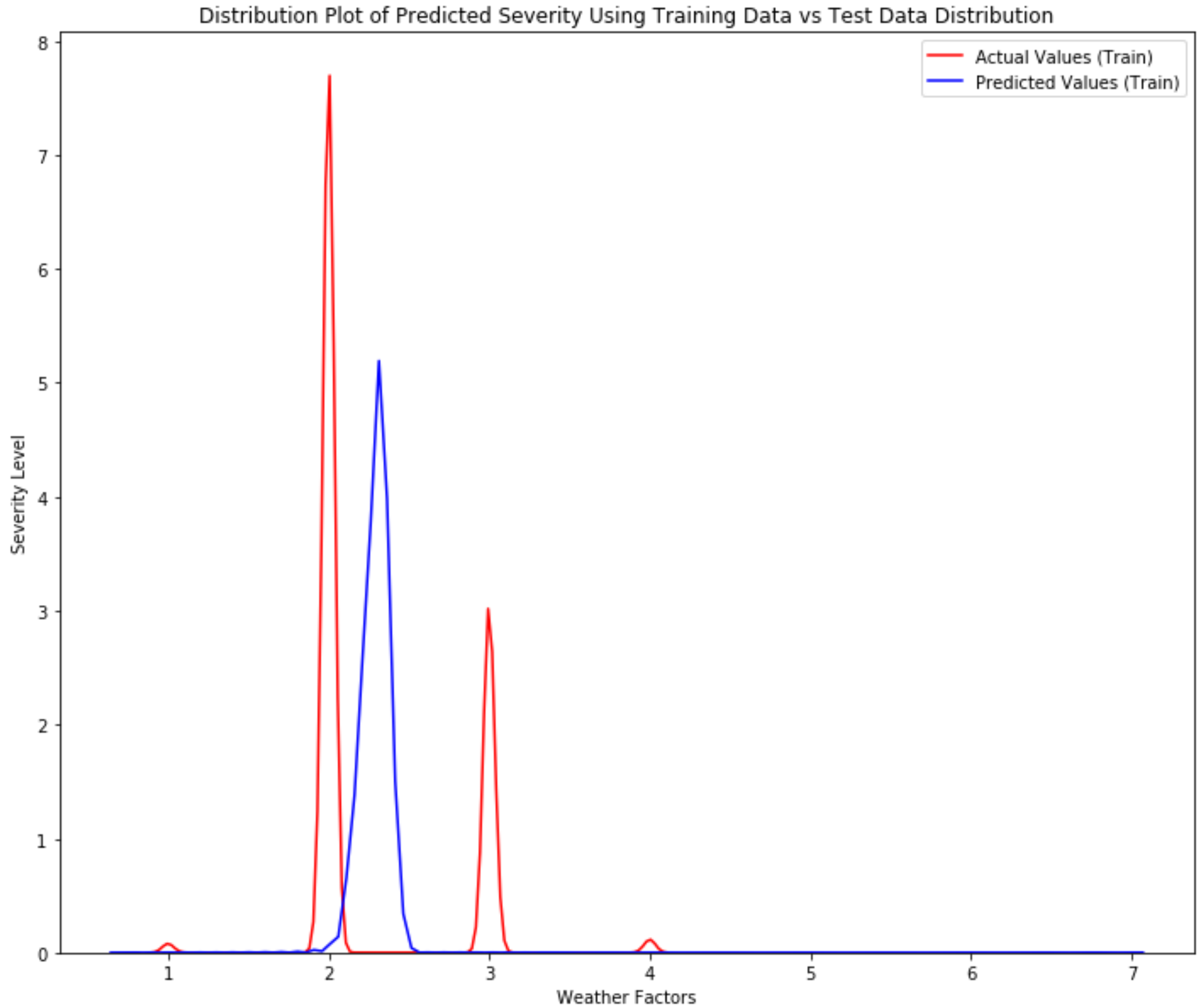
In [35]:
```
def DistributionPlot(RedFunction, BlueFunction, RedName, BlueName, Tit
    width = 12
    height = 10
    plt.figure(figsize=(width, height))

    ax1 = sns.distplot(RedFunction, hist=False, color="r", label=RedNa
    ax2 = sns.distplot(BlueFunction, hist=False, color="b", label=Blue

    plt.title(Title)
    plt.xlabel('Weather Factors')
    plt.ylabel('Severity Level')

    plt.show()
    plt.close()
```

In [36]:
```
Title = 'Distribution Plot of Predicted Severity Using Training Data v
DistributionPlot(y_train, yhat_train,
                 "Actual Values (Train)", "Predicted Values (Train)",
```



Ridge Model

In [37]:
```
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
pr=PolynomialFeatures(degree=2)
x_train_pr=pr.fit_transform(x_train[['Temperature(F)','Wind_Chill(F)',
                'Visibility(mi)','Precipitation(in)',
                'Wind_Speed(mph)']])
x_test_pr=pr.fit_transform(x_test[['Temperature(F)','Wind_Chill(F)','H
                'Visibility(mi)','Precipitation(in)',
                'Wind_Speed(mph)']])
RidgeModel = Ridge(alpha=0.1)
RidgeModel.fit(x_train_pr,y_train)
```

Out[37]:
```
Ridge(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
```

In [38]:
```
yhat = RidgeModel.predict(x_test_pr)
print('predicted:', yhat[0:4])
print('test set :', y_test[0:4].values)
```

```
predicted: [2.15239507 2.47268982 2.34856686 2.23223706]
test set : [2 2 3 2]
```

In [ ]:

In [ ]:

Grid Search

In [39]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
parameters1= [{'alpha': [0.001, 0.01, 0.1,1, 10, 100, 1000, 10000, 10(
RR=Ridge()
RR
```

Out[39]:
```
Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
```

In [40]:
```
Grid1=GridSearchCV(RR,parameters1,cv=6)
Grid1.fit(x_data[['Temperature(F)','Wind_Chill(F)','Humidity(%)','Pres
                'Visibility(mi)','Precipitation(in)',
                'Wind_Speed(mph)']],y_data)
```

Out[40]:
```
GridSearchCV(cv=6, error_score='raise-deprecating',
       estimator=Ridge(alpha=1.0, copy_X=True, fit_intercept=True, ma
x_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid=[{'alpha': [0.001, 0.01, 0.1, 1, 10, 100, 1000, 100
00, 100000]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn'
,
       scoring=None, verbose=0)
```

In [41]:
```
BestRR=Grid1.best_estimator_
BestRR
```

Out[41]:
```
Ridge(alpha=1000, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
```

In [42]:
```
BestRR.score(x_test[['Temperature(F)','Wind_Chill(F)','Humidity(%)','F
                'Visibility(mi)','Precipitation(in)',
                'Wind_Speed(mph)']], y_test)
```

Out[42]:
```
0.030479696225644393
```

Conclusion:

We've run the correlation tests on individual weather factors and the multiple regression test on the combination of all weather factors and found that their correlations are not good. We also use Ridge Regression and Grid Search to improve the model and still don't see a good result.

We can conclusively say that weather factors do not give any prediction of Severity of a car accident.