Basic Anal}O'is- US Accidents I Kaggle

menu [**kaggle**](https://www.kaggle.com/) search

•Search

search

Search

[explore](https://www.kaggle.com/)

[[II](https://www.kaggle.com/competitions)

[table chart](https://www.kaggle.com/datasets) [code](https://www.kaggle.com/notebooks)



[\I:·;.:·:t·-.:-;:·+.·:;](https://www.kaggle.com/notebooks)

[comment](https://www.kaggle.com/discussion) [school](https://www.kaggle.com/learn)



more vert

-...;:·;r:·:

menu [**kaggle**](https://www.kaggle.com/) search

Search

[explore](https://www.kaggle.com/)

[[II](https://www.kaggle.com/competitions)

[table chart](https://www.kaggle.com/datasets) [code](https://www.kaggle.com/notebooks)



[\I:·;.:·:t·-.:-;:·+.·:;](https://www.kaggle.com/notebooks)

[comment](https://www.kaggle.com/discussion) [school](https://www.kaggle.com/learn)



more\_vert

More

arrow\_back

searclose history

accide

Trending Searches trending\_up clustering

trending\_up cs985 trending\_up

machine learning

Popular Tags

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| starter code | beginner | data visualization | deep learning | business | eda | classification |
| 40245 | 15446 | 9522 | 7513 | 6077 | 5730 | 5172 |

tutorial

4879

utility script

4709

computing

3817

natural and physical sciences

3262

arts and entertainment

3174

feature engineering

3053

society

1447

data cleaning

2940

computer science

2611

statistics

2524

internet

2015

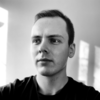
reference

1926

education

1695

Basic Analysis - US Accidents



Python notebook using data from [US Accidents (3.0 million records)](https://www.kaggle.com/sobhanmoosavi/us-accidents) · 3,703 views · 2mo ago



40

[ Copy and Edit](https://www.kaggle.com/kernels/fork/7000844)

41



Version 16

 16 commits

[Notebook](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/notebook)

[Data](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/data) [Log](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/log) [Comments](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/comments)

Notebook

In [1]:

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load in*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import matplotlib.pyplot as plt import seaborn as sns

*# Input data files are available in the "../input/" directory.*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files un der the input directory*

import os

for dirname, \_, filenames **in**  os.walk('/kaggle/input'):

for filename **in**  filenames:

print(os.path.join(dirname, filename))

*# Any results you write to the current directory are saved as output.*

/kaggle/input/us-accidents/US\_Accidents\_May19.csv

Welcome to my quick analysis of the US Accidents data!

First, let's read the data using Pandas. Then we can check what exactly we're to be working with.

In [2]:

data = pd.read\_csv(r'/kaggle/input/us-accidents/US\_Accidents\_May19.csv')

In [3]:

data.describe()

Out[3]:

m

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TMC | Severity | Start\_Lat | Start\_Lng | End\_Lat | End\_Lng | Distance(mi) | Nu |
| count | 1.727177e+06 | 2.243939e+06 | 2.243939e+06 | 2.243939e+06 | 516762.000000 | 516762.000000 | 2.243939e+06 | 78 |
| mean | 2.073527e+02 | 2.382692e+00 | 3.646348e+01 | -  9.485567e+01 | 37.443109 | -96.527543 | 2.879095e-01 | 56 |
| std | 1.940527e+01 | 5.488029e-01 | 4.958759e+00 | 1.709453e+01 | 5.126585 | 17.986406 | 1.532341e+00 | 11 |
| min | 2.000000e+02 | 0.000000e+00 | 2.457022e+01 | -  1.246238e+02 | 24.570110 | -124.497829 | 0.000000e+00 | 1.0 |
| 25% | 2.010000e+02 | 2.000000e+00 | 3.348468e+01 | -  1.171362e+02 | 33.887450 | -117.870577 | 0.000000e+00 | 80 |
| 50% | 2.010000e+02 | 2.000000e+00 | 3.586428e+01 | -  8.818469e+01 | 38.038480 | -90.192310 | 0.000000e+00 | 26 |

5

2

0

3

7

3.000000e+00 4.042111e+01

- 41.393320 -80.895040 1.000000e-02 684

8.085453e+01

1

In [4]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 75% | 2.010000e+02 |  |  |  |  |  |  |  |
| max | 4.060000e+02 | 4.000000e+00 | 4.900076e+01 | -  6.711317e+01 | 49.075000 | -67.109242 | 3.336300e+02 | 96 |
|  | | | | | | | | |

data[:10]

Out[4]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ID | Source | TMC | Severity | Start\_Time | End\_Time | Start\_Lat | Start\_Lng | End\_Lat | End\_Lng | ... | Roundabo |
| 0 | A-  1 | MapQuest | 201.0 | 3 | 2016-02-  08  05:46:00 | 2016-02-  08  11:00:00 | 39.865147 | -  84.058723 | NaN | NaN | ... | False |
| 1 | A-  2 | MapQuest | 201.0 | 2 | 2016-02-  08  06:07:59 | 2016-02-  08  06:37:59 | 39.928059 | -  82.831184 | NaN | NaN | ... | False |
| 2 | A-  3 | MapQuest | 201.0 | 2 | 2016-02-  08  06:49:27 | 2016-02-  08  07:19:27 | 39.063148 | -  84.032608 | NaN | NaN | ... | False |
| 3 | A-  4 | MapQuest | 201.0 | 3 | 2016-02-  08  07:23:34 | 2016-02-  08  07:53:34 | 39.747753 | -  84.205582 | NaN | NaN | ... | False |
| 4 | A-  5 | MapQuest | 201.0 | 2 | 2016-02-  08  07:39:07 | 2016-02-  08  08:09:07 | 39.627781 | -  84.188354 | NaN | NaN | ... | False |
| 5 | A-  6 | MapQuest | 201.0 | 3 | 2016-02-  08  07:44:26 | 2016-02-  08  08:14:26 | 40.100590 | -  82.925194 | NaN | NaN | ... | False |
| 6 | A-  7 | MapQuest | 201.0 | 2 | 2016-02-  08  07:59:35 | 2016-02-  08  08:29:35 | 39.758274 | -  84.230507 | NaN | NaN | ... | False |
| 7 | A-  8 | MapQuest | 201.0 | 3 | 2016-02-  08  07:59:58 | 2016-02-  08  08:29:58 | 39.770382 | -  84.194901 | NaN | NaN | ... | False |
| 8 | A-  9 | MapQuest | 201.0 | 2 | 2016-02-  08  08:00:40 | 2016-02-  08  08:30:40 | 39.778061 | -  84.172005 | NaN | NaN | ... | False |
| 9 | A-  10 | MapQuest | 201.0 | 3 | 2016-02-  08  08:10:04 | 2016-02-  08  08:40:04 | 40.100590 | -  82.925194 | NaN | NaN | ... | False |
|  | | | | | | | | | | | | |

10 rows × 49 columns

In [5]:

data.columns

Out[5]:

Index(['ID', 'Source', 'TMC', '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'], dtype='object')

Now we can start looking for relations between the data. For example, let's take a look at the amount of accidents per state.

To do that (I think) we have to prepare our data, which we can do with a simple **for** loop.

In [6]:

states = data.State.unique()

In [7]:

count\_by\_state=[]

for i **in**  data.State.unique():

count\_by\_state.append(data[data['State']==i].count()['ID'])

In the loop above we also used the **count** function to get the amount.

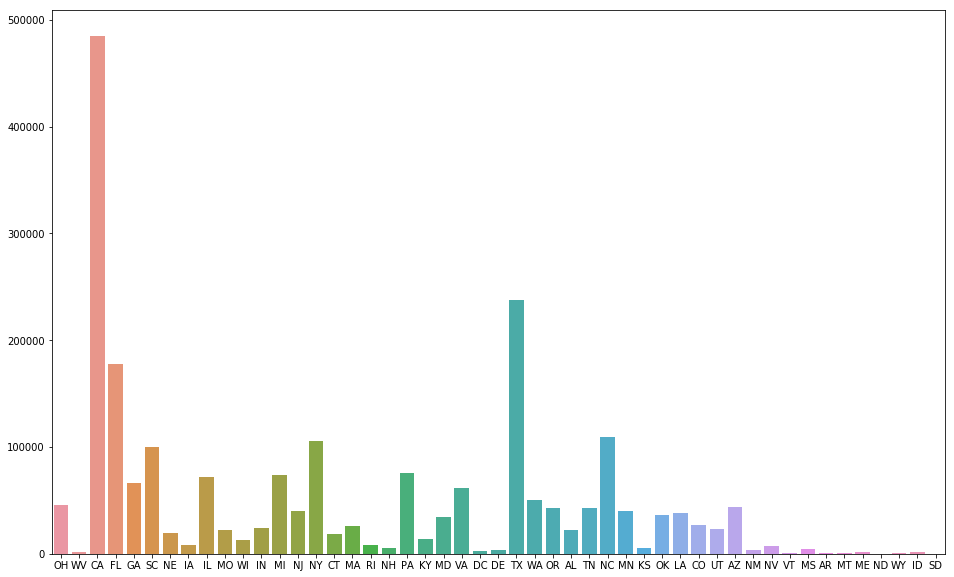
In [8]:

plt.figure(figsize=(16,10))

sns.barplot(states, count\_by\_state)

Out[8]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe7810210b8>



We can clearly see that California had had the biggest amount of accidents.

Next, let's check if severity affects the way the amounts align.

In [9]:

severity\_1\_by\_state = [] severity\_2\_by\_state = [] severity\_3\_by\_state = [] severity\_4\_by\_state = [] for i **in**  states:

severity\_1\_by\_state.append(data[(data['Severity']==1)&(data['State']==i)].cou nt()['ID'])

severity\_2\_by\_state.append(data[(data['Severity']==2)&(data['State']==i)].cou nt()['ID'])

severity\_3\_by\_state.append(data[(data['Severity']==3)&(data['State']==i)].cou nt()['ID'])

severity\_4\_by\_state.append(data[(data['Severity']==4)&(data['State']==i)].cou nt()['ID'])

The graph below shows us how different severity levels compare.

In [10]:

plt.figure(figsize=(20,15))

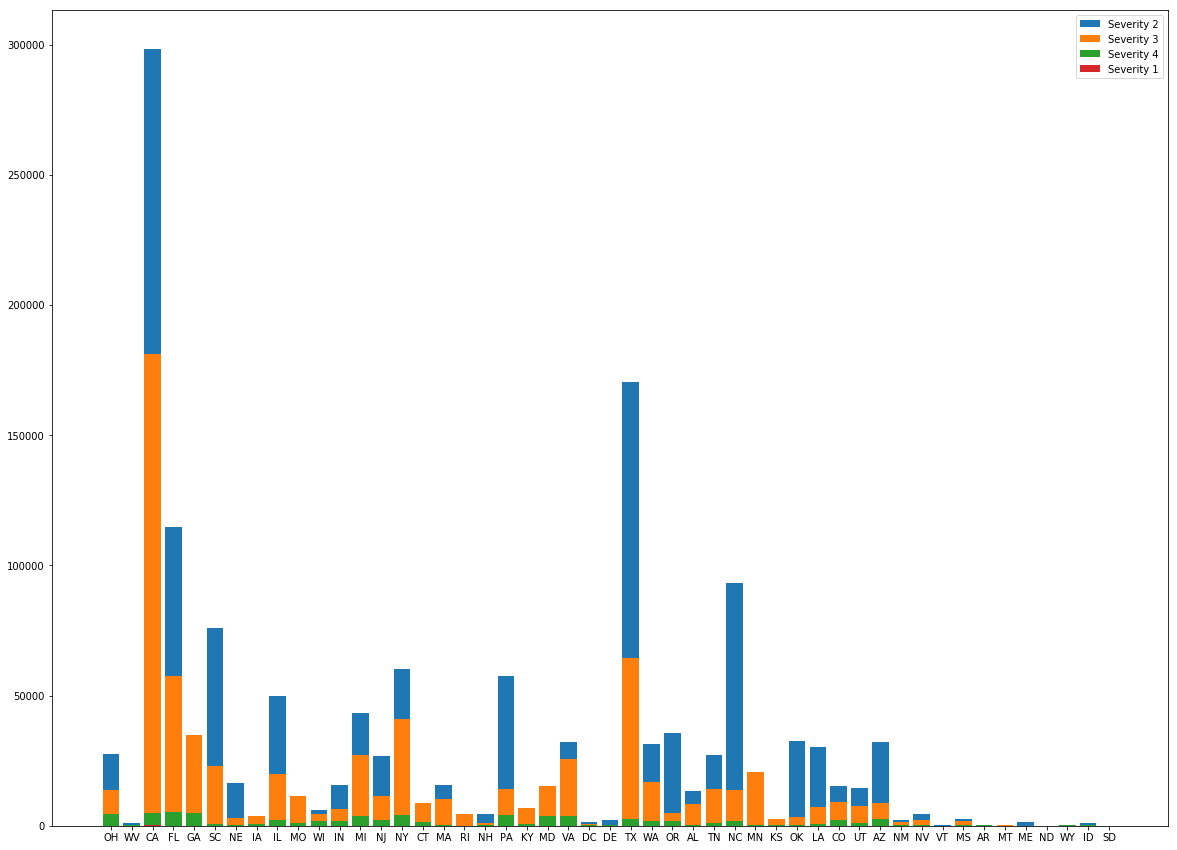
plt.bar(states, severity\_2\_by\_state, label='Severity 2') plt.bar(states, severity\_3\_by\_state, label='Severity 3') plt.bar(states, severity\_4\_by\_state, label='Severity 4')

plt.bar(states, severity\_1\_by\_state, label='Severity 1')

plt.legend()

Out[10]:

<matplotlib.legend.Legend at 0x7fe7a83ec5c0>



Although we plotted four different features, in most cases only three of them appear in our graph.

This shows us, that accidents of severity 1 barely ever occur. Or are barely ever reported.

In [11]:

data.TMC.unique()

Out[11]:

array([201., 241., 247., 246., 341., 406., 245., 248., 200., 244., 203.,

229., 222., 202., 206., 343., 236., 239., 336., 339., 351., nan])

In [12]:

TMC\_counts=data.TMC.value\_counts()

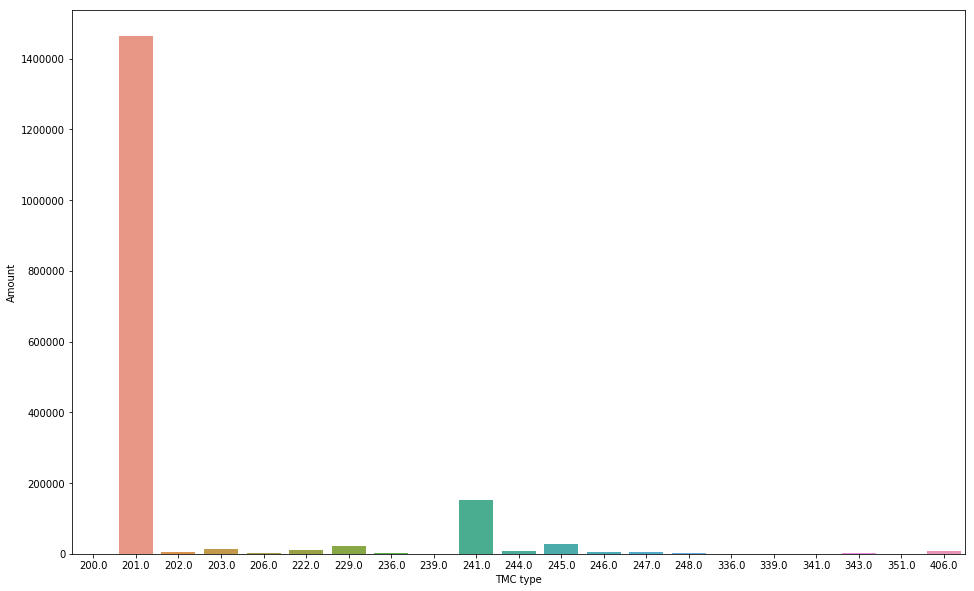
plt.figure(figsize=(16, 10))

ax=sns.barplot(TMC\_counts.index, TMC\_counts)

ax.set(xlabel='TMC type', ylabel='Amount')

Out[12]:

[Text(0, 0.5, 'Amount'), Text(0.5, 0, 'TMC type')]



The graph above shows that the most common type of TMC is 201.0, which according to the event code list

translates to "(Q) accident(s)".

In [13]:

Temperature = data['Temperature(F)']

Severity\_1\_data = data[data['Severity']==1]['Temperature(F)'].mean() Severity\_2\_data = data[data['Severity']==2]['Temperature(F)'].mean() Severity\_3\_data = data[data['Severity']==3]['Temperature(F)'].mean()

Severity\_4\_data = data[data['Severity']==4]['Temperature(F)'].mean()

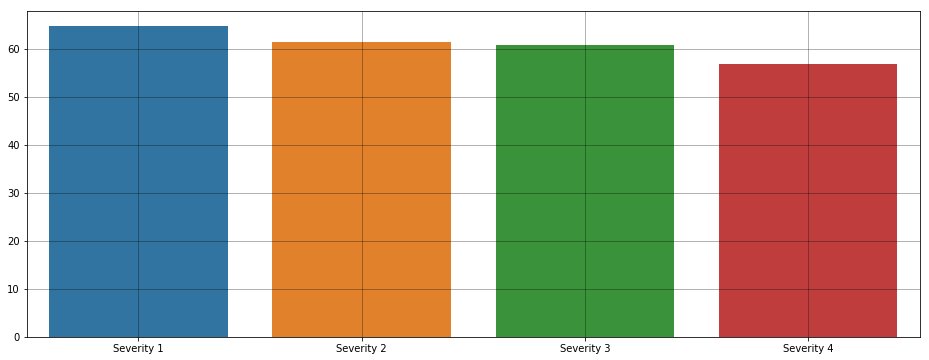
Severity\_labels = ['Severity 1', 'Severity 2', 'Severity 3', 'Severity 4']

Mean\_temp\_by\_severity = [Severity\_1\_data, Severity\_2\_data, Severity\_3\_data, Severi ty\_4\_data]

Is temperature a key factor increasing the severity of an accident?

In [14]:

plt.figure(figsize=(16, 6)) sns.barplot(Severity\_labels, Mean\_temp\_by\_severity) plt.grid(color='black', linestyle='-', linewidth=1, alpha=0.3)



While the differences are small, there is **some** corelation between temperature and severity. What about the weather conditions ?

In [15]:

Weather = data.Weather\_Condition.value\_counts()

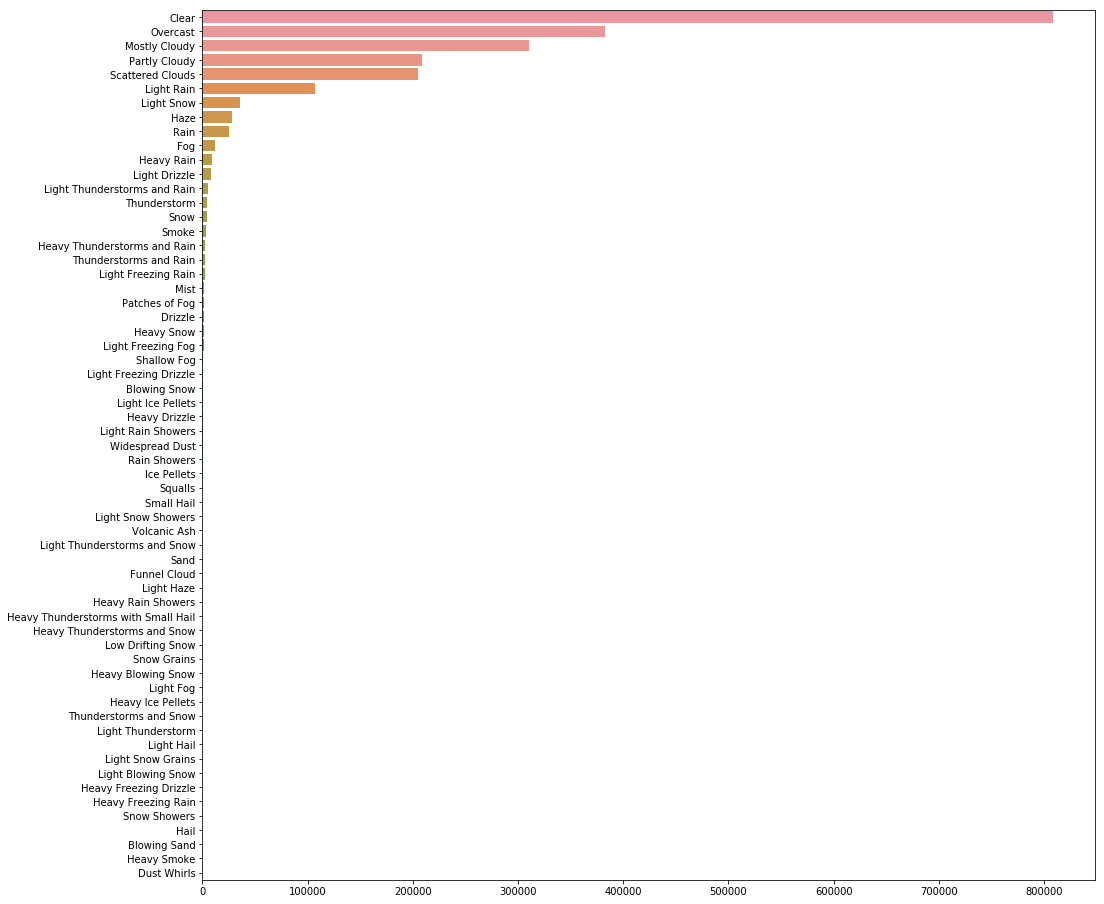
In [16]:

plt.figure(figsize=(16, 16))

sns.barplot(Weather.values, Weather.index)

Out[16]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe7a82b9f98>



In [17]:

severity\_1\_by\_Weather = [] severity\_2\_by\_Weather = [] severity\_3\_by\_Weather = [] severity\_4\_by\_Weather = [] for i **in**  Weather.index:

severity\_1\_by\_Weather.append(data[(data['Severity']==1)&(data['Weather\_Condit ion']==i)].count()['ID'])

severity\_2\_by\_Weather.append(data[(data['Severity']==2)&(data['Weather\_Condit ion']==i)].count()['ID'])

severity\_3\_by\_Weather.append(data[(data['Severity']==3)&(data['Weather\_Condit ion']==i)].count()['ID'])

severity\_4\_by\_Weather.append(data[(data['Severity']==4)&(data['Weather\_Condit

ion']==i)].count()['ID'])

In [18]:

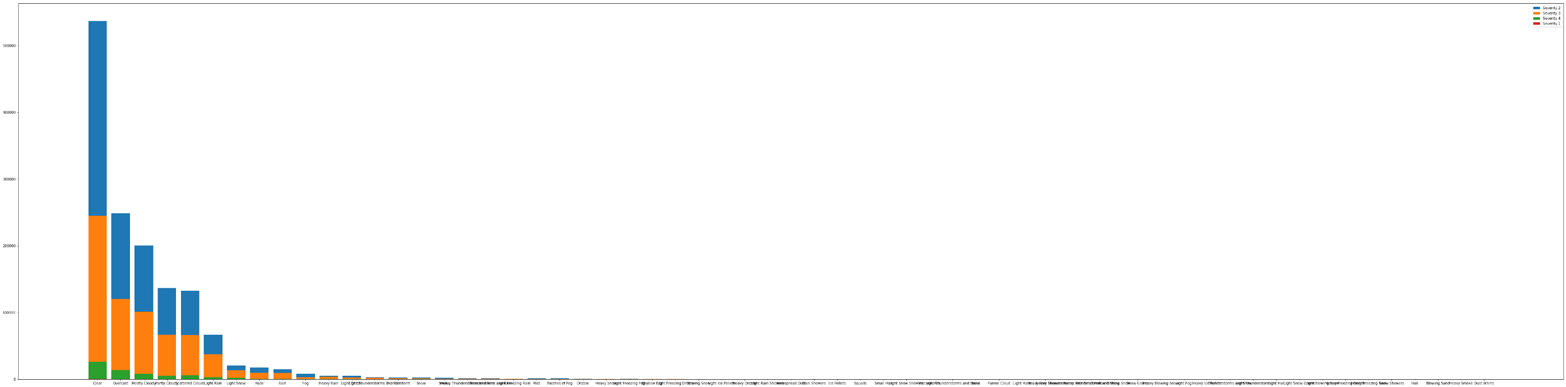
plt.figure(figsize=(80, 20))

plt.bar(Weather.index, severity\_2\_by\_Weather, label='Severity 2') plt.bar(Weather.index, severity\_3\_by\_Weather, label='Severity 3') plt.bar(Weather.index, severity\_4\_by\_Weather, label='Severity 4') plt.bar(Weather.index, severity\_1\_by\_Weather, label='Severity 1')

plt.legend()

Out[18]:

<matplotlib.legend.Legend at 0x7fe7a5dca0b8>



We can observe, that specific weather conditions have slight impact on the severity of an accident.

Let's mute the first rows that are clearly the most frequent conditions and have one more look at less frequent weather conditions.

In [19]:

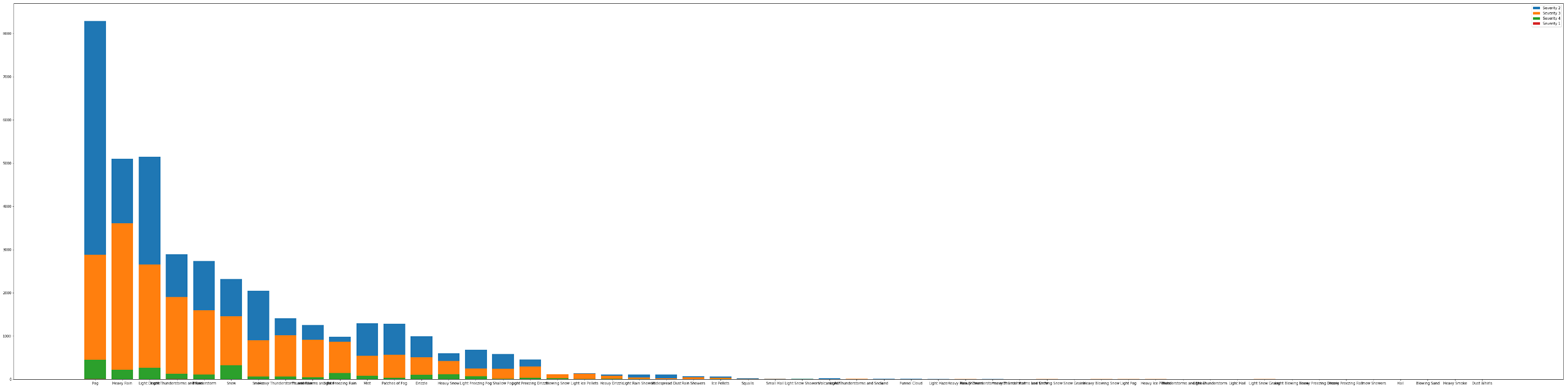
plt.figure(figsize=(80, 20))

plt.bar(Weather.index[9:], severity\_2\_by\_Weather[9:], label='Severity 2') plt.bar(Weather.index[9:], severity\_3\_by\_Weather[9:], label='Severity 3') plt.bar(Weather.index[9:], severity\_4\_by\_Weather[9:], label='Severity 4') plt.bar(Weather.index[9:], severity\_1\_by\_Weather[9:], label='Severity 1')

plt.legend()

Out[19]:

<matplotlib.legend.Legend at 0x7fe7a5aed400>



In [20]:

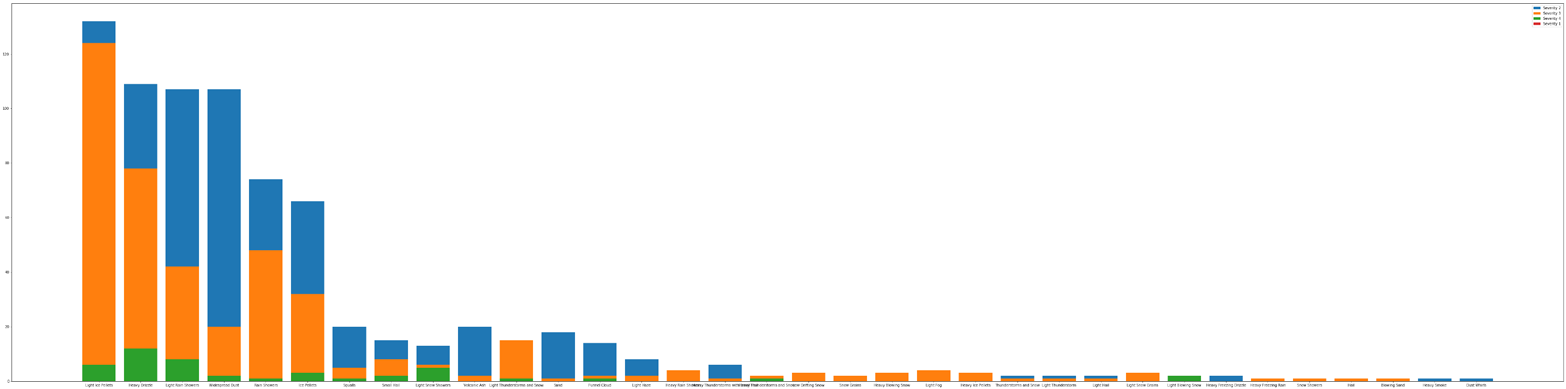
plt.figure(figsize=(80, 20))

plt.bar(Weather.index[27:], severity\_2\_by\_Weather[27:], label='Severity 2') plt.bar(Weather.index[27:], severity\_3\_by\_Weather[27:], label='Severity 3') plt.bar(Weather.index[27:], severity\_4\_by\_Weather[27:], label='Severity 4') plt.bar(Weather.index[27:], severity\_1\_by\_Weather[27:], label='Severity 1')

plt.legend()

Out[20]:

<matplotlib.legend.Legend at 0x7fe7a5695f98>



What were the weather conditions of the accidents where severity is the highest?

In [21]:

percentage\_severity\_1 = [] percentage\_severity\_2 = [] percentage\_severity\_3 = [] percentage\_severity\_4 = []

for i **in** range(len(severity\_1\_by\_Weather)): percentage\_severity\_1.append((severity\_1\_by\_Weather[i]/Weather[i])\*100) percentage\_severity\_2.append((severity\_2\_by\_Weather[i]/Weather[i])\*100) percentage\_severity\_3.append((severity\_3\_by\_Weather[i]/Weather[i])\*100)

percentage\_severity\_4.append((severity\_4\_by\_Weather[i]/Weather[i])\*100)

In [22]:

percentage\_severity\_3[1]+percentage\_severity\_2[1]+percentage\_severity\_1[1]+percen tage\_severity\_4[1]

Out[22]:

99.9992156493639

In [23]:

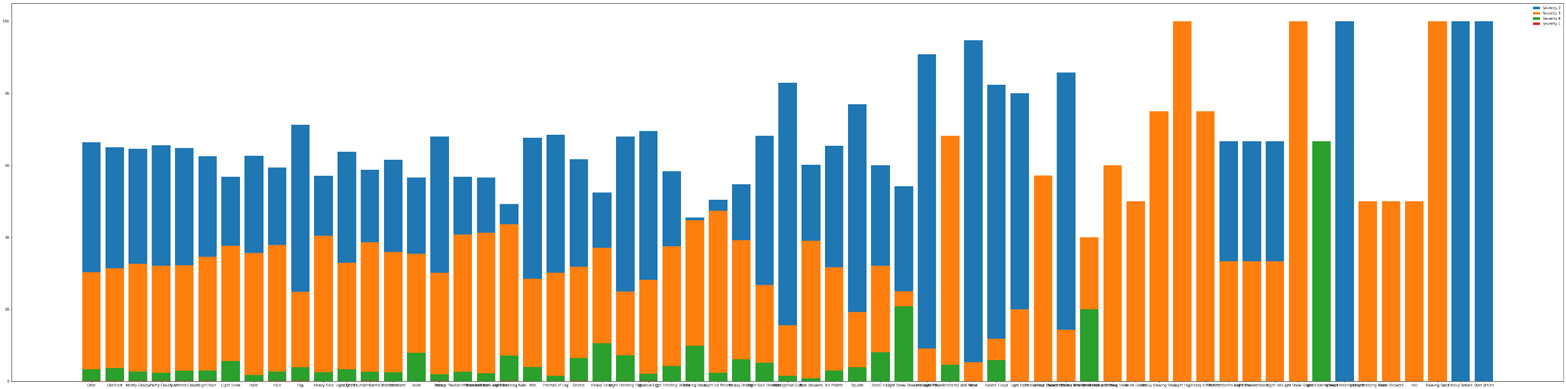
plt.figure(figsize=(80, 20))

plt.bar(Weather.index, percentage\_severity\_2, label='Severity 2') plt.bar(Weather.index, percentage\_severity\_3, label='Severity 3') plt.bar(Weather.index, percentage\_severity\_4, label='Severity 4') plt.bar(Weather.index, percentage\_severity\_1, label='Severity 1')

plt.legend()

Out[23]:

<matplotlib.legend.Legend at 0x7fe76aff9a20>



It seems that accidents of highest severity occur during weather conditions such as **thunderstorm, heavy rain, snow (in general) and blowing sand**, the worst conditions being **light blowing snow** and **heavy thunderstorms and snow**.

This Notebook has been released under the [Apache 2.0](http://www.apache.org/licenses/LICENSE-2.0) open source license.

Did you find this Notebook useful? Show your appreciation with an upvote



40

Data



Data Sources



US Accidents (3.0 million records)



US\_Accidents\_May19.csv

49 columns

[](https://www.kaggle.com/sobhanmoosavi/us-accidents/download/p44lgYpA1uUpTDM3fEsI%2Fversions%2F7rBcGxF5y4DRaC7WGHpu%2Ffiles%2FUS_Accidents_May19.csv?datasetVersionNumber=1)

[](https://www.kaggle.com/sobhanmoosavi/us-accidents)



[US Accidents (3.0 million records)](https://www.kaggle.com/sobhanmoosavi/us-accidents)

A Countrywide Traffic Accident Dataset (2016 - 2019) Last Updated: 9 months ago (Version 1 of 3)

About this Dataset

Description

This is a countrywide traffic accident dataset, which covers 49 states of the United States. The data is collected from February 2016

to December 2019, using several data providers, including two APIs which provide streaming traffic event data. These APIs broadcast traffic events captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road -networks. Currently, there are about 3.0 million accident records in this dataset. Check [here](https://smoosavi.org/datasets/us_accidents) to learn more about this dataset.

Acknowledgements

Please cite the following papers if you use this dataset:

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. “[A Countrywide Traffic Accident](https://arxiv.org/abs/1906.05409)

[Dataset](https://arxiv.org/abs/1906.05409).”, 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."](https://arxiv.org/abs/1909.09638) In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

Content

This data has been collected in real-time, using multiple Traffic APIs. Currently, it contains data which is collected from February

2016 to December 2019 for the Contiguous United States. Check [here](https://smoosavi.org/datasets/us_accidents) to learn more about this dataset.

Inspiration

US-Accidents can be used for numerous applications such as real-time accident prediction, studying accident hotspot locations, casualty analysis and extracting cause and effect rules to predict accidents, or studying the impact of precipitation or other environmental stimuli on accident occurrence.

Usage Policy and Legal Disclaimer

This dataset is being distributed only for Research purposes, under Creative Commons Attribution -Noncommercial-ShareAlike license

(CC BY-NC-SA 4.0). By clicking on download button(s) below, you are agreeing to use this data only for non-commercial, research, or academic applications. You may need to cite the above papers if you use this dataset.

Run Info

Succeeded

True

Exit Code

0

Docker Image Name [kaggle/python@sha256:8250d2bd6f978ec5b11da50735a041cf4dbfef847d2b33f3855c8cabe7e6ef0d](https://registry.hub.docker.com/u/kaggle/python%40sha256%3A8250d2bd6f978ec5b11da50735a041cf4dbfef847d2b33f3855c8cabe7e6ef0d/) ([Dockerfile](https://github.com/Kaggle/docker-python%40sha256%3A8250d2bd6f978ec5b11da50735a041cf4dbfef847d2b33f3855c8cabe7e6ef0d/blob/master/Dockerfile))

Timeout Exceeded

False

Run Time

237 seconds

Queue Time

0 seconds

Output Size

0

Used All Space

False

Failure Message

Log



Download Log

Time Line # Log Message

2.8s 1 [NbConvertApp] Converting notebook notebook .ipynb to notebook

6.6s 2 [NbConvertApp] Executing notebook with kernel: python3

234.8s 3 [NbConvertApp] Writing 453062 bytes to notebook .ipynb

235.6s 4 [NbConvertApp] Converting notebook notebook .ipynb to html

236.6s 5 [NbConvertApp] Support files will be in results files/ [NbConvertApp] Making directory results files

236.6s 6 [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Making directory results files [NbConvertApp] Writing 331655 bytes to results .html

236.6s 7

236.6s 9 Complete. Exited with code 0.

Comments (2) All Comments

Sort by

Hotness



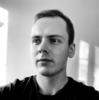
[Surya Prakash](https://www.kaggle.com/suryap2011) • Posted on Latest Version • 2 months ago • Options • Reply





1

That's great! Thanks for sharing. May be you want to review legend/labels to the first chart.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | Options | • | Reply |  |
| [Mikolaj Witkowski](https://www.kaggle.com/vtech6) | Kernel Author | • Posted on Latest Version • 2 months ago • |  |
|  |
|   0 | | | | | | |
| Thank you! Once I'm back from vacation I'll surely look into that, for now my connection doesn't allow me to do much. | | | | | | |

Basic Analysis- US Accidents I Kaggle

[**g;**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/notebook)

[**Notebook**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/notebook)

[**§33**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/data)

[**O.a1.a**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/data)

[-](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/comments)

**L**[**ug**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/log)

[**Comments**](https://www.kaggle.com/vtech6/basic-analysis-us-accidents/comments)