KHANH TRAN

INFO 250 – 001

Date: August 6, 2020

Assignment 4

Plan Your Final Project

1. Identify a dataset for your final project.

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

2. Describe the dataset: (1) what it is about, (2) other information about the dataset a designer should know (such as its history and conditions under which the data is collected) and (3) statistical descriptions of the variables that you are going to use.

- This dataset offers a great volume of data that are meaningful to the goals of the project. There are currently about 3.5 million accident records in this dataset. It covers 49 states of the USA, and the data were collected from February 2016 to June 2020, using two APIs that provide streaming traffic incident (or event) data. Along with the large number of records, this dataset also provides a wide range of attributes for each accident. With 49 columns, analysts can observe and discover many faces of the accidents such as starting-ending time, exact starting-ending location, address, weather conditions, existed crossings, junctions, or bumps, etc.

```
# Quick overview of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3513617 entries, 0 to 3513616
Data columns (total 49 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Source                 object
 2   TMC                    float64
 3   Severity               int64
 4   Start_Time             object
 5   End_Time               object
 6   Start_Lat              float64
 7   Start_Lng              float64
 8   End_Lat                float64
 9   End_Lng                float64
 10  Distance(mi)           float64
 11  Description            object
 12  Number                 float64
 13  Street                 object
 14  Side                   object
 15  City                   object
 16  County                 object
 17  State                  object
 18  Zipcode                object
 19  Country                object
 20  Timezone               object
 21  Airport_Code           object
 22  Weather_Timestamp      object
 23  Temperature(F)         float64
 24  Wind_Chill(F)          float64
 25  Humidity(%)            float64
 26  Pressure(in)           float64
 27  Visibility(mi)         float64
 28  Wind_Direction         object
 29  Wind_Speed(mph)        float64
 30  Precipitation(in)      float64
 31  Weather_Condition      object
 32  Amenity                bool
 33  Bump                   bool
 34  Crossing               bool
 35  Give_Way               bool
 36  Junction               bool
 37  No_Exit                bool
 38  Railway                bool
 39  Roundabout             bool
 40  Station                bool
 41  Stop                   bool
 42  Traffic_Calming        bool
 43  Traffic_Signal         bool
 44  Turning_Loop           bool
 45  Sunrise_Sunset         object
 46  Civil_Twilight         object
 47  Nautical_Twilight      object
 48  Astronomical_Twilight  object
dtypes: bool(13), float64(14), int64(1), object(21)
memory usage: 1008.6+ MB
```

Figure 1: Overview of the dataset

- Here are the meaning and type of data for each attribute in the data file:

  - ID: This is a unique identifier of the accident record with the format of A-x (x is a number).
  - Source: Indicates source of the accident report (i.e. the API which reported the accident.).
  - TMC: A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event. The codes are floats.
  - 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).
  - Start_Time: Shows start time of the accident in local time zone. The data in this attribute have the object type in the original dataset, but they should be converted to datetime type later on.
  - 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. The data in this attribute have the object type in the original dataset, but they should be converted to datetime type later on.
  - Start_Lat: Shows latitude in GPS coordinate of the start point. This attribute contains floats.

- Start_Lng: Shows longitude in GPS coordinate of the start point. This attribute contains floats.
- End_Lat: Shows latitude in GPS coordinate of the end point. This attribute contains floats.
- End_Lng: Shows longitude in GPS coordinate of the end point. This attribute contains floats.
- Distance(mi): The length of the road extent affected by the accident. This attribute contains floats.
- Description: Shows natural language description of the accident. This attribute contains strings.
- Number: Shows the street number in address field. This attribute contains floats.
- Street: Shows the street name in address field.
- Side: Shows the relative side of the street (Right/Left) in address field. This attribute contains strings.
- City: Shows the city in address field. This attribute contains strings.
- County: Shows the county in address field. This attribute contains strings.
- State: Shows the state in address field. This attribute contains strings.
- Zipcode: Shows the zipcode in address field. This attribute contains strings.
- Country: Shows the country in address field. This attribute contains strings.
- Timezone: Shows timezone based on the location of the accident (eastern, central, etc.). This attribute contains strings.
- Airport_Code: Denotes an airport-based weather station which is the closest one to location of the accident. This attribute contains strings.
- Weather_Timestamp: Shows the time-stamp of weather observation record (in local time). This attribute contains strings.
- Temperature(F): Shows the temperature (in Fahrenheit). This attribute contains floats.
- Wind_Chill(F): Shows the wind chill (in Fahrenheit). This attribute contains floats.
- Humidity(%): Shows the humidity (in percentage). This attribute contains floats.
- Pressure(in): Shows the air pressure (in inches). This attribute contains floats.
- Visibility(mi): Shows visibility (in miles). This attribute contains floats.
- Wind_Direction: Shows wind direction. This attribute contains strings.
- Wind_Speed(mph): Shows wind speed (in miles per hour). This attribute contains floats.
- Precipitation(in): Shows precipitation amount in inches, if there is any. This attribute contains floats.

- Weather_Condition: Shows the weather condition (rain, snow, thunderstorm, fog, etc.). This attribute contains strings.
- Amenity: A POI annotation which indicates presence of amenity in a nearby location. This attribute contains booleans.
- Bump: A POI annotation which indicates presence of speed bump or hump in a nearby location. This attribute contains booleans.
- Crossing: A POI annotation which indicates presence of crossing in a nearby location. This attribute contains booleans.
- Give_Way: A POI annotation which indicates presence of give_way in a nearby location. This attribute contains booleans.
- Junction: A POI annotation which indicates presence of junction in a nearby location. This attribute contains booleans.
- No_Exit: A POI annotation which indicates presence of no_exit in a nearby location. This attribute contains booleans.
- Railway: A POI annotation which indicates presence of railway in a nearby location. This attribute contains booleans.
- Roundabout: A POI annotation which indicates presence of roundabout in a nearby location. This attribute contains booleans.
- Station: A POI annotation which indicates presence of station in a nearby location. This attribute contains booleans.
- Stop: A POI annotation which indicates presence of stop in a nearby location. This attribute contains booleans.
- Traffic_Calming: A POI annotation which indicates presence of traffic calming in a nearby location. This attribute contains booleans.
- Traffic_Signal: A POI annotation which indicates presence of traffic signal in a nearby location. This attribute contains booleans.
- Turning_Loop: A POI annotation which indicates presence of turning loop in a nearby location. This attribute contains booleans.
- Sunrise_Sunset: Shows the period of day (i.e. day or night) based on sunrise/sunset. This attribute contains strings ("Day" or "Night").

- Civil_Twilight: Shows the period of day (i.e. day or night) based on civil twilight. This attribute contains strings ("Day" or "Night").
- Nautical_Twilight: Shows the period of day (i.e. day or night) based on nautical twilight. This attribute contains strings ("Day" or "Night").
- Astronomical_Twilight: Shows the period of day (i.e. day or night) based on astronomical twilight. This attribute contains strings ("Day" or "Night").

- Here are some quick statistics for the most important attributes along with the descriptions. The attributes include: "Severity", "Start_Time","State", and "Temperature(F)".

```
# Severity

# Mean
mean_sev = np.mean(df["Severity"].values)
# Mode
mode_sev = stats.mode(df["Severity"].values)[0]
# Count for mode
count_mode_sev = stats.mode(df["Severity"].values)[1]

print("Mean of 'Severity':", mean_sev)
print("Mode of 'Severity':", mode_sev[0])
print("Count of Mode:", count_mode_sev[0])
```

```
Mean of 'Severity': 2.3399286262560772
Mode of 'Severity': 2
Count of Mode: 2373210
```

Figure 2: "Severity" Stats

On average, the severity level is 2.34. Considering 1 is the least impact on traffic and 4 is a significant one, 2.34 is slightly above the average level of impact. There are 2,373,210 cases that fall into level 2 of severity, which is also the mode of this attribute. In other words, based on the dataset, every 2 out of 3 accidents cause a severity of level 2.

Next, let's look at "Start_Time". However, we won't go into details here. We only look at the starting hours of the accident to find out at what time of a day did the accidents happen the most.

```
# Start_Time

# Convert to datetimes
df["Start_Time"] = pd.to_datetime(df["Start_Time"])

# Extract starting hours
start_hours = pd.DatetimeIndex(df["Start_Time"]).hour

# Mode of starting hours
mode_start = stats.mode(start_hours)[0]
# Counts for mode
counts_mode_start = stats.mode(start_hours)[1]

print("The time when accidents happened the most is:", mode_start[0])
print("Number of such cases:", counts_mode_start[0])
```

```
The time when accidents happened the most is: 8
Number of such cases: 326257
```

Figure 3: "Start_Time" Stats

The numbers speak for themselves here. We find out that the largest number of cases happened in the morning at around 8am. Logically, this also fits the fact that 8am is usually one of the rush hours of the day when the roads are the most crowded.

Lastly, we will look at the temperatures during the accidents. Since column "Temperature(F)" has a lot of missing values, we will leave them out before conducting calculations.

```
# Temperature(F)

# Leave out missing values
temps = [
    temp for temp in df["Temperature(F)"].values
    if np.isnan(temp)==False
    ]
# Mean
mean_temp = np.mean(temps)
# Mode
mode_temp = stats.mode(temps)[0]
# Count for mode+-2 Farenheit degreee
count_mode_temp = stats.mode(temps)[1]
count_mode_temp += temps.count(mode_temp[0]-2)
count_mode_temp += temps.count(mode_temp[0]+2)

print("Mean of 'Temperature(F)':", mean_temp)
print("Mode of 'Temperature(F)':", mode_temp[0])
print("Count of Mode+-2 Farenheit degreee:", count_mode_temp[0])
```

```
Mean of 'Temperature(F)': 61.93511900773952
Mode of 'Temperature(F)': 68.0
Count of Mode+-2 Farenheit degreee: 202499
```

Figure 4: "Temperature(F)" Stats.

According to the result, we know that the average temperature at the time of the accidents is 61.94 F. The temperature that appear the most time is 68.0 F. With temperature, we have a more flexible counting approach as this attribute has a rather high variance. I increased the range to 66.0 to 70.0 F and was able to find out over 200,000 cases fall into this range of temperature.

3. Identify the target audience for your work (at least one key audience group) and discuss what they may know about your topic and what aspect of your data may be interested by them.

The audience of this work should be anyone who is interested in knowing more about the statistical aspect of car accidents in the country. However, the most important target is the people who directly participate in traffic in the U.S. The visualizations that I am going to provide in this project will hopefully display the cause and effect rules of a part of the accidents. The graphs will also aid the drivers in terms of raising their awareness of the potential accident causes. In my opinion, the inexperienced drivers will benefit the most from the information. The connection between location, time, weather, etc. and the number and level of severity of the accidents will be the most crucial points to which I think the audience will pay the most attention. For example, how rain and snow affect the tendency of accident occurrence, or how the number of accidents differ according to the time of the day or the country area.

4. Identify the questions that the audience might be curious about your data/visualization.

About the data, the audience can be curious about the size of the dataset, the method of recording the data, and the time period during which the data was recorded. People can also care about the "cleanliness" of the data, its integrity and credibility. Additionally, some may have the need to be well informed about the statistics and figures of each attribute in the dataset, especially the features

that they find themselves familiar with like the state and city they are living in or the particular time of the day when they are out on the road most often.

About the visualization, I believe the most frequent question will be about “How do I interpret this graph?” because some of the graphs that I’m planning to use will be rather a little more complicated than simple bar graphs. They will question about something like: “What does the x axis represent?”, “What does the y axis represent?”, “How do I understand this graph?”, “How many things you are comparing here?”, “Why chose this graph?”, etc.

5. Design (but not implement) how your visualization will look like. (Include the following subquestions.)

- **What format of visualization you are going to use?**
- **Where do you plan to publish it?**
- **What tool(s) are you going to use for the visualization?**
- **What graph type(s) are you going to use?**
- **What data variables are you going to use?**
- **How are you going to map the data variables to the visual patterns?**

- I am planning to use Python-based tools to create the visualization, so Jupyter Notebook is my current format for my visualization. The plots will be created right under each code cell, followed by markdown descriptions if needed. This format is a good choice for displaying graphs along with their source code, thus providing a little more information for the audience. However, some may find this format less appealing than, for example, Tableau dashboards because they are fond of something that mainly focuses on the graphs and not interested in chunks of confusing code.
- I am planning to publish it to my GitHub account.

- I am going to use Python-based library matplotlib and seaborn along with module Basemap from library mpl_toolkits for the visualization. Some extra packages needed for data wrangling and cleaning will include pandas, numpy, scipy, and math.

- I am planning to implement both basic and intermediate types of graphs. The majority will the basic ones like pie chart, line graph, and bar graph. A little more advanced types should be a histogram with a distribution line, a map with a color bar (somewhat similar to what we have in Tableau), and scatter plot. I will try to use as many kinds of visualization as possible because the dataset has a wide variety of attributes which will require different visualizing approaches considering the different nature of data types and their relationships.

- I am going to use “Severity”, “Start_Time”, “End_Time”, “Start_Lat”, “End_Lat”, “Start_Lng”, “End_Lng”, “State”, “Weather_Condition”, “Temperature(F)”, “Precipitation(in)”, and “Distance(mi)”. This is just a temporary list. The final work may include more variables in the analysis.

- At the moment, I am thinking of using color pattern to showcase the level of severity on a map. This idea will include “Severity”, “Start_Lat”, “End_Lat”, “Start_Lng”, and “End_Lng”. Color patterns will also be used in pie charts where I can display the proportion of cases based on severity level or weather conditions. A stacked bar graph is a good option for implementing color pattern such as different weather condition (rain, snow, etc.) have their corresponding cases count stacked upon each other along an x axis of months. For basic bar graphs, the x axis and y axis will respectively represent categorical and numerical data. The categorical data can be the U.S. states or severity level while the numerical data can be cases count. Size pattern will be used in graphs like a scatter plot where I can map the number of cases to the different circle sizes.