accidents

January 4, 2020

1 Práctica 2:

2 Los factores que influyen la cantidad y la severidad de los accidentes de tráfico

2.1 Índice:

3 1. Descripción de los conjuntos de datos

Comentario del estudiante: Disculpen por favor los errores de ortografía. El castellano no es mi idioma nativo. En este análisis vamos a combinar dos conjuntos de datos a fin de modelizar cuales son los factores que influyen a la cantidad y a la severidad de los accidentes. Después de haber identificado a los factores más importantes, con esta información, se puede predecir con machine learning la cantidad de accidentes según diferentes factores. Nos vamos en enfocar en tres cosas: - El número de accidentes relativo al total de tráfico por hora - Los accidentes según el nivel de severidad y la hora - La nfluencia del tiempo meteorológico en la severidad de un accidente

Esto podría permitir, por ejemplo, planear la repartición de los equipos de rescates durante el año y según las condiciones para optimizar el socorro de las personas después de los accidentes u optimizar la seguridad de los coches según los factores más peligrosos.

Dataset: US Accidents (2.25 million records) A Countrywide Traffic Accident Dataset (2016 - 2019) Autor: Sobhan Moosavi

"Se trata de un conjunto de datos de accidentes de tráfico de todo el país, que cubre 49 estados de los Estados Unidos. Los datos se recogen desde febrero de 2016 hasta marzo de 2019, utilizando

varios proveedores de datos, incluyendo dos API que proporcionan datos de eventos de tráfico en flujo. Estas APIs transmiten eventos de tráfico capturados por una variedad de entidades, tales como los departamentos de transporte de los Estados Unidos y de los estados, agencias de aplicación de la ley, cámaras de tráfico y sensores de tráfico dentro de las redes de carreteras. Actualmente, hay alrededor de 2,25 millones de registros de accidentes en este conjunto de datos." (Traducción de la descripción del conjunto de datos)

Dataset: US Traffic, 2015 7.1M Daily Traffic Volume Observations, By Hour and Direction

"Este conjunto de datos de ~2gb contiene volúmenes diarios de tráfico, agrupados por horas. También se incluye información sobre la dirección del flujo y la colocación del sensor." (Traducción de la descripción del conjunto de datos) Descripción de la metodología: Con el dataset US Accidents tendremos varias informaciones sobre los accidentes que ocurrieron, sin embargo, la información está sesgada. Por ejemplo, si queremos saber la influencia de las condiciones condiciones meteorológicas (lluvia, sol etc.) sobre los accidentes, el problema es que tenemos solo al tiempo que hacía cuando hubo un accidente, pero no sabemos nada de todos los accidentes que no ocurrieron. Para ponerlo más claro: es como estar en un gimnasio en Barcelona y preguntar a todas las personas allá el número de veces que se entrenan al mes y después utilizar el promedio para decir que la población de Barcelona va al gimnasio cada semana un cierto número de veces. El problema es que se toma en cuenta solo a la gente que ya va al gimnasio, pero no se toma en cuenta a toda la gente que no va al gimnasio.

Volviendo al tráfico. Imaginamos que hace un tiempo claro el 99% del tiempo y 1% del tiempo lluvia. Cuando hace tiempo claro ocurrieron 1000 accidentes y cuando lluvia ocurrieron 100 accidentes. Si tomamos solo a la proporción de accidentes obtenemos: "Hay 10 veces más accidentes cuando hay tiempo claro que cuando lluvia". De manera absoluta está frase es correcta, pero da la impresión de que es mucho más peligroso conducir cuando hace tiempo claro. Lo que aquí NO se toma en cuenta es que el 99% del tiempo hace un tiempo claro, por eso a pesar de haber un total de accidentes mayor no significa que es más inseguro.

Ahora vamos a calcular el riesgo de accidente relativo al tiempo que hace como si hiciera buen o mal tiempo siempre: Lluvia: 100 x 100 / 1 = 10000 Tiempo claro: 1000 x 100 / 99 = 1010 Aquí podemos ver que el riesgo es alrededor de 10 veces más alto por unidad de tiempo cuando lluvia.

Otro ejemplo similar sería la cantidad de accidentes según la hora del día: durante el día hay más tráfico, cuando más tráfico hay, mayor es la cantidad de accidente absoluta, pero no significa que la cantidad de accidente relativa sea mayor.

Ahora para obtener los factores que influyen a los accidentes con exactitud, necesitaríamos en adición a los datos sobre los accidentes a los datos sobre el tráfico y las condiciones condiciones meteorológicas en cada lugar y en cada momento. Obviamente no existen datos así con toda la información. Por esto utilizamos el dataset "US Traffic, 2015" el cual permite tener una estimación del tráfico general en cada hora, lo cual permitirá calcular la cantidad relativa de accidentes según la hora del día.

Cabe destacar que US Traffic contiene los datos sobre el tráfico en el año 2015 y que US Accidents contiene los datos sobre los accidentes del 2016 al 2019. No se trata del mismo periodo, sin embargo, los datos US Traffic permitirán dar una estimación de la cantidad de tráfico, sobre todo cuando se analiza a un periodo de tiempo largo de, por ejemplo, un mes. El tráfico total de los Estados Unidos en mayo 2015 es seguramente similar al tráfico total en mayo 2016.

3.1 Dataset: US Accidents (2.25 million records) - columnas

0 ID: Este es un identificador único del registro de accidentes. 1 Source: Indica la fuente del informe del accidente (es decir, la API que informó del accidente). 2 TMC: Un accidente de tráfico

puede tener un código de canal de mensajes de tráfico (TMC) que proporciona una descripción más detallada del evento. 3 Severity: Muestra la gravedad del accidente, un número entre 1 y 4, donde 1 indica el menor impacto en el tráfico (es decir, un retraso corto como resultado del accidente) y 4 indica un impacto significativo en el tráfico (es decir, un retraso largo). 4 Start_Time: Muestra la hora de inicio del accidente en el huso horario local. 5 End_Time: Muestra la hora de finalización del accidente en el huso horario local. 6 Start_Lat: Muestra la latitud en coordenadas GPS del punto de inicio. 7 Start_Lng: Muestra la longitud en la coordenada GPS del punto de inicio. 8 End_Lat: Muestra la latitud en la coordenada de GPS del punto final. 9 End_Lng: Muestra la longitud en coordenadas GPS del punto de final. 10 Distance(mi): La longitud de la extensión de la carretera afectada por el accidente. 11 Description: Muestra la descripción en lenguaje natural del accidente. 12 Number: Muestra el número de la calle en el campo de dirección. 13 Street: Muestra el nombre de la calle en el campo de dirección. 14 Side: Muestra el lado relativo de la calle (derecha/izquierda) en el campo de dirección. 15 City: Muestra la ciudad en el campo de direcciones. 16 County: Muestra el condado en el campo de direcciones. 17 State: Muestra el estado en el campo de dirección. 18 Zipcode: Muestra el código postal en el campo de dirección. 19 Country: Muestra el país en la casilla de direcciones. 20 Timezone: Muestra la zona horaria basada en la ubicación del accidente. 21 Airport_Code: Indica una estación meteorológica en un aeropuerto que es la más cercana al lugar del accidente. 22 Weather_Timestamp: Muestra el sello de tiempo del registro de observación meteorológica (en hora local). 23 Temperature(F): Muestra la temperatura (en grados Fahrenheit). 24 Wind_Chill(F): Muestra la sensación térmica (en Fahrenheit). 25 Humidity(%): Muestra la humedad (en porcentaje). 26 Pressure(in): Muestra la presión del aire (en pulgadas). 27 Visibility(mi): Muestra la visibilidad (en millas). 28 Wind_Direction: Muestra la dirección del viento. 29 Wind_Speed(mph): Muestra la velocidad del viento (en millas por hora). 30 Precipitation(in): Muestra la cantidad de precipitación en pulgadas, si la hay. 31 Weather_Condition: Muestra las condiciones climáticas (lluvia, nieve, tormenta, niebla, etc.) 32 Amenity: Indica la presencia de servicios en un lugar cercano. 33 Bump: Indica la presencia de un badén o joroba en un lugar cercano. 34 Crossing: Indica la presencia de un cruce en un lugar cercano. 35 Give_Way: Indica la presencia de un punto muerto en un lugar cercano. 36 Junction: Indica la presencia de un cruce en un lugar cercano. 37 No_Exit: Indica la presencia de no_salida en un lugar cercano. 38 Railway: Indica la presencia de una vía férrea en un lugar cercano. 39 Roundabout: Indica la presencia de una rotonda en un lugar cercano. 40 Station: Indica la presencia de una estación en un lugar cercano. 41 Stop: Indica la presencia de una parada en un lugar cercano. 42 Traffic_Calming: Indica la presencia de traffic_calming en una ubicación cercana. 43 Traffic_Signal: Indica la presencia de una señal de tráfico en una ubicación cercana. 44 Turning_Loop: Una anotación PDI que indica la presencia de bucle_de_vuelta en una ubicación cercana. 45 Sunrise_Sunset: Muestra el periodo del día (es decir, día o noche) basado en la salida/puesta del sol. 46 Civil_Twilight: Muestra el periodo de día (es decir, día o noche) basado en el crepúsculo civil. 47 Nautical_Twilight: Muestra el período de día (es decir, día o noche) basado en el crepúsculo náutico. 48 Astronomical_Twilight: Muestra el período de día (es decir, día o noche) basado en el crepúsculo astronómico.

3.2 Dataset: US Traffic, 2015 - columnas

0 date: fecha 1 day_of_data: día 2 day_of_week: día de la semana 3 direction_of_travel: dirección del trayecto 4 direction_of_travel_name: nombre de la dirección del trayecto 5 fips_state_code: código del estado de fips 6 functional_classification: clasificación funcional 7 functional_classification_name: nombre de la clasificación funcional 8 lane_of_travel: carril de circulación 9 month_of_data: mes 10 record_type: clase de registro 11 restrictions: restric-

ciones 12 station_id: identificación de estación 13 traffic_volume_counted_after_0000_to_0100: volumen de tráfico 0-1am 14 traffic_volume_counted_after_0100_to_0200: volumen de tráfico 1-2am 15 traffic_volume_counted_after_0200_to_0300: volumen de tráfico 2-3am 16 traffic_volume_counted_after_0300_to_0400: volumen de tráfico 3-4am 17 traffic_volume_counted_after_0400_to_0500: volumen de tráfico 4-5am 18 traffic_volume_counted_after_0500_to_0600: volumen de tráfico 5-6am 19 traffic_volume_counted_after_0600_to_0700: volumen de tráfico 6-7am 20 traffic_volume_counted_after_0700_to_0800: volumen de tráfico 7-8am 21 traffic_volume_counted_after_0800_to_0900: volumen de tráfico 8-9am 22 traffic_volume_counted_after_0900_to_1000: volumen de tráfico 9-10am 23 traffic_volume_counted_after_1000_to_1100: volumen de tráfico 10-11am 24 traffic_volume_counted_after_1100_to_1200: volumen de tráfico 11am-12pm 25 traffic_volume_counted_after_1200_to_1300: volumen de tráfico 12pm-1pm 26 traffic_volume_counted_after_1300_to_1400: volumen de tráfico 1-2 27 traffic_volume_counted_after_1400_to_1500: volumen de tráfico 2-3pm 28 traffic_volume_counted_after_1500_to_1600: volumen de tráfico 3-4pm 29 traffic_volume_counted_after_1600_to_1700: volumen de tráfico 4-5pm 30 traffic_volume_counted_after_1700_to_1800: volumen de tráfico 5-6pm 31 traffic_volume_counted_after_1800_to_1900: volumen de tráfico 6-7pm 32 traffic_volume_counted_after_1900_to_2000: volumen de tráfico 7-8pm 33 traffic_volume_counted_after_2000_to_2100: volumen de tráfico 8-9pm 34 traffic_volume_counted_after_2100_to_2200: volumen de tráfico 9-10pm 35 traffic_volume_counted_after_2200_to_2300: volumen de tráfico 10-11pm 36 traffic_volume_counted_after_2300_to_2400: volumen de tráfico 11-12 37 year_of_data: año de los datos

4 2. Análisis exploratorio, integración y selección de los datos de interés a analizar.

[1]:
```
# Importa a las librerías que vamos a necesitar
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Aumenta el ancho máximo de columna de Jupyter a 50
pd.set_option('display.max_columns', 50)
```

4.1 2.1. Conjunto de datos: US Accidents

[2]:
```
# Importa el conjunto de datos US Accidents
accidents = pd.read_csv("US_Accidents_May19.csv")
# accidents = accidents[:200000].copy()
```

[3]:
```
# Calcula la cantidad de líneas y columnas
d = accidents.shape
```

```
print("US Accidents tiene {} líneas y {} columnas.".format(d[0], d[1]))
```

```
US Accidents tiene 2243939 líneas y 49 columnas.
```

[4]:
```
# Muestra las primeras líneas
accidents.head(5)
```

[4]:
```
     ID    Source    TMC  Severity           Start_Time             End_Time  \
0  A-1  MapQuest  201.0         3  2016-02-08 05:46:00  2016-02-08 11:00:00
1  A-2  MapQuest  201.0         2  2016-02-08 06:07:59  2016-02-08 06:37:59
2  A-3  MapQuest  201.0         2  2016-02-08 06:49:27  2016-02-08 07:19:27
3  A-4  MapQuest  201.0         3  2016-02-08 07:23:34  2016-02-08 07:53:34
4  A-5  MapQuest  201.0         2  2016-02-08 07:39:07  2016-02-08 08:09:07

   Start_Lat  Start_Lng  End_Lat  End_Lng  Distance(mi)  \
0  39.865147 -84.058723      NaN      NaN          0.01
1  39.928059 -82.831184      NaN      NaN          0.01
2  39.063148 -84.032608      NaN      NaN          0.01
3  39.747753 -84.205582      NaN      NaN          0.01
4  39.627781 -84.188354      NaN      NaN          0.01

                                         Description  Number  \
0  Right lane blocked due to accident on I-70 Eas...     NaN
1  Accident on Brice Rd at Tussing Rd. Expect del...  2584.0
2  Accident on OH-32 State Route 32 Westbound at ...     NaN
3  Accident on I-75 Southbound at Exits 52 52B US...     NaN
4  Accident on McEwen Rd at OH-725 Miamisburg Cen...     NaN

                     Street Side          City      County State     Zipcode  \
0                    I-70 E    R        Dayton  Montgomery    OH       45424
1                 Brice Rd     L  Reynoldsburg    Franklin    OH  43068-3402
2             State Route 32    R  Williamsburg    Clermont    OH       45176
3                    I-75 S    R        Dayton  Montgomery    OH       45417
4  Miamisburg Centerville Rd    R        Dayton  Montgomery    OH       45459

  Country    Timezone Airport_Code    Weather_Timestamp  Temperature(F)  \
0      US  US/Eastern         KFFO  2016-02-08 05:58:00            36.9
1      US  US/Eastern         KCMH  2016-02-08 05:51:00            37.9
2      US  US/Eastern         KI69  2016-02-08 06:56:00            36.0
3      US  US/Eastern         KDAY  2016-02-08 07:38:00            35.1
4      US  US/Eastern         KMGY  2016-02-08 07:53:00            36.0

   Wind_Chill(F)  Humidity(%)  Pressure(in)  Visibility(mi) Wind_Direction  \
0            NaN         91.0         29.68            10.0           Calm
1            NaN        100.0         29.65            10.0           Calm
2           33.3        100.0         29.67            10.0             SW
3           31.0         96.0         29.64             9.0             SW
4           33.3         89.0         29.65             6.0             SW
```

```
   Wind_Speed(mph)  Precipitation(in) Weather_Condition  Amenity   Bump  \
0              NaN               0.02        Light Rain    False  False
1              NaN               0.00        Light Rain    False  False
2              3.5                NaN          Overcast    False  False
3              4.6                NaN     Mostly Cloudy    False  False
4              3.5                NaN     Mostly Cloudy    False  False

   Crossing  Give_Way  Junction  No_Exit  Railway  Roundabout  Station   Stop  \
0     False     False     False    False    False       False    False  False
1     False     False     False    False    False       False    False  False
2     False     False     False    False    False       False    False  False
3     False     False     False    False    False       False    False  False
4     False     False     False    False    False       False    False  False

   Traffic_Calming  Traffic_Signal  Turning_Loop Sunrise_Sunset  \
0            False           False         False          Night
1            False           False         False          Night
2            False            True         False          Night
3            False           False         False          Night
4            False            True         False            Day

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0          Night             Night                 Night
1          Night             Night                   Day
2          Night               Day                   Day
3            Day               Day                   Day
4            Day               Day                   Day
```

[5]:
```
# Muestra un resumen de los datos en accidents
accidents.describe()
```

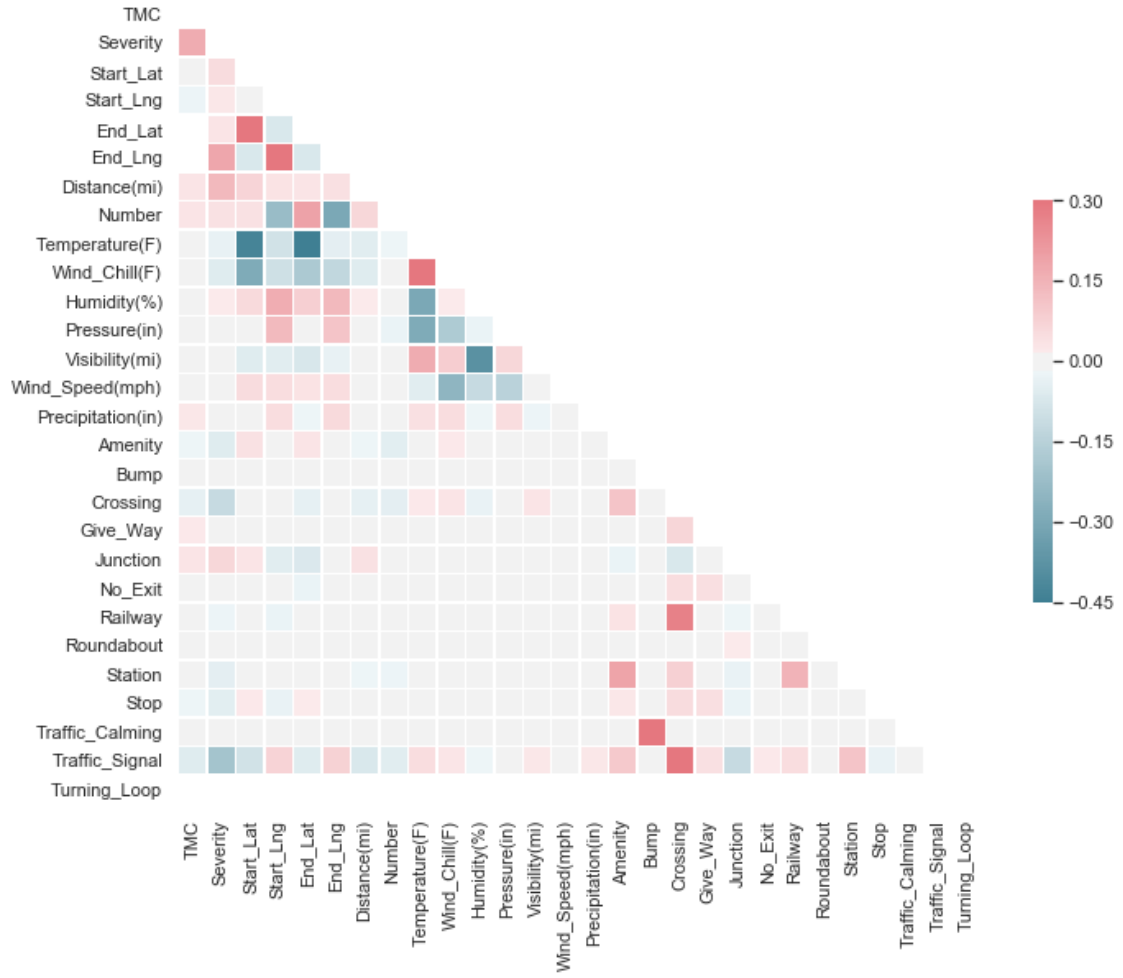
[5]:
```
                TMC      Severity     Start_Lat     Start_Lng        End_Lat  \
count  1.727177e+06  2.243939e+06  2.243939e+06  2.243939e+06  516762.000000
mean   2.073527e+02  2.382692e+00  3.646348e+01 -9.485567e+01      37.443109
std    1.940527e+01  5.488029e-01  4.958759e+00  1.709453e+01       5.126585
min    2.000000e+02  0.000000e+00  2.457022e+01 -1.246238e+02      24.570110
25%    2.010000e+02  2.000000e+00  3.348468e+01 -1.171362e+02      33.887450
50%    2.010000e+02  2.000000e+00  3.586428e+01 -8.818469e+01      38.038480
75%    2.010000e+02  3.000000e+00  4.042111e+01 -8.085453e+01      41.393320
max    4.060000e+02  4.000000e+00  4.900076e+01 -6.711317e+01      49.075000

             End_Lng  Distance(mi)         Number  Temperature(F)  \
count  516762.000000  2.243939e+06  785537.000000    2.181674e+06
mean      -96.527543  2.879095e-01    5625.281008    6.123244e+01
std        17.986406  1.532341e+00   11071.872897    1.914616e+01
min      -124.497829  0.000000e+00       1.000000   -7.780000e+01
25%      -117.870577  0.000000e+00     803.000000    4.890000e+01
```

```
50%       -90.192310  0.000000e+00    2672.000000    6.300000e+01
75%       -80.895040  1.000000e-02    6846.000000    7.590000e+01
max       -67.109242  3.336300e+02  961052.000000    1.706000e+02

       Wind_Chill(F)   Humidity(%)  Pressure(in)  Visibility(mi)  \
count  391569.000000  2.179472e+06  2.186659e+06    2.172579e+06
mean       26.042067  6.592758e+01  3.003747e+01    9.124096e+00
std        13.478333  2.243013e+01  2.267242e-01    2.986359e+00
min       -65.900000  4.000000e+00  0.000000e+00    0.000000e+00
25%        19.200000  5.000000e+01  2.992000e+01    1.000000e+01
50%        28.700000  6.800000e+01  3.003000e+01    1.000000e+01
75%        36.400000  8.500000e+01  3.015000e+01    1.000000e+01
max        45.200000  1.000000e+02  3.304000e+01    1.400000e+02

       Wind_Speed(mph)  Precipitation(in)
count     1.800985e+06      264473.000000
mean      8.844042e+00           0.060439
std       4.973200e+00           0.439698
min       1.200000e+00           0.000000
25%       5.800000e+00           0.000000
50%       8.100000e+00           0.010000
75%       1.150000e+01           0.040000
max       8.228000e+02          10.800000
```

Ahora vamos a visualizar la correlación entre las columnas, para ver cuales columnas tienen una alta correlación con cuales. Esto nos podría ayudar después a identificar los factores relevantes.

[6]:
```python
# Visualiza la correlación de las columnas
from string import ascii_letters

sns.set(style="white")

# Calcula la matriz de correlación
corr = accidents.corr()

# Genera una máscara para el triángulo superior
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Configura la figura de matplotlib
f, ax = plt.subplots(figsize=(11, 9))

# Genera un mapa de colores divergentes personalizado
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Dibuja el mapa de calor con la máscara y la relación de aspecto correcta
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
```

```
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



Podemos ver que hay una correlación entre: - la visibilidad y la humedad (cuando la humedad es alta, puede haber niebla) - La temperatura y la latitud (en el norte hace más frío) - La severidad y el canal de radio (cuando más severo un accident, más se habla de él) - etc. También podemos ver la correlación entre diferentes lugares dónde se producen los accidentes como semaforo y cruzar la calle. Lo que es lógico. No vamos a modelizar estos factores en este analisis, porque no nos podemos enfocar en todos los factores en el ámbito de esto documento.

[7]:
```
# Visualiza el tipo de cada columna
accidents.dtypes
```

[7]:
```
ID                     object
Source                 object
TMC                   float64
Severity                int64
Start_Time             object
```

```
End_Time                  object
Start_Lat                float64
Start_Lng                float64
End_Lat                  float64
End_Lng                  float64
Distance(mi)             float64
Description               object
Number                   float64
Street                    object
Side                      object
City                      object
County                    object
State                     object
Zipcode                   object
Country                   object
Timezone                  object
Airport_Code              object
Weather_Timestamp         object
Temperature(F)           float64
Wind_Chill(F)            float64
Humidity(%)              float64
Pressure(in)             float64
Visibility(mi)           float64
Wind_Direction            object
Wind_Speed(mph)          float64
Precipitation(in)        float64
Weather_Condition         object
Amenity                     bool
Bump                        bool
Crossing                    bool
Give_Way                    bool
Junction                    bool
No_Exit                     bool
Railway                     bool
Roundabout                  bool
Station                     bool
Stop                        bool
Traffic_Calming             bool
Traffic_Signal              bool
Turning_Loop                bool
Sunrise_Sunset            object
Civil_Twilight            object
Nautical_Twilight         object
Astronomical_Twilight     object
dtype: object
```

[8]:
```
# Visualiza la suma de los valores que faltan para cada columna
accidents.isnull().sum()
```

[8]:
```
ID                        0
Source                    0
TMC                  516762
Severity                  0
Start_Time                0
End_Time                  0
Start_Lat                 0
Start_Lng                 0
End_Lat             1727177
End_Lng             1727177
Distance(mi)              0
Description               1
Number              1458402
Street                    0
Side                      0
City                     68
County                    0
State                     0
Zipcode                 646
Country                   0
Timezone               2141
Airport_Code          23664
Weather_Timestamp     47170
Temperature(F)        62265
Wind_Chill(F)       1852370
Humidity(%)           64467
Pressure(in)          57280
Visibility(mi)        71360
Wind_Direction        47190
Wind_Speed(mph)      442954
Precipitation(in)   1979466
Weather_Condition     72004
Amenity                   0
Bump                      0
Crossing                  0
Give_Way                  0
Junction                  0
No_Exit                   0
Railway                   0
Roundabout                0
Station                   0
Stop                      0
Traffic_Calming           0
Traffic_Signal            0
```

```
Turning_Loop                  0
Sunrise_Sunset               78
Civil_Twilight               78
Nautical_Twilight            78
Astronomical_Twilight        78
dtype: int64
```

Selección de las columnas: Ahora vamos a seleccionar a las columnas que nos interesan para el análisis

[9]:
```
accidents_select =␣
↪accidents[['Start_Time','End_Time','State','Severity','Temperature(F)',
                    'Humidity(%)','Visibility(mi)',
                    'Wind_Direction','Weather_Condition','Civil_Twilight']].
↪copy()
```

A fin de poder juntas a los dos conjuntos de datos US_Traffic y US_Accidents. Vamos a crear en accidents_select nuevas columnas: Una columna con la fecha sin el año y una columna con solo la hora cuando el accidente ocurrió.

[10]:
```
# Corta la parte de Start_Time donde hay el mes y el día y crea una nueva␣
↪columna
accidents_select["Date_join"] = accidents_select["Start_Time"].apply(lambda d:␣
↪d[5:10])

# Corta la parte de Start_Time donde está escrito la hora y crea una nueva␣
↪columna
accidents_select["Hour_accident"] = accidents_select["Start_Time"].apply(lambda␣
↪d: d[11:13])
```

[11]:
```
# Visualiza a los cambios
accidents_select.head(2)
```

[11]:
```
            Start_Time             End_Time State  Severity  Temperature(F)  \
0  2016-02-08 05:46:00  2016-02-08 11:00:00    OH         3            36.9
1  2016-02-08 06:07:59  2016-02-08 06:37:59    OH         2            37.9

   Humidity(%)  Visibility(mi) Wind_Direction Weather_Condition  \
0         91.0            10.0           Calm        Light Rain
1        100.0            10.0           Calm        Light Rain

  Civil_Twilight Date_join Hour_accident
0          Night     02-08            05
1          Night     02-08            06
```

4.2 2.2. Conjunto de datos: US_Traffic

[12]:
```
# Importa el conjunto de datos US Traffic, 2015
traffic = pd.read_csv('dot_traffic_2015.txt', header=0, sep=',', quotechar='"')
# traffic = traffic[:200000].copy()
```

[13]:
```
# Calcula la cantidad de líneas y columnas
d = traffic.shape
print("US_Traffic tiene {} líneas y {} columnas.".format(d[0], d[1]))
```

```
US_Traffic tiene 7140391 líneas y 38 columnas.
```

[14]:
```
# Muestra las primeras líneas
traffic.head(3)
```

[14]:

	date	day_of_data	day_of_week	direction_of_travel	direction_of_travel_name	fips_state_code	functional_classification	functional_classification_name	lane_of_travel	month_of_data	record_type	restrictions	station_id	traffic_volume_counted_after_0000_to_0100	traffic_volume_counted_after_0100_to_0200	traffic_volume_counted_after_0200_to_0300
0	2015-04-07	7	3	1	North	56	3R	Rural: Principal Arterial - Other	1	4	3	NaN	000084	4	3	2
1	2015-09-26	26	7	7	West	21	1U	Urban: Principal Arterial - Interstate	2	9	3	NaN	056P94	381	252	218
2	2015-06-16	16	3	3	East	6	1U	Urban: Principal Arterial - Interstate	0	6	3	NaN	077590	585	408	

```
2                                        328

   traffic_volume_counted_after_0300_to_0400  \
0                                          4
1                                        194
2                                        364

   traffic_volume_counted_after_0400_to_0500  \
0                                         43
1                                        220
2                                        696

   traffic_volume_counted_after_0500_to_0600  \
0                                         78
1                                        348
2                                       1929

   traffic_volume_counted_after_0600_to_0700  \
0                                        116
1                                        453
2                                       4228

   traffic_volume_counted_after_0700_to_0800  \
0                                        144
1                                        679
2                                       5634

   traffic_volume_counted_after_0800_to_0900  \
0                                        132
1                                        826
2                                       5673

   traffic_volume_counted_after_0900_to_1000  \
0                                        115
1                                        962
2                                       4636

   traffic_volume_counted_after_1000_to_1100  \
0                                        150
1                                       1158
2                                       3925

   traffic_volume_counted_after_1100_to_1200  \
0                                        184
1                                       1379
2                                       3827
```

```
   traffic_volume_counted_after_1200_to_1300  \
0                                        169
1                                       1376
2                                       4049

   traffic_volume_counted_after_1300_to_1400  \
0                                        136
1                                       1383
2                                       3954

   traffic_volume_counted_after_1400_to_1500  \
0                                        129
1                                       1453
2                                       4077

   traffic_volume_counted_after_1500_to_1600  \
0                                         89
1                                       1617
2                                       4244

   traffic_volume_counted_after_1600_to_1700  \
0                                        122
1                                       1669
2                                       4405

   traffic_volume_counted_after_1700_to_1800  \
0                                        124
1                                       1308
2                                       4609

   traffic_volume_counted_after_1800_to_1900  \
0                                        110
1                                       1068
2                                       4361

   traffic_volume_counted_after_1900_to_2000  \
0                                         69
1                                        928
2                                       3272

   traffic_volume_counted_after_2000_to_2100  \
0                                         73
1                                        885
2                                       2243

   traffic_volume_counted_after_2100_to_2200  \
0                                         28
```

```
1                                        798
2                                       2050

   traffic_volume_counted_after_2200_to_2300  \
0                                         12
1                                        650
2                                       1453

   traffic_volume_counted_after_2300_to_2400  year_of_data
0                                          6            15
1                                        613            15
2                                        892            15
```

[15]:
```
# Visualiza un resumen de los valores de los datos en traffic
traffic.describe()
```

[15]:
```
        day_of_data   day_of_week  direction_of_travel  fips_state_code  \
count  7.140391e+06  7.140391e+06         7.140391e+06     7.140391e+06
mean   1.572650e+01  4.013872e+00         3.883486e+00     2.966967e+01
std    8.769343e+00  1.997511e+00         2.286816e+00     1.670612e+01
min    1.000000e+00  1.000000e+00         0.000000e+00     1.000000e+00
25%    8.000000e+00  2.000000e+00         1.000000e+00     1.300000e+01
50%    1.600000e+01  4.000000e+00         4.000000e+00     3.000000e+01
75%    2.300000e+01  6.000000e+00         5.000000e+00     4.400000e+01
max    3.100000e+01  7.000000e+00         9.000000e+00     5.600000e+01

       lane_of_travel  month_of_data  record_type  restrictions  \
count    7.140391e+06   7.140391e+06    7140391.0           0.0
mean     1.291672e+00   6.520263e+00          3.0           NaN
std      1.080419e+00   3.455234e+00          0.0           NaN
min      0.000000e+00   1.000000e+00          3.0           NaN
25%      1.000000e+00   4.000000e+00          3.0           NaN
50%      1.000000e+00   7.000000e+00          3.0           NaN
75%      2.000000e+00   1.000000e+01          3.0           NaN
max      9.000000e+00   1.200000e+01          3.0           NaN

       traffic_volume_counted_after_0000_to_0100  \
count                               7.140391e+06
mean                                1.145878e+02
std                                 2.818492e+02
min                                -1.000000e+00
25%                                 1.300000e+01
50%                                 4.200000e+01
75%                                 1.260000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_0100_to_0200  \
count                               7.140391e+06
```

```
mean                                 7.874558e+01
std                                  2.202875e+02
min                                 -1.000000e+00
25%                                  8.000000e+00
50%                                  2.700000e+01
75%                                  8.500000e+01
max                                  8.074100e+04

       traffic_volume_counted_after_0200_to_0300  \
count                                7.140391e+06
mean                                 6.622503e+01
std                                  2.102642e+02
min                                 -1.000000e+00
25%                                  7.000000e+00
50%                                  2.100000e+01
75%                                  7.000000e+01
max                                  9.001700e+04

       traffic_volume_counted_after_0300_to_0400  \
count                                7.140391e+06
mean                                 7.016138e+01
std                                  2.242483e+02
min                                 -1.000000e+00
25%                                  7.000000e+00
50%                                  2.300000e+01
75%                                  7.500000e+01
max                                  9.001200e+04

       traffic_volume_counted_after_0400_to_0500  \
count                                7.140391e+06
mean                                 1.171851e+02
std                                  3.227085e+02
min                                 -1.000000e+00
25%                                  1.200000e+01
50%                                  3.800000e+01
75%                                  1.180000e+02
max                                  7.056000e+04

       traffic_volume_counted_after_0500_to_0600  \
count                                7.140391e+06
mean                                 2.454065e+02
std                                  5.723301e+02
min                                 -1.000000e+00
25%                                  2.800000e+01
50%                                  8.600000e+01
75%                                  2.410000e+02
max                                  7.815900e+04
```

```
       traffic_volume_counted_after_0600_to_0700  \
count                               7.140391e+06
mean                                4.334301e+02
std                                 8.359078e+02
min                                -1.000000e+00
25%                                 5.600000e+01
50%                                 1.700000e+02
75%                                 4.570000e+02
max                                 9.002000e+04

       traffic_volume_counted_after_0700_to_0800  \
count                               7.140391e+06
mean                                5.833799e+02
std                                 9.984941e+02
min                                -1.000000e+00
25%                                 9.000000e+01
50%                                 2.640000e+02
75%                                 6.570000e+02
max                                 9.018700e+04

       traffic_volume_counted_after_0800_to_0900  \
count                               7.140391e+06
mean                                5.774976e+02
std                                 9.594217e+02
min                                -1.000000e+00
25%                                 1.070000e+02
50%                                 2.850000e+02
75%                                 6.570000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_0900_to_1000  \
count                               7.140391e+06
mean                                5.600694e+02
std                                 8.917308e+02
min                                -1.000000e+00
25%                                 1.230000e+02
50%                                 3.030000e+02
75%                                 6.450000e+02
max                                 9.530000e+04

       traffic_volume_counted_after_1000_to_1100  \
count                               7.140391e+06
mean                                5.811083e+02
std                                 8.970572e+02
min                                -1.000000e+00
25%                                 1.370000e+02
```

```
50%                                 3.320000e+02
75%                                 6.810000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_1100_to_1200  \
count                               7.140391e+06
mean                                6.184133e+02
std                                 9.371717e+02
min                                -1.000000e+00
25%                                 1.490000e+02
50%                                 3.620000e+02
75%                                 7.250000e+02
max                                 9.020000e+04

       traffic_volume_counted_after_1200_to_1300  \
count                               7.140391e+06
mean                                6.501346e+02
std                                 1.001836e+03
min                                -3.061000e+03
25%                                 1.590000e+02
50%                                 3.860000e+02
75%                                 7.610000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_1300_to_1400  \
count                               7.140391e+06
mean                                6.635075e+02
std                                 1.024515e+03
min                                -1.000000e+00
25%                                 1.620000e+02
50%                                 3.910000e+02
75%                                 7.770000e+02
max                                 9.420000e+04

       traffic_volume_counted_after_1400_to_1500  \
count                               7.140391e+06
mean                                7.009825e+02
std                                 1.092236e+03
min                                -1.000000e+00
25%                                 1.700000e+02
50%                                 4.090000e+02
75%                                 8.220000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_1500_to_1600  \
count                               7.140391e+06
mean                                7.497160e+02
```

```
std                                  1.143318e+03
min                                 -1.000000e+00
25%                                  1.830000e+02
50%                                  4.380000e+02
75%                                  8.910000e+02
max                                  9.999900e+04

       traffic_volume_counted_after_1600_to_1700  \
count                                7.140391e+06
mean                                 7.770437e+02
std                                  1.173933e+03
min                                 -1.000000e+00
25%                                  1.860000e+02
50%                                  4.520000e+02
75%                                  9.340000e+02
max                                  9.999900e+04

       traffic_volume_counted_after_1700_to_1800  \
count                                7.140391e+06
mean                                 7.565536e+02
std                                  1.172116e+03
min                                 -1.000000e+00
25%                                  1.740000e+02
50%                                  4.320000e+02
75%                                  9.070000e+02
max                                  9.999900e+04

       traffic_volume_counted_after_1800_to_1900  \
count                                7.140391e+06
mean                                 6.173322e+02
std                                  1.061545e+03
min                                 -1.000000e+00
25%                                  1.310000e+02
50%                                  3.370000e+02
75%                                  7.220000e+02
max                                  9.999900e+04

       traffic_volume_counted_after_1900_to_2000  \
count                                7.140391e+06
mean                                 4.793756e+02
std                                  9.203711e+02
min                                 -1.000000e+00
25%                                  9.500000e+01
50%                                  2.520000e+02
75%                                  5.510000e+02
max                                  9.999900e+04
```

```
       traffic_volume_counted_after_2000_to_2100  \
count                               7.140391e+06
mean                                3.906426e+02
std                                 8.290271e+02
min                                -1.000000e+00
25%                                 7.200000e+01
50%                                 1.980000e+02
75%                                 4.470000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_2100_to_2200  \
count                               7.140391e+06
mean                                3.274747e+02
std                                 7.989146e+02
min                                -1.000000e+00
25%                                 5.400000e+01
50%                                 1.550000e+02
75%                                 3.670000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_2200_to_2300  \
count                               7.140391e+06
mean                                2.534447e+02
std                                 7.284074e+02
min                                -1.000000e+00
25%                                 3.600000e+01
50%                                 1.090000e+02
75%                                 2.780000e+02
max                                 9.999900e+04

       traffic_volume_counted_after_2300_to_2400  year_of_data
count                               7.140391e+06     7140391.0
mean                                1.798298e+02          15.0
std                                 6.901713e+02           0.0
min                                -1.000000e+00          15.0
25%                                 2.200000e+01          15.0
50%                                 7.000000e+01          15.0
75%                                 1.930000e+02          15.0
max                                 9.999900e+04          15.0
```

[16]:
```
# Visualiza el tipo de cada columna
traffic.dtypes
```

[16]:
```
date                                 object
day_of_data                           int64
day_of_week                           int64
direction_of_travel                   int64
direction_of_travel_name             object
```

```
fips_state_code                                int64
functional_classification                     object
functional_classification_name                object
lane_of_travel                                 int64
month_of_data                                  int64
record_type                                    int64
restrictions                                 float64
station_id                                    object
traffic_volume_counted_after_0000_to_0100      int64
traffic_volume_counted_after_0100_to_0200      int64
traffic_volume_counted_after_0200_to_0300      int64
traffic_volume_counted_after_0300_to_0400      int64
traffic_volume_counted_after_0400_to_0500      int64
traffic_volume_counted_after_0500_to_0600      int64
traffic_volume_counted_after_0600_to_0700      int64
traffic_volume_counted_after_0700_to_0800      int64
traffic_volume_counted_after_0800_to_0900      int64
traffic_volume_counted_after_0900_to_1000      int64
traffic_volume_counted_after_1000_to_1100      int64
traffic_volume_counted_after_1100_to_1200      int64
traffic_volume_counted_after_1200_to_1300      int64
traffic_volume_counted_after_1300_to_1400      int64
traffic_volume_counted_after_1400_to_1500      int64
traffic_volume_counted_after_1500_to_1600      int64
traffic_volume_counted_after_1600_to_1700      int64
traffic_volume_counted_after_1700_to_1800      int64
traffic_volume_counted_after_1800_to_1900      int64
traffic_volume_counted_after_1900_to_2000      int64
traffic_volume_counted_after_2000_to_2100      int64
traffic_volume_counted_after_2100_to_2200      int64
traffic_volume_counted_after_2200_to_2300      int64
traffic_volume_counted_after_2300_to_2400      int64
year_of_data                                   int64
dtype: object
```

[17]:
```
# Visualiza la suma de los valores que faltan para cada columna
traffic.isnull().sum()
```

[17]:
```
date                                           0
day_of_data                                    0
day_of_week                                    0
direction_of_travel                            0
direction_of_travel_name                       0
fips_state_code                                0
functional_classification                      0
functional_classification_name                 0
lane_of_travel                                 0
month_of_data                                  0
```

```
record_type                                    0
restrictions                             7140391
station_id                                     0
traffic_volume_counted_after_0000_to_0100      0
traffic_volume_counted_after_0100_to_0200      0
traffic_volume_counted_after_0200_to_0300      0
traffic_volume_counted_after_0300_to_0400      0
traffic_volume_counted_after_0400_to_0500      0
traffic_volume_counted_after_0500_to_0600      0
traffic_volume_counted_after_0600_to_0700      0
traffic_volume_counted_after_0700_to_0800      0
traffic_volume_counted_after_0800_to_0900      0
traffic_volume_counted_after_0900_to_1000      0
traffic_volume_counted_after_1000_to_1100      0
traffic_volume_counted_after_1100_to_1200      0
traffic_volume_counted_after_1200_to_1300      0
traffic_volume_counted_after_1300_to_1400      0
traffic_volume_counted_after_1400_to_1500      0
traffic_volume_counted_after_1500_to_1600      0
traffic_volume_counted_after_1600_to_1700      0
traffic_volume_counted_after_1700_to_1800      0
traffic_volume_counted_after_1800_to_1900      0
traffic_volume_counted_after_1900_to_2000      0
traffic_volume_counted_after_2000_to_2100      0
traffic_volume_counted_after_2100_to_2200      0
traffic_volume_counted_after_2200_to_2300      0
traffic_volume_counted_after_2300_to_2400      0
year_of_data                                   0
dtype: int64
```

Faltan solamente valores sobre las restricciones. Selección de las columnas: Ahora vamos a seleccionar a las columnas que nos interesan para el análisis. En este caso es la cantidad de tráfico según la fecha, la hora y el estado.

[18]:
```
traffic_select = traffic[['date','fips_state_code',
       'traffic_volume_counted_after_0000_to_0100',
       'traffic_volume_counted_after_0100_to_0200',
       'traffic_volume_counted_after_0200_to_0300',
       'traffic_volume_counted_after_0300_to_0400',
       'traffic_volume_counted_after_0400_to_0500',
       'traffic_volume_counted_after_0500_to_0600',
       'traffic_volume_counted_after_0600_to_0700',
       'traffic_volume_counted_after_0700_to_0800',
       'traffic_volume_counted_after_0800_to_0900',
       'traffic_volume_counted_after_0900_to_1000',
       'traffic_volume_counted_after_1000_to_1100',
       'traffic_volume_counted_after_1100_to_1200',
       'traffic_volume_counted_after_1200_to_1300',
```

```
       'traffic_volume_counted_after_1300_to_1400',
       'traffic_volume_counted_after_1400_to_1500',
       'traffic_volume_counted_after_1500_to_1600',
       'traffic_volume_counted_after_1600_to_1700',
       'traffic_volume_counted_after_1700_to_1800',
       'traffic_volume_counted_after_1800_to_1900',
       'traffic_volume_counted_after_1900_to_2000',
       'traffic_volume_counted_after_2000_to_2100',
       'traffic_volume_counted_after_2100_to_2200',
       'traffic_volume_counted_after_2200_to_2300',
       'traffic_volume_counted_after_2300_to_2400']]
```

[19]:
```
# Renombra a las columnas
traffic_select.columns =␣
 ↪['Date','State','0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16',
```

4.3 2.3. Junta US_Accidents e US_Traffic

Ahora vamos a juntar a los dos conjuntos de datos. US_Traffic cuenta el volumen de tráfico en muchos lugares diferentes. Una solución para obtener un nivel general de tráfico por estado es calcular la media del tráfico por día y por estado.

[20]:
```
# Calcula la media del volumen de tráfico por estado y por fecha
traffic_select = traffic_select.groupby(['Date','State']).mean().reset_index()
```

US_Accidents utiliza la abreviatura de cada estado para identificarlo y US_Traffic utiliza el código fips para identificarlos. Por eso vamos a convertir los códigos fips en abreviaturas utilizando la lista de conversión de wikipedia.

[21]:
```
# Carga los datos de conversión
states_mapper = pd.read_csv("US_States.txt")
states_mapper = states_mapper.set_index("nb")

# Añade una nueva columna con abreviaturas
traffic_select["State_letters"] = traffic_select["State"].apply(lambda d:␣
 ↪states_mapper.loc[d] )

# Añade una nueva columna date_join para juntar a los dos conjuntos␣
 ↪utilizándola
traffic_select["Date_join"] = traffic_select["Date"].apply(lambda d: d[5:])
```

Traffic contiene una columna para el volumen de tráfico para cada hora. Vamos a cambiar su formato para que tenga una columna "Hour" con la hora.

[22]:
```
# Crea una línea para cada volumen de tráfico según la hora
traffic_transf = pd.DataFrame()
hours = [str(d) for d in range(24)]
for h in hours:
    df_1 = traffic_select[['Date', 'State', h, 'State_letters', 'Date_join']].
 ↪copy()
```

```
    df_1["Hour"] = h
    df_1 = df_1.rename(columns = {h: "Traffic_quantity"})
    traffic_transf = traffic_transf.append(df_1)

def add_zero(x):
# Añade un "0" si hay un solo número
    x = str(x)
    if len(x) == 1:
        x = "0" + x
    return x

# Añade un "0" si la hora está escrito con un solo número, por ejemplo: "9" =>␣
 ↪"09"
# para tener el mismo formato que US_Accidents
traffic_transf["Hour"] = traffic_transf["Hour"].apply(add_zero)
```

[23]:
```
# Junta a US_Accidents y US_Traffic
accidents_t = pd.merge(accidents_select, traffic_transf,how='left',
                       left_on=["State","Date_join","Hour_accident"],
                       right_on = ["State_letters","Date_join","Hour"])
```

[24]:
```
# Cambio para tener Date en el format YYYY-MM-DD
accidents_t["Date"] = accidents_t["Start_Time"].apply(lambda d: d[:10])
```

[25]:
```
# Visualiza las columnas
accidents_t.columns
```

[25]:
```
Index(['Start_Time', 'End_Time', 'State_x', 'Severity', 'Temperature(F)',
       'Humidity(%)', 'Visibility(mi)', 'Wind_Direction', 'Weather_Condition',
       'Civil_Twilight', 'Date_join', 'Hour_accident', 'Date', 'State_y',
       'Traffic_quantity', 'State_letters', 'Hour'],
      dtype='object')
```

[26]:
```
# Selecciona a las columnas con relevancia en este caso
accidents_t = accidents_t[['State_x', 'Date', 'Hour_accident',␣
 ↪'Severity','Traffic_quantity',
                           'Temperature(F)','Humidity(%)',
                           'Visibility(mi)','Weather_Condition']]
accidents_t = accidents_t.rename(columns = {"State_x":"State"})
```

[27]:
```
# Visualiza los datos
accidents_t.head(5)
```

[27]:

	State	Date	Hour_accident	Severity	Traffic_quantity	Temperature(F)	\
0	OH	2016-02-08	05	3	54.560189	36.9	
1	OH	2016-02-08	06	2	97.465775	37.9	
2	OH	2016-02-08	06	2	97.465775	36.0	
3	OH	2016-02-08	07	3	119.698662	35.1	
4	OH	2016-02-08	07	2	119.698662	36.0	

```
   Humidity(%)  Visibility(mi) Weather_Condition
0         91.0            10.0        Light Rain
1        100.0            10.0        Light Rain
2        100.0            10.0          Overcast
3         96.0             9.0     Mostly Cloudy
4         89.0             6.0     Mostly Cloudy
```

5 3. Limpieza de datos

5.1 3.1. Datos que contienen ceros o elementos vacíos

[28]:
```
# Visualiza los datos que faltan
accidents_t.isnull().sum()
```

[28]:
```
State                   0
Date                    0
Hour_accident           0
Severity                0
Traffic_quantity    17923
Temperature(F)      62265
Humidity(%)         64467
Visibility(mi)      71360
Weather_Condition   72004
dtype: int64
```

[29]:
```
# Dimensión del nuevo conjunto de datos
accidents_t.shape
```

[29]:
```
(2243939, 9)
```

Podemos ver que faltan datos, sobre todos, sobre la condiciones meteorológicas durante los accidentes. La cantidad de datos que falta no es grande en comparación con las número de líneas del conjunto de datos. Por eso vamos a suprimir a las líneas con datos faltantes (a parte de las tres nombradas antes).

[30]:
```
# Lista de las columnas en las cuales no queremos valores faltantes.
columns_filter = ['State', 'Date', 'Hour_accident', 'Severity',
                  'Traffic_quantity','Temperature(F)', 'Humidity(%)',
                  'Visibility(mi)','Weather_Condition']

# Suprime las líneas con valores faltantes en las columnas de "columns_filter"
for c in columns_filter:
    accidents_t = accidents_t[accidents_t[c].notnull()]
```

[31]:
```
accidents_t.shape
```

[31]:
```
(2140055, 9)
```

Hemos perdido a poco datos.

[32]:
```
# Visualiza los datos que faltan
accidents_t.isnull().sum()
```

[32]:
```
State                0
Date                 0
Hour_accident        0
Severity             0
Traffic_quantity     0
Temperature(F)       0
Humidity(%)          0
Visibility(mi)       0
Weather_Condition    0
dtype: int64
```

[33]:
```
# Visualiza los tipos de cada columna
accidents_t.dtypes
```

[33]:
```
State                 object
Date                  object
Hour_accident         object
Severity               int64
Traffic_quantity     float64
Temperature(F)       float64
Humidity(%)          float64
Visibility(mi)       float64
Weather_Condition     object
dtype: object
```

[34]:
```
# Convierte el formato de las columnas Date, Hour_accident and State
accidents_t["Date"] = pd.to_datetime(accidents_t["Date"])
accidents_t["Hour_accident"] = accidents_t["Hour_accident"].astype('int32')
accidents_t["State"] = accidents_t["State"].astype('category')
```

[35]:
```
# Visualiza los tipos de cada columna
accidents_t.dtypes
```

[35]:
```
State                      category
Date                 datetime64[ns]
Hour_accident                 int32
Severity                      int64
Traffic_quantity            float64
Temperature(F)              float64
Humidity(%)                 float64
Visibility(mi)              float64
Weather_Condition            object
dtype: object
```

[36]:
```
# Calcula el número de valores 0 en cada columna
(accidents_t == 0).sum(axis = 0)
```

[36]:
```
State                   0
Date                    0
Hour_accident       15845
Severity               17
Traffic_quantity       15
Temperature(F)        554
Humidity(%)             0
Visibility(mi)        168
Weather_Condition       0
dtype: int64
```

Podemos ver que las columnas que contienen ceros, es porque tiene sentido que las contengan a parte de Severity. En la información sobre los atributos, está escrito que Severity tiene valores entre 1 y 4. Por eso vamos a suprimir las líneas con Severity = 0.

[37]:
```
accidents_t = accidents_t[accidents_t["Severity"] != 0]
```

5.2 3.2. Identificación y tratamiento de valores extremos.

A fin de verificar si hay valores extremos que pueden provenir de un error, vamos a visualizar los valores máximos y mínimos para cada columna en un formato que no sea String o Category.

[38]:
```
cols = ['Date', 'Hour_accident', 'Severity', 'Traffic_quantity',
       'Temperature(F)', 'Humidity(%)',
       'Visibility(mi)']
accidents_t[cols].min()
```

[38]:
```
Date                2016-02-08 00:00:00
Hour_accident                         0
Severity                              1
Traffic_quantity                      0
Temperature(F)                      -29
Humidity(%)                           4
Visibility(mi)                        0
dtype: object
```

Todos los valores mínimos tienen sentido.

[39]:
```
accidents_t[cols].max()
```

[39]:
```
Date                2019-03-31 00:00:00
Hour_accident                        23
Severity                              4
Traffic_quantity                4357.64
Temperature(F)                    170.6
Humidity(%)                         100
Visibility(mi)                      140
dtype: object
```

Aquí podemos ver que la temperatura máxima (170řF = 77řC) y la velocidad máxima del viento (822 mph = 1322 kmh) son demasiadas altas.

[40]:
```
# Visualiza los valores con más exactitud
accidents_t.describe()
```

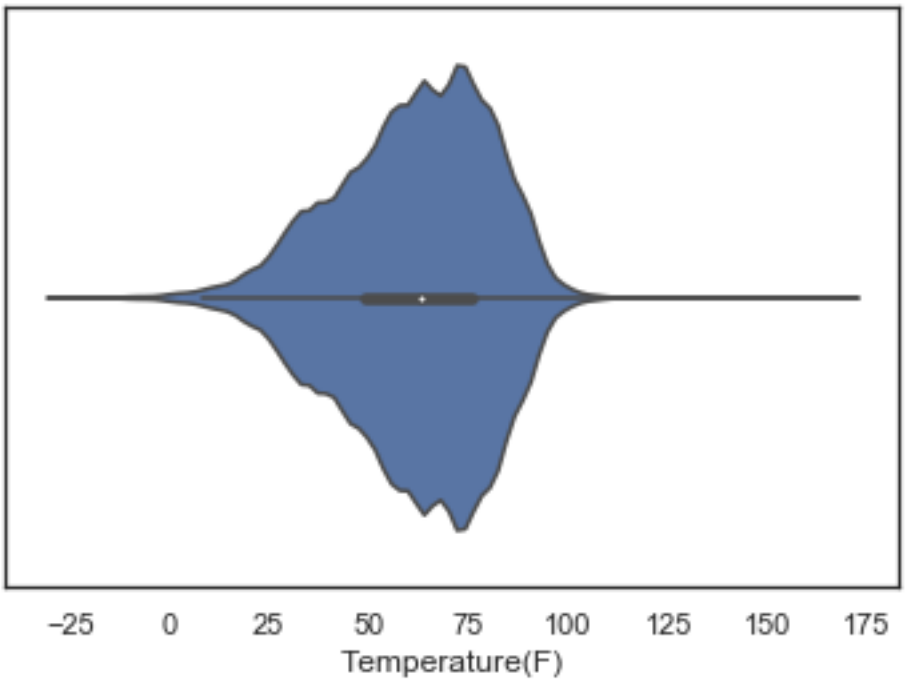
[40]:

	Hour_accident	Severity	Traffic_quantity	Temperature(F)	\
count	2.140038e+06	2.140038e+06	2.140038e+06	2.140038e+06	
mean	1.215630e+01	2.379723e+00	9.592071e+02	6.128366e+01	
std	5.074900e+00	5.461502e-01	7.285032e+02	1.913394e+01	
min	0.000000e+00	1.000000e+00	0.000000e+00	-2.900000e+01	
25%	8.000000e+00	2.000000e+00	4.212140e+02	4.890000e+01	
50%	1.200000e+01	2.000000e+00	6.845393e+02	6.300000e+01	
75%	1.600000e+01	3.000000e+00	1.354669e+03	7.590000e+01	
max	2.300000e+01	4.000000e+00	4.357643e+03	1.706000e+02	

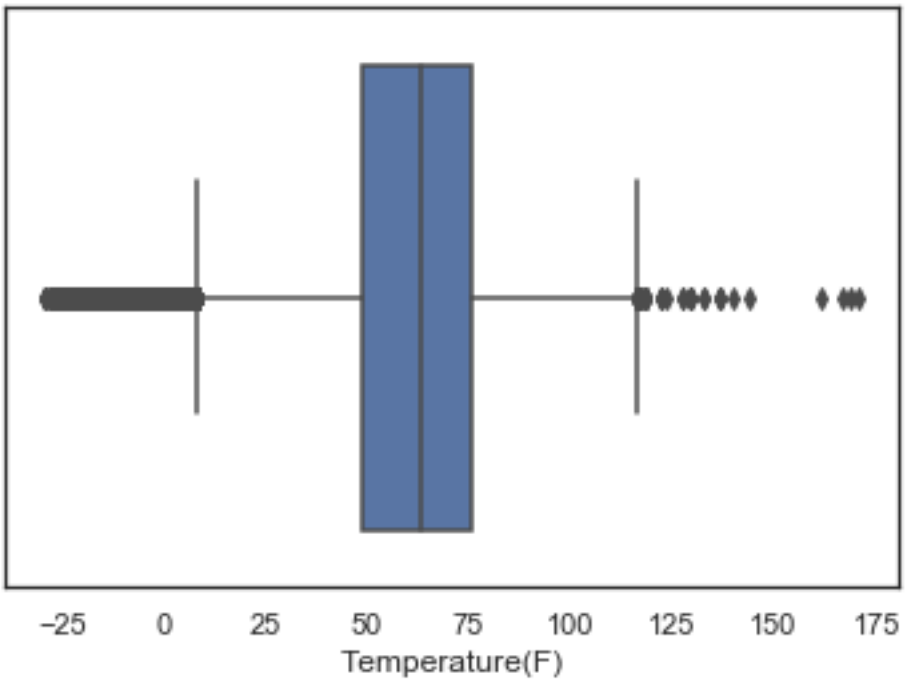
	Humidity(%)	Visibility(mi)
count	2.140038e+06	2.140038e+06
mean	6.586599e+01	9.122871e+00
std	2.244314e+01	2.967502e+00
min	4.000000e+00	0.000000e+00
25%	5.000000e+01	1.000000e+01
50%	6.800000e+01	1.000000e+01
75%	8.500000e+01	1.000000e+01
max	1.000000e+02	1.400000e+02

Dibuja dos representaciones para mostrar las distribuciones de la temperatura:

[41]:
```
sns.violinplot(x=accidents_t["Temperature(F)"])
plt.show()
```



[42]:
```
sns.boxplot(x=accidents_t["Temperature(F)"])
plt.show()
```



Según Wikipedia la mayor temperatura medida en los EE. UU. es de 134 řF / 57 řC. Por eso vamos a suprimir las líneas con una temperatura mayor de 134 řF.

[43]:
```
p = accidents_t[accidents_t["Temperature(F)"] > 134].shape[0]
print("Cantidad de líneas con una temperatura superior a 134 řF: {}".format(p))
```

```
Cantidad de líneas con una temperatura superior a 134 řF: 9
```

[44]:
```
# Elimina a las líneas con una temperatura superior a 134 řF
accidents_t = accidents_t[accidents_t["Temperature(F)"] <= 134]
```

6 4. Análisis de los datos & representación de los resultados

6.1 4.1. Comprobación de la normalidad y homogeneidad de la varianza.

6.1.1 4.1.1. Normalidad

A fin de comprobar si los valores de nuestras variables cuantitativas provienen de una población distribuida normalmente, vamos a utilizar la prueba de normalidad de Anderson. Comprobaremos con un nivel de confianza de 95% si las variables siguen una distribución normal.

[45]:
```
# Importa scipy
import scipy.stats

# Lista de las columnas del conjunto de datos en formato de número
cols_nb = ['Hour_accident', 'Traffic_quantity',
       'Temperature(F)', 'Humidity(%)', 'Visibility(mi)']

# Prueba para cada columna de la lista si sus datos son similares a una␣
 ↪distribución normal
print("£Con 95.0% de confianza, los datos de las siguientes columnas son␣
 ↪similares a una distribución normal según la prueba de Anderson?\n")
for c in cols_nb:
    ad_stat, ad_critic, ad_theor =  scipy.stats.anderson(accidents_t[c])
    if ad_stat < ad_critic[2]:
        print(c + ": SI")
    else:
        print(c + ": NO")
```

```
£Con 95.0% de confianza, los datos de las siguientes columnas son similares a
una distribución normal según la prueba de Anderson?

Hour_accident: NO
Traffic_quantity: NO
Temperature(F): NO
Humidity(%): NO
Visibility(mi): NO
```

Podemos ver que ningún variable sigue una distribución normal con un nivel de confianza de 95%.

6.1.2 4.1.2. Homogeneidad de la varianza

Cantidad de tráfico según la hora del accidente:

En seguida, vamos a estudiar la homogeneidad de varianzas con el test de Fligner-Killeen. Aquí vamos a estudiar la homogeneidad en función de la cantidad de tráfico según la hora. En esta prueba, la hipótesis nula consiste en que las varianzas son iguales.

[46]:
```
# Test de homogeneidad
# Hardcoding, porque scipy.stats.fligner no accepta un lista de listas []
acc_hours = []
for h in range(24):
    acc_hours.append(accidents_t[accidents_t["Hour_accident"] ==␣
 ↪h]["Traffic_quantity"].values)

h0,h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13,h14,h15,h16,h17,h18,h19,h20,h21,h22,h23␣
 ↪= acc_hours
```

```
statistic, pvalue = scipy.stats.
 ↪fligner(h0,h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13,h14,h15,h16,h17,h18,h19,h20,h21,h22,h

if pvalue > 0.05:
    print("El p-valor es {} y es superior a 0.05, aceptamos la hipótesis de que␣
 ↪las varianzas son homogéneas con una confidencia de 95%.".format(pvalue))
else:
    print("El p-valor es {} y es inferior a 0.05, NO aceptamos la hipótesis de␣
 ↪que las varianzas son homogéneas con una confidencia de 95%.".format(pvalue))
```

```
El p-valor es 0.0 y es inferior a 0.05, NO aceptamos la hipótesis de que las
varianzas son homogéneas con una confidencia de 95%.
```

Analisis de la homogeneidad de la varianza en el resto de variables numéricas, teniendo en cuenta los grupos definidos por severity: - Hour_accident - Traffic_quantity - Temperature(F) - Humidity(%) - Visibility(mi)

[47]:
```
# Función para imprimir el texto en graso en Jupyter.
from IPython.display import Markdown, display
def printmarkdown(string):
    display(Markdown(string))
```

[48]:
```
# Selección de las columnas
cols =␣
 ↪['Hour_accident','Traffic_quantity','Temperature(F)','Humidity(%)','Visibility(mi)']

# Loop para reiterar el proceso con cada columna de las columnas en cols.
for col in cols:
    # Separación de los datos por nivel de severidad
    acc_s = []
    for s in [1,2,3,4]:
        acc_s.append(accidents_t[accidents_t["Severity"] == s][col].values)
    s1,s2,s3,s4 = acc_s

    # Hace el test de Fligner-Killeen
    statistic, pvalue = scipy.stats.fligner(s1,s2,s3,s4)

    # Resultado
    printmarkdown("<b>Analisis de la homogeneidad de la varianza en '{}',␣
 ↪teniendo en cuenta los grupos definidos por severity:</b>".format(col))
    if pvalue > 0.05:
        print("El p-valor es {} y es superior a 0.05.\nAceptamos la hipótesis␣
 ↪de que las varianzas son homogéneas con una confianza de 95%.".
 ↪format(pvalue))
    else:
        print("El p-valor es {} y es inferior a 0.05.\nNO aceptamos la␣
 ↪hipótesis de que las varianzas son homogéneas con una confianza de 95%.".
 ↪format(pvalue))
```

Analisis de la homogeneidad de la varianza en 'Hour_accident', teniendo en cuenta los grupos definidos por severity:

```
El p-valor es 0.0 y es inferior a 0.05.
NO aceptamos la hipótesis de que las varianzas son homogéneas con una confianza
de 95%.
```

Analisis de la homogeneidad de la varianza en 'Traffic_quantity', teniendo en cuenta los grupos definidos por severity:

```
El p-valor es 0.0 y es inferior a 0.05.
NO aceptamos la hipótesis de que las varianzas son homogéneas con una confianza
de 95%.
```

Analisis de la homogeneidad de la varianza en 'Temperature(F)', teniendo en cuenta los grupos definidos por severity:

```
El p-valor es 3.5785081850205476e-287 y es inferior a 0.05.
NO aceptamos la hipótesis de que las varianzas son homogéneas con una confianza
de 95%.
```

Analisis de la homogeneidad de la varianza en 'Humidity(%)', teniendo en cuenta los grupos definidos por severity:

```
El p-valor es 1.1198287722953388e-132 y es inferior a 0.05.
NO aceptamos la hipótesis de que las varianzas son homogéneas con una confianza
de 95%.
```

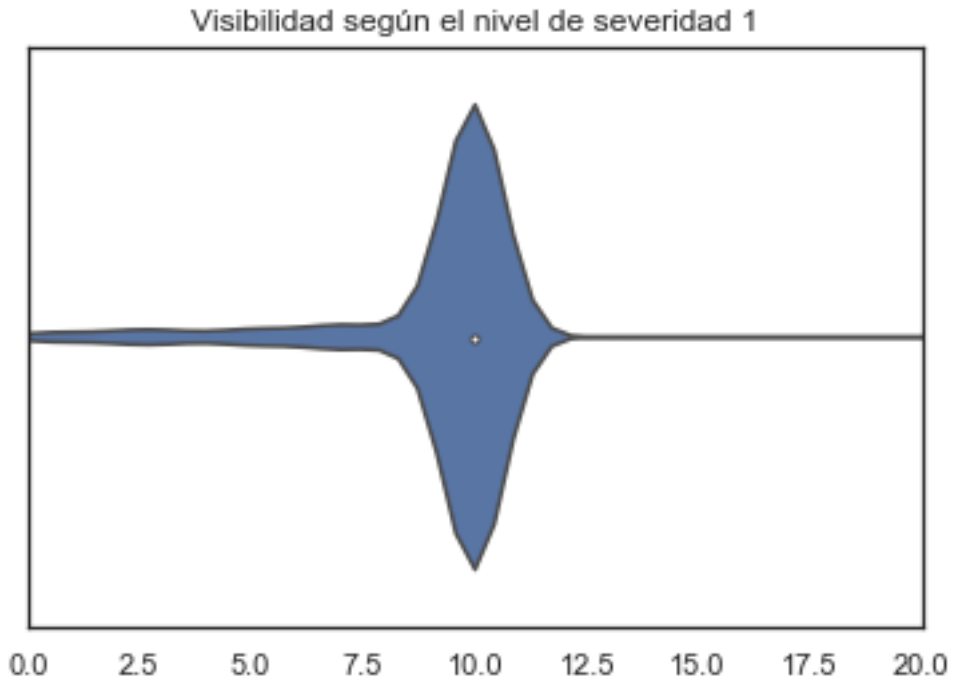
Analisis de la homogeneidad de la varianza en 'Visibility(mi)', teniendo en cuenta los grupos definidos por severity:

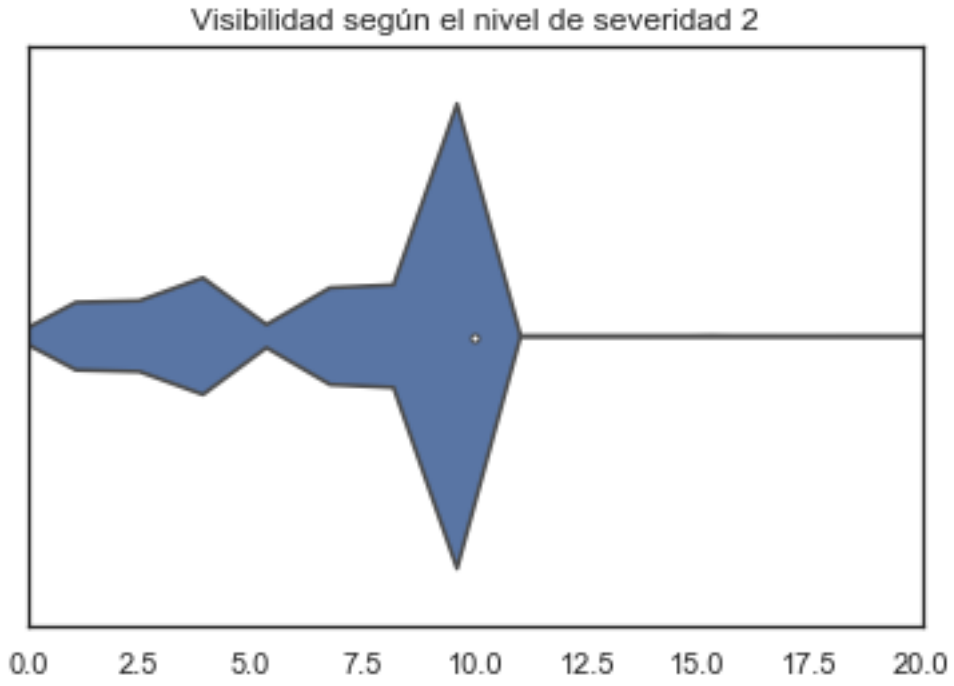
```
El p-valor es 1.266013315267389e-140 y es inferior a 0.05.
NO aceptamos la hipótesis de que las varianzas son homogéneas con una confianza
de 95%.
```

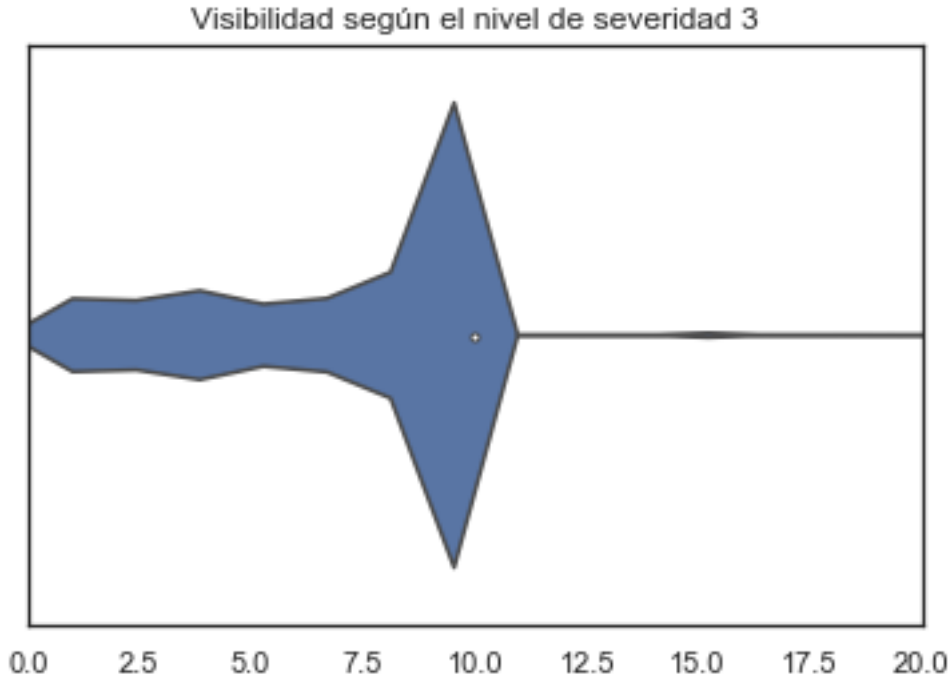
Podemos ver que no hay homogeneidad de la varianza para ningún de los variables teniendo en cuenta los grupos definidos por severidad. Vamos a visualizar "Traffic_quantity" para cada nivel de severidad enseguida:

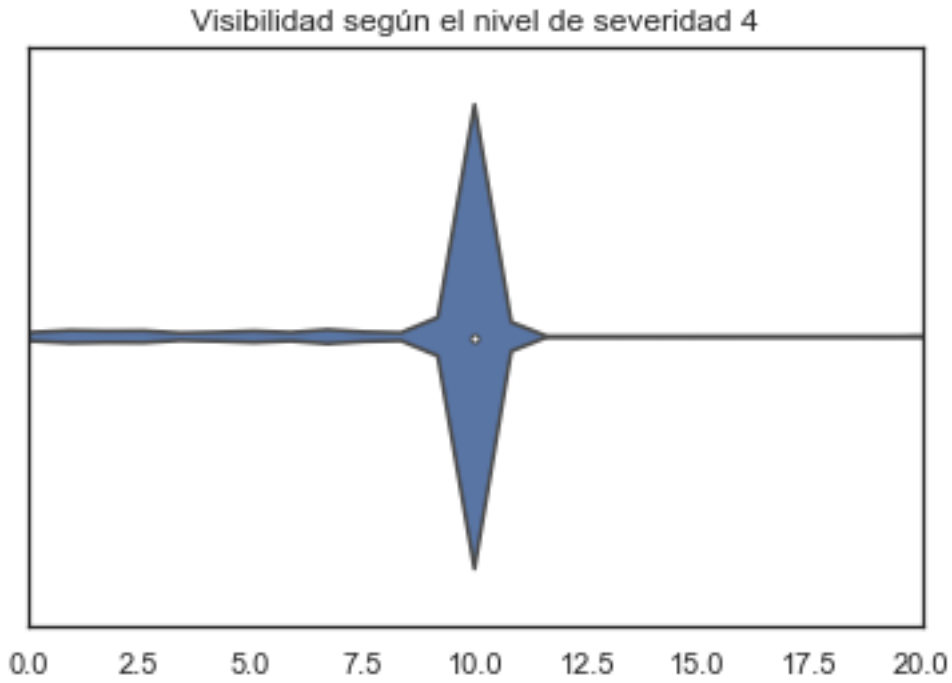
[49]:
```
# Separación de los datos por nivel de severidad
acc_s = []
for s in [1,2,3,4]:
    acc_s.append(accidents_t[accidents_t["Severity"] == s]["Visibility(mi)"].
 ↪values)

# Creación de un violinplot para cada nivel de severidad
s_lvl = 1
for s in acc_s:
    sns.violinplot(s)
    plt.xlim(0,20)
```

```
    plt.title("Visibilidad según el nivel de severidad {}".format(s_lvl))
    plt.show()
    s_lvl += 1
```






Visibilidad según el nivel de severidad 3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

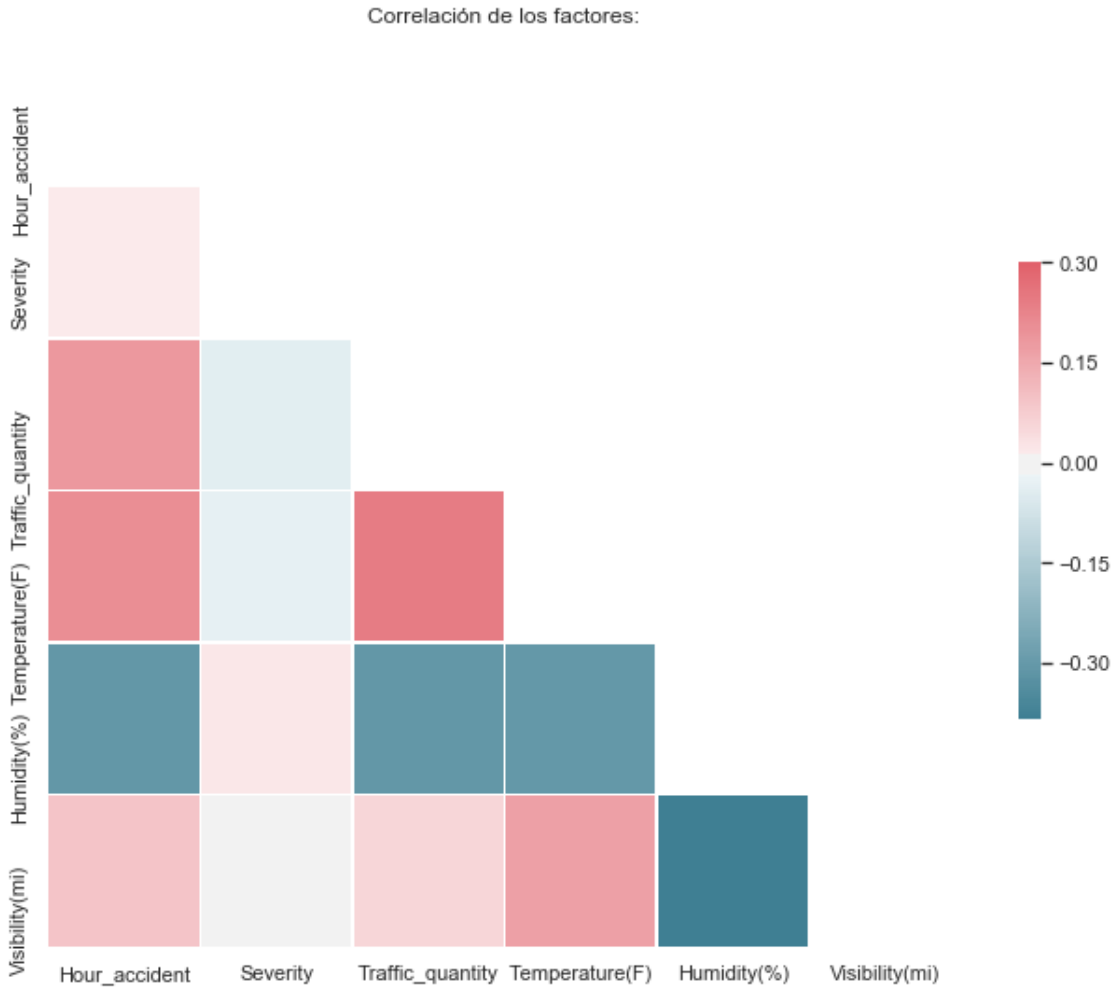

Visibilidad según el nivel de severidad 4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Podemos también ver que hay bastante diferencia entre los violinplots.

6.2 4.2. Aplicación de pruebas estadísticas para comparar los grupos de datos.

Primero vamos a visualizar la correlación de las columnas de accidents_t para tener una idea de los factores que tienen correlación:

[50]:
```
# Plotea la correlación entre los factores
sns.set(style="white")

# Calcula la matriz de correlación
corr = accidents_t.corr()

# Genera una máscara para el triángulo superior
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Configura la figura de matplotlib
f, ax = plt.subplots(figsize=(11, 9))

# Genera un mapa de colores divergentes personalizado
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Dibuja el mapa de calor con la máscara y la relación de aspecto correcta
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title("Correlación de los factores:")
plt.show()
```



Podemos ver que la severidad de los accidentes tiene una pequeña correlación con cada una de las condiciones meteorológicas y con la cantidad de tráfico. La hora del accidente tiene también una pequeña correlación con cada una de las condiciones meteorologías y con la cantidad de tráfico. Es lógico porque según la hora del día, las condiciones meteorológicas cambian de manera general.

[51]:
```
# Visualiza la matriz de correlación con números
corr
```

[51]:
```
                 Hour_accident  Severity  Traffic_quantity  Temperature(F)  \
Hour_accident         1.000000  0.016745          0.186655        0.206916
Severity              0.016745  1.000000         -0.039632       -0.033282
Traffic_quantity      0.186655 -0.039632          1.000000        0.242844
Temperature(F)        0.206916 -0.033282          0.242844        1.000000
Humidity(%)          -0.303692  0.019956         -0.303915       -0.301675
Visibility(mi)        0.092258 -0.009614          0.058768        0.168928

                 Humidity(%)  Visibility(mi)
```

```
Hour_accident       -0.303692        0.092258
Severity             0.019956       -0.009614
Traffic_quantity    -0.303915        0.058768
Temperature(F)      -0.301675        0.168928
Humidity(%)          1.000000       -0.383021
Visibility(mi)      -0.383021        1.000000
```

Aquí podemos ver el nivel de correlación entre los diferentes factores con más precisión.

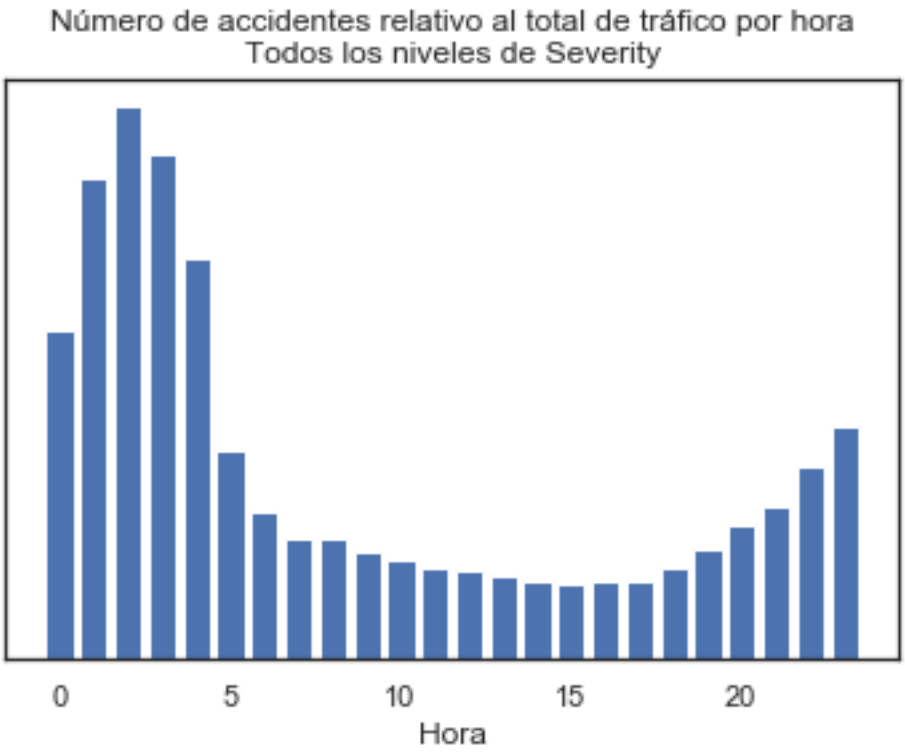
6.2.1 4.2.1. Número de accidentes relativo al total de tráfico por hora

Ahora en esta parte vamos a calcular el número de accidentes relativo al total de tráfico por hora, para saber a qué hora es más peligroso conducir.

[52]:
```
# Normaliza la columna Traffic_quantity
a_min = accidents_t["Traffic_quantity"].min()
a_max = accidents_t["Traffic_quantity"].max()
accidents_t["Traffic_quantity_norm"] = (accidents_t["Traffic_quantity"] -␣
 ↪a_min) / (a_max - a_min)
```

[53]:
```
# Añade una columna dummy para ayudar a calcular la cantidad de accidentes
accidents_t["Number_accident"] = 1
```

[54]:
```
# Crea una dataframe con la hora y la cantidad relativa de accidentes por hora
hour_traffic_total = accidents_t.
 ↪groupby("Hour_accident")[["Number_accident","Traffic_quantity_norm"]].sum()
hour_traffic_total["Relative_accident"] =␣
 ↪round(hour_traffic_total["Number_accident"] /␣
 ↪hour_traffic_total["Traffic_quantity_norm"],1)
hour_traffic_total = hour_traffic_total.reset_index()

# Plotea la dataframe
plt.bar(hour_traffic_total["Hour_accident"],␣
 ↪hour_traffic_total["Relative_accident"])
frame = plt.gca()
frame.axes.yaxis.set_ticklabels([])
plt.xlabel("Hora")
plt.title("Número de accidentes relativo al total de tráfico por hora\nTodos␣
 ↪los niveles de Severity")
plt.show()
```



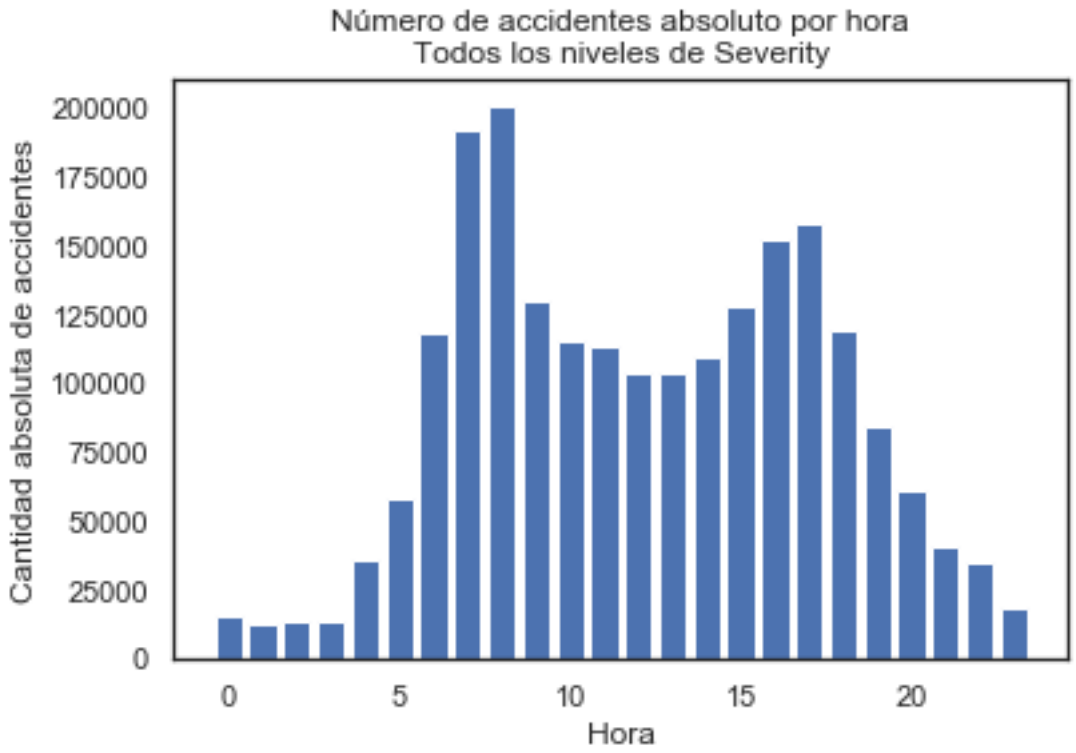
[55]:
```
# Calcula la correlación entre Hour_accident y Relative_accident
hour_traffic_total[["Hour_accident","Relative_accident"]].corr()
```

[55]:

	Hour_accident	Relative_accident
Hour_accident	1.000000	-0.611676
Relative_accident	-0.611676	1.000000

Podemos ver que se producen relativamente muchos más accidente durante la noche.

[56]:
```
# Crea una dataframe con la hora y la cantidad absoluto de accidentes por hora
hour_traffic_total = accidents_t.groupby("Hour_accident")[["Number_accident"]].
 ↪sum()
hour_traffic_total = hour_traffic_total.reset_index()

# Plotea la dataframe
plt.bar(hour_traffic_total["Hour_accident"],␣
 ↪hour_traffic_total["Number_accident"])
plt.xlabel("Hora")
plt.ylabel("Cantidad absoluta de accidentes")
plt.title("Número de accidentes absoluto por hora\nTodos los niveles de␣
 ↪Severity")
plt.show()
```



Durante el día hay más tráfico, pero menos accidentes proporcionalmente.

[57]:
```
# Calcula la correlación entre Hour_accident y Number_accident
hour_traffic_total[["Hour_accident","Number_accident"]].corr()
```

[57]:

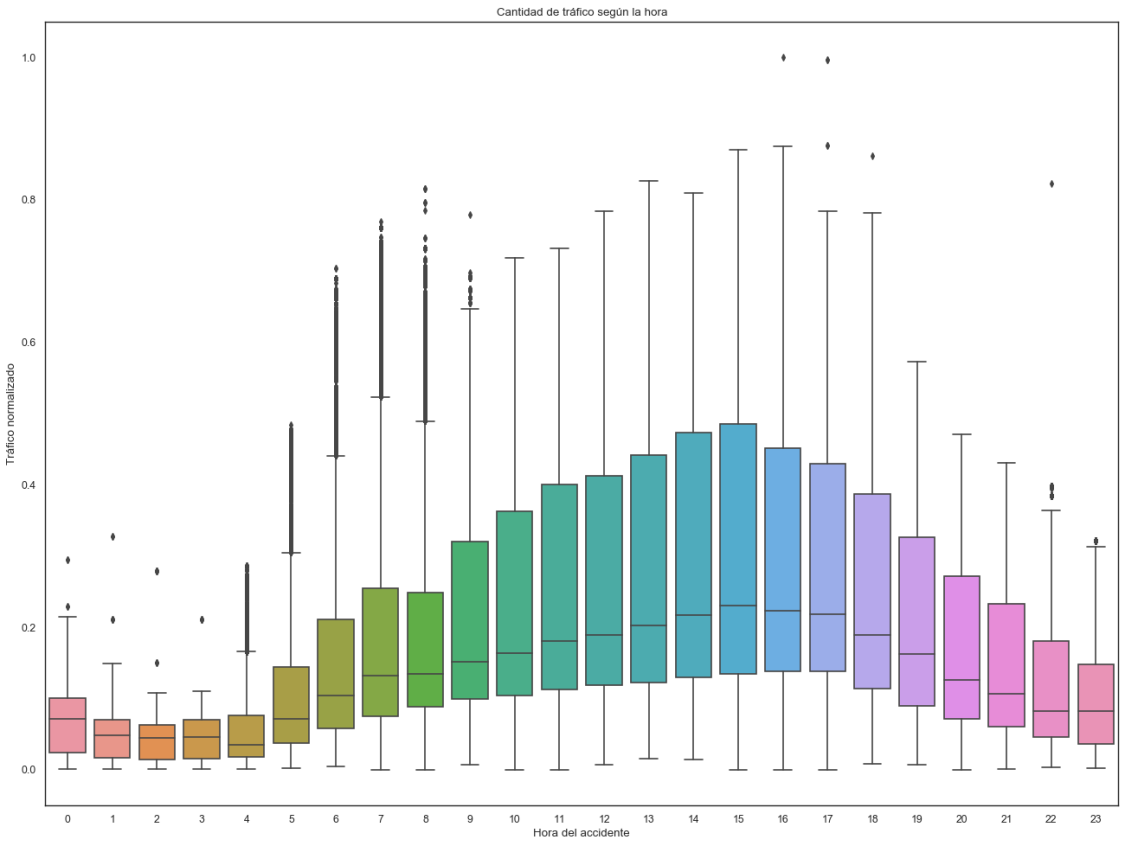
	Hour_accident	Number_accident
Hour_accident	1.00000	0.14988
Number_accident	0.14988	1.00000

Podemos ver que existe una alta correlación entre la cantidad de accidentes y la hora cuando ocurrió. Vamos a utilizar el Coeficiente de Correlación de Pearson (el cual se utiliza por defecto en la funcción corr()).

Lo que podemos ver, es que se producen más accidentes durante el día, porque hay mucho más tráfico, pero la probabilidad de un accidente es mayor durante la noche. Además, el número de accidente tiene un nivel alto de correlación con la hora tanto cuando se trata de la cantidad de accidente absoluta como relativa.

Vamos a visualizar la cantidad de tráfico según la hora mediante un boxplot para darnos cuenta:

[58]:
```
# Plotea un box plot con la cantidad de tráfico según la hora
fig, axes = plt.subplots(1, 1, figsize=(20,15), dpi= 80)
sns.boxplot(x='Hour_accident', y='Traffic_quantity_norm', data=accidents_t)
plt.title("Cantidad de tráfico según la hora")
plt.ylabel("Tráfico normalizado")
plt.xlabel("Hora del accidente")
plt.show()
```



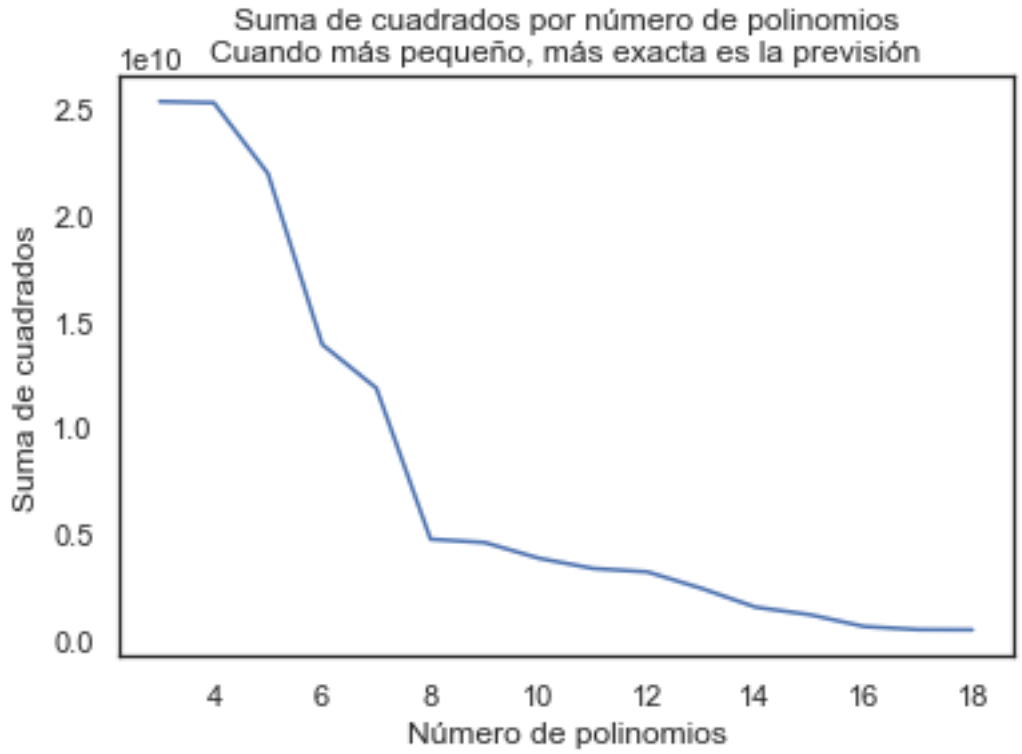
Podemos ver que hay más tráfico y más varianza durante el día.

6.2.2 4.2.2. Predicción del número de accidentes según la hora

Aquí vamos a implementar un sistema para predecir la cantidad de accidentes según la hora. La predicción se hace en este caso para la cantidad total de accidentes en los EE. UU. durante un periodo de tiempo de todo un año (por ejemplo: la cantidad de accidentes total entre las 8 y las 9 de la mañana), no obstante, se podría fácilmente adaptar para predecir la cantidad de accidentes según el día y el lugar. O sea, se podría predecir, por ejemplo, el número de accidentes en el estado de california el próximo 10 de julio entre las 8 y las 9 de la mañana. Para hacer la predicción utilizaremos la regresión polinómica y usaremos el método "The Method of Least Squares" para encontrar al mejor número de polinomios, para reducir al error.

[59]:
```
# Selecciona a los datos con el número de accidentes según la hora
acc_t = accidents_t.groupby("Hour_accident")
acc_t = acc_t.agg(np.sum)["Number_accident"]
y = acc_t.values
X = np.array(acc_t.index)

# Calcula el error según el número de polinomios
result = []
for s in range(3,19):
```

```
    z = np.polyfit(X, y, s)
    pred = np.poly1d(z)
    pred_s = [pred(d) for d in range(24)]
    squa = round(((pd.Series(acc_t) - pd.Series(pred_s ))**2).sum(),0)
    result.append([s,squa])
result = pd.DataFrame(result).set_index(0)

# Plotea los resultados
plt.plot(result)
plt.title("Suma de cuadrados por número de polinomios\nCuando más pequeño, más␣
 ↪exacta es la previsión")
plt.xlabel("Número de polinomios")
plt.ylabel("Suma de cuadrados")
plt.show()
```



Podemos ver en este caso que el número de polinomios más adecuado es 18. Vamos a utilizarlo para hacer unas predicciones.

[60]:
```
# Crea el modelo con 18 polinomios
z = np.polyfit(X, y, 18)
pred = np.poly1d(z)
```

Habíamos antes categorizado a los accidentes según la hora. O sea, cuando la hora del accidente en el conjunto de datos transformado es 8, significa que el accidente ocurió entre 8:00 y 8:59.

Podemos utilizar estar caracteristicas para las predicciones y si queremos solo tener un minuto, podemos dividir por 60. Se trata por cierto de una estimación

[61]:
```
print("Predicción de número de accidentes entre 8:30 y 9:30 de todo un año: {}".
 ↪format(int(pred(8.5))))
print("Predicción de número de accidentes entre 8:00 y 8:01 de todo un año: {}".
 ↪format(int(pred(8.5)/60)))
```

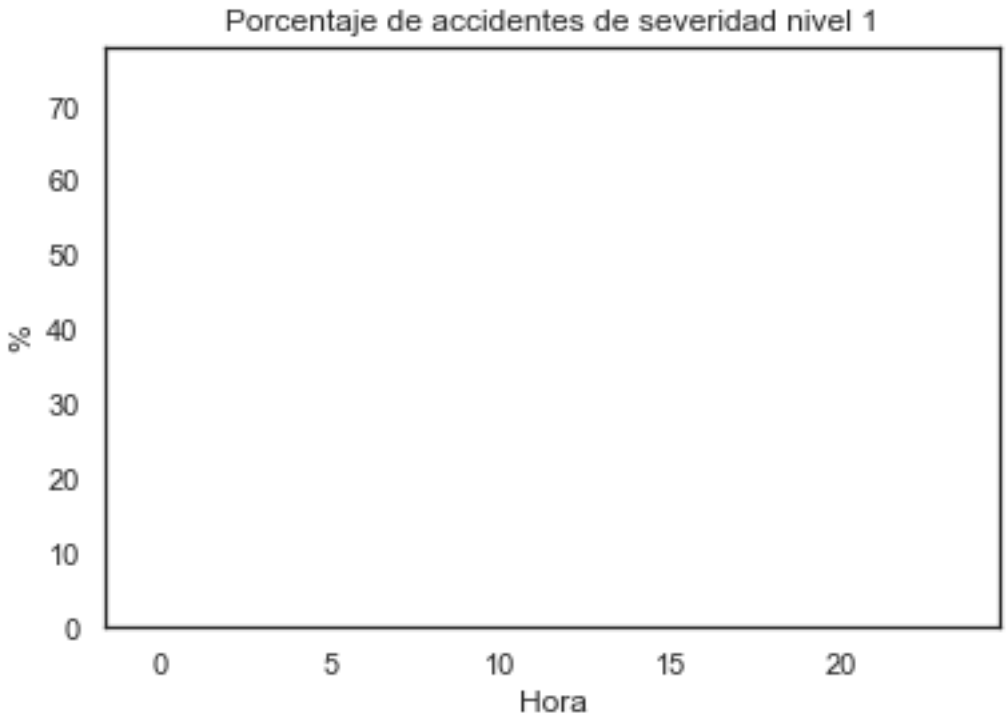
```
Predicción de número de accidentes entre 8:30 y 9:30 de todo un año: 168362
Predicción de número de accidentes entre 8:00 y 8:01 de todo un año: 2806
```

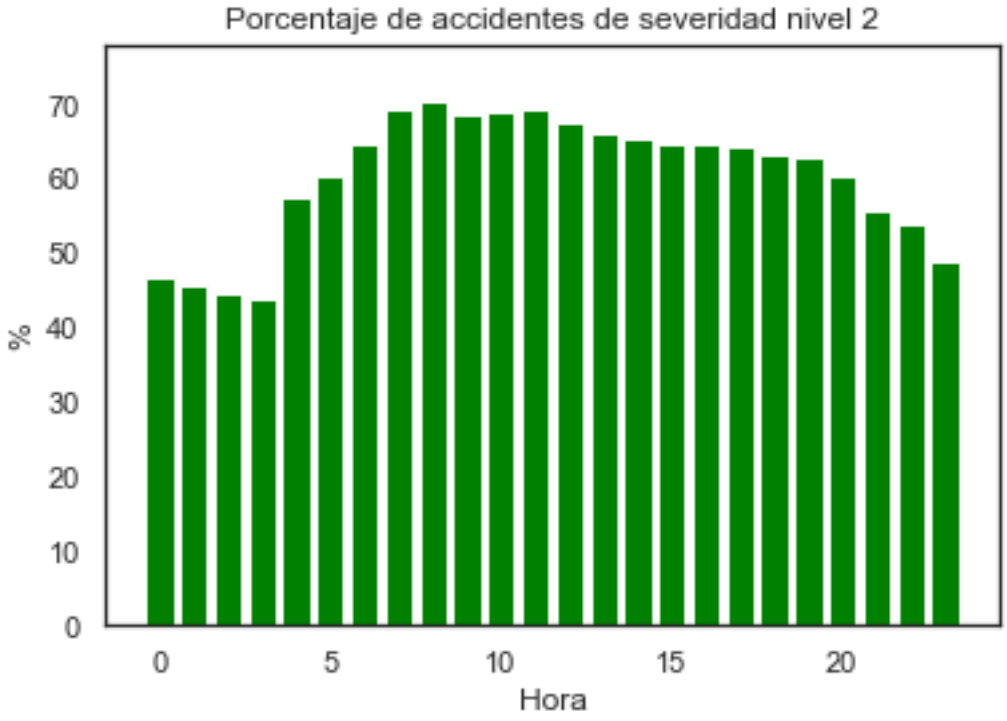
6.2.3 4.2.3. Accidentes según el nivel de severidad y la hora

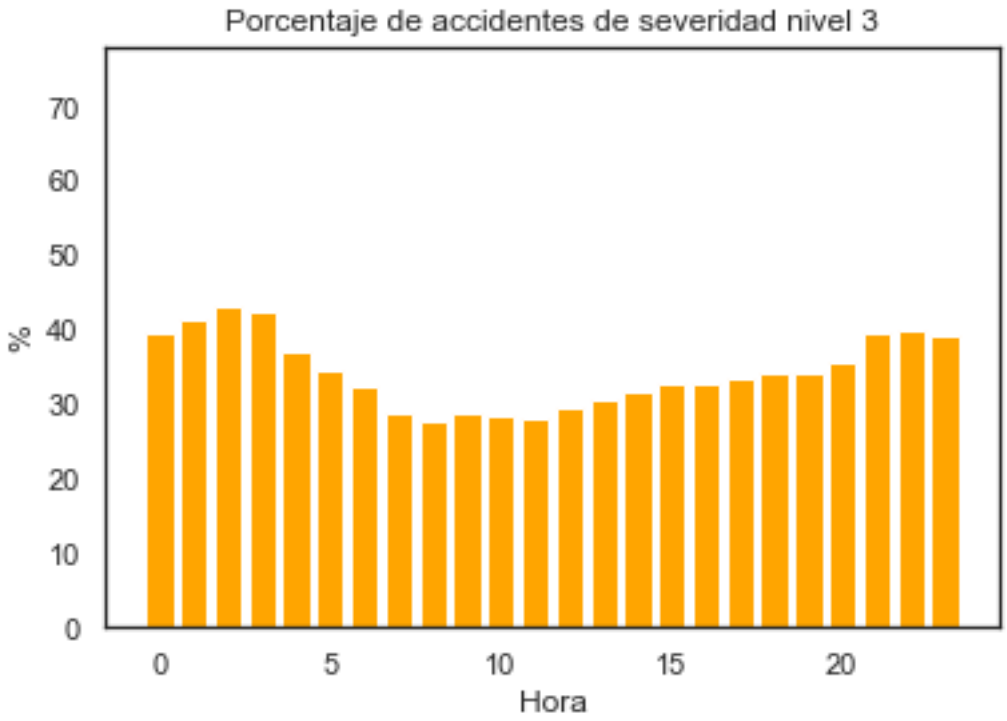
En esta parte vamos a visualizar el porcentaje de accidentes según el nivel de severidad y la hora.

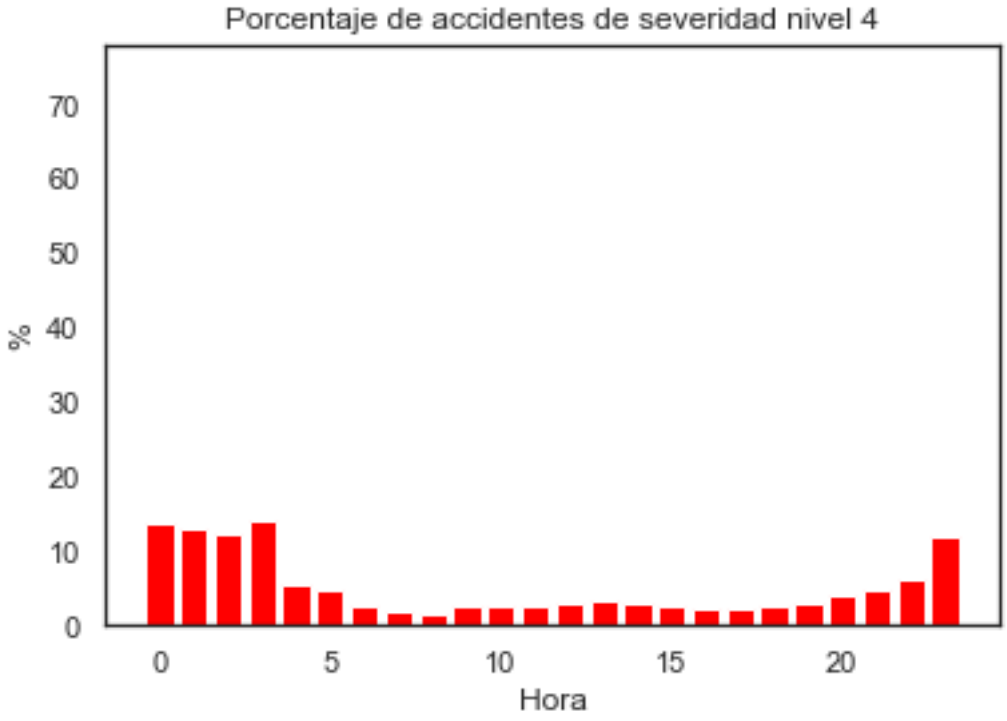
[62]:
```
# Obtiene una lista con los diferentes grados de severidad
severity_lvl = sorted(accidents_t["Severity"].unique())
```

[63]:
```
c = ["grey","green","orange","red"]
i = 0
for s in severity_lvl:
    # Calcula la cantidad relativa de accidente según la hora y la severidad
    # para cada nivel de severidad
    hour_traffic = accidents_t[accidents_t["Severity"] == s].
 ↪groupby("Hour_accident")[["Number_accident","Traffic_quantity_norm"]].sum()
    hour_traffic["Total_accidents"] = hour_traffic_total["Number_accident"]
    hour_traffic["Porcentage_s"] = round(100 * (hour_traffic["Number_accident"]␣
 ↪/ hour_traffic["Total_accidents"]),1)
    hour_traffic = hour_traffic.reset_index()

    # Plotea los resultados
    plt.bar(hour_traffic["Hour_accident"],␣
 ↪hour_traffic["Porcentage_s"],color=c[i])
    plt.title("Porcentaje de accidentes de severidad nivel {}".format(s))
    plt.xlabel("Hora")
    plt.ylabel("%")
    plt.ylim(0,78)
    plt.show()
    i += 1
```


Porcentaje de accidentes de severidad nivel 1
70
60
50
40
30
20
10
0
%
0
5
10
15
20
Hora


Porcentaje de accidentes de severidad nivel 2
70
60
50
40
30
20
10
0
%
0
5
10
15
20
Hora


Porcentaje de accidentes de severidad nivel 3
%
0
10
20
30
40
50
60
70
0
5
10
15
20
Hora


Porcentaje de accidentes de severidad nivel 4
%
0
10
20
30
40
50
60
70
0
5
10
15
20
Hora

Podemos ver que no ve a ningún accidente con la severidad 1. Estos podría venir del hecho que los accidentes con poca severidad se registren siempre y que se visualiza solo al porcentaje. Vamos a verificar la cantidad absoluta de accidentes con el nivel 1 de severidad:

[64]:
```
# Muestra la cantidad absoluta de accidentes según el nivel de severidad
accidents_t["Severity"].value_counts().sort_index()
```

[64]:
```
1        768
2    1392246
3     680643
4      66372
Name: Severity, dtype: int64
```

Hay muy pocos accidentes de severidad de nivel 1.

[65]:
```
# Calcula la correlación entre la severidad del accidente y la hora de este
# Utiliza el metodo: kendall
accidents_t[["Severity","Hour_accident"]].corr(method = "kendall")
```

[65]:

	Severity	Hour_accident
Severity	1.000000	0.019191
Hour_accident	0.019191	1.000000

Podemos ver que hay una pequeña correlación entre la hora del accidente y la severidad.

Prueba de independencia de la hora del accidente y de la severidad:

Ahora vamos a hacer una prueba para ver si la severidad depende de la hora del accidente utilizando el Chi_Squared Test. La hipótesis nula (H0) es que la hora del accidente y la severidad son independientes.

[66]:
```
from scipy.stats import chi2_contingency

# Crea la tabla de contingencia
table = pd.crosstab(accidents_t["Hour_accident"],accidents_t["Severity"])

# Esta función calcula la estadística de chi-square y el valor p para la prueba
# de hipótesis de independencia de las frecuencias observadas en la tabla de␣
 ↪contingencia observada.
stat, p, dof, expected = chi2_contingency(table)

# Interpreta el valor p
alpha = 0.001
print('Significación estadística=%.3f, p=%.3f' % (alpha, p))
print('\n\'Hour_accident\' y \'Severity\' son:')
if p <= alpha:
    print('dependientes (se rechaza a H0)')
else:
    print('independientes (no se rechaza a H0)')
```

```
Significación estadística=0.001, p=0.000

'Hour_accident' y 'Severity' son:
dependientes (se rechaza a H0)
```

Según esta prueba la hora del accidente y la severidad del accidente son dependientes con una confianza de 99,9%.

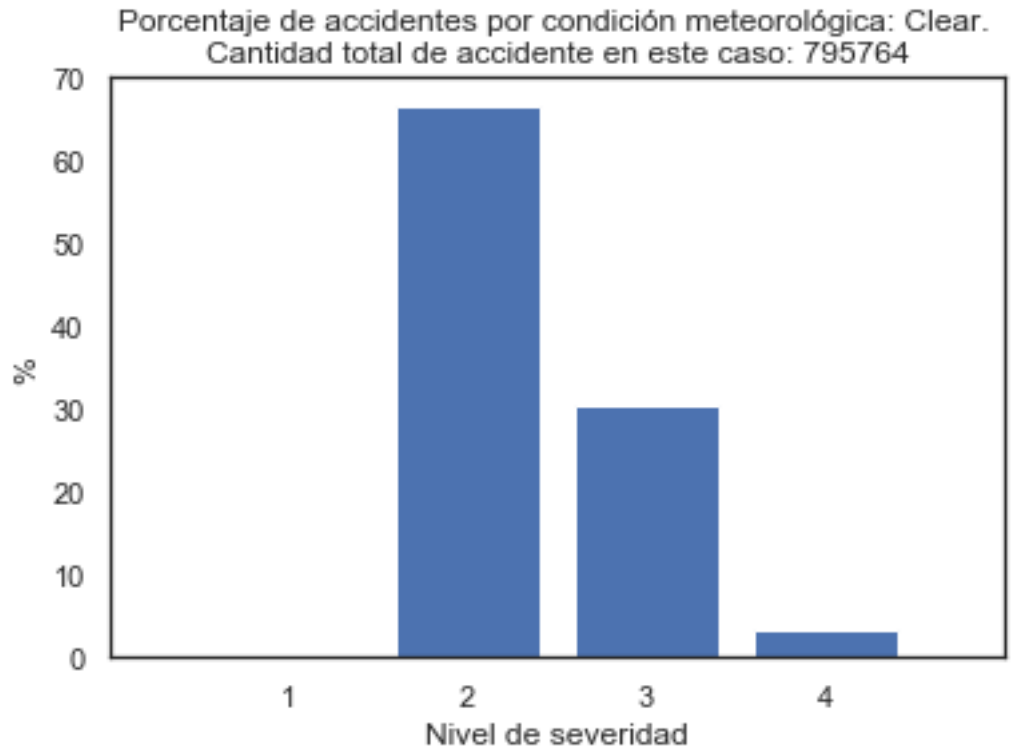
6.2.4 4.2.4. Influencia del tiempo meteorológico en la severidad de un accidente

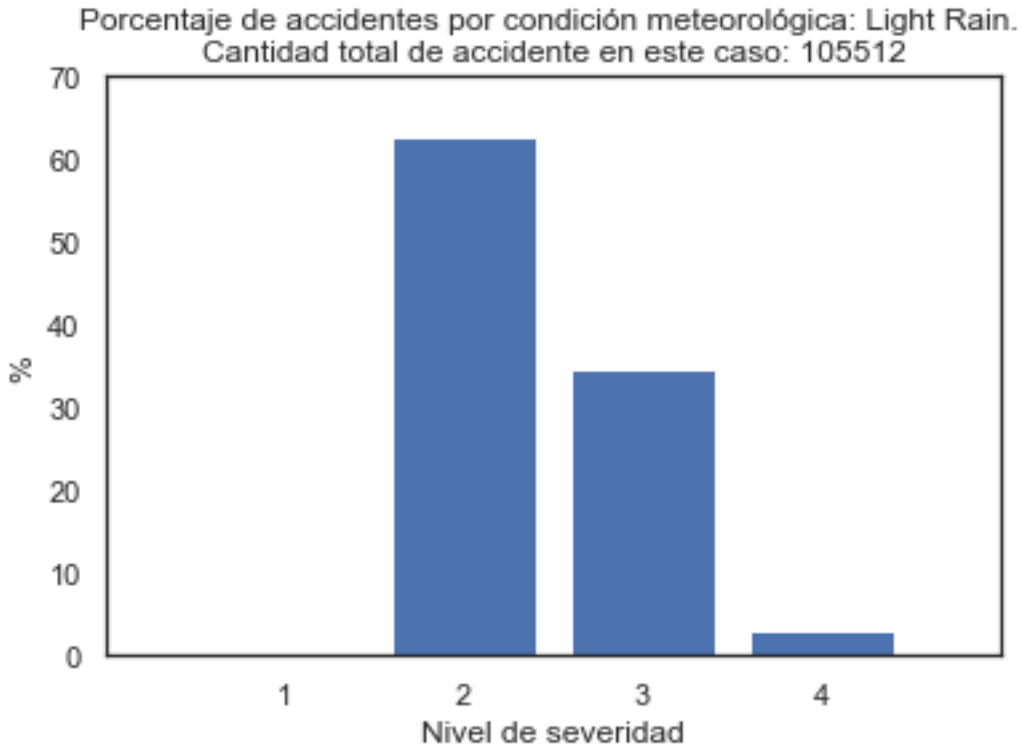
Aquí vamos a ver si el tiempo tiene una influencia en la severidad de un accidente.

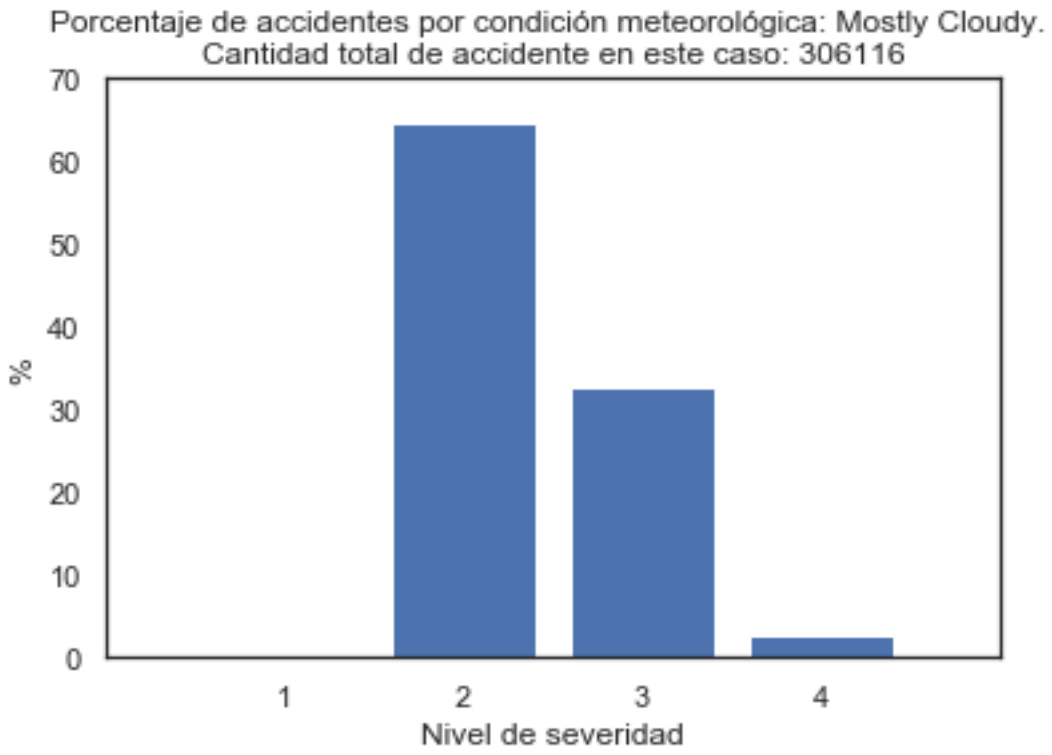
[67]:
```
# Grupa los datos por el tiempo y la severidad
# Calcula la cantidad de accidentes según estos dos factores
weather_severity = accidents_t.groupby(["Weather_Condition",␣
 ↪"Severity"])["Number_accident"].sum().reset_index()
```

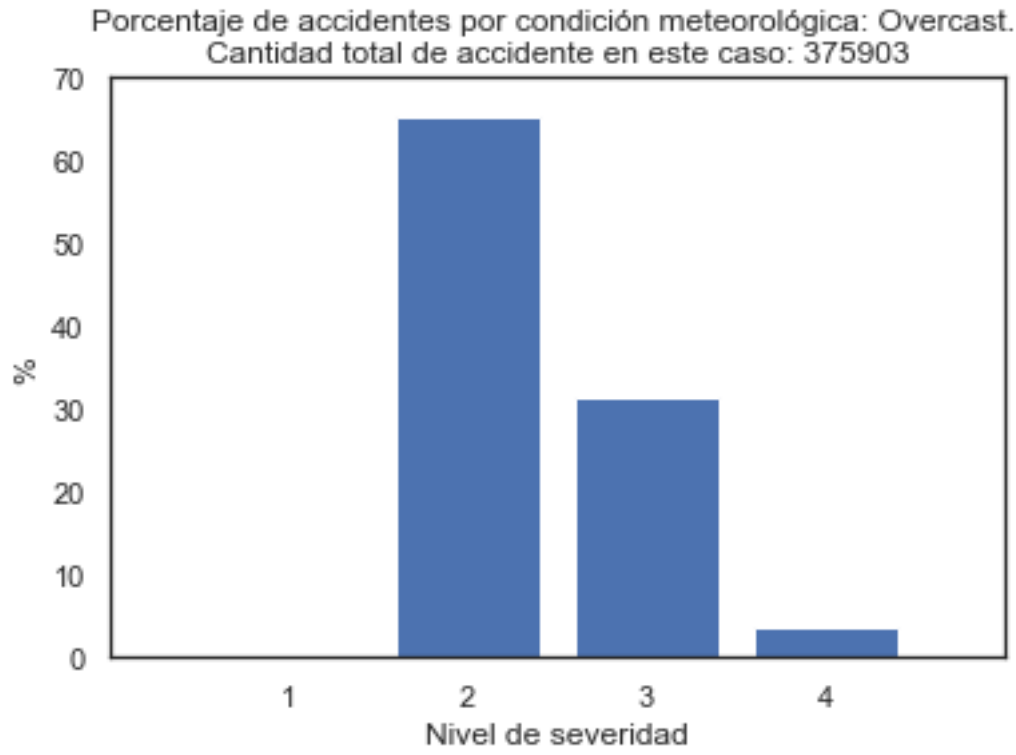
[68]:
```
# Lista con los diferentes tipos de tiempo
different_weather = weather_severity["Weather_Condition"].unique()
```

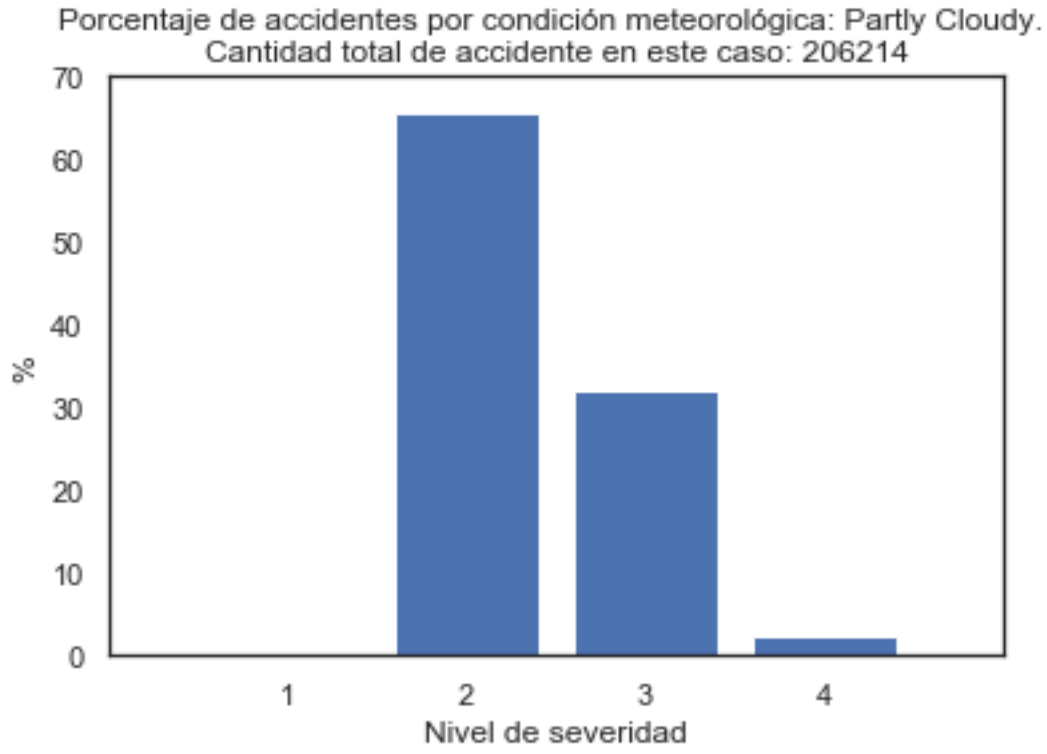
[69]:
```
# Plotea el porcentaje de accidentes según el nivel de seguridad
# para los casos más comunes (>100.000 accidentes)
for w in different_weather[1:]:
    wd = weather_severity[weather_severity["Weather_Condition"] == w].copy()
    sum_acc = wd["Number_accident"].sum()
    wd["Number_accident_%"] = round(100 * (wd["Number_accident"] / sum_acc),1)
    if sum_acc > 100000:
        y_pos = list(wd["Severity"])
        plt.bar(y_pos, wd["Number_accident_%"])
        plt.title("Porcentaje de accidentes por condición meteorológica: {}.␣
 ↪\nCantidad total de accidente en este caso: {}".format(w,sum_acc))
        plt.xlim(0,5)
        plt.ylim(0,70)
        plt.ylabel("%")
        plt.xlabel("Nivel de severidad")
        plt.xticks(y_pos, y_pos)
        plt.show()
```

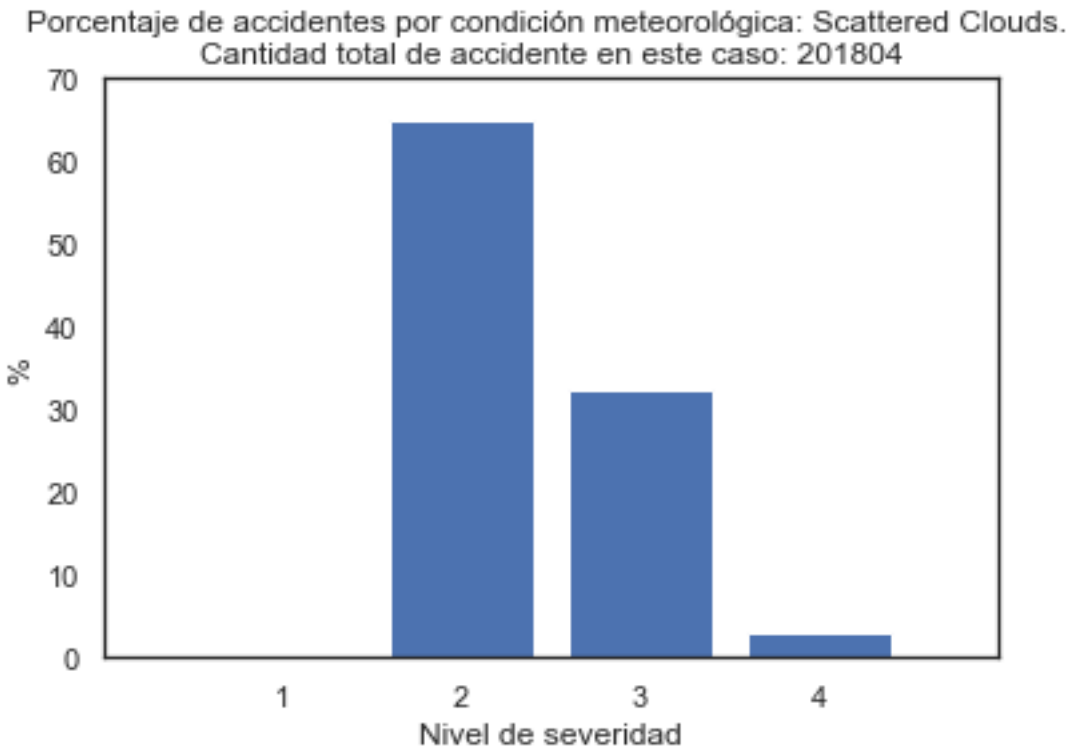

Porcentaje de accidentes por condición meteorológica: Clear.
Cantidad total de accidente en este caso: 795764
70
60
50
40
30
20
10
0
%
1
2
3
4
Nivel de severidad


Porcentaje de accidentes por condición meteorológica: Light Rain.
Cantidad total de accidente en este caso: 105512
70
60
50
40
30
20
10
0
%
1
2
3
4
Nivel de severidad


Porcentaje de accidentes por condición meteorológica: Mostly Cloudy.
Cantidad total de accidente en este caso: 306116
70
60
50
40
30
20
10
0
%
1
2
3
4
Nivel de severidad


Porcentaje de accidentes por condición meteorológica: Overcast.
Cantidad total de accidente en este caso: 375903
70
60
50
40
30
20
10
0
%
1
2
3
4
Nivel de severidad


Porcentaje de accidentes por condición meteorológica: Partly Cloudy.
Cantidad total de accidente en este caso: 206214
70
60
50
40
30
20
10
0
%
1
2
3
4
Nivel de severidad

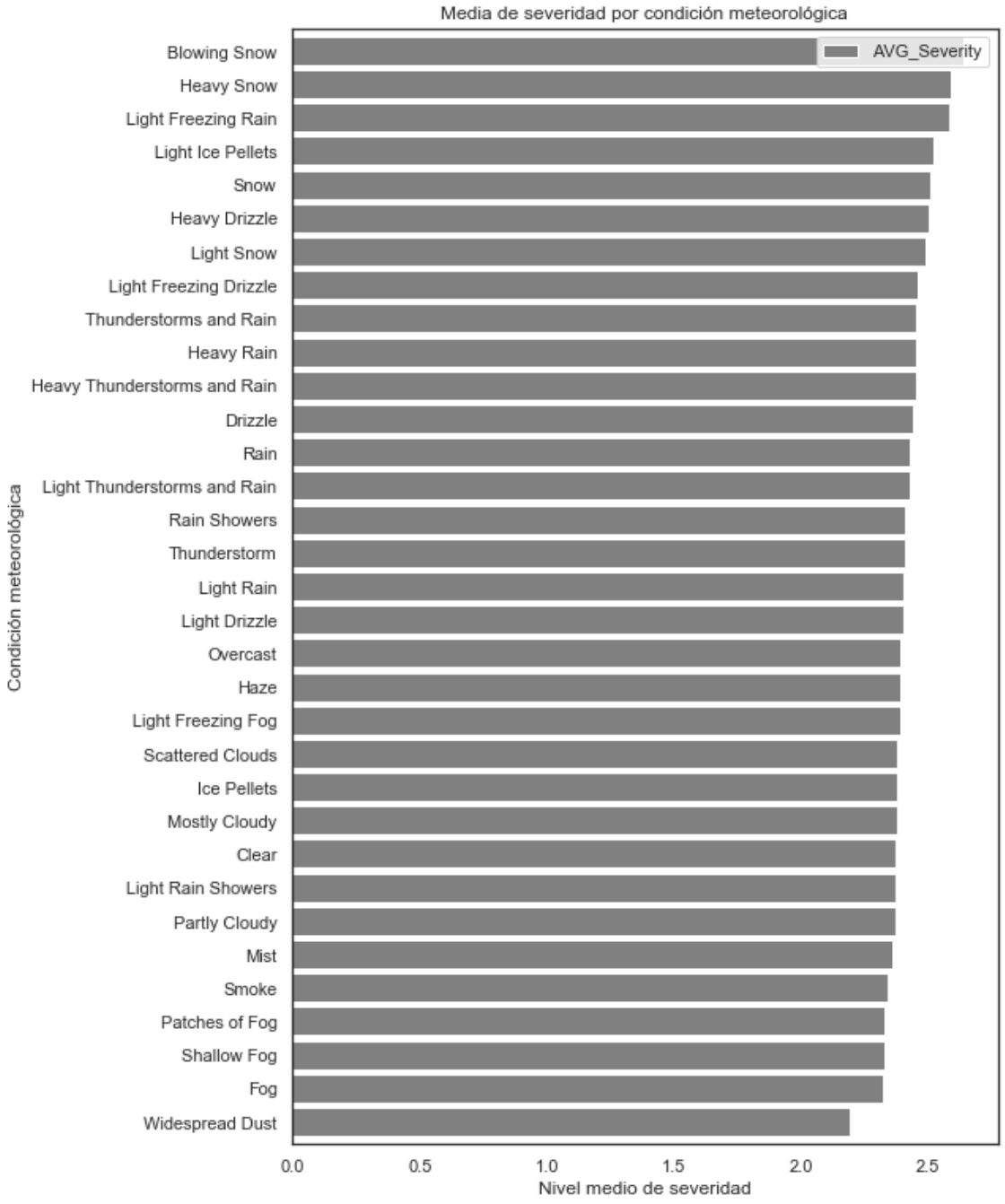


A simple vista, no podemos ver mucha diferencia en la repartición de la severidad según la condición meteorológica. Vamos a calcular la media de severidad por condición meteorológica, para ver si hay diferencias. Para la visualización tomaremos únicamente las condiciones meteorológicas con más de 100 accidentes para tener más exactitud.

[70]:
```
# Crea una lista con la media de severidad según la condición meteorológica
weather_severity["Sev_numb"] = weather_severity["Severity"] *␣
 ↪weather_severity["Number_accident"]
sev_we = []
for w in different_weather[1:]:
    wd = weather_severity[weather_severity["Weather_Condition"] == w].copy()
    sum_acc = wd["Number_accident"].sum()
    if sum_acc > 100:
        severy_avg = round(wd["Sev_numb"].sum() / sum_acc.sum(),2)
        sev_we.append([w, severy_avg])

# Convierte la lista en un DataFrame
sev_we = pd.DataFrame(sev_we, columns = ["Weather_Condition","AVG_Severity"])

# Ordena según la severidad media
sev_we = sev_we.sort_values("AVG_Severity")
```

```
# Plotea el resultado
ax = sev_we.set_index("Weather_Condition").plot(kind="barh", figsize=(8, 13),␣
 ↪color='gray', zorder=2, width=0.85)
ax.set_title("Media de severidad por condición meteorológica")
ax.set_xlabel("Nivel medio de severidad")
ax.set_ylabel("Condición meteorológica")
```

[70]: `Text(0, 0.5, 'Condición meteorológica')`



Se puede ver que, de manera general, cuando hay factores que reducen a la vista de los conductores, la severidad de los accidentes disminuye. Cuando las calles están resbaladizas la severidad aumenta. Para verificar esto, vamos a hacer una prueba de independencia de las condiciones meteorológicas y del nivel de severidad de los accidentes.

Prueba de independencia entre las condiciones meteorológica y entre el nivel de severidad de los accidentes. Ahora vamos a hacer una prueba para ver si la severidad depende de las condiciones meteorológicas utilizando el Chi_Squared Test. La hipótesis nula (H0) es que la condición meteorológica y la severidad son independientes.

[71]:
```
# Crea la tabla de contingencia
table = pd.crosstab(accidents_t["Weather_Condition"],accidents_t["Severity"])

# Esta función calcula la estadística de chi-square y el valor p para la prueba
# de hipótesis de independencia de las frecuencias observadas en la tabla de␣
 ↪contingencia observada.
stat, p, dof, expected = chi2_contingency(table)

# Interpreta el valor p
alpha = 0.001
print('Significación estadística=%.3f, p=%.3f' % (alpha, p))
print('\n\'Weather_Condition\' y \'Severity\' son:')
if p <= alpha:
    print('dependientes (se rechaza a H0)')
else:
    print('independientes (no se rechaza a H0)')
```

```
Significación estadística=0.001, p=0.000

'Weather_Condition' y 'Severity' son:
dependientes (se rechaza a H0)
```

Con un nivel de confianza de 99,9% podemos decir según la prueba que las condiciones meteorológicas y el nivel de severidad de los accidentes son dependientes. Sin embargo, hay condiciones meteorológicas que no son corrientes como podemos ver en la tabla abajo:

[72]:
```
accidents_t["Weather_Condition"].value_counts()
```

[72]:
```
Clear                   795764
Overcast                375903
Mostly Cloudy           306116
Partly Cloudy           206214
Scattered Clouds        201804
Light Rain              105512
Light Snow               35776
Haze                     27276
Rain                     24329
Fog                      11427
Heavy Rain                8768
```

Light Drizzle	7884
Light Thunderstorms and Rain	4868
Thunderstorm	4330
Snow	4054
Smoke	2961
Heavy Thunderstorms and Rain	2447
Thunderstorms and Rain	2187
Light Freezing Rain	1973
Patches of Fog	1847
Mist	1841
Drizzle	1577
Heavy Snow	1127
Light Freezing Fog	995
Shallow Fog	833
Light Freezing Drizzle	776
Blowing Snow	264
Light Ice Pellets	261
Heavy Drizzle	197
Light Rain Showers	156
Widespread Dust	129
Rain Showers	122
Ice Pellets	101
Squalls	26
Small Hail	25
Light Snow Showers	24
Light Thunderstorms and Snow	22
Volcanic Ash	21
Funnel Cloud	17
Light Haze	10
Heavy Thunderstorms with Small Hail	7
Heavy Rain Showers	7
Heavy Thunderstorms and Snow	5
Low Drifting Snow	5
Heavy Ice Pellets	4
Heavy Blowing Snow	4
Snow Grains	4
Light Fog	4
Light Hail	3
Light Blowing Snow	3
Thunderstorms and Snow	3
Light Snow Grains	3
Snow Showers	2
Heavy Freezing Rain	2
Heavy Freezing Drizzle	2
Hail	2
Light Thunderstorm	2
Blowing Sand	1

```
Heavy Smoke                                1
Dust Whirls                                1
Name: Weather_Condition, dtype: int64
```

Vamos a hacer la prueba otra vez utilizando a las condiciones meteorológicas más corrientes durante los accidentes a fin de vez si hay diferencias:

[73]:
```
# Selecciona a la líneas con condiciones meteorológicas que ocurrieron más de␣
 ↪10000 veces.
weather_count = accidents_t["Weather_Condition"].value_counts()
top_weather =  weather_count[weather_count > 10000].index.values
accidents_top_weather = accidents_t[accidents_t["Weather_Condition"].
 ↪apply(lambda d: d in top_weather)]

# Crea la tabla de contingencia
table = pd.
 ↪crosstab(accidents_top_weather["Weather_Condition"],accidents_top_weather["Severity"])

# Esta función calcula la estadística de chi-square y el valor p para la prueba
# de hipótesis de independencia de las frecuencias observadas en la tabla de␣
 ↪contingencia observada.
stat, p, dof, expected = chi2_contingency(table)

# Interpreta el valor p
alpha = 0.001
print('Significación estadística=%.3f, p=%.3f' % (alpha, p))
print('\n\'Hour_accident\' y \'Severity\' son:')
if p <= alpha:
    print('dependientes (se rechaza a H0)')
else:
    print('independientes (no se rechaza a H0)')
```

```
Significación estadística=0.001, p=0.000

'Hour_accident' y 'Severity' son:
dependientes (se rechaza a H0)
```

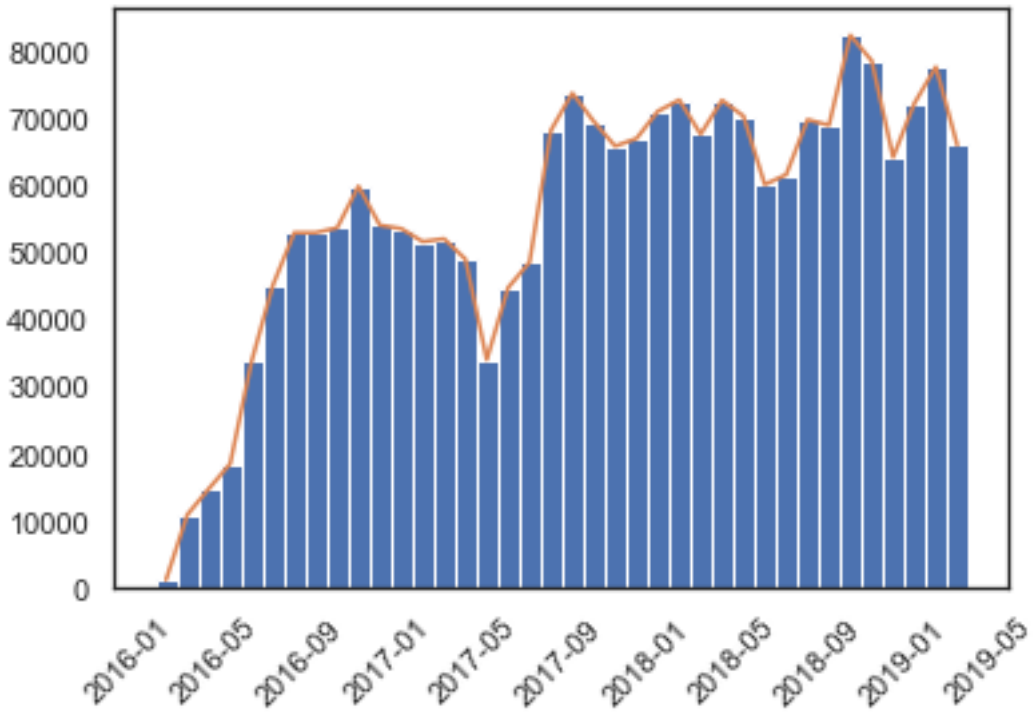
Obtenemos los mismos resultados.

6.2.5 4.2.5. Distribución de la cantidad de accidentes por mes

Ahora vamos a ver cuál es la cantidad de accidentes según el mes.

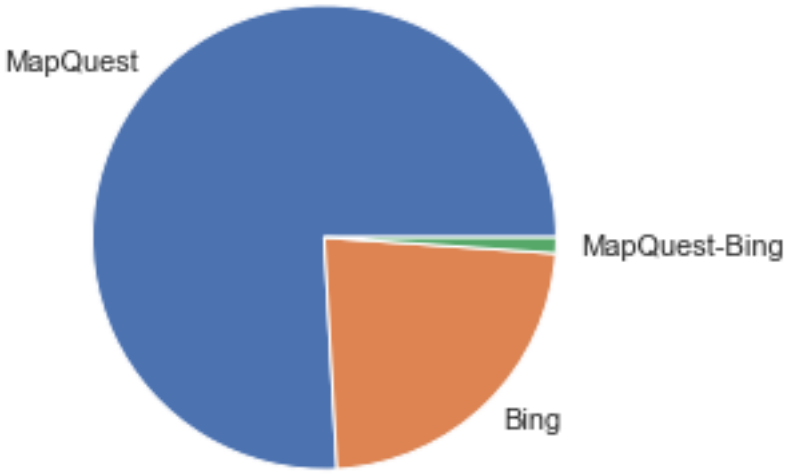
[74]:
```
# Calcula la cantidad de meses en el conjunto de datos
nb_month = len(accidents_t["Date"].apply(lambda d: str(d.year) + str(d.month)).
 ↪unique())

# Plotea un histograma
n,x,_ = plt.hist(accidents_t["Date"], bins = nb_month )
```

```
bin_centers = 0.5*(x[1:]+x[:-1])
plt.plot(bin_centers,n)
plt.xticks(rotation=45)
plt.show()
```



Vamos a visualizar de dónde vienen los datos.

[75]:
```
ax = accidents["Source"].value_counts().plot(kind="pie")
ax.set_ylabel("")
plt.show()
```



Podemos ver que la cantidad de accidentes según el año varia bastante. Esto se debe probablemente al hecho que el conjunto de datos se hizo con diferentes datos y los datos de algunas fuentes empezaron después. Por eso hacer un análisis de cantidad de accidentes según el año para ver si hubo un aumento o una disminuyo no tiene sentido en este caso.

7 5. Conclusiones

Durante el análisis, se realizaron varios tipos de pruebas estadísticas sobre el conjunto de datos que corresponde a los accidentes en los EE. UU. para saber cuáles son los factores que influencian a los accidentes después de haberlo limpiado. El problema mayor para obtener este conocimiento es que el conjunto de datos US_Accidents contiene a los accidentes, pero no contiene a los trayectos que ocurrieron sin problemas. Juntando a otro conjunto de datos US_Traffic hemos podido tener una estimación del tráfico según la hora, lo que permite obtener las horas cuando hay más accidentes de manera relativa y no solo absoluta. Además, hemos utilizado regresión polinómica para predecir la cantidad de accidentes. Usando la severidad del accidente según las condiciones meteorologías, hemos podido ver cuando los accidentes tienen más severidad. Con un nivel de confianza de 99,9% podemos decir que las condiciones meteorologías influyen a la severidad. Sin embargo, para obtener resultados exactos de las influencias, necesitaríamos tener otro conjunto de datos, con las condiciones meteorologías en el pasado según el lugar y juntarlo con los otros que ya tenemos. A fin de cuenta, podemos decir que de manera general se producen proporcionalmente más accidentes durante la noche, la severidad de los accidentes es mayor cuando el suelo es resbaladizo. La severidad de los accidentes es influenciada por la hora y está un poco más alta durante la noche. Este análisis podría ser expendido a otros factores, los cuales no hemos tratado aquí, como la influencia de la temperatura, humidad, del lugar en la calle (intersección, etc.) en la severidad o en el número de los accidentes. Combinando información meteorológica exacta se podría también obtener estimación del riesgo de tener un accidente con qué severidad, según el número de kilómetros recorridos bajo cuales condiciones meteorológicas. Es decir que este estu-

dio podría tener muchas facetas. Los cuales podrían ayudar a optimizar a los sistemas de socorro de personas y a los sistemas de seguridad de los coches.

8 Fuentes

- Jason Brownlee (28.11.2019). 17 Statistical Hypothesis Tests in Python. En Machine Learning Mastery. Australia. https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/
- Jason Brownlee (15.06.2019). A Gentle Introduction to the Chi-Squared Test for Machine Learning. https://machinelearningmastery.com/chi-squared-test-for-machine-learning/

9 Autor:

Contribuciones
Firma
Investigación previa
MB
Redacción de las respuestas
MB
Desarrollo del código
MB

10 Licencia