Limpieza y análisis de datos

ENERO 7, 2020

Tipología y ciclo de vida de los datos
Alejandro Martínez Otal

Índice

Introducción

En este documento se recoge la entrega final de la asignatura ''***Tipología y ciclo de vida de los datos***'' en la que se pretende recoger todos los conceptos que se han adquirido a lo largo del curso de un modo práctico. Se tratan conceptos más globales como el ciclo de vida de los datos grosso modo, características de los datos con los que se va a tratar y posibles complicaciones que pueden ocurrir en casos reales. A continuación, una vez se tiene un juego de datos corregido, se busca sacar provecho de este mediante técnicas: desde creación de gráficos que permiten obtener un *overview* de los datos, hasta la aplicación de algoritmos que permitan realizar predicciones.

Descripción de los datos

Se ha elegido un juego de datos que trata sobre los accidentes de tráfico en EEUU **[1]** (Estados Unidos). Este contiene todos los accidentes registrados desde febrero del 2016 hasta marzo del 2019. El tamaño del fichero es considerable (819 MB) y las dimensiones del juego de datos son de $\mathbf{2243940 \times 49}$.

A continuación, se presenta una breve descripción de cada columna:

- ID: Identificador único del accidente.
- Source: Fuente que ha informado sobre el accidente.
- TMC: Algunos accidentes pueden tener un Traffic Message Channel, que puede dar más detalles sobre el accidente.
- Severity: Indica la severidad en un rango numérico categórico $[1 - 4]$.
- Start_Time: Tiempo de inicio del accidente en tiempo local.
- End_Time: Tiempo de fin del accidente en tiempo local.
- Start_Lat: Coordenadas de latitud GPS donde empezó el accidente.
- End_Lng: Coordenadas de longitud GPS donde empezó el accidente.
- End_Lat: Coordenadas de latitud GPS donde empezó el accidente.
- End_Lng: Coordenadas de longitud GPS donde empezó el accidente.
- Distance(mi): Distancia en millas de la carretera que afectó el accidente.
- Description: Descripción en lenguaje natural del accidente.
- Number: Número de la calle donde se registró el accidente.
- Street: Nombre de la calle donde se registró el accidente.
- Side: Mitad relativa de la vía dónde se produjo el accidente $(Left/Right)$.
- City: Ciudad donde ocurrió el accidente.
- County: Condado donde se registró el accidente.
- State: Estado donde se registró el accidente.
- Zipcode: Código postal donde se registró el accidente.
- Country: País dónde se registró el accidente.
- Timezone: Muestra el tiempo en base a la localización del accidente.
- Airport_Code: Indica una estación de tiempo aeroportuaria más cercana al accidente.
- Weather_Timestamp: Indica el momento, a nivel local, de cuando se capturó la observación del tiempo.

- Temperature(F): Temperatura en F.
- Wind_Chill(F): Efecto del viento sobre la temperatura en F.:
- Humidity(%): Porcentaje de humedad.
- Pressure(in): Presión atmosférica.
- Visibility(mi): Visibilidad.
- Wind_Direction: Dirección del viento.
- Wind_Speed(mph): Velocidad del viento.
- Precipiation(in): Cantidad de precipitaciones.
- Weather_Condition: Condición del tiempo.
- Amenity
- Bump: Existencia de *Bumpers* en la vía.
- Crossing: Existencia de cruces en la vía.
- Give_Way: Existencía de ceda el paso en la vía.
- Junction: Existencia de incorporación en la vía.
- No_Exit: Vía sin salida.
- Roundabout: Existencia de rotonda en la vía.
- Station: Existencia de una estación próxima en la vía.
- Civil_Twilight: Si se produce de día o noche.

Importancia y objetivo del análisis

El objetivo de este análisis se va a centrar en entender analizar si a partir de este juego de datos podemos encontrar algún tipo de relación en estos que nos permitan entender mejor porqué ocurren los accidentes.

Al mismo tiempo, se intentará entender también que factores hacen que un accidente sea más o menos grave.

Finalmente se intentará aplicar un algoritmo predictivo para intentar predecir accidentes.

Limpieza y análisis de los datos

Se procede inicialmente a realizar un análisis general de los datos, se procede a realizar la lectura del fichero mediante el *snippet* de código que se muestra a continuación:

```
data=pd.read_csv(r'US_Accidents_May19.csv')
```

En la Tabla 1 se muestra los tipos de datos que ha asignado automáticamente Python a cada columna, posteriormente si se considera que la interpretación es errónea se aplicarán las correcciones necesarias:

Nombre	Tipo
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

Tabla 1 *Tipos de datos de la columna.*

Queremos empezar a ver investigar nuestro juego de datos, analizar su contenido para ganar empezar a ganar los primeros *insights*.

El juego de datos viene de fuentes americanas, por lo que muchas veces no vienen las unidades de algunas medidas en el sistema métrico, o la temperatura en grados Celsius.

El primer paso, y por comodidad es convertir las unidades a valores que nos sean más familiares.

Las conversiones que se van a practicar son las siguientes:

I. Conversiones de temperatura (Celsius a Fahrenheit) para las columnas:

   a. Temperature(F) a Temperatura(C).

   b. Wind_Chill(F) a Wind_Chill(C).

Para esto se utiliza la función que se muestra a continuación:

```
def f_to_c(str_temp_f):
    f1=float(str_temp_f-32)
    f2=f1**(5/9)
    return f2
data['Temperature(C)']=data['Temperature(F)'].apply(f_to_c)
data['Wind_Chill(C)']=data['Wind_Chill(F)'].apply(f_to_c)
```

II. Conversión de velocidad(mph a kmh) para las columnas:

a. Wind_Speed(mph).

```
def mph_to_kmh(str_speed_mph):
    return str_speed_mph*1.60934
data['Wind_Speed(kmh)']=data['Wind_Speed(mph)'].apply(mph_to_kmh)
```

III. Conversión de distancia(millas a km) de las columnas:

a. Distance(mi).

b. Visibility(mi).

```
def mi_to_km(str_dist_mi):
    return str_dist_mi/0.62137119
data['Distance(km)']=data['Distance(mi)'].apply(mi_to_km)
data['Visibility(km)']=data['Visibility(mi)'].apply(mi_to_km)
```

IV. Conversión de distancia(pulgadas a centímetros) para las columnas:

a. Pressure(in).

b. Precipitation(in).

```
def inch_to_cm(str_dist_inch):
    return str_dist_inch*2.54
data['Pressure(cm)']=data['Pressure(in)'].apply(mi_to_km)
data['Precipitation(cm)']=data['Precipitation(in)'].apply(mi_to_km)
```

El siguiente paso es elegir que columnas creemos que nos serán útiles para resolver el problema planteado anteriormente.
Hemos visto a lo largo del curso que el volumen de datos con el que se trabaja puede afectar mucho al rendimiento de según que operaciones se realicen en estos, por lo que es importante trabajar con los justos y necesarios.
Se han elegido las siguientes columnas:

I. ID.
II. Source.
III. Severity.
IV. Start_Time.
V. End_Time.
VI. Start_Lat.
VII. End_Lat.
VIII. Start_Lng.
IX. End_Lng.
X. Distance(km).
XI. Side.
XII. City.
XIII. Country.
XIV. Humidity(%).
XV. Pressure(cm).
XVI. Visibility(km).
XVII. Wind_Direction.
XVIII. Wind_Speed(kmh).
XIX. Preciptation(cm).
XX. Weather_Condition.
XXI. Astronomical_Twilight.

Una vez reducido el juego de datos a nivel de columnas el siguiente paso que se realiza es analizar los datos para poder detectar posibles valores: nulos, *outliers*… Y como lidiar con ellos para un conjunto de columnas preseleccionado.

Severity

Variable categórica que indica la gravedad del accidente, y más adelante será la variable objetivo para nuestra predicción. Tiene sentido entonces que no nos interesé mantener valores nulos.
Inicialmente hacemos un conteo de valores mediante el código:

```
data_slim['Severity'].value_counts()
```

Que nos devuelve los resultados documentados en la Tabla 2:

Valor	Conteo
0	17
1	814
2	1455524
3	715582
4	720002
Total	**2891939**

Tabla 2 Conteo de valores para la variable Severity

Acorde con la documentación, el valor de *Severity* debería estar entre el rango $[\mathbf{1} - \mathbf{4}]$. Dado al gran número de registros que tenemos, se decide ignorar estos valores, en vez utilizar algoritmos como $\boldsymbol{kNN}$ para encontrar los valores perdidos.

Imputación de valores nulos

Se han elegido las siguientes columnas para validar la existencia de valores nulos:

XXII. Pressure(cm).
XXIII. Visibility(km).
XXIV. Distance(km).
XXV. Wind_Speed(kmh).
XXVI. Preciptation(cm).

Mediante el siguiente código podemos construir la Tabla 3, viendo que existen columnas con valores nulos:

```
COLS_NULL=['Pressure(cm)','Visibility(km)','Distance(km)','Wind_Speed(kmh)'
,'Precipitation(cm)']
for col in COLS_NULL:
    print('La columna {} tiene valores nulos
{}'.format(col,data_slim_red[col].isnull().values.any()))
```

Columna	Nulos
Pressure(cm)	Sí
Visibility(km)	Sí
Distance(km)	No
Wind_Speed(km)	Sí
Precipitation(cm)	Sí

Tabla 3 *Existencia de valores nulos en columnas*

Se decide de usar el algoritmo kNN **[2]** para imputar los valores que faltan. Sin intervención alguna el algoritmo pondrá todos los valores *nan* a el valor elegido basándose en los **3** vecinos más próximos basándose en la distancia Euclidiana. A continuación, se muestra el código utilizado:

```
knn_data=data_slim_red[['Pressure(cm)','Visibility(km)','Dis-
tance(km)','Wind_Speed(kmh)','Precipitation(cm)']].select_dtypes(in-
clude='number').as_matrix()
Age_KNN=KNN(k=3).fit_transform(knn_data)
```

Esto nos devuelve una matriz con todos los valores que inicialmente estaban como nulos que es substituido por los valores originales. Y repitiendo el *snippet* de código previamente obtenemos que ya no existe ningún valor nulo en las columnas elegidas.

El siguiente paso es analizar el caso de *outliers*, y una vez detectados analizar si estos pueden o deben ser reemplazados.
Se usan las mismas columnas de tipo numérico que se han mencionado en el apartado anterior para la imputación de valores nulos.

Para la variable *Pressurce(cm)* observamos el siguiente *boxplot* de la Figura 1 :

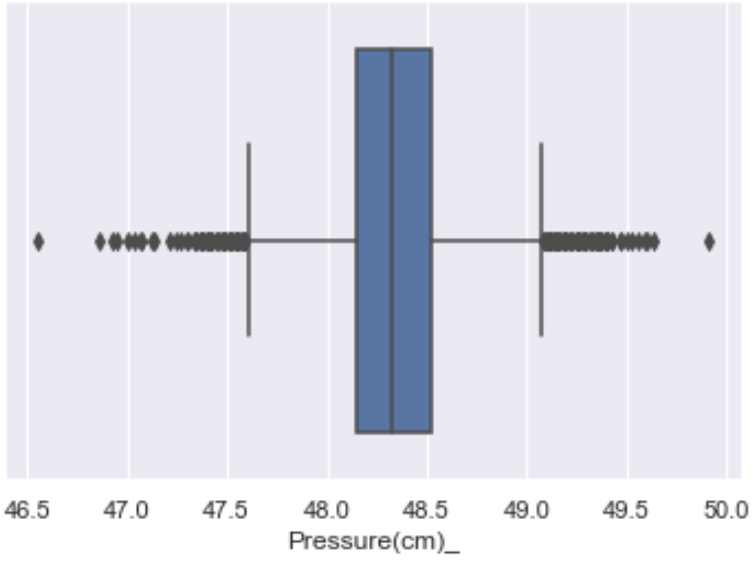

Figura 1 Boxplot de la variable Pressure(cm)

Inicialmente, se pueden observar que existen bastantes valores fuera del rango intercuartílico, para obtener un *insight* más profundo sobre los datos se pasa a dibujar un histograma en la Figura 2:

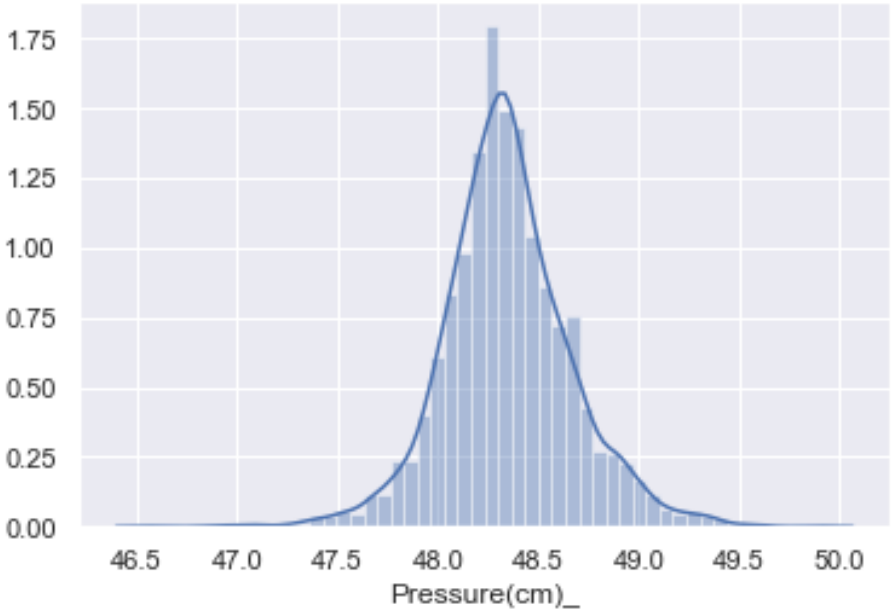

Figura 2 Histograma de la variable Pressure(cm).

Mediante el siguiente código obtenemos una serie de descriptivos sobre la variable:

```
display(complete_df['Pressure(cm)_'].describe())
```

Que se recogen en la Tabla 4 :

Descriptivo	Valor
count	300000
mean	48.34
std	0.32
min	46.56
0.25	48.15
0.5	48.33
0.75	48.52
max	49.91

Tabla 4 Descriptivos estadísticos de la variable Pressure(cm).

Como conclusión, se observa que los datos siguen una distribución bastante normal, y que la desviación estándar es bastante pequeña, entendiendo los outliers mostrados por el *boxplot* aceptables.

Para la variable *Visibility(km)* se observa un escenario diferente para el *boxplot* como se ve en la Figura 3:

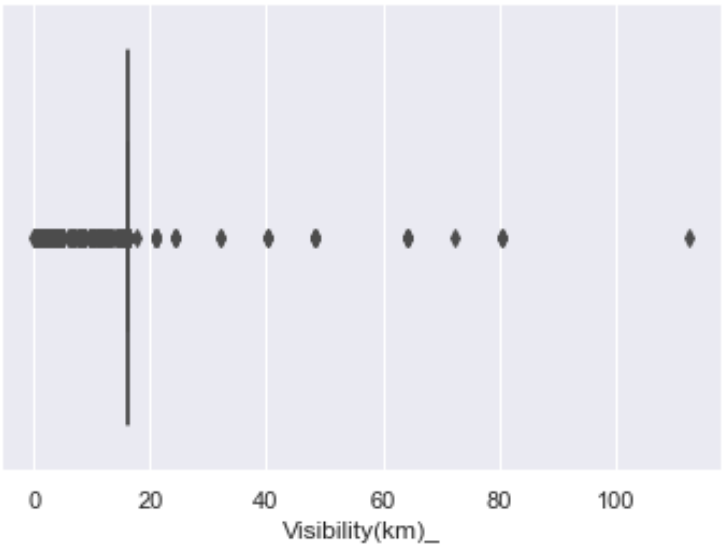

Figura 3 Boxplot para la variable Visiblity(km).

Se representa el histograma en la Figura 4:

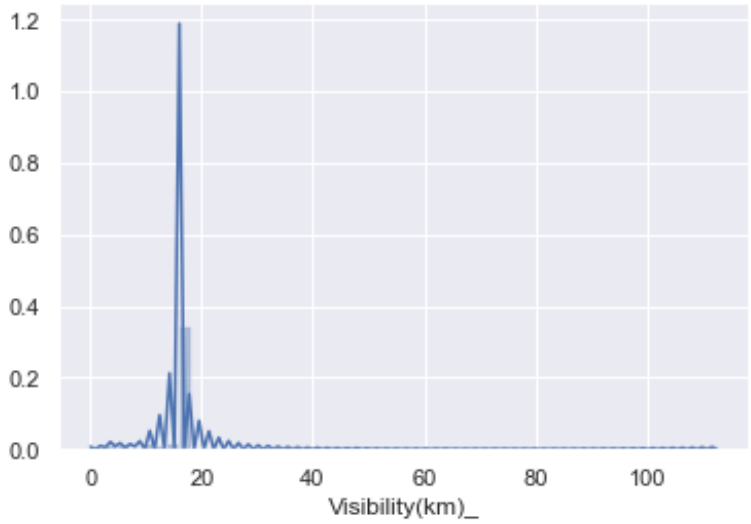

Figura 4 Histograma para la variable Visibility(km).

Se puede observar ya que la distribución de los datos es diferente respecto a la variable anterior, los datos parecen estar mucho más centrados cerca del 20, pero la *"cola"* del histograma por la derecha se alarga hasta valores superiores a 100, por lo que se calculan los estadísticos en la Tabla 5 :

Descriptivo	Valor
count	300000
mean	14.73046
std	4.711233
min	0
0.25	16.09344
0.5	16.09344
0.75	16.09344
max	112.6541

Tabla 5 Estadísticos descriptivos para la variable Visibility(km).

Se puede observar que la desviación estándar es bastante más grande en proporción a la media, y que los rangos intercuartílicos están todos a el mismo valor.
Se entiende que los valores mínimos a 0 son por falta de medidas, pero valores máximos como $112\ km/h$ no parecen muy exagerados, y debido a su baja frecuencia se entienden como *outliers* aceptables, es decir, no se tratan de errores de medición.

Para la variable *Distance(km)* se grafican un *boxplot* en la Figura 5:

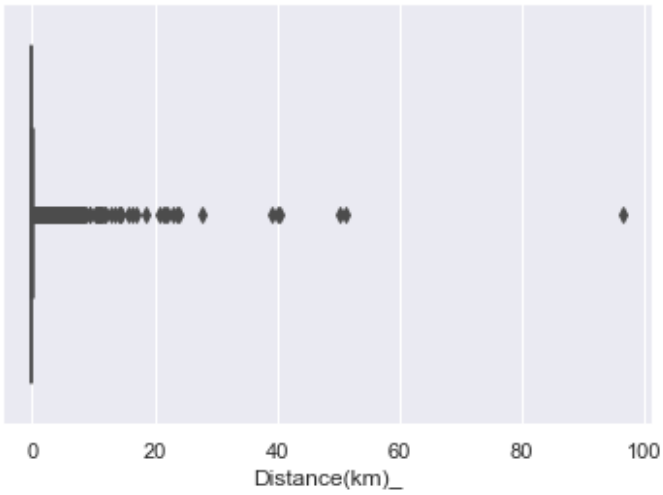

Figura 5 Boxplot para la variable Distance(km).

Parecido al caso anterior parece haber una gran concentración cerca del valor 0, pero hay valores cerca del 100 que visualmente están demasiado alejados, por lo que se estudia el histograma en la Figura 6:

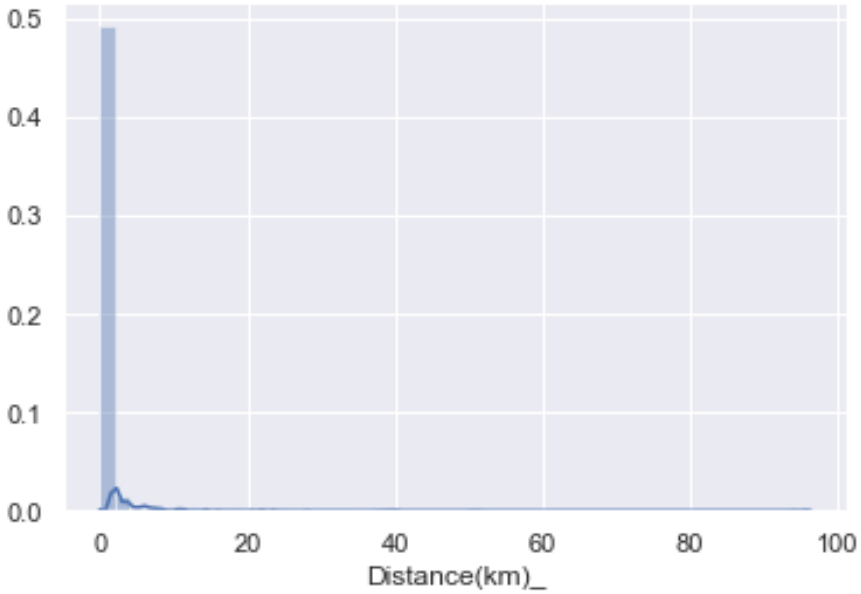

Figura 6 Histograma para la variable Distance(km).

Se muestran algunos datos estadísticos en la Tabla 6:

Descriptivo	Valor
count	300000
mean	0.469322
std	2.491047
min	0
0.25	0
0.5	0
0.75	0.016093
max	96.3305

Tabla 6 Estadísticos para la variable Distance(km).

Se puede observar que el valor respecto la media y la desviación estándar respecto al máximo es bastante grande, por lo que se decide hacer una última comprobación y listar los 5 valores más grandes del atributo en la Tabla 7:

ID	Distance(km)	Δ km (%)
A-2095483	96.33	88.64%
A-772747	51.06	1.44%
A-680539	50.34	24.37%
A-1032681	40.48	1.21%
A-232838	39.99	0.00%

Tabla 7 Variación de los valores de Distance(km).

Se puede observar que existe una gran variación respecto el resto de los valores, por lo que se decide eliminar el registro considerándolo un *outlier*.

Para la variable Wind_Speed(kmh) se obtiene el *boxplot* de la Figura 7:

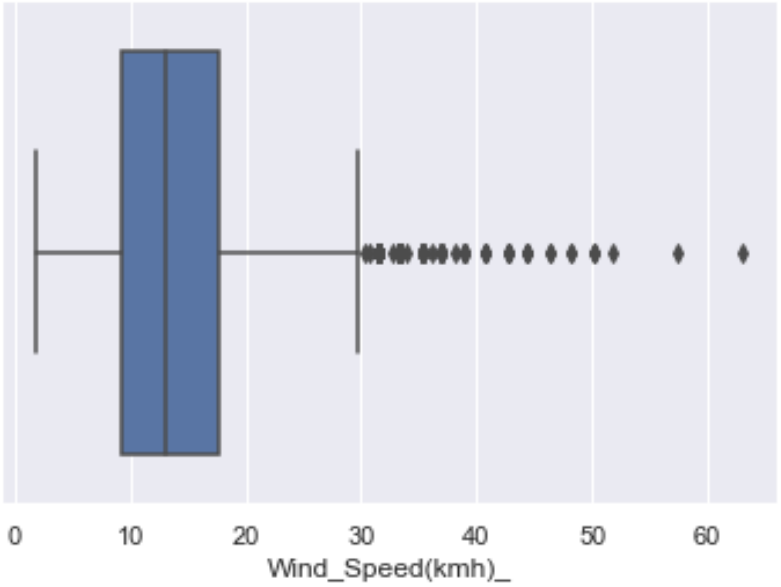

Figura 7 Boxplot para la variable Wind_Speed(kmh).

Se puede ver inicialmente la existencia de algunos valores extremos, se procede a representar el histograma de los datos en la Figura 8:

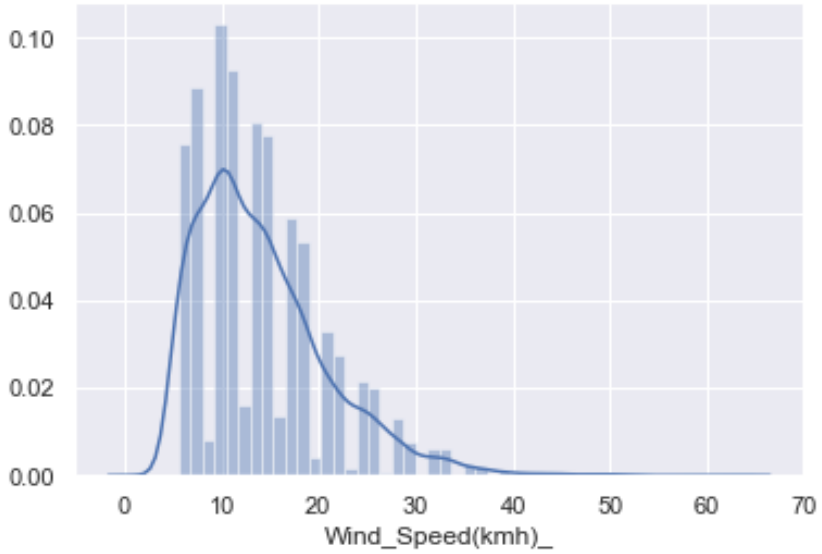

Figura 8 Histograma para la variable Wind_Speed(kmh).

Viendo que los datos están centrados sobre el valor **10**, con una cola por la derecha bastante larga que es extiende hasta cerca de **70**.

En la Tabla 8, se muestra algunos de los descriptivos de los datos:

Descriptivo	Valor
count	299999
mean	14.13
std	6.84
min	1.93
0.25	9.33
0.5	13.04
0.75	17.64
max	62.93

Tabla 8 Descriptivos estadísticos de la variable Wind_Speed(kmh).

Parecido al caso anterior, se observa un valor extremo bastante distinto a la media y a la desviación estándar, por lo que se decide nuevamente listar los **5** valores más grandes de la variable en la Tabla 9:

ID	Wind_Speed(kmh)_	Δ kmh (%)
A-2039135	62.93	9.52%
A-1359139	57.45	10.87%
A-619055	51.82	3.54%
A-912389	50.05	0.00%
A-1578248	50.05	0.00%

Tabla 9 Variación de la variable Wind_Speed(kmh).

Y observamos que los saltos no son tan grandes, por lo que es decide mantener el valor.

Finalmente se repite el procedimiento para la variable Precipitation(cm), obteniendo el *boxplot* de la Figura 9:

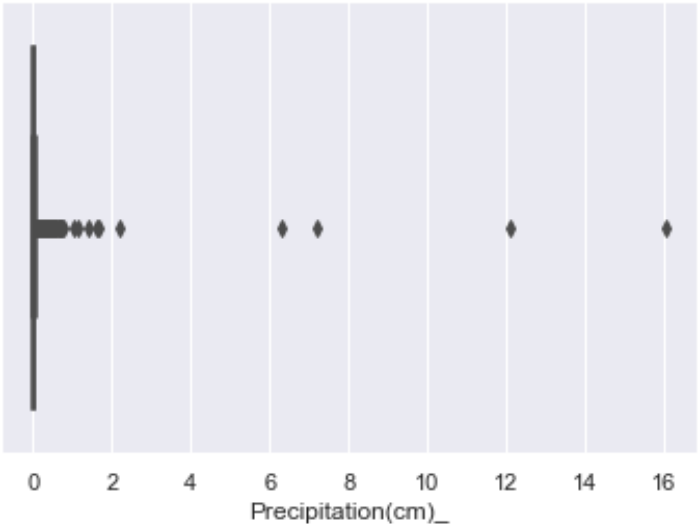

Figura 9 Boxplot para la variable Precipitation(cm).

Se observa inicialmente una distribución muy centrada sobre el valor **0**, se procede a representar el en la Figura 10 para obtener más conclusiones:

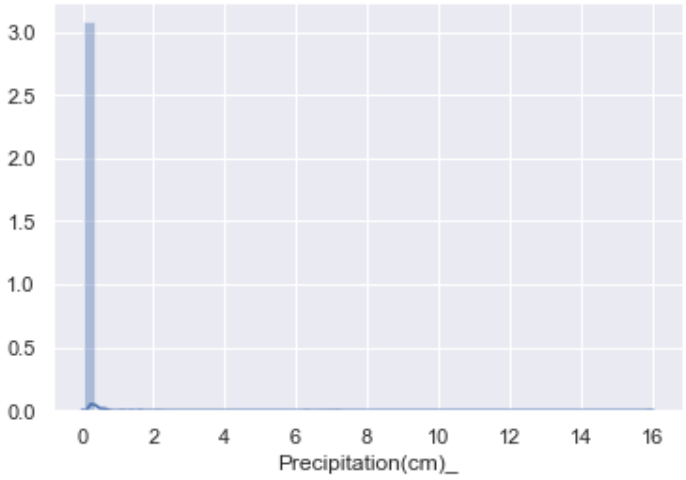

Figura 10 Histograma para la variable Precipitation(cm).

Se confirman las asunciones hechas anteriormente, no obstante, dada la propia naturaleza de la variable no se tiene porqué tratar de *outliers*, pues muchos accidentes pueden seguir produciéndose sin que llueva.

En la Tabla 10 se muestran algunos descriptivos de la variable:

Descriptivo	Valor
count	299999
mean	0.04
std	0.32
min	0.00
25	0.00
50	0.01
75	0.03
max	16.05

Tabla 10 Estadísticos descriptivos de la variable Precipitation(cm).

A pesar de que los valores máximo está bastante lejos de la media y la desviación se decide observar los valores máximos del juego de datos en la Tabla 11:

ID	Precipitation (cm)_	Δ cm (%)
A-376294	16.05	32.72%
A-699407	12.09	67.35%
A-2206228	7.22	14.75%
A-1189963	6.30	187.62%
A-886274	2.19	0.00%

Tabla 11 Valores máximos de la variable Precipitation (cm).

Se decide, viendo que la variación no es excesivamente grande mantener el valor.

Análisis de datos

Uno de los objetivos de este análisis es ver qué factores dan lugar a un accidente, inicialmente vamos a separar y analizar los accidentes por su grado de gravedad, *severity*. Se realiza un conteo de cada grupo en la Figura 11:

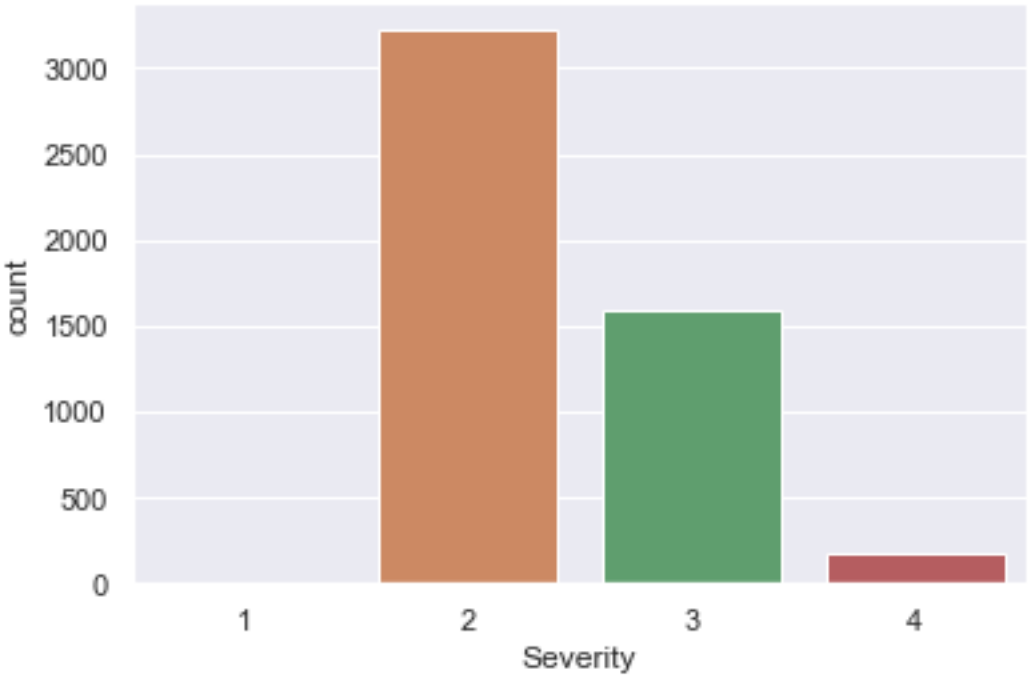

Figura 11 Histograma de Severity.

Podemos ver que la mayoría de accidente se concentran en los valores intermedios, mientras que para la clase $\mathbf{1}$ apenas hay datos.

En la Figura 12 se puede apreciar la evolución temporal para cada tipo de accidente:

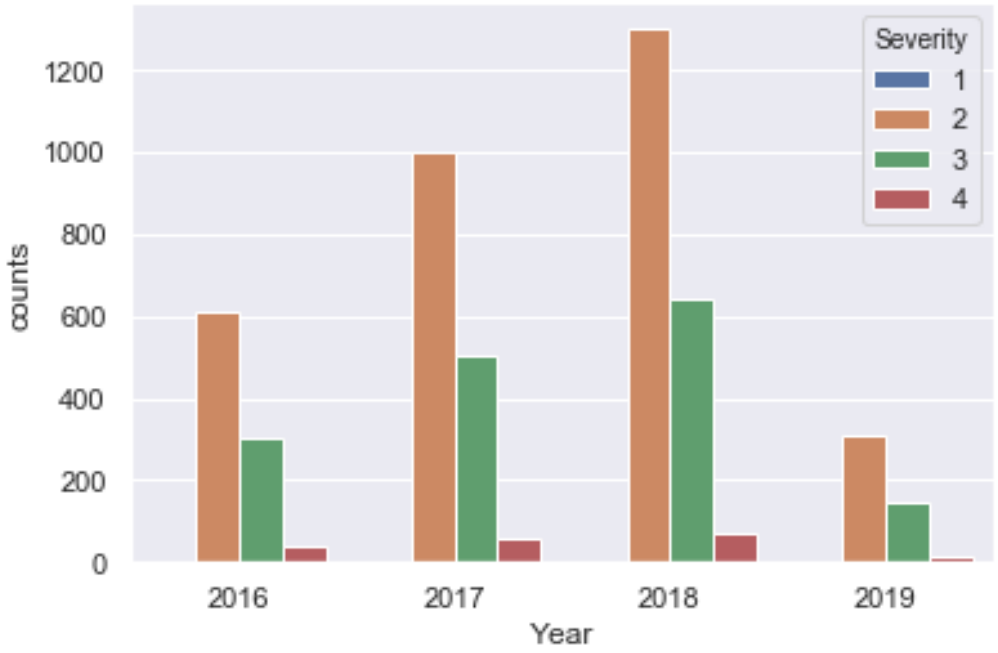

Figura 12 Evolución temporal de los accidentes según su severidad.

Se puede observar que para la mayoría de los accidentes se concentran en el grupo **2**, y que sus valores máximos están todos en el año **2018** para todos los subgrupos.

A continuación, en la Figura 13 se puede observar para cada estado cual el número de accidentes:

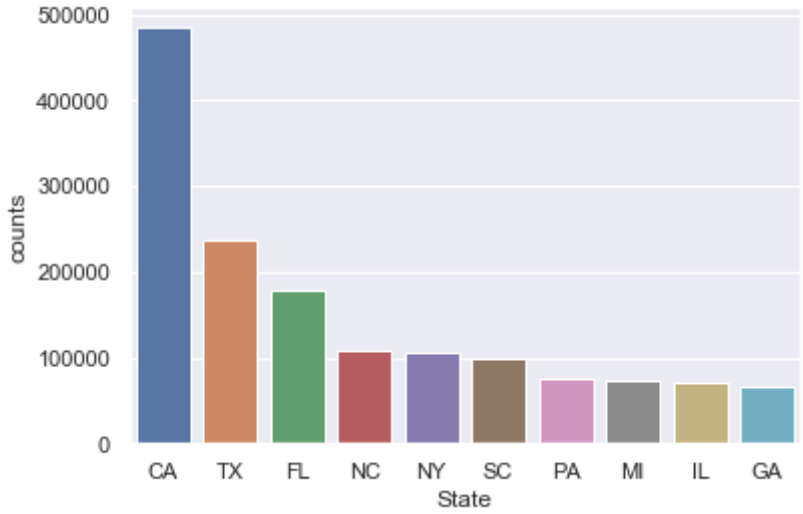

Figura 13 Número de accidentes por estado

Siendo el estado de California el más accidentado se estudia la gravedad, *Severity*, de los accidentes en la Figura 14:

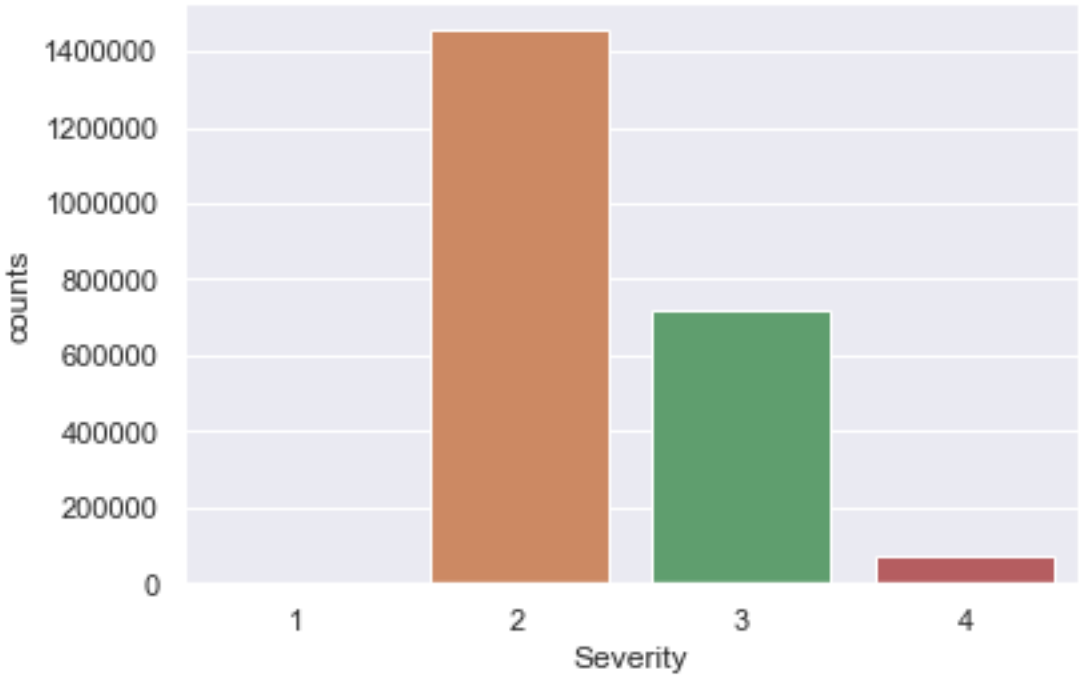

Figura 14 Severidad del accidente en California.

Observando que sigue la tendencia global del juego de datos ,siendo el caso de *Severity* **2** el más común.

Otro dato que nos parece relevante para el estudio es si el accidente ha ocurrido de día o noche. Esto viene dado en la columna *Civil_Twilight*, y se representan los resultados en la Figura 15:

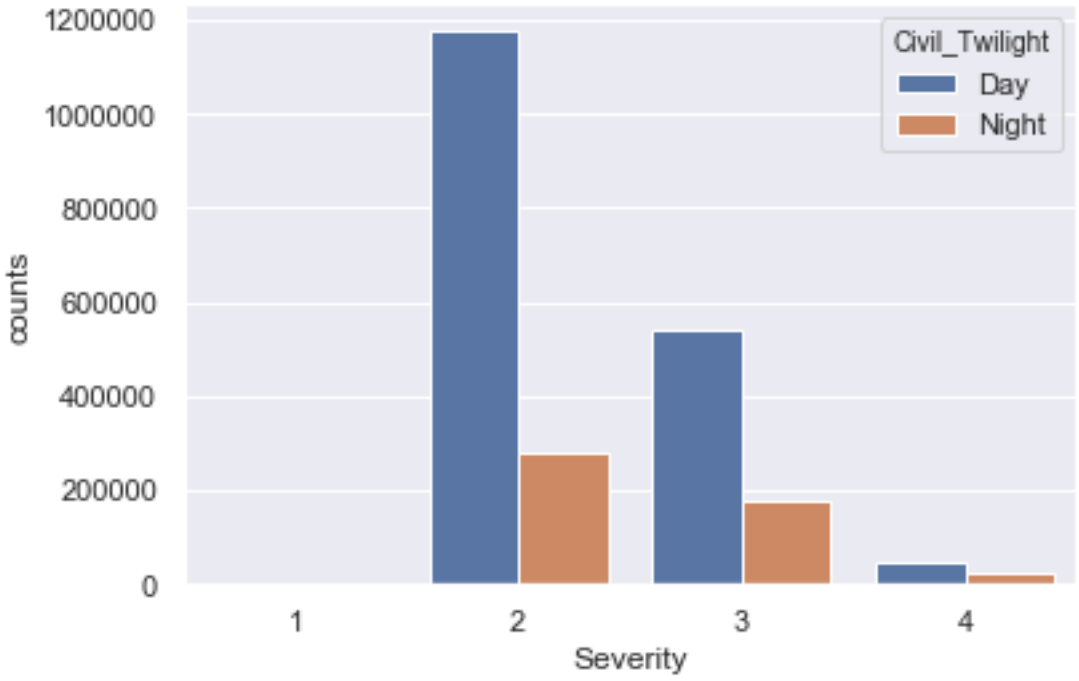

Figura 15 Valores de Civil_Twilight para accidentes.

Se puede ver que es bastante más probable sufrir un accidente al conducir de día que de noche para cualquier tipo de accidente.

Otro factor a tener en cuenta es el tiempo que hace, para hacer esto se han representado para cada *Severity* la meteorología en ese momento en la Figura 16:

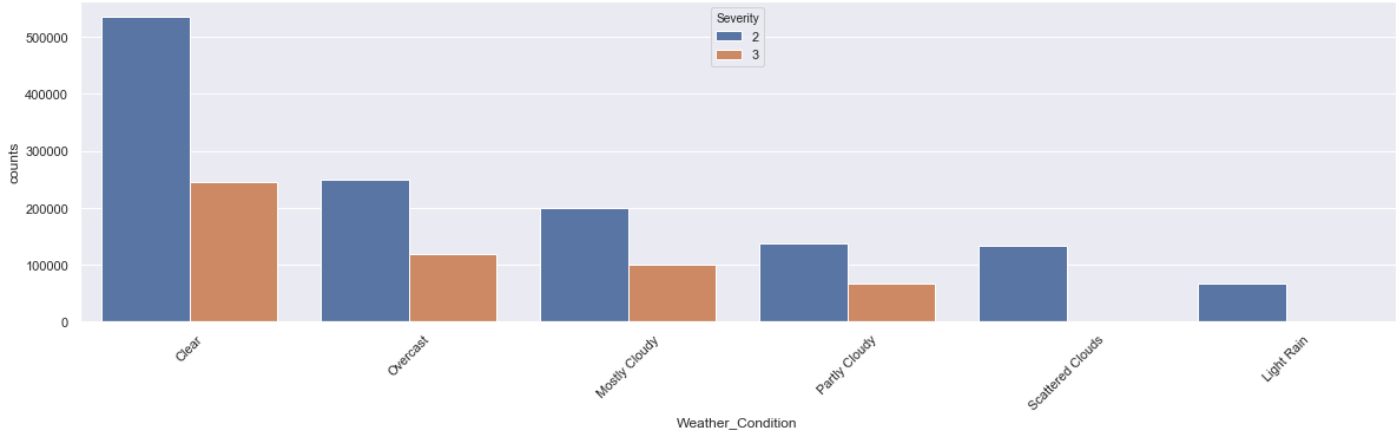

Figura 16 Top 5 causes meteorológicas.

Comprobación de normalidad y homogeneidad de la varianza

A continuación, para las columnas que se han estudiado hasta el momento se va a utilizar la estadística inferencial para asumir si a partir de la población escogida se pueden hacer asunciones sobre la población global.

La primera prueba será el test de Shapiro-Wilk **[3]** este toma como hipótesis nula que los datos de una población $\boldsymbol{X = \{x_1, x_2, \ldots, x_n\}}$ proviene de una población normal basándonos en un valor de α.

En el siguiente snippet de código se muestra la prueba que se ha hecho para realizar el test, junto los resultados que imprime:

```
from scipy.stats import shapiro

for col in COLS_NULL:
    stat, p = shapiro(imputed_vals[col])
    alpha = 0.05
    print('La variable {} retorna un p-value de {}'.format(col,p))
    if p > alpha:
        print('No podemos rechazar H0')
    else:
        print('Podemos rechazar H0')
```

El resultado, de todas las variables retorna valores exponenciales negativos del orden de $\boldsymbol{r \cdot x10^{-24}}$, siendo mucho más pequeños que nuestro valor de $\boldsymbol{\propto = 0,05}$ podemos rechazar la hipótesis nula, que sigue una distribución normal.

Esta última conclusión, se podía deducir ya un poco de los análisis hechos anteriormente.

El siguiente test que se realiza es el de Fligner-Killeen **[4]**. Este test toma como hipótesis nula que la varianza entre dos variables es la misma. Para realizar este test se realizan todas las combinaciones posibles a pares entre las variables que se han escogido para estudiar en este ejercicio, en el siguiente *snippet* de código se presenta la implementación:

```
import itertools
from scipy.stats import fligner

for a in zip(list(itertools.combinations(COLS_NULL, 2))):
    alpha = 0.05
    print('Comparando para variables {} y {}'.format(a[0][0],a[0][1]))
    stat,p=fligner(imputed_vals[a[0][0]],imputed_vals[a[0][1]])
    print('Valor de p {}'.format(p))
    if p > alpha:
        print('No podemos rechazar H0')
    else:
        print('Podemos rechazar H0')
```

Los resultados se presentan a continuación en la Tabla 12:

Variable_1	Variable_2	Rechazo H0
Pressure(cm)_	Visibility(km)_	Sí
Pressure(cm)_	Distance(km)_	Sí
Pressure(cm)_	Wind_Speed(kmh)_	Sí
Pressure(cm)_	Precipitation(cm)_	Sí
Visibility(km)_	Distance(km)_	No
Visibility(km)_	Wind_Speed(kmh)_	Sí
Visibility(km)_	Precipitation(cm)_	Sí
Distance(km)_	Wind_Speed(kmh)_	Sí
Distance(km)_	Precipitation(cm)_	Sí
Wind_Speed(kmh)_	Precipitation(cm)_	Sí

Tabla 12 Resultados del test de varianza.

Por lo que únicamente para la relación entre *Visibility(km) y Wind_Speed(kmh)* no podemos rechazar hipótesis de que la varianza es la misma.

Pruebas estadísticas

Se vuelve a las variables numéricas analizadas en el apartado anteriormente, para ver cómo se relacionan entre ellas mediante el gráfico de la Figura 17 mediante un *heatmap* de la correlación:

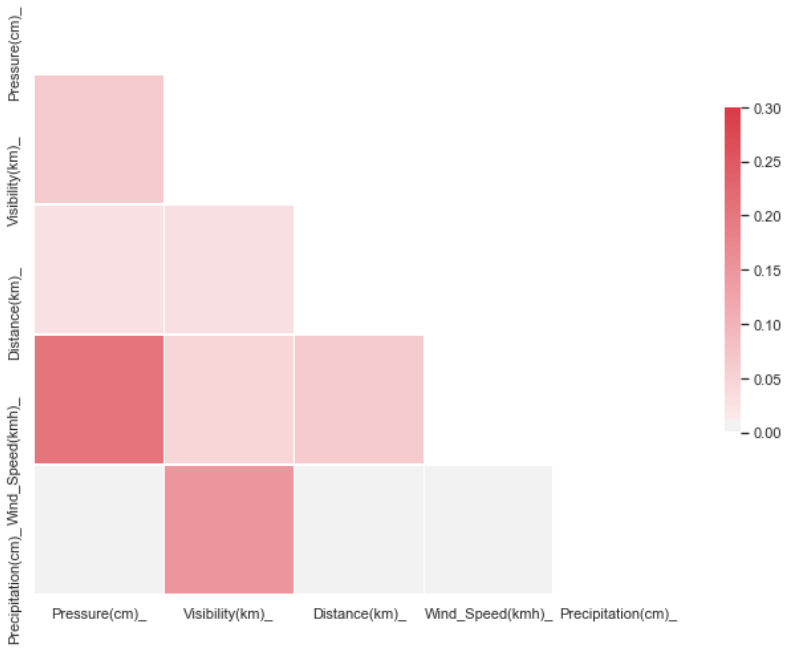

Figura 17 Heatmap de las variables numéricas.

Se puede observar que las variables *Visibility(km) junto con Wind_Speed(kmh)* están más fuerte relacionadas que el resto.

El gráfico anterior se ha realizado con el siguiente *snippet*:

```
sns.set(style="white")

pair_plot_df_no_sev=pair_plot_df.drop(columns=['Severity'])

corr = abs(pair_plot_df_no_sev.corr())

mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(11, 9))

cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

A continuación, se pretender generar un modelo, un regresor lineal para ver como se comporta con el conjunto de datos al clasificar la *Severity* del accidente basándonos en las columnas atributos elegidas.

Una vez elegidas las columnas aplicamos un *HotLabelEncoder* **[5]** a la variable objetivo para aclarar al algoritmo que se trata de una clase, y no de un valor numérico continúo a clasificar:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
complete_df_ml=complete_df[['Pressure(cm)_','Visibility(km)_','Distance(km)
_','Wind_Speed(kmh)_','Precipitation(cm)_']]

y=le.fit_transform(complete_df.loc[:,['Severity']].fillna(value='Severity')
)
```

Previamente a empezar a trabajar con el modelo, se divide el conjunto de datos en entreno y *test* siguiendo una relación $80/20$:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(complete_df_ml,y,test_size=.20,r
andom_state=0)
```

Finalmente aplicamos el algoritmo y obtenemos una exactitud de $0{,}633$:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(xtrain,ytrain)
pred=lr.predict(xtest)
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest,pred))
```

Pasamos a representar la matriz de confusión de los resultados en la Figura 18:

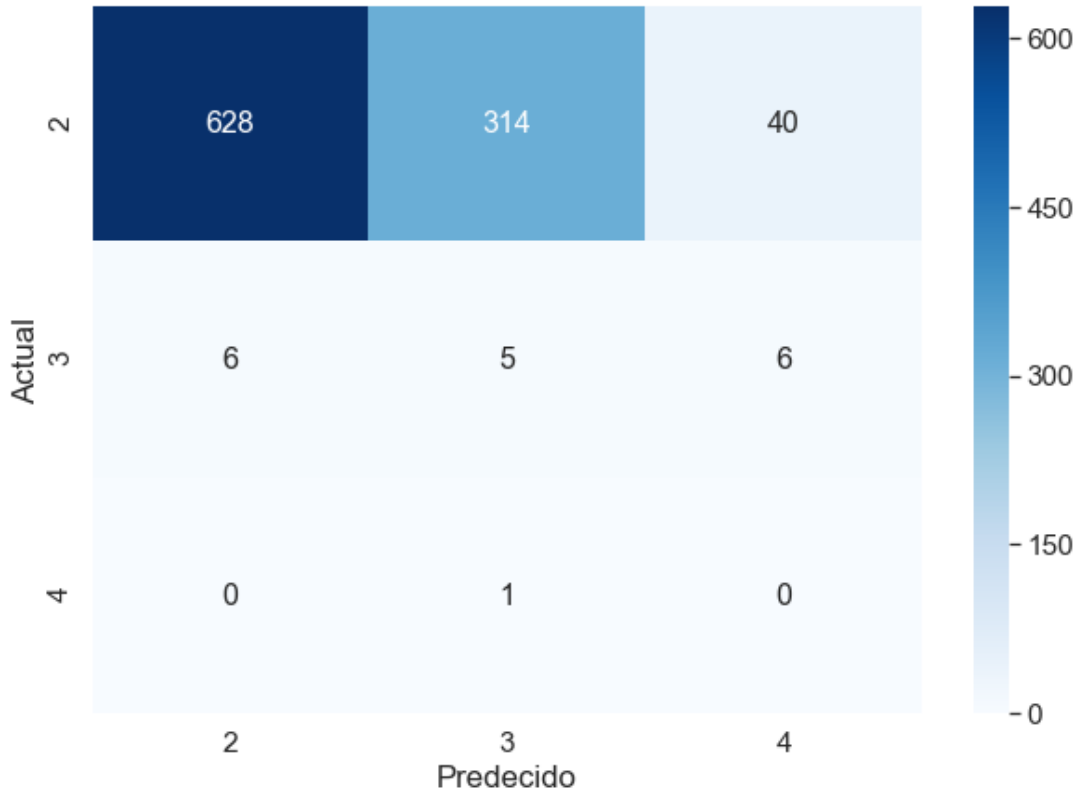

Figura 18 Matriz de confusión para el regresor lineal.

Lo primero que podemos observar es que no se ha clasificado ningún accidente como *Severity* 1. Esto es debido a lo poco representados que están en el conjunto. Después podemos ver que la clase que mejor se ha clasificado es la 2, esto es por el caso contrario al mencionado anteriormente, se disponen de más casos y el regresor lineal puede entender mejor las relaciones que llevan a estos accidentes.

Se procede, con el fin de mejorar los resultados, a aplicar otro tipo de modelo. En este caso se usa un *RandomTreeClassifier*, ayudándonos de la funcionalidad GridSearch(https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) para encontrar los hiperparámetros óptimos para el modelo, usando los siguientes parámetros:

I. Max_depth: Valores entre $[7 - 12]$.

II. N_estimators: Valores $[10,50,100,200,400]$.

Aplicamos el código:

```
from sklearn.model_selection import GridSearchCV


### desactivando futurewarnings por motivos estéticos del informe
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
###
param_grid = {'max_depth': np.arange(7,12),
              'n_estimators' : np.array([10,50,100,200,400])}
ran_forest = RandomForestClassifier(random_state=0)
gs_ran_forest=GridSearchCV(ran_forest,param_grid,cv=4)
gs_results_ran_forest=gs_ran_forest.fit(xtrain,ytrain)

init_pd=pd.pivot_table(pd.DataFrame(gs_results_ran_forest.cv_results_),valu
es='mean_test_score',index='param_max_depth',columns='param_n_estimators')


cmap = sns.cubehelix_palette(light=1, as_cmap=True, start=2, rot=0, dark=0,
reverse=False)
best_params=gs_results_ran_forest.best_params_
sns.heatmap(init_pd,cmap=cmap,annot=True)
```

Que nos genera la matriz de exactitud de la Figura 19:

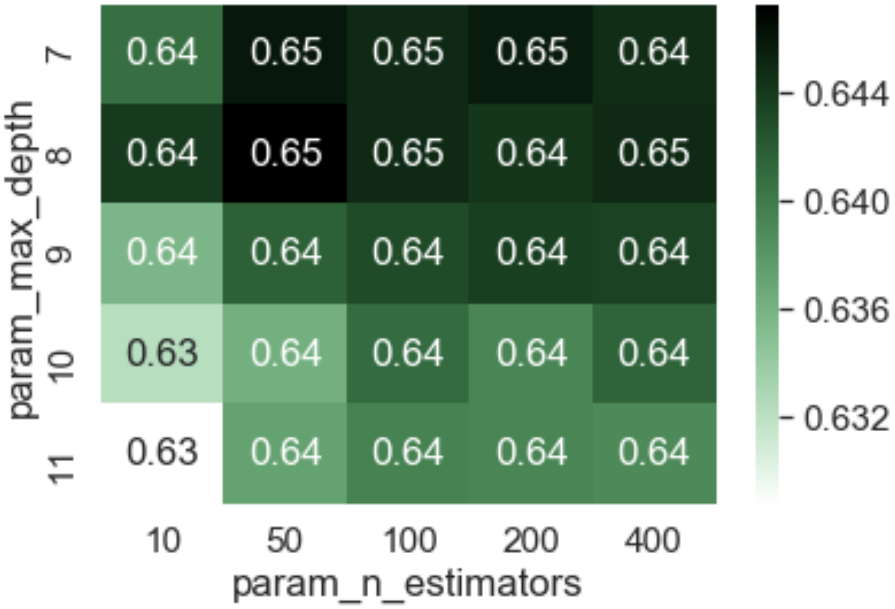

Figura 19 Matriz de resultados para los hiperparámetros.

Podemos observar que a partir de los estimadores *param_n_estimators* 50 *y param_max_depth* 8 el algoritmo se queda estancado a una exactitud de $0{,}65$.

Conclusiones

A lo largo del proceso de los datos hemos podido ir viendo los diferentes problemas que se tienen que afrontar al trabajar con datos: Hemos tenido que detectar los valores nulos, encontrar y analizar los *outliers*.
Debido a las dimensiones del juego de datos $2{,}5$ millones se ha tenido que reducir para llevar a cabo ciertas operaciones a lo largo del estudio.
Otro problema que hemos encontrado es la representación de las clases objetivo en el juego de datos. De los diferentes valores de *Severity* hemos visto que las clases están representadas de modo desigual. La existencia de literatura para tratar con estos casos es extensa **[6]**.
Se ha observado también durante la fase exploratoria que la mayoría de accidentes vienen condicionados por el estado en el que se conduce en vez de , por ejemplo, si se conduce de día o noche.
Lamentablemente también se ha observado que , obviando el 2019 ya que el año no ha terminado, desde el 2016 ha habido una tendencia de aumento en los accidentes para todas las clases.

Referencias

[1] "Kaggle," Kaggle USA Accidents, 2019. [Online]. Available: https://www.kaggle.com/sobhanmoosavi/us-accidents/version/1 .

[2] FancyImpute, "Pypi," FancyImpute, 2019. [Online]. Available: https://pypi.org/project/fancyimpute/.

[3] Wikipedia, "Shapiro-Wilk," Shapiro-Wilk Test, [Online]. Available: https://es.wikipedia.org/wiki/Test_de_Shapiro%E2%80%93Wilk.

[4] W. J. Conover, "Ethz," Fligner-Killeen Test, [Online]. Available: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/fligner.test.html.

[5] LabelEncoder, "SkLearn," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

[6] B. Rocca, "Handling imbalanced datasets in machine learning," towardsdatascience, 2017. [Online]. Available: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28.

Integrantes

Contribuciones	Firma
Investigación previa	**Alejandro Martínez Otal**
Redacción de las respuestas	**Alejandro Martínez Otal**
Desarrollo código	**Alejandro Martínez Otal**