

Big Data

MEMORIA FINAL DEL PROYECTO

Pablo Schneegluth
TECNOCAMPUS-MATARÓ | 4T CURS

Índice

<i>Introducción</i>	3
<i>Contexto, datasets elegidos y justificación del proyecto</i>	3
<i>Definición de los objetivos a cumplir.....</i>	4
<i>Inspección, tratamiento y análisis inicial de datos</i>	4
Valores Nulos.....	4
Outliers.....	7
Distribución de los datos.....	9
Matriz de Correlación	11
<i>PCA</i>	12
<i>Algoritmos de aprendizaje Supervisado</i>	13
Regresión - Regresión Linear	13
Regresión - Regresión Polinomial.....	14
Clasificación – K-Nearest Neighbours (KNN)	16
<i>Algoritmos de aprendizaje No Supervisado.....</i>	18
Clustering – Clustering Jerárquico	18
Clustering – K-Means	19
<i>Conclusiones.....</i>	22

Introducción

Querría estudiar los datos relativos a accidentes de tráfico con fallecidos o heridos graves que se han producido en Catalunya des del año 2010. Estaría interesante saber predecir con bastante precisión que, dados unos parámetros, se pueda saber qué tipo de accidente sucederá y con qué consecuencias acabará.

Contexto, datasets elegidos y justificación del proyecto

El dataset contiene datos de los diferentes accidentes producidos en Cataluña des del 2010, con todo tipo de parámetro descriptivo. Entre ellos la fecha, municipio, número de fallecidos y/o heridos, velocidad de la vía, etc.

El dataset escogido se ha obtenido de este enlace:
<https://analisi.transparenciacatalunya.cat/Transport/Accidents-de-tr-nsit-amb-morts-o-ferits-greus-a-Ca/rmgc-ncpb>

Me interesa mucho este dataset que contiene muchos datos sobre accidentes, principalmente porque es una causa de muerte importante. Como mucha gente utiliza transporte privado, está bien saber sobre los riesgos que conlleva conducir. Personalmente conduzco una motocicleta, y justamente hay una columna que determina si ha habido alguna motocicleta involucrada en el accidente, así que también me servirá para saber los riesgos de ir en motocicleta en comparación a otros vehículos. Con estos datos estaré muy interesante el intentar predecir según algunos parámetros.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Any			pk		nomCom	nomDem	F_MORTS	F_FERITS_GIF_FERITS_LL_F_VICTIMS_F_VIANANTS_F_BICICLETF_COLCLOMOTF_MOTOCOF_VEH_LL_EUF_VEH_PESA_F_ALTRAS_UF_UNIT_DESC_VELOCITAD_AC_AMI														
2	2010	Zona urbana	25/1/10 SE		999999 CANVIARS I S'ESTRUCTURA TERRITORIAL	0	1	0	1	2	0	0	0	0	0	1	0	1	0	1	100 No	
3	2010	Carretera	31/1/10 N-240		99.9	SEGUÍ Ulldecona	0	1	3	4	1	0	0	0	0	1	0	0	0	0	40 No	
4	2010	Carretera	17/5/10 N-II	708.7	FORNELLS D Gironès	Girona	1	0	2	3	4	0	0	0	0	2	2	0	0	0	80 No	
5	2010	Zona urbana	21/1/10 SE		999999 BARCELONA Barcelones	Barcelona	0	2	7	9	2	0	0	0	0	2	0	0	0	0	100 No	
6	2010	Zona urbana	7/5/10 SE		999999 BARCELONA Barcelones	Barcelona	0	1	0	1	1	0	0	0	1	0	0	0	0	0	100 No	
7	2010	Zona urbana	16/4/10 SE		999999 SANT CARLES MOLLET	Tarragona	0	1	1	2	2	0	0	0	1	0	1	0	0	0	40 No	
8	2010	Zona urbana	13/1/10 SE		999999 BARCELONA Barcelones	Barcelona	0	1	0	1	2	0	0	0	0	1	1	0	0	0	100 No	
9	2010	Zona urbana	23/10/10 SE		999999 BARCELONA Barcelones	Barcelona	1	0	1	2	1	0	0	0	1	0	0	0	0	0	100 No	
10	2010	Carretera	19/6/10 AP-7		138 MOLLET DEL VALLS Orient	Barcelona	0	1	2	3	2	0	0	0	0	0	2	0	0	0	0	100 No
11	2010	Carretera	12/7/10 C-31		999999 CERDANYOLA DEL VALLES	Barcelona	0	1	2	3	2	0	0	0	0	1	0	0	0	0	0	100 No
12	2010	Zona urbana	36/10/10 C-31		350 CERDANYOLA DEL VALLES	Barcelona	0	1	1	2	3	2	0	0	0	1	0	0	0	0	0	100 No
13	2010	Zona urbana	12/7/10 SE		999999 OSONA	Barcelona	0	1	0	1	2	0	0	1	0	1	0	0	0	0	30 No	
14	2010	Carretera	23/5/10 N-II	719.4	GIRONA	Girona	0	1	0	1	2	0	0	0	0	1	1	0	0	0	80 No	
15	2010	Zona urbana	17/1/10 SE		999999 BARCELONA Barcelones	Barcelona	0	1	0	1	2	0	0	0	0	1	1	0	0	0	100 No	
16	2010	Zona urbana	13/1/10 SE		999999 TARRAGONA	Tarragona	0	1	0	1	2	1	0	0	0	1	0	0	0	0	100 No	
17	2010	Zona urbana	20/6/10 SE		999999 CLOU	Girona	0	1	1	2	3	1	0	0	0	1	1	0	0	0	30 No	
18	2010	Zona urbana	15/9/10 SE		999999 LLEIDA	Segrià	0	1	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
19	2010	Carretera	28/7/10 B-522	6.1	MANELLI	Osona	0	1	2	3	3	0	0	0	0	0	3	0	0	0	100 No	
20	2010	Zona urbana	27/7/10 SE		999999 BARCELONA Barcelones	Barcelona	1	0	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
21	2010	Zona urbana	20/7/10 C-54	162.8	999999 TERRASSA Vallès Occidental	Tarragona	0	0	5	5	2	0	0	0	0	5	0	0	0	0	100 No	
22	2010	Carretera	24/8/10 G-673	1.5	CAUSES DE LA Selva	Girona	0	1	1	2	2	0	0	0	0	2	0	0	0	0	100 No	
23	2010	Carretera	31/1/10 LV-3021		1 ARTESA DE Segura	Urgell	0	1	0	1	1	0	0	0	0	1	0	0	0	0	No NA No	
24	2010	Zona urbana	19/10/10 SE		999999 MARTORELL Baix Llobregat	Barcelona	0	1	1	2	2	0	0	0	0	1	1	0	0	0	40 No	
25	2010	Carretera	28/5/10 C-17	39.5	999999 TERRASSA Vallès Occidental	Tarragona	0	1	0	1	1	0	0	0	0	0	1	0	0	0	80 No	
26	2010	Zona urbana	8/7/10 SE		999999 TERRASSA	Vallès Occidental	0	1	0	1	2	0	0	0	0	1	1	0	0	0	100 No	
27	2010	Zona urbana	31/1/10 SE		999999 MOLLET DEL VALLS Orient	Barcelona	0	1	0	1	1	0	0	0	0	0	1	0	0	0	100 No	
28	2010	Carretera	25/9/10 SE		999999 PRAT DE LLO Baix Llobregat	Barcelona	1	0	1	0	1	0	0	0	0	0	1	0	0	0	100 No	
29	2010	Zona urbana	23/6/10 SE		999999 FUJOLA La Urgell	Urgell	1	0	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
30	2010	Zona urbana	24/7/10 SE		999999 BARCELONA Barcelones	Barcelona	0	1	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
31	2010	Zona urbana	31/7/10 SE		999999 L'HOSPITALET DE Llobregat	Barcelona	0	1	1	2	2	1	0	0	0	0	0	0	0	0	100 No	
32	2010	Zona urbana	12/7/10 SE		999999 ROQUETES Baix Ebre	Tarragona	0	1	0	1	1	0	0	0	0	0	1	0	0	0	No NA No	
33	2010	Zona urbana	14/9/10 TP-7225	8.1	MORELL El Tarragonès	Tarragona	0	1	1	0	3	0	0	0	1	0	2	0	0	0	50 No	
34	2010	Zona urbana	25/7/10 SE		999999 BARCELONA Barcelones	Barcelona	0	1	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
35	2010	Zona urbana	20/7/10 AL	3.3	999999 TERRASSA Vallès Occidental	Tarragona	0	2	0	2	2	0	0	0	1	1	0	0	0	30 No		
36	2010	Carretera	2/7/10 C-1415B	8.4	LLUÇÀ D'AMÍ Valles Orient	Barcelona	0	1	1	2	2	0	0	0	1	1	0	0	0	100 No		
37	2010	Zona urbana	11/10/10 SE		999999 BARCELONA Barcelones	Barcelona	0	2	2	4	6	0	0	0	1	4	0	1	0	0	100 No	
38	2010	Carretera	3/10/10 C-55	10.7	COLLBATÓ Baix Llobregat	Barcelona	1	1	1	3	4	0	0	0	0	4	0	0	0	100 No		
39	2010	Zona urbana	14/4/10 SE		999999 TERRASSA Vallès Occidental	Tarragona	0	1	0	1	2	1	0	0	0	0	1	0	0	0	100 No	
40	2010	Zona urbana	24/9/10 SE		999999 TERRASSA Vallès Occidental	Tarragona	0	1	0	1	3	0	0	0	1	2	0	0	0	100 No		
41	2010	Carretera	4/7/10 RR-2121	8.2	SANT MARTÍ Alt Penedès	Barcelona	0	2	0	2	1	0	0	0	0	1	0	0	0	70 No		
42	2010	Zona urbana	14/7/10 SE		999999 ALELLA Maresme	Barcelona	0	1	0	1	2	0	0	0	1	1	0	0	0	No NA No		

Definición de los objetivos a cumplir

El principal objetivo de estudiar este dataset sería conocer el porcentaje de accidentes de cada tipo para conocer los que más peso tienen, las principales diferencias de los accidentes de motocicletas comparado con otros vehículos, las probabilidades que dados algunos parámetros ocurra algún tipo de accidente, que causas hacen que unos accidentes sean más mortales que otros, etc.

Inspección, tratamiento y análisis inicial de datos

Para el tratamiento de datos procederé a hacerlo en el lenguaje de Python.

Primero de todo vamos a cargar el archivo en una variable.

In [10]: import pandas as pd df = pd.read_csv('Accidentes.csv') df														
Out[10]:														
Any	zona	dat	via	pk	nomMun	nomCom	nomDem	F_MORTS	F_FERITS_GREUS	...	D_SUPERFICIE	D_TIPUS_VIA	D_TI	
0	2010	Zona urbana	25/01/2010	SE 999999.0	CANOVES I SAMALUS	Valles Oriental	Barcelona	0	1 ...	Sec i net	Via urbana(inclou carrer i carrer residencial)			
1	2010	Carretera	31/10/2010	N-240	99.9	LLEIDA	Segria	Lleida	0	1 ...	Sec i net	Carretera convencional		
2	2010	Carretera	17/05/2010	N-II	708.7	FORNELLS DE LA SELVA	Girones	Girona	1	0 ...	Sec i net	Carretera convencional		
3	2010	Zona urbana	21/08/2010	SE 999999.0	BARCELONA	Barcelones	Barcelona	0	2 ...	Sec i net	Via urbana(inclou carrer i carrer residencial)			
4	2010	Zona urbana	07/05/2010	SE 999999.0	BADALONA	Barcelones	Barcelona	0	1 ...	Sec i net	Via urbana(inclou carrer i carrer residencial)			
...
21156	2021	Zona urbana	21/12/2021	BV-2002	2.8	SANT VICENC DELS HORTS	Baix Llobregat	Barcelona	1	0 ...	Sec i net	Carretera convencional		
21157	2021	Zona urbana	08/11/2021	SE 999999.0		LLEIDA	Segria	Lleida	0	1 ...	Sec i net	Via urbana(inclou carrer i carrer residencial)		

Valores Nulos

Para la inspección de datos vamos a hacer profiling. Así que primero miramos si nuestro dataset contiene valores nulos que nos vayan mal para nuestro estudio.

```
In [5]: null_values = df.isnull()
null_counts = null_values.sum()
print(null_counts)

Any 0
zona 0
dat 0
via 0
pk 1
nomMun 0
nomCom 0
nomDem 0
F_MORTS 0
F_FERITS_GREUS 0
F_FERITS_LLEUS 0
F_VICTIMES 0
F_UNITATS_IMPLICADES 0
F_VIANANTS_IMPLICADES 0
F_BICICLETES_IMPLICADES 0
F_CICLOMOTORS_IMPLICADES 0
F_MOTOCICLETES_IMPLICADES 0
F_VEH_LLEUGERS_IMPLICADES 0
F_VEH_PESANTS_IMPLICADES 0
F_ALTRES_UNIT_IMPLICADES 0
F_UNIT_DESC_IMPLICADES 0
C_VELOCITAT_VIA 2642
D_ACC_AMB_FUGA 0
D_BOIRA 0
D_CARACT_ENTORN 32
D_CARRIL_ESPECIAL 1349
D_CIRCULACIO_MESURES_ESP 40
D_CLIMATOLOGIA 0
D_FUNC_ESP_VIA 0

D_GRAVETAT 0
D_INFLUIT_BOIRA 0
D_INFLUIT_CARACT_ENTORN 0
D_INFLUIT_CIRCULACIO 0
D_INFLUIT_ESTAT_CLIMA 0
D_INFLUIT_INTEN_VENT 0
D_INFLUIT_LLUMINOSITAT 0
D_INFLUIT_MESU_ESP 0
D_INFLUIT_OBJ_CALCADA 0
D_INFLUIT_SOLCS_RASES 0
D_INFLUIT_VISIBILITAT 0
D_INTER_SECCIO 0
D_LIMIT_VELOCITAT 0
D_LLUMINOSITAT 0
D_REGULACIO_PRIORITAT 14970
D_SENTITS_VIA 3496
D_SUBTIPUS_ACCIDENT 0
D_SUBTIPUS_TRAM 14212
D_SUBZONA 0
D_SUPERFICIE 0
D_TIPUS_VIA 0
D_TITULARITAT_VIA 10712
D_TRACAT_ALTIMETRIC 7637
D_VENT 0
grupDiaLab 0
hor 0
grupHor 0
tipAcc 0
tipDia 0
dtype: int64
```

Podemos ver que en algunas columnas contienen valores nulos, algunas más que otras. Para las columnas “pk”, “C_Velocitat_Via”, “D_Caract_Entorn”, “D_Carril_Especial”, “D_Circulacio_Mesures_Esp” y “D_Sentits_Via” eliminaremos los valores nulos para el correcto estudio de los datos y para la mayor exactitud posible ignorando los valores que no deberían estar. No tiene sentido estudiar esos valores siendo nulos.

```
In [29]: df = df.dropna(subset=['pk'])
df = df.dropna(subset=['C_VELOCITAT_VIA'])
df = df.dropna(subset=['D_CARACT_ENTORN'])
df = df.dropna(subset=['D_CARRIL_ESPECIAL'])
df = df.dropna(subset=['D_CIRCULACIO_MESURES_ESP'])
df = df.dropna(subset=['D_SENTITS_VIA'])
```

En cambio, en la fila “D_Regulacio_Prioritat” sí que tienen sentido los valores nulos, ya que, si no había ninguna señal de preferencia hacia ningún vehículo implicado, ese valor quedará vacío. Para ello vamos a substituir los valores nulos por la palabra “Cap”. Lo mismo haremos con la columna “D_Subtipus_Tram” que, si no coincide con ningún tipo de subtipo de tramo, se le cataloga con un valor nulo.

```
In [32]: df['D_REGULACIO_PRIORITAT'].fillna('Cap', inplace=True)
df['D_SUBTIPUS_TRAM'].fillna('Cap', inplace=True)
```

También eliminaremos las columnas que no nos aporten ningún tipo de información relevante o que no nos incumban para los objetivos que tengo marcados.

```
In [5]: #Eliminar columnas innecesarias
df = df.drop(['tipDia', 'pk', 'via', 'F_UNIT_DESC_IMPLICADES', 'D_FUNC_ESP_VIA', 'D_INFILUIT_BOIRA',
'D_INFILUIT_CARACT_ENTORN', 'D_INFILUIT_CIRCULACIO', 'D_INFILUIT_ESTAT_CLIMA', 'D_INFILUIT_INTEN_VENT',
'D_INFILUIT_LLUMINOSITAT', 'D_INFILUIT_MESU_ESP', 'D_INFILUIT_OBJ_CALCADA', 'D_INFILUIT_SOLCS_RASES',
'D_INFILUIT_VISIBILITAT', 'D_INTER_SECCIO', 'D_LIMIT_VELOCITAT', 'D_REGULACIO_PRIORITAT',
'D_SUBTIPUS_ACCIDENT', 'D_SUBTIPUS_TRAM', 'D_SUBZONA', 'D_TIPUS_VIA', 'D_TITULARITAT_VIA',
'D_TRACAT_ALTIMETRIC'], axis=1)
```

Así queda finalmente las filas que nos interesan con los valores limpios de nulos:

	0	0
Any	C_VELOCITAT_VIA	0
zona	D_ACC_AMB_FUGA	0
dat	D_BOIRA	0
nomMun	D_CARACT_ENTORN	0
nomCom	D_CARRIL_ESPECIAL	0
nomDem	D_CIRCULACIO_MEURES_ESP	0
F_MORTS	D_CLIMATOLOGIA	0
F_FERITS_GREUS	D_GRAVETAT	0
F_FERITS_LLEUS	D_LLUMINOSITAT	0
F_VICTIMES	D_SENTITS_VIA	0
F_UNITATS_IMPLICADES	D_SUPERFICIE	0
F_VIANANTS_IMPLICADES	D_VENT	0
F_BICICLETES_IMPLICADES	grupDiaLab	0
F_CICLOMOTORS_IMPLICADES	hor	0
F_MOTOCICLETES_IMPLICADES	grupHor	0
F_VEH_LLEUGERS_IMPLICADES	tipAcc	0
F_VEH_PESANTS_IMPLICADES		
F_ALTRES_UNIT_IMPLICADES	dtype: int64	

Eliminamos también, todas las filas que contengan el valor "Sense Especificar" ya que no nos aportan ningún tipo de valor.

```
mask = (df['D_CARRIL_ESPECIAL'] == 'Sense Especificar') | (df['D_CLIMATOLOGIA'] == 'Sense Especificar') |
(df['D_LLUMINOSITAT'] == 'Sense Especificar') | (df['D_SENTITS_VIA'] == 'Sense Especificar') |
|(df['D_SUPERFICIE'] == 'Sense Especificar') | (df['D_VENT'] == 'Sense Especificar')
df = df.drop(df[mask].index)
```

	142
Altres	142
Carril central	142
Carril d'alentiment	89
Carril lent	76
Carril avançament	41
Habilitació voral/carril addicional	35
Carril reversible	22
Carril habilitat en sentit contrari habitual	13
Sense especificar	1
Name: D_CARRIL_ESPECIAL, dtype: int64	
Bon temps	14349
Pluja débil	756
Pluja forta	220
Calamarsa	10
Nevant	8
Sense especificar	1
Name: D_CLIMATOLOGIA, dtype: int64	
De dia, dia clar	9727
De nit, il·luminació artificial suficient	1866
De nit, sense llum artificial	1391
Alba o capvespre	817
De nit, il·luminació artificial insuficient	810
de dia, dia fosc	731
Sense especificar	2
Name: D_LLUMINOSITAT, dtype: int64	
Doble sentit	11392
Un sol sentit	3882
Sense especificar	70
Name: D_SENTITS_VIA, dtype: int64	
Sec i net	14523
Mullat	591
Inundat	113
Relliscós	87
Gelat	21
Nevat	8
Sense especificar	1
Name: D_SUPERFICIE, dtype: int64	
Calma, vent molt suau	14989
Vent moderat	280
Vent fort	73
Sense especificar	2
Name: D_VENT, dtype: int64	

Outliers

Aquí podemos observar como la velocidad límite máxima es 999 km/h, lo que hace subir la media en prácticamente un punto entero en comparación a la media con los valores tratados.

Claramente la velocidad límite predominante son los 100 km/h.

Con el tratamiento de datos podemos ver que solamente omitimos 38 valores que consideramos outliers.

```
In [15]: #Valores estadisticos
maximo = df['C_VELOCITAT_VIA'].mean() + 3*df['C_VELOCITAT_VIA'].std()
minimo = df['C_VELOCITAT_VIA'].mean() - 3*df['C_VELOCITAT_VIA'].std()
print ("Normal Máximo: media + 3sigma", maximo)
print ("Normal Mínimo: media + 3sigma", minimo)
df['C_VELOCITAT_VIA'].describe()

Normal Máximo: media + 3sigma 201.33213664120507
Normal Mínimo: media + 3sigma -30.431355391205088

Out[15]: count    15360.000000
          mean     85.450391
          std      38.627249
          min      0.000000
          25%     70.000000
          50%     100.000000
          75%     100.000000
          max     999.000000
          Name: C_VELOCITAT_VIA, dtype: float64

In [14]: import numpy as np
new_df = df[(df['C_VELOCITAT_VIA'] < maximo) & (df['C_VELOCITAT_VIA'] > minimo)]
new_df['C_VELOCITAT_VIA'].describe()

Out[14]: count    15322.000000
          mean     84.607362
          std      24.797179
          min     15.000000
          25%     70.000000
          50%     100.000000
          75%     100.000000
          max     120.000000
          Name: C_VELOCITAT_VIA, dtype: float64
```

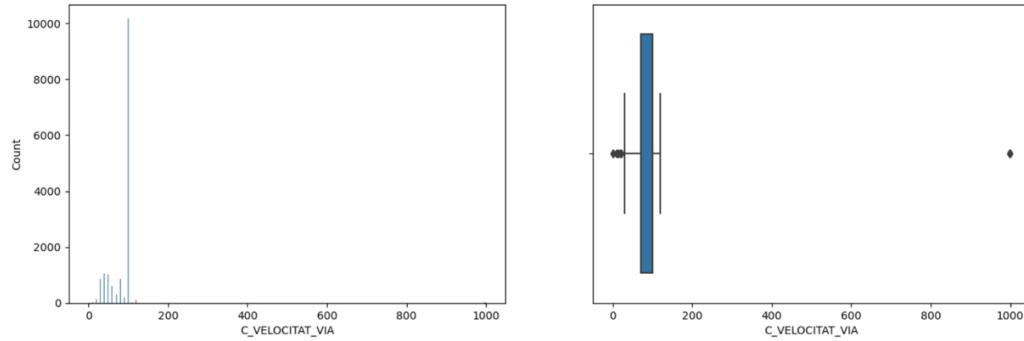
Se puede apreciar en la siguiente imagen que la distribución de la variable predictora de "C_Velocitat_Via" es asimétrica. Contamos con una mayoría de valores que son 100 km/h.

He mirado si hay outliers en la variable predictora de "C_Velocitat_Via" y sí que se pueden apreciar. Lo que no se sabe bien cómo se han registrado esos valores, ya que no tienen ningún sentido que el límite de velocidad límite de una vía sea de 999 km/h.

```
In [7]: #Outliers
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

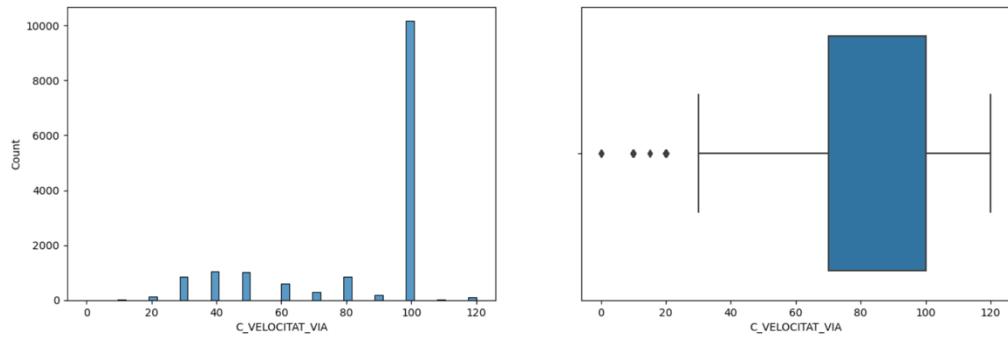
warnings.filterwarnings('ignore')

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.histplot(df['C_VELOCITAT_VIA'])
plt.subplot(1,2,2)
sns.boxplot(df['C_VELOCITAT_VIA'])
plt.show()
```



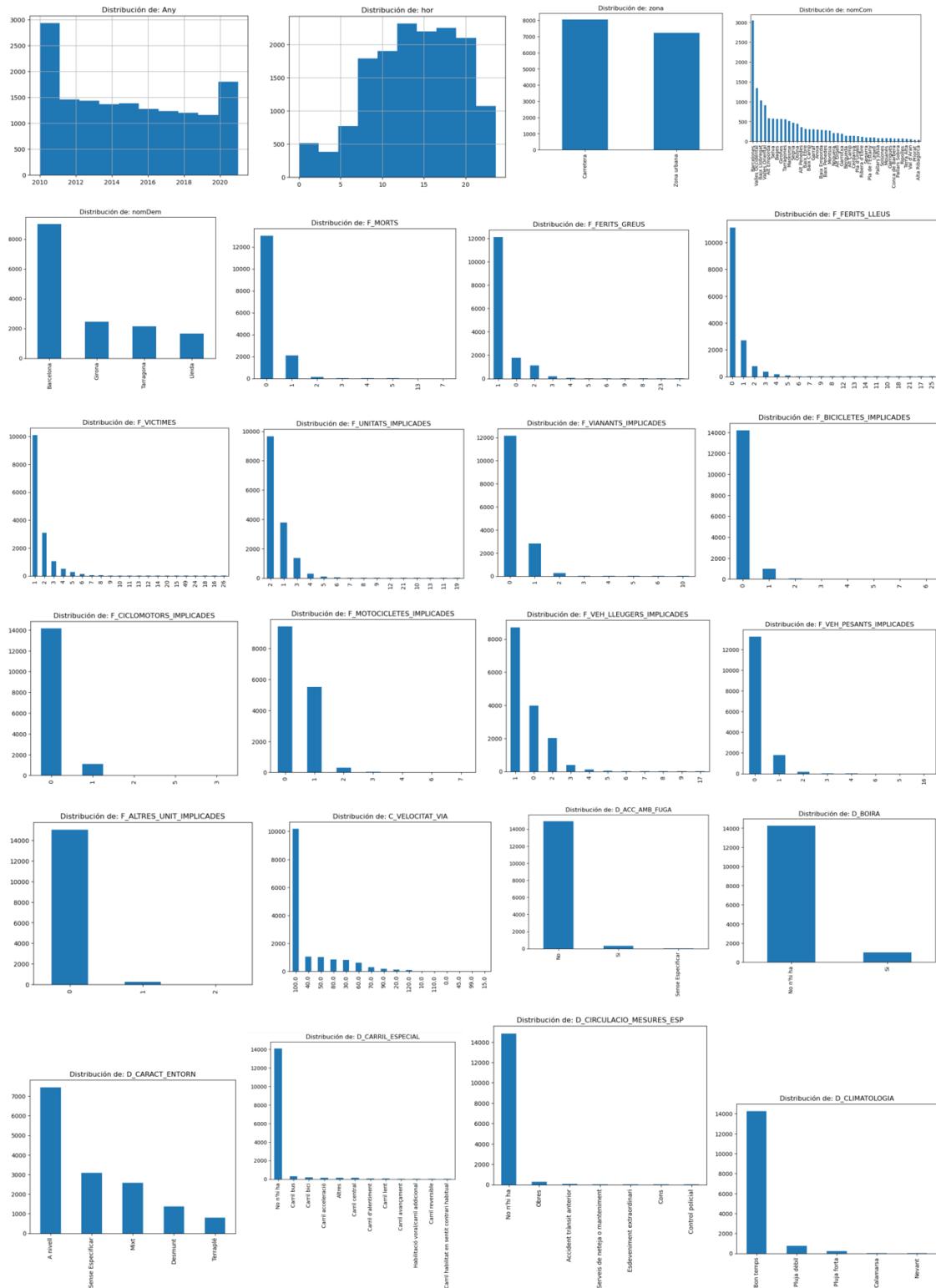
Hacemos el tratamiento de estos outliers y vemos como los valores ya se normalizan y tienen unos límites coherentes para lo que es la velocidad límite de una vía. Y con el límite de 120 km/h cómo es realmente en España.

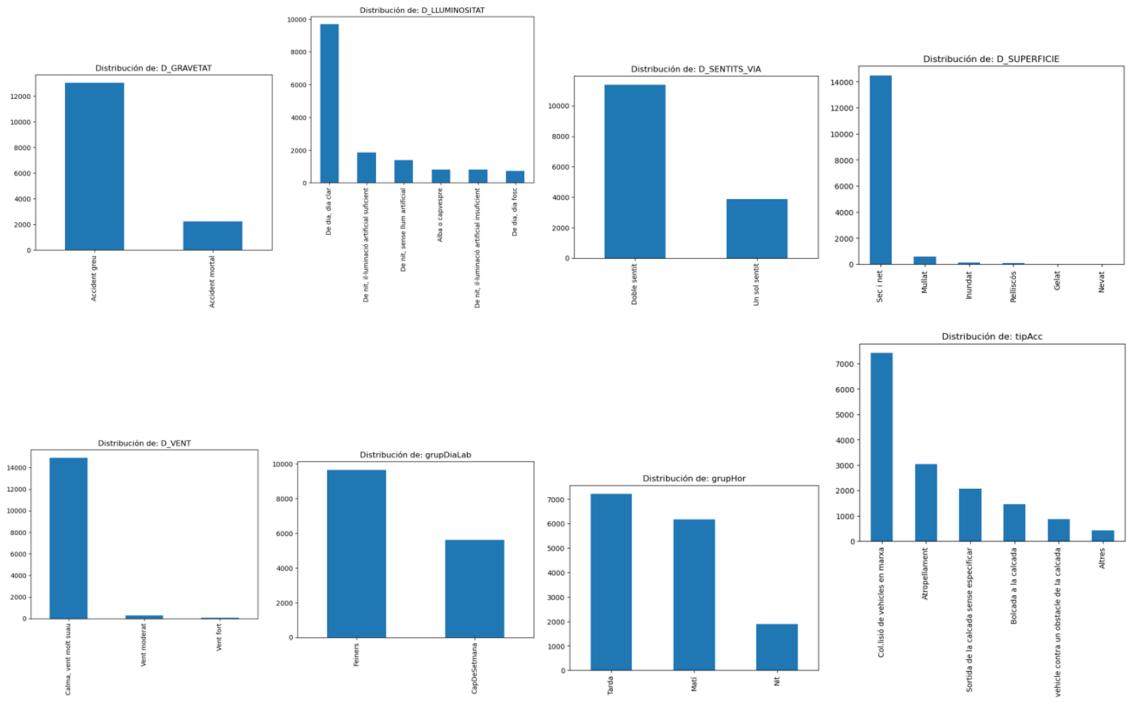
```
In [16]: #Outliers procesados
new_df = df[(df['C_VELOCITAT_VIA'] <= maximo) & (df['C_VELOCITAT_VIA'] >= minimo)]
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.histplot(new_df['C_VELOCITAT_VIA'])
plt.subplot(1,2,2)
sns.boxplot(new_df['C_VELOCITAT_VIA'])
plt.show()
```



Distribución de los datos

Aquí podemos ver la distribución de los datos de las columnas que voy a utilizar para el estudio de mi dataset para los objetivos que me he marcado.



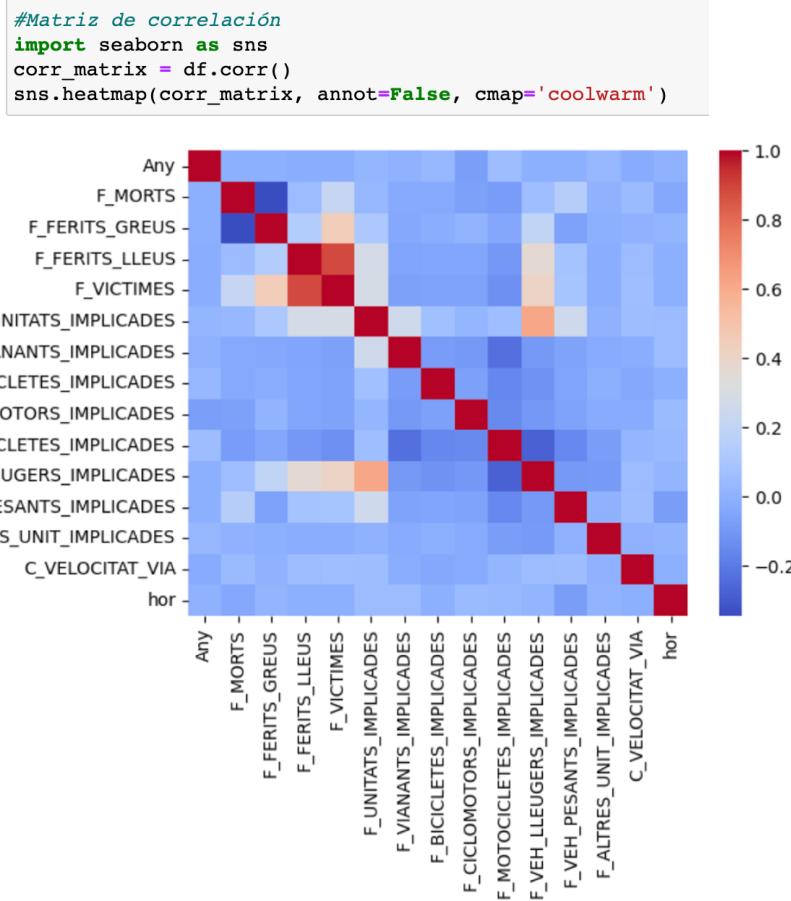


Como podemos observar, la mayor parte de las variables tienen una tendencia de ajustarse a la izquierda, es decir, ya que la mayoría de las variables son las cantidades de heridos, fallecidos, etc, lo normal es que su distribución tenga una mayor cantidad de pocas víctimas y pocas cantidades de valores elevados.

En las demás variables tenemos una distribución más normal a excepción de algunos casos.

Matriz de Correlación

Aquí podemos ver la matriz de correlación la cual nos indica como de relacionadas están las variables entre ellas.



Se puede apreciar cómo hay varias variables que se relacionan fuertemente, como, por ejemplo, la más obvia "Victimes" con "Ferits Lleus". La variable Victimes es la suma de todos los heridos, tanto graves, leves o muertos. A partir de ahora asumiremos que con "todos los accidentes de tráfico" nos referimos a todos los accidentes de tráfico dentro de Cataluña entre los años 2010 y 2021. Con esto podemos ver que la mayoría de las víctimas en un accidente de tráfico acaban siendo heridos leves, seguidos de los graves y en menor cantidad los muertos.

Lo mismo acaba pasando con "Unitats Implicades" y "Veh Lleugers Implicades" ya que las Unidades Implicadas es la suma de todos los tipos de vehículos que aparecen. Significa que la mayoría de los accidentes ocurren con vehículos ligeros. Ya que también no es el tipo de vehículo que tiene más accidentes sino quien tiene también más heridos.

PCA

Primero de todo eliminamos las columnas que son la suma de los valores de otras columnas. Despu s procedemos con el c digo para el PCA.

```
#PCA
newdf = df
newdf = newdf.drop(['F_VICTIMES', 'F_UNITATS_IMPLICADES'], axis=1)

import numpy as np
for column in newdf.columns:
    if newdf[column].dtype == 'object':
        unique_values = newdf[column].nunique()
        if unique_values > 2:
            dummies = pd.get_dummies(newdf[column], prefix=column)
            newdf = pd.concat([newdf, dummies], axis=1)
            newdf.drop(column, axis=1, inplace=True)
        else:
            newdf[column] = pd.Series(np.where(newdf[column].values == 'valor1', 1, 0), newdf.index)

from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

pca_pipe = make_pipeline(StandardScaler(), PCA())
pca_pipe.fit(newdf)

modelo_pca = pca_pipe.named_steps['pca']

pd.DataFrame(
    data = modelo_pca.components_,
    columns = newdf.columns)
```

	Any	zona	F_MORTS	F_FERITS_GREUS	F_FERITS_LLEUS	F_VIANANTS_IMPLICADES	F_BICICLETES_IMPLICADES	F_CICLOMOTORS_IMPLIC.	
0	0.007317	-3.712308e-16	-7.433254e-02	-1.606473e-02	-5.433163e-02	8.528720e-02	-2.230191e-03	1.23373	
1	-0.042547	-5.551115e-17	1.899063e-02	3.533915e-02	4.202611e-02	-2.400228e-02	-3.647655e-02	-5.44029	
2	0.015886	1.387779e-17	2.459979e-02	1.082725e-02	3.468763e-02	-1.307133e-01	1.391033e-02	-3.75716	
3	0.004491	-1.023487e-16	5.275690e-02	-3.828640e-02	-1.066205e-02	7.779270e-02	-1.245372e-02	-9.77651	
4	-0.010970	-5.551115e-17	2.820881e-02	-2.641244e-02	-7.753698e-03	4.333260e-02	1.011153e-03	3.55885	
...	
5163	0.002965	2.513165e-01	1.651240e-16	2.065947e-16	-3.008661e-17	1.765081e-16	-2.842236e-16	3.91397	
5164	-0.000224	-1.365214e-01	1.209563e-16	1.296435e-16	-3.707700e-16	-6.806757e-18	-1.427352e-16	-1.16508	
5165	-0.006784	-3.220299e-02	-6.613633e-17	-3.317659e-17	2.589075e-16	1.590525e-16	-2.843320e-16	-5.03259	
5166	0.004870	7.109923e-01	2.294172e-16	-6.440161e-17	-1.203464e-16	-5.394990e-16	-1.952106e-16	9.20623	
5167	-0.000000	1.429458e-01	1.038666e-16	-2.466729e-17	-7.853181e-17	-8.983970e-17	-3.609123e-17	-9.32663	

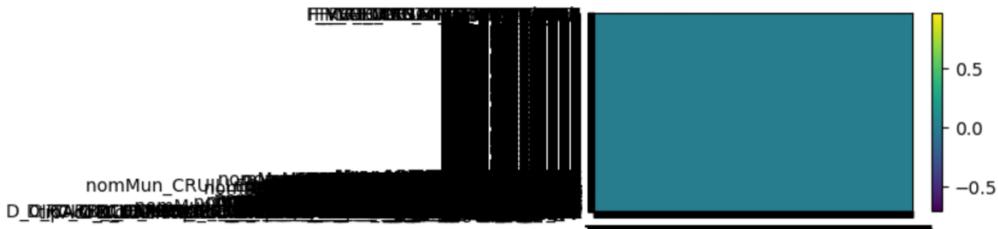
5168 rows x 5168 columns

```

import matplotlib.pyplot as plt

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(4,2))
componentes = modelo_pca.components_
plt.imshow(componentes.T, cmap='viridis', aspect='auto')
plt.yticks(range(len(newdf.columns)), newdf.columns)
plt.xticks(range(len(newdf.columns)), np.arange(modelo_pca.n_components_) +1)
plt.grid(False)
plt.colorbar();

```



Dado que el PCA me ha dado un numero de columnas excesivo, podemos concluir que ninguna columna destaca significativamente respecto a las demás, ya que sino solamente nos habría proporcionado las columnas principales del dataset. Por desgracia, en mi caso, el PCA no ha tenido ninguna utilidad.

Algoritmos de aprendizaje Supervisado

Regresión - Regresión Lineal

Realizaré Regresión Linear para ver la predicción que haría para relacionar el número de víctimas con los vehículos ligeros implicados, es decir, la mayoría de los coches turismo.

```

#Regresión Linear
x = df.iloc[:, [9]].values #F_VICTIMES tiene el índice 9
y = df.iloc[:, 15].values #F_VEH_LLEUGERS_IMPLICADES tiene el índice 15

from sklearn.linear_model import LinearRegression
lin = LinearRegression()

lin.fit(x, y)

print("Intercept:", lin.intercept_)
print("Coeficiente de determinación R^2:", lin.score(x, y))

Intercept: 0.540475313824011
Coeficiente de determinación R^2: 0.16645315197934796

```

Todo y que el coeficiente de determinación es algo bajo, estas dos columnas son de las que más relacionadas están mirando la matriz de correlación. Las otras relaciones que son más fuertes son porque algunas columnas son parte de otras columnas que suman varias.

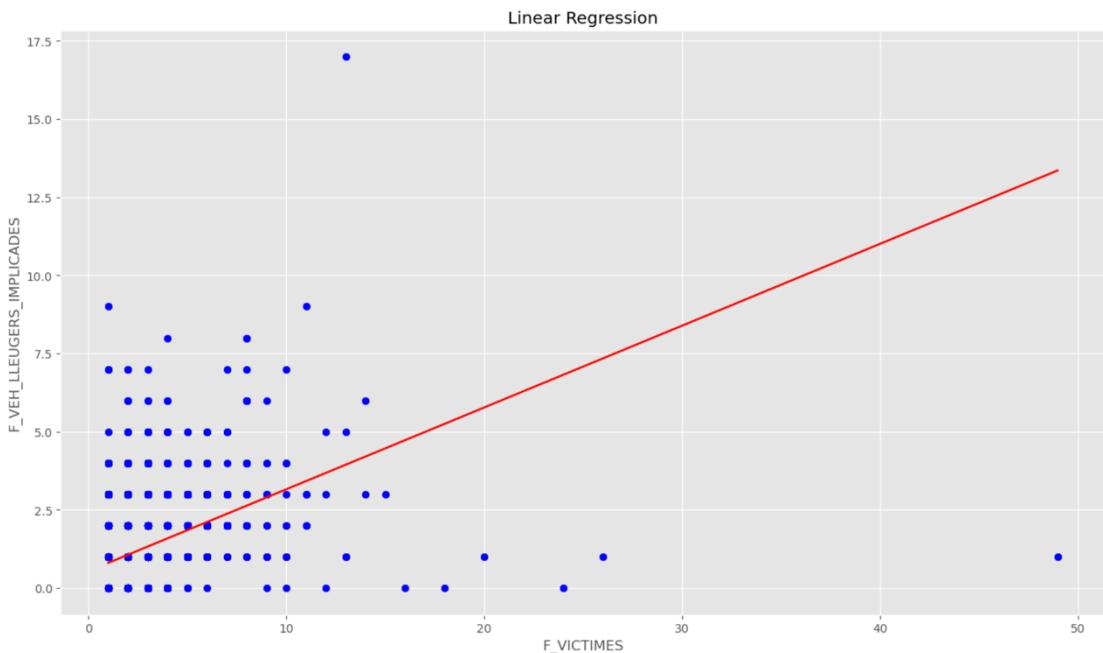
Ahora veremos el grafico del modelo y su predicción.

```

plt.scatter(X, y, color = 'blue')

plt.plot(X, lin.predict(X), color = 'red')
plt.title('Linear Regression')
plt.xlabel('F_VICTIMES')
plt.ylabel('F_VEH_LLEUGERS_IMPLICADES')
plt.show()

```



Podemos ver que la muestra se concentra en la parte inferior izquierda cerca del origen y que hay un par de puntos extremos que se alejan de la mayoría. La predicción podemos ver que es una línea recta (por eso regresión linear) que traza una diagonal de abajo izquierda u origen, hacia arriba a la derecha. Se ve como a más vehículos implicados, habrá más víctimas.

Regresión - Regresión Polinomial

Para ajustar todavía más la predicción a la muestra que tengo, utilizaré la Regresión Polinomial.

```

#Regresión Polinomial
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)

poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)

print("Intercept:", lin2.intercept_)
print("Coeficiente de determinación R^2:", lin2.score(X_poly, y))

```

```

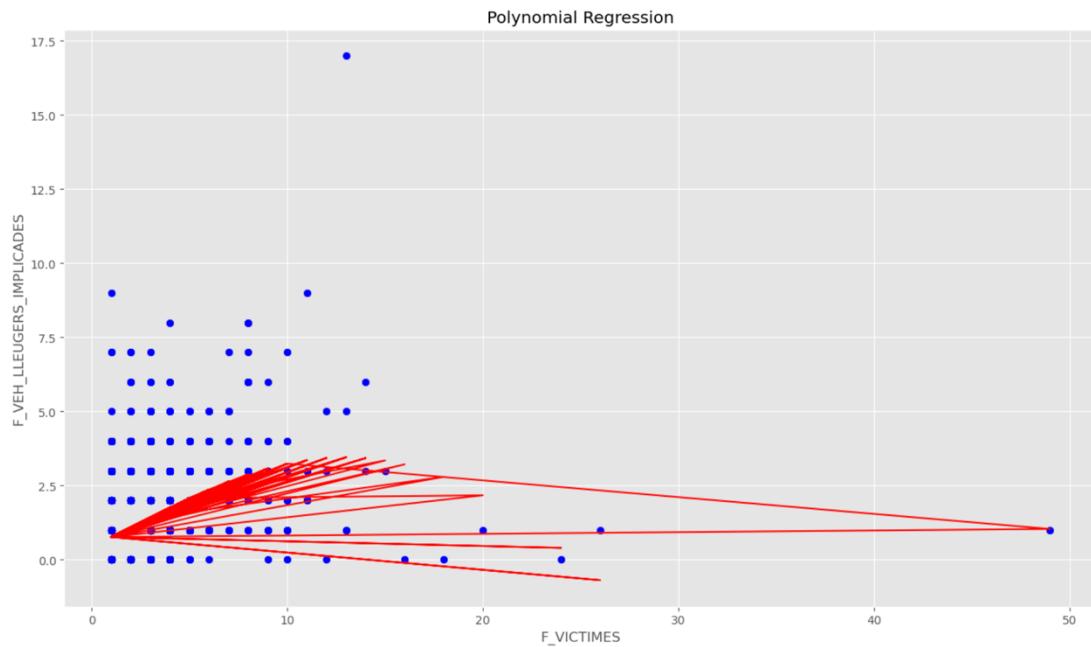
Intercept: 0.44355959878688034
Coeficiente de determinación R^2: 0.20123929255944217

```

Se aprecia que el coeficiente de determinación ha aumentado respecto al de la regresión linear y también ha disminuido el "intercept". Ahora mostraré la predicción con el gráfico de la muestra.

```
: plt.scatter(X, y, color = 'blue')

plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title("Polynomial Regression")
plt.xlabel('F_VICTIMES')
plt.ylabel('F_VEH_LLEUGERS_IMPLICADES')
plt.show()
```



Vemos que ahora no es una sola línea que se dirigía desde el origen hacia arriba a la derecha, sino que se divide en varias líneas rectas para ocupar el máximo espacio posible de la muestra. Vemos como un accidente que tuvo muchas víctimas (casi 50 víctimas) arrastra mucho esta predicción hacia abajo a la derecha.

Clasificación – K-Nearest Neighbours (KNN)

Lo utilizaré para saber e intentar predecir en qué provincia de Cataluña ocurren los accidentes dependiendo del número de vehículos involucrados y el número de víctimas.

```
#KNN
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
import seaborn as sb

%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

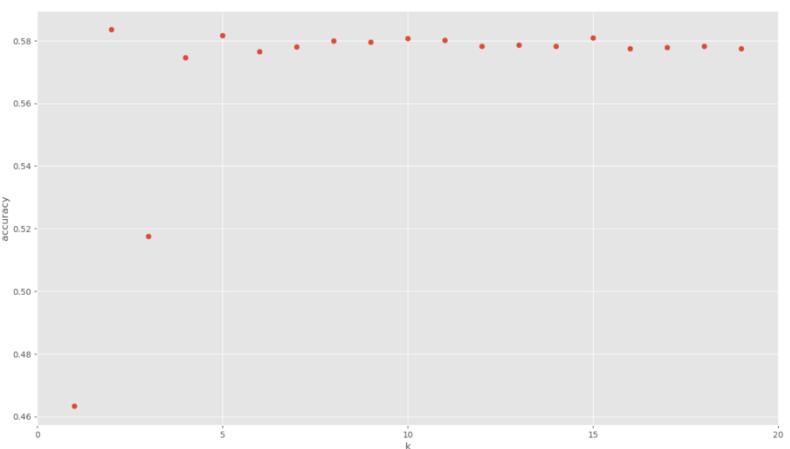
#Pasamos de String a float
nom_dict = {nom: i for i, nom in enumerate(valores_unicos)}
newdf['nomDem'] = df['nomDem'].replace(nom_dict)

X = newdf[['F_VICTIMES', 'F_UNITATS_IMPLICADES']].values
y = newdf['nomDem'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Mediante la siguiente función encontramos la K idónea para nuestro modelo, podemos escogerla nosotros a través del siguiente gráfico en el cual pone para cada K que “accuracy” tendremos. Hay que tener en cuenta que cuanto mayor sea el valor de K, mayor impacto tendrá el algoritmo en los recursos de nuestro equipo a la hora de ejecutarlo.

```
k_range = range(1, 20)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```



Habiendo escogido la K óptima podemos proseguir. Por desgracia, se aprecia que no contamos con muy buen “accuracy” a la hora de predecir nuevos

valores. Todo y que siguen siendo los más altos para las combinaciones de filas de mi dataset.

```
n_neighbors = 2

knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.59
Accuracy of K-NN classifier on test set: 0.58
```

```
: pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.59	0.99	0.74	2234
1	0.14	0.00	0.00	398
2	0.22	0.01	0.02	634
3	0.25	0.00	0.00	552
accuracy			0.58	3818
macro avg	0.30	0.25	0.19	3818
weighted avg	0.43	0.58	0.44	3818

```
: h = .02

cmap_light = ListedColormap(['#FFAAAA', '#ffcc99', '#ffffb3', '#b3ffff'])
cmap_bold = ListedColormap(['#FF0000', '#ff9933', '#FFFF00', '#00ffff'])

clf = KNeighborsClassifier(n_neighbors, weights='distance')
clf.fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

z = np.array(z)
z = z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, z, cmap=cmap_light)

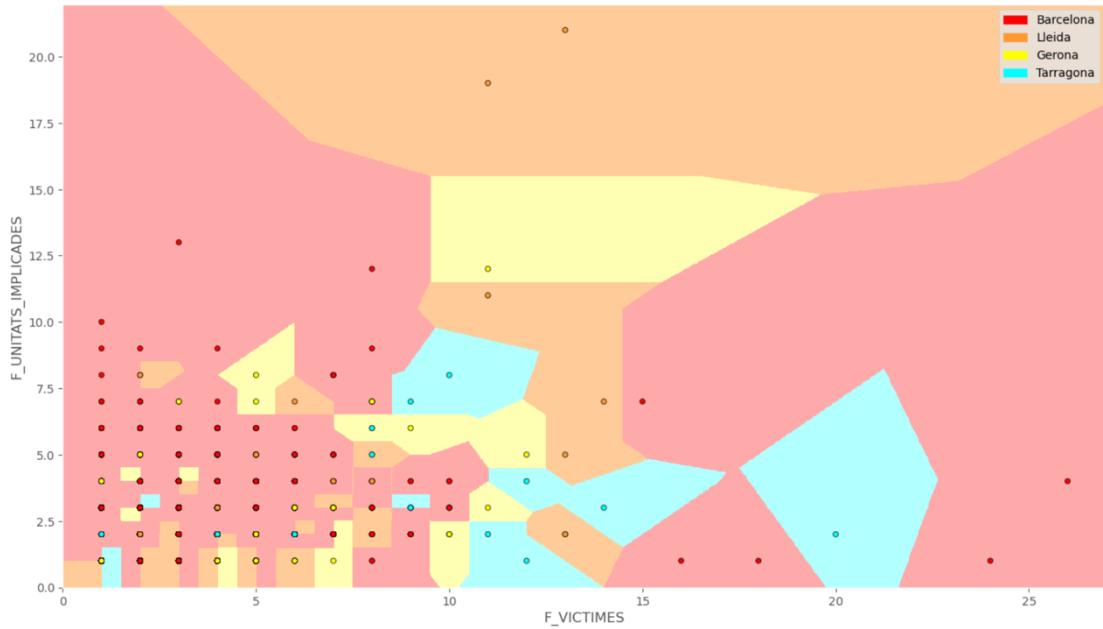
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y_encoded = le.fit_transform(y)

plt.scatter(X[:, 0], X[:, 1], c=y_encoded, cmap=cmap_bold,
            edgecolor='k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())

patch0 = mpatches.Patch(color='#FF0000', label='Barcelona')
patch1 = mpatches.Patch(color='#ff9933', label='Lleida')
patch2 = mpatches.Patch(color='#FFFF00', label='Gerona')
patch3 = mpatches.Patch(color='#00ffff', label='Tarragona')
plt.legend(handles=[patch0, patch1, patch2, patch3])

plt.xlabel('F_VICTIMES')
plt.ylabel('F_UNITATS_IMPLICADES')
plt.show()
```



Con este gráfico podemos predecir si hubiera un nuevo valor dependiendo del número de víctimas y el número de vehículos implicados, en que provincia de Cataluña habrá ocurrido. Por ejemplo, si se produjera un accidente con 13 víctimas involucradas y 14 vehículos, podríamos predecir que habría ocurrido en Gerona.

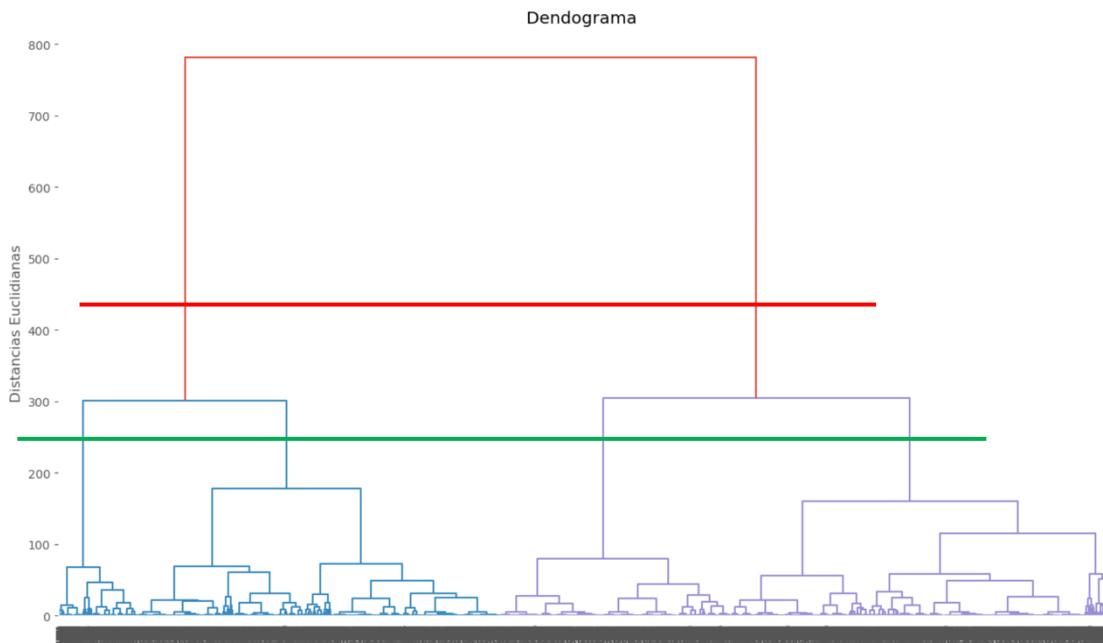
Algoritmos de aprendizaje No Supervisado

Clustering – Clustering Jerárquico

Para el algoritmo no supervisado de clustering utilizaré el clustering jerárquico para ver cómo se distribuyen los accidentes dependiendo de la hora en la que ocurren y el número de víctimas que hay. Seguidamente se muestra el dendrograma con el árbol jerárquico que ha generado el algoritmo.

```
#Clustering Jerárquico
X = newdf.iloc[:, [29, 33]].values
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

import matplotlib.pyplot as plt
plt.title('Dendrograma')
plt.ylabel('Distancias Euclidianas')
plt.show()
```



Aquí se puede apreciar claramente como la distancia máxima la marca la línea roja, ya que es la altura máxima de los enlaces teniendo en cuenta la distancia euclidiana. Si lo separamos con la distancia máxima nos quedarían dos grupos, pero también se podría utilizar la segunda distancia máxima para así obtener algunos grupos más y poder dividir la muestra más determinadamente. Si tenemos en cuenta la línea verde podríamos agrupar la muestra en cuatro grupos suficientemente separados.

Clustering – K-Means

Ahora veremos como clasifica en diferentes clústeres el algoritmo K-Means la relación de columnas de fallecidos con la velocidad de la vía.

```
#K-Means1
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min

%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

X = np.array(dataframe[["op", "ex", "ag"]])
y = np.array(dataframe['categoria'])
X.shape
```

```

fig = plt.figure()
ax = Axes3D(fig)
colores=['blue','red','green','blue','cyan','yellow','orange','black','pink','brown','purple']
asignar=[]
for row in y:
    asignar.append(colores[row])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=asignar,s=60)

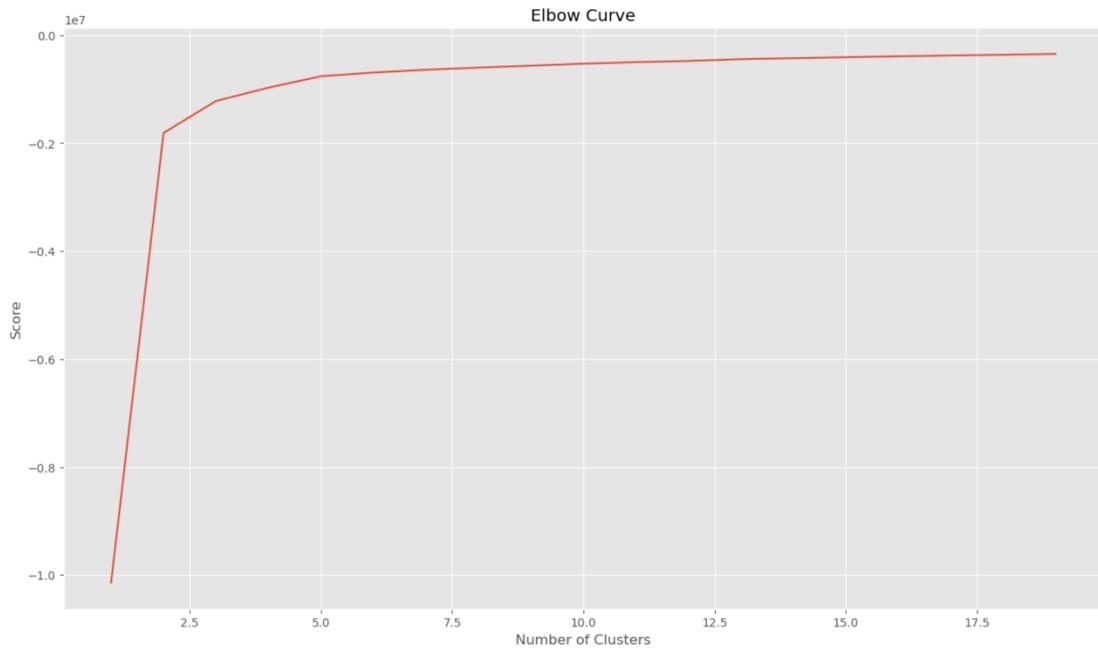
```

Antes que nada, deberíamos realizar la función “del codo” que se muestra a continuación. Sirve para saber que K es la mas óptima para realizar el algoritmo K-Means.

```

Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans
score = [kmeans[i].fit(newdf).score(newdf) for i in range(len(kmeans))]
score
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()

```



En nuestro caso escogeré una K igual a tres, ya que es cuando, una vez estando casi arriba, la curva se aplana.

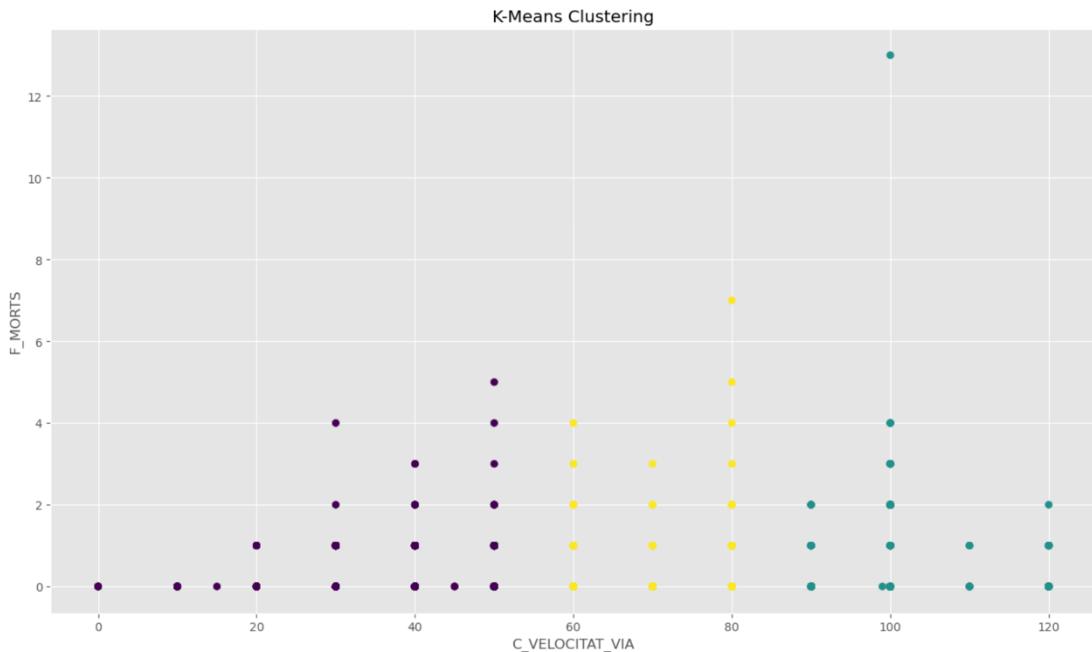
A continuación, veremos el gráfico con los clústeres pintados con diferentes colores. Como hemos dicho anteriormente, la K era igual a tres así que tendremos tres clústeres, tres agrupaciones de valores.

```
#K-Means2
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

x_column = 'C_VELOCITAT_VIA'
y_column = 'F_MORTS'

data = newdf[[x_column, y_column]].to_numpy()
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
labels = kmeans.labels_

plt.scatter(data[:,0], data[:,1], c=labels)
plt.title('K-Means Clustering')
plt.xlabel(x_column)
plt.ylabel(y_column)
plt.show()
```



Se puede apreciar cómo se dividen en tres grupos relativamente homogéneos. El primer clúster con velocidades bajas hasta los 50 km/h. El segundo abarca entre los 60 – 80 km/h. Y el último entre los 90 – 120 km/h.

Conclusiones

El objetivo de este proyecto era obtener más conocimiento sobre los accidentes de tráfico. Ya que, como la mayoría, conduzco casi diariamente, considero que se debe de ser consciente de las posibles consecuencias que tiene hacer una de las actividades más rutinarias.

Pienso que el objetivo se ha cumplido al obtener más información sobre los accidentes de tráfico de Cataluña viéndolos desde nuevos puntos de vista. Se ha visto que la mayoría de los accidentes ocurren, y con diferencia, son en las vías con velocidad de 100 km/h. También se conoce que la mayor concentración de accidentes ocurre sobre las 12:30h del mediodía y que se concentran más de día que durante la noche, ya que también hay menos gente conduciendo. Que la mayoría de los accidentes ocurren en la provincia de Barcelona siendo la mitad del total de los accidentes que ocurren en Cataluña. El accidente de tráfico más común es el de colisión de vehículos en marcha seguido con la mitad de frecuencia, el atropellamiento. También ahora se sabe que el 40% de los accidentes, se implica una motocicleta. Que en el total de accidentes que ocurren, un 15% de estos, hay algún fallecido.

Para este modelo no ha servido de ninguna utilidad el PCA, ya que no hay ninguna columna que destaque por encima de las demás para poder llegar a considerarla componente principal. Las conclusiones, tras haber aplicado los algoritmos, no han llegado a ser todo lo claras y precisas que se hubiera deseado.