ANÁLISIS Y PREDICCIONES DE LA SEGURIDAD VIAL EN UK

Cristian Escudero

Índice

Análisis y predicciones de la seguridad vial en Reino Unido

INTRODUCCIÓN

El objetivo del proyecto es analizar el comportamiento y características más notables de los accidentes y predecir la fatalidad de un accidente apoyándonos en un dataset de accidentes de tráfico entre el 1976 y 2004 ocurrido en Reino Unido.

A partir de la información predicha, se ubicarán sobre un mapa para saber qué puntos son los más críticos y ver si nuestro modelo llega a predecir bien este tipo de accidentes.

Una de las razones por las que he decantado por el proyecto es por cantidad de datos (6millones filas) aprox., por fichero y también por el reto de enfrentarme a un dataset muy desbalanceado.

El proyecto es relevante para mí ya que ha sido una forma de probarme a mí mismo y ver si era capaz de enfrentarme a un dataset tan grande, tan sesgado y lidiar con problemas que pueden surgir en el día a día de un Data Scientist.

El trabajo previo que he podido ver sobre este tema han sido proyectos en Kaggle, pero con datos actuales y periodos más reducidos, como por ejemplo 2015-2017, pero no he visto ningún análisis con un dataset con un periodo de 28 años.

ADQUISICION DE DATOS

Los datos han sido obtenidos del siguiente enlace: Academictorrents. Aunque también puedes obtenerse a partir de la web del gobierno de Reino Unido (datos).

DESCRIPCIÓN DE LOS DATOS

Los datos se componen de 3 archivos en formato csv:

- Casualty7904 → Víctimas
- Vehicles7904 → Vehículos
- Accidents7904 → Accidentes

Como cada archivo tiene bastantes campos, se facilita un diccionario de los datos en formato Excel, donde se puede consultar cada campo y las categorías de cada uno en el repositorio, con el nombre de Data-Guide.xls.

Campos usados

- Accidents
    - Accident_Index object
    - Accident_Severity int64
    - Number_of_Vehicles int64
    - Number_of_Casualties int64
    - Date object
    - Day_of_Week int64
    - 1st_Road_Class int64
    - Road_Type int64
    - Speed_limit int64
    - Month (Generada) object
    - Year(Generada) int64
    - Hour (Generada) datetime
    - Daytime (Generada) object
    - Road_Surface_Conditions int64
    - Weather_Conditions int64
    - Light_Conditions int64

- Casualty
    - Age_Band_of_Casualty int64
    - Sex_of_Casualty int64

- Vehicles
    - Vehicle_Type int64
    - Sex_of_Driver int64
    - Age_Band_of_Driver int64
    - Engine_Capacity_(CC) int64
    - Age_of_Vehicle int64

Semánticas de cada campo

Accident_Index: Índice de accidente.

Accident_Severity: Gravedad de accidente.

Accident Severity	
code	label
1	Fatal
2	Serious
3	Slight

Number_of_Vehicles: Vehículos implicados

Number_of_Casualties: Víctimas implicadas

Date: Fecha del suceso

Day_of_Week: Día de la semana

Day_of_Week	
code	label
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

1st_Road_Class: Tipo de vía

1st road class	
code	label
1	Motorway
2	A(M)
3	A
4	B
5	C
6	Unclassified

Road_Type: Tipo de carretera.

Road Type	
code	label
1	Roundabout
2	One way street
3	Dual carriageway
6	Single carriageway
7	Slip road
9	Unknown
12	One way street/Slip road
-1	Data missing or out of range

Speed_limit: Límite de velocidad

Month: Mes

Year -> Año

Hour -> Hora

Daytime -> Período del día

Daytime	
Commuting to home	hour >=5 and hour <10
Office hours	hour >= 10 and hour < 15
Commuting to work	hour >= 15 and hour < 19
Evening	hour >= 19 and hour < 23
Night	rest of hours : night

Road_Surface_Conditions -> Condiciones de la carretera

Road Surface Conditions	
code	label
1	Dry
2	Wet or damp
3	Snow
4	Frost or ice
5	Flood over 3cm. deep
6	Oil or diesel
7	Mud
-1	Data missing or out of range

Weather_Conditions -> Condiciones Atmosféricas

Weather conditions	
code	label
1	Fine no high winds
2	Raining no high winds
3	Snowing no high winds
4	Fine + high winds
5	Raining + high winds
6	Snowing + high winds
7	Fog or mist
8	Other
9	Unknown
-1	Data missing or out of range

Light_Conditions -> Condiciones de iluminación

Light Conditions	
code	label
1	Daylight
4	Darkness - lights lit
5	Darkness - lights unlit
6	Darkness - no lighting
7	Darkness - lighting unknown
-1	Data missing or out of range

Age_Band_of_Casualty -> Rango edades de las Víctimas

Age_Band_of_Casualty	
code	label
1	0 - 5
2	6 - 10
3	11 - 15
4	16 - 20
5	21 - 25
6	26 - 35
7	36 - 45
8	46 - 55
9	56 - 65
10	66 - 75
11	Over 75

Sex_of_Casualty -> Sexo de las víctimas

Sex_of_Casualty	
code	
1	Male
2	Female

Vehicle_Type -> Tipo de vehículo

Vehicle_Type	
code	label
1	Pedal cycle
2	Motorcycle 50cc and under
3	Motorcycle 125cc and under
4	Motorcycle over 125cc and up to 500cc
5	Motorcycle over 500cc
8	Taxi/Private hire car

9	Car
10	Minibus (8 - 16 passenger seats)
11	Bus or coach (17 or more pass seats)
16	Ridden horse
17	Agricultural vehicle
18	Tram
19	Van / Goods 3.5 tonnes mgw or under
20	Goods over 3.5t. and under 7.5t
21	Goods 7.5 tonnes mgw and over
22	Mobility scooter
23	Electric motorcycle
90	Other vehicle
97	Motorcycle - unknown cc
98	Goods vehicle - unknown weight
-1	Data missing or out of range
103	Motorcycle - Scooter
104	Motorcycle
105	Motorcycle - Combination
106	Motorcycle over 125cc
108	Taxi (excluding private hire cars)
109	Car (including private hire cars)
110	Minibus/Motor caravan
113	Goods vehicle over 3.5 tonnes

Sex_of_Driver -> Sexo de los conductores

Sex_of_Driver	
code	label
1	Male
2	Female

Age_Band_of_Driver -> Rango de edades de los conductores

Age_Band_of_Driver	
code	label
1	0 - 5
2	6 - 10
3	11 - 15
4	16 - 20
5	21 - 25
6	26 - 35
7	36 - 45
8	46 - 55

9	56 - 65
10	66 - 75
11	Over 75

Engine_Capacity_(CC) -> Cilindrada

Age_of_Vehicle -> Edad del vehículo

La selección de estos campos se ha hecho de manera que sean las variables que a mi juicio pueden tener más peso y pueden aportar más al modelo a la hora de realizar una predicción.

Aunque esto no es determinante ya que es una primera reflexión, luego podemos ir descartando o añadiendo datos en función de su comportamiento, distribución, colinealidad etc.

METODOLOGÍA

El proyecto se divide en varios notebooks:

- Limpieza y transformación de los datos
- Visualización
- Feature Engineering
- Machine learning
- Front End
- Test_data

Limpieza y transformación de los datos

En esta fase lo que tratamos de hacer es explorar por cada uno de los datasets qué columnas tenemos, con cuál nos vamos a quedar y hacer las trasformaciones pertinentes.

Se realizan los siguientes procedimientos:

- Importamos librerías.
- Comprobamos la codificación del archivo.

- Comprobamos porcentaje de NaN o elementos nulos.
- Decartamos las columnas que no nos interesan, con alto porcentaje de NaN o valores -1 (missing values)
- Descartamos filas con NaN
- Eliminamos filas con valores -1
- Procesamiento de los datos
    - Se crean nuevas columnas:
        - Month
        - Year
        - Hour
        - Daytime
        - Road_Surface_Conditions_2 → Good/Bad
        - Weather_Conditions_2 → Good/Bad
        - Light_Conditions_2 → Good/Bad
        - Speed_limit_2 → Categorizamos en grupos de velocidades

    - Reemplazo de valores por etiquetas para las siguientes columnas:

        - Accidents dataset
            - Month
            - Daytime
            - Light_Conditions
            - Road_Surface_Conditions
            - Weather_Conditions
            - Speed
            - Accident_Severity
            - Day_of_Week
            - 1st_road_class
            - Road_Type

        - Dfmerged (casualty + vehicles) dataset
            - Sex_of_Casualty
            - Age_Band_of_Casualty
            - Casualty_Type
            - Vehicle_Type
            - Sex_of_Driver
            - Age_Band_of_Driver

Las columnas transformadas se crean para cambiar la estrategia y que el valor con más porcentaje sea distribuido. Así no tendremos muchas categorías con porcentajes próximos al 0 y otra que acumule el mayor porcentaje (90% por ejemplo), ya que esto nos va a dificultar más tarde en el análisis y en la creación del modelo de machine learning.

Una vez tenemos los dataframes limpios y etiquetados procedemos a guardarlos.

accidents_labeled.csv / **veh_cas_labeled.csv** → Los utilizaremos en visualización.

alldfjoined.csv → dataset completo listo para realizar feature engineering.

**Los archivos se guardarán en un path que al inicio del notebook EDA se pedirá al usuario.*
Este path se guardará en un archivo llamado path.txt junto con el path del repositorio, donde se utilizará en los posteriores notebooks de jupyter.

Visualización

En esta parte realizaremos un análisis de todas las características importantes de los 3 datasets.

Para llevar a cabo esta sección utilizaremos las librerías de visualización siguientes:

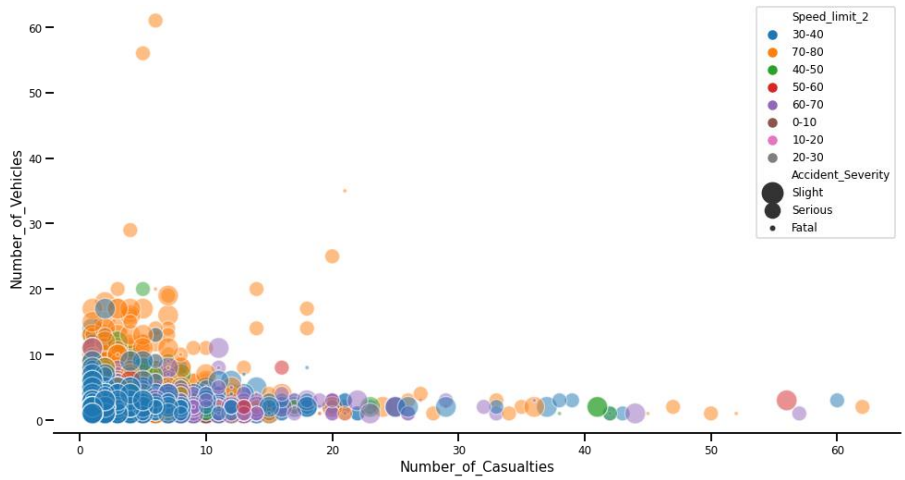
- Seaborn
- Matplotlib
- Plotly – cufflinks

Algo a destacar es la librería de plotly con cufflinks, esta última herramienta nos proporciona una manera rápida de hacer plots directamente con plotly mediante pandas, ahorrándonos código.

Antes de empezar a hacer los gráficos cargaremos los datos que justamente hemos guardo anteriormente.

Gráficas

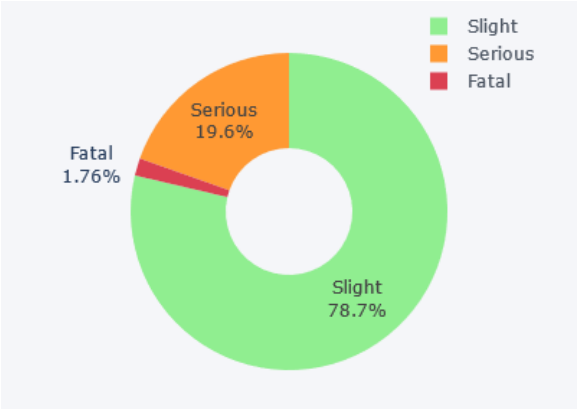
Scatter plot víctimas y vehículos



Podemos ver que lo que más predomina son accidentes leves con velocidades comprendidas entre 70-80 km/h (naranja), 60-70km/h(violeta) y 30-40km/h (azul).

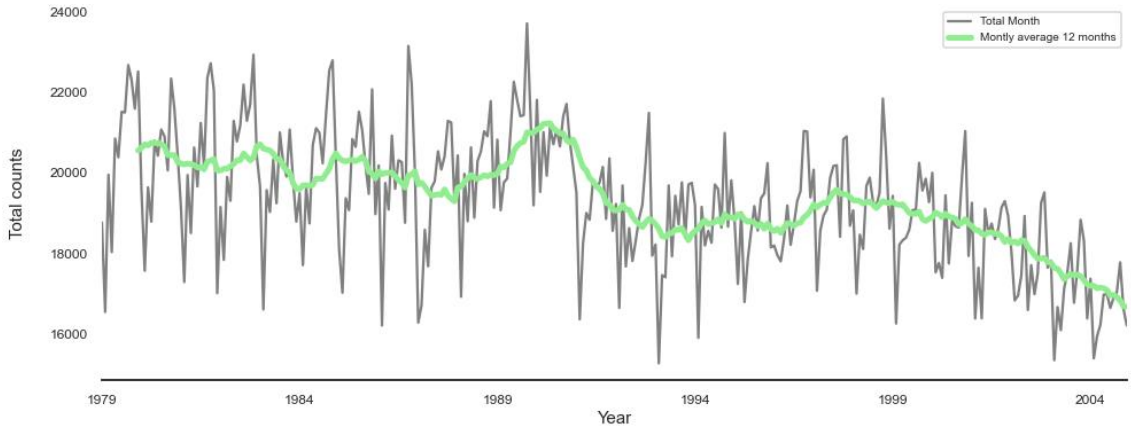
En cuanto a los vehículos implicados se observa que se acumulan más los datos entre 0-20 , no obstante en las víctimas predominan datos acumulados desde 0 hasta 40, siendo más intenso entre 0 y 10.

Pie Chart Accident Severity



En el gráfico vemos como la variable que nos hemos definido como target, está muy desbalanceada, sólo con un aprox. 1.8% de datos fatales, 78.7% en leves y 19.6% en graves.

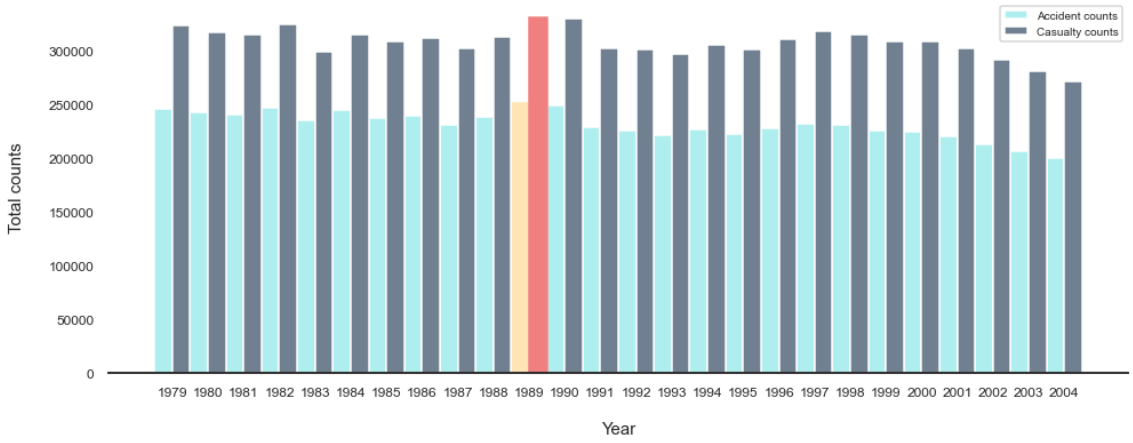
Media de accidentes por mes



Se puede apreciar un pico alrededor de los años 90, donde probablemente se deba a que fue una época donde las compañías de vehículos comercializaron muchos coches accesibles al usuario medio y con una gran cilindrada y unos sistemas de seguridad menos eficientes que hoy en día.

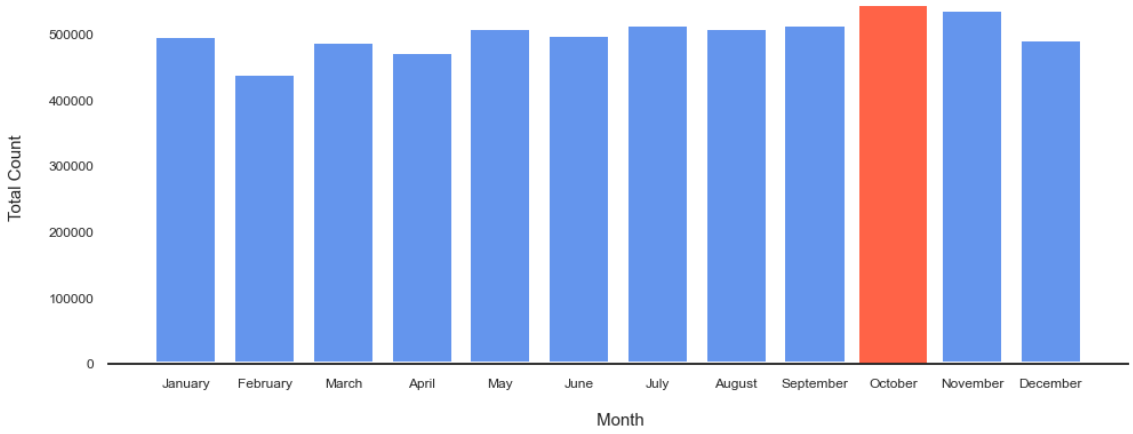
Por alguna razón los accidentes tienden a bajar a media que van pasando los años a partir de los 90, podría indicarnos una clara mejora en los sistemas de seguridad en los automóviles.

Accidentes por año



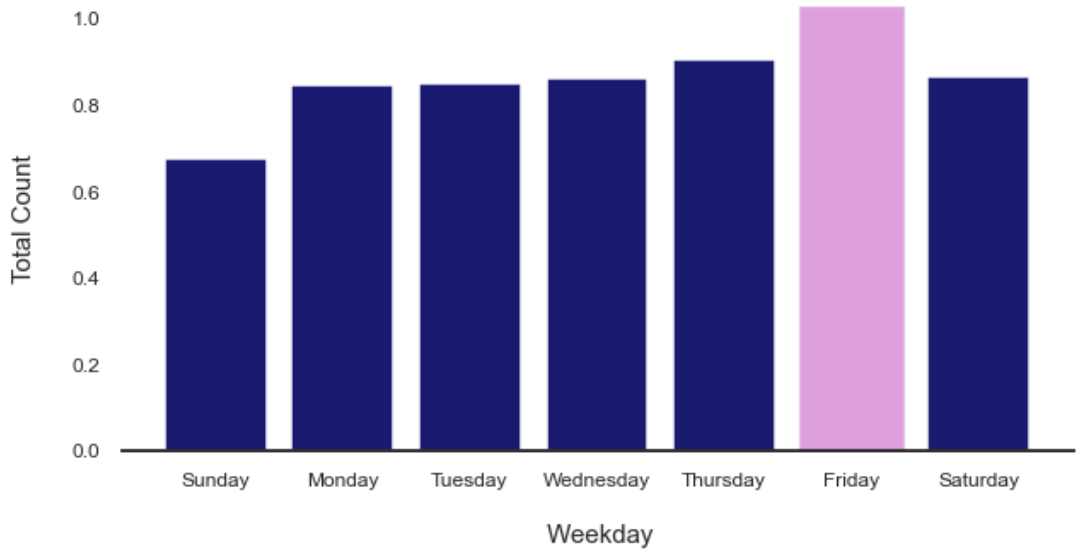
El año con más accidentes y víctimas es 1989.
Tanto accidentes como víctimas presentan una ligera bajada a partir del año 2000.

Accidentes por mes



El mes de octubre es el que más accidentes acumula junto con noviembre. Podríamos pensar que es por las fechas próximas a la navidad ya que es un patrón que se repite todos los años por esta fecha.

Accidentes por día semanal



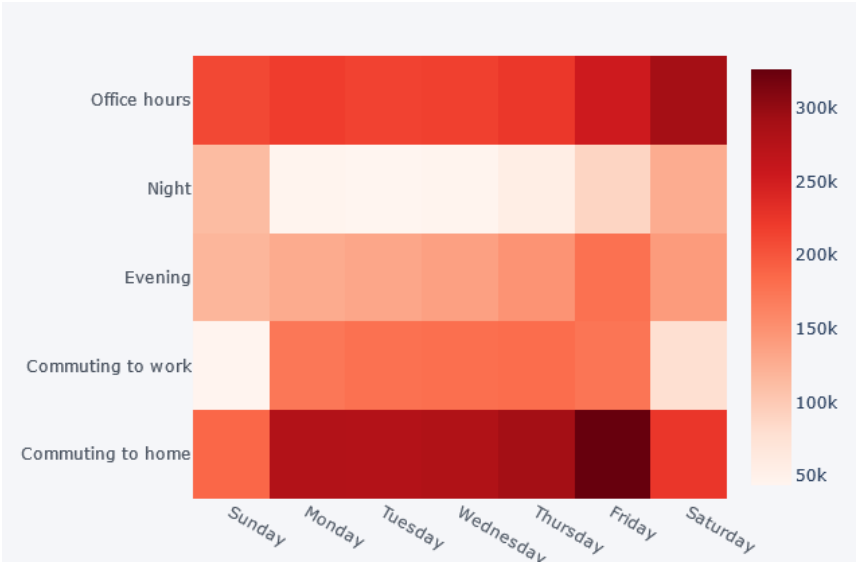
Suceden más accidentes, cuando se acerca el fin de semana, era algo de prever ya que la población el fin de semana tiende a salir más y a hacer planes, con lo que la probabilidad de accidente es mayor.

Mapa de calor Weekday – Daytime

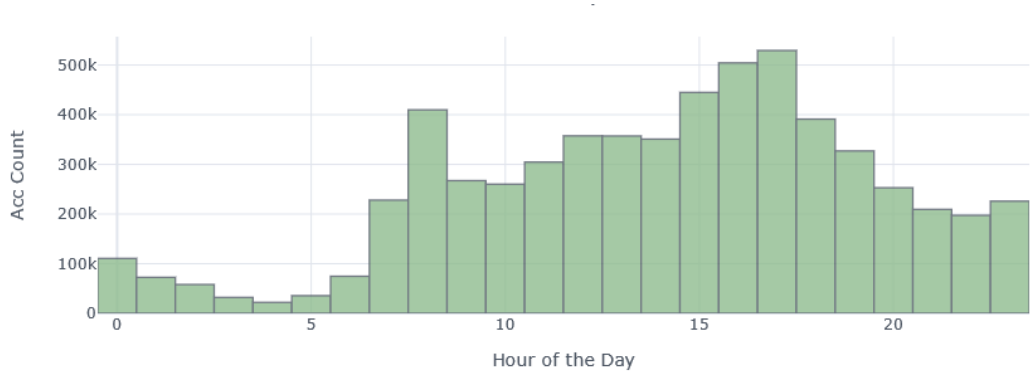
Cuando más accidentes suceden es a la vuelta del trabajo y en horas de oficina el fin de semana.

Es decir que de 15h a 19h es cuando más accidentes hay el fin de semana, y entre semana se acumulan más a la salida del trabajo.

El fin de semana es cuando aumentan los casos de víctimas de accidentes de tráfico y se producen más accidentes severos en horario nocturno.



Histograma Accidentes por cada hora

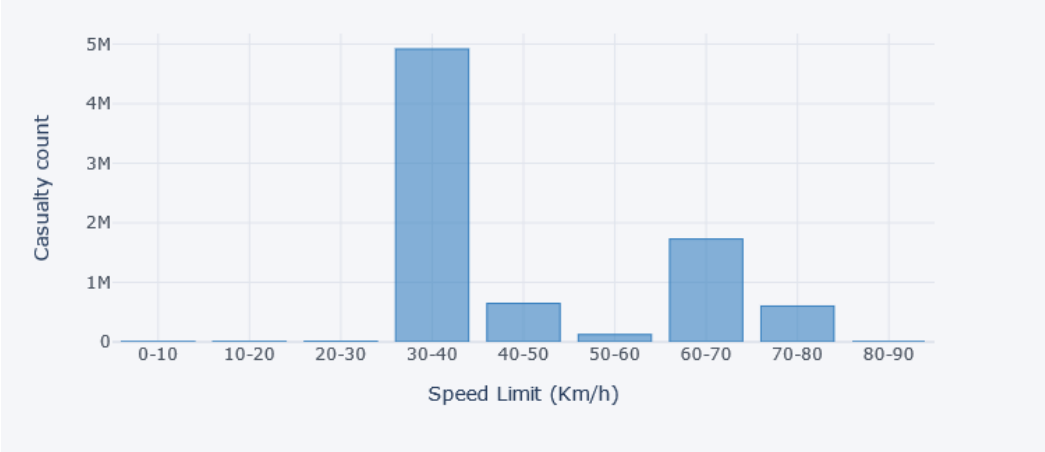


Las horas donde se acumulan más accidentes coincidiendo con nuestro gráfico de antes es de 15h a 19h.

Tenemos un pico máximo a las 17h de más de 500.000 accidentes, y a primera hora de la mañana (8:00h) un pico de 400.000 accidentes, el más alto de toda la mañana, debido al desplazamiento que se realiza al lugar de trabajo.

A partir de las 20h descienden hasta los 200.000 accidentes.

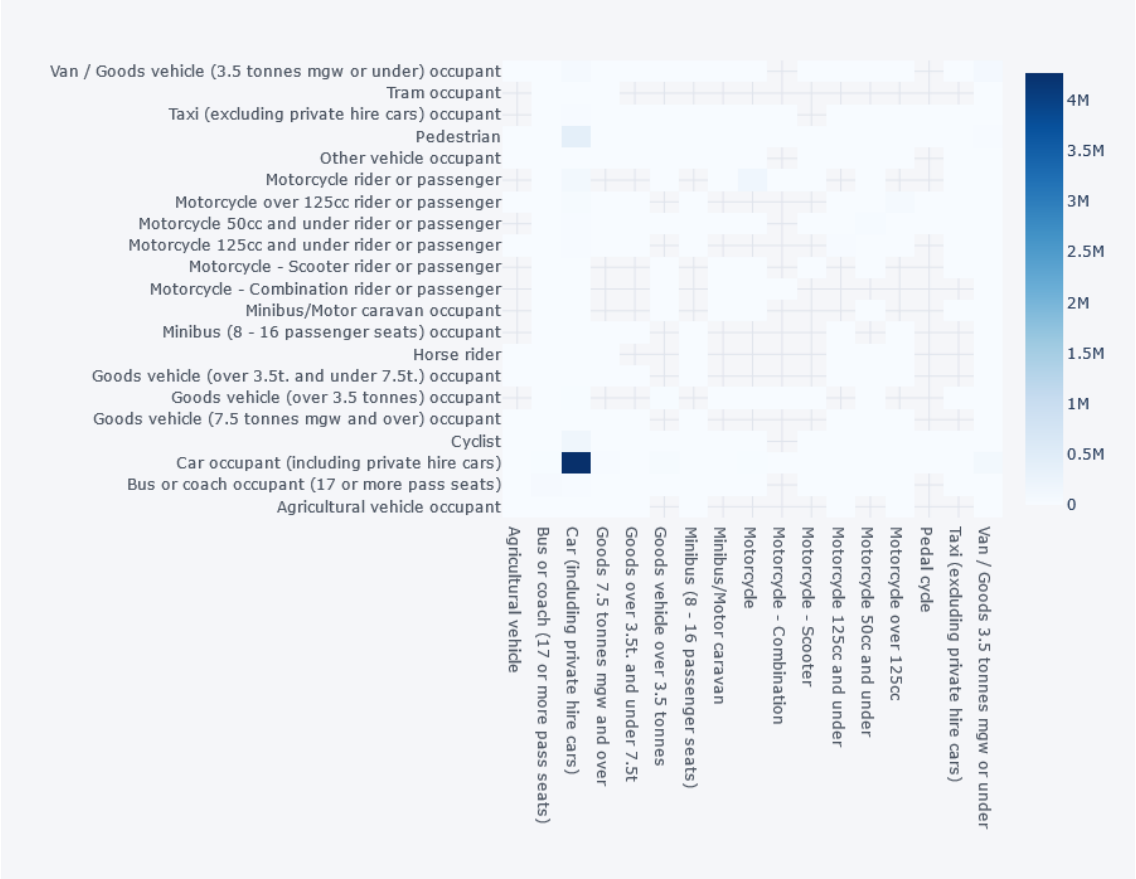
Recuento de víctimas por límite de velocidad



Entre 30-40 km/h es donde más víctimas se reportan, con un 61,3 % de registros, es muy probable que estos accidentes sucedan claramente en la ciudad.

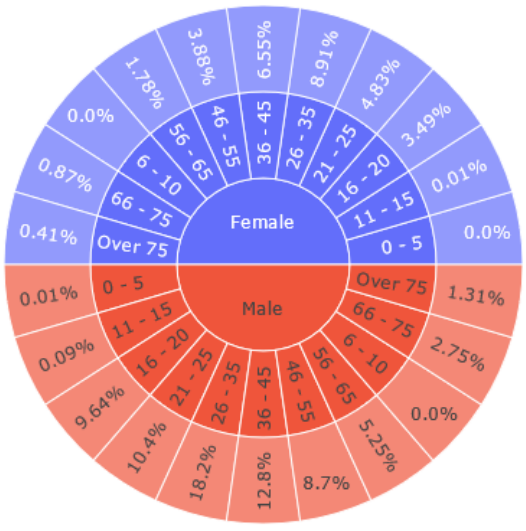
Con un 21.52 % de víctimas le siguen los accidentes ocurridos entre 60-70 km/h.

Relación entre Vehículos y Víctimas (tipos)



Este gráfico muestra la correlación entre los vehículos y las víctimas. Los accidentes mayormente se producen entre coches, y coche y peatón.

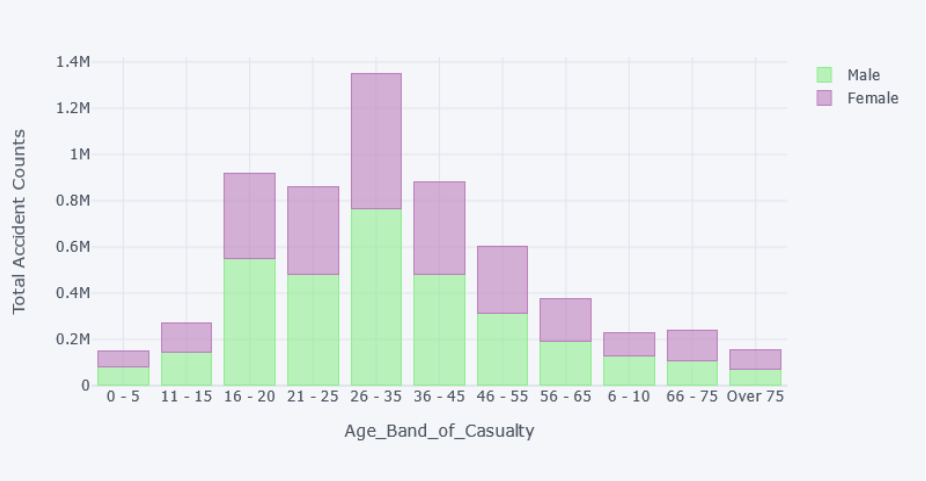
Porcentaje de conductores implicados por sexo



Los conductores que acumulan más accidentes son los hombres con un 69.27 %. En cuanto a las mujeres son el colectivo que menos accidente tiene con el 30 % restante.

Las franjas de edad que más destacan en porcentaje tanto en hombres como en mujeres son entre 26 y 35 años.

Recuento de Accidentes por Sexo



	Age_Band_of_Casualty	Sex_of_Casualty	Acc_Index	%	Sex_of_Casualty	Acc_Index	%
0	0 - 5	Male	81703	2.46	Female	67340	2.47
1	11 - 15	Male	144971	4.37	Female	128245	4.71
2	16 - 20	Male	550043	16.57	Female	368239	13.53
3	21 - 25	Male	483441	14.57	Female	376717	13.85
4	26 - 35	Male	763961	23.02	Female	586090	21.54
5	36 - 45	Male	481622	14.51	Female	402506	14.79
6	46 - 55	Male	311484	9.38	Female	292067	10.73
7	56 - 65	Male	190050	5.73	Female	185947	6.83
8	6 - 10	Male	131061	3.95	Female	99624	3.66
9	66 - 75	Male	110263	3.32	Female	127094	4.67
10	Over 75	Male	70392	2.12	Female	86979	3.20

Los rangos de edad tanto en hombre como en mujer con mayor porcentaje de víctimas son las comprendidas entre 26-35 años.

Pero podemos decir en términos generales que empieza a dispararse a partir de los 16 hasta los 45 años, con un 79 % en los hombres y un 64 %.

En cuanto al porcentaje total de víctimas los hombres sobrepasan en 10 % a las mujeres, con un 45% de mujeres y un 55% de varones.

Cilindradas más populares

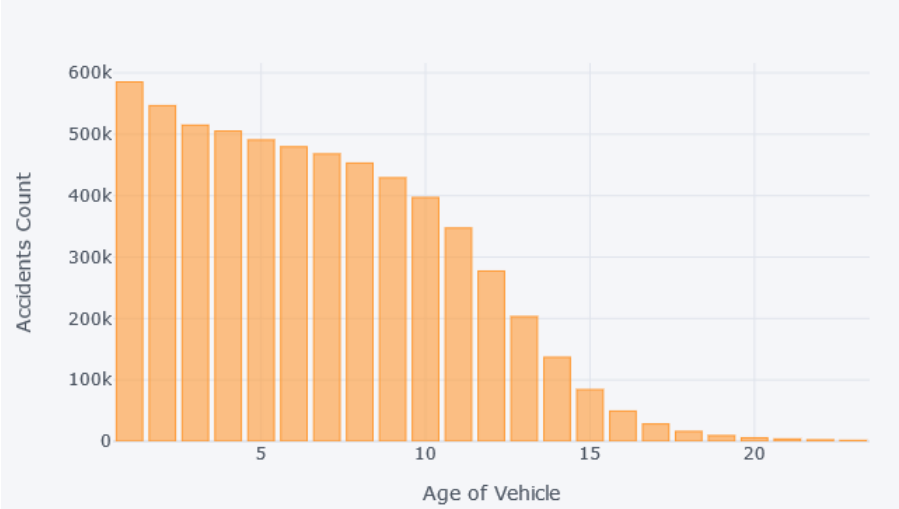
	Engine_Capacity_CC	Count
0	1.60	745749
1	1.30	406231
2	2.00	312915
3	1.12	306948
4	1.39	274888
5	1.99	227946
6	1.80	225098
7	1.59	221414
8	1.00	218941
9	1.27	187584

Las cilindradas más destacadas son, 1.6, 1.3, 2.0, 1.8, 1.0.

Era de esperar ya que son los motores más fiables de esta época e incluso hoy en día siguen siendo los más populares.

Podemos ver que no las cilindradas más altas están reportando un mayor número de accidentes.

Edad de los vehículos

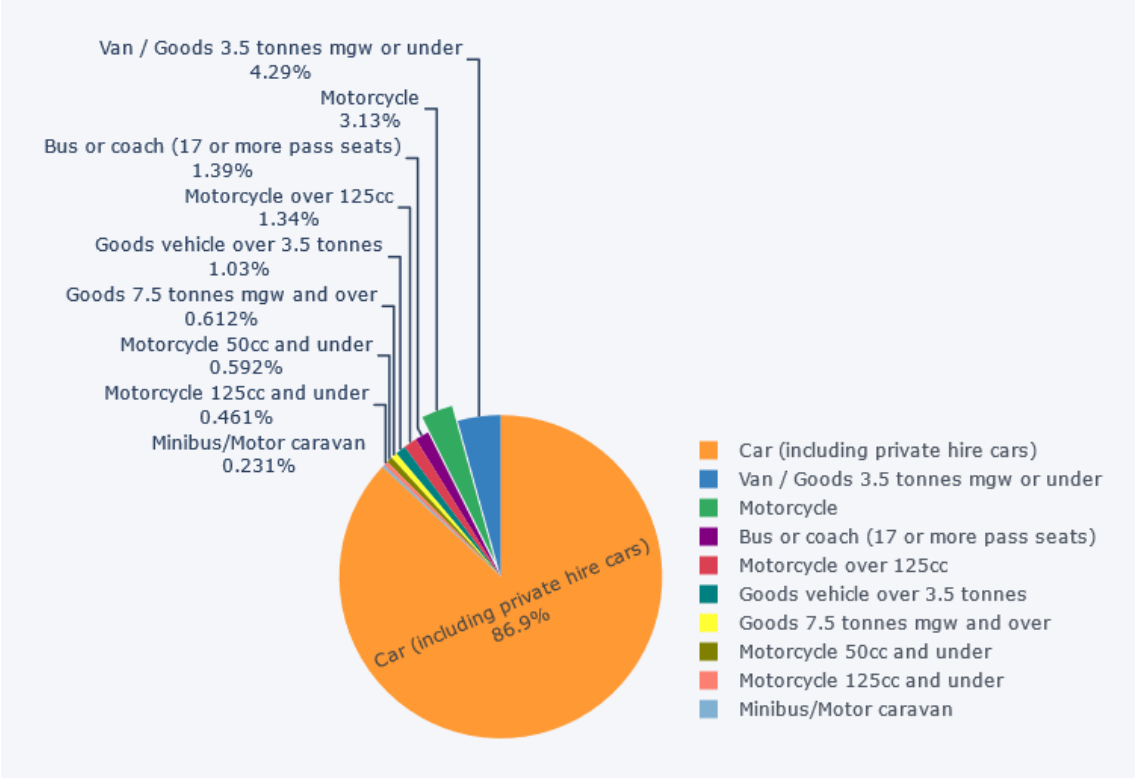



```
0-5 years   :  0.44
5-10 years  :  0.37
10-20 years :  0.19
20-30 years :  0.0
```

Predominan los coches no mayores a 10 años con un porcentaje de 81 % de vehículos, es decir la gran mayoría.

El resto de vehículos son coches entre 10 y 20 años, que representan el 20% restante.

Pie Chart - Accidentes por tipo de vehículo



La mayoría de los accidentes se producen por coches 87%, seguidamente motocicletas con un 5.5 % y (furgonetas/camiones pequeños) con un 4.3 %.

El resto de vehículos aparecen con porcentajes menores al 2%.

Feature Engineering

Es la parte donde se procesan los datos y sufren una segunda transformación para que el modelo predictivo pueda aprender de unos datos limpios y nada sesgados.

Nuestro objetivo aquí es evitar que haya variables muy correlacionadas, aplicar técnicas para mejorar la distribución de las variables numéricas, eliminar outliers y analizar la matriz de correlación.

Variables propuestas para Feature engineering

```
Vehicle_Type                  object
Sex_of_Driver                 object
Age_Band_of_Driver            object
Engine_Capacity_(CC)           int64
Age_of_Vehicle                 int64
Accident_Severity             object
Day_of_Week                   object
Speed_limit                    int64
Daytime                       object
Road_Surface_Conditions_2     object
Weather_Conditions_2          object
Light_Conditions_2            object
```

He seleccionado estas variables ya que tienen una repercusión directa en la variable target "Accident_Severity" y nos hablan de muchas de las condiciones que encontramos en un accidente de tráfico como son las condiciones climáticas, condiciones del asfalto, condiciones de iluminación, factores como cilindrada de los vehículos, la velocidad, día de la semana, años del conductor, tipo de vehículo, etc.

Una vez tenemos seleccionadas las variables con las que vamos a trabajar, comenzamos el procesado de las variables numéricas por un lado y de las variables categóricas por otro.

Variables numéricas:

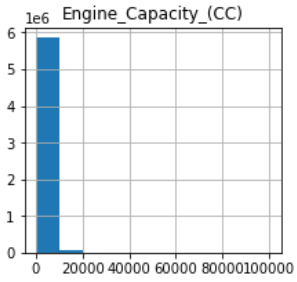
- `Speed_limit` `int64`
- `Engine_Capacity_(CC)` `int64`
- `Age_of_Vehicle` `int64`

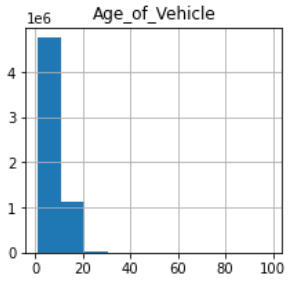
Variables categóricas:

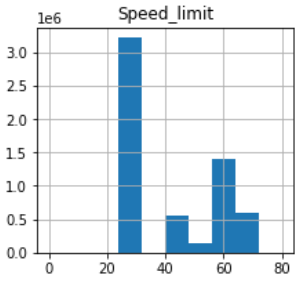
- `Vehicle_Type` `category`
- `Sex_of_Driver` `category`
- `Age_Band_of_Driver` `category`
- `Accident_Severity` `category`
- `Day_of_Week` `category`
- `Daytime` `category`
- `Road_Surface_Conditions_2` `category`
- `Weather_Conditions_2` `category`
- `Light_Conditions_2` `category`

Procesado de variables numéricas

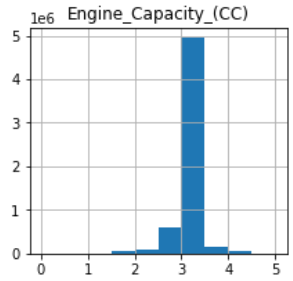
Comprobamos la distribución de cada variable

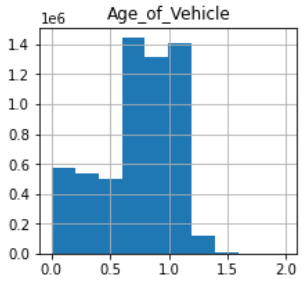


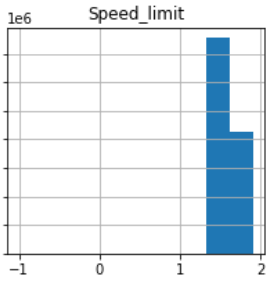




Aplicamos logaritmo a cada variable para obtener una distribución similar a una normal.



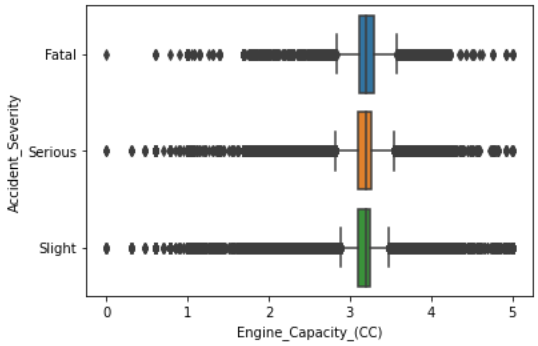


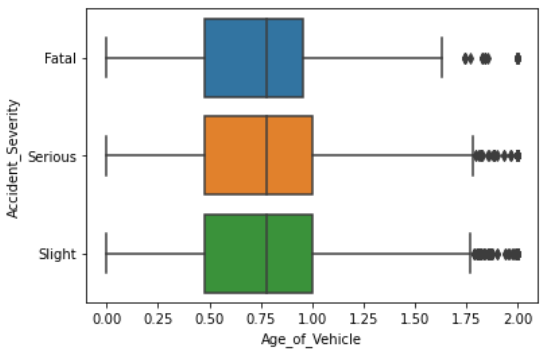


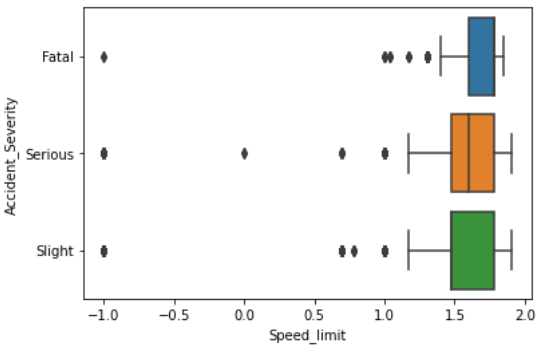
Como podemos ver, hemos mejorado un poco la distribución, aunque no llega a seguir la distribución exacta de una normal, pero nos servirá.

Outliers

Gráficas previas a la eliminación de Outliers.







Engine_Capacity

Es la variable que tiene una repercusión más directa en la variable target ya que el boxplot presenta unos cuantiles muy próximos y la mayoría de datos están por encima del máximo y por debajo del mínimo.

Age_of_Vehicle

Presenta sólo un pequeño porcentaje de outliers por encima del máximo y una mejor distribución de los datos.

Speed_limit

En esta variable se puede observar que el boxplot está desplazado a la derecha ya que ahí encontramos la mayoría de los datos, con ciertas outliers por debajo del mínimo.

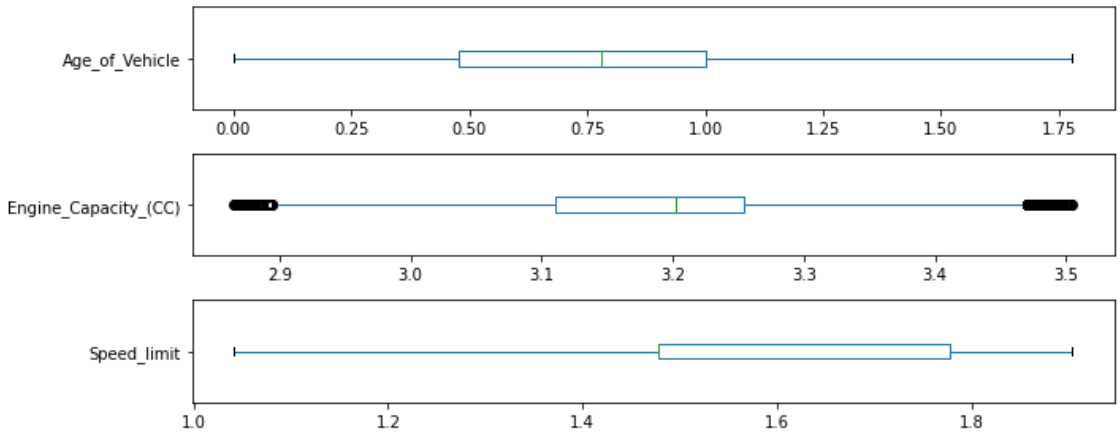
Se procede a eliminar todos los outliers con el método IQR.

Multiplicamos el rango intercuartílico (IQR) por 1,5 (una constante que se utiliza para discernir los valores atípicos).

Agregamos 1.5 x (IQR) al tercer cuartil. Cualquier número mayor que este es un supuesto valor atípico.

Restamos 1.5 x (IQR) del primer cuartil. Cualquier número menor que este es un supuesto valor atípico.

Resultado de aplicar método IQR.



El boxplot de cada variable se muestra mucho más limpio, permitiendo en la medida de lo posible no perder tantos datos.

Si tenemos una variable con outliers y está sesgada esto no beneficiará para nada al modelo predictivo, pero si eliminamos todas las outliers también en cierta manera estamos dejando

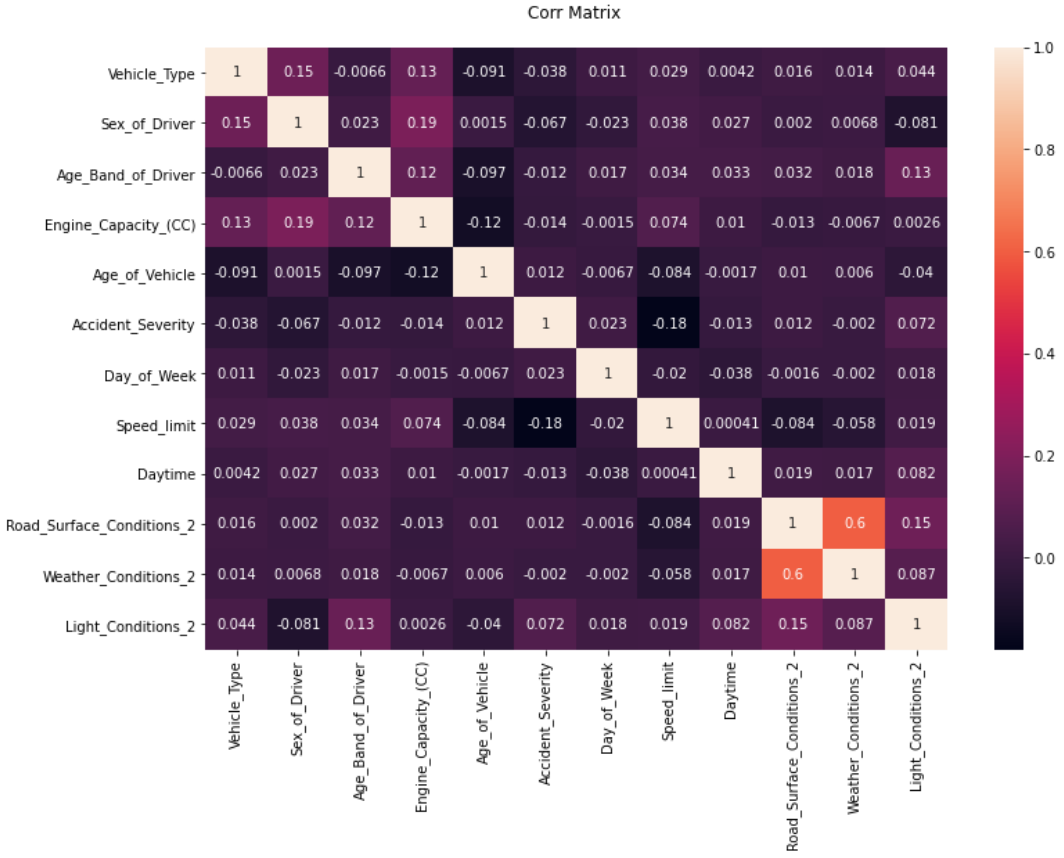
sin esos datos a la variable, con lo cual podría ser que nos genere ruido o incluso no nos aporte nada. La idea es estar siempre en un término medio que nos pueda aportar la máxima información posible sin que esta pueda ser perjudicial.

Procesado de variables categóricas

Para el procesado de las variables categóricas, he optado primero por hacer un Label-encoder tanto a las features como al target.

Una vez hecho esto ya podemos crear la matriz de correlación y visualizar qué comportamiento tenemos.

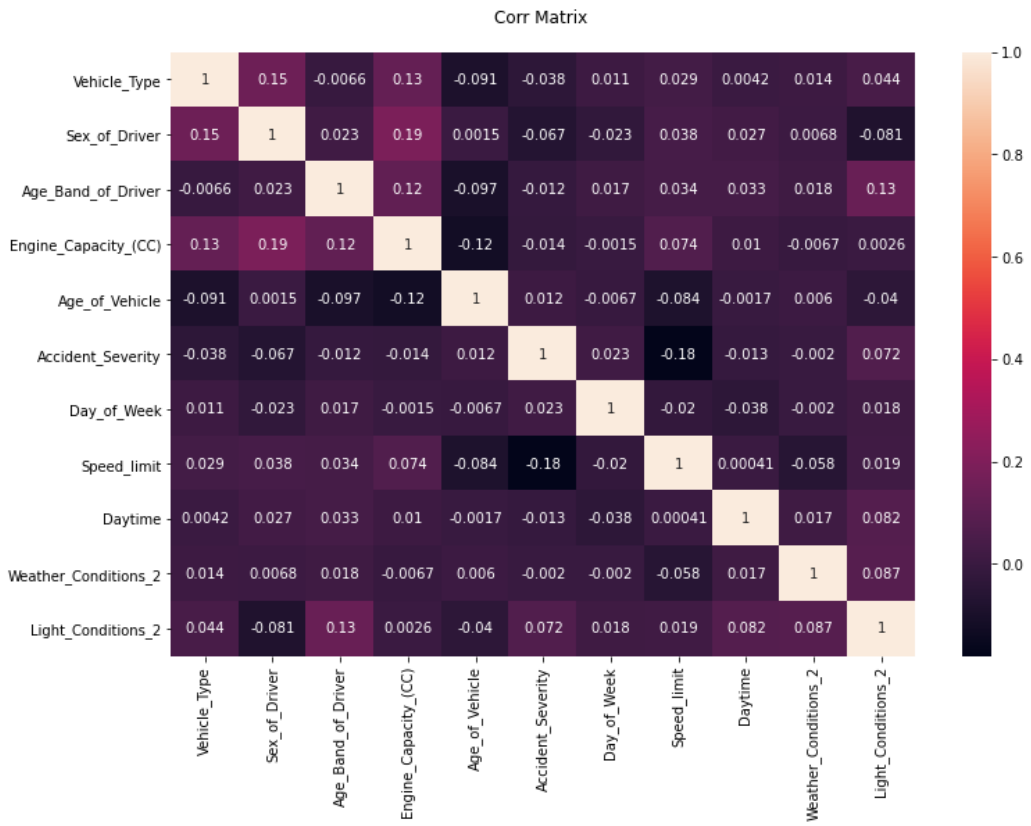
Matriz de correlación



Destacan dos variables **Surface_Conditions_2** y **Weather_Conditions_2** que están estrechamente colineadas. Esto quiere decir que existe una influencia de los datos entre estas dos variables que obviamente tenemos que deshacer.

Al ser dos variables categóricas no podemos aplicar transformaciones como StandardScaler ni PCA, ya que están diseñadas para variables numéricas, por lo que se ha decido eliminar la variable **Surface_Conditions_2,** ya que la condición del asfalto es una variable que dentro de lo que cabe puedes llegar a controlar físicamente, en cambio las condiciones atmosféricas, solo se pueden predecir, por lo que me parece más importante que permanezca en el dataset.

Comprobamos de nuevo la Matrix de Correlación



Finalmente guardamos en un CSV el archivo de los datos con Feature engineering con el nombre df_imbalaced.csv

Machine Learning

Nuestros modelos de machine learning se alimentarán sobre el csv anterior **df_imbalaced,** donde nuestra variable objetivo será Accident_Severity y el resto serán las Features.

Se utilizaun dataset reducido de 100.000 filas aproximadamente ya que no tenemos la capacidad de cómputo necesaria para poder realizar un modelo con muchos datos.

Técnicas usadas

Iteración de modelos de clasificación sobre el dataset desbalanceado

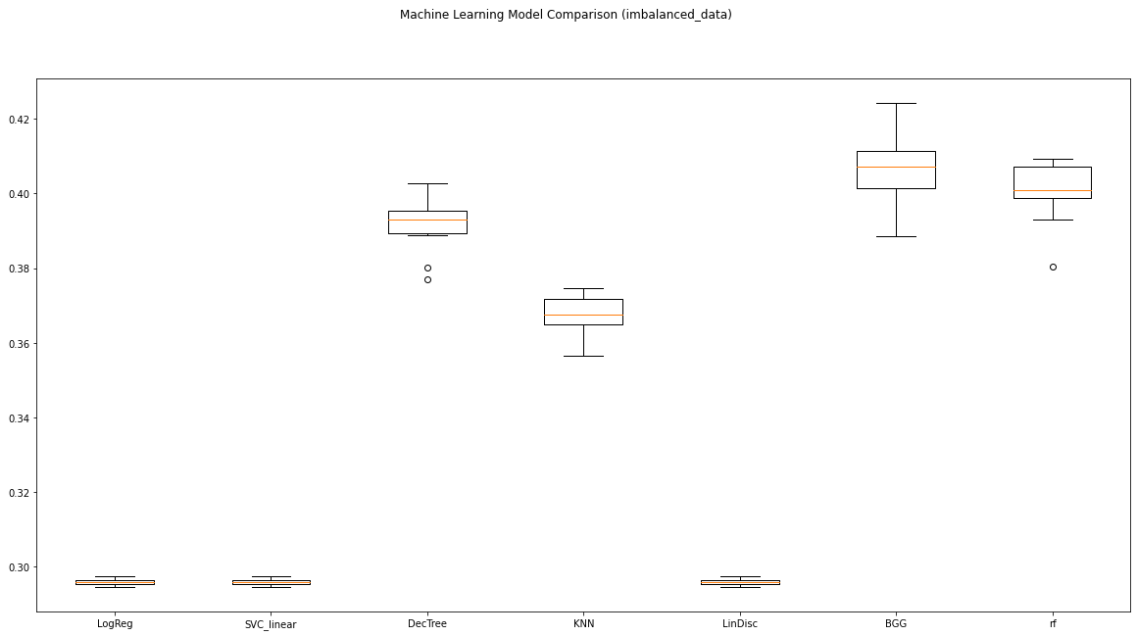
Para realizar el modelo predictivo he decidido empezar iterando sobre varios modelos de clasificación como son:

- LogisticRegression
- LinearSVC
- DecisionTreeClassifier
- KNeighborsClassifier
- LinearDiscriminantAnalysis
- BaggingClassifier
- RandomForestClassifier

He elegido estos modelos por ser los más utilizados en clasificación y ver la diferencia de puntuación entre ellos. Además de ser un buen punto dónde empezar a darse cuenta de qué modelos nos van a ir mejor o peor y así obtener de una manera rápida hacia dónde vamos a enfocarnos

Mediante una validación de datos cruzada (**cross_val_score**) y un n_split = 10 dividimos el dataset en 10 partes. Esta técnica consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación para cada modelo sobre diferentes particiones (10).

La puntuación que he elegido ha sido **f1_macro**, ya que como dataset desbalanceado es la mejor opción aplicar una puntuación que es la media armónica de la precisión y el recall.



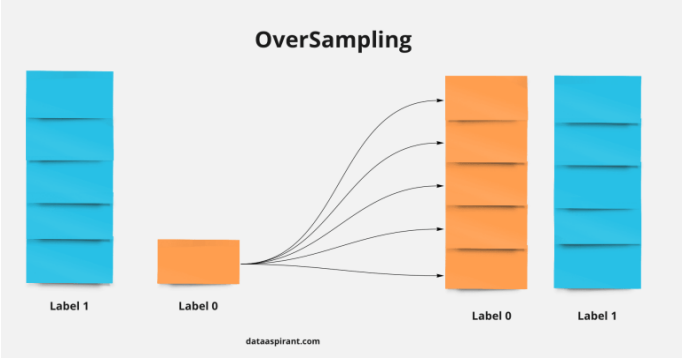
LogReg| Mean=0.295781 STD=0.000854
SVC_linear| Mean=0.295781 STD=0.000854
DecTree| Mean=0.391386 STD=0.007368
KNN| Mean=0.367735 STD=0.005230
LinDisc| Mean=0.295781 STD=0.000854
BGG| Mean=0.407369 STD=0.010500
rf| Mean=0.400516 STD=0.008321

Aunque hemos obtenido una muy baja puntuación, la gráfica nos muestra que hay 4 modelos que destacan por encima del resto: DecTree, KNN, BGG y Randomforest.

Iteración de modelos de clasificación sobre el dataset balanceado (SMOTE)

SMOTE

La técnica de SMOTE (Synthetic Minority Oversampling Technique) se utiliza cuando tenemos un dataset con una variable objetivo donde la multiclase no está balanceada.



En nuestro caso disponemos de 3 categorías para Accident_Severity:

Before Upsampling:

	index	Accident_Severity
0	2	0.794576
1	1	0.177638
2	0	0.027786

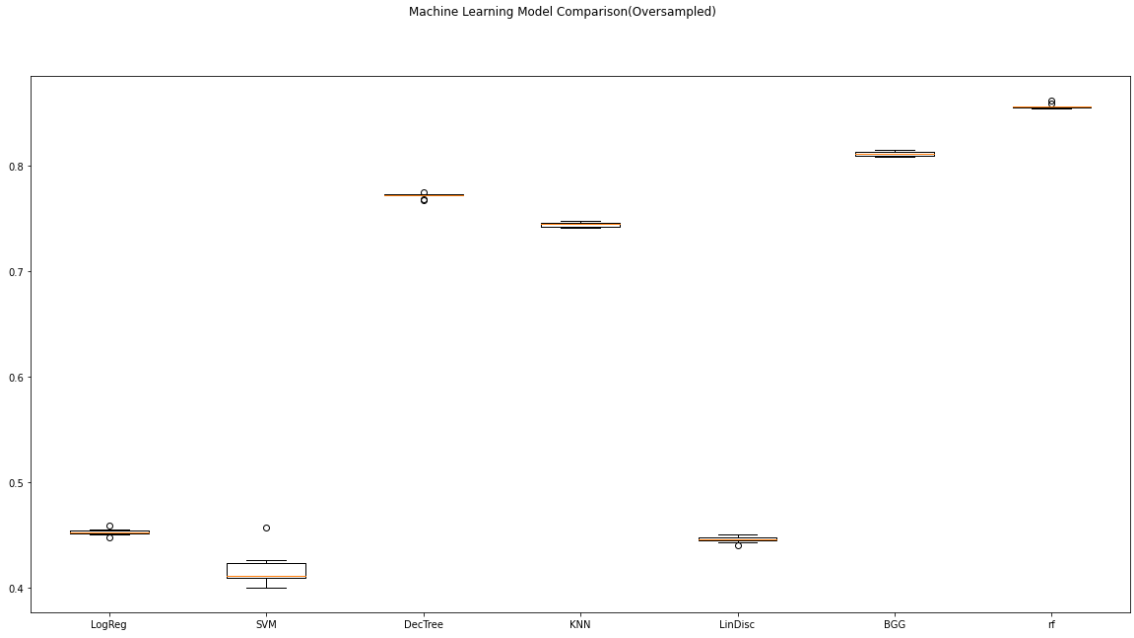
Esto supone un problema porque el modelo no tiene suficientes datos para la categoría Grave y mucho menos para la categoría Fatal.

SMOTE lo que nos proporciona es la creación de datos sintéticos mediante un algoritmo.

Se agregan datos sintéticos entre un caso positivo vecino y el caso positivo original, con lo cual obtenemos un dataset balanceado exactamente igual para cada categoría.

```
After Upsampling with SMOTE:
0    86792
1    86792
2    86792
Name: Accident_Severity, dtype: int64
```

Una vez tenemos el dataset balanceado volvemos a iterar sobre los modelos de clasificación.



LogReg| Mean=0.453294 STD=0.002820
SVM| Mean=0.417583 STD=0.015515
DecTree| Mean=0.771797 STD=0.002141
KNN| Mean=0.744427 STD=0.002306
LinDisc| Mean=0.446255 STD=0.002705
BGG| Mean=0.811545 STD=0.002218
rf| Mean=0.856399 STD=0.002237

Observamos en la gráfica que hemos aumentado mucho más la puntuación con el dataset balanceado.

Analizar e interpretar modelos individualmente (SMOTE)

Oversampling y train-test-split

Pasos antes de evaluar cada modelo:

1. Dividimos en features y target (Accident_Severity).
2. Guardamos variables categóricas y numéricas en listas.
3. Oversampling, devolvemos (X,y) balanceadas.

4. Dividimos en Train test split.

Para cada modelo volvemos a realizar individualmente una evaluación, ya centrándonos en más métricas.
Funciones:

fit_pred_mod()
Parámetros: select_model

Esta función se encargará de tranformar los datos con un Pipeline y Columtransformer, mediante un preprocesado con StandardScaler para las variables numéricas y un Onehotencoder para las variables categóricas. Le pasaremos como input un número entre 1 y 6 ambos incluidos.

eval_score()
Parámetros:
X_train, X_test, y_train, y_test, classifier, yhat, select_model, folder, final_features_list

La segunda función utilizará los parámetros pasados para devolvernos las métricas correspondientes, que son:

- Curva ROC para multiclase
- Matriz de confusión
- Feature importance
- Classification Report

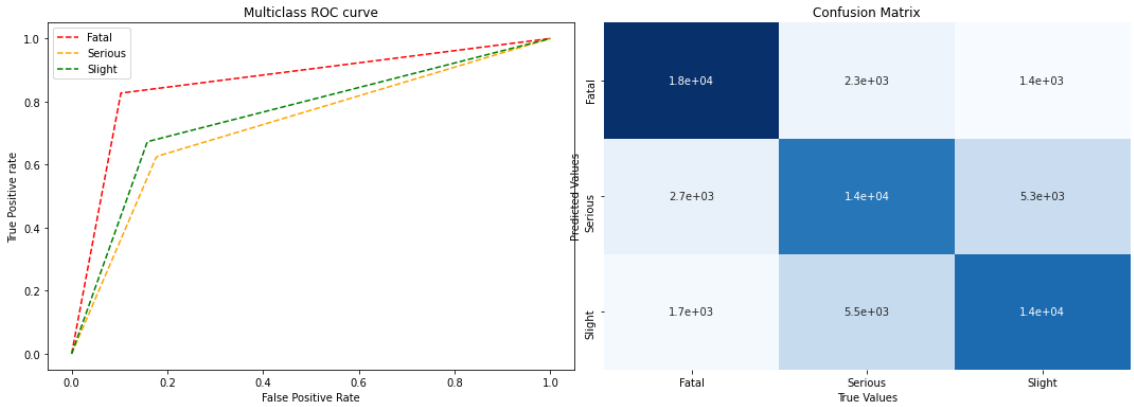
He seleccionado estas métricas porque creo que para un target multiclase es prácticamente obligado hacer una gráfica **ROC_AU_CURVE**, donde el área bajo la curva de cada clase nos dará un valor que cuanto mayor sea, mejor predicción obtendremos.

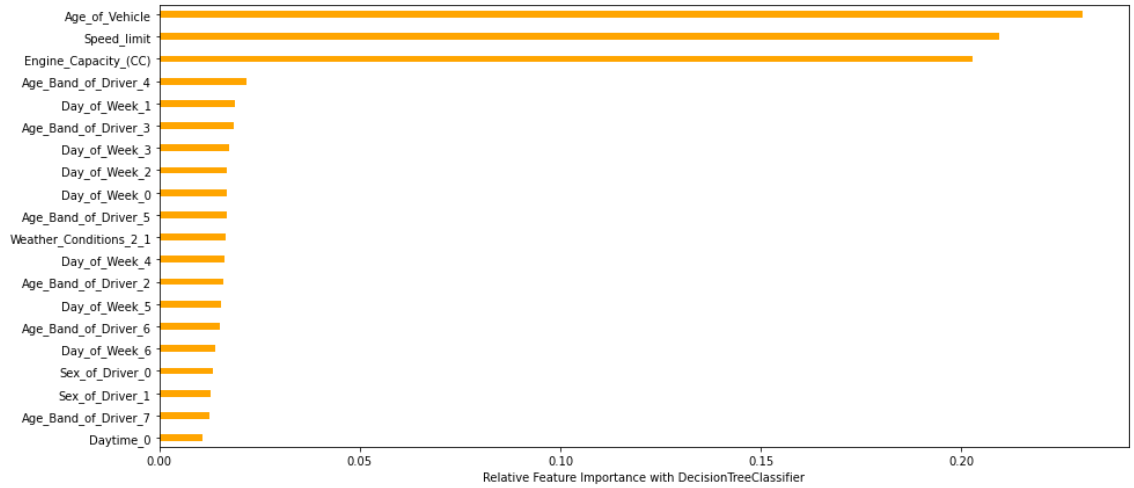
La matriz de confusión nos proporciona información acerca de los verdaderos positivos y negativos (diagonal) como también de los falsos negativos y positivos).
Nuestro objetivo es que destaque siempre la diagonal ya que estaremos en una buena métrica de predicción acertada y fallaremos muy poco.
Cada modelo se guardará en un archivo .pkl en la carpeta models.

Predicted class / Actual class	P	N
P	TP	FN
N	FP	TN

Por último, un classification report donde obtenemos un report de precisión, recall y f1_score por cada clase y una gráfica para ver la importancia de las features en nuestro modelo.

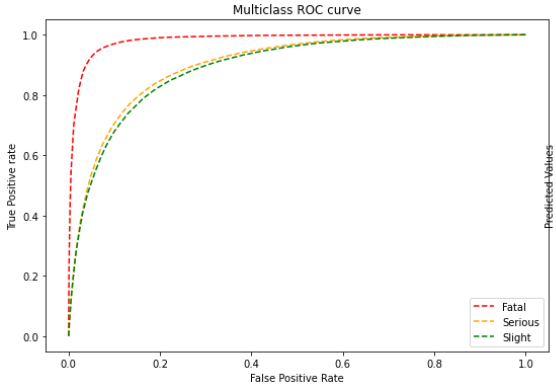
DecisionTreeClassifier

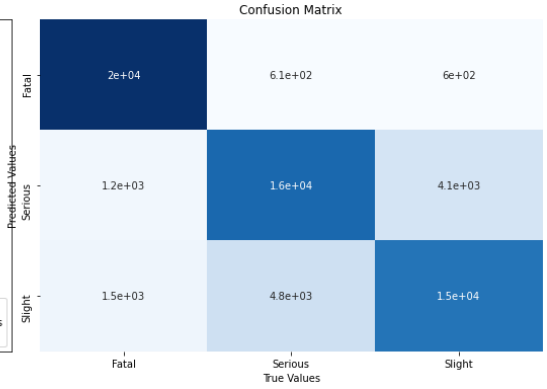


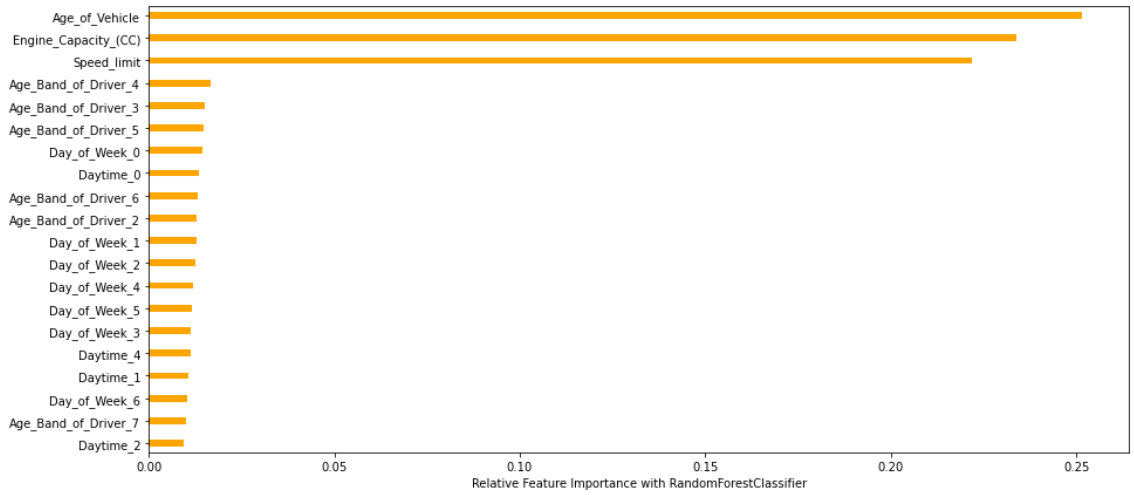



```
DecisionTreeClassifier
model score: 0.706
              precision    recall  f1-score   support

           0       0.80      0.83      0.81     21866
           1       0.63      0.63      0.63     21639
           2       0.68      0.66      0.67     21589

    accuracy                           0.71     65094
   macro avg       0.70      0.71      0.71     65094
weighted avg       0.70      0.71      0.71     65094
```

RandomForestClassifier



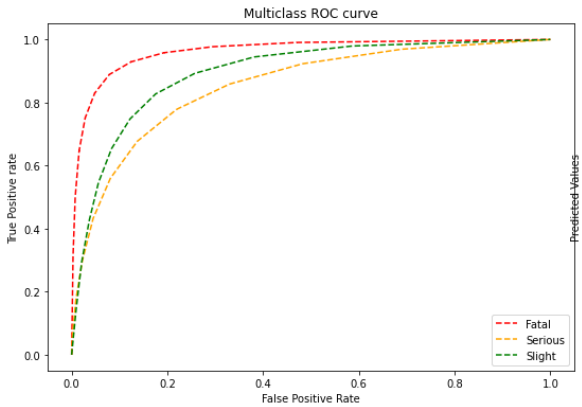


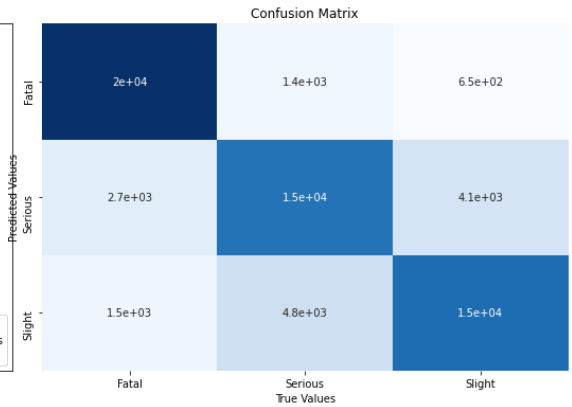


RandomForestClassifier
model score: 0.802

	precision	recall	f1-score	support
0	0.88	0.94	0.91	21678
1	0.75	0.75	0.75	21536
2	0.76	0.71	0.74	21514
accuracy			0.80	64728
macro avg	0.80	0.80	0.80	64728
weighted avg	0.80	0.80	0.80	64728

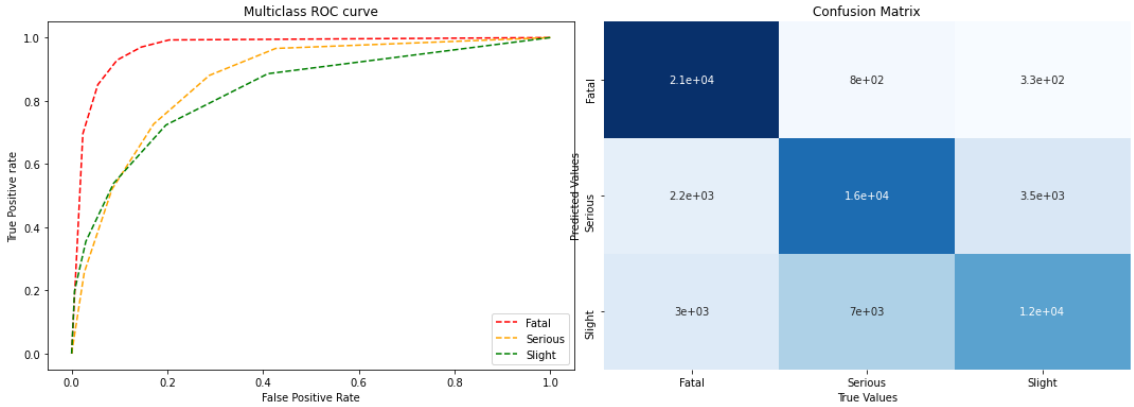
BaggingClassifier





```
BaggingClassifier
model score: 0.767
              precision    recall  f1-score   support

           0       0.82      0.91      0.86     21866
           1       0.70      0.68      0.69     21639
           2       0.76      0.71      0.73     21589

    accuracy                           0.77     65094
   macro avg       0.76      0.77      0.76     65094
weighted avg       0.76      0.77      0.76     65094
```

KNeighborsClassifier




```
KNeighborsClassifier
model score: 0.743
              precision    recall  f1-score   support

           0       0.80      0.95      0.87     21866
           1       0.67      0.74      0.70     21639
           2       0.75      0.54      0.63     21589

    accuracy                           0.74     65094
   macro avg       0.74      0.74      0.73     65094
weighted avg       0.74      0.74      0.73     65094
```

La idea es centrarnos en los accidentes Fatales, seleccionaremos los modelos con más puntuación en la clase 0, es decir la clase Fatal.

Lo que queremos es un modelo que prediga si un accidente va a ser fatal, con una gran posibilidad de acierto.

En cuanto a la puntuación de los modelos.
BaggingClassifier y RandomforestClassifier son los que mejor métricas han reportado en la clase 0.

DecisionTreeClassifier y KNeighborsClassifier han reportados las peores áreas bajo la curva ROC.

Una peculiaridad es que está prediciendo mejor los datos en la clase 0 cuando anteriormente en el dataset desbalanceado era la clase con menos registros, podría ser que tengamos aún alguna variable sesgada que hace que el modelo tenga más falsos negativos de los que debería, o que el propio Oversampling esté produciendo algún tipo de sesgado interno.

En cuanto a las features, Age of vehicle es la indiscutible ganadora en importancia, seguido de Engine Capacity y speed limit.

Es claro que estas 3 variables juegan un papel muy importante en el entrenamiento del modelo.

Precisión

Métrica que utiliza los falsos positivos para la evaluación, casos positivos que el modelo identifica como negativos.

No nos vale ya que si queremos predecir un accidente Fatal para tomar medidas en un punto crítico de la ciudad nos va a dejar ubicaciones de accidentes fatales sin predecir y no queremos eso ya que el objetivo es detectar justamente estos puntos.

Recall

Métrica que utiliza los falsos negativos para la evaluación, casos negativos que detecta como positivos.

Desde un punto de vista hipotético si tuviéramos que emplear dinero público para reforzar una ubicación de accidentes críticos, si el modelo no evalúa bien, nos estaría prediciendo ubicaciones que no son fatales, aunque dentro de lo malo es preferible esta última métrica ya que con un estudio posterior podríamos asegurarnos de la inversión en ese punto.

F1_score

Métrica utilizada en nuestro caso.

El valor F1 asume que nos importa de igual forma precisión y recall.

Se utiliza para combinar las medidas de precisión y recall en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

Podemos ver que el modelo con más puntuación en f1_score es Randomforestclassifier.

Una vez ya hemos generado los modelos anteriores con las respectivas puntuaciones, vamos a realizar una búsqueda de parámetros para cada uno de los modelos, intentado encontrar los mejores para una puntuación superior.

Para ello utilizaremos un pipeline con sklearn y gridsearchcv.

GridsearchCV iterará sobre todos los parámetros que le pasamos previamente y nos dará la mejor puntuación para el modelo correspondiente.

Iteración sobre parámetros con GridsearchCV

Funciones:
param_search()
Parámetros (select_model, dump=None)

La siguiente función para_search() utiliza un pipeline para cada modelo juntamente con GridsearchCV, que su propósito es realizar una validación cruzada mediante una selección de parámetros que previamente hemos de pasar para cada modelo.

GridsearchCV realizará una combinación con cada uno de los parámetros reportándonos aquellos que han sido más satisfactorios para el modelo mediante la métrica que le hemos pasado (f1_macro).

Como vamos a hacer una búsqueda para cada modelo, hemos de crear 4 diccionarios con los parámetros que queremos probar.

```
#MODEL PARAMETERS
#K-Nearest - Neighborg
knn_grid = {
    'classifier__n_neighbors' : list(range(5,17,2))
    }
#DecisionTreeclassifier
tree_grid = {
"classifier__min_samples_split" : list(range(2,8,2)),
"classifier__criterion" : ['gini','entropy']
}

#Bagging Classifier
bag_grid = {
 'classifier__n_estimators': [5, 10, 20 , 30 , 40]
}
#Random Forest
rf_grid = {
    "classifier__n_estimators" : range(100,200,20),
    "classifier__criterion" : ["gini","entropy"]}
```

La ejecución de la función se ha realizado mediante la herramienta Google Ai Platform con notebook Jupyter y una máquina virtual de vcpu de 16 núcleos, tardando aproximadamente unas 5 horas.
Cada uno de los mejores modelos se guardan en un archivo .pkl para utilizarlo posteriormente.
Resultados Gridsearchcv

El mejor resultado ha sido RandomforestClassifier con un 0.82 en f1_macro_score seguido de BaggignClassifier con un 0.81. El resto de modelos podemos descartarlos ya que nos dan una puntuación por debajo de 0.8.

No podemos asegurar que sea el mejor resultado ya que hay muchísimos más parámetros para probar y combinar, pero esto resultaría en un mayor número de iteraciones con un mayor tiempo de espera para cada modelo siendo así una limitación los 16vcpu de AI Platform de Google, por eso mismo me he centrado en los parámetros que a mi juicio son más representativos o tienen un mayor impacto en el modelo.

Curva de validación

En Machine Learning, la validación del modelo la usamos para medir la efectividad de un modelo. Un buen modelo de aprendizaje automático no solo se ajusta muy bien a los datos de entrenamiento, sino que también se puede generalizar a nuevos datos de entrada.

La curva de validación es una técnica gráfica que se puede utilizar para medir la influencia de un solo hiperparámetro. Al observar esta curva, puede determinar si el modelo está desajustado, sobreajustado o simplemente correcto para algún rango de valores de hiperparámetros. Así podemos ver como influye cada uno de los parámetros.

En este caso he seleccionado RandomForestClassifier ya que es el modelo que anteriormente nos ha reportado más puntuación y es el que he elegido para representar las curvas de validación.

El siguiente código muestra la Curva de validación con RandomforestClassifier para 4 hiperparámetros:

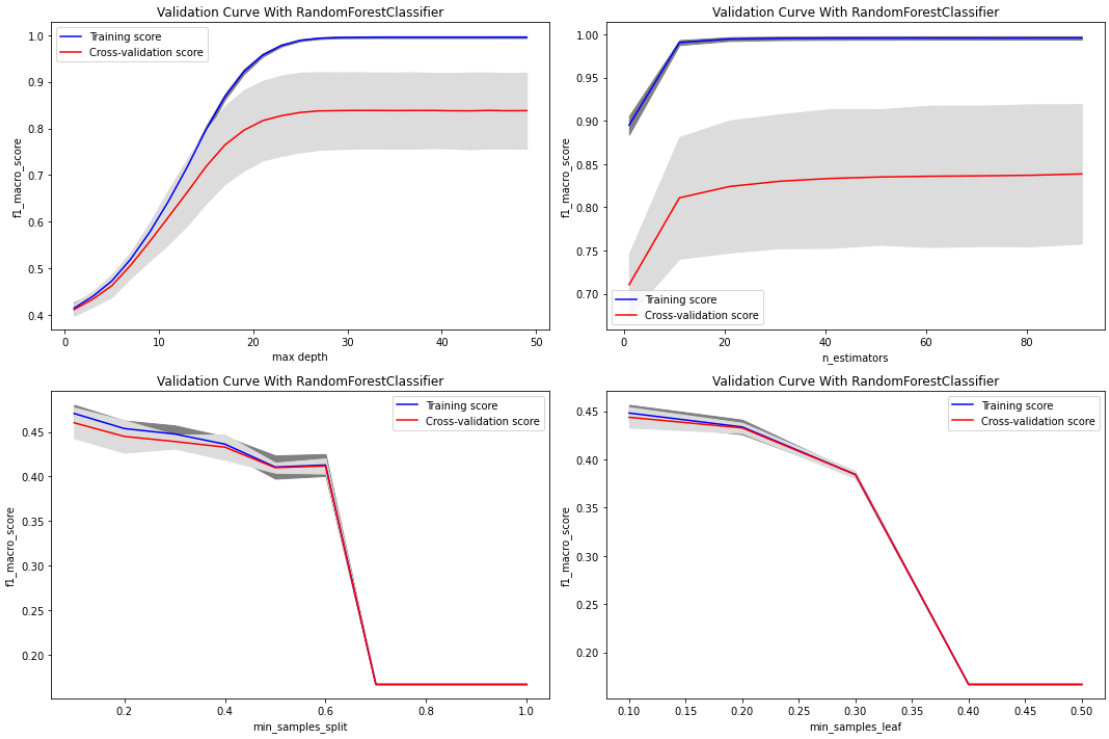
- max_depth
- n_estimators
- min_samples_split
- min_samples_leaf

Normalmente la curva de validación se hace directamente con las features y el target, sin hacer split, ya que la misma herramienta internamente realiza este proceso por nosotros.

Así que le pasamos los parámetros:

- X,y
- Nombre del parámetro a evaluar = max_depth , n_estimators , min_samples_split, min_samples_leaf.
- Rango del parámetro
- CV = 5
- estimator = Randomforestclassifier
- scoring = f1_macro

Se realiza un plot para cada hiperparámetro con validation_curve



Para el parámetro **max_depth** el gráfico muestra claramente que el aumento de la profundidad del árbol en las primeras etapas da como resultado una mejora correspondiente tanto en los datos de entrenamiento como en los conjuntos de prueba.

Este comportamiento continúa hasta una profundidad de alrededor de 10 niveles, después de lo cual se muestra que el modelo se ajusta en exceso al conjunto de datos de entrenamiento a costa de un peor rendimiento en el conjunto de datos de reserva.

En cuanto al parámetro **n_estimators** nos reporta un peor escenario donde se puede apreciar mucho **Overfitting** desde el principio, ya que el conjunto de test y train nunca llegan a aproximarse, quedando 0.2 puntos entre una y otra.

Podemos observar que se produce **Underfitting** cuando se realiza una validación con los parámetros **min_samples_split** y **min_samples_leaf** para datos de test y train.

Test_data

Se ha creado un notebook llamado **test_data.ipynb** donde creamos un dataset de 100.000 filas y añadimos las columnas latitud y longitud para poder cargarlo posteriormente en el Frontend y visualizar las predicciones en un mapa.

*Es necesario ejecutar el notebook antes de realizar el Front-end.

RESULTADOS

Métricas del modelo Randomforest con GridsearchCV

En este apartado nos centramos en comprobar cómo funciona el modelo de Randomforest generado con GridsearchCV con los datos de train y test que ya teníamos precargados en el notebook y también los del modelo sin GridsearchCV.
Cargamos el mejor modelo randomforestclassifier con los mejores parámetros proporcionados con GridsearchCV, para volver a evaluar sobre el conjunto de train y test.

```
clf = joblib.load("gsearchcv/RandomForestClassifier.pkl")
```

```
clf.score(X_train,y_train)
```

```
0.5273417525373828
```

```
y_hat = clf.predict(X_train)
```

```
print(classification_report(y_train,y_hat))
```

```
              precision    recall  f1-score   support

           0       0.69      0.35      0.46     64455
           1       0.43      0.44      0.44     64568
           2       0.54      0.80      0.64     64582

    accuracy                           0.53    193605
   macro avg       0.55      0.53      0.51    193605
weighted avg       0.55      0.53      0.51    193605
```

```
y_hat = clf.predict(X_test)
```

```
print(classification_report(y_test,y_hat))
```

```
              precision    recall  f1-score   support

           0       0.69      0.35      0.46     21592
           1       0.43      0.44      0.43     21479
           2       0.53      0.80      0.64     21465

    accuracy                           0.53     64536
   macro avg       0.55      0.53      0.51     64536
weighted avg       0.55      0.53      0.51     64536
```

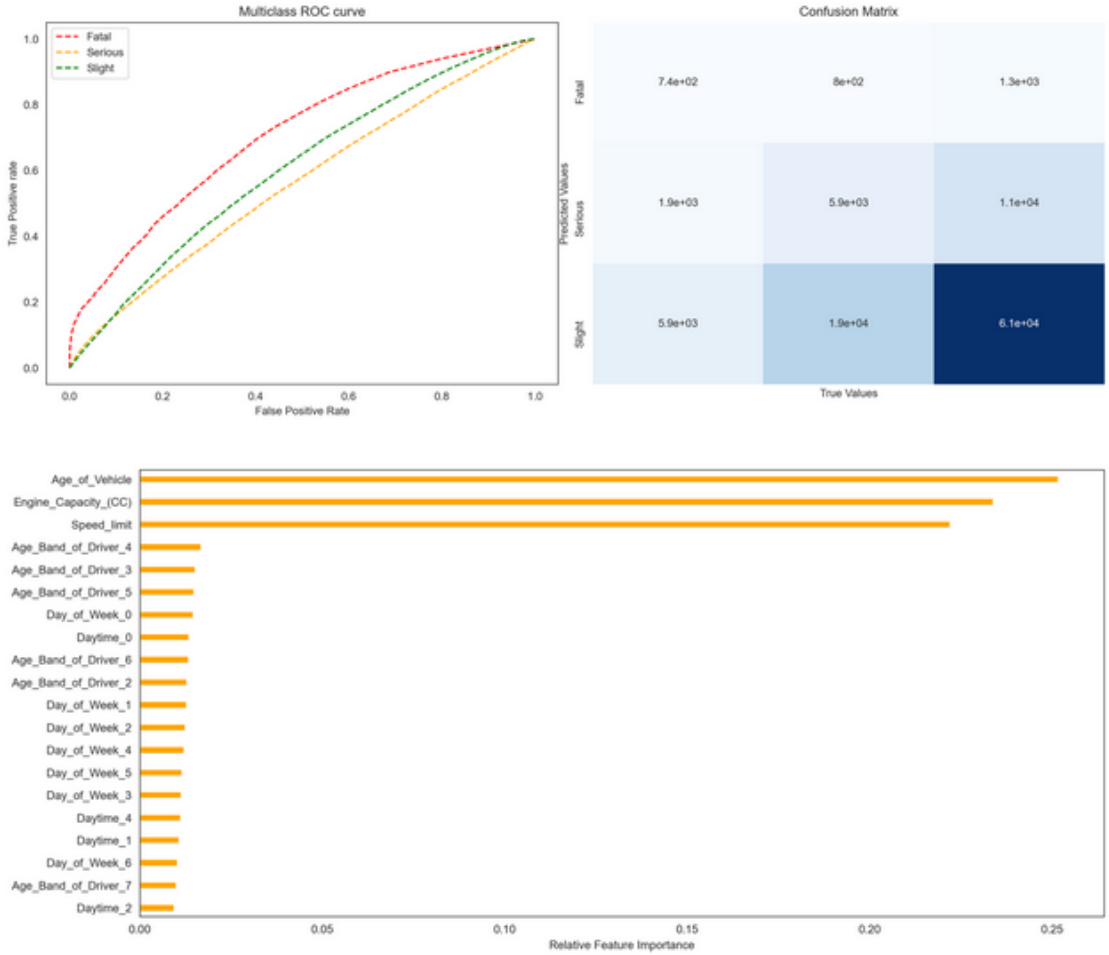
Al volver a comprobar la puntuación del modelo con train y test no obtenemos un buen resultado, a pesar de que debería darnos un resultado muy parecido o idéntico al que hemos

obtenido cuando realizamos la búsqueda de parámetros, ya que es el mismo split que el que hemos hecho durante todo el proceso.

Además mediante la curva de validación hemos visto que se produce **Overfitting** cuando aplicamos hiperparámetros únicos como **max_depth** y **n_estimators**. Y **Underfitting** cuando jugamos con **min_samples_split**, **min_samples_leaf**.
Por ello descartamos el modelo de GridsearchCV para el Frontend.

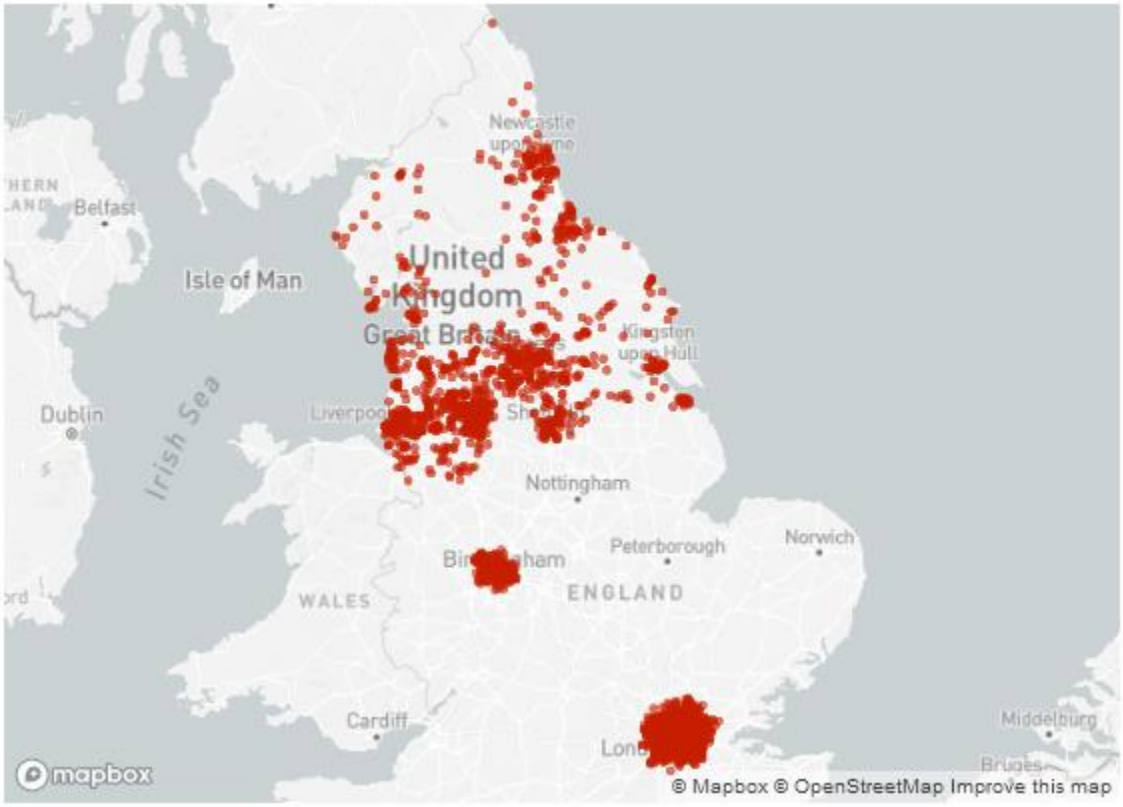
Al tener este resultado se decide añadir los 3 modelos sin hiperparámetros al frontend para evaluar como de bueno es cada modelo.

Frontend RandomForestclassifier (default)



Predicción

Valor real



Los modelos no están teniendo una buena predicción, ya que las gráficas de curva roc y la matriz de confusión demuestran que no es capaz de predecir ninguna de las clases con un f1_score ni siquiera de 0.8. Se asemeja bastante al modelo con Gridsearchcv, que nos devolvía valores muy bajos.

Por tanto, los modelos no se han ajustado bien al dataset, proporcionando valores incorrectos. Como podemos ver en los mapas, la predicción no se realiza como debiera, dándonos puntos donde se concentran demasiado los valores fatales predichos si los comparamos con el valor real.

Esto no es bueno ya que habrá sitios que nunca haya habido accidentes ni se reúnan las condiciones para que sucedan con mayor probabilidad, por lo tanto, los modelos entrenados no nos sirven.

CONCLUSIONES

Personal:

La conclusión que extraigo de todo el proyecto es que ha sido muy enriquecedor el estar luchando día a día, investigando, descubriendo nuevas herramientas ya que es parte de un Data Scientist la actitud de ir más allá e implementar nuevas mejoras. El plantearte como estás enfocando el proyecto y ser muy minucioso en el trabajo diario.

Por eso creo que el haber trabajado con un dataset tan complicado ha tenido sus ventajas en cierta manera para poder aprender más.

Resultados:

En cuanto a los resultados tengo varias hipótesis que quiero comentar.

Los resultados no son nada buenos finalmente, por ello quiero hacer una reflexión y compartir una serie de razonamientos por los cuales ha sucedido esto.

El primer lugar creo que al ser un dataset tan sesgado esto ha dificultado el buen aprendizaje de los modelos a los datos. Ya que muchas de las variables se correlacionan y aún habiendo hecho el preprocesado cuando un target se encuentra muy influenciado por muchas variables creo bajo mi punto de vista que es más difícil que el modelo generalice bien.

Por otro lado, el hecho de haber realizado un Oversampling tiene sus ventajas e inconvenientes, en mi caso al “crear” nuevos datos es posible que estos datos también generen de alguna manera un sesgo o una correlación intrínseca sin que nos demos cuenta y haya hecho que el modelo no acabe de estar a la altura.

Por último, el error humano es seguramente el que ha causado mayor porcentaje de malas decisiones en el proyecto, ya que algunos procedimientos puede que estén mal planteados desde inicio y por ello no se han obtenido buenos resultados, ya sea por falta de experiencia o desconocimiento.

MANUAL FRONT END

La interfaz de front end está basada en streamlit.

Se estructura en un menú con 3 categorías.

- Home
- Visualización
- Modelado

En la categoría Home se puede ver una breve descripción de los datos y el objetivo final que persigue el proyecto.

La pestaña de visualización nos llevará a una interfaz donde mediante un desplegable podemos elegir el gráfico y mediante un radiobutton poder seleccionar si queremos obtener la información de ese gráfico de maneral total o anual para cada gráfico.

Por último, en el apartado de modelado, primero de todo nos saldrá un botón para cargar un archivo CSV. Este archivo lo hemos generado previamente mediante el notebook de **test_data.ipynb**.

Una vez cargado el archivo .csv, podemos seleccionar el modelo que queremos usar y ver su comportamiento.

Los resultados serán una gráfica de curva ROC multiclase, una matriz de confusión y la importancia de las features para ese modelo en concreto.

Se realizarán las predicciones sobre los accidentes fatales y se mostrará en un mapa las predicciones obtenidas, y los datos reales para ver una comparación.

BIBLIOGRAFÍA

Links

https://shandou.medium.com/export-and-create-conda-environment-with-yml-5de619fe5a2

https://machinelearningmastery.com

https://iaml.it/blog/optimizing-sklearn-pipelines

https://coderzcolumn.com/tutorials/data-science/cufflinks-how-to-create-plotly-charts-from-pandas-dataframe-with-one-line-of-code#7cuff

Plotly and cufflinks — An interactive Python visualization tool for EDA and Presentations | by Israel Aminu | Analytics Vidhya | Medium

Matplotlib- seaborn visualization

Hyperparameters Tuning Using GridSearchCV And RandomizedSearchCV

Interactive Plots with Plotly and Cufflinks on Pandas Dataframes | by Ozan Bulum | Medium

sklearn.model_selection.GridSearchCV — scikit-learn 0.24.2 documentation

Building and optimizing pipelines in scikit-learn (Tutorial) | Italian Association for Machine Learning

Preprocessing with sklearn: a complete and comprehensive guide | by Steven Van Dorpe | Towards Data Science