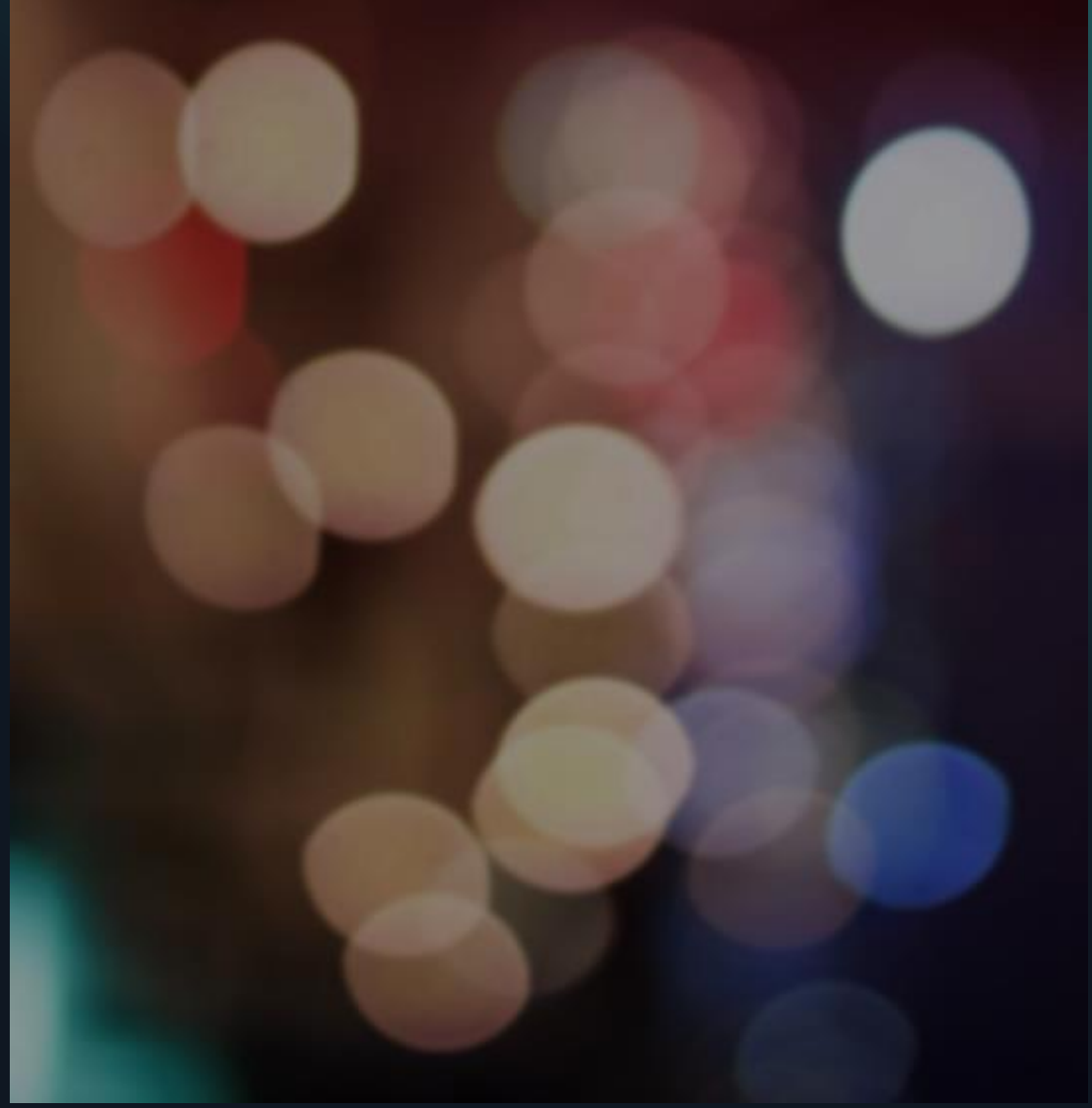
PREDICCIÓN DE MUERTES EN ACCIDENTES

Javier Tornel, Álvaro Muela, Sergio Castell

ÍNDICE



	0	1	2	3	4
C_YEAR	1999	1999	1999	1999	1999
C_MNTH	1	1	1	1	1
C_WDAY	1	1	1	1	1
C_HOUR	20	20	20	08	08
C_SEV	2	2	2	2	2
C_VEHS	02	02	02	01	01
C_CONF	34	34	34	01	01
C_RCFG	UU	UU	UU	UU	UU
C_WTHR	1	1	1	5	5
C_RSUR	5	5	5	3	3
C_RALN	3	3	3	6	6
C_TRAF	03	03	03	18	18
V_ID	01	02	02	01	99
V_TYPE	06	01	01	01	NN
V_YEAR	1990	1987	1987	1986	NNNN
P_ID	01	01	02	01	01
P_SEX	M	M	F	M	M
P_AGE	41	19	20	46	05
P_PSN	11	11	13	11	99
P_ISEV	1	1	2	1	2
P_SAFE	UU	UU	02	UU	UU
P_USER	1	1	2	1	3

0- PROBLEMA

CLASIFICAR LOS ACCIDENTES DE VEHÍCULOS EN FUNCIÓN DE SI HAY O NO MUERTOS

- Tenemos un dataset que contiene datos relativos a accidentes de muchos tipos de vehículos en Canadá.
- Nuestra variable objetivo será C_SEV, que indica si en el accidente hay muertos (1) o no (2).
- Aplicaremos distintos métodos para llevar a cabo una clasificación que nos permita predecir si hay o no muertos.
- Vemos que hay unos valores definidos por U, UU, UUUU, N, NN, NNNN, Q, QQ, QQQQ, X, XX, XXXX que son equivalentes a los valores perdidos, por lo que sustituiremos estos valores por NaN mediante el comando 'replace'.

1- PREPROCESSING

CODIFICACIÓN DE VARIABLES

Se puede observar que todas las variables del dataset son numéricas y de tipo categórico excepto la variables del género, con valores M y F según la persona que conduce sea hombre o mujer, por lo que convertimos dichos valores en valores dicotómicos 1 y 2 respectivamente mediante la siguiente función.

```
from sklearn.preprocessing import LabelEncoder
P_SEX=car['P_SEX'].values
enc = LabelEncoder()
label_encoder = enc.fit(P_SEX)
P_SEX = label_encoder.transform(P_SEX) + 1
label_dict = {1: "M", 2:"F"}

car['P_SEX']=P_SEX

P_SEX
```

IMPUTACIÓN DE MISSINGS

Una vez que todas nuestras variables son todas categóricas y numéricas, queremos ver la cantidad de valores perdidos que tenemos e imputarlos. Vemos como funciona para alguna de las variables de la base de datos.

```
car.isna().sum()
```

1

C_YEAR	0
C_MNTH	385
C_WDAY	1323
C_HOUR	59409
C_SEV	0
C_VEHS	544
C_CONF	463999
C_RCFG	648946
C_WTHR	102988
C_RSUR	248668
C_RALN	463312
C_TRAF	305501

2

IMPUTACIÓN POR LA MODA

Como hay variables que tienen hasta un millón de valores perdidos, no podemos eliminar directamente las filas con missings, por lo que, al ser todas las variables categóricas, imputaremos los valores previamente escritos como NA por la moda, es decir, por el dato que más se repita en la columna.

3

C_YEAR	0
C_MNTH	0
C_WDAY	0
C_HOUR	0
C_SEV	0
C_VEHS	0
C_CONF	0
C_RCFG	0
C_WTHR	0
C_RSUR	0
C_RALN	0
C_TRAF	0

ELEMENTOS DUPLICADOS

Queremos comprobar si hay elementos duplicados en nuestro dataset, que no aportan información nueva pero aumenta las observaciones, teniendo un mayor coste computacional y, en caso positivo, los eliminamos.

```
duplicate_rows_car = car[car.duplicated()]
print("numero de duplicados: ", duplicate_rows_car.shape)
```

```
numero de duplicados:  (6591, 22)
```

```
car=car.drop_duplicates()
```

Confirmamos que hemos eliminado las filas duplicadas:

```
duplicate_rows_car = car[car.duplicated()]
print("numero de duplicados: ", duplicate_rows_car.shape)
```

```
numero de duplicados:  (0, 22)
```

OUTLIERS

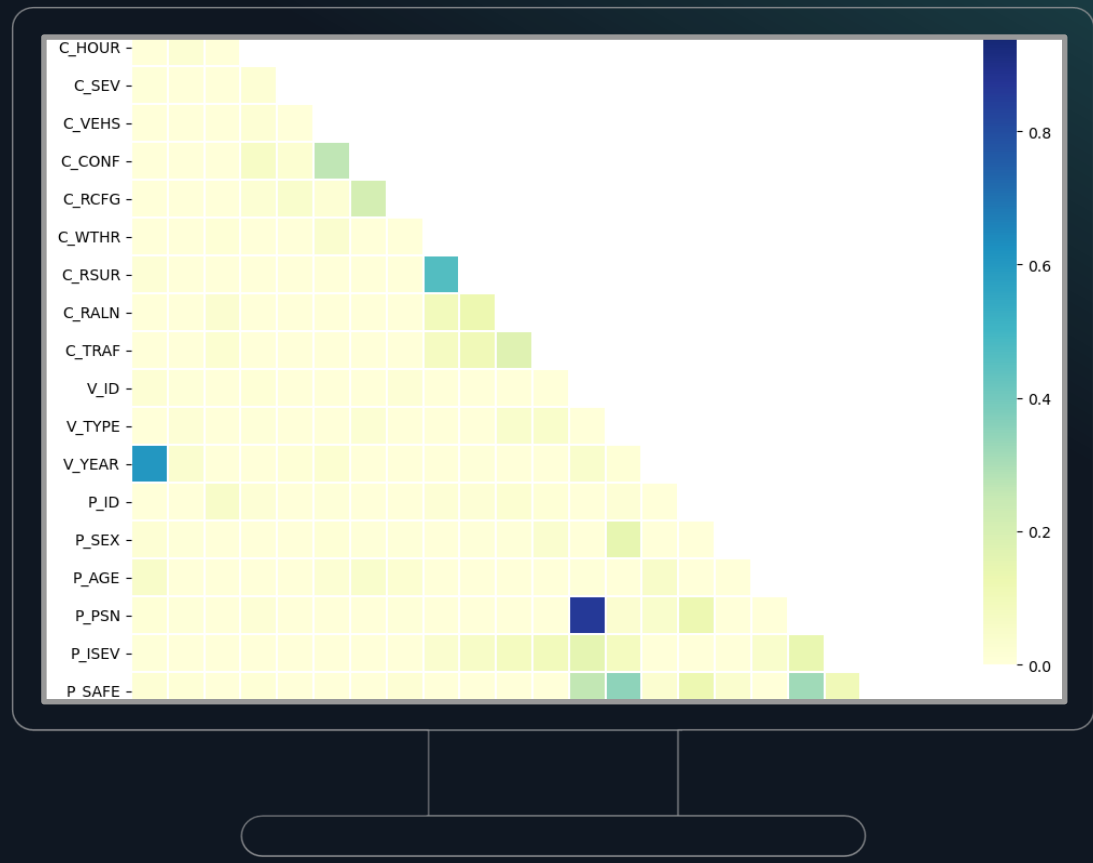
En cuanto a la imputación de valores atípicos, queremos ver si existen filas con outliers en muchas de sus variables para eliminarlas. No eliminamos como tal los datos que sean outliers porque podría darse el caso de que quitásemos muchas observaciones y el estudio se viese perjudicado. En nuestro caso, eliminaremos aquellas filas que contengan 2 o más outliers.

```
def filtrar_outlier_tukey(x):
    q1 = np.percentile(x, 25)
    q3 = np.percentile(x, 75)
    iqr = q3 - q1
    print("[q1=%f, q3=%f, iqr=%f]" % (q1, q3, iqr))

    floor = q1 - 1.5*iqr
    ceiling = q3 + 1.5*iqr
    print("[floor=%f, ceiling=%f]" % (floor, ceiling))

    outlier_indices = list(x.index[(x < floor)|(x > ceiling)])
    outlier_values = list(x[outlier_indices])

    return outlier_indices, outlier_values
```

```
for i in range(22):
    outlier_indices, outlier_values = filtrar_outlier_tukey(car[l[i]])
for i in range(22):
    contador=0
    borrar=0
    j=0
    while j<=(i+1)*21:
        if outlier_indices[j] in range(i*22, 22+(i)*22):
            contador=contador+1
            j=j+1
        else:
            j=j+1
        if contador>=2:
            if borrar==0:
                car.drop([i], axis=0, inplace=True)
            borrar=1
car
```

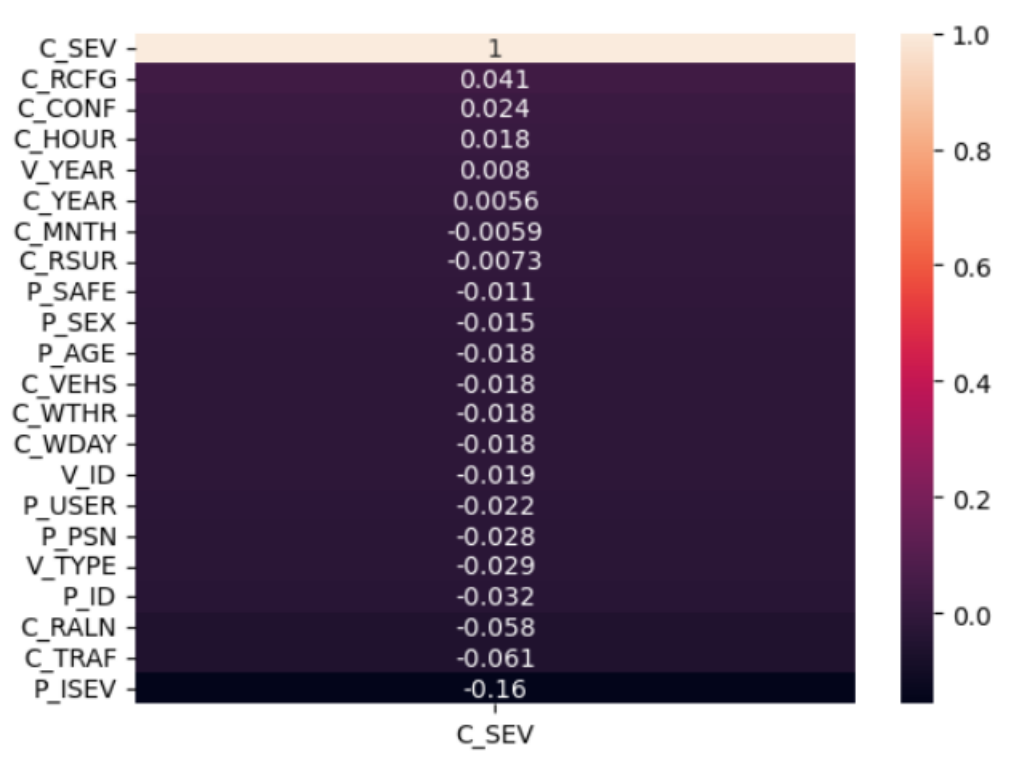
CORRELACIONES

A partir de la matriz de correlaciones se pueden observar las relaciones entre las diferentes variables, lo que nos permite ver de forma general como influyen unas variables en otras. En este caso son bastante bajas, lo que nos indica que una variable no depende casi de las restantes.



SELECCIÓN DE VARIABLES

A. CORRELACIONES CON LA VARIABLE OBJETIVO C_SEV

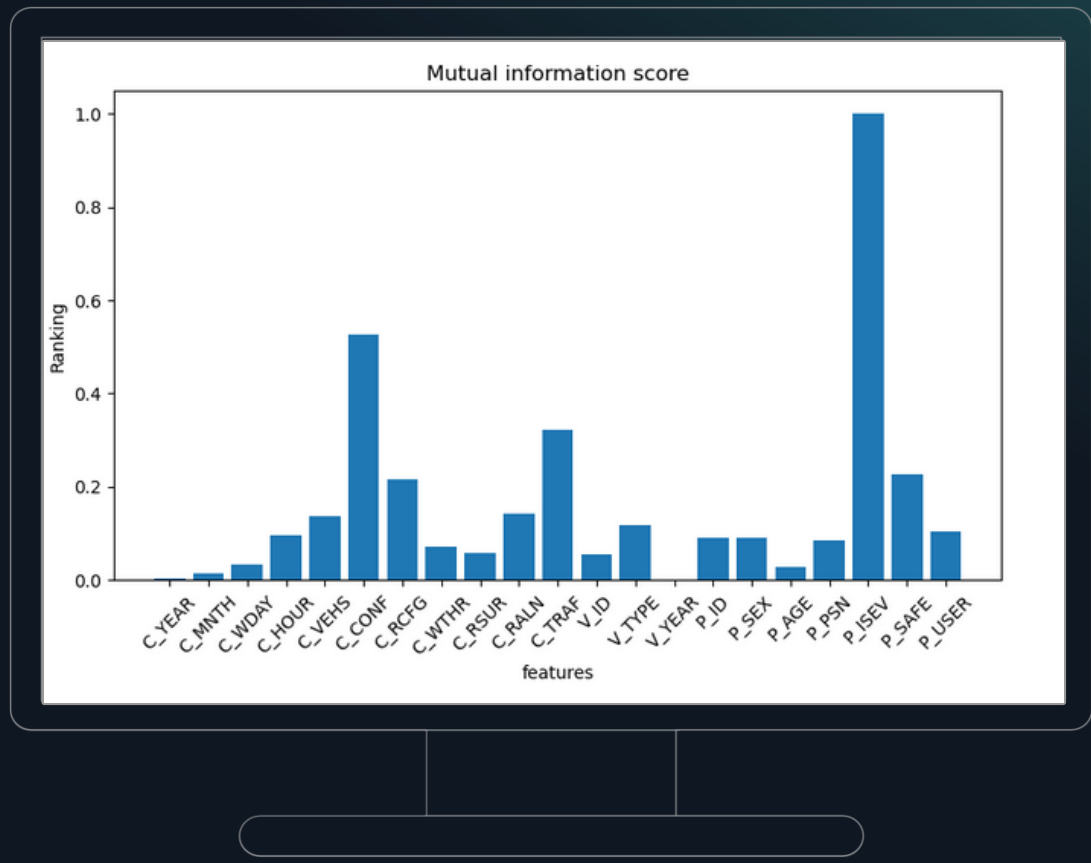


- Una primera forma de seleccionar las variables es mediante las correlaciones con la variable objetivo (C_SEV).
- En este caso no será un criterio útil debido a las bajas correlaciones de todas las variables, por lo que tendremos que buscar otros métodos de selección.
- Emplearemos en este caso el criterio del filtrado de variables.

SELECCIÓN DE VARIABLES

B. FILTRADO MEDIANTE INFORMACIÓN MUTUA

- Este método es útil para datasets con muchas observaciones.
- Nos quedamos con las siguientes variables:
  - C_CONF
  - C_RCFG
  - C_RALN
  - C_TRAF
  - V_TYPE
  - P_ISEV
  - P_SAFE
- A estas variables le añadimos la variable objetivo C_SEV



BALANCEO DE DATOS

De las casi seis millones de observaciones, tenemos que casi en su totalidad no hay muertes en los accidentes, por lo que claramente la variable objetivo está desbalanceada. Por ello, a continuación llevamos a cabo un undersampling y un oversampling para corregir tal desbalanceo. Mediante el método del oversampling, añadimos observaciones "inventadas" del dato con pocos valores, para tener entonces un mayor balanceo. Por tanto, será un método menos realista debido a la inclusión de observaciones que no existen, aunque en principio aporta mejores resultados a la hora de clasificar. Por otro lado, el undersampling elimina observaciones de la clase mayoritaria, quedándonos los valores balanceados. Es más realista pero tendrá peores resultados al estar perdiendo información. Nos quedaremos con la muestra balanceada mediante undersampling, ya que mediante el otro método, habría demasiadas observaciones y sería computacionalmente complicado de trabajar.

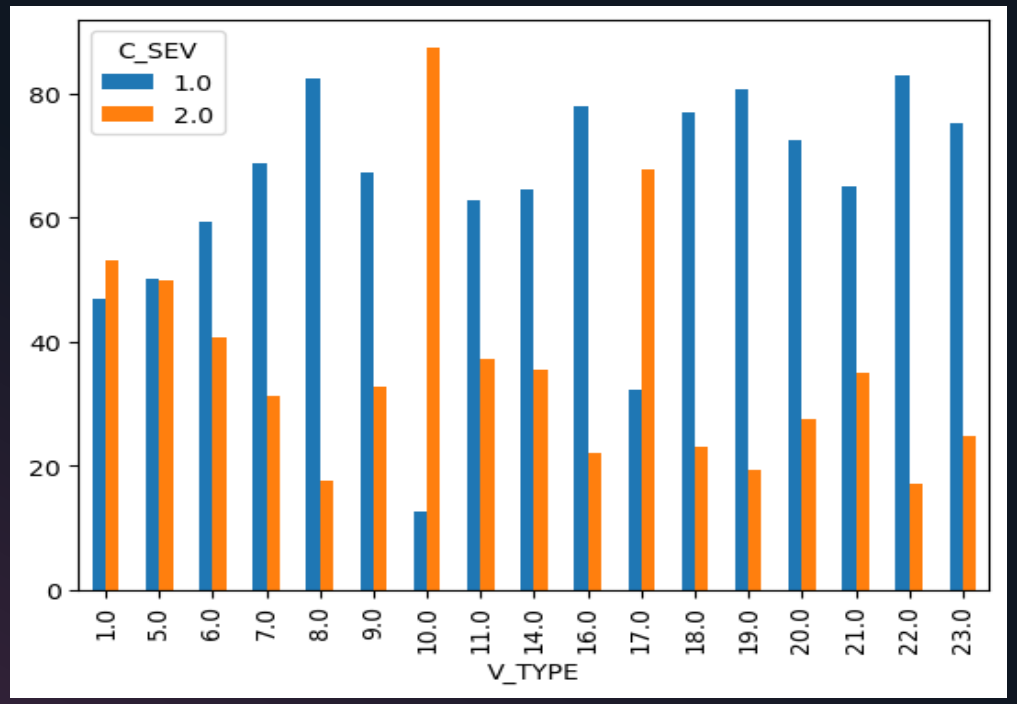
```
car['C_SEV'].value_counts()

2.0    5755198
1.0      98616
```

DATASET FINAL CON LOS DATOS BALANCEADOS

	C_SEV	C_CONF	C_RCFG	C_RALN	C_TRAF	V_TYPE	P_ISEV	P_SAFE
0	1.0	4.0	2.0	4.0	18.0	6.0	3.0	2.0
1	1.0	1.0	2.0	1.0	6.0	1.0	1.0	2.0
2	1.0	1.0	2.0	1.0	6.0	1.0	3.0	2.0
3	1.0	1.0	2.0	1.0	1.0	1.0	2.0	2.0
4	1.0	1.0	2.0	1.0	1.0	1.0	3.0	2.0
...	...	...	...	...	...	...	...	...
197227	2.0	33.0	2.0	1.0	1.0	1.0	1.0	2.0
197228	2.0	33.0	1.0	1.0	18.0	1.0	1.0	2.0
197229	2.0	21.0	3.0	1.0	18.0	1.0	1.0	2.0
197230	2.0	4.0	1.0	1.0	18.0	1.0	2.0	1.0
197231	2.0	21.0	1.0	2.0	18.0	1.0	2.0	2.0

ALGUNOS GRÁFICOS

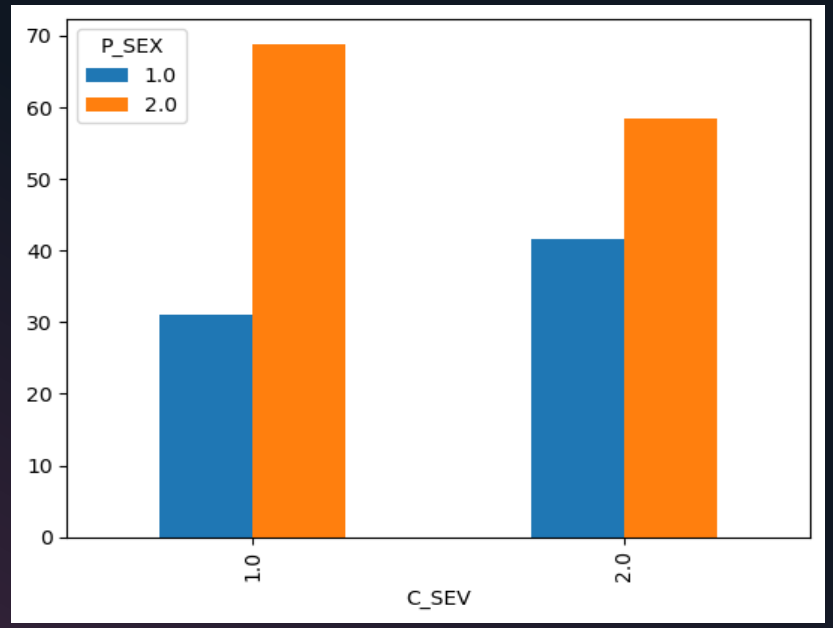
¿Qué tipos de vehículos son más y menos propensos a tener accidentes?



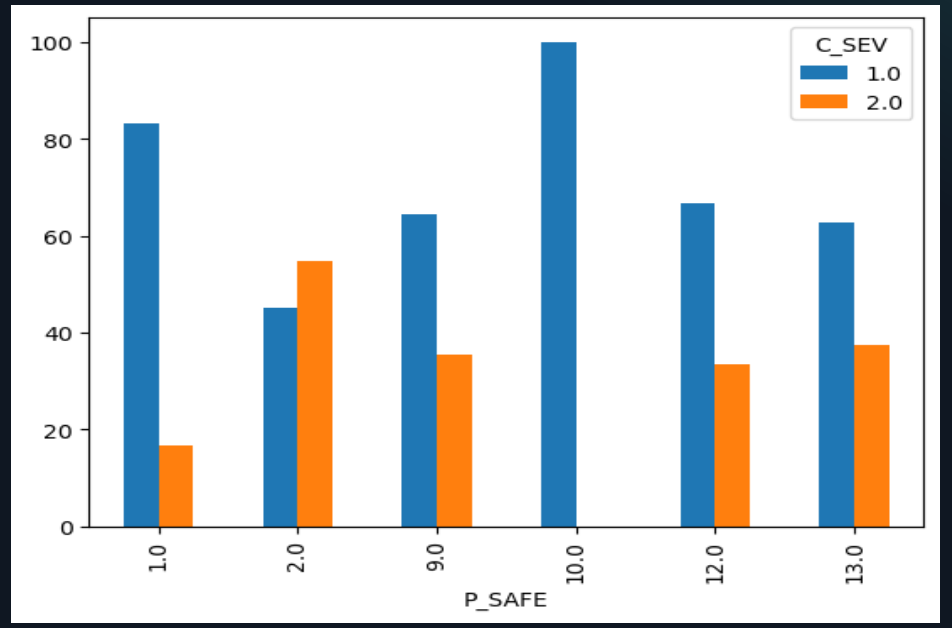
En cuanto al tipo de vehículo, destaca que los buses escolares no tienen casi accidentes con víctimas mortales, algo lógico al tratarse de un vehículo que circula a pocas velocidades y casi siempre por ciudades o pueblos. Otro vehículo con menor cantidad de accidentes mortales son las bicicletas. Por otro lado, el vehículo con más accidentes mortales es el tractor de carretera, seguido de cerca por otros como motos, caravanas, coches convencionales o motos de nieve.

ALGUNOS GRÁFICOS

ACCIDENTES POR TIPO DE CONDUCTOR



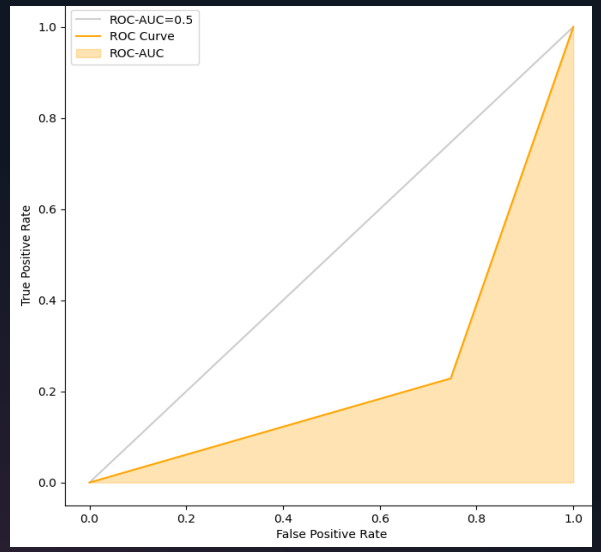
POR MEDIDAS DE SEGURIDAD EMPLEADAS



2- MODELOS DE CLASIFICACIÓN

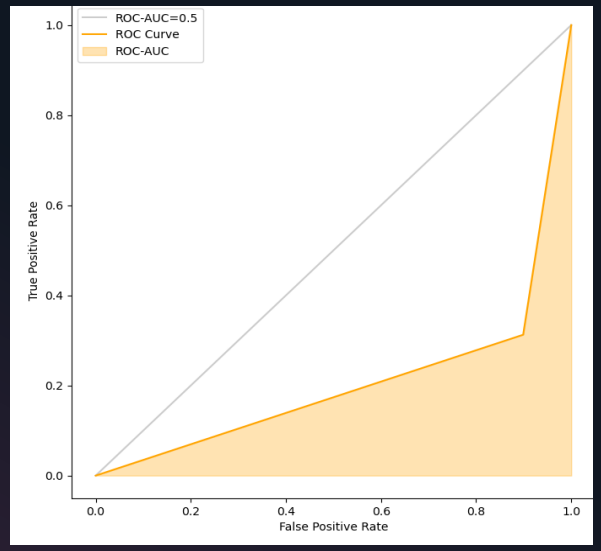
KNN

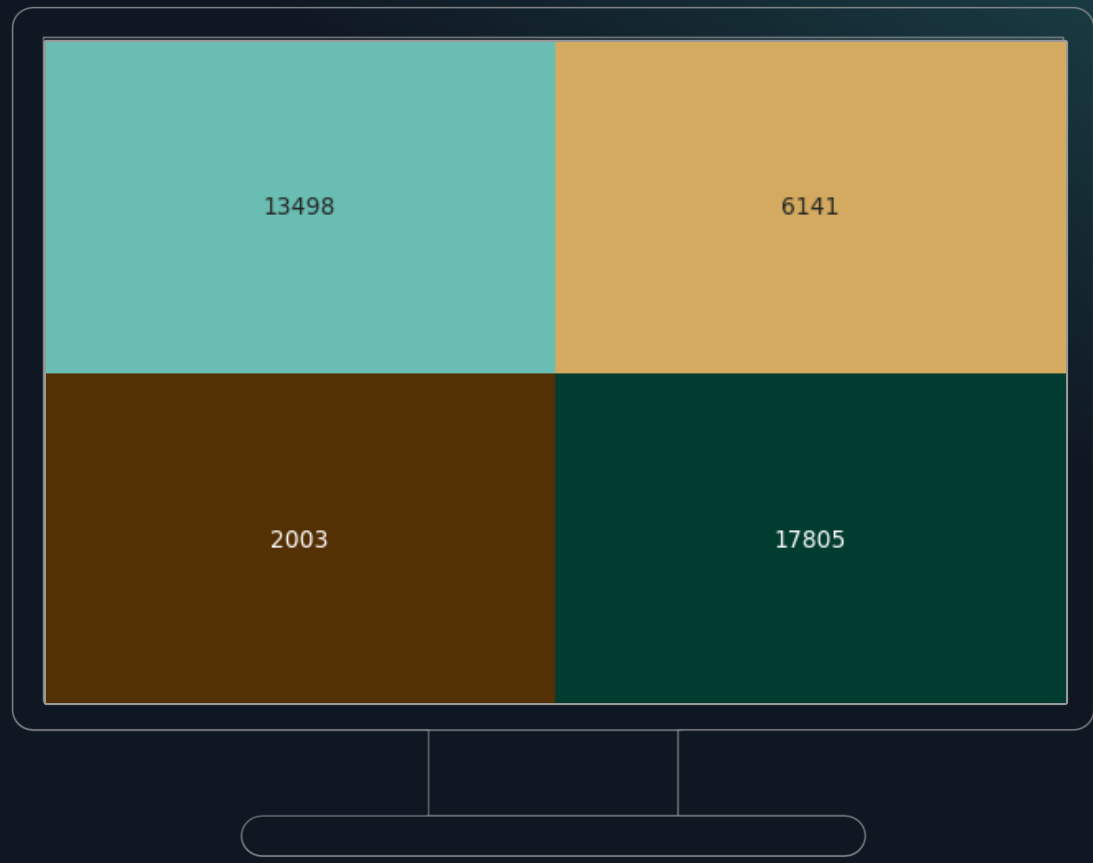
MÉTRICAS	VALORES
ACCURACY	0.758993
PRECISION	0.751190
RECALL	0.771424
F1-SCORE	0.761173
AUC	0.759046



ÁRBOL DE DECISIÓN

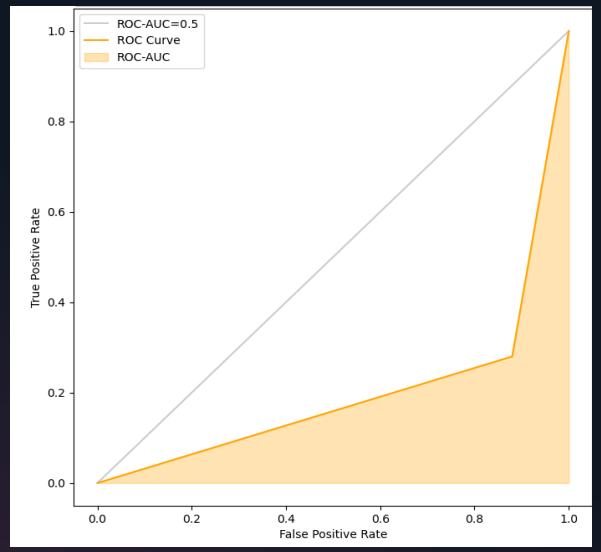
MÉTRICAS	VALORES
ACCURACY	0.793546
PRECISION	0.870783
RECALL	0.687306
F1-SCORE	0.768241
AUC	0.793092





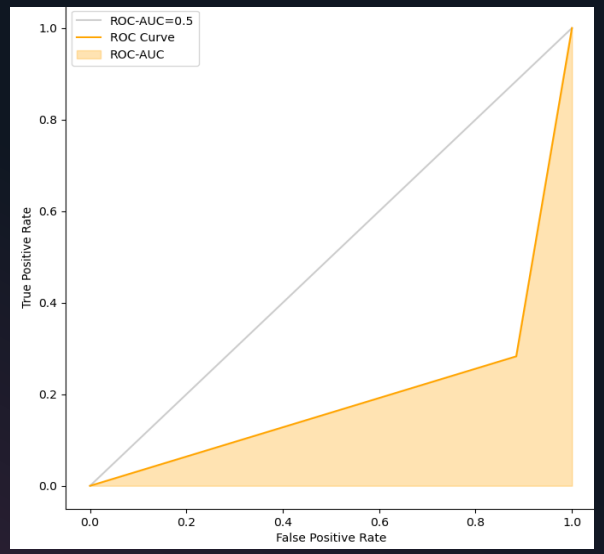
BAGGING

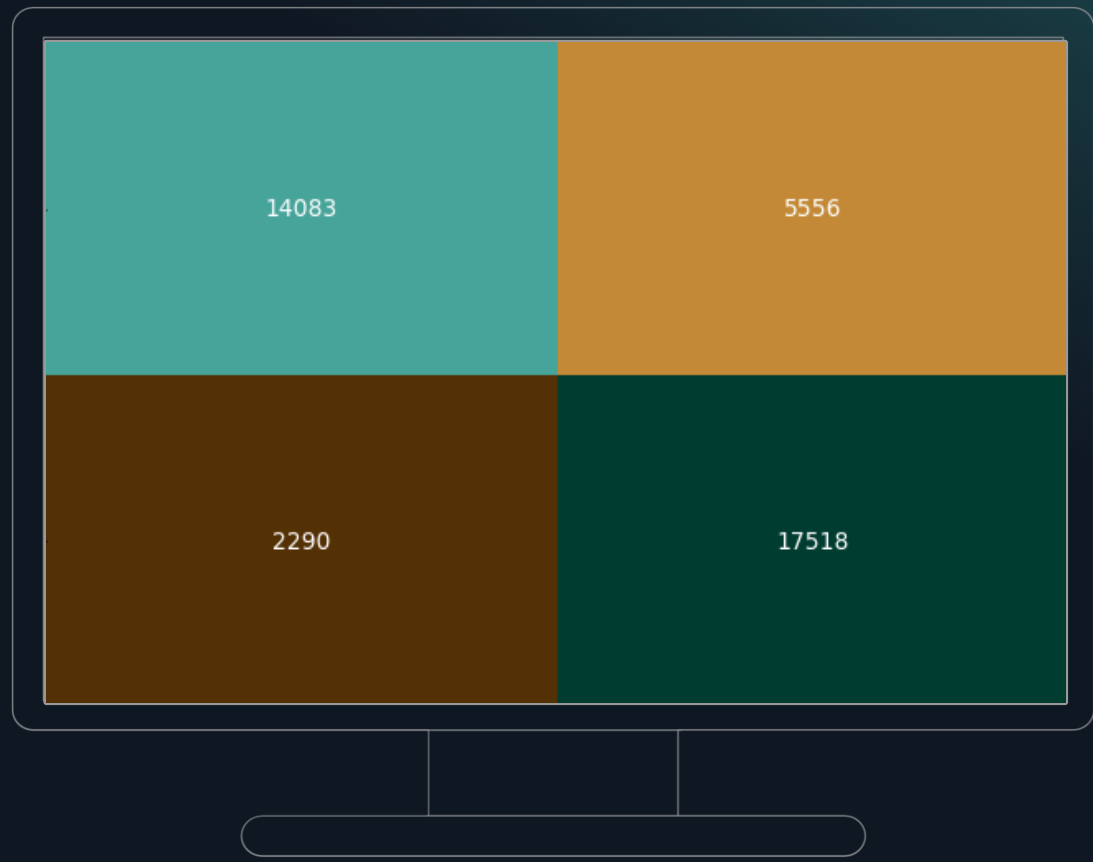
MÉTRICAS	VALORES
ACCURACY	0.800314
PRECISION	0.856079
RECALL	0.719945
F1-SCORE	0.782132
AUC	0.799971



RANDOM FOREST

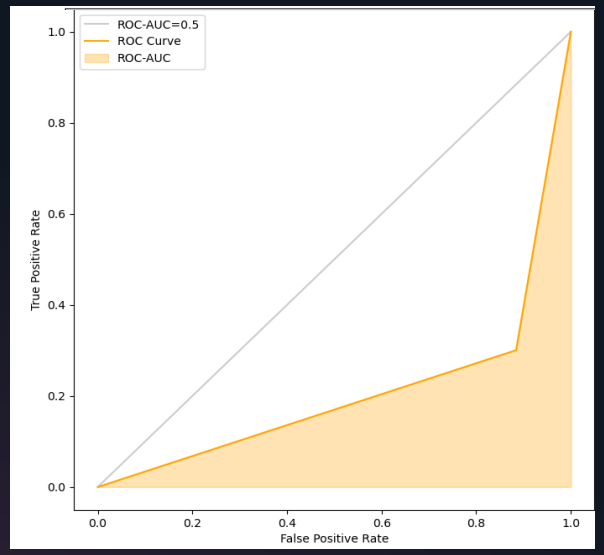
MÉTRICAS	VALORES
ACCURACY	0.801100
PRECISION	0.860136
RECALL	0.717094
F1-SCORE	0.782128
AUC	0.800741





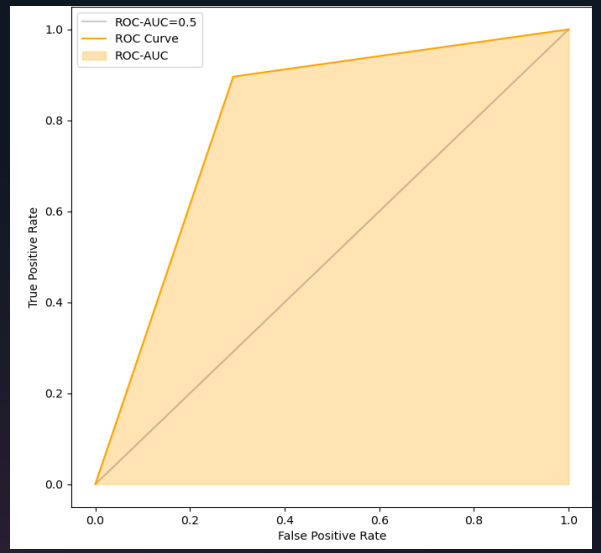
GRADIENT BOOST

MÉTRICAS	VALORES
ACCURACY	0.792228
PRECISION	0.856947
RECALL	0.699425
F1-SCORE	0.770214
AUC	0.791831



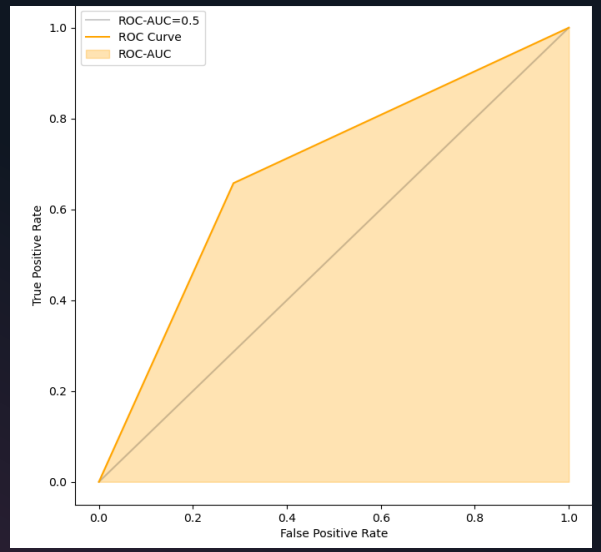
XGBOOST

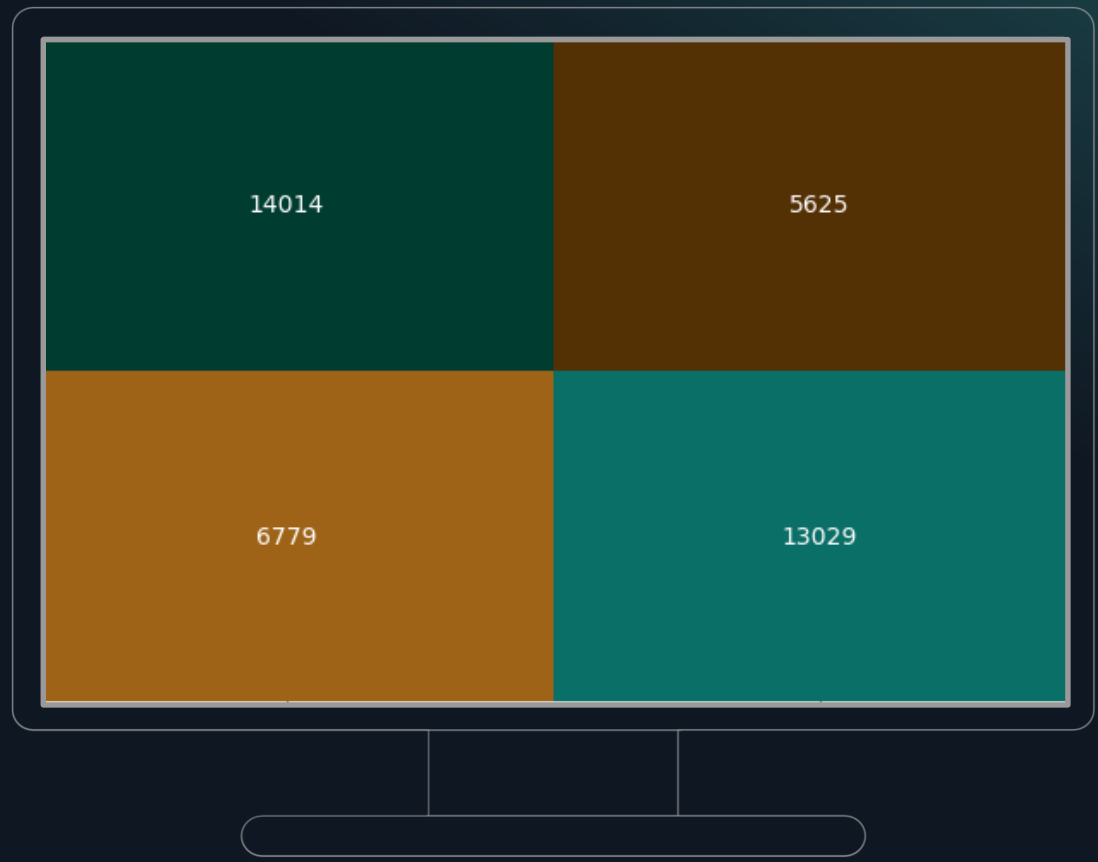
MÉTRICAS	VALORES
ACCURACY	0.802723
PRECISION	0.756178
RECALL	0.896052
F1-SCORE	0.820194
AUC	0.802321



REGRESIÓN LOGÍSTICA

MÉTRICAS	VALORES
ACCURACY	0.685553
PRECISION	0.698456
RECALL	0.657765
F1-SCORE	0.677500
AUC	0.685672





3- ELECCIÓN DEL MEJOR MÉTODO

MÉTRICAS A EMPLEAR

TP (True Positive): Son los valores que el algoritmo clasifica como positivos y que realmente son positivos.
TN (True Negative): Son valores que el algoritmo clasifica como negativos (0 en este caso) y que realmente son negativos.
FP (False Positive): Falsos positivos, es decir, valores que el algoritmo clasifica como positivo cuando realmente son negativos.
FN (False Negative): Falsos negativos, es decir, valores que el algoritmo clasifica como negativo cuando realmente son positivos.
Esta matriz es la base sobre la que se construyen todas las métricas de clasificación, que son las descritas a continuación:

• Accuracy: Representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos. Es recomendable utilizar esta métrica en problemas en los que los datos están balanceados, es decir, que haya misma cantidad de valores de cada clase, como es en nuestro caso. El problema con esta métrica es que nos puede llevar al engaño, es decir, puede hacer que un modelo malo parezca que es mucho mejor de lo que es. Su fórmula es Accuracy = (TP + TN) / (TP + TN + FP + FN)

•Matriz de confusión: Es una matriz en la que cada columna representa el número de predicciones de cada clase, mientras que cada fila representa las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases. Por tanto, los elementos que se encuentran en la diagonal serán aquellos en los que la clasificación es correcta, mientras que el resto de valores representan los errores. Por ejemplo, en la posición 1,2 tendremos que hemos predicho 2 cuando el valor real es 1 y en la posición 1,1 los valores 1 que hemos predicho como 1, es decir, un acierto. Por tanto, cuanto más valores haya en la diagonal, mejor estaremos clasificando. Mediante el accuracy podemos ver cuál es su grado de acierto. Nota: al hacer el gráfico aparece 0,1 en vez de 1 y 2, respectivamente.

• Precision: Calcula el porcentaje de valores que se han clasificado como positivos siendo realmente positivos. Su fórmula es Precision = TP / (TP + FP).

MÉTRICAS A EMPLEAR

• Recall: También conocida como el ratio de verdaderos positivos, es utilizada para saber cuantos valores positivos son correctamente clasificados. Su fórmula es Recall = TP / (TP + FN).

• F-Score: El valor F1 se utiliza para combinar las medidas de precision 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. Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado, por lo que en nuestro caso concreto no nos será una métrica muy representativa.
Se calcula mediante F1 = 2 * ((recall * precision)/(recall + precision))

• Curva ROC: Una curva ROC es un gráfico muy utilizado que representa el porcentaje de verdaderos positivos (True Positive Rate), también conocido como Recall, contra el ratio de falsos positivos (False Positive Rate). La diferencia con el resto de métricas, es que en este caso, el umbral por el que se clasifica un elemento como 0 o 1, se va modificando, para poder ir generando todos los puntos de la gráfica. Su valor numérico viene dado por la siguiente métrica.

• Área bajo la curva (AUC): A partir de la gráfica anterior, se puede obtener una métrica sólida, muy útil para problemas de clasificación binaria. El valor de esta métrica se encuentra en un rango entre 0 y 1, donde 0 es como si tuviéramos un modelo aleatorio y 1 es un resultado óptimo que indica que nuestro modelo generaliza muy bien.

¿QUÉ MÉTRICA ESCOGEMOS?

Una vez explicadas todas las métricas, así como obtenidos todos sus valores para cada una de las métricas, tenemos que decidir cuál de ellas será la más conveniente para llevar a cabo la predicción. Como tenemos datos balanceados, descartamos directamente el F-Score al ser más empleada con datos desbalanceados. Por otro lado, las métricas Precision y Recall se centran en ver la precisión de los positivos clasificados, olvidándose de los negativos. Como queremos tener una visión global de los aciertos, estas métricas no nos serán tan útiles, por lo que también la descartamos. Por último, en cuanto a la curva ROC y su valor asociado AUC, al salirnos tan parecidas en todos los modelos, no será muy preciso clasificar los modelos por mejor o peor con datos tan semejantes. Por tanto, emplearemos la métrica del Accuracy para comparar todos los modelos, pese a que hayamos comentado que a veces pueda ser engañosa y, aunque también salgan valores muy semejantes, es una métrica que refleja de forma general los aciertos del modelo. Aplicando esto al problema desde una perspectiva del mundo real, una aseguradora estaría interesada en predecir bien ambos tipos de siniestros para asignar de manera correcta cada seguro. Por ejemplo, si a alguien con pocas probabilidades de tener un accidente mortal le asignamos tal tipo de seguro, pagará menos y entonces estará contento por el servicio, quedándose así en la compañía. Por otro lado si alguien es más propenso a tener accidentes graves, a la aseguradora le interesa asignarle el seguro que cubra posibles fallecimientos, ya que será más caro para el cliente. Este es otro motivo por el cual queremos predecir bien tanto los 1 como los 2, decantándonos por la elección previamente comentada.

ELECCIÓN DEL MÉTODO SEGÚN LA MÉTRICA I

1

XGBOOST

Con accuracy = 0.802723

2

RANDOM FOREST

Con accuracy = 0.801100

3

BAGGING

Con accuracy = 0.800314

ELECCIÓN DEL MÉTODO SEGÚN LA MÉTRICA II

4

ÁRBOL

Con accuracy = 0.793546

5

GRADIENT BOOST

Con accuracy = 0.792228

6

KNN

Con accuracy = 0.758993

7

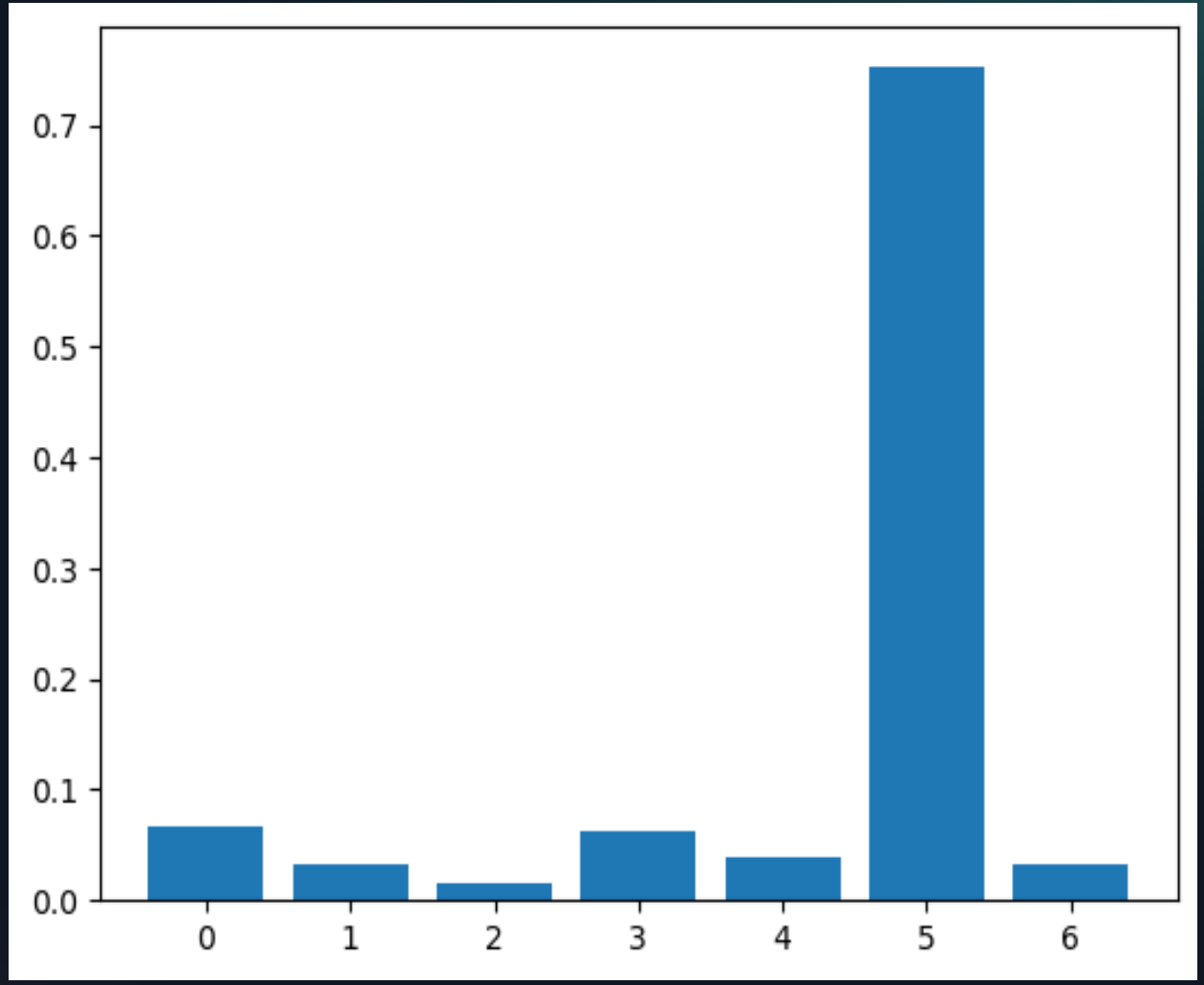
REGRESIÓN LOGÍSTICA

Con accuracy = 0.685553

4- IMPORTANCIA DE LAS VARIABLES

¿VARIABLE MÁS IMPORTANTE?

A partir del XGBOOST vemos la importancia de las variables, obteniendo que la más importante es P_ISEV con un casi 75% de la importancia total



5- CONCLUSIONES

CONCLUSIONES

Para concluir esta práctica, vamos a elaborar unas conclusiones y un repaso a todos los procesos realizados, sobre todo desde el punto de vista de una aseguradora que se plantee este problema en la realidad. Lo primero de todo es obtener todos los datos en un intervalo de años. Como es complicado tener todos los detalles de cada momento en períodos tan largos, es normal encontrarse con valores perdidos. Si bien es cierto que en la práctica imputamos estos valores por la moda, en la realidad no sería tan fácil llegar a esta conclusión, si bien es cierto que es mejor opción que directamente eliminar las observaciones del conjunto de datos, ya que se podría perder mucha información valiosa de cara a la futura predicción. A continuación, nos damos cuenta de que los datos están desbalanceados, por lo que hay que llevar a cabo un balanceo. Esto para la aseguradora es vital, ya que si se trabajara con los datos originales, a la hora de predecir, siempre se obtendrían como predicciones aquel dato que aparezca en abundancia y, entonces, tendríamos muchos fallos de clasificación que resultarían nefastos económicamente hablando para tal aseguradora. Por último, desarrollamos varios modelos de predicción con sus respectivas métricas. Siguiendo los criterios previamente explicados, elegimos el modelo que más nos convenga, que sería el que la aseguradora emplearía para potenciar los aciertos y, así, ganar más clientes y, con ellos, más dinero.