MACHINE LEARNING

PRÁCTICA FINAL

https://github.com/Lyapunov230/Practica_Final_ML.git

ENRIQUE ÁLVAREZ

JAVIER ESTEBAN

LUIS GANCEDO

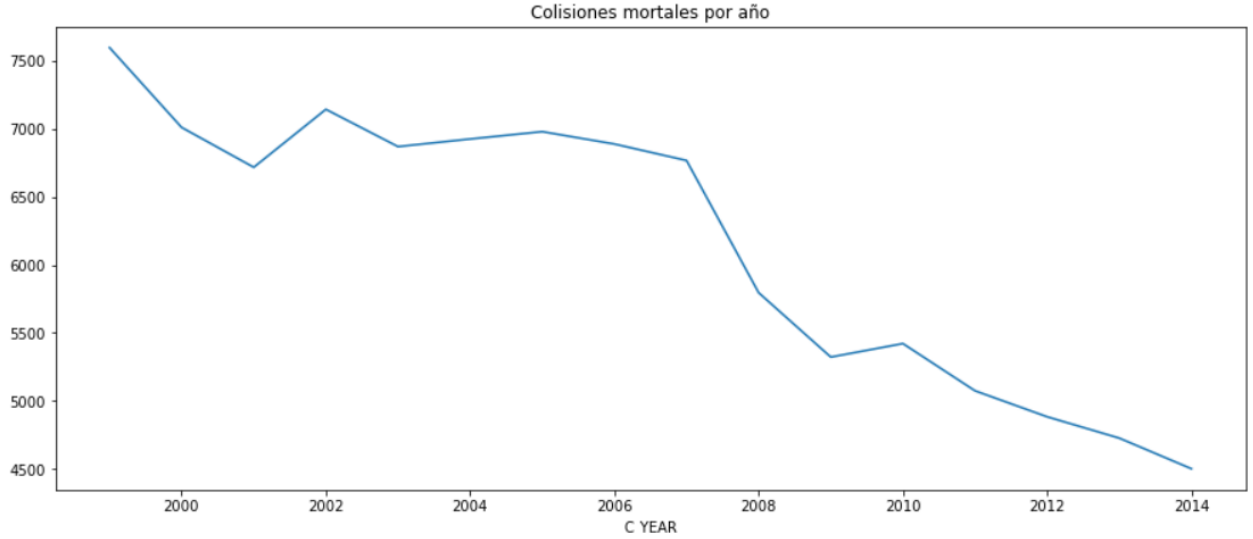
MARÍA LOSA

CANDELA MANJÓN

JOSÉ PARTAL

INTRODUCCIÓN

- Vamos a abordar un problema de Machine Learning realista.
- Analizaremos los accidentes ocurridos en Canadá entre 1999-2014, limitando nuestro análisis al 2014.
- Base de datos:
  - https://www.kaggle.com/datasets/tbsteal/canadian-car-accidents-19942014?select=drivingLegend.pdf
- El objetivo final del trabajo será evaluar la gravedad de los accidentes en Canadá en 2014.



ANÁLISIS DESCRIPTIVO DE LOS DATOS

El primer paso en un trabajo de estas características es realizar un análisis previo de los datos.

Tenemos una base de datos con 5860405 observaciones y 22 variables, que pueden dividirse en 3 grupos:

- Situación en la que ocurre el accidente.
- Vehículo implicado en el accidente.
- Persona invulcrada en el accidente

Atributo	descripción
C_YEAR	Año
C_MNTH	Mes
C_WDAY	Día de la semana
C_HOUR	Hora de la colisión
C_SEV	Gravedad de la colisión
C_VEHS	Número de vehículos involucrados en la colisión
C_CONF	Configuración de la colisión
C_RCFG	Configuración de la carretera (tipo de vía)
C_WTHR	Condiciones meteorológicas
C_RSUR	Estado de la carretera
C_RALN	Alineación de la carretera (línea recta, curva, con pendiente...)
C_TRAF	Control de tráfico (señalización)
V_ID	Número de identificación del vehículo
V_TYPE	Tipo de vehículo
V_YEAR	Año de construcción del vehículo
P_ID	Número de ifentificación de la persona
P_SEX	Sexo de la persona
P_AGE	Edad de la persona
P_PSN	Posición que ocupaba la persona en el vehículo
P_ISEV	Gravedad de la lesión
P_SAFE	Elementos de seguridad
P_USER	Tipo de usuario (conductor, peaton, motorista, copiloto,...)

VALORES PERDIDOS Y TRATAMIENTO

- Estableceremos como valores nulos los siguientes:
  - U/UU: Desconocido.
  - X/XX: La jurisdicción no provee este dato.
  - Q/QQ: Elección distinta a los valores precedentes.
  - N/NN: Dato no aplicable.
- Consideramos estas tres categorías como valores perdidos ya que son equivalentes en el sentido de que no tenemos el dato para la observación correspondiente y por lo tanto imputaremos un valor más adelante.

```
for i in datos.columns:

    datos[i]=np.where((datos[i]=='U') | (datos[i]=='UU') | (datos[i]=='UUUU'), np.nan, datos[i])
    datos[i]=np.where((datos[i]=='X') | (datos[i]=='XX') | (datos[i]=='XXXX'), np.nan, datos[i])
    datos[i]=np.where((datos[i]=='Q') | (datos[i]=='QQ') | (datos[i]=='QQQQ'), np.nan, datos[i])
    datos[i]=np.where((datos[i]=='N') | (datos[i]=='NN') | (datos[i]=='NNNN'), np.nan, datos[i])
```

VALORES PERDIDOS Y TRATAMIENTO

Hacemos un recuento de los datos faltantes de cada variable.

La variable *P_SAFE* tiene un 21% de observaciones faltantes, por lo que decidimos eliminarla.

Como solo consideramos el año 2014, eliminamos la variable *C_YEAR*.

```
datos = datos.drop(['C_YEAR','P_SAFE'],axis=1)
```

	nulos_columnas	porcentaje_columnas
P_SAFE	63207	0.212381
V_YEAR	29777	0.100053
C_RCFG	28139	0.094549
C_CONF	25691	0.086324
P_AGE	22578	0.075864
P_ISEV	20504	0.068895
C_TRAF	19316	0.064903
P_SEX	18921	0.063576
C_RALN	17958	0.060340
V_TYPE	15936	0.053546
P_USER	14267	0.047938
C_RSUR	12081	0.040593
P_PSN	10955	0.036810
C_WTHR	4307	0.014472
C_HOUR	2042	0.006861
P_ID	631	0.002120
C_VEHS	26	0.000087
C_MNTH	24	0.000081
C_WDAY	24	0.000081
V_ID	5	0.000017
C_SEV	0	0.000000
C_YEAR	0	0.000000

VALORES PERDIDOS Y TRATAMIENTO

- Imputamos los datos faltantes por la moda, al ser nuestras variables discretas.

```
pd_series_null_columns = datos.isnull().sum().sort_values(ascending=False)
pd_series_null_rows = datos.isnull().sum(axis=1).sort_values(ascending=False)


pd_null_columnas = pd.DataFrame(pd_series_null_columns, columns=['nulos_columnas'])
pd_null_filas = pd.DataFrame(pd_series_null_rows, columns=['nulos_filas'])
pd_null_filas['target'] = datos['V_TYPE'].copy()
pd_null_columnas['porcentaje_columnas'] = pd_null_columnas['nulos_columnas']/datos.shape[0]
pd_null_filas['porcentaje_filas']= pd_null_filas['nulos_filas']/datos.shape[1]

print('Número de datos faltantes (missings):\n')

pd_null_columnas
```

TIPO DE VARIABLES Y CODIFICACIÓN DE LA VARIABLE CATEGÓRICA

- Tenemos una variable categórica, el resto son numéricas (discretas).
- Codificaremos la variable categórica con *Ordinal Encoder*, asignando el valor 1 si es hombre y el 0 si es mujer.

```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
enc.fit(datos[list_var_cat])
datos[list_var_cat] = enc.transform(datos[list_var_cat])
datos[list_var_cat]=datos[list_var_cat].astype(int)
```

```
list_var_cat, other = funciones_auxiliares.dame_variables_categoricas(dataset=datos)
datos[list_var_cat] = datos[list_var_cat].astype("category")
list_var_continuous = list(datos.select_dtypes('float').columns)
datos[list_var_continuous] = datos[list_var_continuous].astype(float)
```

```
for i in datos.columns:
    datos[i]=datos[i].astype(int)
```

```
1  datos.dtypes
```

```
C_MNTH     float64
C_WDAY     float64
C_HOUR     float64
C_SEV      float64
C_VEHS     float64
C_CONF     float64
C_RCFG     float64
C_WTHR     float64
C_RSUR     float64
C_RALN     float64
C_TRAF     float64
V_ID       float64
V_TYPE     float64
V_YEAR     float64
P_ID       float64
P_SEX       object
P_AGE      float64
P_PSN      float64
P_ISEV     float64
P_USER     float64
dtype: object
```

ANÁLISIS GRÁFICO DE VARIABLES

Antes de proceder con el empleo de técnicas de clasificación, vamos a hacer un análisis gráfico de las variables para entenderlas mejor.

Los sábados son los días más mortales de la semana, siendo la franja horaria entre las 14:00 y las 17:00 horas la que más accidentes causa y la más mortal.

Durante la noche hay menos colisiones que en el día, pero su mortalidad es mayor.

```
1 by_hour_dead =gravedad.groupby('C_HOUR')['C_SEV'].count()
2 plot1 =by_hour_dead.plot(kind='bar', title = 'Colisiones mortales por hora')
```

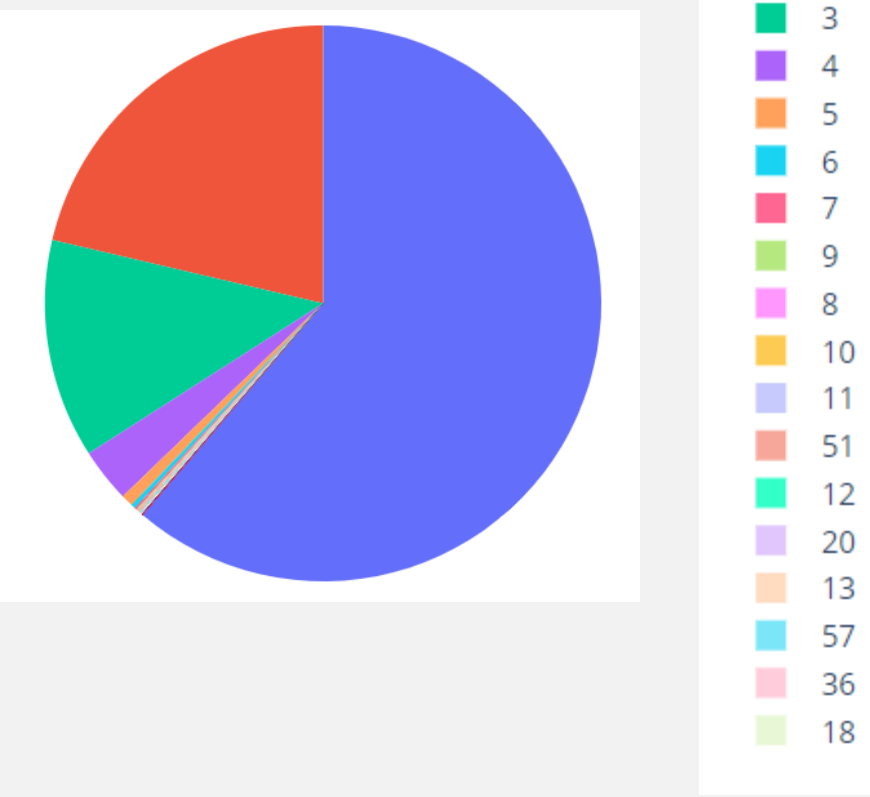

```
1 by_hour = datos.groupby('C_HOUR')['C_SEV'].count()
2 plot2 = by_hour.plot(kind='bar',title='Colisiones por hora');
```

ANÁLISIS GRÁFICO DE VARIABLES

El 87% de los accidentes son causados por un vehículo ligero (codificado con el 1). A los coches ligeros le siguen las motocicletas (codificado con el valor 14) y las bicicletas (codificado con el valor 17) con porcentajes superiores al 2%.

En el 61% de los accidentes hay 2 vehículos implicados, 1 en el 21% y 3 en el 12%.

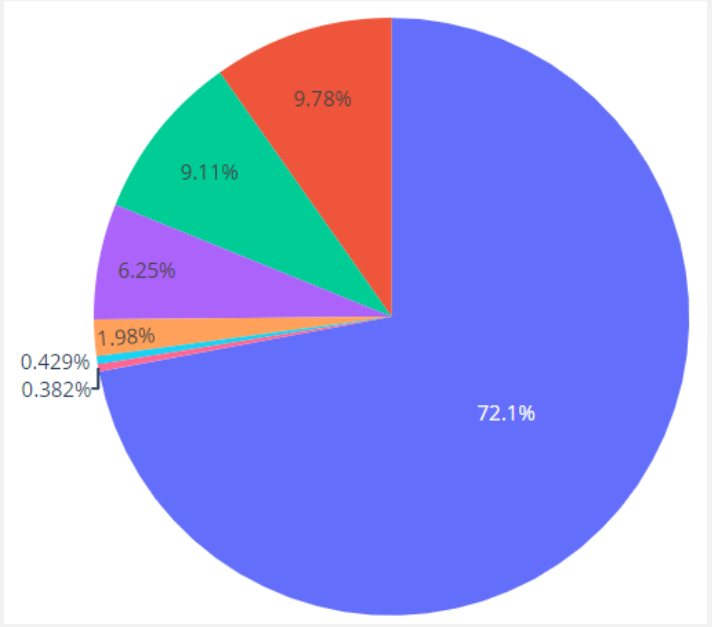
	index	percent	V_TYPE
0	1	87.617771	260761
1	14	2.314087	6887
2	17	2.089969	6220
3	6	1.813435	5397
4	7	1.592342	4739
5	8	1.358480	4043
6	5	1.133019	3372
7	11	1.022808	3044
8	9	0.481835	1434
9	16	0.212693	633
10	20	0.156580	466
11	22	0.066866	199
12	19	0.046033	137
13	23	0.037633	112
14	18	0.025537	76
15	10	0.018816	56
16	21	0.012096	36

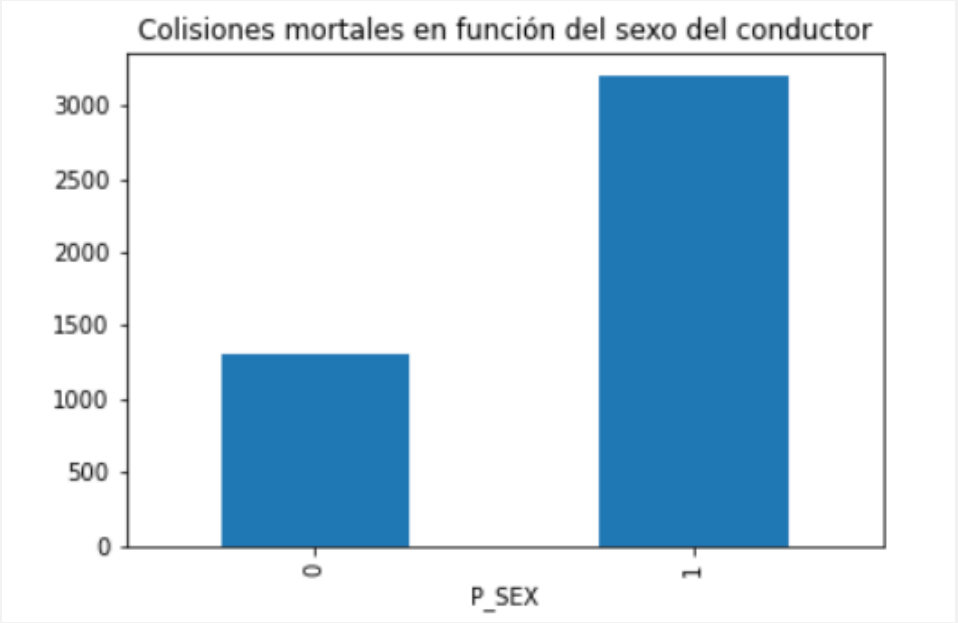


ANÁLISIS GRÁFICO DE VARIABLES

El 72% de los accidentes se producen en días soleados. El 9% de los accidentes se producen con precipitaciones y otro 9% en días nublados.

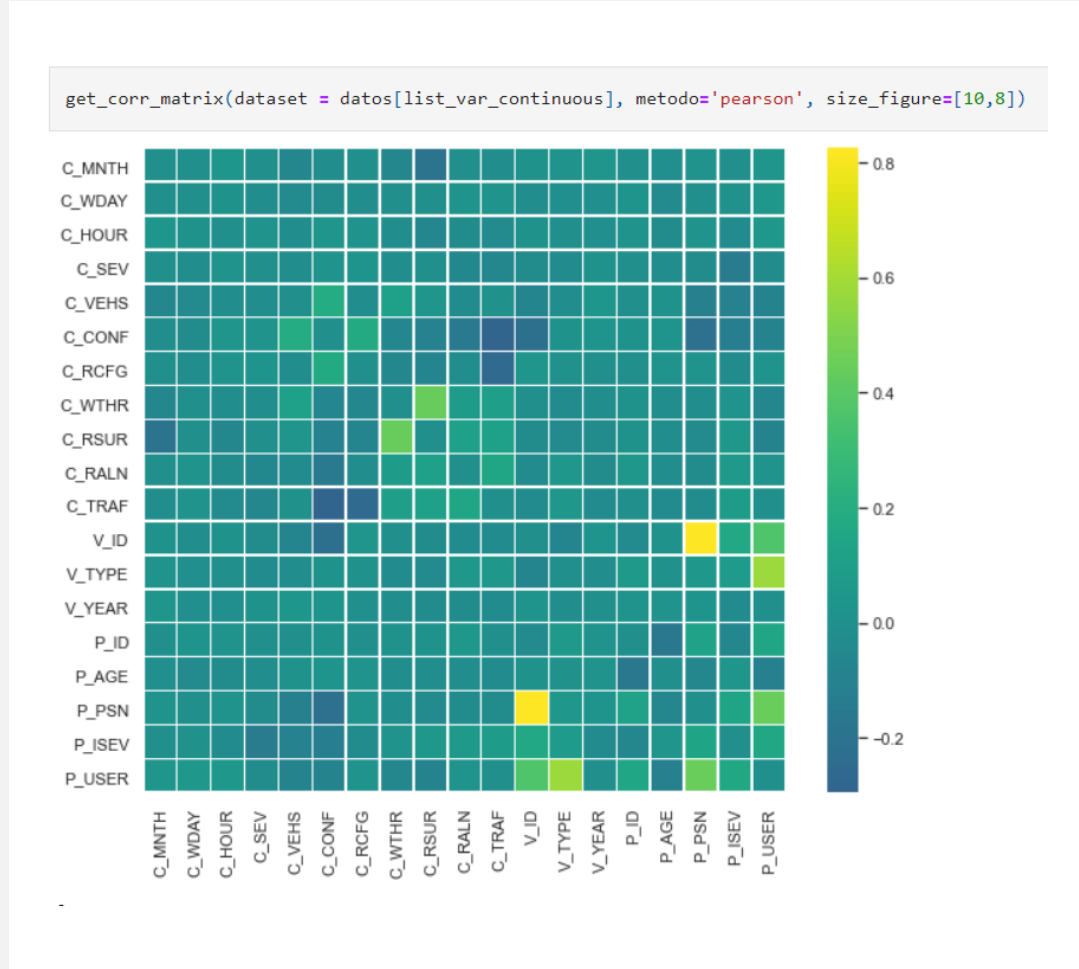
Por su parte, el 98,5% de los accidentes son mortales y los hombres producen la mayoría de los accidentes, aunque en nuestra base hay más observaciones de varones que de mujeres.





ELIMINACIÓN DE VARIABLES

- Primero analizamos la correlación de las variables eliminando aquellas que presenten alta correlación con otra.
- Eliminamos *V_ID* y *P_USER*.
- Eliminamos también *C_RALN*, *C_TRAF* y *P_ID* por no considerarlas relevantes en el análisis.
- Nos queda una base con 297612 observaciones y 15 variables.



APRENDIZAJE NO SUPERVISADO

- En ocasiones no disponemos de ese etiquetado de los datos en un problema real, por lo que se requerirán modelos de aprendizaje automático que puedan clasificar correctamente estos datos, encontrando por sí mismos algunos puntos en común en las características, que se utilizarán para predecir las clases sobre nuevos datos. De esto se encarga el aprendizaje no supervisado.
- Emplearemos dos técnicas muy comunes dentro del aprendizaje no supervisado: reducción dimensional y clustering.

APRENDIZAJE NO SUPERVISADO: PCA

- Técnica de reducción dimensional.
- Dado un número amplio de variables posiblemente correlacionadas, el algoritmo las reduce a un número menor de variables transformadas (componentes principales) que expliquen gran parte de la variabilidad en los datos.
- Nuestras variables tienen unidades de medida y rangos de valores muy distintos, vamos a estandarizarlas con la función *StandardScaler*.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
datos_scaled = scaler.fit_transform(datos2)
datos_scaled
```

APRENDIZAJE NO SUPERVISADO: PCA

- Agrupamos los datos en 9 componentes principales, que explican el 60% de la varianza, el mínimo exigido en un análisis multivariante.

```
pca = PCA(n_components=9)
principalComponents = pca.fit_transform(datos_scaled)
```

```
principal = pd.DataFrame(data=principalComponents, columns=["PC1", "PC2","PC3","PC4","PC5","PC6","PC7","PC8","PC9"])
principal
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
0	1.040440	-0.452714	0.007692	1.481937	0.465326	0.571846	-2.804896	0.052967	-0.584102
1	2.436329	-0.566563	-1.139096	0.171863	0.763552	-0.108534	-1.525604	0.130826	1.167338
2	4.690553	0.085072	-0.935113	0.972698	-0.103153	-0.255042	-2.199406	-0.557673	0.507417
3	-0.240373	0.073403	-1.212088	1.256368	0.610564	0.021418	-0.963836	0.856172	0.672207
4	-0.642947	1.008092	0.211843	0.233376	-0.022968	0.070967	-1.590844	0.729938	0.002554
...	...	...	...	...	...	...	...	...	...
297607	-0.307210	0.324116	1.454461	-1.207345	-0.007649	0.172160	0.057494	-0.733356	0.191183
297608	-1.026438	-3.217437	1.288922	0.188318	4.546586	-3.123931	-1.186132	-1.964278	-1.054794
297609	2.793846	-0.991589	-1.006671	-0.743370	0.474940	0.892295	-0.010569	-0.338697	0.459239
297610	2.811284	-3.598273	5.335754	5.390513	0.865006	-0.676585	3.844598	0.542056	-1.019510
297611	2.425763	-4.376969	5.132251	1.943485	1.457423	-3.175125	3.648679	-1.263758	1.222153

APRENDIZAJE NO SUPERVISADO: K-MEDIAS

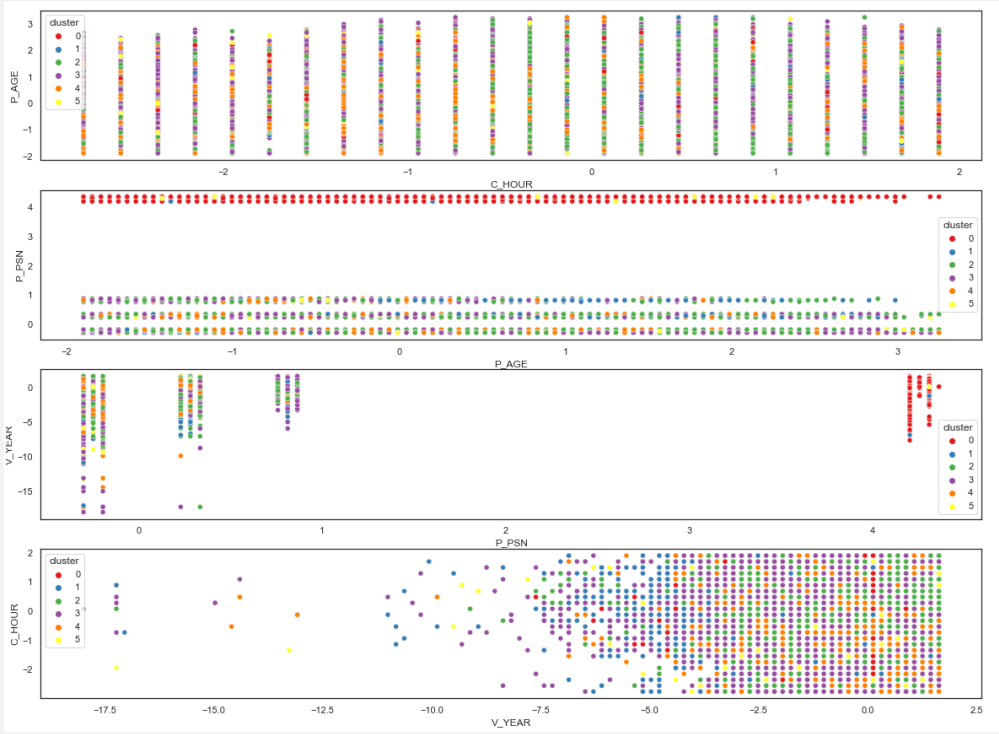
Técnica de clustering.

El algoritmo encuentra K centroides y asigna cada punto de datos a exactamente un grupo con el objetivo de minimizar la varianza dentro del grupo.

Requiere especificar el número de clusters de forma previa. Emplearemos el *método del codo* para su identificación.

Con datos estandarizados muchos clusters y difícil interpretación. Usaremos por ello datos no estandarizados.

```
wcss = []

for i in range(1, 16):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)

# Mostrando los resultados en un gráfico para determinar un codo 'The elbow'
plt.plot(range(1, 16), wcss, 'bx-')
plt.title('The elbow method')
plt.xlabel('Número de clusters')
plt.ylabel('WCSS') # Suma de cuadrados dentro del cluster (Within Cluster Sum Of Squares)
plt.show()
```

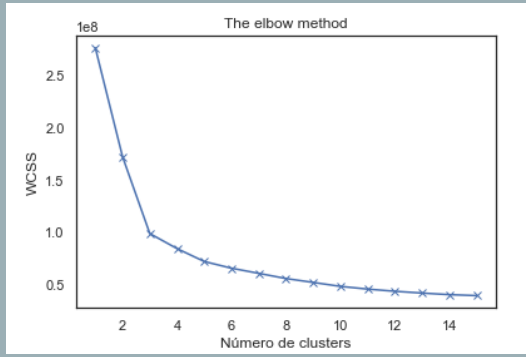


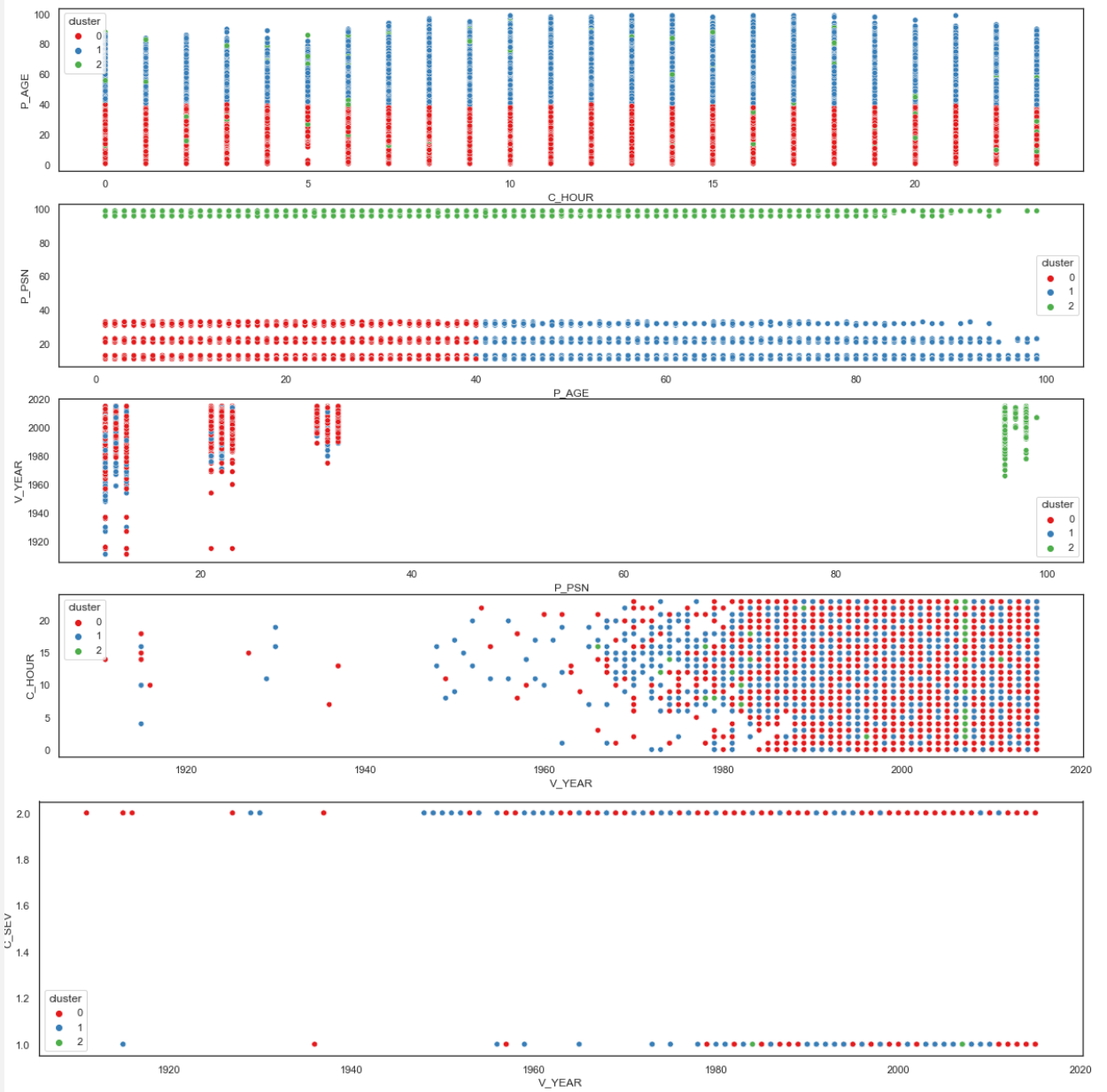
APRENDIZAJE NO SUPERVISADO: K-MEDIAS

Con datos no estandarizados identificamos 3 clusters. Aplicamos el algoritmo para K=3.

En el cluster 0 la edad de los accidentados es inferior en su mayoría a 40 años, en el 1 superior.

Hay más fallecidos en coches fabricados en los 90 y principios de los 2000.

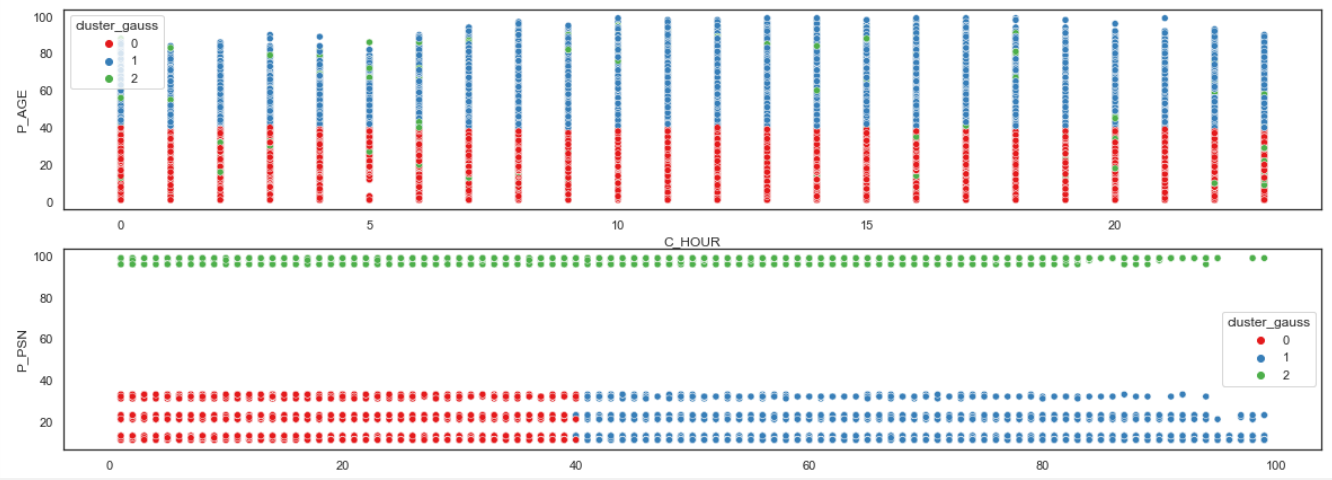




APRENDIZAJE NO SUPERVISADO: MIXTURAS GAUSSIANAS

- Modelo generativo que asume que los datos han sido generados por una mezcla de varias distribuciones normales multivariadas. El algoritmo tiene como objetivo estimar las matrices de media y covarianza de estas distribuciones.
- Generaliza el algoritmo K-Medias: agrega covarianza entre características para que los conglomerados puedan ser elipsoides en lugar de esferas, mientras que los centroides están representados por las medias de cada distribución.
- Arroja valores similares al algoritmo anterior.

```
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=3,random_state=0)
gaussian_pred = gm.fit_predict(datos2)
```



APRENDIZAJE SUPERVISADO

- En este apartado vamos a considerar que sí que tenemos los datos etiquetados y vamos a estar interesados en predecir si tras la colisión se necesitó tratamiento médico o no. Para ello vamos a trabajar con la variable objetivo *P_ISEV*. La variable seleccionada toma 3 valores posibles: ileso, herido o muerte en el acto; por lo que estudiaremos la fatalidad del accidente.
- Vamos a plantear el problema desde la perspectiva de una compañía de seguros que tiene un cliente potencial y tiene que decidir si le compensa o no asegurarlo en función de sus características. Más concretamente vamos a suponer que la aseguradora está interesada en conocer los posibles gastos hospitalarios derivados del accidente (pues corren a cargo de la empresa). Para ello, en primer lugar vamos a codificar la variable objetivo como binaria de manera que tomará el valor 0 si no hay heridos y 1 si hay heridos y/o fallecidos.

APRENDIZAJE SUPERVISADO

- Dividiremos la muestra en entrenamiento (80%) y test (20%) y emplearemos datos estandarizados.

```
from sklearn.model_selection import train_test_split

# Dividimos los datos en entrenamiento y test (80 training, 20 test)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, test_size = .20, random_state = 2, shuffle=True)

print('Datos entrenamiento: ', X_train.shape)
print('Datos test: ', X_test.shape)
```

```
Datos entrenamiento:  (238089, 14)
Datos test:  (59523, 14)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train_s = scaler.transform(X_train)
X_test_s  = scaler.transform(X_test)
```

```
X_train_s  = pd.DataFrame(X_train_s, columns = X.columns)
X_test_s  = pd.DataFrame(X_test_s, columns = X.columns)
```

APRENDIZAJE SUPERVISADO

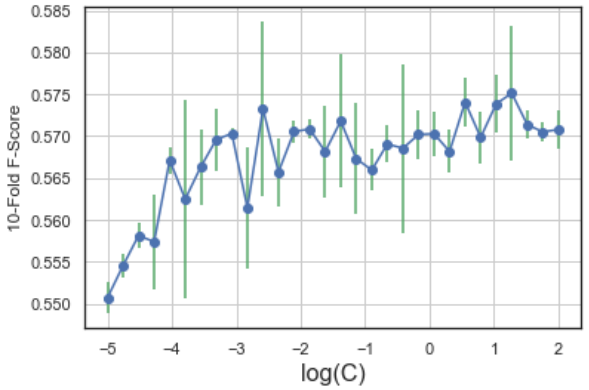
- Evaluaremos nuestros algoritmos de acuerdo a diferentes métricas y atendiendo a la matriz de confusión.
- Las métricas que emplearemos son:
  - ***Accuracy***: número total de predicciones correctas dividido por el número total de predicciones.
  - ***Precisión***: define cuan confiable es un modelo en responder si un punto pertenece a esa clase. Esto es, de los clasificados como positivos, cuántos lo son realmente.
  - ***Recall***: expresa cuan bien puede el modelo detectar a esa clase. Capacidad de la prueba de clasificar correctamente los casos que son 1, es decir, de los que son realmente positivos, cuántos han sido clasificados como tal.
  - ***Especificidad***: análogo al anterior pero para los casos negativos.
  - ***F1 Score***: combina las medidas de precisión y *recall* en un sólo valor. Se calcula haciendo la media armónica entre la precisión y el *recall*.
  - ***Curva ROC***: mide la sensibilidad frente a la especificación para cada umbral para un sistema de clasificación binario.

APRENDIZAJE SUPERVISADO: REGRESIÓN LOGÍSTICA

La regresión logística es un modelo probabilístico que vincula los datos binarios observados con un conjunto de características. Supone que hay un conjunto de pesos, coeficientes, o parámetros β, uno para cada característica, y le asigna unos pesos.

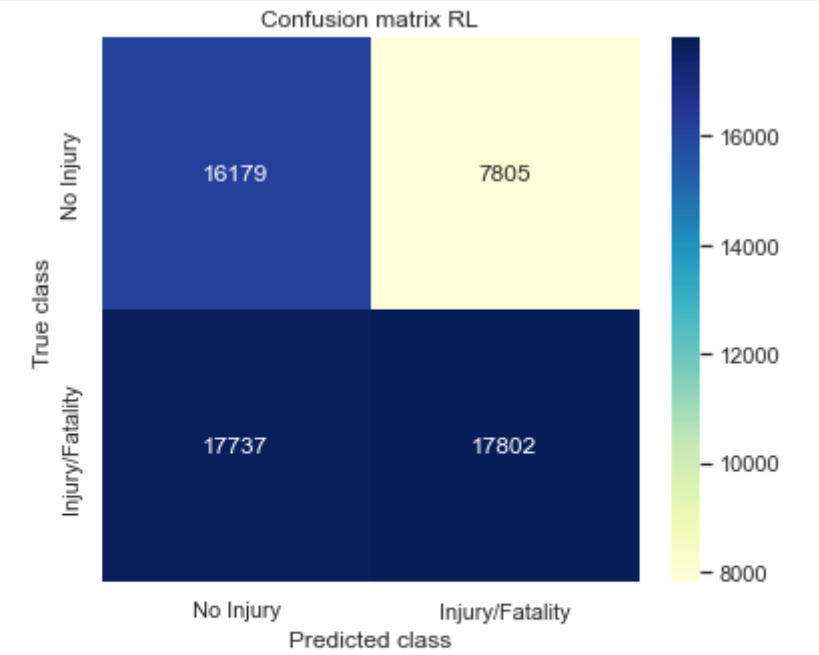
```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

vectorC = np.logspace(-5,2,30)

param_grid = {'C': vectorC }

grid = GridSearchCV(LogisticRegression(random_state = 0, class_weight='balanced'),
                    scoring = 'accuracy',
                    param_grid = param_grid,
                    cv = 3)

grid.fit(X_train, y_train) #Ajustamos el modelo al conjunto de entrenamiento

print("best mean cross-validation score: {:.3f}".format(grid.best_score_))
print("best parameters: {}".format(grid.best_params_))

scores = grid.cv_results_['mean_test_score']
std_scores = grid.cv_results_['std_test_score']
plt.errorbar(np.log10(vectorC),scores,yerr = std_scores, fmt='o-',ecolor='g')
plt.xlabel('log(C)',fontsize=16)
plt.ylabel('10-Fold F-Score')
plt.grid()
plt.show()
```

```
best mean cross-validation score: 0.575
best parameters: {'C': 18.873918221350994}
```

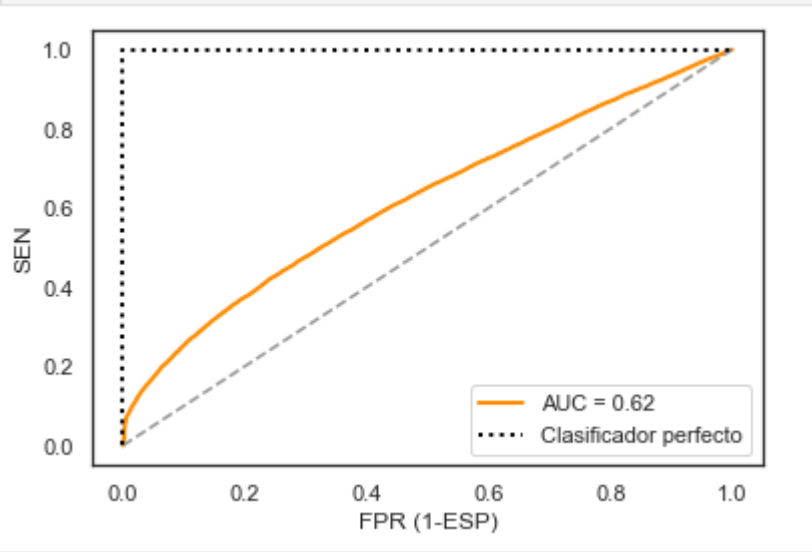



```python
Copt = grid.best_params_['C']

# Hacemos transform y fit a la vez
lr = LogisticRegression(random_state=0, C = Copt, class_weight ='balanced').fit(X_train, y_train)
y_pred_logr = lr.predict(X_test)
```

APRENDIZAJE SUPERVISADO: REGRESIÓN LOGÍSTICA




```
Accuracy: 0.5708885640844715
Precision_NoInjury:  0.4770314895624484
Precision_Injury/Fatality :  0.6952005311047761
Recall_NoInjury :  0.6745747164776518
Recall_Injury/Fatality:  0.5009144883086187
F1_Score_Injury/Fatality:  0.5822784810126583
```



APRENDIZAJE SUPERVISADO: ÁRBOL DE DECISIÓN

- Un árbol de decisión en Machine Learning es una estructura de árbol similar a un diagrama de flujo donde un nodo interno representa una característica (o atributo), la rama representa una regla de decisión y cada nodo hoja representa el resultado.
- El nodo superior en un árbol de decisión en Machine Learning se conoce como el nodo raíz. Aprende a particionar en función del valor del atributo. Divide el árbol de una manera recursiva llamada partición recursiva.
- El hiperparámetro más importante es la **profundidad máxima**: en principio, podríamos hacer que el árbol crezca indefinidamente hasta que todos los ejemplos de train estén perfectamente separados. Pero esto puede conducir a memorizar solo los datos de train (sobreajuste), creando un modelo que generaliza mal a test. Por ello, acotamos la profundidad máxima del árbol mediante *Grid Search*.

APRENDIZAJE SUPERVISADO: REGRESIÓN LOGÍSTICA

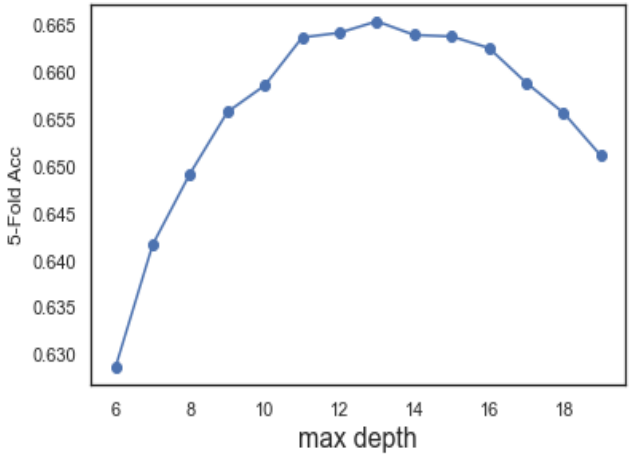
Un árbol de decisión en Machine Learning es una estructura de árbol similar a un diagrama de flujo donde un nodo interno representa una característica (o atributo), la rama representa una regla de decisión y cada nodo hoja representa el resultado.

El nodo superior en un árbol de decisión en Machine Learning se conoce como el nodo raíz. Aprende a particionar en función del valor del atributo. Divide el árbol de una manera recursiva llamada partición recursiva.

El hiperparámetro más importante es la profundidad máxima: en principio, podríamos hacer que el árbol crezca indefinidamente hasta que todos los ejemplos de train estén perfectamente separados. Pero esto puede conducir a memorizar solo los datos de train (sobreajuste), creando un modelo que generaliza mal a test. Por ello, acotamos la profundidad máxima del árbol mediante Grid Search.

```
from sklearn.tree import DecisionTreeClassifier

maxDepth = range(6,20)
param_grid = {'max_depth': maxDepth }

grid = GridSearchCV(DecisionTreeClassifier(), scoring= 'accuracy', param_grid=param_grid, cv = 3)
grid.fit(X_train_s, y_train)
print("best mean cross-validation score: {:.3f}".format(grid.best_score_))
print("best parameters: {}".format(grid.best_params_))

scores = np.array(grid.cv_results_['mean_test_score'])
plt.plot(maxDepth,scores,'-o')
plt.xlabel('max depth',fontsize=16)
plt.ylabel('5-Fold Acc')
plt.show()
```

```
best mean cross-validation score: 0.665
best parameters: {'max_depth': 13}
```




```
maxDepthOptimo = grid.best_params_['max_depth']
treeModel = DecisionTreeClassifier(max_depth = maxDepthOptimo).fit(X_train_s,y_train)
```

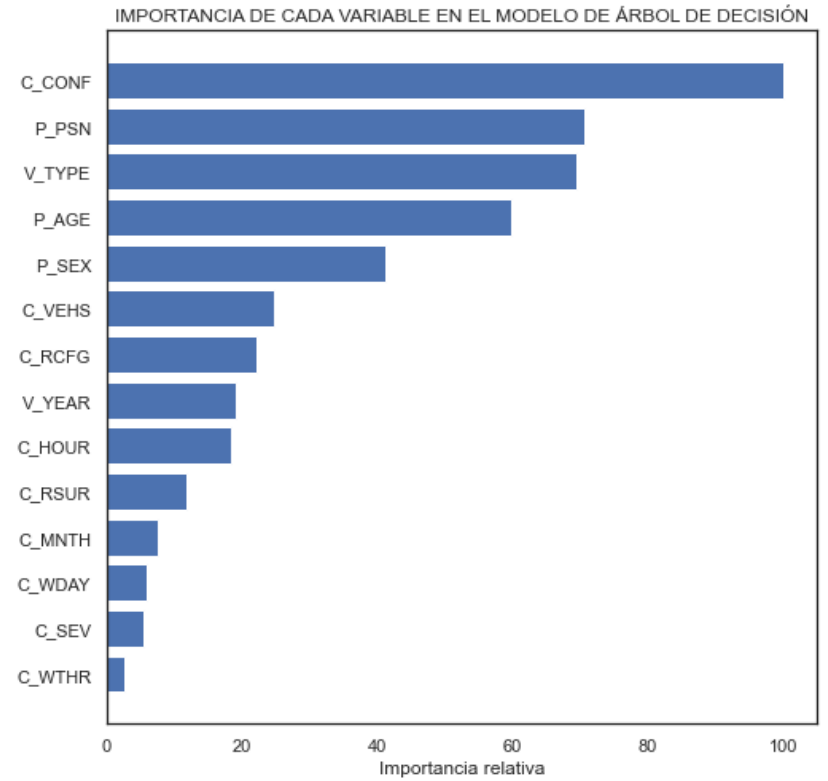
APRENDIZAJE SUPERVISADO: ÁRBOL DE DECISIÓN

La variable más importante a la hora de clasificar la gravedad del accidente mediante el modelo de árboles de decisión es la configuración del accidente. Tiene sentido que este criterio sea el más destacable puesto que, por ejemplo, una colisión entre dos coches en movimiento será más grave que una entre un coche parado y otro en movimiento.

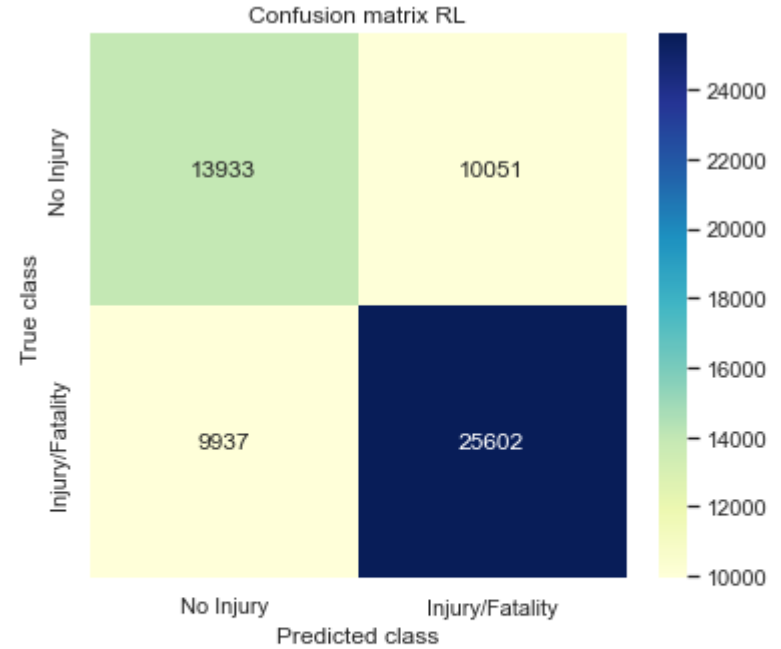
En segundo lugar se encuentra la posición que ocupa la persona dentro del vehículo accidentado. Normalmente los asientos delanteros suelen estar más expuestos y por tanto sería en estos donde las lesiones serían más graves.

Prácticamente con la misma importancia que el anterior tenemos el tipo de vehículo. Este elemento también es importante pues no es lo mismo que la colisión sea con una moto, un coche o una bicicleta, aunque ya vimos que en nuestra base el grupo más numeroso era el formado por los coches familiares.

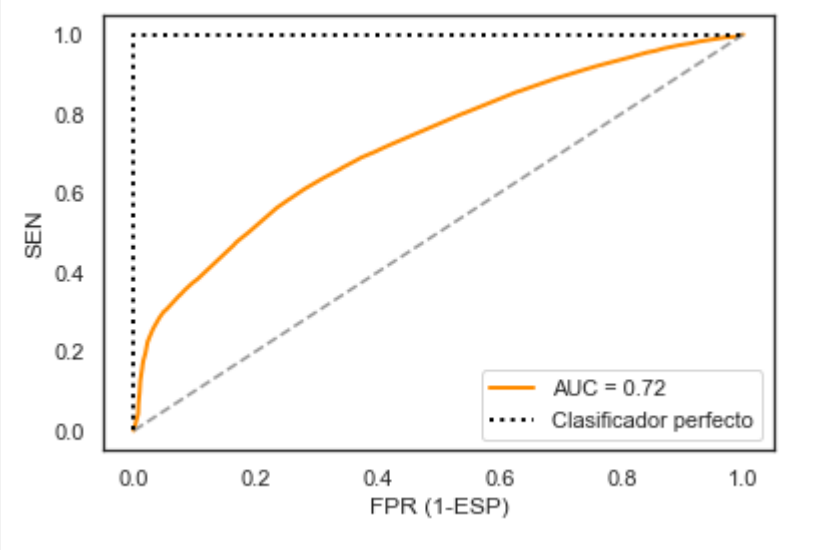
```
feature_importance = treeModel.feature_importances_

feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5

plt.figure(figsize=(8, 8))
plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, X_train_s.keys()[sorted_idx])
plt.xlabel('Importancia relativa')
plt.title('IMPORTANCIA DE CADA VARIABLE EN EL MODELO DE ÁRBOL DE DECISIÓN')
plt.show()
```



APRENDIZAJE SUPERVISADO: ÁRBOL DE DECISIÓN

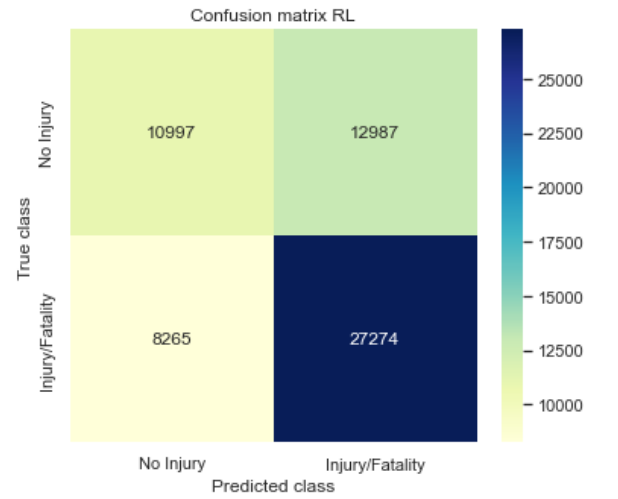



```
Accuracy: 0.6641970330796498
Precision_NoInjury:  0.5837033933808128
Precision_Injury/Fatality :  0.7180882394188427
Recall_NoInjury :  0.58092895263509
Recall_Injury/Fatality:  0.7203916823771068
F1_Score_Injury/Fatality:  0.7192381166423194
```

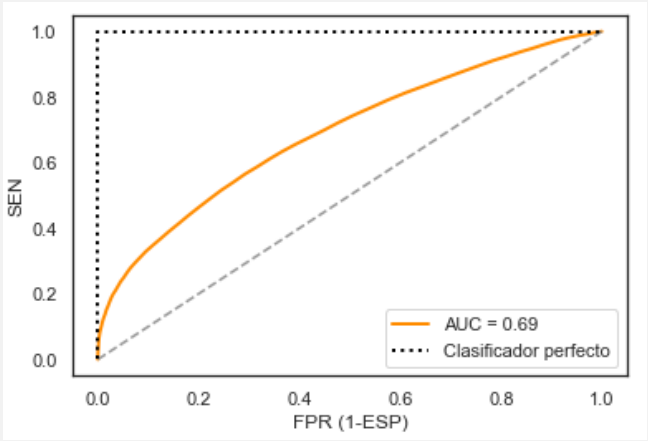


APRENDIZAJE SUPERVISADO: KNN

- Es un método no paramétrico que memoriza los datos de entrenamiento, encuentra los K datos más próximos y predice la clase mayoritaria entre las K clases anteriores.




```
Accuracy: 0.6429615442770021
Precision_NoInjury:  0.5709168310663483
Precision_Injury/Fatality :  0.6774297707458831
Recall_NoInjury :  0.4585140093395597
Recall_Injury/Fatality:  0.7674385885928136
F1_Score_Injury/Fatality:  0.7196306068601583
```

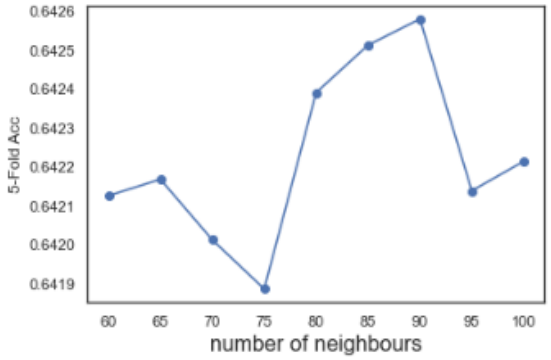



```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

KNN=KNeighborsClassifier()
n_neighbors=np.linspace(60,100,9).astype(int)
h_parameters = {'n_neighbors':n_neighbors}
cv = GridSearchCV(KNN, h_parameters, cv=2, n_jobs=-1, scoring='accuracy')
cv.fit(X_train_s, y_train)

cv.best_params_
```

```
{'n_neighbors': 90}
```

```python
print("best mean cross-validation score: {:.3f}".format(cv.best_score_))
print("best parameters: {}".format(cv.best_params_))

scores = np.array(cv.cv_results_['mean_test_score'])
plt.plot(n_neighbors,scores,'-o')
plt.xlabel('number of neighbours',fontsize=16)
plt.ylabel('5-Fold Acc')
plt.show()
```

```
best mean cross-validation score: 0.643
best parameters: {'n_neighbors': 90}
```




```python
# Este atributo nos devuelve el mejor estimador
cv.best_estimator_
```

```
KNeighborsClassifier(n_neighbors=90)
```

```python
Kopt = cv.best_params_['n_neighbors']
```

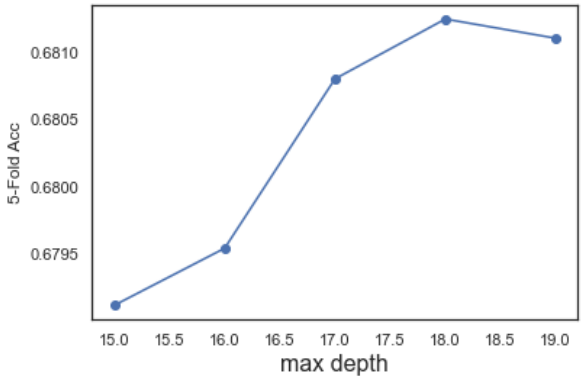
```python
from sklearn.neighbors import KNeighborsClassifier # Importamos el método

knn = KNeighborsClassifier(n_neighbors = Kopt)
knn.fit(X_train_s, y_train)
y_pred_knn = knn.predict(X_test_s)
```

APRENDIZAJE SUPERVISADO: RANDOM FOREST

- Método que ensambla múltiples árboles de decisión.
- Para que cada árbol sea diferente en este caso, lo que se hace es ajustar cada uno sobre una muestra bootstrap del dataset. De esta forma, cada modelo será diferente.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Grid search
maxDepth   = range(15,20)
param_grid = {'max_depth': maxDepth}

grid = GridSearchCV(RandomForestClassifier(n_estimators = 100, random_state=0), scoring= 'accuracy', param_grid=param_grid, cv = 5)
grid.fit(X_train_s, y_train)
print("best mean cross-validation score: {:.3f}".format(grid.best_score_))
print("best parameters: {}".format(grid.best_params_))

scores = np.array(grid.cv_results_['mean_test_score'])
plt.plot(maxDepth,scores,'-o')
plt.xlabel('max depth', fontsize=16)
plt.ylabel('5-Fold Acc')
plt.show()
```

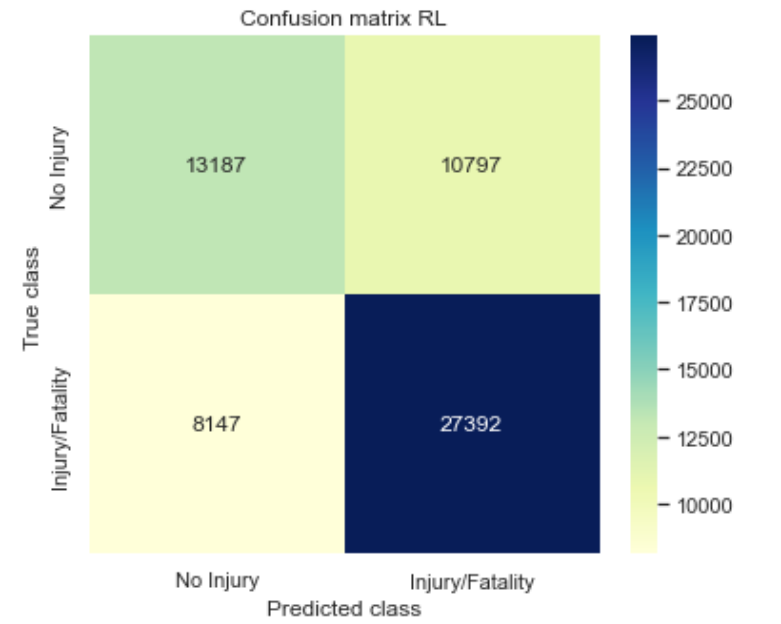
```
best mean cross-validation score: 0.681
best parameters: {'max_depth': 18}
```



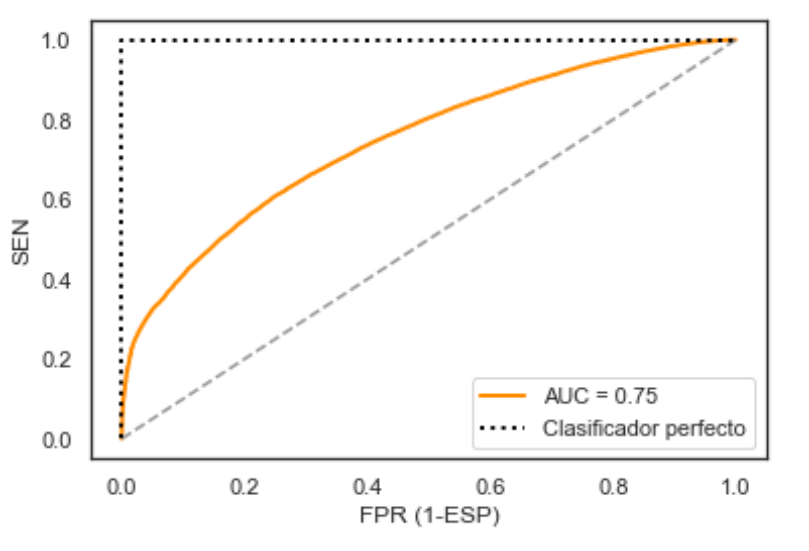

```
maxDepthOptimo = grid.best_params_['max_depth']
rf = RandomForestClassifier(n_estimators = 200, max_depth=maxDepthOptimo).fit(X_train_s,y_train)

#Modelo en el conjunto de datos de entrenamiento
print("Train: ",rf.score(X_train_s,y_train))
```

```
Train:  0.7514164871119623
```

APRENDIZAJE SUPERVISADO: RANDOM FOREST



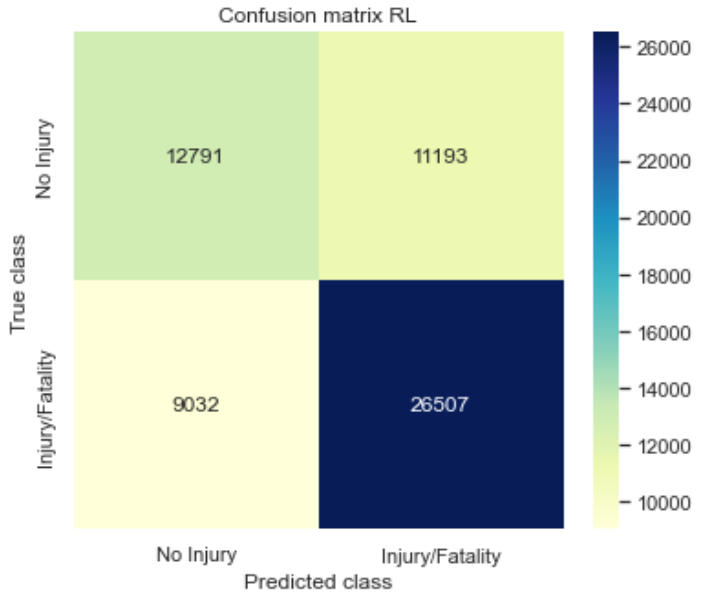

```
Accuracy: 0.681736471616014
Precision_NoInjury:  0.6181213087091029
Precision_Injury/Fatality :  0.7172746078713765
Recall_NoInjury :  0.5498248832555037
Recall_Injury/Fatality:  0.7707588846056445
F1_Score_Injury/Fatality:  0.7430555555555555
```



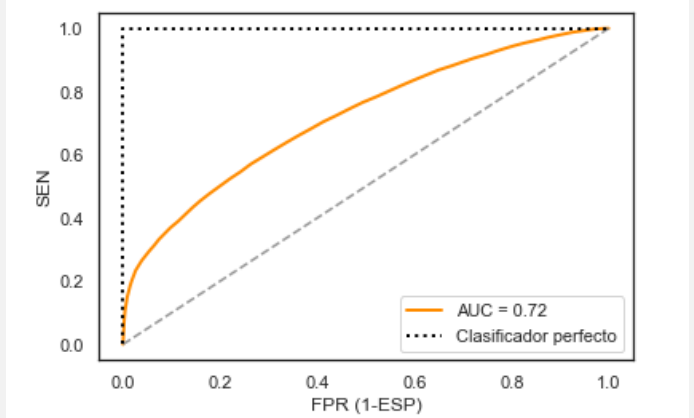
APRENDIZAJE SUPERVISADO: BAGGING

Los métodos de bagging son métodos donde los algoritmos simples son usados en paralelo. El principal objetivo de los métodos en paralelo es el de aprovecharse de la independencia que hay entre los algoritmos simples, ya que el error se puede reducir bastante al promediar las salidas de los modelos simples.

```
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import GridSearchCV

bagging = BaggingClassifier()
n_estimators=np.linspace(55,70,6).astype(int)
h_parameters = {'n_estimators':n_estimators}
cv = GridSearchCV(bagging, h_parameters, scoring='accuracy',cv=2,verbose=3).fit(X_train_s,y_train)
```

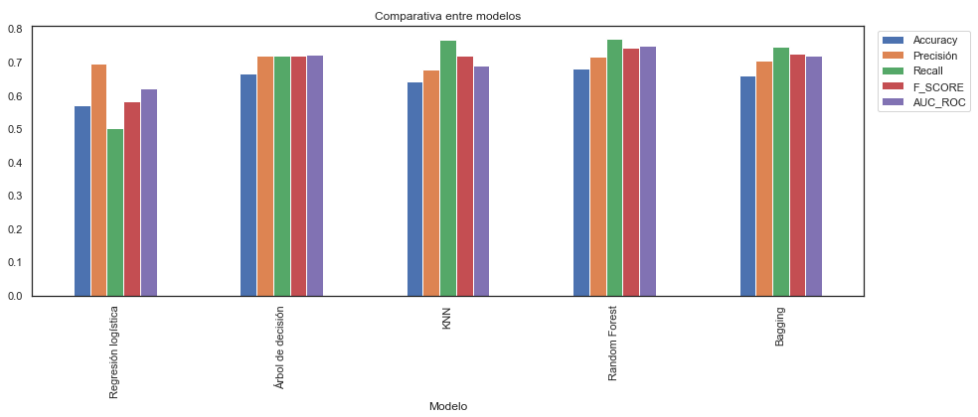



```
Accuracy: 0.6602153789291534
Precision_NoInjury:  0.5861247307886175
Precision_Injury/Fatality :  0.703103448275862
Recall_NoInjury :  0.5333138759172782
Recall_Injury/Fatality:  0.7458566645094122
F1_Score_Injury/Fatality:  0.7238493152555332
```



APRENDIZAJE SUPERVISADO: MÉTRICAS

	Modelo	Accuracy	Precisión	Recall	F_SCORE	AUC_ROC
3	Random Forest	0.681736	0.717275	0.770759	0.743056	0.747579
2	KNN	0.642962	0.677430	0.767439	0.719631	0.690011
4	Bagging	0.660215	0.703103	0.745857	0.723849	0.719074
1	Árbol de decisión	0.664197	0.718088	0.720392	0.719238	0.723494
0	Regresión logística	0.570889	0.695201	0.500914	0.582278	0.619815



Vamos a contrastar las métricas obtenidas mediante cada uno de los algoritmos para así poder compararlos y extraer conclusiones acerca de su utilidad.

Consideramos que la métrica más importante es el *Recall* ya que mide, de entre todos los accidentes con personas heridas/fallecidas realmente heridas, cuántos han sido clasificados como tal. En nuestro modelo estamos interesados en detectar la gravedad del accidente y, será peor para nosotros clasificar un accidente como no grave cuando no lo es realmente, que clasificarlo como grave y que finalmente no haya heridos. Es decir, el coste de un falso negativo es mayor que el de un falso positivo y, por tanto, estamos interesados en medir la sensibilidad/*recall* de cada algoritmo.

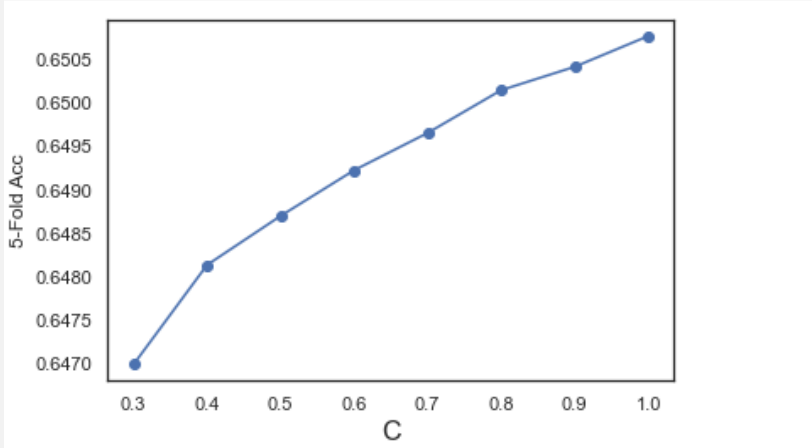
APRENDIZAJE SUPERVISADO: SVM

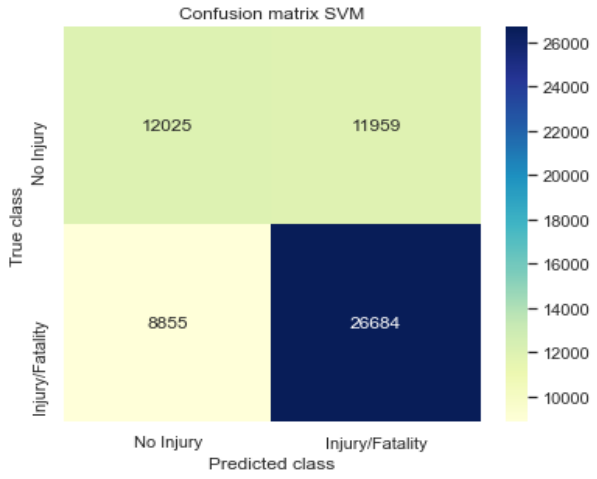
- De forma adicional se ha implementado el algoritmo de *Support Vector Machines*.
- El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos pero en lugar de buscar el hiperplano que mejor separa las clases, se buscan franjas o márgenes (support vector). En dichas franjas no tenemos diferencias suficientes entre las clases como para ser capaces de distinguir una clase de la otra. Por tanto, SVM es un problema de optimización en que trataremos de minimizar la variable objetivo que es, justamente, ese margen, sujeto a las restricciones que nos proporcionan los datos empleados para entrenar el algortimo.
- Vamos a emplear el *kernel Gaussiano* para proyectar nuestras observaciones a una dimensión más y que estas sean linealmente separables.

APRENDIZAJE SUPERVISADO: SVM

```python
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

vectorC=np.linspace(0.3,1,8)
param_grid = {'C': vectorC}

grid = GridSearchCV(SVC(), scoring='accuracy', param_grid=param_grid, cv=2, verbose=3).fit(X_train_s, y_train)

print("best mean cross-validation score: {:.3f}".format(grid.best_score_))
print("best parameters: {}".format(grid.best_params_))
```

```python
scores = np.array(grid.cv_results_['mean_test_score'])
plt.plot(vectorC,scores,'-o')
plt.xlabel('C',fontsize=16)
plt.ylabel('5-Fold Acc')
plt.show()
```






```
Accuracy: 0.6503200443526032
Precision_NoInjury:  0.5759099616858238
Precision_Injury/Fatality :  0.6905260978702482
Recall_NoInjury :  0.5013759172781854
Recall_Injury/Fatality:  0.7508371085286587
F1_Score_Injury/Fatality:  0.7194198053436144
```

CONCLUSIONES

- Con los modelos vistos, somos capaces de clasificar correctamente 7 de cada 10 accidentes y prever si habrá heridos o no. El algoritmo de *Random Forest* es el que mejor *recall* arroja, un 80%.
- Nuestros análisis pueden ser de gran utilidad para las aseguradoras, que tienen que cubrir los gastos médicos de sus asegurados. Este tipo de empresas suelen sufrir de problemas de información asimétrica porque la aseguradora casi nunca tiene toda la información acerca del cliente potencial:
  - Podría ser temerario de lo que aparenta (selección adversa)
  - Que una vez asegurado sea menos precavido y más propenso a cometer un accidente precisamente por estar asegurado (riesgo moral).

CONCLUSIONES

- Tratamos de solucionar parcialmente este problema obteniendo un predictor de la gravedad del accidente mediante distintas características del mismo.
- Pero existe margen de mejora para estudios futuros:
  - Por ejemplo, sería interesante contar con datos más actualizados ya que hoy en día la seguridad en los coches ha mejorado bastante, aunque existen otros peligros, como el uso del móvil al volante, a tener en cuenta.
  - Sería interesante comparar el estudio con otros similares realizados en diferentes países para ver qué patrones siguen y quizás aprender de otros donde la mortalidad sea inferior.
  - Recabar información sobre las condiciones meteorológicas podría ayudar también a reducir el número de accidentes, si el clima influye.