

Datos del alumno

Nombre:

Apellidos:

Grupo:

Introducción.

En este ejemplo vamos a trabajar con un conjunto de datos formado por un total de 14999 observaciones y 10 variables, consistente en estudiar diversas características de los empleados de una empresa en relación con su decisión de abandonar o no la empresa. Las variables en estudio, se van a describir posteriormente.

✓ Estudio descriptivo de los datos.

En este apartado vamos a familiarizarnos con la estructura que toman estos datos, así como con las diversas características de los mismos y posibles relaciones entre ellos.

Lo primero que hacemos es importar las librerías que vamos a necesitar y los datos en cuestión.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
%matplotlib inline
```

#Descarga manual: <https://drive.google.com/file/d/1ZwgdmI525zInDh1SQVm7FZt8kwxfrvOK/view?usp=sharing>

#Descargamos los ficheros de Google Drive

```
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1ZwgdmI525zInDh1SQVm7FZt8kwxfrvOK' -O 'Rotacion_em
```

```
--2024-09-10 14:35:17-- https://drive.google.com/uc?export=download&id=1ZwgdmI525zInDh1SQVm7FZt8kwxfrvOK
Resolving drive.google.com (drive.google.com)... 172.253.122.139, 172.253.122.138, 172.253.122.101, ...
Connecting to drive.google.com (drive.google.com)|172.253.122.139|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1ZwgdmI525zInDh1SQVm7FZt8kwxfrvOK&export=download [following]
--2024-09-10 14:35:17-- https://drive.usercontent.google.com/download?id=1ZwgdmI525zInDh1SQVm7FZt8kwxfrvOK&export=downlo
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.16.132, 2607:f8b0:4004:c06::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 551779 (539K) [application/octet-stream]
Saving to: 'Rotacion_empleados.csv'
```

```
Rotacion_empleados. 100%[=====] 538.85K --.-KB/s in 0.07s
```

```
2024-09-10 14:35:20 (7.54 MB/s) - 'Rotacion_empleados.csv' saved [551779/551779]
```




Si no conseguís descargar el fichero con el anterior comando, también lo tenéis a vuestra disposición en los **Recursos del proyecto** con el nombre `Rotacion_empleados.csv`.

```
# Leemos el conjunto de datos
datos = pd.read_csv("Rotacion_empleados.csv")
```

```
# obtenemos las filas y columnas qu contiene
datos.shape
```

```
(14999, 10)
```

```
# Veamos los primeros datos
datos.head()
```




	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	prc
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

Pasos siguientes:

[Generar código con datos](#)[Ver gráficos recomendados](#)[New interactive sheet](#)

#obtenemos información sobre las variables que contiene este data set

```
datos.info()
```




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   satisfaction_level           14999 non-null  float64
1   last_evaluation              14999 non-null  float64
2   number_project               14999 non-null  int64
3   average_monthly_hours        14999 non-null  int64
4   time_spend_company           14999 non-null  int64
5   Work_accident                14999 non-null  int64
6   left                         14999 non-null  int64
7   promotion_last_5years        14999 non-null  int64
8   sales                        14999 non-null  object
9   salary                       14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

✓ Significado y contenido de las variables.

- **satisfaction_level**: Es el nivel de satisfacción de los empleados, que va de 0-1.
- **last_evaluation**: Se evalúa el desempeño por parte del empleador, el cual también va de 0-1..
- **number_projects**: ¿Cual es el número de proyectos asignados a un empleado?
- **average_monthly_hours**: ¿Cuántas horas promedio trabaja un empleado en un mes?
- **time_spent_company**: significa experiencia del empleado. Número de años de permanencia de un empleado en la empresa.
- **work_accident**: si un empleado ha tenido un accidente de trabajo o no.
- **promotion_last_5years**: si un empleado ha tenido una promoción en los últimos 5 años o no..
- **sales**: departamento/división a la que pertenece el empleado.
- **Salary**: Nivel salarial del empleado en los niveles:low, medium and high.
- **left**: Si el empleado ha dejado la empresa o no. (0: No, 1: Sí)

Observemos que la variable con denominación "sales" no expresa adecuadamente el contenido de la variable, quizá sea más procedente denominarla `department`, que es lo que hacemos a continuación.

```
datos.rename(columns={'sales': 'department'}, inplace=True)
datos.head()
```



	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	prc
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

Pasos siguientes:

[Generar código con datos](#)[Ver gráficos recomendados](#)[New interactive sheet](#)

✓ Sprint 1

✓ Análisis descriptivo de los datos

Antes de iniciar la construcción de cualquier modelo de Machine Learning, es primordial realizar un análisis exploratorio que nos permita familiarizarnos con los datos y extraer las primeras conclusiones de los mismos.

✓ Ejemplo

Evaluamos el porcentaje de empleados que pertenecen a cada uno de los departamentos (variable que anteriormente se llamaba sales y ahora se llama department).

```
datos.department.value_counts(normalize=True)
```



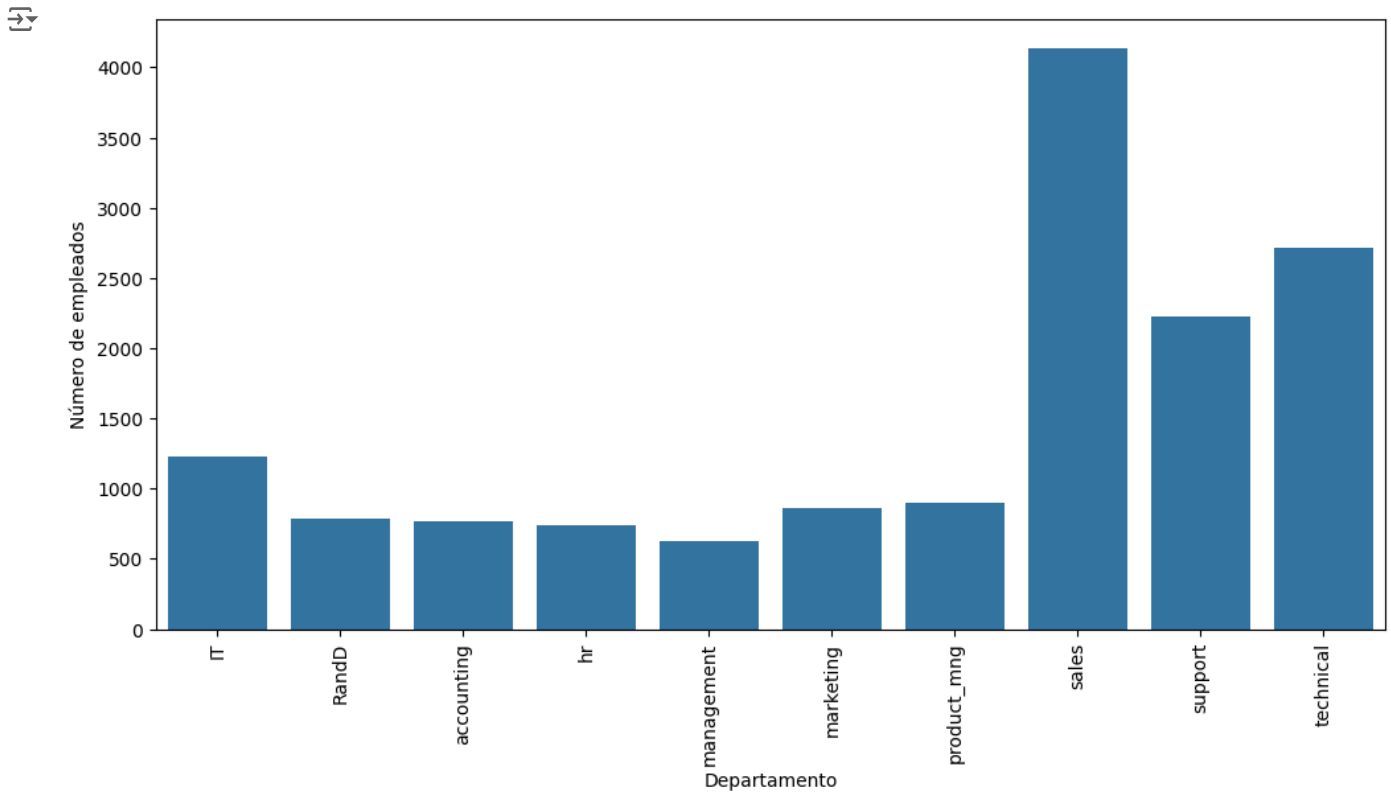
A table with two columns: 'department' and 'proportion'. The 'department' column lists ten departments: sales, technical, support, IT, product_mng, marketing, RandD, accounting, hr, and management. The 'proportion' column shows the corresponding percentage of employees for each department, ranging from 0.276018 for sales to 0.042003 for management. The table is presented in a scrollable format with a horizontal scrollbar at the bottom.

department	proportion
sales	0.276018
technical	0.181345
support	0.148610
IT	0.081805
product_mng	0.060137
marketing	0.057204
RandD	0.052470
accounting	0.051137
hr	0.049270
management	0.042003

Los datos anteriores en formato de gráfico, los obtendremos de la siguiente manera:

```
plt.figure(figsize=(12,6))
data_group = datos.groupby(by=['department'],as_index=False).count()
ax = sns.barplot(x="department", y="left", data=data_group)
ax.set(xlabel='Departamento', ylabel='Número de empleados')

plt.xticks(rotation = 90);
```



Podemos ver que casi un 30% de los empleados presentes en la muestra pertenecen al departamento de ventas.

✓ CUESTION 1

¡Ahora es tu turno! Responde a las siguientes preguntas con los gráficos que consideres más adecuados y su explicación correspondiente:

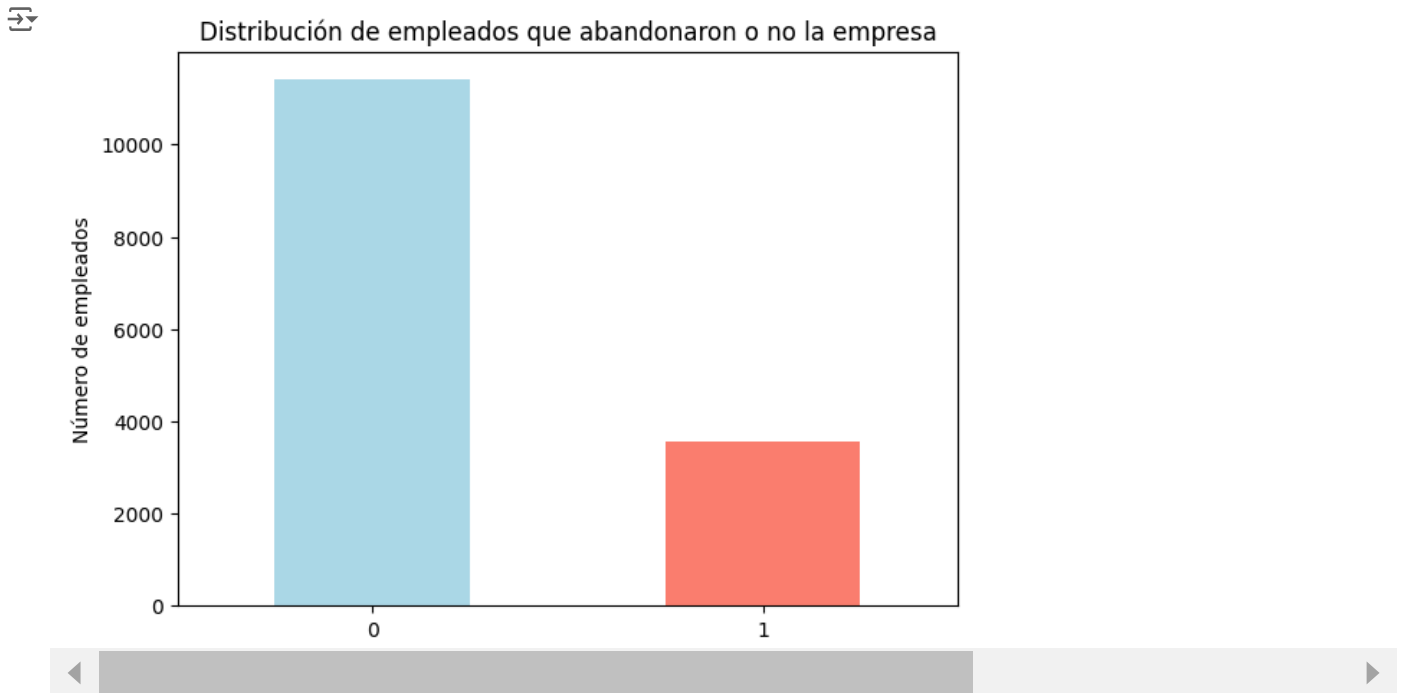
1. Si queremos predecir si un empleado abandona o no la empresa, ¿estamos ante un problema de clases balanceado o desbalanceado? Justifica tu respuesta incluyendo un gráfico que represente el número observaciones en cada una de las clases.

✓ Contar el número de empleados que han abandonado y que no han abandonado la empresa

```
left_distribution = datos['left'].value_counts()
```

✓ Graficar la distribución

```
plt.figure(figsize=(7,5))
left_distribution.plot(kind='bar', color=['lightblue', 'salmon'])
plt.title('Distribución de empleados que abandonaron o no la empresa')
plt.xlabel('Estado del empleado (0 = No, 1 = Sí)')
plt.ylabel('Número de empleados')
plt.xticks(rotation=0)
plt.show()
```



El gráfico anterior muestra la distribución de empleados que abandonaron (1) y que no abandonaron (0) la empresa:

11,428 empleados no abandonaron la empresa (valor 0). 3,571 empleados sí abandonaron la empresa (valor 1).

Conclusión:

Este es un problema de clases desbalanceadas, ya que hay una diferencia significativa entre las dos clases (empleados que se quedaron y empleados que se fueron). La clase de empleados que no abandonaron la empresa es considerablemente mayor que la de aquellos que sí lo hicieron. Este desbalance puede influir en el rendimiento de los modelos de Machine Learning y debe tenerse en cuenta al elegir y evaluar los algoritmos.

2. Estudiar cómo se distribuye la variable `left` en función de los siguientes atributos categóricos:

`number_project`, `time_spend_company`, `Work_accident`, `department`, `promotion_last_5years`, `salary`.

```
# Ajustamos el tamaño de los gráficos
plt.figure(figsize=(12,8))

# Gráfico 1: Distribución de 'left' por 'number_project'
plt.subplot(2, 3, 1)
sns.countplot(x='number_project', hue='left', data=datos, palette='Set2')
plt.title('Distribución por Número de Proyectos')

# Gráfico 2: Distribución de 'left' por 'time_spend_company'
plt.subplot(2, 3, 2)
sns.countplot(x='time_spend_company', hue='left', data=datos, palette='Set2')
plt.title('Distribución por Tiempo en la Empresa')

# Gráfico 3: Distribución de 'left' por 'Work_accident'
plt.subplot(2, 3, 3)
sns.countplot(x='Work_accident', hue='left', data=datos, palette='Set2')
plt.title('Distribución por Accidentes Laborales')

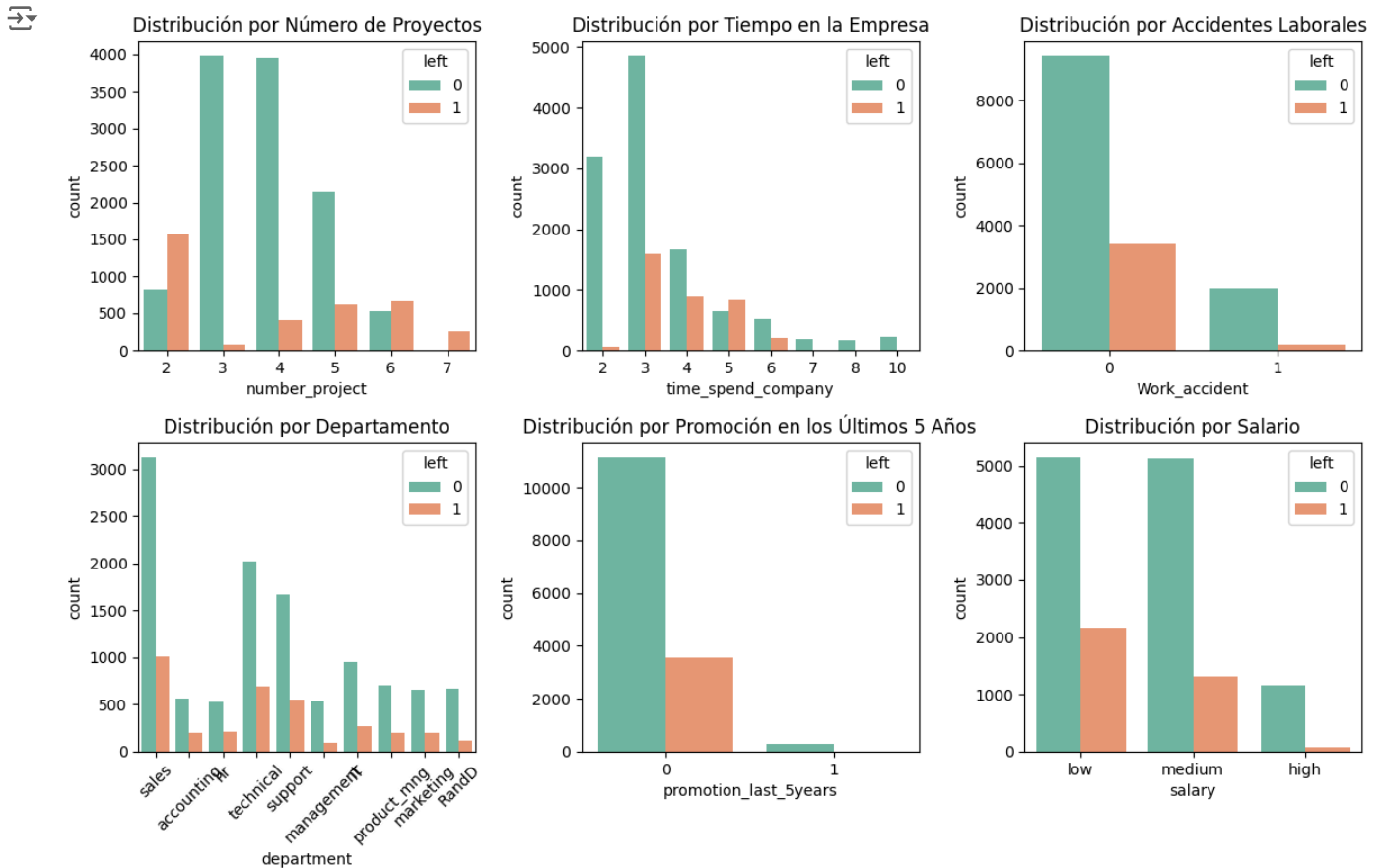
# Gráfico 4: Distribución de 'left' por 'department'
plt.subplot(2, 3, 4)
sns.countplot(x='department', hue='left', data=datos, palette='Set2')
plt.title('Distribución por Departamento')
plt.xticks(rotation=45)

# Gráfico 5: Distribución de 'left' por 'promotion_last_5years'
plt.subplot(2, 3, 5)
sns.countplot(x='promotion_last_5years', hue='left', data=datos, palette='Set2')
plt.title('Distribución por Promoción en los Últimos 5 Años')

# Gráfico 6: Distribución de 'left' por 'salary'
plt.subplot(2, 3, 6)
sns.countplot(x='salary', hue='left', data=datos, palette='Set2')
```

```
plt.title('Distribución por Salario')
```

```
plt.tight_layout()
plt.show()
```



Empieza a programar o a [crear código](#) con IA.

Os facilito una serie de enlaces que os pueden servir de inspiración para realizar esta parte:

- [Introducción a la estadística descriptiva](#): Para aquellos que de seen introducirse a la analítica descriptiva desde cero.
- [Análisis de datos categóricos con Python](#): Enfocado en el estudio de variables categóricas.
- [Análisis exploratorio de datos con Python](#): Estudia la relación entre variables categóricas y variables continuas.

Sin embargo, éstos no dejan de ser unos de los muchos ejemplos sobre cómo realizar el análisis exploratorio, así que tenéis total libertad para explorar y realizar cualquier análisis adicional que se os ocurra.

> Análisis predictivo

[] 4 celdas ocultas

✓ CUESTION 2

- Transformar la variable `salary` de forma que sus categorías se codifiquen de la siguiente manera: low = 0, medium = 1 y high = 2.
- Transformar la variable `department` en variables dummies.

Puedes volver a revisar el vídeo de la Clase 1 Tema 4 - **Laboratorio Preprocesamiento de datos: parte 2**, donde a partir del minuto 5:55 se explica en detalle cómo proceder en estos casos.

```
# Transformar la variable 'salary' a valores numéricos: low = 0, medium = 1, high = 2
datos['salary'] = datos['salary'].map({'low': 0, 'medium': 1, 'high': 2})
```

```
# Transformar la variable 'department' en variables dummies
# Usamos drop_first=True para evitar la multicolinealidad (removemos una categoría como referencia)
```

```
datos = pd.get_dummies(datos, columns=['department'], drop_first=True)
```

```
# Mostrar las primeras filas del DataFrame para verificar las transformaciones
print(datos.head())
```

```

satisfaction_level  last_evaluation  number_project  average_monthly_hours  \
0                0.38             0.53              2                   157
1                0.80             0.86              5                   262
2                0.11             0.88              7                   272
3                0.72             0.87              5                   223
4                0.37             0.52              2                   159

time_spend_company  Work_accident  left  promotion_last_5years  salary  \
0                 3              0     1                   0         0
1                 6              0     1                   0         1
2                 4              0     1                   0         1
3                 5              0     1                   0         0
4                 3              0     1                   0         0

department_RandD    department_accounting  department_hr  \
0                False                  False          False
1                False                  False          False
2                False                  False          False
3                False                  False          False
4                False                  False          False

department_management  department_marketing  department_product_mng  \
0                False                  False          False
1                False                  False          False
2                False                  False          False
3                False                  False          False
4                False                  False          False

department_sales  department_support  department_technical
0                True                False          False
1                True                False          False
2                True                False          False
3                True                False          False
4                True                False          False

```

> Implementación de los modelos

↳ 2 celdas ocultas

✓ CUESTION 3

Con las premisas anteriores obtener los conjuntos de train y test, de manera que el tamaño del conjunto del **test tenga el 20% de observaciones** sobre el conjunto total, y utilizando el parámetro **random_state=123**.

NOTA. Para que la celda posterior en la que guardamos los datos funcione, recomendamos llamar a los conjuntos de train y test resultantes de la siguiente manera: `X_train`, `X_test`, `y_train`, `y_test`.

```
from sklearn.model_selection import train_test_split
```

```

# Separar las características (X) de la variable objetivo (y)
X = datos.drop('left', axis=1) # Eliminar la columna 'left', que es nuestra variable objetivo
y = datos['left'] # Variable objetivo

```

```

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=123)

```

Estos conjuntos de entrenamiento y test los vamos a utilizar tal cual en futuros sprint, así que los guardamos de la siguiente manera:

```

np.save("X_train",X_train)
np.save("X_test",X_test)
np.save("y_train",y_train)
np.save("y_test",y_test)

```

```
# Verificar las dimensiones de los conjuntos resultantes
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

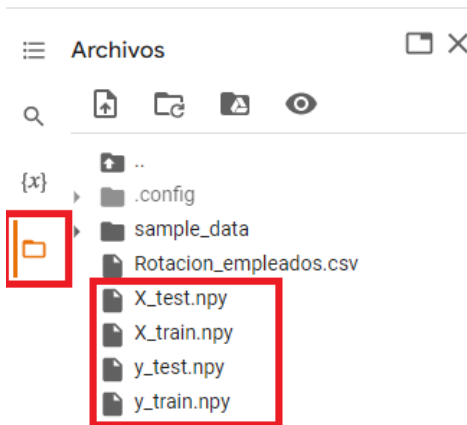
```
(11999, 17) (3000, 17) (11999,) (3000,)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
```

NOTA. Los ficheros anteriores se guardan con la extensión .npv, y así por ejemplo tendremos el fichero "X_train.npv", etc.

Si queréis conservar estas matrices, es importante que descarguéis a vuestro PC estos archivos generados antes de cerrar la sesión de Google Colab para evitar perderlos. Sin embargo, si no lo hacéis no pasa nada, ya que en los siguientes sprints se os dará como input.



✓ Carga de los conjuntos de train y test

En el anterior apartado hemos guardado los conjuntos de entrenamiento y test (son objetos de tipo np.array). Lo que deberemos hacer en los diferentes pasos tanto de este sprint como de los siguientes es recuperar estos datos para trabajar con ellos. Lo haremos con la siguiente instrucción:

Si no conseguís ejecutar los datos anteriores podeis descomentar las siguientes instrucciones para obtener las matrices resultantes

```
##Descarga manual: https://drive.google.com/file/d/1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/101BRGwSd81T00paQvoo5BHDsrPERuz93/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/1u3yI5L_2YI77uF_WT9ysIzw8dydV1QSh/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/1eWo_m2gbihSuUQe0hH8I8Q0uudluQKBT/view?usp=sharing
```

```
##Descargamos los ficheros de Google Drive (si lo ejecutais en un entorno diferente a Google Colab, tenéis que intalar previamente)
#wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9' -O 'y_train.npy'
#wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=101BRGwSd81T00paQvoo5BHDsrPERuz93' -O 'y_test.npy'
#wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1u3yI5L_2YI77uF_WT9ysIzw8dydV1QSh' -O 'X_train.npy'
#wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1eWo_m2gbihSuUQe0hH8I8Q0uudluQKBT' -O 'X_test.npy'
```

```
#X_train, X_test,y_train,y_test = np.load("X_train.npy"),np.load("X_test.npy"),np.load("y_train.npy"),np.load("y_test.npy")
import numpy as np
```

```
# Cargar los archivos .npv usando allow_pickle=True
X_train = np.load("X_train.npy", allow_pickle=True)
X_test = np.load("X_test.npy", allow_pickle=True)
y_train = np.load("y_train.npy", allow_pickle=True)
y_test = np.load("y_test.npy", allow_pickle=True)
```

Empieza a programar o a [crear código](#) con IA.

De esta manera ya estaremos dispuestos en este sprint y en los de las siguientes semanas a trabajar con estos conjuntos de datos.

Ahora procedemos a entrenar los modelos vistos durante esta semana adecuados para predecir la variable left.

✓ CUESTION 4

Entrenar con los datos de train y evaluar sobre datos de test dos de los algoritmos vistos en la lección para predecir si un empleado se va a ir o no de la empresa.

✓ Modelo 1

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

```
# Crear el modelo de Regresión Logística
logreg = LogisticRegression(max_iter=1000)
```

```
# Entrenar el modelo
logreg.fit(X_train, y_train)
```

```
# Realizar predicciones sobre el conjunto de prueba
y_pred_logreg = logreg.predict(X_test)
```

```
# Evaluar el modelo
print("Regresión Logística:")
print("Accuracy:", accuracy_score(y_test, y_pred_logreg))
print(classification_report(y_test, y_pred_logreg))
```

```
➡ Regresión Logística:
Accuracy: 0.7913333333333333
```

	precision	recall	f1-score	support
0	0.82	0.93	0.87	2291
1	0.60	0.35	0.44	709
accuracy			0.79	3000
macro avg	0.71	0.64	0.66	3000
weighted avg	0.77	0.79	0.77	3000

Conclusión:

El modelo de Regresión Logística tiene un buen desempeño para predecir empleados que no abandonan la empresa (Clase 0), pero tiene problemas importantes para capturar correctamente los empleados que sí abandonan la empresa (Clase 1), como lo demuestra el bajo valor de recall (35%) y F1-score (44%) para esta clase.

Dado que el problema que estás abordando es predecir la rotación de empleados, lo más importante es predecir correctamente la Clase 1 (empleados que abandonan). Este modelo puede no ser el más adecuado porque no está capturando bien esta clase.

✓ Modelo 2

```
# Crear el modelo de Random Forest
rf = RandomForestClassifier(random_state=123)
```

```
# Entrenar el modelo
rf.fit(X_train, y_train)
```

```
# Realizar predicciones sobre el conjunto de prueba
y_pred_rf = rf.predict(X_test)
```

```
# Evaluar el modelo
print("Random Forest:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
➡ Random Forest:
Accuracy: 0.9933333333333333
```

	precision	recall	f1-score	support
0	0.99	1.00	1.00	2291
1	0.99	0.98	0.99	709

accuracy			0.99	3000
macro avg	0.99	0.99	0.99	3000
weighted avg	0.99	0.99	0.99	3000

Conclusión:

El modelo de Random Forest muestra un rendimiento sobresaliente tanto en la Clase 0 (empleados que no abandonan) como en la Clase 1 (empleados que abandonan). Tiene una precisión, recall y F1-score muy altos en ambas clases, lo que indica que este modelo es muy eficaz para predecir correctamente si un empleado abandonará la empresa o no.

En comparación con el modelo de Regresión Logística, este modelo es significativamente mejor en términos de capturar la clase minoritaria (empleados que abandonan la empresa), lo que lo convierte en la mejor opción para este problema.

✓ CUESTION 5

Analizar, discutir y comparar los resultados obtenidos por cada uno de los modelos y dejar reflejadas las conclusiones obtenidas en el notebook que se entregue.

El entrenamiento del modelo por sí solo no nos aporta una información relevante, sino que es igual de importante saber interpretar los resultados obtenidos. Es fundamental a investigar otras métricas aparte del accuracy, para saber si el modelo se comporta igual de bien para predecir ambas clases. Unos ejemplos podrían ser:

- Matriz de confusión.
- Recall, precision, f1-score, etc. - `classification_report()`.
- [Curva ROC y AUC](#) - `RocCurveDisplay()`.

```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc, RocCurveDisplay
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Matriz de confusión
def plot_confusion_matrix(model, X_test, y_test, title):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Matriz de Confusión - {title}')
    plt.xlabel('Predicción')
    plt.ylabel('Verdadero')
    plt.show()

# Curva ROC y AUC
def plot_roc_curve(model, X_test, y_test, title):
    RocCurveDisplay.from_estimator(model, X_test, y_test)
    plt.title(f'Curva ROC - {title}')
    plt.show()

# Regresión Logística
plot_confusion_matrix(logreg, X_test, y_test, 'Regresión Logística')
plot_roc_curve(logreg, X_test, y_test, 'Regresión Logística')

# Random Forest
plot_confusion_matrix(rf, X_test, y_test, 'Random Forest')
plot_roc_curve(rf, X_test, y_test, 'Random Forest')
```

