EJERCICIO 1. Mira el vídeo de este enlace sobre la diferencia entre algoritmos de aprendizaje y
modelos en Machine Learning y responde a estas preguntas:

a) ¿Qué tarea quieren que haga el sistema de Machine Learning?

Clasificar adecuadamente los dos tipos de cliente que hay entre inversor y consumidor

b) ¿Qué datos se usan de cada cliente?

Se usará el nivel de ingreso y el nivel de endeudamiento de cada cliente

c) ¿Qué modelo se usa en el vídeo? El modelo estará formado por unos parámetros y un criterio
de decisión ¿Cuales son sus parámetros? ¿Y el criterio?

El modelo de Machine Learning. Tendrá la ecuación de la línea recta con los parámetros finales encontrados por el algoritmo y el criterio donde se hará una clasificación, donde se reemplazará el nivel de endeudamiento y de ingreso para obtener, el cual si es positivo tendrá categoría de inversor y si es negativo consumidor. Con esto podremos llegar a crear predicciones para nuestros datos

f) ¿Cuál es el primer paso del algoritmo?

El de asignar unos valores iniciales totalmente aleatorios a los parámetros que queremos encontrar para garantizar que no se va a influir de manera alguna en los resultados que arroje el algoritmo y también que los parámetros serán encontrados de forma automática, a partir de los datos

Con este paso puede que muchos clientes no sean clasificados de manera correcta todavía, por lo que habrá adicionalmente seguir los siguientes pasos

Tomar la primera recta construida con los parámetros aleatorios, reemplazar los niveles de ingreso y endeudamiento de cada cliente y en base a los resultados obtenidos (positivos o negativos), clasificar a cada cliente como inversor o consumidor (Generar una predicción)

EJERCICIO 2. Consulta el artículo sobre un inconveniente del ML y:

a) Resume en un par de líneas el problema que presenta

El problema que representa es que el Machine Learning produce un gran aumento de las emisiones de carbono, por culpa de los costes energéticos de entrenar modelos, que son altos

b) ¿En qué consisten las 4M que se proponen para solucionarlo?

Consisten en cuatro prácticas que reducirán significativamente las emisiones de carbono y energía.

Modelo, con arquitecturas de modelos de ML más eficientes (modelos dispersos) que pueden mejorar la calidad del ML y reducir el procesamiento entre 3 y 10 veces

Máquina, con procesadores y sistemas optimizados para el entrenamiento del ML, en vez de usar los procesadores de uso general. Con esto se puede mejorar el rendimiento y la eficiencia entre 2 y 5 veces

Mecanización, con la que usar computación en la nube en vez de usar instalaciones reduce el consumo de energía, además de las emisiones entre 1,4 y 2 veces. Los centros que albergan la nube están diseñados a medida y equipados para contener 50000 servidores, ofreciendo así una buena eficacia en el uso de la energía

Optimización de mapas, donde se permite elegir a los clientes la ubicación con la energía más limpia, lo que reduce la huella de carbono bruta entre 5 y 10 veces

Las 4M juntas pueden reducir el consumo de energía en 100 veces y las emisiones en 1000 veces

EJERCICIO 3. Mira el vídeo de este enlace sobre cómo saber si es apropiado usar Machine Learning para resolver un problema o no es una buena idea y sería mejor usar otras alternativas:
a) Escribe la definición de ML que indican en el vídeo

El Machine Learning es un enfoque que permite aprender patrones complejos a partir de datos existentes, y usar estos patrones para realizar predicciones sobre datos nunca antes analizados

b) Escribe un árbol de decisión que esquematice cuando sería buena idea y cuando no.
Nota: Un árbol de decisión es un árbol binario invertido (la raíz arriba) donde cada nodo equivale a una pregunta con dos posibles respuestas de tipo SI/NO a la pregunta del nodo. y su se sigue la respuesta te lleva a una nueva pregunta o a un nodo final que indica una solución. Ejemplo:

Ante cualquier dilema ético que pueda tener consecuencias negativas es mejor no usar el ML ni ningún otro enfoque

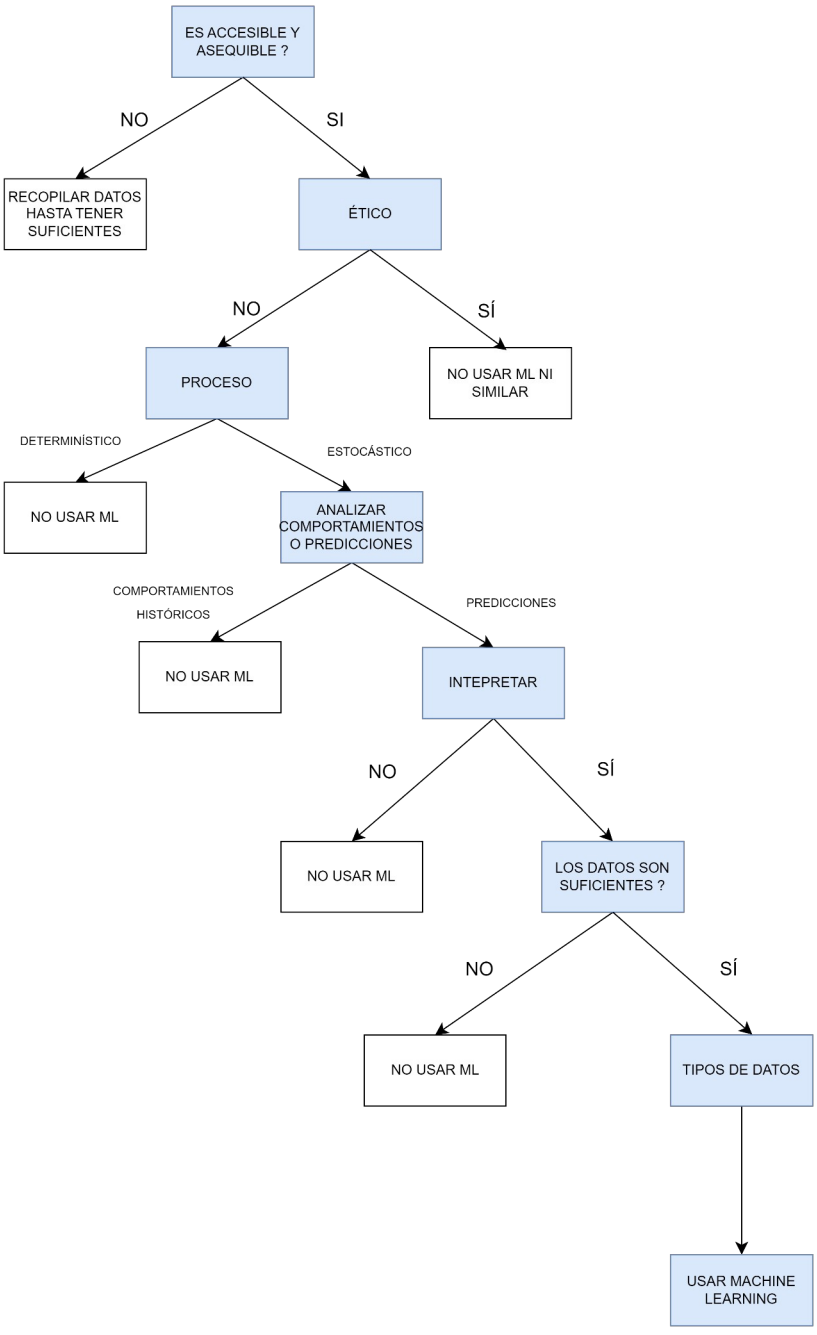
Si los datos obedecen a un proceso determinístico es mejor no usar el ML, además de no ser necesario (determinístico es aleatorio)
Si los datos obedecen a un proceso estocástico sí se puede usar el ML (ejemplo predecir unas ventas futuras con datos de ventas anteriores)

Para saber si los datos son suficientes para el ML podemos usar 3 reglas básicas
Si es un problema de clasificación podemos definir un factor dependiendo del número de clases. Por cada categoría se requieren 100, 1000 o 10000 ejemplos de entrenamiento
Se puede definir un factor dependiendo del número de carácteristicas de cada dato. Por ejemplo si hay 9 carácteristicas el número de ejemplos puede ser de 100, 1000 o 10000 por cada una de ellas

La última regla se puede definir según el número de parámetros. Si es de 100000 parámetros podrá ser de 5 o hasta 50 veces

Si los datos recolectados para el ML no son suficientes la solución es Recolectar más datos
Si sí hay una cantidad de datos adecuada estamos más cerca de confirmar que el ML es la solución..
Después habrá que ver si los datos son estructurados o no estructurados

Si los datos son estructurados la solución es el ML (si los datos son estructurados pero para realizar la prediccion no podemos codificar las reglas explicitamente la solución es el ML). Si los datos son no estructurados o están combinados con datos estrucutrados la solución también es el ML



EJERCICIO 4. Responde a estas cuestiones:
1. ¿Cómo podrías definir con tus palabras lo que es el Machine Learning?

El machine learning es una parte de la inteligencia artificial donde se crean algoritmos para enseñar a las máquinas a que aprendan de datos para hacer algo

2. ¿Puedes nombrar 4 tipos de problemas donde haya tenido éxito?

En el reconocimiento de imágenes, detección de fraudes, predicción de fallos, asistentes virtuales..

3. ¿Qué es un training set etiquetado?

Es un conjunto de datos dividido donde cada ejemplo incluye inputs y outputs para entrenar modelos de machine learning para que sepan aprender la relación entre entrada y salida

4. ¿Cuales son las dos tareas más comunes del aprendizaje supervisado?

Clasificar (clasificación) y regresión

5. ¿Puedes dar nombre a 4 algoritmos de aprendizaje supervisado?

Sí, por ejemplo Regresión Lineal, Regresión Logística, Random Forest y Redes Neuronales

6. ¿Qué tipo de algoritmo de aprendizaje usarías para permitir a un robot andar por terrenos desconocidos?

Aprendizaje reforzado sería mi elección

7. ¿Qué tipo de algoritmo podrías usar para segmentar a tus clientes en varios grupos?

Podría utilizar un aprendizaje no supervisado con K-means o DBSCAN por ejemplos

8. Si necesitas implementar un sistema de filtrado de correos electrónicos que detecte si un email es o no spam, este problema sería de aprendizaje supervisado o no supervisado?

Aprendizaje supervisado, de clasificación más concretamente

9. ¿Qué es un sistema de aprendizaje online? ¿Cómo se llaman los que no son de este tipo? ¿Qué ventajas suele tener con respecto a los otros?

El sistema de aprendizaje online es el que actualiza su modelo continuamente a medida que recibe datos. Los que no son de este tipo son los de aprendizaje por lotes o fuera de línea
La principal ventaja es poder adaptar el modelo en tiempo real

10. ¿Qué significa el término out-of-core learning?

Es una técnica que pertenece a la categoría de aprendizaje online aunque se realiza offline y se refiere a la técnica de entrenar sistemas con conjuntos de datos tan grandes que en la memoria RAM normal no pueden contenerse. Estos datos que se cargan y se procesan lo hacen en fragmentos más pequeños para que la memoria no se sature

11. ¿Qué tipo de algoritmos de aprendizaje usan una medida de similitud para realizar predicciones?

El aprendizaje supervisado (k-Nearest Neighbors (K-vecinos cercanos) y aprendizaje no supervisado (clustering)

12. ¿Qué diferencia hay entre los parámetros de un modelo y los hiperparámetros de los algoritmos de aprendizaje? Indica un ejemplo de cada uno

Los parámetros del modelo son los que el modelo aprende durante el entrenamiento (coeficiente en una regresión lineal) y los hiperparámetros son las configuraciones que el desarrollador elige antes del entrenamiento (profundidad de un árbol en un modelo Random Forest). Ejemplos podrían ser el peso en una red neuronal y el número de neuronas en una capa oculta como hiperparámetro

13. ¿Qué busca un algoritmo de aprendizaje basado en modelo? ¿Cuál es el enfoque más común que utilizan para realizar este trabajo? ¿Cómo hacen las predicciones?

Busca encontrar el mejor modelo matemático posible que represente los datos, generalizando patrones a partir de los datos de entrenamiento para hacer predicciones en los nuevos datos. Se trata de ajustar una función a los datos para minimizar la diferencia entre las predicciones y los valores reales.
Las predicciones se hacen con el modelo tomando las entradas y procesándolas a través de la función aprendida para generar las salidas

14. Indica 4 desafíos que tenga el machine learning

Obtener datos de calidad, que pueden ser muy costosos y necesitar mucho trabajo
Overfitting (sobreajuste), que el modelo se ajuste más de la cuenta a los datos de entrenamiento, dificultando así su generalización a los nuevos datos
Escalabilidad, donde entrenar modelos con grandes volúmenes de datos requieren recursos muy grandes
Interpretabilidad, donde los modelos más complejos pueden ser muy difícil de interpretar, lo cual hace que sea un reto para la confianza en las predicciones del mismo

15. Si tu modelo funciona bien con los datos de entrenamiento pero no generaliza bien con los nuevos datos con los que debe trabajar ¿Qué está ocurriendo? Puedes nombrar 3 posibles causas. Puedes nombrar 3 posibles soluciones

Posibles causas son que no hayan suficientes datos para entrenar, que el modelo sea muy complejo o que haya mucho ruido
Posibles soluciones serían tener más datos, reducir la complejidad del modelo y para el ruido aplicar técnicas de regularización L1 o L2 que ayudan a reducir el sobreajuste

16. ¿Qué es un test set y porqué es necesario usarlo?

Es un conjunto de datos de prueba para medir lo bien o mal que generaliza el modelo cuando debe trabajar con datos nuevos. Se utiliza un 20 % o 30 % aproximadamente de todos los datos para el test set y el resto para el training set. El test set es necesario porque con él se puede comprobar como de bien generaliza el modelo con datos nuevos y así evaluar si está listo para aplicarse en la práctica

17. ¿Cuál es el objetivo de utilizar un validation set?

El objetivo del validation set es ajustar los hiperparámetros del modelo durante el entrenamiento para encontrar la configuración óptima sin sobreajustar el modelo

18. ¿Qué puede ir mal si tuneas los hiperparámetros de un modelo usando el test set?

Que el modelo se ajuste indirectamente a estos nuevos datos y el test set deje de ser una medida independiente y objetiva del rendimiento lo que puede hacer que haya un sobreajuste y la evaluación del modelo sea poco realista ya que los resultados en el test set no reflejarán bien el rendimiento en datos realmente nuevos

EJERCICIO 5. Tenemos estos datos sobre diferentes coches. Responde a estas cuestiones:

	A	B	C	D	E	F	G	H	I
1	mpg	cilindros	desplazamiento	potenciaCV	peso	aceleracion	modelo_year	origen	nombre
2	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
3	15	8	350	165	3693	11.5	70	1	buick skylark 320
4	18	8	318	150	3436	11	70	1	plymouth satellite
5	16	8	304	150	3433	12	70	1	amc rebel sst
6	17	8	302	140	3449	10.5	70	1	ford torino
7	15	8	429	198	4341	10	70	1	ford galaxie 500
8	14	8	454	220	4354	9	70	1	chevrolet impala
9	14	8	440	215	4312	8.5	70	1	plymouth fury iii
10	14	8	455	225	4425	10	70	1	pontiac catalina
11	15	8	390	190	3850	8.5	70	1	amc ambassador dpl
12	15	8	383	170	3563	10	70	1	dodge challenger se
13	14	8	340	160	3609	8	70	1	plymouth 'cuda 340
14	15	8	400	150	3761	9.5	70	1	chevrolet monte carlo
15	14	8	455	225	3086	10	70	1	buick estate wagon (sw)
16	24	4	113	95	2372	15	70	3	toyota corona mark ii
17	22	6	198	95	2833	15.5	70	1	plymouth duster
18	18	6	199	97	2774	15.5	70	1	amc hornet
19	21	6	200	85	2587	16	70	1	ford maverick
20	27	4	97	88	2130	14.5	70	3	datsun pl510
21	26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan

a) ¿Cuántos ejemplos hay?

Hay 20 ejemplos

b) ¿Cuántas característica tiene cada uno?

Tiene 8 características (mpg, cilindros, desplazamiento, potenciaCV, peso, aceleración, modelo_year y origen)

c) Si eliminamos la característica nombre, y queremos predecir el valor de la aceleración a partir del resto de características ¿Cuántas predictoras hay? ¿Cuál es la columna label? ¿Cuál es la columna target? ¿Qué tipo de algoritmo de aprendizaje debemos usar supervisado o no supervisado? ¿Qué tarea estamos haciendo una regresión o una clasificación?

Hay 1 predictora
La columna label es aceleración porque es la que queremos predecir
La columna target es aceleración también porque es la que debemos predecir
Deberemos usar el aprendizaje supervisado porque tenemos etiquetas en el conjunto de datos
Estamos haciendo una regresión porque el valor de aceleración que queremos predecir es una variable continua

d) Si creamos una división en train y test y queremos dejar el 30% de los ejemplos para el algoritmo de entrenamiento del modelo ¿Cuántos ejemplos usará este algoritmo para aprender? ¿Y cuantos para testear el modelo calculado?

Para aprender utilizará el 70 % de los datos y para testear el modelo calculado el 30 % restante

EJERCICIO 6: Mira el vídeo donde se explica en qué consiste el aprendizaje reforzado y responde a estas cuestiones:
a) ¿El aprendizaje reforzado siempre se basa en modelos o nunca lo hace?

Se basa ser en modelos o libre de modelos

b) ¿En resolver qué juegos han tenido éxito los algoritmos de aprendizaje reforzado?

En el juego Go, juegos de Atari antiguos

c) ¿A qué datos tienen acceso los algoritmos de aprendizaje reforzado libres de modelo?

A las recompensas que reciben por sus acciones y a los estados del entorno

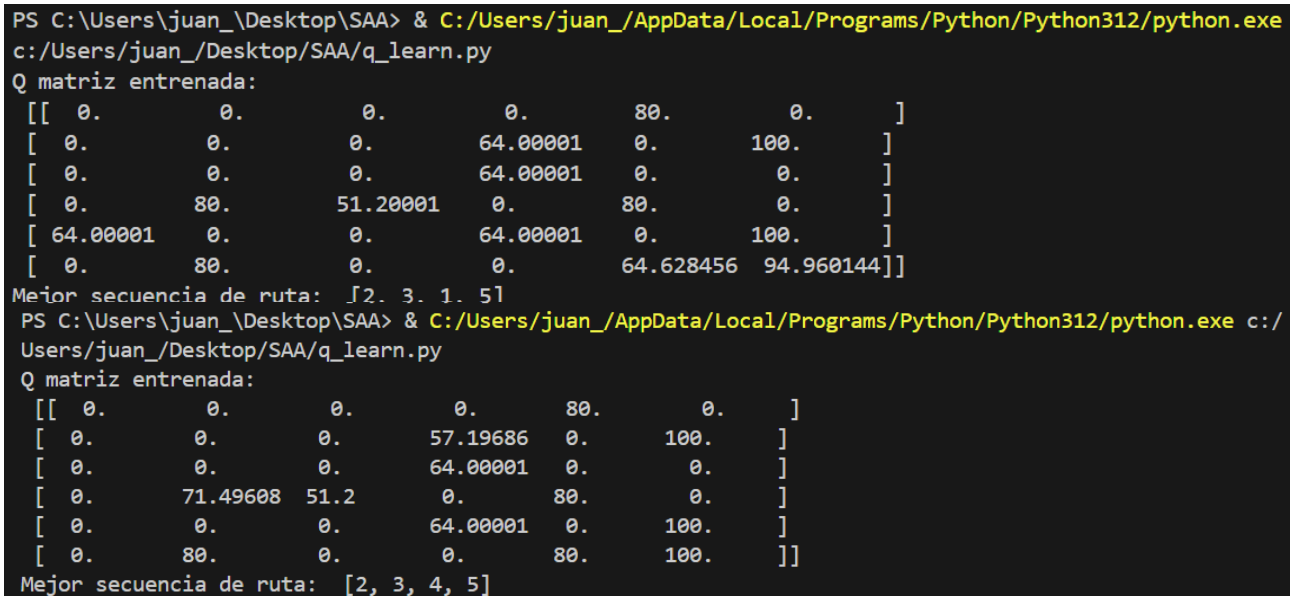
d) ¿Cuáles son los dos algoritmos más importantes del aprendizaje reforzado?

Son Q-Learning y política de gradientes
SARSA (State Action Reward State Action

e) Qué problema ayuda a solucionar el Machine Learning a estos algoritmos?

El aprendizaje reforzado

EJERCICIO 7: Copia el código del ejemplo 1 en el fichero q_learn.py, realiza dos ejecuciones y responde a las preguntas:
a) Entrega capturas del resultado de las ejecuciones



b) Si la primera fila de la matriz R representa estar en la habitación (a) ¿A qué lugares puedes moverte desde ella? ¿Qué recompensa obtiene el agente si lo hace?

Podrá moverse a e y la recompensa será de 80 puntos

c) Lo que aprende el algoritmo ¿Lo almacena en?

Lo almacena en la Q-matriz

d) Si está en la habitación d, ¿Qué dos acciones tienen la mayor recompensa?

En el primer caso acceder a las habitaciones b y e que son 80 puntos
En el segundo caso acceder a las habitaciones b y e también

EJERCICIO 8: Mira el vídeo sobre SAA_U01_EDA_Análisis exploratorio de datos.mp4 y responde a estas preguntas:
a) Enumera los pasos que definen el proceso

Extraer información relevante para después procesarlos
Paso 1 → Tener clara la pregunta a responder
Paso 2 → Tener una idea general de nuestro dataset
Paso 3 → Definir tipos de datos que tenemos
Paso 4 → Elegir el tipo de estadística descriptiva
Paso 5 → Elegir la visualización a utilizar
Paso 6 → Analizar las posibles interacciones entre las variables del dataset
Paso 7 → Sacar conclusiones de todo el análisis

b) En el paso 3 ¿Qué tipos de características nos podemos encontrar en un dataset?

Las variables numéricas o las variables categóricas
Numéricos → Discrectos (números enteros), Continuos (cualquier valor)
Categóricos → Nominales (etiquetar datos), Binarios (una de dos opciones), Ordinales (orden)

c) En el paso 4 ¿Qué métricas se utilizan para describir los datos de una característica?

Se utilizan las de tendencia central y lal variabilidad
Tendencia central → Nos dan una idea general del valor típico que pueden tener nuestros datos (media, mediana)

Variabilidad → Idea general de como de agrupados o dispersos se encuentran los datos (desviación estándar, rango intercuartiles)

d) ¿Qué desventaja tiene la media y la desviación estándar para describir una característica numérica? ¿Hay alguna otra métrica que no tenga esta desventaja?

La desventaja es que las dos son muy sensibles a los valores atípicos (outliers

La mediana y el IQR pueden ser soluciones

e) ¿Qué es el IQR? f) ¿Qué es la mediana, el percentil 50 y el cuartil 2?

El IQR (Interquartile Range) es una medida de dispersión que representa la diferencia entre el cuartil 3 y el cuartil 1
La mediana, el percentil 50 y el cuartil 2 representan lo mismo, el valor central de un conjunto de datos ordenados. Este valor divide los datos en dos mitades, donde el 50 % de los valores están por debajo de él y el 50 % está por encima. Esta medida de tendencia central es resistente a valores atípicos

g) ¿Qué porcentaje de los datos están entre el cuartil 1 y el cuartil 3? ¿Y entre el percentil 25 y el percentil 75? ¿Y qué porcentaje de datos son mayores del cuartil 3?

Entre el cuartil 1 y el cuartil 3 está el percentil 25
Entre el percentil 25 y el percentil 75 está la mediana y el 50 % de los datos
El porcentaje de datos mayores del cuartil 3 es el 25 % ya que este último representa el 75 %

h) Para graficar (representar gráficamente) como se distribuyen los datos de una característica ¿Qué dos tipos de gráficos se suelen utilizar y para qué tipo de características?

Se utilizan el histograma para representar la distribución de datos númericos continuos o discretos, mostrando la frecuencia de los valores en intervalos específicos y el bloxplot para representar la distribución de datos númericos de una forma que permite identificar fácilmente los cuartiles, el rango IQR y los valores atípicos

i) La matriz de correlaciones todas las variables numéricas de un dataset ¿Qué valores tiene en su diagonal?

En la diagonal los valores son 1, ya que cada variable está perfectamente correlacionada consigo misma

EJERCICIO 9: Mira el vídeo SAA_U01_Ingeniería_de_características.mp4 y responde a estas preguntas:
a) Enumera los problemas que según el vídeo pueden tener los datos

Los problemas pueden ser que los datos están incompletos, que los datos no tengan una distribución normal, que los datos no tengan la escala adecuada, que alguna columna tenga datos categóricos y requieran ser numéricas, que sean numéricas pero los datos tengan valores extremos tanto grandes como pequeños y que algunas variables sean irrelevantes y que hayan muchas o muy pocas variables para el modelo

b) Enumera como se llaman las técnicas usadas en ingeniería de características para solucionar los problemas de los datos

Descarte que consiste en eliminar las filas en el set de datos donde alguna fila esté faltante y amputación que consiste en asignar o estimar el valor del dato faltante haciendo uso de diferentes técnicas

c) Indica los tipos de características categóricas podemos encontrarnos y el motivo de de querer transformarlas en información numérica. ¿Cómo se llaman las transformaciones?

Variables ordinales y variables nominales
Las transforamaciones se llaman Label Encoding y One-Hot Encoding

EJERCICIO 10: Mira el vídeo SAA_U01_Selección_de_características.mp4 y responde a estas preguntas:
a) ¿Qué métodos utiliza la selección de características?

El método de filtrado, el método de envoltura (wrapper) y el método embebido

b) ¿Porqué puede resultar beneficioso descartar características y no usarlas para aprender?

Porque así el tiempo de cómputo se reducirá y sin overfitting

c) El método de filtrado, ¿Necesita construir el modelo?

No, no necsita construir el modelo

d) En el método de filtrado ¿Cambian las características a descartar si cambias el modelo a construir?

No, no cambian las características generalmente porque es independiente del modelo con este método

e) Si tenemos 10 características y queremos quedarnos con las 6 más importantes y para ello usamos el método wrapper hacia adelante, ¿Cuántos modelos debemos entrenar y validar para descartar las 4 peores?

Debemos entrenar una a una las características evaluando el desempeño añadiendo la mejor. Después con ella la combinaremos con las restantes para elegir la segunda más importante. Así repetiremos hasta quedarnos con las 6 más importantes y eliminar las 4 peores

EJERCICIO 11: Mira el vídeo SAA_U01_set_de_datos_train_validación_test.mp4 y responde a estas preguntas:
a) ¿Porqué no es buena idea utilizar todos los datos para entrenar al modelo? Si lo hacemos y su comportamiento con estos datos es bueno, por ejemplo acierta el 96% de las veces ¿No es suficiente para decir que es un buen modelo? ¿O hay algo que no sabemos aún para darlo por bueno?

Porque podría aprender muy bien los datos de entrenamiento pero si después tiene que aplicarlos a nuevos datos o a datos desconocidos puede fallar. Se le llama overfitting

b) ¿Porqué nos puede interesar hacer 3 divisiones de los datos (train+validación+test) en ves de dos (train+test)?

Porque así podemos ajustar y evaluar el modelo de una mejor manera y más completa

c) ¿Qué porcentaje se suelen usar para cada partición de datos?

Se suele usar el porcentaje 70 % para entrenamiento, 15 % para validación y el 15 % restante para test, más o menos esos porcentajes

d) Si no tenemos muchos datos, en vez de hacer la partición en 3 datasets (train+validación+prueba) ¿Qué otros métodos se pueden utilizar?

Podemos utilizar validación cruzada croos-validation o validación cruzada leave-one-out

EJERCICIO 12: Vamos a trabajar con los datos ausentes
a) Define este dataframe que construiremos en código

```
# -*- coding: utf-8 -*-
import pandas as pd
from io import StringIO
import sys
csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''
# Si estás usando 2.7, necesitas convertirlo a string unicode
if (sys.version_info < (3, 0)):
    csv_data = csv_data.unicode(csv_data)
df = pd.read_csv(StringIO(csv_data))
print(df)
```

Este dataframe contiene columnas A B C D con valores numéricos. Contiene algunos valores vacíos que son rellenados con NaN

b) En la siguiente imagen aparecen unas sentencias. Imprime el resultado de ejecutarlas y asocia la etiqueta que tiene cada una como un comentario a la derecha, con la tarea que realizan:
• Cuenta el número de valores ausentes que hay en cada columna / característica
• Devuelve los arrays Numpy que contienen los datos del dataframe
• Borra las filas que contienen algún valor ausente
• Borra las columnas que contienen algún valor ausente
• Borra todas las filas donde todos los valores estén ausentes
• Borra las filas que tengan 3 o menos valores porque el resto están ausentes
• Borra una o varias columnas cuyo nombre se indica, si tienen valores ausentes

```
df.dropna(subset=['C']) # 1
df.dropna(how='all')    # 2
df.values               # 3
df.dropna(axis=0)       # 4
df.isnull().sum()       # 5
df.dropna(axis=1)       # 6
df.dropna(thresh=4)     # 7
```

En la siguientes sentencias tenemos estas etiquetas asociadas

df.dropna(subset=['C']) #Borra una o varias columnas cuyo nombre se indica, si tienen valores ausentes
df.dropna(how='all') #Borra todas las filas donde todos los valores estén ausentes
df.values #Devuelve los arrays Numpy que contienen los datos del dataframe
df.dropna(axis=0) #Borra las filas que contienen algún valor ausente
df.isnull().sum() #Cuenta el número de valores ausentes que hay en cada columna / característica
df.dropna(axis=1) #Borra las columnas que contienen algún valor ausente
df.dropna(thresh=4) #Borra las filas que tengan 3 o menos valores porque el resto están ausentes

c) Ahora en el mismo dataset, vamos a imputar la media de las columnas a los valores ausentes. Esa sentencia que usa un método de Pandas lo hace.
Busca y ordena las siguientes sentencias para que realicen algo equivalente

```
print("Datos transformados\n", df.fillna(df.mean()))
```

Este es mi resultado

```
     A     B     C    D
0   1.0   2.0   3.0  4.0
1   5.0   6.0   NaN  8.0
2  10.0  11.0  12.0  NaN
Datos transformados
       A     B     C    D
0   1.0   2.0   3.0  4.0
1   5.0   6.0   7.5  8.0
2  10.0  11.0  12.0  6.0
```

Busca y ordena las siguientes sentencias para que realicen algo equivalente

```
print("Datos transformados\n", x)
from sklearn.impute import SimpleImputer
import numpy as np
si = SimpleImputer(missing_values=np.nan, strategy='mean')
x = si.transform(df.values)
si = si.fit(df.values)
print("Datos originales\n", df.values)
```

```
si = SimpleImputer(missing_values=np.nan, strategy='mean')
si = si.fit(df.values)
x = si.transform(df.values)
print("Datos transformados\n", x)
print("Datos originales\n", df.values)
```

Aquí está el resultado ordenado

EJERCICIO 13: Transformar datos nominales en numéricos

a) Define este dataset

```
import pandas as pd
df = pd.DataFrame([['verde', 'M',  10.1, 'clase2'],
                   ['rojo',  'L',  13.5, 'clase1'],
                   ['azul',  'XL', 15.3, 'clase2']])
df.columns = ['color', 'talla', 'precio', 'class']
```

```
   color talla precio   class
0  verde     M   10.1  clase2
1   rojo     L   13.5  clase1
2   azul    XL   15.3  clase2
```

b) Codifica la columna "class" para que tenga valores numéricos 0,1 para sustituir los valores 'clase1' y 'clase2' utilizando un diccionario que defina la transformación y otro que describa la transformación inversa. Imprime los diccionarios:

```
mapeo = {label: idx for idx, label in enumerate(np.unique(df['class']))}
mapeo_inverso = {v: k for k, v in mapeo.items()}
```

```
df['class'] = df['class'].map(mapeo)
```

La última línea es la que aplica la transformación. Aplica la transformación, imprime el dataframe y deshaz la transformación y vuelve a imprimir el dataframe para comprobar que ha ido todo bien

c) Ahora vamos a realizar lo mismo pero usando un objeto LabelEncoder de la clase ScikitLearn. Define una transformación, la aplicas y muestra la columna modificada. Imprime el dataframe y verás que no ha cambiado, te falta un paso, debes sustituir la columna por sus nuevos valores. Hazlo y vuelve a imprimir el dataframe transformado. Ahora vuelve a deshacer la transformación y dejas al dataframe como al principio

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(df['class'].values)
```

Debes generar una salida similar a esta:

```
Datos transformados
    color talla  precio  class
0  verde     M    10.1      1
1   rojo     L    13.5      0
2   azul    XL    15.3      1
Datos originales
    color talla  precio   class
0  verde     M    10.1  clase2
1   rojo     L    13.5  clase1
2   azul    XL    15.3  clase2
```

```python
#PARTE A
#Creación de DataFrame
df = pd.DataFrame([['verde', 'M', 10.1, 'clase2'],
                   ['rojo', 'L', 13.5, 'clase1'],
                   ['azul', 'XL', 15.3, 'clase2']])
df.columns = ['color', 'talla', 'precio', 'class']

#Mostrar el Dataframe original
print("Dataframe original:")
print(df)
```

```
Dataframe original:
   color talla  precio   class
0  verde     M    10.1  clase2
1   rojo     L    13.5  clase1
2   azul    XL    15.3  clase2
```

```python
#PARTE B
#Crear el mapeo de valores
mapeo = {label: idx for idx, label in enumerate(np.unique(df['class']))}
mapeo_inverso = {v: k for k, v in mapeo.items()}

#Aplicar el mapeo
df['class'] = df['class'].map(mapeo)
print("\nDataFrame con 'class' transformada:")
print(df)

#Revertir la transformación
df['class'] = df['class'].map(mapeo_inverso)
print("\nDataFrame con 'class' revertida:")
print(df)
```

```
DataFrame con 'class' transformada:
   color talla  precio  class
0  verde     M    10.1      1
1   rojo     L    13.5      0
2   azul    XL    15.3      1

DataFrame con 'class' revertida:
   color talla  precio   class
0  verde     M    10.1  clase2
1   rojo     L    13.5  clase1
2   azul    XL    15.3  clase2
```

```python
#PARTE C
from sklearn.preprocessing import LabelEncoder

#Crear el objeto LabelEncoder
le = LabelEncoder()

#Aplicar el LabelEncoder y mostrar la columna transformada
df['class'] = le.fit_transform(df['class'])
print("\nDataFrame con 'class' transformada usando LabelEncoder:")
print(df)

#Revertir la transformación
df['class'] = le.inverse_transform(df['class'])
print("\nDataFrame con 'class' revertida usando LabelEncoder:")
print(df)
```

```
DataFrame con 'class' transformada usando LabelEncoder:
   color talla  precio  class
0  verde     M    10.1      1
1   rojo     L    13.5      0
2   azul    XL    15.3      1

DataFrame con 'class' revertida usando LabelEncoder:
   color talla  precio   class
0  verde     M    10.1  clase2
1   rojo     L    13.5  clase1
2   azul    XL    15.3  clase2
```

EJERCICIO 14: Transformar columnas nominales con más de 2 valores en numéricos usando onehot-Encoded.

a) Completa las partes subrayadas de las sentencias de este código que define nuevos datos X seleccionando las columnas color, talla y precio y codifica el color con un LabelEncoder como en el ejercicio anterior y los imprime. X = df[['color', 'talla', 'prec io']].values Sistemas de Aprendizaje Automático IES Serra Perenxisa U01:INTRODUCCIÓN Página 67 / 68 le = _____ # Instancia el objeto LabelEncoder _____ # Imprime los datos originales X[:, 0] = _____ # Transforma la columna print("X transformado:\n", X) # Imprime los datos transformados

```python
#Parte a)

#Ejemplo de Dataframe
data = {
    'color': ['rojo', 'verde', 'azul', 'rojo', 'verde'],
    'talla': ['S', 'M', 'L', 'XL', 'S'],
    'precio': [10, 15, 20, 25, 30]
}
df = pd.DataFrame(data)

#Seleccionamos las columnas relevantes
X = df[['color', 'talla', 'precio']].values

#Instancia del objeto LabelEncoder
le = LabelEncoder()

#Aplicamos el LabelEncoder a la columna de color
X[:, 0] = le.fit_transform(X[:, 0])

#Datos transformados
print("Datos originales:\n", df)
print("X transformado:\n", X)
```

```
Datos originales
    color talla  precio
0   rojo     S      10
1  verde     M      15
2   azul     L      20
3   rojo    XL      25
4  verde     S      30
X transformado
 [[1 'S' 10]
 [2 'M' 15]
 [0 'L' 20]
 [1 'XL' 25]
 [2 'S' 30]]
```

b) Haz lo mismo con un OneHotEncoder e imprime los datos transformados.

_____ _____ # objeto OneHotEncoder _____ # Imprime los datos originales _____ # Transforma la columna print("X transformado:\n", X) # Imprime los datos transformados

```python
#Parte b)
from sklearn.preprocessing import OneHotEncoder

#Instanciamos el objeto OneHotEncoder
ohe = OneHotEncoder()

# Aplica el OneHotEncoder a la columna de color (index 0)
X_color_encoded = ohe.fit_transform(X[:, 0].reshape(-1, 1)).toarray()

#Concatenamos la transformación con las demás columnas
X_transformed = pd.concat([
    pd.DataFrame(X_color_encoded, columns=ohe.get_feature_names_out(['color'])),
    pd.DataFrame(X[:, 1:], columns=['talla', 'precio'])
], axis=1)

#Imprimir los datos transformados
print("Datos transformados con OneHotEncoder\n", X_transformed)
```

```
Datos transformados con OneHotEncoder
    color_0  color_1  color_2 talla precio
0       0.0      1.0      0.0     S     10
1       0.0      0.0      1.0     M     15
2       1.0      0.0      0.0     L     20
3       0.0      1.0      0.0    XL     25
4       0.0      0.0      1.0     S     30
```

c) Utiliza get_dummies([‘c1’, …]) Para obtener un nuevo dataframe añadiendo nuevas columnas para aplicar one-hot-encoder a las que tengan valores categóricos. Aplica a las columnas color, talla y precio

```
#Parte c)
#Con get_dummies aplicado a las columnas color, talla y precio
df_dummies = pd.get_dummies(df, columns=['color', 'talla'])

print("DataFrame con dummies\n", df_dummies)
```

```
DataFrame con dummies
   precio  color_azul  color_rojo  ...  talla_M  talla_S  talla_XL
0      10       False        True  ...    False     True     False
1      15       False       False  ...     True    False     False
2      20        True       False  ...    False    False     False
3      25       False        True  ...    False    False      True
4      30       False       False  ...    False     True     False
```

d) Repite el apartado c), pero ahora indica que se elimine la primera columna dummy porque realmente se deduce que si no hay valores ausentes en el resto aparecerá valores todos a 0. Usa el parámetro drop_first = True

```
#Parte d)
#Con get_dummies con drop_first=True (eliminando la primera columna)
df_dummies_reduced = pd.get_dummies(df, columns=['color', 'talla'], drop_first=True)

print("DataFrame con dummies drop_first=True\n", df_dummies_reduced)
```

```
DataFrame con dummies drop_first=True
   precio  color_rojo  color_verde  talla_M  talla_S  talla_XL
0      10        True        False    False     True     False
1      15       False         True     True    False     False
2      20       False        False    False    False     False
3      25        True        False    False    False      True
4      30       False         True    False     True     False
```

EJERCICIO 15: Haz un programa python que pregunte por los valores a y b y dibuje la función f(x)=ax2+b en el intervalo [-5, 5] con 100 puntos. Rellena un array de 100 valores x entre [-5,5] y otro fx y dibuja

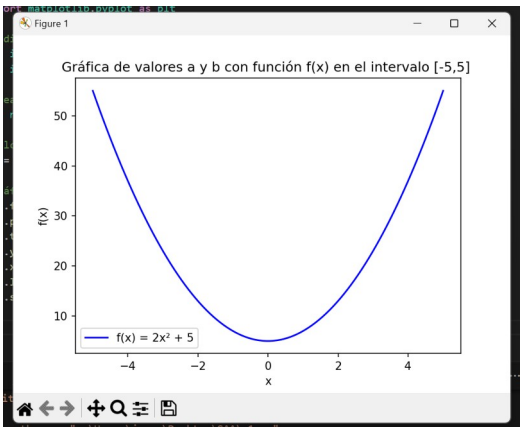
```python
import numpy as np
import matplotlib.pyplot as plt

#Pedir al usuario los valores
a = int(input("Dime el valor de a: "))
b = int(input("Dime el valor de b: "))

#Crear intervalo (array creo)
x = np.linspace(-5, 5, 100)

#Calcular valores de f(x) = ax^2+b (función)
fx = a * x**2 + b

#Gráfico
plt.figure(figsize=(10,8))
plt.plot(x, fx, label=f"f(x) = {a}x² + {b}", color='blue')
plt.title("Gráfica de valores a y b con función f(x) en el intervalo [-5,5] ")
plt.ylabel("f(x)")
plt.xlabel("x")
plt.legend()
plt.show()
```



EJERCICIO 16: Pregunta por el punto de corte (w0) y la pendiente (w1) de una línea recta y haz que el programa la dibuje en el intervalo [-5,5] sabiendo que su ecuación es y = w0 + w1 x

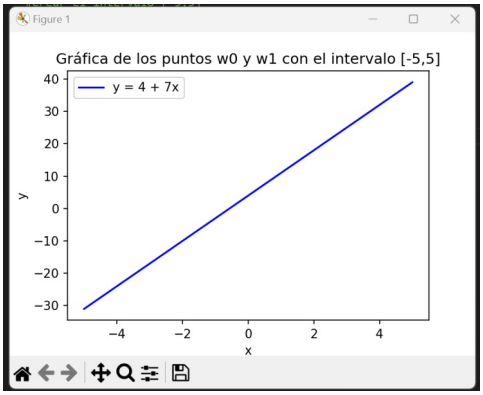
```python
import matplotlib.pyplot as plt
import numpy as np


#Preguntar al usuario
w0 = float(input("Dime el punto w0 "))
w1 = float(input("Dime el punto w1 "))

#Crear el intervalo [-5,5]
x = np.linspace(-5, 5, 100)

#Calcular valores de y con la ecuación y = w0 + w1 x
y = w0 + w1 * x

#Gráfico
plt.figure(figsize=(10,8))
plt.plot(x, y, label=f"y = {w0} + {w1}x", colors='blue')
plt.title("Gráfica de los puntos w0 y w1 con el intervalo [-5,5]")
plt.ylabel('y')
plt.xlabel('x')
plt.legend()
plt.show()
```



EJERCICIO 17: Imagina que tenemos estos datos con el peso y la altura de 4 personas: datos=[[3, 0.4], [8, 0.6], [18, 1.2], [27, 1.4]]
a) Crea un dataframe con los datos
b) Dibuja un sccaterplot() con círculos rojos dejando el peso en el eje X y la altura en el eje Y.
c) Pregunta por el punto de corte (w0) y la pendiente de una recta (w1) que intente acercarse a estos 4 puntos lo máximo posible
d) Dibuja la recta en el gráfico con color azul
e) Calcula e imprime el cuadrado de la distancia de cada punto de datos (cuando x=peso) con el valor de la recta en ese punto
f) Calcula la media de la suma de los cuadrados de esas diferencias
g) Si usamos ese valor como una medida del error que cometemos al intentar que la recta represente a los puntos, es como una función de coste. Intenta crear varias rectas hasta que bajes ese error de 0.5
h) Utiliza esta fórmula tratando a los datos como vectores y matrices para calcular los parámetros W0 y W1 de la mejor recta. Debes crear una columna a unos en los datos X y añadirle la columna peso. Prueba tu programa con esa recta: |1 3| |0.4| X = |1 8| y = |0.6| W = (XTX)-1 XTy |1 18| |1.2| |1 27| |1.4|

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#a) Datos
datos =  [[3,0.4], [8, 0.6], [18, 1.2], [27, 1.4]]
df = pd.DataFrame(datos, columns=['Peso', 'Altura'])

#b) Dibujar un scatterplot()
plt.figure(figsize=(10,8))
plt.scatter(df['Peso'], df['Altura'], color='blue', label='Datos')
plt.ylabel('Altura')
plt.xlabel('Peso')
plt.title('Scatterplot de Peso y Altura')
plt.legend()
plt.show()


#c) Punto de corte y pendiente de una recta
w0 = float(input("Dime el punto de corte w0: "))
w1 = float(input("Dime la pendiente de una recta: "))

#d) Dibujo de la recta en el gráfico con color azul
x_vals = np.linspace(min(df['Peso']), max(df['Peso']), 100)  #Peso
y_vals = w0 + w1 * x_vals  #Recta
plt.plot(x_vals, y_vals, color='blue', label=f"Recta: y = {w0} + {w1}x")
plt.legend()
plt.show()

#e) Calculo del cuadrado de la distancia de cada punto de datos (cuando x=peso)
distanciapuntos = (df['Altura'] - (w0 + w1 * df['Peso']))**2
print(distanciapuntos)

#f) Cálculo de la media de la suma de los cuadrados de esas diferencias
media_suma = np.mean(distanciapuntos)
print(f"\nMedia_suma: {media_suma}") #acaba en error

#g) Creación de varias rectas para bajar el error de 0.5
while media_suma > 0.5:
    print("El error cuadrado medio es mayor a 0.5, vuelve a intentarlo")
    w0 = float(input("Dime un nuevo punto de corte: "))
    w1 = float(input("Dime una nueva pendiente: "))

    #Recalcular las distancias
    distanciapuntos = (df['Altura'] - (w0 + w1 * df['Peso']))**2
    media_suma = np.mean(distanciapuntos)
    print(f"\nNueva Media_suma: {media_suma}") #del error de f

print("\n¡Error cuadrado medio reducido por debajo de 0.5!")


#h) Fórmula con vectores y matrices
#Crear la matriz X y el vector y
X = np.hstack([np.ones((len(df), 1)), df['Peso'].values.reshape(-1, 1)])  # Matriz X con columna de unos y pesos
y = df['Altura'].values.reshape(-1, 1)  # Vector y con alturas

#Calcular los parámetros W = (X^T X)^-1 X^T y
W = np.linalg.inv(X.T @ X) @ X.T @ y
w0_opt, w1_opt = W.flatten()

print("\nParámetros óptimos calculados:")
print(f"w0: {w0_opt:.4f}, w1: {w1_opt:.4f}")

#Recta en scatterplot
plt.figure(figsize=(10,8))
plt.scatter(df['Peso'], df['Altura'], color='red', label='Datos (Peso vs Altura)')
plt.plot(x_vals, w0_opt + w1_opt * x_vals, color='green', label=f"Recta óptima: y = {w0_opt:.2f} + {w1_opt:.2f}x")
plt.ylabel("Altura")
plt.xlabel("Peso")
plt.title("Recta")
```

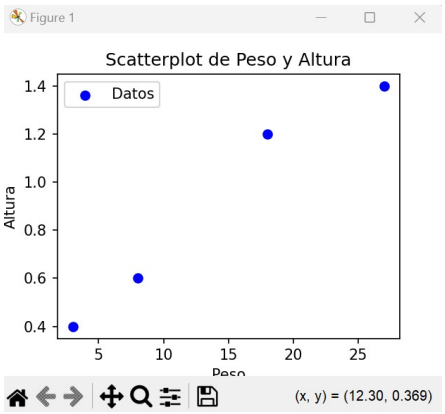


Figure 1
Scatterplot de Peso y Altura
Datos
Altura
Peso
1.4
1.2
1.0
0.8
0.6
0.4
5
10
15
20
25
(x, y) = (12.30, 0.369)

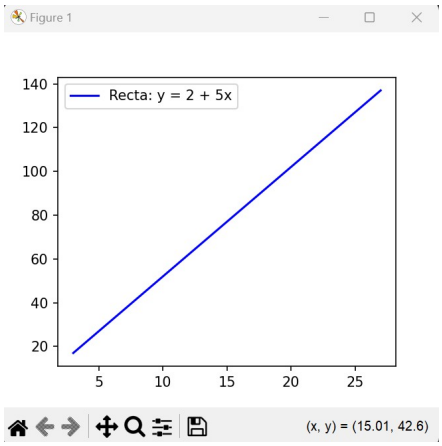


Figure 1
Recta: y = 2 + 5x
140
120
100
80
60
40
20
5
10
15
20
25
(x, y) = (15.01, 42.6)

EJERCICIO 18: Utiliza los datos del fichero mpg.csv para realizar las siguientes operaciones:
a) Crear un dataframe a partir del fichero
b) Eliminar la columna nombre
c) Hacer una versión escalada en [0,1] de los datos con código propio
d) Haz lo mismo con el objeto MinMaxScaler del paquete sklearn.preprocessing

```python
import pandas as pd

#Cargar archivo csv en DataFrame
datos = pd.read_csv('mpg.csv')

#Primeras filas
print(datos.head())

#Eliminación de la columna nombre
datos = datos.drop(columns=['nombre'])

#Mostrar las primeras filas después de la eliminación
print(datos.head())

#Versión escalada en 0,1 de los datos con código propio
#Crear una copia del DataFrame para los datos escalados
datos_scaled = datos.copy()

#Escalar los datos numéricos
numerical_columns = datos_scaled.select_dtypes(include=['float64', 'int64']).columns
datos_scaled[numerical_columns] = datos_scaled[numerical_columns].apply(lambda x: (x - x.min()) / (x.max() - x.min()))

#Mostrar las primeras filas de los datos escalados
print(datos_scaled.head())

#Usar MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

#Inicializar el MinMaxScaler
scaler = MinMaxScaler()

#Aplicar el escalado
datos_scaled_sklearn = datos.copy()
datos_scaled_sklearn[numerical_columns] = scaler.fit_transform(datos[numerical_columns])

#Verificar la transformación
datos_scaled_sklearn.head()
```

```
[Running] python -u "c:\Users\juan_\Desktop\SAA\UD01-Ejercicios\Ejercicio18.py"
    mpg  cilindros  ...  origen                       nombre
0  18.0          8  ...       1  chevrolet chevelle malibu
1  15.0          8  ...       1          buick skylark 320
2  18.0          8  ...       1         plymouth satellite
3  16.0          8  ...       1              amc rebel sst
4  17.0          8  ...       1                ford torino

[5 rows x 9 columns]
    mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0  18.0          8           307.0  ...         12.0           70       1
1  15.0          8           350.0  ...         11.5           70       1
2  18.0          8           318.0  ...         11.0           70       1
3  16.0          8           304.0  ...         12.0           70       1
4  17.0          8           302.0  ...         10.5           70       1

[5 rows x 8 columns]
        mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0  0.239362        1.0        0.617571  ...     0.238095          0.0     0.0
1  0.159574        1.0        0.728682  ...     0.208333          0.0     0.0
2  0.239362        1.0        0.645995  ...     0.178571          0.0     0.0
3  0.186170        1.0        0.609819  ...     0.238095          0.0     0.0
4  0.212766        1.0        0.604651  ...     0.148810          0.0     0.0

[5 rows x 8 columns]
```

EJERCICIO 19: Utiliza los datos del fichero mpg.csv para realizar las siguientes operaciones :
a) Crear un DataFrame eliminando la columna nombre
b) Hacer una versión escalada normalizando los datos de las columnas con tu propio código.
c) Haz lo mismo con el objeto StandardScaler del paquete sklearn.preprocessing
d) Encontrar outliers en las columnas con el método de la mediana con tu propio código
e) Encontrar outliers en las columnas con los z-scores con tu propio código
f) Detectar outliers dibujando boxplots de las columnas

```python
import pandas as pd
import numpy as np

#a) Cargar archivo csv en DataFrame
datos = pd.read_csv('mpg.csv')

#Eliminar la columna 'nombre'
datos_sin_nombre = datos.drop(columns=['nombre'])

#Primeras filas
print(datos_sin_nombre.head())

#b) Versión escalada normalizando los datos de las columnas
#Crear una copia del DataFrame para los datos escalados
datos_scaled = datos_sin_nombre.copy()

#Escalado de los datos numéricos
numerical_columns = datos_scaled.select_dtypes(include=['float64', 'int64']).columns
datos_scaled[numerical_columns] = datos_scaled[numerical_columns].apply(lambda x: (x - x.min()) / (x.max() - x.min()))

#Primeras filas
print(datos_scaled.head())


#c) Lo mismo con StandarScaler
from sklearn.preprocessing import StandardScaler

#Crear una copia del DataFrame para los datos escalados con StandardScaler
scaler = StandardScaler()
datos_scaled_sklearn = datos_sin_nombre.copy()

#Aplicar el escalado a las columnas numéricas
datos_scaled_sklearn[numerical_columns] = scaler.fit_transform(datos_scaled_sklearn[numerical_columns])

#Mostrar las primeras filas de los datos escalados con StandardScaler
print(datos_scaled_sklearn.head())

#d) Encontrar outliers en las columnas con el método de la mediana
#Función para detectar outliers usando la mediana y el IQR
def detectar_outliers_mediana(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    outliers = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
    return outliers

#Detectar outliers en las columnas numéricas
outliers_mediana = detectar_outliers_mediana(datos_sin_nombre[numerical_columns])

#Mostrar los outliers detectados
print(outliers_mediana)

#e) Encontrar outliers en las columnas con los z-scores
def detectar_outliers_zscore(df):
    z_scores = np.abs((df - df.mean()) / df.std())
    outliers = z_scores > 3
    return outliers

#Detectar outliers con z-scores
outliers_zscore = detectar_outliers_zscore(datos_sin_nombre[numerical_columns])

#Mostrar los outliers detectados
print(outliers_zscore)


#f) Detectar outliers dibujando boxplots
import matplotlib.pyplot as plt
import seaborn as sns

#Dibujar boxplots de las columnas numéricas
plt.figure(figsize=(12, 6))
sns.boxplot(data=datos_sin_nombre[numerical_columns])
plt.title("Boxplots de las columnas numéricas")
plt.xticks(rotation=90)
plt.show()
```

```
[Done] exited with code=1 in 101.737 seconds

    mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0  18.0          8           307.0  ...         12.0           70       1
1  15.0          8           350.0  ...         11.5           70       1
2  18.0          8           318.0  ...         11.0           70       1
3  16.0          8           304.0  ...         12.0           70       1
4  17.0          8           302.0  ...         10.5           70       1

[5 rows x 8 columns]
        mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0  0.239362        1.0        0.617571  ...     0.238095          0.0     0.0
1  0.159574        1.0        0.728682  ...     0.208333          0.0     0.0
2  0.239362        1.0        0.645995  ...     0.178571          0.0     0.0
3  0.186170        1.0        0.609819  ...     0.238095          0.0     0.0
4  0.212766        1.0        0.604651  ...     0.148810          0.0     0.0

[5 rows x 8 columns]
        mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year    origen
0 -0.706439   1.498191        1.090604  ...    -1.295498    -1.627426 -0.715145
1 -1.090751   1.498191        1.503514  ...    -1.477038    -1.627426 -0.715145
2 -0.706439   1.498191        1.196232  ...    -1.658577    -1.627426 -0.715145
3 -0.962647   1.498191        1.061796  ...    -1.295498    -1.627426 -0.715145
4 -0.834543   1.498191        1.042591  ...    -1.840117    -1.627426 -0.715145

[5 rows x 8 columns]
      mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0   False      False           False  ...        False        False   False
1   False      False           False  ...        False        False   False
2   False      False           False  ...        False        False   False
3   False      False           False  ...        False        False   False
4   False      False           False  ...        False        False   False
..    ...        ...             ...  ...          ...          ...     ...
393 False      False           False  ...        False        False   False
394 False      False           False  ...         True        False   False
395 False      False           False  ...        False        False   False
396 False      False           False  ...        False        False   False
397 False      False           False  ...        False        False   False

[398 rows x 8 columns]
      mpg  cilindros  desplazamiento  ...  aceleracion  modelo_year  origen
0   False      False           False  ...        False        False   False
1   False      False           False  ...        False        False   False
2   False      False           False  ...        False        False   False
3   False      False           False  ...        False        False   False
4   False      False           False  ...        False        False   False
..    ...        ...             ...  ...          ...          ...     ...
393 False      False           False  ...        False        False   False
394 False      False           False  ...         True        False   False
395 False      False           False  ...        False        False   False
396 False      False           False  ...        False        False   False
397 False      False           False  ...        False        False   False

[398 rows x 8 columns]
```