4. Con el uso de librerías realiza en Python los mismos preprocesamiento del punto 3.

| **CÓDIGO COLAB (onehotencoder)** |
| --- |
| import pandas as pd  from sklearn.preprocessing import OneHotEncoder  # Leer el archivo CSV completo  data = pd.read\_csv("/content/Drive/MyDrive/datos/examen.csv")  # Instanciar el OneHotEncoder  preprocesamiento = OneHotEncoder()  # Seleccionar todas las columnas  # Aplicar OneHotEncoder a todas las columnas (si tienes categóricas y numéricas)  # Para este ejemplo, seleccionamos todas las columnas del archivo CSV  listaDatos = data.values.tolist()  # Ajustar el OneHotEncoder a los datos del CSV  preprocesamiento.fit(listaDatos)  # Ver las categorías detectadas por OneHotEncoder  print("Categorías detectadas por OneHotEncoder:")  print(preprocesamiento.categories\_)  # Transformar los datos a OneHotEncoding y convertir el resultado en una matriz  resultado = preprocesamiento.transform(listaDatos).toarray()  # Mostrar los resultados del OneHotEncoder aplicados a todas las filas del CSV  print("Resultado de la transformación OneHotEncoder:")  print(resultado)  # Si quieres ver un resumen o las primeras filas del resultado transformado  print("Primeras filas transformadas:")  print(resultado[:5]) |
| **CORRIDA DE PANTALLA** |
|  |

| **CÓDIGO DE COLAB (labelencoder)** |
| --- |
| import pandas as pd  from sklearn.preprocessing import LabelEncoder  # Leer el archivo CSV completo  data = pd.read\_csv("/content/Drive/MyDrive/datos/examen.csv")  # Verificar si se ha leído el archivo correctamente  print(f"El archivo tiene {data.shape[0]} filas y {data.shape[1]} columnas.")  # Revisar si hay columnas con valores faltantes  print("\nNúmero de valores faltantes por columna:")  print(data.isnull().sum())  # Inicializar LabelEncoder  label\_encoder = LabelEncoder()  # Aplicar LabelEncoder a todas las columnas que sean de tipo 'object' o 'category'  for columna in data.columns:  if data[columna].dtype == 'object' or data[columna].dtype == 'category':  # Llenar valores NaN con una cadena vacía o algún valor adecuado para no tener errores  data[columna] = data[columna].fillna('desconocido')  # Aplicar LabelEncoder y crear una nueva columna con el sufijo '\_encoded'  data[columna + '\_encoded'] = label\_encoder.fit\_transform(data[columna].astype(str))  # Imprimir los primeros valores de la codificación realizada  print(f"\nColumna: {columna}")  print(data[[columna, columna + '\_encoded']].head())  else:  print(f"La columna {columna} no es de tipo texto, se omite la codificación.")  # Mostrar las primeras filas del DataFrame con las nuevas columnas codificadas  print("\nDataFrame con las columnas codificadas:")  print(data.head(180))  # Mostrar el total de filas  print(f"Total de filas leídas: {data.shape[0]}") |
| **CORRIDA DE PANTALLA** |
|  |

| **CÓDIGO COLAB (discretizacion)** |
| --- |
| import pandas as pd  from sklearn.preprocessing import KBinsDiscretizer  # Leer el archivo CSV completo  try:  data = pd.read\_csv("/content/Drive/MyDrive/datos/examen.csv")  except FileNotFoundError:  print("Error: El archivo CSV no se encontró en la ruta especificada.")  raise  # Seleccionar solo las columnas numéricas del archivo  columnas\_numericas = data.select\_dtypes(include=['float64', 'int64'])  # Comprobar si hay columnas numéricas disponibles  if columnas\_numericas.empty:  print("Error: No hay columnas numéricas disponibles para discretizar.")  else:  # Eliminar filas con valores nulos en las columnas numéricas  columnas\_numericas = columnas\_numericas.dropna()  # Instanciar el KBinsDiscretizer  discretizer = KBinsDiscretizer(n\_bins=5, encode='ordinal', strategy='uniform')  # Ajustar y transformar las columnas numéricas con KBinsDiscretizer  resultado\_discretizado = discretizer.fit\_transform(columnas\_numericas)  # Crear un nuevo DataFrame con los resultados discretizados  df\_discretizado = pd.DataFrame(resultado\_discretizado, columns=columnas\_numericas.columns)  # Mostrar los resultados de la discretización  print("Datos discretizados:")  print(df\_discretizado)  # Si deseas ver solo las primeras 5 filas del resultado discretizado  print("Primeras 5 filas discretizadas:")  print(df\_discretizado.head()) |
| **CORRIDA DE CODIGO** |
|  |

| **CÓDIGO COLAB (normalización)** |
| --- |
| import pandas as pd  from sklearn.preprocessing import Normalizer  # Leer el archivo CSV completo  try:  data = pd.read\_csv("/content/Drive/MyDrive/datos/examen.csv")  except FileNotFoundError:  print("Error: El archivo CSV no se encontró en la ruta especificada.")  raise  # Seleccionar solo las columnas numéricas del archivo  columnas\_numericas = data.select\_dtypes(include=['float64', 'int64'])  # Comprobar si hay columnas numéricas disponibles  if columnas\_numericas.empty:  print("Error: No hay columnas numéricas disponibles para normalizar.")  else:  # Eliminar filas con valores nulos en las columnas numéricas  columnas\_numericas = columnas\_numericas.dropna()  # Instanciar el Normalizer  preprocesamiento = Normalizer(norm='l2')  # Ajustar y transformar las columnas numéricas con Normalizer  resultado\_normalizado = preprocesamiento.fit\_transform(columnas\_numericas)  # Crear un nuevo DataFrame con los resultados normalizados  df\_normalizado = pd.DataFrame(resultado\_normalizado, columns=columnas\_numericas.columns)  # Mostrar los resultados de la normalización  print("Datos normalizados:")  print(df\_normalizado)  # Si deseas ver solo las primeras 5 filas del resultado normalizado  print("Primeras 5 filas normalizadas:")  print(df\_normalizado.head()) |
| **CORRIDA DE PANTALLA** |
|  |