```python
# ==========================================
# EJERCICIO 1: Exploración y limpieza de datos con el dataset "Titanic"
# Dataset: Titanic - train.csv from Kaggle
# Objetivo: Cumplir exactamente los 6 puntos del enunciado
# ==========================================

# Paso 1: Importar bibliotecas necesarias
import pandas as pd # Para manejo de datos en forma de tablas
import numpy as np # Para operaciones numéricas

print("=== EJERCICIO 1: EXPLORACIÓN Y LIMPIEZA DE DATOS - TITANIC ===\n")

# PUNTO 1: Carga el archivo CSV con pandas
try:
df = pd.read_csv("../../Dataset/train.csv", sep=';')
print("PUNTO 1: Archivo train.csv cargado exitosamente con pandas")
# Si solo hay una columna, cambiar separador
if df.shape[1] == 1:
df = pd.read_csv("../../Dataset/train.csv", sep=',')
except FileNotFoundError:
print("Error: No se encontró el archivo train.csv")
print("Asegúrate de que el archivo esté en la carpeta Dataset/")
exit()

# Limpiar el dataset - eliminar columnas 'zero' si existen
important_cols = [col for col in df.columns if 'zero' not in col.lower()]
df = df[important_cols].copy()

print(f"Dataset cargado: {df.shape[0]:,} filas × {df.shape[1]} columnas")
print(f"Columnas disponibles: {list(df.columns)}")

# PUNTO 2: Muestra las primeras 10 filas con .head()
print(f"\nPUNTO 2: Primeras 10 filas del dataset:")
print("=" * 60)
print(df.head(10)) # ESPECÍFICAMENTE 10 filas como pide el enunciado

# PUNTO 3: Calcula cuántos valores faltantes hay por columna
print(f"\nPUNTO 3: Valores faltantes por columna:")
print("=" * 45)
valores_faltantes = df.isnull().sum()
print(valores_faltantes)

# Mostrar también en porcentaje para mejor comprensión
print(f"\nPorcentaje de valores faltantes:")
porcentaje_faltantes = (df.isnull().sum() / len(df)) * 100
for columna, porcentaje in porcentaje_faltantes.items():
if porcentaje > 0:
print(f" {columna}: {porcentaje:.2f}%")
else:
print(f" {columna}: 0.00% (sin valores faltantes)")
```

```
# Crear copia del DataFrame para trabajar
df_limpio = df.copy()

# PUNTO 4: Llena los valores nulos de la columna Age con la mediana
print(f"\nPUNTO 4: Llenando valores nulos de Age con la mediana:")
print("=" * 55)

# Buscar la columna Age (puede tener diferentes nombres)
age_columns = [col for col in df_limpio.columns if 'age' in col.lower()]

if age_columns:
age_col = age_columns[0]
valores_nulos_age_antes = df_limpio[age_col].isnull().sum()
print(f"Columna encontrada: '{age_col}'")
print(f"Valores nulos en {age_col} antes: {valores_nulos_age_antes}")
if valores_nulos_age_antes > 0:
mediana_age = df_limpio[age_col].median()
print(f"Mediana calculada: {mediana_age:.2f}")
df_limpio[age_col] = df_limpio[age_col].fillna(mediana_age)
valores_nulos_age_despues = df_limpio[age_col].isnull().sum()
print(f"Valores nulos en {age_col} después: {valores_nulos_age_despues}")
print(f"Se llenaron {valores_nulos_age_antes} valores faltantes con la mediana")
else:
print(f"La columna {age_col} no tiene valores nulos (ya está limpia)")
print(f"Mediana de {age_col}: {df_limpio[age_col].median():.2f}")
else:
print("No se encontró columna Age en el dataset")

# PUNTO 5: Reemplaza la columna Sex por variables dummy (pd.get_dummies)
print(f"\nPUNTO 5: Reemplazando columna Sex por variables dummy:")
print("=" * 58)

# Buscar la columna Sex
sex_columns = [col for col in df_limpio.columns if 'sex' in col.lower()]

if sex_columns:
sex_col = sex_columns[0]
print(f"Columna encontrada: '{sex_col}'")
print(f"Valores únicos antes: {df_limpio[sex_col].unique()}")
print(f"Tipo de datos: {df_limpio[sex_col].dtype}")
# FORZAR el uso de pd.get_dummies como requiere el enunciado
if df_limpio[sex_col].dtype != 'object':
# Si es numérico, convertir a texto primero para poder usar get_dummies
print("Convirtiendo valores numéricos a texto para usar pd.get_dummies...")
df_limpio[sex_col] = df_limpio[sex_col].map({0: 'female', 1: 'male'})
print(f"Valores después de conversión: {df_limpio[sex_col].unique()}")
# Usar pd.get_dummies como específicamente requiere el enunciado
print("Aplicando pd.get_dummies...")
df_limpio = pd.get_dummies(df_limpio, columns=[sex_col], drop_first=True)
```

```
# Mostrar las nuevas columnas creadas
nuevas_columnas_sex = [col for col in df_limpio.columns if sex_col in col]
print(f"Nuevas columnas creadas: {nuevas_columnas_sex}")
print(f"Columnas totales después de get_dummies: {list(df_limpio.columns)}")
else:
print("No se encontró columna Sex en el dataset")

# Verificar que no hay valores faltantes después de la limpieza
print(f"\nVerificación final de valores faltantes:")
valores_faltantes_final = df_limpio.isnull().sum()
print(valores_faltantes_final)

# PUNTO 6: Guarda el nuevo DataFrame limpio en un archivo titanic_limpio.csv
print(f"\nPUNTO 6: Guardando DataFrame limpio en titanic_limpio.csv:")
print("=" * 62)

try:
df_limpio.to_csv("titanic_limpio.csv", index=False)
print(f"Archivo guardado exitosamente: 'titanic_limpio.csv'")
print(f"Tamaño del archivo: {df_limpio.shape[0]:,} filas × {df_limpio.shape[1]} columnas")
print(f"Columnas en el archivo limpio:")
for i, col in enumerate(df_limpio.columns, 1):
print(f" {i:2d}. {col}")
except Exception as e:
print(f"Error al guardar el archivo: {e}")

# Mostrar muestra del DataFrame limpio final
print(f"\nMuestra del DataFrame limpio final:")
print("=" * 40)
print(df_limpio.head())


# Resumen de cambios realizados
print(f"\nRESUMEN DE CAMBIOS REALIZADOS:")
print("-" * 35)
print(f"• Dataset original: {df.shape[0]:,} filas × {df.shape[1]} columnas")
print(f"• Dataset limpio: {df_limpio.shape[0]:,} filas × {df_limpio.shape[1]} columnas")
if age_columns and age_col in df.columns:
age_antes = df[age_col].isnull().sum()
age_despues = df_limpio[age_col].isnull().sum() if age_col in df_limpio.columns else 0
print(f"• Valores nulos en Age: {age_antes} → {age_despues}")
if sex_columns:
print(f"• Columna Sex convertida a variables dummy usando pd.get_dummies")
print(f"• Archivo limpio guardado: titanic_limpio.csv")
```

```
# ============================================
# EJERCICIO 2: Visualización de datos con el dataset "Netflix Movies and TV Shows"
# Dataset: Netflix dataset de Kaggle
# Objetivo: Cumplir exactamente los 5 puntos del enunciado
# ============================================

# Paso 1: Importar bibliotecas necesarias
import pandas as pd # Para manejo de datos en forma de tablas
import numpy as np # Para operaciones numéricas
import matplotlib.pyplot as plt # Gráficos
import seaborn as sns # Visualizaciones
from collections import Counter # Para análisis de frecuencias

print("=== EJERCICIO 2: VISUALIZACIÓN DE DATOS - NETFLIX ===\n")

# PUNTO 1: Carga el archivo netflix_titles.csv
try:
df = pd.read_csv("../../Dataset/netflix_titles.csv")
print("PUNTO 1: Archivo netflix_titles.csv cargado exitosamente")
print(f"Dataset cargado: {df.shape[0]:,} filas × {df.shape[1]} columnas")
print(f"Columnas disponibles: {list(df.columns)}")
except FileNotFoundError:
print("Error: No se encontró el archivo netflix_titles.csv")
print("Asegúrate de que el archivo esté en la carpeta Dataset/")
exit()

print("\nVista previa de los datos:")
print(df.head())

# PUNTO 2: Muestra cuántos títulos son películas y cuántos son programas de TV
print(f"\nPUNTO 2: Distribución de títulos por tipo:")
print("=" * 45)
tipo_contenido = df['type'].value_counts()
print("Conteo por tipo:")
print(tipo_contenido)

print(f"\nDistribución detallada:")
total_titulos = len(df)
for tipo, cantidad in tipo_contenido.items():
porcentaje = (cantidad / total_titulos) * 100
print(f" {tipo}: {cantidad:,} títulos ({porcentaje:.1f}%)")

# PUNTO 3: Crea un gráfico de barras que muestre cuántas producciones se lanzaron por año
print(f"\nPUNTO 3: Creando gráfico de barras por año de lanzamiento...")
print("=" * 60)

# Limpiar y preparar datos de año
df_clean = df.copy()
if 'release_year' in df_clean.columns:
df_clean['release_year'] = pd.to_numeric(df_clean['release_year'], errors='coerce')
```

```
df_clean = df_clean.dropna(subset=['release_year'])
# Contar producciones por año
producciones_por_año = df_clean['release_year'].value_counts().sort_index()
# Crear el gráfico de barras
plt.figure(figsize=(15, 6))
plt.bar(producciones_por_año.index, producciones_por_año.values, color='#E50914', alpha=0.7)
plt.title('Producciones de Netflix por Año de Lanzamiento', fontsize=16, fontweight='bold')
plt.xlabel('Año de Lanzamiento', fontsize=12)
plt.ylabel('Número de Producciones', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.savefig('netflix_producciones_por_año.png', dpi=300, bbox_inches='tight')
print("Gráfico de barras guardado como 'netflix_producciones_por_año.png'")
plt.close()
# Mostrar estadísticas del año
año_mas_producciones = producciones_por_año.idxmax()
max_producciones = producciones_por_año.max()
print(f"Año con más lanzamientos: {int(año_mas_producciones)} ({max_producciones}
producciones)")
else:
print("No se encontró columna 'release_year' en el dataset")

# PUNTO 4: Visualiza con un countplot de Seaborn cuántas producciones hay por país (top 10)
print(f"\nPUNTO 4: Creando countplot de países (top 10)...")
print("=" * 50)

if 'country' in df.columns:
# Preparar datos de países (algunos títulos tienen múltiples países separados por comas)
paises_lista = []
for paises in df['country'].dropna():
# Dividir por comas y limpiar espacios
paises_separados = [pais.strip() for pais in str(paises).split(',')]
paises_lista.extend(paises_separados)
# Contar y obtener top 10
paises_contador = Counter(paises_lista)
top10_paises = dict(paises_contador.most_common(10))
print(f"TOP 10 PAÍSES CON MÁS PRODUCCIONES:")
for i, (pais, cantidad) in enumerate(top10_paises.items(), 1):
print(f"{i:2d}. {pais}: {cantidad} producciones")
# Crear DataFrame para countplot
df_paises = pd.DataFrame({'country': paises_lista})
# Crear countplot con Seaborn
plt.figure(figsize=(12, 6))
paises_names = list(top10_paises.keys())
sns.countplot(data=df_paises[df_paises['country'].isin(paises_names)],
x='country',
order=paises_names,
palette='viridis')
plt.title('Top 10 Países con Más Producciones en Netflix', fontsize=16, fontweight='bold')
```

```
plt.xlabel('País', fontsize=12)
plt.ylabel('Número de Producciones', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('netflix_top10_paises_countplot.png', dpi=300, bbox_inches='tight')
print("Countplot de países guardado como 'netflix_top10_paises_countplot.png'")
plt.close()
else:
print("No se encontró columna 'country' en el dataset")

# PUNTO 5: Genera un gráfico de torta de los 5 directores más frecuentes
print(f"\nPUNTO 5: Creando gráfico de torta de directores (top 5)...")
print("=" * 58)

if 'director' in df.columns:
# Preparar datos de directores
directores_lista = []
for directores in df['director'].dropna():
# Algunos títulos tienen múltiples directores
directores_separados = [director.strip() for director in str(directores).split(',')]
directores_lista.extend(directores_separados)
# Contar y obtener top 5
directores_contador = Counter(directores_lista)
top5_directores = dict(directores_contador.most_common(5))
print(f"TOP 5 DIRECTORES MÁS FRECUENTES:")
for i, (director, cantidad) in enumerate(top5_directores.items(), 1):
print(f"{i}. {director}: {cantidad} producciones")
# Crear gráfico de torta
plt.figure(figsize=(10, 8))
colores = ['#E50914', '#F40612', '#FF6B6B', '#FF8E8E', '#FFB1B1']
wedges, texts, autotexts = plt.pie(top5_directores.values(),
labels=top5_directores.keys(),
autopct='%1.1f%%',
colors=colores,
startangle=90,
explode=(0.05, 0, 0, 0, 0)) # Resaltar el primero
plt.title('Top 5 Directores Más Frecuentes en Netflix', fontsize=16, fontweight='bold', pad=20)
# Mejorar la apariencia del texto
for autotext in autotexts:
autotext.set_color('white')
autotext.set_fontweight('bold')
plt.axis('equal') # Para que la torta sea circular
plt.tight_layout()
plt.savefig('netflix_top5_directores_torta.png', dpi=300, bbox_inches='tight')
print("Gráfico de torta guardado como 'netflix_top5_directores_torta.png'")
plt.close()
else:
print("No se encontró columna 'director' en el dataset")

# Estadísticas adicionales del dataset
```

```
print(f"\nESTADÍSTICAS ADICIONALES DEL DATASET:")
print("=" * 40)

# Información general
print(f"Total de títulos únicos: {df['title'].nunique():,}")
print(f"Total de directores únicos: {df['director'].nunique():,}")

if 'country' in df.columns:
paises_unicos = len(set(paises_lista)) if 'paises_lista' in locals() else df['country'].nunique()
print(f"Total de países representados: {paises_unicos}")

if 'release_year' in df.columns:
year_min = df['release_year'].min()
year_max = df['release_year'].max()
print(f"Rango de años: {int(year_min)} - {int(year_max)}")


# Resumen de archivos generados
print(f"\nARCHIVOS GENERADOS:")
print("- netflix_producciones_por_año.png (gráfico de barras)")
print("- netflix_top10_paises_countplot.png (countplot de países)")
print("- netflix_top5_directores_torta.png (gráfico de torta)")

print(f"\nEJERCICIO 2 COMPLETADO EXITOSAMENTE!")
```

```python
# ============================================
# EJERCICIO 3: Carga y preparación de datos de texto con el dataset "Fake News"
# Dataset: Fake.csv y True.csv
# Objetivo: Cumplir exactamente los 6 puntos del enunciado
# ============================================

# Paso 1: Importar bibliotecas necesarias
import pandas as pd # Para manejo de datos en forma de tablas
import numpy as np # Para operaciones numéricas

print("=== EJERCICIO 3: PROCESAMIENTO DE DATOS DE TEXTO - FAKE NEWS ===\n")

# PUNTO 1: Carga ambos archivos como dos DataFrames distintos
print("PUNTO 1: Cargando archivos Fake.csv y True.csv como DataFrames distintos:")
print("=" * 70)

# Cargar archivo de noticias falsas
try:
df_fake = pd.read_csv("../../Dataset/fake.csv")
print("Archivo 'fake.csv' cargado exitosamente")
print(f"Noticias falsas: {df_fake.shape[0]:,} registros × {df_fake.shape[1]} columnas")
print(f"Columnas en fake.csv: {list(df_fake.columns)}")
except FileNotFoundError:
try:
# Intentar con nombre en mayúscula
df_fake = pd.read_csv("../../Dataset/Fake.csv")
print("Archivo 'Fake.csv' cargado exitosamente")
print(f"Noticias falsas: {df_fake.shape[0]:,} registros × {df_fake.shape[1]} columnas")
print(f"Columnas en Fake.csv: {list(df_fake.columns)}")
except FileNotFoundError:
print("Error: No se encontró el archivo fake.csv o Fake.csv")
print("Asegúrate de que el archivo esté en la carpeta Dataset/")
exit()

# Cargar archivo de noticias verdaderas
try:
df_true = pd.read_csv("../../Dataset/true.csv")
print("Archivo 'true.csv' cargado exitosamente")
print(f"Noticias verdaderas: {df_true.shape[0]:,} registros × {df_true.shape[1]} columnas")
print(f"Columnas en true.csv: {list(df_true.columns)}")
except FileNotFoundError:
try:
# Intentar con nombre en mayúscula
df_true = pd.read_csv("../../Dataset/True.csv")
print("Archivo 'True.csv' cargado exitosamente")
print(f"Noticias verdaderas: {df_true.shape[0]:,} registros × {df_true.shape[1]} columnas")
print(f"Columnas en True.csv: {list(df_true.columns)}")
except FileNotFoundError:
print("Error: No se encontró el archivo true.csv o True.csv")
print("Asegúrate de que el archivo esté en la carpeta Dataset/")
```

```
exit()

# Mostrar muestra de ambos DataFrames
print(f"\nMuestra de noticias falsas (primeras 3 filas):")
print(df_fake.head(3))

print(f"\nMuestra de noticias verdaderas (primeras 3 filas):")
print(df_true.head(3))

# PUNTO 2: Agrega una columna label con valor 0 para noticias falsas y 1 para verdaderas
print(f"\nPUNTO 2: Agregando columna 'label':")
print("=" * 35)

# Crear copias para evitar modificar los originales
df_fake_labeled = df_fake.copy()
df_true_labeled = df_true.copy()

# Agregar columna label
df_fake_labeled['label'] = 0 # 0 = Noticia falsa
df_true_labeled['label'] = 1 # 1 = Noticia verdadera

print(f"Etiqueta agregada a noticias falsas: label = 0")
print(f"Etiqueta agregada a noticias verdaderas: label = 1")

# Verificar las etiquetas
print(f"\nVerificación de etiquetas:")
print(f"Noticias falsas con label=0: {(df_fake_labeled['label'] == 0).sum():,}")
print(f"Noticias verdaderas con label=1: {(df_true_labeled['label'] == 1).sum():,}")

print(f"\nColumnas después de agregar label:")
print(f"fake.csv: {list(df_fake_labeled.columns)}")
print(f"true.csv: {list(df_true_labeled.columns)}")

# PUNTO 3: Une ambos DataFrames en uno solo
print(f"\nPUNTO 3: Uniendo ambos DataFrames en uno solo:")
print("=" * 45)

# Concatenar los DataFrames
df_combined = pd.concat([df_fake_labeled, df_true_labeled], ignore_index=True)

print(f"DataFrames combinados exitosamente")
print(f"Total de registros: {df_combined.shape[0]:,}")
print(f"Total de columnas: {df_combined.shape[1]}")
print(f"Columnas en dataset combinado: {list(df_combined.columns)}")

# Verificar la distribución de etiquetas
distribucion_labels = df_combined['label'].value_counts().sort_index()
print(f"\nDistribución final de etiquetas:")
print(f"Noticias falsas (0): {distribucion_labels[0]:,}
({distribucion_labels[0]/len(df_combined)*100:.1f}%)")
```

```
print(f"Noticias verdaderas (1): {distribucion_labels[1]:,}
({distribucion_labels[1]/len(df_combined)*100:.1f}%)")

# PUNTO 4: Elimina las columnas subject y date, dejando solo text y label
print(f"\nPUNTO 4: Eliminando columnas subject y date, dejando solo text y label:")
print("=" * 75)

print(f"Columnas antes de eliminar: {list(df_combined.columns)}")

# Verificar qué columnas existen antes de eliminar
columnas_a_eliminar = []
if 'subject' in df_combined.columns:
columnas_a_eliminar.append('subject')
print(f"Columna 'subject' encontrada - será eliminada")
if 'date' in df_combined.columns:
columnas_a_eliminar.append('date')
print(f"Columna 'date' encontrada - será eliminada")

# Eliminar columnas especificadas
if columnas_a_eliminar:
df_processed = df_combined.drop(columns=columnas_a_eliminar)
print(f"Columnas eliminadas: {columnas_a_eliminar}")
else:
df_processed = df_combined.copy()
print(f"No se encontraron las columnas 'subject' o 'date'")

# Verificar si existen las columnas text y label
columnas_deseadas = ['text', 'label']
columnas_disponibles = [col for col in columnas_deseadas if col in df_processed.columns]

if len(columnas_disponibles) == 2:
df_final = df_processed[columnas_deseadas].copy()
print(f"Dataset final con columnas deseadas: {list(df_final.columns)}")
else:
print(f"Advertencia: No se encontraron todas las columnas esperadas")
print(f"Columnas disponibles: {list(df_processed.columns)}")
# Adaptar a las columnas que existan
if 'title' in df_processed.columns and 'text' not in df_processed.columns:
df_final = df_processed[['title', 'label']].copy()
df_final = df_final.rename(columns={'title': 'text'})
print(f"Usando 'title' como 'text'")
elif len(df_processed.columns) >= 2:
# Tomar las últimas dos columnas (asumiendo que una es text-like y otra es label)
text_col = [col for col in df_processed.columns if col != 'label'][0]
df_final = df_processed[[text_col, 'label']].copy()
if text_col != 'text':
df_final = df_final.rename(columns={text_col: 'text'})
print(f"Usando '{text_col}' como 'text'")
else:
df_final = df_processed.copy()
```

```
print(f"Columnas finales: {list(df_final.columns)}")
print(f"Tamaño final: {df_final.shape[0]:,} filas × {df_final.shape[1]} columnas")

# PUNTO 5: Verifica si hay valores nulos
print(f"\nPUNTO 5: Verificando valores nulos:")
print("=" * 35)

valores_nulos = df_final.isnull().sum()
print(f"Valores nulos por columna:")
total_nulos = 0
for columna, nulos in valores_nulos.items():
if nulos > 0:
porcentaje = (nulos / len(df_final)) * 100
print(f" {columna}: {nulos:,} valores nulos ({porcentaje:.2f}%)")
total_nulos += nulos
else:
print(f" {columna}: Sin valores nulos")

if total_nulos > 0:
print(f"\nTotal de valores nulos encontrados: {total_nulos:,}")
print("Eliminando filas con valores nulos...")
filas_antes = len(df_final)
df_final = df_final.dropna()
filas_despues = len(df_final)
print(f"Filas eliminadas: {filas_antes - filas_despues:,}")
print(f"Filas restantes: {filas_despues:,}")
else:
print(f"\nNo se encontraron valores nulos - dataset está completo")

# PUNTO 6: Guarda el nuevo dataset como noticias_procesadas.csv
print(f"\nPUNTO 6: Guardando dataset como noticias_procesadas.csv:")
print("=" * 55)

archivo_salida = "noticias_procesadas.csv"
try:
df_final.to_csv(archivo_salida, index=False)
print(f"Dataset guardado exitosamente como '{archivo_salida}'")
print(f"Ubicación: directorio actual")
print(f"Tamaño del archivo: {len(df_final):,} filas × {len(df_final.columns)} columnas")
# Información del archivo guardado
print(f"\nInformación del archivo guardado:")
print(f"Nombre: {archivo_salida}")
print(f"Columnas: {list(df_final.columns)}")
print(f"Registros totales: {len(df_final):,}")
if 'label' in df_final.columns:
dist_final = df_final['label'].value_counts().sort_index()
print(f"Distribución final:")
print(f" Noticias falsas (0): {dist_final[0]:,}")
print(f" Noticias verdaderas (1): {dist_final[1]:,}")
```

```
except Exception as e:
print(f"Error al guardar el archivo: {e}")

# Mostrar muestra del dataset final
print(f"\nMuestra del dataset procesado final:")
print("=" * 40)
for i, row in df_final.head(3).iterrows():
texto_col = 'text' if 'text' in df_final.columns else df_final.columns[0]
label_val = "VERDADERA" if row['label'] == 1 else "FALSA"
print(f"Noticia {i+1} ({label_val}):")
print(f" {row[texto_col][:100]}...")
print()


# Resumen del procesamiento realizado
print(f"\nRESUMEN DEL PROCESAMIENTO REALIZADO:")
print("=" * 40)
print(f"• Archivos originales procesados: fake.csv y true.csv")
print(f"• Dataset final: {len(df_final):,} noticias etiquetadas")
print(f"• Formato final: texto + etiqueta binaria")
print(f"• Archivo generado: {archivo_salida}")
print(f"• Dataset listo para análisis de texto o machine learning")

print(f"\nEJERCICIO 3 COMPLETADO EXITOSAMENTE!")
```