Guía de Laboratorio: Introducción a Pandas
Curso: Inteligencia Artificial

Docente: Carlos R. P. Tovar

1. Introducción

Pandas es una biblioteca de Python especializada en el manejo y análisis de datos. Proporciona estructuras de datos flexibles y eficientes que facilitan trabajar con datos tabulares y series temporales. Esta guía te introducirá a los conceptos fundamentales de pandas.

2. Objetivos de Aprendizaje

Al finalizar esta guía, serás capaz de:

- Crear y manipular Series y DataFrames
- Cargar datos desde diferentes formatos (CSV, Excel, JSON)
- Realizar operaciones básicas de limpieza y transformación de datos
- Filtrar, seleccionar y agrupar datos
- Realizar análisis exploratorio básico con pandas

3. Requisitos Previos

- Conocimientos básicos de Python
- Instalación de Python y pandas (`pip install pandas`)
- Editor de código o entorno de desarrollo (Jupyter Notebook recomendado)

4. Ejercicios Prácticos

4.1. Parte 1: Fundamentos de Pandas

1. **Creación de Series y DataFrames**

```
import pandas as pd
import numpy as np

# Crear una Serie
```

```
serie_ejemplo = pd.Series([10, 20, 30, 40], index=['a', 'b', 'c', 'd'])
print("Serie ejemplo:")
print(serie_ejemplo)

# Crear un DataFrame desde un diccionario
datos = {
    'Nombre': ['Ana', 'Juan', 'Mar a', 'Carlos'],
    'Edad': [25, 32, 28, 35],
    'Ciudad': ['Lima', 'Bogot ', 'Lima', 'Quito']
}
df_personas = pd.DataFrame(datos)
print("\nDataFrame personas:")
print(df_personas)

```

2. **Atributos básicos de un DataFrame**

```
# Explorar el DataFrame
print("Dimensiones:", df_personas.shape)
print("\nInformaci n del DataFrame:")
print(df_personas.info())
print("\nPrimeras 3 filas:")
print(df_personas.head(3))
print("\nEstad sticas descriptivas:")
print(df_personas.describe())

```

4.2. Parte 2: Carga y Exploración de Datos

1. **Carga de datos desde CSV**

```
# Cargar datos desde un archivo CSV
# Nota: Aseg rate de tener el archivo 'ejemplo_datos.csv' en tu directorio
df = pd.read_csv('ejemplo_datos.csv')

# Explorar los datos cargados
print("Primeras filas:")
print(df.head())
print("\nColumnas disponibles:", df.columns.tolist())
print("\nTipos de datos:")
print(df.dtypes)

```

2. **Manejo de valores nulos**

```
# Identificar valores nulos
print("Valores nulos por columna:")
print(df.isnull().sum())

# Estrategias para manejar valores nulos
# Opci n 1: Eliminar filas con valores nulos
df_sin_nulos = df.dropna()

# Opci n 2: Rellenar valores nulos
df_rellenado = df.fillna({
```

```
    'columna_numerica': df['columna_numerica'].mean(),
    'columna_categorica': 'Desconocido'
})

print(f"DataFrame original: {df.shape}")
print(f"DataFrame sin nulos: {df_sin_nulos.shape}")

```

4.3. Parte 3: Selección y Filtrado de Datos

1. Selección de columnas

```
# Seleccionar una columna (devuelve una Serie)
nombres = df['Nombre']
print("Columna Nombre:")
print(nombres.head())

# Seleccionar m ltiples columnas (devuelve un DataFrame)
subconjunto = df[['Nombre', 'Edad', 'Ciudad']]
print("\nSubconjunto de columnas:")
print(subconjunto.head())

```

2. Filtrado de filas

```
# Filtrar por condici n
mayores_30 = df[df['Edad'] > 30]
print("Personas mayores de 30 a os:")
print(mayores_30)

# M ltiples condiciones
limenos_mayores_25 = df[(df['Ciudad'] == 'Lima') & (df['Edad'] >
    25)]
print("\nLime os mayores de 25 a os:")
print(limenos_mayores_25)

# Filtrar con isin()
ciudades_seleccionadas = df[df['Ciudad'].isin(['Lima', 'Bogot '])]
print("\nPersonas de Lima o Bogot :")
print(ciudades_seleccionadas)

```

4.4. Parte 4: Transformación de Datos

1. Operaciones sobre columnas

```
# Crear una nueva columna
df['Edad_en_decadas'] = df['Edad'] / 10
print("DataFrame con nueva columna:")
print(df.head())

# Aplicar una funci n a una columna
df['Nombre_mayusculas'] = df['Nombre'].str.upper()
print("\nNombres en may sculas:")
print(df[['Nombre', 'Nombre_mayusculas']].head())

```

2. **Agrupación y agregación**

```
# Agrupar por ciudad y calcular estad sticas
agrupado_ciudad = df.groupby('Ciudad')
print("Conteo por ciudad:")
print(agrupado_ciudad.size())

print("\nEdad promedio por ciudad:")
print(agrupado_ciudad['Edad'].mean())

# M ltiples agregaciones
resumen = df.groupby('Ciudad').agg({
    'Edad': ['mean', 'min', 'max', 'count'],
    'Nombre': 'count'
})
print("\nResumen estad stico por ciudad:")
print(resumen)

```

4.5. Parte 5: Ordenamiento y Manipulación de Índices

1. **Ordenamiento de datos**

```
# Ordenar por una columna
df_ordenado_edad = df.sort_values('Edad')
print("Ordenado por edad (ascendente):")
print(df_ordenado_edad[['Nombre', 'Edad']].head())

# Ordenar por m ltiples columnas
df_ordenado_ciudad_edad = df.sort_values(['Ciudad', 'Edad'],
    ascending=[True, False])
print("\nOrdenado por ciudad (A-Z) y edad (descendente):")
print(df_ordenado_ciudad_edad[['Nombre', 'Ciudad', 'Edad']].head())

```

2. **Manejo de índices**

```
# Establecer una columna como  ndice
df_indexado = df.set_index('Nombre')
print("DataFrame con Nombre como  ndice :")
print(df_indexado.head())

# Resetear el  ndice
df_reseteado = df_indexado.reset_index()
print("\nDataFrame con  ndice  reseteado:")
print(df_reseteado.head())

```

5. Ejercicios de Práctica

5.1. Ejercicio 1: Análisis de Datos de Ventas

Carga el siguiente dataset de ventas y realiza las operaciones solicitadas:

```
datos_ventas = {
    'Fecha': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02', '
    2023-01-03'],
    'Producto': ['A', 'B', 'A', 'C', 'B'],
    'Cantidad': [10, 5, 8, 12, 6],
    'Precio_Unitario': [25.0, 40.0, 25.0, 30.0, 40.0]
}
df_ventas = pd.DataFrame(datos_ventas)
```

1. Calcula el total de ventas por producto (Cantidad * Precio Unitario)
2. Encuentra el producto con mayores ventas totales
3. Calcula el promedio de ventas por día

5.2. Ejercicio 2: Limpieza de Datos

Dado el siguiente dataset con problemas, realiza las operaciones de limpieza:

```
datos_problema = {
    'Nombre': ['Ana', 'Juan', None, 'Mar a', 'Carlos'],
    'Edad': [25, 32, 28, None, 35],
    'Puntuacion': [85, 92, 78, 88, None]
}
df_problema = pd.DataFrame(datos_problema)
```

1. Identifica los valores nulos en cada columna
2. Rellena los valores nulos en Edad con la mediana y en Puntuacion con la media
3. Elimina las filas donde el Nombre es nulo

6. Recursos Adicionales

- **Documentación oficial de Pandas**: `https://pandas.pydata.org/docs/`
- **Pandas Cheat Sheet**: `https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf`
- **Tutoriales interactivos**: `https://www.w3schools.com/python/pandas/default.asp`
- **Datasets para practicar**: `https://www.kaggle.com/datasets`

7. Conclusión

Pandas es una herramienta esencial para cualquier persona que trabaje con datos en Python. Esta guía ha cubierto los conceptos fundamentales, pero hay muchas más funcionalidades por explorar. La práctica constante es clave para dominar esta biblioteca.