Gestión de datos categóricos con Python

Sebastián Cotes, Yefferson González, Lorann Peñuela

IES, INFOTEP Instituto Nacional de Formación Técnica Profesional "HVG"

Humberto Velazquez Garcia

Abstract

En esta actividad, se desarrollará una codificación en Python que nos permita Cargar, leer y procesar datos de una encuesta realizada en Colombia. Fueron implementadas varias funciones para modificar los datos en formato CSV y realizar una categorizacion numerica con ellos. Se utilizó la librería "matplotlib" para la representación gráfica y "pandas para el analisis y categorización.

1 Introducción

Python ha tomado gran fuerza entre los programadores en los últimos años. Uno de los principales retos de esta actividad fue usar Python para analizar un documento con datos obtenidos de unas encuestas, comparando un antes y despues de la categorización. En este caso en particular, se creó la función "leercsv(url)" Lee el archivo CSV desde una URL de nuestro GIT y lo convierte en un DataFrame, luego "labelmappings = " crea un diccionario para almacenar los mapeos de codificación y proceder con la categorización.

2 Objetivos

Los objetivos principales de esta actividad son:

- Reforzar el lenguaje Python y aprender a usarlo para generar gráficos y limpieza de datos.
- Cargar, leer y procesar un archivo CSV para su manipulación y análisis.
- Aplicar codificación de etiquetas (Label Encoding) a las columnas no numéricas para convertir datos categóricos en datos numéricos, para que la maquinas pueda interpretar esos datos.

3 Descripción de la actividad

Se nos pide tomar una serie de datos en formato CSV, el cual tiene datos no numericos que impiden que una maquina pueda interpretar dichos datos, por ese motivo se realiza la categorizan o Codificación de Etiquetas (Label Encoding). Ejemplo si tenemos columna color ['rojo', 'azul', 'verde'], pasara a verse así :

['rojo': 0, 'azul': 1, 'verde': 2].

La Codificación de Etiquetas (Label Encoding) es una técnica de preprocesamiento de datos utilizada para convertir variables categóricas en variables numéricas. Esto es esencial cuando se trabaja con algoritmos de aprendizaje automático que solo pueden manejar datos numéricos [2].

- Identificación de categorías: Se identifican las categorías únicas en una columna categórica.
- Asignación de números: A cada categoría se le asigna un número entero único. Por ejemplo, si una columna tiene las categorías "rojo", "verde" y "azul", estas podrían ser codificadas como 0, 1 y 2, respectivamente.
- Reemplazo en el DataFrame: Las categorías originales en el DataFrame se reemplazan por sus correspondientes números enteros.

Ventajas:

Simplicidad: Es fácil de implementar y entender. Compatibilidad: Hace que los datos categóricos sean compatibles con algoritmos de aprendizaje automático que requieren datos numéricos.

Desventajas:

Orden implícito: Puede introducir un orden implícito en las categorías que no existe en los datos originales, lo que puede ser problemático para algunos algoritmos.

3.1 Codificación

Se utilizaron librerías como `pandas` y `matplotlib` para visualizar la representación gráfica y realizar la categorización.

1. `def leer_csv(url)::` Cargar el archivo CSV de nuestro git para poder categorizar.
2. `label_mappings = {}`: Diccionario para almacenar los mapeos de codificación .
3. `df . to_csv` : Guardar un nuevo archivo CSV Modificado.
4. `plt . hist ()`: Genera un histograma para la columna especificada .
5. `plt . scatter ()`:Genera un gráfico de dispersión para DatosBiv.

Listing 1: Escena en Python

```
import pandas as pd
import matplotlib.pyplot as plt
from io import StringIO

url = 'https://raw.githubusercontent.com/YeffersonGonzalez/ml-
    infotep/main/Datasets/houses_medellin.csv'
def leer_csv(url):
    """
    Lee el archivo CSV desde una URL de nuestro GIT y lo
        convierte en un DataFrame.

    Args:
        url (str): La URL del archivo CSV.

    Returns:
        pd.DataFrame: Un DataFrame con los datos del archivo
            CSV.
    """
    # Cargar el archivo CSV de nuestro git para poder
        categorizar
    # Decidimos categorizar por Numeracion o Codificaci n de
        Etiquetas (Label Encoding)
    df = pd.read_csv(url)
    return df
# Datos antes de la Codificacion de etiquetas
print(df)

 """
    Diccionario para almacenar los mapeos de codificaci n y
        crear un nuevo CSV.

    Args:
        label_mappings = {}: Diccionario para almacenar los
            mapeos de codificaci n.
        df.to_csv ('houses_medellin_categorizado.csv', index=
            False): Crea un nuevo CSV con los datos
            categorizados.
    """

label_mappings = {}

# Aplicar codificaci n a las columnas no num ricas y
    almacena
for column in df.select_dtypes(include=['object']).columns:
    df[column], mapping_index = pd.factorize(df[column])
    label_mappings[column] = dict(enumerate(mapping_index))
```

```
# Guardar un nuevo archivo CSV Modificado
df.to_csv('houses_medellin_categorizado.csv', index=False)

# Resusltado de aplicar las Etiquetas
print(label_mappings)
# Leer datos del archivo CSV
df = pd.read_csv('/content/houses_medellin_categorizado.csv')
# Datos despues de las etiquetas
print(df)

DatosU = ['Habitaciones', 'Ba os', 'Estrato', 'Antig edad',
    'Piso  N  ', 'Administraci n',
          'Precio  m  ', 'Parqueaderos', 'Estado', 'Tipo de
              apartamento', 'Precio',
          ' rea  constru da ( m  )', ' rea  privada ( m  )']

DatosBiv = ['Habitaciones', 'Estrato']
"""
    Genera gr ficos univariados (histogramas) para las
        columnas especificadas.

    Args:
        DatosU (list): Una lista de nombres de columnas para
            las cuales se generar n los histogramas.
        DatosBiv (list): Una lista de nombres de columnas para
             las cuales se generar n los gr ficos bivariados.
        plt.hist (): Genera un histograma para la columna
            especificada.
        plt.scatter (): Genera un gr fico de dispersi n para
             DatosBiv.
    """
#gr ficos univariados
for var in DatosU:
    plt.hist(df[var].dropna(), bins=50, edgecolor='black',
        alpha=0.7, color='#90EE90')
    plt.title(f'Histograma de {var}')
    plt.xlabel(var)
    plt.ylabel('Frecuencia')
    plt.show()

#gr ficos bivariados
plt.scatter(df[DatosBiv[0]], df[DatosBiv[1]], alpha=0.5)
plt.xlabel(DatosBiv[0])
plt.ylabel(DatosBiv[1])
plt.title(f'{DatosBiv[0]} vs {DatosBiv[1]}')
plt.show()
```

4 Gráficas de Resultados

Terminada la **codificación**[1], y teniendo en cuenta los requerimientos iniciales, obtenemos como resultado las siguientes gráficas.

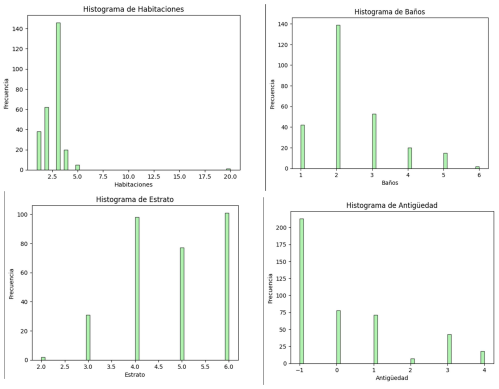

Figure 1: **Análisis Univarioado**

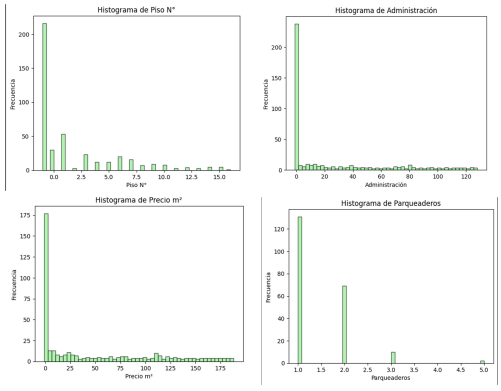

Figure 2: **Análisis Univarioado**

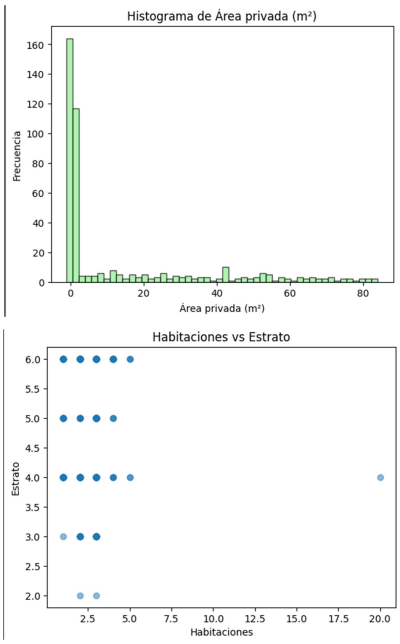

Figure 3: **Análisis Bivarioado**

5 Conclusiones

Esta actividad nos ayudó a tener una ligera idea de cómo usar la librería matplotlib para representar gráficos y de cómo aplicar operaciones vectoriales en la manipulación de posiciones de objetos. También fue posible observar de manera clara los cambios en la escena a medida que se daban las operaciones. En general, este proceso fue una experiencia útil para reforzar conceptos de programación orientada a objetos, uso de operaciones vectoriales y representaciones gráficas en Python.

References

[1] Google colab, simulación [online], available from: `https://colab.research.google.com/drive/17oOEtzJ4YrUOitU87b0feKRNm0T7MaPI?usp=sharing`.

[2] Label encoding (codificación de etiquetas) - ia blog [online], available from: `https://iartificial.blog/glossary/label-encoding-codificacion-de-etiquetas/`.