Análisis Exploratorio de Datos en Python

Germán Daniel Colón Carrasquilla

Informe de resultados obtenidos del análisis de datos exploratorio realizado al caso de estudio. AA4-EV01

Introducción

El análisis de datos es una herramienta que puede fundamentalmente cambiar la perspectiva de la resolución de problemas basados en la toma de decisiones acertadas en conjunto con la apropiada investigación, análisis y visualización de los datos. Para llevar a cabo el manejo de grandes volúmenes de datos y mejorar el entendimiento de las diversas problemáticas se tiene como eje fundamental diferentes programas y lenguajes de programación que facilitan el manejo de estos datos. No obstante, el análisis debe estar siempre orientado a resolver una problemática y dirigido por un especialista para que el desarrollo del mismo no pierda significado, ya que al no estar dotado de un propósito o un fin estos se vuelven datos sin sentidos para las personas que los vayan a interpretar.

Datos

https://www.datos.gov.co/Hacienda-y-Cr-dito-P-blico/Inmuebles-Disponibles-Para-La-Venta/72gd-px77/data

Objetivo

- Realizar el análisis exploratorio según se indica en la guía del curso para el documento suministrado.
- Crear y responder una pregunta problema teniendo en cuenta los datos en el documento.
- Resaltar la importancia de las gráficas en los análisis de datos

Análisis Exploratorio

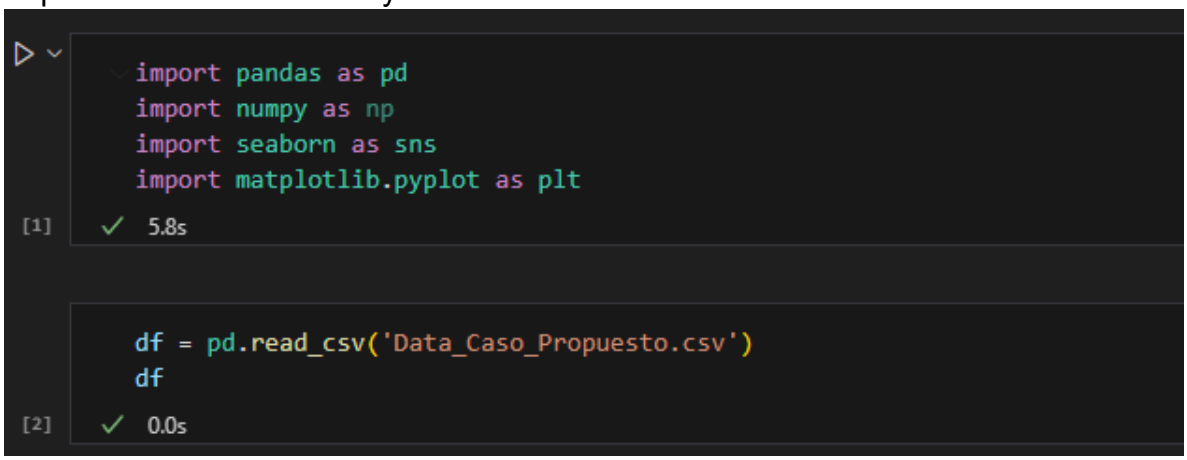
Pregunta problema

¿Variación del precio de los locales o casas según el estrato?

Esta pregunta está relacionada con la información directa del archivo, en ella se busca entender como los precios se determinan por estrato y que rige esta evaluación en cuanto a la determinación de precios.

Siguiendo con lo anterior se muestran los gráficos y las diferentes interpretaciones a cada uno de ellos para mejorar el entendimiento de la información en los datos.

- Importación de la librerías y lectura del archivo csv:



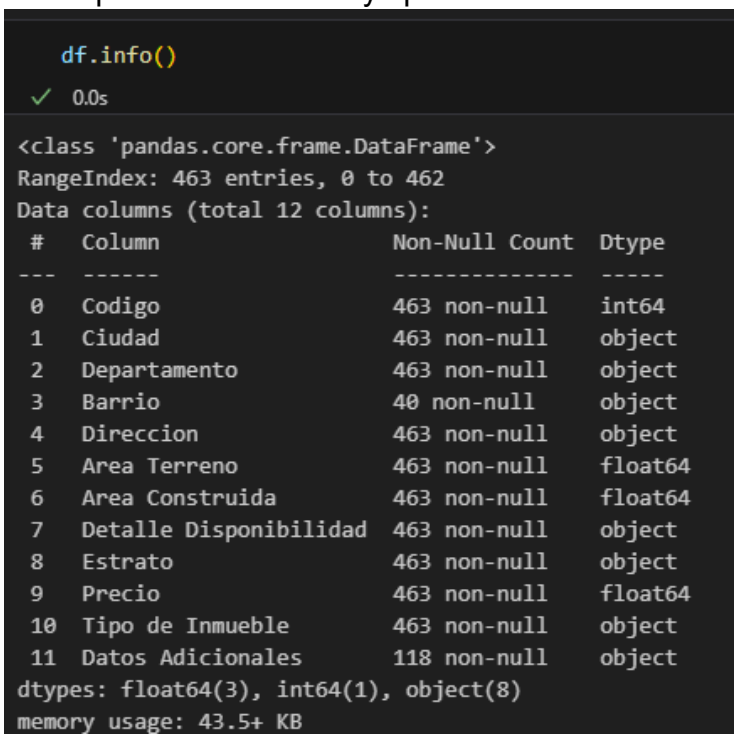
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[1] ✓ 5.8s

```
df = pd.read_csv('Data_Caso_Propuesto.csv')
df
```

[2] ✓ 0.0s

- Descripción de los datos y tipos de datos:



```
df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Codigo                 463 non-null    int64
 1   Ciudad                 463 non-null    object
 2   Departamento           463 non-null    object
 3   Barrio                 40 non-null     object
 4   Direccion              463 non-null    object
 5   Area Terreno           463 non-null    float64
 6   Area Construida        463 non-null    float64
 7   Detalle Disponibilidad 463 non-null    object
 8   Estrato                463 non-null    object
 9   Precio                 463 non-null    float64
 10  Tipo de Inmueble       463 non-null    object
 11  Datos Adicionales      118 non-null    object
dtypes: float64(3), int64(1), object(8)
memory usage: 43.5+ KB
```

El archivo está formado por datos del tipo numérico y de caracteres (letras), el numérico se divide en int (números enteros) y float (números separados por puntos)

- Datos nulos y duplicados:

```
df.isnull().sum()
[4] ✓ 0.1s
```

```
Codigo                    0
Ciudad                    0
Departamento              0
Barrio                  423
Direccion                 0
Area Terreno              0
Area Construida           0
Detalle Disponibilidad    0
Estrato                   0
Precio                    0
Tipo de Inmueble          0
Datos Adicionales       345
dtype: int64
```

```
df = df.drop_duplicates()
✓ 0.0s
```

```
df.info()
✓ 0.0s
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Codigo                  463 non-null    int64
 1   Ciudad                  463 non-null    object
 2   Departamento            463 non-null    object
 3   Barrio                  40 non-null     object
 4   Direccion               463 non-null    object
 5   Area Terreno            463 non-null    float64
 6   Area Construida         463 non-null    float64
 7   Detalle Disponibilidad  463 non-null    object
 8   Estrato                 463 non-null    object
 9   Precio                  463 non-null    float64
 10  Tipo de Inmueble        463 non-null    object
 11  Datos Adicionales       118 non-null    object
dtypes: float64(3), int64(1), object(8)
memory usage: 43.5+ KB
```

Los datos nulos conforman una cantidad importante, no se pueden descartar porque afectaran significativamente el análisis, cabe destacar que para el análisis las variables en las columnas ‘Barrio’ y ‘Datos adicionales’ no se van a utilizar para el análisis.
No hay valores duplicados entre los datos.

- Variables que se descartan:

```
cantidad_valores_cero_areac = (df['Area Construida'] == 0).sum()
cantidad_valores_cero_areac
[24] ✓ 0.0s
np.int64(453)
```

```
cantidad_valores_cero_areat = (df['Area Terreno'] == 0).sum()
cantidad_valores_cero_areat
[25] ✓ 0.0s
np.int64(445)
```

Estas variables, aunque pueden influir en responder la pregunta problema tienen muchos valores en cero por lo que hace imposible que se pueda hacer análisis con ellos.

```
df['Detalle Disponibilidad'].value_counts()
```
[28] ✓ 0.0s

```
Detalle Disponibilidad
COMERCIALIZABLE                    289
COMERCIALIZABLE CON RESTRICCION     83
COMERCIALIZABLE TERCEROS            41
EN PUJA                             23
COMERCIALIZABLE VENTA ANTICIPADA    14
COMERCIALIZABLE FIDUCIA             13
Name: count, dtype: int64
```

La variable 'Detalle Disponibilidad' se descarta porque se sobreentiende que las propiedades están a la venta, por ende, no es necesario ponerlo en el análisis.

Las demás variables diferentes a las mencionadas si se pueden usar para otros tipos de análisis.

- Descripción de la tendencia de los datos:

```
df.describe()
```
[10] ✓ 0.0s

	Codigo	Area Terreno	Area Construida	Precio
count	463.000000	4.630000e+02	463.000000	4.630000e+02
mean	18003.151188	1.515204e+04	87.517279	6.672032e+08
std	1992.191499	1.827101e+05	1137.469077	3.272992e+09
min	2575.000000	0.000000e+00	0.000000	4.650000e+06
25%	18184.500000	0.000000e+00	0.000000	1.230500e+07
50%	18332.000000	0.000000e+00	0.000000	1.587000e+07
75%	18539.500000	0.000000e+00	0.000000	1.379955e+08
max	19344.000000	3.217197e+06	22724.000000	4.523379e+10

- Cantidad de datos por estrato:

```
df['Estrato'].value_counts()
```
[37] ✓ 0.0s

```
Estrato
COMERCIAL     307
RURAL          40
UNO            21
TRES           19
CUATRO         19
INDUSTRIAL     16
DOS            16
SEIS           15
CINCO          10
Name: count, dtype: int64
```

- Creación de nueva variable para responder la pregunta

```
[11]
rango = [4650000, 12300000, 15870000, 137700000, 667000000, 45330000000]
rango_precios = ['BAJO', 'MEDIANO', 'ALTO', 'ELEVADO', 'CARO']
df['Rango_precio'] = pd.cut(df['Precio'], rango, labels= rango_precios)
df.head(5)
✓ 0.0s
```

Se crea la variable para determinar como el precio encasilla la propiedad en un término bajo, mediano, alto, elevado y caro dependiendo de su precio. Los datos para los rangos de precios se tomaron teniendo en cuenta la tendencia de los datos en el apartado anterior.

- Cantidad de propiedades por rango de precio:

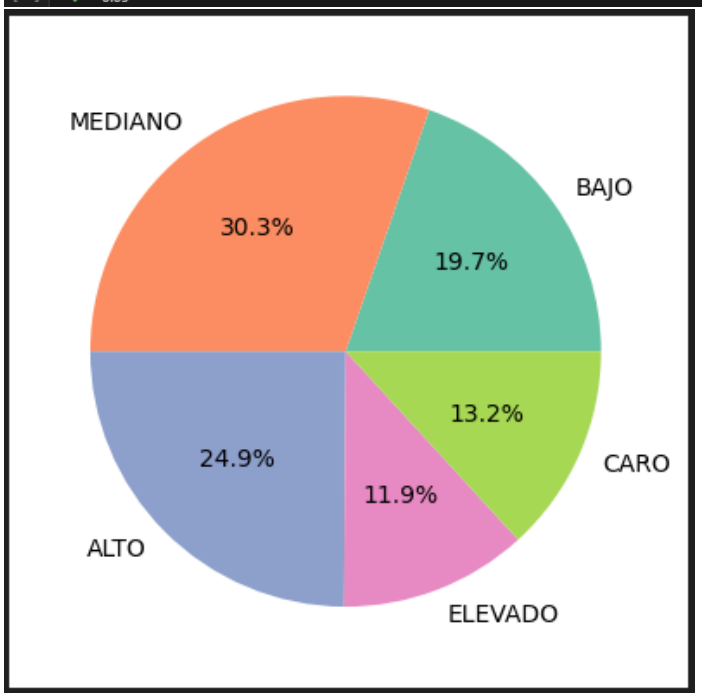
```
[23]
df['Rango_precio'].value_counts()
✓ 0.0s

...
Rango_precio
MEDIANO    140
ALTO       115
BAJO        91
CARO        61
ELEVADO     55
Name: count, dtype: int64
```

Se muestran según el rango de precio en que categoría entran las propiedades.

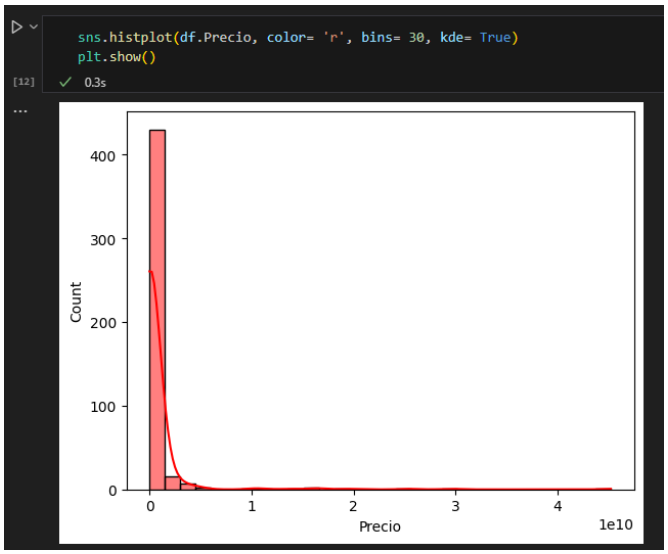
- Gráfico de torta de las categorías:

```
[47]
total_rango_precio = df['Rango_precio'].groupby(df['Rango_precio']).count()
labels = ['BAJO', 'MEDIANO', 'ALTO', 'ELEVADO', 'CARO']
colors = sns.color_palette('Set2')[0:5]
plt.pie(total_rango_precio, labels= labels, colors= colors, autopct= '%.1f%%')
plt.show()
✓ 0.0s
```



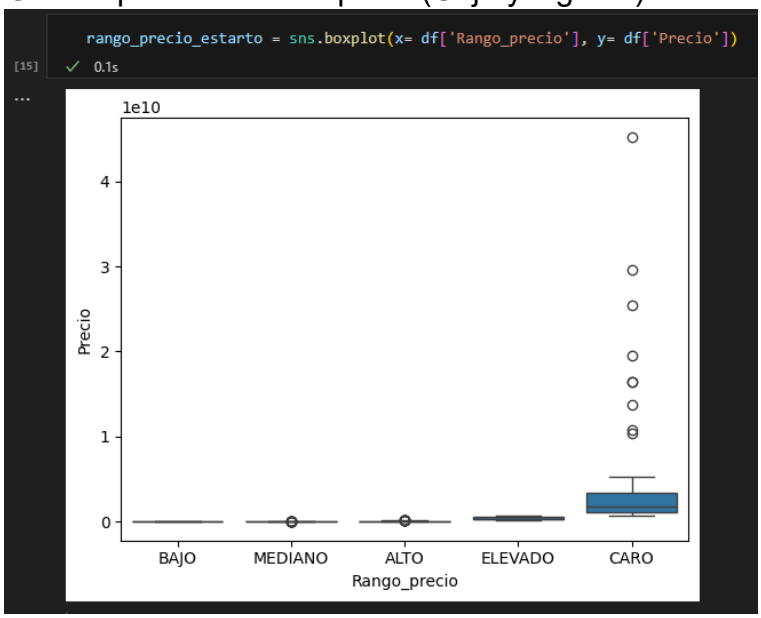
Este grafico ayuda a identificar la cantidad de datos que comparten una característica conocida y la cuantifica en un porcentaje, de esta forma podemos visualizar que porcentaje de cada dato comparte una misma característica.

- Gráfico de frecuencia:



Al ser valores tan grandes y concentrarse en ciertas zonas hace muy difícil apreciar la tendencia del precio por un tipo de propiedad en específico.

- Gráfico para valores atípicos (Caja y bigotes):



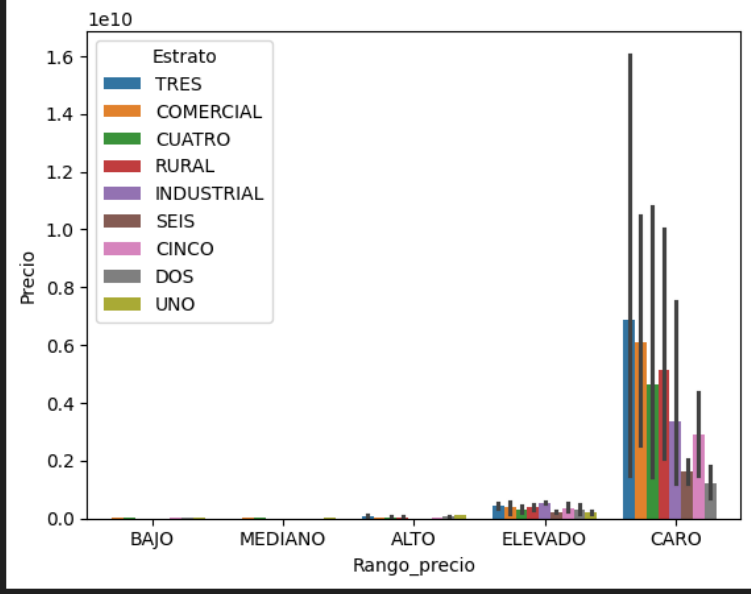
Se presentan muchos más valores atípicos de valor ‘Caro’ que, en las demás categorías, la desigualdad en los datos hace muy difícil medir las demás categorías bajo el mismo gráfico.

- Gráfico para análisis multivariado:

```
sns.barplot(data= df, x= 'Rango_precio', y= 'Precio', hue= 'Estrato')
```

```
[44]  ✓ 1.1s
```

```
<Axes: xlabel='Rango_precio', ylabel='Precio'>
```



En el gráfico se muestran las propiedades con un rango de precio más alto según su estrato, esto nos ayuda a entender que propiedades son más caras y como no depende solo del estrato su precio, aunque también puede ser por la cantidad de las propiedades por estrato no es igualitaria.

Conclusión

- Según los datos podemos afirmar que los precios de las propiedades si varían dependiendo de su estrato, también vemos que la mayoría del valor se concentra en los de estrato 3 y comercial.
- Los precios para las propiedades estrato 3 presentan la mayoría de los valores atípicos ya que en todos los datos solo hay 19 propiedades de este estrato y aun así sustentan la mayor cantidad de los precios altos de todos los datos.
- Dependiendo de que se necesite sustentar el análisis exploratorio de estos datos puede responder más preguntas, sin embargo, en la parte de la recolección de datos donde muchos datos no concuerdan o faltan hace a la vez limitado su desarrollo.
- La importancia de las gráficas en el análisis de datos puede maximizar el entendimiento de los datos y ayudar a tomar las decisiones de forma más consciente de todo lo que rodea a la información dada por los datos.