Universidad Nacional Autónoma de México

Facultad de Ingeniería

PRÁCTICA 1

ANÁLISIS EXPLORATORIO DE DATOS (EDA)

Minería de Datos

Profesor:

Dr. Molero Castillo Guillermo Gilberto

Grupo 1

Alumna:

Monroy Velázquez Alejandra Sarahí

INTRODUCCIÓN

El sector inmobiliario de Melbourne, Australia continúa en auge desde hace algunos años.

Es de interés conocer la tendencia inmobiliaria en dicha ciudad debido a que cada vez es más difícil adquirir una unidad de 2 dormitorios a un precio razonable.

OBJETIVO

Encontrar información de interés para predecir la próxima tendencia inmobiliaria en Melbourne.

DESARROLLO

El conjunto de datos corresponde a Melbourne Housing Snapshot de Kaggle. Este conjunto de datos incluye: dirección, tipo de inmueble, suburbio, método de venta, habitaciones, precio, agente inmobiliario, fecha de venta y Distancia desde C.B.D. (Distrito Central de Negocios).

El diccionario de datos es el siguiente:

Item	Column name	Definición
1	Rooms	Número de habitaciones
2	Price	Precio en dolares
3	Method	**S** - propiedad vendida; **SP** - propiedad vendida antes; **PI** - propiedad transferida; **PN** - vendida antes no revelada; **SN** - vendida no revelada; **NB** - sin oferta; **VB** - oferta del proveedor; **W** - retirada antes de la subasta; **SA** - vendida después de subasta; **SS** - vendida después del precio de subasta no revelado. **N/A** - precio u oferta más alta no disponible.
4	Type	**br** - dormitorio (s); **h** - casa, cabaña, villa, semi, terraza; **u** - unidad, dúplex; **t** - casa adosada; **dev site** – en desarrollo; o **res** - otro residencial.
5	SellerG	Agente de bienes raíces
6	Date	Fecha de venta
7	Distance	Distancia del CBD (Centro de negocios)
8	Regionname	Región general (oeste, noroeste, norte, noreste ...)
9	Propertycount	Número de propiedades que existen en el suburbio
10	Bedroom2	Número de dormitorios (de otra fuente)
11	Bathroom	Cantidad de baños
12	Car	Número de estacionamientos
13	Landsize	Tamaño del terreno
14	BuildingArea	Tamaño del edificio
15	CouncilArea	Consejo de gobierno de la zona (Municipio)

Primero comenzamos la importación de bibliotecas correspondientes que nos ayudarán para la realización del código, las cuales son *pandas* para la manipulación y análisis de datos, *numpy* para crear vectores y matrices, *matplotlib* para la generación de gráficas, así como *seaborn* para la visualización de datos. También *file* para subir el archivo *csv* que contiene todos los datos.

Una vez importadas, se leen y se despliegan en pantalla, en este caso también se puede utilizar *head()* para visualizar solo las primeras cinco filas del *dataframe*.

```
import pandas as pd          #Para la manipulación y analisis de datos
import numpy as np           #Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt #Para la generación de gráficas a partir de los datos
import seaborn as sns        #Para la visualización de datos basados en matplotlib
%matplotlib inline
#Para generar imagenes dentro del cuaderno
```

```
from google.colab import files
files.upload()
```

```
Elegir archivos melb_data.csv
• melb_data.csv(application/vnd.ms-excel) - 2091239 bytes, last modified: 21/9/2021 - 100% done
Saving melb_data.csv to melb_data.csv
{'melb_data.csv': b"Suburb,Address,Rooms,Type,Price,Method,SellerG,Date,Distance,Postcoc
```

```
DatosMelbourne = pd.read_csv('melb_data.csv')
```

```
DatosMelbourne
```

Ahora que el *dataframe* está cargado, comenzamos con los pasos vistos en clase:

1) ***Descripción de la estructura de los datos***

   Lo primero que se hace es ver la forma de la matriz, utilizando el atributo *shape* el cual nos regresa la cantidad de filas y columnas que tiene, en este caso nuestro *dataframe* tenemos 21 columnas con 13580 registros.

```
DatosMelbourne.shape
```

```
(13580, 21)
```

   Luego observamos los tipos de datos, utilizando el atributo *dtypes*, en este caso tenemos datos que son objetos y flotantes:

```
DatosMelbourne.dtypes
```

```
Suburb           object
Address          object
Rooms             int64
Type             object
Price           float64
Method           object
SellerG          object
Date             object
Distance        float64
Postcode        float64
Bedroom2        float64
Bathroom        float64
Car             float64
Landsize        float64
BuildingArea    float64
YearBuilt       float64
CouncilArea      object
Lattitude       float64
Longtitude      float64
Regionname       object
Propertycount   float64
dtype: object
```

Una vez que tenemos una idea general de los datos que conforman nuestro *dataframe* pasamos al siguiente paso.

2) *Identificación de datos faltantes*

En este paso observamos los datos faltantes dentro del *dataframe*, esto nos servirá para tomar decisiones más adelante, ya sea que si no son relevantes los eliminemos o los remplazemos con diferente información, para ello utilizamos la función *isnull().sum()* la cual nos regresa la suma de todos los valores nulos en cada variable.

```
DatosMelbourne.isnull().sum()
```

```
Suburb             0
Address            0
Rooms              0
Type               0
Price              0
Method             0
SellerG            0
Date               0
Distance           0
Postcode           0
Bedroom2           0
Bathroom           0
Car               62
Landsize           0
BuildingArea    6450
YearBuilt       5375
CouncilArea     1369
Lattitude          0
Longtitude         0
Regionname         0
Propertycount      0
dtype: int64
```

Como vemos, *Car*, *BuildingArea*, *YearBuilt* y *CouncilArea* presentan datos faltantes.

Otra opción es usar *info()* la cual es similar:

```
DatosMelbourne.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  object
 1   Address        13580 non-null  object
 2   Rooms          13580 non-null  int64
 3   Type           13580 non-null  object
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object
 6   SellerG        13580 non-null  object
 7   Date           13580 non-null  object
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13518 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   7130 non-null   float64
 15  YearBuilt      8205 non-null   float64
 16  CouncilArea    12211 non-null  object
 17  Lattitude      13580 non-null  float64
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

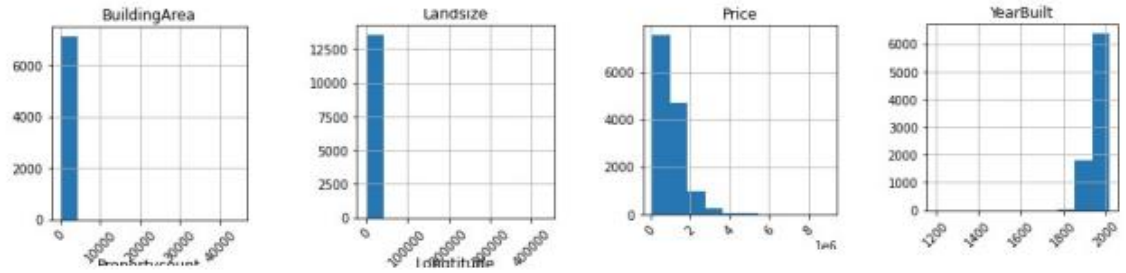
3) *Detección de valores atípicos*

En este paso utilizamos gráficos para visualizar de una forma más facil y rápida los valores atípicos que se encuentran en el *dataframe*. En este paso hacemos cinco pasos intermedios:

a) *Distribución de variables numéricas*

Utilizamos los histogramas para visualizar valores atípicos que podrían ser errores de medición, lo que se buscó fueron limites que no tuvieran sentido, en este caso vemos que *BuildingArea*, *Landsize*, *Price* y *YearBuilt* tienen valores atípicos de acuerdo con las gráficas:

```
DatosMelbourne.hist(figsize=(14,14), xrot=45)
plt.show()
```



b) *Resumen estadístico de variables numéricas*

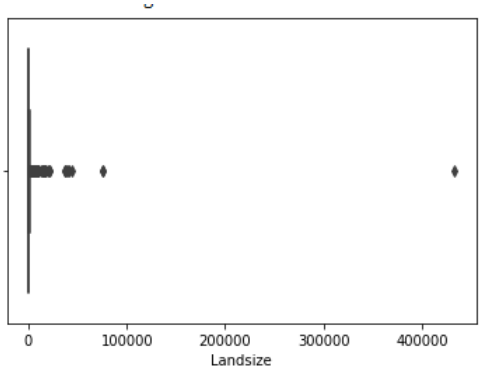
Después utilizamos *describe()* para observar la matriz que muestra un resumen estadístico de las variables numéricas, con base en ello volvemos a encontrar que podemos identificar valores nulos o perdidos.

```
DatosMelbourne.describe()
```

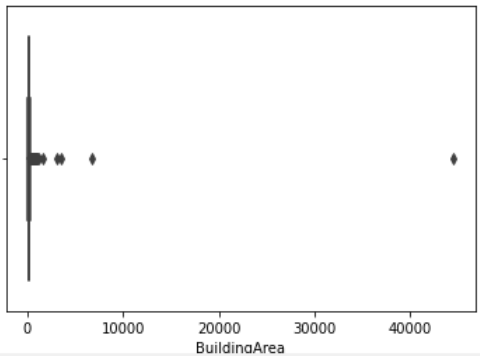
	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000	13518.000000	13
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242	1.610075	
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712	0.962634	3
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	0.000000	
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000	1.000000	
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000	2.000000	
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000	2.000000	
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000	10.000000	433

c) *Diagramas para detectar posibles valores atípicos*

Para una mejor visualización de los datos atípicos se utiliza *seaborn*, lo que nos permite visualizar los diagramas de cajas, con base en ellos podemos observar los valores fuera de rango, lo que una vez más nos confirma que existen valores atípicos en las variables mencionadas.




```
/usr/local/lib/python3.7/dist-packages/seaborn/_de
  FutureWarning
```



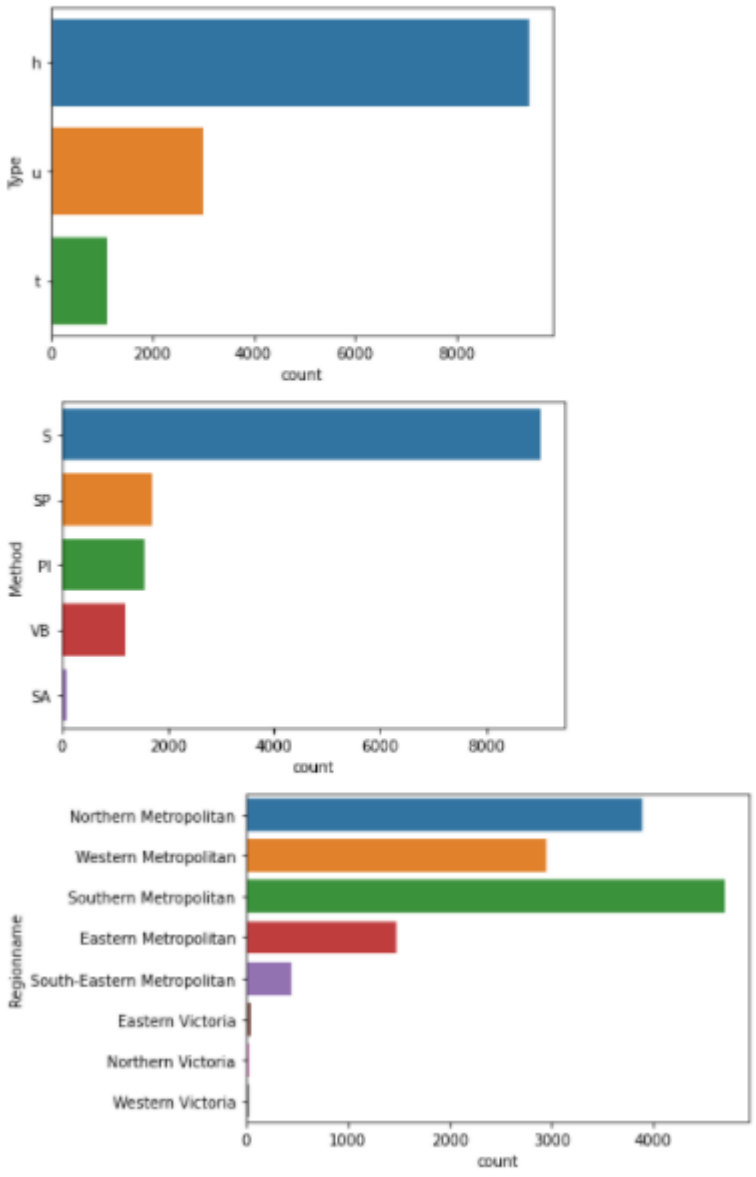
d) *Distribución de variables categóricas*

En este paso observamos el recuento de los valores de cada variable, el número de clases únicas, la clase más frecuente y con qué frecuencia ocurre esa clase en el conjunto de datos. algunas clases tienen demasiados valores únicos, como *Address*, seguida de *Suburb* y *SellerG*. A partir de estos hallazgos, se puede graficar las variables con 10 o menos clases únicas.

```
DatosMelbourne.describe(include='object')
```

	Suburb	Address	Type	Method	SellerG	Date	CouncilArea	Regionname
count	13580	13580	13580	13580	13580	13580	12211	13580
unique	314	13378	3	5	268	58	33	8
top	Reservoir	28 Blair St	h	S	Nelson	27/05/2017	Moreland	Southern Metropolitan
freq	359	3	9449	9022	1565	473	1163	4695

```
for col in DatosMelbourne.select_dtypes(include='object'):
  if DatosMelbourne[col].nunique()<10:sns.countplot(y=col, data=DatosMelbourne)
  plt.show()
```



e) *Agrupación por variables categóricas*

En este paso se hace algo similar al anterior solo que, en lugar de graficarlo, generamos una matriz que nos agrupa las variables por clases, tomando en cuenta solo las variables con 10 o menos clase, y nos muestra la media de cada una.

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea
	mean	mean	mean	mean	mean	mean	mean	mean	mean
Type									
h	3.260874	1.242665e+06	10.979479	3104.080643	3.229336	1.613822	1.771222	617.181924	176.866248
t	2.837522	9.337351e+05	9.851346	3100.777379	2.814183	1.809695	1.555655	279.606822	140.046323
u	1.963871	6.051275e+05	7.607391	3110.797481	1.966523	1.183295	1.128358	477.314219	80.737121

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingAre
	mean	mean	mean	mean	mean	mean	mean	mean	mean
Method									
PI	3.077366	1.133242e+06	9.482097	3106.742327	3.062660	1.714194	1.703918	521.682864	158.78310
S	2.941809	1.087327e+06	10.431523	3106.171359	2.914875	1.498781	1.602581	531.129905	155.78164
SA	3.010870	1.025772e+06	12.385870	3132.304348	3.010870	1.554348	1.769231	699.532609	151.45431
SP	2.795655	8.998924e+05	10.374692	3096.480916	2.785672	1.456254	1.560472	469.346447	128.63136
VB	2.924103	1.166510e+06	8.273728	3107.337781	2.896580	1.675563	1.602359	927.331943	152.41909

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	Build
	mean	mean	mean	mean	mean	mean	mean	mean	mean
Regionname									
Eastern	3.322230	1.104080e+06	13.901088	3111.162475	3.313392	1.698844	1.792916	634.133923	178

4) *Identificación de relaciones entre pares de variables*

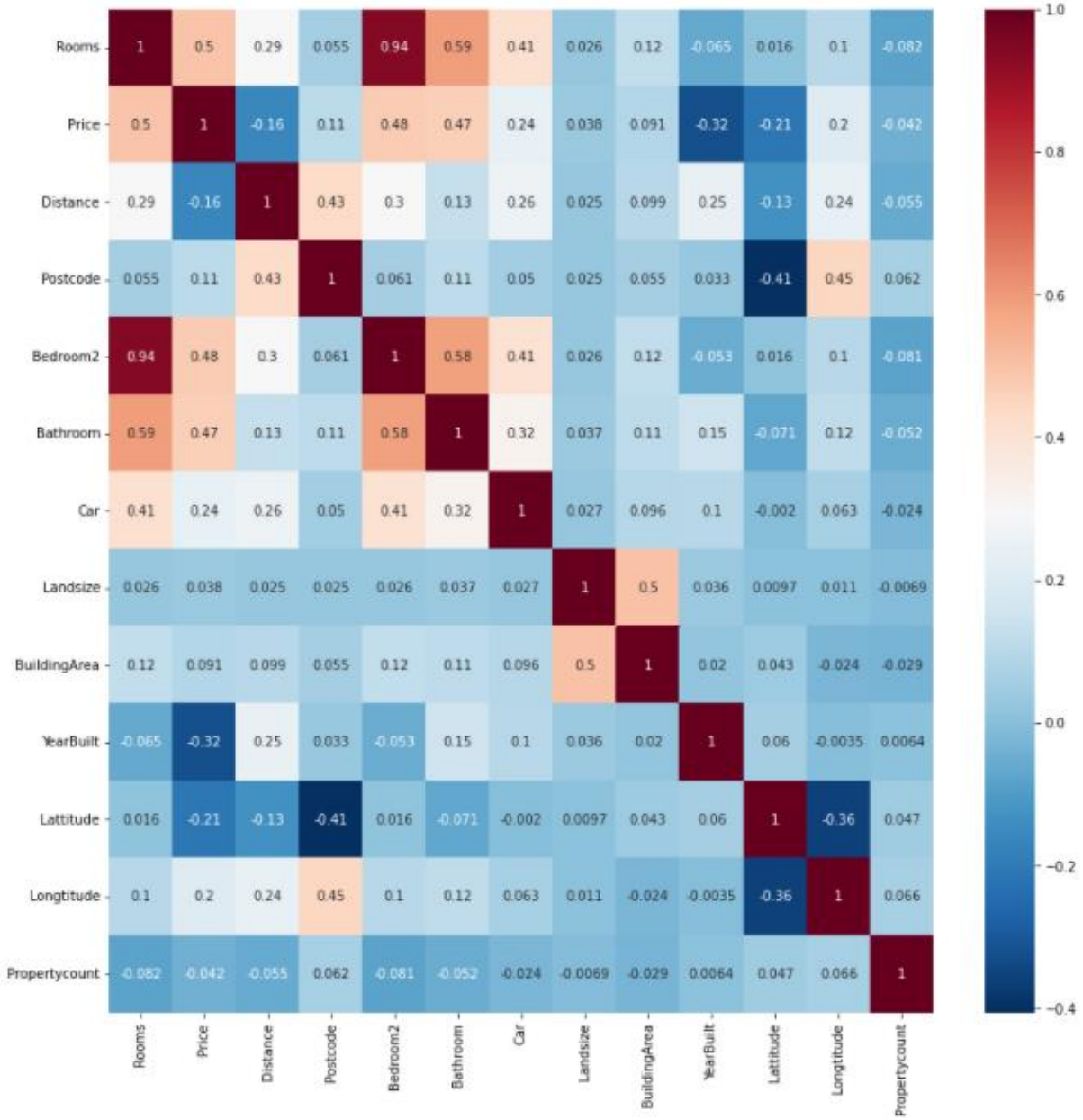
Por último, hacemos uso de la correlación para analizar la relación entre las variables numéricas, utilizando la función *corr()*, esta nos regresara una matriz:

```
DatosMelbourne.corr()
```

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Propertycount
Rooms	1.000000	0.496634	0.294203	0.055303	0.944190	0.592934	0.408483	0.025678	0.124127	-0.065413	0.015948	0.100771	-0.081530
Price	0.496634	1.000000	-0.162522	0.107867	0.475951	0.467038	0.238979	0.037507	0.090981	-0.323617	-0.212934	0.203656	-0.042153
Distance	0.294203	-0.162522	1.000000	0.431514	0.295927	0.127155	0.262994	0.025004	0.099481	0.246379	-0.130723	0.239425	-0.054910
Postcode	0.055303	0.107867	0.431514	1.000000	0.060584	0.113664	0.050289	0.024558	0.055475	0.032863	-0.406104	0.445357	0.062304
Bedroom2	0.944190	0.475951	0.295927	0.060584	1.000000	0.584685	0.405325	0.025646	0.122319	-0.053319	0.015925	0.102238	-0.081350
Bathroom	0.592934	0.467038	0.127155	0.113664	0.584685	1.000000	0.322246	0.037130	0.111933	0.152702	-0.070594	0.118971	-0.052201
Car	0.408483	0.238979	0.262994	0.050289	0.405325	0.322246	1.000000	0.026770	0.096101	0.104515	-0.001963	0.063395	-0.024295
Landsize	0.025678	0.037507	0.025004	0.024558	0.025646	0.037130	0.026770	1.000000	0.500485	0.036451	0.009695	0.010833	-0.006854
BuildingArea	0.124127	0.090981	0.099481	0.055475	0.122319	0.111933	0.096101	0.500485	1.000000	0.019665	0.043420	-0.023810	-0.028840
YearBuilt	-0.065413	-0.323617	0.246379	0.032863	-0.053319	0.152702	0.104515	0.036451	0.019665	1.000000	0.060445	-0.003470	0.006361
Lattitude	0.015948	-0.212934	-0.130723	-0.406104	0.015925	-0.070594	-0.001963	0.009695	0.043420	0.060445	1.000000	-0.357634	0.047086
Longtitude	0.100771	0.203656	0.239425	0.445357	0.102238	0.118971	0.063395	0.010833	-0.023810	-0.003470	-0.357634	1.000000	0.065988
Propertycount	-0.081530	-0.042153	-0.054910	0.062304	-0.081350	-0.052201	-0.024295	-0.006854	-0.028840	0.006361	0.047086	0.065988	1.000000

Para visualizar mejor esta correlación, utilizamos nuevamente la biblioteca *seaborn* para generar un mapa de calor, en este podemos observar que la parte inferior del triángulo es exactamente la misma que el de la parte superior, en este igual podemos ver las correlaciones e identificarlas mediante el calor, entre más rojo es el cuadrado significa que las variables son más similares, lo contrario cuando se acerca al color azul.

```
plt.figure(figsize=(14, 14))
sns.heatmap(DatosMelbourne.corr(), cmap='RdBu_r', annot=True)
plt.show()
```



CONCLUSIÓN

En esta practica nos introducimos al análisis exploratorio de datos, por lo que aprendimos los primeros pasos para analizar un conjunto de datos, en este caso utilizamos la data del sector inmobiliario en Melbourne, con base en ella observamos algunos aspectos estadísticos, empezando desde con cuantos registros se conformaba, así como que tipo de datos contenía, también nos dimos cuenta de los datos faltantes o nulos. Luego utilizamos funciones un poco más interesantes ayudándonos de las librerías para graficar y observar los datos atípicos dentro del *dataframe*, ya que es mucho más facil visualizarlos en ellas. Por último, aprendimos acerca de la correlación para identificar las relaciones entre las variables y una vez más volvimos a hacer uso de un gráfico, es decir del mapa de calor, para visualizar fácilmente esta correlación.

El análisis exploratorio de datos es un proceso que nos ayuda a tener un panorama general de los datos con los que se trabajaran, además ayuda a identificar posibles valores que podrían traernos un problema a la hora de implementar algoritmos, ya sea valores atípicos, nulos, faltantes, etc.