Universidad Nacional Autónoma de México

Facultad de Ingeniería

PRÁCTICA 3

SELECCIÓN DE CARACTERÍSTICAS

Minería de Datos

Profesor:

Dr. Molero Castillo Guillermo Gilberto

Grupo 1

Alumna:

Monroy Velázquez Alejandra Sarahí

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 *drive* para leer el archivo *csv* que contiene todos los datos.

Una vez importadas, el *dataframe* se lee y se despliega en pantalla:

```
import pandas as pd               # Para la manipulación y análisis 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 basado en matplotlib
%matplotlib inline
#Para generar imagenes dentro del cuaderno
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
DatosMelbourne = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/melb_data.csv")
DatosMelbourne
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bedr
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	

✓ 1 s se ejecutó 12:22

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

1) ***Descripción de la estructura de los datos***
   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 remplacemos 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()
```

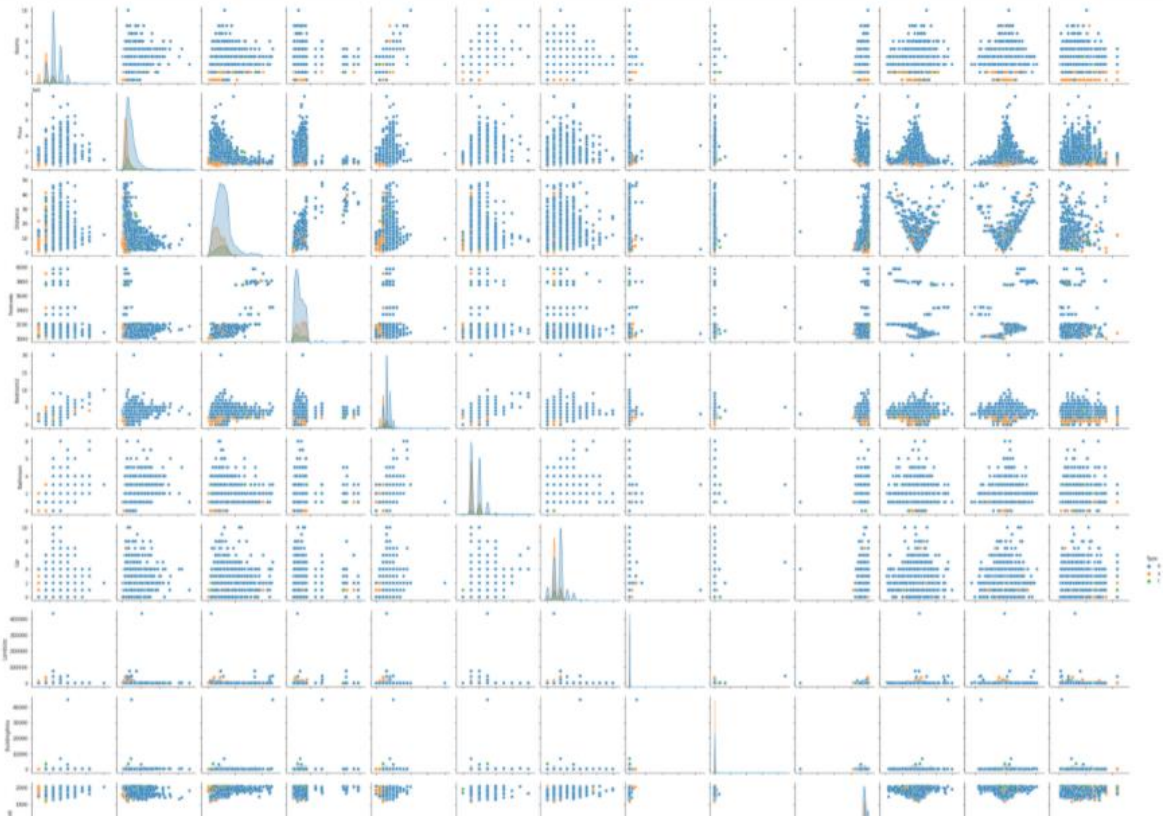
```
Suburb             0
Address            0
Rooms              0
Type               0
Price              0
Method             0
SellerG            0
Date               0
Distance           0
Postcode           0
Bedroom2           0
Bathroom           0
Car               62
Landsize           0
BuildingArea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

Como vemos, en las variables *Car*, *BuildingArea*, *YearBuilt*, y *CouncilArea*, hay datos faltantes.

3) *Evaluación Visual*

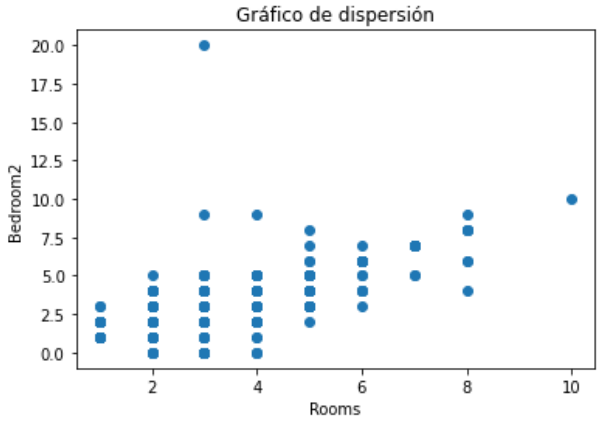
En este paso utilizamos gráficos para visualizar de una forma más facil y rápida los valores numéricos entre pares de variables. Para ello generamos gráficas de dispersión mediante la función *pairplot* de *seaborn*, que nos regresara todas las graficas que se pueden realizar en el conjunto de datos:

```
sns.pairplot(DatosMelbourne, hue='Type')
plt.show()
```



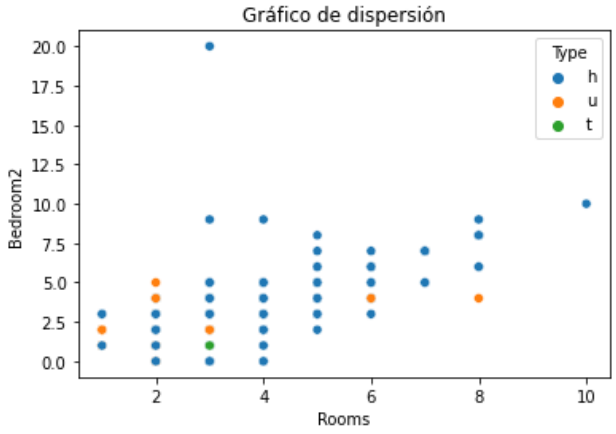
Para visualizar de una mejor forma las gráficas, podemos filtrar para solo ver la grafica del par de variables de nuestro interés, en este caso vamos a observar la grafica de las variables Bedroom2 vs Rooms:

```
plt.plot(DatosMelbourne['Rooms'], DatosMelbourne['Bedroom2'], 'o')
plt.title('Gráfico de dispersión')
plt.xlabel('Rooms')
plt.ylabel('Bedroom2')
plt.show()
```



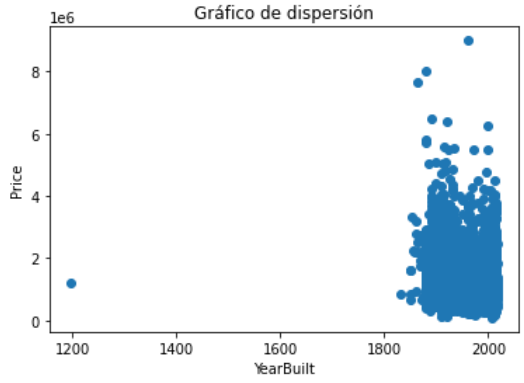
Si interpretamos este grafico podríamos decir que la relación que existe entre ambas variables es una baja correlación positiva, ya que el valor de x aumenta ligeramente a medida que aumenta el valor de y. También podemos cambiar el modo de graficar, podemos colorear el tipo de habitación, recordando que de acuerdo con el diccionario de datos tenemos que *h* corresponde a *"casa, cabañ, villa, semi o terraza"*, *u* corresponde a *"unidad o duplex"*, y *t* corresponde a *"casa adosada"*.

```
sns.scatterplot(x='Rooms', y ='Bedroom2', data=DatosMelbourne, hue='Type')
plt.title('Gráfico de dispersión')
plt.xlabel('Rooms')
plt.ylabel('Bedroom2')
plt.show()
```

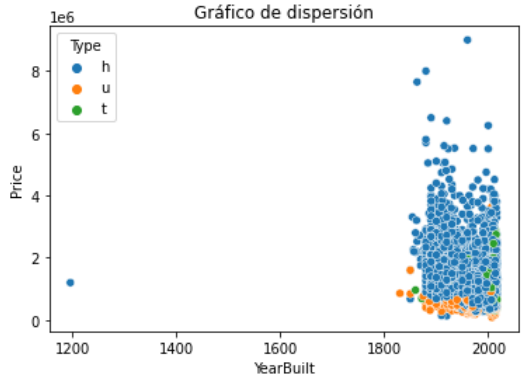


En este otro ejemplo observamos las graficas de *YearBuilt vs Price*, aunque en este caso podríamos decir que existe una correlación positiva, esta es más compleja que la de las gráficas anteriores; aprovechando que tenemos un gráfico también podemos ubicar los valores atípicos, si observamos bien existe un punto que está pegado a la izquierda, por lo que se sesga bastante del conjunto de datos que se agrupan del lado derecho, lo que nos indica que existen datos atípicos.

```
plt.plot(DatosMelbourne['YearBuilt'], DatosMelbourne['Price'], 'o')
plt.title('Gráfico de dispersión')
plt.xlabel('YearBuilt')
plt.ylabel('Price')
plt.show()
```

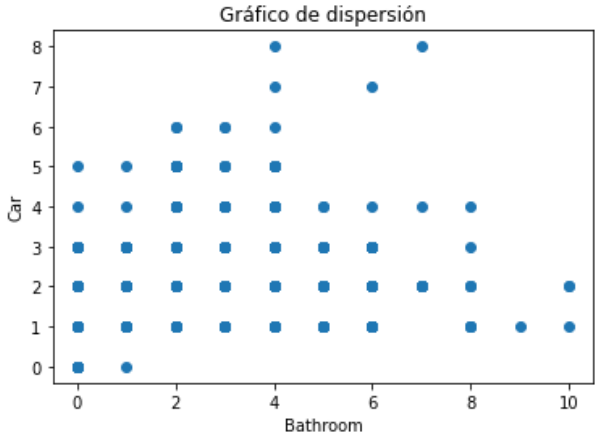



```
sns.scatterplot(x='YearBuilt', y ='Price', data=DatosMelbourne, hue='Type')
plt.title('Gráfico de dispersión')
plt.xlabel('YearBuilt')
plt.ylabel('Price')
plt.show()
```



Como último ejemplo tenemos el de la gráfica de *Car vs Bathroom*, en la cual observamos que no hay una relación entre ambas variables, ya que los puntos que se generar están dispersos por toda la gráfica.

```
plt.plot(DatosMelbourne['Car'], DatosMelbourne['Bathroom'], 'o')
plt.title('Gráfico de dispersión')
plt.xlabel('Bathroom')
plt.ylabel('Car')
plt.show()
```



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

Si queremos ubicar fácilmente las correlaciones con respecto a una variable podemos filtrar los resultados por variable y como extra al utilizar la función *sort_values* podremos acomodar las variables de forma ascendente o descendente de acuerdo al valor de la correlación.

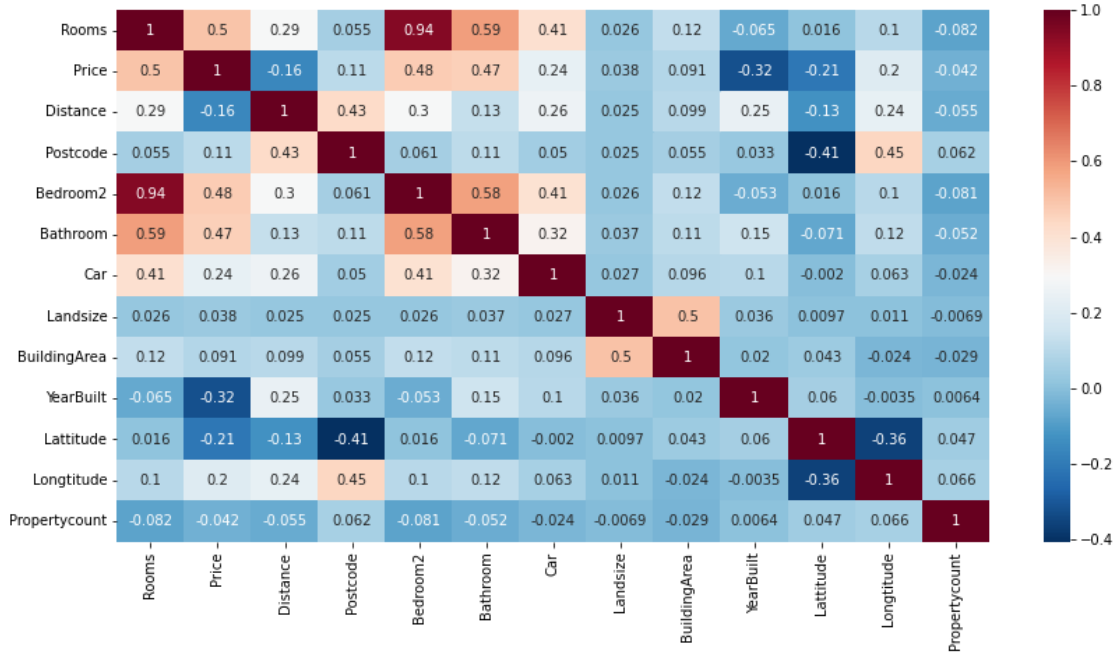
En el siguiente ejemplo observamos las correlaciones con respecto a la variable *Rooms* y las ordenamos de forma descendente:

```
print(DatosMelbourne.corr()['Rooms'].sort_values(ascending=False)[:10], '\n')
```

```
Rooms           1.000000
Bedroom2        0.944190
Bathroom        0.592934
Price           0.496634
Car             0.408483
Distance        0.294203
BuildingArea    0.124127
Longtitude      0.100771
Postcode        0.055303
Landsize        0.025678
Name: Rooms, dtype: float64
```

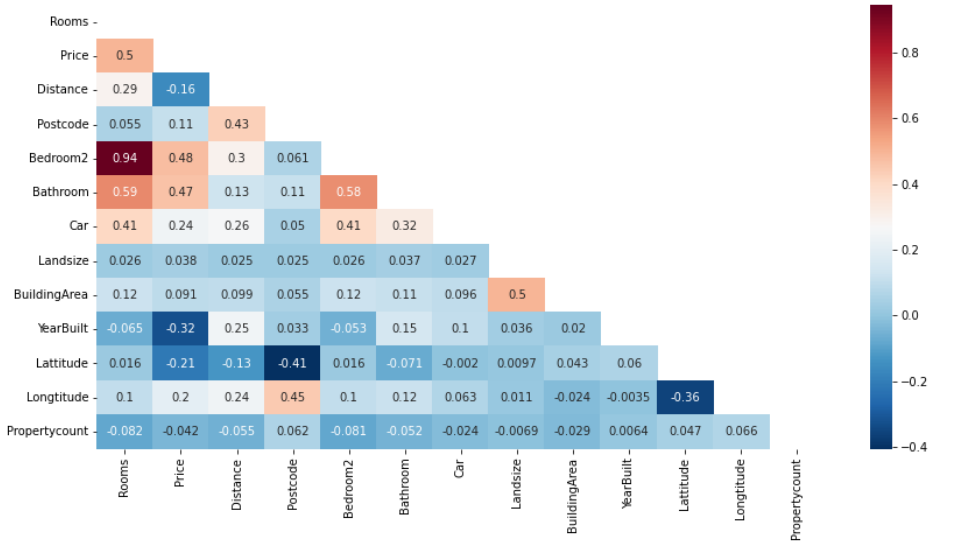
Para visualizar mejor la matriz correlación, utilizamos nuevamente la biblioteca *seaborn* para generar un mapa de calor, entre más rojo es el cuadrado significa que las variables son más similares, lo contrario cuando se acerca al color azul.

```
plt.figure(figsize=(14,7))
sns.heatmap(DatosMelbourne.corr(), cmap='RdBu_r', annot=True)
plt.show()
```

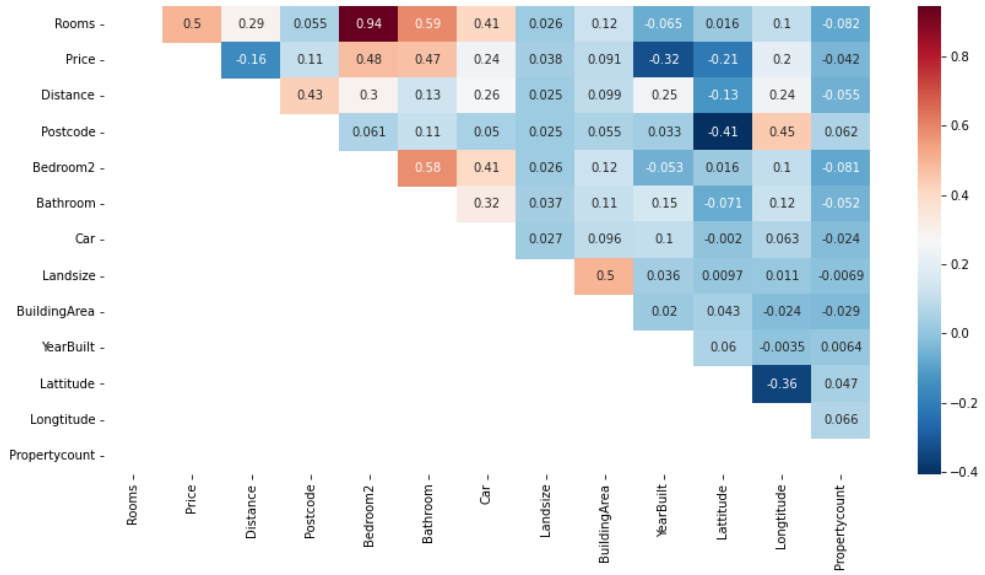


Para una mayor comodidad podemos solamente generar ya sea la parte inferior del mapa de calor o la parte superior, así ya no veríamos los datos duplicados:

```
plt.figure(figsize=(14,7))
MatrizInf = np.triu(DatosMelbourne.corr())
sns.heatmap(DatosMelbourne.corr(), cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```




```
plt.figure(figsize=(14,7))
MatrizSup = np.tril(DatosMelbourne.corr())
sns.heatmap(DatosMelbourne.corr(), cmap='RdBu_r', annot=True, mask=MatrizSup)
plt.show()
```



Ya que tenemos este gráfico, el color nos ayuda a ubicar con mayor rapidez y facilidad aquellas variables con una correlación más alta, ya que

estas no nos ayudarían para seguir con nuestro análisis por lo que en el siguiente paso se eliminarían.

5) *Elección de variables*

En este paso eliminaremos las variables con base en el análisis correlacional del paso anterior, para ello utilizamos la función *drop* para eliminar la variable *Bedroom2* ya que el coeficiente de correlación fue de 0.94

```
DatosMelbourne.drop(columns=['Bedroom2'])
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertycount
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	1.0	1.0	202.0	NaN	NaN	Yarra	-37.79960	144.99840	Northern Metropolitan	4019.0
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	1.0	0.0	156.0	79.0	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4019.0
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	2.0	0.0	134.0	150.0	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4019.0
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	2.0	1.0	94.0	NaN	NaN	Yarra	-37.79690	144.99690	Northern Metropolitan	4019.0
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	1.0	2.0	120.0	142.0	2014.0	Yarra	-37.80720	144.99410	Northern Metropolitan	4019.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13575	Wheelers Hill	12 Strada Cr	4	h	1245000.0	S	Barry	26/08/2017	16.7	3150.0	2.0	2.0	652.0	NaN	1981.0	NaN	-37.90562	145.16761	South-Eastern Metropolitan	7392.0
13576	Williamstown	77 Merrett Dr	3	h	1031000.0	SP	Williams	26/08/2017	6.8	3016.0	2.0	2.0	333.0	133.0	1995.0	NaN	-37.85927	144.87904	Western Metropolitan	6380.0
13577	Williamstown	83 Power St	3	h	1170000.0	S	Raine	26/08/2017	6.8	3016.0	2.0	4.0	436.0	NaN	1997.0	NaN	-37.85274	144.88738	Western Metropolitan	6380.0
13578	Williamstown	96 Verdon St	4	h	2500000.0	PI	Sweeney	26/08/2017	6.8	3016.0	1.0	5.0	866.0	157.0	1920.0	NaN	-37.85908	144.89299	Western Metropolitan	6380.0
13579	Yarraville	6 Agnes St	4	h	1285000.0	SP	Village	26/08/2017	6.3	3013.0	1.0	1.0	362.0	112.0	1920.0	NaN	-37.81188	144.88449	Western Metropolitan	6543.0

13580 rows × 20 columns

Otra opción para eliminar variables seria tomar en cuenta el algoritmo a utilizar, por ejemplo, un árbol de decisión. En este caso eliminamos también las variables *Address, SellerG, Date, Postcode, CouncilArea, Lattitude, Longtitudem* y *Regionname*.

```
DatosMelbourne.drop(columns=['Address', 'SellerG', 'Date', 'Postcode', 'CouncilArea', 'Lattitude', 'Longtitude', 'Regionname'])
```

	Suburb	Rooms	Type	Price	Method	Distance	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Propertycount
0	Abbotsford	2	h	1480000.0	S	2.5	2.0	1.0	1.0	202.0	NaN	NaN	4019.0
1	Abbotsford	2	h	1035000.0	S	2.5	2.0	1.0	0.0	156.0	79.0	1900.0	4019.0
2	Abbotsford	3	h	1465000.0	SP	2.5	3.0	2.0	0.0	134.0	150.0	1900.0	4019.0
3	Abbotsford	3	h	850000.0	PI	2.5	3.0	2.0	1.0	94.0	NaN	NaN	4019.0
4	Abbotsford	4	h	1600000.0	VB	2.5	3.0	1.0	2.0	120.0	142.0	2014.0	4019.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
13575	Wheelers Hill	4	h	1245000.0	S	16.7	4.0	2.0	2.0	652.0	NaN	1981.0	7392.0
13576	Williamstown	3	h	1031000.0	SP	6.8	3.0	2.0	2.0	333.0	133.0	1995.0	6380.0
13577	Williamstown	3	h	1170000.0	S	6.8	3.0	2.0	4.0	436.0	NaN	1997.0	6380.0
13578	Williamstown	4	h	2500000.0	PI	6.8	4.0	1.0	5.0	866.0	157.0	1920.0	6380.0
13579	Yarraville	4	h	1285000.0	SP	6.3	4.0	1.0	1.0	362.0	112.0	1920.0	6543.0

CONCLUSIÓN

En esta práctica tomamos algunos pasos del análisis exploratorio de datos de las practicas pasadas, pero pusimos énfasis en las correlaciones entre las variables del conjunto de datos de Melbourne Housing Snapshot para tomar decisiones acerca de cuáles variables deberían de ser eliminadas, también nos apalancamos de la generación de graficas de dispersión para encontrar patrones o para identificar e interpretar las relaciones entre los pares de variables, por ultimo mediante el mapa de calor identificamos la variable, en este caso solo fue una, que tenia un coeficiente de correlación muy cercano a uno y la eliminamos de nuestro *dataframe*, también eliminamos más variables como otra opción a seguir de acuerdo a un algoritmo en específico.