```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/drive/MyDrive/MELBOURNE_HOUSE_PRICES_LESS (1).csv')
```

+ Code + Text

```
df.head()
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Met
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Met
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Met
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Met

```
df.tail()
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode
63018	Roxburgh Park	3 Carr Pl	3	h	566000.0	S	Raine	31/03/2018	3064
63019	Roxburgh Park	9 Parker Ct	3	h	500000.0	S	Raine	31/03/2018	3064
63020	Roxburgh Park	5 Parkinson Wy	3	h	545000.0	S	Raine	31/03/2018	3064
63021	Thomastown	3/1 Travers St	3	u	NaN	PI	Barry	31/03/2018	3074

```
df.shape
```

```
(63023, 13)
```

```
df.columns
```

```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Postcode', 'Regionname', 'Propertycount', 'Distance',
       'CouncilArea'],
      dtype='object')
```

```
df.duplicated().sum()
```

```
2
```

```
df.isnull().sum()
```

```
Suburb              0
Address             0
Rooms               0
Type                0
Price           14590
Method              0
SellerG             0
Date                0
Postcode            0
Regionname          0
Propertycount       0
Distance            0
CouncilArea         0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63023 entries, 0 to 63022
```

```
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Suburb         63023 non-null  object 
 1   Address        63023 non-null  object 
 2   Rooms          63023 non-null  int64  
 3   Type           63023 non-null  object 
 4   Price          48433 non-null  float64
 5   Method         63023 non-null  object 
 6   SellerG        63023 non-null  object 
 7   Date           63023 non-null  object 
 8   Postcode       63023 non-null  int64  
 9   Regionname     63023 non-null  object 
 10  Propertycount  63023 non-null  int64  
 11  Distance       63023 non-null  float64
 12  CouncilArea    63023 non-null  object 
dtypes: float64(2), int64(3), object(8)
memory usage: 6.3+ MB
```

```
df.describe()
```

	Rooms	Price	Postcode	Propertycount	Distance
count	63023.000000	4.843300e+04	63023.000000	63023.000000	63023.000000
mean	3.110595	9.978982e+05	3125.673897	7617.728131	12.684829
std	0.957551	5.934989e+05	125.626877	4424.423167	7.592015
min	1.000000	8.500000e+04	3000.000000	39.000000	0.000000
25%	3.000000	6.200000e+05	3056.000000	4380.000000	7.000000
50%	3.000000	8.300000e+05	3107.000000	6795.000000	11.400000
75%	4.000000	1.220000e+06	3163.000000	10412.000000	16.700000
max	31.000000	1.120000e+07	3980.000000	21650.000000	64.100000

```
df.nunique()
```

```
Suburb            380
Address         57754
Rooms              14
Type                3
Price            3417
Method              9
SellerG           476
Date              112
Postcode          225
Regionname          8
Propertycount     368
Distance          180
CouncilArea        34
dtype: int64
```

```
cols_to_fill_zero = ['Price']
df[cols_to_fill_zero] = df[cols_to_fill_zero].fillna(0)
df.isna().sum()
```

```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Postcode         0
Regionname       0
Propertycount    0
Distance         0
CouncilArea      0
dtype: int64
```

```
df = pd.get_dummies(df, drop_first = True)
```

```
df.head()
```

	Rooms	Price	Postcode	Propertycount	Distance	Suburb_Aberfeldie	Suburb_Ai
0	3	1490000.0	3067	4019	3.0	0	
1	3	1220000.0	3067	4019	3.0	0	
2	3	1420000.0	3067	4019	3.0	0	
3	3	1515000.0	3040	1543	7.5	1	
4	2	670000.0	3042	3464	10.4	0	

5 rows × 58773 columns

```
X = df.drop('Price', axis=1)
y = df['Price']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-e46c90e64743> in <cell line: 1>()
----> 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state = 2)

NameError: name 'train_test_split' is not defined
```

SEARCH STACK OVERFLOW

```
from sklearn.linear_model import LinearRegression



reg = LinearRegression().fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-e7994597f7a6> in <cell line: 1>()
----> 1 reg = LinearRegression().fit(X_train, y_train)

NameError: name 'X_train' is not defined
```

SEARCH STACK OVERFLOW

```
reg.score(X_test, y_test)


reg.score(X_train, y_train)


from sklearn import linear_model
lasso_reg = linear_model.Lasso(alpha = 50, max_iter=100, tol=0.1)
lasso_reg.fit(X_train, y_train)


lasso_reg.score(X_test, y_test)


lasso_reg.score(X_train, y_train)


from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha = 50, max_iter=100, tol=0.1)
ridge_reg.fit(X_train, y_train)


ridge_reg.score(X_test, y_test)


ridge_reg.score(X_train, y_train)
```