Actividad | Regresion Multiple

Integrantes

Alumno: **Erick de Jesus Hernández Cerecedo**
Matricula: **A01066428**

Información del Curso

Nombre: **Ciencia y analítica de datos**
Profesor: **Jobish Vallikavungal Devassia**

Fechas: **Martes 9 de noviembre de 2022**

In [2]:
```
# Importacion de librerias
import numpy as np
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Modelo Linel
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge, Lasso

# Metricas
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Split train & test
from sklearn.model_selection import train_test_split

# Transformaciones
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler

import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    # Legacy Python that doesn't verify HTTPS certificates by default
    pass
else:
    # Handle target environment that doesn't support HTTPS verification
    ssl._create_default_https_context = _create_unverified_https_context
```

Ejercicio 2. Regresión múltiple.

In [3]:
```
# Lectura de los datos
df = pd.read_csv('https://raw.githubusercontent.com/marypazrf/bdd/main/kc_house_data.csv
df.sample(10)
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wat

17545	3905100840	20140723T000000	500000.0	3	2.25	1580	4379	2.0
17872	7922710450	20150327T000000	731000.0	5	2.50	3670	8960	1.5
11003	3243200310	20140520T000000	300000.0	3	1.00	2120	7735	1.0
140	4232901525	20140627T000000	665000.0	2	1.00	1110	3200	1.0
15320	321059132	20150427T000000	365000.0	3	1.75	1450	61419	1.0
4539	3223069118	20140616T000000	554000.0	3	3.50	3380	108900	2.0
16590	8635750950	20140607T000000	568500.0	4	2.50	2460	4200	2.0
12465	7812801125	20150112T000000	222900.0	2	1.00	1110	6411	1.0
3984	7883607645	20140602T000000	155000.0	1	1.00	720	6000	1.0
504	8906200070	20150210T000000	280000.0	3	1.50	1670	11610	1.0

10 rows × 21 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	flo
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500

75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.0000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.5000

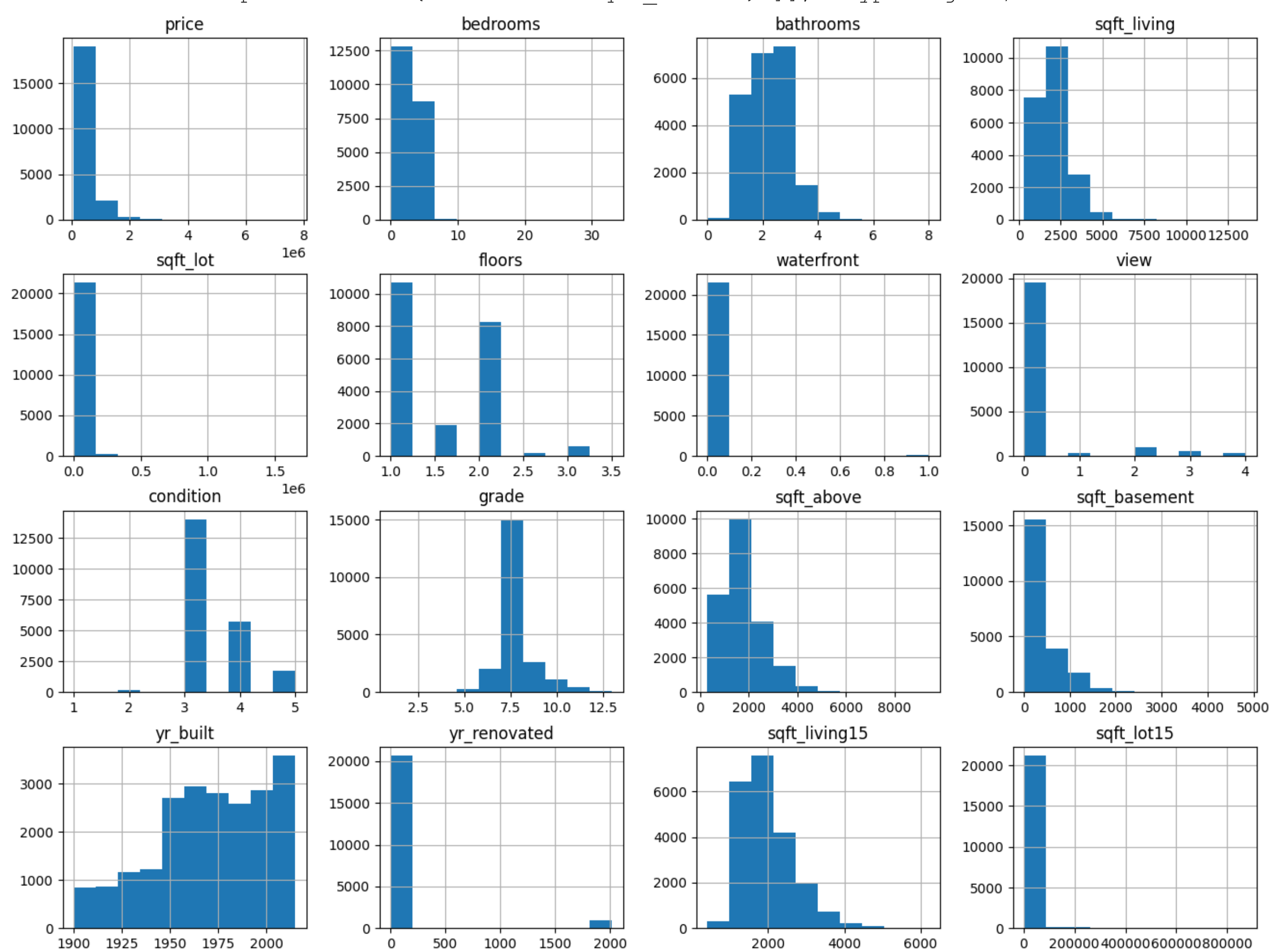
In [6]:

```
df.drop('id', axis = 1, inplace = True)
df.drop('date', axis = 1, inplace = True)
df.drop('zipcode', axis = 1, inplace = True)
df.drop('lat', axis = 1, inplace = True)
df.drop('long', axis = 1, inplace = True)

df.hist(figsize=(16, 12))
```

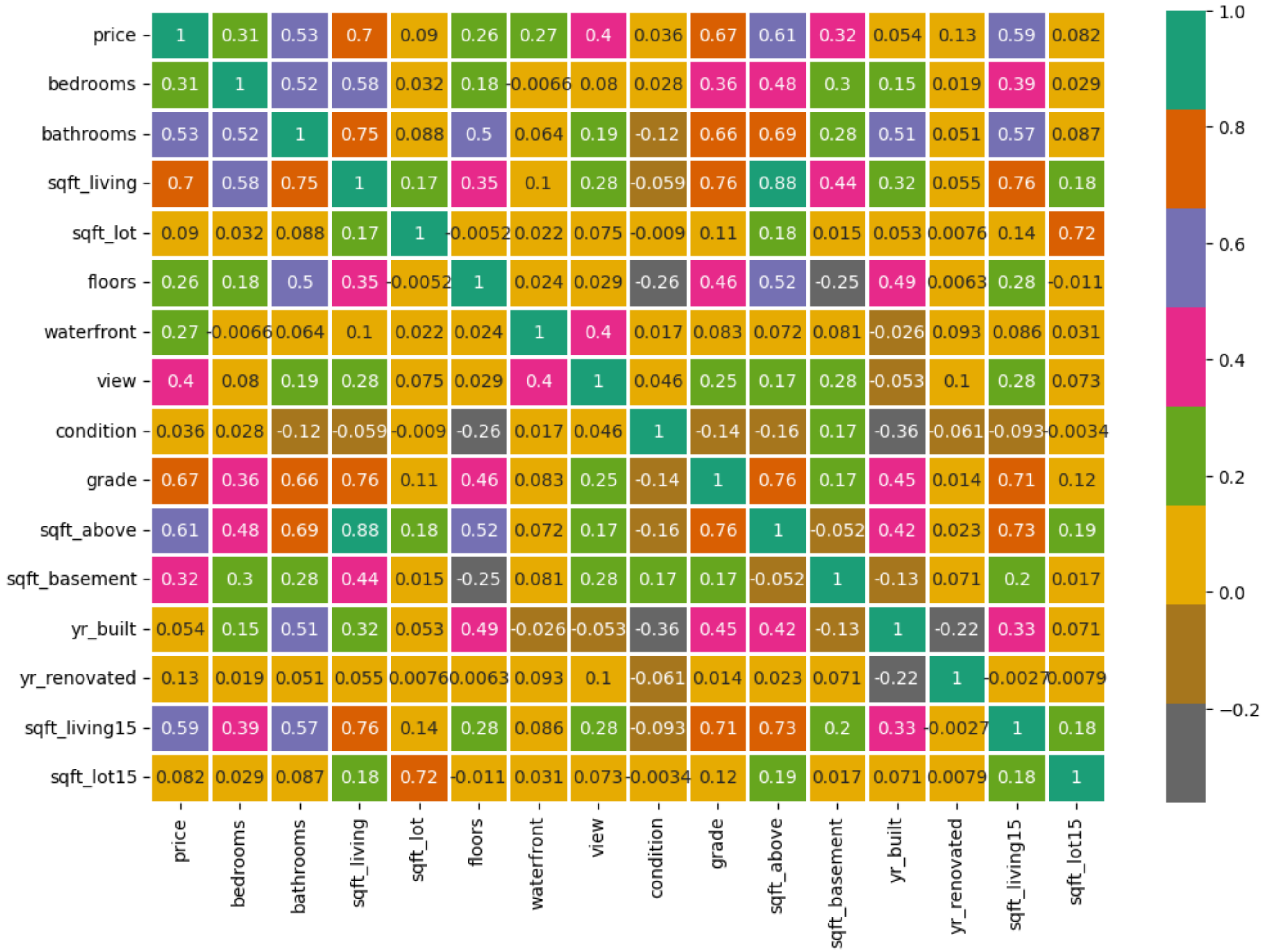
Out[6]:

```
array([[<AxesSubplot: title={'center': 'price'}>,
        <AxesSubplot: title={'center': 'bedrooms'}>,
        <AxesSubplot: title={'center': 'bathrooms'}>,
        <AxesSubplot: title={'center': 'sqft_living'}>],
       [<AxesSubplot: title={'center': 'sqft_lot'}>,
        <AxesSubplot: title={'center': 'floors'}>,
        <AxesSubplot: title={'center': 'waterfront'}>,
        <AxesSubplot: title={'center': 'view'}>],
       [<AxesSubplot: title={'center': 'condition'}>,
        <AxesSubplot: title={'center': 'grade'}>,
        <AxesSubplot: title={'center': 'sqft_above'}>,
        <AxesSubplot: title={'center': 'sqft_basement'}>],
       [<AxesSubplot: title={'center': 'yr_built'}>,
        <AxesSubplot: title={'center': 'yr_renovated'}>,
        <AxesSubplot: title={'center': 'sqft_living15'}>,
        <AxesSubplot: title={'center': 'sqft_lot15'}>]], dtype=object)
```



In [7]:

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='Dark2_r', linewidths = 2)
plt.show()
```



In [8]:
```python
columns = df.columns.drop('price')

features = columns
label = ['price']

X = df[features]
y = df[label]
```

1. Divide los datos. Utiliza la función train_test_split (ya esta en el notebook).

In [9]:
```python
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state

print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 2162
number of training samples: 19451
```

1. Regresión Múltiple Lineal.

- Realiza la regresión lineal: modelo generado (ecuación), sus errores y r cuadrada.

In [10]:
```python
# Instancia del modelo
RM = LinearRegression()

# Entrenamiento del modelo
RM.fit(x_train,y_train)
```

```python
# Generamos prediccion de datos
yhat_RML = RM.predict(x_test)

m = RM.coef_[0]
b = RM.intercept_[0]

print("Ecuacion de la Recta:")
# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficientes: ' +  str(m))
# Este es el valor donde corta el eje Y (en X=0)
print('Termino independiente (b): %.2f' % b)


print("\nMetricas:")
# Error Cuadrado Medio
print("Mean Squared Error (MSE): %.2f" % mean_squared_error(y_test, yhat_RML))
# Raiz del Error Cuadrado Medio
print("Root Mean Squared Error (RMSE): %.2f" % np.sqrt(mean_squared_error(y_test, yhat_R
# Median Absolut Error
print("Median Absolut Error (MAE): %.2f" % mean_absolute_error(y_test, yhat_RML))
# R2 Square
print("R2 Score: %.2f" % r2_score(y_test, yhat_RML))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Varianza score: %.2f' % r2_score(y_test, yhat_RML))
```

```
Ecuacion de la Recta:
Coefficientes: [-3.82008048e+04  4.14661380e+04  1.07992584e+02  1.71356997e-02
  3.16916913e+04  5.52691023e+05  4.12493228e+04  2.12221443e+04
  1.19493216e+05  4.77750271e+01  6.02175565e+01 -3.55090216e+03
  1.32602215e+01  2.90059284e+01 -5.48132603e-01]
Termino independiente (b): 6151359.26

Metricas:
Mean Squared Error (MSE): 53885900364.49
Root Mean Squared Error (RMSE): 232133.37
Median Absolut Error (MAE): 137480.14
R2 Score: 0.66
Varianza score: 0.66
```

1. Regresión Múltiple Polinomial.

- Realiza la regresión polinomial completa, tu modelo generado (ecuación), sus errores y r cuadrada.

In [11]:

```python
# Generamos instancia del Caracteristicas Polinominales
PF = PolynomialFeatures(degree=2)
# Transformamos los datos a este formato
x_train_pr = PF.fit_transform(x_train)

# Instancia del modelo lineal
LR = LinearRegression()
# Entrenamiento del modelo
LR.fit(x_train_pr, y_train)

# Transformacion de los datos de prueba a formato polinominal
x_test_pr = PF.fit_transform(x_test)
yhat_PR = LR.predict(x_test_pr)

m = RM.coef_[0]
b = RM.intercept_[0]

print("Ecuacion de la Recta:")
# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficientes: ' +  str(m))
# Este es el valor donde corta el eje Y (en X=0)
```

```
print('Termino independiente (b): %.2f' % b)


print("\nMetricas:")
# Error Cuadrado Medio
print("Mean Squared Error (MSE): %.2f" % mean_squared_error(y_test, yhat_PR))
# Raiz del Error Cuadrado Medio
print("Root Mean Squared Error (RMSE): %.2f" % np.sqrt(mean_squared_error(y_test, yhat_P
# Median Absolut Error
print("Median Absolut Error (MAE): %.2f" % mean_absolute_error(y_test, yhat_PR))
# R2 Square
print("R2 Score: %.2f" % r2_score(y_test, yhat_PR))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Varianza score: %.2f' % r2_score(y_test, yhat_PR))
```

```
Ecuacion de la Recta:
Coefficientes: [-3.82008048e+04  4.14661380e+04  1.07992584e+02  1.71356997e-02
  3.16916913e+04  5.52691023e+05  4.12493228e+04  2.12221443e+04
  1.19493216e+05  4.77750271e+01  6.02175565e+01 -3.55090216e+03
  1.32602215e+01  2.90059284e+01 -5.48132603e-01]
Termino independiente (b): 6151359.26

Metricas:
Mean Squared Error (MSE): 34694009583.54
Root Mean Squared Error (RMSE): 186263.28
Median Absolut Error (MAE): 121313.26
R2 Score: 0.78
Varianza score: 0.78
```

1. Realiza la regresión con Ridge y Lasso. Incluye la ecuación de tu modelo, errores y r cuadrada.

In [12]:

```
# LASSO

# Definicion del pipeline para las transformaciones
LASSO = Pipeline(
    [
        ('polinomial', PolynomialFeatures(degree=2)),
        ('scaler', StandardScaler()),
        ('lasso', Lasso())
    ]
)



# Transformaciones y entrenamiento
LASSO.fit(x_train, y_train)

# Generamos predicciones
yhat_LASSO = LASSO.predict(x_test)

m_c = LASSO["lasso"].coef_
b_c = LASSO["lasso"].intercept_[0]


print("Informacion de la Recta:")
# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficientes: ' + str(m_c))
# Este es el valor donde corta el eje Y (en X=0)
print('Termino independiente (b): %.2f' % b_c)
```

```
Informacion de la Recta:
Coefficientes: [ 0.00000000e+00  1.15791522e+05  1.18277492e+05  7.14988943e+04
 -1.22931545e+05 -2.41032416e+04  2.94602341e+04  5.53200952e+04
  8.65826016e+04  2.11494845e+05 -1.49809440e+05 -2.50762863e+04
 -1.11450631e+05 -9.83144833e+04  9.07769041e+04 -4.13288781e+04
  1.10114497e+04 -1.28814036e+04 -1.00661961e+05 -2.60904835e+04
```

```
  1.98946416e+04 -6.70740535e+03 -1.95757670e+02 -6.81560972e+03
 -3.60633389e+04  3.11710654e+04  2.47712973e+02 -1.14175892e+05
  1.67191121e+01  6.86087185e+04  2.53674285e+04  3.93767099e+03
  1.83419521e+05 -2.22655215e+03 -5.54912880e+04  1.34149812e+04
  1.28796456e+04 -5.51455026e+04  1.48229954e+05 -4.48458877e+04
 -4.92215091e+04 -1.06129591e+05 -2.71824287e+04 -9.96248691e+04
  1.05009131e+04  1.04616963e+05 -2.32965850e+04  8.81745039e+03
 -2.85440906e+04  8.16178600e+04 -3.92336629e+04  2.27972480e+05
 -2.37169682e+05  4.31254172e+04 -9.66627700e+04 -3.33084540e+02
  5.76440623e+03 -4.48039776e+04 -3.89394418e+02  3.17248859e+04
 -9.29180792e+02 -6.48367162e+03  1.07076133e+04  8.27899889e+04
 -4.56055816e+04 -8.65008286e+03  8.21916192e+04 -5.96268927e+03
  4.12915379e+03  2.04371789e+04  4.98130723e+04 -1.49311042e+04
  1.48794026e+04  1.68491812e+04  3.42439734e+04 -9.06174070e+03
  1.66729366e+04  1.95389243e+04 -1.95453802e+04 -1.01034397e+05
 -2.09882634e+04 -3.49124661e+04 -4.91917166e+03 -4.31345828e+03
 -8.22573435e+04  9.93703156e+04  2.32421101e+04  4.35710995e+04
 -6.28381543e+03  3.56303899e+04 -3.15398983e+03  1.75007872e+04
  9.86115735e+03  1.04515156e+05 -1.15099934e+05 -4.18222578e+04
 -1.50258539e+05 -8.32403792e+03  2.59624442e+04 -3.81909355e+03
  6.75926897e+03 -1.40815735e+04  7.92878230e+04  6.92242211e+04
 -1.43965391e+05 -3.25458085e+04  1.79887355e+05 -5.60724783e+04
  1.17646598e+05  2.24792122e+04  7.13176446e+04 -2.44645156e+05
  5.24708297e+03 -2.04281922e+05 -1.15564025e+05  1.78569786e+05
  2.18660647e+04 -6.76386022e+03  2.59491800e+04  6.41292532e+03
  3.58394229e+04 -7.15559445e+04 -9.15144011e+04  1.02164074e+04
  8.00134478e+03 -4.39279575e+03  7.77205021e+04  7.48211835e+04
 -1.15794232e+05  1.48899459e+05  1.54919558e+04  7.00974557e+04
 -4.25252385e+02  9.90649252e+04  1.87769905e+04  2.08028926e+04]
Termino independiente (b): 539150.74
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.305e+14, tolerance: 2.572e+11
  model = cd_fast.enet_coordinate_descent(
```

In [13]:

```python
print("\nMetricas:")
# Error Cuadrado Medio
print("Mean Squared Error (MSE): %.2f" % mean_squared_error(y_test, yhat_LASSO))
# Raiz del Error Cuadrado Medio
print("Root Mean Squared Error (RMSE): %.2f" % np.sqrt(mean_squared_error(y_test, yhat_L
# Median Absolut Error
print("Median Absolut Error (MAE): %.2f" % mean_absolute_error(y_test, yhat_LASSO))
# R2 Square
print("R2 Score: %.2f" % r2_score(y_test, yhat_LASSO))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Varianza score: %.2f' % r2_score(y_test, yhat_LASSO))
```

```
Metricas:
Mean Squared Error (MSE): 35317680177.38
Root Mean Squared Error (RMSE): 187929.99
Median Absolut Error (MAE): 122444.43
R2 Score: 0.78
Varianza score: 0.78
```

In [14]:

```python
# RIDGE

# Pipeline para transformacion
RIDGE = Pipeline(
    [
        ('polinomial', PolynomialFeatures(degree=2)),
        ('scaler', StandardScaler()),
        ('ridge', Ridge())
    ]
)
```

```python
# Transformacion y entrenamiento
RIDGE.fit(x_train, y_train)

# Generamos predicciones
yhat_RIDGE = RIDGE.predict(x_test)

m_c = RIDGE["ridge"].coef_
b_c = RIDGE["ridge"].intercept_[0]

print("Informacion de la Recta:")
# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficientes: ' + str(m_c))
# Este es el valor donde corta el eje Y (en X=0)
print('Termino independiente (b): %.2f' % b_c)
```

```
Informacion de la Recta:
Coefficientes: [[ 0.00000000e+00  3.60460235e+05 -2.77291112e+05  6.01060372e+04
  -4.99499954e+05 -2.47117323e+05 -1.81348511e+05 -2.00154965e+04
   4.08285923e+05  4.54080960e+05  1.97894841e+04  8.74180919e+04
  -5.55941261e+05 -3.42496199e+05  8.80104706e+05 -4.12867420e+05
   1.03210540e+04  8.14910656e+03 -3.18928488e+04 -1.73919745e+04
   1.84084814e+04 -5.81514660e+03  5.68001517e+02 -7.52269166e+03
  -5.21301284e+04 -2.28083968e+04 -3.23321387e+04 -3.54836567e+05
  -4.48831658e+03  6.38078515e+04  2.21914474e+04 -1.54380758e+04
   5.61512313e+04 -6.70933117e+03 -4.72544279e+04  1.20504877e+04
   9.64710097e+03 -2.90941942e+04  1.53966643e+05  6.51612229e+04
   1.35423065e+03  2.54287499e+05 -2.24257131e+04 -7.80877048e+04
   3.22598570e+03  6.69132209e+03 -3.57207513e+04  2.40777018e+03
   3.94256660e+04 -2.81955328e+04  2.40715163e+04  1.71696282e+05
   8.20435587e+03 -4.86638352e+02 -1.78445705e+05  1.33809967e+04
   8.17728068e+03  1.29770171e+03  3.27735286e+03  2.96567438e+04
  -7.68215107e+02 -6.24093825e+03  9.79148615e+03  6.72246568e+04
  -3.83792651e+04 -5.88224929e+03  4.67792302e+05 -4.91490482e+03
   1.05546773e+04  1.72986151e+04  4.69515508e+04 -1.70924335e+04
   1.65788507e+04  1.25161441e+04 -1.03993341e+04 -1.42769362e+03
   1.33082124e+04  2.68094639e+05 -1.39927122e+04 -7.75956209e+04
  -2.02004439e+04 -1.81348511e+05 -6.88790438e+03 -4.12041687e+02
  -1.09873707e+05  4.99246424e+04  2.36442281e+03  4.20054720e+05
  -4.20689300e+03  4.04076894e+04 -3.01127711e+03  1.85730913e+04
   1.41086002e+04  1.26792262e+05 -3.39477678e+04 -7.03141357e+03
  -9.24663783e+04 -7.67256847e+03  1.80453138e+04 -2.95642641e+03
   4.02952191e+03 -1.73152323e+04  1.53404187e+04  2.25251839e+04
  -4.49278293e+05 -3.33895463e+04  1.63105691e+05 -3.89747605e+04
   1.18759293e+05  1.56622159e+05  8.58235366e+04 -4.83906362e+05
  -7.27666301e+03 -2.48754680e+05 -1.18267982e+05  2.45925711e+03
   2.54092100e+04 -1.25681548e+05  1.45794491e+04  5.75484675e+03
   4.58822465e+03 -4.28825793e+04 -1.35752947e+05  6.01597633e+03
   9.83915940e+03 -1.24115275e+04  5.92464674e+05  7.74244130e+04
  -9.01391323e+05  5.11250244e+05  2.72594602e+05  6.16419089e+04
   1.25733178e+03  1.08319801e+05  1.37484991e+04  2.16284361e+04]]
Termino independiente (b): 539150.74
```

In [15]:

```python
print("\nMetricas:")
# Error Cuadrado Medio
print("Mean Squared Error (MSE): %.2f" % mean_squared_error(y_test, yhat_RIDGE))
# Raiz del Error Cuadrado Medio
print("Root Mean Squared Error (RMSE): %.2f" % np.sqrt(mean_squared_error(y_test, yhat_R
# Median Absolut Error
print("Median Absolut Error (MAE): %.2f" % mean_absolute_error(y_test, yhat_RIDGE))
# R2 Square
print("R2 Score: %.2f" % r2_score(y_test, yhat_RIDGE))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Varianza score: %.2f' % r2_score(y_test, yhat_RIDGE))
```

```
Metricas:
```

```
Mean Squared Error (MSE): 34992695922.08
Root Mean Squared Error (RMSE): 187063.35
Median Absolut Error (MAE): 121523.11
R2 Score: 0.78
Varianza score: 0.78
```

1. Finalmente gráfica :

- MAE (de los cuatro métodos)
- R2 (de los cuatro métodos)

Explica tus resultados, que método se aproxima mejor, ¿por qué?, ¿qué porcentajes de entrenamiento y evaluación usaste? ¿Que error tienes?, ¿es bueno?, ¿Cómo lo sabes? Agrega las conclusiones

In [16]:

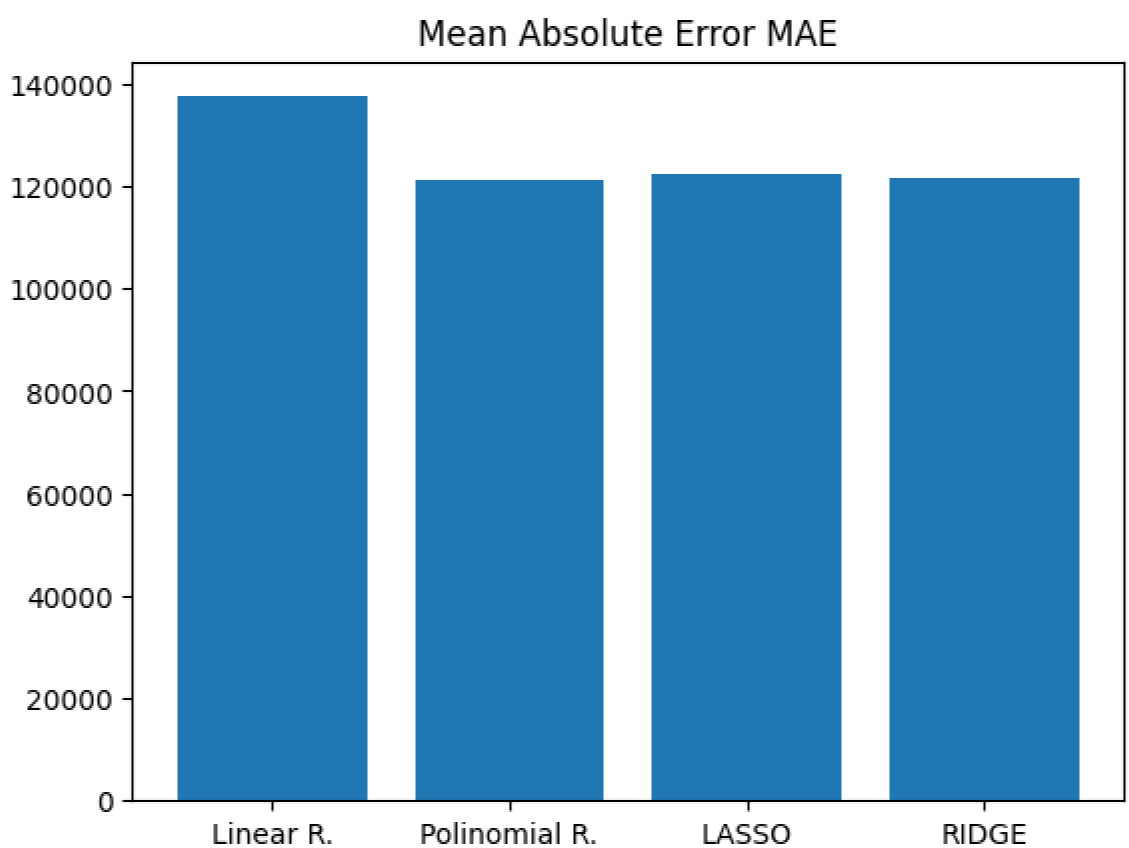
```python
# Nombres de los modelos
nombres = ['Linear R.', 'Polinomial R.', 'LASSO', 'RIDGE']

# Valores de error MAE
LR_MAE = mean_absolute_error(y_test, yhat_RML)
PR_MAE = mean_absolute_error(y_test, yhat_PR)
LASSO_MAE = mean_absolute_error(y_test, yhat_LASSO)
RIDGE_MAE = mean_absolute_error(y_test, yhat_RIDGE)

plt.title("Mean Absolute Error MAE")
plt.bar(nombres, [LR_MAE, PR_MAE, LASSO_MAE, RIDGE_MAE])

# Impresion de valores
print('MAE Regresión Lineal:', LR_MAE, '\nMAE Regresión Polinomial:', PR_MAE, '\nMAE Las
```

```
MAE Regresión Lineal: 137480.13882731108
MAE Regresión Polinomial: 121313.26309545722
MAE Lasso: 122444.42902741059
MAE Ridge: 121523.10703205499
```



In [17]:

```python
# Valores de error R2
LR_R2 = r2_score(y_test, yhat_RML)
```

```python
PR_R2 = r2_score(y_test, yhat_PR)
LASSO_R2 = r2_score(y_test, yhat_LASSO)
RIDGE_R2 = r2_score(y_test, yhat_RIDGE)

plt.title("R2 Scored")
plt.bar(nombres, [LR_R2, PR_R2, LASSO_R2, RIDGE_R2])

print('R2 Regresión Lineal:', LR_R2, '\nR2 Regresión Polinomial:', PR_R2, '\nR2 Lasso:',
```

```
R2 Regresión Lineal: 0.6579723205007854
R2 Regresión Polinomial: 0.7797881911573368
R2 Lasso: 0.7758295933693782
R2 Ridge: 0.777892352086667
```