Steps_Apply_Linear

May 12, 2020

0.0.1 About the Data Set

Your neighbor is a real estate agent and wants some help predicting housing prices for regions in the USA. It would be great if you could somehow create a model for her that allows her to put in a few features of a house and returns back an estimate of what the house would sell for.

She has asked you if you could help her out with your new data science skills. You say yes, and decide that Linear Regression might be a good path to solve this problem!

Your neighbor then gives you some information about a bunch of houses in regions of the United States,it is all in the data set: USA_Housing.csv.

The data contains the following columns:

- 'Avg. Area Income': Avg. Income of residents of the city house is located in.
- 'Avg. Area House Age': Avg Age of Houses in same city
- 'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city
- 'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city
- 'Area Population': Population of city house is located in
- 'Price': Price that the house sold at
- 'Address': Address for the house

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
USAhousing = pd.read_csv('USA_Housing.csv')
```

[3]:
```
USAhousing.head()
```

[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
1	79248.642455	6.002900	6.730821			
2	61287.067179	5.865890	8.512727			
3	63345.240046	7.188236	5.586729			
4	59982.197226	5.040555	7.839388			

```
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05

                                                  Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1  188 Johnson Views Suite 079\nLake Kathleen, CA…
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                          USS Barnett\nFPO AP 44820
4                         USNS Raymond\nFPO AE 09386
```

[4]:
```
USAhousing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income                5000 non-null float64
Avg. Area House Age             5000 non-null float64
Avg. Area Number of Rooms       5000 non-null float64
Avg. Area Number of Bedrooms    5000 non-null float64
Area Population                 5000 non-null float64
Price                           5000 non-null float64
Address                         5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.5+ KB
```

[5]:
```
USAhousing.describe()
```

[5]:
```
        Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
count        5000.000000          5000.000000                5000.000000
mean        68583.108984             5.977222                   6.987792
std         10657.991214             0.991456                   1.005833
min         17796.631190             2.644304                   3.236194
25%         61480.562388             5.322283                   6.299250
50%         68804.286404             5.970429                   7.002902
75%         75783.338666             6.650808                   7.665871
max        107701.748378             9.519088                  10.759588

        Avg. Area Number of Bedrooms  Area Population         Price
count                    5000.000000      5000.000000  5.000000e+03
mean                        3.981330     36163.516039  1.232073e+06
std                         1.234137      9925.650114  3.531176e+05
min                         2.000000       172.610686  1.593866e+04
25%                         3.140000     29403.928702  9.975771e+05
50%                         4.050000     36199.406689  1.232669e+06
75%                         4.490000     42861.290769  1.471210e+06
```

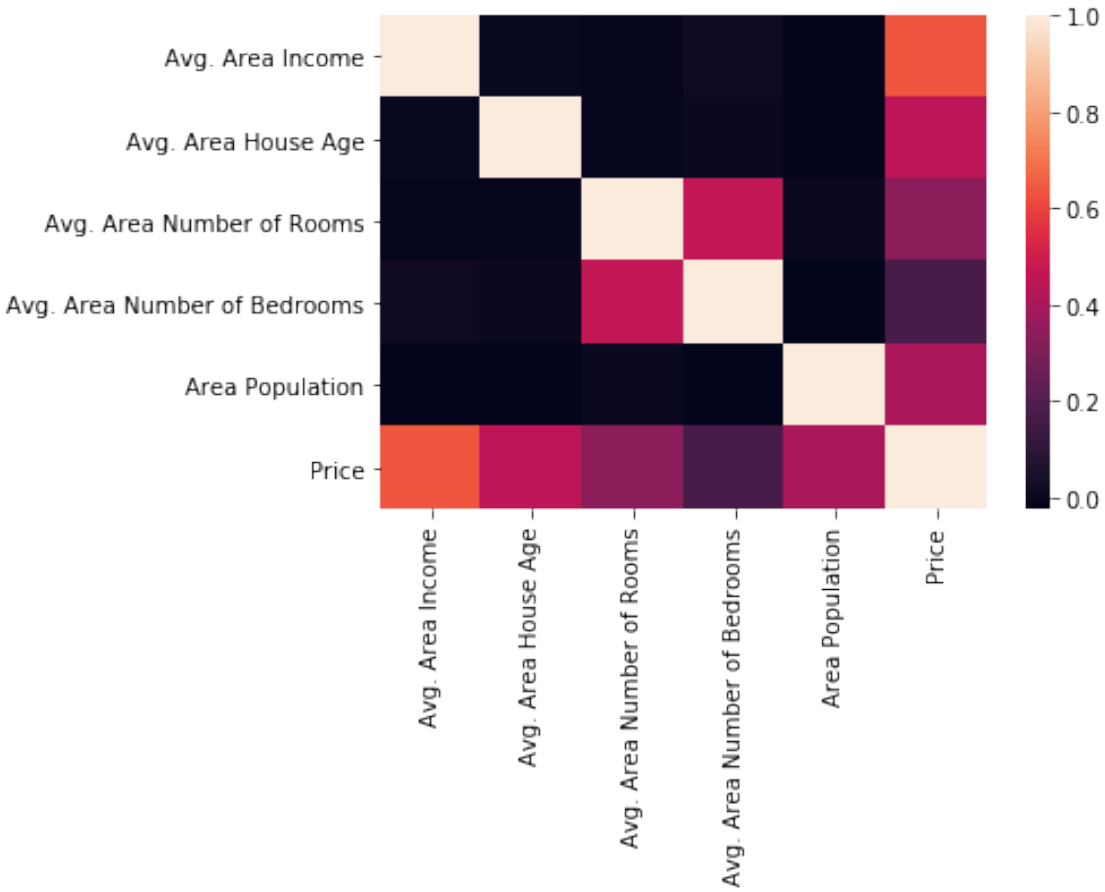
```
max                          6.500000      69621.713378   2.469066e+06
```

[6]:
```
USAhousing.columns
```

[6]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

[7]:
```
sns.heatmap(USAhousing.corr())
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fc43679c88>
```



[8]:
```
X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of␣
↪Rooms',
                'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
↪random_state=101)
```

1 Actual Starts with Linear Regression

[11]:
```
from sklearn.linear_model import LinearRegression
```

[12]:
```
lm = LinearRegression()
```

[13]:
```
lm.fit(X_train,y_train)
```

[13]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

the point at which a given line cuts a coordinate axis; the value of the coordinate at that point. $\mathbf{y = mx + b}$

- b=Intercept
- m=constant or CoEfficient
- x=dependent

[14]:
```
print(lm.intercept_)
```

```
-2640159.796851911
```

[16]:
```
print(lm.coef_)
```

```
[2.15282755e+01 1.64883282e+05 1.22368678e+05 2.23380186e+03
 1.51504200e+01]
```

Co-Efficient

A number used to multiply a variable. Example: 6z means 6 times z, and "z" is a variable, so 6 is a coefficient. Variables with no number have a coefficient of 1.

[17]:
```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

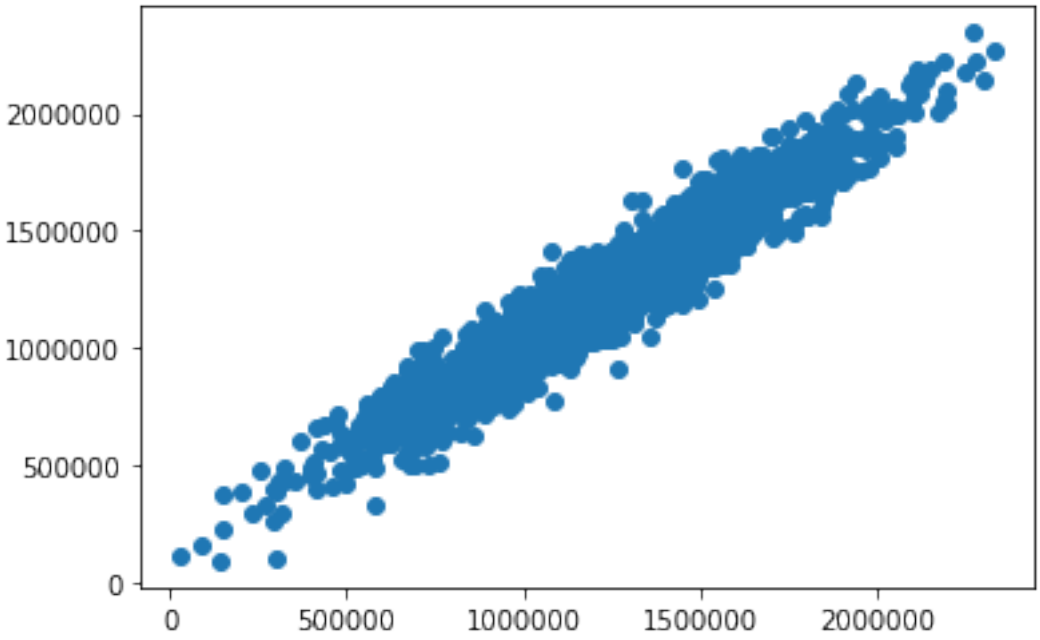
[17]:

	Coefficient
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

[18]:
```
predictions = lm.predict(X_test)
```

[19]:
```
plt.scatter(y_test,predictions)
```

[19]: <matplotlib.collections.PathCollection at 0x1fc443e0ba8>



Description about Evaluation Metrics *Absolute Error = Actual value - predicted value* **MAE = (Absolute Error 1 + Absolute Error 2 + Absolute Error 3 + Absolute Error 4) / n**

1.1 Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

[21]:
```
from sklearn import metrics
```

[22]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 82288.22251914957
MSE: 10460958907.209501
RMSE: 102278.82922291153
```

2 Only Steps