Deloitte Intern

May 21, 2022

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# stats models for good measure
import statsmodels.formula.api as smf

# SciKit Learn packages
from sklearn.model_selection import train_test_split # for splitting data
from sklearn.preprocessing import StandardScaler     # feature scaling
from sklearn import metrics                          # for evaluation metrics

from sklearn.linear_model import LinearRegression    # linear regression
```

1 Load Data

[2]:
```
df = pd.read_csv('DB_Data_Final.csv')
df
```

[2]:

	70	70		70			
0	2019-12-31	87683.64	0.354	1986488.82		727.22	44.908
1	2019-11-30	85986.81	0.297	1961429.56		726.54	44.554
2	2019-10-31	85336.45	0.499	1945600.55		725.86	44.257
3	2019-09-30	84691.95	0.526	1952250.49		725.19	43.758
4	2019-08-31	84053.23	0.577	1935492.43		724.52	43.232
..	…	…	…	…	…	…	
115	2010-05-31	33318.19	0.477	663351.37		656.99	4.251
116	2010-04-30	32985.64	1.039	656561.22		655.56	3.774
117	2010-03-31	32656.70	0.699	650012.90		654.14	2.735
118	2010-02-28	32331.35	1.036	636072.26		652.73	2.036
119	2010-01-31	32009.52	1.357	625609.29		651.33	1.000

[120 rows x 6 columns]

[3]:
```
df.rename(columns={' ':'Date','70      ':'Average_salary',' ':␣
↪'House_pricing_index',
```

```
                  '    ':'Money_supply','70    ':'Average_population'},␣
↪inplace = True)

df
```

[3]:
```
           Date  Average_salary  70         Money_supply  \
0    2019-12-31        87683.64           0.354    1986488.82
1    2019-11-30        85986.81           0.297    1961429.56
2    2019-10-31        85336.45           0.499    1945600.55
3    2019-09-30        84691.95           0.526    1952250.49
4    2019-08-31        84053.23           0.577    1935492.43
..          …              …              …             …
115  2010-05-31        33318.19           0.477     663351.37
116  2010-04-30        32985.64           1.039     656561.22
117  2010-03-31        32656.70           0.699     650012.90
118  2010-02-28        32331.35           1.036     636072.26
119  2010-01-31        32009.52           1.357     625609.29

     Average_population  House_pricing_index
0                727.22               44.908
1                726.54               44.554
2                725.86               44.257
3                725.19               43.758
4                724.52               43.232
..                  …                  …
115              656.99                4.251
116              655.56                3.774
117              654.14                2.735
118              652.73                2.036
119              651.33                1.000

[120 rows x 6 columns]
```

[4]:
```
df.shape
```

[4]:
```
(120, 6)
```

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

[5]:
```
       Average_salary  70         Money_supply  Average_population  \
count      120.000000      120.000000  1.200000e+02          120.000000
mean     56853.864500        0.377208  1.272736e+06          693.589917
std      15769.135303        0.507795  4.060049e+05           19.440457
min      32009.520000       -1.154000  6.256093e+05          651.330000
25%      43301.252500        0.086500  9.234390e+05          679.542500
50%      55117.970000        0.395000  1.235543e+06          693.510000
75%      69535.425000        0.712250  1.629568e+06          709.750000
```

```
max      87683.640000         1.803000  1.986489e+06          727.220000
```

```
       House_pricing_index
count           120.000000
mean             18.145200
std              11.709229
min               1.000000
25%               8.562250
50%              14.393000
75%              26.387500
max              44.908000
```

2 Create dataframes for regression

[6]:
```
xdata = {'Average_salary': df['Average_salary'],
         'Money_supply': df['Money_supply'],
         'Average_population': df['Average_population']}

X = pd.DataFrame(xdata)
X.head()
```

[6]:

	Average_salary	Money_supply	Average_population
0	87683.64	1986488.82	727.22
1	85986.81	1961429.56	726.54
2	85336.45	1945600.55	725.86
3	84691.95	1952250.49	725.19
4	84053.23	1935492.43	724.52

[7]:
```
ydata = {'House_pricing_index': df['House_pricing_index']}
Y = pd.DataFrame(ydata)
Y
```

[7]:

	House_pricing_index
0	44.908
1	44.554
2	44.257
3	43.758
4	43.232
..	…
115	4.251
116	3.774
117	2.735
118	2.036
119	1.000

[120 rows x 1 columns]

[8]:
```
scaler = StandardScaler().fit(X)
X_scaled = scaler.transform(X)
X_scaled = pd.DataFrame(X_scaled)
X_scaled.rename(columns={0:'Average_salary',1:'Money_supply',2:␣
↪'Average_population'}, inplace = True)
X_scaled
```

[8]:

	Average_salary	Money_supply	Average_population
0	1.963268	1.765362	1.737155
1	1.855212	1.703382	1.702030
2	1.813797	1.664231	1.666905
3	1.772755	1.680679	1.632296
4	1.732080	1.639230	1.597687
..	…	…	…
115	-1.498773	-1.507222	-1.890561
116	-1.519950	-1.524017	-1.964428
117	-1.540897	-1.540213	-2.037778
118	-1.561616	-1.574693	-2.110611
119	-1.582110	-1.600572	-2.182928

[120 rows x 3 columns]

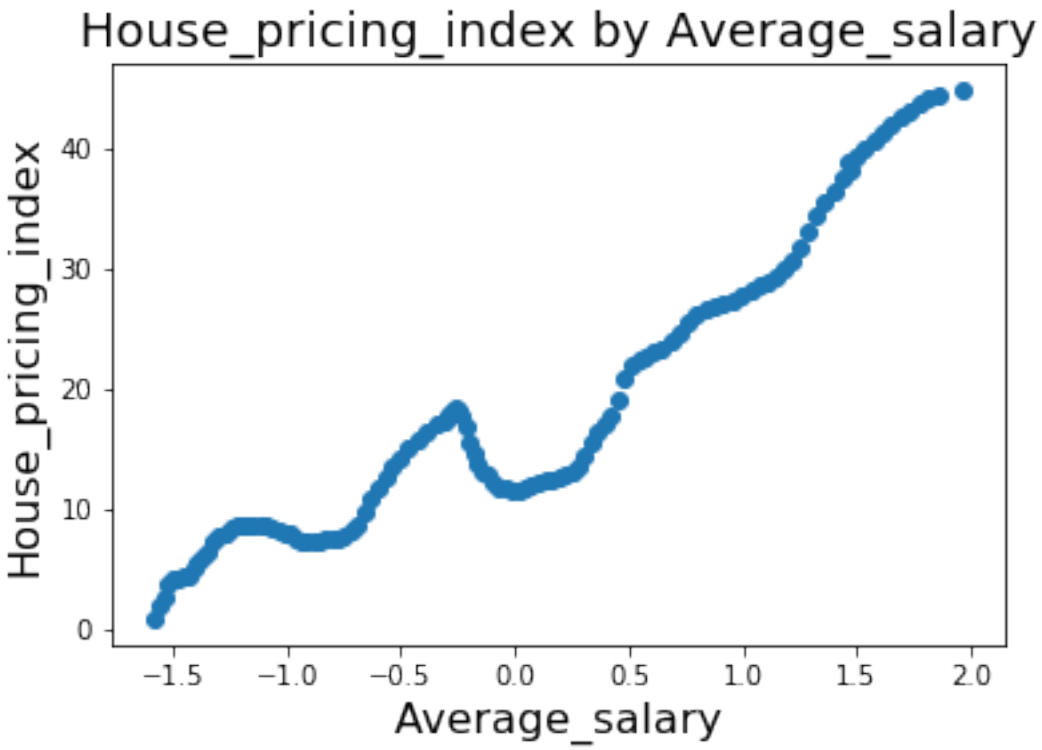
[9]:
```
df2 = pd.concat([X_scaled, Y], axis=1, join='inner')
df2
```

[9]:

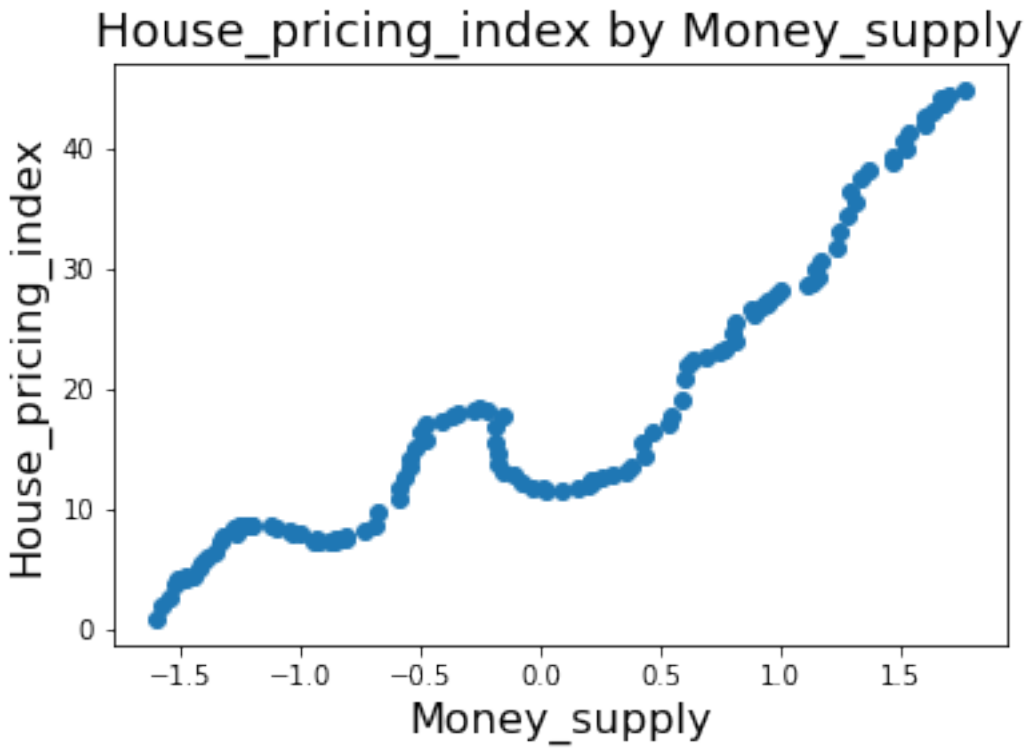
	Average_salary	Money_supply	Average_population	House_pricing_index
0	1.963268	1.765362	1.737155	44.908
1	1.855212	1.703382	1.702030	44.554
2	1.813797	1.664231	1.666905	44.257
3	1.772755	1.680679	1.632296	43.758
4	1.732080	1.639230	1.597687	43.232
..	…	…	…	…
115	-1.498773	-1.507222	-1.890561	4.251
116	-1.519950	-1.524017	-1.964428	3.774
117	-1.540897	-1.540213	-2.037778	2.735
118	-1.561616	-1.574693	-2.110611	2.036
119	-1.582110	-1.600572	-2.182928	1.000

[120 rows x 4 columns]

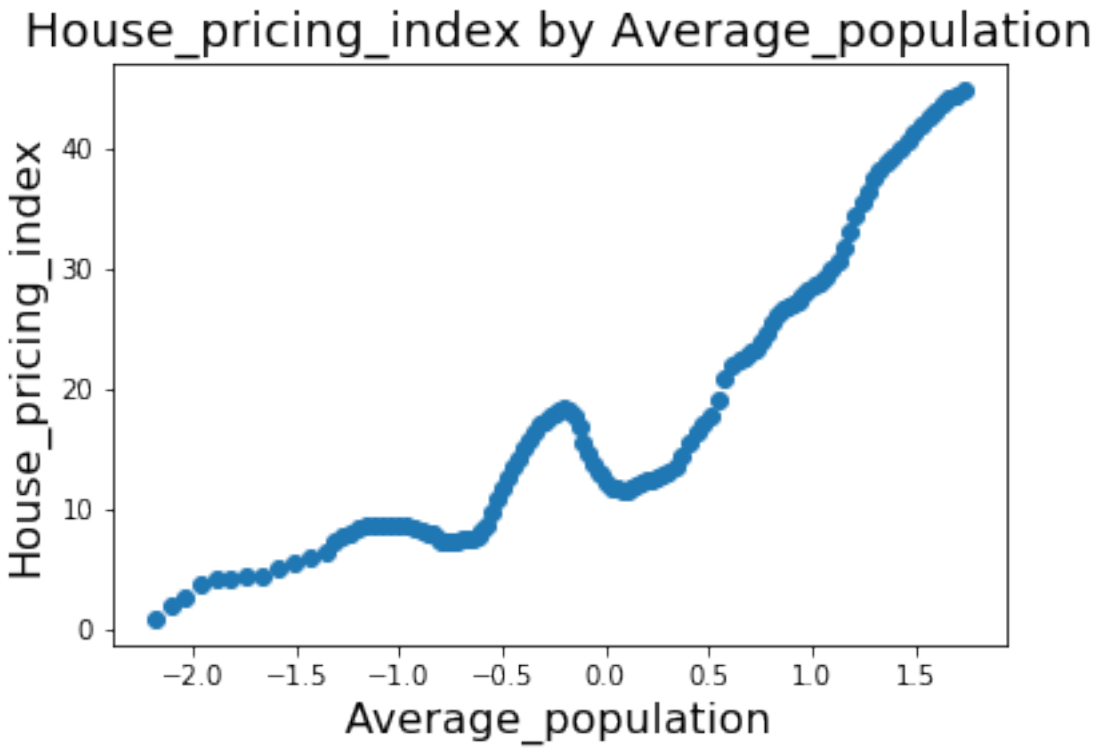
[10]:
```
plt.scatter(df2['Average_salary'], df2['House_pricing_index'])
plt.xlabel('Average_salary', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.title('House_pricing_index by Average_salary', size=18)
plt.show()
```



[11]:
```
plt.scatter(df2['Money_supply'], df2['House_pricing_index'])
plt.xlabel('Money_supply', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.title('House_pricing_index by Money_supply', size=18)
plt.show()
```



[12]:
```
plt.scatter(df2['Average_population'], df2['House_pricing_index'])
plt.xlabel('Average_population', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.title('House_pricing_index by Average_population', size=18)
plt.show()
```



[13]:
```
results = smf.ols('House_pricing_index ~ Average_population + Money_supply +␣
↪Average_salary', data=df2).fit() # estimate our OLS regression!
results.summary()
```

[13]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:     House_pricing_index   R-squared:                       0.956
Model:                             OLS   Adj. R-squared:                  0.955
Method:                  Least Squares   F-statistic:                     838.8
Date:                 Sat, 21 May 2022   Prob (F-statistic):           1.93e-78
Time:                         10:42:18   Log-Likelihood:                -277.69
No. Observations:                  120   AIC:                             563.4
Df Residuals:                      116   BIC:                             574.5
Df Model:                            3
Covariance Type:             nonrobust
==============================================================================
======
                          coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
```

```
------
Intercept             18.1452      0.227     79.842      0.000      17.695
18.595
Average_population    -4.5426      1.639     -2.771      0.007      -7.789
-1.296
Money_supply         -36.7004      3.494    -10.504      0.000     -43.621
-29.780
Average_salary        52.1568      3.345     15.592      0.000      45.531
58.782
==============================================================================
Omnibus:                        7.946   Durbin-Watson:                   0.253
Prob(Omnibus):                  0.019   Jarque-Bera (JB):                3.585
Skew:                          -0.130   Prob(JB):                        0.167
Kurtosis:                       2.194   Cond. No.                         35.7
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[14]:
```
results.params
```

[14]:
```
Intercept             18.145200
Average_population    -4.542628
Money_supply         -36.700428
Average_salary        52.156841
dtype: float64
```

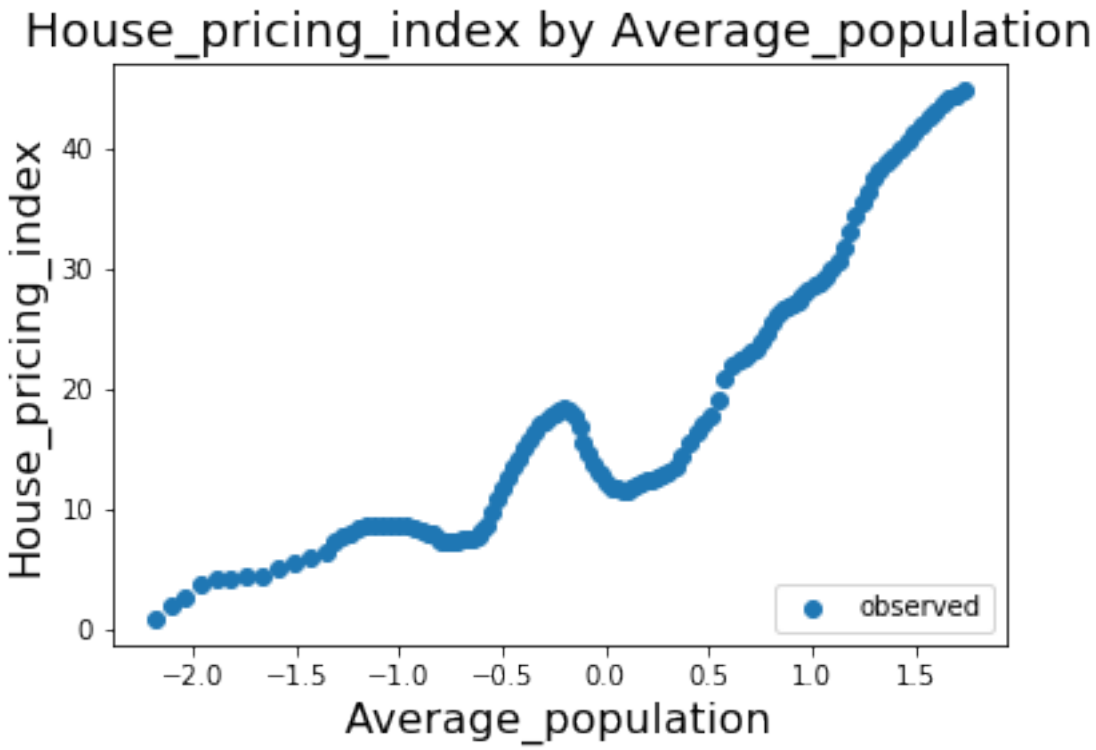
House_Pricing_Index = 18.145200 - 4.542628Average_population - 36.700428Money_supply + 52.156841Average_salary

[17]:
```
slope = results.params['Average_population']  # extract slope / coefficient
intercept = results.params['Intercept'] # extract intercept

# create fit line
x = np.arange(0.5, 3, 0.5) # synthetic x values
y = slope * x + intercept # synthethic y values using equation for a line

plt.scatter(X_scaled['Average_population'], df['House_pricing_index'],␣
 ↪label='observed') # data

plt.xlabel('Average_population', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.legend(loc='lower right')
plt.title('House_pricing_index by Average_population', size=18)
plt.show()
```



[18]:
```
for i, j in zip(x,y):
    print('Estimated House_pricing_index for {} Average_population: {:.6}'.
 ↪format(i,j))
```

```
Estimated House_pricing_index for 0.5 Average_population: 15.8739
Estimated House_pricing_index for 1.0 Average_population: 13.6026
Estimated House_pricing_index for 1.5 Average_population: 11.3313
Estimated House_pricing_index for 2.0 Average_population: 9.05994
Estimated House_pricing_index for 2.5 Average_population: 6.78863
```

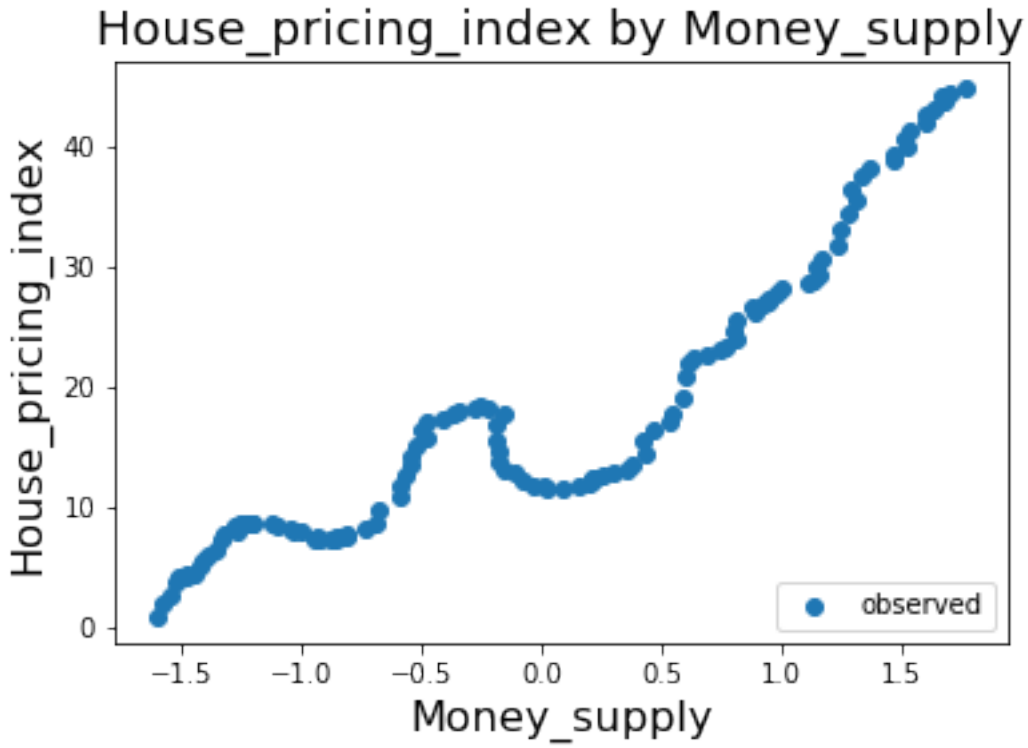
[20]:
```
slope = results.params['Money_supply']  # extract slope / coefficient
intercept = results.params['Intercept'] # extract intercept

# create fit line
x = np.arange(0.5, 3, 0.5) # synthetic x values
y = slope * x + intercept # synthethic y values using equation for a line

plt.scatter(X_scaled['Money_supply'], df['House_pricing_index'],␣
 ↪label='observed') # data

plt.xlabel('Money_supply', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.legend(loc='lower right')
```

```
plt.title('House_pricing_index by Money_supply', size=18)
plt.show()
```



[75]:
```
for a, b in zip(x,y):
    print('Estimated House_pricing_index for {} Money_supply: {:.6}'.
 ↪format(a,b))
```

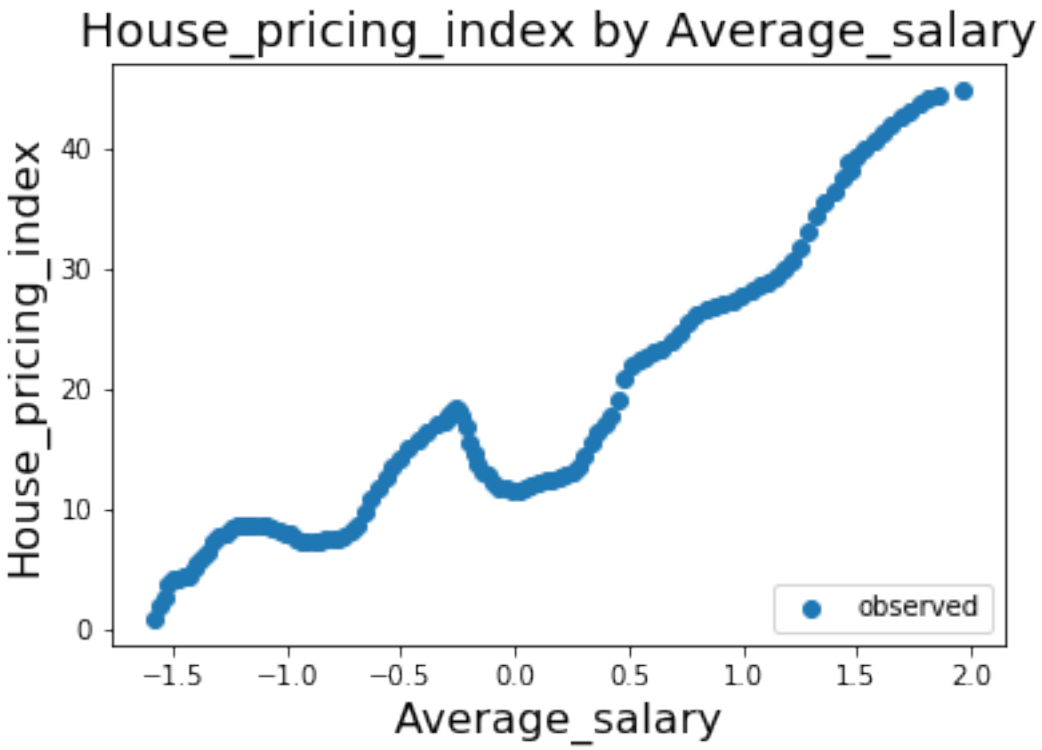
```
Estimated House_pricing_index for 0.5 Money_supply: -0.205014
Estimated House_pricing_index for 1.0 Money_supply: -18.5552
Estimated House_pricing_index for 1.5 Money_supply: -36.9054
Estimated House_pricing_index for 2.0 Money_supply: -55.2557
Estimated House_pricing_index for 2.5 Money_supply: -73.6059
```

[21]:
```
slope = results.params['Average_salary']  # extract slope / coefficient
intercept = results.params['Intercept'] # extract intercept

# create fit line
x = np.arange(0.5, 3, 0.5) # synthetic x values
y = slope * x + intercept # synthethic y values using equation for a line

plt.scatter(X_scaled['Average_salary'], df['House_pricing_index'],␣
 ↪label='observed') # data
```

```
plt.xlabel('Average_salary', size=16)
plt.ylabel('House_pricing_index', size=16)
plt.legend(loc='lower right')
plt.title('House_pricing_index by Average_salary', size=18)
plt.show()
```



[22]:
```
for c, d in zip(x,y):
    print('Estimated House_pricing_index for {} Average_salary: {:.6}'.
 ↪format(c,d))
```

```
Estimated House_pricing_index for 0.5 Average_salary: 44.2236
Estimated House_pricing_index for 1.0 Average_salary: 70.302
Estimated House_pricing_index for 1.5 Average_salary: 96.3805
Estimated House_pricing_index for 2.0 Average_salary: 122.459
Estimated House_pricing_index for 2.5 Average_salary: 148.537
```

3 Regression and Prediction

[23]:
```
x_train_regress, x_test_regress, y_train_regress, y_test_regress =␣
 ↪train_test_split(X_scaled, Y,
                                                    test_size=0.33,
```

```
                                                    random_state=42)  # setting␣
↪random state for repeatable results
```

[24]:
```
trained = LinearRegression().fit(x_train_regress, y_train_regress)
print ('Intercept', trained.intercept_)
print ('Coefficient',trained.coef_)
```

```
Intercept [18.12177852]
Coefficient [[ 52.04124416 -36.94201237  -4.21850587]]
```

House_Pricing_Index = 18.12177852 - 4.21850587Average_population - 36.94201237Money_supply + 52.04124416Average_salary

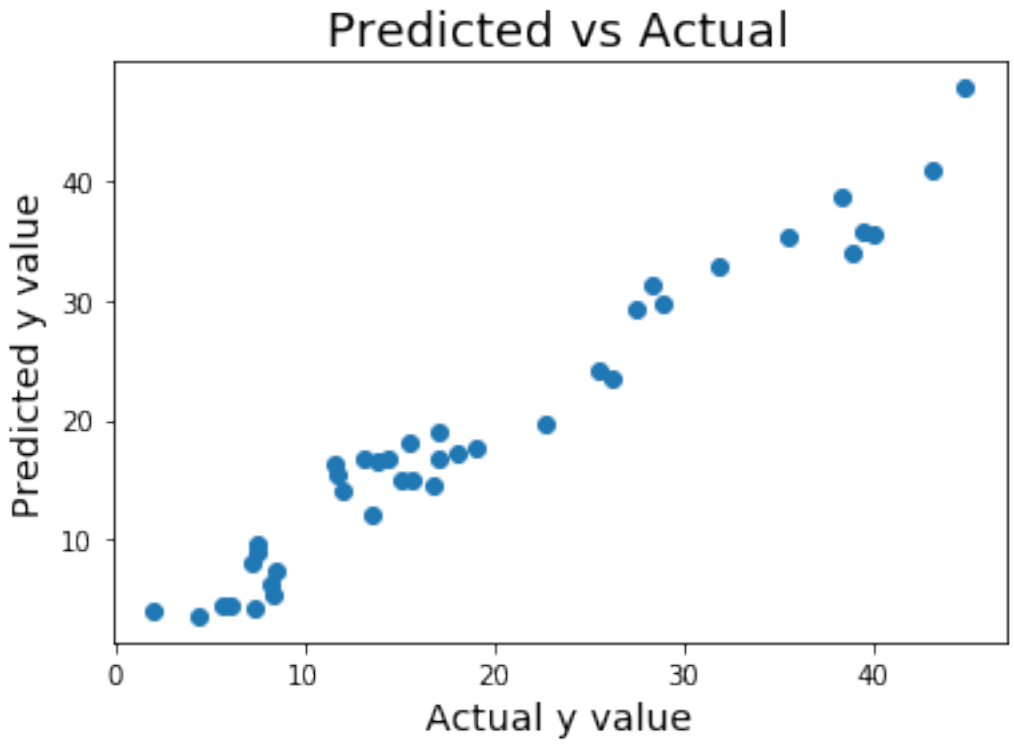
[25]:
```
y_predicted = trained.predict(x_test_regress)
df3 = pd.DataFrame(y_predicted)
```

[26]:
```
df4 = pd.concat([y_test_regress, df3], axis=1, join='inner')
df4.rename(columns = {'House_pricing_index':'Actual Value', 0:'Predicted␣
↪Value'}, inplace = True)
df4.head()
```

[26]:

	Actual Value	Predicted Value
4	43.232	29.228849
26	27.462	35.398346
10	39.463	32.841178
18	31.854	9.686608
11	38.936	16.592338

[27]:
```
plt.scatter(y_test_regress, y_predicted)
plt.title('Predicted vs Actual', size=18)
plt.xlabel('Actual y value', size = 14)
plt.ylabel('Predicted y value', size = 14)
plt.show()
```



[28]:
```
print(np.sqrt(metrics.mean_squared_error(y_test_regress, y_predicted)))
```

```
2.403105388923609
```

[29]:
```
print('R-Squared is',metrics.r2_score(y_test_regress, y_predicted))
```

```
R-Squared is 0.9597502132606902
```

[ ]:

[ ]:

[ ]: