In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

%matplotlib inline
plt.style.use('seaborn-poster')
```

In [3]:
```
df = pd.read_csv('/Users/swllms/DAT-10-14-SW/class material/Unit3/Data
/housing.csv')
```

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

Out[4]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LST |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 | 4 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 | 9 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 | 5 |

In [5]:
```
from sklearn.linear_model import LinearRegression
```

In [21]:
```
lreg=LinearRegression()
```

In [12]:
```
X = df.iloc[:, :13]
```

In [17]:
```
y = df['PRICE']
```

In [20]:
```
X
```

Out[20]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **5** | 0.02985 | 0.0 | 2.18 | 0 | 0.458 | 6.430 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 394.12 |
| **6** | 0.08829 | 12.5 | 7.87 | 0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5 | 311 | 15.2 | 395.60 |
| **7** | 0.14455 | 12.5 | 7.87 | 0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5 | 311 | 15.2 | 396.90 |
| **8** | 0.21124 | 12.5 | 7.87 | 0 | 0.524 | 5.631 | 100.0 | 6.0821 | 5 | 311 | 15.2 | 386.63 |
| **9** | 0.17004 | 12.5 | 7.87 | 0 | 0.524 | 6.004 | 85.9 | 6.5921 | 5 | 311 | 15.2 | 386.71 |
| **10** | 0.22489 | 12.5 | 7.87 | 0 | 0.524 | 6.377 | 94.3 | 6.3467 | 5 | 311 | 15.2 | 392.52 |
| **11** | 0.11747 | 12.5 | 7.87 | 0 | 0.524 | 6.009 | 82.9 | 6.2267 | 5 | 311 | 15.2 | 396.90 |
| **12** | 0.09378 | 12.5 | 7.87 | 0 | 0.524 | 5.889 | 39.0 | 5.4509 | 5 | 311 | 15.2 | 390.50 |
| **13** | 0.62976 | 0.0 | 8.14 | 0 | 0.538 | 5.949 | 61.8 | 4.7075 | 4 | 307 | 21.0 | 396.90 |
| **14** | 0.63796 | 0.0 | 8.14 | 0 | 0.538 | 6.096 | 84.5 | 4.4619 | 4 | 307 | 21.0 | 380.02 |
| **15** | 0.62739 | 0.0 | 8.14 | 0 | 0.538 | 5.834 | 56.5 | 4.4986 | 4 | 307 | 21.0 | 395.62 |
| **16** | 1.05393 | 0.0 | 8.14 | 0 | 0.538 | 5.935 | 29.3 | 4.4986 | 4 | 307 | 21.0 | 386.85 |
| **17** | 0.78420 | 0.0 | 8.14 | 0 | 0.538 | 5.990 | 81.7 | 4.2579 | 4 | 307 | 21.0 | 386.75 |
| **18** | 0.80271 | 0.0 | 8.14 | 0 | 0.538 | 5.456 | 36.6 | 3.7965 | 4 | 307 | 21.0 | 288.99 |
| **19** | 0.72580 | 0.0 | 8.14 | 0 | 0.538 | 5.727 | 69.5 | 3.7965 | 4 | 307 | 21.0 | 390.95 |
| **20** | 1.25179 | 0.0 | 8.14 | 0 | 0.538 | 5.570 | 98.1 | 3.7979 | 4 | 307 | 21.0 | 376.57 |
| **21** | 0.85204 | 0.0 | 8.14 | 0 | 0.538 | 5.965 | 89.2 | 4.0123 | 4 | 307 | 21.0 | 392.53 |
| **22** | 1.23247 | 0.0 | 8.14 | 0 | 0.538 | 6.142 | 91.7 | 3.9769 | 4 | 307 | 21.0 | 396.90 |
| **23** | 0.98843 | 0.0 | 8.14 | 0 | 0.538 | 5.813 | 100.0 | 4.0952 | 4 | 307 | 21.0 | 394.54 |
| **24** | 0.75026 | 0.0 | 8.14 | 0 | 0.538 | 5.924 | 94.1 | 4.3996 | 4 | 307 | 21.0 | 394.33 |
| **25** | 0.84054 | 0.0 | 8.14 | 0 | 0.538 | 5.599 | 85.7 | 4.4546 | 4 | 307 | 21.0 | 303.42 |
| **26** | 0.67191 | 0.0 | 8.14 | 0 | 0.538 | 5.813 | 90.3 | 4.6820 | 4 | 307 | 21.0 | 376.88 |
| **27** | 0.95577 | 0.0 | 8.14 | 0 | 0.538 | 6.047 | 88.8 | 4.4534 | 4 | 307 | 21.0 | 306.38 |
| **28** | 0.77299 | 0.0 | 8.14 | 0 | 0.538 | 6.495 | 94.4 | 4.4547 | 4 | 307 | 21.0 | 387.94 |
| **29** | 1.00245 | 0.0 | 8.14 | 0 | 0.538 | 6.674 | 87.3 | 4.2390 | 4 | 307 | 21.0 | 380.23 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **476** | 4.87141 | 0.0 | 18.10 | 0 | 0.614 | 6.484 | 93.6 | 2.3053 | 24 | 666 | 20.2 | 396.21 |
| **477** | 15.02340 | 0.0 | 18.10 | 0 | 0.614 | 5.304 | 97.3 | 2.1007 | 24 | 666 | 20.2 | 349.48 |
| **478** | 10.23300 | 0.0 | 18.10 | 0 | 0.614 | 6.185 | 96.7 | 2.1705 | 24 | 666 | 20.2 | 379.70 |
| **479** | 14.33370 | 0.0 | 18.10 | 0 | 0.614 | 6.229 | 88.0 | 1.9512 | 24 | 666 | 20.2 | 383.32 |
| **480** | 5.82401 | 0.0 | 18.10 | 0 | 0.532 | 6.242 | 64.7 | 3.4242 | 24 | 666 | 20.2 | 396.90 |
| **481** | 5.70818 | 0.0 | 18.10 | 0 | 0.532 | 6.750 | 74.9 | 3.3317 | 24 | 666 | 20.2 | 393.07 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **482** | 5.73116 | 0.0 | 18.10 | 0 | 0.532 | 7.061 | 77.0 | 3.4106 | 24 | 666 | 20.2 | 395.28 |
| **483** | 2.81838 | 0.0 | 18.10 | 0 | 0.532 | 5.762 | 40.3 | 4.0983 | 24 | 666 | 20.2 | 392.92 |
| **484** | 2.37857 | 0.0 | 18.10 | 0 | 0.583 | 5.871 | 41.9 | 3.7240 | 24 | 666 | 20.2 | 370.73 |
| **485** | 3.67367 | 0.0 | 18.10 | 0 | 0.583 | 6.312 | 51.9 | 3.9917 | 24 | 666 | 20.2 | 388.62 |
| **486** | 5.69175 | 0.0 | 18.10 | 0 | 0.583 | 6.114 | 79.8 | 3.5459 | 24 | 666 | 20.2 | 392.68 |
| **487** | 4.83567 | 0.0 | 18.10 | 0 | 0.583 | 5.905 | 53.2 | 3.1523 | 24 | 666 | 20.2 | 388.22 |
| **488** | 0.15086 | 0.0 | 27.74 | 0 | 0.609 | 5.454 | 92.7 | 1.8209 | 4 | 711 | 20.1 | 395.09 |
| **489** | 0.18337 | 0.0 | 27.74 | 0 | 0.609 | 5.414 | 98.3 | 1.7554 | 4 | 711 | 20.1 | 344.05 |
| **490** | 0.20746 | 0.0 | 27.74 | 0 | 0.609 | 5.093 | 98.0 | 1.8226 | 4 | 711 | 20.1 | 318.43 |
| **491** | 0.10574 | 0.0 | 27.74 | 0 | 0.609 | 5.983 | 98.8 | 1.8681 | 4 | 711 | 20.1 | 390.11 |
| **492** | 0.11132 | 0.0 | 27.74 | 0 | 0.609 | 5.983 | 83.5 | 2.1099 | 4 | 711 | 20.1 | 396.90 |
| **493** | 0.17331 | 0.0 | 9.69 | 0 | 0.585 | 5.707 | 54.0 | 2.3817 | 6 | 391 | 19.2 | 396.90 |
| **494** | 0.27957 | 0.0 | 9.69 | 0 | 0.585 | 5.926 | 42.6 | 2.3817 | 6 | 391 | 19.2 | 396.90 |
| **495** | 0.17899 | 0.0 | 9.69 | 0 | 0.585 | 5.670 | 28.8 | 2.7986 | 6 | 391 | 19.2 | 393.29 |
| **496** | 0.28960 | 0.0 | 9.69 | 0 | 0.585 | 5.390 | 72.9 | 2.7986 | 6 | 391 | 19.2 | 396.90 |
| **497** | 0.26838 | 0.0 | 9.69 | 0 | 0.585 | 5.794 | 70.6 | 2.8927 | 6 | 391 | 19.2 | 396.90 |
| **498** | 0.23912 | 0.0 | 9.69 | 0 | 0.585 | 6.019 | 65.3 | 2.4091 | 6 | 391 | 19.2 | 396.90 |
| **499** | 0.17783 | 0.0 | 9.69 | 0 | 0.585 | 5.569 | 73.5 | 2.3999 | 6 | 391 | 19.2 | 395.77 |
| **500** | 0.22438 | 0.0 | 9.69 | 0 | 0.585 | 6.027 | 79.7 | 2.4982 | 6 | 391 | 19.2 | 396.90 |
| **501** | 0.06263 | 0.0 | 11.93 | 0 | 0.573 | 6.593 | 69.1 | 2.4786 | 1 | 273 | 21.0 | 391.99 |
| **502** | 0.04527 | 0.0 | 11.93 | 0 | 0.573 | 6.120 | 76.7 | 2.2875 | 1 | 273 | 21.0 | 396.90 |
| **503** | 0.06076 | 0.0 | 11.93 | 0 | 0.573 | 6.976 | 91.0 | 2.1675 | 1 | 273 | 21.0 | 396.90 |
| **504** | 0.10959 | 0.0 | 11.93 | 0 | 0.573 | 6.794 | 89.3 | 2.3889 | 1 | 273 | 21.0 | 393.45 |
| **505** | 0.04741 | 0.0 | 11.93 | 0 | 0.573 | 6.030 | 80.8 | 2.5050 | 1 | 273 | 21.0 | 396.90 |

506 rows × 13 columns

In [22]:
```
lreg.fit(X, y)
```
Out[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [35]:
```
lreg.score(X, y) #74%
```
Out[35]:
```
0.7406426641094094
```

In [23]:
```
lreg.coef_
```

Out[23]:
```
array([-1.08011358e-01,  4.64204584e-02,  2.05586264e-02,  2.6867338
2e+00,
       -1.77666112e+01,  3.80986521e+00,  6.92224640e-04, -1.4755668
5e+00,
        3.06049479e-01, -1.23345939e-02, -9.52747232e-01,  9.3116832
7e-03,
       -5.24758378e-01])
```

In [26]:
```
coeffs = pd.DataFrame({
    'Variable': X.columns,
    'Weight' : lreg.coef_
}).sort_values(by='Weight', ascending=False)
```

In [27]:
```
coeffs #For every value of variable add the value of weight.
#example for every RM(room) add 3.808865
#A coeff is the value if increased by exactly 1
#the table makes all values weighted the same
#shows the marginal impact of each variable to effect y.
```

Out[27]:

| | Variable | Weight |
|---|---|---|
| 5 | RM | 3.809865 |
| 3 | CHAS | 2.686734 |
| 8 | RAD | 0.306049 |
| 1 | ZN | 0.046420 |
| 2 | INDUS | 0.020559 |
| 11 | B | 0.009312 |
| 6 | AGE | 0.000692 |
| 9 | TAX | -0.012335 |
| 0 | CRIM | -0.108011 |
| 12 | LSTAT | -0.524758 |
| 10 | PTRATIO | -0.952747 |
| 7 | DIS | -1.475567 |
| 4 | NOX | -17.766611 |

In [28]:
```
#STANDARDIZING NOTES: Critical step for setting up many models correct
ly
#Gives all numeric variables a mean of 0, and a variance of 1
#Puts all weights on an equal footing
#Do this by subtracting each value by its mean & divide it by its stan
dard deviation.

#This is the manual way to do it. Use Scalers to preprocess this: See
bonus info up tomorrow.
```

In [31]:
```
(X - X.mean()).describe() #Called centering the data.
```

Out[31]:

| | CRIM | ZN | INDUS | CHAS | NOX | RI |
|---|---|---|---|---|---|---|
| **count** | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+0 |
| **mean** | -3.024370e-15 | 2.076161e-14 | -2.800395e-14 | -1.189760e-16 | 2.571505e-16 | -8.999389e-1 |
| **std** | 8.601545e+00 | 2.332245e+01 | 6.860353e+00 | 2.539940e-01 | 1.158777e-01 | 7.026171e-0 |
| **min** | -3.607204e+00 | -1.136364e+01 | -1.067678e+01 | -6.916996e-02 | -1.696951e-01 | -2.723634e+0 |
| **25%** | -3.531479e+00 | -1.136364e+01 | -5.946779e+00 | -6.916996e-02 | -1.056951e-01 | -3.991344e-0 |
| **50%** | -3.357014e+00 | -1.136364e+01 | -1.446779e+00 | -6.916996e-02 | -1.669506e-02 | -7.613439e-0 |
| **75%** | 6.355894e-02 | 1.136364e+00 | 6.963221e+00 | -6.916996e-02 | 6.930494e-02 | 3.388656e-0 |
| **max** | 8.536268e+01 | 8.863636e+01 | 1.660322e+01 | 9.308300e-01 | 3.163049e-01 | 2.495366e+0 |

In [33]:
```
X_std = (X - X.mean()) /X.std() #this is how your data should look bef
ore feeding it into model
#hovers around 0 and numberic range is small.
```

In [34]:
```
X_std.describe() #std is now 1 across the board, Need to make sure dat
a is all on the same scale
```

Out[34]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM |
|---|---|---|---|---|---|---|
| **count** | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 | 5.060000e+02 |
| **mean** | 8.326673e-17 | 3.466704e-16 | -3.016965e-15 | 3.999875e-16 | 3.167427e-15 | -1.258809e-14 |
| **std** | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 |
| **min** | -4.193669e-01 | -4.872402e-01 | -1.556302e+00 | -2.723291e-01 | -1.464433e+00 | -3.876413e+00 |
| **25%** | -4.105633e-01 | -4.872402e-01 | -8.668328e-01 | -2.723291e-01 | -9.121262e-01 | -5.680681e-01 |
| **50%** | -3.902803e-01 | -4.872402e-01 | -2.108898e-01 | -2.723291e-01 | -1.440749e-01 | -1.083583e-01 |
| **75%** | 7.389247e-03 | 4.872402e-02 | 1.014995e+00 | -2.723291e-01 | 5.980871e-01 | 4.822906e-01 |
| **max** | 9.924110e+00 | 3.800473e+00 | 2.420170e+00 | 3.664771e+00 | 2.729645e+00 | 3.551530e+00 |

In [37]:
```
lreg.fit(X_std, y)
```

Out[37]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [38]:
```
lreg.coef_
```

Out[38]:
```
array([-0.92906457,  1.08263896,  0.14103943,  0.68241438, -2.058753
61,
        2.67687661,  0.01948534, -3.10711605,  2.6648522 , -2.078836
89,
       -2.06264585,  0.85010886, -3.74733185])
```

In [39]:
```
xcoeffs = pd.DataFrame({
    'Variable': X_std.columns,
    'Weight' : lreg.coef_
}).sort_values(by='Weight', ascending=False)
xcoeffs
```

Out[39]:

| | Variable | Weight |
|---|---|---|
| 5 | RM | 2.676877 |
| 8 | RAD | 2.664852 |
| 1 | ZN | 1.082639 |
| 11 | B | 0.850109 |
| 3 | CHAS | 0.682414 |
| 2 | INDUS | 0.141039 |
| 6 | AGE | 0.019485 |
| 0 | CRIM | -0.929065 |
| 4 | NOX | -2.058754 |
| 10 | PTRATIO | -2.062646 |
| 9 | TAX | -2.078837 |
| 7 | DIS | -3.107116 |
| 12 | LSTAT | -3.747332 |

In [40]:
```
lreg.score(X_std, y)
#if your r2 did not change your predictions would be practically the s
ame
#the range is reduce TAX and STAT are now larger than NOX
#this is a better way to read and see the coeffs.
#Remeber and increase of 1 is an increase of 1 standard d.
#if you dont take this step (Standardizing) your results will not be v
alid.

#note this is not to be confused with normalizing, they are different.
```

Out[40]: 0.7406426641094095

In [ ]:
```
#When does standardization apply:
    #Linear models
    #Anything that uses a weight penalty (l1, l2, etc)
    #Any algorithm that uses gradient descent
```

In [ ]:
```
#Cross Validation Notes:
#How do we know our model will generalize to the ourside world?
#We don't know how well our information will work with uknown data.
#Never fit your model to all of your data. #use cross validation to va
lidate your data.
```

In [ ]:
```
#Bias vs. Variance Tradeoff: looking for the optiomum model
    #Bias refer to when your data doesnt adequately capture the info n
ecessary to solve our problem
    #Variance refer to when your model mistaknely inerprets random cor
reclations as meaningful
        #(r2 value looks high but when new data comes in it crashes)
```

Basic idea of Cross Validation: Sepearte your data inot two groups a test set and training set. Test set is not touched until the very end, and is only used for final model eval Training set is used for fine-tuning and eval your model Note: either randomized or based on date.

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
2, random_state=2019)
```

In [43]:
```
#train and text are used to split the data where the test size is 20%
#the random state ensure that our random numbers are generated the sam
e
#(randomly shuffled the same way for everyone)
X_train.head()
```

Out[43]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **249** | 0.19073 | 22.0 | 5.86 | 0 | 0.431 | 6.718 | 17.5 | 7.8265 | 7 | 330 | 19.1 | 393.74 |
| **51** | 0.04337 | 21.0 | 5.64 | 0 | 0.439 | 6.115 | 63.0 | 6.8147 | 4 | 243 | 16.8 | 393.97 |
| **151** | 1.49632 | 0.0 | 19.58 | 0 | 0.871 | 5.404 | 100.0 | 1.5916 | 5 | 403 | 14.7 | 341.60 |
| **486** | 5.69175 | 0.0 | 18.10 | 0 | 0.583 | 6.114 | 79.8 | 3.5459 | 24 | 666 | 20.2 | 392.68 |
| **235** | 0.33045 | 0.0 | 6.20 | 0 | 0.507 | 6.086 | 61.5 | 3.6519 | 8 | 307 | 17.4 | 376.75 |

In [46]:
```
lreg.fit(X_train, y_train) #fit on the training sets
```

Out[46]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [47]:
```
lreg.score(X_test, y_test) #score on the test sets
```

Out[47]:
```
0.6174065999127849
```

In [ ]:
```
#additional step is to create a validation set: test set within the te
st set.
#Think of it as a dress rehersal for the real test set. You would scor
e on the validation set
#remember you may not always have the test set (Because this would com
e later)
#and this is where you would use validation set
```