In [57]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import sklearn
from sklearn.datasets import load_boston

import seaborn as sns

from matplotlib import rcParams
sns.set_style("white")

boston = load_boston()
bos = pd.DataFrame(boston.data)
bos.head(2)
```

Out[57]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.9	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9	9.14

In [58]:
```
boston.data.shape
```

Out[58]: (506, 13)

In [59]:
```
bos.describe()
```

Out[59]:

	0	1	2	3	4	5	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.593761	11.363636	11.136779	0.069170	0.554695	6.284634	68.574
std	8.596783	23.322453	6.860353	0.253994	0.115878	0.702617	28.148
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500
75%	3.647423	12.500000	18.100000	0.000000	0.624000	6.623500	94.075
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00

In [60]:
```
boston.keys()
```

Out[60]:
```
dict_keys(['data', 'DESCR', 'feature_names', 'target'])
```

In [61]:
```
display(boston.feature_names)
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD'
,
       'TAX', 'PTRATIO', 'B', 'LSTAT'],
      dtype='<U7')
```

In [62]:
```
display(boston.target)
```

```
array([ 24. ,  21.6,  34.7,  33.4,  36.2,  28.7,  22.9,  27.1,  16.5,
        18.9,  15. ,  18.9,  21.7,  20.4,  18.2,  19.9,  23.1,  17.5,
        20.2,  18.2,  13.6,  19.6,  15.2,  14.5,  15.6,  13.9,  16.6,
        14.8,  18.4,  21. ,  12.7,  14.5,  13.2,  13.1,  13.5,  18.9,
        20. ,  21. ,  24.7,  30.8,  34.9,  26.6,  25.3,  24.7,  21.2,
        19.3,  20. ,  16.6,  14.4,  19.4,  19.7,  20.5,  25. ,  23.4,
        18.9,  35.4,  24.7,  31.6,  23.3,  19.6,  18.7,  16. ,  22.2,
        25. ,  33. ,  23.5,  19.4,  22. ,  17.4,  20.9,  24.2,  21.7,
        22.8,  23.4,  24.1,  21.4,  20. ,  20.8,  21.2,  20.3,  28. ,
        23.9,  24.8,  22.9,  23.9,  26.6,  22.5,  22.2,  23.6,  28.7,
        22.6,  22. ,  22.9,  25. ,  20.6,  28.4,  21.4,  38.7,  43.8,
        33.2,  27.5,  26.5,  18.6,  19.3,  20.1,  19.5,  19.5,  20.4,
        19.8,  19.4,  21.7,  22.8,  18.8,  18.7,  18.5,  18.3,  21.2,
        19.2,  20.4,  19.3,  22. ,  20.3,  20.5,  17.3,  18.8,  21.4,
        15.7,  16.2,  18. ,  14.3,  19.2,  19.6,  23. ,  18.4,  15.6,
        18.1,  17.4,  17.1,  13.3,  17.8,  14. ,  14.4,  13.4,  15.6,
        11.8,  13.8,  15.6,  14.6,  17.8,  15.4,  21.5,  19.6,  15.3,
        19.4,  17. ,  15.6,  13.1,  41.3,  24.3,  23.3,  27. ,  50. ,
        50. ,  50. ,  22.7,  25. ,  50. ,  23.8,  23.8,  22.3,  17.4,
        19.1,  23.1,  23.6,  22.6,  29.4,  23.2,  24.6,  29.9,  37.2,
        39.8,  36.2,  37.9,  32.5,  26.4,  29.6,  50. ,  32. ,  29.8,
        34.9,  37. ,  30.5,  36.4,  31.1,  29.1,  50. ,  33.3,  30.3,
        34.6,  34.9,  32.9,  24.1,  42.3,  48.5,  50. ,  22.6,  24.4,
        22.5,  24.4,  20. ,  21.7,  19.3,  22.4,  28.1,  23.7,  25. ,
        23.3,  28.7,  21.5,  23. ,  26.7,  21.7,  27.5,  30.1,  44.8,
        50. ,  37.6,  31.6,  46.7,  31.5,  24.3,  31.7,  41.7,  48.3,
        29. ,  24. ,  25.1,  31.5,  23.7,  23.3,  22. ,  20.1,  22.2,
        23.7,  17.6,  18.5,  24.3,  20.5,  24.5,  26.2,  24.4,  24.8,
        29.6,  42.8,  21.9,  20.9,  44. ,  50. ,  36. ,  30.1,  33.8,
        43.1,  48.8,  31. ,  36.5,  22.8,  30.7,  50. ,  43.5,  20.7,
        21.1,  25.2,  24.4,  35.2,  32.4,  32. ,  33.2,  33.1,  29.1,
        35.1,  45.4,  35.4,  46. ,  50. ,  32.2,  22. ,  20.1,  23.2,
        22.3,  24.8,  28.5,  37.3,  27.9,  23.9,  21.7,  28.6,  27.1,
        20.3,  22.5,  29. ,  24.8,  22. ,  26.4,  33.1,  36.1,  28.4,
        33.4,  28.2,  22.8,  20.3,  16.1,  22.1,  19.4,  21.6,  23.8,
        16.2,  17.8,  19.8,  23.1,  21. ,  23.8,  23.1,  20.4,  18.5,
        25. ,  24.6,  23. ,  22.2,  19.3,  22.6,  19.8,  17.1,  19.4,
        22.2,  20.7,  21.1,  19.5,  18.5,  20.6,  19. ,  18.7,  32.7,
```

```
       16.5,  23.9,  31.2,  17.5,  17.2,  23.1,  24.5,  26.6,  22.9,
       24.1,  18.6,  30.1,  18.2,  20.6,  17.8,  21.7,  22.7,  22.6,
       25. ,  19.9,  20.8,  16.8,  21.9,  27.5,  21.9,  23.1,  50. ,
       50. ,  50. ,  50. ,  50. ,  13.8,  13.8,  15. ,  13.9,  13.3,
       13.1,  10.2,  10.4,  10.9,  11.3,  12.3,   8.8,   7.2,  10.5,
        7.4,  10.2,  11.5,  15.1,  23.2,   9.7,  13.8,  12.7,  13.1,
       12.5,   8.5,   5. ,   6.3,   5.6,   7.2,  12.1,   8.3,   8.5,
        5. ,  11.9,  27.9,  17.2,  27.5,  15. ,  17.2,  17.9,  16.3,
        7. ,   7.2,   7.5,  10.4,   8.8,   8.4,  16.7,  14.2,  20.8,
       13.4,  11.7,   8.3,  10.2,  10.9,  11. ,   9.5,  14.5,  14.1,
       16.1,  14.3,  11.7,  13.4,   9.6,   8.7,   8.4,  12.8,  10.5,
       17.1,  18.4,  15.4,  10.8,  11.8,  14.9,  12.6,  14.1,  13. ,
       13.4,  15.2,  16.1,  17.8,  14.9,  14.1,  12.7,  13.5,  14.9,
       20. ,  16.4,  17.7,  19.5,  20.2,  21.4,  19.9,  19. ,  19.1,
       19.1,  20.1,  19.9,  19.6,  23.2,  29.8,  13.8,  13.3,  16.7,
       12. ,  14.6,  21.4,  23. ,  23.7,  25. ,  21.8,  20.6,  21.2,
       19.1,  20.6,  15.2,   7. ,   8.1,  13.6,  20.1,  21.8,  24.5,
       23.1,  19.7,  18.3,  21.2,  17.5,  16.8,  22.4,  20.6,  23.9,
       22. ,  11.9])
```

In [63]:
```
print(boston.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over
25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds r
iver; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1
940
        - DIS      weighted distances to five Boston employment centre
s
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of bla
```

```
cks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at
Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedon
ic
prices and the demand for clean air', J. Environ. Economics & Manageme
nt,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diag
nostics
...', Wiley, 1980.   N.B. Various transformations are used in the tabl
e on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning pap
ers that address regression
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influ
ential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learn
ing. In Proceedings on the Tenth International Conference of Machine L
earning, 236-243, University of Massachusetts, Amherst. Morgan Kaufman
n.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

In [64]:
```
bos = pd.DataFrame(boston.data)
bos.head(2)
```

Out[64]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.9	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9	9.14

In [65]:
```
bos.columns = boston.feature_names
bos.head(2)
```

Out[65]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.9
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9

In [66]:
```
bos['PRICE'] = boston.target
bos.head(2)
```

Out[66]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.9
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9

In [67]:
```
bos.rename(columns={'CRIM': 'Crime_Rate', 'ZN': 'Lrg_Res_Zones', 'INDUS
':'Industry'}, inplace=True)
bos.head(2)
```

Out[67]:

	Crime_Rate	Lrg_Res_Zones	Industry	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0

In [68]:
```
sns.set(style='ticks')
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.regplot(y="PRICE", x="Crime_Rate", data=bos, fit_reg=True)
plt.ylim(0,50)
plt.title("Home Price Versus Crime Rate")
```

Out[68]: <matplotlib.text.Text at 0x1179404e0>

In [69]:
```
sns.set(style='ticks')
sns.jointplot(y="PRICE", x="Crime_Rate", data=bos, size=10, kind='reg',
joint_kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
plt.ylim(0,50)
plt.title("Home Price Versus Crime Rate")
```

Out[69]: <matplotlib.text.Text at 0x118139208>

In [70]:
```
sns.set(style='ticks')
#fig, ax = plt.subplots()
# the size of A4 paper
#fig.set_size_inches(11.7, 8.27)
sns.jointplot(y="PRICE", x="AGE", data=bos, size=10, kind='reg', joint_
kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
plt.ylim(0,50) # set Y axis range to minimum of zero
plt.title("Average Age of Home Versus Home Price")
```

Out[70]:
```
<matplotlib.text.Text at 0x11872a550>
```

In [71]:
```
sns.set(style='ticks')
sns.jointplot(y="PRICE", x="AGE", data=bos, size=10, kind='kde')
plt.ylim(0,50) # set Y axis range to minimum of zero
plt.title("Average Age of Home Versus Home Price")
```

Out[71]:
```
<matplotlib.text.Text at 0x1185e4780>
```

In [ ]:
```

```

In [72]:
```
feature_col = ['Crime_Rate']
X = bos[feature_col]
y = bos.PRICE
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X, y)

print(lm.intercept_)
print(lm.coef_)
```

```
24.0162197635
[-0.41277468]
```

In [73]:
```
X_new = pd.DataFrame({'Crime_Rate': [87]})
X_new.head()
```

Out[73]:

	Crime_Rate
0	87

In [74]:
```
lm.predict(X_new)
```

Out[74]:
```
array([-11.89517768])
```

In [75]:
```
X_new = pd.DataFrame({'Crime_Rate': [bos.Crime_Rate.min(), bos.Crime_Ra
te.max()]})
X_new.head()
```

Out[75]:

	Crime_Rate
0	0.00632
1	88.97620

In [76]:
```
preds = lm.predict(X_new)
preds
```

Out[76]:
```
array([ 24.01361103, -12.71090301])
```

In [77]:
```
bos.plot(kind='scatter', x='Crime_Rate', y='PRICE')
plt.plot(X_new, preds, c='red', linewidth=2)
```

Out[77]:
```
[<matplotlib.lines.Line2D at 0x117a3a908>]
```

In [78]:
```
import statsmodels.formula.api as smf
lm = smf.ols(formula='PRICE ~ Crime_Rate', data=bos).fit()
lm.conf_int()
```

Out[78]:

	0	1
Intercept	23.212074	24.820366
Crime_Rate	-0.499150	-0.326399

In [79]:
```
# print the p-values for the model coefficients
lm.pvalues
```

Out[79]:
```
Intercept     2.168010e-227
Crime_Rate     2.083550e-19
dtype: float64
```

In [80]:
```
lm.rsquared
```

Out[80]:
```
0.14886609291873587
```

In [81]:
```
feature_cols = ['Crime_Rate', 'Lrg_Res_Zones', 'Industry', 'CHAS', 'NOX
', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']
X = bos[feature_cols]
y = bos.PRICE

lm = LinearRegression()
lm.fit(X, y)

print(lm.intercept_)
print(lm.coef_)
```

```
36.4911032804
[ -1.07170557e-01   4.63952195e-02   2.08602395e-02   2.68856140e+00
  -1.77957587e+01   3.80475246e+00   7.51061703e-04  -1.47575880e+00
   3.05655038e-01  -1.23293463e-02  -9.53463555e-01   9.39251272e-03
  -5.25466633e-01]
```

In [82]:
```
lm = smf.ols(formula='PRICE ~ Crime_Rate + Lrg_Res_Zones + Industry + C
HAS + NOX + RM + AGE + DIS + RAD + TAX + PTRATIO + B + LSTAT', data=bos
).fit()
lm.conf_int()
lm.summary()
```

Out[82]:

OLS Regression Results

Dep. Variable:	PRICE	**R-squared:**	0.741
Model:	OLS	**Adj. R-squared:**	0.734
Method:	Least Squares	**F-statistic:**	108.1
Date:	Tue, 28 Aug 2018	**Prob (F-statistic):**	6.95e-135
Time:	23:54:22	**Log-Likelihood:**	-1498.8
No. Observations:	506	**AIC:**	3026.
Df Residuals:	492	**BIC:**	3085.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	36.4911	5.104	7.149	0.000	26.462	46.520
Crime_Rate	-0.1072	0.033	-3.276	0.001	-0.171	-0.043
Lrg_Res_Zones	0.0464	0.014	3.380	0.001	0.019	0.073
Industry	0.0209	0.061	0.339	0.735	-0.100	0.142
CHAS	2.6886	0.862	3.120	0.002	0.996	4.381
NOX	-17.7958	3.821	-4.658	0.000	-25.302	-10.289
RM	3.8048	0.418	9.102	0.000	2.983	4.626
AGE	0.0008	0.013	0.057	0.955	-0.025	0.027
DIS	-1.4758	0.199	-7.398	0.000	-1.868	-1.084
RAD	0.3057	0.066	4.608	0.000	0.175	0.436
TAX	-0.0123	0.004	-3.278	0.001	-0.020	-0.005
PTRATIO	-0.9535	0.131	-7.287	0.000	-1.211	-0.696
B	0.0094	0.003	3.500	0.001	0.004	0.015
LSTAT	-0.5255	0.051	-10.366	0.000	-0.625	-0.426

Omnibus:	178.029	**Durbin-Watson:**	1.078
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	782.015
Skew:	1.521	**Prob(JB):**	1.54e-170
Kurtosis:	8.276	**Cond. No.**	1.51e+04

In [ ]: