Boston

February 21, 2025

1 Exercise 15: This problem involves the Boston data set, which we saw in the lab for this chapter.

1.1 Import notebook funcs

[1]:
```
from notebookfuncs import *
```

1.2 Import userfuncs

[2]:
```
from userfuncs import *
```

1.3 Import libraries

[3]:
```
from ISLP import load_data
from summarytools import dfSummary
import seaborn as sns
```

[4]:
```
Boston = load_data("Boston")
dfSummary(Boston)
```

[4]: <pandas.io.formats.style.Styler at 0x73cb484d78f0>

1.4 We will now try to predict per capita crime rate using the other variables in this data set.

1.5 In other words, per capita crime rate (crim) is the response, and the other variables are the predictors.

1.6 (a) For each predictor, fit a simple linear regression model to predict the response.

[5]:
```
def regress_for_each_predictor(data=None,response=None):
  if (data is None or response is None):
    return None
  col_names = list(data.columns.values)
  col_names.remove(response)
  rows = []
  for col in col_names:
```

```python
    formula = f"{response} ~ {col}"
    model = smf.ols(formula, data=data)
    results = model.fit()
    results_df = pd.DataFrame({"Regressor": col,
                              "Coefficient": results.params.iloc[1],
                              "P-value": results.pvalues.iloc[1],
                             "R-Squared": results.rsquared}, index=[0])
    rows.append(results_df)

  rows = pd.concat(rows)
  rows.set_index(["Regressor"],inplace=True)
  rows.sort_values("R-Squared", inplace=True, ascending=False)
  return rows

regressors = regress_for_each_predictor(data=Boston,response="crim")
```

[5]:

```
           Coefficient       P-value  R-Squared
Regressor
rad           0.617911  2.693844e-56   0.391257
tax           0.029742  2.357127e-47   0.339614
lstat         0.548805  2.654277e-27   0.207591
nox          31.248531  3.751739e-23   0.177217
indus         0.509776  1.450349e-21   0.165310
medv         -0.363160  1.173987e-19   0.150780
dis          -1.550902  8.519949e-19   0.144149
age           0.107786  2.854869e-16   0.124421
ptratio       1.151983  2.942922e-11   0.084068
rm           -2.684051  6.346703e-07   0.048069
zn           -0.073935  5.506472e-06   0.040188
chas         -1.892777  2.094345e-01   0.003124
```

1.6.1 Describe your results.

[6]:

```python
regressors[regressors["P-value"] > 0.05]
```

[6]:

```
           Coefficient   P-value  R-Squared
Regressor
chas         -1.892777  0.209435   0.003124
```

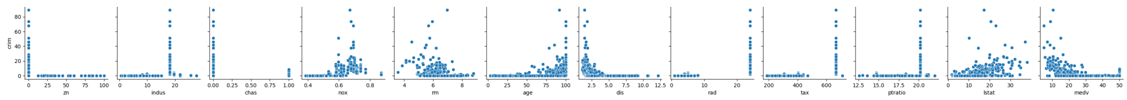
- From the above, we see that *chas* (Charles River dummy variable whether tract bounds river or not) is the only regressor that is not statistically significant in the simple linear regressions of each variable for *crim*.

1.6.2 In which of the models is there a statistically significant association between the predictor and the response?

- All of the variables except *chas* are statistically significant in a regression for *crim*.

1.6.3 Create some plots to back up your assertions.

[7]:
```
col_names = list(Boston.columns.values)
col_names.remove("crim")
sns.pairplot(Boston, x_vars=col_names,y_vars="crim");
```



1.7 (b) Fit a multiple regression model to predict the response using all of the predictors.

[8]:
```
def regress_on_all_predictors(data=None,response=None):
  if (data is None or response is None):
    return None
  columns = list(Boston.columns.values)
  columns.remove(response)
  formula = response + " ~ " + " + ".join(columns)
  model = smf.ols(formula, data=data)
  results = model.fit()
  results_df = pd.DataFrame({"Regressor": columns,
                             "Coefficient": results.params[1:],
                             "P-value": results.pvalues[1:],
                            "R-Squared": results.rsquared})
  results_df.set_index(["Regressor"],inplace=True)
  return results_df

all_regressors = regress_on_all_predictors(data=Boston, response="crim")
```

[8]:

Regressor	Coefficient	P-value	R-Squared
zn	0.045710	1.534403e-02	0.449338
indus	-0.058350	4.857094e-01	0.449338
chas	-0.825378	4.858406e-01	0.449338
nox	-9.957587	6.036986e-02	0.449338
rm	0.628911	3.007385e-01	0.449338
age	-0.000848	9.623231e-01	0.449338
dis	-1.012247	3.725942e-04	0.449338
rad	0.612465	8.588123e-12	0.449338
tax	-0.003776	4.657565e-01	0.449338
ptratio	-0.304073	1.033932e-01	0.449338
lstat	0.138801	6.739844e-02	0.449338
medv	-0.220056	2.605302e-04	0.449338

1.8 Describe your results.

[9]:
```
all_regressors[all_regressors["P-value"] > 0.05]
```

[9]:

Regressor	Coefficient	P-value	R-Squared
indus	-0.058350	0.485709	0.449338
chas	-0.825378	0.485841	0.449338
nox	-9.957587	0.060370	0.449338
rm	0.628911	0.300738	0.449338
age	-0.000848	0.962323	0.449338
tax	-0.003776	0.465757	0.449338
ptratio	-0.304073	0.103393	0.449338
lstat	0.138801	0.067398	0.449338

1.9 For which predictors can we reject the null hypothesis $H_0 : \beta_j = 0$?

[10]:
```
all_regressors[all_regressors["P-value"] < 0.05]
```

[10]:

Regressor	Coefficient	P-value	R-Squared
zn	0.045710	1.534403e-02	0.449338
dis	-1.012247	3.725942e-04	0.449338
rad	0.612465	8.588123e-12	0.449338
medv	-0.220056	2.605302e-04	0.449338

1.10 (c) How do your results from (a) compare to your results from (b)?

- In the multilinear regression model over all regressors, we discover that only *zn*, *dis*, *rad* and *medv* are statistically significant and the rest aren't.
- This implies that there is multicollinearity in the data.

1.10.1 Create a plot displaying the univariate regression coefficients from (a) on the x-axis, and the multiple regression coefficients from (b) on the y-axis. That is, each predictor is displayed as a single point in the plot. Its coefficient in a simple linear regression model is shown on the x-axis, and its coefficient estimate in the multiple linear regression model is shown on the y-axis.

[11]:
```
# merge the two dataframes
combined = regressors.merge(all_regressors, left_index=True, right_index=True,␣
↪suffixes=["_simple", "_all"])
# select only the needed columns
combined = combined[["Coefficient_simple", "Coefficient_all"]]
```

[11]:

Regressor	Coefficient_simple	Coefficient_all
rad	0.617911	0.612465
tax	0.029742	-0.003776

```
lstat                 0.548805          0.138801
nox                  31.248531         -9.957587
indus                 0.509776         -0.058350
medv                 -0.363160         -0.220056
dis                  -1.550902         -1.012247
age                   0.107786         -0.000848
ptratio               1.151983         -0.304073
rm                   -2.684051          0.628911
zn                   -0.073935          0.045710
chas                 -1.892777         -0.825378
```

[12]:
```
import plotly.express as px
fig = px.scatter(combined, x="Coefficient_simple",␣
 ↪y="Coefficient_all",color=combined.index)
fig.show()
```

1.11 (d) Is there evidence of non-linear association between any of the predictors and the response?

1.11.1 To answer this question, for each predictor X, fit a model of the form $Y = \beta_0 + \beta_1 * X + \beta_2 * X^2 + \beta_3 * X^3 + \epsilon$

[13]:
```
def regress_non_linear_for_each_predictor(data=None,response=None):
  if (data is None or response is None):
    return None
  col_names = list(data.columns.values)
  col_names.remove(response)
  rows = []
  keys = []
  for col in col_names:
    formula = f"{response} ~ {col} + I({col} ** 2) + I({col} ** 3)"
    model = smf.ols(formula, data=data)
    results = model.fit()
    results_df = pd.DataFrame({"Coefficient": results.params.iloc[1:],
                              "P-value": results.pvalues.iloc[1:],
                             "R-Squared": results.rsquared})
    rows.append(results_df)
    keys.append(col)

  rows = pd.concat(rows,keys=keys)
  return rows

nonlinears = regress_non_linear_for_each_predictor(data=Boston,response="crim")
```

[13]:
```
                           Coefficient       P-value  R-Squared
zn      zn               -3.321884e-01  2.612296e-03   0.058242
        I(zn ** 2)        6.482634e-03  9.375050e-02   0.058242
```

```
         I(zn ** 3)       -3.775793e-05  2.295386e-01   0.058242
indus    indus            -1.965213e+00  5.297064e-05   0.259658
         I(indus ** 2)     2.519373e-01  3.420187e-10   0.259658
         I(indus ** 3)    -6.976009e-03  1.196405e-12   0.259658
chas     chas             -6.309255e-01  2.094345e-01   0.003124
         I(chas ** 2)     -6.309255e-01  2.094345e-01   0.003124
         I(chas ** 3)     -6.309255e-01  2.094345e-01   0.003124
nox      nox              -1.279371e+03  2.758372e-13   0.296978
         I(nox ** 2)       2.248544e+03  6.811300e-15   0.296978
         I(nox ** 3)      -1.245703e+03  6.961110e-16   0.296978
rm       rm               -3.915014e+01  2.117564e-01   0.067786
         I(rm ** 2)        4.550896e+00  3.641094e-01   0.067786
         I(rm ** 3)       -1.744770e-01  5.085751e-01   0.067786
age      age               2.736531e-01  1.426608e-01   0.174231
         I(age ** 2)      -7.229596e-03  4.737733e-02   0.174231
         I(age ** 3)       5.745307e-05  6.679915e-03   0.174231
dis      dis              -1.555435e+01  6.374792e-18   0.277825
         I(dis ** 2)       2.452072e+00  4.941214e-12   0.277825
         I(dis ** 3)      -1.185986e-01  1.088832e-08   0.277825
rad      rad               5.127360e-01  6.234175e-01   0.400037
         I(rad ** 2)      -7.517736e-02  6.130099e-01   0.400037
         I(rad ** 3)       3.208996e-03  4.823138e-01   0.400037
tax      tax              -1.533096e-01  1.097075e-01   0.368882
         I(tax ** 2)       3.608266e-04  1.374682e-01   0.368882
         I(tax ** 3)      -2.203715e-07  2.438507e-01   0.368882
ptratio  ptratio          -8.236054e+01  3.028663e-03   0.113782
         I(ptratio ** 2)   4.635347e+00  4.119552e-03   0.113782
         I(ptratio ** 3)  -8.476032e-02  6.300514e-03   0.113782
lstat    lstat            -4.490656e-01  3.345300e-01   0.217932
         I(lstat ** 2)     5.577942e-02  6.458736e-02   0.217932
         I(lstat ** 3)    -8.573703e-04  1.298906e-01   0.217932
medv     medv             -5.094831e+00  2.637707e-28   0.420200
         I(medv ** 2)      1.554965e-01  3.260523e-18   0.420200
         I(medv ** 3)     -1.490103e-03  1.046510e-12   0.420200
```

[14]:
```
nonlinears[nonlinears["P-value"] <= 0.05].filter(like="I(", axis=0)
```

[14]:
```
                          Coefficient       P-value  R-Squared
indus    I(indus ** 2)       0.251937  3.420187e-10   0.259658
         I(indus ** 3)      -0.006976  1.196405e-12   0.259658
nox      I(nox ** 2)      2248.544053  6.811300e-15   0.296978
         I(nox ** 3)     -1245.702874  6.961110e-16   0.296978
age      I(age ** 2)        -0.007230  4.737733e-02   0.174231
         I(age ** 3)         0.000057  6.679915e-03   0.174231
dis      I(dis ** 2)         2.452072  4.941214e-12   0.277825
         I(dis ** 3)        -0.118599  1.088832e-08   0.277825
ptratio  I(ptratio ** 2)     4.635347  4.119552e-03   0.113782
```

```
        I(ptratio ** 3)    -0.084760  6.300514e-03   0.113782
medv    I(medv ** 2)        0.155496  3.260523e-18   0.420200
        I(medv ** 3)       -0.001490  1.046510e-12   0.420200
```

- Thus, we can see that regressors *indus*, *nox*, *age*, *dis*, *ptratio* and *medv* are non-linear after fitting a cubic regression for each of these individually.

[15]:
```
allDone();
```

```
<IPython.lib.display.Audio object>
```