

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

# Table of contents

|   |    |
|---|----|
| Import notebook funcs . . . . .   | 1  |
| Import userfuncs . . . . .  | 1  |
| Import libraries . . . . .  | 2  |
| We will now try to predict per capita crime rate using the other variables in this data set. . .  | 5  |
| In other words, per capita crime rate (crim) is the response, and the other variables are the predictors. . . . .   | 5  |
| (a) For each predictor, fit a simple linear regression model to predict the response. . . . .   | 5  |
| Describe your results. . . . .  | 6  |
| In which of the models is there a statistically significant association between the predictor and the response? . . . . .   | 6  |
| Create some plots to back up your assertions. . . . .   | 6  |
| (b) Fit a multiple regression model to predict the response using all of the predictors. . . . .  | 7  |
| Describe your results. . . . .  | 8  |
| For which predictors can we reject the null hypothesis $H_0 : \beta_j = 0$ ? . . . . .  | 8  |
| (c) How do your results from (a) compare to your results from (b)? . . . . .  | 8  |
| 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. . . . . | 9  |
| (d) Is there evidence of non-linear association between any of the predictors and the response? 10  |    |
| To answer this question, for each predictor X, fit a model of the form $Y = \beta_0 + \beta_1 * X + \beta_2 * X^2 + \beta_3 * X^3 + \epsilon$ . . . . .   | 10 |

## Import notebook funcs

```
from notebookfuncs import *
```

## Import userfuncs

```
from userfuncs import *
```


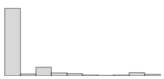
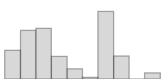
## Import libraries

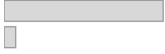
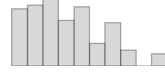
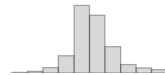

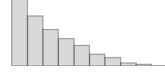
```
from ISLP import load_data
from summarytools import dfSummary
import seaborn as sns
```


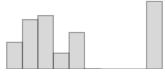
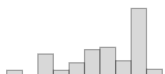
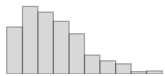
```
Boston = load_data("Boston")
dfSummary(Boston)
```

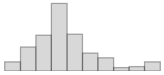
Table 1: **Data Frame Summary**

Boston  
Dimensions: 506 x 13  
Duplicates: 0

| No | Variable                  | Stats / Values  | Freqs / (% of Valid) | Graph   | Missing     |
|----|---------------------------|---|----------------------|---|-------------|
| 1  | <b>crim</b><br>[float64]  | Mean (sd) :<br>3.6 (8.6)<br>min < med <<br>max:<br>0.0 < 0.3 <<br>89.0<br>IQR (CV) :<br>3.6 (0.4)     | 504 distinct values  |    | 0<br>(0.0%) |
| 2  | <b>zn</b><br>[float64]    | Mean (sd) :<br>11.4 (23.3)<br>min < med <<br>max:<br>0.0 < 0.0 <<br>100.0<br>IQR (CV) :<br>12.5 (0.5) | 26 distinct values   |  | 0<br>(0.0%) |
| 3  | <b>indus</b><br>[float64] | Mean (sd) :<br>11.1 (6.9)<br>min < med <<br>max:<br>0.5 < 9.7 <<br>27.7<br>IQR (CV) :<br>12.9 (1.6)   | 76 distinct values   |  | 0<br>(0.0%) |

| No | Variable                | Stats / Values   | Freqs / (% of Valid)     | Graph   | Missing     |
|----|-------------------------|--|--------------------------|---|-------------|
| 4  | <b>chas</b><br>[int64]  | 1. 0<br>2. 1   | 471 (93.1%)<br>35 (6.9%) |    | 0<br>(0.0%) |
| 5  | <b>nox</b><br>[float64] | Mean (sd) :<br>0.6 (0.1)<br>min < med <<br>max:<br>0.4 < 0.5 <<br>0.9<br>IQR (CV) :<br>0.2 (4.8)       | 81 distinct values       |    | 0<br>(0.0%) |
| 6  | <b>rm</b><br>[float64]  | Mean (sd) :<br>6.3 (0.7)<br>min < med <<br>max:<br>3.6 < 6.2 <<br>8.8<br>IQR (CV) :<br>0.7 (8.9)       | 446 distinct values      |    | 0<br>(0.0%) |
| 7  | <b>age</b><br>[float64] | Mean (sd) :<br>68.6 (28.1)<br>min < med <<br>max:<br>2.9 < 77.5 <<br>100.0<br>IQR (CV) :<br>49.0 (2.4) | 356 distinct values      |  | 0<br>(0.0%) |
| 8  | <b>dis</b><br>[float64] | Mean (sd) :<br>3.8 (2.1)<br>min < med <<br>max:<br>1.1 < 3.2 <<br>12.1<br>IQR (CV) :<br>3.1 (1.8)      | 412 distinct values      |  | 0<br>(0.0%) |

| No | Variable                    | Stats / Values   | Freqs / (% of Valid)  | Graph   | Missing     |
|----|-----------------------------|--|---|---|-------------|
| 9  | <b>rad</b><br>[int64]       | 1. 24<br>2. 5<br>3. 4<br>4. 3<br>5. 6<br>6. 2<br>7. 8<br>8. 1<br>9. 7  | 132 (26.1%)<br>115 (22.7%)<br>110 (21.7%)<br>38 (7.5%)<br>26 (5.1%)<br>24 (4.7%)<br>24 (4.7%)<br>20 (4.0%)<br>17 (3.4%) |    | 0<br>(0.0%) |
| 10 | <b>tax</b><br>[int64]       | Mean (sd) :<br>408.2 (168.5)<br>min < med <<br>max:<br>187.0 < 330.0<br>< 711.0<br>IQR (CV) :<br>387.0 (2.4) | 66 distinct values  |    | 0<br>(0.0%) |
| 11 | <b>ptratio</b><br>[float64] | Mean (sd) :<br>18.5 (2.2)<br>min < med <<br>max:<br>12.6 < 19.1 <<br>22.0<br>IQR (CV) :<br>2.8 (8.5)         | 46 distinct values  |  | 0<br>(0.0%) |
| 12 | <b>lstat</b><br>[float64]   | Mean (sd) :<br>12.7 (7.1)<br>min < med <<br>max:<br>1.7 < 11.4 <<br>38.0<br>IQR (CV) :<br>10.0 (1.8)         | 455 distinct values   |  | 0<br>(0.0%) |

| No | Variable                 | Stats / Values  | Freqs / (% of Valid) | Graph   | Missing     |
|----|--------------------------|---|----------------------|---|-------------|
| 13 | <b>medv</b><br>[float64] | Mean (sd) :<br>22.5 (9.2)<br>min < med <<br>max:<br>5.0 < 21.2 <<br>50.0<br>IQR (CV) :<br>8.0 (2.4) | 229 distinct values  |  | 0<br>(0.0%) |

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

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

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

```
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:
        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")
```

| Regressor | Coefficient | P-value      | R-Squared |
|-----------|-------------|--------------|-----------|
| 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  |

**Describe your results.**

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

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

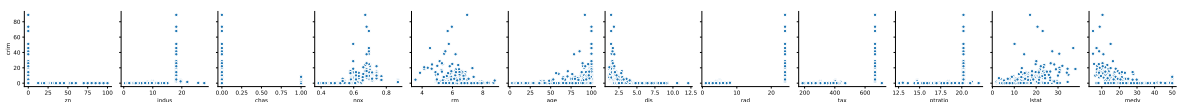
- From the above, we see that *chas* (Charles River dummy variable whether tract bounds river or not) is the only regressor that is not statistically significant in the simple linear regressions of each variable for *crim*.

**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*.

**Create some plots to back up your assertions.**

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



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

```
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")
```

| 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  |



Describe your results.

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

| 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  |

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

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

| 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  |

(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.

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.

```
# 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"]]
```

|           | Coefficient_simple | Coefficient_all |
|-----------|--------------------|-----------------|
| Regressor |                    |                 |
| 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       |

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

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): text/html

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

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

```
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")
```

---

|       |   |
|-------|---|
| zn    | zn<br>I(zn ** 2)<br>I(zn ** 3)          |
| indus | indus<br>I(indus ** 2)<br>I(indus ** 3) |
| chas  | chas<br>I(chas ** 2)<br>I(chas ** 3)    |
| nox   | nox<br>I(nox ** 2)<br>I(nox ** 3)       |
| rm    | rm<br>I(rm ** 2)<br>I(rm ** 3)          |

---

|         |   |
|---------|---|
| age     | age<br>I(age ** 2)<br>I(age ** 3)             |
| dis     | dis<br>I(dis ** 2)<br>I(dis ** 3)             |
| rad     | rad<br>I(rad ** 2)<br>I(rad ** 3)             |
| tax     | tax<br>I(tax ** 2)<br>I(tax ** 3)             |
| ptratio | ptratio<br>I(ptratio ** 2)<br>I(ptratio ** 3) |
| lstat   | lstat<br>I(lstat ** 2)<br>I(lstat ** 3)       |
| medv    | medv<br>I(medv ** 2)<br>I(medv ** 3)          |

---

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

---

|         |                                    |
|---------|------------------------------------|
| indus   | I(indus ** 2)<br>I(indus ** 3)     |
| nox     | I(nox ** 2)<br>I(nox ** 3)         |
| age     | I(age ** 2)<br>I(age ** 3)         |
| dis     | I(dis ** 2)<br>I(dis ** 3)         |
| ptratio | I(ptratio ** 2)<br>I(ptratio ** 3) |
| medv    | I(medv ** 2)<br>I(medv ** 3)       |

---

- 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.

```
allDone();
```

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