COM S 474/574 Spring 2020

Introduction to Machine Learning

Homework 1

1 Directions:

- **Due: Sunday February 16, 2020 at 10:00pm** Late submissions will be accepted for 12 hours after that time, with a 15% penalty. (the enforcement is strict, beginning at 10:01pm, except for extreme situations; having a poor wifi connection or minor computer problems is not sufficient for the penalty to be waived.)
- Upload the homework to Canvas as a pdf file. Written responses must be typed. Text should be Times New Roman font size 12 or similar size in other fonts.

  Unless specified otherwise, plots should be computer generated (make sure axis ranges are appropriate, tick marks and labels are legible, etc.). You can use Microsoft Word (or similar) or latex, then convert to pdf.
- Any non-administrative questions must be asked in office hours or (if a brief response is sufficient) Piazza.
- We recommend using Python with the machine learning library scikit-learn. Anaconda packages that library and other useful ones (like Matplotlib, Numpy, Statsmodels, and Pandas) together with both Spyder (a nice IDE similar to Matlab or RStudio) and Jupyter notebook. We recommend getting Python 3.7:

  https://www.anaconda.com/distribution/

2 Problems

Problem 1. [60 points] For this problem, you will model feature y using polynomials of x up to order 30,

$$y = \sum_{i=0}^{30} a_i x^i \quad .$$

In addition to looking at issues like training error, test error, overfitting, and so on, we will also explore how the amount of data (# of samples n) matter.

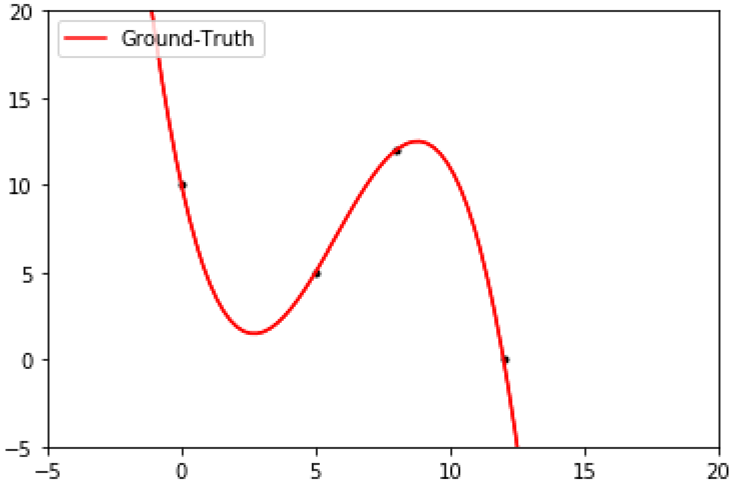
(a). First, let's generate a *ground-truth* model that the data comes from for reference (in practice we usually won't have this, but this will help us gain intuition). Recall that if we have p + 1 data points, we can fit that perfectly with a pth order polynomial. Fit a third order polynomial to the following four points

(0,10) (5,5) (8,12) (12,0).

You can fit it using linear regression (such as with scikit-learn https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
LinearRegression.html

and treating x^0, x^1, x^2, and x^3 as separate covariates.

(b). Make a plot of those points (big black circles) and the fit polynomial (red curve) with an x range of [−5,20] and a y range of [−5,20] to confirm the fit.



(c). The scikit-learn built-in linear regression function (and most built-in functions for other languages) use mean square error (MSE) as the only, or at least default, total-loss function. In 1-3 sentences, explain why the loss-function matters or not *for that fit specifically*. In other words, if you used a different total-loss function in part (a)., like mean absolute error or a weighted average of an asymmetric loss function, would you have gotten a different polynomial? Yes, different loss functions would generate a different polynomial.

(d). Now let's generate some data using that polynomial.

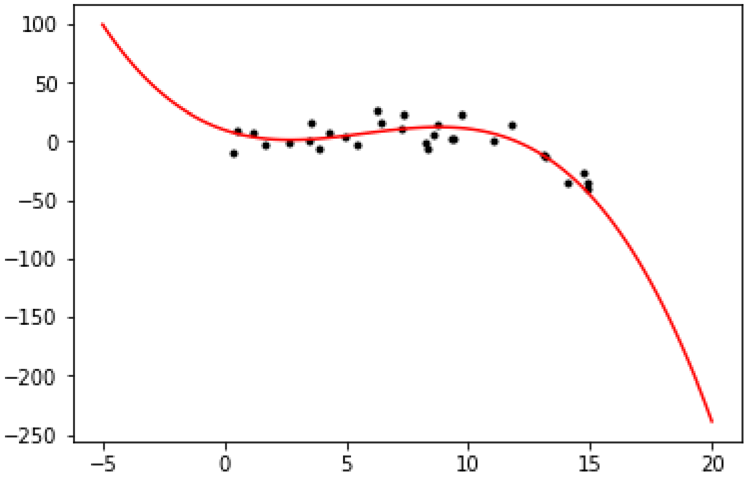
- First, there might be some distribution to the x values in the data. For this problem, we'll use the uniform distribution over the interval [0,15]. Numpy has a function for generating uniform random variables:

https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy. random.uniform.html

Generate n = 30 random variables. Each of these is an x value.

- Now for the y values. Generate y using the polynomial you fit in part (a). *plus* independent and identically distributed noise. Use the normal distribution, N(0,10), https://numpy.org/devdocs/reference/random/generated/numpy.random.Generator.normal.html#numpy.random.Generator.normal

- Now make a plot of the fit polynomial (red) and the n = 30 data points (black circles) you generated. The data should follow the curves of the polynomial but be scattered about it.

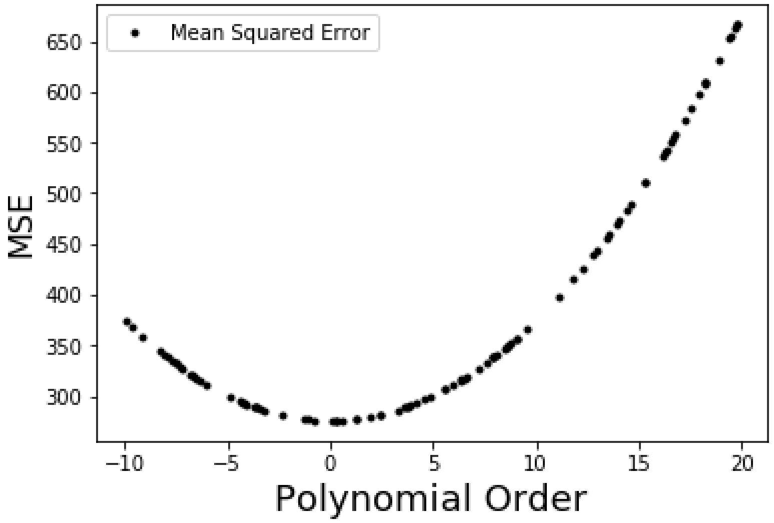


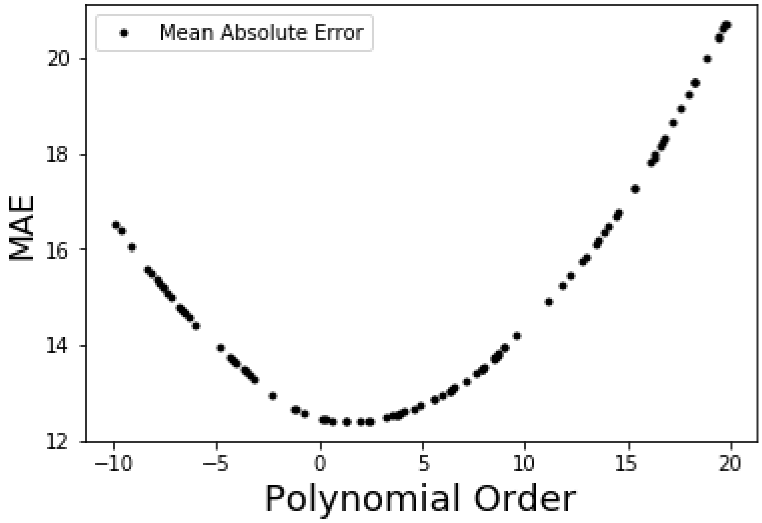
(e). We are now ready to start fitting models. First, let's look at how well constant models $y = a_0$ fit the data under different loss functions. Make a plot with a_0-values as the horizontal axis, ranging from [−10,20] with 100 values linearly spaced, and the total-loss along the vertical axis, for each of the total-loss functions

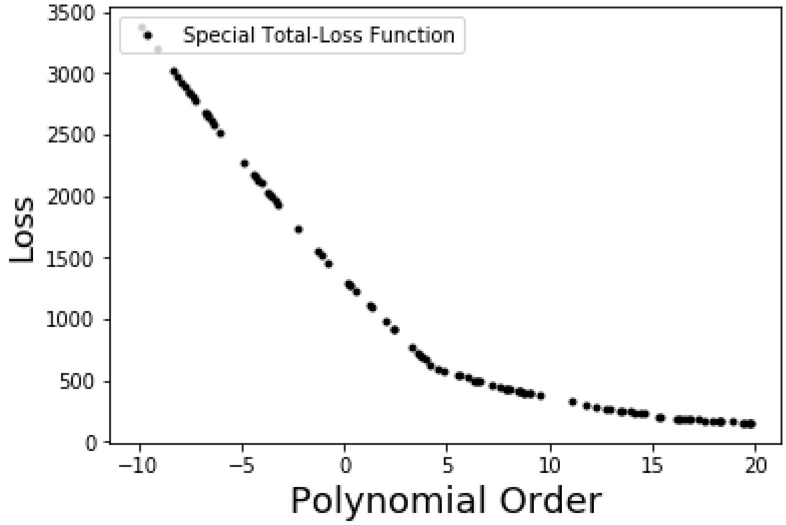
- mean squared error (MSE)$\frac{1}{n}\sum_{i=1}^{n}|res(i)|^2$
- mean absolute error (MAE) $\frac{1}{n}\sum_{i=1}^{n}|res(i)|^1$
- a special total-loss function $\sum_{i=1}^{n}\frac{1}{|x(i)-5|+0.01}l(res(i))$ where $x(i)$ is the x-feature value of sample i and the loss function $l(\cdot)$ is

$$l(res) = \begin{cases} -\frac{1}{5}res & \text{if } res < 0 \\ 10res & \text{if } res \geq 0 \end{cases}.$$

where $res(i) = y(i) - y(i)$ denotes the residual of the ith sample.


Mean Squared Error
MSE
650
600
550
500
450
400
350
300
−10
−5
0
5
10
15
20
Polynomial Order


Mean Absolute Error
MAE
20
18
16
14
12
−10
−5
0
5
10
15
20
Polynomial Order


Special Total-Loss Function
Loss
3500
3000
2500
2000
1500
1000
500
0
−10
−5
0
5
10
15
20
Polynomial Order

(f). In 2-4 sentences, explain

i. does the loss function $l(\cdot)$ prefer over-estimating or under-estimating y values?

The loss function l(.) prefers overestimating.

ii. does the loss function put more emphasis or less emphasis on points close to $x = 5$?

Yes, anything less than 5 gets weighed harder(higher loss)

(g). Calculate the mean and the median of the y-values for the data. Do you notice anything special about MAE and MSE minimizers? Explain in 2-5 sentences why that makes sense (or not) based on how residual magnitudes get penalized. Alternatively, you can use calculus to prove that they make sense (or not); i.e. you can solve for the minimizers.

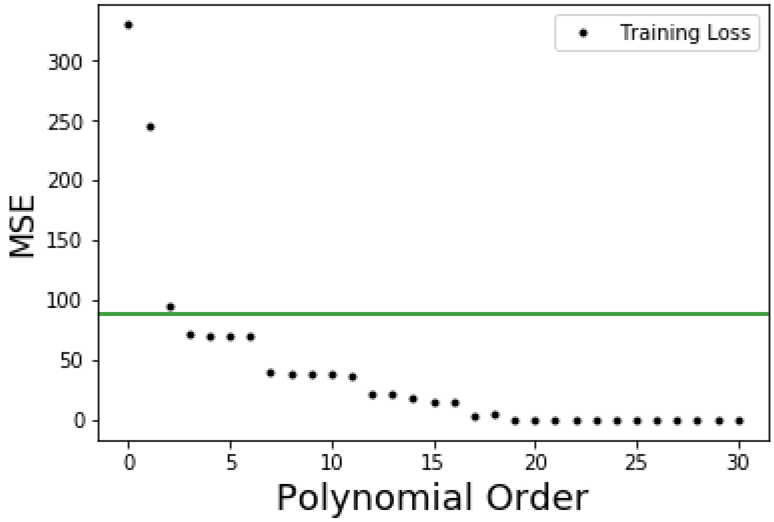
Mean: 0.0044596: The mean of the data would minimize the MSE. This can be seen when the curve is at its lowest.

Median: 1.7465 The median minimizes the sum of the absolute deviations. This can be seen when the curve is at its lowest.

(h). Now let's look at higher-order polynomials. Treating each x^i as a different covariate, we will fit the data with polynomials from order 0 to order 30. Caution: Some regression functions return wrong answers when the number of coefficients approaches the number of samples. If your results make sense for polynomials up to 10 or so, but do not make sense for higher orders, that's fine, submit what you have. Since MSE is the built-in total-loss function, we will use that (though keep in mind we could use others if someone had already coded the fitting procedure).

i. First, to evaluate over-fitting, we will use validation. Use the first 20 samples for fitting, reserving the rest as a validation set.

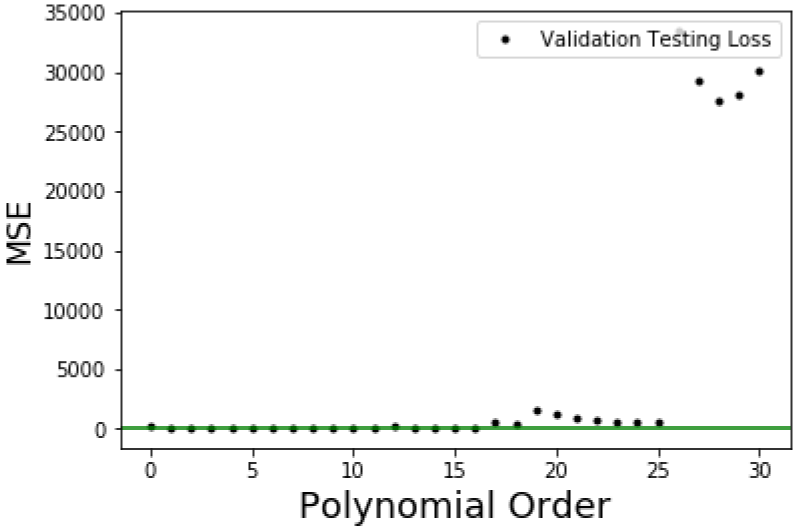
ii. Make a plot with title "Training Loss" plotting the MSE for the 20 samples used to fit for each polynomial for $p = 0$ to $p = 30$. In this plot, include a green horizontal line for the ground-truth model's MSE on the training data.



iii. In 2-5 sentences, comment on how the training loss curve behaves, such as where it has a steep slope, where it has a shallow slope, where it goes to 0. Do not simply describe the plot, but explain why it is that way, using knowledge of the ground-truth model.

The plot goes down steeply until around 15th order polynomial, then gradually is sinking towards 0. As previously stated, if there is $p + 1$ data points, they can be fit perfectly with a pth order polynomial. Because of this, a polynomial order of close to 19 or above should be approaching 0 MSE.

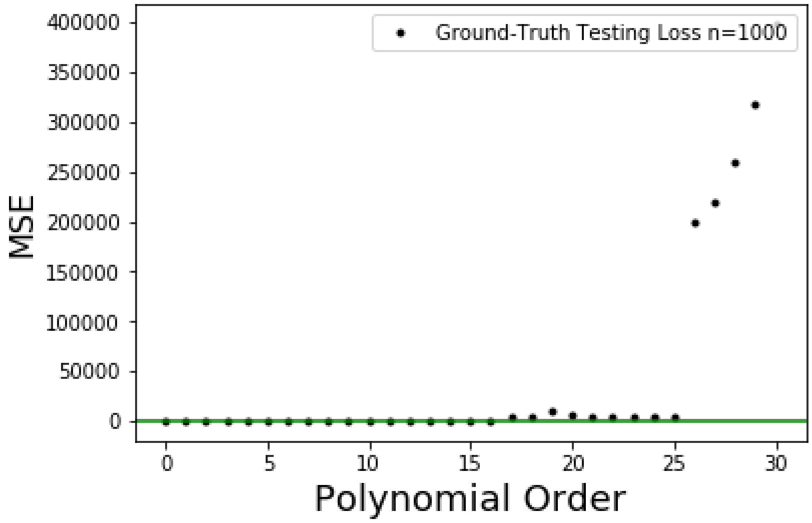
iv. Make another plot with the same axes, titled "Validation Testing Loss," plotting the MSE for the 10 samples held out. In this plot, include a green horizontal line for the ground-truth model's MSE on the validation data.



v. In 3-6 sentences, comment on how the validation testing loss curve behaves, explaining why it is that way, using knowledge of the ground-truth model. Your comment should also explain the similarities and differences between the training loss curve and the validation loss curve.

The MSE is not increasing much at first, then it increases steeply once the order of the polynomial is 25 and above. Both the training loss and validation curve change their behavior around the 20th order. The validation curve behaves this way because the data was overfit. By that I mean the high order polynomials fit the training data nice, however, the same polynomials with high order(20 and above) were not good for fitting the validation data. The tall peaks of a high order polynomials would create a big MSE of they corresponded to a data point that was far from it in the validation data.

vi. Now generate a new data-set with n = 1000. Make another plot, titled "GroundTruth Testing Loss n = 1000" with the MSE of the polynomials fitted to the n = 30 data set evaluated on this new, larger data-set. (This is a similar exercise to (h).iv except you are using a much more data) Use the same axes as the other plots. (keep in mind that in practice, we cannot do this since we won't know the ground truth.) In this plot, include a green horizontal line for the ground-truth model's MSE on this n = 1000 sample data-set.



vii. In 2-4 sentences, comment on how well (or not) the validation set testing loss approximates the ground-truth testing loss. It approximates it pretty well. Both curves are showing a big increase in MSE at around the 25th order and both are steady before that. The MSEs for the Ground truth are on a greater scale than the validation MSEs.

viii. Next, let's look at model complexity instead of using validation data. Fit polynomials using all n = 30 data points and keep track of the corresponding MSE.

Then calculate

$$\begin{aligned}\text{Total Complexity} &= \text{Total Training Loss} + \lambda \cdot \text{Model Complexity}\\ &= \quad MSE \quad + \quad \lambda p\end{aligned}$$

where n refers to the number of samples used to fit (so n = 30), p is the the model order, and λ is a threshold we choose. In class, for examples, we've used λ = 1 for illustration, but in practice we need to be a bit careful. We need to convert scales, like if we want to add

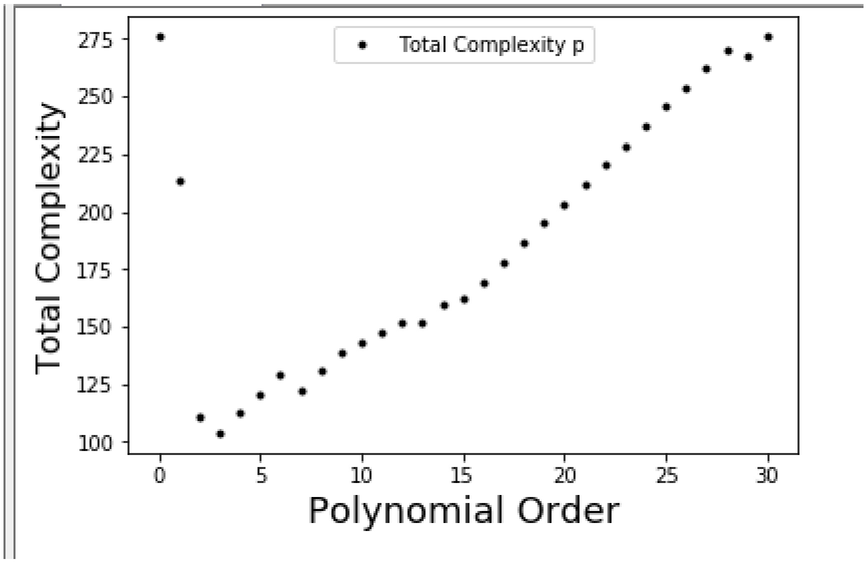
3.2 miles + 17 inches,

the summed number is not 20.2. Instead we need to make sure they are on the same *scale*. (Chapter 6.1.3 of the textbook describes a few λ values that have been derived by statisticians, such as Mallow's C_p, AIC, and BIC.) For us, let's pick a λ such that the model with p = 30 has the same total-complexity as the model with p = 0.

- Write what that λ should be in terms of p and the MSE of certain models. Use the notation $MSE(i)$ for the MSE for the fitted polynomial of order i.

  Lambda= (MSE(0)-MSE(30))/30

- Make a plot of the total-complexity as a function of the model order p. Use the title "Total Complexity p"

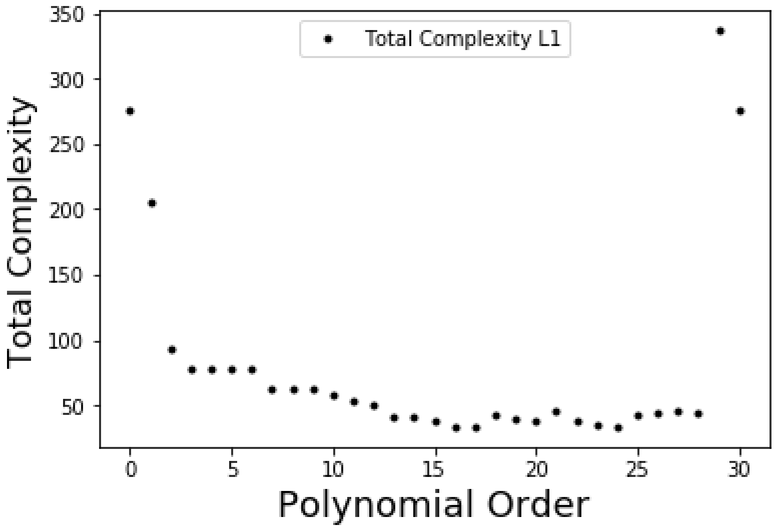


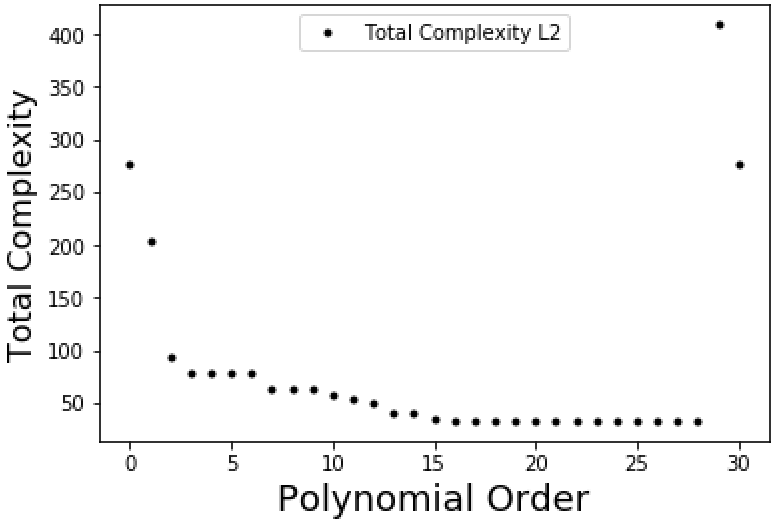
- In 2-5 sentances, explain the behavior of the total complexity curve with reference to the validation testing curve and the ground truth testing curves.

The curve decreases to till about the 4hh order then increases. This and the ground truth and validation plots show the consequences of overfitting. It results in increased complexity and increased MSE.

ix. We can also use other penalties. Reusing the fitting from part (h).viii, select a new λ and make new plots for penalties other than λp

•$\lambda_1 \sum_{i=0}^{p} |a_i|^1$, with plot titled "Total Complexity L1" • $\lambda_2 \sum_{i=0}^{p} |a_i|^2$ with plot titled "Total Complexity L2."





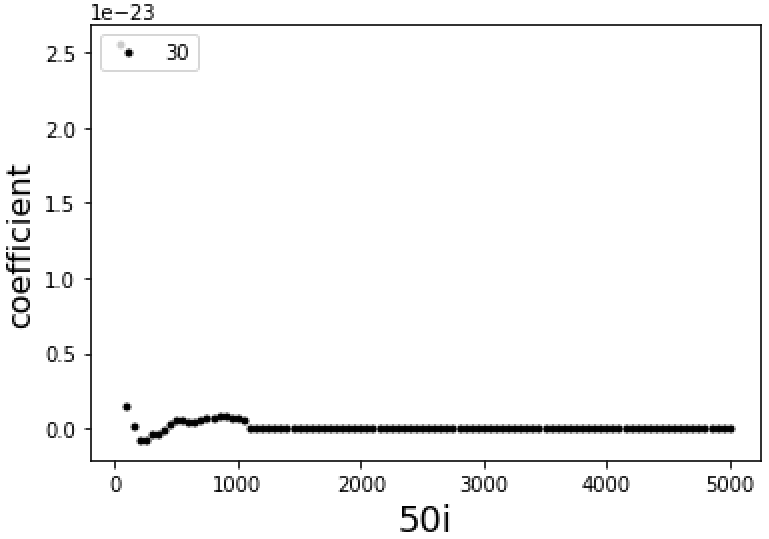
In 2-5 sentences, discuss why the curves behave the way that they do in comparison/contrast to the curves using validation testing data and ground-truth testing data. At the highest order polynomial, the complexity increases to that of the lowest. This is because of the relationahip between MSE and lambda in the equation of total complexity.

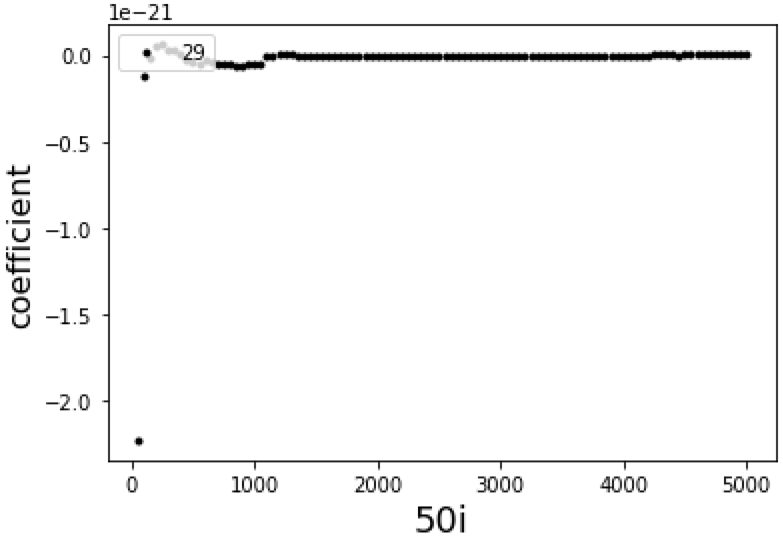
(i). Lastly, let's see how much getting more data helps to find the correct model.

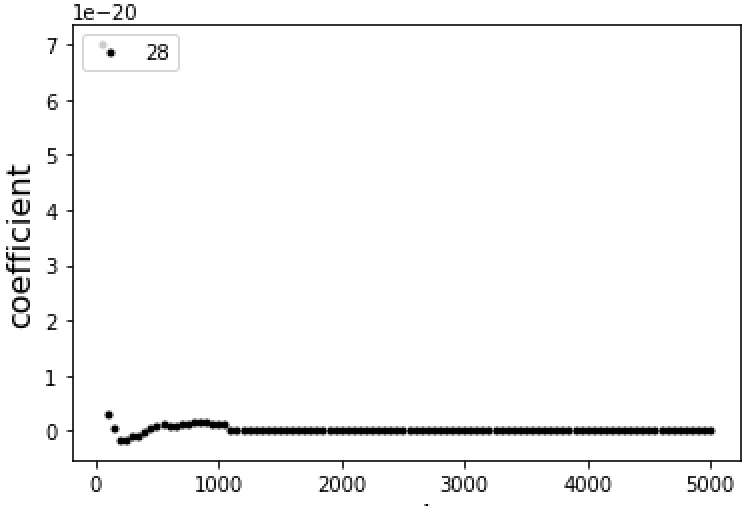
i. Generate n = 5000 samples which will be used for fitting.

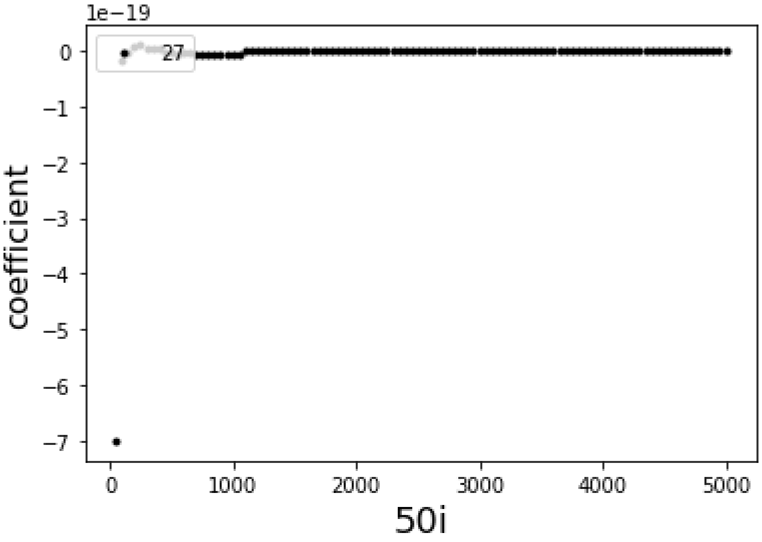
ii. For i = 1,...,100, fit the best p = 30 polynomial using n = 50i samples from the 5000 sample data set.

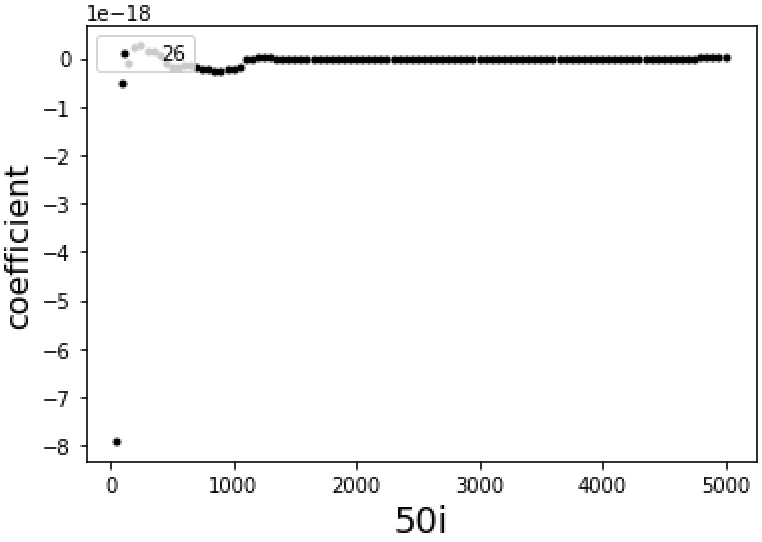
iii. Plot each coefficient {a_0,a_2,a_4,a_6,a_8} as a function of 50i in separate plots. (so one plot for a_0, etc). In each plot, draw a green line for the corresponding coefficient in the ground-truth polynomial.

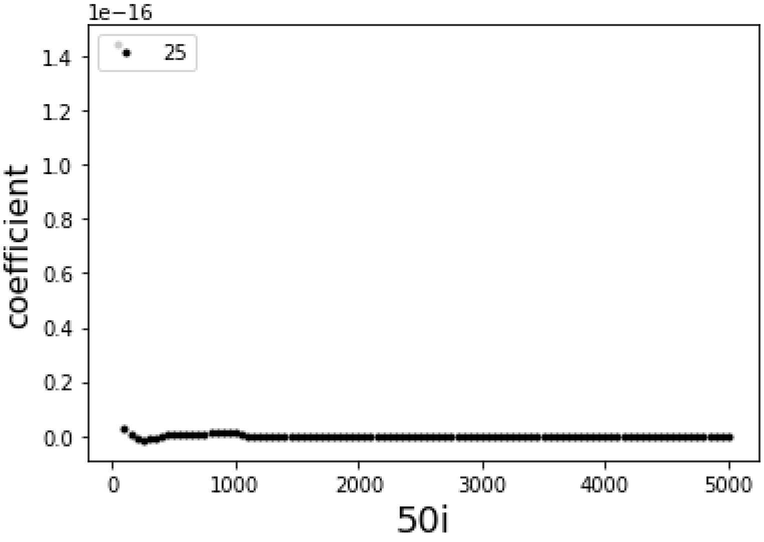

1e−23
30
2.5
2.0
1.5
1.0
0.5
0.0
coefficient
0
1000
2000
3000
4000
5000
50i

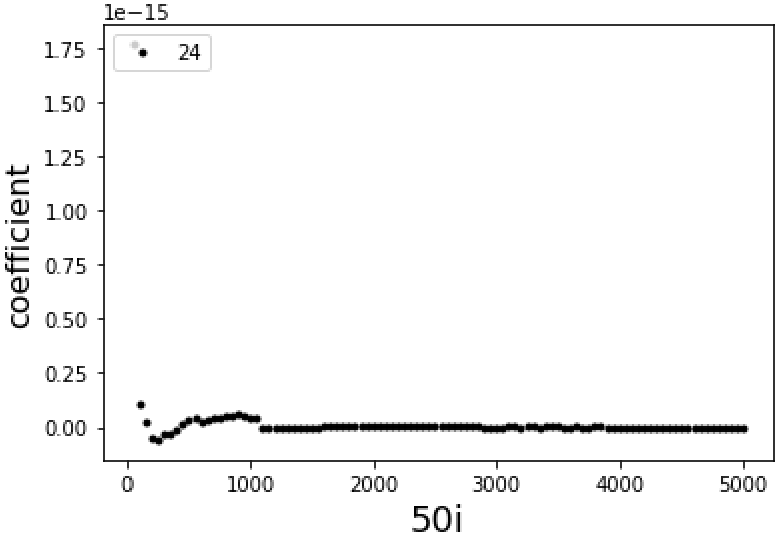

1e−21
29
0.0
−0.5
−1.0
−1.5
−2.0
coefficient
0
1000
2000
3000
4000
5000
50i

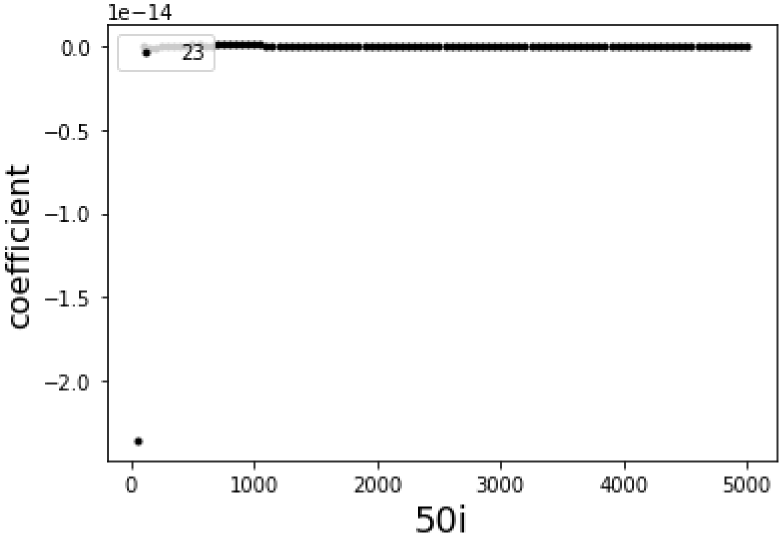

1e−20
28
7
6
5
4
3
2
1
0
coefficient
0
1000
2000
3000
4000
5000

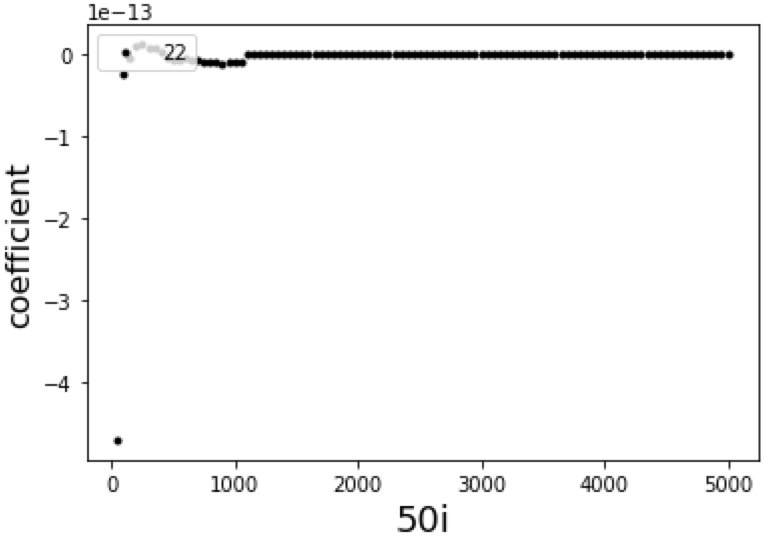

1e−19
27
0
−1
−2
−3
−4
−5
−6
−7
coefficient
0
1000
2000
3000
4000
5000
50i

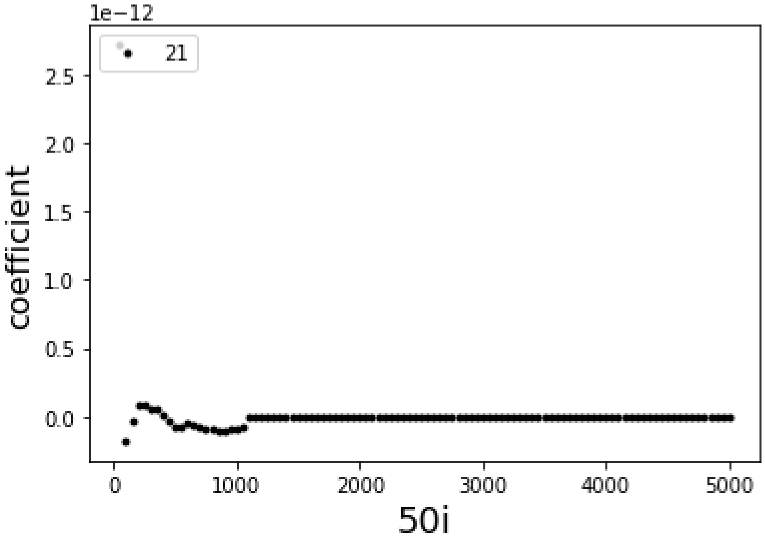

1e−18
26
0
−1
−2
−3
−4
−5
−6
−7
−8
coefficient
0
1000
2000
3000
4000
5000
50i

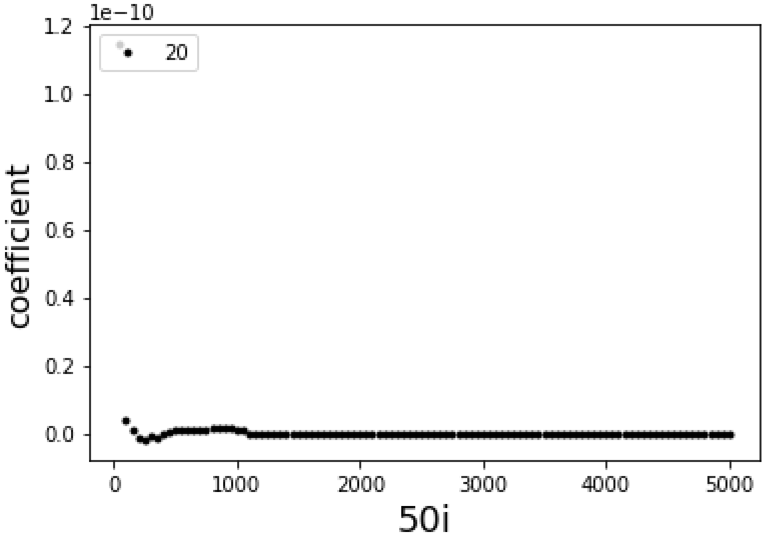

1e−16
25
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
coefficient
0
1000
2000
3000
4000
5000
50i

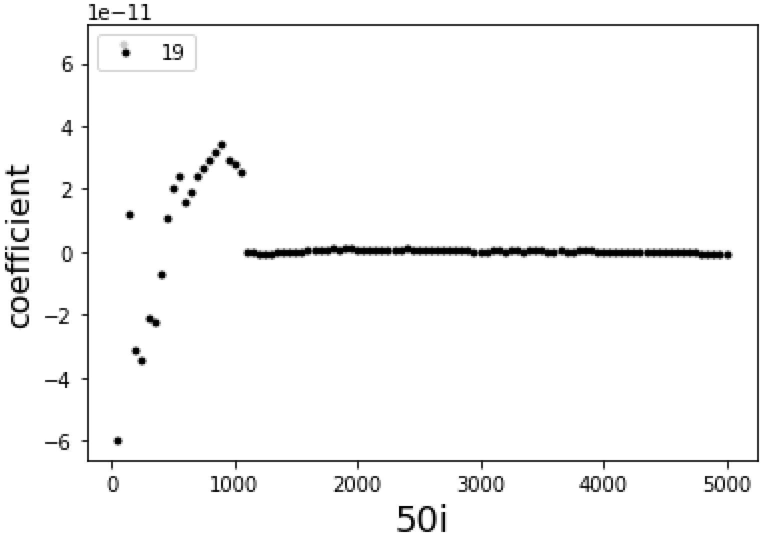

1e−15
24
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
coefficient
0
1000
2000
3000
4000
5000
50i

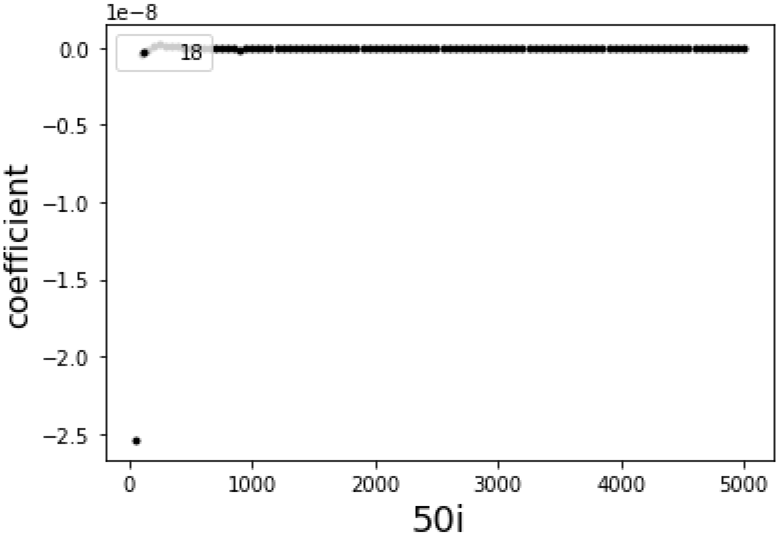

1e−14
23
0.0
−0.5
−1.0
−1.5
−2.0
coefficient
0
1000
2000
3000
4000
5000
50i

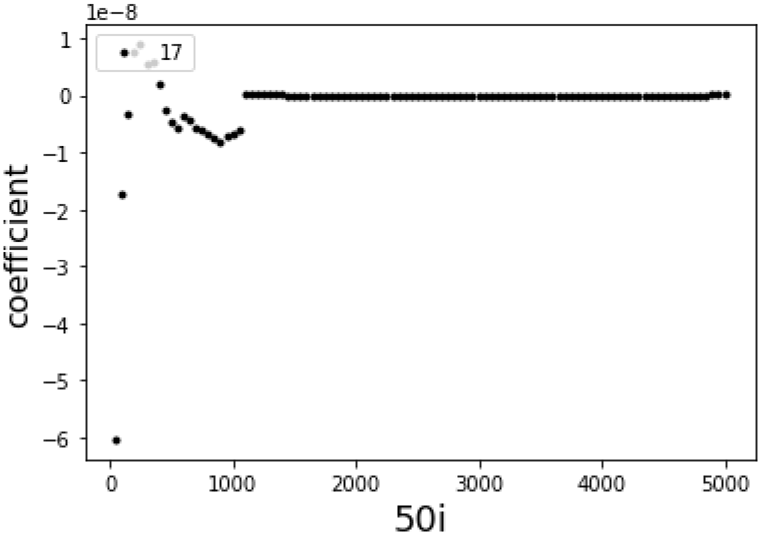

1e−13
22
0
−1
−2
−3
−4
coefficient
0
1000
2000
3000
4000
5000
50i

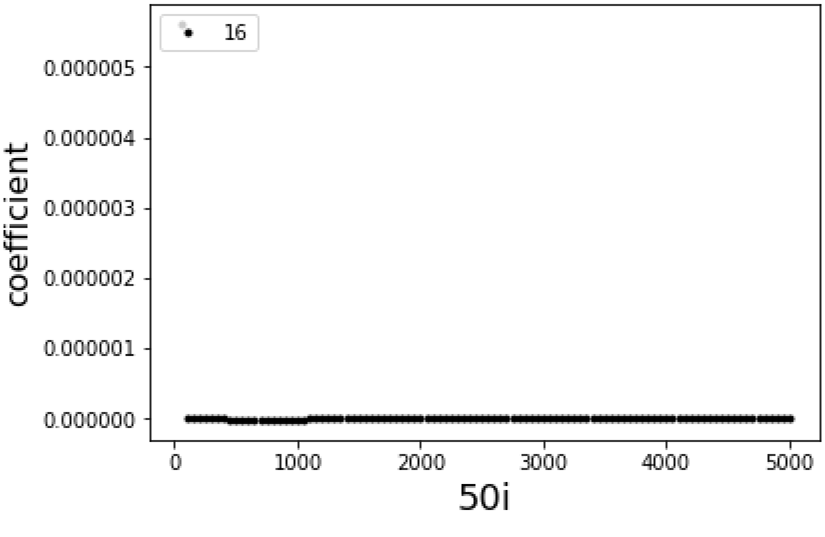

1e−12
21
2.5
2.0
1.5
1.0
0.5
0.0
coefficient
0
1000
2000
3000
4000
5000
50i

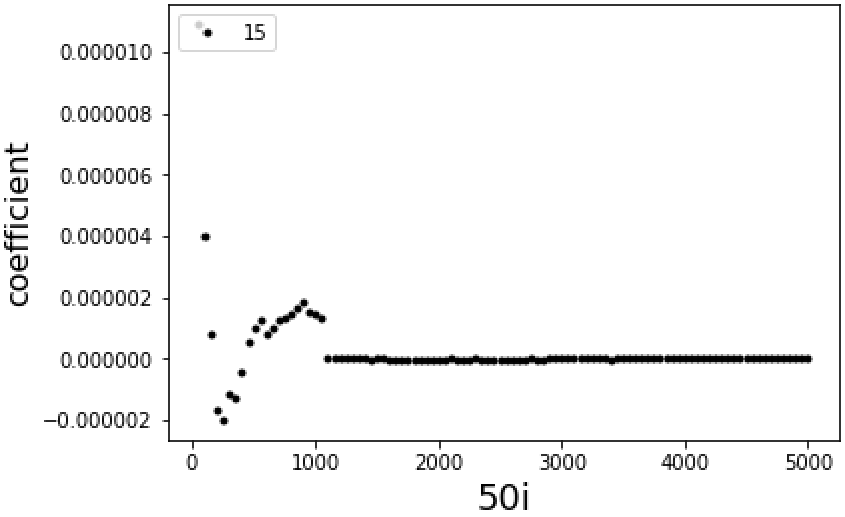

1e−10
20
1.2
1.0
0.8
0.6
0.4
0.2
0.0
coefficient
0
1000
2000
3000
4000
5000
50i

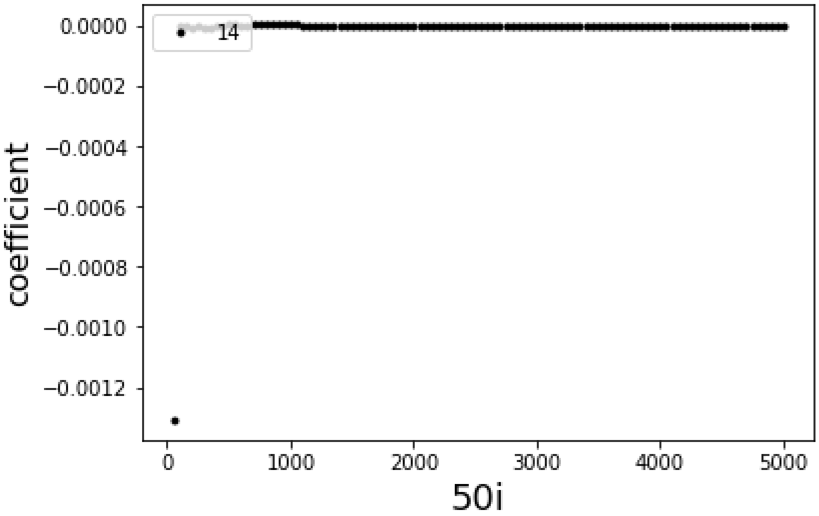

1e−11
19
6
4
2
0
−2
−4
−6
coefficient
0
1000
2000
3000
4000
5000
50i

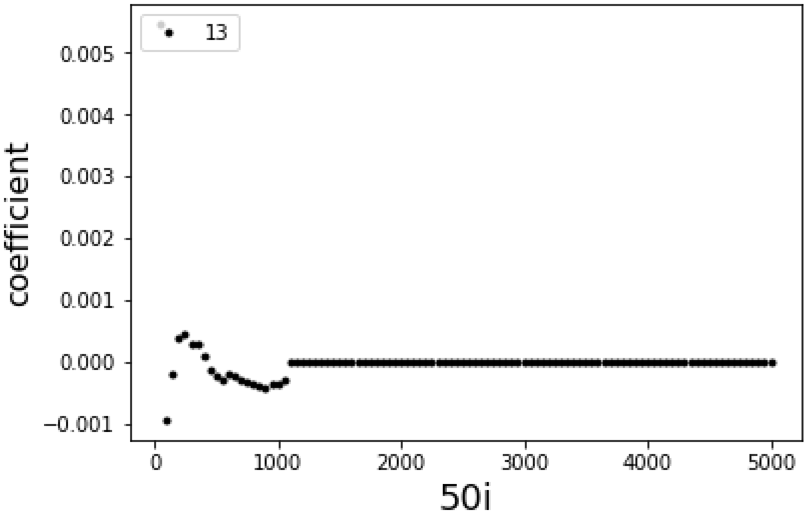

1e−8
18
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
coefficient
0
1000
2000
3000
4000
5000
50i

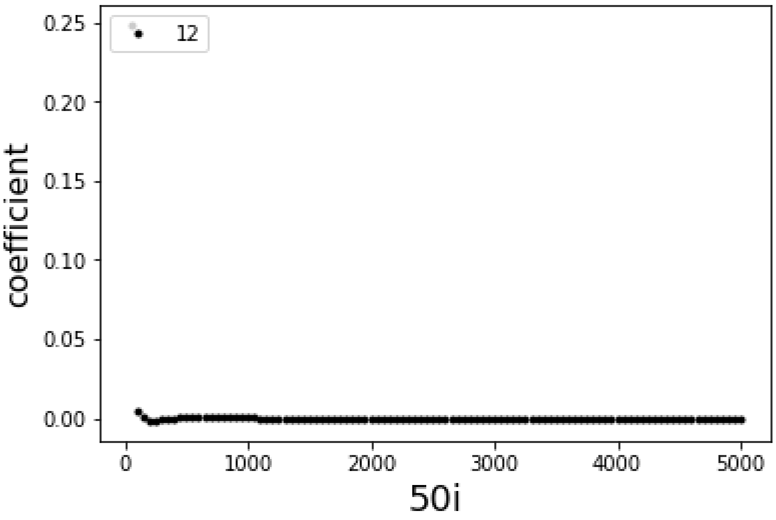

1e−8
17
1
0
−1
−2
−3
−4
−5
−6
coefficient
0
1000
2000
3000
4000
5000
50i

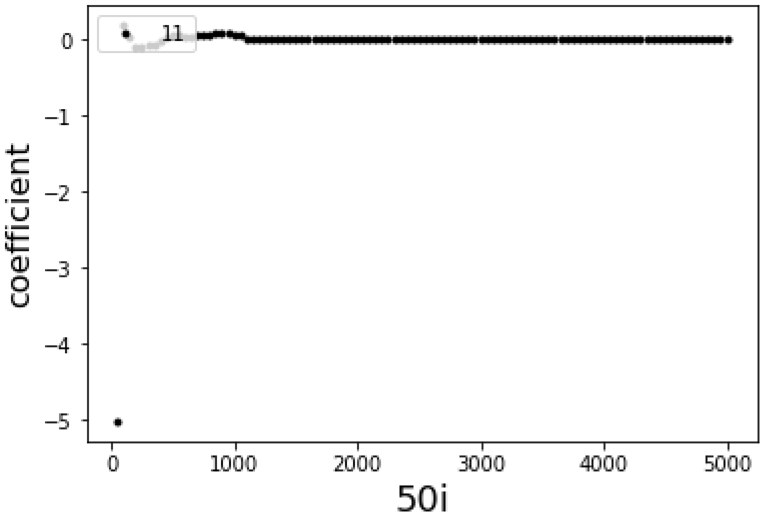

16
0.000005
0.000004
0.000003
0.000002
0.000001
0.000000
coefficient
0
1000
2000
3000
4000
5000
50i

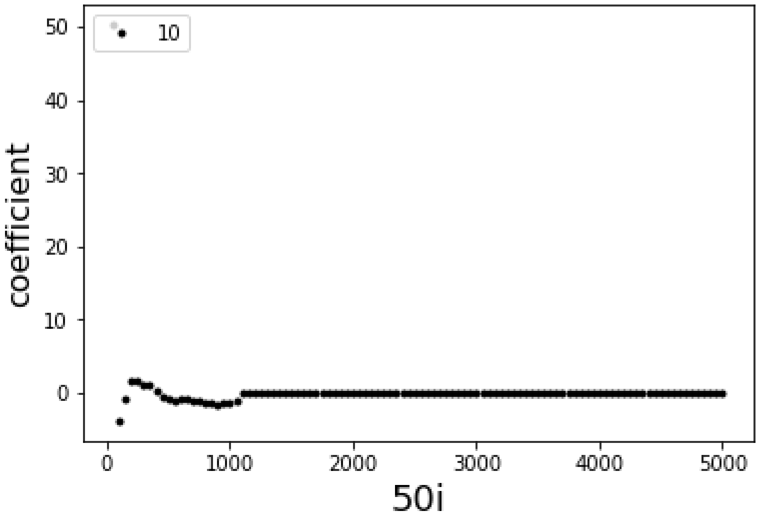

15
0.000010
0.000008
0.000006
0.000004
0.000002
0.000000
−0.000002
coefficient
0
1000
2000
3000
4000
5000
50i

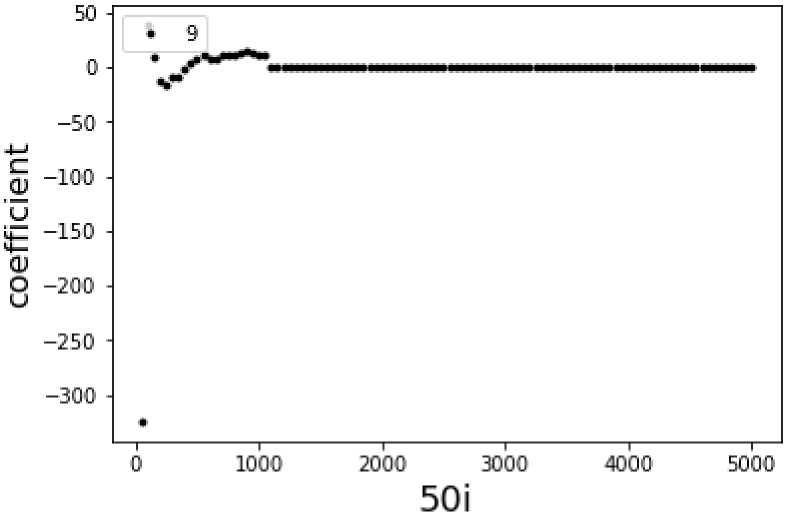

14
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
−0.0012
coefficient
0
1000
2000
3000
4000
5000
50i

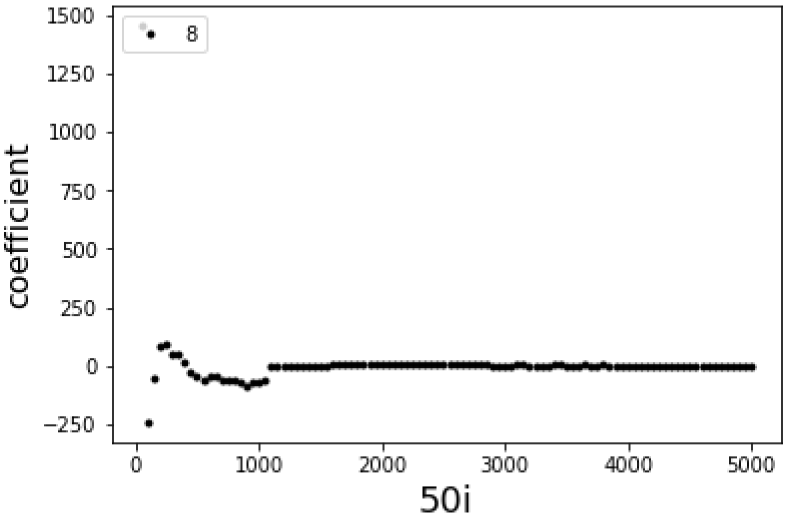

13
0.005
0.004
0.003
0.002
0.001
0.000
−0.001
coefficient
0
1000
2000
3000
4000
5000
50i

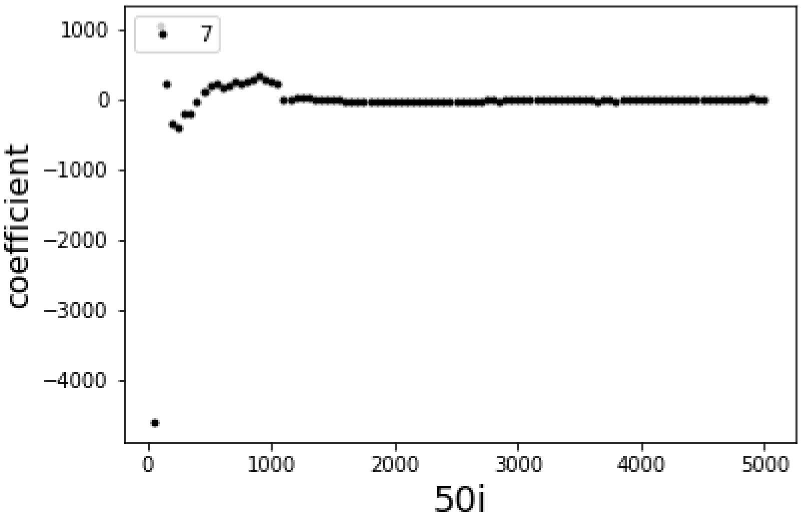

12
coefficient
0.25
0.20
0.15
0.10
0.05
0.00
0
1000
2000
3000
4000
5000
50i

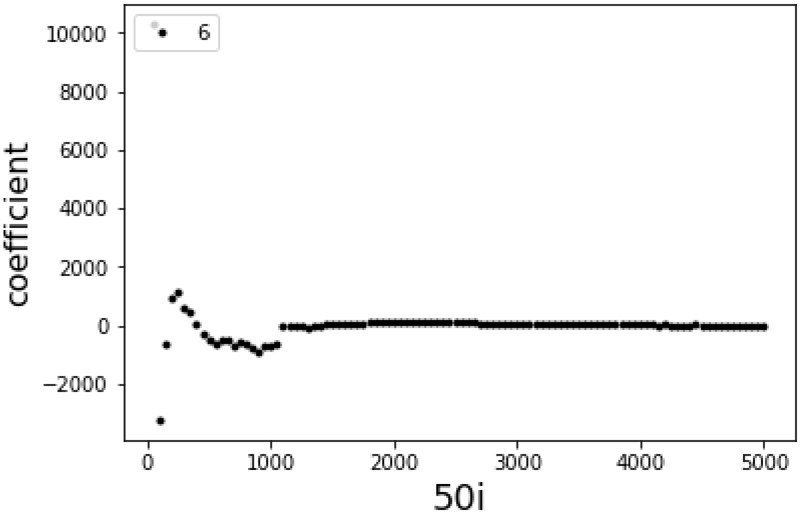

11
coefficient
0
−1
−2
−3
−4
−5
0
1000
2000
3000
4000
5000
50i

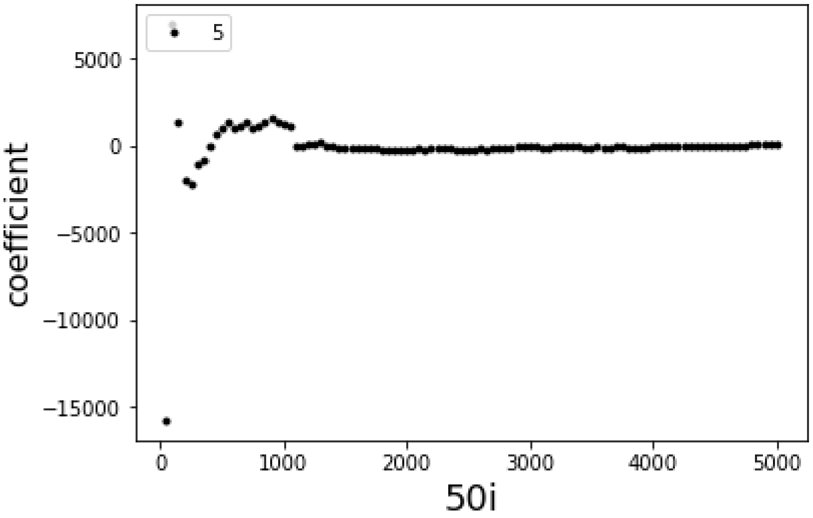

10
coefficient
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
50i

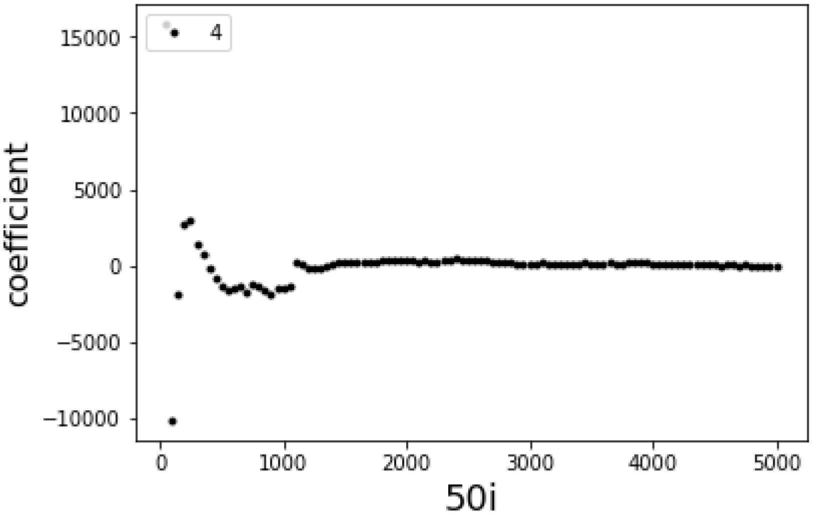

9
coefficient
50i
50
0
−50
−100
−150
−200
−250
−300
0
1000
2000
3000
4000
5000

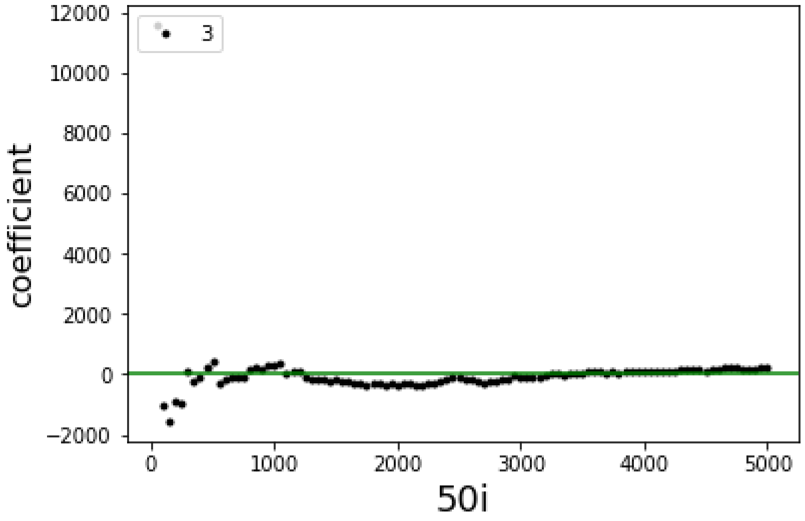

8
coefficient
50i
1500
1250
1000
750
500
250
0
−250
0
1000
2000
3000
4000
5000

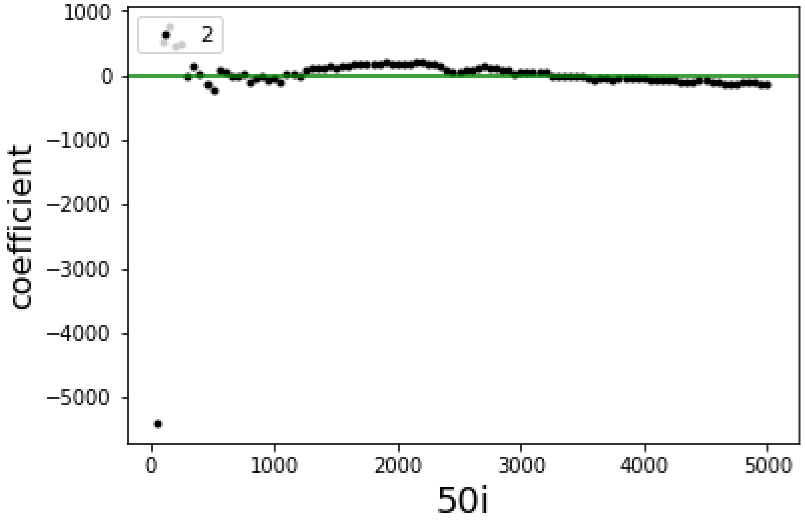

7
coefficient
50i
1000
0
−1000
−2000
−3000
−4000
0
1000
2000
3000
4000
5000

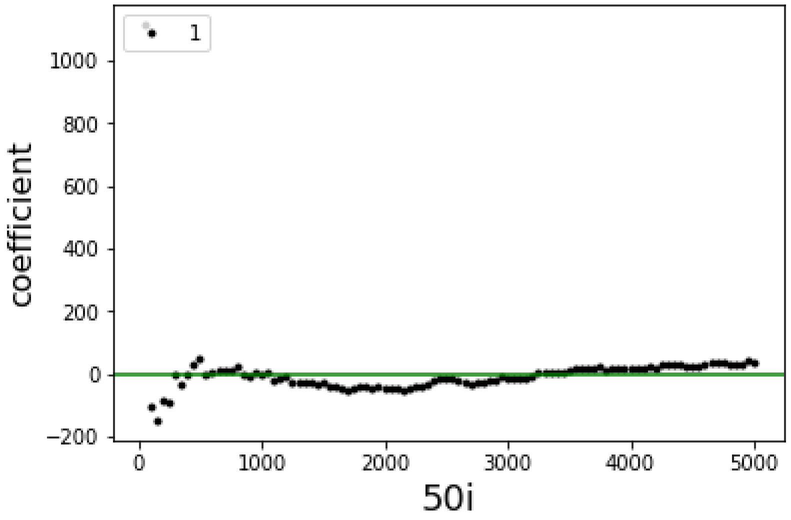

6
coefficient
10000
8000
6000
4000
2000
0
−2000
0
1000
2000
3000
4000
5000
50i

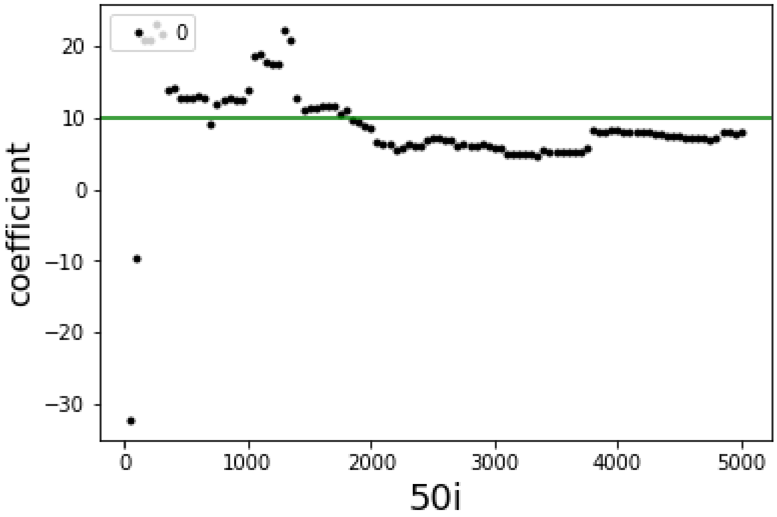

5
coefficient
5000
0
−5000
−10000
−15000
0
1000
2000
3000
4000
5000
50i


4
coefficient
15000
10000
5000
0
−5000
−10000
0
1000
2000
3000
4000
5000
50i


3
12000
10000
8000
6000
4000
2000
0
−2000
coefficient
0
1000
2000
3000
4000
5000
50i


2
1000
0
−1000
−2000
−3000
−4000
−5000
coefficient
0
1000
2000
3000
4000
5000
50i


1
1000
800
600
400
200
0
−200
coefficient
0
1000
2000
3000
4000
5000
50i



iv. In 2-5 sentences, discuss what is happening to the coefficients and comment on how the amount of over-fitting depends on the number of samples used.

The coefficients are approaching 0 as they increase 0-30. As the number of samples used increases, all coefficients approach 0(fluctuate around it). The amount of overfitting will decrease as sample size increases. A large sample size isn't as affected by outliers or noise as a small one is.

Problem 2. [40 points] For this problem, you will find a model to predict housing prices using the data set housingdata.csv on canvas. Background information is described in http://lib.stat.cmu.edu/datasets/boston

Description: There are 14 attributes for about 500 samples. The attributes are:

- 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 river; 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 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000

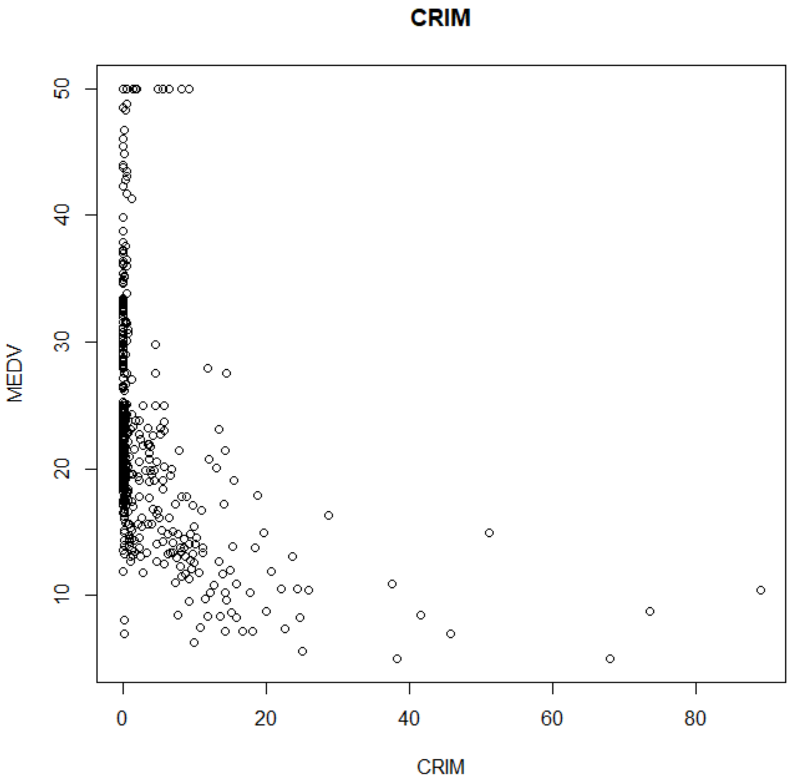
- PTRATIO - pupil-teacher ratio by town
- B - $1000(Bk - 0.63)^2$ where Bk is the proportion of black residents by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

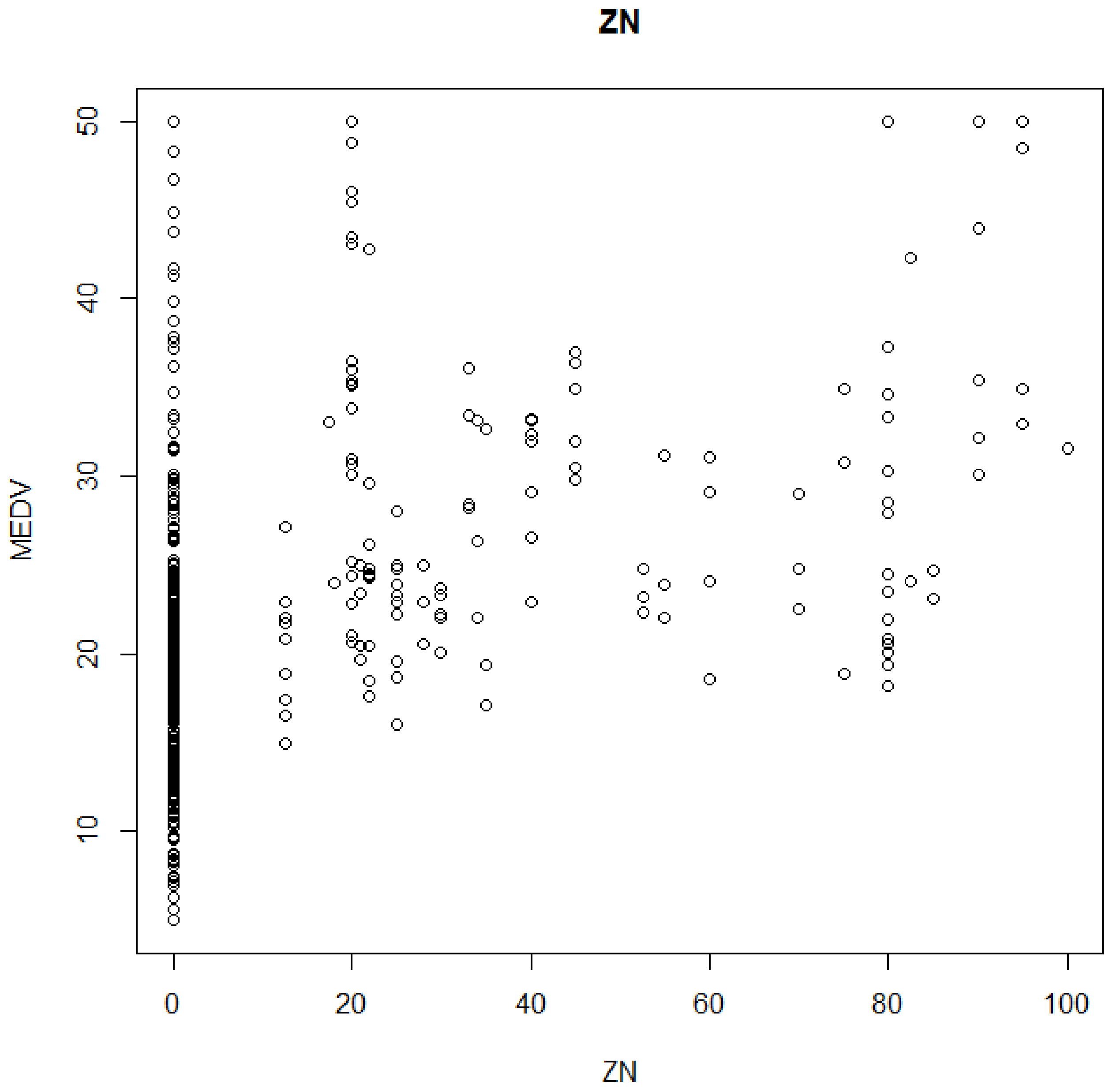
Our goal will be to use this dataset to predict median values (MEDV) for new towns or towns whose socio-economics have changed, based on the other factors. We will use MSE as the total-loss function throughout this problem.

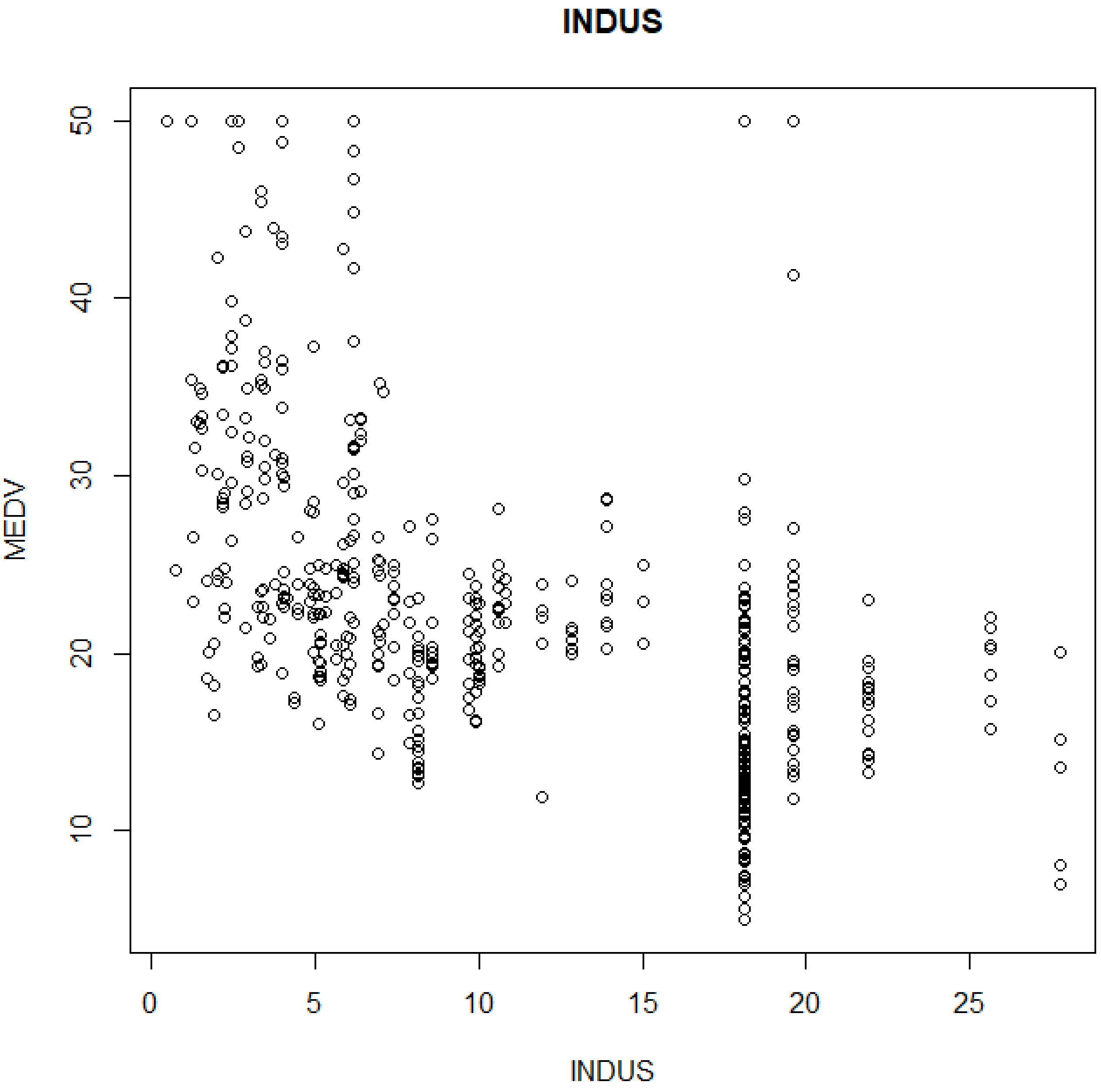
CHR

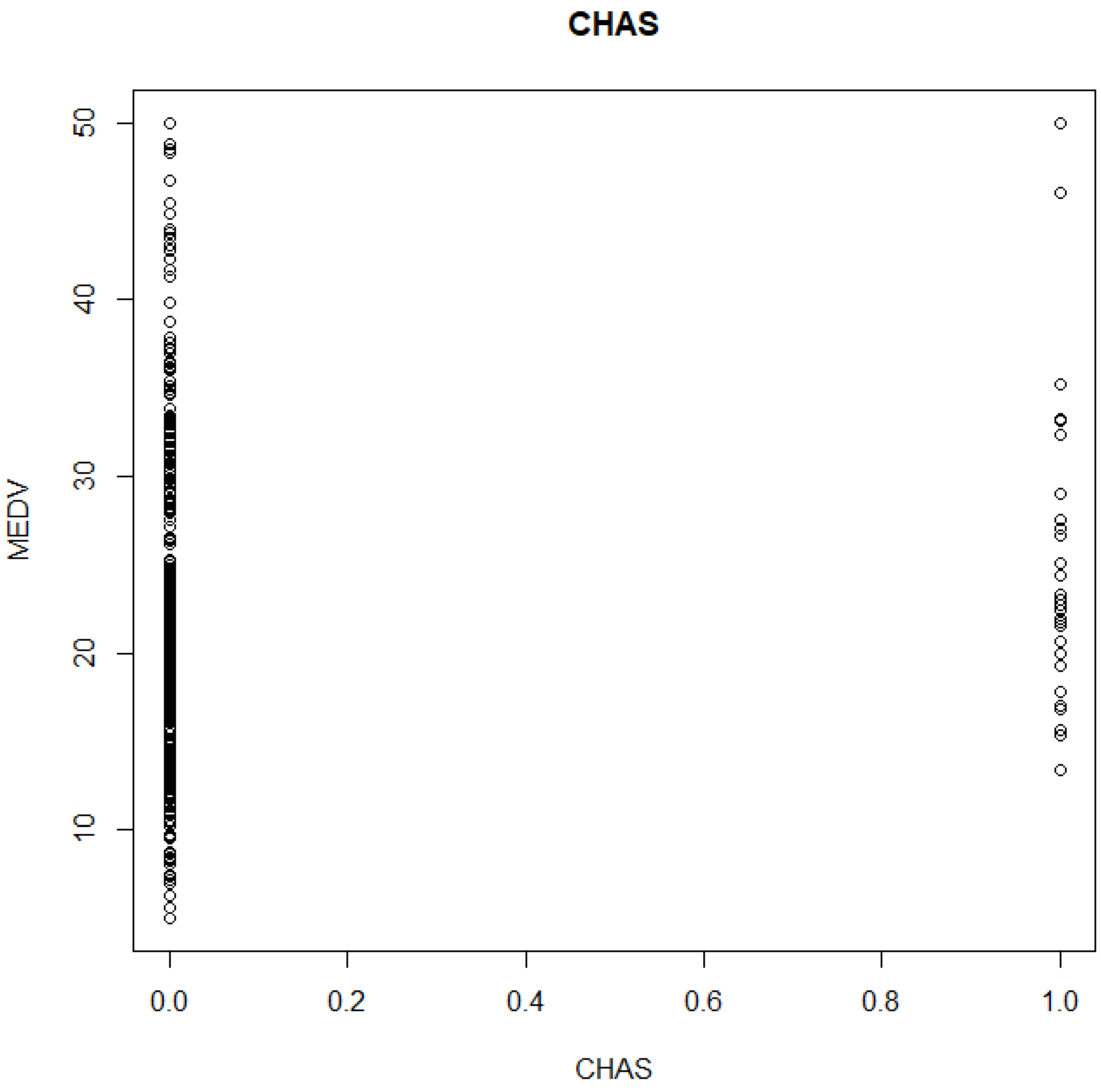
(a). First, make a scatterplot for each of the attributes with MEDV (with MEDV along the vertical axis). You should have 13 plots total.

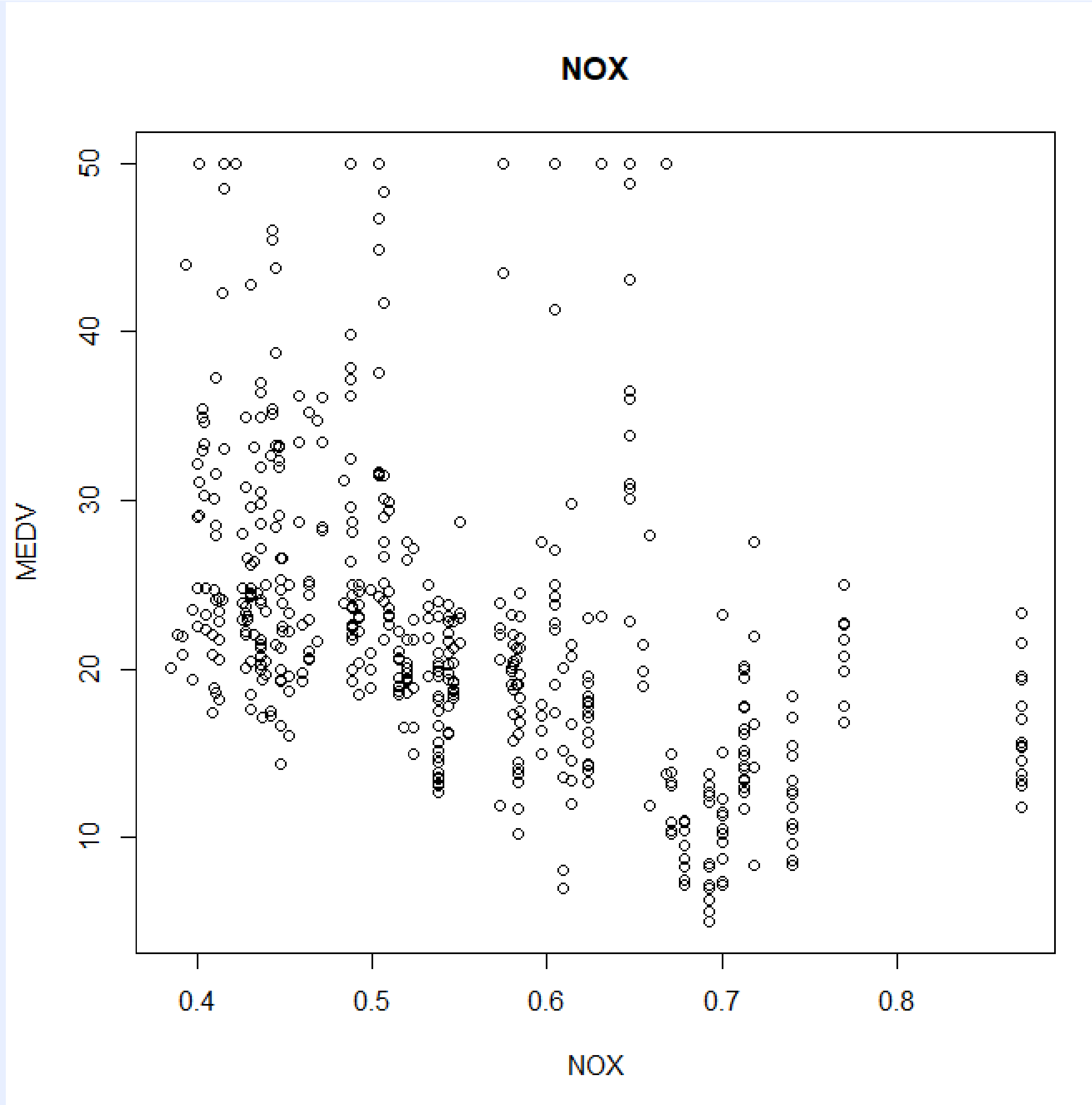
- The vertical axis should be MEDV, and the range should be the same for all 13 plots. Label the axis "MEDV."
- the title and x-axis label of each plot should be the attribute name.

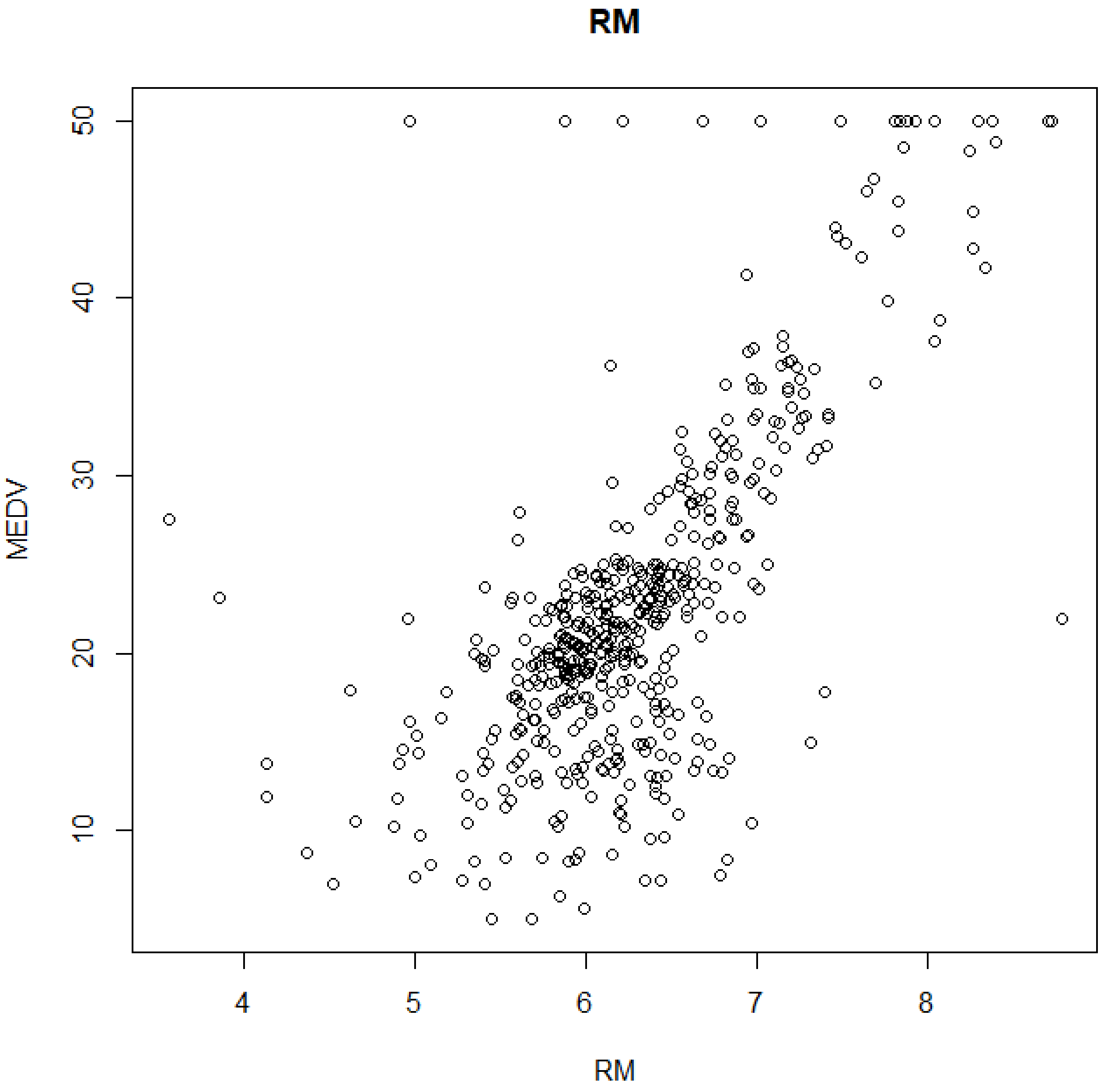


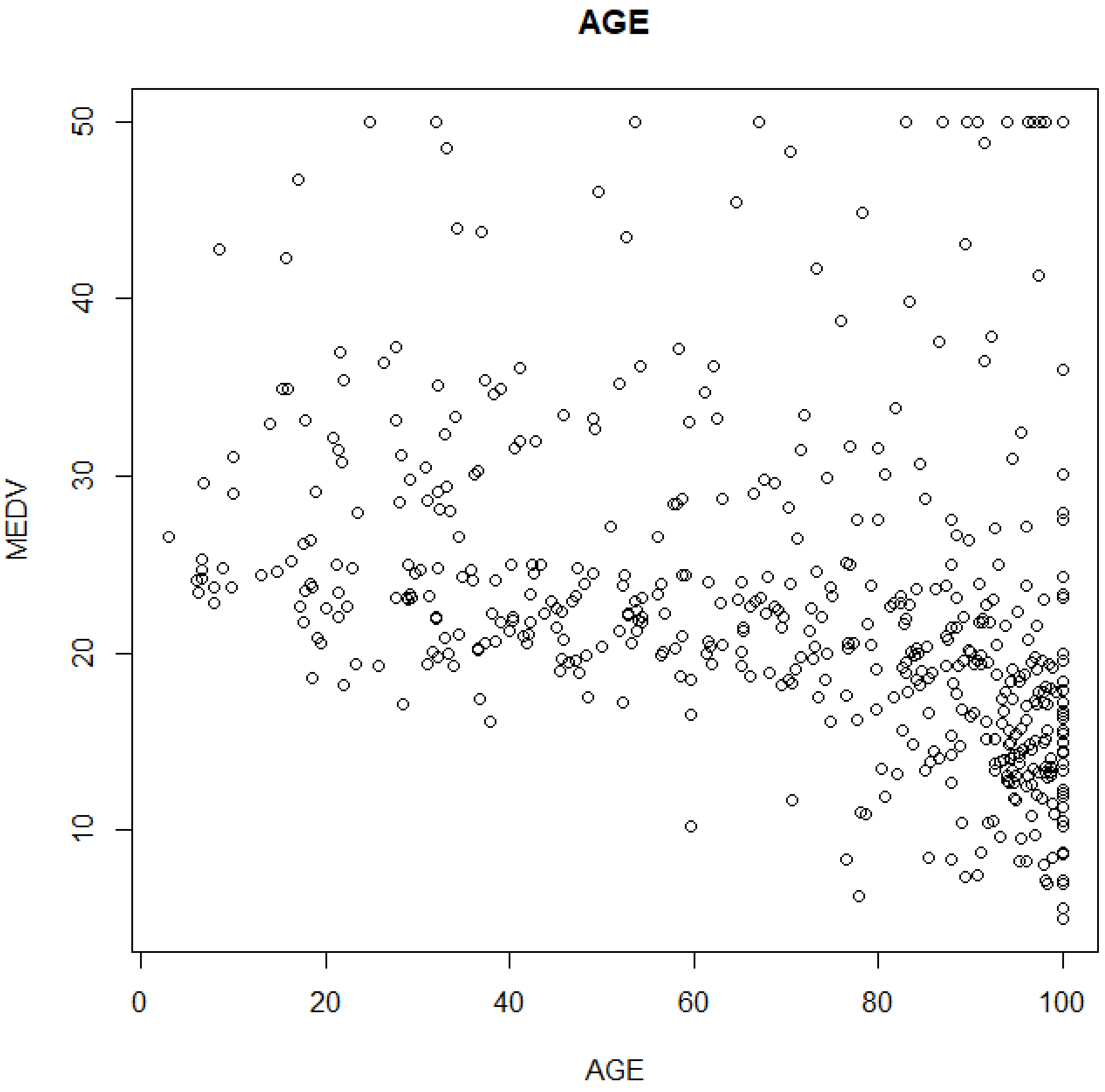

ZN
MEDV
10
20
30
40
50
0
20
40
60
80
100
ZN

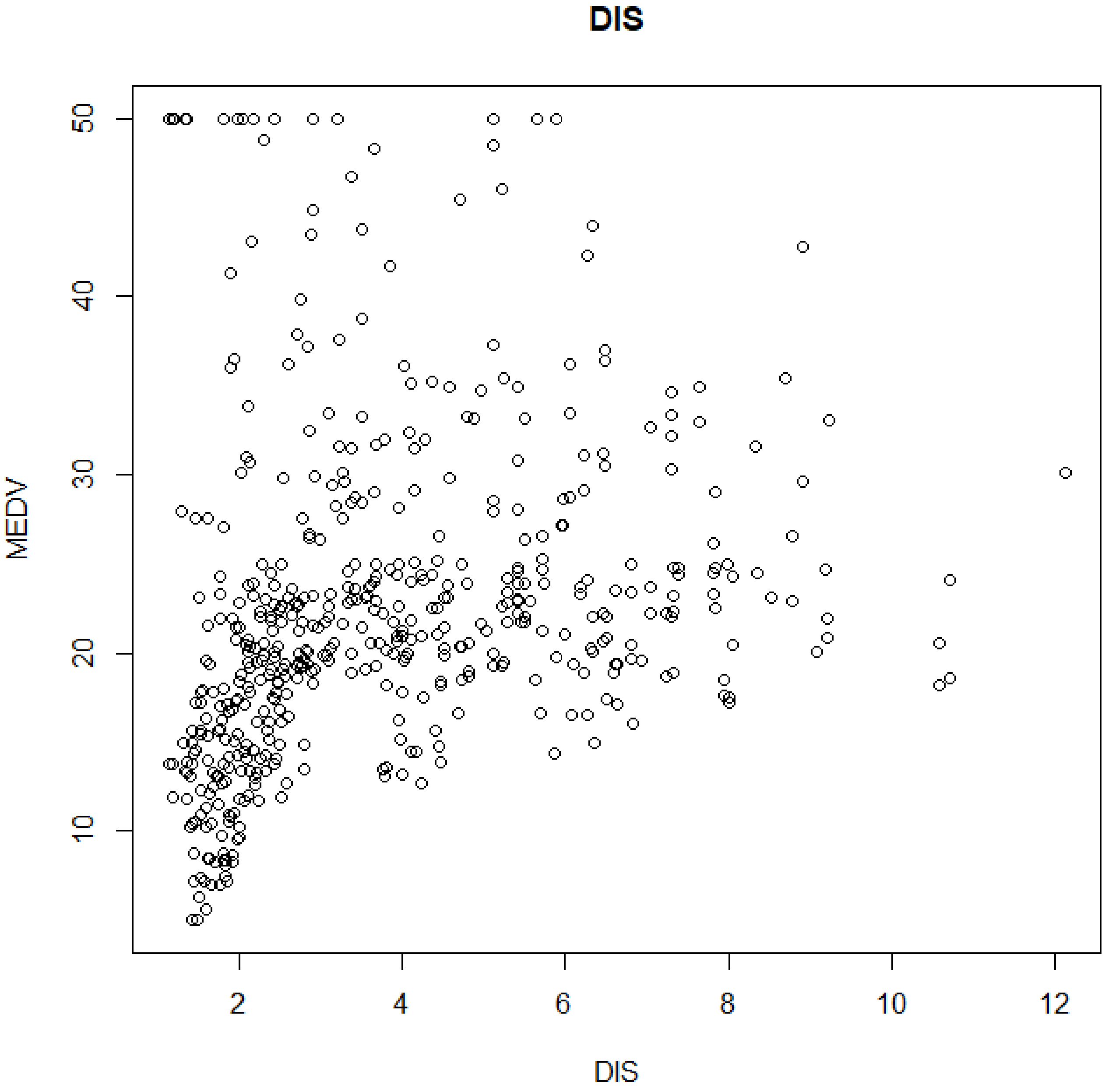

INDUS
MEDV
50
40
30
20
10
0
5
10
15
20
25
INDUS

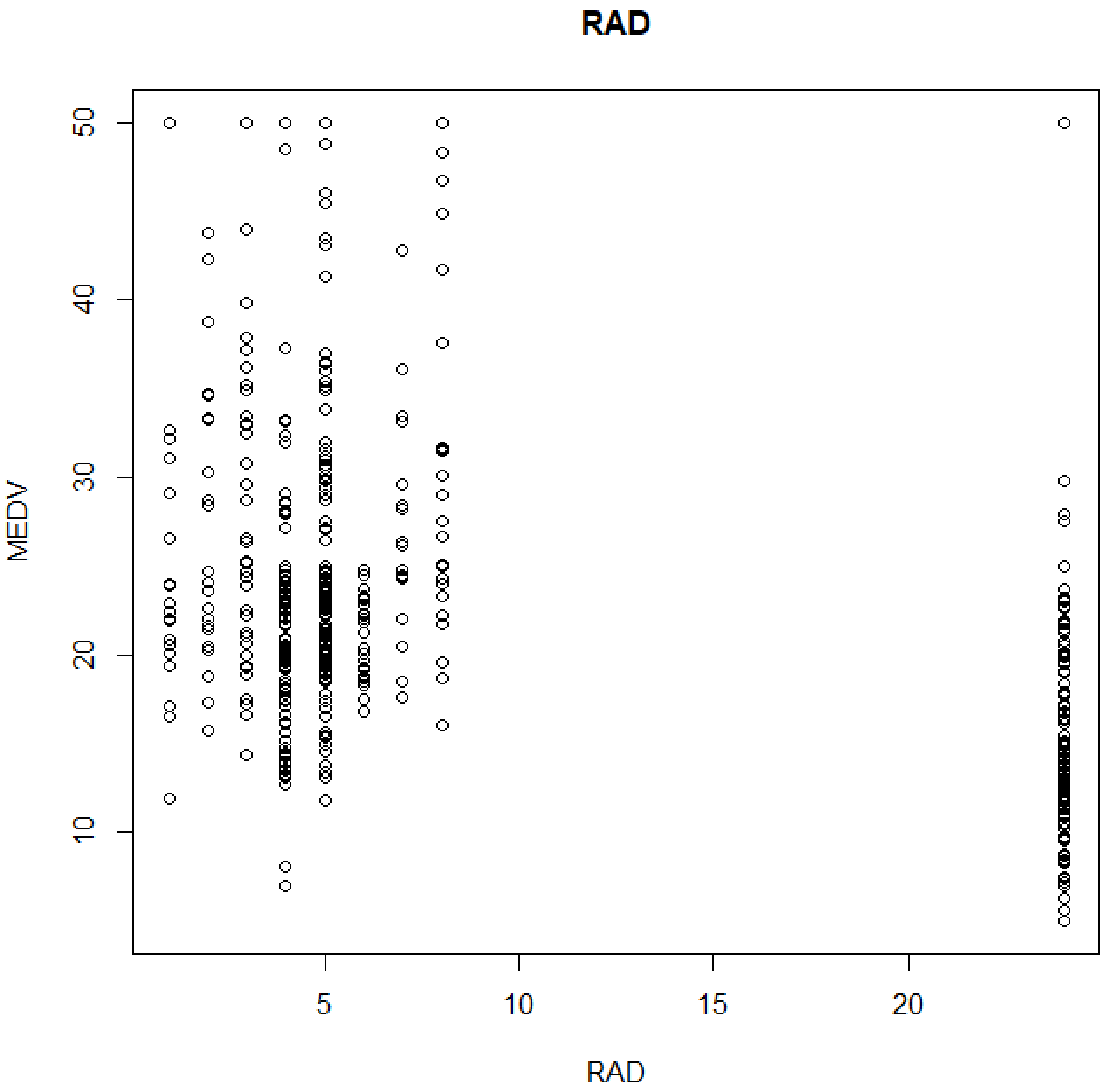

CHAS
MEDV
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0
CHAS

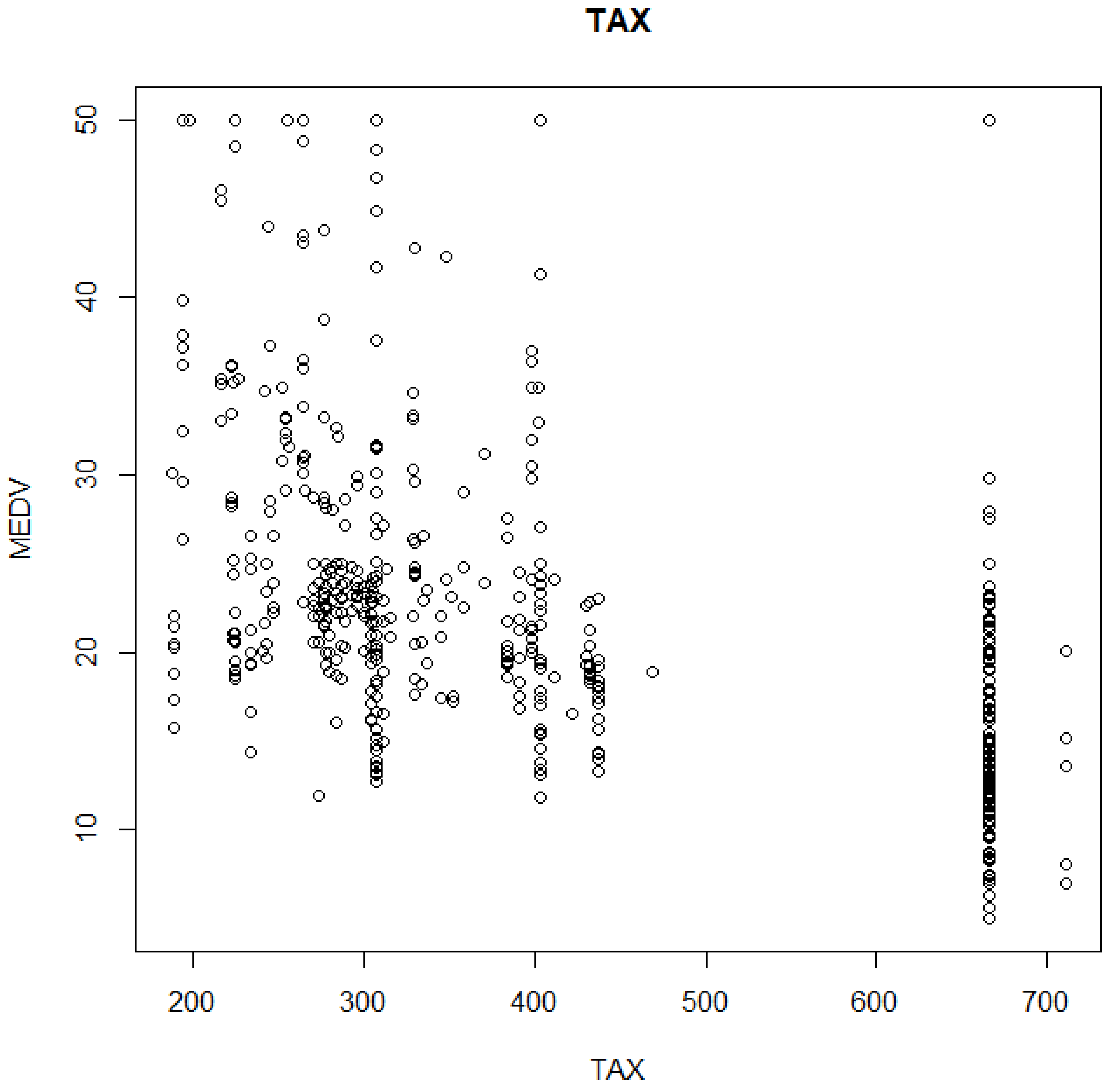

NOX
MEDV
10
20
30
40
50
0.4
0.5
0.6
0.7
0.8
NOX

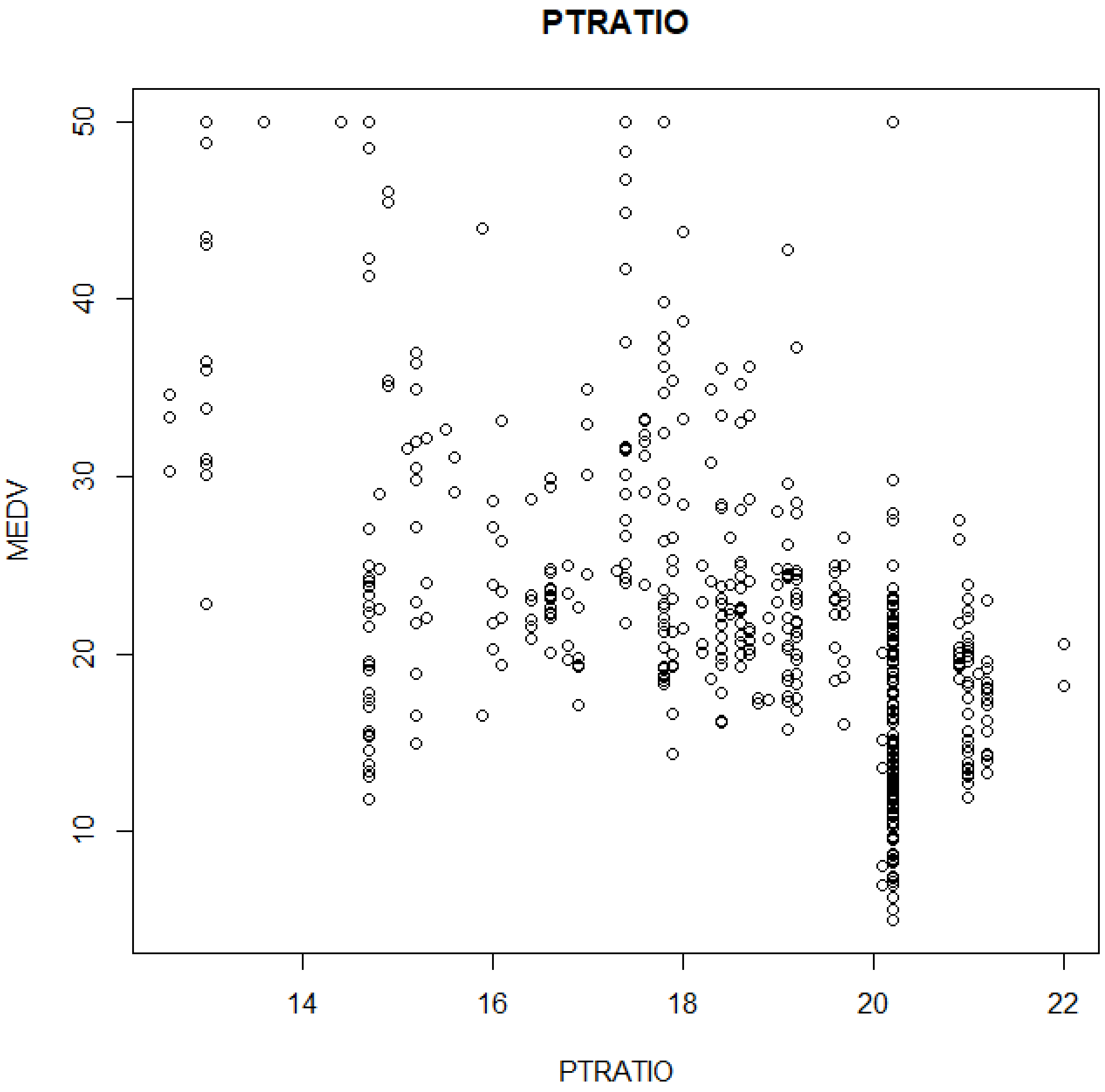

RM
MEDV
50
40
30
20
10
4
5
6
7
8
RM

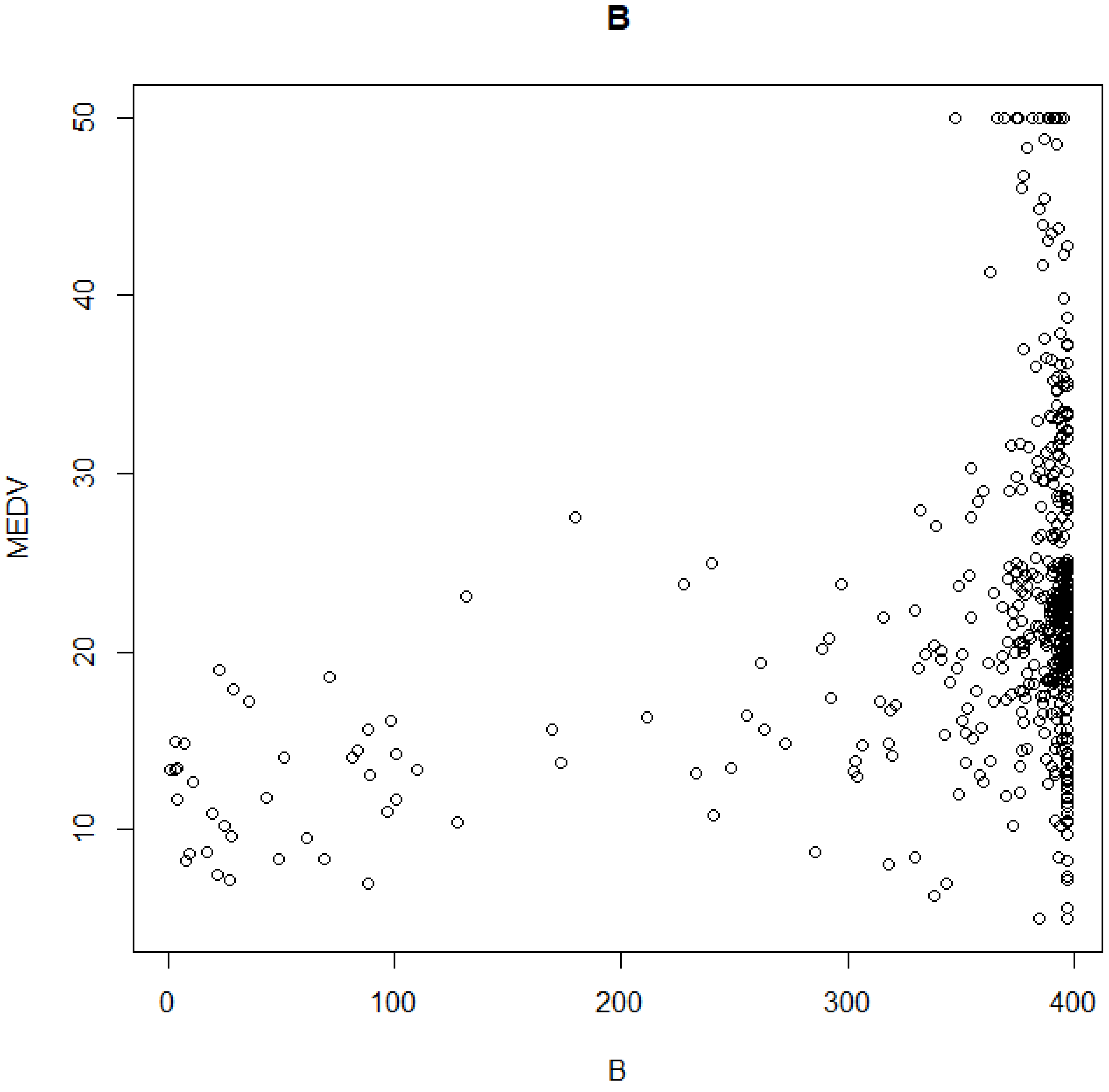

AGE
MEDV
50
40
30
20
10
0
20
40
60
80
100
AGE

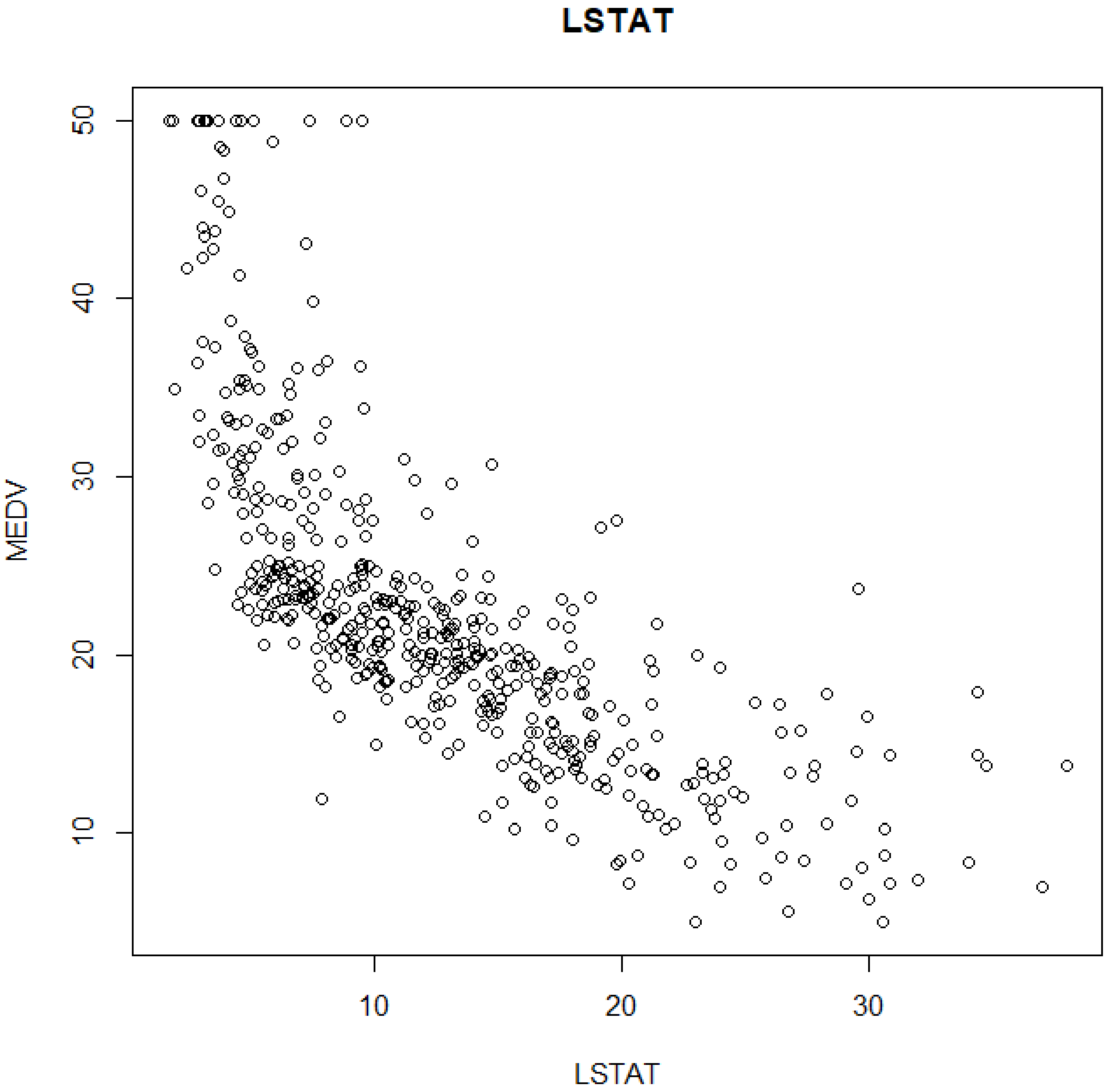

DIS
MEDV
50
40
30
20
10
2
4
6
8
10
12
DIS


RAD
MEDV
50
40
30
20
10
5
10
15
20
RAD


TAX
MEDV
10
20
30
40
50
200
300
400
500
600
700
TAX


PTRATIO
MEDV
10
20
30
40
50
14
16
18
20
22
PTRATIO


B
MEDV
10
20
30
40
50
0
100
200
300
400
B



In 3-6 sentences describe which features might be most and least relevant to include and explain why. (Though keep in mind these figures are just low-dimensional projections and sometimes there might be complex interactions between features.)

As RM increases, MEDV appears to also. As Crime, LSTAT increase, MEDV appears to decrease. If CHAS is 0, there is more of a spread in MEDV. NOX and PTRATIO do not seem correlated with MEDV, the spread is seemingly random.

(b). Find the best fitting linear model for every subset of AGE, INDUS, NOX, RM, TAX using the first n = 400 samples. Reserve the remaining samples for validation.

(c). For *i* = 0 to *i* = 5, select the model with *i* attributes from part (b). that has the lowest validation set MSE.

i. Write down what the best subset was for each *i*. Briefly comment on to what extent they are nested.

0: Mean of training MEDV

1:RM

2: AGE-RM

3: AGE-RM-TAX

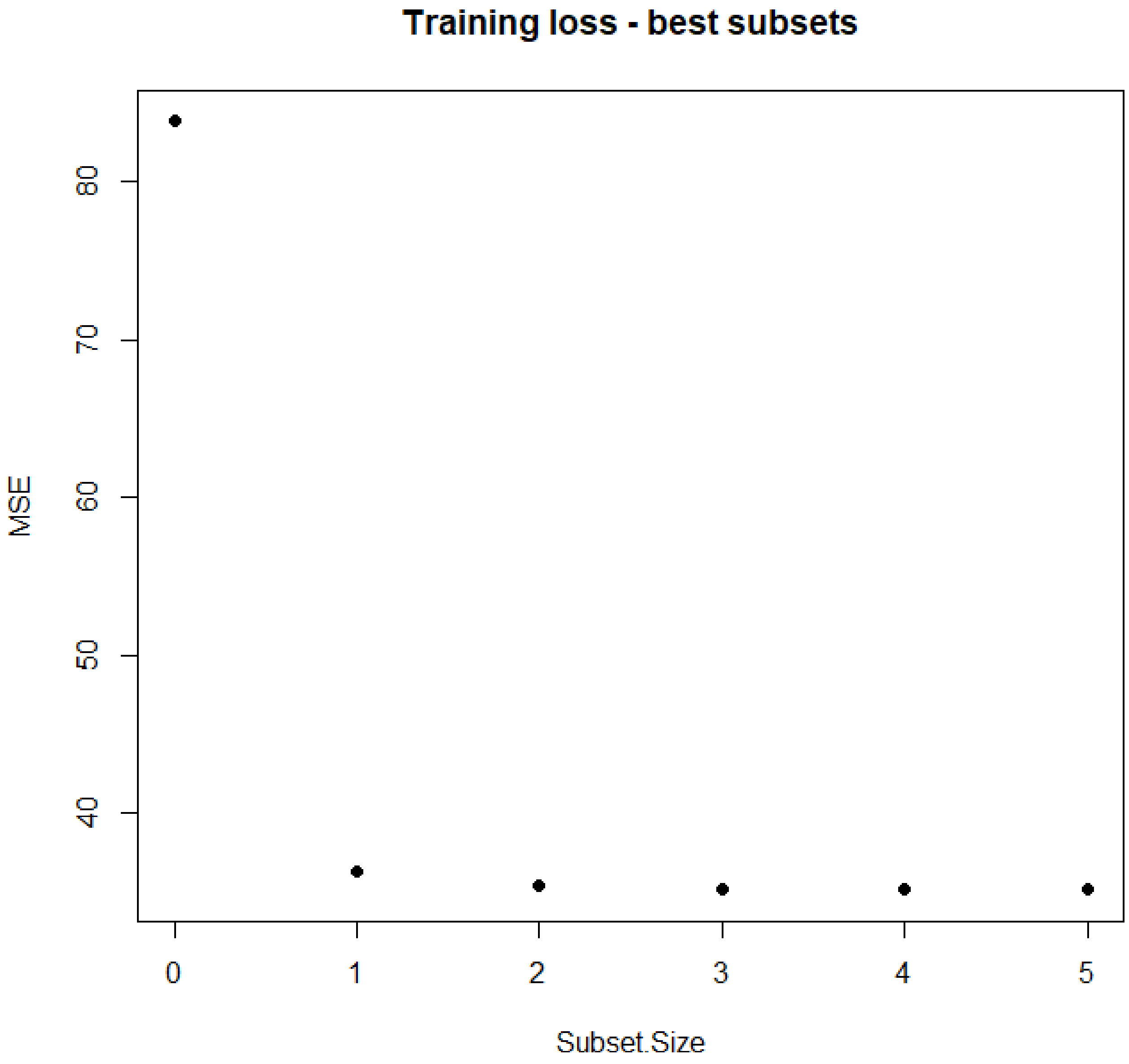
4: AGE-NOX-RM-TAX

5: AGE-INDUS-NOX-RM-TAX

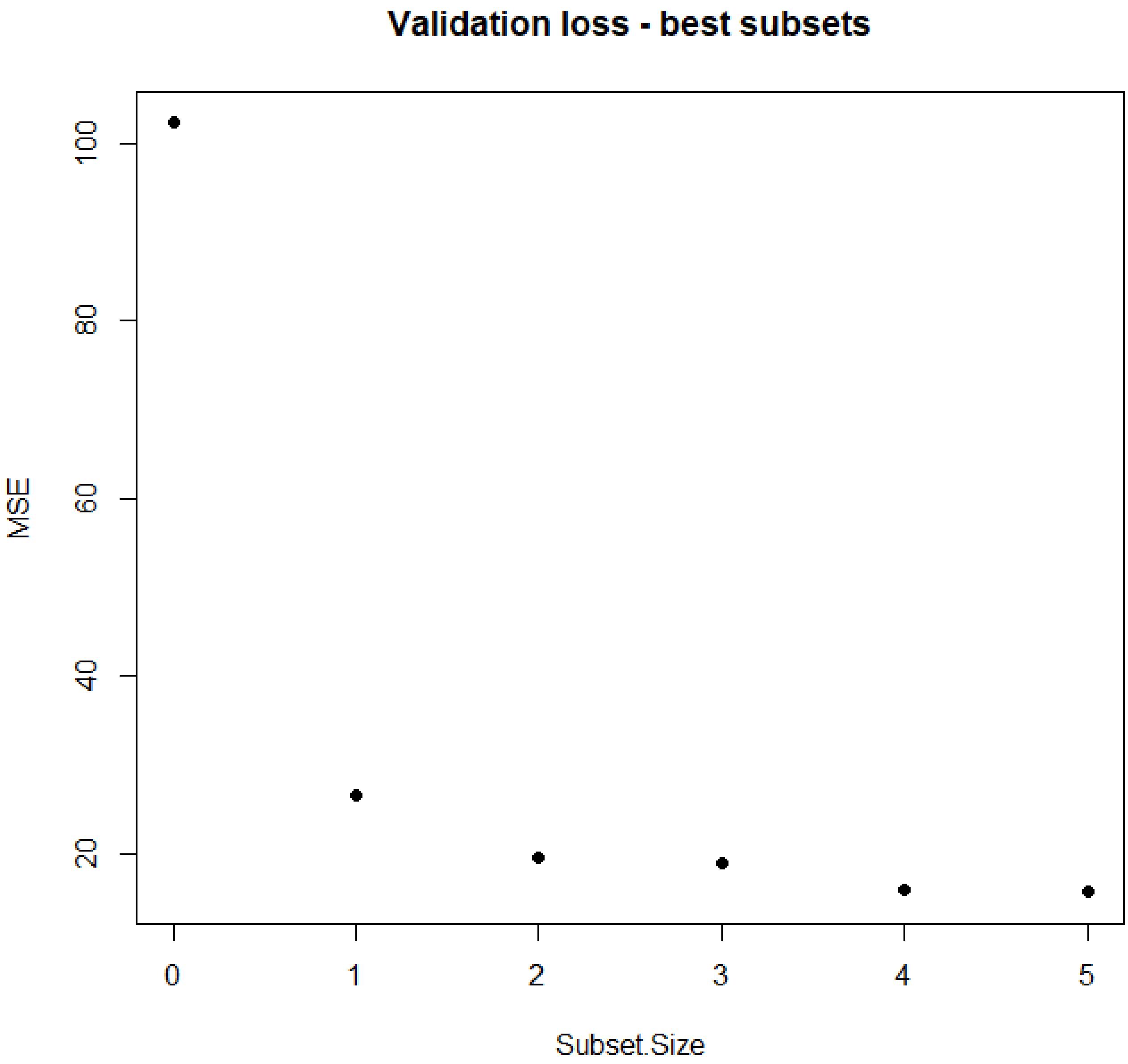
As far as nesting goes, all terms in a lower number subset are included in the higher number subset.

Make a plot of the training data total loss of the best fitting model with *i* variables

- The horizontal axis is the number of variables *i* for *i* = 0 to *i* = 5.
- Title the plot “Training loss - best subsets”



- Using the validation data set (samples not used in fitting), make a similar plot of the validation set’s MSE, titled “Validation loss - best subsets”



- In 2-4 sentences, comment on how the the two plots are similar/differ. iii.

  They both have the i=0 being the greatest MSE. They both have lower MSE as i increases. They have different MSE for each i between the two plots.

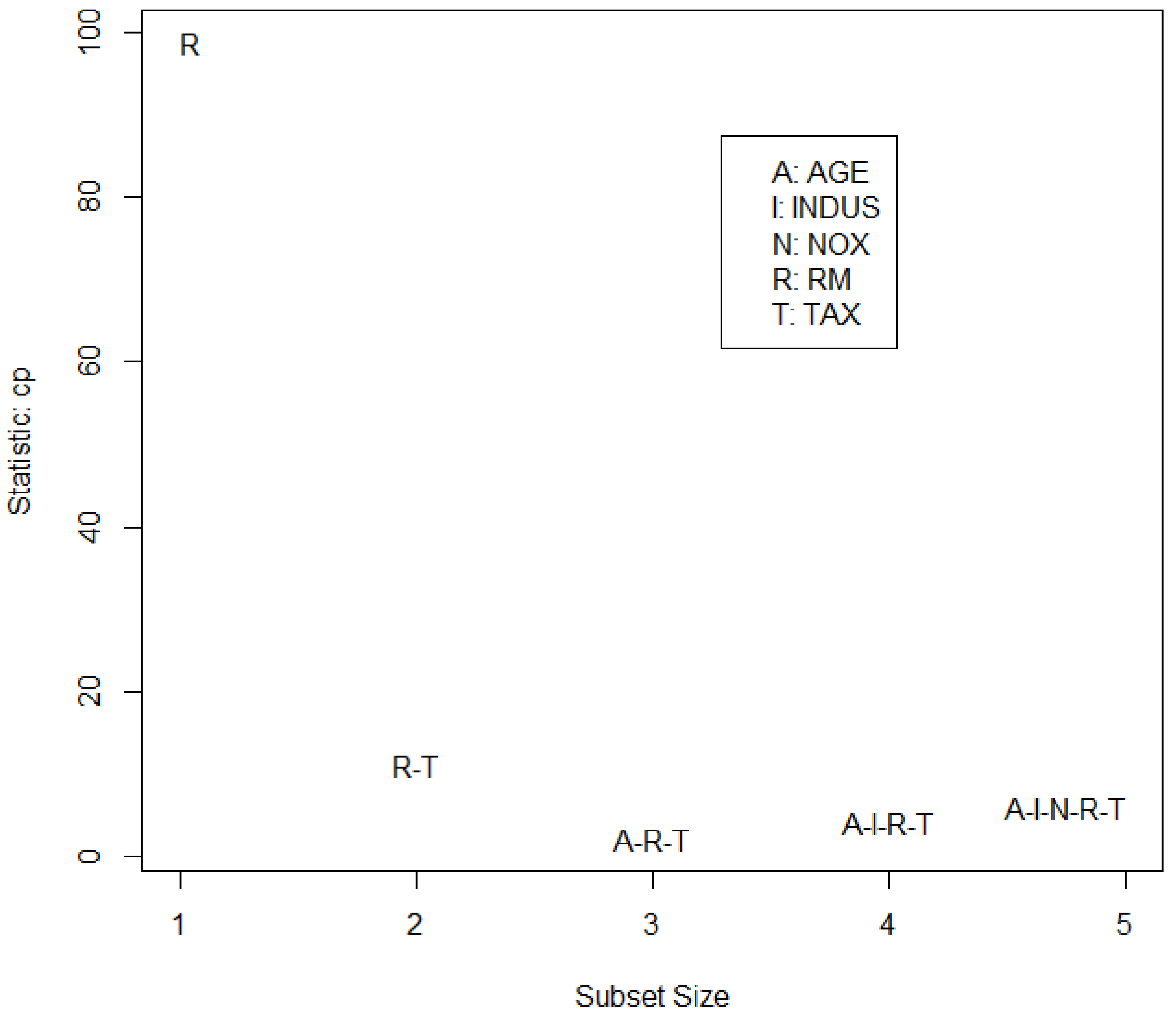
- Now find the best fitting linear model for every subset of AGE, INDUS, NOX, RM, TAX using all the samples. Instead of using a validation set, we will use a complexity penalty. We will measure total complexity using Mallow's C_p,

$$C_p = \text{MSE} + \frac{2\widehat{\sigma}^2}{n} i$$

where *i* is the $\frac{2\hat{\sigma}^2}{n}$ number of attributes used and σ^2 is the MSE using all 5 variables. b

Thus, you can viewas a carefully chosen λ.

- Make a plot of the total complexity C_p as a function of the number of variables *i*, where for each *i* you are using the best fitting model that has *i* variables. • Discuss in 2-5 sentences how the results from using total complexity are similar/different to using a validation set. This plot has the same bestsubset for each i as the other plots. This plots cp values don't always decrease as i increases like the other plots do with the MSE. This is because the cp function finds the best model for each i, not the total best of all I's.



iv. Next, let's consider alternative model complexities. Documentation for scikitlearn's implementation of ridge regression (with L2 norm on coefficients) is at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model. Ridge.html and LASSO (with L1 norm on coefficients) is at https://scikit-learn. org/stable/modules/generated/sklearn.linear_model.Lasso.html

i. Split up the data again, with the first n = 400 samples for fitting. Reserve the remaining samples for validation.

ii. For ridge regression, make a plot of the validatation set's total complexity (MSE + λ*L2 norm) as a function of λ. λ should range from 0 (no penalty on model complexity) to a value large enough that the total complexity (MSE + λ*L2 norm) is close to that of the λ = 0 model.

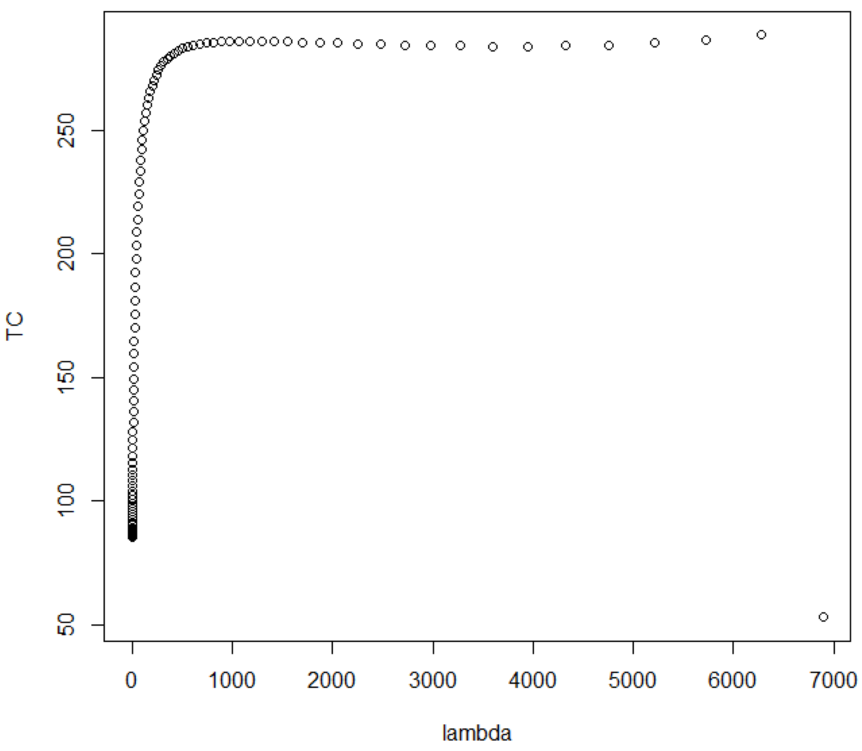
iii. Report what model minimizes that curve and discuss how it compares with the model selected using Mallows C_p and using validation error without model complexity penalties.

The model where lambda is 5.856749. This lambda has the following coefficients.

AGE	INDUS	NOX	RM	TAX
-0.017855850	-0.099340176	-2.008449786	5.213926922	-0.004427487

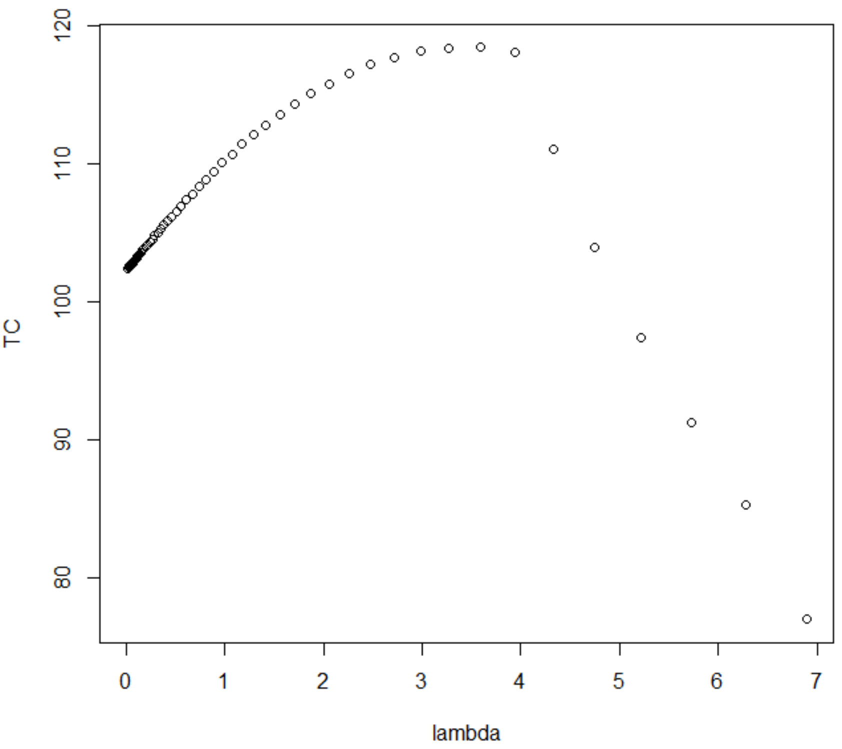
RM is the biggest term in here, and it is also the most nested and best i=0 term in the Mallows cp and validation plot. NOX is the least nested and is the last term to be added to i=5 in the validation and Mallow cp plot, and here it is the lowest value.

Ridge



iv. Repeat the previous two steps for LASSO.

LASSO



```
coef(Ridge)[-1,13]
    AGE   INDUS     NOX      RM     TAX
0.000000 0.000000 0.000000 6.325624 0.000000
```

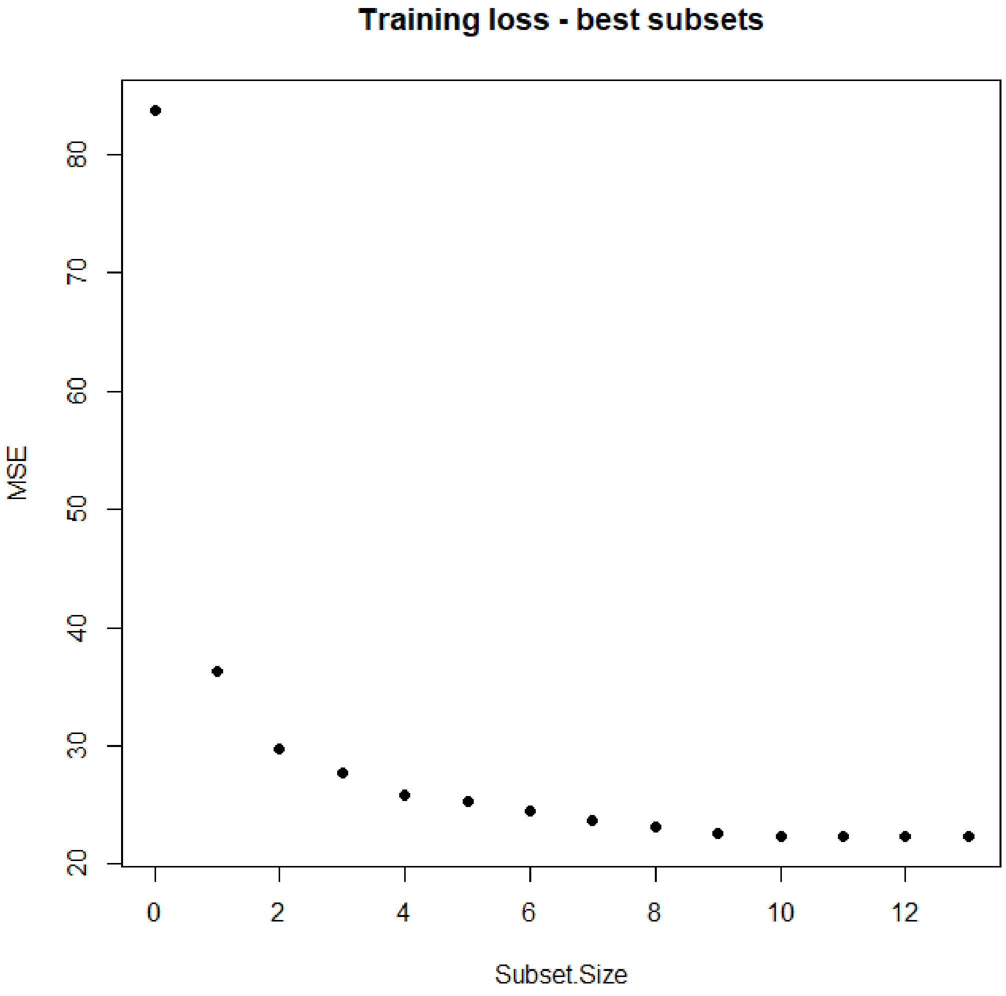
Best Lambda(lowest MSE) was 2.256915. Like in Ridge, RM is the biggest coefficient. The LASSO method reduced the others to zero.

v. In the steps above, we did not normalize the attributes to each have values in [0,1]. In 3-5 sentences, discuss whether that matters for the the feature being predicted and/or the features we are using in the prediction; would the results be different? If you were to use logistic regression on features with values like [1,0], then the results would be different. Features that are non-continuous can be better fit to the data using logistic regression. Logistic regression would make the fit better than a linear fit giving you a better predicter. This

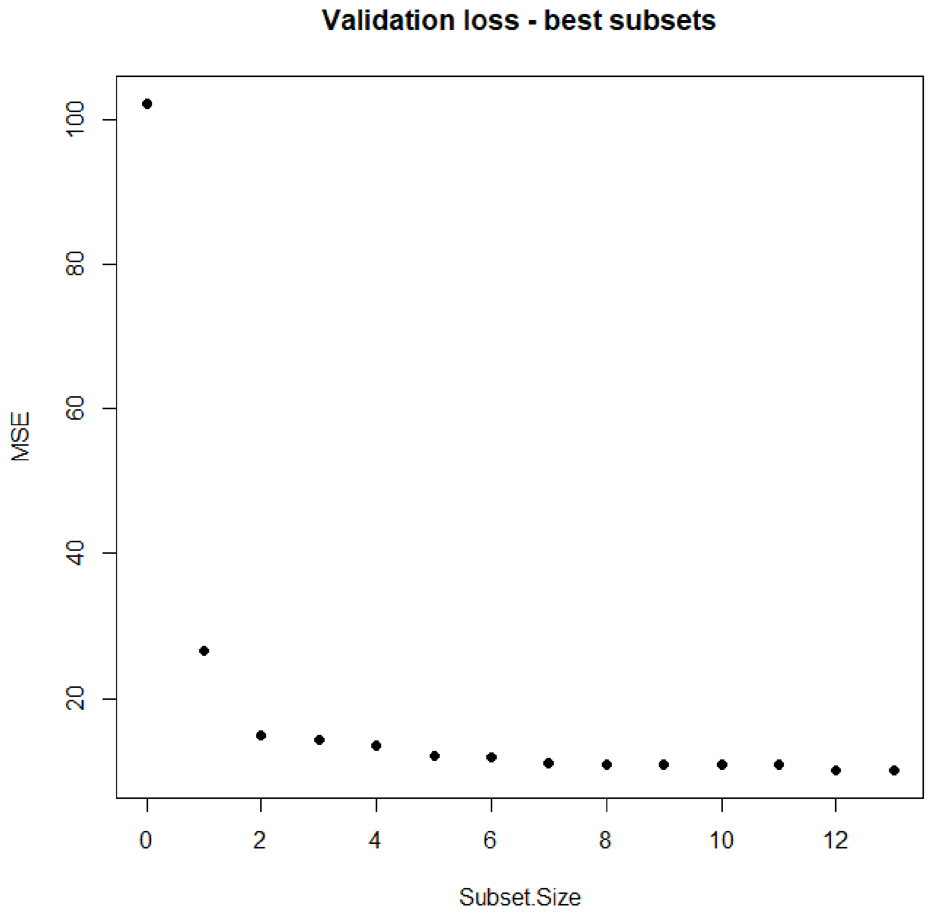
(d). The above model selections were for a subset of attributes. Now let's find a good model among all possible subsets of features. But we need to be smart about how we search, because there are 2^{13} subsets of features.

i. Perform a forward search (using validation MSE to compare) to select which models to branch off of. Use *n* = 400 samples for fitting and the rest for validation. Go from the 0th order model to the full model (using all features).

   i. Make a plot of training set MSE as a function of the number of features



   ii. Make a plot of validation set MSE as a function of the number of features



iii. Report which subset of features is best (in terms of validation MSE) across all possible subsets of features.

0: Mean of training MEDV

1: RM, data

2: RM + LSTAT

3: RM + PTRATIO + LSTAT

4: RM + LSTAT + PTRATIO + DIS

5: RM + LSTAT + PTRATIO + DIS + NOX

6: RM + LSTAT + PTRATIO + DIS + NOX + RAD

7: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM

8: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX

9: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX+ ZN

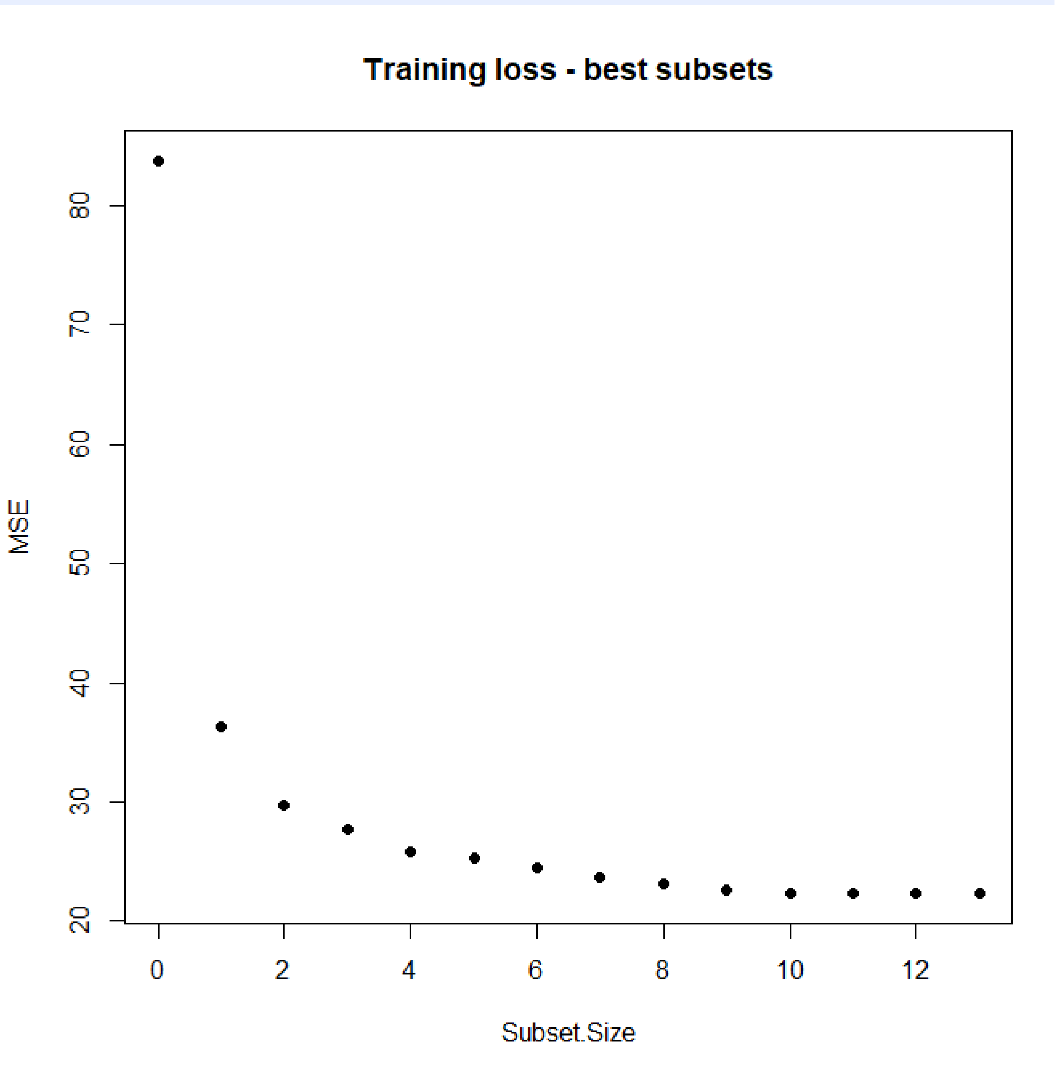
10: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX + ZN + CHAS

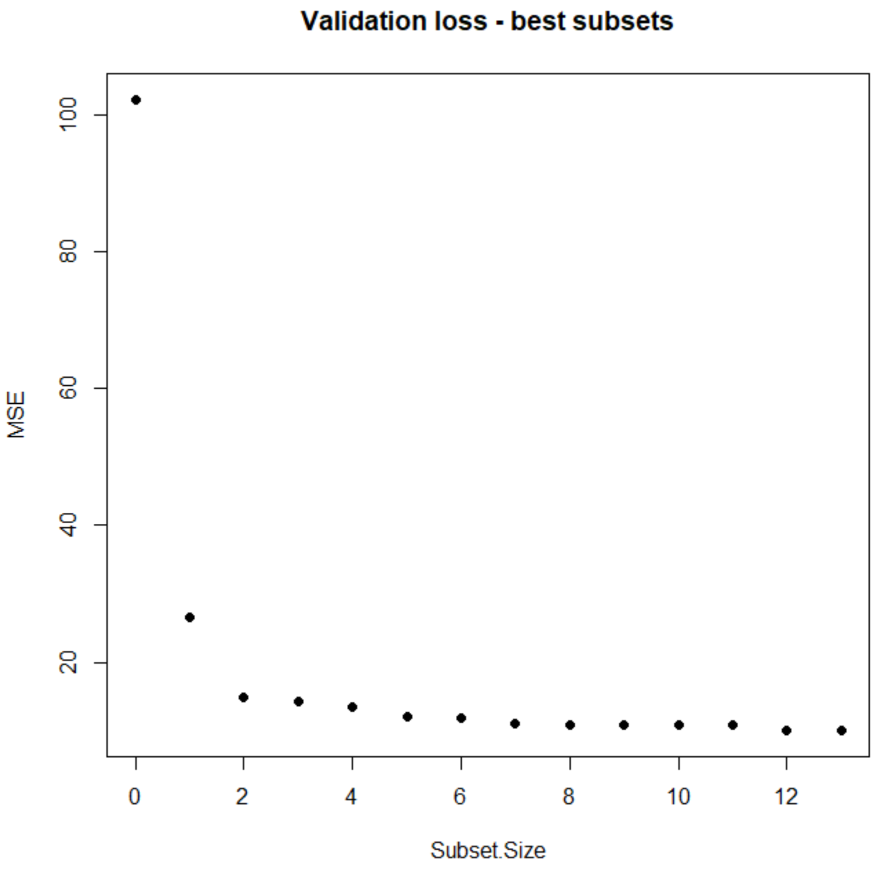
11: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX + ZN + CHAS + INDUS

12: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX + ZN + CHAS + INDUS + B

13: RM + LSTAT + PTRATIO + DIS + NOX + RAD + CRIM + TAX + ZN + CHAS + INDUS + B +AGE

ii. Repeat using a backward search. Use *n* = 400 samples for fitting and the rest for validation. Go from the the full model (using all features) to the 0th order model. Plot and report similar to the forward search.





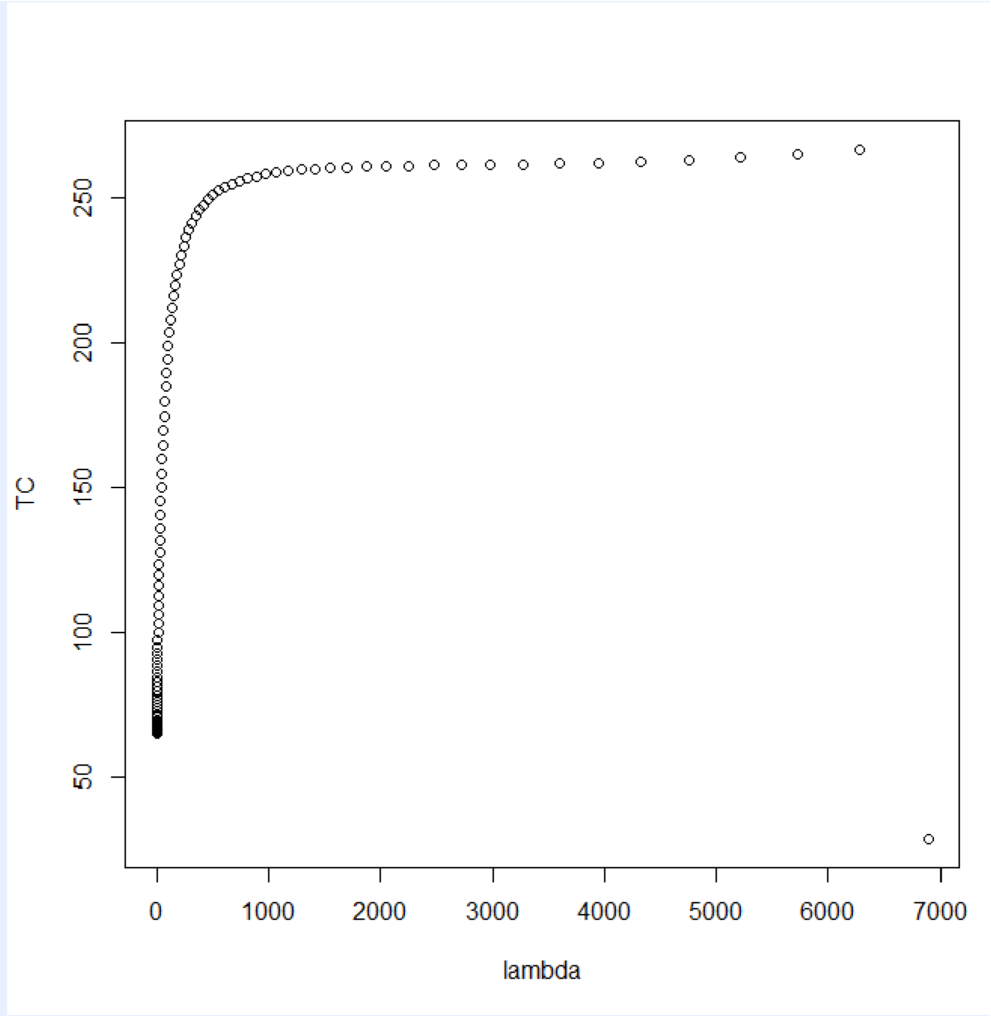
The bestsubset for each i is the same as the forward search.

iii. In 3-5 sentences, comment on similarities/differences in the nested sequence of models considered and the corresponding best models found using each method. Note that in practice, we would stop the forward/backward early unless the results kept improving. Both have the same results.

iv. Repeat the steps from part (c).iv for LASSO and Ridge regression, except use all features and in the discussion compare to the results using validation instead of Mallow's C_p.

In

RIDGE



3.678209 is the best lambda(lowest MSE).

v. CRIM ZN INDUS CHAS NOX RM

vi. -0.099519486 0.021565126 -0.053894079 2.209491298 -4.127975044 4.151776125

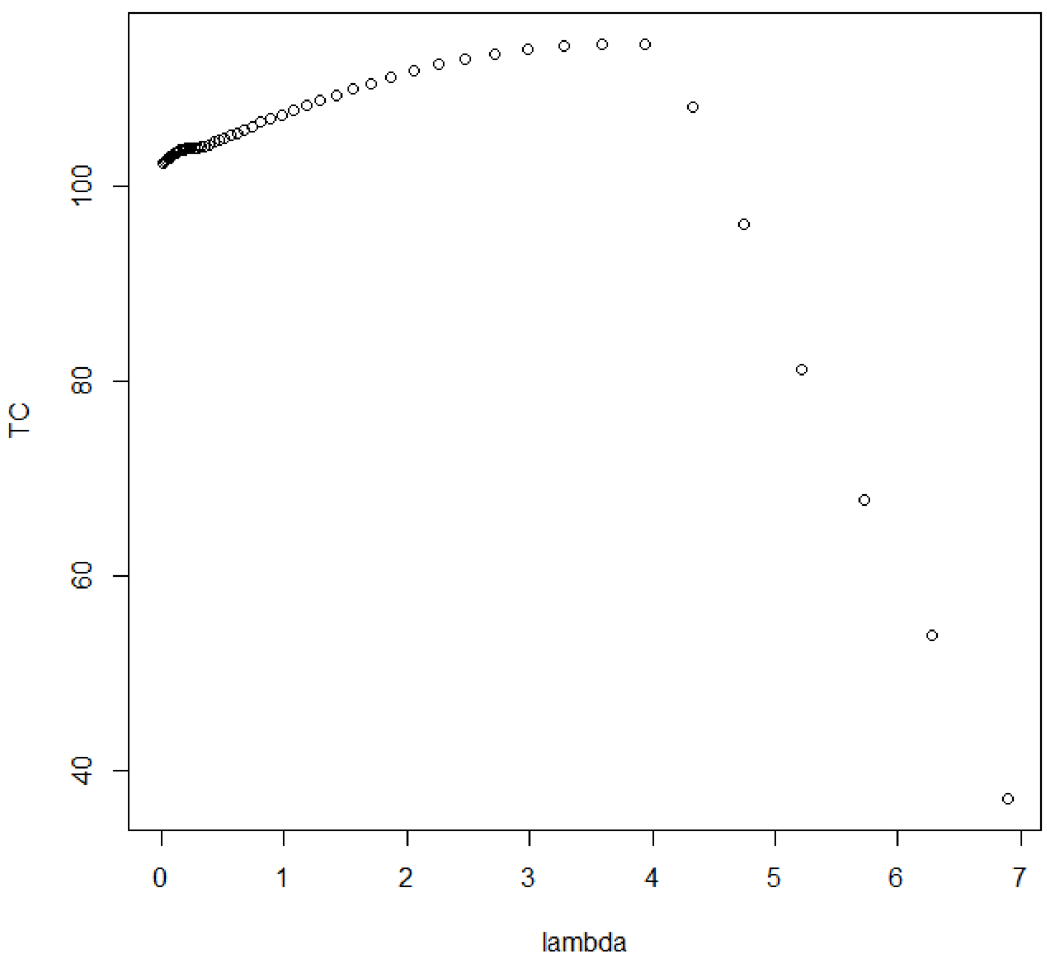
vii. AGE DIS RAD TAX PTRATIO B

viii. -0.004004613 -0.580249528 0.119174816 -0.003409452 -0.621712538 0.004904460

ix. LSTAT

x. -0.388747227

LASSO



```
      CRIM         ZN       INDUS        CHAS         NOX          RM
-0.049672676  0.007099472  0.000000000  1.351135664 -1.083946839
5.342147331
       AGE        DIS         RAD         TAX     PTRATIO           B
 0.000000000 -0.476836208  0.043992850  0.000000000 -0.586276169
0.000000000
     LSTAT
-0.514013221
```

Lambda 0.3199119 has the lowest MSE

Both Ridge and LASSO hold RM as the highest coefficient. RM in the forward and backward search was the most nested and single best i=1. The terms that follow after RM in Ridge and LASSO (in terms of being the 2nd,3rd greatest coefficient ect) are for the most part the most nested in the forward and backward search.