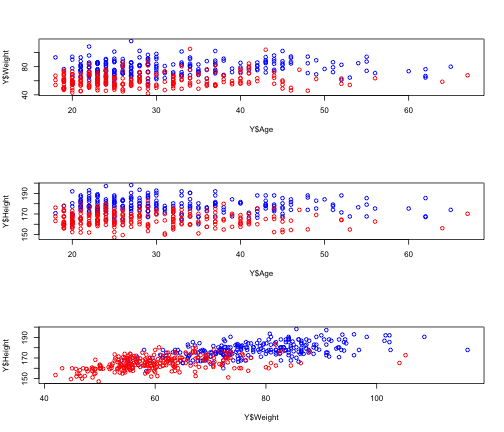
1. Read the body dataset into R using the load() function. This dataset contains: X: A dataframe containing 21 different types of measurements on the human body. Y: A dataframe that contains the age, weight (kg), height (cm), and the gender of each person in the sample. Let’s say we forgot how the gender is coded in this dataset. Using a simple visualization, explain how you can tell which gender is which.

Ans -



1. Reserve 200 observations from your dataset to act as a test set and use the remaining 307 as a training set. On the training set, use both pcr and plsr to fit models to predict a person’s weight based on the variables in X. Use the options scale= TRUE and validation=‘CV’. Why does it make sense to scale our variables in this case?

Ans – The reasons that we want to scale the variables in our case are:

Predictors of similar magnitude are easier to compare: the wrist.Diam is 10.4cm, scalling can tell us if this is a large wrist or small. 

A change of units might aid interpretability: if the further analysis use unit mm instead of cm, scalling can make sure that we can still compare.

Numerical stability is enhanced when all the predictors are on a similar scale: if the wrist.Diam is 20cm, is 30cm-wrist.girth big or small for this wrist.Diam.

The reason that we CV our variable in this case is: We are cross-validating within our pcr/plsr model fits. In other words, this is cross-validating our choice of model fit by using cross-validation on the 307 training data samples we chose. This will avoid overfitting and giving us an approximate model for the testing data set.

c) Run summary () on each of the objects calculated above, and compare the training % variance explained from the pcr output to the plsr output. Do you notice any consistent patterns (in comparing the two)? Is that pattern surprising? Explain why or why not.

Ans - No. This is not a surprising result.PLSR and PCR are both methods to model a response variable when there are a large number of predictor variables, and those predictors are highly correlated or even collinear. Both methods construct new predictor variables, known as components, as linear combinations of the original predictor variables, thus the trend of PLSR and PCR should be the same. We can see this from the % variance explained of PLSR and PCR. However, they construct those components in different ways. PCR creates components to explain the observed variability in the predictor variables, without considering the response variable at all. On the other hand, PLSR does take the response variable into account, and therefore often leads to models that are able to fit the response variable with fewer components. Thus there will be some difference between % variance explained of the PLSR and PCR.

## Data: X dimension: 307 21

## Y dimension: 307 1

## Fit method: svdpc

## Number of components considered: 21

##

## VALIDATION: RMSEP

## Cross-validated using 10 random segments.

## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps

## CV 12.81 3.090 2.996 2.810 2.759 2.764 2.771

## adjCV 12.81 3.087 2.993 2.804 2.756 2.760 2.766

## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps

## CV 2.770 2.770 2.788 2.786 2.773 2.764 2.763

## adjCV 2.764 2.765 2.784 2.779 2.767 2.757 2.757

## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps

## CV 2.763 2.732 2.702 2.658 2.628 2.626

## adjCV 2.756 2.729 2.697 2.630 2.617 2.614

## 20 comps 21 comps

## CV 2.641 2.645

## adjCV 2.628 2.632

##

## TRAINING: % variance explained

## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps

## X 62.84 74.60 79.53 84.06 86.40 88.29 89.90

## Weight 94.24 94.69 95.41 95.54 95.56 95.58 95.61

## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps

## X 91.40 92.64 93.81 94.93 95.85 96.68 97.42

## Weight 95.61 95.62 95.67 95.70 95.76 95.80 95.83

## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps

## X 97.97 98.48 98.91 99.31 99.58 99.82

## Weight 95.94 96.13 96.42 96.42 96.46 96.47

## 21 comps

## X 100.00

## Weight 96.48

**# summary for plsr**

## Data: X dimension: 307 21

## Y dimension: 307 1

## Fit method: kernelpls

## Number of components considered: 21

##

## VALIDATION: RMSEP

## Cross-validated using 10 random segments.

## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps

## CV 12.81 3.014 2.768 2.683 2.650 2.614 2.599

## adjCV 12.81 3.013 2.766 2.679 2.636 2.603 2.589

## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps

## CV 2.598 2.604 2.608 2.610 2.610 2.610 2.610

## adjCV 2.588 2.593 2.597 2.599 2.599 2.599 2.599

## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps

## CV 2.611 2.610 2.610 2.610 2.610 2.610

## adjCV 2.599 2.599 2.599 2.599 2.599 2.599

## 20 comps 21 comps

## CV 2.610 2.610

## adjCV 2.599 2.599

##

## TRAINING: % variance explained

## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps

## X 62.83 72.47 79.16 80.56 83.26 86.23 87.88

## Weight 94.54 95.53 95.89 96.33 96.42 96.46 96.47

## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps

## X 89.25 90.25 91.36 92.48 93.16 93.84 94.55

## Weight 96.47 96.47 96.48 96.48 96.48 96.48 96.48

## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps

## X 95.41 96.48 97.31 98.04 98.84 99.60

## Weight 96.48 96.48 96.48 96.48 96.48 96.48

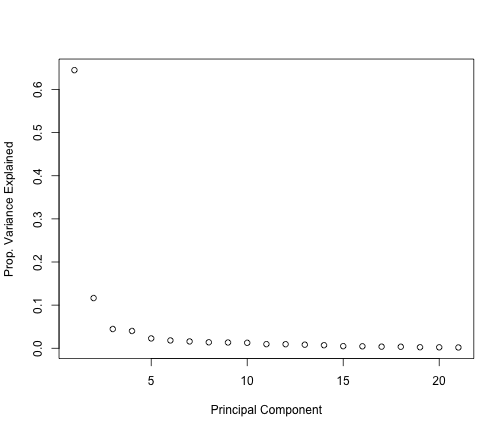
## 21 comps

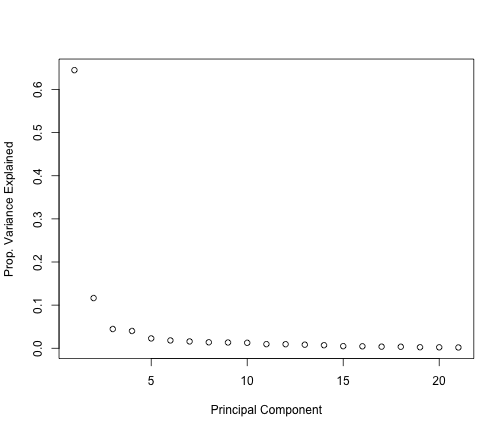
## X 100.00

## Weight 96.48

1. For each of the models, pick a number of components that you would use to predict future values of weight from X. Please include any further analysis you use to decide on the number of components.

Ans – For the Principle Component Analysis Model, I want to choose the first 5 components. For the Partial Linear Square Regression Model, we want to choose the first 8 components. This is due to the one standard deviation rule.





## Data: X dimension: 307 21

## Y dimension: 307 1

## Fit method: kernelpls

## Number of components considered: 21

##

## VALIDATION: RMSEP

## Cross-validated using 10 random segments.

## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps

## CV 13.57 3.182 2.922 2.848 2.860 2.852 2.845

## adjCV 13.57 3.180 2.920 2.843 2.849 2.840 2.832

## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps

## CV 2.849 2.842 2.837 2.837 2.839 2.839 2.839

## adjCV 2.836 2.829 2.824 2.824 2.826 2.826 2.826

## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps

## CV 2.839 2.839 2.839 2.839 2.839 2.839

## adjCV 2.826 2.826 2.826 2.826 2.826 2.826

## 20 comps 21 comps

## CV 2.839 2.839

## adjCV 2.826 2.826

##

## TRAINING: % variance explained

## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps

## X 64.47 74.93 80.24 81.88 83.85 86.47 88.31

## Weight 94.59 95.57 95.94 96.17 96.26 96.30 96.32

## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps

## X 89.65 90.69 91.68 92.95 93.61 94.58 95.26

## Weight 96.32 96.32 96.32 96.32 96.32 96.32 96.32

## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps

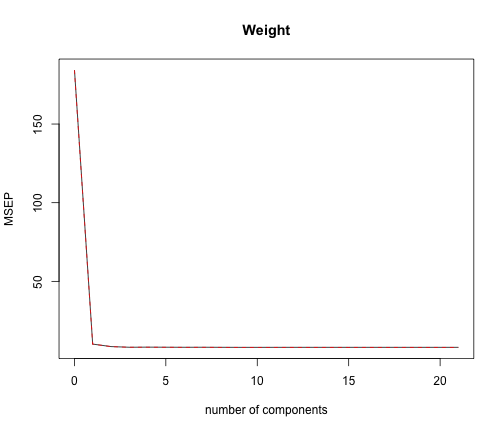
## X 95.95 97.08 97.82 98.38 98.94 99.68

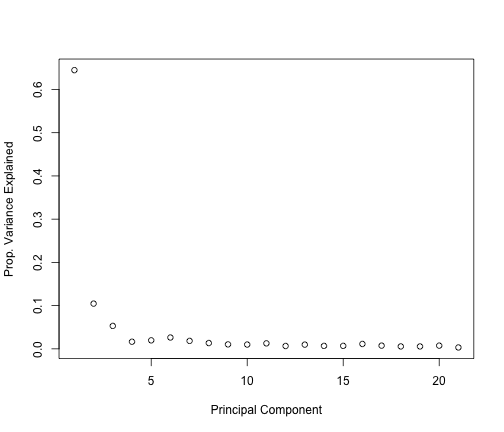
## Weight 96.32 96.32 96.32 96.32 96.32 96.32

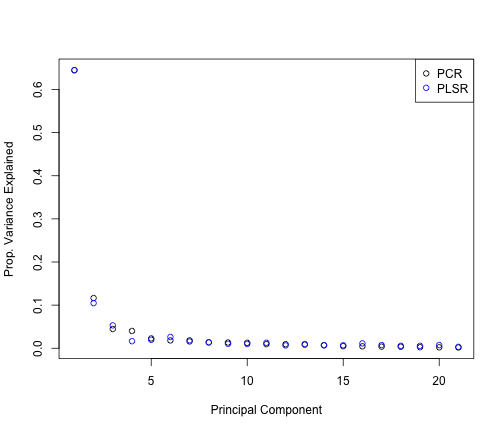
## 21 comps

## X 100.00

## Weight 96.32







e ) Practically speaking, it might be nice if we could guess a person’s weight without measuring 21 different quantities. Do either of the methods performed above allow us to do that? If not, pick another method that will and fit it on the training data

Ans – Neither of the methods performed above allow us to guess person's weight. PCA and PLR are used for summarizing the data set with a smallr number of representative variables that collectively explain most of the variability in the original set. Thus the two methods are not a good way of guessing the weight. Due to the large amount of predictors, it is hard to interpret the model and guess the weight if we include all the predictors. In order to decrease the number of predictors in the model and increase interpretability, we want to use lasso here. As we can see from the coefficients above the lasso model eliminated 4 predictors from the original 21 predictors.

1. Compare all 3 methods in terms of performance on the test set. Keep in mind that you should only run one version of each model on the test set. Any necessary selection of parameters should be done only with the training set

Ans – The best performer in terms of test MSE was the PLSR model. The lasso also had similar performance with the advantage of eliminating unnecessary variables.

## [1] 9.295

## [1] 7.672

## [1] 7.85

2.

(a) Produce a plot of test MSE (as in Figure 8.8 in the text) as a function of number

of trees for Bagging and Random Forests. You should produce one plot with two

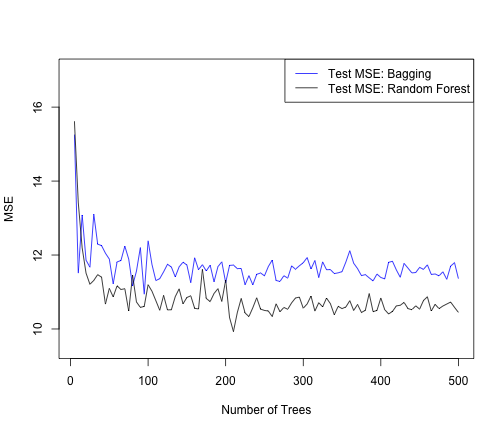
curves, one corresponding to Bagging and the other to Random Forests. Hint: If

you read the documentation for the randomForest() function, you can find a way

to obtain the data for both curves with only one call each to the randomForest()

function.

Ans -



(b) Which variables does your random forest identify as most important? How do

they compare with the most important variables as identified by Bagging.

Ans – Down here I still use the training data set as the training data set is a big part of the entire data set, and I use default number of trees, 500, this time. For RandomForest, in terms of %IncMSE, Knee.Girth, 21.622356, is identified as the most important variable in the random forest; In terms of IncNodePurity, Waist.Girth, 11968.3360, is identified as the most important variable.

## %IncMSE IncNodePurity

## Wrist.Diam 5.469 725.7

## Wrist.Girth 6.819 1614.9

## Forearm.Girth 13.936 6731.5

## Elbow.Diam 8.188 740.7

## Bicep.Girth 13.333 8018.1

## Shoulder.Girth 11.399 3378.4

## Biacromial.Diam 10.247 280.6

## Chest.Depth 7.922 780.6

## Chest.Diam 10.695 1485.1

## Chest.Girth 15.034 8396.5

## Navel.Girth 8.760 447.8

## Waist.Girth 20.812 11250.7

## Pelvic.Breadth 7.551 300.6

## Bitrochanteric.Diam 10.931 481.0

## Hip.Girth 20.654 1835.3

## Thigh.Girth 15.729 674.6

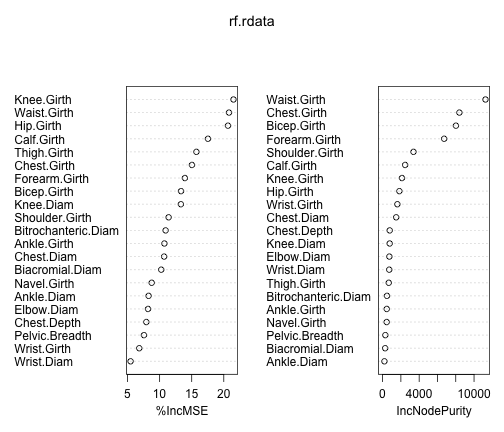
## Knee.Diam 13.310 777.8

## Knee.Girth 21.511 2121.3

## Calf.Girth 17.533 2465.8

## Ankle.Diam 8.277 199.0

## Ankle.Girth 10.741 455.3



And for Bagging, in terms of %IncMSE, Waist.Girth, 30.152524, is identified as the most important variable in the bagging; In terms of IncNodePurity, Waist.Girth, 22573.6587, is identified as the most important variable in the bagging.

## %IncMSE IncNodePurity

## Wrist.Diam 5.014 109.8

## Wrist.Girth 5.795 243.1

## Forearm.Girth 11.680 5550.1

## Elbow.Diam 8.381 214.3

## Bicep.Girth 12.542 8097.6

## Shoulder.Girth 11.301 987.1

## Biacromial.Diam 11.128 226.6

## Chest.Depth 8.748 188.8

## Chest.Diam 11.219 710.9

## Chest.Girth 14.705 7006.7

## Navel.Girth 5.201 147.9

## Waist.Girth 30.049 22041.3

## Pelvic.Breadth 5.993 142.5

## Bitrochanteric.Diam 5.577 280.7

## Hip.Girth 22.912 1518.4

## Thigh.Girth 15.685 383.9

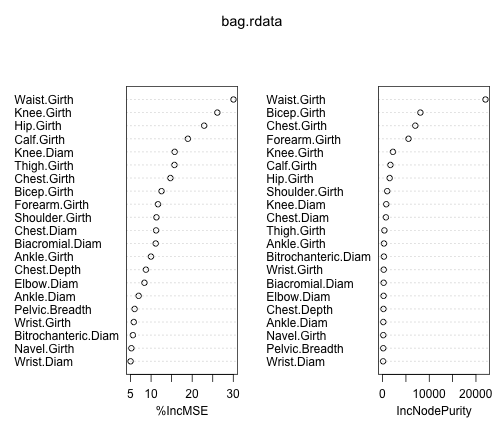
## Knee.Diam 15.738 784.2

## Knee.Girth 26.118 2219.1

## Calf.Girth 18.953 1678.5

## Ankle.Diam 6.978 154.7

## Ankle.Girth 9.972 298.2



(c) Compare the test error of your random forest (with 500 trees) against the test

errors of the three methods you evaluated in problem 1(f). Does your random

forest make better predictions than your predictions from problem 1?

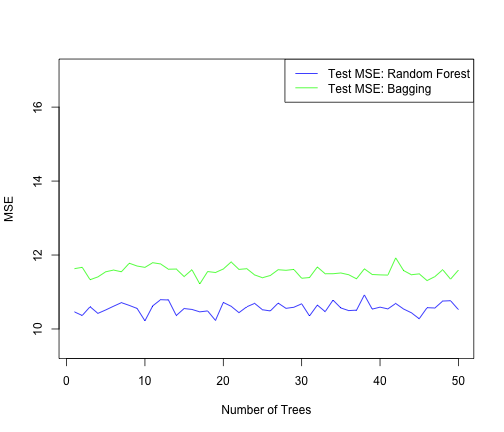
Ans - The MSE given by the randomforest with 500 trees is 10.76685. This is slightly worse than that given in the 3(f). In 3(f), pls model had a test error around 8.65, pcr had 9.27, and forward stepwise 8.63.

## [1] 10.77

(d) The randomForest() function uses 500 as the default number of trees. For this

problem, would it be valuable to include more trees? How can you tell?

Ans – As we can see that the 500 is enough tree. I ploted the MSEs of the random forest and bagging for the tree number from 501 to 500. We can see that the MSEs for the two methods are almost constant and does not have big flatuation.



3.

This problem involves the OJ data set which is part of the ISLR2 package.

(a) Create a training set containing a random sample of 800 observations, and a test

set containing the remaining observations.

Ans - Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

data('OJ')

inTrain <- sample(nrow(OJ), 800, replace = FALSE)

(b) Fit a support vector classifier to the training data using cost = 0.01, with Purchase

as the response and the other variables as predictors. Use the summary() function

to produce summary statistics, and describe the results obtained.

Ans - ##

## Call:

## svm(formula = Purchase ~ ., data = training, kernel = "linear",

## cost = 0.01)

##

##

## Parameters:

## SVM-Type: C-classification

## SVM-Kernel: linear

## cost: 0.01

## gamma: 0.05555556

##

## Number of Support Vectors: 432

##

## ( 215 217 )

##

##

## Number of Classes: 2

##

## Levels:

## CH MM

(c) What are the training and test error rates?

Ans - ## Accuracy Kappa

## 0.8337500 0.6429578

Ans bewlow we have the testing erro rates

## Accuracy Kappa

## 0.8185185 0.6184461

(d) Use the tune() function to select an optimal cost. Consider values in the range

0.01 to 10.

Ans - ## Support Vector Machines with Linear Kernel

##

## 800 samples

## 17 predictor

## 2 classes: 'CH', 'MM'

##

## Pre-processing: centered (17), scaled (17)

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 720, 719, 720, 720, 720, 720, ...

## Resampling results across tuning parameters:

##

## cost Accuracy Kappa

## 0.0100000 0.8350733 0.6461731

## 0.5357895 0.8312766 0.6399449

## 1.0615789 0.8287920 0.6345550

## 1.5873684 0.8300420 0.6370335

## 2.1131579 0.8325420 0.6426612

## 2.6389474 0.8337920 0.6449752

## 3.1647368 0.8337920 0.6454282

## 3.6905263 0.8337920 0.6454282

## 4.2163158 0.8337920 0.6454282

## 4.7421053 0.8325574 0.6424956

## 5.2678947 0.8300729 0.6361935

## 5.7936842 0.8300887 0.6368307

## 6.3194737 0.8300887 0.6368307

## 6.8452632 0.8313387 0.6395977

## 7.3710526 0.8338387 0.6450078

## 7.8968421 0.8350733 0.6474599

## 8.4226316 0.8338387 0.6450078

## 8.9484211 0.8351045 0.6479713

## 9.4742105 0.8363391 0.6504234

## 10.0000000 0.8350891 0.6478764

##

## Accuracy was used to select the optimal model using the largest value.

## The final value used for the model was cost = 9.474211.

(e) Compute the training and test error rates using this new value for cost.

Ans - ## Accuracy Kappa

## 0.8425000 0.6636592

Below is the testing error rate using the new value for cost

## Accuracy Kappa

## 0.8259259 0.6350302

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel.

Use the default value for gamma.

Ans - ##

## Call:

## svm(formula = Purchase ~ ., data = training, method = "radial",

## cost = 0.01)

##

##

## Parameters:

## SVM-Type: C-classification

## SVM-Kernel: radial

## cost: 0.01

## gamma: 0.05555556

##

## Number of Support Vectors: 617

##

## ( 306 311 )

##

##

## Number of Classes: 2

##

## Levels:

## CH MM

The training result

## Accuracy Kappa

## 0.6175 0.0000

The testing result

## Accuracy Kappa

## 0.5888889 0.0000000

## Support Vector Machines with Radial Basis Function Kernel

##

## 800 samples

## 17 predictor

## 2 classes: 'CH', 'MM'

##

## Pre-processing: centered (17), scaled (17)

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 720, 721, 720, 721, 719, 719, ...

## Resampling results across tuning parameters:

##

## C Accuracy Kappa

## 0.0100000 0.6175111 0.0000000

## 0.5357895 0.8339053 0.6408100

## 1.0615789 0.8226229 0.6151154

## 1.5873684 0.8226070 0.6155446

## 2.1131579 0.8226383 0.6156924

## 2.6389474 0.8201537 0.6096426

## 3.1647368 0.8226537 0.6150804

## 3.6905263 0.8239037 0.6191599

## 4.2163158 0.8226537 0.6162525

## 4.7421053 0.8226537 0.6162525

## 5.2678947 0.8226537 0.6162525

## 5.7936842 0.8239037 0.6192329

## 6.3194737 0.8226379 0.6168430

## 6.8452632 0.8226067 0.6169279

## 7.3710526 0.8189030 0.6090505

## 7.8968421 0.8176371 0.6066085

## 8.4226316 0.8138867 0.5983630

## 8.9484211 0.8138555 0.5978028

## 9.4742105 0.8138555 0.5978028

## 10.0000000 0.8126055 0.5947856

##

## Tuning parameter 'sigma' was held constant at a value of 0.05691

## Accuracy was used to select the optimal model using the largest value.

## The final values used for the model were sigma = 0.05691 and C = 0.5357895.

## Accuracy Kappa

## 0.8525000 0.6801995

## Accuracy Kappa

## 0.8333333 0.6495934

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial

kernel. Set degree = 2.

Ans - ##

## Call:

## svm(formula = Purchase ~ ., data = training, method = "polynomial",

## degree = 2, cost = 0.01)

##

##

## Parameters:

## SVM-Type: C-classification

## SVM-Kernel: radial

## cost: 0.01

## gamma: 0.05555556

##

## Number of Support Vectors: 617

##

## ( 306 311 )

##

##

## Number of Classes: 2

##

## Levels:

## CH MM

## Accuracy Kappa

## 0.6175 0.0000

## Accuracy Kappa

## 0.5888889 0.0000000

## Support Vector Machines with Polynomial Kernel

##

## 800 samples

## 17 predictor

## 2 classes: 'CH', 'MM'

##

## Pre-processing: centered (17), scaled (17)

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 720, 719, 720, 720, 720, 721, ...

## Resampling results across tuning parameters:

##

## C Accuracy Kappa

## 0.0100000 0.8263946 0.6210096

## 0.5357895 0.8200662 0.6149439

## 1.0615789 0.8188320 0.6111409

## 1.5873684 0.8175817 0.6087481

## 2.1131579 0.8163317 0.6063188

## 2.6389474 0.8163004 0.6074451

## 3.1647368 0.8188162 0.6126773

## 3.6905263 0.8138000 0.6019272

## 4.2163158 0.8150342 0.6049109

## 4.7421053 0.8162842 0.6073426

## 5.2678947 0.8175342 0.6097928

## 5.7936842 0.8162842 0.6070086

## 6.3194737 0.8162688 0.6064396

## 6.8452632 0.8162688 0.6064396

## 7.3710526 0.8150029 0.6035346

## 7.8968421 0.8162688 0.6064805

## 8.4226316 0.8162688 0.6064805

## 8.9484211 0.8162688 0.6064805

## 9.4742105 0.8162688 0.6064805

## 10.0000000 0.8175188 0.6089411

##

## Tuning parameter 'degree' was held constant at a value of 2

##

## Tuning parameter 'scale' was held constant at a value of TRUE

## Accuracy was used to select the optimal model using the largest value.

## The final values used for the model were degree = 2, scale = TRUE and C

## = 0.01.

## Accuracy Kappa

## 0.8525000 0.6810121

## Accuracy Kappa

## 0.8222222 0.6246742

(h) Overall, which approach seems to give the best results on this data?

Ans - Overall I would saying looking at the models…they are very similar, but the radial kernel does best by a small margin.

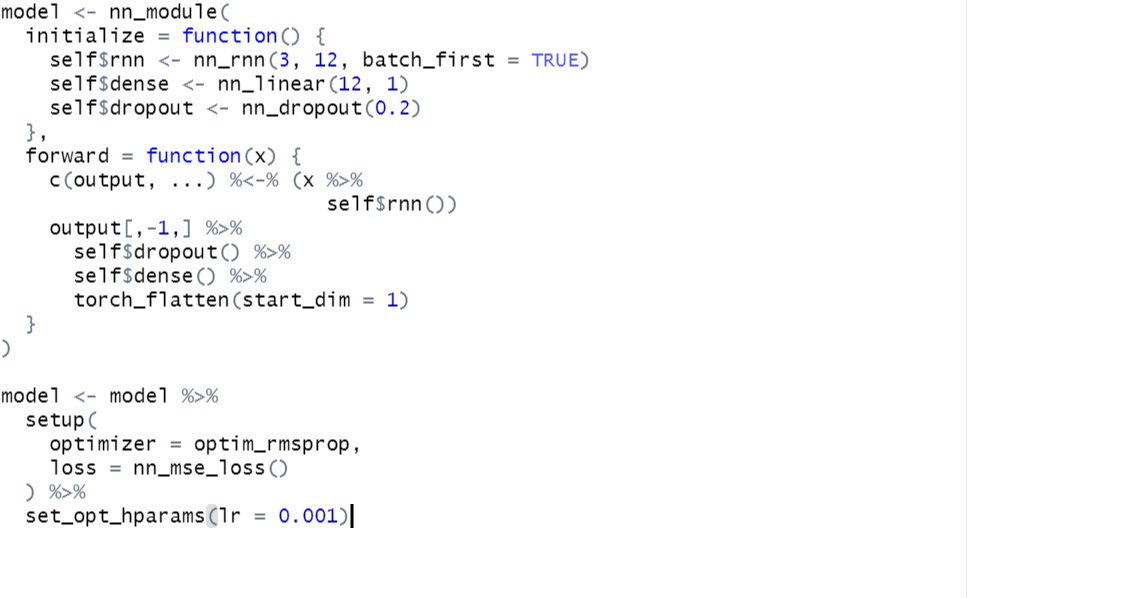
4.

Consider the RNN fit to the NYSE data in Section 10.9.6 (This code is given in the attached R

markdown file of Chapter 10 R Lab). Modify the code to allow inclusion of the variable

day\_of\_week, and fit the RNN. Compute the test set R2

Ans - We have done this in four steps. The first simply extracts the n × 15 matrix of lagged versions of the three predictor variables from arframe. The second converts this matrix to a n × 3 × 5 array. We can do this by simply changing the dimension attribute, since the new array is filled column wise. The third step reverses the order of lagged variables, so that index 1 is furthest back in time, and index 5 closest. The final step rearranges the coordinates of the array (like a partial transpose) into the format that the RNN module in keras expects. Now we are ready to proceed with the RNN, which uses 12 hidden units.

Ans - 

The output layer has a single unit for the response.

We fit the model in a similar fashion to previous networks. We supply the fit function with test data as validation data, so that when we monitor its progress and plot the history function we can see the progress on the test data. Of course we should not use this as a basis for early stopping, since then the test performance would be biased.

A picture containing text, screenshot, font, algebra

Description automatically generated

## [1] 0.4085004

We could replace the nn\_module command above with the following command:

A picture containing text, screenshot, font, white

Description automatically generated

The -1 in the formula avoids the creation of a column of ones for the intercept. The variable day\\_of\\_week is a five-level factor (there are five trading days), and the -1 results in five rather than four dummy variables. The rest of the steps to fit a nonlinear AR model should by now be familiar.

A screenshot of a computer program

Description automatically generated with medium confidence

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated with medium confidence

## [1] 0.4196072