Homework 2

#for nicer table output  
library(knitr)

# Conceptual Problems

## Book Problems

Complete the following problems from the Introduction to Statistical Learning with R book (I’m not sure if the problems are in the same order in the python book so use the R book to identify which problems to do).

Section 3.7

* Book Problem 3
* Book Problem 4

Section 6.6

* Book Problem 1
* Book Problem 2
* Book Problem 3
* Book Problem 4

# Implementation Problems

## Problem 1

### Optional - caret Walkthrough

Again, I’m fine with you using any software to complete these problems. I’ll provide guidance on using the [caret package](http://topepo.github.io/caret/index.html) here. If you plan to use another package or other software, feel free to skip this section!

Common tasks when fitting our **predictive** models:

1. Split data into train and test sets
2. Preprocess data (transformations)
3. Fit and tune models on the training set
4. Predict on the test set
5. Split data into train and test sets using createDataPartition(). We’ve seen how to do this in our notes!

library(caret)

Loading required package: ggplot2

Loading required package: lattice

set.seed(31)  
train\_index <- createDataPartition(iris$Sepal.Length, p = 0.7, list = FALSE)  
iris\_train <- iris[train\_index, ]  
iris\_test <- iris[-train\_index, ]

1. Preprocess data. We can easily do transformations on our data. Notes:

* If you use a transformation that relies on the data (i.e. centering or scaling), you must do the transformation only using the training set data!
* When transforming the test set, you must use the training set transformation(s) exactly
  + For instance, if you center and scale a predictor using the training set’s mean and standard deviation, you should use that **same mean and standard deviation from the training set** to standardize the test set values

We can use preProcess() for this purpose!

#standardize all numeric columns   
pre\_process <- preProcess(iris\_train, method = c("center", "scale"))  
pre\_process

Created from 107 samples and 5 variables  
  
Pre-processing:  
 - centered (4)  
 - ignored (1)  
 - scaled (4)

* We can manually transform data if we want (but this isn’t usually necessary):

iris\_train\_transformed <- predict(pre\_process, iris\_train)  
iris\_test\_transformed <- predict(pre\_process, iris\_test)   
head(iris\_train\_transformed) |>  
 kable()

|  | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| --- | --- | --- | --- | --- | --- |
| 2 | -1.1495372 | -0.1587053 | -1.314657 | -1.312561 | setosa |
| 4 | -1.5102425 | 0.0771484 | -1.258836 | -1.312561 | setosa |
| 5 | -1.0293022 | 1.2564166 | -1.314657 | -1.312561 | setosa |
| 6 | -0.5483618 | 1.9639776 | -1.147194 | -1.052720 | setosa |
| 7 | -1.5102425 | 0.7847093 | -1.314657 | -1.182641 | setosa |
| 8 | -1.0293022 | 0.7847093 | -1.258836 | -1.312561 | setosa |

Instead, caret allows us to pass a preProcess argument within the train() function. This will automatically take care of preprocessing prediction made by the model!

1. Fit and tune models on the training set. We’ve seen how to do this in our notes using the train() function. We can use trainControl() to specify our tuning process and use tuneGrid to specify our candidate tuning parameter values.

Set up our training process with trainControl():

train\_control <- trainControl(method = "cv", number = 3)

Set up our tuning grid for a LASSO model:

tune\_grid <- data.frame(alpha = 1, lambda = seq(from = 0.0005, to = 0.5, by = 0.0005))  
head(tune\_grid) |>  
 kable()

| alpha | lambda |
| --- | --- |
| 1 | 0.0005 |
| 1 | 0.0010 |
| 1 | 0.0015 |
| 1 | 0.0020 |
| 1 | 0.0025 |
| 1 | 0.0030 |

Now we add our training and preprocessing info to our train() call (I tend to just define these things in the call). The `method = “glmnet”

iris\_fit <- train(Sepal.Length ~ ., data = iris\_train,   
 method = "glmnet",   
 preProcess = c("center", "scale"),  
 tuneGrid = tune\_grid,  
 trControl = trainControl(method = "cv", number = 3))  
head(iris\_fit$results[, c("alpha", "lambda", "RMSE", "Rsquared")]) |>  
 kable()

| alpha | lambda | RMSE | Rsquared |
| --- | --- | --- | --- |
| 1 | 0.0005 | 0.3145023 | 0.8644042 |
| 1 | 0.0010 | 0.3143967 | 0.8642565 |
| 1 | 0.0015 | 0.3144221 | 0.8639911 |
| 1 | 0.0020 | 0.3145788 | 0.8636038 |
| 1 | 0.0025 | 0.3148699 | 0.8630922 |
| 1 | 0.0030 | 0.3152962 | 0.8624540 |

1. We can easily predict on future observations via predict(). The $finalModel object is used for predictions and caret preserves the proper preprocessing results for us.

pred <- predict(iris\_fit, newdata = iris\_test)

We can find common metrics on the test set with postResample()

postResample(pred, obs = iris\_test$Sepal.Length) |>  
 kable()

|  | x |
| --- | --- |
| RMSE | 0.3120525 |
| Rsquared | 0.8589633 |
| MAE | 0.2637245 |

For a classification task we’ll see how confusionMatrix() can be useful here as well.

**End optional section.**

### Fitting an Elastic Net Model

For this problem we’ll fit an Elastic net model to our bike sharing data set. The code below reads the data in and does the modifications from the notes:

library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.0  
✔ lubridate 1.9.2 ✔ tibble 3.2.1  
✔ purrr 1.0.1 ✔ tidyr 1.3.0  
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
✖ purrr::lift() masks caret::lift()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

#read in data from previous set of notes  
bike\_share <- read\_csv("https://www4.stat.ncsu.edu/online/datasets/SeoulBikeData.csv",  
 local = locale(encoding = "latin1"))

Rows: 8760 Columns: 14  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (4): Date, Seasons, Holiday, Functioning Day  
dbl (10): Rented Bike Count, Hour, Temperature(°C), Humidity(%), Wind speed ...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

bike\_share <- bike\_share |>  
 rename("date" = "Date",  
 "rented\_bike\_count" = `Rented Bike Count`,  
 "hour" = "Hour",  
 "temperature" = `Temperature(°C)`,  
 "humidity" = `Humidity(%)`,  
 "wind\_speed" = `Wind speed (m/s)`,  
 "visibility" = `Visibility (10m)`,  
 "dew\_point\_temperature" = `Dew point temperature(°C)`,  
 "solar\_radiation" = `Solar Radiation (MJ/m2)`,  
 "rainfall" = `Rainfall(mm)`,  
 "snowfall" = `Snowfall (cm)`,  
 "seasons" = "Seasons",  
 "holiday" = "Holiday",  
 "functioning\_day" = "Functioning Day"   
 ) |>  
 mutate(date = dmy(date), #convert the date variable from character  
 seasons = factor(seasons),  
 holiday = factor(holiday),  
 functioning\_day = factor(functioning\_day),  
 log\_rented\_bike\_count = log(rented\_bike\_count)) |>  
 filter(functioning\_day == "Yes")

1. First, split the data into a train/test set.
2. Create your own grid of tuning parameter values (in R the expand.grid() function can be useful for creating a data frame at every combination of two vectors), use 5 fold CV, preprocess your numeric predictors, and select your optimal tuning parameters on the training set.
3. Determine the RMSE on the test data set using your best model.

## Problem 2

The datasets package in R has a data set called trees (datasets::trees in R). The data is also available at <https://www4.stat.ncsu.edu/online/datasets/trees.csv> if you are using another software.

1. First, create scatter plots between the predictors and the response (two scatter plots).

Now we want to fit a couple of linear regression models to predict Height of the trees.

1. Fit a model with main effects for both Girth and Volume. Are either of these predictors statistically significant? Report the relevant hypothesis tests, p-values, and interpretations. Are the assumptions of inference met (at least roughly) here? Provide evidence (plots!).
2. Produce a predicted height for a tree with a Girth of 10 and a Volume of 12.
3. Fit a model with main effects for both Girth and Volume along with their interaction. Are either of these predictors statistically significant? Report the relevant hypothesis tests, p-values, and interpretations. Are the assumptions of inference met (at least roughly) here? Can you explain any discrepancies in the importance of the predictors in this model as compared to the first model we fit?
4. Produce a predicted height for a tree with a Girth of 10 and a Volume of 12.
5. Fit two different models. First a model with just Volume as a predictor. Second a model with Volume, Volume squared, and Volume cubed. Recall the I() function if you are using R. Perform the F-test discussed in the notes to determine if the squared and cubic terms are important for the model.

## Problem 3

Using the Carseats data from the ISLR2 package, split the data into a training and a test set (80/20). Then use best subset, backward, and forward selection to determine the optimal model size for each type of model (similar to how we did this in the notes, no need to use CV here). Once you’ve determined the best size for each model, fit that to the entire data set.