Homework 2

AUTHOR
Collin Real (yhi267)

Reminder: All homework solutions must be written up independently, even though you are allowed to discuss with other students. You need to save your homework assignment in a pdf/html format and upload it with the R code (.R or .rmd) into the Canvas before 11:59pm CT on the due day. No late homework assignment will be graded in any circumstance.

Problem 1 (50 points): Infrared (IR) spectroscopy technology is used to determine the chemical makeup of a substance. The theory of IR spectroscopy holds that unique molecular structures absorb IR frequencies differently. In practice a spectrometer fires a series of IR frequencies into a sample material, and the device measures the absorbance of the sample at each individual frequency. This series of measurements creates a spectrum profile which can then be used to determine the chemical makeup of the sample material.

A Tecator Infratec Food and Feed Analyzer instrument was used to analyze 215 samples of meat across 100 frequencies. A sample of these frequency profiles is displayed in Fig. 6.20. In addition to an IR profile, analytical chemistry determined the percent content of water, fat, and protein for each sample. If we can establish a predictive relationship between IR spectrum and fat content, then food scientists could predict a sample's fat content with IR instead of using analytical chemistry. This would provide costs savings, since analytical chemistry is a more expensive, time-consuming process.

a. Start R and use these commands to load the data:

```
library(caret)
library(e1071)
library(nnet)
library(earth)
library(kernlab)
library(elasticnet)
data(tecator)
```

```
?tecator
```

```
# Inspect the structure of absorp matrix
str(absorp)
```

```
 num [1:215, 1:100] 2.62 2.83 2.58 2.82 2.79 ...
```

```
# Inspect the structure of endpoints matrix
str(endpoints)
```

```
num [1:215, 1:3] 60.5 46 71 72.8 58.3 44 44 69.3 61.4 61.4 ...
```

```
moisture <- endpoints[,1]
fat <- endpoints[,2]
protein <- endpoints[,3]
```

b. Split the data into a training and a test set the response of the percentage of protein, pre-process the data as appropriate.

```
# Extract the protein data
protein = endpoints[,3]
colnames(absorp) <- paste0("V", 1:ncol(absorp))

# Split the data
set.seed(123)  # For reproducibility
trainindex <- createDataPartition(protein, p = .8,
                                  list = FALSE,
                                  times = 1)
traindata <- absorp[trainindex,]
testdata <- absorp[-trainindex,]
trainprotein <- protein[trainindex]
testprotein <- protein[-trainindex]

# Preprocess the data
preProcValues <- preProcess(traindata, method = c("center", "scale"))
traindataTransformed <- predict(preProcValues, traindata)
testdataTransformed <- predict(preProcValues, testdata)
```

```
# Display the structure of the transformed data
str(traindataTransformed)
```

```
num [1:174, 1:100] -0.6089 -0.0256 -0.11 0.378 1.0692 ...
- attr(*, "dimnames")=List of 2
 ..$ : NULL
 ..$ : chr [1:100] "V1" "V2" "V3" "V4" ...
```

```
str(testdataTransformed)
```

```
num [1:41, 1:100] -0.52406 0.00276 0.42899 -0.74749 1.19644 ...
- attr(*, "dimnames")=List of 2
 ..$ : NULL
 ..$ : chr [1:100] "V1" "V2" "V3" "V4" ...
```

c. Build at least three models described Chapter 6: ordinary least squares, PCR, PLS, Ridge, and ENET. For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

Principal Component Regression

```
pcr <- train(traindataTransformed, trainprotein, method = "pcr",
                  trControl = trainControl(method = "cv", number = 10),
```

```
                  tuneLength = 10)

best_pcr <- pcr$bestTune
best_pcr
```

```
   ncomp
10    10
```

Ordinary Least Squares

```
ols <- train(traindataTransformed, trainprotein, method = "lm")
summary(ols)
```

```

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.11284 -0.12785  0.00793  0.15999  0.67158

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.766e+01  3.179e-02 555.541  < 2e-16 ***
V1           8.609e+01  5.605e+02   0.154  0.87834
V2          -1.749e+03  9.905e+02  -1.766  0.08163 .
V3           3.692e+03  1.850e+03   1.996  0.04966 *
V4          -6.752e+03  3.094e+03  -2.182  0.03233 *
V5           8.175e+03  3.764e+03   2.172  0.03310 *
V6          -3.714e+03  3.318e+03  -1.119  0.26670
V7           2.669e+03  2.177e+03   1.226  0.22418
V8          -1.906e+03  1.379e+03  -1.382  0.17111
V9           3.186e+02  1.073e+03   0.297  0.76725
V10         -9.548e+02  1.383e+03  -0.690  0.49224
V11          3.468e+02  2.045e+03   0.170  0.86578
V12         -3.593e+03  3.345e+03  -1.074  0.28634
V13          3.744e+03  4.232e+03   0.885  0.37915
V14          2.795e+02  3.769e+03   0.074  0.94110
V15         -2.370e+03  2.588e+03  -0.916  0.36266
V16          3.428e+03  1.715e+03   1.999  0.04932 *
V17         -6.081e+02  1.330e+03  -0.457  0.64896
V18         -1.456e+03  1.292e+03  -1.127  0.26360
V19         -1.762e+03  1.986e+03  -0.888  0.37766
V20          1.394e+03  3.473e+03   0.401  0.68943
V21          6.063e+03  4.473e+03   1.355  0.17946
V22         -1.142e+04  5.038e+03  -2.266  0.02643 *
V23          7.375e+03  4.151e+03   1.777  0.07980 .
V24         -1.115e+03  2.902e+03  -0.384  0.70189
V25          7.596e+02  1.828e+03   0.415  0.67902
V26         -1.796e+03  1.281e+03  -1.403  0.16497
V27          1.384e+03  1.492e+03   0.928  0.35664
V28         -1.156e+03  1.743e+03  -0.664  0.50908
V29          1.356e+03  2.775e+03   0.489  0.62650
V30         -1.739e+03  4.246e+03  -0.409  0.68341
```

V31	2.116e+03	4.805e+03	0.440	0.66094	
V32	−1.584e+03	4.259e+03	−0.372	0.71095	
V33	1.715e+03	3.087e+03	0.556	0.58022	
V34	−2.526e+03	2.286e+03	−1.105	0.27266	
V35	2.679e+03	1.645e+03	1.628	0.10779	
V36	−2.148e+03	1.530e+03	−1.403	0.16475	
V37	2.559e+03	1.764e+03	1.451	0.15116	
V38	−2.874e+03	2.169e+03	−1.325	0.18926	
V39	−5.259e+02	2.737e+03	−0.192	0.84819	
V40	4.406e+03	4.126e+03	1.068	0.28911	
V41	−4.284e+03	5.258e+03	−0.815	0.41793	
V42	−2.015e+03	5.252e+03	−0.384	0.70230	
V43	7.425e+03	4.576e+03	1.623	0.10895	
V44	−1.422e+03	3.352e+03	−0.424	0.67265	
V45	−6.595e+03	2.678e+03	−2.463	0.01615	*
V46	4.926e+03	1.690e+03	2.915	0.00472	**
V47	−2.558e+02	8.422e+02	−0.304	0.76219	
V48	−1.671e+03	1.133e+03	−1.475	0.14455	
V49	6.236e+02	1.786e+03	0.349	0.72798	
V50	1.332e+03	2.112e+03	0.631	0.53032	
V51	−4.386e+02	3.090e+03	−0.142	0.88751	
V52	−1.528e+03	3.947e+03	−0.387	0.69969	
V53	3.140e+03	3.987e+03	0.787	0.43356	
V54	−4.351e+03	3.294e+03	−1.321	0.19064	
V55	4.241e+03	2.606e+03	1.627	0.10800	
V56	−3.263e+03	1.871e+03	−1.743	0.08546	.
V57	1.952e+03	1.493e+03	1.308	0.19501	
V58	−4.492e+02	1.225e+03	−0.367	0.71501	
V59	−3.681e+02	1.134e+03	−0.325	0.74648	
V60	2.471e+02	1.040e+03	0.238	0.81284	
V61	−4.492e+02	9.326e+02	−0.482	0.63148	
V62	1.181e+03	9.943e+02	1.188	0.23863	
V63	−2.407e+03	1.374e+03	−1.752	0.08400	.
V64	3.116e+03	2.145e+03	1.453	0.15045	
V65	−2.128e+03	3.362e+03	−0.633	0.52871	
V66	8.902e+02	4.479e+03	0.199	0.84302	
V67	−1.546e+03	4.768e+03	−0.324	0.74672	
V68	1.968e+03	4.214e+03	0.467	0.64185	
V69	4.016e+02	3.163e+03	0.127	0.89931	
V70	−3.330e+03	2.158e+03	−1.543	0.12710	
V71	4.407e+03	1.671e+03	2.637	0.01020	*
V72	−3.167e+03	1.591e+03	−1.991	0.05024	.
V73	1.841e+03	1.404e+03	1.312	0.19378	
V74	4.641e+02	1.388e+03	0.334	0.73901	
V75	−4.377e+02	1.384e+03	−0.316	0.75271	
V76	−2.393e+03	1.395e+03	−1.716	0.09044	.
V77	2.372e+03	1.213e+03	1.956	0.05435	.
V78	1.430e+02	1.474e+03	0.097	0.92301	
V79	−2.026e+03	1.486e+03	−1.364	0.17681	
V80	−4.792e+02	1.874e+03	−0.256	0.79893	
V81	3.413e+03	2.007e+03	1.701	0.09326	.
V82	−5.212e+03	2.302e+03	−2.264	0.02651	*
V83	2.575e+03	2.695e+03	0.956	0.34238	
V84	3.497e+03	2.793e+03	1.252	0.21454	
V85	−6.831e+03	2.840e+03	−2.406	0.01868	*

```
V86          6.149e+03  3.292e+03   1.868  0.06578 .
V87         -7.247e+02  3.759e+03  -0.193  0.84768
V88         -4.792e+03  3.655e+03  -1.311  0.19394
V89          4.867e+03  3.314e+03   1.468  0.14629
V90         -1.926e+02  3.962e+03  -0.049  0.96137
V91          2.546e+03  4.534e+03   0.562  0.57609
V92         -7.643e+03  3.876e+03  -1.972  0.05241 .
V93          3.946e+03  2.992e+03   1.319  0.19131
V94          1.661e+03  2.703e+03   0.614  0.54082
V95         -3.567e+03  2.278e+03  -1.566  0.12177
V96          3.306e+03  2.038e+03   1.622  0.10910
V97         -1.749e+03  2.069e+03  -0.845  0.40070
V98          1.383e+03  1.878e+03   0.737  0.46375
V99         -2.236e+03  1.775e+03  -1.260  0.21158
V100         1.132e+03  8.290e+02   1.366  0.17629
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4194 on 73 degrees of freedom
Multiple R-squared:  0.992, Adjusted R-squared:  0.9811
F-statistic: 90.77 on 100 and 73 DF,  p-value: < 2.2e-16
```

Partial Least Squares

```
pls <- train(traindataTransformed, trainprotein, method = "pls",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 10)

best_pls <- pls$bestTune
best_pls
```

```
   ncomp
10    10
```

Based on the cross-validation, the optimal values of the tuning parameters for both PCR & PLS models is 10.

Evaluate

```
# Predictions
ols_preds <- predict(ols, newdata = testdataTransformed)
pcr_preds <- predict(pcr, newdata = testdataTransformed)
pls_preds <- predict(pls, newdata = testdataTransformed)

# Evaluation
ols_res <- postResample(ols_preds, testprotein)
pcr_res <- postResample(pcr_preds, testprotein)
pls_res <- postResample(pls_preds, testprotein)

# Display the results
ols_res
```

```
     RMSE  Rsquared       MAE
1.1510006 0.8744041 0.7897741
```

```
pcr_res
```

```
     RMSE  Rsquared       MAE
0.7985893 0.9242419 0.6548281
```

```
pls_res
```

```
     RMSE  Rsquared       MAE
0.7353164 0.9379046 0.5888808
```

d. Build nonlinear models in Chapter 7: SVM, neural network, MARS, and KNN models. Since neural networks are especially sensitive to highly correlated predictors, does pre-processing using PCA help the model? For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

```
pcaPreProc <- preProcess(traindata, method = c("center", "scale", "pca"))
traindataPCA <- predict(pcaPreProc, traindata)
testdataPCA <- predict(pcaPreProc, testdata)
```

Support Vector Machine

```
svm <- train(traindataTransformed, trainprotein, method = "svmRadial",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 10)
best_svm <- svm$bestTune
print(best_svm)
```

```
      sigma  C
7 0.1316487 16
```

```
svm_preds <- predict(svm, newdata = testdataTransformed)
svm_res <- postResample(svm_preds, testprotein)
print(svm_res)
```

```
     RMSE  Rsquared       MAE
1.6514511 0.6796184 1.1385809
```

Neural Network

```
nn <- train(traindataPCA, trainprotein, method = "nnet",
                  trControl = trainControl(method = "cv", number = 10),
                  tuneLength = 10, trace = FALSE, linout = TRUE)
best_nn <- nn$bestTune
print(best_nn)
```

```
   size        decay
13    3 0.0002371374
```

```
nn_preds <- predict(nn, newdata = testdataPCA)
```

```
nn_res <- postResample(nn_preds, testprotein)
print(nn_res)
```

```
     RMSE  Rsquared       MAE 
2.5493629 0.2341758 2.2361913 
```

Multivariate Adaptive Regression Splines

```
mars <- train(traindataTransformed, trainprotein, method = "earth",
                    trControl = trainControl(method = "cv", number = 10),
                    tuneLength = 10)
best_mars <- mars$bestTune
print(best_mars)
```

```
  nprune degree
8     20      1
```

```
mars_preds <- predict(mars, newdata = testdataTransformed)
mars_res <- postResample(mars_preds, testprotein)
print(mars_res)
```

```
     RMSE  Rsquared       MAE 
0.8360361 0.9164617 0.6599512 
```

k-Nearest Neighbors

```
knn <- train(traindataTransformed, trainprotein, method = "knn",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 10)
best_knn <- knn$bestTune
print(best_knn)
```

```
  k
2 7
```

```
knn_preds <- predict(knn, newdata = testdataTransformed)
knn_res <- postResample(knn_preds, testprotein)
print(knn_res)
```

```
     RMSE  Rsquared       MAE 
2.1325705 0.4868149 1.7132404 
```

Optimal Tuning Parameters

Support Vector Machine

- sigma: 0.1316487
- C: 16

Neural Network

- size: 3
- decay: 0.0002371374

MARS
- nprune: 20
- degree: 1

kNN
- k: 7

```
# Neural Network (without PCA)
nn <- train(traindataTransformed, trainprotein, method = "nnet",
                  trControl = trainControl(method = "cv", number = 10),
                  tuneLength = 10, trace = FALSE, linout = TRUE)
nn_preds <- predict(nn, newdata = testdataTransformed)
nn_res <- postResample(nn_preds, testprotein)

# Neural Network (with PCA)
nn_pca <- train(traindataPCA, trainprotein, method = "nnet",
                      trControl = trainControl(method = "cv", number = 10),
                      tuneLength = 10, trace = FALSE, linout = TRUE)
nn_pca_preds <- predict(nn_pca, newdata = testdataPCA)
nn_pca_res <- postResample(nn_pca_preds, testprotein)
cat("Without PCA:\n", nn_res)
```

```
Without PCA:
 0.5119102 0.9686138 0.4026765
```

```
cat("With PCA:\n", nn_pca_res)
```

```
With PCA:
 2.432218 0.3148997 1.922737
```

The neural network with PCA **performs worse** than the neural net without PCA. Without PCA, the model has a lower RMSE, higher R-squared, and lower MAE.

e. Which model from parts c and d has the best predictive ability? Is any model significantly better or worse than the others?

The **Partial Least Squares** model has the best predictive ability because it has the lowest RMSE and highest R-squared. In general, linear models performed better than nonlinear models.

Problem 2 (30 points): Developing a model to predict permeability (see Sect. 1.4 of the textbook) could save significant resources for a pharmaceutical company, while at the same time more rapidly identifying molecules that have a sufficient permeability to become a drug:

a. Start R and use these commands to load the data

```
# Load library and attach data
library(AppliedPredictiveModeling)
```

```
data(permeability)

# Explore the data structure
str(fingerprints)
```

```
 num [1:165, 1:1107] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:165] "1" "2" "3" "4" ...
  ..$ : chr [1:1107] "X1" "X2" "X3" "X4" ...
```

```
str(permeability)
```

```
 num [1:165, 1] 12.52 1.12 19.41 1.73 1.68 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:165] "1" "2" "3" "4" ...
  ..$ : chr "permeability"
```

b. The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparse meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using the nearZeroVar function from the caret package. How many predictors are left for modeling?

```
dim(fingerprints)
```

```
[1]  165 1107
```

```
colnames(fingerprints) <- paste0("X", 1:ncol(fingerprints))

# Identify near-zero variance predictors
nzv <- nearZeroVar(fingerprints)

# Filter near-zero variance predictors
filtered_fingerprints <- fingerprints[, -nzv]

remaining_predictors <- ncol(filtered_fingerprints)
cat("Remaining predictors:", remaining_predictors)
```

```
Remaining predictors: 388
```

c. Split the data into a training and a test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R2?

```
permeability_df <- data.frame(filtered_fingerprints, permeability = permeability)

# Split the data
set.seed(123)
trainindex <- createDataPartition(permeability_df$permeability, p = .8,
                                  list = FALSE,
                                  times = 1)
traindata <- permeability_df[trainindex,]
testdata <- permeability_df[-trainindex,]
```

```
# Preprocess the data
preProcValues <- preProcess(traindata, method = c("center", "scale"))
traindataTransformed <- predict(preProcValues, traindata)
testdataTransformed <- predict(preProcValues, testdata)

# Fit PLS model
pls <- train(permeability ~ ., data = traindataTransformed, method = "pls",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 20)

# Best tuning parameter
best_pls <- pls$bestTune
print("The number of latent variables that are optimal:")
```

```
[1] "The number of latent variables that are optimal:"
```

```
print(best_pls)
```

```
  ncomp
8     8
```

```
# R-squared resampled estimate
resampled_R2 <- max(pls$results$Rsquared)
cat("The resampled estimate of R-squared is:", resampled_R2)
```

```
The resampled estimate of R-squared is: 0.5496288
```

d. Predict the response for the test set. What is the test set estimate of R2?

```
# Predict test set
pls_preds <- predict(pls, newdata = testdataTransformed)

# Calculate test set R-squared
test_R2 <- R2(pls_preds, testdataTransformed$permeability)
cat("The test set estimate of R-squared is:", test_R2)
```

```
The test set estimate of R-squared is: 0.3729547
```

e. Try building other models discussed in this chapter. Do any have better predictive performance?

Ridge Regression

```
ridge <- train(permeability ~ ., data = traindataTransformed, method = "ridge",
                     trControl = trainControl(method = "cv", number = 10),
                     tuneLength = 20)
ridge_preds <- predict(ridge, newdata = testdataTransformed)
ridge_res <- postResample(ridge_preds, testdataTransformed$permeability)
print(ridge_res)
```

```
     RMSE  Rsquared       MAE
0.7921829 0.4007391 0.5682279
```

Lasso Regression

```
lasso <- train(permeability ~ ., data = traindataTransformed, method = "lasso",
                     trControl = trainControl(method = "cv", number = 10),
                     tuneLength = 20)
lasso_preds <- predict(lasso, newdata = testdataTransformed)
lasso_res <- postResample(lasso_preds, testdataTransformed$permeability)
print(lasso_res)
```

```
     RMSE  Rsquared       MAE
0.6835848 0.3617548 0.4443482
```

The **Lasso Regression** has better predictive performance with a lower RMSE and MAE; however, the amount of variance this model explains is significantly reduced.

Problem 3 (20 points): Return to the permeability problem outlined in Problem 2. Train several nonlinear regression models and evaluate the resampling and test set performance.

a. Which nonlinear regression model that we learned in Chapter 7 gives the optimal resampling and test set performance?

```
# SVM model
svm <- train(permeability ~ ., data = traindataTransformed, method = "svmRadial",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 10)
best_svm <- svm$bestTune
print(best_svm)
```

```
        sigma  C
7 0.001635739 16
```

```
svm_preds <- predict(svm, newdata = testdataTransformed)
svm_res <- postResample(svm_preds, testdataTransformed$permeability)
print(svm_res)
```

```
     RMSE  Rsquared       MAE
0.6559098 0.4872192 0.4199596
```

```
# Neural Network model
nn <- train(permeability ~ ., data = traindataTransformed, method = "nnet",
                  trControl = trainControl(method = "cv", number = 10),
                  tuneLength = 10, trace = FALSE, linout = TRUE)
best_nn <- nn$bestTune
print(best_nn)
```

```
  size      decay
```

```
8    1 0.01778279
```

```
nn_preds <- predict(nn, newdata = testdataTransformed)
nn_res <- postResample(nn_preds, testdataTransformed$permeability)
print(nn_res)
```

```
     RMSE  Rsquared       MAE
0.7979915 0.2768253 0.5537823
```

```
# MARS model
mars <- train(permeability ~ ., data = traindataTransformed, method = "earth",
                    trControl = trainControl(method = "cv", number = 10),
                    tuneLength = 10)
best_mars <- mars$bestTune
print(best_mars)
```

```
  nprune degree
1      2      1
```

```
mars_preds <- predict(mars, newdata = testdataTransformed)
mars_res <- postResample(mars_preds, testdataTransformed$permeability)
print(mars_res)
```

```
     RMSE  Rsquared       MAE
0.7232002 0.2541716 0.5483070
```

```
# KNN model
knn <- train(permeability ~ ., data = traindataTransformed, method = "knn",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneLength = 10)
best_knn <- knn$bestTune
print(best_knn)
```

```
  k
3 9
```

```
knn_preds <- predict(knn, newdata = testdataTransformed)
knn_res <- postResample(knn_preds, testdataTransformed$permeability)
print(knn_res)
```

```
     RMSE  Rsquared       MAE
0.7236208 0.2436855 0.4910987
```

b. Do any of the nonlinear models outperform the optimal linear model you previously developed in Problem 2? If so, what might this tell you about the underlying relationship between the predictors and the response?

The SVM model has a lower RMSE and MAE, indicating it has better predictive ability, but is unable to explain as much of the variance as the PLS model. The SVM model should be used if accuracy is the priority of the study, and the PLS model should be used for better understanding the overall relationships in the data.

c. Would you recommend any of the models you have developed to replace the

c. Would you recommend any of the models you have developed to replace the permeability laboratory experiment?

I would recommend the SVM model as a replacement for the permeability lab experiment due to its enhanced predictive ability. I'm more interested in the model's predictive power rather than its ability to explain the variance/relationships in the dataset.