Predictive Modeling Homework 3

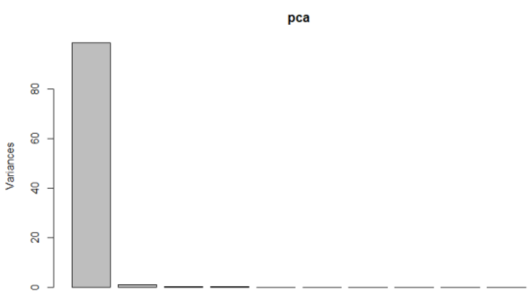
Ian Boulis

6.1

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

> library(caret)
> data(tecator)

B. Use PCA to determine the effective dimension of these data, use 95% as a cut-off value.



From the screeplot, it appears that almost all the variance is accounted for in the first PCA. This is verified by computing the variance of the PCAs:

```
> head(varpca)
[1] 98.626192582  0.969705229  0.279324276  0.114429868  0.006460911  0.002624591
```

C. Split the data into a training and a test set (80/20). Use 3-fold CV to build the PCR model (95% cut off). What is the average RMSE?

```
Linear Regression

174 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 117, 115, 116
Resampling results:

  RMSE    Rsquared   MAE
  7.2135  0.7921591  4.190897
```

D. Use the model on (C) to predict the response for the test set. What is the RMSE?

Using the model from part C to predict on the test set resulted in a RMSE of 2.8921448 and R^2 value of 0.94

```
> defaultSummary(dapa)
     RMSE  Rsquared       MAE
2.8921448 0.9472097 1.7976082
```

6.2

B. Filter out the predictors that have low frequency using the nearZeroVar function. How many predictors are left for modeling?

Values	
nzr	int [1:719] 7 8 9 10 13 14 17 18 19 22 ...

There is a total of 719 predictors with a zero or near zero variance. From our 1,107 predictors, we have 388 predictors left for model building.

C. Split the data into a training and 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 R^2?

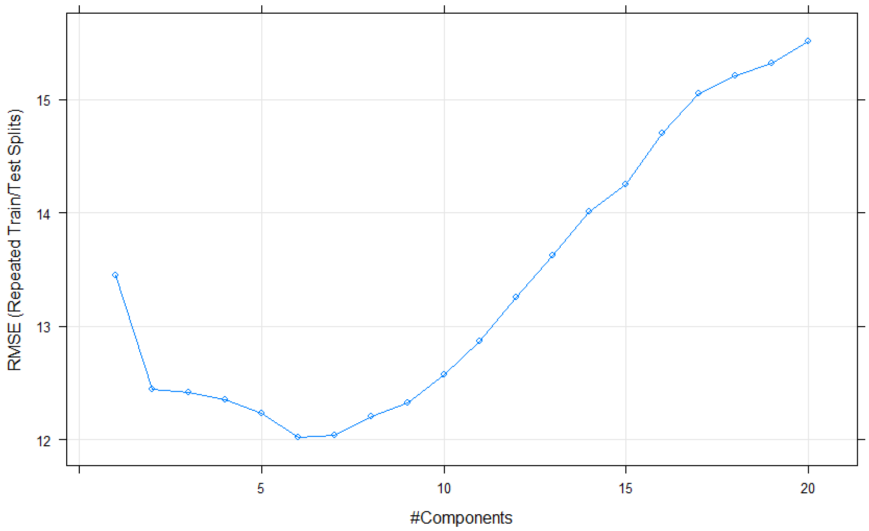
```
Partial Least Squares

133 samples
388 predictors

Pre-processing: centered (388), scaled (388)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 101, 101, 101, 101, 101, 101, ...
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared   MAE
   1     13.44745  0.3158445  10.296566
   2     12.44250  0.4304343   8.625833
   3     12.41566  0.4342389   9.140396
   4     12.35435  0.4472174   9.125192
   5     12.22983  0.4706266   8.956198
   6     12.01679  0.4883515   8.829465
   7     12.03354  0.4921005   9.012227
   8     12.20621  0.4864443   9.259302
   9     12.32250  0.4868289   9.247622
  10     12.56818  0.4756307   9.411210
  11     12.87080  0.4593575   9.605767
  12     13.25810  0.4389713   9.838692
  13     13.62212  0.4222968  10.099519
  14     14.01455  0.4023258  10.352074
  15     14.25061  0.3935322  10.521174
  16     14.70161  0.3712271  10.869507
  17     15.04990  0.3572226  11.100422
  18     15.21098  0.3536296  11.234272
  19     15.31762  0.3520845  11.388072
  20     15.51716  0.3471868  11.555851

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 6.
```



As seen in the pls output from R, the optimal number of latent variables is 6, with a corresponding value R^2 of 0.4883515. However, the number of components with the highest R^2 value is 7 components, with an R^2 value of 0.4921005.

D. Predict the response for the test set. What is the test set estimate of R^2?

Using the same model from part (C) resulted in a RMSE of 10.3124854 and an R^2 value of 0.70

```
> defaultSummary(dapa)
      RMSE   Rsquared        MAE
10.3124854  0.7055006  7.7924576
```

6.3

B. A small percentage of calls in the predictor set contains missing values. Use an imputation function to fill in these missing values

```
impute <- preProcess(trp, method=c("center","scale","BoxCox","knnImpute"))
```

After splitting the data, I used the preProcess function on the predictor training set to compact the pre-processing steps. I took a look at the distribution for the response variable (Yield in this data set), decided to center and scale then preform the Box-Cox transformation, then used the built in knnImpute method to impute the data. The default value for k in this context is k=5.

C. Split the data into a training and a test set (80%/20% in training/test sets), pre-process the data, and use 3-fold CV to build lm, Ridge, lasso, and elastic net models on the training set. What is the average RMSE for each model on the training set? What are the best tuning parameters for each model?

As mentioned in part B, the data was prepressed with center and scaling, and then a Box-Cox transformation.

Model	**Best Tuning Parameter**	**Training**		**Testing**	
		RMSE	**R^2**	**RMSE**	**R^2**
Linear Model		2.714181	0.4164144	14.2545	0.04543042
Ridge	$\Lambda = 0.1$	3.60031	0.3977611	1.980587	0.3005124
LASSO	Fraction= 0.1	1.122202	0.657	1.671107	0.15535395
E-Net	Fraction=0.525 $\Lambda = 0.1$	1.135579	0.6296772	1.297186	0.5380803

```
Linear Regression

144 samples
 46 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results:

  RMSE      Rsquared   MAE
  2.714181  0.4164144  1.371673

Tuning parameter 'intercept' was held constant at a value of TRUE
```

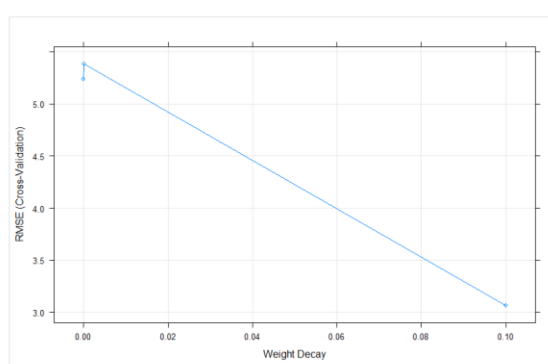
```
Ridge Regression

144 samples
 46 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  lambda  RMSE      Rsquared   MAE
  0e+00   9.506434  0.1129453  2.367087
  1e-04   9.469312  0.1178802  2.355339
  1e-01   3.600305  0.3977611  1.351270

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.1.
```



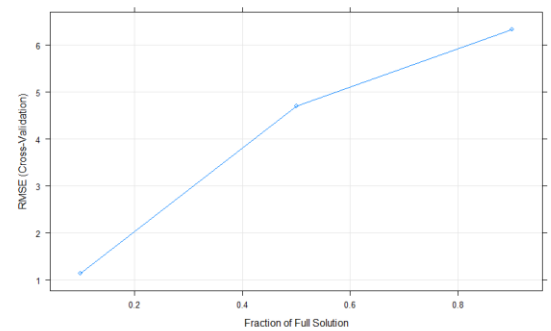
```
The lasso

144 samples
 46 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  fraction  RMSE      Rsquared   MAE
  0.1       1.122202  0.6570048  0.9113331
  0.5       3.393380  0.2721675  1.4136536
  0.9       4.616505  0.1955870  1.7368916

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.1.
```



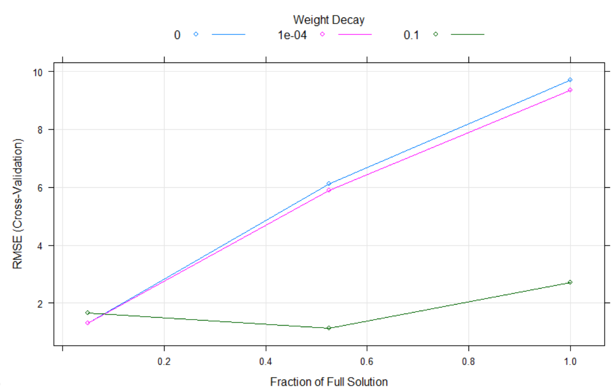

```
Elasticnet

144 samples
 46 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  lambda  fraction  RMSE      Rsquared   MAE
  0e+00   0.050     1.298305  0.6186981  1.0442678
  0e+00   0.525     6.121063  0.2764137  1.8889712
  0e+00   1.000     9.705170  0.1606099  2.5893353
  1e-04   0.050     1.315788  0.6174787  1.0572792
  1e-04   0.525     5.887176  0.2843516  1.8437054
  1e-04   1.000     9.357351  0.1638396  2.5133665
  1e-01   0.050     1.663812  0.4833315  1.3337512
  1e-01   0.525     1.135579  0.6296772  0.9127071
  1e-01   1.000     2.719747  0.4157516  1.3019369

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.525 and lambda = 0.1.
```



D. Predict the response for the test set for each model in (C). What is the RMSE for each model? How does the RMSE compare with the RMSE in the training set?

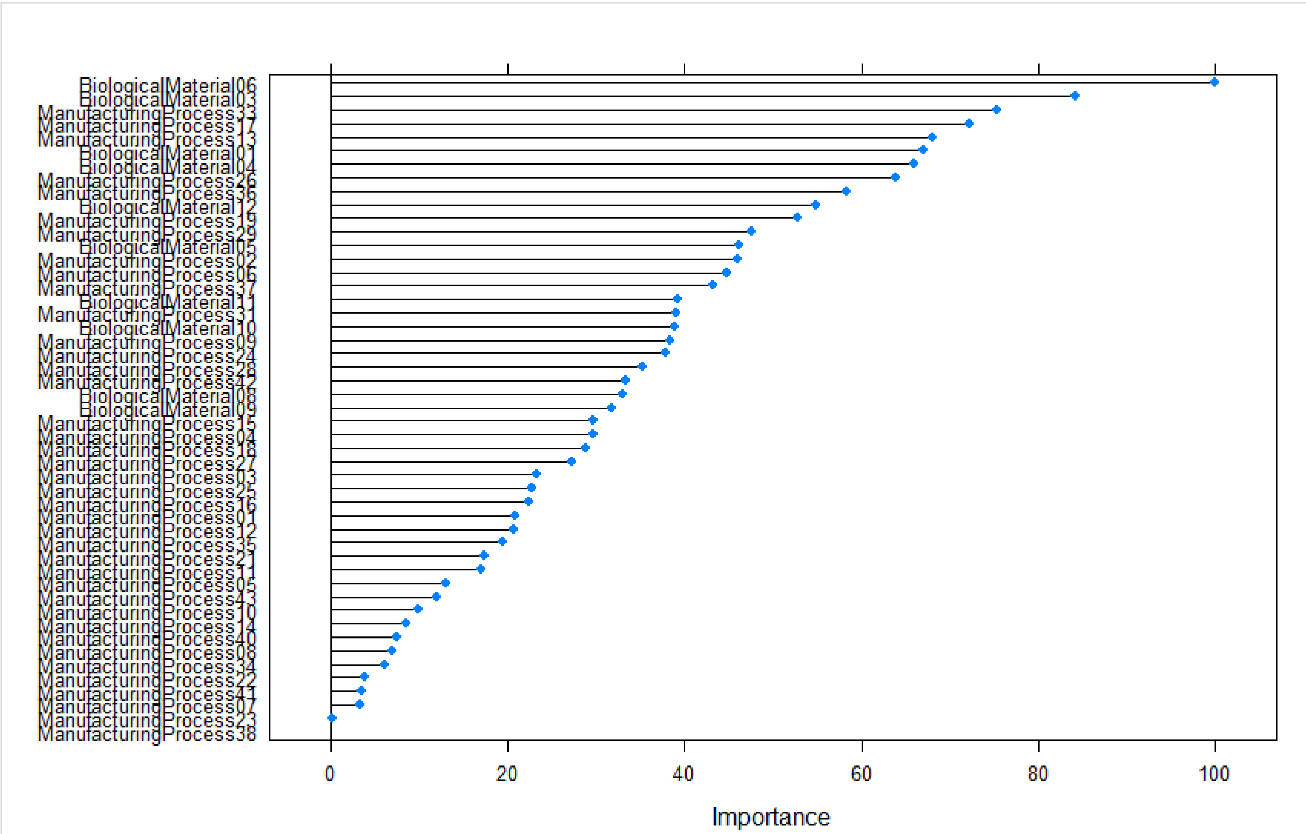
The root mean squared error along with the R^2 values for all four of the models can be seen in the table in part C. For the linear model, there was quite a stark difference between the training and test set. The RMSE increased significantly and the R^2 value got significantly worse. For the ridge model, the RMSE showed a better value, with it almost halving between the training and test sets, while the R^2 value did decrease it was not as significant as the RMSE. The lasso model did not exhibit that much of a change in RMSE between training and testing, however, the R^2 did get significantly worse. Finally, the E-Net model exhibited minor changes to both RMSE and R^2.

E. Which model has the best predictive ability? Explain which model you would use for predicting the response.

I would choose the E-Net model for predicting the response in this data set. From the table in part C, the E-Net- model has the most comparable RMSE and R^2 values. The difference between the training and testing sets for this model are both ~0.1, with RMSE increasing slightly over 0.1, and R^2 decreasing less than 0.1.

F. Based on the best model, which predictors are most important.

Plotting the varImp function we can see that “BiologicalMaterial06” is the most important predictor.



```
##6.1##
##A##
library(AppliedPredictiveModeling)
library(caret)
data(tecator)
##B##
pca <- prcomp(absorp, center=TRUE, scale=TRUE, tol=0.95)
screeplot(pca)
varpca <- pca$sdev^2/sum(pca$sdev^2)*100
head(varpca)
##C##
absorp <- data.frame(absorp)
endpoints <- data.frame(endpoints)
partition <- createDataPartition(endpoints$X2, p=4/5, list=FALSE)
trabsorp <- absorp[partition, ]
teabsorp <- absorp[-partition, ]
trpro <- endpoints$X2[partition]
tepro <- endpoints$X2[-partition ]
ctrl <- trainControl(method="cv", number=3)
Linmod <- train(trabsorp,trpro, method = "lm", trControl = ctrl, preProcess=c("center","scale"))
Linmod
##D##
##test <- train(teabsorp, tepro, method="lm", trControl=ctrl)##
pred <- predict(Linmod, teabsorp)
dapa <- data.frame(obs=tepro, pred=pred)
defaultSummary(dapa)
```

```
##6.2##

##A##

library(AppliedPredictiveModeling)

data("permeability")

library(pls)

##B##

nzr <- nearZeroVar(fingerprints)

nzv <- fingerprints[,-nzr]

##C##

partition <- createDataPartition(permeability, p=4/5, list=FALSE)

trF <- nzv[partition,]

trP <- permeability[partition,]

teF <- nzv[-partition,]

teP <- permeability[-partition]

ctrl <- trainControl(method="repeatedcv",3,3)

pls <- train(trF,trP, method="pls", tuneGrid=expand.grid(ncomp=1:20), trControl=ctrl, preProc =
c("center", "scale"))

plot(pls)

pls

##D##

test <- predict(pls,teF)

dapa <- data.frame(obs=teP, pred=test)

defaultSummary(dapa)
```

```
##6.3##
##A##
library(AppliedPredictiveModeling)
library(caret)
library(RANN)
data("ChemicalManufacturingProcess")

##B##
response <- subset(ChemicalManufacturingProcess, select="Yield")
predict <- subset(ChemicalManufacturingProcess, select=-Yield)
partition <- createDataPartition(response$Yield, p=4/5, list=FALSE)
trp <- predict[partition,]
trr <- response[partition,]
tep <- predict[-partition,]
ter <- response[-partition,]
impute <- preProcess(trp, method=c("center","scale","BoxCox","knnImpute"))
impute

##c##D##E##
trainP <- predict(impute,trp)
nzv <- nearZeroVar(trainP)
trainP <- trainP[-nzv]
corr <- cor(trainP)
hc <- findCorrelation(corr)
trainP <- trainP[, -hc]

timpute <- preProcess(tep, method=c("BoxCox","center","scale","knnImpute"))
tep <- predict(timpute,tep)
nzv <- nearZeroVar(tep)
tep <- tep[-nzv]
```

```r
corr <- cor(tep)
hc <- findCorrelation(corr)
tep <- tep[, -hc]


ctrl <- trainControl(method="cv", 3)
lm <- train(x=trainP, y=trr, method="lm", trControl=ctrl)
ridge <- train(x=trainP, y=trr, method="ridge", trControl=ctrl)
lasso <- train(x=trainP, y=trr, method="lasso", trControl=ctrl)
enet <- train(x=trainP, y=trr, method="enet", trControl=ctrl)
tlm <- train(x=tep, y=ter, method="lm", trControl=ctrl)   ##tlm = test linear model##
tridge <- train(x=tep, y=ter, method="ridge", trControl=ctrl)
tlasso <- train(x=tep, y=ter, method="lasso", trControl=ctrl)
tenet <- train(x=tep, y=ter, method="enet", trControl=ctrl)
##F##
plot(varImp(tenet))
```