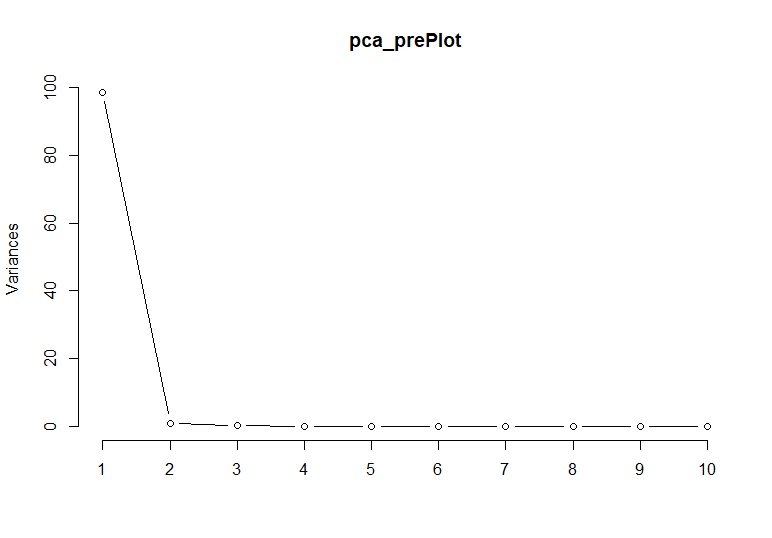
Deanna Springgay

Homework 3

6.1

b) After using PCA to reduce the dimensions of the predictors, the scree plot below indicates that only the first PC is responsible for almost all the variance seen in the data. As a result, the effective dimension for the data is 215 x 1.



c) The RMSE for the data at seed 80 is 0.9069537. The train function returned the following:

Linear Regression

175 samples

1 predictor

No pre-processing

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 115, 117, 118

Resampling results:

RMSE Rsquared MAE

0.9069537 0.2123689 0.7338545

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

d) Running the prediction on the test set returns the RMSE of 0.8663587.

6.2

b) Removing predictors with near zero variance leaves 388 predictors for modeling.

c) Tuning the PLS model indicates that 5 variables are optimal and from this value R² is estimated to be 0.5680874. The output of the train function at seed 80 is as follows:

Partial Least Squares

132 samples

388 predictors

Pre-processing: centered (388), scaled (388)

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 119, 119, 118, 119, 119, 119, ...

Resampling results across tuning parameters:

ncomp RMSE Rsquared MAE

1 13.53847 0.3607088 10.355781

2 12.27115 0.5015298 8.780704

3 12.10827 0.5425497 9.263932

4 12.21064 0.5231330 9.612332

5 11.79890 0.5680874 9.103178

6 11.87708 0.5602446 9.066720

7 11.89119 0.5612925 9.174733

8 12.18187 0.5534121 9.551131

9 12.23159 0.5592414 9.522041

10 12.49627 0.5398298 9.704232

11 12.49109 0.5391408 9.509330

12 12.86087 0.5288506 9.669900

13 13.13384 0.4882350 9.895988

14 13.23226 0.4853990 10.192768

15 13.20880 0.4793521 10.134532

16 13.31044 0.4670755 10.175323

17 13.45443 0.4546536 10.409101

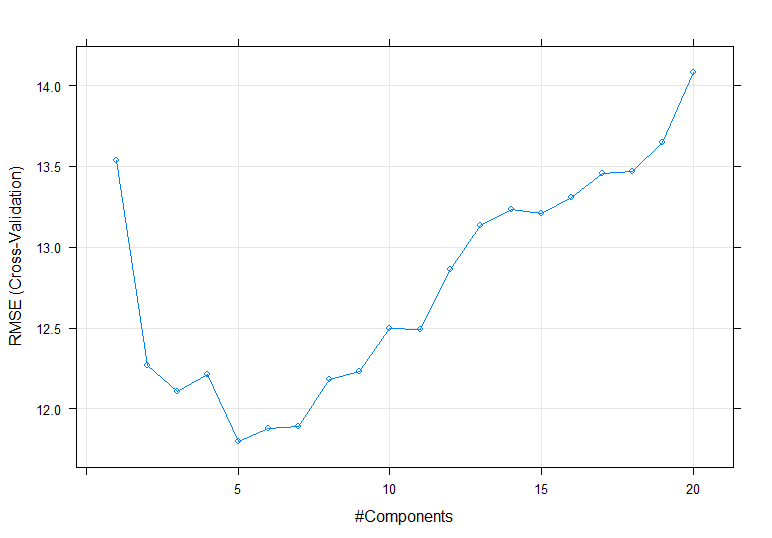
18 13.46875 0.4451923 10.515963

19 13.64878 0.4269145 10.754191

20 14.08113 0.4006257 11.085222

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was ncomp = 5.



d) The test set R² value is 0.3735393.

6.3

b) imputedManufacturing <- predict(preProcess(ChemicalManufacturingProcess, "knnImpute"), ChemicalManufacturingProcess)

c)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Best Tuning Parameter** | **Training** | | **Testing** | |
| **RMSE** | **R²** | **RMSE** | **R²** |
| Linear Model | NA | 5.237901 | 0.06319766 | 0.4634609 | 0.783976 |
| Ridge | lambda = 0.4285714. | 0.9106220 | 0.49032879 | 0.5780152 | 0.7070842 |
| Lasso | alpha = 1 lambda = 0.1 | 0.666336 | 0.5755465 | 0.6179639 | 0.6397258 |
| E-Net | fraction = 0.2 lambda = 0.01 | 0.6562641 | 0.5602628 | 0.5825427 | 0.6704829 |

Linear Regression

176 samples

57 predictor

Pre-processing: centered (57), scaled (57)

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 118, 117, 117

Resampling results:

RMSE Rsquared MAE

5.237901 0.06319766 1.316034

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

Ridge Regression

176 samples

57 predictor

Pre-processing: centered (57), scaled (57)

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 117, 117, 118

Resampling results across tuning parameters:

lambda RMSE Rsquared MAE

0.00000000 3.6218966 0.03926094 1.0874280

0.07142857 1.2168354 0.32305037 0.6542799

0.14285714 1.0447534 0.38721368 0.6203053

0.21428571 0.9694083 0.42653994 0.6193069

0.28571429 0.9315530 0.45541880 0.6220230

0.35714286 0.9143423 0.47630849 0.6280144

0.42857143 0.9106220 0.49032879 0.6356898

0.50000000 0.9166279 0.49861072 0.6452264

0.57142857 0.9300426 0.50237454 0.6561371

0.64285714 0.9492441 0.50278758 0.6685626

0.71428571 0.9729875 0.50085100 0.6862966

0.78571429 1.0002691 0.49735734 0.7059640

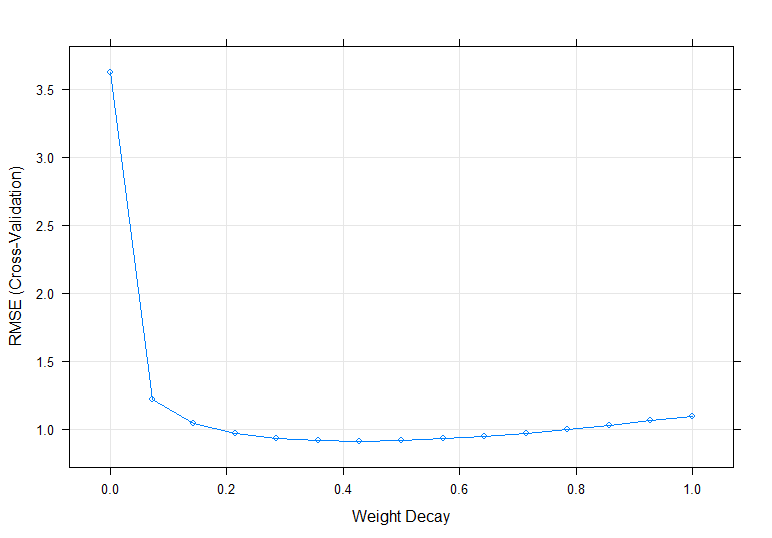
0.85714286 1.0302673 0.49289893 0.7258610

0.92857143 1.0623120 0.48790017 0.7465065

1.00000000 1.0958625 0.48265514 0.7674043

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was lambda = 0.4285714.



glmnet

176 samples

57 predictor

Pre-processing: centered (57), scaled (57)

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 116, 117, 119

Resampling results across tuning parameters:

lambda RMSE Rsquared MAE

0.1 0.6663368 0.5755465 0.5374191

0.2 0.7004277 0.5973482 0.5637347

0.3 0.7668954 0.5890993 0.6198540

0.4 0.8528866 0.5181677 0.6871005

0.5 0.9280339 0.4549232 0.7510163

0.6 0.9892736 0.3512036 0.8029230

0.7 0.9976602 NaN 0.8103499

0.8 0.9976602 NaN 0.8103499

0.9 0.9976602 NaN 0.8103499

1.0 0.9976602 NaN 0.8103499

1.1 0.9976602 NaN 0.8103499

1.2 0.9976602 NaN 0.8103499

1.3 0.9976602 NaN 0.8103499

1.4 0.9976602 NaN 0.8103499

1.5 0.9976602 NaN 0.8103499

1.6 0.9976602 NaN 0.8103499

1.7 0.9976602 NaN 0.8103499

1.8 0.9976602 NaN 0.8103499

1.9 0.9976602 NaN 0.8103499

2.0 0.9976602 NaN 0.8103499

2.5 0.9976602 NaN 0.8103499

3.0 0.9976602 NaN 0.8103499

3.5 0.9976602 NaN 0.8103499

4.0 0.9976602 NaN 0.8103499

4.5 0.9976602 NaN 0.8103499

5.0 0.9976602 NaN 0.8103499

6.0 0.9976602 NaN 0.8103499

7.0 0.9976602 NaN 0.8103499

8.0 0.9976602 NaN 0.8103499

9.0 0.9976602 NaN 0.8103499

10.0 0.9976602 NaN 0.8103499

11.0 0.9976602 NaN 0.8103499

12.0 0.9976602 NaN 0.8103499

13.0 0.9976602 NaN 0.8103499

14.0 0.9976602 NaN 0.8103499

15.0 0.9976602 NaN 0.8103499

16.0 0.9976602 NaN 0.8103499

17.0 0.9976602 NaN 0.8103499

18.0 0.9976602 NaN 0.8103499

19.0 0.9976602 NaN 0.8103499

20.0 0.9976602 NaN 0.8103499

21.0 0.9976602 NaN 0.8103499

22.0 0.9976602 NaN 0.8103499

23.0 0.9976602 NaN 0.8103499

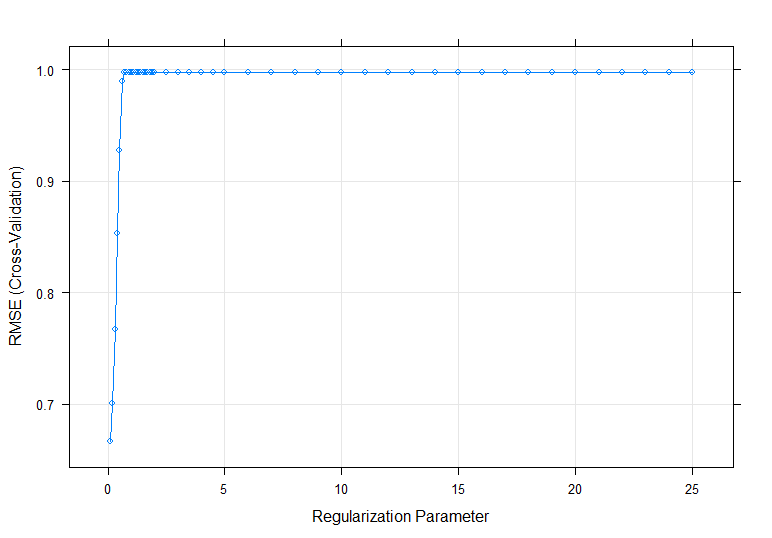
24.0 0.9976602 NaN 0.8103499

25.0 0.9976602 NaN 0.8103499

Tuning parameter 'alpha' was held constant at a value of 1

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were alpha = 1 and lambda = 0.1.



Elasticnet

176 samples

57 predictor

Pre-processing: centered (57), scaled (57)

Resampling: Cross-Validated (3 fold)

Summary of sample sizes: 117, 118, 117

Resampling results across tuning parameters:

lambda fraction RMSE Rsquared MAE

0.00 0.05 0.6658892 0.5799575 0.5430411

0.00 0.10 0.6565904 0.5606049 0.5323771

0.00 0.15 0.6717278 0.5587172 0.5328097

0.00 0.20 0.8158084 0.4669588 0.5772898

0.00 0.25 0.9510434 0.4034898 0.6200750

0.00 0.30 0.9895483 0.3889522 0.6547525

0.00 0.35 1.2710512 0.3614495 0.6912281

0.00 0.40 1.5599758 0.3489787 0.7504219

0.00 0.45 1.8599550 0.3480100 0.8018684

0.00 0.50 2.1236775 0.3432779 0.8492517

0.00 0.55 2.4146965 0.3275157 0.9156961

0.00 0.60 2.7376987 0.3170699 0.9753734

0.00 0.65 2.9350420 0.3073740 1.0182078

0.00 0.70 2.8216549 0.2985727 1.0157905

0.00 0.75 2.7363136 0.2907736 1.0175049

0.00 0.80 2.6680980 0.2840644 1.0195943

0.00 0.85 2.6068276 0.2778983 1.0219636

0.00 0.90 2.5538853 0.2719032 1.0248305

0.00 0.95 2.5005896 0.2632838 1.0271743

0.00 1.00 2.4668600 0.2521607 1.0319121

0.01 0.05 0.8160709 0.5447766 0.6619834

0.01 0.10 0.6934907 0.5716294 0.5621964

0.01 0.15 0.6590693 0.5604584 0.5350865

0.01 0.20 0.6562641 0.5602628 0.5259108

0.01 0.25 0.6611305 0.5611508 0.5230578

0.01 0.30 0.6860247 0.5469137 0.5256586

0.01 0.35 0.7164375 0.5330687 0.5303549

0.01 0.40 0.7487078 0.5214769 0.5377615

0.01 0.45 0.7694007 0.5117853 0.5531723

0.01 0.50 0.8128456 0.4836471 0.5770094

0.01 0.55 0.9722465 0.4351846 0.6060448

0.01 0.60 1.1436606 0.4034755 0.6386843

0.01 0.65 1.3016171 0.3805663 0.6667365

0.01 0.70 1.4613001 0.3628308 0.7012198

0.01 0.75 1.6324809 0.3523149 0.7395733

0.01 0.80 1.7034639 0.3441509 0.7640138

0.01 0.85 1.7553018 0.3371891 0.7848562

0.01 0.90 1.8236473 0.3306810 0.8066407

0.01 0.95 1.8919089 0.3245286 0.8268642

0.01 1.00 1.9500774 0.3188324 0.8442820

0.10 0.05 0.8911442 0.4761410 0.7235854

0.10 0.10 0.7966795 0.5542829 0.6468888

0.10 0.15 0.7230258 0.5700694 0.5859204

0.10 0.20 0.6775483 0.5749432 0.5510946

0.10 0.25 0.6580672 0.5659480 0.5368609

0.10 0.30 0.6670094 0.5448785 0.5332671

0.10 0.35 0.6704744 0.5441187 0.5290519

0.10 0.40 0.6626837 0.5568423 0.5260914

0.10 0.45 0.6673307 0.5569706 0.5246411

0.10 0.50 0.6778321 0.5525559 0.5254407

0.10 0.55 0.6993484 0.5417206 0.5286521

0.10 0.60 0.7657766 0.5126804 0.5407714

0.10 0.65 0.8174698 0.4982658 0.5555231

0.10 0.70 0.8567976 0.4817753 0.5686540

0.10 0.75 0.8984670 0.4629065 0.5797446

0.10 0.80 0.9399659 0.4472213 0.5913588

0.10 0.85 1.0172109 0.4287119 0.6063176

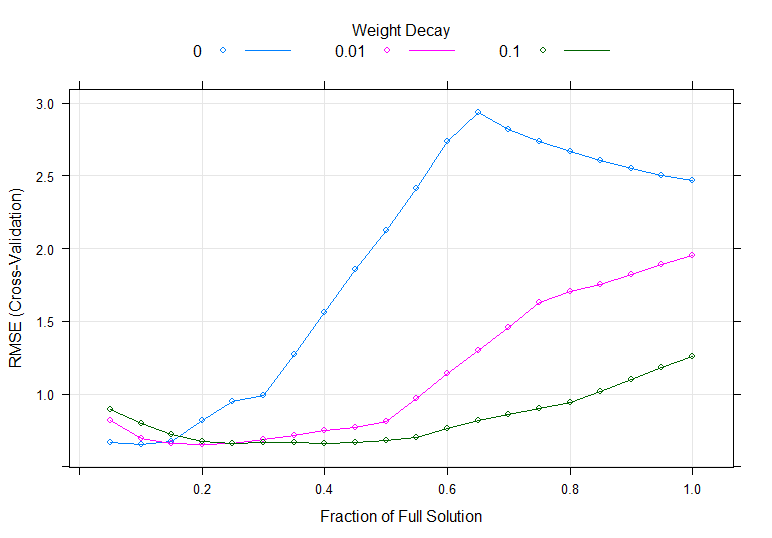
0.10 0.90 1.1023967 0.4130199 0.6218892

0.10 0.95 1.1858675 0.4002483 0.6368261

0.10 1.00 1.2598131 0.3904175 0.6504648

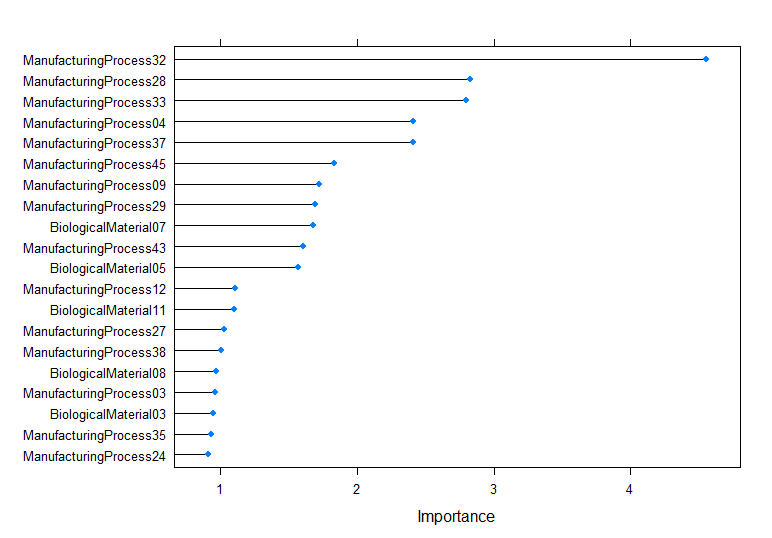
RMSE was used to select the optimal model using the smallest value.

The final values used for the model were fraction = 0.2 and lambda = 0.01.



e) The linear model has the best predictive ability due to the lowest testing RMSE and the highest testing R² value. Even though the training model may be inaccurate, when it’s applied to new data it has the highest accuracy at seed 80.

f) The top 3 most important predictors are ManufacturingProcess32, ManufacturingProcess28, and ManufacturingProcess33. The following variable importance plot shows the top 20 most important predictors:



R Code

library(AppliedPredictiveModeling)

library(caret)

library(e1071)

library(elasticnet)

library(glmnet)

library(MASS)

library(pls)

library(RANN)

library(tidyverse)

####6.1####

set.seed(80)

data(tecator)

?tecator

absorp <- as.data.frame(absorp)

endpoints <- as.data.frame(endpoints)

#absorp - absorbance data

#endpoints - percentages of water, fat, protein (cols 1-3)

pca\_absorp <- predict(preProcess(absorp, method = c("center", "scale", "pca")), absorp) #default cutoff is 95%

pca\_prePlot <- prcomp(absorp, scale = TRUE, center = TRUE)

screeplot(pca\_prePlot, type = c("lines"))

#dimensions after pca: 215 x 2

endpoints <- predict(preProcess(endpoints, method = c("center", "scale")), endpoints)

preSplit <- data.frame(PCA = pca\_absorp[,1], Fat = endpoints[,2])

trainingRows <- createDataPartition(preSplit[,1], p=0.80, list=FALSE)

training <- preSplit[trainingRows,]

testing <- preSplit[-trainingRows,]

ctrl <- trainControl(method = "cv", number = 3)

lmFit1 <- train(Fat ~ PCA, data = training, method = "lm", trControl = ctrl)

lmFit1

#an attempt to make errors go AWAY

PCA <- testing[,1]

lmPred1 <- predict(lmFit1, testing[,1])

lmValues1 <- data.frame(obs = testing$Fat, pred = lmPred1)

defaultSummary(lmValues1)

####6.2####

set.seed(80)

data(permeability)

?permeability

fingerprints <- as.data.frame(fingerprints)

#fingerprints - matrix of binary fingerprint indicator variables

#permeability - permeability values for each compound

remove <- nearZeroVar(fingerprints)

fingers <- fingerprints[,-remove]

#388 predictors left for modeling

fingers$permeability <- permeability

trainingRows <- createDataPartition(fingers[,1], p=0.80, list=FALSE)

training <- fingers[trainingRows,]

testing <- fingers[-trainingRows,]

ctrl <- trainControl(method = "cv", number = 10)

plsTune <- train(permeability ~ ., data = training, method = "pls", tuneLength = 20,

trControl = ctrl, preProc = c("center", "scale"))

plsTune

plot(plsTune)

xTest <- testing[,1:388]

predicted <- predict(plsTune, xTest)

lmValues2 <- data.frame(obs = testing[,389], pred = predicted)

colnames(lmValues2) <- c("obs", "pred")

defaultSummary(lmValues2)

####6.3####

set.seed(80)

data(ChemicalManufacturingProcess)

?ChemicalManufacturingProcess

#yield is outcome

#seeing missing values

image(is.na(ChemicalManufacturingProcess), main = "Missing Values", xlab = "Observation", ylab = "Variable", xaxt = "n", yaxt = "n", bty = "n")

axis(1, seq(0, 1, length.out = nrow(ChemicalManufacturingProcess)), 1:nrow(ChemicalManufacturingProcess), col = "white")

#imputing missing values

imputedManufacturing <- predict(preProcess(ChemicalManufacturingProcess, "knnImpute"), ChemicalManufacturingProcess)

image(is.na(imputedManufacturing), main = "Missing Values", xlab = "Observation", ylab = "Variable", xaxt = "n", yaxt = "n", bty = "n")

axis(1, seq(0, 1, length.out = nrow(imputedManufacturing)), 1:nrow(imputedManufacturing), col = "white")

#splitting data

trainingRows <- createDataPartition(imputedManufacturing[,1], p=0.80, list=FALSE)

training <- imputedManufacturing[trainingRows,]

testing <- imputedManufacturing[-trainingRows,]

ctrl <- trainControl(method = "cv", number = 3)

xTest <- imputedManufacturing[,2:58]

##lm##

lmFit2 <- train(Yield ~ ., data = imputedManufacturing, method = "lm", preProc = c("center", "scale"), trControl = ctrl)

lmFit2

#plot(lmFit2)

predicted <- predict(lmFit2, xTest)

lmValues2 <- data.frame(obs = imputedManufacturing[,1], pred = predicted)

defaultSummary(lmValues2)

##Ridge##

ridgeGrid <- data.frame(.lambda = seq(0, 1, length = 15))

ridgeRegFit <- train(Yield ~ ., data = imputedManufacturing, method = "ridge", preProc = c("center", "scale"), trControl = ctrl, tuneGrid = ridgeGrid)

ridgeRegFit

plot(ridgeRegFit)

predicted <- predict(ridgeRegFit, xTest)

ridgeValues <- data.frame(obs = imputedManufacturing[,1], pred = predicted)

defaultSummary(ridgeValues)

##lasso##

lassoGrid <- expand.grid(alpha = 1, lambda = c(seq(0.1, 2, by =0.1) , seq(2, 5, 0.5) , seq(5, 25, 1)))

lassoFit <- train(Yield ~ ., data = imputedManufacturing, method = "glmnet", preProc = c("center", "scale"), trControl = ctrl, tuneGrid = lassoGrid)

lassoFit

plot(lassoFit)

predicted <- predict(lassoFit, xTest)

lassoValues <- data.frame(obs = imputedManufacturing[,1], pred = predicted)

defaultSummary(lassoValues)

##elastic net##

enetGrid <- expand.grid(.lambda = c(0, 0.01, .1), .fraction = seq(.05, 1, length = 20))

enetTune <- train(Yield ~ ., data = imputedManufacturing, method = "enet", preProc = c("center", "scale"), trControl = ctrl, tuneGrid = enetGrid)

enetTune

plot(enetTune)

predicted <- predict(enetTune, xTest)

enetValues <- data.frame(obs = imputedManufacturing[,1], pred = predicted)

defaultSummary(enetValues)

##Variable Importance Plot##

#lm has the lowest RMSE

lmImp <- varImp(lmFit2, scale = FALSE)

lmImp

plot(lmImp, top = 20)