Michigan Technological University, Computer Science

MA 5790 Combined Section - Predictive Modeling Assignment 3

Anil Silwal

November 2, 2018

Problem 6.1

(a) Start R and use these commands to load the data

(b) In this example the predictors are the measurements at the individual frequencies. Because the frequencies lie in a systematic order (850√ˇ¨1, 050nm), the predictors have a high degree of correlation. Hence, the data lie in a smaller dimension than the total number of predictors (100). Use PCA to determine the effective dimension of these data. What is the effective dimension.

(c) Split the data into a training and a test set, pre-process the data, and build each variety of models described in this chapter. For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

(d) Which model has the best predictive ability? Is any model significantly better or worse than the others?

(e) Explain which model you would use for predicting the fat content of a sample.

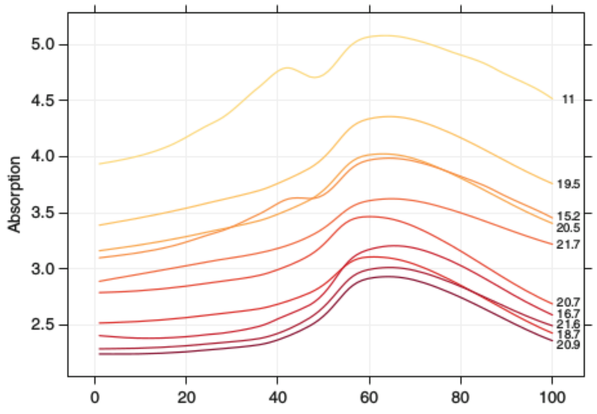


Fig. 6.20: A sample of ten spectra of the Tecator data. The colors of the curves reflect the absorption values, where *yellow* indicates low absorption and *red* is indicative of high absorption

Solution 6.1(b):

Applying the PCA, the results shows that the first component which has PCA value = 98.6261, contains almost all information. This means the true dimensionality is much lower than number of predictors.
In principal component analysis(PCA), we often use the scree plot to figure out how many important components to include in the model. Plotting the screeplot using "screeplot()" function, the elbow comes at the 2nd component in our data and the first two components in our model would be best to retained. **Thus, the first 2 component would be the effective dimension**.

However, this is based on linear combination of data; the other non-linear summarization of data may be also useful.

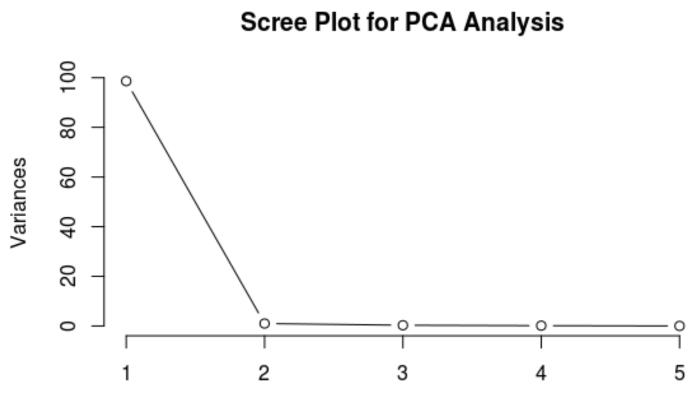


Figure 1: Scree Plot showing important Components

Solution 6.1(c, d, e):

Given, we have 3 endpoints - water, protein and fat. And as per the given question, the predictive relationship between IR spectrum and fat content is required to provide cost saving as analytical chemistry is a more expensive, time-consuming process.

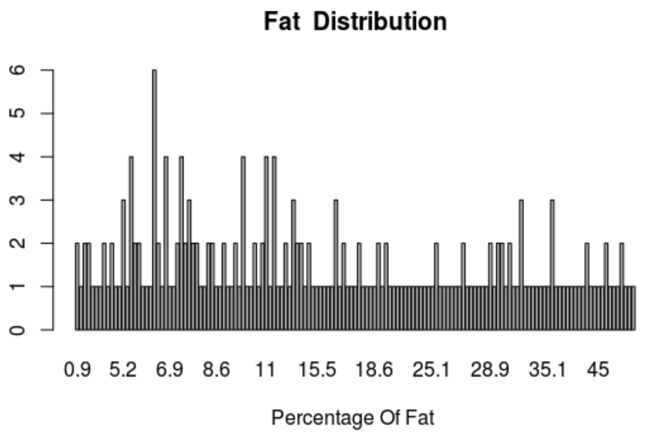
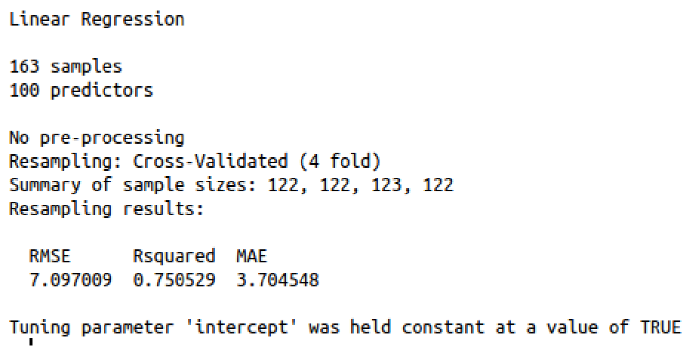


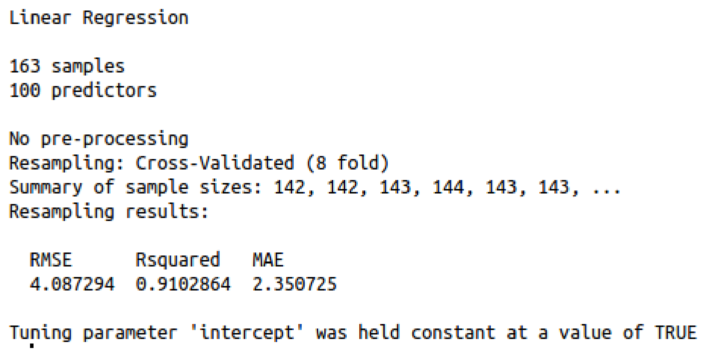
Figure 2: Response Predictor Distribution i.e Fat

Given the sample size, we split the data into 80% and 20% and repeat = 4 10-fold cross validation to tune our model. For resampling, I tried k=4 , k=8 and k=10. At last the 10 fold cross-validation resampling gives better RMSE value and I choose the 10 folded resampling and repeated 4 times in a given simpler linear regression method.



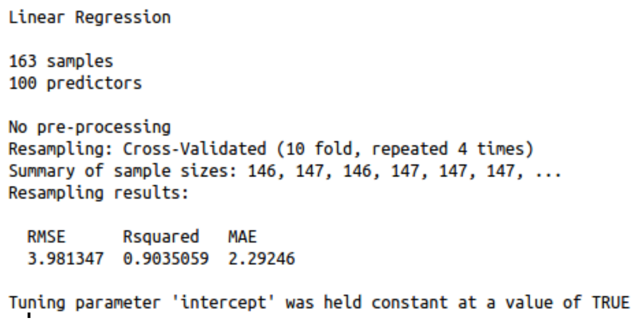
```
Linear Regression

163 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 122, 122, 123, 122
Resampling results:

  RMSE      Rsquared  MAE
  7.097009  0.750529  3.704548

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

Figure 3: Resampling for k = 4



```
Linear Regression

163 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (8 fold)
Summary of sample sizes: 142, 142, 143, 144, 143, 143, ...
Resampling results:

  RMSE      Rsquared   MAE
  4.087294  0.9102864  2.350725

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

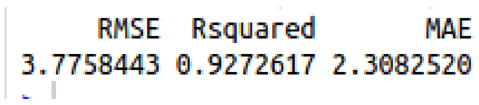
Figure 4: Resampling for k = 8



```
Linear Regression

163 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 146, 147, 146, 147, 147, 147, ...
Resampling results:

  RMSE      Rsquared   MAE
  3.981347  0.9035059  2.29246

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

Figure 5: Resampling for k = 10

The prediction accuracy based on this simple linear regression model is:



```
     RMSE  Rsquared       MAE
3.7758443 0.9272617 2.3082520
```

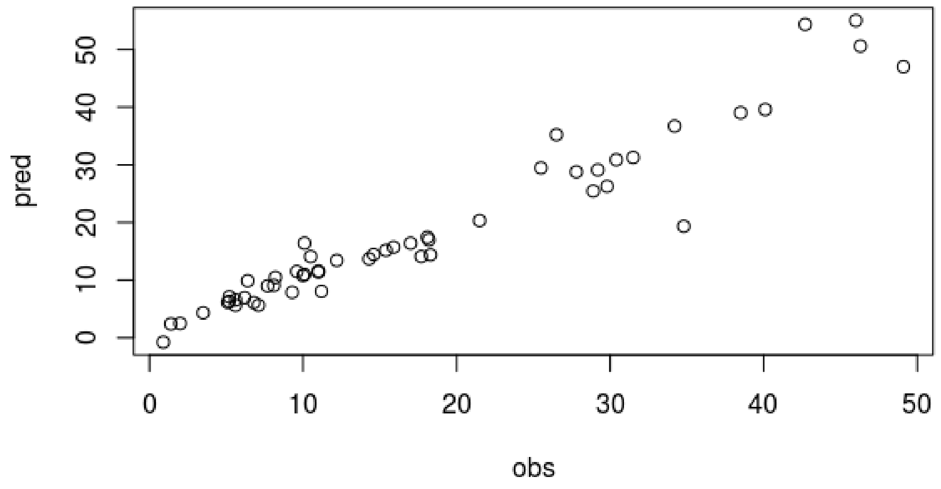


Figure 6: Simple Linear Regression Model - Prediction's Accuracy

After that, I applied PCR and PLS method to the given data:

```
Principal Component Analysis

163 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 146, 146, 146, 147, 146, 147, ...
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
   1     11.285269  0.2375411  9.175386
   2     11.268372  0.2585440  9.035455
   3      8.495126  0.5685672  6.658762
   4      4.298917  0.8981045  3.559623
   5      3.388837  0.9464940  2.695047
   6      3.099499  0.9546983  2.460895
   7      3.090544  0.9545041  2.439653
   8      3.087446  0.9566478  2.399654
   9      2.964698  0.9587683  2.345524
  10      2.978828  0.9580843  2.283250
  11      2.757382  0.9638273  2.196121
  12      2.764683  0.9633439  2.205798
  13      2.805820  0.9622680  2.230794
  14      2.871945  0.9609657  2.263338
  15      2.820530  0.9631437  2.188388
  16      2.610498  0.9678740  1.979684
  17      2.586934  0.9678644  1.955452
  18      2.528137  0.9685569  1.930185
  19      2.597241  0.9655178  1.949137
  20      2.712400  0.9608449  1.999735
  21      2.643882  0.9622737  1.910566
  22      2.481492  0.9667971  1.819169
  23      2.518659  0.9658297  1.834688
  24      2.568508  0.9643670  1.848284

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 22.
```

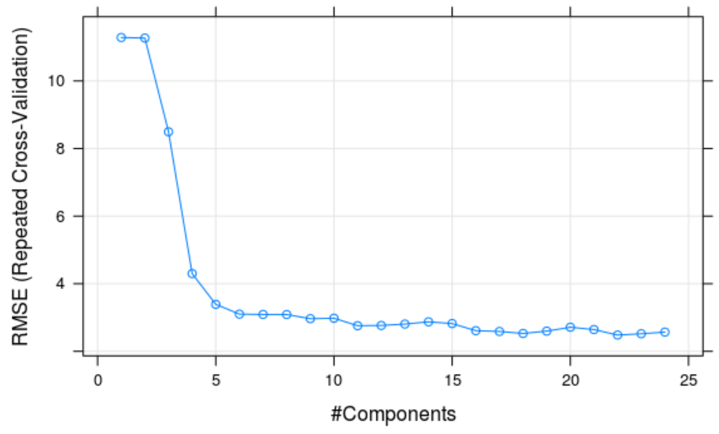
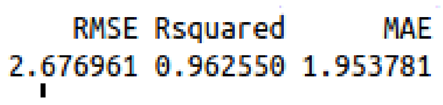


Figure 7: PCR Model

The prediction accuracy based on this model is:



```
    RMSE Rsquared      MAE
2.676961 0.962550 1.953781
```

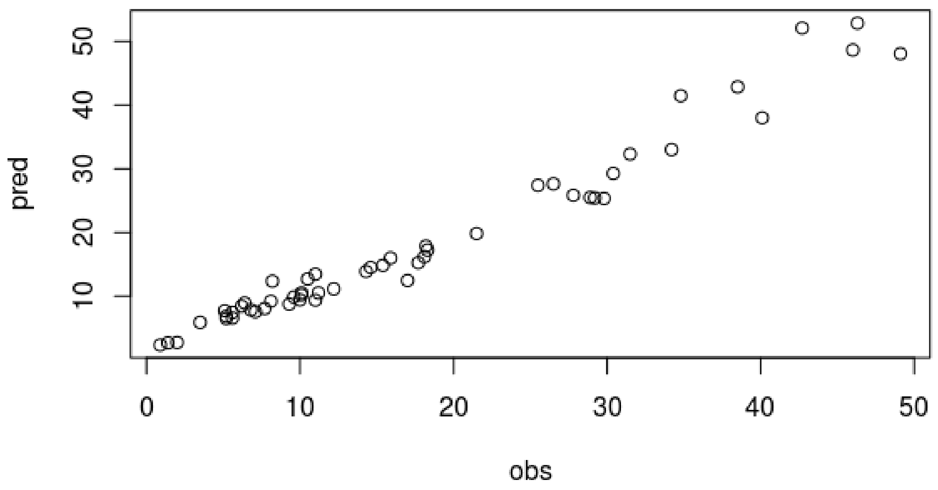


Figure 8: PCA Model - Prediction’s Accuracy

```
Partial Least Squares

163 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 146, 147, 146, 146, 148, 147, ...
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
   1     11.210142  0.2533495  9.135502
   2      7.061965  0.6969594  5.450560
   3      5.219216  0.8242954  4.039968
   4      4.142995  0.9002930  3.431444
   5      3.135929  0.9501175  2.422840
   6      3.045431  0.9519150  2.401466
   7      3.003821  0.9544468  2.316960
   8      2.925475  0.9564938  2.240443
   9      2.823914  0.9567421  2.216611
  10      2.737971  0.9592964  2.156776
  11      2.699708  0.9608291  2.164580
  12      2.603303  0.9637960  1.981274
  13      2.379695  0.9702258  1.809272
  14      2.383486  0.9694343  1.772381
  15      2.606049  0.9614942  1.880854
  16      2.772072  0.9527932  1.891770
  17      2.666571  0.9569108  1.840283
  18      2.559881  0.9585328  1.760982
  19      2.488756  0.9596306  1.707179
  20      2.607386  0.9526803  1.731890
  21      2.878102  0.9348846  1.796906
  22      3.088140  0.9160315  1.827737
  23      3.109935  0.9120810  1.831501
  24      3.087757  0.9125988  1.835312

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 13.
```

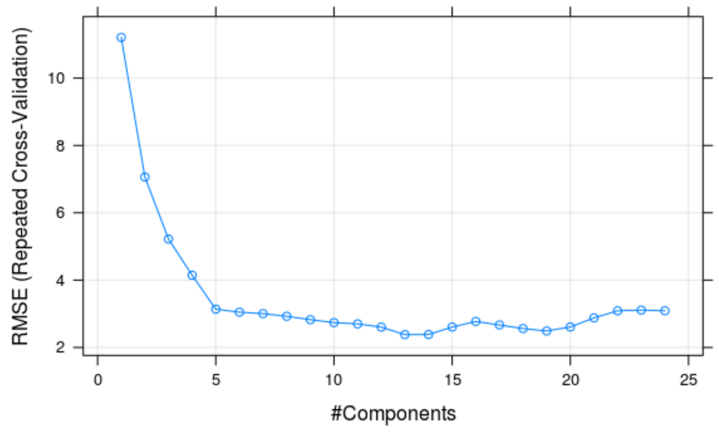


Figure 9: PLS Model

The prediction accuracy based on this model is:

```
    RMSE Rsquared      MAE
2.395839 0.966689 1.826058
```

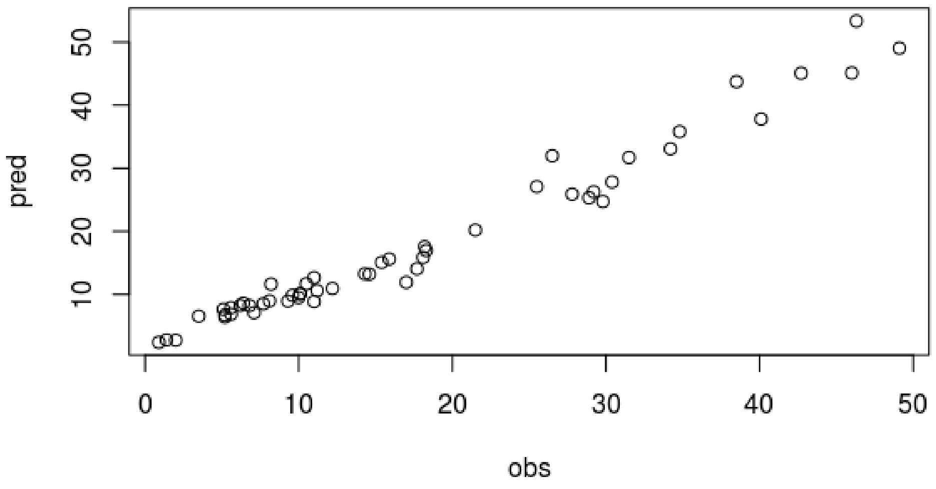


Figure 10: PLS Model - Prediction's Accuracy

```
Ridge Regression

163 samples
100 predictors

Pre-processing: centered (100), scaled (100)
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 146, 147, 146, 146, 147, 147, ...
Resampling results across tuning parameters:

  lambda      RMSE      Rsquared   MAE
  0.00000000  3.625519  0.9229186  2.276718
  0.07142857  4.802791  0.8742580  3.793913
  0.14285714  5.586271  0.8308913  4.349821
  0.21428571  6.171174  0.7887722  4.728845
  0.28571429  6.635387  0.7493139  5.040823
  0.35714286  7.024822  0.7131838  5.302266
  0.42857143  7.366853  0.6805018  5.535095
  0.50000000  7.678371  0.6511182  5.751513
  0.57142857  7.970192  0.6247635  5.956757
  0.64285714  8.249463  0.6011284  6.157070
  0.71428571  8.521023  0.5799047  6.363167
  0.78571429  8.788208  0.5608050  6.577567
  0.85714286  9.053341  0.5435702  6.797944
  0.92857143  9.318055  0.5279717  7.021094
  1.00000000  9.583493  0.5138100  7.252147

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.
```

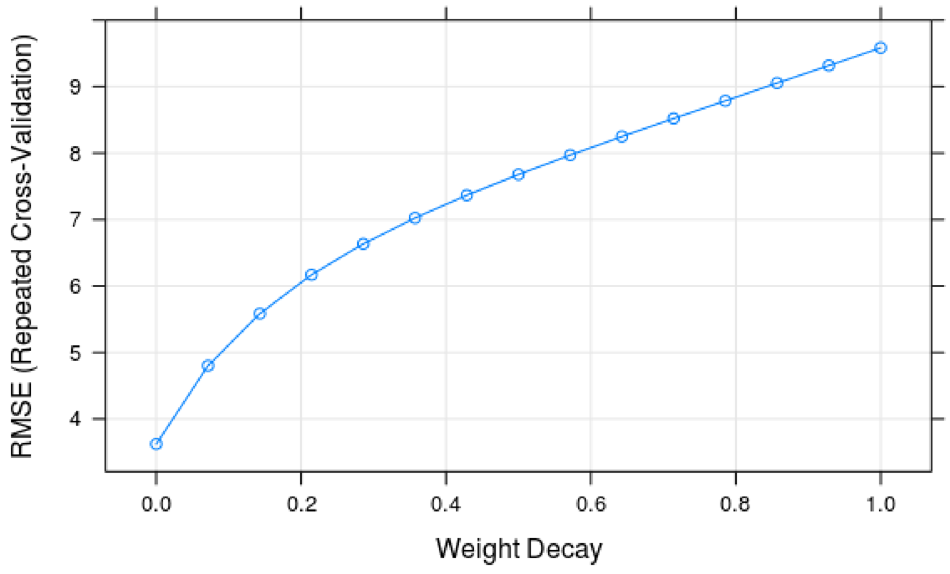
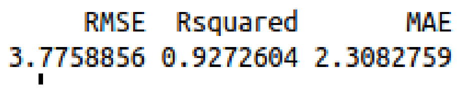


Figure 11: Ridge Regression - Model

The prediction accuracy based on this model is:



```
     RMSE  Rsquared       MAE
3.7758856 0.9272604 2.3082759
```

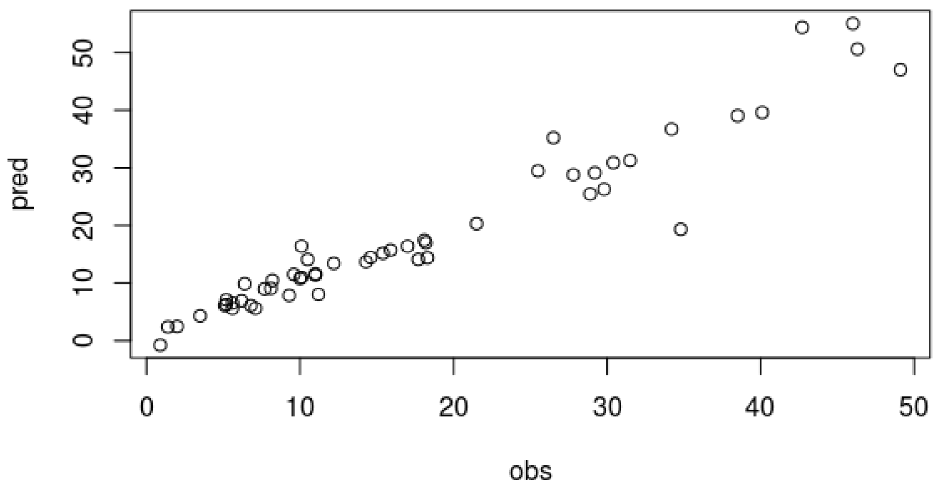


Figure 12: Ridge Regression - Prediction’s Accuracy

```
The lasso

163 samples
100 predictors

Pre-processing: centered (100), scaled (100)
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 147, 146, 147, 146, 146, 147, ...
Resampling results across tuning parameters:

  fraction   RMSE      Rsquared   MAE
  0.1000000  2.557048  0.9586631  1.663415
  0.1473684  2.536936  0.9569477  1.620058
  0.1947368  2.489033  0.9581858  1.611681
  0.2421053  2.504299  0.9565958  1.612422
  0.2894737  2.588675  0.9537632  1.643274
  0.3368421  2.665490  0.9511100  1.676271
  0.3842105  2.754083  0.9477797  1.714502
  0.4315789  2.843932  0.9442388  1.765119
  0.4789474  2.940890  0.9402209  1.818507
  0.5263158  3.042442  0.9359181  1.877187
  0.5736842  3.161280  0.9307773  1.945045
  0.6210526  3.294699  0.9246004  2.010607
  0.6684211  3.430530  0.9185784  2.078964
  0.7157895  3.554727  0.9133475  2.142798
  0.7631579  3.689140  0.9077066  2.208040
  0.8105263  3.826560  0.9020930  2.278452
  0.8578947  3.961111  0.8968452  2.349657
  0.9052632  4.096749  0.8916446  2.419650
  0.9526316  4.238345  0.8863030  2.491619
  1.0000000  4.375948  0.8811980  2.562891

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.1947368.
```

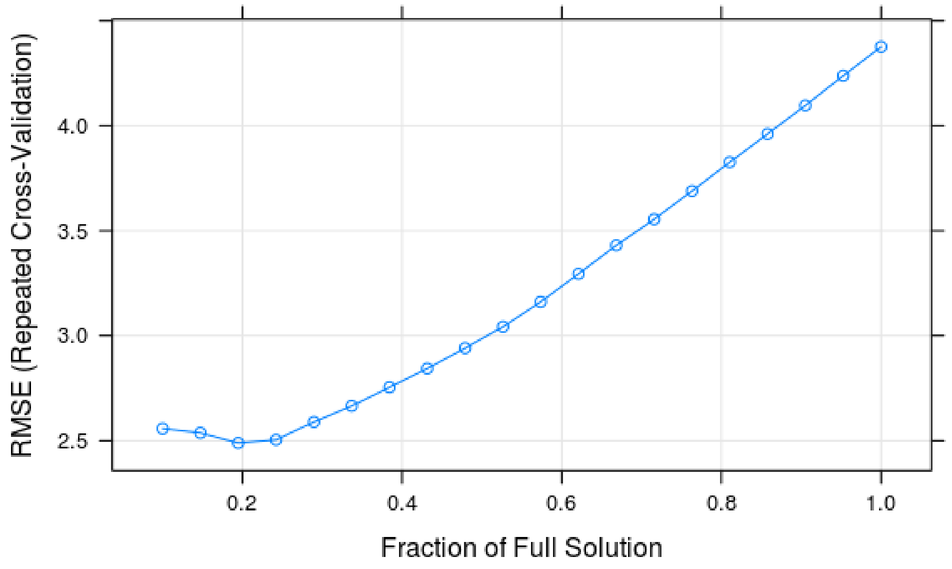


Figure 13: Lasso - Model

The prediction accuracy based on this model is:

```
     RMSE  Rsquared       MAE
3.7779787 0.9205828 2.0205018
```

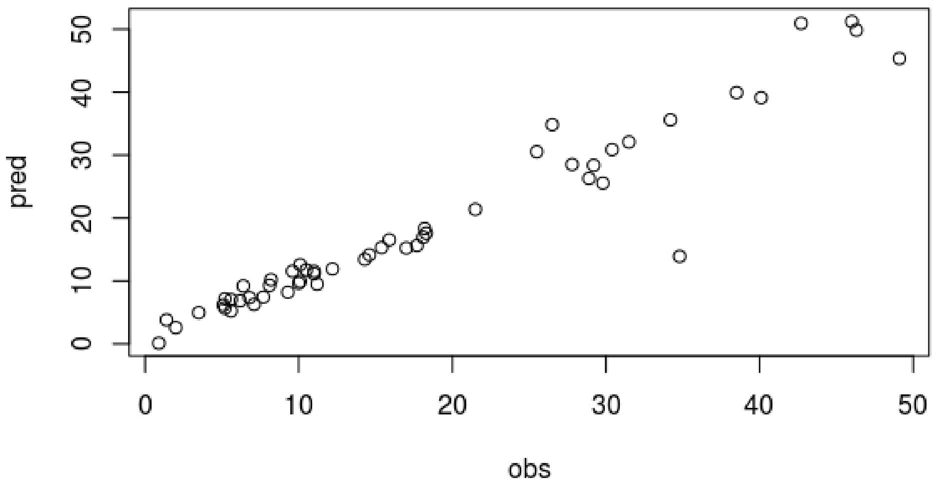
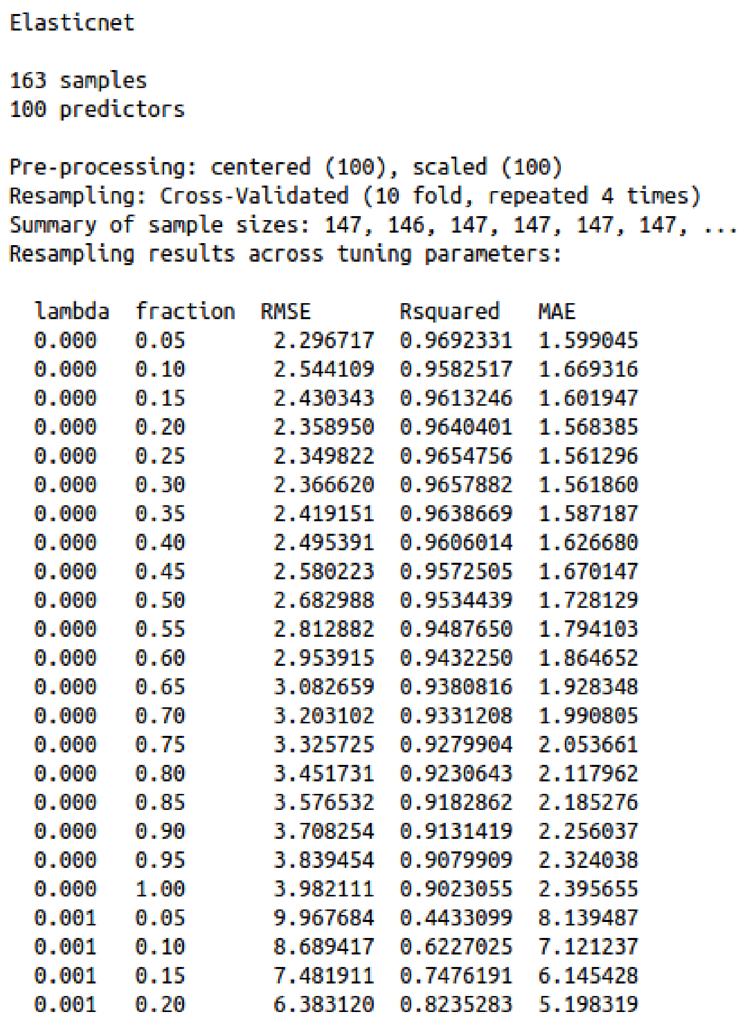
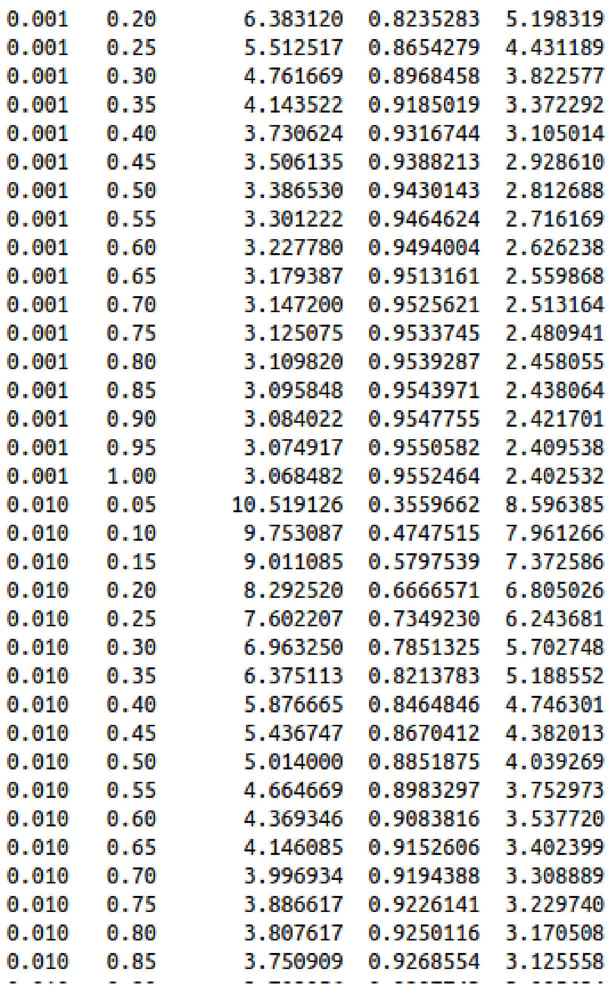


Figure 14: Lasso Model - Prediction’s Accuracy



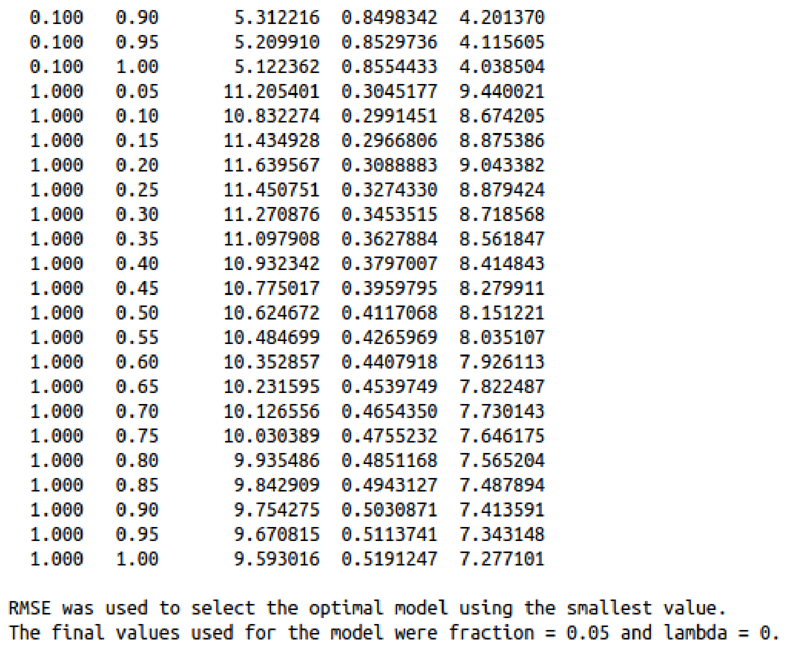
```
Elasticnet

163 samples
100 predictors

Pre-processing: centered (100), scaled (100)
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 147, 146, 147, 147, 147, 147, ...
Resampling results across tuning parameters:

  lambda  fraction  RMSE      Rsquared   MAE
  0.000   0.05       2.296717  0.9692331  1.599045
  0.000   0.10       2.544109  0.9582517  1.669316
  0.000   0.15       2.430343  0.9613246  1.601947
  0.000   0.20       2.358950  0.9640401  1.568385
  0.000   0.25       2.349822  0.9654756  1.561296
  0.000   0.30       2.366620  0.9657882  1.561860
  0.000   0.35       2.419151  0.9638669  1.587187
  0.000   0.40       2.495391  0.9606014  1.626680
  0.000   0.45       2.580223  0.9572505  1.670147
  0.000   0.50       2.682988  0.9534439  1.728129
  0.000   0.55       2.812882  0.9487650  1.794103
  0.000   0.60       2.953915  0.9432250  1.864652
  0.000   0.65       3.082659  0.9380816  1.928348
  0.000   0.70       3.203102  0.9331208  1.990805
  0.000   0.75       3.325725  0.9279904  2.053661
  0.000   0.80       3.451731  0.9230643  2.117962
  0.000   0.85       3.576532  0.9182862  2.185276
  0.000   0.90       3.708254  0.9131419  2.256037
  0.000   0.95       3.839454  0.9079909  2.324038
  0.000   1.00       3.982111  0.9023055  2.395655
  0.001   0.05       9.967684  0.4433099  8.139487
  0.001   0.10       8.689417  0.6227025  7.121237
  0.001   0.15       7.481911  0.7476191  6.145428
  0.001   0.20       6.383120  0.8235283  5.198319
```

Figure 15: Elastic Net Model



```
0.001   0.20       6.383120  0.8235283  5.198319
0.001   0.25       5.512517  0.8654279  4.431189
0.001   0.30       4.761669  0.8968458  3.822577
0.001   0.35       4.143522  0.9185019  3.372292
0.001   0.40       3.730624  0.9316744  3.105014
0.001   0.45       3.506135  0.9388213  2.928610
0.001   0.50       3.386530  0.9430143  2.812688
0.001   0.55       3.301222  0.9464624  2.716169
0.001   0.60       3.227780  0.9494004  2.626238
0.001   0.65       3.179387  0.9513161  2.559868
0.001   0.70       3.147200  0.9525621  2.513164
0.001   0.75       3.125075  0.9533745  2.480941
0.001   0.80       3.109820  0.9539287  2.458055
0.001   0.85       3.095848  0.9543971  2.438064
0.001   0.90       3.084022  0.9547755  2.421701
0.001   0.95       3.074917  0.9550582  2.409538
0.001   1.00       3.068482  0.9552464  2.402532
0.010   0.05      10.519126  0.3559662  8.596385
0.010   0.10       9.753087  0.4747515  7.961266
0.010   0.15       9.011085  0.5797539  7.372586
0.010   0.20       8.292520  0.6666571  6.805026
0.010   0.25       7.602207  0.7349230  6.243681
0.010   0.30       6.963250  0.7851325  5.702748
0.010   0.35       6.375113  0.8213783  5.188552
0.010   0.40       5.876665  0.8464846  4.746301
0.010   0.45       5.436747  0.8670412  4.382013
0.010   0.50       5.014000  0.8851875  4.039269
0.010   0.55       4.664669  0.8983297  3.752973
0.010   0.60       4.369346  0.9083816  3.537720
0.010   0.65       4.146085  0.9152606  3.402399
0.010   0.70       3.996934  0.9194388  3.308889
0.010   0.75       3.886617  0.9226141  3.229740
0.010   0.80       3.807617  0.9250116  3.170508
0.010   0.85       3.750909  0.9268554  3.125558
```

Figure 16: Elastic Net Model- Contd.



```
  0.100   0.90       5.312216  0.8498342  4.201370
  0.100   0.95       5.209910  0.8529736  4.115605
  0.100   1.00       5.122362  0.8554433  4.038504
  1.000   0.05      11.205401  0.3045177  9.440021
  1.000   0.10      10.832274  0.2991451  8.674205
  1.000   0.15      11.434928  0.2966806  8.875386
  1.000   0.20      11.639567  0.3088883  9.043382
  1.000   0.25      11.450751  0.3274330  8.879424
  1.000   0.30      11.270876  0.3453515  8.718568
  1.000   0.35      11.097908  0.3627884  8.561847
  1.000   0.40      10.932342  0.3797007  8.414843
  1.000   0.45      10.775017  0.3959795  8.279911
  1.000   0.50      10.624672  0.4117068  8.151221
  1.000   0.55      10.484699  0.4265969  8.035107
  1.000   0.60      10.352857  0.4407918  7.926113
  1.000   0.65      10.231595  0.4539749  7.822487
  1.000   0.70      10.126556  0.4654350  7.730143
  1.000   0.75      10.030389  0.4755232  7.646175
  1.000   0.80       9.935486  0.4851168  7.565204
  1.000   0.85       9.842909  0.4943127  7.487894
  1.000   0.90       9.754275  0.5030871  7.413591
  1.000   0.95       9.670815  0.5113741  7.343148
  1.000   1.00       9.593016  0.5191247  7.277101

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.05 and lambda = 0.
```

Figure 17: Elastic Net Model- Contd.

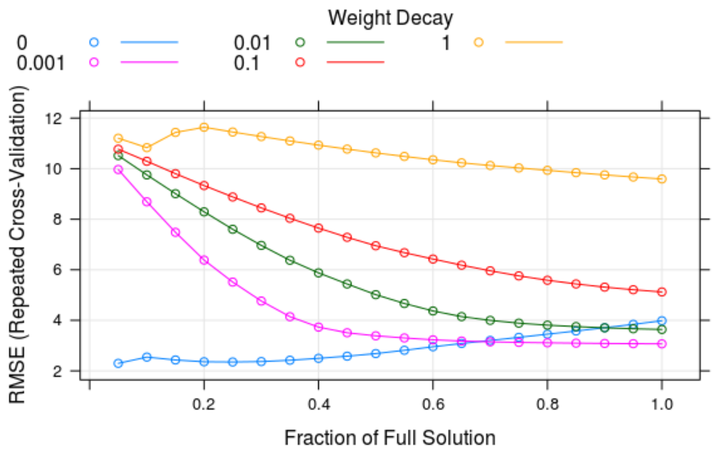
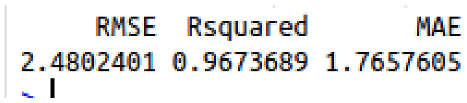


Figure 18: Elastic Net Graph

The prediction accuracy based on this model is:



```
     RMSE  Rsquared       MAE
2.4802401 0.9673689 1.7657605
```

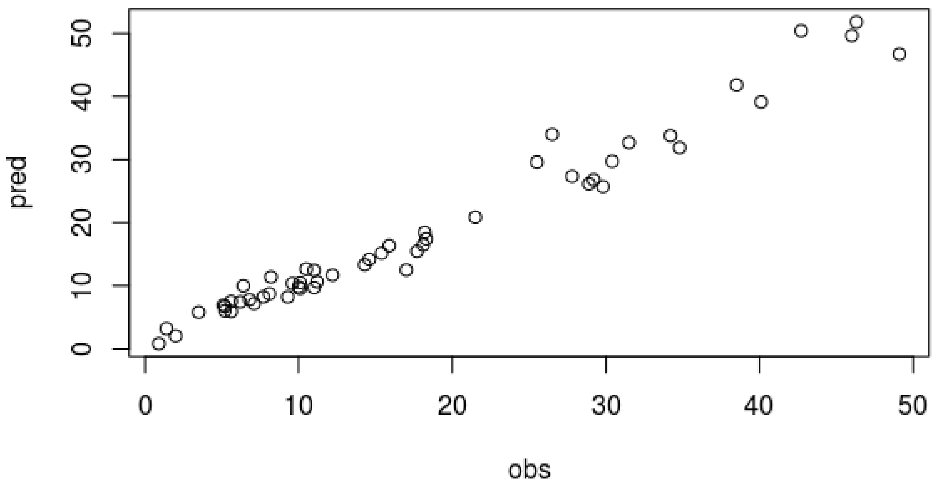


Figure 19: Elastic Net Model- Prediction

Solution 6.1(d)

Comparing the results from different models, The **PLS** has minimum RMSE which is 2.395839. The **PCA** and **Elastic Net** also has nearly similar RMSE which are 2.676961 and 2.4802401 respectively. This suggests, **PLS**, **PCA**, **Elastic Net** are better than others.
While, the **Lasso** has maximum RMSE which is 3.77797. And this Lasso Model is comparatively worst model for this dataset. Other than that, **Simple Regression** Model has also worst performance and shows, RMSE = 3.7758856 which is also the worst case in comparison to other models.

Solution 6.1(e)

Comparing the results from different models, the **PLS** also has minimum RMSE which is 2.395839. This suggests, **PLS** is most likely used model for predicting Fat in this dataset.

Comparison table for different Performance

Model	RMSE Training	R^2 Training	RMSE Testing	R^2 Testing
Simple	3.981347	0.9035059	3.7758443	0.9272617
PCA	2.481492	0.9667971	2.676961	0.962550
PLS	2.379695	0.97.2258	2.395839	0.966689
Ridge	3.625519	3.625519	3.7758856	0.9272604
Lasso	2.489033	0.9581858	3.7779787	0.9205828
Elastic Net	2.296717	0.9692331	2.4802401	0.9673689

Problem 6.2 Developing a model to predict permeability (see Sect. 1.4) 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

(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?

(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 R^2 ?

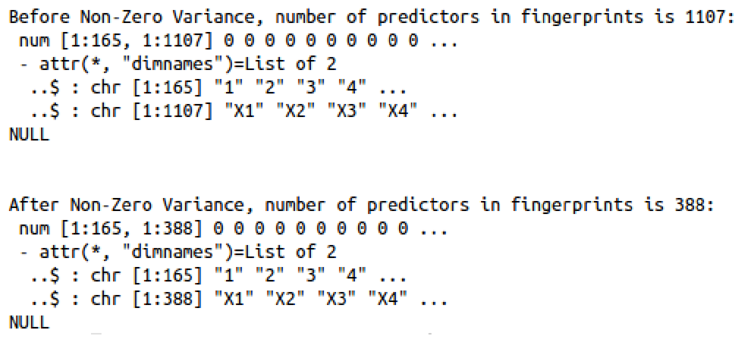
(d) Predict the response for the test set. What is the test set estimate of R^2 ?

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

(f) Would you recommend any of your models to replace the permeability laboratory experiment?

Solution 6.2(b)

Initially, we have 1107 predictors in fingerprints. Applying non-zero-variance, there were 719 predictors found having non-zero-variance fingerprints. Then after removing these predictors, only 388 predictors are left for the modeling. The fingerprints predictors are reduced to almost 25 percent comparing to original predictors.



```
Before Non-Zero Variance, number of predictors in fingerprints is 1107:
 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" ...
NULL


After Non-Zero Variance, number of predictors in fingerprints is 388:
 num [1:165, 1:388] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:165] "1" "2" "3" "4" ...
  ..$ : chr [1:388] "X1" "X2" "X3" "X4" ...
NULL
```

Figure 20: Result showing predictors before and after applying near-zero-variance() function

Solution 6.2(c)

When determining the data splitting method, we should focus on two primary characteristics:

- the number of samples relative to the number of predictors in the data, and
- the distribution of samples across classes.

After the near-zero-variance, the data sample i.e sample = 165 and predictors = 388. Still number of predictors are larger than the sample data. In such case, we use stratified random splitting and for that we use createDataPartition() function given by library(caret). For this, we used 75% as training data and 25% as testing data from sample. For PLS Tuning Parameter, the 4 fold cross validation with 5 repeats technique was used to to resample number of train datas. For PLS tuning, I take 15 components and calculate the RMSE, Rsquared and MAE values.

```
Partial Least Squares

125 samples
388 predictors

Pre-processing: centered (388), scaled (388)
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 93, 93, 96, 93, 94, 94, ...
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared   MAE
   1     13.31396  0.2325657   9.729249
   2     12.68750  0.3255819   9.073525
   3     12.70913  0.3463417   9.351738
   4     13.00558  0.3419348   9.482017
   5     12.75070  0.3749557   9.167606
   6     12.74253  0.3683852   9.258273
   7     12.61378  0.3761673   9.251668
   8     12.64845  0.3782935   9.442780
   9     12.79393  0.3776872   9.520957
  10     13.00754  0.3696524   9.661983
  11     13.13951  0.3661599   9.658102
  12     13.12691  0.3690678   9.658389
  13     13.24511  0.3637436   9.759184
  14     13.45138  0.3548575   9.843516
  15     13.67831  0.3428259  10.006368

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 7.
```

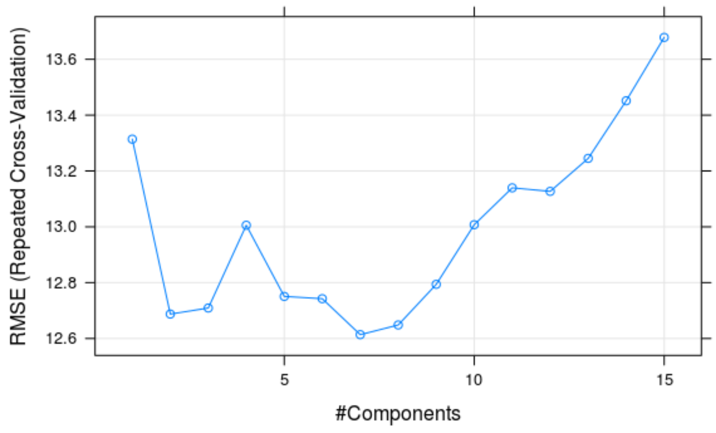
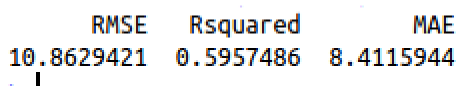


Figure 21: Result showing PLS Tuning Parameter For Permeability Data

The result shows the optimal number of latent variables that maximizes R^2 is 7th component, which is 0.3761673.

The prediction accuracy based on this model is:



```
      RMSE   Rsquared        MAE
10.8629421  0.5957486  8.4115944
```

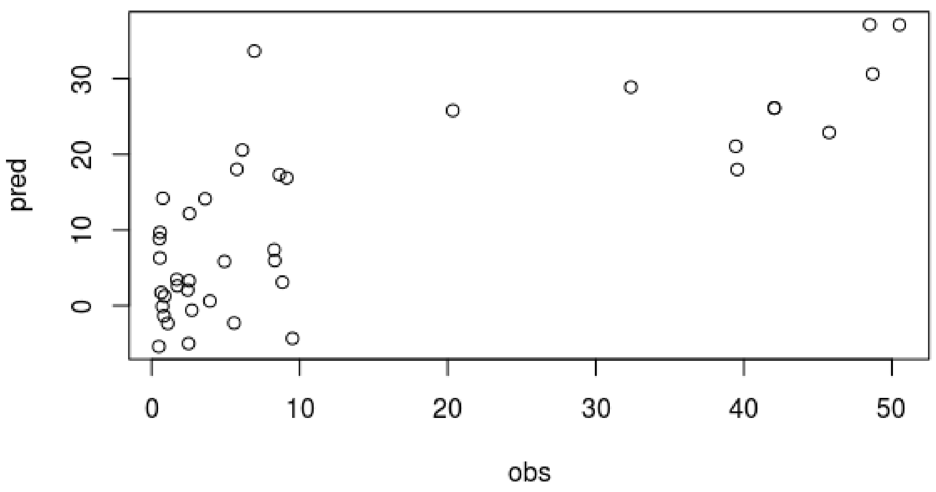


Figure 22: PLS : Prediction-Accuracy

The result shows the corresponding resampled estimate of R^2 is 0.5957486, given by PLS.

```
Ridge Regression

125 samples
388 predictors

Pre-processing: centered (388), scaled (388)
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 93, 93, 94, 95, 94, 94, ...
Resampling results across tuning parameters:

  lambda      RMSE      Rsquared   MAE
  0.08000000  13.67989  0.3494300  9.952184
  0.09214286  13.55044  0.3559901  9.863265
  0.10428571  13.45758  0.3615510  9.803490
  0.11642857  13.43308  0.3644957  9.803520
  0.12857143  13.35137  0.3694529  9.749217
  0.14071429  13.32510  0.3718913  9.739208
  0.15285714  13.30119  0.3747566  9.730755
  0.16500000  13.29086  0.3773629  9.727753
  0.17714286  13.28210  0.3794440  9.732622
  0.18928571  13.27738  0.3812073  9.744017
  0.20142857  13.27802  0.3828925  9.753622
  0.21357143  13.29302  0.3842816  9.778467
  0.22571429  13.30597  0.3855011  9.802696
  0.23785714  13.32910  0.3865729  9.837620
  0.25000000  13.35203  0.3875474  9.869504

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.1892857.
```

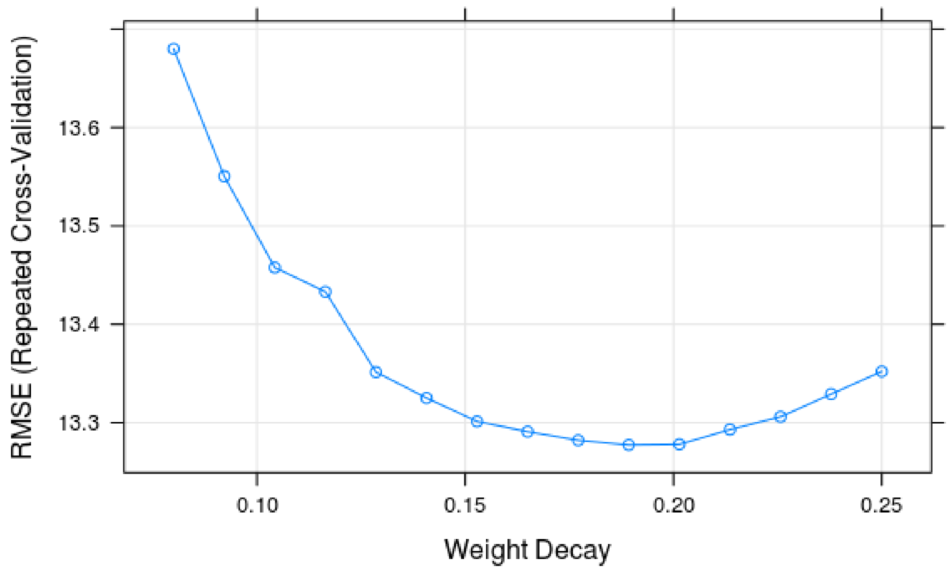
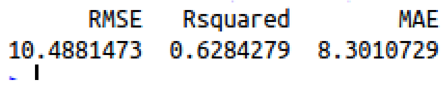


Figure 23: Ridge Regression - Model

The prediction accuracy based on this model is:



```
      RMSE   Rsquared        MAE
10.4881473  0.6284279  8.3010729
```

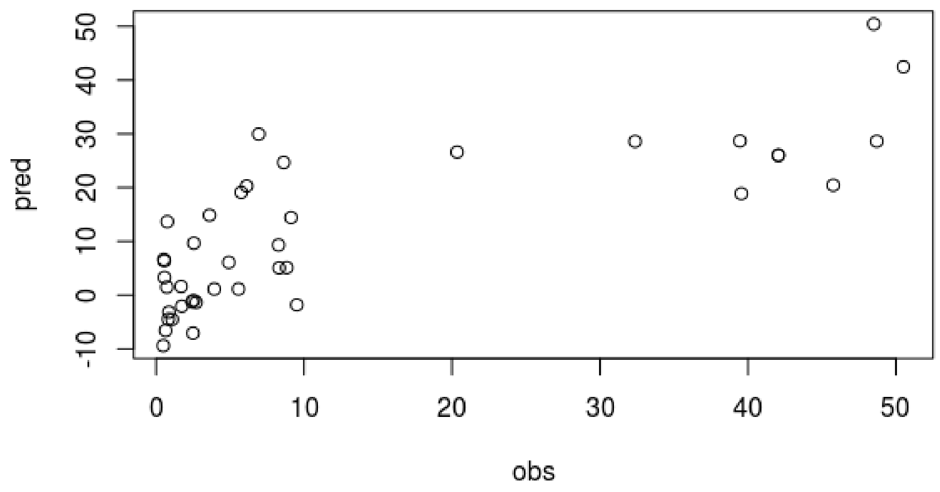
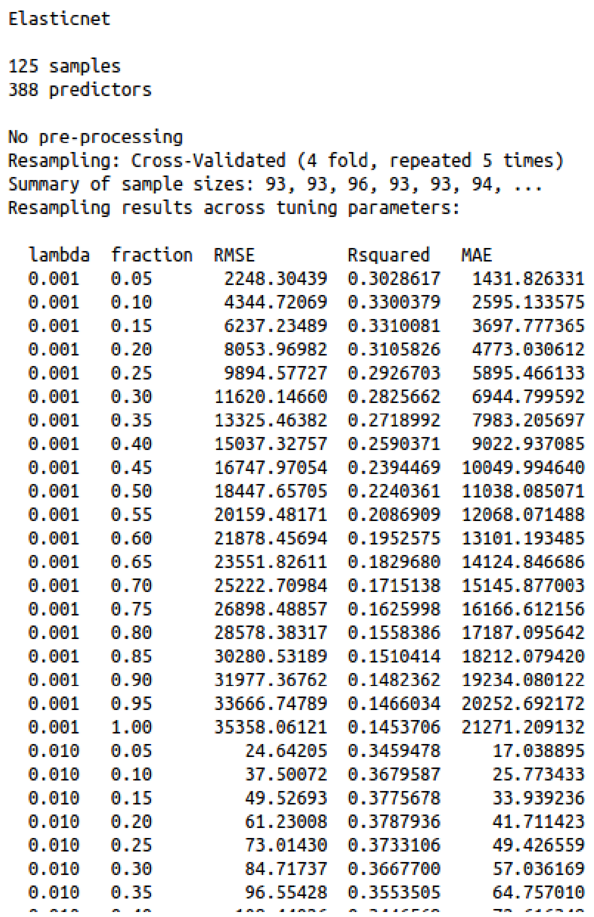
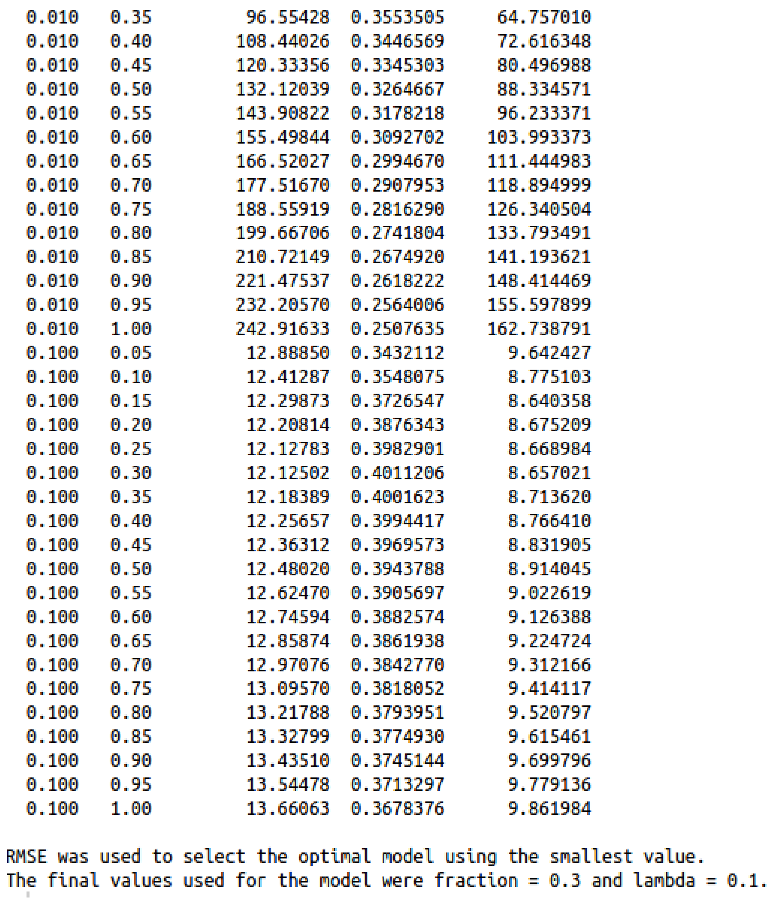


Figure 24: Ridge Regression - Prediction's Accuracy



```
Elasticnet

125 samples
388 predictors

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 93, 93, 96, 93, 93, 94, ...
Resampling results across tuning parameters:

  lambda  fraction  RMSE         Rsquared   MAE
  0.001   0.05       2248.30439  0.3028617   1431.826331
  0.001   0.10       4344.72069  0.3300379   2595.133575
  0.001   0.15       6237.23489  0.3310081   3697.777365
  0.001   0.20       8053.96982  0.3105826   4773.030612
  0.001   0.25       9894.57727  0.2926703   5895.466133
  0.001   0.30      11620.14660  0.2825662   6944.799592
  0.001   0.35      13325.46382  0.2718992   7983.205697
  0.001   0.40      15037.32757  0.2590371   9022.937085
  0.001   0.45      16747.97054  0.2394469  10049.994640
  0.001   0.50      18447.65705  0.2240361  11038.085071
  0.001   0.55      20159.48171  0.2086909  12068.071488
  0.001   0.60      21878.45694  0.1952575  13101.193485
  0.001   0.65      23551.82611  0.1829680  14124.846686
  0.001   0.70      25222.70984  0.1715138  15145.877003
  0.001   0.75      26898.48857  0.1625998  16166.612156
  0.001   0.80      28578.38317  0.1558386  17187.095642
  0.001   0.85      30280.53189  0.1510414  18212.079420
  0.001   0.90      31977.36762  0.1482362  19234.080122
  0.001   0.95      33666.74789  0.1466034  20252.692172
  0.001   1.00      35358.06121  0.1453706  21271.209132
  0.010   0.05         24.64205  0.3459478     17.038895
  0.010   0.10         37.50072  0.3679587     25.773433
  0.010   0.15         49.52693  0.3775678     33.939236
  0.010   0.20         61.23008  0.3787936     41.711423
  0.010   0.25         73.01430  0.3733106     49.426559
  0.010   0.30         84.71737  0.3667700     57.036169
  0.010   0.35         96.55428  0.3553505     64.757010
```

Figure 25: Elastic Net Model



```
  0.010   0.35         96.55428  0.3553505     64.757010
  0.010   0.40        108.44026  0.3446569     72.616348
  0.010   0.45        120.33356  0.3345303     80.496988
  0.010   0.50        132.12039  0.3264667     88.334571
  0.010   0.55        143.90822  0.3178218     96.233371
  0.010   0.60        155.49844  0.3092702    103.993373
  0.010   0.65        166.52027  0.2994670    111.444983
  0.010   0.70        177.51670  0.2907953    118.894999
  0.010   0.75        188.55919  0.2816290    126.340504
  0.010   0.80        199.66706  0.2741804    133.793491
  0.010   0.85        210.72149  0.2674920    141.193621
  0.010   0.90        221.47537  0.2618222    148.414469
  0.010   0.95        232.20570  0.2564006    155.597899
  0.010   1.00        242.91633  0.2507635    162.738791
  0.100   0.05         12.88850  0.3432112      9.642427
  0.100   0.10         12.41287  0.3548075      8.775103
  0.100   0.15         12.29873  0.3726547      8.640358
  0.100   0.20         12.20814  0.3876343      8.675209
  0.100   0.25         12.12783  0.3982901      8.668984
  0.100   0.30         12.12502  0.4011206      8.657021
  0.100   0.35         12.18389  0.4001623      8.713620
  0.100   0.40         12.25657  0.3994417      8.766410
  0.100   0.45         12.36312  0.3969573      8.831905
  0.100   0.50         12.48020  0.3943788      8.914045
  0.100   0.55         12.62470  0.3905697      9.022619
  0.100   0.60         12.74594  0.3882574      9.126388
  0.100   0.65         12.85874  0.3861938      9.224724
  0.100   0.70         12.97076  0.3842770      9.312166
  0.100   0.75         13.09570  0.3818052      9.414117
  0.100   0.80         13.21788  0.3793951      9.520797
  0.100   0.85         13.32799  0.3774930      9.615461
  0.100   0.90         13.43510  0.3745144      9.699796
  0.100   0.95         13.54478  0.3713297      9.779136
  0.100   1.00         13.66063  0.3678376      9.861984

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.3 and lambda = 0.1.
```

Figure 26: Elastic Net Model- Contd.

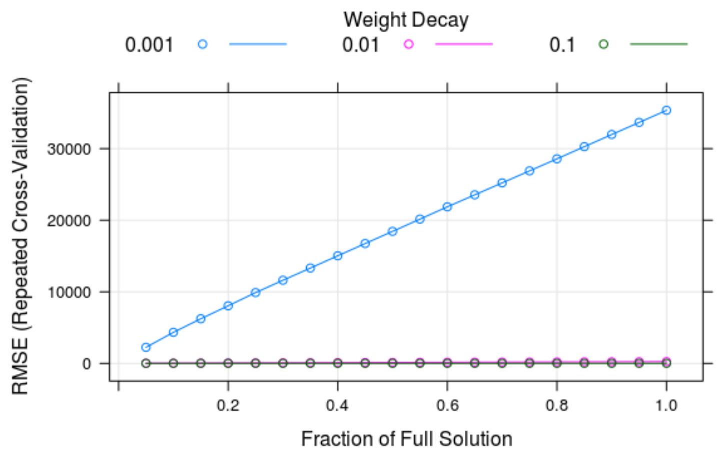


Figure 27: Elastic Net Graph

The prediction accuracy based on this model is:

```
      RMSE   Rsquared        MAE
10.8629421  0.5957486  8.4115944
```

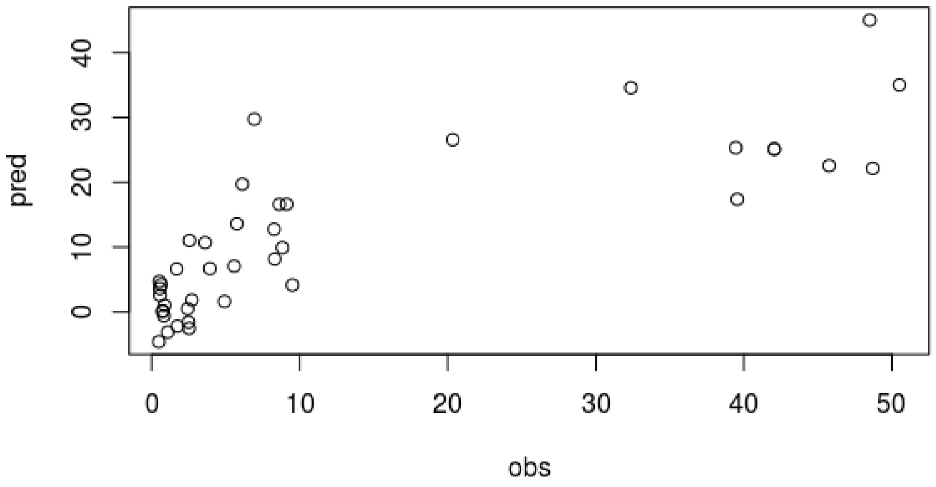


Figure 28: Elastic Net Model- Prediction

Comparing the results from different models, The **Elastic net** has minimum RMSE which is 10.0757239.

Solution 6.1(f)

Comparing the results from different models, the **Elastic net** has minimum RMSE which is 10.0757239. This suggests, **Elastic Net** is most likely used model for predicting Permeability in this dataset.

Comparison table for different Performance

Model	RMSE Training	R^2 Training	RMSE Testing	R^2 Testing
PLS	12.61378	0.3761673	10.8629421	0.5957486
Ridge	13.277328	0.3812073	10.4881743	0.6284279
Elastic Net	12.12502	0.4011406	10.0757239	0.6605789

```
library(mlbench)
library(caret)
library(e1071)
library(AppliedPredictiveModeling)
library(dplyr)
library(data.table)

##################################
#Question 6.1
#################################
library(AppliedPredictiveModeling)
library(MASS)
library(caret)
library(elasticnet)
library(lars)
library(pls)

data(tecator)

#########################
# question 6.1(b)
#########################

colName = -
for (i in 1:100)-
  colName[i]¡- paste("X",i)

colnames(absorp)¡-colName

## The base R function prcomp can be used for PCA. In the code below,
## the data are centered and scaled prior to PCA.
pcaObject ¡- prcomp(absorp, center = TRUE, scale. = TRUE)

# The standard deviations for the columns in the data are stored in pcaObject as a sub-object called ad:
# Calculate the cumulative percentage of variance which each component
# accounts for.
percentVariance ¡- pcaObject$sd 2 /sum(pcaObject$sd 2 ) *100
print(percentVariance[1:5])

# taking only 5 variances set npcs = 5
screeplot(pcaObject, npcs = 5, type = "lines", main = "Scree Plot for PCA Analysis")


#########################
#########################

#########################
# question 6.1(c,d,e)
#########################

# Response variable = Fat

# BarPlot of response variable

counts ¡- table( endpoints[,2])
barplot(counts, main="Fat  Distribution",
        xlab="Percentage Of Fat ")


# splitting data into 80% and 20% based on Fat Response
set.seed(12345)

trainingRows =  createDataPartition(endpoints[,2], p = .75, list= FALSE)

trainAbsorption ¡- absorp[ trainingRows, ]
```

```
testAbsorption ¡- absorp[-trainingRows, ]
trainFat ¡- endpoints[trainingRows, 2]
testFat ¡- endpoints[-trainingRows, 2]

ctrl ¡- trainControl(method = ”repeatedcv”, repeats=4)

cat(”“n”)

set.seed(12345)

# simple linear regression
meatln ¡- train(x = trainAbsorption , y = trainFat, method = ”lm”, trControl = ctrl)
print(meatln)


prediction¡-predict(meatln,testAbsorption)
accuracy3¡-data.frame(obs=testFat,pred=prediction)
defaultSummary(accuracy3)
plot(accuracy3)



cat(”“n”)

# PCR method
meatPCR ¡- train(x = trainAbsorption , y = trainFat, method = ”pcr”, trControl = ctrl, tuneLength = 24)
print(meatPCR)
plot(meatPCR)

cat(”“n”)
cat(”“n”)

# PLS method
meatPLS ¡- train(x = trainAbsorption , y = trainFat, method = ”pls”, trControl = ctrl, tuneLength = 24)
print(meatPLS)
plot(meatPLS)
cat(”“n”)



# Ridge Regression Method
meatRg ¡- train(x = trainAbsorption , y = trainFat, method = ”ridge”,
                 trControl = ctrl,
                 preProcess = c(”center”,”scale”),
                 tuneGrid = expand.grid(lambda = seq(0,1,length=15)))

print(meatRg)
plot(meatRg)
cat(”“n”)

prediction¡-predict(meatRg,testAbsorption)
accuracy2¡-data.frame(obs=testFat,pred=prediction)
defaultSummary(accuracy2)
plot(accuracy2)


# Lasso Regression Method
meatLasso ¡- train(x = trainAbsorption , y = trainFat, method = ”lasso”,
                trControl = ctrl,
                preProcess = c(”center”,”scale”),
                tuneGrid = expand.grid(fraction = seq(0.1,1,length=20)))

print(meatLasso)
plot(meatLasso)
cat(”“n”)
```

```
prediction¡-predict(meatLasso,testAbsorption)
accuracy3¡-data.frame(obs=testFat,pred=prediction)
defaultSummary(accuracy3)
plot(accuracy3)



# Elastic Net Method
meatEls ¡- train(x = trainAbsorption , y = trainFat, method = ”enet”,
                 trControl = ctrl,
                 preProcess = c(”center”,”scale”),
                 tuneGrid = expand.grid(lambda = c(0,.001,.01,.1,1),
                                        fraction = seq(0.05,1,length=20)))

print(meatEls)
plot(meatEls)
cat(”“n”)

prediction¡-predict(meatEls,testAbsorption)
accuracy1¡-data.frame(obs=testFat,pred=prediction)
defaultSummary(accuracy1)
plot(accuracy1)


##################################
#Question 6.2
##################################

library(AppliedPredictiveModeling)
library(MASS)
library(caret)
library(elasticnet)
library(lars)
library(pls)

data(permeability)

#########################
# question 6.2(b)
#########################
cat(”Before Non-Zero Variance, number of predictors in fingerprints is 1107: “n”)
print(str(fingerprints))
cat(”“n“n”)

cat(”After Non-Zero Variance, number of predictors in fingerprints is 388: “n”)
NZVfingerprints ¡- nearZeroVar(fingerprints)
noNZVfingerprints ¡- fingerprints[,-NZVfingerprints]
print(str(noNZVfingerprints))
cat(”“n“n”)

#########################
#########################

#########################
# question 6.2(c)
#########################

# stratified random sample splitting with 75% training and 25% testing

set.seed(12345)
trainingRows =  createDataPartition(permeability, p = .75, list= FALSE)

trainFingerprints ¡- noNZVfingerprints[trainingRows,]
```

```
trainPermeability ¡- permeability[trainingRows,]

testFingerprints ¡- noNZVfingerprints[-trainingRows,]
testPermeability ¡- permeability[-trainingRows,]

set.seed(12345)

ctrl ¡- trainControl(method = ”repeatedcv”, repeats=5, number = 4)


# PLS Model
permeabiltyPLS ¡- train(x = trainFingerprints , y = trainPermeability, method = ”pls”, tuneGrid = expand.gri
print(permeabiltyPLS)
plot(permeabiltyPLS, metric =”Rsquared”, main = ”PLS Tuning Parameter for Permeability Data”)

# PLS prediction
prediction¡-predict(permeabiltyPLS,testFingerprints)
accuracy ¡- data.frame(obs=testPermeability, pred=prediction)
defaultSummary(accuracy)
plot(accuracy)

cat(””n”)

# # Ridge Regression Method
permeabiltyRg ¡- train(x = trainFingerprints , y = trainPermeability, method = ”ridge”,
                trContrrol = ctrl,
                tuneGrid = expand.grid(lambda = seq(0.08,0.25,length=15)))

prediction¡-predict(permeabiltyRg,testFingerprints)
accuracy ¡- data.frame(obs=testPermeability, pred=prediction)
defaultSummary(accuracy)
plot(accuracy)

cat(””n”)

#
# # Elastic Net Method
PermeabilityEls ¡- train(x = trainFingerprints , y = trainPermeability, method = ”enet”,
                 trControl = ctrl,
                 tuneGrid = expand.grid(lambda = c(.001,.01,.1),
                                        fraction = seq(0.05,1,length=20)))

print(meatEls)
plot(meatEls)
cat(””n”)

# # Elastic Net Method Prediction
prediction¡-predict(PermeabilityEls,testFingerprints)
accuracy ¡- data.frame(obs=testPermeability, pred=prediction)
defaultSummary(accuracy)
plot(accuracy)

cat(””n”)


# Lasso Regression Method
permeabilityLasso ¡- train(x = trainFingerprints , y = trainPermeability, method = ”lasso”,
                           trControl = ctrl,
                           tuneGrid = expand.grid(fraction = seq(0.0005,0.005,length=20)))


print(PermeabilityLasso)
plot(PermeabilityLasso)
cat(””n”)
#
```

```
# # Lasso Model Prediction
prediction¡-predict(PermeabilityLasso,testFingerprints)
accuracy ¡- data.frame(obs=testPermeability, pred=prediction)
defaultSummary(accuracy)
plot(accuracy)

#
# #########################
# #########################
#
```

References:

1. Applied Predictive Modeling : @authors Max Kuhn. Kjell Johnson
2. https://archive.ics.uci.edu/ml/index.php