**DSCI 425 – Supervised Learning (62 pts.)**

**Assignment 3 – Neural Networks for Regression**

**predicting strength of concrete**

Concrete is the most important material in civil engineering. The concrete compressive strength is thought to be a highly nonlinear function of age and ingredients.

**Variable Information:**  
Given below are the variables contained the file **Concrete.csv** on course website. These data come from a collection of 17 experiments where the compressive strength (MPa) of concrete was determined under different formulations and length of curing (days). These data consist of n = 1030 observations on nine variables (8 predictors and 1 response). There are no cases with missing values!  
  
Name / Data Type / Description/Measurement Units (red denotes variable has zeroes)

* Cement () - continuous – kg of cement per cubic meter of concrete
* Blast Furnace Slag () - continuous – kg of slag per cubic meter of concrete
* Fly Ash () - continuous -- kg of fly ash per cubic meter of concrete
* Water () - continuous -- kg of water per cubic meter of concrete
* Superplasticizer () - continuous -- kg of superplasticizer per cubic meter of concrete
* Coarse Aggregate () - continuous -- kg of course aggregate per cubic meter of concrete
* Fine Aggregate () - continuous -- kg of fine aggregate per cubic meter of concrete
* Age - discrete – age of concrete measured in days (1-365)
* Concrete compressive strength - continuous – compressive strength in *megapascals* ()

Data source: I-Cheng Yeh, "*Modeling of strength of high performance concrete using artificial neural networks*," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998)

1. Develop a neural network for these data using the nnet package in R. Use some form of cross-validation to choose an “optimal” neural network model fit to these data. **Explain you model development process including supporting R code/results.** Include a plot the predicted and actual values from your neural network model, both in the transformed and untransformed scales, assuming you used a transformation of the compressive strength of the concrete. (15 pts.)

To start, we will perform all of the transformations on this dataset that were done for assignment 2, as our feeling regarding such actions have not changed. Such changes are shown below, but our discussion of the process will be left out for the sake of conciseness. We also are using the same seed for our sampling as the previous assignment and keeping our R-versions consistent, so that validation and training sets will line up properly.

{

#Create lm objects and update the model

Concrete.trans = Concrete

set.seed(1)

sam = sample(1:1030, size = floor(.6666\*1030), replace = F)

lm1 = lm(Strength~., data = Concrete.trans[sam,])

lm2 = update(lm1, Strength~. - CourseAgg - FineAge, data = Concrete.trans[sam,])

lm2.step = step(lm2)

#Apply transformations

#pairs.plus(Concrete.trans)

Concrete.trans$Age = log(Concrete.trans$Age)

Concrete.trans$Superplast = yjPower(Concrete.trans$Superplast, 0.3)

Concrete.trans$BlastFurn = yjPower(Concrete.trans$FlyAsh, -0.1)

Concrete.trans$BlastFurn = log(Concrete.trans$BlastFurn+1)

Concrete.trans$Cement = bcPower(Concrete.trans$Cement, 0.2)

Concrete.trans$Water = bcPower(Concrete.trans$Water, 0.8)

Concrete.trans$Strength = bcPower(Concrete.trans$Strength, 0.6)

lm.trans = update(lm2.step, Strength~. , data = Concrete.trans[sam,])

#Adding polynomials to lm.trans

lm.poly = update(lm.trans, Strength~. + poly(Superplast, 2) + poly(Age,3) + poly(Water, 2))

data.frame(R.sq = c(summary(lm.poly)$r.squared\*100), Adj.R.sq=c(summary(lm.poly)$adj.r.squared)\*100)

summary(lm.poly)

write.csv(Concrete.trans, file = "Concrete.trans.csv")

}

For further preparation, we also modified the nnet.sscv function provided to work with our Y transformation. We used the Ecfun package’s invBoxCox to undo the transformation.

nnet.sscv.undobc = function(x,y,fit,p=.667,B=100,size=3,decay=fit$decay,skip=T,

linout=T,maxit=10000){

n = length(y)

MSEP = rep(0,B)

MAEP = rep(0,B)

MAPEP = rep(0,B)

ss = floor(n\*p)

for (i in 1:B){

sam = sample(1:n,ss,replace=F)

fit2 = nnet(x[sam,],y[sam],size=size,linout=linout,skip=skip,decay=decay,

maxit=maxit,trace=F)

ynew = predict(fit2,newdata=x[-sam,])

ystar = invBoxCox(y, 0.6)

ynew = invBoxCox(ynew, 0.6)

MSEP[i]=mean((ystar[-sam]-ynew)^2)

MAEP[i]=mean(abs(ystar[-sam]-ynew))

MAPEP[i]=mean(abs(ystar[-sam]-ynew)/ystar[-sam])

}

RMSEP = sqrt(mean(MSEP))

MAE = mean(MAEP)

MAPE = mean(MAPEP)

cat("RMSEP\n")

cat("===============\n")

cat(RMSEP,"\n\n")

cat("MAE\n")

cat("===============\n")

cat(MAE,"\n\n")

cat("MAPE\n")

cat("===============\n")

cat(MAPE,"\n\n")

temp = data.frame(MSEP=MSEP,MAEP=MAEP,MAPEP=MAPEP)

return(temp)

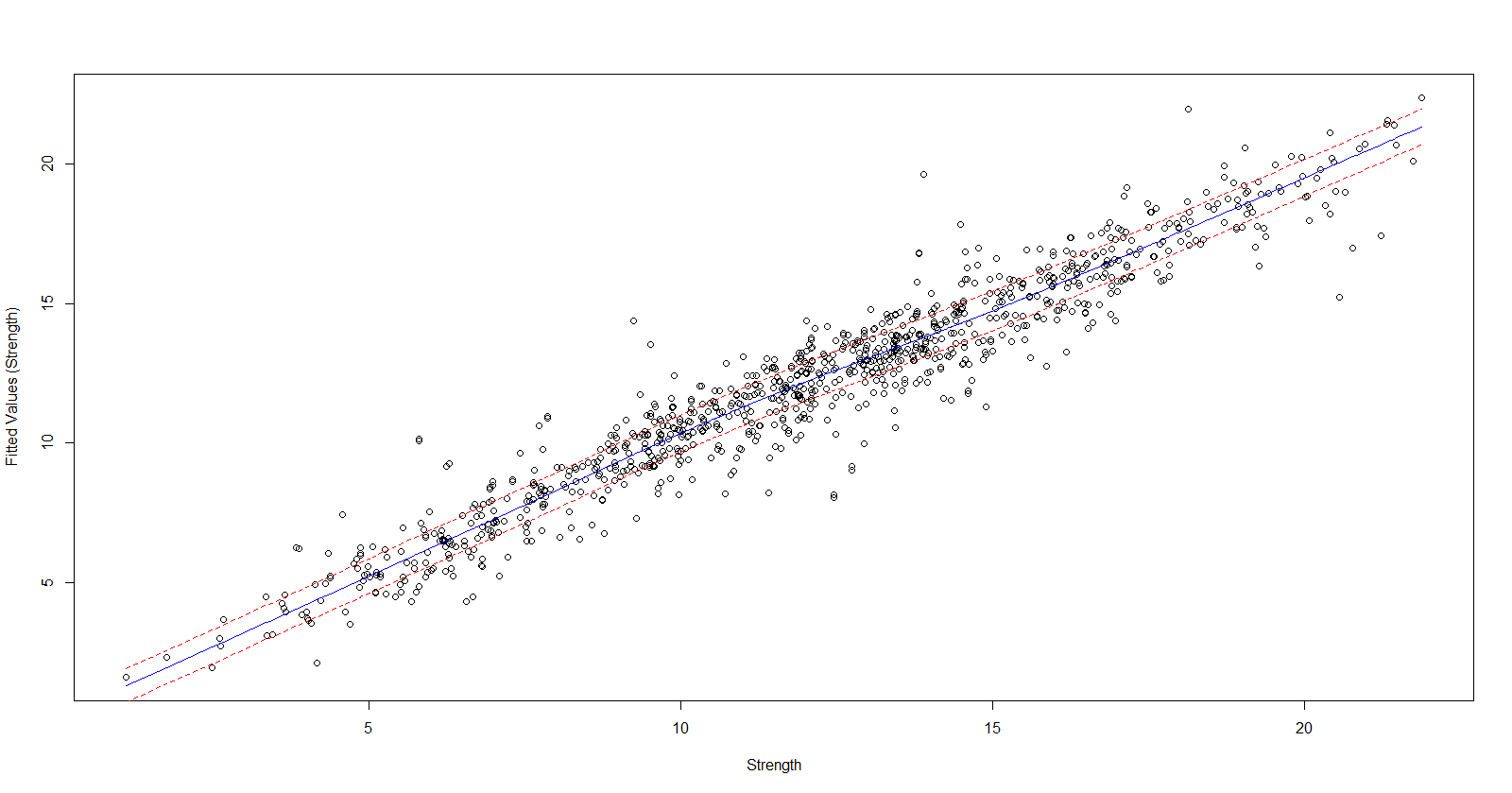
}

Now we can begin to start the process.

concrete.nn = nnet(Strength~., data = Concrete.trans, size = 10, linout = T, skip = T, maxit = 10000, decay = 0.001 )

summary(concrete.nn)

trendscat(Concrete.trans$Strength, fitted(concrete.nn), xlab = "Strength", ylab = "Fitted Values (Strength)")



cor(Concrete.trans$Strength, fitted(concrete.nn))^2

[,1]

[1,] 0.9232631

X = model.matrix(Strength~., data = Concrete.trans)[,-1]

Y = Concrete.trans$Strength

#Crossvalidation to choose a model

Now, we will begin to try to find the optimal number of nodes.

nnet.sscv.undobc(X,Y,concrete.nn, B = 20, size = 10, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

6.928472

MAE

===============

5.117516

MAPE

===============

0.1690735

MSEP MAEP MAPEP

1 42.20589 4.553442 0.1513221

2 36.47086 4.543853 0.1476454

3 69.03536 6.356005 0.2074229

4 53.35156 5.472622 0.2014797

5 50.86838 5.284149 0.1685449

6 41.32538 4.366085 0.1396963

7 40.95062 4.937747 0.1611668

8 48.65044 5.242851 0.1757694

9 48.71861 5.371367 0.1827731

10 39.73046 4.597272 0.1596614

11 34.68516 4.490082 0.1532898

12 64.62757 6.125789 0.1929521

13 45.37153 4.701649 0.1456717

14 49.96355 5.148309 0.1706484

15 43.48172 4.751932 0.1439478

16 57.66239 5.635149 0.1797413

17 43.85573 4.967428 0.1793716

18 48.92901 5.168526 0.1617575

19 52.15566 5.512942 0.1874684

20 48.03447 5.123115 0.1711398

Not horrible for an initial run-through, although our MAPE is quite impressive compared to previous models, we will work by playing with various factors to attempt to improve it as much as we can. Next, we will try less nodes, although this is likely going to negatively impact our metrics, there’s a possibility that a simpler model may be the solution.

> nnet.sscv.undobc(X,Y,concrete.nn, B = 20, size = 8, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.422189

MAE

===============

5.297053

MAPE

===============

0.1707535

MSEP MAEP MAPEP

1 59.83778 5.599104 0.1723279

2 58.74288 5.847102 0.1927552

3 48.98044 5.305295 0.1708226

4 53.42375 5.159871 0.1647904

5 57.39672 5.872429 0.1842843

6 38.90150 4.144174 0.1148887

7 48.20920 5.158465 0.1683633

8 48.18191 5.233513 0.1744633

9 43.36130 4.774433 0.1524784

10 60.22068 5.921356 0.1928724

11 48.22562 5.128004 0.1726139

12 49.76679 5.008914 0.1719175

13 112.39721 4.661402 0.1427477

14 56.49378 5.521208 0.1785869

15 64.10281 6.145705 0.1876357

16 62.29148 5.894267 0.1961223

17 34.04840 4.339017 0.1491474

18 55.60940 5.625437 0.1877355

19 57.80825 5.671852 0.1746865

20 43.77793 4.929506 0.1658309

> nnet.sscv.undobc(X,Y,concrete.nn, B = 20, size = 8, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.422189

MAE

===============

5.297053

MAPE

===============

0.1707535

MSEP MAEP MAPEP

1 59.83778 5.599104 0.1723279

2 58.74288 5.847102 0.1927552

3 48.98044 5.305295 0.1708226

4 53.42375 5.159871 0.1647904

5 57.39672 5.872429 0.1842843

6 38.90150 4.144174 0.1148887

7 48.20920 5.158465 0.1683633

8 48.18191 5.233513 0.1744633

9 43.36130 4.774433 0.1524784

10 60.22068 5.921356 0.1928724

11 48.22562 5.128004 0.1726139

12 49.76679 5.008914 0.1719175

13 112.39721 4.661402 0.1427477

14 56.49378 5.521208 0.1785869

15 64.10281 6.145705 0.1876357

16 62.29148 5.894267 0.1961223

17 34.04840 4.339017 0.1491474

18 55.60940 5.625437 0.1877355

19 57.80825 5.671852 0.1746865

20 43.77793 4.929506 0.1658309

8 nodes were clearly worse, although that may be the result of the 13th iteration, so we will try it again.

> nnet.sscv.undobc(X,Y,concrete.nn, B = 20, size = 8, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.046353

MAE

===============

5.204476

MAPE

===============

0.167951

MSEP MAEP MAPEP

1 41.78721 4.667282 0.1560655

2 39.94044 4.926829 0.1573080

3 61.15761 5.689059 0.1781244

4 47.95838 5.206386 0.1604907

5 42.78656 4.724125 0.1539419

6 33.74772 4.113586 0.1301719

7 46.00012 4.945549 0.1691954

8 42.04761 4.961267 0.1623008

9 63.43460 6.126033 0.2003583

10 50.68496 5.436841 0.1741960

11 52.41729 4.918697 0.1491896

12 47.75225 5.453992 0.1823952

13 39.42555 4.770308 0.1508629

14 66.44524 6.184930 0.2054391

15 58.30279 5.895867 0.1861123

16 59.72108 5.777906 0.1863383

17 46.02381 4.878304 0.1503177

18 55.86008 5.556746 0.1858185

19 43.81666 4.705300 0.1523910

20 53.71170 5.150521 0.1680026

This certainly is better without having such an extreme result. However, it still fails to improve. Up next we decided to try 5 just to see if the trend continues as the nodes decrease.

nnet.sscv.undobc(X,Y,concrete.nn, B = 20, size = 5, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.670899

MAE

===============

5.758918

MAPE

===============

0.1878979

MSEP MAEP MAPEP

1 58.49334 5.675381 0.1814132

2 69.09526 6.239047 0.1952364

3 61.46554 5.827643 0.1970164

4 53.30773 5.634324 0.1675558

5 50.91470 5.477039 0.1801894

6 57.96805 5.798493 0.1849397

7 55.45433 5.557227 0.1821288

8 58.52789 5.319077 0.1803671

9 58.54599 5.744945 0.1908777

10 39.91487 4.813849 0.1686804

11 67.39378 6.194138 0.1862264

12 59.13656 5.534705 0.1889202

13 62.71256 5.961575 0.1903121

14 67.03911 6.295253 0.2020273

15 54.48866 5.503832 0.1800953

16 63.79003 5.984657 0.1886587

17 63.44425 6.181331 0.2100792

18 53.24876 5.393522 0.1714421

19 62.26196 6.110122 0.2165002

20 59.65033 5.932209 0.1952908

The trend continues as expected, although it should be noted that cutting the number of nodes in half only increased the MAPE by about 2%, which says something about how robust neural networks can be. Unfortunately, this does not help us get a better model. Moving forward, we will try to increase the number of nodes instead of lowering it. Doubling the initial amount to 20 will be what we try first.

> concrete.nn2 = nnet(Strength~., data = Concrete.trans, size = 20, linout = T, skip = T, maxit = 10000, decay = 0.001 )

# weights: 209

initial value 236055921.513096

iter 10 value 45443449.033225

iter 20 value 2539639.917998

iter 30 value 809255.343286

iter 40 value 4409.082301

iter 50 value 3610.050485

iter 60 value 3450.343665

iter 70 value 3363.726927

iter 80 value 3242.394391

iter 90 value 3197.259742

iter 100 value 2986.386687

iter 110 value 2927.607750

iter 120 value 2880.400595

iter 130 value 2845.248872

iter 140 value 2829.042898

iter 150 value 2826.488930

iter 160 value 2824.474925

iter 170 value 2821.769432

final value 2820.756044

converged

> nnet.sscv.undobc(X,Y,concrete.nn2, B = 20, size = 5, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.48063

MAE

===============

5.553514

MAPE

===============

0.1791841

MSEP MAEP MAPEP

1 59.30702 5.619452 0.1824102

2 62.32574 5.751343 0.1821350

3 71.38641 6.306629 0.1997294

4 39.12725 4.429138 0.1508145

5 61.39795 6.027506 0.1972733

6 47.52892 4.938669 0.1573586

7 58.87716 5.630012 0.1710317

8 61.44775 5.885333 0.1889172

9 65.94430 6.074539 0.1817877

10 49.24657 5.318523 0.1795560

11 63.49907 6.163610 0.2021536

12 51.20902 5.375783 0.1752239

13 50.05170 5.292125 0.1752190

14 56.41646 5.552995 0.2042218

15 40.34870 4.705705 0.1597546

16 49.21462 5.081357 0.1587595

17 68.57727 6.421984 0.1996248

18 60.10170 5.757888 0.1672519

19 45.13742 4.936332 0.1550571

20 58.05157 5.801358 0.1954019

Our results have stagnated. Doubling the number of nodes should have made a more notable impact then fractions of a percentages, especially using something that large. So are problem likely lies elsewhere in our model. Perhaps our regression is getting in the way of the model’s performance, so an attempt was made again with 10 nodes having it removed.

> concrete.nn3 = nnet(Strength~., data = Concrete.trans, size = 10, linout = T, skip = F, maxit = 10000, decay = 0.001 )

# weights: 101

initial value 194184.204502

final value 17177.293540

converged

> nnet.sscv.undobc(X,Y,concrete.nn3, B = 20, size = 10, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

6.96364

MAE

===============

5.016101

MAPE

===============

0.1652912

MSEP MAEP MAPEP

1 70.81615 5.325735 0.1589369

2 38.98374 4.815806 0.1569582

3 33.70471 4.055971 0.1452389

4 49.22890 5.057397 0.1659998

5 35.71209 4.526487 0.1551388

6 58.94281 5.286610 0.1764939

7 50.95005 4.935022 0.1614314

8 47.01865 5.045905 0.1703704

9 49.55304 4.977572 0.1637779

10 52.54180 5.643173 0.1897062

11 35.76517 4.700917 0.1528407

12 50.66378 5.384170 0.1717481

13 62.06032 6.098245 0.2011496

14 48.75258 4.967330 0.1632928

15 54.76358 5.631640 0.1807603

16 35.52287 4.420186 0.1499180

17 43.79246 4.418209 0.1476072

18 54.39225 4.988107 0.1572042

19 40.51284 4.484006 0.1505701

20 56.16778 5.559541 0.1866803

Removing the skip layer had almost no impact. Perhaps our max iterations is to low, as the documentation suggests that one may need to jack it up significantly in order to get the best results.

> concrete.nn4 = nnet(Strength~., data = Concrete.trans, size = 10, linout = T, skip = F, maxit = 50000, decay = 0.001 )

# weights: 101

initial value 182263.398815

final value 17177.354645

converged

>

> nnet.sscv.undobc(X,Y,concrete.nn4, B = 20, size = 5, maxit = 10000, decay = 0.001, linout = T)

RMSEP

===============

7.717155

MAE

===============

5.761859

MAPE

===============

0.1855343

MSEP MAEP MAPEP

1 71.02641 6.338181 0.1918474

2 60.71804 5.795452 0.1867026

3 69.95729 6.341144 0.2045130

4 61.24800 5.671217 0.1789734

5 55.81221 5.568129 0.1922377

6 48.27522 5.288466 0.1627201

7 50.40735 5.320643 0.1567780

8 55.22204 5.529910 0.1747163

9 44.72119 4.996788 0.1684352

10 67.02531 6.184411 0.1829860

11 66.21327 5.931493 0.2001467

12 59.44575 5.719152 0.1777122

13 65.10390 6.246747 0.2110402

14 59.83844 5.578199 0.1882715

15 64.26599 6.131949 0.2023405

16 67.60931 6.259164 0.1985768

17 65.57738 6.014404 0.1818821

18 53.32035 5.564635 0.1899864

19 64.21298 6.038713 0.1842295

20 41.08909 4.718385 0.1765909

Once again, there is no amount of difference in our metrics that we would consider truly significant. This leaves us with the thought that perhaps we aren’t punishing the model for overfitting drastically enough. Therefore, we elected to try and increase decay to see if that makes more of an impact.

> nnet.sscv.undobc(X,Y,concrete.nn4, B = 20, size = 10, maxit = 10000, decay = 0.005, linout = T)

RMSEP

===============

6.757826

MAE

===============

4.819851

MAPE

===============

0.1575307

MSEP MAEP MAPEP

1 58.33862 5.597676 0.1791951

2 39.30745 4.465956 0.1430181

3 35.32276 4.321208 0.1452584

4 61.02288 5.871385 0.1977973

5 61.99729 6.030925 0.2000676

6 44.93868 4.820652 0.1549334

7 63.95930 4.585101 0.1426623

8 62.78217 5.919062 0.1954913

9 37.94853 4.368695 0.1389377

10 34.65265 4.387855 0.1481905

11 31.01756 3.910696 0.1341861

12 48.59418 5.089852 0.1640939

13 32.08323 3.947923 0.1309994

14 31.48846 4.032015 0.1424298

15 42.46043 4.613777 0.1343870

16 57.40524 5.689778 0.1803368

17 38.75570 4.345162 0.1391898

18 34.11054 4.192926 0.1424511

19 50.63765 5.230773 0.1763723

20 46.54079 4.975602 0.1606157

While the improvements are miniscule, these are some of the best results yet. However, most of our testing thus far has been from the enacting of extremes. A more balanced approach likely will lead us to an optimal solution.

nnet.sscv.undobc(X,Y,concrete.nn5, B = 30, size = 9, maxit = 100000, decay = 0.01, linout = T)

RMSEP

===============

6.523354

MAE

===============

4.697858

MAPE

===============

0.1531427

MSEP MAEP MAPEP

1 41.74075 4.650920 0.1501403

2 48.52365 5.040294 0.1584166

3 45.54055 5.064334 0.1641142

4 48.26219 4.886502 0.1711865

5 44.79124 5.057849 0.1720531

6 41.00505 4.596589 0.1473253

7 29.51773 4.006795 0.1383674

8 45.55905 4.873425 0.1596682

9 37.29754 4.482744 0.1421056

10 51.99251 5.139618 0.1630049

11 60.74199 5.867400 0.1918762

12 37.17427 4.402429 0.1500245

13 34.36744 4.485256 0.1458390

14 50.45036 4.513574 0.1347101

15 30.22424 4.128541 0.1315252

16 38.25450 4.511880 0.1499320

17 34.40341 4.369998 0.1382825

18 39.16964 4.512287 0.1504365

19 51.16925 4.977378 0.1626638

20 47.82819 4.617558 0.1416126

21 38.05189 4.491874 0.1447091

22 56.40523 5.664609 0.1763524

23 38.65565 4.465464 0.1552907

24 29.87631 4.089140 0.1387033

25 39.05310 4.527301 0.1456776

26 43.85529 4.813287 0.1665997

27 37.84878 4.463726 0.1479171

28 46.52443 5.094686 0.1590225

29 43.47460 4.334837 0.1379050

30 44.86551 4.805429 0.1588198

Here, we have used a combination of efforts and managed to lower the total amount even lower, although not by any huge amount. Still, this is the best one tried yet. We have some ideas for a few more, which we will attempt below to see what they result in.

> nnet.sscv.undobc(X,Y,concrete.nn5, B = 30, size = 6, maxit = 100000, decay = 0.01, linout = T)

RMSEP

===============

6.987689

MAE

===============

5.146594

MAPE

===============

0.16606

MSEP MAEP MAPEP

1 35.94723 4.461102 0.1431756

2 37.91736 4.577142 0.1450238

3 45.09267 4.965685 0.1601340

4 35.76216 4.314253 0.1438869

5 68.37110 6.091942 0.1854834

6 42.84485 4.829507 0.1557255

7 54.18984 5.604808 0.1816576

8 43.37872 4.602394 0.1425792

9 60.70221 6.018327 0.1900693

10 44.73100 4.865911 0.1560252

11 51.18520 5.110309 0.1677322

12 49.16745 5.284900 0.1673847

13 44.57714 4.897390 0.1568085

14 58.47393 5.736200 0.1812966

15 39.28516 4.589825 0.1440569

16 43.53453 4.670088 0.1595848

17 46.36014 5.090236 0.1734501

18 62.20102 5.892907 0.1881025

19 47.22062 5.226942 0.1579081

20 38.09643 4.698214 0.1486656

21 42.06468 4.761350 0.1485738

22 53.24798 5.542452 0.1855233

23 61.51344 5.509813 0.1795022

24 45.98001 5.040098 0.1636047

25 68.20341 6.279635 0.1906767

26 49.36515 4.887219 0.1666790

27 54.67984 5.507258 0.1887195

28 34.34333 4.456420 0.1543447

29 67.08293 6.260187 0.2008274

30 39.31460 4.625300 0.1545971

> nnet.sscv.undobc(X,Y,concrete.nn5, B = 30, size = 7, maxit = 100000, decay = 0.02, linout = T)

RMSEP

===============

6.872325

MAE

===============

5.045542

MAPE

===============

0.1627542

MSEP MAEP MAPEP

1 41.60383 4.549655 0.1362783

2 45.61013 5.015509 0.1653266

3 47.33779 5.301104 0.1636686

4 39.35810 4.711785 0.1585565

5 66.01042 6.264618 0.1863067

6 44.77287 4.889465 0.1616606

7 35.87447 4.262481 0.1411931

8 41.11660 4.719309 0.1534768

9 58.82595 5.875571 0.1918544

10 42.56136 4.379262 0.1460268

11 51.37096 5.358703 0.1726238

12 35.34576 4.269702 0.1410690

13 56.65308 5.821542 0.1908384

14 57.31264 5.379076 0.1573593

15 39.63334 4.394703 0.1439009

16 47.35459 5.160330 0.1707205

17 50.74873 5.442174 0.1826082

18 46.79453 4.867680 0.1531788

19 43.75028 4.950847 0.1629584

20 52.66947 4.948204 0.1580224

21 47.82857 5.259235 0.1755876

22 48.33131 4.875576 0.1542869

23 51.18615 5.434907 0.1730599

24 36.25625 4.269106 0.1451540

25 36.15602 4.583989 0.1504524

26 40.11199 4.734089 0.1540675

27 61.15727 6.042038 0.1814680

28 44.61482 4.852778 0.1538368

29 62.52078 5.969740 0.1976005

30 43.99751 4.783077 0.1594829

> nnet.sscv.undobc(X,Y,concrete.nn5, B = 30, size = 9, maxit = 100000, decay = 2, linout = T)

RMSEP

===============

6.604547

MAE

===============

4.867758

MAPE

===============

0.1613072

MSEP MAEP MAPEP

1 39.53301 4.589368 0.1430644

2 44.47169 4.865057 0.1483791

3 42.91745 4.898519 0.1747110

4 41.44850 4.642999 0.1537012

5 38.79132 4.531977 0.1441612

6 39.65641 4.701690 0.1494675

7 40.72549 4.674858 0.1575879

8 41.97980 4.675835 0.1489167

9 40.26858 4.656938 0.1622602

10 41.25742 4.818427 0.1544057

11 44.12163 4.711519 0.1548767

12 39.60880 4.665272 0.1670438

13 41.95411 4.879630 0.1689413

14 41.86763 4.796021 0.1477754

15 39.08331 4.611126 0.1543721

16 44.89999 4.968863 0.1724396

17 42.88280 4.707095 0.1623578

18 57.39192 5.324507 0.1564206

19 49.21982 5.182175 0.1677060

20 41.23915 4.630256 0.1561216

21 51.48698 5.347940 0.1791218

22 33.02899 4.303529 0.1419200

23 43.31282 5.039186 0.1618882

24 42.36527 4.906863 0.1755481

25 54.17304 5.264422 0.1818438

26 46.15171 5.167834 0.1706413

27 57.99086 5.458711 0.1719516

28 41.61021 4.886248 0.1583478

29 39.63205 4.850466 0.1604473

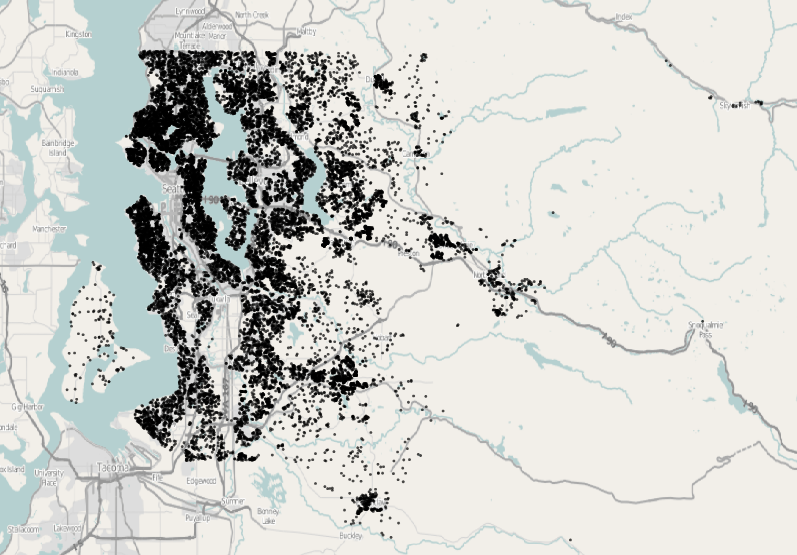
30 45.53031 5.275408 0.1927957

1. Using training (66%) and validation (33%) sets compare the predictive performance (RMSEP, MAE, MAPE) of your neural network model from part (a) and your best MLR and MARS models from Assignment 2. Use the same training/validation set for all three, thus you will need to run your best MLR and MARS models again using the same training and test sets used for the neural network model. (12 pts.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Method | (training data) | (MPa) | (MPa) | (%) |
| MLR best |  |  |  |  |
| MARS best |  |  |  |  |
| Neural Net |  |  |  |  |

**Problem 2 – PREDICTING SELLING PRICE OF HOMES IN KING COUNTY, WA**

The data for these sales comes from the official public records of home sales in the King County area, Washington State. The data set contains 21,606 homes that sold between May 2014 and May 2015. The table below gives variable names and descriptions. The map below shows the location of all 21,606 homes you will be working with.

****

**Variables in King County, WA Datasets**

* ID – id number (DO NOT USE IN YOUR MODELS!)
* **price** - Price of each home sold
* **bedrooms** - Number of bedrooms
* **bathrooms** - Number of bathrooms, where .5 accounts for a room with a toilet but no shower.
* **sqft\_living** - Square footage of the apartments interior living space.
* **sqft\_lot** - Square footage of the land space.
* **floors** - Number of floors.
* **waterfront** - A categorical variable for whether the apartment/home was overlooking the waterfront or not (1 = yes, 0 = no).
* **view** - An ordinal index from 0 to 4 of how good the view of the property has.
* **condition** - An index from 1 to 5 on the condition of the apartment**.**
* **grade** - An ordinal index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.  Other intermediary values indicate conditions in between these descriptors.
* **sqft\_above** - The square footage of the interior housing space that is above ground level.
* **sqft\_basement** - The square footage of the interior housing space that is below ground level.
* **yr\_built** - The year the house was initially built.
* **yr\_renovated** - The year of the house’s last renovation, 0 indicates it has not been renovated.
* **renovated** – indicator of whether or not the home has been renovated (1 = yes, 0 = no)
* **zipcode –** ZIP code area the house is in (Note: ZIP codes are NOT numeric!)
* **lat -** Lattitude of the home
* **long**- Longitude of the home
* **sqft\_living15**- The mean square footage of the interior living space of the nearest fifteen neighboring homes.
* **sqft\_lot15** -The mean square footage of the land lots of the nearest fifteen neighboring homes.
* **Test Set** – denotes whether the home is in the Test Set or the Training Set. These sets are the same as those for Assignment 1.

1. Using the **King County Homes (full).JMP** file on the course website develop a neural network model for predicting home price in JMP. Be sure to use some form of cross-validation to fine-tune your model. DO NOT USE BOOSTING! Also I would NOT recommend using multiple tours to fit the model, as this will take a long time for even a modest number of tours. Include plots of the actual vs. predicted in both the log-scale (assuming you used as the response) and in the original scale (both the predicted and actual prices in $).

Discuss the process you used to arrive at your final model. Include a diagram of your final model from JMP. Include results of the cross-validation for your final model. (20 pts.)

1. Can you identify which variables appear to be the most important in predicting the selling price (or log selling price) from your final model? If so, which variables seem the most important? Can you create plots/visualizations or create summary statistics that show which predictors/terms are most important? I DO NOT HAVE A SPECIFIC THING I AM LOOKING FOR HERE, I JUST WANT TO SEE IF YOU CAN COME UP WITH SOMETHING. (10 pts.)
2. Use your model to predict the selling price of the homes in the test set (denoted by the ***Test Set*** column) and submit those predictions in the same format as your predictions for Assignment 1. I will again create a leaderboard based upon these predictions. I will demonstrate how to do this in class – remind me! (5 pts.)