stackedGen

November 13, 2019

1 Stacking

Stacked generalization is an ensemble technique that involves two stages. The first stage trains a set of M models on N observations and stores their predictions in a $L_1 = N \times M$ data set. The second stage uses the dataset of predictions to train an ensemble model, called a **generalizer**. The generalizer is meant to learn the strengths and weaknesses of each model through their predictions, and built a more accurate model.

The goal of a stacked generalization algorithm is not to better understand the relationship between a set of covariates X and a target variable y. The goal is to make as accurate a model as possible.

1.1 Stages

We will use the Boston housing market dataset to understand stacking. The goal will be to predict the median value of homes in Boston given several characteristics of the housing market.

[1]:
```
require(mlbench)
data(BostonHousing)

dim(BostonHousing)
summary(BostonHousing)
```

```
Loading required package: mlbench
```

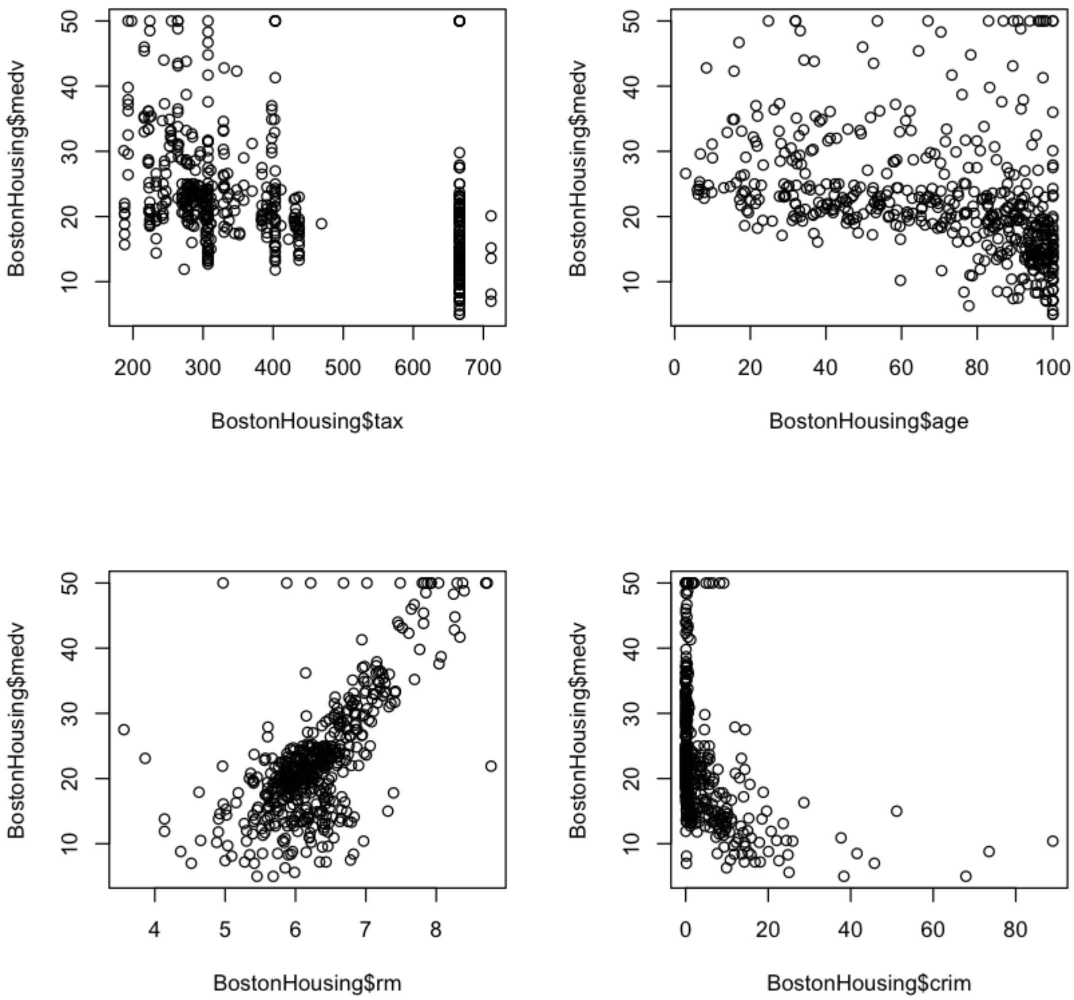
1. 506 2. 14

```
      crim                zn             indus       chas        nox
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1:  35   1st Qu.:0.4490
 Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240
 Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710
       rm             age              dis              rad
 Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
 1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000
 Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000
```

```
 Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549
 3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000
 Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
      tax           ptratio             b               lstat
 Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73
 1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95
 Median :330.0   Median :19.05   Median :391.44   Median :11.36
 Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65
 3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95
 Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97
      medv
 Min.   : 5.00
 1st Qu.:17.02
 Median :21.20
 Mean   :22.53
 3rd Qu.:25.00
 Max.   :50.00
```

[2]:

```
par(mfrow=c(2,2))
plot(BostonHousing$tax,BostonHousing$medv)
plot(BostonHousing$age,BostonHousing$medv)
plot(BostonHousing$rm,BostonHousing$medv)
plot(BostonHousing$crim,BostonHousing$medv)
```



1.2 (1) Partition data into training and testing.

Stacking occurs in stages. The first stage partitions your data into a training and test set.

[3]:
```
percentTraining = 0.80
BostonHousing['train'] = runif(nrow(BostonHousing)) < percentTraining

training = BostonHousing[BostonHousing$train==1,]
testing  = BostonHousing[BostonHousing$train==0,]

print(head(BostonHousing,10))
```

```
print('size of training data')
dim(training)

print('size of testing data')
dim(testing)
```

```
      crim   zn indus chas   nox    rm   age    dis rad tax ptratio      b
1  0.00632 18.0  2.31    0 0.538 6.575  65.2 4.0900   1 296    15.3 396.90
2  0.02731  0.0  7.07    0 0.469 6.421  78.9 4.9671   2 242    17.8 396.90
3  0.02729  0.0  7.07    0 0.469 7.185  61.1 4.9671   2 242    17.8 392.83
4  0.03237  0.0  2.18    0 0.458 6.998  45.8 6.0622   3 222    18.7 394.63
5  0.06905  0.0  2.18    0 0.458 7.147  54.2 6.0622   3 222    18.7 396.90
6  0.02985  0.0  2.18    0 0.458 6.430  58.7 6.0622   3 222    18.7 394.12
7  0.08829 12.5  7.87    0 0.524 6.012  66.6 5.5605   5 311    15.2 395.60
8  0.14455 12.5  7.87    0 0.524 6.172  96.1 5.9505   5 311    15.2 396.90
9  0.21124 12.5  7.87    0 0.524 5.631 100.0 6.0821   5 311    15.2 386.63
10 0.17004 12.5  7.87    0 0.524 6.004  85.9 6.5921   5 311    15.2 386.71
   lstat medv train
1   4.98 24.0  TRUE
2   9.14 21.6  TRUE
3   4.03 34.7  TRUE
4   2.94 33.4  TRUE
5   5.33 36.2  TRUE
6   5.21 28.7  TRUE
7  12.43 22.9  TRUE
8  19.15 27.1  TRUE
9  29.93 16.5  TRUE
10 17.10 18.9  TRUE
[1] "size of training data"
```

1. 410 2. 15

```
[1] "size of testing data"
```

1. 96 2. 15

1.3 (2) Cross validation for out-of-sample predictions

Next we split our training data into K folds. For every fold k, we train on the left over $K - 1$ folds and make predictions on fold k. We repeat this process for all M component models.

We will use a KNN neighbor regression, linear regression, polynomial regression, and regression tree to predict the median value of houses (medv).

[4]:
```
require(FNN)   # for the KNN model
require(rpart) # for the TBR model

K = 10
```

```r
training = training[,names(training)!='train']
trainingFolds = split(training,1:K)

dataSetOfPredictions = matrix()
for(k in 1:K){
    sprintf("Fold %d", k)

    outOfSample = trainingFolds[[k]]
    outOfSampleX = outOfSample[,names(outOfSample)!='medv']

    leftOver = setdiff(1:K,k)
    trainingSamples = do.call(rbind,trainingFolds[leftOver])

    trainY = trainingSamples[,names(trainingSamples)=='medv']
    trainX = trainingSamples[,names(trainingSamples)!='medv']

    # KNN model
    m1_predictions = knn.reg(train = as.matrix(trainingSamples$crim)
                ,test = as.matrix(outOfSampleX$crim)
                ,y=as.matrix(trainY),k=10)$pred

    #linear regression
    m2 = lm(medv~., data = trainingSamples)
    m2_predictions = predict(m2,outOfSampleX)

    #polynomial regression
    m3 = lm(medv~tax + age + rm + crim + I(tax^2) + I(age^2) + I(rm^2) + I(crim^2), data = trainingSamples)
    m3_predictions = predict(m3,outOfSampleX)

    #TBR
    m4 = rpart(medv ~ ., method="anova", data=trainingSamples)
    m4_predictions = predict(m4,outOfSampleX)

    # build data set of out-of-sample predictions
    if (k==1){
        allDataSetOfPredictions = cbind(m1_predictions,m2_predictions,m3_predictions,m4_predictions,outOfSample$medv)
    } else {
        dataSetOfPredictions    = cbind(m1_predictions,m2_predictions,m3_predictions,m4_predictions,outOfSample$medv)
        allDataSetOfPredictions = rbind(allDataSetOfPredictions,dataSetOfPredictions,outOfSample$medv)
    }
}
```

```
Loading required package: FNN
```

```
Loading required package: rpart

Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
Warning message in rbind(allDataSetOfPredictions, dataSetOfPredictions,
outOfSample$medv):
âĂIJnumber of columns of result is not a multiple of vector length (arg 3)âĂİ
```

[5]:

```
cbind(m1_predictions,m2_predictions,m3_predictions,m4_predictions,outOfSample$medv)
```

A matrix: 41 ÃŮ 5 of type dbl

	m1_predictions	m2_predictions	m3_predictions	m4_predictions	
10	21.46	19.027995	20.009662	18.359615	18.9
22	20.87	17.810408	19.477677	20.613793	19.6
34	16.27	14.443087	18.148213	18.359615	13.1
47	21.18	20.583063	22.681509	20.613793	20.0
62	21.46	18.960698	19.837244	18.359615	16.0
75	27.02	25.537439	24.221358	22.750877	24.1
85	23.64	24.591988	24.919208	22.750877	23.9
99	27.32	33.994662	40.953380	38.985714	43.8
111	21.85	20.466214	21.705076	20.613793	21.7
123	23.23	21.456362	21.415900	18.359615	20.5
136	36.97	17.452666	19.949913	14.569231	18.1
148	18.52	9.053367	16.316344	14.569231	14.6
158	16.25	33.179382	25.085168	27.868293	41.3
172	21.91	24.644861	17.224248	20.613793	19.1
187	24.94	34.923693	41.720271	38.985714	50.0
200	26.25	30.213712	29.624447	27.868293	34.9
213	19.14	22.254644	21.145573	18.359615	22.4
224	29.22	29.513213	24.083355	27.868293	30.1
237	32.22	29.175402	24.427489	27.868293	25.1
249	21.90	21.472451	24.011281	22.750877	24.5
259	27.13	35.801714	31.175662	32.234615	36.0
273	24.27	28.610831	25.888060	22.750877	24.4
291	27.41	33.957245	29.779209	27.868293	28.5
306	24.77	30.769238	26.579860	27.868293	28.4
317	23.76	17.670191	19.780897	18.359615	17.8
331	22.60	21.167586	22.090658	22.750877	19.8
341	22.56	21.728596	22.432882	22.750877	18.7
352	27.02	20.792584	24.974866	27.868293	24.1
362	19.63	19.196094	18.421405	20.613793	19.9
373	15.13	25.780359	15.137599	33.057143	50.0
387	10.47	6.670162	9.923174	9.361538	10.5
398	14.62	16.625857	14.448075	14.569231	8.5
409	14.79	13.831855	14.236169	18.359615	17.2
420	13.57	14.164846	20.958169	9.361538	8.4
434	22.50	17.084606	19.196868	14.569231	14.3
448	16.94	18.255262	16.135249	14.569231	12.6
460	17.96	18.817369	16.780565	14.569231	20.0
470	13.94	18.949890	14.634050	18.359615	20.1
483	22.38	28.447699	25.308605	32.234615	25.0
495	21.70	20.586318	20.789736	20.613793	24.5
506	23.36	22.706229	20.925367	33.057143	11.9

We now have a dataset that includes an out-of-sample for prediction, for all 4 models and for all observation in our training set.

[6]:
```
allDataSetOfPredictions = data.frame(allDataSetOfPredictions)
names(allDataSetOfPredictions) = c('m1','m2','m3','m4','y')
```

```
print(head(allDataSetOfPredictions))
```

```
       m1       m2       m3       m4    y
X1  32.47 30.30560 24.90512 23.60122 24.0
X11 20.21 18.50869 21.81050 16.97647 15.0
X23 18.72 15.52379 20.15813 16.97018 15.2
X36 24.24 24.12824 21.02700 23.60122 18.9
X48 20.78 17.68533 21.36763 16.97018 16.6
X63 21.35 24.29733 23.99810 23.60122 22.2
```

1.4 (3) Build Aggregator

The next step trains a model that aggregates the M models together by training on the data set of out-of-sample predictions. We can consider a linear regression model as our aggregator.

[7]:
```
agg = lm(y~m1+m2+m3+m4,data = data.frame(allDataSetOfPredictions))
print(summary(agg))
```

```
Call:
lm(formula = y ~ m1 + m2 + m3 + m4, data = data.frame(allDataSetOfPredictions))

Residuals:
     Min       1Q   Median       3Q      Max
-25.1627  -2.2186  -0.3911   1.8806  28.4699

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.25579    0.94770  -1.325   0.1859
m1          -0.04067    0.04703  -0.865   0.3877
m2           0.48766    0.05554   8.780   <2e-16 ***
m3           0.14123    0.05736   2.462   0.0142 *
m4           0.47185    0.05293   8.915   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 4.408 on 414 degrees of freedom
Multiple R-squared:  0.7628,    Adjusted R-squared:  0.7605
F-statistic: 332.8 on 4 and 414 DF,  p-value: < 2.2e-16
```

These estimates are for our training set. We still need to make predictions on our held out test set.

1.5 (4) make predictions on test set

1.5.1 (4.1) Train Models on whole training set

[8]:
```
# KNN model
    m1_predictions = knn.reg(train = as.matrix(training$crim)
                            ,test = as.matrix(testing$crim)
                            ,y=as.matrix(training$medv),k=10)$pred

    #linear regression
    m2 = lm(medv~., data = training)
    m2_predictions = predict(m2,testing)

    #polynomial regression
    m3 = lm(medv~tax + age + rm + crim + I(tax^2) + I(age^2) + I(rm^2) +␣
 ↪I(crim^2), data = training)
    m3_predictions = predict(m3,testing)

    #TBR
    m4 = rpart(medv ~ ., method="anova", data=training)
    m4_predictions = predict(m4,testing)


testSetPredictions =␣
 ↪cbind(m1_predictions,m2_predictions,m3_predictions,m4_predictions)
testSetPredictions = data.frame(testSetPredictions)
names(testSetPredictions) = c('m1','m2','m3','m4')

AggregatorModelPredictionsOnTestSet = predict(agg,data.frame(testSetPredictions))
print(AggregatorModelPredictionsOnTestSet)
```

```
       20        21        26        30        35        40        45        52
19.425249 14.770881 15.017293 21.199896 15.189805 28.215449 23.499058 23.922685
       53        56        58        60        64        71        73        87
26.177563 32.121589 32.353396 22.429286 27.197722 25.004058 24.485570 21.572056
       88        90        93       105       108       115       120       129
24.582588 31.364341 26.688665 20.814034 20.544854 22.750873 20.511577 17.991880
      130       131       137       141       159       160       161       163
15.363997 20.353915 16.265534 12.211035 35.172640 33.866898 37.488204 43.554484
      177       180       181       183       186       189       190       194
23.059418 32.390298 40.853508 32.909490 22.865273 28.476424 33.447676 32.102800
      201       206       211       218       225       233       234       240
31.554960 22.049231 19.629651 24.650720 43.769164 43.453029 43.218009 26.517492
      248       261       263       270       271       276       278       279
20.425024 33.048085 45.073319 23.197886 21.712013 32.983478 33.690091 27.347859
      280       282       283       285       288       295       296       298
33.857480 33.187698 43.987926 31.997501 25.514842 22.666929 26.841570 18.334608
```

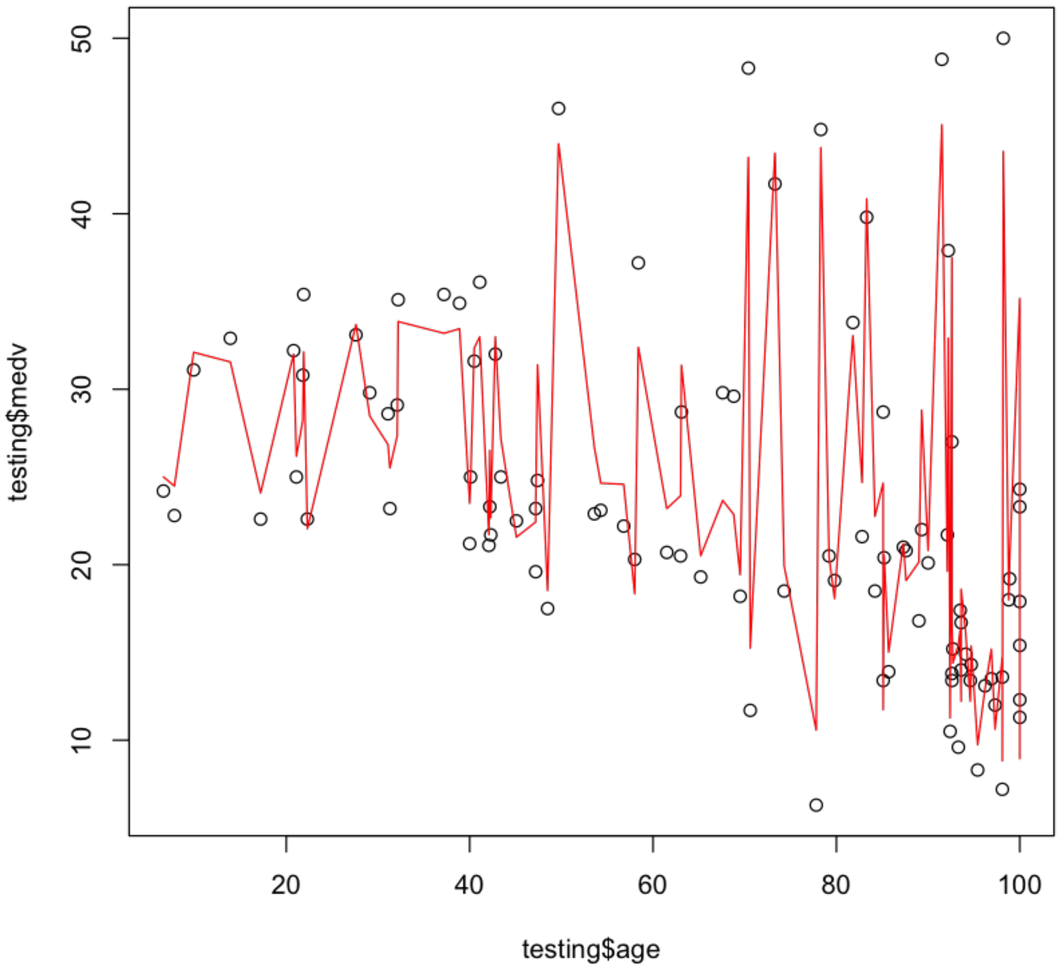
```
      301       305       314       322       324       325       330       346 
31.390072 32.980208 24.685590 24.647241 19.933483 24.752430 24.088464 18.529734 
      364       379       383       384       386       394       400       413 
20.151830 12.897429 11.719090 11.563689  8.823183 18.208772 10.571225  8.947883 
      423       424       425       426       436       437       441       444 
19.098862 11.735513 15.250283  9.736015 12.219236 15.431773 11.269219 17.527636 
      451       455       474       477       478       487       489       505 
16.851743 16.132665 23.666398 18.608707 10.621200 18.068553 14.390647 28.811103 
```

1.6 (5) Compare your aggregated model to the test set

[9]:
```
testSetAndAggPredictions = data.frame('T' = testing$medv
                                      ,'P' = AggregatorModelPredictionsOnTestSet
                                      ,'M1' = m1_predictions
                                      ,'M2' = m2_predictions
                                      ,'M3' = m3_predictions
                                      ,'M4' = m4_predictions)
SSE_AGG = sum( (testSetAndAggPredictions$T - testSetAndAggPredictions$P)^2 )

SSE_M1 = sum( (testSetAndAggPredictions$T - testSetAndAggPredictions$M1)^2 )
SSE_M2 = sum( (testSetAndAggPredictions$T - testSetAndAggPredictions$M2)^2 )
SSE_M3 = sum( (testSetAndAggPredictions$T - testSetAndAggPredictions$M3)^2 )
SSE_M4 = sum( (testSetAndAggPredictions$T - testSetAndAggPredictions$M4)^2 )


print(SSE_AGG)
print(SSE_M1)
print(SSE_M2)
print(SSE_M3)
print(SSE_M4)
```

```
[1] 1076.369
[1] 5304.312
[1] 1440.279
[1] 1380.285
[1] 2250.818
```

[22]:
```
plot(testing$age, testing$medv)

S = order(testing$age)

lines(testing$age[S], testSetAndAggPredictions$P[S], col='red')
```



testing$medv
testing$age
10
20
30
40
50
20
40
60
80
100


```
[ ]:
```