

# Data Mining Assignment 8

Xuan Han  
han.xua@husky.neu

November 20, 2015

```
> load('realEstate.RData')
> binQua = ifelse(realEstate$Quality == 1, 1, 0)
> binQua = as.factor(binQua)
> all.data = data.frame(realEstate, binQua)
> totalInstance = dim(realEstate)[1]
> trainSize = 350
> testSize = totalInstance - trainSize
> set.seed(1)
> trainIndex = sample(1:522, 350)
> trainSet = all.data[trainIndex, c(-1, -10)]
> validateSet = all.data[-trainIndex, c(-1, -10)]
```

---

## 1: Tree

---

(b)

```
> library(tree)
> tree.realEstate = tree(binQua~., trainSet)
> summary(tree.realEstate)
```

```
Classification tree:
tree(formula = binQua ~ ., data = trainSet)
Variables actually used in tree construction:
[1] "Sales" "SqFeet" "Year" "LotSize"
Number of terminal nodes: 10
Residual mean deviance: 0.09508 = 32.33 / 340
Misclassification error rate: 0.02286 = 8 / 350
```

1. Training error rate is 0.02286.
2. There are 10 terminal nodes.
3. Training error changes everytime with different samples.

(c)

```
> tree.realEstate
```

```
node), split, n, deviance, yval, (yprob)
* denotes terminal node
```

- ```
1) root 350 294.200 0 ( 0.85143 0.14857 )
 2) Sales < 364500 271 23.620 0 ( 0.99262 0.00738 )
   4) Sales < 322500 248 0.000 0 ( 1.00000 0.00000 ) *
   5) Sales > 322500 23 13.590 0 ( 0.91304 0.08696 )
      10) Sales < 335125 10 10.010 0 ( 0.80000 0.20000 ) *
      11) Sales > 335125 13 0.000 0 ( 1.00000 0.00000 ) *
 3) Sales > 364500 79 103.900 1 ( 0.36709 0.63291 )
```

```

6) Sales < 527875 48 64.440 0 ( 0.60417 0.39583 )
12) SqFeet < 3403.5 35 39.900 0 ( 0.74286 0.25714 )
24) Year < 1986.5 22 8.136 0 ( 0.95455 0.04545 )
48) LotSize < 21303 5 5.004 0 ( 0.80000 0.20000 ) *
49) LotSize > 21303 17 0.000 0 ( 1.00000 0.00000 ) *
25) Year > 1986.5 13 17.320 1 ( 0.38462 0.61538 )
50) LotSize < 22249 5 0.000 1 ( 0.00000 1.00000 ) *
51) LotSize > 22249 8 10.590 0 ( 0.62500 0.37500 ) *
13) SqFeet > 3403.5 13 14.050 1 ( 0.23077 0.76923 )
26) Sales < 458250 5 6.730 0 ( 0.60000 0.40000 ) *
27) Sales > 458250 8 0.000 1 ( 0.00000 1.00000 ) *
7) Sales > 527875 31 0.000 1 ( 0.00000 1.00000 ) *

```

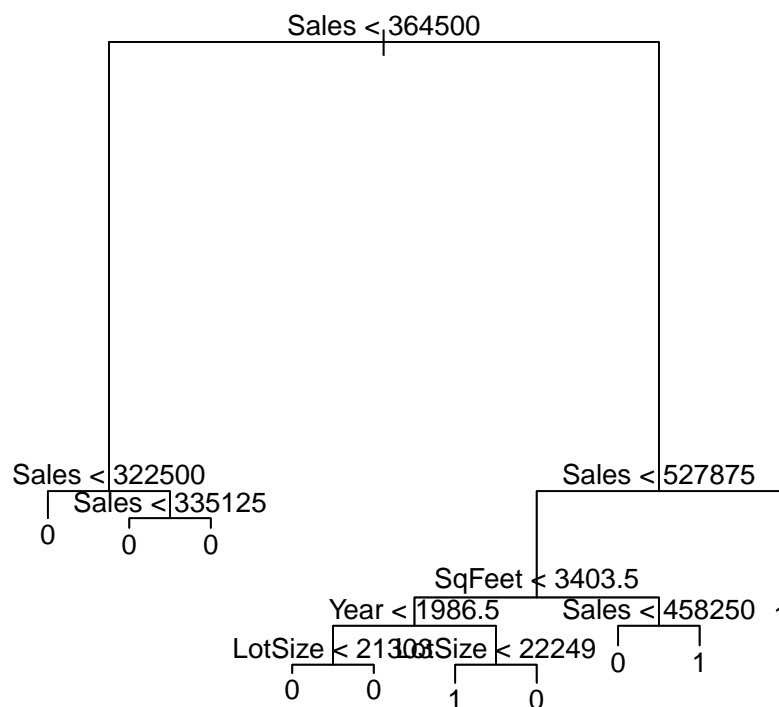
1. Let's look at node 2:
2. The split criterion is  $\text{Sales} < 322500$
3. There are 271 data points in this node.
4. This terminal node label is 0.
5. Deviance is 23.62.
6. 0.99262 percent of the data points are labeled 0, and 0.00738 of the data points are labeled 1.

(d)

```

> plot(tree.realEstate)
> text(tree.realEstate, pretty = 0)

```



1. This tree has depth 4.

2. The most important predictor is Sales. The first split differentiated most of the data points. And it appeared 5 times.

(e)

```
> tree.pred = predict(tree.realEstate, validateSet, type = 'class')
> table(tree.pred, validateSet$binQua)
```

```
tree.pred  0   1
          0 152   6
          1   4  10
```

```
> (6 + 4) / 172
```

```
[1] 0.05813953
```

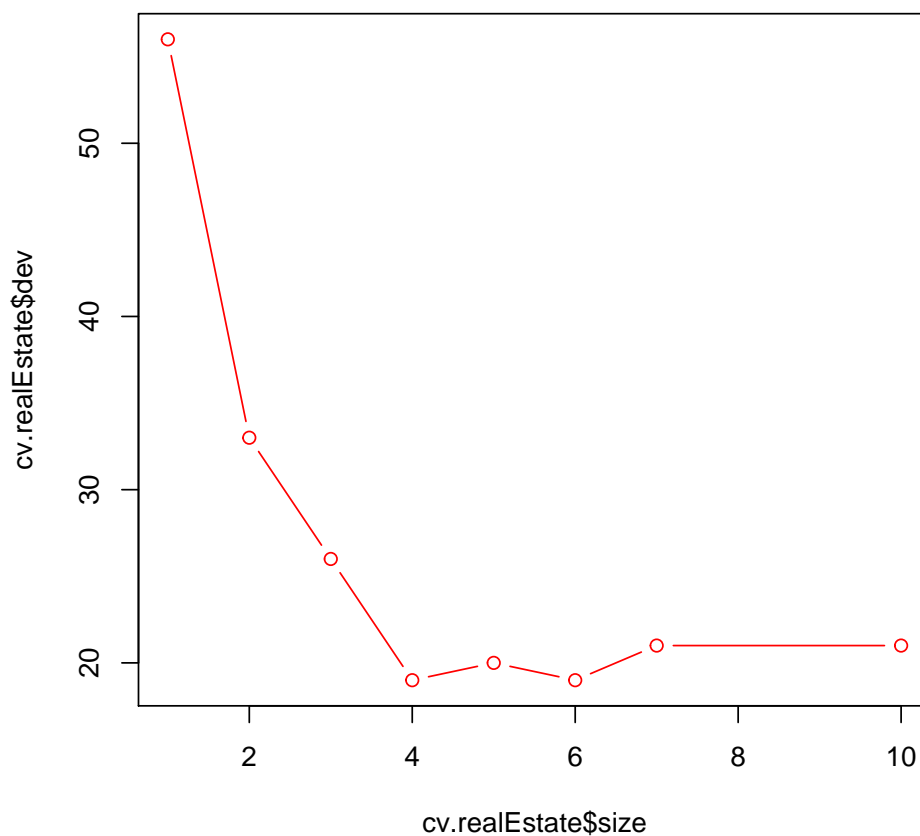
1. Test error rate is 0.05813953.

(f)

```
> cv.realEstate = cv.tree(tree.realEstate, FUN = prune.misclass)
```

(g)

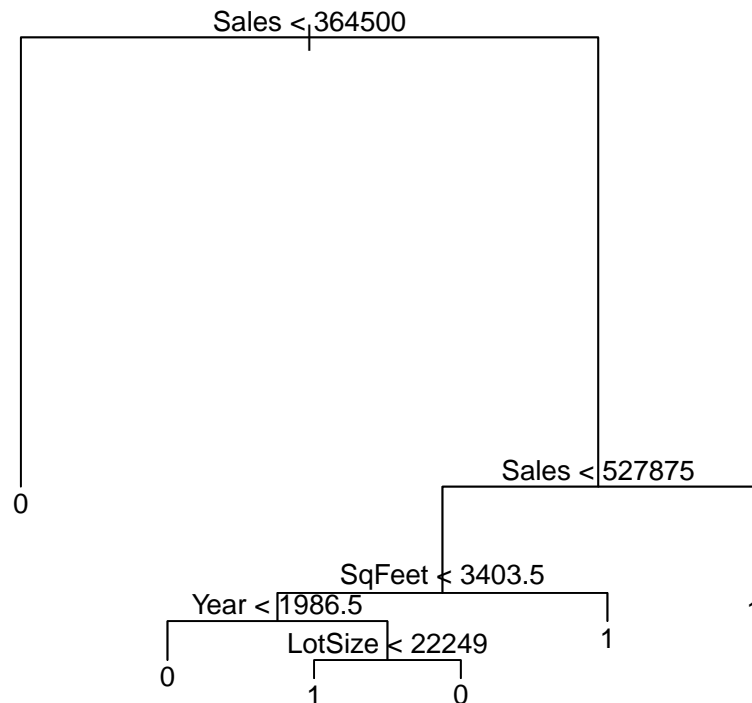
```
> plot(cv.realEstate$size, cv.realEstate$dev, type = 'b', col = 'red')
```



1. As show in the plot, tree size 4 and 6 corresponding to the lowest cross-validated error rate.

(h)

```
> prune.realEstate = prune.misclass(tree.realEstate, best = 6)
> plot(prune.realEstate)
> text(prune.realEstate, pretty = 0)
```



(j)

```
> prune.pred.train = predict(prune.realEstate, trainSet, type = 'class')
> table(prune.pred.train, trainSet$binQua)
```

```
prune.pred.train  0  1
                 0 295  6
                 1  3 46
```

```
> (6 + 3) / 350
```

```
[1] 0.02571429
```

1. Train error rate after prune is 0.0257, which is higher than unpruned tree.

(k)

```
> prune.pred.validate = predict(prune.realEstate, validateSet, type = 'class')
> table(prune.pred.validate, validateSet$binQua)
```

```
prune.pred.validate  0  1
                    0 148  5
                    1  8 11
```

```
> (8 + 5) / 172
```

```
[1] 0.0755814
```

1. Test error rate after prune is 0.0755814, which is higher than unpruned tree.

---

## 2: Bagging

---

```
> library(randomForest)
> set.seed(1)
> bag.realEstate = randomForest(binQua ~ ., data = trainSet, mtry = 11, importance = T, ntree = 100)
> bag.realEstate
```

Call:

```
randomForest(formula = binQua ~ ., data = trainSet, mtry = 11,      importance = T, ntree = 100)
      Type of random forest: classification
      Number of trees: 100
No. of variables tried at each split: 11
```

OOB estimate of error rate: 6.29%

Confusion matrix:

```
      0  1 class.error
0 289  9  0.03020134
1  13 39  0.25000000
```

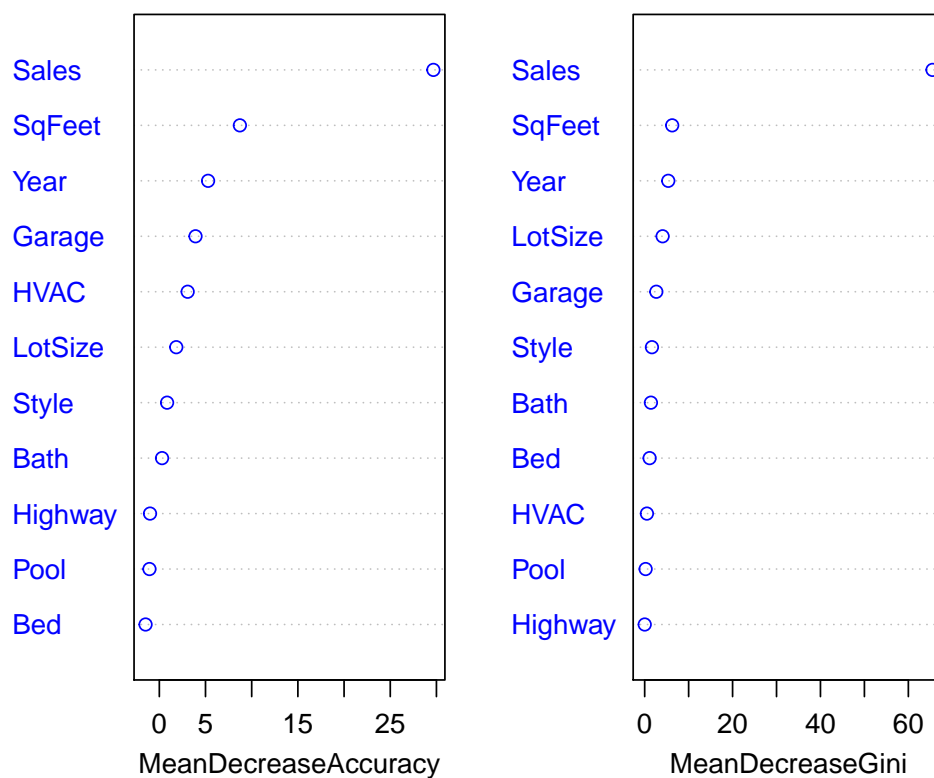
```
> varImpPlot(bag.realEstate, col = 'blue')
> yhat.bag = predict(bag.realEstate, newdata = validateSet)
> table(yhat.bag, validateSet$binQua)
```

```
yhat.bag  0  1
      0 150  4
      1  6 12
```

```
> (6 + 4) / 172
```

```
[1] 0.05813953
```

bag.realEstate



1. We can see that Sales is the most important predictor.
2. Train error rate is 0.0629, higher than the best single-tree classification.
3. Test error rate is 0.05813953, equal with the best single-tree classification.

---

**3: RandomForest**

---

```
> set.seed(1)
> forest.realEstate = randomForest(binQua ~ ., data = trainSet, mtry = 1, importance = T, ntree = 100)
> forest.realEstate
```

Call:

```
randomForest(formula = binQua ~ ., data = trainSet, mtry = 1, importance = T, ntree = 100)
      Type of random forest: classification
      Number of trees: 100
```

No. of variables tried at each split: 1

OOB estimate of error rate: 6.29%

Confusion matrix:

```
      0  1 class.error
0 295  3  0.01006711
1  19 33  0.36538462
```

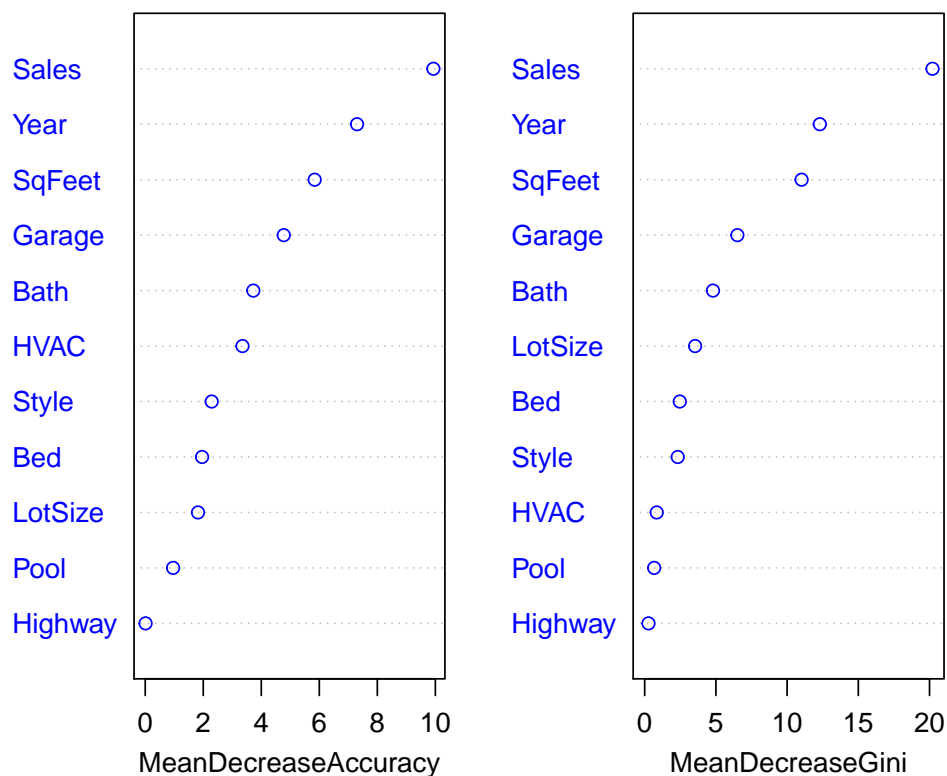
```
> varImpPlot(forest.realEstate, col = 'blue')
> yhat.forest = predict(forest.realEstate, newdata = validateSet)
> table(yhat.forest, validateSet$binQua)
```

```
yhat.forest  0  1
           0 153  4
           1  3 12
```

```
> (4 + 3) / 172
```

```
[1] 0.04069767
```

forest.realEstate



1. Train is 0.629, higher than best single tree.
2. Test error is 0.04069767, lower than best single tree.
3. Importance map changed a litter bit. Other variables have more infulence now, which is as expected.
4. Note that I set mtry = 1, which gives the best test error.



---

## 4: Boosting

---

```
> require(gbm)
> set.seed(1)
> shrinkages = seq(from = 0, to = 0.4, by = 0.01)
> trainAccu = rep(NA, length(shrinkages))
> testAccu = rep(NA, length(shrinkages))
> counter = 1
> for (s in shrinkages) {
+   boost.realEstate = gbm(binQua ~ ., data = trainSet, distribution = "multinomial", n.trees = 1000)
+   yhat.boost.array = predict(boost.realEstate, newdata = validateSet, n.trees = 1000, type = 'response')
+   yhat.boost = ifelse(yhat.boost.array[,1,1] > yhat.boost.array[,2,1], 0, 1)
+   train.correct.rate = sum(yhat.boost == trainSet$binQua) / length(trainSet$binQua)
+   test.correct.rate = sum(yhat.boost == validateSet$binQua) / length(validateSet$binQua)
+
+   trainAccu[counter] = train.correct.rate
+   testAccu[counter] = test.correct.rate
+   counter = counter + 1
+ }
> par(mfrow = c(2, 2))
> plot(shrinkages, trainAccu, type = 'l')
> plot(shrinkages, testAccu, type = 'l')
> best = which.max(testAccu)
> 1 - testAccu[best]
```

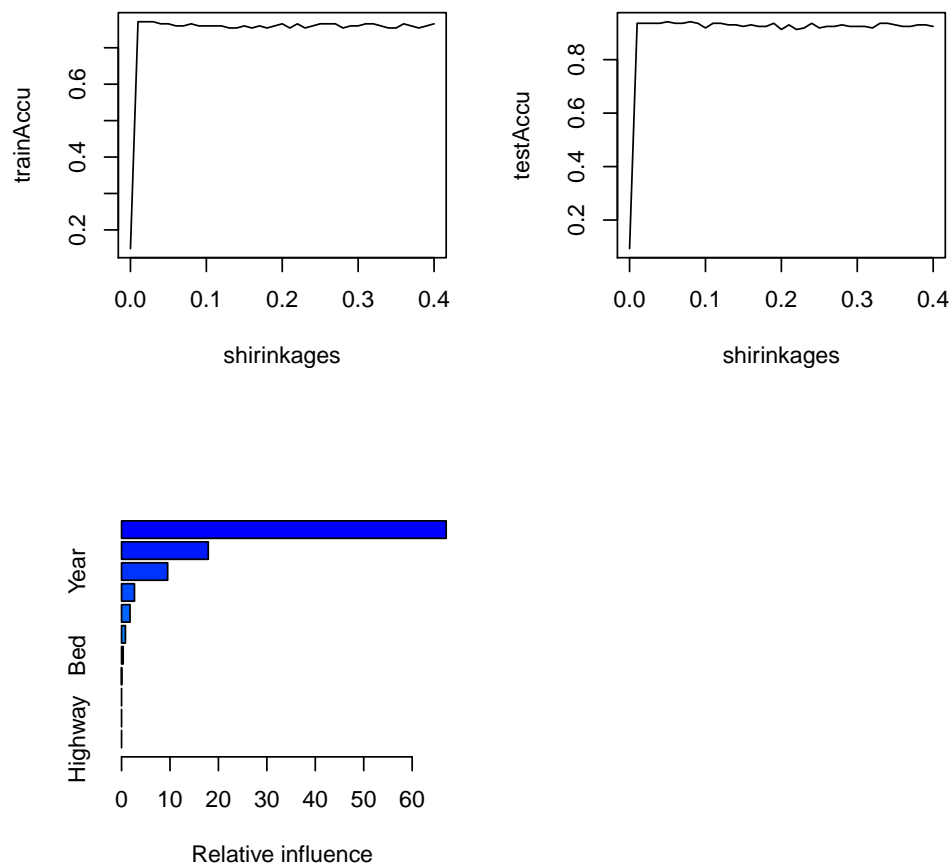
```
[1] 0.05813953
```

```
> 1 - trainAccu[best]
```

```
[1] 0.2342857
```

```
> summary(boost.realEstate)
```

|         | var     | rel.inf      |
|---------|---------|--------------|
| Sales   | Sales   | 6.702787e+01 |
| SqFeet  | SqFeet  | 1.789904e+01 |
| Year    | Year    | 9.516756e+00 |
| Bath    | Bath    | 2.668982e+00 |
| Garage  | Garage  | 1.751340e+00 |
| LotSize | LotSize | 7.986863e-01 |
| Bed     | Bed     | 3.334082e-01 |
| Style   | Style   | 3.877868e-03 |
| Pool    | Pool    | 3.577933e-05 |
| HVAC    | HVAC    | 0.000000e+00 |
| Highway | Highway | 0.000000e+00 |



1. Train error rate is 0.2342, much higher than best single tree model.
2. Test error rate is 0.05813953, equal with best single tree.
3. The most important variable is Sales and SqFeet.