CME250 Kaggle Contest Report

Nicholas Huang — nykh@stanford.edu — 006042630

June 14, 2016

1 Basic Information

Account name used in Kaggle contest: **nykh**

2 Prediction Method

I used a Random Forest of regression trees for this contest, with the **mtry** set to about $\frac{1}{3}p$. The random forest also required the use of imputation, for which I used an R library that implements the **Predictive Mean Matching** method.

3 Justification

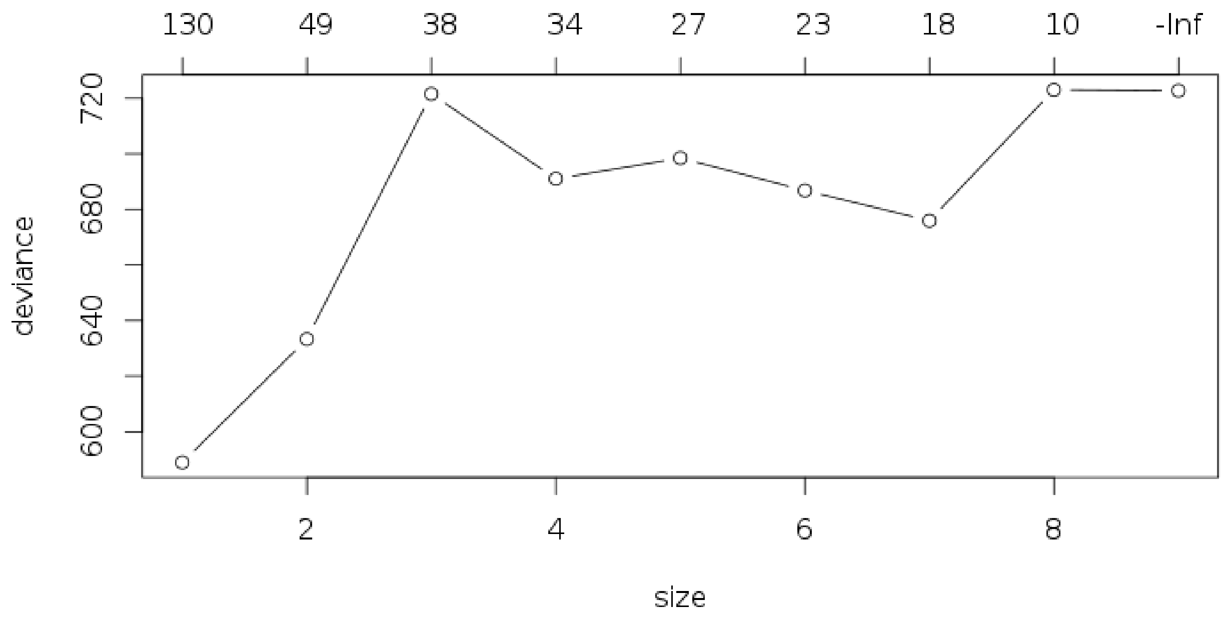
I started out using a pruned regression tree because it doesn't require imputation and has good interpretability. When tested on the testset, it gives Mean Squared Error to be about 7.74. I then chose the random forest method because I want to have a more accurate prediction, and also because I was already familiar with the method. Unfortunately it requires imputation. With the help of an imputation library the random forest gave me a MSE of 5.75, which is a drastic improvement, at which point I settled with the method.

4 Observation

One interesting tihng I found out while building the tree was the cross validation value of derivation has a strange relationship with the size, which doesn't conform to our expected shape. If I plot the `cv` object

```
    plot(cv.kaggle, type='b')
```

The result is shown below This is also why I chose 7 as the number of nodes for the pruned tree.



5 Appendix: Code listing

```
library(tree)
library(mice) # imputation library
library(randomForest)
set.seed(1)
data = read.csv('DataForFitting.csv')
train = sample(1:nrow(data), nrow(data)/2)

# regression tree requires no imputation
tree.kaggle = tree(Age~., data, subset=train)
summary(tree.kaggle)
cv.kaggle = cv.tree(tree.kaggle)
plot(cv.kaggle$size, cv.kaggle$dev, type='b')
prune.kaggle = prune.tree(tree.kaggle, best=7)

kaggle.test = data[-train, "Age"]
yhat.test = predict(tree.kaggle, newdata=data[-train,])         # MSE=8.01
yhat.test.prune = predict(prune.kaggle, newdata=data[-train,]) # MSE=7.74

# random forest requires imputation
input = mice(data, meth='pmm', seed=200)
data.imputed = complete(imput)
rf.kaggle = randomForest(Age~., data=data.imputed, subset=train, mtry=7,
    importance=T)
yhat.test.rf = predict(rf.kaggle, newdata=data.imputed[-train,]) # MSE=5.75

data.prediction = read.csv('PredictThese.csv')
data.predictors = cbind(Age=0, data.prediction[,2:22])
```

```

# surgery work to fix quirk of random forest where inconsistency in level
    prevents
# it from prediction
# solution found:
#   http://stackoverflow.com/questions/2375587/reorder-levels-of-a-data-
    frame-without-changing-order-of-values
data.predictors$PastJob = factor(data.predictors$PastJob, levels=levels(data
    $PastJob))
data.predictors$Voted = factor(data.predictors$Voted, levels=levels(data$
    Voted))
yhat.rf = predict(rf.kaggle, newdata=data.predictors)

submission = data.frame(ID=1:length(yhat.rf), Age=yhat.rf)
write.csv(submission, file="Submission.csv", row.names=FALSE)
```

algorithm/regression–tree.R