GRAD

STAT 149 Spring 2016 Final Project, Process Workbook #4

Submitted by: Kendrick Lo (Harvard ID: 70984997), Amy Lee (Harvard ID: 60984077)

This is a continuation of exploration of models from Process Workbooks #1 through #3, and focuses primarily on averaging and blending techniques, to try to give our models a little boost given that we are jostling for the top position on the public leaderboard.

Note that since the final results will be based on a hidden portion of the test data, performance on the public test set is not necessarily indicative of final performance, and the rankings may change significantly. Basing decisions on which models to commit may be aided by using some objective measure such as validation on a withheld portion of data (but due to time constraints, and since this is beyond the scope of the assignment, we did not do this here).

Reference: http://mlwave.com/kaggle-ensembling-guide/

```
# prediction files
best_GLM = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model26.csv", header=TRUE) # 0.53906
best_GAM = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model26f.csv", header=TRUE) # 0.53897

best_RF1 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model15e.csv", header=TRUE) # 0.54365
best_RF2 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model15d.csv", header=TRUE) # 0.54510

best_GBM1 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model_gbm1.csv", header=TRUE) # 0.54186
best_GBM2 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/model_gbm2.csv", header=TRUE) # 0.54271

best_NN1 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/nn4.csv", header=TRUE) # 0.53730
best_NN2 = read.csv("~/Desktop/stat149/StatKaggle/ensembles/nn3.csv", header=TRUE) # 0.53759

## additional models via scikitlearn

# Gradient Boosting (including squared and cubed age/memmonths)
best_sqgb = read.csv("~/Desktop/stat149/StatKaggle/ensembles/rfboost_new.csv", header=TRUE) # 0.53411

# Majority Classifiers (including squared and cubed age/memmonths)
# {Logistic, RF, GB, NN, Knn, SVM}
best_major = read.csv("~/Desktop/stat149/StatKaggle/ensembles/majority_1.csv", header=TRUE) # 0.54159

# SVM
best_svm = read.csv("~/Desktop/stat149/StatKaggle/ensembles/SVM.csv", header=TRUE) # 0.56438

# KNN
best_knn = read.csv("~/Desktop/stat149/StatKaggle/ensembles/KNN.csv", header=TRUE) # 0.55957
```

Correlation-based Averaging of Predictions

```
matrix_preds_1 = cbind(best_GAM$lapsed, best_RF1$lapsed,
                       best_GBM1$lapsed, best_NN1$lapsed)
```

```
avg_1 = cbind(best_GLM$Id, rowMeans(matrix_preds_1))

# average 5
matrix_preds_2 = cbind(best_GAM$lapsed, best_RF1$lapsed,
                       best_GBM1$lapsed, best_NN1$lapsed,
                       best_sqgb$lapsed)

cor(matrix_preds_2)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.8939037 0.9671814 0.9496587 0.9605971
## [2,] 0.8939037 1.0000000 0.8841572 0.8861818 0.9244563
## [3,] 0.9671814 0.8841572 1.0000000 0.9253046 0.9517568
## [4,] 0.9496587 0.8861818 0.9253046 1.0000000 0.9408465
## [5,] 0.9605971 0.9244563 0.9517568 0.9408465 1.0000000
```

```
avg_2 = cbind(best_GLM$Id, rowMeans(matrix_preds_2))
```

We initially tried submitting predictions based on a simple average of 4-5 of our best performing models (not including `sqgb`) of different types, and got a significant boost in our Kaggle score with the first averaging of model (0.53115), securing a second place position. The second averaging model resulting in a greater improvement, scoring 0.53049 and the competition lead (at the time).

It is really interesting to note how well the averaging technique worked. We had anticipated getting a result between the best and the worse scores obtained from the models we averaged, but in fact, we obtained a score that exceeded all four/five of the models considered individually. A good explanation (with an analogy to error correction) of how this can occur is provided in the earlier-cited reference.

It was also notable that when we computed the correlation between various models being considered, the correlation between predictions coming from the same model type were pretty high (as expected), at nearly 97.5% and above; however, the correlations between predictions coming from different model types were a few percentage points lower. The averaging seemed to be work best when using good models from different model classes.

```
# Write as submission file
colnames(avg_1) = c("Id", "lapsed")
write.csv(avg_1, file="~/Desktop/stat149/StatKaggle/ensembles/avg_1.csv", row.names=FALSE)
colnames(avg_2) = c("Id", "lapsed")
write.csv(avg_2, file="~/Desktop/stat149/StatKaggle/ensembles/avg_2.csv", row.names=FALSE)
```

Stacking

Finally, we attempted to build a stacked model. For each model, we would obtain predictions on the training set, and use those predictions as features in a new model, in a further layer. More specifically, for each model, we trained on one half of the training set to make predictions on the other help, and vice-versa. Therefore, we were able to get predictions for data in the whole training set, while ensuring that each observations would not "see" the answer for its record during training (otherwise, the predictions would surely be overconfident since the model already "knows" what the answer is supposed to be).

"The basic idea behind stacked generalization is to use a pool of base classifiers, then using another classifier to combine their predictions, with the aim of reducing the generalization error... A stacker model gets more information on the problem space by using the first-stage predictions as features, than if it was trained in isolation."

Load data

```
train = read.csv("~/Desktop/stat149/StatKaggle/train_add3.csv", header=TRUE)
test = read.csv("~/Desktop/stat149/StatKaggle/test_add3.csv", header=TRUE)
Id = read.csv("~/Desktop/stat149/StatKaggle/id.csv", header=TRUE)
```

```
# split up the training data
# n = 43436
# record 11650 is temporarily copied to the bottom half
# and then removed, as R needs each region level
# represented in both halves of the training set

toptrain = train[1:20000,]
bottomtest = rbind(train[20001:43436,], train[11650,])
bottomtest$lapsed <- NULL

toptest = train[1:20000,]
bottomtrain = rbind(train[20001:43436,], train[11650,])
toptest$lapsed <- NULL

actuals_binary = as.integer(train$lapsed=="Y")
actuals_factor = train$lapsed
```

Get predictions on training data for different models

```
# from Kaggle site
# used to compare sets of predictions
# should be pretty low (~0.5) if rows are not accidentally swapped
MultiLogLoss <- function(act, pred)
    {
      eps = 1e-15;
      nr <- nrow(pred)
      pred = matrix(sapply( pred, function(x) max(eps,x)), nrow = nr)
      pred = matrix(sapply( pred, function(x) min(1-eps,x)), nrow = nr)
      ll = sum(act*log(pred) + (1-act)*log(1-pred))
      ll = ll * -1/(nrow(act))
      return(ll);
    }
```

```
# Stacked Model 1: Logistic Regression
model26.glm.top <- glm(lapsed ~ age * (age_12_under + age_13_15 + age_16_18
                                  + age_19_24 + age_25_64 + age_65_plus)
                    + sex + region + nregions + memtype + mem_mag1 + mem_mag2
                    + hasemail + r1 + r2 + r3 + r.quick + extra + intl
                    + r.intl + allgames5yr * (games_0 + games_1_5 + games_6_10
                                              + games_11_20 + games_21_34
                                              + games_35_49 + games_50_plus)
                    + fastevents + medevents + slowevents + nfloor + age.na
                    + r1.na + r2.na + r3.na + r.quick.na + r.intl.na
                    + memmonths * (mon_less30 + mon_31 + mon_32 + mon_33
```

```
                                  + mon_34 + mon_35 + mon_36 + mon_37_60
                                  + mon_61_84 + mon_85_120 + mon_121_263
                                  + mon_264_plus)
                    + allgames1yr * (games_0 + games_1_5 + games_6_10
                                     + games_11_20 + games_21_34 + games_35_49
                                     + games_50_plus) + age:memtype
                    + memtype:r1 + sex:r1 + memtype:hasemail + age:sex
                    + memtype:hasemail:r1 + sex:hasemail:r1
                    + age:sex:memtype, family = "binomial", data = toptrain)

glmbottompreds = predict(model26.glm.top, newdata=bottomtest, type="response")

model26.glm.bottom <- glm(lapsed ~ age * (age_12_under + age_13_15 + age_16_18
                                   + age_19_24 + age_25_64 + age_65_plus)
                    + sex + region + nregions + memtype + mem_mag1 + mem_mag2
                    + hasemail + r1 + r2 + r3 + r.quick + extra + intl
                    + r.intl + allgames5yr * (games_0 + games_1_5 + games_6_10
                                              + games_11_20 + games_21_34
                                              + games_35_49 + games_50_plus)
                    + fastevents + medevents + slowevents + nfloor + age.na
                    + r1.na + r2.na + r3.na + r.quick.na + r.intl.na
                    + memmonths * (mon_less30 + mon_31 + mon_32 + mon_33
                                  + mon_34 + mon_35 + mon_36 + mon_37_60
                                  + mon_61_84 + mon_85_120 + mon_121_263
                                  + mon_264_plus)
                    + allgames1yr * (games_0 + games_1_5 + games_6_10
                                     + games_11_20 + games_21_34 + games_35_49
                                     + games_50_plus) + age:memtype
                    + memtype:r1 + sex:r1 + memtype:hasemail + age:sex
                    + memtype:hasemail:r1 + sex:hasemail:r1
                    + age:sex:memtype, family = "binomial", data = bottomtrain)

glmtoppreds = predict(model26.glm.bottom, newdata=toptest, type="response")

glmpreds = c(glmtoppreds, glmbottompreds)[1:43436]

preds = cbind(glmpreds, 1-glmpreds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

```
# Stacked Model 2: GAM
library(gam)
model26f.top.gam <- gam(lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18
                                  + age_19_24 + age_25_64 + age_65_plus)
                    + sex + region + s(nregions) + memtype + mem_mag1
                    + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3)
                    + s(r.quick) + extra + intl + s(r.intl)
                    + s(allgames5yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + s(fastevents) + s(medevents) + s(slowevents)
                    + s(nfloor) + age.na + r1.na + r2.na + r3.na
                    + r.quick.na + r.intl.na
```

```
                    + s(memmonths) * (mon_less30 + mon_31 + mon_32 + mon_33
                                      + mon_34 + mon_35 + mon_36 + mon_37_60
                                      + mon_61_84 + mon_85_120 + mon_121_263
                                      + mon_264_plus)
                    + s(allgames1yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail
                    + age:sex + memtype:hasemail:r1 + sex:hasemail:r1
                    + age:sex:memtype, family = "binomial", data = toptrain)

gambottompreds = predict(model26f.top.gam, newdata=bottomtest, type="response")

model26f.bottom.gam <- gam(lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18
                                      + age_19_24 + age_25_64 + age_65_plus)
                    + sex + region + s(nregions) + memtype + mem_mag1
                    + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3)
                    + s(r.quick) + extra + intl + s(r.intl)
                    + s(allgames5yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + s(fastevents) + s(medevents) + s(slowevents)
                    + s(nfloor) + age.na + r1.na + r2.na + r3.na
                    + r.quick.na + r.intl.na
                    + s(memmonths) * (mon_less30 + mon_31 + mon_32 + mon_33
                                      + mon_34 + mon_35 + mon_36 + mon_37_60
                                      + mon_61_84 + mon_85_120 + mon_121_263
                                      + mon_264_plus)
                    + s(allgames1yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail
                    + age:sex + memtype:hasemail:r1 + sex:hasemail:r1
                    + age:sex:memtype, family = "binomial", data = bottomtrain)

gamtoppreds = predict(model26f.bottom.gam, newdata=toptest, type="response")

gampreds = c(gamtoppreds, gambottompreds)[1:43436]

preds = cbind(gampreds, 1-gampreds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

```
# Stacked Model 3: Random Forest 1
library(randomForest)

regionprops = summary(train$region)
regionprops = regionprops/43436
tempregion = as.character(train$region)
for (i in 1:55){tempregion <- replace(tempregion, tempregion == names(regionprops[i]), regionprops[i])}
tempregion2 = as.character(test$region)
for (i in 1:55){tempregion2 <- replace(tempregion2, tempregion2 == names(regionprops[i]), regionprops[i]
```

```
train2 <- train
train2$region = as.double(tempregion)

# train <- train2

# Resplit
toptrain = train2[1:20000,]
bottomtest = train2[20001:43436,]
bottomtest$lapsed <- NULL

toptest = train2[1:20000,]
bottomtrain = train2[20001:43436,]
toptest$lapsed <- NULL

# actuals = as.integer(train$lapsed=="Y")

model15e.top.rf <- randomForest(lapsed ~ ., data=toptrain, ntree=3000, mtry=11, do.trace=10)

predicted = predict(model15e.top.rf, bottomtest, type="prob")
rf1bottompreds = predicted[,2]

model15e.bottom.rf <- randomForest(lapsed ~ ., data=bottomtrain, ntree=3000, mtry=11, do.trace=10)

predicted = predict(model15e.bottom.rf, toptest, type="prob")
rf1toppreds = predicted[,2]

rf1preds = c(rf1toppreds, rf1bottompreds)

preds = cbind(rf1preds, 1-rf1preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check

# Stacked Model 4: Random Forest 2

model15d.top.rf <- randomForest(lapsed ~ ., data=toptrain, ntree=3000, do.trace=10)

predicted = predict(model15d.top.rf, bottomtest, type="prob")
rf2bottompreds = predicted[,2]

model15d.bottom.rf <- randomForest(lapsed ~ ., data=bottomtrain, ntree=3000, do.trace=10)

predicted = predict(model15d.bottom.rf, toptest, type="prob")
rf2toppreds = predicted[,2]

rf2preds = c(rf2toppreds, rf2bottompreds)

preds = cbind(rf2preds, 1-rf2preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

```
# Stacked Model 5: GBM 1
library(gbm)
```

```r
train2 <- train

# convert logicals to factors
train2$mon_less30 = as.factor(train2$mon_less30)
train2$mon_31 = as.factor(train2$mon_31)
train2$mon_32 = as.factor(train2$mon_32)
train2$mon_33 = as.factor(train2$mon_33)
train2$mon_34 = as.factor(train2$mon_34)
train2$mon_35 = as.factor(train2$mon_35)
train2$mon_36 = as.factor(train2$mon_36)
train2$mon_37_60 = as.factor(train2$mon_37_60)
train2$mon_61_84 = as.factor(train2$mon_61_84)
train2$mon_85_120 = as.factor(train2$mon_85_120)
train2$mon_121_263 = as.factor(train2$mon_121_263)
train2$mon_264_plus = as.factor(train2$mon_264_plus)
train2$mon_less30 = as.factor(train2$mon_less30)

train2$games_0 = as.factor(train2$games_0)
train2$games_1_5 = as.factor(train2$games_1_5)
train2$games_6_10 = as.factor(train2$games_6_10)
train2$games_11_20 = as.factor(train2$games_11_20)
train2$games_21_34 = as.factor(train2$games_21_34)
train2$games_35_49 = as.factor(train2$games_35_49)
train2$games_50_plus = as.factor(train2$games_50_plus)

train2$age_12_under = as.factor(train2$age_12_under)
train2$age_13_15 = as.factor(train2$age_13_15)
train2$age_16_18 = as.factor(train2$age_16_18)
train2$age_19_24 = as.factor(train2$age_19_24)
train2$age_25_64 = as.factor(train2$age_25_64)
train2$age_65_plus = as.factor(train2$age_65_plus)

lapsed_binary = as.integer(train2$lapsed=="Y")

# train <- train2
train2$lapsed = lapsed_binary

# Resplit (no factors for region now)
toptrain = train2[1:20000,]
bottomtest = train2[20001:43436,]
bottomtest$lapsed <- NULL

toptest = train2[1:20000,]
bottomtrain = train2[20001:43436,]
toptest$lapsed <- NULL

# actuals = train$lapsed

gbm1.top.fit <- gbm(lapsed ~ ., data=toptrain, dist="adaboost", n.tree=400, cv.folds=10)

gbm1bottompreds = predict(gbm1.top.fit, newdata=bottomtest, n.tree=400, type="response")

gbm1.bottom.fit <- gbm(lapsed ~ ., data=bottomtrain, dist="adaboost", n.tree=400, cv.folds=10)
```

```
gbm1toppreds = predict(gbm1.bottom.fit, newdata=toptest, n.tree=400, type="response")

gbm1preds = c(gbm1toppreds, gbm1bottompreds)

preds = cbind(gbm1preds, 1-gbm1preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check

# Stacked Model 6: GBM 2

gbm2.top.fit <- gbm(lapsed ~ ., data=toptrain, dist="adaboost", n.tree=171, cv.folds=10)

gbm2bottompreds = predict(gbm2.top.fit, newdata=bottomtest, n.tree=171, type="response")

gbm2.bottom.fit <- gbm(lapsed ~ ., data=bottomtrain, dist="adaboost", n.tree=171, cv.folds=10)

gbm2toppreds = predict(gbm2.bottom.fit, newdata=toptest, n.tree=171, type="response")

gbm2preds = c(gbm2toppreds, gbm2bottompreds)

preds = cbind(gbm2preds, 1-gbm2preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

```
# Model 7: Neural Network 1
library(nnet)
library(caret)

# Data Preparation
# Index of columns to standardize
m_age = mean(train$age); sd_age = sd(train$age)
train$age = (train$age - m_age)/sd_age; test$age = (test$age - m_age)/sd_age

m_nregions = mean(train$nregions); sd_nregions = sd(train$nregions)
train$nregions = (train$nregions - m_nregions)/sd_nregions; test$nregions = (test$nregions - m_nregions)

m_memmonths = mean(train$memmonths); sd_memmonths = sd(train$memmonths)
train$memmonths = (train$memmonths - m_memmonths)/sd_memmonths; test$memmonths = (test$memmonths - m_mem

m_r1 = mean(train$r1); sd_r1 = sd(train$r1)
train$r1 = (train$r1 - m_r1)/sd_r1; test$r1 = (test$r1 - m_r1)/sd_r1

m_r2 = mean(train$r2); sd_r2 = sd(train$r2)
train$r2 = (train$r2 - m_r2)/sd_r2; test$r2 = (test$r2 - m_r2)/sd_r2

m_r3 = mean(train$r3); sd_r3 = sd(train$r3)
train$r3 = (train$r3 - m_r3)/sd_r3; test$r3 = (test$r3 - m_r3)/sd_r3

m_r.quick = mean(train$r.quick); sd_r.quick = sd(train$r.quick)
train$r.quick = (train$r.quick - m_r.quick)/sd_r.quick; test$r.quick = (test$r.quick - m_r.quick)/sd_r.q

m_r.intl = mean(train$r.intl); sd_r.intl = sd(train$r.intl)
train$r.intl = (train$r.intl - m_r.intl)/sd_r.intl; test$r.intl = (test$r.intl - m_r.intl)/sd_r.intl
```

```
m_allgames1yr = mean(train$allgames1yr); sd_allgames1yr = sd(train$allgames1yr)
train$allgames1yr = (train$allgames1yr - m_allgames1yr)/sd_allgames1yr; test$allgames1yr = (test$allgame

m_allgames5yr = mean(train$allgames5yr); sd_allgames5yr = sd(train$allgames5yr)
train$allgames5yr = (train$allgames5yr - m_allgames5yr)/sd_allgames5yr; test$allgames5yr = (test$allgame

m_fastevents = mean(train$fastevents); sd_fastevents = sd(train$fastevents)
train$fastevents = (train$fastevents - m_fastevents)/sd_fastevents; test$fastevents = (test$fastevents -

m_medevents = mean(train$medevents); sd_medevents = sd(train$medevents)
train$medevents = (train$medevents - m_medevents)/sd_medevents; test$medevents = (test$medevents - m_me

m_slowevents = mean(train$slowevents); sd_slowevents = sd(train$slowevents)
train$slowevents = (train$slowevents - m_slowevents)/sd_slowevents; test$slowevents = (test$slowevents -

m_nfloor = mean(train$nfloor); sd_nfloor = sd(train$nfloor)
train$nfloor = (train$nfloor - m_nfloor)/sd_nfloor; test$nfloor = (test$nfloor - m_nfloor)/sd_nfloor

train$age.na = as.factor(train$age.na); test$age.na = as.factor(test$age.na)
train$r1.na = as.factor(train$r1.na); test$r1.na = as.factor(test$r1.na)
train$r2.na = as.factor(train$r2.na); test$r2.na = as.factor(test$r2.na)
train$r3.na = as.factor(train$r3.na); test$r3.na = as.factor(test$r3.na)
train$r.quick.na = as.factor(train$r.quick.na); test$r.quick.na = as.factor(test$r.quick.na)
train$mon_less30 = as.factor(train$mon_less30); test$mon_less30  = as.factor(test$mon_less30)
train$mon_31 = as.factor(train$mon_31); test$mon_31  = as.factor(test$mon_31)
train$mon_32 = as.factor(train$mon_32); test$mon_32  = as.factor(test$mon_32)
train$mon_33 = as.factor(train$mon_33); test$mon_33  = as.factor(test$mon_33)
train$mon_34 = as.factor(train$mon_34); test$mon_34  = as.factor(test$mon_34)
train$mon_35 = as.factor(train$mon_35); test$mon_35  = as.factor(test$mon_35)
train$mon_36 = as.factor(train$mon_36); test$mon_36  = as.factor(test$mon_36)
train$mon_37_60 = as.factor(train$mon_37_60); test$mon_37_60  = as.factor(test$mon_37_60)
train$mon_61_84 = as.factor(train$mon_61_84); test$mon_61_84  = as.factor(test$mon_61_84)
train$mon_85_120 = as.factor(train$mon_85_120); test$mon_85_120  = as.factor(test$mon_85_120)
train$mon_121_263 = as.factor(train$mon_121_263); test$mon_121_263  = as.factor(test$mon_121_263)
train$mon_264_plus = as.factor(train$mon_264_plus); test$mon_264_plus  = as.factor(test$mon_264_plus)
train$games_0 = as.factor(train$games_0); test$games_0  = as.factor(test$games_0)
train$games_1_5 = as.factor(train$games_1_5); test$games_1_5  = as.factor(test$games_1_5)
train$games_6_10 = as.factor(train$games_6_10); test$games_6_10  = as.factor(test$games_6_10)
train$games_11_20 = as.factor(train$games_11_20); test$games_11_20  = as.factor(test$games_11_20)
train$games_21_34 = as.factor(train$games_21_34); test$games_21_34  = as.factor(test$games_21_34)
train$games_35_49 = as.factor(train$games_35_49); test$games_35_49  = as.factor(test$games_35_49)
train$games_50_plus = as.factor(train$games_50_plus); test$games_50_plus  = as.factor(test$games_50_plus

## one hot encoding of response variable
dummies <- dummyVars(lapsed~ ., data = train)
lapsed = train$lapsed
train2 = cbind(lapsed, data.frame(predict(dummies, newdata = train)))

# train <- train2

# Resplit (no factors for region now)
toptrain = train2[1:20000,]
bottomtest = train2[20001:43436,]
```

```
bottomtest$lapsed <- NULL

toptest = train2[1:20000,]
bottomtrain = train2[20001:43436,]
toptest$lapsed <- NULL

# actuals = train$lapsed

nn1.top <- train(lapsed~ ., data=toptrain, method='nnet', trace = TRUE,
                 maxit=2000, MaxNWts=2000,
                 tuneGrid=expand.grid(.size=c(3),
                                      .decay=c(0.1))) # train

prediction <- predict(nn1.top, newdata=bottomtest, type="prob")
nn1bottompreds = prediction[,2]

nn1.bottom <- train(lapsed~ ., data=bottomtrain, method='nnet', trace = TRUE,
                 maxit=2000, MaxNWts=2000,
                 tuneGrid=expand.grid(.size=c(3),
                                      .decay=c(0.1))) # train

prediction <- predict(nn1.bottom, newdata=toptest, type="prob")
nn1toppreds = prediction[,2]

nn1preds = c(nn1toppreds, nn1bottompreds)

preds = cbind(nn1preds, 1-nn1preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check

# Model 8: Neural Network 2

nn2.top <- train(lapsed~ ., data=toptrain, method='nnet', trace = TRUE,
                 maxit=2000, MaxNWts=2000,
                 tuneGrid=expand.grid(.size=c(2),
                                      .decay=c(1.0))) # train

prediction <- predict(nn2.top, newdata=bottomtest, type="prob")
nn2bottompreds = prediction[,2]

nn2.bottom <- train(lapsed~ ., data=bottomtrain, method='nnet', trace = TRUE,
                 maxit=2000, MaxNWts=2000,
                 tuneGrid=expand.grid(.size=c(3),
                                      .decay=c(0.1))) # train

prediction <- predict(nn2.bottom, newdata=toptest, type="prob")
nn2toppreds = prediction[,2]

nn2preds = c(nn2toppreds, nn2bottompreds)

preds = cbind(nn2preds, 1-nn2preds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

```r
# Model 9: Gradient Boosting Model with Polynomial Factors
temp = read.csv("~/Desktop/stat149/StatKaggle/ensembles/gradient_boost_half_train.csv", header=TRUE)

gbpolypreds = temp$lapsed

preds = cbind(gbpolypreds, 1-gbpolypreds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check

# Model 10: Support Vector Machine (via sci kit learn)
temp = read.csv("~/Desktop/stat149/StatKaggle/ensembles/svm_half_train.csv", header=TRUE)

svmpreds = temp$lapsed

preds = cbind(svmpreds, 1-svmpreds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check

# Model 11: K-Nearest Neighbor (via sci kit learn, k=50 via c.v.)
temp = read.csv("~/Desktop/stat149/StatKaggle/ensembles/knn_half_train.csv", header=TRUE)

knnpreds = temp$lapsed

preds = cbind(knnpreds, 1-knnpreds)
acts = cbind(actuals_binary, 1-actuals_binary)
MultiLogLoss(acts, preds) / 2  # just to check
```

Combine predictions

```r
# gbm1preds/gbm2preds were highly insignificant, so drop them
trainX = cbind(glmpreds, gampreds, rf1preds, rf2preds, nn1preds, nn2preds, gbpolypreds, svmpreds, knnpr

trainY = actuals_factor

testX = cbind(best_GLM$lapsed, best_GAM$lapsed, best_RF1$lapsed, best_RF2$lapsed, best_NN1$lapsed, best

# write.csv(trainX, file="~/Desktop/stat149/StatKaggle/ensembles/trainX.csv", row.names=FALSE)
# write.csv(trainY, file="~/Desktop/stat149/StatKaggle/ensembles/trainY.csv", row.names=FALSE)
# write.csv(testX, file="~/Desktop/stat149/StatKaggle/ensembles/testX.csv", row.names=FALSE)
```

Level 1 Stacking

Logistic Regression

```r
new_train = as.data.frame(trainX)
new_train$lapsed = as.factor(trainY)

stacked1.glm = glm(lapsed ~ ., family=binomial, data=new_train)
summary(stacked1.glm)
```

```
new_test = as.data.frame(testX)
colnames(new_test) = colnames(new_train)[1:9]

stacked1.newpred = predict(stacked1.glm, new_test, type="response")
lapsed = stacked1.newpred

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked1.csv", row.names=FALSE)
```

Random Forest

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
stacked2.rf <- randomForest(lapsed ~ ., data=new_train, ntree=500, do.trace=100)
print(stacked2.rf)
```

```
#Predict Output
predicted = predict(stacked2.rf, new_test, type="prob")
lapsed = predicted[,2]
# Write as submission file
colnames(Id) = "Id"
outputdf = cbind(Id, lapsed)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked2.csv", row.names=FALSE)
```

Neural Net

```
## one hot encoding of response variable
dummies <- dummyVars(lapsed~ ., data = new_train)
lapsed = new_train$lapsed
new_train2 = cbind(lapsed, data.frame(predict(dummies, newdata = new_train)))

temp2 = cbind(Id, new_test)
dummies2 <- dummyVars(Id~ ., data = temp2)
test2 = data.frame(predict(dummies2, newdata = temp2))
```

```
stacked3 <- train(lapsed ~ ., data=new_train2, method='nnet', trace = TRUE,
                maxit=2000, MaxNWts=5000,
                tuneGrid=expand.grid(.size=c(1, 2, 3, 4, 5, 8, 10),
                                     .decay=c(1.0, 0, 0.01, 0.001, 0.1)))
stacked3
```

```
prediction3 <- predict(stacked3, newdata=test2, type="prob")
lapsed = prediction3[,2]
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked3.csv", row.names=FALSE)
```

The two best models were logistic regression, and our tuned Neural Network so far. We tried fitting a Generalized Additive Model as well, and blended in some additional key features from the raw data set as well to see if that affected performance.

```
# reload
train = read.csv("~/Desktop/stat149/StatKaggle/train_add3.csv", header=TRUE)
test = read.csv("~/Desktop/stat149/StatKaggle/test_add3.csv", header=TRUE)

# age, memmonths, hasemail, allgames1yr, r3

new_train_supp = as.data.frame(trainX)
new_train_supp$lapsed = as.factor(trainY)
new_train_supp$age = train$age
new_train_supp$memmonths = train$memmonths
new_train_supp$hasemail = train$hasemail
new_train_supp$allgames1yr = train$allgames1yr
new_train_supp$r3 = train$r3
new_train_supp$r3na = as.factor(train$r3.na)

stacked6.glm = glm(lapsed ~ ., family=binomial, data=new_train_supp)
summary(stacked6.glm)

stacked7.gam = gam(lapsed ~ s(glmpreds) + s(gampreds) + s(rf1preds) + s(rf2preds) + s(nn1preds) + s(nn2

anova(stacked1.glm, stacked6.glm, test="Chi") # not significant
anova(stacked6.glm, stacked7.gam, test="Chi") # significant

stacked9.gam = gam(lapsed ~ s(glmpreds) + s(gampreds) + s(rf1preds) + s(rf2preds) + s(nn1preds) + s(nn2

new_test_supp = as.data.frame(testX)
new_test_supp$age = test$age
new_test_supp$memmonths = test$memmonths
new_test_supp$hasemail = test$hasemail
new_test_supp$allgames1yr = test$allgames1yr
new_test_supp$r3 = test$r3
new_test_supp$r3na = as.factor(test$r3.na)

colnames(new_test_supp) = c(colnames(new_train_supp)[1:9], colnames(new_train_supp)[11:16])

stacked6.newpred = predict(stacked6.glm, new_test_supp, type="response")
lapsed = stacked6.newpred

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked6.csv", row.names=FALSE)

stacked7.newpred = predict(stacked7.gam, new_test_supp, type="response")
lapsed = stacked7.newpred

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked7.csv", row.names=FALSE)
```

```
stacked9.newpred = predict(stacked9.gam, new_test_supp, type="response")
lapsed = stacked9.newpred

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked9.csv", row.names=FALSE)

# save data files
# write.csv(new_train_supp, file="~/Desktop/stat149/StatKaggle/ensembles/stacksupptrain.csv", row.names
# write.csv(new_test_supp, file="~/Desktop/stat149/StatKaggle/ensembles/stacksupptest.csv", row.names=F

# Data Preparation
# Index of columns to standardize
m_age = mean(train$age); sd_age = sd(train$age)
train$age = (train$age - m_age)/sd_age; test$age = (test$age - m_age)/sd_age

m_memmonths = mean(train$memmonths); sd_memmonths = sd(train$memmonths)
train$memmonths = (train$memmonths - m_memmonths)/sd_memmonths; test$memmonths = (test$memmonths - m_mem

m_r3 = mean(train$r3); sd_r3 = sd(train$r3)
train$r3 = (train$r3 - m_r3)/sd_r3; test$r3 = (test$r3 - m_r3)/sd_r3

m_allgames1yr = mean(train$allgames1yr); sd_allgames1yr = sd(train$allgames1yr)
train$allgames1yr = (train$allgames1yr - m_allgames1yr)/sd_allgames1yr; test$allgames1yr = (test$allgame

new_train_supp$age = train$age
new_train_supp$memmonths = train$memmonths
new_train_supp$allgames1yr = train$allgames1yr
new_train_supp$r3 = train$r3

new_test_supp$age = test$age
new_test_supp$memmonths = test$memmonths
new_test_supp$allgames1yr = test$allgames1yr
new_test_supp$r3 = test$r3

## one hot encoding of response variable
dummies <- dummyVars(lapsed~ ., data = new_train_supp)
lapsed = new_train_supp$lapsed
new_train_supp2 = cbind(lapsed, data.frame(predict(dummies, newdata = new_train_supp)))

temp2 = cbind(Id, new_test_supp)
dummies2 <- dummyVars(Id~ ., data = temp2)
test_supp2 = data.frame(predict(dummies2, newdata = temp2))

stacked8 <- train(lapsed ~ ., data=new_train_supp2, method='nnet', trace = TRUE,
                  maxit=2000, MaxNWts=5000,
                  tuneGrid=expand.grid(.size=c(1, 2, 3, 4, 5, 8, 10),
                                       .decay=c(1.0, 0, 0.01, 0.001, 0.1)))
stacked8
```

```
prediction8 <- predict(stacked8, newdata=test_supp2, type="prob")
lapsed = prediction8[,2]
```

```
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="~/Desktop/stat149/StatKaggle/ensembles/finalstacked8
          .csv", row.names=FALSE)
```

The Generalized Additive Model fit to predictions from a range of models, along with `age`, `memmonths`, `hasemail`, `allgames1yr`, and `r3`/`r3.na` data from the original training set added back in again (our earlier investigations suggested these might be important variables), gave us the best result: 0.52925. Although the error on the test set ultimately increased when evaluated on the private test set (0.53861), it was just enough to edge out the other contestants for the win.