# Homework 3

Rachael Latimer, Makayla Whitney

12/1/2021

#Part 1: Predicting a Categorical Outcome using Regularized Logistic Regression For this assignment you will work with the twitter dataset and try to predict the sentiment of a given tweet. I uploaded the datasets on the website for convenience and to make sure everyone works on the same dataset. Download the tweet dataset using the R code below. Then, also create the blueprint as shown below.

```
# Load the following packages needed for modeling in this assignment

  require(caret)
```

```
## Loading required package: caret

## Loading required package: ggplot2

## Loading required package: lattice
```

```
  require(recipes)
```

```
## Loading required package: recipes

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'recipes'

## The following object is masked from 'package:stats':
##
##     step
```

```
  require(ranger)
```

```
## Loading required package: ranger
```

```
## Warning: package 'ranger' was built under R version 4.1.2
```

```
# Import the tweet dataset with embeddings

tweet <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/main/con

# Recipe for the tweet dataset

blueprint_tweet <- recipe(x  = tweet,
                          vars  = colnames(tweet),
                          roles = c('outcome',rep('predictor',772))) %>%
  step_dummy('month',one_hot=TRUE) %>%
  step_harmonic('day',frequency=1,cycle_size=7, role='predictor') %>%
  step_harmonic('date',frequency=1,cycle_size=31,role='predictor') %>%
  step_harmonic('hour',frequency=1,cycle_size=24,role='predictor') %>%
  step_normalize(paste0('Dim',1:768)) %>%
  step_normalize(c('day_sin_1','day_cos_1',
                   'date_sin_1','date_cos_1',
                   'hour_sin_1','hour_cos_1')) %>%
  step_rm(c('day','date','hour')) %>%
  step_num2factor(sentiment,
                  transform = function(x) x + 1,
                  levels=c('Negative','Positive'))
```

##Task 1.1. Split the original data into two subsets: training and test. Let the training data have the 80% of cases and the test data have the 20% of the cases.

```
set.seed(12032021)  # for reproducibility

  loc      <- sample(1:nrow(tweet), round(nrow(tweet) * 0.8))
  tweet_tr  <- tweet[loc, ]
  tweet_te  <- tweet[-loc, ]
```

##Task 1.2. Use the caret::train() function and ranger engine to train a model with 10-fold cross-validation for predicting the probability of sentiment being positive using a Bagged Trees model with 500 trees.

```
folds = cut(seq(1,nrow(tweet_tr)),breaks=10,labels=FALSE)

my.indices <- vector('list',10)
    for(i in 1:10){
      my.indices[[i]] <- which(folds!=i)
    }

cv <- trainControl(method = "cv",
                   index  = my.indices,
                   classProbs = TRUE,
                   summaryFunction = mnLogLoss)
```

```r
grid <- expand.grid(mtry = 772,splitrule='gini',min.node.size=2)
#grid


bags_tweet <- caret::train(blueprint_tweet,
data = tweet_tr,
method = 'ranger',
trControl = cv,
tuneGrid = grid,
metric = 'logLoss',
num.trees = 500,
max.depth = 60)
```

```
## Loading required namespace: e1071
```

##Task 1.3. Use the caret::train() function and ranger engine to train a model with 10-fold cross-validation for predicting the probability of sentiment being positive using a Random Forest model with 500 trees. Set the number of predictors to consider to 250 for each tree while growing a random forest.

```r
grid <- expand.grid(mtry = 250,splitrule='gini',min.node.size=2)
#grid

#takes a few minutes
rfs_tweet <- caret::train(blueprint_tweet,
data = tweet_tr,
method = 'ranger',
trControl = cv,
tuneGrid = grid,
metric = 'logLoss',
num.trees = 500,
max.depth = 60)
```

##Task 1.4 Evaluate the performance of the Bagged Tree models (1.2) and Random Forest Model (1.3) on the test dataset. Calculate and report logLoss (LL), area under the reciever operating characteristic curver (AUC), overall accuracy (ACC), true positive rate (TPR), true negative rate (TNR), and precision (PRE) for three models. When calculating ACC, TPR, TNR, and PRE, assume that we use a cut-off value of 0.5 for the predicted probabilities. Summarize these numbers in a table like the following. Decide and comment on which model you would use to predict sentiment of a tweet moving forward.

```r
ll_bags <- bags_tweet$results$logLoss

ll_rfs <- rfs_tweet$results$logLoss
#checking bagged tree model on the test dataset
tweet_predicted_te_bags <- predict(bags_tweet, tweet_te, type='prob')

tweet_predicted_te_rfs <- predict(rfs_tweet, tweet_te, type='prob')
# Compute the AUC
require(cutpointr)
```

```
## Loading required package: cutpointr

##
## Attaching package: 'cutpointr'
## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
```

```
cut.obj_bags <- cutpointr(x     = tweet_predicted_te_bags$Positive,
                      class = tweet_te$sentiment)
```

```
## Assuming the positive class is 1

## Assuming the positive class has higher x values
```

```
auc_bags<-auc(cut.obj_bags)

cut.obj_rfs <- cutpointr(x     = tweet_predicted_te_rfs$Positive,
                      class = tweet_te$sentiment)
```

```
## Assuming the positive class is 1
## Assuming the positive class has higher x values
```

```
auc_rfs<-auc(cut.obj_rfs)

# Confusion matrix assuming the threshold is 0.5
  #bagged trees
pred_class_bags <- ifelse(tweet_predicted_te_bags$Positive>.5,1,0)

confusion_bags <- table(tweet_te$sentiment,pred_class_bags)

  #random forest
pred_class_rfs <- ifelse(tweet_predicted_te_rfs$Positive>.5,1,0)

confusion_rfs <- table(tweet_te$sentiment,pred_class_rfs)
#Accuracy
  #bagged trees
acc_bags<-(confusion_bags[2,2]+confusion_bags[1,1])/
  (confusion_bags[2,2]+confusion_bags[1,1]+confusion_bags[1,2]+confusion_bags[2,2])
  #random forest
acc_rfs<-(confusion_rfs[2,2]+confusion_rfs[1,1])/
  (confusion_rfs[2,2]+confusion_rfs[1,1]+confusion_rfs[1,2]+confusion_rfs[2,2])

# True Negative Rate
  #bagged trees
tnr_bags<-confusion_bags[1,1]/(confusion_bags[1,1]+confusion_bags[1,2])
  #random forest
tnr_rfs<-confusion_rfs[1,1]/(confusion_rfs[1,1]+confusion_rfs[1,2])
# True Positive Rate
  #bagged trees
tpr_bags<-confusion_bags[2,2]/(confusion_bags[2,1]+confusion_bags[2,2])
  #random forest
tpr_rfs<-confusion_rfs[2,2]/(confusion_rfs[2,1]+confusion_rfs[2,2])
# Precision
  #bagged trees
pre_bags<-confusion_bags[2,2]/(confusion_bags[1,2]+confusion_bags[2,2])
```

```r
  #random forest
pre_rfs<-confusion_rfs[2,2]/(confusion_rfs[1,2]+confusion_rfs[2,2])

#Table prep

bag <- data.frame(Model = c("Bagged Trees"),
                    LL = c(ll_bags),
                    AUC = c(auc_bags),
                    ACC = c(acc_bags),
                    TPR = c(tpr_bags),
                    TNR = c(tnr_bags),
                    PRE = c(pre_bags))

rfs <- data.frame(Model = c("Random Forests Model"),
                    LL = c(ll_rfs),
                    AUC = c(auc_rfs),
                    ACC = c(acc_rfs),
                    TPR = c(tpr_rfs),
                    TNR = c(tnr_rfs),
                    PRE = c(pre_rfs))

#Table
Table1 <- rbind(bag, rfs)
Table1
```

```
##                  Model        LL       AUC       ACC       TPR       TNR
## 1         Bagged Trees 0.5147783 0.8645036 0.6182796 0.7465753 0.7857143
## 2 Random Forests Model 0.5347516 0.8726650 0.6116751 0.8219178 0.7857143
##         PRE
## 1 0.7676056
## 2 0.7843137
```

```r
#The Random Forests model has higher AUC (.87 compared to .86), so using this to
#predict tweet sentiment moving forward.
```

##Task 1.5 Compare the performance of the Bagged Trees Model and Random Forest Model in this assignment to the performance of logistic regression models from the previous assignment. Comment on what you observe. Did Bagged Trees or Random Forest Models perform better than Logistic Regression Models?

The conclusion from the previous assignment was the Logistic Regression with the Ridge Penalty performed the best of the Logistic Regression models. The Logistic Regression with Ridge Penalty also performed better than the Bagged Trees or Random Forest Models with an AUC of .91 and and ACC of .78.

#Part 2: Predicting a Continous Outcome using Regularized Linear Regression For this assignment you will work with the Oregon dataset and try to predict the scores.I uploaded the datasets on the website for convenience and to make sure everyone works on the same dataset. Download the Oregon testing dataset using the R code below. Then, also create the blueprint as shown below.

```r
# Load the following packages needed for modeling in this assignment

  require(caret)
  require(recipes)
  require(ranger)
```

```r
# Import the oregon dataset

oregon <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/main/co

# Recipe for the oregon dataset

  outcome <- 'score'

  id      <- 'id'

  categorical <- c('sex','ethnic_cd','tst_bnch','migrant_ed_fg','ind_ed_fg',
                   'sp_ed_fg','tag_ed_fg','econ_dsvntg','stay_in_dist',
                   'stay_in_schl','dist_sped','trgt_assist_fg',
                   'ayp_dist_partic','ayp_schl_partic','ayp_dist_prfrm',
                   'ayp_schl_prfrm','rc_dist_partic','rc_schl_partic',
                   'rc_dist_prfrm','rc_schl_prfrm','grp_rpt_dist_partic',
                   'grp_rpt_schl_partic','grp_rpt_dist_prfrm',
                   'grp_rpt_schl_prfrm')

  numeric <- c('enrl_grd')

  cyclic <- c('date','month')


blueprint_oregon <- recipe(x     = oregon,
                    vars  = c(outcome,categorical,numeric,cyclic),
                    roles = c('outcome',rep('predictor',27))) %>%
  step_indicate_na(all_of(categorical),all_of(numeric)) %>%
  step_zv(all_numeric()) %>%
  step_impute_mean(all_of(numeric)) %>%
  step_impute_mode(all_of(categorical)) %>%
  step_harmonic('date',frequency=1,cycle_size=31,role='predictor') %>%
  step_harmonic('month',frequency=1,cycle_size=12,role='predictor') %>%
  step_ns('enrl_grd',deg_free=3) %>%
  step_normalize(c(paste0(numeric,'_ns_1'),paste0(numeric,'_ns_2'),paste0(numeric,'_ns_3'))) %>%
  step_normalize(c("date_sin_1","date_cos_1","month_sin_1","month_cos_1")) %>%
  step_dummy(all_of(categorical),one_hot=TRUE) %>%
  step_rm(c('date','month'))

  #View(blueprint_oregon %>% prep() %>% summary)
```

##Task 2.1. Split the original data into two subsets: training and test. Let the training data have the 80% of cases and the test data have the 20% of the cases.

```r
set.seed(12012021)

loc      <- sample(1:nrow(oregon), round(nrow(oregon) * 0.8))
ordata_train  <- oregon[loc, ]
ordata_test  <- oregon[-loc, ]
```

##Task 2.2. Use the caret::train() function and ranger engine to train a model with 10-fold cross-validation for predicting the scores using a Bagged Trees model with 500 trees.

```
# Cross validation settings
ordata_train = ordata_train[sample(nrow(ordata_train)),]

folds = cut(seq(1,nrow(ordata_train)),breaks=10,labels=FALSE)

my.indices <- vector('list',10)
    for(i in 1:10){
      my.indices[[i]] <- which(folds!=i)
    }

cv <- trainControl(method = "none",
                   index  = my.indices)
#running without cross-validation due to time issues

grid <- expand.grid(mtry = 27,splitrule='variance',min.node.size=2)
#grid

or_baggedtrees <- caret::train(blueprint_oregon,
                               data      = ordata_train,
                               method    = 'ranger',
                               trControl = cv,
                               tuneGrid  = grid,
                               num.trees = 500,
                               max.depth = 60)
```

```
## Growing trees.. Progress: 11%. Estimated remaining time: 4 minutes, 0 seconds.
## Growing trees.. Progress: 24%. Estimated remaining time: 3 minutes, 14 seconds.
## Growing trees.. Progress: 37%. Estimated remaining time: 2 minutes, 42 seconds.
## Growing trees.. Progress: 49%. Estimated remaining time: 2 minutes, 10 seconds.
## Growing trees.. Progress: 62%. Estimated remaining time: 1 minute, 37 seconds.
## Growing trees.. Progress: 74%. Estimated remaining time: 1 minute, 6 seconds.
## Growing trees.. Progress: 87%. Estimated remaining time: 33 seconds.
## Growing trees.. Progress: 99%. Estimated remaining time: 2 seconds.
```

```
or_baggedtrees$times
```

```
## $everything
##    user  system elapsed
## 1972.35    2.73  261.05
##
## $final
##    user  system elapsed
## 1970.74    2.48  259.19
##
## $prediction
## [1] NA NA NA
```

##Task 2.3. Use the caret::train() function and ranger engine to train a model with 10-fold cross-validation for predicting the scores using a Random Forest model with 500 trees. Set the number of predictors to consider to 250 for each tree while growing a random forest.

```r
# Cross validation settings
ordata_train = ordata_train[sample(nrow(ordata_train)),]

folds = cut(seq(1,nrow(ordata_train)),breaks=10,labels=FALSE)

my.indices <- vector('list',10)
    for(i in 1:10){
      my.indices[[i]] <- which(folds!=i)
    }

cv <- trainControl(method = "none",
                       index  = my.indices)
#running without cross-validation due to time issues

rgrid <- expand.grid(mtry = 250,splitrule='variance',min.node.size=2)
#rgrid

or_rforest <- caret::train(blueprint_oregon,
                        data      = ordata_train,
                        method    = 'ranger',
                        trControl = cv,
                        tuneGrid  = grid,
                        num.trees = 500,
                        max.depth = 60)
```

```
## Growing trees.. Progress: 11%. Estimated remaining time: 4 minutes, 0 seconds.
## Growing trees.. Progress: 24%. Estimated remaining time: 3 minutes, 12 seconds.
## Growing trees.. Progress: 37%. Estimated remaining time: 2 minutes, 37 seconds.
## Growing trees.. Progress: 50%. Estimated remaining time: 2 minutes, 5 seconds.
## Growing trees.. Progress: 63%. Estimated remaining time: 1 minute, 33 seconds.
## Growing trees.. Progress: 76%. Estimated remaining time: 1 minute, 0 seconds.
## Growing trees.. Progress: 88%. Estimated remaining time: 29 seconds.
```

```r
or_rforest$times
```

```
## $everything
##    user  system elapsed
## 1930.79    3.13  256.14
##
## $final
##    user  system elapsed
## 1929.45    2.78  254.45
##
## $prediction
## [1] NA NA NA
```

##Task 2.4 Evaluate the performance of the Bagged Tree models (2.2) and Random Forest Model (2.3) on the test dataset. Calculate and report the root mean squared error (RMSE), mean absolute error (MAE), and R-square.

```r
#checking bagged tree model on the test dataset
ordata_predict_test <- predict(or_baggedtrees, ordata_test)

#calculate RMSE for bagged tree model
bag_rmse <- sqrt(mean((ordata_test$score - ordata_predict_test)^2))
bag_rmse
```

```
## [1] 90.54204
```

```r
#90.54204

#calculate MAE for bagged tree model
orpredicted_te <- predict(or_baggedtrees, ordata_test)

#MAE
bag_mae <- mean(abs(ordata_test$score - orpredicted_te))
bag_mae
```

```
## [1] 70.07822
```

```r
#70.07822

#calculate rsqd for bagged tree model
bag_rsqd <- cor(ordata_test$score,orpredicted_te)^2
bag_rsqd
```

```
## [1] 0.398061
```

```r
#0.398061



#checking random forest model on the test dataset
rpredict_te <- predict(or_rforest,ordata_test)

# RMSE for random forest model
ran_rmse <- sqrt(mean((ordata_test$score - rpredict_te)^2))
ran_rmse
```

```
## [1] 90.54163
```

```r
#90.53776

#calculate MAE for random tree model
test_predict <- predict(or_baggedtrees, ordata_test)

#MAE
ran_mae <- mean(abs(ordata_test$score - test_predict))
bag_mae
```

```
## [1] 70.07822
```

```
#70.07822

#calculate rsqd for bagged tree model
ran_rsqd <- cor(ordata_test$score, test_predict)^2
bag_rsqd
```

```
## [1] 0.398061
```

```
#0.398061
```

##Task 2.4.2 Summarize these numbers in a table like the following.

```
bagmod <- data.frame(Model = c("Bagged Trees Model"),
                    RMSE = c(bag_rmse),
                     MAE = c(bag_mae),
                     Rsq = c(bag_rsqd))

ranmod <- data.frame(Model = c("Random Forest Model"),
                    RMSE = c(ran_rmse),
                     MAE = c(ran_mae),
                     Rsq = c(ran_rsqd))

#Final Table
SumTable <- rbind(bagmod, ranmod)
SumTable
```

```
##                   Model     RMSE      MAE      Rsq
## 1  Bagged Trees Model 90.54204 70.07822 0.398061
## 2 Random Forest Model 90.54163 70.07822 0.398061
```

##Task 2.5 Compare the performance of the Bagged Trees Model and Random Forest Model in this assignment to the performance of linear regression models from the previous assignment. Comment on what you observe. Did Bagged Trees or Random Forest Models perform better than Linear Regression Models in predicting the test scores?

The Bagged Trees and Random Forest Models both performed with higher RMSE and MAE scores, but lower R-squared scores. Due to the higher RMSE scores, I would argue that the models from this assignment performed poorly compared to the previous assignment's models. I would still choose a linear regression model with no penalty to predict scores.