Appendix

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE, eval = FALSE)
indent1 = '    '
indent2 = paste(rep(indent1, 2), collapse='')
indent3 = paste(rep(indent1, 3), collapse='')

library(knitr)
library(tidyverse)
library(ggmap)
library(maps)
library(Rtsne)
library(NbClust)
library(tree)
library(randomForest)
library(maptree)
library(class)
library(reshape2)
library(ggplot2)
library(kableExtra)
library(dplyr)
library(glmnet)
library(ROCR)
library(cluster)
library(plotmo)
library(gbm)
```

Data

```
## read data and convert candidate from string to factor
election.raw <- read_delim("/Users/laurenwong/Downloads/data/election/election.csv",
  delim = ",") %>%
  mutate(candidate=as.factor(candidate))

census_meta <- read_delim("/Users/laurenwong/Downloads/data/census/metadata.csv",
  delim = ";", col_names = FALSE)
census <- read_delim("/Users/laurenwong/Downloads/data/census/census.csv",
  delim = ",")
```

Election Data

```
kable(election.raw %>%
  filter(county == "Los Angeles County")) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
    full_width=FALSE)
```

```
##Subsetting Election (county-level)
election.raw <- subset(election.raw, !election.raw$fips==2000)
dim(election.raw)
```

Census Data

```
kable(census %>% head, "html")  %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
  full_width=FALSE) %>%
  scroll_box(width = "100%")
```

```
kable(census_meta)
```

Data wrangling

```
#federal level
election_federal <- filter(election.raw, is.na(county) & fips =="US")
election_federal
```

```
#state level State-level summary rows have names of each states as fips value.
election_state <- filter(election.raw, fips != "US" & is.na(county) & fips != "DC" &
  as.character(election.raw$fips) == as.character(election.raw$state))
election_state
```

```
#county level
election <- filter(election.raw, !is.na(county))
election
```

6

```
num.candidate<- unique(election.raw$candidate)
length(num.candidate)
num.candidate
```

```
#bar plot
ggplot(election.raw, aes(x=candidate, y=(votes))) +
  geom_bar(stat="identity",fill="lightblue") +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  coord_flip()
```

Visualization

```
# create county_winner variable
county_winner <- election %>%
group_by(fips) %>%
mutate(total=sum(votes), pct = votes/total) %>%
top_n(1)
# create state_winner variable
state_winner <- election_state %>%
group_by(fips) %>%
mutate(total=sum(votes), pct = votes/total) %>%
top_n(1)
```

```
states = map_data("state")
ggplot(data = states) +
geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +
coord_fixed(1.3)+guides(fill=FALSE)+ggtitle("State-Level Map")
```

8

```
counties <- map_data("county")
ggplot(data = counties) +
  geom_polygon(aes(x = long, y = lat, fill = subregion, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  ggtitle("County-Level Map")
```

9

```
states = map_data("state")
states = mutate(states, fips = state.abb[match(states[,5],tolower(state.name))])
states <- left_join(state_winner,states,by=c("state"="fips"))
```

```
ggplot(data=states) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=F) +
  ggtitle("Winning Candidate by State")
```

10

```
county.fips <- maps::county.fips %>%
  separate(polyname, c("region", "subregion"), sep=",") %>%
  mutate(fips=as.factor(fips))
join_first=left_join(counties, county.fips, by=c("region","subregion"))
join_counties=left_join(join_first, county_winner, by=c("fips"))
```

```
ggplot(data=join_counties) +
  geom_polygon(aes(x = long, y = lat, fill = candidate,
  group = group), color = "white") +
  coord_fixed(1.3) +
  ggtitle("Winning Candidate per County")
```

11

```
#Mutate data to make the Minority column
census_new <- census %>%
  dplyr::mutate (Minority = Hispanic + Black + Native + Asian + Pacific)
#will mutate this data more for number 11
```

```r
#group by white people
census_white <- aggregate(census_new$White, by=list(census_new$State),
    FUN = sum, na.rm = TRUE)
#group by minority
census_min <- aggregate(census_new$Minority, by=list(census_new$State),
    FUN = sum, na.rm = TRUE)

#plot for white
plot_white <- ggplot(census_white, aes(x=census_white$Group.1,y=census_white$x)) +
  geom_point(color= "maroon", size=0.7) +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  labs(x="States", y="White")

#plot for minority
plot_min <- ggplot(census_min, aes(x=census_min$Group.1,y=census_min$x)) +
  geom_point(color= "navy", size=0.7) +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  labs(x="States", y="Minority")

plot_white
plot_min
```

12

```r
#clean census data
census.del <- census[2:37] %>%
  na.exclude(census) %>%
  mutate(Men = Men/TotalPop*100,
         Employed = Employed/TotalPop*100,
         Citizen = Citizen/TotalPop*100,
         Minority = Hispanic + Black + Native + Asian + Pacific) %>%
  dplyr::select(-Walk, -PublicWork, -Construction, -Hispanic,
  -Black, -Native, -Asian, -Pacific, -Women)
```

```r
#sub-county
census.subct <- census.del %>%
  group_by(State,County) %>%
  add_tally(TotalPop) %>%
  mutate(CountyTotal = n, Weight = TotalPop/CountyTotal) %>%
  select(-n) %>%
  ungroup
```

```r
#county
census.ct <- census.subct %>%
  group_by(State,County) %>%
  summarise_at(vars(Men:CountyTotal), funs(weighted.mean))
head(census.ct)
```

Dimensionality reduction

13

```
#county-level
pc1 <- prcomp(census.ct[3:28], scale=TRUE, center=TRUE)
#county has only 26 PCs
dim(pc1$x)

#sub-county level
pc2 <- prcomp(census.subct[3:30], scale = TRUE, center = TRUE)
#sub-county has only 28 PCs
dim(pc2$x)

#convert to dataframe
ct.pc <- data.frame(pc1$x[,1:2])
subct.pc <- data.frame(pc2$x[,1:2])

#rotation matrix
ct_rotation <- data.frame(pc1$rotation[,1])
abs(ct_rotation)
ct_rotation

subct_rotation <- data.frame(pc2$rotation[,1])
abs(subct_rotation)
subct_rotation
```

14

```
#PC1 uses census.ct for counties
pr.var1 <- pc1$sd^2 #variance explained by each principal component
pve1 <- pr.var1 / sum(pr.var1) #proportion of variance explained by each principal component
cumulative_pve1 <- cumsum(pve1)

#PC2 uses census.subct for subcounties
pr.var2 <- pc2$sd^2 #variance explained by each principal component
pve2 <- pr.var2 / sum(pr.var2) #proportion of variance explained by each principal component
cumulative_pve2 <- cumsum(pve2)

par(mfrow=c(2, 2)) #graphs side by side

#plot of county-level
plot(pve1, type="l", lwd=3, xlab="Principal Component 1", ylab="Counties PVE")
plot(cumulative_pve1, type="l", xlab="Principal Component 1", ylab="County Cumulative
PVE", ylim=c(0, 1), lwd=3)

#plot of sub-county level
plot(pve2, type="l", lwd=3, xlab="Principal Component 2", ylab="Subcounties PVE")
plot(cumulative_pve2, type="l", xlab="Principal Component 2", ylab="Subcounties Cumulative
PVE", ylim=c(0, 1), lwd=3)
```

```
#minimum number of PCs needed for 90% of variance
PC1.Num <- which(cumulative_pve1>=0.9)[1]
#PC1.Num #county
PC2.Num <- which(cumulative_pve2>=0.9)[1]
#PC2.Num #sub-county
```

Clustering

15

```
census.ct_clust <- scale(census.ct[3:28])
census.ct_dist <- dist(census.ct_clust, method = "euclidean")
census.ct_hclust <- hclust(census.ct_dist, method = "complete")
clust1 <- cutree(census.ct_hclust, k = 10)
table(clust1)

#reclust with 2 principal components
ct.pc <- data.frame(pc1$x[,1:2])
ct.pc_clust <- scale(ct.pc)
ct.pc_dist <- dist(ct.pc_clust, method = "euclidean")
ct.pc_hclust <- hclust(ct.pc_dist, method = "complete")
clust2 <- cutree(ct.pc_hclust, k = 10)
table(clust2)

#San Mateo
clust1[which(census.ct$County == "San Mateo")]
clust2[which(census.ct$County == "San Mateo")]
```

Classification

```
tmpwinner <- county_winner %>%
  ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%               ## state abbreviations
  mutate_at(vars(state, county), tolower) %>%                           ## to all lowercase
  mutate(county = gsub(" county| columbia| city| parish", "", county))  ## remove suffixes
tmpcensus <- census.ct %>%
  dplyr::ungroup() %>%
  mutate_at(vars(State, County), tolower)

election.cl <- tmpwinner %>%
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit

## save meta information
election.meta <- election.cl %>%
  select(c(county, fips, state, votes, pct, total))

## save predictors and class labels
```

```
election.cl = election.cl %>%
  select(-c(county, fips, state, votes, pct, total))
```

```
set.seed(10)
n <- nrow(election.cl)
in.trn <- sample.int(n, 0.8*n)
trn.cl <- election.cl[ in.trn,]
tst.cl <- election.cl[-in.trn,]
```

```
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(trn.cl), breaks = nfold, labels = FALSE))
```

```
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow = 3, ncol = 2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

16

```
#training error
trn.clX = trn.cl %>% select(-candidate)
trn.clY = trn.cl$candidate
#test error
tst.clX = tst.cl %>% select(-candidate)
tst.clY = tst.cl$candidate

tree_params <- tree.control(nrow(trn.cl))
#using a 10 fold CV to select tree which minimizes cv misclassification error.
model_tree <- tree(as.factor(candidate) ~ ., data = trn.cl, control = tree_params)
set.seed(1)
cv <- cv.tree(model_tree, folds, FUN = prune.misclass, K = 10)
tree_select <- which(cv$dev == min(cv$dev))

#select the smallest tree size w/ that minimum rate
best_size <- min(cv$size[tree_select]) #best size is 9

#visualize the tree before pruning
draw.tree(model_tree, nodeinfo=TRUE, cex=0.50)
title("Unpruned Tree")
```

```
#visualize the tree after pruning
#prune the tree to the size found and plot with draw.tree
pruned_tree <- prune.tree(model_tree, best = best_size,method="misclass")
#summary(pruned_tree)
draw.tree(pruned_tree, nodeinfo=TRUE, cex=0.50)
title("Pruned Tree")

# traning error
pred_pruned <- predict(pruned_tree, trn.clX, type = "class")
pruned_error <- calc_error_rate(pred_pruned, trn.clY)
```

```r
# test error
pred_tr <- predict(pruned_tree, tst.clX, type = "class")
error_tr <- calc_error_rate(pred_tr, tst.clY)
```

```r
# putting errors into records
records[1,1] <- pruned_error
records[1,2] <- error_tr
kable(records)
```

17

```r
glm_fit <- glm(factor(candidate)~., data = trn.cl, family = "binomial")
summary(glm_fit)
```

```r
set.seed(1)
#glm train
glm_train <- predict(glm_fit, newdata = trn.cl, type = "response")
winner_train <- factor(ifelse(glm_train < 0.5, "Donald Trump", "Hillary Clinton"),
    levels=c("Donald Trump", "Hillary Clinton"))
result <- factor(ifelse(trn.cl$candidate == "Donald Trump","Donald Trump", "Hillary Clinton"))
table(predicted = winner_train, true = result)

#glm test
glm_test <- predict(glm_fit, newdata = tst.cl, type = "response")
winner_test <- factor(ifelse(glm_test < 0.5, "Donald Trump", "Hillary Clinton"),
    levels=c("Donald Trump", "Hillary Clinton"))
result2 <- factor(ifelse(tst.cl$candidate == "Donald Trump","Donald Trump", "Hillary Clinton"))
table(predicted = winner_test, true = result2)
```

```r
#train and test errors
glm_train_error <- calc_error_rate(winner_train, result)
glm_test_error <- calc_error_rate(winner_test, result2)
#print matrix
records[2,1] <- glm_train_error
records[2,2] <- glm_test_error
kable(records)
```

18

```r
y.trn <- ifelse(trn.cl[,1] == "Donald Trump", 0, 1)
x.trn <- model.matrix(candidate~. , trn.cl)[,-1]

set.seed(1)
cv_lasso <- cv.glmnet(lambda = c(1, 5, 10, 50) * 1e-4, x.trn, y.trn, foldid = folds,
    alpha =1,family = "binomial")
plot(cv_lasso)
bestlambda <- cv_lasso$lambda.min #1e-04
abline(v = log(bestlambda), col="navy", lwd = 3, lty = 2)

lasso_mod <- glmnet(x.trn, y.trn, alpha = 1, family = "binomial")
coeff <- predict(lasso_mod, type = "coefficients", s = bestlambda)
```

```
plot.glmnet(lasso_mod, xvar="lambda", label = TRUE)

x.tst = model.matrix(candidate~., tst.cl)[,-1]

set.seed(1)
lasso_train_pred = predict(lasso_mod, newx = x.trn, s = bestlambda)
lasso_train = ifelse(lasso_train_pred < 0.5, "Donald Trump","Hillary Clinton")
las_train_err = calc_error_rate(as.tibble(lasso_train), trn.cl[,1])

lasso_test_pred = predict(lasso_mod, newx = x.tst, s = bestlambda)
lasso_test = ifelse(lasso_test_pred < 0.5, "Donald Trump","Hillary Clinton")
las_test_err = calc_error_rate(as.tibble(lasso_test), tst.cl[,1])

#non-zero coeff
#as.matrix(coeff)
```

```
#matrix
records[3,1] <- las_train_err
records[3,2] <- las_test_err
kable(records)
```

19

```
#mutating candidates 0 = Hillary, 1 = Trump
trn.m = trn.cl %>%
  mutate(candidate = as.factor(ifelse(candidate == 0, "Hillary Clinton", "Donald Trump")))
```

```
#decision tree
pred_tree <- prediction(as.numeric(pred_pruned), as.numeric(trn.cl$candidate))
perf_tree <- performance(pred_tree, measure = 'tpr', x.measure = 'fpr')
plot(perf_tree, col = "maroon", lwd = 3, main = "ROC Curves")
abline(0,1)

#logistic
pred_log <- prediction(as.numeric(glm_train), as.numeric(trn.cl$candidate))
perf_log <- performance(pred_log, measure = 'tpr', x.measure = 'fpr')
plot(perf_log, add = TRUE, col = "forestgreen", lwd = 9)
abline(0,1)

#lasso
pred_lasso <- prediction(as.numeric(glm_train), as.numeric(trn.cl$candidate))
perf_lasso <- performance(pred_log, measure = 'tpr', x.measure = 'fpr')
plot(perf_lasso, add = TRUE, col = "lightblue", lwd = 3)
abline(0,1)
legend("bottomright", legend = c("decision tree", "log", "lasso"),
    col = c("maroon","forestgreen", "lightblue"), lty = 1, cex = 0.7)
```

```
auc_tree <- performance(pred_tree, "auc")@y.values #0.8297726
auc_tree
auc_log <- performance(pred_log, "auc")@y.values  #0.9528419
auc_log
auc_lasso <- performance(pred_lasso, "auc")@y.values  #0.9528419
```

```
auc_lasso
```

Taking it further

20

```
set.seed(1)
#Trump = 0, Clinton = 1
true_test <- as.numeric(ifelse(tst.cl$candidate == "Donald Trump", 0,1))
boost.elect.cl <- gbm(ifelse(candidate == "Donald Trump", 0,1)~., data = trn.cl,
    distribution = "bernoulli", n.trees = 800)
#summary(boost.elect.cl, main = "Boosting Election.cl")

par(mfrow = c(1,2))
plot(boost.elect.cl, i = "Minority", ylab= "y(Minority)")
```

```
plot(boost.elect.cl, i = "SelfEmployed", ylab= "y(SelfEmployed)")

yhat.boost <- predict(boost.elect.cl, newdata = tst.cl, n.trees = 800, type = "response")
```

```
#confusion matrix
boost.error <- table(pred = yhat.boost, truth = true_test)
test.boost.error <- 1 - sum(diag(boost.error))/sum(boost.error) #0.9983713
record1 <- matrix(c(test.boost.error, test.boost.error), nrow = 1, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boosting")
kable(record1)
```

```
set.seed(1)
trn.cl$candidate <- factor(trn.cl$candidate)
bag.elect.cl <- randomForest(candidate~., data = trn.cl, mtry = 10, importance = TRUE)

plot(bag.elect.cl)
legend("center", colnames(bag.elect.cl$err.rate), col = 1:4, cex = 0.8, fill = 1:4)

bag.elect.cl <- randomForest(candidate~., data = trn.cl, mtry = 10, ntree = 700,
    importance = TRUE)
yhat.bag <- predict(bag.elect.cl, newdata = tst.cl)

#confusion matrix
bag.error <- table(pred = yhat.bag, truth = true_test)
test.bag.error <- 1 - sum(diag(bag.error))/sum(bag.error)   #0.04885993
record1 <- matrix(c(test.boost.error, test.bag.error), nrow = 2, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boost error","bag error")
kable(record1)
```

```
set.seed(1)
options(stringsAsFactors = FALSE)
true_test <- as.numeric(ifelse(tst.cl$candidate == "Donald Trump", 0,1))

#glimpse(election) #18,007 observations, 5 variables
```

```r
#change candidate to factor
trn.cl$candidate <- factor(trn.cl$candidate)
rf.election <- randomForest(candidate~., data = trn.cl, mtry = 3, ntree = 1000,
    importance = TRUE)
plot(rf.election)

yhat.rf <- predict(rf.election, newdata = tst.cl)

#importance(rf.election)
varImpPlot(rf.election, sort = TRUE,
    main = "Variable Importance for Random Forest Election", n.var = 5)
```

```r
#tree, log reg, and lasso records
kable(records)

#create matrix
rf.error <- table(pred = yhat.rf, truth = true_test)
test.rf.error <- 1 - sum(diag(rf.error))/sum(rf.error)  #0.04885993

record1 <- matrix(c(test.boost.error, test.bag.error, test.rf.error), nrow = 3, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boosting", "bagging","random forest")
kable(record1)
```