Problem Set 4

Political Data Science - Spring 2020

Gencer, Alper Sukru

Due May 1, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered within an R script. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded.
2. Work on git. Fork the repository found at https://github.com/domlockett/PDS-PS3 and add your code, committing and pushing frequently. Use meaningful commit messages – these may affect your grade.
3. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. If you have any questions regarding the Problem Set, contact the TAs or use their office hours.
5. For students new to programming, this may take a while. Get started.
6. You will need to install ggplot2 and dplyr to complete this dataset.

Question 1 - Sample Statistics

Load the following data: http://politicaldatascience.com/PDS/Datasets/GSS-data.csv.

The variable poleff11 asks participants to rate their level of agreement with the statement "People like me don't have any say about what the government does" (see the codebook for more information on all variables in this dataset at: http://politicaldatascience.com/PDS/Datasets/gss_codebook.csv).

1. Convert this variable into a numeric where higher values indicate higher levels of political efficacy (1- strongly agrees with the statement; 5- strongly disagrees with the statment) and all other values ('Cant choose' etc.) become NA's.
2. What is the proportion of individuals from the entire sample who feel as though they "have a say in the government?"
3. Using a sample of 25 from this dataset. What is the average proportion who feel as thought hey have a say?
4. Pull a random sample of 25 from the poleff11 data and calculate the mean for this outcome. Now repeat this process 500 times and store these values in a variable called trials 25.
5. Now create a variable called trials 100 where we do 500 trials with $n = 100$ instead of 25.
6. Draw a histogram of the sampling distribution for the two trials ($n = 25$ vs. $n = 100$) you just conducted. Give the plots meaningful titles and axis labels. Save these plots in your repository.
7. What notable difference occur when we use a larger sample size in our trials?

Answer 1 - Sample Statistics

Load the following data: http://politicaldatascience.com/PDS/Datasets/GSS-data.csv.

```
rm(list = ls())
gss.data <- read.csv("http://politicaldatascience.com/PDS/Datasets/GSS-data.csv")
gss.data <- gss.data[-c(2349, 2350), ] ## included some unrelated information
```

Convert this variable into a numeric where higher values indicate higher levels of political efficacy (1- strongly agrees with the statement; 5- strongly disagrees with the statment) and all other values ('Cant choose' etc.) become NA's:

```
levels(gss.data$poleff11)
```

```
## [1] ""                           "Agree"
## [3] "Cant choose"                "Disagree"
## [5] "Neither agree nor disagree" "No answer"
## [7] "Not applicable"             "Strongly agree"
## [9] "Strongly disagree"
```

```
gss.data$poleff11.recoded[gss.data$poleff11 == "Strongly agree"] <- 1
gss.data$poleff11.recoded[gss.data$poleff11 == "Agree"] <- 2
gss.data$poleff11.recoded[gss.data$poleff11 == "Neither agree nor disagree"] <- 3
gss.data$poleff11.recoded[gss.data$poleff11 == "Disagree"] <- 4
gss.data$poleff11.recoded[gss.data$poleff11 == "Strongly disagree"] <- 5
gss.data$poleff11.recoded[gss.data$poleff11 == "Cant choose"] <- NA
gss.data$poleff11.recoded[gss.data$poleff11 == "Not applicable"] <- NA
gss.data$poleff11.recoded[gss.data$poleff11 == "No answer"] <- NA
```

What is the proportion of individuals from the entire sample who feel as though they "have a say in the government?":

```
########  For this, I'll be looking at the proportion of people who "disagree (4) \ strongly disagree (

#---  including NA values:
prop_have.a.say.na <-sum(table(gss.data$poleff11.recoded)[4:5])/nrow((gss.data))
prop_have.a.say.na
```

```
## [1] 0.1937819
```

```
#---  excluding NA values:
prop_have.a.say <-sum(table(gss.data$poleff11.recoded)[4:5])/length(na.omit(gss.data$poleff11.recoded))
prop_have.a.say
```

```
## [1] 0.3953084
```

Using a sample of 25 from this dataset. What is the average proportion who feel as thought hey have a say?:

```r
set.seed(20200425)

#---  including NA values:
sample25_na <-sample(gss.data$poleff11.recoded, size =25)
sum(sample25_na>=4, na.rm = T)/25
```

```
## [1] 0.24
```

```r
#---  excluding NA values:
sample25_no.na <-sample(na.omit(gss.data$poleff11.recoded), size =25)
sum(sample25_no.na>=4, na.rm = T)/25
```

```
## [1] 0.36
```

\subsection{Pull a random sample of 25 from the poleff11 data and calculate the mean for this outcome. Now repeat this process 500 times and store these values in a variable called \textbf{trials_25}:}

```r
########  For convenience, I will be solving by removing NA's.

#---  Random one sample:
mean(sample(na.omit(gss.data$poleff11.recoded), size =25, replace = T))
```

```
## [1] 2.6
```

```r
#---  Random 500 samples:
trials_25 <- NULL
for(i in 1:500){
  trials_25 <- c(trials_25, mean(sample(na.omit(gss.data$poleff11.recoded), size =25, replace = T)))
}
```

\subsection{Now create a variable called trials_100 where we do 500 trials with n=100 instead of 25:}

```r
########  For convenience, I will be solving by removing NA's.

#---  Random one sample with n = 100:
mean(sample(na.omit(gss.data$poleff11.recoded), size =100, replace = T))
```

```
## [1] 2.92
```

```r
#---  Random 500 samples with n = 100:
trials_100 <- NULL
for(i in 1:500){
  trials_100 <- c(trials_100, mean(sample(na.omit(gss.data$poleff11.recoded), size =100, replace = T)))
}
```

Draw a histogram of the sampling distribution for the two trials (n=25 vs. n=100) you just conducted. Give the plots meaningful titles and axis labels. Save these plots in your repository.:

```r
library(tidyverse)
##install.packages("ggpubr")
library(ggpubr)

df.trials <- as.data.frame(cbind(trials_25, trials_100))
```

```r
p_trials_100 <- ggplot(df.trials) +
  geom_histogram(aes(trials_100), bins = 100, fill = "red") +
  xlab("Sample Means") +
  ggtitle("Sampling Dist. with Size 100") +
  ylim(0,50) +
  xlim(2,4) +
  theme_light()

p_trials_25 <- ggplot(df.trials) +
  geom_histogram(aes(trials_25), bins = 100, fill = "blue") +
  xlab("Sample Means") +
  ggtitle("Sampling Dist. with Size 25") +
  ylim(0,50) +
  xlim(2,4) +
  theme_light()

ggarrange(p_trials_100, p_trials_25,
                    ncol = 2, nrow = 1)
```

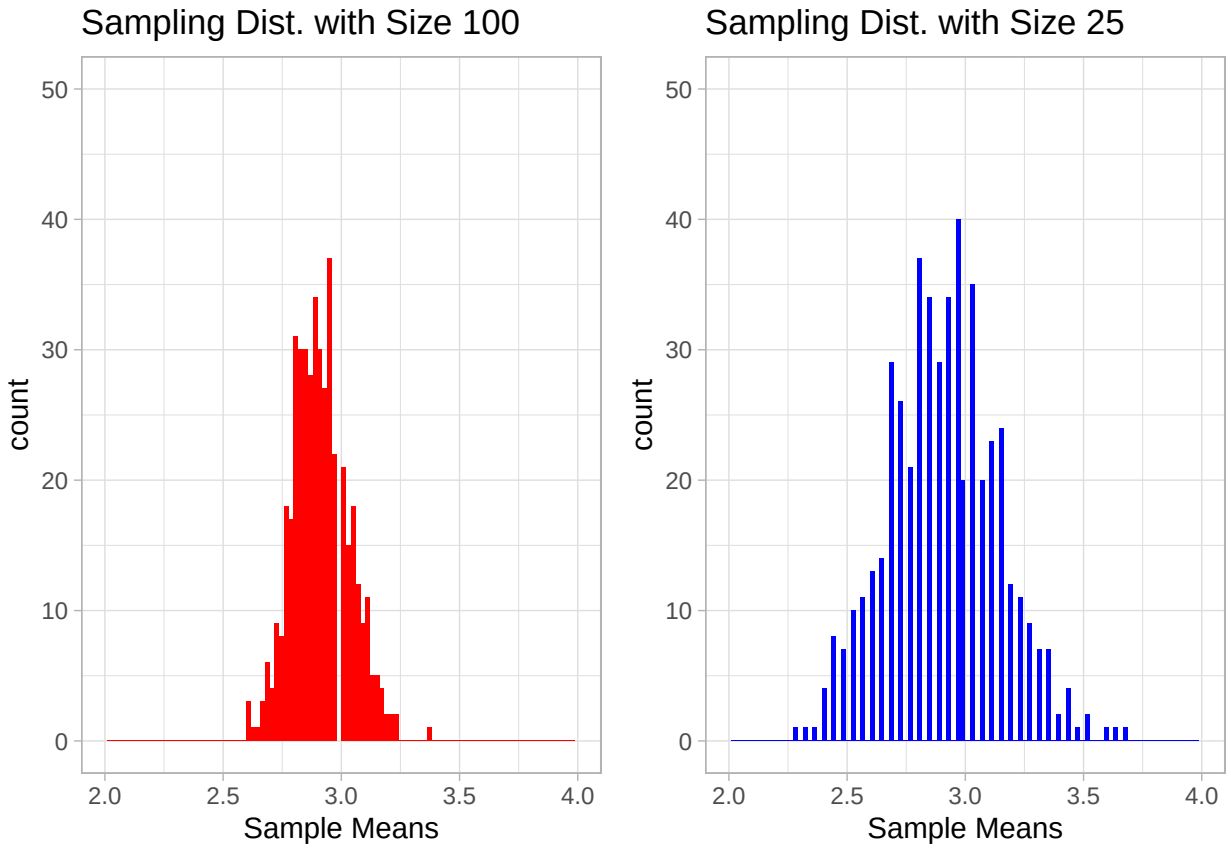


Figure 1: Sampling Distributions with Varying Sample Sizes

What notable difference occur when we use a larger sample size in our trials?:

As it can be seen in the graph, the spead of the sampling distribution with sample size 100 is narrower than the one with sample size 25.

```
library(knitr)

table <- cbind(mean(na.omit(gss.data$poleff11.recoded)), mean(trials_100),
      sd(trials_100), mean(trials_25), sd(trials_25))

colnames(table) <- c("True Mean", "Size-100 Mean", "Size-100 SD",
                  "Size-25 Mean", "Size-25 SD")

kable(table)
```

True Mean	Size-100 Mean	Size-100 SD	Size-25 Mean	Size-25 SD
2.908775	2.91692	0.1183373	2.90648	0.2374067

Also see from the table that the mean of size-100 trials is closer to the true mean and at the same time, the sampling distribution of sample mean has less variance (smaller standard error).

Question 2 - Supervised Learning

Load the following data: http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateData92-16.csv. This is is data for incumbents running for re-election to the US Senate.

```
rm(list = ls())
poll.data <- read.csv("http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateDataS
```

- Poll Percentage.of.Vote.won.x is the percentage of the vote the candidate won.
- The other variabels are mostly self-explanatory or have been used before in class.
- However, this datset differes in that it is organized at the poll level. That is, there is one row for each poll of each senate race.
- So there are some new variables including: the polling firm, the starting date of the poll, the "days left" until Eleciton Day, sample size, and the numberSupport (the number of respondents in that poll who indicated they supported the incumbent candidate.)
- There is also a win variable that indicates whether the incumbent candidate won the election.

Re-organize the data so it is a the election level (as opposed to the poll level):

- This means you will have to figure out how to reduce the polling data into a summary statistic.
- You might try to do this a couple of different ways based on sample size and date of the poll for use later.

```
########  Here, I will create a variable based on the existing poll:
####        1) Total Number Support divided by Total Sample Size and

poll.data.2 <- poll.data %>%
  group_by(Candidateidentifier) %>%
  mutate(aveg_support =  (sum(numberSupport) / sum(samplesize))*100)

########  Now, get rid of repeated polls and keep only incumbents:

poll.data.2 <- poll.data.2 %>%
  group_by(Candidateidentifier) %>%
  filter(row_number(Percentage.of.Vote.won.x) == 1) %>%
  filter(Incumbent == 1) %>%
  select(-c("pollster", "numberSupport", "samplesize", "poll_period", "daysLeft"))
```

Randomly select 20 percent of your data to use as a "validation sample" to assess the quality of your model. You will use this division of the data in the rest of the problems below:

```
########  20 percent of data as validation (test) data:

library(rsample)

#---  See the following courses chunk below:
```

Using the Poll Percentage.of.Vote.won.x variable, create at least two linear regression models to predict vote share for incumbents:

- You are free to do this any way you want, but you must assess the quality of your model using cross-validation.
- Train your model on your "training" data (80
- Provide an appropriate summary statistic for your compeing models using only the validation set. (Meaning: what is your out-of-sample performance?)

```
########  Linear Baseline Models:
####

rmse0 <- c()        ####  Model 0   Percentage.of.Vote.won.x ~ Democrat
rmse1 <- c()        ####  Model 1   Percentage.of.Vote.won.x ~ Democrat + aveg_support
rmse2 <- c()        ####  Model 2   Percentage.of.Vote.won.x ~ Democrat + pvi
rmse3 <- c()        ####  Model 3   Percentage.of.Vote.won.x ~ Democrat + cycle
rmse4 <- c()        ####  Model 4   Percentage.of.Vote.won.x ~ Democrat + weightexperience



####
####  Model 0   Percentage.of.Vote.won.x ~ Democrat
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse0 <- c(rmse0, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse0)
```

```
## [1] 8.399232
```

```
####
####  Model 1   Percentage.of.Vote.won.x ~ Democrat + aveg_support
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + aveg_support, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse1 <- c(rmse1, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse1)
```

```
## [1] 3.925801
```

```
####
####  Model 2   Percentage.of.Vote.won.x ~ Democrat + pvi
####
```

```
for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + pvi, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse2 <- c(rmse2, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse2)
```

```
## [1] 8.412185
```

```
####
####  Model 3   Percentage.of.Vote.won.x ~ Democrat + cycle
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + factor(cycle), data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse3 <- c(rmse3, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse3)
```

```
## [1] 8.74203
```

```
####
####  Model 4   Percentage.of.Vote.won.x ~ Democrat + weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + weightexperience, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse4 <- c(rmse4, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse4)
```

```
## [1] 8.329421
```

```
df.1 <- data.frame(cbind(mean(rmse0), mean(rmse1), mean(rmse2), mean(rmse3), mean(rmse4)))
colnames(df.1) <- c("Baseline (BL)", "BL + Aveg. Support", "BL + pvi", "BL + cycle", "BL + weightexperi
kable(df.1)
```

Baseline (BL)	BL + Aveg. Support	BL + pvi	BL + cycle	BL + weightexperience
8.399233	3.925801	8.412185	8.74203	8.329421

See that the models with 1) Average Support, 2) pvi, 3) Weighted Experience, and 4) Statesare the models we are to focus on.

```
########  Linear Baseline Models:
####

rmse6 <- c()        ####  Model 6   Percentage.of.Vote.won.x ~ Democrat + poly(aveg_support, 2)
rmse7 <- c()        ####  Model 7   Percentage.of.Vote.won.x ~ Democrat * aveg_support
rmse8 <- c()        ####  Model 8   Percentage.of.Vote.won.x ~ Democrat + aveg_support + weightexperien
rmse9 <- c()        ####  Model 9   Percentage.of.Vote.won.x ~ Democrat * aveg_support + pvi
rmse11<- c()        ####  Model 10  Percentage.of.Vote.won.x ~ Democrat + Incumbent * weightexperience




####
####  Model 6   Percentage.of.Vote.won.x ~ Democrat + poly(aveg_support, 2)
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + poly(aveg_support, 2) , data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse6 <- c(rmse6, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse6) # See that it is almost same with model without poly. So discard this model!
```

```
## [1] 3.917937
```

```
####
####  Model 7   Percentage.of.Vote.won.x ~ Democrat * aveg_support
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat * aveg_support, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse7 <- c(rmse7, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse7) # See that it is worse than the baseline model without interaction.
```

```
## [1] 3.979696
```

```
            # So discard this model!



####
####  ####  Model 8   Percentage.of.Vote.won.x ~ Democrat + aveg_support + weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
```

```
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + aveg_support + weightexperience , data = elect_trai
  pred <- predict(model, newdata = elect_test)
  rmse8 <- c(rmse8, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse8) # See that it is worse than the baseline model only democrat and aveg_support.
```

```
## [1] 3.973098
```

```
            # So discard this model!


####
####  ####  Model 9   Percentage.of.Vote.won.x ~ Democrat + aveg_support + pvi
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat * aveg_support + pvi, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse9 <- c(rmse9, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse9) # See that it is worse than the baseline model only democrat and aveg_support.
```

```
## [1] 4.003991
```

```
            # So discard this model!



df.2 <- data.frame(cbind(mean(rmse1), mean(rmse6), mean(rmse7), mean(rmse8), mean(rmse9)))
colnames(df.2) <- c("BL", "Poly aveg_support", "Int. aveg_support", "aveg_support + weight",
                    "aveg_support + pvi")
kable(df.2)
```

BL	Poly aveg_support	Int. aveg_support	aveg_support + weight	aveg_support + pvi
3.925801	3.917937	3.979696	3.973098	4.003991

As it can be seen above, the smallest root mean square errors belongs to the additive model of democrat and average support.

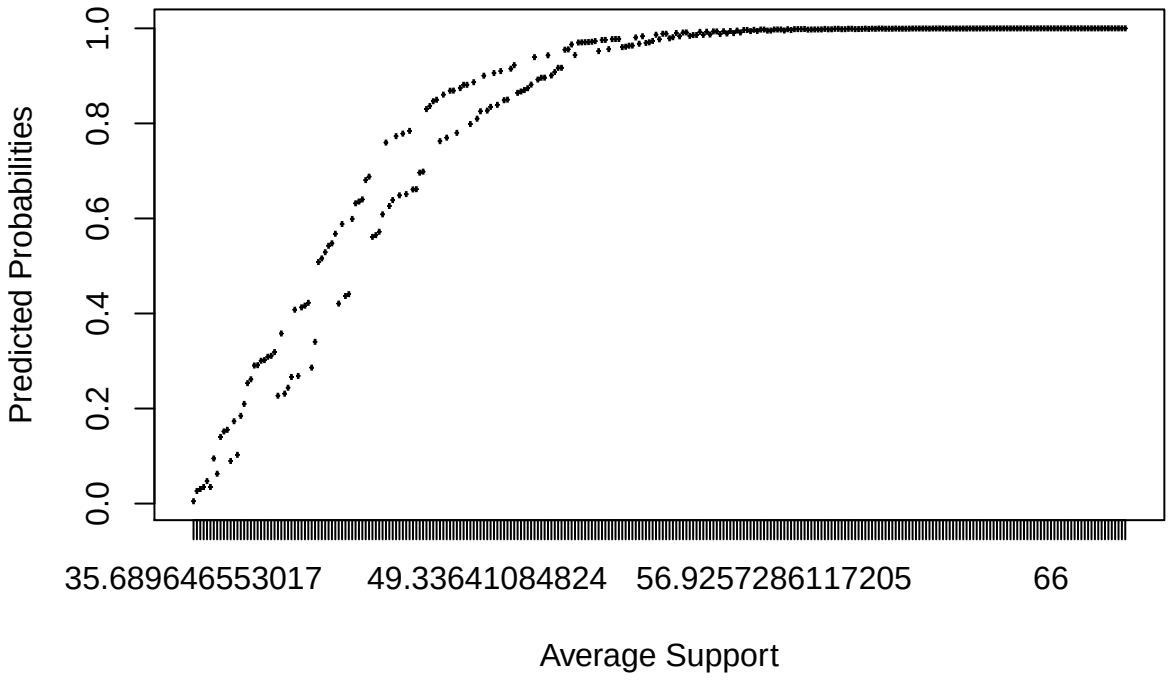
Now, using the win variable as your outcome, create at least 3 classification models. You should again assess each model on your "validation" set using appropriate methods. You must fit at least one of each:

- linear classifier,
- random forest model,
- K-nearest neighbors.

```
####
########  LINEAR CLASSIFIER
####

Model1 <- glm(win ~ Democrat + aveg_support, family="binomial", data=poll.data.2)
summary(Model1)
```

```
##
## Call:
## glm(formula = win ~ Democrat + aveg_support, family = "binomial",
##     data = poll.data.2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.85812   0.00779   0.05875   0.28235   1.87224
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -22.64320    3.62068  -6.254 4.00e-10 ***
## Democrat      -0.63934    0.48891  -1.308    0.191
## aveg_support   0.50362    0.07894   6.380 1.77e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.18  on 278  degrees of freedom
## Residual deviance: 120.32  on 276  degrees of freedom
## AIC: 126.32
##
## Number of Fisher Scoring iterations: 7
```

```
########  Within sample prediction (Fitted Values)
####
Model1preds <- predict(Model1, type="response") ##  Type = response assures that the outcome is squashe

########  Let's look at the predicted probabilities for each value of Weighted Experience in the datase
####
boxplot(Model1preds ~ poll.data.2$aveg_support, xlab="Average Support", ylab="Predicted Probabilities")
```




```
binaryPred <- (Model1preds > 0.5) * 1
table <- table(binaryPred, poll.data.2$win)
nrow(poll.data.2)
```

```
## [1] 279
```

```
precision <- table[2,2] / (table[2,1] + table[2,2])
precision
```

```
## [1] 0.9205021
```

```
recall <- table[2,2] / (table[2,2] + table[1,2])
recall
```

```
## [1] 0.952381
```

```
accuracy <- (table[2,2] + table[1,1]) / (nrow(poll.data.2))
accuracy
```

```
## [1] 0.8924731
```

```
(as.data.frame(cbind(c("precision", "recall", "accuracy"), c(precision, recall, accuracy))))
```

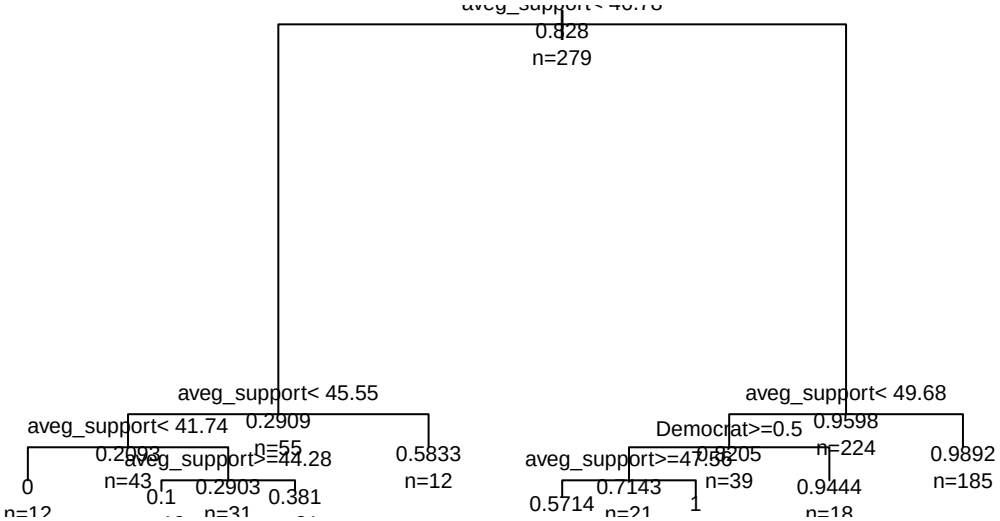
```
##          V1                V2
## 1 precision 0.920502092050209
## 2    recall 0.952380952380952
## 3  accuracy  0.89247311827957
```

```
####
########  TREE MODELS
```

```
####


##install.packages("rpart")
library(rpart)

equation <- as.formula("win ~ Democrat + aveg_support")
tree_mod1 <- rpart(equation, data = poll.data.2)

########  Let's look at our tree:
####
tree_mod1
```

```
## n= 279
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 279 39.7419400 0.8279570
##    2) aveg_support< 46.77782 55 11.3454500 0.2909091
##      4) aveg_support< 45.54963 43  7.1162790 0.2093023
##        8) aveg_support< 41.74285 12  0.0000000 0.0000000 *
##        9) aveg_support>=41.74285 31  6.3870970 0.2903226
##         18) aveg_support>=44.27713 10  0.9000000 0.1000000 *
##         19) aveg_support< 44.27713 21  4.9523810 0.3809524 *
##      5) aveg_support>=45.54963 12  2.9166670 0.5833333 *
##    3) aveg_support>=46.77782 224  8.6383930 0.9598214
##      6) aveg_support< 49.6791 39  5.7435900 0.8205128
##       12) Democrat>=0.5 21  4.2857140 0.7142857
##         24) aveg_support>=47.55931 14  3.4285710 0.5714286 *
##         25) aveg_support< 47.55931 7  0.0000000 1.0000000 *
##       13) Democrat< 0.5 18  0.9444444 0.9444444 *
##      7) aveg_support>=49.6791 185  1.9783780 0.9891892 *
```

```
plot(tree_mod1)
text(tree_mod1, use.n = T, all = T, cex = 0.7)
```




```
########  Let's check for accuracy meansures:
####

#---    TREE 1:
treePreds1 <- predict(tree_mod1)
table.tree.1 <- table((treePreds1>=0.5)*1, poll.data.2$win)
accuracy.tree.1 <- round((table.tree.1[1,1] + table.tree.1[2,2]) / nrow(poll.data.2),4)
brier.tree.1 <- round(sqrt(sum((poll.data.2$win - treePreds1)^2)/nrow(poll.data.2)),4)
kable(as.data.frame(cbind(c("Accuracy", "Brier Score"), c(accuracy.tree.1, brier.tree.1))))
```

V1	V2
Accuracy	0.9176
Brier Score	0.2328

```
#---    TREE 2:
tree_mod2 <- rpart(equation, data=poll.data.2, control=rpart.control(cp=.02))
treePreds2 <- predict(tree_mod2)
table.tree.2 <- table((treePreds2>=0.5)*1, poll.data.2$win)
accuracy.tree.2 <- round((table.tree.2[1,1] + table.tree.2[2,2]) / nrow(poll.data.2),4)
brier.tree.2 <- round(sqrt(sum((poll.data.2$win - treePreds2)^2)/nrow(poll.data.2)),4)
kable(as.data.frame(cbind(c("Accuracy", "Brier Score"), c(accuracy.tree.2, brier.tree.2))))
```

V1	V2
Accuracy	0.9176

V1	V2
Brier Score	0.2523

```
#---    TREE 3:
tree_mod3 <- rpart(equation, data=poll.data.2, control=rpart.control(cp=0.0001))
treePreds3 <- predict(tree_mod3)
table.tree.3 <- table((treePreds3>=0.5)*1, poll.data.2$win)
accuracy.tree.3 <- round((table.tree.3[1,1] + table.tree.3[2,2]) / nrow(poll.data.2),4)
brier.tree.3 <- round(sqrt(sum((poll.data.2$win - treePreds3)^2)/nrow(poll.data.2)),4)
kable(as.data.frame(cbind(c("Accuracy", "Brier Score"), c(accuracy.tree.3, brier.tree.3))))
```

V1	V2
Accuracy	0.9211
Brier Score	0.2279

```
########  Let's compare scores:
####

kable(as.data.frame(cbind(c("Accuracy", "Brier Score"), c(accuracy.tree.1, brier.tree.1),
                    c(accuracy.tree.2, brier.tree.2), c(accuracy.tree.3, brier.tree.3))))
```

V1	V2	V3	V4
Accuracy	0.9176	0.9176	0.9211
Brier Score	0.2328	0.2523	0.2279

```
#---  Brier score of the third model is smallest!



####
########  RANDOM FOREST MODEL
####

library(randomForest)

poll.data.2$win.factor <- as.factor(poll.data.2$win)
equation.2 <- as.formula("win.factor ~ Democrat + aveg_support")


mod1_forest <- randomForest(equation.2, data=poll.data.2, ntree=600, mtry=2, maxnodes = 5)
mod1_forest # This confusion matrix is "out of bag"
```

```
##
## Call:
##  randomForest(formula = equation.2, data = poll.data.2, ntree = 600,      mtry = 2, maxnodes = 5)
##                Type of random forest: classification
##                      Number of trees: 600
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 10.75%
```

```
## Confusion matrix:
##    0   1 class.error
## 0 32  16  0.33333333
## 1 14 217  0.06060606
```

```
########    Let's check the accuracy:
####
pred.forest.1 <- predict(mod1_forest)
table(pred.forest.1, poll.data.2$win)
```

```
##
## pred.forest.1   0   1
##             0  32  14
##             1  16 217
```

```
####
########  K-NEAREST NEIGHBORS
####



library(class)

########  Creating Control Matrix
####
poll.data.2.X <- poll.data.2[,c("win", "Democrat", "aveg_support")]
poll.data.2.X$win <- as.numeric(poll.data.2.X$win) - 1
########  Creating DV Vector
####
poll.data.2.X$win <- (as.numeric(poll.data.2.X$win) + rnorm(length(poll.data.2.X$win), 0, .001))

########  Run on Test Matrix
####
mod1_knn <- knn(poll.data.2.X, test=poll.data.2.X, cl=poll.data.2$win, k=10)
table(mod1_knn, poll.data.2$win)
```

```
##
## mod1_knn   0   1
##        0  41   4
##        1   7 227
```

Now you are going to assess your classifiers using the 2018 election.:

- Most of the data you need is here: http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateData1
- BUT, this dataset is missing (a) the final outcome and (b) a lot of the polling data.
- Scrape the election results and polls from ballotpedia.org.
- This does not need to be perfect, but should demonstrate the basic skills covered on webscraping.
- Assess how each of your classifiers performs for 2018 using appropriate metrics.

```
rm(list = ls())
gss.data <- read.csv("http://politicaldatascience.com/PDS/Datasets/GSS-data.csv")
gss.data <- gss.data[-c(2349, 2350), ] ## included some unrelated information

gss.data.18 <- read.csv("http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateDa
```