2016-12-14

FINAL PROJECT

CS634 DATA MINING

PROFESSOR SENJUTI BASU ROY

LIN TANG

31400221

CONTENT

Problem Definition **2**

Instructions **2**

  Platform 2

  Packages 2

  Data 2

Problem 1 .... **3**

Problem 2 .... **6**

Problem 3 .... **9**

Problem 4 .... **12**

Source R code **16**

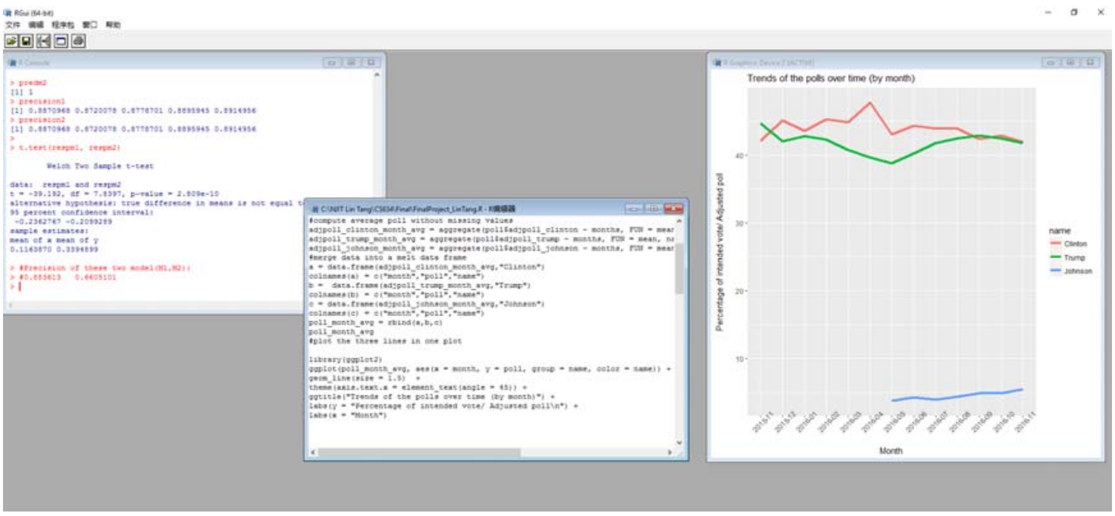
Problem Definition

1. What are the trends of the polls over time (by month).
2. Build two different models (M1 and M2) to predict who is going to be the likely winner.
3. Using the given dataset to augment the election results with ground truth and perform 5-fold cross validation.
4. Based on 3, perform 5-fold cross validation and compute precision of M1 and M2. Does one of the model better than the other with 5% significance level?

Instructions

Platform

We use R language to build the program and software RGui run the program. This program need to crawl the data from the csv file and clean the data, then use the data to solve these four problems.



Packages

Here are the packages using in the program:
library(ggplot2)
library(randomForest)

Data

Data Source: presidential_polls.csv

Some Variables Explanation:

M1	Model 1, Random forest
M2	Model 2, logistic regression
pollnum	The columns that we choose in the csv file
outcome	The output when the function finished
trData	Train data
vaData	Validation data

Problem 1

What are the trends of the polls over time (by month)

Preprocessing steps

To describe the trends of the polls, we use adjusted polls (The “adj” value reflects calibration for historical statistical bias of the individual polls which is perfectly sound and standard) of the three persons: Clinton, Trump and Johnson. We ignore Mcmullin because his data always missing.

We use the field created_date as the date of the polls, but note that the raw data is not in correct format, it should be in MM-DD-YY format, but some observations have YY in MM place, for example 2011/1/16 should be 2016-01-11, so we need to first clean the date.

Algorithm

After clean the date data, as we plot the month trend, we also need to aggregate data in month, we use **average adjusted polls.**

The outline of preprocess of date and poll in this section is:

1. Clean the date which format is YY/MM/DD (remove "20" at the beginning of the dates if any)
2. Extract months (Unique year-month combinations)
3. Use months to compute month average of polls for each of three candidates
4. Plot the three time series in one plot

Code Screenshot

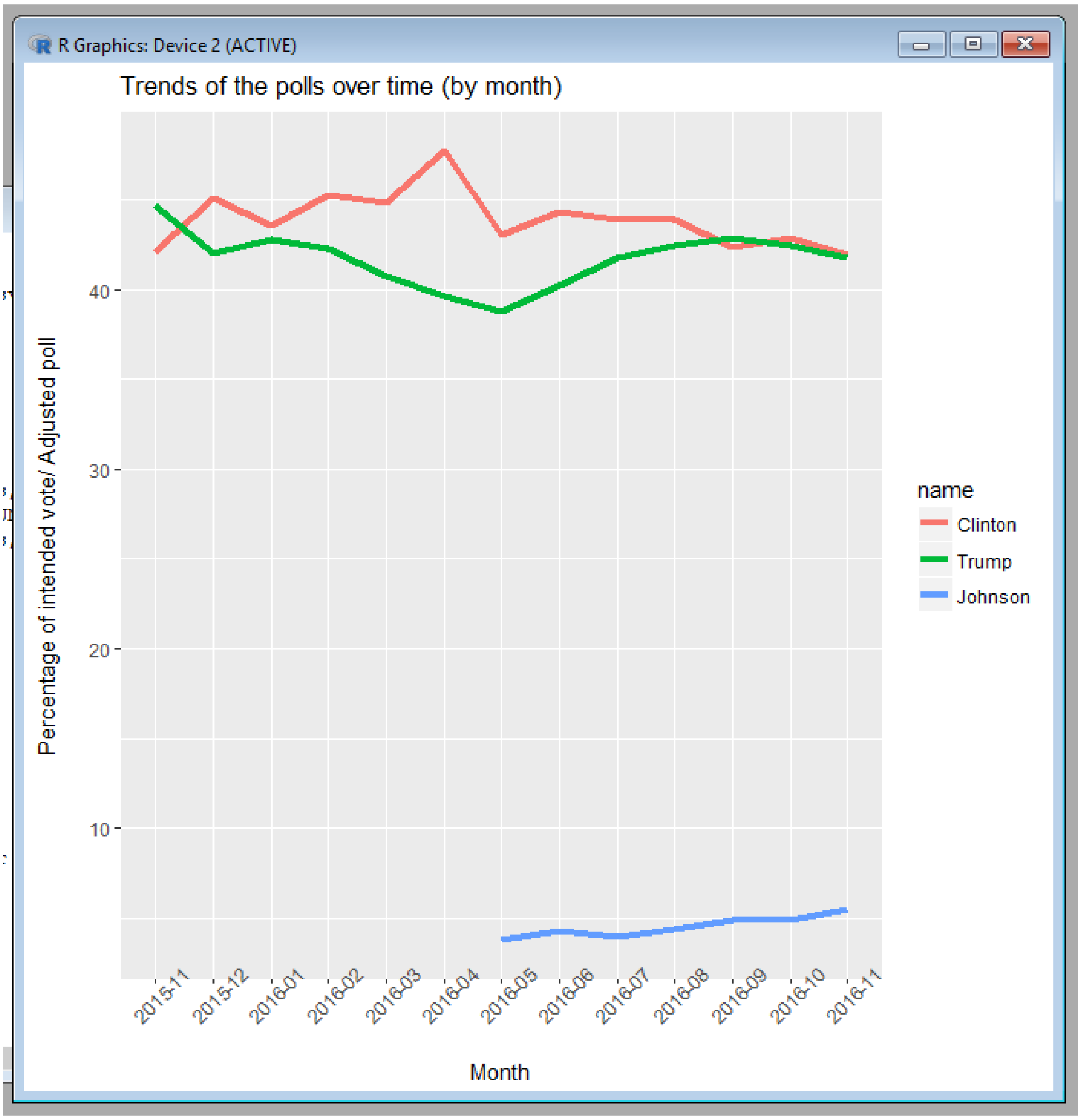
C:\NJIT Lin Tang\CS634\Final\FinalProject_LinTang.R - R编辑器

```r
#Problem 1
#read the csv file
poll = read.csv("C:/NJIT Lin Tang/CS634/Final/presidential_polls.csv")
#Remove the "20" at the beginning of date if any
poll$createddate = gsub("^20","",as.character(poll$createddate))
#specify date format
poll$createddate = as.Date(poll$createddate, format = "%m/%d/%y")
#extract months
months = strftime(poll$createddate, "%Y-%m")
#compute average poll without missing values
adjpoll_clinton_month_avg = aggregate(poll$adjpoll_clinton ~ months, FUN = mean, na.rm = TRUE, data = poll)
adjpoll_trump_month_avg = aggregate(poll$adjpoll_trump ~ months, FUN = mean, na.rm = TRUE, data = poll)
adjpoll_johnson_month_avg = aggregate(poll$adjpoll_johnson ~ months, FUN = mean, na.rm = TRUE, data = poll)
#merge data into a melt data frame
a = data.frame(adjpoll_clinton_month_avg,"Clinton")
colnames(a) = c("month","poll","name")
b =  data.frame(adjpoll_trump_month_avg,"Trump")
colnames(b) = c("month","poll","name")
c = data.frame(adjpoll_johnson_month_avg,"Johnson")
colnames(c) = c("month","poll","name")
poll_month_avg = rbind(a,b,c)
poll_month_avg
#plot the three lines in one plot

library(ggplot2)
ggplot(poll_month_avg, aes(x = month, y = poll, group = name, color = name)) +
geom_line(size = 1.5)  +
theme(axis.text.x = element_text(angle = 45)) +
ggtitle("Trends of the polls over time (by month)") +
labs(y = "Percentage of intended vote/ Adjusted poll\n") +
labs(x = "Month")
```

Result

```
      month      poll    name
1  2015-11 42.088709 Clinton
2  2015-12 45.100835 Clinton
3  2016-01 43.532300 Clinton
4  2016-02 45.210689 Clinton
5  2016-03 44.846171 Clinton
6  2016-04 47.691723 Clinton
7  2016-05 42.989256 Clinton
8  2016-06 44.348386 Clinton
9  2016-07 43.893398 Clinton
10 2016-08 43.858259 Clinton
11 2016-09 42.343710 Clinton
12 2016-10 42.886844 Clinton
13 2016-11 41.958401 Clinton
14 2015-11 44.653257   Trump
15 2015-12 41.984720   Trump
16 2016-01 42.761798   Trump
17 2016-02 42.228436   Trump
18 2016-03 40.703941   Trump
19 2016-04 39.640849   Trump
20 2016-05 38.789773   Trump
21 2016-06 40.174495   Trump
22 2016-07 41.740653   Trump
23 2016-08 42.471355   Trump
24 2016-09 42.821015   Trump
25 2016-10 42.433762   Trump
26 2016-11 41.776364   Trump
27 2016-05  3.803746 Johnson
28 2016-06  4.328492 Johnson
29 2016-07  3.952220 Johnson
30 2016-08  4.421517 Johnson
31 2016-09  4.869042 Johnson
32 2016-10  4.891761 Johnson
33 2016-11  5.488467 Johnson
```



Problem 2

Predict who is going to be the likely winner

Preprocessing step

As our task is to predict who is likely to be the winner, we only need to predict who would be the person has most votes. And it is obviously that Johnson is not likely to be the winner, so we do not consider him in our model. Thus, our problem is to predict **a binary outcome**. So we transform the regression problem to a classification problem. Here we create two models to do the prediction. The models are random forest and logistic regression.

Algorithm

Model 1: Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operated by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Here is the code template of Random decision forest in R language:

Random Forests

Random forests improve predictive accuracy by generating a large number of bootstrapped trees (based on random samples of variables), classifying a case using each tree in this new "forest", and deciding a final predicted outcome by combining the results across all of the trees (an average in regression, a majority vote in classification). Breiman and Cutler's random forest approach is implimented via the randomForest package.

Here is an example.

```
# Random Forest prediction of Kyphosis data
library(randomForest)
fit <- randomForest(Kyphosis ~ Age + Number + Start,   data=kyphosis)
print(fit) # view results
importance(fit) # importance of each predictor
```

Link: http://www.statmethods.net/advstats/cart.html

Model 2: Logistic Regression

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is useful because it can take any real input, whereas the output always takes values between zero and one and hence is interpretable as a probability. The logistic function is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Let us assume that t is a linear function of a single explanatory variable x (the case where t is a linear combination of multiple explanatory variables is treated similarly). We can then express t as follows:

$$t = \beta_0 + \beta_1 x$$

And the logistic function can now be written as:

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Here is the code template of Logistic regression in R language:

Logistic Regression

Logistic regression is useful when you are predicting a binary outcome from a set of continuous predictor variables. It is frequently preferred over discriminant function analysis because of its less restrictive assumptions.

```
# Logistic Regression
# where F is a binary factor and
# x1-x3 are continuous predictors
fit <- glm(F~x1+x2+x3,data=mydata,family=binomial())
summary(fit) # display results
confint(fit) # 95% CI for the coefficients
exp(coef(fit)) # exponentiated coefficients
exp(confint(fit)) # 95% CI for exponentiated coefficients
predict(fit, type="response") # predicted values
residuals(fit, type="deviance") # residuals
```

You can use **anova(***fit1*,*fit2*, **test="Chisq")** to compare nested models. Additionally, **cdplot(***F*~*x*, **data=***mydata*) will display the conditional density plot of the binary outcome *F* on the continuous *x* variable.

Link: http://www.statmethods.net/advstats/glm.html

Predictors

We first encode 1 if Clinton is the winner, and 0 if Trump is the winner. Thus, we have a binary outcome. And we will use the meaningful predictors, we will give up to use predictor: cycle, branch, url or fields like them, as they are constant all the same not helpful for our predictions. Also, we do not choose those predictors which have too many levels like pollster, state. The predictors we use in the program are:

1. Poll_wt
2. Grade
3. Population
4. Samplesize

5. Diffdays
6. Collecteddays

The predictor **diffdays** is the days between startdate and enddate.

The predictor **collecteddays** is the days from the beginning day, because we can see time has effect on the votes (The later days' poll weight are higher than the old days' poll weight. Many of the older surveys have a poll weight of pretty much zero, meaning they are not being used at all in estimating current predictions). Trump seems has a rising trend in the later days.

Partial Code Screenshot

```
#Problem 2/3/4
library(randomForest)
poll$outcome = ifelse(poll$adjpoll_clinton > poll$adjpoll_trump, 1, 0)
poll$diffdays = as.Date(poll$enddate, format = "%m/%d/%y") - as.Date(poll$startdate, format = "%m/%d/%y")
poll$collecteddays = poll$createddate - min(poll$createddate)
pollnum = na.omit(poll[,c(9,10,11,12,13,28,29,30)])
#M1 = randomForest(factor(outcome) ~  grade + samplesize + population + poll_wt + diffdays + collecteddays, data = pollnum)
#M2 = glm(factor(outcome) ~  grade + samplesize + population + poll_wt + diffdays + collecteddays, data = pollnum, family = binomial)
```

```
pred1 = c(pred1, predict(m1Train,vaData, type="class"))
```

```
pred2 = c(pred2, ifelse(predict(m2Train,vaData, type="response") > 0.5,1,0))
```

```
predictm1 = ifelse(mean(pred1) > 0.5,1,0)
predictm2 = ifelse(mean(pred2) > 0.5,1,0)
```

```
#Who is the winner of Random forests  1:Clinton wins, 0: Trump wins:
predictm1
#Who is the winner of Logistic regression  1:Clinton wins, 0: Trump wins:
predictm2
```

Result

```
> #Who is the winner of Random forests  1:Clinton wins, 0: Trump wins:
> predictm1
[1] 1
> #Who is the winner of Logistic regression  1:Clinton wins, 0: Trump wins:
> predictm2
[1] 1
```

So, both of these two models predict Clinton is the winner.

Problem 3

Augment the result with ground truth and perform 5-fold cross validation

Preprocessing step

As we do not know the result of this election now, all of our data is training data, we actually do not have testing data. So we need to perform cross validation, using part of our training data as testing data (validation data).

Follow the following steps to perform the 5-fold cross validation:

- Split the training data set into equally 5-folds
- Use one of the folds as validation data and all of the other 4 folds as training data
- Train our two models on the training data and test on the validation data, record the results
- Combine all the results into one result and compare it with the true outcome (outcome field in whole data set) and do it for two models.
- Compute the error rate.

Algorithm

We use this equation to calculate the error rate:

$$error\ rate = \frac{FP + FN}{P + N}.$$

Code in R:

```
tbm2 = table(ifelse(predict(m2Train,vaD
print(tbm2)
tbm4 = as.matrix(tbm2)
TP2 <- tbm4[1, 1]
FP2 <- tbm4[2, 1]
FN2 <- tbm4[1, 2]
TN2 <- tbm4[2, 2]
P2 <- TP2 + FN2
N2 <- FP2 + TN2
error_rate2 <- (FP2 + FN2) / (P2 + N2)
```

Ground truth

As we mentioned in problem 2, we just choose Clinton and Trump as the competitors, and **we first encode 1 if Clinton is the winner, and 0 if Trump is the winner. Thus, we have a binary outcome. So, if Clinton is the winner, the output is 1, otherwise the output is 0. In the other words, the outcome is 0 or 1.**

1 fold size

```
> print(dim(pollSub[[1]]))
[1] 2046    8
```

Partial Code Screenshot

```
#####Random forests Model
tbm1 = table(predict(m1Train,vaData, type="class"), vaData$outcome)
print(tbm1)
tbm3 = as.matrix(tbm1)
TP1 <- tbm3[1, 1]
FP1 <- tbm3[2, 1]
FN1 <- tbm3[1, 2]
TN1 <- tbm3[2, 2]
P1 <- TP1 + FN1
N1 <- FP1 + TN1
accuracy1 <- (TP1 + TN1) / (P1 + N1)
error_rate1<- (FP1 + FN1) / (P1 + N1)
precision1 <- TP1 / (TP1 + FP1)
recall1 <- TP1 / P1
error1_vec<-c(error1_vec,error_rate1)
accuracy1_vec<-c(accuracy1_vec,accuracy1)
precision1_vec<-c(precision1_vec,precision1)
recall1_vec<-c(recall1_vec,recall1)
pred1 = c(pred1, predict(m1Train,vaData, type="class"))
```

```
#####Random forests Model
#Error rate values of Random forests:
error1_vec
#Accuracy values of Random forests:
accuracy1_vec
#Precision values of Random forests:
precision1_vec
#Recall values of Random forests:
recall1_vec

#####Logistic regression Model
#Error rate values of Logistic regression:
error2_vec
#Accuracy values of Logistic regression:
accuracy2_vec
#Precision values of Logistic regression:
precision2_vec
#Recall values of Logistic regression:
recall2_vec
```

Result

```
> #Result of 2/3/4
> #####Random forests Model
> #Error rate values of Random forests:
> error1_vec
[1] 0.1129032 0.1279922 0.1221299 0.1104055 0.1085044
```

```
> #####Logistic regression Model
> #Error rate values of Logistic regression:
> error2_vec
[1] 0.3528837 0.3404983 0.3370787 0.3409868 0.3260020
```

Problem 4

Perform 5-fold cross validation and compute precision of M1 and M2
Does one of the model better than the other with 5% significance level

Preprocessing step

As we mentions in problem 3, we record all result of every fold, here is the five confusion matrix tables of two models:

We use the function `print(tbm2)` to print matrix for every loop. The Odd number table is for model 1, the even number of table is for model two.

	0	1
0	666	94
1	137	1149

	0	1
0	210	129
1	593	1114

	0	1
0	619	77
1	185	1166

	0	1
0	232	125
1	572	1118

	0	1
0	618	114
1	136	1179

	0	1
0	218	154
1	536	1139

	0	1
0	631	101
1	125	1190

	0	1
0	239	181
1	517	1110

	0	1
0	624	98
1	124	1200

	0	1
0	227	146
1	521	1152

Then we use 00 as TP, 10 as FP, 01 as FN, 11 as TN

Algorithm

① Precision:

$$precision = \frac{TP}{TP + FP}$$

② Error rate:

$$error\ rate = \frac{FP + FN}{P + N}.$$

③ Accuracy:

$$accuracy = \frac{TP + TN}{P + N}.$$

④ Recall:

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}.$$

⑤ 5% significant level:

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}},$$

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))]^2.$$

Code screenshot

```
#####Random forests Model
tbm1 = table(predict(m1Train,vaData, type="class"), vaData$outcome)
print(tbm1)
tbm3 = as.matrix(tbm1)
TP1 <- tbm3[1, 1]
FP1 <- tbm3[2, 1]
FN1 <- tbm3[1, 2]
TN1 <- tbm3[2, 2]
P1 <- TP1 + FN1
N1 <- FP1 + TN1
accuracy1 <- (TP1 + TN1) / (P1 + N1)
error_rate1<- (FP1 + FN1) / (P1 + N1)
precision1 <- TP1 / (TP1 + FP1)
recall1 <- TP1 / P1
error1_vec<-c(error1_vec,error_rate1)
accuracy1_vec<-c(accuracy1_vec,accuracy1)
precision1_vec<-c(precision1_vec,precision1)
recall1_vec<-c(recall1_vec,recall1)
pred1 = c(pred1, predict(m1Train,vaData, type="class"))
```

```
#####Logistic regression Model
tbm2 = table(ifelse(predict(m2Train,vaData, type="response") > 0.5,1,0), vaDa
print(tbm2)
tbm4 = as.matrix(tbm2)
TP2 <- tbm4[1, 1]
FP2 <- tbm4[2, 1]
FN2 <- tbm4[1, 2]
TN2 <- tbm4[2, 2]
P2 <- TP2 + FN2
N2 <- FP2 + TN2
error_rate2 <- (FP2 + FN2) / (P2 + N2)
accuracy2 <- (TP2 + TN2) / (P2 + N2)
precision2 <- TP2 / (TP2 + FP2)
recall2 <- TP2 / P2
error2_vec<-c(error2_vec,error_rate2)
accuracy2_vec<-c(accuracy2_vec,accuracy2)
precision2_vec<-c(precision2_vec,precision2)
recall2_vec<-c(recall2_vec,recall1)
pred2 = c(pred2, ifelse(predict(m2Train,vaData, type="response") > 0.5,1,0))
```

Result

```
> #Result of 2/3/4
> #####Random forests Model
> #Error rate values of Random forests:
> error1_vec
[1] 0.1129032 0.1279922 0.1221299 0.1104055 0.1085044
> #Accuracy values of Random forests:
> accuracy1_vec
[1] 0.8870968 0.8720078 0.8778701 0.8895945 0.8914956
> #Precision values of Random forests:
> precision1_vec
[1] 0.8293898 0.7699005 0.8196286 0.8346561 0.8342246
> #Recall values of Random forests:
> recall1_vec
[1] 0.8763158 0.8893678 0.8442623 0.8620219 0.8642659
>
```

```
> #####Logistic regression Model
> #Error rate values of Logistic regression:
> error2_vec
[1] 0.3528837 0.3404983 0.3370787 0.3409868 0.3260020
> #Accuracy values of Logistic regression:
> accuracy2_vec
[1] 0.6471163 0.6595017 0.6629213 0.6590132 0.6739980
> #Precision values of Logistic regression:
> precision2_vec
[1] 0.2615193 0.2885572 0.2891247 0.3161376 0.3034759
> #Recall values of Logistic regression:
> recall2_vec
[1] 0.8763158 0.8893678 0.8442623 0.8620219 0.8642659
```

```
>
> print(paste('Precision of Random forests: ', mean(precision1_vec)))
[1] "Precision of Random forests:  0.817559923321551"
> print(paste('Precision of Logistic regression: ', mean(precision2_vec)))
[1] "Precision of Logistic regression:  0.2917629373894"
>
```

```
> print(paste('The t value of the two models is : ', t))
[1] "The t value of the two models is :  -47.5232974716549"
> |
```

Analyze the result

We want to see does one of the model better than the other with 5% significance level. The sig/2 = 0.025, and the df = 4. And our t value is -47.52 < -2.776. Then our value of t lies in the rejection region, within the distribution's tails. This means that we can **reject the null hypothesis** that the means of M1 and M2 are the same and conclude that there is a statistically significant difference between the two models. So, we go back to look at the precision values, error rate values and accuracy values, Model 1 have a higher precision and lower error rate. So the **Random Forest** model is better than **Logistic regression** model.

TABLE B: t-DISTRIBUTION CRITICAL VALUES

	Tail probability p											
df	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	.765	.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	.741	.941	1.190	1.533	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	.727	.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.893	6.869
6	.718	.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	.711	.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	.706	.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	.703	.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	.700	.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	.697	.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	.695	.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318
13	.694	.870	1.079	1.350	1.771	2.160	2.282	2.650	3.012	3.372	3.852	4.221
14	.692	.868	1.076	1.345	1.761	2.145	2.264	2.624	2.977	3.326	3.787	4.140

Source R code

```
#Problem 1
#read the csv file
poll = read.csv("C:/NJIT Lin Tang/CS634/Final/presidential_polls.csv")
#Remove the "20" at the beginning of date if any
poll$createddate = gsub("^20","",as.character(poll$createddate))
#specify date format
poll$createddate = as.Date(poll$createddate, format = "%m/%d/%y")
```

```
#extract months
months = strftime(poll$createddate, "%Y-%m")
#compute average poll without missing values
adjpoll_clinton_month_avg = aggregate(poll$adjpoll_clinton ~ months, FUN = mean, na.rm = TRUE,
data = poll)
adjpoll_trump_month_avg = aggregate(poll$adjpoll_trump ~ months, FUN = mean, na.rm = TRUE,
data = poll)
adjpoll_johnson_month_avg = aggregate(poll$adjpoll_johnson ~ months, FUN = mean, na.rm =
TRUE, data = poll)
#merge data into a melt data frame
a = data.frame(adjpoll_clinton_month_avg,"Clinton")
colnames(a) = c("month","poll","name")
b =   data.frame(adjpoll_trump_month_avg,"Trump")
colnames(b) = c("month","poll","name")
c = data.frame(adjpoll_johnson_month_avg,"Johnson")
colnames(c) = c("month","poll","name")
poll_month_avg = rbind(a,b,c)
poll_month_avg
#plot the three lines in one plot

library(ggplot2)
ggplot(poll_month_avg, aes(x = month, y = poll, group = name, color = name)) +
geom_line(size = 1.5)   +
theme(axis.text.x = element_text(angle = 45)) +
ggtitle("Trends of the polls over time (by month)") +
labs(y = "Percentage of intended vote/ Adjusted poll\n") +
labs(x = "Month")



#Problem 2/3/4
library(randomForest)
poll$outcome = ifelse(poll$adjpoll_clinton > poll$adjpoll_trump, 1, 0)
poll$diffdays = as.Date(poll$enddate, format = "%m/%d/%y") - as.Date(poll$startdate, format =
"%m/%d/%y")
poll$collecteddays = poll$createddate - min(poll$createddate)
pollnum = na.omit(poll[,c(9,10,11,12,13,28,29,30)])
#M1 = randomForest(factor(outcome) ~   grade + samplesize + population + poll_wt + diffdays +
collecteddays, data = pollnum)
#M2  =  glm(factor(outcome)  ~    grade  +  samplesize  +  population  +  poll_wt  +  diffdays  +
collecteddays, data = pollnum, family = binomial)

set.seed(2016)
nrows <- dim(pollnum)[1]
```

```
#randomize
pollVld <- pollnum[sample(1:nrows), ]
kfold <- 5

splitIndex <- (1:nrows)%%kfold
splitFactor <- factor(splitIndex[order(splitIndex)])

pollSub <- split(pollVld,splitFactor)
print(dim(pollSub[[1]]))

error1_vec <- error2_vec <- NULL
accuracy1_vec <- accuracy2_vec <- NULL
precision1_vec <- precision2_vec <- NULL
recall1_vec <- recall2_vec <- NULL
pred1 <- pred2 <- NULL

for(iValid in seq(1,kfold)) {
  trData <- NULL
  vaData <- NULL
  for(j in seq(1,kfold)) {
    if(j!=iValid){
      trData <- rbind(trData,pollSub[[j]])
    }
    else {
      vaData <- pollSub[[j]]
    }
  }

  m1Train = randomForest(factor(outcome) ~ grade + samplesize + population + poll_wt + diffdays
+ collecteddays, data = trData)
  m2Train = glm(factor(outcome) ~  grade + samplesize + population + poll_wt + diffdays +
collecteddays, data = trData, family = binomial)

  #####Random forests Model
  tbm1 = table(predict(m1Train,vaData, type="class"), vaData$outcome)
  print(tbm1)
  tbm3 = as.matrix(tbm1)
  TP1 <- tbm3[1, 1]
  FP1 <- tbm3[2, 1]
  FN1 <- tbm3[1, 2]
  TN1 <- tbm3[2, 2]
  P1 <- TP1 + FN1
  N1 <- FP1 + TN1
  accuracy1 <- (TP1 + TN1) / (P1 + N1)
```

```
    error_rate1<- (FP1 + FN1) / (P1 + N1)
    precision1 <- TP1 / (TP1 + FP1)
    recall1 <- TP1 / P1
    error1_vec<-c(error1_vec,error_rate1)
    accuracy1_vec<-c(accuracy1_vec,accuracy1)
    precision1_vec<-c(precision1_vec,precision1)
    recall1_vec<-c(recall1_vec,recall1)
    pred1 = c(pred1, predict(m1Train,vaData, type="class"))

    #####Logistic regression Model
    tbm2 = table(ifelse(predict(m2Train,vaData, type="response") > 0.5,1,0), vaData$outcome)
    print(tbm2)
    tbm4 = as.matrix(tbm2)
    TP2 <- tbm4[1, 1]
    FP2 <- tbm4[2, 1]
    FN2 <- tbm4[1, 2]
    TN2 <- tbm4[2, 2]
    P2 <- TP2 + FN2
    N2 <- FP2 + TN2
    error_rate2 <- (FP2 + FN2) / (P2 + N2)
    accuracy2 <- (TP2 + TN2) / (P2 + N2)
    precision2 <- TP2 / (TP2 + FP2)
    recall2 <- TP2 / P2
    error2_vec<-c(error2_vec,error_rate2)
    accuracy2_vec<-c(accuracy2_vec,accuracy2)
    precision2_vec<-c(precision2_vec,precision2)
    recall2_vec<-c(recall2_vec,recall1)
    pred2 = c(pred2, ifelse(predict(m2Train,vaData, type="response") > 0.5,1,0))
}
predictm1 = ifelse(mean(pred1) > 0.5,1,0)
predictm2 = ifelse(mean(pred2) > 0.5,1,0)

#Result of 2/3/4
#####Random forests Model
#Error rate values of Random forests:
error1_vec
#Accuracy values of Random forests:
accuracy1_vec
#Precision values of Random forests:
precision1_vec
#Recall values of Random forests:
recall1_vec

#####Logistic regression Model
```

```
#Error rate values of Logistic regression:
error2_vec
#Accuracy values of Logistic regression:
accuracy2_vec
#Precision values of Logistic regression:
precision2_vec
#Recall values of Logistic regression:
recall2_vec


#Who is the winner of Random forests    1:Clinton wins, 0: Trump wins:
predictm1
#Who is the winner of Logistic regression    1:Clinton wins, 0: Trump wins:
predictm2

print(paste('Precision of Random forests: ', mean(precision1_vec)))
print(paste('Precision of Logistic regression: ', mean(precision2_vec)))

avg_err1 <- mean(error1_vec)
avg_err2 <- mean(error2_vec)
difference <- error1_vec - error2_vec - (avg_err1 - avg_err2)
k <- length(precision1_vec)
var <- t(difference)%*%difference/k
t <- (avg_err1 - avg_err2)/sqrt(var/k)
print(paste('The t value of the two models is : ', t))
```