Name: Zirui Zheng (Michael)

Course: Machine learning with R

Step 1 - Explore and Prepare the data

Explore the data by checking the structure and summary.

Structure:

```
'data.frame': 4521 obs. of 16 variables:
$ age : int 61 38 49 45 56 31 51 36 42 39 ...
$ job : Factor w/ 12 levels "admin.","blue-collar",..: 11 3 2 5 10 10 8 7 5 10 ...
$ marital : Factor w/ 3 levels "divorced","married",..: 2 2 1 1 2 3 2 3 2 3 ...
$ education: Factor w/ 4 levels "primary","secondary",..: 1 3 1 3 2 2 2 3 3 2 ...
$ default : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
$ balance : int 1414 4600 1584 1786 1176 2296 1840 1882 1958 1650 ...
$ housing : Factor w/ 2 levels "no","yes": 1 1 2 2 1 2 1 1 2 2 ...
$ loan : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 ...
$ contact : Factor w/ 3 levels "cellular","telephone",..: 1 3 3 3 1 3 2 1 1 1 ...
$ day : int 30 16 3 8 10 6 9 21 20 20 ...
$ month : Factor w/ 12 levels "apr","aug","dec",..: 5 7 7 9 6 7 4 2 10 2 ...
$ pdays : int -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
$ previous: int 0 0 0 0 0 0 0 0 0 ...
$ poutcome: Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4 4 4 ...
$ y : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 1 1 ...
```

Summary:

```
job
                                 education default
                      marital
                                                      balance
                                                                 housing loan
Min. :21.00 management: 969 divorced: 528 primary: 678 no: 4445 Min. :-3499
no:1962 no:3830
1st Qu.:35.00 blue-collar:946 married:2797 secondary:2306 yes: 76 1st Qu.: 1529 yes:2559 yes: 691
Median: 41.00 technician: 768 single: 1196 tertiary: 1350
                                                            Median: 1900
Mean :43.17 admin. :478
                                unknown: 187
                                                       Mean : 2869
3rd Qu.:51.00 services :417
                                                3rd Qu.: 2925
Max. :89.00 retired :230
                                               Max. :72751
        (Other) :713
                        month
                                  campaign
                                                pdays
                                                            previous
  contact
               day
poutcome
cellular:2896 Min.: 1.00 may:1398 Min.: 1.000 Min.:-1.00 Min.: 0.0000
failure: 490
telephone: 301 1st Qu.: 9.00 jul : 706 1st Qu.: 1.000 1st Qu.: -1.00 1st Qu.: 0.0000
other: 197
unknown:1324 Median:16.00 aug:633 Median:2.000 Median:-1.00 Median:
0.0000 success: 129
        Mean :15.92 jun :531 Mean :2.794 Mean :39.77 Mean :0.5426
unknown:3705
        3rd Qu.:21.00 nov : 389 3rd Qu.: 3.000 3rd Qu.: -1.00 3rd Qu.: 0.0000
        Max. :31.00 apr : 293 Max. :50.000 Max. :871.00 Max. :25.0000
                (Other): 571
```

y no :4000 yes: 521

Use any() function to see that there's no missing value:

[1] FALSE

After the exploration, I found that the data set is similar to the "credit.csv" file we did in class. There not not just numeric data in the file, and our predictor "y"is categorical by "yes" and "no". Thus, I think it better to use a decision tree model to start: to build decision trees to decide if one subscribed a term deposit. As I am not going to use numerical methods like regression, thus, I am not going to use logistic algorithm, dummy variables, or to make it nominal distributed.

Step 2 - Train models

Sample the train data with 80% rate. After splitting the train and test data, check the proportion of y in both sets:

train:

no yes 0.8849558 0.1150442

test:

no yes 0.8839779 0.1160221

Use C5.0 function to build a decision tree model.

Call:

C5.0.default($x = f_{train_dt[-16]}$, $y = f_{train_dt}$)

Classification Tree

Number of samples: 3616 Number of predictors: 15

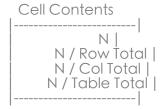
Tree size: 9

Non-standard options: attempt to group attributes

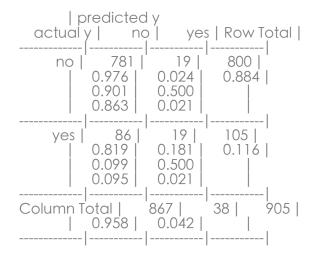
no yes 867 38

Step 3 - Evaluate performance

The CrossTable() shows that kappa is 0.873.



Total Observations in Table: 905



Error rate by checking the difference between prediction and test data:

[1] 0.1160221

Step 4 - Improve performance

**Use adabag library to do an adaboost on the model. However, the result doesn't look good in Kappa compared to decision tree above. Plus, I used detach() function to detach the adabag library, since I found it conflicting with ipred when using bagging method.

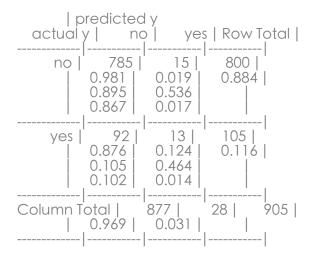
```
Observed Class
Predicted Class no yes
no 3968 326
yes 32 195
Estimate Std.Err 2.5% 97.5% P-value
kappa 0.4854 0.0227 0.4409 0.5299 1.961e-101
```

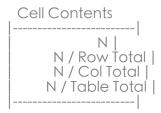
**Use C5.0 to boost the decision tree. I ran a for loop so that I can check different numbers of trials (10, 20, 30). Also, in the for loop, I generated crosstable and mean error accordingly, followed by the increasing order of trials:

Cell Contents	

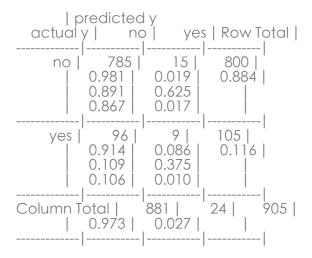
```
| N |
| N / Row Total |
| N / Col Total |
| N / Table Total |
|------|
```

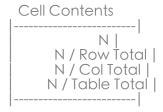
Total Observations in Table: 905



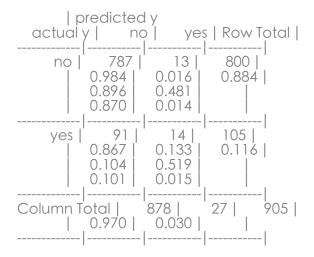


Total Observations in Table: 905





Total Observations in Table: 905



The kappa for trials 10, 20, 30: 0.869, 0.864, 0.873

Error rate followed by increasing order of trials:

[1] 0.1182320 0.1226519 0.1149171

Trials 30 has a better result. But after running 40, 50, 60 trials (too slow the process to put it here in the report), trials 30 is still the best approach, while it remains similar results with decision tree model.

**I started to think if the results would be better with evenly distributed split partition. So I created another basic decision tree model using createDataPartition(). Check the proportion of train and test data:

Great. Then use C5.0 to generate a tree:

Call:

```
C5.0.default(x = f_{train_i}[-16], y = f_{train_i}(y)
```

Classification Tree

Number of samples: 3617 Number of predictors: 15

Tree size: 4

Non-standard options: attempt to group attributes

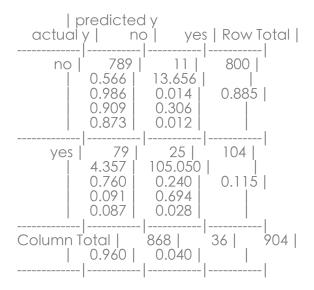
no yes 868 36

After prediction, run a CrossTable() to see that the kappa is 0.892.

```
Cell Contents

|------|
N |
Chi-square contribution |
N / Row Total |
N / Col Total |
N / Table Total |
|-------
```

Total Observations in Table: 904



Error rate:

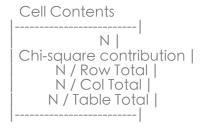
[1] 0.09955752

Thus, this method has a better prediction result. I tried run boosting on that, failing to get a better result. I feel it too redundant to write it in the report, as different trials actually give the same results under such circumstances.

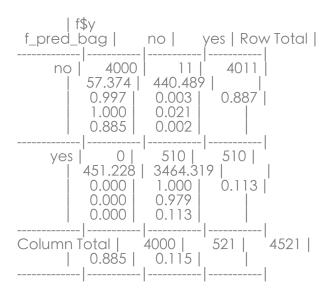
^{**}Use ipred bagging to improve performance with nbagg = 25:

Error:

[1] 0.00243309



Total Observations in Table: 4521



Kappa of CrossTable: 0.99

Both low error rate and high Kappa show that the Bagging method improved the prediction a lot. It is now the best approach.

**Use tuning in decision tree. Below is the summary of the tuning process.

Bagged CART

4521 samples 15 predictor 2 classes: 'no', 'yes'

No pre-processing Resampling: Cross-Validated (10 fold) Summary of sample sizes: 4069, 4069, 4069, 4069, 4069, ... Resampling results:

Accuracy Kappa 0.8845396 0.2238534

3 attributes winnowed Estimated importance of remaining attributes:
44% contactunknown

```
44% contactunknown
7% poutcomesuccess
2% balance
1% jobtechnician
1% day
1% jobmanagement
1% maritalsingle
1% monthsep
<1% maritalmarried
<1% loanyes
<1% monthaug
<1% previous
<1% age
<1% jobblue-collar
<1% jobentrepreneur
<1% jobhousemaid
<1% jobretired
<1% jobself-employed
<1% jobservices
<1% jobstudent
<1% jobunemployed
<1% educationsecondary
<1% education tertiary
<1% educationunknown
<1% defaultyes
<1% housingyes
<1% contacttelephone
<1% monthfeb
```

<1% monthjan <1% monthjul <1% monthjun <1% monthmay <1% monthnov <1% monthoct <1% campaign

```
<1% pdays
  <1% poutcomeother
  <1% poutcomeunknown
---- Trial 0: ----
Rules:
Rule 0/1: (4392/438, lift 1.0)
       poutcomesuccess <= 0
       -> class no [0.900]
Rule 0/2: (129/46, lift 5.6)
       poutcomesuccess > 0
       -> class yes [0.641]
Default class: no
---- Trial 1: ----
Rules:
Rule 1/1: (4178.3/1082.7, lift 1.0)
       age <= 62
       monthoct <= 0
       pdays <= 374
      -> class no [0.741]
Rule 1/2: (49.3/13, lift 2.5)
       monthoct <= 0
      pdays > 374
       -> class yes [0.728]
Rule 1/3: (132.3/41.8, lift 2.4)
       monthoct > 0
      -> class yes [0.682]
Rule 1/4: (190.8/75.2, lift 2.1)
       age > 62
       pdays <= 374
       -> class yes [0.605]
Default class: no
---- Trial 2: ----
Rules:
Rule 2/1: (1068.7/228.3, lift 1.3)
       contactunknown > 0
      -> class no [0.786]
Rule 2/2: (3908.3/1426, lift 1.0)
       jobstudent <= 0
       monthjun <= 0
      -> class no [0.635]
Rule 2/3: (147.3/34, lift 2.1)
```

contactunknown <= 0 monthjun > 0 -> class yes [0.765] Rule 2/4: (95.1/33.8, lift 1.7) iobstudent > 0 contactunknown <= 0 monthjun <= 0 -> class yes [0.641] Default class: no ---- Trial 3: ----Rules: Rule 3/1: (4241.4/1775.5, lift 1.0) poutcomesuccess <= 0 -> class no [0.581] Rule 3/2: (279.6/110.2, lift 1.4) poutcomesuccess > 0 -> class yes [0.605] Default class: no ---- Trial 4: ----Rules: Rule 4/1: (979.8/289, lift 1.3) contactunknown > 0 -> class no [0.705] Rule 4/2: (542.5/185, lift 1.2) loanves > 0 -> class no [0.658] Rule 4/3: (618.3/229.5, lift 1.2) balance <= 1493 loanyes <= 0 -> class no [0.628] Rule 4/4: (2675.8/1196.9, lift 1.2) balance > 1493 loanyes <= 0 contactunknown <= 0 -> class yes [0.553] Default class: no ---- Trial 5: ----Rules: Rule 5/1: (124/27.2, lift 1.4)

balance > 9122 monthjun <= 0 monthsep <= 0 -> class no [0.776]

Rule 5/2: (930.4/295.9, lift 1.2) contactunknown > 0 -> class no [0.682]

Rule 5/3: (2977/1232.7, lift 1.0)
balance <= 3334
monthjun <= 0
monthsep <= 0
pdays <= 404
-> class no [0.586]

Rule 5/4: (46.8/10.5, lift 1.7)
contactunknown <= 0
monthjun <= 0
monthsep <= 0
pdays > 404
-> class yes [0.764]

Rule 5/5: (164.4/60.9, lift 1.4) contactunknown <= 0 monthjun > 0 -> class yes [0.628]

Rule 5/6: (81.6/32.4, lift 1.4) contactunknown <= 0 monthsep > 0 -> class yes [0.600]

Rule 5/7: (761.3/315.2, lift 1.3)
balance > 3334
balance <= 9122
contactunknown <= 0
monthsep <= 0
-> class yes [0.586]

Default class: no

---- Trial 6: ----

Rules:

Rule 6/1: (104.9/23.2, lift 1.4) educationunknown <= 0 campaign > 9 -> class no [0.774]

Rule 6/2: (155.3/43.6, lift 1.3) educationunknown > 0 monthoct <= 0 -> class no [0.716]

Rule 6/3: (2072.1/748, lift 1.2) maritalmarried > 0 monthoct <= 0 previous <= 1

```
-> class no [0.639]
```

Rule 6/4: (177.8/71, lift 1.3) monthoct > 0 -> class yes [0.599]

Rule 6/5: (675.7/285.5, lift 1.3)
educationunknown <= 0
monthoct <= 0
campaign <= 9
previous > 1
-> class yes [0.577]

Rule 6/6: (1821.2/851.8, lift 1.2) maritalmarried <= 0 educationunknown <= 0 campaign <= 9 -> class yes [0.532]

Default class: no

---- Trial 7: ----

Rules:

Default class: no

*** boosting reduced to 7 trials since last classifier is very inaccurate

Evaluation on training data (4521 cases):

Trial	Rules
	No Errors
0 1 2 3 4 5 6 boost	2 484(10.7%) 4 555(12.3%) 4 565(12.5%) 2 484(10.7%) 4 1965(43.5%) 7 850(18.8%) 6 1790(39.6%) 492(10.9%) <<

(a) (b) <-classified as 3927 73 (a): class no 419 102 (b): class yes

Attribute usage:

100.00%	poutcomesuccess
99.65%	pdays
99.62%	monthoct
97.97%	age

90.02%	monthjun
89.38%	maritalmarried
88.52%	balance
88.08%	jobstudent
85.80%	monthsep
80.42%	contactunknown
79.81%	loanyes
64.83%	previous
49.10%	educationunknown
45.08%	campaign

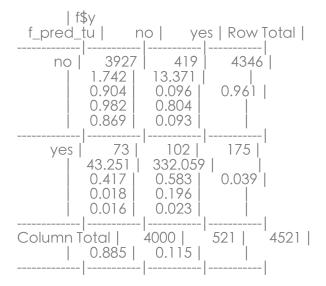
Time: 0.2 secs

```
f_pred_tu no yes
no 3927 419
yes 73 102
```

```
Cell Contents

|------|
N |
Chi-square contribution |
N / Row Total |
N / Col Total |
N / Table Total |
|------
```

Total Observations in Table: 4521



CrossTable show that the [Pr(a)-Pr(e)]/[1-Pr(3)] = 0.89, pretty good, but not as good as ipred bagging.

^{**}Use customized bagging, and below is the result:

```
.model .trials .winnow
1 tree
        1 FALSE
2 tree
         5 FALSE
3 tree
        10 FALSE
4 tree
        15 FALSE
        20 FALSE
5 tree
6 tree
         25 FALSE
7 tree 30 FALSE
8 tree 35 FALSE
C5.0
4521 samples
 15 predictor
 2 classes: 'no', 'yes'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4069, 4069, 4069, 4068, 4069, 4069, ...
Resampling results across tuning parameters:
trials Accuracy Kappa
     0.8922801 0.2325560
     0.8854236 0.1404052
    0.8852004 0.1625564
 15
    0.8829885 0.1749768
 20
    0.8854222 0.1753795
 25
     0.8858646 0.1874779
 30
     0.8856434 0.1868124
     0.8856434 0.1868124
```

Tuning parameter 'model' was held constant at a value of tree
Tuning parameter 'winnow' was held constant at a
value of FALSE
Kappa was used to select the optimal model using the one SE rule.
The final values used for the model were trials = 1, model = tree and winnow = FALSE.

It shows that when trial = 1, the model has the best prediction, but not as good as ipred bagging.

Step 4.5 Another Model

Use Random Forest

Call:
randomForest(formula = y ~ ., data = f)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3

OOB estimate of error rate: 11.1%

Confusion matrix: no yes class.error no 3934 66 0.0165000 yes 436 85 0.8368522 I built a random forest on the data, but the result doesn't look good. It appeared to be much worse than most of the models above. Thus, I started resampling using both random forest and C50. The results are as follows, in the order of Random Forest resampling and C50 resampling:

```
Random Forest
4521 samples
 15 predictor
 2 classes: 'no', 'yes'
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 4069, 4069, 4069, 4069, 4068, 4069, ...
Resampling results across tuning parameters:
 mtry Accuracy Kappa
 2 0.8854902 0.02817444
 4 0.8900023 0.15028977
 8 0.8887418 0.18531178
 16 0.8885428 0.21570099
Kappa was used to select the optimal model using the largest value.
The final value used for the model was mtry = 16.
C5.0
4521 samples
 15 predictor
 2 classes: 'no', 'yes'
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 4069, 4069, 4069, 4068, 4069, 4069, ...
Resampling results across tuning parameters:
 trials Accuracy Kappa
 10 0.8879229 0.1729337
 20
    0.8880113 0.2046911
 30
     0.8877680 0.2035707
Tuning parameter 'model' was held constant at a value of tree
Tuning parameter 'winnow' was held constant at a
value of FALSE
```

Step 5 - Select the final model

During my work, I found out that most models have accuracy in 85% to 90%, which are all pretty decent results. In pragmatic world, these figures could give a good predictions on future trends. However, in <u>decision tree model with ipred bagging</u>, the accuracy is over 99%. This is quite impressive that I tried to run it more times to make sure no bug exists. In the end, I would pick this method as the best approach.

The final values used for the model were trials = 20, model = tree and winnow = FALSE.

Kappa was used to select the optimal model using the largest value.

In addition to conclusion, I had some interesting discoveries. In common sense, more trials in machine learning might lead to more accuracy, while it had been proved wrong in my work. Also, when boosting fail to improve the accuracy in one scenario, I should still try it in other models, because it might work.

Overall I am glad to have this final exam as it taught me a lot of insights during practice.