Tuan Dang
05/28/2021

Final Project: Machine Learning Classification Models

Part 1: Logistic Regression

Reviewing the data:

We will start by looking at the variables.
An overwhelming majority of the target variable is "no" for the observations with failed marketing attempts, and I'm not sure if this would affect the predictability of the model. This variable will later be encoded to be 1 for "yes" and 0 for "no."

```
> table(data$y)
  no  yes
3668  451
```

Most participants seem to be in the middle age range, with some outliers on both ends.

```
> summary(data$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   32.00   38.00   40.11   47.00   88.00
```

The job predictor is not evenly distributed. There are some unknown observations but the number is low enough to be negligible.

```
> table(data$job)
       admin.   blue-collar  entrepreneur     housemaid    management       retired
self-employed      services
         1012           884           148           110           324           166           159
393
      student    technician    unemployed       unknown
           82           691           111            39
```

The marital predictor seems ordinary, although it is interesting that they grouped divorced and widowed into the same category.

```
> table(data$marital)

divorced  married   single  unknown
     446     2509     1153       11
```

```
> table(data$education)
         basic.4y          basic.6y          basic.9y       high.school        illiterate
              429               228               574               921                 1
professional.course university.degree           unknown
              535              1264               167
```

An overwhelming majority of the default predictor is a “no,” which, I think, makes this a bad predictor, since defaulters probably wouldn’t answer to this campaign.

```
> table(data$default)
     no unknown     yes
   3315     803       1
```

```
> table(data$housing)
     no unknown     yes
   1839     105    2175
```

Loan has a similar problem to the default predictor.

```
> table(data$loan)
     no unknown     yes
   3349     105     665
```

```
> table(data$contact)
 cellular telephone
     2652      1467
```

The month predictor is heavily skewed to some months more than others, with some only having a few dozen observations out of roughly 4000.

```
> table(data$month)
 apr  aug  dec  jul  jun  mar  may  nov  oct  sep
 215  636   22  711  530   48 1378  446   69   64
```

```
> table(data$day_of_week)
fri mon thu tue wed
768 855 860 841 795
```

Duration shows some signs of an outlier.

```
> summary(data$duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0   103.0   181.0   256.8   317.0  3643.0
```

```
> summary(data$campaign)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   2.537   3.000  35.000
```

pdays does not seem like a very useful predictor and can be removed from the data set.

```
> summary(data$pdays)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0   999.0   999.0   960.4   999.0   999.0
```

previous is heavily skewed to the left.

```
> summary(data$previous)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.1903  0.0000  6.0000
```

An overwhelming majority of poutcome is “nonexistent,” which might not be a bad thing. It could be significant that there is no previous data regarding success in marketing.

```
> table(data$poutcome)
    failure nonexistent     success
        454        3523         142
```

Creating the first model:

After reviewing the variables, we will start with a logistic regression model containing all of the variables based on a randomized training data set of 80% of the observations to give us a sense of how it would work. I chose not to look at the correlation between each variable since there are so many, and I couldn't think of a significant relationship between each pair. I'm mostly choosing variables based on significant levels and anova tests, which could be wrong.

```
install.packages("gmodels")
library(gmodels)
data$result <- ifelse(data$y == "yes", 1, 0)
train_sample <- sample(4119,3319)
data_train <- data[train_sample,]
data_test <- data[-train_sample,]
```

```
model1 <- glm(result ~ age + job + marital + education + default + housing + loan
+ contact + month + day_of_week + duration + campaign + pdays + previous +
poutcome, data = data_train, family = binomial)
```

```
Call:
glm(formula = result ~ age + job + marital + education + default +
    housing + loan + contact + month + day_of_week + duration +
    campaign + pdays + previous + poutcome, family = binomial,
    data = data_train)

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -4.786e+00  1.141e+00  -4.196 2.72e-05 ***
age                 2.046e-02  8.925e-03   2.292 0.021910 *
jobblue-collar     -3.224e-01  2.934e-01  -1.099 0.271883
jobentrepreneur    -1.225e+00  5.761e-01  -2.127 0.033436 *
```

jobhousemaid	3.388e-01	4.812e-01	0.704	0.481377	
jobmanagement	-8.117e-01	3.362e-01	-2.414	0.015759	*
jobretired	9.939e-02	3.702e-01	0.268	0.788315	
jobself-employed	-2.989e-01	4.280e-01	-0.698	0.484965	
jobservices	2.048e-01	2.987e-01	0.685	0.493095	
jobstudent	7.615e-01	4.217e-01	1.806	0.070935	.
jobtechnician	-4.459e-02	2.415e-01	-0.185	0.853524	
jobunemployed	4.321e-01	4.360e-01	0.991	0.321684	
jobunknown	-5.578e-01	7.546e-01	-0.739	0.459775	
maritalmarried	2.921e-01	2.588e-01	1.129	0.259047	
maritalsingle	3.168e-01	2.954e-01	1.073	0.283407	
maritalunknown	3.808e-01	1.172e+00	0.325	0.745269	
educationbasic.6y	3.273e-01	4.421e-01	0.740	0.459148	
educationbasic.9y	3.298e-01	3.538e-01	0.932	0.351300	
educationhigh.school	1.886e-01	3.366e-01	0.560	0.575267	
educationilliterate	-8.971e+00	3.247e+02	-0.028	0.977961	
educationprofessional.course	3.414e-01	3.696e-01	0.924	0.355564	
educationuniversity.degree	5.949e-01	3.396e-01	1.752	0.079808	.
educationunknown	8.567e-01	4.098e-01	2.091	0.036559	*
defaultunknown	-1.371e-02	2.177e-01	-0.063	0.949788	
housingunknown	-1.218e+00	6.524e-01	-1.866	0.061994	.
housingyes	-9.124e-02	1.493e-01	-0.611	0.541169	
loanunknown	NA	NA	NA	NA	
loanyes	-1.512e-01	2.045e-01	-0.739	0.459819	
contacttelephone	-1.360e+00	2.271e-01	-5.989	2.11e-09	***
monthaug	-7.208e-01	3.074e-01	-2.344	0.019061	*
monthdec	1.185e+00	6.560e-01	1.806	0.070882	.
monthjul	-1.394e+00	3.278e-01	-4.251	2.13e-05	***
monthjun	4.242e-01	3.347e-01	1.268	0.204969	
monthmar	1.563e+00	4.819e-01	3.244	0.001179	**
monthmay	-8.456e-01	2.943e-01	-2.874	0.004059	**
monthnov	-1.259e+00	3.477e-01	-3.621	0.000294	***
monthoct	9.223e-01	4.360e-01	2.115	0.034400	*
monthsep	1.272e-01	4.777e-01	0.266	0.790078	
day_of_weekmon	9.095e-02	2.331e-01	0.390	0.696402	

```
day_of_weekthu          -2.202e-02  2.355e-01  -0.093 0.925509
day_of_weektue           1.017e-01  2.379e-01   0.428 0.668904
day_of_weekwed           3.224e-01  2.423e-01   1.331 0.183282
duration                 4.857e-03  2.717e-04  17.873  < 2e-16 ***
campaign                -1.542e-01  5.077e-02  -3.038 0.002385 **
pdays                    2.639e-04  8.268e-04   0.319 0.749570
previous                 3.990e-01  1.950e-01   2.047 0.040691 *
poutcomenonexistent      2.745e-01  3.239e-01   0.847 0.396778
poutcomesuccess          2.698e+00  8.103e-01   3.329 0.000872 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’  0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2240.6  on 3318  degrees of freedom
Residual deviance: 1382.2  on 3272  degrees of freedom
AIC: 1476.2
```

This gives a very long list of variables, although we can see some that stood out to give us a sense of what variables to keep. I then tried a second model which is exactly the same as the first except this time we are excluding the “duration” predictor, since we are warned that this predictor can be problematic for a realistic model.

```
model2 <- glm(result ~ age + job + marital + education + default + housing + loan
+ contact + month + day_of_week + campaign + pdays + previous + poutcome,
data = data_train, family = binomial)
```

I am only including the AIC score for model2, since everything else is similar to model1. As we can see, including “duration” is very significant to the model, considering the AIC increased by nearly 500!

AIC: 1932.4

We will now look at an ANOVA test of model2 to see what variables we might want to keep.

anova(model2, test = "Chisq")

Analysis of Deviance Table

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)	
NULL			3318	2240.6		
age	1	19.504	3317	2221.1	1.004e-05	***
job	11	53.834	3306	2167.3	1.266e-07	***
marital	3	5.508	3303	2161.8	0.1381416	
education	7	10.306	3296	2151.5	0.1718682	
default	1	10.664	3295	2140.8	0.0010923	**
housing	2	0.932	3293	2139.9	0.6274754	
loan	1	0.248	3292	2139.6	0.6185467	
contact	1	44.201	3291	2095.4	2.964e-11	***
month	9	117.232	3282	1978.2	< 2.2e-16	***
day_of_week	4	1.899	3278	1976.3	0.7543768	
campaign	1	14.958	3277	1961.3	0.0001099	***
pdays	1	105.441	3276	1855.9	< 2.2e-16	***
previous	1	1.845	3275	1854.1	0.1743845	
poutcome	2	13.634	3273	1840.4	0.0010949	**

Improving the logistic regression model:

Based on the ANOVA test of model2, I made a new model with the most promising variables. This model does not include “duration,” which we will add later to see how it improves the predictability of the model.

model3 <- glm(result ~ age + job + contact + month + campaign + poutcome, data = data_train, family = binomial)

The AIC score improved from model2.

AIC: 1911.3

Confusion matrix:

At this point, we can start to create confusion matrices to measure the accuracy of our model. Out of all the threshold probabilities that I tried, this gives the best result for model3. This gives an accuracy rate of 89.5% (716/800).

data_predictions <- predict(model3, newdata = data_test, type = "response")

pre <- ifelse(data_predictions > 0.55, 1, 0)

CrossTable(data_test$result[1:800], pre[1:800])

```
Total Observations in Table:  800
                      | pre[1:800]
data_test$result[1:800] |         0 |         1 | Row Total |
------------------------|-----------|-----------|-----------|
                      0 |       697 |         3 |       700 |
                        |     0.388 |    13.718 |           |
                        |     0.996 |     0.004 |     0.875 |
                        |     0.896 |     0.136 |           |
                        |     0.871 |     0.004 |           |
------------------------|-----------|-----------|-----------|
                      1 |        81 |        19 |       100 |
                        |     2.715 |    96.023 |           |
                        |     0.810 |     0.190 |     0.125 |
```

```
                      |     0.104 |     0.864 |           |
                      |     0.101 |     0.024 |           |
-------------------------|-----------|-----------|-----------|
          Column Total |       778 |        22 |       800 |
                      |     0.973 |     0.028 |           |
-------------------------|-----------|-----------|-----------|
```

Now, we will add “duration” to see how much it improves accuracy.

```
model4 <- glm(result ~ age + job + contact + month + campaign + poutcome +
duration, data = data_train, family = binomial)
```

AIC: 1458.8

The AIC of course improved quite a bit, while the accuracy somewhat increased, although not significantly, to 90.75% (726/800).

```
pre <- ifelse(data_predictions > 0.5, 1, 0)

CrossTable(data_test$result[1:800], pre[1:800])
```

```
Total Observations in Table:  800
                      | pre[1:800]
data_test$result[1:800] |         0 |         1 | Row Total |
-------------------------|-----------|-----------|-----------|
                    0 |       688 |        12 |       700 |
                      |     1.536 |    23.041 |           |
                      |     0.983 |     0.017 |     0.875 |
                      |     0.917 |     0.240 |           |
                      |     0.860 |     0.015 |           |
-------------------------|-----------|-----------|-----------|
                    1 |        62 |        38 |       100 |
```

```
                       |    10.753 |   161.290 |           |
                       |     0.620 |     0.380 |     0.125 |
                       |     0.083 |     0.760 |           |
                       |     0.077 |     0.048 |           |
-----------------------|-----------|-----------|-----------|
          Column Total |       750 |        50 |       800 |
                       |     0.938 |     0.062 |           |
-----------------------|-----------|-----------|-----------|
```

Part 2: k-NN

Now, we will move on to the k-NN classification method. Before we can create a model, we need to prepare the data for the method to work. Specifically, we need to convert the predictors to have numerical values.

Converting categorical predictors to numerical:

We will create a copy of the data set solely for the purpose of having numerical values.

```
newdata <- data
library(dplyr)
MakeNum <- function(x) as.numeric(as.factor(x))
newdata <- mutate_at(newdata, 3:10, MakeNum)
newdata$poutcome <- as.numeric(as.factor(newdata$poutcome))
```

Normalizes the data set:

```
install.packages("class")
library(class)
normalize <- function(x){
+ return ((x-min(x))/(max(x)-min(x)))}
str(newdata)
newdata_norm <- as.data.frame(lapply(newdata[,1:14],normalize))
```

View(newdata_norm)

Creating the model:

```
train <- newdata_norm[train_sample,]
train_label <- newdata[train_sample,16]
test <- newdata_norm[-train_sample,]
test_label <- newdata[-train_sample,16]
```

The first model will be a 3-NN model which includes all variables, which again serves as our benchmark.

```
kmodel <- knn(train=train, test=test, cl=train_label, k=3)
library(gmodels)
CrossTable(test_label, model)
```

```
Total Observations in Table:  800
             | kmodel
  test_label |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       686 |        14 |       700 |
             |     0.091 |     2.835 |           |
             |     0.980 |     0.020 |     0.875 |
             |     0.885 |     0.560 |           |
             |     0.858 |     0.018 |           |
-------------|-----------|-----------|-----------|
           1 |        89 |        11 |       100 |
             |     0.640 |    19.845 |           |
             |     0.890 |     0.110 |     0.125 |
             |     0.115 |     0.440 |           |
             |     0.111 |     0.014 |           |
-------------|-----------|-----------|-----------|
Column Total |       775 |        25 |       800 |
             |     0.969 |     0.031 |           |
```

```
-------------|-----------|-----------|-----------|
```

The accuracy ended up being around 87% (697/800), which is not very good, but we will now improve the model by using the predictors that we used for the logistic regression model that yielded the best results. We will be excluding "duration" for now.
This change slightly increased the accuracy to 87.6% (701/800). Furthermore, changing the k value from 3 to 9 increases the accuracy to 88.75% (710/800), which is a decent improvement. Although it is not as good as our logistic regression model (716/800).

```
newdata_norm <- as.data.frame(lapply(newdata[,c(1,2,8,9,12,14)],normalize))
train <- newdata_norm[train_sample,]
train_label <- newdata[train_sample,16]
test <- newdata_norm[-train_sample,]
test_label <- newdata[-train_sample,16]
kmodel2 <- knn(train=train, test=test, cl=train_label, k=3)
CrossTable(test_label, kmodel2)
```

```
Total Observations in Table:  800


             | kmodel2 
  test_label |         0 |         1 | Row Total | 
-------------|-----------|-----------|-----------|
           0 |       679 |        21 |       700 | 
             |     0.417 |     7.346 |           | 
             |     0.970 |     0.030 |     0.875 | 
             |     0.897 |     0.488 |           | 
             |     0.849 |     0.026 |           | 
-------------|-----------|-----------|-----------|
           1 |        78 |        22 |       100 | 
             |     2.921 |    51.422 |           | 
             |     0.780 |     0.220 |     0.125 | 
             |     0.103 |     0.512 |           | 
```

```
             |     0.098 |     0.028 |           |
-------------|-----------|-----------|-----------|
Column Total |       757 |        43 |       800 |
             |     0.946 |     0.054 |           |
-------------|-----------|-----------|-----------|
```

```
kmodel2 <- knn(train=train, test=test, cl=train_label, k=9)
CrossTable(test_label, kmodel2)

Total Observations in Table:  800


             | kmodel2
  test_label |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       692 |         8 |       700 |
             |     0.321 |     9.563 |           |
             |     0.989 |     0.011 |     0.875 |
             |     0.894 |     0.308 |           |
             |     0.865 |     0.010 |           |
-------------|-----------|-----------|-----------|
           1 |        82 |        18 |       100 |
             |     2.249 |    66.942 |           |
             |     0.820 |     0.180 |     0.125 |
             |     0.106 |     0.692 |           |
             |     0.102 |     0.022 |           |
-------------|-----------|-----------|-----------|
Column Total |       774 |        26 |       800 |
             |     0.968 |     0.032 |           |
-------------|-----------|-----------|-----------|
```

Now, we will include “duration” into this model, with k=9. This slightly increases the accuracy to 713/800, but is still not nearly as good as its “equivalent” logistic regression model (726/800).

```
newdata_norm <- as.data.frame(lapply(newdata[,c(1,2,8,9,11,12,14)],normalize))
train <- newdata_norm[train_sample,]
train_label <- newdata[train_sample,16]
test <- newdata_norm[-train_sample,]
test_label <- newdata[-train_sample,16]
kmodel3 <- knn(train=train, test=test, cl=train_label, k=9)
CrossTable(test_label, kmodel3)
```

Part 3: Decision Trees

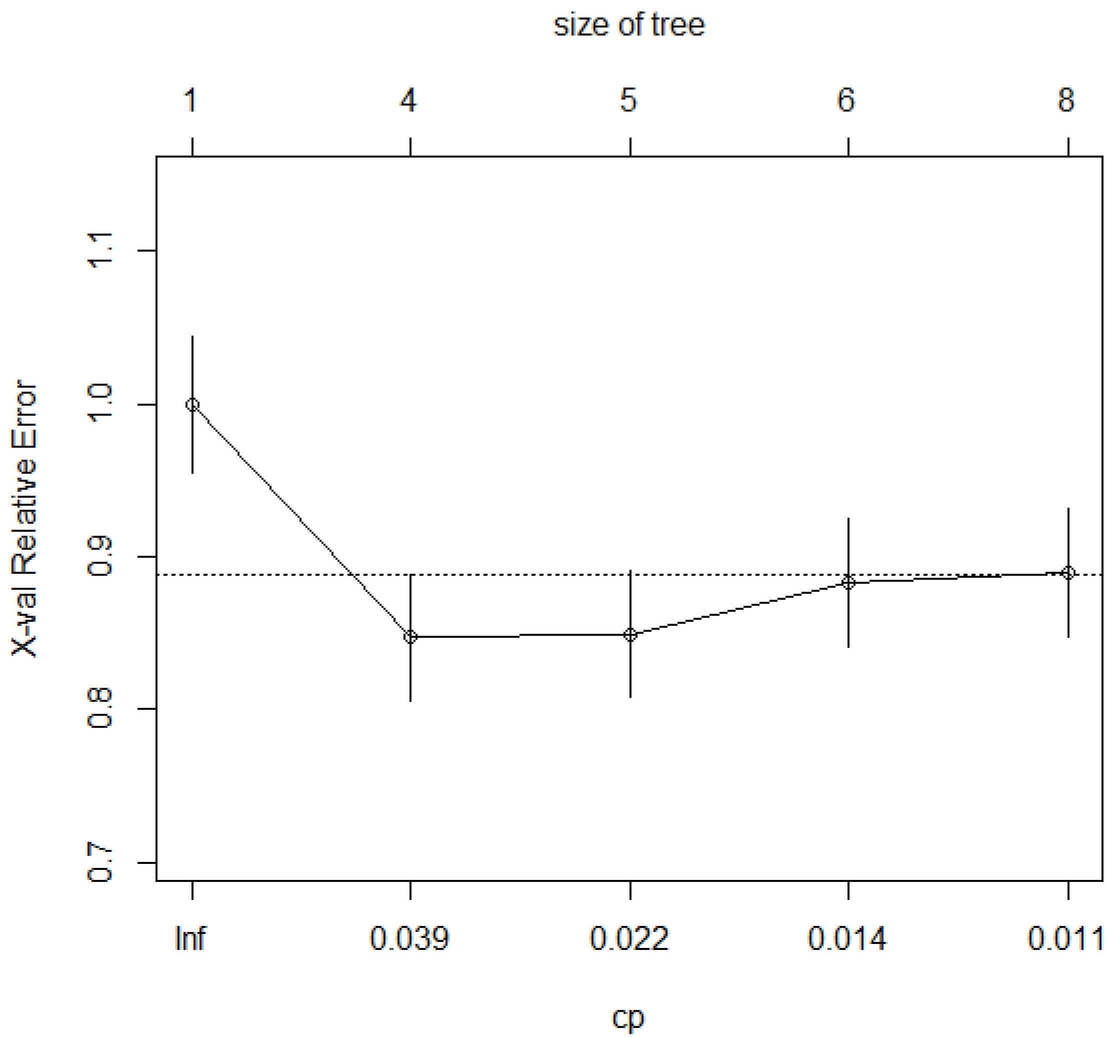
```
install.packages("rpart")
install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
tmodel <- rpart(y ~ ., data = data)
rpart.plot(tmodel)
pred<-predict(tmodel, type="class")
library(gmodels)
CrossTable(data$y,pred)
```

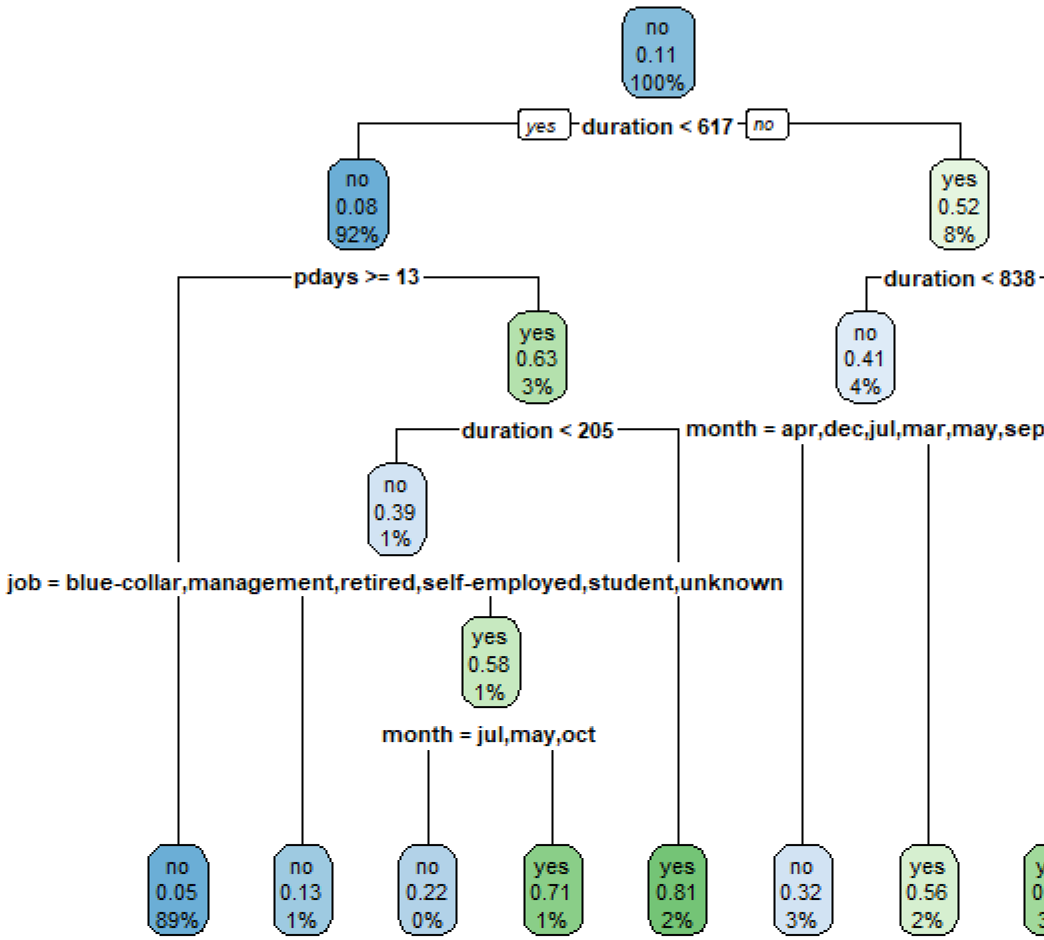
We will start with the most basic model which includes all predictors with no modifications or pruning. This gives an accuracy rate of 91.6% (3775/4119), which is very good and is better than all other models before. It is also interesting that in the decision tree graph below, it only shows a few predictors, which I guess could mean it only uses those variables. It would be very interesting if the model knows what variables to use.

```
Total Observations in Table:  4119
             | pred
      data$y |        no |       yes | Row Total |
```

```
-------------|-----------|-----------|-----------|
          no |      3564 |       104 |      3668 |
             |     9.197 |   111.068 |           |
             |     0.972 |     0.028 |     0.891 |
             |     0.937 |     0.330 |           |
             |     0.865 |     0.025 |           |
-------------|-----------|-----------|-----------|
         yes |       240 |       211 |       451 |
             |    74.802 |   903.322 |           |
             |     0.532 |     0.468 |     0.109 |
             |     0.063 |     0.670 |           |
             |     0.058 |     0.051 |           |
-------------|-----------|-----------|-----------|
Column Total |      3804 |       315 |      4119 |
             |     0.924 |     0.076 |           |
-------------|-----------|-----------|-----------|
```

The following graph shows us the cp level to optimize the size of our tree without compromising accuracy. It shows that cp should be 0.01 or lower.


size of tree
1
4
5
6
8
X-val Relative Error
1.1
1.0
0.9
0.8
0.7
Inf
0.039
0.022
0.014
0.011
cp



Next we will try a pruned version of the tree with cp = 0.01, which actually yields the same results. I also tried cp = 0.04, which was predicted to be a bad cp level by the graph and did give a worse accuracy for the model.

```
tmodel_pruned <- prune(tmodel, cp = 0.01)
pred<-predict(tmodel_pruned, type="class")
CrossTable(data$y,pred)
```

Next, we will modify the tree model to only include the "best" predictors from before. This actually gives a worse model than before, when we used all variables (3710 compared to 3775 correct predictions). Then, when we also include “duration,” it increases to 3760 correct predictions, which is better but is also worse than before.

```
tmodel <- rpart(y ~ age + job + contact + month + campaign + poutcome, data = data)
pred<-predict(tmodel, type="class")
CrossTable(data$y,pred)
```

```
Total Observations in Table:  4119
             | pred
      data$y |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |      3618 |        50 |      3668 |
             |     1.650 |    46.222 |           |
             |     0.986 |     0.014 |     0.891 |
             |     0.910 |     0.352 |           |
             |     0.878 |     0.012 |           |
-------------|-----------|-----------|-----------|
         yes |       359 |        92 |       451 |
             |    13.423 |   375.928 |           |
             |     0.796 |     0.204 |     0.109 |
             |     0.090 |     0.648 |           |
             |     0.087 |     0.022 |           |
-------------|-----------|-----------|-----------|
Column Total |      3977 |       142 |      4119 |
             |     0.966 |     0.034 |           |
-------------|-----------|-----------|-----------|
```

```
tmodel <- rpart(y ~ age + job + contact + month + campaign + poutcome + duration, data = data)

pred<-predict(tmodel, type="class")
```

CrossTable(data$y,pred)

```
Total Observations in Table:  4119
             | pred
      data$y |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |      3573 |        95 |      3668 |
             |     7.134 |    97.062 |           |
             |     0.974 |     0.026 |     0.891 |
             |     0.931 |     0.337 |           |
             |     0.867 |     0.023 |           |
-------------|-----------|-----------|-----------|
         yes |       264 |       187 |       451 |
             |    58.017 |   789.406 |           |
             |     0.585 |     0.415 |     0.109 |
             |     0.069 |     0.663 |           |
             |     0.064 |     0.045 |           |
-------------|-----------|-----------|-----------|
Column Total |      3837 |       282 |      4119 |
             |     0.932 |     0.068 |           |
-------------|-----------|-----------|-----------|
```

Part 4: Bayesian Classification

Encoding continuous predictors into categorical predictors:

Since age and duration are continuous predictors, we need to encode them into categorical predictors for the Bayesian method to work. We will split each variable into quartiles, which yields a well-balanced sample size for each category.

```
quantile(newdata$age, prob = c(.25, .5, .75))
25% 50% 75%
 32  38  47
```

```
newdata$age <- ifelse(newdata$age <= 32, 1, ifelse(newdata$age <= 38, 2,
ifelse(newdata$age <= 47, 3, 4)))
table(newdata$age)
   1    2    3    4
1135  971 1010 1003
```

```
quantile(newdata$duration, prob = c(.25, .5, .75))
25% 50% 75%
103 181 317
```

```
newdata$duration <- ifelse(newdata$duration <= 103, 1, ifelse(newdata$duration
<= 181, 2, ifelse(newdata$duration <= 317, 3, 4)))
table(newdata$duration)
   1    2    3    4
1035 1038 1017 1029
```

Creating the model:

```
install.packages("e1071")
library(e1071)
train_sample <- sample(4119,3319)
train <-
newdata[train_sample,c("age","job","contact","month","campaign","poutcome")]
train_labels <- newdata[train_sample,c("result")]
test <-
newdata[-train_sample,c("age","job","contact","month","campaign","poutcome")]
test_labels <- newdata[-train_sample,c("result")]
bmodel <- naiveBayes(train, train_labels, laplace = 0)
preds <- predict(bmodel,test,type="raw")
predclass <- ifelse(preds[,2] >= .5, 1, 0)
CrossTable(test_labels,predclass)
```

For this method, we will start with a model with only the selected variables from before (excluding “duration”). The 0.5 threshold yields the best result of 85%

(680/800) which is the worst result of all the methods so far. I also tried a laplace estimator of 1 as recommended but that gave the same results.

```
Total Observations in Table:  800
             | predclass
 test_labels |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       656 |        76 |       732 |
             |     0.375 |     2.626 |           |
             |     0.896 |     0.104 |     0.915 |
             |     0.937 |     0.760 |           |
             |     0.820 |     0.095 |           |
-------------|-----------|-----------|-----------|
           1 |        44 |        24 |        68 |
             |     4.038 |    28.265 |           |
             |     0.647 |     0.353 |     0.085 |
             |     0.063 |     0.240 |           |
             |     0.055 |     0.030 |           |
-------------|-----------|-----------|-----------|
Column Total |       700 |       100 |       800 |
             |     0.875 |     0.125 |           |
-------------|-----------|-----------|-----------|
```

We will now add "duration" to the model. A starting threshold probability of 0.5 gave 87.6% (701/800) which is decent. I tried other thresholds and, surprisingly, 0.9 gave a pretty good result of 91.6% (733/800) which is actually the best accuracy out of all models.

```
train <-
newdata[train_sample,c("age","job","contact","month","campaign","poutcome","d
uration")]
```

```
test <-
newdata[-train_sample,c("age","job","contact","month","campaign","poutcome","d
uration")]
bmodel <- naiveBayes(train, train_labels, laplace = 0)
preds <- predict(bmodel,test,type="raw")
predclass <- ifelse(preds[,2] >= .5, 1, 0)
CrossTable(test_labels,predclass)
```

```
Total Observations in Table:  800
             | predclass
 test_labels |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       675 |        57 |       732 |
             |     0.547 |     4.726 |           |
             |     0.922 |     0.078 |     0.915 |
             |     0.941 |     0.687 |           |
             |     0.844 |     0.071 |           |
-------------|-----------|-----------|-----------|
           1 |        42 |        26 |        68 |
             |     5.889 |    50.874 |           |
             |     0.618 |     0.382 |     0.085 |
             |     0.059 |     0.313 |           |
             |     0.052 |     0.032 |           |
-------------|-----------|-----------|-----------|
Column Total |       717 |        83 |       800 |
             |     0.896 |     0.104 |           |
-------------|-----------|-----------|-----------|
```

```
predclass <- ifelse(preds[,2] >= .9, 1, 0)

CrossTable(test_labels,predclass)
```

Total Observations in Table: 800

	predclass		
test_labels	0	1	Row Total
0	722	10	732
	0.119	4.419	
	0.986	0.014	0.915
	0.927	0.476	
	0.902	0.012	
1	57	11	68
	1.282	47.572	
	0.838	0.162	0.085
	0.073	0.524	
	0.071	0.014	
Column Total	779	21	800
	0.974	0.026	