Data Mining & Machine Learning Module Assignment

Mark Hession

A00239393

Table of Contents

Contents

Regression:

1.1

This dataset is about relative CPU performance data, described in terms of its cycle time, memory size, etc. The data we will be using from the data set is CACH: cache memory in kilobytes (integer)CHMIN: minimum channels in units (integer), CHMAX: maximum channels in units (integer), PRP: published relative performance (integer), ERP: estimated relative performance from the original article (integer). We will be predicting the ERP of a system based on its CACH/CHMIN/CHMAX/PRP.

1.2

This is my dataset prior to any alterations.

	ï..Vendor.Name	Model.Name	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP	ERP
1	adviser	32/60	125	256	6000	256	16	128	198	199
2	amdahl	470v/7	29	8000	32000	32	8	32	269	253
3	amdahl	470v/7a	29	8000	32000	32	8	32	220	253
4	amdahl	470v/7b	29	8000	32000	32	8	32	172	253
5	amdahl	470v/7c	29	8000	16000	32	8	16	132	132
6	amdahl	470v/b	26	8000	32000	64	8	32	318	290
7	amdahl	580-5840	23	16000	32000	64	16	32	367	381
8	amdahl	580-5850	23	16000	32000	64	16	32	489	381
9	amdahl	580-5860	23	16000	64000	64	16	32	636	749
10	amdahl	580-5880	23	32000	64000	128	32	64	1144	1238
11	apollo	dn320	400	1000	3000	0	1	2	38	23
12	apollo	dn420	400	512	3500	4	1	6	40	24
13	basf	Jul-65	60	2000	8000	65	1	8	92	70
14	basf	Jul-68	50	4000	16000	65	1	8	138	117

This is my dataset after it is altered to suit the regression process better.

	CACH	CHMIN	CHMAX	PRP	ERP
1	256	16	128	198	199
2	32	8	32	269	253
3	32	8	32	220	253
4	32	8	32	172	253
5	32	8	16	132	132
6	64	8	32	318	290
7	64	16	32	367	381
8	64	16	32	489	381
9	64	16	32	636	749
10	128	32	64	1144	1238
11	0	1	2	38	23
12	4	1	6	40	24
13	65	1	8	92	70
14	65	1	8	138	117

1.3

There is no training/testing data in the way there is for decision tree and knn. Instead, I used data from my dataset to compare to my predicted outcome to compare the accuracy. This is explained in more detail later.

```
# using values from dataset to compare accuracy
dataCheck = data.frame( CACH= 256, CHMIN=16, CHMAX=128, PRP=198)
dataCheck
predict(machine.lm,dataCheck)
#prediction is 196, dataset is 199, within 90% accuracy
```

1.4

```
#model generation
machine.lm= lm(ERP ~ CACH + CHMIN + CHMAX + PRP, data=machine)
summary(machine.lm)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.573668   3.563633  -0.161    0.872
CACH         0.013971   0.095511   0.146    0.884
CHMIN        0.764623   0.554724   1.378    0.170
CHMAX        0.008129   0.139700   0.058    0.954
PRP          0.907108   0.026254  34.552   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 39.89 on 204 degrees of freedom
Multiple R-squared:  0.9349,    Adjusted R-squared:  0.9336
```

From our model generation, where we will try to predict the ERP, we can see our R-Squared value is 0.93, giving us a 93% accuracy for our prediction, which we will test later.

```
> machine.lm$coefficients
(Intercept)        CACH       CHMIN       CHMAX         PRP
-0.57366843  0.01397076  0.76462328  0.00812878  0.90710772
> |
```

1.5

We can see that based on our new data frame inserts of newdata1, we have a predicted ERP of 341.

```
#prediction
newdata1 = data.frame( CACH= 128, CHMIN=8, CHMAX=64, PRP=367)
newdata1
predict(machine.lm,newdata1)
```

```
> predict(machine.lm,newdata1)
       1
340.7604
>
```

When we test our accuracy of our regression, we can take values from our original dataset and see how the ERP compares to our prediction.

```
# using values from dataset to compare accuracy
dataCheck = data.frame( CACH= 256, CHMIN=16, CHMAX=128, PRP=198)
dataCheck
predict(machine.lm,dataCheck)
```

```
> # using values from dataset to compare accuracy
> dataCheck = data.frame( CACH= 256, CHMIN=16, CHMAX=128, PRP=198)
> dataCheck
  CACH CHMIN CHMAX PRP
1  256    16   128 198
> predict(machine.lm,dataCheck)
       1
195.8846
> #prediction is 196, dataset is 199, within 90% accuracy
>
```

	CACH	CHMIN	CHMAX	PRP	ERP
1	256	16	128	198	199

Our dataset had an ERP of 199, our predicted ERP for the same values was 196, which is within the 93% accuracy bounds from before.

1.6

From our results based on our model, we can see when passing in the values CACH, CHMIN, CHMAX, PRP, we have a 93% accurate result, as proved when we inserted data from our dataset and compared it to our prediction result. For our dataset, to be able to predict the ERP based on these values is beneficial to the consumer as it gives the Estimated Real Performance of a CPU, based off the CPUs cache information, and you can compare it to the PRP of the CPU. This gives consumers an accurate estimate of what type of performance they can expect for the CPU they will buy. The biggest take away is the connection between the PRP and the ERP, which seem to have a difference which will scale, depending on the values inserted. The more modest and realistic a value you insert for PRP like in our original dataset you get a more accurate representation of what the ERP will be and how they compare as marketing vs actual performance.

Decision Tree:

2.1 – For this analysis, I have chosen a dataset which is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The goal is to predict if a client will say yes or no to a term deposit subscription. The dataset includes several details such as the client's age/job/education along with banking details such as if they have a loan and what type of contact, they have. The decision tree will show the branching paths based on the client's information and predict how likely they are to say yes or no based on that information.

2.2

Bank Dataset before alterations

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
1	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
2	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
3	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
4	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
5	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no
6	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3	failure	no
7	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2	other	no
8	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0	unknown	no
9	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0	unknown	no

Bank dataset after alterations

	age	marital	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
1	30	married	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
2	33	married	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
3	35	single	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
4	30	married	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
5	59	married	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no
6	35	single	no	747	no	no	cellular	23	feb	141	2	176	3	failure	no
7	36	married	no	307	yes	no	cellular	14	may	341	1	330	2	other	no

```
> bank <- read.csv("data/bank/bank.csv")
> str(bank)
'data.frame':   4521 obs. of  17 variables:
 $ age       : int  30 33 35 30 59 35 36 39 41 43 ...
 $ job       : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
 $ marital   : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
 $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
 $ default   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ balance   : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
 $ housing   : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
 $ loan      : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
 $ contact   : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
 $ day       : int  19 11 16 3 5 23 14 6 14 17 ...
 $ month     : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
 $ duration  : int  79 220 185 199 226 141 341 151 57 313 ...
 $ campaign  : int  1 1 1 4 1 2 1 2 2 1 ...
 $ pdays     : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
 $ previous  : int  0 4 1 0 0 3 2 0 0 2 ...
 $ poutcome  : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
 $ y         : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
>
```

```
> summary(bank)
      age                job          marital        education    default       balance       housing      loan
 Min.   :19.00   management :969   divorced: 528   primary  : 678   no :4445   Min.   :-3313   no :1962   no :3830
 1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306   yes:  76   1st Qu.:   69   yes:2559   yes: 691
 Median :39.00   technician :768   single  :1196   tertiary :1350              Median :  444
 Mean   :41.17   admin.     :478                   unknown  : 187              Mean   : 1423
 3rd Qu.:49.00   services   :417                                               3rd Qu.: 1480
 Max.   :87.00   retired    :230                                               Max.   :71188
                 (Other)    :713
     contact           day             month          duration       campaign          pdays            previous
 cellular :2896   Min.   : 1.00   may    :1398   Min.   :   4   Min.   : 1.000   Min.   : -1.00   Min.   : 0.0000
 telephone: 301   1st Qu.: 9.00   jul    : 706   1st Qu.: 104   1st Qu.: 1.000   1st Qu.: -1.00   1st Qu.: 0.0000
 unknown  :1324   Median :16.00   aug    : 633   Median : 185   Median : 2.000   Median : -1.00   Median : 0.0000
                  Mean   :15.92   jun    : 531   Mean   : 264   Mean   : 2.794   Mean   : 39.77   Mean   : 0.5426
                  3rd Qu.:21.00   nov    : 389   3rd Qu.: 329   3rd Qu.: 3.000   3rd Qu.: -1.00   3rd Qu.: 0.0000
                  Max.   :31.00   apr    : 293   Max.   :3025   Max.   :50.000   Max.   :871.00   Max.   :25.0000
                                  (Other): 571
   poutcome      y
 failure: 490   no :4000
 other  : 197   yes: 521
 success: 129
 unknown:3705
```

2.3

Splitting the data frames.

For the training and testing, I split it 80/20, 80% for training and 20% for testing.

```
set.seed(1)
bank_rand <- bank[order(runif(4521)), ]

# split the data frames
bank_train <- bank_rand[1:3617, ]
bank_test <- bank_rand[3618:4521, ]


#train and predict
library(C50)
model <- C5.0(y ~ ., data = bank_train)
```

2.4

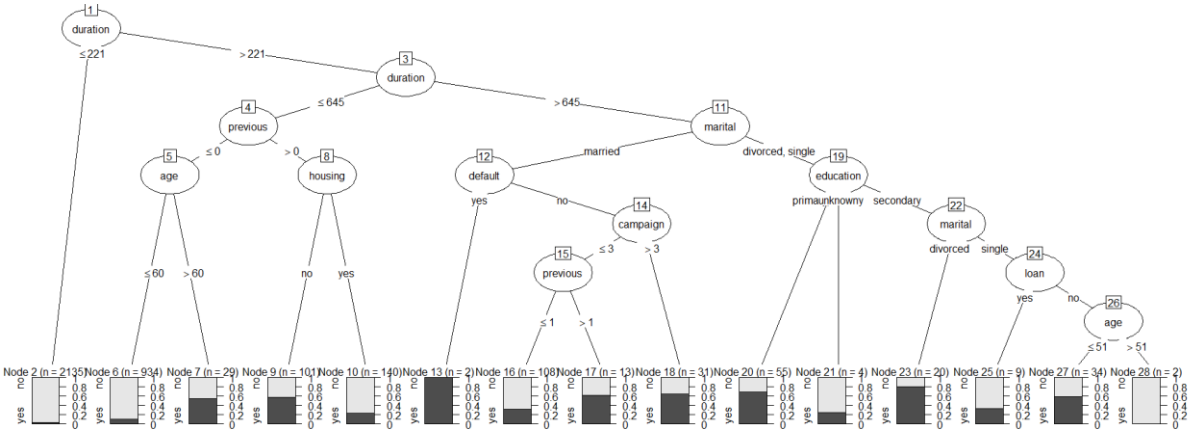
Model

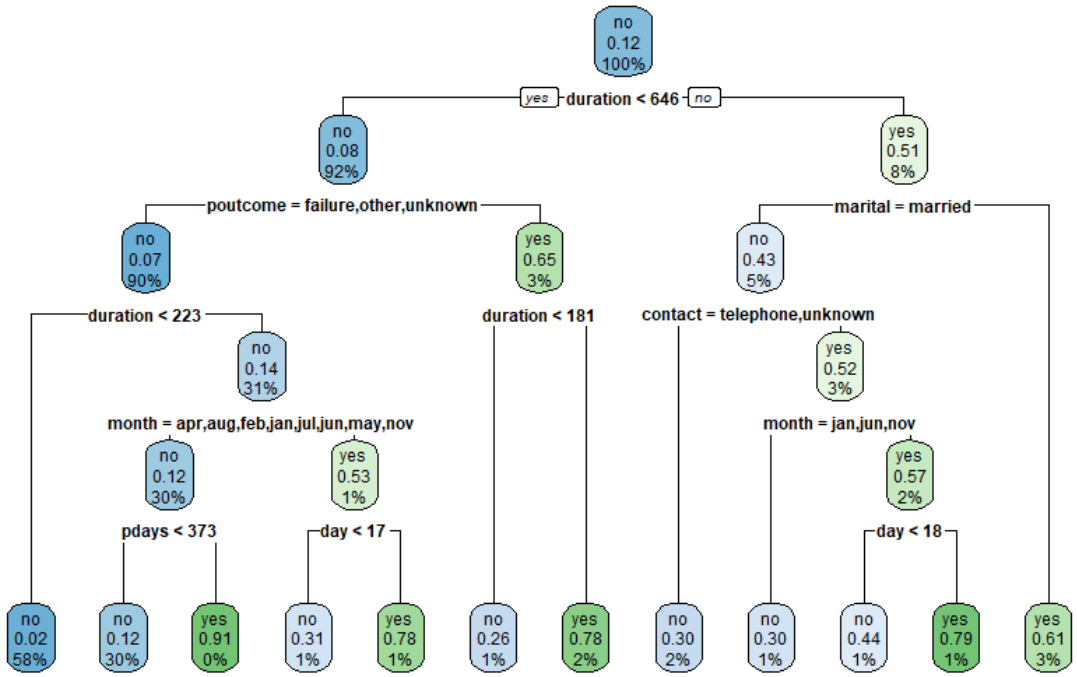
```
# display simple facts about the tree
model
# display detailed information about the tree
summary(model)
# Evaluating model performance
# Create a factor vector of predictions on test data
predictions <- predict(model, bank_test)


# cross tabulation of predicted versus actual classes
library(gmodels)
CrossTable(predictions, bank_test$y,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('predicted y', 'actual y'))
```

```
Call:
C5.0.formula(formula = y ~ ., data = bank_train)

Classification Tree
Number of samples: 4421
Number of predictors: 14

Tree size: 38
```

2.5

Predictions





Total Observations in Table: 100

predicted y	actual y: no	actual y: yes	Row Total
no	87 0.870	8 0.080	95
yes	2 0.020	3 0.030	5
Column Total	89	11	100

2.6

Evaluation

From our prediction, we can see the branching paths on the decision tree, depending on certain data per client we can see how that influences the outcome. An example is people who had a contract duration over 646 and were married said yes and they made up 3% of the total amount of people in the dataset, but those that were unmarried could be split into two groups, those who had a campaign lower or higher than 4, being 3% no and 1% yes.

The interesting data in my opinion, is seeing the difference between those who are married/divorced/single. Married people typically said yes, especially those who already had credit. Those without credit weren't as overwhelmingly responsive to the term deposit but even with factoring in their campaign and previous contact amount, they were just over 50% likely to say yes. This is presumably new couples looking for finance for a home together or maybe raise and fund having children.

There is also a correlation between the age of the client and their likelihood of taking on a term deposit. Those who were above or equal to 60 years were over 90% likely to say no, while those under 60 were approximately likely to say no 45% of the time, though this is dived into deeper when we factor in their marital status and if they already have a loan. Those who were single, had no personal loan and were older than 51 years of age were 99% likely to say no but those under or equal to 51% were around 55% likely to say yes.

KNN:

3.1

For my KNN prediction, I used the same dataset as decision trees, to compare the results given for each. Since there is no model for KNN, I will only be able to compare the reported number of success/failures and compare them to the paths of the decision tree. We are still trying to predict the likelihood of a yes or no response from a banks client on if they would like a term deposit or not. To predict this, we use the data from the dataset of clients, a mix of yes and no responses.

3.2

Dataset before alterations

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown
2	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure
3	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure
4	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown
5	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown
6	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3	failure
7	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2	other
8	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0	unknown

Dataset after alterations

	y	age	balance	day	duration	campaign	pdays	previous
1	no	30	1787	19	79	1	-1	0
2	no	33	4789	11	220	1	339	4
3	no	35	1350	16	185	1	330	1
4	no	30	1476	3	199	4	-1	0
5	no	59	0	5	226	1	-1	0
6	no	35	747	23	141	2	176	3
7	no	36	307	14	341	1	330	2
8	no	39	147	6	151	2	-1	0
9	no	41	221	14	57	2	-1	0

I removed all non-numerical columns and reordered so the “y” column, which is soon to be my success/failure column, is at the front, for simplicity sake down the line.

```
> table(bank$y)

  no  yes
4000  521
> 
```

```
> # recode y as a factor
> bank$y <- factor(bank$y,
+                          levels = c("yes", "no"),
+                          labels = c("Success", "Fail"))
> 
```

	y
1	Fail
2	Fail
3	Fail
4	Fail
5	Fail
6	Fail
7	Fail
8	Fail
9	Fail
10	Fail
11	Fail
12	Fail
13	Fail
14	Success

Creating the normalize function

```
> # create normalization function
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x)))
+ }
>
>
> # test normalization function - result should be identical
> normalize(c(1, 2, 3, 4, 5))
[1] 0.00 0.25 0.50 0.75 1.00
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
> 
```

```
> # note doesn't include the labels
> bank_n <- as.data.frame(lapply(bank[2:8], normalize))
> summary(bank_n$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.2059  0.2941  0.3260  0.4412  1.0000
> 
```

3.3

For the training and testing split, I decided to go 80/20 again like for decision tree so I can more accurately compare the results.

```
# create training and test data (no labels)
bank_train <- bank_n[1:3617, ]
bank_test <- bank_n[3618:4521, ]


# create labels for training and test data
bank_train_labels <- bank[1:3617, 1]
bank_test_labels <- bank[3618:4521, 1]
```

```
## Step 3: Training a model on the data ----
library(class)
predictions <- knn(train = bank_train, test =
                     bank_test, cl = bank_train_labels, k=21)
```

3.4

Model

Knn does not have a model. It is about finding the best K value. I tried out multiple K values before deciding on 30, which is roughly the square root of 904 (the number of observations), which is typically the best value to use.

```
# k=1
predictions <- knn(train = bank_train, test = bank_test,
                   cl = bank_train_labels, k=1)
CrossTable(predictions, bank_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)

# k=5
predictions <- knn(train = bank_train, test = bank_test,
                   cl = bank_train_labels, k=5)
CrossTable(predictions, bank_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)

# k=11
predictions <- knn(train = bank_train, test = bank_test,
                   cl = bank_train_labels, k=11)
CrossTable(predictions, bank_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)

# k=17
predictions <- knn(train = bank_train, test = bank_test,
                   cl = bank_train_labels, k=17)
CrossTable(predictions, bank_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)
```

3.5

K=30 gave me the highest accuracy with 87.2%

```
## Step 3: Training a model on the data ----
# square root of 904 is roughly 30 so that is our K value.
library(class)
predictions <- knn(train = bank_train, test =
                     bank_test, cl = bank_train_labels, k=30)

## Step 4: Evaluating model performance ----
# load the "gmodels" library
#install.packages("gmodels")
library(gmodels)

# Create the cross tabulation of predicted vs. actual
CrossTable(predictions, bank_test_labels,
           prop.chisq = FALSE,
           prop.c = FALSE, prop.r = FALSE)
```

```
Total Observations in Table:  904

             | bank_test_labels
 predictions |   Success |      Fail | Row Total |
-------------|-----------|-----------|-----------|
     Success |         8 |         9 |        17 |
             |     0.009 |     0.010 |           |
-------------|-----------|-----------|-----------|
        Fail |       103 |       784 |       887 |
             |     0.114 |     0.867 |           |
-------------|-----------|-----------|-----------|
Column Total |       111 |       793 |       904 |
-------------|-----------|-----------|-----------|
```

Some other cross table results with various K values:

K=1

```
Total Observations in Table:  904


             | bank_test_labels
 predictions |   Success |      Fail | Row Total |
-------------|-----------|-----------|-----------|
     Success |        33 |        72 |       105 |
             |     0.037 |     0.080 |           |
-------------|-----------|-----------|-----------|
        Fail |        78 |       721 |       799 |
             |     0.086 |     0.798 |           |
-------------|-----------|-----------|-----------|
Column Total |       111 |       793 |       904 |
-------------|-----------|-----------|-----------|
```

K=11

```
Total Observations in Table:  904

             | bank_test_labels
 predictions |   Success |      Fail | Row Total |
-------------|-----------|-----------|-----------|
     Success |        14 |        12 |        26 |
             |     0.015 |     0.013 |           |
-------------|-----------|-----------|-----------|
        Fail |        97 |       781 |       878 |
             |     0.107 |     0.864 |           |
-------------|-----------|-----------|-----------|
Column Total |       111 |       793 |       904 |
-------------|-----------|-----------|-----------|
```

K=27

```
Total Observations in Table:  904

             | bank_test_labels
 predictions |   Success |      Fail | Row Total |
-------------|-----------|-----------|-----------|
     Success |         8 |        12 |        20 |
             |     0.009 |     0.013 |           |
-------------|-----------|-----------|-----------|
        Fail |       103 |       781 |       884 |
             |     0.114 |     0.864 |           |
-------------|-----------|-----------|-----------|
Column Total |       111 |       793 |       904 |
-------------|-----------|-----------|-----------|
```

3.6

Evaluation.

With KNN, it is difficult to define the concepts, you just must take the result you get and see what you can observe from it. When we look at the cross table result for this dataset, we can see a significant amount of failures compared to successes. This is in line with what we expected based on our decision tree prediction, which had far more failures than successes and we could path what factors lead to choosing one over the other. When we calculate our accuracy by adding (8+781)/904 and multiply that by 100, we can see we have 87.2%, which is a very good accuracy rating.