Data-Driven Approach To Predict Success Of Bank Telemarketing

Nisha Selvarajan

10/24/2020

Contents

Objectives

Data-Driven Approach To Predict Success Of Bank Telemarketing

Nowadays, marketing spending in the banking industry is massive, meaning that it is essential for banks to optimize marketing strategies and improve effectiveness. Understanding customers' need leads to more effective marketing plans, smarter product designs and greater customer satisfaction.The main objective of this project is to increase the effectiveness of the bank's telemarketing campaign.This project will enable the bank to develop a more granular understanding of its customer base, predict customers' response to its telemarketing campaign and establish a target customer profile for future marketing plans.

By analyzing customer features, such as demographics and transaction history, the bank will be able to predict customer saving behaviours and identify which type of customers is more likely to make term deposits. The bank can then focus its marketing efforts on those customers. This will not only allow the bank to secure deposits more effectively but also increase customer satisfaction by reducing undesirable advertisements for certain customers.

We are given the data of direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (target variable y). The goal here is to model the probability of buying, as a function of the customer features.

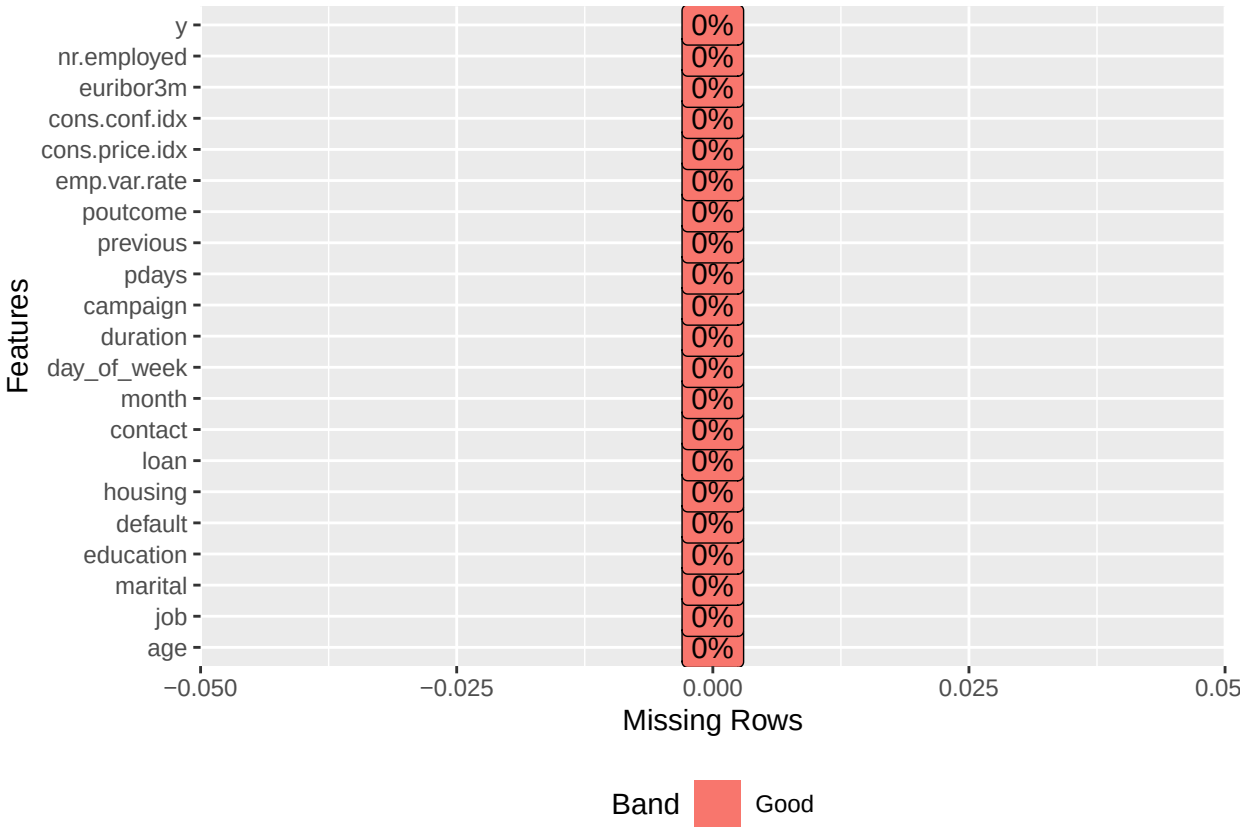
Data Description

- Data set which is utilized for this research has been taken from University of California, Irvine machine learning repository (https://archive.ics.uci.edu/ml/datasets/Bank+Marketing?package=regsel&version=0.2) which is openly available for the public for research purpose. The dataset contains 41188 marketing campaigns observations with 20 input features. The details of 20 attributes are following.

Names	Description
age	Numeric - Age of the client
job	Categorical - Type of Job
marital	Categorical - Marital Status of Client
education	Categorical - Education qualification of client
default	Categorical - Has credit in default?
housing	Categorical - Has housing loan?
loan	Categorical - Has personal loan?
contact	Categorical - Contact like cellular,telephone
month	Categorical - Last Contact Month of Year
day_of_week	Categorical - Last Contact Day of the Week
duration	Numerical - Last Contact Duration in Seconds
campaign	Numerical - No of contacts performed for Campaign
pdays	Numerical - No of days passed after previous campaign contact.
previous	Numerical - No of contacts performed before this campaign for this client
poutcome	Categorical - Outcome of the previous marketing campaign
emp.var.rate	Numerical - Employment Variation Rate - quarterly indicator
cons.price.idx	Numerical - Consumer price index - monthly indicator
cons.conf.idx	Numerical - Consumer confidence index - monthly indicator
euribor3m	Numerical- Euribor 3 month rate - daily indicator
nr.employed	Numerical- No of employees - quarterly indicator
y	Categorical- Has the client subscribed a term deposit?

Data Analysis

- Plot missing values of all the features in the dataset.



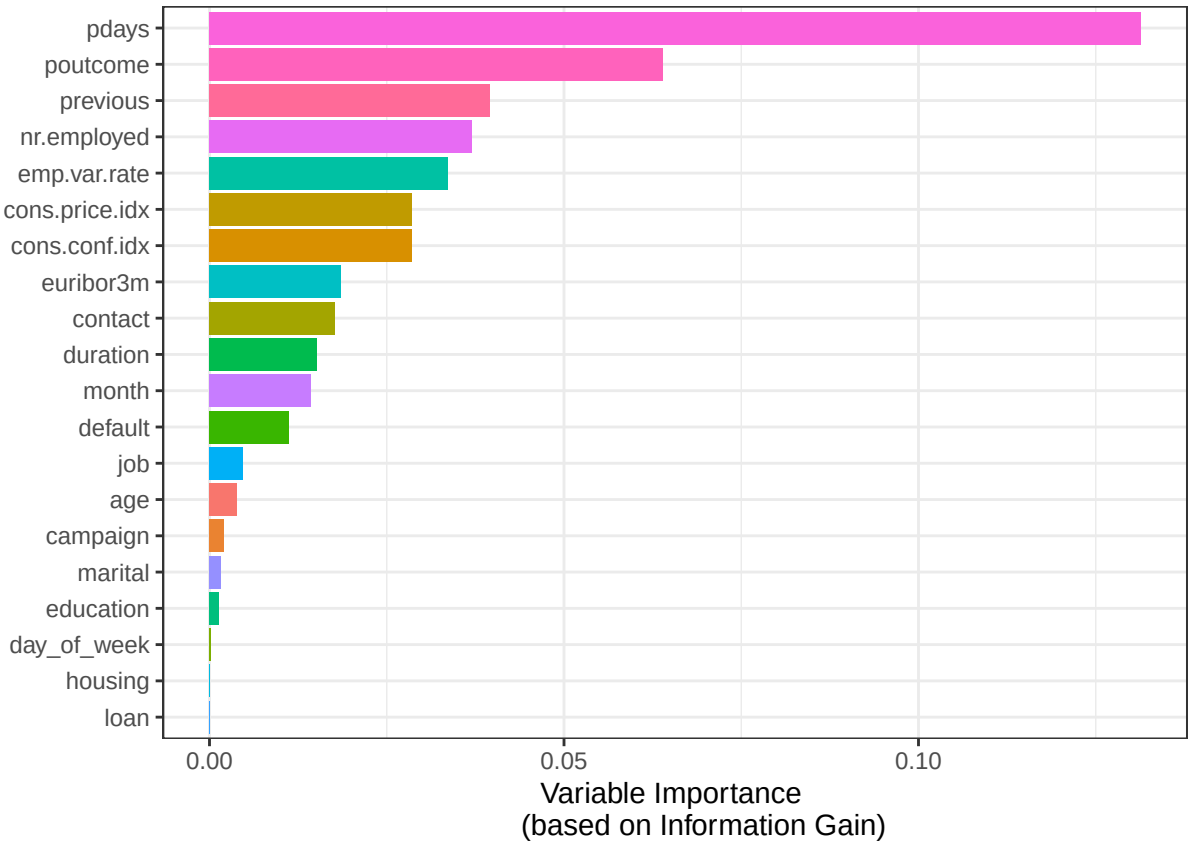
- Ploting histograms for numerical variables.

- Get metric table with many indicators for all numerical variables, automatically skipping the non-numerical variables.

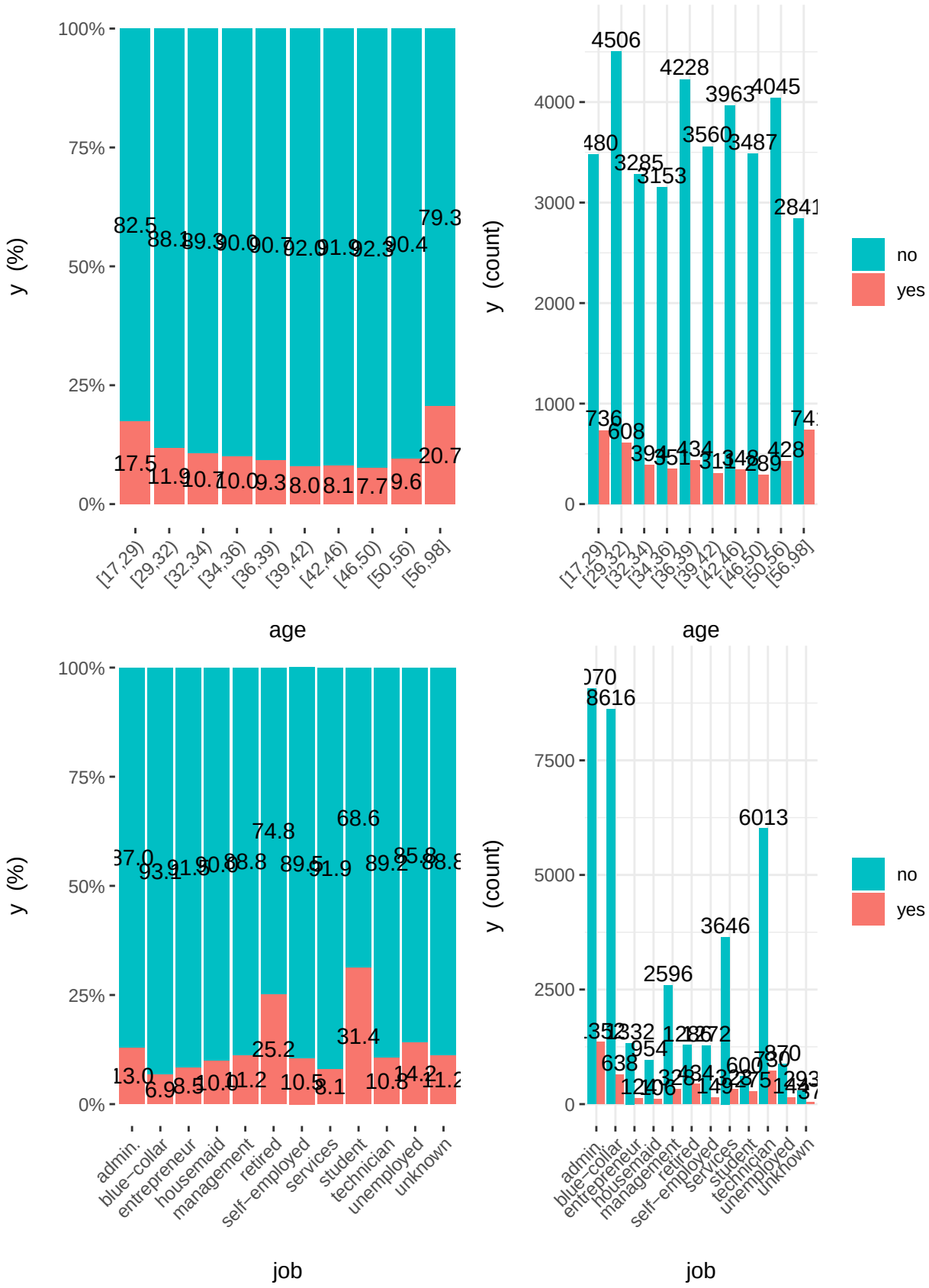
```
##          variable         mean     std_dev variation_coef       p_01     p_05
## 1             age   40.0240604  10.4212500    0.260374632   23.00000   26.000
## 2        duration  258.2850102 259.2792488    1.003849386   11.00000   36.000
## 3        campaign    2.5675925   2.7700135    1.078836903    1.00000    1.000
## 4           pdays  962.4754540 186.9109073    0.194198103    3.00000  999.000
## 5        previous    0.1729630   0.4949011    2.861311858    0.00000    0.000
## 6    emp.var.rate    0.0818855   1.5709597   19.184834048   -3.40000   -2.900
## 7  cons.price.idx   93.5756644   0.5788400    0.006185797   92.20100   92.713
## 8   cons.conf.idx  -40.5026003   4.6281979   -0.114269154  -49.50000  -47.100
## 9       euribor3m    3.6212908   1.7344474    0.478958331    0.65848    0.797
## 10    nr.employed 5167.0359109  72.2515277    0.013983167 4963.60000 5017.500
##        p_25     p_50     p_75     p_95     p_99   skewness  kurtosis     iqr
## 1    32.000   38.000   47.000   58.000   71.000  0.7846682  3.791070  15.000
## 2   102.000  180.000  319.000  752.650 1271.130  3.2630224 23.245334 217.000
## 3     1.000    2.000    3.000    7.000   14.000  4.7623333 39.975160   2.000
## 4   999.000  999.000  999.000  999.000  999.000 -4.9220107 25.226619   0.000
## 5     0.000    0.000    0.000    1.000    2.000  3.8319027 23.106230   0.000
## 6    -1.800    1.100    1.400    1.400    1.400 -0.7240692  1.937352   3.200
## 7    93.075   93.749   93.994   94.465   94.465 -0.2308792  2.170146   0.919
## 8   -42.700  -41.800  -36.400  -33.600  -26.900  0.3031688  2.641340   6.300
## 9     1.344    4.857    4.961    4.966    4.968 -0.7091621  1.593222   3.617
## 10 5099.100 5191.000 5228.100 5228.100 5228.100 -1.0442244  2.996094 129.000
##            range_98         range_80
## 1          [23, 71]         [28, 55]
## 2     [11, 1271.13]        [59, 551]
## 3           [1, 14]           [1, 5]
## 4          [3, 999]       [999, 999]
## 5            [0, 2]           [0, 1]
## 6       [-3.4, 1.4]      [-1.8, 1.4]
## 7  [92.201, 94.465] [92.893, 94.465]
## 8    [-49.5, -26.9]   [-46.2, -36.1]
## 9  [0.65848, 4.968]   [1.046, 4.964]
## 10 [4963.6, 5228.1] [5076.2, 5228.1]
```

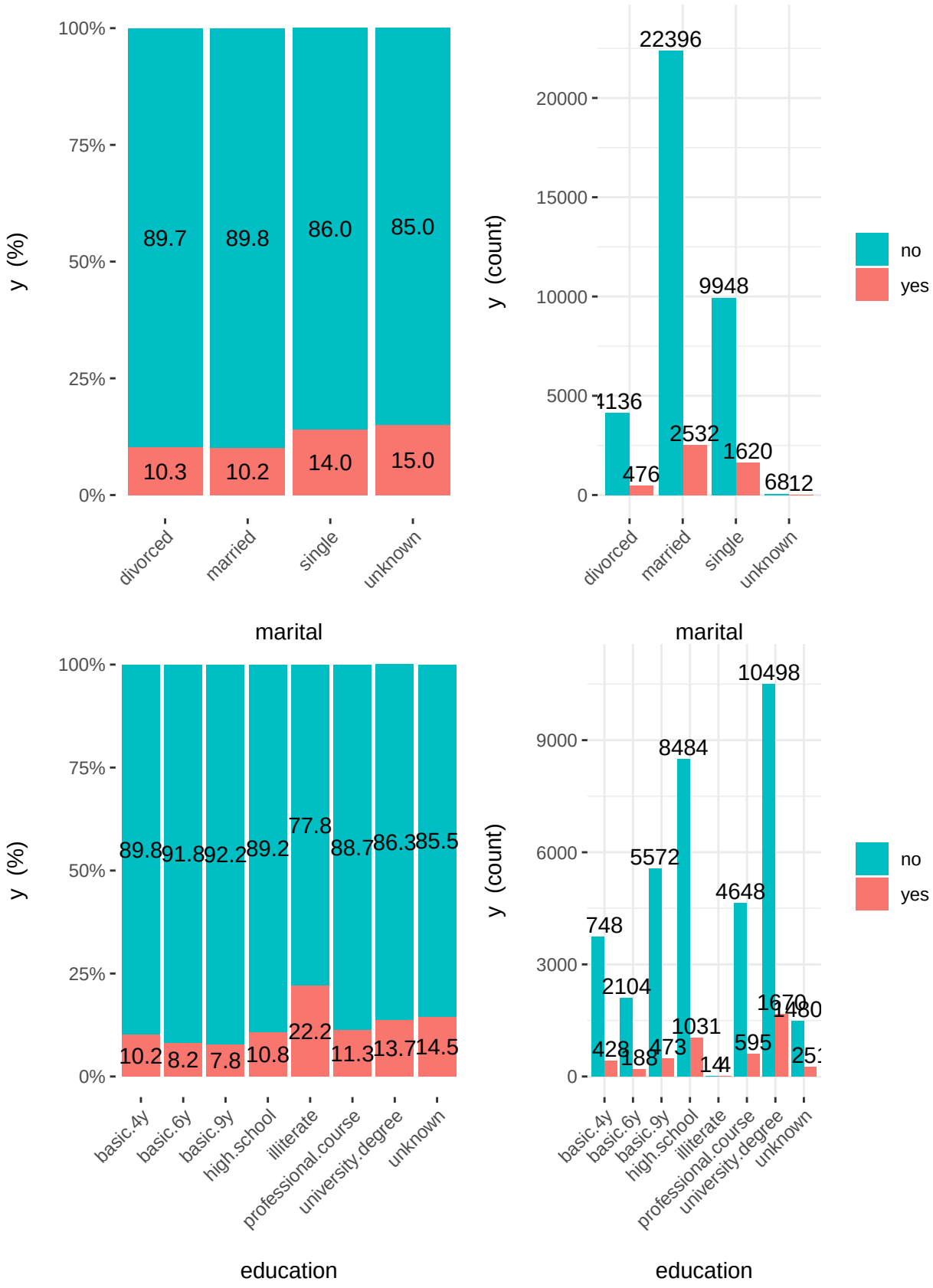
Variable Importance & Crossplot to Deposit

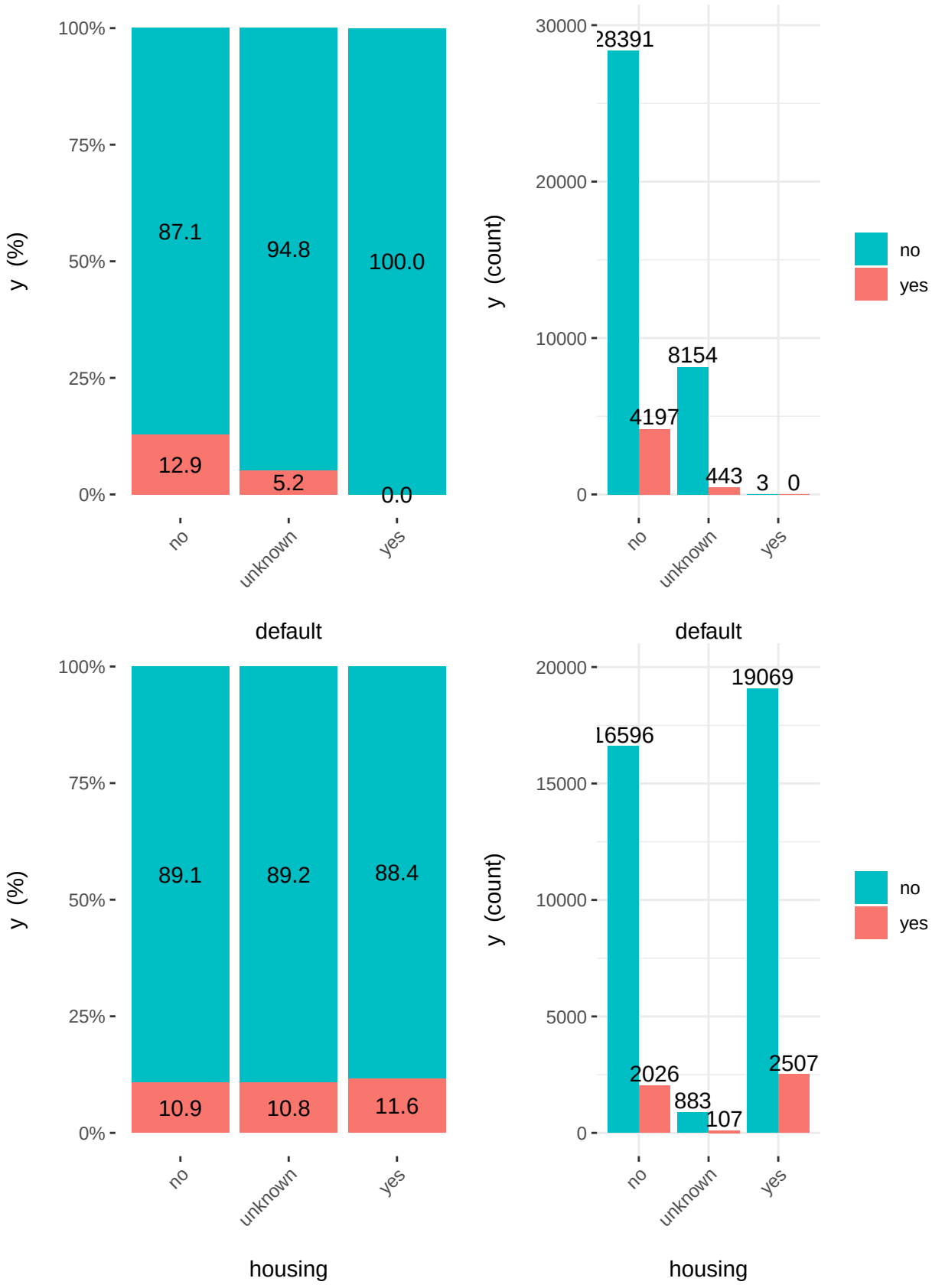
- Plot variable importance with several metrics such as entropy (en), mutual information(mi), information gain (ig) and gain ratio (gr).

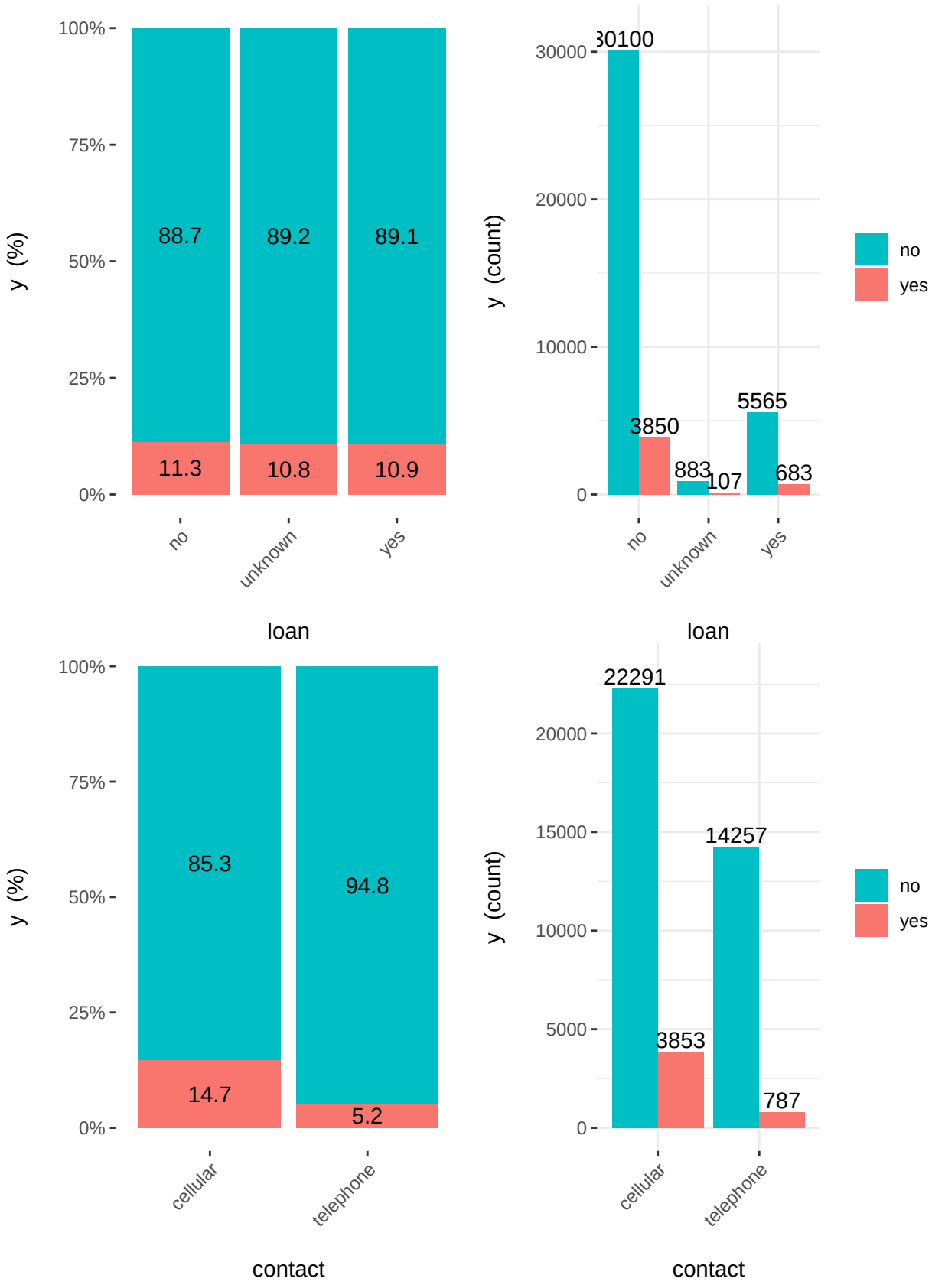


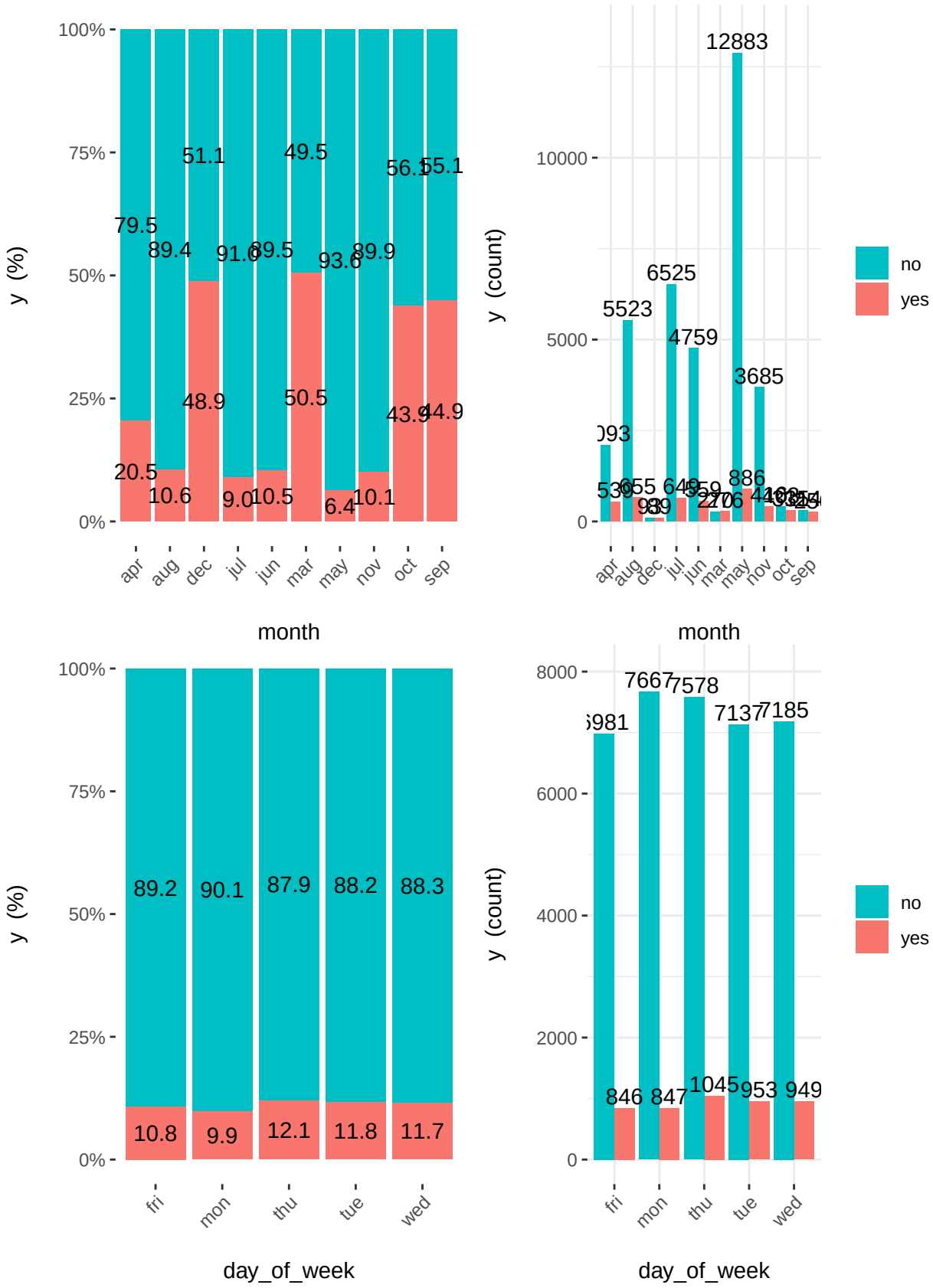
- Bivariate analysis crosss plot showing relationship of each and every variable with respect to target variable

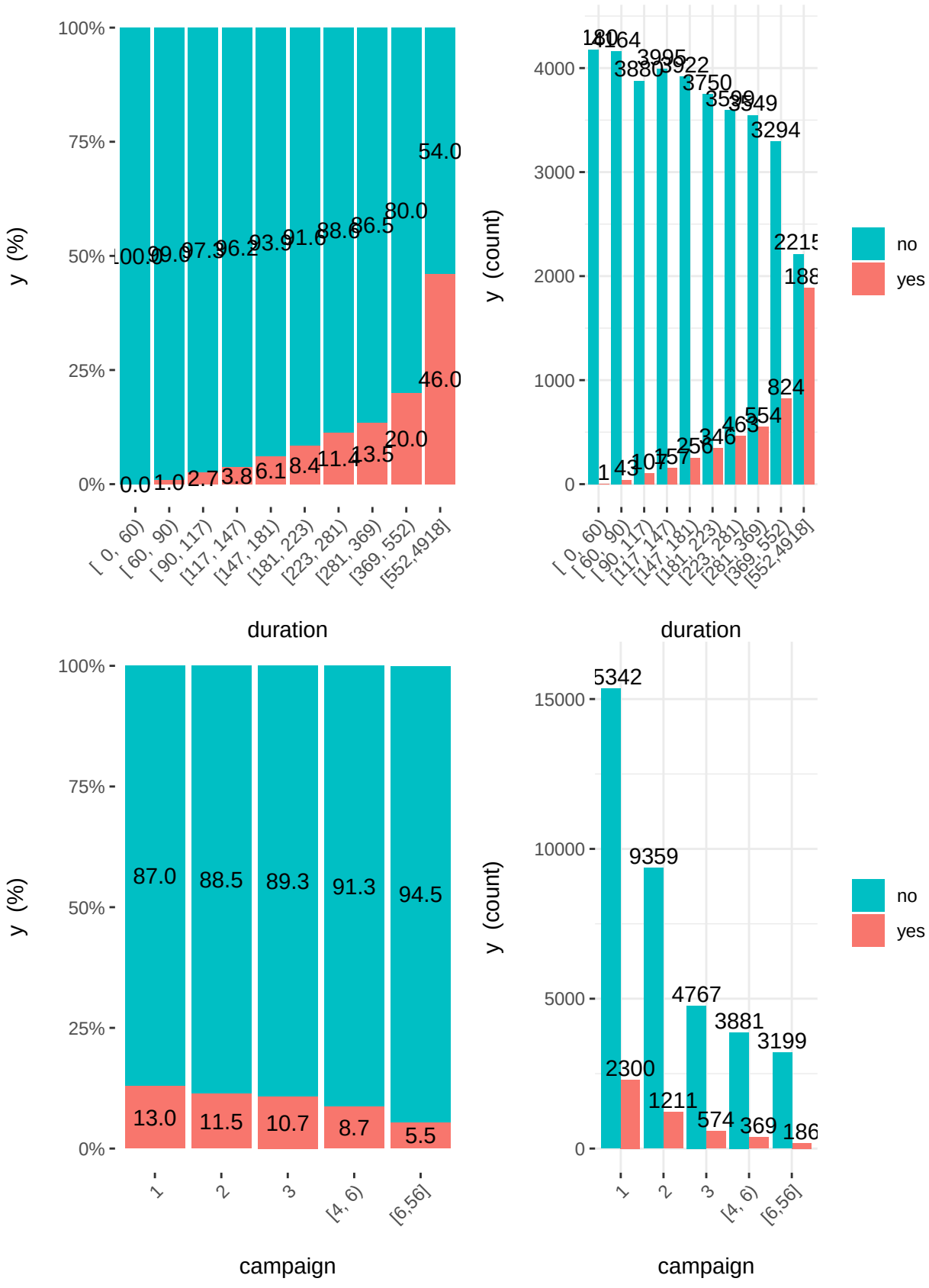

100%
75%
50%
25%
0%
y (%)
82.5
88.1
89.3
90.0
90.7
92.0
91.9
92.3
90.4
79.3
17.5
11.9
10.7
10.0
9.3
8.0
8.1
7.7
9.6
20.7
[17,29)
[29,32)
[32,34)
[34,36)
[36,39)
[39,42)
[42,46)
[46,50)
[50,56)
[56,98]
age
4000
3000
2000
1000
0
y (count)
4506
4228
3963
4045
3560
3487
3285
3153
480
2841
736
608
394
351
434
313
348
289
428
74
age
no
yes
100%
75%
50%
25%
0%
y (%)
37.0
93.1
91.9
90.6
88.8
74.8
89.5
91.9
68.6
89.2
85.8
88.8
13.0
6.9
8.5
10.0
11.2
25.2
10.5
8.1
31.4
10.8
14.2
11.2
admin.
blue-collar
entrepreneur
housemaid
management
retired
self-employed
services
student
technician
unemployed
unknown
job
7500
5000
2500
0
y (count)
8616
6013
3646
2596
954
638
870
admin.
blue-collar
entrepreneur
housemaid
management
retired
self-employed
services
student
technician
unemployed
unknown
job
no
yes

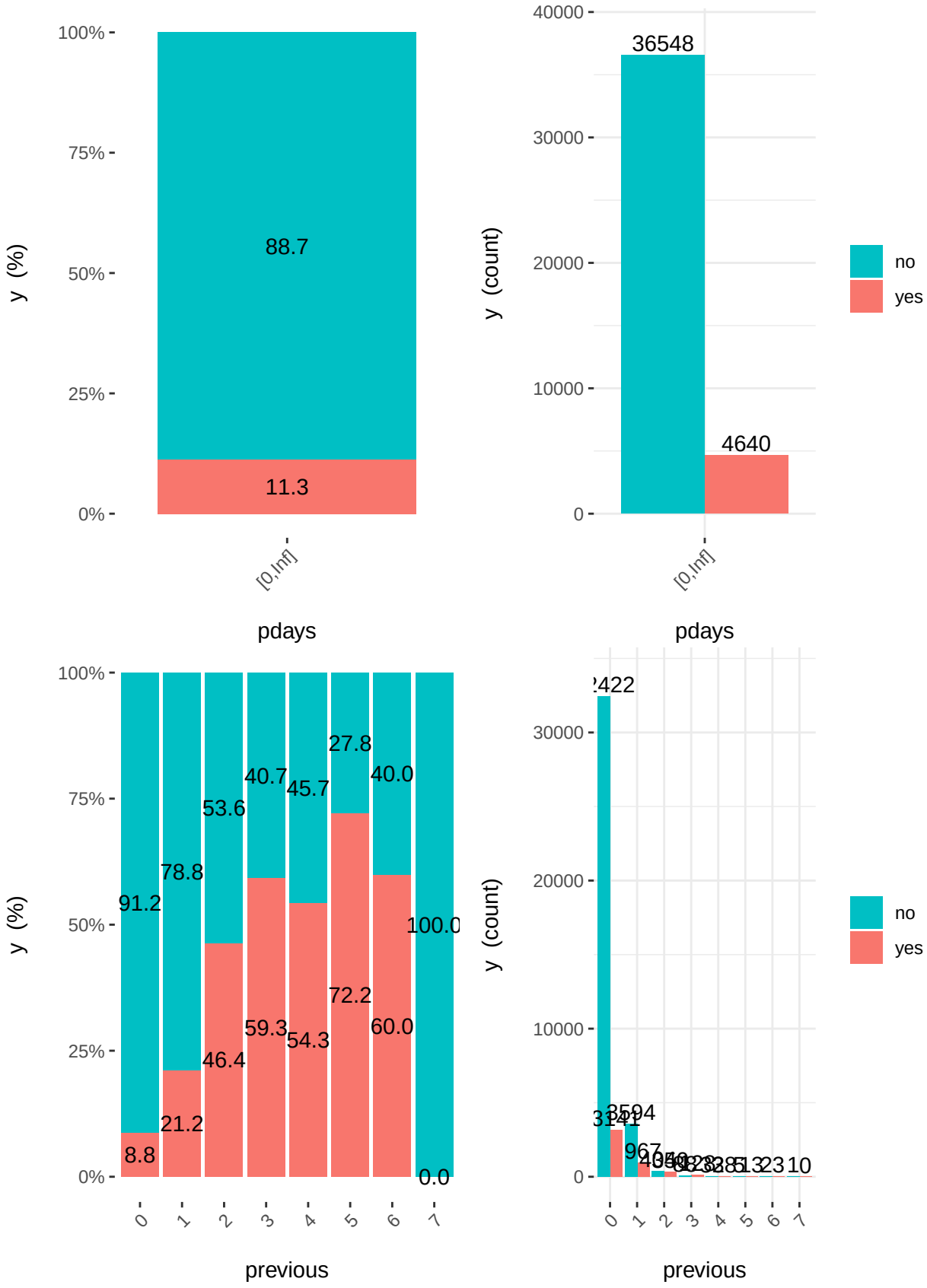

100%
75%
50%
25%
0%
y (%)
89.7
89.8
86.0
85.0
10.3
10.2
14.0
15.0
divorced
married
single
unknown
marital
y (count)
20000
15000
10000
5000
0
22396
9948
1136
2532
1620
476
6812
no
yes
89.8
91.8
92.2
89.2
77.8
88.7
86.3
85.5
10.2
8.2
7.8
10.8
22.2
11.3
13.7
14.5
basic.4y
basic.6y
basic.9y
high.school
illiterate
professional.course
university.degree
unknown
education
9000
6000
3000
10498
8484
5572
4648
748
2104
1031
595
428
188
473
144
1670
1480

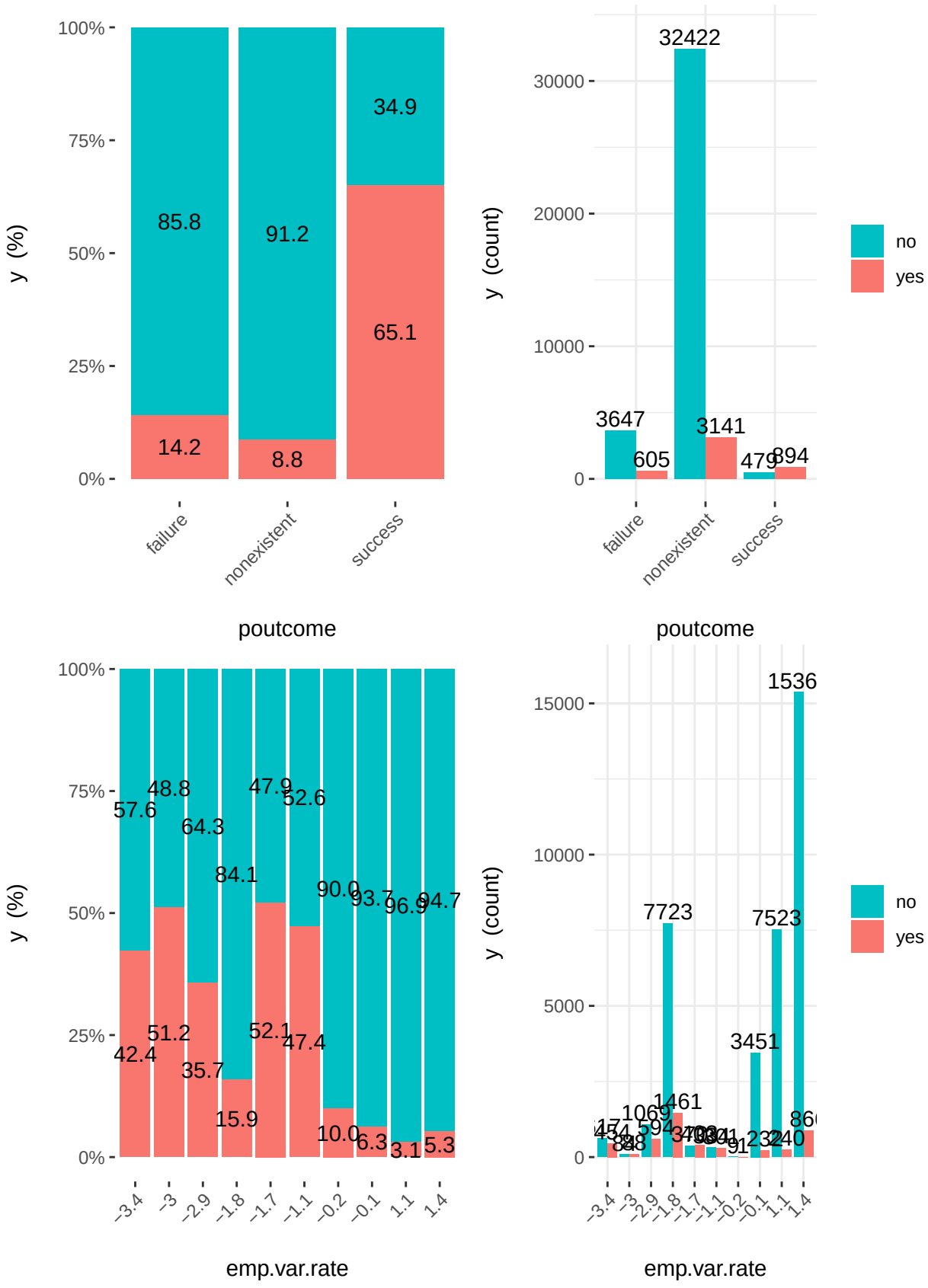

100%
75%
50%
25%
0%
y (%)
87.1
94.8
100.0
12.9
5.2
0.0
no
unknown
yes
default
30000
20000
10000
0
y (count)
28391
4197
8154
443
3
0
no
unknown
yes
default
no
yes
100%
75%
50%
25%
0%
y (%)
89.1
89.2
88.4
10.9
10.8
11.6
no
unknown
yes
housing
20000
15000
10000
5000
0
y (count)
16596
2026
883
107
19069
2507
no
unknown
yes
housing
no
yes

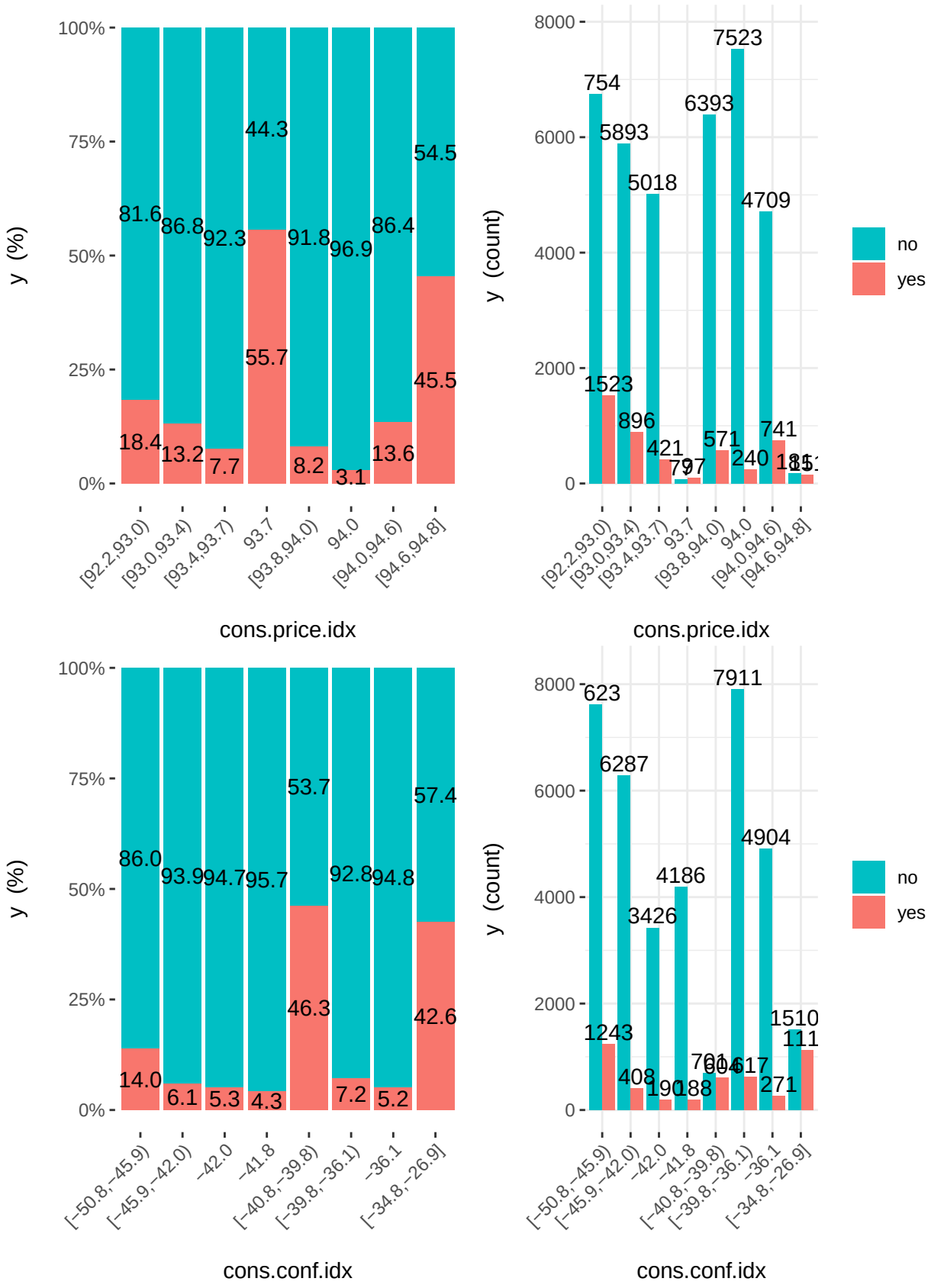

100%
75%
50%
25%
0%
y (%)
88.7
89.2
89.1
11.3
10.8
10.9
no
unknown
yes
loan
30000
20000
10000
0
y (count)
30100
3850
883
107
5565
683
no
yes
loan
100%
75%
50%
25%
0%
y (%)
85.3
94.8
14.7
5.2
cellular
telephone
contact
22291
20000
15000
10000
5000
0
y (count)
14257
3853
787
no
yes
cellular
telephone
contact

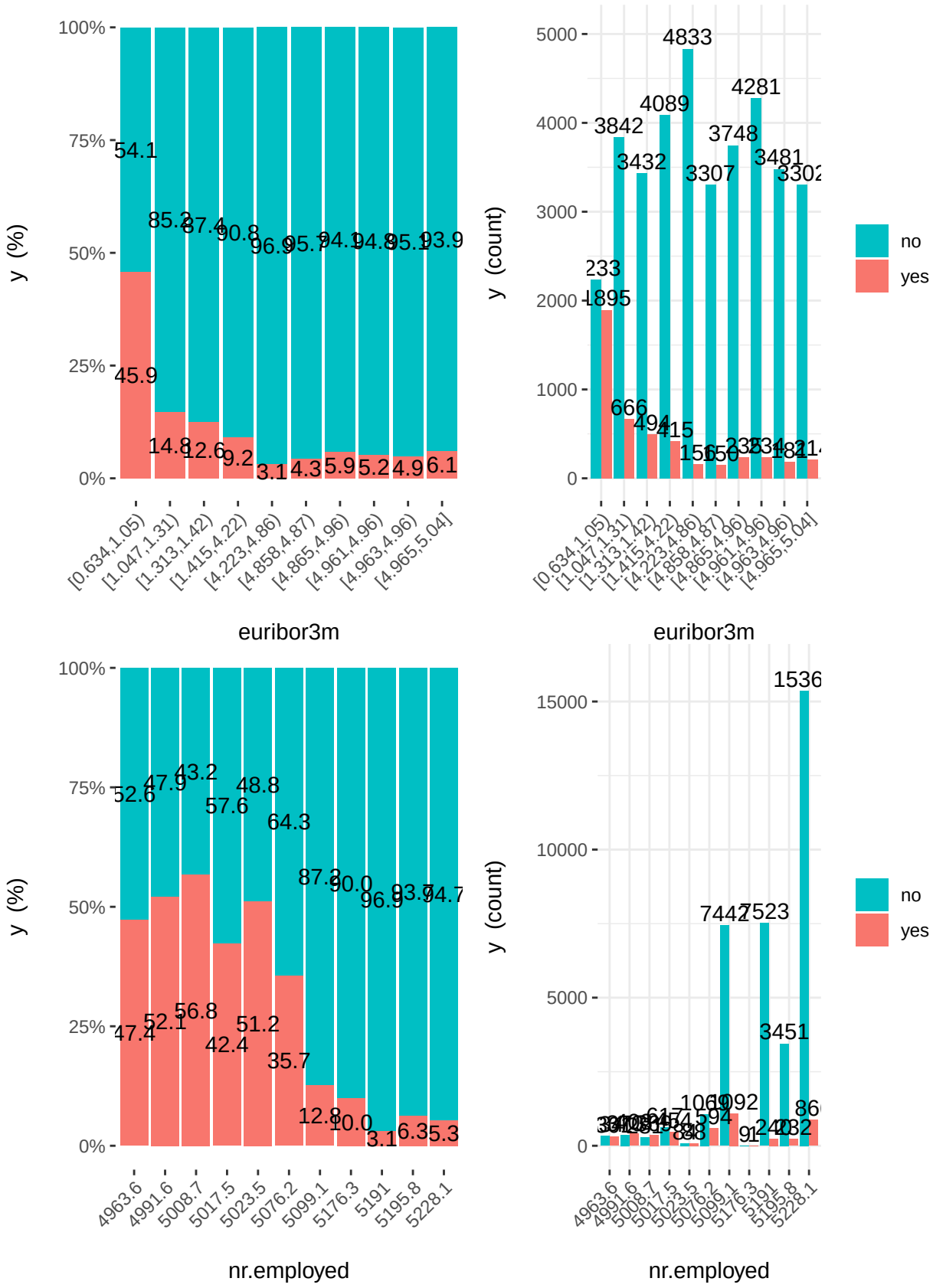

100%
75%
50%
25%
0%
y (%)
79.5
89.4
51.1
91.0
89.5
49.5
93.6
89.9
56.5
55.1
20.5
10.6
48.9
9.0
10.5
50.5
6.4
10.1
43.9
44.9
apr
aug
dec
jul
jun
mar
may
nov
oct
sep
month
y (count)
10000
5000
0
12883
6525
5523
4759
3685
886
no
yes
month
100%
75%
50%
25%
0%
y (%)
89.2
90.1
87.9
88.2
88.3
10.8
9.9
12.1
11.8
11.7
fri
mon
thu
tue
wed
day_of_week
8000
6000
4000
2000
0
y (count)
7667
7578
7137
7185
846
847
1045
953
949
no
yes
day_of_week


100%
75%
50%
25%
0%
y (%)
100.0
99.0
97.3
96.2
93.9
91.6
88.6
86.5
80.0
54.0
0.0
1.0
2.7
3.8
6.1
8.4
11.4
13.5
20.0
46.0
[0, 60)
[60, 90)
[90, 117)
[117, 147)
[147, 181)
[181, 223)
[223, 281)
[281, 369)
[369, 552)
[552,4918]
duration
4000
3000
2000
1000
0
y (count)
3880
3995
3750
3294
824
554
463
346
256
157
107
43
1
no
yes
duration
100%
75%
50%
25%
0%
y (%)
87.0
88.5
89.3
91.3
94.5
13.0
11.5
10.7
8.7
5.5
1
2
3
[4, 6)
[6,56]
campaign
15000
10000
5000
0
y (count)
5342
9359
4767
3881
3199
2300
1211
574
369
186
no
yes
campaign


100%
75%
50%
25%
0%
y (%)
88.7
11.3
[0,Inf]
pdays
40000
30000
20000
10000
0
y (count)
36548
4640
[0,Inf]
pdays
no
yes
100%
75%
50%
25%
0%
y (%)
91.2
8.8
78.8
21.2
53.6
46.4
40.7
59.3
45.7
54.3
27.8
72.2
40.0
60.0
100.0
0.0
0
1
2
3
4
5
6
7
previous
30000
20000
10000
0
y (count)
3422
3594
967
0
1
2
3
4
5
6
7
previous
no
yes


100%
75%
50%
25%
0%
y (%)
85.8
14.2
91.2
8.8
34.9
65.1
failure
nonexistent
success
poutcome
30000
20000
10000
0
y (count)
3647
605
32422
3141
479
894
no
yes
poutcome
57.6
42.4
48.8
51.2
64.3
35.7
84.1
15.9
47.9
52.1
52.6
47.4
90.0
10.0
93.7
6.3
96.9
3.1
94.7
5.3
−3.4
−3
−2.9
−1.8
−1.7
−1.1
−0.2
−0.1
1.1
1.4
emp.var.rate
15000
10000
5000
0
7723
1461
1069
594
3451
7523
1536
91
232
240
no
yes
emp.var.rate


y (%)
100%
75%
50%
25%
0%
81.6
86.8
92.3
44.3
91.8
96.9
86.4
54.5
18.4
13.2
7.7
55.7
8.2
3.1
13.6
45.5
[92.2,93.0)
[93.0,93.4)
[93.4,93.7)
93.7
[93.8,94.0)
94.0
[94.0,94.6)
[94.6,94.8]
cons.price.idx
y (count)
8000
6000
4000
2000
0
754
5893
5018
6393
7523
4709
1523
896
421
741
240
no
yes
cons.price.idx
y (%)
86.0
93.9
94.7
95.7
53.7
92.8
94.8
57.4
14.0
6.1
5.3
4.3
46.3
7.2
5.2
42.6
[−50.8,−45.9)
[−45.9,−42.0)
−42.0
−41.8
[−40.8,−39.8)
[−39.8,−36.1)
−36.1
[−34.8,−26.9]
cons.conf.idx
y (count)
623
6287
3426
4186
7911
4904
1243
408
1510
111
271
no
yes
cons.conf.idx


100%
75%
50%
25%
0%
y (%)
54.1
85.2
87.4
90.8
96.9
95.7
94.1
94.8
95.1
93.9
45.9
14.8
12.6
9.2
3.1
4.3
5.9
5.2
4.9
6.1
[0.634,1.05)
[1.047,1.31)
[1.313,1.42)
[1.415,4.22)
[4.223,4.86)
[4.858,4.87)
[4.865,4.96)
[4.961,4.96)
[4.963,4.96)
[4.965,5.04]
euribor3m
5000
4000
3000
2000
1000
0
y (count)
4833
4281
4089
3842
3748
3481
3432
3307
3302
233
1895
666
494
415
156
150
no
yes
100%
75%
50%
25%
0%
y (%)
52.6
47.9
43.2
57.6
48.8
64.3
87.3
90.0
96.9
93.7
94.7
47.4
52.1
56.8
42.4
51.2
35.7
12.8
10.0
3.1
6.3
5.3
4963.6
4991.6
5008.7
5017.5
5023.5
5076.2
5099.1
5176.3
5191
5195.8
5228.1
nr.employed
15000
10000
5000
0
y (count)
1536
7442
7523
3451
1092
no
yes

Prepare Data for Classification

- Select variables relevant to customers:Based on the variable importance, we will use pdays, poutcome,previous, duration, cons.price.idx,cons.conf.idx,contact feature for further analysis.

```
## 'data.frame':    41188 obs. of  8 variables:
##  $ Term_Deposit                      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
##  $ NumberOfDaysPassedAfterLastContact: num  999 999 999 999 999 999 999 999 999 999 ...
##  $ PreviousMarketingOutCome          : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ NoOfContactsPerformed             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ LastContactDuration               : num  261 149 226 151 307 198 139 217 380 50 ...
##  $ ContactCommunicationType          : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ ConsumerPriceIndex                : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -3
##  $ ConsumerConfidenceIndex           : num  94 94 94 94 94 ...
```

- Load the cleaned dataset: - Convert categorical variable to numerical variable.
- Data slicing:
  - Dataset is split into 80 percent of training data, 20 % of test set.
- TrainingParameters :
  - train() method is passed with repeated cross-validation resampling method for 10 number of resampling iterations repeated for 3 times.

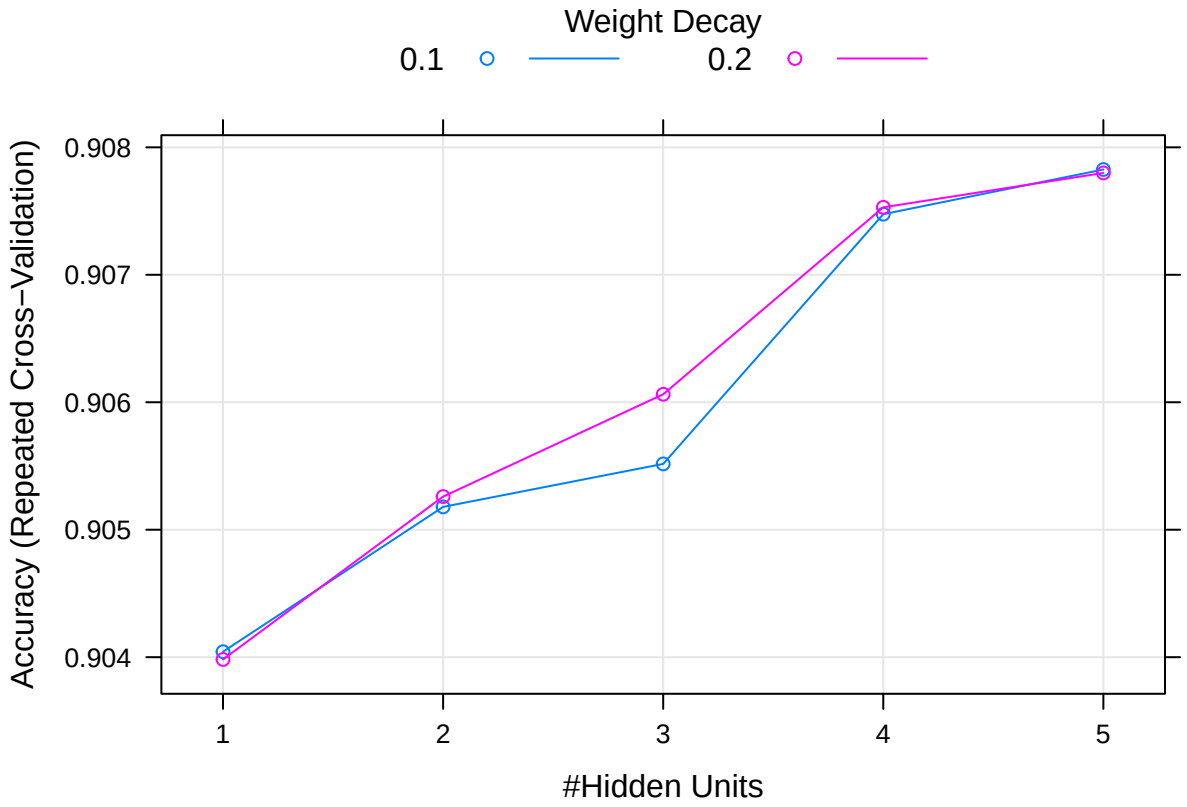
Machine Learning: Classification using Neural Networks

- Model Training
  - We can us neuralnet() to train a NN model. Also, the train() function from caret can help us tune parameters. We can plot the result to see which set of parameters is fit our data the best.
  - nnnet package by defualt uses the Logisitc Activation function.
  - Data Pre-Processing With Caret: The scale transform calculates the standard deviation for an attribute and divides each value by that standard deviation.
  - The center transform calculates the mean for an attribute and subtracts it from each value.
  - Combining the scale and center transforms will standardize your data.
  - Attributes will have a mean value of 0 and a standard deviation of 1.
  - Training transforms can prepared and applied automatically during model evaluation.
  - Transforms applied during training are prepared using the preProcess() and passed to the train() function via the preProcess argument.
  - Backpropagation algorithm is a supervised learning method for multilayer feed-forward networks from the field of Artificial Neural Networks.
  - The principle of the backpropagation approach is to model a given function by modifying internal weightings of input signals to produce an expected output signal. The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to modify its internal state.
  - We use Backpropagation as algorithm in neural network package.

```
nnetGrid <-  expand.grid(size = seq(from = 1, to = 5, by = 1)
                         ,decay = seq(from = 0.1, to = 0.2, by = 0.1)
                         )
nn_model <- train(Term_Deposit ~ ., subTrain,
                  method = "nnet",  algorithm = 'backprop',
                  trControl= TrainingParameters,
                  preProcess=c("scale","center"),
                  na.action = na.omit,
                  #metric = "ROC",
                  tuneGrid = nnetGrid,
                  trace=FALSE,
                  verbose=FALSE)
```

- Based on the caret neural network model, train sets hidden layer.caret neural network picks the best neural network based on size, decay.We can visualize accuracy for different hidden layers below:

```
##    size decay  Accuracy     Kappa  AccuracySD    KappaSD
## 1     1   0.1 0.9040427 0.4358269 0.002567681 0.01507662
## 2     1   0.2 0.9039820 0.4367773 0.002584615 0.01548641
## 3     2   0.1 0.9051791 0.4418804 0.002210548 0.02086579
## 4     2   0.2 0.9052600 0.4422005 0.002728089 0.01602451
## 5     3   0.1 0.9055163 0.4370649 0.003408263 0.04122454
## 6     3   0.2 0.9060626 0.4388124 0.003642564 0.04049697
## 7     4   0.1 0.9074754 0.4514426 0.003252861 0.02746812
## 8     4   0.2 0.9075294 0.4470578 0.002480409 0.03099499
## 9     5   0.1 0.9078261 0.4603130 0.002673938 0.02354327
## 10    5   0.2 0.9077991 0.4425523 0.004547698 0.07854141
```



- Prediction
  - Now, our model is trained with accuracy = 0.8889 We are ready to predict classes for our test set.

```
##
## prediction  no yes
##        no  284  25
##        yes   8  16
```

- Confusion matrix & Accuracy of Neural Network model:

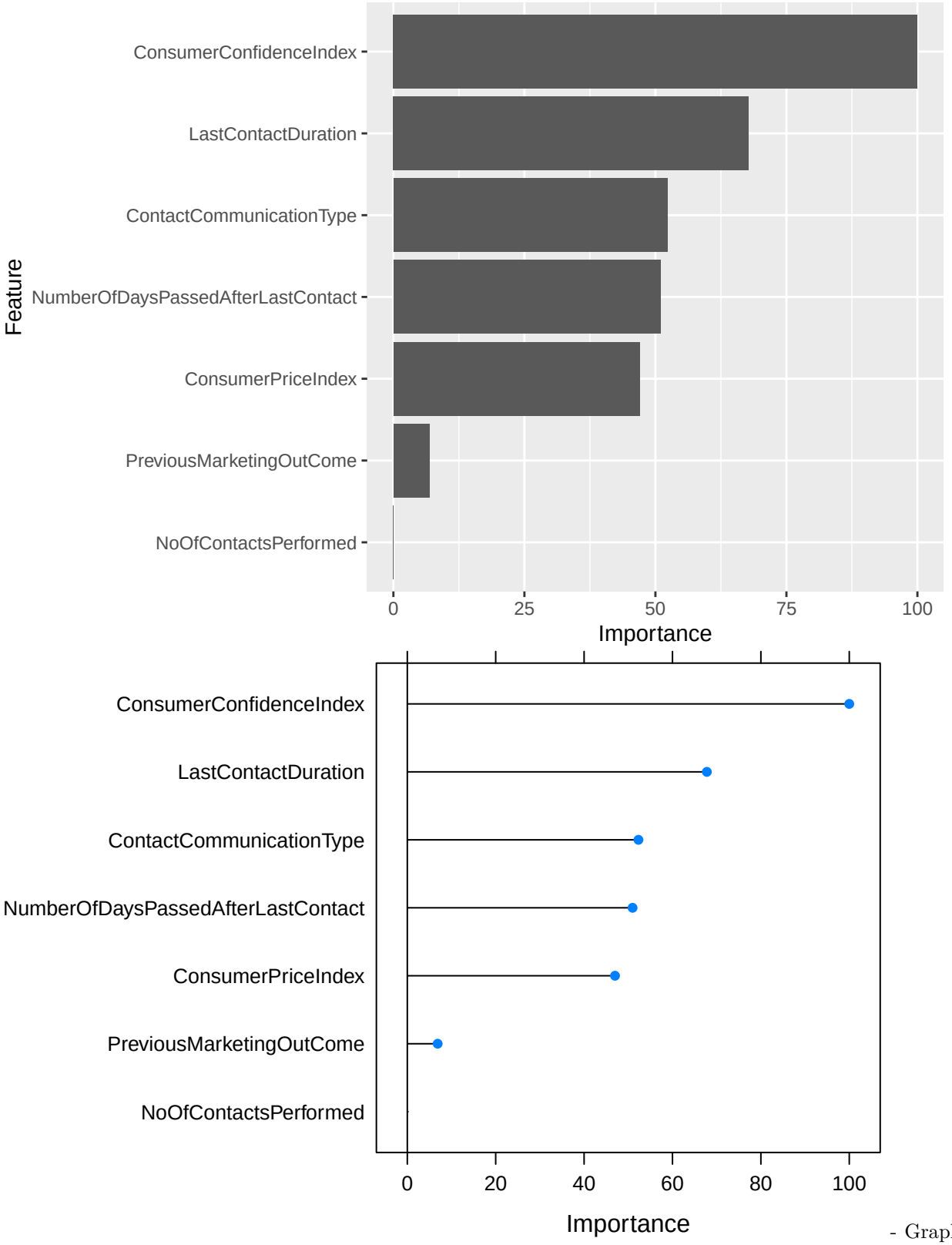
```
## [1] 0.9009009
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
```

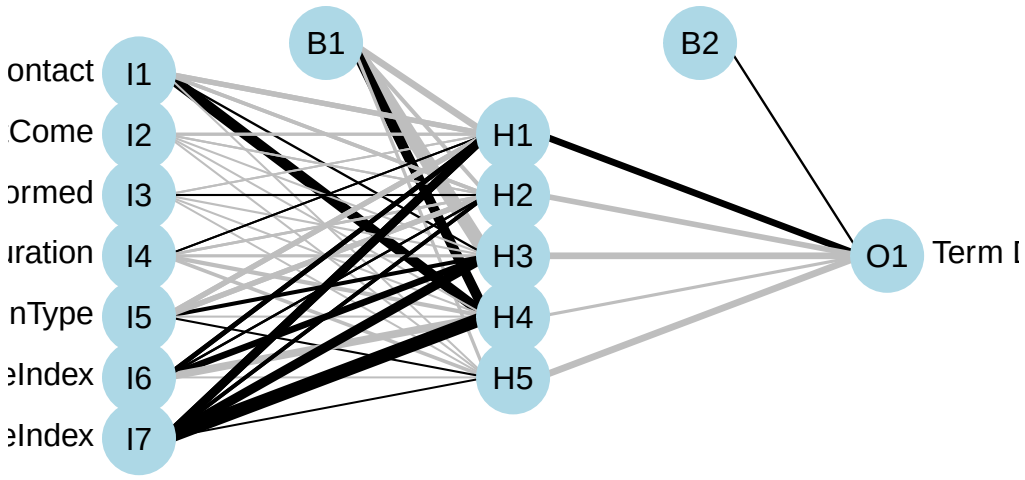
```
##        no  284  25
##        yes   8  16
##
##                Accuracy : 0.9009
##                  95% CI : (0.8636, 0.9308)
##     No Information Rate : 0.8769
##     P-Value [Acc > NIR] : 0.103011
##
##                   Kappa : 0.4415
##
##  Mcnemar's Test P-Value : 0.005349
##
##             Sensitivity : 0.9726
##             Specificity : 0.3902
##          Pos Pred Value : 0.9191
##          Neg Pred Value : 0.6667
##              Prevalence : 0.8769
##          Detection Rate : 0.8529
##    Detection Prevalence : 0.9279
##       Balanced Accuracy : 0.6814
##
##        'Positive' Class : no
##
```

- Confusion matrix & Accuracy of Neural Network model:
  - Plotting nnet variable importance

```
## nnet variable importance
##
##                                    Overall
## ConsumerConfidenceIndex            100.000
## LastContactDuration                 67.780
## ContactCommunicationType            52.298
## NumberOfDaysPassedAfterLastContact  50.969
## ConsumerPriceIndex                  46.995
## PreviousMarketingOutCome             6.867
## NoOfContactsPerformed                0.000
```



- Graphical Representation of our Neural Network



Machine Learning: Classification using SVM

- SVM is another classification method that can be used to predict if a client falls into either 'yes' or 'no' class.
- The linear, polynomial and RBF or Gaussian kernel in SVM are simply different in case of making the hyperplane decision boundary between the classes.
- The kernel functions are used to map the original dataset (linear/nonlinear) into a higher dimensional space with view to making it linear dataset.
- Usually linear and polynomial kernels are less time consuming and provides less accuracy than the rbf or Gaussian kernels.
- The k cross validation is used to divide the training set into k distinct subsets. Then every subset is used for training and others k-1 are used for validation in the entire trainging phase. This is done for the better training of the classification task.Overall, if you are unsure which kernel method would be best, a good practice is use of something like 10-fold cross-validation for each training set and then pick the best algorithm.

SVM Classifier using Linear Kernel

- Caret package provides train() method for training our data for various algorithms. We just need to pass different parameter values for different algorithms. Before train() method, we will first use trainControl() method.
- We are setting 3 parameters of trainControl() method. The "method" parameter holds the details about resampling method. We can set "method" with many values like "boot", "boot632", "cv", "repeatedcv", "LOOCV", "LGOCV" etc. For this project, let's try to use repeatedcv i.e, repeated cross-validation.
- The "number" parameter holds the number of resampling iterations. The "repeats" parameter contains the complete sets of folds to compute for our repeated cross-validation. We are using setting number =10 and repeats =3. This trainControl() methods returns a list. We are going to pass this on our train() method.
- Before training our SVM classifier, set.seed().
- For training SVM classifier, train() method should be passed with "method" parameter as "svmLinear". We are passing our target variable Term_Deposit. The "Term_Deposit.~." denotes a formula for using all attributes in our classifier and Term_Deposit. as the target variable. The "trControl" parameter

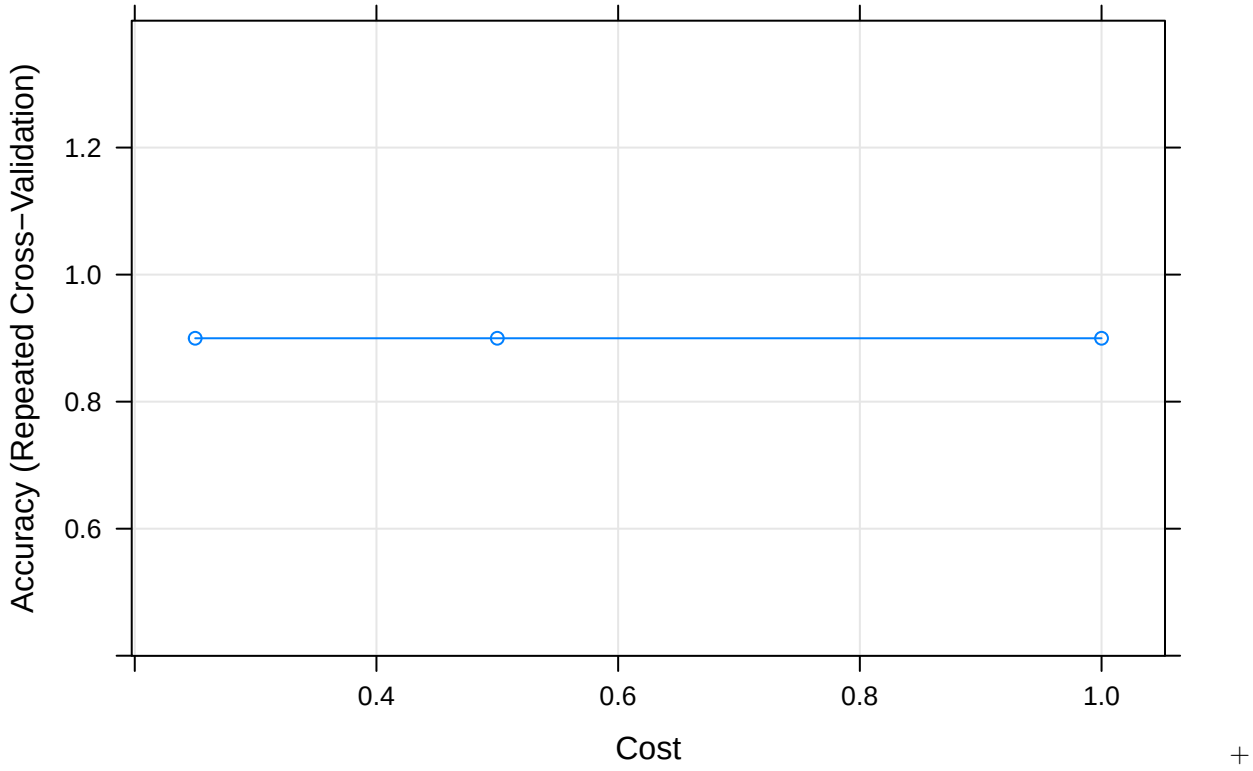
should be passed with results from our trianControl() method. The "preProcess" parameter is for preprocessing our training data.

- As discussed earlier for our data, preprocessing is a mandatory task. We are passing 2 values in our "preProcess" parameter "center" & "scale". These two help for centering and scaling the data. After preProcessing these convert our training data with mean value as approximately "0" and standard deviation as "1". The "tuneLength" parameter holds an integer value. This is for tuning our algorithm.

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 2)
set.seed(323)
grid <- expand.grid(C = c( 0.25, 0.5, 1))
svm_Linear_Grid <- train(Term_Deposit ~ ., data = subTrainSVM, method = "svmLinear", trControl=trctrl, p
                         tuneGrid = grid,
                         tuneLength = 10)
svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 9888 samples
##    7 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold, repeated 2 times)
## Summary of sample sizes: 8898, 8900, 8899, 8900, 8899, 8899, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8997786  0.2837249
##   0.50  0.8997786  0.2837249
##   1.00  0.8997786  0.2837249
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.25.
```

- The above model is showing that our classifier is giving best accuracy on C = 0.25 Let's try to make predictions using this model for our test set and check its accuracy.



Accuracy on the test set by train control is 89% using C=0.25.

```
## [1] 0.9166667
```

- Final prediction accuracy on the test set is 0.9166667.

SVM Classifier using Non-Linear Kernel

- Now, we will try to build a model using Non-Linear Kernel like Radial Basis Function. For using RBF kernel, we just need to change our train() method's "method" parameter to "svmRadial". In Radial kernel, it needs to select proper value of Cost "C" parameter and "sigma" parameter.

```
set.seed(323)
grid_radial <- expand.grid(sigma = c(0.25, 0.5,0.9),
 C = c(0.25, 0.5,1))
svm_Radial <- train(Term_Deposit ~ ., data = subTrainSVM, method = "svmRadial",
trControl=trctrl,
preProcess = c("center", "scale"),tuneGrid = grid_radial,
tuneLength = 10)

svm_Radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 9888 samples
##    7 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (10 fold, repeated 2 times)
## Summary of sample sizes: 8898, 8900, 8899, 8900, 8899, 8899, ...
## Resampling results across tuning parameters:
```

```
##
##   sigma  C     Accuracy   Kappa
##   0.25   0.25  0.9131796  0.4235344
##   0.25   0.50  0.9128756  0.4232314
##   0.25   1.00  0.9153028  0.4430922
##   0.50   0.25  0.9163140  0.4454370
##   0.50   0.50  0.9198031  0.4851975
##   0.50   1.00  0.9230389  0.5121697
##   0.90   0.25  0.9220279  0.4926883
##   0.90   0.50  0.9236464  0.5192152
##   0.90   1.00  0.9271349  0.5520005
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.9 and C = 1.
```

- SVM-RBF kernel calculates variations and will present us best values of sigma & C. Based on the output best values of sigma= 0.9 & C=1 Let's check our trained models' accuracy on the test set.

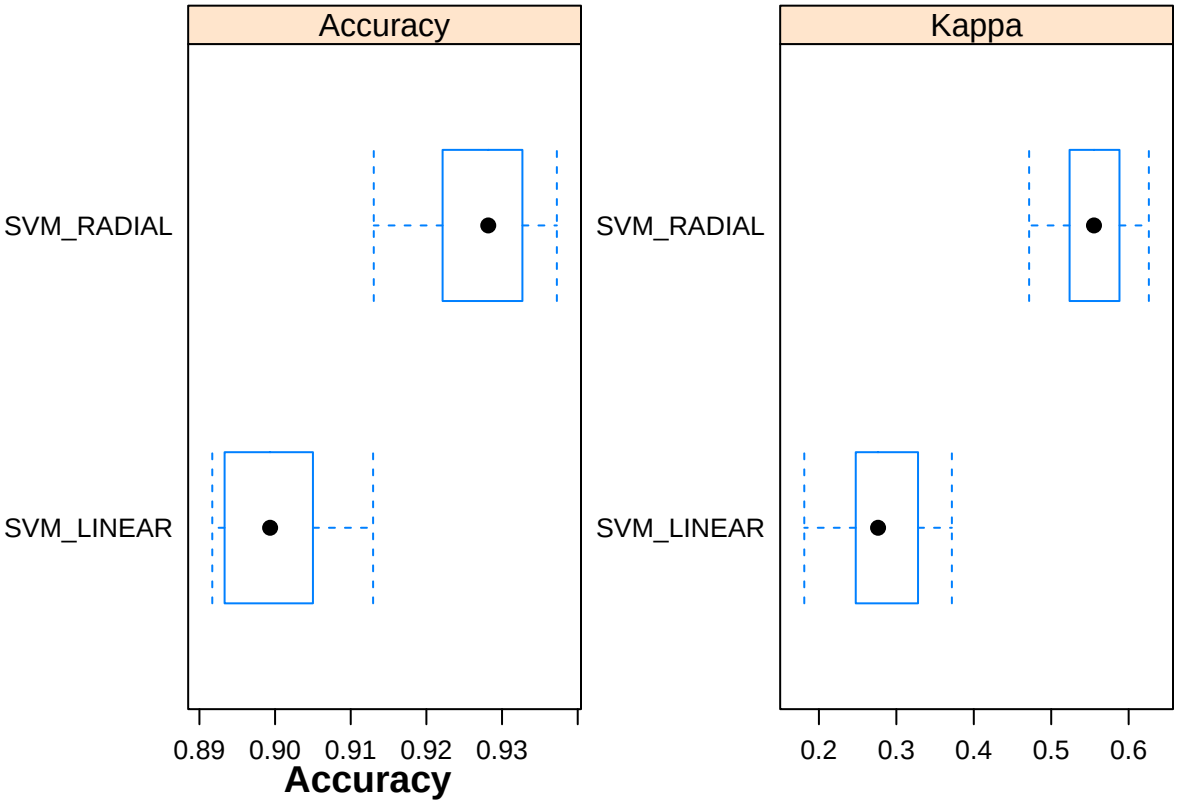
```
## [1] 0.8333333
```

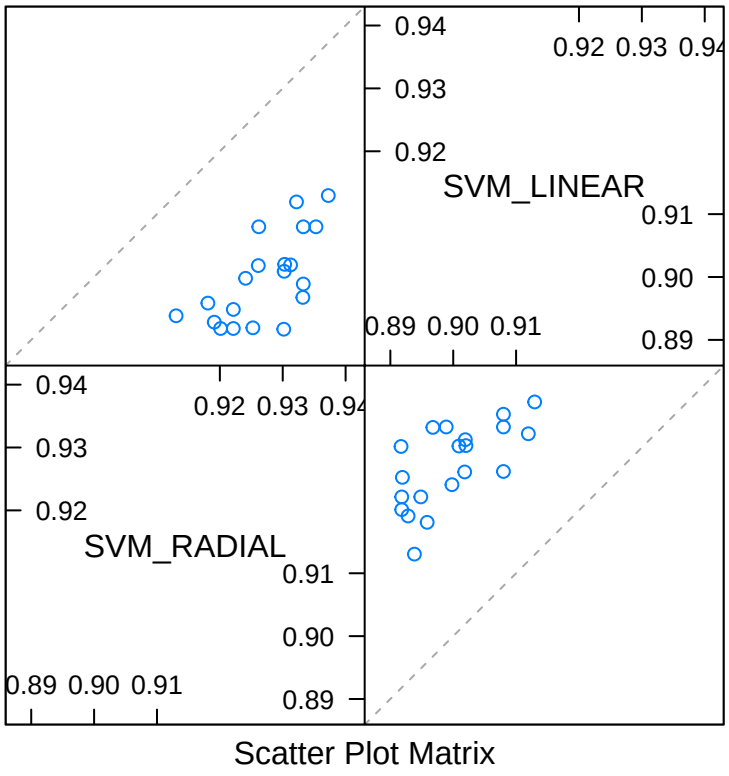
- Final prediction accuracy on the test set is 0.8333333

Comparision between SVM models

- Comparision between SVM Linear and Radial Models.

```
##
## Call:
## summary.resamples(object = algo_results)
##
## Models: SVM_RADIAL, SVM_LINEAR
## Number of resamples: 20
##
## Accuracy
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## SVM_RADIAL 0.9130435 0.9221436 0.9281750 0.9271349 0.9324393 0.9372470    0
## SVM_LINEAR 0.8917004 0.8935794 0.8993427 0.8997786 0.9035121 0.9129555    0
##
## Kappa
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## SVM_RADIAL 0.4714582 0.5264890 0.5552111 0.5520005 0.5850553 0.6261048    0
## SVM_LINEAR 0.1811203 0.2504363 0.2763777 0.2837249 0.3243298 0.3717218    0
```





Scatter Plot Matrix

Conclusion

From the above implementation, the results are impressive and convincing in terms of using a machine learning algorithm to decide on the marketing campaign of the bank. Majority of the attributes in the dataset contribute significantly to the building of a predictive model. All the three ML approach acheives good accuracy rate(>85%) and are easier to implement.