

Data-Driven Approach To Predict Success Of Bank Telemarketing

Nisha Selvarajan

10/24/2020

- Objectives
- Data Description
- Data Analysis
- Variable Importance & Crossplot to Deposit
- Prepare Data for Classification
- Machine Learning: Classification using Neural Networks
- Machine Learning: Classification using SVM
- SVM Classifier using Linear Kernel
- SVM Classifier using Non-Linear Kernel
- Comparison between SVM models
- Conclusion

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

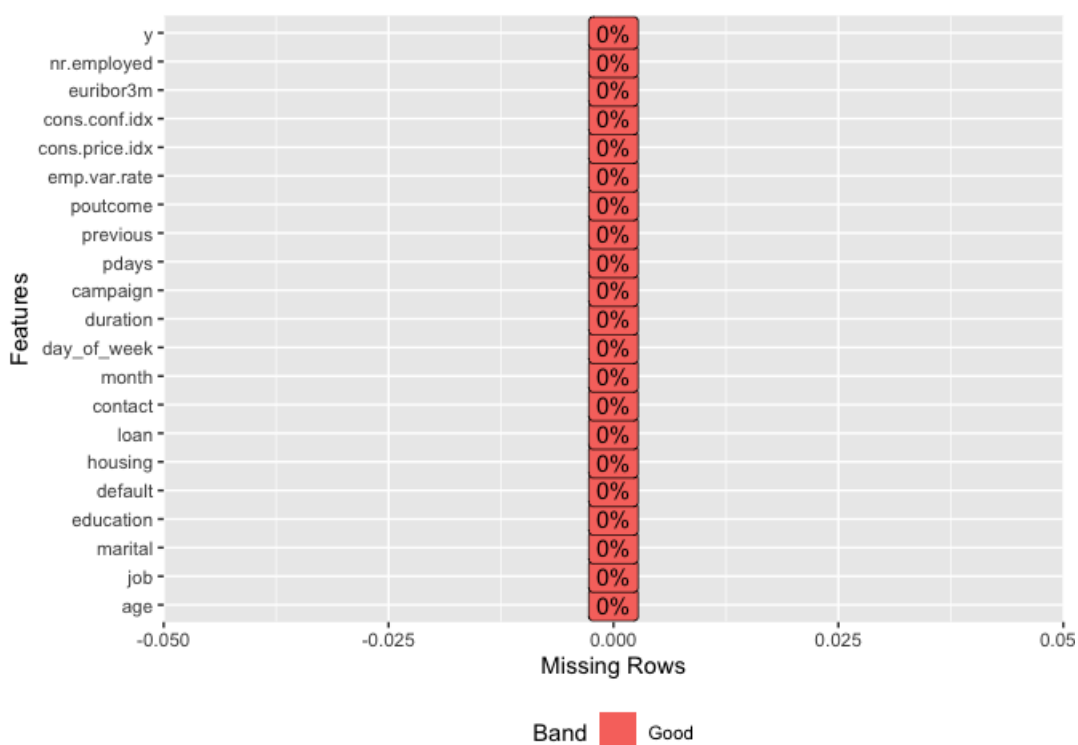
- Data set which is utilized for this research has been taken from University of California, Irvine machine learning repository (<https://archive.ics.uci.edu/ml/datasets/Bank+Marketing?package=regsel&version=0.2>) which is openly available for the public for research purpose. The dataset contains 41188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

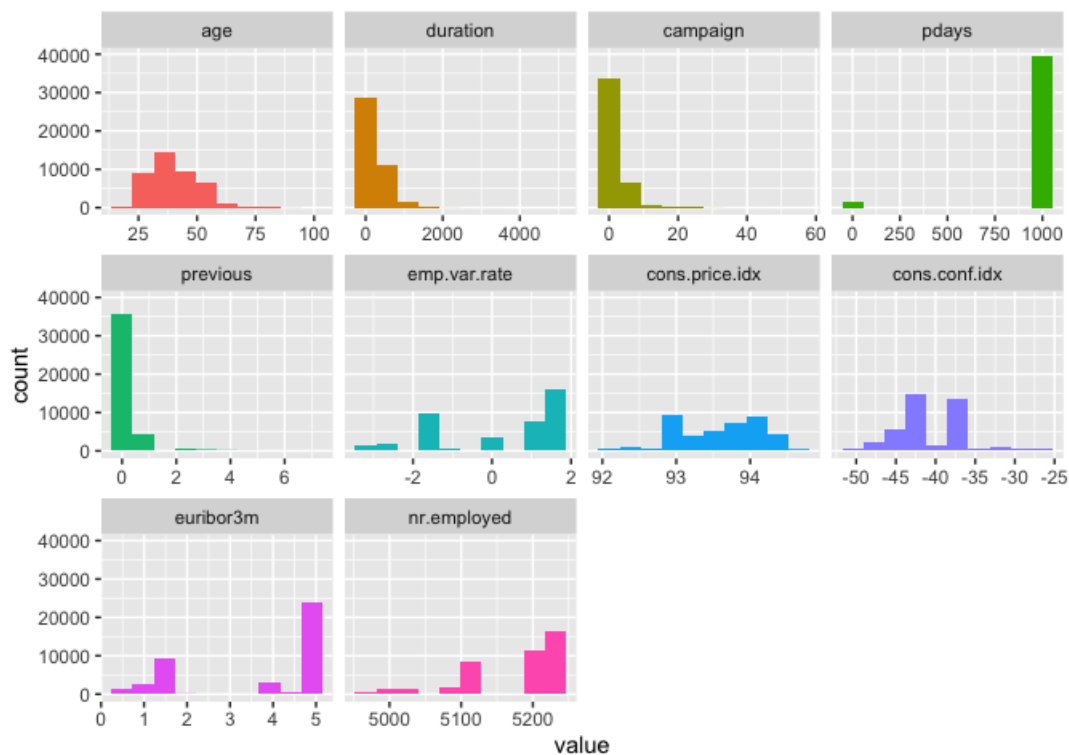
Names	Description
pdays	Numerical - No of days passed after previous campaign contact.
previous	Numerical - No of contacts performed before this campaign for this client
poutcome	Categorical - Outcome of the previous marketing campaign
emp.var.rate	Numerical - Employment Variation Rate - quarterly indicator
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.

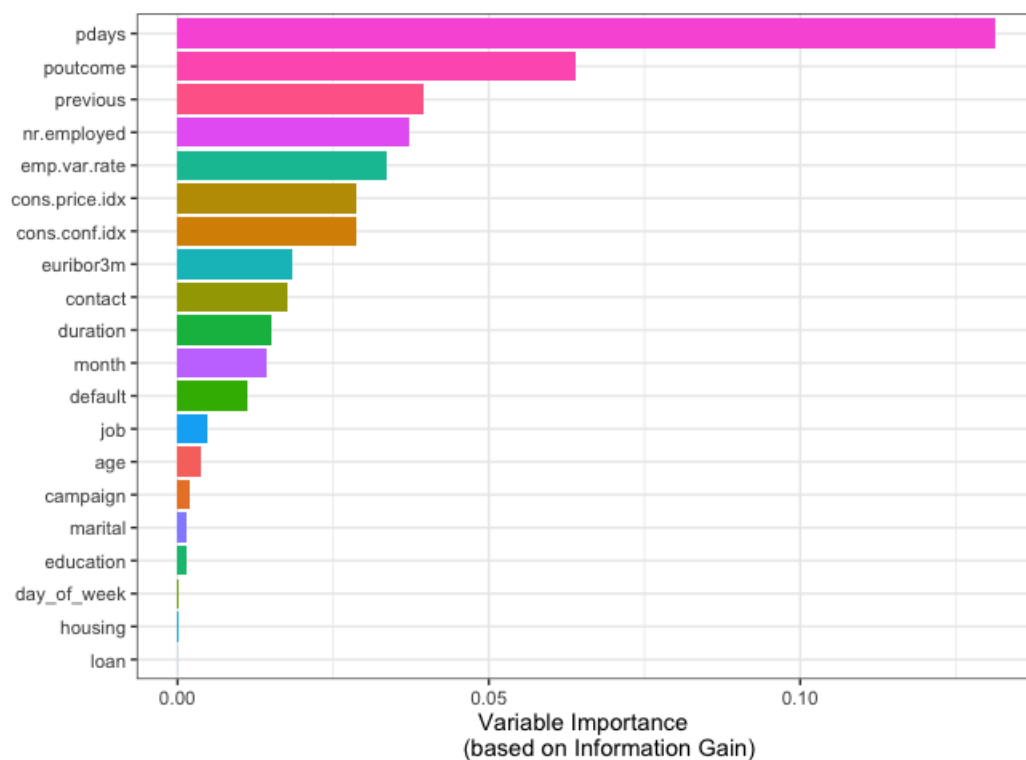


- Plotting histograms for numerical variables.

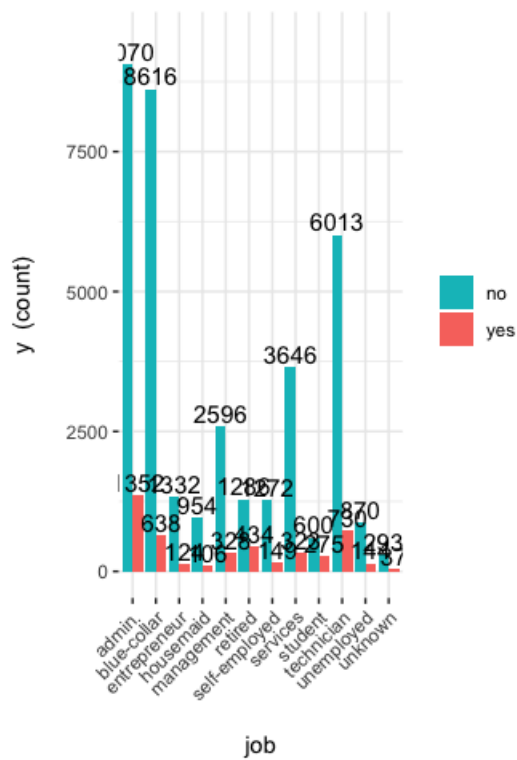
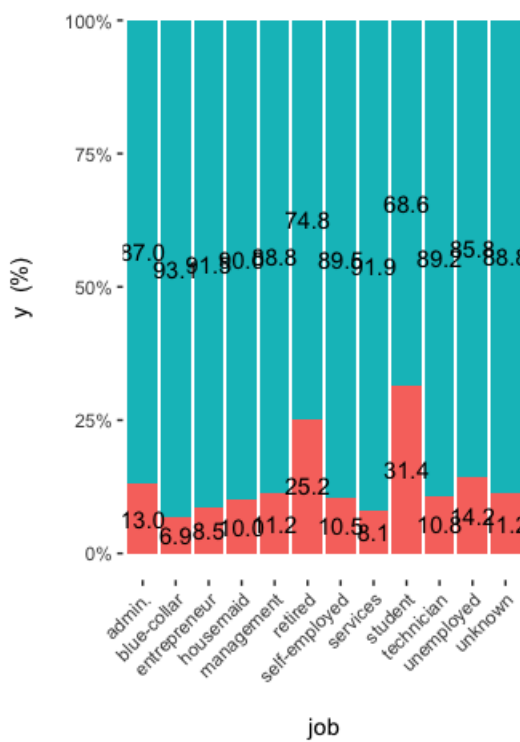
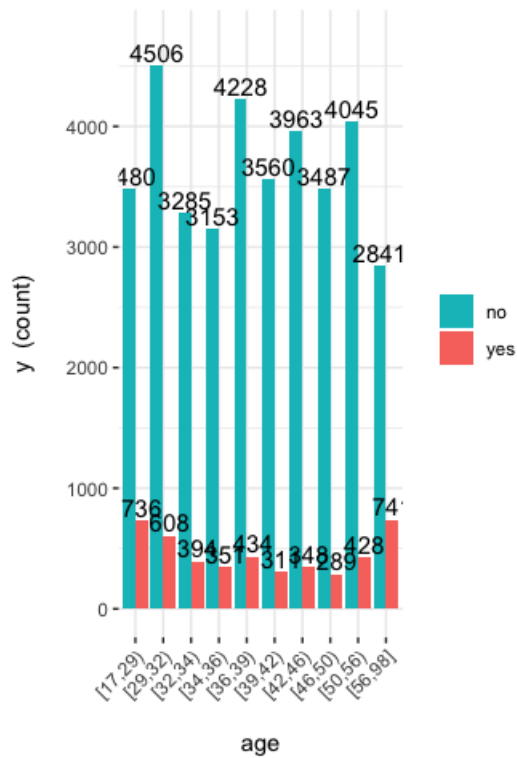
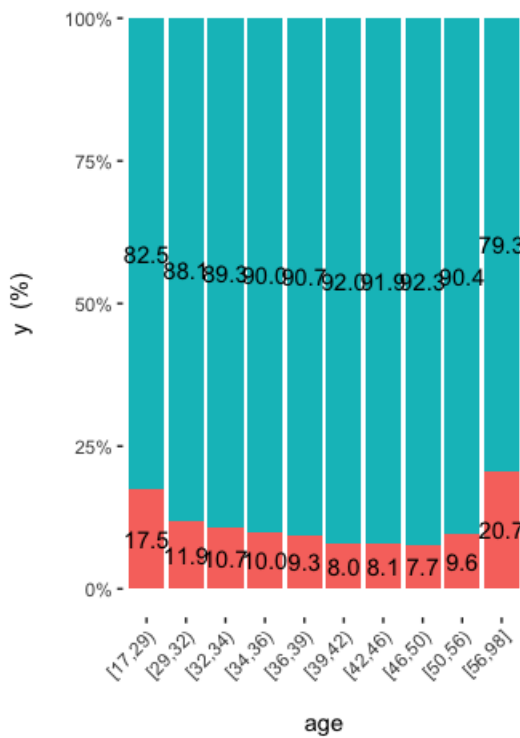


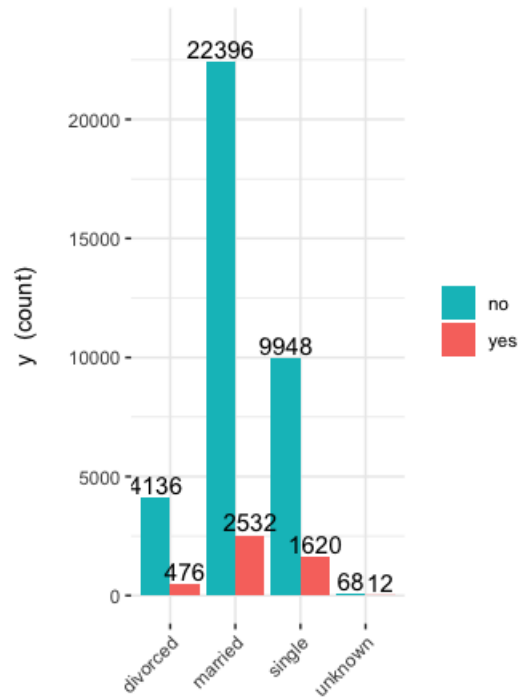
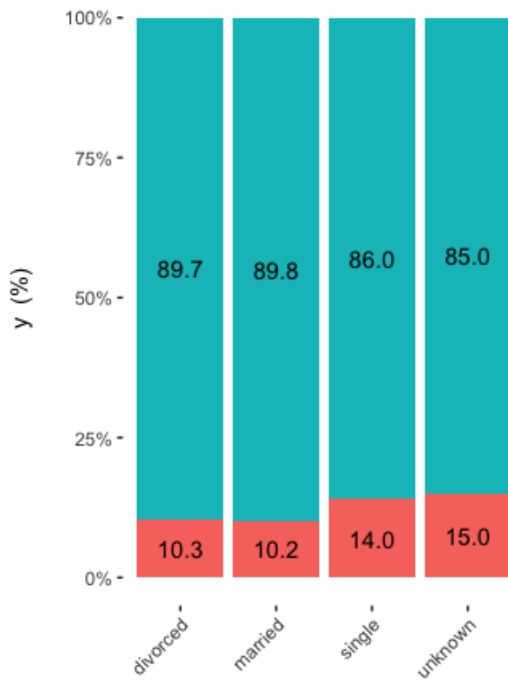
Variable Importance & Crossplot to Deposit

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



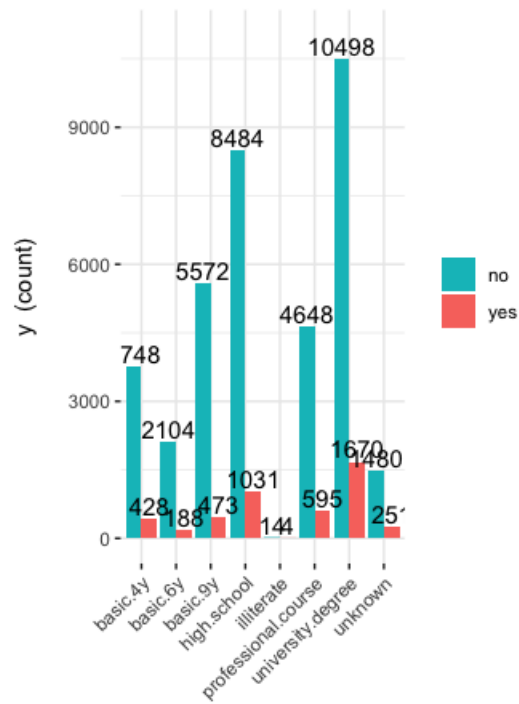
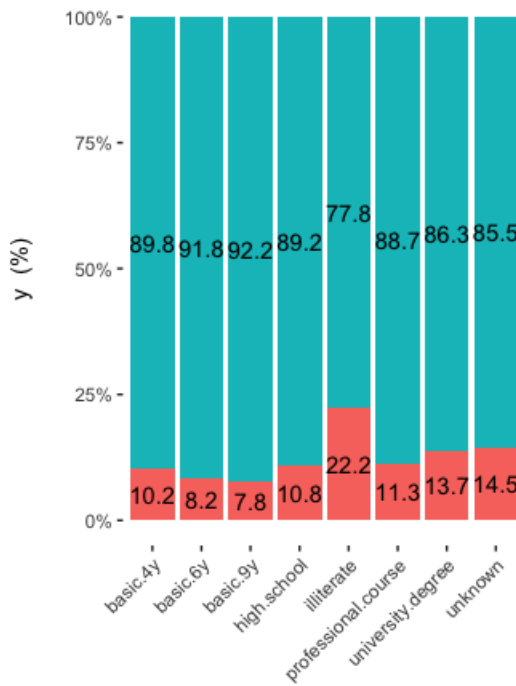
- Bivariate analysis cross plot showing relationship of each and every variable with respect to target variable





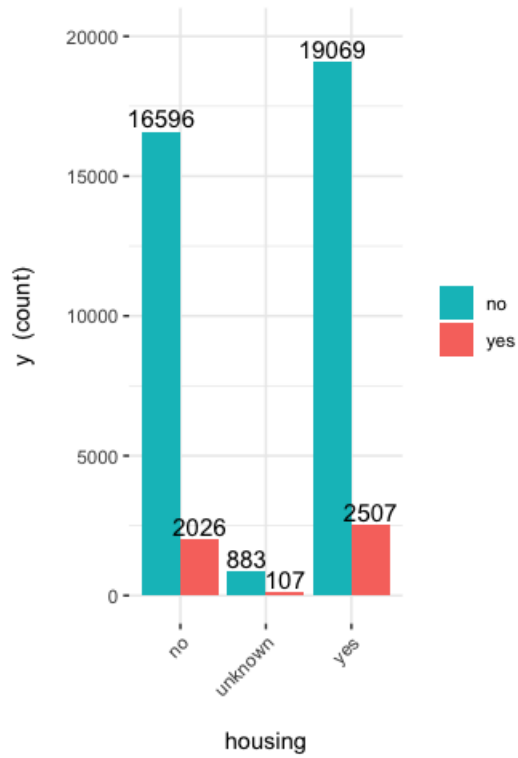
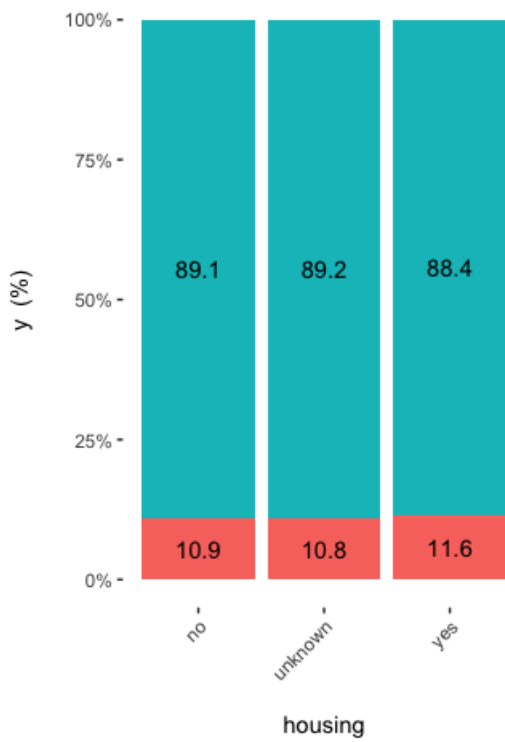
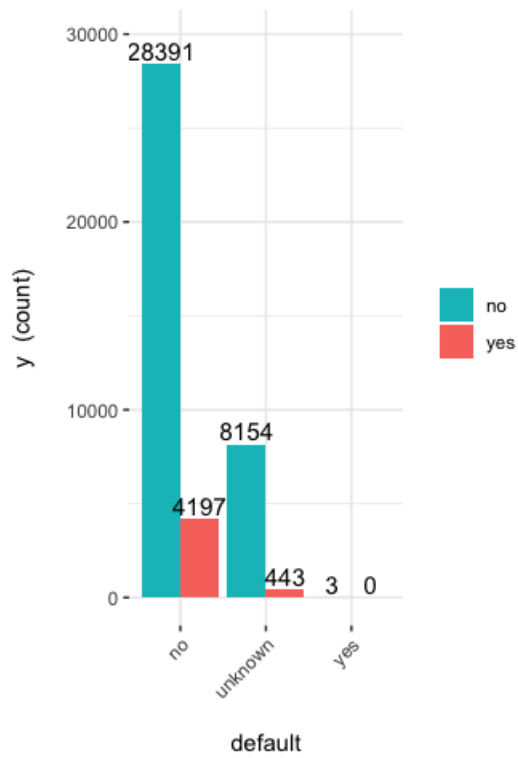
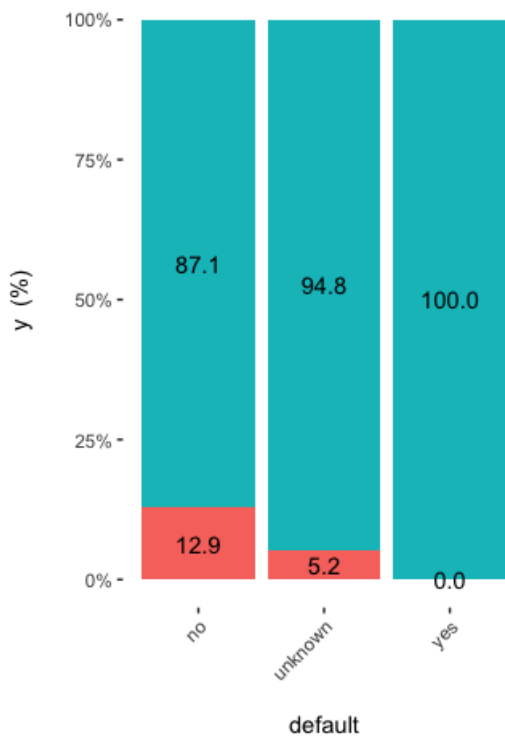
marital

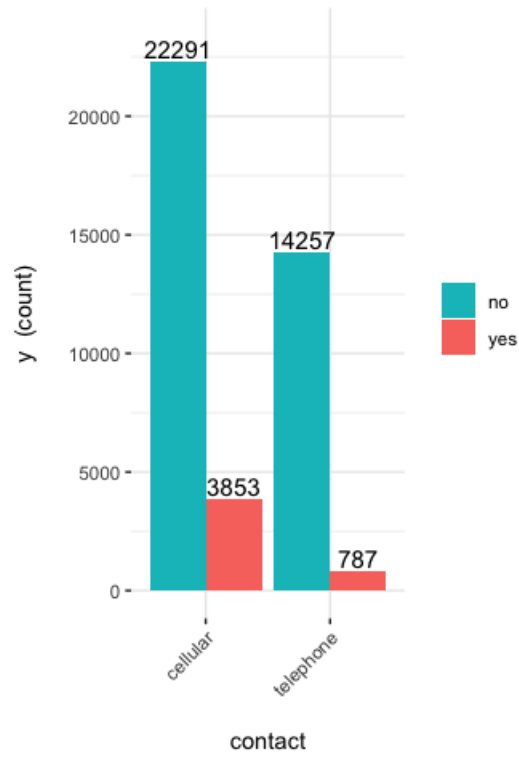
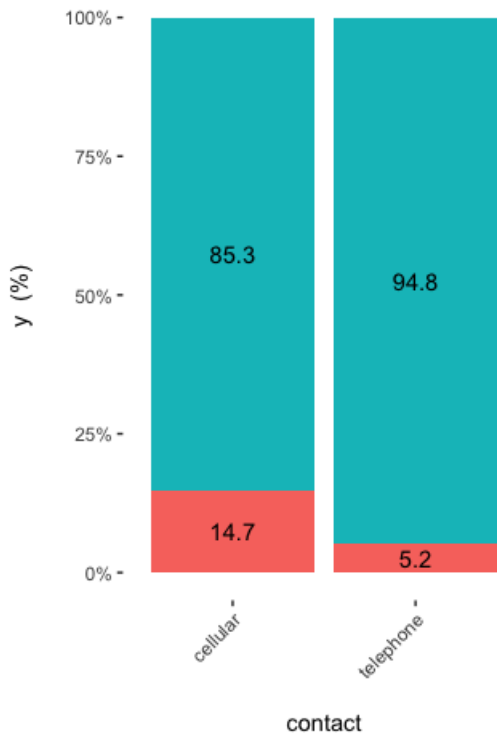
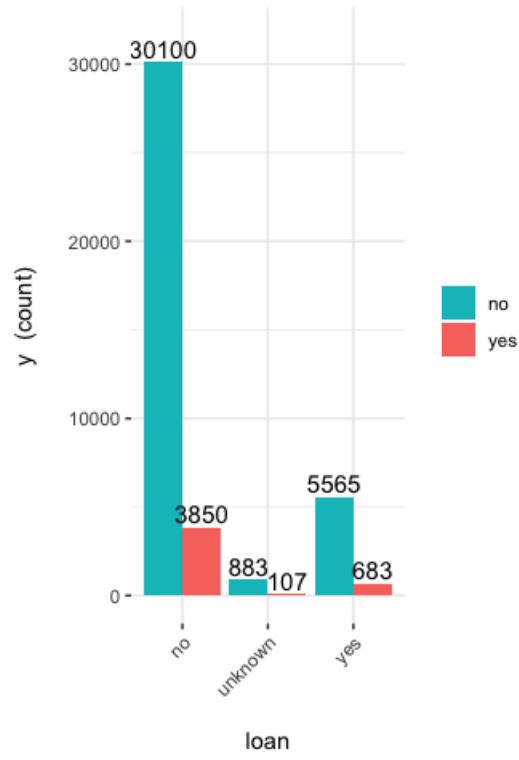
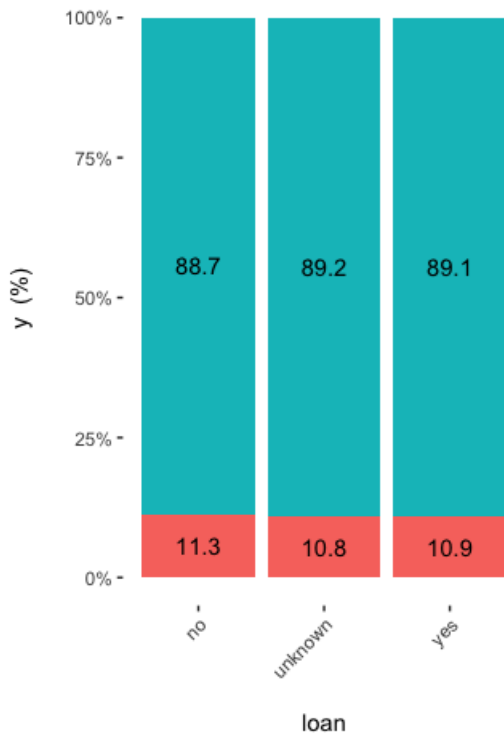
marital

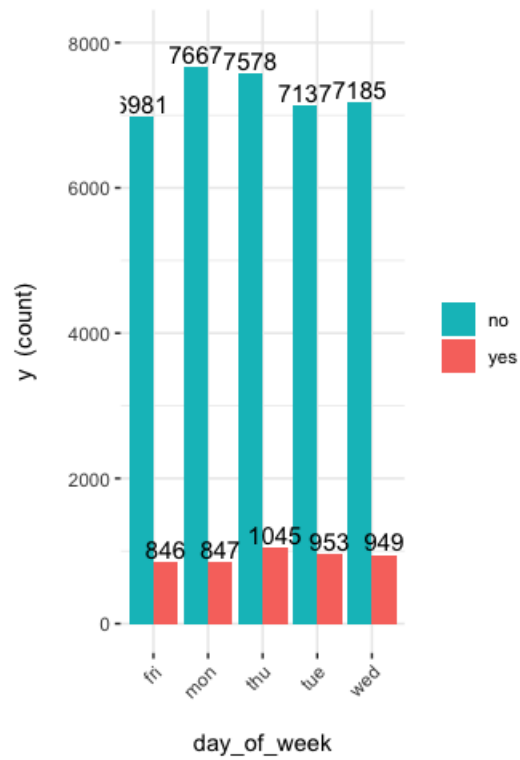
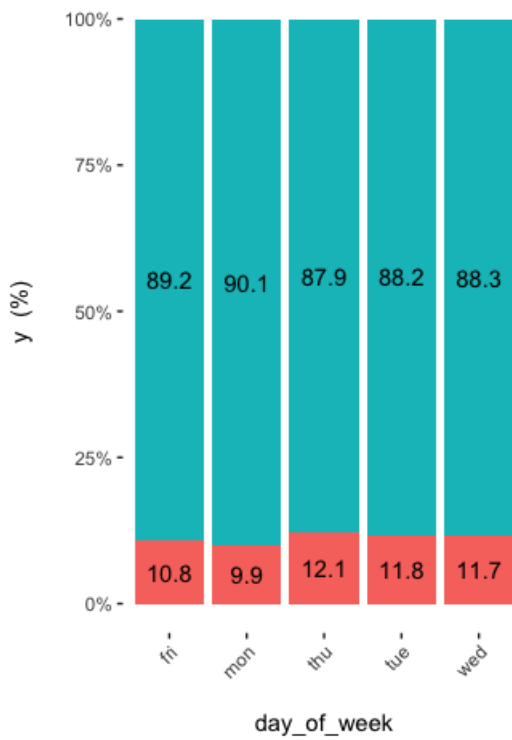
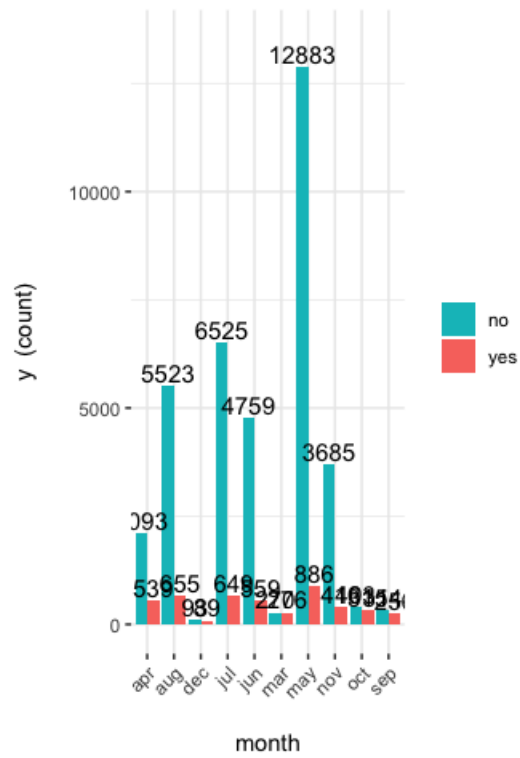
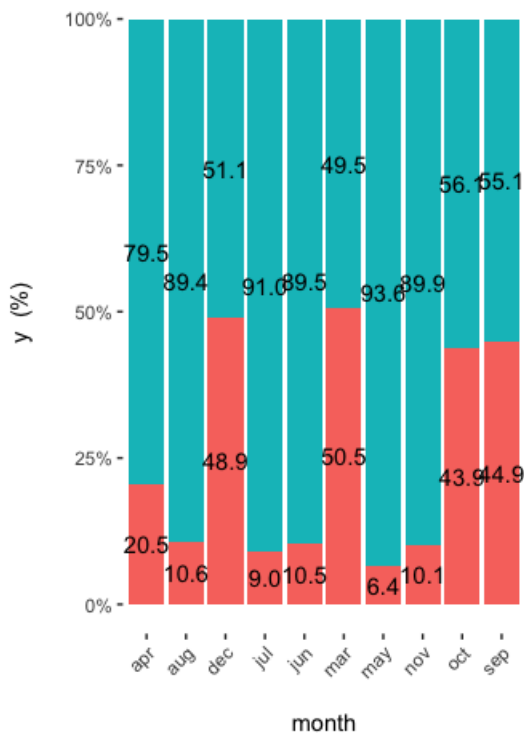


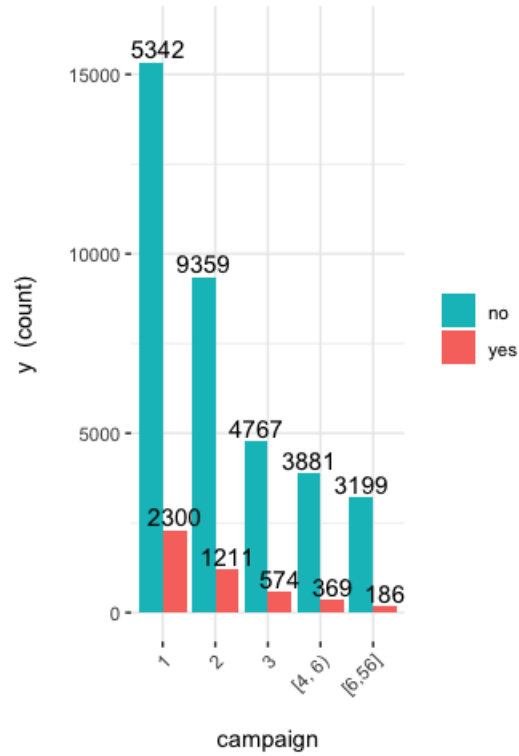
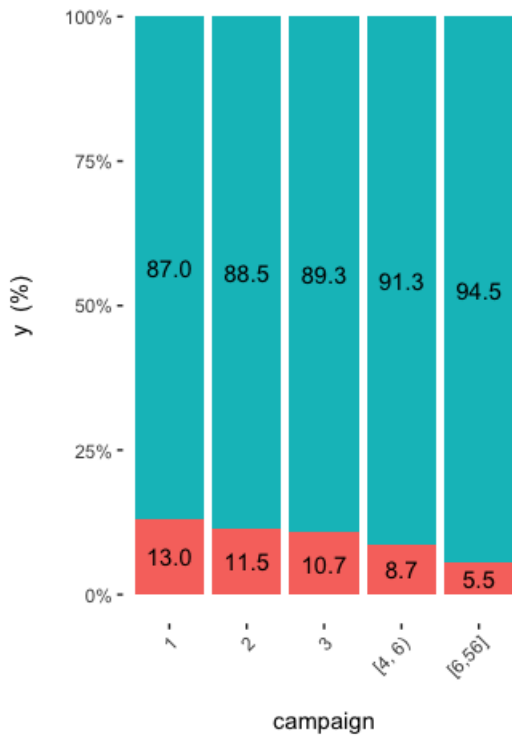
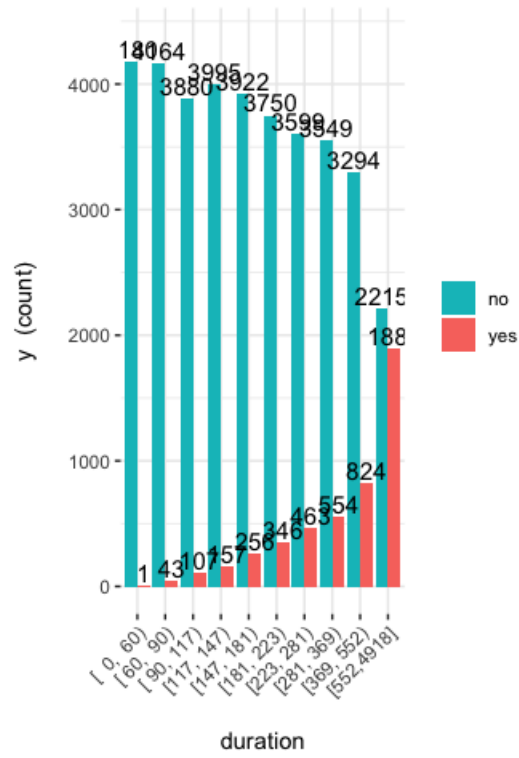
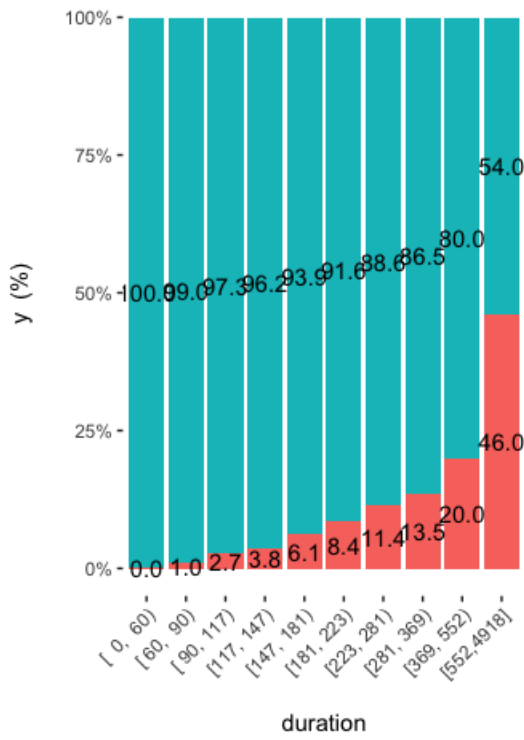
education

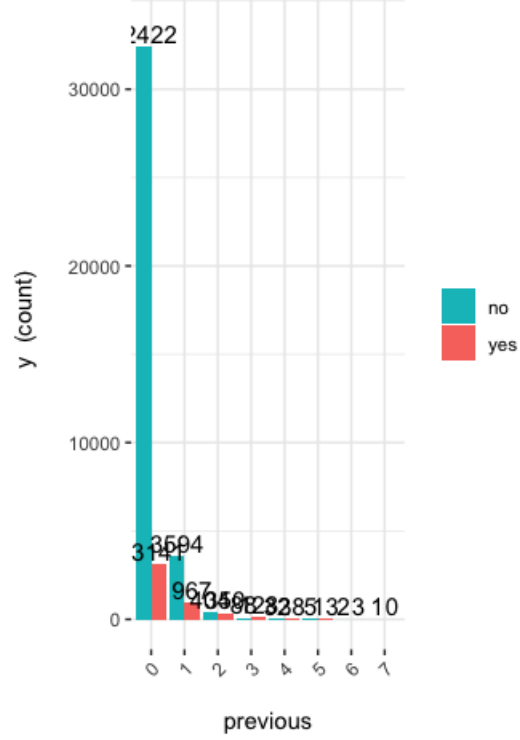
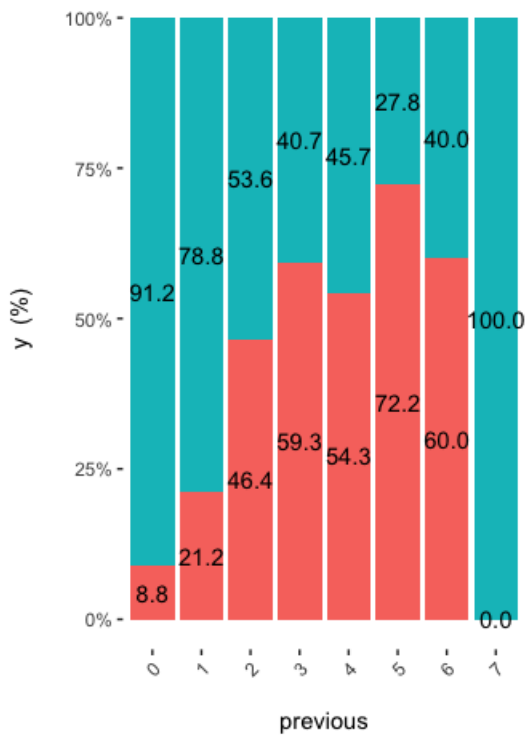
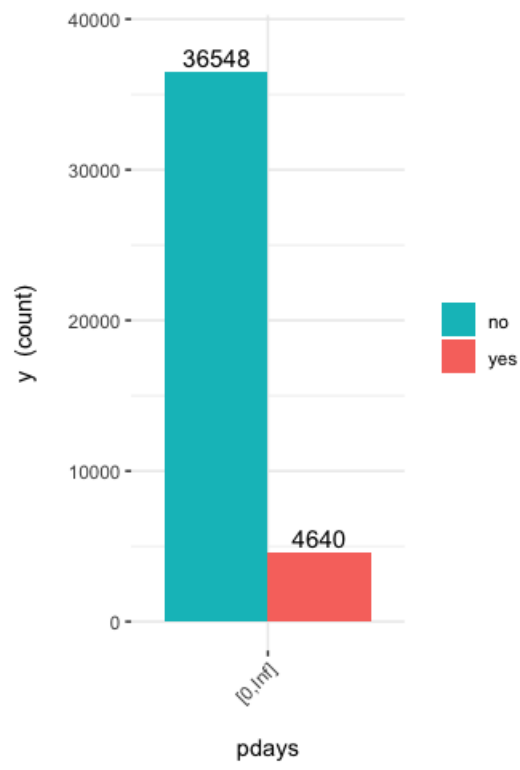
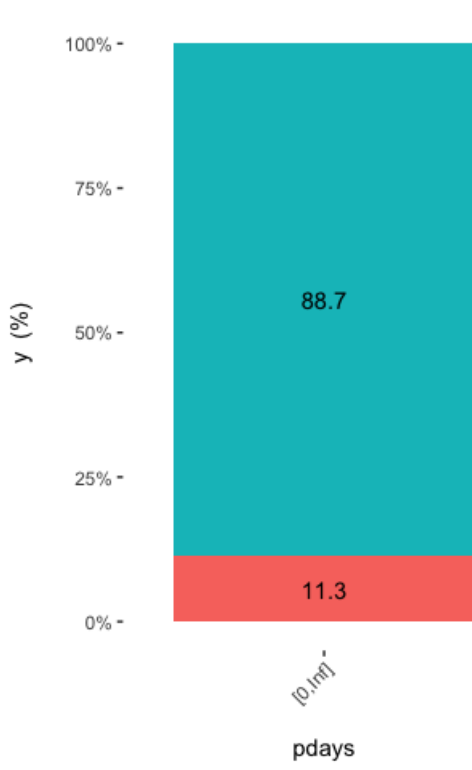
education

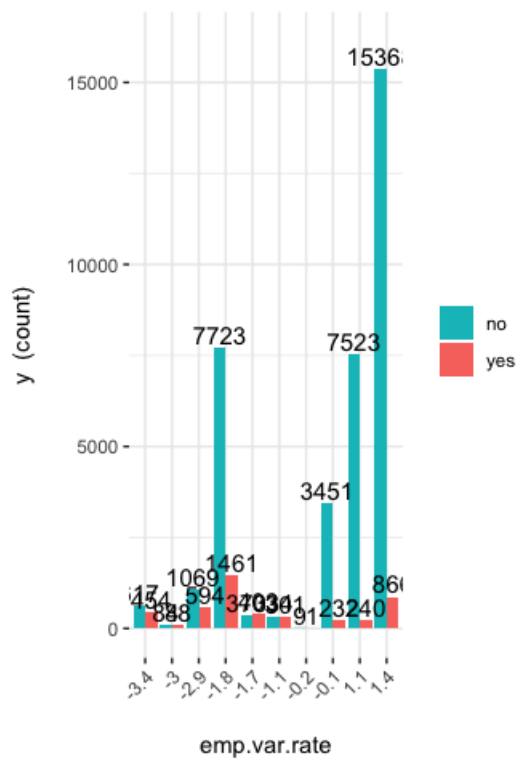
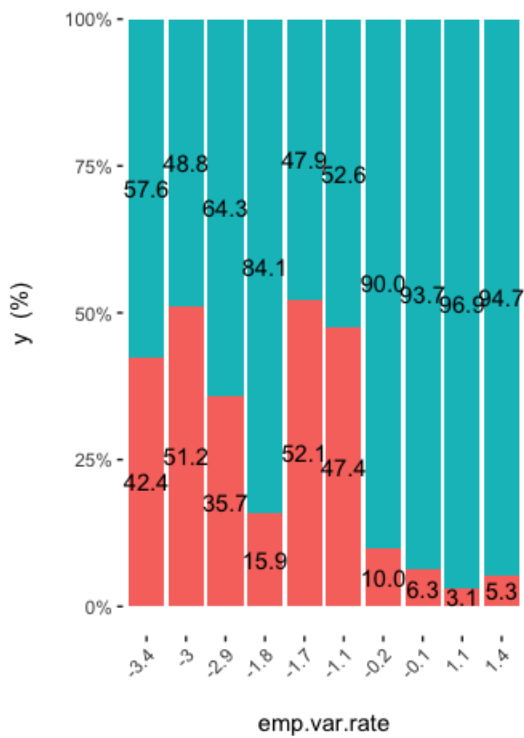
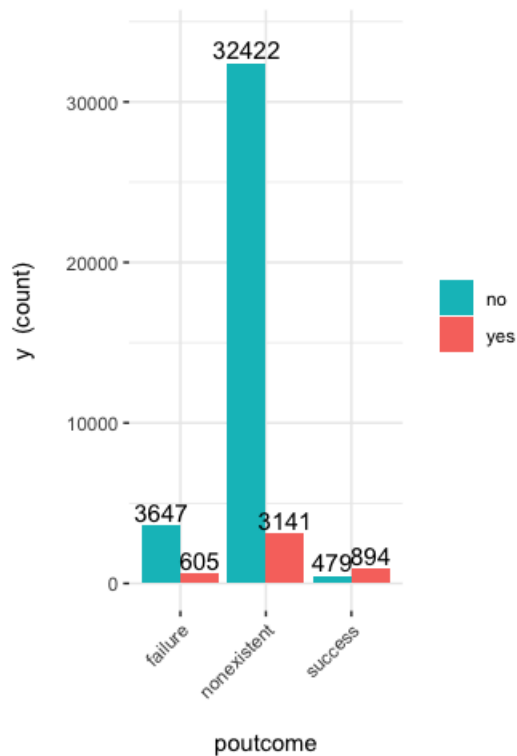
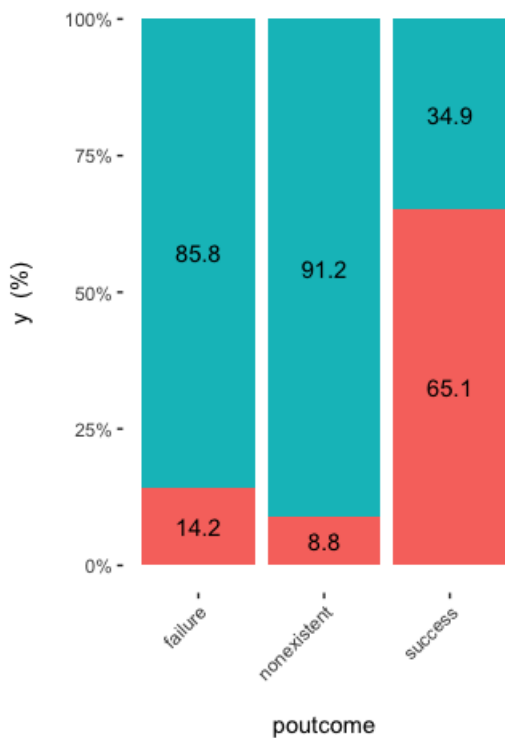


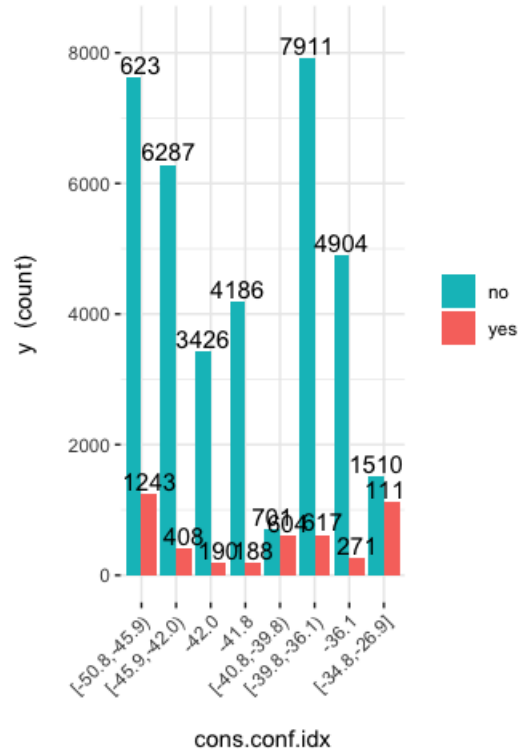
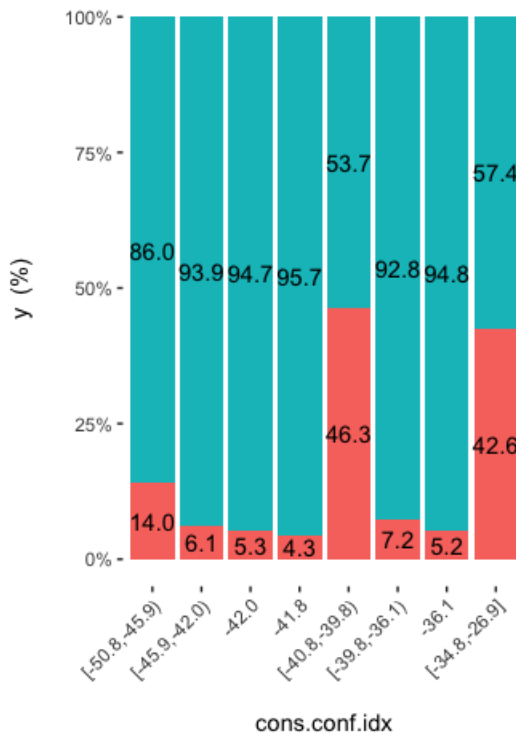
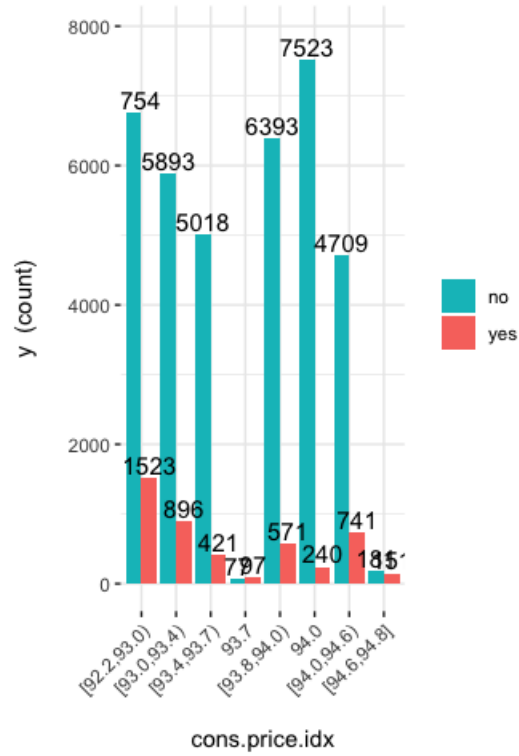
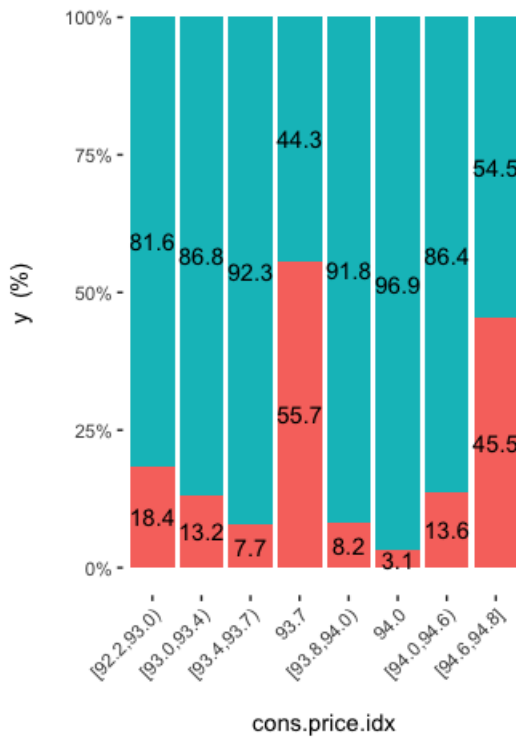


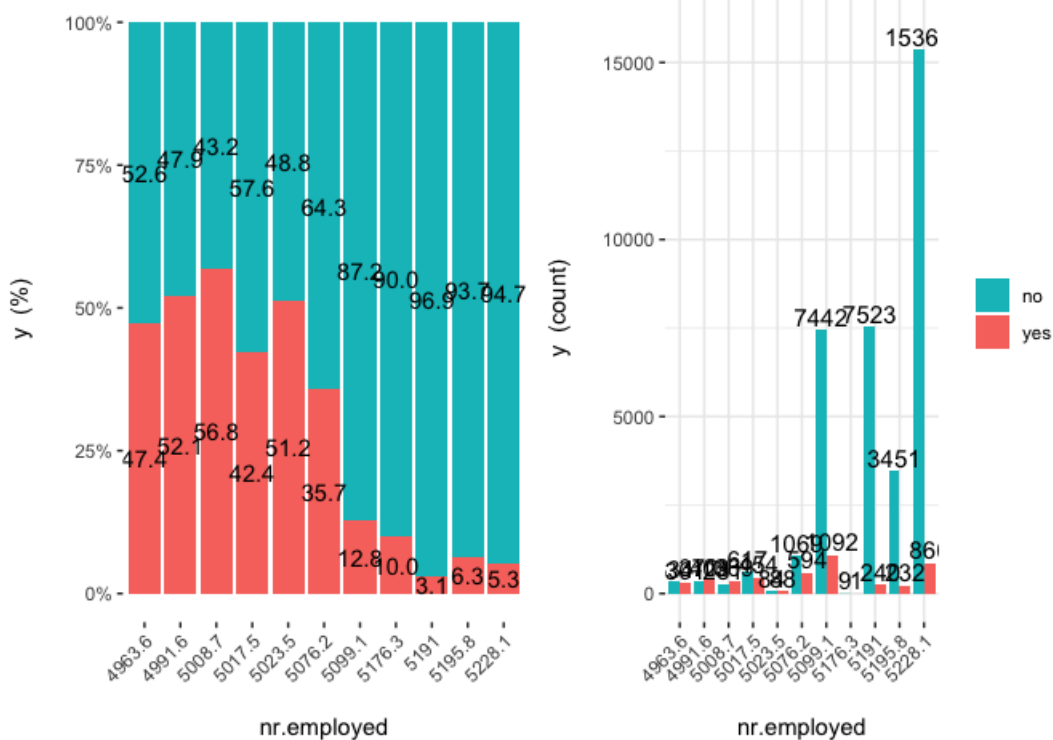
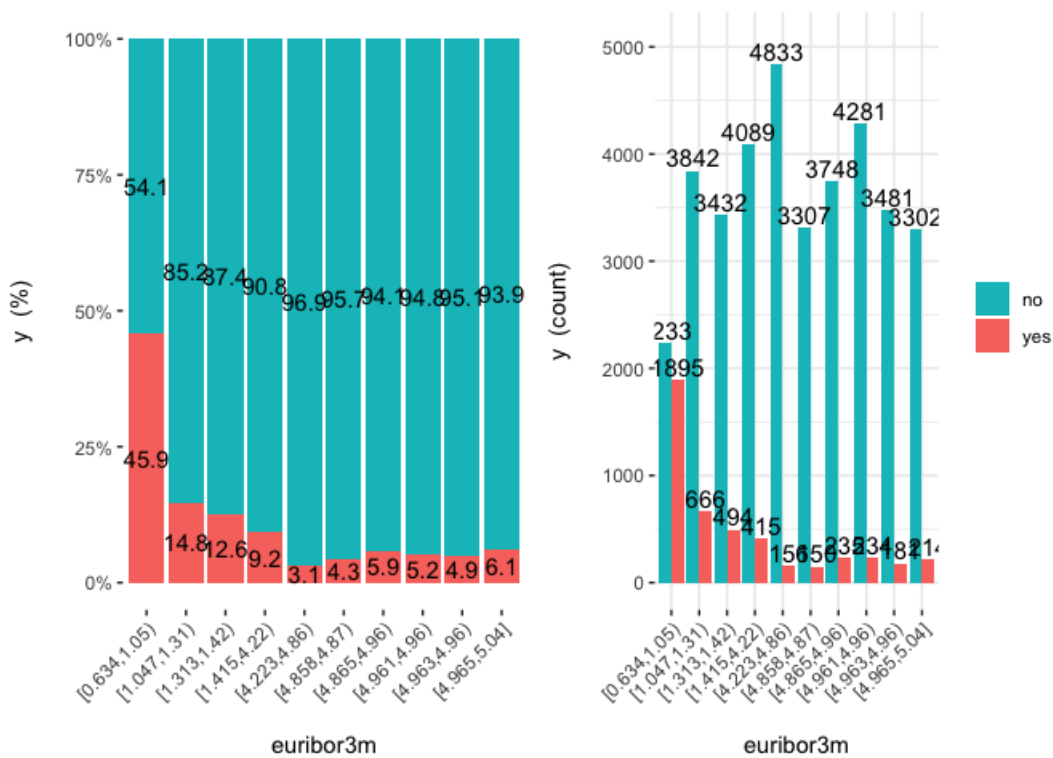












Prepare Data for Classification

- 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.

```

$ Term_Deposit                : Factor w/ 2 levels "no","yes": 1 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 ...
$ NoOfContactsPerformed        : num  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 ...
$ ConsumerPriceIndex           : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
...
$ 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.

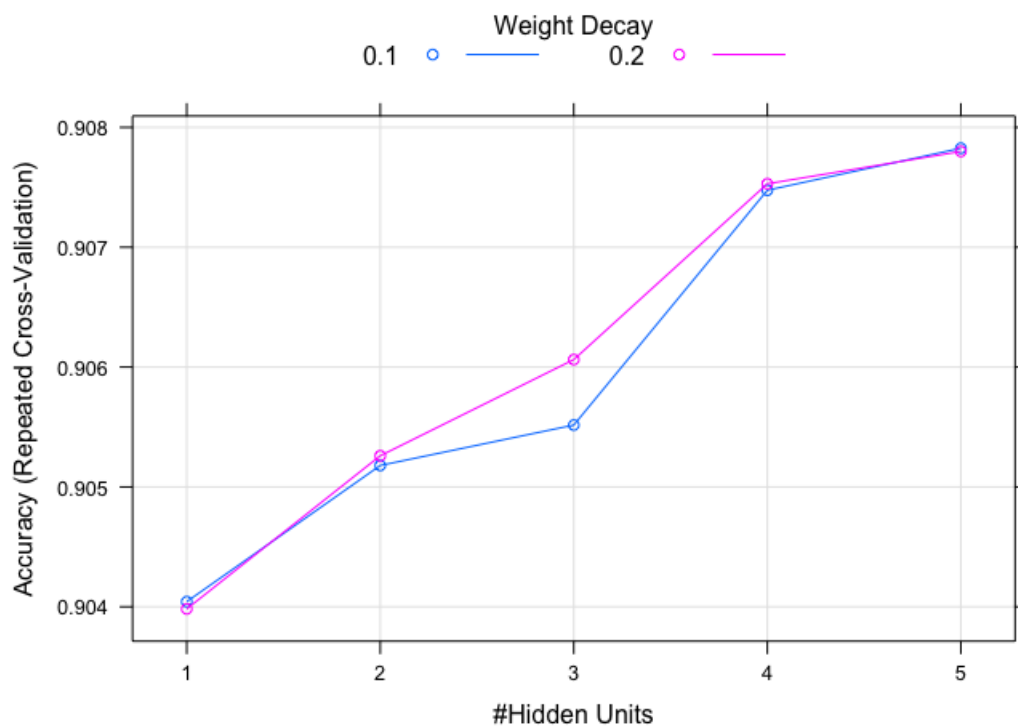
Machine Learning: Classification using Neural Networks

- Model Training
 - We can use 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 default uses the Logisitic 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
```

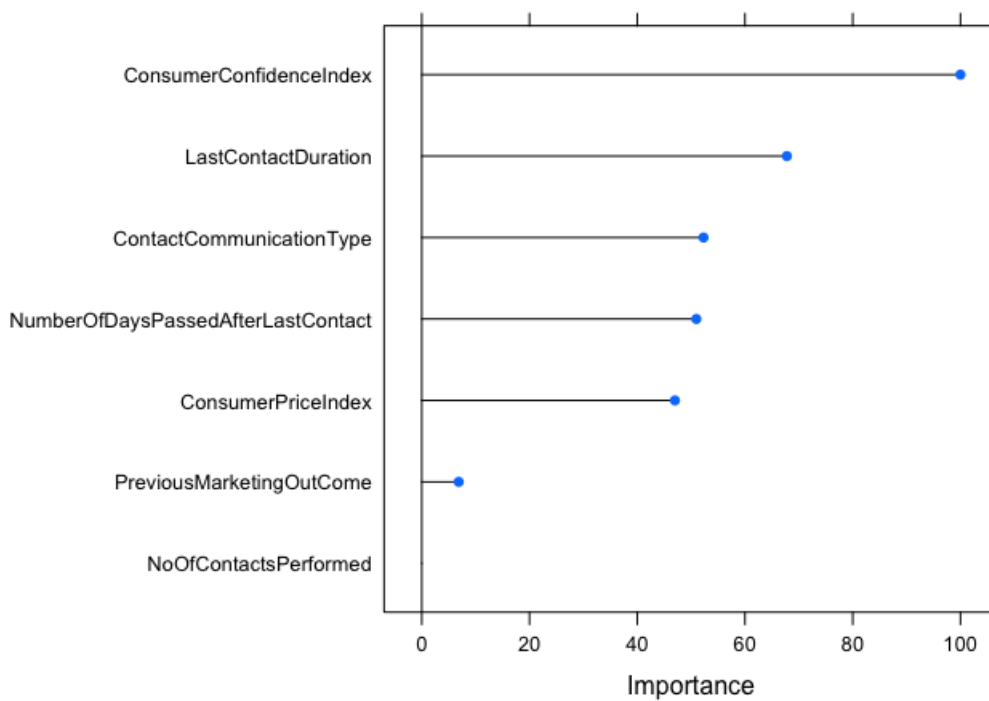
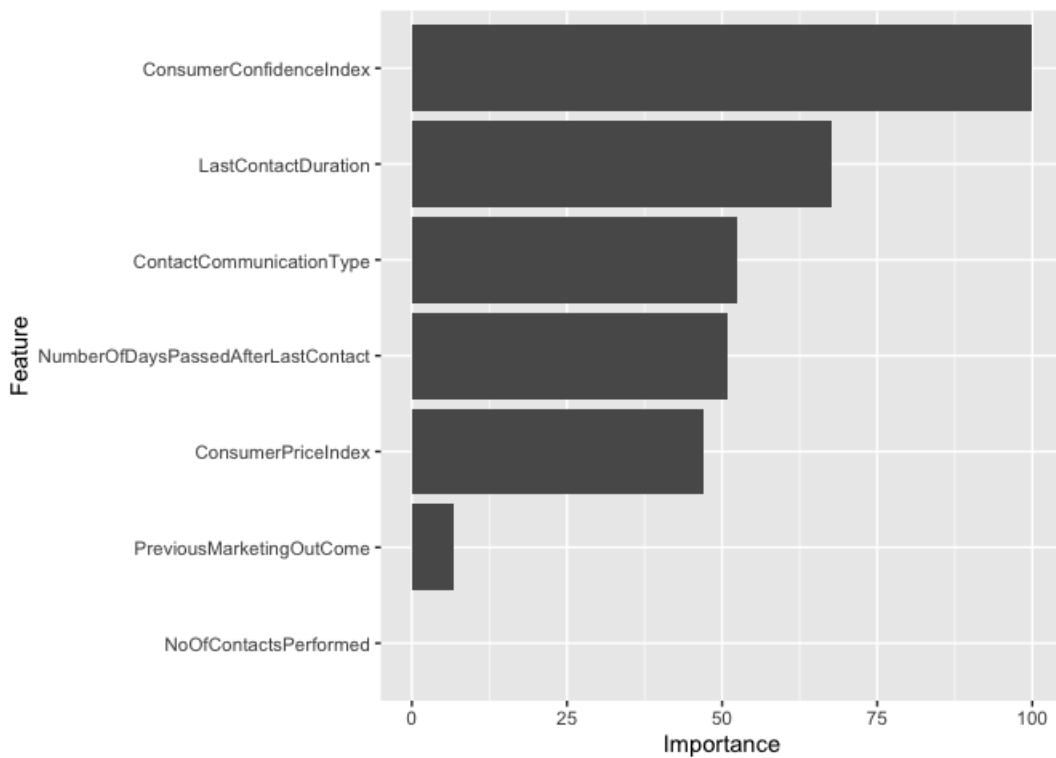
```
Sensitivity : 0.9726
Specificity : 0.3902
Pos Pred Value : 0.9191
Neg Pred Value : 0.6667
Prevalence : 0.8769
Detection Rate : 0.8529
Detection Prevalence : 0.9279
Balanced Accuracy : 0.6814
```

```
'Positive' Class : no
```

- Plotting nnet variable importance

nnet variable importance

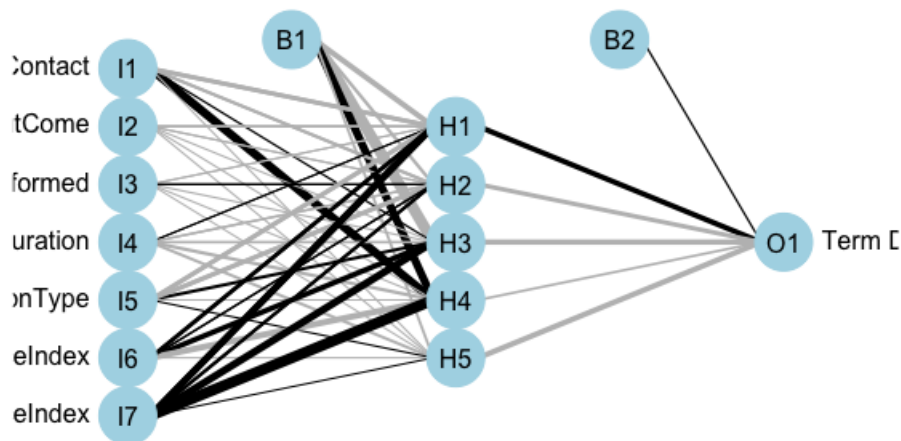
	Overall
ConsumerConfidenceIndex	100.000
LastContactDuration	67.780
ContactCommunicationType	52.298
NumberOfDaysPassedAfterLastContact	50.969
ConsumerPriceIndex	46.995
PreviousMarketingOutCome	6.867
NoOfContactsPerformed	0.000



- Graphical Representation

of our Neural Network

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 training 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 trainControl() 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, preProcess = c("center", "scale"),
                        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

Comparison between SVM models

- Comparison between SVM Linear and Radial Models.

```
Call:
summary.resamples(object = algo_results)
```

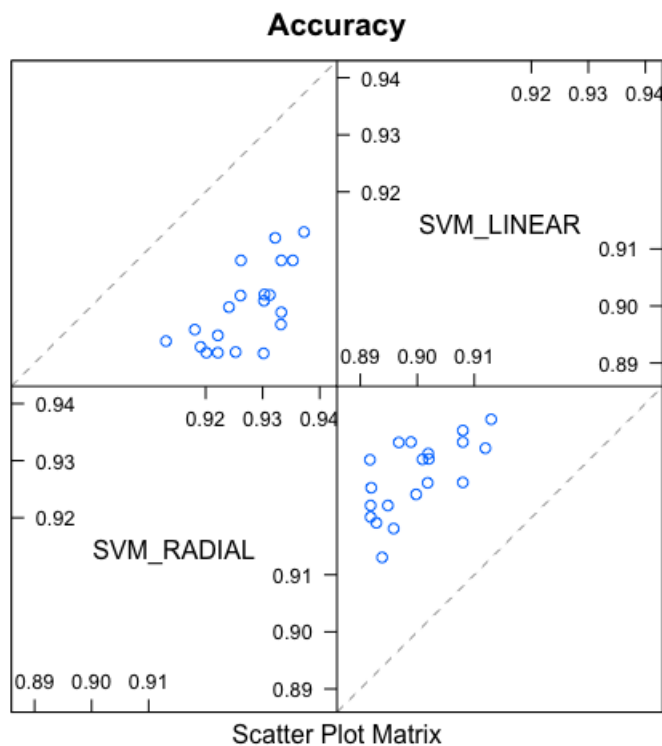
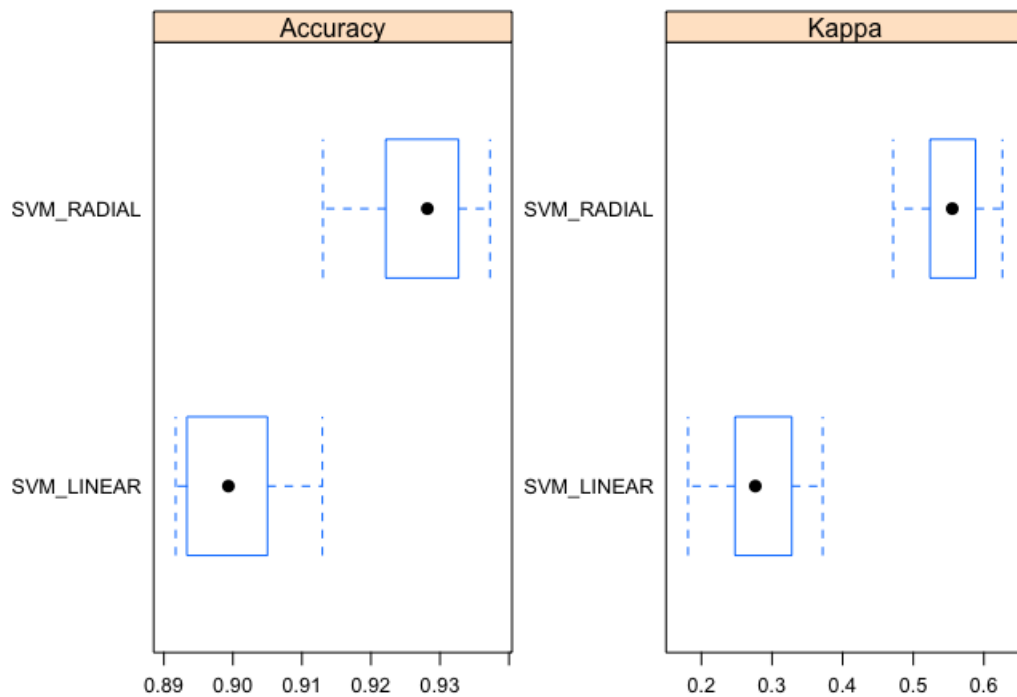
```
Models: SVM_RADIAL, SVM_LINEAR
Number of resamples: 20
```

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



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 achieves good accuracy rate(>85%) and are easier to implement.