BUA 751 - Machine Learning for Business

Bank Marketing

Content

- **Business Problem**
- **Visualization** -
  - Scatterplot
- **Statistical Analysis** -
  - Descriptive Statistics
  - Correlation Matrix
  - VIF (Variance Inflation Factor)
  - Factor Analysis
- **Categorization Analysis** -
  - Logit
  - Perceptron
  - SVM (Support Vector Machine)
  - Neural Networks
  - k-Nearest Neighbor
  - Naive Bayes
  - Decision Trees
  - Random Forest

Business Problem

A European bank is conducting a marketing campaign. We need to determine which input variables affect the desired outcome of a client taking out a term deposit.

Primary dataset contains **45,211 rows** and **17 columns**. Sample dataset contains **4,521 rows** and **17 columns**.

There are **16 input** variables (a mix of numerical, categorical and binary variables) & **1 output** variable (binary variable).

Source: *https://archive.ics.uci.edu/ml/datasets/bank+marketing*

Visualization

Overall view of relationship between dependent variable (y) with all continuous variables (age, balance, day, duration.

Steps -

1. Plotting ‘Scatterplot Matrix’ using Rcmdr in R.
2. Selecting 4 attributes at a time and plotting them by groups.
3. Creating 4 outputs containing 4 attributes respectively.

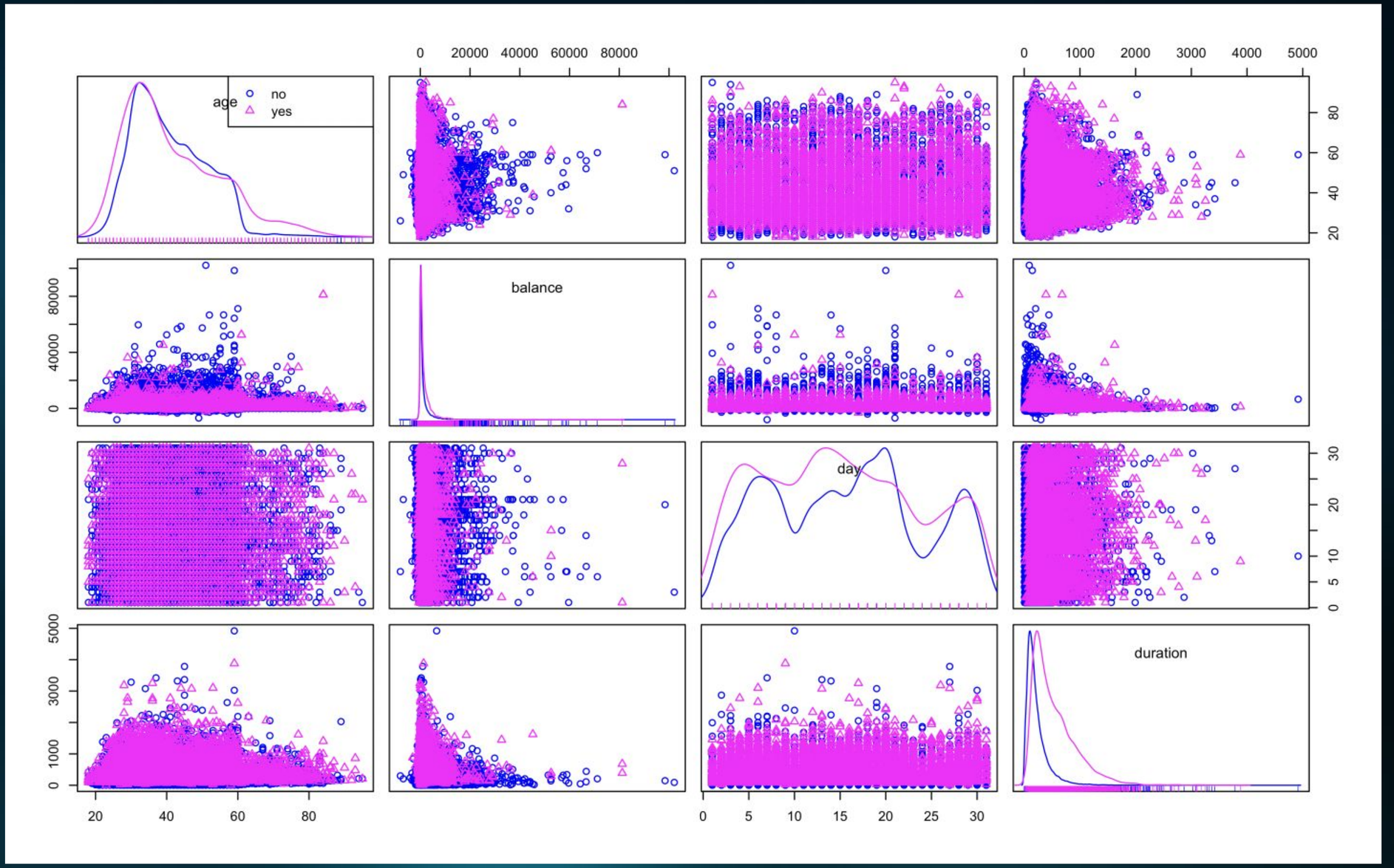


Fig 1: Overall view of relationship between dependent variable (y) with all continuous variables

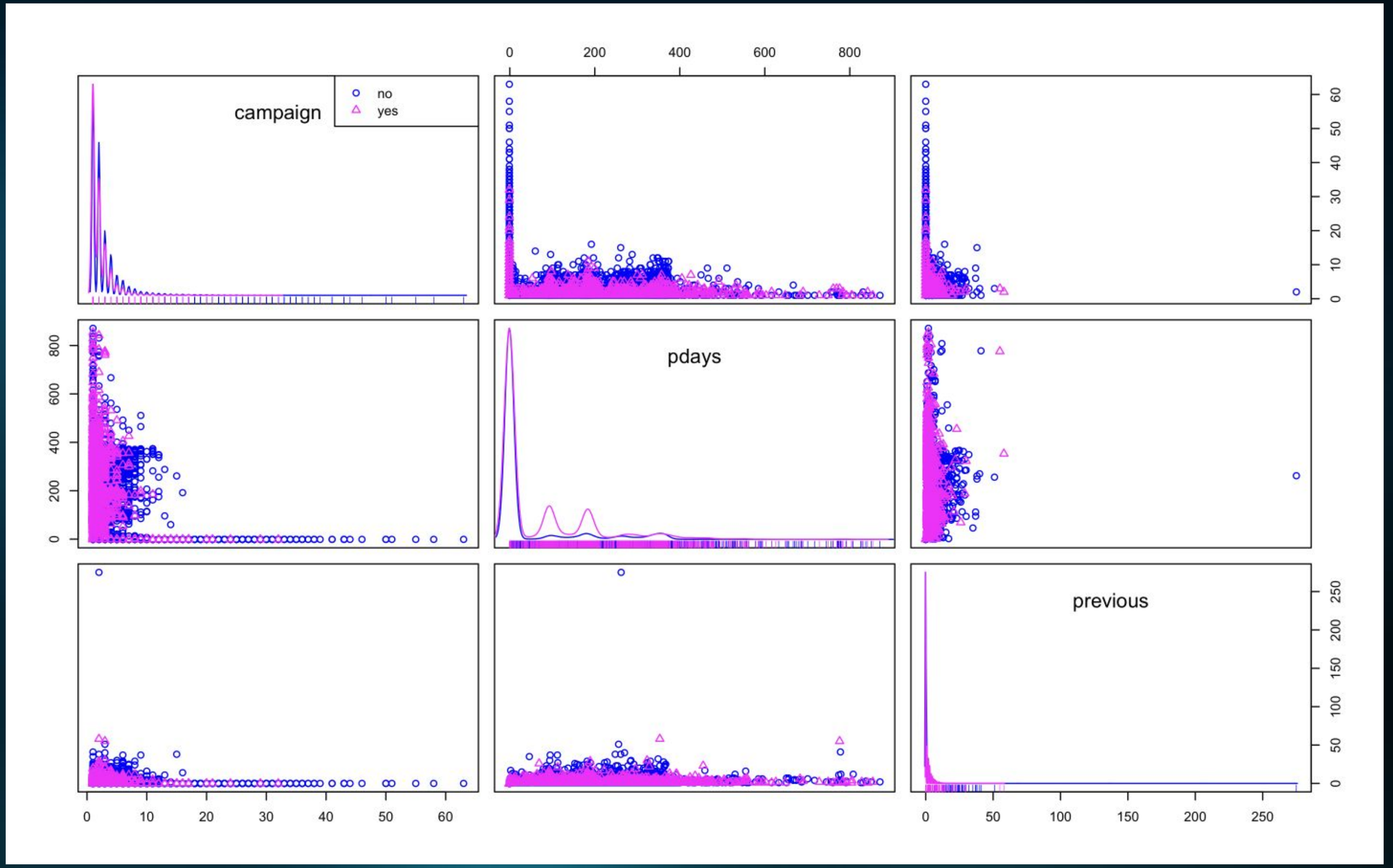


Fig 2: Overall view of relationship between dependent variable (y) with all other attributes that are continuous variables

```r
#EDUCATION

Bank <- data.frame(table(Bank$education,Bank$y1))
colnames(Bank) <- c("education","response","number_of_customers")
ggplot(data=Bank, aes(x=education,y=number_of_customers, fill=response))+
  geom_bar(stat = 'identity', position = 'dodge')+
  labs(X="Number of customers",
       y=NULL, title="Campaign Result Education Distribution")
```

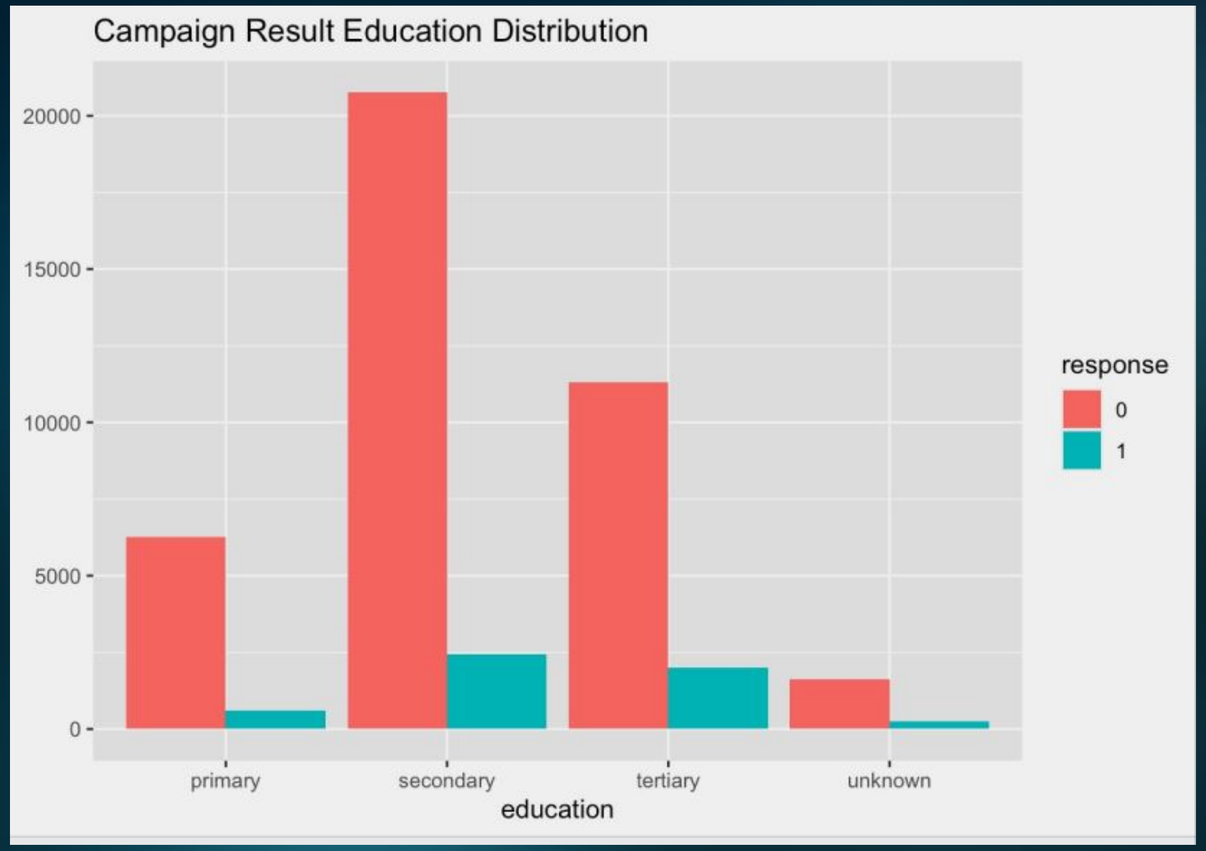


Fig : Campaign result education distribution by group

```r
Bank = data.frame(table(Bank$marital, Bank$y1))
colnames(Bank) <- c("marital","response","number_of_customers")
library(ggplot2)
ggplot(data=Bank, aes(x=marital,y=number_of_customers, fill=response))+
  geom_bar(stat = 'identity', position = 'dodge')+
  labs(X="Number of customers",
       y=NULL,
       title="Campaign Marital Status Distribution")
```

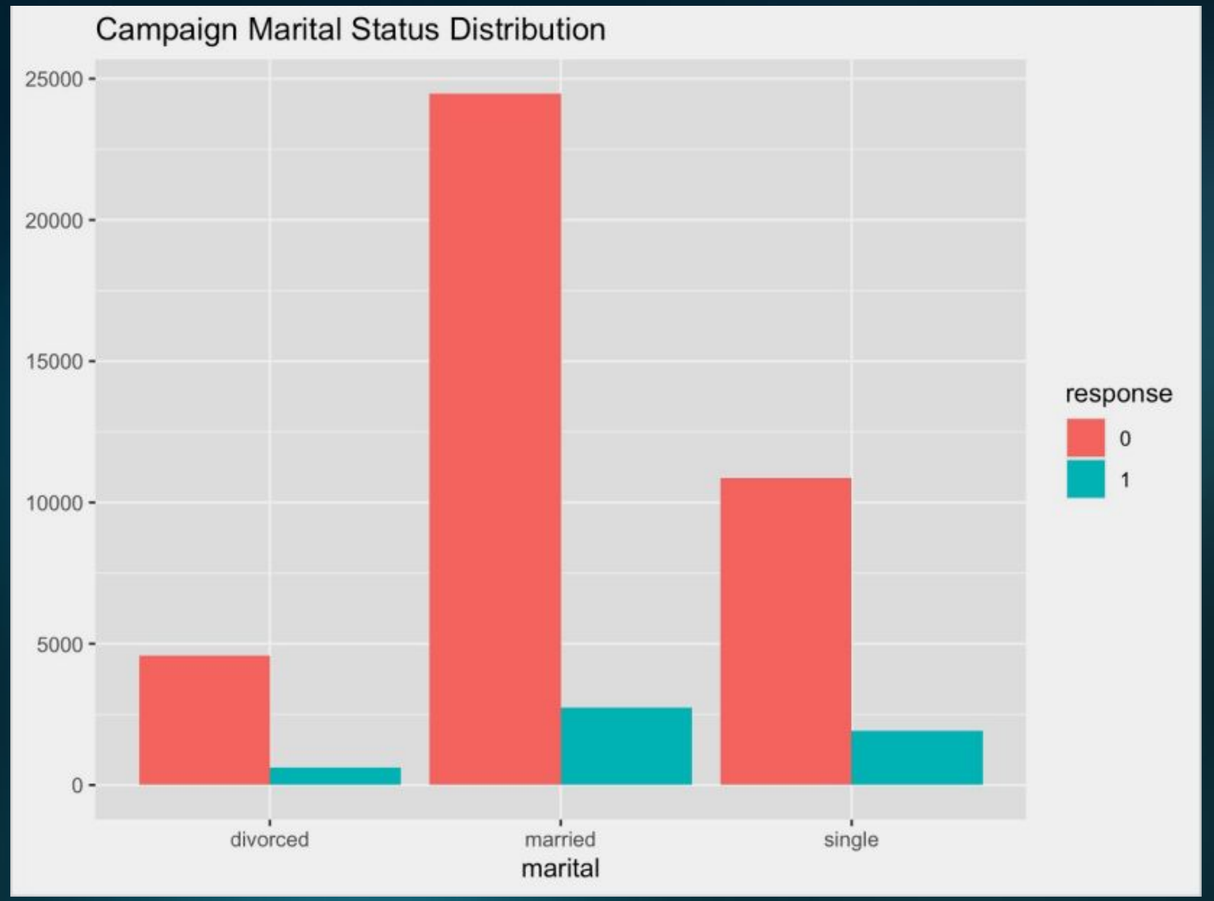


Fig : Campaign result marriage distribution by group

```r
#Customer Default Distribution

def_tab <- data.frame(table(Bank$default))
colnames(def_tab) <- c("default","number_of_customers")
ggplot(data=def_tab, aes(x=default,y=number_of_customers, fill=default))+
  geom_bar(stat = 'identity', position = 'dodge')+
  labs(X="Number of customers",
       y=NULL,
       title="Default Customer Distribution")
```

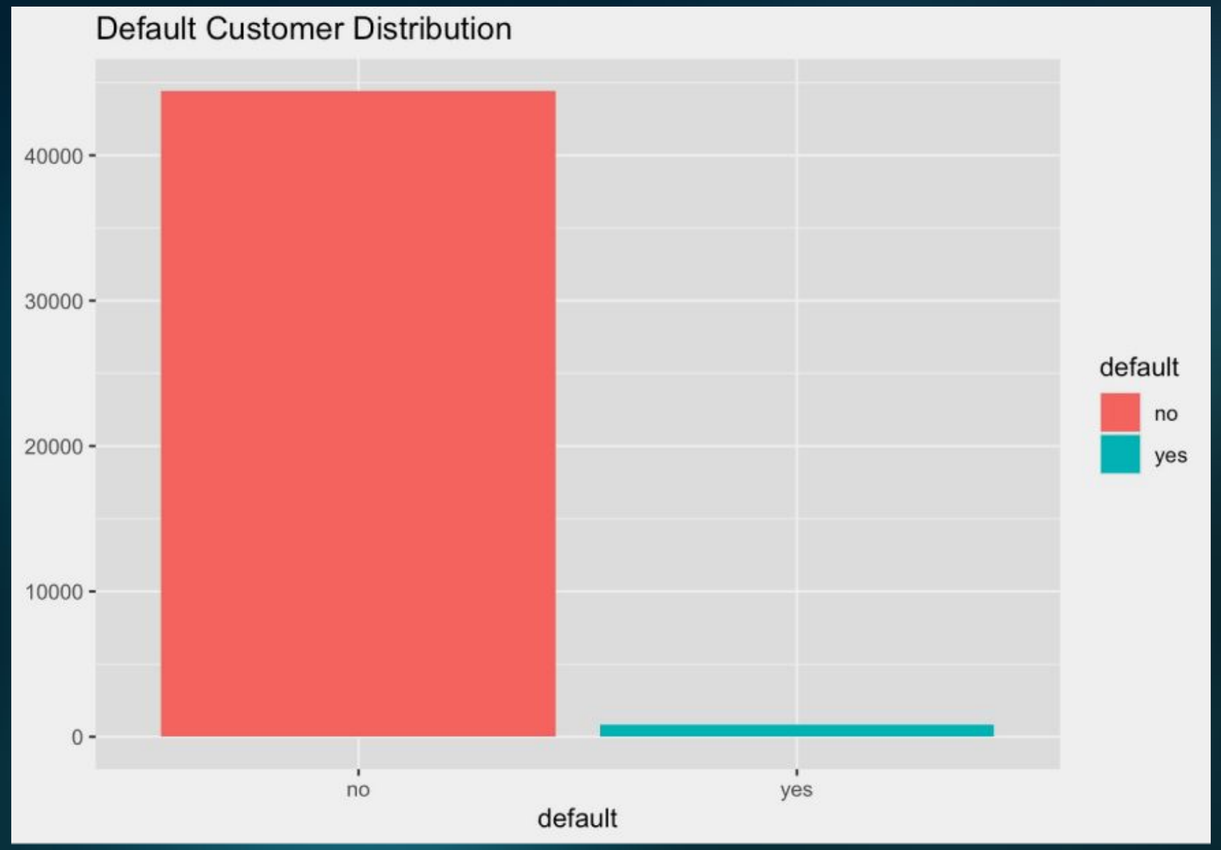


Fig : Campaign result default distribution by group

```r
#JOB-WISE RESPONSE
job_y_tab <- data.frame(table(Bank$job, Bank$y))
colnames(job_y_tab) <- c("job","response","count")
ggplot(data=job_y_tab, aes(x=count,y=reorder(job,count), fill=response))+
geom_bar(stat = 'identity', position = 'dodge')+
labs(X="Number of customers",y=NULL, title="Campaign Result Job Distribution")
```

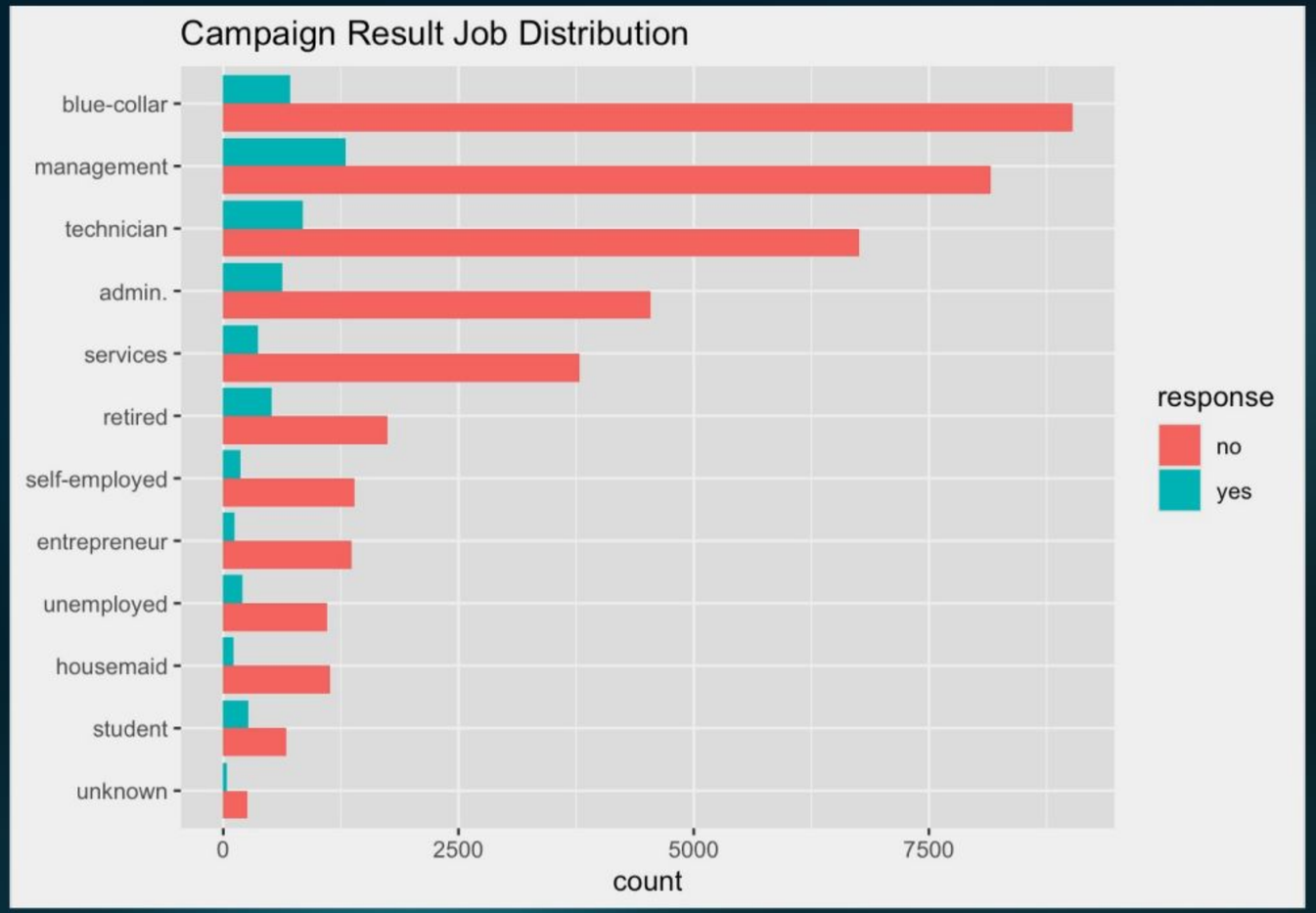


Fig : Campaign result job-wise distribution by group

```r
#LOAN STATUS AND CUSTOMER RESPONSE
loan_tab <- data.frame(table(Bank$loan,Bank$y))
colnames(loan_tab) <- c("loan", "response","count")
ggplot(data=loan_tab, aes(x=loan,y=count,fill=response))+
  geom_bar(stat = 'identity',position = 'dodge')+
  labs(X="Number of customers",
       y=NULL,
       title="Loan Status & Customer Response")+theme_light()
```

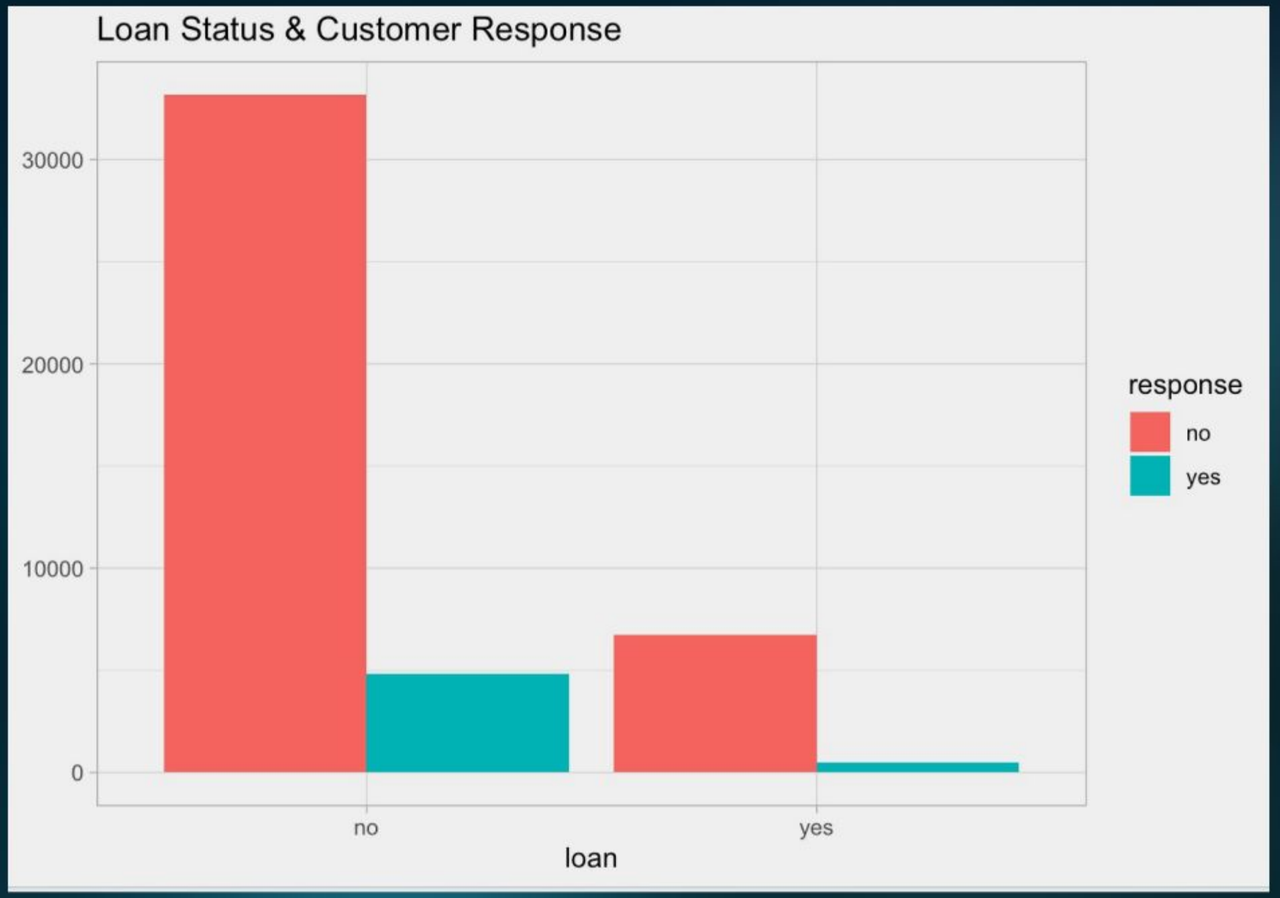


Fig : Campaign result loan status & customer response distribution by group

Statistical Analysis

Overall **descriptive statistics** - mean, standard deviation, Inter-Quartile Region (IQR), skewness, kurtosis, quantiles and number of observations of all continuous variables. dependent variable (y)

Steps -

1. Plotting ‘numSummary’ function using Rcmdr.

```r
library(Rcmdr)
numSummary(bank[,c("age", "balance", "campaign", "D_default", "D_housing", "D_loan", "day", "duration", "pdays", "previous"),
                drop=FALSE], statistics=c("mean", "sd", "IQR", "quantiles", "skewness", "kurtosis"), quantiles=c(0,.25,.5,.75,1), type="2")
```

	mean	sd	IQR	skewness	kurtosis	0%	25%	50%	75%	100%	n
age	40.93621021	10.6187620	15	0.68481793	0.3195704	18	33	39	48	95	45211
balance	1362.27205769	3044.7658292	1356	8.36030833	140.7515466	-8019	72	448	1428	102127	45211
campaign	2.76384066	3.0980209	2	4.89865017	39.2496508	1	1	2	3	63	45211
D_default	0.01802659	0.1330489	0	7.24537511	50.4976943	0	0	0	0	1	45211
D_housing	0.55583818	0.4968778	1	-0.22476613	-1.9495664	0	0	1	1	1	45211
D_loan	0.16022649	0.3668200	0	1.85261710	1.4322535	0	0	0	0	1	45211
day	15.80641879	8.3224762	13	0.09307901	-1.0598974	1	8	16	21	31	45211
duration	258.16307978	257.5278123	216	3.14431810	18.1539153	0	103	180	319	4918	45211
pdays	40.19782796	100.1287460	0	2.61571547	6.9351952	-1	-1	-1	-1	871	45211
previous	0.58032337	2.3034410	0	41.84645447	4506.8606602	0	0	0	0	275	45211

Fig 3: Overall descriptive statistics of all numerical and binary variables

Correlation Matrix

A correlation matrix is simply a table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all possible correlated pairs of values in a table

Steps:

1. Import dataset into R
2. Using cor( ) function finding Pearsons correlation among the variables
3. Using R code generating a correlation list of top 10 correlated variables in descending order
4. Using ggpairs( ) function to create an area graph, scatterplot and correlation of 2 most correlated variables in the dataset

```
myData <- bank[,c(1:10)]
res <- cor(myData)
round(res,4)
```

	age	day	duration	campaign	pdays	previous	balance	D_housing	D_loan	D_default
age	1.0000	-0.0091	-0.0046	0.0048	-0.0238	0.0013	0.0978	-0.1855	-0.0157	-0.0179
day	-0.0091	1.0000	-0.0302	0.1625	-0.0930	-0.0517	0.0045	-0.0280	0.0114	0.0094
duration	-0.0046	-0.0302	1.0000	-0.0846	-0.0016	0.0012	0.0216	0.0051	-0.0124	-0.0100
campaign	0.0048	0.1625	-0.0846	1.0000	-0.0886	-0.0329	-0.0146	-0.0236	0.0100	0.0168
pdays	-0.0238	-0.0930	-0.0016	-0.0886	1.0000	0.4548	0.0034	0.1242	-0.0228	-0.0300
previous	0.0013	-0.0517	0.0012	-0.0329	0.4548	1.0000	0.0167	0.0371	-0.0110	-0.0183
balance	0.0978	0.0045	0.0216	-0.0146	0.0034	0.0167	1.0000	-0.0688	-0.0844	-0.0667
D_housing	-0.1855	-0.0280	0.0051	-0.0236	0.1242	0.0371	-0.0688	1.0000	0.0413	-0.0060
D_loan	-0.0157	0.0114	-0.0124	0.0100	-0.0228	-0.0110	-0.0844	0.0413	1.0000	0.0772
D_default	-0.0179	0.0094	-0.0100	0.0168	-0.0300	-0.0183	-0.0667	-0.0060	0.0772	1.0000

Fig 4: Overall correlation matrix of all numerical and binary variables

```r
library(dplyr)
library(tidyr)

myData <- data.frame(corMatrix)
cor(myData[,c(1:10)]) %>%
  as.data.frame(myData) %>%
  mutate(var1 = rownames(.)) %>%
  gather(var2, value, -var1) %>%
  arrange(desc(value)) %>%
  group_by(value) %>%
  filter(row_number()==1)
```

	var1	var2	value
	<chr>	<chr>	<dbl>
1	1	age	1
2	6	pdays	0.455
3	4	day	0.162
4	8	pdays	0.124
5	7	age	0.0978
6	10	D_loan	0.0772
7	9	D_housing	0.0413
8	8	previous	0.0371
9	7	duration	0.0216
10	10	campaign	0.0168

Fig 5: Overall correlation list of all numerical and binary variables rankwise (in descending order).

```
library(GGally)
ggpairs(bank[, c("previous", "pdays")])
```

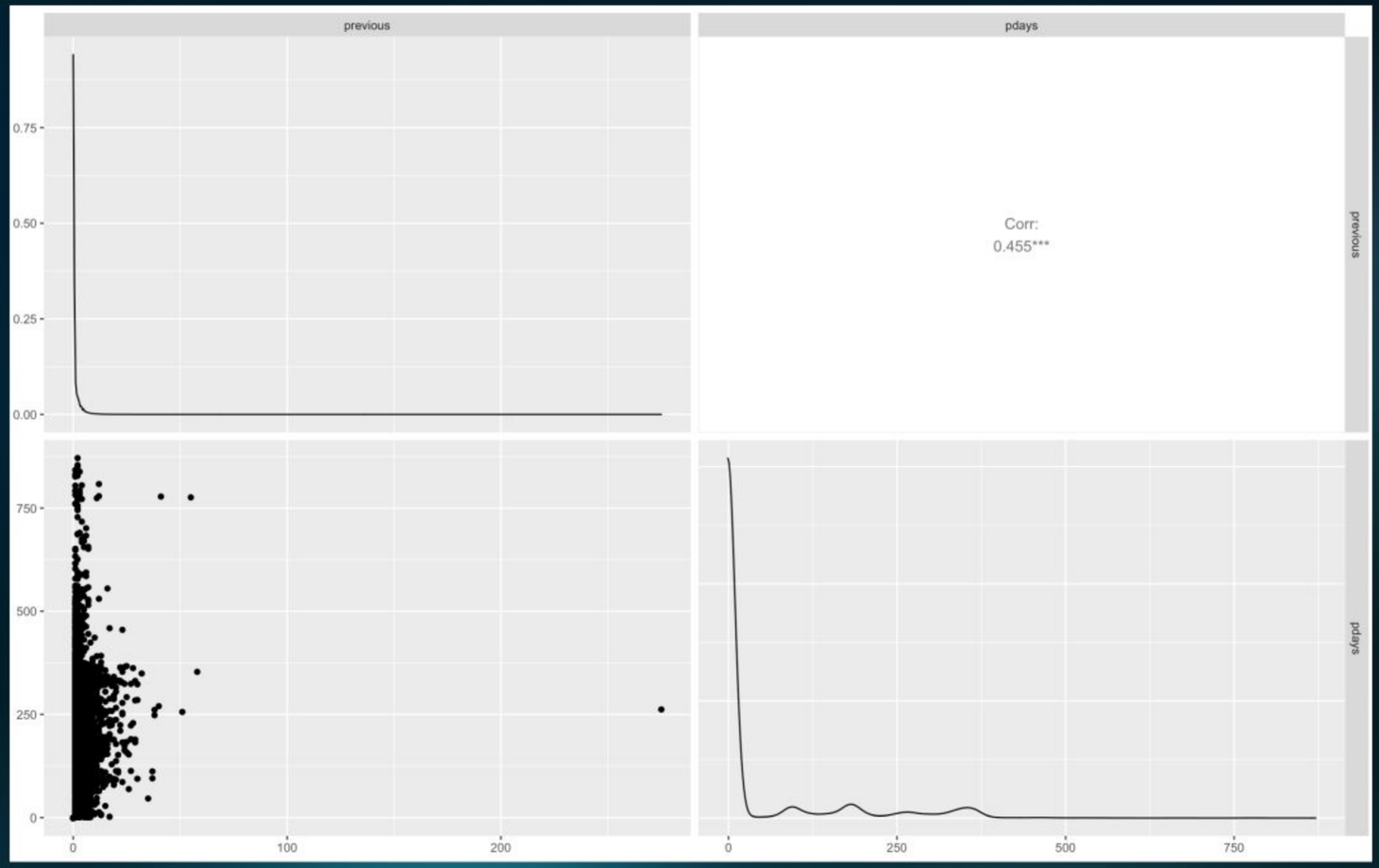


Fig 6: ggpairs function depicting scatterplot, area graphs and correaltion of 2 highest correlated variables.

Variance Inflation Factor (VIF)

VIF is a measure of the amount of multicollinearity in regression analysis. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model.

Steps -

1. Conduct logistic regression with “y” as dependent variable and columns 1-10 as independent variables using Rcmdr.
2. Run numerical diagnosis using ‘vif’ function.
3. Interpreting the results making note of variables having VIF value less than 1, between 1 and 5 and greater than 5.

poutcome1	4.072452
pdays	3.894604
previous	1.325227
age	1.282478
marital1	1.219666
contact1	1.19626
housing1	1.158345
month1	1.099495
education1	1.067347
job1	1.047432
campaign	1.044648
day	1.044309
balance	1.036892
loan1	1.021038
default1	1.011803
duration	1.009846

Fig 7: VIF Analysis.

Variance Inflation Factor (VIF)

In general terms,

- VIF equal to 1 = variables are not correlated & multicollinearity does not exist in the regression model.
- VIF between 1 and 5 = variables are moderately correlated
- VIF greater than 5 = variables are highly correlated & higher posibility of multicollinearity to exist.

In our case, maximum variables after converting them into numeric have a VIF value between 1 and 2.

Factor Analysis

Factor analysis assumes that a hidden factor determines your x-variables. It calculates the factor and is then used instead of x-variables. It basically identifies how many unique concepts are captured in the variables in your data.

Steps -

1. Using the scree( ) function create a scree plot that will indicate how the measures collapse into unique factors.
2. Determining exactly how many factors do we need using Rcmdr.
3. Determining the variables to be used in the regression.

```
install.packages("psych",dependencies=TRUE)
library(psych)
scree(bank)
```

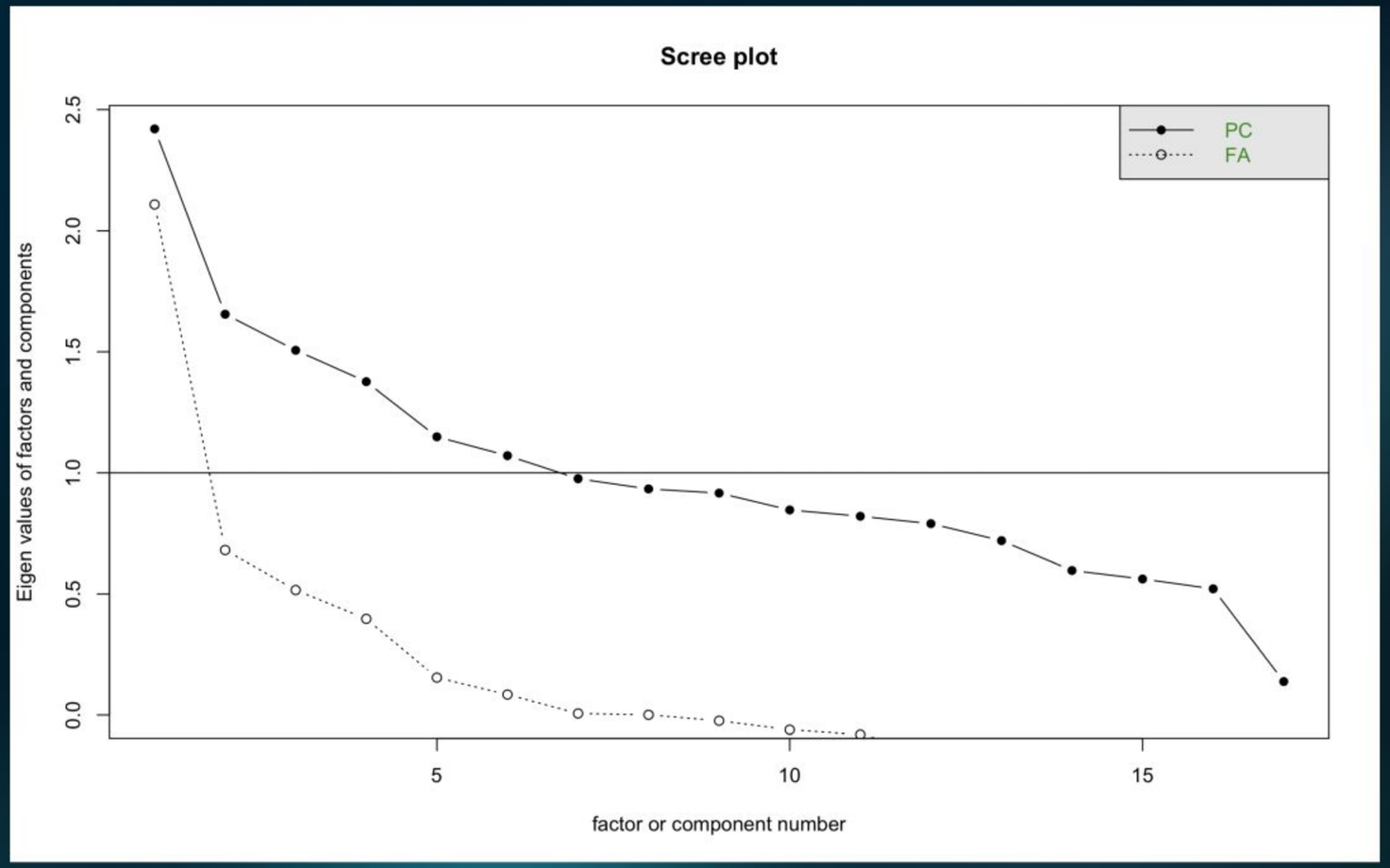
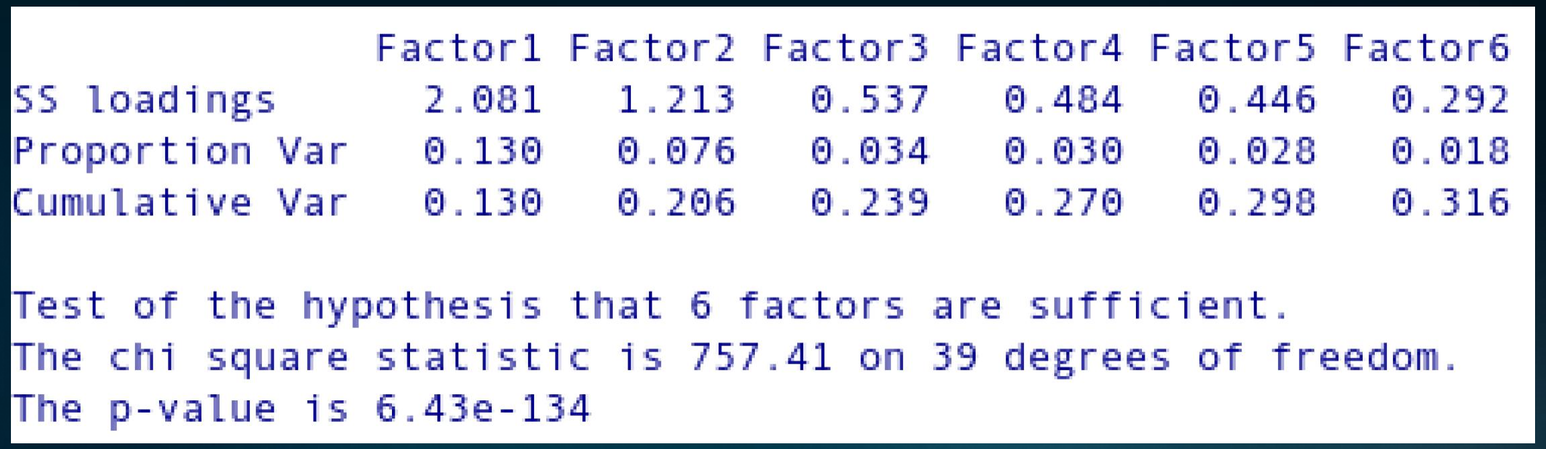


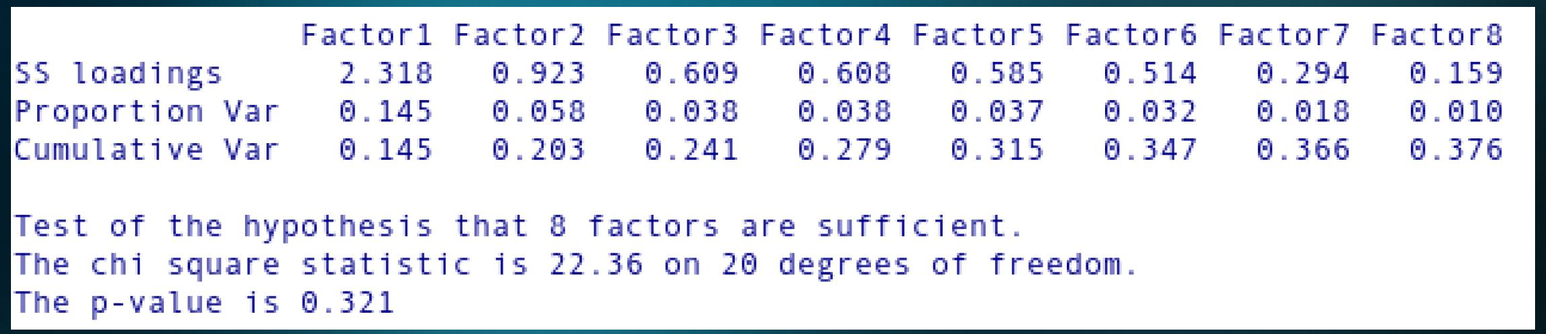
Fig 8: Scree plot depicts that there are 6 factors (above Eigen value 1).



```
                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
SS loadings       2.081   1.213   0.537   0.484   0.446   0.292
Proportion Var    0.130   0.076   0.034   0.030   0.028   0.018
Cumulative Var    0.130   0.206   0.239   0.270   0.298   0.316

Test of the hypothesis that 6 factors are sufficient.
The chi square statistic is 757.41 on 39 degrees of freedom.
The p-value is 6.43e-134
```

Fig 9: With factor set as 6, we did not get a p-value greater than 0.05, hence we used the sample dataset



```
                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
SS loadings       2.318   0.923   0.609   0.608   0.585   0.514   0.294   0.159
Proportion Var    0.145   0.058   0.038   0.038   0.037   0.032   0.018   0.010
Cumulative Var    0.145   0.203   0.241   0.279   0.315   0.347   0.366   0.376

Test of the hypothesis that 8 factors are sufficient.
The chi square statistic is 22.36 on 20 degrees of freedom.
The p-value is 0.321
```

Fig 10: With a factor set as **8**, we did get p-value greater than 0.05. This means that the original variables can be collapsed into **8** concepts.

Logit

Logistic regression, also called a logit model, is used to model dichotomous outcome variables. In the logit model the log odds of the outcome is modeled as a linear combination of the predictor variables.

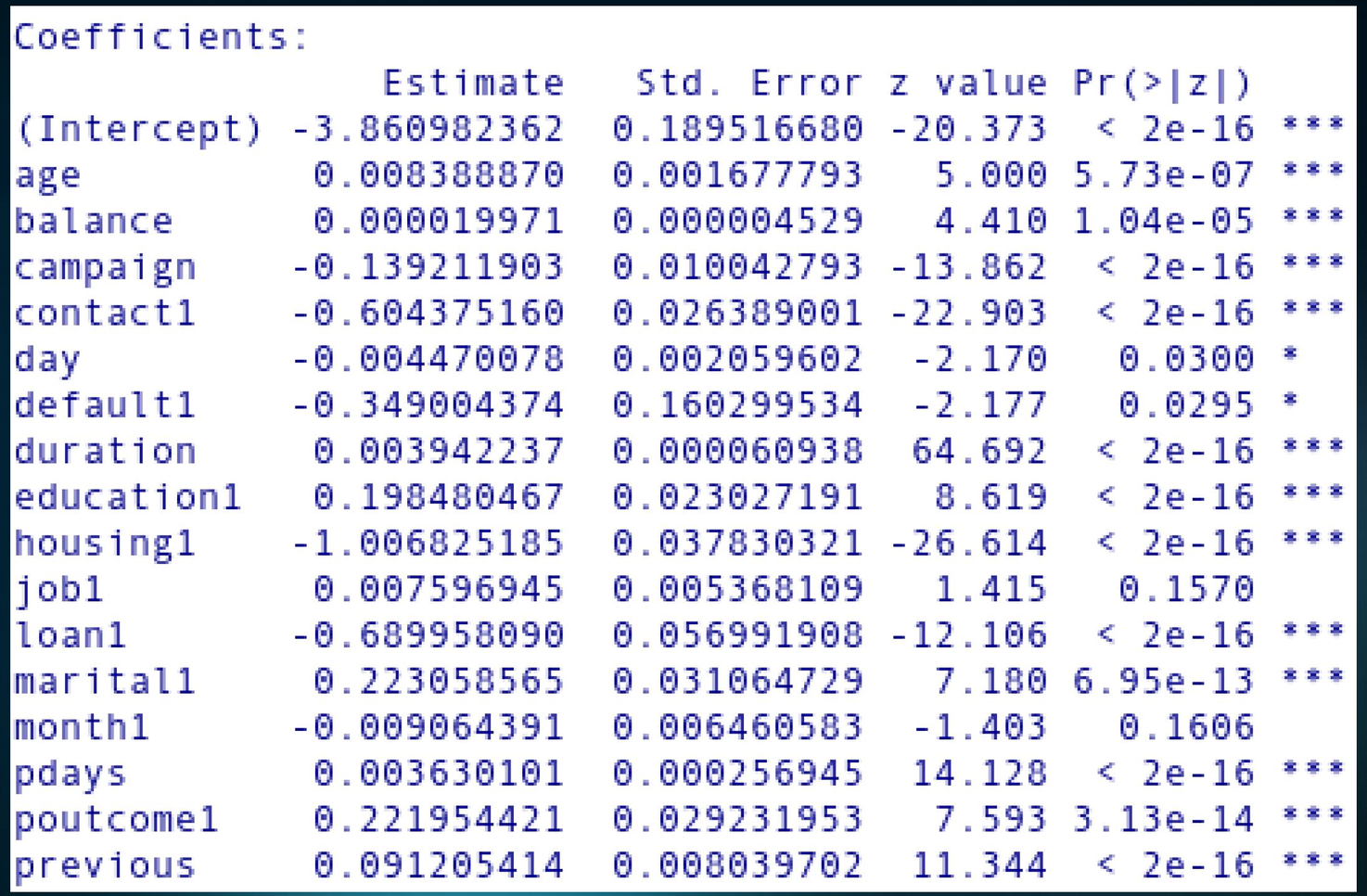
Steps -

1. Using the main dataset and carrying logit analysis using Rcmdr.
2. Using the main dataset with dummy variables to carry logit analysis using Rcmdr.
3. Capturing result and comparing both the models.

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-2.535637780	0.183703164	-13.803	< 2e-16	***
age	0.000112719	0.002205165	0.051	0.959233	
balance	0.000012835	0.000005148	2.493	0.012651	*
campaign	-0.090781782	0.010137033	-8.955	< 2e-16	***
contact[T.telephone]	-0.163374330	0.075185612	-2.173	0.029784	*
contact[T.unknown]	-1.623216856	0.073171806	-22.184	< 2e-16	***
day	0.009968922	0.002496619	3.993	6.53e-05	***
default[T.yes]	-0.016681215	0.162837013	-0.102	0.918407	
duration	0.004193695	0.000064532	64.986	< 2e-16	***
education[T.secondary]	0.183528258	0.064792557	2.833	0.004618	**
education[T.tertiary]	0.378941502	0.075319068	5.031	4.88e-07	***
education[T.unknown]	0.250478833	0.103896567	2.411	0.015915	*
housing[T.yes]	-0.675384337	0.043869060	-15.395	< 2e-16	***
job[T.blue-collar]	-0.309872593	0.072669201	-4.264	2.01e-05	***
job[T.entrepreneur]	-0.357103762	0.125564459	-2.844	0.004455	**
job[T.housemaid]	-0.504001652	0.136469021	-3.693	0.000221	***
job[T.management]	-0.165278440	0.073292526	-2.255	0.024130	*
job[T.retired]	0.252362639	0.097217516	2.596	0.009436	**
job[T.self-employed]	-0.298336079	0.111996400	-2.664	0.007726	**
job[T.services]	-0.223797106	0.084064904	-2.662	0.007763	**
job[T.student]	0.382135715	0.109029897	3.505	0.000457	***
job[T.technician]	-0.176016548	0.068931178	-2.554	0.010664	*
job[T.unemployed]	-0.176713126	0.111642461	-1.583	0.113456	
job[T.unknown]	-0.313264379	0.233463307	-1.342	0.179656	
loan[T.yes]	-0.425371663	0.059989904	-7.091	1.33e-12	***
marital[T.married]	-0.179453495	0.058910580	-3.046	0.002318	**
marital[T.single]	0.092497647	0.067260667	1.375	0.169066	
month[T.aug]	-0.693907553	0.078474461	-8.842	< 2e-16	***
month[T.dec]	0.691124324	0.176682753	3.912	9.17e-05	***
month[T.feb]	-0.147320938	0.089413545	-1.648	0.099427	.
month[T.jan]	-1.261718795	0.121702801	-10.367	< 2e-16	***
month[T.jul]	-0.830795589	0.077404978	-10.733	< 2e-16	***
month[T.jun]	0.453622601	0.093669266	4.843	1.28e-06	***
month[T.mar]	1.589890543	0.119853742	13.265	< 2e-16	***
month[T.may]	-0.399111424	0.072285121	-5.521	3.36e-08	***
month[T.nov]	-0.873398521	0.084409802	-10.347	< 2e-16	***
month[T.oct]	0.881437433	0.108030525	8.159	3.37e-16	***
month[T.sep]	0.874058052	0.119497320	7.314	2.58e-13	***
pdays	-0.000102685	0.000306089	-0.335	0.737268	
poutcome[T.other]	0.203478400	0.089855382	2.265	0.023543	*
poutcome[T.success]	2.291056017	0.082348964	27.821	< 2e-16	***
poutcome[T.unknown]	-0.091793506	0.093474710	-0.982	0.326093	
previous	0.010152353	0.006502908	1.561	0.118476	

Fig 11: Using the main dataset. The result shows all variables are statistically significant apart from **age, default(yes), job(unemployed), job(unknown), marital(single), month(feb), pdays, poutcome(unknown), previous**.



```
Coefficients:
                Estimate   Std. Error z value Pr(>|z|)
(Intercept) -3.860982362  0.189516680 -20.373  < 2e-16 ***
age          0.008388870  0.001677793   5.000 5.73e-07 ***
balance      0.000019971  0.000004529   4.410 1.04e-05 ***
campaign    -0.139211903  0.010042793 -13.862  < 2e-16 ***
contact1    -0.604375160  0.026389001 -22.903  < 2e-16 ***
day         -0.004470078  0.002059602  -2.170   0.0300 *
default1    -0.349004374  0.160299534  -2.177   0.0295 *
duration     0.003942237  0.000060938  64.692  < 2e-16 ***
education1   0.198480467  0.023027191   8.619  < 2e-16 ***
housing1    -1.006825185  0.037830321 -26.614  < 2e-16 ***
job1         0.007596945  0.005368109   1.415   0.1570
loan1       -0.689958090  0.056991908 -12.106  < 2e-16 ***
marital1     0.223058565  0.031064729   7.180 6.95e-13 ***
month1      -0.009064391  0.006460583  -1.403   0.1606
pdays        0.003630101  0.000256945  14.128  < 2e-16 ***
poutcome1    0.221954421  0.029231953   7.593 3.13e-14 ***
previous     0.091205414  0.008039702  11.344  < 2e-16 ***
```

Fig 12: Using the main dataset with dummy variables. The result shows all variables are statistically significant apart from **job** and **month**.

Result Interpretation example: For every increase in **age**, the **log** odds of a client taking out a term deposit increases by **0.008**

Perceptron

Perceptron is a linear classifier (binary). Also, it is used in supervised learning. It helps to classify the given input data by using a single layer neural network. Perceptron fails as it can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

Steps -

1. Using the main dataset and creating the perceptron learning algorithm.
2. Using Excel doing analysis to verify if Perceptron fails or not.
3. Repeating steps 1 and 2 for sample dataset.

```
> bankSubset2D <-
bank[,c("age","job1","marital1","education1","default1","balance","housing1","loan1","contact1","day","month1","duration","campaign",
"pdays","previous","poutcome1","y")]
> bankSubset2D$class <- lapply(bankSubset2D$y, function(x) {
+  if(x == "no")
+  bankSubset2D$class <- -1
+  else if(x == "yes")
+  bankSubset2D$class <- 1
+  else
+  bankSubset2D$class <- NULL
+ })
```

```
> X<-
bankSubset2D[,c("age","job1","marital1","education1","balance","housing1","default1","loan1","contact1","day","month1","duration","ca
mpaign","pdays","previous","poutcome1")] # Input Matrix
>
> y <- bankingSubset2D$class # Output Vector
```

```
> perceptron <- function(X, y, numEpochs) {
+  results <- list()
+  w <- runif(ncol(X), -10, 10) #Initalize weights
+
+  # For loop - number of generations(epochs) - number of times dataset is ran through
+  for(j in 1:numEpochs) {
+  predictedResult <- numeric(length=100) # Initalize predictedResult vector
+  numIncorrect = 0 # Keeps track of # of missclassified points
+
+  # For loop - loop throught dataset
+  for(i in 1:length(y)) {
+  xi = as.numeric(unlist(X[i,])) # Convert dataframe to vector
+  predictedResult[i] = sign(w %*% xi) # Predict the point
+
+  # If predicted point is incorrect - change weight
+  if(predictedResult[i] != y[i]) {
+  numIncorrect = numIncorrect + 1 # Add one to # of missclassified points
+  w <- w + as.numeric(y[i]) * xi # Update the weight w <- w + WiXi
+  }
+  }
+  # Print results of this generation(epoch)
+  cat("\nEpoch #: ", j)
+  cat("\nNumber Incorrect: ", numIncorrect)
+  cat("\nFinal Weight: ", w)
+  }
+ }
> results = perceptron(X,y, 8)
```

Fig 13: Perceptron learning algorithm using main dataset containing 45212 observations.

```
Epoch #:  2
Number Incorrect:  6145
Final Weight:  -42724.13 -8115.913 -3681.637 -3307.598 -2907.76 -1409.122 -236.5062 -628.5662 -3428.406 -23852.35 -9512.37 8955.84
-5033.558 2089.312 171.7858 -5441.52
Epoch #:  3
Number Incorrect:  6029
Final Weight:  -47004.13 -9900.913 -4608.637 -4014.598 -2659.76 -1875.122 -319.5062 -864.5662 -4301.406 -27940.35 -11872.37 8605.84
-6498.558 417.3118 293.7858 -6517.52
Epoch #:  4
Number Incorrect:  5921
Final Weight:  -49682.13 -11350.91 -5394.637 -4562.598 -1203.76 -2280.122 -390.5062 -1040.566 -5025.406 -31473.35 -13863.37 11907.84
-7899.558 2845.312 485.7858 -7370.52
Epoch #:  5
Number Incorrect:  5932
Final Weight:  -51339.13 -12569.91 -6124.637 -5064.598 -3264.76 -2690.122 -462.5062 -1260.566 -5789.406 -34353.35 -15596.37 12127.84
-9138.558 2628.312 640.7858 -8106.52
Epoch #:  6
Number Incorrect:  5822
Final Weight:  -49245.13 -13386.91 -6638.637 -5393.598 -1142.76 -3050.122 -529.5062 -1420.566 -6396.406 -35697.35 -16807.37 12157.84
-10217.56 3044.312 821.7858 -8510.52
Epoch #:  7
Number Incorrect:  5860
Final Weight:  -49050.13 -14308.91 -7244.637 -5775.598 -2386.76 -3418.122 -589.5062 -1618.566 -7040.406 -37154.35 -18298.37 9848.84
-11295.56 947.3118 779.7858 -9026.52
Epoch #:  8
Number Incorrect:  5788
Final Weight:  -48313.13 -15124.91 -7842.637 -6156.598 -3429.76 -3769.122 -652.5062 -1805.566 -7676.406 -38218.35 -19646.37 12904.84
-12279.56 2868.312 928.7858 -9537.52
```

Fig 14: Perceptron outcome using main dataset containing 45212 observations with 5788 incorrect.

Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
y	Weights	-48313.13	-15124.91	-7842.637	-6156.598	-3429.76	-3769.122	-652.5062	-1805.566	-7676.406	-38218.35	-19646.37	12904.84	-12279.56	2868.312	928.7858	-9537.52	Sum
no		-2802162	-75624.55	-15685.27	-18469.79	0	-8077228.45	-652.5062	0	-23029.22	-191091.8	-98231.85	3368163.24	-12279.56	-2868.312	0	-38150.08	-7987309.64
no		-2125778	-151249.1	-23527.91	-12313.2	0	-109304.538	-652.5062	0	-23029.22	-191091.8	-98231.85	1948630.84	-12279.56	-2868.312	0	-38150.08	-839844.901
no		-1594333	-45374.73	-15685.27	-12313.2	0	-7538.244	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	980767.84	-12279.56	-2868.312	0	-38150.08	-1062585.74
no		-2270717	-30249.82	-15685.27	-24626.39	0	-5676297.73	-652.5062	0	-23029.22	-191091.8	-98231.85	1187245.28	-12279.56	-2868.312	0	-38150.08	-7196634.32
no		-1594333	-181498.9	-23527.91	-24626.39	0	-3769.122	0	0	-23029.22	-191091.8	-98231.85	2555158.32	-12279.56	-2868.312	0	-38150.08	361751.915
no		-1690960	-75624.55	-15685.27	-18469.79	0	-870667.182	-652.5062	0	-23029.22	-191091.8	-98231.85	1793772.76	-12279.56	-2868.312	0	-38150.08	-1243936.87
no		-1352768	-75624.55	-23527.91	-18469.79	0	-1684797.53	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	2800350.28	-12279.56	-2868.312	0	-38150.08	-722945.991
no		-2029151	-45374.73	-7842.637	-18469.79	-3429.76	-7538.244	-652.5062	0	-23029.22	-191091.8	-98231.85	4903839.2	-12279.56	-2868.312	0	-38150.08	2425729.299
no		-2802162	-90749.46	-15685.27	-6156.598	0	-456063.762	-652.5062	0	-23029.22	-191091.8	-98231.85	645242	-12279.56	-2868.312	0	-38150.08	-3091877.91
no		-2077465	-151249.1	-23527.91	-12313.2	0	-2235089.35	-652.5062	0	-23029.22	-191091.8	-98231.85	709766.2	-12279.56	-2868.312	0	-38150.08	-4156181.22
no		-1980838	-15124.91	-7842.637	-12313.2	0	-1017662.94	-652.5062	0	-23029.22	-191091.8	-98231.85	2864874.48	-12279.56	-2868.312	0	-38150.08	-535210.809
no		-1401081	-15124.91	-23527.91	-12313.2	0	-1469957.58	-652.5062	0	-23029.22	-191091.8	-98231.85	1767963.08	-12279.56	-2868.312	0	-38150.08	-1520344.56
no		-2560596	-151249.1	-15685.27	-12313.2	0	-22614.732	-652.5062	0	-23029.22	-191091.8	-98231.85	6671802.28	-12279.56	-2868.312	0	-38150.08	3543040.812
no		-2802162	-151249.1	-15685.27	-24626.39	0	-267607.662	-652.5062	0	-23029.22	-191091.8	-98231.85	916243.64	-12279.56	-2868.312	0	-38150.08	-2711389.6
no		-2753848	-120999.3	-15685.27	-12313.2	0	-610597.764	-652.5062	0	-23029.22	-191091.8	-98231.85	2245442.16	-12279.56	-2868.312	0	-38150.08	-1634305.04
no		-2463970	-90749.46	-15685.27	-6156.598	0	-863128.938	-652.5062	0	-23029.22	-191091.8	-98231.85	4555408.52	-12279.56	-2868.312	0	-38150.08	749415.3438
no		-2174091	-15124.91	-23527.91	-24626.39	0	-48998.586	-652.5062	0	-23029.22	-191091.8	-98231.85	1264674.32	-12279.56	-2868.312	0	-38150.08	-1387997.61
no		-2753848	-30249.82	-15685.27	-6156.598	0	-195994.344	-652.5062	0	-23029.22	-191091.8	-98231.85	490383.92	-12279.56	-2868.312	0	-38150.08	-2877853.8
no		-2898788	-90749.46	-15685.27	-6156.598	0	-226147.32	-652.5062	0	-23029.22	-191091.8	-98231.85	2826159.96	-12279.56	-2868.312	0	-38150.08	-777669.768
no		-1594333	-120999.3	-15685.27	-12313.2	0	0	-652.5062	0	-23029.22	-191091.8	-98231.85	696861.36	-12279.56	-2868.312	0	-38150.08	-1412772.96
no		-1352768	-30249.82	-15685.27	-12313.2	0	-2725075.21	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	3381068.08	-12279.56	-2868.312	0	-38150.08	-1123131.9
no		-2705535	-75624.55	-15685.27	-18469.79	0	-2936146.04	-652.5062	0	-23029.22	-191091.8	-98231.85	2116393.76	-12279.56	-2868.312	0	-38150.08	-4001370.45
no		-1546020	-30249.82	-23527.91	-6156.598	0	-86689.806	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	2064774.4	-12279.56	-2868.312	0	-38150.08	4021.2628
no		-1207828	-120999.3	-15685.27	-12313.2	0	-188456.1	-652.5062	0	-23029.22	-191091.8	-98231.85	4413455.28	-12279.56	-2868.312	0	-38150.08	2501869.904
no		-1932525	-90749.46	-15685.27	-6156.598	0	0	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	2335776.04	-12279.56	-2868.312	0	-38150.08	-77449.3342
no		-2125778	-15124.91	-15685.27	-12313.2	0	1402113.384	-652.5062	0	-23029.22	-191091.8	-98231.85	2219632.48	-12279.56	-2868.312	0	-38150.08	1086541.488
no		-1884212	-75624.55	-23527.91	-18469.79	0	-961126.11	-652.5062	0	-23029.22	-191091.8	-98231.85	3819832.64	-12279.56	-2868.312	0	-38150.08	490568.9288
no		-2512283	-45374.73	-15685.27	-12313.2	0	-425910.786	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	1638914.68	-12279.56	-2868.312	0	-38150.08	-1740760.91
no		-2222404	-75624.55	-23527.91	-12313.2	0	927204.012	-652.5062	0	-23029.22	-191091.8	-98231.85	3290734.2	-24559.12	-2868.312	0	-38150.08	1505485.739
no		-1739273	-151249.1	-23527.91	-12313.2	0	-998817.33	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	4490884.32	-12279.56	-2868.312	0	-38150.08	1197595.261
no		-2753848	-151249.1	-15685.27	-12313.2	0	-3162293.36	0	-1805.566	-23029.22	-191091.8	-98231.85	2903589	-12279.56	-2868.312	0	-38150.08	-3559256.67
no		-2367343	-75624.55	-15685.27	-18469.79	0	-1424728.12	-652.5062	0	-23029.22	-191091.8	-98231.85	2968113.2	-12279.56	-2868.312	0	-38150.08	-1300041.18
no		-2898788	-15124.91	-15685.27	-12313.2	0	-146995.758	-652.5062	-1805.566	-23029.22	-191091.8	-98231.85	2684206.72	-12279.56	-2868.312	0	-38150.08	-772809.06
no		-2850475	-30249.82	-15685.27	-12313.2	0	0	-652.5062	0	-23029.22	-191091.8	-98231.85	2916493.84	-12279.56	-2868.312	0	-38150.08	-358532.396

Fig 15: Perceptron outcome excel analysis using main dataset containing 45212 observations proves that perceptrons fails for this dataset.

```
> bankSubset2D <-
bank[,c("age","job1","marital1","education1","default1","balance","housing1","loan1","contact1","day","month1","duration","campaign",
"pdays","previous","poutcome1","y")]
> bankSubset2D$class <- lapply(bankSubset2D$y, function(x) {
+  if(x == "no")
+  bankSubset2D$class <- -1
+  else if(x == "yes")
+  bankSubset2D$class <- 1
+  else
+  bankSubset2D$class <- NULL
+ })
> X<-
bankSubset2D[,c("age","job1","marital1","education1","balance","housing1","default1","loan1","contact1","day","month1","duration","ca
mpaign","pdays","previous","poutcome1")] # Input Matrix
> y <- bankSubset2D$class # Output Vector
```

```
> perceptron <- function(X, y, numEpochs) {
+  results <- list()
+  w <- runif(ncol(X), -10, 10) #Initalize weights
+
+  # For loop - number of generations(epochs) - number of times dataset is ran through
+  for(j in 1:numEpochs) {
+  predictedResult <- numeric(length=100) # Initalize predictedResult vector
+  numIncorrect = 0 # Keeps track of # of missclassified points
+
+  # For loop - loop throught dataset
+  for(i in 1:length(y)) {
+  xi = as.numeric(unlist(X[i,])) # Convert dataframe to vector
+  predictedResult[i] = sign(w %*% xi) # Predict the point
+
+  # If predicted point is incorrect - change weight
+  if(predictedResult[i] != y[i]) {
+  numIncorrect = numIncorrect + 1 # Add one to # of missclassified points
+  w <- w + as.numeric(y[i]) * xi # Update the weight w <- w + WiXi
+  }
+  }
+  # Print results of this generation(epoch)
+  cat("\nEpoch #: ", j)
+  cat("\nNumber Incorrect: ", numIncorrect)
+  cat("\nFinal Weight: ", w)
+  }
+ }
> results = perceptron(X,y, 8)
```

Fig 16: Perceptron learning algorithm using sample dataset containing 4521 observations.

```
Epoch #:  2
Number Incorrect:  992
Final Weight:  -15623.76 -2238.97 -1032.489 -936.1549 -1727.269 -530.0313 -36.41268 -187.6472 -1195.986 -7555.53 -2580.939 2877.051
-1484.56 1489.555 202.7788 -1846.6
Epoch #:  3
Number Incorrect:  910
Final Weight:  -19264.76 -2771.97 -1351.489 -1190.155 -1726.269 -741.0313 -53.41268 -245.6472 -1637.986 -9390.53 -3245.939 3386.051
-1907.56 1755.555 311.7788 -2399.6
Epoch #:  4
Number Incorrect:  879
Final Weight:  -21719.76 -3220.97 -1602.489 -1374.155 -959.2685 -941.0313 -72.41268 -304.6472 -2026.986 -10897.53 -3719.939 3936.051
-2301.56 2421.555 406.7788 -2830.6
Epoch #:  5
Number Incorrect:  870
Final Weight:  -23326.76 -3506.97 -1825.489 -1537.155 -1082.269 -1129.031 -91.41268 -362.6472 -2360.986 -12153.53 -4159.939 4421.051
-2667.56 2993.555 553.7788 -3208.6
Epoch #:  6
Number Incorrect:  873
Final Weight:  -25039.76 -3859.97 -2034.489 -1702.155 -1959.269 -1331.031 -108.4127 -423.6472 -2712.986 -13367.53 -4527.939 3849.051
-3013.56 3044.555 647.7788 -3578.6
Epoch #:  7
Number Incorrect:  851
Final Weight:  -26214.76 -4129.97 -2234.489 -1847.155 -1172.269 -1502.031 -126.4127 -481.6472 -3035.986 -14249.53 -4864.939 4766.051
-3307.56 3289.555 774.7788 -3908.6
Epoch #:  8
Number Incorrect:  845
Final Weight:  -27491.76 -4370.97 -2421.489 -1963.155 -1175.269 -1674.031 -143.4127 -543.6472 -3343.986 -15206.53 -5154.939 4543.051
-3585.56 3442.555 908.7788 -4236.6
```

Fig 17: Perceptron outcome using main dataset containing 4521 observations with 845 incorrect.

y	Weights	-27491.76	-4370.97	-2421.489	-1963.155	-1175.269	-1674.031	-143.4127	-543.6472	-3343.986	-15206.53	-5154.939	4543.051	-3585.56	3442.555	908.7788	-4236.6	Sum
no		-824752.8	-48080.67	-4842.978	-1963.155	0	-2991493.397	0	0	-3343.986	-288924.1	-51549.39	358901.03	-3585.56	-3442.555	0	-16946.4	-3880023.932
no		-907228.1	-34967.76	-4842.978	-3926.31	0	-8016934.459	-143.4127	-543.6472	-3343.986	-167271.8	-25774.7	999471.22	-3585.56	1167026.1	3635.1152	-4236.6	-7002666.838
no		-962211.6	-21854.85	-7264.467	-5889.465	0	-2259941.85	-143.4127	0	-3343.986	-243304.5	-20619.76	840464.44	-3585.56	1136043.2	908.7788	-4236.6	-1554979.663
no		-824752.8	-21854.85	-4842.978	-5889.465	0	-2470869.756	-143.4127	-543.6472	-10031.96	-45619.59	-30929.63	904067.15	-14342.24	-3442.555	0	-16946.4	-2546142.137
no		-1622014	-8741.94	-4842.978	-3926.31	0	0	-143.4127	0	-10031.96	-76032.65	-25774.7	1026729.5	-3585.56	-3442.555	0	-16946.4	-748752.7727
no		-962211.6	-21854.85	-7264.467	-5889.465	0	-1250501.157	0	0	-3343.986	-349750.2	-10309.88	640570.19	-7171.12	605889.68	2726.3364	-4236.6	-1373347.106
no		-989703.4	-30596.79	-4842.978	-5889.465	0	-513927.517	-143.4127	0	-3343.986	-212891.4	-25774.7	1549180.4	-3585.56	1136043.2	1817.5576	-8473.2	**887868.7149**
no		-1072179	-43709.7	-4842.978	-3926.31	0	-246082.557	-143.4127	0	-3343.986	-91239.18	-25774.7	686000.7	-7171.12	-3442.555	0	-16946.4	-832800.8327
no		-1127162	-13112.91	-4842.978	-5889.465	0	-369960.851	-143.4127	0	-10031.96	-212891.4	-25774.7	258953.91	-7171.12	-3442.555	0	-16946.4	-1538416.018
no		-1182146	-34967.76	-4842.978	-1963.155	0	147314.728	-143.4127	-543.6472	-3343.986	-258511	-20619.76	1421975	-3585.56	506055.59	1817.5576	-4236.6	**562259.2887**
no		-1072179	-34967.76	-4842.978	-3926.31	0	-15692366.59	-143.4127	0	-10031.96	-304130.6	-25774.7	1240252.9	-3585.56	-3442.555	0	-16946.4	-15932084.54
no		-1182146	-4370.97	-4842.978	-3926.31	0	-441944.184	-143.4127	0	-3343.986	-258511	-20619.76	513364.76	-7171.12	-3442.555	0	-16946.4	-1434043.599
no		-989703.4	-43709.7	-4842.978	-5889.465	0	-1856500.379	0	0	-3343.986	-197684.9	-41239.51	1490120.7	-7171.12	-3442.555	0	-16946.4	-1680353.617
yes		-549835.2	-39338.73	-7264.467	-3926.31	0	-840363.562	0	0	-3343.986	-456195.9	-20619.76	1185736.3	-3585.56	-3442.555	0	-16946.4	-759126.115
no		-852244.6	-8741.94	-4842.978	-3926.31	0	-602651.16	-143.4127	-543.6472	-3343.986	-440989.4	-5154.939	404331.54	-3585.56	829655.76	908.7788	-4236.6	-695508.3901
no		-1099670	-21854.85	-4842.978	-5889.465	0	-324762.014	0	-543.6472	-3343.986	-440989.4	-41239.51	858636.64	-7171.12	-3442.555	0	-16946.4	-1112059.658
no		-1539539	-43709.7	-4842.978	-3926.31	0	-6818328.263	0	0	-3343.986	-410576.3	-41239.51	1085789.2	-17927.8	-3442.555	0	-16946.4	-7818033.185
no		-1017195	-4370.97	-7264.467	-5889.465	0	-3878729.827	-143.4127	0	-3343.986	-304130.6	-20619.76	517907.81	-3585.56	523268.36	1817.5576	-4236.6	-4206516.032
no		-687294	-8741.94	-7264.467	-1963.155	0	369960.851	-143.4127	0	-10031.96	-349750.2	-25774.7	1135762.8	-3585.56	-3442.555	0	-16946.4	**390785.2683**
no		-852244.6	-34967.76	-4842.978	-3926.31	0	-220972.092	0	0	-3343.986	-106445.7	-36084.57	672371.55	-3585.56	523268.36	908.7788	-8473.2	-78338.0422
no		-1044687	-21854.85	-2421.489	-7852.62	0	0	-143.4127	0	-3343.986	-273717.5	-56704.33	436132.9	-7171.12	-3442.555	0	-16946.4	-1002152.286
no		-1154654	-21854.85	-2421.489	-5889.465	0	-26784.496	0	0	-3343.986	-288924.1	-56704.33	636027.14	-10756.68	-3442.555	0	-16946.4	-955695.1
no		-1209637	-34967.76	-7264.467	-3926.31	0	-177447.286	0	0	-10031.96	-182478.4	-30929.63	495192.56	-7171.12	-3442.555	0	-16946.4	-1189050.731
no		-1209637	-13112.91	-4842.978	-3926.31	0	-155684.883	0	0	-3343.986	-106445.7	-36084.57	567881.38	-7171.12	-3442.555	0	-16946.4	-992757.49
no		-714785.8	-17483.88	-4842.978	-5889.465	0	-908998.833	0	0	-3343.986	-456195.9	-5154.939	767775.62	-10756.68	-3442.555	0	-16946.4	-1380065.757
no		-1127162	-21854.85	-4842.978	-5889.465	0	-9848324.373	0	0	-3343.986	-304130.6	-56704.33	826835.28	-7171.12	-3442.555	0	-16946.4	-10572977.53
no		-1512047	-8741.94	-4842.978	-1963.155	0	-1049617.437	-143.4127	0	-10031.96	-76032.65	-25774.7	1122133.6	-3585.56	-3442.555	0	-16946.4	-1591035.944
no		-1841948	-26225.82	-4842.978	-7852.62	0	-1165125.576	0	0	-6687.972	-258511	-41239.51	540623.07	-3585.56	361468.28	1817.5576	-4236.6	-2456346.666
no		-1539539	-30596.79	-4842.978	-3926.31	0	-1312440.304	0	-543.6472	-3343.986	-456195.9	-36084.57	676914.6	-7171.12	-3442.555	0	-16946.4	-2738158.524
no		-1457063	-4370.97	-4842.978	-3926.31	0	-175773.255	0	-543.6472	-3343.986	-319337.1	-41239.51	336185.77	-7171.12	-3442.555	0	-16946.4	-1701815.369
yes		-1869440	-26225.82	-2421.489	-3926.31	0	-7012515.859	0	0	-6687.972	-212891.4	-36084.57	4075116.7	-7171.12	-3442.555	0	-16946.4	-5122636.451

Fig 18: Perceptron outcome excel analysis using main dataset containing 4521 observations proves that perceptrons fails for this dataset.

Support Vector Machine (SVM)

Support Vector Machines (SVM), a fast and dependable classification algorithm that performs very well with a limited amount of data to analyze. Support Vector Machines are not limited to straight lines or planes like Perceptrons. They can develop convex shapes around the data.

Steps -

1. Using the main dataset and creating the SVM learning algorithm.
2. Using y as dependent variable choosing random independent variables to test.
3. Making conclusion.

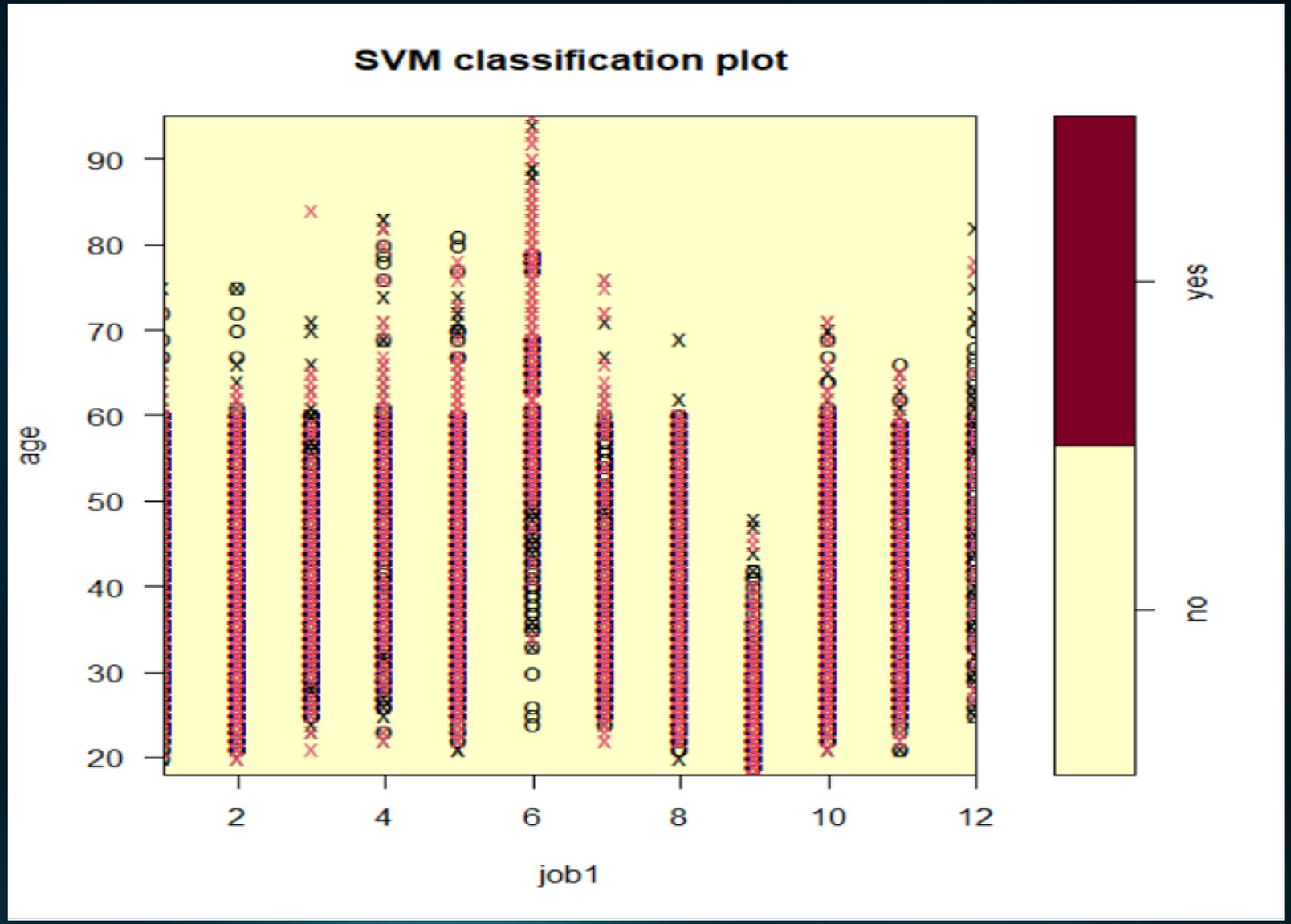


Fig 19: SVM 1 - **y~ age+job1**

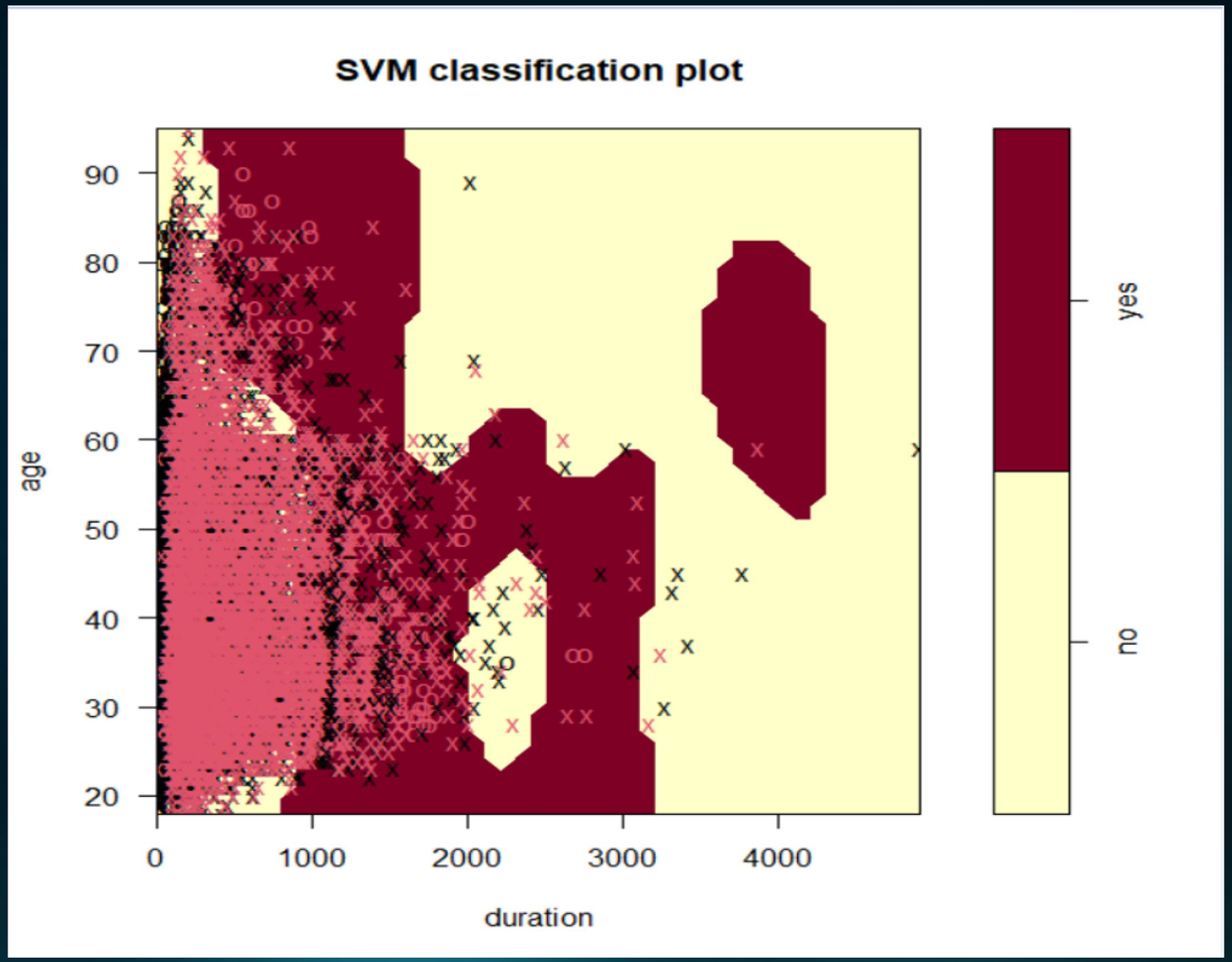


Fig 20: SVM 2 - **y~ age+duration**

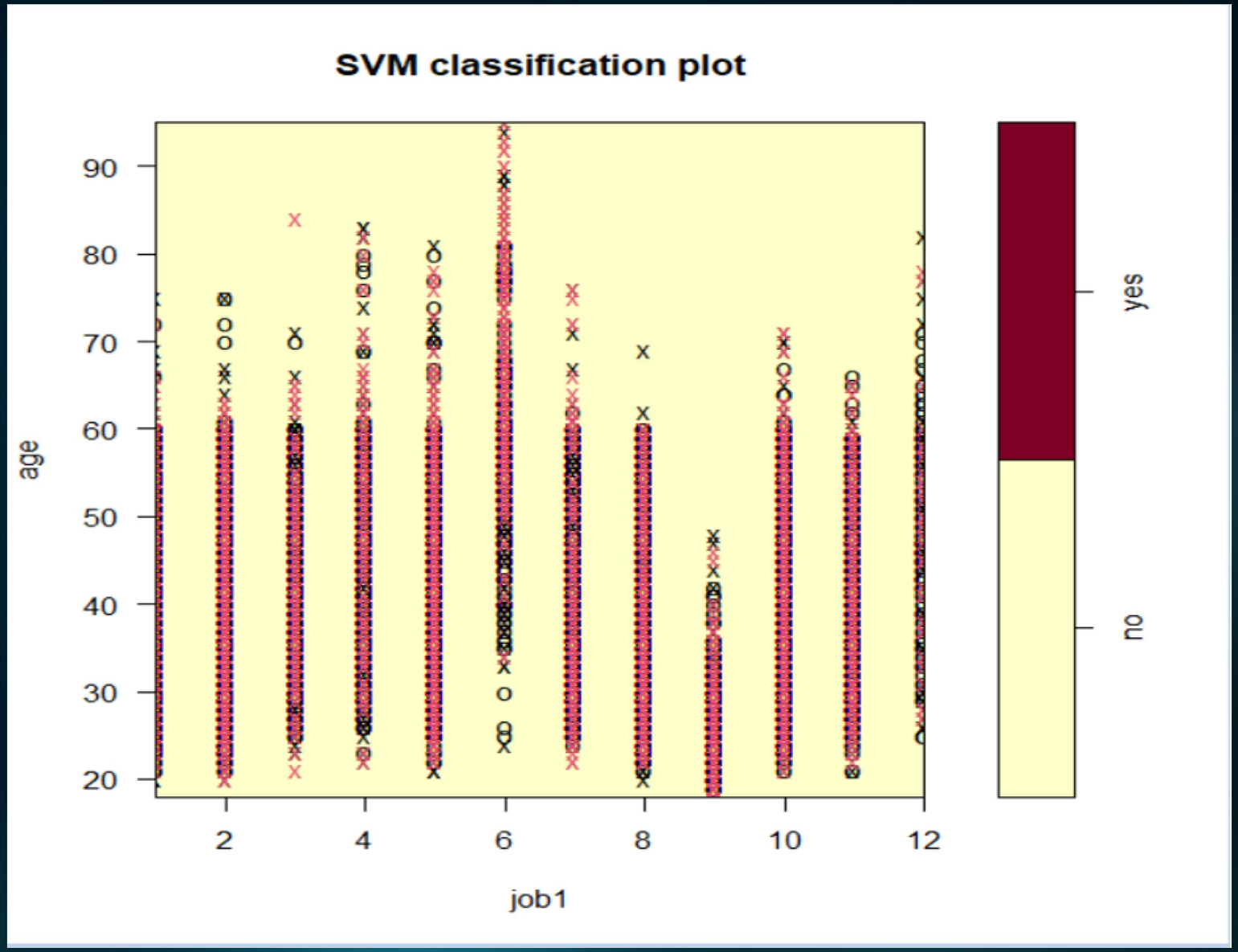


Fig 21: SVM 3 - **"y" , "age", "job1", "marital1"**

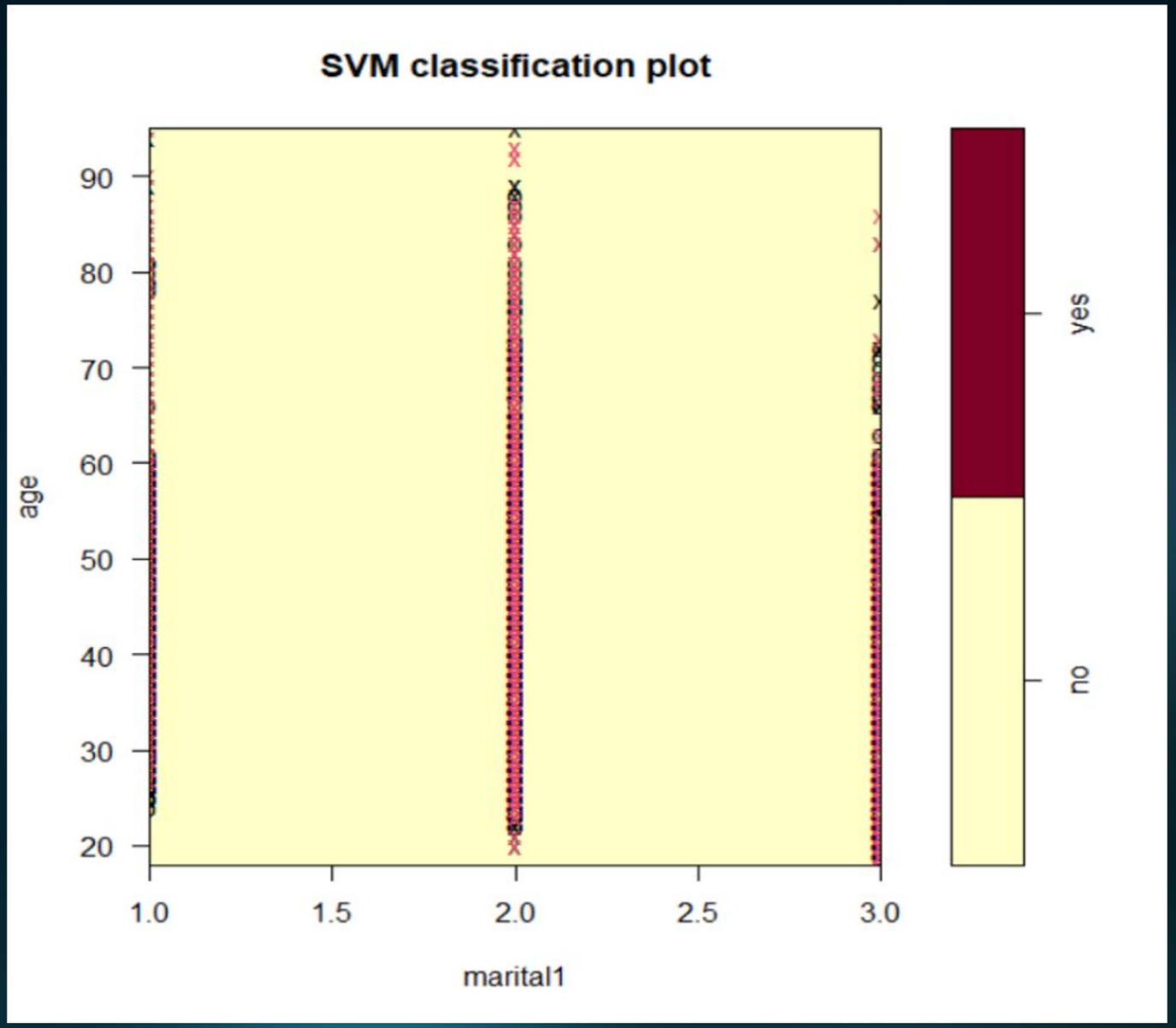


Fig 22: SVM 4 - **age ~ marital1,slice = list(job1=2,education1=2)**

Neural Networks

Neural networks are a series of algorithms that mimic the operations of an animal brain to recognize relationships between vast amounts of data. Neural networks with several process layers are known as "deep" networks and are used for deep learning algorithms.

Steps -

1. Using the main dataset and creating the neural network learning algorithm.
2. Using the sample dataset and creating the NN learning algorithm.
3. Interpreting how many hidden layers does the job.

```
> trainingnorm <- as.data.frame(lapply(banking_train,normalize))
> testingnorm <- as.data.frame(lapply(banking_test,normalize))
> bankingnet <- neuralnet(y1~ age +job1+marital1+education1+default1+balance+housing1+loan
hidden: 2    thresh: 0.01    rep: 1/1    steps: stepmax min thresh: 0.0164970084036887
Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

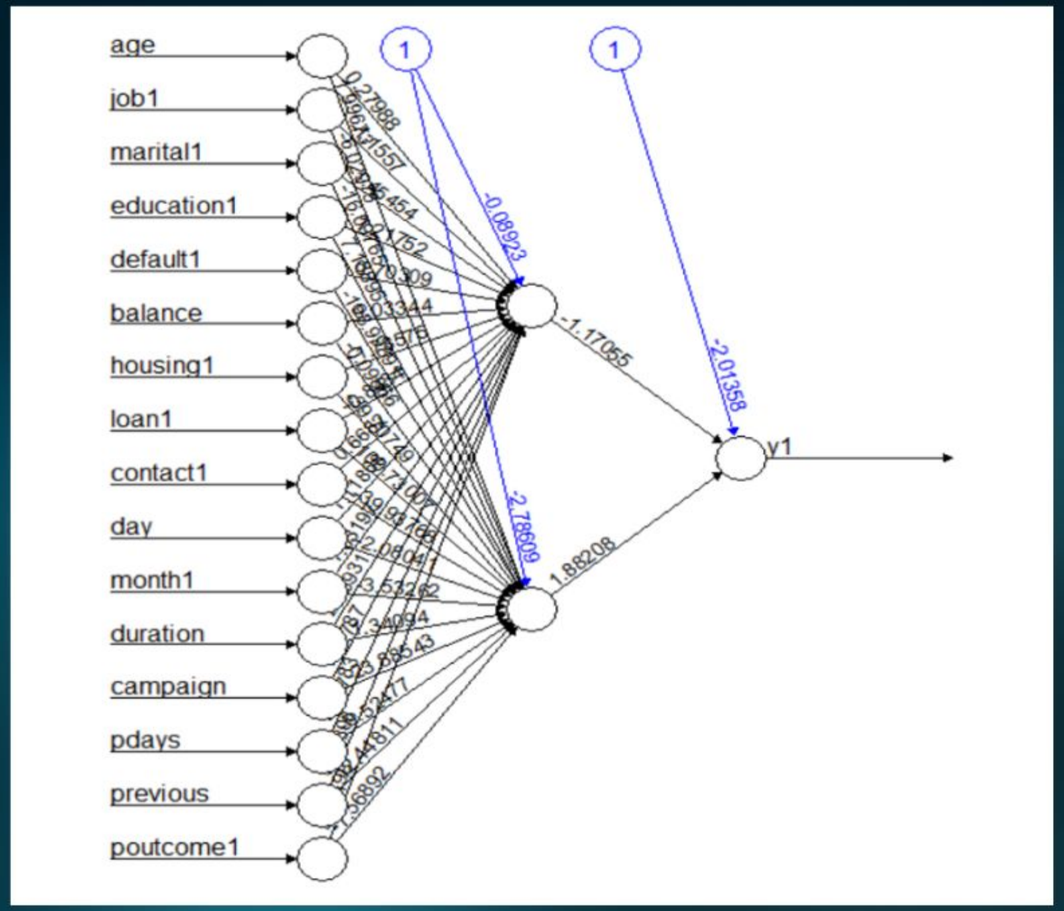


Fig 23: NN fails for the main dataset, but it works for the sample dataset.

```
> cor(bankingnet.results$net.result, banking_test$y1)
             [,1]
[1,] 0.3769068
```

```
hidden: 2    thresh: 0.01    rep: 1/1    steps:    2200 error: 100.74393        time: 1.27 secs
> bankingnet$result.matrix
                                  [,1]
error                      100.743931457
reached.threshold            0.009903411
steps                     2200.000000000
Intercept.to.1layhid1       -0.089228669
age.to.1layhid1              0.279881864
job1.to.1layhid1            -1.155695178
marital1.to.1layhid1         1.454535339
education1.to.1layhid1       1.217519255
default1.to.1layhid1         0.703091004
balance.to.1layhid1          0.033441285
housing1.to.1layhid1        -1.785748013
loan1.to.1layhid1           -0.843395221
contact1.to.1layhid1         0.668197668
day.to.1layhid1              1.118682817
month1.to.1layhid1           1.031965233
duration.to.1layhid1         2.269312326
campaign.to.1layhid1         1.547869843
pdays.to.1layhid1            0.647825258
previous.to.1layhid1         0.383064333
poutcome1.to.1layhid1        2.604975960
Intercept.to.1layhid2       -2.786088400
age.to.1layhid2              1.996725207
job1.to.1layhid2            -6.029745010
marital1.to.1layhid2       -16.097649793
education1.to.1layhid2       7.188945851
default1.to.1layhid2      -198.998912366
balance.to.1layhid2         -0.098662677
housing1.to.1layhid2       -39.707489402
loan1.to.1layhid2         -100.730070232
contact1.to.1layhid2       -39.937681652
day.to.1layhid2             -2.080411137
month1.to.1layhid2          -3.532624567
duration.to.1layhid2         1.340941555
campaign.to.1layhid2       -23.885427270
pdays.to.1layhid2           -0.524773438
previous.to.1layhid2        12.448107824
poutcome1.to.1layhid2       -7.568922913
Intercept.to.y1             -2.013575077
1layhid1.to.y1              -1.170554341
1layhid2.to.y1               1.882084365
```

Fig 24: NN fails for the main dataset, but it works for the sample dataset. Prediction on 50% testing data with a correlation of 0.37

k-Nearest Neighbors

KNN is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

Steps -

1. k-nearest neighbor code considering all 16 variables using main dataset.
2. Using different values of k to determine the highest accuracy of results.

```
> bank$y1 <- factor(bank$y1, levels=c(1,0), labels=c("Term Deposit","No Term Deposit"))
>
> normalize <- function(x) { return ((x-min(x))/(max(x)-min(x))) }
>
> bank_norm <- as.data.frame(lapply(bank[1:16], normalize))
>
> bank_index <- sample(nrow(bank_norm), 1/2 * nrow(bank_norm))
>
> bank_train <- bank_norm[bank_index, ]
>
> bank_test <- bank_norm[-bank_index, ]
>
> bank_train_labels <- bank[bank_index,17]
>
> bank_test_labels <- bank[-bank_index,17]
>
> bank_test_pred <- knn(train = bank_train, test = bank_test, cl=bank_train_labels, k=30)
>
> CrossTable(x=bank_test_labels, y=bank_test_pred, prop.chisq=FALSE)
```

Fig 26: k-nearest neighbor code considering all 16 variables using main dataset where all variables are converted into numeric values.

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


Total Observations in Table:  22606


                 | bank_test_pred
bank_test_labels |    Term Deposit | No Term Deposit |       Row Total |
-----------------|-----------------|-----------------|-----------------|
    Term Deposit |             215 |            2395 |            2610 |
                 |           0.082 |           0.918 |           0.115 |
                 |           0.660 |           0.107 |                 |
                 |           0.010 |           0.106 |                 |
-----------------|-----------------|-----------------|-----------------|
 No Term Deposit |             111 |           19885 |           19996 |
                 |           0.006 |           0.994 |           0.885 |
                 |           0.340 |           0.893 |                 |
                 |           0.005 |           0.880 |                 |
-----------------|-----------------|-----------------|-----------------|
    Column Total |             326 |           22280 |           22606 |
                 |           0.014 |           0.986 |                 |
-----------------|-----------------|-----------------|-----------------|
```

Fig 26: Output of k-nearest neighbor depicting with k=30, we get an accuracy of 88.91%.

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


Total Observations in Table:  22606


                 | bank_test_pred
bank_test_labels |    Term Deposit | No Term Deposit |       Row Total |
-----------------|-----------------|-----------------|-----------------|
    Term Deposit |             356 |            2254 |            2610 |
                 |           0.136 |           0.864 |           0.115 |
                 |           0.637 |           0.102 |                 |
                 |           0.016 |           0.100 |                 |
-----------------|-----------------|-----------------|-----------------|
 No Term Deposit |             203 |           19793 |           19996 |
                 |           0.010 |           0.990 |           0.885 |
                 |           0.363 |           0.898 |                 |
                 |           0.009 |           0.876 |                 |
-----------------|-----------------|-----------------|-----------------|
    Column Total |             559 |           22047 |           22606 |
                 |           0.025 |           0.975 |                 |
-----------------|-----------------|-----------------|-----------------|
```

Fig 27: Output of k-nearest neighbor depicting with k=15, we get an accuracy of 89.13%.

```
> bank_test_pred <- knn(train = bank_train, test = bank_test, cl=bank_train_labels, k=10)
> CrossTable(x=bank_test_labels, y=bank_test_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  22606


                 | bank_test_pred
bank_test_labels |    Term Deposit | No Term Deposit |       Row Total |
-----------------|-----------------|-----------------|-----------------|
    Term Deposit |             469 |            2141 |            2610 |
                 |           0.180 |           0.820 |           0.115 |
                 |           0.628 |           0.098 |                 |
                 |           0.021 |           0.095 |                 |
-----------------|-----------------|-----------------|-----------------|
 No Term Deposit |             278 |           19718 |           19996 |
                 |           0.014 |           0.986 |           0.885 |
                 |           0.372 |           0.902 |                 |
                 |           0.012 |           0.872 |                 |
-----------------|-----------------|-----------------|-----------------|
    Column Total |             747 |           21859 |           22606 |
                 |           0.033 |           0.967 |                 |
-----------------|-----------------|-----------------|-----------------|
```

Fig 28: Output of k-nearest neighbor depicting with k=10, we get an accuracy of 89.3%.

Naive Bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. It is a supervised learning techniques, meaning that it requires a dependent (Y) variable. Naive Bayes model is easy to build and particularly useful for very large data sets.

Steps -

1. Naive Bayes code using R on main dataset.
2. Creating prediction table and checking the accuracy of the model.

```
> bank$y <- factor(bank$y, levels=c("no","yes"), labels=c("NO Term Deposit","TAKEN Term Deposit"))
> bank_index <- sample(nrow(bank), 1/2 * nrow(bank))
> bank_train <- bank[bank_index, ]
> bank_test <- bank[-bank_index, ]
> bank_model <- naiveBayes(y ~ ., data=bank, laplace=1)
> bank_model
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
   NO Term Deposit TAKEN Term Deposit
         0.8830152          0.1169848

Conditional probabilities:
                    age
Y                        [,1]     [,2]
  NO Term Deposit    40.83899 10.17266
  TAKEN Term Deposit 41.67007 13.49778

                    job
Y                         admin. blue-collar entrepreneur   housemaid  management     retired self-employed    services
  NO Term Deposit    0.113712626 0.225997897  0.034181399 0.028346772 0.204287074 0.043797265   0.034882556 0.094806431
  TAKEN Term Deposit 0.119222788 0.133748349  0.023391813 0.020750802 0.245614035 0.097528768   0.035465007 0.069798151
                    job
Y                        student  technician  unemployed     unknown
  NO Term Deposit    0.016777683 0.169229228 0.027595533 0.006385536
  TAKEN Term Deposit 0.050933786 0.158649311 0.038294661 0.006602528

                    marital
Y                     divorced   married    single
  NO Term Deposit    0.1148654 0.6126487 0.2724859
  TAKEN Term Deposit 0.1177249 0.5207861 0.3614890
```

```
                    education
Y                         primary  secondary   tertiary    unknown
  NO Term Deposit    0.15681511 0.51978661 0.28317387 0.04022442
  TAKEN Term Deposit 0.11184583 0.46306442 0.37729076 0.04779898

                    default
Y                            no         yes
  NO Term Deposit    0.98086364 0.01913636
  TAKEN Term Deposit 0.98998299 0.01001701

                    balance
Y                         [,1]     [,2]
  NO Term Deposit    1303.715 2974.195
  TAKEN Term Deposit 1804.268 3501.105

                    housing
Y                           no       yes
  NO Term Deposit    0.4189961 0.5810039
  TAKEN Term Deposit 0.6340956 0.3659044

                    loan
Y                            no         yes
  NO Term Deposit    0.83065324 0.16934676
  TAKEN Term Deposit 0.90833491 0.09166509

                    contact
Y                      cellular  telephone    unknown
  NO Term Deposit    0.62409518 0.06304321 0.31286162
  TAKEN Term Deposit 0.82577475 0.07388511 0.10034014

                    day
Y                         [,1]     [,2]
  NO Term Deposit    15.89229 8.294728
  TAKEN Term Deposit 15.15825 8.501875
```

Fig 29: Naive Bayes code using R on main dataset & it's result.

```
                    month
Y                            apr         aug         dec         feb         jan         jul         jun         mar
  NO Term Deposit    0.058997346 0.139229729 0.002879752 0.055316272 0.031602144 0.156984024 0.120098162 0.005759503
  TAKEN Term Deposit 0.109036031 0.129975476 0.019053009 0.083380494 0.026976042 0.118468214 0.103188078 0.046972269
                    month
Y                            may         nov         oct         sep
  NO Term Deposit    0.321580608 0.089347423 0.010417188 0.007787850
  TAKEN Term Deposit 0.174684022 0.076212035 0.061120543 0.050933786

                    duration
Y                        [,1]     [,2]
  NO Term Deposit    221.1828 207.3832
  TAKEN Term Deposit 537.2946 392.5253

                    campaign
Y                        [,1]     [,2]
  NO Term Deposit    2.846350 3.212767
  TAKEN Term Deposit 2.141047 1.921826

                    pdays
Y                        [,1]      [,2]
  NO Term Deposit    36.42137  96.75713
  TAKEN Term Deposit 68.70297 118.82227

                    previous
Y                         [,1]     [,2]
  NO Term Deposit    0.5021542 2.256771
  TAKEN Term Deposit 1.1703536 2.553272

                    poutcome
Y                       failure      other    success    unknown
  NO Term Deposit    0.10729850 0.03842108 0.01337474 0.84090568
  TAKEN Term Deposit 0.11694691 0.05819006 0.18496127 0.63990176
```

Fig 30: Naive Bayes code using R on main dataset & it's continued result.

```
> bank_pred <- predict(bank_model, bank_test, type="class")
> bank_pred_table <- table(bank_test$y, bank_pred)
> bank_pred_table
                    bank_pred
                     NO Term Deposit TAKEN Term Deposit
  NO Term Deposit              18387                1528
  TAKEN Term Deposit            1264                1427
> sum(diag(bank_pred_table))/nrow(bank_test)
[1] 0.876493
```

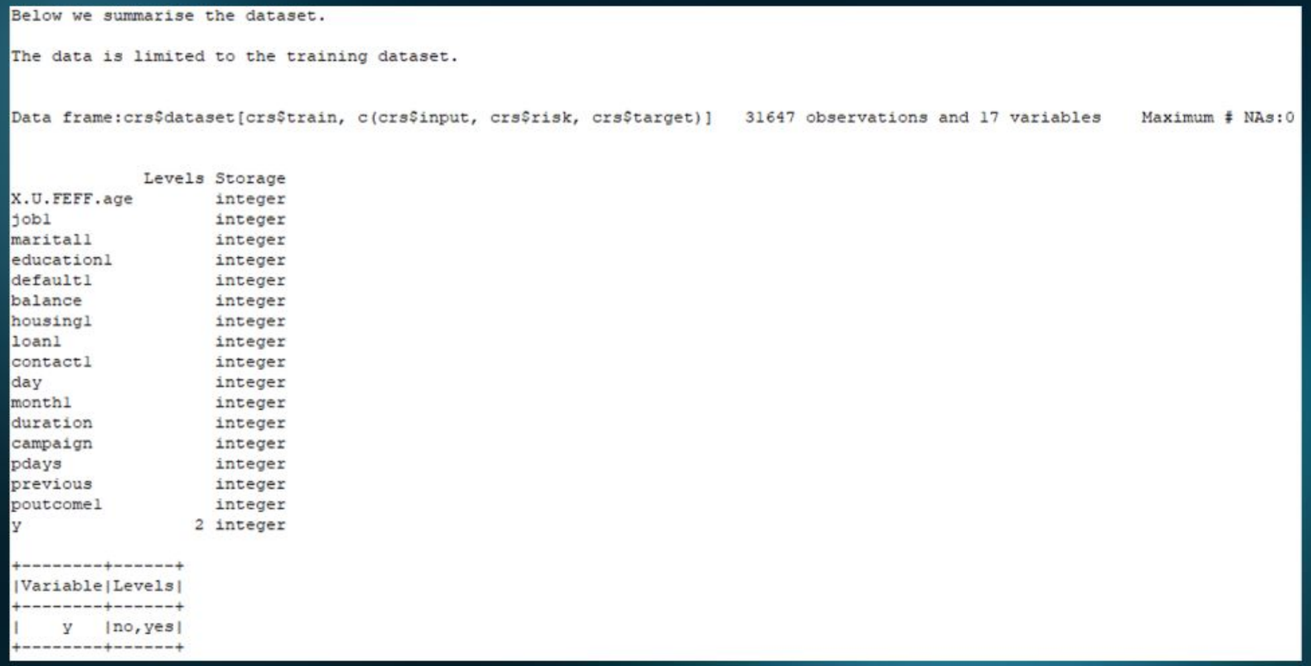
Fig 31: Naive Bayes prediction table on main dataset & accuracy of **87.64%**.

Decision Trees

A decision tree evaluates the explanatory (X) variables in a dataset and determines the most important variables in making a decision. The technique is based on information theory and the entropy calculation. Entropy is the measure of noise or randomness in a system. The decision tree selects the first variable based on which one reduces entropy the most, then selects a second decision variable, and so on, until additional variables add no value. It then builds an inverted tree with the decision structure.

Steps -

1. Using R version 4.1.0 with rattle( ) function we get the above descriptive summary. Rattle selects 70% of the data for a training dataset; 30% for testing.
2. Building & drawing decision trees.
3. Testing the decision tree model & plotting the error matrix.



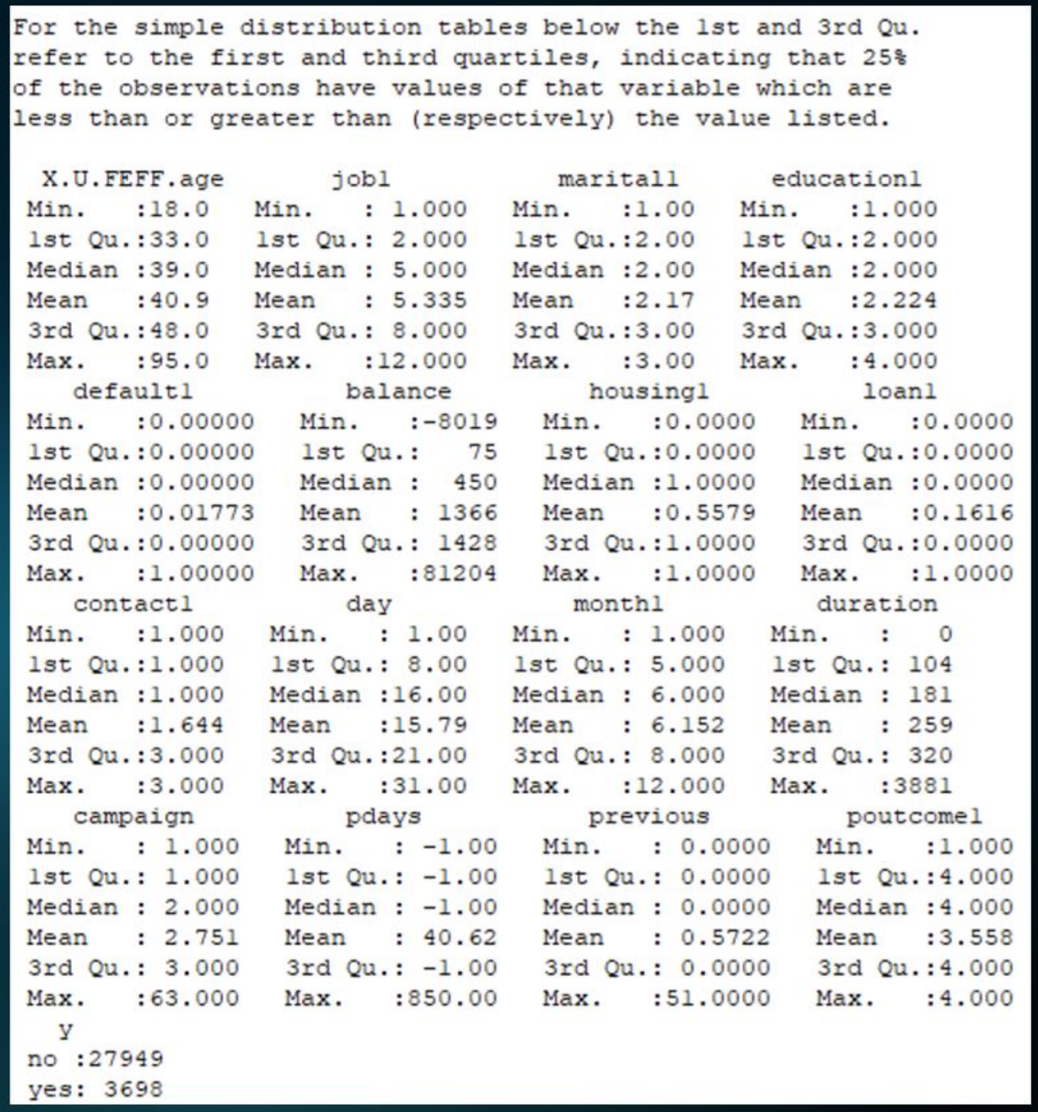
```
Below we summarise the dataset.

The data is limited to the training dataset.


Data frame:crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)]   31647 observations and 17 variables    Maximum # NAs:0


             Levels Storage
X.U.FEFF.age        integer
jobl                integer
maritall            integer
educationl          integer
defaultl            integer
balance             integer
housingl            integer
loanl               integer
contactl            integer
day                 integer
monthl              integer
duration            integer
campaign            integer
pdays               integer
previous            integer
poutcomel           integer
y                 2 integer

+---------+------+
|Variable|Levels|
+---------+------+
|    y    |no,yes|
+---------+------+
```



```
For the simple distribution tables below the 1st and 3rd Qu.
refer to the first and third quartiles, indicating that 25%
of the observations have values of that variable which are
less than or greater than (respectively) the value listed.

 X.U.FEFF.age       jobl          maritall      educationl
Min.   :18.0   Min.   : 1.000   Min.   :1.00   Min.   :1.000
1st Qu.:33.0   1st Qu.: 2.000   1st Qu.:2.00   1st Qu.:2.000
Median :39.0   Median : 5.000   Median :2.00   Median :2.000
Mean   :40.9   Mean   : 5.335   Mean   :2.17   Mean   :2.224
3rd Qu.:48.0   3rd Qu.: 8.000   3rd Qu.:3.00   3rd Qu.:3.000
Max.   :95.0   Max.   :12.000   Max.   :3.00   Max.   :4.000
   defaultl          balance         housingl          loanl
Min.   :0.00000   Min.   :-8019   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.00000   1st Qu.:   75   1st Qu.:0.0000   1st Qu.:0.0000
Median :0.00000   Median :  450   Median :1.0000   Median :0.0000
Mean   :0.01773   Mean   : 1366   Mean   :0.5579   Mean   :0.1616
3rd Qu.:0.00000   3rd Qu.: 1428   3rd Qu.:1.0000   3rd Qu.:0.0000
Max.   :1.00000   Max.   :81204   Max.   :1.0000   Max.   :1.0000
   contactl          day             monthl          duration
Min.   :1.000   Min.   : 1.00   Min.   : 1.000   Min.   :   0
1st Qu.:1.000   1st Qu.: 8.00   1st Qu.: 5.000   1st Qu.: 104
Median :1.000   Median :16.00   Median : 6.000   Median : 181
Mean   :1.644   Mean   :15.79   Mean   : 6.152   Mean   : 259
3rd Qu.:3.000   3rd Qu.:21.00   3rd Qu.: 8.000   3rd Qu.: 320
Max.   :3.000   Max.   :31.00   Max.   :12.000   Max.   :3881
   campaign          pdays          previous         poutcomel
Min.   : 1.000   Min.   : -1.00   Min.   : 0.0000   Min.   :1.000
1st Qu.: 1.000   1st Qu.: -1.00   1st Qu.: 0.0000   1st Qu.:4.000
Median : 2.000   Median : -1.00   Median : 0.0000   Median :4.000
Mean   : 2.751   Mean   : 40.62   Mean   : 0.5722   Mean   :3.558
3rd Qu.: 3.000   3rd Qu.: -1.00   3rd Qu.: 0.0000   3rd Qu.:4.000
Max.   :63.000   Max.   :850.00   Max.   :51.0000   Max.   :4.000
  y
no :27949
yes: 3698
```

Fig 32: Using R version 4.1.0 with rattle( ) function we get the above descriptive summary. Rattle selects 70% of the data for a training dataset; 30% for testing.

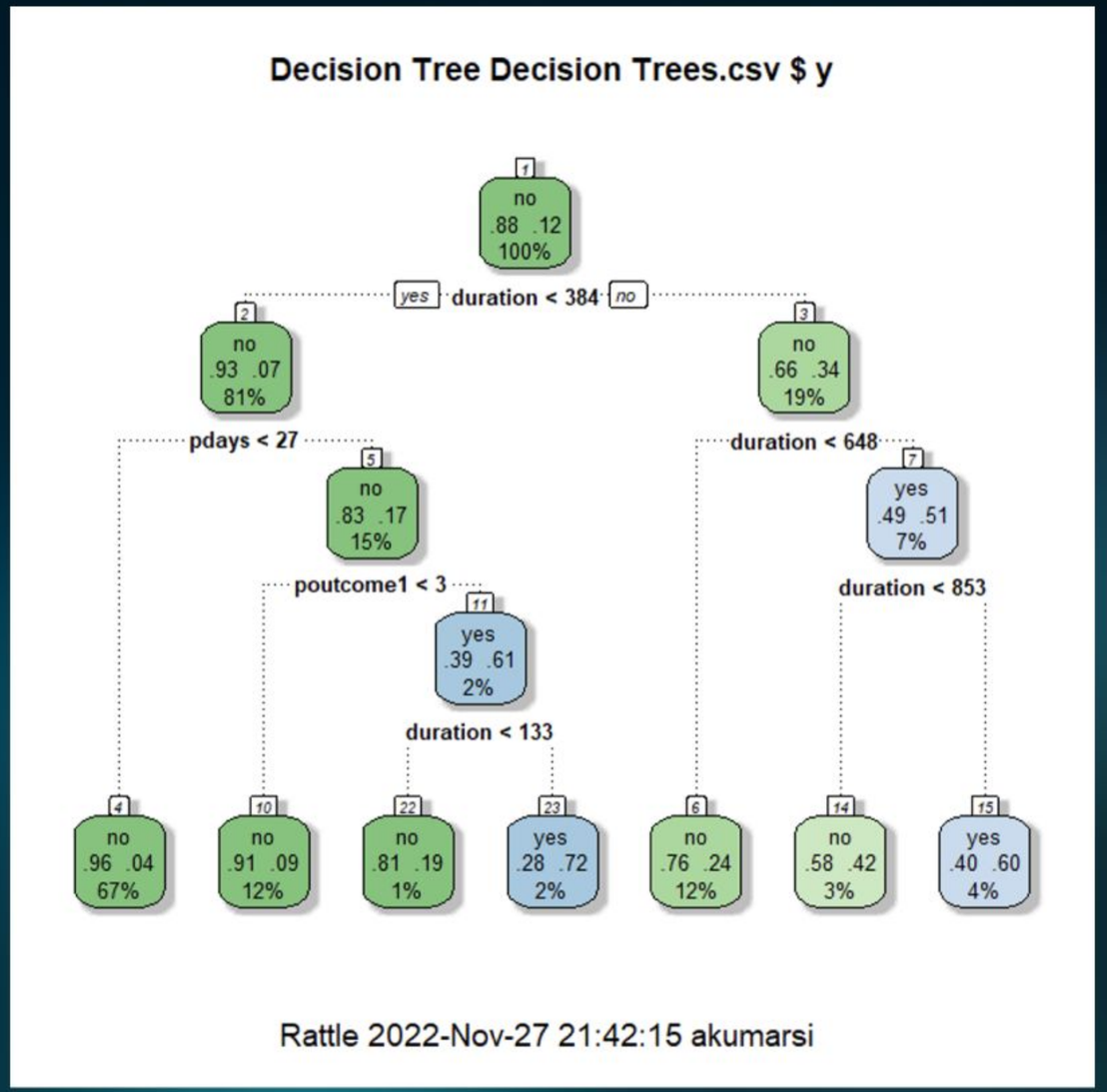


Fig 33: Building & drawing the Decision Tree. It takes the following variables into consideration - **duration, pdays & poutcome.**

```
Error matrix for the Decision Tree model on Decision Trees.csv [test] (counts):

      Predicted
Actual    no yes Error
   no   5819 162   2.7
   yes   575 227  71.7

Error matrix for the Decision Tree model on Decision Trees.csv [test] (proportions):

      Predicted
Actual    no yes Error
   no   85.8 2.4   2.7
   yes   8.5 3.3  71.7

Overall error: 10.9%, Averaged class error: 37.2%

Rattle timestamp: 2022-11-27 21:47:42 akumarsi
============================================================
```

Fig 34: Testing the decision tree model and plotting the error matrix.

Random Forests

Random Forest is a powerful and versatile supervised machine learning algorithm that grows and combines multiple decision trees to create a “forest.” The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone.

Steps -

1. Using rattle( ) function executing random forests with 500 forests and 2 variables.
2. Plotting ‘mean decrease accuracy’ using rattle ( ).
3. Plotting OOB errors and OOB-ROC using rattle( ).

```
Summary of the Random Forest Model
==================================

Number of observations used to build the model: 31647
Missing value imputation is active.

Call:
 randomForest(formula = y ~ ., 
              data = crs$dataset[crs$train, c(crs$input, crs$target)], 
              ntree = 500, mtry = 2, importance = TRUE, replace = FALSE, na.action = randomForest::na.roughfix)

               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 10.07%
Confusion matrix:
       no yes class.error
no  27628 321  0.01148521
yes  2866 832  0.77501352

Analysis of the Area Under the Curve (AUC)
==========================================

Call:
roc.default(response = crs$rf$y, predictor = as.numeric(crs$rf$predicted))

Data: as.numeric(crs$rf$predicted) in 27949 controls (crs$rf$y no) < 3698 cases (crs$rf$y yes).
Area under the curve: 0.6068

95% CI: 0.6-0.6135 (DeLong)
```

```
Variable Importance
===================

                no   yes MeanDecreaseAccuracy MeanDecreaseGini
duration     65.52 86.06                81.06            779.87
month1       47.66  9.88                48.13            204.93
poutcome1    40.59  7.17                45.87            188.41
day          40.10  4.88                41.06            199.96
housing1     29.62 20.05                37.12             83.35
X.U.FEFF.age 29.54 24.00                36.09            231.10
contact1     31.92 14.88                35.89             65.57
pdays        25.06 19.22                27.63            169.62
campaign     11.88 14.53                19.21             87.10
previous     18.07 15.76                19.02             88.08
job1          9.85  6.82                13.30            104.45
education1    7.71  8.28                12.82             60.10
loan1        -2.06 19.45                11.61             27.46
marital1      1.67 13.51                10.30             52.83
balance       1.65  9.24                 7.65            211.08
default1      1.70  5.90                 4.62              5.14

Time taken: 1.17 mins

Rattle timestamp: 2022-11-27 21:51:16 akumarsi
================================================================================
```

Fig 35: Executing random forests with 500 forests and 2 variables

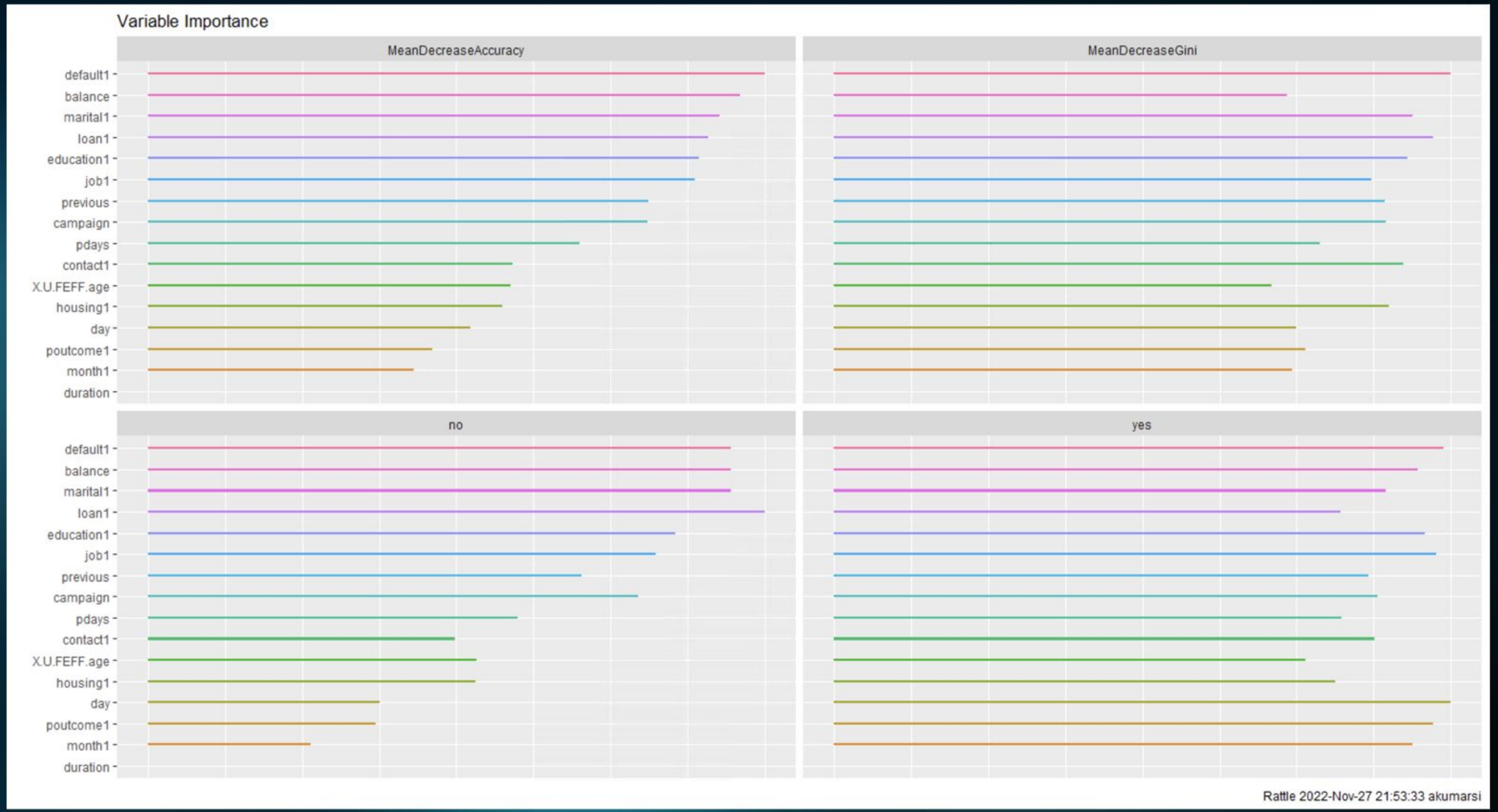


Fig 36: "Mean decrease accuracy" measures how much accuracy decreases if you exclude this variable. This infers that **deafult, balance, marital, loan & education** are the top 5 variables to predict if the client takes a term deposit or not.

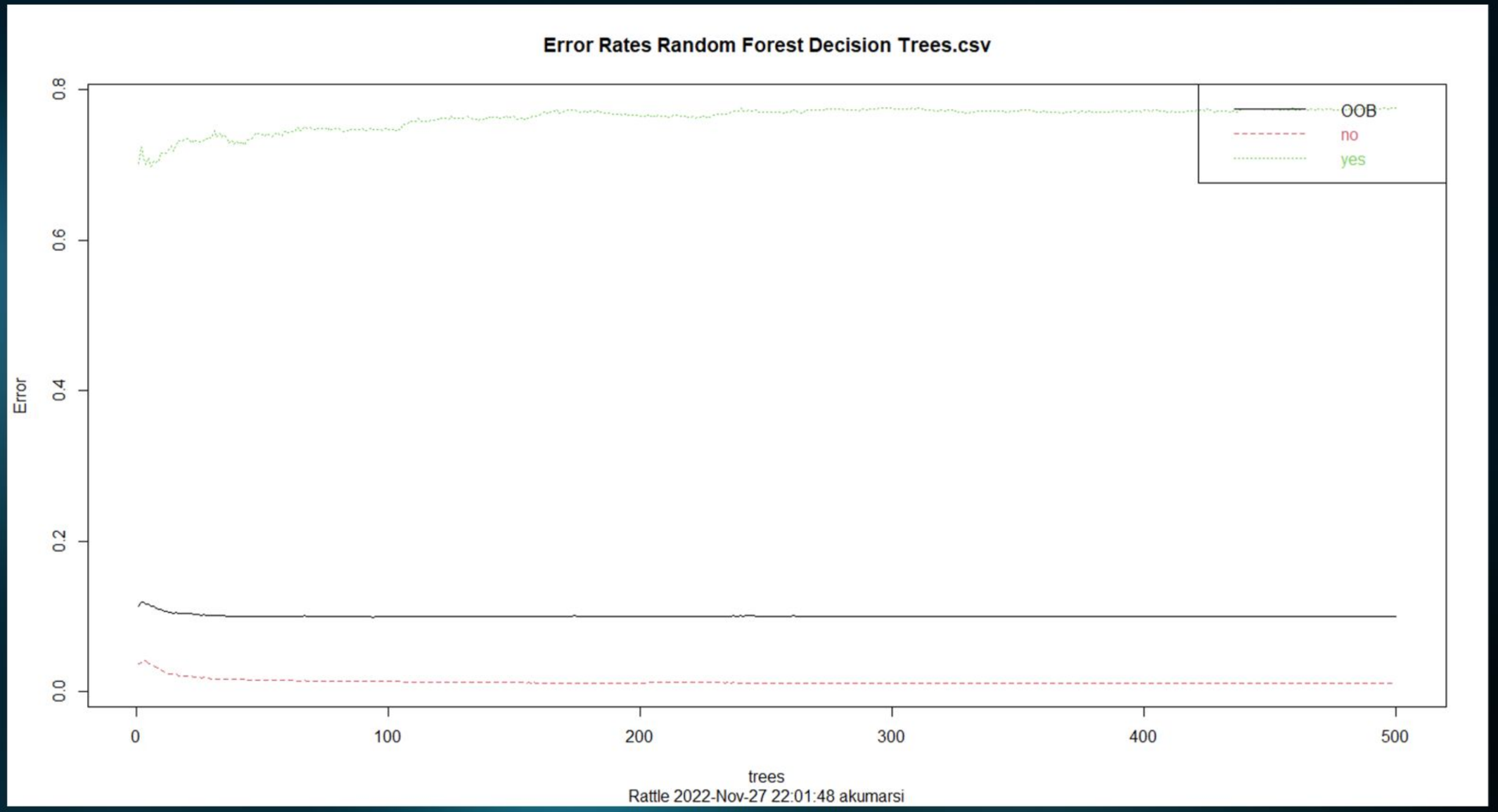


Fig 37: Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests.

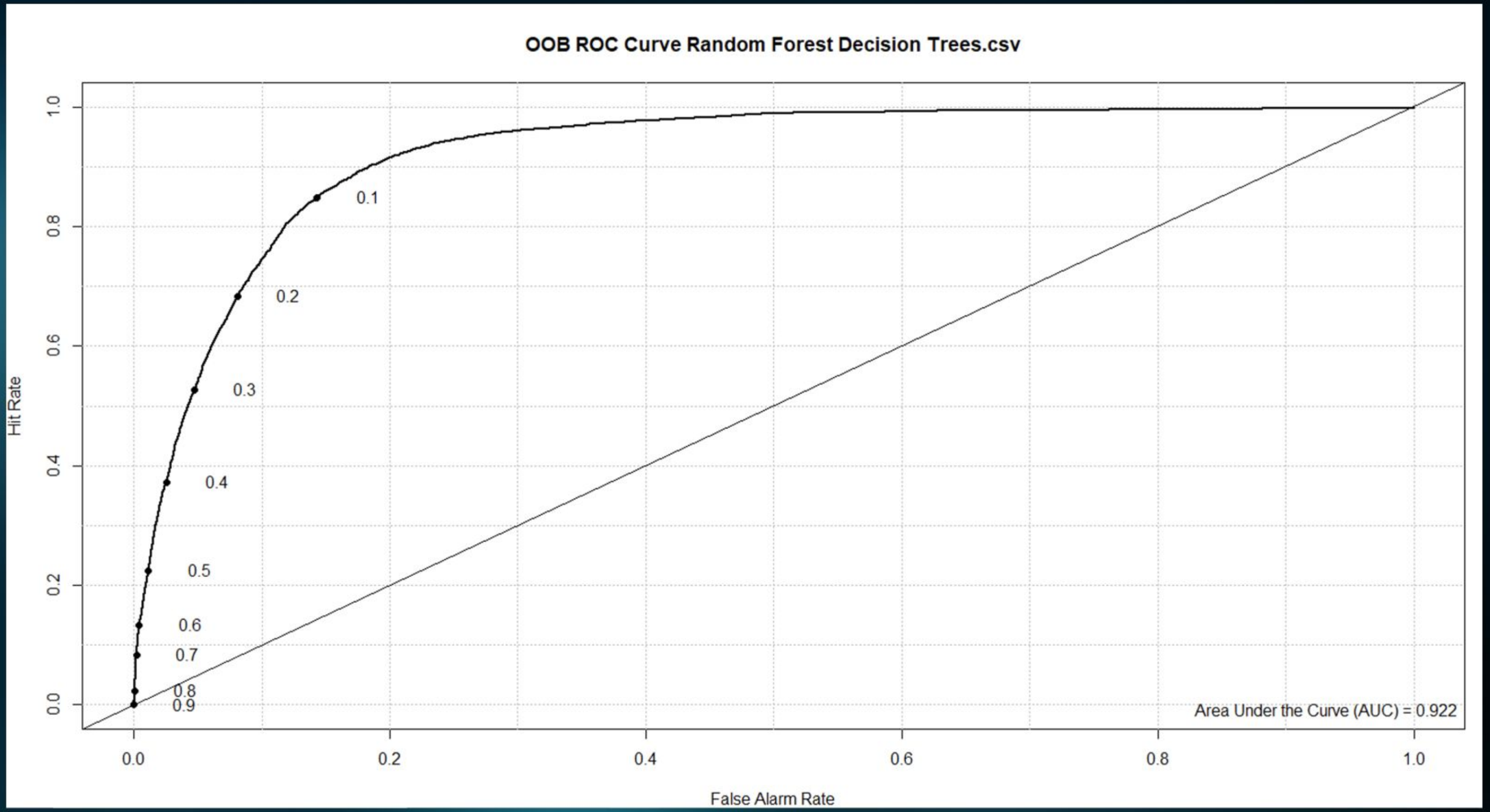


Fig 38: Out-of-bag (OOB) ROC

Conclusion

The main dataset has **11.7%** success campaign & sample dataset has **11.5%** success campaign.

- **Logit** – Helped to identify variables which were statistically significant.
- Except **Job** and **month**, all the variables in the logit analysis of the dataset with dummy variables were statistically significant i.e. p value is less than 0.05
- **Perceptrons** -Didn't work.
- **SVM** – Was able to classify the dataset.
- **Neural Networks** worked on the sample dataset but not on the complete dataset even after normalization.
- **Neural networks** with one hidden layer on the sample dataset worked.

Conclusion

- **Naïve Bayes** – Prediction accuracy was **87%** on average
- **K-Nearest Neighbour** -Prediction accuracy was **88%** on average
- **Decision Trees** & **Random forests**- worked well with main dataset and variables **default, balance, marital, loan & education** affected the accuracy of the campaign the most.
- If the **Last Contact duration** is more than **384 seconds** then the client will not subscribe for term deposit else he will
- If **pdays** i.e. number of days that passed by after the client was last contacted from a previous campaign is less than **27** days and **poutcome** ie. outcome of the previous marketing campaign (categorical: "unknown","other","failure","success") is less than **3** i.e. it was either failure or other then the client will end up with the subscription.

Conclusion

• As per the decision tree , **duration ,pdays and poutcome** are the most important variables which helped in predicting whether client will subscribe for term deposit or not

• If the last contact duration is not less than 384 seconds ie.6 minutes 24 seconds then 66% of the time the client will not subscribe for a term deposit and 34% of the time ,the client will subscribe for a term deposit

• If the last contact duration is less than 648 seconds, i.e. 10 min 48 sec then 76% of the times ,the client will not subscribe for a term deposit and only 24% of the times the client will subscribe for a term deposit

• If the last contact duration is not less than 648 seconds, i.e. 10 min 48 sec then 49% of the times ,the client will not subscribe for a term deposit and 51% of the times the client will subscribe for a term deposit

• If the last contact duration is less than 853 seconds, i.e. 14 min 13 sec then 58% of the times ,the client will not subscribe for a term deposit and 42% of the times the client will subscribe for a term deposit

• If the last contact duration is not less than 853 seconds, i.e. 14 min 13 sec then 40% of the times ,the client will not subscribe for a term deposit and 60% of the times the client will subscribe for a term deposit

Insights for the Bank

The bank should target the following customer's data having the following attributes -

- **Higher education** of customers will affect the perception of the term deposit offered by the campaign.
- **Average yearly balance** (in Euro) of customers will affect the decision whether the client will take the term deposit ot not.
- Clients that have taken a **loan** & have **defaulted** in the past are less likely to invest in a term deposit.
- Overall, the campaign should focus on increasing the **last contact duration (seconds)** as the client is more likely to subscribe for the term deposit.

THANK YOU!

AKASH KUMAR SINGH
AKSHATA RAVINDER
NEHAL TAYA
ELMIR GULUYEV