

Project 4

Dipesh Poudel

12/28/2021

```
library(haven)
bank_loan_df <- read_sav("P4_bankloan_5000_clients.sav")

bank_loan_df$defaulted_loan<-as.factor(bank_loan_df$defaulted_loan)
bank_loan_df$education_level<-as.factor(bank_loan_df$education_level)

str(bank_loan_df)

## tibble [5,000 x 9] (S3: tbl_df/tbl/data.frame)
##  $ age                : num [1:5000] 41 30 40 41 57 45 36 39 43 34 ...
##  ..- attr(*, "label")= chr "Age in years"
##  ..- attr(*, "format.spss")= chr "F4.0"
##  ..- attr(*, "display_width")= int 6
##  $ education_level     : Factor w/ 5 levels "1","2","3","4",...: 3 1 1 1 1 1 1 1 1 3 ...
##  $ current_employ_year : num [1:5000] 17 13 15 15 7 0 1 20 12 7 ...
##  ..- attr(*, "label")= chr "Years with current employer"
##  ..- attr(*, "format.spss")= chr "F4.0"
##  $ current_address_year: num [1:5000] 12 8 14 14 37 13 3 9 11 12 ...
##  ..- attr(*, "label")= chr "Years at current address"
##  ..- attr(*, "format.spss")= chr "F4.0"
##  ..- attr(*, "display_width")= int 9
##  $ income_household    : num [1:5000] 35.9 46.7 61.8 72 25.6 28.1 19.6 80.5 68.7 33.8 ...
##  ..- attr(*, "label")= chr "Household income in thousands"
##  ..- attr(*, "format.spss")= chr "F8.2"
##  ..- attr(*, "display_width")= int 10
##  $ debt_income_ratio   : num [1:5000] 11.9 17.9 10.6 29.7 15.9 ...
##  ..- attr(*, "label")= chr "Debt to income ratio (x100)"
##  ..- attr(*, "format.spss")= chr "F8.2"
##  ..- attr(*, "display_width")= int 10
##  $ credit_card_debt    : num [1:5000] 0.504 1.353 3.439 4.166 1.498 ...
##  ..- attr(*, "label")= chr "Credit card debt in thousands"
##  ..- attr(*, "format.spss")= chr "F8.2"
##  ..- attr(*, "display_width")= int 10
##  $ other_debts         : num [1:5000] 3.77 7 3.14 17.2 2.56 ...
##  ..- attr(*, "label")= chr "Other debt in thousands"
##  ..- attr(*, "format.spss")= chr "F8.2"
##  ..- attr(*, "display_width")= int 10
##  $ defaulted_loan      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  - attr(*, "label")= chr "Bank Loan Default -- Binning"
##  - attr(*, "notes")= chr [1:7] "DOCUMENT This is a hypothetical data file that concerns a bank's eff
```

Use the attached SPSS data containing 5000 cases of bank clients and fit all the supervised classifiers models as described below. Copy all the outputs you will get at/from R/R Studio here and write interpretation for each of the outputs in this file.

Use “defaulted_loan” variable as dependent variable and use other variables as independent variables to predict by:

Fitting logistic regression classifier, KNN classifier, ANN-MLP classifier, Naïve Bayes classifier, SVM classifier and Decision Tree classifier with train/test validation sets, leave-one-out cross validation sets, 10-fold cross validation sets and 3 times repeated 10-fold cross validation sets for this problem and select the best predictive model and find the important variables (variable importance) with relative frequencies and its graph for that model

Fitting bagging, random forest and extreme gradient boosting tree models and select the best predictive model for train/test validation sets and find the variable importance with table and graph for that model

Who will get the loan as per the best models identified above? Why? Explain with justifications.

Train Test Validation

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

Splitting the data into training and testing set

```
set.seed(1234)
ind<-sample(2,nrow(bank_loan_df),replace=T,prob = c(0.7,0.3))
train_data<-bank_loan_df[ind==1,]
test_data<-bank_loan_df[ind==2,]
```

Logistic Regression With train/test Validation

Training Logistic Regression Model

```
logistic_clf<-train(defaulted_loan~.,
  data=train_data,
  method="glm",
  family="binomial"
)
summary(logistic_clf)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.6490  -0.6635  -0.3442   0.1409   3.2833
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.235986   0.272446  -4.537 5.72e-06 ***
## age           0.006492   0.008297   0.782  0.4339
## education_level2 0.227329   0.110244   2.062  0.0392 *
## education_level3 0.260781   0.156468   1.667  0.0956 .
## education_level4 0.285038   0.186776   1.526  0.1270
```

```
## education_level5      0.020994    0.447370    0.047    0.9626
## current_employ_year -0.182777    0.012678   -14.416 < 2e-16 ***
## current_address_year -0.094317    0.010300    -9.157 < 2e-16 ***
## income_household     -0.002470    0.003879    -0.637    0.5244
## debt_income_ratio     0.099652    0.012885     7.734 1.04e-14 ***
## credit_card_debt      0.425066    0.044558     9.540 < 2e-16 ***
## other_debts           0.006704    0.030495     0.220    0.8260
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 3994.4  on 3524  degrees of freedom
## Residual deviance: 2850.2  on 3513  degrees of freedom
## AIC: 2874.2
##
## Number of Fisher Scoring iterations: 6
```

Making the Prediction

```
predicted_val_log<-predict(logistic_clf,newdata = test_data)
```

Confusion Matrix for Evaluation

```
confusionMatrix(test_data$defaulted_loan, predicted_val_log)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1038   76
##           1  191  170
##
##               Accuracy : 0.819
##               95% CI : (0.7984, 0.8383)
##           No Information Rate : 0.8332
##           P-Value [Acc > NIR] : 0.9322
##
##               Kappa : 0.4513
##
##  Mcnemar's Test P-Value : 3.022e-12
##
##           Sensitivity : 0.8446
##           Specificity : 0.6911
##           Pos Pred Value : 0.9318
##           Neg Pred Value : 0.4709
##           Prevalence : 0.8332
##           Detection Rate : 0.7037
##           Detection Prevalence : 0.7553
##           Balanced Accuracy : 0.7678
##
##           'Positive' Class : 0
##
```

KNN Model with train/test validation

Training KNN Model

```
knn_clf<-train(defaulted_loan~.,data = train_data,
               method="knn",
               preProcess = c("center", "scale"),
               tuneLength = 10
               )
```

Getting the Result of the Model

```
knn_clf$result
```

```
##      k Accuracy      Kappa AccuracySD      KappaSD
## 1    5 0.7475369 0.2958532 0.010849610 0.02884625
## 2    7 0.7582432 0.3117892 0.008841404 0.02356125
## 3    9 0.7674740 0.3267350 0.008578768 0.02333298
## 4   11 0.7701279 0.3240461 0.009131611 0.02925444
## 5   13 0.7743100 0.3300675 0.009833257 0.02766763
## 6   15 0.7780591 0.3362558 0.008906519 0.02560595
## 7   17 0.7810133 0.3395283 0.011387056 0.03375783
## 8   19 0.7834197 0.3443761 0.010369322 0.02759248
## 9   21 0.7832864 0.3399306 0.008661037 0.02399463
## 10  23 0.7843194 0.3395441 0.010371607 0.02569968
```

Confusion Matrix for Model Evaluation

```
predicted_val_knn<-predict(knn_clf,newdata = test_data)
confusionMatrix(test_data$defaulted_loan, predicted_val_knn)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction      0      1
##              0 1044    70
##              1  234   127
##
##              Accuracy : 0.7939
##              95% CI : (0.7723, 0.8143)
##              No Information Rate : 0.8664
##              P-Value [Acc > NIR] : 1
##
##              Kappa : 0.3414
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.8169
##              Specificity : 0.6447
##              Pos Pred Value : 0.9372
##              Neg Pred Value : 0.3518
##              Prevalence : 0.8664
##              Detection Rate : 0.7078
##              Detection Prevalence : 0.7553
```

```
##      Balanced Accuracy : 0.7308
##
##      'Positive' Class : 0
##
```

Naïve Bayes classifier

Training the Naïve Bayes classifier

```
library(e1071)
nb_clf<-naiveBayes(defaulted_loan~.,data=train_data)
```

Summary of the Model

```
(nb_clf)

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      0      1
## 0.7460993 0.2539007
##
## Conditional probabilities:
##      age
## Y      [,1]      [,2]
## 0 36.22966 7.836511
## 1 32.79218 7.690558
##
##      education_level
## Y      1      2      3      4      5
## 0 0.566539924 0.268821293 0.100380228 0.055513308 0.008745247
## 1 0.442458101 0.294972067 0.144134078 0.101675978 0.016759777
##
##      current_employ_year
## Y      [,1]      [,2]
## 0 9.775665 7.340640
## 1 5.426816 5.490989
##
##      current_address_year
## Y      [,1]      [,2]
## 0 9.001141 7.038583
## 1 5.698324 5.384183
##
##      income_household
## Y      [,1]      [,2]
## 0 48.73426 39.55991
## 1 44.26358 88.72586
##
##      debt_income_ratio
```

```
## Y      [,1]      [,2]
##  0  8.59219  5.526728
##  1 14.56893  7.947622
##
##      credit_card_debt
## Y      [,1]      [,2]
##  0 1.286985  1.583750
##  1 2.661458  5.948471
##
##      other_debts
## Y      [,1]      [,2]
##  0 2.881188  3.38933
##  1 4.271832 14.45939
```

Making the Prediction in the test data

```
predicted_val_nb<-predict(nb_clf,newdata = test_data)
```

Confusion Matrix for Evaluation

```
confusionMatrix(predicted_val_nb,test_data$defaulted_loan)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0      1
##              0 1071  285
##              1   43   76
##
##              Accuracy : 0.7776
##              95% CI : (0.7555, 0.7986)
##              No Information Rate : 0.7553
##              P-Value [Acc > NIR] : 0.02363
##
##              Kappa : 0.2223
##
##  Mcnemar's Test P-Value : < 2e-16
##
##              Sensitivity : 0.9614
##              Specificity : 0.2105
##              Pos Pred Value : 0.7898
##              Neg Pred Value : 0.6387
##              Prevalence : 0.7553
##              Detection Rate : 0.7261
##              Detection Prevalence : 0.9193
##              Balanced Accuracy : 0.5860
##
##              'Positive' Class : 0
##
```

Support Vector Machine (SVM) Model

Training the Model

```
svm_clf<-train(defaulted_loan~.,
               data=train_data,
               method="svmLinear"
               )
svm_clf

## Support Vector Machines with Linear Kernel
##
## 3525 samples
##    8 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3525, 3525, 3525, 3525, 3525, ...
## Resampling results:
##
##   Accuracy   Kappa
##  0.8021072  0.3860155
##
## Tuning parameter 'C' was held constant at a value of 1
```

Making the Prediction for test data

```
predicted_val_svm<-predict(svm_clf,newdata = test_data)
```

Confusion Matrix for Model Evaluation

```
confusionMatrix(predicted_val_svm,test_data$defaulted_loan)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##           0 1055  218
##           1   59  143
##
##              Accuracy : 0.8122
##              95% CI : (0.7913, 0.8318)
##    No Information Rate : 0.7553
##    P-Value [Acc > NIR] : 9.898e-08
##
##              Kappa : 0.4032
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.9470
##              Specificity : 0.3961
##              Pos Pred Value : 0.8288
##              Neg Pred Value : 0.7079
```

```
##           Prevalence : 0.7553
##       Detection Rate : 0.7153
##   Detection Prevalence : 0.8631
##       Balanced Accuracy : 0.6716
##
##       'Positive' Class : 0
##
```

Decision Tree Model

```
dtree_clf<-train(defaulted_loan~.,
                 data = train_data,
                 method="rpart",
                 parms = list(split = "information"),
                 tuneLength=10
                 )
dtree_clf
```

```
## CART
##
## 3525 samples
##    8 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3525, 3525, 3525, 3525, 3525, 3525, ...
## Resampling results across tuning parameters:
##
##    cp          Accuracy    Kappa
## 0.002793296  0.7628193  0.3313959
## 0.002979516  0.7639354  0.3327615
## 0.003072626  0.7641848  0.3335925
## 0.003351955  0.7654893  0.3365711
## 0.004469274  0.7724129  0.3390493
## 0.005586592  0.7756679  0.3393153
## 0.006703911  0.7765469  0.3352378
## 0.024581006  0.7714426  0.3095524
## 0.027374302  0.7698968  0.3065844
## 0.060335196  0.7588103  0.2140312
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.006703911.
```

Making the Prediction for test data

```
predicted_val_dtree<-predict(dtree_clf,newdata = test_data)
```

Confusion Matrix for Model Evaluation

```
confusionMatrix(predicted_val_dtree,test_data$defaulted_loan)
```

```
## Confusion Matrix and Statistics
##
```



```
##           Reference
## Prediction    0    1
##           0 1024  214
##           1   90  147
##
##           Accuracy : 0.7939
##           95% CI : (0.7723, 0.8143)
##       No Information Rate : 0.7553
##       P-Value [Acc > NIR] : 0.0002488
##
##           Kappa : 0.3693
##
## Mcnemar's Test P-Value : 1.732e-12
##
##           Sensitivity : 0.9192
##           Specificity : 0.4072
##       Pos Pred Value : 0.8271
##       Neg Pred Value : 0.6203
##           Prevalence : 0.7553
##       Detection Rate : 0.6942
##       Detection Prevalence : 0.8393
##       Balanced Accuracy : 0.6632
##
##       'Positive' Class : 0
##
```

Artificial Neural Network (ANN) Model

Training the Model

```
ann_clf <- train(defaulted_loan ~ ., data = train_data,
  method = "nnet",
  preProcess = c("center", "scale"),
  maxit = 250,      # Maximum number of iterations
  tuneGrid = data.frame(size = 1, decay = 0),
  # tuneGrid = data.frame(size = 0, decay = 0), skip=TRUE, # Technically, this is log-reg
  metric = "Accuracy")
```

```
## # weights:  14
## initial  value 2613.046683
## iter  10 value 1718.710496
## iter  20 value 1592.438604
## iter  30 value 1536.737176
## iter  40 value 1526.182944
## iter  50 value 1523.444206
## final   value 1523.436772
## converged
## # weights:  14
## initial  value 2275.207222
## iter  10 value 1500.345761
## iter  20 value 1463.723902
## iter  30 value 1441.634831
## iter  40 value 1440.920899
## iter  50 value 1440.592609
```

```

## final value 1440.590315
## converged
## # weights: 14
## initial value 2011.769843
## iter 10 value 1590.196075
## iter 20 value 1519.334741
## iter 30 value 1502.384720
## iter 40 value 1476.810780
## iter 50 value 1435.085892
## iter 60 value 1431.729533
## iter 70 value 1425.368597
## iter 80 value 1423.689809
## iter 90 value 1423.555939
## iter 100 value 1422.791026
## iter 110 value 1422.554549
## final value 1422.551557
## converged
## # weights: 14
## initial value 2479.745029
## iter 10 value 1601.520279
## iter 20 value 1518.517527
## iter 30 value 1474.971927
## iter 40 value 1466.334640
## iter 50 value 1455.793401
## iter 60 value 1454.149511
## iter 70 value 1454.129241
## iter 80 value 1453.915406
## final value 1453.913052
## converged
## # weights: 14
## initial value 2366.550939
## iter 10 value 1883.976424
## iter 20 value 1681.129153
## iter 30 value 1643.111043
## iter 40 value 1621.281542
## iter 50 value 1592.906786
## iter 60 value 1468.651879
## iter 70 value 1450.313797
## iter 80 value 1437.662224
## iter 90 value 1437.429726
## iter 100 value 1437.426525
## final value 1437.423512
## converged
## # weights: 14
## initial value 2050.310339
## iter 10 value 1513.483377
## iter 20 value 1469.996776
## iter 30 value 1432.895914
## iter 40 value 1429.608101
## iter 50 value 1423.732388
## iter 60 value 1422.957948
## iter 70 value 1422.945941
## iter 80 value 1422.793176
## iter 90 value 1422.718358

```

```

## final value 1422.716653
## converged
## # weights: 14
## initial value 2187.750269
## iter 10 value 1471.610549
## iter 20 value 1449.536581
## iter 30 value 1420.041762
## iter 40 value 1417.532990
## iter 50 value 1414.778455
## iter 60 value 1413.179057
## iter 70 value 1413.069570
## iter 80 value 1412.814861
## iter 90 value 1412.536441
## iter 100 value 1412.508856
## iter 100 value 1412.508850
## final value 1412.508536
## converged
## # weights: 14
## initial value 2637.923581
## iter 10 value 1848.212047
## iter 20 value 1542.841896
## iter 30 value 1519.402393
## iter 40 value 1501.916077
## iter 50 value 1444.195842
## iter 60 value 1418.819503
## iter 70 value 1417.131761
## iter 80 value 1414.024434
## iter 90 value 1413.665977
## iter 100 value 1413.660328
## iter 110 value 1413.599396
## final value 1413.594665
## converged
## # weights: 14
## initial value 2409.667095
## iter 10 value 1575.492806
## iter 20 value 1480.411304
## iter 30 value 1437.463163
## iter 40 value 1408.070562
## iter 50 value 1348.155065
## iter 60 value 1325.258577
## iter 70 value 1324.615789
## iter 80 value 1322.176531
## iter 90 value 1321.842433
## iter 100 value 1321.838683
## iter 110 value 1321.789496
## final value 1321.783612
## converged
## # weights: 14
## initial value 2049.521479
## iter 10 value 1555.402100
## iter 20 value 1485.679690
## iter 30 value 1451.920665
## iter 40 value 1413.604630
## iter 50 value 1380.341360

```

```

## iter 60 value 1372.708311
## iter 70 value 1372.634106
## iter 80 value 1371.219266
## iter 90 value 1370.973682
## iter 100 value 1370.968755
## final value 1370.968170
## converged
## # weights: 14
## initial value 2108.533206
## iter 10 value 1554.193197
## iter 20 value 1494.311468
## iter 30 value 1439.432253
## iter 40 value 1432.298981
## iter 50 value 1426.023898
## iter 60 value 1425.172669
## iter 70 value 1425.133430
## iter 80 value 1424.649173
## iter 90 value 1424.559696
## iter 100 value 1424.550213
## iter 110 value 1424.466599
## final value 1424.449570
## converged
## # weights: 14
## initial value 3069.050813
## iter 10 value 1809.001967
## iter 20 value 1637.663344
## iter 30 value 1571.844599
## iter 40 value 1556.842334
## iter 50 value 1547.601978
## iter 60 value 1510.933576
## iter 70 value 1506.355243
## iter 80 value 1499.905156
## iter 90 value 1498.665328
## iter 100 value 1498.647728
## iter 110 value 1498.205249
## iter 120 value 1498.126404
## iter 130 value 1498.091099
## iter 140 value 1498.026506
## iter 150 value 1497.998916
## iter 150 value 1497.998910
## iter 150 value 1497.998909
## final value 1497.998909
## converged
## # weights: 14
## initial value 3053.196708
## iter 10 value 1675.266156
## iter 20 value 1587.260676
## iter 30 value 1523.930254
## iter 40 value 1493.615864
## iter 50 value 1449.271249
## iter 60 value 1426.826405
## iter 70 value 1426.273203
## iter 80 value 1423.485520
## iter 90 value 1422.952227

```

```

## iter 100 value 1422.941938
## iter 110 value 1422.773572
## iter 120 value 1422.704453
## final value 1422.697687
## converged
## # weights: 14
## initial value 2461.934385
## iter 10 value 1619.457718
## iter 20 value 1500.294509
## iter 30 value 1449.336774
## iter 40 value 1443.822339
## iter 50 value 1438.278951
## iter 60 value 1437.450193
## iter 70 value 1437.441790
## iter 80 value 1437.299523
## iter 90 value 1437.151023
## final value 1437.134109
## converged
## # weights: 14
## initial value 2272.912304
## iter 10 value 1545.994078
## iter 20 value 1440.691377
## iter 30 value 1429.004198
## iter 40 value 1428.847520
## iter 50 value 1427.510998
## iter 60 value 1427.173009
## final value 1427.171721
## converged
## # weights: 14
## initial value 2229.789195
## iter 10 value 1607.824388
## iter 20 value 1496.395918
## iter 30 value 1483.435576
## iter 40 value 1483.264726
## final value 1483.128850
## converged
## # weights: 14
## initial value 2363.581537
## iter 10 value 1708.413328
## iter 20 value 1419.496271
## iter 30 value 1411.499881
## iter 40 value 1411.376652
## iter 50 value 1410.918817
## iter 60 value 1410.889472
## iter 60 value 1410.889471
## iter 60 value 1410.889470
## final value 1410.889470
## converged
## # weights: 14
## initial value 2386.636495
## iter 10 value 1795.460809
## iter 20 value 1472.597944
## iter 30 value 1459.401295
## iter 40 value 1458.562957

```

```

## iter 50 value 1456.631058
## iter 60 value 1456.211004
## iter 70 value 1456.199531
## iter 80 value 1456.057885
## final value 1455.988515
## converged
## # weights: 14
## initial value 2557.140579
## iter 10 value 1688.340243
## iter 20 value 1575.514282
## iter 30 value 1509.840106
## iter 40 value 1485.220745
## iter 50 value 1471.633699
## iter 60 value 1468.544822
## iter 70 value 1468.529390
## iter 80 value 1468.145745
## iter 90 value 1468.076451
## final value 1468.076308
## converged
## # weights: 14
## initial value 2204.118914
## iter 10 value 1613.533151
## iter 20 value 1514.089012
## iter 30 value 1449.324654
## iter 40 value 1426.690241
## iter 50 value 1408.317651
## iter 60 value 1403.718211
## iter 70 value 1403.417116
## iter 80 value 1402.206312
## iter 90 value 1401.822516
## iter 100 value 1401.819736
## iter 100 value 1401.819722
## iter 100 value 1401.819719
## final value 1401.819719
## converged
## # weights: 14
## initial value 2518.315536
## iter 10 value 1659.381805
## iter 20 value 1496.279232
## iter 30 value 1461.978410
## iter 40 value 1459.474552
## iter 50 value 1453.375486
## iter 60 value 1452.324189
## iter 70 value 1452.310161
## iter 80 value 1451.940475
## iter 90 value 1451.842742
## iter 100 value 1451.841758
## iter 110 value 1451.808035
## iter 120 value 1451.747226
## iter 130 value 1451.727623
## final value 1451.727485
## converged
## # weights: 14
## initial value 2378.121571

```

```

## iter 10 value 1608.647123
## iter 20 value 1510.844930
## iter 30 value 1462.379329
## iter 40 value 1449.371277
## iter 50 value 1436.709725
## iter 60 value 1435.182485
## iter 70 value 1435.143193
## iter 80 value 1434.886743
## iter 90 value 1434.857763
## final value 1434.857534
## converged
## # weights: 14
## initial value 2929.686189
## iter 10 value 1683.642233
## iter 20 value 1509.986311
## iter 30 value 1454.368821
## iter 40 value 1436.199554
## iter 50 value 1425.713528
## iter 60 value 1424.623923
## iter 70 value 1424.545482
## iter 80 value 1424.310247
## final value 1424.309693
## converged
## # weights: 14
## initial value 3580.391285
## iter 10 value 1685.090901
## iter 20 value 1629.433601
## iter 30 value 1609.588787
## iter 40 value 1548.076563
## iter 50 value 1498.774731
## iter 60 value 1454.729637
## iter 70 value 1445.003622
## iter 80 value 1439.518839
## iter 90 value 1439.013290
## iter 100 value 1438.994744
## iter 110 value 1438.942018
## iter 110 value 1438.942012
## iter 110 value 1438.942012
## final value 1438.942012
## converged
## # weights: 14
## initial value 2185.002164
## iter 10 value 1608.968509
## iter 20 value 1408.574131
## iter 30 value 1393.457515
## iter 40 value 1392.689886
## iter 50 value 1390.290347
## iter 60 value 1389.929644
## iter 70 value 1389.914760
## iter 80 value 1389.649669
## iter 90 value 1389.490833
## final value 1389.489779
## converged
## # weights: 14

```

```
## initial value 2064.984403
## iter 10 value 1569.276090
## iter 20 value 1505.105628
## iter 30 value 1444.432875
## iter 40 value 1435.022730
## iter 50 value 1427.108260
## iter 60 value 1425.341474
## iter 70 value 1425.319799
## iter 80 value 1425.033977
## iter 90 value 1424.945179
## final value 1424.944896
## converged
```

Making the Predictions for Test data

```
predicted_val_ann<-predict(ann_clf,newdata = test_data)
```

Confusion Matrix for the Model Evaluation

```
confusionMatrix(predicted_val_ann,test_data$defaulted_loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1036  191
##           1   78  170
##
##           Accuracy : 0.8176
##           95% CI : (0.797, 0.837)
##           No Information Rate : 0.7553
##           P-Value [Acc > NIR] : 5.382e-09
##
##           Kappa : 0.4483
##
## Mcnemar's Test P-Value : 8.565e-12
##
##           Sensitivity : 0.9300
##           Specificity : 0.4709
##           Pos Pred Value : 0.8443
##           Neg Pred Value : 0.6855
##           Prevalence : 0.7553
##           Detection Rate : 0.7024
##           Detection Prevalence : 0.8319
##           Balanced Accuracy : 0.7004
##
##           'Positive' Class : 0
##
```