Credit Card Default Analysis

Rudrani Angira, Srivatsan Iyer

April 23, 2016

General Utility class (Data.R)

```
get_credit_dataset <- function() {
  dataset = read.csv("credit.csv",header=TRUE,skip=1)
  ones = dataset[dataset$default.payment.next.month == 1,]
  zeroes = dataset[dataset$default.payment.next.month == 0,]

  res = rbind(
    ones[sample(nrow(ones), 6000),],
    zeroes[sample(nrow(ones), 6000),]
  )
  return(res[sample(nrow(res)),])
}


split_data <- function(data) {
  pos = nrow(data) * 0.8
  return(list(
    train_features=data[1:pos, 2:(ncol(data)-1)],
    train_responses=data[1:pos, ncol(data)],
    test_features=data[(pos + 1):nrow(data), 2:(ncol(data)-1)],
    test_responses=data[(pos + 1):nrow(data),ncol(data)]
  ))
}

split_data_row <- function(data) {
  return(list(
    train=head(data, n=0.8*nrow(data)),
    test=tail(data, n=0.2*nrow(data))
  ))
}
```

General Utility class (Preprocess.R)

```
preprocess <- function(dataset, bucket_size){
  dataset <- subset(dataset, select = -c(1))
  # making buckets for the columns for Bill_Amt and Pay_Amt
  df <- subset(dataset, select = -c((2:4),(6:11)))
  for(i in 1:(ncol(df)-1))
  {
    str=paste(colnames(df)[i],"NEW")
    cat(paste0(str))
```

```
    bins<-bucket_size
    cutpoints<-unique(quantile(df[,i],(0:bins)/bins))
    df[,str]<-NA
    df[,str]=as.numeric(cut(df[,i],cutpoints,include.lowest=TRUE))
  }

  df <- subset(df, select = -c(1:14))
  df1<- subset(dataset, select = -c(1,5,(12:24)))
  df=cbind(df,df1)
  colnames(df)[1]=c("default")
  df2 <- subset(df, select = -c(1))
  names(df) <- sub(" ", ".", names(df))
  trainset <- head(df, n=nrow(dataset)*0.8)
  testset <- tail(df, n=nrow(dataset)*0.2)

  return(list(train=trainset, test=testset))
}
```

Summary for the feature vectors

```
dataset <- read.csv("credit.csv",header=TRUE,skip=1)
summary(dataset)
```

```
##        ID          LIMIT_BAL           SEX          EDUCATION
##  Min.   :    1   Min.   :  10000   Min.   :1.000   Min.   :0.000
##  1st Qu.: 7501   1st Qu.:  50000   1st Qu.:1.000   1st Qu.:1.000
##  Median :15000   Median : 140000   Median :2.000   Median :2.000
##  Mean   :15000   Mean   : 167484   Mean   :1.604   Mean   :1.853
##  3rd Qu.:22500   3rd Qu.: 240000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :30000   Max.   :1000000   Max.   :2.000   Max.   :6.000
##     MARRIAGE          AGE            PAY_0            PAY_2
##  Min.   :0.000   Min.   :21.00   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:1.000   1st Qu.:28.00   1st Qu.:-1.0000   1st Qu.:-1.0000
##  Median :2.000   Median :34.00   Median : 0.0000   Median : 0.0000
##  Mean   :1.552   Mean   :35.49   Mean   :-0.0167   Mean   :-0.1338
##  3rd Qu.:2.000   3rd Qu.:41.00   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :3.000   Max.   :79.00   Max.   : 8.0000   Max.   : 8.0000
##      PAY_3             PAY_4             PAY_5             PAY_6
##  Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   :-0.1662   Mean   :-0.2207   Mean   :-0.2662   Mean   :-0.2911
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000
##    BILL_AMT1         BILL_AMT2         BILL_AMT3         BILL_AMT4
##  Min.   :-165580   Min.   :-69777   Min.   :-157264   Min.   :-170000
##  1st Qu.:   3559   1st Qu.:  2985   1st Qu.:   2666   1st Qu.:   2327
##  Median :  22382   Median : 21200   Median :  20088   Median :  19052
##  Mean   :  51223   Mean   : 49179   Mean   :  47013   Mean   :  43263
##  3rd Qu.:  67091   3rd Qu.: 64006   3rd Qu.:  60165   3rd Qu.:  54506
```

```
##  Max.   : 964511   Max.   :983931   Max.   :1664089   Max.   : 891586
##    BILL_AMT5        BILL_AMT6          PAY_AMT1         PAY_AMT2
##  Min.   :-81334   Min.   :-339603   Min.   :     0   Min.   :      0
##  1st Qu.:  1763   1st Qu.:   1256   1st Qu.:  1000   1st Qu.:    833
##  Median : 18104   Median :  17071   Median :  2100   Median :   2009
##  Mean   : 40311   Mean   :  38872   Mean   :  5664   Mean   :   5921
##  3rd Qu.: 50190   3rd Qu.:  49198   3rd Qu.:  5006   3rd Qu.:   5000
##  Max.   :927171   Max.   : 961664   Max.   :873552   Max.   :1684259
##     PAY_AMT3         PAY_AMT4          PAY_AMT5           PAY_AMT6
##  Min.   :     0   Min.   :     0   Min.   :     0.0   Min.   :     0.0
##  1st Qu.:   390   1st Qu.:   296   1st Qu.:   252.5   1st Qu.:   117.8
##  Median :  1800   Median :  1500   Median :  1500.0   Median :  1500.0
##  Mean   :  5226   Mean   :  4826   Mean   :  4799.4   Mean   :  5215.5
##  3rd Qu.:  4505   3rd Qu.:  4013   3rd Qu.:  4031.5   3rd Qu.:  4000.0
##  Max.   :896040   Max.   :621000   Max.   :426529.0   Max.   :528666.0
##  default.payment.next.month
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2212
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

```
trainset <- dataset[1:15000, 2:25]
testset <- dataset[15001:30000,2:25]

# Correlation between the continuous variables (BILL_AMT and PAY_AMT)
# It can be viewed that the BILL_AMT columns are correlated with each other but there is # # no signifi

data=dataset[1:15000,13:24]
cor(data[,1:12])
```

```
##            BILL_AMT1  BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6
## BILL_AMT1 1.00000000 0.95229982 0.8810510 0.8559326 0.8379851 0.8133242
## BILL_AMT2 0.95229982 1.00000000 0.9148728 0.8873175 0.8666386 0.8397643
## BILL_AMT3 0.88105103 0.91487281 1.0000000 0.9157875 0.8891903 0.8616146
## BILL_AMT4 0.85593257 0.88731750 0.9157875 1.0000000 0.9503005 0.9123564
## BILL_AMT5 0.83798511 0.86663864 0.8891903 0.9503005 1.0000000 0.9504989
## BILL_AMT6 0.81332420 0.83976428 0.8616146 0.9123564 0.9504989 1.0000000
## PAY_AMT1  0.14263068 0.29327058 0.2476912 0.2349094 0.2183935 0.2009011
## PAY_AMT2  0.09393846 0.08557701 0.3607816 0.2122274 0.1831374 0.1757368
## PAY_AMT3  0.19317663 0.19073256 0.1686215 0.3124196 0.2625181 0.2513166
## PAY_AMT4  0.17744778 0.17147406 0.1737457 0.1445871 0.3008712 0.2595687
## PAY_AMT5  0.17278901 0.16859182 0.2104096 0.1761039 0.1575323 0.3201159
## PAY_AMT6  0.15029919 0.15598062 0.1639361 0.1616830 0.1473934 0.1029035
##             PAY_AMT1   PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
## BILL_AMT1 0.1426307 0.09393846 0.1931766 0.1774478 0.1727890 0.1502992
## BILL_AMT2 0.2932706 0.08557701 0.1907326 0.1714741 0.1685918 0.1559806
## BILL_AMT3 0.2476912 0.36078164 0.1686215 0.1737457 0.2104096 0.1639361
## BILL_AMT4 0.2349094 0.21222740 0.3124196 0.1445871 0.1761039 0.1616830
## BILL_AMT5 0.2183935 0.18313736 0.2625181 0.3008712 0.1575323 0.1473934
## BILL_AMT6 0.2009011 0.17573675 0.2513166 0.2595687 0.3201159 0.1029035
## PAY_AMT1  1.0000000 0.15628596 0.1730528 0.1602097 0.1764671 0.2079791
## PAY_AMT2  0.1562860 1.00000000 0.1640356 0.1420433 0.2391375 0.1639878
```

```
## PAY_AMT3  0.1730528 0.16403559 1.0000000 0.1889879 0.1611369 0.1653899
## PAY_AMT4  0.1602097 0.14204328 0.1889879 1.0000000 0.1570888 0.1368017
## PAY_AMT5  0.1764671 0.23913747 0.1611369 0.1570888 1.0000000 0.1582775
## PAY_AMT6  0.2079791 0.16398781 0.1653899 0.1368017 0.1582775 1.0000000
```

K Nearest Neighbors

```
library(class)

source('data.R')

run_knn <- function(train_features, train_responses, test_features, test_responses, k) {
  result = knn(train_features, test_features, train_responses, k = k, prob = FALSE)
  t = table(result,test_responses)
  return(list(
    confusion=t,
    accuracy=(t[1,1] + t[2,2])/sum(t)
  ))
}

plot_best_k <- function(train_features, train_responses, test_features, test_responses) {
  all_k = seq(10,150,10)
  accuracies = c()
  for (k in all_k) {
    res = run_knn(train_features, train_responses, test_features, test_responses, k)
    accuracies = c(accuracies, res$accuracy)
  }
  plot(all_k, accuracies, type="l", xlab="K values", ylab="Accuracy", main="Plot of accuracy against 'K
}

dataset <- get_credit_dataset()
data = split_data(dataset)
#plot_best_k(data$train_features, data$train_responses, data$test_features, data$test_responses)

result = run_knn(data$train_features, data$train_responses, data$test_features, data$test_responses, 70
print(result$confusion)
```

```
##       test_responses
## result   0   1
##      0 668 413
##      1 535 784
```

```
print(result$accuracy)
```

```
## [1] 0.605
```

Decision Tree Implementation

```
library(Hmisc)
library(MASS)
library(rpart)
library(data.tree)

source("data.R")
source("Preprocess.R")

entropy <- function(responses) {
  sum = 0
  for (unique_val in unique(responses)) {
    prob = length(responses[responses==unique_val]) / length(responses)
    sum = sum - (prob * log(prob))
  }
  return(sum)
}

get_max_prob <- function(responses) {
  if (nrow(as.matrix(responses[responses==1])) > nrow(as.matrix(responses[responses==0]))) {
    return(1)
  } else {
    return(0)
  }
}

id3 <- function(node, features, responses, cutoff) {
  if (length(unique(responses)) == 1) {
    node$splitBy = NULL
    node$response = responses[1]
    return()
  }

  if (node$depth >= cutoff) {
    node$splitBy = NULL
    node$response = get_max_prob(responses)
    return()
  }

  #set temporary responses in case we encounter new edge in testing phase
  node$response = get_max_prob(responses)

  min_feature_entropy = 10e10
  min_feature_entropy_index = -1
  for (feature_index in 1:ncol(features)) {
    sum_entropy = 0
    for (unique_val in unique(features[,feature_index])) {
      subset = responses[features[,feature_index] == unique_val]
      sum_entropy = sum_entropy + length(subset) / nrow(features) * entropy(subset)
    }
    if (min_feature_entropy > sum_entropy) {
      min_feature_entropy =  sum_entropy
      min_feature_entropy_index = feature_index
    }
```

```
  }

  node$splitBy = colnames(features)[min_feature_entropy_index]
  for (unique_val in unique(features[,min_feature_entropy_index])) {
    child = node$AddChild(as.character(paste(colnames(features)[min_feature_entropy_index],"=",unique_va
    child$depth = node$depth + 1
    id3(child,
        features[features[min_feature_entropy_index]==unique_val, -c(min_feature_entropy_index)],
        responses[features[min_feature_entropy_index]==unique_val],
        cutoff)
  }
}

dtree_train <- function(features, responses, cutoff) {
  node = Node$new("root")
  node$depth = 0
  id3(node, features, as.vector(responses), cutoff)
  return(node)
}

dtree_predict <- function(node, row) {
  if (is.null(node$splitBy)) {
    return(node$response)
  }
  if (exists(paste(node$splitBy,"=",get(node$splitBy, row)), node)) {
    return(dtree_predict(get(paste(node$splitBy,"=",get(node$splitBy, row)), node), row))
  } else {
    return(node$response)
  }
}

dtree_test <- function(node, features) {
  res = c()
  for (i in 1:nrow(features)) {
    row = features[i,]
    ret = dtree_predict(node, row)
    res = c(res, ret)
  }
  return(res)
}

train_test <- function(trainset, testset, cutoff, verbose=FALSE) {
  model = dtree_train(trainset[,2:ncol(trainset)], as.matrix(trainset[,c(1)]), cutoff)
  if (verbose) {
    print(model, "splitBy", "response")
  }
  results = dtree_test(model, testset[,2:ncol(testset)])
  #View(results)
  results=cbind(testset[,c(1)],results)
  ret = table(results[,1],results[,2])
  if (verbose) {
    print(ret)
  }
```

```
  accuracy = (ret[1,1]+ret[2,2])/(ret[1,1] + ret[1,2] + ret[2,1] + ret[2,2])
  return(list(tn=ret[1,1], tp=ret[2,2], accuracy=accuracy))
}

plot_bucket_size <- function(dataset) {
  result = c()
  bucket_sizes = c(seq(1,10), seq(12, 21, 3))
  for (i in bucket_sizes) {
    ret = preprocess(dataset, i)
    trainset = ret$train
    testset = ret$test
    res = train_test(trainset, testset, 4)
    cat(paste("Accuracy with bucket_size=, ", i, " is: ", res$accuracy, "\n"))
    result = c(result, res)
  }
  plot(bucket_sizes, result[seq(3, length(result), 3)], type="l", ylab="Overall Accuracy", xlab="Decisi
}

plot_cutoff <- function(dataset) {
  result = c()
  cut_offs = seq(1,15)
  for (cutoff in cut_offs) {
    ret = preprocess(dataset, 5)
    trainset = ret$train
    testset = ret$test
    res = train_test(trainset, testset, cutoff)
    cat(paste("Accuracy with cut_off= ", cutoff, " is: ", res$accuracy, "\n"))
    result = c(result, res)
  }
  plot(cut_offs, result[seq(3, length(result), 3)], type="l", ylab="Overall Accuracy", xlab="Decision t
}

run_dtree <- function(dataset) {
  ret = preprocess(dataset, 5)
  trainset = ret$train
  testset = ret$test
  res = train_test(trainset, testset, 3, verbose=TRUE)
  cat(paste("Accuracy: ", res$accuracy, "\n"))
}

dataset <- get_credit_dataset()
#plot_bucket_size(dataset)
#plot_cutoff(dataset)
run_dtree(dataset)
```

```
## LIMIT_BAL NEWAGE NEWBILL_AMT1 NEWBILL_AMT2 NEWBILL_AMT3 NEWBILL_AMT4 NEWBILL_AMT5 NEWBILL_AMT6 NEWPA
## 1   root                                          PAY_0        1
## 2    ¦--PAY_0 = -1                          PAY_AMT1.NEW        0
## 3    ¦   ¦--PAY_AMT1.NEW = 1                       PAY_3        1
## 4    ¦   ¦   ¦--PAY_3 = 0                                      1
## 5    ¦   ¦   ¦--PAY_3 = 2                                      1
## 6    ¦   ¦   ¦--PAY_3 = -1                                     0
## 7    ¦   ¦   ¦--PAY_3 = -2                                     0
```

```
## 8    ¦   ¦   ¦--PAY_3 = 3                                1
## 9    ¦   ¦   °--PAY_3 = 4                                1
## 10   ¦   ¦--PAY_AMT1.NEW = 2              PAY_4         0
## 11   ¦   ¦   ¦--PAY_4 = 2                                1
## 12   ¦   ¦   ¦--PAY_4 = -1                               0
## 13   ¦   ¦   ¦--PAY_4 = 0                                0
## 14   ¦   ¦   °--PAY_4 = -2                               0
## 15   ¦   ¦--PAY_AMT1.NEW = 4              PAY_4         0
## 16   ¦   ¦   ¦--PAY_4 = -2                               0
## 17   ¦   ¦   ¦--PAY_4 = -1                               0
## 18   ¦   ¦   ¦--PAY_4 = 0                                0
## 19   ¦   ¦   °--PAY_4 = 2                                1
## 20   ¦   °--PAY_AMT1.NEW = 3         LIMIT_BAL.NEW       0
## 21   ¦       ¦--LIMIT_BAL.NEW = 3                        0
## 22   ¦       ¦--LIMIT_BAL.NEW = 1                        0
## 23   ¦       ¦--LIMIT_BAL.NEW = 2                        0
## 24   ¦       ¦--LIMIT_BAL.NEW = 5                        0
## 25   ¦       °--LIMIT_BAL.NEW = 4                        0
## 26   ¦--PAY_0 = 0                          PAY_4         0
## 27   ¦   ¦--PAY_4 = 2                      PAY_6         1
## 28   ¦   ¦   ¦--PAY_6 = 2                                1
## 29   ¦   ¦   ¦--PAY_6 = 0                                1
## 30   ¦   ¦   ¦--PAY_6 = -2                               1
## 31   ¦   ¦   ¦--PAY_6 = -1                               1
## 32   ¦   ¦   ¦--PAY_6 = 6                                0
## 33   ¦   ¦   ¦--PAY_6 = 3                                1
## 34   ¦   ¦   °--PAY_6 = 4                                1
## 35   ¦   ¦--PAY_4 = 0                 LIMIT_BAL.NEW       0
## 36   ¦   ¦   ¦--LIMIT_BAL.NEW = 1                        0
## 37   ¦   ¦   ¦--LIMIT_BAL.NEW = 4                        0
## 38   ¦   ¦   ¦--LIMIT_BAL.NEW = 3                        0
## 39   ¦   ¦   ¦--LIMIT_BAL.NEW = 2                        0
## 40   ¦   ¦   °--LIMIT_BAL.NEW = 5                        0
## 41   ¦   ¦--PAY_4 = -1                     PAY_3         0
## 42   ¦   ¦   ¦--PAY_3 = 2                                1
## 43   ¦   ¦   ¦--PAY_3 = 0                                0
## 44   ¦   ¦   ¦--PAY_3 = -1                               0
## 45   ¦   ¦   °--PAY_3 = -2                               0
## 46   ¦   ¦--PAY_4 = -2                BILL_AMT4.NEW       0
## 47   ¦   ¦   ¦--BILL_AMT4.NEW = 1                        0
## 48   ¦   ¦   ¦--BILL_AMT4.NEW = 4                        1
## 49   ¦   ¦   ¦--BILL_AMT4.NEW = 2                        1
## 50   ¦   ¦   °--BILL_AMT4.NEW = 3                        0
## 51   ¦   ¦--PAY_4 = 3                 BILL_AMT2.NEW       1
## 52   ¦   ¦   ¦--BILL_AMT2.NEW = 3                        0
## 53   ¦   ¦   ¦--BILL_AMT2.NEW = 2                        1
## 54   ¦   ¦   °--BILL_AMT2.NEW = 4                        1
## 55   ¦   °--PAY_4 = 4                                    1
## 56   ¦--PAY_0 = -2                     PAY_AMT1.NEW       0
## 57   ¦   ¦--PAY_AMT1.NEW = 3          BILL_AMT2.NEW       0
## 58   ¦   ¦   ¦--BILL_AMT2.NEW = 2                        0
## 59   ¦   ¦   ¦--BILL_AMT2.NEW = 1                        0
## 60   ¦   ¦   ¦--BILL_AMT2.NEW = 5                        0
## 61   ¦   ¦   ¦--BILL_AMT2.NEW = 4                        0
```

```
## 62   ¦   ¦   °--BILL_AMT2.NEW = 3                      1
## 63   ¦   ¦--PAY_AMT1.NEW = 1         LIMIT_BAL.NEW     0
## 64   ¦   ¦   ¦--LIMIT_BAL.NEW = 4                      0
## 65   ¦   ¦   ¦--LIMIT_BAL.NEW = 5                      1
## 66   ¦   ¦   ¦--LIMIT_BAL.NEW = 3                      0
## 67   ¦   ¦   ¦--LIMIT_BAL.NEW = 1                      0
## 68   ¦   ¦   °--LIMIT_BAL.NEW = 2                      0
## 69   ¦   ¦--PAY_AMT1.NEW = 4               AGE.NEW     0
## 70   ¦   ¦   ¦--AGE.NEW = 1                            0
## 71   ¦   ¦   ¦--AGE.NEW = 4                            0
## 72   ¦   ¦   ¦--AGE.NEW = 2                            0
## 73   ¦   ¦   ¦--AGE.NEW = 3                            0
## 74   ¦   ¦   °--AGE.NEW = 5                            0
## 75   ¦   °--PAY_AMT1.NEW = 2         LIMIT_BAL.NEW     0
## 76   ¦       ¦--LIMIT_BAL.NEW = 3                      0
## 77   ¦       ¦--LIMIT_BAL.NEW = 5                      0
## 78   ¦       ¦--LIMIT_BAL.NEW = 2                      0
## 79   ¦       ¦--LIMIT_BAL.NEW = 4                      0
## 80   ¦       °--LIMIT_BAL.NEW = 1                      0
## 81   ¦--PAY_0 = 2                           PAY_4       1
## 82   ¦   ¦--PAY_4 = 2                      AGE.NEW     1
## 83   ¦   ¦   ¦--AGE.NEW = 3                            1
## 84   ¦   ¦   ¦--AGE.NEW = 1                            1
## 85   ¦   ¦   ¦--AGE.NEW = 5                            1
## 86   ¦   ¦   ¦--AGE.NEW = 4                            1
## 87   ¦   ¦   °--AGE.NEW = 2                            1
## 88   ¦   ¦--PAY_4 = 0                  PAY_AMT3.NEW     1
## 89   ¦   ¦   ¦--PAY_AMT3.NEW = 1                       1
## 90   ¦   ¦   ¦--PAY_AMT3.NEW = 3                       1
## 91   ¦   ¦   ¦--PAY_AMT3.NEW = 2                       1
## 92   ¦   ¦   °--PAY_AMT3.NEW = 4                       1
## 93   ¦   ¦--PAY_4 = 3                 BILL_AMT4.NEW     1
## 94   ¦   ¦   ¦--BILL_AMT4.NEW = 2                      1
## 95   ¦   ¦   ¦--BILL_AMT4.NEW = 1                      0
## 96   ¦   ¦   ¦--BILL_AMT4.NEW = 3                      1
## 97   ¦   ¦   ¦--BILL_AMT4.NEW = 5                      1
## 98   ¦   ¦   °--BILL_AMT4.NEW = 4                      1
## 99   ¦   ¦--PAY_4 = 4                     EDUCATION     1
## 100  ¦   ¦   °--... 3 nodes w/ 0 sub                   NA
## 101  ¦   °--... 5 nodes w/ 19 sub                      NA
## 102  °--... 7 nodes w/ 112 sub                         NA
##
##       0   1
##   0 995 208
##   1 512 685
## Accuracy:  0.7
```

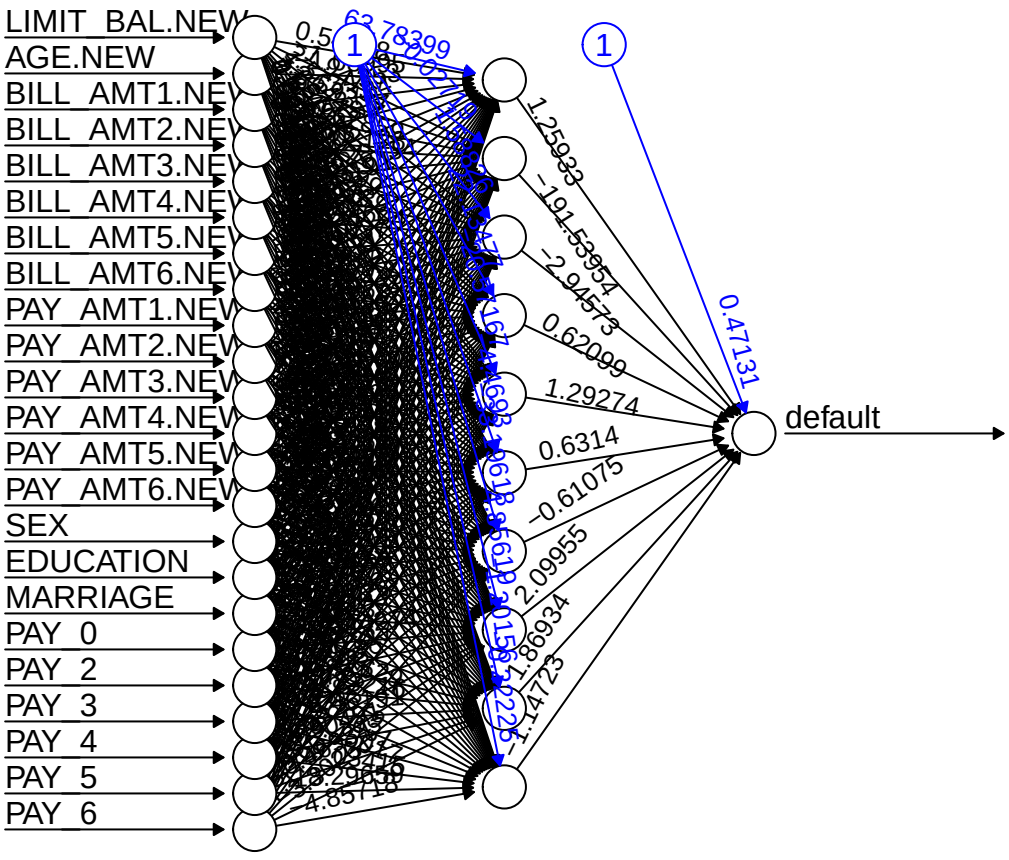
Neural Network

```
source("data.R")
```

```
set.seed(1234567890)
library("neuralnet")
require(neuralnet)

normalize_columns <- function(data, columns) {
  for (i in 1:nrow(data)) {
    s = sum(data[i, columns])
    if (s == 0) {
      data[i, columns] = 0
    } else {
      data[i, columns] = data[i, columns] / s
    }
  }
  return(data)
}

run_nnet <- function(dataset, hidden, bucket_size, verbose=TRUE) {
  result=preprocess(dataset, bucket_size)
  trainset=result$train
  testset=result$test
  n <- names(trainset[,-c(1)])
  f <- as.formula(paste("default ~", paste(n[!n %in% "y"], collapse = " + ")))
  credit <- neuralnet(f, trainset, hidden = hidden, lifesign = "minimal",
                      linear.output = FALSE, threshold = 0.1)
  if (verbose) {
    plot(credit, rep = "best")
  }

  temp_test <- subset(testset, select = c(2:ncol(testset)))
  credit.results <- compute(credit, temp_test)
  results <- data.frame(actual = testset$default, prediction = credit.results$net.result)
  diff=sum(abs(results$actual-results$prediction))
  results$prediction <- round(results$prediction)
  tbl = table(results$actual,results$prediction)

  if (verbose) {
    View(results$prediction)
    View(results)
    print(tbl)
  }
  return(list(
    accuracy=(tbl[1,1] + tbl[2,2])/sum(tbl),
    table=tbl
  ))
}

plot_bucket_size_nnet <- function(dataset) {
  all_buckets = seq(3,15,3)
  accuracies = c()
  for (bucket_size in all_buckets) {
    res = run_nnet(dataset, 5, bucket_size, verbose = FALSE)
    accuracies = c(accuracies, res$accuracy)
  }
```

```
  plot(all_bucket, accuracies, type="l")
}

plot_hidden_count_nnet <- function(dataset) {
  all_hidden = c(4,7,9,11,15)
  accuracies = c()
  for (hidden in all_hidden) {
    res = run_nnet(dataset, hidden, 15, verbose = FALSE)
    accuracies = c(accuracies, res$accuracy)
  }
  plot(all_hidden, accuracies, type="l")
}

dataset <- get_credit_dataset()
#plot_hidden_count_nnet(dataset)
#plot_bucket_size_nnet(dataset)
run_nnet(dataset, 10, 15, verbose=TRUE)
```

```
## LIMIT_BAL NEWAGE NEWBILL_AMT1 NEWBILL_AMT2 NEWBILL_AMT3 NEWBILL_AMT4 NEWBILL_AMT5 NEWBILL_AMT6 NEWPA
```




```
##
##       0   1
##   0 922 280
##   1 439 759
```

```
## $accuracy
## [1] 0.7004166667
##
## $table
##
##       0   1
##   0 922 280
##   1 439 759
```