Assignment Question 1

Preprocessing installation of libraries, assigning color palettes.

```
library(kohonen)
library(dummies)
library(ggplot2)
library(sp)
library(maptools)
library(reshape2)
library(rgeos)
library(arules)

pretty_palette <- c("#1f77b4", '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2')
coolBlueHotRed <- function(n, alpha = 1) {rainbow(n, end=4/6, alpha=alpha)[n:1]}

```

Use cor function to determine the correlation of those attributes(features).

```
> for (i in 1:45){
+   print(cor(data_raw[46],data_raw[i]))
+ }
              functionary
credit.rating  -0.1198832
              re.balanced..paid.back..a.recently.overdrawn.current.acount
credit.rating                                                   0.2642301
              FI3O.credit.score
credit.rating         0.3847463
                  gender
credit.rating 0.02131317
              X0..accounts.at.other.banks
credit.rating                 0.004716928
              credit.refused.in.past.
credit.rating              0.02745321
              years.employed
credit.rating   -0.004292185
              savings.on.other.accounts
credit.rating                 0.3165199
              self.employed.
credit.rating    0.008359808
              max..account.balance.12.months.ago
credit.rating                          0.00328374
              min..account.balance.12.months.ago
credit.rating                          0.04338131
              avrg..account.balance.12.months.ago
credit.rating                           0.04791753
              max..account.balance.11.months.ago
credit.rating                          0.02630287
```

What is worth noting down is balanced pay back at 0.264, FicoScore at 0.384, savings on other account at 0.316, minimum account balance 12 months ago at 0.0455 and average account balance 12 months ago at 0.0479.

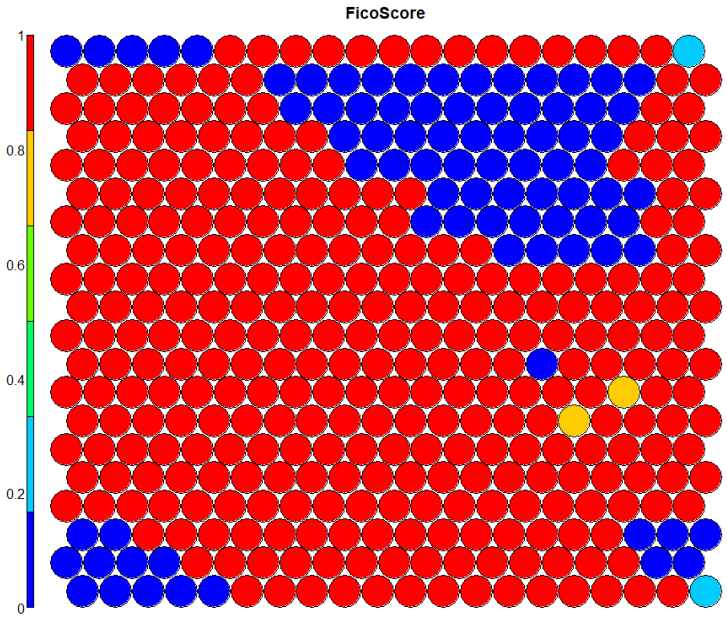
After reading in the data into a dataframe, I realize that there is minimum/maximum/average balance for each month up to 12 month. I have decided to find the mean of minimum/maximum/average balance of all the 12 months. I believe this will make the data more useful.

```
data_raw <- read.csv("./creditworthiness.csv")
head(data_raw)

MiniBalance <- (data_raw$min..account.balance.1.months.ago + data_raw$min..account.balance.2.months.ago + data_raw$min..account.balance.3.months.ago +
                  data_raw$min..account.balance.4.months.ago + data_raw$min..account.balance.5.months.ago + data_raw$min..account.balance.6.months.ago +
                  data_raw$min..account.balance.7.months.ago + data_raw$min..account.balance.8.months.ago + data_raw$min..account.balance.9.months.ago +
                  data_raw$min..account.balance.10.months.ago + data_raw$min..account.balance.11.months.ago + data_raw$min..account.balance.12.months.ago)/ 12

MaxBalance <- (data_raw$max..account.balance.1.months.ago + data_raw$max..account.balance.2.months.ago + data_raw$max..account.balance.3.months.ago +
                data_raw$max..account.balance.4.months.ago + data_raw$max..account.balance.5.months.ago + data_raw$max..account.balance.6.months.ago +
                  data_raw$max..account.balance.7.months.ago + data_raw$max..account.balance.8.months.ago + data_raw$max..account.balance.9.months.ago +
                  data_raw$max..account.balance.10.months.ago + data_raw$max..account.balance.11.months.ago + data_raw$max..account.balance.12.months.ago
                  ) / 12

AverageBalance <- (data_raw$avrg..account.balance.1.months.ago + data_raw$avrg..account.balance.2.months.ago + data_raw$avrg..account.balance.3.months.ago +
                  data_raw$avrg..account.balance.4.months.ago + data_raw$avrg..account.balance.5.months.ago + data_raw$avrg..account.balance.6.months.ago +
                  data_raw$avrg..account.balance.7.months.ago + data_raw$avrg..account.balance.8.months.ago + data_raw$avrg..account.balance.9.months.ago +
                  data_raw$avrg..account.balance.10.months.ago + data_raw$avrg..account.balance.11.months.ago + data_raw$avrg..account.balance.12.months.ago) / 12

head(MiniBalance)
head(MaxBalance)
head(AverageBalance)

data_raw$MiniBalance = MiniBalance
data_raw$MaxBalance = MaxBalance
data_raw$AverageBalance = AverageBalance
```

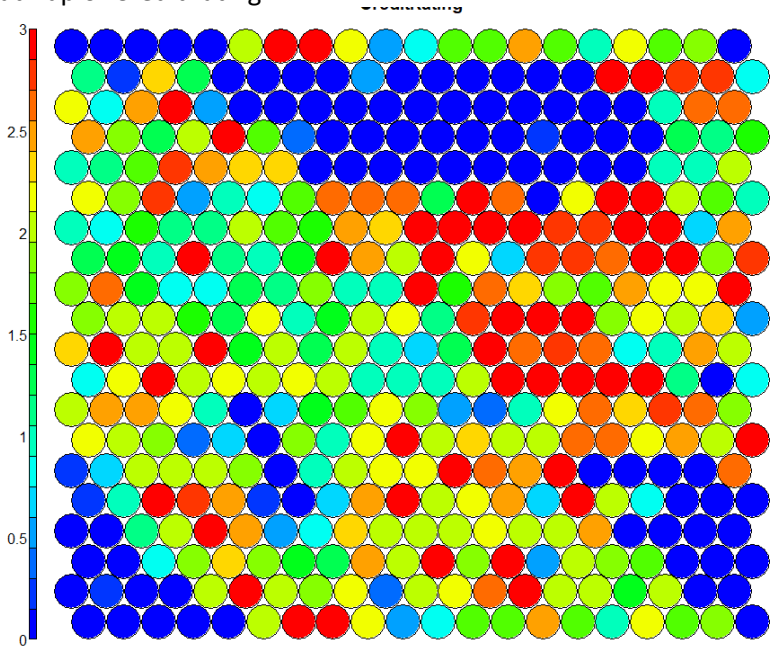
I selected the variables including the target to train the SOM. Variables has also been renamed for better understanding.

```
> head(data_train)
  functionary Payback FicoScore gender RefusedBefore years.employed MiniBalance MaxBalance AverageBalance CreditRating
1           1       1         0      0             1              3    2.833333   2.833333       3.833333            0
2           0       0         1      0             0              5    3.500000   2.666667       2.666667            0
3           0       1         0      1             0              3    2.583333   2.833333       3.750000            0
4           0       1         0      0             0              1    3.333333   2.750000       3.083333            0
5           1       1         1      0             0              1    3.416667   2.833333       3.750000            2
6           0       1         1      1             0              1    2.833333   3.250000       3.500000            2
> summary(data_train)
  functionary        Payback         FicoScore          gender      RefusedBefore    years.employed   MiniBalance
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :1.000   Min.   :1.750
 1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:1.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:2.750
 Median :0.0000   Median :1.0000   Median :1.0000   Median :0.000   Median :0.0000   Median :3.000   Median :3.000
 Mean   :0.2784   Mean   :0.8516   Mean   :0.8196   Mean   :0.494   Mean   :0.1336   Mean   :3.011   Mean   :2.998
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.:3.250
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :5.000   Max.   :4.333
   MaxBalance    AverageBalance   CreditRating
 Min.   :1.333   Min.   :1.667   Min.   :0.00
 1st Qu.:2.750   1st Qu.:2.750   1st Qu.:1.00
 Median :3.000   Median :3.000   Median :2.00
 Mean   :2.999   Mean   :3.011   Mean   :1.58
 3rd Qu.:3.250   3rd Qu.:3.250   3rd Qu.:2.00
 Max.   :4.250   Max.   :4.250   Max.   :3.00
> |
```

One of the interesting pairs of attributes is Fiscoscore and CreditRating.
Heatmap of FicoScore

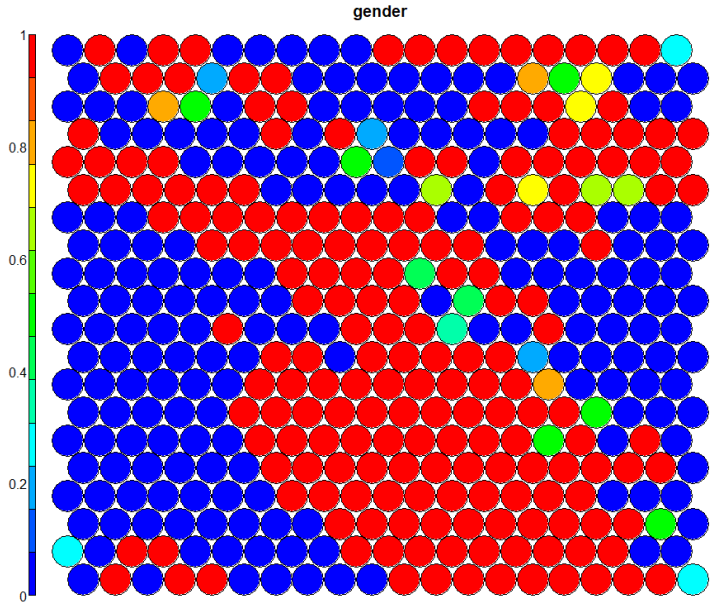


Heatmap of CreditRating

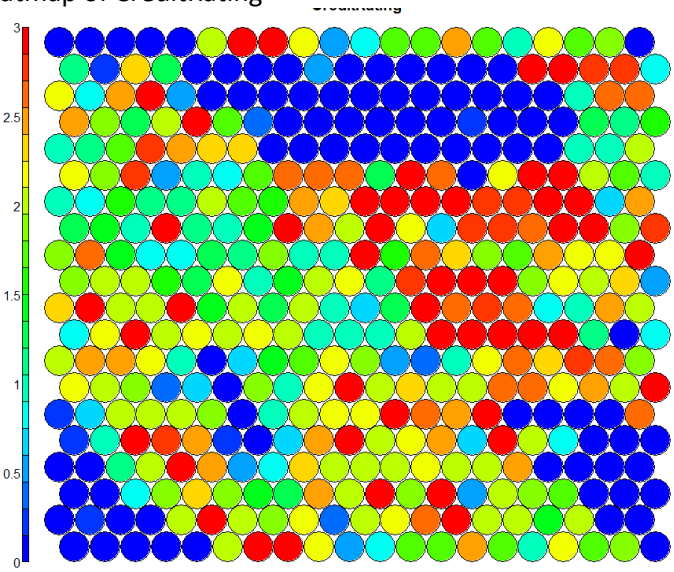


With these 2 heatmaps, I can clearly identify similar four clusters of blue from FicoScore and CreditRating. It seems that there's a very high confidence that people with poor Ficoscore didn't get any credit rating. Fico score are one brand of credit score. It is also based on the data in credit reports. I believe there should be a correlation to the target credit rating.

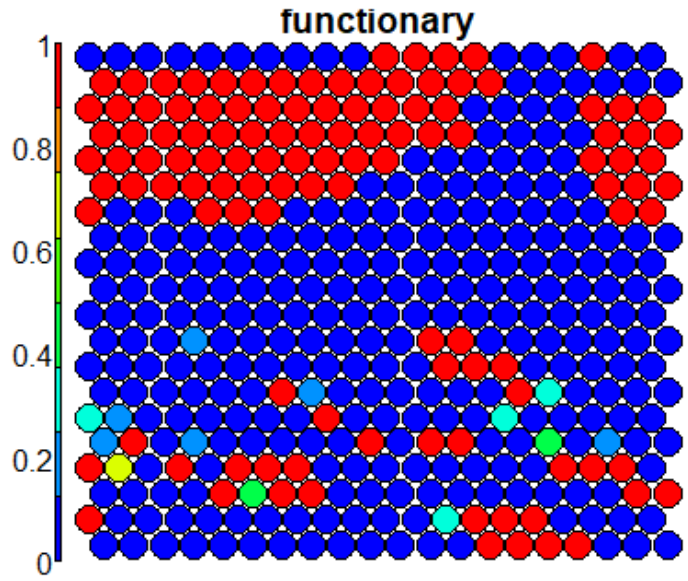
Another interesting pair is gender and and CreditingRating
Heatmap of gender

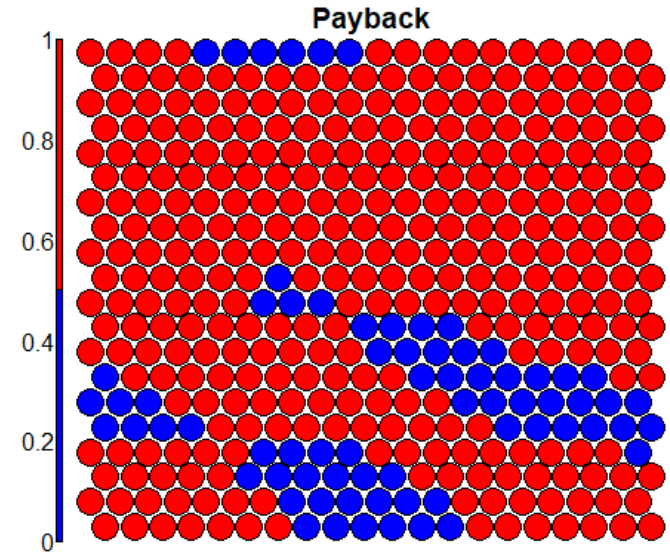


Heatmap of CreditRating

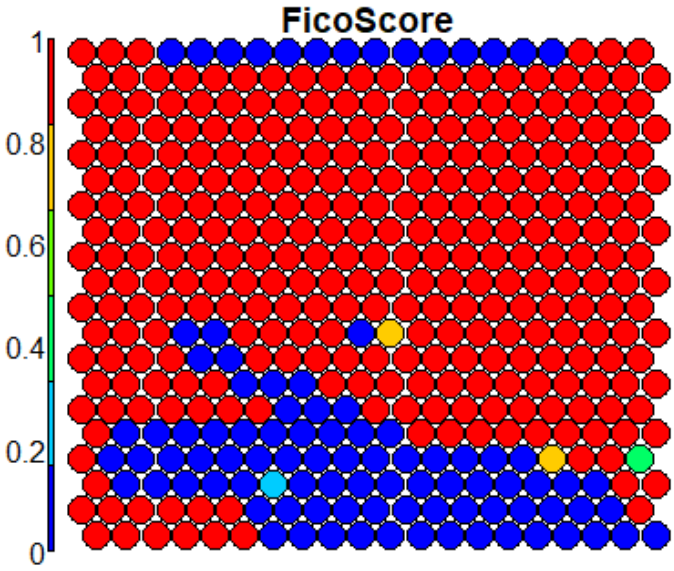


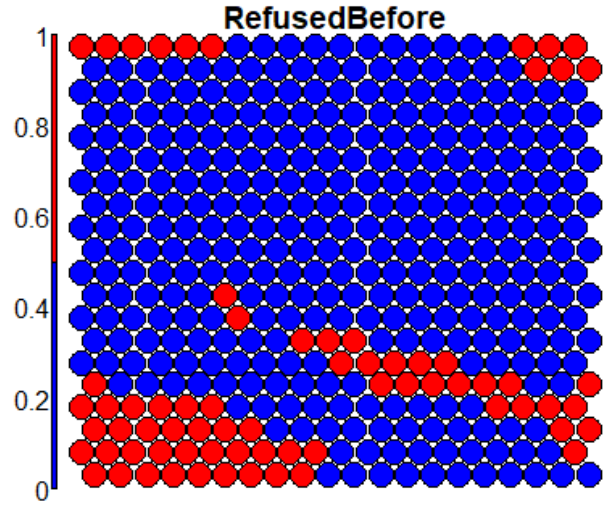
From these 2 heatmaps, I can actually see that most females (red circular) in gender has a credit rating. I can spot 3 clusters of blue in credit rating(no credit) that is closely correlated to the 3 clusters of blue in gender(male).





From the 2 heatmaps above, it is also interesting to see that the red clusters, those that were functionary(red) did pay back the balance of a recently overdrawn current account (Red on Payback). This tell me that people who are functionary have the highest chance to pay back the balance.





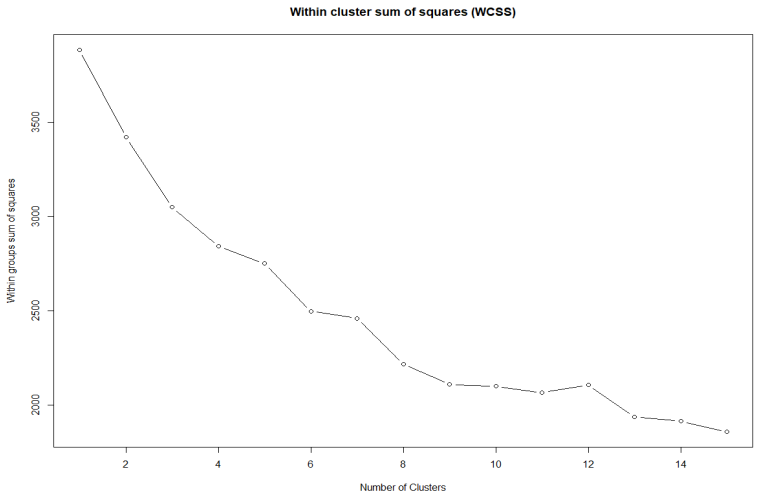
From the 2 heatmaps of FicoScore and RefusedBefore.
The 2 clusters of blue from Ficoscore also have a correlation to the blue cluster in RefusedBefore.
This actually also implies that most people with poor ficoscore also had their credit refused in the past.

Another interesting correlation is that people who are functionary has the best CreditRating = A.

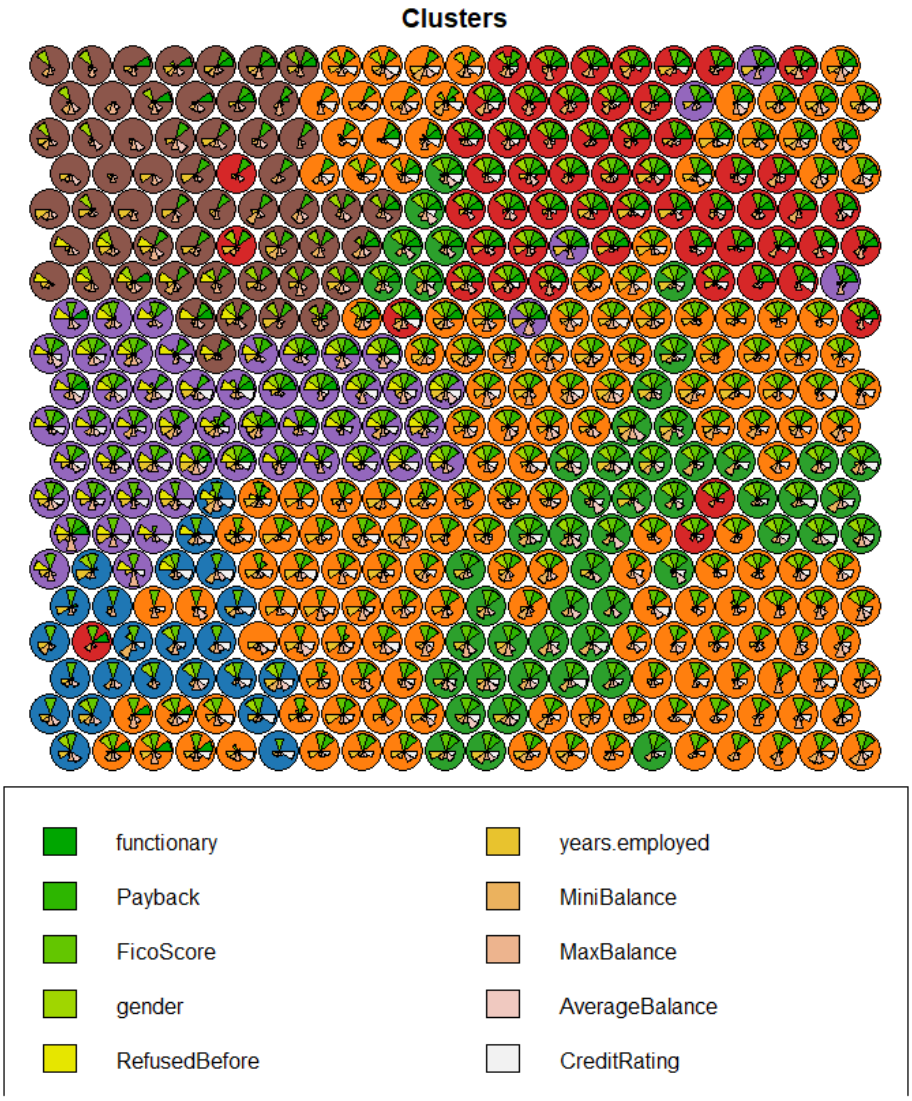
```
> rules <- apriori(data_train, parameter=list(minlen=2, supp=0.01, conf = 0.65), appearance = list(rhs=c("CreditRat
 = list(verbose=F))
> inspect(rules)
    lhs                                rhs              support confidence coverage     lift count
[1] {functionary=1,
     Payback=1,
     FicoScore=1,
     AverageBalance=2.83333333333333} => {CreditRating=1}  0.0104  0.6500000  0.0160 3.364389    26
[2] {functionary=1,
     Payback=1,
     FicoScore=1,
     RefusedBefore=0,
     AverageBalance=2.83333333333333} => {CreditRating=1}  0.0100  0.6756757  0.0148 3.497286    25
[3] {functionary=1,
     Payback=1,
     FicoScore=1,
     gender=1,
     years.employed=4}                => {CreditRating=1}  0.0152  0.6551724  0.0232 3.391162    38
[4] {functionary=1,
     FicoScore=1,
     gender=1,
     RefusedBefore=0,
     years.employed=4}                => {CreditRating=1}  0.0152  0.6666667  0.0228 3.450656    38
[5] {functionary=1,
     Payback=1,
     gender=1,
     RefusedBefore=0,
     years.employed=4}                => {CreditRating=1}  0.0152  0.6551724  0.0232 3.391162    38
[6] {functionary=1,
     Payback=1,
     FicoScore=1,
     gender=1,
     RefusedBefore=0,
     years.employed=4}                => {CreditRating=1}  0.0152  0.6909091  0.0220 3.576134    38
> |
```

When using the brute force approach for mining association rules, we can see that most people who has the best credit rating is functionary. This is supported by at least 0.01 and confidence of 0.65.
It is also worth to state that among the six cases, four has been employed for more than 5 years.

Overall, the most interesting five attributes to me was functionary , Payback , Ficoscore , gender and RefusedBefore.
Before the experiment, I thought that AverageBalance will have a strong correlation with CreditRating or Functionary. I couldn't find any correlation regarding the balances with CreditRating or functionary. The balances variables were also not cluster contiguously on the map, making it hard to find similarities and making connections.



We can also see that when clustering, the sum of squares error decreases as there is more clusters. This also shows how identical it is within the clusters as it increases.



After using hierarchical clustering to cluster the codebook vectors. We can see the plot with codes. We can see that the clusters found are contiguous on the map surface. Red takes up the top right. Brown takes up top left. Purple takes up middle left and blue takes up bottom left. Orange however seems to be less contiguous on the map. The data is well mapped into distinct clusters. This is an indication that the learning problem is a simple one. The SOM was able to find correlation between data points.

Assignment Question 2

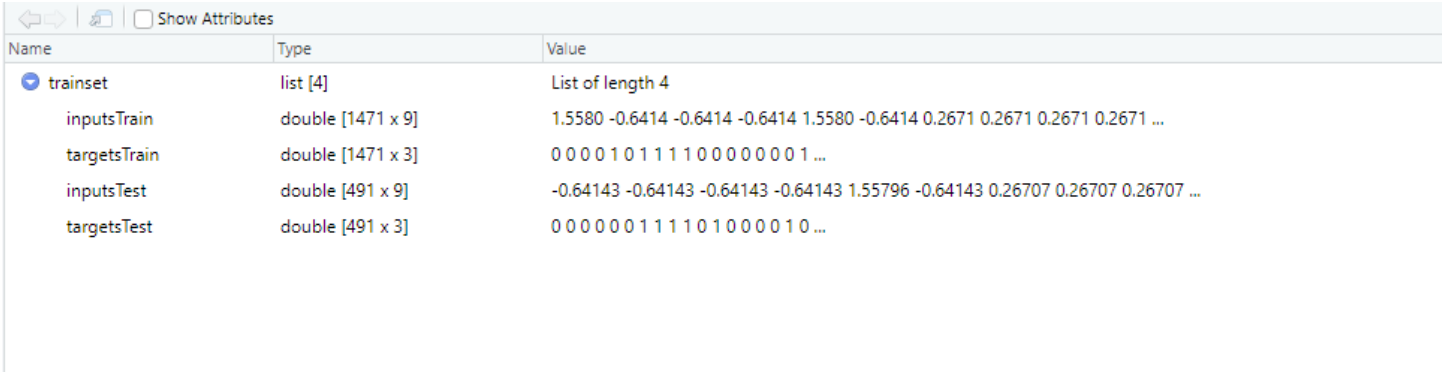
I will start by analyzing the prediction capabilities of the MLP.
Unknown data (where target class is 0) and known data are first separated into dataframes.
unknownData:

	functionary	Payback	FicoScore	gender	RefusedBefore	years.employed	MiniBalance	MaxBalance	AverageBalance	CreditRating
1	1	1	0	0	1	3	2.833333	2.833333	3.833333	0
2	0	0	1	0	0	5	3.500000	2.666667	2.666667	0
3	0	1	0	1	0	3	2.583333	2.833333	3.750000	0
4	0	1	0	0	0	1	3.333333	2.750000	3.083333	0
11	0	1	1	1	0	3	3.666667	3.583333	2.500000	0
13	1	0	1	0	0	5	2.750000	2.916667	2.916667	0
16	0	0	1	0	0	5	3.500000	2.333333	2.750000	0
18	0	1	0	0	0	4	2.500000	2.666667	2.916667	0
24	1	0	1	1	0	4	2.750000	3.666667	3.166667	0
27	0	1	0	0	0	4	2.916667	2.833333	2.583333	0
28	0	0	1	0	1	3	2.916667	2.416667	2.500000	0

Known data:

	functionary	Payback	FicoScore	gender	RefusedBefore	years.employed	MiniBalance	MaxBalance	AverageBalance	CreditRating
5	1	1	1	0	0	1	3.416667	2.833333	3.750000	2
6	0	1	1	1	0	1	2.833333	3.250000	3.500000	2
7	0	1	1	0	0	2	3.416667	2.666667	2.250000	2
8	0	1	1	1	0	2	2.500000	3.916667	3.666667	2
9	1	1	1	1	0	5	3.250000	2.750000	3.166667	1
10	0	1	1	0	0	5	2.416667	3.416667	3.000000	3
12	0	1	1	1	0	4	3.083333	2.916667	2.583333	3
14	0	1	1	1	0	1	2.666667	3.000000	3.000000	1
15	0	0	1	0	0	5	3.000000	3.166667	3.500000	3
17	0	1	1	1	0	1	3.583333	3.166667	3.416667	2
19	1	1	1	0	0	3	3.333333	3.500000	2.583333	2

Values are separated from the target. I spilt the dataset into training and testing set at 0.25 ratio.
2/3 of the data will be to training, 1/3 will be for testing.



The layers I have set for the MLP is 15, with the learning rate of 0.01 and 150 iterations.
Confusion matrix for prediction of testing and training:

```
trainset <- splitForTrainingAndTest(trainValues, trainTargets, ratio=0.25)
trainset <- normTrainingAndTestSet(trainset)

model <- mlp(trainset$inputsTrain, trainset$targetsTrain, size=15, learnFuncParams=c(0.01), maxit=150, inputsTest=trainset$inputsTest,
             targetsTest=trainset$targetsTest)
predictTestSet <- predict(model,trainset$inputsTest)

confusionMatrix(trainset$targetsTrain,fitted.values(model))
confusionMatrix(trainset$targetsTest,predictTestSet)
```

```
> confusionMatrix(trainset$targetsTrain,fitted.values(model))
        predictions
targets   1   2   3
      1 209 136  11
      2 121 557  53
      3  38 203 143
> confusionMatrix(trainset$targetsTest,predictTestSet)
        predictions
targets   1   2   3
      1  75  48   4
      2  41 185  13
      3  18  69  38
>
```

Accuracy = (75 + 185 + 38) / (75 + 48 + 4 + 41 + 185 + 13 + 18 + 69 + 38) x 100
= 298 / 491 x 100 = 60.69

Looking at the confusion matrix for the training dataset, I can see that the prediction for 1 is good. There are 209 datapoints where the model got the prediction right for 1. 2 Seems to perform better. There are 557 datapoints where the model got the prediction right for 2. However this may not be the case for 3. Although the target is 3, most of the predictions made by the model on the value turns out to be 2. The data were not able to form the diagonal line. The target being 3, 203 predictions were made on 2 and 143 predictions were made on 3. This means that the model is inaccurate on predicting the target 3. The root cause for poor model accuracy is underfitting. I believe there is underfitting because there is not enough flexibility due to me transforming the minimum/maximum/average balance for each month up to 12 month to a mean.

Looking at the confusion matrix for the testing dataset, the predictions share the same pattern as the training data. This shows that it is perform well on the testing set. I just need to get a good model fit for the training set.

```
> summary(model)
SNNS network definition file V1.4-3D
generated at Fri Feb 11 19:43:44 2022

network name : RSNNS_untitled
source files :
no. of units : 17
no. of connections : 60
no. of unit types : 0
no. of site types : 0


learning function : Std_Backpropagation
update function   : Topological_Order


unit default section :

act      | bias     | st | subnet | layer | act func     | out func
---------|----------|----|--------|-------|--------------|-------------
 0.00000 |  0.00000 | i  |      0 |     1 | Act_Logistic | Out_Identity
---------|----------|----|--------|-------|--------------|-------------


unit definition section :

no. | typeName | unitName   | act      | bias     | st | position | act func     | out func | sites
----|----------|------------|----------|----------|----|----------|--------------|----------|-------
  1 |          | Input_1    | -0.64143 | -0.17973 | i  | 1,0,0    | Act_Identity |          |
  2 |          | Input_2    |  0.26707 | -0.12263 | i  | 2,0,0    | Act_Identity |          |
  3 |          | Input_3    |  0.26561 |  0.02631 | i  | 3,0,0    | Act_Identity |          |
  4 |          | Input_4    | -0.96296 | -0.13957 | i  | 4,0,0    | Act_Identity |          |
  5 |          | Input_5    |  2.79340 | -0.00051 | i  | 5,0,0    | Act_Identity |          |
  6 |          | Input_6    |  1.44222 |  0.13498 | i  | 6,0,0    | Act_Identity |          |
  7 |          | Input_7    | -0.38551 | -0.27564 | i  | 7,0,0    | Act_Identity |          |
  8 |          | Input_8    | -0.60630 |  0.23989 | i  | 8,0,0    | Act_Identity |          |
  9 |          | Input_9    | -1.50044 |  0.21106 | i  | 9,0,0    | Act_Identity |          |
 10 |          | Hidden_2_1 |  0.42868 |  0.56318 | h  | 1,2,0    |              |          |
 11 |          | Hidden_2_2 |  0.77340 |  0.08654 | h  | 2,2,0    |              |          |
 12 |          | Hidden_2_3 |  0.94124 |  0.41143 | h  | 3,2,0    |              |          |
 13 |          | Hidden_2_4 |  0.87246 |  0.25722 | h  | 4,2,0    |              |          |
 14 |          | Hidden_2_5 |  0.38972 | -0.20581 | h  | 5,2,0    |              |          |
 15 |          | Output_1   |  0.04932 | -0.09387 | o  | 1,4,0    |              |          |
 16 |          | Output_2   |  0.54116 | -1.21171 | o  | 2,4,0    |              |          |
 17 |          | Output_3   |  0.47023 | -0.53049 | o  | 3,4,0    |              |          |
----|----------|------------|----------|----------|----|----------|--------------|----------|-------


connection definition section :

target | site | source:weight
-------|------|---------------------------------------------------------------------------------------------------------------------------
    10 |      |  9: 0.08423,  8:-0.04067,  7:-0.23035,  6:-0.13071,  5:-0.32517,  4:-0.00861,  3: 1.20357,  2: 0.55029,  1: 0.33609
    11 |      |  9: 0.05579,  8:-0.05387,  7: 0.26002,  6:-0.13264,  5:-0.12031,  4:-0.37782,  3: 0.88034,  2: 0.86342,  1:-1.54574
    12 |      |  9:-0.06359,  8: 0.18360,  7: 0.01729,  6: 0.13924,  5: 0.56788,  4:-0.29759,  3:-0.65260,  2:-0.60418,  1:-1.00695
    13 |      |  9:-0.00578,  8: 0.00040,  7:-0.09352,  6:-0.28130,  5: 0.59562,  4:-0.29682,  3:-0.80777,  2:-0.33457,  1:-0.59427
    14 |      |  9: 0.21691,  8:-0.20329,  7: 0.04388,  6: 0.09942,  5:-0.21302,  4:-0.01301,  3: 0.59362,  2:-0.13887,  1:-0.45990
    15 |      | 14:-0.19110, 13:-1.38426, 12:-1.53753, 11:-0.92180, 10: 1.34685
    16 |      | 14: 0.55037, 13:-0.59583, 12: 0.25533, 11: 1.74148, 10: 0.22137
    17 |      | 14:-0.34900, 13: 1.05127, 12: 1.01722, 11:-0.67877, 10:-1.87181
-------|------|---------------------------------------------------------------------------------------------------------------------------
> 
```

```
> model
Class: mlp->rsnns
Number of inputs: 9
Number of outputs: 3
Maximal iterations: 120
Initialization function: Randomize_Weights
Initialization function parameters: -0.3 0.3
Learning function: Std_Backpropagation
Learning function parameters: 0.008
Update function:Topological_Order
Update function parameters: 0
Patterns are shuffled internally: TRUE
Compute error in every iteration: TRUE
Architecture Parameters:
$size
[1] 5

All members of model:
 [1] "nInputs"               "maxit"                 "initFunc"              "initFuncParams"        "learnFunc"
 [6] "learnFuncParams"       "updateFunc"            "updateFuncParams"      "shufflePatterns"       "computeIterativeError"
[11] "snnsObject"            "archParams"            "IterativeFitError"     "IterativeTestError"    "fitted.values"
[16] "fittedTestValues"      "nOutputs"
> |
```

```
> weightMatrix(model)
          Input_1 Input_2 Input_3 Input_4 Input_5 Input_6 Input_7 Input_8 Input_9   Hidden_2_1  Hidden_2_2  Hidden_2_3
Input_1         0       0       0       0       0       0       0       0       0  0.336093903 -1.54573631 -1.00695240
Input_2         0       0       0       0       0       0       0       0       0  0.550288916  0.86341709 -0.60418433
Input_3         0       0       0       0       0       0       0       0       0  1.203565240  0.88033879 -0.65259719
Input_4         0       0       0       0       0       0       0       0       0 -0.008609708 -0.37782457 -0.29759103
Input_5         0       0       0       0       0       0       0       0       0 -0.325174063 -0.12031101  0.56787729
Input_6         0       0       0       0       0       0       0       0       0 -0.130707368 -0.13264373  0.13923609
Input_7         0       0       0       0       0       0       0       0       0 -0.230349243  0.26002035  0.01728687
Input_8         0       0       0       0       0       0       0       0       0 -0.040666271 -0.05387434  0.18360458
Input_9         0       0       0       0       0       0       0       0       0  0.084231131  0.05578800 -0.06358504
Hidden_2_1      0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Hidden_2_2      0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Hidden_2_3      0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Hidden_2_4      0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Hidden_2_5      0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Output_1        0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Output_2        0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
Output_3        0       0       0       0       0       0       0       0       0  0.000000000  0.00000000  0.00000000
              Hidden_2_4  Hidden_2_5   Output_1   Output_2   Output_3
Input_1    -0.5942703485 -0.45990297  0.0000000  0.0000000  0.0000000
Input_2    -0.3345656395 -0.13887058  0.0000000  0.0000000  0.0000000
Input_3    -0.8077721000  0.59362423  0.0000000  0.0000000  0.0000000
Input_4    -0.2968214750 -0.01300607  0.0000000  0.0000000  0.0000000
Input_5     0.5956211090 -0.21302481  0.0000000  0.0000000  0.0000000
Input_6    -0.2812950909  0.09942237  0.0000000  0.0000000  0.0000000
Input_7    -0.0935163274  0.04387620  0.0000000  0.0000000  0.0000000
Input_8     0.0003989649 -0.20328552  0.0000000  0.0000000  0.0000000
Input_9    -0.0057806526  0.21691327  0.0000000  0.0000000  0.0000000
Hidden_2_1  0.0000000000  0.00000000  1.3468496  0.2213687 -1.8718104
Hidden_2_2  0.0000000000  0.00000000 -0.9217972  1.7414848 -0.6787678
Hidden_2_3  0.0000000000  0.00000000 -1.5375284  0.2553279  1.0172211
Hidden_2_4  0.0000000000  0.00000000 -1.3842570 -0.5958285  1.0512655
Hidden_2_5  0.0000000000  0.00000000 -0.1910990  0.5503687 -0.3489971
Output_1    0.0000000000  0.00000000  0.0000000  0.0000000  0.0000000
Output_2    0.0000000000  0.00000000  0.0000000  0.0000000  0.0000000
Output_3    0.0000000000  0.00000000  0.0000000  0.0000000  0.0000000
> |
```

```
> extractNetInfo(model)
$infoHeader
                 name                value
1       no. of units                   17
2 no. of connections                   60
3  no. of unit types                    0
4  no. of site types                    0
5  learning function Std_Backpropagation
6    update function   Topological_Order
```

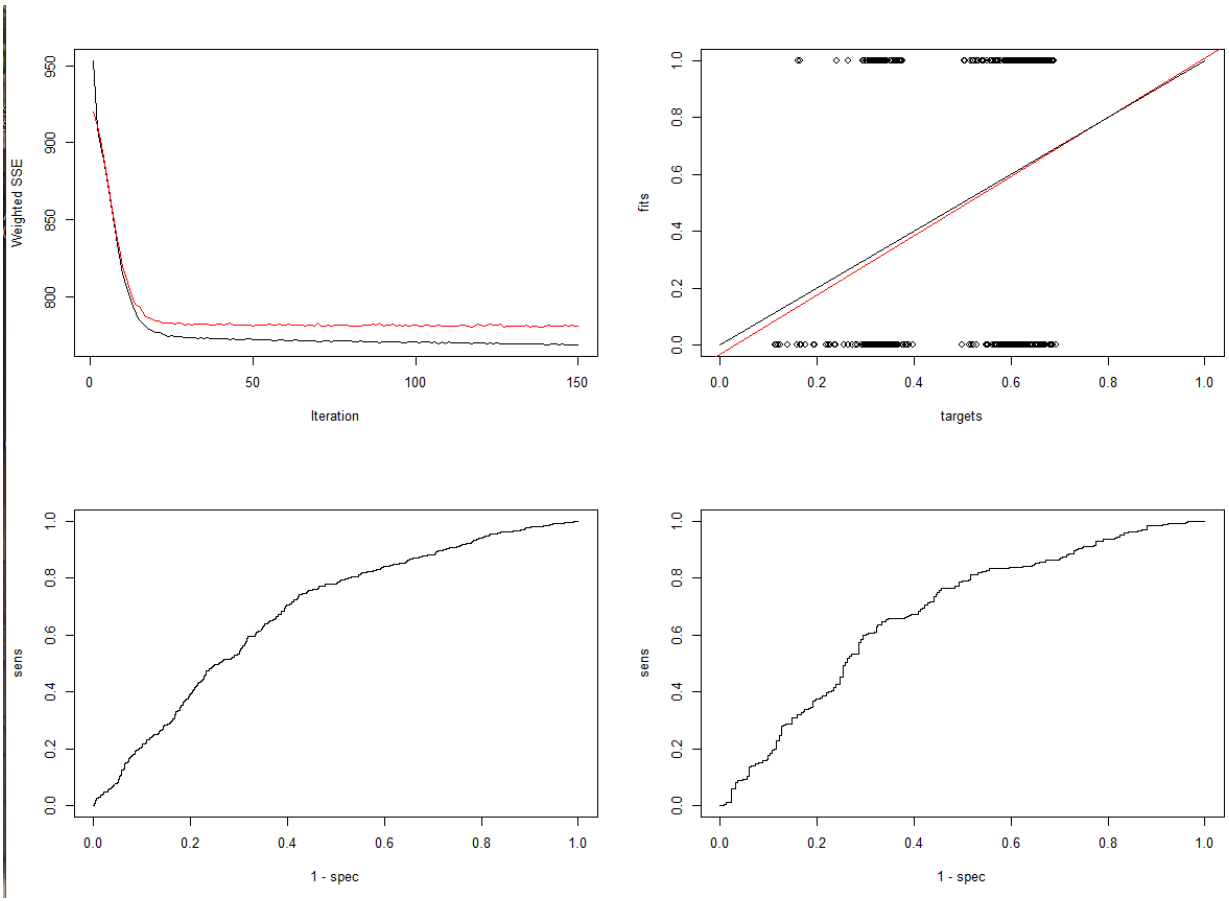
$unitDefinitions

	unitNo	unitName	unitAct	unitBias	type	posX	posY	posZ	actFunc	outFunc	sites
1	1	Input_1	-0.64142680	-0.17972916	UNIT_INPUT	1	0	0	Act_Identity	Out_Identity	
2	2	Input_2	0.26707307	-0.12263377	UNIT_INPUT	2	0	0	Act_Identity	Out_Identity	
3	3	Input_3	0.26561025	0.02631465	UNIT_INPUT	3	0	0	Act_Identity	Out_Identity	
4	4	Input_4	-0.96295655	-0.13956544	UNIT_INPUT	4	0	0	Act_Identity	Out_Identity	
5	5	Input_5	2.79339838	-0.00051108	UNIT_INPUT	5	0	0	Act_Identity	Out_Identity	
6	6	Input_6	1.44222105	0.13498172	UNIT_INPUT	6	0	0	Act_Identity	Out_Identity	
7	7	Input_7	-0.38550630	-0.27564326	UNIT_INPUT	7	0	0	Act_Identity	Out_Identity	
8	8	Input_8	-0.60629767	0.23988950	UNIT_INPUT	8	0	0	Act_Identity	Out_Identity	
9	9	Input_9	-1.50043845	0.21106237	UNIT_INPUT	9	0	0	Act_Identity	Out_Identity	
10	10	Hidden_2_1	0.42868114	0.56318432	UNIT_HIDDEN	1	2	0	Act_Logistic	Out_Identity	
11	11	Hidden_2_2	0.77339977	0.08654139	UNIT_HIDDEN	2	2	0	Act_Logistic	Out_Identity	
12	12	Hidden_2_3	0.94123912	0.41142637	UNIT_HIDDEN	3	2	0	Act_Logistic	Out_Identity	
13	13	Hidden_2_4	0.87246400	0.25722110	UNIT_HIDDEN	4	2	0	Act_Logistic	Out_Identity	
14	14	Hidden_2_5	0.38971603	-0.20580645	UNIT_HIDDEN	5	2	0	Act_Logistic	Out_Identity	
15	15	Output_1	0.04932265	-0.09387177	UNIT_OUTPUT	1	4	0	Act_Logistic	Out_Identity	
16	16	Output_2	0.54116160	-1.21171379	UNIT_OUTPUT	2	4	0	Act_Logistic	Out_Identity	
17	17	Output_3	0.47022900	-0.53048617	UNIT_OUTPUT	3	4	0	Act_Logistic	Out_Identity	

$fullWeightMatrix

	Input_1	Input_2	Input_3	Input_4	Input_5	Input_6	Input_7	Input_8	Input_9	Hidden_2_1	Hidden_2_2	Hidden_2_3
Input_1	0	0	0	0	0	0	0	0	0	0.336093903	-1.54573631	-1.00695240
Input_2	0	0	0	0	0	0	0	0	0	0.550288916	0.86341709	-0.60418433
Input_3	0	0	0	0	0	0	0	0	0	1.203565240	0.88033879	-0.65259719
Input_4	0	0	0	0	0	0	0	0	0	-0.008609708	-0.37782457	-0.29759103
Input_5	0	0	0	0	0	0	0	0	0	-0.325174063	-0.12031101	0.56787729
Input_6	0	0	0	0	0	0	0	0	0	-0.130707368	-0.13264373	0.13923609
Input_7	0	0	0	0	0	0	0	0	0	-0.230349243	0.26002035	0.01728687
Input_8	0	0	0	0	0	0	0	0	0	-0.040666271	-0.05387434	0.18360458
Input_9	0	0	0	0	0	0	0	0	0	0.084231131	0.05578800	-0.06358504
Hidden_2_1	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Hidden_2_2	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Hidden_2_3	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Hidden_2_4	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Hidden_2_5	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Output_1	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Output_2	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000
Output_3	0	0	0	0	0	0	0	0	0	0.000000000	0.00000000	0.00000000

	Hidden_2_4	Hidden_2_5	Output_1	Output_2	Output_3
Input_1	-0.5942703485	-0.45990297	0.0000000	0.0000000	0.0000000
Input_2	-0.3345656395	-0.13887058	0.0000000	0.0000000	0.0000000
Input_3	-0.8077721000	0.59362423	0.0000000	0.0000000	0.0000000
Input_4	-0.2968214750	-0.01300607	0.0000000	0.0000000	0.0000000
Input_5	0.5956211090	-0.21302481	0.0000000	0.0000000	0.0000000
Input_6	-0.2812950909	0.09942237	0.0000000	0.0000000	0.0000000
Input_7	-0.0935163274	0.04387620	0.0000000	0.0000000	0.0000000
Input_8	0.0003989649	-0.20328552	0.0000000	0.0000000	0.0000000
Input_9	-0.0057806526	0.21691327	0.0000000	0.0000000	0.0000000
Hidden_2_1	0.0000000000	0.00000000	1.3468496	0.2213687	-1.8718104
Hidden_2_2	0.0000000000	0.00000000	-0.9217972	1.7414848	-0.6787678
Hidden_2_3	0.0000000000	0.00000000	-1.5375284	0.2553279	1.0172211
Hidden_2_4	0.0000000000	0.00000000	-1.3842570	-0.5958285	1.0512655
Hidden_2_5	0.0000000000	0.00000000	-0.1910990	0.5503687	-0.3489971
Output_1	0.0000000000	0.00000000	0.0000000	0.0000000	0.0000000
Output_2	0.0000000000	0.00000000	0.0000000	0.0000000	0.0000000
Output_3	0.0000000000	0.00000000	0.0000000	0.0000000	0.0000000



Top Left:
The iteration and weight square sum error plot seems to be good. As the number of iteration increases, the error decreases. We can see a drastic decrease in errors for the first 20 iteration. There seems to be no problem with overfitting. The plot is quite ideal as both the training and testing both decreases.

Top Right:
The regression plot is also very ideal as the red line is close to the black line. They are very close to each other and they overlap at around 0.6 fits and targets.

Bottom Left: The prediction model's area under ROC. It can be interpreted that the model is neither underfitting nor overfitting the dataset. Hence, the accuracy of implementing the model on the test set will be less likely to be reduced.

Bottom, right:
The plot shows the Area under the ROC Curve based on test set. It can be interpreted that the predictions made on the test set was better than the predictions made on the validation set. Hence, by not overtraining the prediction model to the dataset, it made the model more flexible inn generalizing towards unseen data.

The model seems to have poor performance on the training data for the target 3. There is error on the target 3 on both the training and test set. My solution to this is to increase the flexibility of the data. Instead of merging the minimum/maximum/average balance for each month up to 12 month to a mean, I have decided to use the first 35 variables as attributes to finding the target. I will lower the capacity of the model so the model needs to focus on the relevant patterns in training data, resulting in better generalization. I stabilize the structure by decreasing the layer to 11 while increasing the learning rate to 0.05. I have also increased the testing set to from 25% to 35%.

```
#split dataset into traing and test set
trainset <- splitForTrainingAndTest(trainValues, trainTargets, ratio=0.35)
trainset <- normTrainingAndTestSet(trainset)

model <- mlp(trainset$inputsTrain, trainset$targetsTrain, size=11, learnFuncParams=c(0.05), maxit=150, inputsTest=trainset$inputsTest,
             targetsTest=trainset$targetsTest)
predictTestSet <- predict(model,trainset$inputsTest)

confusionMatrix(trainset$targetsTrain,fitted.values(model))
confusionMatrix(trainset$targetsTest,predictTestSet)
```

```
> confusionMatrix(trainset$targetsTrain,fitted.values(model))
        predictions
targets   1   2   3
      1 233  86  37
      2  55 642  34
      3  39  57 288
> confusionMatrix(trainset$targetsTest,predictTestSet)
        predictions
targets   1   2   3
      1  67  48  12
      2  48 153  38
      3  20  52  53
```

Accuracy = (67 + 153 + 53) / (67 + 48 + 12 + 48 + 153 + 38 + 20 + 52 + 53) x 100
273 / 491 x 100 = 55.6.

Accuracy was lower than before, but model was able to predict target 3 better.