# Assignment 4.1 [Python & R]

## University of San Diego

## ADS 502

## Dingyi Duan

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
%load_ext rpy2.ipython
```

#### Python Packages

In [3]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.tools.tools as stattools

from scipy import stats
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

#### R Package

In [4]:
```
%%R
library(car)
library(C50)
library(nnet)
library(e1071)
library(caret)
library(rpart)
library(readr)
library(corrplot)
library(rpart.plot)
library(randomForest)
library(NeuralNetTools)
```

```
R[write to console]: Loading required package: carData

R[write to console]: Loading required package: lattice

R[write to console]: Loading required package: ggplot2

R[write to console]: corrplot 0.90 loaded

R[write to console]: randomForest 4.6-14

R[write to console]: Type rfNews() to see new features/changes/bug fixes.

R[write to console]:
Attaching package: 'randomForest'


R[write to console]: The following object is masked from 'package:ggplot2':

    margin
```

**20. Consider the XOR problem where there are four training points: (1, 1, −), (1, 0, +), (0, 1, +), (0, 0, −). Transform the data into the following feature space: φ=(1, 2x1, 2x2, 2x1x2, x12, x22). Find the maximum margin linear decision boundary in the transformed space.**

In [5]:
```
# This will be done using R
```

In [6]:
```
%%R
# Set up the matrix for four training matrix
training_pts <- data.frame(matrix(c(1,1,0,0,1,0,1,0), nrow = 4))
colnames(training_pts) <- c("factor1", "factor2")
```

In [7]:
```
%%R
# 6D mapping
x <- c("x1", "x2", "x3", "x4", "x5", "x6")
```

In [8]:
```
%%R
# Create a df for the map
df <- data.frame(matrix(ncol = 6, nrow = 4))
colnames(df) <- x
```

In [9]:
```
%%R
# Formulate the matrix: The hyperplane formula -> label = w1*x1 + w2*x2 + w3*x3 +
df$x1 <- 1
df$x2 <- sqrt(2)*training_pts$factor1
df$x3 <- sqrt(2)*training_pts$factor2
df$x4 <- sqrt(2)*training_pts$factor1 * training_pts$factor2
df$x5 <- training_pts$factor1 ^ 2
df$x6 <- training_pts$factor2 ^ 2
df$label <- c(-1, 1, 1, -1)
```

In [10]:
```
%%R
# Use Support Vector Machine model
svm_model <- svm(label ~ x1 + x2 + x3 + x4 + x5 + x6, df,
                 type='C-classification',
                 kernal='linear', scale=F)
```

In [11]:
```
%%R
# Constant b
b <- svm_model$rho
b
```

```
[1] 0.03382507
```

In [12]:
```
%%R
# Output for coefficients w
w <- t(svm_model$coefs) %*% svm_model$SV
```

In [13]:
```
%%R
w
```

```
     x1 x2 x3       x4 x5 x6
[1,]  0  0  0 1.414214  0  0
```

x4 is our 4th element in the map which is x1*x2, so our maximum margin linear decision boundary is x1x2.

**For the following exercises, work with the clothing_sales_training and clothing_sales_test data sets. Use either Python or R to solve each problem.**

**13. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.**

**Python**

In [14]:
```
clothing_sales_train_py = pd.read_csv("D:/2021-Spring-textbooks/ADS-502/Website D
clothing_sales_test_py = pd.read_csv("D:/2021-Spring-textbooks/ADS-502/Website Da
```

In [15]:
```
clothing_sales_train_py.head()
```

Out[15]:

| | CC | Days | Web | Sales per Visit |
|---|---|---|---|---|
| **0** | 0 | 333.0 | 0 | 184.230000 |
| **1** | 0 | 171.5 | 0 | 38.500000 |
| **2** | 0 | 213.0 | 0 | 150.326667 |
| **3** | 1 | 71.4 | 1 | 104.240000 |
| **4** | 1 | 145.0 | 0 | 782.080000 |

In [16]:
```
clothing_sales_test_py.head()
```

Out[16]:

| | CC | Days | Web | Sales per Visit |
|---|---|---|---|---|
| **0** | 1 | 174.00 | 0 | 64.5000 |
| **1** | 1 | 87.62 | 0 | 105.7575 |
| **2** | 0 | 49.00 | 0 | 87.4400 |
| **3** | 0 | 72.50 | 0 | 60.0000 |
| **4** | 0 | 264.00 | 0 | 318.5000 |

In [17]:
```
# Seprate the predictors and response variable as X and y // Subsetting using pd.
X_train_py = pd.DataFrame(clothing_sales_train_py[['Days', 'Web']])
y_train_py = pd.DataFrame(clothing_sales_train_py[['CC']])
```

In [18]:
```
# Add a constant to the X data frame in order to include a constant term in our r
X_train_py = sm.add_constant(X_train_py)
```

In [19]:
```
# Run the logistic model
logreg_py = sm.Logit(y_train_py, X_train_py).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.655955
         Iterations 5
```

In [20]:
```
# Summary of the model
logreg_py.summary2()
```

Out[20]:

| | | | |
|---|---|---|---|
| Model: | Logit | Pseudo R-squared: | 0.053 |
| Dependent Variable: | CC | AIC: | 1909.5825 |
| Date: | 2021-07-26 01:12 | BIC: | 1925.4226 |
| No. Observations: | 1451 | Log-Likelihood: | -951.79 |
| Df Model: | 2 | LL-Null: | -1004.9 |
| Df Residuals: | 1448 | LLR p-value: | 8.3668e-24 |
| Converged: | 1.0000 | Scale: | 1.0000 |
| No. Iterations: | 5.0000 | | |

| | Coef. | Std.Err. | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 0.4962 | 0.0887 | 5.5968 | 0.0000 | 0.3224 | 0.6699 |
| **Days** | -0.0037 | 0.0004 | -8.4491 | 0.0000 | -0.0046 | -0.0028 |
| **Web** | 1.2537 | 0.3307 | 3.7914 | 0.0001 | 0.6056 | 1.9018 |

### R

In [21]:
```
%%R
clothing_sales_train_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Websi
clothing_sales_test_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Websit
```

In [22]:
```
%%R
# Plug and play using glm() for logistic regression model
logreg_r <- glm(formula = CC ~ Days + Web,
                data = clothing_sales_train_r,
                family = binomial)
```

In [23]:
```
%%R
# View the summary of the model
summary(logreg_r)
```

```
Call:
glm(formula = CC ~ Days + Web, family = binomial, data = clothing_sales_train_
r)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.9035  -1.1458  -0.6078   1.0895   2.1044  

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.4961706  0.0886529   5.597 2.18e-08 ***
Days        -0.0037016  0.0004381  -8.449  < 2e-16 ***
Web          1.2536955  0.3306672   3.791  0.00015 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2009.9  on 1450  degrees of freedom
Residual deviance: 1903.6  on 1448  degrees of freedom
AIC: 1909.6

Number of Fisher Scoring iterations: 4
```

### 14. Are there any variables that should be removed from the model? If so, remove them and rerun the model.

### Python

In [24]:
```
logreg_py.summary2()
```

Out[24]:

| | | | |
|---|---|---|---|
| Model: | Logit | Pseudo R-squared: | 0.053 |
| Dependent Variable: | CC | AIC: | 1909.5825 |
| Date: | 2021-07-26 01:12 | BIC: | 1925.4226 |
| No. Observations: | 1451 | Log-Likelihood: | -951.79 |
| Df Model: | 2 | LL-Null: | -1004.9 |
| Df Residuals: | 1448 | LLR p-value: | 8.3668e-24 |
| Converged: | 1.0000 | Scale: | 1.0000 |
| No. Iterations: | 5.0000 | | |

| | Coef. | Std.Err. | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 0.4962 | 0.0887 | 5.5968 | 0.0000 | 0.3224 | 0.6699 |
| **Days** | -0.0037 | 0.0004 | -8.4491 | 0.0000 | -0.0046 | -0.0028 |
| **Web** | 1.2537 | 0.3307 | 3.7914 | 0.0001 | 0.6056 | 1.9018 |

We can see the p-values are all below 0.05, so that's good. Then we check for multicollinearity.

In [25]:
```
# Use VIF values to check for multicollinearity
vif_X = clothing_sales_train_py[['Days', 'Web']]
vif_data = pd.DataFrame()
vif_data["feature"] = vif_X.columns
```

In [26]:
```
# Calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(vif_X.values, i)
                          for i in range(len(vif_X.columns))]
```

In [27]:
```
vif_data
```

Out[27]:

| | feature | VIF |
|---|---|---|
| **0** | Days | 1.010184 |
| **1** | Web | 1.010184 |

**Low VIF values, not highly correlated. Keep both attributes.**

**R**

In [28]:
```
%%R
summary(logreg_r)
```

```
Call:
glm(formula = CC ~ Days + Web, family = binomial, data = clothing_sales_train_
r)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.9035  -1.1458  -0.6078   1.0895   2.1044  

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.4961706  0.0886529   5.597 2.18e-08 ***
Days        -0.0037016  0.0004381  -8.449  < 2e-16 ***
Web          1.2536955  0.3306672   3.791  0.00015 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2009.9  on 1450  degrees of freedom
Residual deviance: 1903.6  on 1448  degrees of freedom
AIC: 1909.6

Number of Fisher Scoring iterations: 4
```

We can see the p-values are all below 0.05, so that's good. Then we check for multicollinearity.

In [29]:
```
%%R
vif(logreg_r)
```

```
    Days      Web 
1.001107 1.001107 
```

**Low VIF values, not highly correlated. Keep both attributes.**

**15. Write the descriptive form of the logistic regression model using the coefficients obtained from Question 1.**

**Python**

y = (exp(0.4962 - 0.0037*Days* + *1.2537*Web )) / (1 + exp(0.4962 - 0.0037*Days* + *1.2537*Web))

**R**

y = (exp(0.4962 - 0.0037*Days* + *1.2537*Web )) / (1 + exp(0.4962 - 0.0037*Days* + *1.2537*Web))

### 16. Validate the model using the test data set.

### Python

In [30]:
```
# Repeat the steps using test set
X_test_py = pd.DataFrame(clothing_sales_test_py[['Days', 'Web']])
y_test_py = pd.DataFrame(clothing_sales_test_py[['CC']])
X_test_py = sm.add_constant(X_test_py)
```

In [31]:
```
logreg_test_py = sm.Logit(y_test_py, X_test_py).fit()
logreg_test_py.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.656885
         Iterations 5
```

Out[31]:

| | | | |
|---|---|---|---|
| Model: | Logit | Pseudo R-squared: | 0.052 |
| Dependent Variable: | CC | AIC: | 1838.7104 |
| Date: | 2021-07-26 01:12 | BIC: | 1854.4324 |
| No. Observations: | 1395 | Log-Likelihood: | -916.36 |
| Df Model: | 2 | LL-Null: | -966.40 |
| Df Residuals: | 1392 | LLR p-value: | 1.8534e-22 |
| Converged: | 1.0000 | Scale: | 1.0000 |
| No. Iterations: | 5.0000 | | |

| | Coef. | Std.Err. | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 0.4634 | 0.0873 | 5.3105 | 0.0000 | 0.2924 | 0.6345 |
| **Days** | -0.0035 | 0.0004 | -8.2261 | 0.0000 | -0.0043 | -0.0026 |
| **Web** | 1.0973 | 0.2830 | 3.8780 | 0.0001 | 0.5427 | 1.6519 |

### R

In [32]:
```
%%R
# Repeat the steps using test set
logreg_test_r <- glm(formula = CC ~ Days + Web,
                data = clothing_sales_test_r,
                family = binomial)
summary(logreg_test_r)
```

```
Call:
glm(formula = CC ~ Days + Web, family = binomial, data = clothing_sales_test_r)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8458  -1.1588  -0.5775   1.1022   2.0513

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.4634478  0.0872706   5.310 1.09e-07 ***
Days        -0.0034721  0.0004221  -8.226  < 2e-16 ***
Web          1.0972994  0.2829570   3.878 0.000105 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1932.8  on 1394  degrees of freedom
Residual deviance: 1832.7  on 1392  degrees of freedom
AIC: 1838.7

Number of Fisher Scoring iterations: 4
```

**17. Obtain the predicted values of the response variable for each record in the data set.**

**Python**

In [33]:
```
# Predictions are probabilities using test data. If probability > 0.5, the instar
predictions_prob_py = logreg_test_py.predict(X_test_py)
```

In [34]:
```
predictions_prob_py
```

Out[34]:
```
0       0.464884
1       0.539722
2       0.572808
3       0.552734
4       0.388604
          ...
1390    0.535372
1391    0.570546
1392    0.464021
1393    0.540903
1394    0.548008
Length: 1395, dtype: float64
```

In [35]:
```
# Convert prediction probabilities of >0.5 as for positive cases
predictions_py = (logreg_test_py.predict(X_test_py) > 0.5).astype(int)
predictions_py
```

Out[35]:
```
0       0
1       1
2       1
3       1
4       0
       ..
1390    1
1391    1
1392    0
1393    1
1394    1
Length: 1395, dtype: int32
```

In [36]:
```
# Confusion Matrix
confusion_matrix(y_test_py, predictions_py)
```

Out[36]:
```
array([[410, 307],
       [217, 461]], dtype=int64)
```

### R

In [37]:
```
%%R
clothing_sales_test_r$predictions_prob <- predict(object = logreg_r, newdata = cl
```

In [38]:
```
%%R
clothing_sales_test_r$predictions_prob
```

```
  [1] 0.4630895 0.5428533 0.5780543 0.5567058 0.3820027 0.5708153 0.5620339
  [8] 0.7991159 0.5933228 0.5425042 0.2251005 0.3776430 0.3165261 0.5560206
 [15] 0.5779099 0.5535062 0.8128358 0.5440381 0.5402065 0.1570451 0.5254650
 [22] 0.5805803 0.1248921 0.5217717 0.5180760 0.4963369 0.5849634 0.3759046
 [29] 0.2991773 0.5975138 0.5812561 0.4058585 0.5423572 0.5420448 0.4957817
 [36] 0.5286944 0.5489278 0.5512181 0.5179281 0.5480111 0.5737147 0.4557352
 [43] 0.8230801 0.5977808 0.6024798 0.1106758 0.5139159 0.3145275 0.5825620
 [50] 0.5747012 0.4262596 0.5305387 0.4989280 0.6057642 0.4289608 0.6127433
 [57] 0.5682107 0.2187095 0.5171517 0.3137300 0.8280954 0.5854756 0.4206669
 [64] 0.1092268 0.5546312 0.4640100 0.4981876 0.5756057 0.4450516 0.5065159
 [71] 0.4940236 0.7891381 0.5406662 0.5046653 0.5799942 0.5614963 0.5194621
 [78] 0.6000038 0.4952265 0.5422010 0.5516759 0.5015839 0.5222335 0.3599807
 [85] 0.3519193 0.4720741 0.3785134 0.1915801 0.4806147 0.5425042 0.5000384
 [92] 0.4247317 0.5853318 0.4520651 0.7379719 0.3274346 0.5619064 0.5388271
 [99] 0.3028288 0.5795614 0.5470941 0.5528017 0.4824630 0.5344919 0.4283534
[106] 0.5835248 0.4649308 0.4600262 0.5849005 0.5553351 0.3229670 0.5709695
[113] 0.5026296 0.5516759 0.5905510 0.5142858 0.4100336 0.3863814 0.5094764
[120] 0.5695453 0.4912480 0.8227618 0.4667729 0.5573452 0.5213098 0.4950414
[127] 0.5918751 0.4130187 0.8382994 0.5676385 0.5251881 0.5040546 0.5165416
```

In [39]:
```
%%R
# Convert prediction probabilities of >0.5 as for positive cases
clothing_sales_test_r$predictions <- (clothing_sales_test_r$predictions_prob > 0.
```

In [40]:
```
%%R
clothing_sales_test_r$predictions
```

```
   [1] 0 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 0 1
1
  [38] 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1
1
  [75] 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 1 1 1 0 1 0 1 0 0 1 1
0
 [112] 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1
1
 [149] 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 0 1 1
1
 [186] 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1
1
 [223] 0 1 0 1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 0
0
 [260] 0 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1
1
 [297] 1 1 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1
1
 [334] 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0 0
0
 [371] 1 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0
0
 [408] 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0
1
 [445] 0 0 0 0 0 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1
1
 [482] 0 0 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1 0
0
 [519] 1 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0
1
 [556] 1 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 0 1 1
1
 [593] 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1
0
 [630] 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0
1
 [667] 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 0 0 1
0
 [704] 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0
1
 [741] 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 0
0
 [778] 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 1 0 1 1 0 1
1
 [815] 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1
1
 [852] 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0
1
 [889] 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0
1
 [926] 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 0
0
 [963] 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 1 0 0 1 1
0
```

```
[1000] 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1
1
[1037] 0 1 0 0 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 0 0 0
1
[1074] 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0 1 1 1 0
1
[1111] 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 1 1 1 1
0
[1148] 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1
1
[1185] 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 1 0
0
[1222] 0 0 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 0 1 1
1
[1259] 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 0 0 0
1
[1296] 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0
1
[1333] 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0
1
[1370] 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 1 1
```

In [41]:

```
%%R
# Confusion Matrix
clothing_sales_test_r[c('CC', 'predictions')] <- lapply(clothing_sales_test_r[c('
confusionMatrix(clothing_sales_test_r$predictions, clothing_sales_test_r$CC, posi
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 405 215
         1 312 463

               Accuracy : 0.6222
                 95% CI : (0.5962, 0.6477)
    No Information Rate : 0.514
    P-Value [Acc > NIR] : 2.532e-16

                  Kappa : 0.2468

 Mcnemar's Test P-Value : 2.892e-05

            Sensitivity : 0.6829
            Specificity : 0.5649
         Pos Pred Value : 0.5974
         Neg Pred Value : 0.6532
             Prevalence : 0.4860
         Detection Rate : 0.3319
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.6239

       'Positive' Class : 1
```

**For the following exercises, work with the bank_marketing_training and the bank**

**marketing_test data set. Use either Python or R to solve each problem.**

### 24. Prepare the data set for neural network modeling, including standardizing the variables.

### R

In [42]:
```
%%R
bank_marketing_train_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Websi
bank_marketing_test_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Websit
```

In [43]:
```
%%R
# Check for null
sum(is.na(bank_marketing_train_r))
```

```
[1] 0
```

In [44]:
```
%%R
head(bank_marketing_train_r)
```

```
  age         job  marital   education default housing loan   contact month
1  56   housemaid  married    basic.4y      no      no   no telephone   may
2  57    services  married high.school unknown      no   no telephone   may
3  41 blue-collar  married     unknown unknown      no   no telephone   may
4  25    services   single high.school      no     yes   no telephone   may
5  29 blue-collar   single high.school      no      no  yes telephone   may
6  57   housemaid divorced    basic.4y      no     yes   no telephone   may
  day_of_week duration campaign days_since_previous previous previous_outcome
1         mon      261        1                 999        0      nonexistent
2         mon      149        1                 999        0      nonexistent
3         mon      217        1                 999        0      nonexistent
4         mon      222        1                 999        0      nonexistent
5         mon      137        1                 999        0      nonexistent
6         mon      293        1                 999        0      nonexistent
  emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed response
1          1.1         93.994         -36.4     4.857        5191       no
2          1.1         93.994         -36.4     4.857        5191       no
3          1.1         93.994         -36.4     4.857        5191       no
4          1.1         93.994         -36.4     4.857        5191       no
5          1.1         93.994         -36.4     4.857        5191       no
6          1.1         93.994         -36.4     4.857        5191       no
```

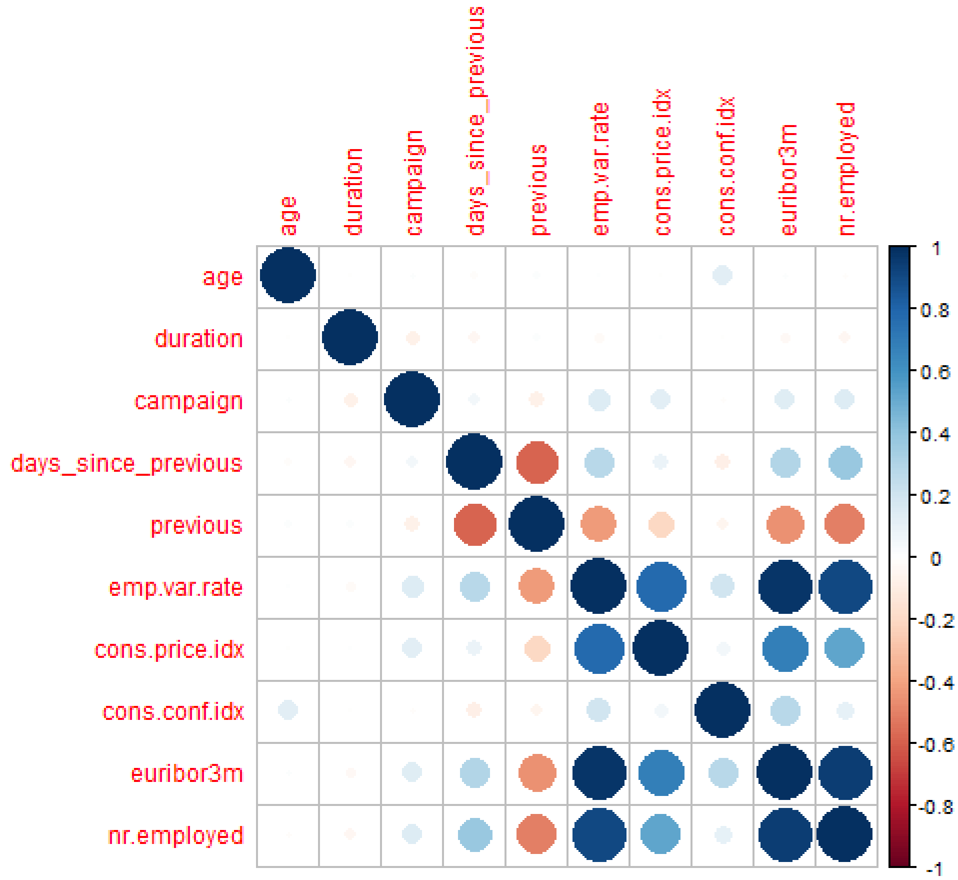
In [45]:
```
%%R
# Extract numeric columns
bank_train_num_r <- bank_marketing_train_r[ c('age', 'duration', 'campaign',
                                              'days_since_previous', 'previous'
                                             'cons.price.idx', 'cons.conf.idx',
                                             'nr.employed')]
head(bank_train_num_r)
```

```
  age duration campaign days_since_previous previous emp.var.rate
1  56      261        1                 999        0          1.1
2  57      149        1                 999        0          1.1
3  41      217        1                 999        0          1.1
4  25      222        1                 999        0          1.1
5  29      137        1                 999        0          1.1
6  57      293        1                 999        0          1.1
  cons.price.idx cons.conf.idx euribor3m nr.employed
1         93.994         -36.4     4.857        5191
2         93.994         -36.4     4.857        5191
3         93.994         -36.4     4.857        5191
4         93.994         -36.4     4.857        5191
5         93.994         -36.4     4.857        5191
6         93.994         -36.4     4.857        5191
```

In [46]:
```
%%R
# Extract categorical columns
bank_train_cat_r <- bank_marketing_train_r[ , c('job', 'marital', 'education', 'c
                                              'housing', 'loan', 'contact','mor
                                              'previous_outcome','response')]
head(bank_train_cat_r)
```

```
          job  marital   education default housing loan   contact month
1   housemaid  married    basic.4y      no      no   no telephone   may
2    services  married high.school unknown      no   no telephone   may
3 blue-collar  married     unknown unknown      no   no telephone   may
4    services   single high.school      no     yes   no telephone   may
5 blue-collar   single high.school      no      no  yes telephone   may
6   housemaid divorced    basic.4y      no     yes   no telephone   may
  day_of_week previous_outcome response
1         mon      nonexistent       no
2         mon      nonexistent       no
3         mon      nonexistent       no
4         mon      nonexistent       no
5         mon      nonexistent       no
6         mon      nonexistent       no
```

In [47]:
```
%%R
# Check for multicollinearity in numeric features using corrplot
corrplot(cor(bank_train_num_r), method="circle")
```



In [48]:
```
%%R
# Correlation coefficients
round(cor(bank_train_num_r),2)
```

```
                    age duration campaign days_since_previous previous
age                1.00     0.00     0.01               -0.02     0.02
duration           0.00     1.00    -0.07               -0.04     0.02
campaign           0.01    -0.07     1.00                0.05    -0.08
days_since_previous -0.02   -0.04     0.05                1.00    -0.58
previous           0.02     0.02    -0.08               -0.58     1.00
emp.var.rate       0.00    -0.03     0.15                0.27    -0.42
cons.price.idx     0.00     0.00     0.13                0.08    -0.21
cons.conf.idx      0.13     0.00    -0.02               -0.09    -0.05
euribor3m          0.01    -0.03     0.13                0.30    -0.45
nr.employed       -0.01    -0.04     0.14                0.37    -0.50
                    emp.var.rate cons.price.idx cons.conf.idx euribor3m
age                         0.00           0.00          0.13      0.01
duration                   -0.03           0.00          0.00     -0.03
campaign                    0.15           0.13         -0.02      0.13
days_since_previous         0.27           0.08         -0.09      0.30
previous                   -0.42          -0.21         -0.05     -0.45
emp.var.rate                1.00           0.78          0.20      0.97
cons.price.idx              0.78           1.00          0.06      0.69
```

For corrlation greater than 0.7, consider multicollinearity

In [49]:
```
%%R
# Remove cons.price.idx, emp.var.rate, euribor3m, nr.employed
bank_train_num_r[ ,c('cons.price.idx', 'emp.var.rate', 'euribor3m', 'nr.employed'
head(bank_train_num_r)
```

```
  age duration campaign days_since_previous previous cons.conf.idx
1  56      261        1                 999        0         -36.4
2  57      149        1                 999        0         -36.4
3  41      217        1                 999        0         -36.4
4  25      222        1                 999        0         -36.4
5  29      137        1                 999        0         -36.4
6  57      293        1                 999        0         -36.4
```

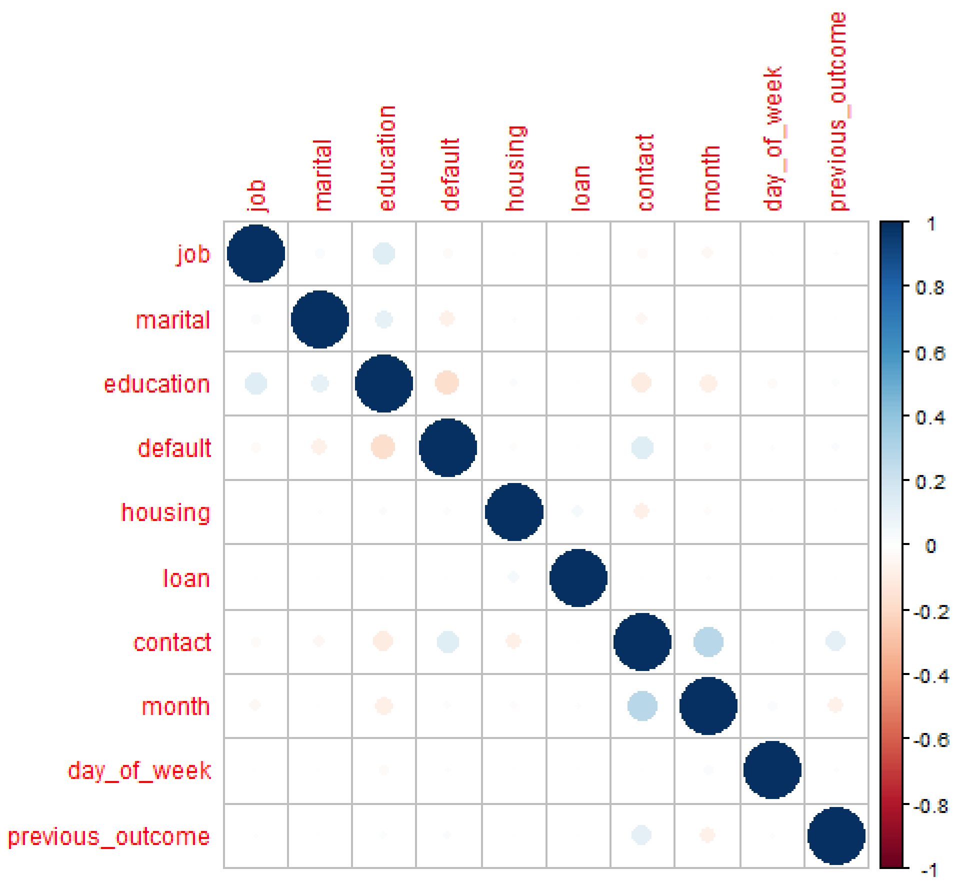
In [50]:
```
%%R
# Standardize the numerical attributes
bank_train_num_r$age.mm <- (bank_train_num_r$age - min(bank_train_num_r$age)) /
(max(bank_train_num_r$age) - min(bank_train_num_r$age))
bank_train_num_r$duration.mm  <- (bank_train_num_r$duration  - min(bank_train_nur
(max(bank_train_num_r$duration) - min(bank_train_num_r$duration))
bank_train_num_r$campaign.mm  <- (bank_train_num_r$campaign  - min(bank_train_nur
(max(bank_train_num_r$campaign) - min(bank_train_num_r$campaign))
bank_train_num_r$days_since_previous.mm <- (bank_train_num_r$days_since_previous
(max(bank_train_num_r$days_since_previous) - min(bank_train_num_r$days_since_prev
bank_train_num_r$previous.mm <- (bank_train_num_r$previous - min(bank_train_num_r
(max(bank_train_num_r$previous) - min(bank_train_num_r$previous))
bank_train_num_r$cons.conf.idx.mm <- (bank_train_num_r$cons.conf.idx - min(bank_t
(max(bank_train_num_r$cons.conf.idx) - min(bank_train_num_r$cons.conf.idx))
head(bank_train_num_r)
```

```
  age duration campaign days_since_previous previous cons.conf.idx    age.mm
1  56      261        1                 999        0         -36.4 0.5270270
2  57      149        1                 999        0         -36.4 0.5405405
3  41      217        1                 999        0         -36.4 0.3243243
4  25      222        1                 999        0         -36.4 0.1081081
5  29      137        1                 999        0         -36.4 0.1621622
6  57      293        1                 999        0         -36.4 0.5405405
  duration.mm campaign.mm days_since_previous.mm previous.mm cons.conf.idx.mm
1  0.05307035           0                      1           0        0.6025105
2  0.03029687           0                      1           0        0.6025105
3  0.04412363           0                      1           0        0.6025105
4  0.04514030           0                      1           0        0.6025105
5  0.02785685           0                      1           0        0.6025105
6  0.05957706           0                      1           0        0.6025105
```

In [51]:
```
%%R
# Change categorical data to factors for the model
bank_train_cat_r$job <- factor(bank_train_cat_r$job)
bank_train_cat_r$marital <- factor(bank_train_cat_r$marital)
bank_train_cat_r$education <- factor(bank_train_cat_r$education)
bank_train_cat_r$default <- factor(bank_train_cat_r$default)
bank_train_cat_r$housing <- factor(bank_train_cat_r$housing)
bank_train_cat_r$loan <- factor(bank_train_cat_r$loan)
bank_train_cat_r$contact <- factor(bank_train_cat_r$contact)
bank_train_cat_r$month <- factor(bank_train_cat_r$month)
bank_train_cat_r$day_of_week <- factor(bank_train_cat_r$day_of_week)
bank_train_cat_r$previous_outcome <- factor(bank_train_cat_r$previous_outcome)
bank_train_cat_r$response <- factor(bank_train_cat_r$response)
```

In [52]:
```
%%R
# Change type of the categorical variables to factors to numbers to check for mul
bank_train_cat_num_r <- subset (bank_train_cat_r, select = -response)
bank_train_cat_num_r$job <- as.numeric(factor(bank_train_cat_r$job))
bank_train_cat_num_r$marital <- as.numeric(factor(bank_train_cat_r$marital))
bank_train_cat_num_r$education <- as.numeric(factor(bank_train_cat_r$education))
bank_train_cat_num_r$default <- as.numeric(factor(bank_train_cat_r$default))
bank_train_cat_num_r$housing <- as.numeric(factor(bank_train_cat_r$housing))
bank_train_cat_num_r$loan <- as.numeric(factor(bank_train_cat_r$loan))
bank_train_cat_num_r$contact <- as.numeric(factor(bank_train_cat_r$contact))
bank_train_cat_num_r$month <- as.numeric(factor(bank_train_cat_r$month))
bank_train_cat_num_r$day_of_week <- as.numeric(factor(bank_train_cat_r$day_of_wee
bank_train_cat_num_r$previous_outcome <- as.numeric(factor(bank_train_cat_r$previ
```

In [53]:
```
%%R
# Check for multicollinearity in categorical features using corrplot
corrplot(cor(bank_train_cat_num_r), method="circle")
```



In [54]:
```
%%R
# Check correlation coefficients
cor(bank_train_cat_num_r)
```

```
                        job      marital    education       default
job             1.000000000  0.0281238530  0.136793828 -0.0286272440
marital         0.028123853  1.0000000000  0.106222323 -0.0762771113
education       0.136793828  0.1062223228  1.000000000 -0.1798668381
default        -0.028627244 -0.0762771113 -0.179866838  1.0000000000
housing         0.009582003  0.0135293479  0.020243286 -0.0155394016
loan           -0.009139961 -0.0021552953  0.004254739 -0.0004944247
contact        -0.026676616 -0.0469451279 -0.106988674  0.1370650180
month          -0.037696891 -0.0071016985 -0.086758570 -0.0147433177
day_of_week    -0.001750037  0.0008636259 -0.025153700 -0.0135207274
previous_outcome  0.011166753  0.0060754395  0.016550309  0.0229150769
                    housing          loan      contact         month
job             0.009582003 -0.0091399611 -0.026676616 -0.037696891
marital         0.013529348 -0.0021552953 -0.046945128 -0.007101698
education       0.020243286  0.0042547392 -0.106988674 -0.086758570
default        -0.015539402 -0.0004944247  0.137065018 -0.014743318
housing         1.000000000  0.0445323724 -0.084357552 -0.016788880
loan            0.044532372  1.0000000000 -0.008001605 -0.012057949
contact        -0.084357552 -0.0080016051  1.000000000  0.278624764
```

Categorical features are not highly correlated.

In [55]:
```
%%R
# Reassemble the original dataframe
bank_marketing_train_r <- data.frame(bank_train_num_r,bank_train_cat_r)
head(bank_marketing_train_r)
```

```
  age duration campaign days_since_previous previous cons.conf.idx    age.mm
1  56      261        1                 999        0         -36.4 0.5270270
2  57      149        1                 999        0         -36.4 0.5405405
3  41      217        1                 999        0         -36.4 0.3243243
4  25      222        1                 999        0         -36.4 0.1081081
5  29      137        1                 999        0         -36.4 0.1621622
6  57      293        1                 999        0         -36.4 0.5405405
  duration.mm campaign.mm days_since_previous.mm previous.mm cons.conf.idx.mm
1  0.05307035           0                      1           0        0.6025105
2  0.03029687           0                      1           0        0.6025105
3  0.04412363           0                      1           0        0.6025105
4  0.04514030           0                      1           0        0.6025105
5  0.02785685           0                      1           0        0.6025105
6  0.05957706           0                      1           0        0.6025105
          job  marital   education default housing loan   contact month
1   housemaid  married    basic.4y      no      no   no telephone   may
2    services  married high.school unknown      no   no telephone   may
3 blue-collar  married     unknown unknown      no   no telephone   may
4    services   single high.school      no     yes   no telephone   may
5 blue-collar   single high.school      no      no  yes telephone   may
6   housemaid divorced    basic.4y      no     yes   no telephone   may
  day_of_week previous_outcome response
1         mon      nonexistent       no
2         mon      nonexistent       no
3         mon      nonexistent       no
4         mon      nonexistent       no
5         mon      nonexistent       no
6         mon      nonexistent       no
```

In [56]:
```
%%R
# Run the neural network algorithm
nnet_r <- nnet(response ~ age.mm + duration.mm + campaign.mm + days_since_previou
               education + default + housing + loan + contact + month + day_of_we
               data = bank_marketing_train_r,
               size = 16) # number of nodes in the hidden layer
```

```
# weights:  817
initial  value 33164.520898
iter  10 value 7732.490731
iter  20 value 6024.311311
iter  30 value 5533.244940
iter  40 value 5256.491055
iter  50 value 5038.648672
iter  60 value 4896.573216
iter  70 value 4749.423838
iter  80 value 4592.642818
iter  90 value 4504.226243
iter 100 value 4428.517336
final  value 4428.517336
stopped after 100 iterations
```

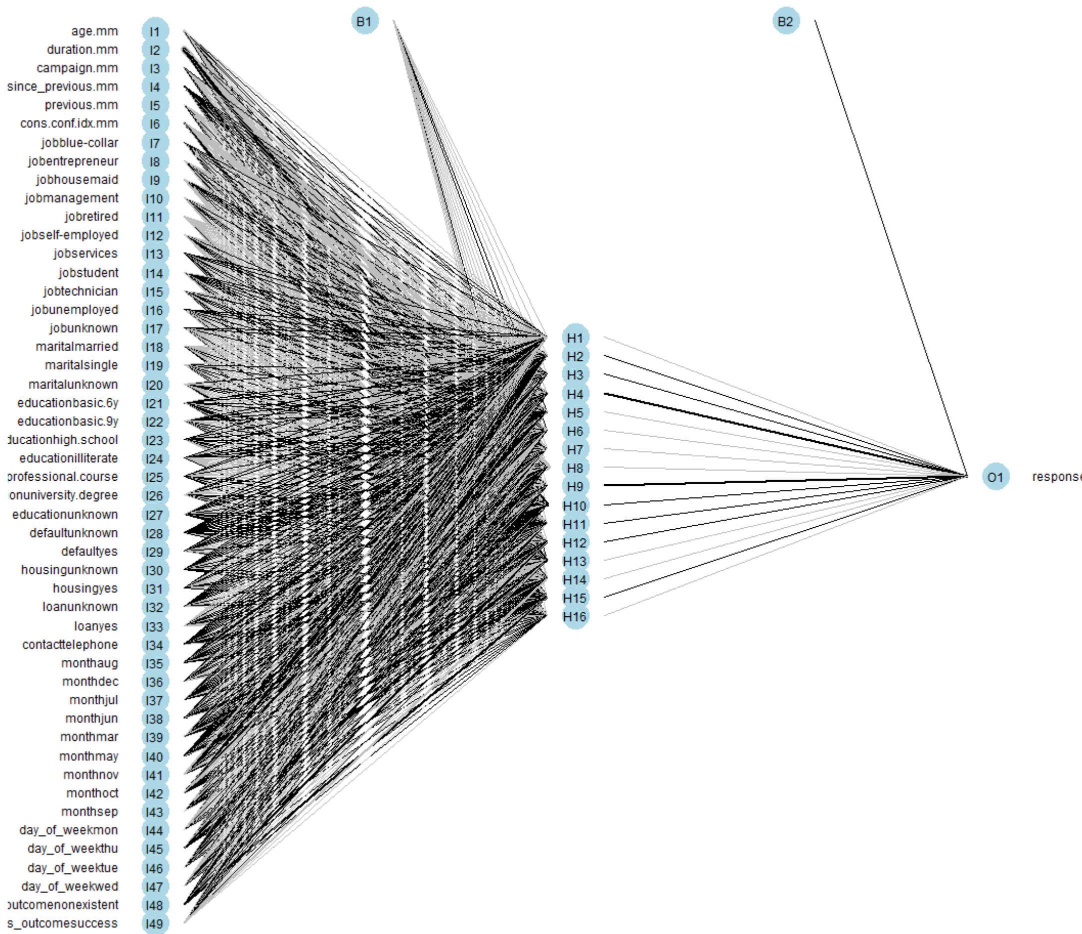
In [57]:
```
%%R
# Make predictions
bank_marketing_train_r$pred <- predict(object = nnet_r, newdata = bank_marketing_
bank_marketing_train_r$pred
```

```
   [1] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [13] "no"  "no"  "no"  "no"  "no"  "no"  "yes" "no"  "no"  "no"  "no"  "n
o"
  [25] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [37] "no"  "no"  "no"  "no"  "no"  "no"  "yes" "no"  "no"  "no"  "no"  "n
o"
  [49] "yes" "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [61] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [73] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [85] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
  [97] "no"  "no"  "yes" "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
o"
 [109] "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "no"  "n
```

**26. Plot the neural network.**

In [58]:
```
%%R -w 1200 -h 1200

plotnet(nnet_r)
```



**27. Evaluate the neural network model using the test data set. Construct a contingency table to compare the actual and predicted values of Response.**

In [59]:
```
%%R
# Remove the same variables from test set
bank_marketing_test_r[ ,c('cons.price.idx', 'emp.var.rate', 'euribor3m', 'nr.empl
```

In [60]:
```
%%R
# Standardize the numerical attributes
bank_marketing_test_r$age.mm <- (bank_marketing_test_r$age - min(bank_marketing_t
(max(bank_marketing_test_r$age) - min(bank_marketing_test_r$age))
bank_marketing_test_r$duration.mm  <- (bank_marketing_test_r$duration  - min(bank
(max(bank_marketing_test_r$duration) - min(bank_marketing_test_r$duration))
bank_marketing_test_r$campaign.mm  <- (bank_marketing_test_r$campaign  - min(bank
(max(bank_marketing_test_r$campaign) - min(bank_marketing_test_r$campaign))
bank_marketing_test_r$days_since_previous.mm <- (bank_marketing_test_r$days_since
(max(bank_marketing_test_r$days_since_previous) - min(bank_marketing_test_r$days_
bank_marketing_test_r$previous.mm <- (bank_marketing_test_r$previous - min(bank_m
(max(bank_marketing_test_r$previous) - min(bank_marketing_test_r$previous))
bank_marketing_test_r$cons.conf.idx.mm <- (bank_marketing_test_r$cons.conf.idx -
(max(bank_marketing_test_r$cons.conf.idx) - min(bank_marketing_test_r$cons.conf.i
```

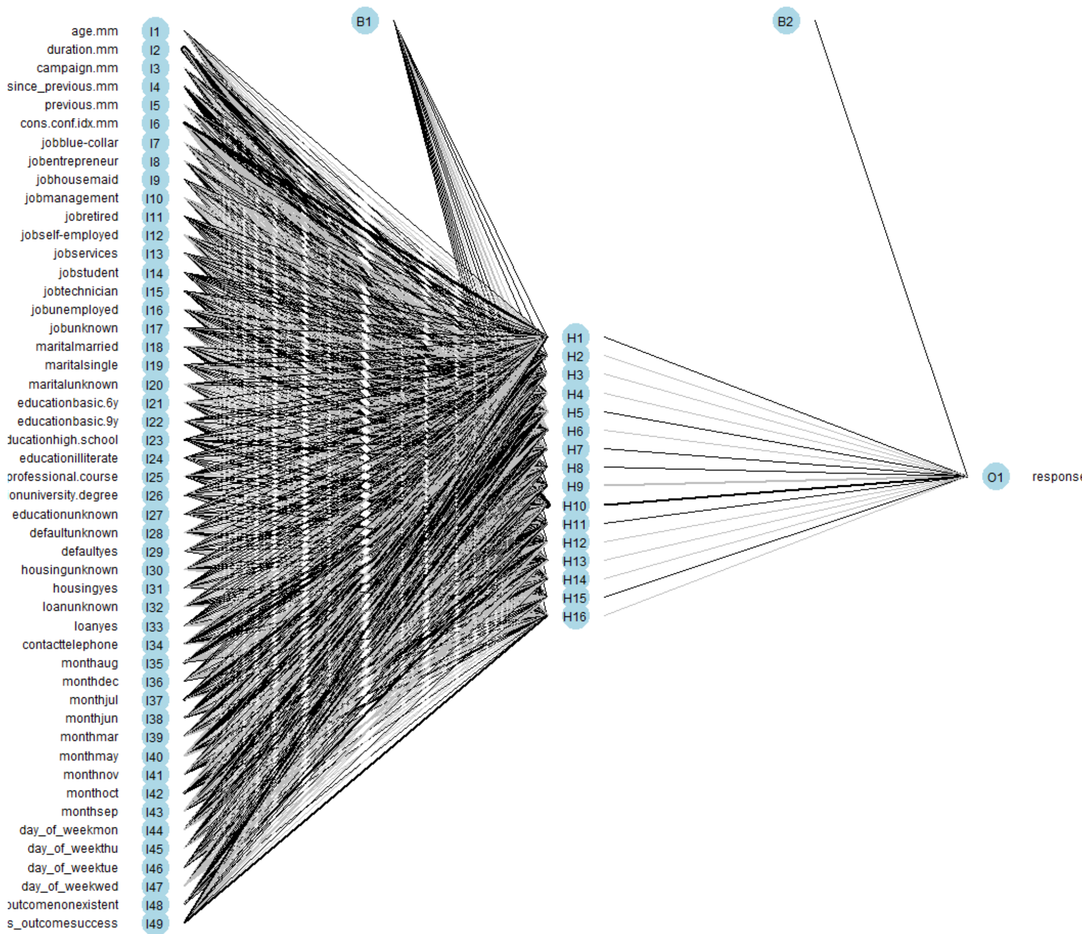
In [61]:
```
%%R
# Convert the categorical data to factors
bank_marketing_test_r$job <- factor(bank_marketing_test_r$job)
bank_marketing_test_r$marital <- factor(bank_marketing_test_r$marital)
bank_marketing_test_r$education <- factor(bank_marketing_test_r$education)
bank_marketing_test_r$default <- factor(bank_marketing_test_r$default)
bank_marketing_test_r$housing <- factor(bank_marketing_test_r$housing)
bank_marketing_test_r$loan <- factor(bank_marketing_test_r$loan)
bank_marketing_test_r$contact <- factor(bank_marketing_test_r$contact)
bank_marketing_test_r$month <- factor(bank_marketing_test_r$month)
bank_marketing_test_r$day_of_week <- factor(bank_marketing_test_r$day_of_week)
bank_marketing_test_r$previous_outcome <- factor(bank_marketing_test_r$previous_c
bank_marketing_test_r$response <- factor(bank_marketing_test_r$response)
```

In [62]:
```
%%R
nnet02_r <- nnet(response ~ age.mm + duration.mm + campaign.mm + days_since_previ
                  + job + marital + education + default + housing + loan + contact
               data = bank_marketing_test_r,
               size = 16)
```

```
# weights:  817
initial  value 9653.438932
iter  10 value 7225.304469
iter  20 value 5998.400409
iter  30 value 5449.601963
iter  40 value 5228.402876
iter  50 value 5081.077753
iter  60 value 4955.075944
iter  70 value 4810.081623
iter  80 value 4675.562929
iter  90 value 4579.915575
iter 100 value 4490.769405
final  value 4490.769405
stopped after 100 iterations
```

In [63]:
```
%%R -w 1200 -h 1200

plotnet(nnet02_r)
```



In [64]:
```
%%R
# Predictions from test set
bank_marketing_test_r$pred <- predict(object = nnet02_r, newdata = bank_marketing
```

In [65]:
```
%%R
# Build a contingency table and compare the acutal response in test data set with

t1 <- table(bank_marketing_test_r$response, bank_marketing_test_r$pred)
row.names(t1) <- c("Actual: 0", "Actual: 1")
colnames(t1) <- c("Predicted: 0", "Predicted: 1")
t1 <- addmargins(A = t1, FUN = list(Total = sum), quiet = TRUE)
t1
```

```
             Predicted: 0 Predicted: 1 Total
  Actual: 0         23194          692 23886
  Actual: 1          1305         1683  2988
  Total             24499         2375 26874
```

In [66]:
```
%%R
TN_r = t1[1:1,1:1]
FP_r = t1[1:1,2:2]
FN_r = t1[2:2,1:1]
TP_r = t1[2:2,2:2]
cat(paste("TN:", TN_r,"\nFP:", FP_r,"\nFN:", FN_r,"\nTP:", TP_r))
```

```
TN: 23194
FP: 692
FN: 1305
TP: 1683
```

**28. Which baseline model do we compare your neural network model against? Did it outperform the baseline according to accuracy?**

**We use All Negative Model since most of the responses are 'no', then we outperformed the**

**baseline model by 3%.**

In [67]:
```
%%R
cat(paste("Accuracy for ANN model is: ", round((TN_r+TP_r)/(TN_r+TP_r+FN_r+FP_r),
          "\nAccuracy for All Negative Model is: ", round(table(bank_marketing_te
                                                        dim(bank_marketing_test
```

```
Accuracy for ANN model is:  0.9257
Accuracy for All Negative Model is:  0.8888
```

**29. Using the same predictors you used for your neural network model, build models to predict Response using the following algorithms:**

**a. CART**

In [68]:
```
%%R
cart_r <- rpart(formula = response ~ age.mm + duration.mm + campaign.mm + days_si
                cons.conf.idx.mm + job + marital + education + default + housing
                day_of_week + previous_outcome, data = bank_marketing_train_r, me
```

**b. C5.0**

In [69]:
```
%%R
c5_r <- C5.0(formula = response ~ age.mm + duration.mm + campaign.mm + days_since
                cons.conf.idx.mm + job + marital + education + default + housing
                day_of_week + previous_outcome, data = bank_marketing_train_r, co
```

**c. Naïve Bayes**

In [70]:
```
%%R
nb_r <- naiveBayes(formula = response ~ age.mm + duration.mm + campaign.mm + days
                cons.conf.idx.mm + job + marital + education + default + housing
                day_of_week + previous_outcome, data = bank_marketing_train_r)
```

**30. Compare the results of your neural network model with the three models from the previous exercise, according to the following criteria. Discuss in detail which model performed best and worst according to each criterion.**

In [71]:
```
%%R
# X from test set
X_for_prediction_r <- data.frame(age.mm = bank_marketing_train_r$age.mm,
                                 duration.mm = bank_marketing_train_r$duration.mm
                                 campaign.mm = bank_marketing_train_r$campaign.mm
                                 days_since_previous.mm = bank_marketing_train_r$
                                 previous.mm = bank_marketing_train_r$previous.mm
                                 cons.conf.idx.mm = bank_marketing_train_r$cons.c
                                 job = bank_marketing_train_r$job,
                                 marital = bank_marketing_train_r$marital,
                                 education = bank_marketing_train_r$education,
                                 default = bank_marketing_train_r$default,
                                 housing = bank_marketing_train_r$housing,
                                 loan = bank_marketing_train_r$loan,
                                 contact = bank_marketing_train_r$contact,
                                 month = bank_marketing_train_r$month,
                                 day_of_week = bank_marketing_train_r$day_of_week
                                 previous_outcome = bank_marketing_train_r$previo
```

In [72]:
```
%%R
# Predictions for CART, C5.0, Naïve Bayes
pred_CART <- predict(object = cart_r, newdata = X_for_prediction_r, type = "class
pred_C50 <- predict(object = c5_r, newdata = X_for_prediction_r)
pred_NB <- predict(object = nb_r, newdata = bank_marketing_train_r)
```

In [73]:
```
%%R
# Build a contingency table for each model
t_CART <- table(bank_marketing_test_r$response, pred_CART)
row.names(t_CART) <- c("Actual: 0", "Actual: 1")
colnames(t_CART) <- c("Predicted: 0", "Predicted: 1")
t_CART <- addmargins(A = t_CART, FUN = list(Total = sum), quiet = TRUE)
t_CART
```

```
          pred_CART
           Predicted: 0 Predicted: 1 Total
  Actual: 0       23313          573 23886
  Actual: 1        1892         1096  2988
  Total           25205         1669 26874
```

In [74]:
```
%%R
TN_CART = t_CART[1:1,1:1]
FP_CART = t_CART[1:1,2:2]
FN_CART = t_CART[2:2,1:1]
TP_CART = t_CART[2:2,2:2]
cat(paste("TN_CART:", TN_CART,"\nFP_CART:", FP_CART,"\nFN_CART:", FN_CART,"\nTP_C
```

```
TN_CART: 23313
FP_CART: 573
FN_CART: 1892
TP_CART: 1096
```

In [75]:
```
%%R
t_C50 <- table(bank_marketing_test_r$response, pred_C50)
row.names(t_C50) <- c("Actual: 0", "Actual: 1")
colnames(t_C50) <- c("Predicted: 0", "Predicted: 1")
t_C50 <- addmargins(A = t_C50, FUN = list(Total = sum), quiet = TRUE)
t_C50
```

```
          pred_C50
           Predicted: 0 Predicted: 1 Total
  Actual: 0        23070          816 23886
  Actual: 1         1491         1497  2988
  Total            24561         2313 26874
```

In [76]:
```
%%R
TN_C50 = t_C50[1:1,1:1]
FP_C50 = t_C50[1:1,2:2]
FN_C50 = t_C50[2:2,1:1]
TP_C50 = t_C50[2:2,2:2]
cat(paste("TN_C50:", TN_C50,"\nFP_C50:", FP_C50,"\nFN_C50:", FN_C50,"\nTP_C50:",
```

```
TN_C50: 23070
FP_C50: 816
FN_C50: 1491
TP_C50: 1497
```

In [77]:
```
%%R
t_NB <- table(bank_marketing_test_r$response, pred_NB)
row.names(t_NB) <- c("Actual: 0", "Actual: 1")
colnames(t_NB) <- c("Predicted: 0", "Predicted: 1")
t_NB <- addmargins(A = t_NB, FUN = list(Total = sum), quiet = TRUE)
t_NB
```

```
          pred_NB
           Predicted: 0 Predicted: 1 Total
  Actual: 0        22706         1180 23886
  Actual: 1         1681         1307  2988
  Total            24387         2487 26874
```

In [78]:
```
%%R
TN_NB = t_NB[1:1,1:1]
FP_NB = t_NB[1:1,2:2]
FN_NB = t_NB[2:2,1:1]
TP_NB = t_NB[2:2,2:2]
cat(paste("TN_NB:", TN_NB,"\nFP_NB:", FP_NB,"\nFN_NB:", FN_NB,"\nTP_NB:", TP_NB))
```

```
TN_NB: 22706
FP_NB: 1180
FN_NB: 1681
TP_NB: 1307
```

#### a. Accuracy

In [79]:
```
%%R
cat(paste("Accuracy for CART model is: ", round((TN_CART+TP_CART)/(TN_CART+TP_CAR
          "\nAccuracy for C50 model is: ", round((TN_C50+TP_C50)/(TN_C50+TP_C50+F
          "\nAccuracy for NB model is: ", round((TN_NB+TP_NB)/(TN_NB+TP_NB+FN_NB+
```

```
Accuracy for CART model is:  0.9083
Accuracy for C50 model is:  0.9142
Accuracy for NB model is:  0.8935
```

#### b. Sensitivity

In [80]:
```
%%R
cat(paste("Sensitivity for CART model is: ", round(TP_CART/(TP_CART+FN_CART),4),
          "\nSensitivity for C50 model is: ", round(TP_C50/(TP_C50+FN_C50),4),
          "\nSensitivity for NB model is: ", round(TP_NB/(TP_NB+FN_NB),4)))
```

```
Sensitivity for CART model is:  0.3668
Sensitivity for C50 model is:  0.501
Sensitivity for NB model is:  0.4374
```

#### c. Specificity

In [81]:
```
%%R
cat(paste("Specificity for CART model is: ", round(TN_CART/(TN_CART+FP_CART),4),
          "\nSpecificity for C50 model is: ", round(TN_C50/(TN_C50+FP_C50),4),
          "\nSpecificity for NB model is: ", round(TN_NB/(TN_NB+FP_NB),4)))
```

```
Specificity for CART model is:  0.976
Specificity for C50 model is:  0.9658
Specificity for NB model is:  0.9506
```

In general, the higher the accuracy, sensitivity, specificity the better. So if we average all 3 scores, CART: 0.75, C50: 0.79, NB: 0.76, we will have C50 > NB > CART. Since we have an imbalanced data class and we use All negative model, we then focus more on the responses of 'yes' which is true positive. So the false negative will cost us more given there are rare enough positive cases and true positive is more important. Hence we'll look at sensitivity and the C50 model has the highest sensitivity score, which makes it the best model; and despite the high accuracy and specificity for CART model, it has the lowest sensitivity score so it's the worst in this case. So final order is C50 > NB > CART.

#### For Exercises 14–20, work with the adult_ch6_training and adult_ch6_test data sets. Use either Python or R to solve each problem.

#### 19. Use random forests on the training data set to predict income using marital status and capital gains and losses.

#### Python

In [82]:
```
adult_ch6_train_py = pd.read_csv("D:/2021-Spring-textbooks/ADS-502/Website Data S
adult_ch6_test_py = pd.read_csv("D:/2021-Spring-textbooks/ADS-502/Website Data Se
adult_ch6_train_py.head()
```

Out[82]:

| | Marital status | Income | Cap_Gains_Losses |
|---|---|---|---|
| **0** | Never-married | <=50K | 0.02174 |
| **1** | Divorced | <=50K | 0.00000 |
| **2** | Married | <=50K | 0.00000 |
| **3** | Married | <=50K | 0.00000 |
| **4** | Married | <=50K | 0.00000 |

In [83]:
```
# Response variable
y_train_py = adult_ch6_train_py[['Income']]
```

In [84]:
```
# Make dummies for categorical variable 'Marital status'
mar_np_py = np.array(adult_ch6_train_py['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np_py, drop=True, dictnames =
mar_cat_pd = pd.DataFrame(mar_cat)
X_train_py = pd.concat((adult_ch6_train_py[['Cap_Gains_Losses']], mar_cat_pd), ax
```

In [85]:
```
# The random forest command in Python requires a response variable formatted as a
# so we use numpy’s ravel() command to create that format.
rfy_py = np.ravel(y_train_py)
```

In [86]:
```
# Run the model
rf_train_py = RandomForestClassifier(n_estimators = 100,
criterion="gini").fit(X_train_py,rfy_py)
```

In [87]:
```
rf_train_py.predict(X_train_py)
```

Out[87]:
```
array(['<=50K', '<=50K', '<=50K', ..., '<=50K', '<=50K', '<=50K'],
      dtype=object)
```

**R**

In [88]:
```
%%R
adult_ch6_train_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Website Da
adult_ch6_test_r <- read.csv(file = "D:/2021-Spring-textbooks/ADS-502/Website Dat
head(adult_ch6_train_r)
```

```
  Marital.status Income Cap_Gains_Losses
1  Never-married  <=50K          0.02174
2       Divorced  <=50K          0.00000
3        Married  <=50K          0.00000
4        Married  <=50K          0.00000
5        Married  <=50K          0.00000
6        Married   >50K          0.00000
```

In [89]:
```
%%R
# Change column to eliminate special characters
colnames(adult_ch6_train_r)[1] <- "maritalStatus"
```

In [90]:
```
%%R
# Convert categorical into factors
adult_ch6_train_r$Income <- factor(adult_ch6_train_r$Income)
adult_ch6_train_r$maritalStatus <- factor(adult_ch6_train_r$maritalStatus)
```

In [91]:
```
%%R
rf_train_r <- randomForest(formula = Income ~ maritalStatus + Cap_Gains_Losses,
                           data = adult_ch6_train_r, ntree = 100, type = 'classificatio
```

In [92]:
```
%%R
rf_train_r$predicted
```

```
    1     2     3     4     5     6     7     8     9    10    11    12    13 
<=50K <=50K <=50K <=50K <=50K <=50K  >50K  >50K <=50K <=50K <=50K <=50K <=50K 
   14    15    16    17    18    19    20    21    22    23    24    25    26 
<=50K <=50K <=50K <=50K <=50K <=50K  >50K <=50K <=50K <=50K <=50K <=50K <=50K 
   27    28    29    30    31    32    33    34    35    36    37    38    39 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   40    41    42    43    44    45    46    47    48    49    50    51    52 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   53    54    55    56    57    58    59    60    61    62    63    64    65 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   66    67    68    69    70    71    72    73    74    75    76    77    78 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   79    80    81    82    83    84    85    86    87    88    89    90    91 
 >50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K  >50K <=50K <=50K <=50K <=50K 
   92    93    94    95    96    97    98    99   100   101   102   103   104 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
  105   106   107   108   109   110   111   112   113   114   115   116   117 
<=50K <=50K <=50K <=50K <=50K <=50K  >50K <=50K <=50K <=50K <=50K <=50K <=50K 
  118   119   120   121   122   123   124   125   126   127   128   129   130 
```

**20. Use random forests using the test data set that utilizes the same target and predictor variables. Does the test data result match the training data result?**

### Python

In [93]:
```
y_test_py = adult_ch6_test_py[['Income']]
mar_np_py = np.array(adult_ch6_test_py['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np_py, drop=True, dictnames =
mar_cat_pd = pd.DataFrame(mar_cat)
X_test_py = pd.concat((adult_ch6_test_py[['Cap_Gains_Losses']], mar_cat_pd), axis
rfy_py = np.ravel(y_test_py)
```

In [94]:
```
rf_test_py = RandomForestClassifier(n_estimators = 100,
criterion="gini").fit(X_test_py,rfy_py)
```

In [95]:
```
rf_test_py.predict(X_test_py)
```

Out[95]:
```
array(['<=50K', '>50K', '<=50K', ..., '<=50K', '<=50K', '<=50K'],
      dtype=object)
```

In [96]:
```
cm_train = confusion_matrix(y_train_py, rf_train_py.predict(X_train_py))
cm_test = confusion_matrix(y_test_py, rf_test_py.predict(X_test_py))
```

In [97]:
```
cm_train
```

Out[97]:
```
array([[14237,    34],
       [ 3138,  1352]], dtype=int64)
```

In [98]:
```
cm_test
```

Out[98]:
```
array([[4668,    6],
       [1034,  447]], dtype=int64)
```

In [99]:
```
TN_train_py = cm_train[0][0]
FP_train_py = cm_train[0][1]
FN_train_py = cm_train[1][0]
TP_train_py = cm_train[1][1]
TN_test_py = cm_test[0][0]
FP_test_py = cm_test[0][1]
FN_test_py = cm_test[1][0]
TP_test_py = cm_test[1][1]
```

In [100]:
```
print('Accuracy for training data is: ',round((TN_train_py+TP_train_py)/(TN_trair
      '\nAccuracy for test data is: ', round((TN_test_py+TP_test_py)/(TN_test_py+T
```

```
Accuracy for training data is:  0.8309
Accuracy for test data is:  0.831
```

In [101]:
```
print('Specificity for training data is: ',round(TN_train_py/(TN_train_py+FP_trai
      '\nSpecificity for test data is: ',round(TN_test_py/(TN_test_py+FP_test_py),
```

```
Specificity for training data is:  0.9976
Specificity for test data is:  0.9987
```

In [102]:
```
print('Sensitivity for training data is: ', round(TP_train_py/(TP_train_py+FN_tra
      '\nSensitivity for test data is: ', round(TP_test_py/(TP_test_py+FN_test_py)
```

```
Sensitivity for training data is:  0.3011
Sensitivity for test data is:  0.3018
```

**The training data result matches test data result.**

**R**

In [103]:
```
%%R
colnames(adult_ch6_test_r)[1] <- "maritalStatus"
adult_ch6_test_r$Income <- factor(adult_ch6_test_r$Income)
adult_ch6_test_r$maritalStatus <- factor(adult_ch6_test_r$maritalStatus)
rf_test_r <- randomForest(formula = Income ~ maritalStatus + Cap_Gains_Losses,
                     data = adult_ch6_test_r, ntree = 100, type = 'classificatior
```

In [104]:
```
%%R
rf_test_r$predicted
```

```
    1     2     3     4     5     6     7     8     9    10    11    12    13 
<=50K  >50K <=50K <=50K <=50K <=50K  >50K <=50K <=50K <=50K <=50K <=50K  >50K 
   14    15    16    17    18    19    20    21    22    23    24    25    26 
<=50K <=50K <=50K <=50K <=50K <=50K  >50K  >50K <=50K <=50K <=50K <=50K <=50K 
   27    28    29    30    31    32    33    34    35    36    37    38    39 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   40    41    42    43    44    45    46    47    48    49    50    51    52 
<=50K <=50K <=50K <=50K <=50K <=50K  >50K <=50K  >50K <=50K <=50K <=50K <=50K 
   53    54    55    56    57    58    59    60    61    62    63    64    65 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   66    67    68    69    70    71    72    73    74    75    76    77    78 
 >50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   79    80    81    82    83    84    85    86    87    88    89    90    91 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
   92    93    94    95    96    97    98    99   100   101   102   103   104 
<=50K <=50K  >50K  >50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
  105   106   107   108   109   110   111   112   113   114   115   116   117 
<=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K <=50K 
  118   119   120   121   122   123   124   125   126   127   128   129   130 
```

In [105]:
```
%%R
t_train <- table(adult_ch6_train_r$Income, rf_train_r$predicted)
row.names(t_train) <- c("Actual: 0", "Actual: 1")
colnames(t_train) <- c("Predicted: 0", "Predicted: 1")
t_train <- addmargins(A = t_train, FUN = list(Total = sum), quiet = TRUE)

t_test <- table(adult_ch6_test_r$Income, rf_test_r$predicted)
row.names(t_test) <- c("Actual: 0", "Actual: 1")
colnames(t_test) <- c("Predicted: 0", "Predicted: 1")
t_test <- addmargins(A = t_test, FUN = list(Total = sum), quiet = TRUE)
```

In [106]:
```
%%R
t_train
```

```
            Predicted: 0 Predicted: 1 Total
  Actual: 0        14155          116 14271
  Actual: 1         3261         1229  4490
  Total            17416         1345 18761
```

In [107]:
```
%%R
t_test
```

```
            Predicted: 0 Predicted: 1 Total
  Actual: 0         4623           51  4674
  Actual: 1         1081          400  1481
  Total             5704          451  6155
```

In [108]:
```
%%R
TN_train_r = t_train[1:1,1:1]
FP_train_r = t_train[1:1,2:2]
FN_train_r = t_train[2:2,1:1]
TP_train_r = t_train[2:2,2:2]
TN_test_r = t_test[1:1,1:1]
FP_test_r = t_test[1:1,2:2]
FN_test_r = t_test[2:2,2:2]
TP_test_r = t_test[2:2,2:2]
```

In [109]:
```
%%R
cat(paste('Accuracy for training data is: ', round((TN_train_r+TP_train_r)/(TN_tr
          '\nAccuracy for test data is: ', round((TN_test_r+TP_test_r)/(TN_test_r
```

```
Accuracy for training data is:  0.82
Accuracy for test data is:  0.9176
```

In [110]:
```
%%R
cat(paste('Specificity for training data is: ', round(TN_train_r/(TN_train_r+FP_t
          '\nSpecificity for test data is: ', round(TN_test_r/(TN_test_r+FP_test_
```

```
Specificity for training data is:  0.9919
Specificity for test data is:  0.9891
```

In [111]:
```
%%R
cat(paste('Sensitivity for training data is: ', round(TP_train_r/(TP_train_r+FN_t
          '\nSensitivity for test data is: ', round(TP_test_r/(TP_test_r+FN_test_
```

```
Sensitivity for training data is:  0.2737
Sensitivity for test data is:  0.5
```

**Compare to the Python code, the Accuracy and Sensitivity don't match as good for training vs. test data result in the R code.**