Rețele neuronale artificiale în R - continuare

Problema 1.

Vom folosi setul de date Carseats din pachetul ISLR si vom construi modele de predictie a vanzarilor (variabila Sales) in functie de restul variabilelor prezent in acest set de date. Vom face o comparatie intre rezultatele obtinute folosind un model de regresie linear si o retea neuronala artificiala.

```
library(ISLR)
library(neuralnet)
library(boot)
set.seed(123)
data <- Carsets
```

Mai intai verificam ca nu exista date lipsa, altfel va trebuie sa imputam datele lipsa:

```
apply(data,2,function(x) sum(is.na(x)))
      Sales   CompPrice      Income Advertising  Population       Price
ShelveLoc         Age   Education       Urban          US
          0           0           0           0           0           0
0           0           0           0           0
```

Nu exista data lipsa!

Vizualizam setul de date:

```
View(data)
```

Constatam ca o serie de coloane au valori non-numerice. Vom adopta 2 solutii:

- Eliminam datele non-numerice si lucram doar cu variabilele numerice (in acest caz am putea "pierde" influenta variabilelor non-numerice asupra vanzarilor – variabila Sales)
- Transformam valorile non-numerice in valori numerice

1. Eliminam variabilele non-numerice

```
#optiunea 1 excludem datele non-numerice
data <-data[,c(1:6,8,9)]
```

Vom continua prin divizarea aleatoare a setului de date intr-un set de date de antrenare si un set de test, apoi vom calcula parametrii unui model de regresie liniara pentru setul de antrenare si ii vom testa folosind setul de date de test. Vom folosi functia `glm()` in loc de `lm()`. Motivele le vom vedea mai tarziu cand vom folosi tehnica validarii incrucisate.

```
#impartim setul de date in 2 subseturi: antrenare(75%) si testare
(25%)
index <- sample(1:nrow(data),round(0.75*nrow(data)))
train <- data[index,]
test <- data[-index,]

#estimam un model de regresie liniara
lm.fit <- glm(Sales~., data=train)
summary(lm.fit)

#folosind modelul estimat calculam valorile variabilei raspuns
pentru setul de date de test
pr.lm <- predict(lm.fit,test)

#calculam MSE pentru setul de date de test
MSE.lm <- sum((pr.lm - test$Sales)^2)/nrow(test)
MSE.lm

Call:
glm(formula = Sales ~ ., data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.1996  -1.2566  -0.1579   1.1478   4.8917

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.4492195  1.3011416   5.725 2.56e-08 ***
CompPrice    0.1013676  0.0090152  11.244  < 2e-16 ***
Income       0.0119177  0.0040399   2.950  0.00344 **
Advertising  0.1284559  0.0174637   7.356 1.93e-12 ***
Population   0.0007644  0.0007975   0.959  0.33857
Price       -0.0999428  0.0058178 -17.179  < 2e-16 ***
Age         -0.0407971  0.0069480  -5.872 1.17e-08 ***
Education   -0.0557201  0.0426674  -1.306  0.19261
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 3.70352)
```

```
    Null deviance: 2504.1  on 299  degrees of freedom
Residual deviance: 1081.4  on 292  degrees of freedom
AIC: 1254

Number of Fisher Scoring iterations: 2
Number of Fisher Scoring iterations: 2
```

Functia `sample(x, size)` genereaza un vector cu dimensiunea specificata alcatuit din elemente selectate aleator dintre elemente vectorului `x`. Extragerea elementelor este fara inlocuire: `index` un vector aleator de indici. Deoarece lucram cu un model de regresie liniara, vom apela la Mean Squared Error (MSE – medie patratica a erorii) ca masura a distantei dintre predictii si datele reale.

```
MSE.lm <- sum((pr.lm - test$Sales)^2)/nrow(test)
> MSE.lm
[1] 12.37378
```

Pregatirea datelor pentru utilizarea lor cu o RNA

Inainte de a folosi o retea neuronala, este nevoie de anumite prelucrari ale datelor.

Pasul 1. Prepocesarea datelor

O practica buna care este utilizata des este aceea de a normaliza datele inainte de utilizarea retelei. Normalizarea datelor este un pas foarte important: in functie de setul de date utilizat, fara normalizare se pot obtine rezultate fara valoare predictiva sau se poate ingreuna foarte mult procesul de antrenare al retelei (algoritmul nu va converge). Exista diferite metode de normalizare a datelor: scalare min-max, z-normalization etc. In exemplul urmator vom folosi metoda min-max si vom scala datele in intervalul [0,1]. Scalarea datelor in intervalul [0,1] sau chiar [-1,1] conduce la rezultate bune in practica.

Vom scala si diviza setul de date:

```
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
```

```
scaled <- as.data.frame(scale(data, center = mins, scale = maxs -
mins))

train_ <- scaled[index,]
test_ <- scaled[-index,]
```

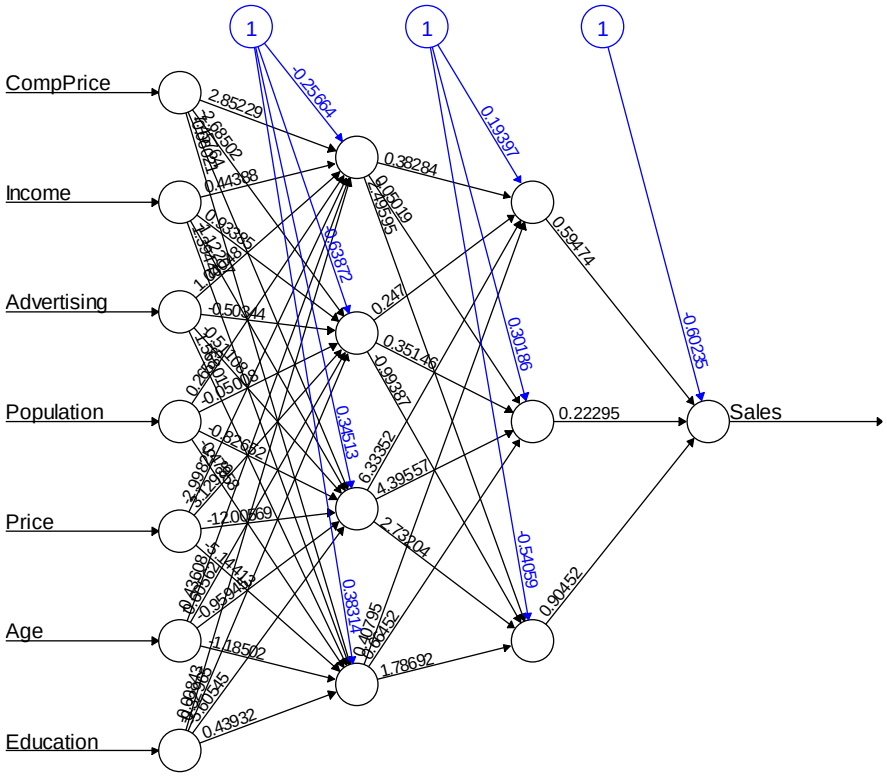
Functia `scale()` returneaza o matrice care este apoi convertita la un `data.frame`.

Pasul 2. Parametrii retelei

Nu exista nici o regula teoretica care sa ne spuna cate nivele si cati neuroni sa aiba o retea, dar sunt mai multe reguli practice. Din punct de vedere practic, un singur nivel ascuns este suficient pentru foarte multe aplicatii. Numarul de neuroni din nivelul ascuns ar trebui sa fie cuprins intre numarul de neuroni din nivelul de intrare si cel de iesire (se poate folosi ca valoare 2/3 din numarul neuronilor de intrare). Subliniem faptul ca acestea sunt doar reguli practice dar care nu garanteaza o solutie optima iar cea mai buna solutie este experimentarea.

In exemplul nostru vom folosi o retea cu doua nivele ascunse in configuratia: 7:4:3:1. 7 reprezinta numarul variabilelor de intrare iar 1 reprezinta singura variabila de iesire (variabila raspuns `medv`).

```
n <- names(train_)
f <- as.formula(paste("Sales ~", paste(n[!n %in% "Sales"], collapse = " +
")))
nn <- neuralnet(f,data=train_,hidden=c(4,3),rep = 10, linear.output=T)

#reprezentare grafica a retelei
plot(nn)
```



Pasul 3. Predictia valorilor Sales folosind reteaua neuronala

Vom incerca sa efectuam predictii asupra valorilor variabilei raspuns Sales din setul de test si vom calcula MSE. Trebuie sa tinem cont de faptul ca reteaua va produce rezultate normalizate si va trebui sa scalam aceste rezultate in sens invers pentru a obtine valori cu semnificatie.

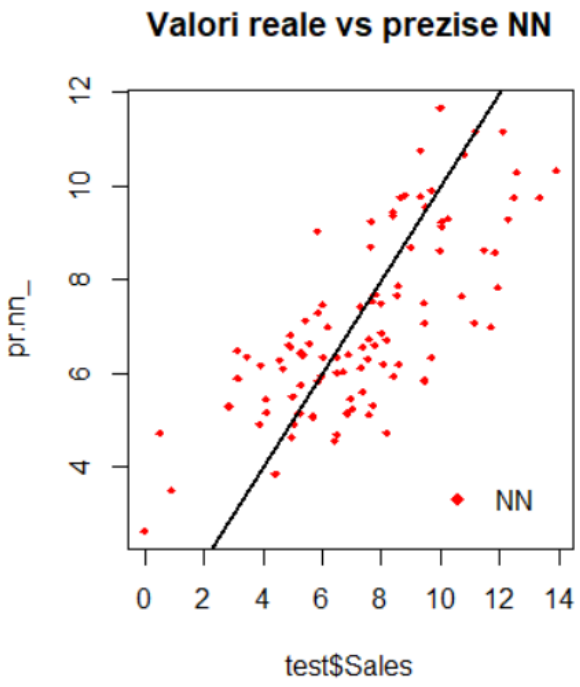
```
#aplicam reteaua neuronala pentru setul de date de test
pr.nn <- neuralnet::compute(nn,test_)

pr.nn_ <- pr.nn$net.result*(max(data$Sales)- min(data$Sales))+
min(data$Sales)
test.r <- (test_$Sales)*(max(data$Sales)-min(data$Sales))+min(data$Sales)

#calculm MSE pentru setul de date de test
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)
print(paste(MSE.lm,MSE.nn))
"3.52285453150786 3.64166927396841"
```

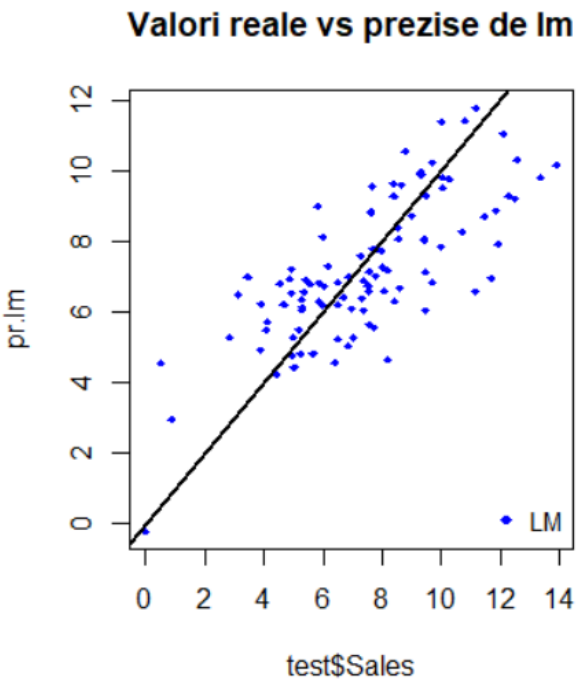
Observatii: Erorile obtinute in cele doua cazuri (regresie liniara si retea neuronala) sunt comparabile, dar eroarea in cazul modelului de regresie este mai mica.

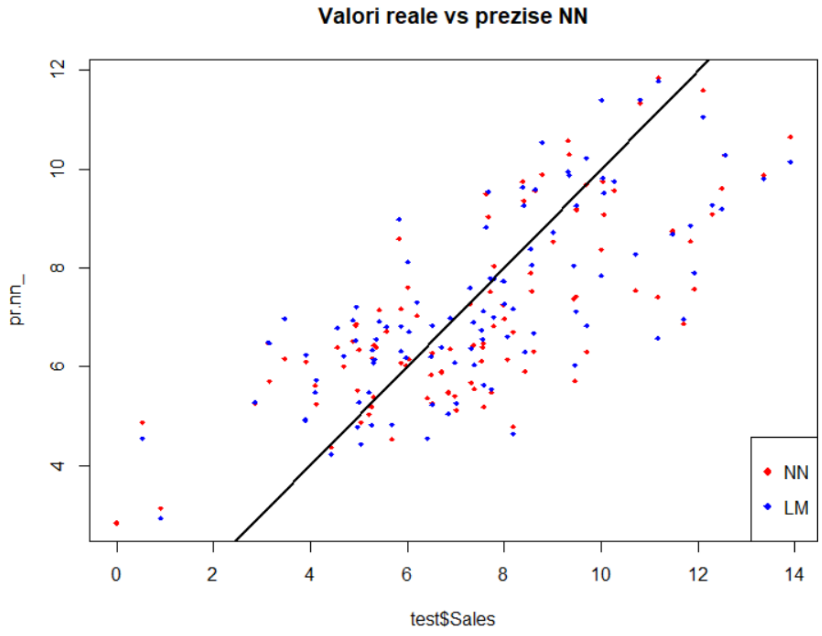
Tema: Aplicati metoda validarii incrucisate pentru a confirma rezultatele de mai sus.

O comparatie intre performantele modelului de regresie liniara si ale retelei neuronale este prezentata mai jos.

```
#comparatie intre performatele modelului liniar si al retelei neuronale
par(mfrow=c(1,2))

plot(test$Sales,pr.nn_,col='red',main='Valori reale vs prezise
NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')

plot(test$Sales,pr.lm,col='blue',main='Valori reale vs prezise de
lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)

par(mfrow=c(1,1))
plot(test$Sales,pr.nn_,col='red',main='Valori reale vs prezise
NN',pch=18,cex=0.7)
points(test$Sales,pr.lm,col='blue',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend=c('NN','LM'),pch=18,col=c('red','blue'))
```


Valori reale vs prezise NN
pr.nn_
test$Sales
NN
0
2
4
6
8
10
12
14


Valori reale vs prezise de lm
pr.lm
test$Sales
LM
0
2
4
6
8
10
12
14


Valori reale vs prezise NN
pr.nn_
test$Sales
NN
LM
4
6
8
10
12
0
2
4
6
8
10
12
14

2. Transformam valorile non-numerice in valori numerice. Facem urmatoarele conventii Yes = 1, No = 0 / Bad = 0, Medium = 1, Good = 2.

Codul R este prezentat mai jos.

```
#optiunea 2 - transformam valorile non-numerice in valori numerice
set.seed(123)
data <- Carseats

#transformam coloanele factor in character
data$ShelveLoc <- as.character(data$ShelveLoc)
data$Urban <- as.character(data$Urban)
data$US <- as.character(data$US)


#valori unice pe coloana SelvefLoc
unique(data$ShelveLoc)
#conventie: bad = 0, medium  = 1 good = 2
indexBad <- which(data$ShelveLoc=="Bad")
data[indexBad,'ShelveLoc'] <- 0
indexMedium<-which(data$ShelveLoc=="Medium")
data[indexMedium,'ShelveLoc'] <- 1
indexGood<-which(data$ShelveLoc=="Good")
data[indexGood,'ShelveLoc'] <- 2

#conventie yes = 1 no = 0
indexYes <- which(data$Urban =="Yes")
data[indexYes,'Urban'] <- 1

indexNo <- which(data$Urban =="No")
data[indexNo,'Urban'] <- 0

indexYes <- which(data$US =="Yes")
data[indexYes,'US'] <- 1

indexNo <- which(data$US =="No")
data[indexNo,'US'] <- 0

#transformam cele  3 coloane in tipul numeric
data$ShelveLoc <- as.numeric(data$ShelveLoc)
data$Urban <- as.numeric(data$Urban)
data$US <- as.numeric(data$US)

#impartim setul de date in 2 subseturi: antrenare(75%) si testare (25%)
index <- sample(1:nrow(data),round(0.75*nrow(data)))
train <- data[index,]
test <- data[-index,]

#estimam un model de regresie liniara
lm.fit <- glm(Sales~., data=train)
summary(lm.fit)
```

```
Call:
glm(formula = Sales ~ ., data = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.54619  -0.71957  -0.03721   0.64164   2.82378

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.371e+00  7.061e-01   7.607 3.97e-13 ***
CompPrice    9.536e-02  4.773e-03  19.978  < 2e-16 ***
Income       1.418e-02  2.167e-03   6.545 2.71e-10 ***
Advertising  1.264e-01  1.293e-02   9.775  < 2e-16 ***
Population   9.323e-05  4.310e-04   0.216    0.829
Price       -9.480e-02  3.098e-03 -30.602  < 2e-16 ***
ShelveLoc    2.435e+00  8.843e-02  27.534  < 2e-16 ***
Age         -4.747e-02  3.736e-03 -12.707  < 2e-16 ***
Education   -3.317e-02  2.263e-02  -1.466    0.144
Urban        9.109e-02  1.311e-01   0.695    0.488
US          -2.492e-01  1.756e-01  -1.419    0.157
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.056031)

    Null deviance: 2412.35  on 299  degrees of freedom
Residual deviance:  305.19  on 289  degrees of freedom
AIC: 880.51



Number of Fisher Scoring iterations: 2

#folosind modelul estimat calculam valorile variabilei raspuns pentru setul
de date de test
pr.lm <- predict(lm.fit,test)

#calculam MSE pentru setul de date de test
MSE.lm <- sum((pr.lm - test$Sales)^2)/nrow(test)
MSE.lm

[1] 1.248357

#pregatim datele pentru a fi utilizate cu o retea neuronala
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)

scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins))
```

```
#impartim setul de date in doua subseturi: unul de antrenare si altul de
test
train_ <- scaled[index,]
test_ <- scaled[-index,]

#construim reteaua neuronala si estimam parametrii
n <- names(train_)
f <- as.formula(paste("Sales ~", paste(n[!n %in% "Sales"], collapse = " +
")))
nn <- neuralnet(f,data=train_,hidden=c(7),rep=10, linear.output=T)

#reprezentare grafica a retelei
plot(nn)

#aplicam reteaua neuronala pentru setul de date de test
pr.nn <- neuralnet::compute(nn,test_)

pr.nn_ <- pr.nn$net.result * (max(data$Sales) - min(data$Sales)) +
min(data$Sales)
test.r <- (test_$Sales)*(max(data$Sales) - min(data$Sales))+
min(data$Sales)

#calculm MSE pentru setul de date de test
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)

print(paste(MSE.lm,MSE.nn))

[1] "1.24835699543199 1.22169729843482"

#comparatie intre performatele modelului liniar si al retelei neuronale
par(mfrow=c(1,2))

plot(test$Sales,pr.nn_,col='red',main='Valori reale vs prezise
NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')

plot(test$Sales,pr.lm,col='blue',main='Valori reale vs prezise de
lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)

par(mfrow=c(1,1))

plot(test$Sales,pr.nn_,col='red',main='Valori reale vs prezise
NN',pch=18,cex=0.7)
points(test$Sales,pr.lm,col='blue',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend=c('NN','LM'),pch=18,col=c('red','blue'))
```

Se observa ca in acest caz eroarea obtinuta de reteaua neuronala este mai mica decat in cazul modelului de regresie liniara.

Validam acest lucru prin procedeul cross-validation cu k = 10

```
#validare incrucisata cu K=10
#incepem cu modelul liniar
set.seed(123)
lm.fit <- glm(Sales~.,data=data)
cv.glm(data,lm.fit,K=10)$delta[1]
[1] 1.125338

#acum trecem la reteaua neuronala
cv.error <- NULL
k <- 10

library(plyr)
pbar <- create_progress_bar('text')
pbar$init(k)

for(i in 1:k){
    index <- sample(1:nrow(data),round(0.9*nrow(data)))
    train.cv <- scaled[index,]
    test.cv <- scaled[-index,]

    nn <- neuralnet(f,data=train.cv,hidden=c(7),rep = 10, linear.output=T)

    pr.nn <- neuralnet::compute(nn,test.cv)
    pr.nn              <-              pr.nn$net.result*(max(data$Sales)-
min(data$Sales))+min(data$Sales)

    test.cv.r             <-             (test.cv$Sales)*(max(data$Sales)-
min(data$Sales))+min(data$Sales)

    cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv)

    pbar$step()
}

mean(cv.error)
[1] 1.640759
```

Se observa ca desi valorile erorii in cele doua cazuri sunt foarte apropiate, totusi modelul de regresie liniara da rezultate mai bune.

Obervatie finala

Retele neuronale se aseamana cu o cutie neagra: explicarea output-ului este mult mai dificila decat in cazul unui model linear. In plus, trebuie luate o serie de precautii inainte de utilizarea retelei.

Problema 2.

Dorim sa contruim un model care sa prezica daca un student ce participa la un interviu de angajare va fi sau nu angajat.
Pentru aceasta avem la dispozitie doua seturi de date:

- Primul set contine punctajul obtinut de participantii la interviul de angajare la intrebarile tehnice (variabila TKS);
- Al doilea set contine punctajul obtinut de participantii la interviul de angajare la intrebarile privind abilitatile de comunicare (variabila CSS);

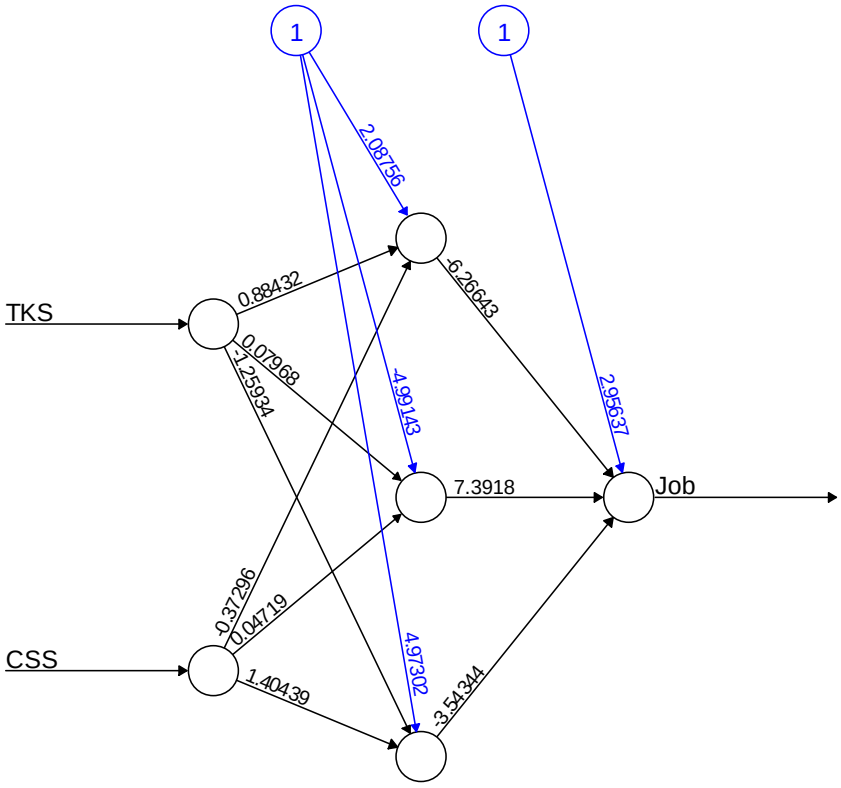
Pentru un set de 9 participanti la interviu a fost inregistrat rezultatul sub forma unei variabile cu doua valori: 1 – studentul a fost angajat, 0 – studentul nu a fost angajat.

TKS	CSS	Job
20	90	1
10	20	0
30	40	0
20	50	0
80	50	1
30	80	1
30	85	1
40	50	0
85	40	1

Rezolvare:
Divizam setul de date in doua subseturi: unul de antrenare a retelei, al doilea de testare. Primul subset va avea primele 6 inregistrari, al doilea urmatoarele 3.
Codul sursa R este prezentat in continuare:

```
library(neuralnet)

#setul de date de antrenament
TKS=c(20,10,30,20,80,30)
CSS=c(90,20,40,50,50,80)
Job=c(1,0,0,0,1,1)
df=data.frame(TKS,CSS,Job)

#construim reteaua neuronala
nn=neuralnet(Job~TKS+CSS,data=df,   hidden=3,   act.fct   =   "logistic",
linear.output = FALSE)
```

```
plot(nn)


#setul de date de test
TKS_test=c(30,40,85)
CSS_test=c(85,50,40)
Job_test=c(1,0,1)
test=data.frame(TKS_test,CSS_test)

#utilizam reteaua pentru a prezice valorile variabilei Job
predicted_job=compute(nn,test)
predicted_job$net.result
#calculam eroarea de predictie
MSE.nn <- sum((predicted_job$net.result - Job_test)^2)/nrow(test)
MSE.nn

# transformam valorile obtinute in 1 si 0
prob <- predicted_job$net.result
pred <- ifelse(prob > 0.5, 1, 0)
pred
```



Error: 0.001184 Steps: 81