IBM Employers' attrition

Giulia Chiaretti (800928), Federica Fiorentini (807124)

Stare bene in azienda significa far star bene l'azienda. Quello che puÃ² sembrare all'apparenza un gioco di parole Ã¨, nei fatti, una realtÃ di cui prendere atto. Nella realtÃ imprenditoriale una delle sfide maggiori al giorno d'oggi Ã¨ riuscire ad impostare una politica di welfare aziendale corretta, che consiste nell'offrire servizi e prestazioni che migliorino la qualitÃ di vita sul luogo di lavoro.

Per evidenziare i fattori principali che portano un dipendente ad abbandonare la propria azienda Ã¨ stato impostato un problema di classificazione utilizzando un dataset fornito dalla piattaforma Kaggle.

Il dataset Ã¨ riferito a 1470 dipendenti della societÃ multinazionale IBM, su cui sono state osservate 35 variabili:

- *Age*: variabile numerica riferita all'etÃ del dipendente;
- *Attrition*: variabile target booleana che indica il licenziamento da parte del dipendente(0=No, 1=Si);
- *BusinessTravel*: variabile categorica che indica la frequenza di trasferte (1="No Travel", 2="Travel Farely", 3="Travel Frequently");
- *DailyRate*: variabile numerica che indica lo stipendio giornaliero;
- *Department*: variabile categorica che indica il settore d'impiego (1="HR", 2="R&D", 3="Sales");
- *DistanceFromHome*: variabile numerica che indica la distanza casa-lavoro misurata in miglia;
- *Education*: variabile categorica che indica il livello di istruzione (1='Below College'(/licenza media), 2='College'(/diploma superiore), 3='Bachelor'(/laurea triennale), 4='Master'(/laurea magistrale o master), 5='Doctor' (/dottorato di ricerca);
- *EducationField*: variabile categorica che indica area di studio (1=HR, 2=LIFE SCIENCES, 3=MARKETING, 4=MEDICAL SCIENCES, 5=OTHERS, 6= TEHCNICAL);
- *EmployeeCount*: variabile di dubbio significato, sempre pari ad 1;
- *EmployeeNumber*: ID del dipendente;
- *EnvironmentSatisfaction*: variabile categorica che indica il grado di apprezzamento del contesto lavorativo (1 'Low', 2 'Medium', 3 'High', 4 'Very High');
- *Gender*: variabile binaria che indica il sesso del dipendente (1=femmina, 2=maschio);
- *HourlyRate*: variabile numerica che indica il compenso orario;
- *JobInvolvement*: variabile categorica che indica il coinvolgimento nell'ambiente di lavoro (1 'Low', 2 'Medium', 3 'High', 4 'Very High');
- *JobLevel*: variabile categorica che indica il livello all'interno dell'azienda (da 1=junior a 5=partner);
- *JobRole*: variabile categorica che indica il ruolo (categorica 1=HC REP, 2=HR, 3=LAB TECHNICIAN, 4=MANAGER, 5= MANAGING DIRECTOR, 6= REASEARCH DIRECTOR, 7= RESEARCH SCIENTIST, 8=SALES EXECUTIEVE, 9= SALES REPRESENTATIVE);
- *JobSatisfaction*: variabile categorica che indica il grado di soddisfazione del lavoro svolto (1 'Low', 2 'Medium', 3 'High', 4 'Very High');
- *MaritalStatus*: variabile categorica che indica lo stato civile (1= divorced, 2= married, 3 =single);
- *MonthlyIncome*: variabile numerica che indica lo stipedio mensile;
- *MonthlyRate*:
- *NumCompaniesWorked*: variabile numerica che indica il numero di compagnie in cui ha lavorato prima di quella attuale;
- *Over18*: variabile booleana che indica se il dipendente Ã¨ maggiorenne (1=si, 2=no);
- *OverTime*: variabile booleana che indica se il dipendente svolge del lavoro straordinario (1=no, 2=yes);
- *PercentSalaryHike*: variabile numerica che indica l' aumento di stipendio percentuale tra il 2015 e il 2016;
- *PerformanceRating*: variabile categorica che indica la valutazione delle performance del dipendente (1 'Low', 2 'Good', 3 'Excellent', 4 'Outstanding');
- *RelationshipSatisfaction*: variabile categorica che indica il grado di apprezzamento delle relazioni in ambito lavorativo (1 'Low', 2 'Medium', 3 'High', 4 'Very High');

- *StandardHours*: variabile numerica di dubbia interpretazione e pari sempre ad 80;
- *StockOptionLevel*: indice che misura diritto di acquistare un determinato ammontare di azioni della societÃ (da 1= basso a 3 = alto);
- *TotalWorkingYears*: variabile numerica che indica gli anni di esperienza lavorativa;
- *TrainingTimesLastYear*: variabile numerica che indica il totale di ore di formazione svolte;
- *WorkLifeBalance*: variabile categorica basata sul rapporto tra tempo speso al lavoro e tempo libero (1 'Bad', 2 'Good', 3 'Better', 4 'Best')
- *YearsAtCompany*: variabile numerica che indica il numero di anni lavorativi nell'azienda;
- *YearsInCurrentRole*: variabile numerica che indica il numero di anni da cui il dipendente riveste lo stesso ruolo;
- *YearsSinceLastPromotion*: variabile numerica che indica il numero di anni trascorsi dall'ultima promozione;
- *YearsWithCurrManager*: variabile numerica che indica numero di anni che il dipendente sotto lo stesso manager.

```
library('arules')
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library("ggplot2")
library("tidyr")
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:arules':
##
##     intersect, recode, setdiff, setequal, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
library("miscset")
```

```
##
## Attaching package: 'miscset'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
## The following object is masked from 'package:arules':
##
##     info
```

```
library("purrr")
library('knitr')
library('gridExtra')
```

```
##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library('caTools')
library('e1071')
library('glmnet')
```

```
## Loading required package: foreach
##
## Attaching package: 'foreach'
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
## Loaded glmnet 2.0-18
```

```
library('ROSE')
```

```
## Loaded ROSE 0.0-3
```

```
library('psych') #factor analysis
```

```
##
## Attaching package: 'psych'
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library('GPArotation') #factor analysis
library('robustHD') #per a funzione standardize
```

```
## Loading required package: perry
## Loading required package: parallel
## Loading required package: robustbase
##
## Attaching package: 'robustbase'
## The following object is masked from 'package:psych':
##
##     cushny
```

```
library('MASS') #per LDA
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library('caret') #confusion matrix e importance
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library('mlbench') #feature importance
library('randomForest')
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library('rpart')  #decision tree
library('rpart.plot')
library('pander')
library('klaR')
```

Importazione del file csv

```
data <- read.csv("C:/Users/GiuliaChiaretti(Stag/Desktop/Data Science Lab/PROGETTO/WA_Fn-UseC_-HR-Employ
colnames(data)[1] <- "Age"

data=data[,-c(9,10,22,27)]


summary(data)
```

```
##       Age        Attrition            BusinessTravel   DailyRate
##  Min.   :18.00   No :1233   Non-Travel       : 150   Min.   : 102.0
##  1st Qu.:30.00   Yes: 237   Travel_Frequently: 277   1st Qu.: 465.0
##  Median :36.00              Travel_Rarely    :1043   Median : 802.0
##  Mean   :36.92                                       Mean   : 802.5
##  3rd Qu.:43.00                                       3rd Qu.:1157.0
##  Max.   :60.00                                       Max.   :1499.0
##
##                   Department  DistanceFromHome   Education
##  Human Resources       : 63   Min.   : 1.000   Min.   :1.000
##  Research & Development:961   1st Qu.: 2.000   1st Qu.:2.000
##  Sales                 :446   Median : 7.000   Median :3.000
##                               Mean   : 9.193   Mean   :2.913
##                               3rd Qu.:14.000   3rd Qu.:4.000
##                               Max.   :29.000   Max.   :5.000
##
##           EducationField EnvironmentSatisfaction    Gender
##  Human Resources : 27    Min.   :1.000           Female:588
##  Life Sciences   :606    1st Qu.:2.000           Male  :882
##  Marketing       :159    Median :3.000
##  Medical         :464    Mean   :2.722
##  Other           : 82    3rd Qu.:4.000
##  Technical Degree:132    Max.   :4.000
##
##    HourlyRate     JobInvolvement    JobLevel
##  Min.   : 30.00   Min.   :1.00   Min.   :1.000
##  1st Qu.: 48.00   1st Qu.:2.00   1st Qu.:1.000
##  Median : 66.00   Median :3.00   Median :2.000
##  Mean   : 65.89   Mean   :2.73   Mean   :2.064
##  3rd Qu.: 83.75   3rd Qu.:3.00   3rd Qu.:3.000
##  Max.   :100.00   Max.   :4.00   Max.   :5.000
##
##                       JobRole    JobSatisfaction  MaritalStatus
##  Sales Executive          :326   Min.   :1.000   Divorced:327
##  Research Scientist       :292   1st Qu.:2.000   Married :673
##  Laboratory Technician    :259   Median :3.000   Single  :470
##  Manufacturing Director   :145   Mean   :2.729
##  Healthcare Representative:131   3rd Qu.:4.000
##  Manager                  :102   Max.   :4.000
##  (Other)                  :215
##  MonthlyIncome    MonthlyRate    NumCompaniesWorked OverTime
##  Min.   : 1009   Min.   : 2094   Min.   :0.000      No :1054
##  1st Qu.: 2911   1st Qu.: 8047   1st Qu.:1.000      Yes: 416
##  Median : 4919   Median :14236   Median :2.000
##  Mean   : 6503   Mean   :14313   Mean   :2.693
##  3rd Qu.: 8379   3rd Qu.:20462   3rd Qu.:4.000
##  Max.   :19999   Max.   :26999   Max.   :9.000
##
##  PercentSalaryHike PerformanceRating RelationshipSatisfaction
##  Min.   :11.00     Min.   :3.000     Min.   :1.000
##  1st Qu.:12.00     1st Qu.:3.000     1st Qu.:2.000
##  Median :14.00     Median :3.000     Median :3.000
##  Mean   :15.21     Mean   :3.154     Mean   :2.712
##  3rd Qu.:18.00     3rd Qu.:3.000     3rd Qu.:4.000
```

```
##  Max.   :25.00     Max.   :4.000     Max.   :4.000
##
##  StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance
##  Min.   :0.0000   Min.   : 0.00     Min.   :0.000         Min.   :1.000
##  1st Qu.:0.0000   1st Qu.: 6.00     1st Qu.:2.000         1st Qu.:2.000
##  Median :1.0000   Median :10.00     Median :3.000         Median :3.000
##  Mean   :0.7939   Mean   :11.28     Mean   :2.799         Mean   :2.761
##  3rd Qu.:1.0000   3rd Qu.:15.00     3rd Qu.:3.000         3rd Qu.:3.000
##  Max.   :3.0000   Max.   :40.00     Max.   :6.000         Max.   :4.000
##
##  YearsAtCompany   YearsInCurrentRole YearsSinceLastPromotion
##  Min.   : 0.000   Min.   : 0.000     Min.   : 0.000
##  1st Qu.: 3.000   1st Qu.: 2.000     1st Qu.: 0.000
##  Median : 5.000   Median : 3.000     Median : 1.000
##  Mean   : 7.008   Mean   : 4.229     Mean   : 2.188
##  3rd Qu.: 9.000   3rd Qu.: 7.000     3rd Qu.: 3.000
##  Max.   :40.000   Max.   :18.000     Max.   :15.000
##
##  YearsWithCurrManager
##  Min.   : 0.000
##  1st Qu.: 2.000
##  Median : 3.000
##  Mean   : 4.123
##  3rd Qu.: 7.000
##  Max.   :17.000
##
```

Sono state eliminate 4 variabili perchÃ¨ inutili a discriminare: - *EmployeeCount*: variabile sempre pari ad 1 e di dubbio significato; - *EmployeeNumber*: ID del dipendente; - *Over18*: perchÃ¨ sempre pari ad 1 (tutti i dipendenti sono maggiorenni); - *StandardHours*: variabile di dubbio significato e sempre pari a 80.

Analizzando il summary delle variabili Ã¨ possibile osservare che non sono presenti NA.

Alcune variabili (WorkLifeBalance, StockOptionLevel, PerformanceRating, JobSatisfaction, RelationshipSatisfaction, JobLevel, JobInvolvement, EnvironmentSatisfaction e Education) sono categoriche ma sono state importate come interi e, quindi, si procede con la modifica del type.

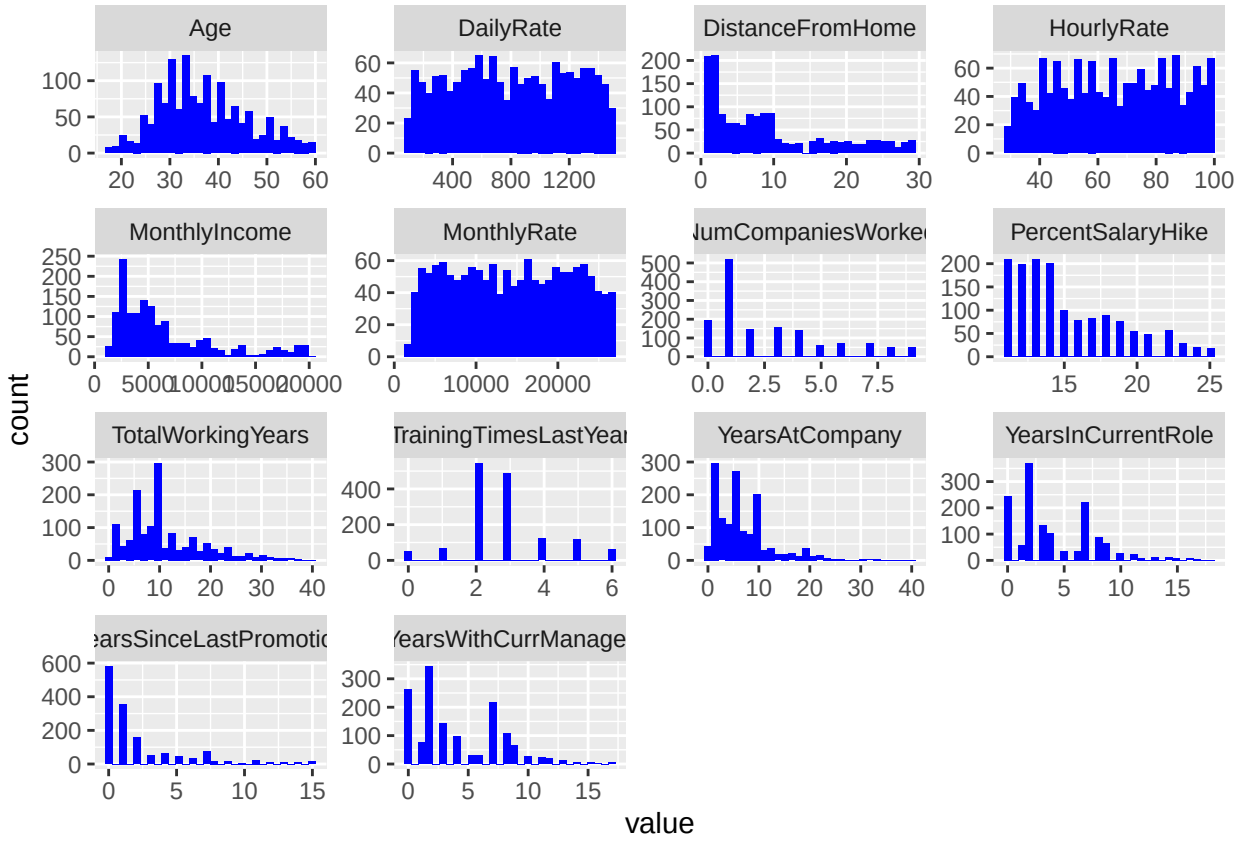
```
names <- c('WorkLifeBalance' ,'StockOptionLevel','PerformanceRating','JobSatisfaction',
           'RelationshipSatisfaction','JobLevel','JobInvolvement','EnvironmentSatisfaction','Education')
data[,names] <- lapply(data[,names] , factor)
```

Data esploration

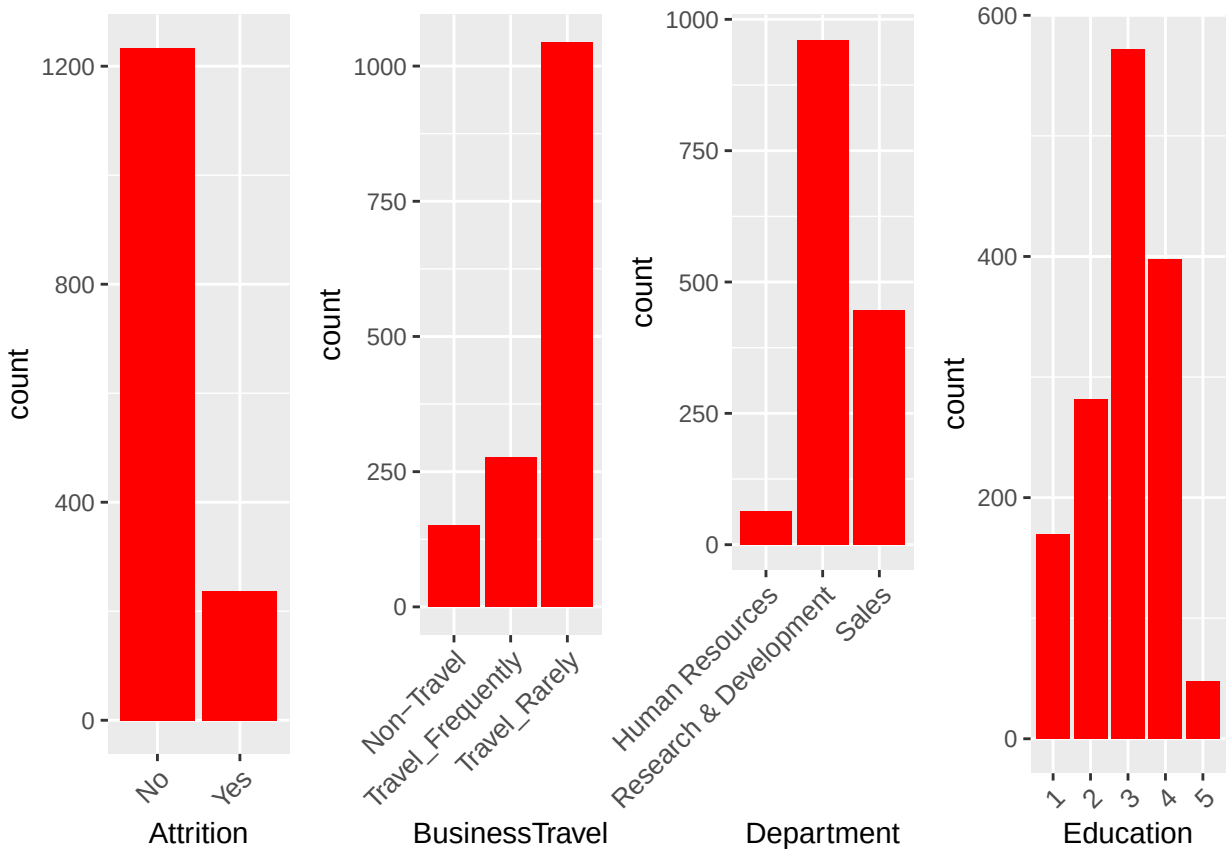
Analisi univariata delle variabili

```
#variabili numeriche

data[1:31] %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(x = value)) +
  facet_wrap(~ key, scales = "free") +
  geom_histogram(fill = "blue")
```

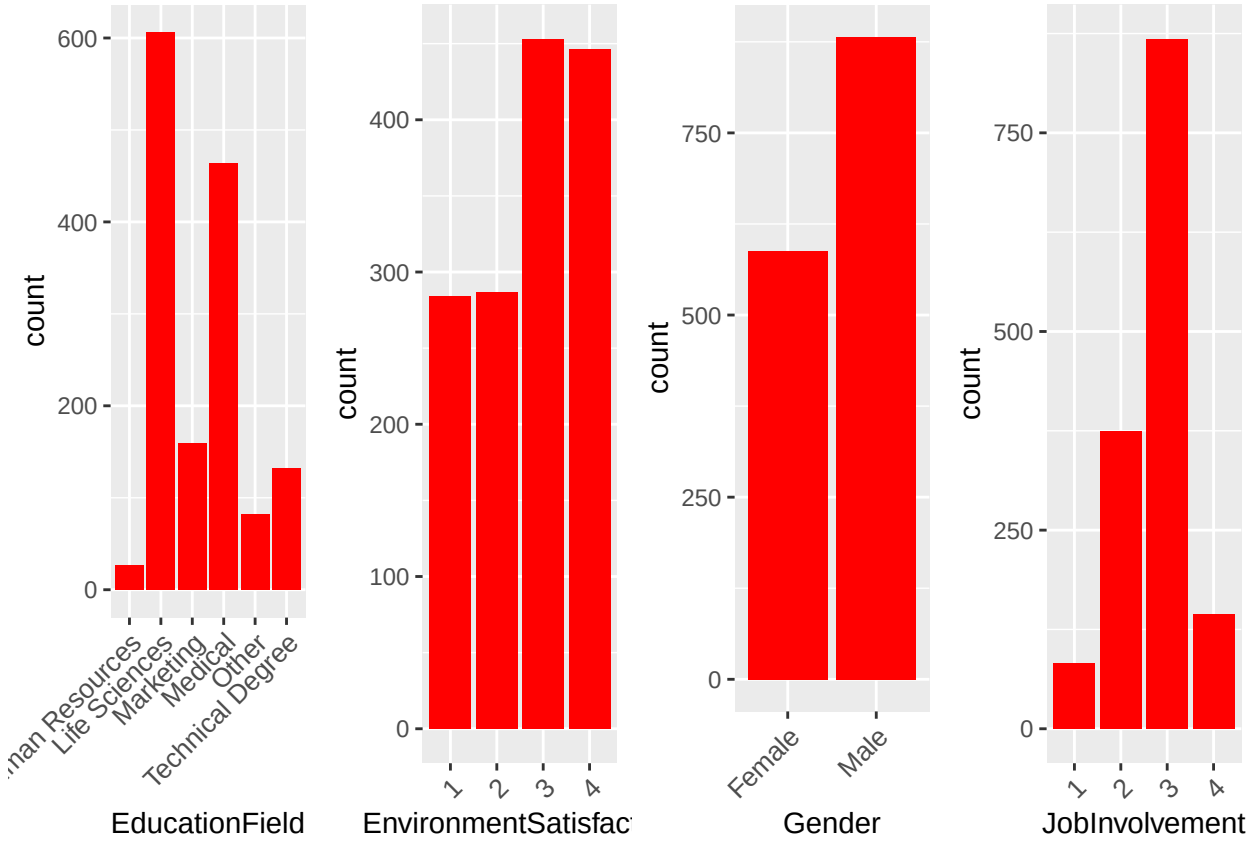
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



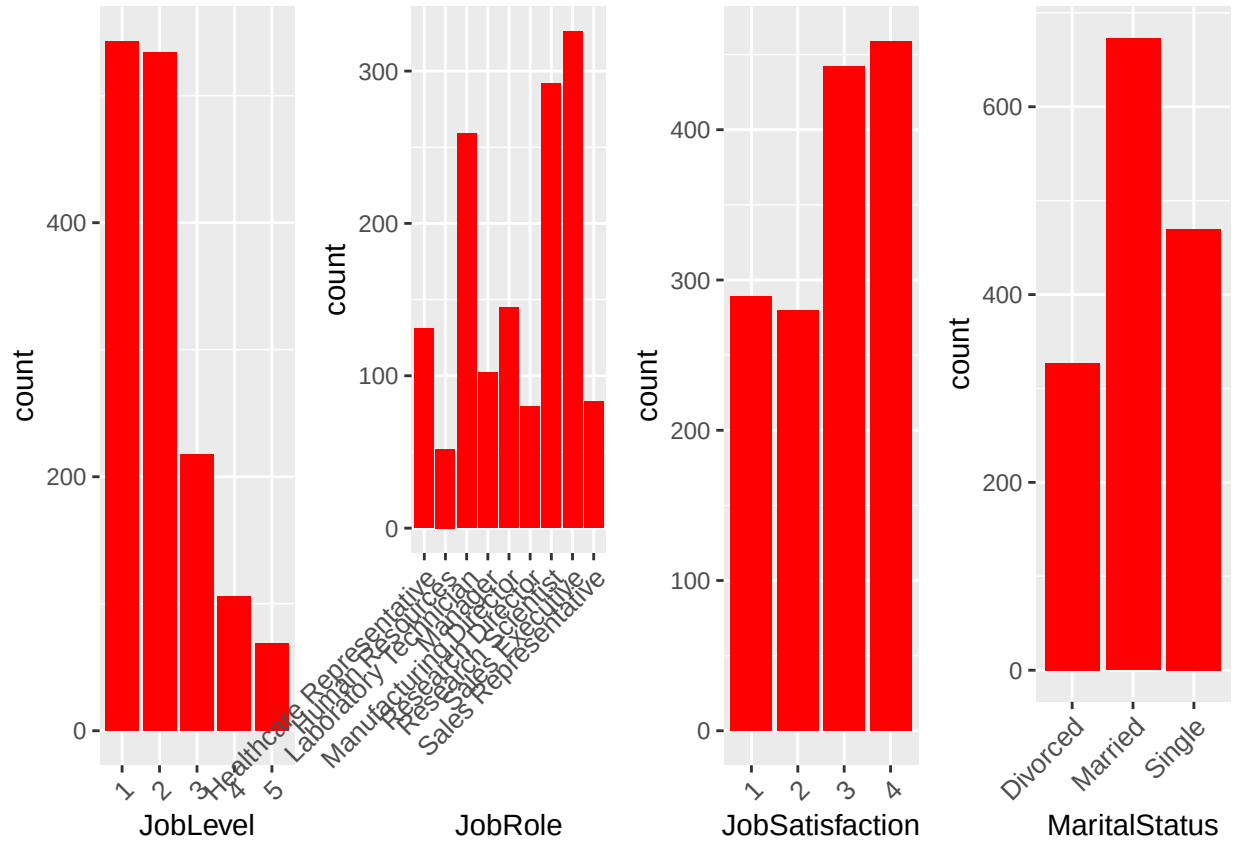

```
#variabili categoriche

var_cat <- names(which(sapply(data,is.factor)))

ggplotGrid(ncol = 4,lapply(var_cat[1:4],function(col) {
  ggplot(data, aes_string(col)) + geom_bar(fill = "red") +
    theme(axis.text.x = element_text(size  = 10,
                                     angle = 45,
                                     hjust = 1,
                                     vjust = 1))
        }))
```



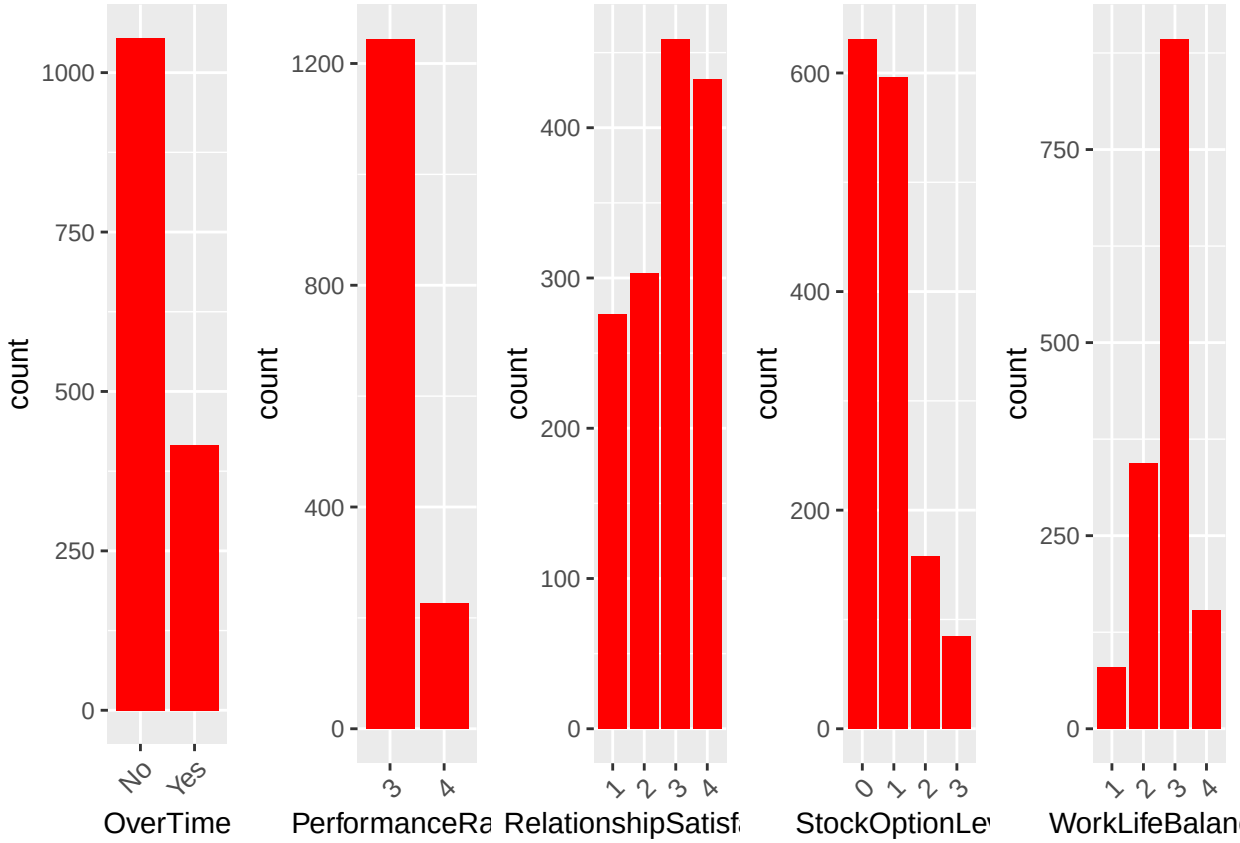

```
ggplotGrid(ncol = 4,lapply(var_cat[5:8],function(col) {
  ggplot(data, aes_string(col)) + geom_bar(fill = "red") +
    theme(axis.text.x = element_text(size  = 10,
                                     angle = 45,
                                     hjust = 1,
                                     vjust = 1))
        }))
```




```
ggplotGrid(ncol = 4,lapply(var_cat[9:12],function(col) {
  ggplot(data, aes_string(col)) + geom_bar(fill = "red") +
    theme(axis.text.x = element_text(size  = 10,
                                     angle = 45,
                                     hjust = 1,
                                     vjust = 1))
        }))
```




```
ggplotGrid(ncol = 5,lapply(var_cat[13:17],function(col) {
  ggplot(data, aes_string(col)) + geom_bar(fill = "red") +
    theme(axis.text.x = element_text(size  = 10,
                                     angle = 45,
                                     hjust = 1,
                                     vjust = 1))
        }))
```



Data pre-processing

** FEATURE SELECTION**

Il numero delle variabili ottenuto a seguito dell'eliminazione di alcune di esse, Ã¨ pari a 31 attributi. Aumentare la complessitÃ del modello con una dimensionalitÃ elevata, non sempre aiuta a migliore la capacitÃ predittiva del modello di classificazione. A tal proposito, si effettua un'analisi delle variabili rimanenti con l'obiettivo di selezionare solamente quelle piÃ¹ utili a spiegare la variabile target.

Si procede, quindi, con una feature selction rispetto alle variabili categoriche, effettuata tramite il test *Chi-Quadro* che testa l'ipotesi di indipendenza degli attributi con la variabile target.

```
data_factor=data[,which(sapply(data,is.factor))]

chi_test=function(dataset) {
  matrice=matrix(NA, ncol=3,nrow=dim(dataset)[2]-1)
for (i in 1:dim(dataset)[2]-1) {
  matrice[i,1]=colnames(dataset)[i+1]
  matrice[i,2]=chisq.test(table(dataset$Attrition,dataset[,i+1]))$p.value
  matrice[i,3]=chisq.test(table(dataset$Attrition,dataset[,i+1]))$statistic
}
  colnames(matrice)=c("variabile categorica","p-value","statistica test")
  return(kable(matrice))

}

chi_test(data_factor)
```

variabile categorica	p-value	statistica test
BusinessTravel	5.60861447644993e-06	24.1824136856552
Department	0.00452560657447963	10.7960073224107
Education	0.545525337656595	3.07396139823672
EducationField	0.00677398013902522	16.0246741195854
EnvironmentSatisfaction	5.12346890628942e-05	22.5038814358423
Gender	0.290572449028912	1.1169671241971
JobInvolvement	2.86318063671342e-06	28.4920212346593
JobLevel	6.63468471545892e-15	72.5290131066739
JobRole	2.75248163805065e-15	86.1902536767043
JobSatisfaction	0.000556300451038756	17.505077010348
MaritalStatus	9.45551106034082e-11	46.1636765408487
OverTime	8.15842372153834e-21	87.5642936582877
PerformanceRating	0.990074546593458	0.000154754394282715
RelationshipSatisfaction	0.154972443710526	5.24106785971371
StockOptionLevel	4.37939033610836e-13	60.5983010861222
WorkLifeBalance	0.000972569884534883	16.3250970916474

Dati i risultati del test, si puÃ² rifiutare l'ipotesi nulla di indipendenza con la risposta per tutte le variabile ad eccezione di:

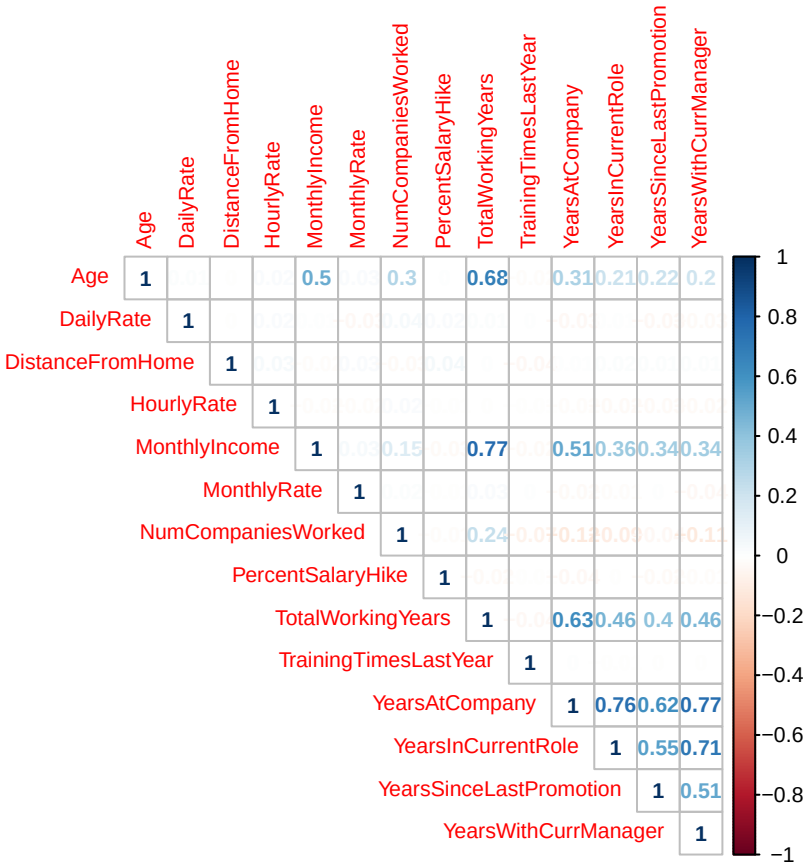
- Education (p-value = 0.54);
- Gender (p-value=0.29);
- PerformanceRating (p-value=0.99);
- RelationshipSatisfaction (p-value=0.15).

Si procede, quindi, con la rimozione delle variabili categoriche sopra elencate che risultano indipendenti dalla variabile risposta.

```
data=data[,-which(names(data) %in% c("Education","Gender","PerformanceRating","RelationshipSatisfaction
```

Per svolgere la feature selection sulle variabili numeriche, Ã¨ utile inizialmente analizzare la correlazione tra esse, in modo tale da evitare la presenza di multicollinearitÃ nei modelli che andrebbe a penalizzarli.

```
var_num <- which(sapply(data,is.numeric))
corrplot(cor(data[var_num]),type = "upper",method='number',tl.cex = .7,cl.cex = .7,number.cex = 0.7)
```

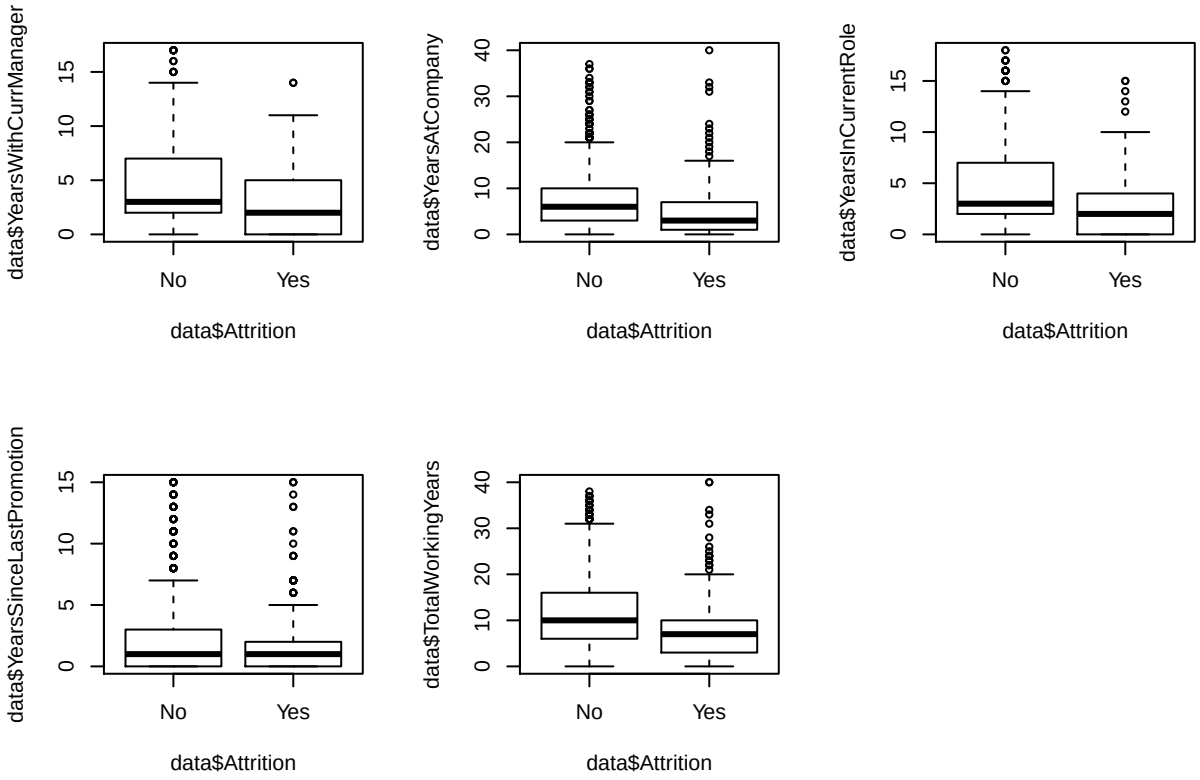


Si osservano delle correlazioni significativamente elevate tra le seguenti variabili:

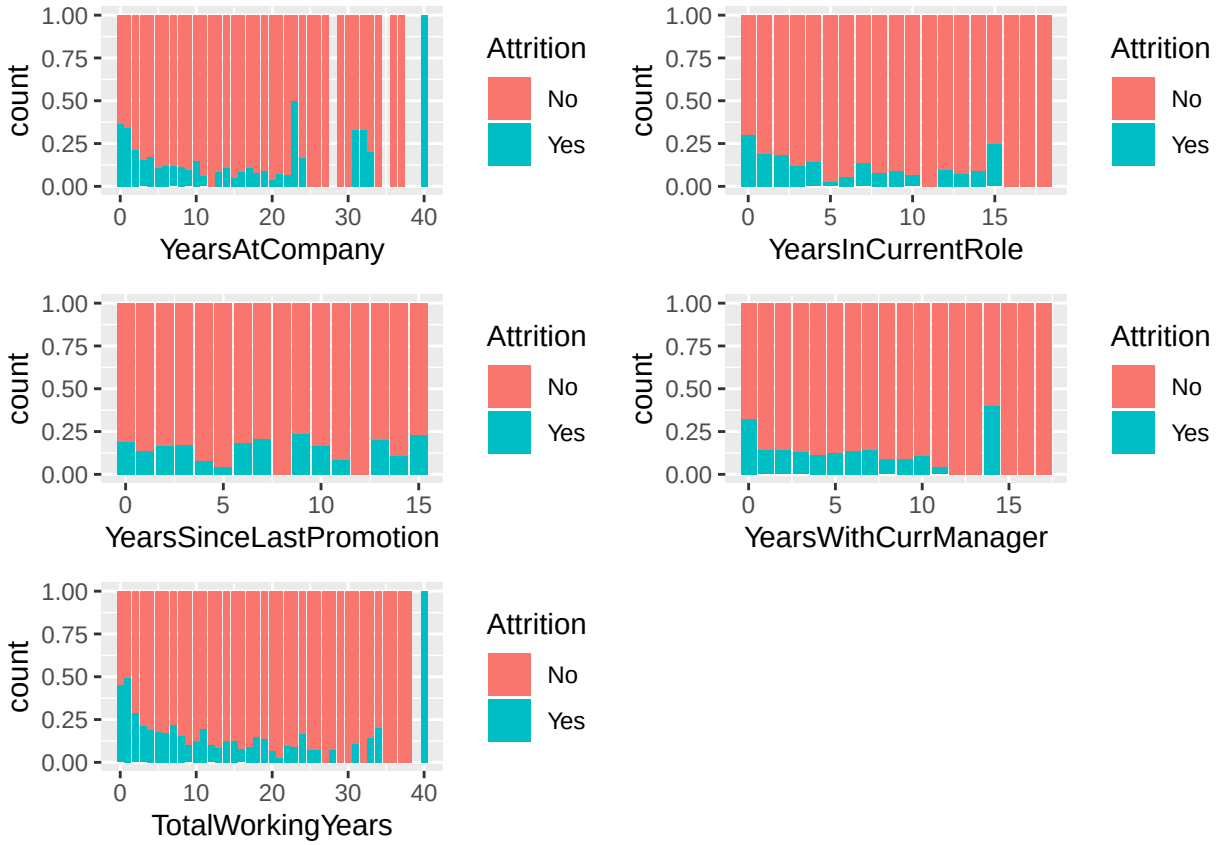
- TotalWorkingYear;
- MontlyIncome;
- TotalWorkingYear;
- YearsAtCompany;
- YearsWithCurrentManager.

Si analizza, quindi, la distribuzione di queste 5 variabili rispetto alla variabile target.

```
par(mfrow=c(2,3))
plot(data$YearsWithCurrManager ~data$Attrition)
plot(data$YearsAtCompany ~data$Attrition)
plot(data$YearsInCurrentRole ~data$Attrition)
plot(data$YearsSinceLastPromotion ~data$Attrition)
plot(data$TotalWorkingYears ~data$Attrition)

par(mfrow=c(1,1))
```




```
distr1 <- ggplot(data,aes(x = YearsAtCompany,fill = Attrition)) +
  geom_bar(position = "fill")

distr2 <- ggplot(data,aes(x = YearsInCurrentRole,fill = Attrition)) +
  geom_bar(position = "fill")

distr3 <- ggplot(data,aes(x = YearsSinceLastPromotion,fill = Attrition)) +
  geom_bar(position = "fill")

distr4 <- ggplot(data,aes(x = YearsWithCurrManager,fill = Attrition)) +
  geom_bar(position = "fill")

distr5 <- ggplot(data,aes(x = TotalWorkingYears,fill = Attrition)) +
  geom_bar(position = "fill")

grid.arrange(distr1, distr2, distr3, distr4, distr5, ncol=2)
```



Dai grafici, si nota che la distribuzione delle variabili riferite agli "anni lavorativi" Ã¨ differente all'interno delle due classi di *Attrition*, ad eccezione della variabile *YearsSinceLastPromotion*. Sembra, quindi, che questa variabile non sia utile a discriminare l'appartenenza alle due classi.

Prima di decidere di rimuoverla dall'analisi, perÃ², si applicano due metodologie che hanno l'obiettivo di creare uno o piÃ¹ fattori che "riassumano", tramite una combinazione lineare, le 5 variabili in questione relative agli "anni lavorativi".

Linear Discriminant Analysis

La prima tecnica utilizzata Ã¨ la *Linear Discriminant Analysis*, una tecnica che permette di trovare una combinazione lineare di variabili per aumentare la separazione tra due o piÃ¹ classi di oggetti.

Una delle ipotesi della LDA Ã¨ la normalitÃ delle variabili che, se non rispettata, non si ha la garanzia di ottenere la soluzione ottimale.

Si effettua, quindi, il test di normalitÃ sulle variabili di interesse, raggruppate nel dataframe "years".

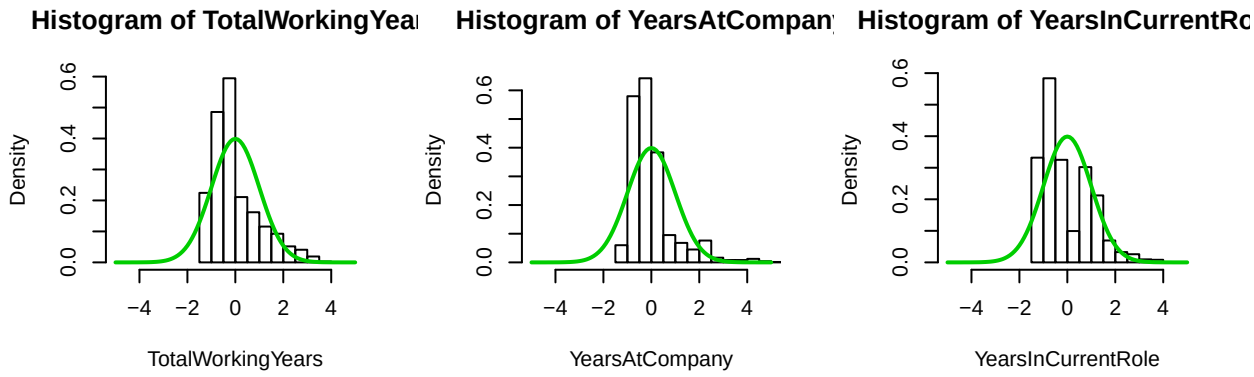
```
years <- data[,which(names(data) %in% c("YearsWithCurrManager","YearsAtCompany","YearsInCurrentRole","Y

shap_test=function(dataset) {
  matrice=matrix(NA, ncol=4,nrow=dim(dataset)[2])
for (i in 1:dim(dataset)[2]) {
  matrice[i,1]=colnames(dataset)[i]
  matrice[i,2]=shapiro.test(dataset[,i])$method
  matrice[i,3]=shapiro.test(dataset[,i])$statistic
  matrice[i,4]=shapiro.test(dataset[,i])$p.value
}
  colnames(matrice)=c("variabile","test method","test statistic","p-value")
```

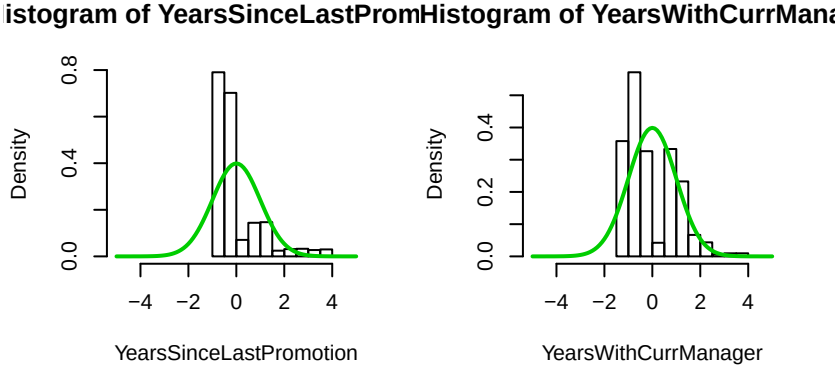
```
  return(kable(matrice))

}

shap_test(years)
```

variabile	test method	test statistic	p-value
TotalWorkingYears	Shapiro-Wilk normality test	0.907429228308795	5.63092202468782e-29
YearsAtCompany	Shapiro-Wilk normality test	0.838992773412801	3.66886709614548e-36
YearsInCurrentRole	Shapiro-Wilk normality test	0.89618655128116	2.142707144451e-30
YearsSinceLastPromotion	Shapiro-Wilk normality test	0.703727752949589	4.77296485294073e-45
YearsWithCurrManager	Shapiro-Wilk normality test	0.897456525718067	3.05579557946192e-30

Nessuna delle variabili "years" risulta seguire una distribuzione normale. Si procede, perciÃ², con la standardizzazione di queste per poi effettuare la LDA.

```
stand=function(dataset) {
  years_s = years
for (i in 1:dim(dataset)[2]) {
  years_s[,i]=standardize(dataset[,i], centerFun = mean, scaleFun = sd)
}
  return(years_s)

}

years_stand=stand(years)
par(mfrow=c(2,3))
for (i in 1:5){
  hist(years_stand[,i],probability  =T,xlim=c(-5,5),main = paste('Histogram of', colnames(years_stand)[
  curve(dnorm(x,mean(years_stand[,i]),sd(years_stand[,i])),add=T,lwd =2,col = 3)
}
```





Dagli istogrammi empirici a cui Ã¨ stata sovrapposta la distribuzione della normale standard, si evidenzia un andamento *normale*.

Per completezza, si verifica le variabili abbiano una media nulla e varianza unitaria.

```
mean_var=function(dataset) {
  matrice=matrix(NA, ncol=3,nrow=dim(dataset)[2])
for (i in 1:dim(dataset)[2]) {
  matrice[i,1]=colnames(dataset)[i]
  matrice[i,2]=round(mean(dataset[,i]),7)
  matrice[i,3]=var(dataset[,i])
}
  colnames(matrice)=c("variabile","mean","variance")
  return(kable(matrice))

}
mean_var(years_stand)
```

variabile	mean	variance
TotalWorkingYears	0	1
YearsAtCompany	0	1
YearsInCurrentRole	0	1
YearsSinceLastPromotion	0	1
YearsWithCurrManager	0	1

A seguito della standardizzazione, quindi, Ã¨ confermata l'ipotesi di normalitÃ delle variabili e si puÃ²

procedere con la LDA.

```
LDA_data= years_stand
LDA_data$Attrition = data$Attrition

linearDA = lda(formula=Attrition ~., data=LDA_data)

linearDA
```

```
## Call:
## lda(Attrition ~ ., data = LDA_data)
##
## Prior probabilities of groups:
##        No       Yes
## 0.8387755 0.1612245
##
## Group means:
##     TotalWorkingYears YearsAtCompany YearsInCurrentRole
## No          0.07497243      0.0589005         0.07036256
## Yes        -0.39004643     -0.3064317        -0.36606347
##     YearsSinceLastPromotion YearsWithCurrManager
## No               0.01447124           0.06845797
## Yes             -0.07528710          -0.35615474
##
## Coefficients of linear discriminants:
##                                 LD1
## TotalWorkingYears       -0.7047775
## YearsAtCompany           0.3499226
## YearsInCurrentRole      -0.6028229
## YearsSinceLastPromotion  0.4945390
## YearsWithCurrManager    -0.4832899
```

LDA evaluation

```
predict_LDA <- predict(linearDA, LDA_data)$class #LDA predicted Attrition

probability_LDA <- predict(linearDA, LDA_data)$posterior #LDA predicted Class probability

all.equal(predict_LDA,LDA_data$Attrition)
```

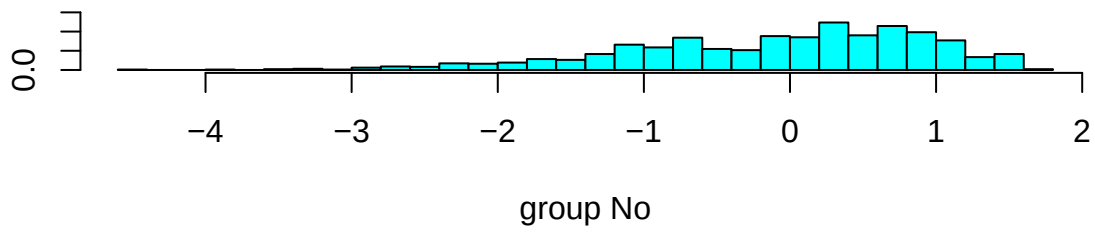
```
## [1] "237 string mismatches"
```

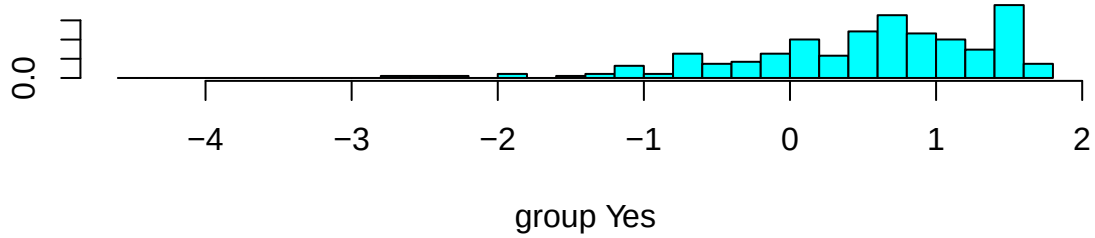
```
confusionMatrix(predict_LDA, LDA_data$Attrition, positive = 'Yes')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1233  237
##        Yes    0    0
##
##                Accuracy : 0.8388
##                  95% CI : (0.819, 0.8572)
##     No Information Rate : 0.8388
##     P-Value [Acc > NIR] : 0.5173
##
##                   Kappa : 0
```

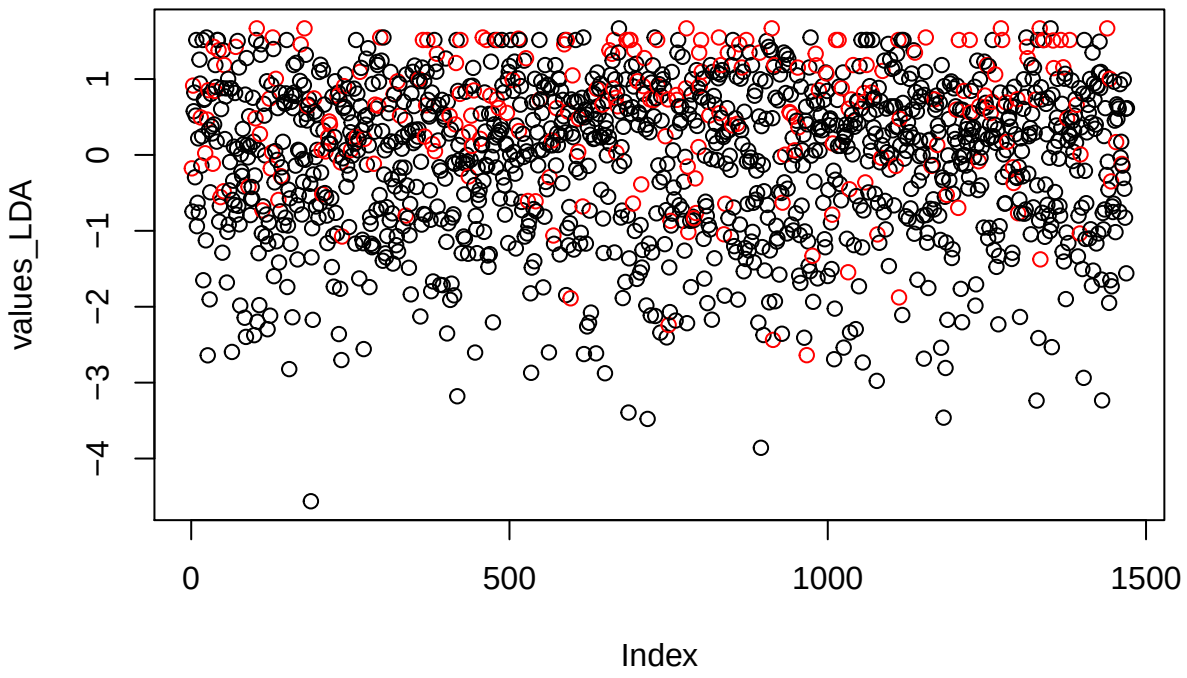
```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.8388
##              Prevalence : 0.1612
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

```
values_LDA <- predict(linearDA, LDA_data)$x
ldahist(data=values_LDA, g= LDA_data$Attrition)
```






```
plot(values_LDA, col=LDA_data$Attrition)
```



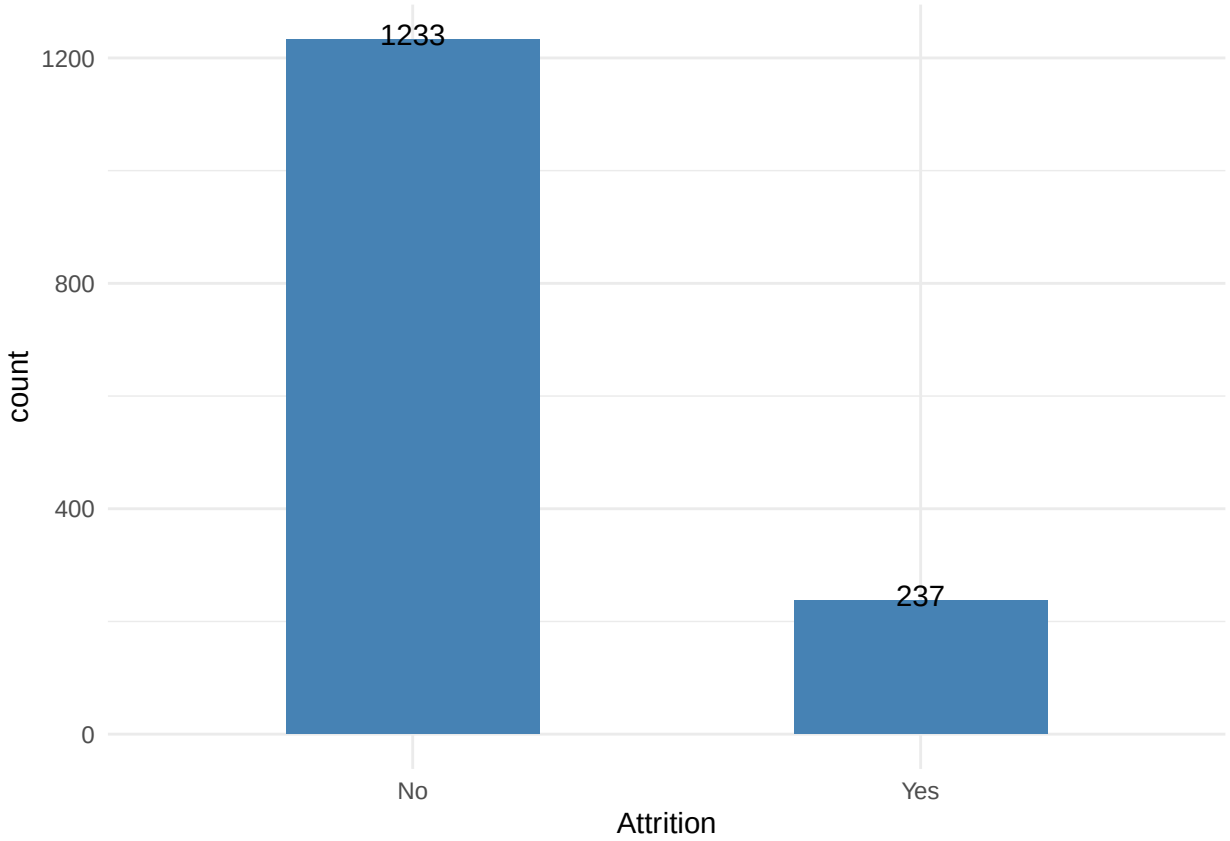
Si puÃ² vedere che, tramite la LDA, vengono previsti erroneamente 237 osservazioni che corrispondono alla categoria Attrition=Yes, ovvero la categoria di interesse. Come si evince sia degli istogrammi che dallo scatterplot, la LDA non discrimina opportunamente le due classi. Questo potrebbe essere causato dal problema delle classi sbilanciate.

Come si nota dal seguente istogramma, infatti, le osservazioni non sono equamente distribuite tra le due classi di *Attrition*.

```
unbalanced_class=ggplot(data,aes(x=Attrition))+
  geom_bar(width=0.5,fill="steelblue")+
  stat_count(binwidth=1, geom="text", aes(label=..count..), vjust=0.25) +
  theme_minimal()
```

```
## Warning: Ignoring unknown parameters: binwidth
```

```
options(repr.plot.width=8,repr.plot.height=3)
unbalanced_class
```



Si prova, quindi, ad effettuare la LDA a seguito di un oversampling.

```
LDA_data_over<- ovun.sample(Attrition ~ ., data = LDA_data, method = "over", N=2466)$data
table(LDA_data_over$Attrition)
```

```
##
##   No  Yes
## 1233 1233
```

```
linearDA_over = lda(formula=Attrition ~., data=LDA_data_over)
predict_LDA_over <- predict(linearDA_over, LDA_data_over)$class
probability_LDA_over <- predict(linearDA_over, LDA_data_over)$posterior
all.equal(predict_LDA_over,LDA_data_over$Attrition)
```

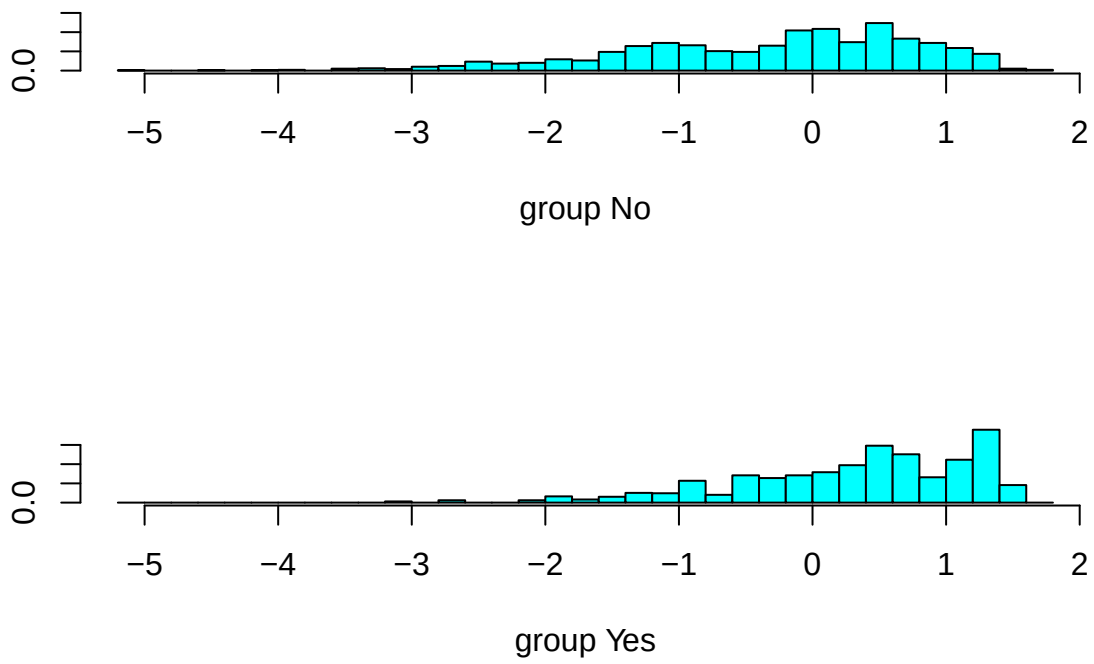
```
## [1] "945 string mismatches"
```

```
confusionMatrix(predict_LDA_over, LDA_data_over$Attrition, positive = "Yes")
```

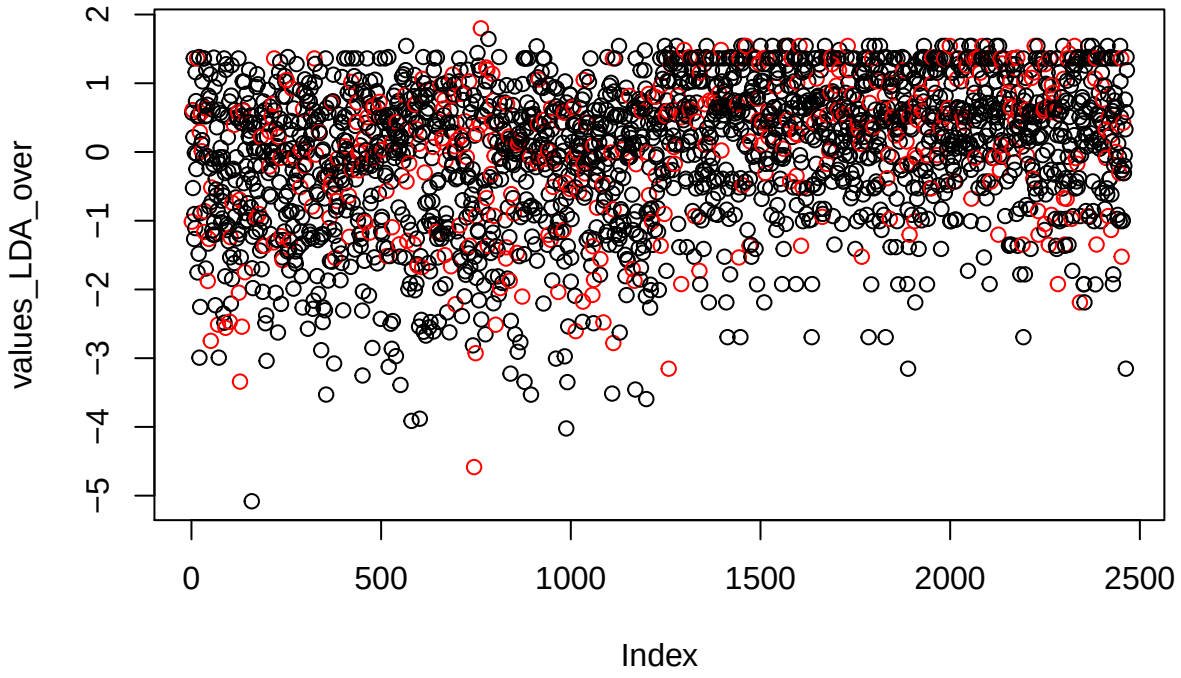
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  670 382
##        Yes 563 851
##
##                Accuracy : 0.6168
##                  95% CI : (0.5973, 0.636)
##     No Information Rate : 0.5
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2336
##
##  Mcnemar's Test P-Value : 4.759e-09
##
##             Sensitivity : 0.6902
##             Specificity : 0.5434
##          Pos Pred Value : 0.6018
##          Neg Pred Value : 0.6369
##              Prevalence : 0.5000
##          Detection Rate : 0.3451
##    Detection Prevalence : 0.5734
##       Balanced Accuracy : 0.6168
##
##        'Positive' Class : Yes
##
```

```
values_LDA_over <- predict(linearDA_over, LDA_data_over)$x
ldahist(data=values_LDA_over, g= LDA_data_over$Attrition)
```




```
plot(values_LDA_over, col=LDA_data$Attrition)
```



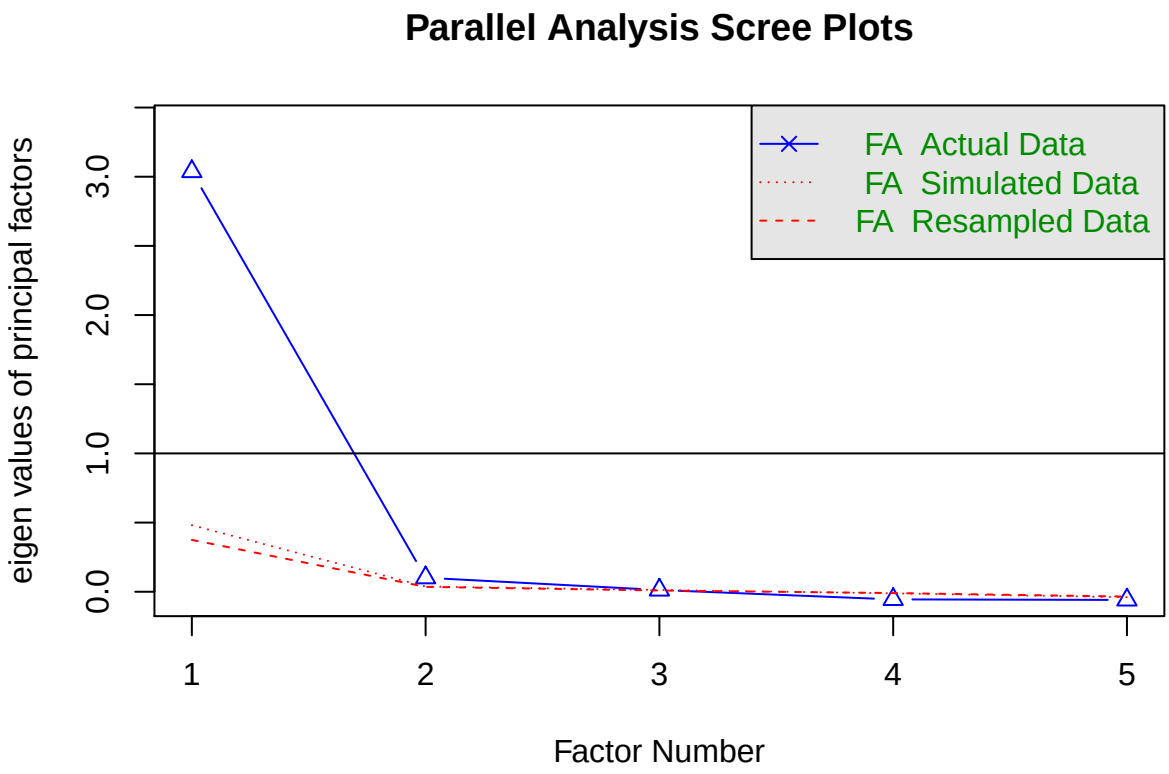
Si evince che, anche effettuando un oversampling, i risultati non sono soddisfacenti. Infatti, le osservazioni erroneamente classificate sono 959 su 2466. Anche in questo caso, dall'istogramma e dallo scatterplot, si conclude che la LDA non Ã¨ utile a discrimare le due categorie. Si esclude, perciÃ², questa tecnica per risolvere i problemi di dimensionalitÃ elevata e di multicollinearitÃ .

** EXPLORATORY FACTOR ANALYSIS **

Si Ã¨ scelto di effettuare un'altra tecnica di dimensionality reduction, la *Factor Analysis*, un metodo utilizzato per descrivere la variabilitÃ tra le variabili osservate correlate, tramite un numero potenzialmente inferiore di fattori latenti.

Il grafico seguente viene utilizzato per individuare il numero appropriato di fattori da estrarre. Lo scree plot rappresenta, per ogni numero di fattori, il relativo autovalore, ovvero la percentuale di varianza spiegata.

```
# scree plot
parallel <- fa.parallel(years, fm = 'minres', fa = 'fa')
```




```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  NA
```

La parallel analysis consiglia di estrarre un numero di fattori pari a 2. Come si evince dal grafico, infatti, aggiungere il terzo fattore non sarebbe conveniente in quanto aggiunge una percentuale di varianza spiegata irrilevante.

Sempre dallo scree-plot, Ã¨ possibile notare che c'Ã¨ molta differenza di variabilitÃ spiegata tra il primo e secondo fattore. Si Ã¨ pensato, quindi, di verificare se un solo fattore fosse sufficiente a discriminare le osservazioni.

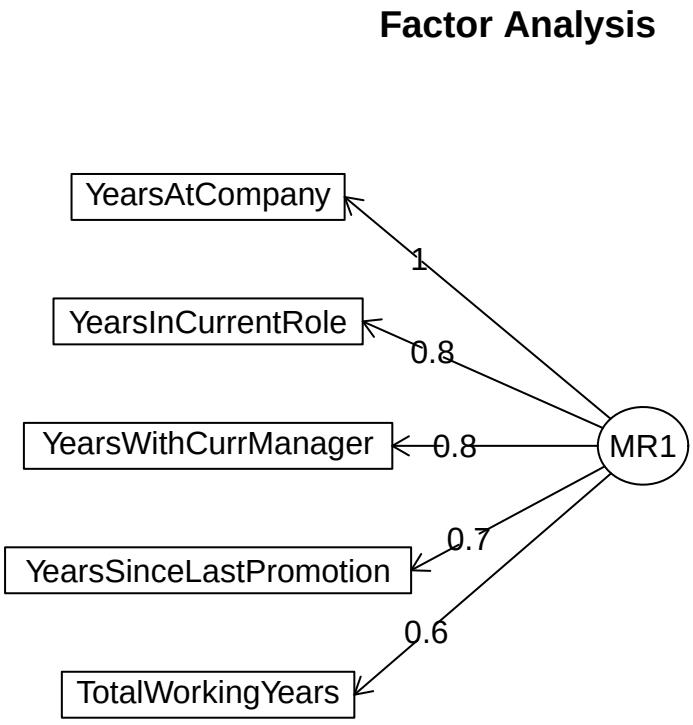
Si procede con lo svolgilmento della Factor Analysis con un solo fattore.

```
onefactor <- fa(years,nfactors = 1,rotate = "promax",fm="minres")

summary(onefactor)
```

```
##
## Factor analysis with Call: fa(r = years, nfactors = 1, rotate = "promax", fm = "minres")
##
## Test of the hypothesis that 1 factor is sufficient.
## The degrees of freedom for the model is 5  and the objective function was  0.08
## The number of observations was  1470  with Chi Square =  114.19  with prob <  5.3e-23
##
## The root mean square of the residuals (RMSA) is  0.03
## The df corrected root mean square of the residuals is  0.04
##
## Tucker Lewis Index of factoring reliability =  0.948
## RMSEA index =  0.122  and the 10 % confidence intervals are  0.103 0.142
```

```
## BIC =  77.73
```

```
fa.diagram(onefactor)
```



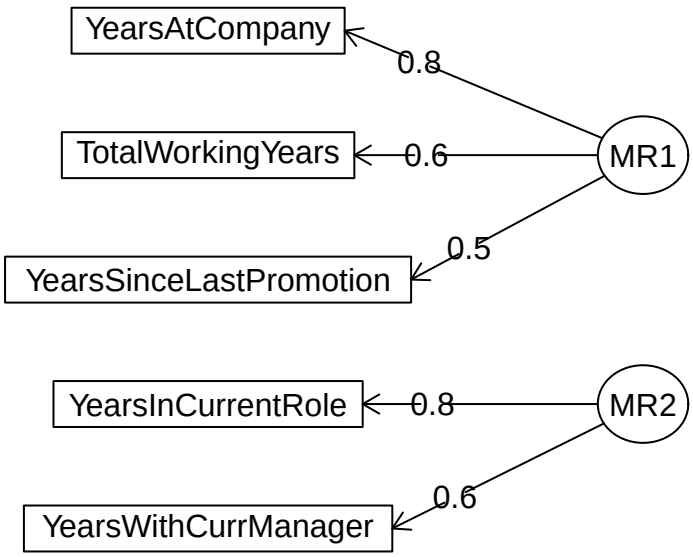
Nel summary si osserva che la FA con solo un fattore ha un pvalue prossimo a 0 che porta a rifiutare l'ipotesi nulla "1 factor is sufficient".

Si svolge, quindi, la FA con due fattori.

```
twofactor <- fa(years,nfactors = 2,rotate = "varimax",fm="minres", scores='regression')
summary(twofactor)
```

```
##
## Factor analysis with Call: fa(r = years, nfactors = 2, rotate = "varimax", scores = "regression",
##     fm = "minres")
##
## Test of the hypothesis that 2 factors are sufficient.
## The degrees of freedom for the model is 1  and the objective function was  0
## The number of observations was  1470  with Chi Square =  6.52  with prob <  0.011
##
## The root mean square of the residuals (RMSA) is  0.01
## The df corrected root mean square of the residuals is  0.02
##
## Tucker Lewis Index of factoring reliability =  0.987
## RMSEA index =  0.061  and the 10 % confidence intervals are  0.024 0.11
## BIC =  -0.78
```

```
fa.diagram(twofactor)
```



Siccome il p-value risulta essere maggiore rispetto all'analisi con un solo fattore, si preferisce "riassumere" le cinque variabili in questione tramite i due fattori.

Si analizzano i loadings delle 5 variabili sui due fattori.

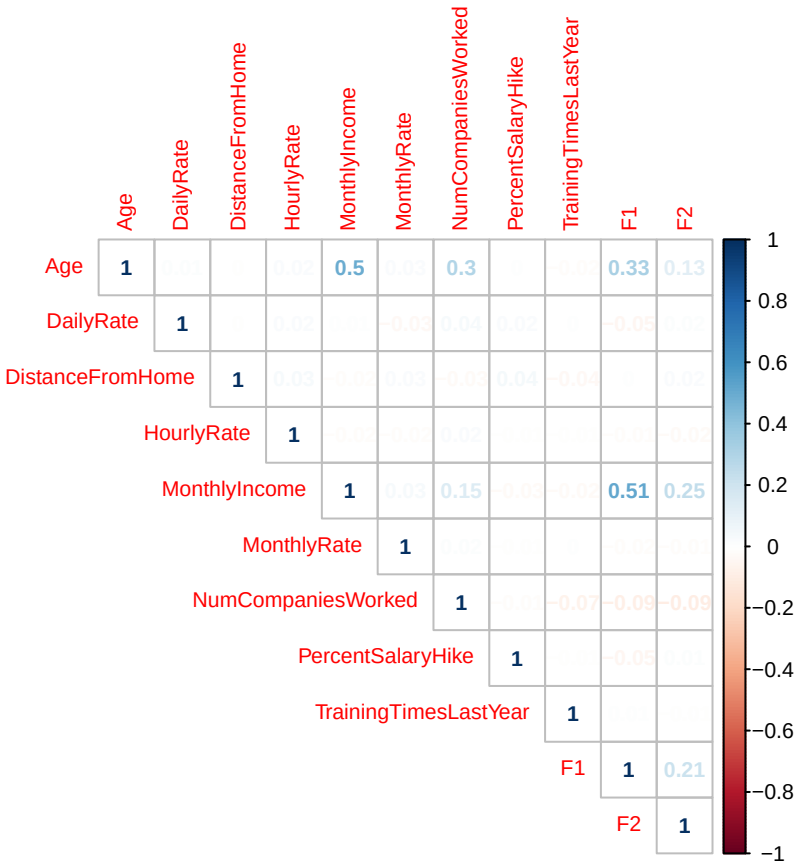
```
print(twofactor$loadings,cutoff = 0.3)
```

```
##
## Loadings:
##                           MR1   MR2
## TotalWorkingYears         0.573
## YearsAtCompany            0.845 0.518
## YearsInCurrentRole        0.393 0.830
## YearsSinceLastPromotion 0.476 0.431
## YearsWithCurrManager      0.522 0.614
##
##                   MR1   MR2
## SS loadings     1.696 1.599
## Proportion Var 0.339 0.320
## Cumulative Var 0.339 0.659
```

In conclusione, con la factor analysis vengono creati due nuovi fattori dati dalla combinazione lineare delle variabili precedenti. In particolare dai loadings si osserva che *YearsAtCompany, TotalWorkingYears e YearSinceLastPromotion* pesano di piÃ¹ sul primo fattore rispetto al secondo e, quindi, F1 potrebbe essere una sitesi dell'**esperienza lavorativa del dipendente**. *YearsInCurrentRole e YearsWithCurrentManager*, invece, pesano di piÃ¹ sul secondo fattore che,invece, potrebbe essere una sintesi degli **esperienza lavorativa nello stesso team** del dipendente.

Aggiungiamo, infine, al dataset le 2 nuove variabili create, eliminando le 5 variabili precedenti.

```
data$F1=twofactor$scores[,1]
data$F2=twofactor$scores[,2]

data <- data[,-which(names(data) %in% c("YearsWithCurrManager","YearsAtCompany","YearsInCurrentRole","Y
dataWithFactor_num<- which(sapply(data,is.numeric))
corrplot(cor(data[dataWithFactor_num]),type = "upper",method='number',tl.cex = .7,cl.cex = .7,number.ce
```



A questo punto abbiamo risolto il problema della multicollinaritÃ , come si nota dal grafico riportato sopra. Vediamo, infatti, che l'unica correlazione significativa Ã¨ di 0.5 tra F1 e MonthlyIncome.

Quindi, a questo punto del preprocessing sono state eliminate le seguenti variabili:

1- EmployCount (perchÃ¨ sempre pari a 1) 2- EmployNumber (perchÃ¨ Ã¨ l'ID del dipendente) 3- Over18 (percÃ¨ sempre pari a uno in quanto tutti ii dipendenti sono maggiorenni) 4- StandardHour (perchÃ¨ sempre pari a 80) 5- Education (perchÃ¨ dal chi-quadro test risultava indipendente alla variabile target) 6- Gender (perchÃ¨ dal chi-quadro test risultava indipendente alla variabile target) 7- PerformanceRating (perchÃ¨ dal chi-quadro test risultava indipendente alla variabile target) 8- RelationshipSatisfaction (perchÃ¨ dal chi-quadro test risultava indipendente alla variabile target) 9- YearsWithCurrManager (perchÃ¨ riassunte nei 2 Factor) 10- YearsAtCompany (perchÃ¨ riassunte nei 2 Factor) 11- YearsInCurrentRole (perchÃ¨ riassunte nei 2 Factor) 12- YearsSinceLastPromotion (perchÃ¨ riassunte nei 2 Factor) 13- TotalWorkingYears (perchÃ¨ riassunte nei 2 Factor)

Sono state aggiunti i due fattori: 1- F1 2- F2

Da uno stato iniziale di 35 variabili si Ã¨ ottenuto un dataset contenente 24 variabili.

Aggiungiamo un'analisi dell'importana delle variabili perchÃ¨ ci interessa indagare l'importanza in particolare

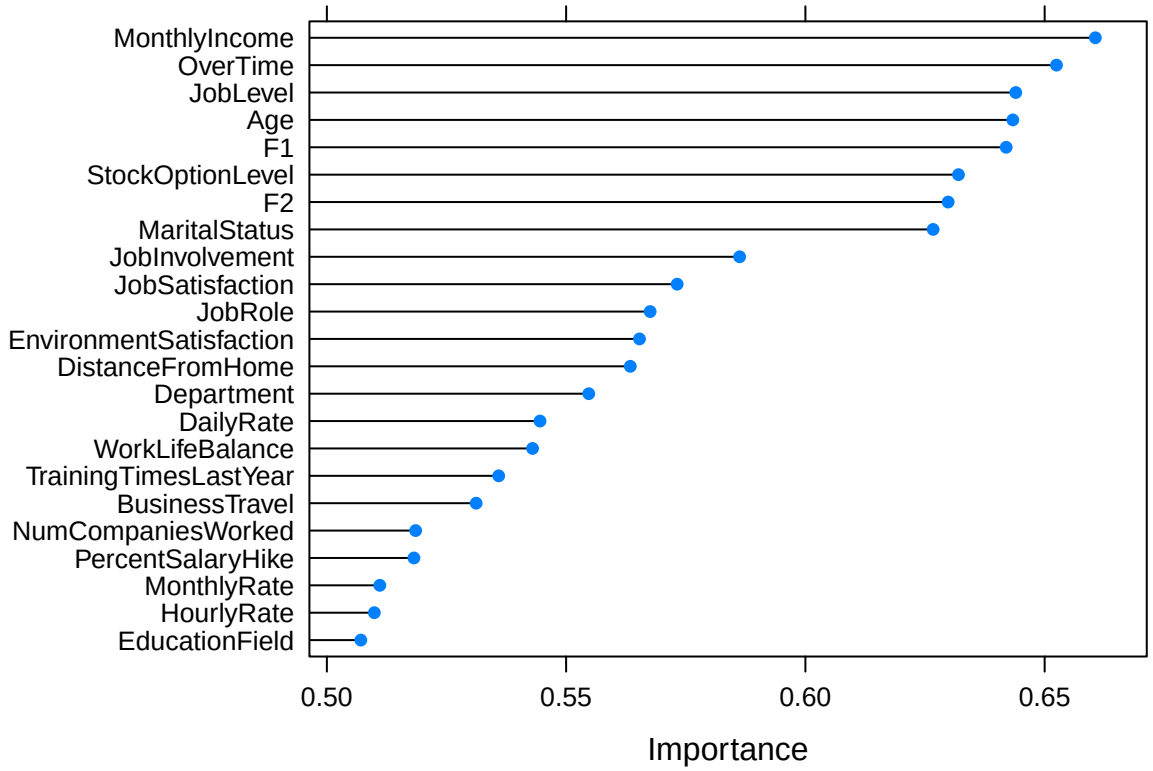
di HourlyRate, DailyRate e MonthlyRate perchÃ¨ sono di difficile interpretazione.

Rimangono comunque alcune variabili di dubbia interpretazione (HourlyRate, DailyRate e MonthlyRate) e, per valutare la loro importanza rispetto ad Attrition, viene rappresentato un ranking delle variabili basato sul modello Learning Vector Quantization (LVQ).

```
data_over<- ovun.sample(Attrition ~ ., data = data, method = "over", N=2466)$data

set.seed(7)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(Attrition~., data=data_over, method="lvq", preProcess="scale", trControl=control)
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 23)
##
##                          Importance
## MonthlyIncome                0.6606
## OverTime                     0.6525
## JobLevel                     0.6440
## Age                          0.6434
## F1                           0.6420
## StockOptionLevel             0.6320
## F2                           0.6298
## MaritalStatus                0.6267
## JobInvolvement               0.5863
## JobSatisfaction              0.5732
## JobRole                      0.5676
## EnvironmentSatisfaction      0.5653
## DistanceFromHome             0.5634
## Department                   0.5547
## DailyRate                    0.5445
## WorkLifeBalance              0.5430
## TrainingTimesLastYear        0.5359
## BusinessTravel               0.5312
## NumCompaniesWorked           0.5186
## PercentSalaryHike            0.5182
```

```
# plot importance
plot(importance)
```



Come si si puÃ² vedere dal grafico le variabili *HourlyRate, DailyRate e MonthlyRate* non risultano di particolare importanza quindi, dal momento il significato non Ã¨ chiaro, si preferisce eliminarle dall'analisi.

```
data <- data[,-which(names(data) %in% c("MonthlyRate","DailyRate","HourlyRate"))]
dim(data)
```

```
## [1] 1470   21
```

Data Modeling

Terminata il pre-processing, si passa alla fase di **Data Modeling**.

Si Ã¨ scelto di sviluppare i seguenti modelli:

- *Decision Tree*;
- *Random Forest*;
- *Stochastic Gradient Boosting*;
- *Neural Network*;
- *Support Vector Machine*;
- *Logistic Regression*.

Per ottenere una validazione dei classificatori utilizzati Ã¨ stato utilizzato un approccio basato sulla *cross validation*, il cosiddetto **K-Folds Cross Validation**. Questa tecnica statistica suddivide il dataset in k-partizioni di eguale numerositÃ e assicura che tutti i record vengano utilizzati almeno una volta sia nel *training set* che nel *test set*. Il numero di folds utilizzato Ã¨ pari a k=10. Nei casi in cui non Ã¨ risultato computazionalmente troppo pesante, Ã¨ stata utilizzata una *repeated k-folds cross validation*.

I parametri caratteristici di ogni modello, non sono stati scelti a priori ma Ã¨ stato utilizzato un sistema di tuning al fine di utilizzare il parametro migliore.

Infine, sia per tunare che per confrontare i modelli, Ã¨ stata la ROC.

I modelli vengono appresi sul train set, pari al 67% del dataset iniziale e su cui viene effettuato l'oversampling per risolvere il problema delle classi sbilanciate.

Essendo in presenza di classi sbilanciate, la misura piÃ¹ appropriata per confrontare i modello non Ã¨ l'*accurancy* ma bensÃ¬ la *precision* e la *recall*, riassunte dalla **F1-Measure** (media armonica delle due precedenti metriche).

```
#train e test set
smp_size <- floor(0.67 * nrow(data))

set.seed(1)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)

train <- data[train_ind, ]
test <- data[-train_ind, ]

#oversampling train set
train<- ovun.sample(Attrition ~ ., data = train, method = "over", N=1660)$data
```

Decision Tree

Il decision tree, in particolare, viene anche utilizzato per selezionare un numero inferiore di variabili con cui stimare i modelli *Logistic Regression, Neural Network e Support Vector Machine*. Tutti gli altri modelli, invece, eseguono un'eliminazione recursiva delle variabili automaticamente e non sono penalizzati dall'inserimento di tutte le variabili.

```
set.seed(123)
metric <- "ROC"
Ctrl <- trainControl(method = "cv" , number=10, classProbs = TRUE,
summaryFunction = twoClassSummary)
rpartTune <- train(Attrition ~ ., data = train, method = "rpart",tuneLength = 15, trControl = Ctrl, meti
rpartTune
```

```
## CART
##
## 1660 samples
##   20 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1494, 1494, 1494, 1495, 1493, 1494, ...
## Resampling results across tuning parameters:
##
##   cp           ROC        Sens       Spec
##   0.006053269  0.8409983  0.7492947  0.8657344
##   0.006658596  0.8411526  0.7468851  0.8657344
##   0.007263923  0.8311051  0.7372025  0.8548910
##   0.007869249  0.8269247  0.7287687  0.8537149
##   0.008474576  0.8199200  0.7251543  0.8417671
##   0.010290557  0.8031067  0.7033353  0.8165806
##   0.010895884  0.7878979  0.6803115  0.8129805
##   0.012106538  0.7858431  0.6875404  0.7997418
##   0.015738499  0.7629050  0.6524978  0.7997275
##   0.016949153  0.7626220  0.6500735  0.7997275
```

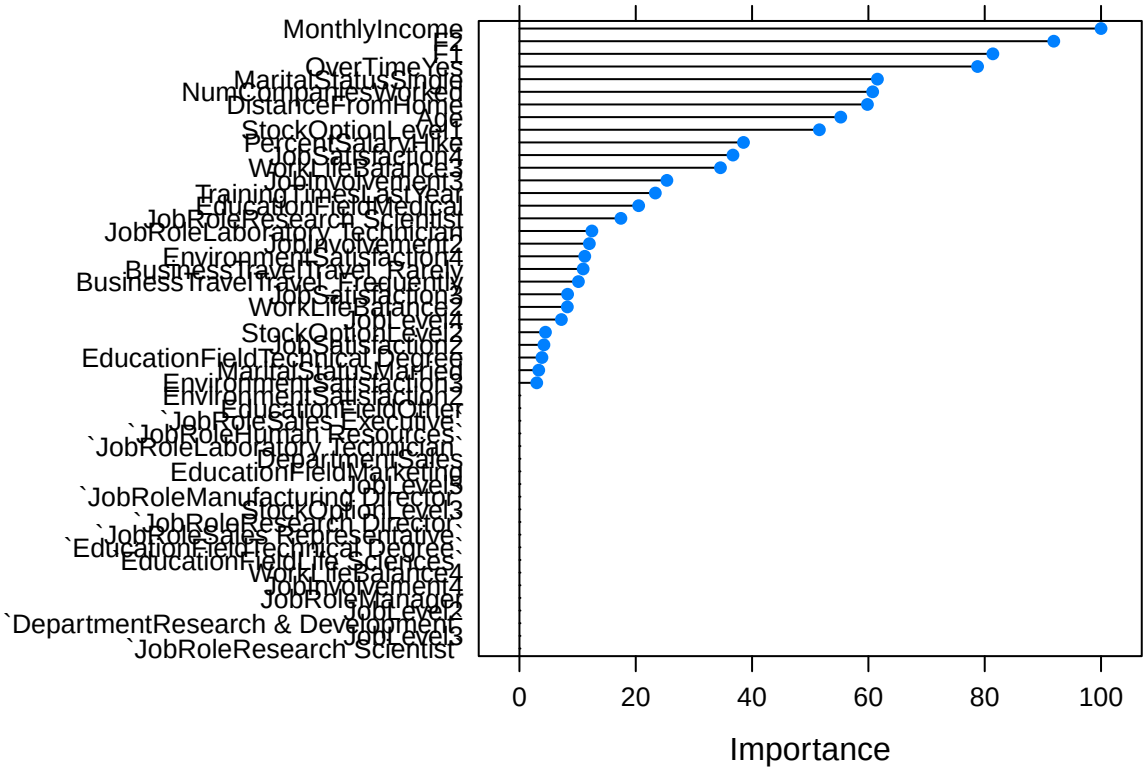
```
##    0.018765133  0.7416238  0.6368205  0.7891279
##    0.025423729  0.7217793  0.6271966  0.7697361
##    0.029055690  0.7013568  0.6876579  0.6618761
##    0.043583535  0.6769120  0.7334411  0.5811532
##    0.289346247  0.5464071  0.3120482  0.7807659
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.006658596.
```

Il tuning del CP indica come parametro migliore cp = 0.003614458.

```
pander(getTrainPerf(rpartTune))
```

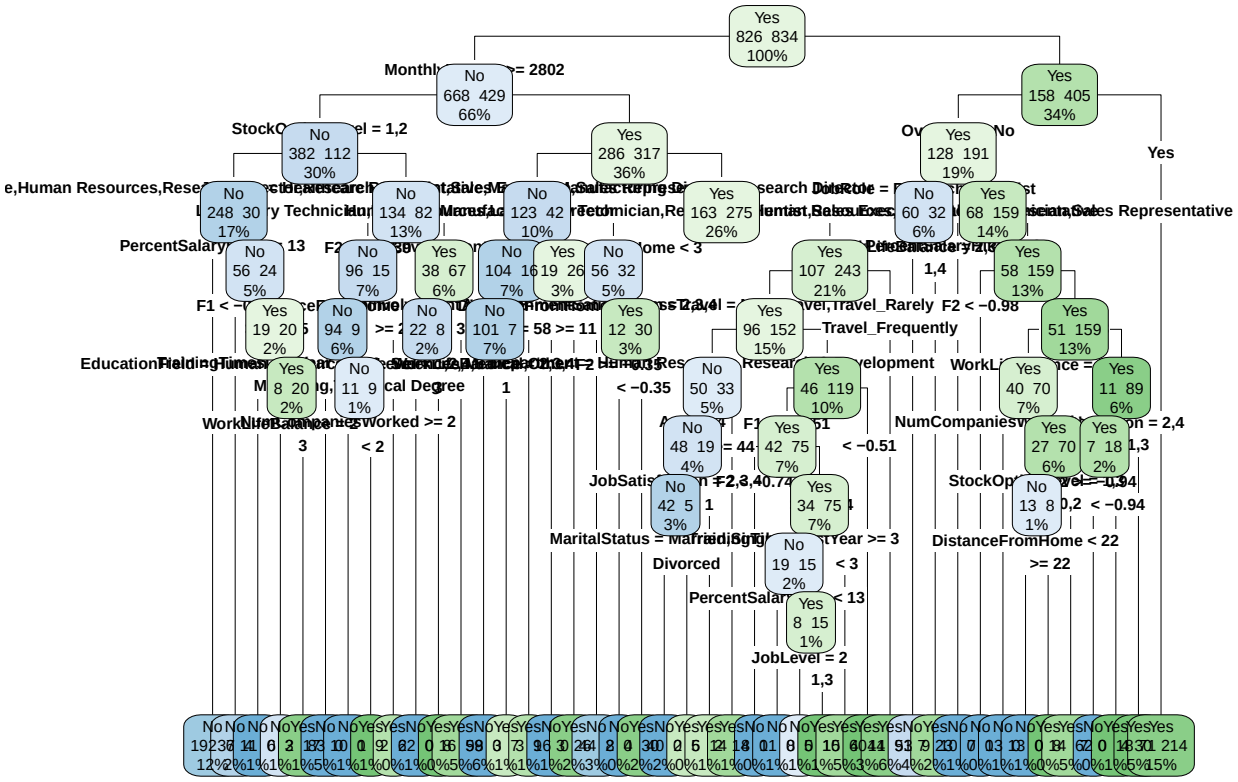
TrainROC	TrainSens	TrainSpec	method
0.8412	0.7469	0.8657	rpart

```
Vimportance <- varImp(rpartTune)
plot(Vimportance)
```



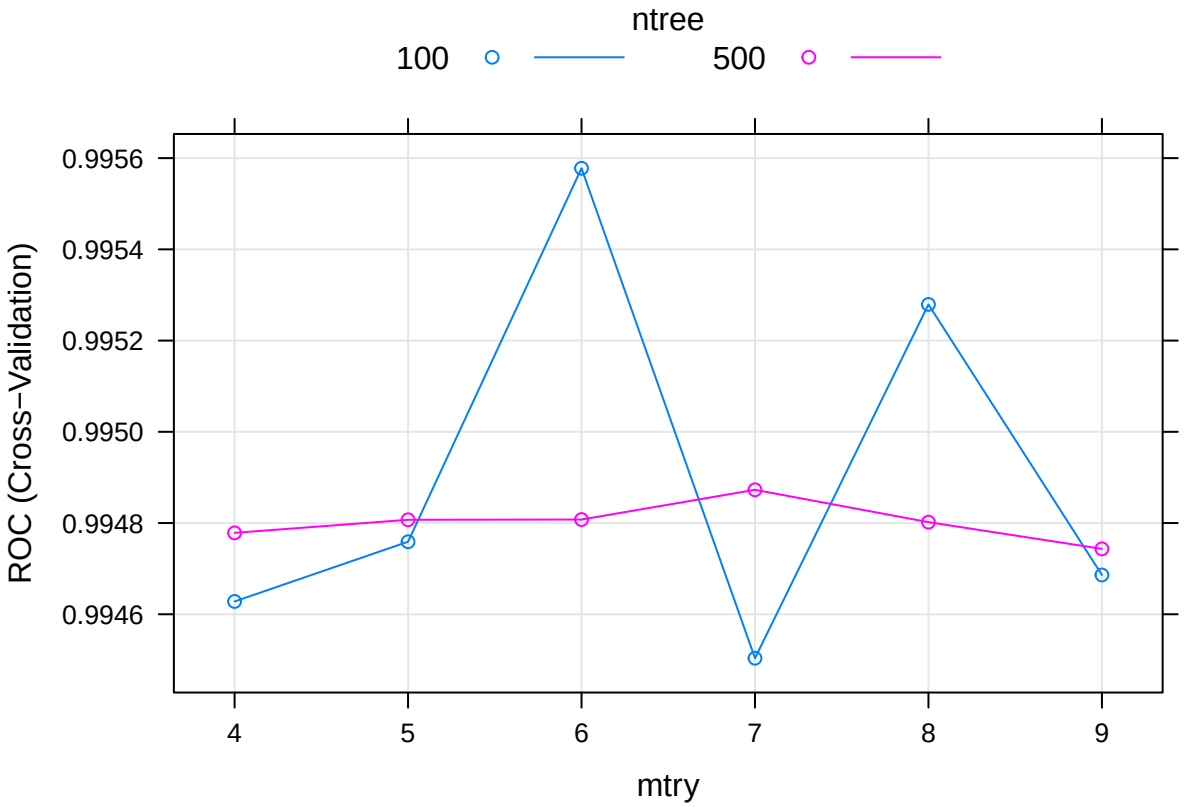

```
set.seed(123)
Ctrl_save <- trainControl(method = "cv" , number=10, summaryFunction = twoClassSummary,
classProbs = TRUE, savePredictions = TRUE)
rpartTuneMy <- train(Attrition ~ ., data = train, method = "rpart",
tuneGrid=data.frame(cp=0.003614458),
trControl = Ctrl_save, metric=metric)
```

```
set.seed(123)
mytree <- rpart(Attrition ~ ., data = train, method = "class", cp = 0.003614458)
rpart.plot(mytree, type = 4, extra = 101, cex = 0.5)
```



Random Forest

```
customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"),
class = rep("numeric", 2),
label = c("mtry", "ntree"))
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes
set.seed(123)
tunegrid <- expand.grid(.mtry=c(4:9), .ntree=c(100,500))
rpartTuneMyRf <- train(Attrition ~ ., data = train, method = customRF,
tuneGrid=tunegrid, trControl = Ctrl, metric=metric)
plot(rpartTuneMyRf)
```



Dal grafico si evince che il modello migliore Ã¨ con 4 foreste e 500 alberi.

```
set.seed(123)
tunegrid <- expand.grid(.mtry=4, .ntree=500)
rpartTuneMyRf_ok <- train(Attrition ~ ., data = train, method = customRF,
tuneGrid=tunegrid, trControl = Ctrl_save, metric=metric)
```

Naive-Bayes

```
set.seed(123)
grid <- expand.grid(fL=0, usekernel = TRUE, adjust=1)
NBfit <- train(Attrition ~ ., data = train, method="nb", tuneGrid=grid,
trControl=Ctrl_save, metric=metric)
pander(getTrainPerf(NBfit))
```

TrainROC	TrainSens	TrainSpec	method
0.8494	0.2857	0.976	nb

Stochastic Gradient Boosting

```
## Loaded gbm 2.1.5
```

```
set.seed(123)
grid <- expand.grid(n.trees=150, interaction.depth=3, shrinkage=0.1, n.minobsinnode=10)
STGfit.one.shot <- train(Attrition ~ ., data = train, method="gbm", tuneGrid=grid, trControl=Ctrl_save,
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.3586             nan     0.1000    0.0139
##      2        1.3293             nan     0.1000    0.0130
##      3        1.3072             nan     0.1000    0.0100
##      4        1.2860             nan     0.1000    0.0113
##      5        1.2654             nan     0.1000    0.0094
##      6        1.2469             nan     0.1000    0.0076
##      7        1.2300             nan     0.1000    0.0067
##      8        1.2150             nan     0.1000    0.0051
##      9        1.2006             nan     0.1000    0.0057
##     10        1.1888             nan     0.1000    0.0027
##     20        1.0621             nan     0.1000    0.0046
##     40        0.9059             nan     0.1000    0.0012
##     60        0.8054             nan     0.1000    0.0008
##     80        0.7311             nan     0.1000    0.0005
##    100        0.6678             nan     0.1000    0.0002
##    120        0.6112             nan     0.1000    0.0005
##    140        0.5624             nan     0.1000   -0.0000
##    150        0.5392             nan     0.1000    0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3572             nan     0.1000    0.0140
##      2        1.3301             nan     0.1000    0.0129
##      3        1.3060             nan     0.1000    0.0123
##      4        1.2824             nan     0.1000    0.0078
##      5        1.2614             nan     0.1000    0.0066
##      6        1.2379             nan     0.1000    0.0093
##      7        1.2212             nan     0.1000    0.0060
##      8        1.2049             nan     0.1000    0.0058
##      9        1.1870             nan     0.1000    0.0070
##     10        1.1720             nan     0.1000    0.0078
##     20        1.0642             nan     0.1000    0.0024
##     40        0.9082             nan     0.1000    0.0018
##     60        0.8031             nan     0.1000    0.0008
##     80        0.7275             nan     0.1000    0.0013
##    100        0.6623             nan     0.1000    0.0008
##    120        0.6020             nan     0.1000    0.0001
##    140        0.5547             nan     0.1000   -0.0005
##    150        0.5335             nan     0.1000    0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3516             nan     0.1000    0.0160
##      2        1.3227             nan     0.1000    0.0130
##      3        1.2980             nan     0.1000    0.0099
##      4        1.2793             nan     0.1000    0.0066
##      5        1.2601             nan     0.1000    0.0076
##      6        1.2417             nan     0.1000    0.0085
##      7        1.2239             nan     0.1000    0.0088
##      8        1.2066             nan     0.1000    0.0062
##      9        1.1884             nan     0.1000    0.0081
##     10        1.1718             nan     0.1000    0.0075
##     20        1.0553             nan     0.1000    0.0034
##     40        0.9027             nan     0.1000    0.0019
##     60        0.7905             nan     0.1000    0.0005
##     80        0.7158             nan     0.1000    0.0009
```

```
##    100        0.6519             nan     0.1000    0.0008
##    120        0.5978             nan     0.1000   -0.0000
##    140        0.5515             nan     0.1000    0.0001
##    150        0.5325             nan     0.1000    0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3527             nan     0.1000    0.0146
##      2        1.3224             nan     0.1000    0.0123
##      3        1.2993             nan     0.1000    0.0108
##      4        1.2751             nan     0.1000    0.0091
##      5        1.2566             nan     0.1000    0.0076
##      6        1.2409             nan     0.1000    0.0068
##      7        1.2226             nan     0.1000    0.0062
##      8        1.2070             nan     0.1000    0.0062
##      9        1.1921             nan     0.1000    0.0057
##     10        1.1776             nan     0.1000    0.0060
##     20        1.0561             nan     0.1000    0.0030
##     40        0.9061             nan     0.1000    0.0023
##     60        0.8100             nan     0.1000    0.0009
##     80        0.7296             nan     0.1000    0.0004
##    100        0.6667             nan     0.1000    0.0005
##    120        0.6155             nan     0.1000    0.0003
##    140        0.5687             nan     0.1000   -0.0000
##    150        0.5487             nan     0.1000   -0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3542             nan     0.1000    0.0159
##      2        1.3259             nan     0.1000    0.0128
##      3        1.2991             nan     0.1000    0.0127
##      4        1.2777             nan     0.1000    0.0090
##      5        1.2555             nan     0.1000    0.0081
##      6        1.2371             nan     0.1000    0.0072
##      7        1.2223             nan     0.1000    0.0048
##      8        1.2062             nan     0.1000    0.0069
##      9        1.1875             nan     0.1000    0.0073
##     10        1.1741             nan     0.1000    0.0052
##     20        1.0595             nan     0.1000    0.0018
##     40        0.9080             nan     0.1000    0.0021
##     60        0.8086             nan     0.1000    0.0002
##     80        0.7267             nan     0.1000    0.0012
##    100        0.6643             nan     0.1000    0.0006
##    120        0.6145             nan     0.1000    0.0010
##    140        0.5701             nan     0.1000   -0.0002
##    150        0.5464             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3564             nan     0.1000    0.0122
##      2        1.3280             nan     0.1000    0.0136
##      3        1.3055             nan     0.1000    0.0102
##      4        1.2802             nan     0.1000    0.0085
##      5        1.2610             nan     0.1000    0.0080
##      6        1.2417             nan     0.1000    0.0089
##      7        1.2206             nan     0.1000    0.0080
##      8        1.2060             nan     0.1000    0.0060
```

```
##      9        1.1882             nan     0.1000    0.0072
##     10        1.1773             nan     0.1000    0.0032
##     20        1.0653             nan     0.1000    0.0016
##     40        0.9126             nan     0.1000    0.0020
##     60        0.8102             nan     0.1000    0.0014
##     80        0.7357             nan     0.1000    0.0004
##    100        0.6742             nan     0.1000   -0.0001
##    120        0.6174             nan     0.1000    0.0004
##    140        0.5719             nan     0.1000   -0.0002
##    150        0.5472             nan     0.1000    0.0003
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3533             nan     0.1000    0.0154
##      2        1.3247             nan     0.1000    0.0109
##      3        1.3001             nan     0.1000    0.0103
##      4        1.2732             nan     0.1000    0.0113
##      5        1.2498             nan     0.1000    0.0093
##      6        1.2303             nan     0.1000    0.0081
##      7        1.2134             nan     0.1000    0.0055
##      8        1.1938             nan     0.1000    0.0071
##      9        1.1772             nan     0.1000    0.0074
##     10        1.1611             nan     0.1000    0.0068
##     20        1.0416             nan     0.1000    0.0033
##     40        0.8893             nan     0.1000    0.0010
##     60        0.7916             nan     0.1000    0.0015
##     80        0.7113             nan     0.1000    0.0002
##    100        0.6458             nan     0.1000    0.0009
##    120        0.5941             nan     0.1000    0.0011
##    140        0.5472             nan     0.1000    0.0000
##    150        0.5270             nan     0.1000   -0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3549             nan     0.1000    0.0140
##      2        1.3269             nan     0.1000    0.0121
##      3        1.3049             nan     0.1000    0.0104
##      4        1.2829             nan     0.1000    0.0094
##      5        1.2631             nan     0.1000    0.0079
##      6        1.2434             nan     0.1000    0.0071
##      7        1.2243             nan     0.1000    0.0071
##      8        1.2112             nan     0.1000    0.0045
##      9        1.1963             nan     0.1000    0.0067
##     10        1.1793             nan     0.1000    0.0076
##     20        1.0625             nan     0.1000    0.0025
##     40        0.9108             nan     0.1000    0.0030
##     60        0.8109             nan     0.1000    0.0018
##     80        0.7290             nan     0.1000    0.0007
##    100        0.6688             nan     0.1000    0.0008
##    120        0.6129             nan     0.1000    0.0008
##    140        0.5659             nan     0.1000    0.0002
##    150        0.5464             nan     0.1000   -0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3542             nan     0.1000    0.0123
##      2        1.3303             nan     0.1000    0.0105
```

```
##      3        1.3041             nan     0.1000    0.0105
##      4        1.2806             nan     0.1000    0.0099
##      5        1.2589             nan     0.1000    0.0095
##      6        1.2382             nan     0.1000    0.0096
##      7        1.2177             nan     0.1000    0.0089
##      8        1.2003             nan     0.1000    0.0068
##      9        1.1881             nan     0.1000    0.0041
##     10        1.1741             nan     0.1000    0.0048
##     20        1.0577             nan     0.1000    0.0040
##     40        0.9099             nan     0.1000    0.0019
##     60        0.8046             nan     0.1000    0.0005
##     80        0.7268             nan     0.1000    0.0012
##    100        0.6652             nan     0.1000    0.0004
##    120        0.6121             nan     0.1000    0.0007
##    140        0.5663             nan     0.1000   -0.0001
##    150        0.5461             nan     0.1000    0.0003
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3554             nan     0.1000    0.0122
##      2        1.3290             nan     0.1000    0.0140
##      3        1.3065             nan     0.1000    0.0104
##      4        1.2840             nan     0.1000    0.0099
##      5        1.2634             nan     0.1000    0.0085
##      6        1.2393             nan     0.1000    0.0088
##      7        1.2209             nan     0.1000    0.0065
##      8        1.2012             nan     0.1000    0.0080
##      9        1.1845             nan     0.1000    0.0078
##     10        1.1689             nan     0.1000    0.0061
##     20        1.0538             nan     0.1000    0.0037
##     40        0.9020             nan     0.1000    0.0013
##     60        0.8008             nan     0.1000    0.0012
##     80        0.7250             nan     0.1000    0.0002
##    100        0.6605             nan     0.1000    0.0001
##    120        0.6099             nan     0.1000    0.0006
##    140        0.5634             nan     0.1000    0.0003
##    150        0.5447             nan     0.1000    0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3537             nan     0.1000    0.0153
##      2        1.3255             nan     0.1000    0.0110
##      3        1.2991             nan     0.1000    0.0117
##      4        1.2769             nan     0.1000    0.0104
##      5        1.2544             nan     0.1000    0.0096
##      6        1.2348             nan     0.1000    0.0077
##      7        1.2196             nan     0.1000    0.0055
##      8        1.2028             nan     0.1000    0.0061
##      9        1.1856             nan     0.1000    0.0073
##     10        1.1703             nan     0.1000    0.0059
##     20        1.0542             nan     0.1000    0.0035
##     40        0.9079             nan     0.1000    0.0026
##     60        0.8029             nan     0.1000    0.0006
##     80        0.7267             nan     0.1000    0.0010
##    100        0.6658             nan     0.1000    0.0003
##    120        0.6143             nan     0.1000    0.0010
```

```
##     140         0.5659             nan     0.1000    0.0013
##     150         0.5440             nan     0.1000    0.0003
```

Vengono selezionate le 4 variabili che risultano piÃ¹ importanti dall'analisi del decision tree: - Age - Monthly Income - F1 - F2

```
TRAINSELECT2 <- train[, c(1,2,13,20,21)]
pander(summary(TRAINSELECT2))
```

Age	Attrition	MonthlyIncome	F1	F2
Min. :18.00	No :826	Min. : 1009	Min. :-0.80298	Min. :-2.0681
1st Qu.:29.00	Yes:834	1st Qu.: 2585	1st Qu.:-0.57358	1st Qu.:-0.7086
Median :34.00	NA	Median : 4267	Median :-0.35288	Median :-0.3889
Mean :35.63	NA	Mean : 5615	Mean :-0.07544	Mean :-0.1128
3rd Qu.:42.00	NA	3rd Qu.: 6656	3rd Qu.: 0.01569	3rd Qu.: 0.5478
Max. :60.00	NA	Max. :19943	Max. : 6.41055	Max. : 3.1744

Neural Network

Per tunare la NNET viene effettuato il pre-processing tramite PCA, nromalizzazione e standardizzazione per poi scegliere il migliore.

```
#PCA
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR1 <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
                            method = "nnet",
                            preProcess = 'pca',
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 100)
pander(getTrainPerf(nnetFit_defgridDR1))
```

TrainROC	TrainSens	TrainSpec	method
0.7429	0.6803	0.7086	nnet

```
pander(nnetFit_defgridDR1$bestTune)
```

	size	decay
20	5	3e-04

```
#normalizzazione
set.seed(123)
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR3 <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
method = "nnet",
preProcess = c('range'),
metric=metric,
trControl=Ctrl, tuneGrid=tunegrid,
trace = FALSE,
maxit = 100)
```

```
pander(getTrainPerf(nnetFit_defgridDR3))
```

TrainROC	TrainSens	TrainSpec	method
0.7452	0.6549	0.7109	nnet

```
pander(nnetFit_defgridDR3$bestTune)
```

	size	decay
18	5	1e-04

```
#standardizzazione
set.seed(123)
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR2 <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
method = "nnet",
preProcess = c('center', "scale"),
metric=metric,
trControl=Ctrl, tuneGrid=tunegrid,
trace = FALSE,
maxit = 100)
pander(getTrainPerf(nnetFit_defgridDR2))
```

TrainROC	TrainSens	TrainSpec	method
0.7369	0.6223	0.7325	nnet

```
pander(nnetFit_defgridDR2$bestTune)
```

	size	decay
17	5	1e-05

Si nota che il pre-process migliore Ã¨ la standardizzazione.

```
set.seed(123)
tunegrid <- expand.grid(size=5, decay =0.00001)
nnetFit_finale <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
method = "nnet",
preProcess = c( 'center' ,'scale'),
metric=metric,
trControl=Ctrl_save, tuneGrid=tunegrid,
trace = FALSE,
maxit = 100)
```

Support Vector Machine

```
set.seed(123)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

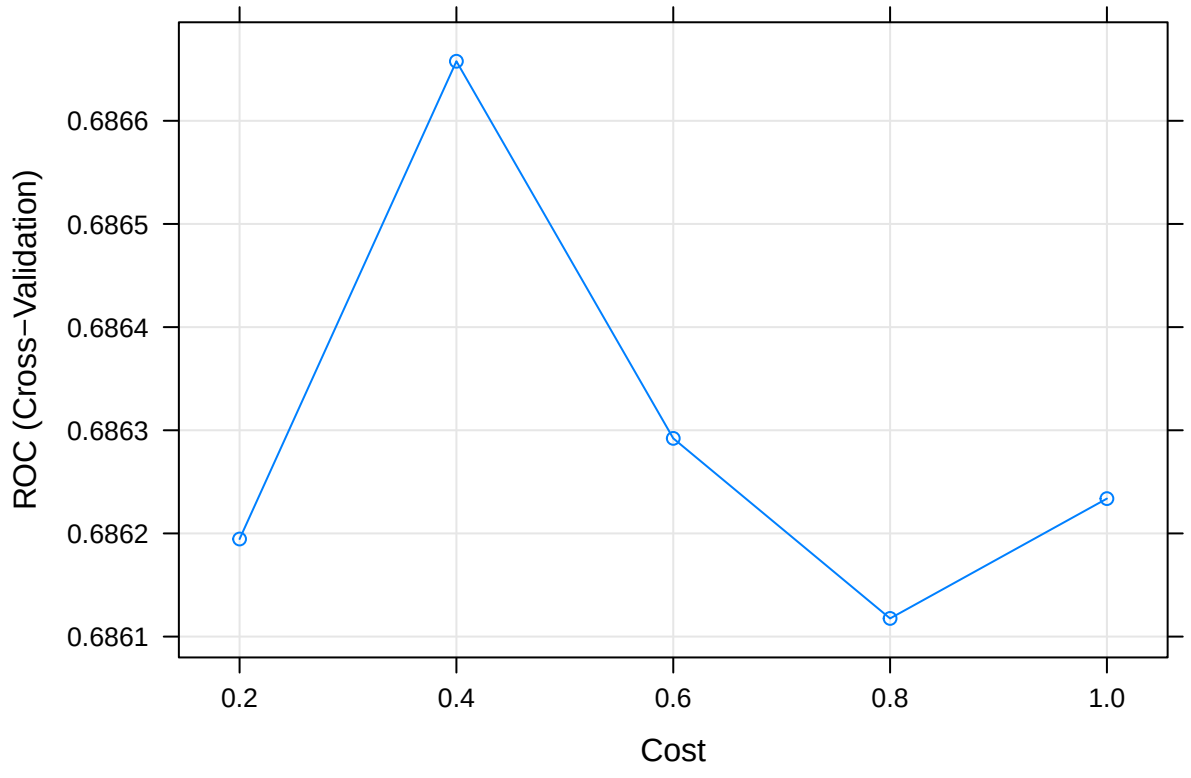
```
## The following object is masked from 'package:psych':
##
##     alpha
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
## The following object is masked from 'package:arules':
##
##     size
```

```
svmGrid <- expand.grid(C=seq(0.2,1,0.2))
svm.tune <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
method = "svmLinear",
preProc = c("center", "scale"),
tuneGrid = svmGrid,
metric = metric,
trControl = Ctrl)
plot(svm.tune)
```




```
set.seed(123)
svm.tune.ok <- train(TRAINSELECT2[-2], TRAINSELECT2$Attrition,
method = "svmLinear",
```

```
preProc = c("center", "scale"),
tuneGrid = data.frame(C=0.2),
metric = metric,
trControl = Ctrl_save)
```

Logistic Regression

```
#con tutte le variabili
set.seed(123)
logistic <- train(Attrition~., data=train, trControl=Ctrl_save, metric=metric,
method="glm",family=binomial())
pander(getTrainPerf(logistic))
```

TrainROC	TrainSens	TrainSpec	method
0.8865	0.7834	0.8358	glm

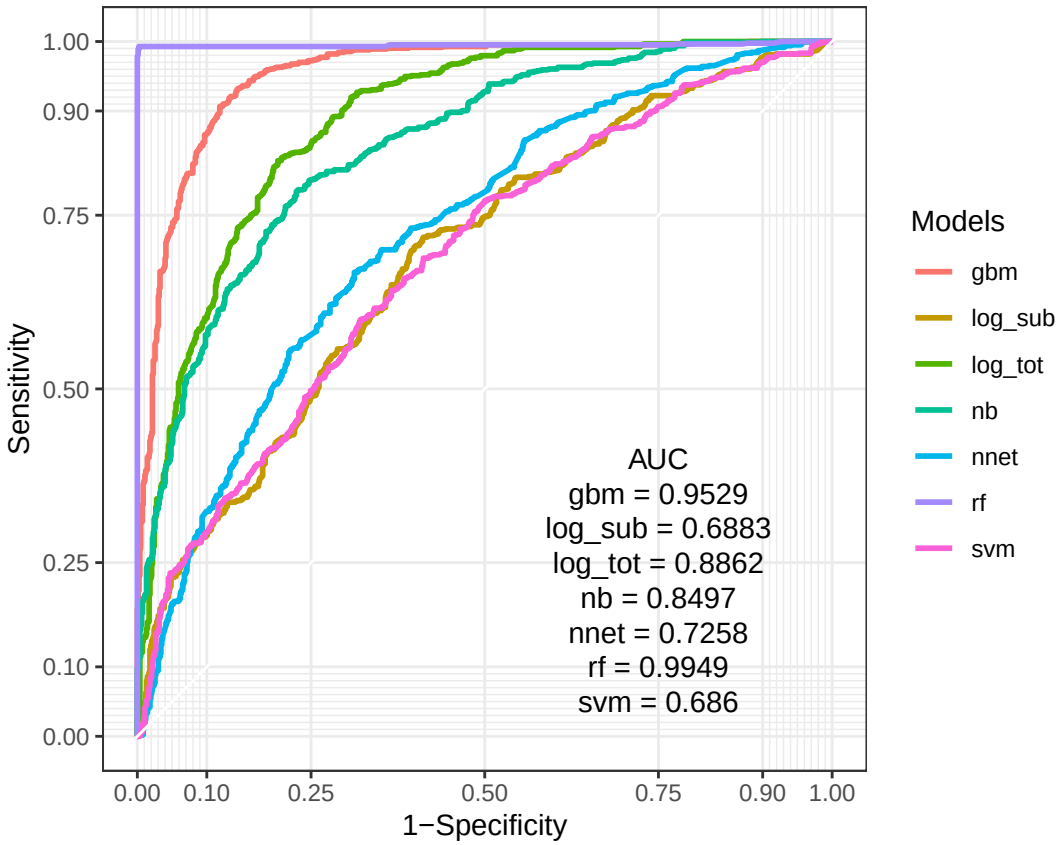
```
#con le 4 variabili selezionate

set.seed(123)
logistic_sub <- train(Attrition~., data=TRAINSELECT2, trControl=Ctrl_save, metric=metric,
method="glm",family=binomial())
pander(getTrainPerf(logistic_sub))
```

TrainROC	TrainSens	TrainSpec	method
0.6882	0.5423	0.7313	glm

Si confrontano le ROC curve dei modelli stimati aggiungendo il valore dell'AUC (area under the curve) calcolato su ogni modello.

```
roc_values <- cbind(as.data.frame(logistic_sub$pred$obs), as.data.frame(logistic_sub$pred$Yes))
gbm <- as.data.frame(STGfit.one.shot$pred$Yes)
log_tot <- as.data.frame(logistic$pred$Yes)
nb <- as.data.frame(NBfit$pred$Yes)
nnet <- as.data.frame(nnetFit_finale$pred$Yes)
rf <- as.data.frame(rpartTuneMyRf_ok$pred$Yes)
svm <- as.data.frame(svm.tune.ok$pred$Yes)

roc_values <- cbind(roc_values, log_tot, gbm, nb, nnet, rf, svm)
names(roc_values) <- c("obs","log_sub", "log_tot", "gbm", "nb", "nnet", "rf", "svm")
library(plotROC)
longtest <- melt_roc(roc_values, "obs", c("log_sub", "log_tot", "gbm", "nb", "nnet", "rf","svm"))
longtest$D <- ifelse(longtest$D=="Yes",1,0)
names(longtest)[3] <- "Models"
g <- ggplot(longtest, aes(m=M, d=D, color=Models)) +
geom_roc(n.cuts=0) +
coord_equal() +
style_roc(xlab="1-Specificity", ylab="Sensitivity")

g + annotate("text", x=0.75, y=0.4, label="AUC") +
annotate("text", x=0.75, y=0.35, label=paste("gbm =", round(calc_auc(g)$AUC[1], 4))) +
annotate("text", x=0.75, y=0.30, label=paste("log_sub =", round(calc_auc(g)$AUC[2], 4))) +
annotate("text", x=0.75, y=0.25, label=paste("log_tot =", round(calc_auc(g)$AUC[3], 4))) +
```

```
annotate("text", x=0.75, y=0.20, label=paste("nb =", round(calc_auc(g)$AUC[4], 4))) +
annotate("text", x=0.75, y=0.15, label=paste("nnet =", round(calc_auc(g)$AUC[5], 4))) +
annotate("text", x=0.75, y=0.10, label=paste("rf =", round(calc_auc(g)$AUC[6], 4))) +
annotate("text", x=0.75, y=0.05, label=paste("svm =", round(calc_auc(g)$AUC[7], 4)))
```



Il modello migliore, basandosi sulla ROC Curve, risulta essere il Random Forest con un AUC pari a 0.9389.

Per verificare ulteriormente la bontÃ di questo modello si confrontano i valori relativi a *precision, recall e F1-measure* calcolati testando i diversi classificatori sul test set.

```
test_model<- function(model, y, len = NULL, search = "grid") {
  test_pred <- predict(model,test)
  prec <- precision(data = test_pred, reference = test$Attrition)
  Fmeas <- F_meas(data = test_pred, reference = test$Attrition)
  rec <- recall(data = test_pred, reference = test$Attrition)

  return (c(prec, rec, Fmeas))
}
```

```
performance_value= as.data.frame(test_model(rpartTuneMyRf_ok))
performance_value$gbm= test_model(STGfit.one.shot)
performance_value$log_tot= test_model(logistic)
performance_value$nb= test_model(NBfit)
performance_value$nnet= test_model(nnetFit_finale)
colnames(performance_value)[1]= 'rf'
rownames(performance_value)= c('Precision','Recall','F1-measure')
kable(performance_value)
```

	rf	gbm	log_tot	nb	nnet
Precision	0.8618421	0.9109948	0.9361111	0.9203540	0.8742690
Recall	0.9656020	0.8550369	0.8280098	0.2555283	0.7346437
F1-measure	0.9107764	0.8821293	0.8787484	0.4000000	0.7983979

Come era prevedibile, il Random Forest risulta avere le metriche migliori, in particolare ha un valore della F1-measure pari a 0.9193.

Conclusioni

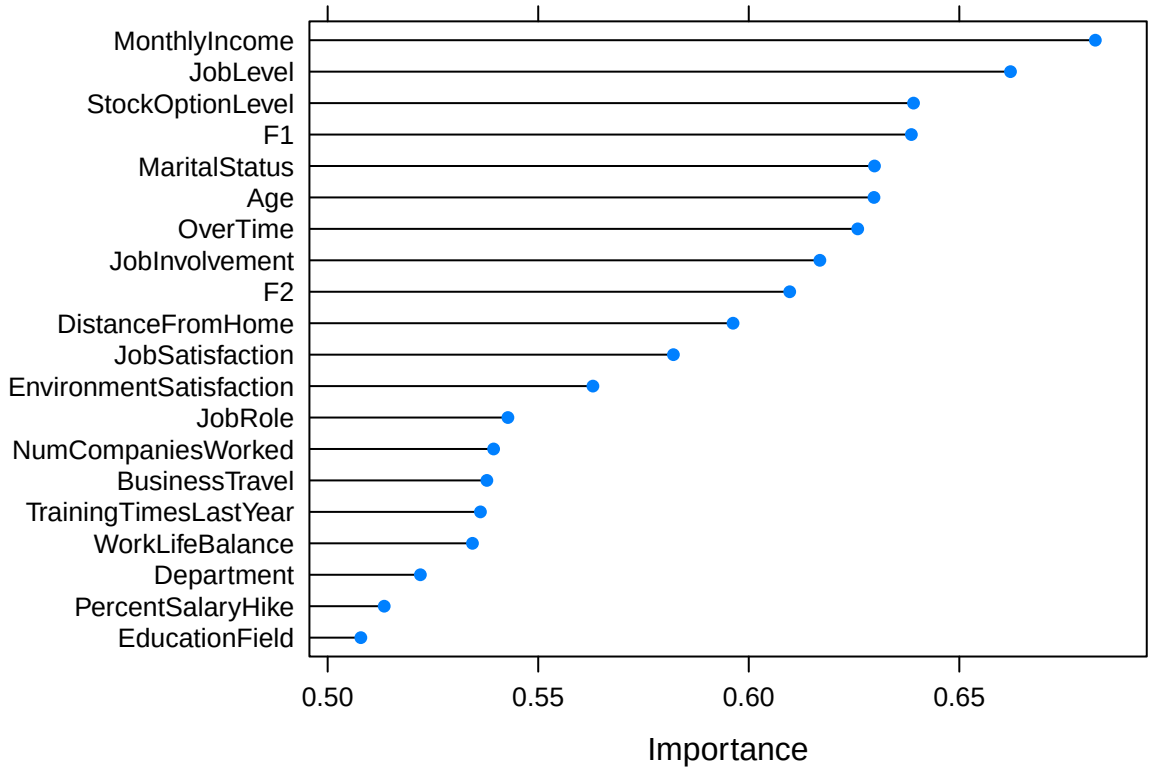
Per definire quali sono i fattori che piÃ¹ inducono un dipendente ad abbandonare l'azienda, si utilizza il modello Random Forest poichÃ¨ Ã¨ risultato essere il miglior classificatore.

Si stila un ranking delle variabili in base alla loro importanza nel prevedere la risposta *Attrition*.

```
importance_rf <- varImp(rpartTuneMyRf_ok, scale=FALSE)
# summarize importance
print(importance_rf)
```

```
## ROC curve variable importance
##
##                          Importance
## MonthlyIncome                0.6823
## JobLevel                     0.6622
## StockOptionLevel             0.6391
## F1                           0.6386
## MaritalStatus                0.6299
## Age                          0.6297
## OverTime                     0.6259
## JobInvolvement               0.6169
## F2                           0.6097
## DistanceFromHome             0.5963
## JobSatisfaction              0.5821
## EnvironmentSatisfaction      0.5630
## JobRole                      0.5428
## NumCompaniesWorked           0.5394
## BusinessTravel               0.5378
## TrainingTimesLastYear        0.5363
## WorkLifeBalance              0.5344
## Department                   0.5220
## PercentSalaryHike            0.5134
## EducationField               0.5079
```

```
# plot importance
plot(importance_rf)
```



Si puÃ² concludere che, nel prevedere se un dipendente abbandonerÃ la propria azienda, le variabili che maggiormente influiscono sulla risposta (con un'importanza maggiore del 60%) sono: - *MonthlyIncome*; - *Overtime*; - F1: *"Esperienza lavorativa del dipendente"* data dalla combinazione lineare di - *JobLevel*; - *Age*; - F2: *"Esperienza lavorativa nello stesso team"*; - *MaritalStatus*; - *StockOptionLevel*.

Applicazioni del modello

Per concludere, si effettuano dei test con un individuo esempio. L'obiettivo di questi test Ã¨ far vedere concretamente e confermare l'impotanza delle variabili cosÃ¬ come Ã¨ emersa grazie al modello Random Forest. Di seguito Ã¨ possibile osservare come, al cambiare del valore di una sola delle esplicative risultate piÃ¹ importanti, cambia la classe di appartenenza dell'individuo in questione.

Si utilizza a questo scopo un dipendente presente nel test set. In questo caso, il modello random forest predice correttamente la classe positiva Attrition.

A questo punto, vengono modificate le variabili "Monthly Income", "Overtime", "MaritalStatus" e "JobLevel", lasciando in ogni step invariate le altre.

```
user_test1=test[1,]

user_test5<-user_test4<-user_test3<-user_test2<-user_test1
user_test2$MonthlyIncome<-4180
user_test3$MaritalStatus<-'Married'
user_test4$OverTime<-'No'
user_test5$JobLevel <- as.factor(5)
user_test5$MonthlyIncome <- 3000
```

```
predict(rpartTuneMyRf_ok,user_test1)
```

```
## [1] No
## Levels: No Yes
```

```
predict(rpartTuneMyRf_ok,user_test2) #stipendio raddoppiato
```

```
## [1] No
## Levels: No Yes
```

```
predict(rpartTuneMyRf_ok,user_test3) #marital status da single a married
```

```
## [1] No
## Levels: No Yes
```

```
predict(rpartTuneMyRf_ok,user_test4) #overtime da yes a no
```

```
## [1] No
## Levels: No Yes
```

```
predict(rpartTuneMyRf_ok,user_test5) #jobLevel da 1 a 5
```

```
## [1] No
## Levels: No Yes
```

```
predict(rpartTuneMyRf_ok,user_test1, type="prob")
```

```
##     No  Yes
## 1 0.79 0.21
```

```
predict(rpartTuneMyRf_ok,user_test2, type="prob") #stipendio raddoppiato
```

```
##      No   Yes
## 1 0.788 0.212
```

```
predict(rpartTuneMyRf_ok,user_test3, type="prob") #marital status da single a married
```

```
##      No   Yes
## 1 0.782 0.218
```

```
predict(rpartTuneMyRf_ok,user_test4, type="prob") #overtime da yes a no
```

```
##     No  Yes
## 1 0.85 0.15
```

```
predict(rpartTuneMyRf_ok,user_test5, type="prob") #jobLevel da 1 a 5
```

```
##      No   Yes
## 1 0.724 0.276
```

E' possibile vedere che il modello prevede che la variabile Attrition del dipendente in questione passi da essere positiva e negativa nei seguenti casi: se lo stipendio raddoppia, se il dipendente passa da essere "single" a "married" e se non sono piÃ¹ richiesti straordinari (OverTime=No).

Cambiando il job level del dipendente, invece, anche se quest'ultimo viene promosso da un livello in azienda pari a 1 fino al livello piÃ¹ alto 5, comunque preferirÃ abbandonare la compagnia, sempre a paritÃ delle altre variabili. Tuttavia, si puÃ² osservare che, se questa ipotetica promozione viene accompagnata da un minimo aumento dello stipendio mensile, allora Attrition viene predetta come negativa e il dipendente con una probabilitÃ pari a 55% non abbandonerÃ la compagnia.

Concludendo, questi esempi sono serviti per mettere in luce e confermare l'importanza delle variabili emersa grazie al modello Random Forest.