Giorgia Gossi
Aurora Musitelli
Chiara Zani

ELABORATO 2: DATA MINING

DATA DICTIONARY		
1	State	Sato:Lo stato in cui risiede il cliente, indicato da una sigla di due lettere
2	Account.length	Durata account: il numero di giorni in cui questo account è stato attivo
3	Area.code	Prefisso: il prefisso a tre cifre del numero di telefono del cliente corrispondente
4	International.plan	Piano internazionale: se il cliente ha un piano tariffario internazionale: yes/no
5	Voice.mail.plan	Piano Voice Mail: se il cliente ha una funzione di posta vocale: yes/no
6	Number.vmail.messages	Messaggio VMail: il numero medio di messaggi di posta vocale al mese
7	Total.day.minutes	Minuti giornalieri: il numero totale di minuti di chiamata utilizzati durante il giorno
8	Total.day.calls	Chiamate giornaliere: il numero totale di chiamate effettuate durante il giorno
9	Total.day.charge	Day Charge: il costo fatturato delle chiamate diurne
10	Total.eve.minutes	Eve Mins: il numero totale di minuti di chiamata utilizzati durante la serata
11	Total.eve.calls	Eve Calls: il numero totale di chiamate effettuate durante la serata
12	Total.eve.charge	Eve Charge: il costo fatturato delle chiamate serali
13	Total.night.minutes	Minuti notturni: il numero totale di minuti di chiamata utilizzati durante la notte
14	Total.night.calls	Chiamate notturne: il numero totale di chiamate effettuate durante la notte
15	Total.night.charge	Night Charge: il costo fatturato delle chiamate notturne
16	Total.intl.minutes	Intl Mins: il numero totale di minuti internazionali
17	Total.intl.calls	Chiamate internazionali: il numero totale di chiamate internazionali
18	Total.intl.charge	Intl Charge: il costo fatturato per le chiamate internazionali
19	Customer.service.calls	Chiamate CustServ: il numero di chiamate effettuate al Servizio Clienti
20	Churn	Churn: se il cliente ha lasciato il servizio: vero=1/falso=0

Orange Telecom è la maggiore impresa di telecomunicazioni in Francia. Con 170.000 dipendenti e 230,7 milioni di clienti nel mondo, è una delle principali aziende mondiali del settore. Vediamo più da vicino di cosa si occupa esattamente l'operatore telefonico Orange. In particolare, le sue attività sono:

- la telefonia fissa, Internet, telefonia IP, videotelefonia, televisione digitale con "Orange TV" e contenuti multimediali;
- la telefonia mobile;
- i servizi di comunicazione aziendale (con il marchio Orange Business Services).

L'obiettivo dell'analisi è quello di prevedere se il cliente della compagnia telefonica Orange Telecom deciderà di abbandonarla o di rimanere fedele (**CHURN ANALYSIS**).

ANALISI

Carichiamo il dataset:

```
str(ds)

## 'data.frame':    3333 obs. of  20 variables:
##  $ State                 : chr  "KS" "OH" "NJ" "OH" ...
##  $ Account.length        : int  128 107 137 84 75 118 121 147 141 74 ...
##  $ Area.code             : int  415 415 415 408 415 510 510 415 415 415 ...
##  $ International.plan    : chr  "No" "No" "No" "Yes" ...
##  $ Voice.mail.plan       : chr  "Yes" "Yes" "No" "No" ...
##  $ Number.vmail.messages : int  25 26 0 0 0 0 24 0 37 0 ...
##  $ Total.day.minutes     : chr  "265.1" "161.6" "243.4" "299.4" ...
##  $ Total.day.calls       : int  110 123 114 71 113 98 88 79 84 127 ...
##  $ Total.day.charge      : chr  "45.07" "27.47" "41.38" "50.9" ...
##  $ Total.eve.minutes     : chr  "197.4" "195.5" "121.2" "61.9" ...
##  $ Total.eve.calls       : int  99 103 110 88 122 101 108 94 111 148 ...
##  $ Total.eve.charge      : chr  "16.78" "16.62" "10.3" "5.26" ...
##  $ Total.night.minutes   : chr  "244.7" "254.4" "162.6" "196.9" ...
##  $ Total.night.calls     : int  91 103 104 89 121 118 118 96 97 94 ...
##  $ Total.night.charge    : chr  "11.01" "11.45" "7.32" "8.86" ...
##  $ Total.intl.minutes    : chr  "10.0" "13.7" "12.2" "6.6" ...
##  $ Total.intl.calls      : int  3 3 5 7 3 6 7 6 5 5 ...
##  $ Total.intl.charge     : chr  "2.7" "3.7" "3.29" "1.78" ...
##  $ Customer.service.calls: int  1 1 0 2 3 0 3 0 0 0 ...
##  $ Churn                 : chr  "False" "False" "False" "False" ...
```

1. Distribuzione della variabile target y: è realistica dal punto di vista interpretativo?

```
table(ds$Churn)/nrow(ds)

##
##     False      True
## 0.8550855 0.1449145

table(ds$Churn)


## False  True
##  2850   483
```

distribuzione della variabile target realistica, ossia l'85% (2850 soggetti) dei clienti non lascia la compagnia Orange telecom mentre il 14% (483 soggetti) dei clienti lascia la compagnia Orange Telecom.

2. Utilizziamo la matrice di costi e profitti: costa di più classificare come sleale(churn, 'True') un cliente leale (non churn, 'False')

```
##            False_pred True_pred
## False_obs          10       -100
## True_obs           -5          1
```

Funzione che calcola il profitto totale come metrica per tunare modelli

```
f1 <- function(data, lev = NULL, model = NULL) {
  risul <- confusionMatrix(data$pred, data$obs)
  confmat <- risul$table
  f1_val <- costmat[1,1] * confmat[1,1] + costmat[1,2] * confmat[1,2] + costmat[2,
1] * confmat[2,1] + costmat[2,2] * confmat[2,2]
  c(F1 = f1_val)
}
```

PREPROCESSING

Prima della costruzione dei modelli abbiamo svolto il pre-processing sulle covariate. Inizialmente abbiamo ricodificato la variabile target come variabile fattoriale e successivamente abbiamo deciso di non eliminare subito le variabili collineari e quelle con nearzerovariance, per vedere se saranno eliminate durante la model selection effettuata dall'albero.

1.VALORI MANCANTI

```
sapply(ds, function(x)(sum(is.na(x))))
#conteggio dei valori mancanti
##                   State          Account.length              Area.code
##                       0                       0                       0
##     International.plan        Voice.mail.plan  Number.vmail.messages
##                       0                       0                       0
##      Total.day.minutes        Total.day.calls       Total.day.charge
##                       0                       0                       0
##      Total.eve.minutes        Total.eve.calls       Total.eve.charge
##                       0                       0                       0
##    Total.night.minutes      Total.night.calls     Total.night.charge
##                       0                       0                       0
##     Total.intl.minutes       Total.intl.calls      Total.intl.charge
##                       0                       0                       0
## Customer.service.calls                   Churn
##                       0                       0
```

il dataset non presenta nessun valore mancante

2.COLLINEARITA'

```
correlatedPredictors = findCorrelation(R, cutoff = 0.95, names = TRUE)
correlatedPredictors
## [1] "Total.intl.charge"   "Total.eve.charge"    "Total.night.minutes"
## [4] "Total.day.charge"
```

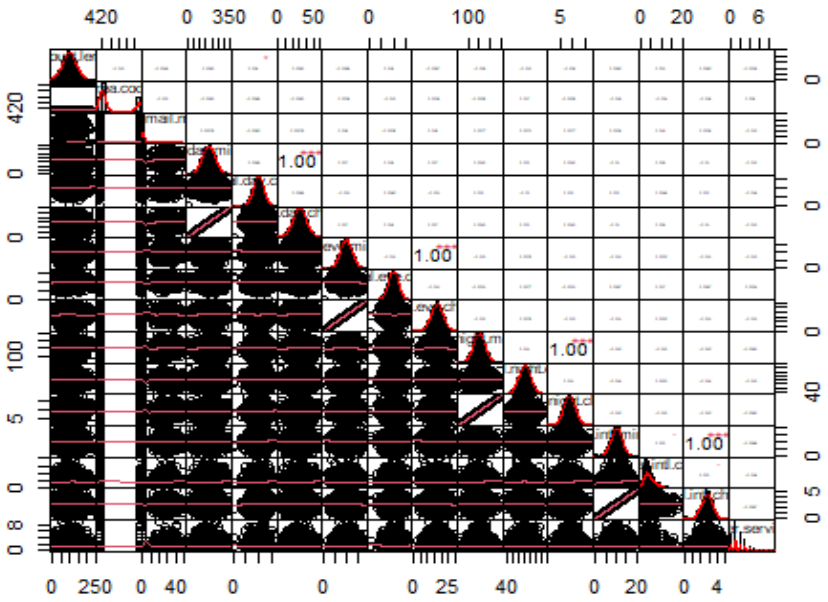
risultano correlate tra di loro le seguenti covariate:

"Total.day.minutes" e "Total.day.charge" = cor 0.999

"Total.intl.charge" e "Total.intl.minutes" = cor 0.999

"Total.eve.charge" e "Total.eve.minutes" = cor 0.999

"Total.night.charge" e "Total.night.minutes" = cor 0.999



3.COVARIATE CON VARIANZA ZERO (zeroVar) E VARIANZA VICINO A ZERO (nzv)

```
nzv = nearZeroVar(training, saveMetrics = TRUE)
nzv
```

```
##                         freqRatio percentUnique zeroVar   nzv
## State                    1.257143    1.91297824   FALSE FALSE
## Account.length           1.060606    7.68942236   FALSE FALSE
## Area.code                1.941090    0.11252813   FALSE FALSE
## International.plan       8.874074    0.07501875   FALSE FALSE
## Voice.mail.plan          2.637108    0.07501875   FALSE FALSE
## Number.vmail.messages   38.660000    1.57539385   FALSE  TRUE
## Total.day.minutes        1.000000   55.85146287   FALSE FALSE
## Total.day.calls          1.050847    4.31357839   FALSE FALSE
## Total.day.charge         1.000000   55.85146287   FALSE FALSE
## Total.eve.minutes        1.142857   54.08852213   FALSE FALSE
## Total.eve.calls          1.032258    4.50112528   FALSE FALSE
## Total.eve.charge         1.000000   48.79969992   FALSE FALSE
## Total.night.minutes      1.166667   54.16354089   FALSE FALSE
## Total.night.calls        1.044776    4.42610653   FALSE FALSE
## Total.night.charge       1.181818   33.19579895   FALSE FALSE
## Total.intl.minutes       1.148936    5.92648162   FALSE FALSE
## Total.intl.calls         1.081511    0.78769692   FALSE FALSE
## Total.intl.charge        1.148936    5.92648162   FALSE FALSE
## Customer.service.calls   1.554276    0.37509377   FALSE FALSE
## Churn                    5.871134    0.07501875   FALSE FALSE
```

Number vmail messages near zero variance.

3.Altre strategie di preprocessing sono lo SCALING che facciamo durante il tuning dei modelli che lo richiedono

Suddividiamo il dataset di partenza in training, validation e score

```
library(caret)
set.seed(1234)
split <- createDataPartition(y=ds$Churn, p = 0.3, list = FALSE)
validation <- ds[split,]
train <- ds[-split,]
split1 <- createDataPartition(y=validation$Churn, p = 0.1, list = FALSE)
score <- validation[split1,]
validation <- validation[-split1,]
```

adesso il dataset da utilizzare e' 'train' (inteso come training)

STEP1: Costruzione di modelli

Nello step 1 abbiamo elaborato diversi modelli i quali sono stati costruiti cercando di massimizzare come metrica il profitto totale. Questo è definito come somma pesata degli elementi della matrice di confusione del modello, ad ogni sua iterazione, e utilizzando come pesi della somma le celle corrispondenti della matrice di profitti e costi; in questo modo l'algoritmo di ogni modello salva come regola decisionale finale quella definita nell'iterazione che massimizza il profitto.

Il primo modello che abbiamo valutato è stato l'albero, che utilizzeremo anche come model selector

TREE

```
library(caret)
set.seed(1)
metric <- "F1"
cvCtrl <- trainControl(method = "cv", number=10, search="grid", classProbs = TRUE,
                       summaryFunction = f1)
Tree <- train(Churn ~ ., data = train, method = "rpart",
                      tuneLength = 10, metric=metric,
                      trControl = cvCtrl)
Tree
```

```
## CART
##
## 2333 samples
##   19 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
## Resampling results across tuning parameters:
```

```
##
##   cp          F1
##   0.00443787    884.7
##   0.00887574    894.8
##   0.01183432    842.4
##   0.01479290    831.2
##   0.01775148    747.8
##   0.02514793    584.0
##   0.02662722    555.2
##   0.04289941    151.6
##   0.04881657   -155.5
##   0.08284024  -1171.8
##
## F1 was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.00887574.
```

```
getTrainPerf(Tree)
```

```
##   TrainF1 method
## 1   894.8  rpart
```

```
confusionMatrix(Tree)
```

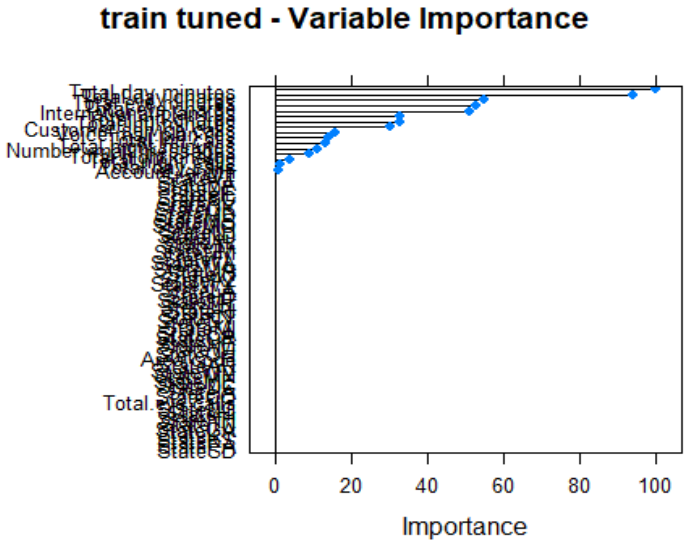
```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##

##           Reference
## Prediction False True
##      False  83.4  4.5
##      True    2.1 10.0
##
##  Accuracy (average) : 0.934
```

```
# var imp of the tree
varImp(object=Tree)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 68)
##
##                          Overall
## Total.day.minutes      100.0000
## Total.day.charge        94.2033
## Total.eve.minutes       54.9611
## Total.eve.charge        52.5834
## International.planYes   51.2196
## Total.intl.charge       32.6996
## Total.intl.minutes      32.6996
## Customer.service.calls  30.3801
```

```
## Voice.mail.planYes      15.7219
## Total.intl.calls        14.1719
## Total.night.minutes     13.1534
## Number.vmail.messages   11.2021
## Total.night.charge       8.7340
## Total.night.calls        3.6753
## Total.day.calls          1.2352
## Account.length           0.8319
## StateDE                  0.0000
## StateNE                  0.0000
## StateAR                  0.0000
## StateMA                  0.0000
```

```
plot(varImp(object=Tree), main="train tuned - Variable Importance")
```




```
# select only important variables
vi=as.data.frame(Tree$finalModel$variable.importance)
viname=row.names(vi)
viname
```

```
##  [1] "Total.day.minutes"      "Total.day.charge"       "Customer.service.calls"
##  [4] "Total.eve.charge"       "Total.eve.minutes"      "Total.intl.charge"
##  [7] "Total.intl.minutes"     "Total.intl.calls"       "International.planYes"
## [10] "Number.vmail.messages"  "Voice.mail.planYes"     "Total.night.minutes"
## [13] "Total.night.charge"     "Total.eve.calls"        "StateME"
## [16] "StateMT"                "Account.length"         "Total.day.calls"
## [19] "StateKY"                "StateAL"                "StateCA"
## [22] "Total.night.calls"      "StateMS"                "StateVT"
## [25] "StateNJ"
```

Notiamo che l'albero non ha risolto la collinearità quindi abbiamo tolto le variabili minutes che erano collineari e number.vmail.messages che presentava near zero variance.

```
# creiamo I due dataset con la model selection dell'albero
list=c("Total.day.charge" ,     "Total.eve.charge" ,
 "Customer.service.calls",     "Total.intl.charge" ,     "International.plan" ,
  "Total.intl.calls"  , "Voice.mail.plan",
"Total.night.charge" ,    "Total.day.calls" ,       "Total.night.calls" ,     "Acc
ount.length" ,
 "Area.code"  ,            "Total.eve.calls" ,       "State")

train2=train[,list]
validation2=validation[,list]
validation2=cbind(validation$Churn, validation2)
names(validation2)[1] <- 'Churn'
```

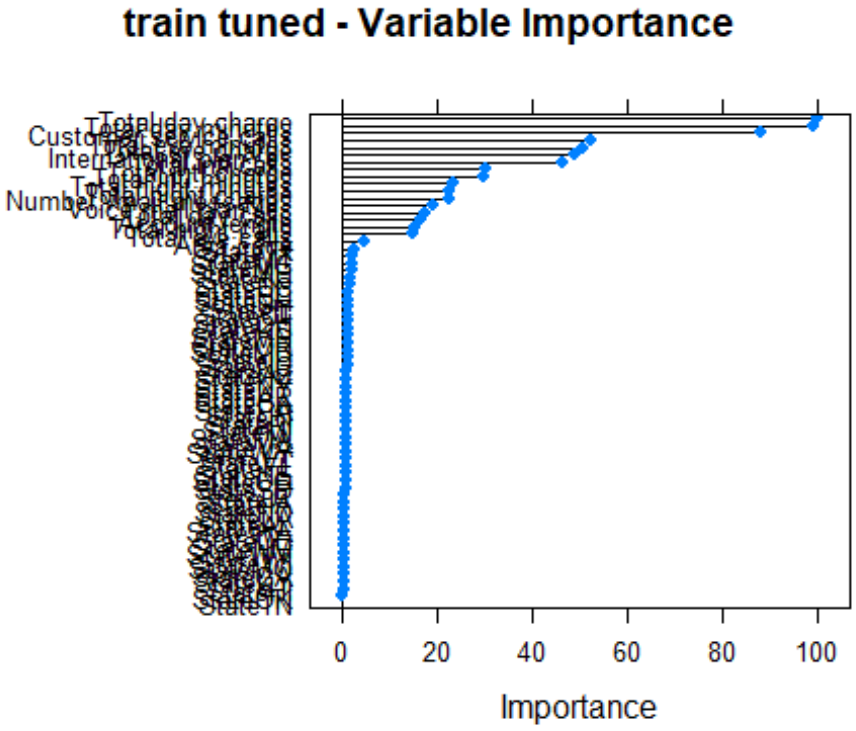
RANDOM FOREST

Il secondo modello che abbiamo valutato è il Random Forest

```
library(caret)
set.seed(1)
ctrl =trainControl(method="cv", number = 10, classProbs = T,
                   summaryFunction=f1)
rfTune <- train(Churn ~ ., data = train, method = "rf",
                tuneLength = 10, metric=metric,
                trControl = ctrl)
rfTune
```

```
## Random Forest
##
## 2333 samples
##   19 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
## Resampling results across tuning parameters:
##
##   mtry  F1
##    2    -1183.0
##    9      554.9
##   16      974.2
##   24     1013.1
##   31     1021.7
##   38     1005.6
##   46     1008.6
##   53     1004.1
##   60      973.8
##   68     1004.1
```

```
## F1 was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 31.

confusionMatrix(rfTune)

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  84.4  4.1
##      True    1.1 10.4
##
##  Accuracy (average) : 0.9481

# permutation importance
vimp=varImp(rfTune)
plot(varImp(object=rfTune),main="train tuned - Variable Importance")
```



Abbiamo tolto le var collineari e quella con nzv

```
list2=c( "Account.length" ,        "International.plan" , "Voice.mail.plan"  ,
"Total.day.calls"    ,    "Total.day.charge"    ,
  "Total.eve.calls"  ,      "Total.eve.charge"   ,      "Total.night.calls"  ,
"Total.night.charge"  ,   "Total.intl.minutes"   ,  "Total.intl.calls"  ,     "Tot
al.intl.charge"    ,
 "Customer.service.calls")
```

```
#creiamo I nuovi dataset di training e validation con model selection della random forest

train3=train[,list2]
train3=cbind(train$Churn, train3)
names(train3)[1] <- "Churn"

validation3=validation[,list2]
validation3=cbind(validation$Churn, validation3)
names(validation3)[1] <- "Churn"
```

GLM

Il terzo modello valutato è un logistico

```
set.seed(1)
ctrl =trainControl(method="cv", number = 10, classProbs = T,
                   summaryFunction=f1)
glm=train(Churn~.,
          data=train2,method = "glm", metric= metric,
          trControl = ctrl, tuneLength=10, trace=TRUE, na.action = na.pass)

## Generalized Linear Model
##
## 2333 samples
##   14 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
## Resampling results:
##
##   F1
##   -566.8

confusionMatrix(glm)

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  82.3 10.5
##      True    3.2  3.9
##
##  Accuracy (average) : 0.8628
```

K-NEAREST NEIGHTBOUR

Come quarto modello abbiamo valutato un Nearest Neighbor

```
set.seed(1)
ctrl =trainControl(method="cv", number = 10, classProbs = T,
                   summaryFunction=f1)
grid = expand.grid(k=seq(5,20,3))
knn=train(Churn~.,
          data=train2,method = "knn",
          trControl = ctrl, tuneLength=10, na.action = na.pass, metric=metric,
          tuneGrid=grid, preProcess=c("scale","corr"))
knn
```

```
## k-Nearest Neighbors
##
## 2333 samples
##   14 predictor
##    2 classes: 'False', 'True'
##
## Pre-processing: scaled (63)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
## Resampling results across tuning parameters:
##
##   k   F1
##    5  -1279.2
##    8  -1311.7
##   11  -1374.9
##   14  -1374.9
##   17  -1385.0
##   20  -1385.0
##
## F1 was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
confusionMatrix(knn)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  84.8 13.9
##      True    0.7  0.6
##
##  Accuracy (average) : 0.8534
```

LASSO

Il quinto modello costruito è Lasso

```
set.seed(1)
ctrl =trainControl(method="cv", number = 10, classProbs = T,
                   summaryFunction=f1)
grid = expand.grid(.alpha=1,.lambda=seq(0, 1, by = 0.01))
lasso=train(Churn~.,
            data=train2,method = "glmnet",
            trControl = ctrl, tuneLength=10, na.action = na.pass, metric=metric,
            tuneGrid=grid)
lasso
```

```
## glmnet
##
## 2333 samples
##   14 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
## Resampling results across tuning parameters:
##
##
## Tuning parameter 'alpha' was held constant at a value of 1
## F1 was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.
```

```
confusionMatrix(lasso)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  82.3 10.5
##      True    3.2  3.9
##
##  Accuracy (average) : 0.8628
```

NAIVE BAYES

Il sesto modello costruito è il Naïve Bayes

```
set.seed(1)
ctrl =trainControl(method="cv", number = 10, classProbs = T,
                   summaryFunction=f1)
naivebayes=train(Churn~.,
                 data=train2, method = "naive_bayes", metric=metric,
                 trControl = ctrl, tuneLength=10, na.action = na.pass)
naivebayes
```

```
## Naive Bayes
##
## 2333 samples
##   14 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2100, 2100, 2100, 2099, 2099, 2101, ...
```

```
confusionMatrix(naivebayes)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction False True
##      False  46.9  5.5
##      True   38.6  9.0
##
##  Accuracy (average) : 0.5589
```

PLS regression

Come settimo modello abbiamo svolto un Partial Least Square

```
library(pls)

set.seed(1234)
Control=trainControl(method= "cv",number=10, classProbs=TRUE,
                     summaryFunction=f1)
pls=train(Churn~. , data=train2 , method = "pls", metric=metric,
          trControl = Control, tuneLength=10)
pls
```

```
## Partial Least Squares
## 2333 samples
```

```
##   14 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2099, 2100, 2099, 2099, 2099, 2100, ...
## Resampling results across tuning parameters:
##
##   ncomp  F1
##    1     -1385.0
##    2     -1385.0
##    3     -1385.0
##    4     -1385.0
##    5     -1385.0
##    6     -1385.0
##    7     -1344.6
##    8     -1337.5
##    9     -1297.5
##   10     -1242.5
##
## F1 was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 10.
```

```
confusionMatrix(pls)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  85.3 13.8
##      True    0.3  0.6
##
##  Accuracy (average) : 0.859
```

RETE NEURALE

L'ottavo ed ultimo modello valutato è la Rete Neurale

```
ctrl = trainControl(method="cv", number=10, summaryFunction = f1, classProbs = TRU
E)
nnetFit_CV <- train(Churn~. , data=train3 ,
                    method = "nnet",
                    tuneLength = 10, metric=metric, trControl=ctrl,
                    trace = FALSE,
                    maxit = 100)
nnetFit_CV
```

```
## Neural Network
##
## 2333 samples
##   13 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2099, 2100, 2099, 2100, 2101, 2099, ...

## F1 was used to select the optimal model using the largest value.
## The final values used for the model were size = 15 and decay = 0.003162278.

confusionMatrix(nnetFit_CV)

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction False True
##      False  82.4  9.3
##      True    3.1  5.2
##
##  Accuracy (average) : 0.8757
```

STEP 2: Assesment

MODELLI STIMATI:

1 - rfTune
2 - Tree
3 - lasso
4 - glm
5 - knn
6 - naivebayes
7 - pls
8 - nnetFit_CV

Per ogni modello stimiamo gli EPi(churn) e EPi(non churn), se il profitto di classificare un'unità come churn è maggiore di quello di classificarla come non churn allora sarà classificata come churn

```
#---------step 2-------------------------------------------------------
#1 modello rfTune
costmat

##           False_pred True_pred
## False_obs         10      -100
## True_obs          -5         1
```

```r
predProb <-  predict(rfTune,  validation, type = "prob")[,1] #probabilita' previst
a di essere non churn (False)

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation$Churn,decision)
decision_table
```

```
##        decision
##         False True
##   False   769    0
##   True     95   35
```

```r
total_profit_model_rf = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*cost
mat[2,1]+decision_table[1,2]*costmat[1,2]+decision_table[2,2]*costmat[2,2])
total_cost_model_rf   = -total_profit_model_rf

mean_profit_model_rf = total_profit_model_rf/ nrow(validation)
mean_cost_model_rf   = -mean_profit_model_rf



#2 modello Tree
predProb <-  predict(Tree,  validation, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation$Churn,decision)
decision_table
```

```
##        decision
##         False True
##   False   769    0
##   True    101   29
```

```r
total_profit_tree = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat[
2,1]+decision_table[1,2]*costmat[1,2]+decision_table[2,2]*costmat[2,2])
total_cost_tree   = -total_profit_tree



mean_profit_tree = total_profit_tree/ nrow(validation)
mean_cost_tree   = -mean_profit_tree
```

```r
#3 modello lasso
predProb <-  predict(lasso,  validation2, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation2$Churn,decision)
decision_table
```

```
##        decision
##         False True
##   False   769    0
##   True    129    1
```

```r
total_profit_lasso = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat
[2,1]+decision_table[1,2]*costmat[1,2]+decision_table[2,2]*costmat[2,2])
total_cost_lasso   = -total_profit_lasso

mean_profit_lasso = total_profit_lasso/ nrow(validation2)
mean_cost_lasso   = -mean_profit_lasso



#4 modello glm
predProb <-  predict(glm,  validation2, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation2$Churn,decision)
decision_table
```

```
##        decision
##         False True
##   False   769    0
##   True    129    1
```

```r
total_profit_glm = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat[2
,1]+decision_table[1,2]*costmat[1,2]+decision_table[2,2]*costmat[2,2])
total_cost_glm  = -total_profit_glm

mean_profit_glm = total_profit_glm/ nrow(validation2)
mean_cost_glm   = -mean_profit_glm
```

```r
#5 modello knn
predProb <-  predict(knn, validation2, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation2$Churn,decision)
decision_table
```

```
##        decision
##         False
##   False   769
##   True    130
```

```r
total_profit_knn = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat[2
,1])
total_cost_knn   = -total_profit_knn

mean_profit_knn = total_profit_knn/ nrow(validation2)
mean_cost_knn   = -mean_profit_knn

#6 modello naivebayes
predProb <-  predict(naivebayes,  validation2, type = "prob")[,1]


EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation2$Churn,decision)
decision_table
```

```
##        decision
##         False True
##   False   544  225
##   True     63   67
```

```r
total_profit_naivebayes = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*co
stmat[2,1]+decision_table[1,2]*costmat[1,2]+decision_table[2,2]*costmat[2,2])
total_cost_naivebayes   = -total_profit_naivebayes


mean_profit_naivebayes = total_profit_naivebayes/ nrow(validation2)
mean_cost_naivebayes   = -mean_profit_naivebayes
```

```r
#7 modello pls
predProb <-  predict(pls,  validation2, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation2$Churn,decision)
decision_table
```

```
##        decision
##         False
##   False   769
##   True    130
```

```r
total_profit_pls = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat[2
,1])
total_cost_pls   = -total_profit_pls


mean_profit_pls = total_profit_pls/ nrow(validation2)
mean_cost_pls   = -mean_profit_pls



#8 modello rete neurale
predProb <-  predict(nnetFit_CV,  validation3, type = "prob")[,1]

EP_churn  = (1-predProb)*costmat[2,2] + predProb*costmat[1,2]
EP_nchurn = predProb*costmat[1,1] + (1-predProb)*costmat[2,1]

decision = ifelse(EP_churn > EP_nchurn, 'True','False')

decision_table = table(validation3$Churn,decision)
decision_table
```

```
##        decision
##         False
##   False   769
##   True    130
```

```r
total_profit_nnet = (decision_table[1,1]*costmat[1,1]+decision_table[2,1]*costmat[
2,1])
total_cost_nnet   = -total_profit_nnet


mean_profit_nnet = total_profit_nnet/ nrow(validation3)
mean_cost_nnet   = -mean_profit_nnet
```

Confrontiamo il profitto medio dei modelli: scegliamo quello con profitto medio più alto

```
mean_profit_glm

## [1] 7.837597

mean_profit_knn

## [1] 7.830923

mean_profit_lasso

## [1] 7.837597

mean_profit_model_rf

## [1] 8.064516

mean_profit_naivebayes

## [1] -19.2525

mean_profit_nnet

## [1] 7.830923

mean_profit_pls

## [1] 7.830923

mean_profit_tree

## [1] 8.024472
```

Vince il modello **random forest** con profitto medio di 8.06

STEP 3: Evaluation

Dobbiamo scegliere la soglia che ci garantisca un profitto più alto

```
#---------step 3-----------------------------------------------------------------
----------
# NB evento True=Churn

# Calcoliamo la matrice in cui vedere i profitti totali e medi al variare della
soglia
result_df <- data.frame(
  threshold      = seq( from = 0.00, to = 1.0, by = 0.01),
  total_exp_profit = rep( 0, 101),
  mean_exp_profit  = rep( 0, 101)
  )
```

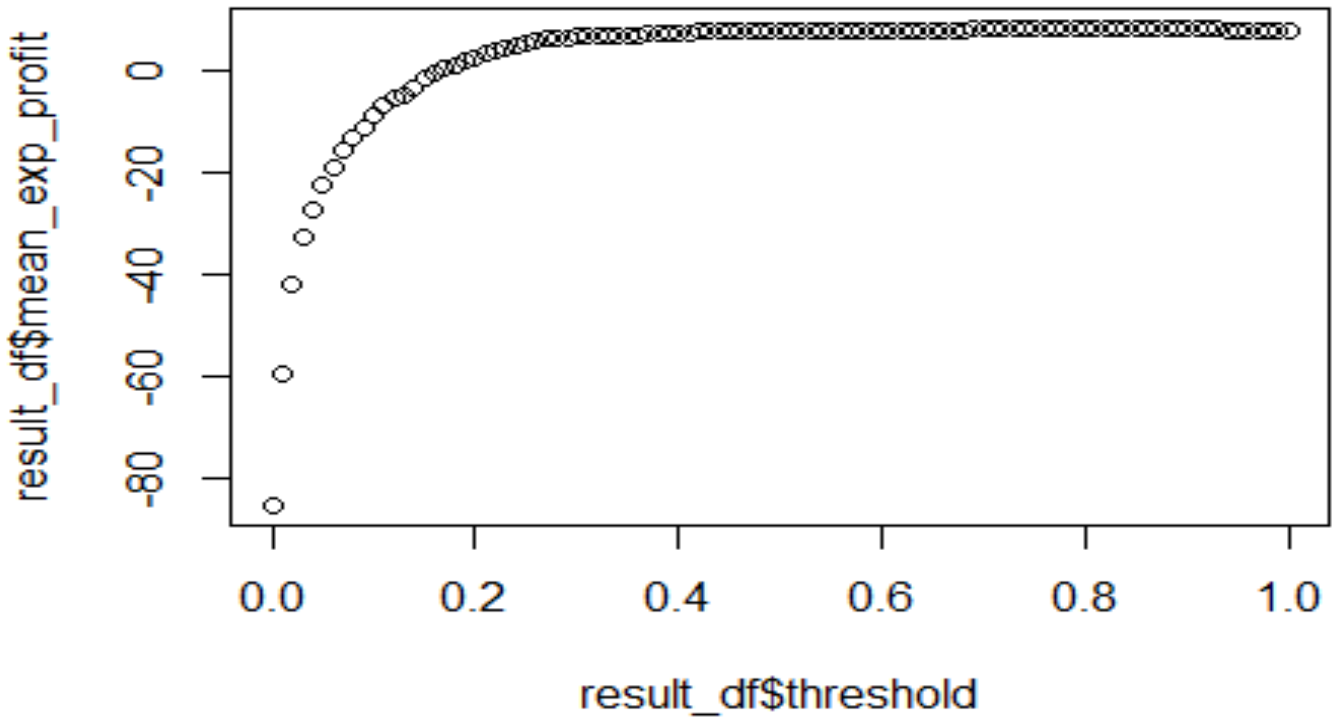
```r
predProb_C <- predict(rfTune,  validation, type = "prob")[,2]  # prob di essere predetti come True, cioè come churn

y<-ifelse(validation$Churn=='True', 2,1) #y=2 vuol dire True (churn), y=1 False (non churn)

i <- 0
for(threshold in seq(from = 0.00, to = 1.0, by = 0.01)){
  i <- i + 1
  prediction_v <- 1 + as.numeric(predProb_C >= threshold)
  match_count  <- sum(prediction_v == y)

  true_negative_count <- sum( prediction_v * y == 1    )        # nb true target 1*predicted 1 = results is 1
  true_positive_count <- sum( prediction_v * y == 4    )        # nb true target 2*predicted 2 = results is 4

  false_positive_count <- sum( prediction_v > y        )        #    predicted 2 , true target 1
  false_negative_count <- sum( prediction_v < y        )        #    predicted 1 , true target 2

  total_exp_profit <-
    (-100)  * false_positive_count +
    (-5)  * false_negative_count +
    1 * true_positive_count  +
    10 * true_negative_count

  mean_exp_profit <- total_exp_profit / nrow(validation)

  result_df$mean_exp_profit[i]   <- mean_exp_profit
  result_df$total_exp_profit[i]  <- total_exp_profit

}

head(result_df)
```

```
##   threshold total_exp_profit mean_exp_profit
## 1      0.00           -76770       -85.39488
## 2      0.01           -53602       -59.62403
## 3      0.02           -37890       -42.14683
## 4      0.03           -29328       -32.62291
## 5      0.04           -24604       -27.36819
## 6      0.05           -20320       -22.60289
```

Questi sono i profitti medi attesi

Grafico: come cambia il profitto medio a seconda della soglia

```
plot(result_df$threshold,result_df$mean_exp_profit)
```




```
soglia1 <- result_df[which(result_df$mean_exp_profit == max(result_df$mean_exp_pro
fit)), ]
soglia1
```

```
##    threshold total_exp_profit mean_exp_profit
## 71      0.70             7532        8.378198
## 72      0.71             7532        8.378198
```

Due soglie presentano lo stesso profitto atteso quindi decidiamo di scegliere la soglia con valore 0.7

```
soglia<-0.7
```

Applichiamo la soglia sul dataset di validation

```
p_validation = predict(rfTune, validation, "prob")
probc_validation=p_validation[,2] #prob di essere classificato come churn
# applico la regola decisionale utilizzando la soglia dello step 3
pred_validation=ifelse(probc_validation>soglia, "True","False")
table_val <- table(validation$Churn,pred_validation)
table_val
```

```
# Decision matrix sul validation
```

```
##        pred_validation
##         False True
##   False   769    0
##   True     48   82
```

```
table_val/dim(validation)[1]
```

```
##        pred_validation
##              False       True
##   False 0.85539488 0.00000000
##   True  0.05339266 0.09121246
```

Profitto totale e medio sul dataset di validation con soglia step 3

```
total_profit_val  =  (table_val[1,1]*costmat[1,1]+table_val[1,2]*costmat[1,2]+tabl
e_val[2,1]*costmat[2,1]+table_val[2,2]*costmat[2,2])
total_profit_val
```

```
## [1] 7532
```

```
mean_profit_val   =  total_profit_val/ nrow(validation)
mean_profit_val
```

```
## [1] 8.378198
```

STEP 4: Score su nuovi dati

Dataset di score senza colonna del target

```
score1 <- score
score1$Churn <- NULL
```

Applichiamo la soglia dello step 3 al dataset di score

```
score1$prob = predict(rfTune, score1, "prob")
prob = score1$prob
probc=prob[,2] #prob di essere classificato come churn
# applico la regola decisionale utilizzando la soglia dello step 3
score1$pred1=ifelse(probc>soglia, "True","False")
head(score1$pred1)
```

```
## [1] "False" "False" "False" "False" "False" "False"
```

Incrociamo i risultati ottenuti sullo score senza target con lo score da cui siamo partite

```
table_mod <- table(score$Churn,score1$pred1)
table_mod
```

```
##
##         False True
##   False    85    1
##   True      2   13
```

```
table_mod/dim(score)[1]
```

```
##
##              False       True
##   False 0.84158416 0.00990099
##   True  0.01980198 0.12871287
```

Profitto totale e profitto medio ottenuti sul dataset di score

```
total_profit_mod  =  (table_mod[1,1]*costmat[1,1]+table_mod[1,2]*costmat[1,2]+tabl
e_mod[2,1]*costmat[2,1]+table_mod[2,2]*costmat[2,2])
total_profit_mod
```

```
## [1] 753
```

```
mean_profit_mod   =  total_profit_mod/ nrow(score)
mean_profit_mod
```

```
## [1] 7.455446
```

Decidiamo di valutare anche le performance classificative sulle unità con la decision ottimale dei costi/profitti. Vediamo la tabella di decisione del validation:

```
table_val
```

```
##        pred_validation
##         False True
##   False   769    0
##   True     48   82
```

```
# calcoliamo l'accuracy sul dataset di validation
```

```
acc <- (table_val[1,1]+table_val[2,2]) / (table_val[1,1]+table_val[2,2]+table_val[
1,2]+table_val[2,1])
acc
```

```
## [1] 0.9466073
```

COMMENTO FINALE:

Il modello che abbiamo scelto risulta avere un buon profitto medio atteso e ottime performance classificative.