Generazione di modelli per predire l'interesse dei clienti per l'acquisto di una polizza

Francesco Gozzoli

29/10/2021

INTRODUZIONE

Il dataset contiene informazioni sull'interesse di clienti di una compagnia di assicurazioni per l'acquisto di polizze auto. Inizialmente le colonne che compongono il dataset si trovano nella situazione mostrata dalla funzione`summary`.

```
insurance <- read.csv("../insurance.csv")
summary(insurance)
```

```
##        id            Gender               Age        Driving_License
##  Min.   :     1   Length:381109      Min.   :20.00   Min.   :0.0000
##  1st Qu.: 95278   Class :character   1st Qu.:25.00   1st Qu.:1.0000
##  Median :190555   Mode  :character   Median :36.00   Median :1.0000
##  Mean   :190555                      Mean   :38.82   Mean   :0.9979
##  3rd Qu.:285832                      3rd Qu.:49.00   3rd Qu.:1.0000
##  Max.   :381109                      Max.   :85.00   Max.   :1.0000
##   Region_Code    Previously_Insured Vehicle_Age        Vehicle_Damage
##  Min.   : 0.00   Min.   :0.0000     Length:381109      Length:381109
##  1st Qu.:15.00   1st Qu.:0.0000     Class :character   Class :character
##  Median :28.00   Median :0.0000     Mode  :character   Mode  :character
##  Mean   :26.39   Mean   :0.4582
##  3rd Qu.:35.00   3rd Qu.:1.0000
##  Max.   :52.00   Max.   :1.0000
##  Annual_Premium   Policy_Sales_Channel    Vintage         Response
##  Min.   :  2630   Min.   :  1          Min.   : 10.0   Min.   :0.0000
##  1st Qu.: 24405   1st Qu.: 29          1st Qu.: 82.0   1st Qu.:0.0000
##  Median : 31669   Median :133          Median :154.0   Median :0.0000
##  Mean   : 30564   Mean   :112          Mean   :154.3   Mean   :0.1226
##  3rd Qu.: 39400   3rd Qu.:152          3rd Qu.:227.0   3rd Qu.:0.0000
##  Max.   :540165   Max.   :163          Max.   :299.0   Max.   :1.0000
```

PRE-PROCESSING

Per poter essere utilizzate efficientemente, alcune colonne hanno bisogno di essere manipolate. In particolare:

- La colonna `Gender`, viene trasformata nella colonna `Male`, che ammette valori binari `0` e `1` dove `1` rappresenta il sesso maschile e `0` quello femminile.

  ```
  insurance[which(insurance$Gender == "Male"),]$Gender = 1
  insurance[which(insurance$Gender == "Female"),]$Gender = 1
  colnames(insurance)[2] = "Male"
  insurance$Male <- as.factor(insurance$Male)
  ```

- La colonna `Vehicle_Age` viene convertita da categorica ordinata a numerica. I valori utilizzati sono:
  - -1 per le auto immatricolate da meno di un anno
  - 0 per le auto immatricolate tra gli 1 e i 2 anni precedenti
  - 1 per le auto immatricolate da più di 2 anni

```
insurance[which(insurance$Vehicle_Age == "< 1 Year"),]$Vehicle_Age <- -1
insurance[which(insurance$Vehicle_Age == "1-2 Year"),]$Vehicle_Age <- 0
insurance[which(insurance$Vehicle_Age == "> 2 Years"),]$Vehicle_Age <- 1
insurance$Vehicle_Age <- as.numeric(insurance$Vehicle_Age)
```

- La colonna `Vehicle_Damage` viene convertita da categorica nominale a factor. I valori utilizzati sono:
  - 0 per le auto che non sono mai state coinvolte in incidenti
  - 1 per le auto che sono state coinvolte in incidenti

```
insurance[which(insurance$Vehicle_Damage == "Yes"),]$Vehicle_Damage <- 1
insurance[which(insurance$Vehicle_Damage == "No"),]$Vehicle_Damage <- 0
insurance$Vehicle_Damage <- as.factor(insurance$Vehicle_Damage)
```

Le colonne categoriche e binarie restanti vengono convertite in factor

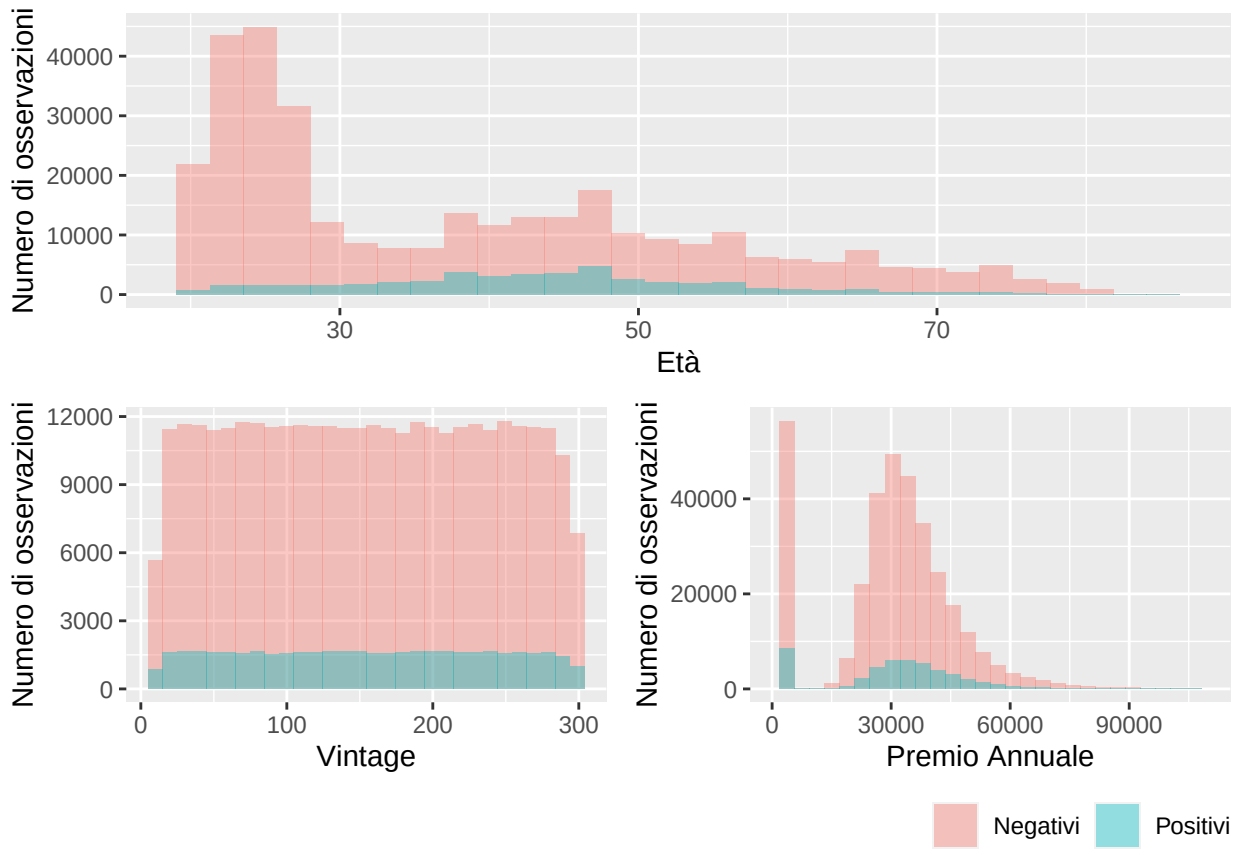
```
insurance$Driving_License <- as.factor(insurance$Driving_License)
insurance$Region_Code <- as.factor(insurance$Region_Code)
insurance$Previously_Insured <- as.factor(insurance$Previously_Insured)
insurance$Policy_Sales_Channel <- as.factor(insurance$Policy_Sales_Channel)
insurance$Response <- as.factor(insurance$Response)
```

STATISTICHE DESCRITTIVE

Attraverso la funzione ggplot si generano i grafici che mostrano la distribuzione di frequenza delle variabili numeriche `Age`, `Vintage`, `Annual_Premium`

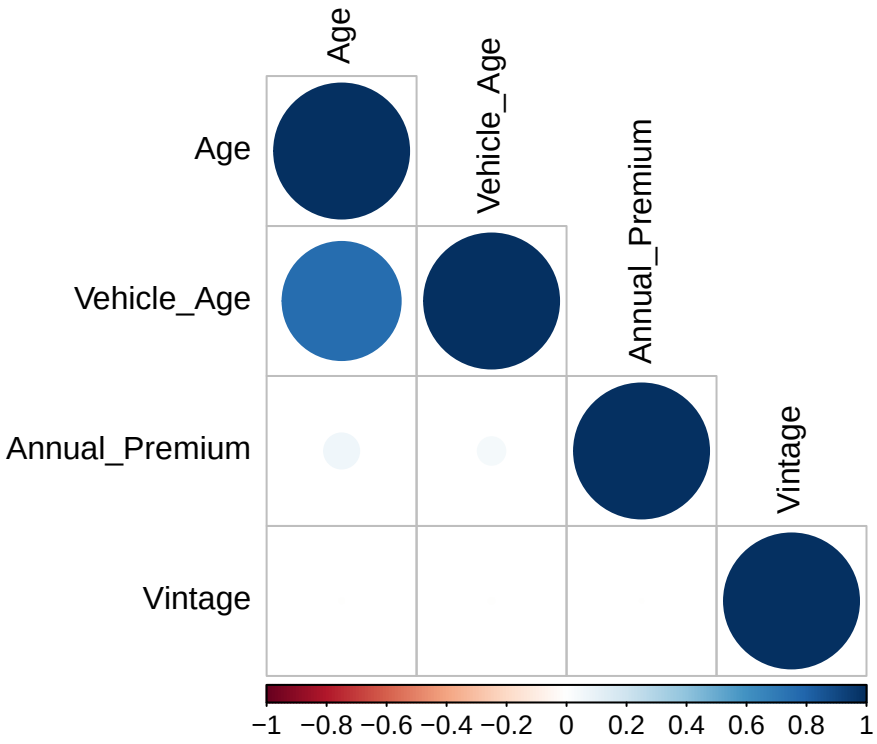
```
library(ggplot2)

age_plot <- ggplot(insurance,
                aes(x=Age, fill=Response))+
    geom_histogram(alpha=0.4,position="identity")+
    xlab("Età")+
    scale_y_continuous("Numero di osservazioni") + guides(fill=guide_legend(title=NULL)) +
  scale_fill_discrete(labels=c("Negativi","Positivi")) +
  theme(legend.position = c(1,1),legend.justification=c(1,1))

#Frequenze Vintage
vintage_plot <- ggplot(insurance,
             aes( x=Vintage, fill=Response))+
  geom_histogram(alpha=0.4,position="identity")+
  xlab("Vintage")+
  scale_y_continuous("Numero di osservazioni")

#Frequenze annual_premium
premium_plot <- ggplot( insurance,
             aes( x=Annual_Premium, fill=Response))+
  geom_histogram(alpha=0.4,position="identity")+
  xlab("Premio Annuale")+
  scale_y_continuous("Numero di osservazioni") +
  scale_x_continuous(limits = c(0, 110000))
```

```
library(ggpubr)
ggarrange(age_plot, ggarrange(vintage_plot, premium_plot, ncol = 2, legend = "none"),
          nrow = 2, common.legend = T, legend = "bottom")
```



Nonostante il numero non elevato di features presenti nel dataset possiamo cercare la correlazione presente tra esse. Dalla correlation matrix si può notare che le features `Vehicle_Age` e `Age` sono piuttosto correlate tra loro. Nonostante questo ho deciso di mantenerle entrambe poichè il vantaggio dal punto di vista computazionale è minimo, evitando cosi perdita di informazioni

```
library(corrplot)
correlationMatrix <- stats::cor(insurance[c(3, 7, 9, 11)])
corrplot(correlationMatrix, method="circle", type="lower", tl.col="black")
```

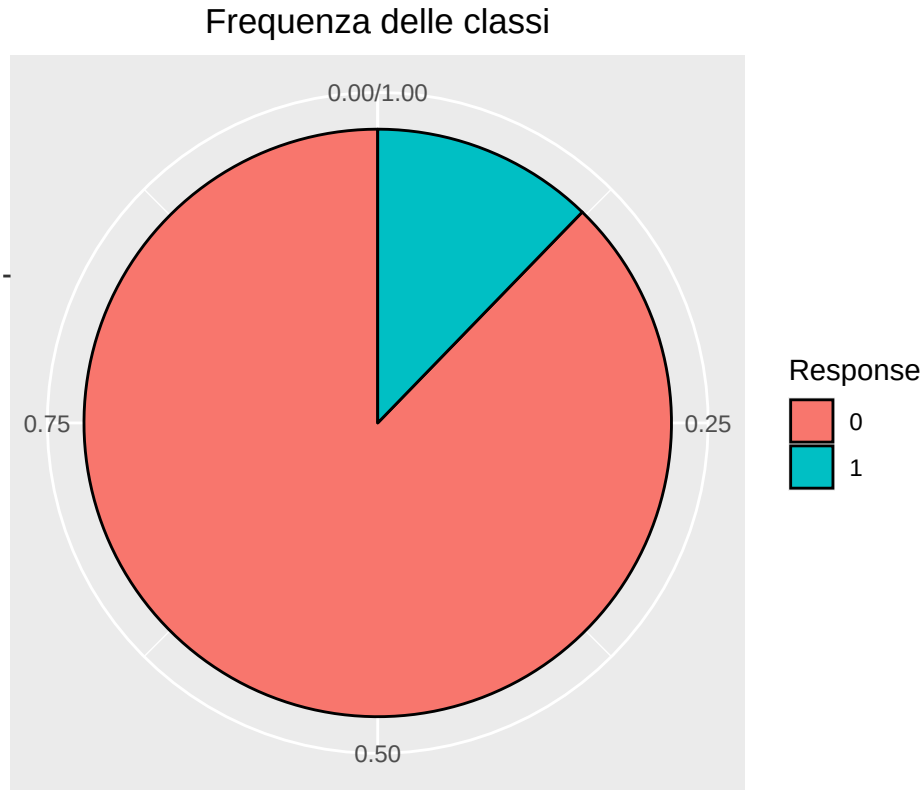


Uno degli aspetti più critici del dataset in analisi è lo sbilanciamento tra le classi, come si evince dal seguente piechart. Infatti le percentuali evidenziano uno sbilanciamento di 88% a favore della classe dei negativi contro il 12% della classe dei positivi

```
freq <- as.data.frame(table(insurance$Response))
colnames(freq)[1] = "Response"
freq$perc <- prop.table(freq$Freq)
print(freq)
```

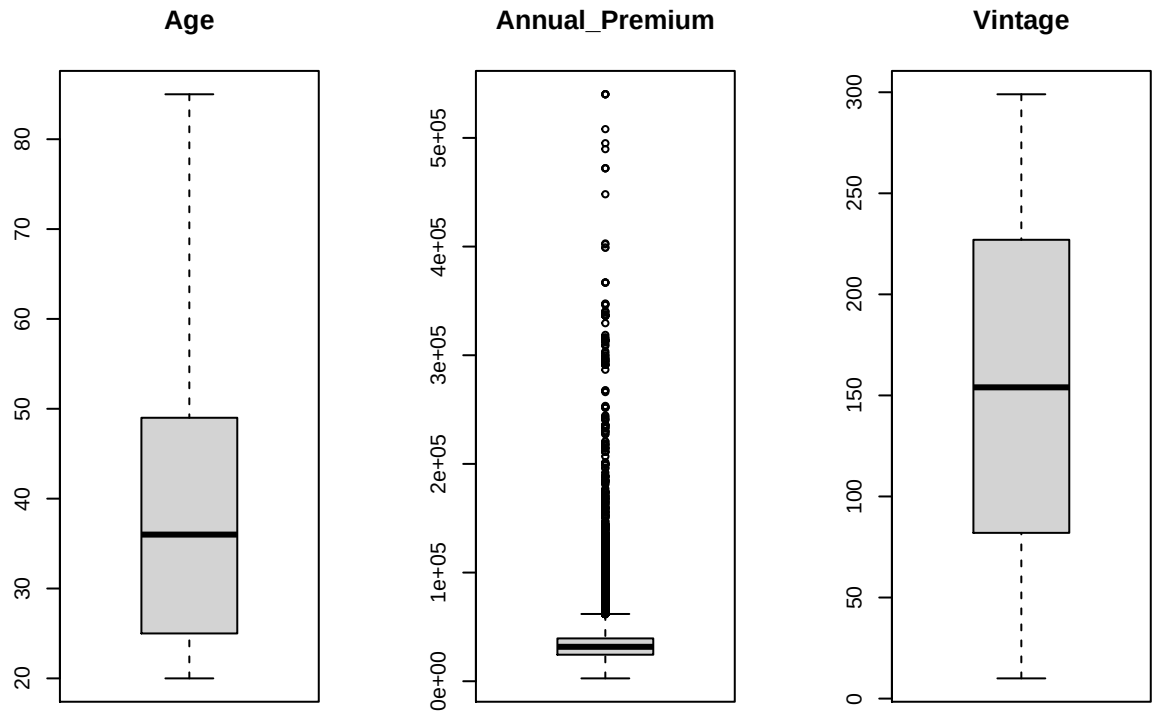
```
##   Response   Freq      perc
## 1        0 334399 0.8774366
## 2        1  46710 0.1225634
```

```
ggplot(freq, aes(x = "", y = perc, fill = Response)) +
  geom_col(color = "black") +
  coord_polar(theta = "y") +
  xlab("") +
  ylab("") +
  ggtitle("Frequenza delle classi") +
  theme(plot.title = element_text(hjust = 0.5))
```



Per rappresentare graficamente la distribuzione dei valori delle 3 features numeriche `Age`, `Annual_Premium`, `Vintage`e controllare la presenza di outliers si stampano i relativi boxplot

```
par(mfrow=c(1,3))
boxplot(insurance$Age, main="Age")
boxplot(insurance$Annual_Premium, main="Annual_Premium")
boxplot(insurance$Vintage, main="Vintage")
```




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

MODELLI DI CLASSIFICAZIONE

SEPARAZIONE STRATIFICATA

Per addestrare e validare i modelli che verranno realizzati si divide il dataset originale nei due dataset di training e testing nella classica proporzione percentuale di 70-30. Il criterio di separazione scelto è quello stratificato perchè consente di mantenere la proporzione tra le istanze della classe di maggioranza e quella di minoranza all'interno dei due dataset. E' stata fatta questa scelta perchè il dataset originale è molto sbilanciato, ed in questo modo evitiamo di creare dei dataset ancora più sbilanciati

```
library(splitstackshape) #per suddivisione test/train stratificata
set.seed(1)
train <- as.data.frame(stratified(insurance, c('Response'), 0.7))
test <- insurance[which(!(insurance$id %in% train$id)),]
insurance <- insurance[, -1]
train <- train[, -1]
test <- test[,-1]
```

Nonostante lo sbilanciamento si prova a generare un primo modello, in particolare viene generato un albero di decisione. Per fare ciò ci si può avvalere della funzione `rpart`, che genera un albero implementando l'algoritmo CART ed utilizzando quindi l'indice di Gini come criterio per lo splitting [1]

```
library(rpart) #per alberi di decisione
tree <- rpart(Response~., train, method = "class", control = rpart.control(cp = 0.00001))
```

[1] https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

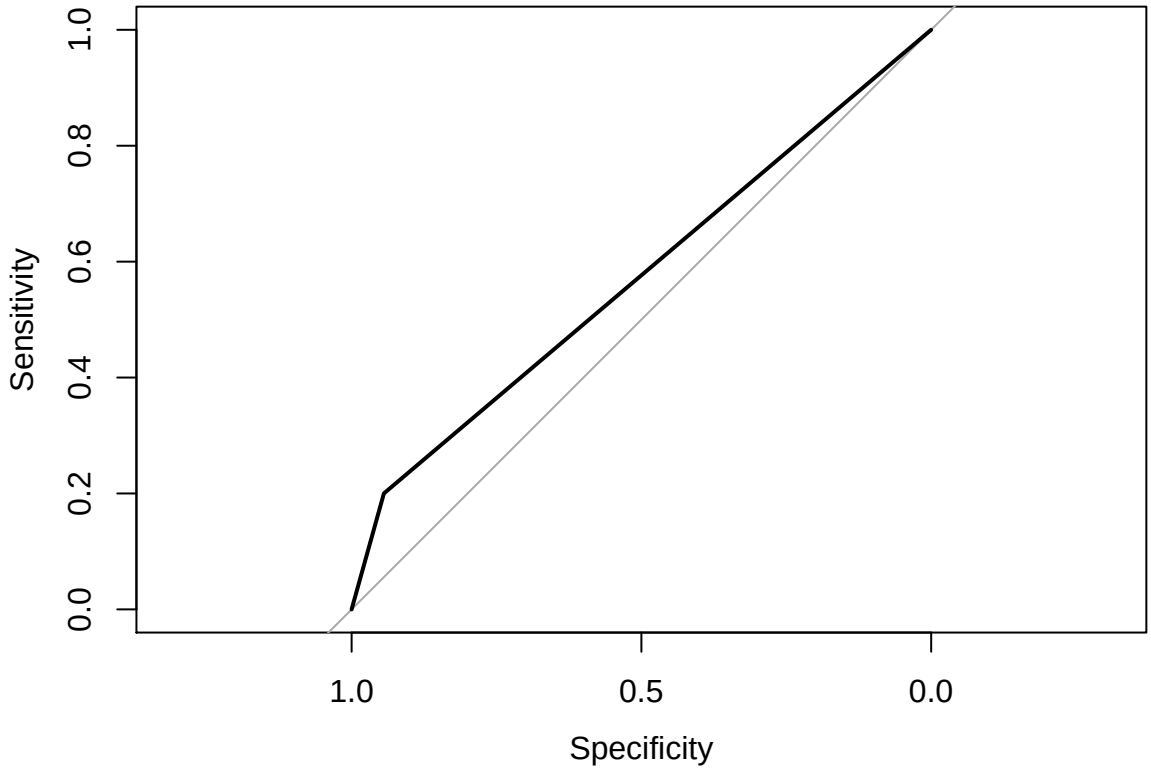
```r
prediction <- predict(tree, test, type = "class")
library(caret) #per confusionmatrix
confusionMatrix(as.factor(prediction),as.factor(test$Response))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 94733 11209
##          1  5587  2804
##
##                Accuracy : 0.8531
##                  95% CI : (0.851, 0.8551)
##     No Information Rate : 0.8774
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1745
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9443
##             Specificity : 0.2001
##          Pos Pred Value : 0.8942
##          Neg Pred Value : 0.3342
##              Prevalence : 0.8774
##          Detection Rate : 0.8286
##    Detection Prevalence : 0.9266
##       Balanced Accuracy : 0.5722
##
##        'Positive' Class : 0
##
```

```r
auc = roc(test$Response, as.numeric(prediction))
print(auc)
```

```
##
## Call:
## roc.default(response = test$Response, predictor = as.numeric(prediction))
##
## Data: as.numeric(prediction) in 100320 controls (test$Response 0) < 14013 cases (test$Response 1).
## Area under the curve: 0.5722
```

```r
plot(auc)
```



Come si può notare dalla curva ROC e dalla ConfusionMatrix la classificazione non è per niente soddisfacente poichè nonostante riesca a classificare le istanze della classe di maggioranza fatica a classificare correttamente le istanze della classe di minoranza Questo è un problema perchè oltre ad essere un punto debole del modello, l'obiettivo della classificazione è quello di massimizzare la corretta identificazione di potenziali clienti interessati piuttosto che di quelli non interessati. In generale quindi si preferisce avere un maggior tasso di falsi positivi piuttosto che di falsi negativi

RIBILANCIAMENTO

Alla luce delle prestazioni mostrate da questo primo modello è bene procedere ad un ribilanciamento delle classi. Per fare ciò mi sono avvalso di due tecniche: * Tecnica di sottocampionamento casuale che consiste nel rimuovere casualmente istanze della classe di maggioranza fino ad avere un bilanciamento di circa 50-50 * Sovracampionamento effettuato attraverso SMOTE, che genera istanze sintetiche della classe di minoranza sfruttando un algoritmo di k-nearest neighbors (K = 5 nel mio caso)

```
set.seed(1)
rem <- ubUnder(train[,1:11], train$Response, perc=14, method = "percUnder")
casual.balance.train <- data.frame(rem$X, rem$Y)
casual.balance.train$rem.Y <- NULL
names(casual.balance.train)[11] <- "Response"
#casual.balance.test <- c(test, train[rem$id.rm,])
rm(rem)
table(casual.balance.train$Response)
```

```
##
##     0     1
## 32771 32697
```