Week2

Matteo Gambera

2 aprile 2020

Caret package

- Data splitting
- Taining testing fun
- moddel comparison

Ci sono molti algoritmi

- regression
- random forests
- e molti altri

```
library(caret); library(kernlab); data(spam)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'kernlab'

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

```
## p = 0.75 significa il 75% dei dati per il training
inTrain <- createDataPartition(y=spam$type, p = 0.75 , list = FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
dim(training)
```

```
## [1] 3451   58
```

```
dim(testing)
```

```
## [1] 1150   58
```

```r
1 - dim(testing)[1]/dim(spam)[1]
```

```
## [1] 0.7500543
```

Ora creiamo il modello

```r
set.seed(32343)
modelfit <- train(type ~ . , data = training , method = "glm");
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modelfit
```

```
## Generalized Linear Model
##
## 3451 samples
##   57 predictor
##    2 classes: 'nonspam', 'spam'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9265194  0.8455301
```

Risultato:

- 3451 sono i dati utilizzati
- 57 sono le colonne utilizzate
- 2 le variabili predette
- il miglior modello è basato su resempling e bootstrapped con 25 ripetizioni

Adesso dobbiamo fittare il modello Prendo il modello finale dall'oggetto modelfit

```
modelfit$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##       (Intercept)              make           address               all
##         -1.592693         -0.461605         -0.120538          0.247212
##             num3d               our              over            remove
##          3.932846          0.593971          0.663696          3.110965
##          internet             order              mail           receive
##          0.655962          0.727837          0.072279         -0.346804
##              will            people            report         addresses
##         -0.062553         -0.209626          0.192395          0.743952
##              free          business             email               you
##          0.854673          0.941736          0.143426          0.067263
##            credit              your              font            num000
##          2.769801          0.247956          0.162221          2.305611
```

```
##            money             hp            hpl          george
##         0.611831      -2.808856      -2.921037       -7.544061
##           num650            lab           labs          telnet
##         0.421412      -3.815547       0.411672       -3.535669
##           num857           data         num415           num85
##         1.107277      -0.646973       0.306538       -3.405581
##       technology        num1999          parts              pm
##         1.209840       0.148689      -0.549325       -0.951794
##           direct             cs        meeting        original
##        -0.390849     -46.501909      -2.591756       -1.219062
##          project             re            edu           table
##        -1.805664      -0.701146      -1.362112       -4.617167
##       conference   charSemicolon charRoundbracket charSquarebracket
##        -4.243536      -1.202733      -0.022225       -1.290358
##    charExclamation     charDollar       charHash      capitalAve
##         0.550940       5.733907       2.345252       -0.002841
##      capitalLong   capitalTotal
##         0.007114       0.001051
##
## Degrees of Freedom: 3450 Total (i.e. Null);  3393 Residual
## Null Deviance:       4628
## Residual Deviance: 1251  AIC: 1367
```

Ora dobbiamo fare un predict del nostro set di dati test

```r
prediction <- predict(modelfit , newdata = testing)
head(prediction, n= 20)
```

```
##  [1] spam    spam    spam    spam    spam    spam    spam    spam    spam
## [10] spam    spam    nonspam spam    spam    spam    spam    spam    spam
## [19] spam    spam
## Levels: nonspam spam
```

Ora vado a confrontare con i risultati reali

```r
confusionMatrix(prediction, testing$type)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     649   48
##    spam         48  405
##
##                Accuracy : 0.9165
##                  95% CI : (0.899, 0.9319)
##     No Information Rate : 0.6061
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8252
##
##  Mcnemar's Test P-Value : 1
```

```
##
##             Sensitivity : 0.9311
##             Specificity : 0.8940
##          Pos Pred Value : 0.9311
##          Neg Pred Value : 0.8940
##              Prevalence : 0.6061
##          Detection Rate : 0.5643
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.9126
##
##        'Positive' Class : nonspam
##
```

Data slicing

per creare data trainging and testing

```
## p = 0.75 significa il 75% dei dati per il training
inTrain <- createDataPartition(y=spam$type, p = 0.75 , list = FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
```

K-fold

```
set.seed(1)
## k è il numero di folds, list crea un indicizzazione dei fold
## return train = true ritorno il training altrimenti il test
folds <- createFolds(y = spam$type, k = 10 ,
                     list = TRUE, returnTrain = TRUE )
## guardo le dimensioni dei fold
sapply(folds, length)
```

```
## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
##   4140   4141   4140   4141   4141   4142   4141   4141   4141   4141
```

Guardo il primo fold

```
folds[[1]][1:10]
```

```
##  [1]  1  2  5  6  7  8  9 11 12 13
```

Resampling

```
set.seed(1)
## times è il numero di resample, list crea un indicizzazione dei resample
## return train = true ritorno il training altrimenti il test
folds <- createResample(y = spam$type, times = 10 ,
```

```r
                     list = TRUE)
## guardo le dimensioni dei fold
sapply(folds, length)
```

```
## Resample01 Resample02 Resample03 Resample04 Resample05 Resample06 Resample07
##       4601       4601       4601       4601       4601       4601       4601
## Resample08 Resample09 Resample10
##       4601       4601       4601
```

```r
folds[[1]][1:10]
```

```
##  [1]  4  6  7  7 14 15 15 15 15 16
```

Time Slices

for forecasting, in cui si vuole avere valori continui nel tempo

```r
set.seed(1)
## Creo il time vector
tme <- 1:1000
## Divido in finestre da 20 e voglio predite i 10 valori dopo
folds <- createTimeSlices(y = tme, initialWindow = 20,
                          horizon = 10)
names(folds)
```

```
## [1] "train" "test"
```

controlliamo come sono strutturati i sample

```r
folds$train[1]
```

```
## $Training020
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
folds$train[2]
```

```
## $Training021
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
```

```r
folds$test[1]
```

```
## $Testing020
##  [1] 21 22 23 24 25 26 27 28 29 30
```

```r
folds$test[2]
```

```
## $Testing021
##  [1] 22 23 24 25 26 27 28 29 30 31
```

Training options

metric options

- RMSE = root mean square error
- Rsquared = R^2 from regression models
- Accuracy dice quanti ne ha predetti giusti
- kappa a measure of concordance

e un sacco di altre cose qui link

Plotting predictor

```
library(ISLR); library(ggplot2); library(caret)
data(Wage)
?Wage
```

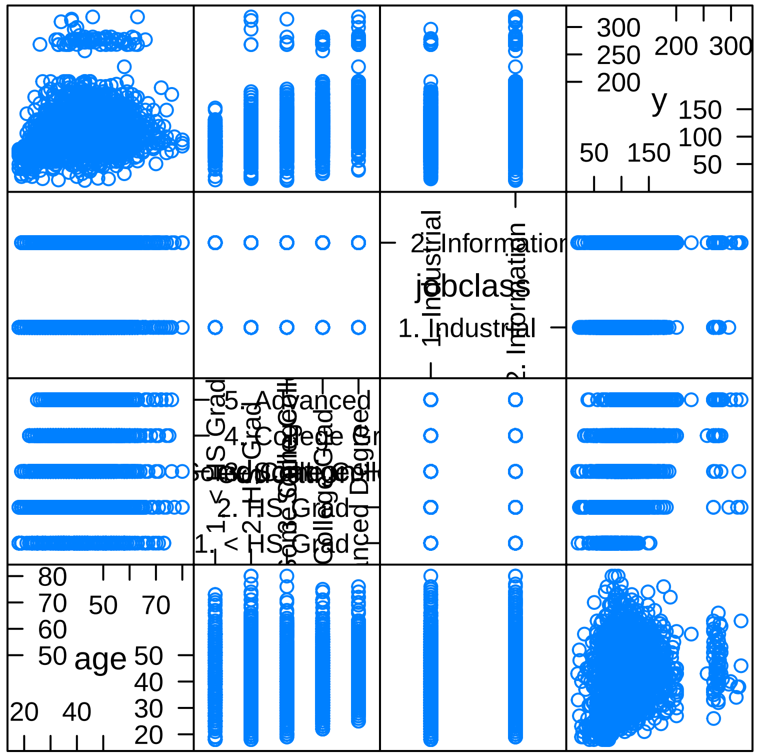
```
inTrain <- createDataPartition(y=Wage$wage, p = 0.7 , list = FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   11
```

```
## [1] 898  11
```

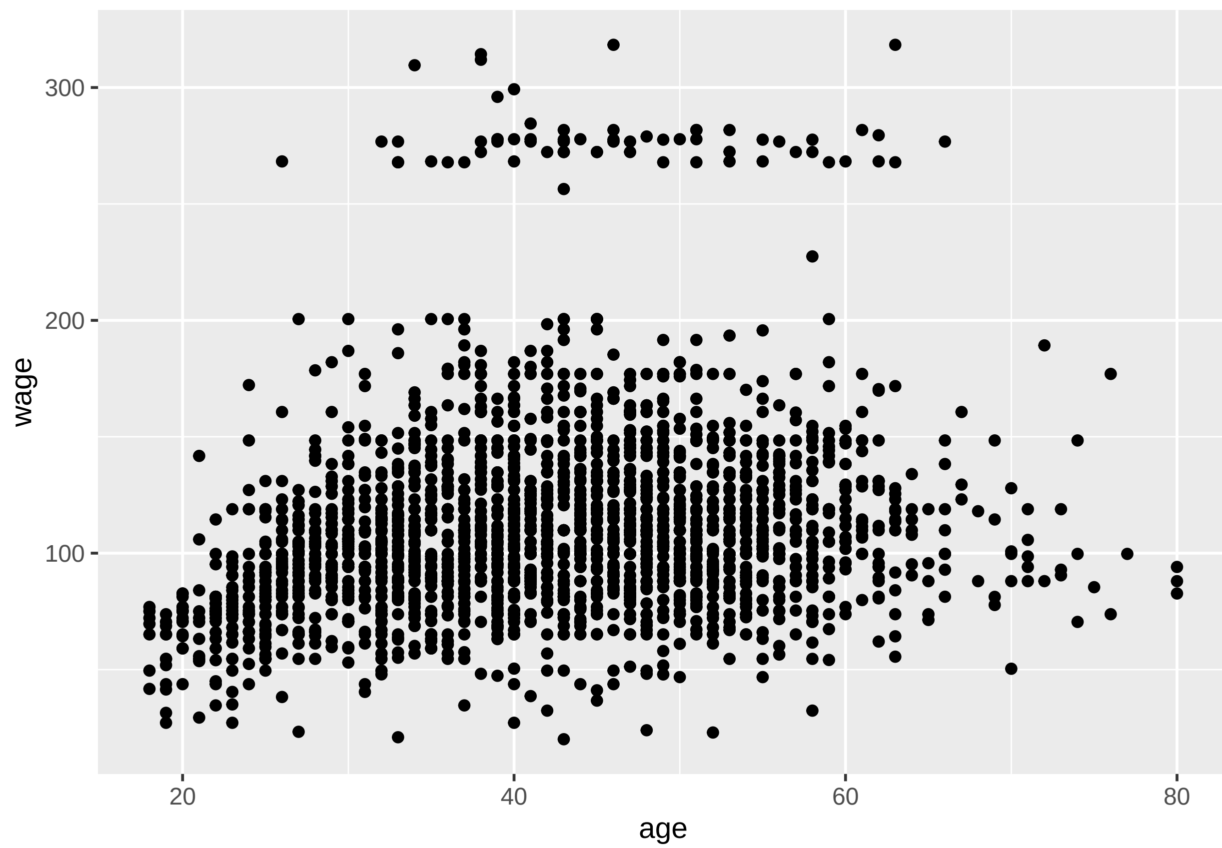
Confuso per quest dataset

```
featurePlot(x = training[,c("age", "education", "jobclass")],
            y = training$wage,
            plot = "pairs")
```

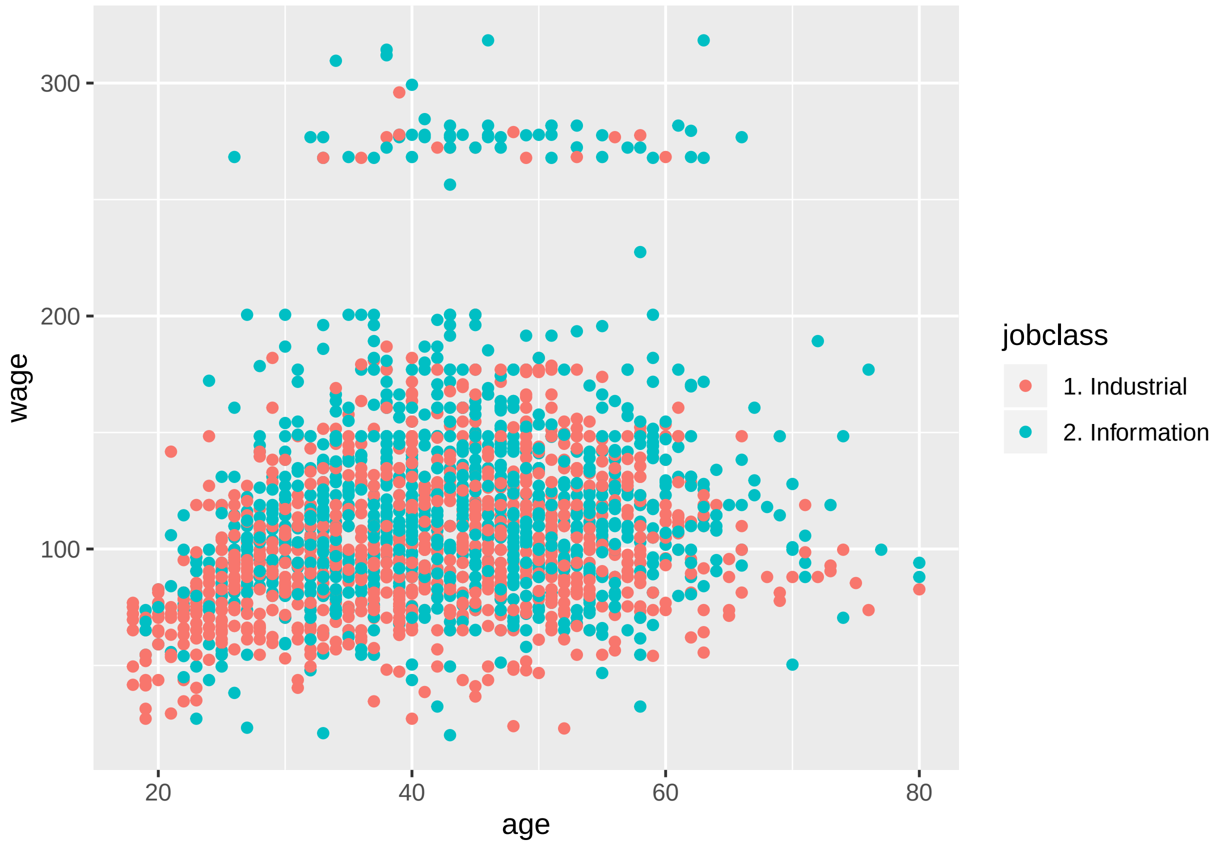


Scatter Plot Matrix

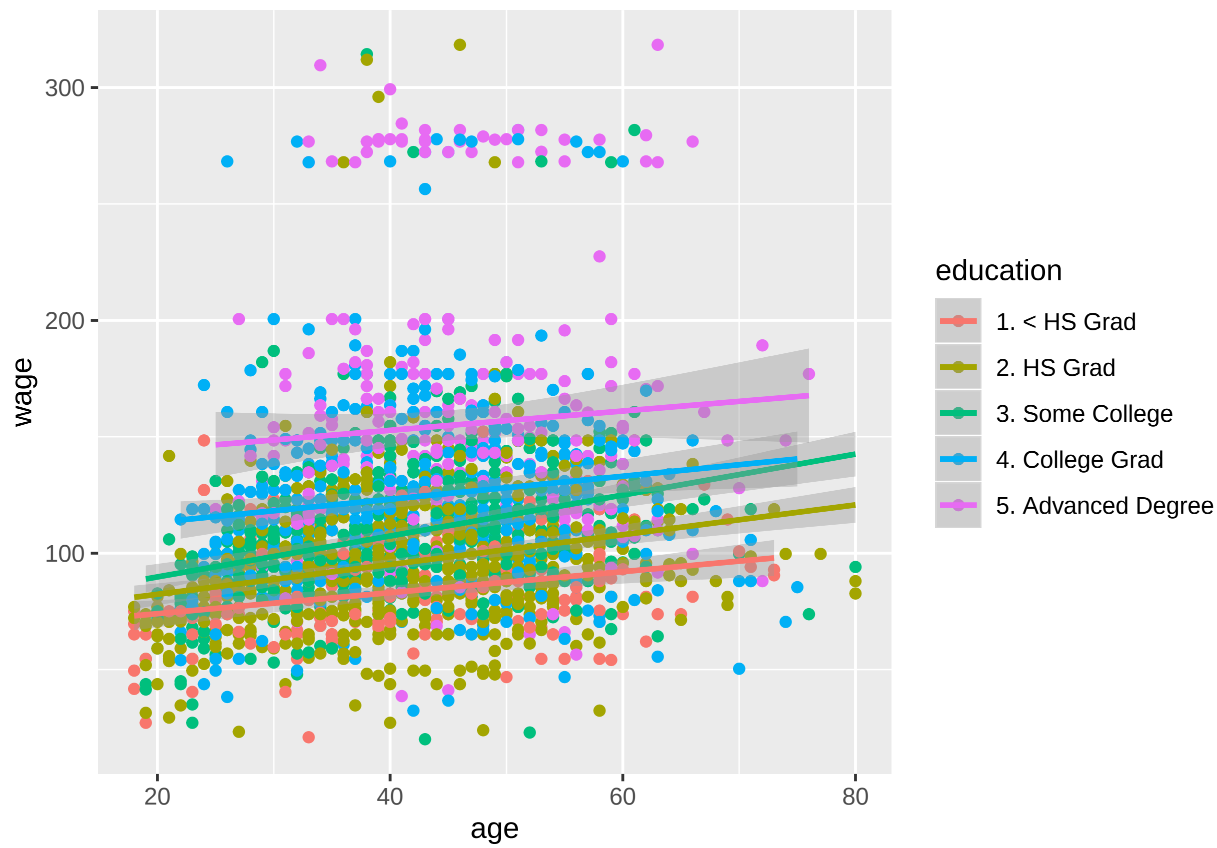
```r
qplot(age,wage, data = training)
```




```r
qplot(age, wage,colour = jobclass, data = training)
```




```r
gg <- qplot(age,wage,colour = education, data = training)
gg + geom_smooth(method = "lm" , formula = y ~ x)
```




```
library(Hmisc);
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

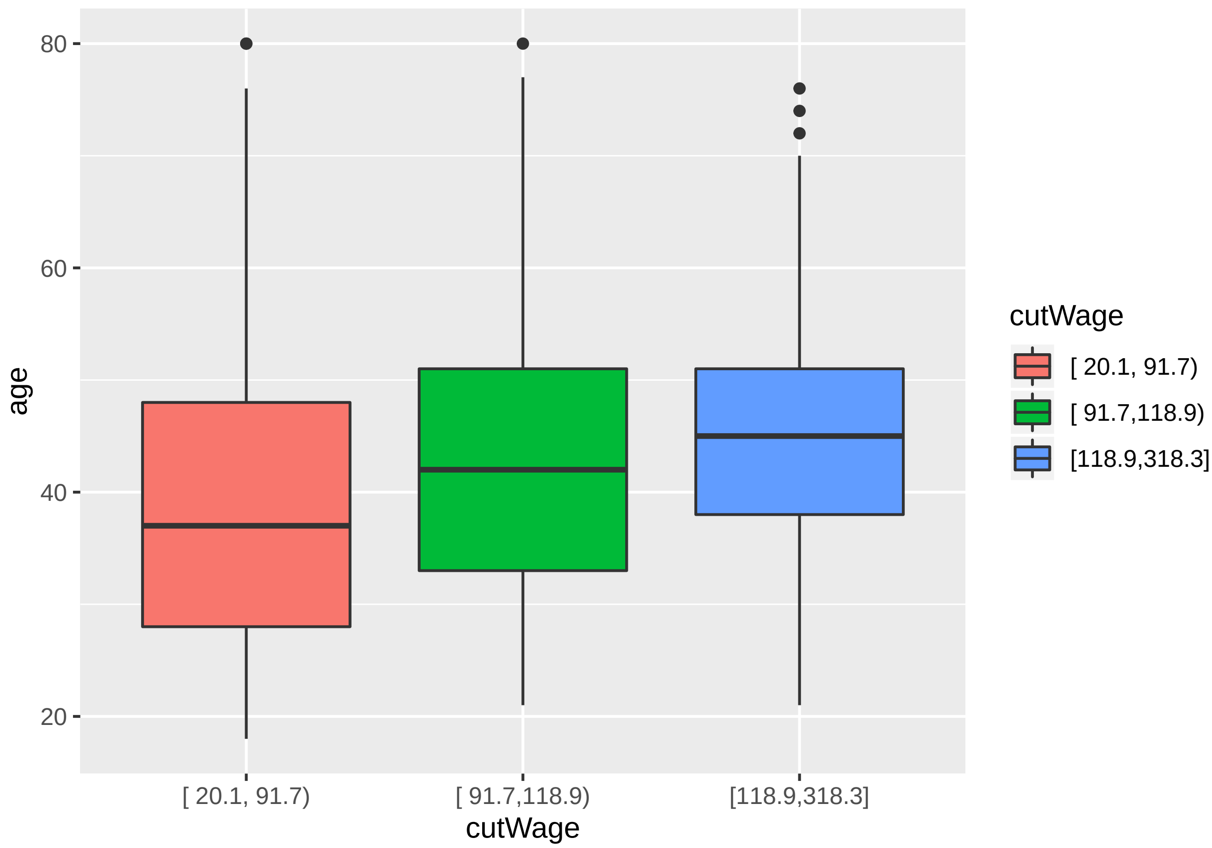
```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
## g sceglie in quante parti dividere wage
cutWage <- cut2(training$wage, g = 3)
table(cutWage)
```

```
## cutWage
## [ 20.1, 91.7) [ 91.7,118.9) [118.9,318.3]
##           707           718           677
```

```
p1 <- qplot(cutWage, age, data = training, fill = cutWage,
            geom = c("boxplot"))
p1
```




```
p2 <- qplot(cutWage, age, data = training, fill = cutWage,
            geom = c("boxplot", "jitter"))
#grid.arrange(p1,p2,ncol=2)
```

```
t1 <- table(cutWage, training$jobclass)
t1
```

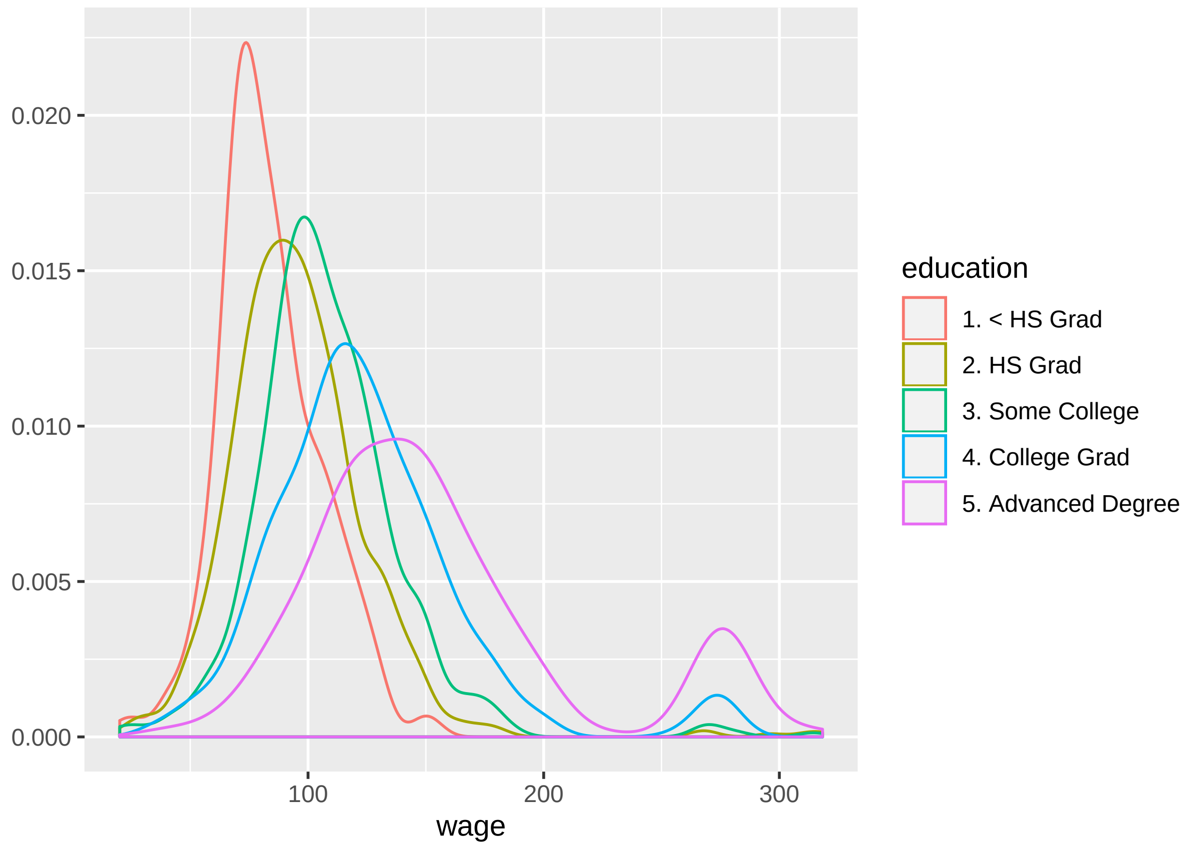
```
##
## cutWage         1. Industrial 2. Information
##   [ 20.1, 91.7)           451            256
##   [ 91.7,118.9)           368            350
##   [118.9,318.3]           269            408
```

se voglio la tabella con le proporzioni

```r
## 1 per le righe, 2 per le colonne
prop.table(t1,1)
```

```
##
## cutWage         1. Industrial 2. Information
##   [ 20.1, 91.7)     0.6379066      0.3620934
##   [ 91.7,118.9)     0.5125348      0.4874652
##   [118.9,318.3]     0.3973412      0.6026588
```

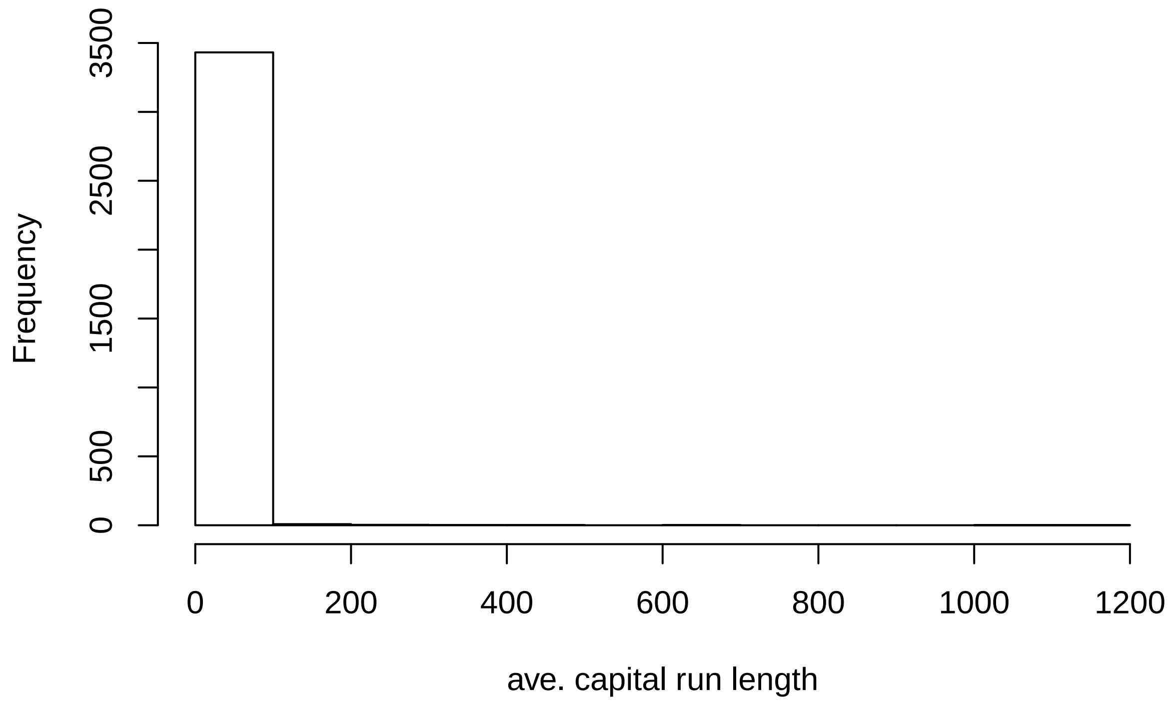
```r
qplot(wage, colour = education, data = training, geom = "density")
```



Preprocessing

center and scaling

```r
inTrain <- createDataPartition(y=spam$type, p = 0.75 , list = FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
hist(training$capitalAve , main = "", xlab = "ave. capital run length")
```



Ce ne sono veramente pochi con più di 100, difficili da vedere e analizzare, dobbiamo fare pre processing

```
mean(training$capitalAve)
```

```
## [1] 5.581866
```

```
sd(training$capitalAve)
```

```
## [1] 34.83702
```

La devstd è molto più grande della media, per non, non far capire un cazzo all'algoritmo ML, vanno prima standardizzati

```
traincapave <- training$capitalAve
traincapaves <- (traincapave - mean(traincapave))/sd(traincapave)
mean(traincapaves)
```

```
## [1] 1.157656e-17
```

```
sd(traincapaves)
```

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

Stessa cosa va fatta per i test

```r
testcapave <- testing$capitalAve
testcapaves <- (testcapave - mean(testcapave))/sd(testcapave)
mean(testcapaves)
```

```
## [1] -8.519024e-18
```

```r
sd(testcapaves)
```

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

C'è gia una funzione che si occupa di standardizzare

```r
## Gli passiamo tutto tranne il 58 che è l'outcome di cui
## ci stiamo preoccupando, centro ogni variabile e la scalo
preObj <- preProcess(training[,-58], method = c("center", "scale"))
trainCapAves <- predict(preObj, training[,-58])$capitalAve
mean(trainCapAves)
```

```
## [1] 1.157656e-17
```

```r
sd(trainCapAves)
```

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

Passiamo ora al test, prendo il valore calcolato con il preprocessing e lo applico al testset

```r
testCapAves <- predict(preObj, testing[,-58])$capitalAve
mean(testCapAves)
```

```
## [1] -0.04482993
```

```r
sd(testCapAves)
```

```
## [1] 0.5630029
```

Posso direttamnete passare il preprocess alla funzione train

```r
set.seed(32343)
modelFit <- train( type ~ . , data = training ,
                   preProcess = c("center", "scale"),
                   method = "glm")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modelFit
```

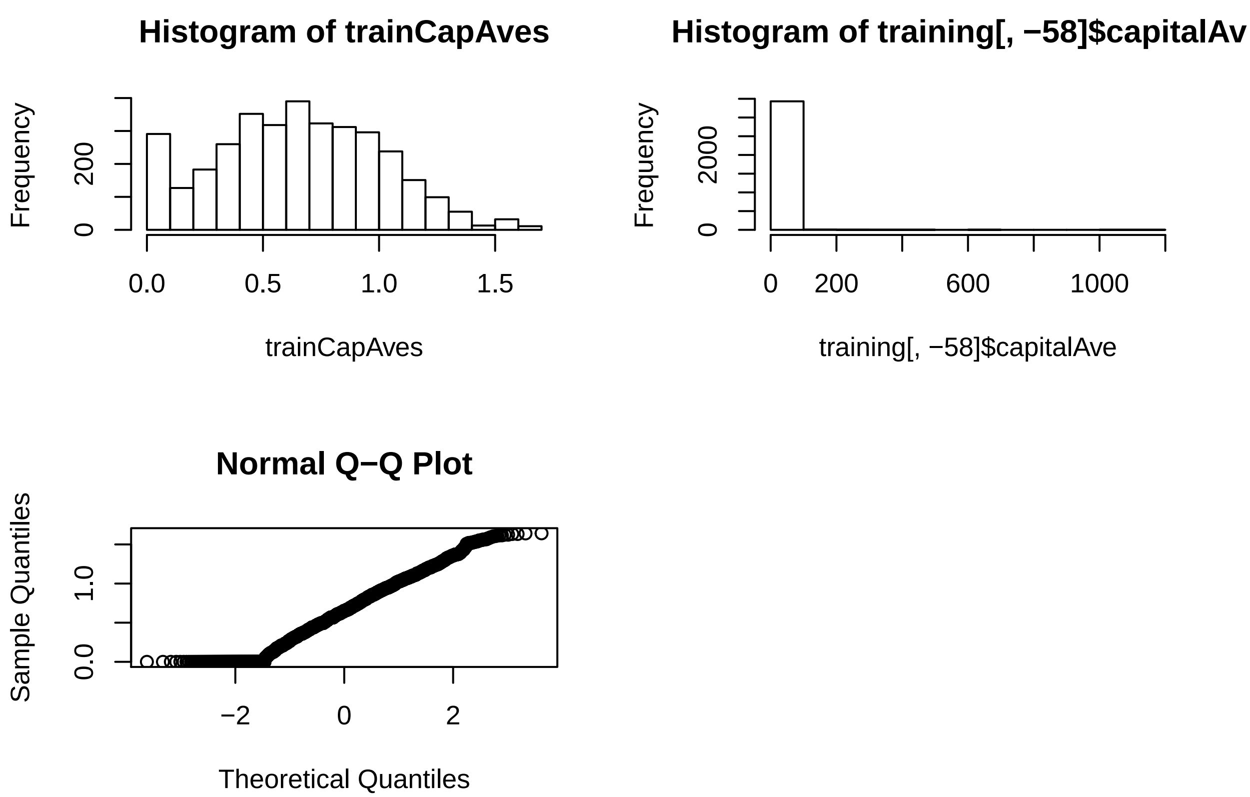
```
## Generalized Linear Model
##
## 3451 samples
```

```
##   57 predictor
##    2 classes: 'nonspam', 'spam'
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9210036  0.8340326
```

box-cox transforms

Prende dei dati continui e tenta di renderli come dei dati normali

```
preObj <- preProcess(training[,-58], method = c("BoxCox"))
trainCapAves <- predict(preObj, training[,-58])$capitalAve
par(mfrow = c(2,2)); hist(trainCapAves); hist(training[,-58]$capitalAve);qqnorm(trainCapAves)
```



non è una bella curva gaussiana, infatti all'inizio ho uno spike, con il normal Q-Q plot lo si può vedere allinizio ### Imputing data prediction algorithm spesso sbagliano quando ci sono dati mancanti

```
library(RANN)
set.seed(13343)
## Creo dei valori NA
training$capAve <- training$capitalAve
```

```
selectNa <- rbinom(dim(training)[1], size = 1, prob = 0.05)==1
training$capAve[selectNa] <- NA

## Impute and standardize
## k nearest neighbors trova le k (es 10) data vectors che più
## assomigliano ai missing values, fanno una media e sostituiscono il NA
preObj <- preProcess(training[,-58], method = "knnImpute")
capAve <- predict(preObj, training[,-58])$capAve

## standardizzo
capAveTruth <- training$capitalAve
capAveTruth <- (capAveTruth - mean(capAveTruth))/sd(capAveTruth)
quantile(capAve - capAveTruth)
```

```
##            0%           25%           50%           75%          100%
## -1.020914e+01 -1.053199e-02 -7.354207e-03 -6.634344e-04  4.487981e+00
```

Covariete creation

faccio una compressione delle informazioni contenute in una riga ad esempio se ho un testo scritto, vado ad estrapolare solo il numero di volte in cui appare "you" oppure altre cose

```
## per dare più peso alle differenze gli elevo al quadrato, molto utile per ML
spam$capitalAvesq <- spam$capitalAve^2
```

```
inTrain <- createDataPartition(y = Wage$wage, p = 0.7 , list= FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   11
```

```
## [1] 898  11
```

Covariate va applicato SOLO al training

```
table(training$jobclass)
```

```
##
##  1. Industrial 2. Information
##           1069           1033
```

Per ML è difficile comprendere delle differenze qualitative (tipo nomi: ind/inf) Quello che si fa è assegnarli una variabile quantitative

```
dummies <- dummyVars( wage ~ jobclass , data = training)
head(predict(dummies, newdata = training))
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                      1                        0
## 86582                       0                        1
## 161300                      1                        0
## 155159                      0                        1
## 376662                      0                        1
## 450601                      1                        0
```

nearZeroVar identifica le variabile che hanno poco variabilità e quindi che non sono dei buoni predittori

```r
nsv <- nearZeroVar(training, saveMetrics = TRUE)
nsv
```

```
##            freqRatio percentUnique zeroVar   nzv
## year        1.034384    0.33301618   FALSE FALSE
## age         1.116883    2.90199810   FALSE FALSE
## maritl      3.374713    0.23786870   FALSE FALSE
## race        8.146226    0.19029496   FALSE FALSE
## education   1.438525    0.23786870   FALSE FALSE
## region      0.000000    0.04757374    TRUE  TRUE
## jobclass    1.034850    0.09514748   FALSE FALSE
## health      2.480132    0.09514748   FALSE FALSE
## health_ins  2.199391    0.09514748   FALSE FALSE
## logwage     1.133333   19.12464320   FALSE FALSE
## wage        1.133333   19.12464320   FALSE FALSE
```

ad esempio region non ha variabilità

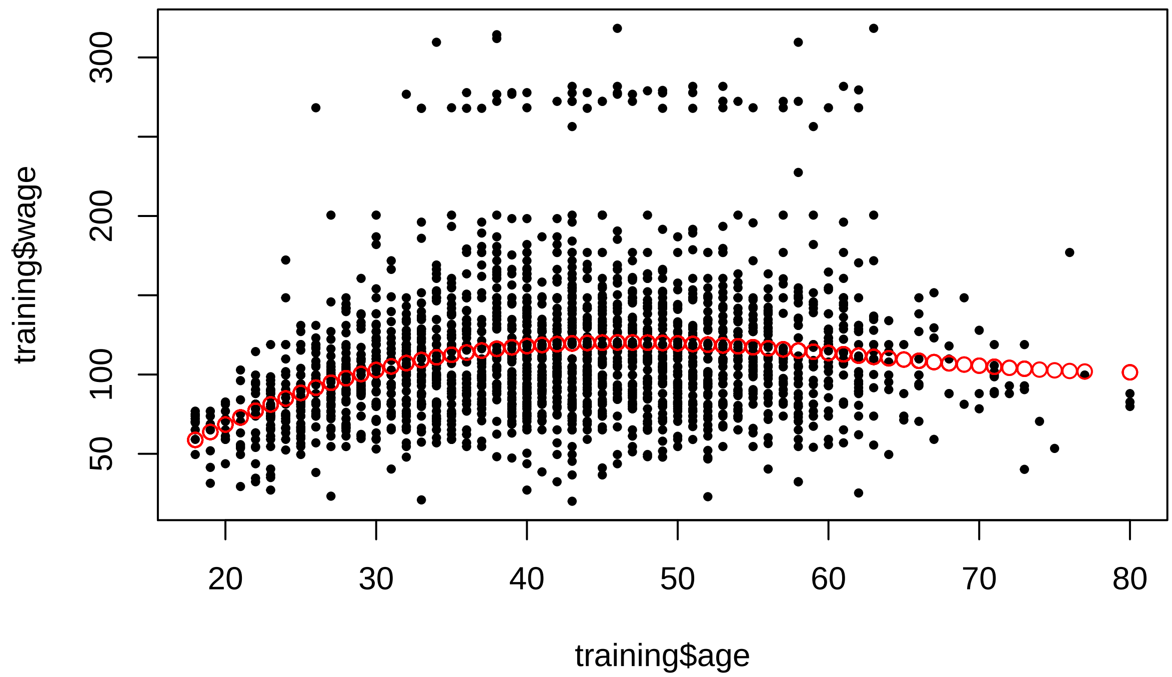
basis spline

```r
library(splines)
## vado a creare una nuova variabile, che contiene
## ancora age però scalata per facilitare la parte computazionale
## df=3 mi da age,age^2,age^3
bsBasis <- bs(training$age, df = 3)
head(bsBasis)
```

```
##              1          2           3
## [1,] 0.0000000 0.00000000 0.000000000
## [2,] 0.2368501 0.02537679 0.000906314
## [3,] 0.4163380 0.32117502 0.082587862
## [4,] 0.4308138 0.29109043 0.065560908
## [5,] 0.3063341 0.42415495 0.195763821
## [6,] 0.4241549 0.30633413 0.073747105
```

In questo modo posso avvere fit curvi

```r
lm1 <- lm(wage ~ bsBasis, data = training)
plot(training$age, training$wage , pch=19, cex = 0.5)
points(training$age, predict(lm1,newdata = training), col = "red")
```



Sulla parte del test vado a predirre dalla variabile bsbasis un nuovo set di dati in questo modo non sono legate a quelle del trainingset

```
head(predict(bsBasis, age = teasting$age))
```

```
##              1          2           3
## [1,] 0.0000000 0.00000000 0.000000000
## [2,] 0.2368501 0.02537679 0.000906314
## [3,] 0.4163380 0.32117502 0.082587862
## [4,] 0.4308138 0.29109043 0.065560908
## [5,] 0.3063341 0.42415495 0.195763821
## [6,] 0.4241549 0.30633413 0.073747105
```

Preprocessing with principal components Analysis

Qaundo ho variabili molto correlate tra di loro non ha senso inserirle tutte nel ML

correleted predictors

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y = spam$type, p = 0.75 , list = FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
```

```r
## calcolo la correlazione tra tutte le colonne, 58 è outcome
M <- abs(cor(training[,-58]))
## siccome non sono interessato alla correlazione tra se stesse le tolgo
diag(M) <- 0
## seleziono solo quelle con un certo valore
which(M > 0.8, arr.ind = T)
```

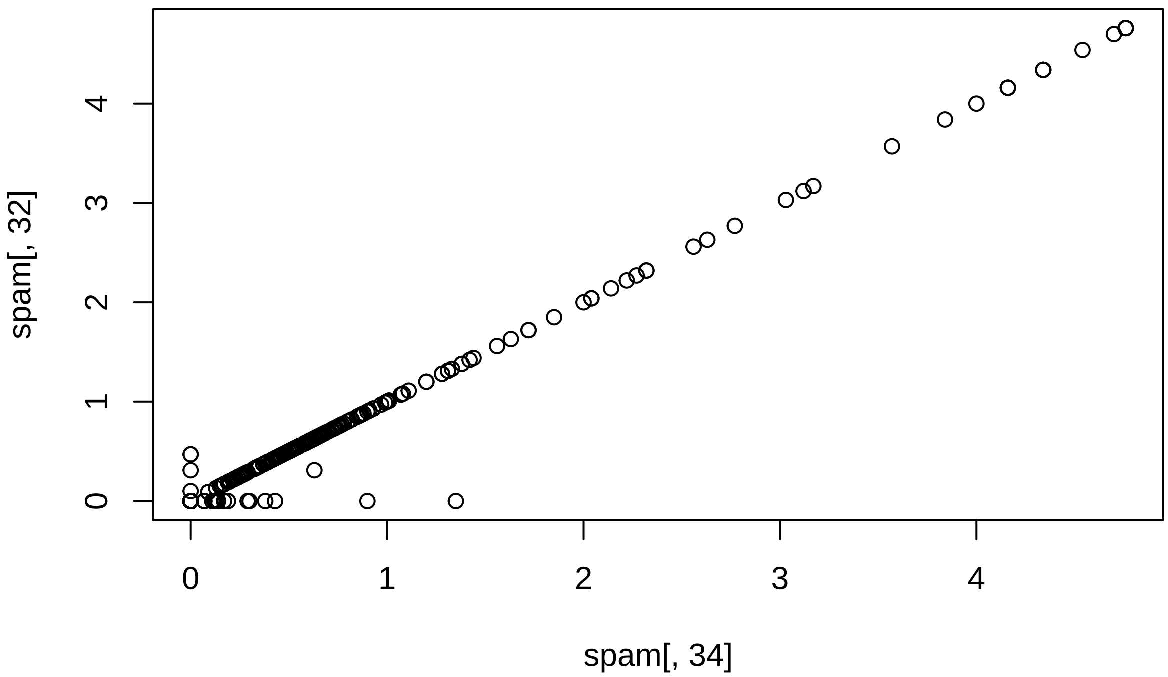
```
##        row col
## num857  32  31
## num415  34  31
## telnet  31  32
## num415  34  32
## direct  40  32
## telnet  31  34
## num857  32  34
## direct  40  34
## num857  32  40
## num415  34  40
```

fa capire quali cose appaiono molto assieme, ad esempio il numero 857 e 415 (n°tel) vado a vedere quindi le colonne in cui appaiono 34,32

```r
names(spam)[c(34,32)]
```

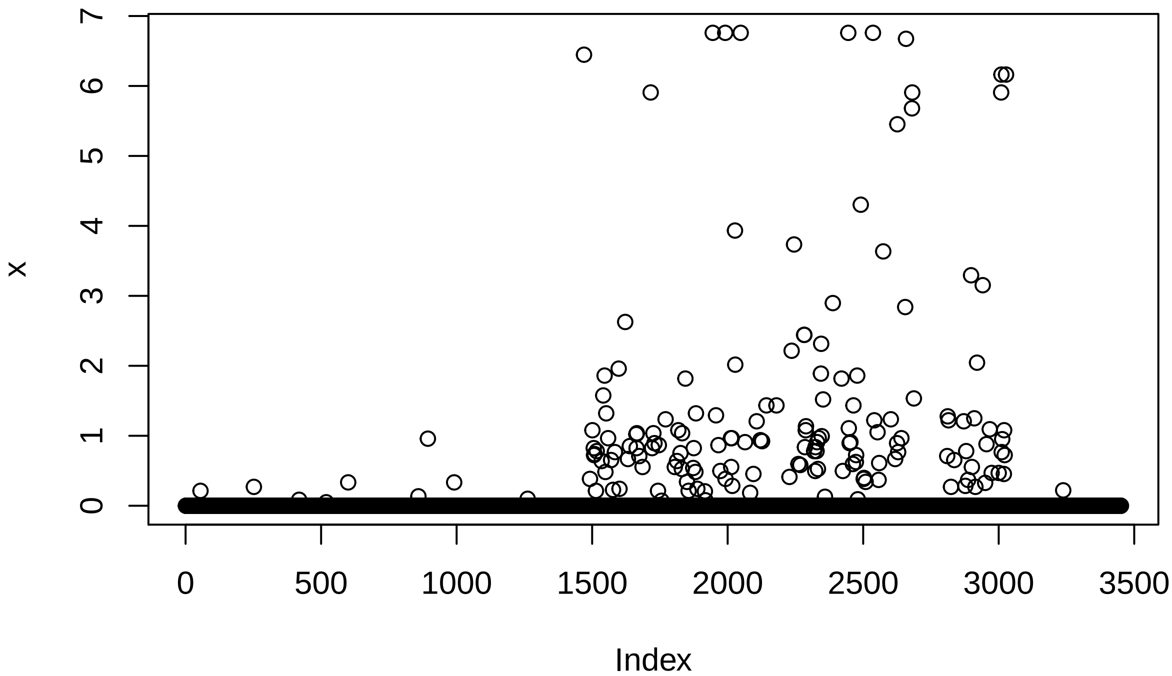
```
## [1] "num415" "num857"
```

```r
plot(spam[,34],spam[,32])
```

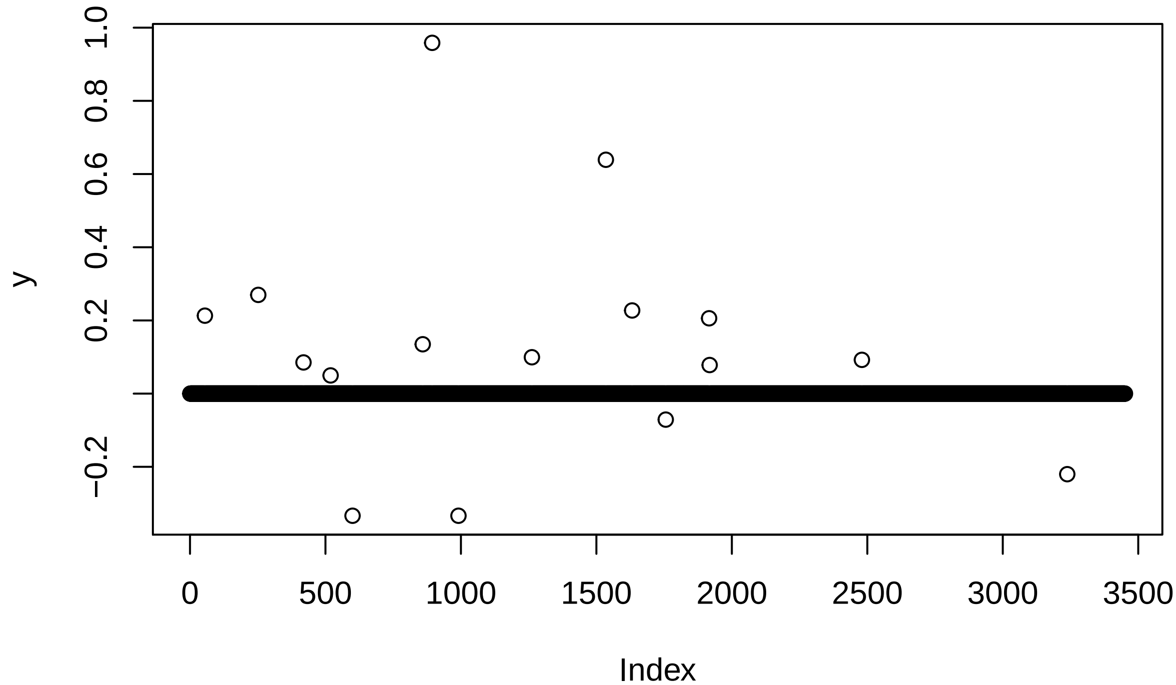


Non è quindi necessario inserirli tutte e due, dobbiamo combinarle con dei pesi comne cominarle???

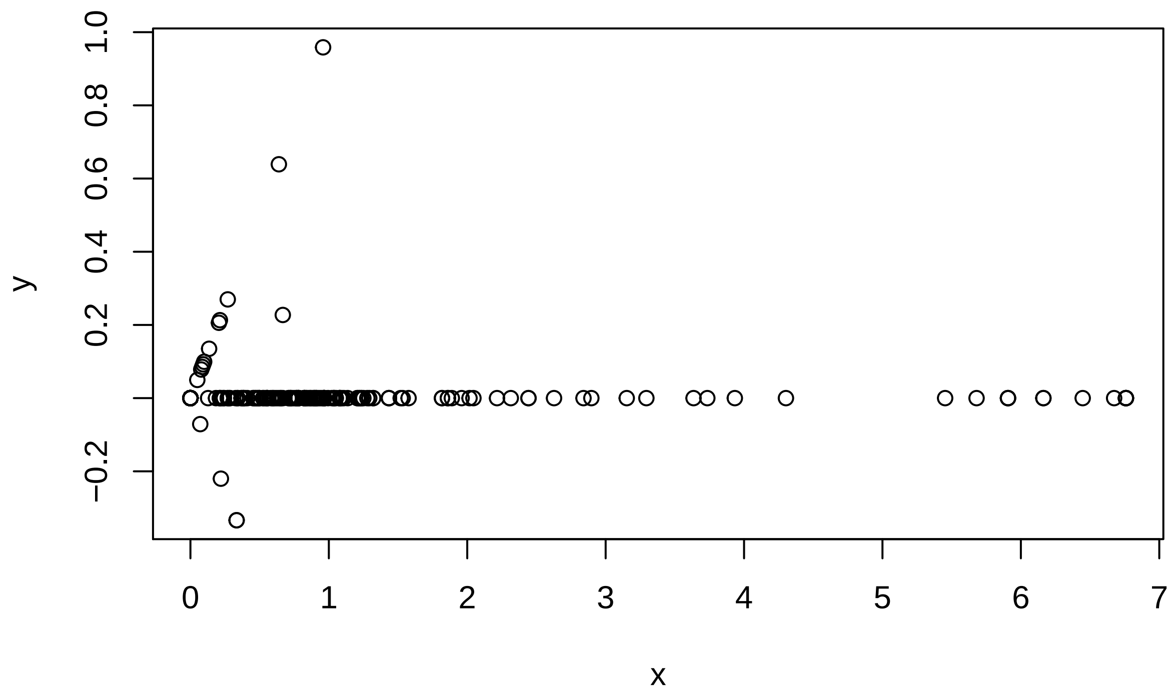
```
x <- 0.71*training$num415 + 0.71*training$num857
y <- 0.71*training$num415 - 0.71*training$num857
plot(x)
```




```r
plot(y)
```




```r
plot(x,y)
```



La maggio parte delle informazioni(quindi più variabile) me le da x quindi sommare, il predittore sarà quindi la somma

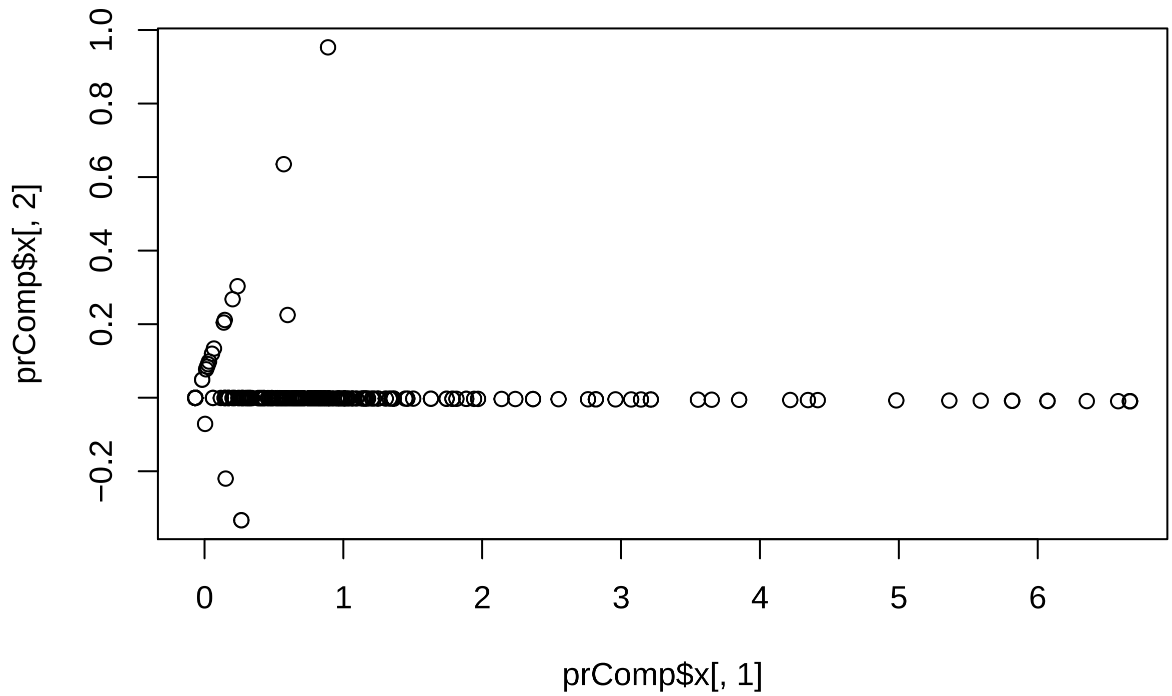
related solution PCA/SVD

singular value decomposition principal component

```
smallspam <- spam[, c(34,32)]
prComp <- prcomp(smallspam)
prComp
```

```
## Standard deviations (1, .., p=2):
## [1] 0.46482184 0.02063535
##
## Rotation (n x k) = (2 x 2):
##               PC1        PC2
## num415 0.7080625  0.7061498
## num857 0.7061498 -0.7080625
```

```
plot(prComp$x[,1], prComp$x[,2])
```



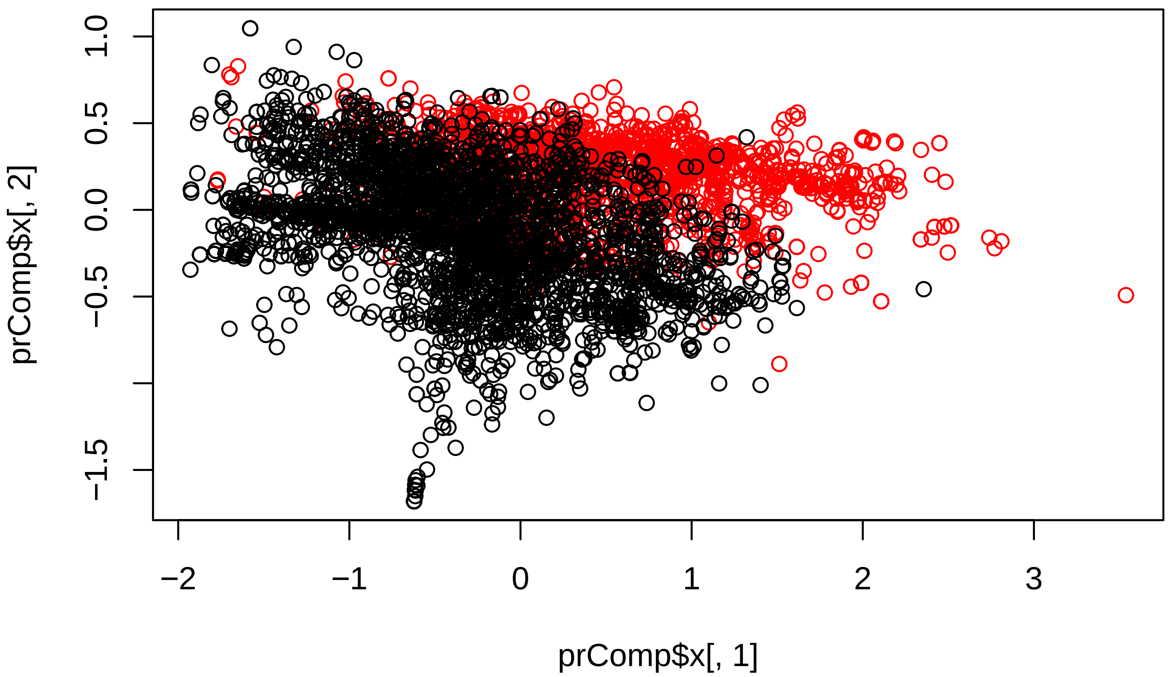
molto simile a quello di prima

```
prComp$rotation
```

```
##              PC1        PC2
## num415 0.7080625  0.7061498
## num857 0.7061498 -0.7080625
```

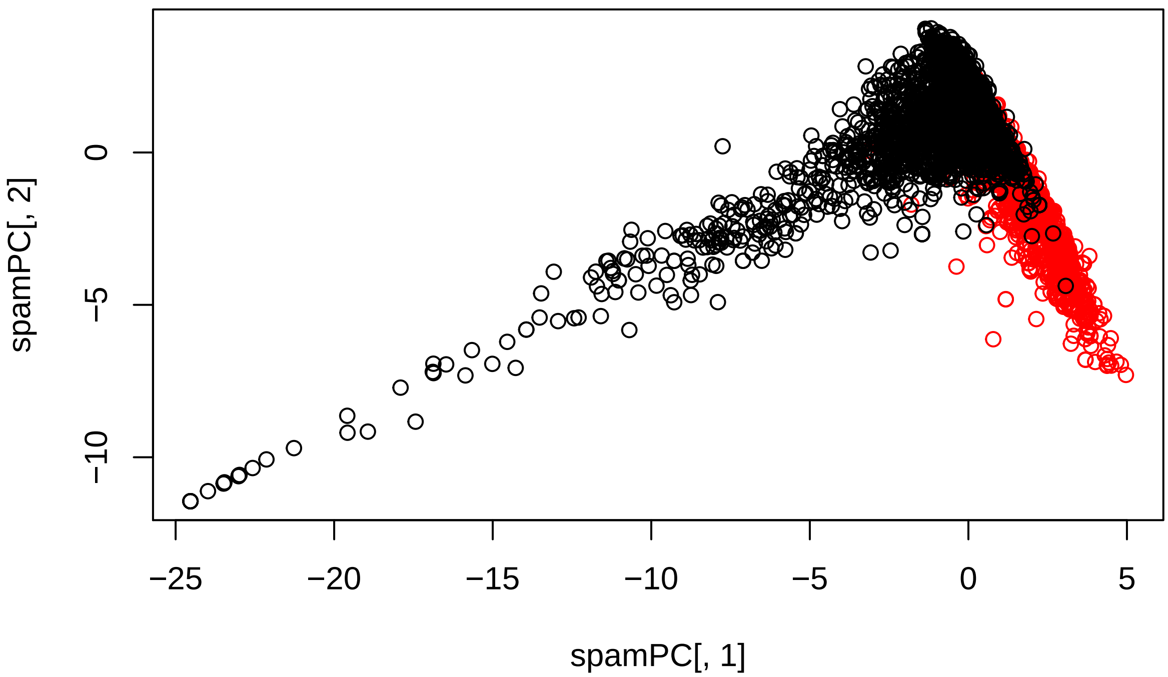
la prima colonna somma la seconda sottrae

```
## creo una variabile per colorare
typecolor <- ((spam$type == "spam")*1 +1)
## calcola i componenti principali dell'intero dataset
## con log li rendo un po più gaussiani
prComp <- prcomp(log10(spam[,-58]+1))
## non sono più somme ma sono cose più complicate
plot(prComp$x[,1],prComp$x[,2], col = typecolor )
```



Sulle x cè un po di divisione su spam e non spam ### PCA with caret

```
## pcaComp è il numero di componenti principali
preProc <- preProcess(log10(spam[,-58]+1), method = "pca", pcaComp =2)
spamPC <- predict(preProc, log10(spam[,-58]+1))
plot(spamPC[,1],spamPC[,2], col = typecolor)
```



preprocessing with PCA

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y = spam$type, p = 0.75 , list = FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]

preProc <- preProcess(log10(training[,-58]+1), method = "pca", pcaComp =2)

trainPC <- predict(preProc, log10(training[,-58]+1))

#modelFit <- train(training$type ~ . , method="glm" , data = trainPC)## bho
```

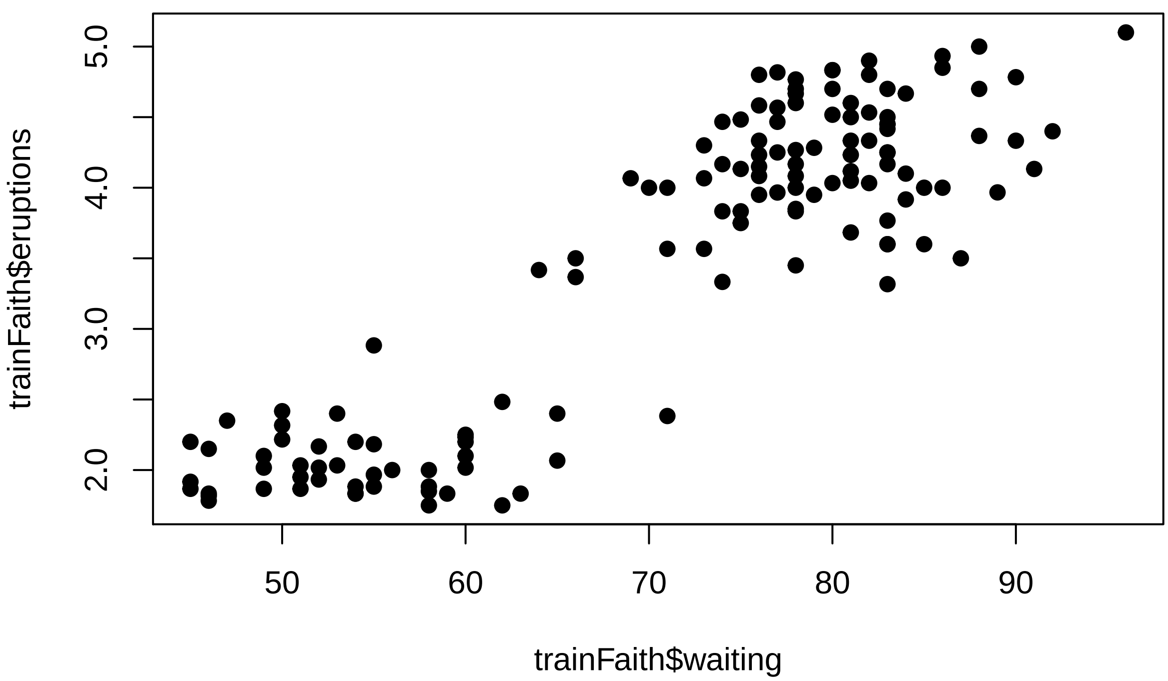
```
#testPC <- predict(preProc, log10(testing[,-58]+1))
#confusionMatrix(testing$type, predict(modelFit,testPC))
```

Predicting with Regression

```
data(faithful); set.seed(333)
inTrain <- createDataPartition(y = faithful$waiting, p = 0.5, list = FALSE)
trainFaith <- faithful[inTrain,]
testFaith <- faithful[-inTrain,]
head(trainFaith)
```

```
##    eruptions waiting
## 3      3.333      74
## 6      2.883      55
## 7      4.700      88
## 8      3.600      85
## 9      1.950      51
## 11     1.833      54
```

```
plot(trainFaith$waiting, trainFaith$eruptions, pch=19)
```



Andiamo a fittare ED = b0 + b1WTi + ei

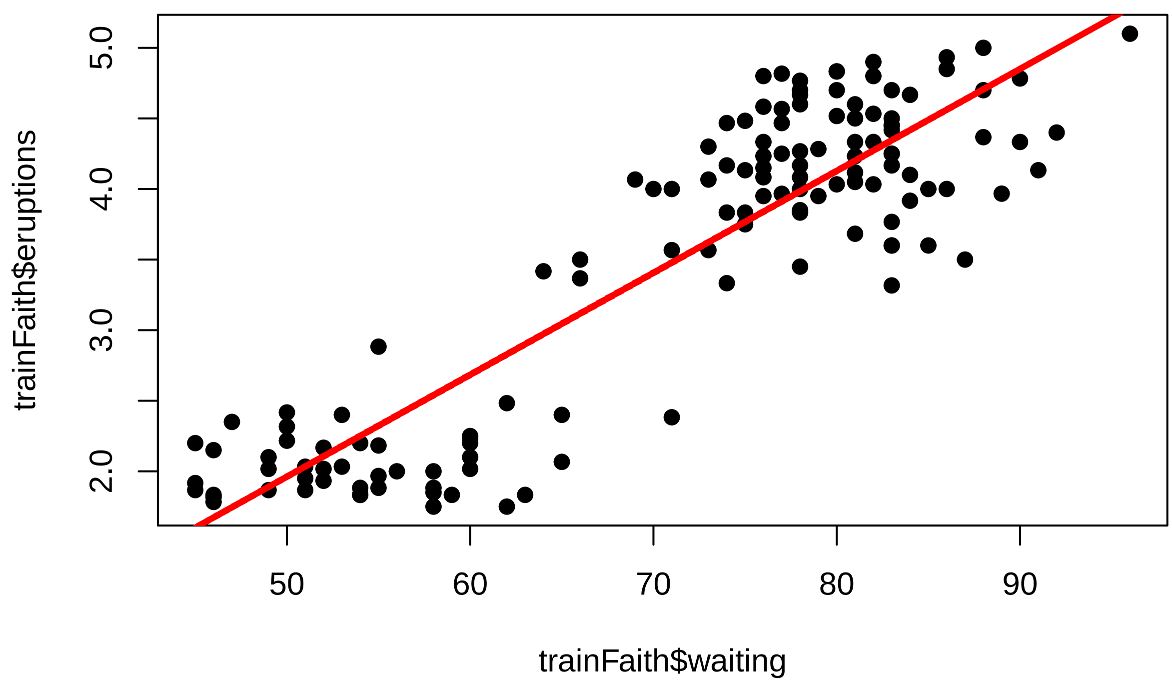
```
lm1 <- lm( data = trainFaith , eruptions ~ waiting)
summary(lm1)
```

```
##
## Call:
## lm(formula = eruptions ~ waiting, data = trainFaith)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13375 -0.36778  0.06064  0.36578  0.96057
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.648629   0.226603  -7.275 2.55e-11 ***
```

```
## waiting      0.072211   0.003136  23.026  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4941 on 135 degrees of freedom
## Multiple R-squared:  0.7971, Adjusted R-squared:  0.7956
## F-statistic: 530.2 on 1 and 135 DF,  p-value: < 2.2e-16
```

intercept estimate è b0 waiting estimate è b1

```r
plot(trainFaith$waiting, trainFaith$eruptions, pch=19)
lines(trainFaith$waiting,lm1$fitted ,lwd =3, col = "red")
```



predict a new value ED^ = b0^ + b1^WT non abbiamo errore perchè non sappiamo quanto sia

```r
## 80 per individuare il waiting
coef(lm1)[1] + coef(lm1)[2]*80
```

```
## (Intercept)
##    4.128276
```

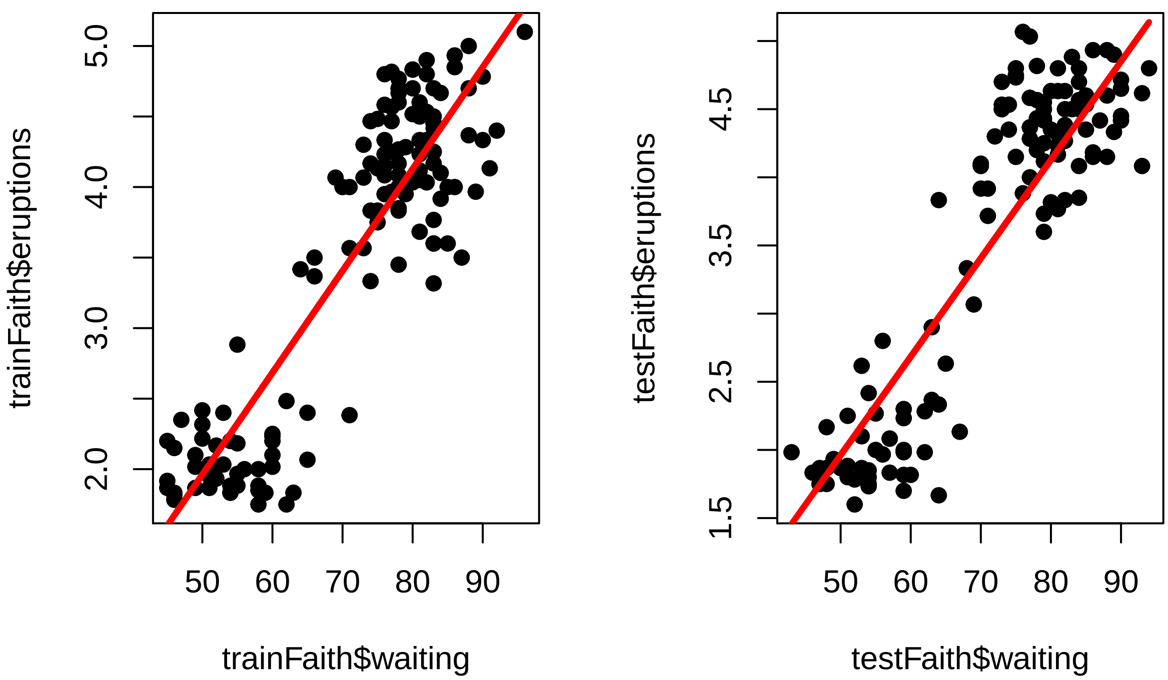
```r
newdata <- data.frame(waiting=80)
newdata
```

```
##   waiting
## 1      80
```

```r
paste0("predizione: ",predict(lm1, newdata))
```

```
## [1] "predizione: 4.12827560449901"
```

```r
par(mfrow = c(1,2))
plot(trainFaith$waiting, trainFaith$eruptions, pch=19)
lines(trainFaith$waiting,predict(lm1) ,lwd =3, col = "red")
plot(testFaith$waiting, testFaith$eruptions, pch=19)
lines(testFaith$waiting,predict(lm1, newdata = testFaith) ,lwd =3, col = "red")
```



La linea di regrassione nella figura test è quella ottenuta attraverso i dati del training, si può vedere che non fitta perfettamente

get training set/test set errors

calcolo RMSE on training

```r
## sottraggo ai valori predetti i valori reali del train
sqrt(sum((lm1$fitted - trainFaith$eruptions)^2))
```

```
## [1] 5.740844
```

calcolo RMSE on test

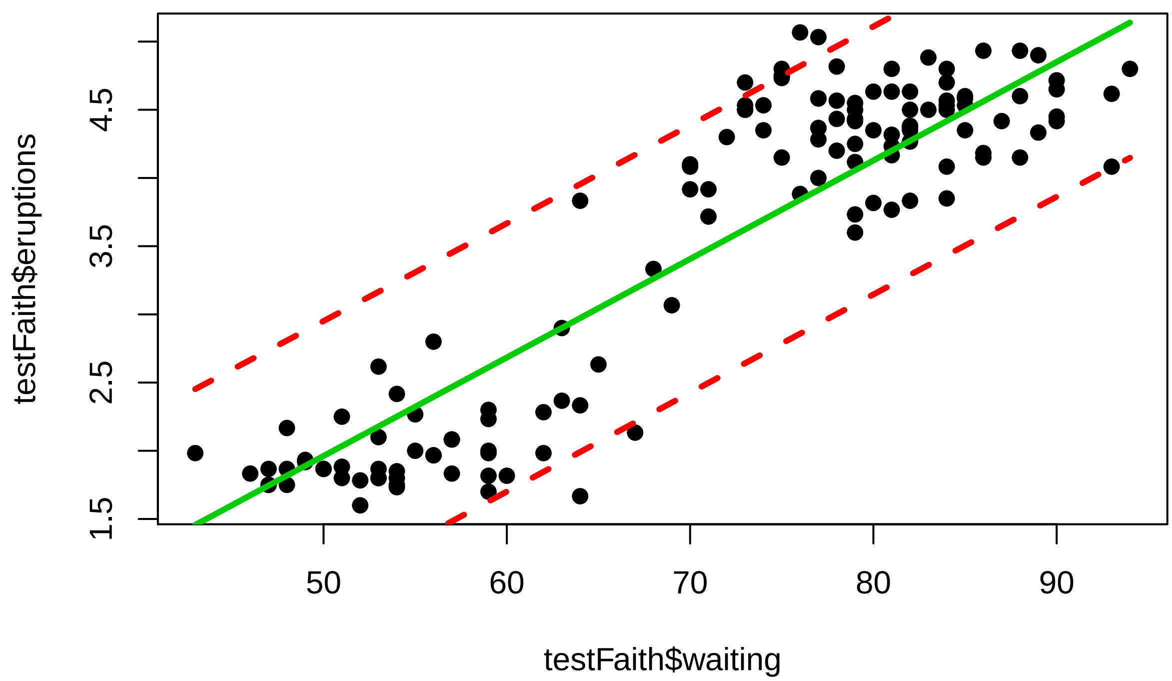
```
## sottraggo ai valori predetti attravers il train i valori reali del test
sqrt(sum((predict(lm1, newdata = testFaith) - testFaith$eruptions)^2))
```

```
## [1] 5.853745
```

Abbastanza normale che sia più grande

prediction intervals

```
## vado a calcolare una nuova predizione pred1 per il test set utilizzando
## il modello lm1 ricavato dal train e gli dico che vvoglio intervallo di predizione
pred1 <- predict(lm1, newdata = testFaith, interval = "prediction")
## ordino i dati per il test set
ord <- order(testFaith$waiting)
plot(testFaith$waiting , testFaith$eruptions, pch =19)
matlines(testFaith$waiting[ord] , pred1[ord,], type = "l", col= c(3,2,2),
         lty = c(1,2,2) , lwd = 3)
```



same process with caret

```
modfit <- train(eruptions ~ waiting , data = trainFaith, method = "lm")
summary(modfit$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13375 -0.36778  0.06064  0.36578  0.96057
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.648629   0.226603  -7.275 2.55e-11 ***
## waiting      0.072211   0.003136  23.026  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4941 on 135 degrees of freedom
## Multiple R-squared:  0.7971, Adjusted R-squared:  0.7956
## F-statistic: 530.2 on 1 and 135 DF,  p-value: < 2.2e-16
```

Predicting with regression, multiple covariates

```r
library(ISLR)
data(Wage);
## togliamo logwage che è quella che vogliomo predirre
Wage <- subset(Wage, select = -c(logwage))
summary(Wage)
```

```
##       year           age                     maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##               education                     region               jobclass
##  1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544
##  2. HS Grad        :971   1. New England       :   0   2. Information:1456
##  3. Some College   :650   3. East North Central:   0
##  4. College Grad   :685   4. West North Central:   0
##  5. Advanced Degree:426   5. South Atlantic    :   0
##                           6. East South Central:   0
##                           (Other)              :   0
##             health      health_ins        wage
##  1. <=Good     : 858   1. Yes:2083   Min.   : 20.09
##  2. >=Very Good:2142   2. No : 917   1st Qu.: 85.38
##                                      Median :104.92
##                                      Mean   :111.70
##                                      3rd Qu.:128.68
##                                      Max.   :318.34
##
```

```r
inTrain <- createDataPartition(y=Wage$wage, p = 0.7 , list = FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   10
```

```
## [1] 898  10
```

Ora si potrebbe fare un bel feature plot ### fit a linear model ED = b0 + b1age +b2jobclass + yklevelk jobclass diventa 1 o 0 education diventa 1 2 3 o 4

```r
## in automatico R converte i fattori in numeri
modFit <- train(wage ~ age + jobclass + education, method = "lm",
                data = training)

finMod <- modFit$finalModel
print(modFit)
```

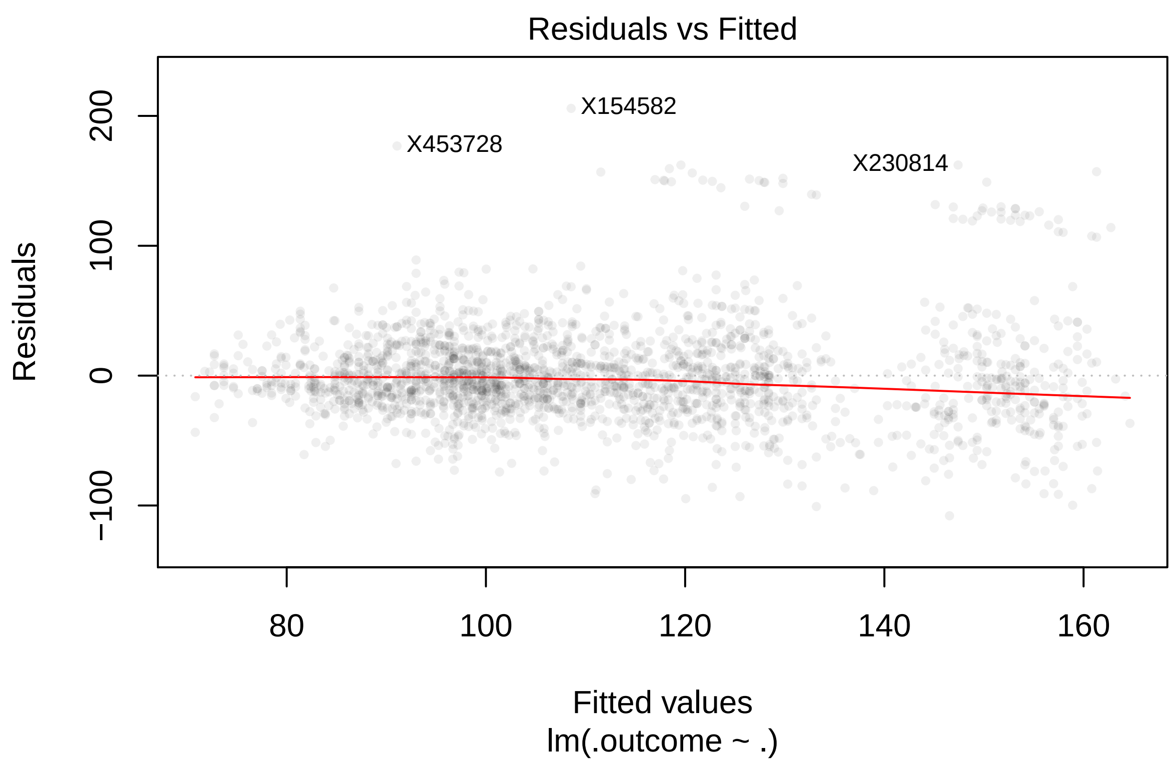
```
## Linear Regression
##
## 2102 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   35.56759  0.2589245  24.87554
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
print(finMod)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##                   (Intercept)                            age
##                       61.7127                         0.4793
##      `jobclass2. Information`          `education2. HS Grad`
##                        4.2139                        12.1096
##    `education3. Some College`     `education4. College Grad`
##                       24.4166                        39.4505
## `education5. Advanced Degree`
##                       65.1802
```

diagnostic

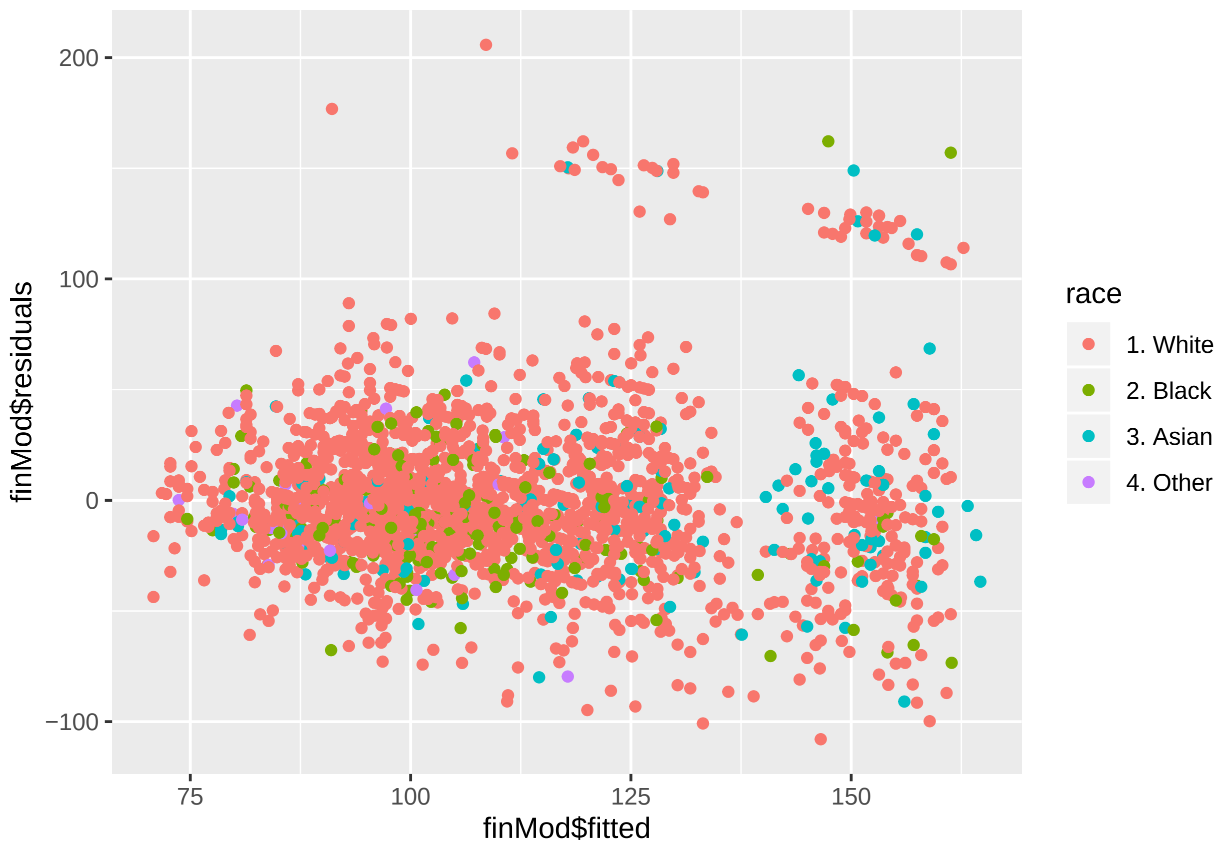
```
plot(finMod, 1, pch = 19, cex = 0.5, col ="#00000010")
```



Ci sono alcuni valori marcati che sono outlayer e magari da esplorare e trovare qualche predittore che le spiega meglio

color by variables not used in the model

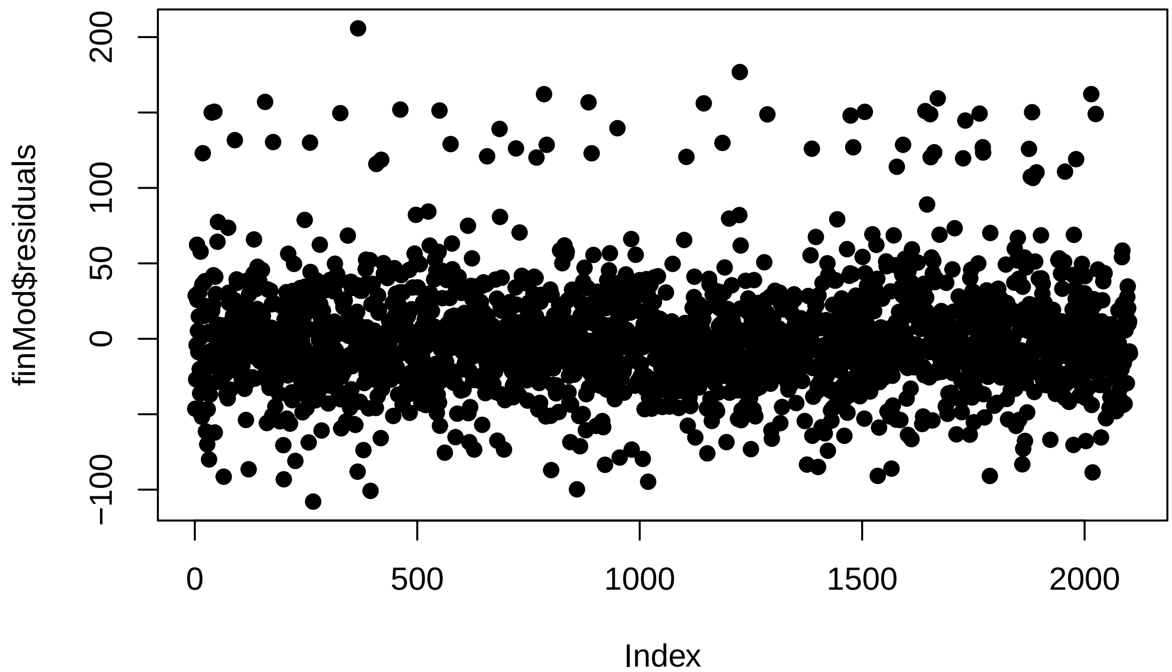
```
qplot(finMod$fitted, finMod$residuals, colour = race, data = training)
```



può spiegare gli outlayer

plot by index

```
plot(finMod$residuals , pch =19)
```

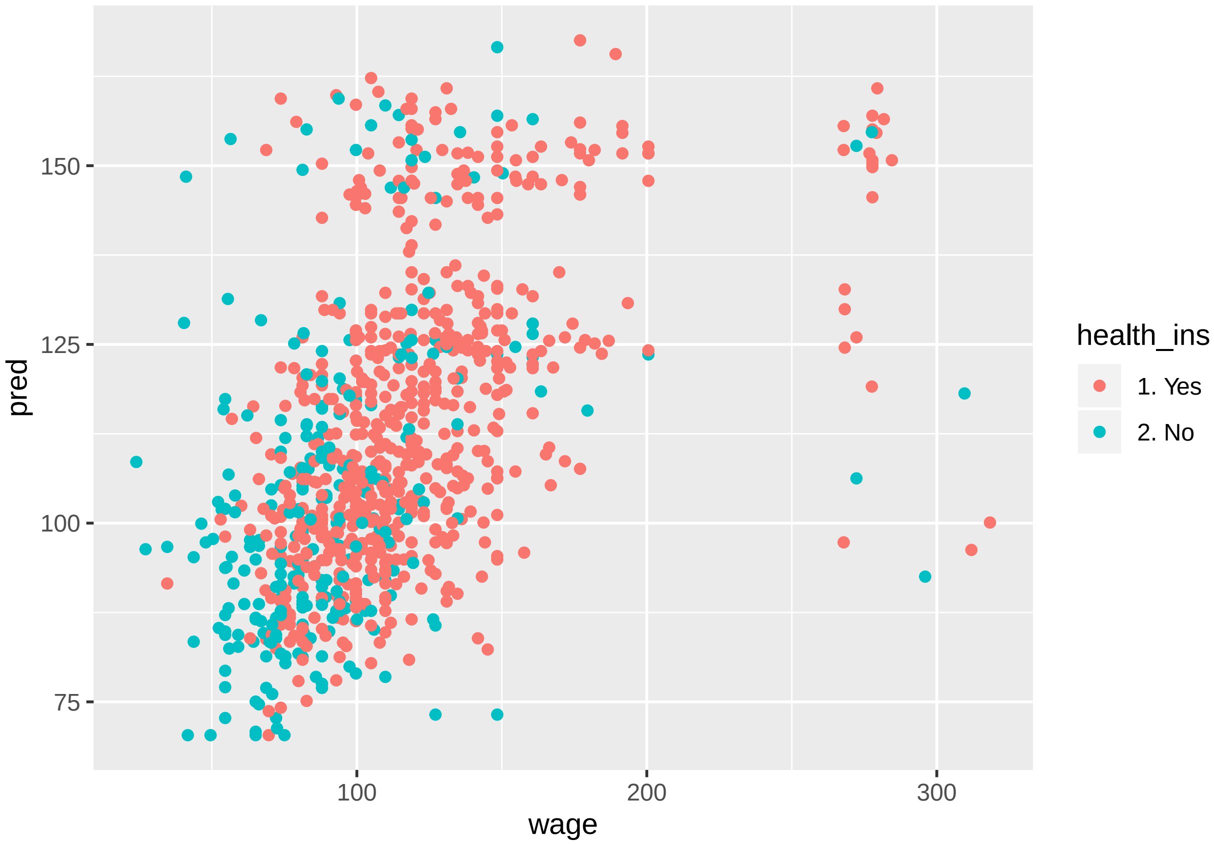


L'index identifica in che riga stiamo osservando, se cè un trend o un gruppo outlayer probabillmente si è mancato di aggiungere qualche predittore nel modello

Predicted versu truth in test set

risultato della predizione con il test set, decido di colorare per vedere se ho mancato qualche predittore

```
pred <- predict(modFit, testing)
## in colour posso mettere qualsiasi variabile non utilizzata
qplot(wage, pred, colour = health_ins, data = testing)
```



compara il wage del test set con i valori predetti attraverso il training, la perfezione sarebbe una linea a 45° Non ha senso poi andare a rifare il modello inserendo questo predittore, è un post mortem analysis per vedere perchè ha fallito il nostroML

se si vuole utilizzare all covariates

includo tutti i predittori

```
modFitAll <- train(wage ~ . , data = training, method  = "lm")
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
```

```
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
```

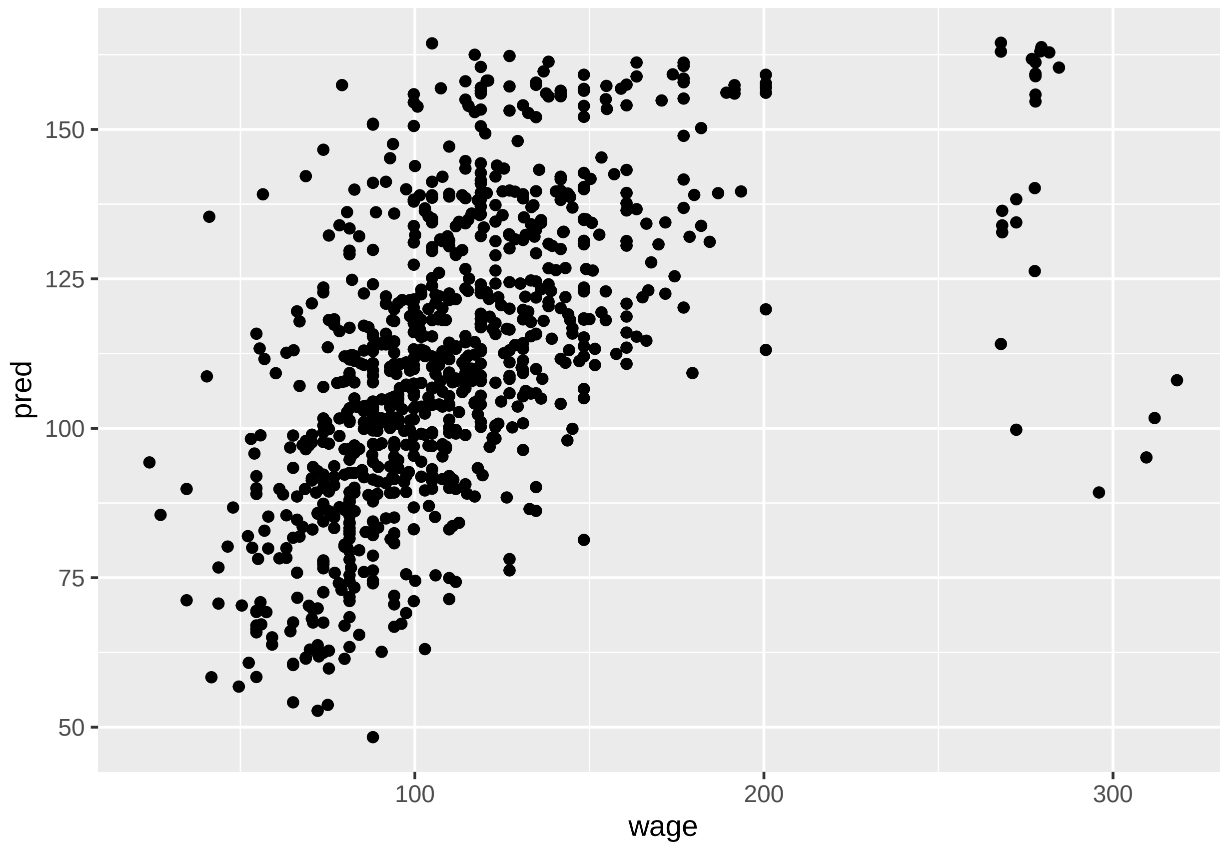
```
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
pred <- predict(modFitAll, testing)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
qplot(wage, pred, data = testing)
```



Anche con tutti i predittori alcuni punti non sono ben spiegati