R Project - Supervised Learning with Logistic Regression

Machine Learning in Management - year 2022/23

Matteo Scarciglia

Contents

Starting point

Differently from the previous situation, where the asked final output was a numerical result, here is represented the condition in which the searched knowledge lies between a qualitative (or categorical) output. In such situation what a regression model do is to give a univocal solution which is find using the probability. For this reason, the regression problem that involves a categorical output is referred as classifier and here one of those is built using multiple logistic regression.

The main idea is to meet the probability of occurrence of certain event and relate to them a value which is understandable as Yes/Not or what else. More precisely, there is a logistic function which will be explain later in this project.

Data Description: Importing Dataset

The dataset was published on UC Irvine Machine Learning Repository, which is the public archive of the University of California Irvine. This data approach student's achievement in secondary education of two Portuguese school and include student's grades, demographic, social and school related features collected by reports and pools.

```
dataset <- readxl::read_xls("student-mat.xls")

attach(dataset)

dim(dataset)
```

```
## [1] 395  31
```

```
names(dataset)
```

```
##  [1] "school"     "sex"        "age"        "address"    "famsize"
##  [6] "Pstatus"    "Medu"       "Fedu"       "Mjob"       "Fjob"
## [11] "reason"     "guardian"   "traveltime" "studytime"  "failures"
## [16] "schoolsup"  "famsup"     "paid"       "activities" "nursery"
## [21] "higher"     "internet"   "romantic"   "famrel"     "freetime"
## [26] "goout"      "Dalc"       "Walc"       "health"     "absences"
## [31] "Grade"
```

As can be see, there are quite a 31 variables measured, regarding the school provenience, sex, family size and more. As first, a supervised learning algorithm like Best Subset Selection will be involved for choose all the variables who really matter for being explored. To solve this, some model will be created and the best will be chosen according to a metric.

The target variable is Grade, that indicates whether the final grade is above the median or not. This will be 1 if yes while 0 otherwise. Further information about the dataset are available here: https://archive-beta.ics.uci.edu/dataset/320/student+performance.

Transform, Tidy and Prepare the data

The first best practice as always is checking if there are missing values:

```
sum(is.na(dataset))
```

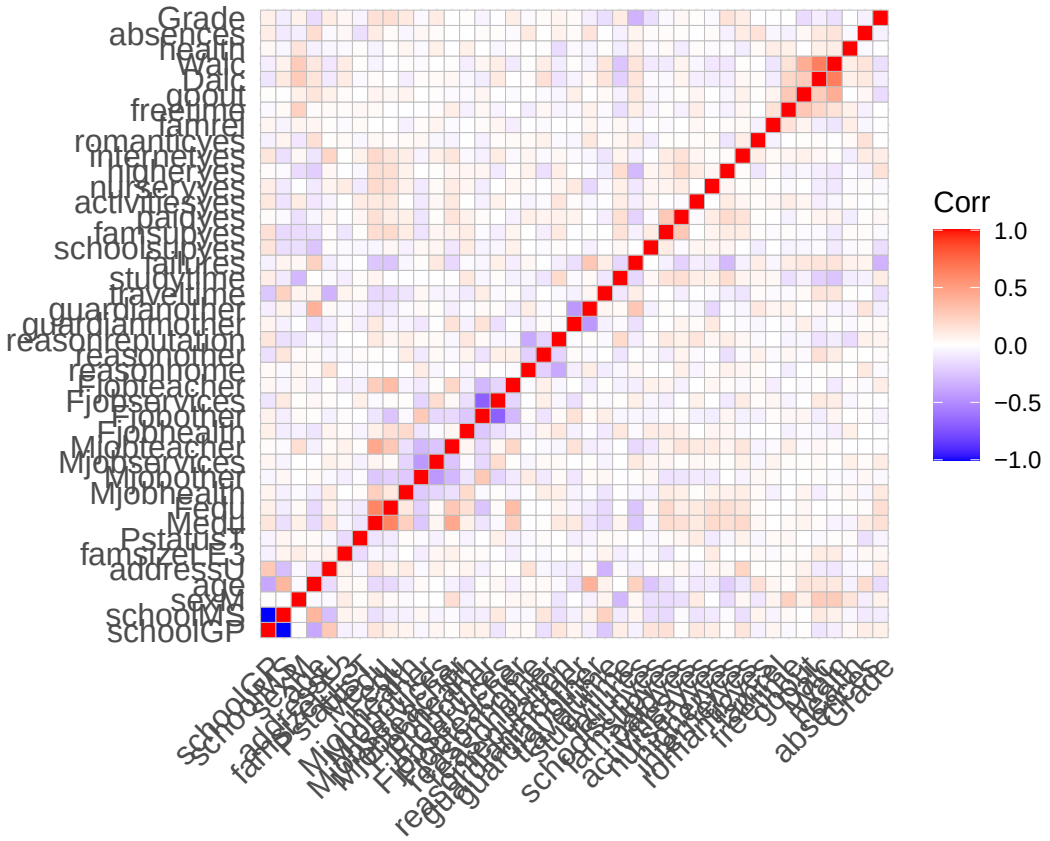
```
## [1] 0
```

There are not missing values, so a deep cleaning is not required.

Knowing more about data

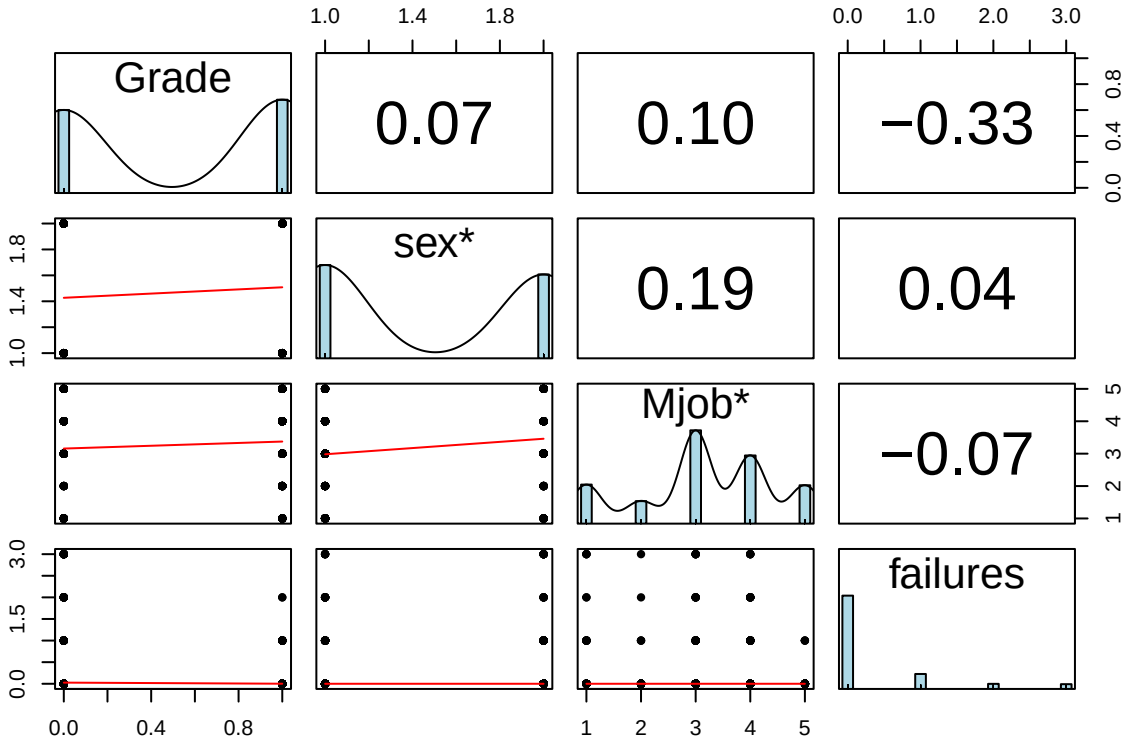
Knowing more about data as first can be useful for various order of reasons. Now, since a linear model will be created looking for correlations can be smart:

```
library(ggcorrplot)

heatmap <- ggcorrplot::ggcorrplot(cor(model.matrix(data=dataset[,], ~.-1)))

heatmap
```



Nothing worrying regard to a linear model construction should be inside the data. Some values in heatmap change just because the model.matrix() function automatically turns out the dummy variables in multiple attributes with 0 and 1. Here below is represented some other descriptive plots:

```
library(psych)

pro <- data.frame(dataset$Grade,dataset$sex,dataset$Mjob,dataset$failures)

colnames(pro) <- c("Grade","sex","Mjob","failures")

pairs.panels(pro, method="pearson", hist.col="lightblue",density=TRUE,
             ellipses=FALSE)
```



Choosing the best variables

Since there are 31 variables which are quite a lot, probably some of them are not useful as others. To proceed without removing them can turns out useless component in the model which not only make the model more difficult to understand but influence the output badly also. For have a comparison, here below an OLS linear model with all coefficient's estimates is built:

```
library(stats)

set.seed(1)

model <- glm(data=dataset, Grade~., family=gaussian)

summary(model)
```

```
##
## Call:
## glm(formula = Grade ~ ., family = gaussian, data = dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9998  -0.4034   0.1264   0.3513   0.9356
##
## Coefficients:
```

```
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.0558732  0.4962926   2.128  0.03407 *
## schoolMS         -0.0738146  0.0876722  -0.842  0.40039
## sexM              0.1098604  0.0553820   1.984  0.04806 *
## age              -0.0303937  0.0240581  -1.263  0.20729
## addressU          0.0321602  0.0646957   0.497  0.61943
## famsizeLE3        0.0537957  0.0540764   0.995  0.32051
## PstatusT         -0.0440456  0.0801772  -0.549  0.58311
## Medu             -0.0106004  0.0357935  -0.296  0.76729
## Fedu              0.0562281  0.0307485   1.829  0.06829 .
## Mjobhealth        0.2682210  0.1238476   2.166  0.03100 *
## Mjobother         0.1169996  0.0789883   1.481  0.13943
## Mjobservices      0.2191265  0.0883668   2.480  0.01361 *
## Mjobteacher      -0.0476534  0.1149896  -0.414  0.67882
## Fjobhealth       -0.1110336  0.1592652  -0.697  0.48616
## Fjobother        -0.0847260  0.1133096  -0.748  0.45511
## Fjobservices     -0.0548413  0.1170678  -0.468  0.63974
## Fjobteacher       0.0478220  0.1436016   0.333  0.73932
## reasonhome       -0.0073393  0.0613380  -0.120  0.90483
## reasonother       0.0217071  0.0905523   0.240  0.81069
## reasonreputation  0.0265542  0.0638591   0.416  0.67779
## guardianmother    0.0667229  0.0604290   1.104  0.27027
## guardianother     0.0820120  0.1106982   0.741  0.45927
## traveltime       -0.0244010  0.0375430  -0.650  0.51615
## studytime         0.0570387  0.0318600   1.790  0.07426 .
## failures         -0.1809049  0.0368719  -4.906 1.42e-06 ***
## schoolsupyes     -0.2356120  0.0738675  -3.190  0.00155 **
## famsupyes        -0.0835665  0.0530186  -1.576  0.11588
## paidyes           0.0058142  0.0529143   0.110  0.91257
## activitiesyes    -0.0488507  0.0492807  -0.991  0.32223
## nurseryyes       -0.1225220  0.0608404  -2.014  0.04478 *
## higheryes         0.1004439  0.1193742   0.841  0.40068
## internetyes       0.0393533  0.0686205   0.573  0.56667
## romanticyes       0.0187895  0.0519729   0.362  0.71792
## famrel           -0.0227912  0.0272381  -0.837  0.40330
## freetime          0.0289797  0.0262887   1.102  0.27105
## goout            -0.0566244  0.0248658  -2.277  0.02337 *
## Dalc              0.0174416  0.0366469   0.476  0.63441
## Walc             -0.0297247  0.0274690  -1.082  0.27994
## health           -0.0061108  0.0178333  -0.343  0.73205
## absences         -0.0008773  0.0032083  -0.273  0.78466
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2070171)
##
##     Null deviance: 98.415  on 394  degrees of freedom
## Residual deviance: 73.491  on 355  degrees of freedom
## AIC: 538.68
##
## Number of Fisher Scoring iterations: 2
```

The significant variables are just a few, so the previous condition becomes true. As seen in the previous project, one widely used method for choosing the best subset of variables which compose the best model

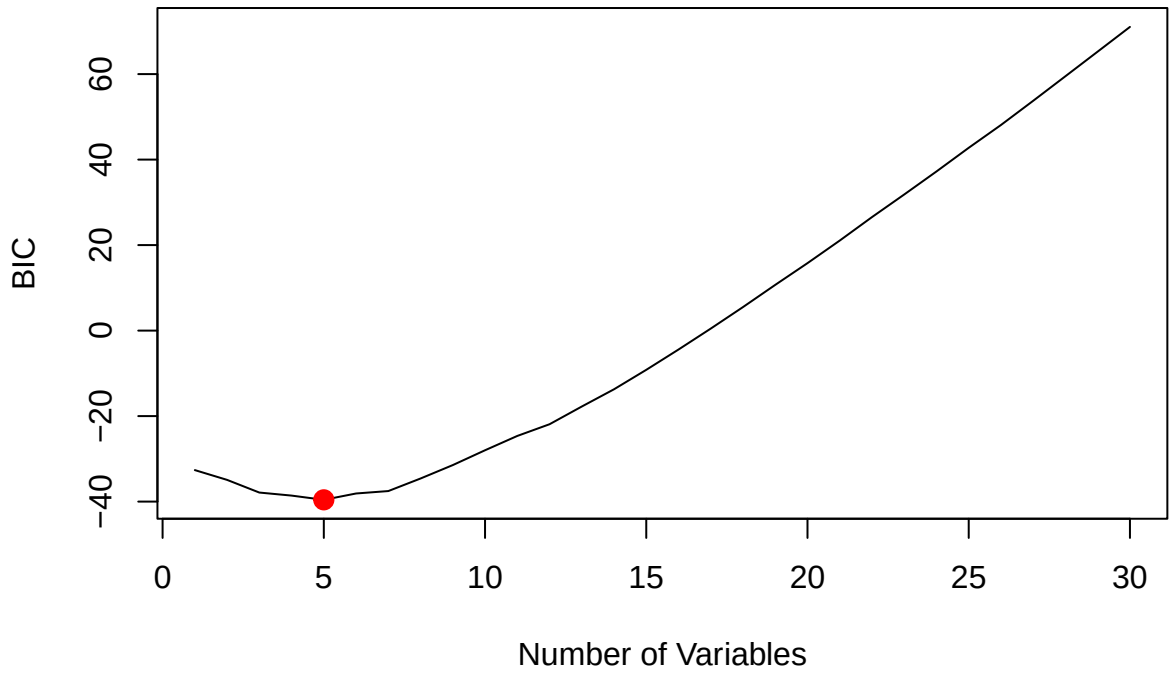
is the best subset selection approach. Without getting deeper in it's work principle, here is applied the regbestsubsets() function to find the best model according to a p number of predictors:

```
library(leaps)

set.seed(1)

model_best <- regsubsets(data=dataset, Grade~., method="forward", nvmax=30)
```

The best model among these is the model who have the lowest test error. According to the BIC statistic previously explained, that is most indicate where a model with a low number of predictors is better or even when the n number of observation is not too much than p variables, a plot is created and then the best model is chosen:

```
pimin.bic <- which.min(summary(model_best)$bic)

plot(summary(model_best)$bic, xlab="Number of Variables", ylab="BIC", type="line")+
  points(pimin.bic, summary(model_best)$bic[pimin.bic], col="red", cex=2, pch=20)
```

```
## Warning in plot.xy(xy, type, ...): il tipo plot 'line' sarà troncato al primo
## carattere
```




```
## integer(0)
```

The model which has the lowest test error is the model which contains 5 variables:

```
coef(model_best,5)
```

```
##  (Intercept)   Mjobhealth Mjobservices     failures schoolsupyes        goout
##   0.74095702   0.22036705   0.18554766  -0.21700388  -0.20915318  -0.05779364
```

From now, this is the base for the following application, so let's instance a new dataset containing only these variables:

```
data_prova <- data.frame(Grade, Mjob, failures, schoolsup, goout)
```

Logistic Regression

Now, let's create another model which contains only these variables:

```
model_def <- glm(data=data_prova, Grade ~ ., family=gaussian)

summary(model_def)
```

```
##
## Call:
## glm(formula = Grade ~ ., family = gaussian, data = data_prova)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8723  -0.4630   0.1871   0.3778   0.9102
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.64119    0.08687   7.381 9.68e-13 ***
## Mjobhealth    0.32549    0.09869   3.298 0.001063 **
## Mjobother     0.15919    0.07068   2.252 0.024863 *
## Mjobservices  0.29048    0.07439   3.905 0.000111 ***
## Mjobteacher   0.07949    0.08532   0.932 0.352063
## failures     -0.21629    0.03163  -6.839 3.12e-11 ***
## schoolsupyes -0.21206    0.06889  -3.078 0.002229 **
## goout        -0.05940    0.02083  -2.851 0.004589 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2072369)
##
##     Null deviance: 98.415  on 394  degrees of freedom
## Residual deviance: 80.201  on 387  degrees of freedom
## AIC: 509.19
##
## Number of Fisher Scoring iterations: 2
```

As expected, differently to the previous one more variables are statistically significant in explaining the variance. The results are interpreted as for all the values fixed, a unitary increase for *failures* variable

involves a reduction of 0.22 of probability to be above the average. Following these logic, a student that has 1 higher past class failure has the **20% lower probability to be above the average**.

The results, in fact, have to be intended as probability of occurrence of a certain event, fixed all the others. For example, let's make one prediction:

```
preds <- predict(model_def, data.frame(Mjob=c("health","other","services","teacher"), failures=1, schoo
# the result is the probability related to a student which Mjob= health, failures=1, ecc.

names(preds)[1:4] <- c("health","other","services","teacher")
```

For a student that go out with friends 4 time, has not extra educational support and has 1 failure, these are the probabilities to be above the average grade according to his mother's job:

```
preds
```

```
##    health     other  services   teacher
## 0.5127745 0.3464798 0.4777693 0.2667778
```

The probability will decrease a lot if the number of failures increases:

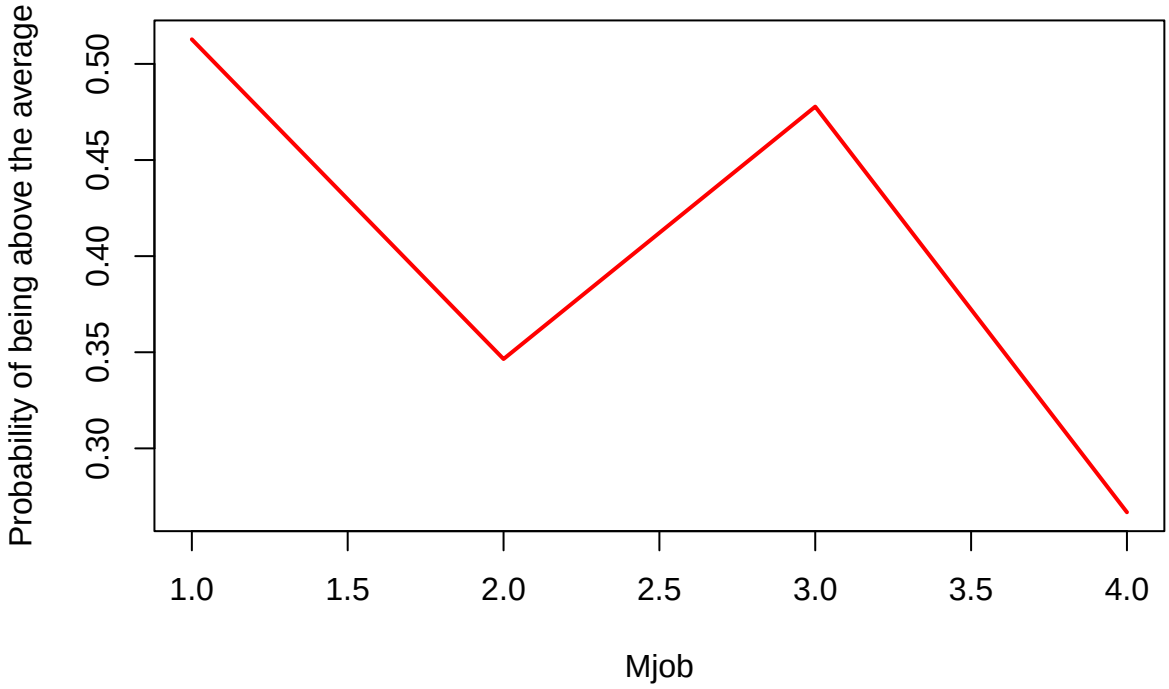
```
preds_4 <- predict(model_def, data.frame(Mjob=c("health","other","services","teacher"), failures=4, sch

names(preds_4)[1:4] <- c("health","other","services","teacher")

preds_4
```

```
##     health      other   services    teacher
## -0.1360936 -0.3023883 -0.1710988 -0.3820903
```

The probability plotted according to mother's Job:

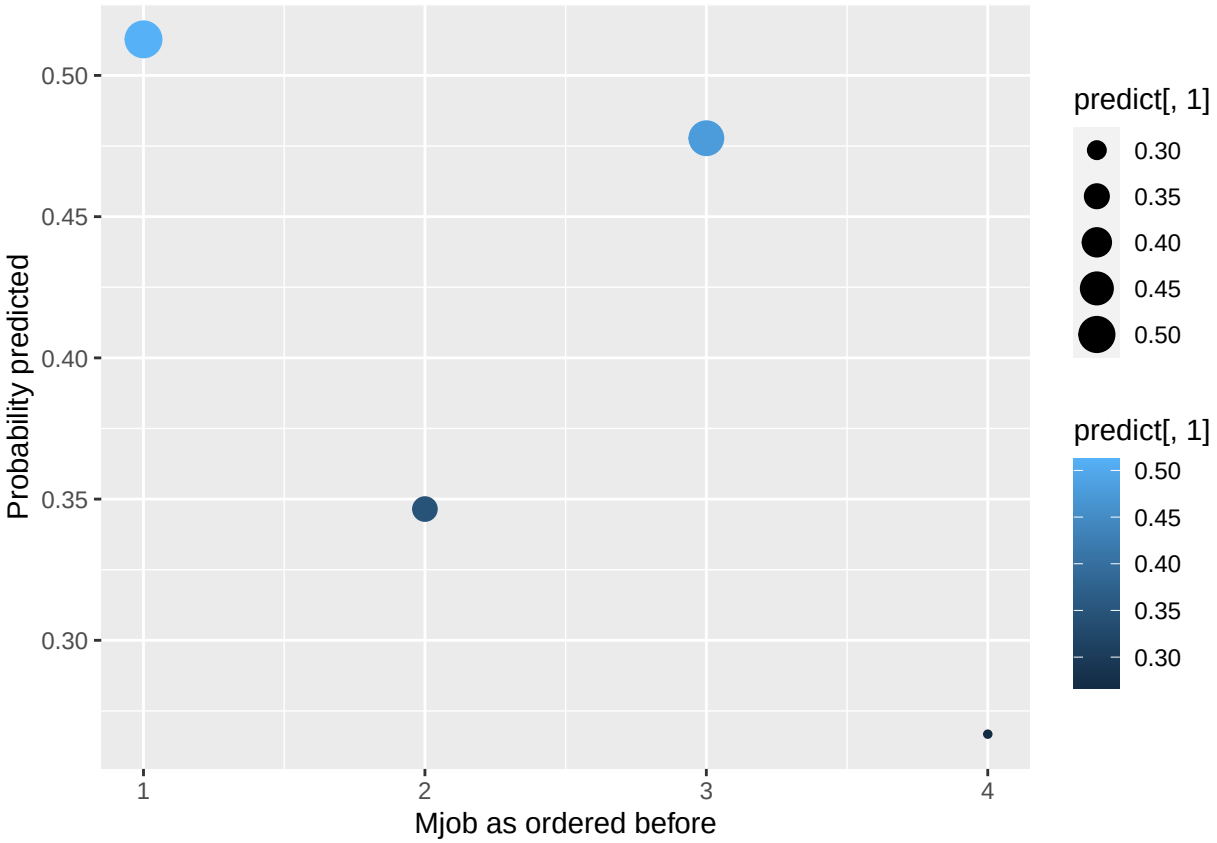
```
plot(preds, type="line", ylab="Probability of being above the average",
     xlab="Mjob",lwd=2,col="red")
```

```
## Warning in plot.xy(xy, type, ...): il tipo plot 'line' sarà troncato al primo
## carattere
```



Or even:

```
library(ggplot2)

predict <- as.data.frame(preds)

ggplot(predict,aes(x=1:4,y=predict[,1]))+
  geom_point(aes(colour=predict[,1], size=predict[,1]))+
  labs(x="Mjob as ordered before",y="Probability predicted")
```



Model evalutation

There are several ways to assess the performance of a logistic regression model, but the most used are using the following metrics:

- **Accuracy**, which is the percentage of correct observation and indicates how often the classifier is correct:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- **Precision**, which shows how accurately the classifier predict positive outcomes by mean of the proportion of positive predictions out of all the positive:

$$Precision = \frac{TP}{TP + FP}$$

- **Sensitivity**, which is the percentage of positive outcomes correctly predicted:

$$Sensitivity = \frac{TP}{TP + FN}$$

- **Specificity**, which is the proportion of negative outcomes predicted as negative:

$$Specificity = \frac{TN}{TN + FP}$$

All of these metrics can be computed using a confusion matrix, crafted by confusionMatrix() function in R. The elements of the confusion matrix are the **True Positive (TP)** in which both prediction and outcome are positive, the **False Positive (FP)** when prediction is positive and outcome is negative, the **True Negative (TN)** when prediction is negative and outcome is negative too, and the **False Negative (FN)** in which prediction is negative but outcome is positive.

As first, let's create a rudimental confusion matrix made comparing the predicted outcome and the originals:

```
model_best_pred <- ifelse(predict(model_def, type="response")>0.5, 1, 0)

prova.mat=table(model_best_pred, data_prova$Grade)

prova.mat
```

```
##
## model_best_pred   0   1
##               0 109  40
##               1  77 169
```

The columns represent the actual value while for every row is represented the predicted one, classified by mean of the related probability. In the previous chunck, using ifelse() function every probability above the 50% is converted to be 1 and vice versa.

Now that the table matrix is created the actual confusion matrix can be crafted:

```
library(caret)

conf.mat <- confusionMatrix(prova.mat)

conf.mat
```

```
## Confusion Matrix and Statistics
##
##
## model_best_pred   0   1
##               0 109  40
##               1  77 169
##
##                Accuracy : 0.7038
##                  95% CI : (0.6561, 0.7484)
##     No Information Rate : 0.5291
##     P-Value [Acc > NIR] : 1.064e-12
##
##                   Kappa : 0.399
##
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 0.5860
```

```
##             Specificity : 0.8086
##          Pos Pred Value : 0.7315
##          Neg Pred Value : 0.6870
##              Prevalence : 0.4709
##          Detection Rate : 0.2759
##    Detection Prevalence : 0.3772
##       Balanced Accuracy : 0.6973
##
##        'Positive' Class : 0
##
```

All the metrics are recalled in order:

```
conf.mat$overall["Accuracy"] # Accuracy
```

```
##  Accuracy
## 0.7037975
```

```
conf.mat$byClass["Pos Pred Value"] # Precision
```

```
## Pos Pred Value
##      0.7315436
```

```
conf.mat$byClass["Sensitivity"] # Sensitivity
```

```
## Sensitivity
##   0.5860215
```

```
conf.mat$byClass["Specificity"] # Specificity
```

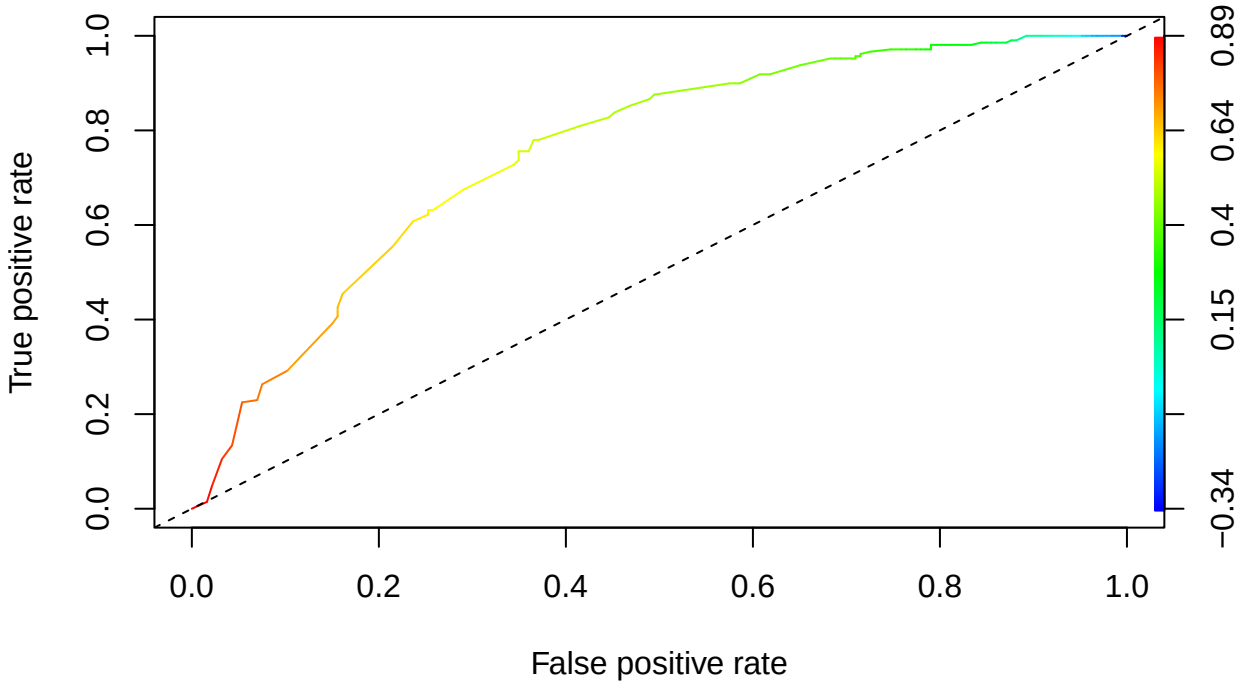
```
## Specificity
##   0.8086124
```

The accuracy of the model is not too high considering that is 0.70; however, this means that the classifier is correct 70% of the time, which meaning that works decently without claiming to be infallible and it is simple to interpret.

The ROC curve

The ROC curve illustrates the diagnostic ability of a classifier as its discrimination threshold is varied. Previously, all the probability above the 0.50 were coerced to became 1 and vice versa; this means that the threshold is of the 50% and changing this means changing the interpretation of the classifier. The ROC curve is created by plotting the true positive rate (TPR - Sensitivity) against the false positive rate (FPR) at various threshold settings. The FPR, also known as the fall-out, is the number of false positives divided by the number of false positives plus the number of true negative, or more easily by one minus the specificity. A ROC curve plots TPR vs. FPR at different classification thresholds. **Lowering the classification threshold classifies more items as positive**, thus increasing both False Positive (FP) and True Positive (TP). The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The color indicates the corresponding values of true positive rate and false positive rate vs. a specific cutoff level instead.

```
library(ROCR)

pred <- prediction(predict(model_def, newdata=data_prova,type="response"),
                   data_prova$Grade)
perf <- performance(pred,"tpr","fpr")

plot(perf,col="red",colorize=TRUE)+
abline(a=0, b=1,lty="dashed")
```




```
## integer(0)
```

The area under the ROC curve **(AUC)** is a way to assess the predictive ability of a model over all possible thresholds. If it is 1.0, the model predict 100% true positive and 0% false positive, so the dashed line in the plot corresponds to an AUC of 0.5 and corresponds to a model which makes random predictions. Let's check the AUC:

```
AUC <- performance(pred, measure="auc")

AUC@y.values
```

```
## [[1]]
## [1] 0.7546818
```

The AUC is 0.75 which is in the middle among a random chosen predictions and a model which is infallible.