Statistical Learning Project

Jacopo Guidolin - Mattia Miolato - Giovanni Vedana

22 June 2021

Model Selection

We start by splitting our dataset into a train and a test set, where the training samples are 66% of the total number of samples. Initially we have divided our samples into three different sets, exploiting also a validation set, but then we preferred to use only train and test set since there is a small number of fraudulent transactions.

In what follows we use the train set to fit all the models and to search for the best threshold that identify one sample in a class rather than in the other. Instead we leave the test set exclusively to validate the model.

```r
set.seed(1)
index <- sample(1:284807, 284807)
train <- scale_data[index[1:floor(284807*0.66)],]
test <- scale_data[index[(floor(284807*0.66) + 1): 284807],]
```

We assure that the samples belonging to class 1 are equally distributed in the train and test set.

```r
sum(train$Class == 1)
```

```
## [1] 328
```

```r
sum(test$Class == 1)
```

```
## [1] 164
```

Now we try different approaches testing the performances of different models: in particular we will focus on the logistic regression, the LDA and the QDA. We tried also an approach with the KNN but it was unfeasible for our task due to the extremely computational cost of this method.

For what concerns this section we will fit our models considering all the variables and looking if one model gives significant better results.

Logistic Regression

Our first trial is a simple logistic regression considering all the feature of the transactions.

```r
genreg <- glm(Class~., data = train, family = binomial)
```

By looking at the `summary` of the regression given by R we noticed that not all the variables are significant for our purpose: almost every feature indeed has a p-value higher than 5%.

Now we predict the probabilities to belong to class 1 of the samples in the train set and we build the ROC curve: then we set the decision threshold depending on the number of false negatives in the training set. With this procedure we leave the test set only for the last step of model evaluation, and it remains unseen by the model until the computation of the accuracies.

```
predicted.prob.train <- predict(genreg, train, type="response")
roc.bin <- roc(train$Class, predicted.prob.train)
th <- coords(roc.bin, "best", ret = "threshold")
```

Now, in order to validate the model, we use the test set for looking the behaviour of the model on new data.

Then we look at the confusion matrix considering the best threshold given by the `roc` function in R, where we consider a threshold better than another if it gives a smaller number or false negatives.

```
predicted.prob1 <- predict(genreg, test, type="response")
predicted.values1 <- rep(0, 96835)
predicted.values1[predicted.prob1 > th$threshold] <- 1
table(predicted.values1, test$Class)
```

```
##
## predicted.values1     0     1
##                 0 95330    23
##                 1  1341   141
```

Nevertheless this model is very basic it gives good performance on the False Negatives, since 86% of fraudulent transactions are classified as fraudulent. However, due to the unbalanceness of the dataset, there is an issue in the number of False Positive: even if 98% of the good transactions are in the class '0' indeed the probability that a transaction is really fraudulent if it is in the class '1' is $141/(1341+141) = 0.09$.

For our purpose the false negatives are the fraudulent transactions that the bank allows, while the false positives are basically the customers that the bank losts: in what follows we will try to find a model that minimize both the false positives and the false negatives giving priority to this last class since for a bank is very dangerous to have fraudulent transactions.

Gaussian Regression

Now we proceed in a similar way but assuming that the samples come from different distributions, assuming first that they are normally distributed and then that they follow a Poisson probability.

```
genreg.gau <- glm(Class~., data = train, family = gaussian)

predicted.prob.train <- predict(genreg.gau, train, type="response")
predicted.prob2 <- predict(genreg.gau, test, type="response")
```

```
roc.gau <- roc(train$Class, predicted.prob.train)
```

Wew reason in a similar way computing the best threshold on the train set and then verifying what is the performance on the test set.

```
th <- coords(roc.gau, "best", ret = "threshold")
predicted.values2 <- rep(0, 96835)
predicted.values2[predicted.prob2 > th$threshold] <- 1
table(predicted.values2, test$Class)
```

```
##
## predicted.values2     0     1
##                 0 93890    20
##                 1  2781   144
```

This performance of this model looks similar to the first logistic regression, since the false negative decrease of three units, while the false positives are twice than before. ## Poisson Regression

```
genreg.poi <- glm(Class~., data = train, family = poisson)
predicted.prob.train <- predict(genreg.poi, train, type="response")
```

```
roc.poi <- roc(test$Class, predicted.prob.train)
th <- coords(roc.poi, "best", ret = "threshold")
```

```
th <- coords(roc.poi, "best", ret = "threshold")
predicted.prob3 <- predict(genreg.poi, test, type="response")
predicted.values3 <- rep(0, 96835)
predicted.values3[predicted.prob3 > th$threshold] <- 1
table(predicted.values3, test$Class)
```

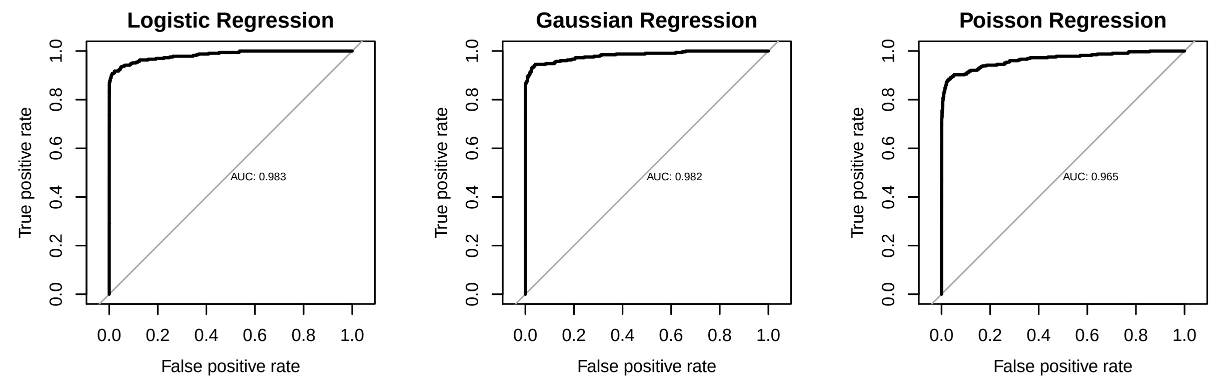
```
##
## predicted.values3     0     1
##                 0 93843    25
##                 1  2828   139
```

Even in this case we see that this model gives results similar to those we obtained before.

ROC curves

Here, in order to further compare these model we confront the ROC curves and the Precision - Recall curves of the models.

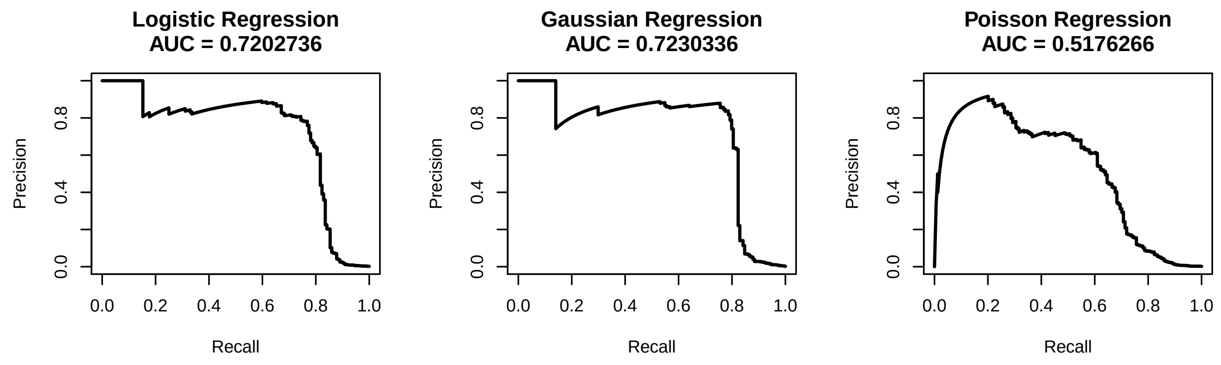
```
par(mfrow=c(1,3))
plot(roc.bin, print.auc=TRUE, legacy.axes=TRUE,
     xlab="False positive rate", ylab="True positive rate", main = 'Logistic Regression')
plot(roc.gau, print.auc=TRUE, legacy.axes=TRUE,
     xlab="False positive rate", ylab="True positive rate", main = 'Gaussian Regression')
plot(roc.poi, print.auc=TRUE, legacy.axes=TRUE,
     xlab="False positive rate", ylab="True positive rate", main = 'Poisson Regression')
```



Precision - Recall curves

```r
pr.curve1 <- pr.curve(predicted.prob1, weights.class0 = test$Class, curve = TRUE)
pr.curve2 <- pr.curve(predicted.prob2, weights.class0 = test$Class, curve = TRUE)
pr.curve3 <- pr.curve(predicted.prob3, weights.class0 = test$Class, curve = TRUE)

par(mfrow=c(1,3))
plot(pr.curve1, color = 1, lwd = 2, main = 'Logistic Regression')
plot(pr.curve2, color = 1, lwd = 2, main = 'Gaussian Regression')
plot(pr.curve3, color = 1, lwd = 2, main = 'Poisson Regression')
```



After have compared the ROC curves and the Precision - Recall curves we see that these regressions give similar results and there is not a predominant model.

Linear Discriminant Analysis (LDA)

We built also a LDA classifier and we noticed that this model gives a much lower number of false positive than before, but the number of misclassified fraudulent transactions is much bigger: in what follows we will focus on the logistic regression since we prefer to minimize the false negative.

```r
lda.fit <- lda(Class~., data = train)

lda.pred <- predict(lda.fit, test)

table(lda.pred$class, test$Class)
```

```
##
##          0     1
##   0 96654    44
##   1    17   120
```

Quadratic Discriminant Analysis (QDA)

We tried also a QDA, but it does not seem the right approach to solve out problem.

```r
qda.fit <- qda(Class~., data = train)

qda.pred <- predict(qda.fit, test)

table(qda.pred$class, test$Class)
```

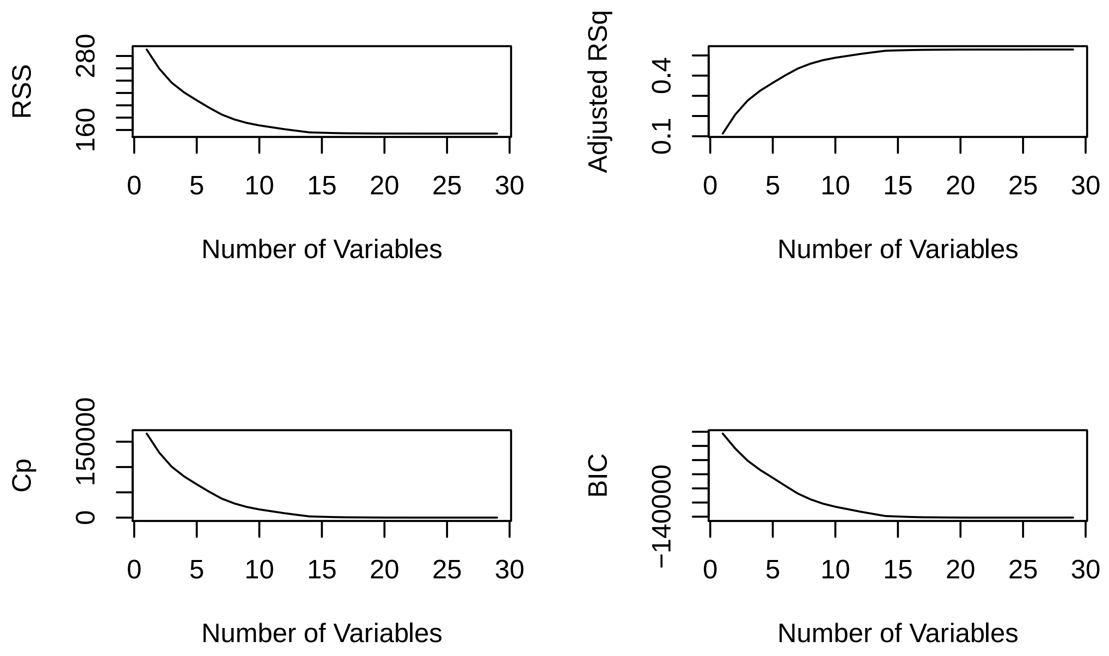
```
##
##          0     1
##   0 94362    27
##   1  2309   137
```

Variable Selection

In this section we try to identify what variables are really important for our classification: our first attempt is to consider subsets of different dimension and for each model we confront the values of some metrics on these regressions. In particular we consider the RSS, the Adjusted R2, the Cp and the BIC.

```r
regfit.full <- regsubsets(Class~., nvmax = 29, data = train)
reg.summary <- summary(regfit.full)
```

```r
par(mfrow=c(2,2))
# plot 1
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
# plot 2
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
# plot 3
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
# plot 4
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
```



Looking at the plots generated by the R function `regsubsets` it seems that adding new variables give better performances, since all the metrics improve. However we noticed that the model with 14 variable has a low value of RSS, Cp and BIC and an high value of Adjusted R2, and adding more variables the improvement is not significant. In particular the model with 14 variables reaches the 98% of the best value of the criteria. So we look how this model behaves since for equal performances we prefer simpler models.

```
prop <- c(prop,reg.summary$rss[which.min(reg.summary$rss)]/reg.summary$rss[14])
prop <- c(prop,reg.summary$adjr2[14]/reg.summary$adjr2[which.max(reg.summary$adjr2)])
prop <- c(prop,(reg.summary$cp[14] -reg.summary$cp[which.max(reg.summary$cp)])/
  (reg.summary$cp[which.min(reg.summary$cp)] - reg.summary$cp[which.max(reg.summary$cp)]))
prop <- c(prop,(reg.summary$bic[14] -reg.summary$bic[which.max(reg.summary$bic)])/
  (reg.summary$bic[which.min(reg.summary$bic)] - reg.summary$bic[which.max(reg.summary$bic)]))
```

```
## [1] 0.9873913 0.9887106 0.9857270 0.9811403
```

```
coef(regfit.full, 14)
```

```
##  (Intercept)           V1           V2           V3           V4           V5
##  0.001701383 -0.003995179  0.003701485 -0.007716801  0.005485290 -0.003760756
##           V7           V9          V10          V11          V12          V14
## -0.007444340 -0.004037319 -0.008891220  0.006430348 -0.010763529 -0.012550739
##          V16          V17          V18
## -0.008064613 -0.013389685 -0.004574911
```

```
reg.14 <- glm(Class~+V1+V2+V3+V4+V5+V7+V9+V10+V11+V12+V14+V16+V17+V18, data = train )
```

```
predicted.prob.train <- predict(reg.14, train, type="response")
roc.out <- roc(train$Class, predicted.prob.train)
```

```
th <- coords(roc.out, "best", ret = "threshold")
predicted.prob <- predict(reg.14, test, type="response")
predicted.values <- rep(0, 96835)
predicted.values[predicted.prob > th$threshold] <- 1
table(predicted.values, test$Class)
```

```
##
## predicted.values     0     1
##                0 94612    24
##                1  2059   140
```

The obtained model does not give better results in terms of sensitivity and specificity, so prefer to we focus in other variable selection techniques.

Now we will proceed first with a backward selection based on the p-value, then we will try to minimize the AIC, and finally we will try to implicitly select variables with a L1 regularization.

Step backward depending on p-value

Starting from the summary pof the first logistic regression, we drop at each step the variable with highest p-value until all the variables are reasonably significant. Proceeding in this way we drop 10 variables obtaining the following generalized linear model:

```
genreg.p<-glm(Class ~ .-V25 -V26 -V16 -V24 -V7 -V15 -V3 -V11 -V6 -V2,
              data = train, family = binomial)
```

As we have done before we now look at the confusion matrix on the test set:

```
##
## predicted.values     0     1
##                0 94251    21
##                1  2420   143
```

Dropping variables depending on the p-value it seems that the performance of the model does not improve, so we try to select the features looking at the AIC. Anyway we noticed that this model has less false negative than the same model without any sort of variable selection.

Step forward (AIC)

We firstly try with a step forward variables selection with 10 variables choosing at each step the variable that gives the model with smallest AIC.

```
biggest <- formula(glm(Class~., data = train, family = binomial))
step(glm(Class~+1, data = train, family = binomial),
     steps = 10, scope = biggest, direction = 'forward')
```

With this procedure we obtain the following variables:

```
genreg_step <- glm(formula = Class~V14+V10+V4+V3+V16+V8+V13+V21+V22+V23,
                   family = binomial, data = train)
```

Confronting the two previous model we see that a lot of variables that we consider in this case were taken into account also in the model with the p-value selection.

Looking now at the summary of this model we notice that all the variables are highly significant.

```
summary(genreg_step)
```

```
##
## Call:
## glm(formula = Class ~ V14 + V10 + V4 + V3 + V16 + V8 + V13 +
##     V21 + V22 + V23, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.0820  -0.0280  -0.0180  -0.0116   4.2273
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.70913    0.16467 -52.888  < 2e-16 ***
## V14         -0.69512    0.03839 -18.108  < 2e-16 ***
## V10         -0.54726    0.06382  -8.574  < 2e-16 ***
## V4           0.97673    0.07994  12.219  < 2e-16 ***
## V3           0.10238    0.05331   1.921  0.05478 .
## V16         -0.23564    0.05486  -4.295 1.74e-05 ***
## V8          -0.22538    0.02839  -7.940 2.02e-15 ***
## V13         -0.42366    0.09681  -4.376 1.21e-05 ***
## V21          0.26343    0.04946   5.326 1.00e-07 ***
## V22          0.44582    0.11210   3.977 6.98e-05 ***
## V23         -0.10143    0.03220  -3.150  0.00164 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4821.7  on 187971  degrees of freedom
## Residual deviance: 1424.4  on 187961  degrees of freedom
## AIC: 1446.4
##
## Number of Fisher Scoring iterations: 11
```

```
##
## predicted.values     0     1
##                0 94624    23
##                1  2047   141
```

In this case the number of False Negative is smaller than before, but the False Positive are almost twice.

Step backward (AIC)

In an a similar way we now try to select the variables reasoning backward and dropping 10 variables.

```r
genreg_step_back <- glm(formula = Class ~ V1+V4+V5+V6+V8+V9+V10+V12+V13+V14+V17
                        +V19+V20+V21+V22+V23+V27+V28+Amount)
```

```r
predicted.prob<- predict(genreg_step_back, test, type="response")
th <- coords(roc.out, "best", ret = "threshold")
predicted.values<- rep(0, 96835)
predicted.values[predicted.prob > th$threshold] <- 1
table(predicted.values, test$Class)
```

```
##
## predicted.values     0     1
##                0 95410    32
##                1  1261   132
```

By the previous models it seems that selecting variables manually doe snot improve the performances, so we try to select variables through a regularization.

Regularization

Here we try to implicitly select features with a L1 regularization: this type of regularization indeed focuses on models with a high number of null coefficient.

We start by trying with a lambda = 0.2:

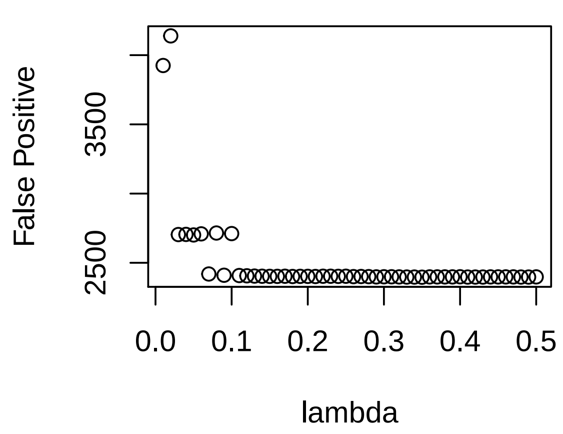
```r
ridge.mod <- glmnet(subset(train, select = -Class), train$Class, alpha=0, lambda = 0.2)
```

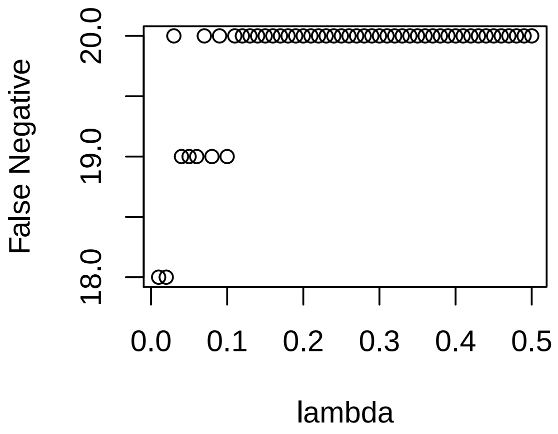
```r
th.ridge <- coords(roc.ridge, "best", ret = "threshold")
predicted.prob.ridge <- predict(ridge.mod, as.matrix(subset(test, select = -Class)),
                                type="response")
predicted.values.ridge <- rep(0, 96835)
predicted.values.ridge[predicted.prob.ridge > th.ridge$threshold] <- 1
table(predicted.values.ridge, test$Class)
```

```
##
## predicted.values.ridge     0     1
##                      0 94269    20
##                      1  2402   144
```

Now we tune the hyperparameter lambda of the L1 regression by fitting each regularized model searching for the best value lambda in `grid <- 1:100/100`. After haved fitted those model we confront the number of False Positive and False Negative:

```r
par(mfrow=c(1,2))
plot(grid, FP, ylab = 'False Positive', xlab = 'lambda')
plot(grid, FN, ylab = 'False Negative', xlab = 'lambda')
```





The same model without regularization gave 23 false negatives, while this new model gives only 20 of these misclassifications, so it seems that regularizing the logistic regression the model better recognizes if a fraudulent transactions is in class 1. Now we try to combine a regularization L1 with a variable selection in order to decrease both the misclassifications.

As we have done previously we drop 10 variables looking at the p-value and we choose the hyperparameter lambda equal to 0.2.

```
ridge.mod.small <- glmnet(subset(train, select = -V25-V26-V16-V24-V7-V15-V3-V11-V6-V2
                   -Class),train$Class, alpha=0, lambda = 0.2)

pred.prob.ridge.train <- predict(ridge.mod.small, as.matrix(subset(train,
                   select =-V25-V26-V16-V24-V7-V15-V3-V11-V6-V2-Class)),type="response")
```

```
roc.ridge <- roc(train$Class, pred.prob.ridge.train)
```

```
th.ridge <- coords(roc.ridge, "best", ret = "threshold")
pred.prob.ridge <- predict(ridge.mod.small, as.matrix(subset(test,
                   select=-V25-V26-V16-V24-V7-V15-V3-V11-V6-V2-Class)),type="response")
pred.values.ridge <- rep(0, 96835)
pred.values.ridge[pred.prob.ridge > th.ridge$threshold] <- 1

table(pred.values.ridge, test$Class)
```

```
##
## pred.values.ridge     0     1
##                 0 96670     0
##                 1     1   164
```

The obtained results are surprising since now only one sample is misclassified, and moreover all the fraudulent transactions are well classified. To validate the model we check also the performance on the train set, and the following confusion matrix shows that only one sample is in the wrong class.

```
##
```

```
## pred.values.train      0      1
##                 0 187643      0
##                 1      1    328
```