Introduction to Machine Learning

(NPFL054)

Homework 1

François Leroy, PhD student at CZU

2021-04-26

Contents

1. Multiple linear regression

1.1

Consider mpg as the target value. Perform a multiple linear regression using all the attributes except name. Print the results. Provide an interpretation of each hypothesis parameter in the model.

```r
# Perform the multiple linear regression
lm <-
  lm(mpg ~ ., data = subset(Auto, select = -name))
# Print the output
summary(lm)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = subset(Auto, select = -name))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

First of all, the adjusted $R^2 = 0.82$, which means that 82% of the variance of the data is explained by this models. This is a very trustful model.

Hereafter, I will talk only about the covariates that have a significant influence (*i.e.* $p - value \leq 0.05$) on the `mpg` variable (*i.e.* rows with an asterix such as `displacement`, `weight`, `year` and `origin`):

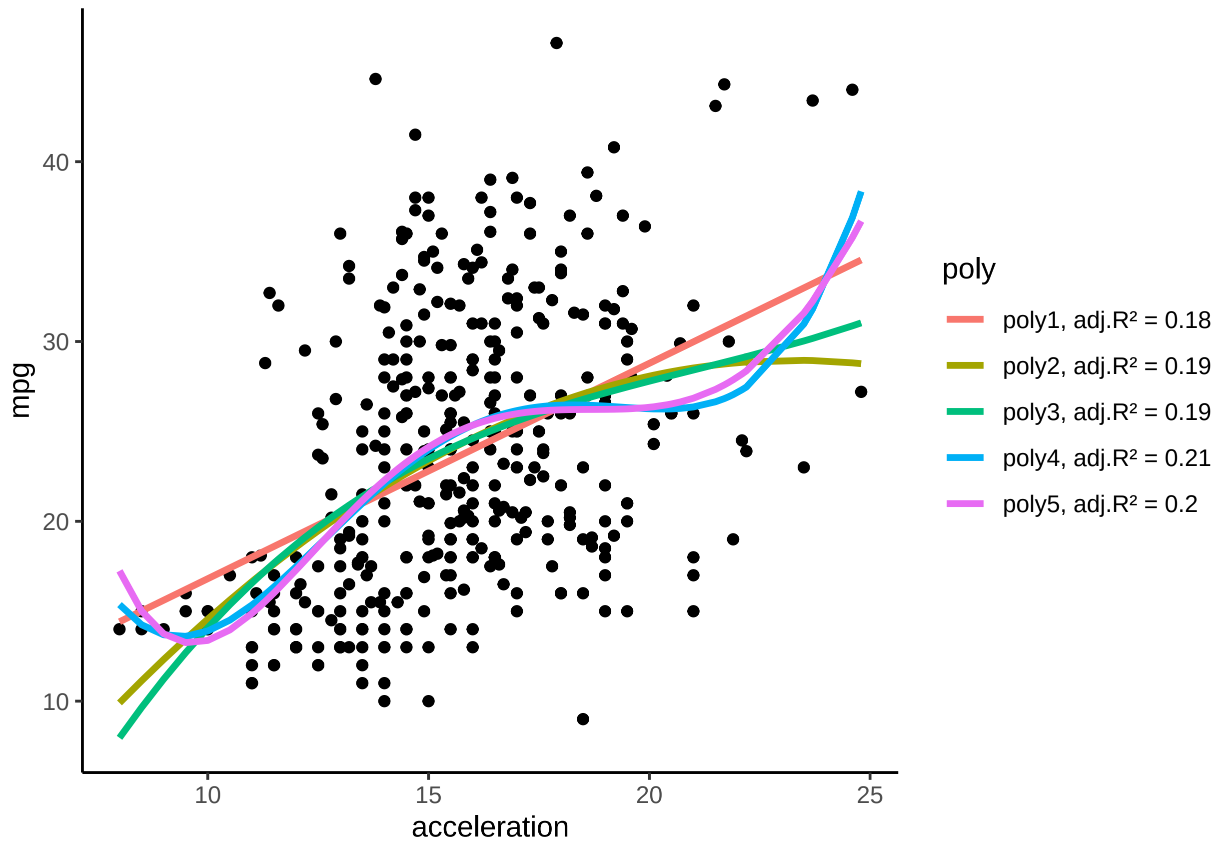
- The miles per galon unit (*i.e.* `mpg`) expresses the fuel economy of a vehicle. Thus, when the coefficient of the `lm` is negative, it means that the vehicle will tend to go less far with a unit of fuel. Here, this is the case for the `weight` variable which means that a heavier vehicle will consume more fuel than a lighter one for the same distance travelled.
- The other significant relationships with the `displacement`, `year` and `origin` are positive which means that a more recent car, with a higher displacement volume and with a higher origin will tend to consume less fuel.

1.2

Perform polynomial regression to predict mpg using acceleration. Plot the polynomial fits for the polynomial degrees 1 to 5 and report the values of Adjusted R2.

```r
## Perform the 5 polynomial linear regression
for (i in 1:5){
  assign(paste0("fit", i),
         lm(mpg ~ poly(acceleration, i), data = subset(Auto, select = -name)))
}
## Plot them on a single plot
#### First merge the predicted values of mpg with the acceleration
Auto %>%
  select(acceleration) %>%
  cbind(poly1 = fit1$fitted.values,
        poly2 = fit2$fitted.values,
        poly3 = fit3$fitted.values,
        poly4 = fit4$fitted.values,
        poly5 = fit5$fitted.values) %>%
##### Then format the data for ggplot
  pivot_longer(cols = poly1:poly5,
               names_to = "poly",
               values_to = "mpg") %>%
  mutate(rsq = case_when(
    poly == "poly1" ~ round(summary(fit1)$adj.r.squared, digits = 2),
    poly == "poly2" ~ round(summary(fit2)$adj.r.squared, digits = 2),
    poly == "poly3" ~ round(summary(fit3)$adj.r.squared, digits = 2),
    poly == "poly4" ~ round(summary(fit4)$adj.r.squared, digits = 2),
    poly == "poly5" ~ round(summary(fit5)$adj.r.squared, digits = 2)
  )) %>%
  unite(poly, c("poly", "rsq"), sep = ", adj.R² = ") %>%
#### Now plot it
```

```
ggplot()+
geom_point(aes(acceleration, mpg), data = subset(Auto, select = -name))+
geom_line(aes(acceleration, mpg, color = poly), size = 1.2)+
theme_classic()
```



2. Develop a model to predict whether a given car gets high or low gas mileage

2.1

Create a binary attribute, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. Create a single data set containing both mpg01 and the other Auto attributes except mpg. Compute entropy of mpg01.

```
# Discretizing mpg
Auto$mpg01 <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
# Creating dataset with discrete mpg
classif_data <-subset(Auto, select = -mpg)
## Compute entropy
# First compute the proba of each class
p <- table(classif_data$mpg01) / length(classif_data$mpg01)
## Then compute the entropy
ent_mpg <- -sum(p * log2(p))
```

Here, the entropy of $mpg01 = 1$ which is totally expected because `mpg` has been discretized by its median, which means that 50% of `mpg` values are above (and thus equal to 1) and 50% are below (and thus equal to 0). Consequently, the probability of `mpg01` being 0 or 1 are both equal to `0.5` and the entropy is maximum.

2.2

Split the data into a training set and a test set 80:20

```
set.seed(123) # to reproduce the results
## Training dataset
train <- classif_data[sample(nrow(classif_data), nrow(classif_data)*0.8),]
```

```r
## Test dataset
test <- classif_data[sample(nrow(classif_data), nrow(classif_data)*0.2),]
```

2.3

Make a trivial classifier (without using the features) and evaluate it on the test set. Compute its accuracy.

```r
# Most frequent class of the training data
table(train$mpg01)
```

```
##
##   0   1
## 154 159
```

The most frequent class of the training dataset is 1. Thus, the trivial classifier will always predict 1.

```r
## Confusion matrix of the trivial classifier
table(test$mpg01, rep(1, nrow(test)))
```

```
##
##      1
##   0 42
##   1 36
```

```r
## Compute accuracy
accuracy <- 36/nrow(test)
```

Here, the accuracy = 0.4615385, which means that only 46% of the target values will be classified correctly.

2.4

Perform logistic regression on train in order to predict mpg01 using all the features except name. Use a threshold of 0.5 to cut the predicted probabilities to make class predictions.

2.4.1 Compute the training error rate.

```
## Learn the model
logit_reg <- glm(mpg01 ~ ., family = binomial(link = "logit"),
                 data = subset(train, select = -name))
## Use the model to predict on the learning dataset
train_prediction <- predict(logit_reg, type = "response")
## Transform the prediction into 0/1
train_prediction <- ifelse(train_prediction > .5, 1, 0)
## Confusion matrix
cm_train_logit <- table(train$mpg01, train_prediction)
## Training error rate
train_error_rate <- 1 - sum(diag(cm_train_logit))/sum(cm_train_logit)
```

The training error rate of this logistic regression is 0.0830671.

2.4.2 Produce a confusion matrix comparing the true test target values to the predicted test target values. Compute the test error rate, Sensitivity, and Specificity.

```
## Use the model to predict on the test dataset
test_prediction <- predict(logit_reg, type = "response", newdata = test)
## Transform the prediction into 0/1
test_prediction <- ifelse(test_prediction > .5, 1, 0)
## Confusion matrix
cm_test_logit <- table(test$mpg01, test_prediction)
## Test error rate:
test_error_rate <- 1 - sum(diag(cm_test_logit))/sum(cm_test_logit)
## Sensitivity
sensi <- cm_test_logit[2,2]/(cm_test_logit[2,2] + cm_test_logit[2,1])
## Specificity
speci <- cm_test_logit[1,1]/(cm_test_logit[1,1]+cm_test_logit[1,2])
```

The training error rate, sensitivity and specificity are respectively equal to 0.1153846, 0.9166667 and 0.8571429.

2.4.3 Provide an interpretation of each hypothesis parameter in the model.

The **test error rate** of 0.1153846 means that around 12% of the predictions in both class are incorrect, *i,e.* $mpg = 1$ classified as 0 or $mpg = 0$ classified as 1.

The **sensitivity** equal to 0.9166667 means that only 92% of the example actually equal to 1 will correctly be classified as 1.

On the other hand, the **specificity** equal to 0.8571429 means that only 86% of the examples equal to 0 will be corretly classified in the class 0.

2.5

In the previous exercise you used a threshold of 0.5. Re-run the experiment from the previous exercise with different threshold values, namely 0.1, 0.3, 0.6, 0.9.

```r
thresholds <- c(0.1, 0.3, 0.5, 0.6, 0.9)
for(i in 1:length(thresholds)){
  ## Use the model to predict on the test dataset
  test_prediction <- predict(logit_reg, type = "response", newdata = test)
  ## Transform the prediction into 0/1
  test_prediction <- ifelse(test_prediction > thresholds[i], 1, 0)
  ## Confusion matrix
  cm_test_logit <- table(test$mpg01, test_prediction)
  ## Precision
  prec <- cm_test_logit[2,2]/(cm_test_logit[2,2] + cm_test_logit[1,2])
  ## Recall
  recall <- cm_test_logit[2,2]/(cm_test_logit[2,2] + cm_test_logit[2,1])
  ## F-measure
  fmeasure <- 2*(prec*recall)/(prec+recall)
  #Print
  cat(paste0("-"," For threshold = ", thresholds[i],
             " , Precision = ", prec,
             " , Recall = ", recall,
             " and F-measure = ", fmeasure), sep = "\n")
}
```

- For threshold = 0.1 , Precision = 0.765957446808511 , Recall = 1 and F-measure = 0.867469879518072
- For threshold = 0.3 , Precision = 0.772727272727273 , Recall = 0.944444444444444 and F-measure = 0.85
- For threshold = 0.5 , Precision = 0.846153846153846 , Recall = 0.916666666666667 and F-measure = 0.88

- For threshold = 0.6 , Precision = 0.833333333333333 , Recall = 0.833333333333333 and F-measure = 0.833333333333333
- For threshold = 0.9 , Precision = 0.961538461538462 , Recall = 0.694444444444444 and F-measure = 0.806451612903226

We can see the that recall (*i.e.* precision) is decreasing with the increasing threshold. It means that with a low threshold and a high recall, most of the mpg = 1 will be correctly classified into class 1. For instance, for threshold = 0.1, recall = 1, which means that all the actual mpg = 1 are classified as 1.

On the other hand, we can see that the the precision has the opposite behavior to the recall: the precision increases with the increasing threshold. A higher precision means that the predicted class = 1 will contain mostly actual values of mpg = 1 and will contain less misclassifications.

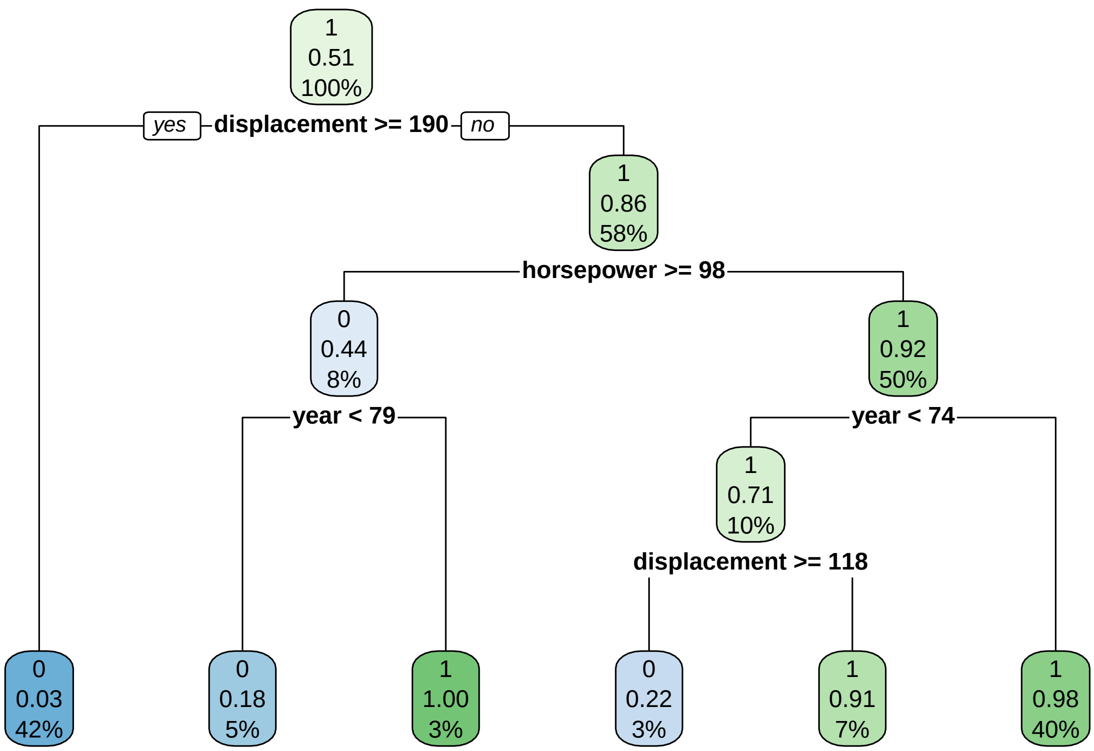
We can see that the F-score, which takes into account both precision and recall, is the highest for threshold = 0.5. Thus, it seems that we should prefer this threshold of 0.5 over the others because it optimizes the recall and precision metrics.

2.6

Perform decision tree algorithm on train to predict mpg01 using all the features except name.

2.6.1 Create a plot of the tree. Compute the training error rate. Compute the test error rate.

```
## Learn the decision tree
dt <- rpart(as.factor(mpg01) ~ ., data = subset(train, select = -name))
## Plot it
rpart.plot(dt)
```



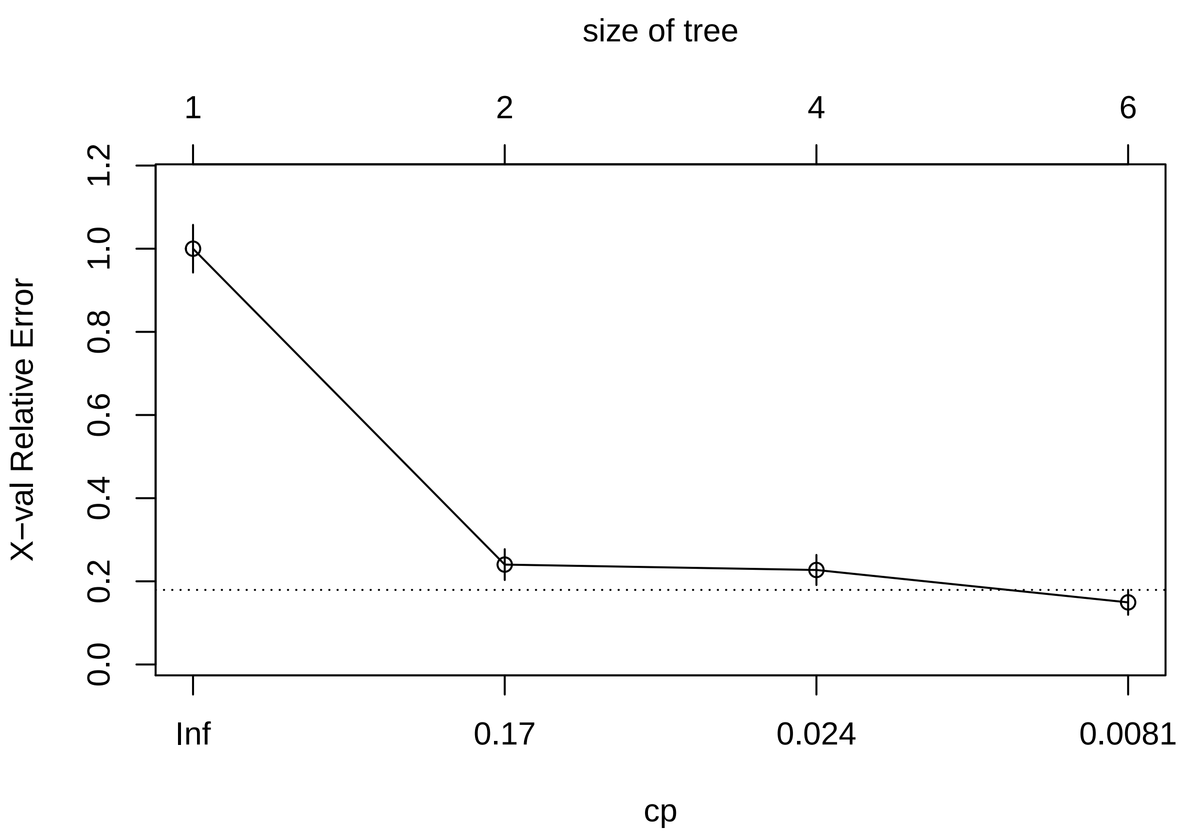

```
## Training data
predtrain <- predict(dt, type = "class")
cmtrain <- table(train$mpg01, predtrain)
trainerrorrate <- 1 - (sum(diag(cmtrain))/sum(cmtrain))
```

```r
## Test data
predtest <- predict(dt, type = "class", newdata = test)
cmtest <- table(test$mpg01, predtest)
testerrorrate <- 1 - (sum(diag(cmtest))/sum(cmtest))
```

The training error rate = 0.0447284 and the test error rate = 0.1025641.

2.6.2 Tune the cp parameter. Choose the best value of cp, and evaluate your model again. What is the best value of cp? Why? Explain it explicitly. Compute the accuracy of the model with your best cp.

```r
tunedDT <- rpart(as.factor(mpg01) ~ ., data = subset(train, select = -name),
                 cp = 0.004)
plotcp(tunedDT)
```



CP	nsplit	rel error	xerror	xstd
0.8051948	0	1.0000000	1.0000000	0.0574336
0.0357143	1	0.1948052	0.2402597	0.0370905
0.0162338	3	0.1233766	0.2272727	0.0362046
0.0040000	5	0.0909091	0.1493506	0.0299757

The most optimal value of cp (*i.e.* complexity parameter) is the one minimizing the `xerror`, which is the generalization error (*i.e.* true error or expected error) that estimates the probability of error on an other potential dataset of distribution D.

The accuracy of the new decision tree is 0.8974359.

2.7

Compare the best models trained in the previous exercises 4., 5., and 6. Which one could be considered as the best?

Method	Error rate
Logistic Regression	0.1153846
Decision Tree	0.1025641

Using the test error rate as the final metric to choose the best modeling method, it seems that the decision tree has a slightly lower error rate and should be considered the best.