Overfitting

Carolina Alves de Lima Salge

3/29/2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1
```

```r
churn <- read_csv("churn.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   SeniorCitizen = col_double(),
##   tenure = col_double(),
##   MonthlyCharges = col_double(),
##   TotalCharges = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```r
# transform categories to numbers
churn <- churn %>%
  mutate(genderN = case_when(
    gender == "Male" ~ 1,
    gender == "Female" ~ 0
    )) %>%
  mutate(PartnerN = case_when(
    Partner == "Yes" ~ 1,
    Partner == "No" ~ 0
    )) %>%
  mutate(DependentsN = case_when(
    Dependents == "Yes" ~ 1,
    Dependents == "No" ~ 0
    )) %>%
  mutate(PhoneServiceN = case_when(
    PhoneService == "Yes" ~ 1,
    PhoneService == "No" ~ 0
    )) %>%
  mutate(MultipleLinesN = case_when(
    MultipleLines == "Yes" ~ 1,
    MultipleLines == "No" ~ 0,
    MultipleLines == "No phone service" ~ 0
    )) %>%
  mutate(InternetServiceN = case_when(
    InternetService == "Fiber optic" ~ 2,
    InternetService == "DSL" ~ 1,
    InternetService == "No" ~ 0
    )) %>%
  mutate(OnlineSecurityN = case_when(
    OnlineSecurity == "Yes" ~ 1,
    OnlineSecurity == "No" ~ 0,
    OnlineSecurity == "No internet service" ~ 0
    )) %>%
  mutate(OnlineBackupN = case_when(
    OnlineBackup == "Yes" ~ 1,
    OnlineBackup == "No" ~ 0,
    OnlineBackup == "No internet service" ~ 0
    )) %>%
  mutate(DeviceProtectionN = case_when(
    DeviceProtection == "Yes" ~ 1,
```

```
    DeviceProtection == "No" ~ 0,
    DeviceProtection == "No internet service" ~ 0
    )) %>%
  mutate(TechSupportN = case_when(
    TechSupport == "Yes" ~ 1,
    TechSupport == "No" ~ 0,
    TechSupport == "No internet service" ~ 0
    )) %>%
  mutate(StreamingTVN = case_when(
    StreamingTV == "Yes" ~ 1,
    StreamingTV == "No" ~ 0,
    StreamingTV == "No internet service" ~ 0
    )) %>%
  mutate(StreamingMoviesN = case_when(
    StreamingMovies == "Yes" ~ 1,
    StreamingMovies == "No" ~ 0,
    StreamingMovies == "No internet service" ~ 0
    )) %>%
  mutate(ContractN = case_when(
    Contract == "Month-to-month" ~ 0,
    Contract == "One year" ~ 1,
    Contract == "Two year" ~ 1
    )) %>%
  mutate(PaperlessN = case_when(
    PaperlessBilling == "Yes" ~ 1,
    PaperlessBilling == "No" ~ 0
    )) %>%
  mutate(PaymentN = case_when(
    PaymentMethod == "Electronic check" ~ 0,
    PaymentMethod == "Mailed check" ~ 0,
    PaymentMethod == "Bank transfer (automatic)" ~ 1,
    PaymentMethod == "Credit card (automatic)" ~ 1
    )) %>%
  mutate(ChurnN = case_when(
    Churn == "Yes" ~ 1,
    Churn == "No" ~ 0
    ))

# only select numeric variables
df <- churn %>% dplyr::select(Churn, SeniorCitizen, tenure,
                              MonthlyCharges, TotalCharges, genderN:PaymentN)

# drop missing values NAs
df1 <- drop_na(df)

# transform target into a factor
df1$Churn <- as.factor(df1$Churn)

set.seed(12L) # set a starting seed to be able to get reproducible results

# partition data
trainIndex <- createDataPartition(df1$Churn, # target variable
                                  p = 0.8, # percentage that goes to training
```

```
                                  list = FALSE, # results will not be in a list
                                  times = 1) # number of partitions to create

churn_train <- df1[trainIndex, ] # data frame for training
churn_test <- df1[-trainIndex, ] # data frame for testing


# model induction
model <- train(Churn ~ .,
               data = churn_train, # use training set
               method = "glmnet",
                 # alpha and lambda paramters to try
                 tuneGrid = data.frame(alpha=1, # 1 is lasso 0 is ridge
                                       lambda=seq(0.0001,1))) # strength of penalty

# now predict outcomes in test set
p <- predict(model, churn_test, type = 'raw')

# add predictions to initial dataset
churn_test$pred_churn <- p

# how did we do? confusion matrix
confusionMatrix(data = churn_test$pred_churn,
                reference = churn_test$Churn,
                mode = "prec_recall",
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  924 171
##        Yes 108 202
##
##                Accuracy : 0.8014
##                  95% CI : (0.7796, 0.822)
##     No Information Rate : 0.7345
##     P-Value [Acc > NIR] : 2.974e-09
##
##                   Kappa : 0.4618
##
##  Mcnemar's Test P-Value : 0.0002058
##
##               Precision : 0.6516
##                  Recall : 0.5416
##                      F1 : 0.5915
##              Prevalence : 0.2655
##          Detection Rate : 0.1438
##    Detection Prevalence : 0.2206
##       Balanced Accuracy : 0.7185
##
##        'Positive' Class : Yes
##
```