Hands-on Machine Learning with R

Contents

5 Logistic Regression

When your response is binary, you cannot use linear reg. *Logistic regression* is the analagous alg for predicting binary outcomes.

5.1 Prerequisites

```r
# Create training (70%) and test (30%) sets for the
# rsample::attrition data.
set.seed(123)  # for reproducibility
churn_split <- initial_split(df, prop = .7, strata = "Attrition")
churn_train <- training(churn_split)
churn_test  <- testing(churn_split)
```

!!! missing library `broom` since `tidy()` is used later on.

We start with the attrition dataset, which has a Yes/No response, and we split it into a 70/30 % training/test split.

5.2 Why Logistic Regression?

Predicting the probability of a dichotomous event - yes/no - with linear regression would lead to unreasonable predicitons: negative probabilities, or probabilities greater than one. The sigmoidal shape of the logistic curve prevents this. There are many functions that restrict outputs to [0,1] for all inputs, and the logistic function is just one of them:

$$p(X) = \frac{e^a}{1+e^a}$$

Where a cxan be any linear transformation of predictors such as:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1}}{1+e^{\beta_0+\beta_1 X}}$$

here $p(X)$ is the probability that the outcome is positive, and the β parameters are the coefficients. In the limits the equation tends to zero and one, which is what we want. Rearranging the equation we start out by taking the odds of X = 1, which is the probability of X = 1 divided by the probability of X = 0

$$\frac{P(X)}{1-P(X)} = \frac{\frac{e^a}{1+e^a}}{1-\frac{e^a}{1+e^a}}$$

If you multiply the right side with $\frac{1+e^a}{1+e^a}$, it simplifies to e^a. Then just take the natural log of the whole thing:

$$\frac{P(X)}{1-P(X)} = a = \beta_0 + \beta_1 X$$

So yeah, applying the *logit transformation* to $P(X)$ gives us a linear transformation similar to the one from the simple regression model. This also allows us to have an intuitive interpretation of the model: the odds of attrition (left hand side) increase multiplicatively with e_1^β for every unit increase in X.

5.3 Simple Logistic Regression

We fit two logistic regressions using `glm()` since log.reg is a sub class of the generalised linear regression with `family = "binomial"`. One where the predictor is monthly income, the other where it is overtime.

```
model1 <- glm(Attrition ~ MonthlyIncome, family = "binomial", data = churn_train)
model2 <- glm(Attrition ~ OverTime, family = "binomial", data = churn_train)
```

The `glm()` function uses *maximum likelihood* estimation to find the best estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$

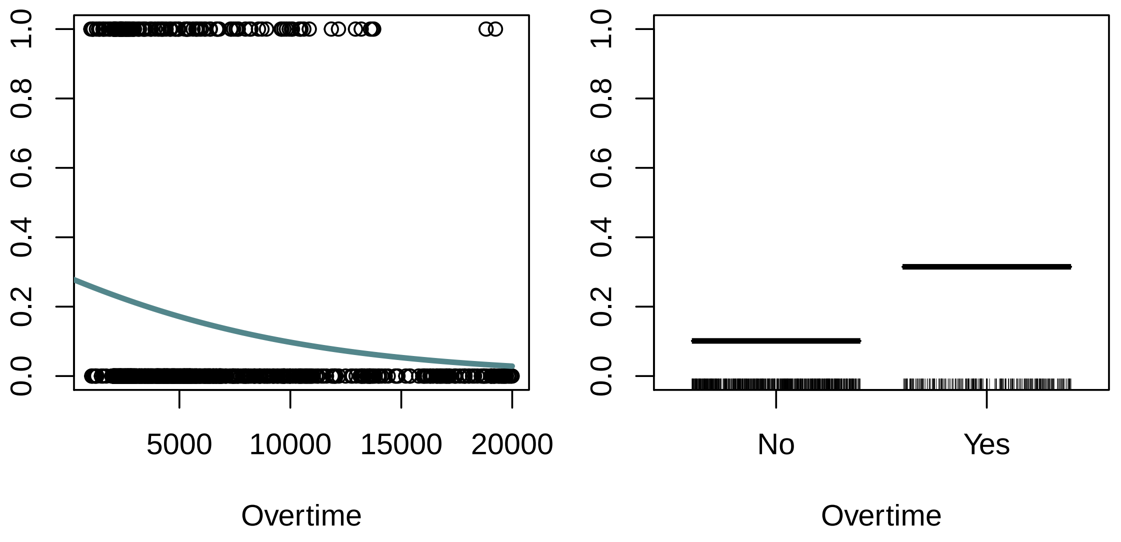

Figure 1: Predicted probabilities of attrition

Below are the coefficients for the models which are interpreted like so: The estimated coefficent $\hat{\beta}_1$ is -0.00013. which is negative. So an increase in monthly income is associated with a decrease in the probability of attrition. In the second model, those that work overtime are more likely to attrit than those that don't

```
broom::tidy(model1)
```

```
## # A tibble: 2 x 5
##   term           estimate std.error statistic       p.value
##   <chr>             <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   -0.924    0.155         -5.96 0.00000000259
## 2 MonthlyIncome -0.000130 0.0000264     -4.93 0.000000836
```

```
broom::tidy(model2)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    -2.18     0.122    -17.9  6.76e-72
## 2 OverTimeYes     1.41     0.176      8.00 1.20e-15
```

Using the exponential transformation for interpretation:

```
exp(coef(model1))
```

```
##   (Intercept) MonthlyIncome
##     0.3970771     0.9998697
```

```
exp(coef(model2))
```

```
## (Intercept) OverTimeYes
##   0.1126126   4.0812121
```

SO an increase in monthly income by one dollar, decreases (!!! error in book, says increase) multiplicatively the odds of attritting by 0.9999. And the odds of attritting increase by 4.08 times if an employe works overtime as opposed to not.

We can also use estimated standard errors to get confidence intervals for the coefficients.

```
confint(model1)
```

```
##                       2.5 %        97.5 %
## (Intercept)   -1.2267754960 -6.180062e-01
## MonthlyIncome -0.0001849796 -8.107634e-05
```

```
confint(model2)
```

```
##                 2.5 %    97.5 %
## (Intercept) -2.430458 -1.952330
## OverTimeYes  1.063246  1.752879
```

5.4 Mulitple Logistic Regression

Extending the simple logistic regression equation above to include more predictors is straightforward.

We can combine both predictors in model 3:

```
model3 <- glm(
  Attrition ~ MonthlyIncome + OverTime,
  family = "binomial",
  data = churn_train)
tidy(model3)
```

```
## # A tibble: 3 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -1.43     0.176         -8.11 5.25e-16
## 2 MonthlyIncome -0.000139 0.0000270     -5.15 2.62e- 7
## 3 OverTimeYes    1.47     0.180          8.16 3.43e-16
```

Both are highly significant. The figure below shows the predicted probabilities of attrition by income, which decreases as income goes up, but is much higher for perople working onvertime (blue) than not (pink). Overall there is little correlation between overtime and income as you can see on the right, with verry similar income distributions for both groups. Found the code here

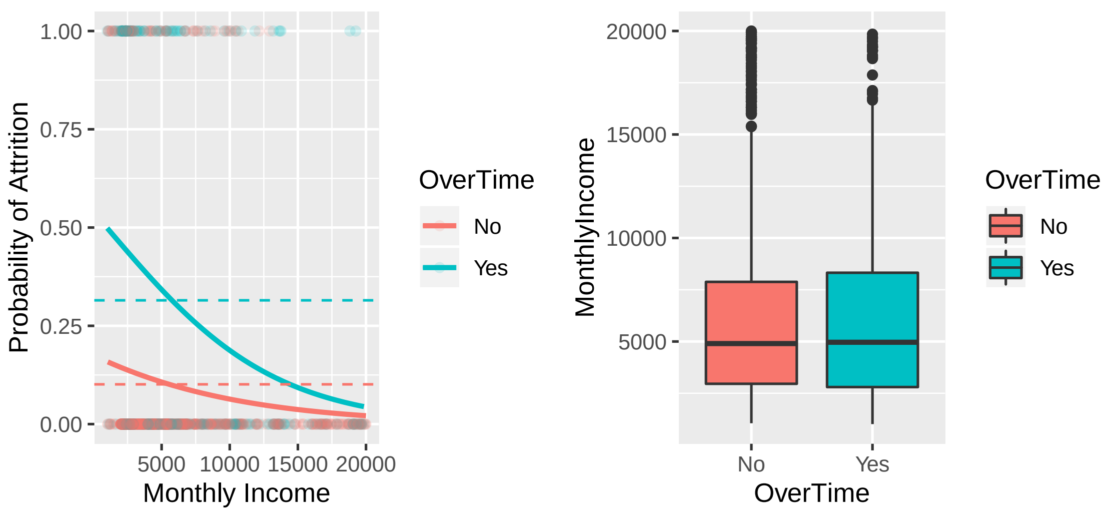

Figure 2: Predicted probabilities of attrition in multinomial logistic model

5.5 Assessing model accuracy

SO now we can use train and crossvalidation on all three models and compare their classification accuracy.

```r
set.seed(122)
cv_model1 <- train(
  Attrition ~ MonthlyIncome,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)


set.seed(123)
cv_model2 <- train(
  Attrition ~ MonthlyIncome + OverTime,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

set.seed(123)
cv_model3 <- train(
  Attrition ~ .,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

# extract out of sample performance measures
summary(
  resamples(
    list(
      model1 = cv_model1,
      model2 = cv_model2,
      model3 = cv_model3   )
  )
)$statistics$Accuracy
```

```
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## model1 0.8349515 0.8349515 0.8365385 0.8388478 0.8431373 0.8446602    0
## model2 0.8349515 0.8349515 0.8365385 0.8388478 0.8431373 0.8446602    0
## model3 0.8365385 0.8495146 0.8792476 0.8757893 0.8907767 0.9313725    0
```

So we have ten-fold crossvalidation and accuracy is measured for each of the ten resamples, and the summary then shows the distribution across the ten samples. !!! OK, so this is confusing first of all, because the models have now changed from the ones we were using before. And secondly, there is the odd result that models 1 and 2 have the exact same accuracy, not just on average, but the whole distribution. Which I don't really get. Aaah, see below.

But so the average accuracy rate with one or two predictors is 83.33%, but adding all the predictors in the dataset we get up to 87.58 % on average.

So we can look at the confustion matrices to get a better idea of the accuracy.

```
pred_class1 <- predict(cv_model1, churn_train)
confusionMatrix(
  data = relevel(pred_class1, ref = "Yes"),
  reference = relevel(churn_train$Attrition, ref = "Yes")
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes   0   0
##        No  166 864
##
##                Accuracy : 0.8388
##                  95% CI : (0.8149, 0.8608)
##     No Information Rate : 0.8388
##     P-Value [Acc > NIR] : 0.5207
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.8388
##              Prevalence : 0.1612
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

```
pred_class3 <- predict(cv_model3, churn_train)

confusionMatrix(
  data = relevel(pred_class3, ref = "Yes"),
  reference = relevel(churn_train$Attrition, ref = "Yes")
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes  93  25
##        No   73 839
##
##                Accuracy : 0.9049
##                  95% CI : (0.8853, 0.9221)
##     No Information Rate : 0.8388
##     P-Value [Acc > NIR] : 5.360e-10
##
##                   Kappa : 0.6016
##
##  Mcnemar's Test P-Value : 2.057e-06
##
##             Sensitivity : 0.56024
##             Specificity : 0.97106
##          Pos Pred Value : 0.78814
##          Neg Pred Value : 0.91996
##              Prevalence : 0.16117
##          Detection Rate : 0.09029
##    Detection Prevalence : 0.11456
##       Balanced Accuracy : 0.76565
##
##        'Positive' Class : Yes
##
```

So the first two models aren't predicting any attrition at all. Neither of them. So in both cases we have high specificity (100 % prediction of no attrition) and low (zero) sensitivity (zero % prediction of attrition). The full model only predicts 56 % of the attrition cases correctly (low specificity).

!!! We can see from the confusion matrix the "no informaiton rate", which is the proportion of attrition to no-attrition in the data is 83.33%, which is of course the same as the average accuracy in the first two models. this is not mentioned in the book, but would be useful.

So let's plot the ROC curve (receiver operating characteristic) to compare models 1 and 3. Using `prediction` and `performance` functions from the `ROCR` package to calculate the curves. we can see how model 3 dramatically shifts the curve to the top left - which is what we want. !!! A shame prediction and performance aren't explained in a bit more detail.

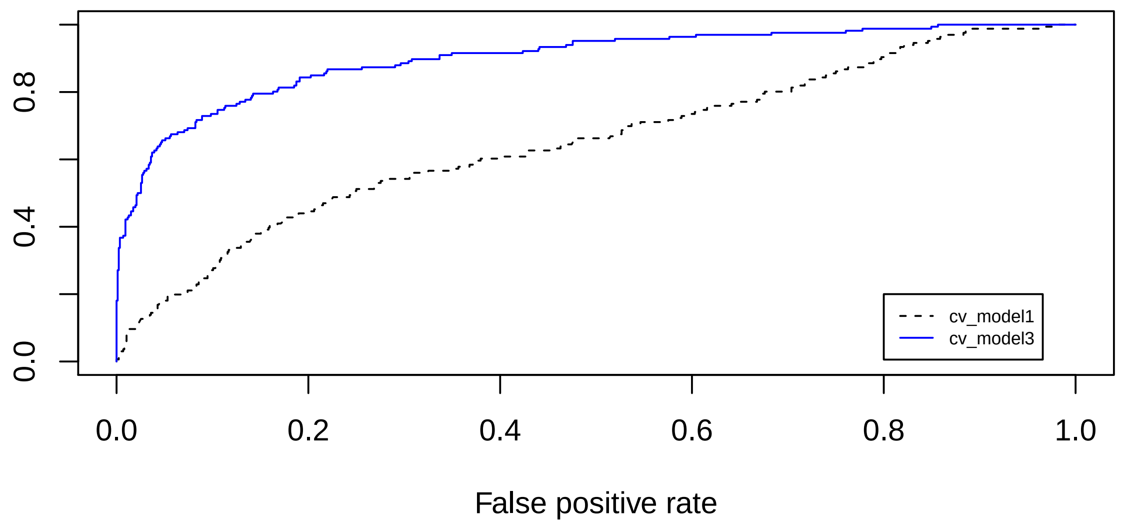

Figure 3: ROC curves for models 1 and 3

5.5.1 Partial Least Squares

Like in the OLS regression case we can see if reducning the dimensions improves the model by using PLS. Again on a 10-fold cross validation.

```r
# Perform 10-fold CV on a PLS model tuning the number of PCs to
# use as predictors
set.seed(123)
cv_model_pls <- train(
  Attrition ~ .,
  data = churn_train,
  method = "pls",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 16
)
cv_model_pls$bestTune
```

```
##    ncomp
## 14    14
```

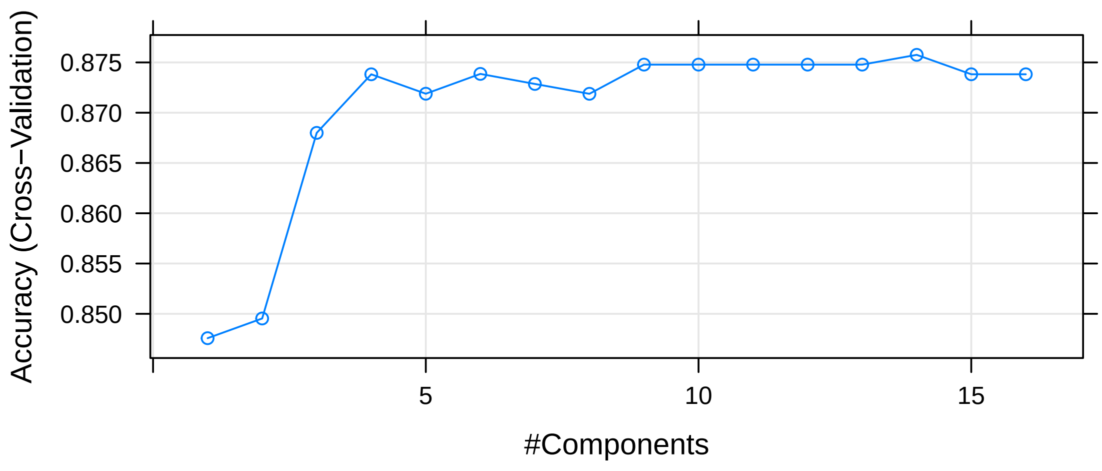

Figure 4: PLS tuning

!!! with this seed I get the error `Warning message: In fitFunc(X, Y, ncomp, Y.add = Y.add, center = center, ...) : No convergence in 100 iterations`. not sure what this means, but should be mentioned in the book, no?

5.6 Model concerns

Residual analysis is not so straightforward with outcomes of 0 and 1. But some attempts have been made. But they are not really explained in the book. Except for some pointers to look at pseudo-residuals and surrogate residuals, both of which can be implemented in R.

5.7 Feature interpretation

Similarly to the linear models, once we have our preferred model we can try to extract the most important features to see which ones have the most impact.

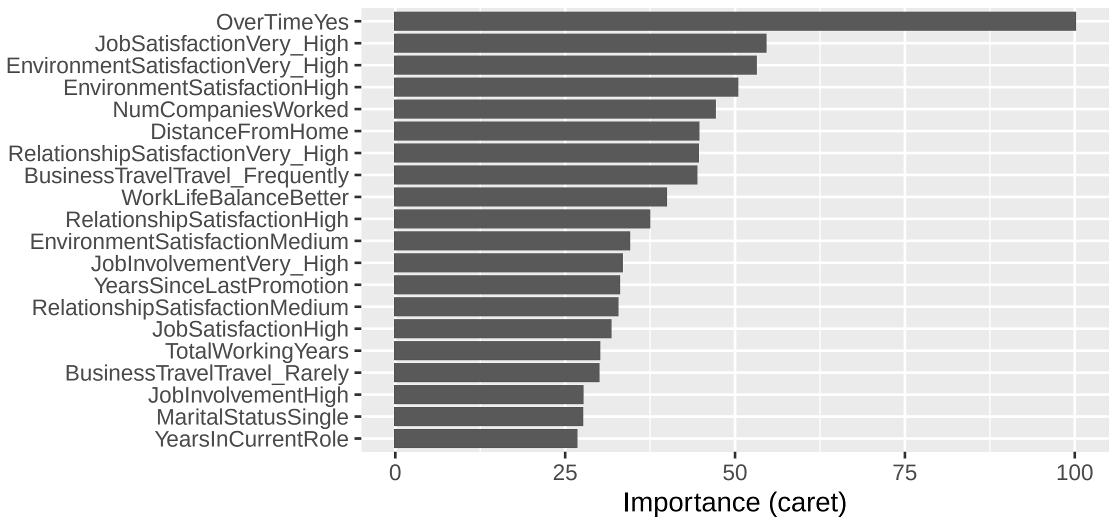

Figure 5: Main features in model 3

We can also draw partial dependence plots for the top four predictors. The code is a bit complicated here using the prediciton function `pred.fun` argument, but I think this is really just because of how the variable is coded. “We just need to write a function that computes the predicted class probability of interest averaged across all observations”. Otherwise the argument `prob = TRUE` also lets you easily make PDPs for classification problems like this. More explanationcan be found here.

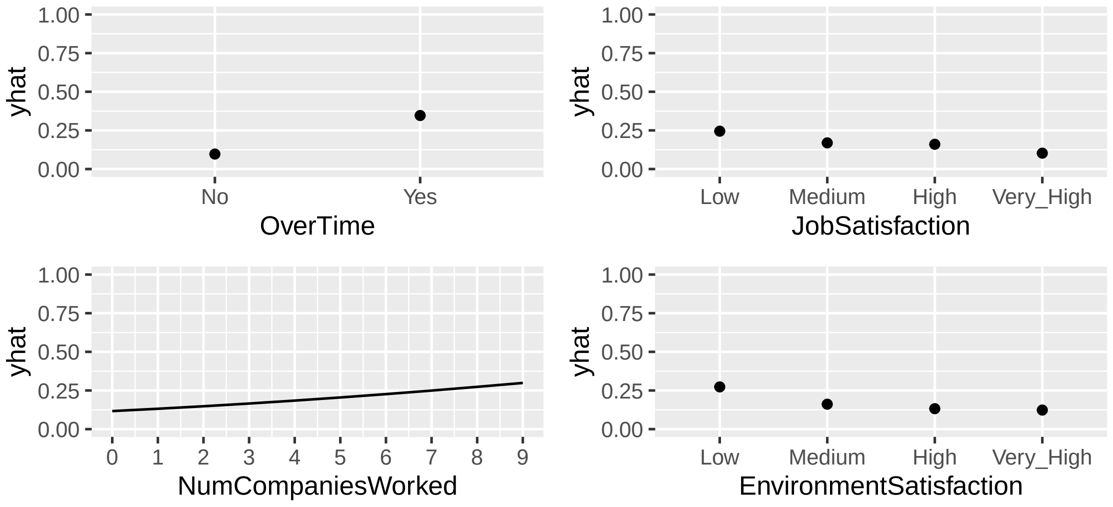

Figure 6: Main features in model 3

5.8 Final thoughts

So log reg has all the similar issues than linear does - limited to a linear relationship between the coefficients, the problem of multi-colinearity. And the problem that it only applies to dichotomous outcomes. Although multinomial classification is a solution to that, it comes with further assumptions and decreased accuracy of estimates. There are other models that work better for binary and multinomial classification, we’ll get to them now.