Logistic Regression in R

Data Literacy

Matthias Frühwirth

Institute for Retailing & Data Science

Recap and Intro

Last week

- Causal Inference
- DAGs & Diff-in-Diff
- Quarto Presentations

Today

- Logistic Regression
- Quarto Documents

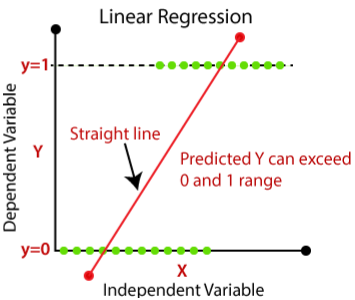
Intro to Logistic Regression (Logit)

Logistic Regression

- Can be used if outcome is a binary variable (e.g. 1 = Purchase Complete, 0 = Cart Abandoned)
- Generalized Linear Model (linear regression + non-linear link function, marginal effects are non-linear)
- Interested in modelling probabilities of outcomes:

  Q: *"How does making payment one-click affect the probability that the purchase is completed"? What about browsing time on the website?*
- Sits right at the intersection of econometrics and machine learning
- Widely used baseline for inference, prediction/classification with category outcomes

Graphical



As we change X, we change the probability that Y becomes 1.

Basics

- We assume that the outcome is Bernoulli distributed (outcome of a Bernoulli trial): this means with probability p the outcome is 1 and with probability $1 - p$ it is 0.
- We are interested in p, so logistic regression models the probability of a binary outcome (Y = 0, 1). Probability Form:

$$P(Y = 1 \mid X) = \frac{1}{1 + e^{-\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k}}$$

- Equivalent log odds form: The model assumes a **logit (log-odds) link** between predictors and the probability:

$$\log\left(\frac{P(Y = 1 \mid X)}{1 - P(Y = 1 \mid X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$$

Odds and Interpretation

- We can interpret a coefficient β_k as changes to the log-odds; and $\exp(\beta_k)$ as changes to odds.

$$\log\left(\underbrace{\frac{P(Y=1 \mid X)}{1-P(Y=1 \mid X)}}_{\text{Odds}}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k.$$

- Odds reflect how likely an event is compared to it not happening ("ratio of 1 to 0"), rather than its share of all outcomes (which would be the probability).
- **Example**: The baseline odds for 100 customers (20 purchase complete, 80 cart abandoned). Then you estimate $\beta_k = 0.5$:

$$\text{Odds} = \frac{20}{80} = 0.25 \qquad e^{\beta_k=0.5} \approx 1.65 \qquad \text{New Odds} = 0.25 \times 1.65 \approx 0.4125.$$

- For a one-unit increase in β_k, the odds increase by about $(\exp(\beta)-1)\cdot 100\% = 65\%$. However, the effect on the actual probability depends on the starting value of the regressors.

Probabilites

- Coefficients from the regression output can be directly interpreted as changes to (log-)odds. If $\beta_1 > 0$, odds increase, as well as probabilities (and reverse). The change does not depend on the actual level of X.
- The reason: Odds are multiplicative, log-odds are additive (changes are "just added").
- However, the change in the outcome probability P$(Y = 1 \mid X)$ depends on the starting level of the covariates.
- Also note:

$$P(Y=1) = \frac{\text{Odds}}{1 + \text{Odds}}$$

- Previous Example (odds before: 0.25, odds after: 0.41)

$$p_1 = \frac{20}{20+80} = \frac{0.25}{1+1.25} = 0.2, \quad p_{\text{new}} \approx \frac{0.4125}{1.4125} \approx 0.291.$$

- Interpretation: For the given baseline probability/odds, increasing the explanatory variable by one unit increases the probability of Y by approx. 9.1%.

How to do in R:

- Use the `glm` (generalized linear models) function from base R.

```
logit_model <- glm(Y ~ X1 + X2, data = df, family = binomial)
summary(logit_model)
```

- Model Fit (Pseudo-R^2):

```
pscl::pR2(logit_model)
```

- Calculate predicted probabilities:

```
predict(logit_model, type = "response")
```

- Get changes in probabilities (average marginal effects, keeps other covariates constant for each obs.)

```
#install.packages("margins")
library(margins)
marginal_effects <- margins(logit_model)
summary(marginal_effects)
```

Example Logit: Movie Box-Office-Bombs

- 1812 movies released in U.S. cinemas between 1995 and 2024 with a budget of at least 25 million dollars.
- We are interested in if a film is a box office flop ("bomb"). Defined as ROI $< 66\%$ and no international success $\rightarrow$ Y $= 1$ (is a bomb).
- Explanatory variables (in the data) are audience and critical scores, as well as film properties like runtime and genre.
- Source: BoxOfficeMojo + IMDb + RottenTomatoes

John Carter (2012)

- Budget: 250 million USD
- Domestic Box Office: 73 million USD
- Audience Score: 60 & Critic Score: 52 (RT), runtime: 132 minutes
- One of the biggest bombs of all time (Disney took a 200 million dollar write-off)

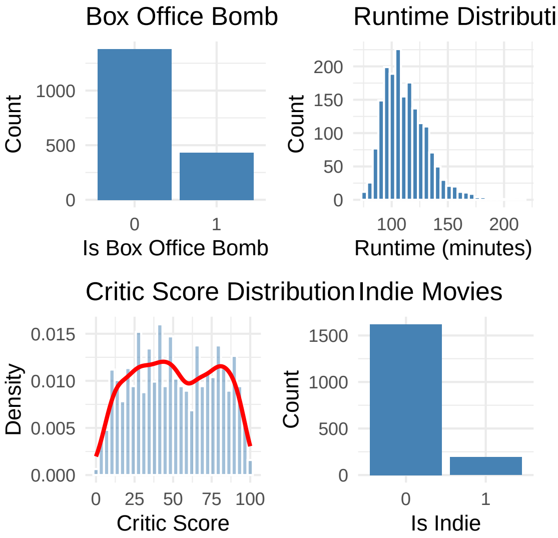
Data

```r
df_movie <- read_csv("movie_select.csv")

df_movie %>%
  select(title,budget, domestic,runtime,audience_score,critic_score,is_bo_bomb) %>%
  arrange(-budget) %>% head(7)
```

```
# A tibble: 7 x 7
  title            budget domestic runtime audience_score critic_score is_bo_bomb
  <chr>             <dbl>    <dbl>   <dbl>          <dbl>        <dbl>      <dbl>
1 Indiana Jones ~     387      174     154             88           70          1
2 Avengers: Endg~     356      858     181             90           94          0
3 Fast X(2023)        340      146     141             84           56          0
4 Avengers: Infi~     321      678     149             92           85          0
5 Pirates of the~     300      309     169             72           44          0
6 Mission: Impos~     291      172     163             94           96          0
7 Solo: A Star W~     275      213     135             63           69          0
```

Distributions



Estimate Model

```r
logit_model <- glm(is_bo_bomb ~  runtime + imdb_avg_rating +
                     audience_score + critic_score + is_indie +
                     ge_action + ge_animation, #+ factor(title_year),
                   data = df_movie,
                   family = binomial)

pscl::pR2(logit_model) # Print R2 (McFadden) + eval
```

```
fitting null model for pseudo-r2
         llh      llhNull           G2     McFadden         r2ML         r2CU
-888.6360852 -995.2245458  213.1769211    0.1070999    0.1109905    0.1664965
```

Model Summary

```r
summary(logit_model) # Print summary
```

```

Call:
glm(formula = is_bo_bomb ~ runtime + imdb_avg_rating + audience_score +
    critic_score + is_indie + ge_action + ge_animation, family = binomial,
    data = df_movie)

Coefficients:
                 Estimate Std. Error z value   Pr(>|z|)
(Intercept)      1.470635   0.543181   2.707   0.006780 **
runtime          0.010314   0.003738   2.759   0.005799 **
imdb_avg_rating -0.346688   0.116730  -2.970   0.002978 **
audience_score  -0.018976   0.005279  -3.595   0.000325 ***
critic_score    -0.009036   0.003473  -2.602   0.009272 **
is_indie         0.820175   0.168235   4.875 0.00000109 ***
ge_action       -0.524517   0.124737  -4.205 0.00002611 ***
ge_animation    -0.402121   0.265862  -1.513   0.130401
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1990.4  on 1811  degrees of freedom
Residual deviance: 1777.3  on 1804  degrees of freedom
AIC: 1793.3

Number of Fisher Scoring iterations: 4
```

Interpreting Coefficients (Odds)

- **Runtime (Estimate = 0.0103):**
  For each additional minute of runtime, the log-odds of being a box office bomb increase by 0.0103. This corresponds to an odds ratio of $\exp(0.0103) \approx 1.01$, meaning about a **1% increase** in the odds per minute.

- **Audience Score (Estimate = -0.0189):**
  Each one-point increase in the RT audience score decreases the log-odds by 0.019. This also implies implies a **(exp(-0.018976) - 1)*100% = -1.879% change** (decrease) in the odds of being a box office bomb per point.

- **Is Indie (Estimate = 0.8201):**
  Being an indie film (when (is_indie = 1)) increases the log-odds by 0.82 relative to non-indie films. This translates to an odds ratio of an odds ratio of 2.27, indicating that indie films have approximately **127% higher odds** of being a box office bomb compared to non-indie films.

Marginal Effects (on $P(Y = 1)$)

```r
marginal_effects <- margins(logit_model)
summary(marginal_effects)
```

```
          factor     AME     SE       z      p   lower   upper
  audience_score -0.0030 0.0008 -3.6310 0.0003 -0.0047 -0.0014
    critic_score -0.0014 0.0006 -2.6134 0.0090 -0.0025 -0.0004
       ge_action -0.0840 0.0197 -4.2652 0.0000 -0.1226 -0.0454
    ge_animation -0.0644 0.0425 -1.5140 0.1300 -0.1477  0.0190
 imdb_avg_rating -0.0555 0.0185 -2.9953 0.0027 -0.0918 -0.0192
        is_indie  0.1313 0.0263  4.9858 0.0000  0.0797  0.1829
         runtime  0.0017 0.0006  2.7765 0.0055  0.0005  0.0028
```

Interpretation. On average:

- a one-unit decrease in the audience score decreases the probability that the film is a box office bomb by 0.3%.
- a one-unit increase in the runtime increases the probability that the film is a box office bomb by 0.17%.
- switching from a non-indie to an indie film increases the predicted probability by roughly 13.1 [7.9, 18.2] percentage points.

Prediction: Predict/classify new data

Create a stinker (long and bad reviews)

```r
new_bad_film <- data.frame(
  runtime = 240,
  imdb_avg_rating = 6.2, audience_score = 22, critic_score = 34,    # BAD REVIEWS!
  is_indie = 1, ge_action = 0, ge_animation = 0
)

predicted_prob <- predict(logit_model, newdata = new_bad_film, type = "response")
cat("The probability of bombing is: ", predicted_prob)
```

```
The probability of bombing is:  0.868991
```

Create a masterpiece (great reviews and not too long)

```r
new_great_film <- data.frame(
  runtime = 120,
  imdb_avg_rating = 9.2, audience_score = 89, critic_score = 79,
  is_indie = 0, ge_action = 0, ge_animation = 1
)

predicted_prob <- predict(logit_model, newdata = new_great_film, type = "response")
cat("The probability of bombing is: ", predicted_prob)
```

```
The probability of bombing is:  0.03605312
```

Prediction and Classification (on existing data) and Model Eval

- Since logit is a common classifier, we can also evaluate our model on the quality of the models predicition (not just probabilities)
- We can compare actual classes to predicted classes and look at how often the model is wrong ("cross entropy loss").

Add the predicted probability as a new column and construct the predicted class (threshold 0.5)

```r
set.seed(100)
df_movie <- df_movie %>%
  mutate(
    predicted_prob = predict(logit_model, type = "response"),
    predicted_class = if_else(predicted_prob >= 0.5, 1, 0)
  )

df_movie %>% select(title,is_bo_bomb,predicted_prob,predicted_class) %>%
  slice_sample(n = 6)
```

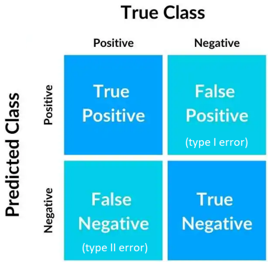
```
# A tibble: 6 x 4
  title                    is_bo_bomb predicted_prob predicted_class
  <chr>                         <dbl>          <dbl>           <dbl>
1 Mumford(1999)                     1         0.184                0
2 The Social Network(2010)          0         0.0749               0
3 Sanctum(2011)                     0         0.277                0
4 Lucky Numbers(2000)               1         0.554                1
5 Nanny McPhee(2005)                0         0.154                0
6 Barney's Version(2010)            1         0.135                0
```

Confusion Matrix and Metrics (from package yardstick)

```
df_movie <- df_movie %>%
  mutate(
    truth = factor(is_bo_bomb, levels = c(1, 0), labels = c("Yes", "No")),
    predicted = factor(predicted_class, levels = c(1, 0), labels = c("Yes", "No"))
  )

cm <- yardstick::conf_mat(df_movie, truth, predicted)
```

```
          Truth
Prediction  Yes   No
       Yes   63   50
       No   369 1330
```



Metrics

There are several metrics than can be used to evaluate a binary classification model (can also use yardstick functions), which depend on these 4 values.

```
          Truth
Prediction  Yes   No
       Yes   63   50
       No   369 1330
```

```r
TP <- 63 # True positives
TN <- 1330 # True negatives
FP <- 50 # False positives
FN <- 369 # False negatives
```

- Accuracy (overall correctness):

```r
(TP + TN) / (TP + TN + FP + FN)
```

```
[1] 0.7687638
```

- Sensitivity/Recall: indicates how well the model identifies actual positives.

```r
TP / (TP + FN)
```

```
[1] 0.1458333
```

- Specificity/True Negative Rate: measures how well the model identifies actual negatives.

```r
TN / (TN + FP)
```

```
[1] 0.9637681
```

- Precision: reflects the accuracy of the positive predictions.

```r
TP / (TP + FP)
```

```
[1] 0.5575221
```

Final thoughts

- Our model misses a lot of positives. We could experiment with lowering the threshold, if our goal is to find more positives (trade-off between recall and precision).
- Use XGBoost or similarly flexible model (always use out-of-sample testing).
- Probably lots of missing confounders :(
- We could try to add or engineer more features (regressors) like interaction terms, “text-based stuff” and crew info.