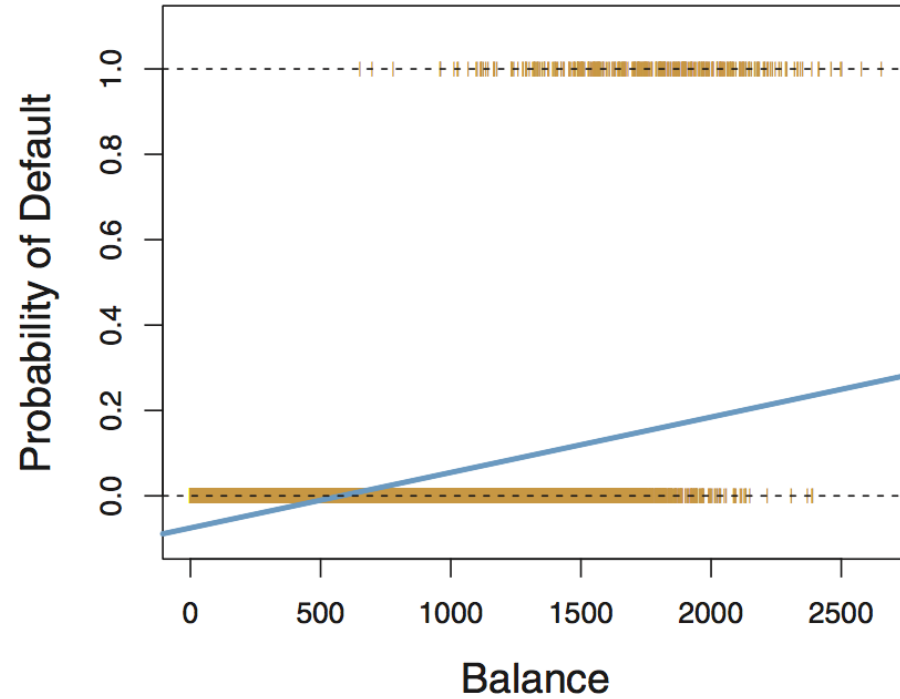


Supervised Learning

Logistic regression

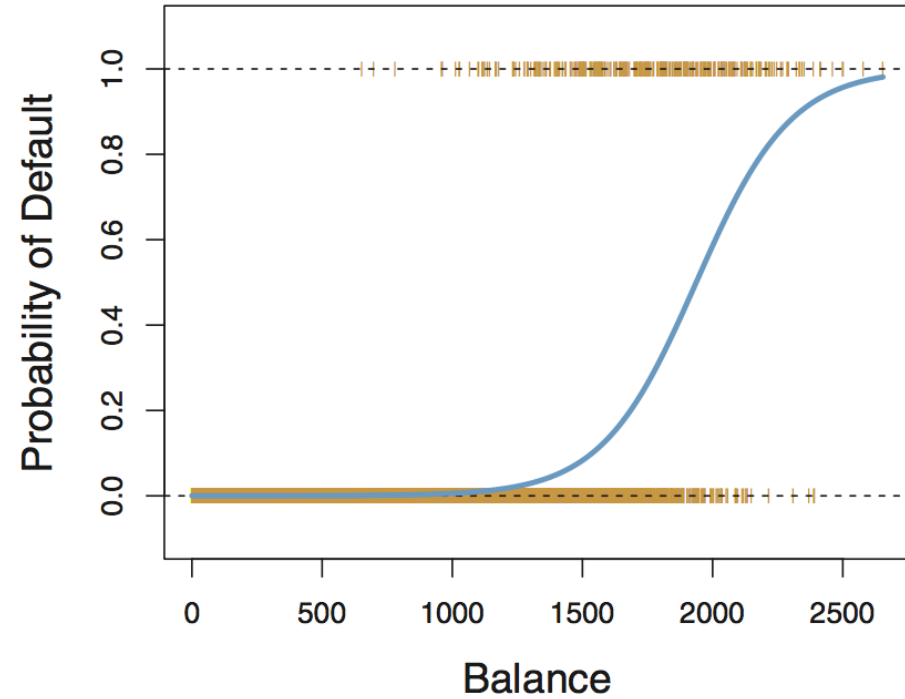
June 24th, 2021

The setting: Figure 4.2 (ISLR)



Left: Linear regression

- not limited to be within $[0, 1]$!



Right: **Logistic regression**

- respects the observed range of outcomes!

Generalized linear models (GLMs) review

Linear regression: estimate **mean value** of response variable Y , given predictor variables x_1, \dots, x_p :

$$\mathbb{E}[Y|x] = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$

In a **GLM**, we include a **link function** g that transforms the linear model:

$$g(\mathbb{E}[Y|x]) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$

- Use g to reduce the range of possible values for $\mathbb{E}[Y|x]$ from $(-\infty, \infty)$ to, e.g., $[0, 1]$ or $[0, \infty)$, etc.

In a GLM you specify a **probability distribution family** that governs the observed response values

- e.g. if Y are zero and the positive integers, the family could be **Poisson**
- e.g. if Y are just 0 and 1, the family is **Bernoulli** and extends to **Binomial** for n independent trials

Logistic regression

Assuming that we are dealing with two classes, the possible observed values for Y are 0 and 1,

$$Y|x \sim \text{Binomial}(n = 1, p = \mathbb{E}[Y|x]) = \text{Bernoulli}(p = \mathbb{E}[Y|x])$$

To limit the regression between $[0, 1]$: use the **logit** function, aka the **log-odds ratio**

$$\text{logit}(p(x)) = \log \left[\frac{p(x)}{1 - p(x)} \right] = \log \left[\frac{\mathbb{E}[Y|x]}{1 - \mathbb{E}[Y|x]} \right] = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$

meaning

$$p(x) = \mathbb{E}[Y|x] = \frac{e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p}}$$

Major difference between linear and logistic regression

Logistic regression **involves numerical optimization**

- y_i is observed response for n observations - either 0 or 1
- we need to use an iterative algorithm to find β 's that maximize the **likelihood**

$$\prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

- **Newton's method:** start with initial guess, calculate gradient of log-likelihood, add amount proportional to the gradient to parameters, moving up log-likelihood surface
- means logistic regression runs more slowly than linear regression
- if you're interested: **you use iteratively re-weighted least squares, Section 12.3.1**

Inference with logistic regression

Major motivation for logistic regression (and all GLMs) is **inference**

- how does the response change when we change a predictor by one unit?

For linear regression, the answer is straightforward

$$\mathbb{E}[Y|x] = \beta_0 + \beta_1 x_1$$

For logistic regression... it is a little *less* straightforward,

$$E[Y|x] = \frac{e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p}}$$

- the predicted response varies **non-linearly** with the predictor variable values
- one convention is to fall back upon the concept of **odds**

The odds interpretation

Pretend the predicted probability is 0.8 given a particular predictor variable value

- just pretend we only have one predictor variable

This means that if we were to repeatedly sample response values given that predictor variable value: **we expect class 1 to appear 4 times as often as class 0**

$$Odds = \frac{\mathbb{E}[Y|x]}{1 - \mathbb{E}[Y|x]} = \frac{0.8}{1 - 0.8} = 4 = e^{\beta_0 + \beta_1 x}$$

Thus we say that for the given predictor variable value, the *Odds* are 4 (or 4-1) in favor of class 1

How does the odds change if I change the value of a predictor variable by one unit?

$$Odds_{\text{new}} = e^{\beta_0 + \beta_1(x+1)} = e^{\beta_0 + \beta_1 x} e^{\beta_1} = e^{\beta_1} Odds_{\text{old}}$$

For every unit change in x , the odds change by a **factor** e^{β_1}

Example data: NFL field goal attempts

Created dataset using `nflscrapR-data` of all NFL field goal attempts from 2009 to 2019

```
nfl_fg_attempts <- read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/glm_examples,  
nfl_fg_attempts
```

```
## # A tibble: 10,811 × 11  
##   kicke...1 kicke...2   qtr score...3 home_...4 posteam poste...5 kick_...6 pbp_s...7 abs_s...8  
##   <chr>      <chr>    <dbl>   <dbl> <chr>      <chr>      <chr>      <dbl>    <dbl>    <dbl>  
##  1 00-002... R.Biro...     1       0 PIT      TEN      away        37     2009      0  
##  2 00-002... R.Biro...     2       0 PIT      TEN      away        31     2009      0  
##  3 00-002... R.Biro...     4       0 PIT      TEN      away        45     2009      0  
##  4 00-002... J.Reed       4      -3 PIT      PIT      home        32     2009      3  
##  5 00-002... J.Reed       5       0 PIT      PIT      home        33     2009      0  
##  6 00-000... P.Daws...     1       0 CLE      CLE      home        37     2009      0  
##  7 00-001... R.Long...     1      -3 CLE      MIN      away        21     2009      3  
##  8 00-000... P.Daws...     2      -7 CLE      CLE      home        20     2009      7  
##  9 00-001... R.Long...     4      12 CLE      MIN      away        37     2009     12  
## 10 00-000... J.Hans...     1     -14 NO       DET      away        47     2009     14  
## # ... with 10,801 more rows, 1 more variable: is_fg_made <dbl>, and abbreviated  
## #   variable names 1kicker_player_id, 2kicker_player_name, 3score_differential,  
## #   4home_team, 5posteam_type, 6kick_distance, 7pbp_season, 8abs_score_diff
```

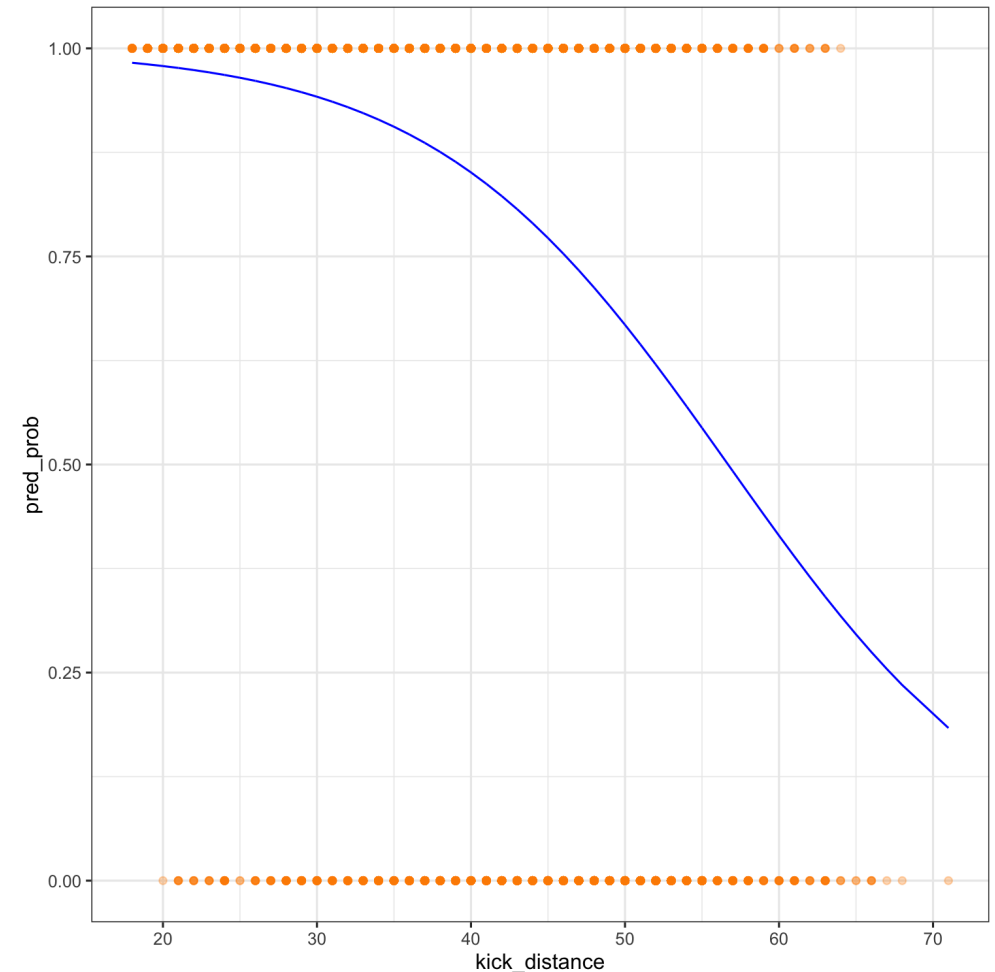

Fitting a logistic regression model

- We use the `glm` function (similar to `lm`)
- **Specify the family is `binomial`**

```
init_logit <- glm(is_fg_made ~ kick_distance,  
                  data = nfl_fg_attempts,  
                  family = "binomial")
```

- View predicted probability relationship

```
nfl_fg_attempts %>%  
  mutate(pred_prob = init_logit$fitted.values)  
  ggplot(aes(x = kick_distance)) +  
    geom_line(aes(y = pred_prob),  
              color = "blue") +  
    geom_point(aes(y = is_fg_made),  
               alpha = 0.3,  
               color = "darkorange") +  
  theme_bw()
```



```
summary(init_logit)
```

```
##
## Call:
## glm(formula = is_fg_made ~ kick_distance, family = "binomial",
##      data = nfl_fg_attempts)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   5.916656   0.145371  40.70  <2e-16 ***
## kick_distance -0.104365   0.003255 -32.06  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 9593.1  on 10810  degrees of freedom
## Residual deviance: 8277.5  on 10809  degrees of freedom
## AIC: 8281.5
##
## Number of Fisher Scoring iterations: 5
```

What is Deviance?

For model of interest \mathcal{M} the total deviance is:

$$D_{\mathcal{M}} = -2 \log \frac{\mathcal{L}_{\mathcal{M}}}{\mathcal{L}_{\mathcal{S}}} = 2 (\log \mathcal{L}_{\mathcal{S}} - \log \mathcal{L}_{\mathcal{M}})$$

- $\mathcal{L}_{\mathcal{M}}$ is the likelihood for model \mathcal{M}
- $\mathcal{L}_{\mathcal{S}}$ is the likelihood for the **saturated** model, with n parameters! (i.e., a perfect fit)
- Can think of $\mathcal{L}_{\mathcal{S}}$ as some constant that does not change

Deviance is a measure of goodness of fit: the smaller the deviance, the better the fit

- Generalization of RSS in linear regression to any distribution family

Logistic regression output

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.7752	0.2420	0.4025	0.6252	1.5136

The **deviance residuals** are contributions to total deviance (signed square roots of unit deviances)

$$d_i = \text{sign}(y_i - \hat{p}_i) \sqrt{-2[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]}$$

where y_i is the i^{th} observed response and \hat{p}_i is the estimated probability of success

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	5.916656	0.145371	40.70	<2e-16 ***
kick_distance	-0.104365	0.003255	-32.06	<2e-16 ***

The intercept of the prediction curve is $e^{5.916656}$ and $Odds_{\text{new}}/Odds_{\text{old}} = e^{-0.104365}$.

Logistic regression output

Null deviance: 9593.1 on 10810 degrees of freedom
Residual deviance: 8277.5 on 10809 degrees of freedom
AIC: 8281.5

```
logLik(init_logit) # the maximum log-likelihood value
```

```
## 'log Lik.' -4138.732 (df=2)
```

- **Residual deviance** is -2 times -4138.732, or 8277.5 (*What about the saturated model?*)
 - Null deviance corresponds to intercept-only model
- **AIC** is $2k - 2 \log \mathcal{L} = 2 \cdot k - 2 \cdot (-4138.732) = 8281.5$
 - where k is the number of degrees of freedom (here, $df = 2$)
- These are all metrics of quality of fit of the model
- **We will consider these to be less important than test-set performances**

Logistic regression predictions

To generate logistic regression predictions there are few things to keep in mind...

- the `fitted.values` **are on the probability scale**: all are between 0 and 1
- but the **default** for `predict(init_logit)` is **the log-odds scale!**
- we change this with the `type` argument: `predict(init_logit, type = "response")`

How do we predict the class? e.g make or miss field goal?

```
pred_fg_outcome <- ifelse(init_logit$fitted.values > 0.5,  
                          "make", "miss")
```

- typically if predicted probability is > 0.5 then we predict success, else failure

Model assessment

Most straight-forward way is the **confusion matrix** (rows are predictions, and columns are observed):

```
table("Predictions" = pred_fg_outcome, "Observed" = nfl_fg_attempts$is_fg_made)
```

```
##           Observed
## Predictions    0    1
##           make 1662 8994
##           miss   94   61
```

In-sample misclassification rate:

```
mean(ifelse(fitted(init_logit) < 0.5, 0, 1) != nfl_fg_attempts$is_fg_made)
```

```
## [1] 0.1593747
```

Brier score:

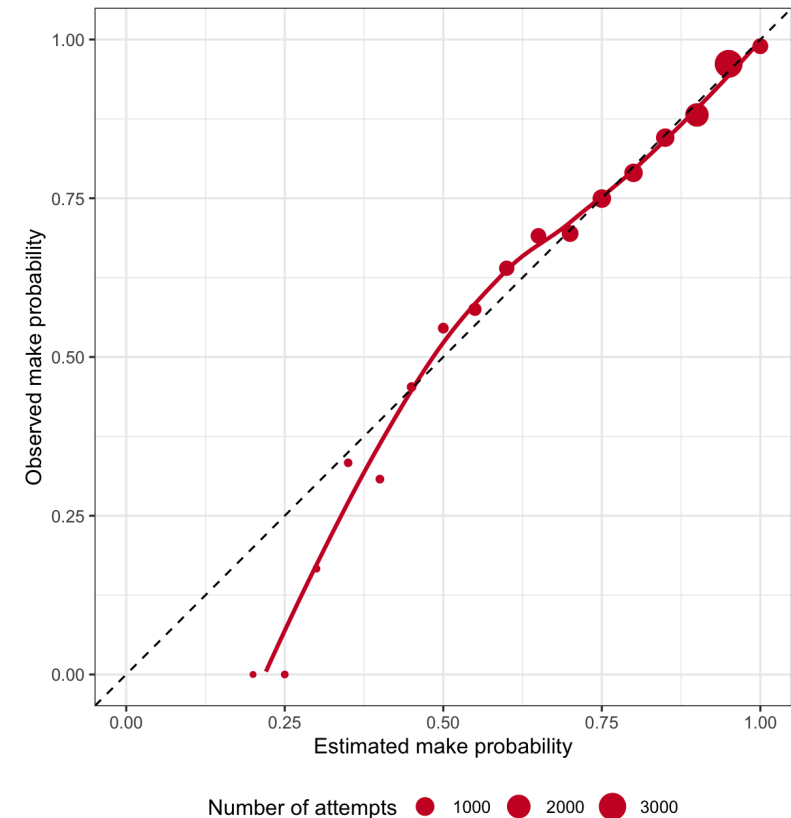
```
mean((nfl_fg_attempts$is_fg_made - fitted(init_logit))^2)
```

```
## [1] 0.1197629
```

Well-calibrated if actual probabilities match predicted probabilities

```
nfl_fg_attempts %>%
  mutate(pred_prob = init_logit$fitted.values,
         bin_pred_prob = round(pred_prob / 0.1))
# Group by bin_pred_prob:
group_by(bin_pred_prob) %>%
# Calculate the calibration results:
summarize(n_attempts = n(),
         bin_actual_prob = mean(is_fg_made))
ggplot(aes(x = bin_pred_prob, y = bin_actual_prob)) +
  geom_point(aes(size = n_attempts)) +
  geom_smooth(method = "loess", se = FALSE) +
  geom_abline(slope = 1, intercept = 0,
             color = "black", linetype = "dashed") +
  coord_equal() +
  scale_x_continuous(limits = c(0,1)) +
  scale_y_continuous(limits = c(0,1)) +
  labs(size = "Number of attempts",
       x = "Estimated make probability",
       y = "Observed make probability") +
  theme_bw() +
  theme(legend.position = "bottom")
```

If model says the probability of rain for a group of days is 50%, it better rain on half those days... **or something is incorrect about the probability!**



BONUS: Leave-one-season-out cross validation (with **purrr**)

In many datasets rather than random holdout folds, you might have particular holdouts of interest (e.g. seasons, games, etc.)

```
nfl_fg_loso_cv_preds <- # generate holdout predictions for every row based season
  map_dfr(unique(nfl_fg_attempts$pbp_season),
    function(season) {
      # Separate test and training data:
      test_data <- nfl_fg_attempts %>%
        filter(pbp_season == season)
      train_data <- nfl_fg_attempts %>%
        filter(pbp_season != season)

      # Train model:
      fg_model <- glm(is_fg_made ~ kick_distance, data = train_data,
        family = "binomial")

      # Return tibble of holdout results:
      tibble(test_pred_probs = predict(fg_model, newdata = test_data,
        type = "response"),
        test_actual = test_data$is_fg_made,
        test_season = season)
    })
```

Overall holdout performance

Misclassification rate:

```
nfl_fg_loso_cv_preds %>%  
  mutate(test_pred = ifelse(test_pred_probs < .5, 0, 1)) %>%  
  summarize(mcr = mean(test_pred != test_actual))
```

```
## # A tibble: 1 × 1  
##       mcr  
##   <dbl>  
## 1 0.160
```

Brier score:

```
nfl_fg_loso_cv_preds %>%  
  summarize(brier_score = mean((test_actual - test_pred_probs)^2))
```

```
## # A tibble: 1 × 1  
##   brier_score  
##   <dbl>  
## 1 0.120
```

Holdout performance by season

```
nfl_fg_loso_cv_preds %>%  
  mutate(test_pred = ifelse(test_pred_probs < .5, 0, 1)) %>%  
  group_by(test_season) %>%  
  summarize(mcr = mean(test_pred != test_actual)) %>%  
  ggplot(aes(x = test_season, y = mcr)) +  
  geom_bar(stat = "identity", width = .1) + geom_point(size = 5) +  
  theme_bw() +  
  scale_x_continuous(breaks = unique(nfl_fg_loso_cv_preds$test_season))
```

