

# LOGISTIC REGRESSION

A Powerful Tool for Classification

# BASICS OF LOGISTIC REGRESSION

Classification Algorithm:

- **Two categories for the outcome variable (analysed in what follows):** e.g. Unemployed true or false
- Multiple categories for the outcome variable (not covered here)
  - unordered logistic regression
  - ordered logistic regression »

# A MOCK-UP EXAMPLE TO INTRODUCE THE IDEA

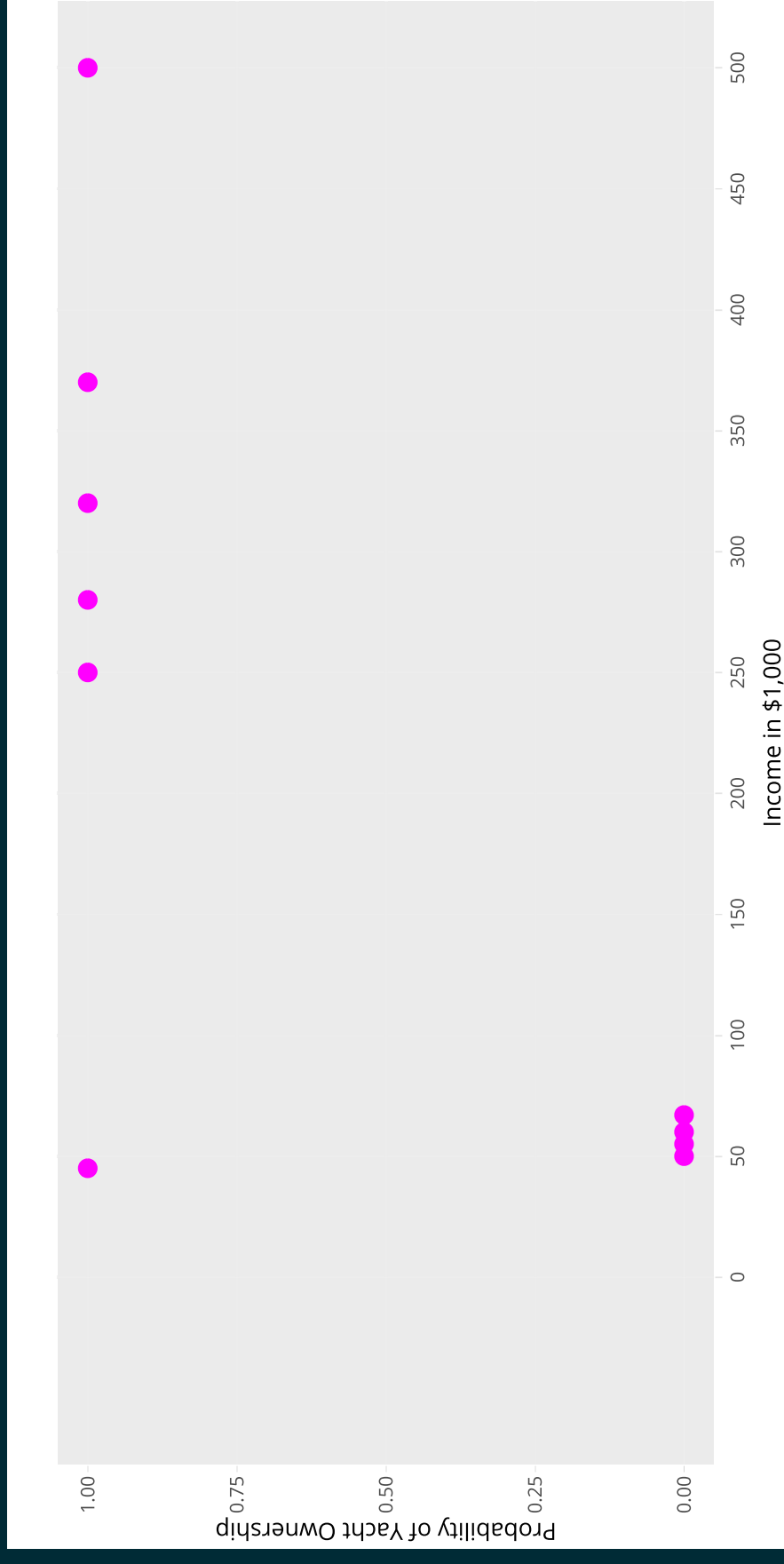
► Code

Income and Yacht Ownership

Name	Income	Yacht
Jack	45	1
Sarah	50	0
Carl	55	0
Eric	60	0
Zoe	67	0
James	250	1
Enrico	280	1
Erica	320	1
Stephanie	370	1
Susan	500	1

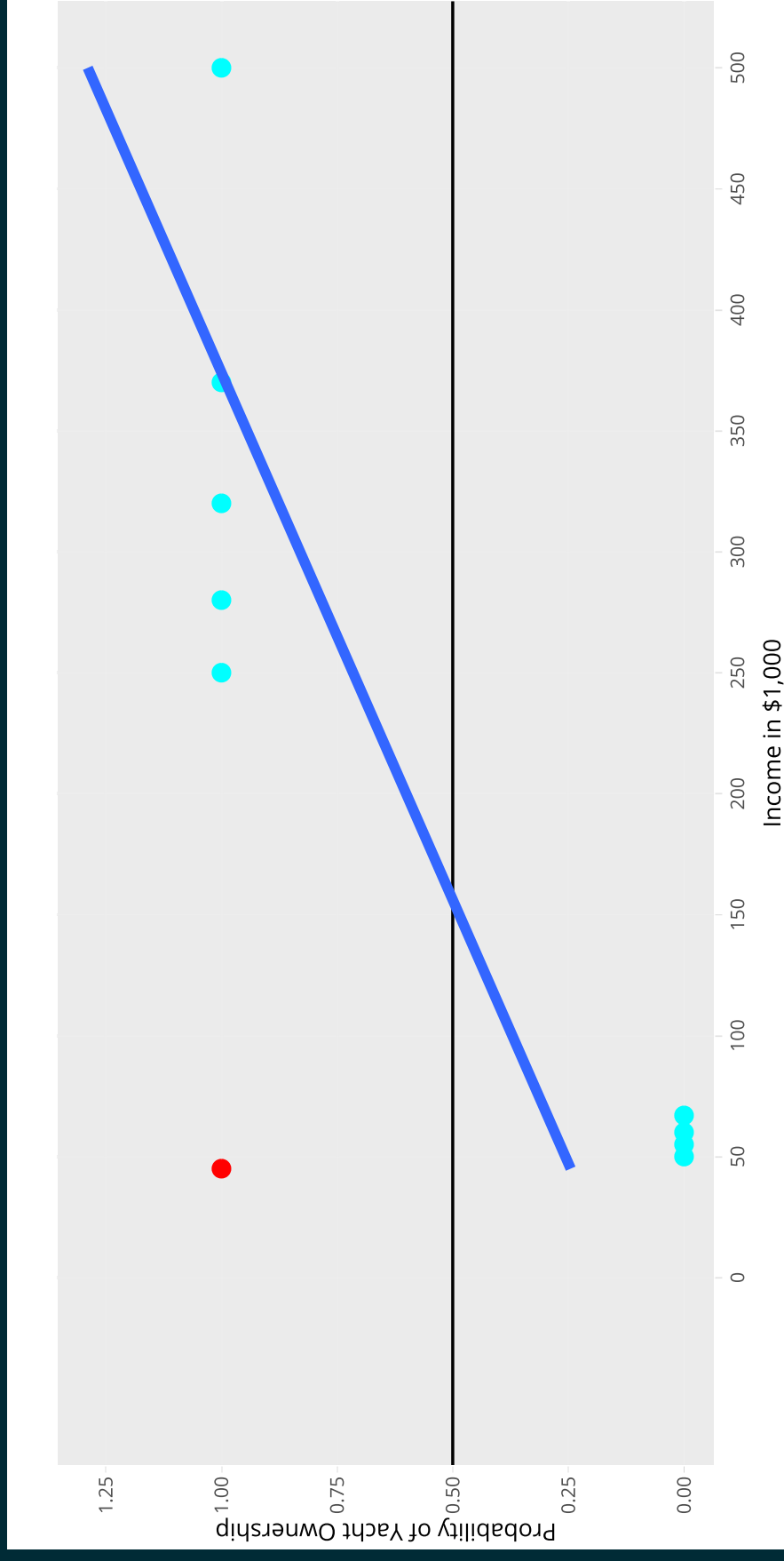
# USING OLS IS A TEMPTING (BUT BAD) IDEA

► Code

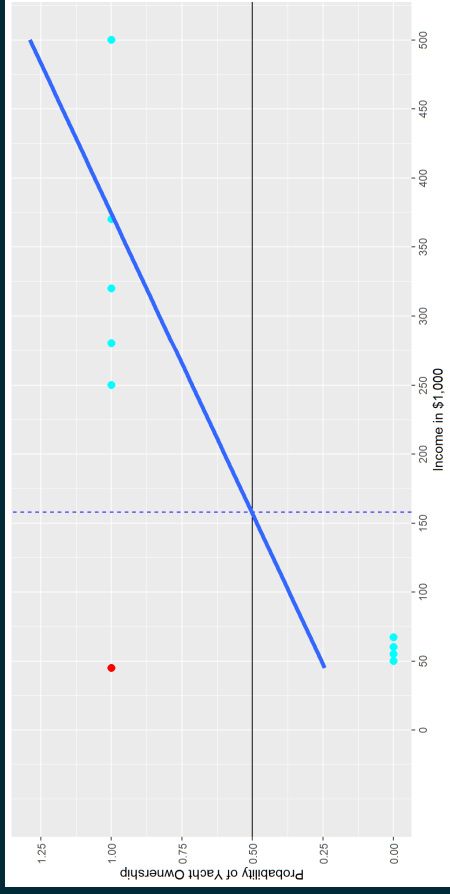


# USING OLS IS A TEMPTING (BUT BAD) IDEA

► Code

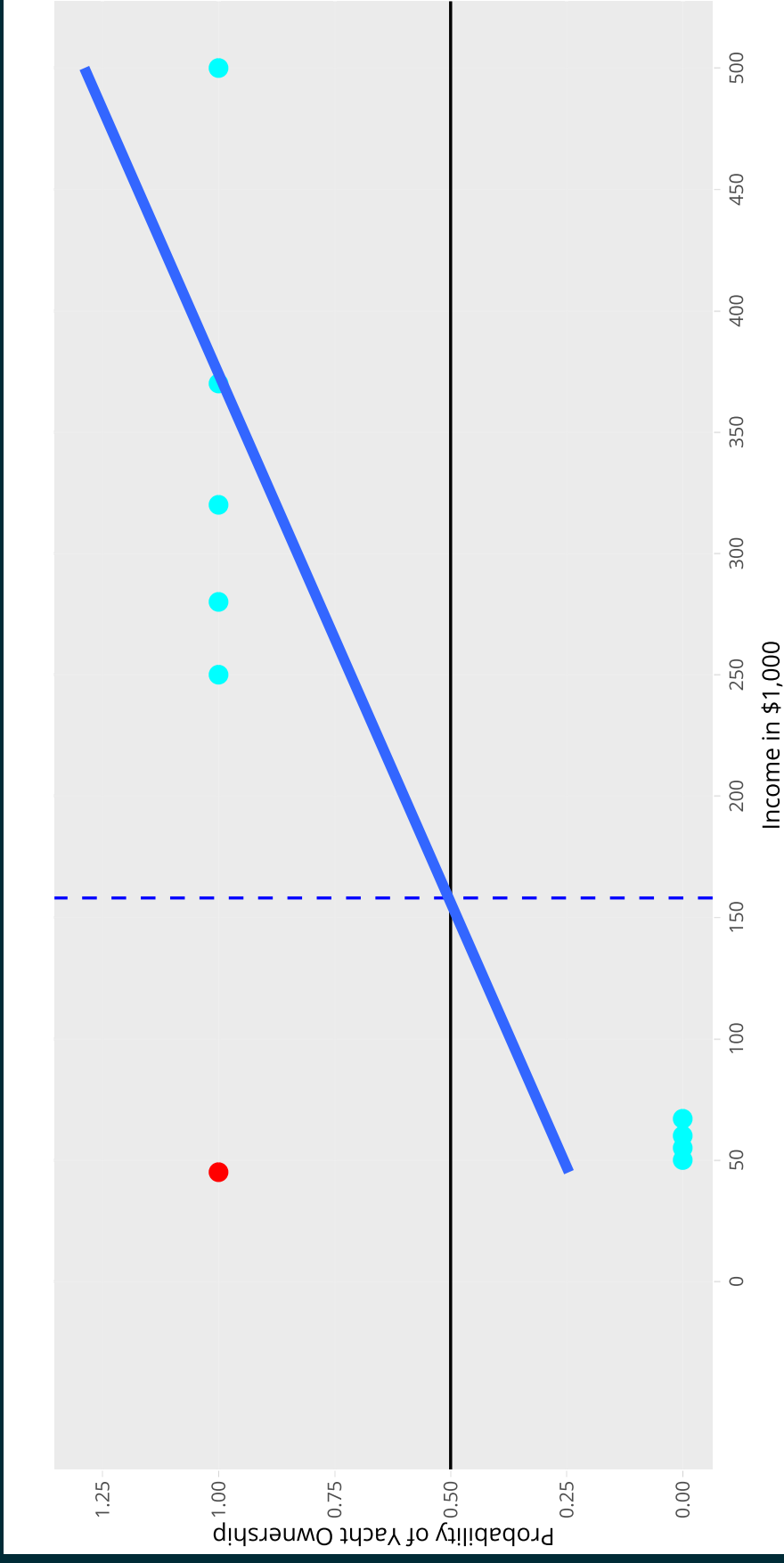


# QUICK WAY TO FIND A DECISION BOUNDARY



1. Find the intersection point between the prediction line and the horizontal 0.5 probability line.
2. Draw a vertical line through the intersection point. This line is called a **decision boundary**.
3. All incomes left of the *decision boundary* (income smaller than 158) are predicted as “no”. All incomes right of the *decision boundary* (income greater than 158) are predicted as “yes”.»

# WHY OLS FOR CLASSIFICATION IS A BAD IDEA

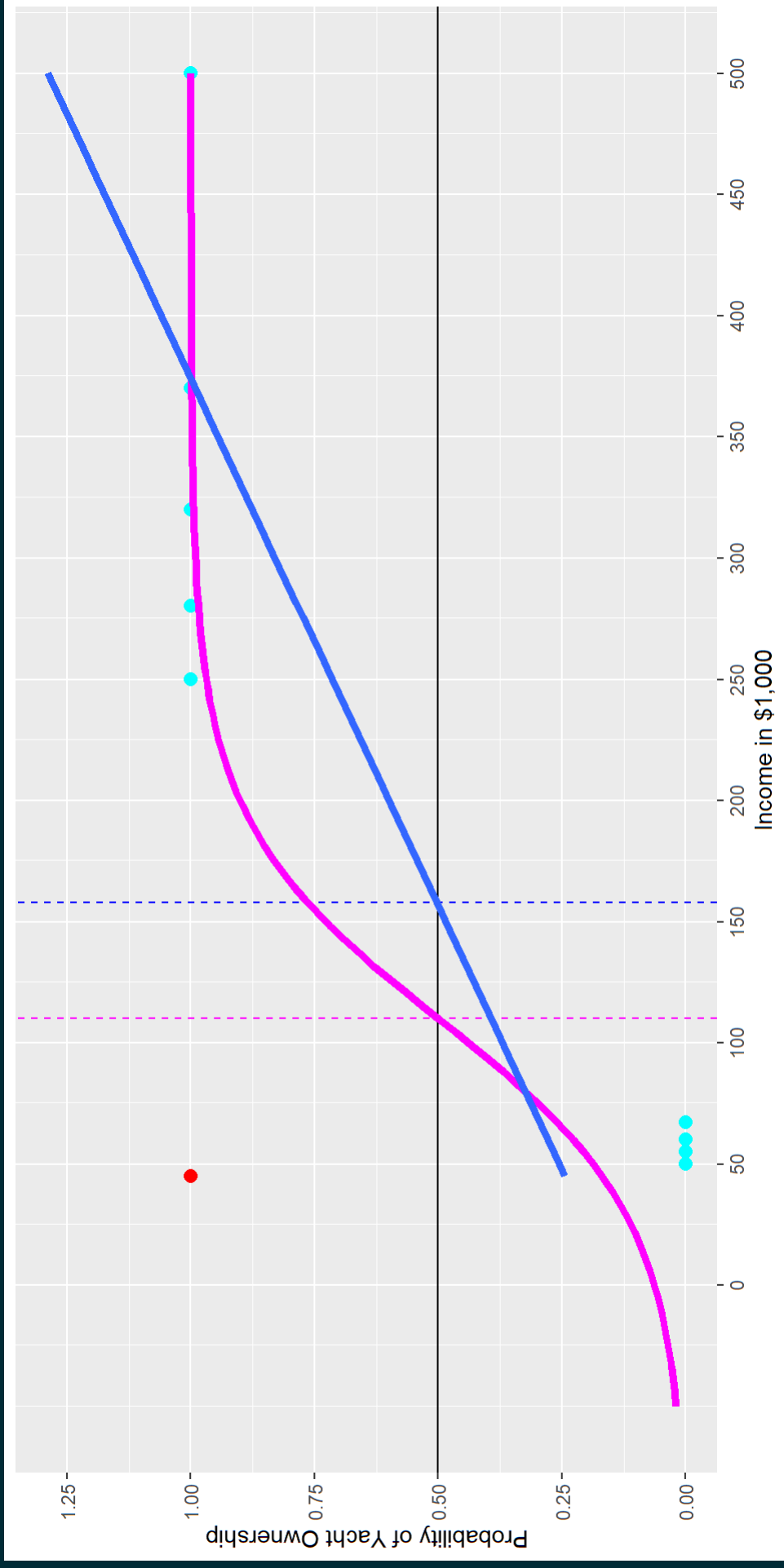


Note, incomes > \$370,000 are predicted with a probability > 100% to be yacht owners(?)

E.g. probability of owning a yacht for an income of \$500,000 is 125% (?)

A similar problem can occur with negative probabilities!

# A STEP-FUNCTION AS AN ALTERNATIVE TO OLS





# POPULAR STEP-FUNCTIONS (SIGMOID FUNCTION)

- The **Hyperbolic Tangent function**.
- The **Arc Tangent function**.
- The **Logistic function** (confusingly sometimes also called the *sigmoid function*).

$$y_i = \frac{1}{1 + e^{-x_i}}$$

»

# THE LOGISTIC FUNCTION

- The **Logistic function** (confusingly sometimes also called the *sigmoid function*):

$$y_i = \frac{1}{1 + e^{-x_i}}$$

We use:  $y_i = P_{yes,i}^{prob}$  and  $x_i = \beta_1 Inc_i + \beta_2$  which gives us:

$$P_{yes,i}^{prob} = \frac{1}{1 + e^{-(\beta_1 Inc_i + \beta_2)}}$$

$\beta_1$  and  $\beta_2$  change slope and position

$\beta_1 = 1$  and  $\beta_2 = 0$  gives the org. logistic function. 🤖

# WHAT MAKES THE LOGISTIC FUNCTION SO SPECIAL?

## – COMPARED TO OTHER SIGMOID (STEP) FUNCTIONS –

### Time for some mathematical magic:

Logistic function  $Prob_{yes,i}$  := probability for positive event (e.g. yacht ownership: yes):

$$Prob_{yes,i} = \frac{1}{1 + e^{-(\beta_1 \cdot x_i + \beta_2)}}$$

Take the inverse on both sides of the equation:

$$\frac{1}{Prob_{yes,i}} = 1 + e^{-(\beta_1 \cdot x_i + \beta_2)}$$

Subtract 1 on both sides:

$$\frac{1}{P_{yes,i}^{prob}} - 1 = e^{-(\beta_1 \cdot x_i + \beta_2)}$$

Consider that  $-1 = -\frac{P_{yes,i}^{prob}}{P_{yes,i}^{prob}}$  and substitute  $-1$  accordingly, we get after simplification:

$$\frac{1 - P_{yes,i}^{prob}}{P_{yes,i}^{prob}} = e^{-(\beta_1 \cdot x_i + \beta_2)}$$

$1 - P_{yes,i}^{prob}$  equals by definition  $P_{no,i}^{prob}$ :

$$\frac{P_{no,i}^{prob}}{P_{yes,i}^{prob}} = e^{-(\beta_1 \cdot x_i + \beta_2)}$$

Take again the inverse on both sides:

$$\frac{P_{yes,i}^{prob}}{P_{no,i}^{prob}} = e^{\beta_1 \cdot x_i + \beta_2}$$

Take the logarithm on both sides:

$$\ln \left( \frac{P_{yes,i}^{prob}}{P_{no,i}^{prob}} \right) = \beta_1 \cdot x_i + \beta_2$$

# ONE MORE STEP – ODDS VS PROBABILITIES

- The fraction of the yes/no probabilities can be interpreted as *Odds* as they are often used in betting.
- Example: The probability of getting two heads when flipping two coins is  $P_{yes,i}^{prob} = 0.25$ .
- Consequently, the probability of **not** getting two heads when flipping two coins is  $P_{no,i}^{prob} = 0.75$ .
- *Odds* for 2 Heads compared to **not** 2 heads is 1 to 3 or 33%:

$$Odds = \frac{P_{yes,i}^{prob}}{P_{no,i}^{prob}} = \frac{0.25}{0.75} = \frac{1}{3} = 0.33$$

# INTERPRETATION OF THE $\beta$ S: YACHT OWNERSHIP

$$\ln(Odds) = \ln\left(\frac{Prob_{yes,i}}{Prob_{no,i}}\right) = 0.02 \cdot Inc_i + (-2.7)$$

Model results after running and printing the workflow():

## ► Code

```
== Workflow [trained]
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor
0 Recipe Steps

-- Model

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)

Coefficients:
(Intercept)      Income
-2.68660      0.02448

Degrees of Freedom: 9 Total (i.e. Null); 8 Residual
Null Deviance:      13.46
Residual Deviance:  5.654 AIC: 9.654
```

<https://econ.lange-analytics.com/aiobook/>





# INTERPRETATION OF THE $\beta$ S: YACHT OWNERSHIP

$$\ln(Odds) = \ln\left(\frac{P_{yes,i}^{prob}}{P_{no,i}^{prob}}\right) = 0.02 \cdot Inc_i + (-2.7)$$

- If income increases by 1 (\$1,000) the logarithm of the odds increases by 0.02.
- Since change of a logarithm is a relative change (**percentage**):

If income increases by 1 (\$1,000) the odds increases by 2% (0.02).  
(careful with the results because data were made up and N is too small!)

# CONFUSION MATRIX

Note, in the mockup we did not create training and testing data. Therefore, we use DataYachts (the data we used to fit/train the workflow) here. This is not a proper methodology but good enough for the mock-up:

## ► Code

```
Truth
Prediction 0 1
           0 4 1
           1 0 5
```

# REAL WORLD CHURN ANALYSIS WITH LOGISTIC REGRESSION – THE DATA

We use data (7,043 customers) of the fictional telecommunication company *TELCO*, generated by *IBM* for training purposes:

- The outcome variable *Churn* indicates, if a customer departed within the last month ( $Churn = Yes$ ) or not ( $Churn = No$ ).
- Predictor variables contain:
  - Customers' *Gender* (*Female* or *Male*),
  - Customers' *SeniorCitizen* status (0 for no or 1 for yes),
  - Customers' *Tenure* with *TELCO* (month of membership), as well as
  - Customers' *MonthlyCharges* (in US-\$).

# REAL WORLD CHURN ANALYSIS WITH LOGISTIC REGRESSION – THE DATA

## ► Code

	Churn	Gender	SeniorCitizen	Tenure	MonthlyCharges
1	No	Female	0	1	29.85
2	No	Male	0	34	56.95
3	Yes	Male	0	2	53.85
4	No	Male	0	45	42.30
5	Yes	Female	0	2	70.70
6	Yes	Female	0	8	99.65

# REAL WORLD CHURN ANALYSIS WITH LOGISTIC REGRESSION

## – DO IT YOURSELF –

Create the Churn analysis with logistic regression. Click on the link in the footer to get an R-script with a skeleton for the analysis. 🧐

# RESULTS FROM CHURN ANALYSIS WITH LOGISTIC REGRESSION

Confusion Matrix:

	Yes	No
Yes	239	150
No	322	1403

Accuracy:

.metric	.estimator	.estimate
accuracy	binary	0.7767266

Sensitivity:

.metric	.estimator	.estimate
sensitivity	binary	0.426025

Specificity:

.metric	.estimator	.estimate
specificity	binary	0.9034127

**Hint:** What do the column sums of the confusion matrix tell you?

# PROBLEM: UNBALLANCED TRAINING DATA

Churn	n
Yes	1308
No	3621

**Majority Class:** *Churn = No* has 3621 observations in the training dataset.

**Minority class** *Churn = Yes* has 1308 observations in the training dataset.

# WHAT CAN WE DO?

Churn	n
Yes	1308
No	3621

- **Downsampling:** Randomly **delete observations from majority class** until ratio of the observations from the majority and the minority class reaches the desired ratio (e.g., 1:1).
- **Upsampling:** In simplest version, **creates new observations for the minority class** by copying randomly chosen observations from the minority class until the ratio of the observations from the majority and the minority class reaches the desired ratio (e.g., 1:1).
- Often, a combination of *downsampling* and *upsampling* is performed.



# PERFORMING DOWN-SAMPLING WITH `step_downsample()`

You need to add the R package `themis`. Then in your script, you can add `step_downsample(Churn)` to the recipe (don't forget to execute the following command lines again). As a reminder our original DataTrain had 4,929 observations,  $Churn_{Yes} = 1308$ ,  $Churn_{No} = 3621$ :

► Code

Churn	n
Yes	1308
No	1308

Note, the number of observations has decreased by 2313. This is an information loss!

# PERFORMING UP-SAMPLING WITH `step_upsample()`

You need to add the R package `themis`. Then in your script, you can add `step_upsample(Churn)` to the recipe (don't forget to execute the following command lines again). As a reminder our original `DataTrain` had 4,929 observations,  $Churn_{Yes} = 1308$ ,  $Churn_{No} = 3621$ :

► Code

Churn	n
Yes	3621
No	3621

Note, the number of observations has increased by 2313. The information in the dataset has not increased!

# PERFORMING UP-SAMPLING WITH `step_smote()`. WHAT IS THE ADVANTAGE

As a reminder our original DataTrain had 4,929 observations,  $Churn_{Yes} = 1308$ ,  $Churn_{No} = 3621$ :

► Code

Churn	n
Yes	3621
No	3621

Instead of copying a record from the training dataset, `step_smote()` finds the Nearest Neighbor to that record and creates a new record that has features generated as a weighted average between the Nearest Neighbor and the original record.