Binary data and logistic regression

GENERALIZED LINEAR MODELS IN PYTHON

Ita Cirovic Donev
Data Science Consultant

Binary response data

- Two-class response $\rightarrow$ `0`, `1`

Examples:

- Credit scoring $\rightarrow$ `"Default"`/`"Non-Default"`
- Passing a test $\rightarrow$ `"Pass"`/`"Fail"`
- Fraud detection $\rightarrow$ `"Fraud"`/`"No-Fraud"`
- Choice of a product $\rightarrow$ `"Product ABC"`/`"Product XYZ"`

Binary data

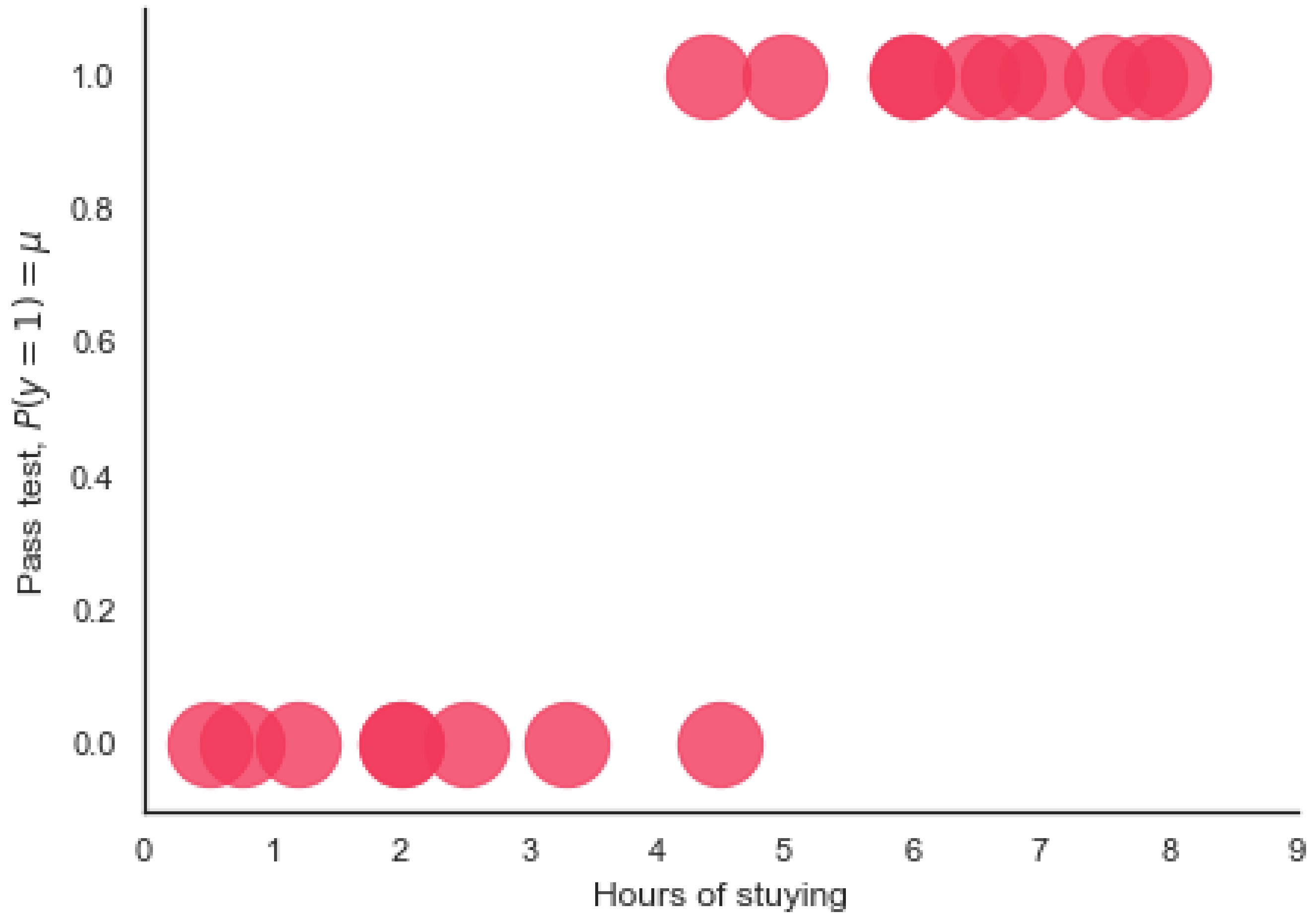
UNGROUPED

- Single event
- Flip one coin
- Two of possible outcomes: 0/1
- $Bernoulli(p)$ or
- $Binomial(n = 1, p)$

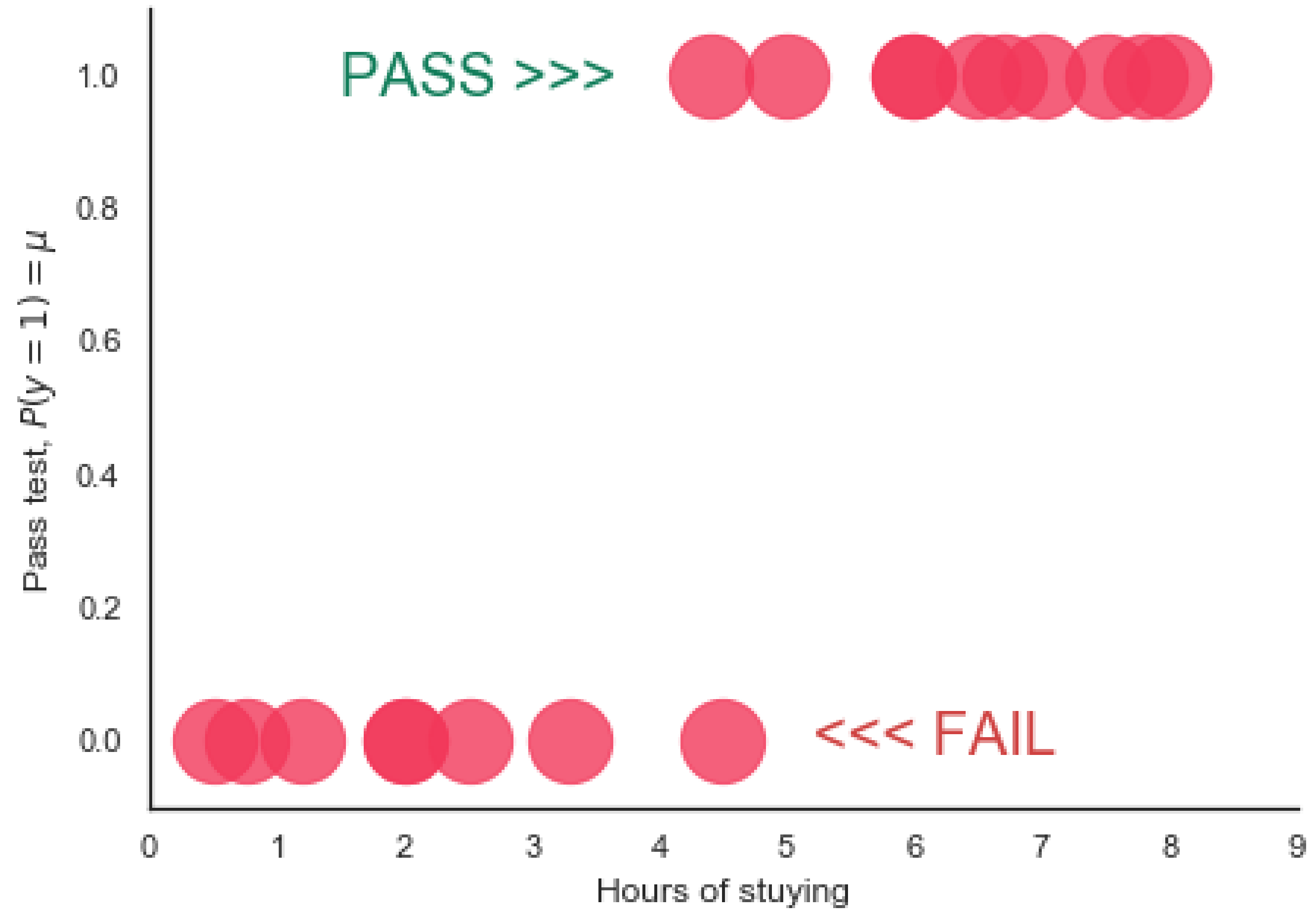
GROUPED

- Multiple events
- Flip multiple coins
- Number of successes in a given n number of trials
- $Binomial(n, p)$

Logistic function



Logistic function

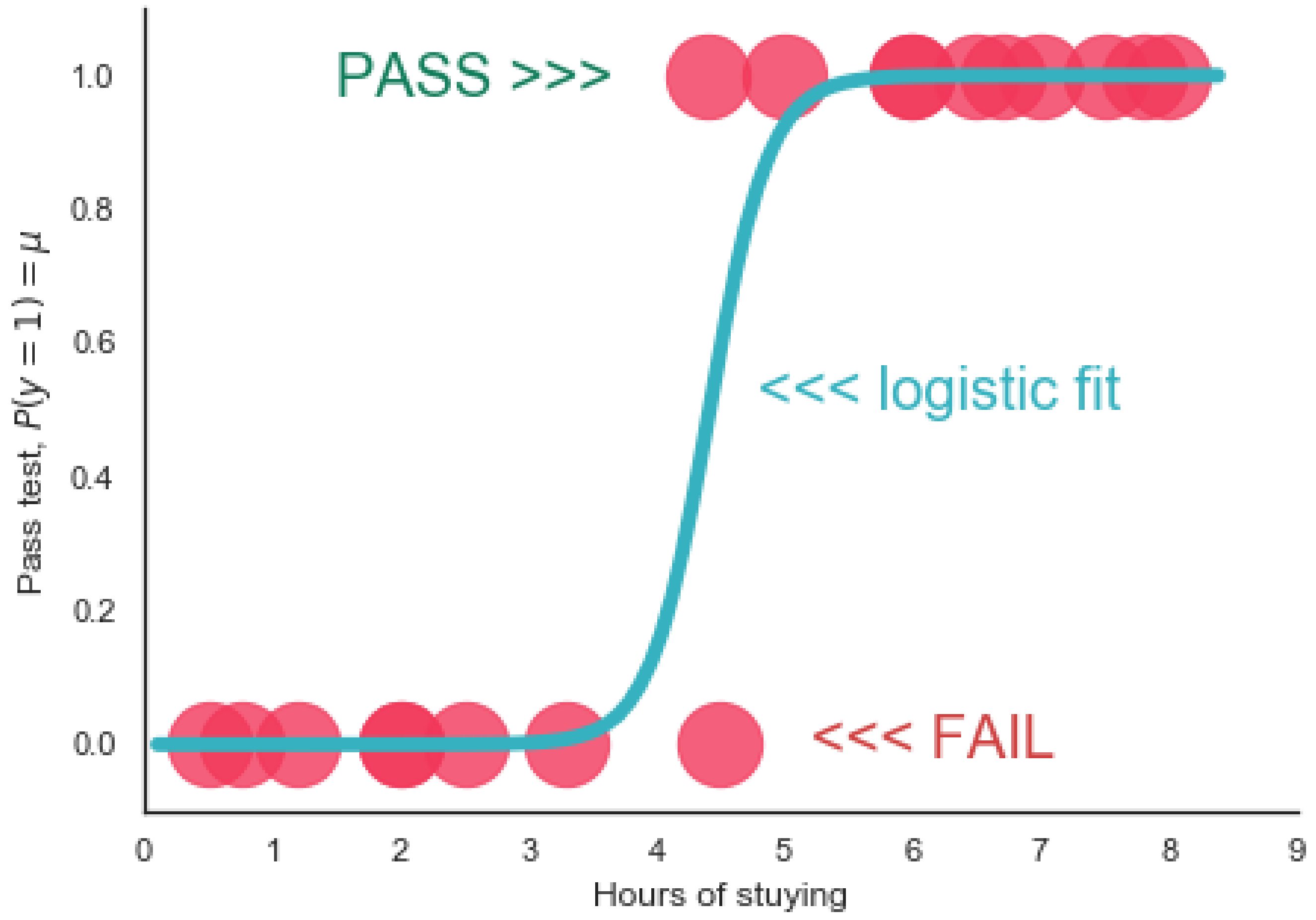


- Test outcome: $PASS = 1$ or $FAIL = 0$
- Want to model

$$P(y=1) = \beta_0 + \beta_1 x_1$$

$$P(\text{Pass}) = \beta_0 + \beta_1 \times \text{Hours of study}$$

Logistic function



- Test outcome: $PASS = 1$ or $FAIL = 0$
- Want to model

$$P(y=1) = \beta_0 + \beta_1 x_1$$

$$P(\text{Pass}) = \beta_0 + \beta_1 \times \text{Hours of study}$$

  - Use logistic function

$$f(z) = \frac{1}{(1+\exp(-z))}$$

Odds and odds ratio

$$ODDS = \frac{\text{event occuring}}{\text{event NOT occuring}}$$

$$\text{ODDS RATIO} = \frac{odds1}{odds2}$$

Odds example

- 4 games

Win Win Win Lose

- Odds are 3 to 1

$$\text{Odds} = \frac{\text{Win Win Win}}{\text{Lose}}$$

Odds and probabilities

$$\text{odds} \neq \text{probability}$$

$$\text{odds} = \frac{\text{probability}}{1 - \text{probability}}$$

$$\text{probability} = \frac{\text{odds}}{1 - \text{odds}}$$

From probability model to logistic regression

Step 1. Probability model

$$E(y) = \mu = P(y = 1) = \beta_0 + \beta_1 x_1$$

Step 2. Logistic function

$$f(z) = \frac{1}{(1+\exp(-z))}$$

Step 3. Apply logistic function $\rightarrow$ INVERSE-LOGIT

$$\mu = \frac{1}{1+\exp(-(\beta_0+\beta_1 x_1))} = \frac{\exp(\beta_0+\beta_1 x_1)}{1+\exp(\beta_0+\beta_1 x_1)}$$

$$1 - \mu = \frac{1}{1+\exp(\beta_0+\beta_1 x_1)}$$

From probability model to logistic regression

- Probability $\rightarrow$ odds

$$ODDS = \frac{\mu}{1-\mu} = exp(\beta_0 + \beta_1 x_1)$$

- Log transformation $\rightarrow$ LOGISTIC REGRESSION

$$LOGIT(\mu) = log(\frac{\mu}{1-\mu}) = \beta_0 + \beta_1 x_1$$

Logistic regression in Python

Function - `glm()`

```
model_GLM = glm(formula = 'y ~ x',
                data = my_data,
                family = sm.families.Binomial()a).fit
```

Input

```
y = [0,1,1,0,...]
y = ['No','Yes','Yes',...]
y = ['Fail','Pass','Pass',...]
```

Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

Interpreting coefficients

GENERALIZED LINEAR MODELS IN PYTHON

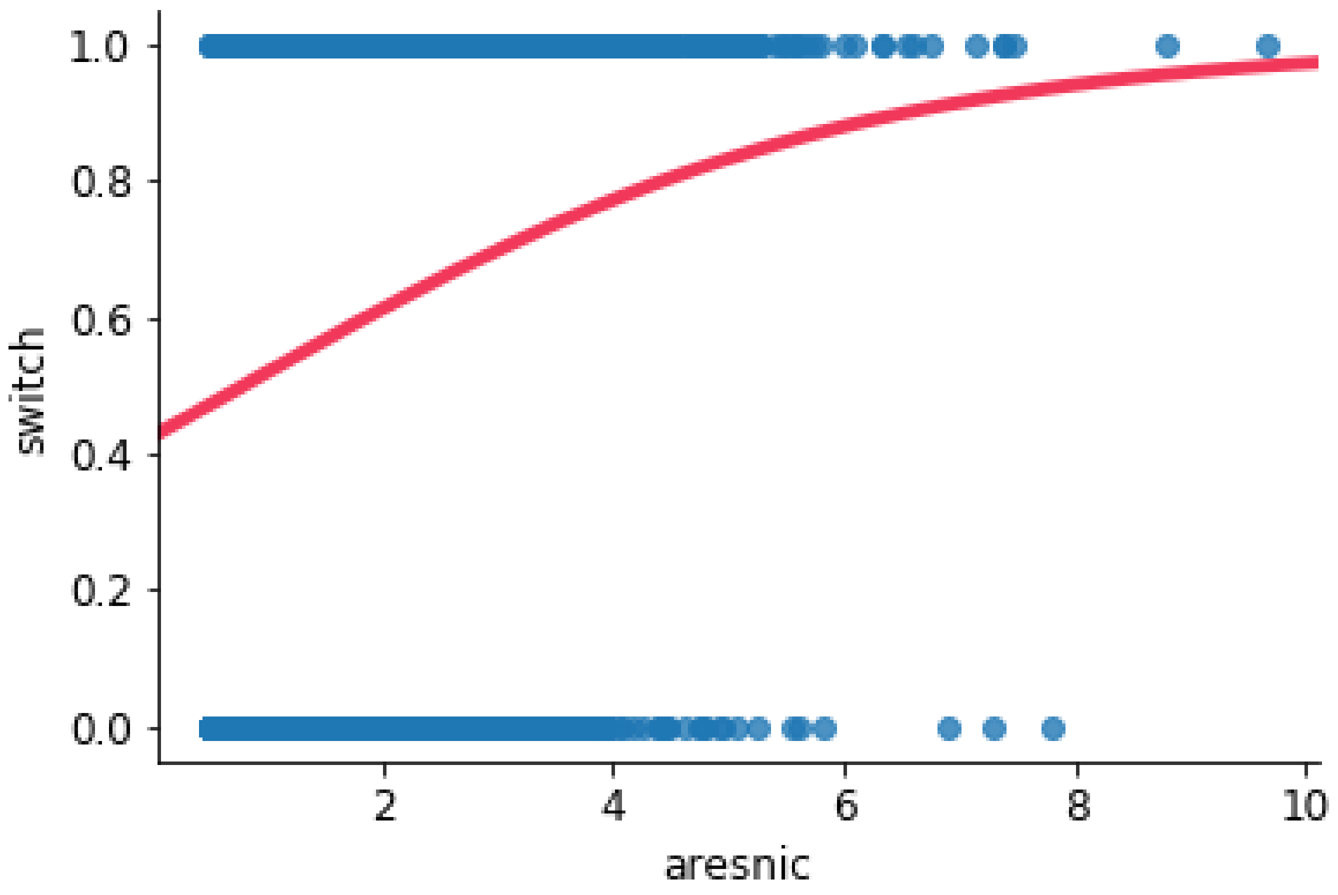
Ita Cirovic Donev
Data Science Consultant

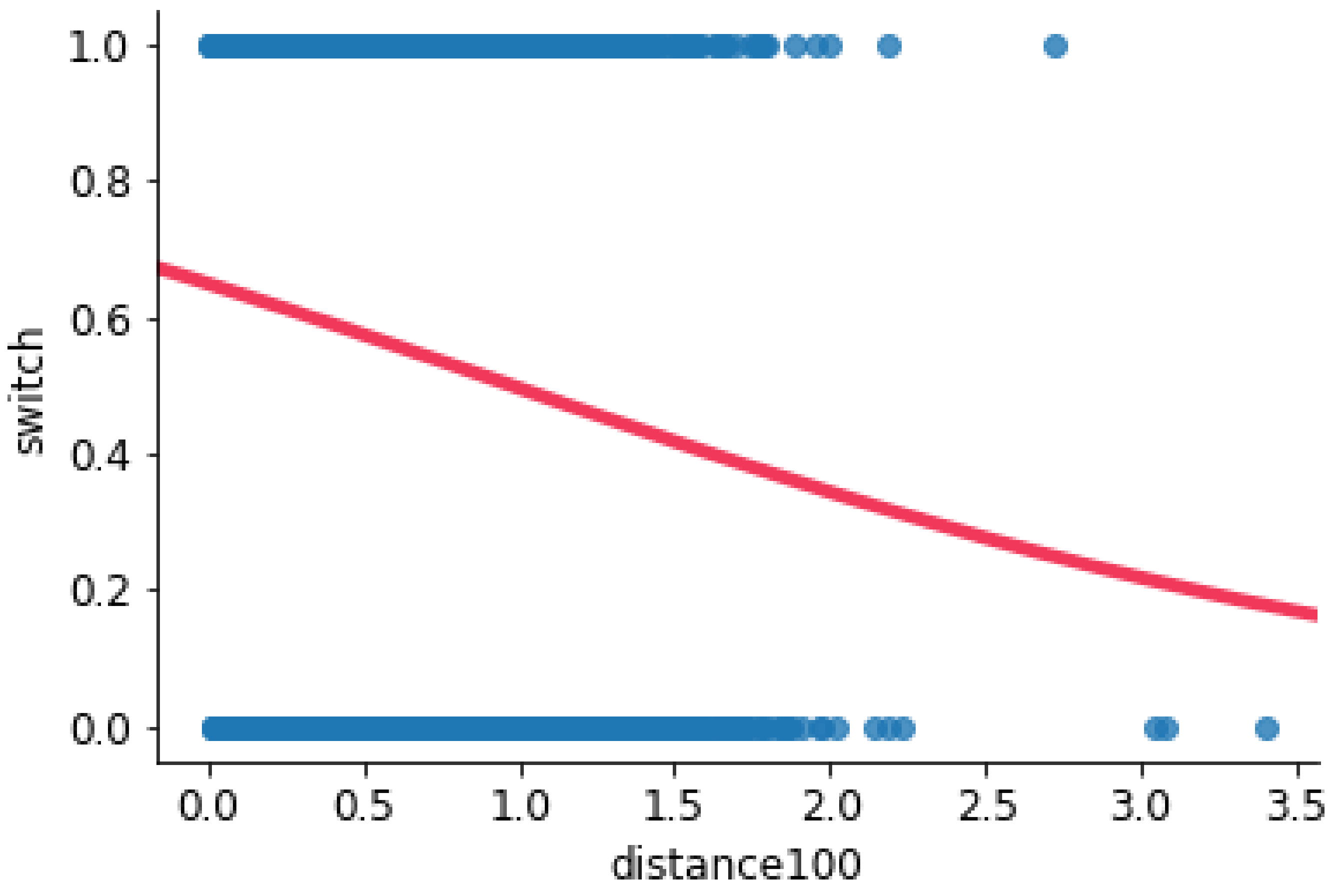
Model coefficients

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -97.869
Date:                Sat, 23 Feb 2019   Deviance:                       195.74
Time:                        13:03:32   Pearson chi2:                     168.
No. Iterations:                     5   Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.6947      0.880     -4.198      0.000      -5.420      -1.970
weight         1.8151      0.377      4.819      0.000       1.077       2.553
==============================================================================
```

Coefficient beta

- $\beta > 0 \rightarrow$ ascending curve
- $\beta < 0 \rightarrow$ descending curve





Linear vs logistic

LINEAR MODEL

```
glm('y ~ weight',
    data = crab,
    family = sm.families.Gaussian())
```

$$\mu = -0.14 + 0.32 * weight$$

For every *one-unit* increase in weight

- $\text{estimated probability}$ increases by 0.32

LOGIT MODEL

```
glm('y ~ weight',
    data = crab,
    family = sm.families.Binomial())
```

$$log(odds) = -3.69 + 1.8 * weight$$

For every *one-unit* increase in weight

- $\log(\text{odds})$ increase by 1.8

Log odds interpretation

- Logistic model

$$log(\frac{\mu}{1-\mu}) = \beta_0 + \beta_1 x_1$$

- Increase x by one-unit

$$log(\frac{\mu}{1-\mu}) = \beta_0 + \beta_1 (x_1 + 1)$$

Log odds interpretation

- Logistic model

$$log(\frac{\mu}{1-\mu}) = \beta_0 + \beta_1 x_1$$

- Increase x by one-unit

$$log(\frac{\mu}{1-\mu}) = \beta_0 + \beta_1(x_1 + 1) = \beta_0 + \beta_1 x_1 + \beta_1$$

- Take the exponential

$$(\frac{\mu}{1-\mu}) = \exp(\beta_0 + \beta_1 x_1)\exp(\beta_1)$$

Conclusion $\rightarrow$ the odds are multiplied by $\exp(\beta_1)$

Log odds interpretation

- Crab model `y ~ weight`

$$log(\frac{\mu}{1-\mu}) = -3.6947 + 1.815 * weight$$

- The odds of satellite crab multiply by $\exp(1.815) = 6.14$ for a unit increase in weight

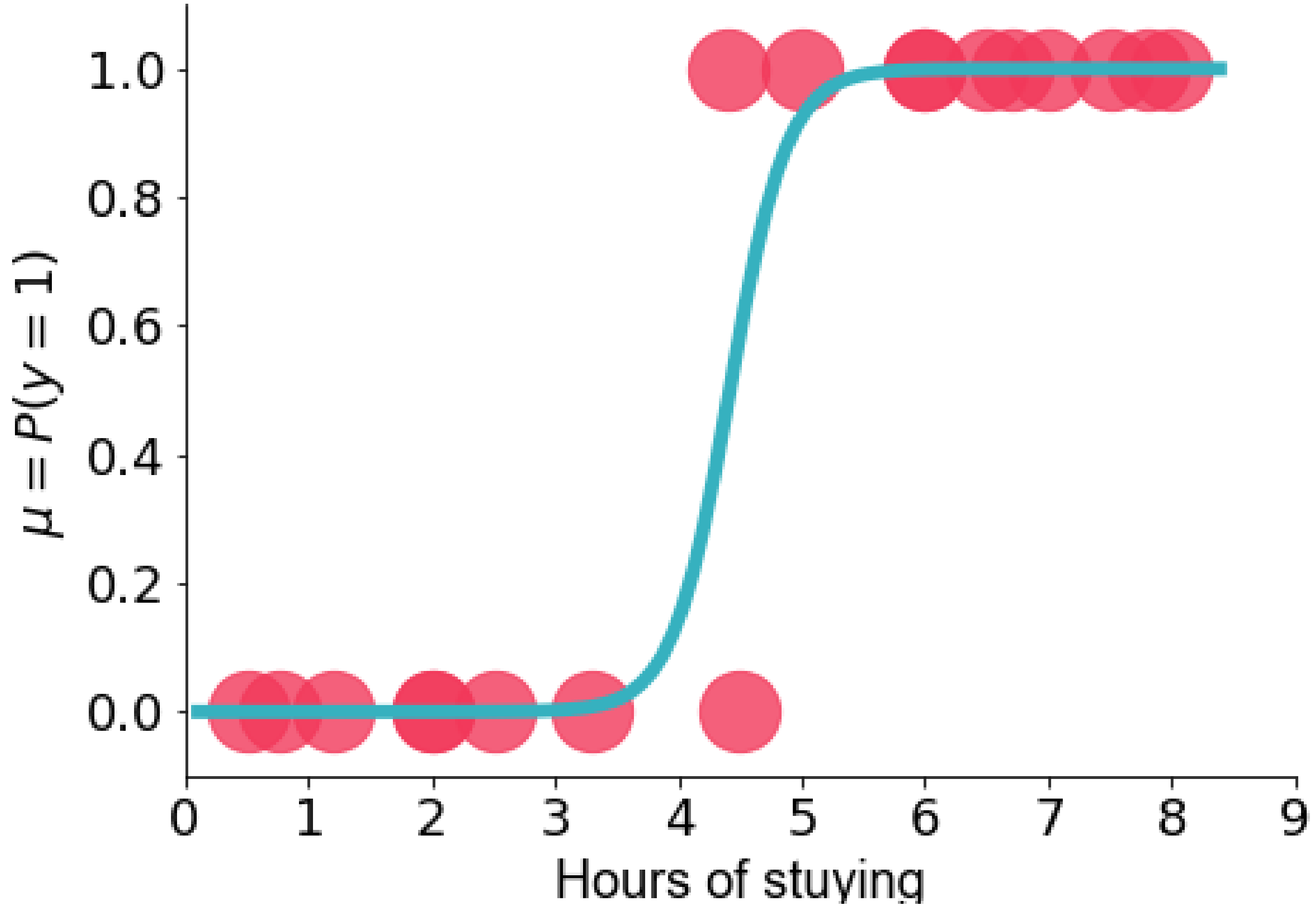
Log odds interpretation

- Crab model `y ~ weight`

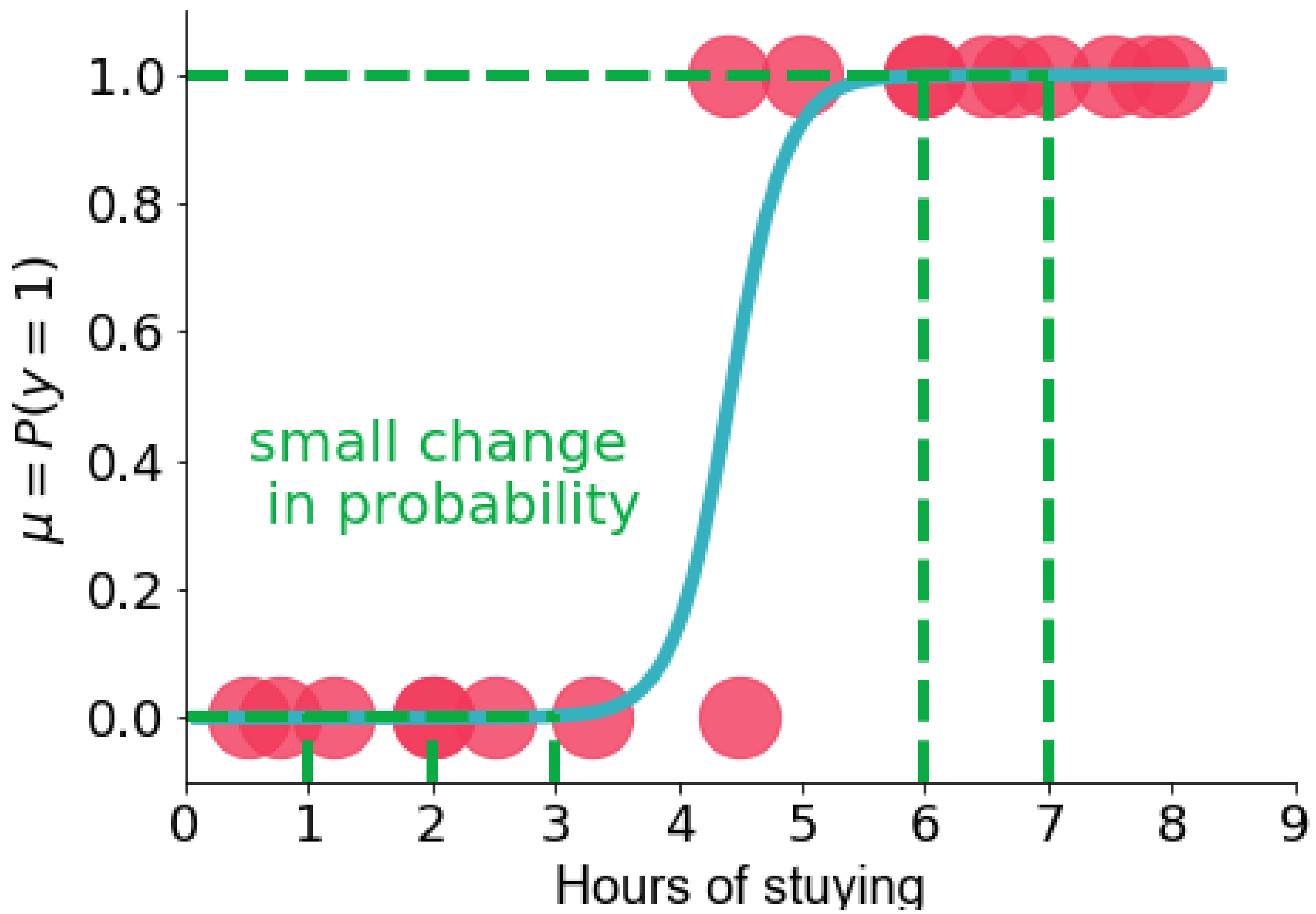
$$log(\frac{\mu}{1-\mu}) = -3.6947 + 1.8151 * weight$$

- The odds of satellite crab multiply by $\exp(1.8151) = 6.14$ for a unit increase in weight
- The intercept coefficient of -3.6947 denotes the **baseline log odds**
  - $\exp(-3.6947) = 0.0248$ are the odds when $weight = 0$.

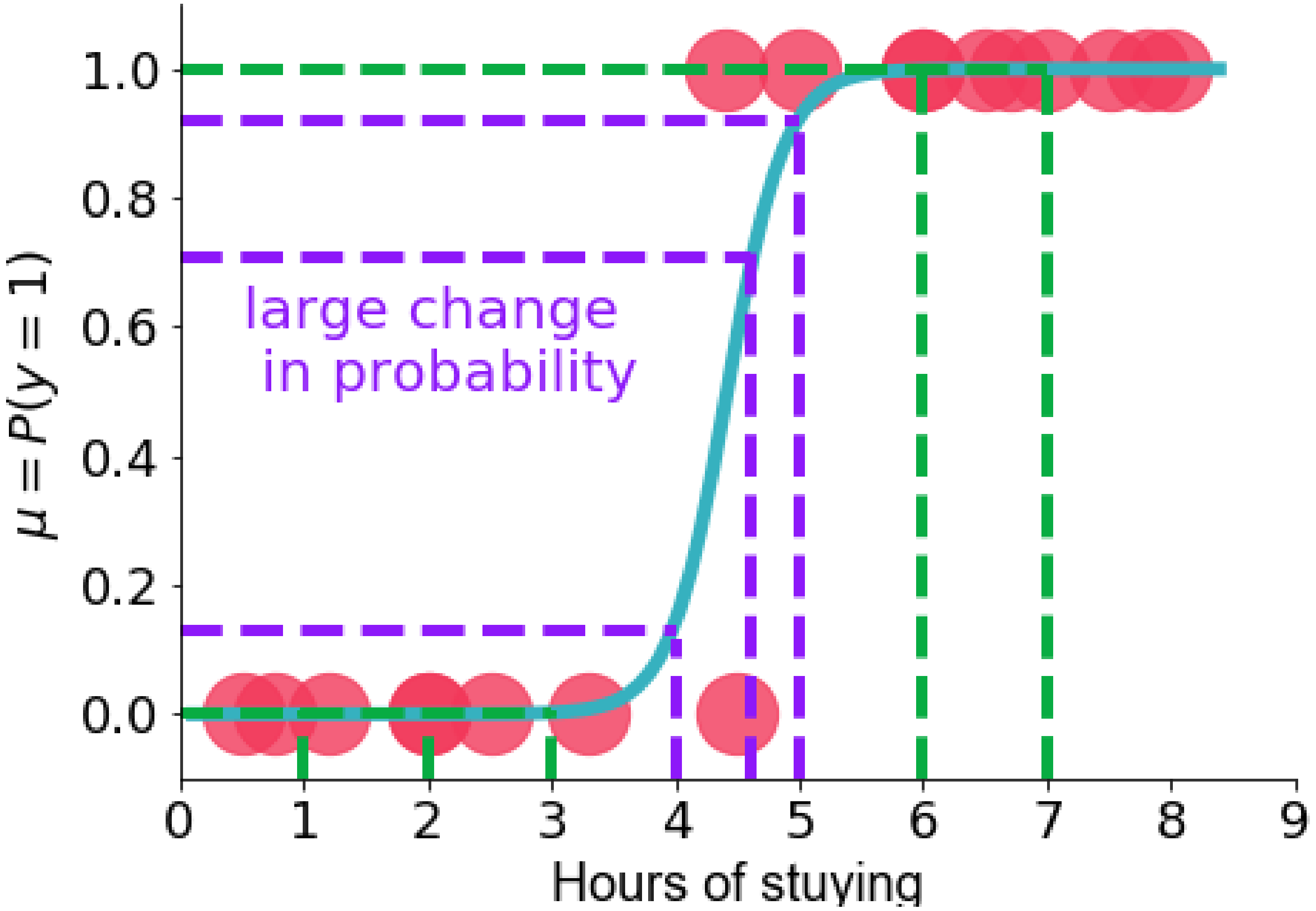
Probability vs logistic fit



Probability vs logistic fit

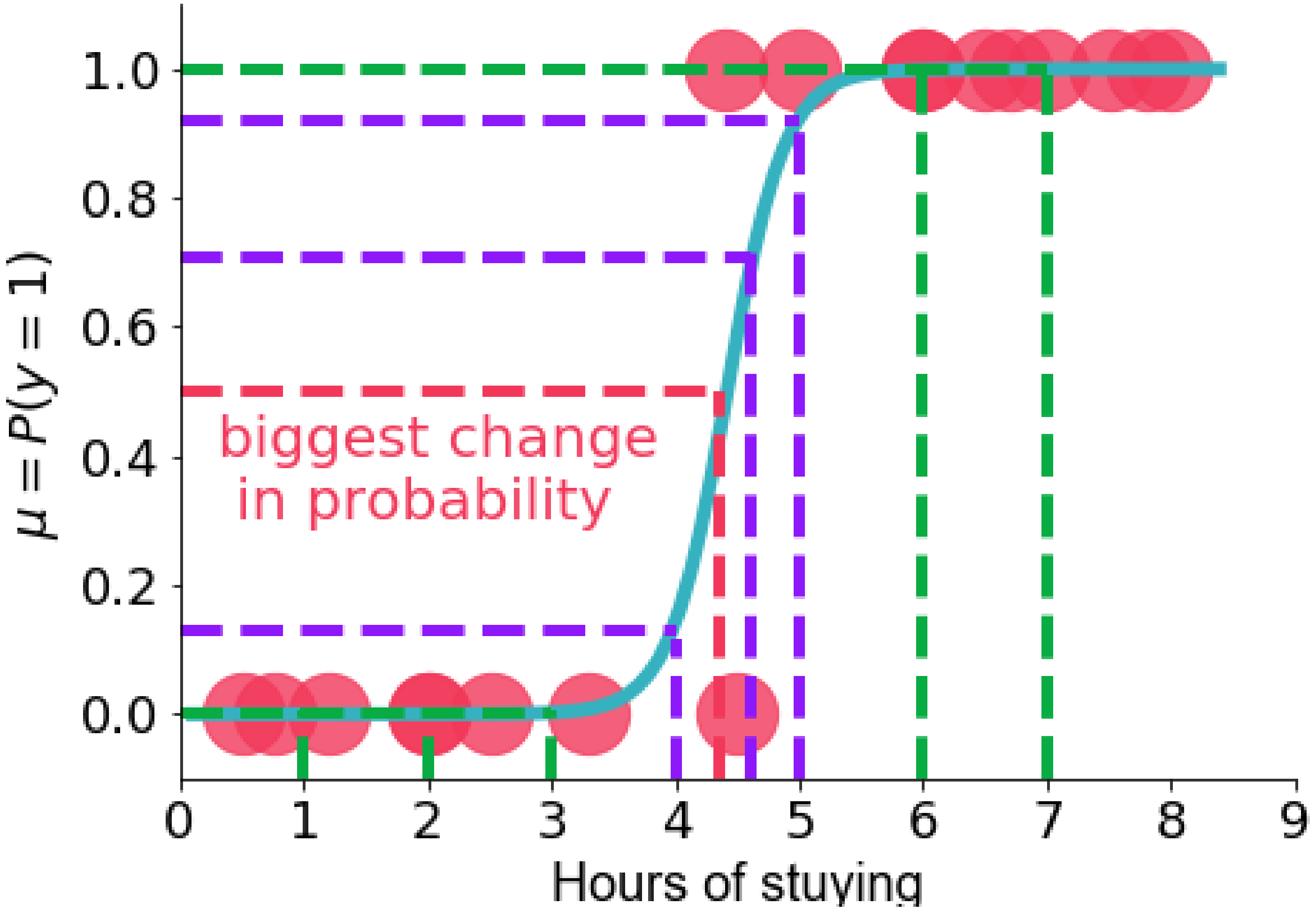


Probability vs logistic fit



- slope $\rightarrow \beta \times \mu(1 - \mu)$

Probability vs logistic fit



- slope $\rightarrow \beta \times \mu(1 - \mu)$

Compute change in estimated probability

```
# Choose x (weight) and extract model coefficients
x = 1.5
intercept, slope = model_GLM.params
```

```
# Compute estimated probability
est_prob = np.exp(intercept + slope * x)/(1 + np.exp(intercept + slope * x))
```

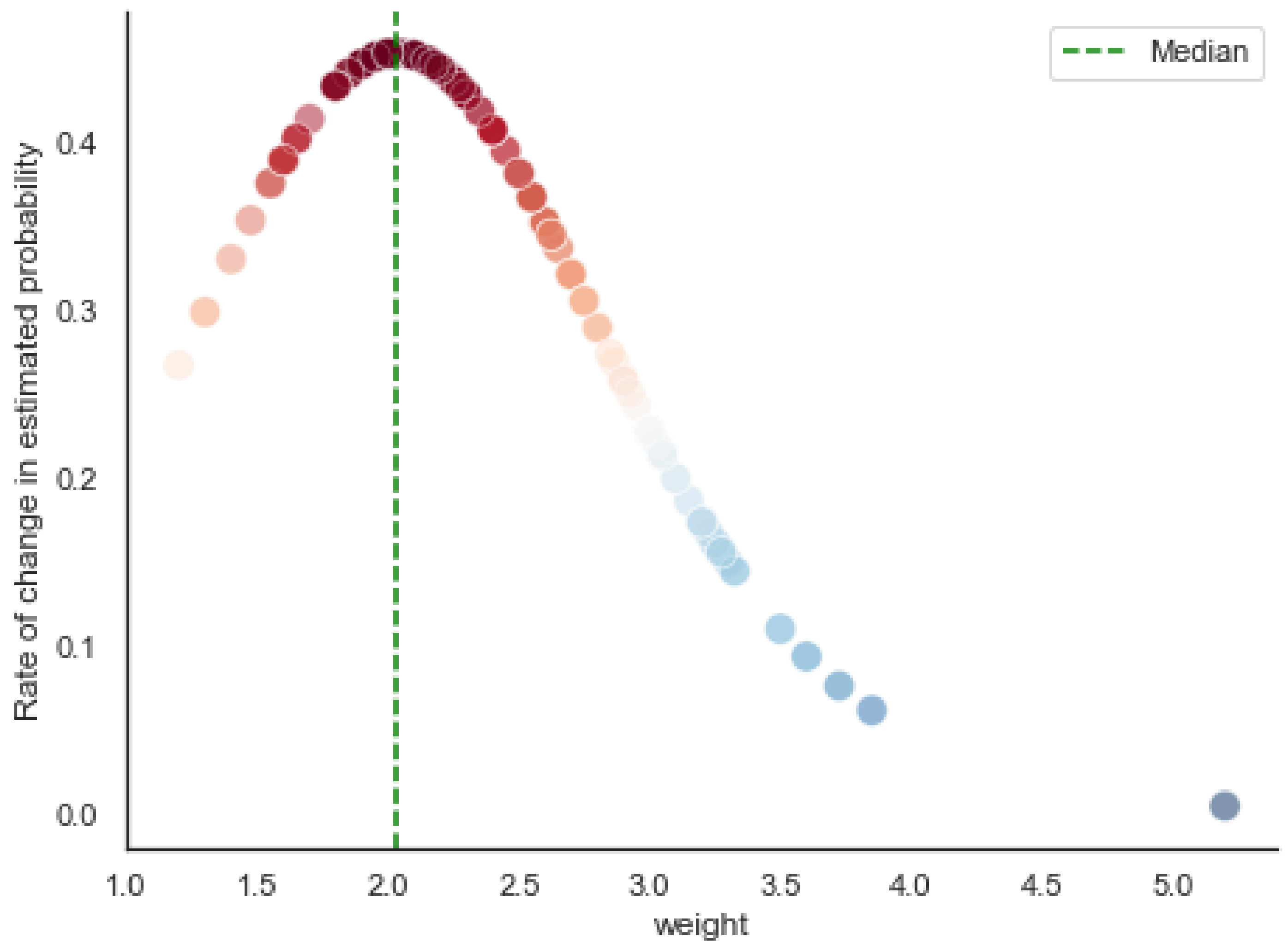
```
0.2744
```

```
# Compute incremental change in estimated probability given x
ic_prob = slope * est_prob * (1 - est_prob)
```

```
0.3614
```

Rate of change in probability for every x

$$logit = -3.6947 + 1.8151 * weight$$



Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

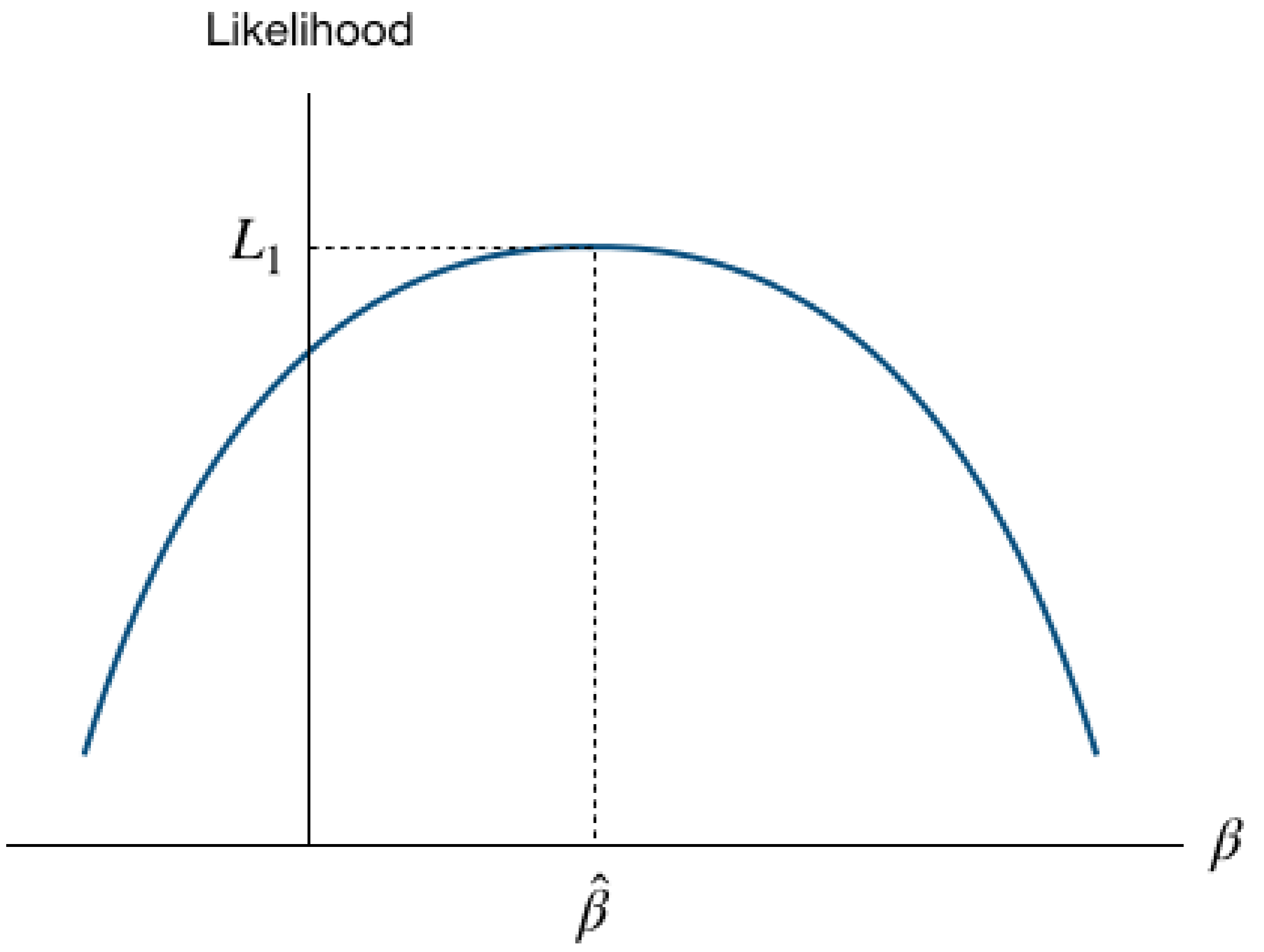
Interpreting model inference

GENERALIZED LINEAR MODELS IN PYTHON

Ita Cirovic Donev
Data Science Consultant

Estimation of beta coefficient

- Maximum likelihood estimation (MLE)
- Estimated coefficient, $\hat{\beta}$
  - log-likelihood takes on the maximum value



Estimation of beta coefficient

- Iteratively reweighted least squares (IRLS)

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -97.869
Date:                Sun, 24 Feb 2019   Deviance:                       195.74
Time:                        12:18:44   Pearson chi2:                     168.
No. Iterations:                     5   Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.6947      0.880     -4.198      0.000      -5.420      -1.970
weight         1.8151      0.377      4.819      0.000       1.077       2.553
==============================================================================
```

Significance testing

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -97.869
Date:                Sun, 24 Feb 2019   Deviance:                       195.74
Time:                        12:18:44   Pearson chi2:                     168.
No. Iterations:                     5   Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.6947      0.880     -4.198      0.000      -5.420      -1.970
weight         1.8151      0.377      4.819      0.000       1.077       2.553
==============================================================================
```

Standard error (SE)

- Flatter peak

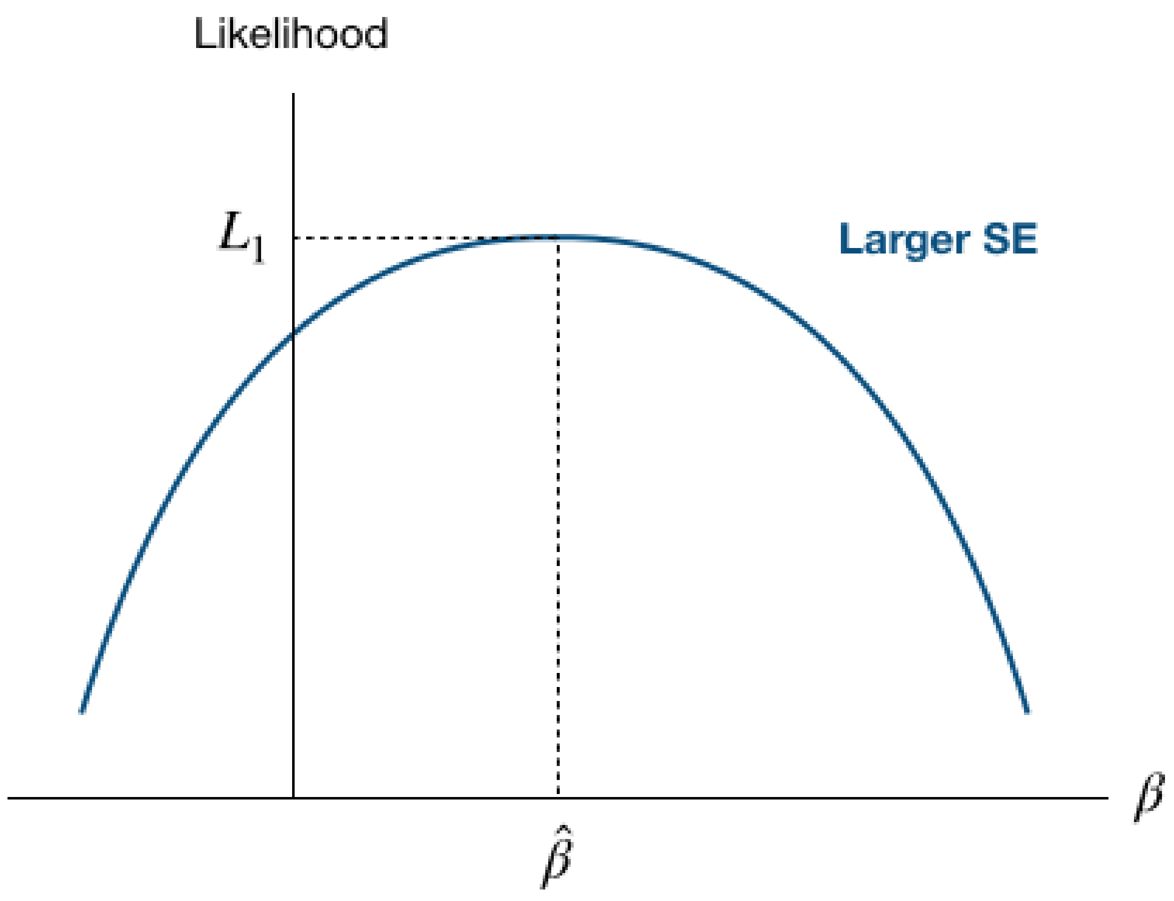
$\longrightarrow$ Location of maximum harder to define

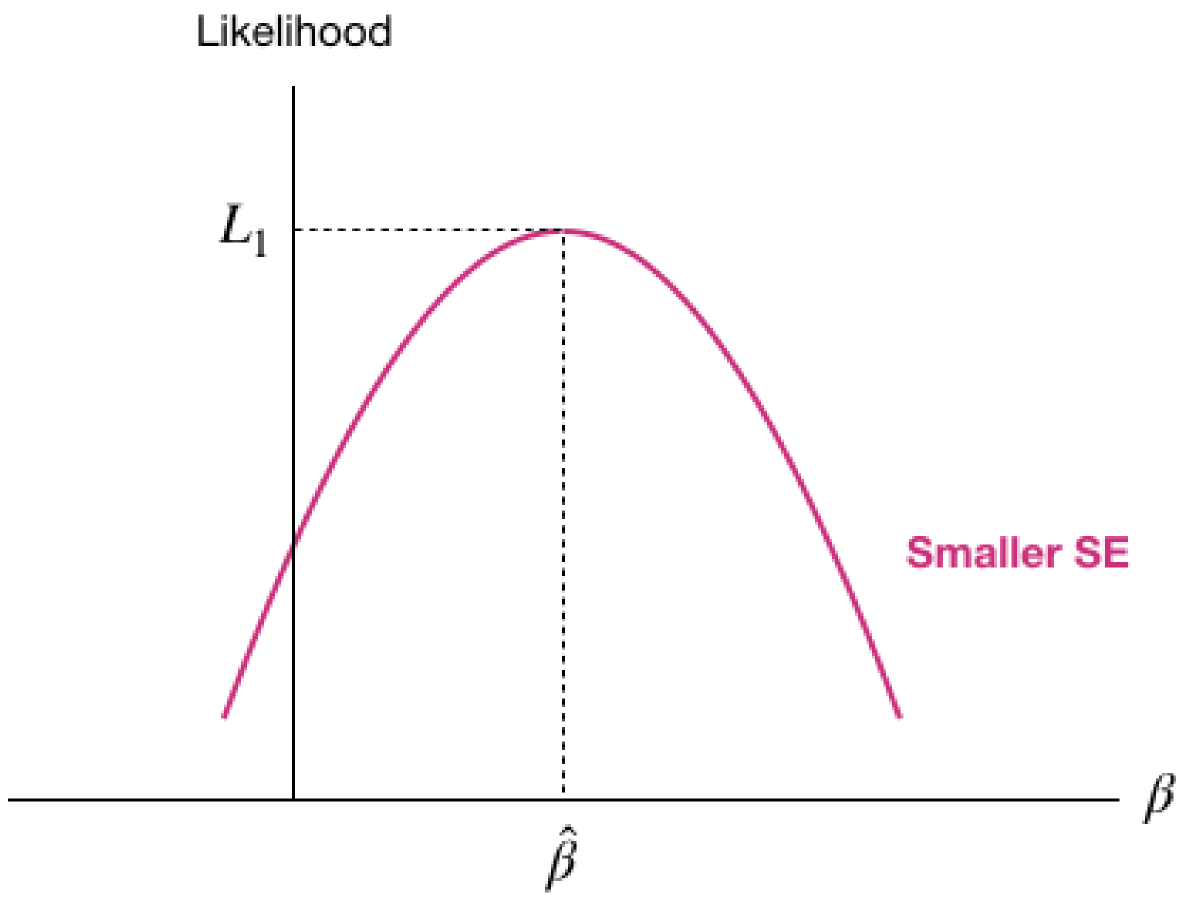
$\longrightarrow$ Larger SE

- Sharper peak

$\longrightarrow$ Location of maximum more clearly defined

$\longrightarrow$ Smaller SE





Computation of the standard error

```
# Extract variance-covariance matrix
print(model_GLM.cov_params())
```

```
             Intercept    weight
Intercept     0.774762 -0.325087
weight       -0.325087  0.141903
```

```
# Compute standard error for weight
std_error = np.sqrt(0.141903)
```

```
0.3767
```

Variance-covariance matrix

	X	Y	Z
X	Variance X	Covariance X,Y	Covariance X,Z
Y	Covariance Y,X	Variance Y	Covariance Y,Z
Z	Covariance Z,X	Covariance Z,Y	Variance Z

Significance testing

- **z-statistic**

$$z = \hat{\beta}/SE$$

- z large $\Rightarrow$ coefficient $\neq 0 \Rightarrow$ variable significant
- Rule of thumb: cut-off value of 2

Example: horseshoe crab model

`y ~ weight`

$$z = 1.8151/0.377 = 4.819$$

Confidence intervals for beta

- Uncertainty of the estimates
- **95% confidence intervals** for β

$$[lower, upper]$$

$$[\hat{\beta} - 1.96 \times SE, \hat{\beta} + 1.96 \times SE]$$

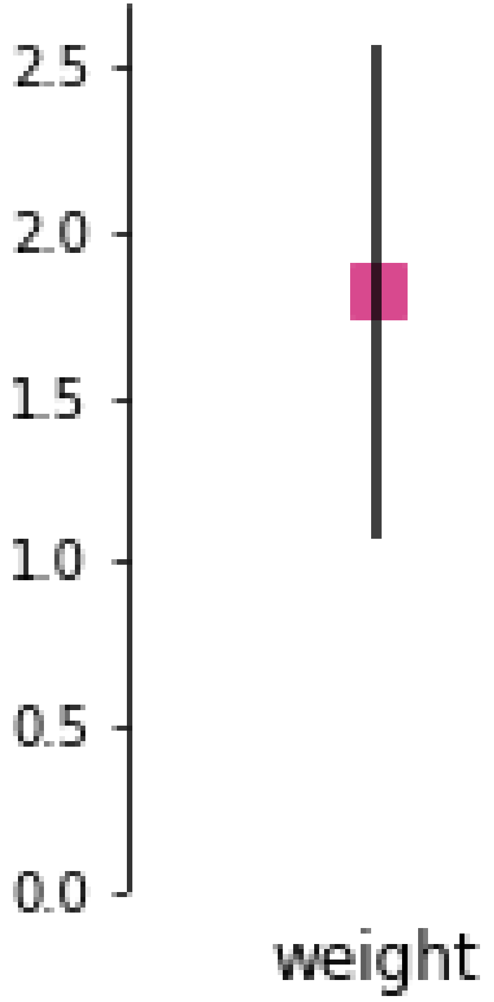
Computing confidence intervals

Example: horseshoe crab model

```
                  coef     std err
----------------------------------
Intercept      -3.6947       0.880
weight          1.8151       0.377
```

$$[1.8151 - 1.96 \times 0.377, 1.8151 + 1.96 \times 0.377]$$

$$[1.07618, 2.55402]$$



Extract confidence intervals

```
print(model_GLM.conf_int())
```

```
                  0         1
Intercept -5.419897 -1.969555
weight     1.076826  2.553463
```

Extract confidence intervals

```
print(model_GLM.conf_int())
```

```
                  lower         1
Intercept -5.419897 -1.969555
weight     1.076826  2.553463
```

Extract confidence intervals

```
print(model_GLM.conf_int())
```

```
                  0     upper
Intercept -5.419897 -1.969555
weight     1.076826  2.553463
```

Confidence intervals for odds

1. Extract confidence intervals for β
2. Exponentiate endpoints

```
print(np.exp(model_GLM.conf_int()))
```

```
                  0          1
Intercept  0.004428   0.139519
weight     2.935348  12.851533
```

Let's practice!

Computing and describing predictions

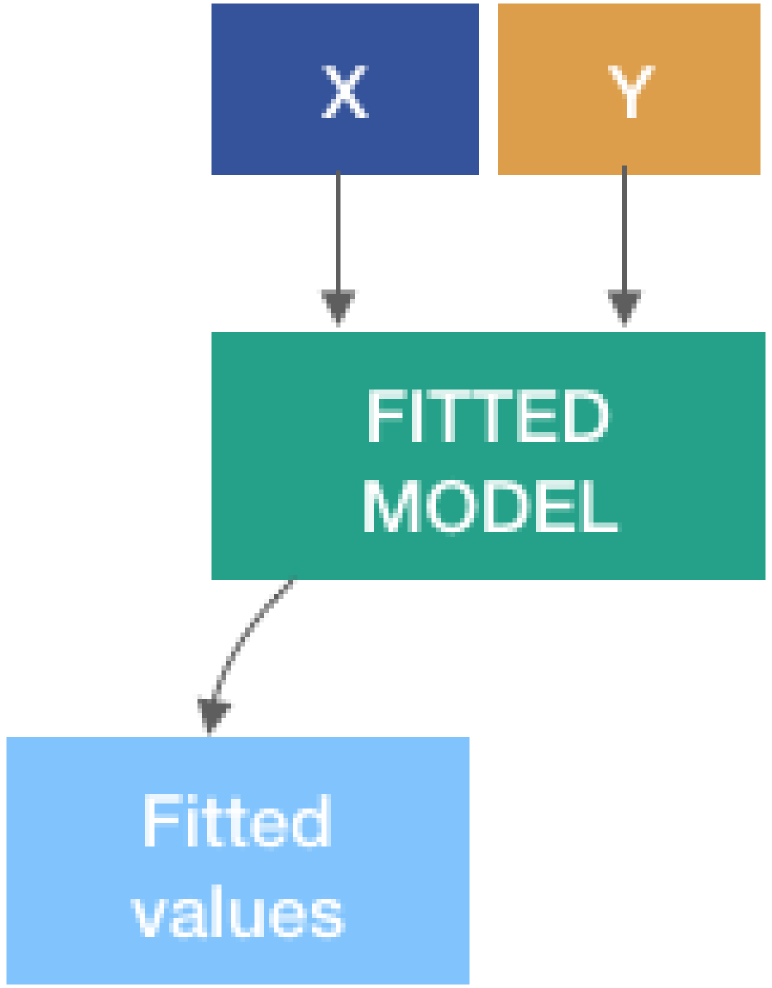
GENERALIZED LINEAR MODELS IN PYTHON

Ita Cirovic Donev
Data Science Consultant

Computing predictions

After obtaining model fit

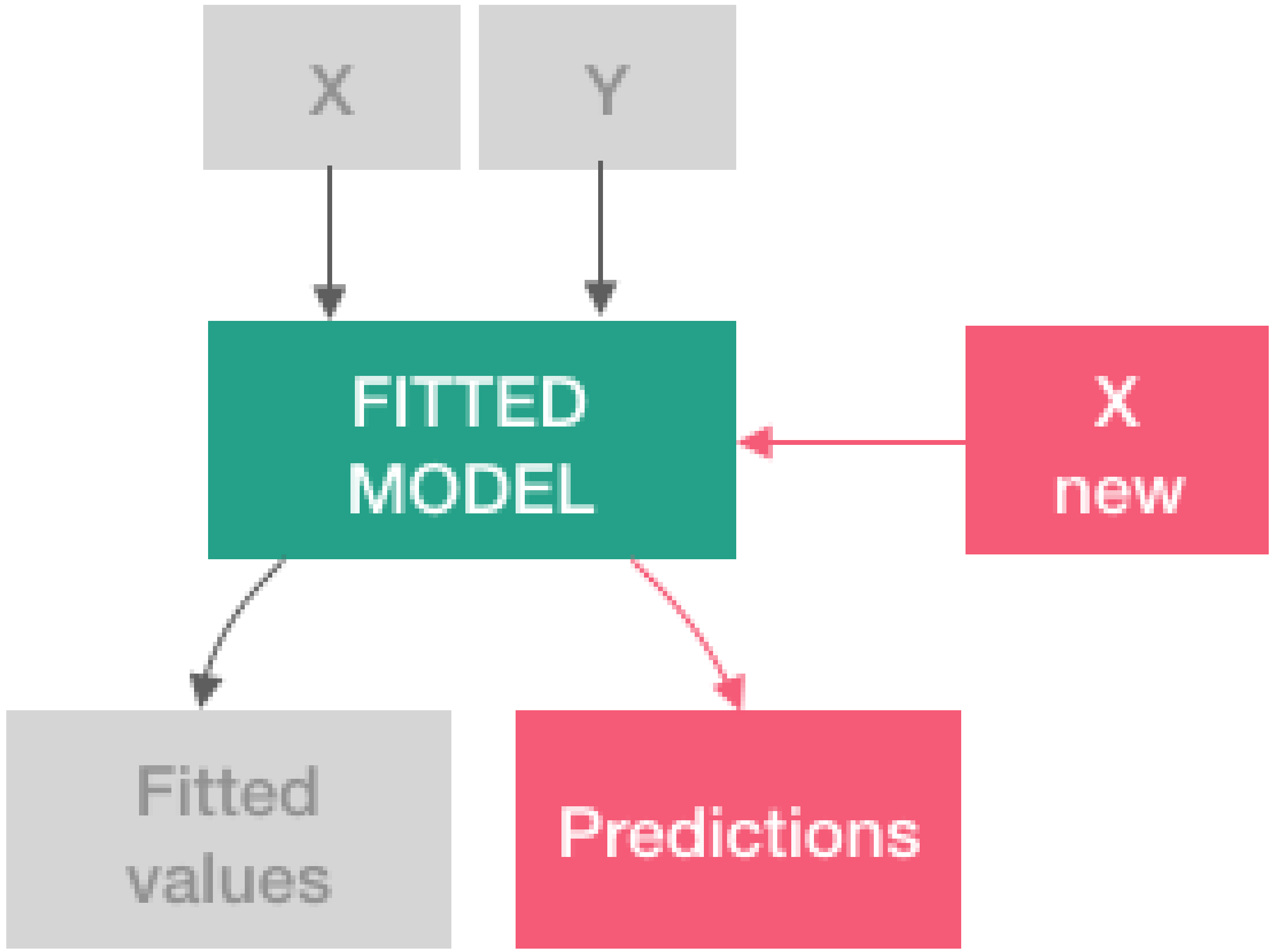
1. **Fitted values** for original x values



Computing predictions

After obtaining model fit

1. **fitted values** for original x values
2. **New** values of x for **predicted values**



Computing predictions

- Horseshoe crab model `y ~ weight`

$$\mu = \frac{\exp(-3.6947 + 1.8151 \times weight)}{1 + \exp(-3.6947 + 1.8151 \times weight)}$$

- New measurement: `weight = 2.85`

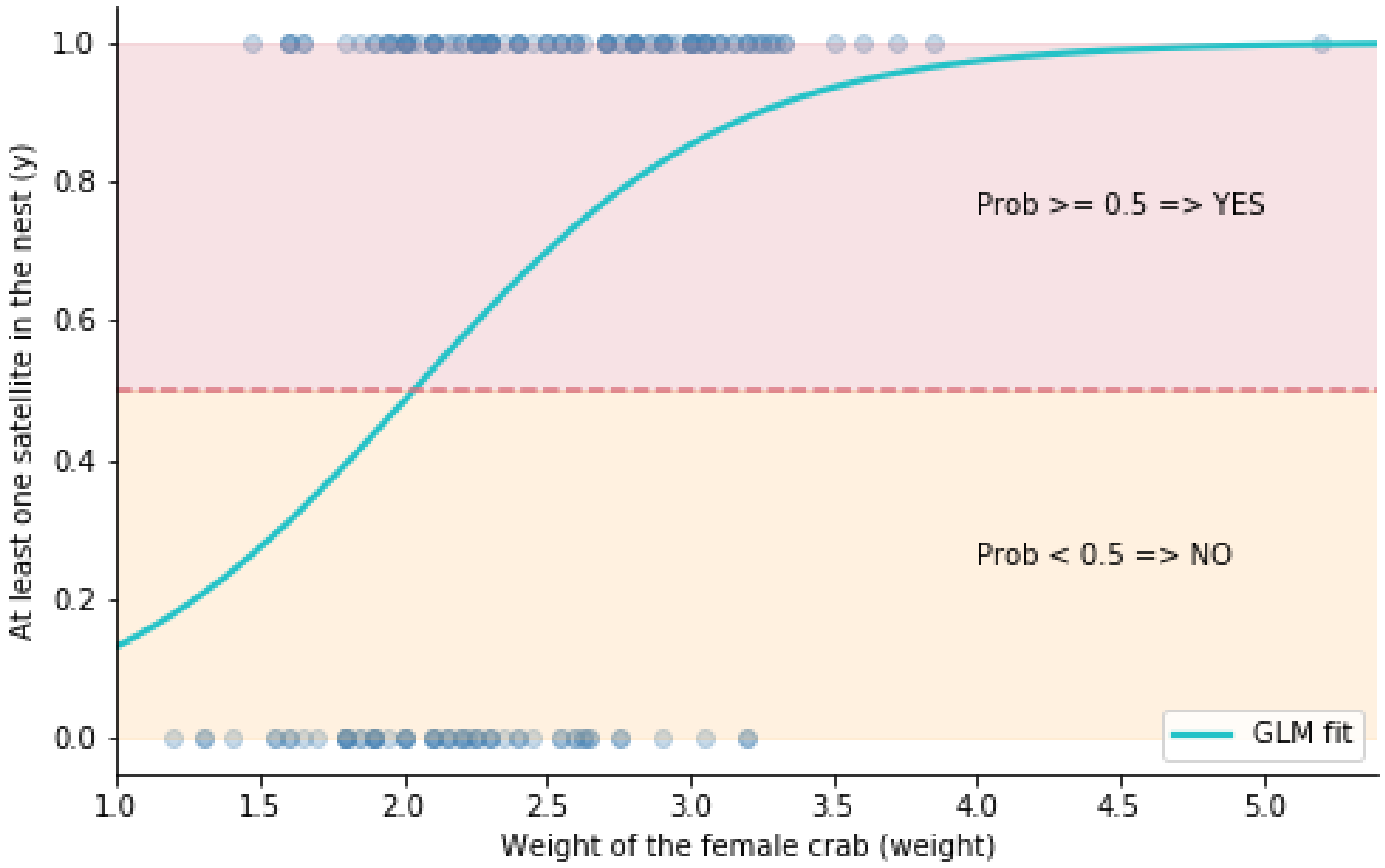
$$\mu = \frac{\exp(-3.6947 + 1.8151 \times \color{blue}{2.85})}{1 + \exp(-3.6947 + 1.8151 \times \color{blue}{2.85})} = 0.814$$

Predictions in Python

- Compute model predictions for dataset `new_data`

```
# Compute model predictions
model_GLM.predict(exog = new_data)
```

From probabilities to classes



Computing class predictions

```
# Extract fitted probabilities from model
crab['fitted'] = model.fittedvalues.values
```

```
# Define cut-off value
cut_off = 0.4
```

```
# Compute class predictions
crab['pred_class'] = np.where(crab['fitted'] > cut_off, 1, 0)
```

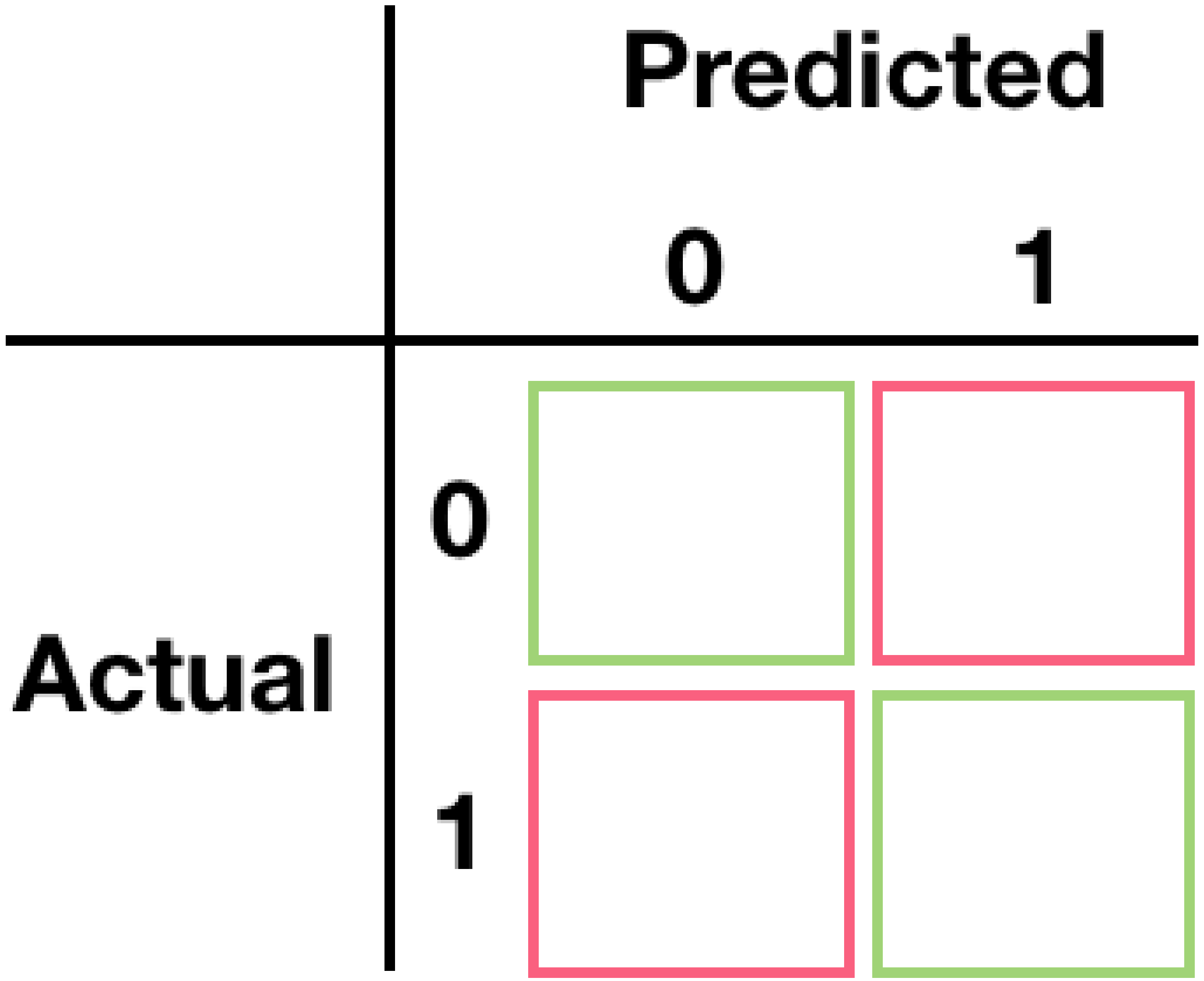
Computing class predictions

```
# Count occurences for each class
crab['pred_class'].value_counts()
```

```
1    151
0     22
```

Cut-off	$\hat{y} = 1$	$\hat{y} = 0$
$\mu = 0.4$	151	22
$\mu = 0.5$	126	47

Confusion matrix



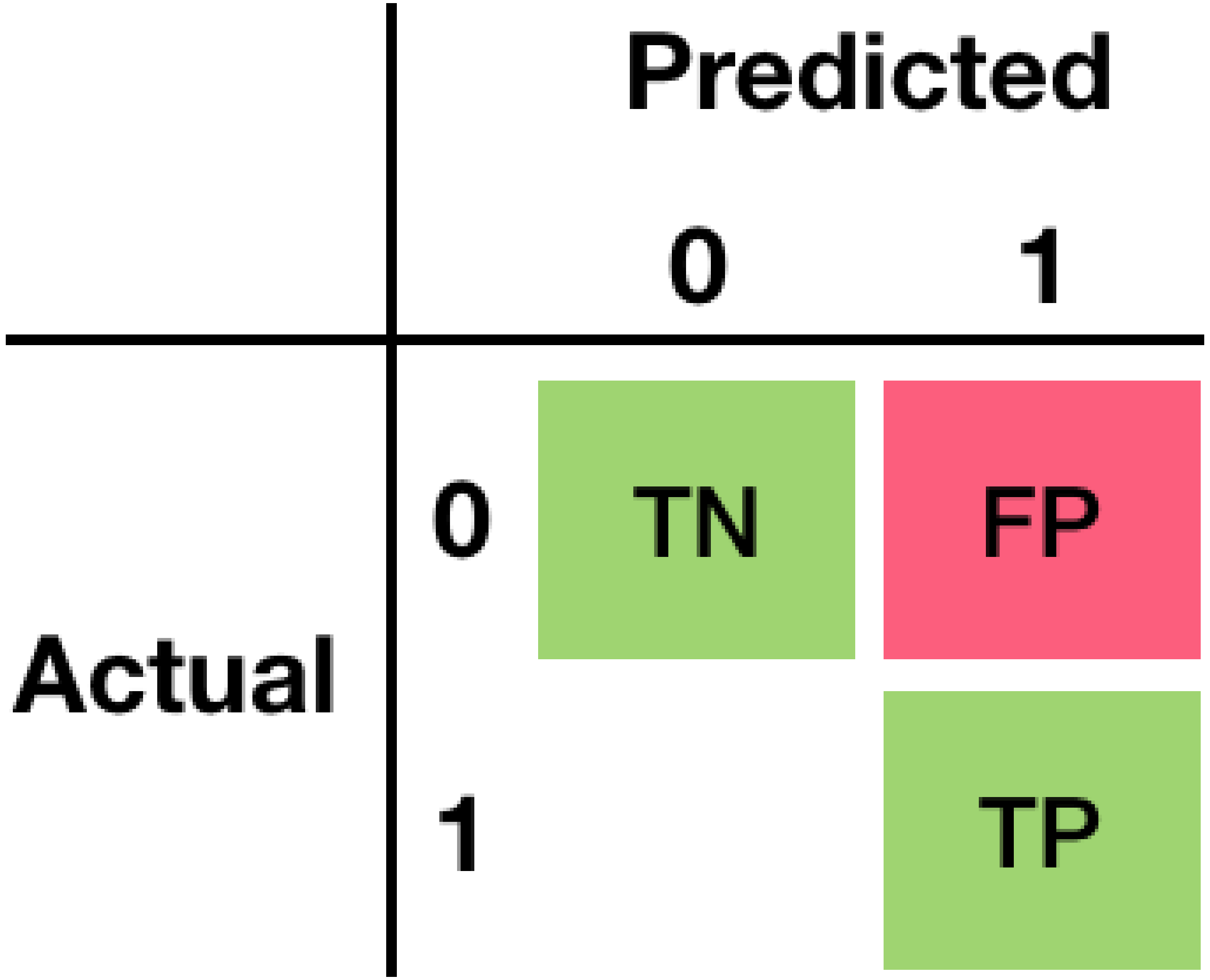
Confusion matrix - True Negatives

		Predicted	
		0	1
Actual	0	TN	
	1		

Confusion matrix - True Positives

		Predicted	
		0	1
Actual	0	TN	
	1		TP

Confusion matrix - False Positives



Confusion matrix - False Negatives

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

Confusion matrix in Python

```
print(pd.crosstab(y_actual, y_predicted,
            rownames=['Actual'], colnames=['Predicted'],
            margins = True))
```

```
Predicted   0    1  All
Actual
0          15   47   62
1           7  104  111
All        22  151  173
```

Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON