Going beyond linear regression

GENERALIZED LINEAR MODELS IN PYTHON

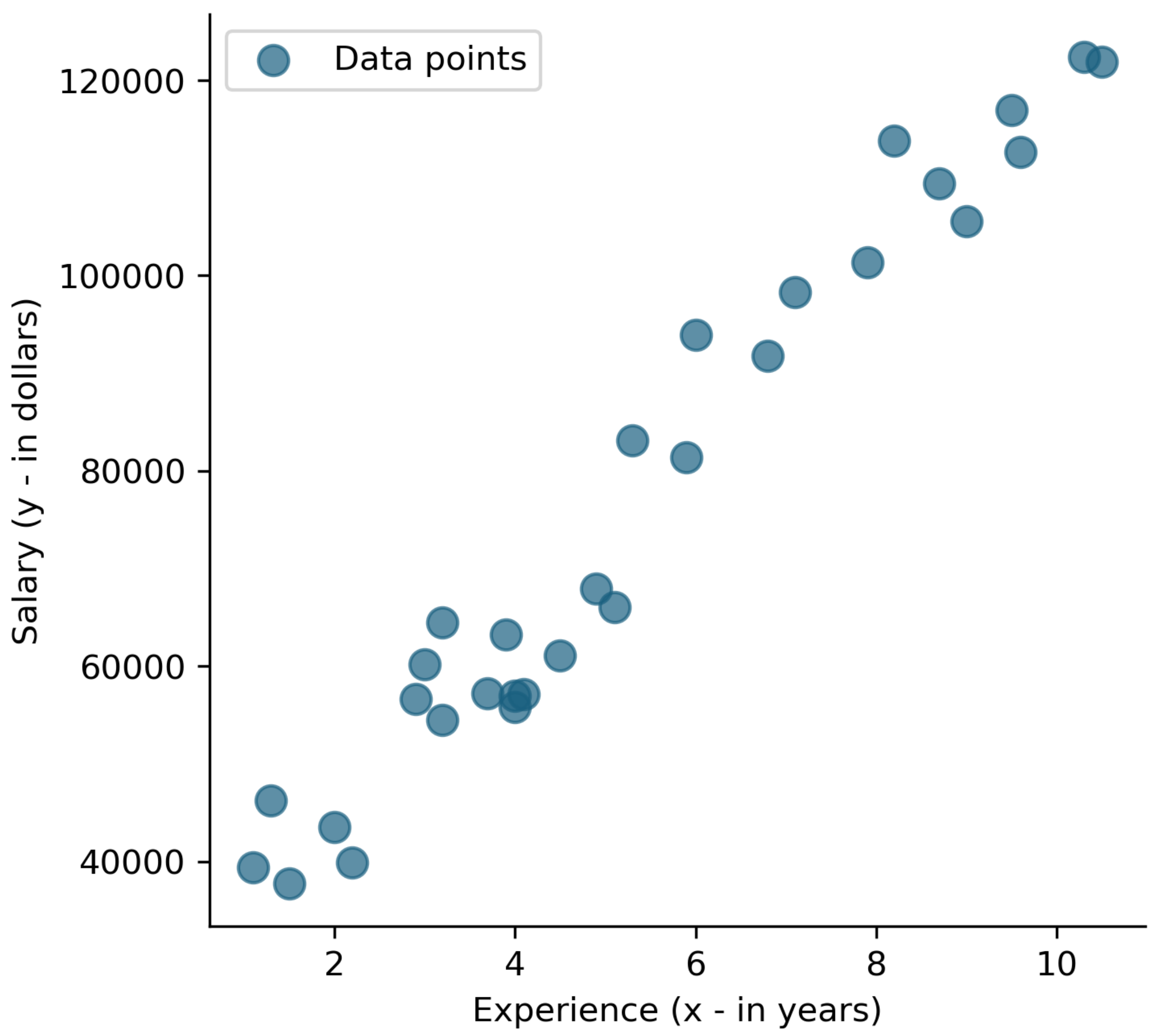
Ita Cirovic Donev
Data Science Consultant

Course objectives

- Learn building blocks of GLMs
- Train GLMs
- Interpret model results
- Assess model performance
- Compute predictions

- Chapter 1: How are GLMs an extension of linear models
- Chapter 2: Binomial (logistic) regression
- Chapter 3: Poisson regression
- Chapter 4: Multivariate logistic regression

Review of linear models

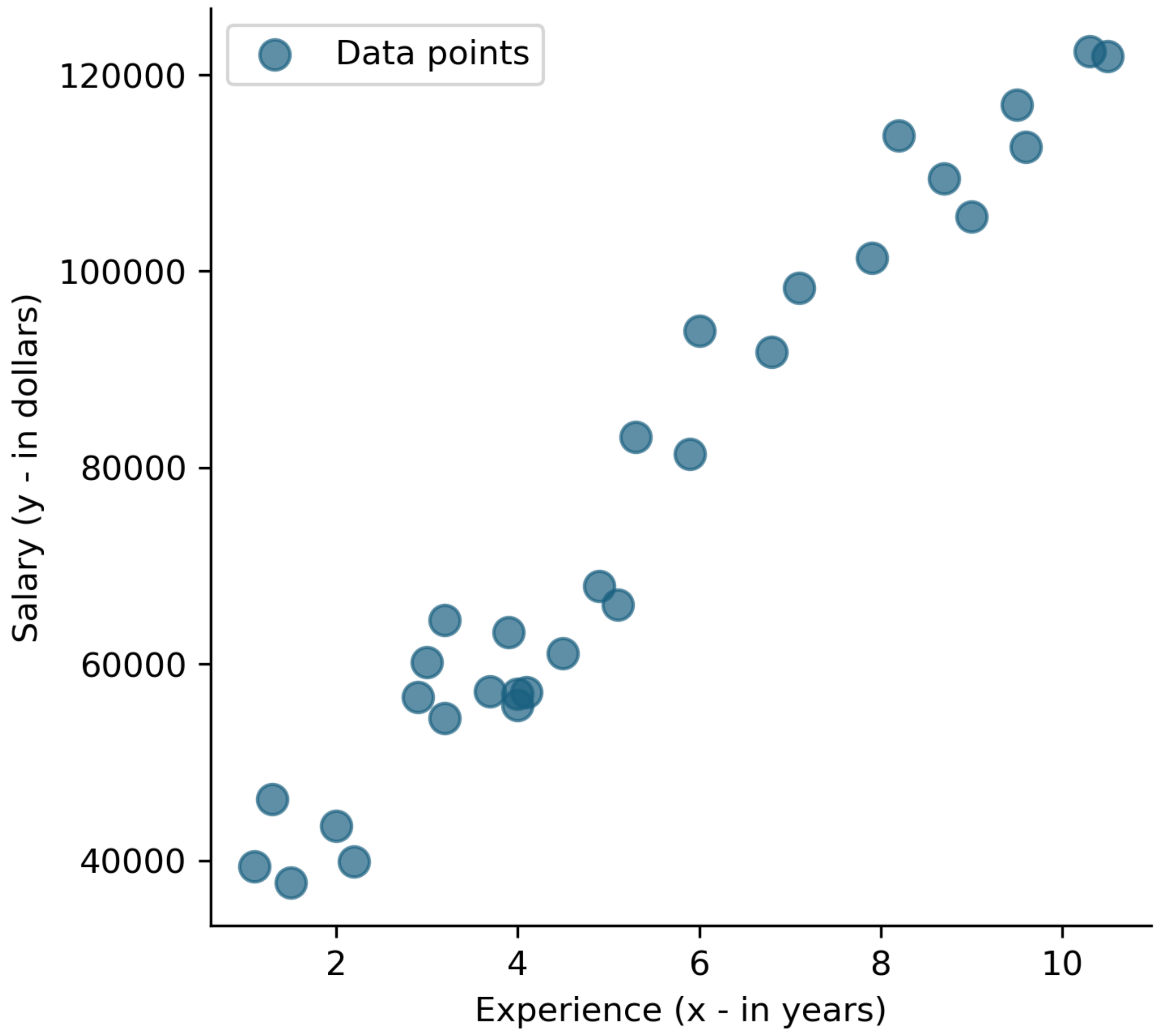


$$\text{salary} \sim \text{experience}$$

$$\text{salary} = \beta_0 + \beta_1 \times \text{experience} + \epsilon$$

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

Review of linear models



$$\text{salary} \sim \text{experience}$$

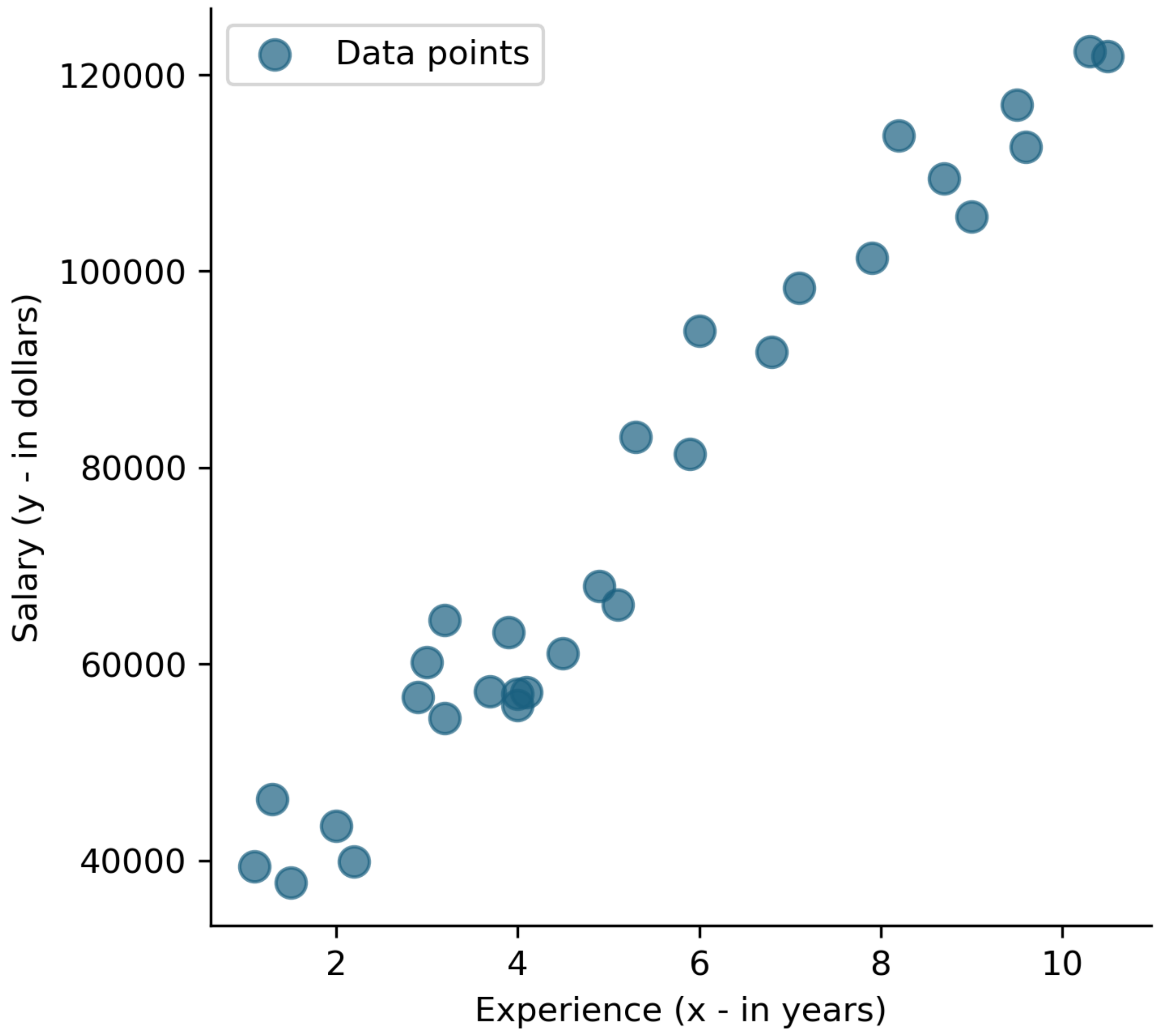
$$\text{salary} = \beta_0 + \beta_1 \times \text{experience} + \epsilon$$

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

where:

y - response variable (output)

Review of linear models



$$\text{salary} \sim \text{experience}$$

$$\text{salary} = \beta_0 + \beta_1 \times \text{experience} + \epsilon$$

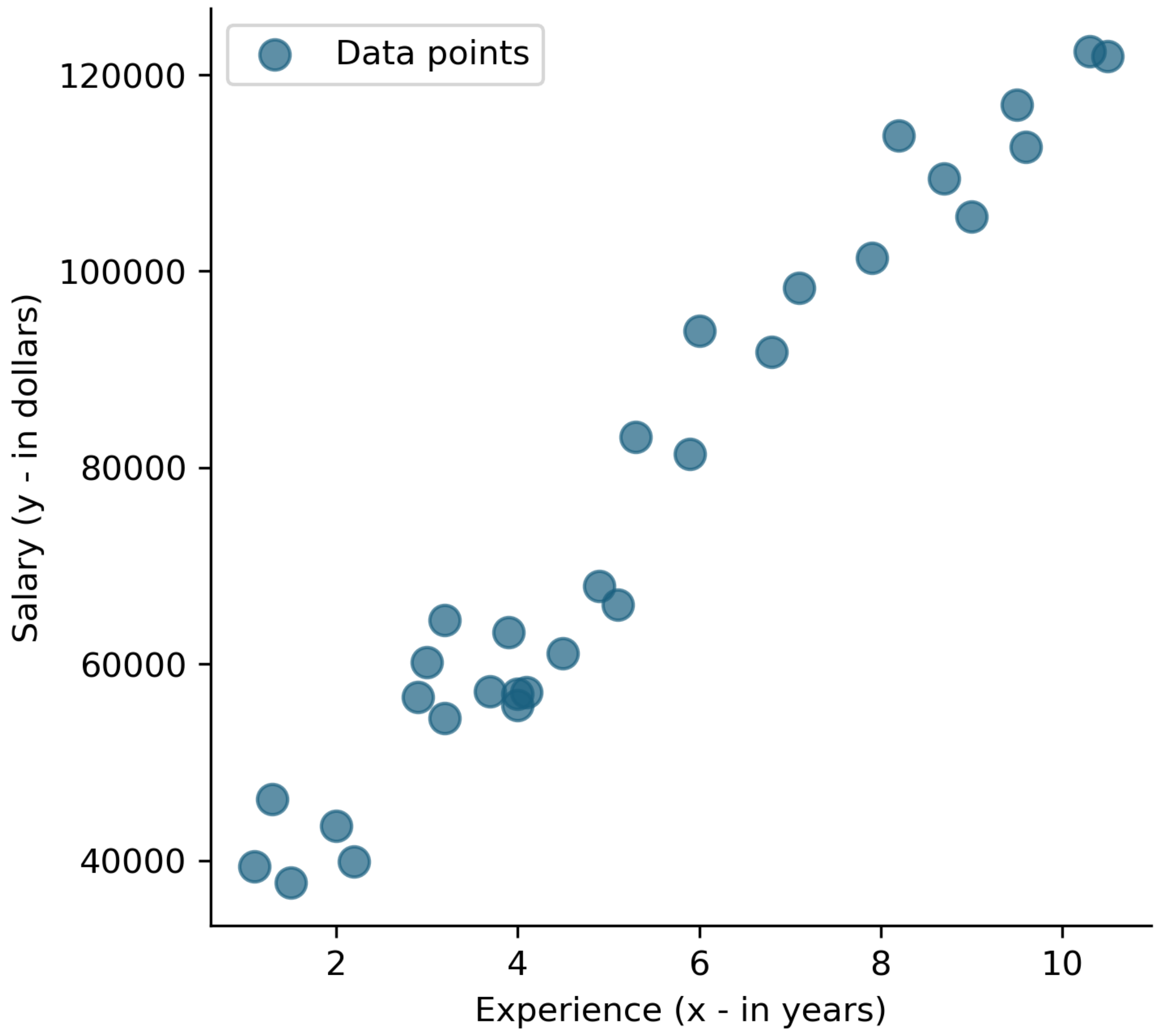
$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

where:

y - response variable (output)

x - explanatory variable (input)

Review of linear models



$$\text{salary} \sim \text{experience}$$

$$\text{salary} = \beta_0 + \beta_1 \times \text{experience} + \epsilon$$

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

where:

y - response variable (output)

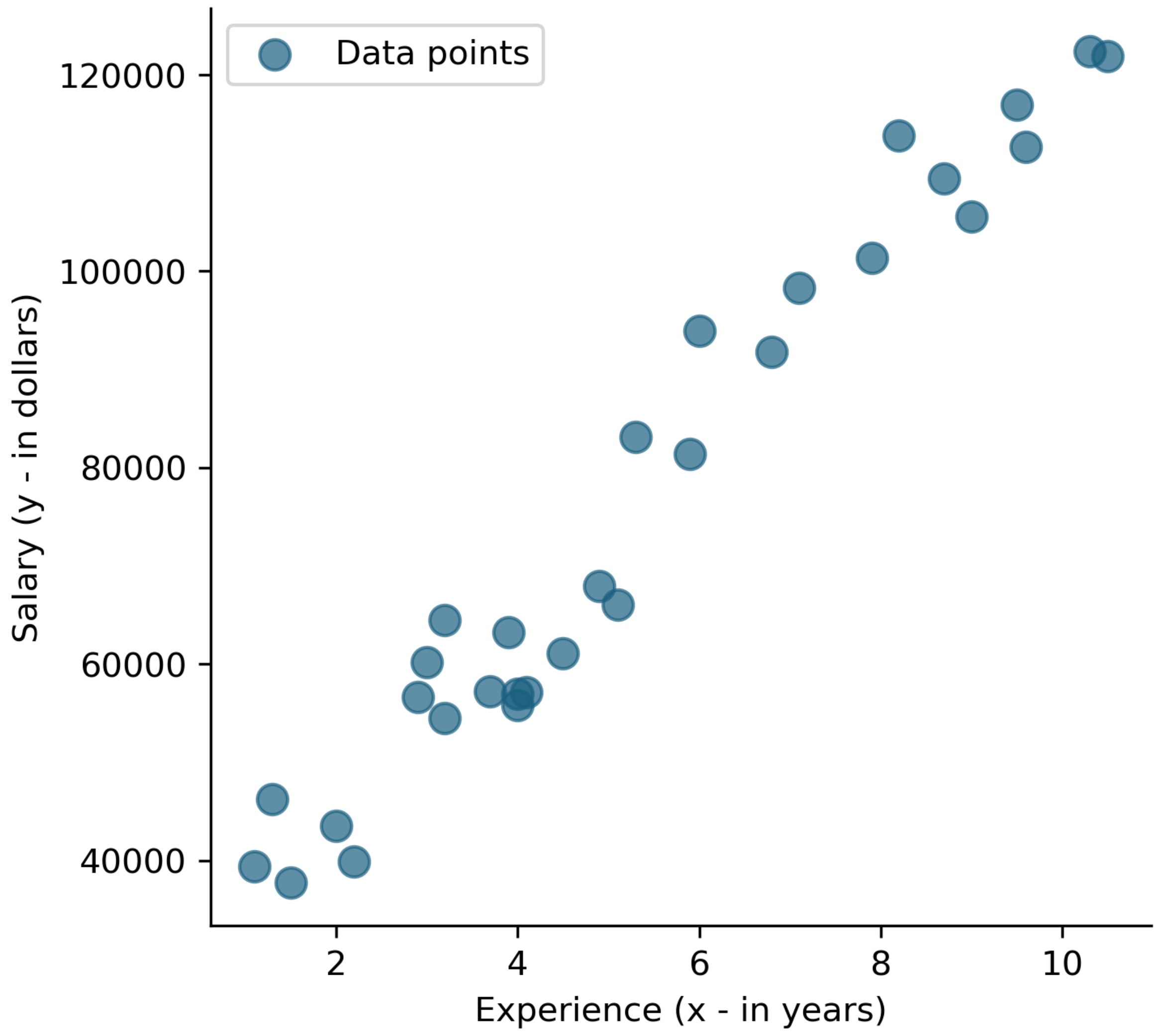
x - explanatory variable (input)

β - model parameters

β_0 - intercept

β_1 - slope

Review of linear models



$$\text{salary} \sim \text{experience}$$

$$\text{salary} = \beta_0 + \beta_1 \times \text{experience} + \epsilon$$

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

where:

y - response variable (output)

x - explanatory variable (input)

β - model parameters

β_0 - intercept

β_1 - slope

ϵ - random error

LINEAR MODEL - `ols()`

```
from statsmodels.formula.api import ols
```

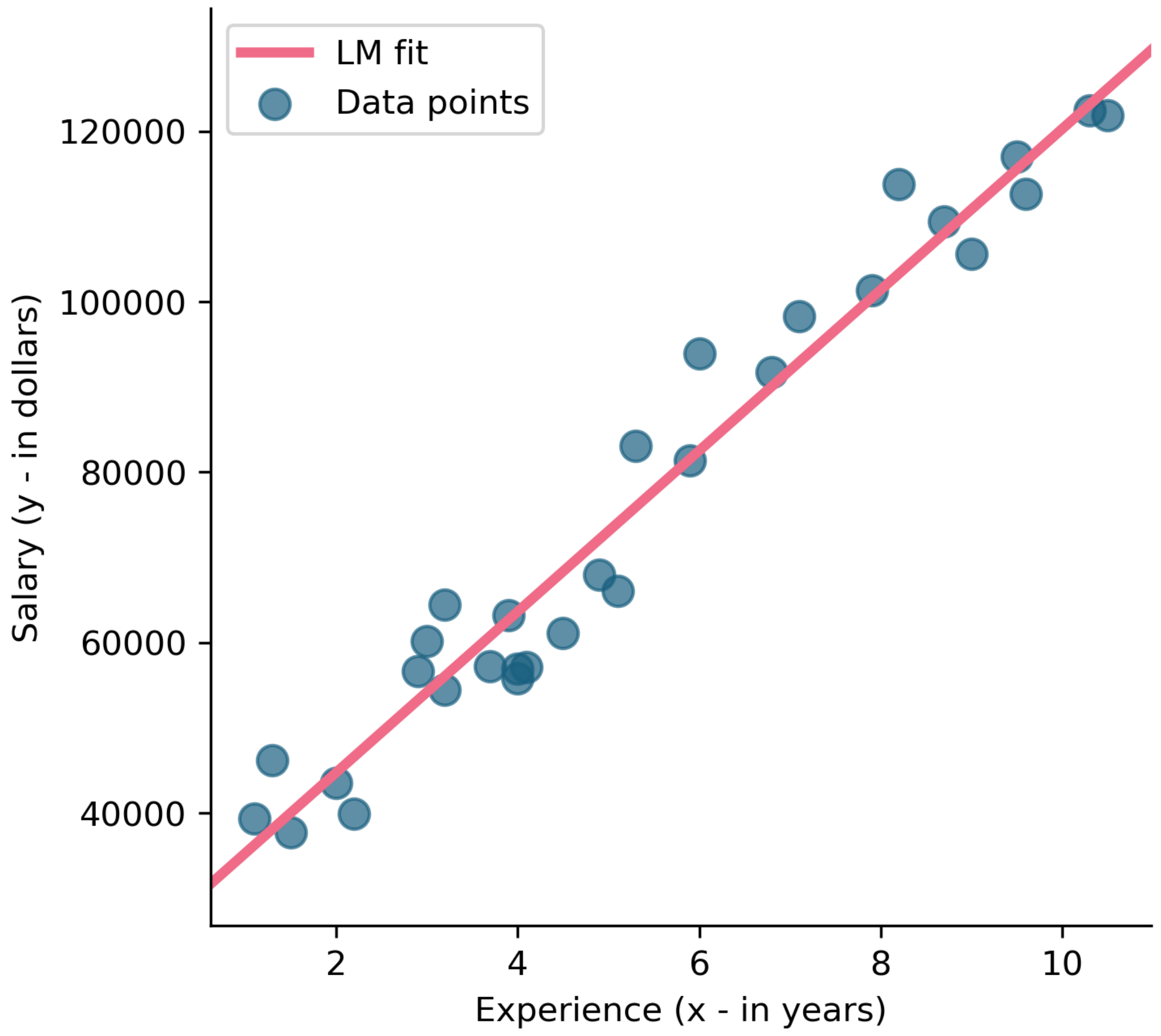
```
model = ols(formula = 'y ~ X',
            data = my_data).fit()
```

GENERALIZED LINEAR MODEL - `glm()`

```
import statsmodels.api as sm
from statsmodels.formula.api import glm
```

```
model = glm(formula = 'y ~ X',
            data = my_data,
            family = sm.families.____).fit
```

Assumptions of linear models



$$\text{salary} = 25790 + 9449 \times \text{experience}$$

Regression function

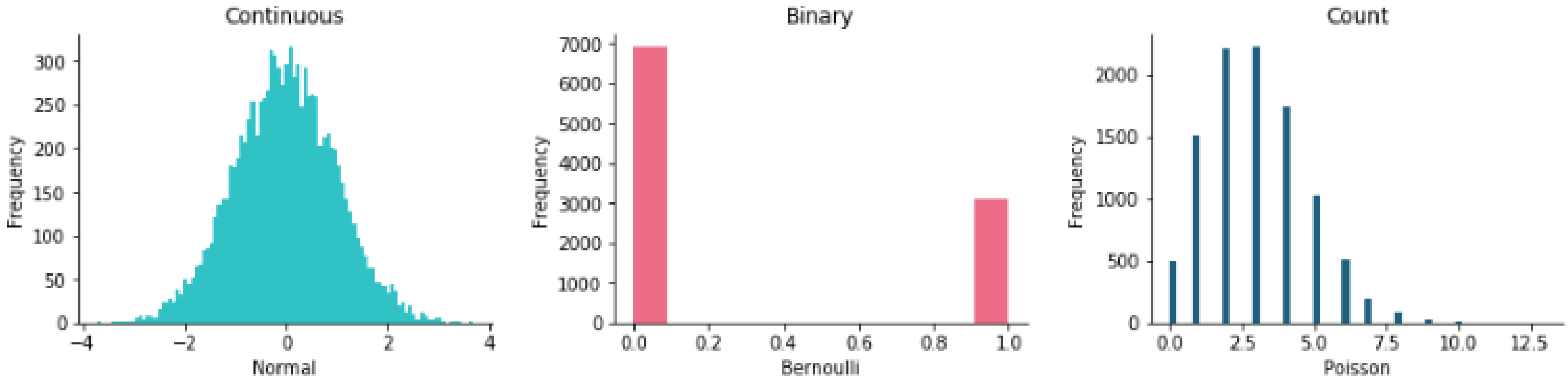
$$E[y] = \mu = \beta_0 + \beta_1 x_1$$

Assumptions

- Linear in parameters
- Errors are independent and normally distributed
- Constant variance

What if ... ?

- The response is binary or count $\rightarrow$ NOT continuous



- The variance of y is not constant $\rightarrow$ depends on the mean

Dataset - nesting of horseshoe crabs

Variable Name	Description
`sat`	Number of satellites residing in the nest
`y`	There is at least one satellite residing in the nest; 0/1
`weight`	Weight of the female crab in kg
`width`	Width of the female crab in cm
`color`	1 - light medium, 2 - medium, 3 - dark medium, 4 - dark
`spine`	1 - both good, 2 - one worn or broken, 3 - both worn or broken

[1] A. Agresti, An Introduction to Categorical Data Analysis, 2007.

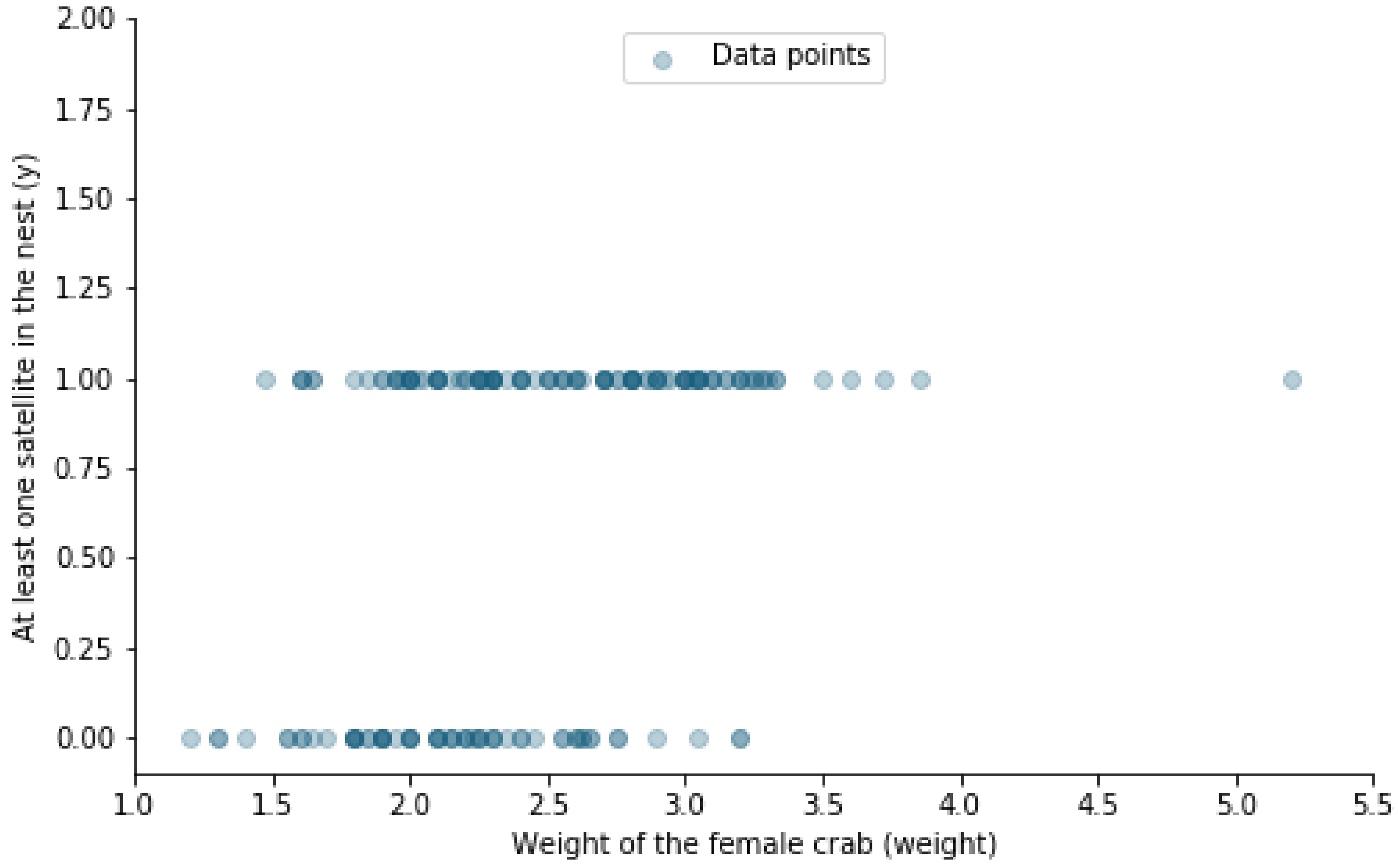
Linear model and binary response

$$\text{satellite crab} \sim \text{female crab weight}$$

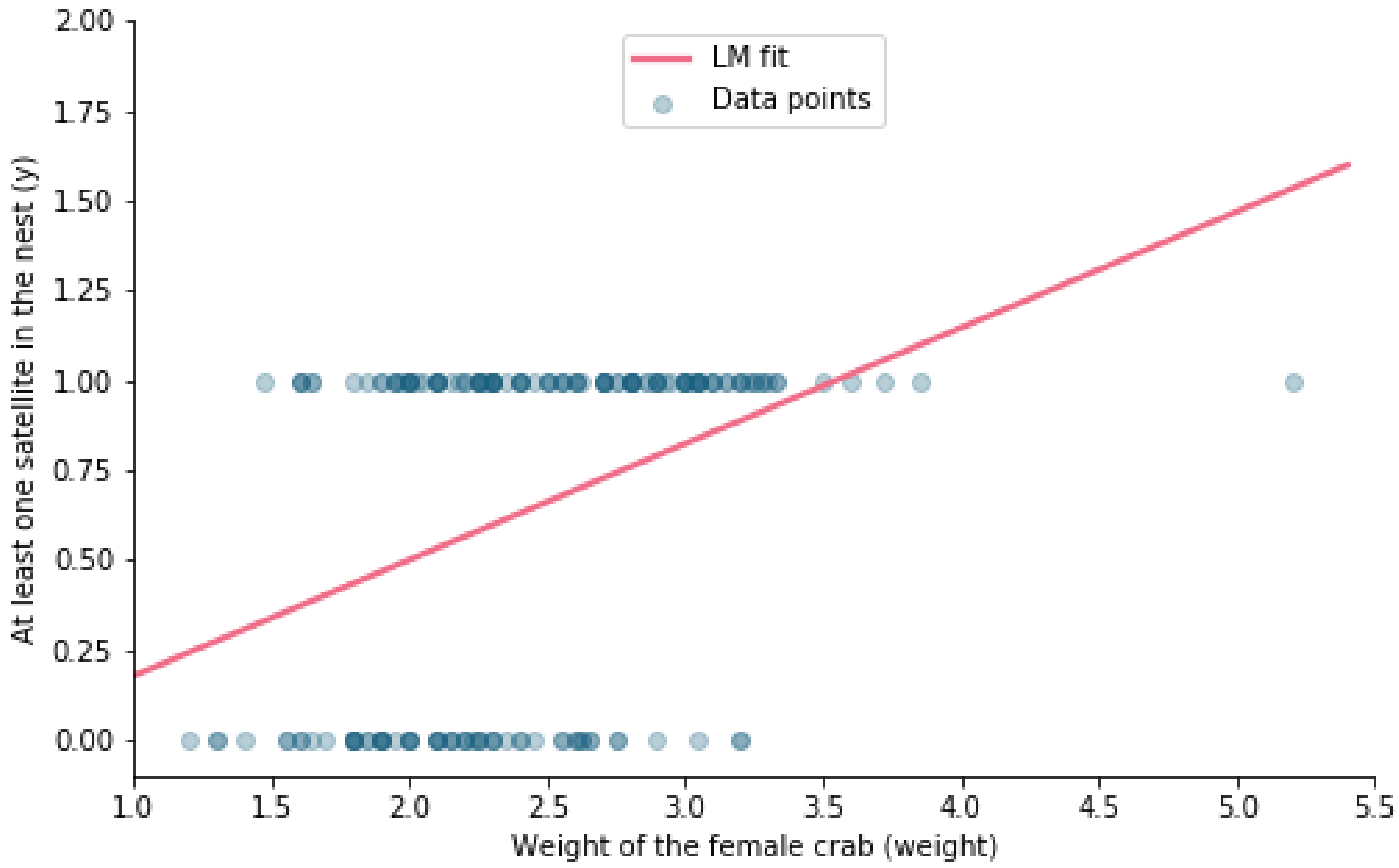
`y ~ weight`

$$P(\text{satellite crab is present}) = P(y = 1)$$

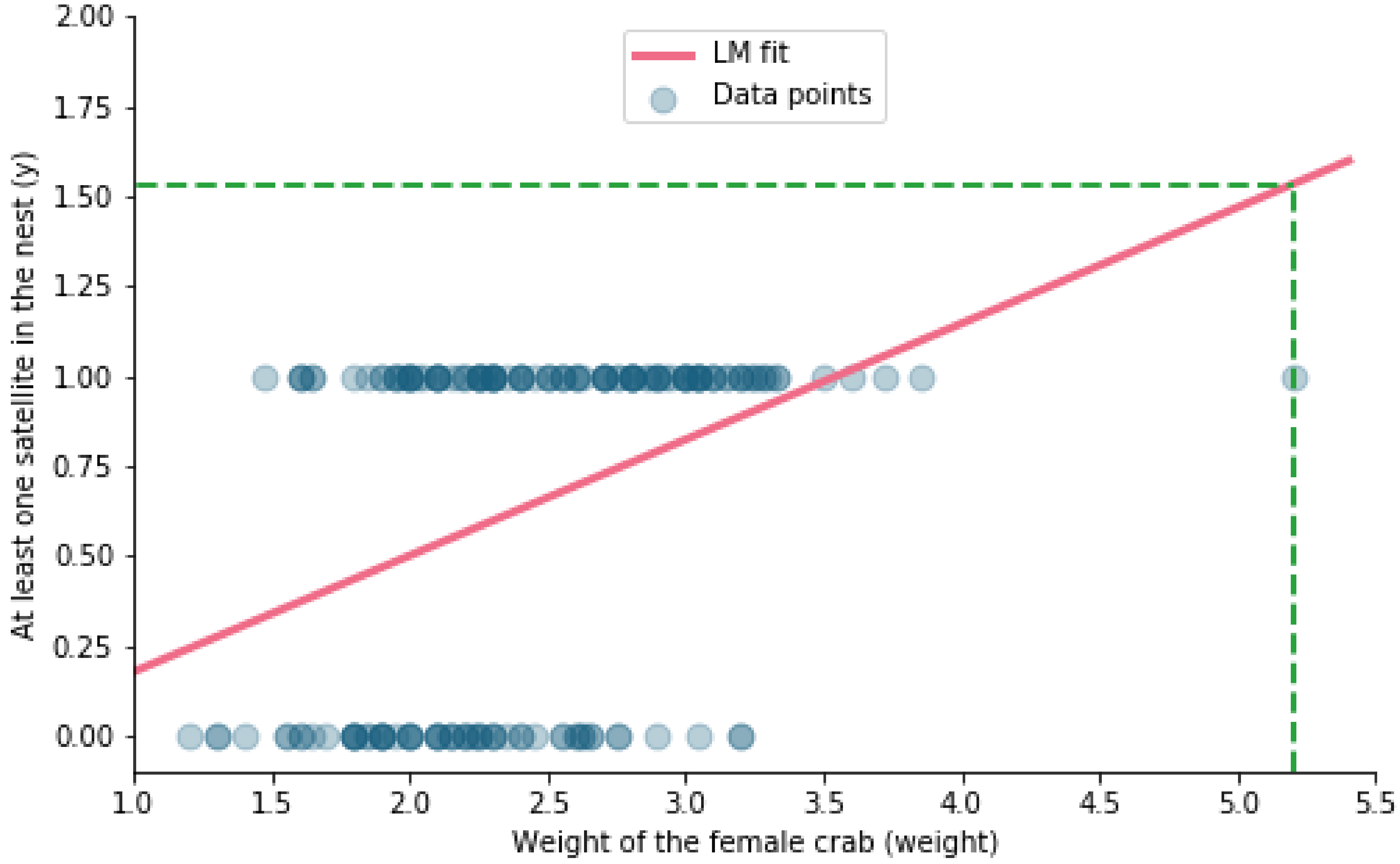
Linear model and binary response



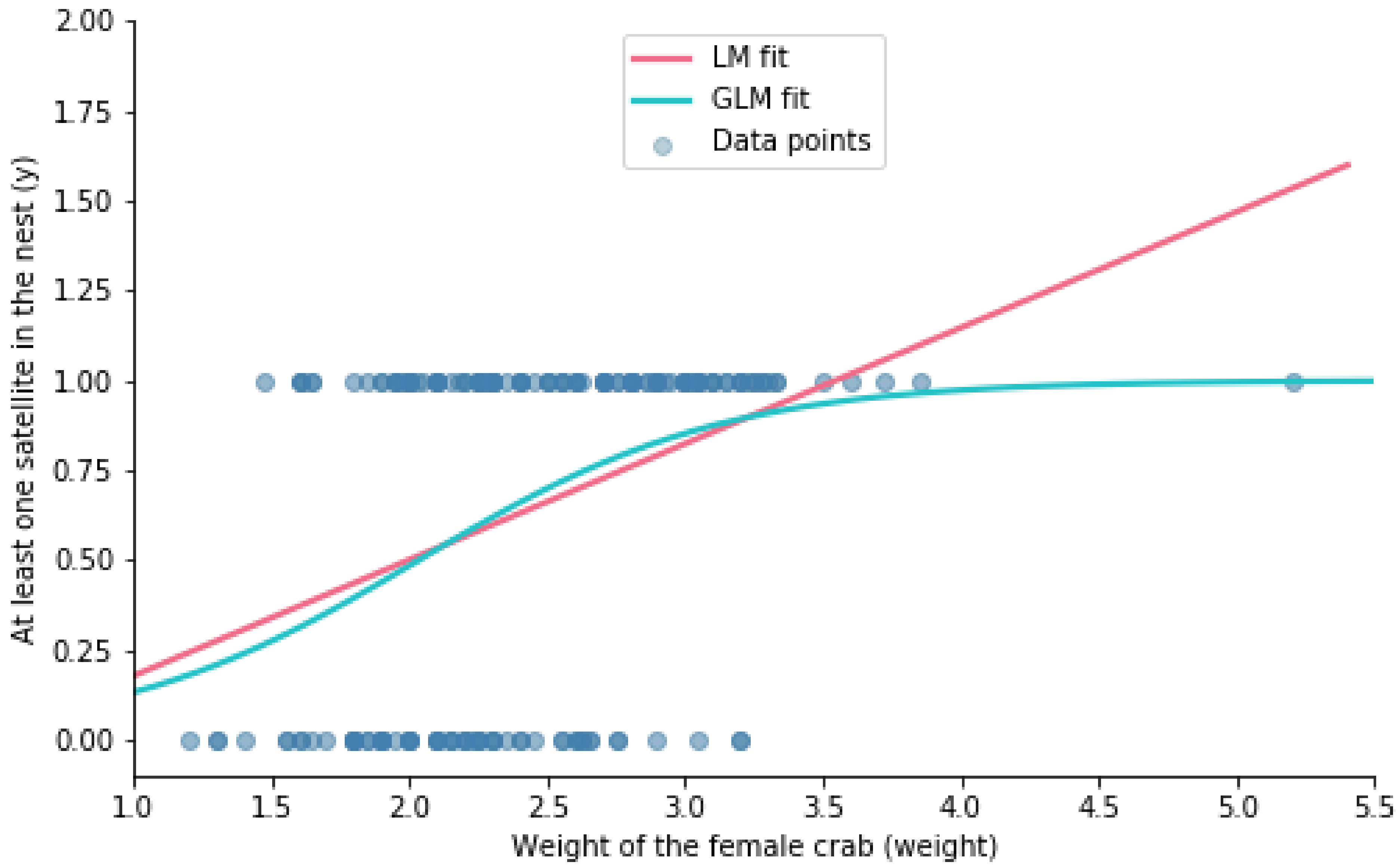
Linear model and binary response



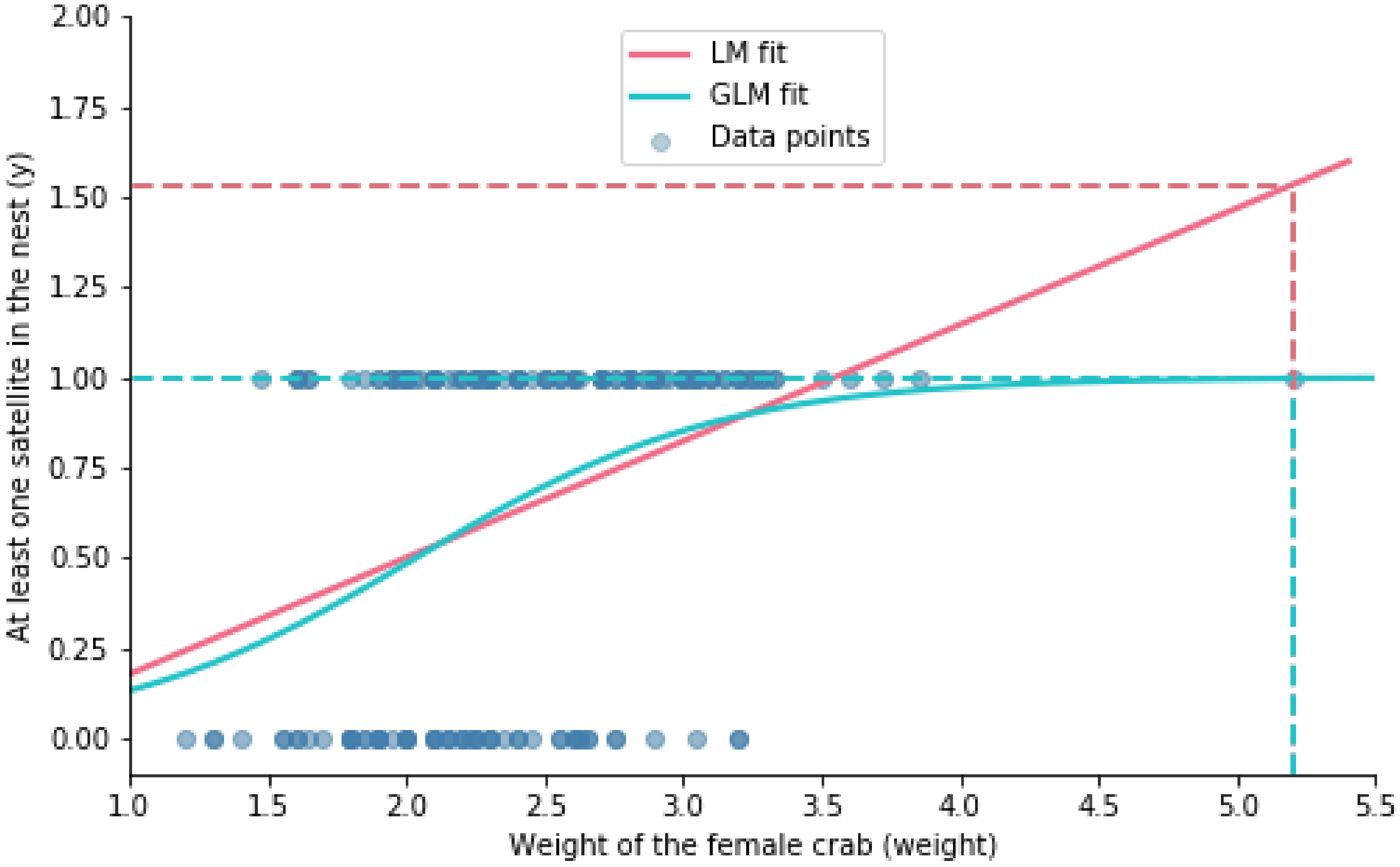
Linear model and binary response



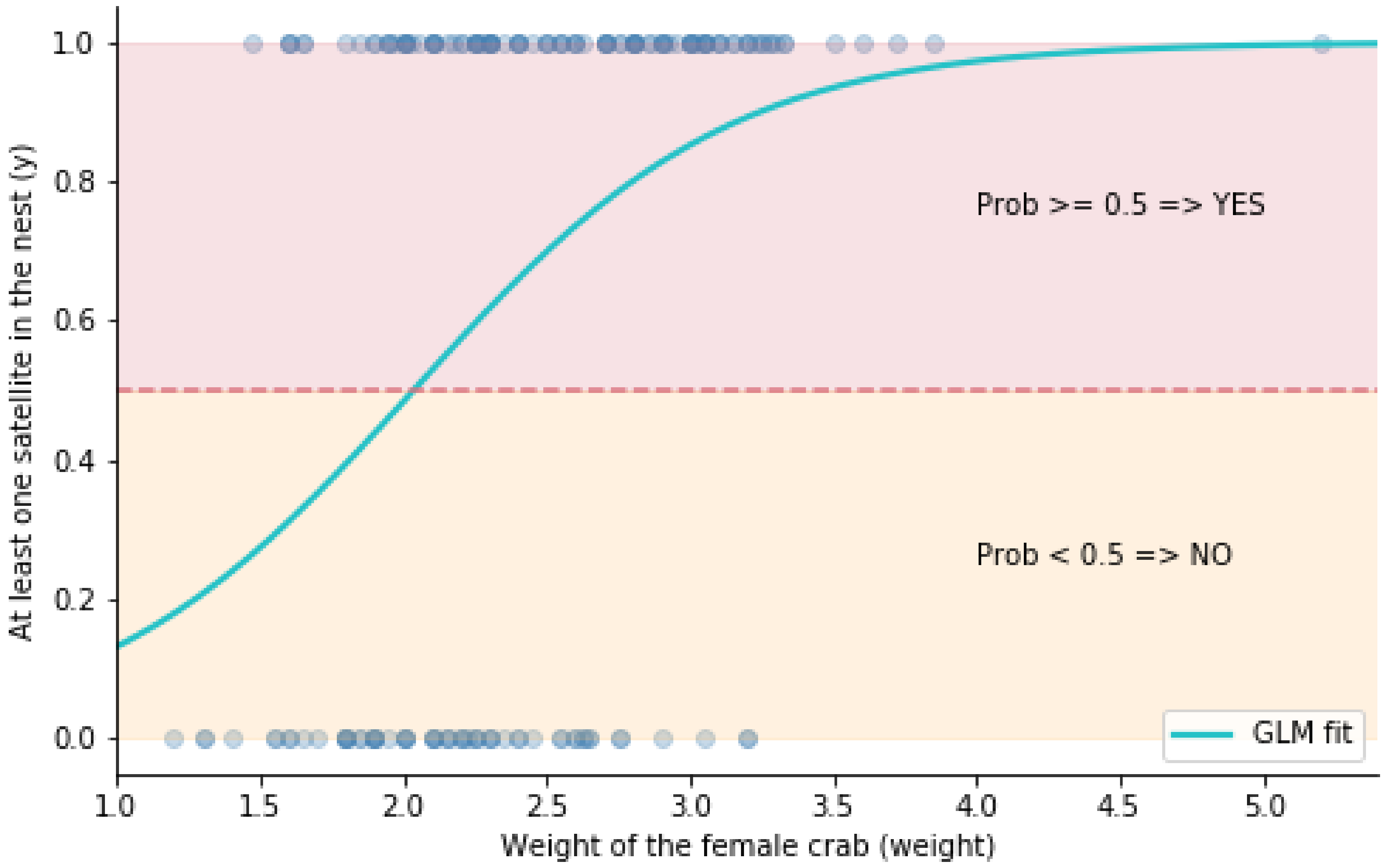
Linear model and binary data



Linear model and binary data



From probabilities to classes



Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

How to build a GLM?

GENERALIZED LINEAR MODELS IN PYTHON

Ita Cirovic Donev
Data Science Consultant

Components of the GLM

Random Component

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Components of the GLM

Random Component

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Systematic Component

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Components of the GLM

Random Component

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Systematic Component

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Interaction

$$g(E[y]) = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \beta_3 x_1 * x_2$$

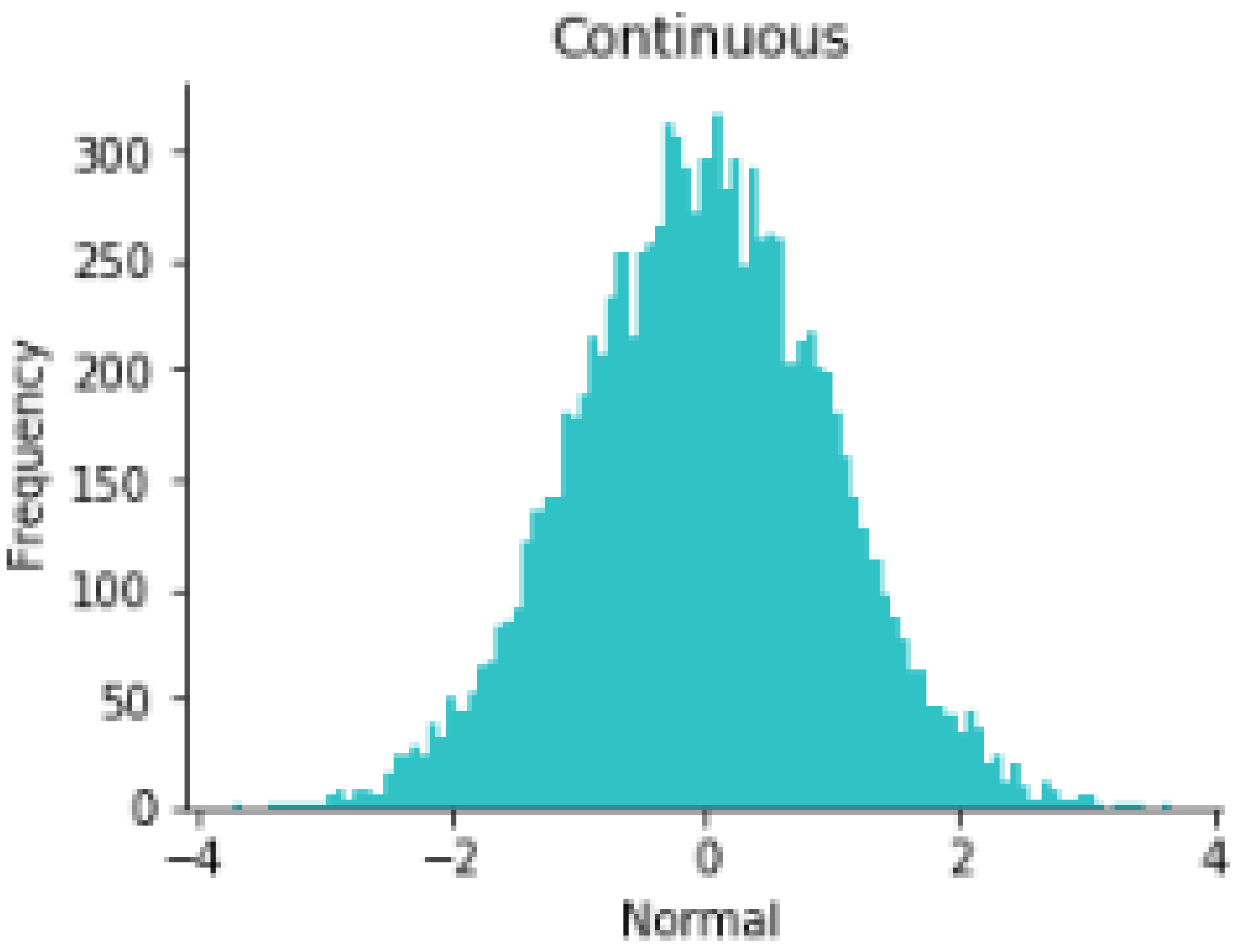
Components of the GLM

Random Component	$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$
Systematic Component	$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$
Interaction	$g(E[y]) = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \beta_3 x_1 * x_2$
Curvilinear	$g(E[y]) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$

Components of the GLM

Random Component	$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$
Systematic Component	$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$
Link Function	$g(E[y]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$

Continuous $\rightarrow$ Linear Regression



Data type: continuous

Domain: $(-\infty, \infty)$

Examples: house price, salary, person's height

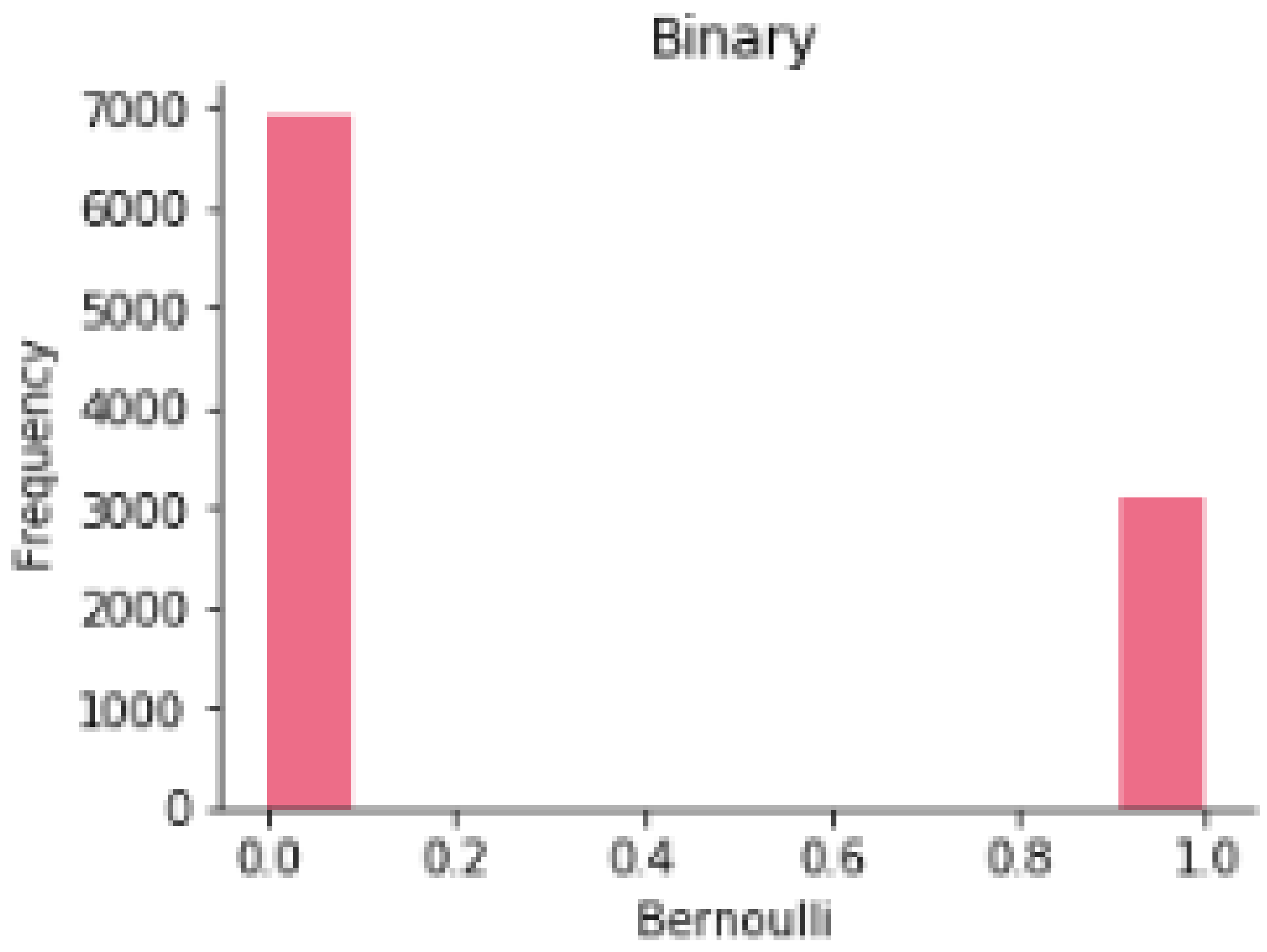
Family: `Gaussian()`

Link: identity

$$g(\mu) = \mu = E(y)$$

Model = Linear regression

Binary $\rightarrow$ Logistic regression



Data type: binary

Domain: $0, 1$

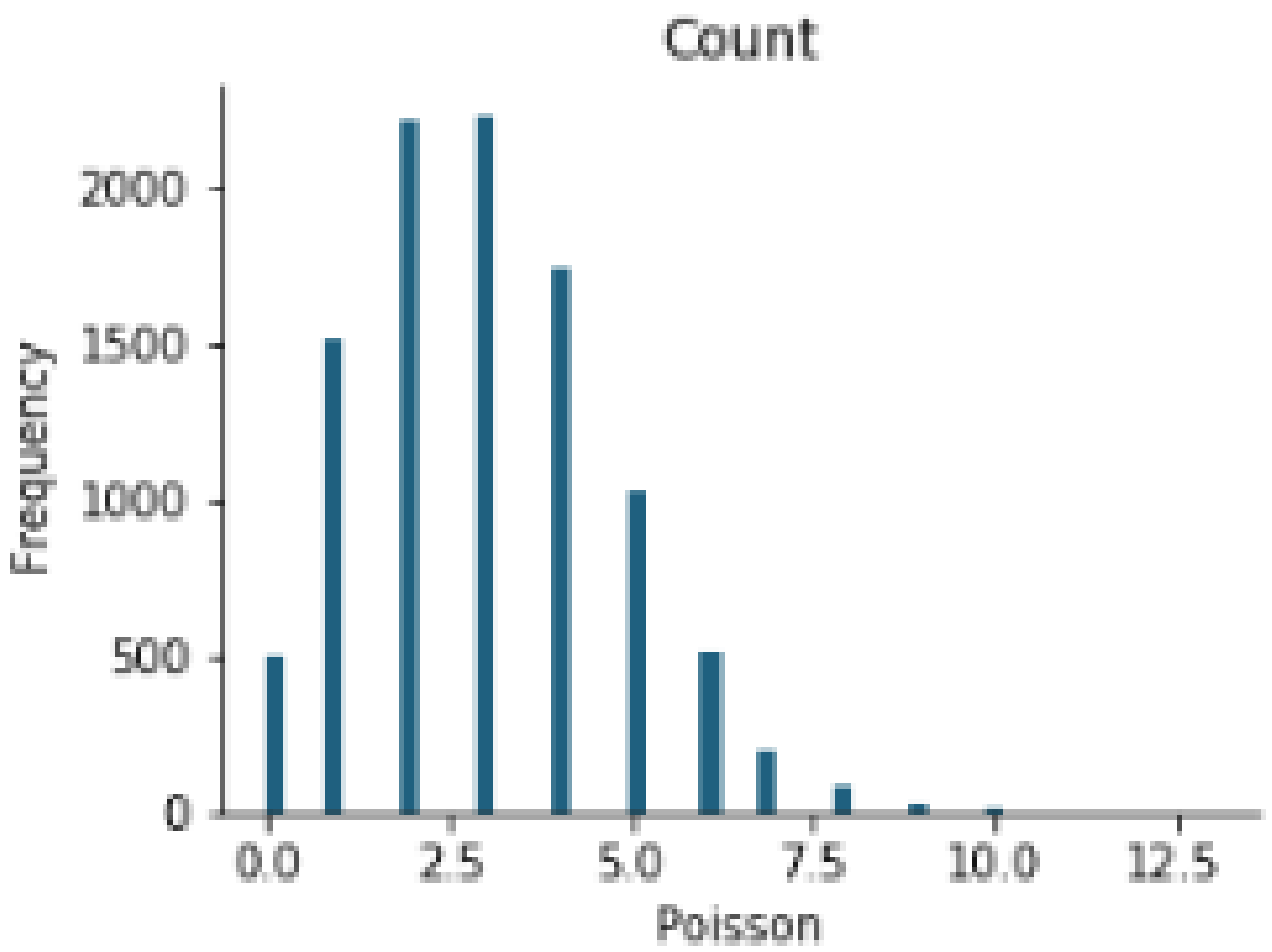
Examples: True/False

Family: `Binomial()`

Link: logit

Model = Logistic regression

Count $\rightarrow$ Poisson regression



Data type: count

Domain: $0, 1, 2, ..., \infty$

Examples: number of votes, number of hurricanes

Family: `Poisson()`

Link: logarithm

Model = Poisson regression

Link functions

Density	**Link:** $\eta = g(\mu)$	**Default link**	**`glm(family=...)`**
Normal	$\eta = \mu$	identity	`Gaussian()`
Poisson	$\eta = log(\mu)$	logarithm	`Poisson()`
Binomial	$\eta = log[p/(1-p)]$	logit	`Binomial()`
Gamma	$\eta = 1/\mu$	inverse	`Gamma()`
Inverse Gaussian	$\eta = 1/\mu^2$	inverse squared	`InverseGaussian()`

Benefits of GLMs

- A unified framework for many different data distributions
  - Exponential family of distributions
- Link function
  - Transforms the expected value of y
  - Enables linear combinations
  - Many techniques from linear models apply to GLMs as well

Let's practice

GENERALIZED LINEAR MODELS IN PYTHON

How to fit a GLM in Python?

GENERALIZED LINEAR MODELS IN PYTHON

Ita Cirovic Donev
Data Science Consultant

statsmodels

- Importing statsmodels

```
import statsmodels.api as sm
```

- Support for formulas

```
import statsmodels.formula.api as smf
```

- Use `glm()` directly

```
from statsmodels.formula.api import glm
```

Process of model fit

1. Describe the model $\rightarrow$ `glm()`
2. Fit the model $\rightarrow$ `.fit()`
3. Summarize the model $\rightarrow$ `.summary()`
4. Make model predictions $\rightarrow$ `.predict()`

Describing the model

FORMULA based

```
from statsmodels.formula.api import glm
```

```
model = glm(formula, data, family)
```

ARRAY based

```
import statsmodels.api as sm
```

```
X = sm.add_constant(X)
model = sm.glm(y, X, family)
```

Formula Argument

$$\texttt{response} \sim \texttt{explanatory variable(s)}$$

$$\texttt{output} \sim \texttt{input(s)}$$

```
formula = 'y ~ x1 + x2'
```

- `C(x1)` : treat `x1` as categorical variable
- `-1` : remove intercept
- `x1:x2` : an interaction term between `x1` and `x2`
- `x1*x2` : an interaction term between `x1` and `x2` and the individual variables
- `np.log(x1)` : apply vectorized functions to model variables

Family Argument

```
family = sm.families.____()
```

The family functions:

- `Gaussian(link = sm.families.links.identity)` $\longrightarrow$ the default family
- `Binomial(link = sm.families.links.logit)`
  - `probit`, `cauchy`, `log`, and `cloglog`
- `Poisson(link = sm.families.links.log)`
  - `identity` and `sqrt`

Other distribution families you can review at statsmodels website.

Summarizing the model

```
print(model_GLM.summary())
```

```
                 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:                -97.226
Date:                Mon, 21 Jan 2019   Deviance:                       194.45
Time:                        11:30:01   Pearson chi2:                     165.
No. Iterations:                     4   Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -12.3508      2.629     -4.698      0.000     -17.503      -7.199
width          0.4972      0.102      4.887      0.000       0.298       0.697
==============================================================================
```

Regression coefficients

`.params` prints regression coefficients

```
model_GLM.params
```

```
Intercept   -12.350818
width         0.497231
dtype: float64
```

`.conf_int(alpha=0.05, cols=None)` prints confidence intervals

```
model_GLM.conf_int()
```

```
                   0          1
Intercept -17.503010 -7.198625
width       0.297833  0.696629
```

Predictions

- Specify all the model variables in test data
- `.predict(test_data)` computes predictions

```
model_GLM.predict(test_data)
```

```
0     0.029309
1     0.470299
2     0.834983
3     0.972363
4     0.987941
```

Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON