

# Multivariable logistic regression

GENERALIZED LINEAR MODELS IN PYTHON



**Ita Cirovic Donev**  
Data Science Consultant

# Multivariable setting

- Model formula

$$\text{logit}(y) = \beta_0 + \beta_1 x_1$$

# Multivariable setting

- Model formula

$$\text{logit}(y) = \beta_0 + \beta_1 x_1$$

# Multivariable setting

- Model formula

$$\text{logit}(y) = \beta_0 + \beta_1 x_1 + \beta_2 \textcolor{red}{x}_2 + \dots + \beta_p \textcolor{red}{x}_p$$

# Multivariable setting

- Model formula

$$\text{logit}(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

- In Python

```
model = glm('y ~ x1 + x2 + x3 + x4',  
            data = my_data,  
            family = sm.families.Binomial()).fit()
```

# Example - well switching

```
formula = 'switch ~ distance100 + arsenic'
```

```
wells_fit = glm(formula = formula, data = wells,  
                 family = sm.families.Binomial()).fit()
```

```
=====
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	0.0027	0.079	0.035	0.972	-0.153	0.158
distance100	-0.8966	0.104	-8.593	0.000	-1.101	-0.692
arsenic	0.4608	0.041	11.134	0.000	0.380	0.542

```
=====
```

# Example - well switching

	coef	std err	z	P> z	[0.025	0.975]
-----						
Intercept	0.0027	0.079	0.035	0.972	-0.153	0.158
distance100	-0.8966	0.104	-8.593	0.000	-1.101	-0.692
arsenic	0.4608	0.041	11.134	0.000	0.380	0.542

- Both coefficients are statistically significant
- Sign of coefficients logical
- A unit-change in `distance100` corresponds to a negative difference of 0.89 in the logit
- A unit-change in `arsenic` corresponds to a positive difference of 0.46 in the logit

# Impact of adding a variable

- Impact of `arsenic` variable
- `distance100` changes from -0.62 to -0.89
- Further away from the safe well
  - More likely to have higher arsenic levels

	coef	std err
Intercept	0.0027	0.079
distance100	-0.8966	0.104
arsenic	0.4608	0.041

	coef	std err
Intercept	0.6060	0.060
distance100	-0.6291	0.097



# Multicollinearity

- Variables that are **correlated** with other model variables



- Increase in standard errors of coefficients
  - Coefficients may not be statistically significant

<sup>1</sup> [https://en.wikipedia.org/wiki/Correlation\\_and\\_dependence](https://en.wikipedia.org/wiki/Correlation_and_dependence)

# Presence of multicollinearity?

What to look for?

- Coefficient is not significant, but variable is highly correlated with  $y$
- Adding/removing a variable significantly changes coefficients
- Not logical sign of the coefficient
- Variables have high pairwise correlation

# Variance inflation factor (VIF)

- Most widely used diagnostic for multicollinearity
  - Computed for each explanatory variable
  - How inflated the variance of the coefficient is
- Suggested threshold  $VIF > 2.5$
- In Python

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

# Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

# Comparing models

GENERALIZED LINEAR MODELS IN PYTHON



**Ita Cirovic Donev**  
Data Science Consultant

# Deviance

- Formula

$$D = -2LL(\beta)$$

- Measure of error
- Lower deviance  $\rightarrow$  better model fit
- Benchmark for comparison is the **null deviance**  $\rightarrow$  intercept-only model
- Evaluate
  - Adding a random noise variable would, on average, decrease deviance by 1
  - Adding  $p$  predictors to the model deviance should decrease by more than  $p$

# Deviance in Python

```
Generalized Linear Model Regression Results
=====
Dep. Variable:          switch    No. Observations:          3020
Model:                GLM        Df Residuals:              3018
Model Family:        Binomial    Df Model:                  1
Link Function:        logit      Scale:                    1.0000
Method:              IRLS        Log-Likelihood:           -2038.1
Date:                Mon, 08 Apr 2019    Deviance:                 4076.2
Time:                10:24:56          Pearson chi2:             3.02e+03
No. Iterations:      4              Covariance Type:          nonrobust
=====
               coef    std err          z      P>|z|      [0.025    0.975]
-----
Intercept      0.6060     0.060     10.047     0.000     0.488     0.724
distance100    -0.6219     0.097     -6.383     0.000    -0.813    -0.431
=====
```

# Compute deviance

- Extract null-deviance and deviance

```
# Extract null deviance  
print(model.null_deviance)
```

```
4118.0992
```

```
# Extract model deviance  
print(model.deviance)
```

```
4076.2378
```

- Compute deviance using log likelihood

```
print(-2*model.llf)
```

```
4076.2378
```

- Reduction in deviance by 41.86
- Including `distance100` improved the fit



# Model complexity

- `model_1` and `model_2`, where
  - $L1 > L2$
  - Number of parameters higher in `model_2`
- `model_2` is **overfitting**

# Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

# Model formula

GENERALIZED LINEAR MODELS IN PYTHON



**Ita Cirovic Donev**  
Data Science Consultant

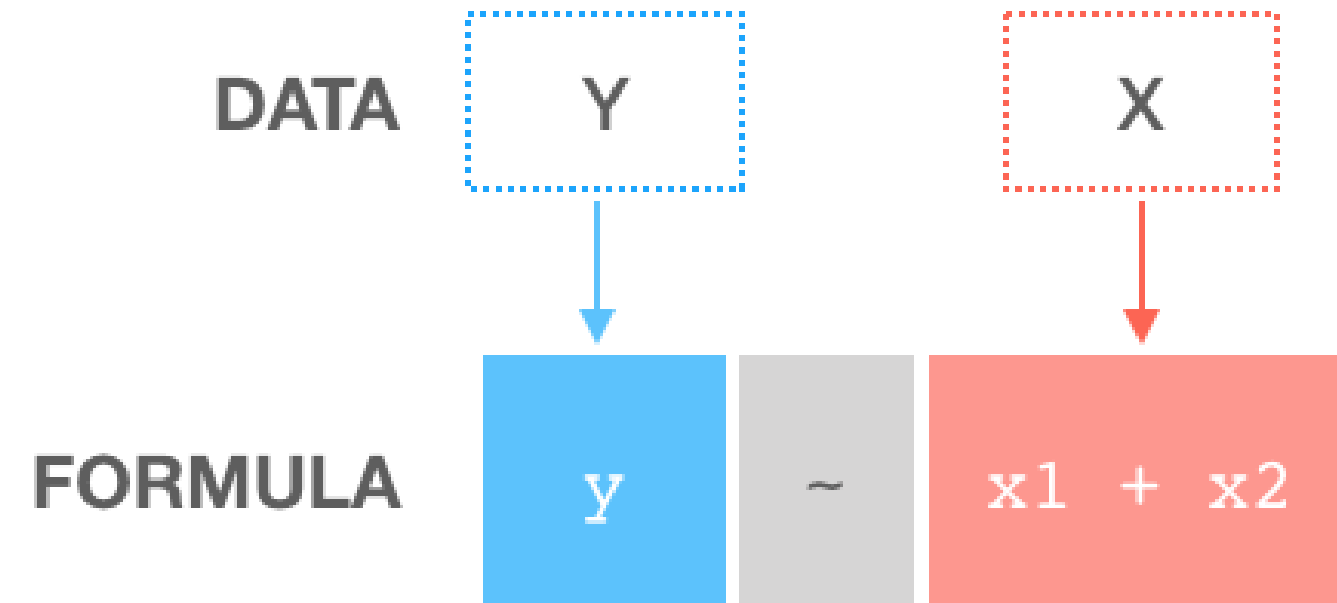
# Formula and model matrix

DATA

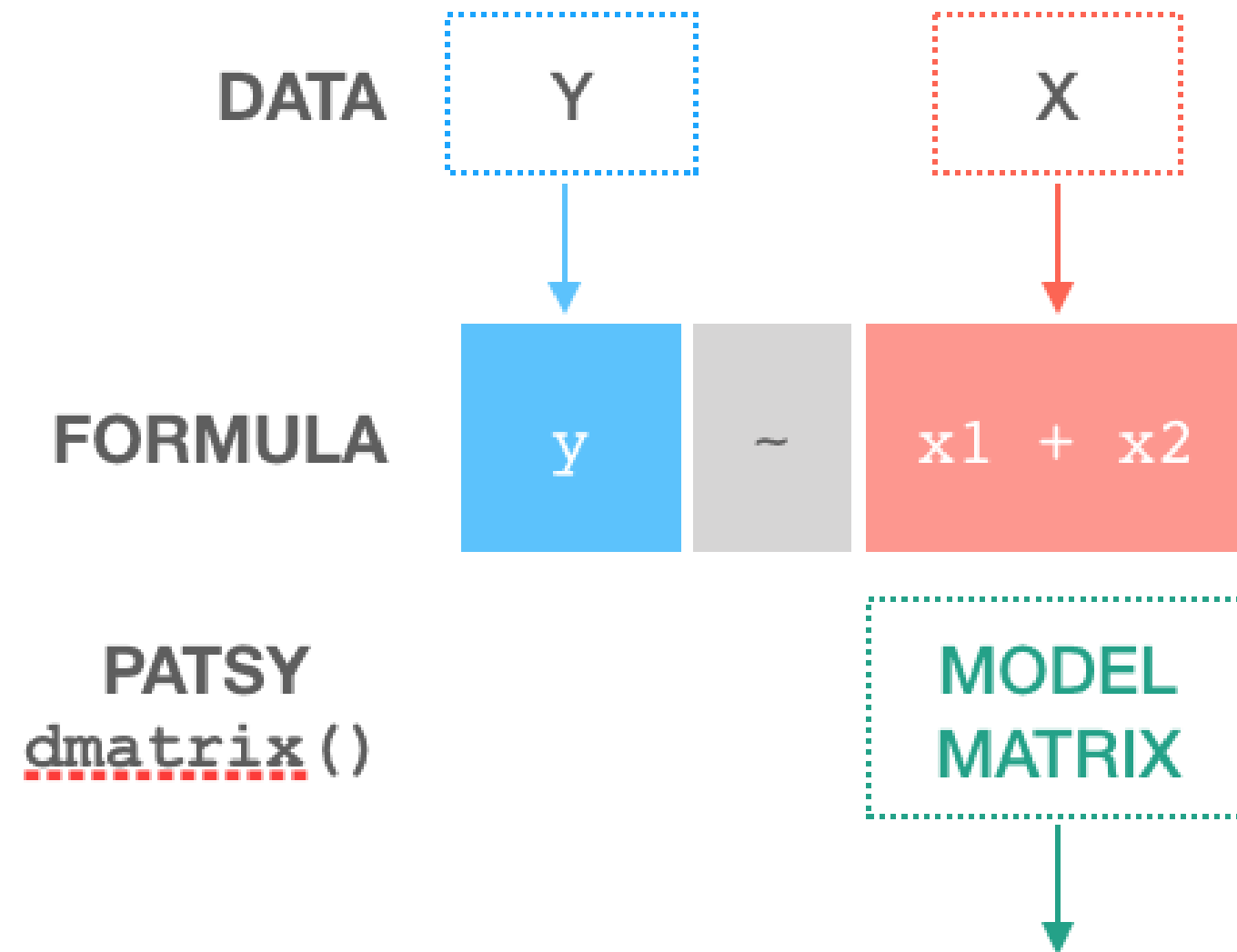
Y

X

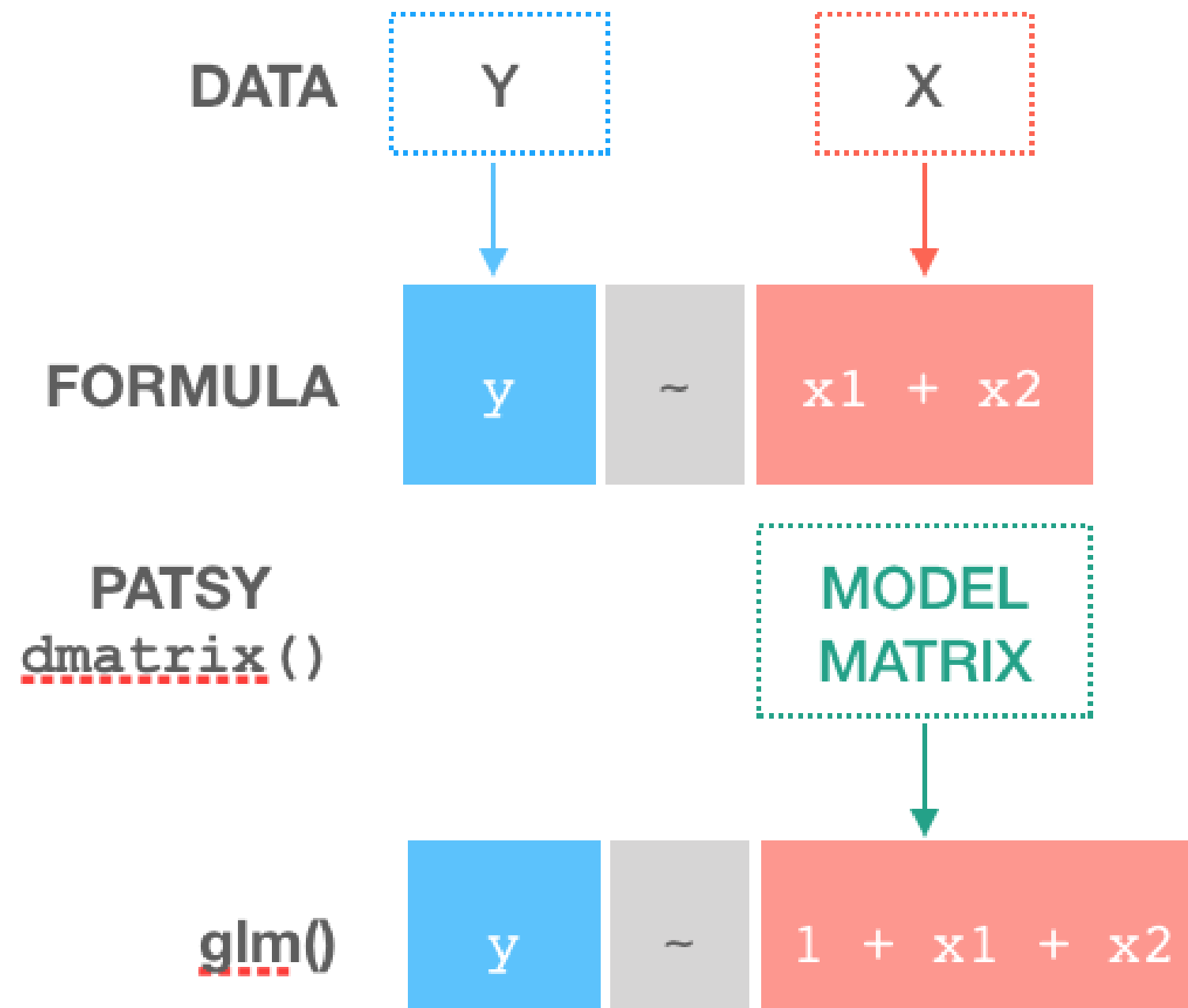
# Formula and model matrix



# Formula and model matrix



# Formula and model matrix



# Model matrix

- Model matrix:  $y \sim \mathbf{X}$
- Model formula

```
'y ~ x1 + x2'
```

- Check model matrix structure

```
from patsy import dmatrix  
dmatrix('x1 + x2')
```

Intercept	x1	x2
1	1	4
1	2	5
1	3	6



# Variable transformation

```
import numpy as np
```

```
'y ~ x1 + np.log(x2)'
```

```
dmatrix('x1 + np.log(x2)')
```

```
DesignMatrix with shape (3, 3)
```

Intercept	x1	np.log(x2)
1	1	1.38629
1	2	1.60944
1	3	1.79176

# Centering and standardization

- Stateful transforms

```
'y ~ center(x1) + standardize(x2)'
```

```
dmatrix('center(x1) + standardize(x2)')
```

```
DesignMatrix with shape (3, 3)
  Intercept  center(x1)  standardize(x2)
         1         -1        -1.22474
         1          0         0.00000
         1          1         1.22474
```

# Build your own transformation

```
def my_transformation(x):  
    return 4 * x
```

```
dmatrix('x1 + x2 + my_transformation(x2)')
```

DesignMatrix with shape (3, 4)

Intercept	x1	x2	my_transformation(x2)
1	1	4	16
1	2	5	20
1	3	6	24

# Arithmetic operations

```
x1 = np.array([1, 2, 3])  
x2 = np.array([4, 5, 6])
```

```
dmatrix('I(x1 + x2)')
```

DesignMatrix with shape (3, 2)

Intercept	I(x1 + x2)
1	5
1	7
1	9

```
x1 = [1, 2, 3]  
x2 = [4, 5, 6]
```

```
dmatrix('I(x1 + x2)')
```

DesignMatrix with shape (6, 2)

Intercept	I(x1 + x2)
1	1
1	2
1	3
1	4
1	5
1	6

# Coding the categorical data

Color

Red

Green

Blue

# Coding the categorical data

Color	Color
Red	Green
Green	Red
Blue	Red
	Blue
	...

# Coding the categorical data

Color	Color
Red	Green
Green	Red
Blue	Red
	Blue
	...

encoding

Red	Green	Blue
0	1	0
1	0	0
1	0	0
0	0	1
...	...	...

# Patsy coding

- Strings and booleans are automatically coded
- Numerical  $\rightarrow$  categorical
  - `C()` function
- Reference group
  - Default: first group
  - `Treatment`
  - `levels`



# The C() function

- Numeric variable

```
dmatrix('color', data = crab)
```

```
DesignMatrix with shape (173, 2)
  Intercept  color
           1     2
           1     3
           1     1
[... rows omitted]
```

- How many levels?

```
crab['color'].value_counts()
```

```
2    95
3    44
4    22
1    12
```

# The C() function

- Categorical variable

```
dmatrix('C(color)', data = crab)
```

```
DesignMatrix with shape (173, 4)
```

Intercept	C(color)[T.2]	C(color)[T.3]	C(color)[T.4]
1	1	0	0
1	0	1	0
1	0	0	0

```
[... rows omitted]
```

# Changing the reference group

```
dmatrix('C(color, Treatment(4))', data = crab)
```

```
DesignMatrix with shape (173, 4)
```

Intercept	C(color)[T.1]	C(color)[T.2]	C(color)[T.3]
1	0	1	0
1	0	0	1
1	1	0	0
[... rows omitted]			

# Changing the reference group

```
l = [1, 2, 3, 4]  
dmatrix('C(color, levels = 1)', data = crab)
```

```
DesignMatrix with shape (173, 4)  
Intercept  C(color)[T.2]  C(color)[T.3]  C(color)[T.4]  
          1             1           0           0  
          1             0           1           0  
          1             0           0           0  
[... rows omitted]
```

# Multiple intercepts

```
'y ~ C(color)-1'  
dmatrix('C(color)-1', data = crab)
```

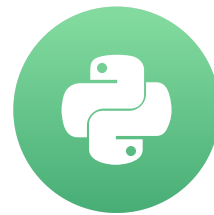
```
DesignMatrix with shape (173, 4)  
C(color)[1] C(color)[2] C(color)[3] C(color)[4]  
          0          1          0          0  
          0          0          1          0  
          1          0          0          0  
[... rows omitted]
```

# Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

# Categorical and interaction terms

GENERALIZED LINEAR MODELS IN PYTHON



**Ita Cirovic Donev**  
Data Science Consultant

# Categorical variables

- Simple binary variable
  - Yes, No
- Nominal variables
  - Color: red, green, blue
- Ordinal variables
  - Levels of education: Education1, Education2, ..., Education4



# Analysis of covariance

- Explanatory variables
  - $x_1$ : categorical (binary)
  - $x_2$ : continuous
- Logistic model

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

# Analysis of covariance

- Explanatory variables
  - $x_1$ : categorical (binary)
  - $x_2$ : continuous
- Logistic model

$$\text{logit}(y = 1 | \textcolor{red}{X}) = \beta_0 + \beta_1 \textcolor{red}{x}_1 + \beta_2 \textcolor{red}{x}_2$$

# Analysis of covariance

- Explanatory variables
  - $x_1$ : categorical (binary)
  - $x_2$ : continuous
- Logistic model

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

- If  $x_1 = 0$  then

$$\text{logit}(y = 1|x_1 = 0, x_2) = \beta_0 + 0 + \beta_2 x_2$$

# Analysis of covariance

- Explanatory variables
  - $x_1$ : categorical (binary)
  - $x_2$ : continuous

- Logistic model

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 \mathbf{x_1} + \beta_2 x_2$$

- If  $x_1 = 0$  then

$$\text{logit}(y = 1|x_1 = 0, x_2) = \beta_0 + 0 + \beta_2 x_2$$

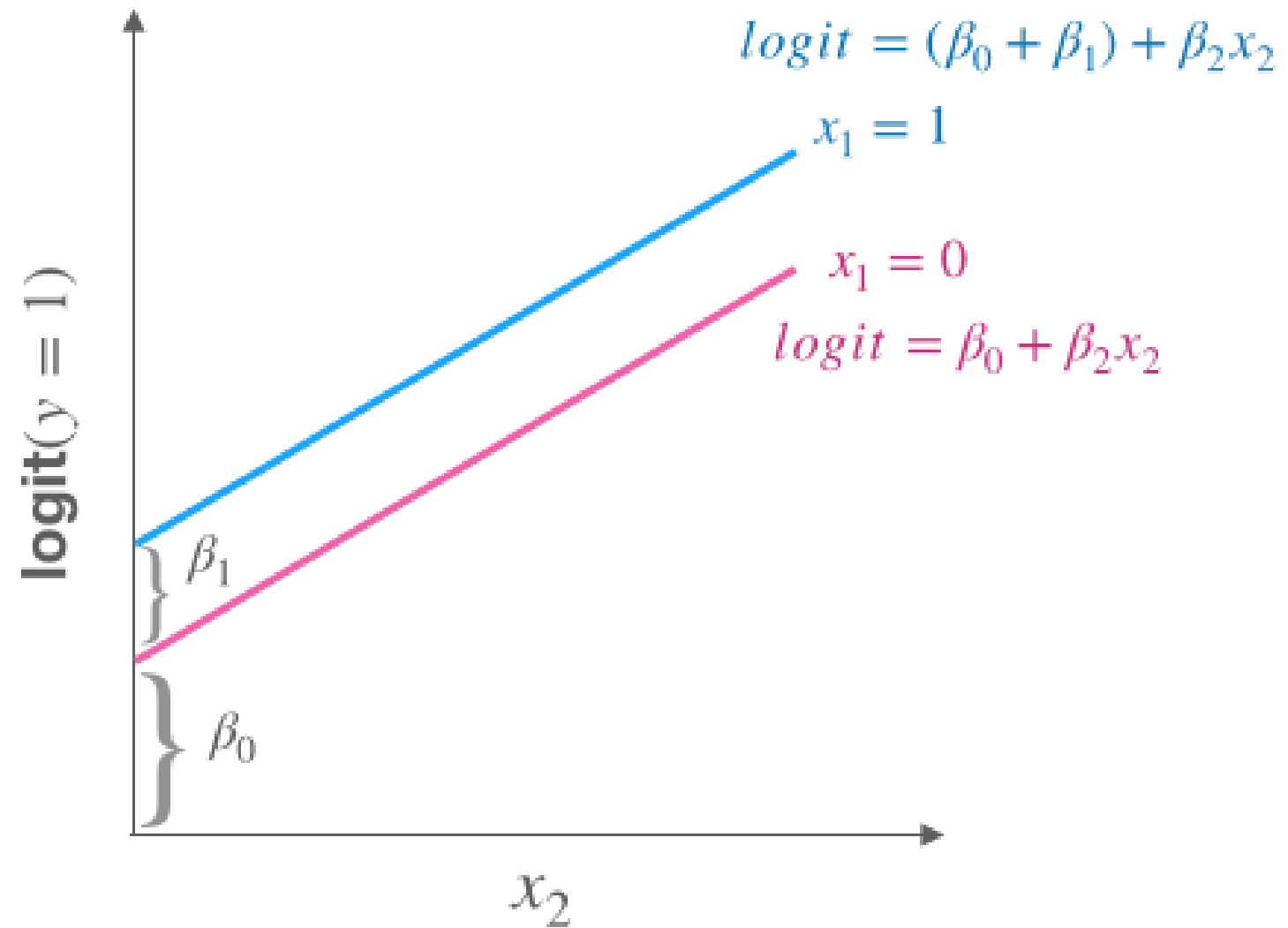
- If  $x_1 = 1$  then

$$\text{logit}(y = 1|\mathbf{x_1 = 1}, x_2) = \beta_0 + \mathbf{\beta_1} + \beta_2 x_2$$

$$\text{logit}(y = 1|\mathbf{x_1 = 1}, x_2) = (\beta_0 + \mathbf{\beta_1}) + \beta_2 x_2$$

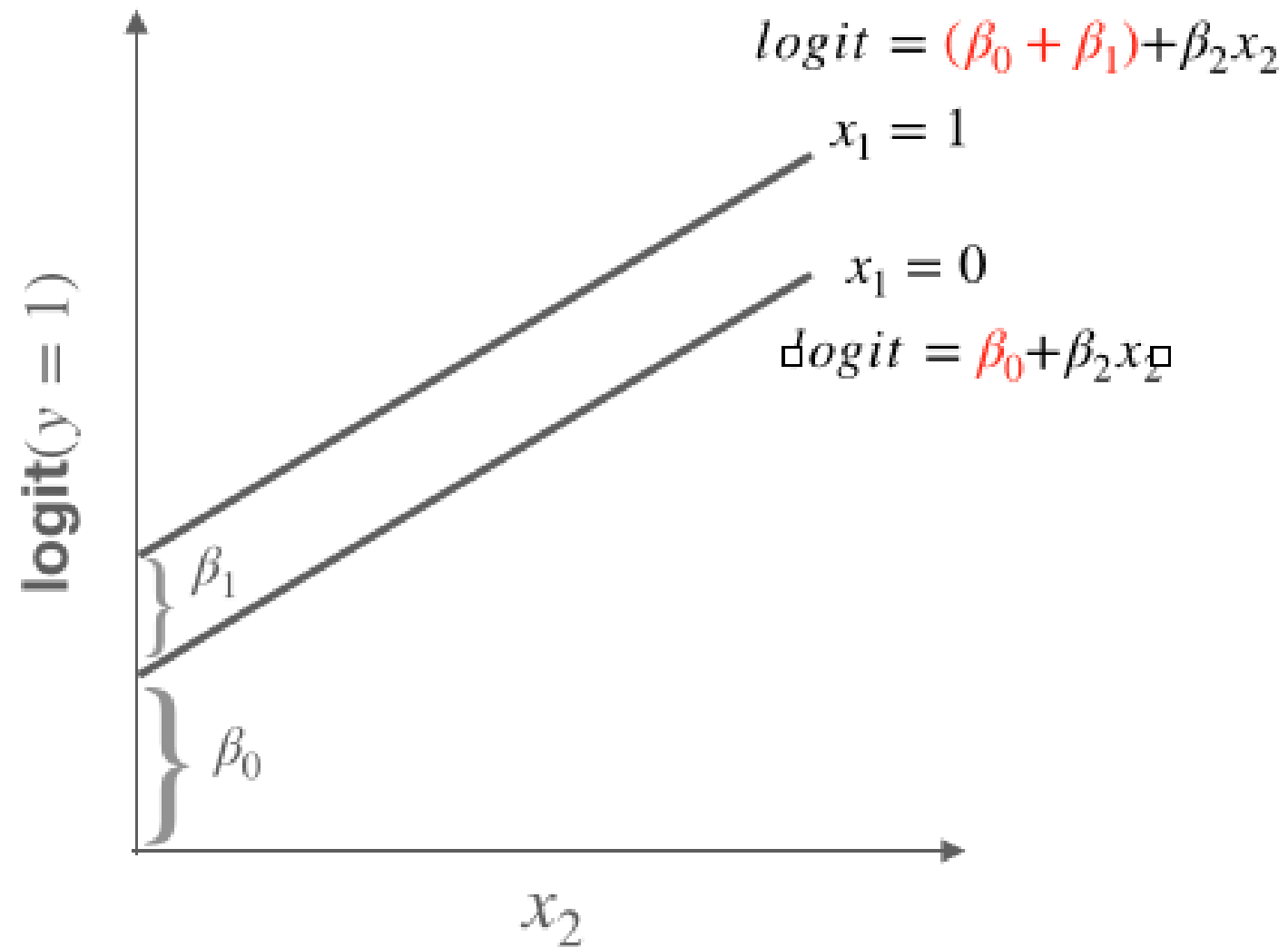
# Assumptions

No interaction



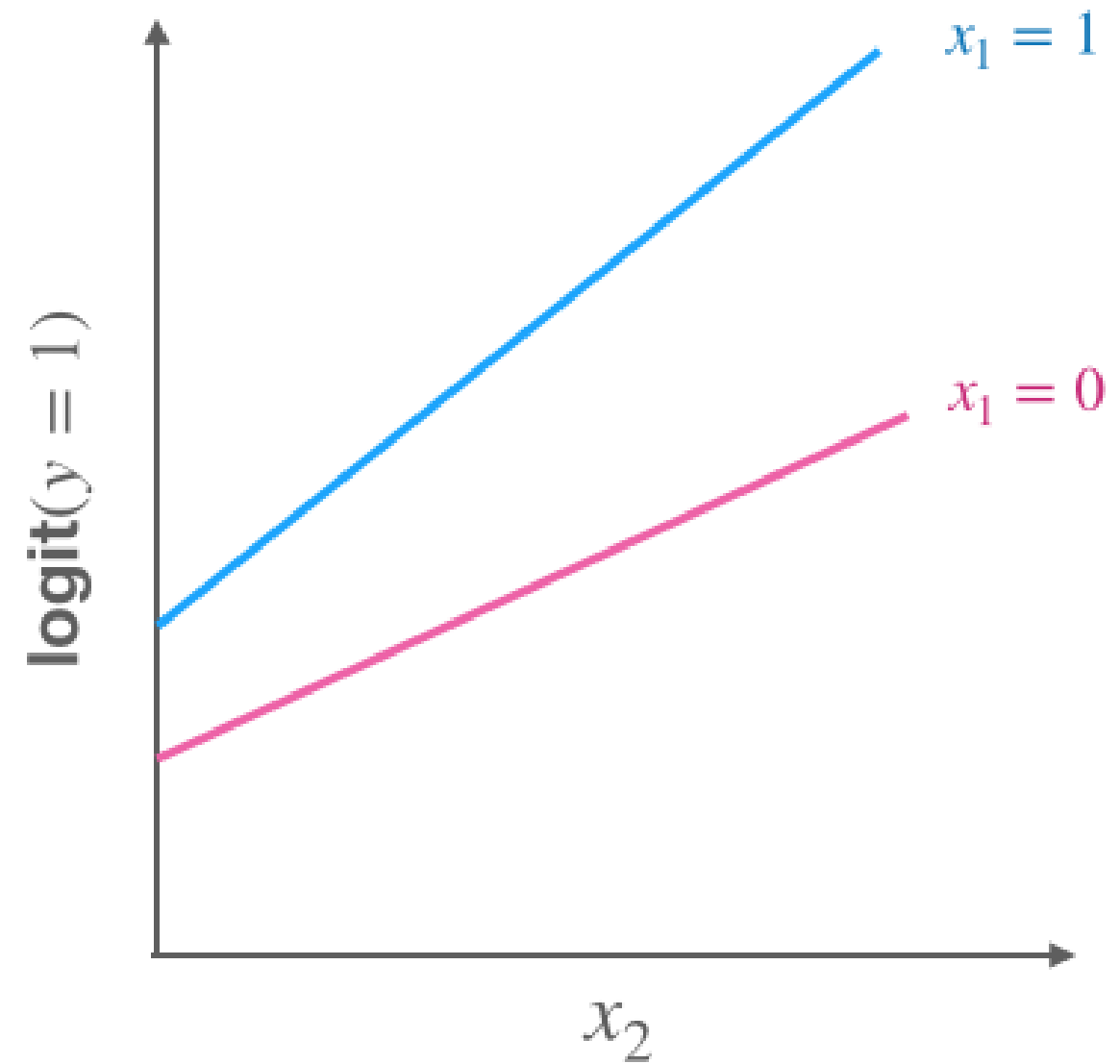
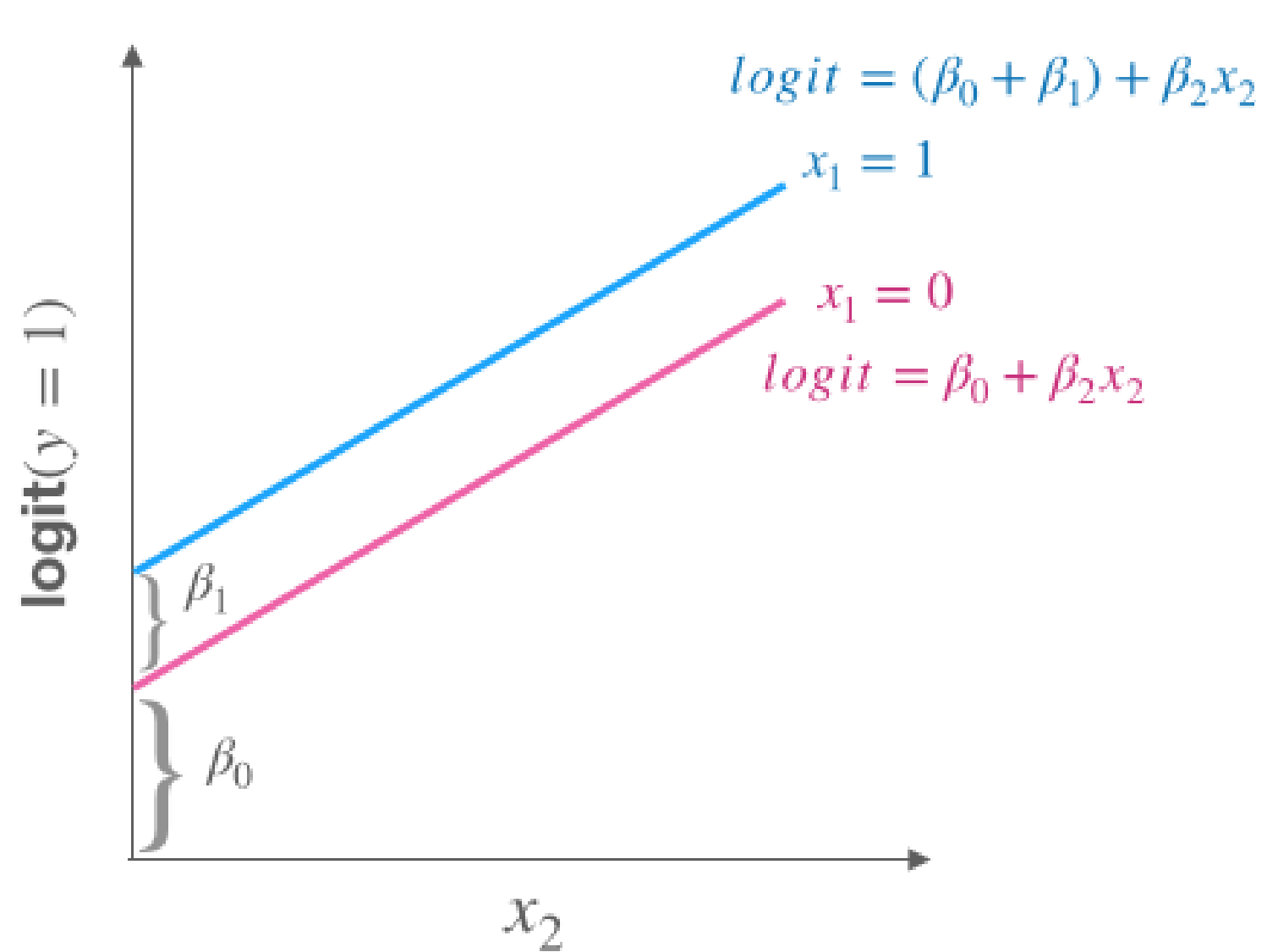
# Assumptions

No interaction



# Assumptions

No interaction



# Interactions

- Not equal slopes  $\rightarrow$  presence of **interaction**
- The effect of  $x_1$  on  $y$  depends on the level of  $x_2$  and vice versa
- Logistic model allowing for interactions

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$



# Interactions

- Not equal slopes  $\rightarrow$  presence of **interaction**
- The effect of  $x_1$  on  $y$  depends on the level of  $x_2$  and vice versa

- Logistic model allowing for interactions

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

- If  $x_1 = 0$  then

$$\text{logit}(y = 1|\textcolor{red}{x}_1 = \textcolor{red}{0}, x_2) = \beta_0 + \textcolor{red}{0} + \beta_2 x_2 + \textcolor{red}{0}$$

# Interactions

- Not equal slopes  $\rightarrow$  presence of **interaction**
- The effect of  $x_1$  on  $y$  depends on the level of  $x_2$  and vice versa

- Logistic model allowing for interactions

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

- If  $x_1 = 0$  then

$$\text{logit}(y = 1|x_1 = 0, x_2) = \beta_0 + \beta_2 x_2$$

- If  $x_1 = 1$  then

$$\text{logit}(y = 1|\mathbf{x_1 = 1}, x_2) = \beta_0 + \beta_1 + \beta_2 x_2 + \beta_3 x_2$$

$$\text{logit}(y = 1|\mathbf{x_1 = 1}, x_2) = (\beta_0 + \beta_1) + (\beta_2 + \beta_3)x_2$$

# Interactions

- Not equal slopes  $\rightarrow$  presence of **interaction**
- The effect of  $x_1$  on  $y$  depends on the level of  $x_2$  and vice versa

- Logistic model allowing for interactions

$$\text{logit}(y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

- If  $x_1 = 0$  then

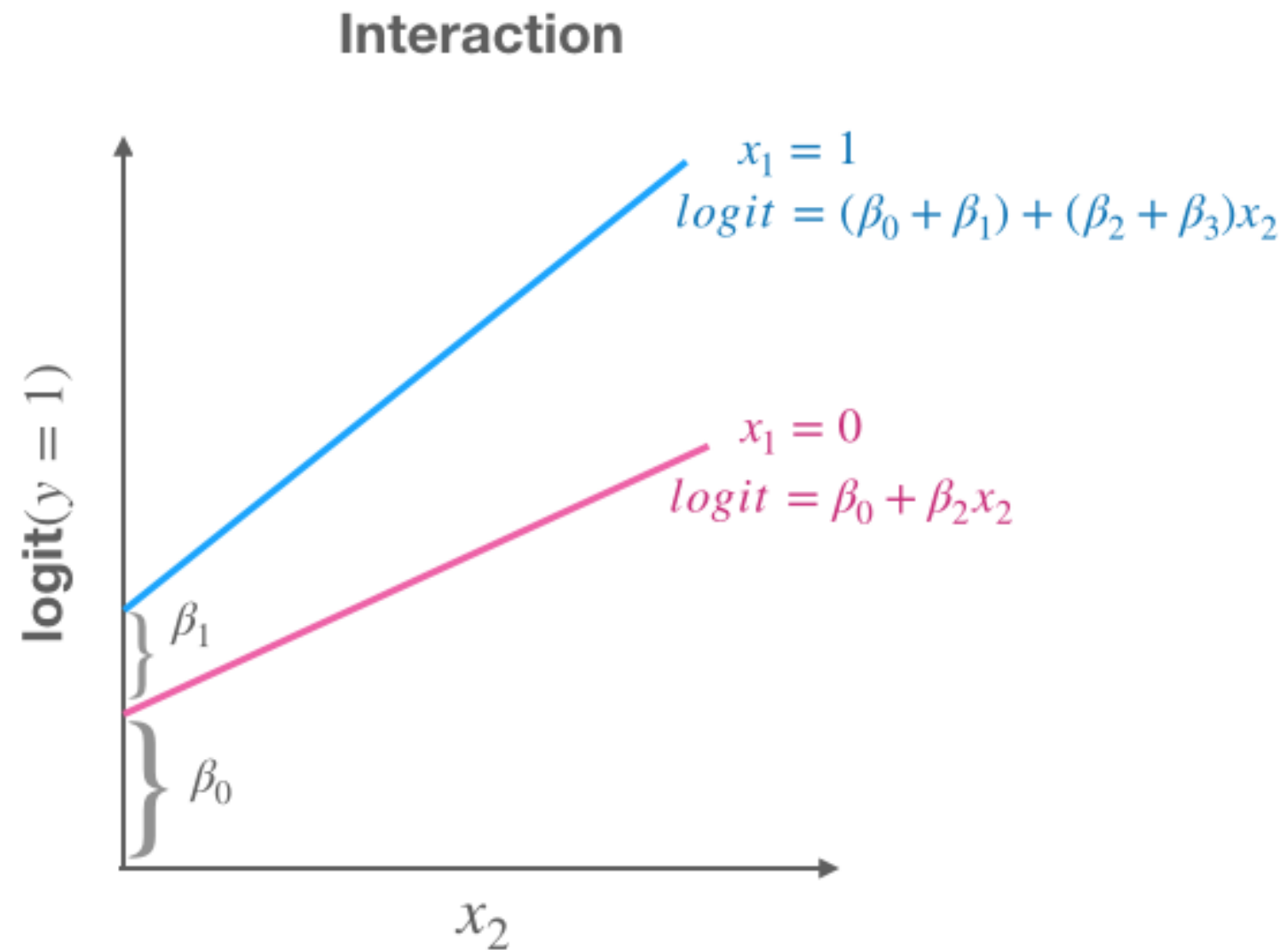
$$\text{logit}(y = 1|x_1 = 0, x_2) = \beta_0 + \beta_2 x_2$$

- If  $x_1 = 1$  then

$$\text{logit}(y = 1|x_1 = 1, x_2) = \beta_0 + \beta_1 + \beta_2 x_2 + \beta_3 x_2$$

$$\text{logit}(y = 1|x_1 = 1, x_2) = (\beta_0 + \beta_1) + (\beta_2 + \beta_3)x_2$$

# Visualizing interactions



Interactions allow for:

- intercept and slope different for  $x_1$
- $\beta_1$ : difference between the two intercepts
- $\beta_3$ : difference between the two slopes

# Interaction types

- binary  $\times$  binary
- binary  $\times$  categorical
- binary  $\times$  continuous
- continuous  $\times$  categorical
- continuous  $\times$  continuous
- categorical  $\times$  categorical
- more than 2 variable interactions

# Let's practice!

GENERALIZED LINEAR MODELS IN PYTHON

# Congratulations!

GENERALIZED LINEAR MODELS IN PYTHON



**Ita Cirovic Donev**  
Data Science Consultant

## MODEL

- Data  $\rightarrow$
- Link function  $\rightarrow$
- Model  $\rightarrow$
- 1-unit increase in  $x \rightarrow$

## LINEAR MODEL

- Continuous
- Identity
- $y = \beta_0 + \beta_1 x_1$
- increases  **$y$**  by  $\beta_1$

## LOGISTIC REGRESSION

- Binary
- Logit
- $\text{logit}(y) = \beta_0 + \beta_1 x_1$
- increases **log odds** by  $\beta_1$

## POISSON REGRESSION

- Count
- Logarithm
- $\log(\lambda) = \beta_0 + \beta_1 x_1$
- **multiplies  $\lambda$**  by  $\exp(\beta_1)$



## MAIN PYTHON FUNCTIONS

- Fit the model  
`statmodels` →

## LOGISTIC REGRESSION

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

## LINEAR MODEL

```
glm('y ~ x', data)
```

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

## POISSON REGRESSION

```
glm('y ~ x', data,  
    family = sm.families.Poisson())
```

# Next steps...

- DataCamp courses
- Excellent reference books
  - *Regression Modeling Strategies* by Frank E. Harrell, Jr.
  - *An Introduction to Categorical Data Analysis* by Alan Agresti
  - *Applied Predictive Modeling* by Max Kuhn and Kjell Johnson

# Happy modeling!

GENERALIZED LINEAR MODELS IN PYTHON