Logistic Regression

Cesar Acosta Ph.D.

Logistic Regression

Logistic Regression models
are used with classification problems
when the response has ***two categories***

These are called binary classification problems

Logistic Regression

Preparation

Logistic regression - Preparation

- *Odds of random event*
- *Indicator random variable*
- *Bernoulli random variable*
- *Logistic distribution function*

Logistic regression - Preparation

- *Odds of random event*
- *Indicator random variable*
- *Bernoulli random variable*
- *Logistic distribution function*

discrete random variables (Indicator, Bernoulli)

continuous random variable (Logistic)

Odds of a random event

A random event 'A' may be observed with probability π

Odds of a random event

A random event 'A' may be observed with probability π

The odds of event A

$$\text{Odds}\,[A] = \frac{\pi}{1-\pi}$$

Odds of a random event

A random event 'A' may be observed with probability π

The odds of event A

$$\text{Odds }[A] = \frac{\pi}{1-\pi}$$

how much likely is that A occurs

than it is that A does not occur

Odds of a random event - Example

Assume that 2/3 of voters are in favor of candidate A and 1/3 in favor of candidate B

Odds of a random event - Example

Assume that 2/3 of voters are in favor of candidate A and 1/3 in favor of candidate B

The odds of candidate A

$$\text{Odds } [A] = \frac{\pi}{1-\pi} = \frac{2/3}{1-2/3} = \frac{2}{1}$$

Odds of a random event - Example

Assume that 2/3 of voters are in favor of candidate A

and 1/3 in favor of candidate B

The odds of candidate A

$$\text{Odds [A]} = \frac{\pi}{1-\pi} = \frac{2/3}{1-2/3} = \frac{2}{1}$$

The probability of voting for A is twice
the probability of voting for other candidate

Odds of a random event - Example

Assume that 2/3 of voters are in favor of candidate A and 1/3 in favor of candidate B

The odds of candidate A

$$\text{Odds}[A] = \frac{\pi}{1-\pi} = \frac{2/3}{1-2/3} = \frac{2}{1}$$

The odds of candidate A are 2-to-1

INDICATOR of a random event

Definition: The indicator r.v. of event A has pdf

$$y \begin{cases} 1 & \text{if event A occurs} \\ 0 & \text{otherwise} \end{cases}$$

where P[A] = π

INDICATOR RANDOM VARIABLE

Definition: The indicator r.v. of event A has pdf

$$y \begin{cases} 1 & \text{with probability } P[A] = \pi \\ 0 & \text{otherwise} \end{cases}$$

BERNOULLI random variable

Definition: A r.v. Y is called Bernoulli if its pdf is

$$y \begin{cases} 1 & \text{with probability } \pi \\ 0 & \text{with probability } 1-\pi \end{cases}$$

P[Y = 1] = π

The odds of y *being equal to 1 is*

$$\text{Odds } [Y = 1] = \frac{\pi}{1-\pi}$$

BERNOULLI random variable

Definition: A r.v. Y is called Bernoulli if its pdf is

$$y \begin{cases} 1 & \text{with probability} \quad \pi \\ 0 & \text{with probability} \quad 1-\pi \end{cases}$$

$$\begin{aligned} E[Y] &= 1\,P[Y=1] + 0\,P[Y=0] \\ &= 1 \quad \pi \;\; + 0 \;\; (1-\pi) \\ &= \pi \end{aligned}$$

BERNOULLI random variable

Definition: A r.v. Y is called Bernoulli if its pdf is

$$y \begin{cases} 1 & \text{with probability} \quad \pi \\ 0 & \text{with probability} \quad 1-\pi \end{cases}$$

$$E[Y] = P[Y = 1]$$

BERNOULLI random variable

Definition: A r.v. Y is called Bernoulli if its pdf is

$$P[Y = y] = \pi^{y} (1-\pi)^{1-y} \qquad y = 0,1$$

BERNOULLI random variable - Example

A Bernoulli r.v. is defined for customer gender as

$$y \begin{cases} 1 & \text{if customer is male} \quad \text{wp. } \pi \\ 0 & \text{if customer is female} \quad \text{wp. } 1-\pi \end{cases}$$

BERNOULLI random variable - Example

A Bernoulli r.v. is defined for customer gender as

$$
y \begin{cases} 1 & \text{if category male} \quad \text{wp. } \pi \\ 0 & \text{if category female} \quad \text{wp. } 1-\pi \end{cases}
$$

$$
\frac{P[Y=1]}{P[Y=0]} = \frac{\pi}{1-\pi}
$$

the odds of a male customer

BERNOULLI random variable - Example

A Bernoulli r.v. is defined for customer gender as

$$y \begin{cases} 1 & \text{if category male} \quad \text{wp. } \pi \\ 0 & \text{if category female} \quad \text{wp. } 1-\pi \end{cases}$$

$$\frac{P[Y=1]}{P[Y=0]} = \frac{\pi}{1-\pi}$$

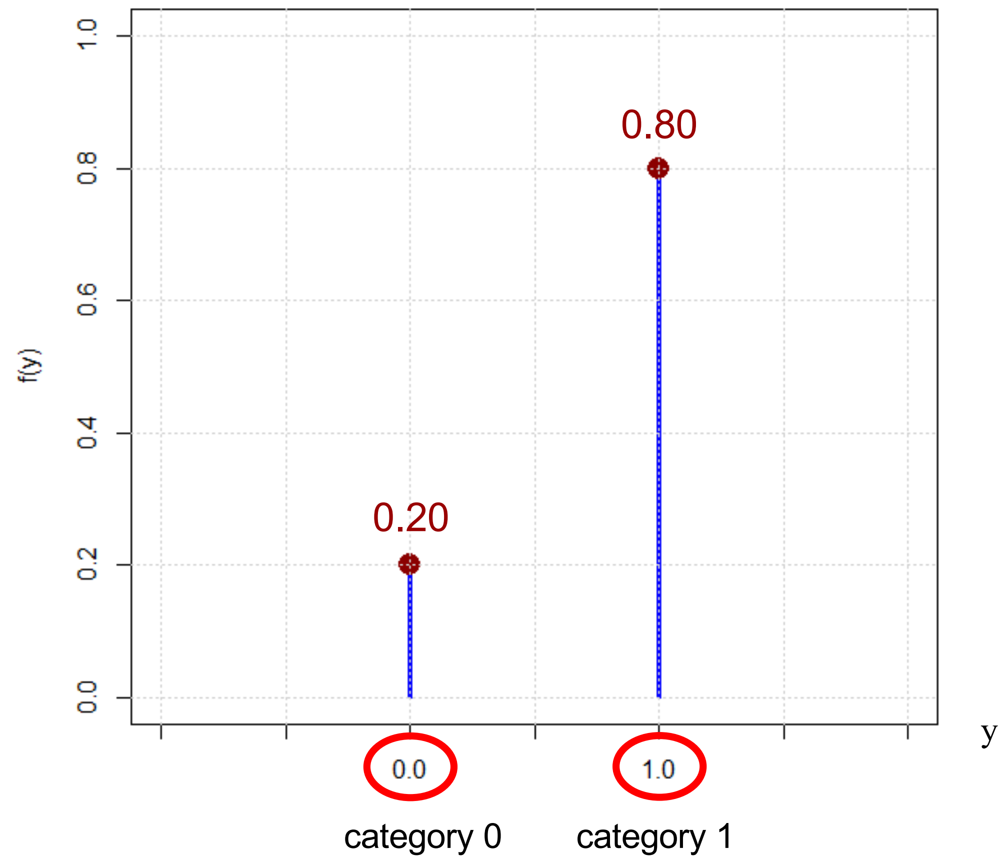
how much likely is a customer male than female

Bernoulli probability function

$$y \begin{cases} 1 & wp. \quad 0.80 \\ 0 & wp. \quad 0.20 \end{cases}$$

$$f(y) = P[Y = y]$$
$$= 0.8^{\,y}\, 0.2^{\,1-y}$$

$$y = 0,1$$



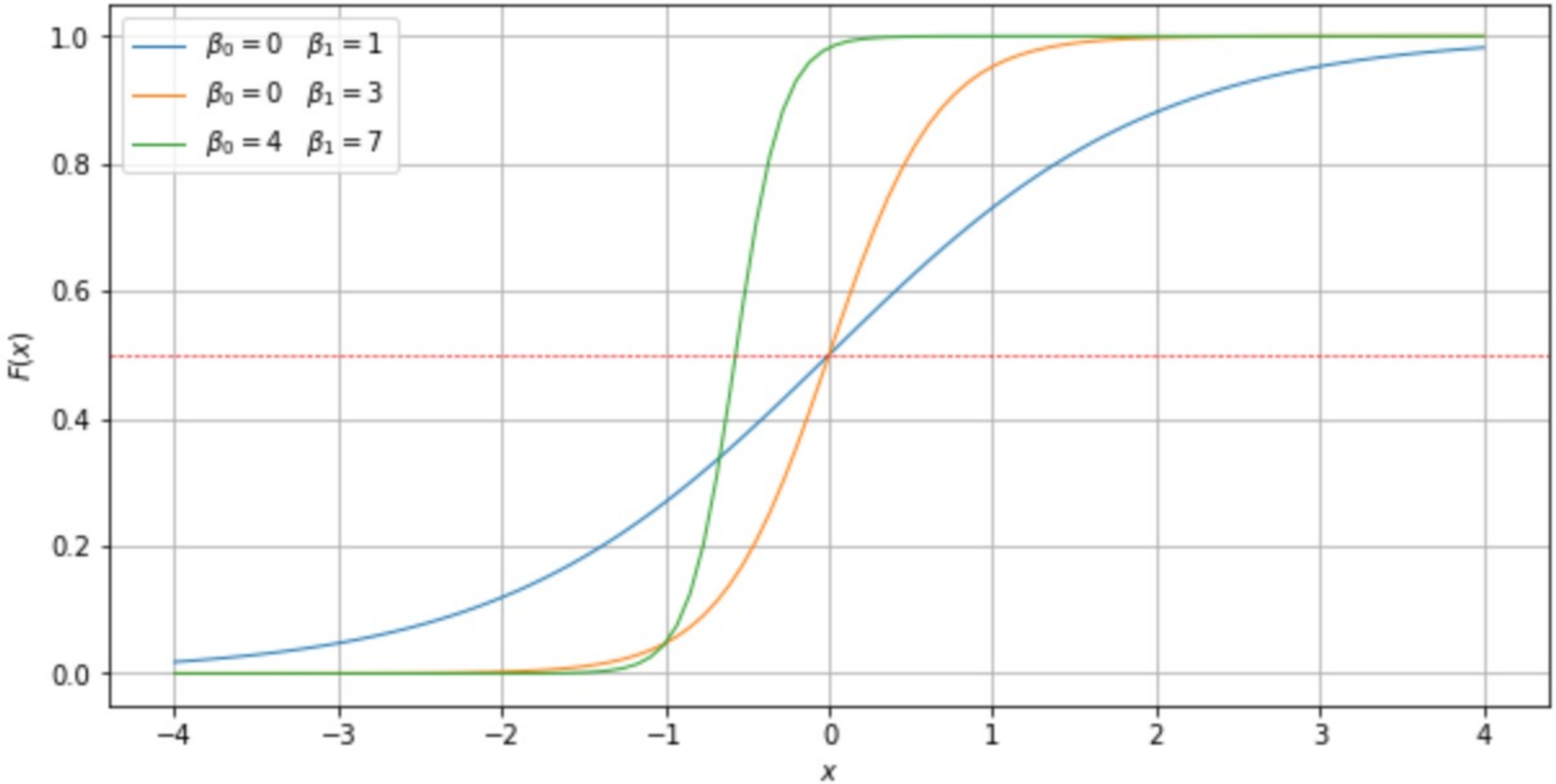
LOGISTIC RANDOM VARIABLE

A continuous random variable X is called Logistic if

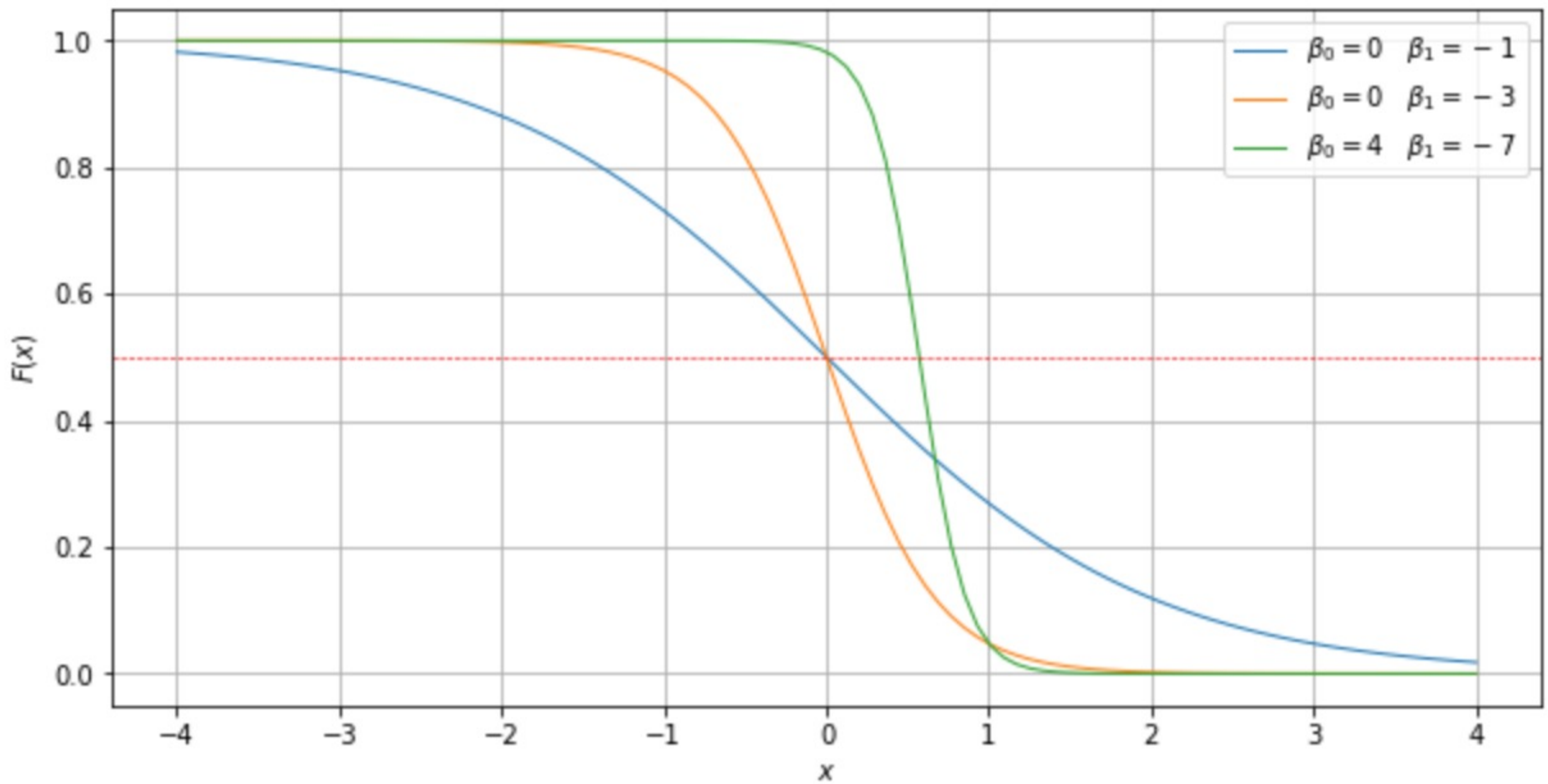
pdf $$f(x) = k \frac{e^{-\beta_0 - \beta_1 x}}{\left[1 + e^{-\beta_0 - \beta_1 x}\right]^2} \qquad -\infty < x < \infty$$

cdf $$F(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

Logistic distributions (cdf) - β_1 positive



Logistic function - β_1 negative



Logistic distributions (cdf) - β_1 positive

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def logistic(x, beta0, beta1):
    return 1.0 / (1.0 + np.exp(-beta0 - np.dot(beta1, x)))
```

```
x = np.linspace(-4, 4, 100)
plt.figure(figsize=(10,5))
plt.plot(x,logistic(x,0,1), label=r"$\beta_0 = 0\quad\beta_1=1$",lw=1)
plt.plot(x,logistic(x,0,3), label=r"$\beta_0 = 0\quad\beta_1=3$",lw=1)
plt.plot(x,logistic(x,4,7), label=r"$\beta_0 = 4\quad\beta_1=7$",lw=1)
plt.axhline(y=0.50,linestyle='--',c='r',lw=0.6)
plt.xlabel("$x$")
plt.ylabel("$F(x)$")
plt.legend()
```

Logistic Regression

Introduction

EXAMPLE

Predict if an English citizen agrees with Brexit

X: years of working experience

Y: Agrees (A)

Disagrees (D)

X	Y
33	A
27	A
12	D
41	A
	.
	.
19	D

Logistic Regression

Predict if an English citizen agrees with Brexit

X: years of working experience

Y: category 1 (agrees)

category 0 (disagrees)

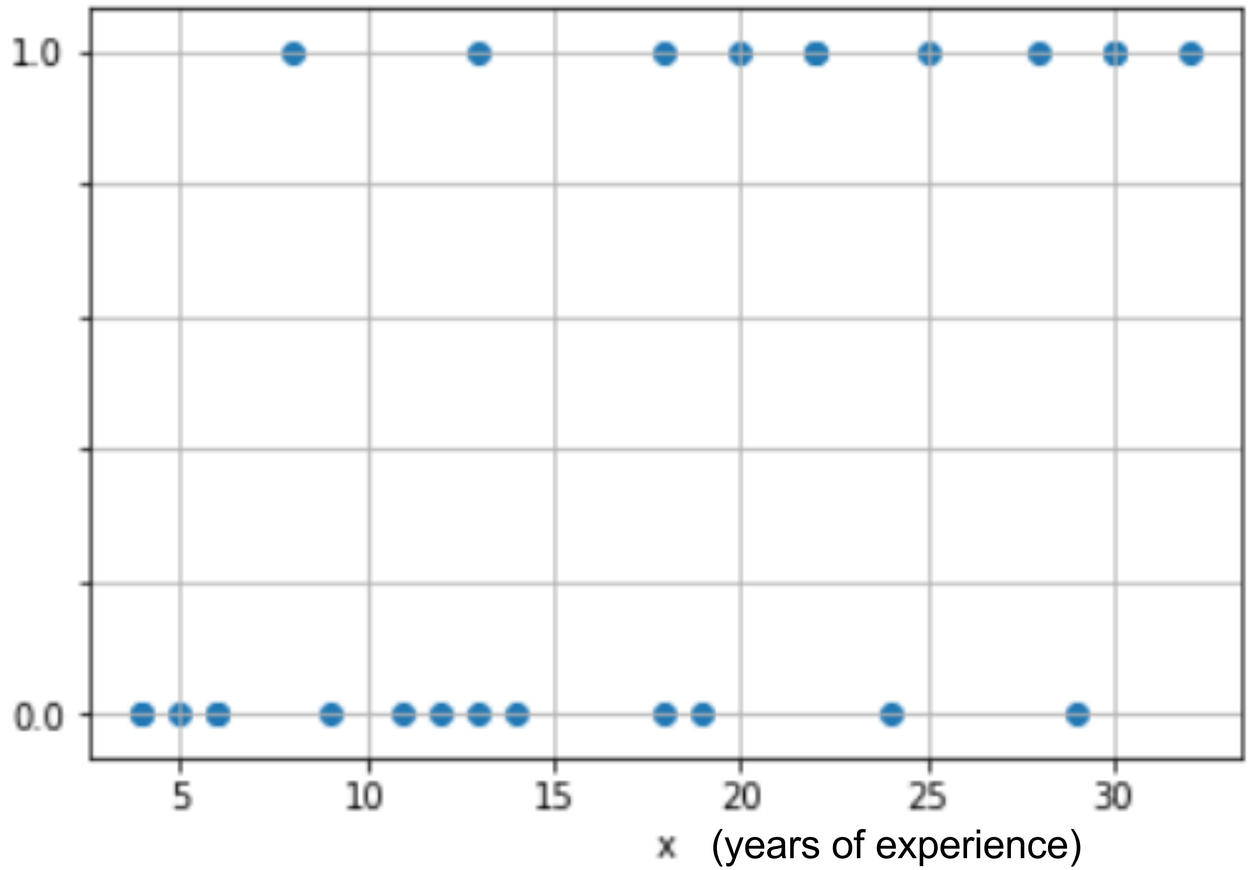
X	Y
33	1
27	1
12	0
41	1
	.
	.
19	0

Scatterplot

category 1

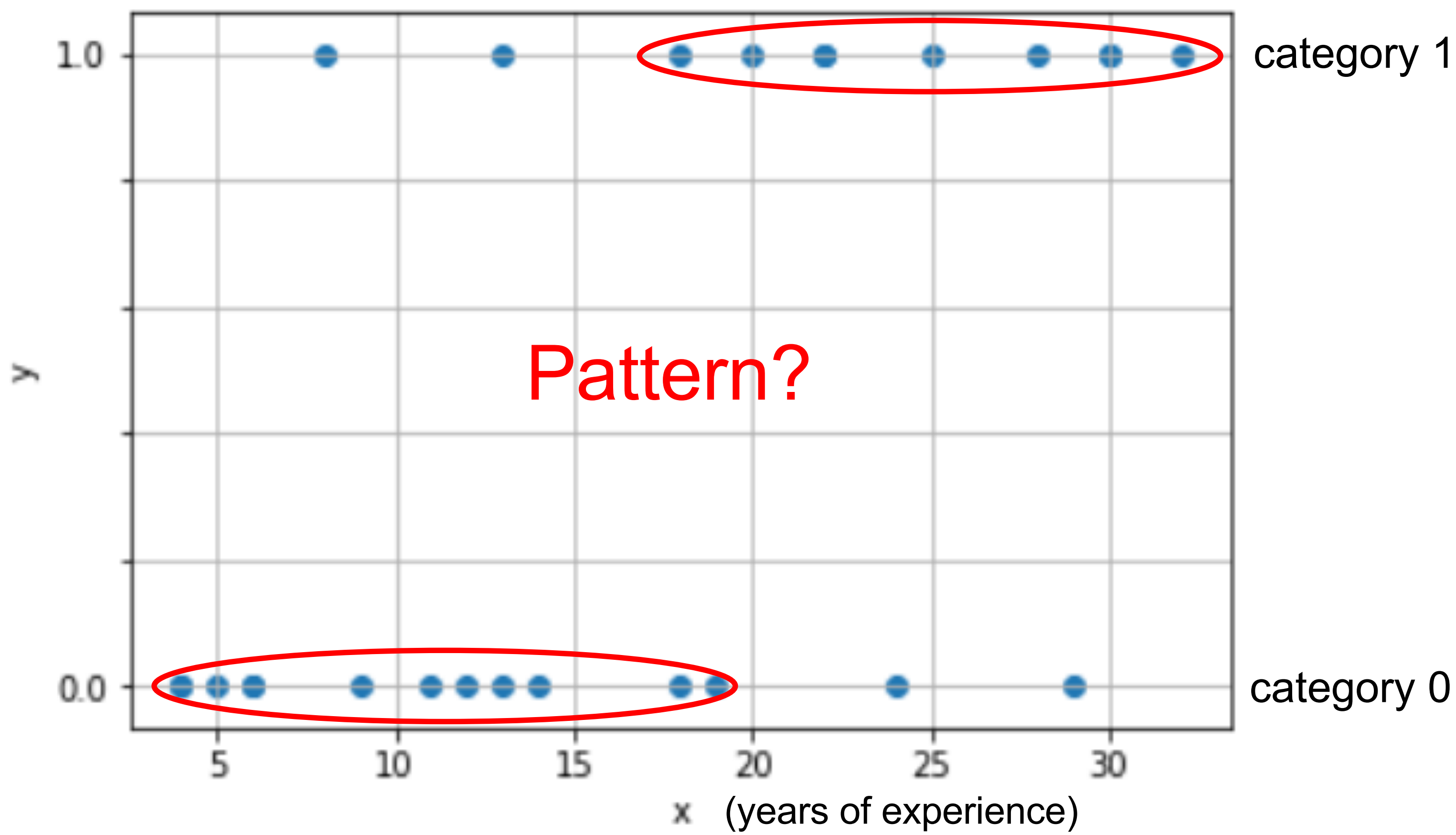
Y is a Bernoulli random variable

category 0



X	Y
33	1
27	1
12	0
41	1
	.
	.
19	0

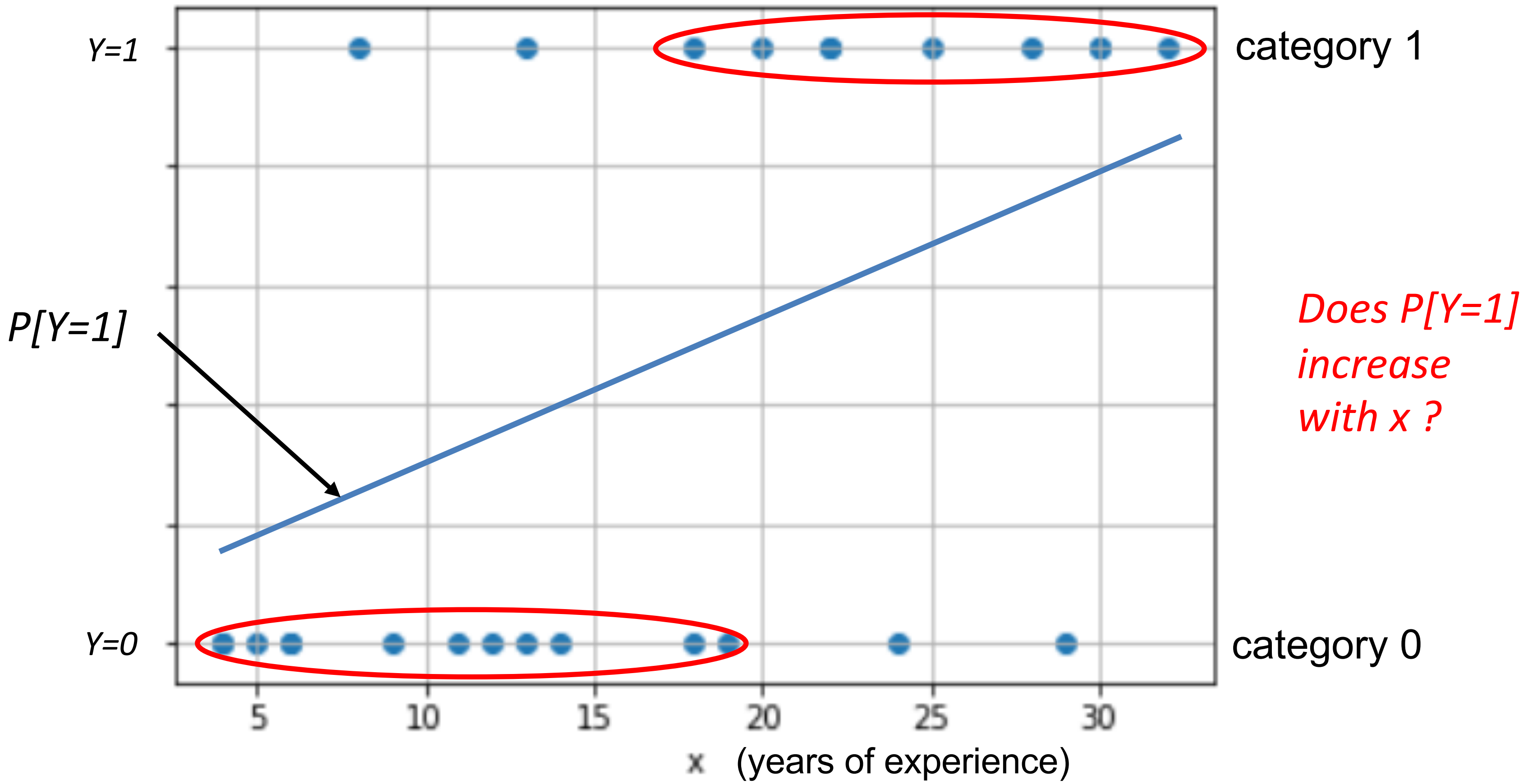
Scatterplot



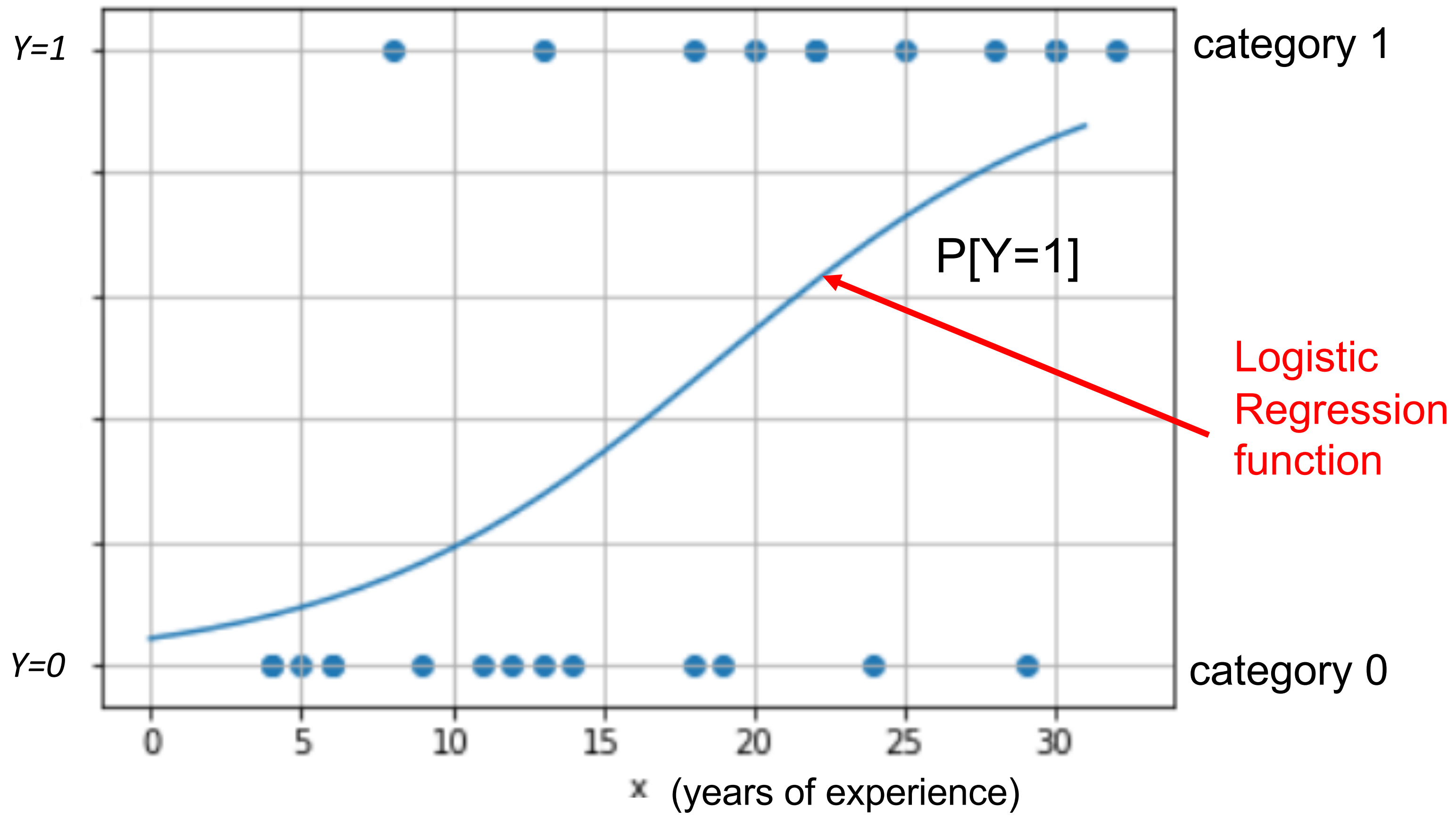
Logistic regression

Is there a relation between Y and X?

Scatterplot

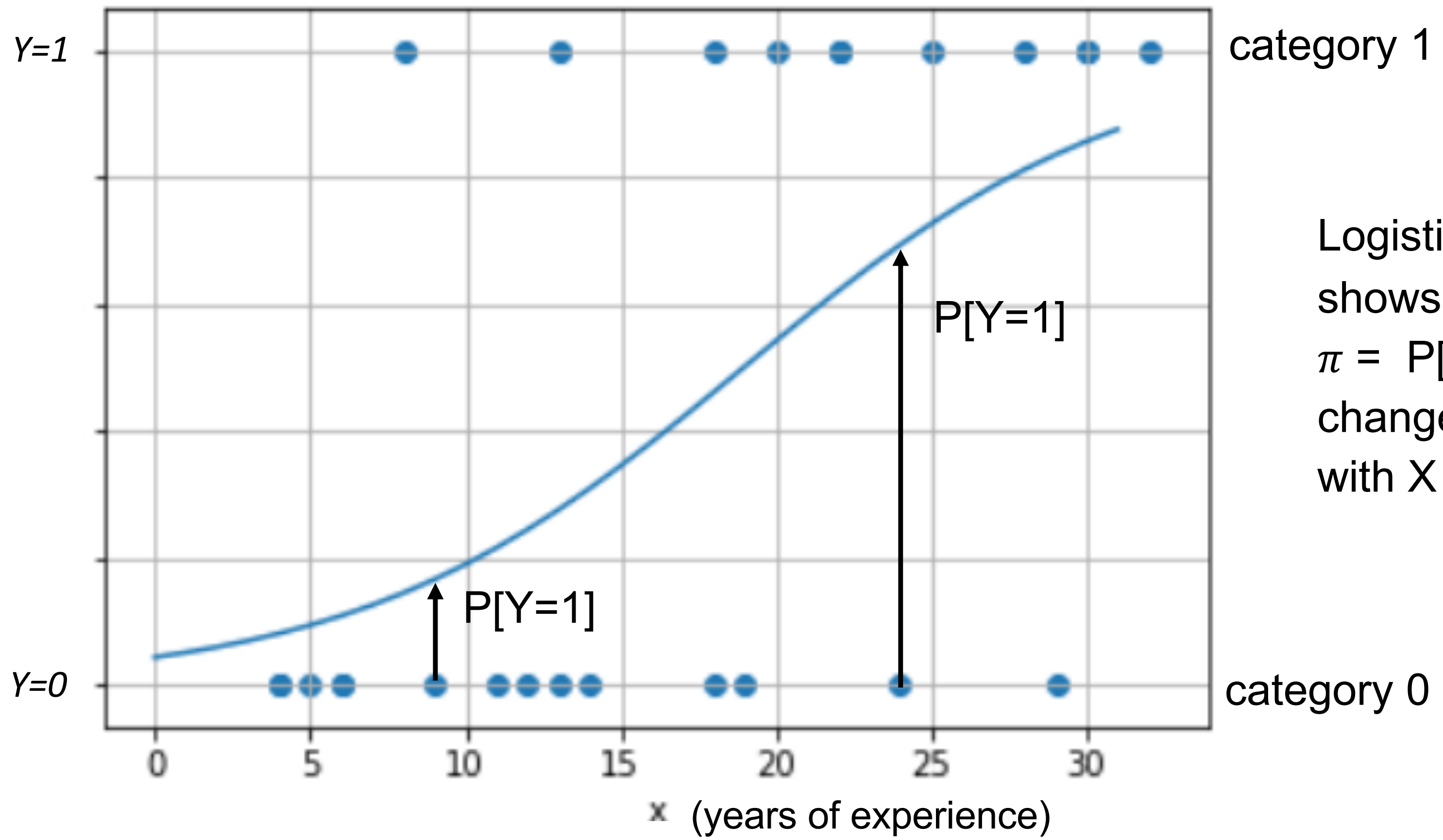


Is there a relation between Y and X?



Logistic regression

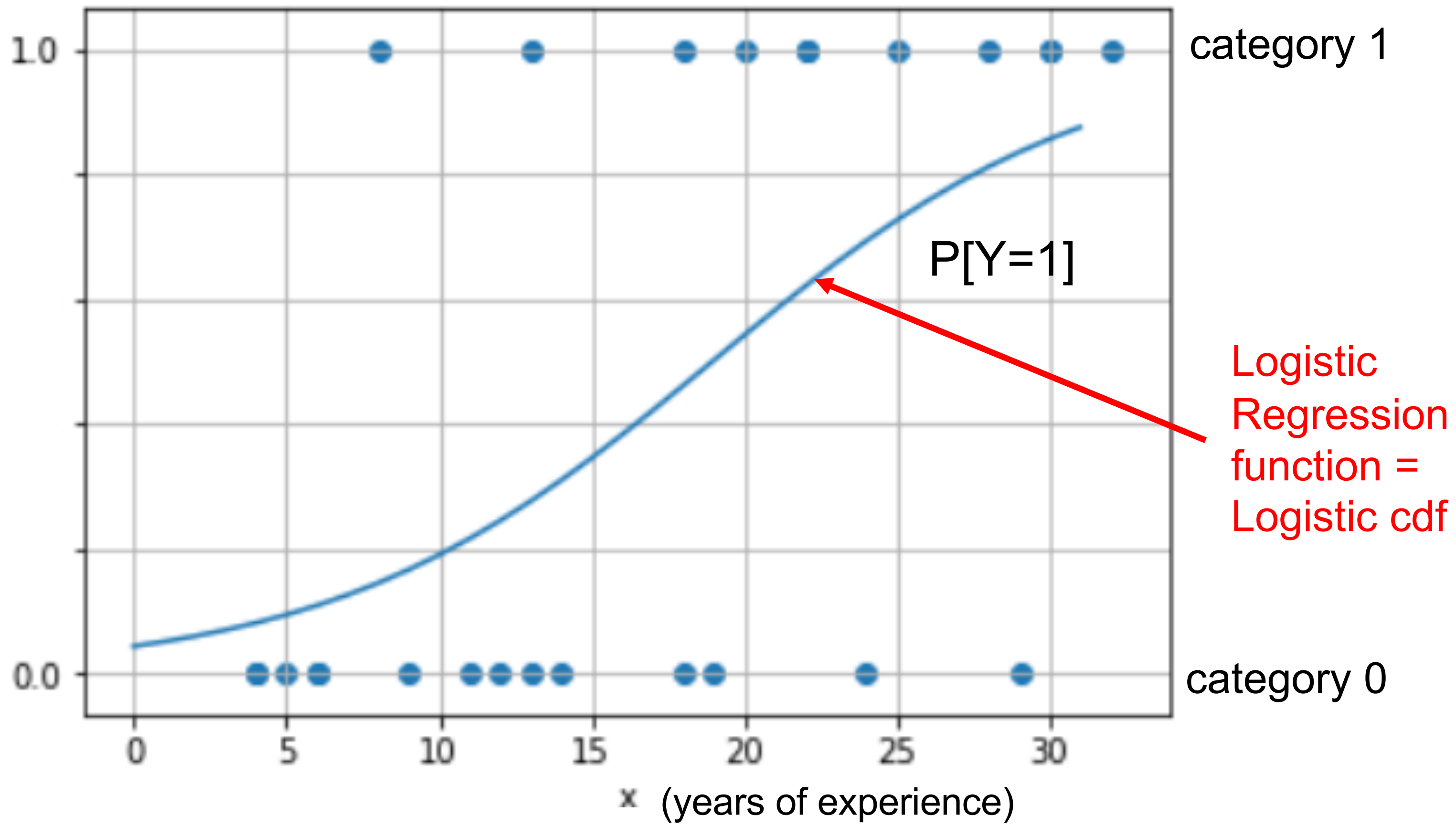
At each x-value there exists a Bernoulli random variable Y



Logistic curve shows that π = P[Y=1] changes with X

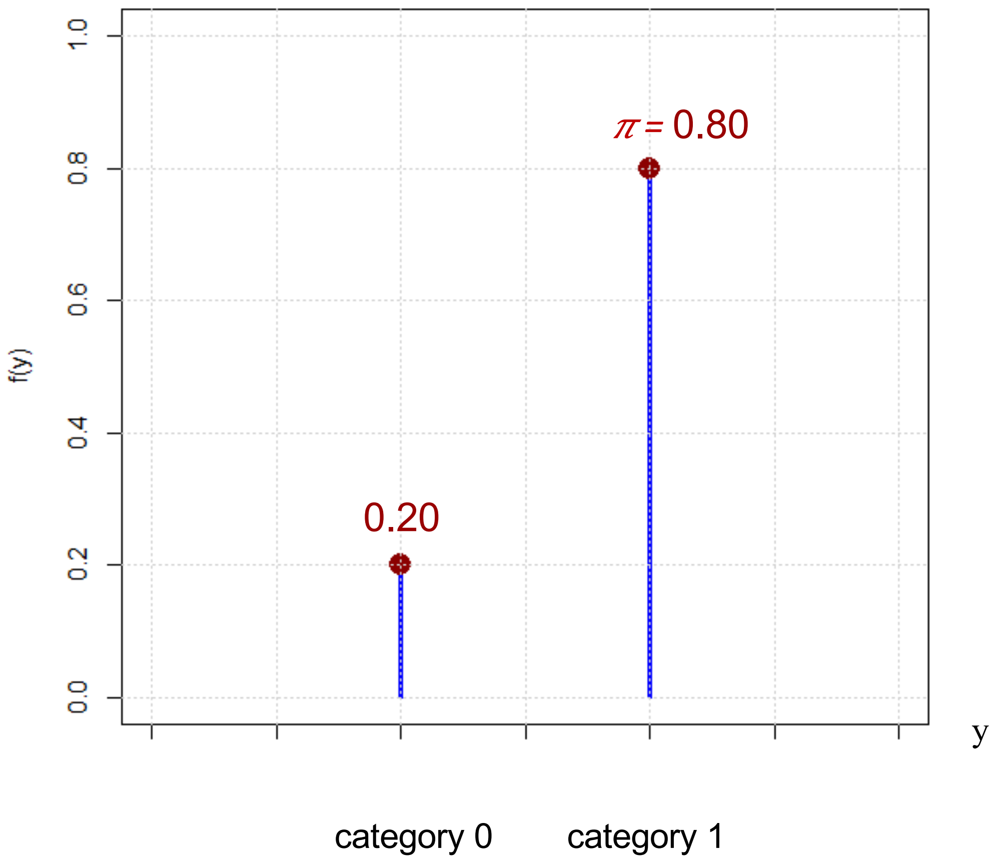
Logistic regression

Logistic regression estimates the probability that a person with certain experience is in category 1



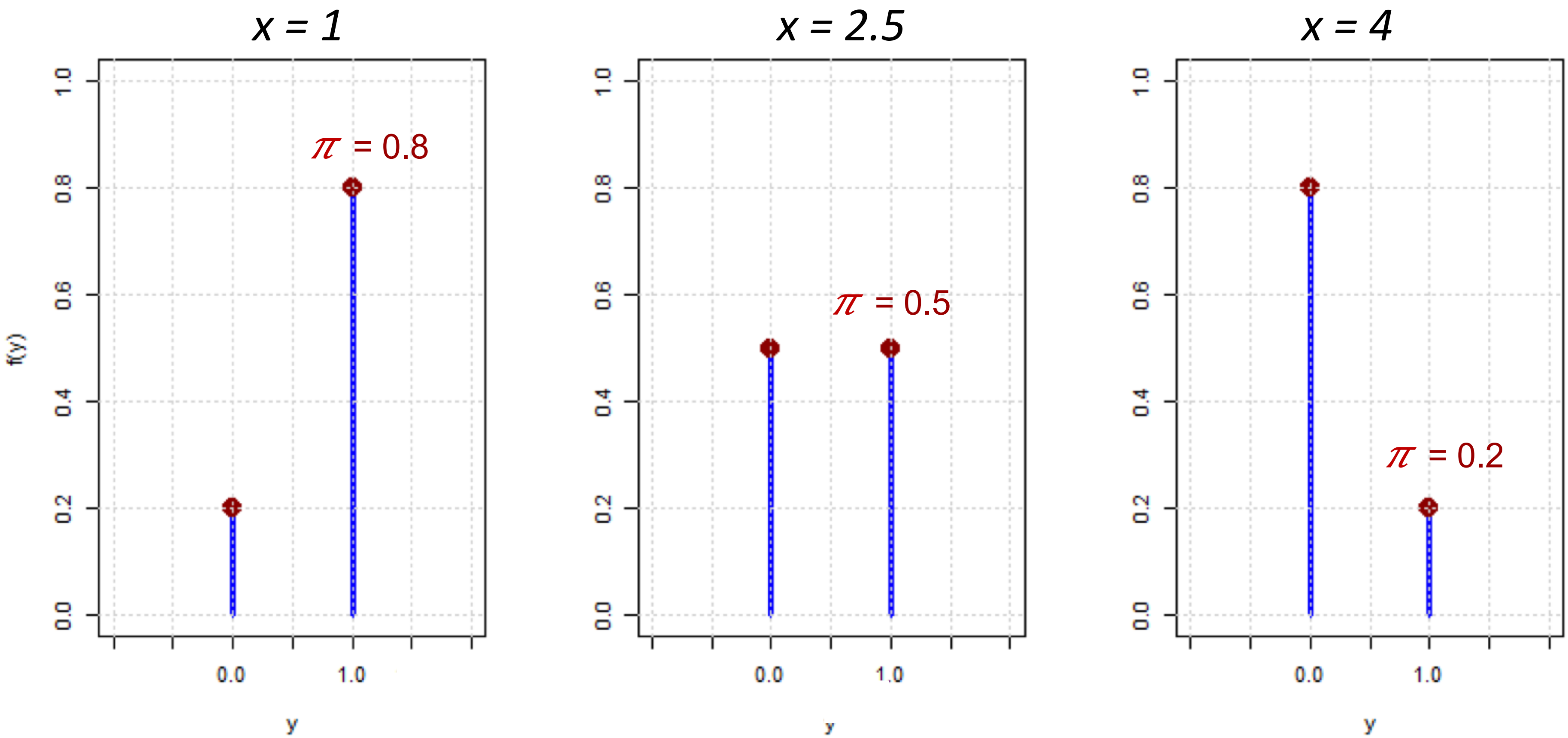
Bernoulli probability function

$$y \begin{cases} 1 & \text{wp.} \quad \pi = 0.80 \\ 0 & \text{wp.} \quad 1-\pi = 0.20 \end{cases}$$



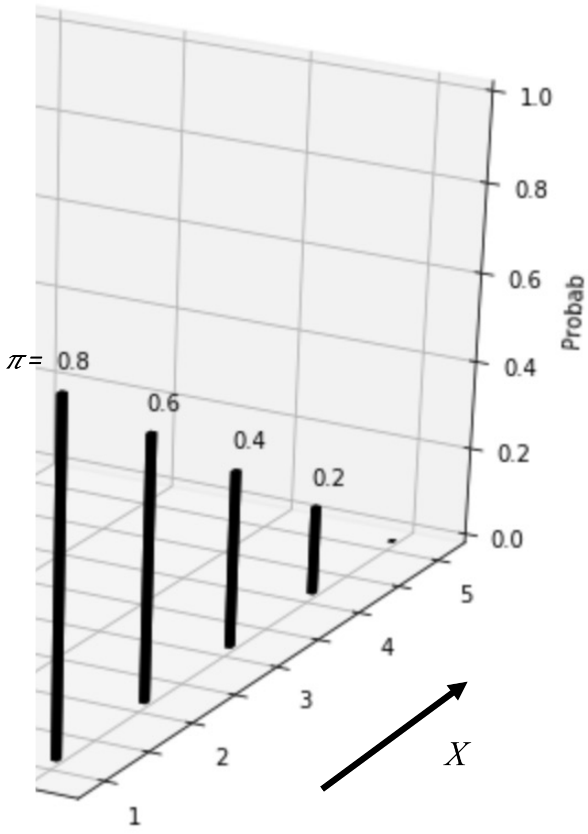
Suppose that $\pi = P[Y=1]$ changes with variable X (not shown here)

Bernoulli probability functions at 3 different x-values



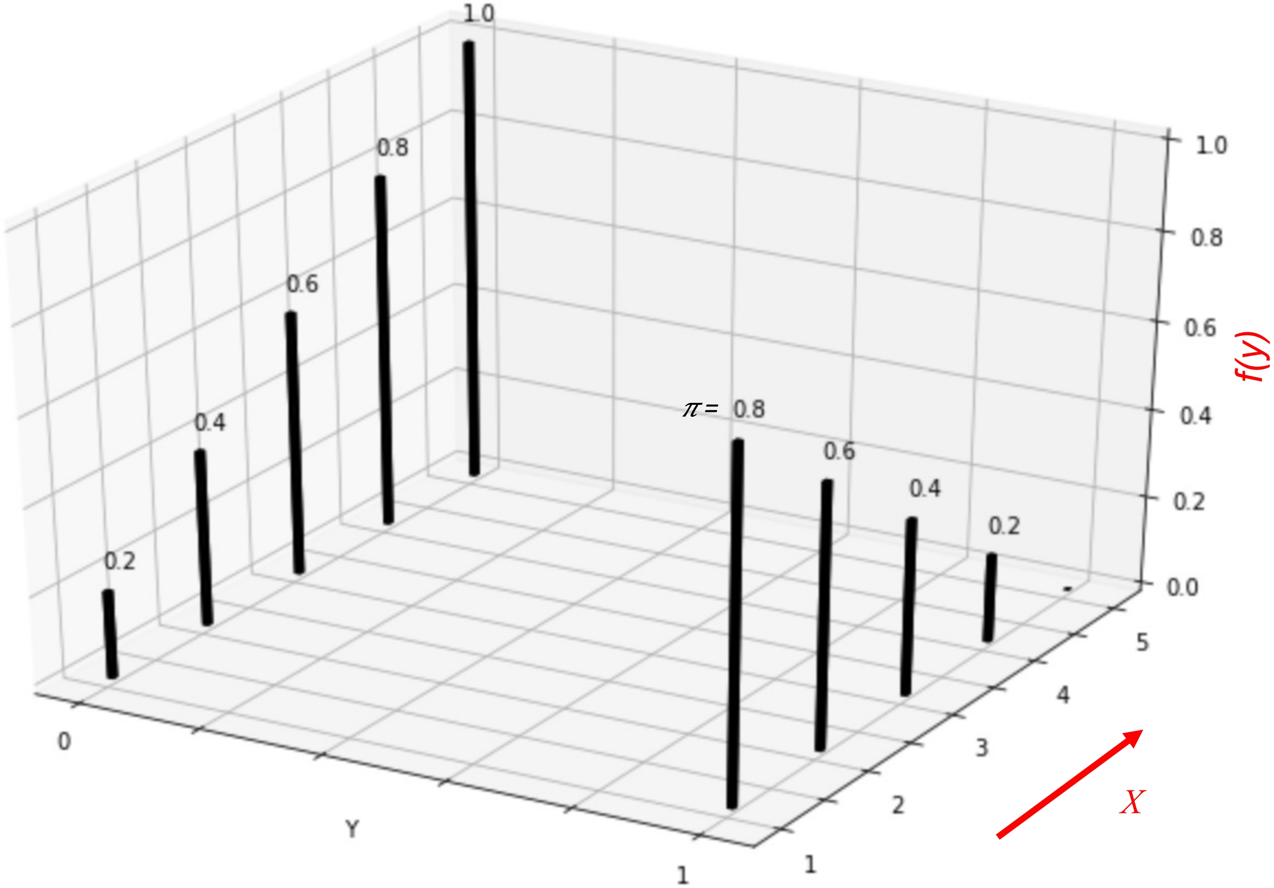
Logistic regression

- There is a pdf for Y at each value of X
- As X increases the pdf changes



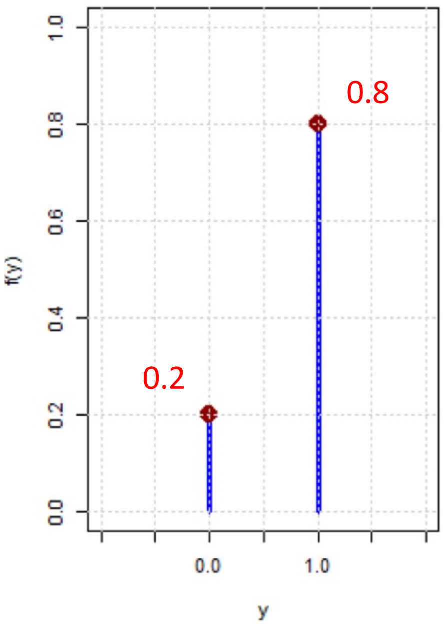
Logistic regression

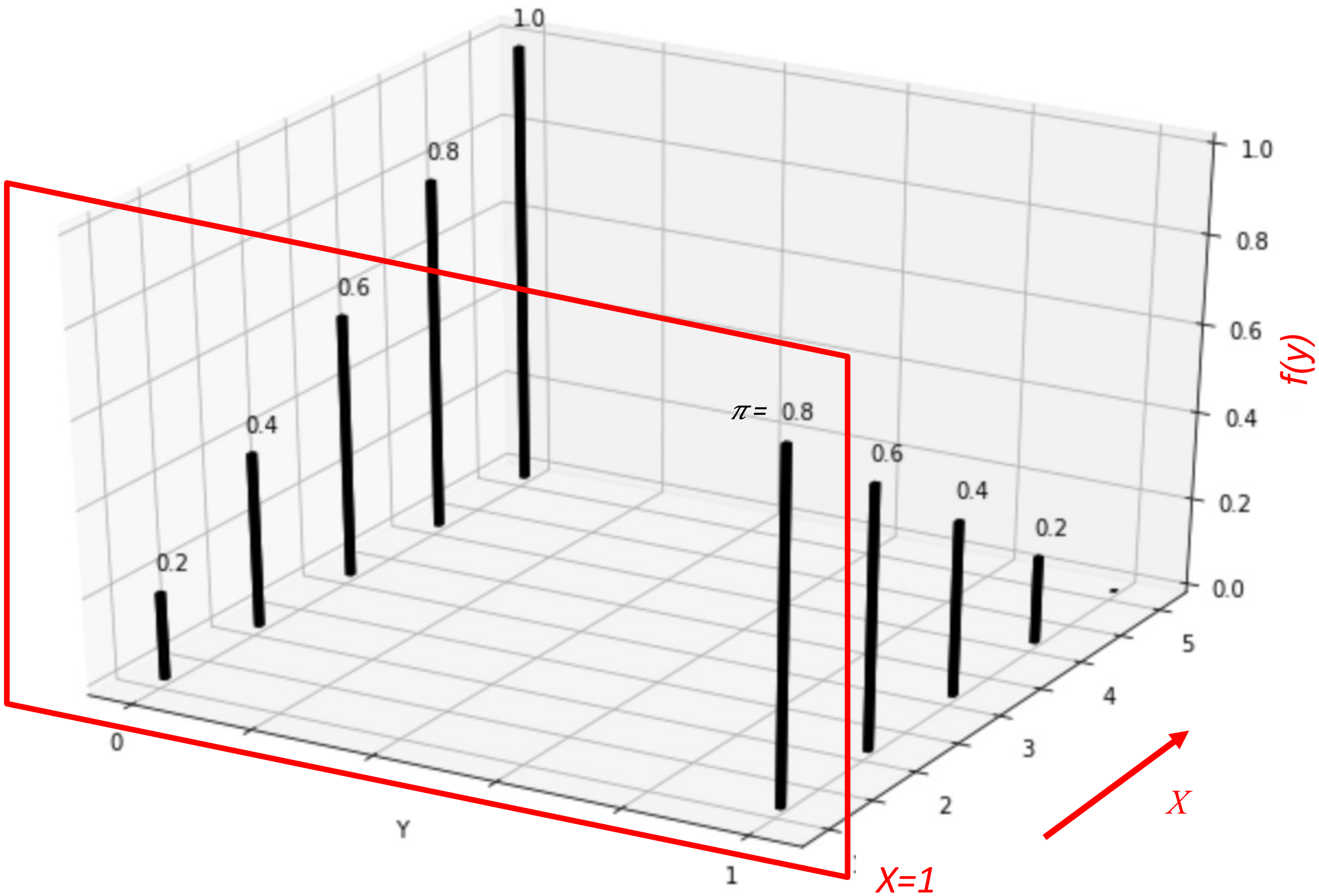
- There is a pdf for Y at each value of X
- As X increases the pdf changes



Logistic regression

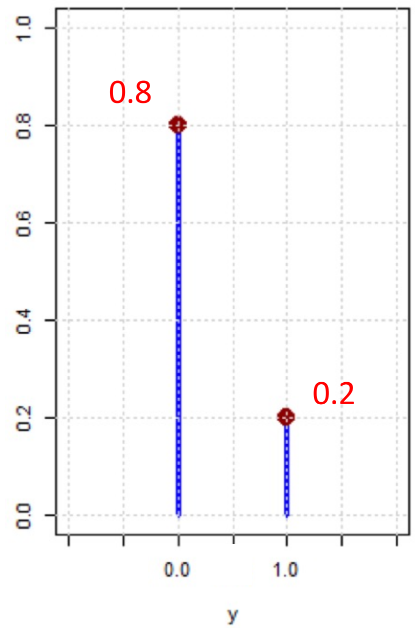
- There is a pdf for Y at each value of X
- As X increases the pdf changes
- For X=1 the pdf of Y is

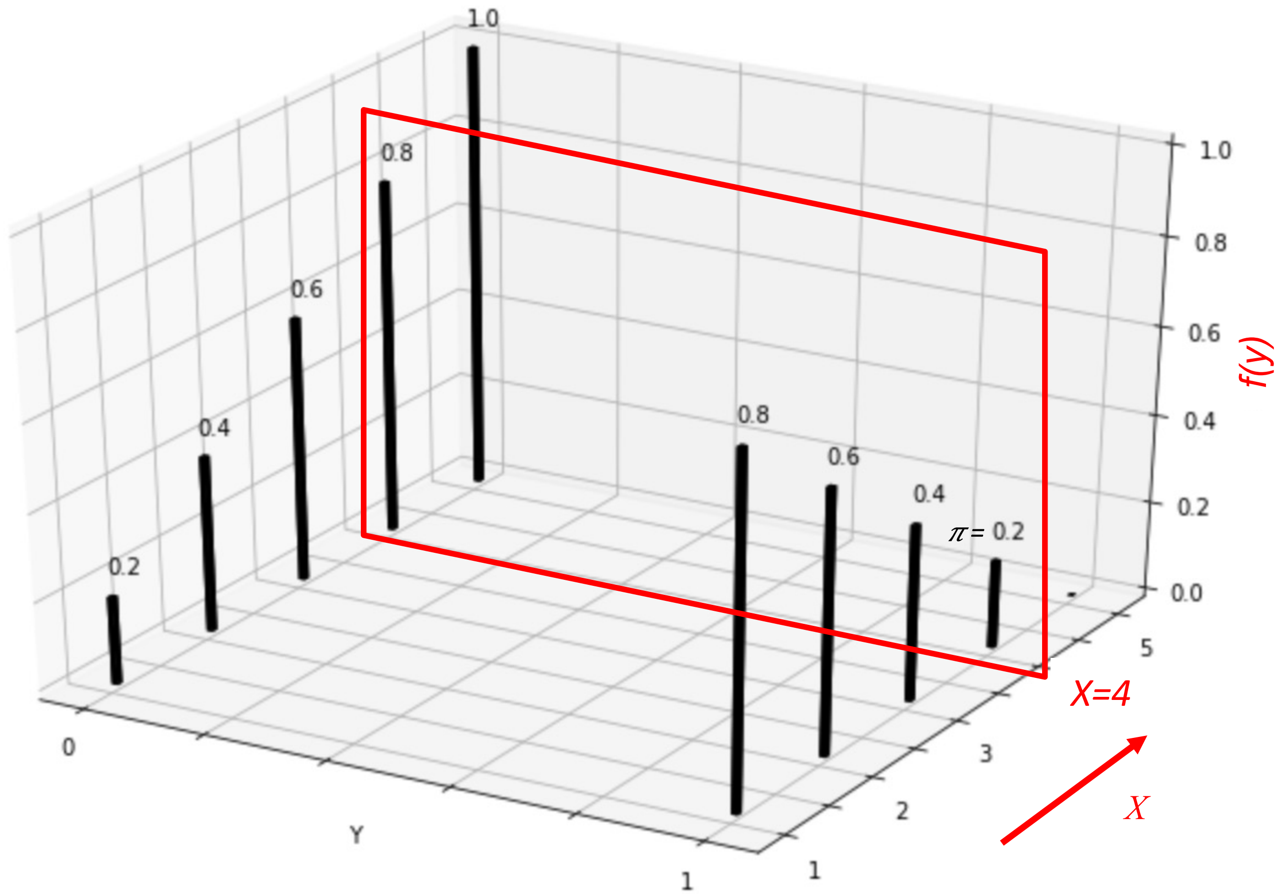




Logistic regression

- There is a pdf for Y at each value of X
- As X increases the pdf changes
- For X=4 the pdf of Y is

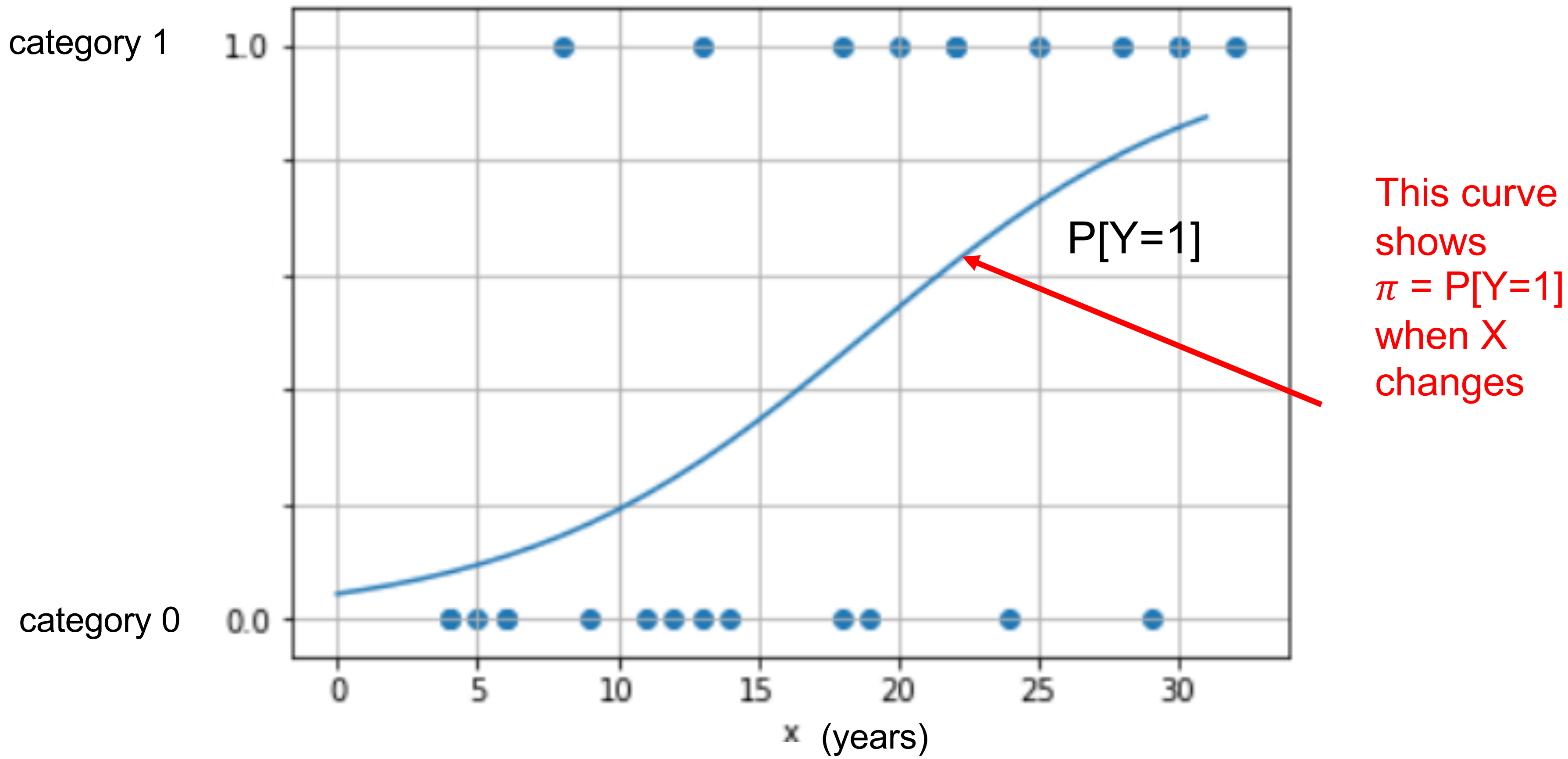




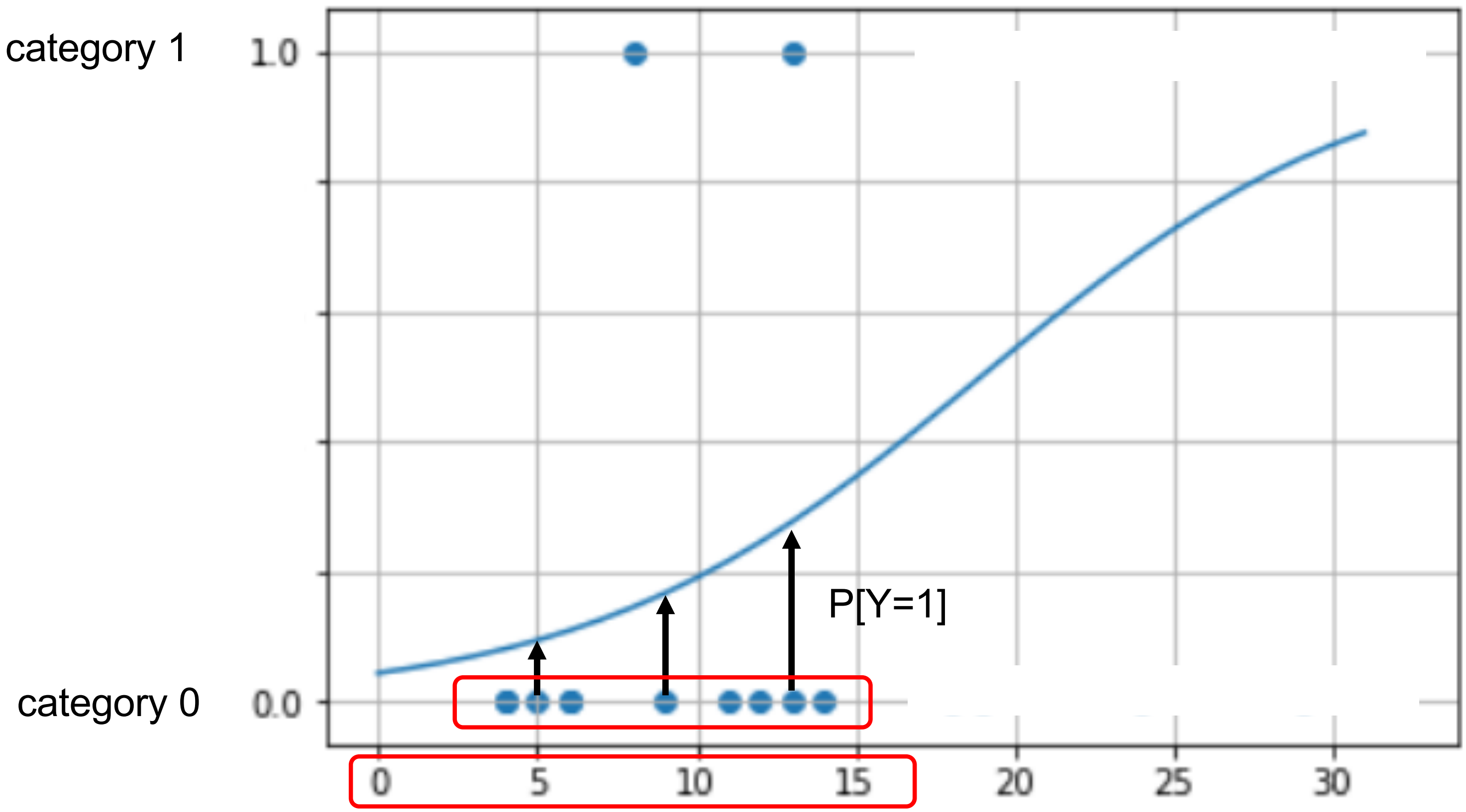
Logistic regression

Is there a relation between P[Y=1] and X?

Logistic regression

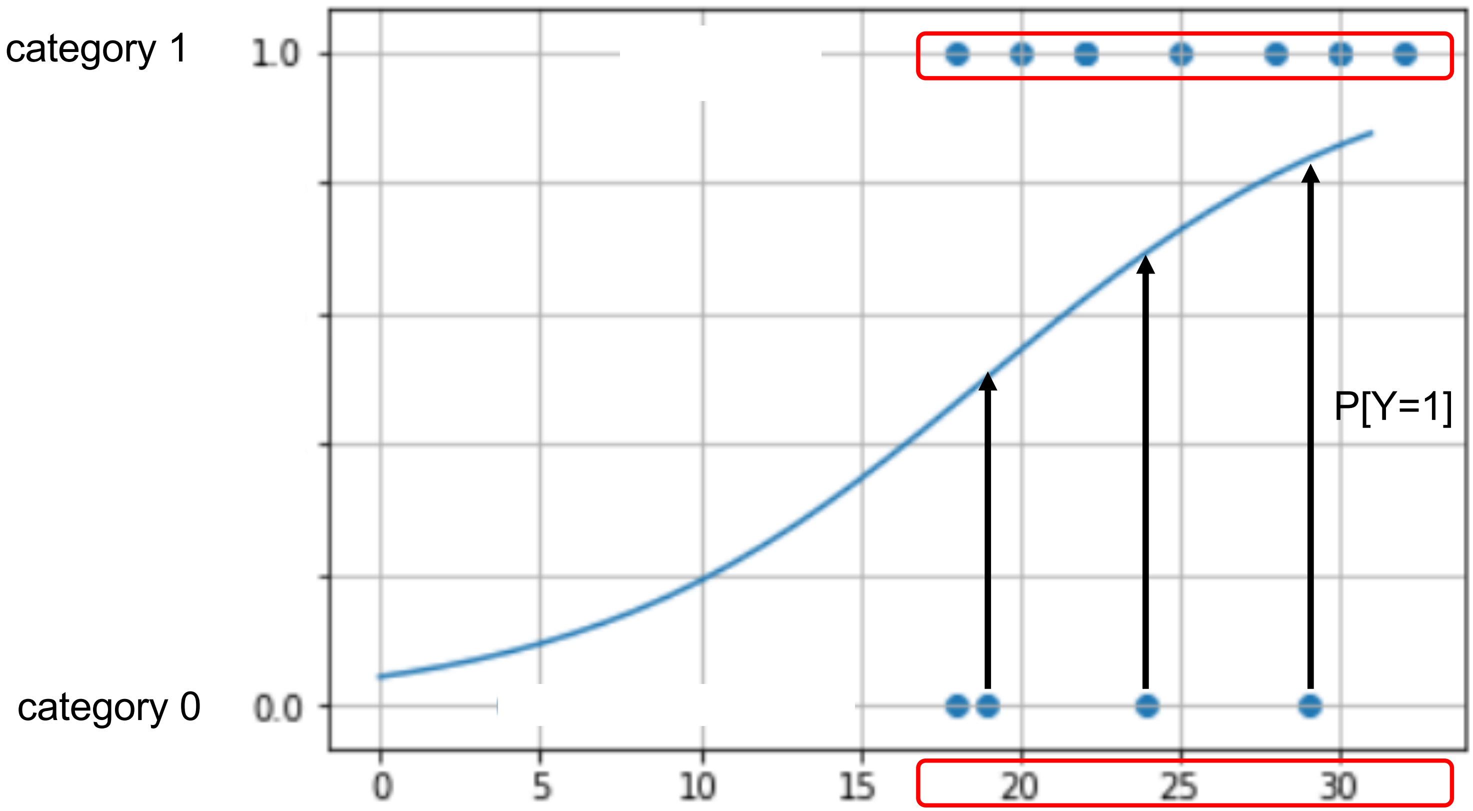


Logistic regression



When x is small, P[Y=1] is small, so most observations are in category 0

Logistic regression



When x is large, P[Y=1] is large, so most observations are in category 1

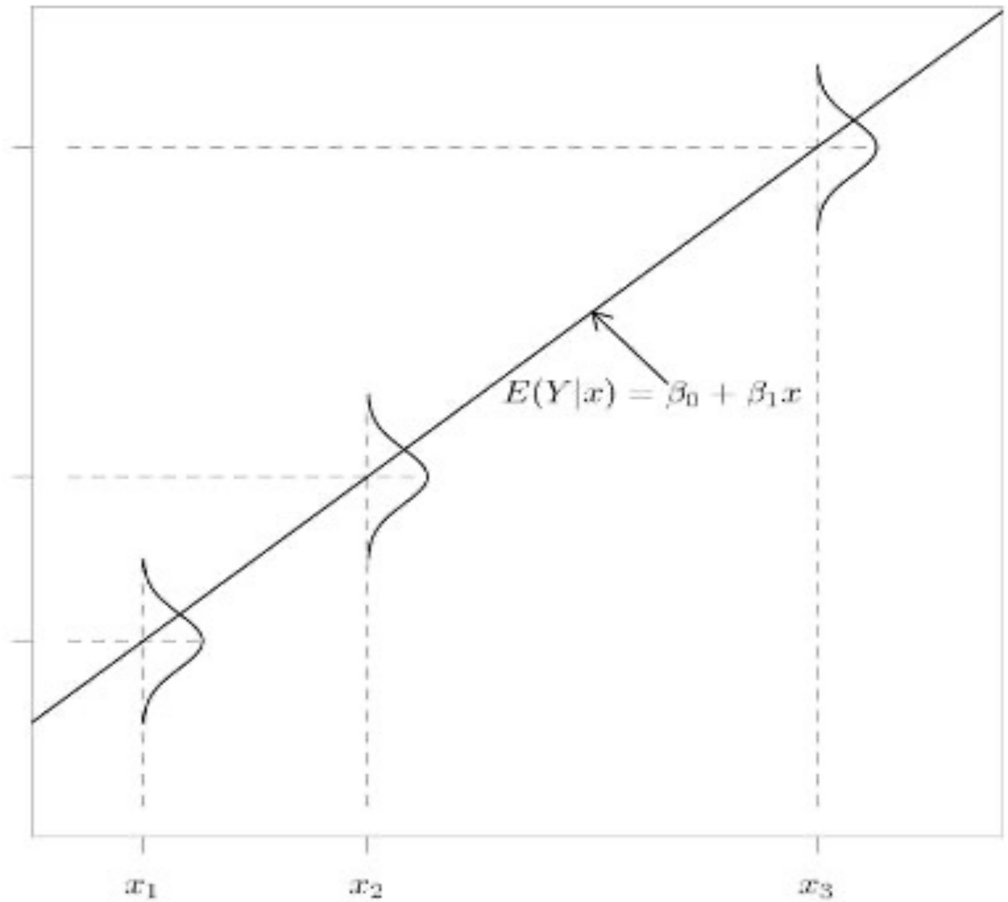
Logistic regression

Is there a relation between E[Y] and X?

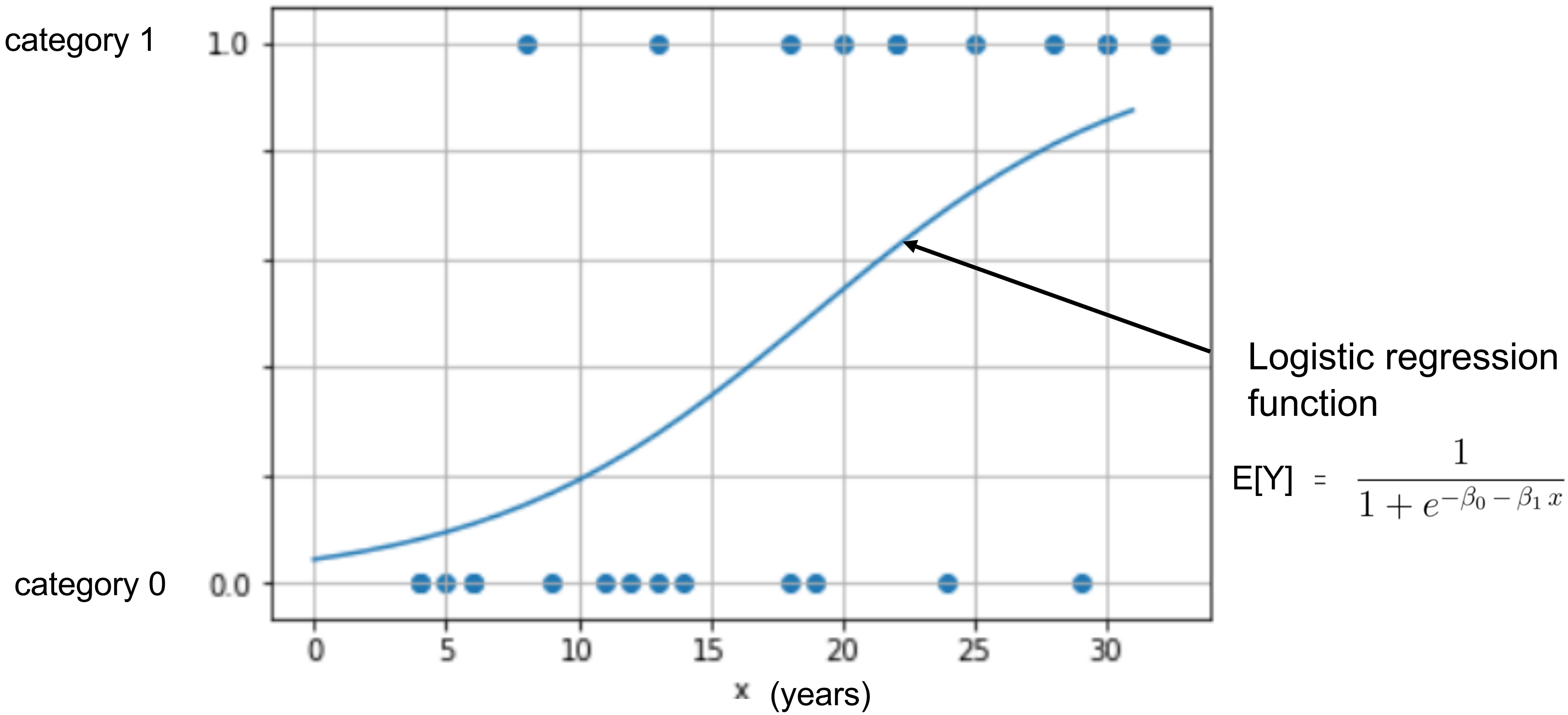
LINEAR REGRESSION function

linear regression function

$$E[Y] = \beta_0 + \beta_1 x$$



LOGISTIC REGRESSION function



$$E[Y] = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

Simple Logistic Regression

Logistic Regression

Logistic Regression models estimate the probability that a data point belongs to category [Y=1]

LOGISTIC REGRESSION ASSUMPTION

As x increases, π varies along the logistic cdf

$$\pi = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

LOGISTIC REGRESSION MODEL

.

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

logistic regression function

LOGISTIC REGRESSION MODELS - EQUIVALENT

.

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

logistic regression function

$$\underbrace{\ln\left(\frac{\pi}{1-\pi}\right)} = \beta_0 + \beta_1 x$$

logit regression function

log-odds or
logit of π

Logistic regression Assumption

This regression relation between π_i and x_i

$$\pi_i = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_i}}$$

Logistic regression Assumption

This regression relation between π_i and x_i

$$\pi_i = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_i}}$$

is estimated by

$$\hat{\pi}_i = \frac{1}{1 + e^{-b_0 - b_1 x_i}}$$

PREDICTIONS

- *P[Y=1]* is predicted with

$$\hat{\pi}_i = \frac{1}{1 + e^{-b_0 - b_1 x_i}}$$

- The category of Y is predicted by the following rule
  - if $\hat{\pi}_i \geq 0.5$ predict $\hat{y} = 1$
  - $\hat{\pi}_i < 0.5$ predict $\hat{y} = 0$

The cutoff may be different

Logistic regression Assumptions

- Linear regression assumptions do not apply
- π changes with x
- As x increases, π changes, moving along an *S* shape curve (the logistic cdf is the S shape curve)
- There is a Y Bernoulli r.v. at each different x
- For different X, the *Y* variables are independent

PREDICTIONS

- Probabilities are predicted by

$$\hat{\pi}_i = \frac{1}{1 + e^{-b_0 - b_1 x_i}}$$

- How are (b_0, b_1) found?

LOGISTIC REGRESSION MODELS - EQUIVALENT

.

$$\ln\left(\frac{\pi}{1-\pi}\right) = b_0 + b_1\,x$$

cannot use OLS
to find b_0 and b_1

$$\pi = \frac{1}{1+e^{-b_0 - b_1\,x}}$$

instead we use the
maximum likelihood method

PARAMETERS ESTIMATION

Predict if an English citizen agrees with Brexit

X_i : *years of working experience*

$$y_i \begin{cases} 1 & \text{with probability } \pi_i \\ 0 & \text{with probability } 1-\pi_i \end{cases}$$

assume that π exists but it is unknown ↓

i	X_i	Y_i	π_i
1	33	1	π_1
2	27	1	π_2
3	12	0	π_3
4	41	1	π_4
		.	
		.	
n	19	0	π_n

PARAMETERS ESTIMATION

Predict if an English citizen agrees with Brexit

X_i : *years of working experience*

$$y_i \begin{cases} 1 & \text{with probability } \pi_i \\ 0 & \text{with probability } 1-\pi_i \end{cases}$$

Assume that y is Bernoulli r.v.

$$P[Y = y] = \pi^{y} (1-\pi)^{1-y} \qquad y = 0,1$$

assume that π exists but it is unknown ↓

i	X_i	Y_i	π_i
1	33	1	π_1
2	27	1	π_2
3	12	0	π_3
4	41	1	π_4
		.	
		.	
n	19	0	π_n

PARAMETERS ESTIMATION

Predict if an English citizen agrees with Brexit

X_i : *years of working experience*

$$y_i \begin{cases} 1 & \text{with probability } \pi_i \\ 0 & \text{with probability } 1-\pi_i \end{cases}$$

Then the likelihood of first citizen is

$$P[\, Y_1 = y_1 \,] = \pi_1^{y_1} (1 - \pi_1)^{1-y_1}$$

assume that π exists but it is unknown
↓

i	X_i	Y_i	π_i
1	33	1	π_1
2	27	1	π_2
3	12	0	π_3
4	41	1	π_4
		.	
		.	
n	19	0	π_n

PARAMETERS ESTIMATION

The likelihood of each citizen's category is

$$P[Y_1 = y_1] = \pi_1^{y_1}(1-\pi_1)^{1-y_1}$$
$$P[Y_2 = y_2] = \pi_2^{y_2}(1-\pi_2)^{1-y_2}$$
$$\vdots \qquad \vdots$$
$$P[Y_n = y_n] = \pi_n^{y_n}(1-\pi_n)^{1-y_n}$$

$y_1, y_2, \ldots, y_n = 0$ or 1

assume that π exists but it is unknown
↓

i	X_i	Y_i	π_i
1	33	1	π_1
2	27	1	π_2
3	12	0	π_3
4	41	1	π_4
		.	
		.	
n	19	0	π_n

PARAMETERS ESTIMATION

The likelihood of all of them is given by the joint pdf

$$
\begin{aligned}
P[Y_1 = y_1] &= \pi_1^{y_1}\left(1-\pi_1\right)^{1-y_1} \\
P[Y_2 = y_2] &= \pi_2^{y_2}\left(1-\pi_2\right)^{1-y_2} \\
\vdots \quad & \quad \vdots \\
P[Y_n = y_n] &= \pi_n^{y_n}\left(1-\pi_n\right)^{1-y_n}
\end{aligned}
$$

$$P[Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n] = \prod_{i=1}^{n} \pi_i^{y_i}\left(1-\pi_i\right)^{1-y_i}$$

i	X_i	Y_i	π_i
1	33	1	π_1
2	27	1	π_2
3	12	0	π_3
4	41	1	π_4
		.	
		.	
n	19	0	π_n

PARAMETERS ESTIMATION

The joint pdf and the likelihood

pdf → $$P[Y_1 = y_1, Y_2 = y_2, ..., Y_n = y_n] = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

function of (y_1,y_2,… y_n)
π_1, π_2,… π_n are known

likelihood function → $$L(\pi_1, \pi_2, ..., \pi_n) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

function of (π_1, π_2,… π_n)
y_1,y_2,… y_n are known

LIKELIHOOD FUNCTION

$$L(\pi_1, \pi_2, ..., \pi_n) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

$$\pi_i = \frac{e^{b_0+b_1x_i}}{1 + e^{b_0+b_1x_i}}$$

$$1 - \pi_i = \frac{1}{1 + e^{b_0+b_1x_i}}$$

LIKELIHOOD FUNCTION

$$L(\pi_1, \pi_2, \ldots, \pi_n) = \prod_{i=1}^{n} \pi_i^{y_i} (1-\pi_i)^{1-y_i}$$

$$\pi_i = \frac{e^{b_0+b_1 x_i}}{1+e^{b_0+b_1 x_i}}$$

$$1-\pi_i = \frac{1}{1+e^{b_0+b_1 x_i}}$$

$$L(b_0, b_1) = \prod_{i=1}^{n} \left[\frac{e^{b_0+b_1 x_i}}{1+e^{b_0+b_1 x_i}}\right]^{y_i} \left[\frac{1}{1+e^{b_0+b_1 x_i}}\right]^{1-y_i}$$

$$L(b_0, b_1) = \prod_{i=1}^{n} \frac{\left(e^{b_0+b_1 x_i}\right)^{y_i}}{1+e^{b_0+b_1 x_i}}$$

Method of MLE

Find b_0 and b_1 such that $L(b_0,b_1)$ is as large as possible

LIKELIHOOD FUNCTION

likelihood function

$$L(b_0, b_1) \; = \; \prod_{i=1}^{n} \frac{\left(e^{b_0+b_1 x_i}\right)^{y_i}}{1 + e^{b_0+b_1 x_i}}$$

log-likelihood function

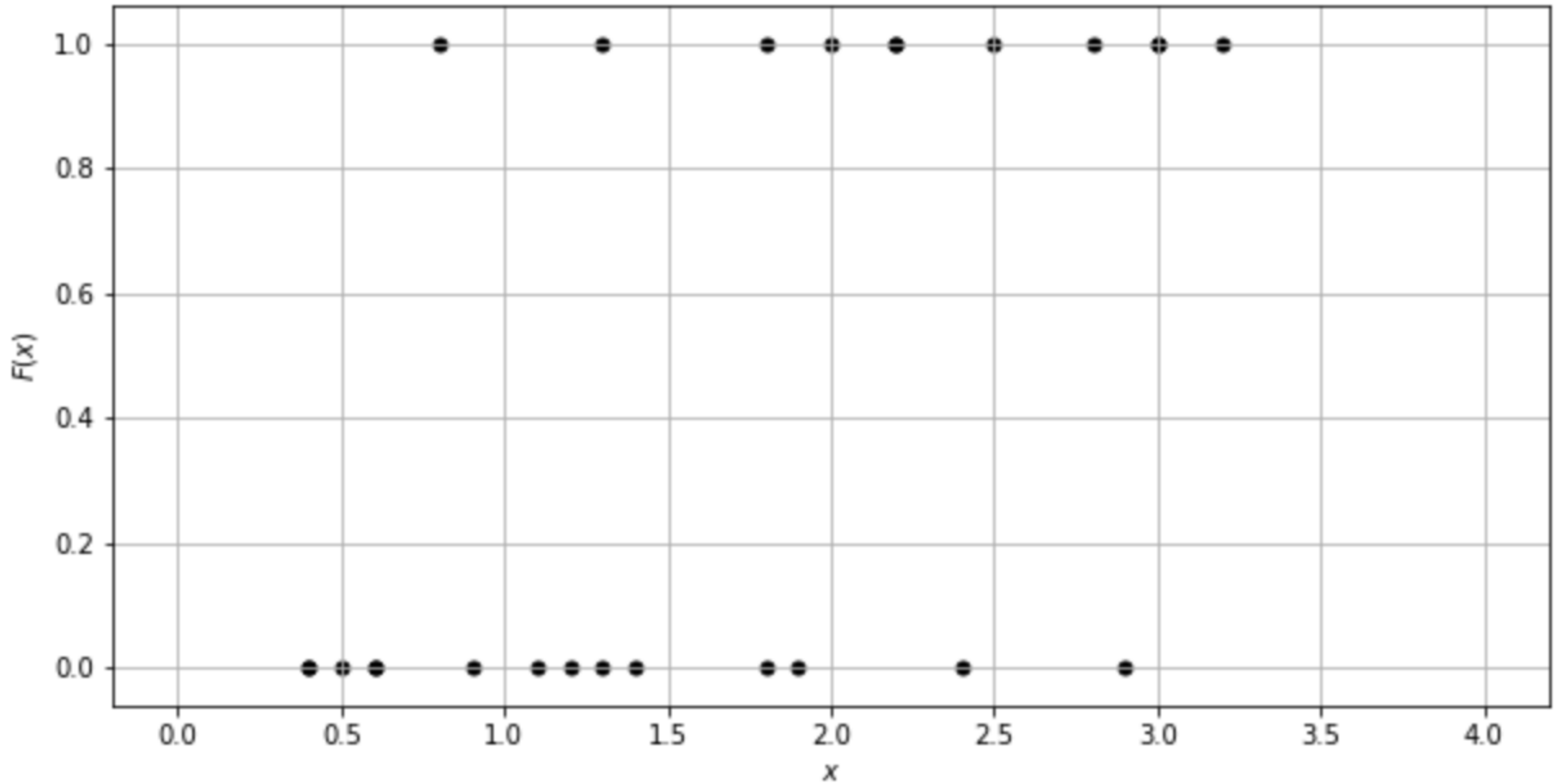
$$\log \; L(b_0, b_1) \; = \; \sum_{i=1}^{n} y_i \, (b_0 + b_1 \, x_i) \; - \; \sum_{i=1}^{n} \log(1 + e^{b_0+b_1 \, x_i})$$

log LIKELIHOOD FUNCTION

$$\log\ L(b_0, b_1) \ = \ \sum_{i=1}^{n} y_i\,(b_0 + b_1\, x_i) \ - \ \sum_{i=1}^{n} \log(1 + e^{b_0 + b_1\, x_i})$$

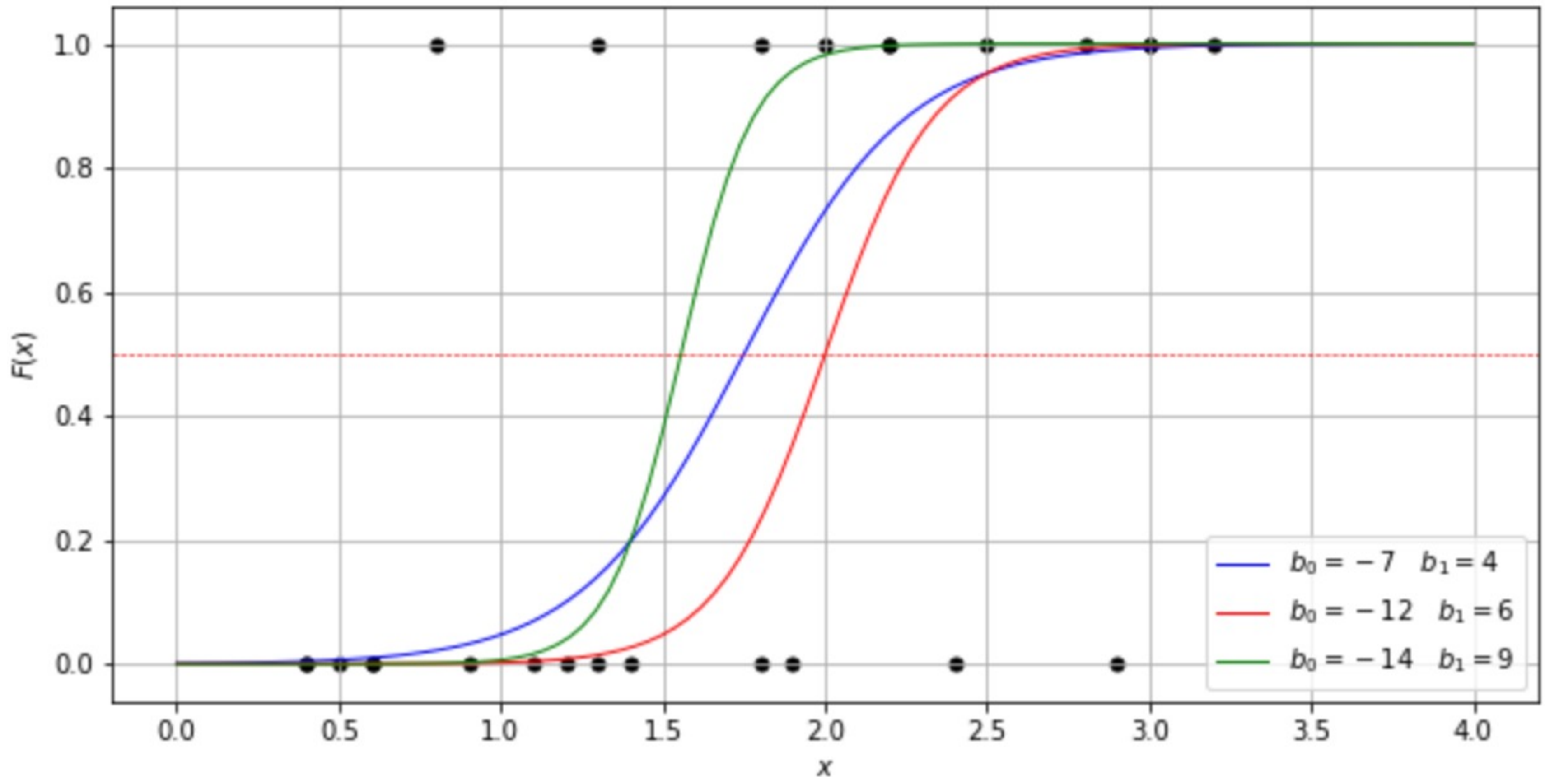
Finally, find b_0 and b_1 that maximize $\log L(b_0, b_1)$ using a numerical procedure (i.e., gradient search)

Find b_0 and b_1 that maximize $\log L$



Find b_0 and b_1 that maximize $\log L$

.



Logistic regression

What is the meaning of β_1?

Linear regression

What is the meaning of β_1?

In linear regression, β_1 is the slope.

It means that if X increases one unit

then Y changes β_1 units

LOGISTIC REGRESSION

What is the meaning of β_1?

- In logistic regression the meaning of β_1 is related to the Odds of Y=1

- [Odds of category Y=1] = $\dfrac{\pi}{1-\pi}$

What is β_1 ?

Since π_i changes with x_i, then the odds changes with x_i

	probability	*odds for category 1*
when $X = x_1$	*$P[Y=1] = \pi_1$*	$O_1 = \frac{\pi_1}{1 - \pi_1}$
when $X = x_2$	*$P[Y=1] = \pi_2$*	$O_2 = \frac{\pi_2}{1 - \pi_2}$

Logistic regression -parameters

P[Y = 1] $$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$ *is a function of* x

Logistic regression -parameters

.

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

$$\frac{1}{\pi} = 1 + e^{-\beta_0 - \beta_1 x}$$

Logistic regression -parameters

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

$$\frac{1}{\pi} = 1 + e^{-\beta_0 - \beta_1 x}$$

$$\frac{1}{\pi} - 1 = e^{-\beta_0 - \beta_1 x}$$

Logistic regression -parameters

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

$$\frac{1}{\pi} = 1 + e^{-\beta_0 - \beta_1 x}$$

$$\frac{1}{\pi} - 1 = e^{-\beta_0 - \beta_1 x}$$

$$\frac{1 - \pi}{\pi} = e^{-\beta_0 - \beta_1 x}$$

Logistic regression -parameters

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

$$\frac{1}{\pi} = 1 + e^{-\beta_0 - \beta_1 x}$$

$$\frac{1}{\pi} - 1 = e^{-\beta_0 - \beta_1 x}$$

$$\frac{1 - \pi}{\pi} = e^{-\beta_0 - \beta_1 x}$$

odds $\frac{\pi}{1 - \pi} = e^{\beta_0 + \beta_1 x}$ *is a function of* x

Logistic regression -parameters

$$\frac{\pi}{1-\pi} = e^{\beta_0 + \beta_1 x}$$

$$O = e^{\beta_0 + \beta_1 x}$$

the odds as a function of x

Logistic regression -parameters

$$\frac{\pi}{1-\pi} = e^{\beta_0 + \beta_1 x}$$

$$O = e^{\beta_0 + \beta_1 x}$$

the odds as a function of x

$$\ln O = \beta_0 + \beta_1 x$$

the log odds is a linear function of x

Logistic regression -parameters

Compare the odds,

when X changes from x_1 to x_2

$$O_1 = e^{\beta_0} e^{\beta_1 x_1}$$

$$O_2 = e^{\beta_0} e^{\beta_1 x_2}$$

Logistic regression -parameters

Compare the odds,

when X changes from x_1 to x_2

$$O_1 = e^{\beta_0} e^{\beta_1 x_1}$$

$$O_2 = e^{\beta_0} e^{\beta_1 x_2}$$

odds ratio

$$\frac{O_2}{O_1} = e^{\beta_1 (x_2 - x_1)}$$

Logistic regression -parameters

Compare the odds,

when X changes from x_1 to x_2

$$O_1 = e^{\beta_0} e^{\beta_1 x_1}$$

$$O_2 = e^{\beta_0} e^{\beta_1 x_2}$$

odds ratio →
$$\frac{O_2}{O_1} = e^{\beta_1 (x_2 - x_1)}$$

If $x_2 - x_1 = 1$
$$\frac{O_2}{O_1} = e^{\beta_1}$$

Meaning of β_1

if X increases one unit, then

$$\frac{O_2}{O_1} = e^{\beta_1}$$

- *the odds-ratio changes e^{β_1} units*

$$\ln\left(\frac{O_2}{O_1}\right) = \beta_1$$

- *the log odds changes β_1 units*

Logistic regression with sklearn

```
from sklearn.linear_model import LogisticRegression
```

```
model1 = LogisticRegression(solver="lbfgs", random_state=42)
model1.fit(X, y)
```

```
yhat = model1.predict(X)
```

yhat is the predicted category

```
y_proba = model1.predict_proba(X)
```

y_proba is the probability that y=1

Simple logistic regression Example

SIMPLE LOGISTIC REGRESSION - EXAMPLE

- File task.csv has data of 25 data analysts
- Each one was given the same amount of time to complete a data science project
- The data shows the analyst experience (in months)
- It also shows if the project was successfully completed (Y = 1) or not (Y = 0)
- It is of interest to predict if a new analyst is able to successfully complete such a project given his experience (in months)

SIMPLE LOGISTIC REGRESSION - EXAMPLE

- Predict the success of a data science project based on the experience of the analyst
- Interpret the estimated b_1
- Predict probability of success of an analyst with 22 months of experience
- Plot the fitted logistic curve along with the scatterplot of the response and the predictor
- Find the error rate on the entire data set
- Use holdout cross validation (70% of train set) to estimate the test error rate.

SIMPLE LOGISTIC REGRESSION - EXAMPLE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

SIMPLE LOGISTIC REGRESSION - EXAMPLE

```
df = pd.read_csv('task.csv')
df[:5]
```

	Experience	Success
0	14	0
1	29	0
2	6	0
3	25	1
4	18	1

predict **Success** using
Experience as predictor

SIMPLE LOGISTIC REGRESSION - EXAMPLE

```
df = pd.read_csv('task.csv')
df[:5]
```

	Experience	Success
0	14	0
1	29	0
2	6	0
3	25	1
4	18	1

```
df.shape
```

```
(25, 2)
```

```
y = df.Success
X = df.drop('Success',axis = 1)
X
```

	Experience
0	14
1	29
2	6
3	25
4	18
5	4

SIMPLE LOGISTIC REGRESSION MODEL

```
model = LogisticRegression(solver='lbfgs')
model.fit(X,y);
```

```
# coefficient b0
b0 = (model.intercept_)
print(b0)
```

```
[-3.04760123]
```

```
# coefficient b1
b1 = model.coef_
print(b1)
```

```
[[0.1608086]]
```

SIMPLE LOGISTIC REGRESSION - PREDICTION

```
# predict probability of a success

model.predict_proba(newval)
```

```
array([[0.37984928, 0.62015072]])
```

```
# probab of success is 0.62
```

P[Y=1]

```
# predict outcome (0:failure, or 1:success)

model.predict(newval)
```

```
array([1])
```

```
# model predicts a success
```

[Y=1]

SIMPLE LOGISTIC REGRESSION – INTERPRET b_1

Find e^{β_1}

```
odds_ratio = np.exp(b1)
odds_ratio
```

```
array([[1.17446016]])
```

```
# Odds of success increase by 17.44% with each additional month of experience
```

SIMPLE LOGISTIC REGRESSION – PLOT LOGISTIC CURVE

```
df2 = pd.DataFrame()
xaxis = list(range(32))
df2['xaxis'] = xaxis
df2[:5]
```

	xaxis
0	0
1	1
2	2
3	3
4	4

```
y_proba = model.predict_proba(df2)[:,1]
```

get probability of success only

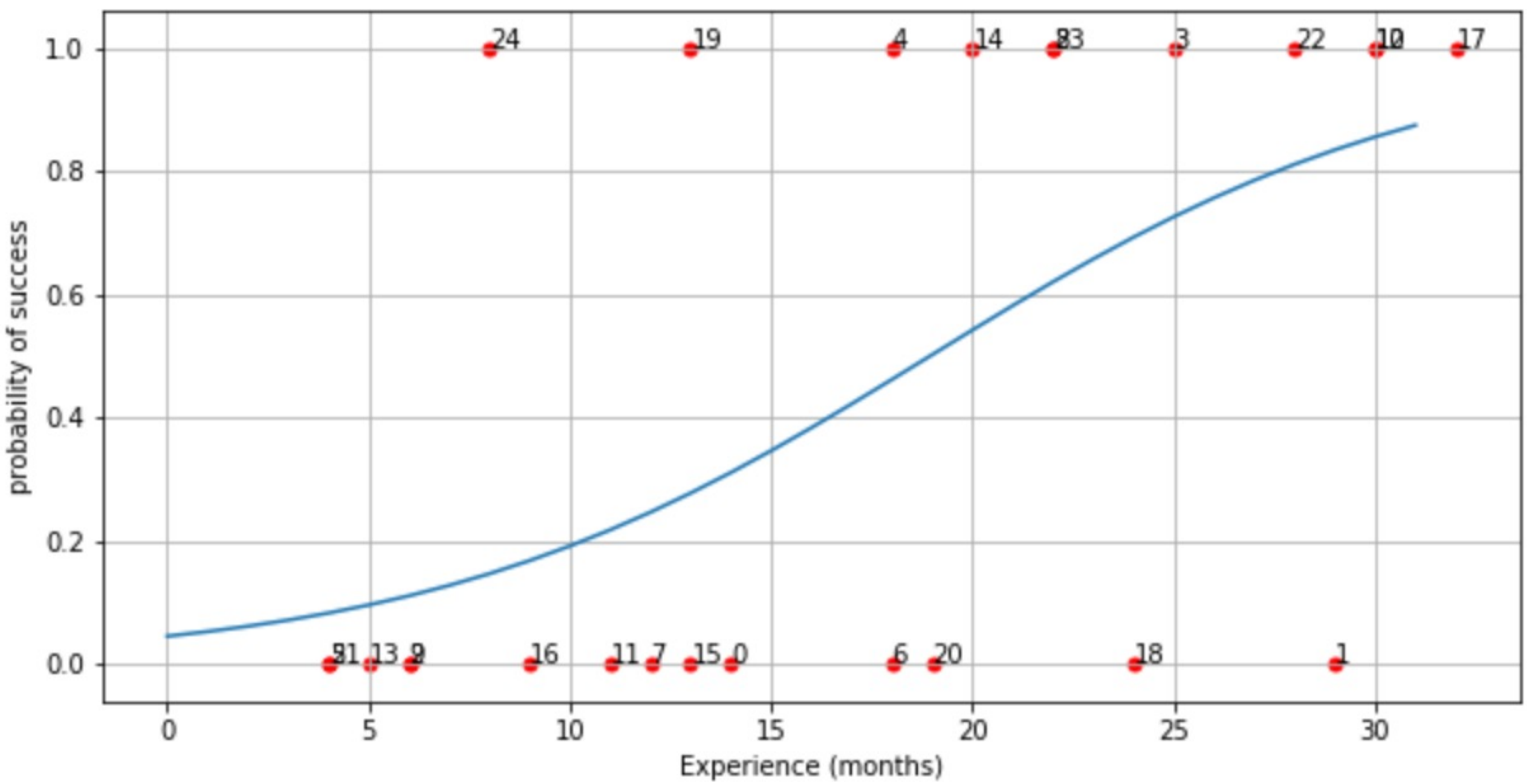
```
df2['y_proba'] = y_proba
df2[:5]
```

	xaxis	y_proba
0	0	0.045321
1	1	0.052810
2	2	0.061457
3	3	0.071414
4	4	0.082840

```
# df2 has the logistic curve coordinates
```

```
plt.figure(figsize=(10,5))
plt.scatter(X,y,s=25,c='r')
plt.plot(xaxis,y_proba)
```

SIMPLE LOGISTIC REGRESSION – SCATTERPLOT AND LOGISTIC CURVE



SIMPLE LOGISTIC REGRESSION – PREDICT CATEGORIES

```
yhat = model.predict(X)
```

```
df2 = pd.DataFrame()
df2['Success'] = y
df2['prediction'] = yhat
df2[:5]
```

	Success	prediction
0	0	0
1	0	1
2	0	0
3	1	1
4	1	0

```
# see prediction errors only
df2.loc[df2.Success!=df2.prediction]
```

	Success	prediction
1	0	1
4	1	0
18	0	1
19	1	0
20	0	1
24	1	0

SIMPLE LOGISTIC REGRESSION – CROSSTABULATION FOR PREDICTIONS

.

```
pd.crosstab(df2.prediction,df2.Success)
```

Success	0	1
prediction		
0	11	3
1	3	8

```
# error rate
```

```
6/25
```

```
0.24
```

HOLDOUT CROSS VALIDATION

```
X_train, X_test, y_train, y_test = train_test_split(X,y,stratify=y,test_size=0.3,
                                                    shuffle = True,random_state=1)
```

```
X_test.shape
```

```
(8, 1)
```

```
model = LogisticRegression(solver="lbfgs").fit(X_train,y_train)
yhat = model.predict(X_test)
```

```
df3 = pd.DataFrame()
df3['Success'] = Y_test
df3['prediction'] = yhat.astype(int)
df3
```

HOLDOUT CROSS VALIDATION

```
df3
```

	Success	prediction
14	1	1
13	0	0
18	0	1
24	1	0
7	0	0
8	1	1
12	1	1
15	0	0

```
# test set prediction errors
df3.loc[df3.Success!=df3.prediction]
```

	Success	prediction
18	0	1
24	1	0

```
pd.crosstab(df3.prediction,df3.Success)
```

Success	0	1
prediction		
0	3	1
1	1	3

test error rate = 2/8 = 0.25

Multiple logistic Regression

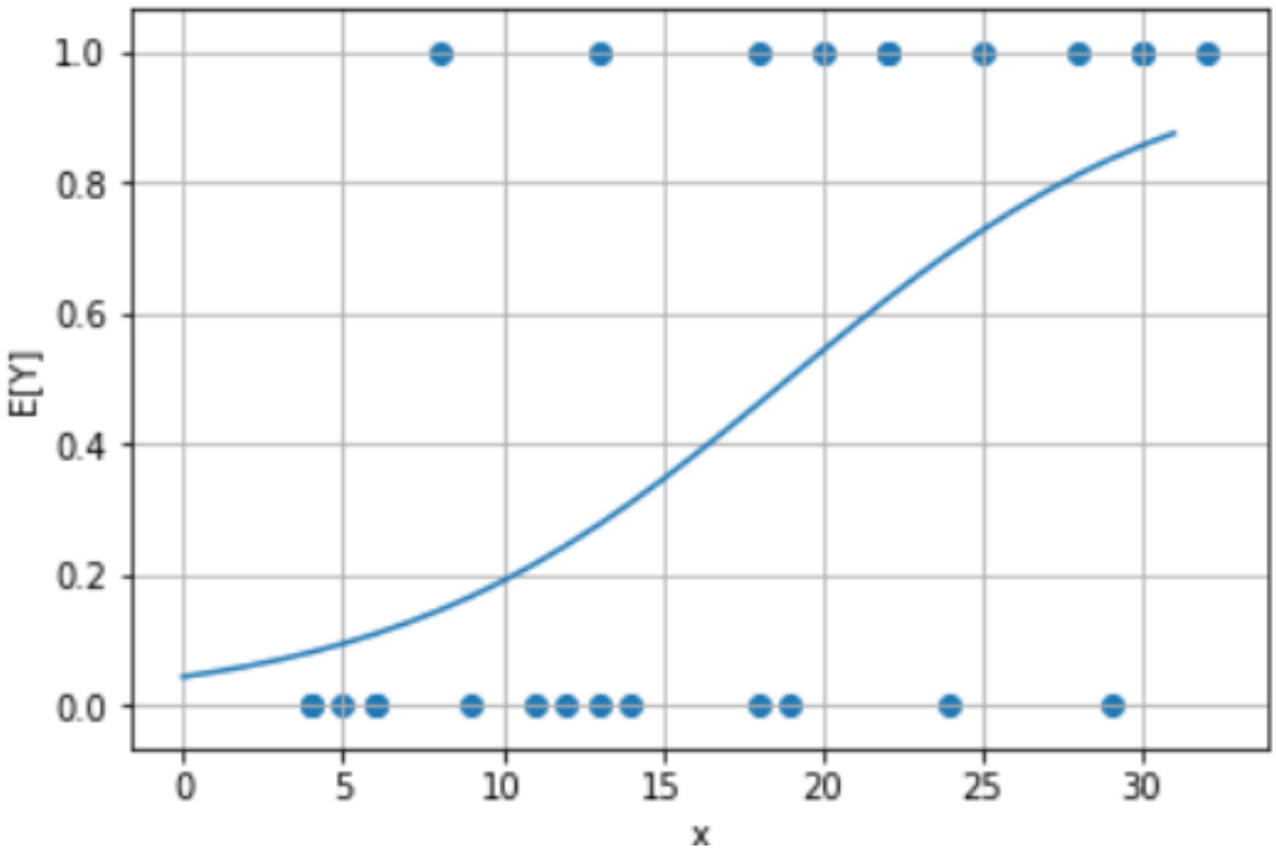
Simple Logistic regression

As x increases, π varies along the logistic cdf

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

where x is the predictor (feature)

Simple Logistic regression function



Multiple Logistic regression **– TWO PREDICTORS**

π *varies along the surface*

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_1 - \beta_2 x_2}}$$

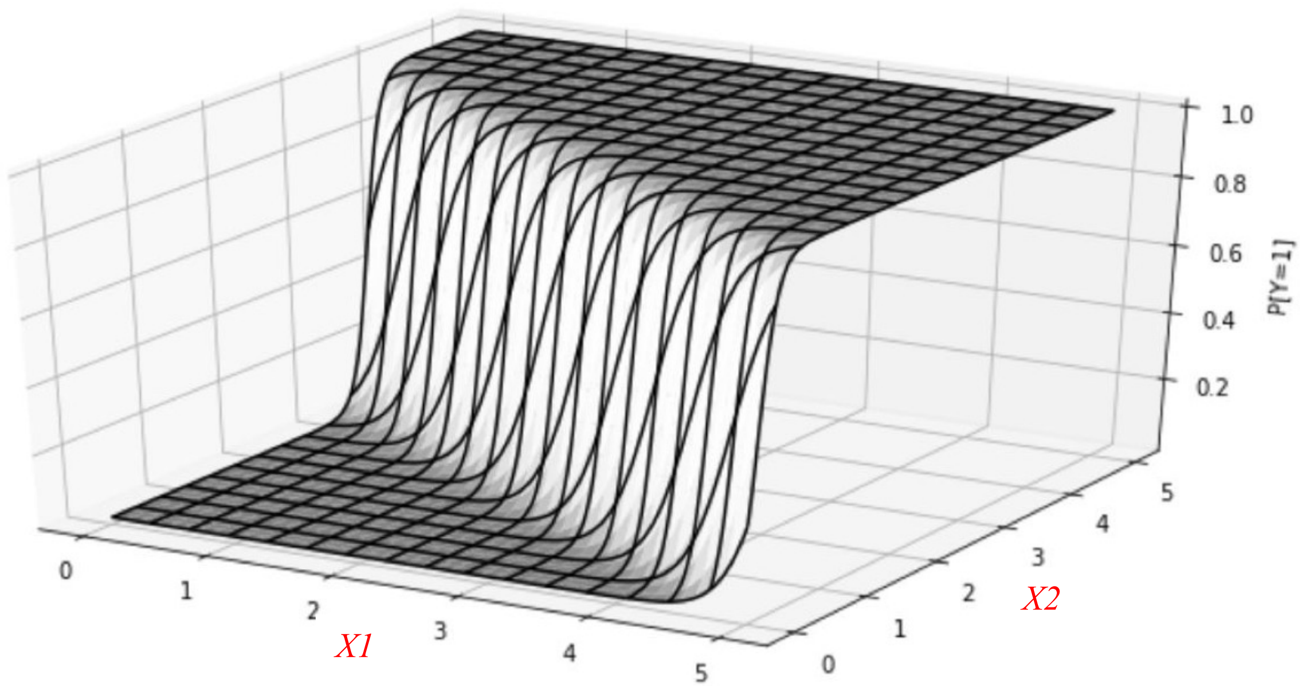
where x_1 *and* x_2 *are the predictors*

π *changes as* x_1 *and* x_2 *change*

Multiple Logistic regression **– Two predictors**

π varies along the surface

Y	X1	X2
0	2.5	1.5
0	1.7	0.6
1	2.3	1.1
0	0.8	2.5
1	1.1	0.9
1	1.5	1.9



Multiple Logistic regression – ***n* predictors**

π varies along the surface

$$\pi = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_1 \ldots - \beta_n x_n}}$$

where x_1 , x_2 , … x_n are numerical and/or categorical

Multinomial Regression models
are used with classification problems
when the response has
more than two categories

Logistic regression

Example
Cancer data

Cancer Data - EXAMPLE

- The Cancer data from *sklearn* contains data from 569 patients.
- It includes 30 lab measurements associated with breast cancer tumors. These are the predictors.
- Some patients have cancer but not all. The target np.array identifies these patients.
- Build a Logistic Regression model to predict whether new patients have cancer.
- Compare predictions not scaling or scaling the lab measurements
- find the test accuracy rate using Holdout and K-fold Cross validation

Cancer Data - NOTES

- Logistic regression does not have hyperparameters
- Validation sets are not needed
- All regression models may improve by scaling the data

Cancer Data

30 measurements

Y	<-------	----------	-----------	-------	**average**	**values**	----------	----------	----------	---------->	<-------	--------	-----------	--------	**worst**	**values**	----------	----------	----------
out	radius	texture	perimeter	area	smoothness	compactness	concavity	concave p	symmetry	fractal_dir	radius	texture	perimeter	area	smoothness	compactness	concavity	concave p	symmetry
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	15.47	23.75	103.4	741.6	0.1791	0.5249	0.5355	0.1741	0.3985
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	22.88	27.66	153.2	1606	0.1442	0.2576	0.3784	0.1932	0.3063
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	17.06	28.14	110.6	897	0.1654	0.3682	0.2678	0.1556	0.3196
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	15.49	30.73	106.2	739.3	0.1703	0.5401	0.539	0.206	0.4378
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	15.09	40.68	97.65	711.4	0.1853	1.058	1.105	0.221	0.4366
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	19.19	33.88	123.8	1150	0.1181	0.1551	0.1459	0.09975	0.2948
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	20.42	27.28	136.5	1299	0.1396	0.5609	0.3965	0.181	0.3792
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	20.96	29.94	151.7	1332	0.1037	0.3903	0.3639	0.1767	0.3176
M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	16.84	27.66	112	876.5	0.1131	0.1924	0.2322	0.1119	0.2809
M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	15.03	32.01	108.8	697.7	0.1651	0.7725	0.6943	0.2208	0.3596
M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	17.46	37.13	124.1	943.2	0.1678	0.6577	0.7026	0.1712	0.4218
M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	19.07	30.88	123.4	1138	0.1464	0.1871	0.2914	0.1609	0.3029
M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	20.96	31.48	136.8	1315	0.1789	0.4233	0.4784	0.2073	0.3706
M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	27.32	30.88	186.8	2398	0.1512	0.315	0.5372	0.2388	0.2768
B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	15.11	19.26	99.7	711.2	0.144	0.1773	0.239	0.1288	0.2977
B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	14.5	20.49	96.09	630.5	0.1312	0.2776	0.189	0.07283	0.3184
B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	10.23	15.66	65.13	314.9	0.1324	0.1148	0.08867	0.06227	0.245
M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	18.07	19.08	125.1	980.9	0.139	0.5954	0.6305	0.2393	0.4667
M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278	29.17	35.59	188	2615	0.1401	0.26	0.3155	0.2009	0.2822

Logistic Regression - EXAMPLE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

from sklearn.datasets import load_breast_cancer
```

```
cancer = load_breast_cancer()
cancer.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names',
           'DESCR', 'feature_names', 'filename', 'data_module'])
```

Cancer Data

X

Y	<-------	-----------	-----------	-------	average	values	-----------	-----------	-----------	---------->	<-------	--------	-----------	--------	worst	values	-----------	-----------	-----------
out	radius	texture	perimeter	area	smoothness	compactness	concavity	concave p	symmetry	fractal_dir	radius	texture	perimeter	area	smoothness	compactness	concavity	concave p	symmetry
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	15.47	23.75	103.4	741.6	0.1791	0.5249	0.5355	0.1741	0.3985
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	22.88	27.66	153.2	1606	0.1442	0.2576	0.3784	0.1932	0.3063
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	17.06	28.14	110.6	897	0.1654	0.3682	0.2678	0.1556	0.3196
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	15.49	30.73	106.2	739.3	0.1703	0.5401	0.539	0.206	0.4378
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	15.09	40.68	97.65	711.4	0.1853	1.058	1.105	0.221	0.4366
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	19.19	33.88	123.8	1150	0.1181	0.1551	0.1459	0.09975	0.2948
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	20.42	27.28	136.5	1299	0.1396	0.5609	0.3965	0.181	0.3792
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	20.96	29.94	151.7	1332	0.1037	0.3903	0.3639	0.1767	0.3176
M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	16.84	27.66	112	876.5	0.1131	0.1924	0.2322	0.1119	0.2809
M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	15.03	32.01	108.8	697.7	0.1651	0.7725	0.6943	0.2208	0.3596
M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	17.46	37.13	124.1	943.2	0.1678	0.6577	0.7026	0.1712	0.4218
M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	19.07	30.88	123.4	1138	0.1464	0.1871	0.2914	0.1609	0.3029
M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	20.96	31.48	136.8	1315	0.1789	0.4233	0.4784	0.2073	0.3706
M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	27.32	30.88	186.8	2398	0.1512	0.315	0.5372	0.2388	0.2768
B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	15.11	19.26	99.7	711.2	0.144	0.1773	0.239	0.1288	0.2977
B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	14.5	20.49	96.09	630.5	0.1312	0.2776	0.189	0.07283	0.3184
B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	10.23	15.66	65.13	314.9	0.1324	0.1148	0.08867	0.06227	0.245
M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	18.07	19.08	125.1	980.9	0.139	0.5954	0.6305	0.2393	0.4667
M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278	29.17	35.59	188	2615	0.1401	0.26	0.3155	0.2009	0.2822

Logistic Regression - EXAMPLE

```
y = cancer.target
X = cancer.data
```

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(solver = 'lbfgs')
model.fit(X,y)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning:
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Logistic Regression - EXAMPLE

```
model = LogisticRegression(solver = 'lbfgs',max_iter = 10000).fit(X,y)
```

```
model.coef_
```

```
array([[ 1.04416478,  0.17898383, -0.27990407,  0.02266274, -0.17006924,
        -0.23359807, -0.53158508, -0.28025859, -0.25813621, -0.03334163,
        -0.0773966 ,  1.26133937,  0.11350693, -0.10833712, -0.0234327 ,
         0.04709577, -0.05336947, -0.03659769, -0.04004551,  0.01069554,
         0.16582275, -0.43557953, -0.10345568, -0.01409449, -0.33754995,
        -0.73599929, -1.42645241, -0.57099767, -0.72018236, -0.10256733]])
```

```
model.intercept_
```

```
array([27.71390837])
```

Holdout cross validation

SCALING

All predictors with values in the same range/scale

Benefits

- Improve prediction accuracy
- Reduce computer time

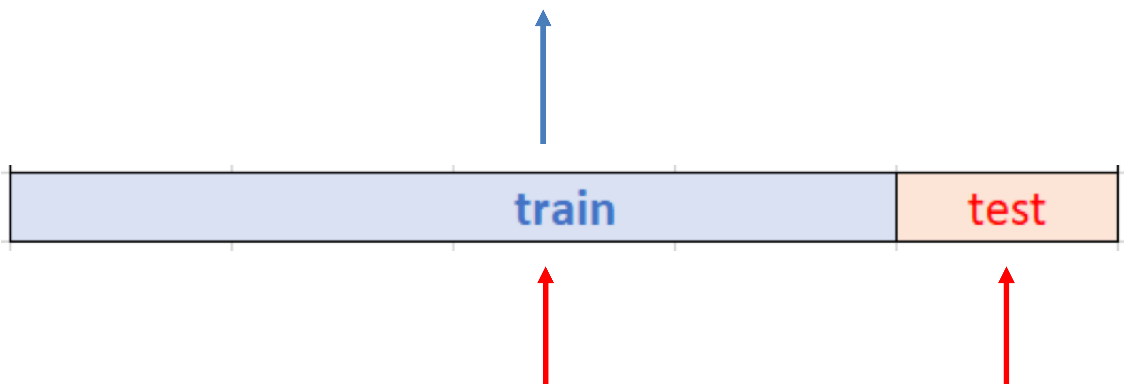
SCALING

Sklearn options for scaling X

- MinMaxScaler( ) to make the values of all features in [0,1]
  by substracting the column Min
  and dividing by the column range
- StandardScaler( ) to make the values of all features
  (with mean 0 and, standard deviation 1)
  by substracting the column mean
  and dividing by the column std. deviation

Logistic Regression – Holdout Cross Validation with scaling

1. Find mean and standard deviation from each column in the train set



2. Scale train set and test set
(using the mean and stardard deviation found from the train set)

Logistic Regression – Holdout Cross Validation

```
y = cancer.target
X = cancer.data

print(X.shape,y.shape)
```

```
(569, 30) (569,)
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,stratify=y,
                                                 random_state=66)
```

```
print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

```
(426, 30) (143, 30) (426,) (143,)
```

```
426/569
```

```
0.7486818980667839
```

```
# train set is about 75% of dataset (default)
```

Logistic Regression – Split dataset for Holdout CV

.

```
y = cancer.target
X = cancer.data

print(X.shape,y.shape)
```

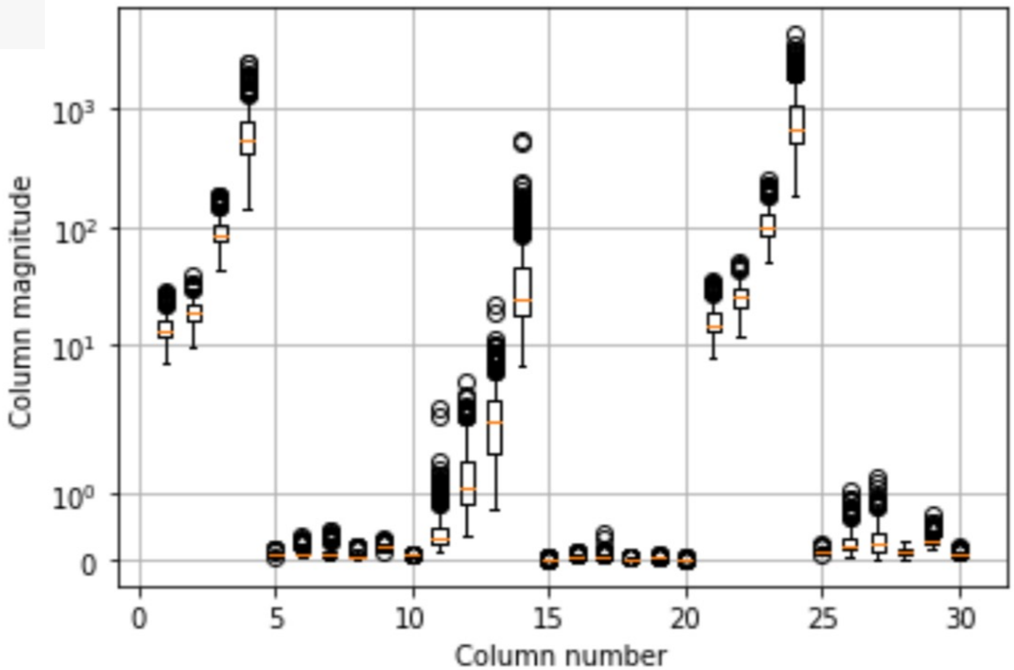
```
(569, 30) (569,)
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,stratify=y,
                                                  random_state=66)
```

Logistic Regression – predictors Ranges

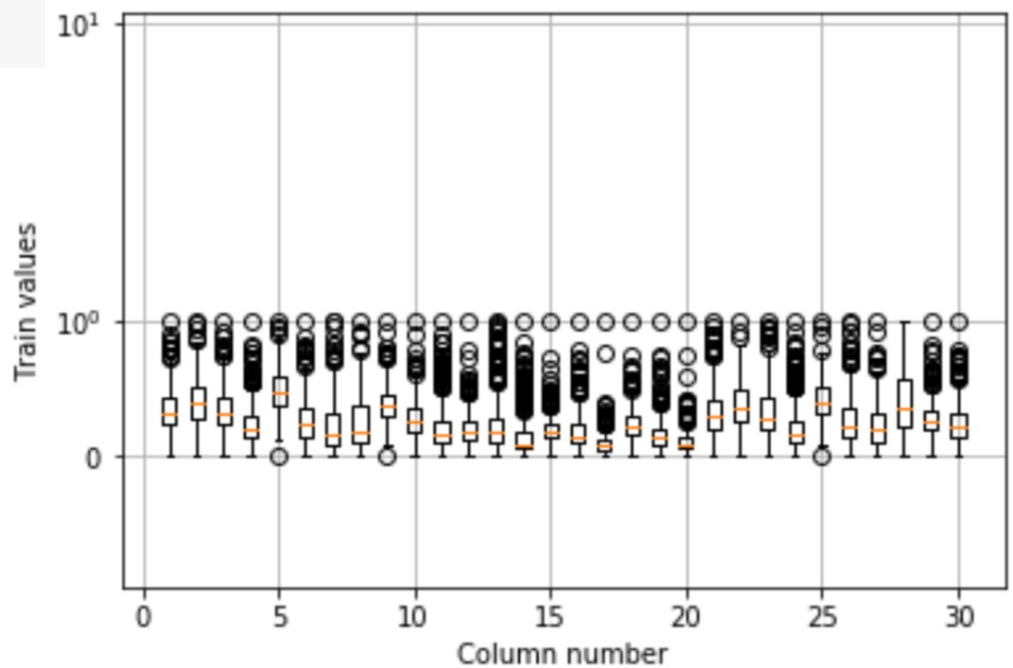
.

```
# range of predictors (not scaled)
plt.boxplot(cancer.data, manage_ticks=False)
plt.yscale("symlog")
plt.xlabel("Column number")
plt.ylabel("Column magnitude")
plt.grid();
```



Logistic Regression – SCALED predictors in the train set

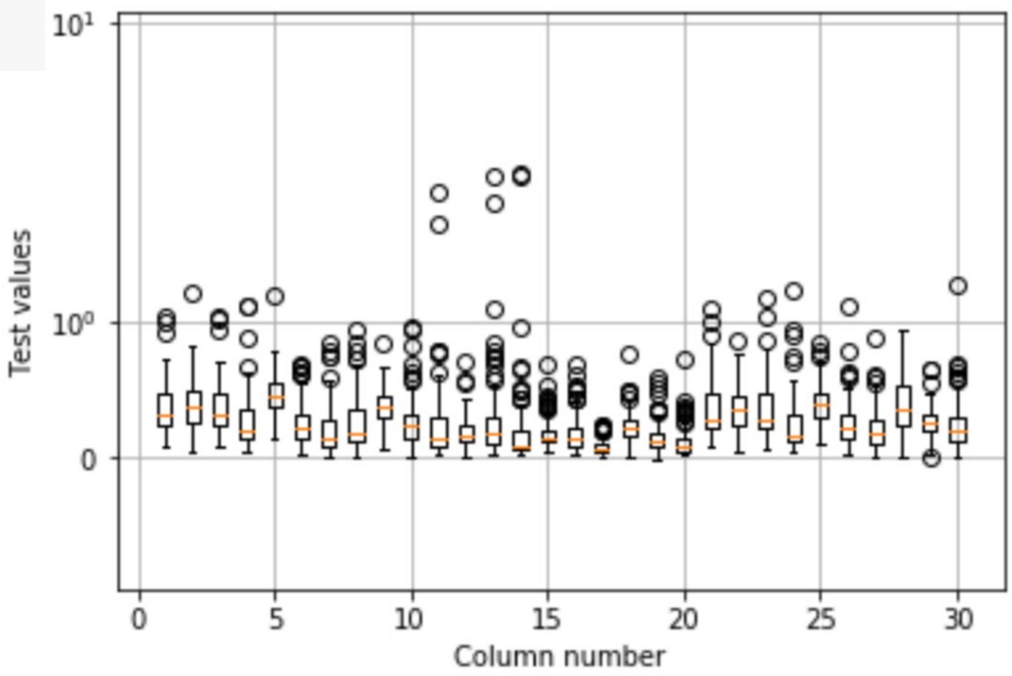
```
plt.boxplot(X_train_scaled, manage_ticks=False)
plt.yscale("symlog")
plt.xlabel("Column number")
plt.ylim(-1,11)
plt.ylabel("Train values")
plt.grid();
```



Logistic Regression – SCALED predictors in the test set

.

```
plt.boxplot(X_test_scaled, manage_ticks=False)
plt.yscale("symlog")
plt.xlabel("Column number")
plt.ylim(-1,11)
plt.ylabel("Test values")
plt.grid();
```



Logistic Regression – Holdout Cross Validation

.

```
# Compare test accuracy rate without and with scaling
```

not
scaling
the data

```
model = LogisticRegression(solver = 'lbfgs',max_iter=10000)
model.fit(X_train,y_train)
yhat = model.predict(X_test)
model.score(X_test,y_test)
```

```
0.9440559440559441
```

scaling
the data

```
model = LogisticRegression(solver = 'lbfgs')
model.fit(X_train_scaled,y_train)
yhat = model.predict(X_test_scaled)
model.score(X_test_scaled,y_test)
```

```
0.972027972027972
```

Holdout Cross Validation – scaling-

.

Logistic Regression

```
model = LogisticRegression(solver = 'lbfgs')
model.fit(X_train_scaled,y_train)
yhat = model.predict(X_test_scaled)
model.score(X_test_scaled,y_test)
```

```
0.972027972027972
```

KNN (K=2)

```
scaler = MinMaxScaler()
scaler.fit(X);

X_scaled = scaler.transform(X)
X_test_scaled = scaler.transform(X_test)
```

```
model2 = KNeighborsClassifier(n_neighbors=2)
model2.fit(X_scaled, y);
```

```
model2.score(X_test_scaled,y_test)
```

```
0.9230769230769231
```

K-fold cross validation

Logistic Regression – Stratified K-Fold Cross validation (No scaling)

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
```

← for classification problems

```
kfold = StratifiedKFold(n_splits = 5,shuffle = True,random_state=1)
```

```
model1 = LogisticRegression(solver = 'lbfgs',max_iter = 10000)
scores = cross_val_score(model1,X,y,cv=kfold)
scores
```

← Use all data set X,y

```
array([0.94736842, 0.94736842, 0.93859649, 0.95614035, 0.96460177])
```

```
scores.mean()
```

```
0.9508150908244062
```

k-fold Cross Validation with scaling

.

fit scaler to train set 1. Then scale train set 1 and test set 1

fit scaler to train set 2. Then scale train set 2 and test set 2

fit scaler to train set 3. Then scale train set 3 and test set 3

…

…

fit scaler to train set k. Then scale train set k and test set k

- At each fold fit the scaler to the corresponding train set
- Then scale the train and test sets for that fold

Logistic Regression – Stratified K-Fold Cross validation (Scaling)

```
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
```

```
kfold = StratifiedKFold(n_splits = 5,shuffle = True,random_state=1)
```

```
scaler = MinMaxScaler()
model1 = LogisticRegression(solver = 'lbfgs')
pipe1 = Pipeline([('transformer1', scaler), ('estimator1', model1)])
scores = cross_val_score(pipe1,X,y,cv=kfold)
scores
```

```
array([0.94736842, 0.98245614, 0.96491228, 0.96491228, 0.95575221])
```

```
scores.mean()
```

```
0.9630802670392795
```

k-Fold Cross Validation – scaling- Logistic Regression vs KNN with best K

```
scaler = MinMaxScaler()
model1 = LogisticRegression(solver = 'lbfgs')
pipe1 = Pipeline([('transformer1', scaler), ('estimator1', model1)])
scores = cross_val_score(pipe1,X,y,cv=kfold)
scores
```

```
array([0.97391304, 0.97391304, 0.97345133, 0.96460177, 0.97345133])
```

```
scores.mean()
```

```
0.9718661023470565
```

```
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(X_train_scaled, y_train);
```

```
model2.score(X_test_scaled,y_test)
```

```
0.9440559440559441
```

Review

LIBRARIES

.

```
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

from sklearn.model_selection import cross_val_score
```

HOLDOUT CROSS VALIDATION

```
y = df.response
X = df.drop(['response'],axis=1,inplace=True)
```

split

```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size = 0.75,
                                                 stratify=y,
                                                 random_state=1)
```

scale

```
scaler = MinMaxScaler()
scaler.fit(X_train);
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

train and test the model

```
model1 = LogisticRegression(solver = 'lbfgs')
model1.fit(X_train_scaled,y_train)
yhat = model1.predict(X_test_scaled)
model1.score(X_test_scaled,y_test)
```

K-Fold CROSS VALIDATION – SCALING WITHIN EACH FOLD

```
y = df.response
X = df.drop(['response'],axis=1,inplace=True)
```

```
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
```

```
kfold = StratifiedKFold(n_splits = 5,shuffle = True,random_state=1)
scaler = MinMaxScaler()

model1 = LogisticRegression(solver = 'lbfgs')
pipe1 = Pipeline([('transformer1', scaler), ('estimator1', model1)])
scores = cross_val_score(pipe1,X,y,cv=kfold)
scores
```

```
array([0.94736842, 0.98245614, 0.96491228, 0.96491228, 0.95575221])
```

```
scores.mean()
```

```
0.9630802670392795
```