Introduction to
Logistic Regression

Background

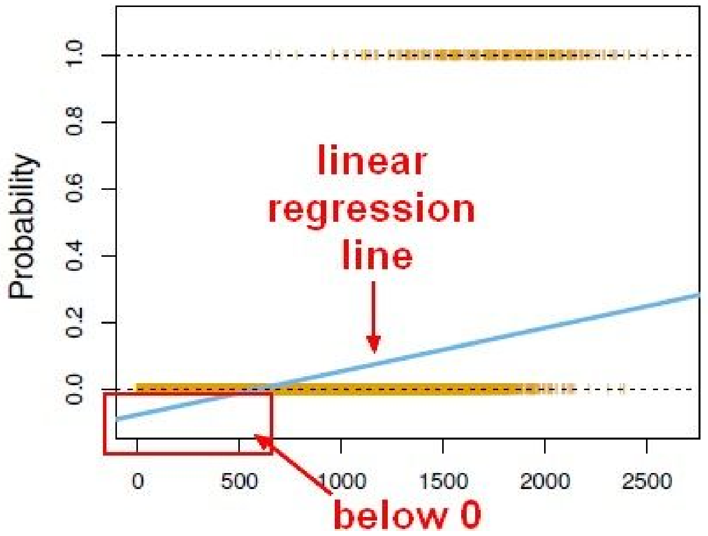
- We want to learn about Logistic Regression as a method for **Classification.**
- Some examples of classification problems:
  - Spam versus “Ham” emails
  - Loan Default (yes/no)
  - Disease Diagnosis
- Above were all examples of Binary Classification

Background

- So far we’ve only seen regression problems where we try to predict a continuous value.
- Although the name may be confusing at first, logistic regression allows us to solve classification problems, where we are trying to predict discrete categories.
- The convention for binary classification is to have two classes 0 and 1.

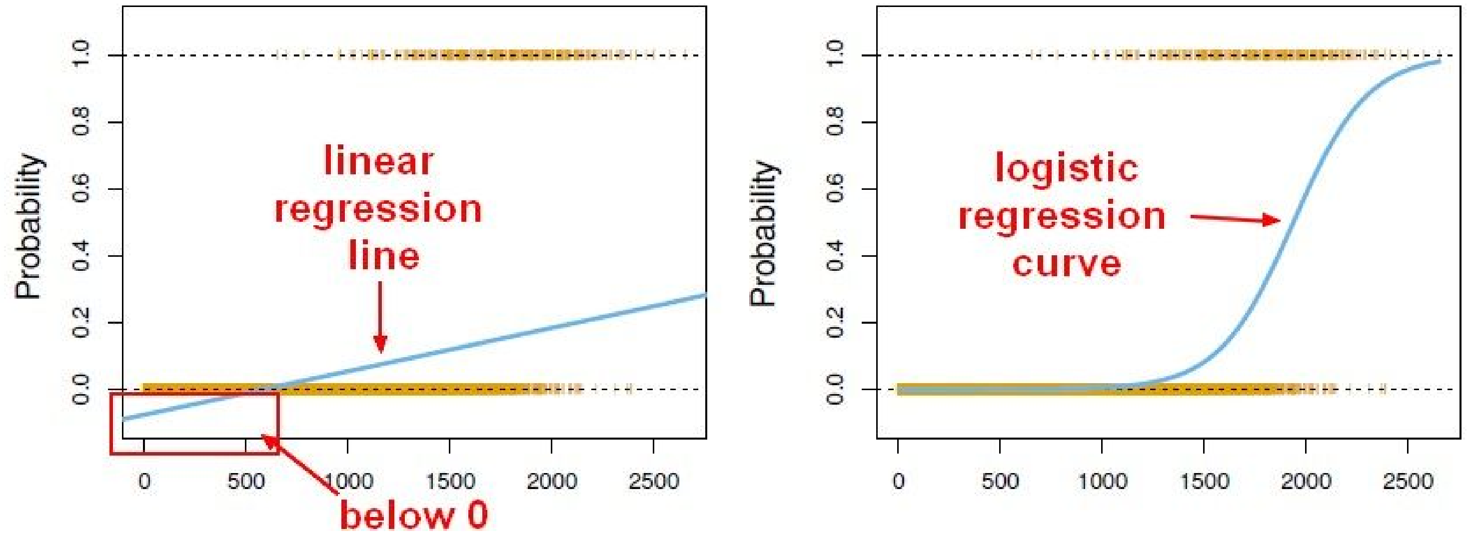
Background

- We can’t use a normal linear regression model on binary groups. It won’t lead to a good fit:



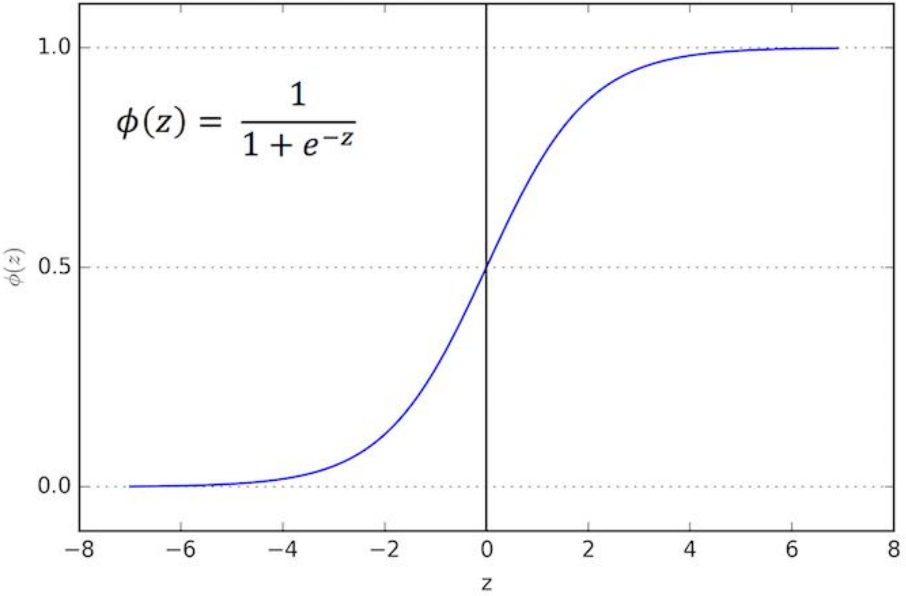
Background

- Instead we can transform our linear regression to a logistic regression curve.



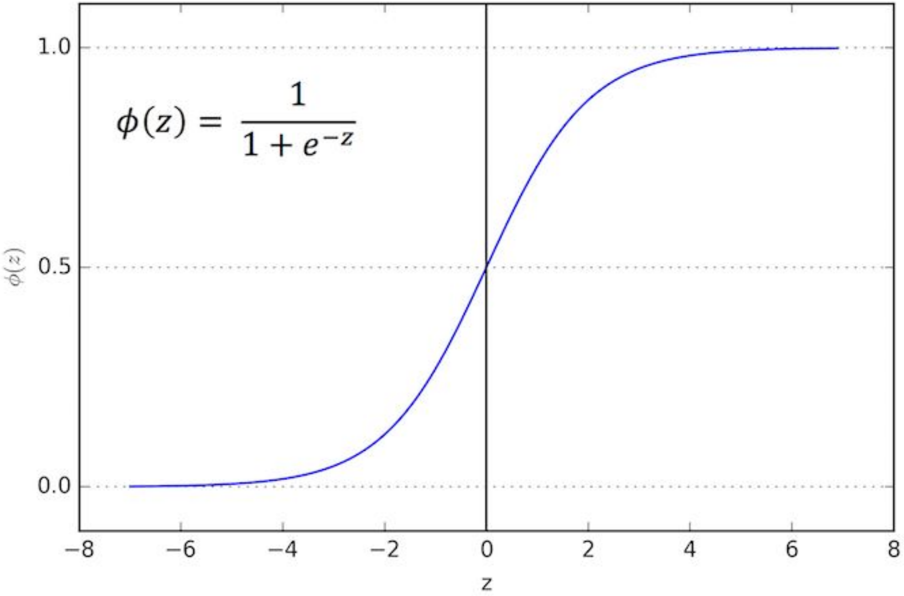
Sigmoid Function

- The Sigmoid (aka Logistic) Function takes in any value and outputs it to be between 0 and 1.



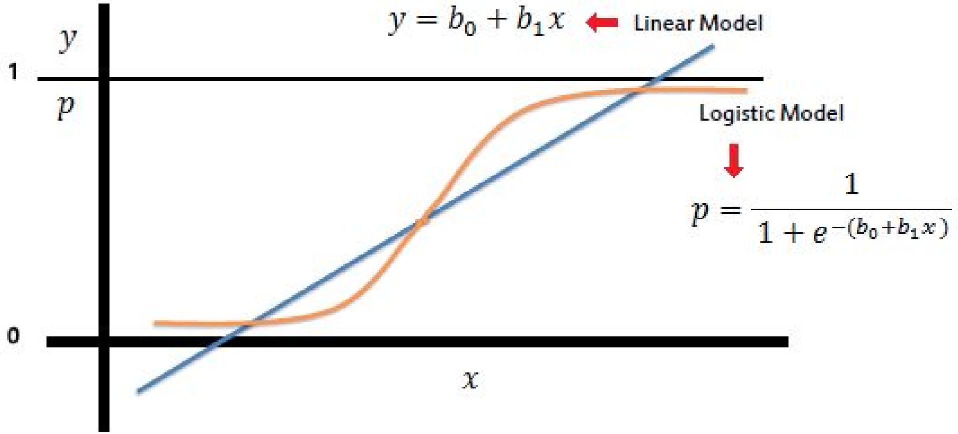
Sigmoid Function

- This means we can take our Linear Regression Solution and place it into the Sigmoid Function.



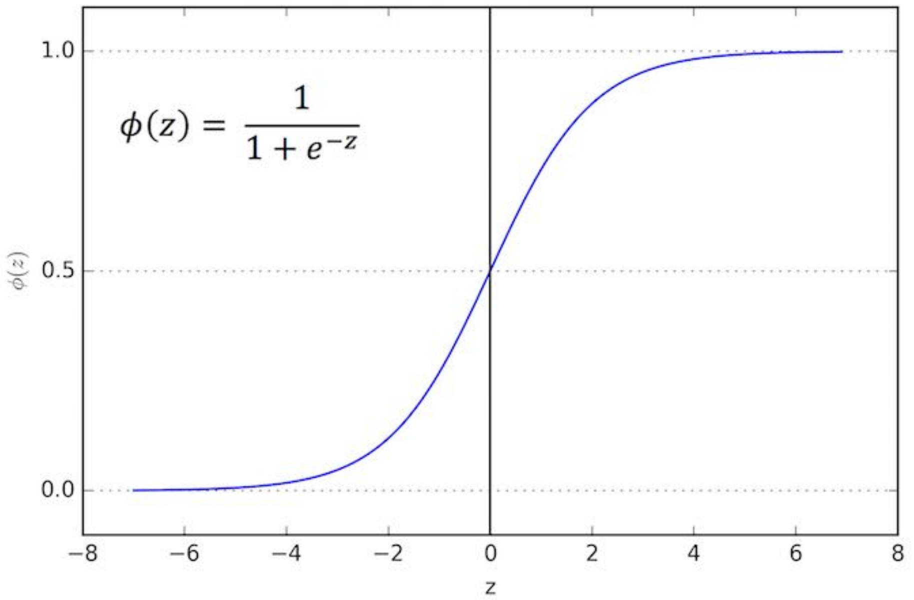
Sigmoid Function

- This means we can take our Linear Regression Solution and place it into the Sigmoid Function.



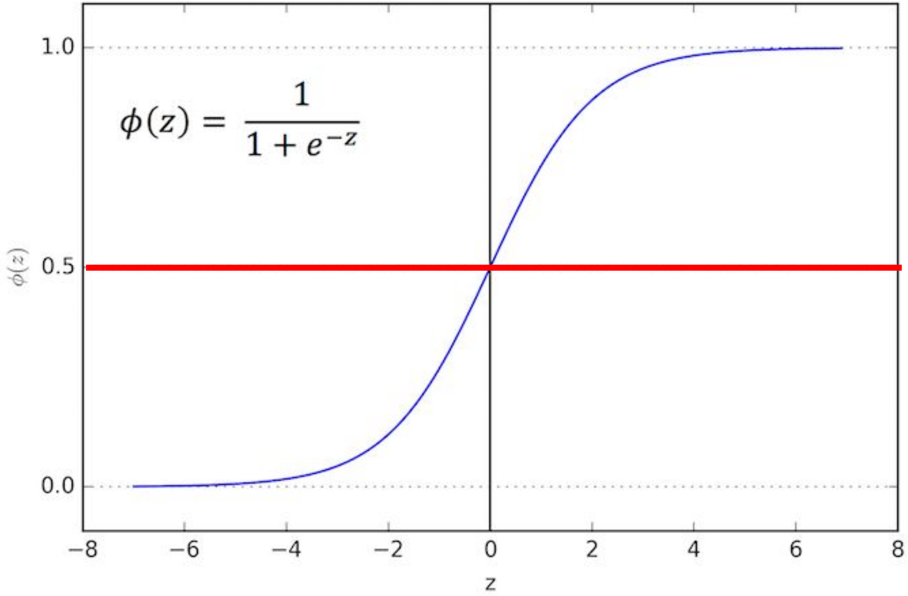
Sigmoid Function

- This results in a probability from 0 to 1 of belonging in the 1 class.



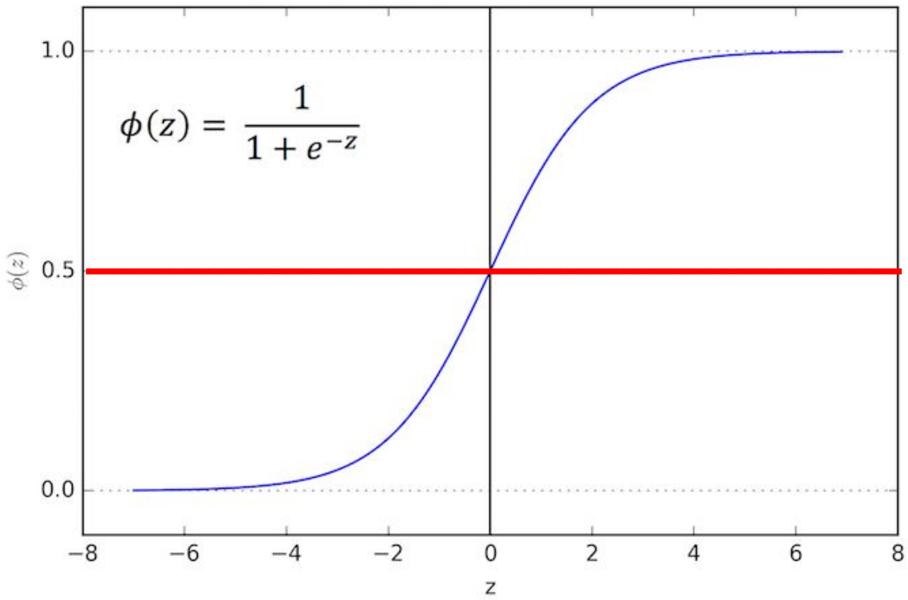
Sigmoid Function

- We can set a cutoff point at 0.5, anything below it results in class 0, anything above is class 1.



Review

- We use the logistic function to output a value ranging from 0 to 1. Based off of this probability we assign a class.



Model Evaluation

- After you train a logistic regression model on some training data, you will evaluate your model's performance on some test data.
- You can use a confusion matrix to evaluate classification models.

Model Evaluation

- We can use a confusion matrix to evaluate our model.
- For example, imagine testing for disease.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Example: Test for presence of disease
NO = negative test = False = 0
YES = positive test = True = 1

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Basic Terminology:

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy:

- Overall, how often is it **correct**?
- (TP + TN) / total = 150/165 = 0.91

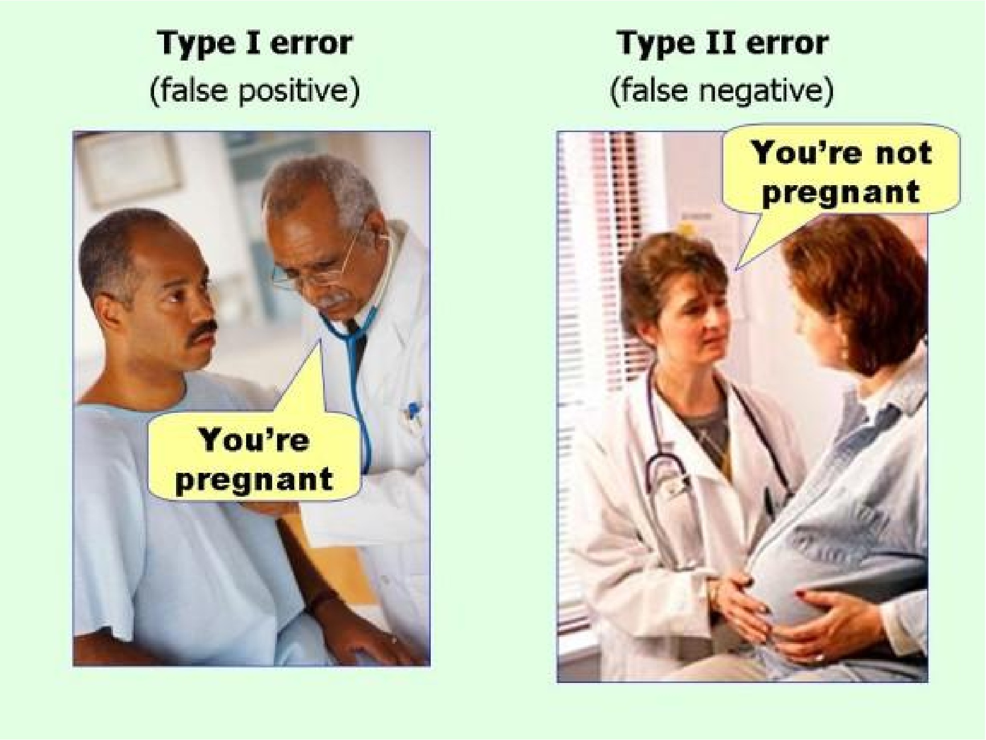
Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Misclassification Rate (Error Rate):

- Overall, how often is it **wrong**?
- (FP + FN) / total = 15/165 = 0.09

Confusion Matrix



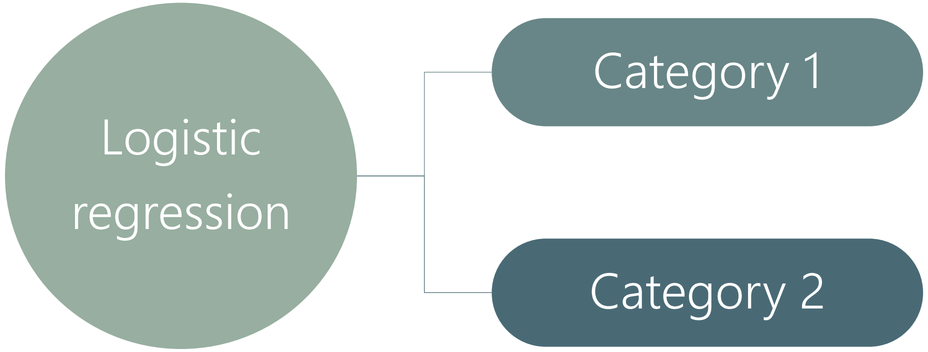
COURSE NOTES: LOGISTIC REGRESSION

Logistic regression vs Linear regression

Logistic regression implies that the possible outcomes are **not** numerical but rather categorical.

Examples for categories are:

- Yes / No
- Will buy / Won't Buy
- 1 / 0



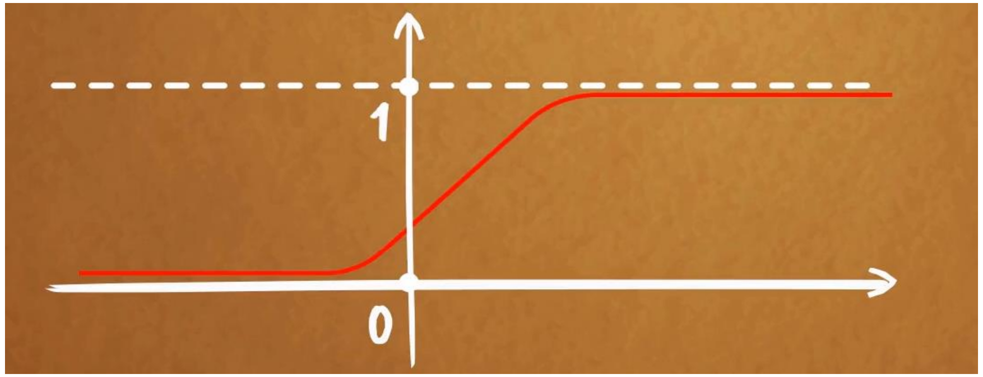
Linear regression model: $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + \varepsilon$

Logistic regression model: $p(X) = \dfrac{e^{(\beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k)}}{1 + e^{(\beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k)}}$

Logistic model

The logistic regression predicts the probability of an event occurring.

input → probability



Visual representation of a logistic function

Logistic regression model

Logistic regression model

$$\frac{p(X)}{1-p(X)} = e^{(\beta_0+\beta_1 X_1+\ldots+\beta_k X_k)}$$

The logistic regression model is not very useful in itself. The right-hand side of the model is an exponent which is very computationally inefficient and generally hard to grasp.

Logit regression model

When we talk about a 'logistic regression' what we usually mean is 'logit' regression – a variation of the model where we have taken the log of both sides.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \log\left(e^{(\beta_0+\beta_1 x+\cdots\beta_k x_k)}\right)$$

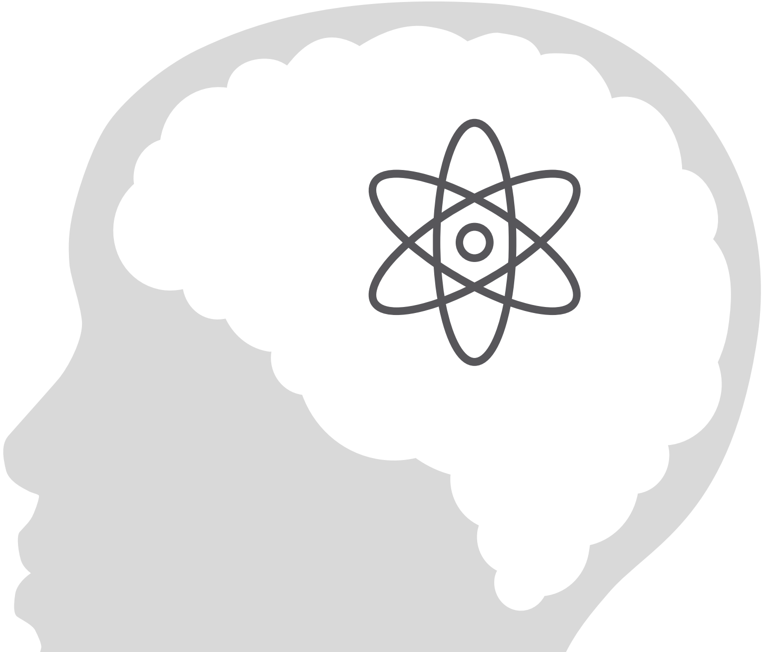
$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 x + \cdots \beta_k x_k$$

$$\mathbf{log\,(odds)} = \boldsymbol{\beta_0 + \beta_1 x + \cdots \beta_k x_k}$$

ODDS $= \dfrac{p(X)}{1-p(X)}$

Coin flip odds:
The odds of getting heads are 1:1 (or simply 1)

Fair die odds:
The odds of getting 4 are 1:5 (1 to 5)



Logistic regression model

Dep. Variable:	y	**No. Observations:**	518
Model:	Logit	**Df Residuals:**	516
Method:	MLE	**Df Model:**	1
Date:	Thu, 28 Nov 2019	**Pseudo R-squ.:**	0.2121
Time:	15:01:00	**Log-Likelihood:**	-282.89
converged:	True	**LL-Null:**	-359.05
		LLR p-value:	5.387e-35

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-1.7001	0.192	-8.863	0.000	-2.076	-1.324
duration	0.0051	0.001	9.159	0.000	0.004	0.006

The dependent variable, y; This is the variable we are trying to predict.

Indicates whether our model found a solution or not.

Coefficient of the intercept, b_0; sometimes we refer to this variable as constant or bias.

Coefficient of the independent variable i: b_i; this is usually the most important metric – it shows us the relative/absolute contribution of each independent variable of our model. For a logistic regression, the coefficient contributes to the **log odds** and cannot be interpreted directly.

McFadden's pseudo-R-squared, used for comparing variations of the same model. Favorable range [0.2,0.4].

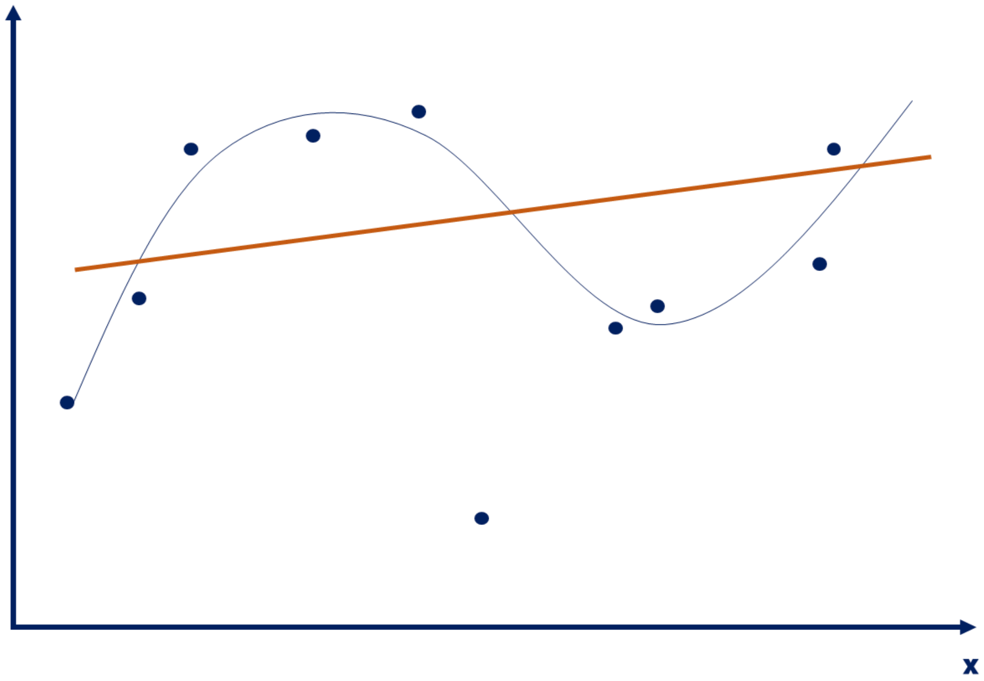
Log-**Likelihood*** (the log of the likelihood function). Always negative. We aim for this to be as high as possible.

Log-**Likelihood**-Null is the log-likelihood of a model which has no independent variables. It is used as the benchmark 'worst' model.

Log-**Likelihood** Ratio p-value measures of our model is statistically different from the benchmark 'worst' model.

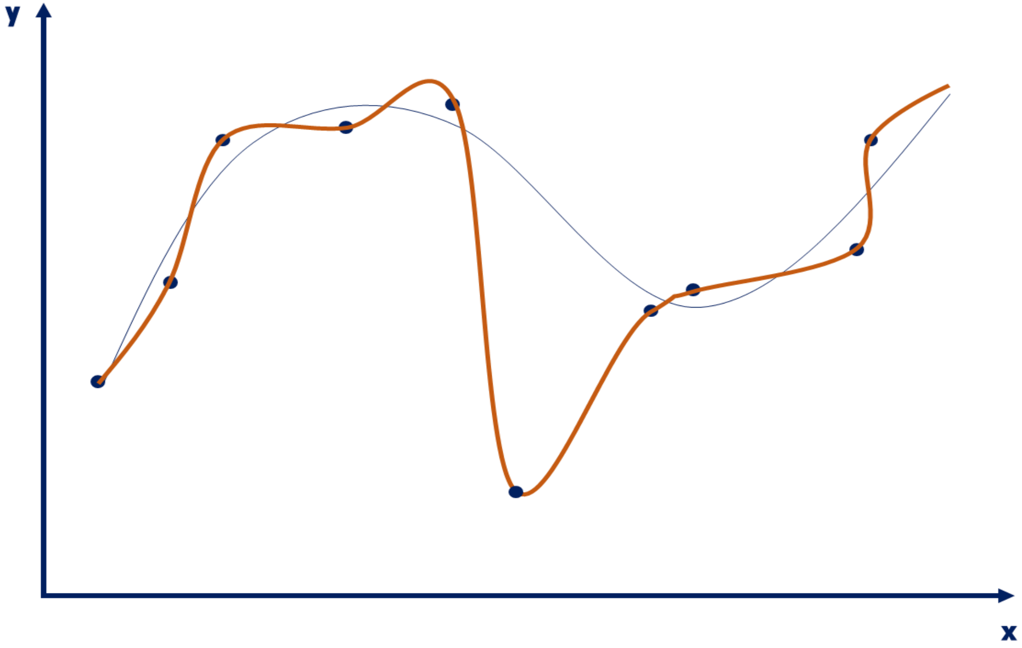
*Likelihood function: a function which measures the goodness of fit of a statistical model.
MLE (Maximum Likelihood Estimation) tries to maximize the likelihood function.

Underfitting



The model has not captured the underlying logic of the data.

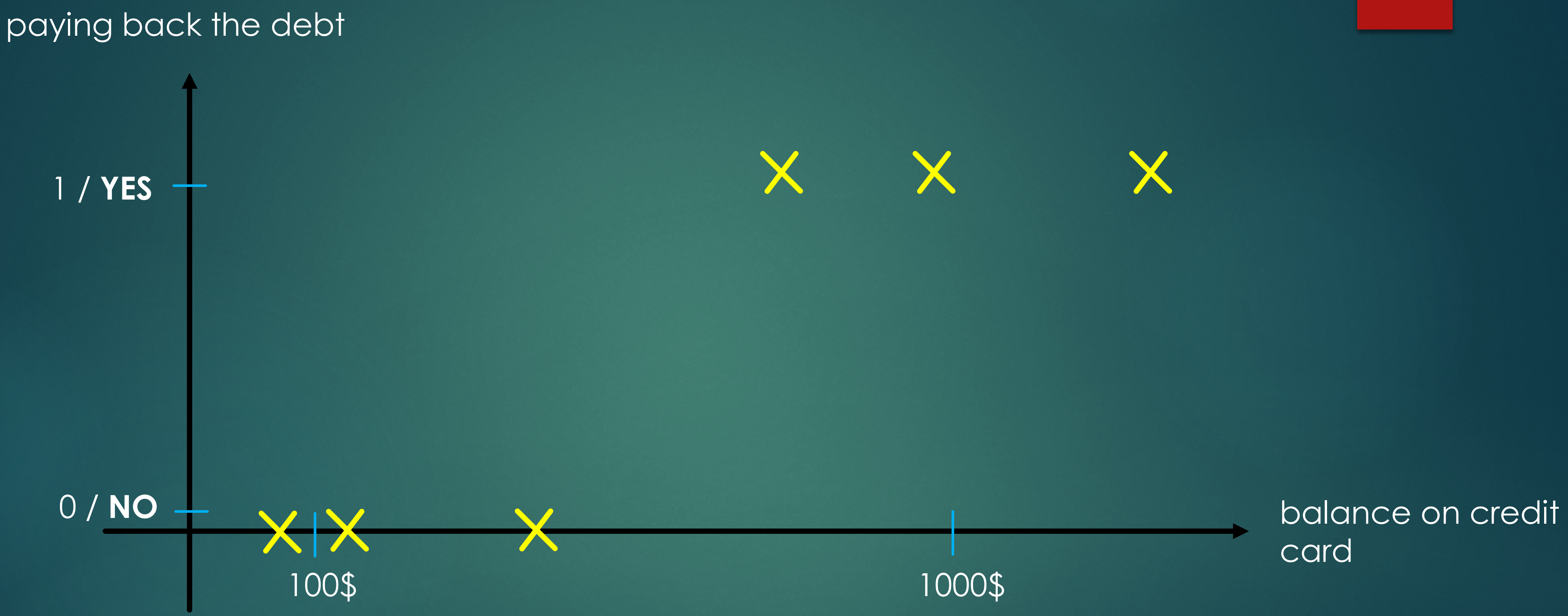
Overfitting

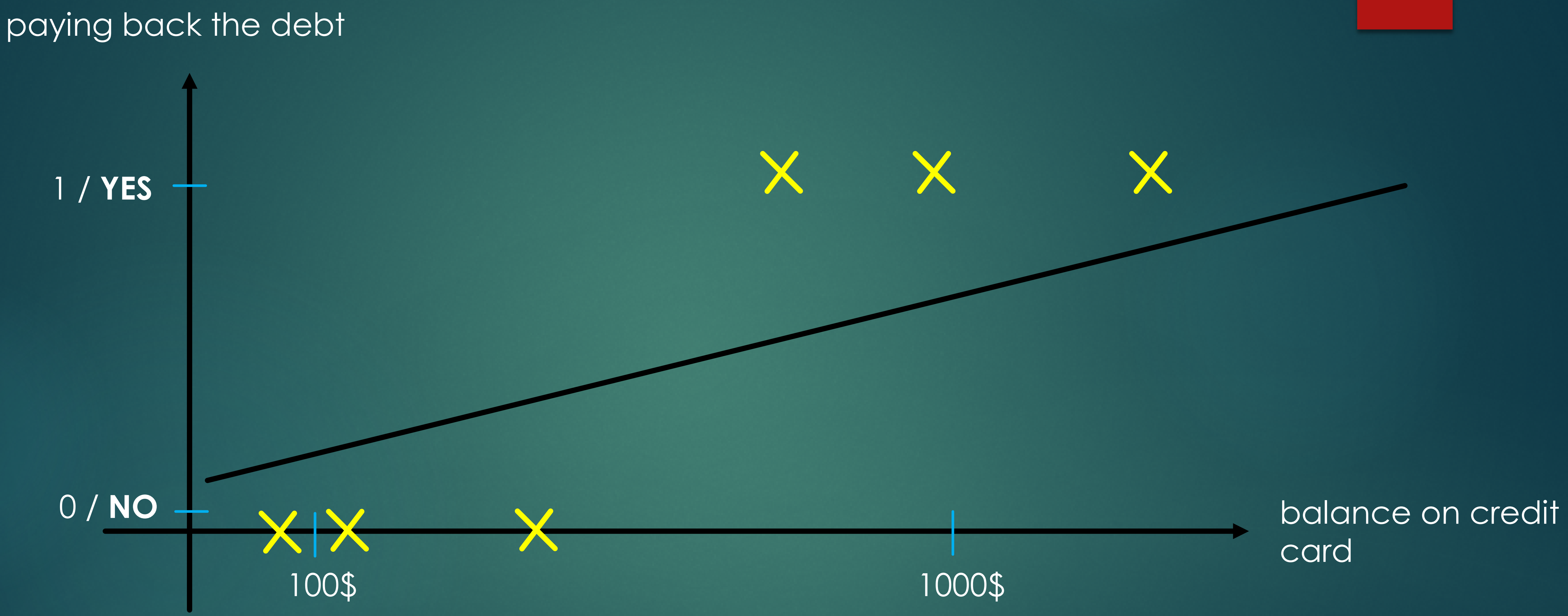


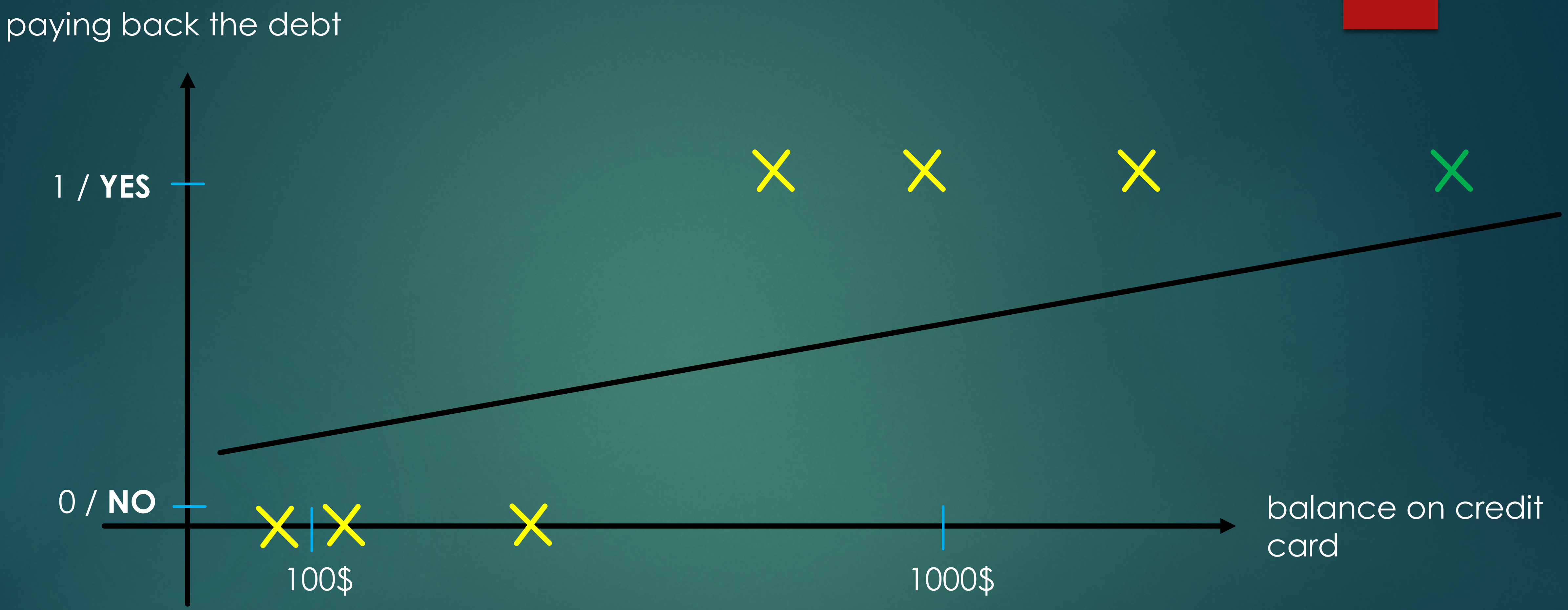
Our training has focused on the particular training set so much it has “missed the point”.

MACHINE LEARNING

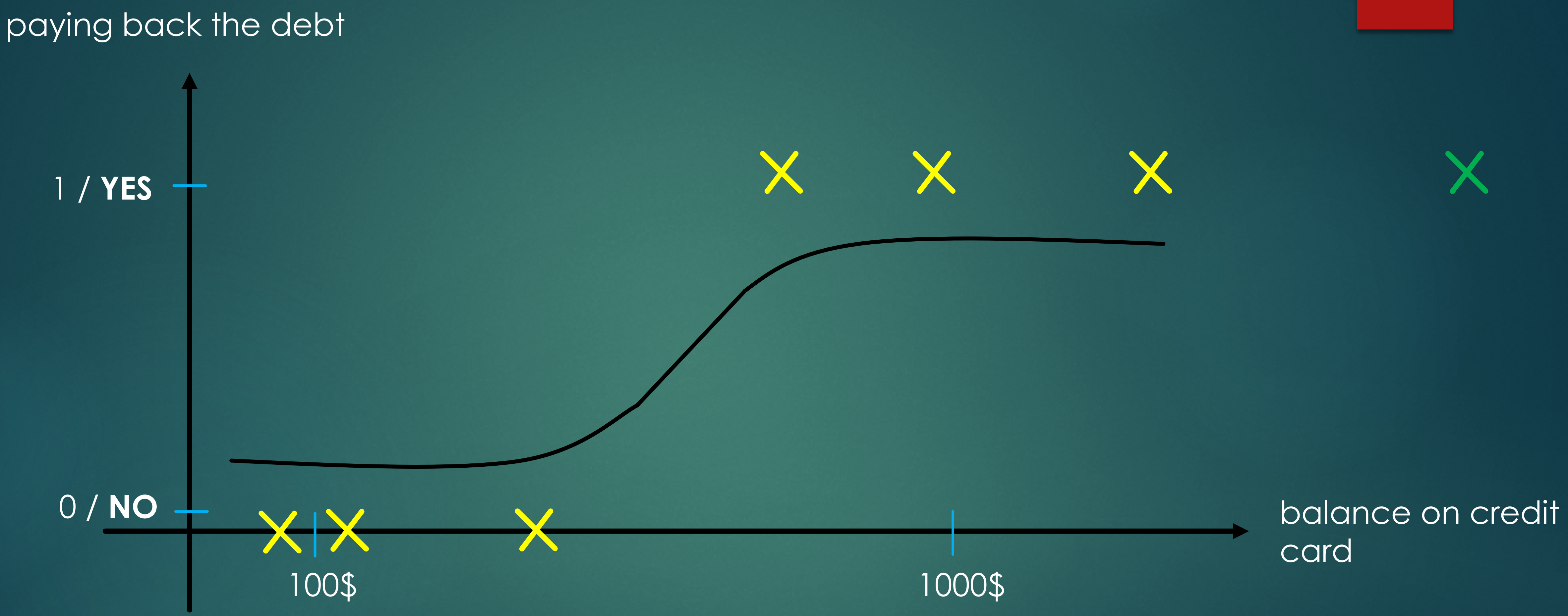
LOGISTIC REGRESSION


paying back the debt
1 / YES
0 / NO
100$
1000$
balance on credit card


paying back the debt
1 / YES
0 / NO
100$
1000$
balance on credit card



Sensitive to outliers: now the linear regression model is going to give us very bad predictions + we want to get some probability !!!



Sensitive to outliers: now the linear regression model is going to give us very bad predictions + we want to get some probability !!!

The **p(x) = P(default=1 | balance = x)** is the probability of default when we know the balance !!!

$$p(x) = \frac{e^{b_0 + b_1 * x}}{1 + e^{b_0 + b_1 * x}}$$

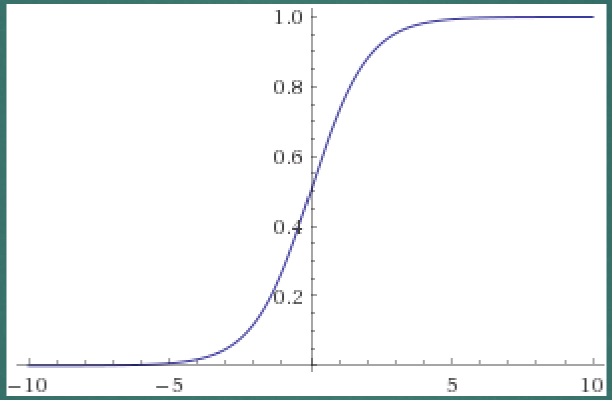
„sigmoid function”

It has a value between **0** and **1**

Logistic regression fits the $\mathbf{b_0}$ and $\mathbf{b_1}$ parameters, these are the regression parameters

This fitted curve is not linear: we can make it linear with the help of the **logit** transformation

Logistic function



„SIGMOID FUNCTION”

$\text{logit}\ p(x) = b_0 + b_1 * x$ **„logit transformation”**

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = b_0 + b_1 * x$$

The point of the **logit** transformation is to make it linear: so logistic regression is a linear regression on the logit transform !!!

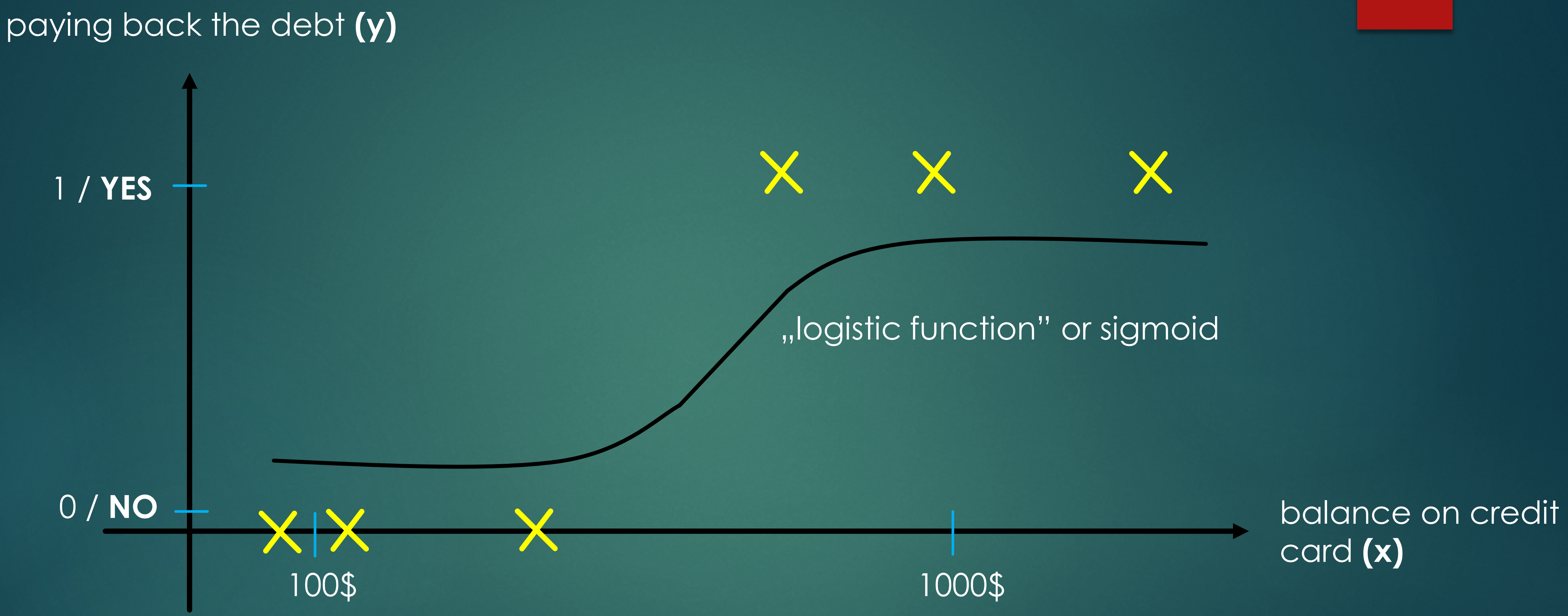
How to fit the parameters?

- maximum likelihood method
- gradient descent method

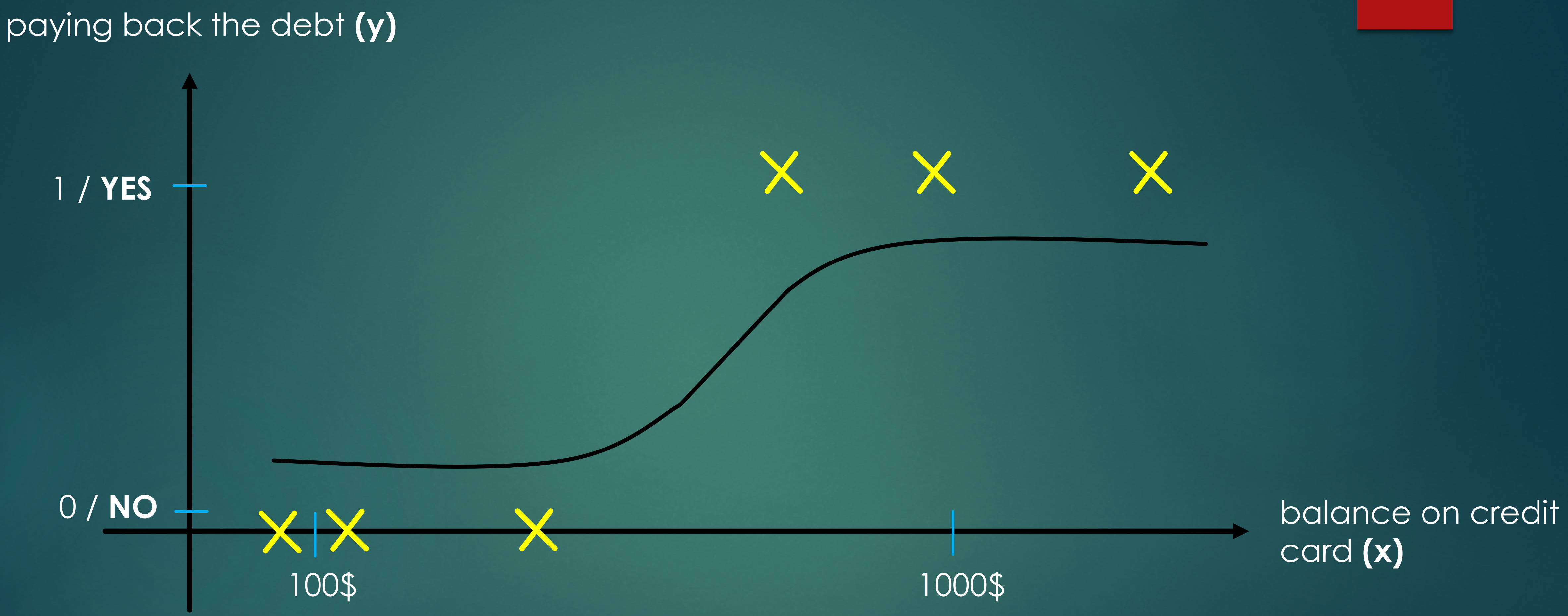
Multivariate logistic regression

We try to make some predictions → whether the given person will default or not
~ we have some data → income + balance + age // 3 features !!!

$$p(x) = \frac{e^{b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n}}{1 + e^{b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n}}$$



It is better: it is between **[0:1]** + we want to assign a probability to each balance



It is better: it is between **[0:1]** + we want to assign a probability to each balance sigmoid function

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_{\beta}(x) = g(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 * x)}}$$

linear model when
$z = \beta_0 + \beta_1 * x$

$$g(z=-inf) = 0$$

$$g(z=0) = 0.5$$

$$g(z=inf) = 1$$

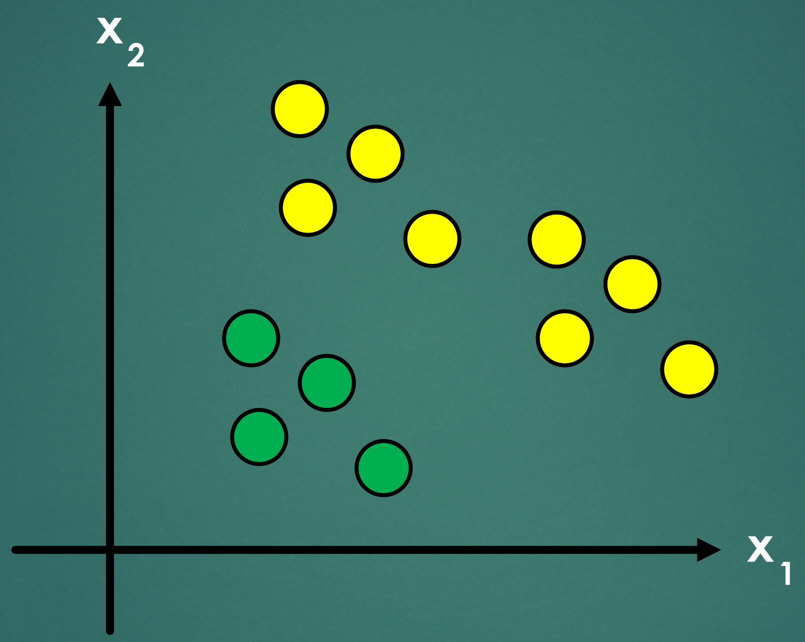
This sigmoid function is always in the interval **[0:1]** so it is good for predicting probablilities !!!

Logistic regression

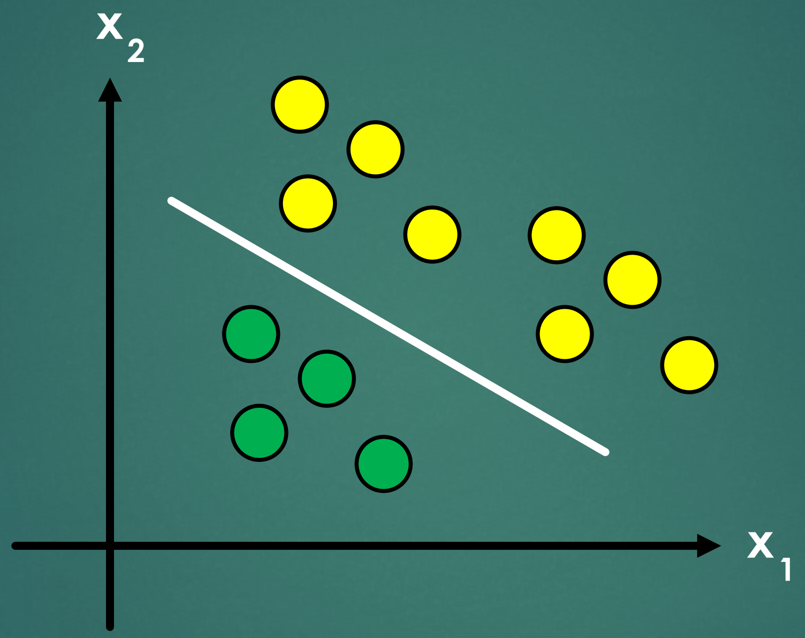
- It is a linear classifier !!!
- We have to fit the **ß** parameters first, after that the **g(z)** is going to give us the predictions
- **h(x)** is the hypothesis → it is going to tell us the probability of **y** when we have the given **x** input
- For examp in the credit scoring example: **h(x) > 0.5** → **y=1** which means no default
- If **h(x) < 0.5** → **y=0** // the given person has defaulted

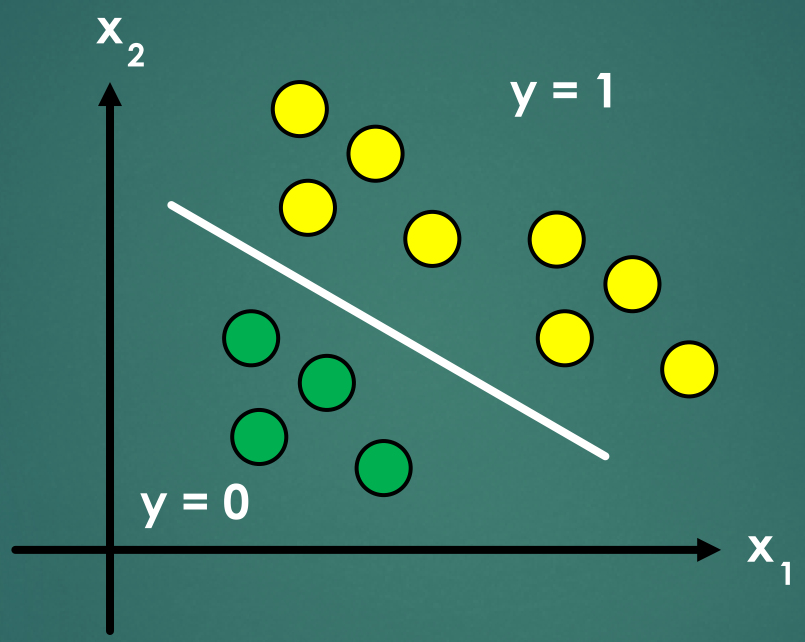
IT IS THE SAME AS:

- z < 0 default
- z > 0 no default
- z = 0 „decision boundary”

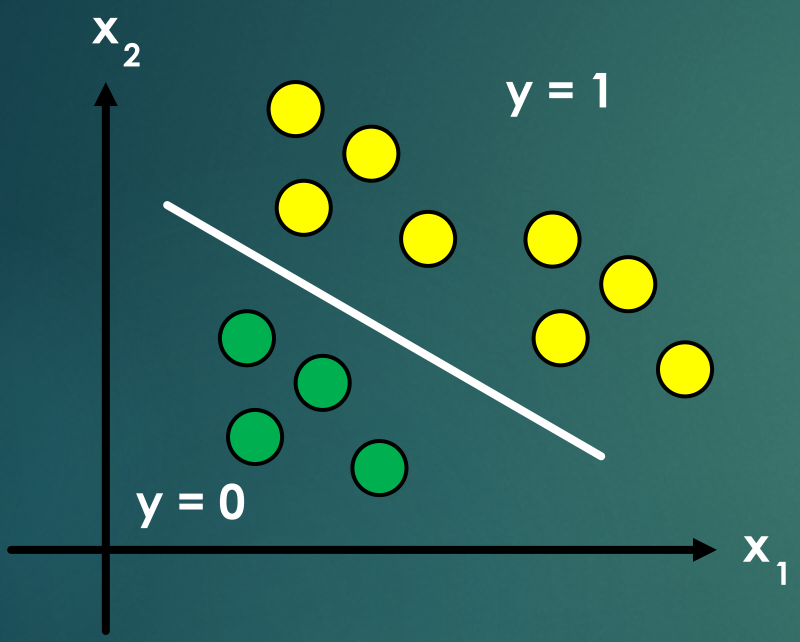


x_2
x_1


x2
x1



x_2
y = 1
y = 0
x_1



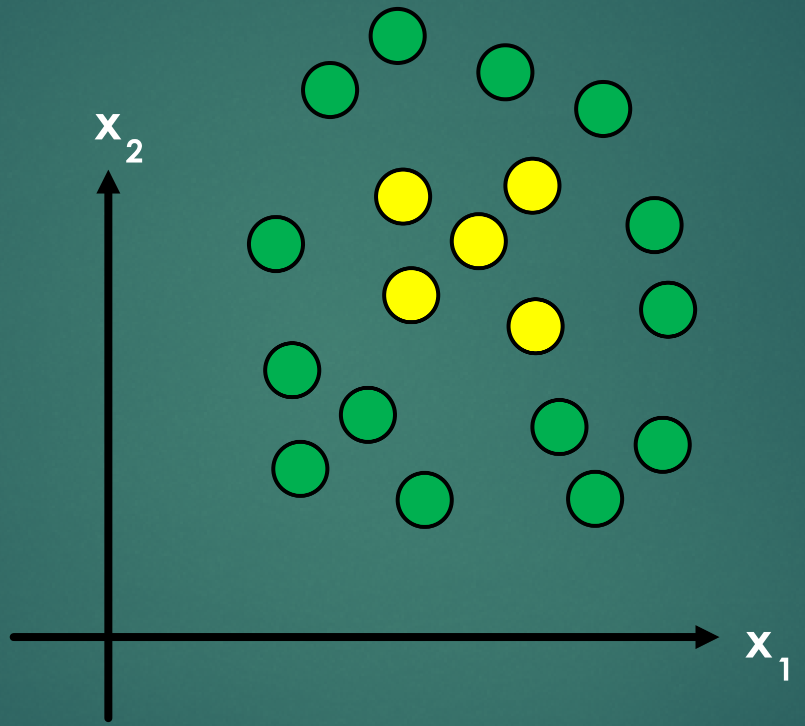
$h_\beta(x) = g(\beta_0 + \beta_1 * x_1 + \beta_2 * x_2)$ this is our model

We can calculate the **ß** values with the help of gradient descent !!!

$\beta_0 = -3 \quad \beta_1 = 1 \quad \beta_2 = 1$

$-3 + x_1 + x_2 = 0$ this is the decision boundary

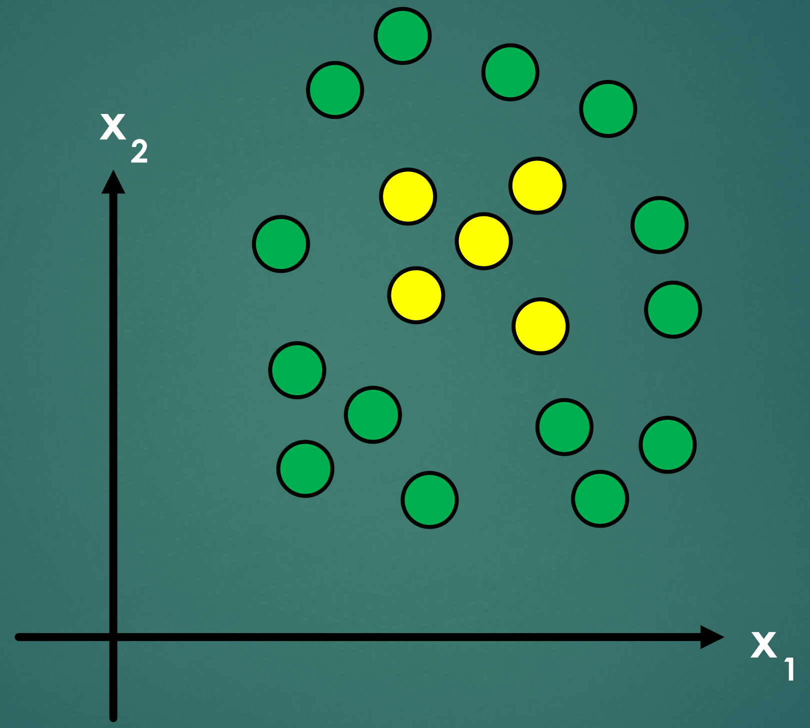
$X_2 = 3 - x_1$



x_2
x_1

$$h_{\beta}(x) = g(\beta_0 + \beta_1 * x_1^2 + \beta_2 * x_2^2)$$

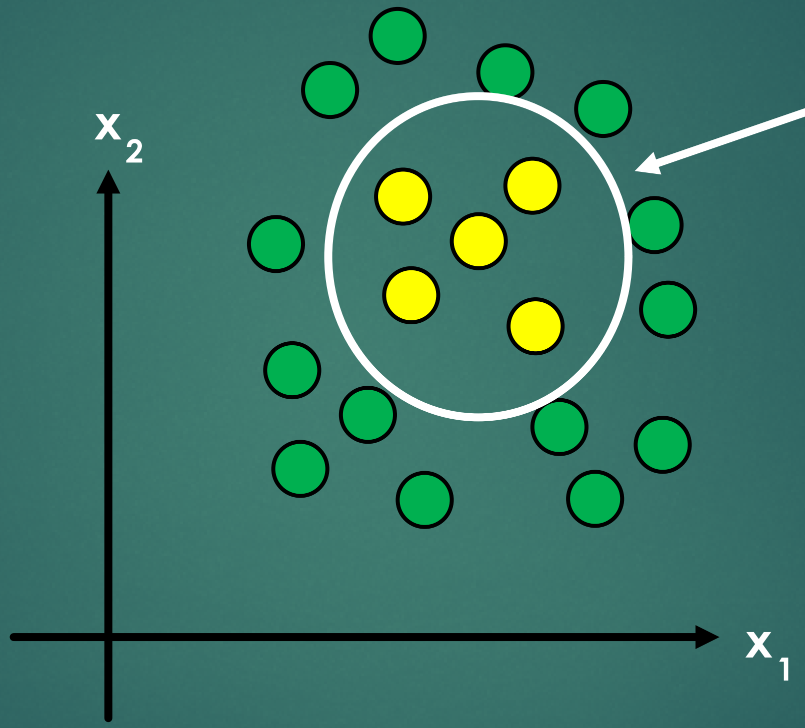
this is our model



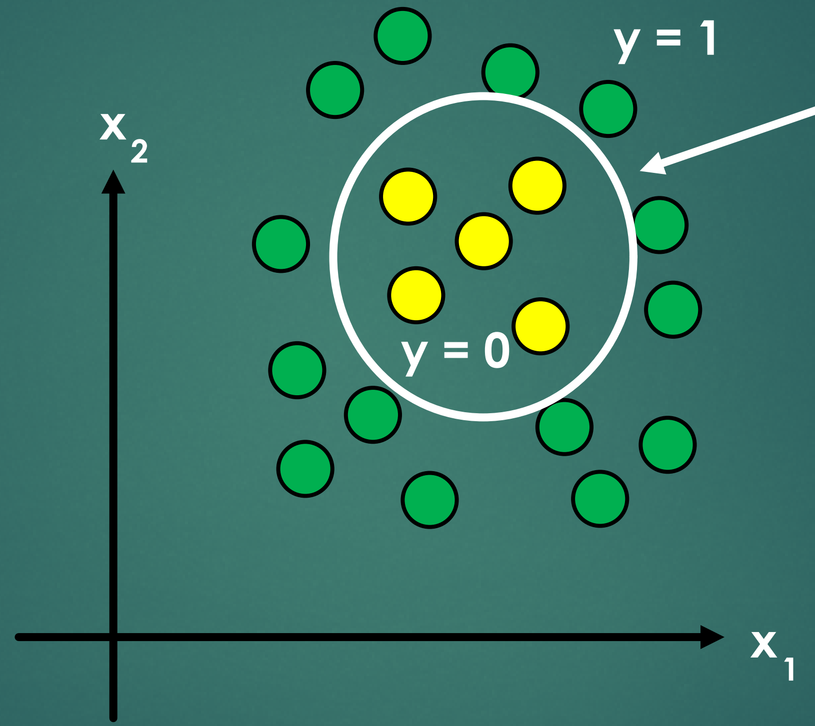
$$h_{\beta}(x) = g(\beta_0 + \beta_1 * x_1^2 + \beta_2 * x_2^2)$$

this is our model

this is the decision boundary now



$$h_\beta(x) = g(\beta_0 + \beta_1 * x_1^2 + \beta_2 * x_2^2)$$ this is our model



this is the decision boundary now

Confsusion matrix

		PREDICTED	
		0	1
ACTUAL	0	122	12
	1	34	89

Describes the performance of a classification model

→ diagonal elements: the correct classifications

→ off-diagonals: incorrect predictions