# Implementing Logistic Regression Models in Python

**Vitthal Srinivasan**
CO-FOUNDER, LOONYCORN

www.loonycorn.com

# Overview

Set up a logistic regression to predict whether a stock will rise or fall

Solve this logistic regression in Python

Extend the logistic regression to include multiple explanatory variables

“Make the common use-case easy and the difficult use-case possible.”

# Regression: Excel, R or Python?

**Excel**

Create a regression slide for an important presentation

**R**

Create a regression case study for a seminar

**Python**

Build trading model that scrapes websites, combines sentiment analysis and regression

# Regression: Excel, R or Python?

**Excel**

**Presentations**

**R**

**Seminars**

**Python**

**Trading models**

# R for Regression

R

Presentations

R

Seminars

R

Trading models

# Demo

## Implement Logistic Regression in Python

# Logistic Regression in Python

**Cause**

**Changes in S&P 500**

**Effect**

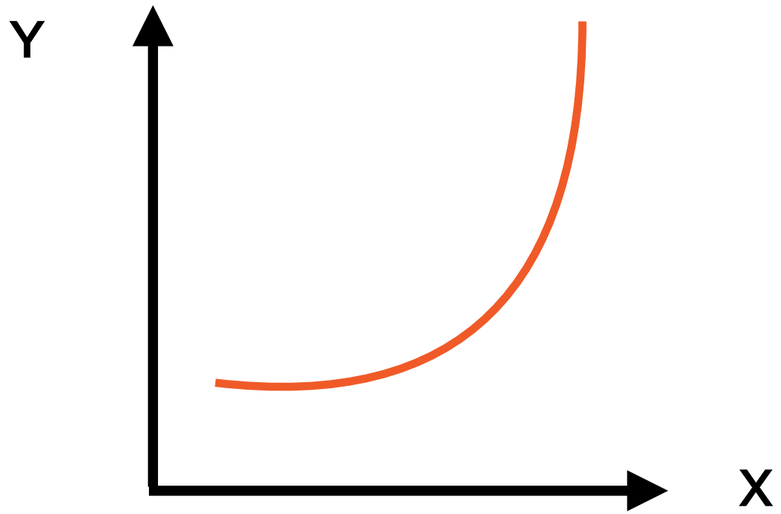
**Changes in price of Google Stock**

# Logistic Regression in Python

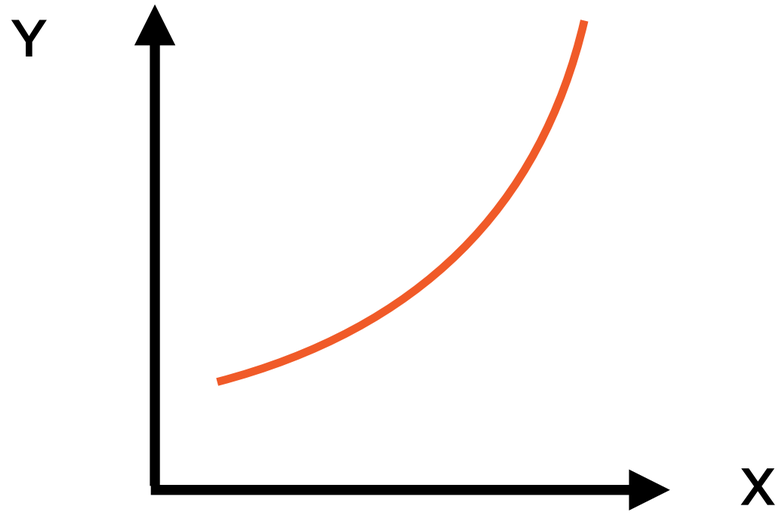
**y = Returns on Google stock (GOOG)**

**x = Returns on S&P 500 (S&P500)**

# Never Regress Non-Stationary Data




Exponential




Polynomial

**Smoothly trending data will lead to poor quality regression models**

# First Differences

$y'_{12} = \log y_2 - \log y_1$

$x'_{12} = \log x_2 - \log x_1$

Regress y' and x'

**Log Differences**

$y'_{12} = (y_2 - y_{1)}/y_1$

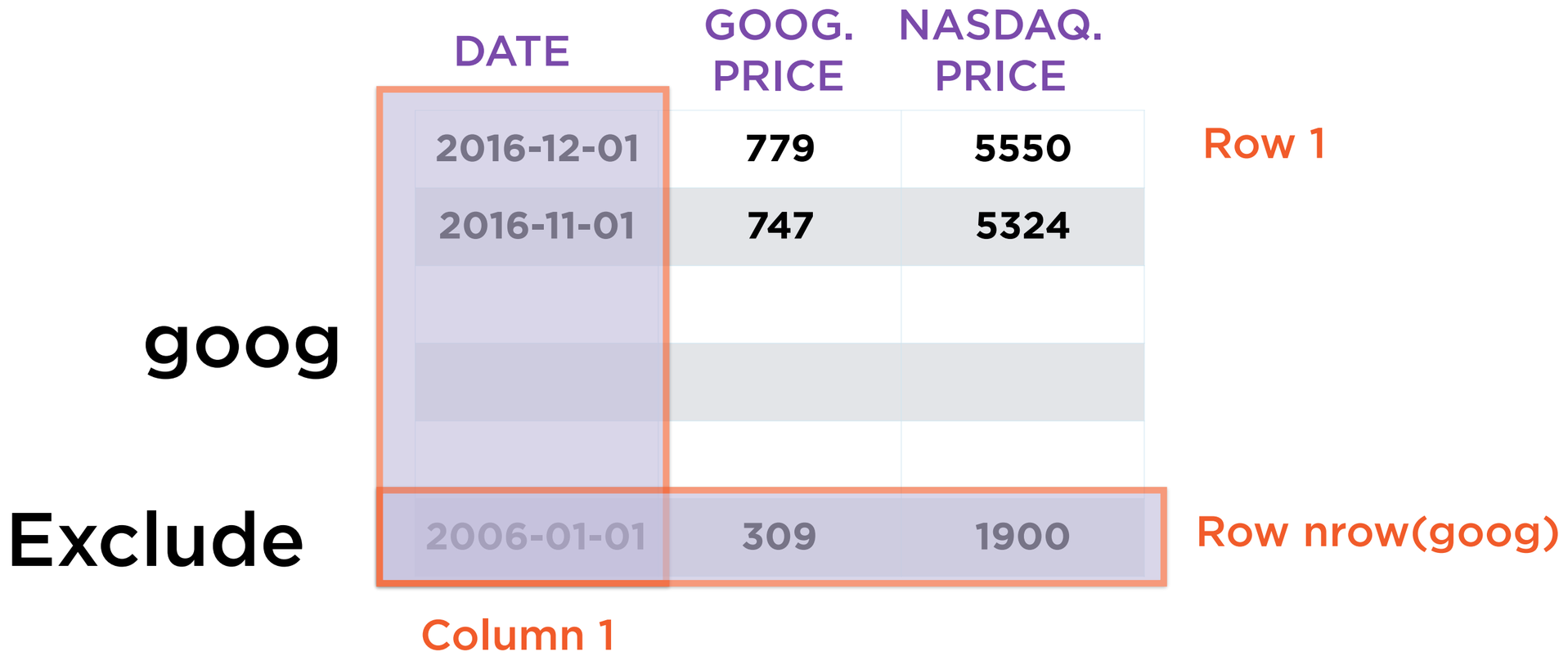
$x'_{12} = (x_2 - x_{1)}/x_1$

Regress y' and x'

**Returns**

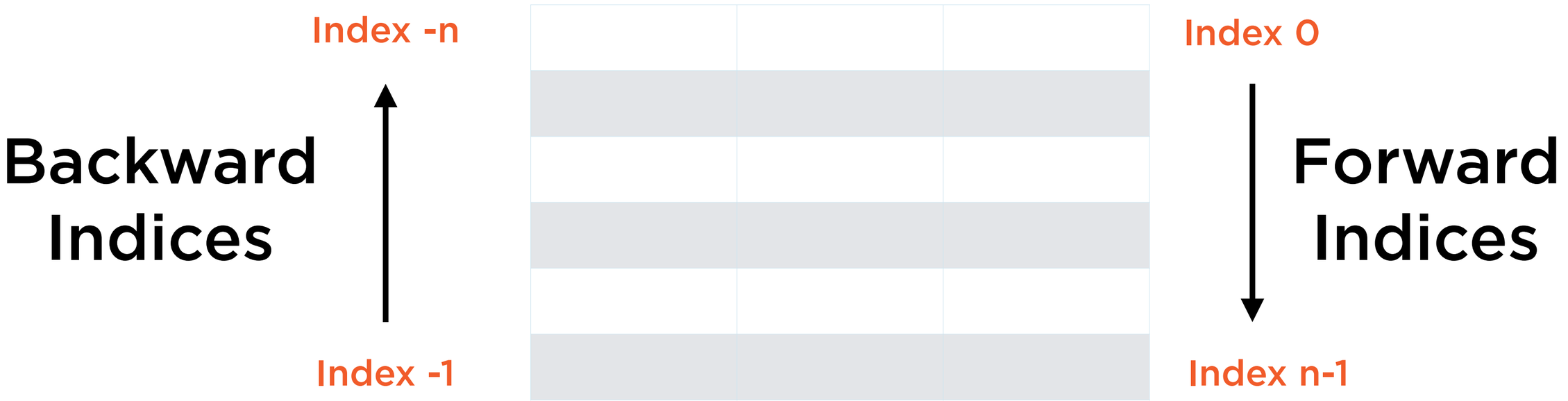
Take first differences of smooth data converting either to log differences or returns

# Negative Indices in R

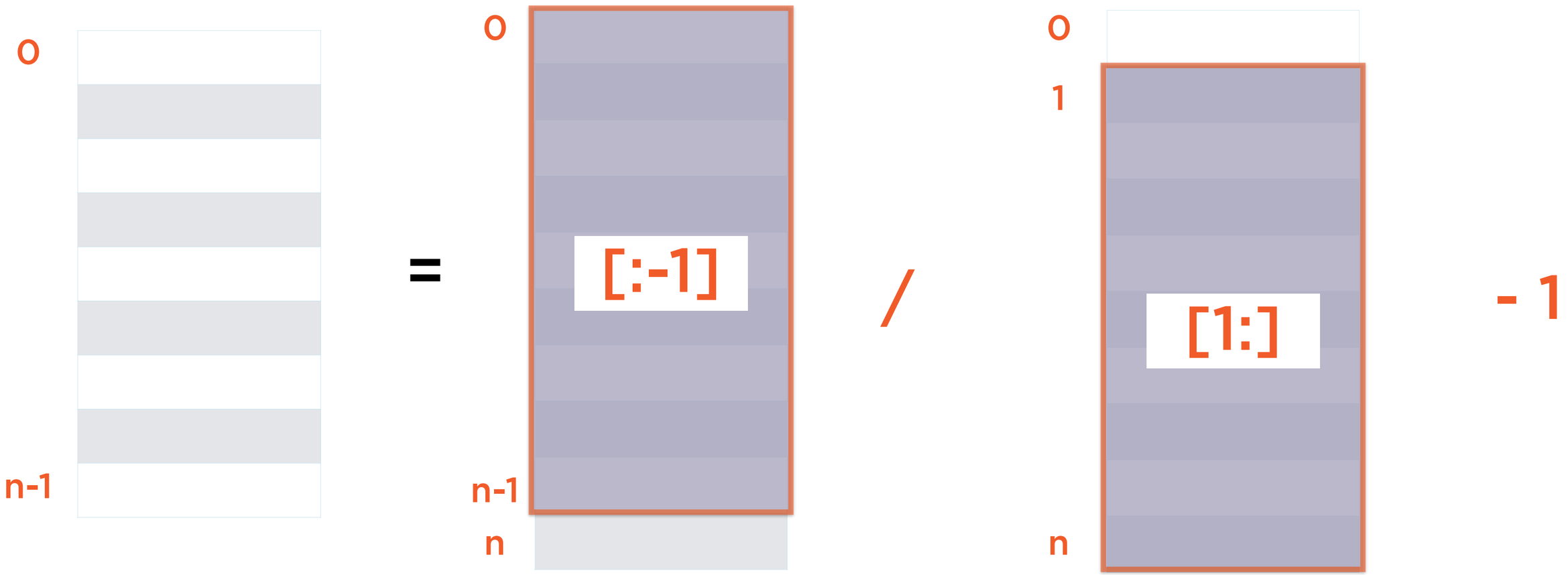



```
goog[-nrow(goog),-1]
```

# Negative Indices In Python



# Prices to Returns




**Returns = Prices[:-1] / Prices[1:] - 1**

# Using Logistic Regression

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

P(y) = Probability of Google going up in the current month i

x = Returns on S&P 500 for current month

```
logit = sm.Logit(yData,xData)
```

# A Much Harder Problem

$$p(y_i) = \frac{1}{1 + e^{-(A + B^{GOOG} x^{GOOG}_{i-1} + B^{SP500} x^{SP500}_{i-1})}}$$

P(y) = Probability of Google going up in the current month i

$x^{GOOG}$ = Returns on GOOG for previous month

$x^{SP500}$ = Returns on S&P 500 for previous month

```
logit = sm.Logit(yData,xData)
```

# Two Approaches to Deadlines

**Start 5 minutes before deadline**

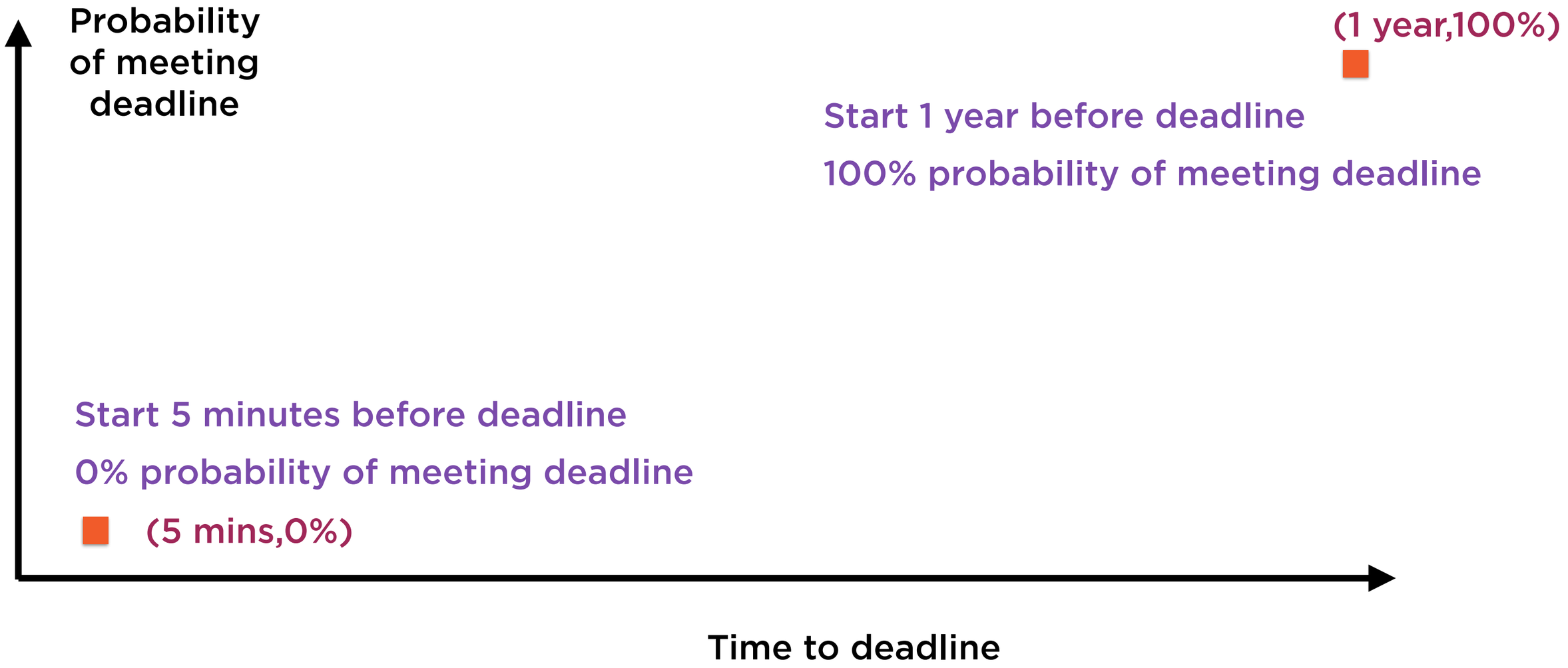
Good luck with that

**Start 1 year before deadline**

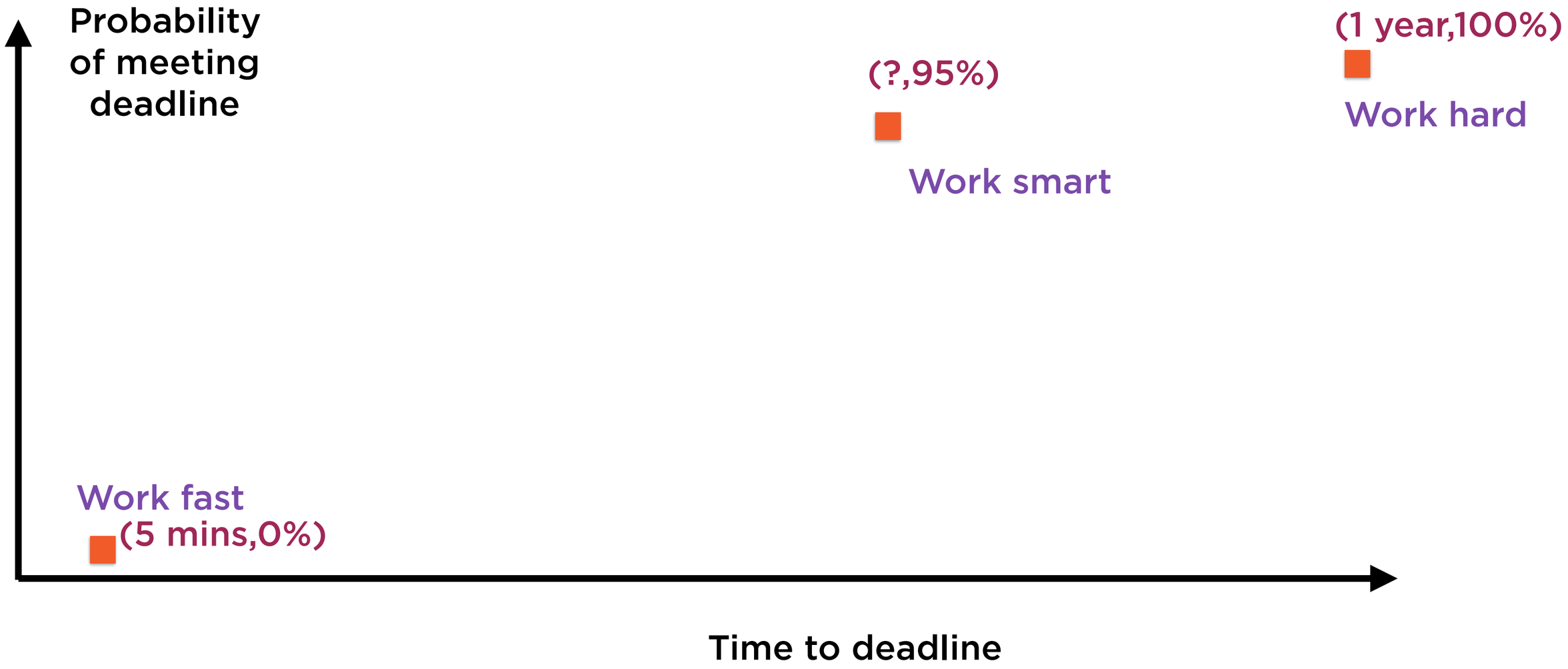
Maybe overkill

## Neither approach is optimal

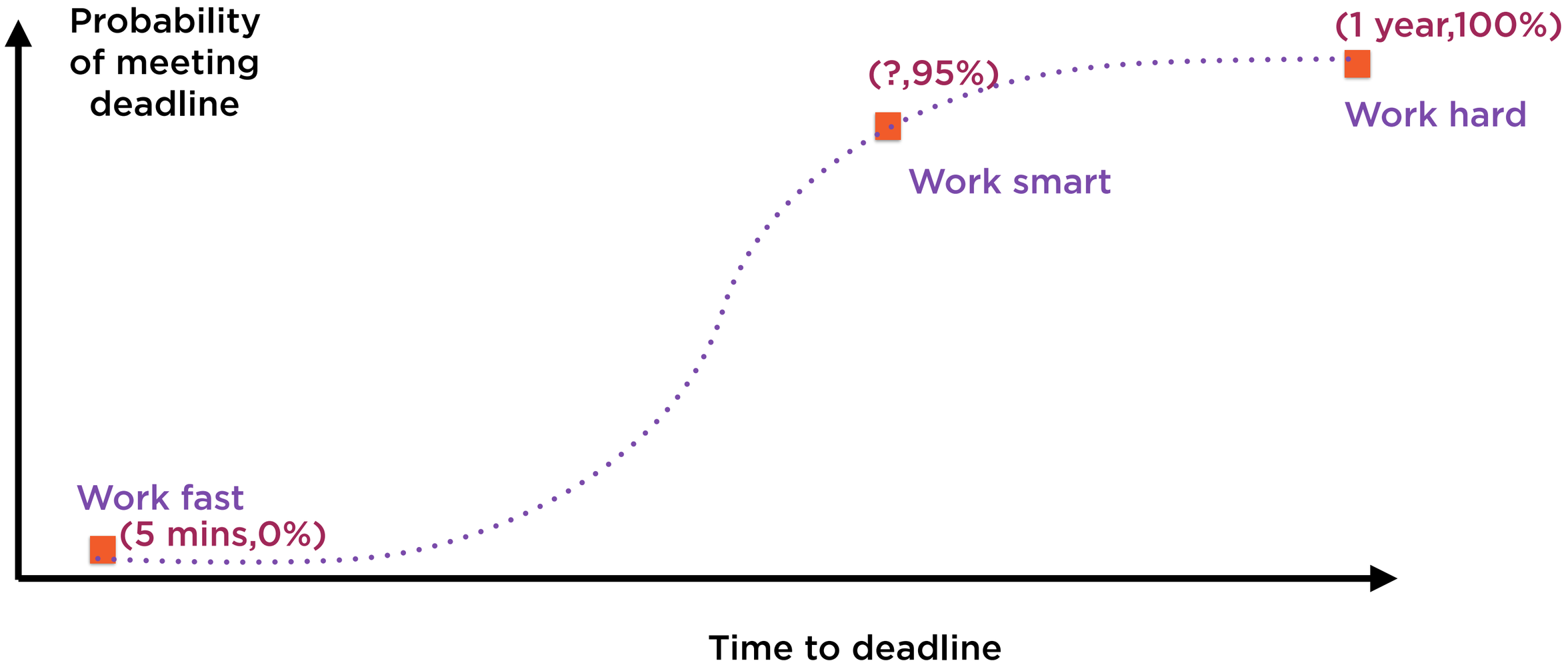
# Working Hard, Fast, Smart



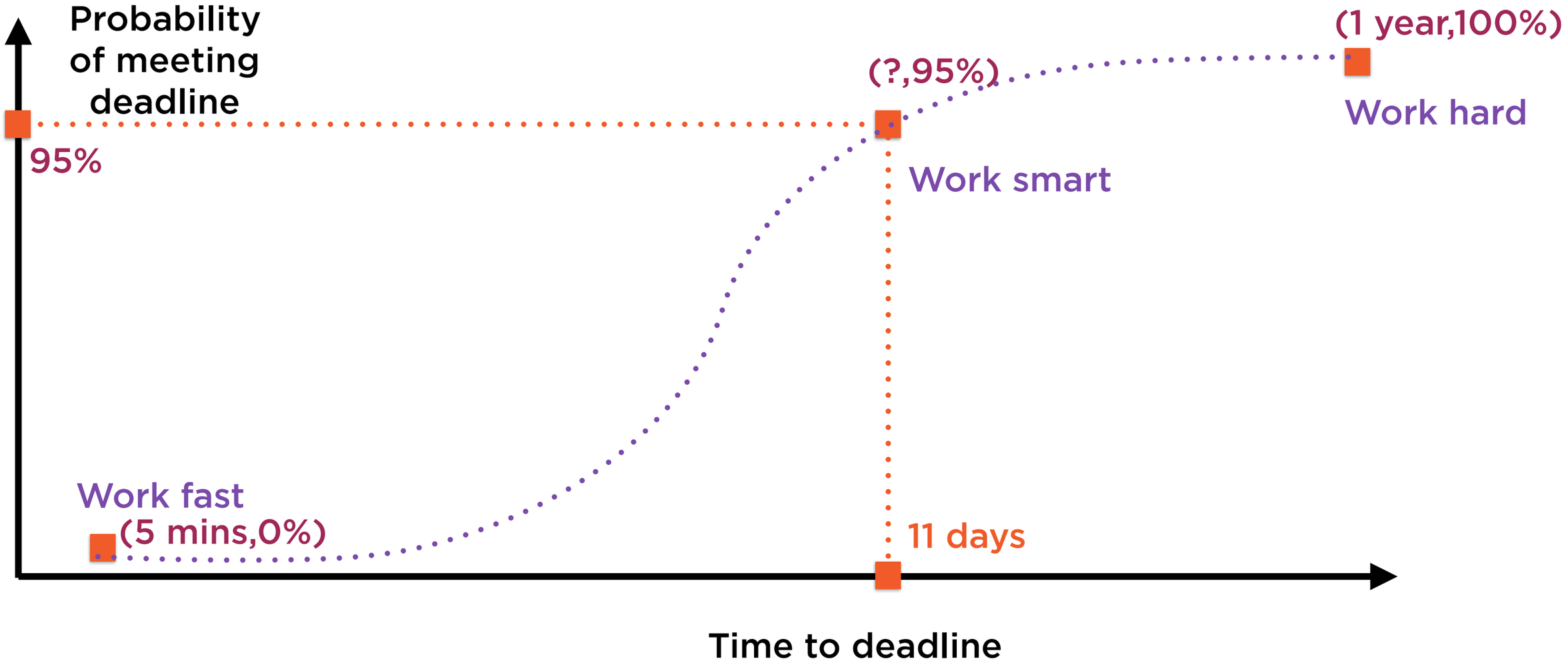
# Working Hard, Fast, Smart



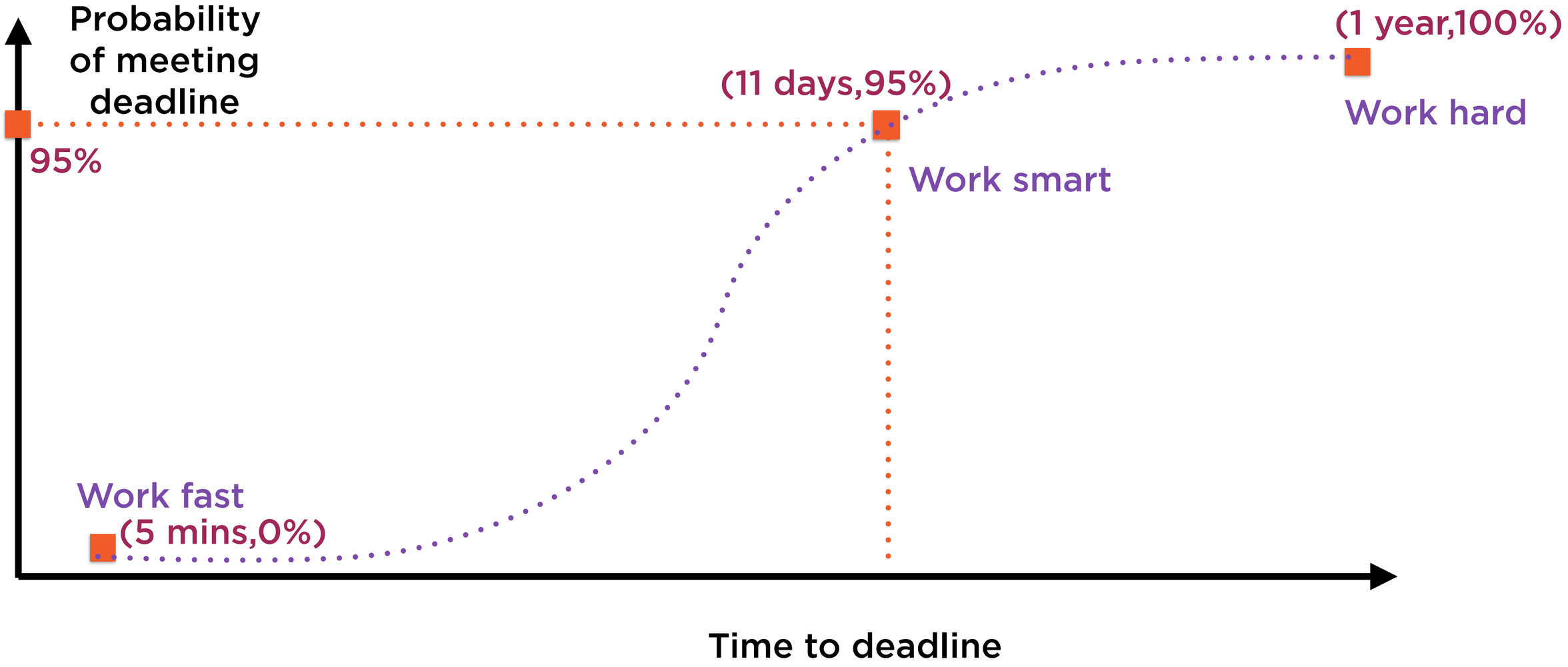
# Working Hard, Fast, Smart



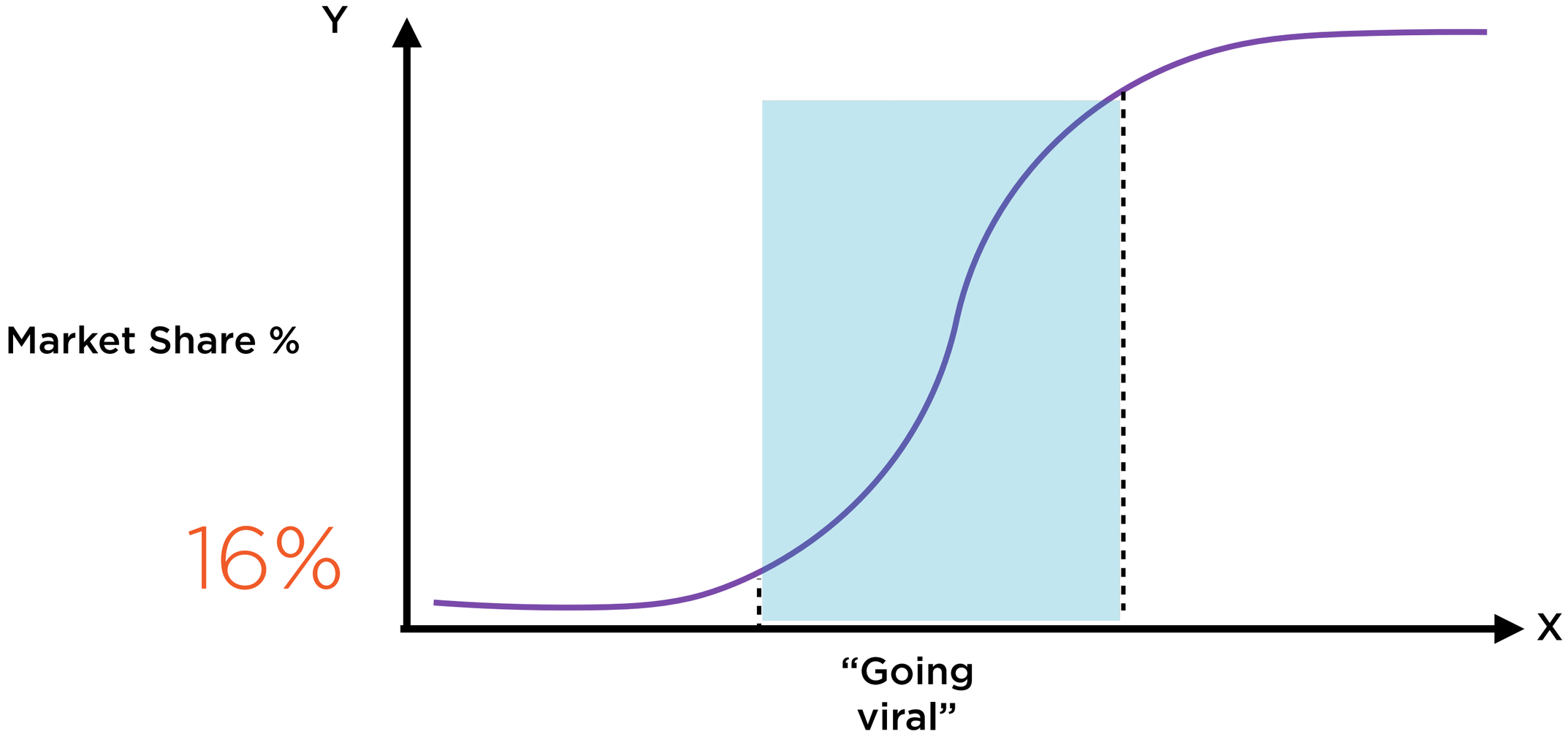
# Working Hard, Fast, Smart



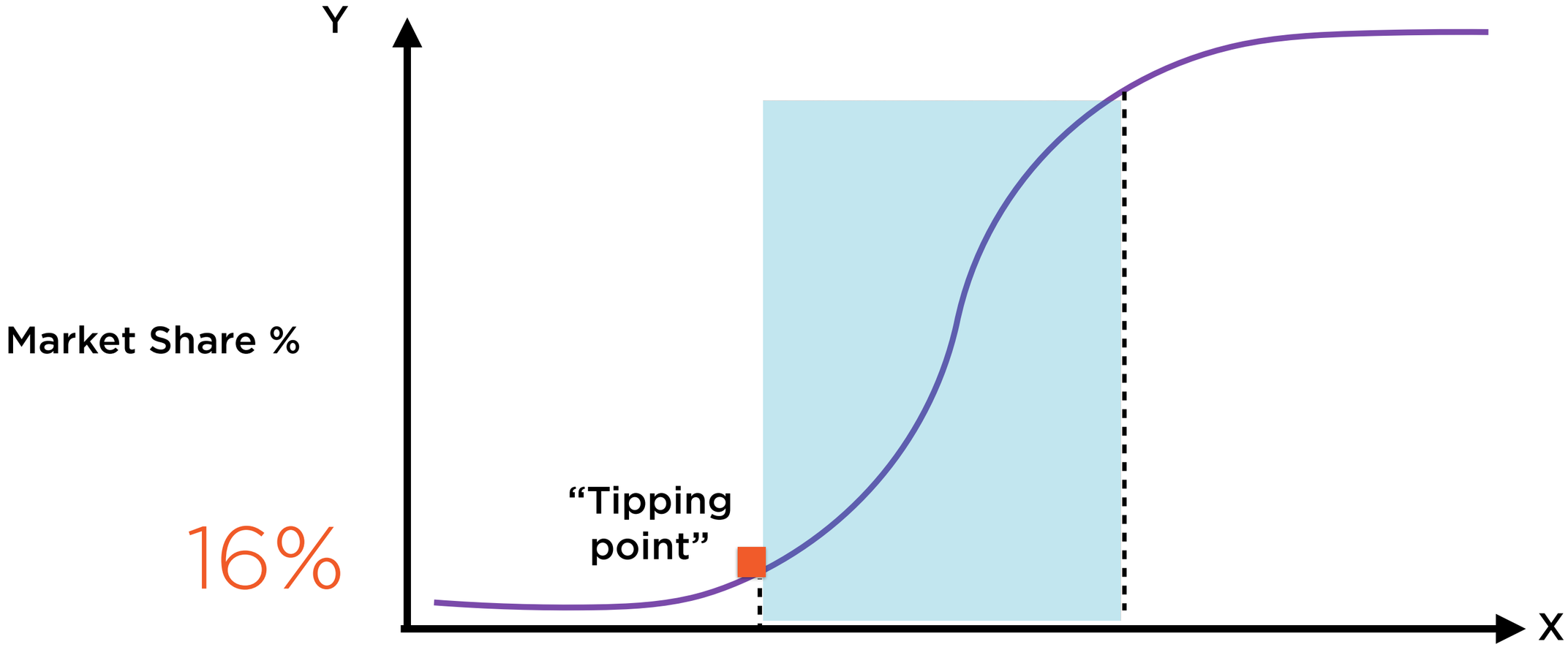
# Working Hard, Fast, Smart



# Diffusion of Innovation



# Diffusion of Innovation



$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

Logistic regression involves finding the "best fit" such curve

- A is the intercept

- B is the regression coefficient

*(e is the constant 2.71828)*

# Regression: Excel, R or Python?

**Excel**

Create a regression slide for an important presentation

**R**

Create a regression case study for a seminar

**Python**

Build trading model that scrapes websites, combines sentiment analysis and regression

# Summary

**Logistic regression can be very easily implemented in Python**