

Mohamed Noordeen Alaudeen

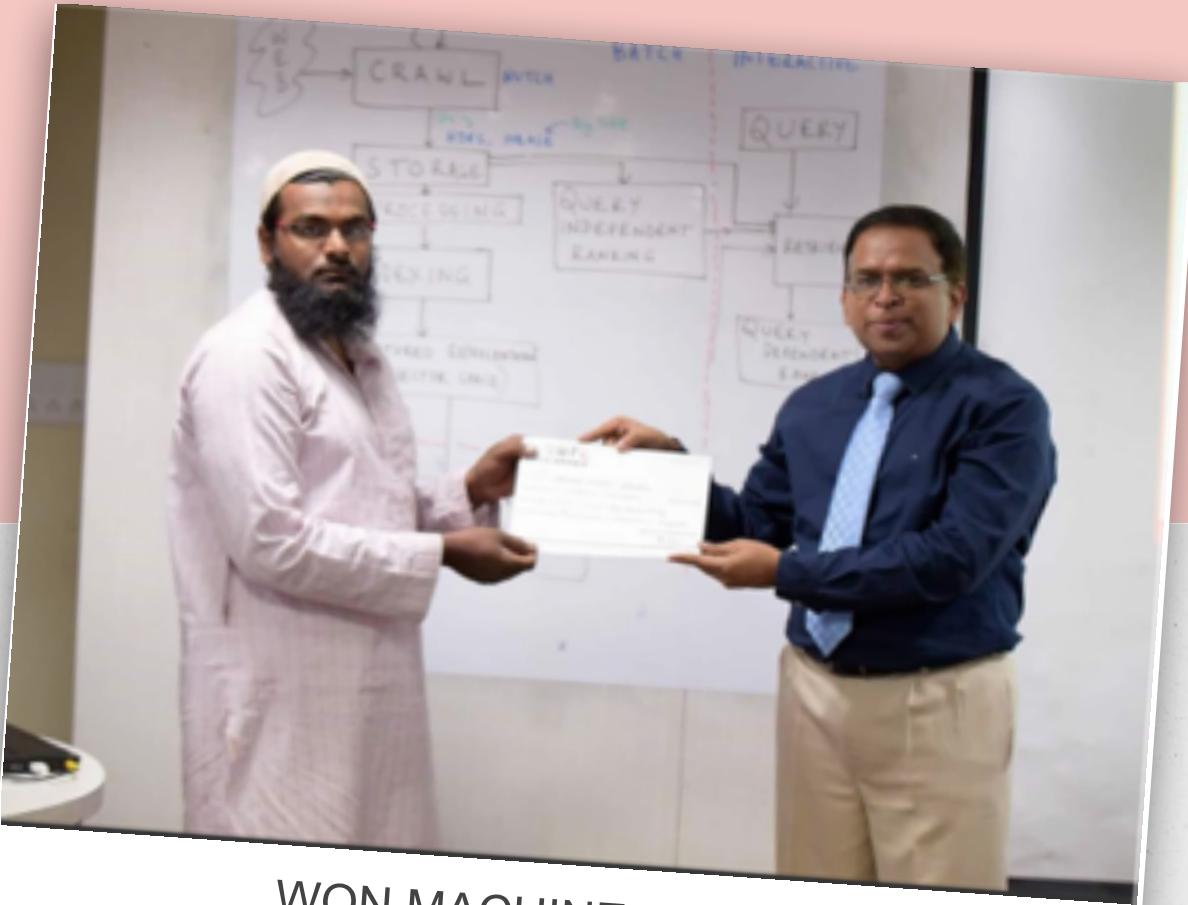
Senior Data Scientist - Logitech

About me



logitech®





WON MACHINE LEARNING
CONTEST



WON BEST PROJECT
AWARD

International School of Engineering

awards

Certificate in Engineering Excellence in Big Data Analytics and Optimization

to

Mohamed Noordeen Alaudeen

on successful completion of all the requirements of the 352-hour program
conducted between December 10, 2016 and June 04, 2017 followed by a project defense.

This program is certified for quality of content, assessment and pedagogy by the Language Technologies Institute (LTI)
of Carnegie Mellon University (CMU). LTI also provided assistance in curriculum development for this program.



Dated this second day of August, two thousand and seventeen.

Dakshinamurthy V.K.
Dr. Dakshinamurthy V Kolluru

President



Sridhar Pappu
Dr. Sridhar Pappu
Executive VP - Academics



Printed details are on the back

BEST STUDENT OF THE BATCH



GUEST LECTURES



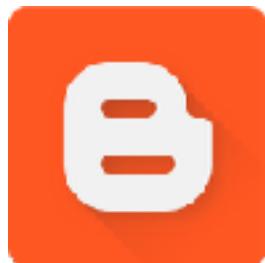
NOT THE FIRST TIME FOR
US

MY WORK

<https://github.com/nursnaaz>

<https://www.linkedin.com/in/nursnaaz/>

<http://www.technaaz.com/>



Blogger

Summary

Python

General purpose high level programming language

Who gifted Python?

Guido Van Rossam

1989

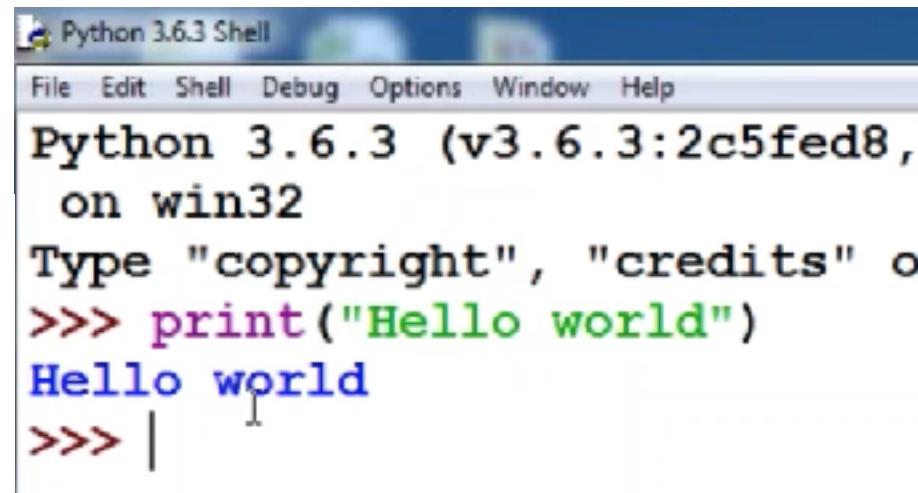
National Research institute in Netherland

Official Birthday - Feb 20 1991



```
1 class
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello World!");
6     }
7 }
8
```

```
>>> a,b=10,20
>>> print(a+b)
30
>>> a,b,c,d,e=10,20,30,40,50
```



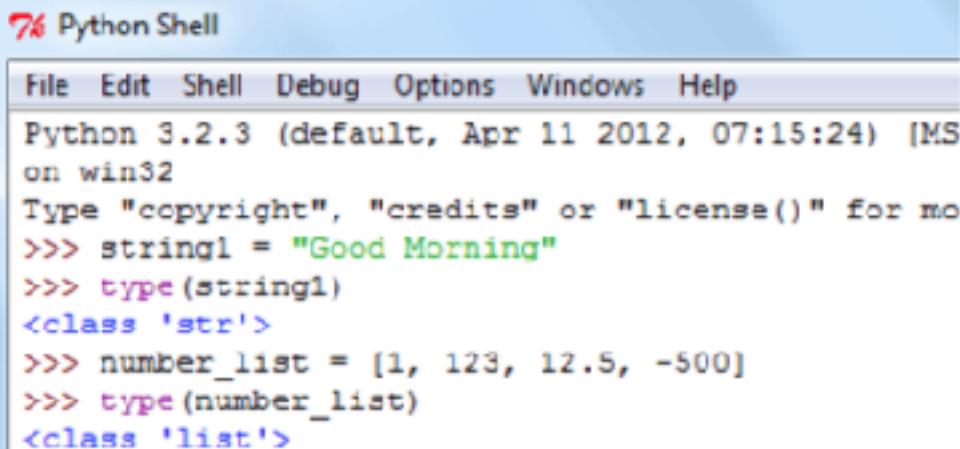
The screenshot shows a Windows-style application window titled "Python 3.6.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays the Python interpreter's response to the executed Java code. It starts with the Python version and build information, followed by a prompt to type "copyright" or "credits". Then, it executes the print statement from the Java code, outputting "Hello world". Finally, it shows the standard Python prompt "=>>> |".

```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8,
on win32
Type "copyright", "credits" o
>>> print("Hello world")
Hello world
>>> |
```

Statistically typed programming - C, Java

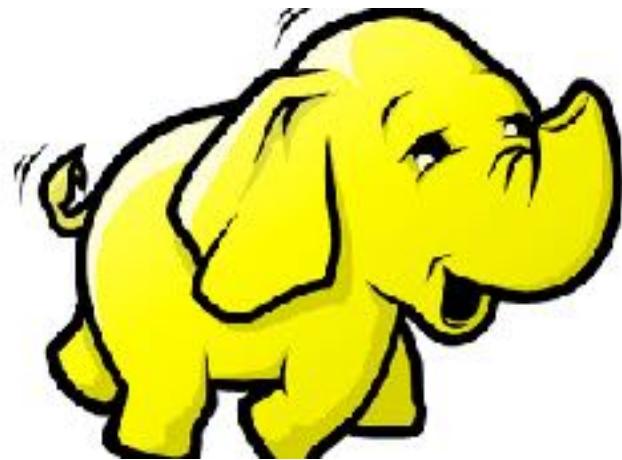
```
//Program entry
public static void Main(string[] args0)
{
    int a, b, result; //Integer variables
```

Dynamically typed programming - Python

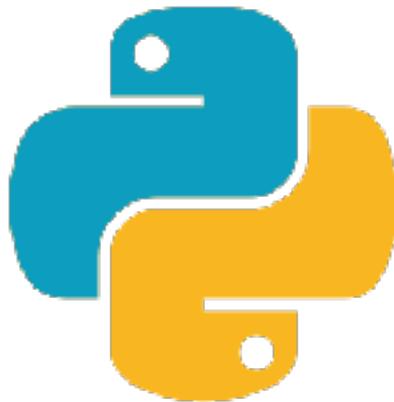


```
76 Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MS
on win32
Type "ccopyright", "credits" or "license()" for mo
>>> string1 = "Good Morning"
>>> type(string1)
<class 'str'>
>>> number_list = [1, 123, 12.5, -500]
>>> type(number_list)
<class 'list'>
```

Hadoop



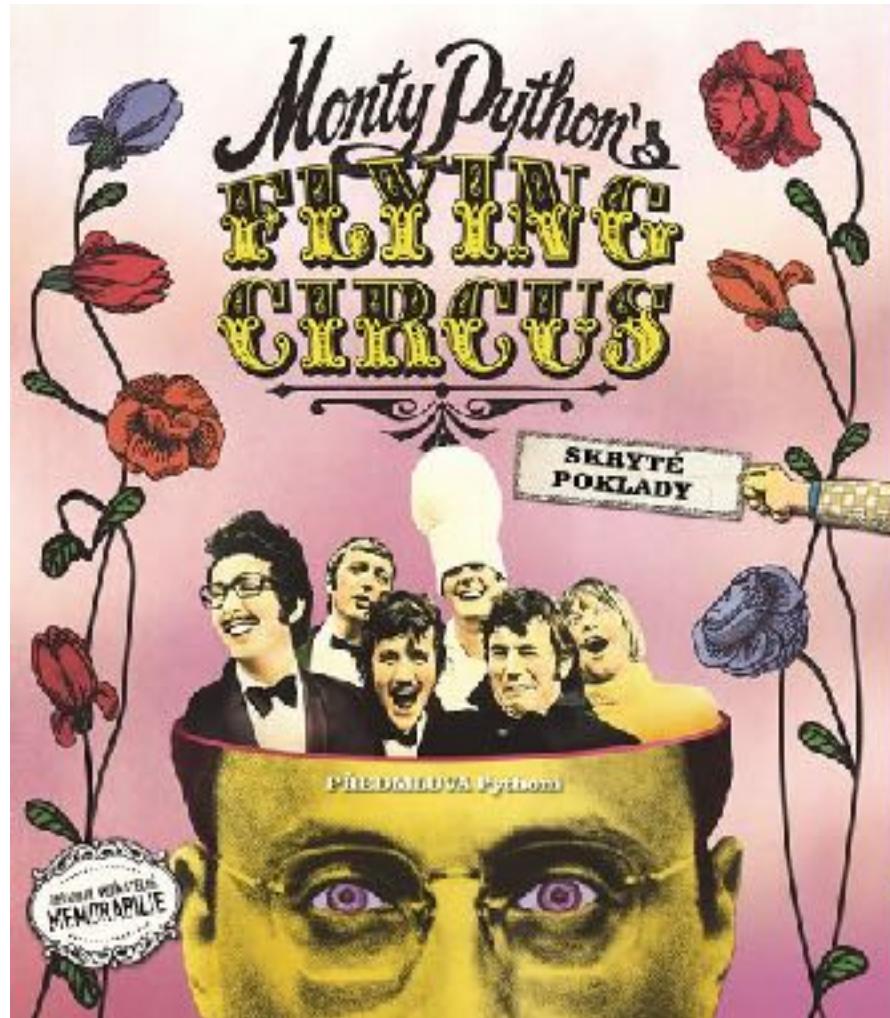
Python



Why the name Python?

Monty Python circus

BBC from 1969 -1974



Borrowed all the concept

1. Functional programing from C
 2. OOPs from C++
 3. Scripting from perl and shell script
 4. Modular features from Modula 3
-
1. Syntax from C and ABC Language

Uses of Python

1. Desktop Application
2. Web Application(Django)
3. Database Application
4. Network Application
5. Games
6. **Data Analysis(Numpy, Pandas)**
7. **Machine learning(Scikit-learn)**
8. **AI(Tensorflow, caffe, Keras, etc..)**
9. IOT

Not good for mobile application

Which companies using python?

Who all are using Python?



YAHOO!

Google

YouTube



BitTorrent

IBM

Dropbox



CANONICAL

NETFLIX

Quora



and the list goes on...

Features of Python

1. Simple and easy to learn
2. Freeware and open source
3. Platform independent
4. Portability
5. Dynamically typed
6. Both procedure oriented and object oriented
7. Interpreted
8. Extensible (Native support)
9. Extensive Library

Flavours of Python

1. CPython - Cython
2. JPython - Jython
3. IronPython
4. PyPy - Performance using JIT
5. RubyPthon
6. Anaconda
7. Stackless(Python for concurrency)

Version of Python

1. Python 1 introduced in Jan 1994
2. Python 2.0 introduced in Oct 2000
3. Python 3.0 introduced in Dec 2008

Any new version should provide support for old version program
Exception in python

Python 3 is not backward compatibility with Python 2

Python 2 support is not supported after 2020

Disadvantages of Python

1. Weak in mobile computing
2. Get slow in speed
3. Run time errors
4. Underdeveloped Database access layers

Basic Statistics

Synopsis

- 1. Statistics**
- 2. Descriptive and Inferential Statistics**
- 3. Data Types**
- 4. Summarizing the Data**
- 5. Frequency Distribution**

1. Statistics

Statistics consists of principles and method for

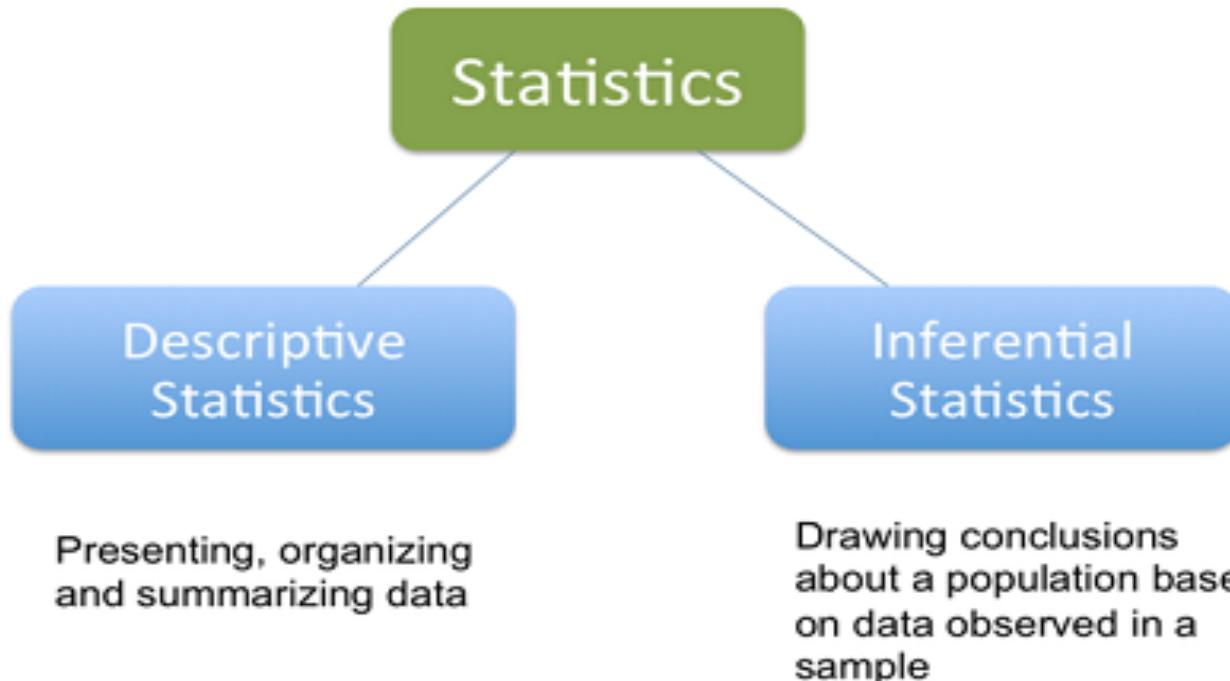
- 1. Collecting Data**
- 2. Analyzing Data**
- 3. Interpreting/Explaining the results**
- 4. Presentation of Data**

1. Statistics

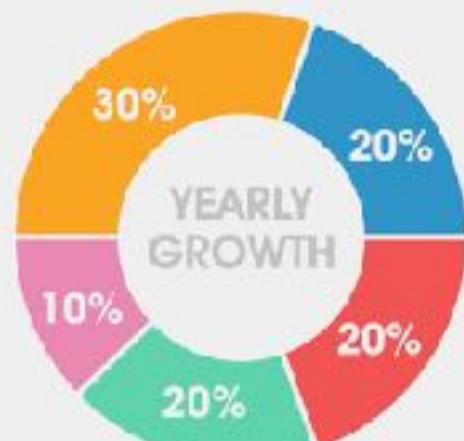
Statistics has been described as

1. Turning **Data** into **information**
2. Data Based **decision** making

2. Descriptive and Inferential Statistics



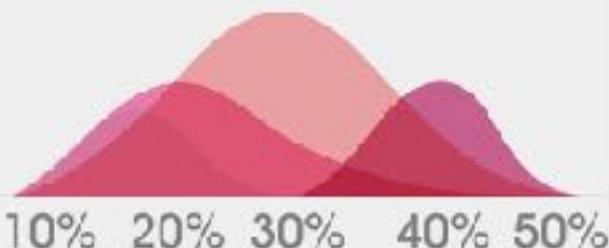
2. Descriptive and Inferential Statistics



DESCRIPTIVE STATISTICS
described data.

INFERENTIAL STATISTICS

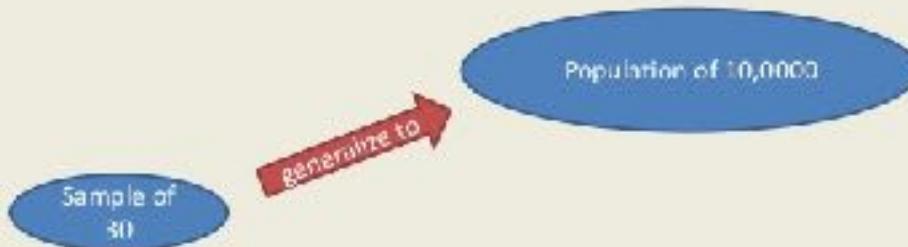
studies a sample
of the same data.



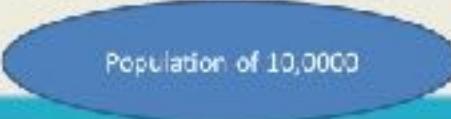
2. Descriptive and Inferential Statistics

Quick Reminder: Inferential vs. Descriptive

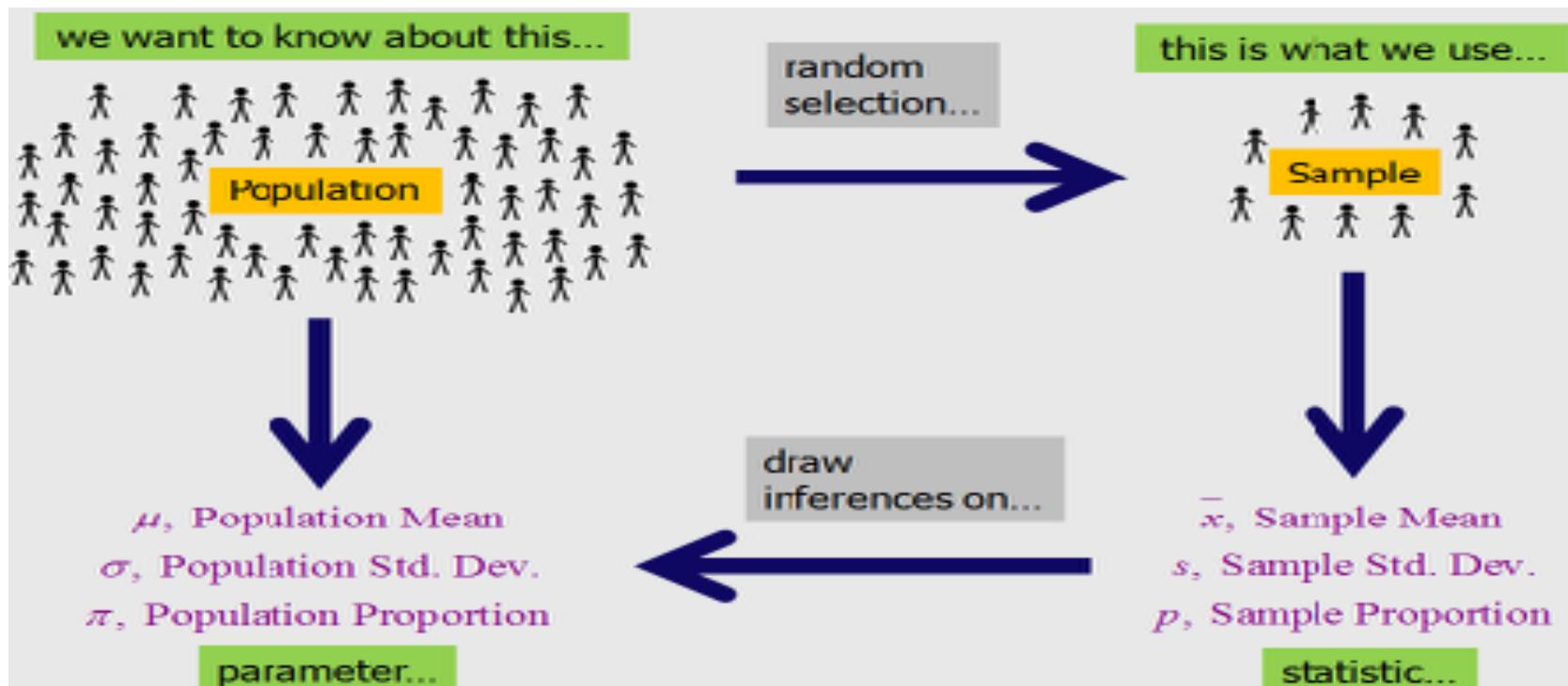
Inferential statistics generalize information from a random sample to a population.



Descriptive statistics do not generalize because they deal with the population itself.



Population and Sample



Reference : <http://www.dxbydt.com/the-size-of-your-sample/>

Census and Survey

Census:

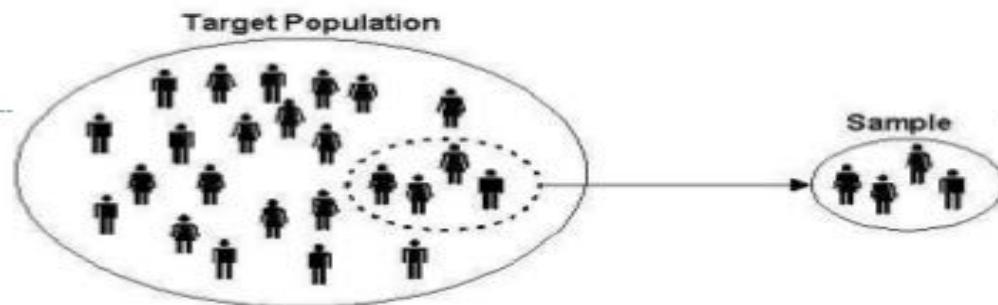
Gathering data from the whole population of interest. For example, elections, 10-year census, etc.

Survey:

Gathering data from the sample in order to make conclusions about the population.

For example, opinion polls, quality control checks in manufacturing units, etc.

WHEN DATA IS TO BE COLLECTED FROM EACH MEMBER OF THE POPULATION, IT IS KNOWN AS CENSUS SURVEY



WHEN DATA IS TO BE COLLECTED ONLY FROM SOME MEMBERS OF THE POPULATION, IT IS KNOWN AS SAMPLE SURVEY

Parameter and Statistic

Parameter: A descriptive measure of the population.

For example, population mean, population variance, population standard deviation, etc.

Statistic: A descriptive measure of the sample.

For example, sample mean, sample variance, sample standard deviation, etc.

Population Vs. Sample



Population of Interest



= 1 unit



Population ← Sample

Parameter ← Statistic

We measure the sample using statistics in order to draw inferences about the population and its parameters.

Identify Population Data or Sample Data?

- The US Government takes a census of its citizens every 10 years to gather information.
 - a) **Population**
 - b) Sample
- You want to know what sports teens prefer so you send out a survey to all the students in your high school.
 - a) Population
 - b) **Sample**
- You want data on the shoe size of all West students, so you interview every student at school.
 - a) **Population**
 - b) Sample

Identify as Parameter or Statistics?

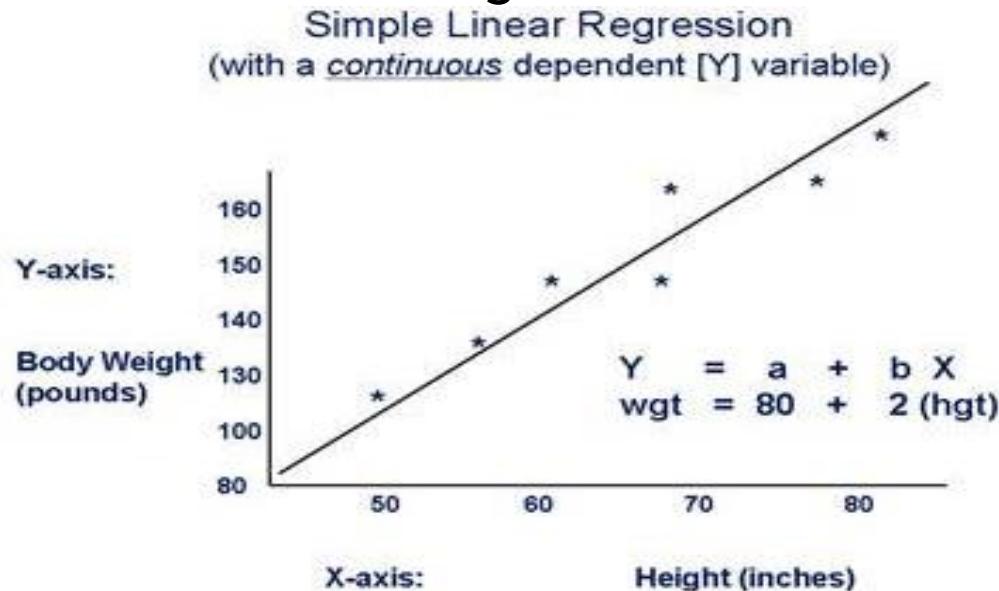
- You want to know the mean income of the people who subscribe to People magazine, so you question 100 subscribers.
 - a) Parameter b) Statistic
- You want to know the average height of the students in this math class, so you have everyone in the class write their height on a sheet of paper.
 - a) Parameter b) Statistic

Parameter and Statistics

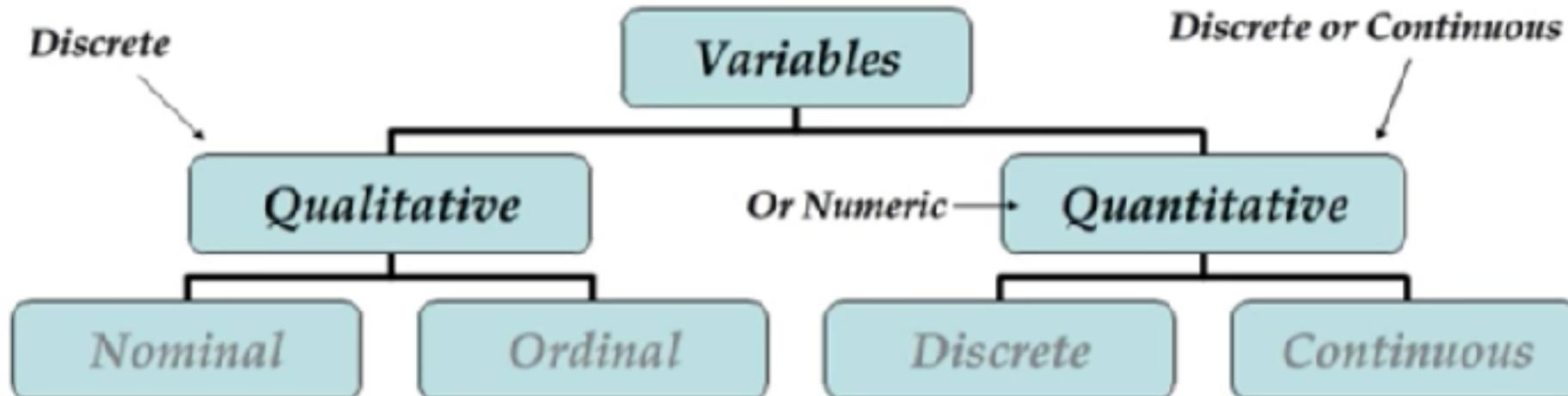
- Greek - Population Parameter
 - Mean - μ
 - Variance - σ^2
 - Standard Deviation - σ
- Roman - Sample Statistic
 - Mean - \bar{x}
 - Variance - s^2
 - Standard Deviation - s

Variables - Dependent and Independent

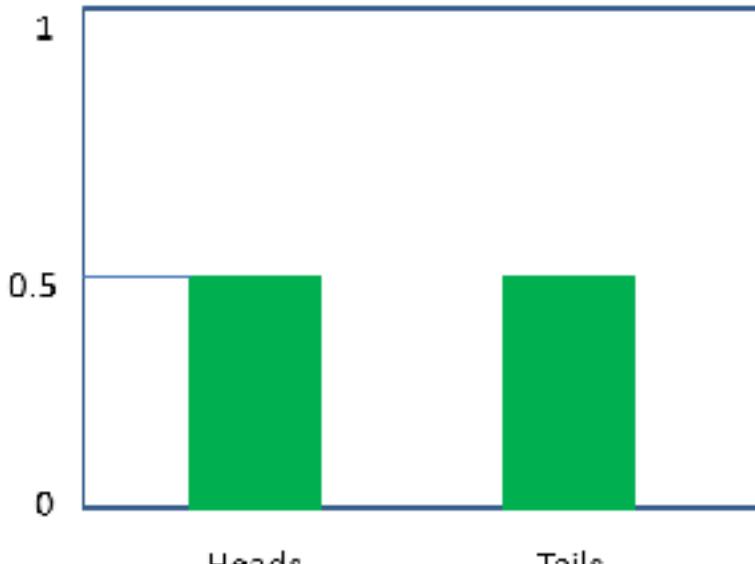
- Dependent variables on y-axis and Independent on x-axis.
- Dependent variable also called Target variable or Class variable.



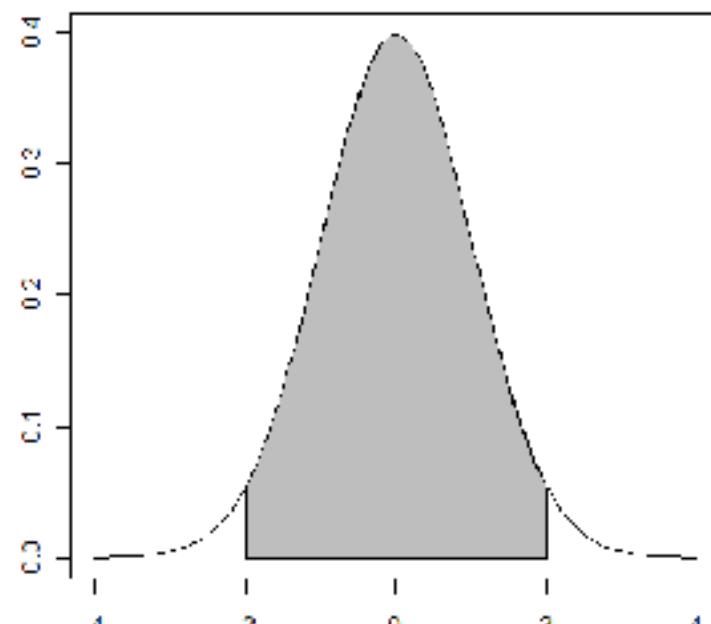
3. Data Types



Discrete and Continuous



Countable



Measurable

Numerical or Categorical?

Age	Gender	Major	Units	Housing	GPA
18	Male	Psychology	16	Dorm	3.6
21	Male	Nursing	15	Parents	3.1
20	Female	Business	16	Apartment	2.8

- Numerical
 - Age
 - Units
 - GPA
- Categorical
 - Gender
 - Major
 - Housing

Discrete or Continuous?

- Time between customer arrivals at a retail outlet
Continuous
- Sampling 100 voters in an exit poll and determining how many voted for the winning candidate
Discrete
- Lengths of newly designed automobiles -
Continuous
- No. of customers arriving at a retail outlet during a five- minute period
Discrete
- No. of defects in a batch of 50 items
Discrete

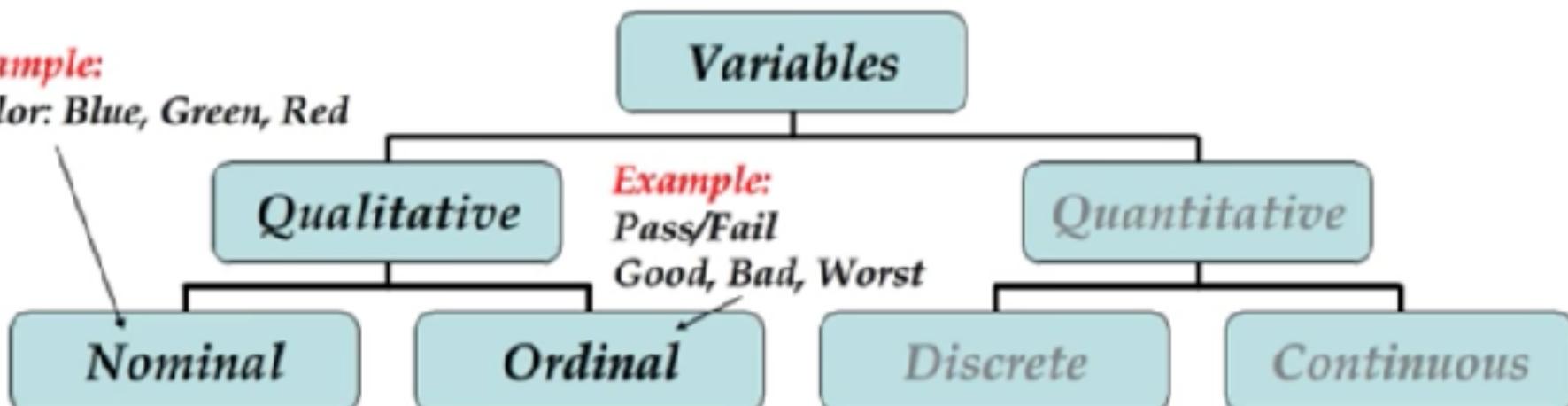
Numeric Data (Quantitative) Examples

- Height
- Weight
- Time
- Volume
- Number of iPads sold
- Number of complaints received at the call centre
- Number of employees
- Percentage return on a stock
- Rupee change in stock price

3. Data Types

Example:

Color: Blue, Green, Red



Categorical Data (Qualitative)

Nominal Examples

- Employee ID
- Gender
- Religion
- Ethnicity
- Pin codes
- Place of birth
- Aadhaar numbers

Ordinal Examples

- Mutual fund risk ratings
- Fortune 50 rankings

- Movie ratings

While there is an order, difference between consecutive levels are not always equal.

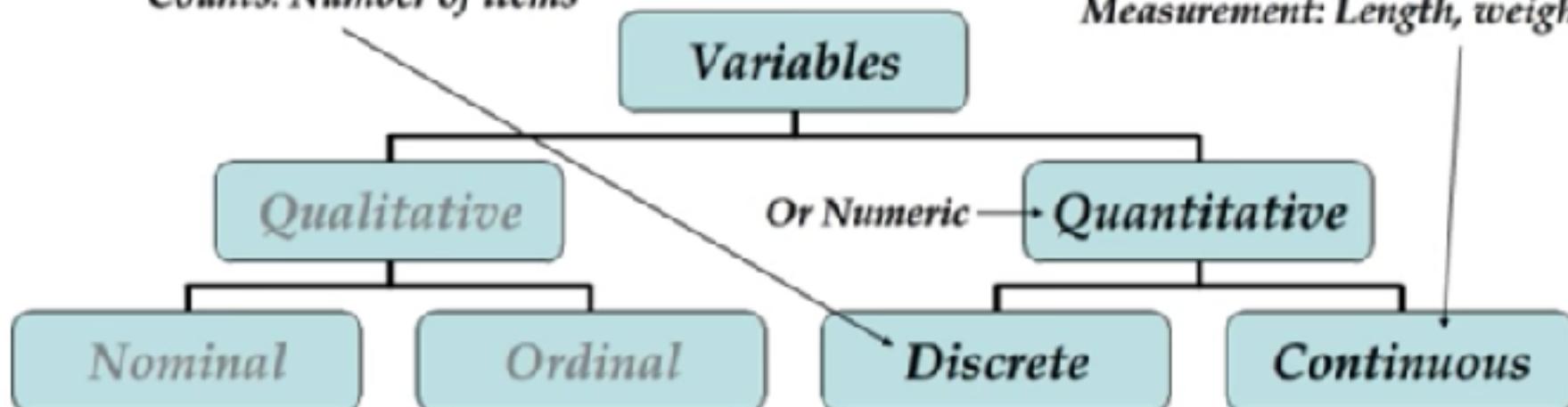
3. Data Types

Example:

Counts: Number of items

Example:

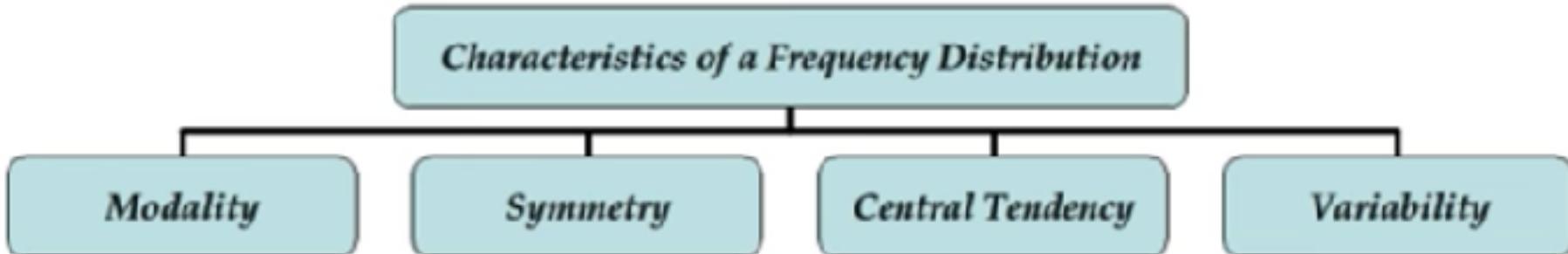
Measurement: Length, weight



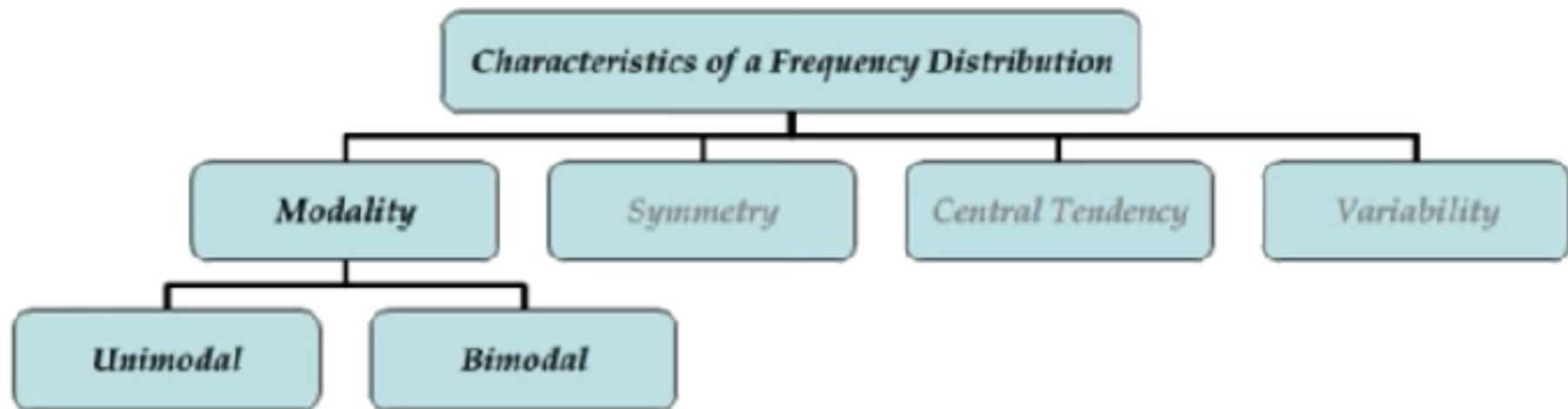
3. Summarizing the Data

1. Frequency Distribution
2. Bar Chart
3. Histograms

3. Summarizing the Data



3. Summarizing the Data



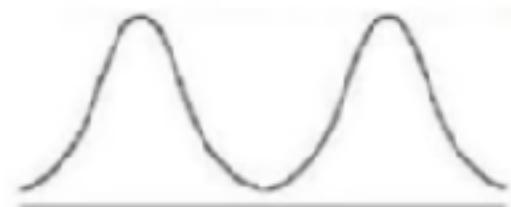
3. Summarizing the Data

Characteristics of a Frequency Distribution

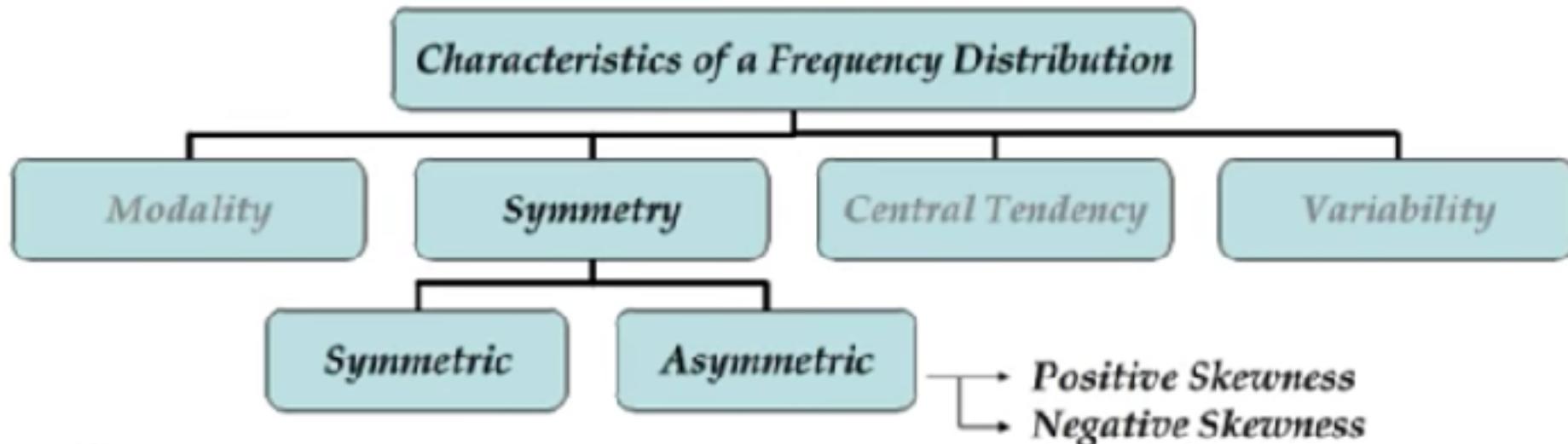
Modality

Unimodal

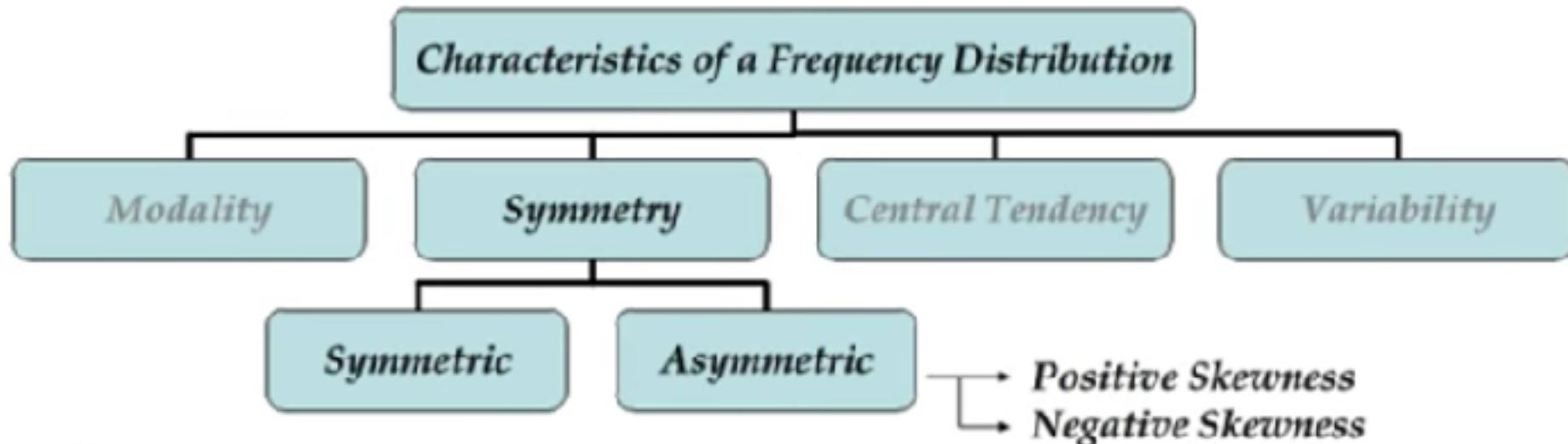
Bimodal



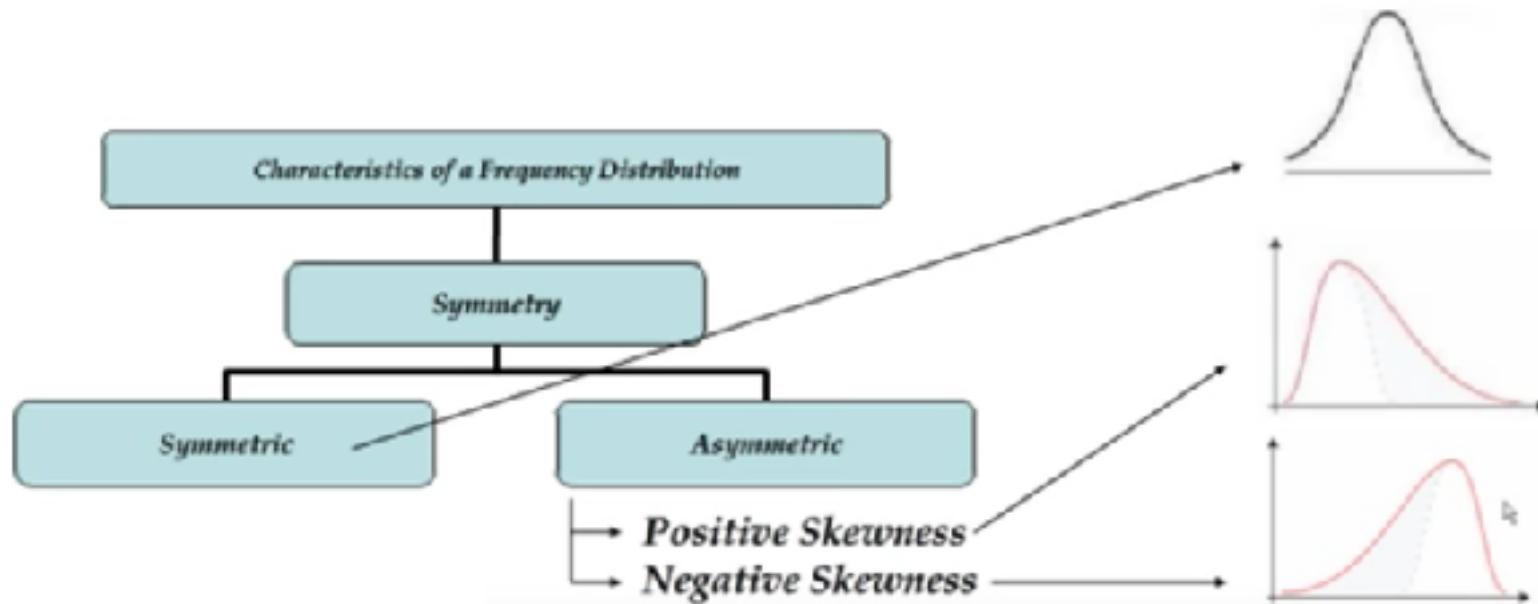
3. Summarizing the Data



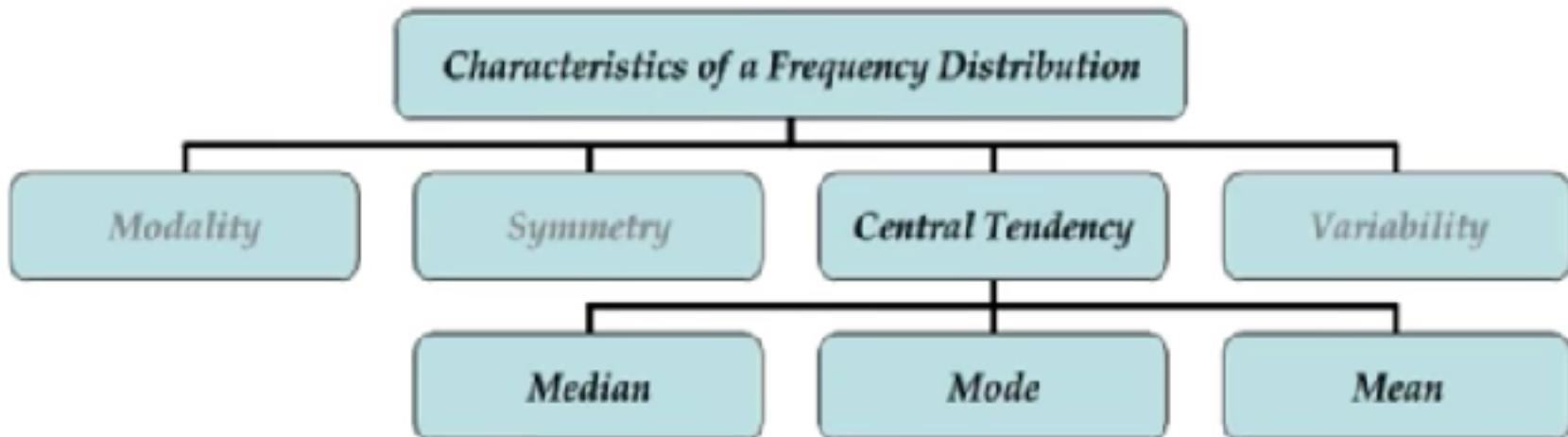
3. Summarizing the Data



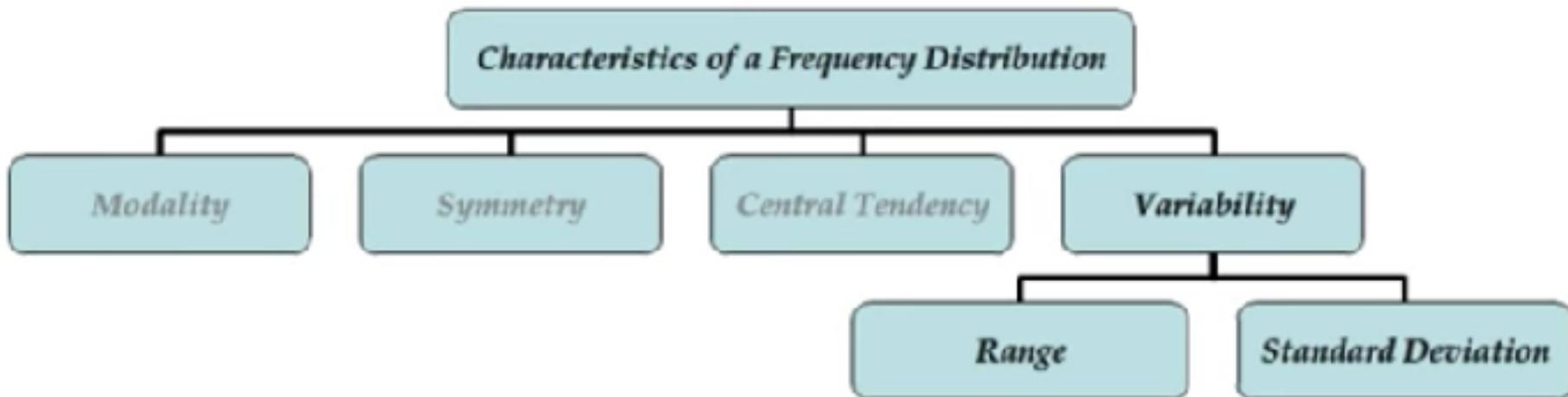
3. Summarizing the Data



3. Summarizing the Data



3. Summarizing the Data



Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37

Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37
SUM	351	351

Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37
SUM	351	351
MEAN	39	39

Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37
SUM	351	351
MEAN	39	39
MEDIAN	40	40

Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37
SUM	351	351
MEAN	39	39
MEDIAN	40	40
RANGE	120	23

Standard Deviation

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{n}}$$

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n - 1}}$$

S.No	x	x^2	x-mean	Abs(x-mean)	(x-mean)^2
1	5	25	0.3333333333	0.3333333333	0.1111111111
2	7	49	2.3333333333	2.3333333333	5.4444444444
3	4	16	-0.6666666667	0.6666666667	0.4444444444
4	2	4	-2.6666666667	2.6666666667	7.1111111111
5	6	36	1.3333333333	1.3333333333	1.7777777778
6	2	4	-2.6666666667	2.6666666667	7.1111111111
7	8	64	3.3333333333	3.3333333333	11.1111111111
8	5	25	0.3333333333	0.3333333333	0.1111111111
9	3	9	-1.6666666667	1.6666666667	2.7777777778
SUM	42	232	0	15.333333333	36
Average	4.6666666667	25.77777778			

Match	Player A	Player B
1	40	40
2	40	35
3	7	45
4	40	52
5	0	30
6	90	40
7	3	29
8	11	43
9	120	37
SUM	351	351
MEAN	39	39
MEDIAN	40	40
STANDARD DEVIATION	41.5180683558376	7.28010988928052

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

c

Points scored per game	7	9	10	11	13
Frequency, f	1	2	4	2	1

Points scored per game	3	6	7	10	11	13	30
Frequency, f	2	1	2	3	1	1	1

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

Points scored per game

7	9	10	11	13	
Frequency, f	1	2	4	2	1

Points scored per game	3	6	7	10	11	13	30
Frequency, f	2	1	2	3	1	1	1

MEAN = ? MEDIAN = ? MODE = ?

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

Points scored per game

7	9	10	11	13	
Frequency, f	1	2	4	2	1

Points scored per game	3	6	7	10	11	13	30
Frequency, f	2	1	2	3	1	1	1

$$\text{MEAN} = \text{MEDIAN} = \text{MODE} = 10$$

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

Points scored per game

7	9	10	11	13	
Frequency, f	1	2	4	2	1

Points scored per game	3	6	7	10	11	13	30
Frequency, f	2	1	2	3	1	1	1

$$\text{MEAN} = \text{MEDIAN} = \text{MODE} = 10 \quad \text{RANGE} = 5, 5, 27$$

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

Points scored per game

7	9	10	11	13	
Frequency, f	1	2	4	2	1

Points scored per game	3	6	7	10	11	13	30
Frequency, f	2	1	2	3	1	1	1

MEAN = MEDIAN = MODE = 10 RANGE = 5 , 5 , 27 Reject Player 3

Basketball coach Statson is in a dilemma choosing between 3 players all having the same average scores.

Points scored per game	7	8	9	10	11	12	13
Frequency, f	1	1	2	2	2	1	1

□

Points scored per game	7	9	10	11	13
Frequency, f	1	2	4	2	1

STANDARD DEVIATION

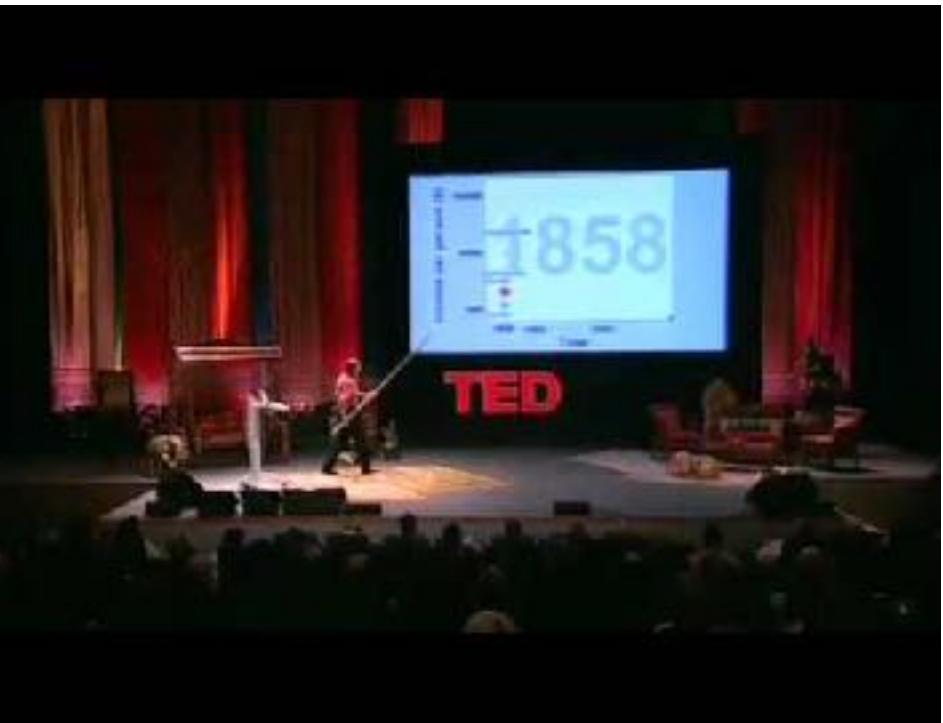
Player 1 = 1.7873008824606

Player 2 = 3.30823887354653

What is your Decision?????????

CONSISTENCY INDEX: TOP EIGHT RUN-SCORERS

Player	For	Ci	Sd	Ave	M	I	No	Runs
Ricky Ponting	Aus	1.05	59.47	56.88	128	215	26	10750
Sunil Gavaskar	Ind	1.06	54.42	51.12	125	214	16	10122
Allan Border	Aus	1.08	54.45	50.37	156	265	44	11174
Rahul Dravid	Ind	1.17	61.07	52.28	131	227	26	10509
Sachin Tendulkar	Ind	1.18	64.28	54.28	156	256	27	12429
Jacques Kallis	Sa	1.19	64.91	54.58	128	216	33	9988
Brian Lara	Wi	1.24	65.33	52.89	131	232	6	11953
Steve Waugh	Aus	1.26	64.16	51.06	168	260	46	10927



Data Preprocessing

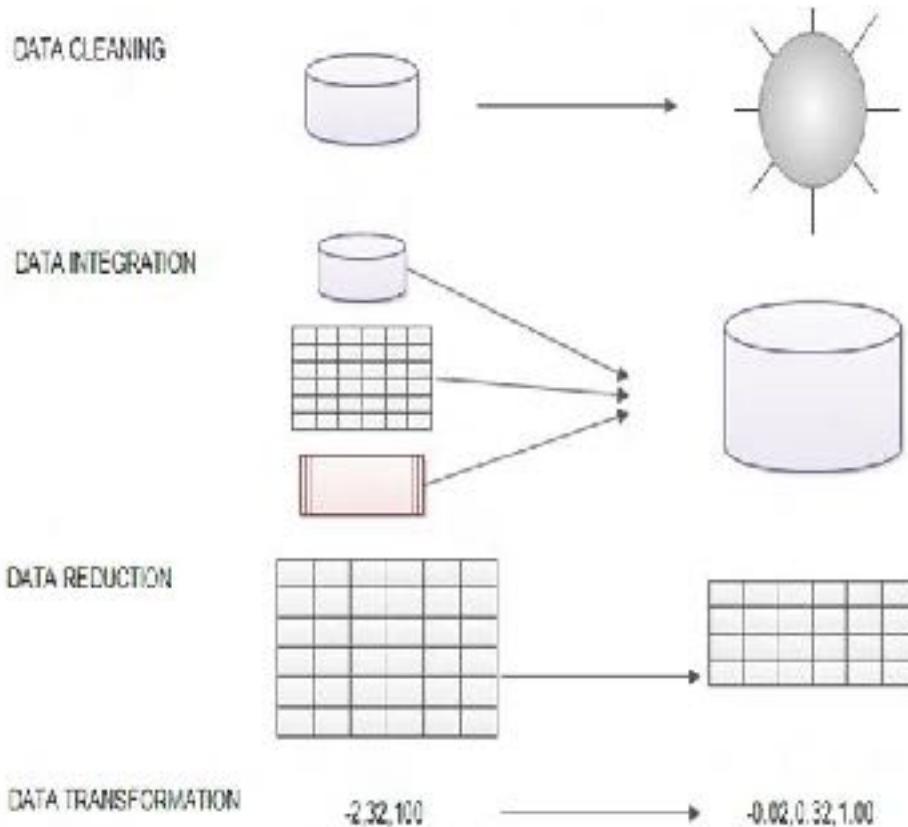
Real World Data

Any Problem?

S.No	Credit_rating	Age	Income	Credit_cards
1	0.00	21	10000	y
2	1.0		2500	n
3	2.0	62	-500	y
4	100.012	42		n
5	yes	200	1	y
6	30	0	Seventy thousand	No

Data Preprocessing

- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation



Data Cleaning

1. Missing Data
 - Central Imputation
 - KNN Imputation
- 2. Noisy Data
- Smoothing
- Clustering
1. Outlier Removal
- Using Boxplot

The diagram illustrates a data cleaning process. It starts with a 'raw' table at the top and a 'cleaned' table at the bottom. Blue arrows point from the raw table to the cleaned table, indicating the transformation steps. The raw table has columns: company name, furigana, postal code, address, and telephone number. The cleaned table has columns: company name, juridical personality, furigana, postal code, all prefectures, address, and telephone number.

company name	furigana	postal code	address		telephone number
AlphaPurchase Co., Ltd	Alpha Purchase	107-0061	Aoyama Building 12-th Floor, 1-2-3, Kita-Aoyama, Minato-ku, Tokyo		03-5772-7801
AAA Foundation	AAA	1500002	Kami-meguro, Meguro-ku X-X-X		0312345678
BBBB, Inc.	BBBB	123	Minami-Azabu, Minato-ku XX-1-1		03(1234)9876

company name	juridical personality	furigana	postal code	all prefectures	address	telephone number
Alpha Purchase	Co., Ltd	Alpha Purchase	1070 061	Tokyo	Aoyama Building 12th floor, 1-2-3, Kita-Aoyama, Minato-ku	0357727801
AAA	Foundation	AAA	1500 002	Tokyo	Kami-meguro, Meguro-ku X-X-X	0312345678
BBBB	Inc.	BBBB	1230 001	Tokyo	Minami-Azabu, Minato-ku XX-1-1	0312349876

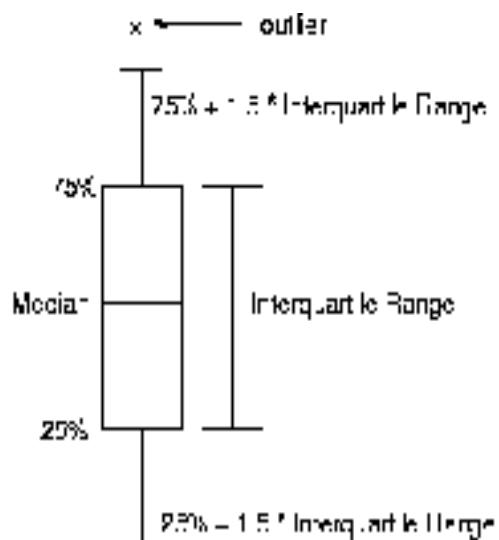
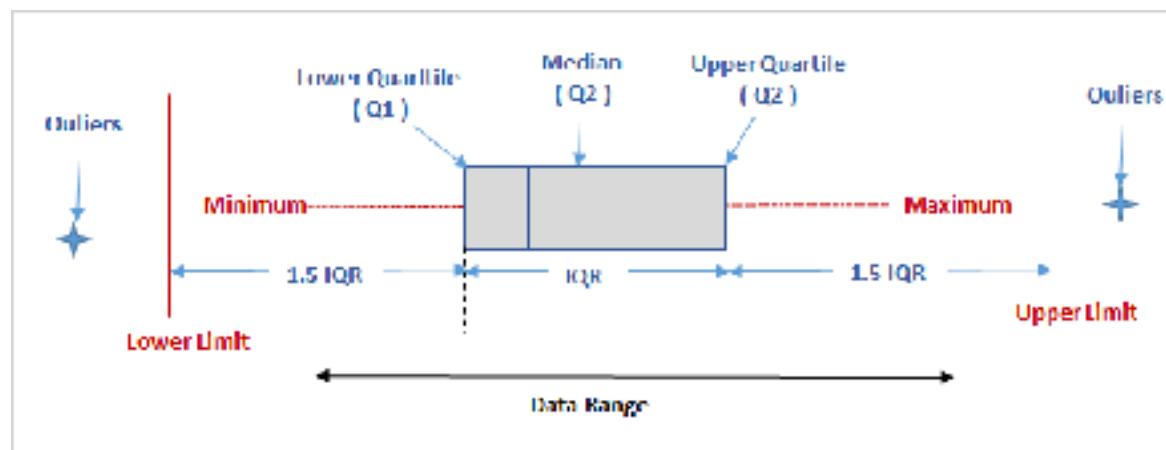
Imputation

- Replace with mean or a median
- When to use mean?
- Replace with nearest neighbour
- How much nearest to see?

S.No	Qualification	Age	Income
1	B.Tech	25	30k
2	M.Tech	30	50k
3	B.Tech	26	32k
4	B.Tech	25	?
5	M.Tech	29	60k
6	B.Tech	?	30k

Outlier

- BoxPlot



Data Transformation

- Normalization

Min-max normalization

1. Min Max Normalization
 2. Z - Score Normalization
 3. Decimal scaling

Decimal scaling

$$v = v / 10^j$$

Normalization: Example II

- Min-Max normalization on an employee database

- max difference for salary: $100000 - 15000 = 85000$
 - max difference for age: $52 - 25 = 27$
 - New rule for age and salary = 0; new max storage and salary = 1

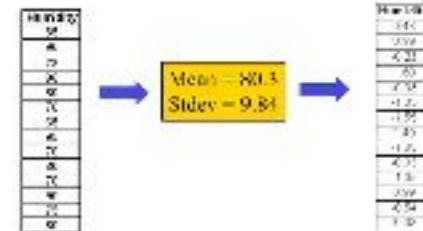
$$x_i^* = \frac{x_i - \min x_i}{\max x_i - \min x_i} (\text{newmax} - \text{newmin}) + \text{newmin}$$

ID	Gender	Age	Salary
1	F	27	13,000
2	M	5	64,000
3	M	52	103,000
4	F	33	25,000
5	M	46	45,000

ID	Gender	Age	Salary
1	1	20.0	200
2	0	25.0	250
3	0	10.0	100
4	1	22.0	220
5	0	22.0	220

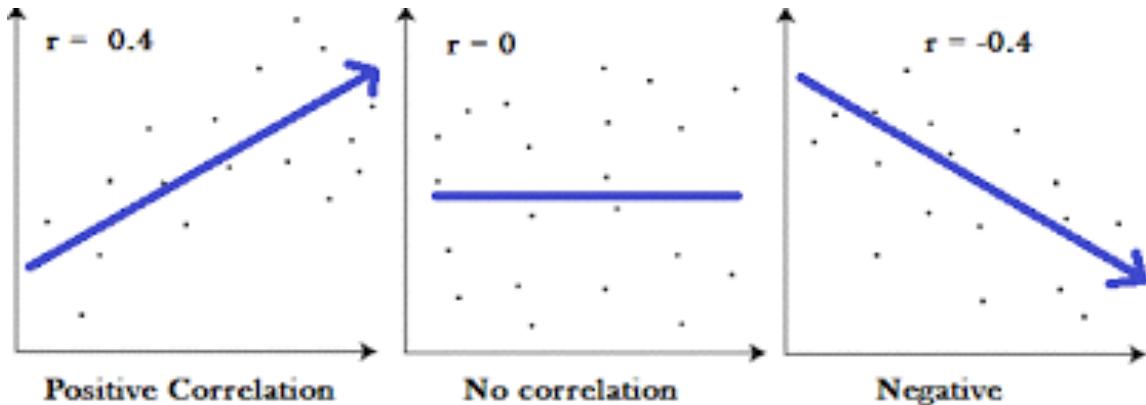
Normalization: Example

- * z-score normalization: $v' = (v - \text{Mean}) / \text{Stdev}$
 - * Example: normalizing the "Humidity" attribute:



Data Integration

- Check for correlation
- Remove uncorrelated data



$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

Data Reduction

- Data Cube Aggregation

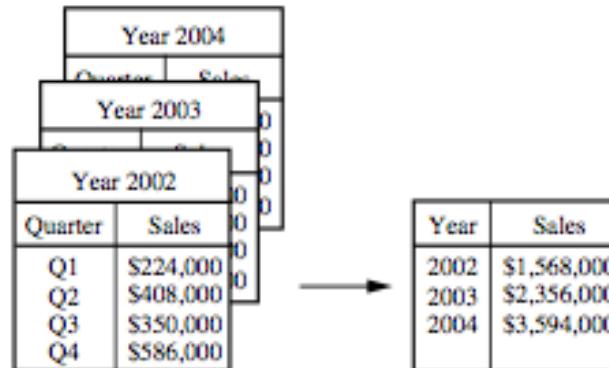


Figure 2.13 Sales data for a given branch of *AllElectronics* for the years 2002 to 2004. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.

Relationship

$Y = \text{?????????}$

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17

Relationship

$$Y = 2 + 3(X)$$

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17

What is 2 here?

$$Y = 2 + 3(X)$$

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17

Find the Y in ?

$$Y = 2 + 3(X)$$

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	?
1	?

Value for Y with given X

$$Y = 2 + 3(X)$$

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	32
1	5

Terminology

$$Y = 2 + 3(X)$$

Y = Model

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	32
1	5

Terminology

$$Y = 2 + 3(X)$$

Y = Model

2 = Intercept

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	32
1	5

Terminology

$$Y = 2 + 3(X)$$

Y = Model

2 = Intercept

3 = Slope

X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	32
1	5

Terminology

$$Y = 2 + 3(X)$$

Y = Model

2 = Intercept

3 = Slope

X = input

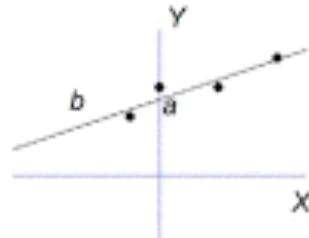
X	Y
2	8
6	20
4	14
3	11
7	23
4	14
2	8
5	17
10	32
1	5

Formula for a line

Linear regression equation
(without error)

$$\hat{Y} = bX + a$$

predicted values of Y b = slope = rate of predicted ↑/↓ for Y scores for each unit increase in X Y-intercept = level of Y when X is 0



$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Dependent Variable Population Y intercept Population Slope Coefficient Independent Variable Random Error term

Linear component Random Error component

Linear Regression

Welcome to the world of data science

What is linear?

What is linear?

A Straight line

What is Regression?

What is Regression?

Relationship between two points

What is Linear Regression?

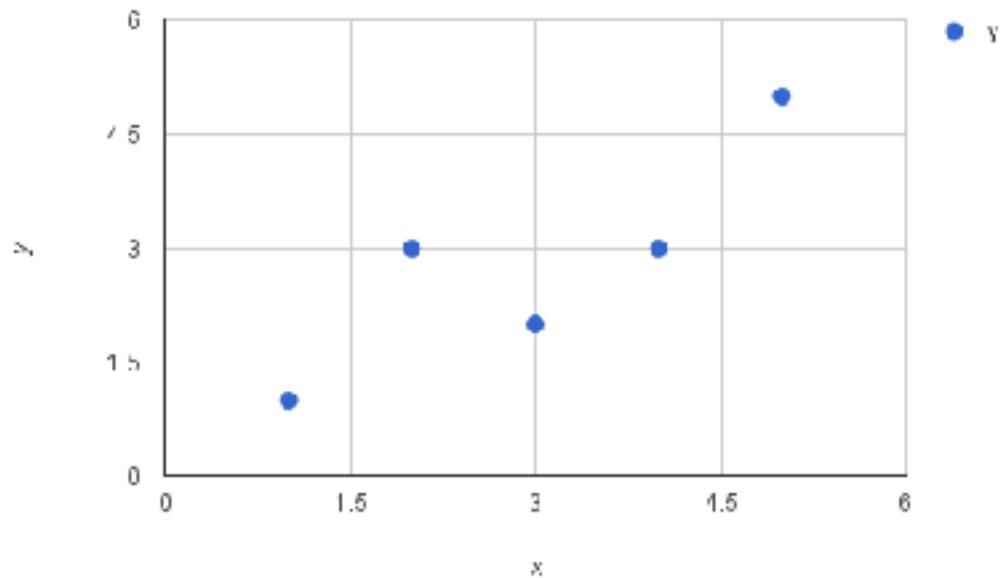
What is Linear Regression?

A Straight line that attempts to predict
the relationship between two points

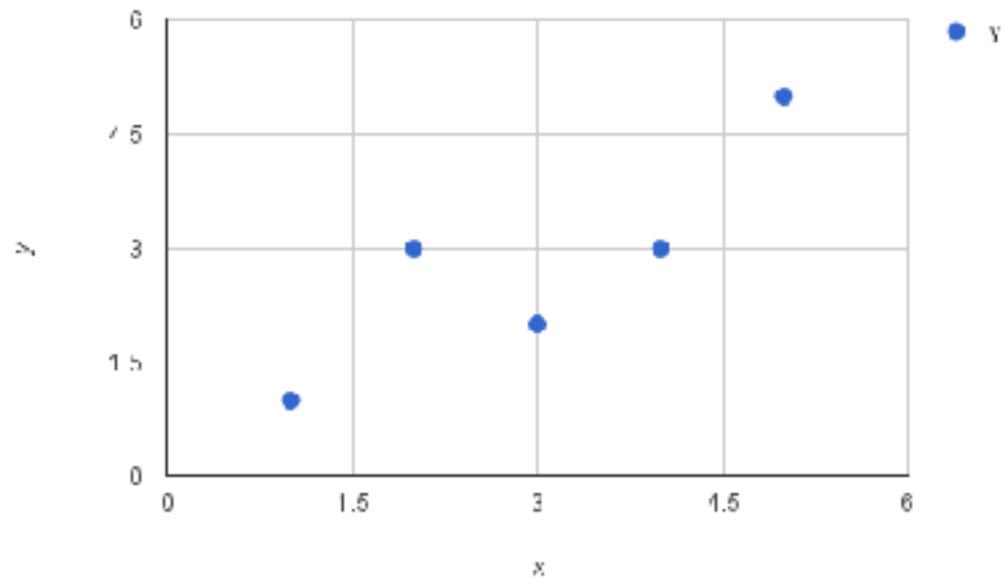
Help me in finding the relationship?

x	y
1	1
2	3
4	3
3	2
5	5

X versus Y



X versus Y



$$y = B_0 + B_1 * x$$

$$B1 = \text{sum}((xi - \text{mean}(x)) * (yi - \text{mean}(y))) / \text{sum}((xi - \text{mean}(x))^2)$$

$$B0 = \text{mean}(y) - B1 * \text{mean}(x)$$

x	mean(x)	x - mean(x)
1	3	-2
2	3	-1
4	3	1
3	3	0
5	3	2

$$B1 = 8 / 10$$

$$B1 = 0.8$$

$$B0 = \text{mean}(y) - B1 * \text{mean}(x)$$

or

$$B0 = 2.8 - 0.8 * 3$$

or

$$B0 = 0.4$$

$$y = B0 + B1 * x$$

or

$$y = 0.4 + 0.8 * x$$

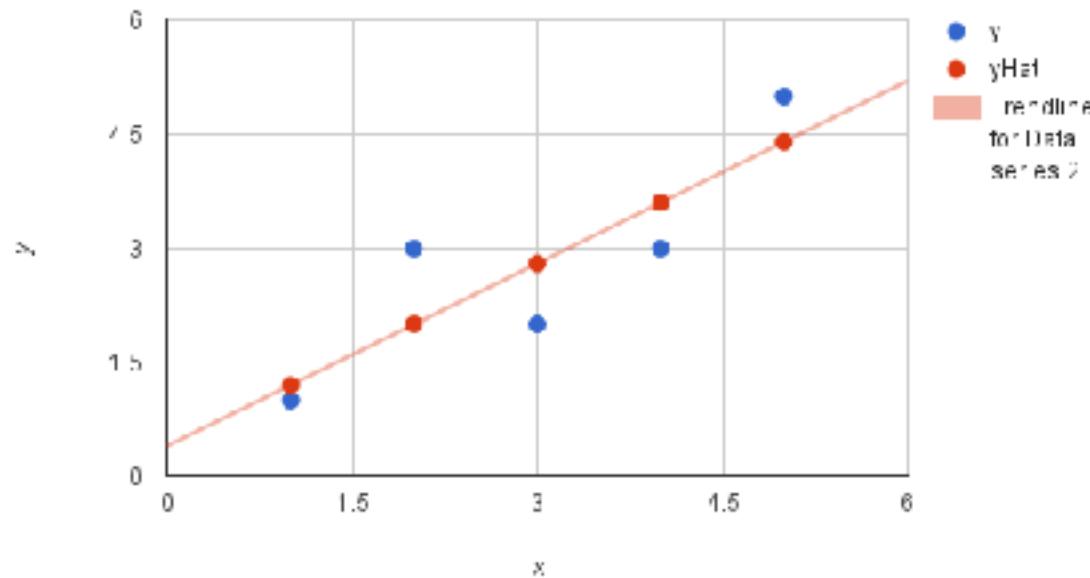
y	mean(y)	y - mean(y)
1	2.8	-1.8
3	2.8	0.2
3	2.8	0.2
2	2.8	-0.8
5	2.8	2.2

x - mean(x)	y - mean(y)	Multiplicati on
-2	-1.8	3.6
-1	0.2	-0.2
1	0.2	0.2
0	-0.8	0
2	2.2	4.4

x - mean(x)	squared
-2	4
-1	1
1	1
0	0
2	4

x	y	predicted y
1	1	1.2
2	3	2
4	3	3.6
3	2	2.8
5	5	4.4

x and y



RMSE = 0.692

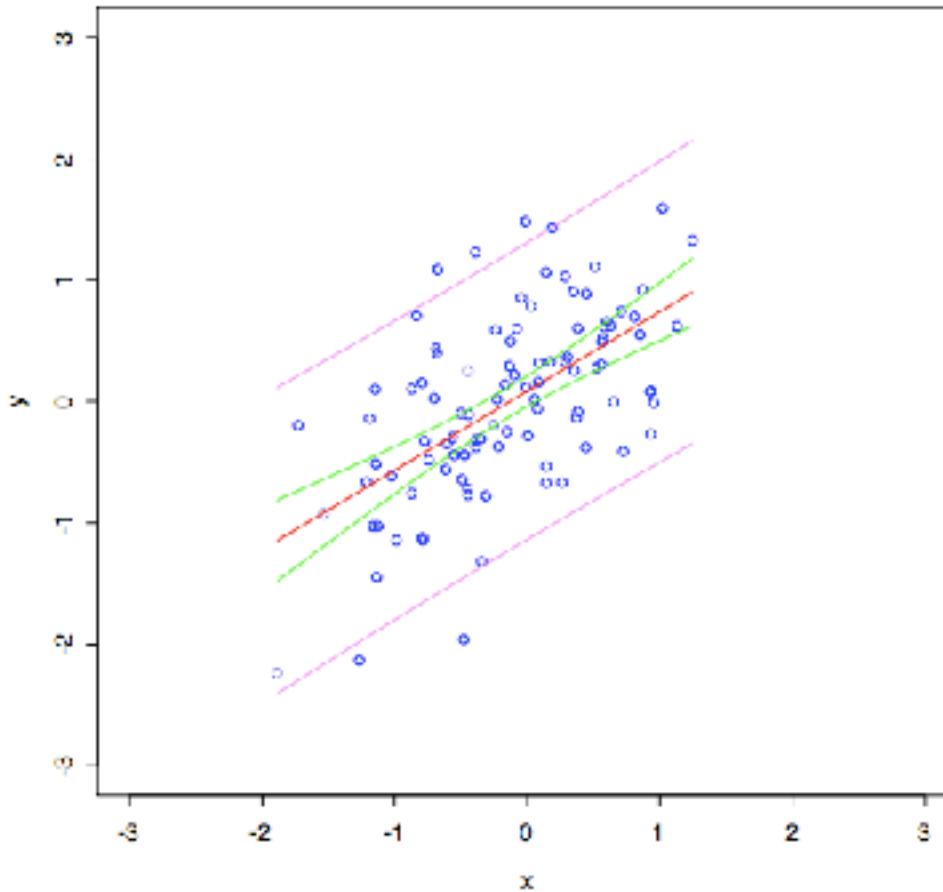
Gradient Descent

Finding the optimum relationship
where the error is minimal.

Finding the intercept and coefficients
value.

Find the solution?

Any Suggestions?

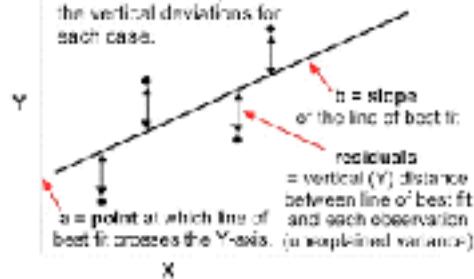


Line of best fit

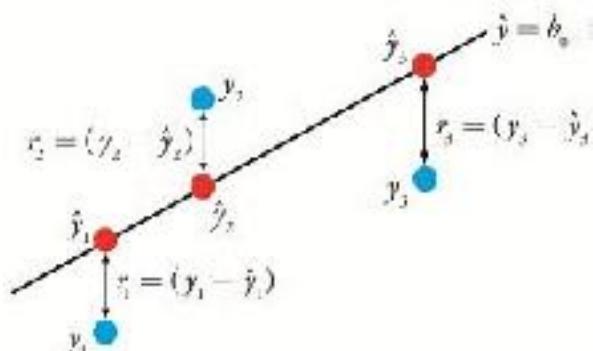
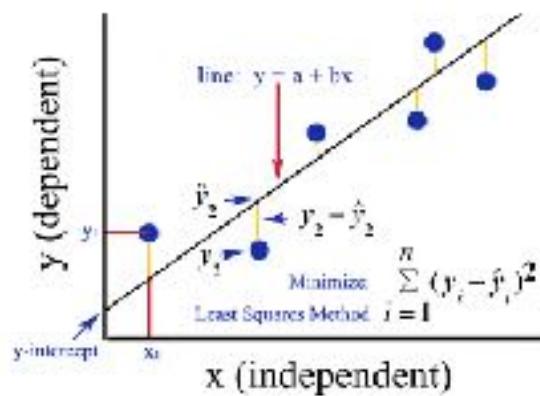
— — —
Ordinary least square line

Least squares criterion

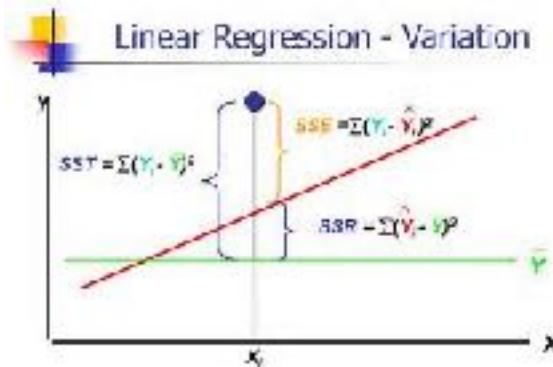
The line of best fit minimises
the total sum of squares of
the vertical deviations for
each case.



residuals
= vertical (y) distance
between line of best fit
and each observation
(unexplained variance)



Linear Regression - Variation



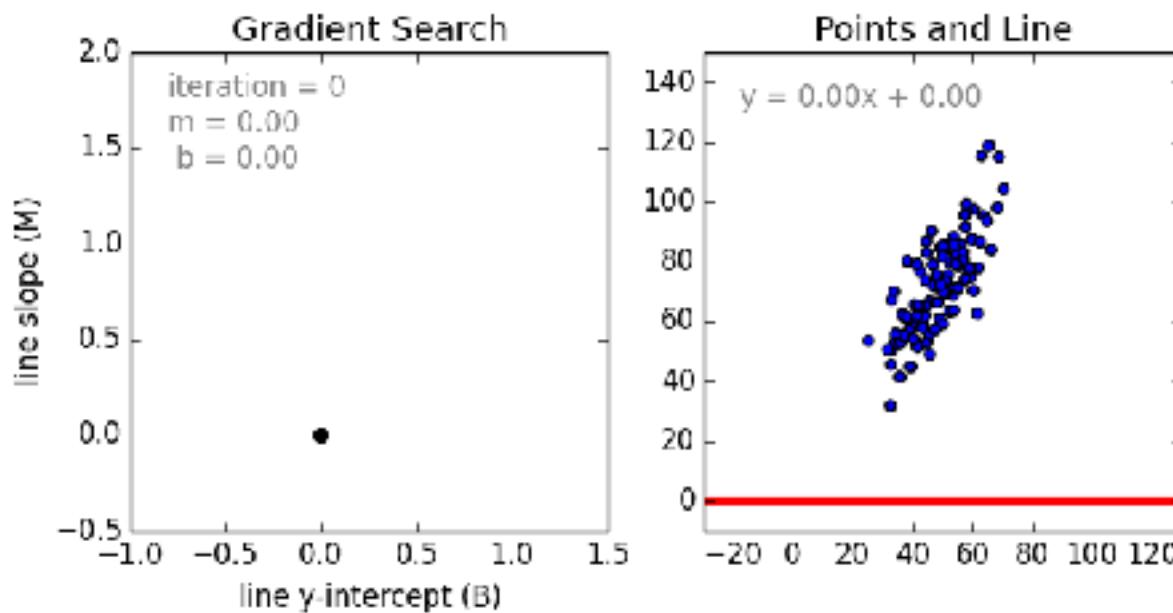
Cost Function

$$\text{Error}_{(m,b)} = \frac{1}{N} \sum_{i=1}^N (y_i - (mx_i + b))^2$$

Gradient Descent

Learning Rate

Momentum



Partial Derivative

Finding the direction of coefficient and slope moves in.

$$\frac{\partial}{\partial m} = \frac{2}{N} \sum_{i=1}^N -x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial b} = \frac{2}{N} \sum_{i=1}^N -(y_i - (mx_i + b))$$

Error Metrics

Mean squared error

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n e_t^2$$

Root mean squared error

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n e_t^2}$$

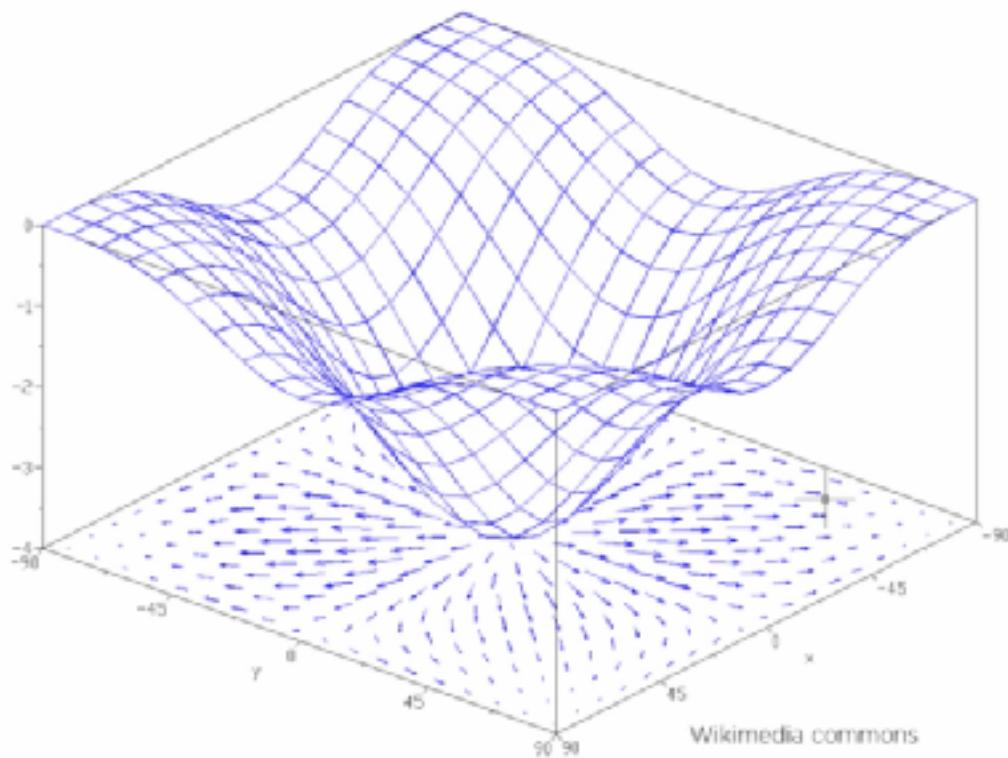
Mean absolute error

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |e_t|$$

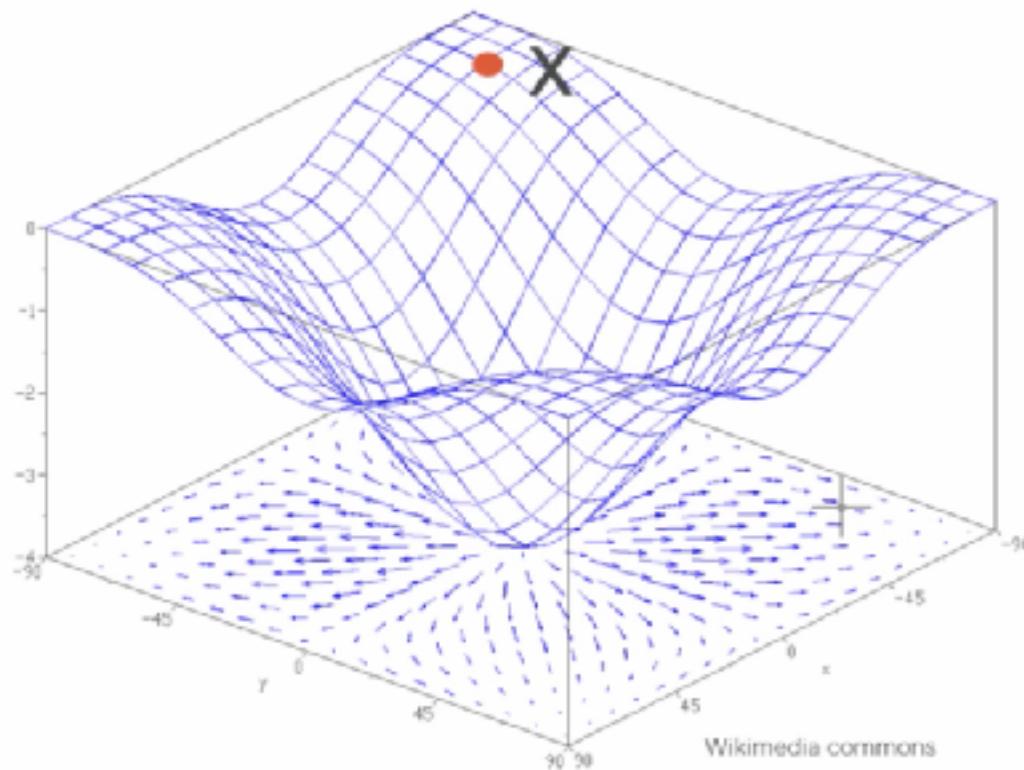
Mean absolute percentage error

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{e_t}{y_t} \right|$$

Gradient Descent

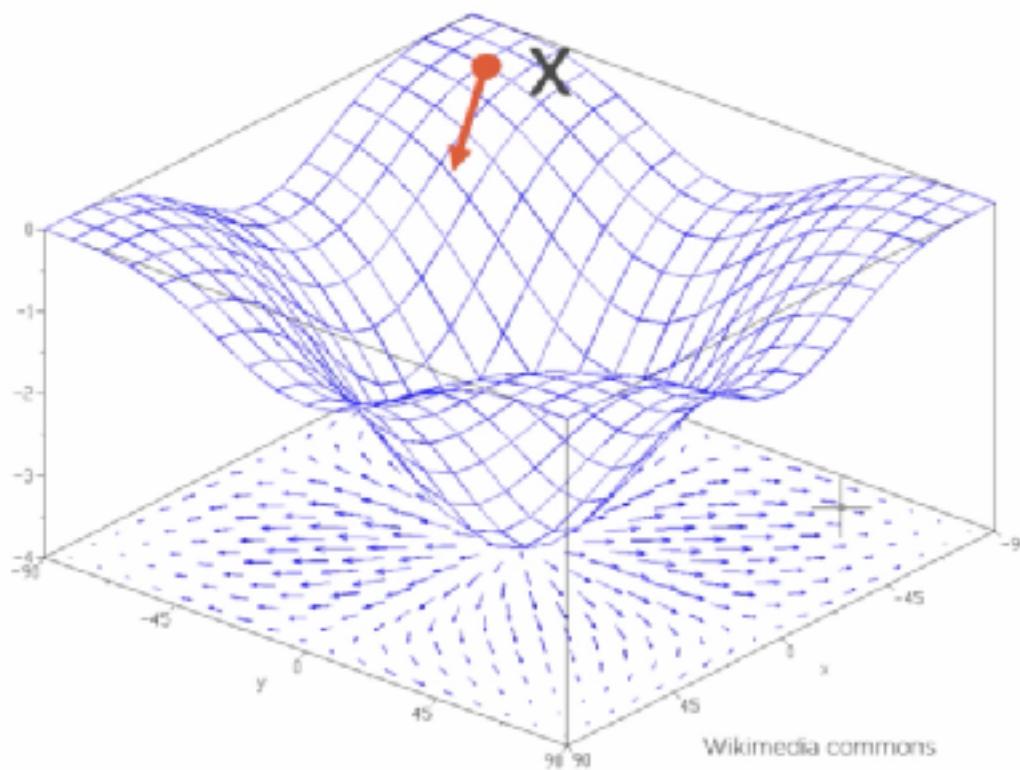


Gradient Descent

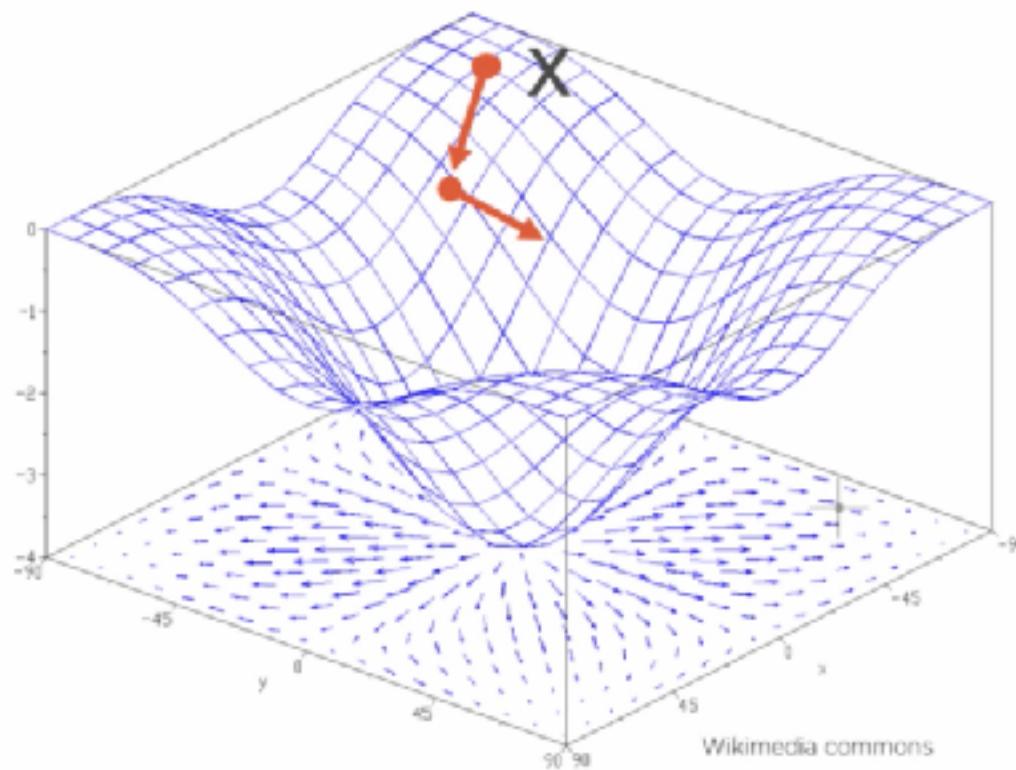


Wikimedia commons

Gradient Descent

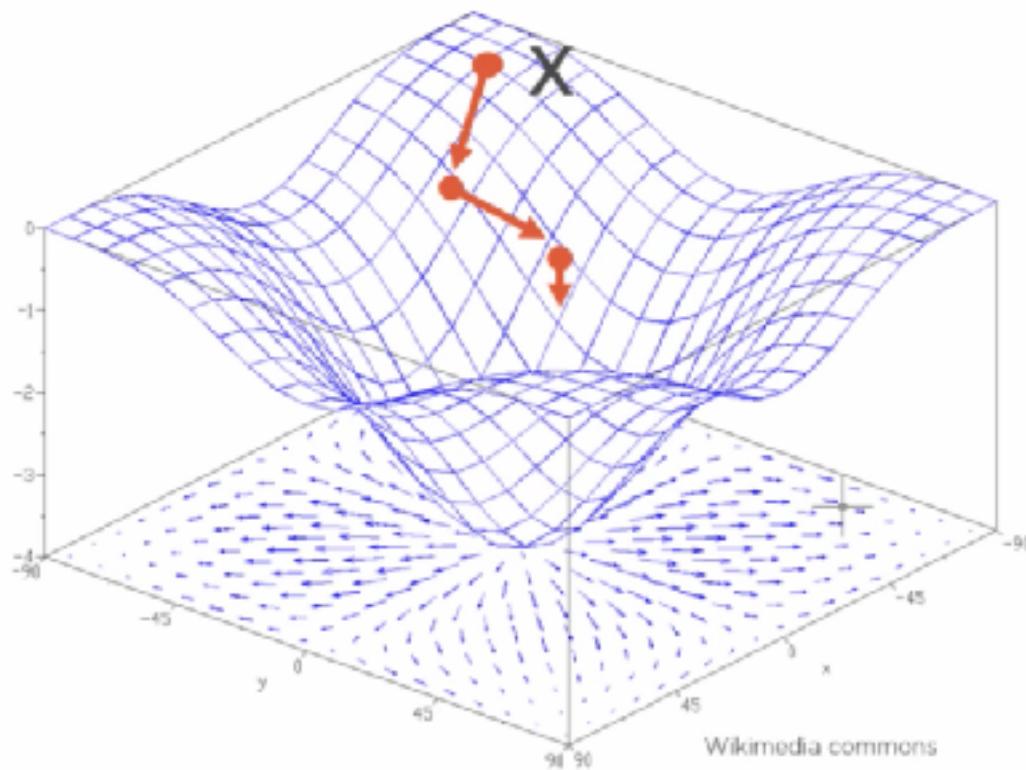


Gradient Descent



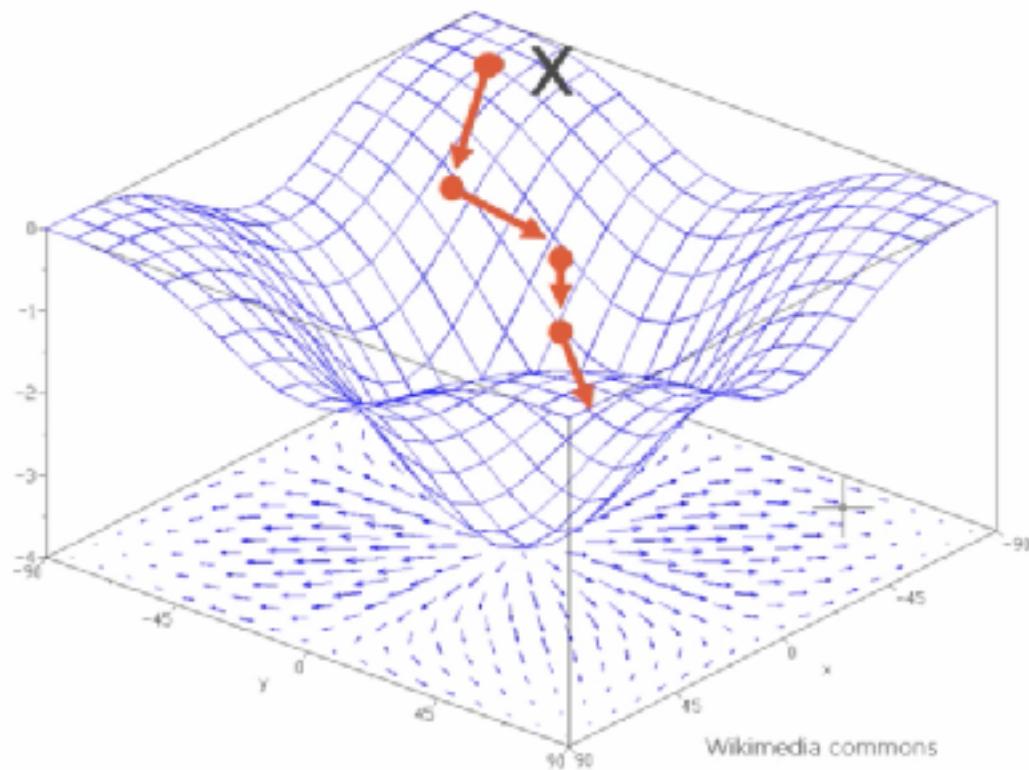
Wikimedia commons

Gradient Descent



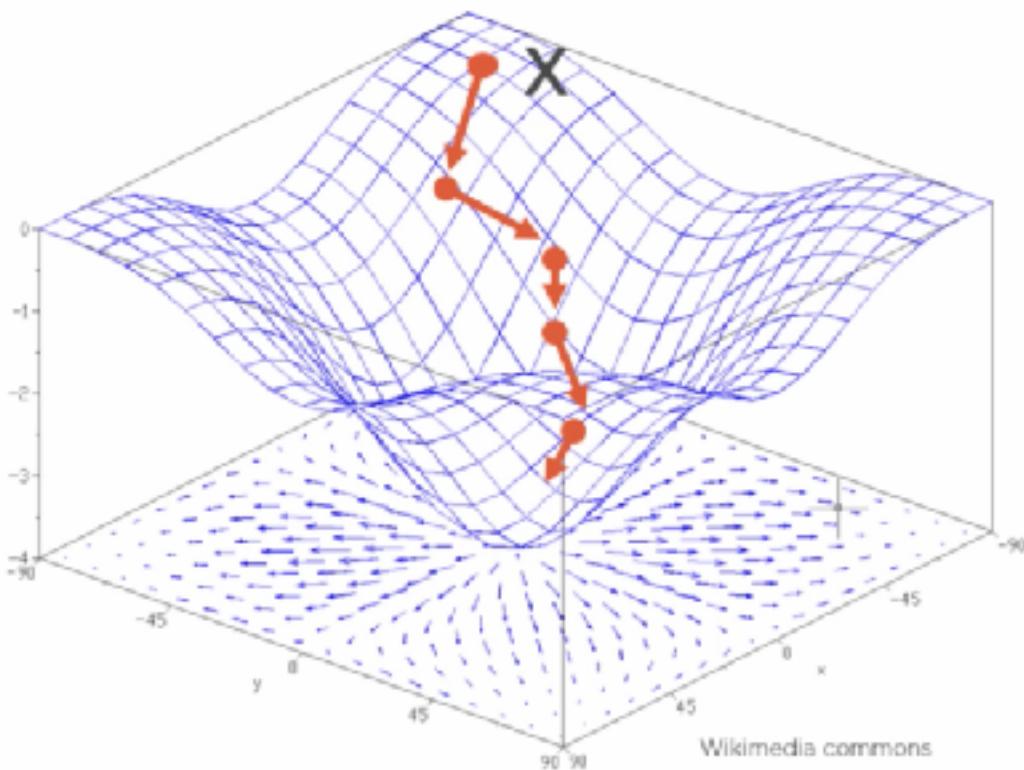
Wikimedia commons

Gradient Descent



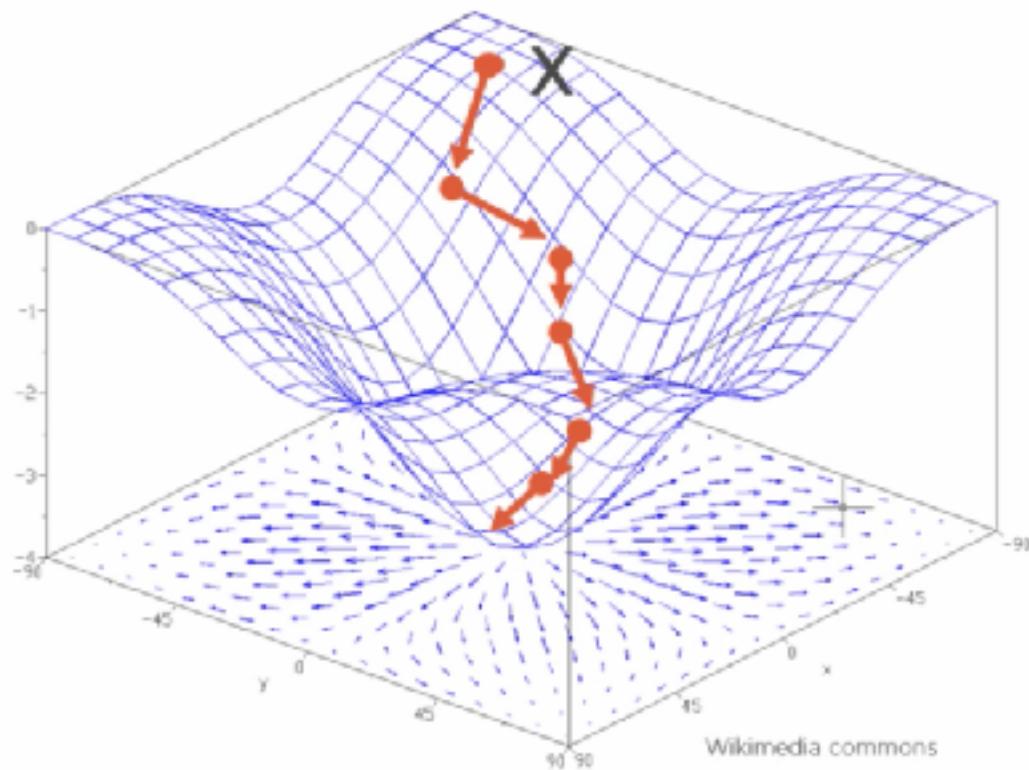
Wikimedia commons

Gradient Descent

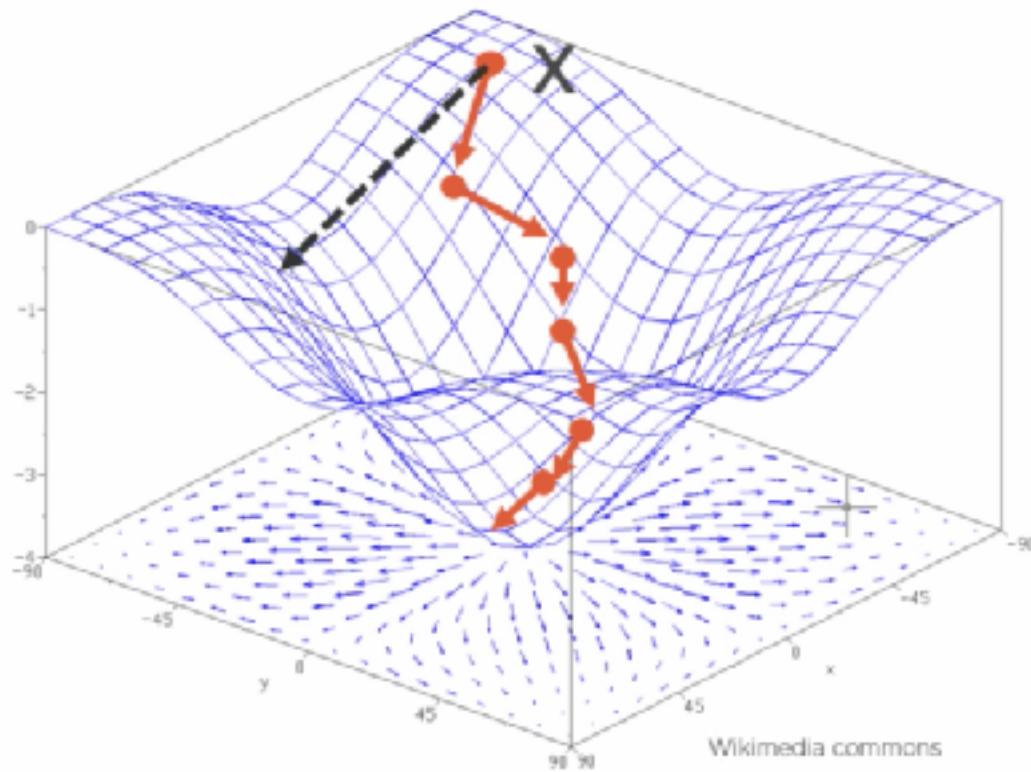


Wikimedia commons

Gradient Descent

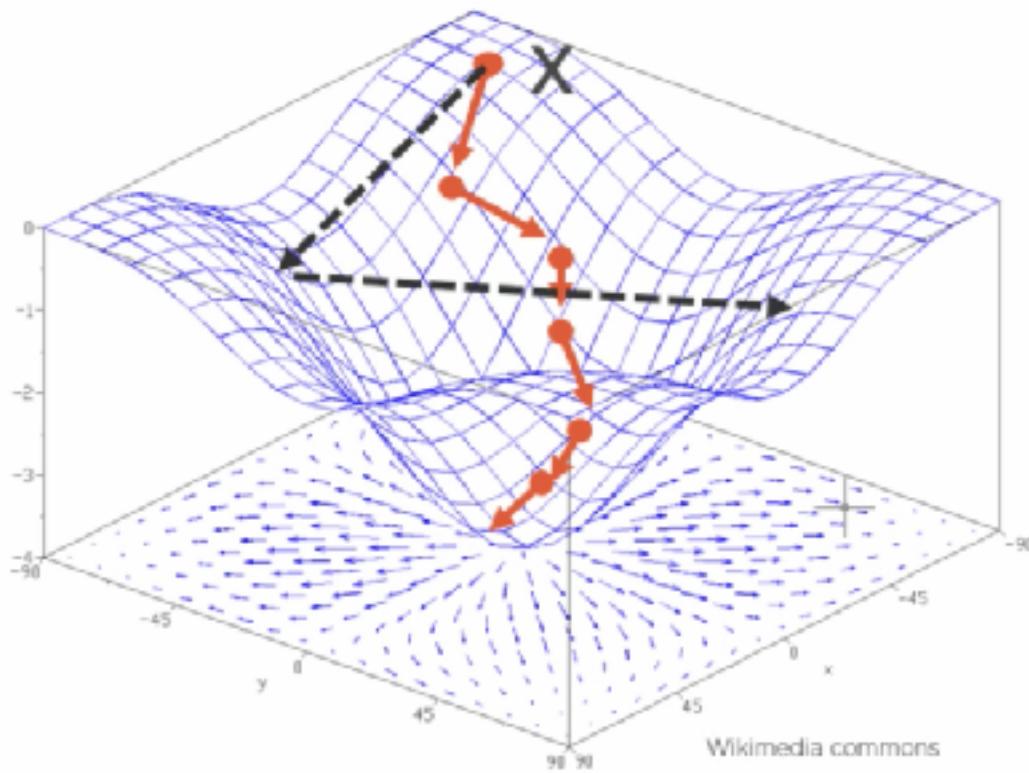


Gradient Descent

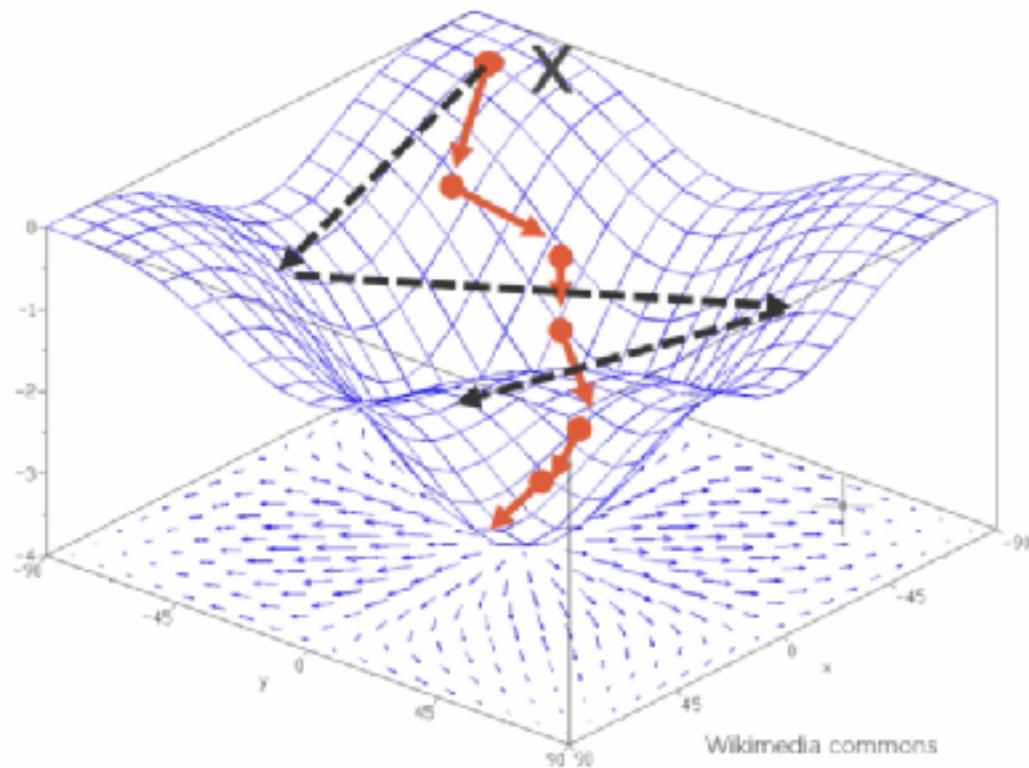


Wikimedia commons

Gradient Descent



Gradient Descent



Iteration	Error
1	9.556915033600001
2	9.514033718864932
3	9.471355093177891
4	9.42887819847207
5	9.302648387373978
10	9.302648387373978
20	9.260968926175824
30	8.775918820666949
40	8.392252947074406
50	8.02634104901006
60	7.677361561773854
100	6.160260505649477
200	4.018554474422596
300	2.685046327855845
400	1.854748522005687
800	0.6906129091698867
1000	0.5644839798882763
1600	0.4891352315933852

Step 1

import statement:

```
1 from sklearn import linear_model
```

Step 2

I have the height and weight data of some people. Let's use this data to do linear regression and try to predict the weight of other people.

```
1 height=[[4.0],[4.5],[5.0],[5.2],[5.4],[5.8],[6.1],[6.2],[6.4],[6.8]]
2 weight=[ 42 , 44 , 49, 55 , 53 , 58 , 60 , 64 , 66 , 69]
3
4 print("height weight")
5 for row in zip(height, weight):
6     print(row[0][0],"->",row[1])
```

Step 3

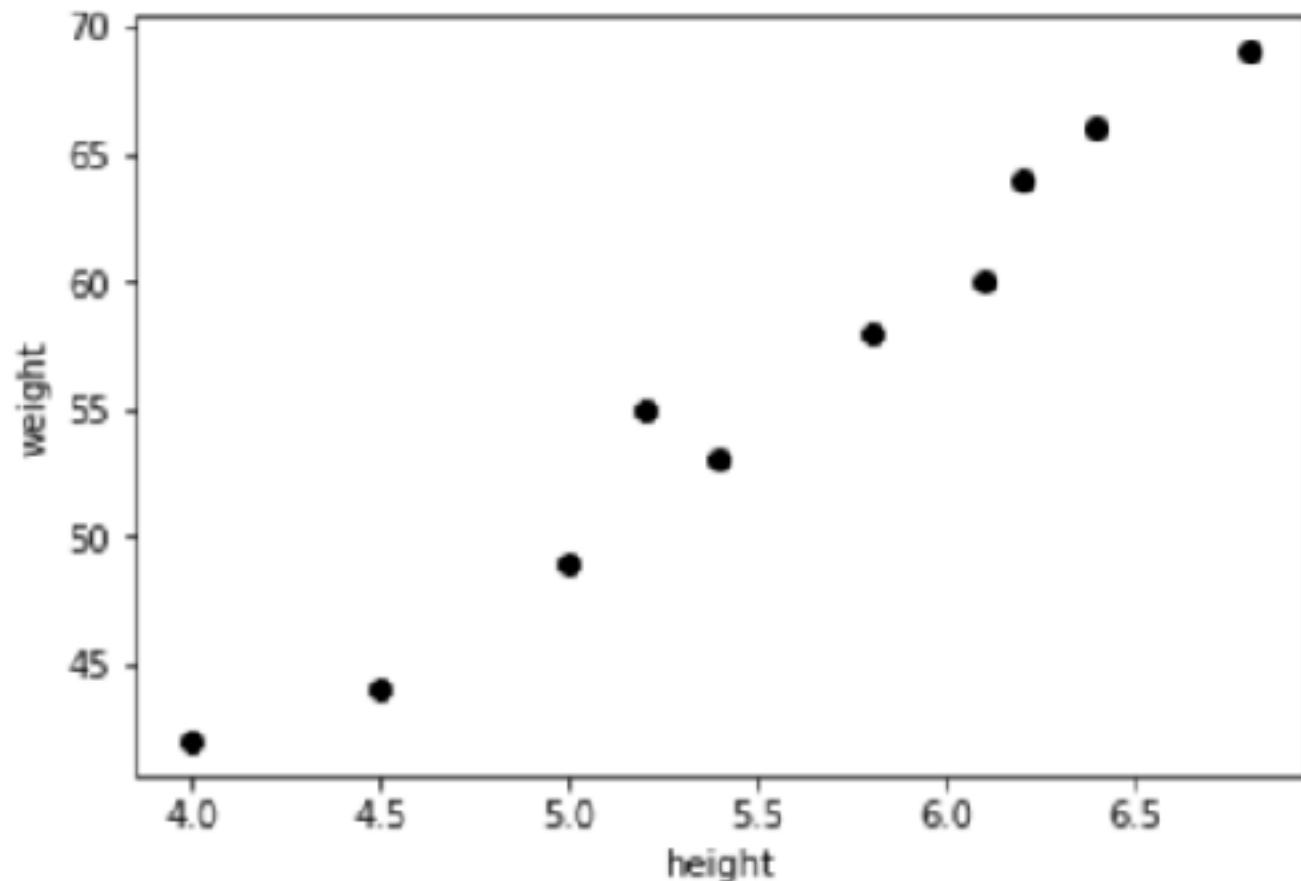
import statement to plot graph using matplotlib:

```
1 import matplotlib.pyplot as plt
```

Plotting the height and weight data:

```
1 plt.scatter(height,weight,color='black')
2 plt.xlabel("height")
3 plt.ylabel("weight")
```

Output:



Step 4

Declaring the linear regression function and call the `fit` method to learn from data:

```
1 reg=linear_model.LinearRegression()  
2 reg.fit(height,weight)
```

Slope and intercept:

```
1 m=reg.coef_[0]  
2 b=reg.intercept_  
3 print("slope=",m, "intercept=",b)
```

Output:

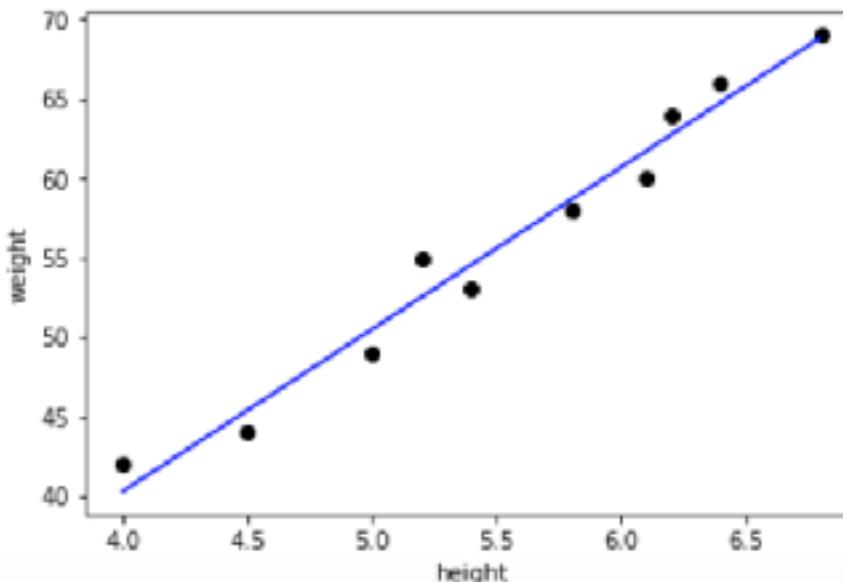
```
1 slope= 10.1936218679 intercept= -0.4726651480
```

Step 5

Using the values of slope and intercept to construct the line to fit our data points:

```
1 plt.scatter(height,weight,color='black')
2 predicted_values = [reg.coef_ * i + reg.intercept_ for i in height]
3 plt.plot(height, predicted_values, 'b')
4 plt.xlabel("height")
5 plt.ylabel("weight")
```

Output:



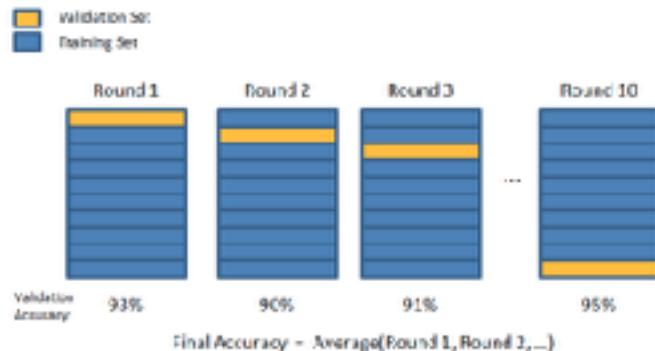
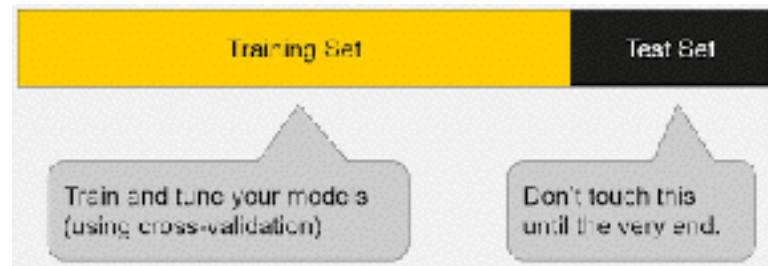
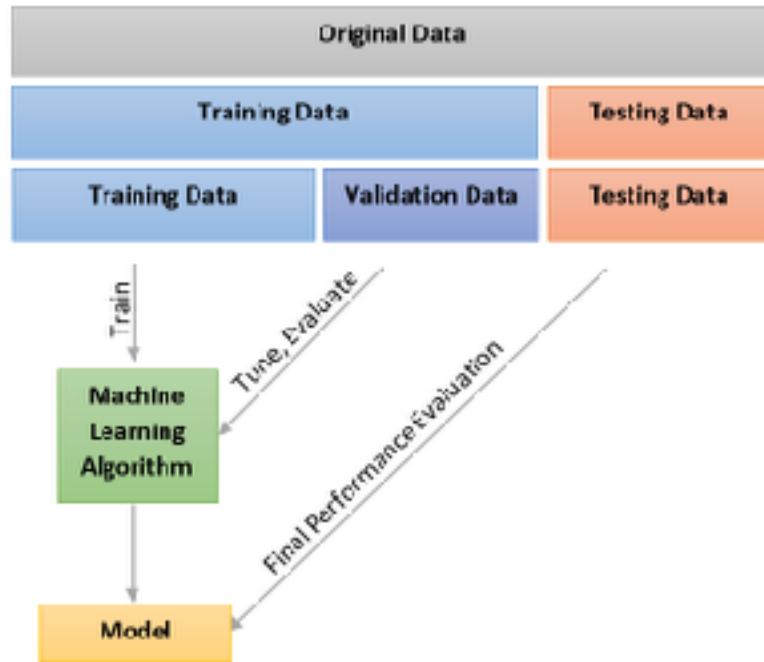
Advantage of Linear Regression

- Linear regression implements a statistical model that, when relationships between the independent variables and the dependent variable are almost linear, shows optimal results.
- Best place to understand the data analysis
- Easily Explicable

Disadvantages

- Linear regression is often inappropriately used to model non-linear relationships.
- Linear regression is limited to predicting numeric output.
- A lack of explanation about what has been learned can be a problem.
- Prone to bias variance problem

How to evaluate our model?



Overfitting vs Underfitting



Training Data(Less Error)



Testing Data (More Error)

Overfitting vs Underfitting

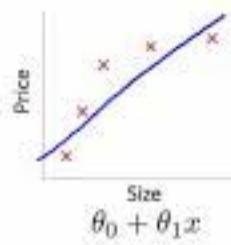


Training Data (More Error)

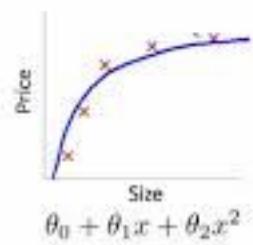


Testing (Still More Error)

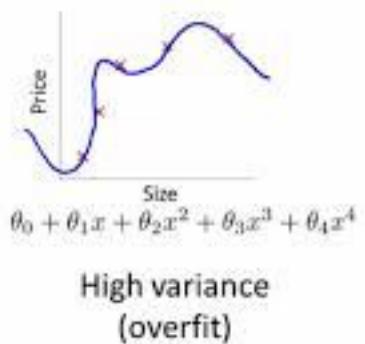
Variance and Bias Trade off



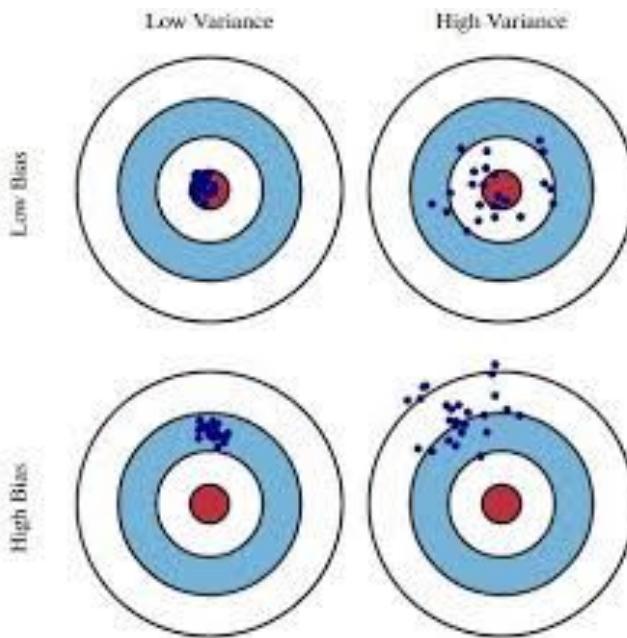
High bias
(underfit)



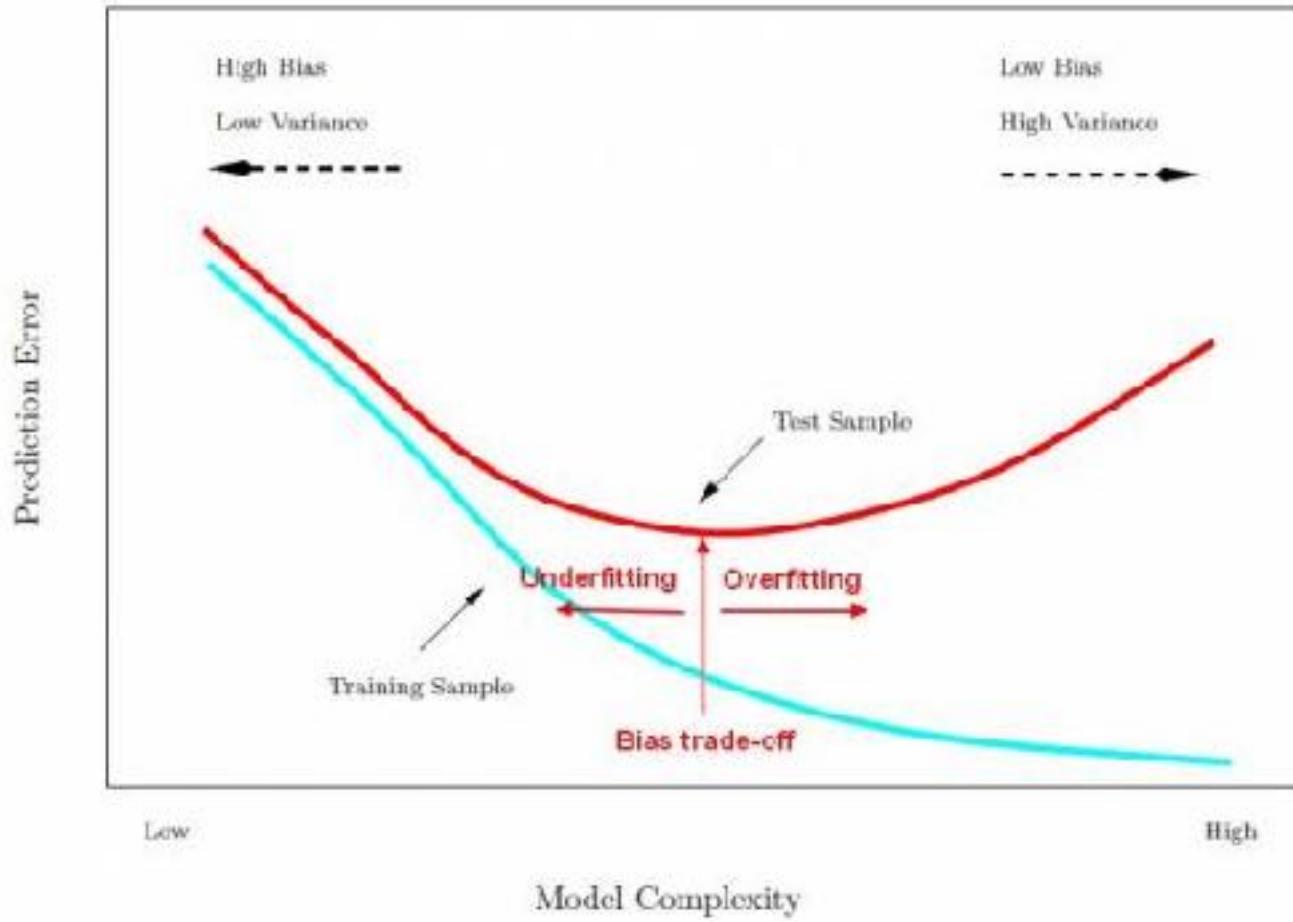
"Just right"



High variance
(overfit)



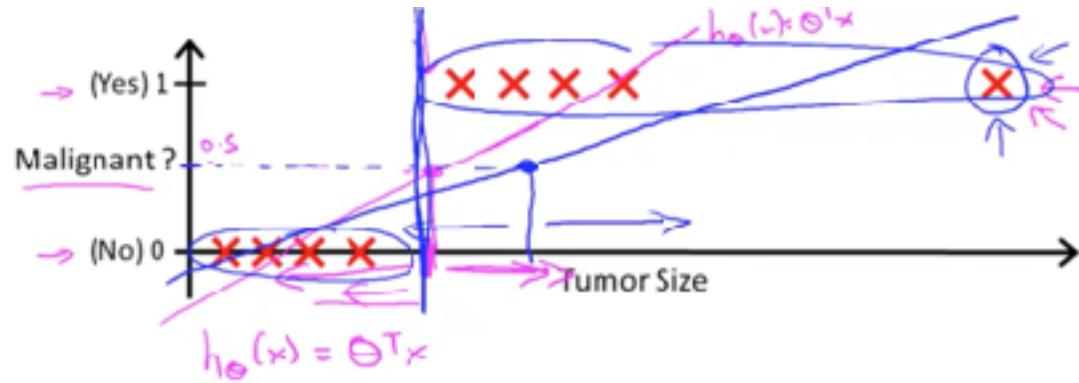
Ideal Model should have Low variance and Low Bias



Logistic Regression

Find who will likely to get Cardiovascular Disease

Cigarettes/Day	BMI	CVD
4	27	yes
6	25	yes
5	15	No
2	21	No
1	30	Yes
0	28	No
10	25	Yes



→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

$$g(y) = \beta_0 + \beta(\text{Age})$$

Since probability must always be positive, we'll put the linear equation in exponential form

To make the probability less than 1, we must divide p by a number greater than p. This can simply be done by:

$$p = \exp(\beta_0 + \beta(\text{Age})) / \exp(\beta_0 + \beta(\text{Age})) + 1 = e^{(\beta_0 + \beta(\text{Age}))} / e^{(\beta_0 + \beta(\text{Age}))} + 1 \quad \text{--- (c)}$$

$$p = e^y / 1 + e^y \quad \text{--- (d)}$$

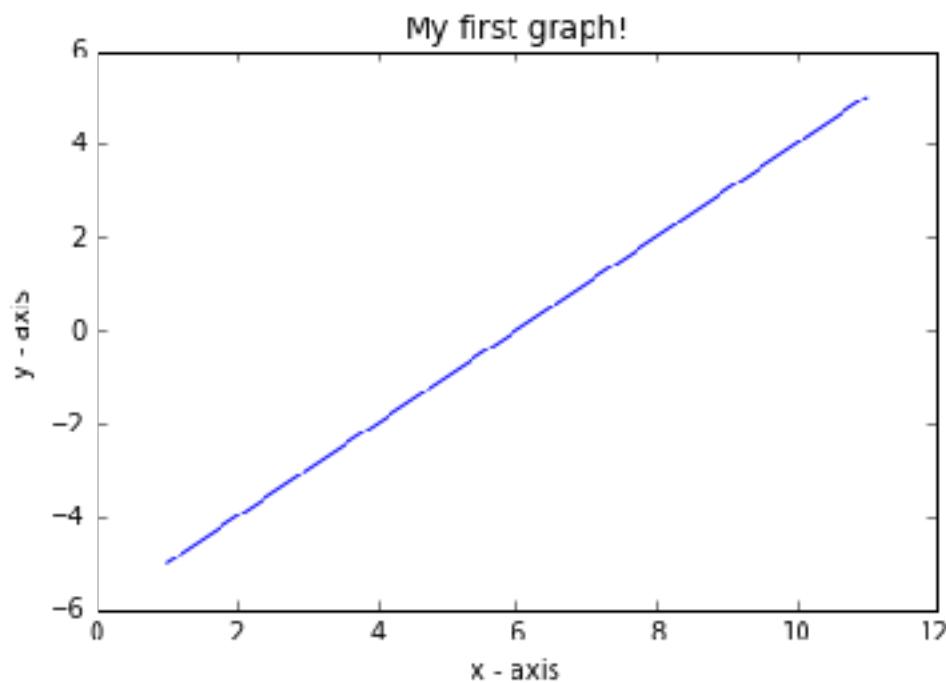
$$q = 1 - p = 1 - (e^y / 1 + e^y) \quad \text{--- (e)}$$

$$\frac{p}{1-p} - e^y$$

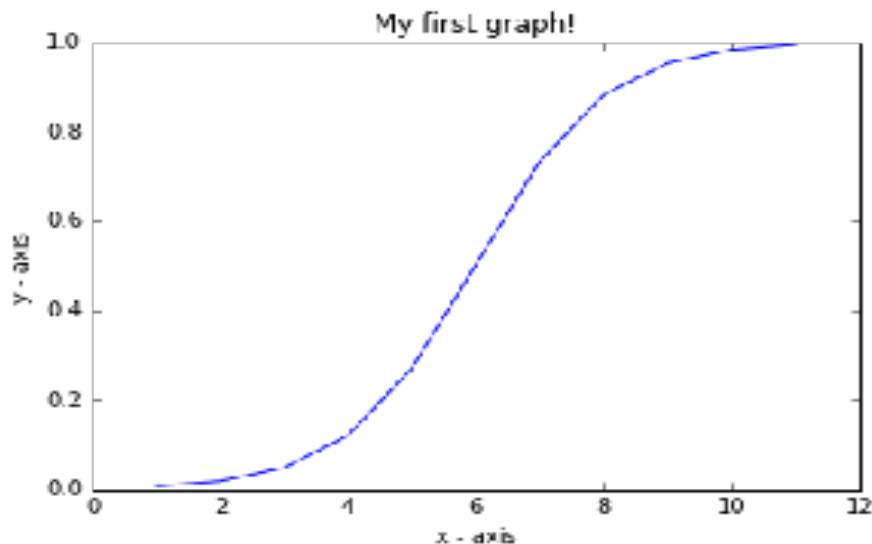
$$\log \left(\frac{p}{1-p} \right) = y$$

$$\log \left(\frac{p}{1-p} \right) = \beta_0 + \beta(\text{Age})$$

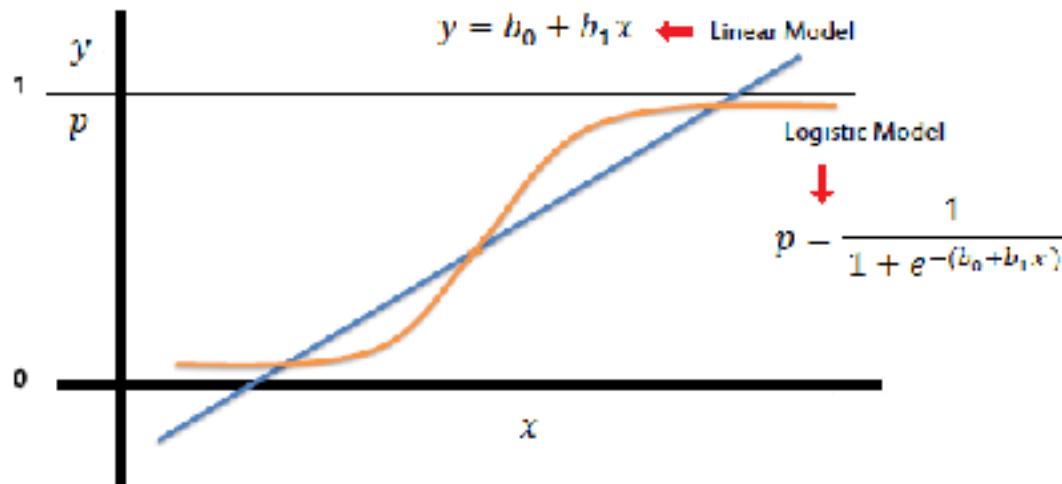
X	Y
1	-5
2	-4
3	-3
4	-2
5	-1
6	0
7	1
8	2
9	3
10	4
11	5



X	Sigmoid(Y)
1	0.00669285092
2	0.01798620996
3	0.04742587318
4	0.119202922
5	0.2689414214
6	0.5
7	0.7310585786
8	0.880797078
9	0.9525741268
10	0.98201379
11	0.9933071491



Logistic Regression Equation



Step 1

Import modules

```
In [1]: import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
```

Step 2

Prepare Data for Logistic Regression

To prepare the data, I want to add an intercept column as well as dummy variables for `occupation` and `occupation_husb`, since I'm treating them as categorical variables. The `dmatrices` function from the `patsy module` can do that using formula language.

```
In [10]: # create dataframes with an intercept column and dummy variables for
# occupation and occupation_husb
y, X = dmatrices('affair ~ rate_marriage + age + yrs_married + children + \
                  religious + educ + C(occupation) + C(occupation_husb)',
                  dta, return_type="dataframe")
print X.columns
```

Step 3

Model Evaluation Using a Validation Set

So far, we have trained and tested on the same set. Let's instead split the data into a training set and a testing set.

```
In [16]: # evaluate the model by splitting into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

Step 4

We now need to predict class labels for the test set. We will also generate the class probabilities, just to take a look.

```
In [17]: # predict class labels for the test set  
predicted = model2.predict(X_test)  
print predicted
```

Error Metrics

Classification Metrics -> Confusion Matrix (1)

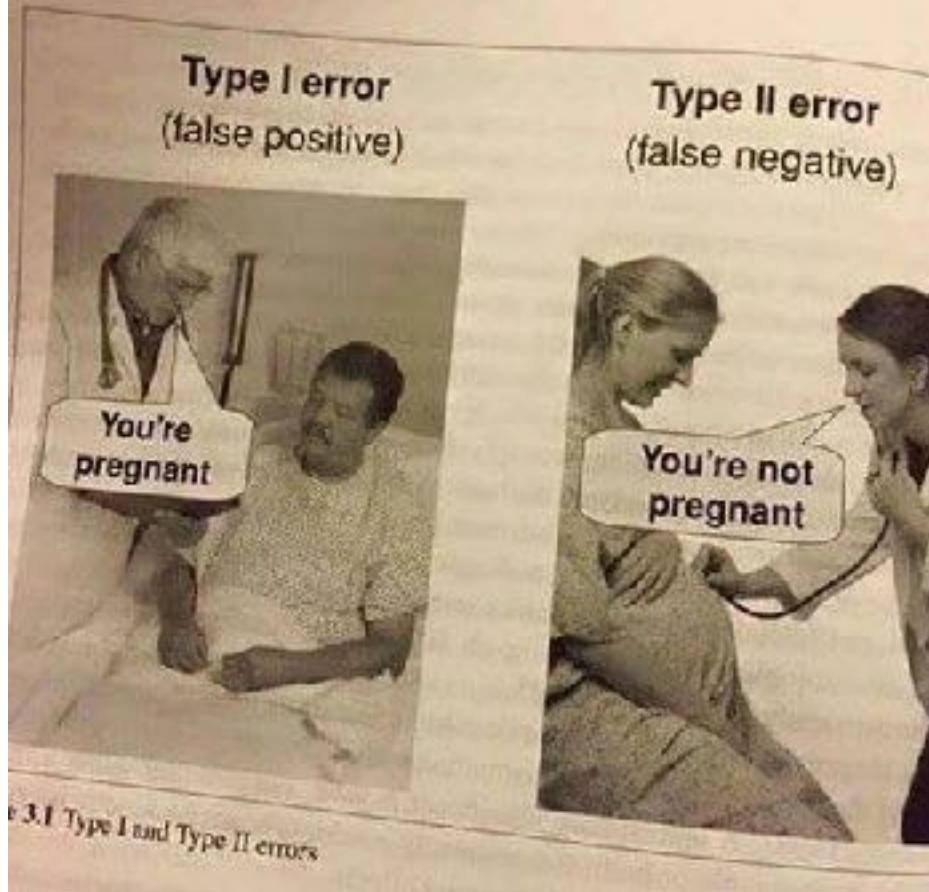
		"Golden Standard" (Real Truth Values)		
		Positive	Negative	
Observed	Predicted positive	True Positive	False Positive (Type 1 error)	Precision
	Predicted Negative	False Negative (Type 2 error)	True Negative	
		Recall/ Sensitivity	(Specificity)	

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}}$$

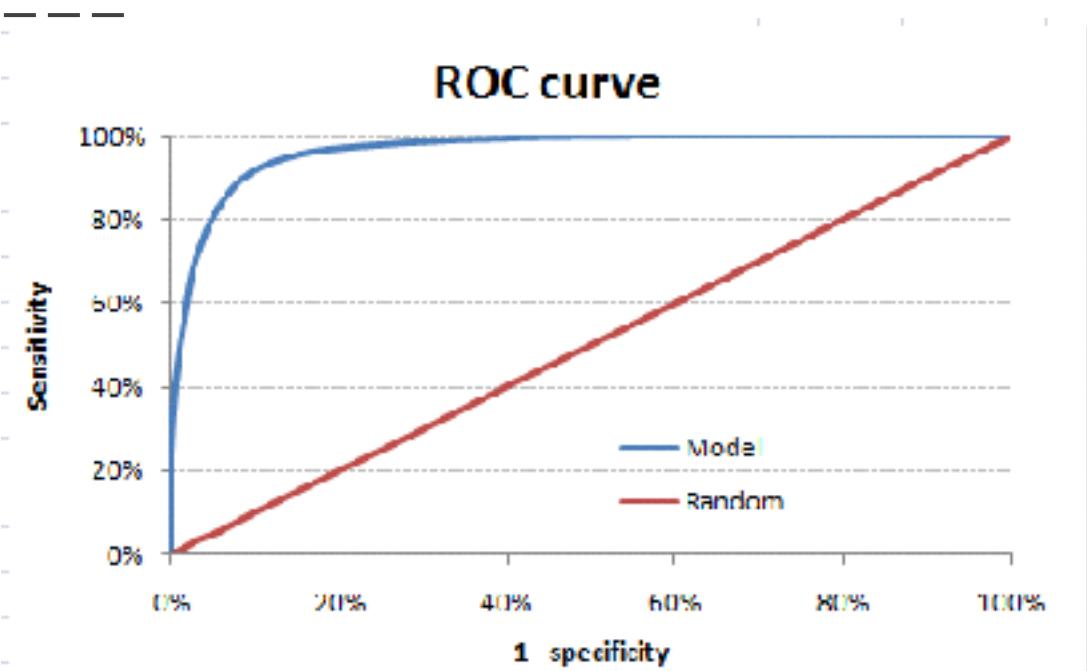
$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$



Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	Positive Predictive Value	$a/(a+b)$
	Negative	c	d	Negative Predictive Value	$d/(c+d)$
		Sensitivity	Specificity	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

Count of ID	Target				
Model		1	0	Grand Total	
1		3,834	639	4,473	85.7%
0		16	951	967	1.7%
Grand Total		3,850	1,590	5,440	
		99.6%	40.19%		88.0%

ROC CURVE



- 90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)

Logistic Regression Advantages

- Convenient probability scores for observations
- Efficient implementations available across tools
- Multicollinearity is not really an issue and can be countered with L2 regularisation to an extent
- Wide spread industry comfort for logistic regression solutions [oh that's important too!]

Logistic Regression Disadvantages

- Doesn't perform well when feature space is too large
- Doesn't handle large number of categorical features/variables well
- Relies on transformations for non-linear features
- Relies on entire data [Not a very serious drawback I'd say]

K-Nearest Neighbors

KNN Algorithm

1. To classify document d into class c
2. Define k -neighborhood N as k nearest neighbors (according to a given distance or similarity measure) of d
3. Count number of documents k_c in N that belong to c
4. Estimate $P(c|d)$ as k_c/k
5. Choose as class $\operatorname{argmax}_c P(c|d)$ [= majority class]

Finding similar rows

Distance functions

Euclidean

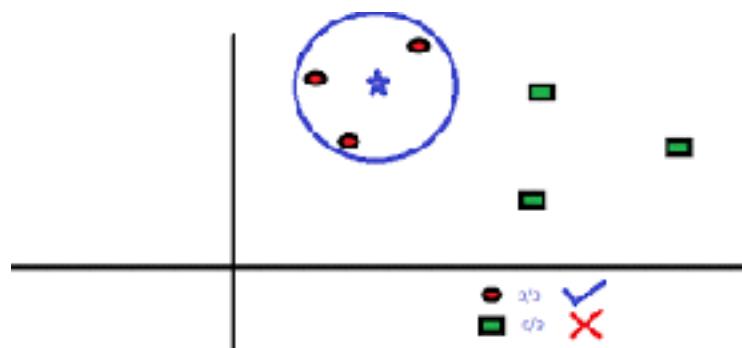
$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$



We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	X2(Strength) in kg/square meter	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	X2(Strength) in kg/square meter	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with $X1 = 3$ and $X2 = 7$.

Without another expensive survey, can we guess what the classification of this new tissue is?

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)
7	7	$(7-3)^2 + (7-7)^2 = 16$
7	4	$(7-3)^2 + (4-7)^2 = 25$
3	4	$(3-3)^2 + (4-7)^2 = 9$
1	4	$(1-3)^2 + (4-7)^2 = 13$

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3-Nearest Neighbors?
7	7	$(7-3)^2 + (7-7)^2 = 16$	3	Yes
7	4	$(7-3)^2 + (4-7)^2 = 25$	4	No
3	4	$(3-3)^2 + (4-7)^2 = 9$	1	Yes
1	4	$(1-3)^2 + (4-7)^2 = 13$	2	Yes

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more than 3

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more than 3

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	$(7-3)^2 + (7-7)^2 = 16$	3	Yes	Bad
7	4	$(7-3)^2 + (4-7)^2 = 25$	4	No	-
3	4	$(3-3)^2 + (4-7)^2 = 9$	1	Yes	Good
1	4	$(1-3)^2 + (4-7)^2 = 13$	2	Yes	Good

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more than 3

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	$(7-3)^2 + (7-7)^2 = 16$	3	Yes	Bad
7	4	$(7-3)^2 + (4-7)^2 = 25$	4	No	-
3	4	$(3-3)^2 + (4-7)^2 = 9$	1	Yes	Good
1	4	$(1-3)^2 + (4-7)^2 = 13$	2	Yes	Good

We have 2 good and 1 bad, since $2 > 1$ then we conclude that a new paper tissue that pass laboratory test with $X1 = 3$ and $X2 = 7$ is included in **Good category**

Step 1

```
1 import numpy as np
2 from sklearn.preprocessing import Imputer
3 from sklearn.cross_validation import train_test_split
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score
```

Step 2 - Import Data

Step 3

```
1 X_train, X_test, y_train, y_test = train_test_split(
2     X, Y, test_size = 0.3, random_state = 100)
3 y_train = y_train.ravel()
4 y_test = y_test.ravel()
```

Step 4

```
1 for K in range(25):
2     K_value = K+1
3     neigh = KNeighborsClassifier(n_neighbors = K_value, weights='uniform', algorithm='auto')
4     neigh.fit(X_train, y_train)
5     y_pred = neigh.predict(X_test)
6     print "Accuracy is ", accuracy_score(y_test,y_pred)*100,"% for K-Value:",K_value
```

KNN Advantage

- Ease to interpret output
- Calculation time
- Predictive Power

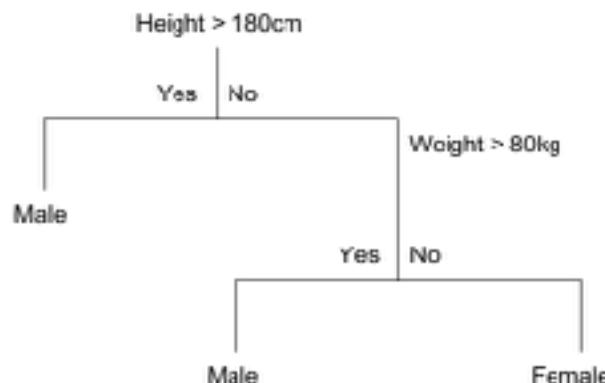
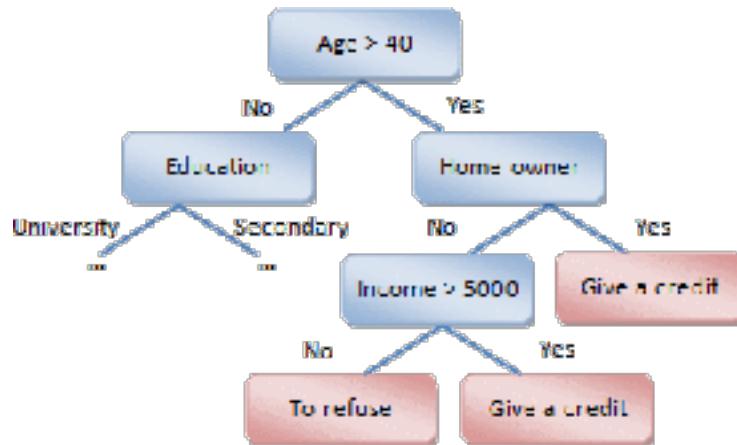
KNN Disadvantage

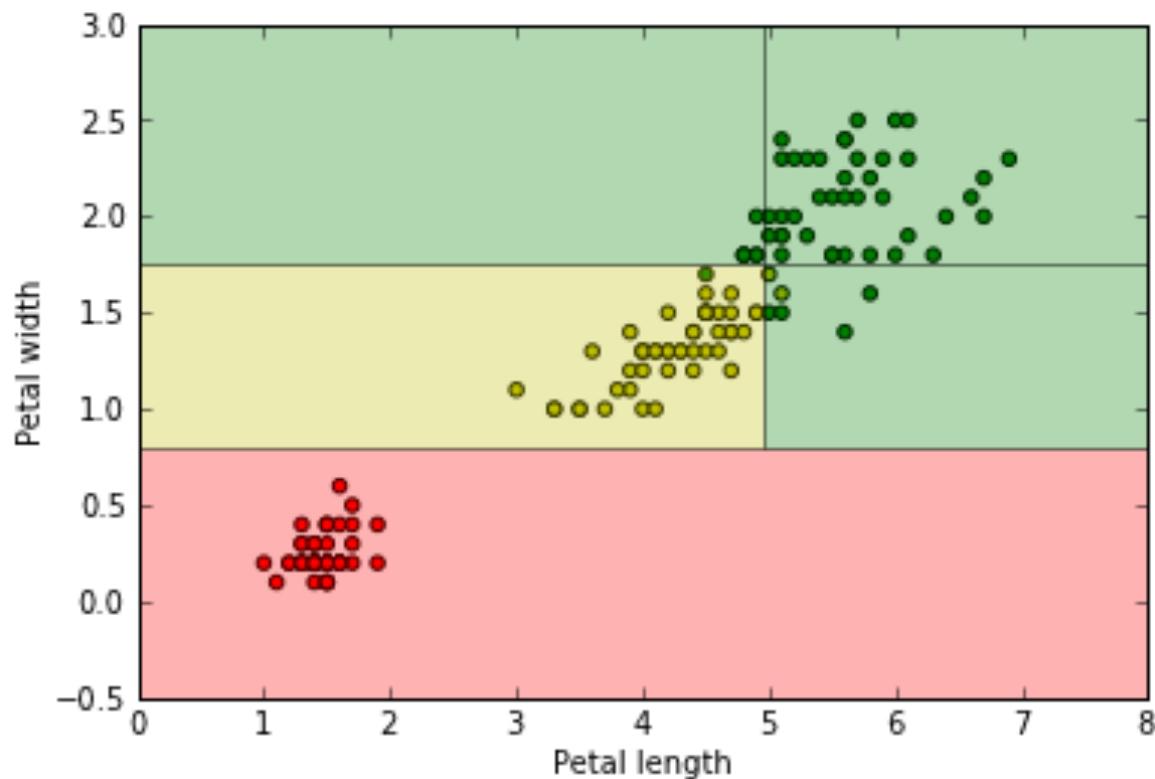
- Large search problem to find nearest neighbours
- Storage of data
- Must know we have a meaningful distance function

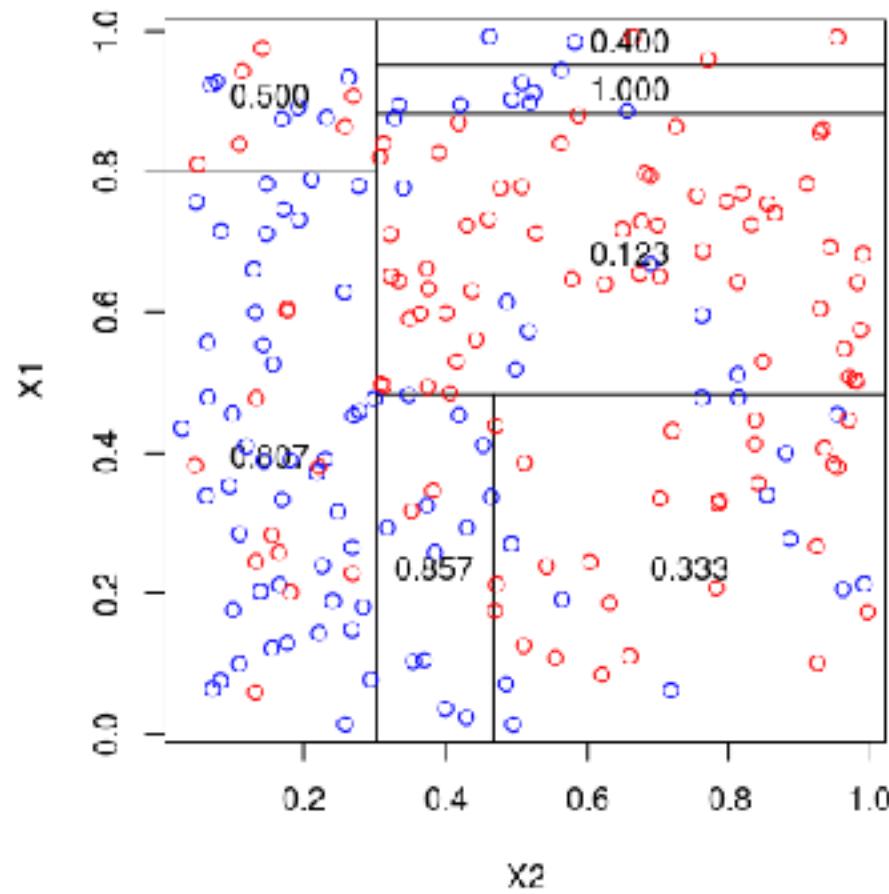
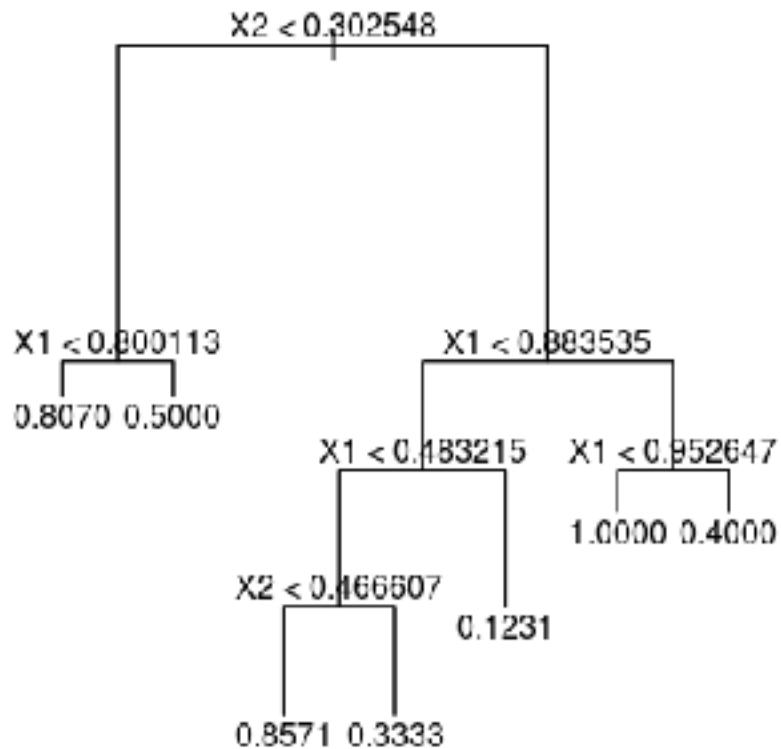
Decision Tree

How it works?

- Decision tree uses a tree structure to represent number of possible decision paths and an outcome for each path
- Decision Trees can be applied to both classification & regression problems
- Classification:
Entropy, Information Gain
- Regression:
Mean Square Error







— — —

[http://www.saedsayad.com/
decision_tree.htm](http://www.saedsayad.com/decision_tree.htm)

Recursive divide-and-conquer fashion

- First: select attribute for root node

Create branch for each possible attribute value

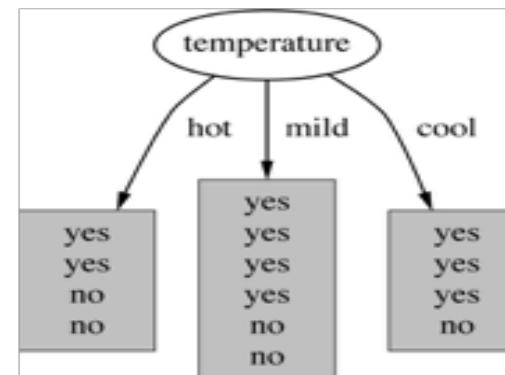
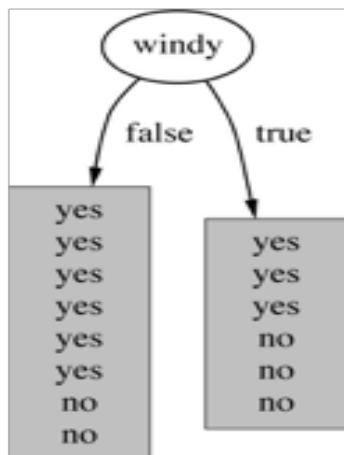
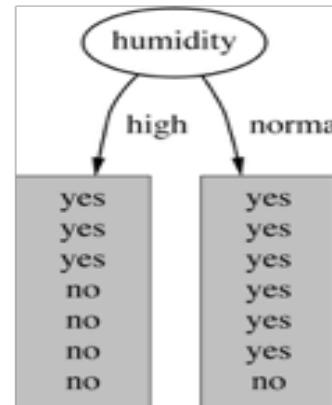
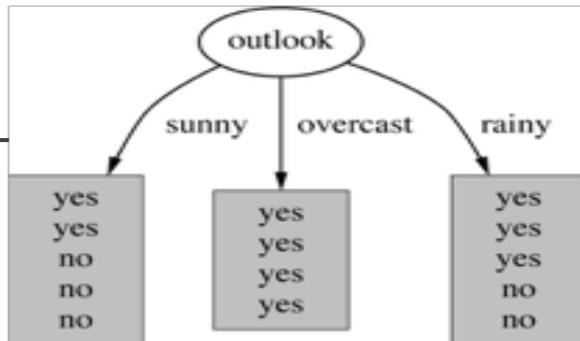
- Then: split instances into subsets

One for each branch extending from the node

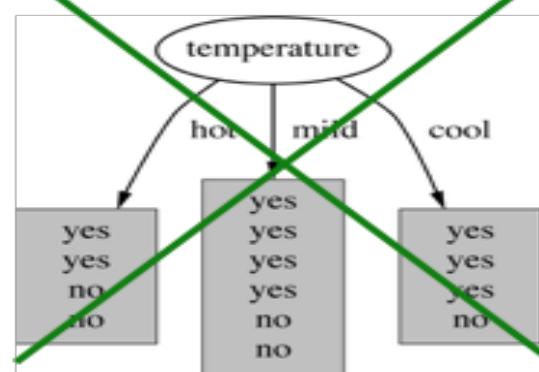
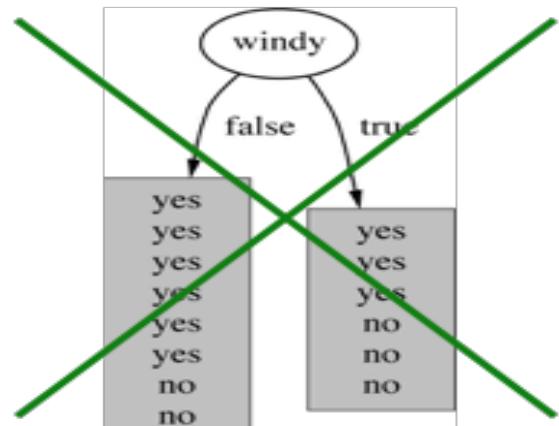
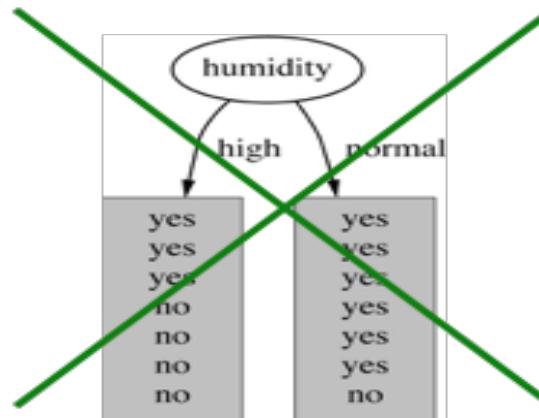
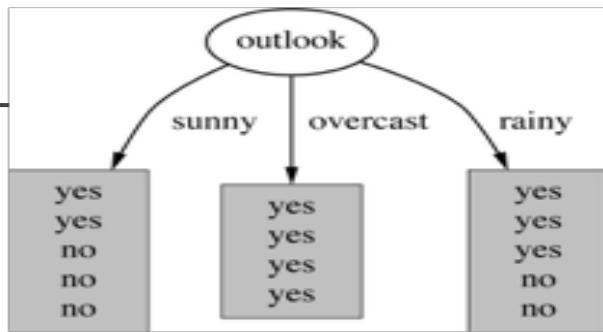
- Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

CUTLOOK	TEMP	HUMIDITY	WINDY	PLAY
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

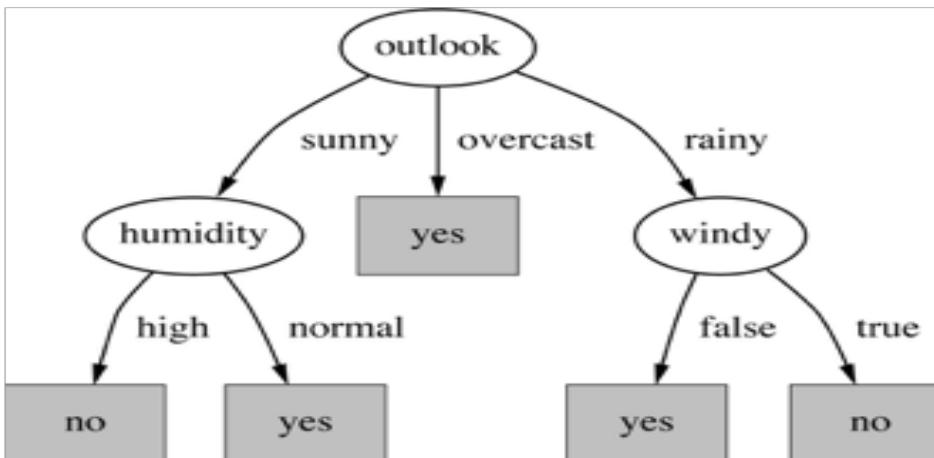
Which attribute to select?



Which attribute to select?



Final decision tree



⇒ **Splitting stops when data can't be split any further**



Computing Information Gain

- Information gain: information before splitting – information after splitting

$$\begin{aligned}\text{gain}(\text{Outlook}) &= \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \text{ bits}\end{aligned}$$

- Information gain for attributes from weather data:

$$\begin{array}{ll}\text{gain}(\text{Outlook}) & = 0.247 \text{ bits} \\ \text{gain}(\text{Temperature}) & = 0.029 \text{ bits} \\ \text{gain}(\text{Humidity}) & = 0.152 \text{ bits} \\ \text{gain}(\text{Windy}) & = 0.048 \text{ bits}\end{array}$$



Points to consider

- Information gain:
- Increases with the average purity of the subsets
- **Entropy:**
- to calculate the homogeneity of a sample.
- Pruning:
- Prevent overfitting to noise in the data

Decision Tree Pros

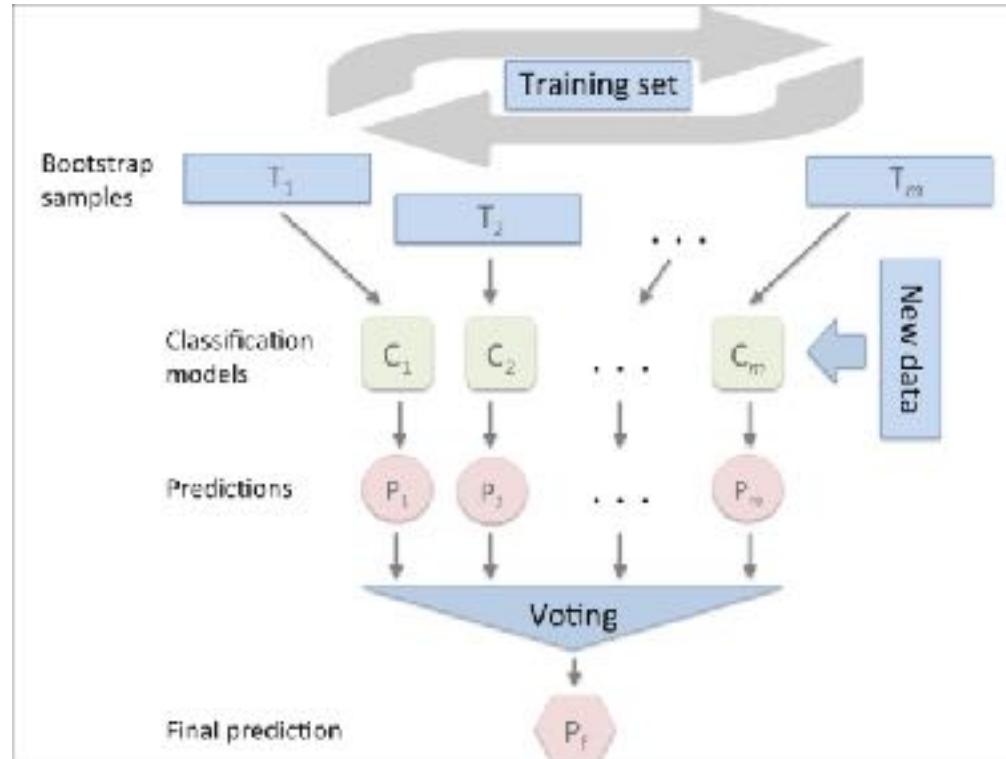
- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to **interpret** for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

Decision Tree Cons

- Inadequacy in applying regression and predicting continuous values
- Unsuitability for estimation of tasks to predict values of a continuous attribute
- Limited to one output per attribute, and inability to represent tests that refer to two or more different objects
- Possibility of overfitting
- High variance and bias problem to overcome we can go for bagging technique

Ensemble

Machine learning paradigm
which combine weak
learners to become a strong
learner



Random Forest

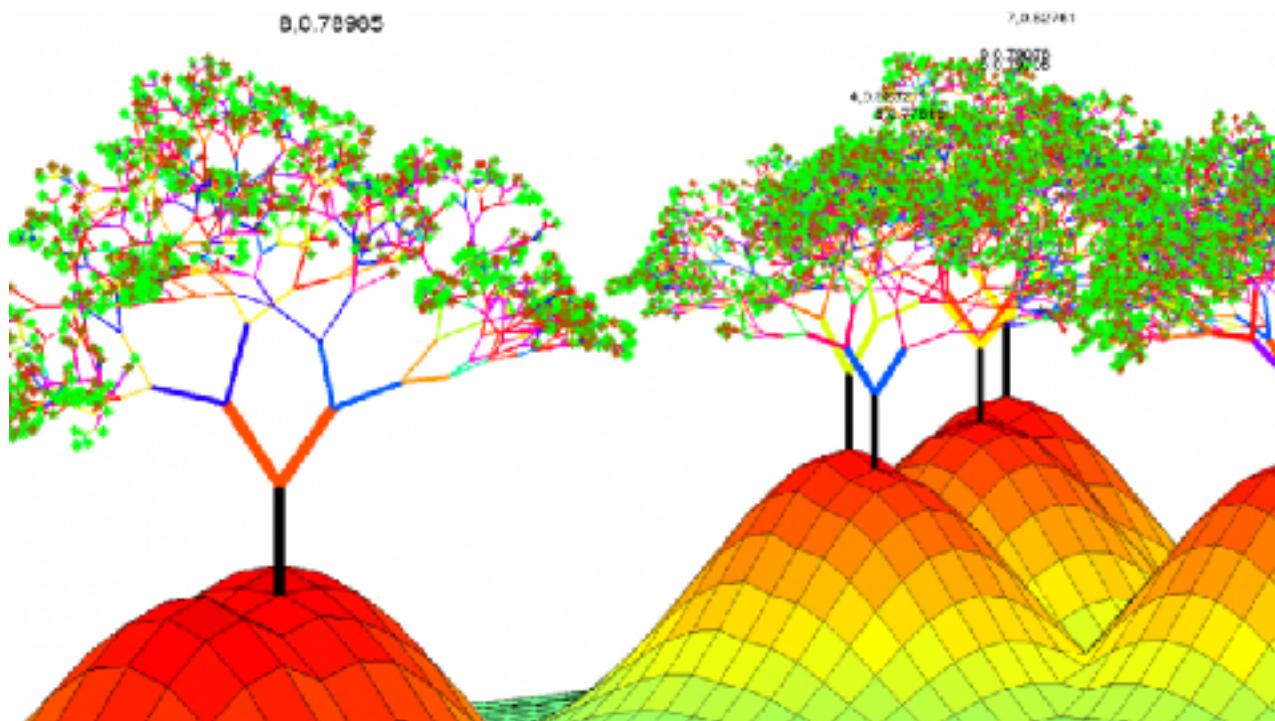
*Most used algorithm- Bagging
Technique*

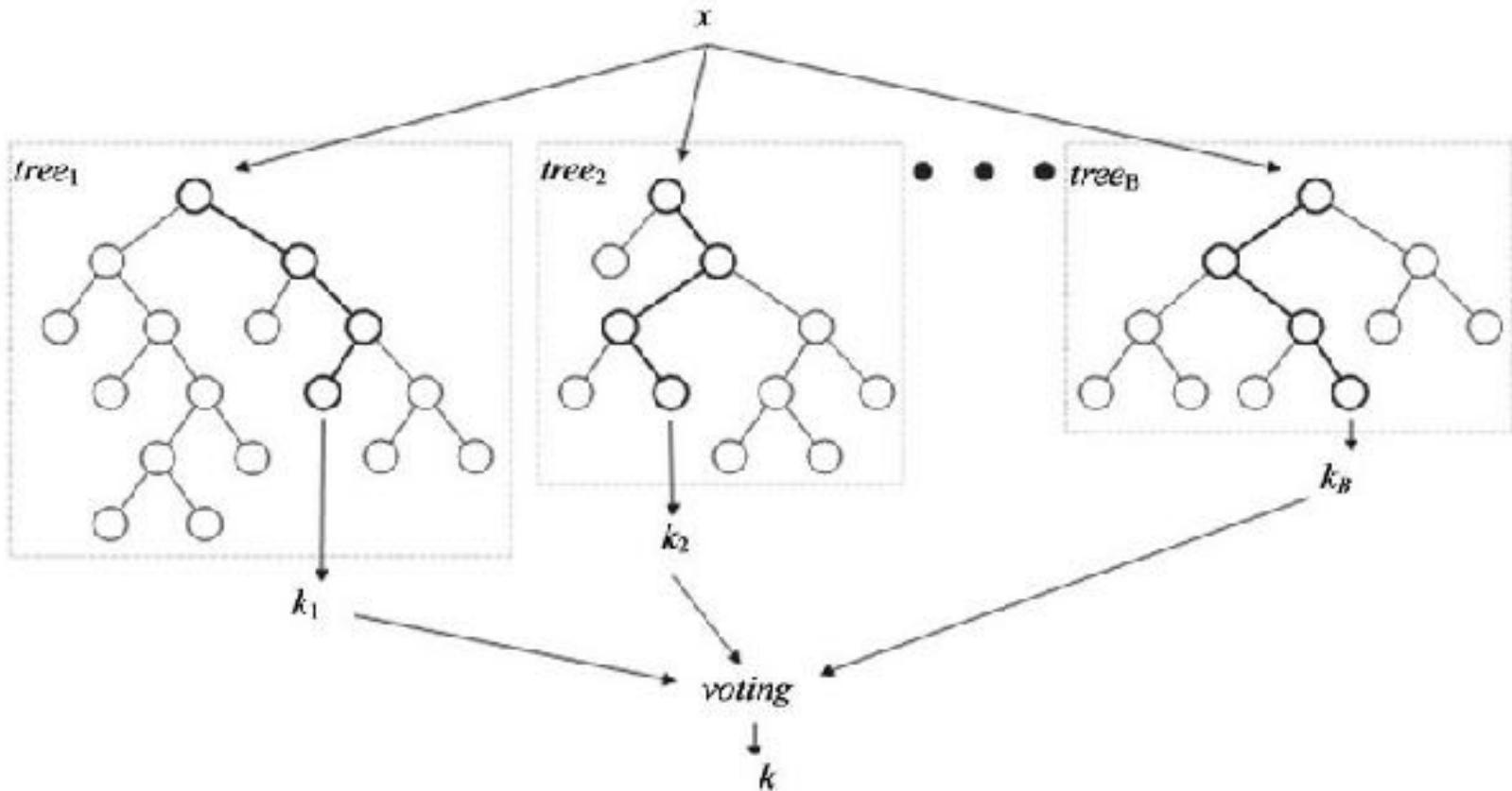
Bagging

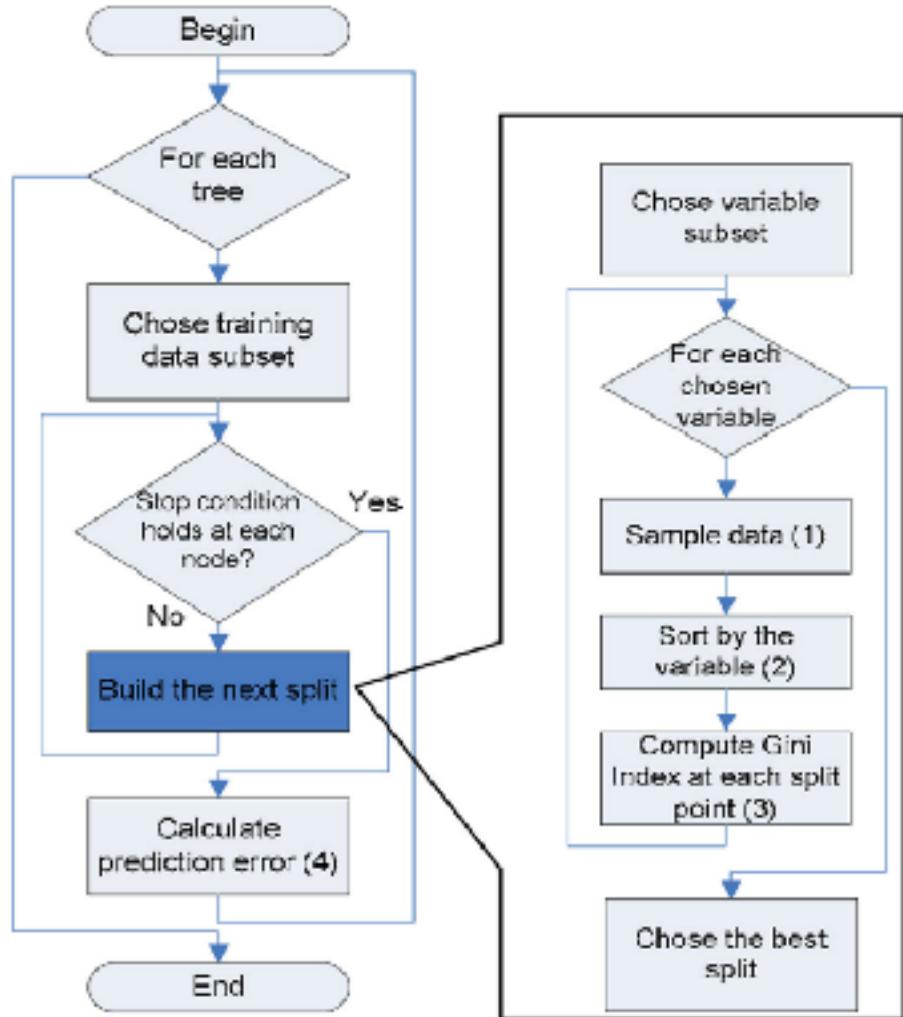
Bootstrap **aggregating**, also called **bagging**, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

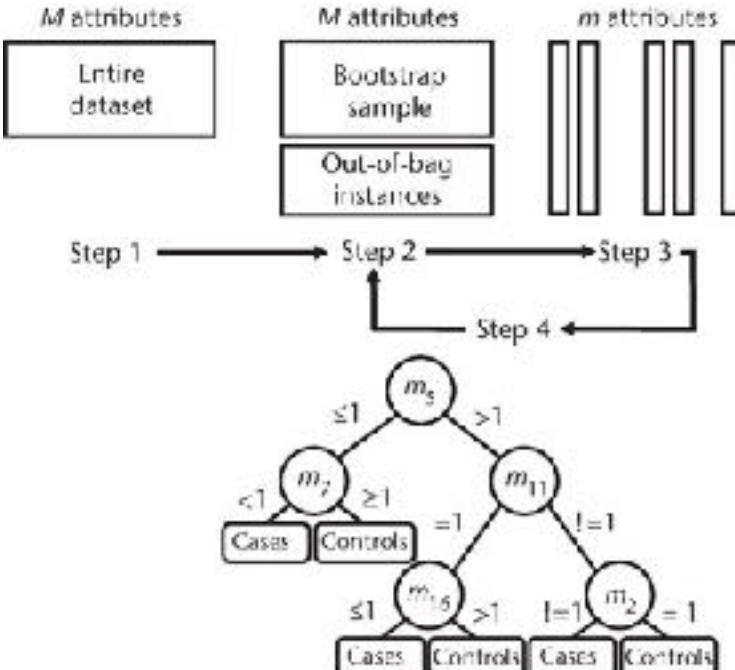
It also reduces variance and helps to avoid overfitting.

What insight can u get?









Points to remember

Feature selection

- For classification a good default is: $m = \sqrt{p}$
- For regression a good default is: $m = p/3$

Estimated Performance

- For each bootstrap sample taken from the training data, there will be samples left behind that were not included. These samples are called **Out-Of-Bag samples** or OOB.
- The performance of each model on its left out samples when averaged can provide an estimated accuracy of the bagged models. This estimated performance is often called the OOB estimate of performance.
- These performance measures are reliable test error estimate and correlate well with cross validation estimates.

Variable importance

- It will find most important variable for feature selection based on gini index

Random forest cons

It is one of the most accurate learning algorithms available.

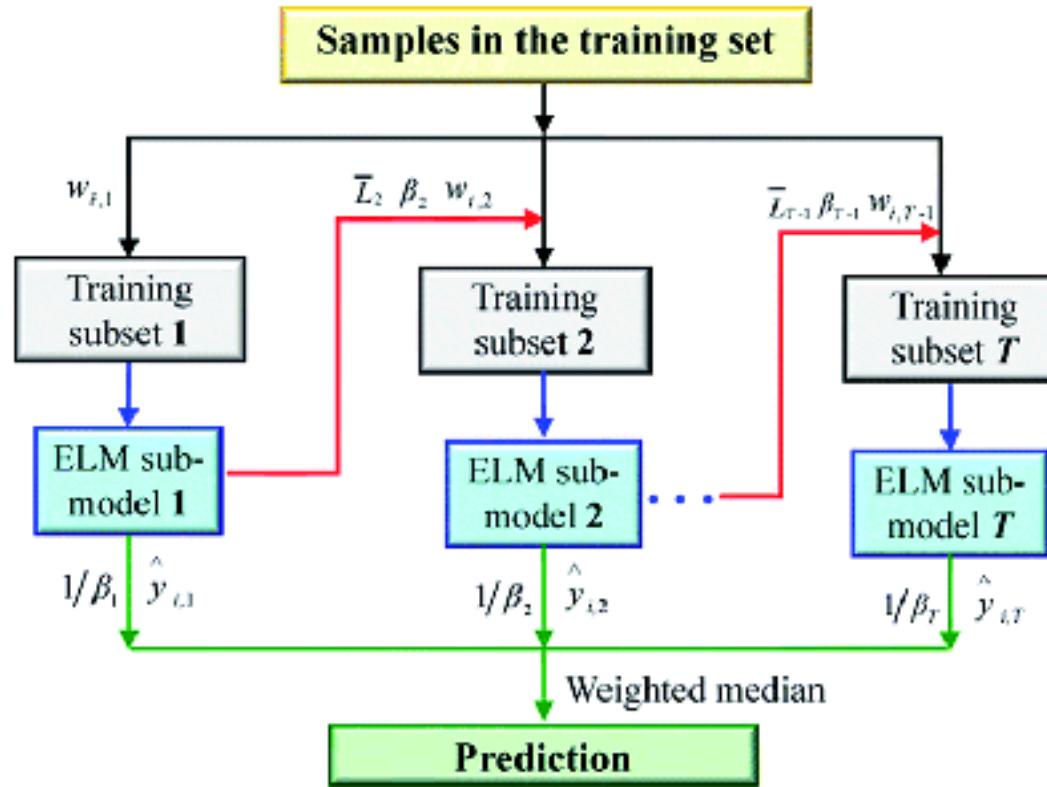
For many data sets, it produces a highly accurate classifier.

It runs efficiently on large databases.

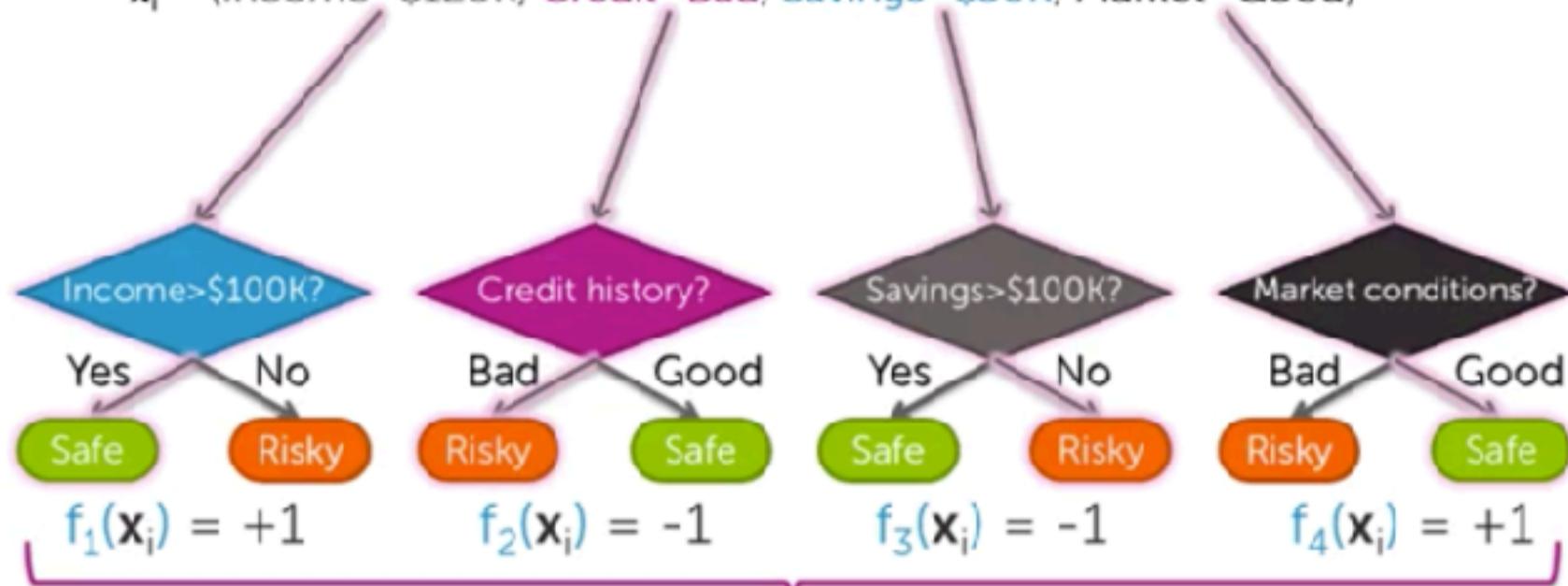
Boosting

Competition winning algorithm

Boosting



$\mathbf{x}_i = (\text{Income}=\$120K, \text{Credit}=\text{Bad}, \text{Savings}=\$50K, \text{Market}=\text{Good})$



Combine?

Ensemble
model

Learn coefficients

$$F(\mathbf{x}_i) = \text{sign}(w_1 f_1(\mathbf{x}_i) + w_2 f_2(\mathbf{x}_i) + w_3 f_3(\mathbf{x}_i) + w_4 f_4(\mathbf{x}_i))$$

Prediction with ensemble



Combine?

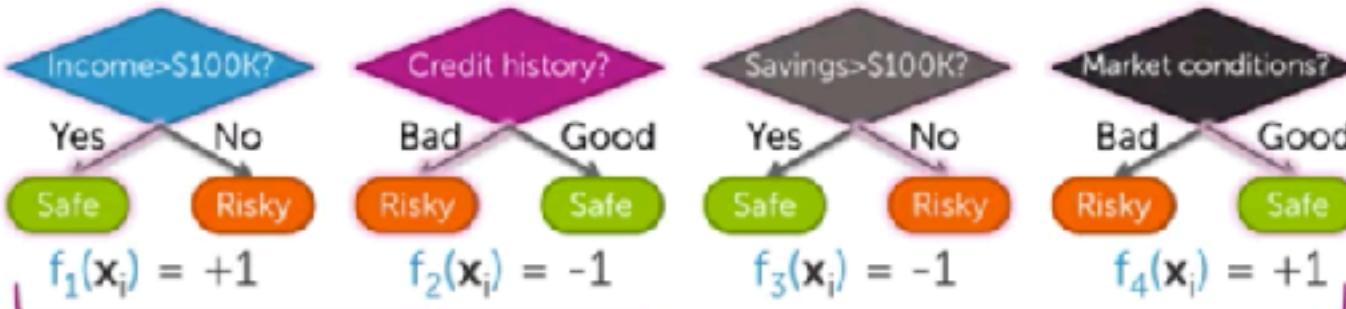
Ensemble model

Learn coefficients

$$F(x_i) = \text{sign}(w_1 f_1(x_i) + w_2 f_2(x_i) + w_3 f_3(x_i) + w_4 f_4(x_i))$$

w_1	2
w_2	1.5
w_3	1.5
w_4	0.5

Prediction with ensemble



Combine?

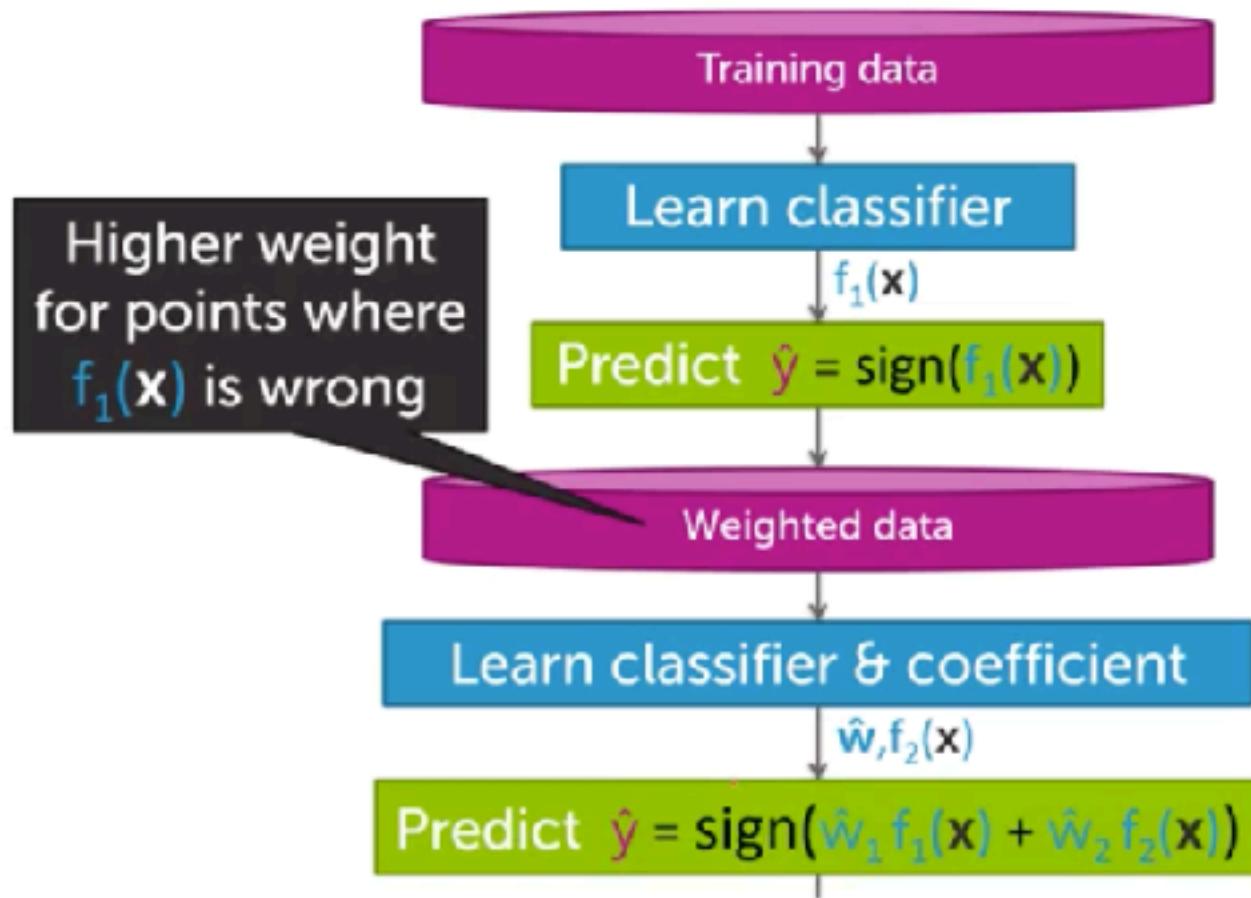
Ensemble model

Learn coefficients

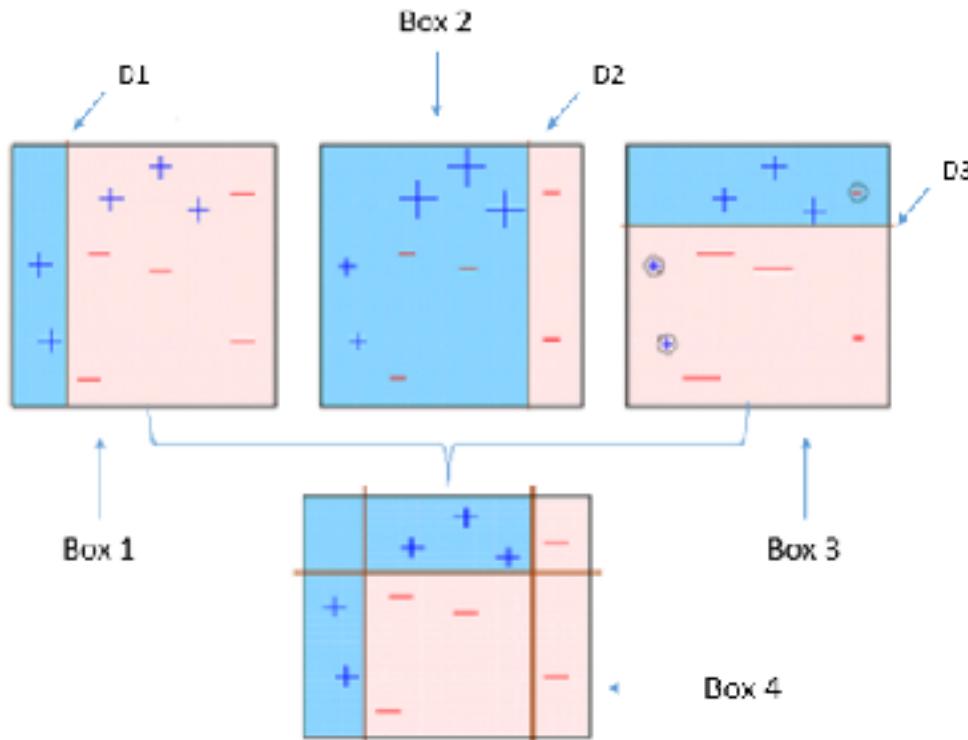
$$\begin{aligned}F(x_i) &= \text{sign}(w_1 f_1(x_i) + w_2 f_2(x_i) + w_3 f_3(x_i) + w_4 f_4(x_i)) \\&= \text{sign}(2 \cdot +1 + 1 \cdot -1 + 1 \cdot -1 + 0 \cdot +1) \\&= \text{sign}(-0.5) \Rightarrow g_i = -1\end{aligned}$$

w ₁	2
w ₂	1.5
w ₃	1.5
w ₄	0.5

Boosting = Greedy learning ensembles from data

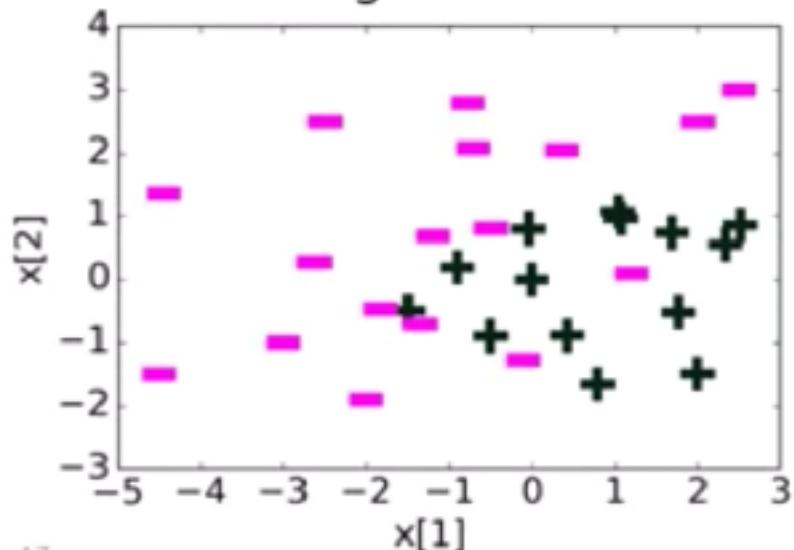


AdaBoost

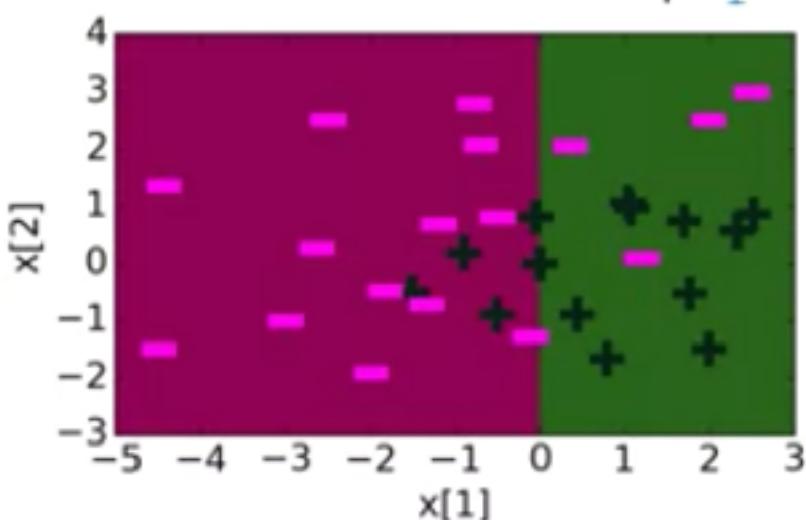


Iteration = 1

Original data

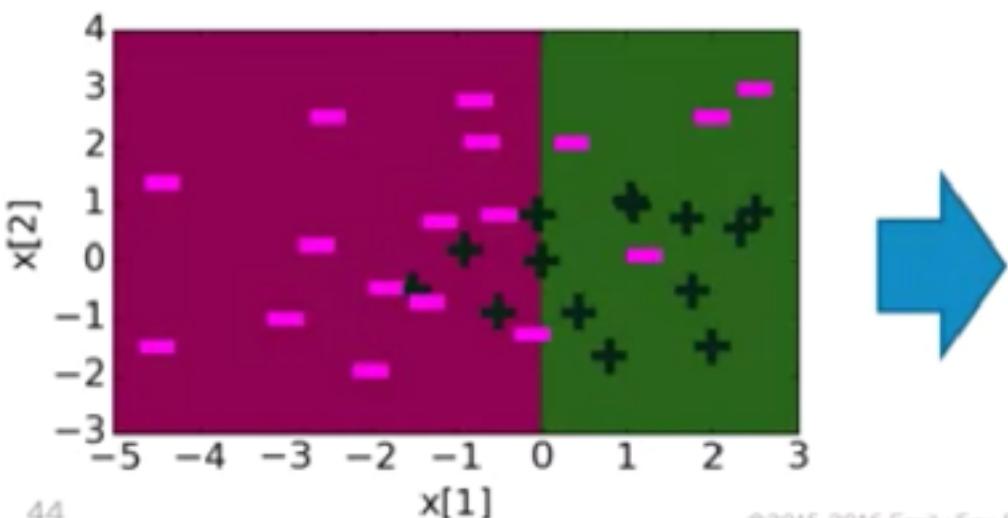


Learned decision stump $f_1(\mathbf{x})$

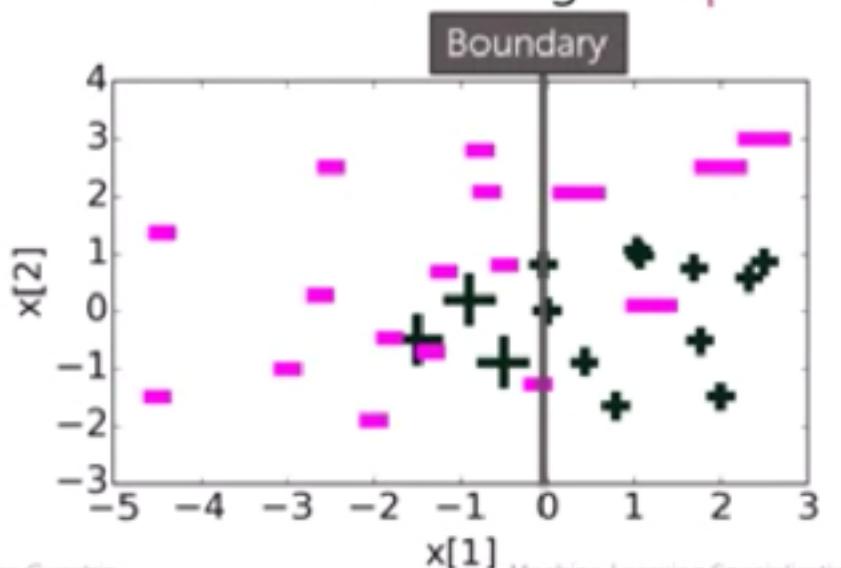


Updated Weights

Learned decision stump $f_1(\mathbf{x})$

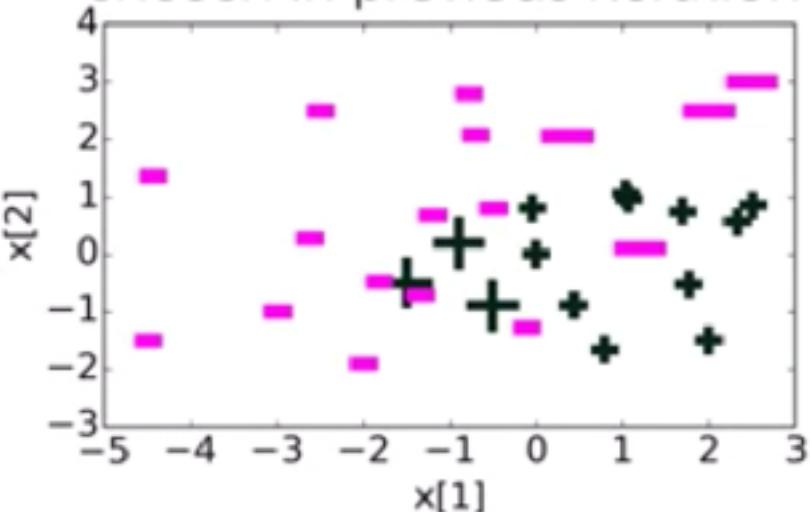


New data weights α_i

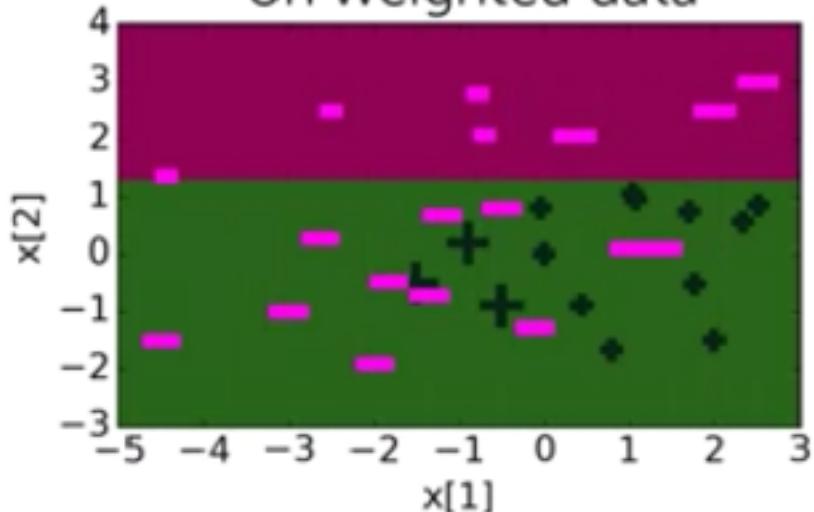


Iteration = 2

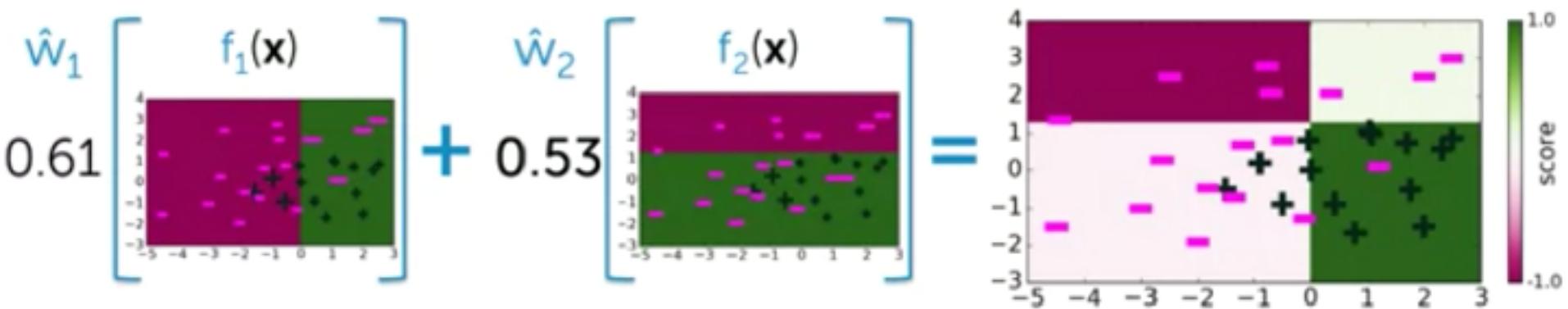
Weighted data: using α_i
chosen in previous iteration



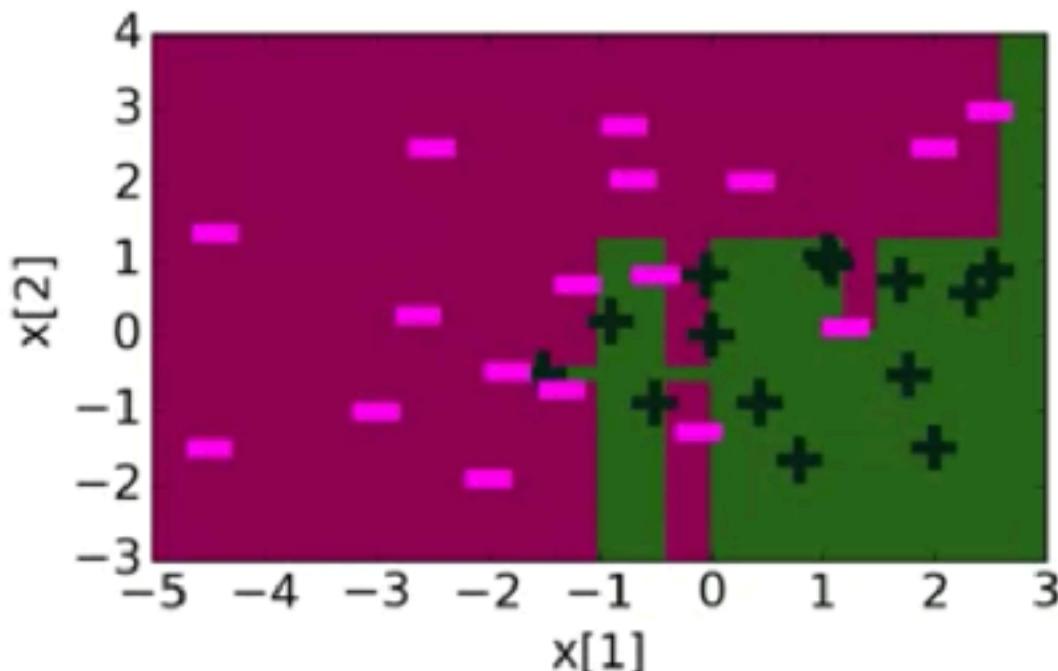
Learned decision stump $f_2(\mathbf{x})$
on weighted data



Ensemble becomes Weighted
sum of learned classifiers



Decision boundary of ensemble classifier after 30 iterations



training_error = 0

AdaBoost: learning ensemble

- Start same weight for all points: $\alpha_i = 1/N$

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

- For $t = 1, \dots, T$

- Learn $f_t(\mathbf{x})$ with data weights α_i

- Compute coefficient \hat{w}_t

- Recompute weights α_i

- Normalize weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^N \alpha_j}$$

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \dots, T$
 - Learn $f_t(x)$: pick decision stump with lowest weighted training error according to α_i
 - Compute coefficient \hat{w}_t
 - Recompute weights α_i
 - Normalize weights α_i
- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(x) \right)$$

Finding best next decision stump

Consider splitting on each feature:



Finding best next decision stump

Consider splitting on each feature:



Finding best next decision stump

Consider splitting on each feature:



weighted_error
= 0.2



weighted_error
= 0.35



Finding best next decision stump

Consider splitting on each feature:



weighted_error
= 0.2

weighted_error
= 0.35

weighted_error
= 0.3

Finding best next decision stump

Consider splitting on each feature:



$$\text{weighted_error} = 0.2$$

$$\text{weighted_error} = 0.35$$

$$\text{weighted_error} = 0.3$$

$$\text{weighted_error} = 0.4$$

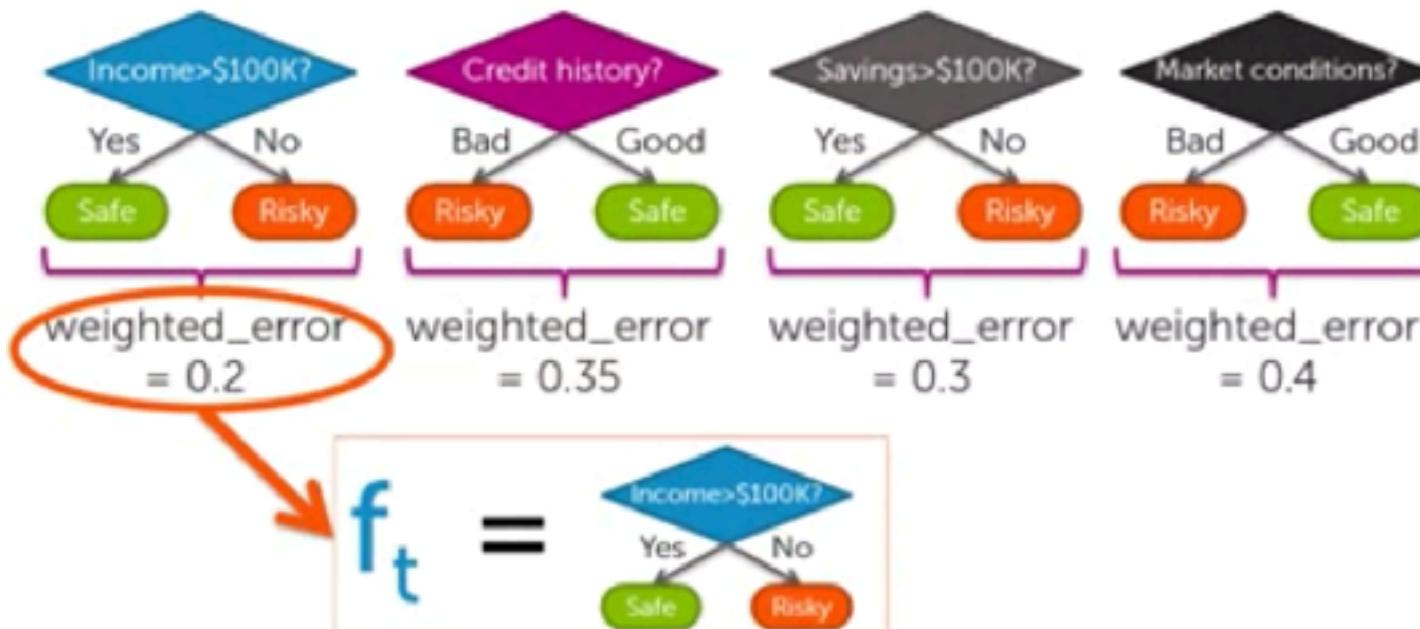
Finding best next decision stump

Consider splitting on each feature:



Finding best next decision stump

Consider splitting on each feature:



$$\hat{W}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right) = 0.69$$

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \dots, T$
 - Learn $f_t(\mathbf{x})$: pick decision stump with lowest weighted training error according to α_i
 - Compute coefficient \hat{w}_t
 - Recompute weights α_i
 - Normalize weights α_i
- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

Updating weights α_i



Credit	Income	y
A	\$130K	Safe
B	\$80K	Risky
C	\$110K	Risky
A	\$110K	Safe
A	\$90K	Safe
B	\$120K	Safe
C	\$30K	Risky
C	\$60K	Risky
B	\$95K	Safe
A	\$60K	Safe
A	\$98K	Safe

Updating weights α_i



Credit	Income	y	\hat{y}
A	\$130K	Safe	Safe
B	\$80K	Risky	Risky
C	\$110K	Risky	Safe
A	\$110K	Safe	Safe
A	\$90K	Safe	Risky
B	\$120K	Safe	Safe
C	\$30K	Risky	Risky
C	\$60K	Risky	Risky
B	\$95K	Safe	Risky
A	\$60K	Safe	Risky
A	\$98K	Safe	Risky

Updating weights α_i



$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(x_i) \neq y_i \end{cases}$$

Credit	Income	y	\hat{y}	Previous weight α	New weight α
A	\$130K	Safe	Safe	0.5	
B	\$80K	Risky	Risky	1.5	
C	\$110K	Risky	Safe	1.5	
A	\$110K	Safe	Safe	2	
A	\$90K	Safe	Risky	1	
B	\$120K	Safe	Safe	2.5	
C	\$30K	Risky	Risky	3	
C	\$60K	Risky	Risky	2	
B	\$95K	Safe	Risky	0.5	
A	\$60K	Safe	Risky	1	
A	\$98K	Safe	Risky	0.5	

Updating weights α_i



$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t} = \alpha_i e^{-0.69} = \alpha_i / 2 & , \text{ if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t} = \alpha_i e^{0.69} = 2 \alpha_i & , \text{ if } f_t(x_i) \neq y_i \end{cases}$$

Credit	Income	y	\hat{y}	Previous weight α	New weight α
A	\$130K	Safe	Safe	0.5	
B	\$80K	Risky	Risky	1.5	
C	\$110K	Risky	Safe	1.5	
A	\$110K	Safe	Safe	2	
A	\$90K	Safe	Risky	1	
B	\$120K	Safe	Safe	2.5	
C	\$30K	Risky	Risky	3	
C	\$60K	Risky	Risky	2	
B	\$95K	Safe	Risky	0.5	
A	\$60K	Safe	Risky	1	
A	\$98K	Safe	Risky	0.5	

Updating weights α_i



$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t} = \alpha_i e^{-0.69} = \alpha_i / 2, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t} = \alpha_i e^{0.69} = 2\alpha_i, & \text{if } f_t(x_i) \neq y_i \end{cases}$$

Credit	Income	y	\hat{y}	Previous weight α	New weight α
A	\$130K	Safe	Safe	0.5	0.5/2 = 0.25
B	\$80K	Risky	Risky	1.5	0.75
C	\$110K	Risky	Safe	1.5	2 * 1.5 = 3
A	\$110K	Safe	Safe	2	1
A	\$90K	Safe	Risky	1	2
B	\$120K	Safe	Safe	2.5	1.25
C	\$30K	Risky	Risky	3	1.5
C	\$60K	Risky	Risky	2	1
B	\$95K	Safe	Risky	0.5	1
A	\$60K	Safe	Risky	1	2
A	\$98K	Safe	Risky	0.5	1

Python Code

```
from sklearn.ensemble import AdaBoostClassifier #For Classification  
from sklearn.ensemble import AdaBoostRegressor #For Regression  
from sklearn.tree import DecisionTreeClassifier
```

```
dt = DecisionTreeClassifier()  
clf = AdaBoostClassifier(n_estimators=100, base_estimator=dt, learning_rate=1)  
#Above I have used decision tree as a base estimator, you can use any ML learner as base estimator if it accepts sample weight  
clf.fit(x_train,y_train)
```

You can tune the parameters to optimize the performance of algorithms, I've mentioned below the key parameters for tuning:

- **n_estimators**: It controls the number of weak learners.
- **learning_rate**: Controls the contribution of weak learners in the final combination. There is a trade-off between **learning_rate** and **n_estimators**.
- **base_estimators**: It helps to specify different ML algorithm.

You can also tune the parameters of base learners to optimize its performance.

Gradient Boosting

```
Y = M(x) + error
```

```
error = G(x) + error2
```

```
error2 = H(x) + error3
```

```
Y = M(x) + G(x) + H(x) + error3
```

```
Y = alpha * M(x) + beta * G(x) + gamma * H(x) + error4
```

Boosting Advantage

Boosting (machine learning)

Boosting is a machine learning ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms which convert weak learners to strong ones.

Boosting Disadvantage

Time and computation expensive.

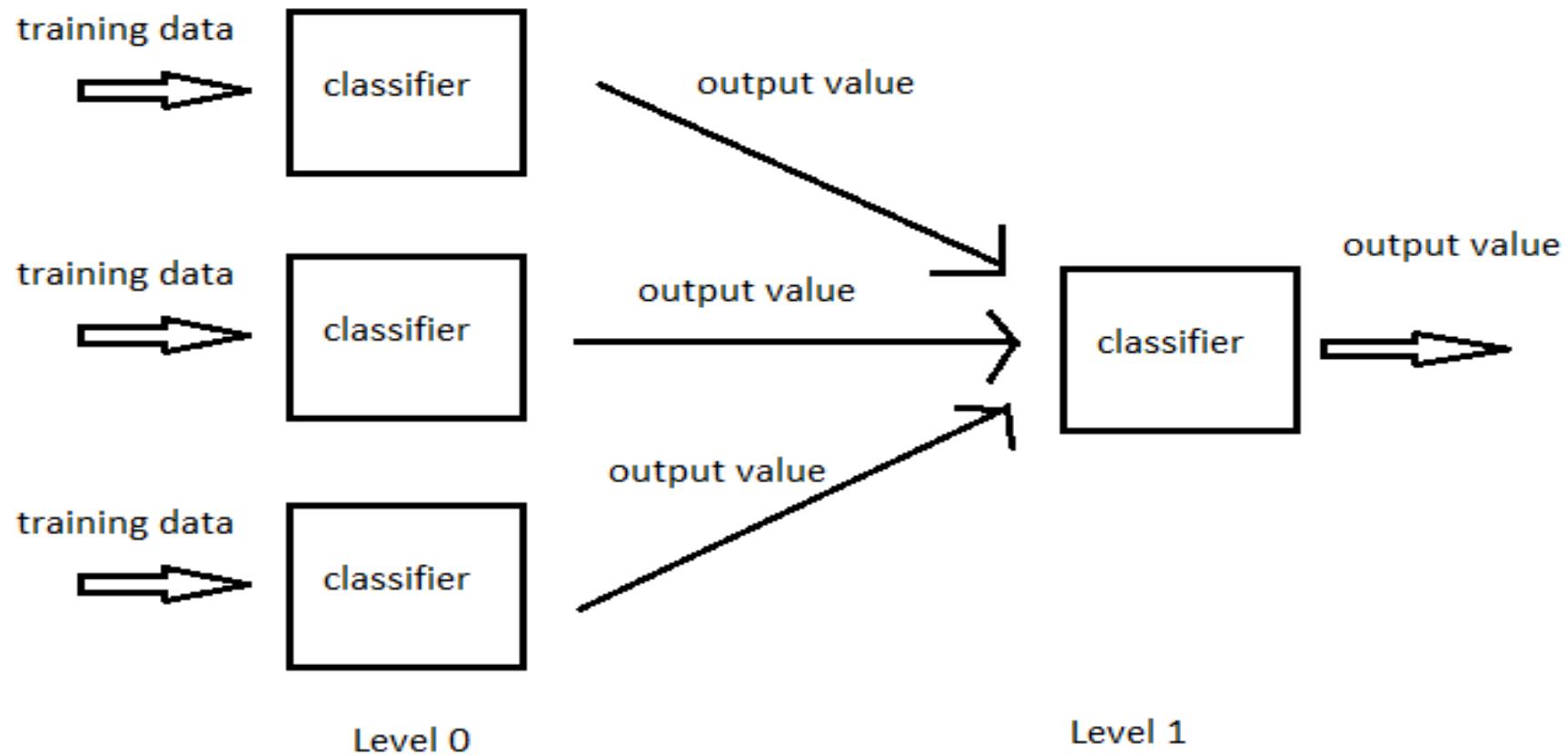
Hard to implement in real time platform.

Complexity of the classification increases.

Stacking

Competition winning algorithm

Concept Diagram of Stacking



```
# Caution! All models and parameter values are just
# demonstrational and shouldn't be considered as recommended.
# Initialize 1st level models.
models = [
    ExtraTreesClassifier(random_state = 0, n_jobs = -1,
        n_estimators = 100, max_depth = 3),
    RandomForestClassifier(random_state = 0, n_jobs = -1,
        n_estimators = 100, max_depth = 3),
    XGBClassifier(seed = 0, n_jobs = -1, learning_rate = 0.1,
        n_estimators = 100, max_depth = 3)]
```

```
# Compute stacking features
S_train, S_test = stacking(models, X_train, y_train, X_test,
    regression = False, metric = accuracy_score, n_folds = 4,
    stratified = True, shuffle = True, random_state = 0, verbose = 2)

# Initialize 2nd level model
model = XGBClassifier(seed = 0, n_jobs = -1, learning_rate = 0.1,
    n_estimators = 100, max_depth = 3)
```

```
# Fit 2nd level model
model = model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Final prediction score
print('Final prediction score: {:.8f}' % accuracy_score(y_test, y_pred))
```

```
task: [classification]
metric: [accuracy_score]

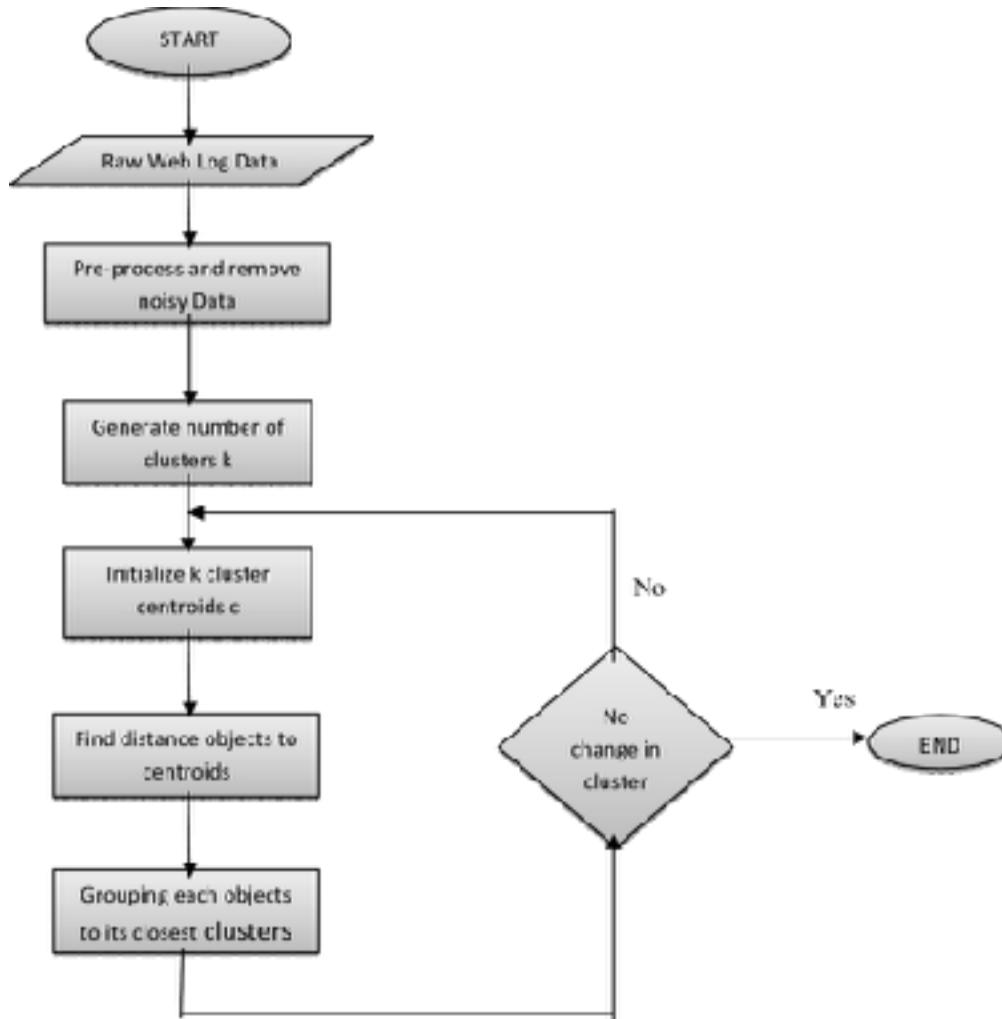
model 0: [ExtraTreesClassifier]
    fold 0: [0.93548387]
    fold 1: [0.96666667]
    fold 2: [1.00000000]
    fold 3: [0.89655172]
    -----
    MEAN: [0.95000000]

model 1: [RandomForestClassifier]
    fold 0: [0.87096774]
    fold 1: [0.96666667]
    fold 2: [1.00000000]
    fold 3: [0.93103440]
    -----
    MEAN: [0.94166667]

model 2: [XGBClassifier]
    fold 0: [0.83870968]
    fold 1: [0.93333333]
    fold 2: [1.00000000]
    fold 3: [0.93103440]
    -----
    MEAN: [0.92500000]

Final prediction score: [0.96666667]
```

K-Means Clustering

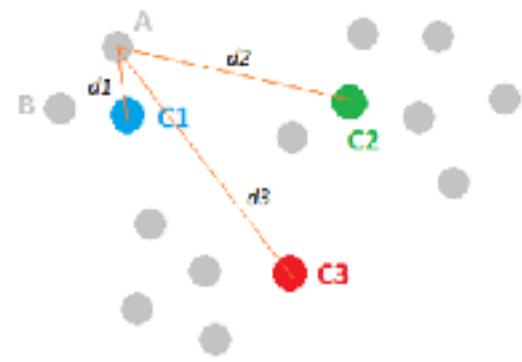
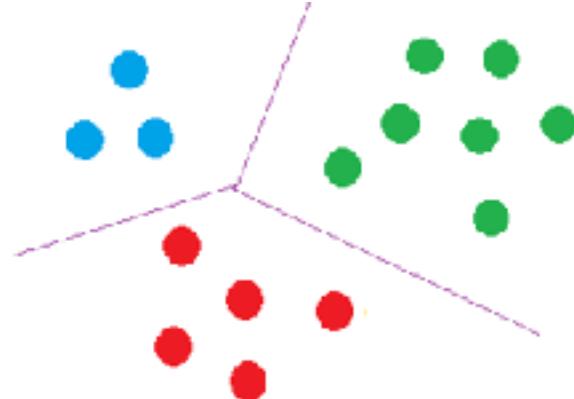
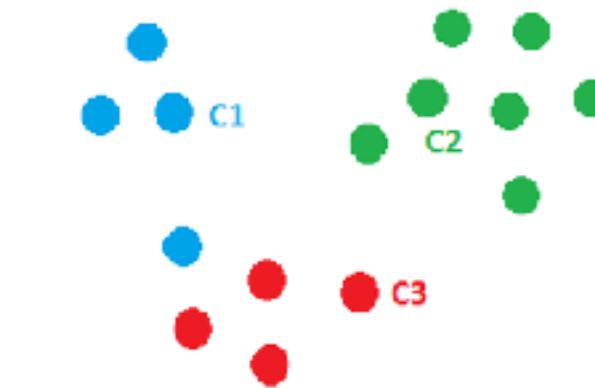
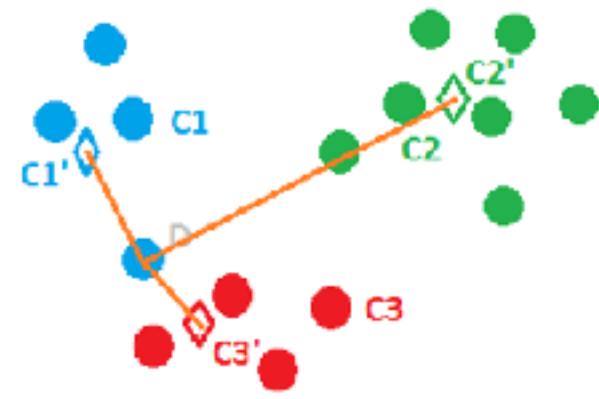


Step one: Initialize cluster centers

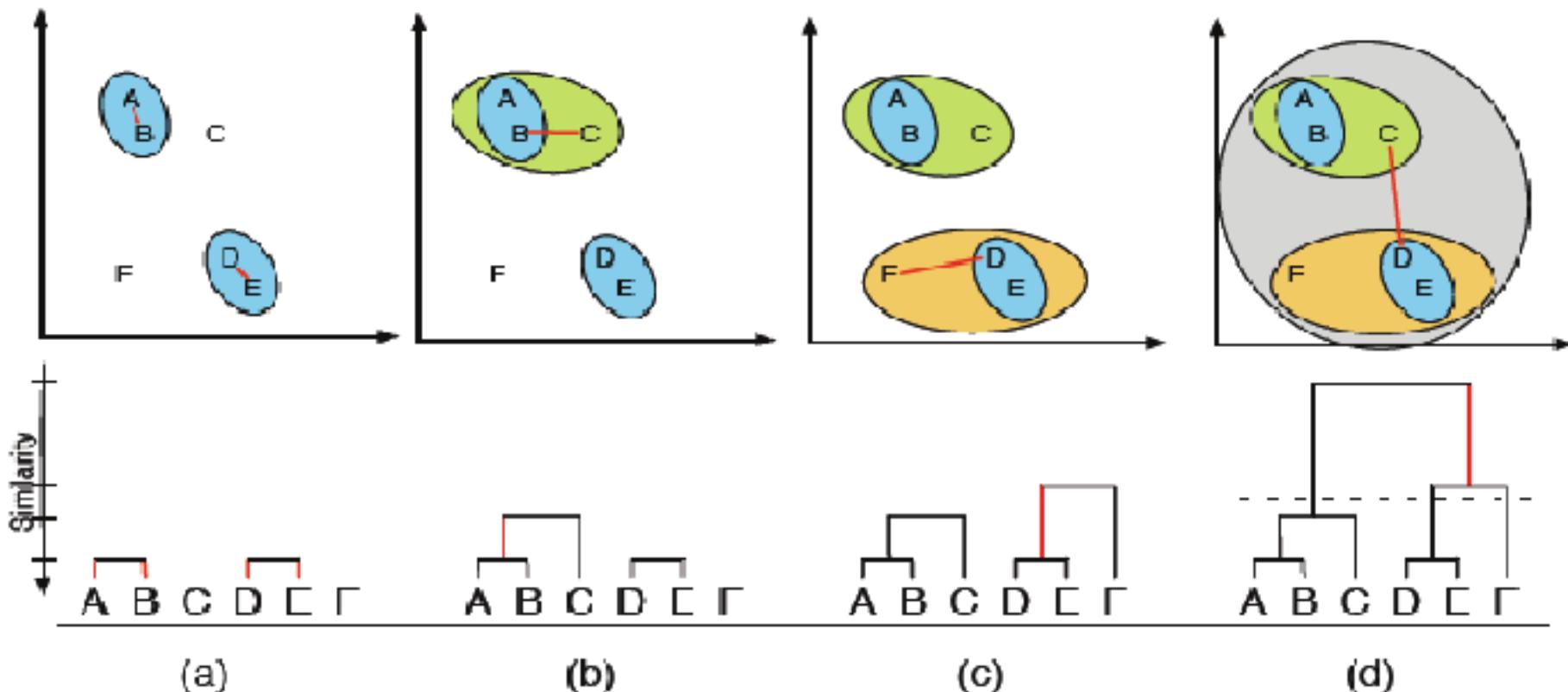
Step two: Assign observations to the closest cluster center

Step three: Revise cluster centers as mean of assigned observations

Step four: Repeat step 2 and step 3 until convergence



Example: Hierarchical Agglomerative Clustering



Support Vector Machine

Agenda

1. SVM Basics

- Visual introduction
- Example in Python

2. Additional Complexities

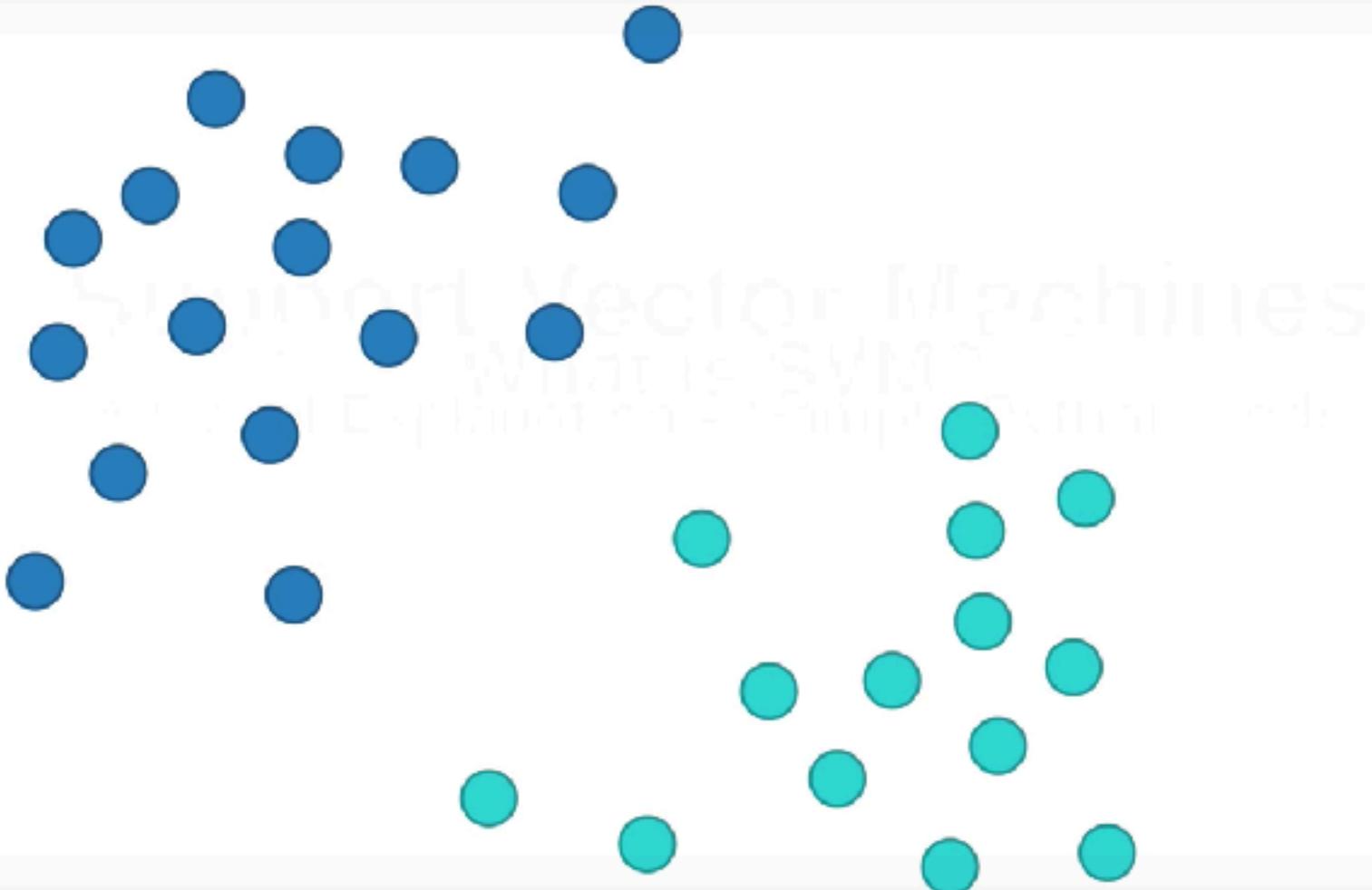
- Higher dimensions
- Multiple classes
- C parameter
- Kernel trick

3. Closing Remarks

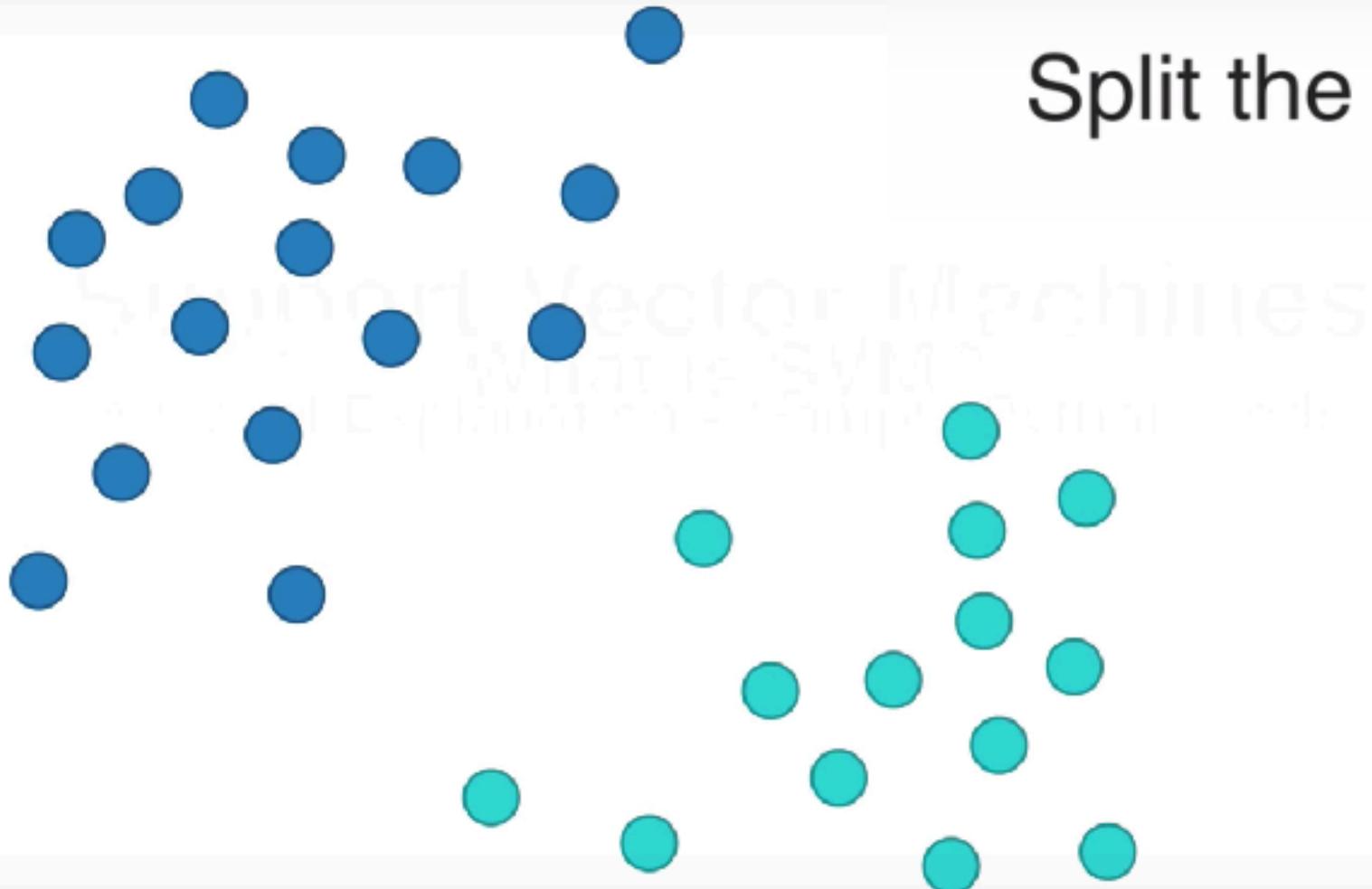
- Pros and cons
- Other techniques

What is SVM?

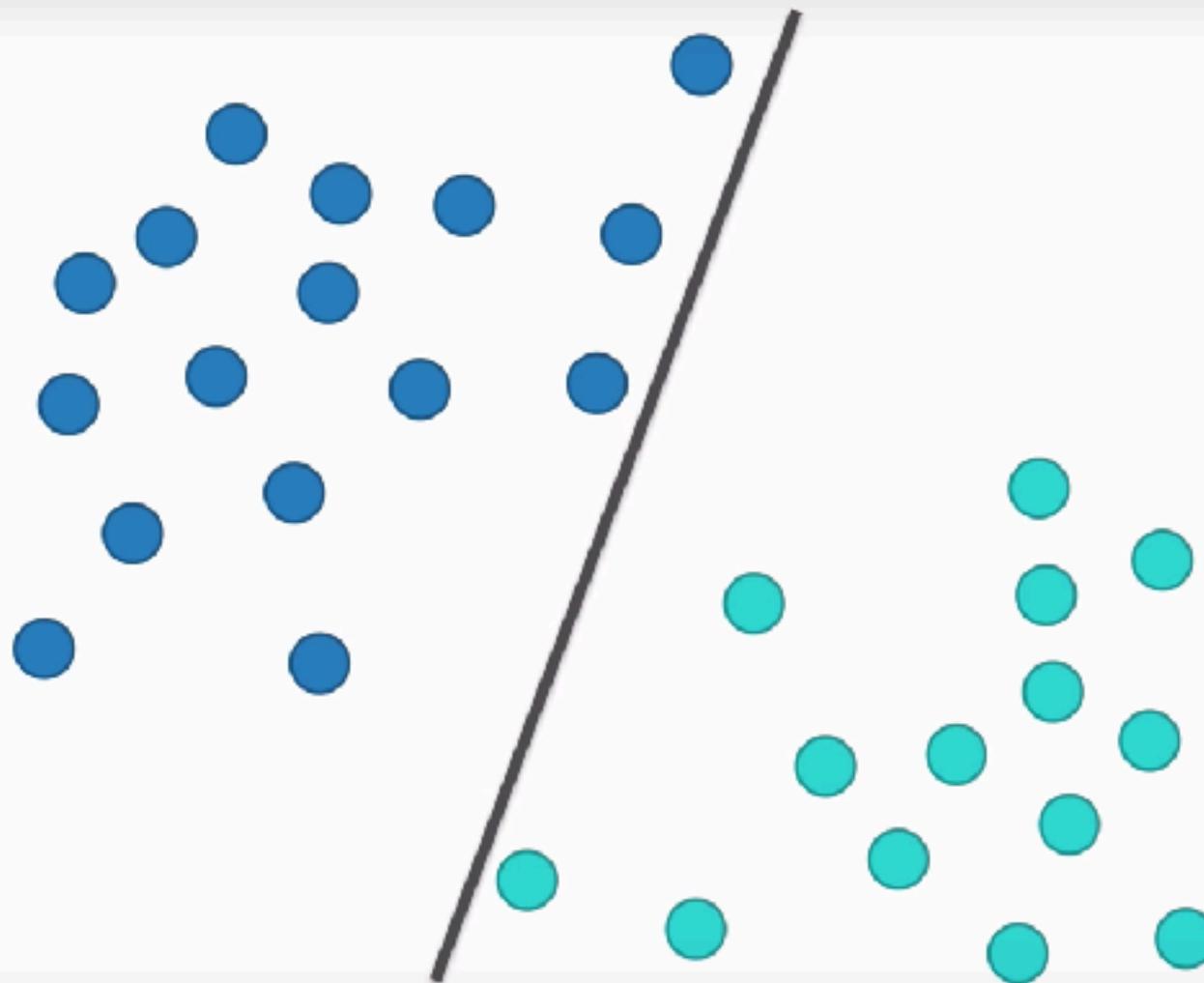
It's a classification technique.



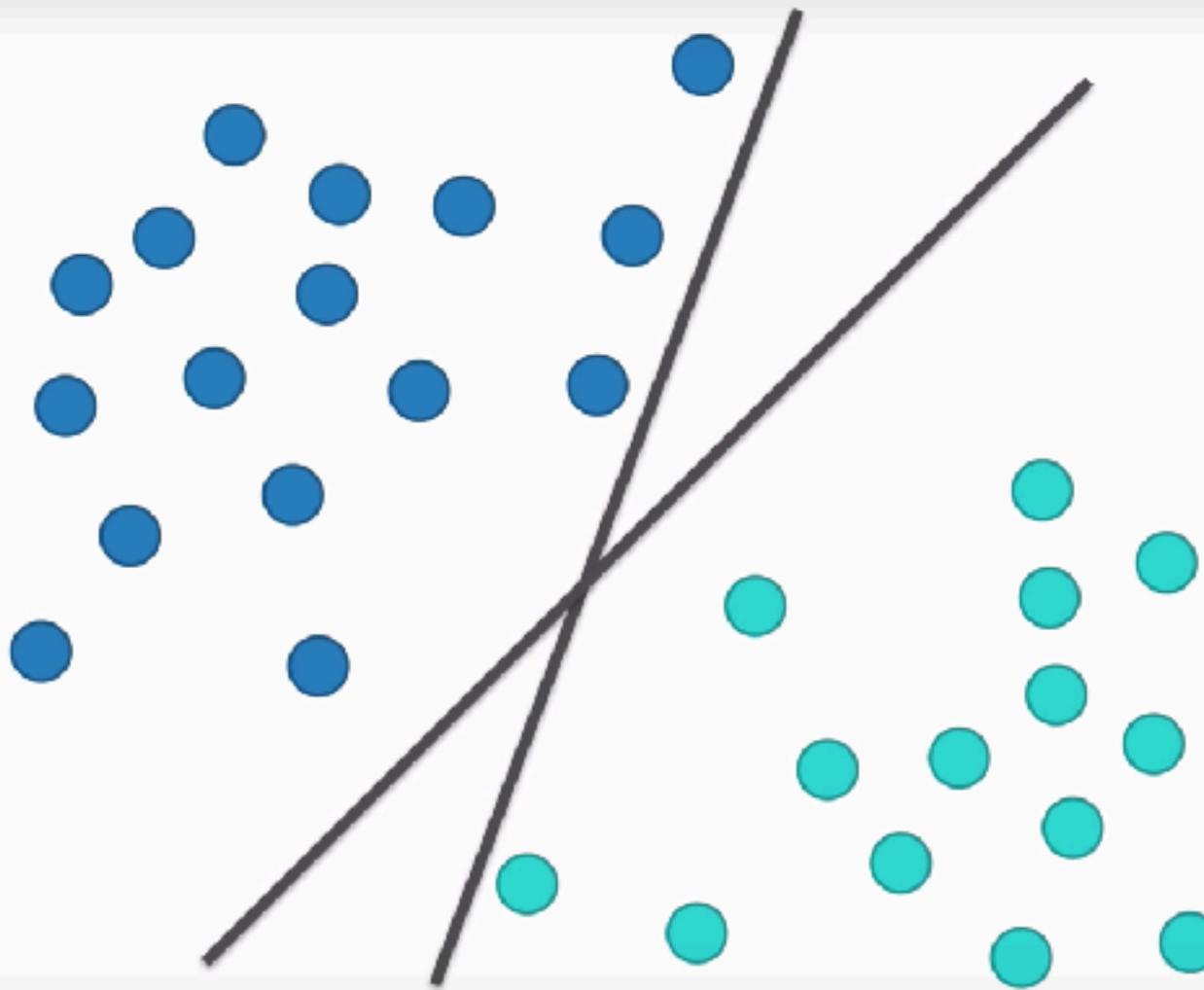
Split the data



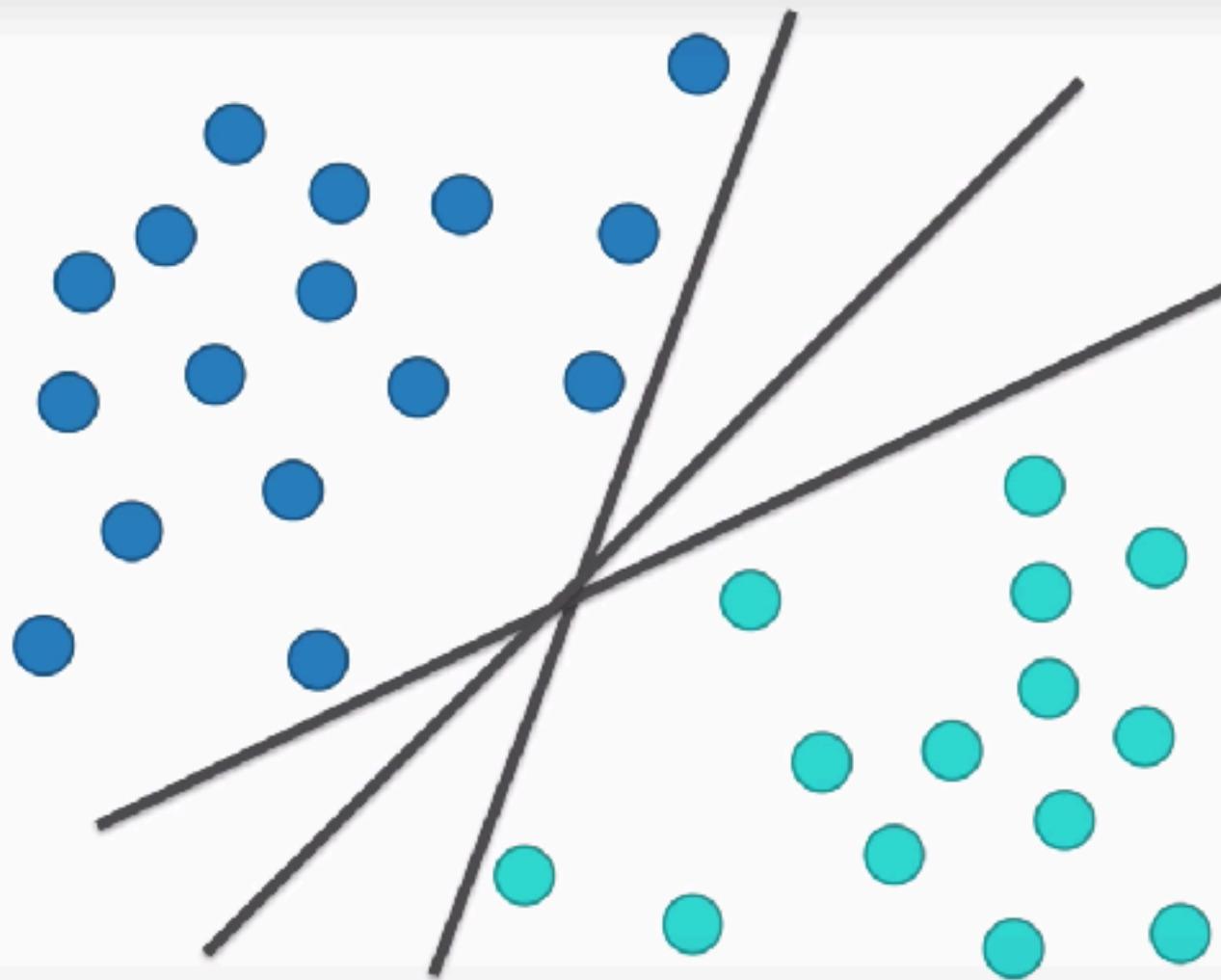
Split the data

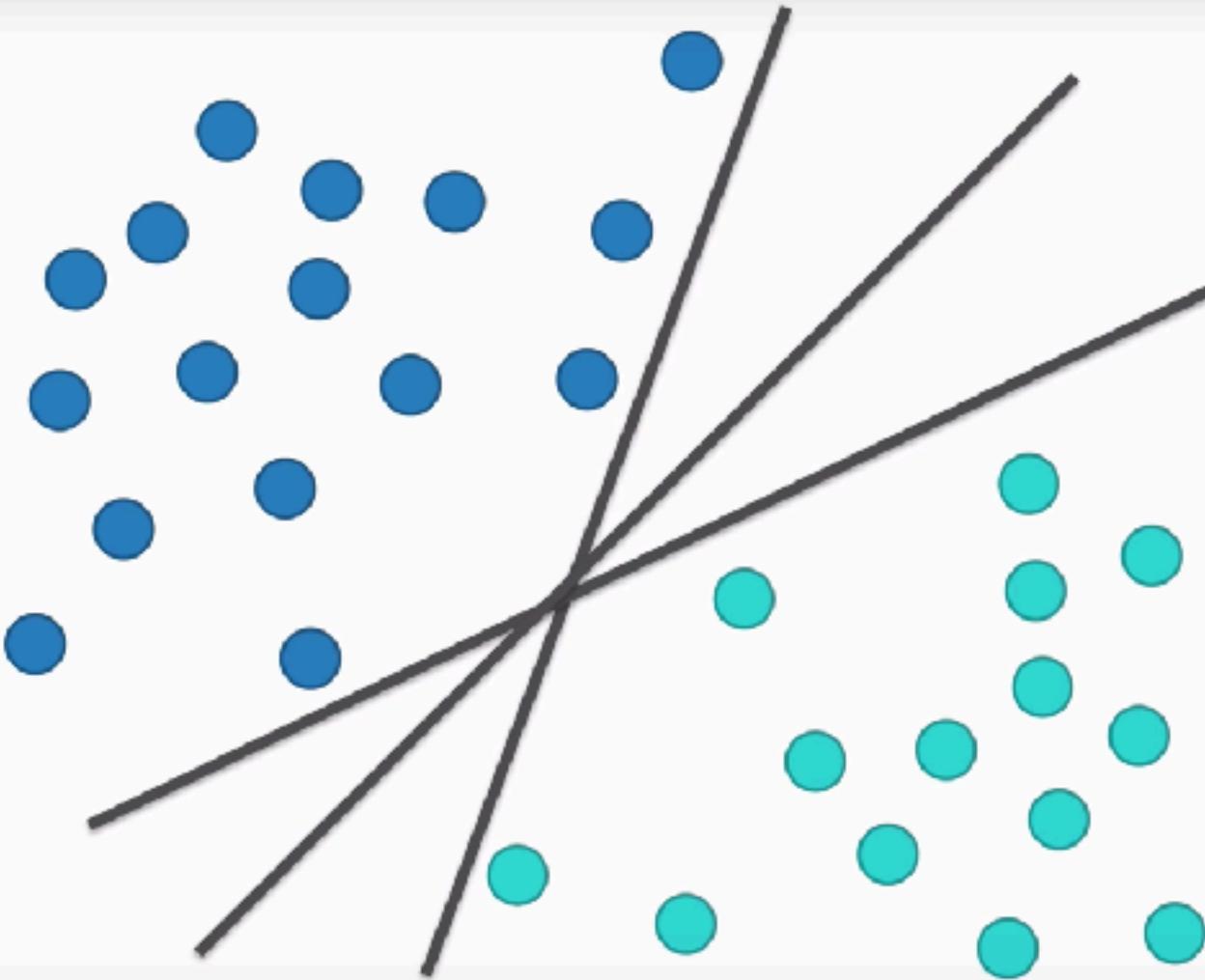


Split the data

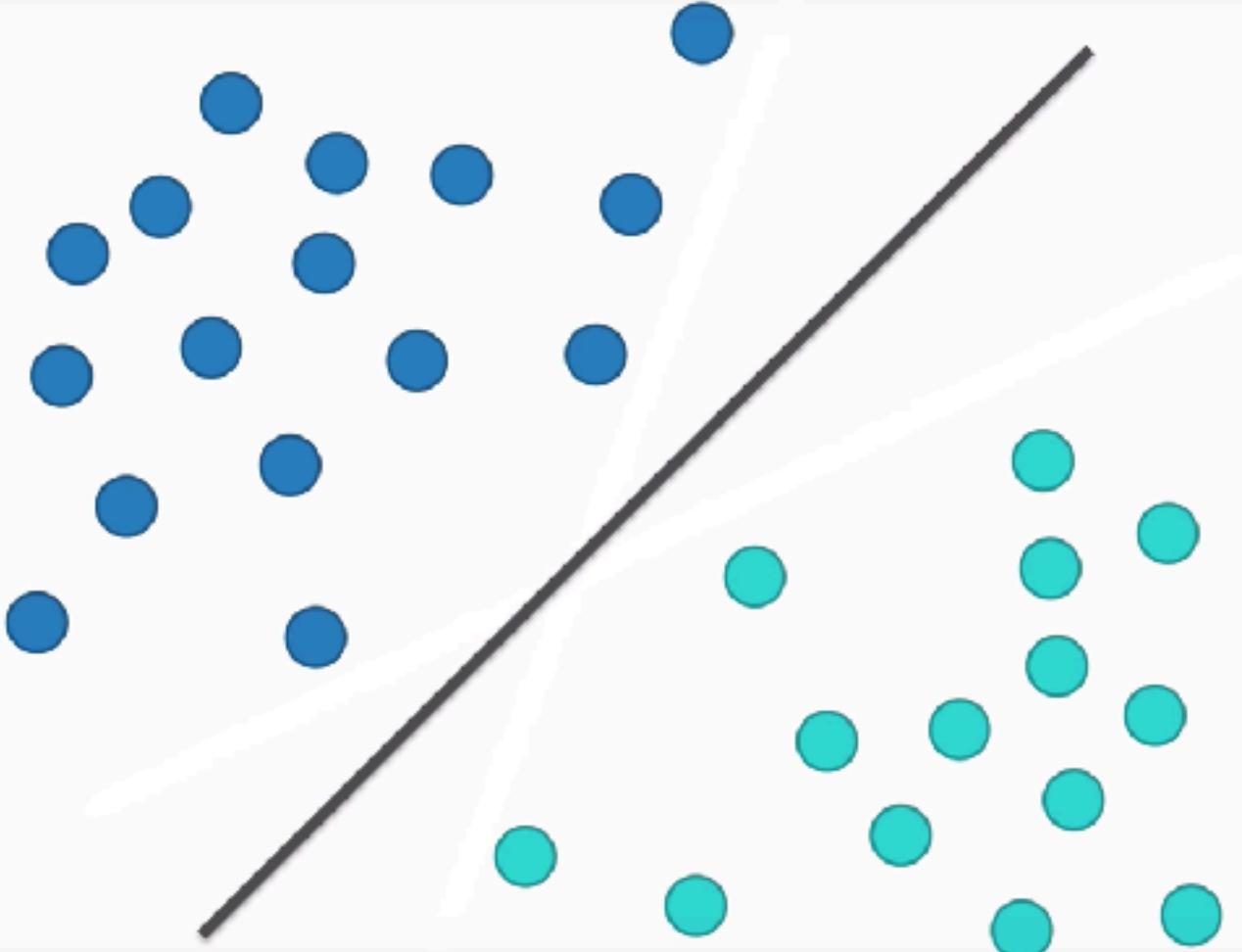


Split the data



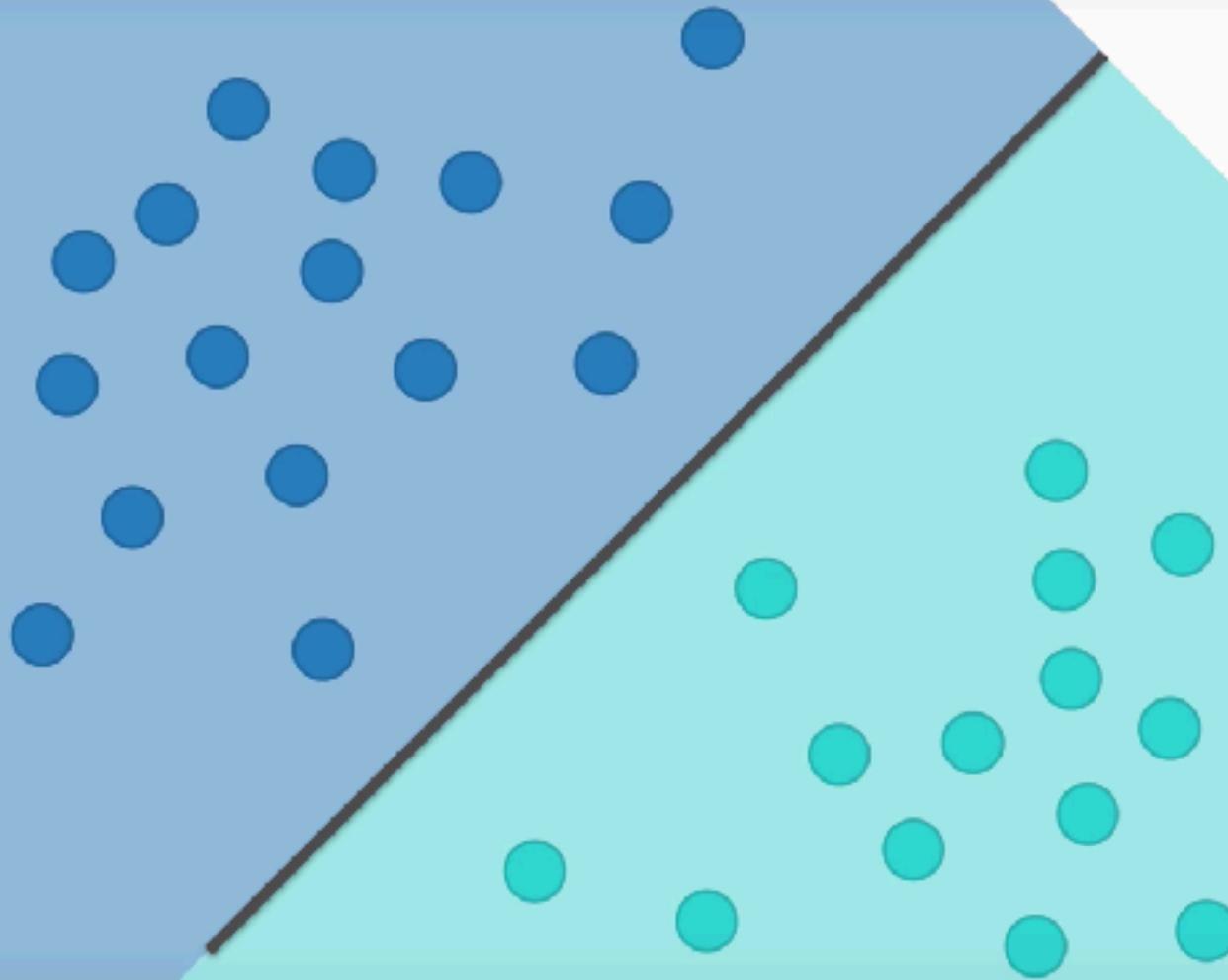


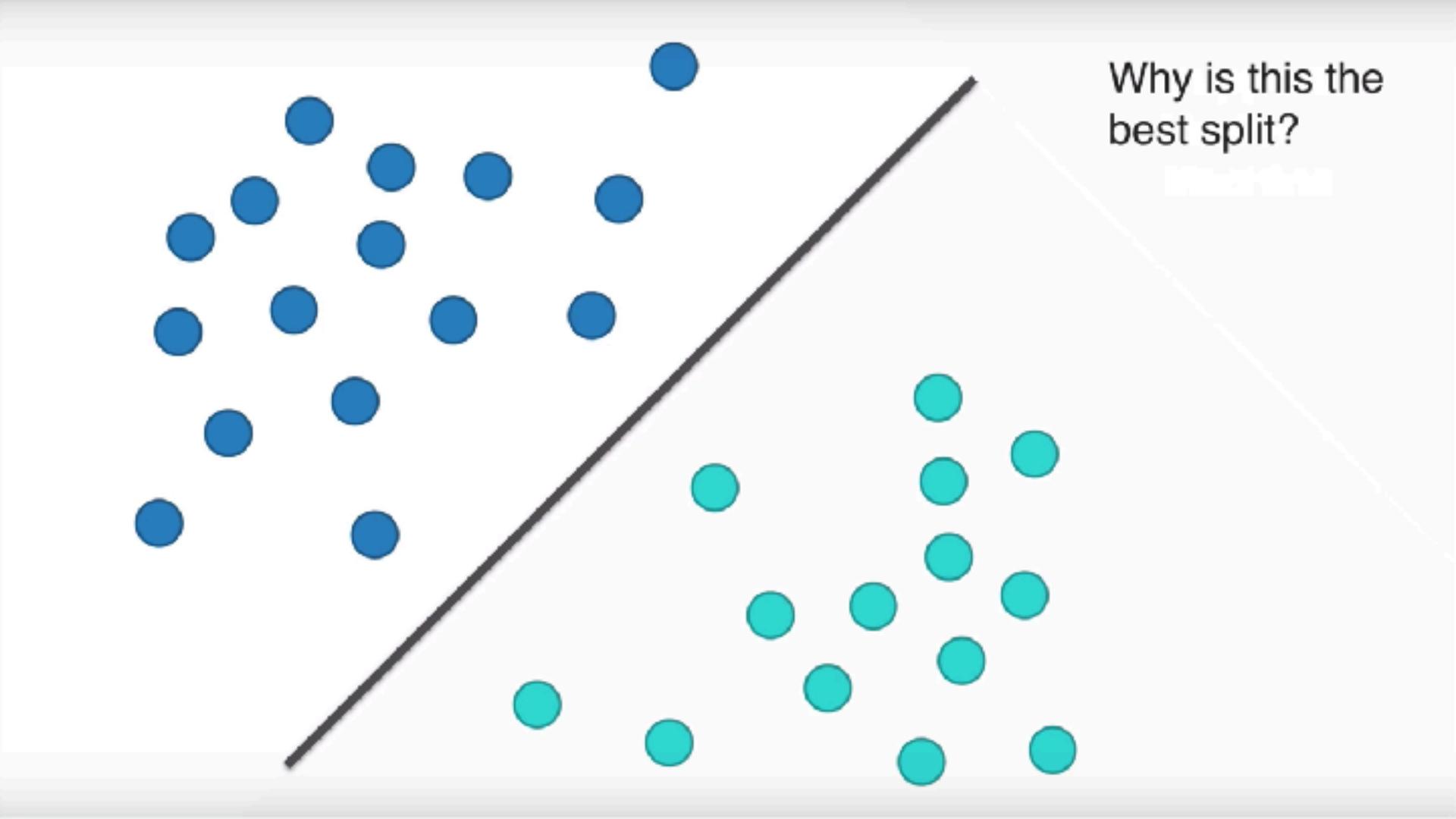
Split the data
in the best
possible way



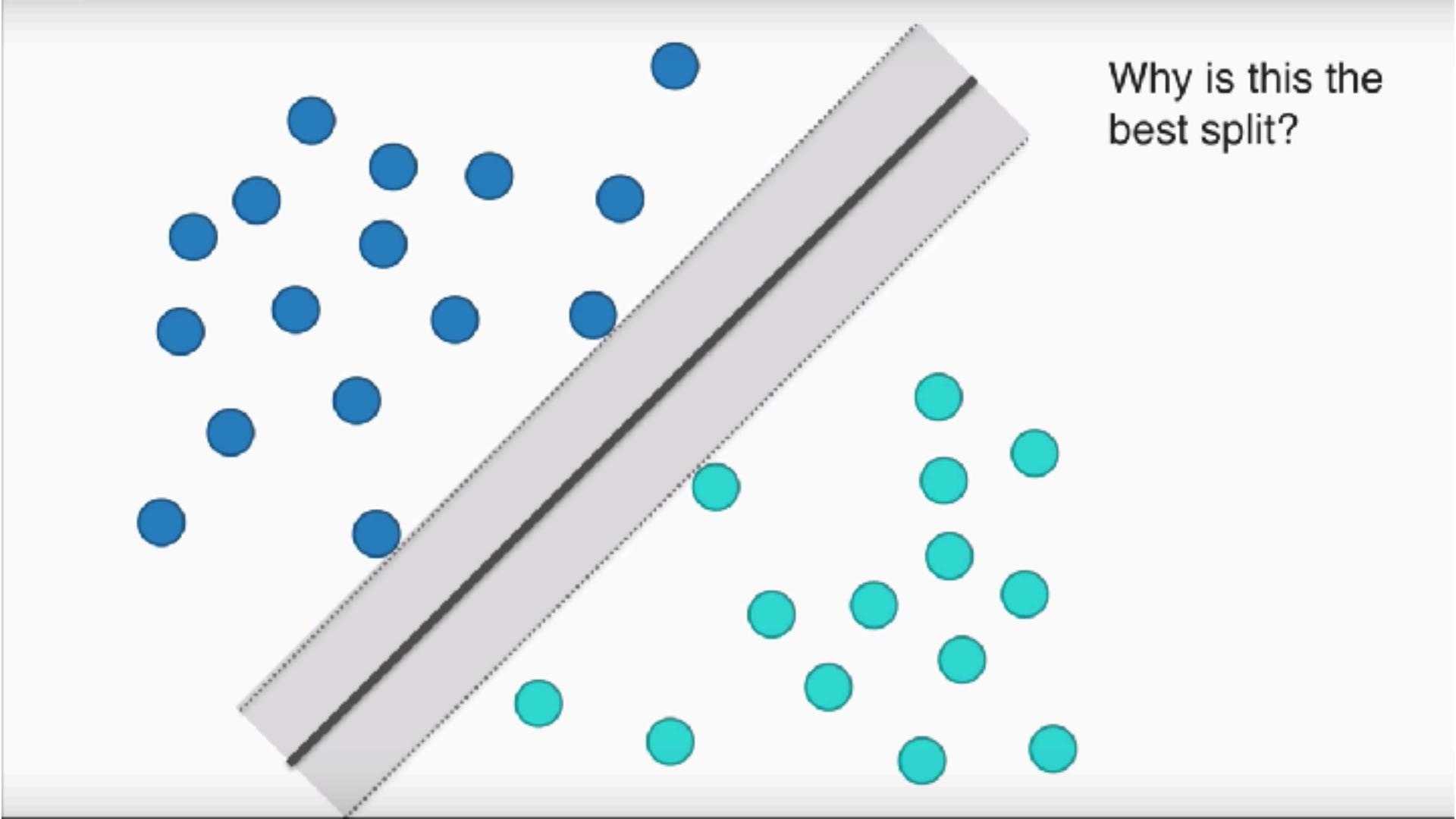
**Split the data
in the best
possible way**

Support Vector Machine

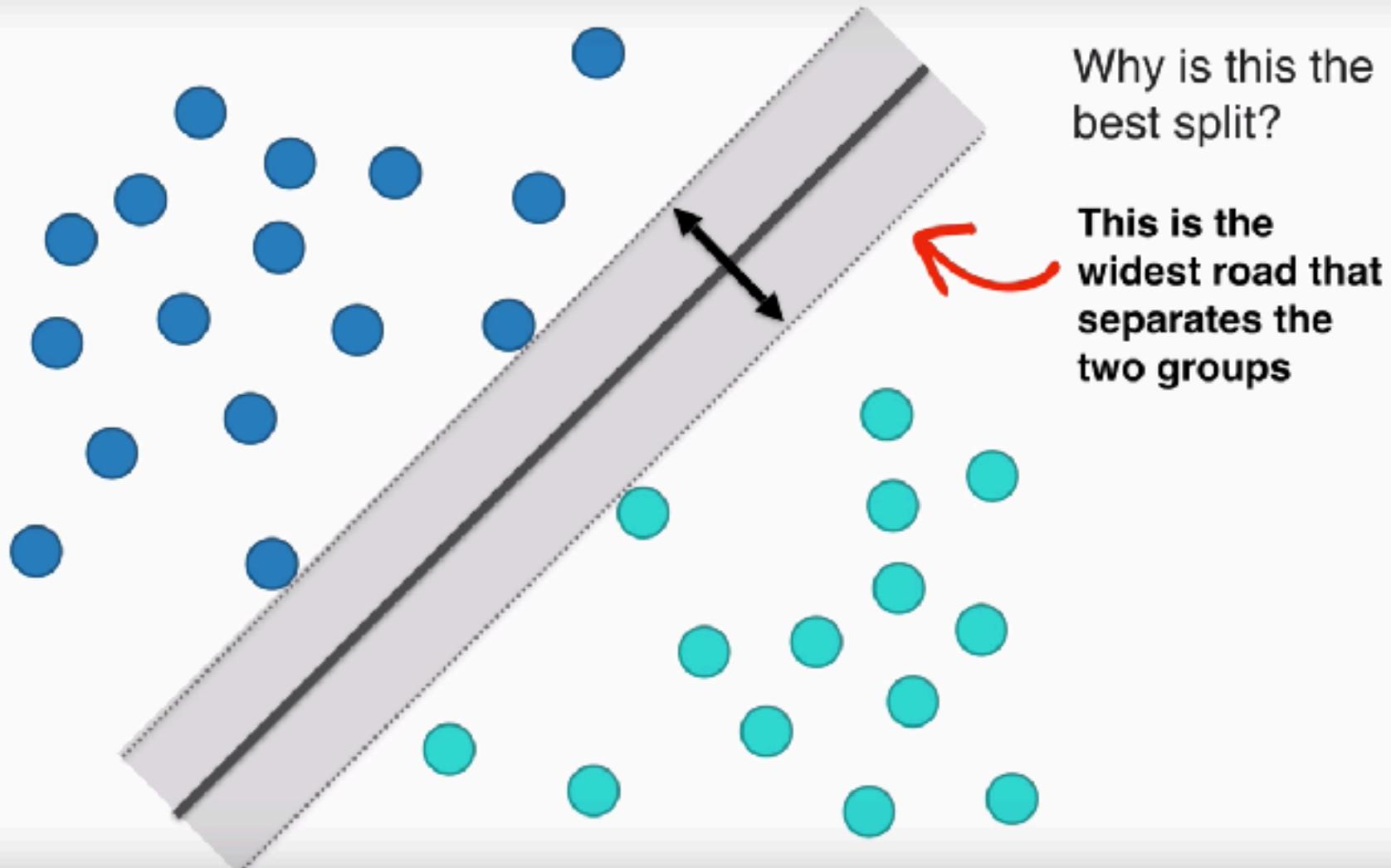




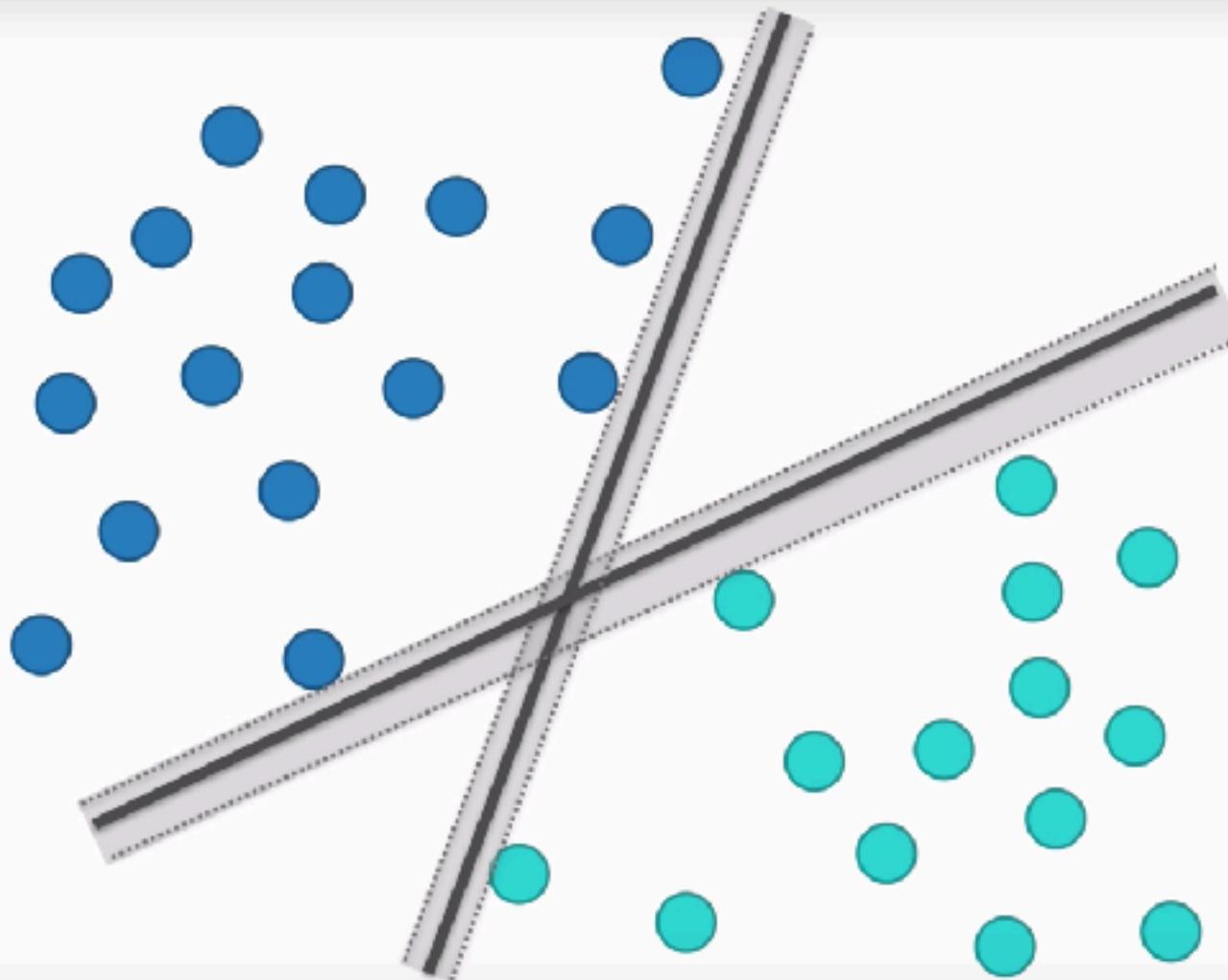
Why is this the
best split?



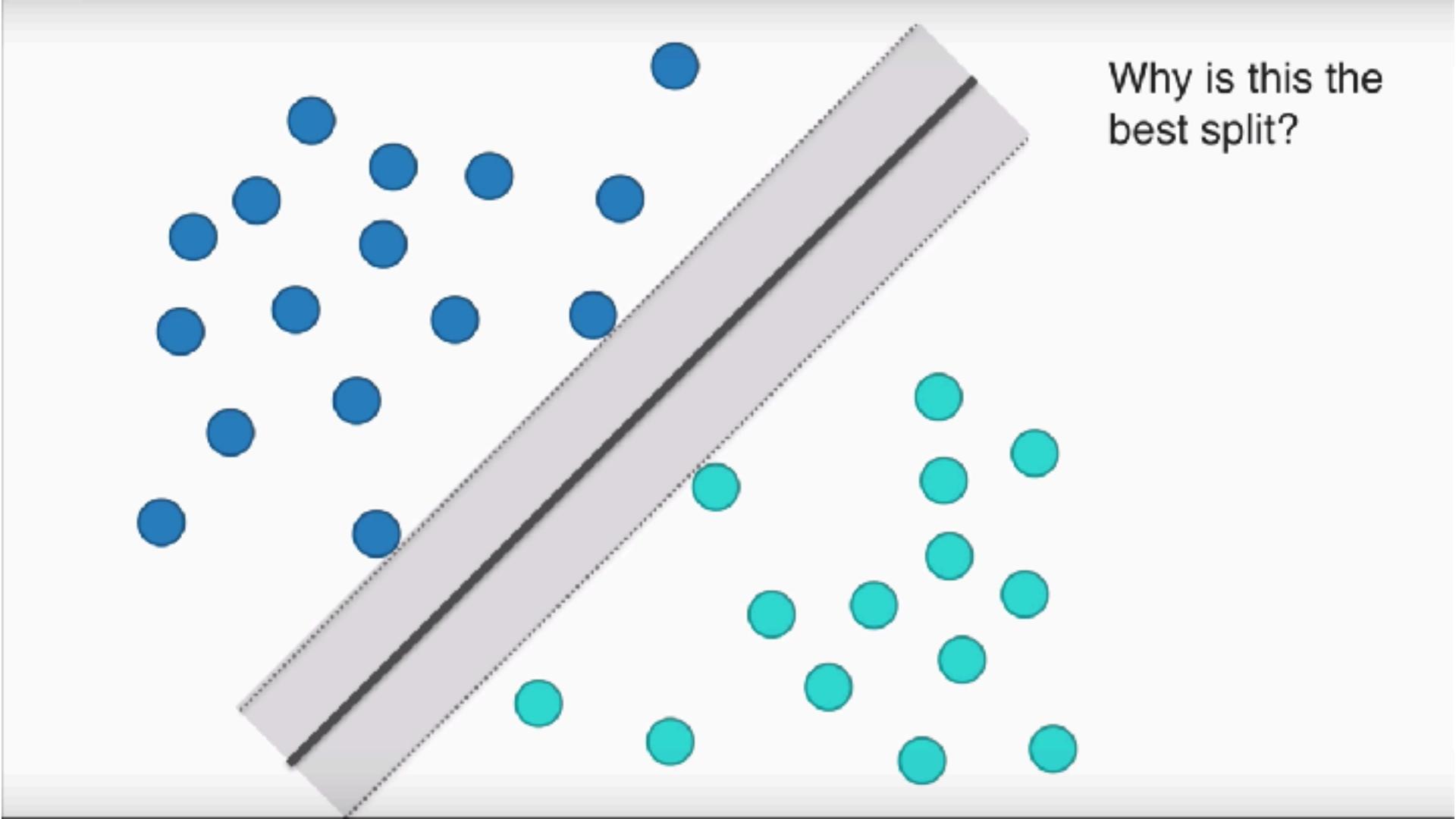
Why is this the
best split?



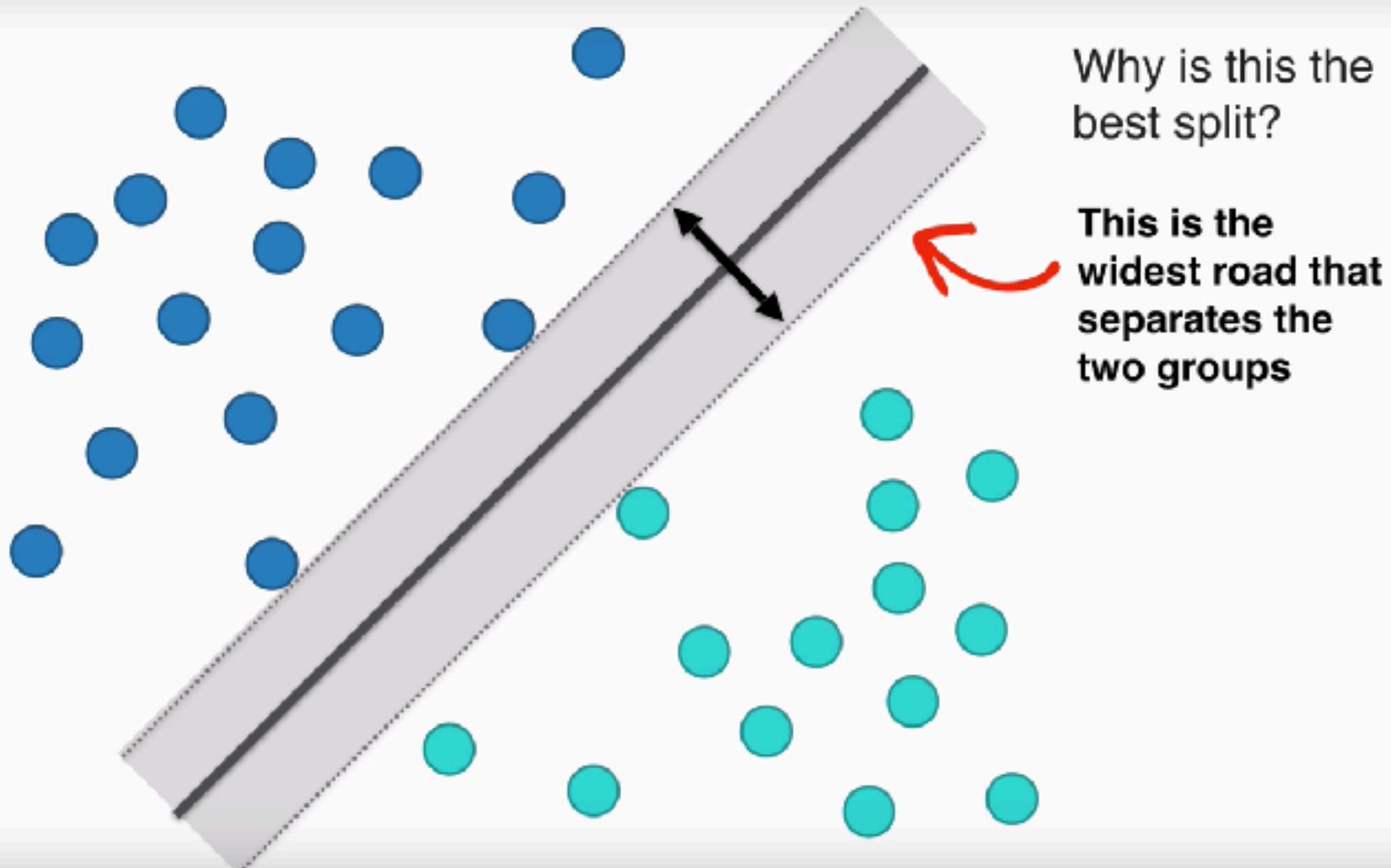
Why is this the
best split?

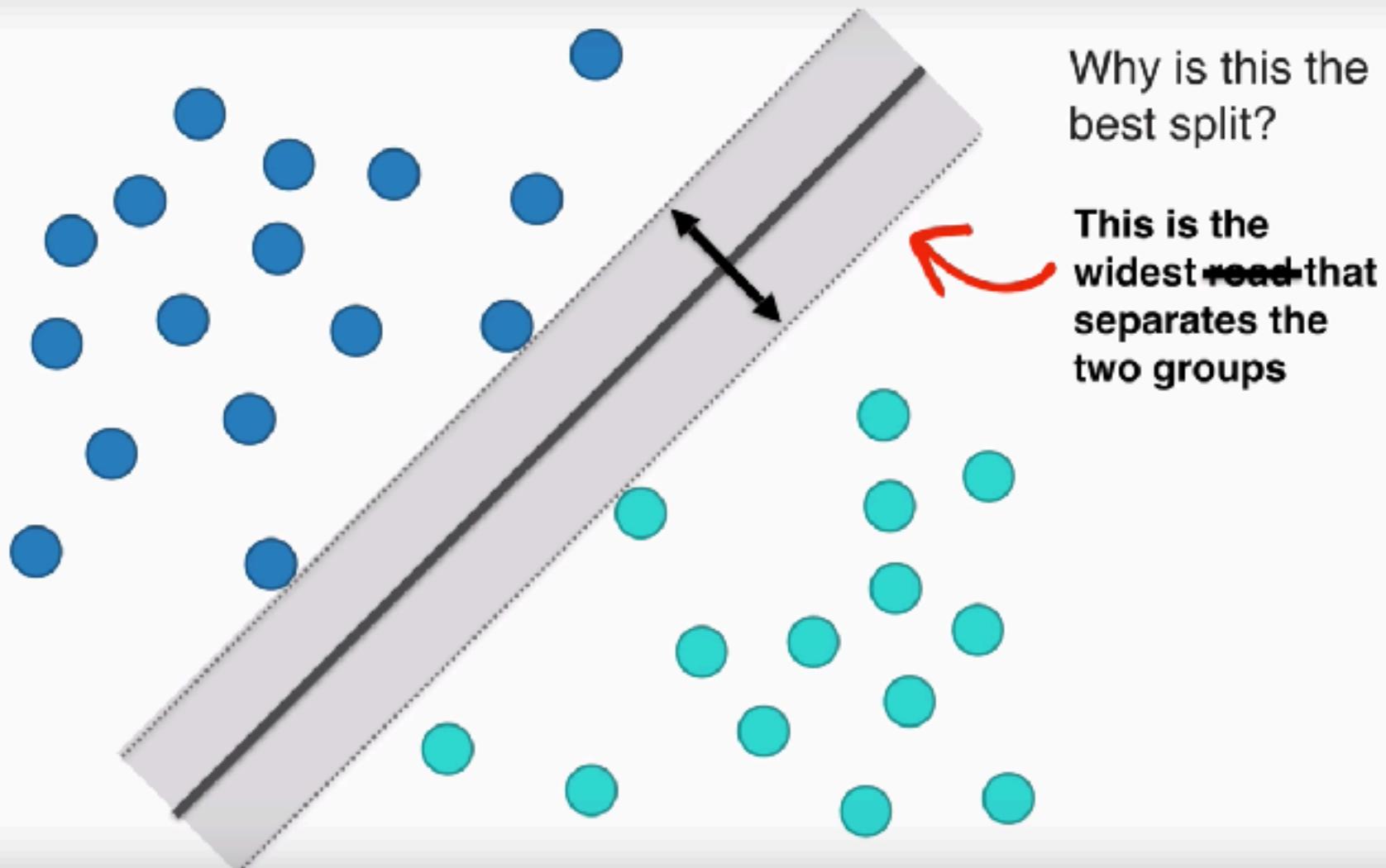


Why is this the
best split?



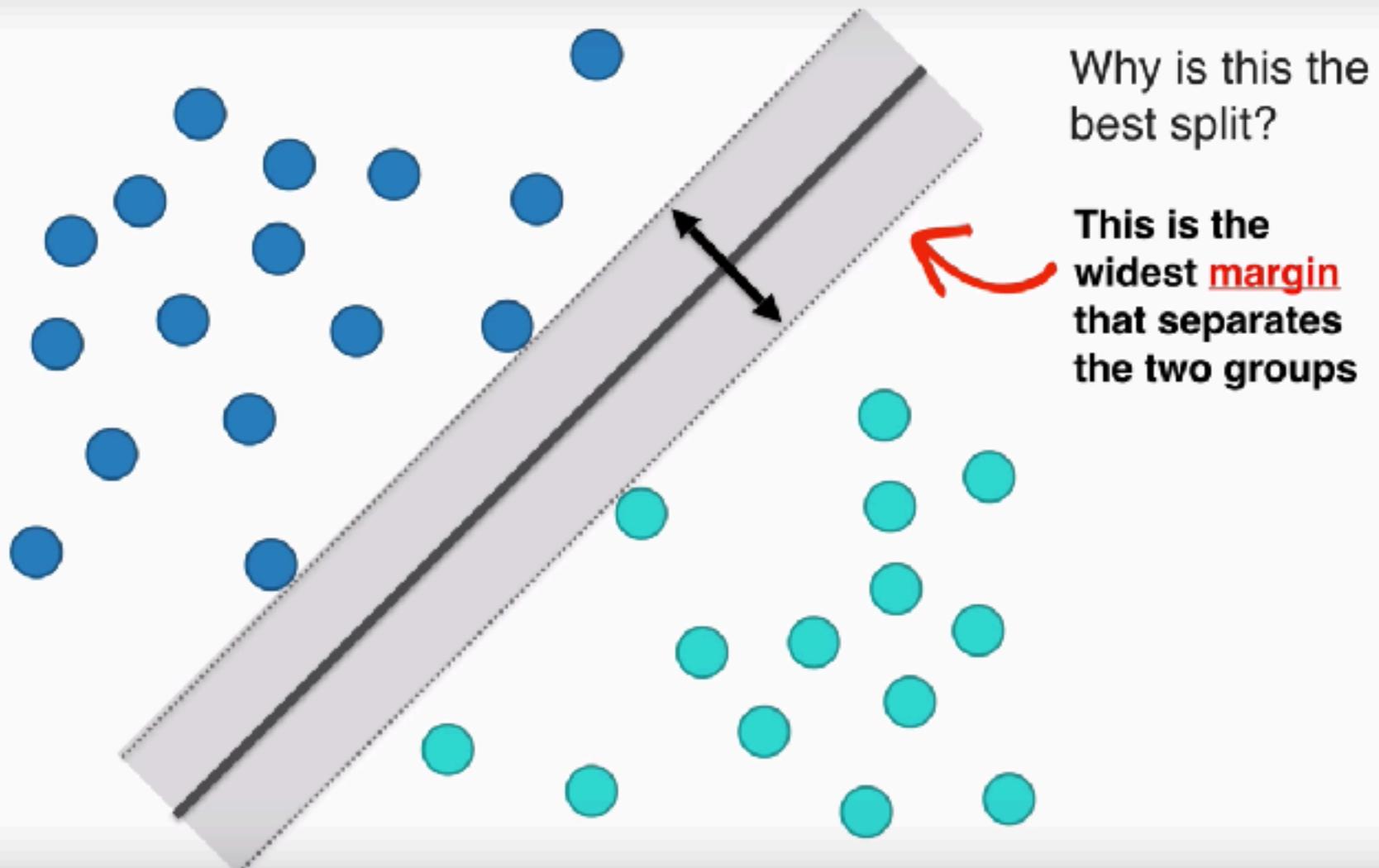
Why is this the
best split?





Why is this the
best split?

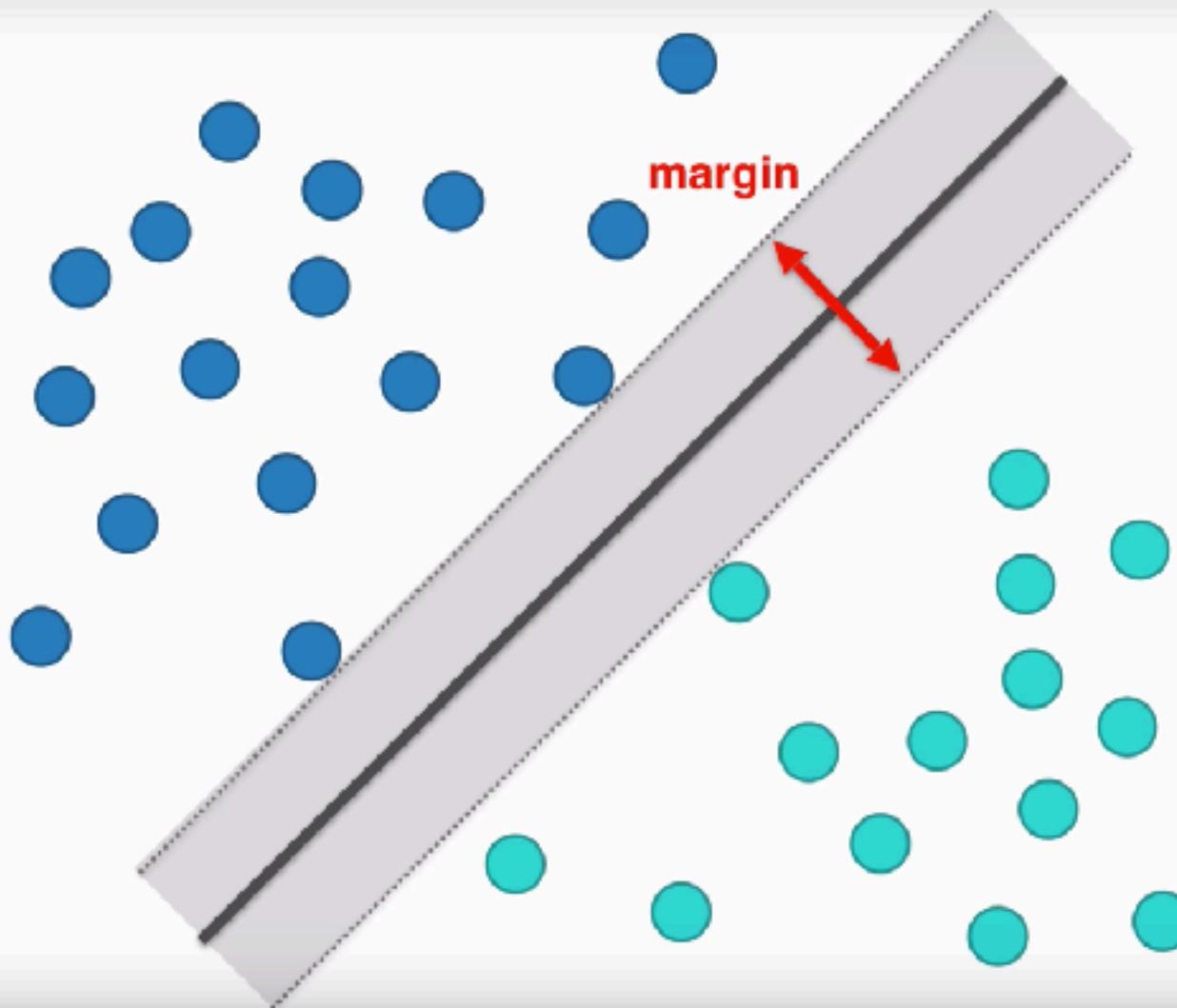
This is the
widest ~~width~~
that
separates the
two groups

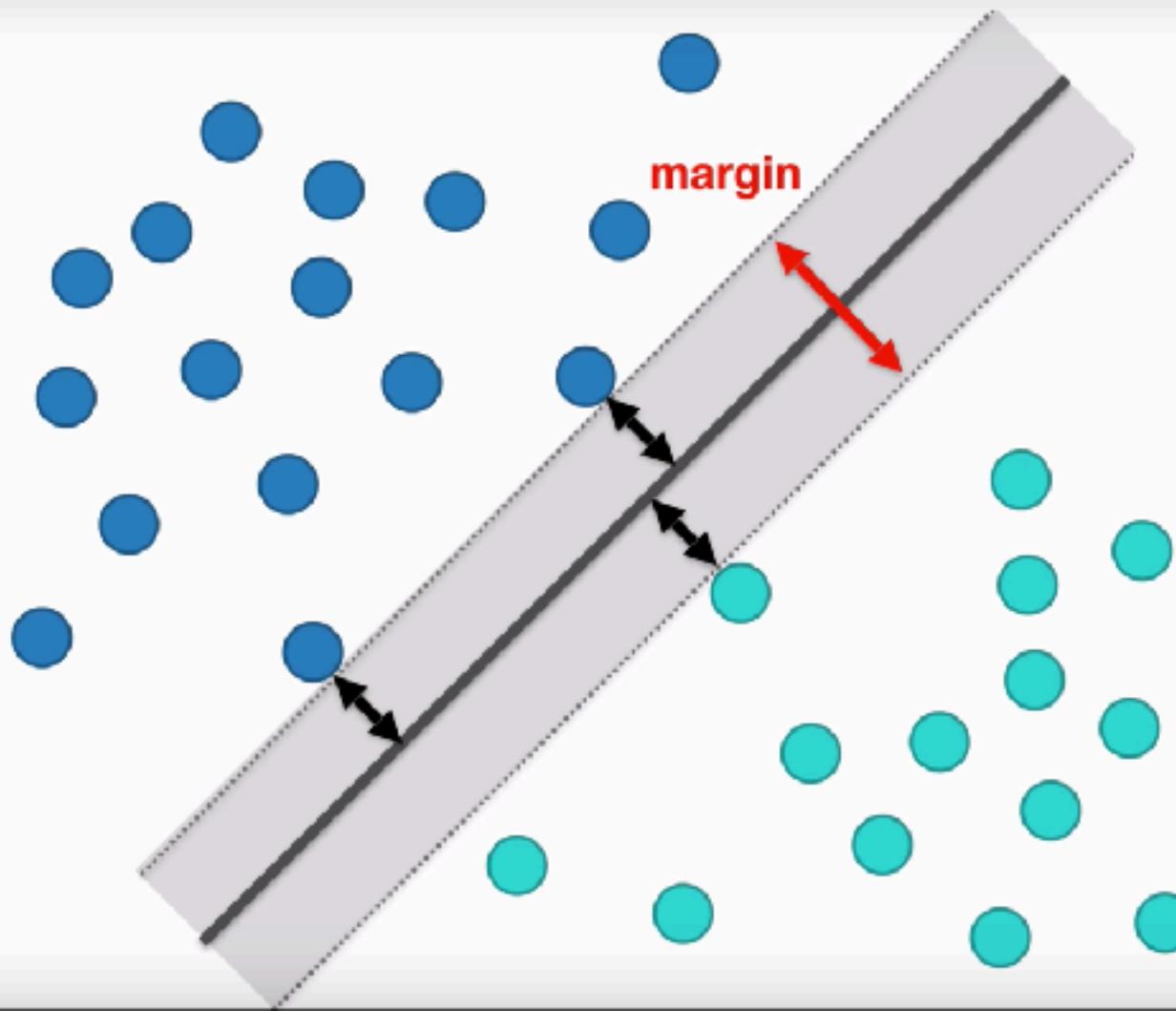


Why is this the best split?

This is the widest margin that separates the two groups

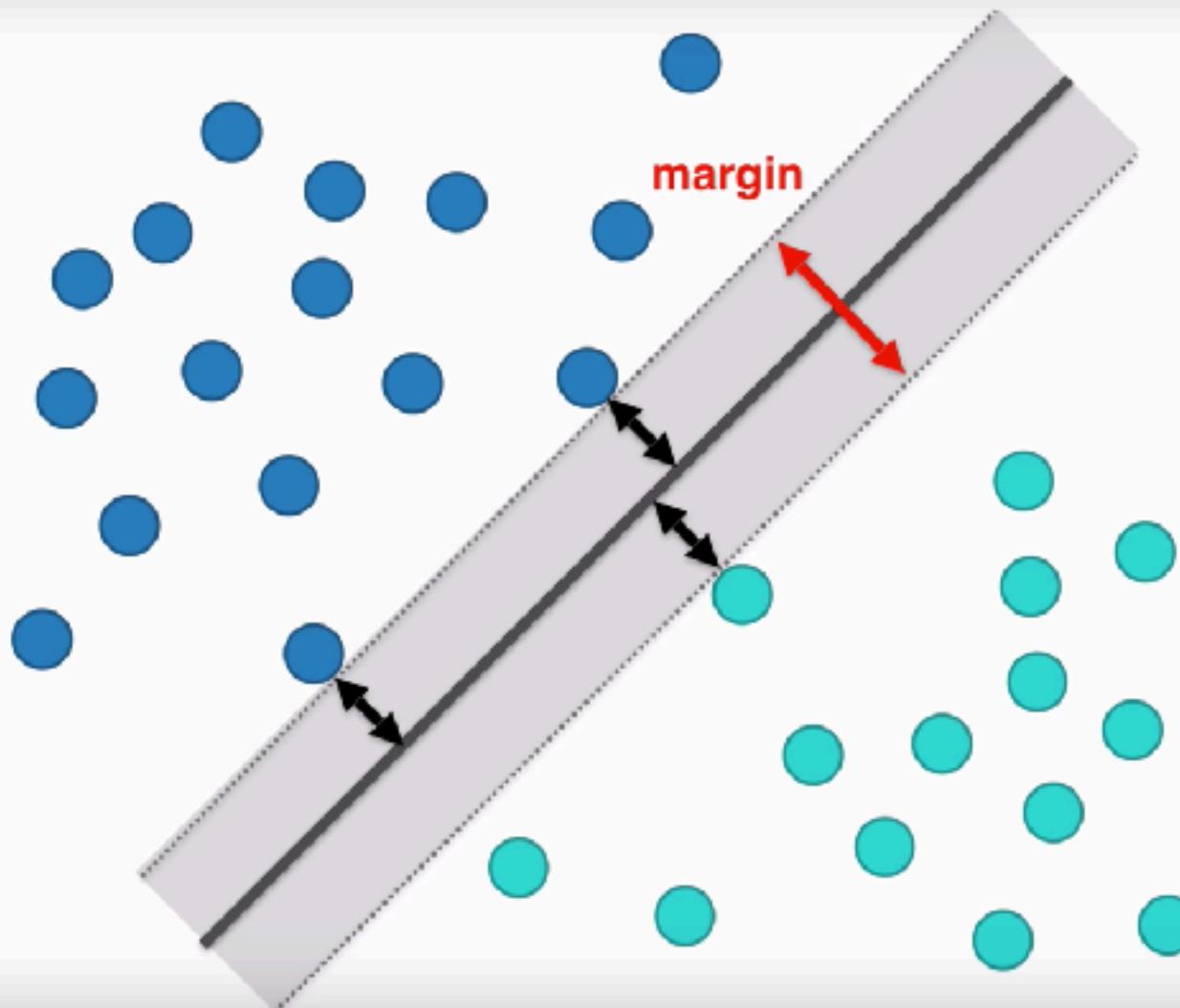
Why is this the
best split?





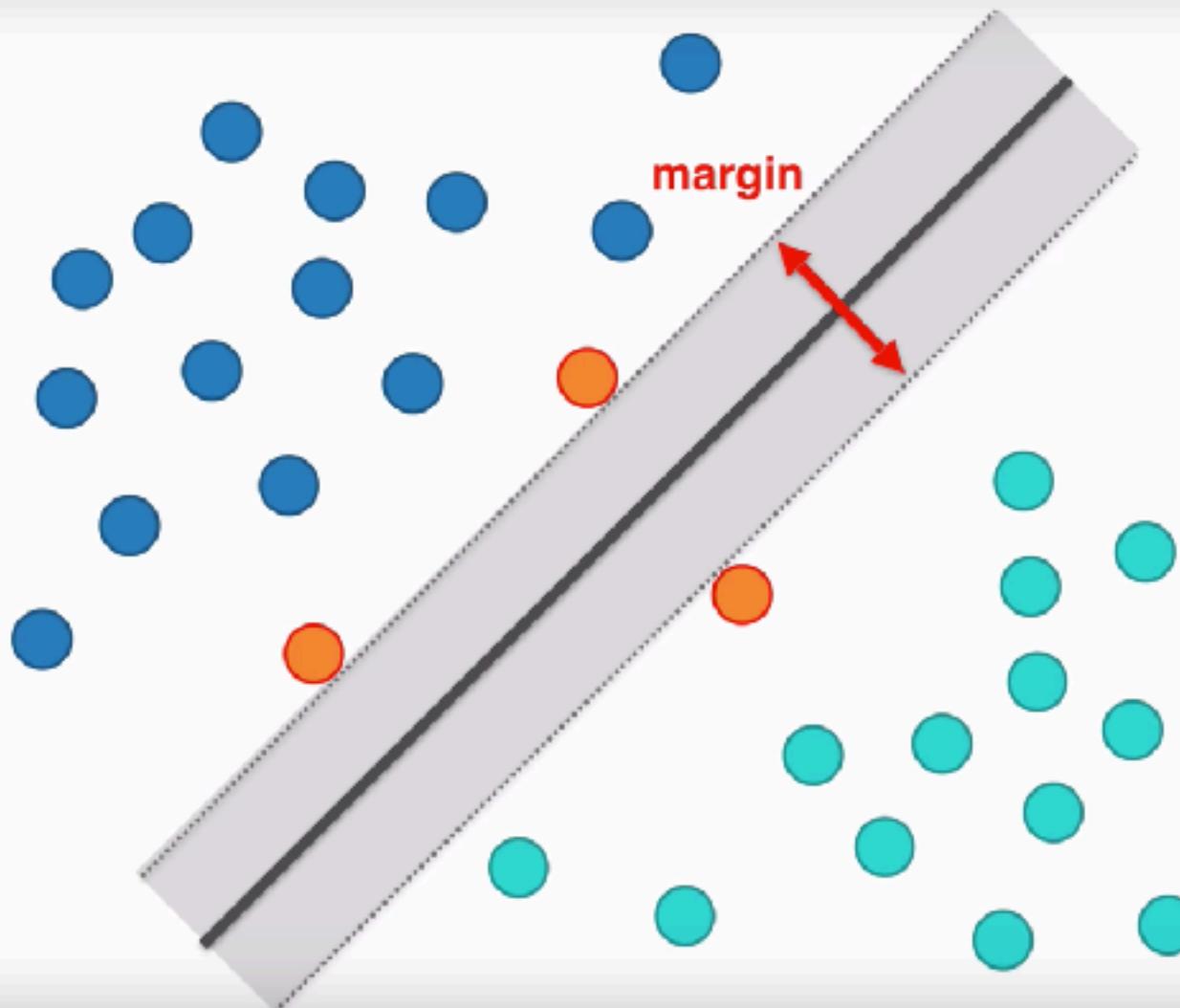
Why is this the best split?

The distance between the points and the line are as far as possible



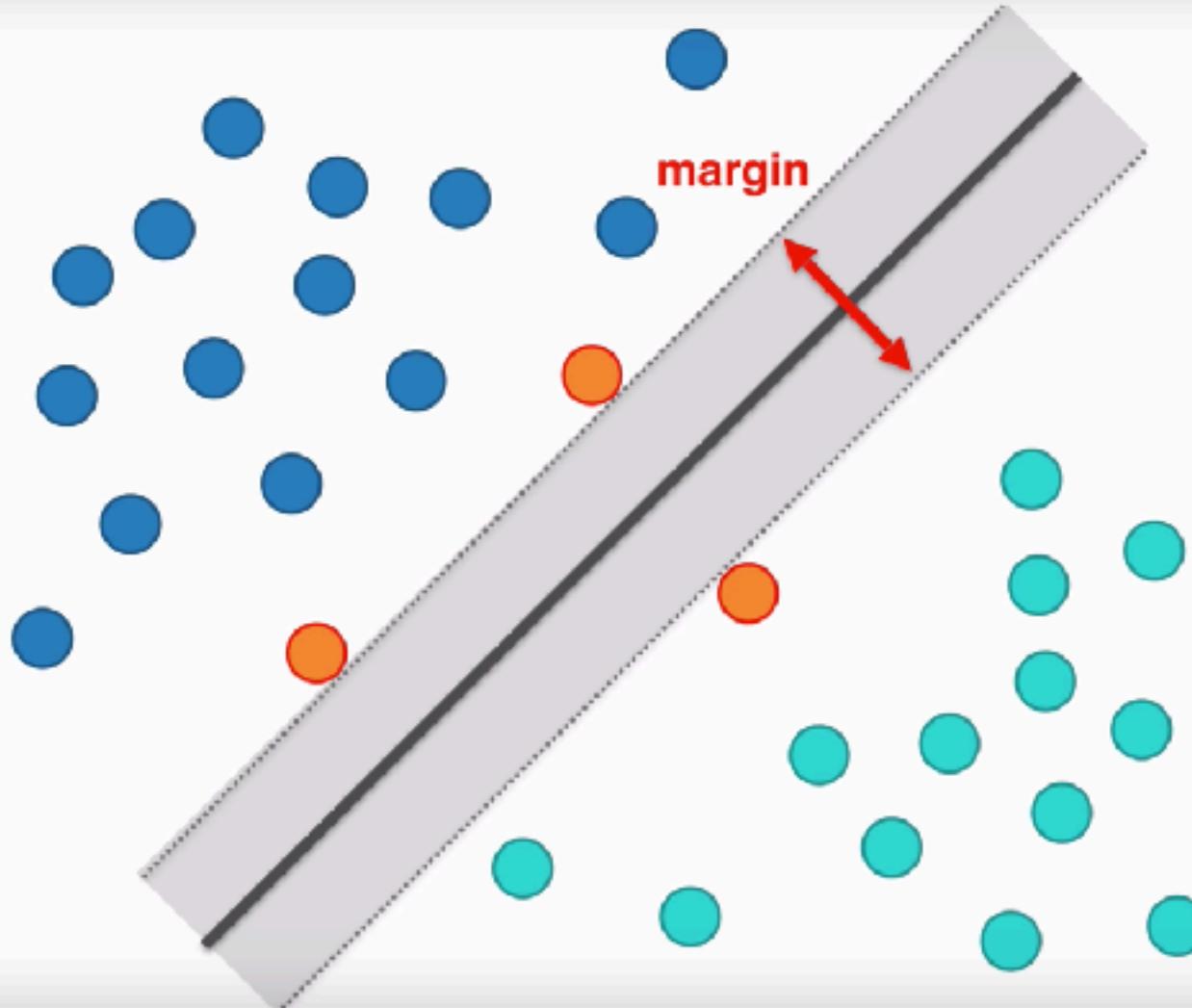
Why is this the best split?

The distance between the points and the line are as far as possible



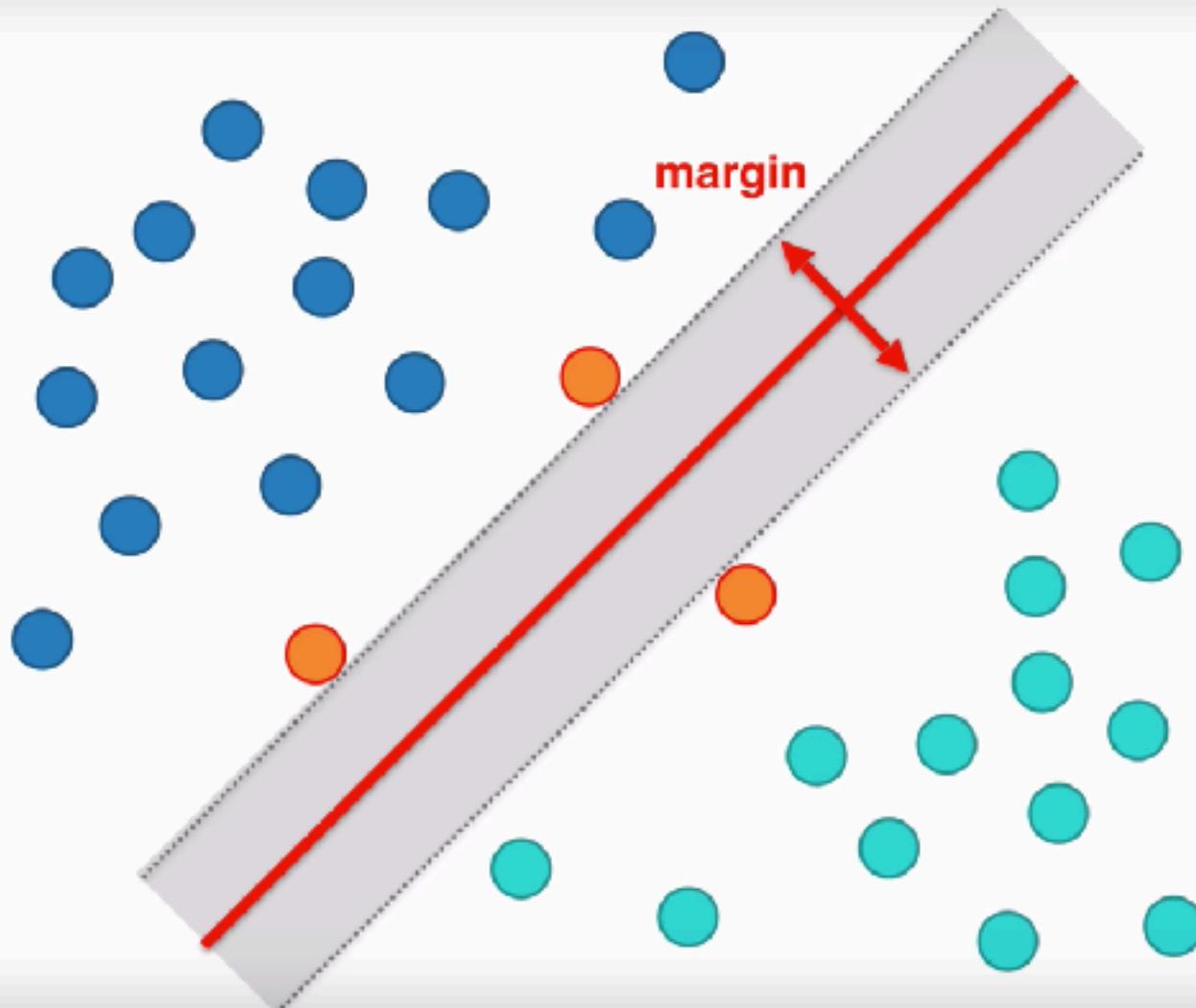
Why is this the best split?

The distance between the support vectors and the line are as far as possible



Why is this the best split?

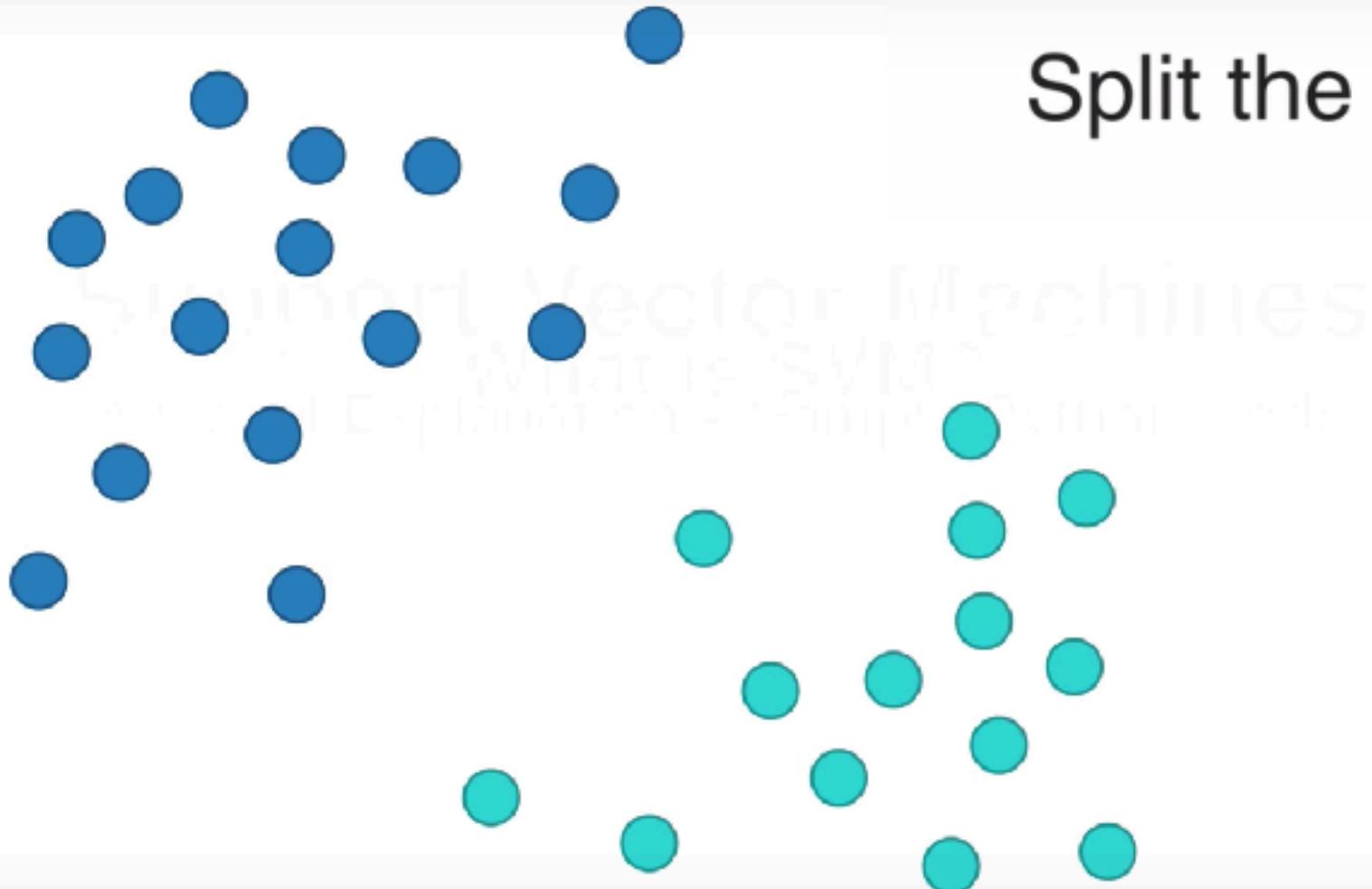
The distance between the support vectors and the line are as far as possible

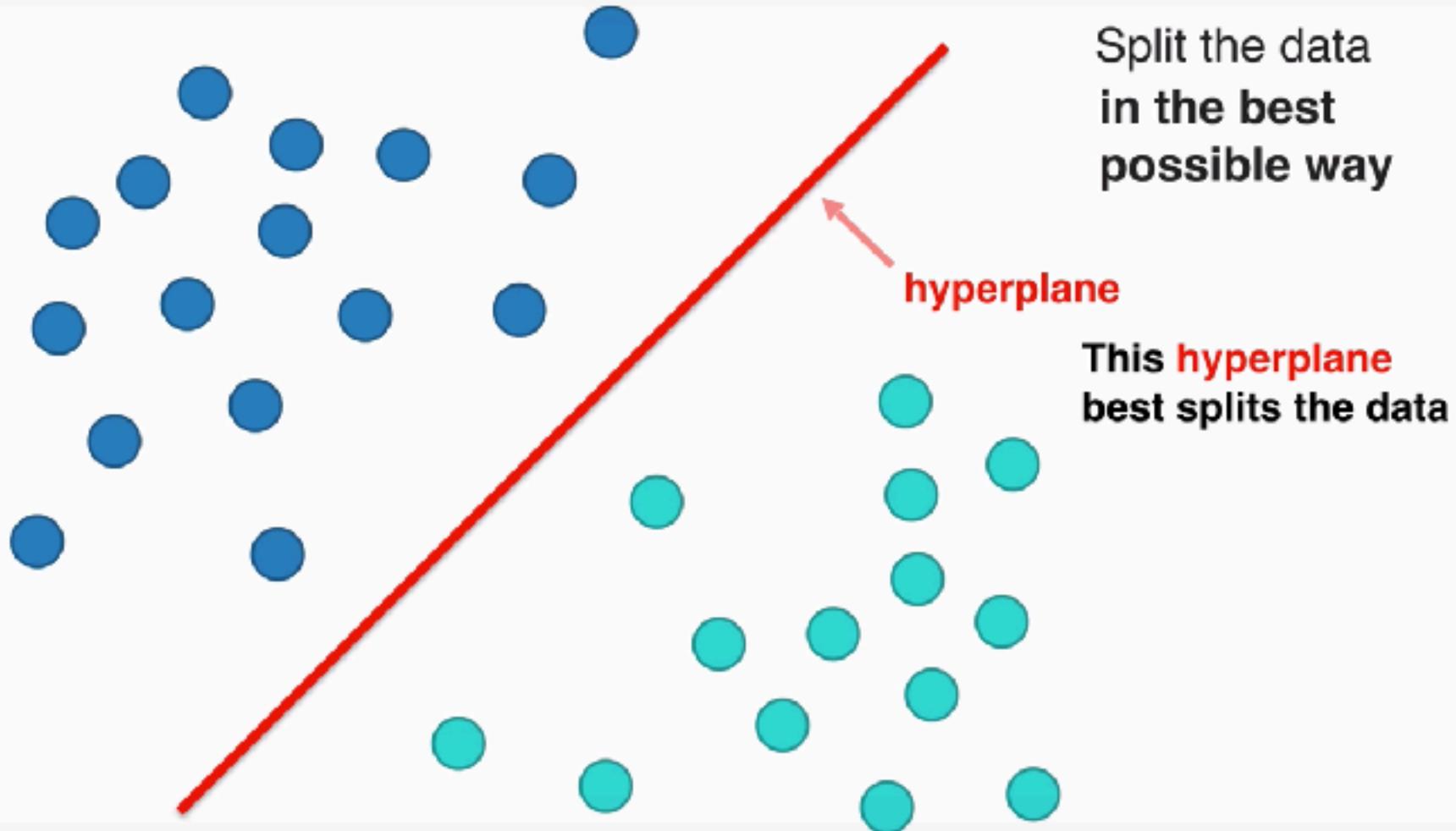


Why is this the best split?

The distance between the support vectors and the hyperplane are as far as possible

Split the data

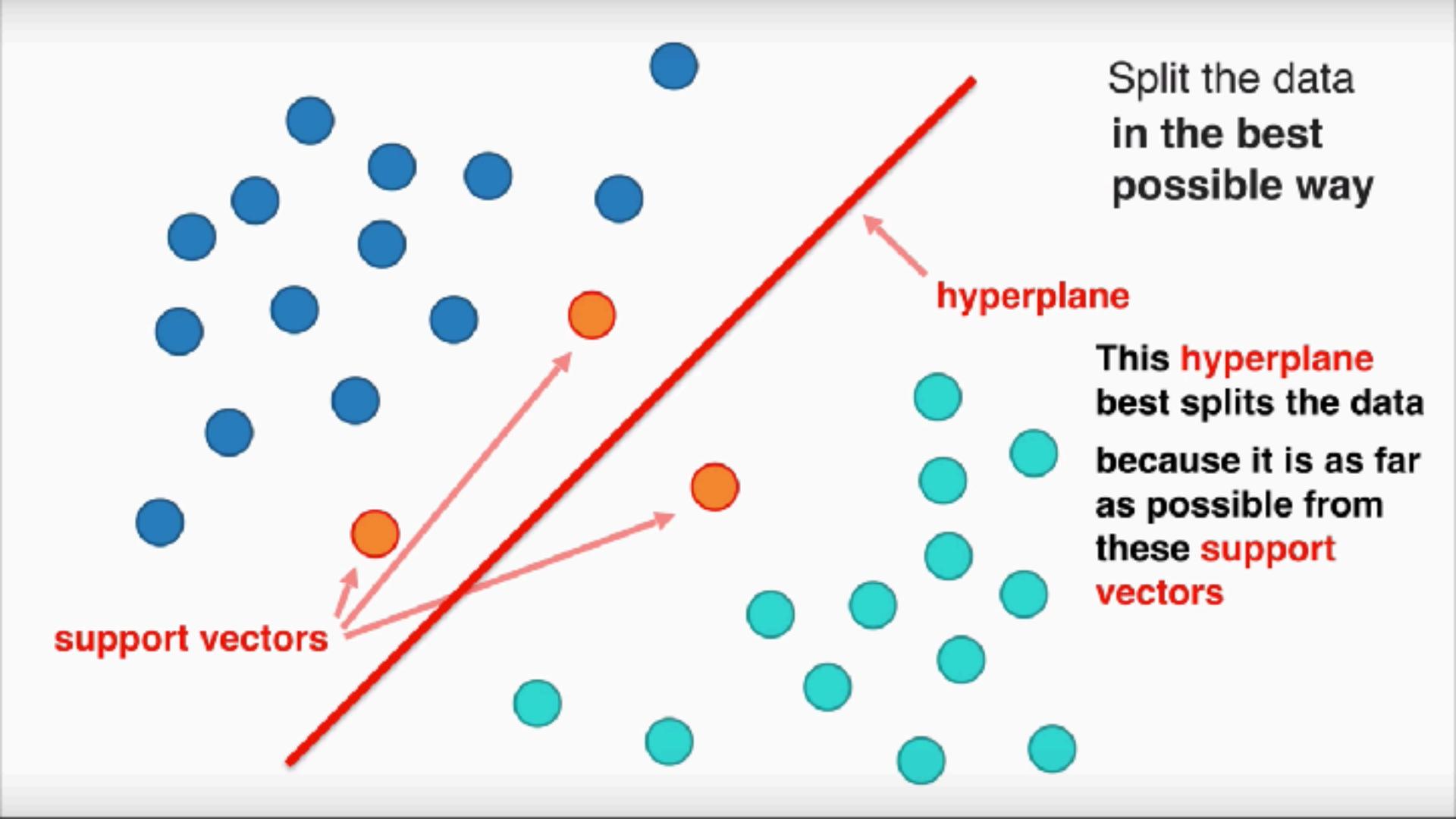




Split the data
in the best
possible way

hyperplane

This **hyperplane**
best splits the data

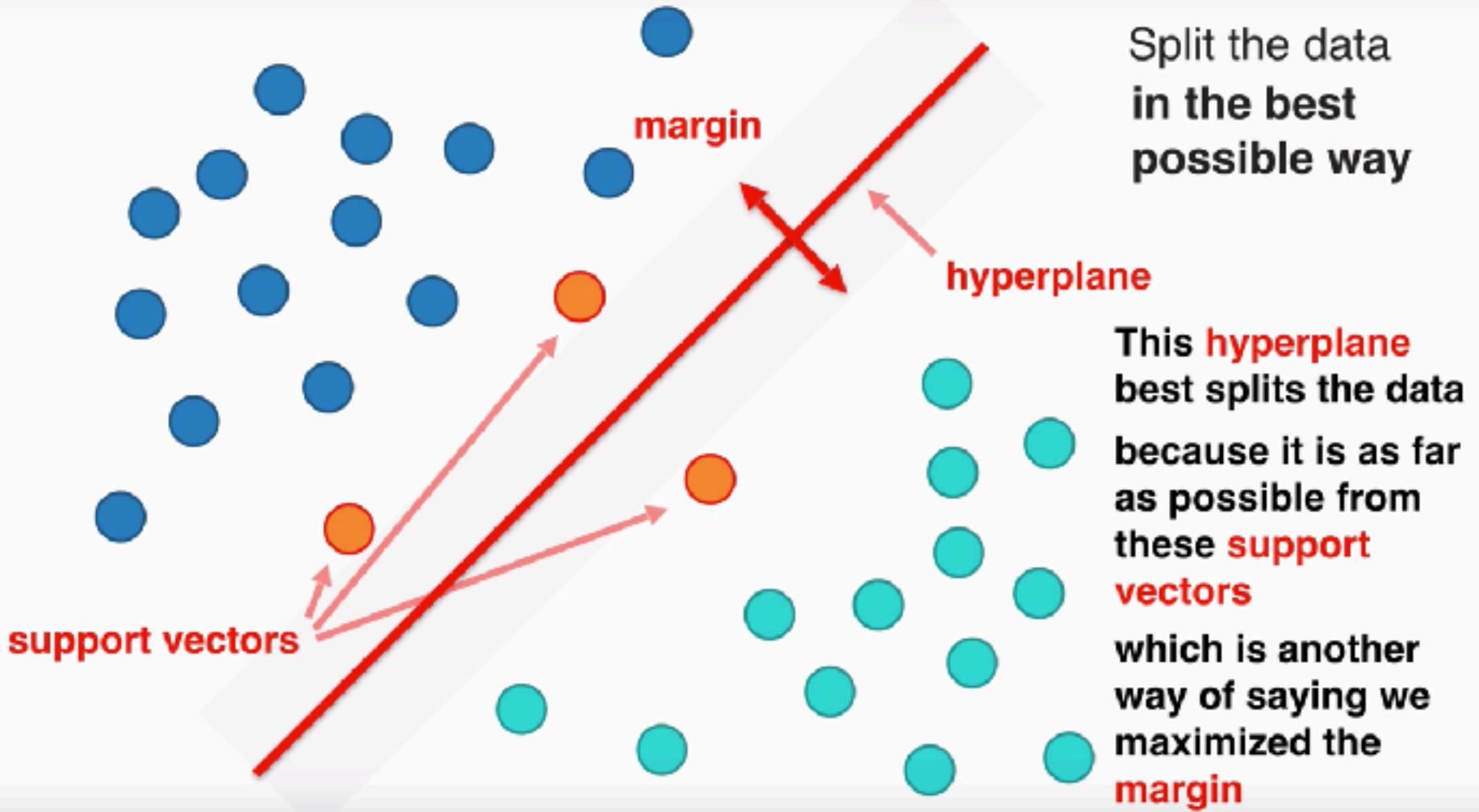


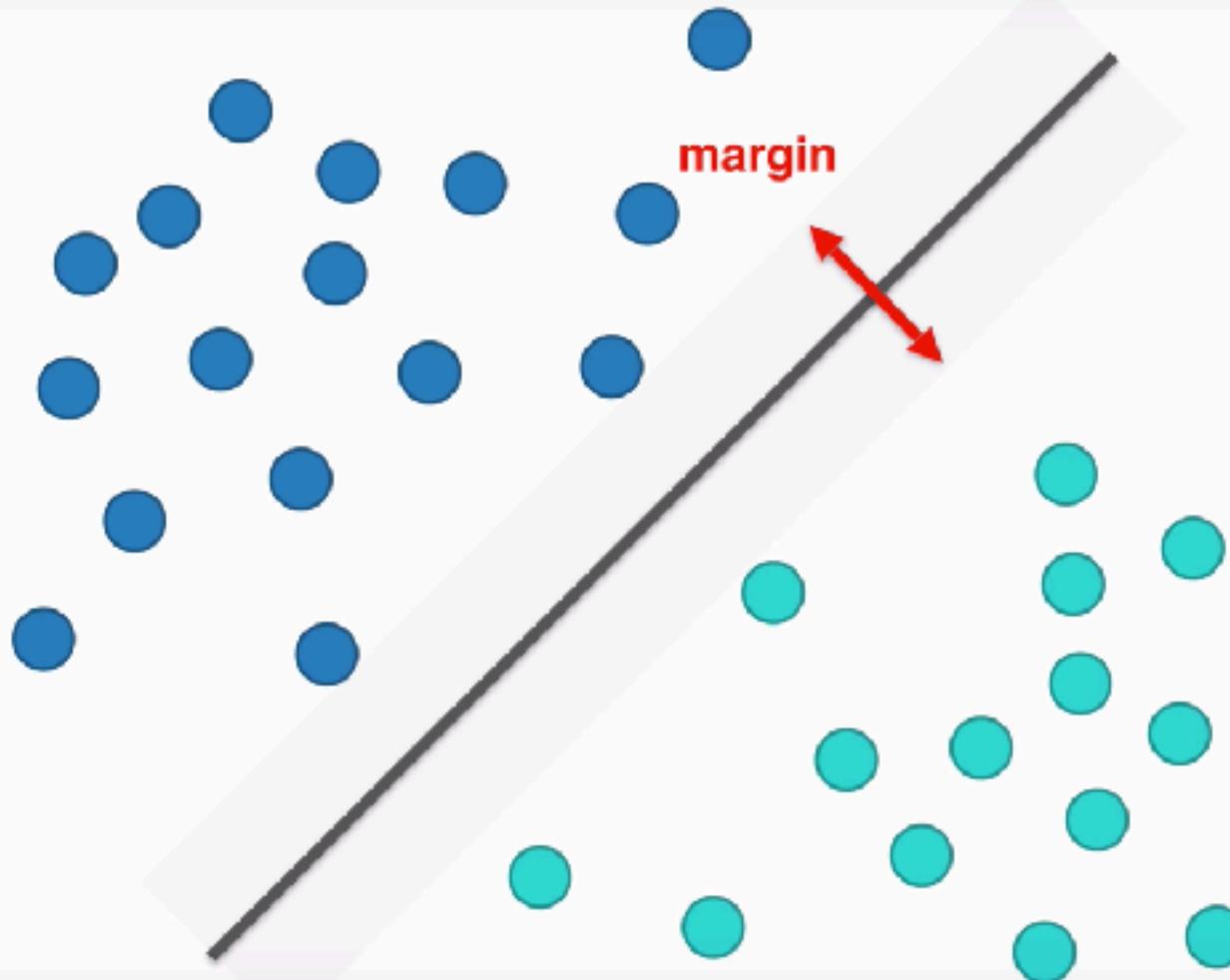
Split the data
in the best
possible way

hyperplane

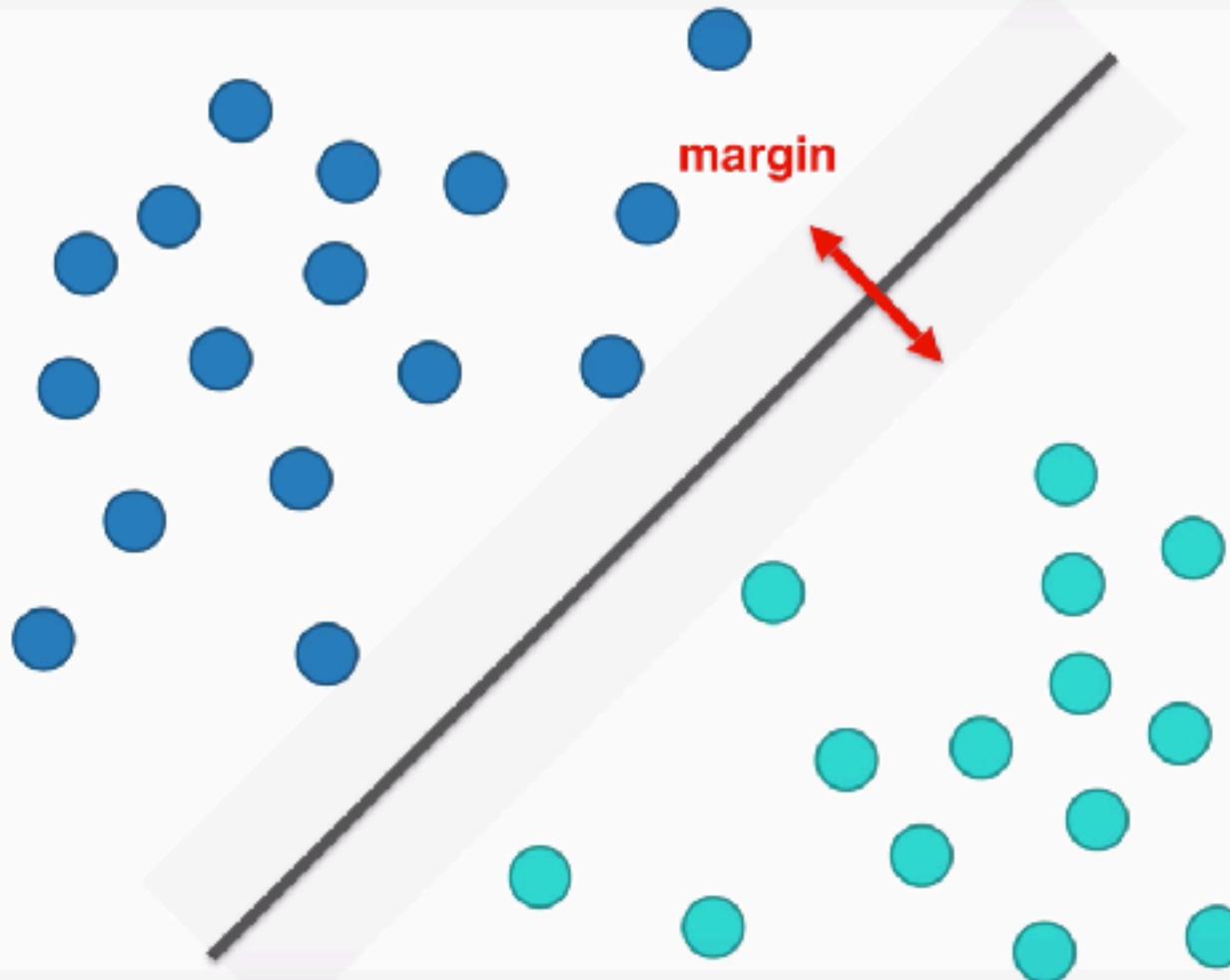
This **hyperplane**
best splits the data
because it is as far
as possible from
these **support
vectors**

support vectors



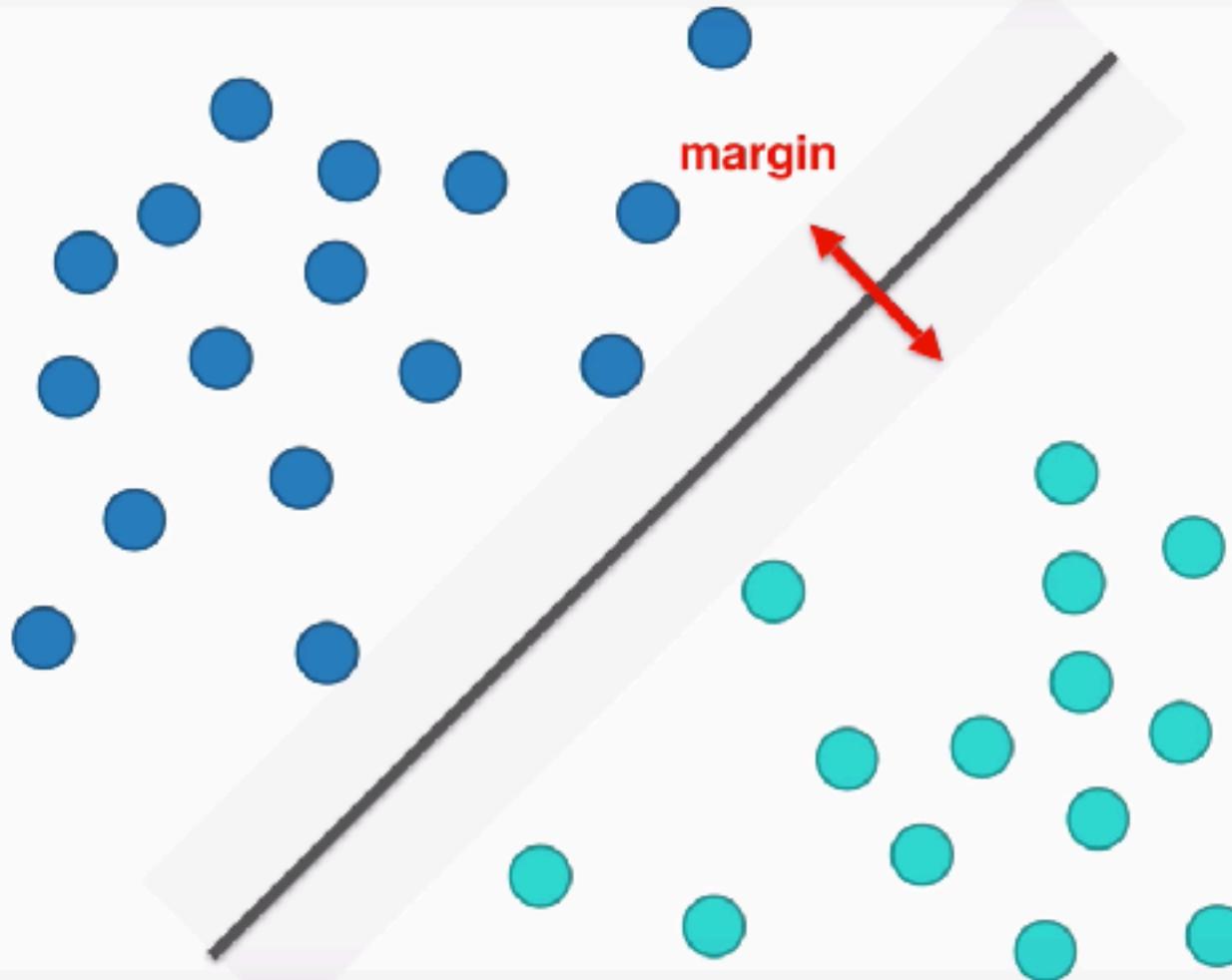


How do you
maximize the
margin?



How do you
maximize the
margin?

This is a
constrained
optimization
problem



How do you
maximize the
margin?

This is a
constrained
optimization
problem

which can be
solved using
the **Lagrange**
Multipliers
technique

So now what?

**Let's apply this to a real world
problem.**

Cupcakes



Muffins



versus

“a cupcake is just a muffin with frosting”

“a muffin is just a cupcake with random bits of stuff in it”

Cupcakes vs Muffins

The Challenge

Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

The Steps

1. Find the data
2. Apply a data science model
3. Review the results

Cupcakes vs Muffins

The Challenge

Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

The Steps

1. Find the data
2. Apply a data science model
3. Review the results

1. Find the data

The image shows a screenshot of a Google search interface. On the left, the Google logo is visible. To its right is a search bar containing the text "basic muffin recipe". Below the search bar, there is a list of recent searches. The first item in the list is "basic muffin recipe", followed by two other items: "basic muffin recipe" and "basic cupcake recipe". To the right of these search terms are two "Remove" links. At the top right of the search bar area are icons for microphone and search.

basic muffin recipe		
basic muffin recipe	Remove	Remove
basic cupcake recipe		

1. Find the data

Google basic muffin recipe  

basic muffin recipe Remove
basic cupcake recipe Remove

Recorded the top 10 muffin and top 10 cupcake recipes

1. Find the data

Problem

Each recipe yields different amounts of batter

Blended Recipe 10 oz. Donut Recipe 10 oz. each

Blended Recipe	10 oz.	Donut Recipe	10 oz.	each
----------------	--------	--------------	--------	------

1. Find the data

Problem

Each recipe yields different amounts of batter

Solution

Amount-based

Recipe	Flour	Sugar	Other
Muffin1	2 cups	1/2 cup	...
Cupcake1	2 cups	3/4 cup	...



Percent-based

Recipe	Flour	Sugar	Other	Total Volume
Muffin1	47%	24%	...	100%
Cupcake1	42%	21%	...	100%

1. Find the data

Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
Muffin	55	28	3	7	5	2	0	0
Muffin	47	24	12	6	9	1	0	0
Muffin	47	23	18	6	4	1	0	0
Muffin	50	25	12	6	5	2	1	0
Muffin	55	27	3	7	5	2	1	0
Muffin	54	27	7	5	5	2	0	0
Muffin	47	26	10	10	4	1	0	0
Muffin	50	17	17	8	6	1	0	0
Muffin	50	17	17	11	4	1	0	0
Cupcake	39	0	26	19	14	1	1	0
Cupcake	34	17	20	20	5	2	1	0
Cupcake	39	13	17	19	10	1	1	0
Cupcake	38	15	23	15	8	0	1	0
Cupcake	42	18	25	9	5	1	0	0
Cupcake	36	14	21	14	11	2	1	0
Cupcake	38	15	31	8	6	1	1	0
Cupcake	36	16	24	12	9	1	1	0
Cupcake	34	17	23	11	13	0	1	0

Cupcakes vs Muffins

The Challenge

Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

The Steps

1. Find the data ✓
2. Apply a data science model
3. Review the results

Cupcakes vs Muffins

The Challenge

Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

DATA

The Steps

1. Find the data ✓
2. Apply a data science model
3. Review the results

Python Script

```
Jupyter Notebook - Python3 (Python 3.6.5) - Colab
```

Cancer vs Survival with SVM

Step 1 Import Data

```
# Step 1: Import the required libraries for this algorithm.
```

Step 2: Load the cancer dataset from the scikit-learn library.

```
# Importing the required libraries
```

Step 3: Import the dataset.

```
# Importing the dataset
```

Step 4: Check the first few rows of the dataset.

```
# Check first few rows of the dataset
```

Sample Number	Clump Thickness	Uniform Cell Size	Uniform Cell Shape	Marginal Note	Single Epithelial Cell Size	Bare Nuclei	Normal Nucleoli	Mitoses
0	2	3	3	3	2	1	1	0
1	4	4	4	4	3	2	1	0
2	3	3	3	3	2	1	1	0
3	2	2	2	2	1	1	1	0
4	3	3	3	3	2	1	1	0

Step 5: Visualize the data.

```
# Step 5: Visualize the data
```

Step 6: Fit the model.

```
# Step 6: Fit the model
```

Step 7: Predict the output.

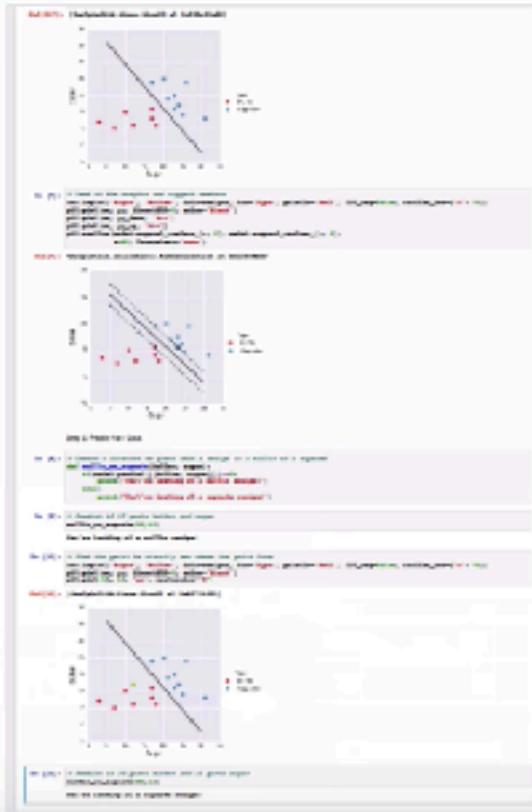
```
# Step 7: Predict the output
```

Step 8: Check the accuracy.

```
# Step 8: Check the accuracy
```

Step 9: Visualize the results.

```
# Step 9: Visualize the results
```



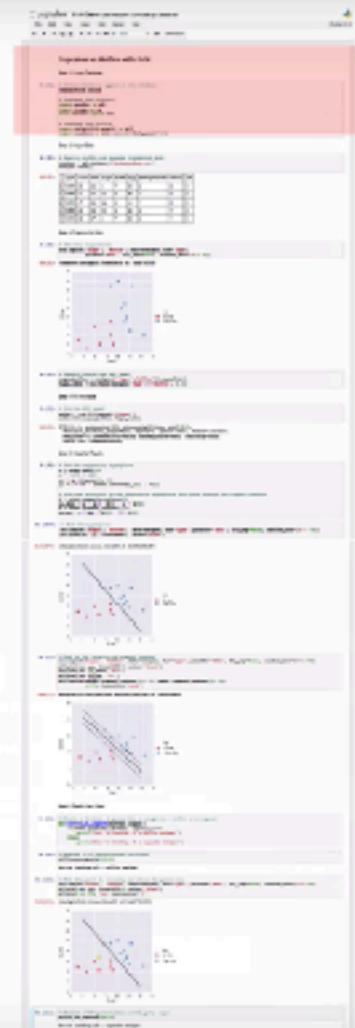
- Import Libraries
- Import Data
- Prepare the Data
- Fit the Model
- Visualize Results
- Predict New Case

a. Import Libraries

Cupcakes vs Muffins with SVM

Step 1: Import Libraries

```
In [13]: # Allows charts to appear in the notebook  
%matplotlib inline  
  
# Libraries for analysis  
import pandas as pd  
import numpy as np  
from sklearn import svm  
  
# Libraries for visuals  
import matplotlib.pyplot as plt  
import seaborn as sns; sns.set(font_scale=1.2)
```



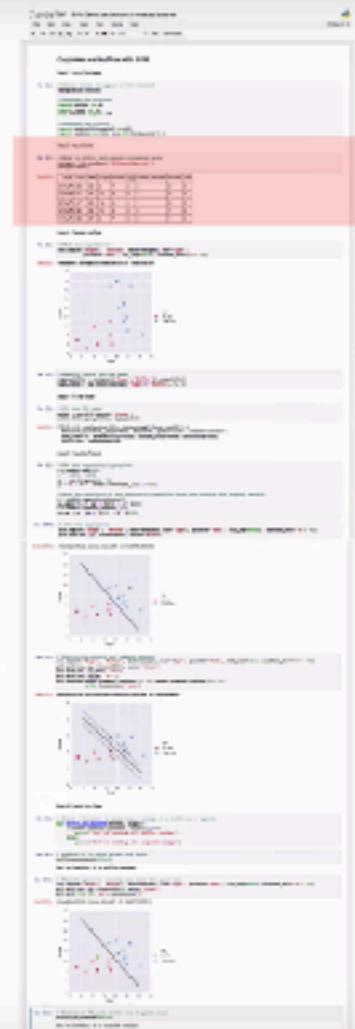
b. Import Data

Step 2: Import Data

```
In [2]: # Read in muffin and cupcake ingredient data  
recipes = pd.read_csv('data_recipes.csv')  
recipes.head()
```

Out [2]:

	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
0	Muffin	55	28	3	7	5	2	0	0
1	Muffin	47	24	12	6	9	1	0	0
2	Muffin	47	23	18	6	4	1	0	0
3	Muffin	50	25	12	6	5	2	1	0
4	Muffin	55	27	3	7	5	2	1	0

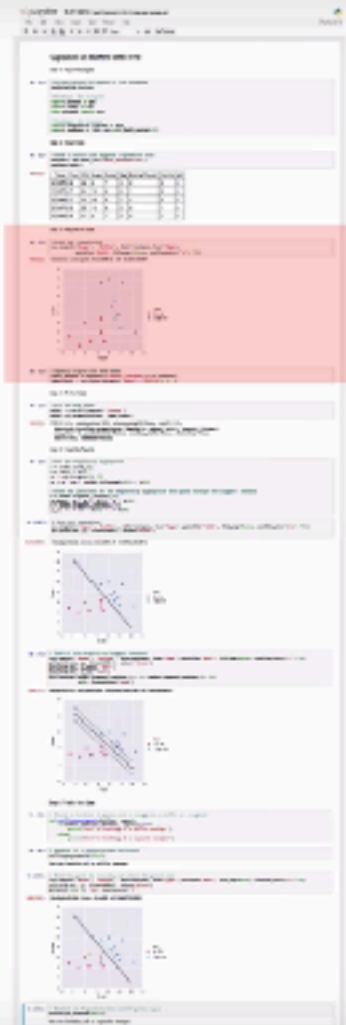
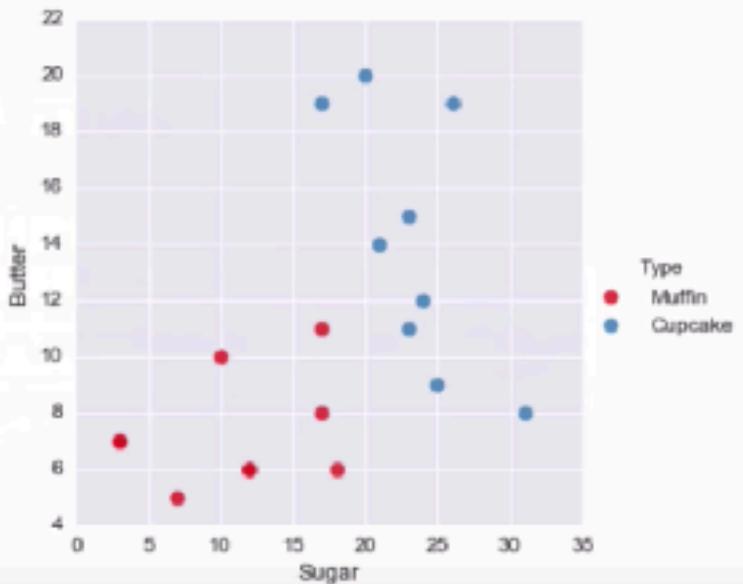


c. Prepare the Data

Step 3: Prepare the Data

```
In [3]: # Plot two ingredients  
sns.lmplot('Sugar', 'Butter', data=recipes, hue='Type',  
           palette='Set1', fit_reg=False, scatter_kws={"s": 70})
```

```
Out[3]: <seaborn.axisgrid.FacetGrid at 0xad7af28>
```



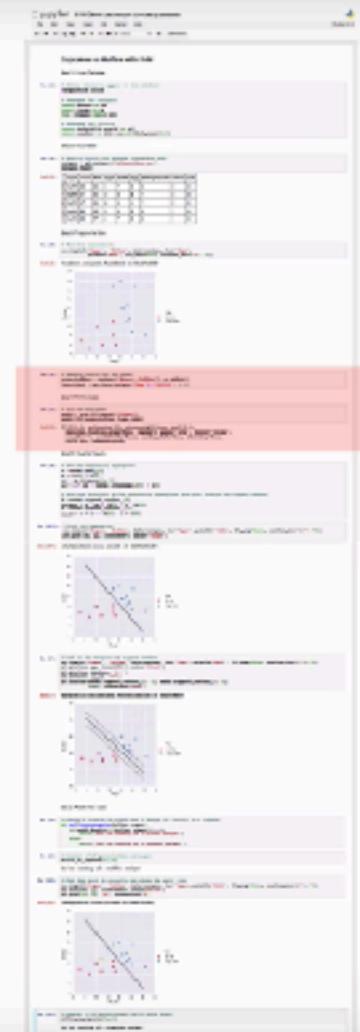
d. Fit the Model

Step 4: Fit the Model

```
In [4]: # Specify inputs for the model  
sugar_butter = recipes[['Sugar', 'Butter']].as_matrix()  
type_label = np.where(recipes['Type']=='Muffin', 0, 1)
```

```
In [5]: # Fit the SVM model  
model = svm.SVC(kernel='linear')  
model.fit(sugar_butter, type_label)
```

```
Out[5]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
           decision_function_shape=None, degree=3, gamma='auto', kernel='linear',  
           max_iter=-1, probability=False, random_state=None, shrinking=True,  
           tol=0.001, verbose=False)
```



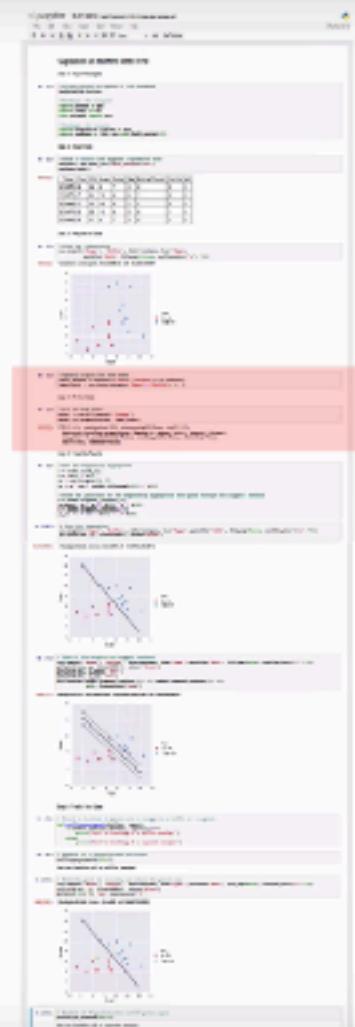
d. Fit the Model

Step 4: Fit the Model

```
In [4]: # Specify inputs for the model  
sugar_butter = recipes[['Sugar','Butter']].as_matrix()  
type_label = np.where(recipes['Type']=='Muffin', 0, 1)
```

```
In [5]: # Fit the SVM model  
model = svm.SVC(kernel='linear')  
model.fit(sugar_butter, type_label)
```

```
Out[5]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
           decision_function_shape=None, degree=3, gamma='auto', kernel='linear',  
           max_iter=-1, probability=False, random_state=None, shrinking=True,  
           tol=0.001, verbose=False)
```

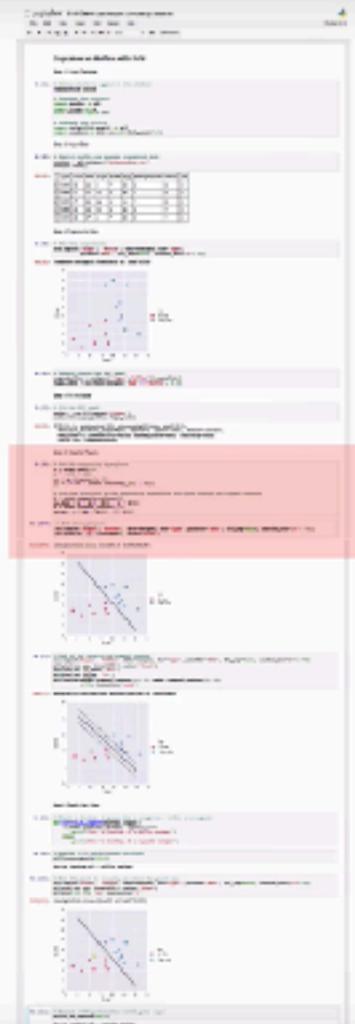


e. Visualize Results

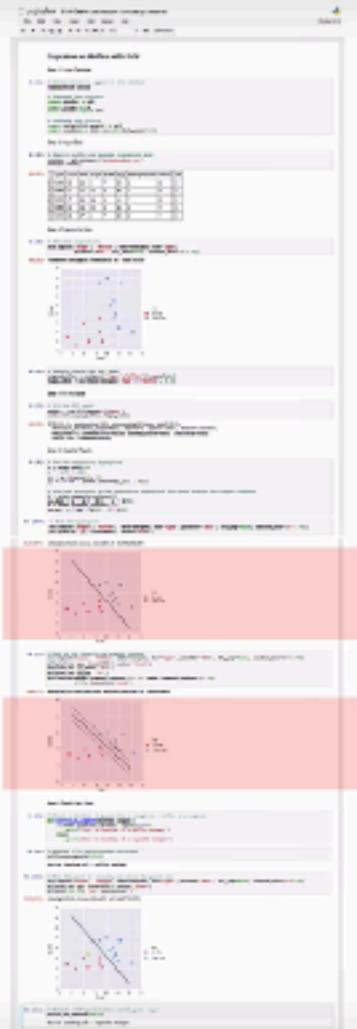
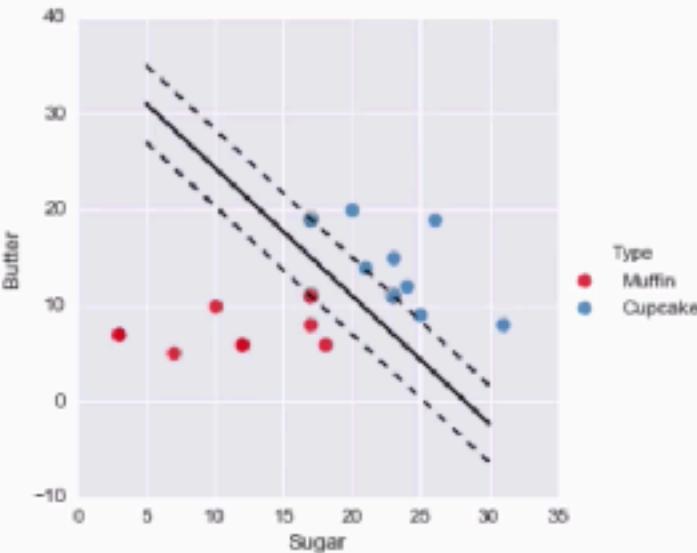
Step 5: Visualize Results

```
In [6]: # Get the separating hyperplane
w = model.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(5, 30)
yy = a * xx - (model.intercept_[0]) / w[1]

# Plot the parallels to the separating hyperplane
# that pass through the support vectors
b = model.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = model.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])
```



e. Visualize Results



Cupcakes vs Muffins

The Challenge

Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

The Steps

1. Find the data ✓
2. Apply a data science model ✓
3. Review the results

3. Review the Results

The Challenge

Classify recipes as cupcakes or muffins. ✓

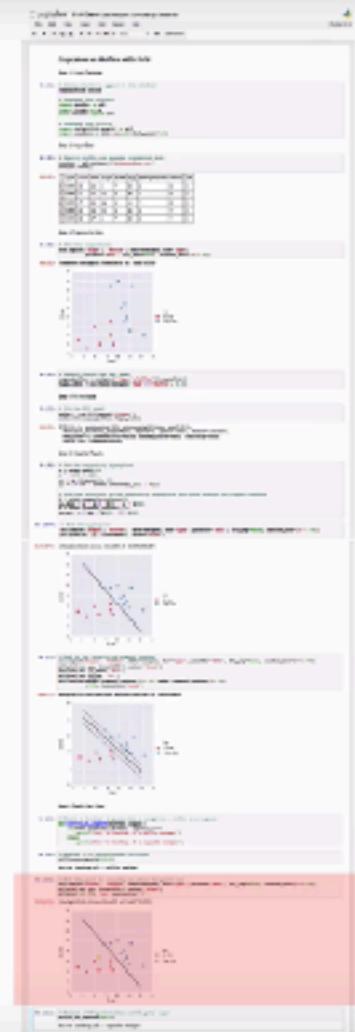
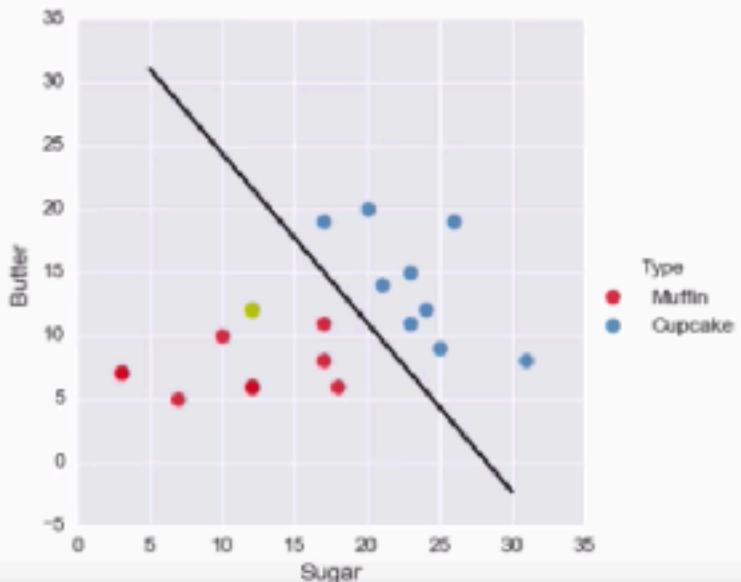
When given a new recipe, determine if it's a cupcake or a muffin.



f. Predict New Case

```
In [10]: # Plot the point to visually see where the point lies
sns.lmplot('Sugar', 'Butter', data=recipes, hue='Type',
            palette='Set1', fit_reg=False, scatter_kws={"s": 70})
plt.plot(xx, yy, linewidth=2, color='black')
plt.plot(12, 12, 'yo', markersize='9')
```

Out[10]: [`<matplotlib.lines.Line2D at 0xb075160>`]



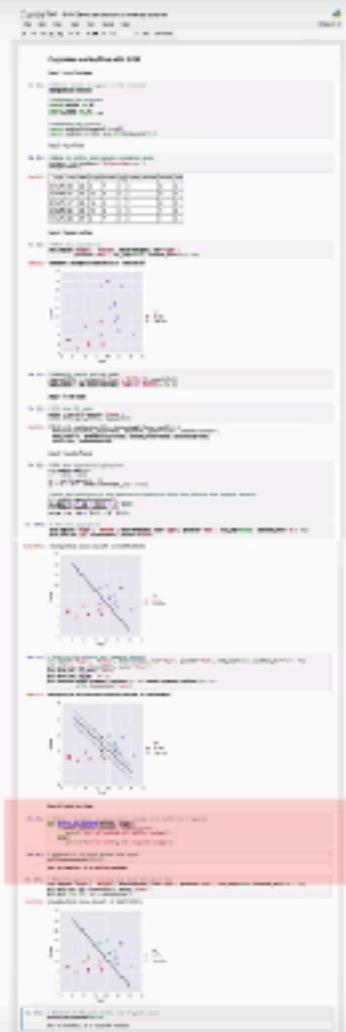
f. Predict New Case

Step 6: Predict New Case

```
In [8]: # Create a function to guess when a recipe is a muffin
# or a cupcake using the SVM model we created
def muffin_or_cupcake(butter, sugar):
    if(model.predict([[butter, sugar]]) == 0):
        print('You\'re looking at a muffin recipe!')
    else:
        print('You\'re looking at a cupcake recipe!')
```

```
In [9]: # Predict if 12 parts butter and sugar
muffin_or_cupcake(12,12)
```

You're looking at a muffin recipe!



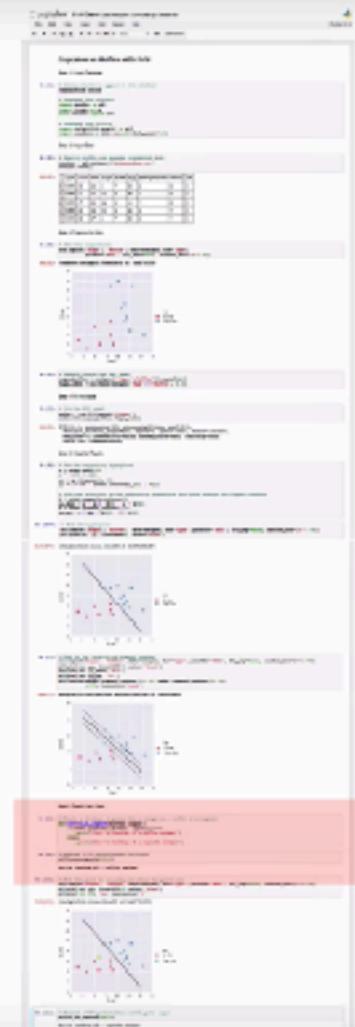
f. Predict New Case

Step 6: Predict New Case

```
In [8]: # Create a function to guess when a recipe is a muffin  
# or a cupcake using the SVM model we created  
def muffin_or_cupcake(butter, sugar):  
    if(model.predict([[butter, sugar]]) == 0):  
        print('You\'re looking at a muffin recipe!')  
    else:  
        print('You\'re looking at a cupcake recipe!')
```

```
In [9]: # Predict if 12 parts butter and sugar  
muffin_or_cupcake(12, 12)
```

You're looking at a muffin recipe!



Cupcakes



Muffins



versus

Cupcakes and muffins can be classified using Support Vector Machines!

Basic case ✓

**Up next: How to make SVM
even more powerful**

Agenda

1. SVM Basics

- Visual introduction
- Example in Python

2. Additional Complexities

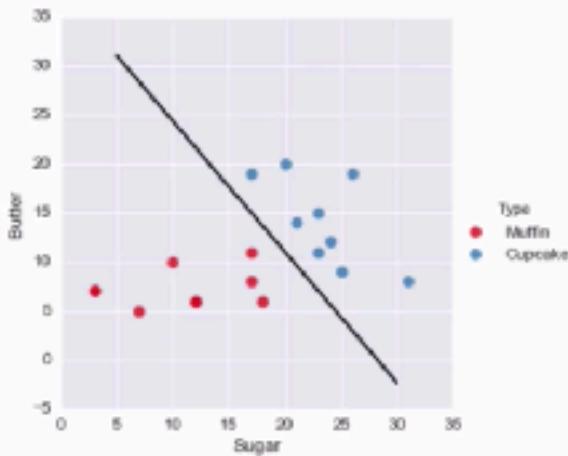
- Higher dimensions
- Multiple classes
- C parameter
- Kernel trick

3. Closing Remarks

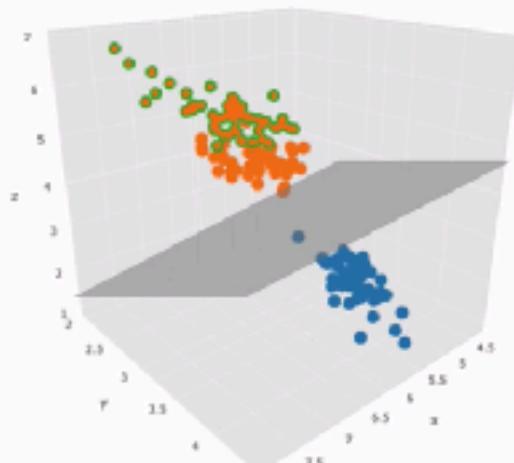
- Pros and cons
- Other techniques

Higher Dimensions: Visual

2D: Separate
with Line



3D: Separate
with Plane



Higher Dimensions

4D+: Separate
with Hyperplane

Hard to
Visualize

C Parameter

Multiple Classes

Kernel Trick

Higher Dimensions: Code

- Hard to visualize, but easy to code

```
# Fit SVM model for sugar & butter
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

```
[ 3,  7],
[12,  6],
[18,  6],
[12,  6],
[ 3,  7], ...
```

Higher Dimensions: Code

- Hard to visualize, but easy to code

```
# Fit SVM model for sugar & butter
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

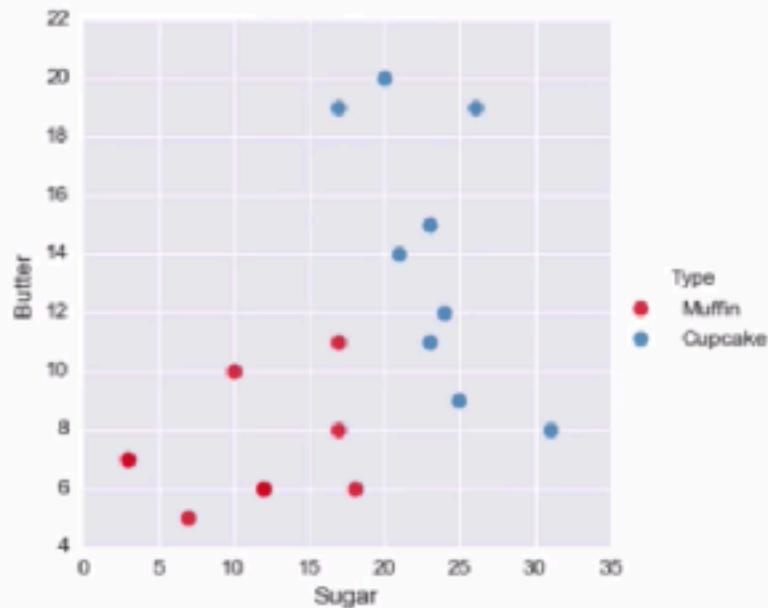
```
[ 3,  7],
[12,  6],
[18,  6],
[12,  6],
[ 3,  7], ...
```

all_ingredients

```
[55, 28,  3,  7,  5,  2,  0,  0],
[47, 24, 12,  6,  9,  1,  0,  0],
[47, 23, 18,  6,  4,  1,  0,  0],
[50, 25, 12,  6,  5,  2,  1,  0],
[55, 27,  3,  7,  5,  2,  1,  0], ...
```

C Parameter: Visual

Demo Data



Higher Dimensions

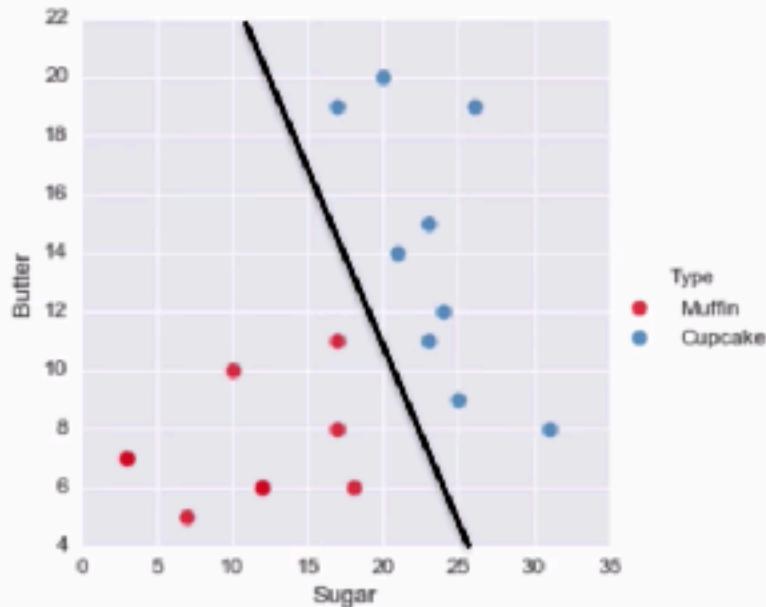
C Parameter

Multiple Classes

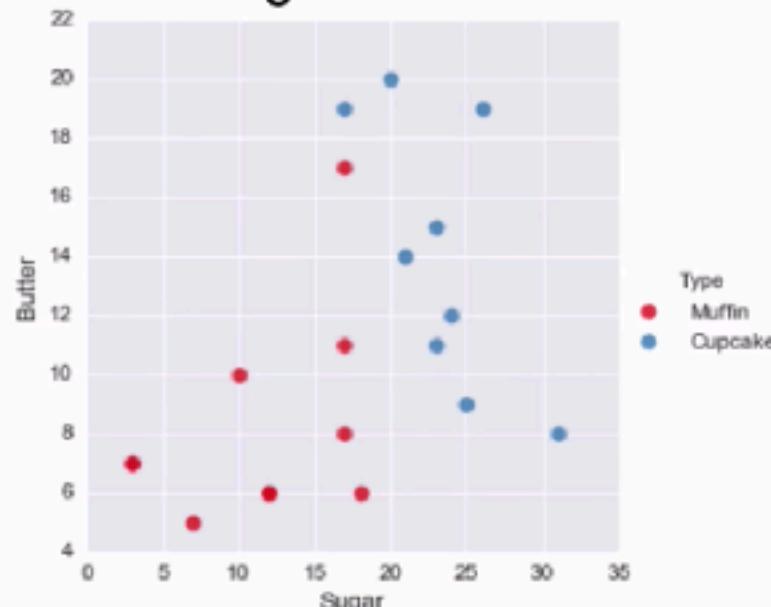
Kernel Trick

C Parameter: Visual

Demo Data



Original Data



Higher Dimensions

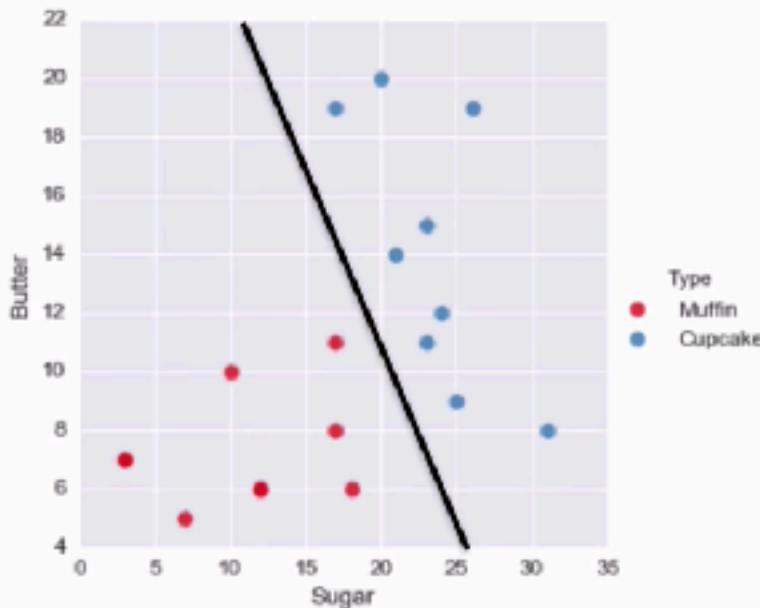
C Parameter

Multiple Classes

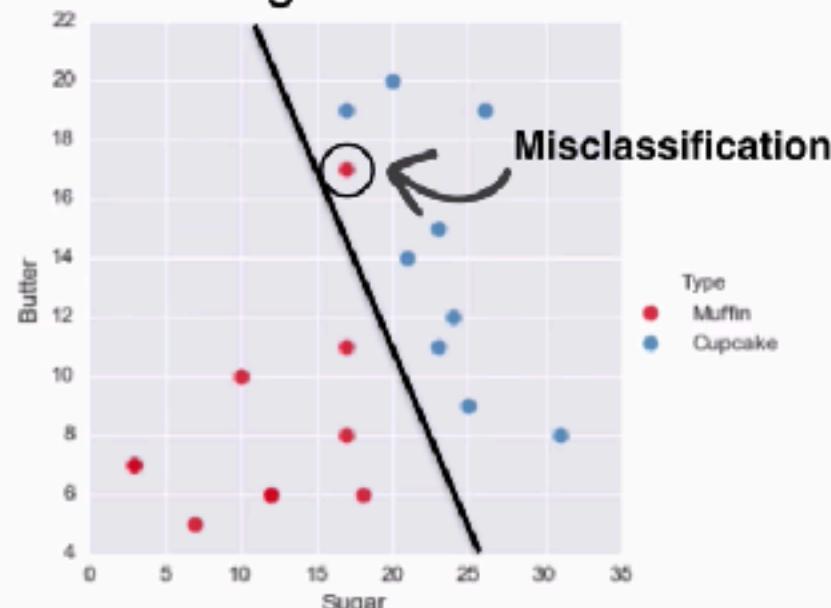
Kernel Trick

C Parameter: Visual

Demo Data



Original Data



Higher Dimensions

C Parameter

Multiple Classes

Kernel Trick

C Parameter: Code

The C parameter allows you to decide how much you want to penalize misclassified points

```
In [5]: # Fit the SVM model
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)

Out[5]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

Default Value

C Parameter: Comparison

```
# Fit the SVM model with a LOW C  
model = svm.SVC(kernel='linear', C=2**-5)  
model.fit(sugar_butter, type_label)
```

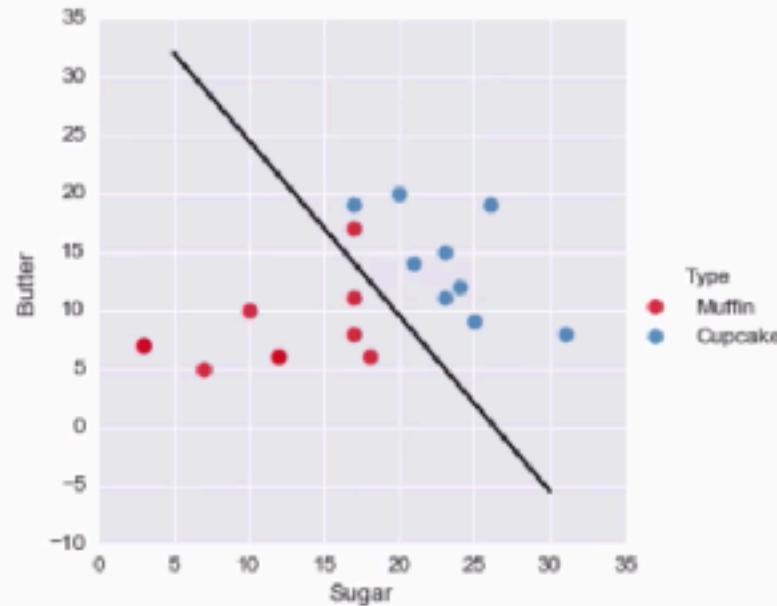
```
# Fit the SVM model with a HIGH C  
model = svm.SVC(kernel='linear', C=2**5)  
model.fit(sugar_butter, type_label)
```



C Parameter: Comparison

```
# Fit the SVM model with a LOW C
```

```
model = svm.SVC(kernel='linear', C=2**-5)  
model.fit(sugar_butter, type_label)
```

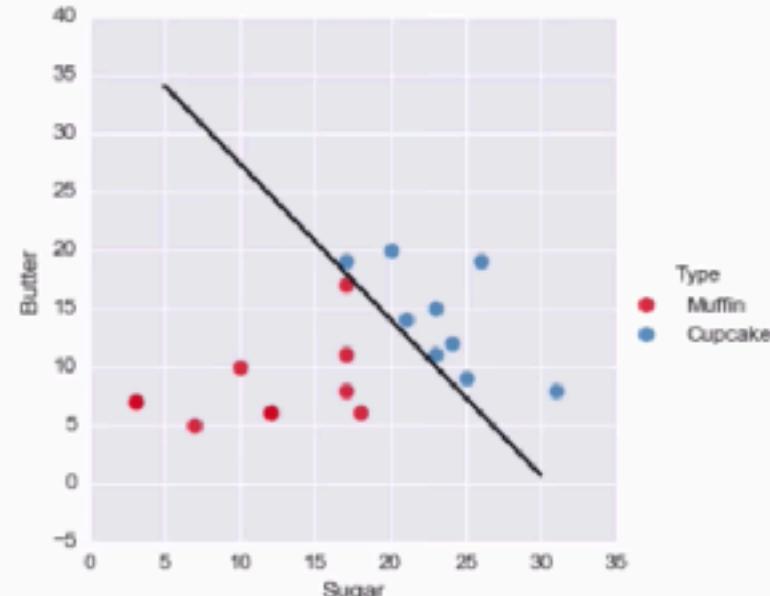


Higher Dimensions

C Parameter

```
# Fit the SVM model with a HIGH C
```

```
model = svm.SVC(kernel='linear', C=2**5)  
model.fit(sugar_butter, type_label)
```



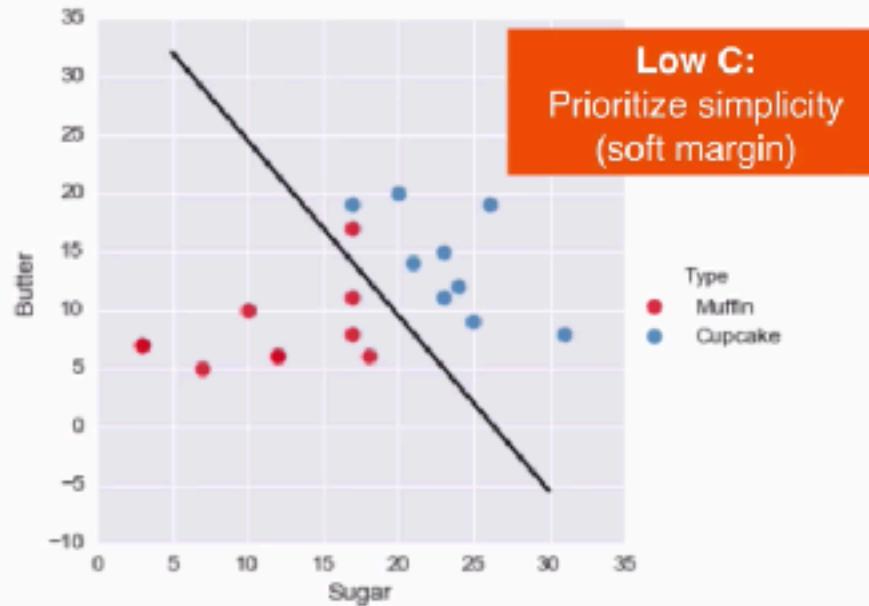
Multiple Classes

Kernel Trick

C Parameter: Comparison

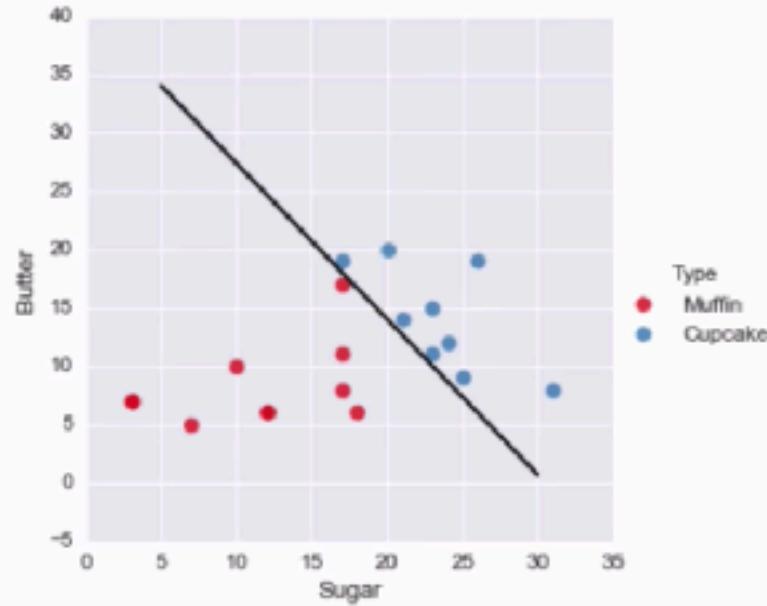
```
# Fit the SVM model with a LOW C
```

```
model = svm.SVC(kernel='linear', C=2**-5)  
model.fit(sugar_butter, type_label)
```



```
# Fit the SVM model with a HIGH C
```

```
model = svm.SVC(kernel='linear', C=2**5)  
model.fit(sugar_butter, type_label)
```



Higher Dimensions

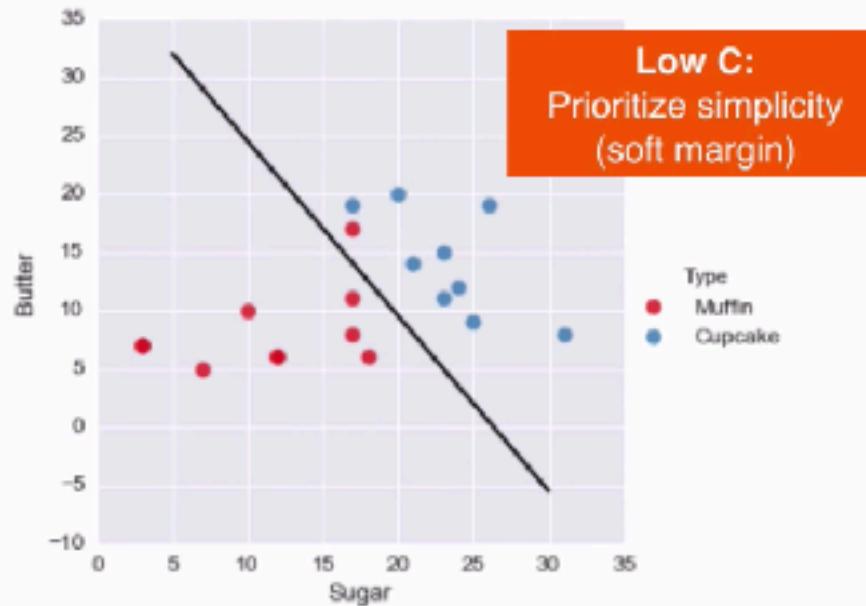
C Parameter

Multiple Classes

Kernel Trick

C Parameter: Comparison

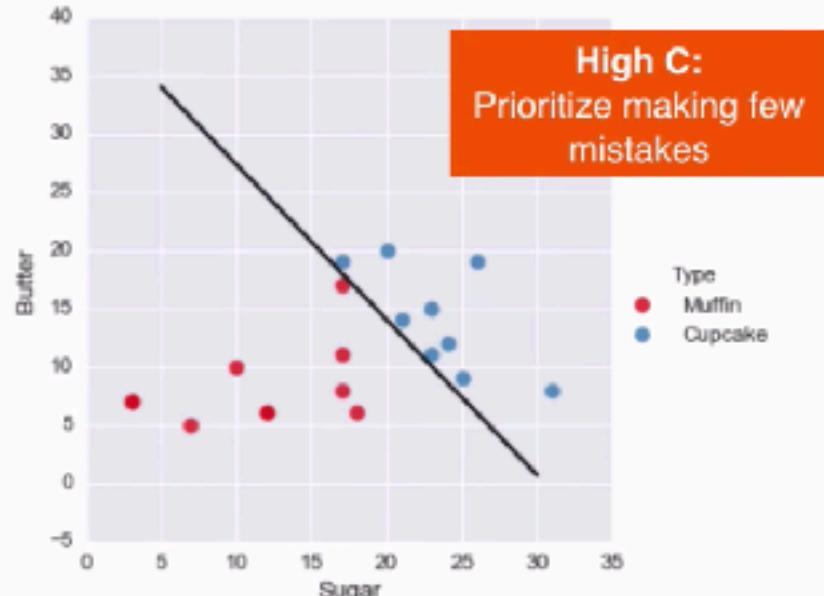
```
# Fit the SVM model with a LOW C  
model = svm.SVC(kernel='linear', C=2**-5)  
model.fit(sugar_butter, type_label)
```



Higher Dimensions

C Parameter

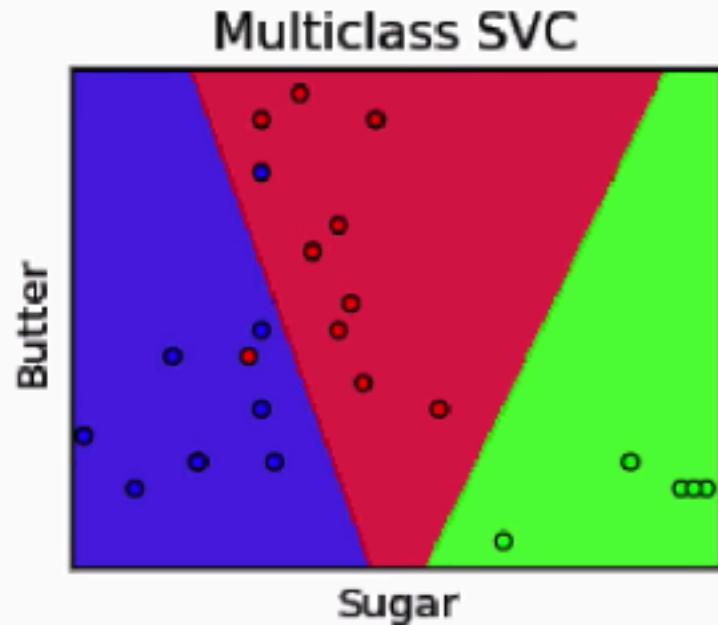
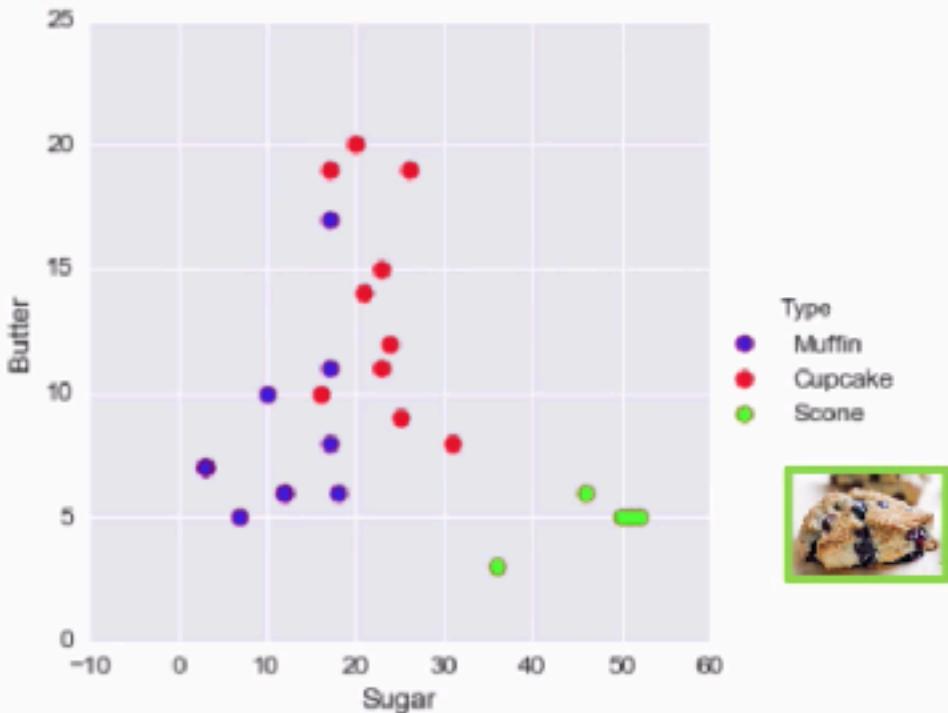
```
# Fit the SVM model with a HIGH C  
model = svm.SVC(kernel='linear', C=2**5)  
model.fit(sugar_butter, type_label)
```



Multiple Classes

Kernel Trick

Multiple Classes: Visual



Higher Dimensions

C Parameter

Multiple Classes

Kernel Trick

Multiple Classes: Code

Original Code
(2 classes)

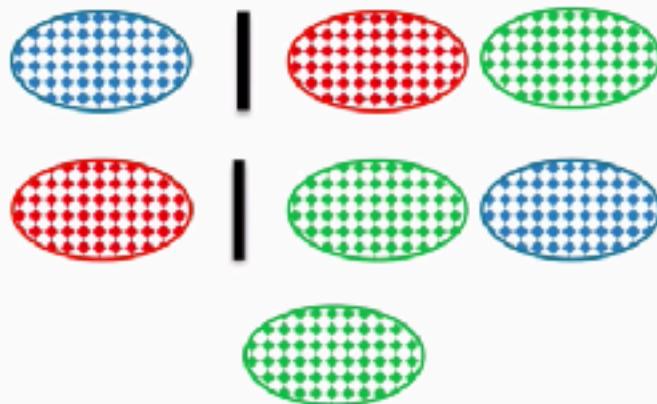
```
# Fit the SVM model
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

Updated Code
(3+ classes)

```
# Fit the SVM model for more than 2 classes
model = svm.SVC(kernel='linear', decision_function_shape='ovr')
model.fit(sugar_butter, type_label)
```

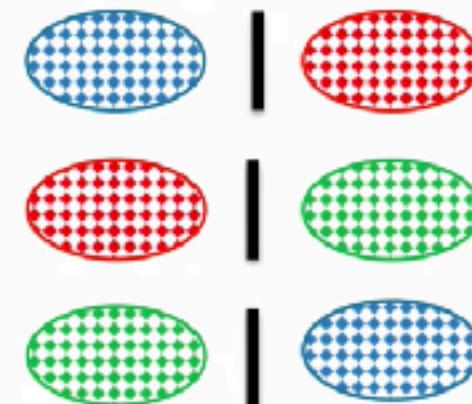
Multiple Classes: Comparison

OVR: One vs Rest



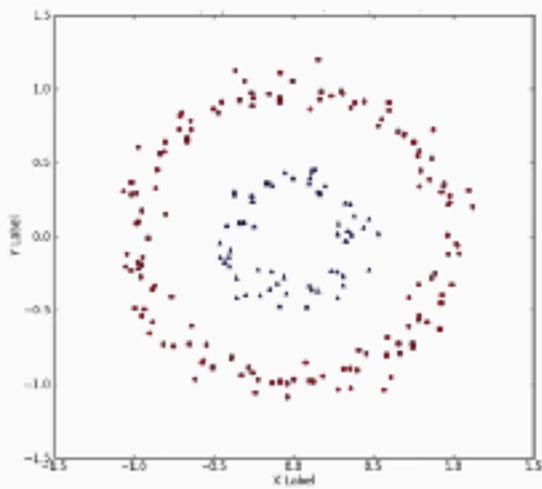
Pros: Fewer classifications
Cons: Classes may be imbalanced

OVO: One vs One



Pros: Less sensitive to imbalance
Cons: More classifications

Kernel Trick: Visual



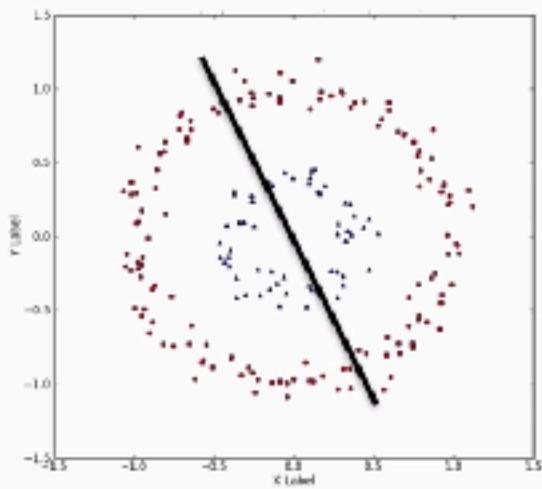
Higher Dimensions

C Parameter

Multiple Classes

Kernel Trick

Kernel Trick: Visual



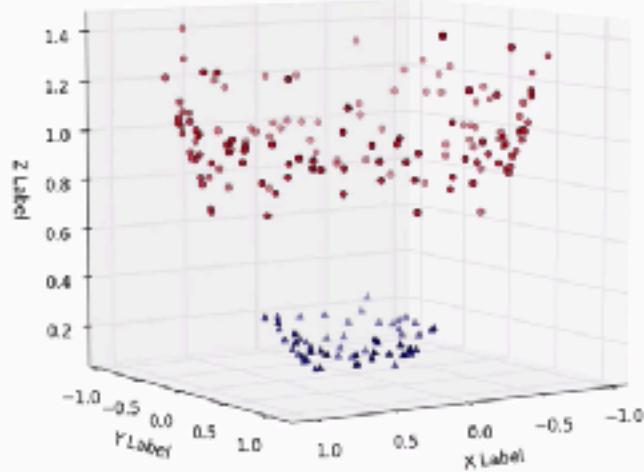
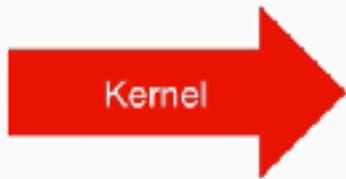
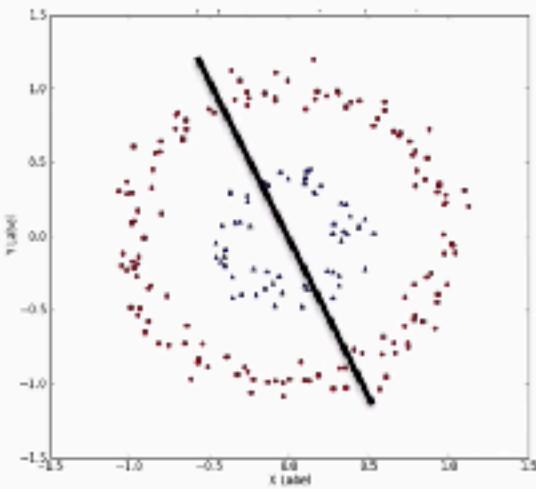
Higher Dimensions

C Parameter

Multiple Classes

Kernel Trick

Kernel Trick: Visual



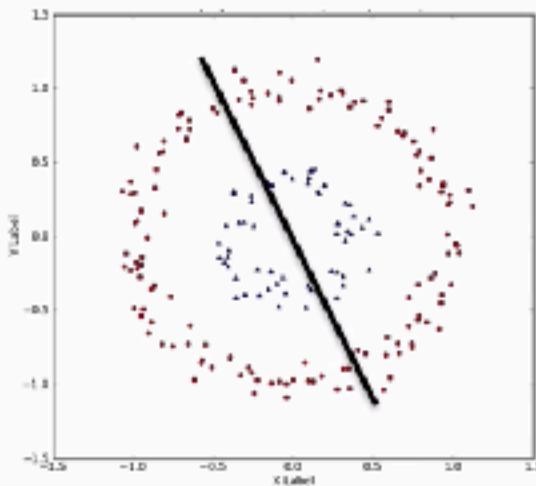
Higher Dimensions

C Parameter

Multiple Classes

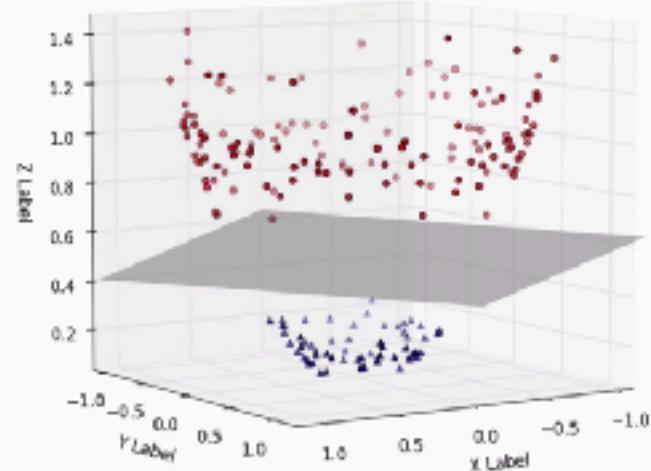
Kernel Trick

Kernel Trick: Visual



Kernel Options

- Linear
- Radial Basis Function
- Polynomial
- Sigmoid



Higher Dimensions

C Parameter

Multiple Classes

Kernel Trick

Kernel Trick: Code

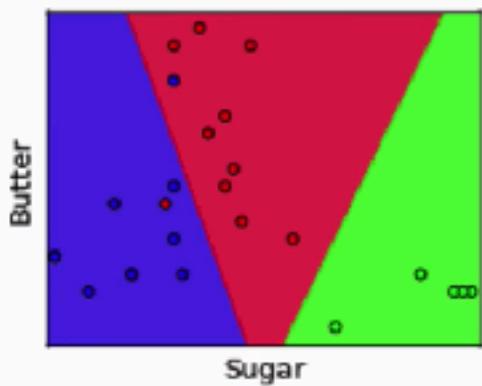
Original Code
(linear)

```
# Fit basic SVC model (linear kernel)
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

Updated Code
(RBF)

```
# Fit the SVC model with radial kernel
model = svm.SVC(kernel='rbf', C=1, gamma=2**-5)
model.fit(sugar_butter, type_label)
```

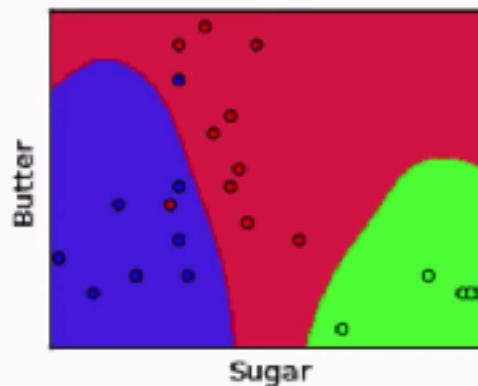
Kernel Trick: Comparison



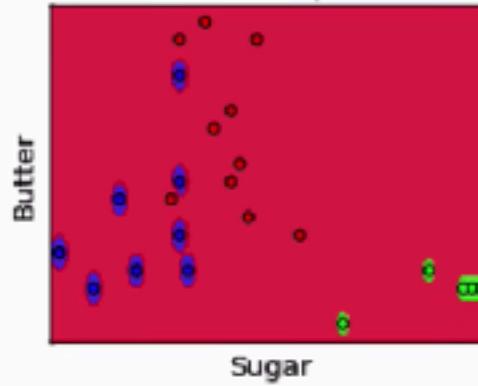
Kernel: Linear
C: 1

- Muffin
- Cupcake
- Scone

Higher Dimensions



Kernel: RBF
C: 1
Gamma: 2^{-5}



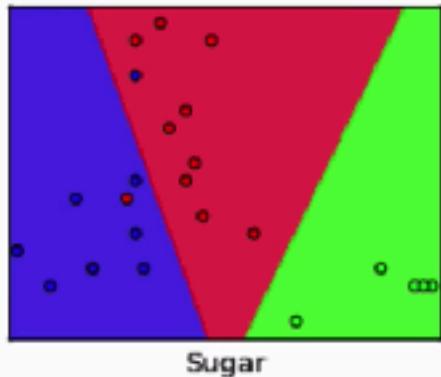
Kernel: RBF
C: 1
Gamma: 2^{-1}

Multiple Classes

Kernel Trick

Kernel Trick: Comparison

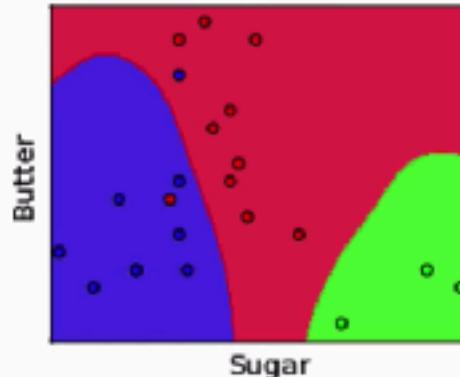
Butter



Kernel: Linear
C: 1

- Muffin
- Cupcake
- Scone

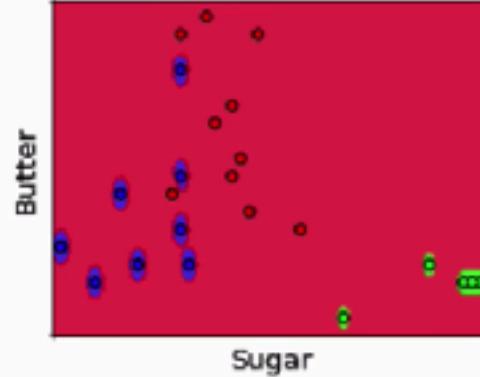
Higher Dimensions



Kernel: RBF
C: 1
Gamma: 2^{-5}

Small Gamma:
Less complexity

C Parameter



Kernel: RBF
C: 1
Gamma: 2^{+1}

Large Gamma:
More complexity

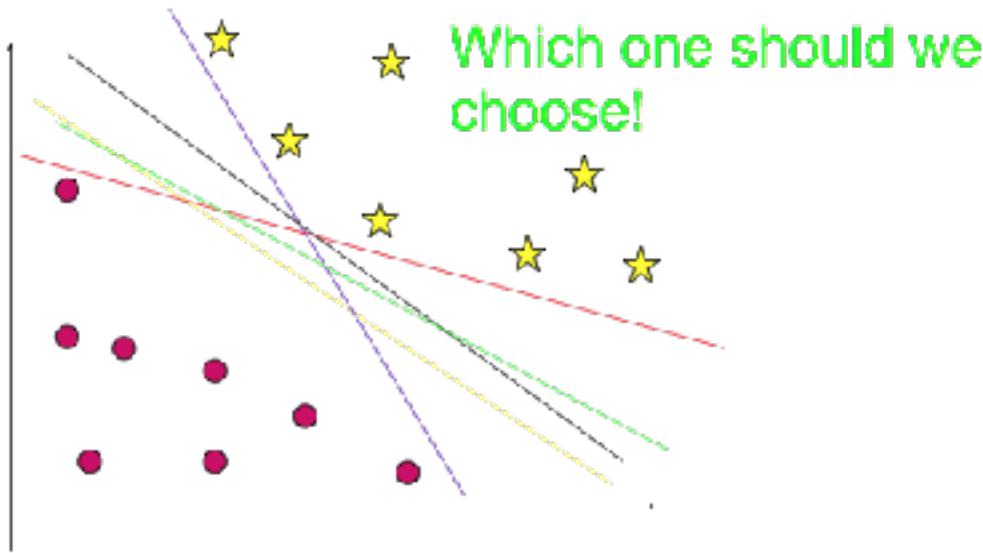
Multiple Classes

Kernel Trick

Pros and Cons of SVM

- **Pros**
 - Good at dealing with high dimensional data
 - Works well on small data sets
- **Cons**
 - Picking the right kernel and parameters can be computationally intensive

$$\begin{aligned}y_t &= +1 \\y_t &= -1\end{aligned}$$



Yes, There are many possible separating hyperplanes
It could be this one or this or this or maybe....!

SVM : Linear separable case.

The optimization problem:

-Our optimization problem so far:

I do remember the
Lagrange Multipliers
from Calculus!

$$\text{minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$



We will solve this problem by introducing Lagrange multipliers α_i associated with the constraints:

$$\text{minimize } L_p(w, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

$$\text{s.t. } \alpha_i \geq 0$$

SVM : Linear separable case.

The optimization problem cont':

So our primal optimization problem now:

$$\begin{aligned} \text{minimize } L_p(w, b, \alpha) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i(x_i \cdot w + b) - 1) \\ \text{s.t. } \alpha_i &\geq 0 \end{aligned}$$

We start solving this problem:

$$\frac{\partial L_p}{\partial w} = 0 \quad \rightarrow \quad w = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \rightarrow \quad \sum_{i=1}^n \alpha_i y_i = 0$$

SVM : Linear separable case. Introducing The Legrangian Dual Problem.

By substituting the above results in the primal problem and doing some math manipulation we get:
Lagrangian Dual Problem:

$$\begin{aligned} \text{maximize } L_D(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=0}^n \sum_{j=0}^n \alpha_i \alpha_j y_i y_j x_i' x_j \\ \text{s.t } \alpha_i &\geq 0 \text{ and } \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

$\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ are now our variables, one for each sample point x_i .

Summary

Supervised Learning

Linear Regression
Logistic Regression
SVM
Ensemble
KNN

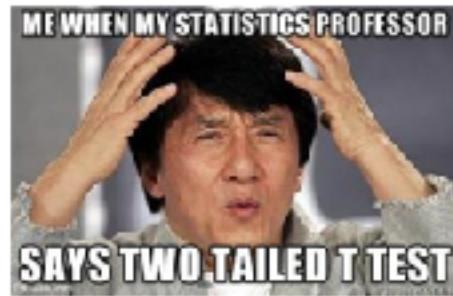
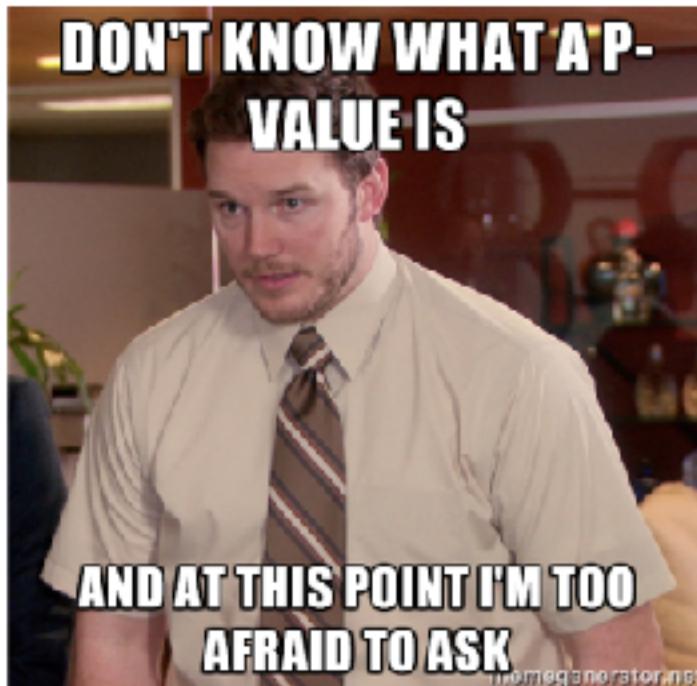
Unsupervised

K-means

Hierarchical



Revise Statistics and Probability



Learn algorithm how it works rather
concentrating on output

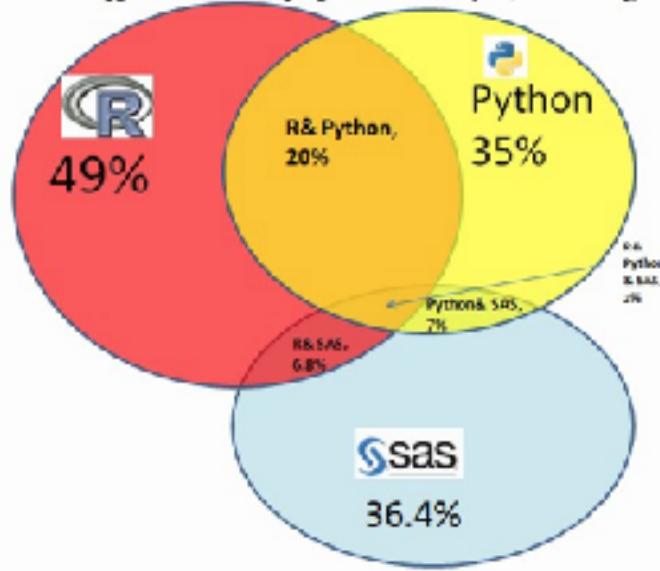


HOW IT WORKS



Learn and practice a programming language

KDnuggets 2014 Poll: Languages used for Analytics/Data Mining, 2



```
208 limit_val = 4;  
209 $("#" + limit_val).val(4);  
210 update_slider();  
211 function(limit_val){  
212 $("#" + wordlist_out).val("hi");  
213 var b = k();  
214 h();  
215 var c = 10, s = "", d = parallel();  
216 sseInt($("#slider_parallel").val());  
217 function("LDOT_total") + 40;  
218 function("read") + 4;  
219 function("check") + 4;  
220 }  
221
```

Big data and Data science are happily married



Practice all algorithm with each use case



Make use of Kaggle and Analytics vidhya

kaggle



Contribute your work to Opensource



Write a blog



Make your resume attractive

MOHAMED NOORDEEN A
Data Scientist
Tiger Analytics
mucensaz@gmail.com
<https://www.linkedin.com/in/nurensaz/>
<https://github.com/mucensaz/>
www.techanae.com
(+91) 9789-830001

PROFILE SUMMARY

- A Data Scientist with 6 years of experiences in advance analytics, Machine Learning, Predictive Modelling, Data Mining, Text Analytics and Statistical Data analysis.



MOHAMED NOORDEEN A
Senior Software Engineer/Data Scientist
TIGER ANALYTICS

CONTACT

- PHONE: +91 9789-830001
- EMAIL: mucensaz@gmail.com
- LINKEDIN: www.linkedin.com/in/nurensaz/
- GITHUB: [www.github.com/mucensaz/](https://github.com/mucensaz/)
- WEBSITE: www.techanae.com

SKILLS



DATA/ANALYTICS PROFESSIONAL

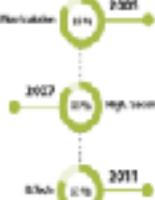
I am a Data Analytics professional with acquired skills of mining hidden patterns from large sets of unstructured, semi-structured and unstructured data. I am proficient in Python for building analytical models and R for statistical analysis.

SUPERIOR SKILLS

Developed a robust data mining system in the domain of medical fraud detection for two-clients based in USA. Developed a robust machine learning system using unsupervised learning techniques like K-means, DBSCAN, etc. for the said clients.

Implemented using linear scaling model for predicting crime problems in online competitions. Implemented various machine learning models for predicting traffic jams.

ACADEMIC



CORE COMPETENCIES



CERTIFICATION

- 2017 Certified Programmer Big Data Analytics & Administration at INSOFT Group
- 2016 Certified Data Mining developer by Infoware TACT
- 2013 Oracle Certified Professional Java SE Programmer
- 2012 Oracle Database 11g Certified Expert

Contact

Mohamed Noordeen
nursnaaz@gmail.com
www.technaaz.com



$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\frac{P}{1-P} = e^{a+bX}$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

