Mohamed Noordeen Alaudeen

Senior Data Scientist - Logitech

About me

– – –

TIGER ANALYTICS

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.

INSOFE

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

lti

Dr. Dakshinamurthy V Kolluru
President

Dr. Sridhar Pappu
Executive VP - Academics

GUEST LECTURES

NOT THE FIRST TIME FOR US

MY WORK

https://github.com/nursnaaz
https://www.linkedin.com/in/nursnaaz/
http://www.technaaz.com/

Basic Statistics

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	2.333333333	2.333333333	5.444444444
3	4	16	-0.6666666667	0.6666666667	0.4444444444
4	2	4	-2.666666667	2.666666667	7.111111111
5	6	36	1.333333333	1.333333333	1.777777778
6	2	4	-2.666666667	2.666666667	7.111111111
7	8	64	3.333333333	3.333333333	11.11111111
8	5	25	0.3333333333	0.3333333333	0.1111111111
9	3	9	-1.666666667	1.666666667	2.777777778
SUM	42	232	0	15.33333333	36
Average	4.666666667	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

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

MEAN = MEDIAN = 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

MEAN = MEDIAN = MODE = 10 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

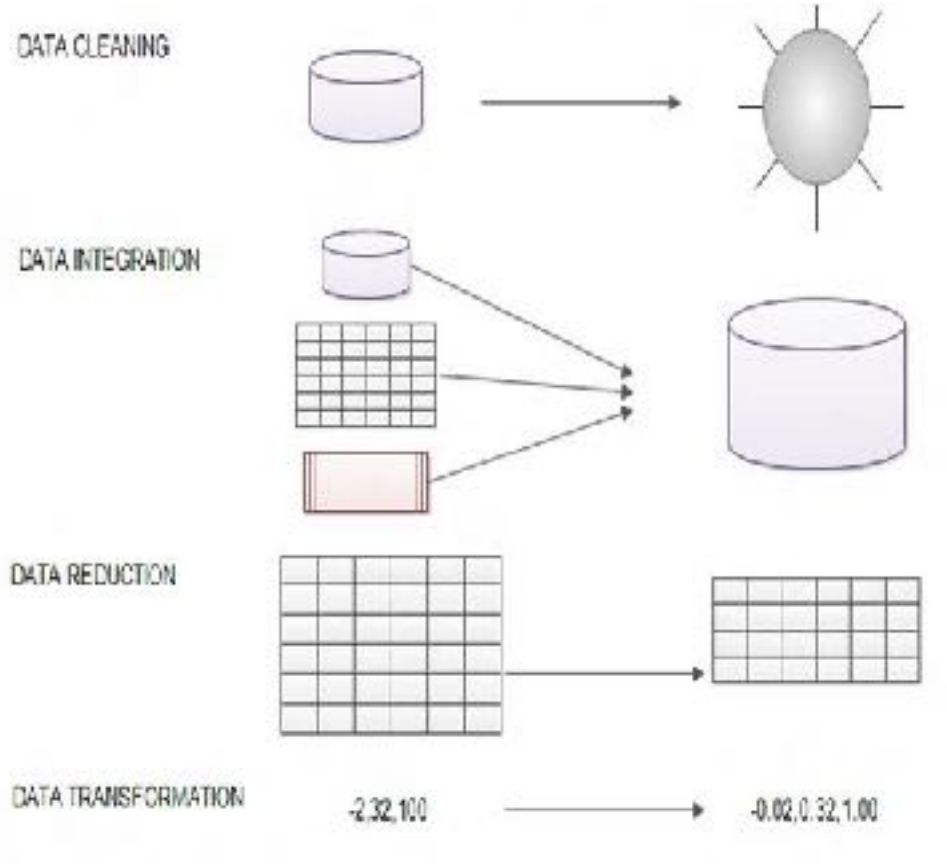
Real World Data

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

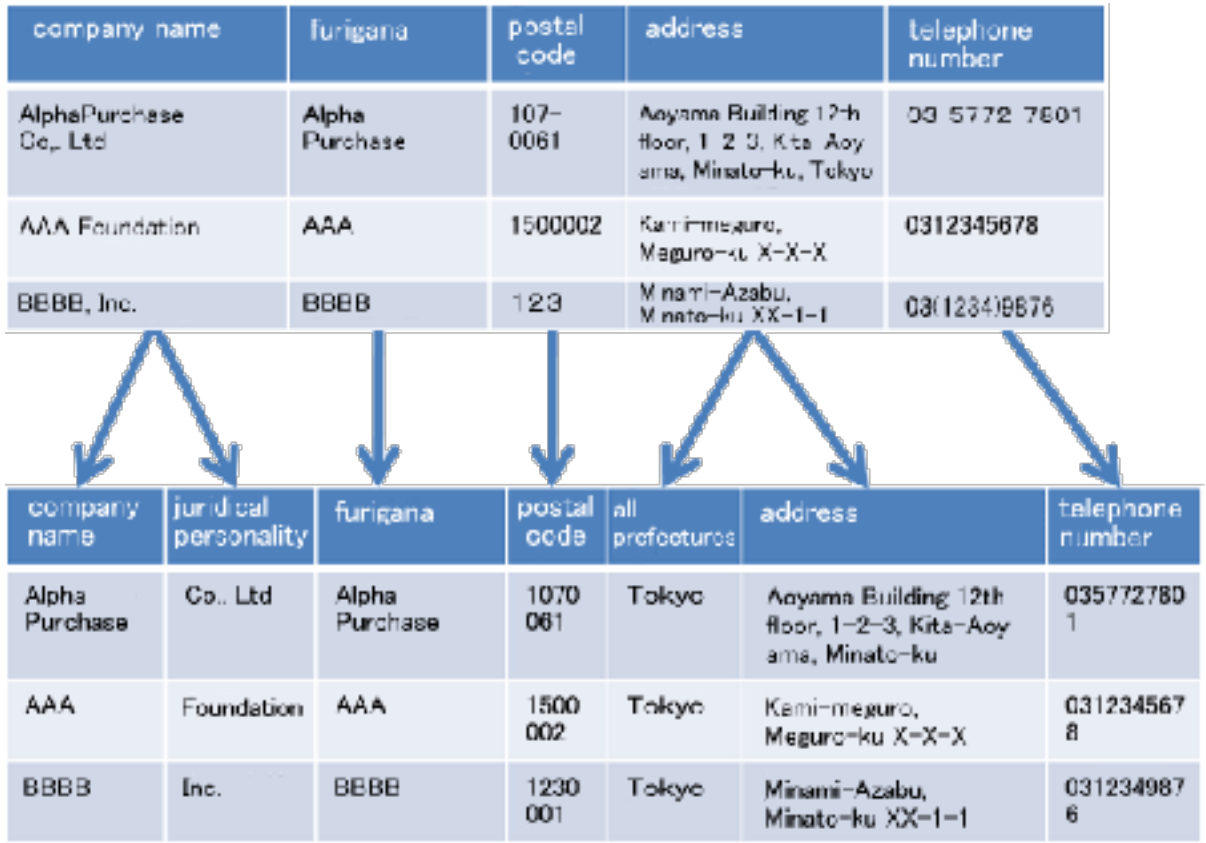
Data Preprocessing

- 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



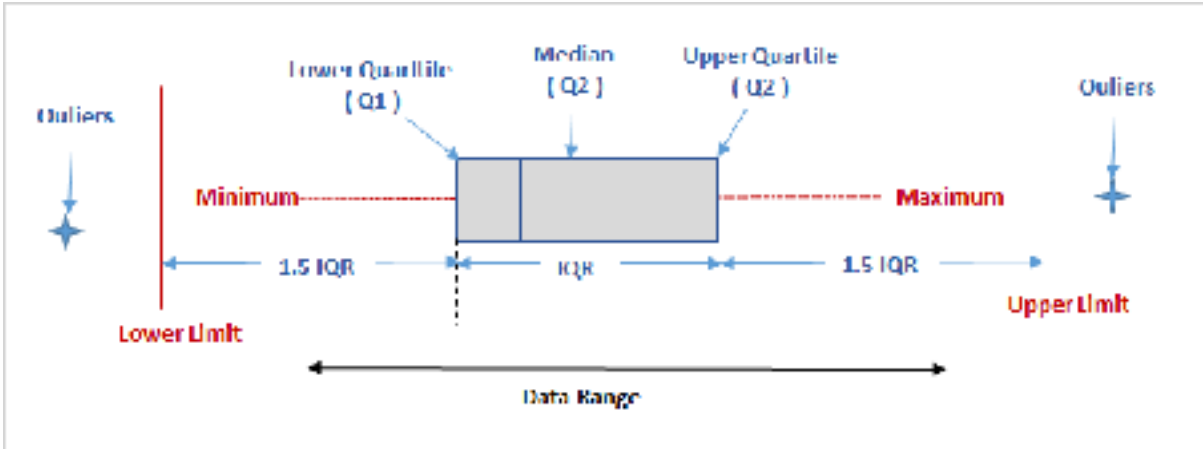
Imputation

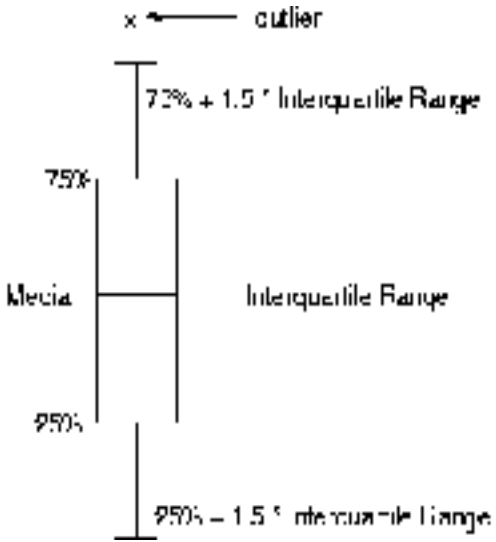
- Replace with mean or a median
- When to use mean?
- Replace with nearest neighbour
- How much nearest to see?

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

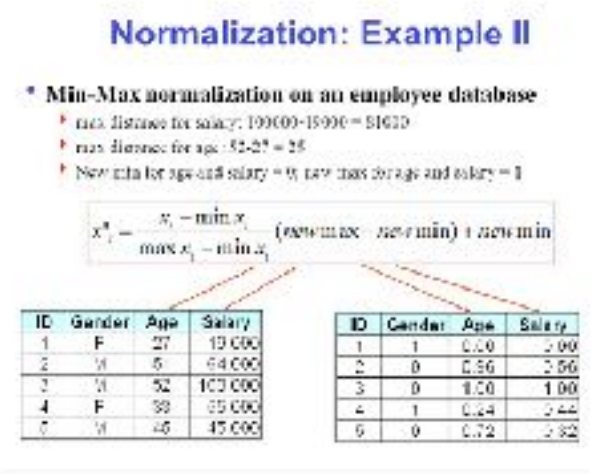
- Normalization

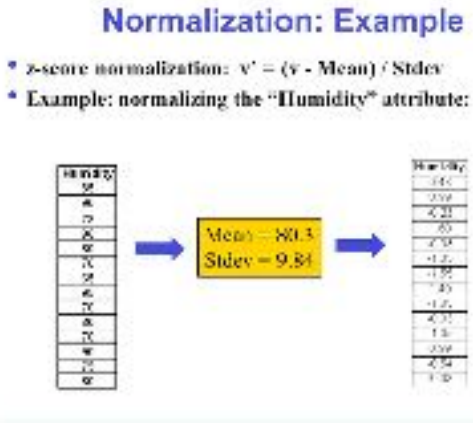
Min-max normalization

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

Decimal scaling

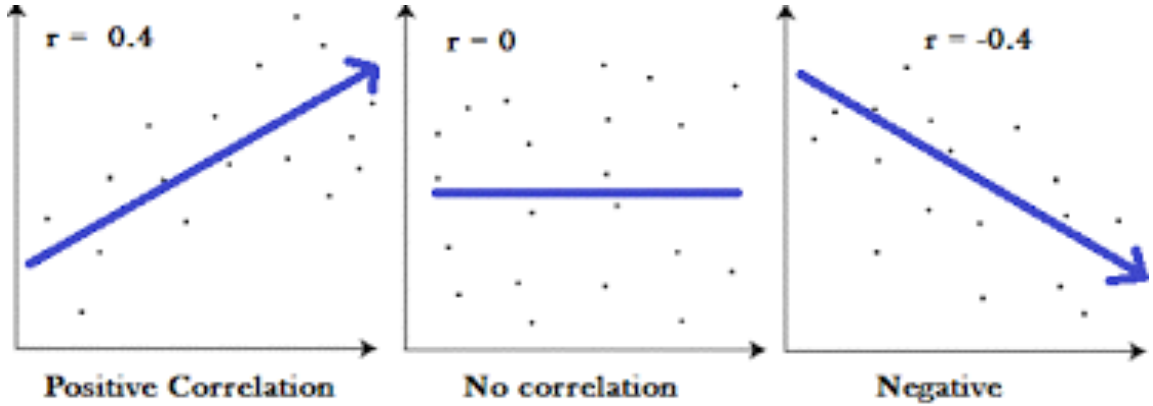
v= v/10^j





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

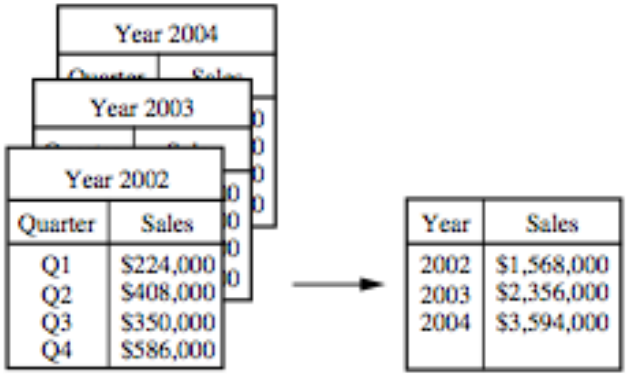


Figure 2.13 Sales data for a given branch of *AllElectronics* for the years 2002 to 2004. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.

Relationship

Y = ?????????

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

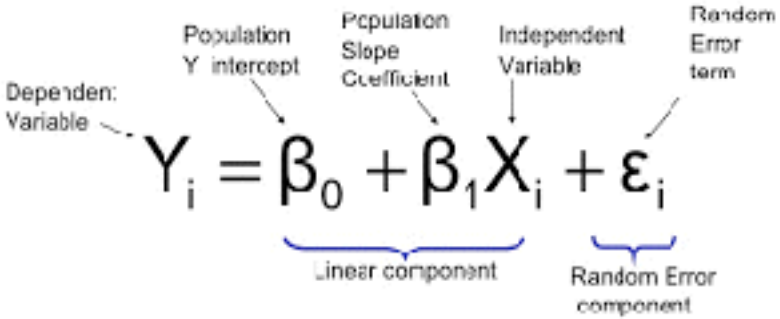
Linear regression equation
(without error)

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

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



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

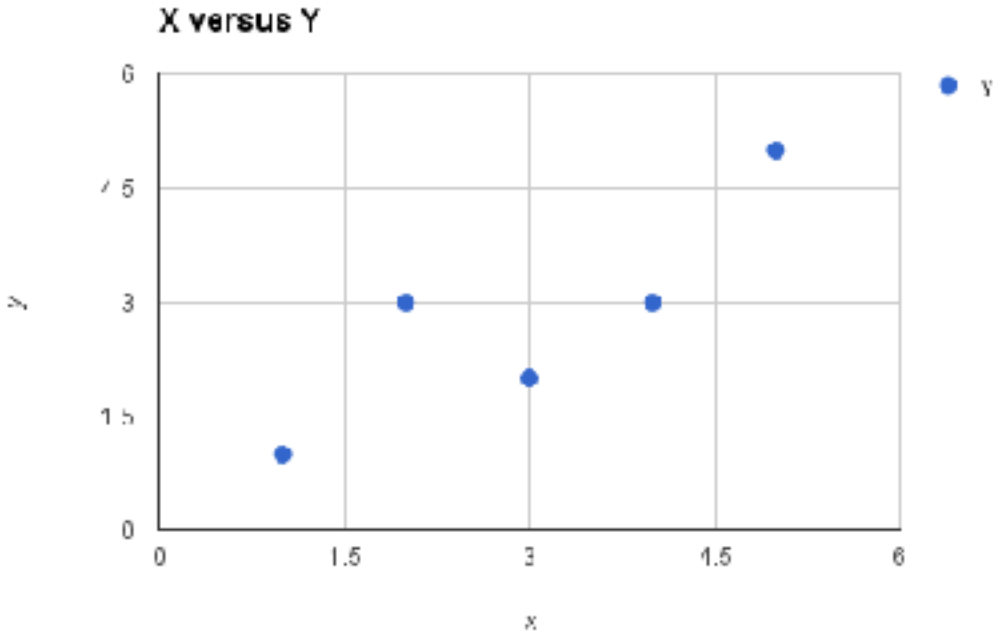
What is Linear Regression?

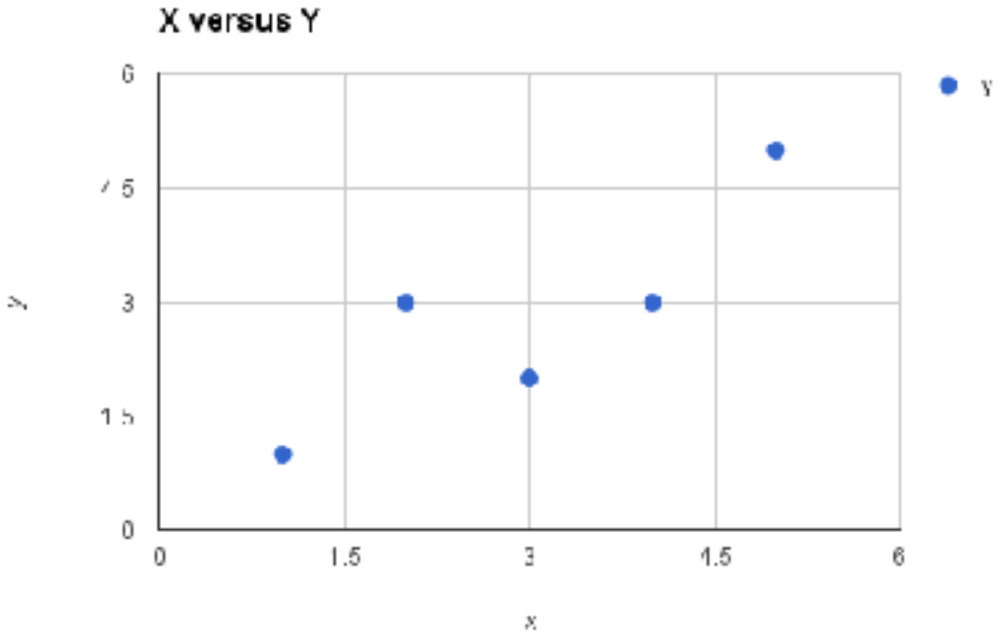
What is Linear Regression?

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
6
4.5
3
1.5
0
y
0
1.5
3
4.5
6
x
Y



$$y = B0 + B1 * x$$

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

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

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

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

B1 = 8 / 10

B1 = 0.8

B0 = mean(y) – B1 * mean(x)

or

B0 = 2.8 – 0.8 * 3

or

B0 = 0.4

y = B0 + B1 * x

or

y = 0.4 + 0.8 * x

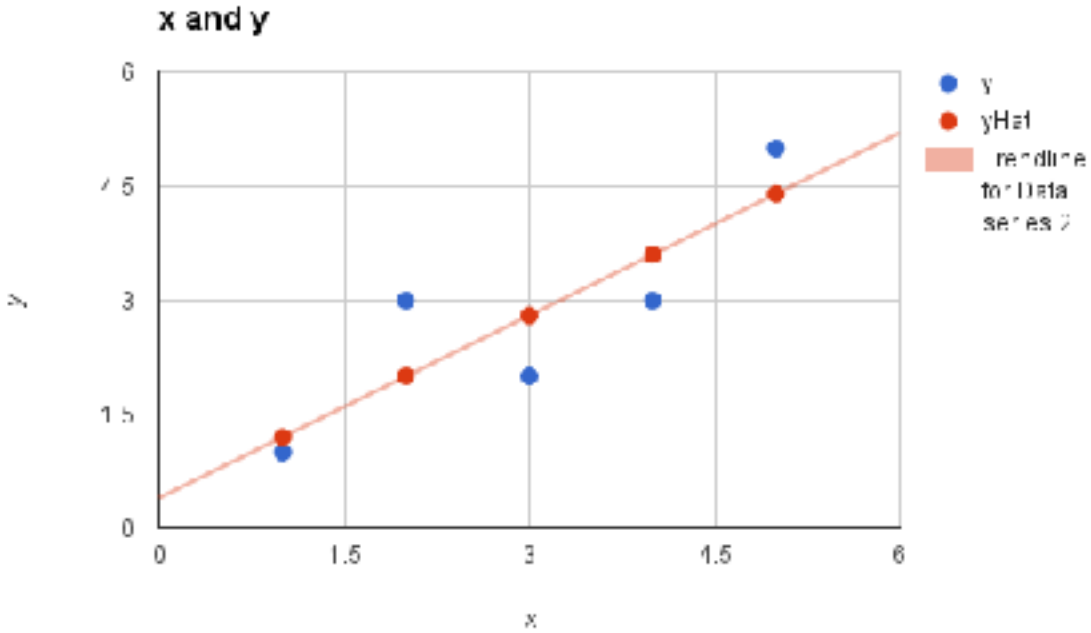
x - mean(x)	y - mean(y)	Multiplication
-2	-1.8	3.6
-1	0.2	-0.2
1	0.2	0.2
0	-0.8	0
2	2.2	4.4

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

8

10

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



RMSE = 0.692

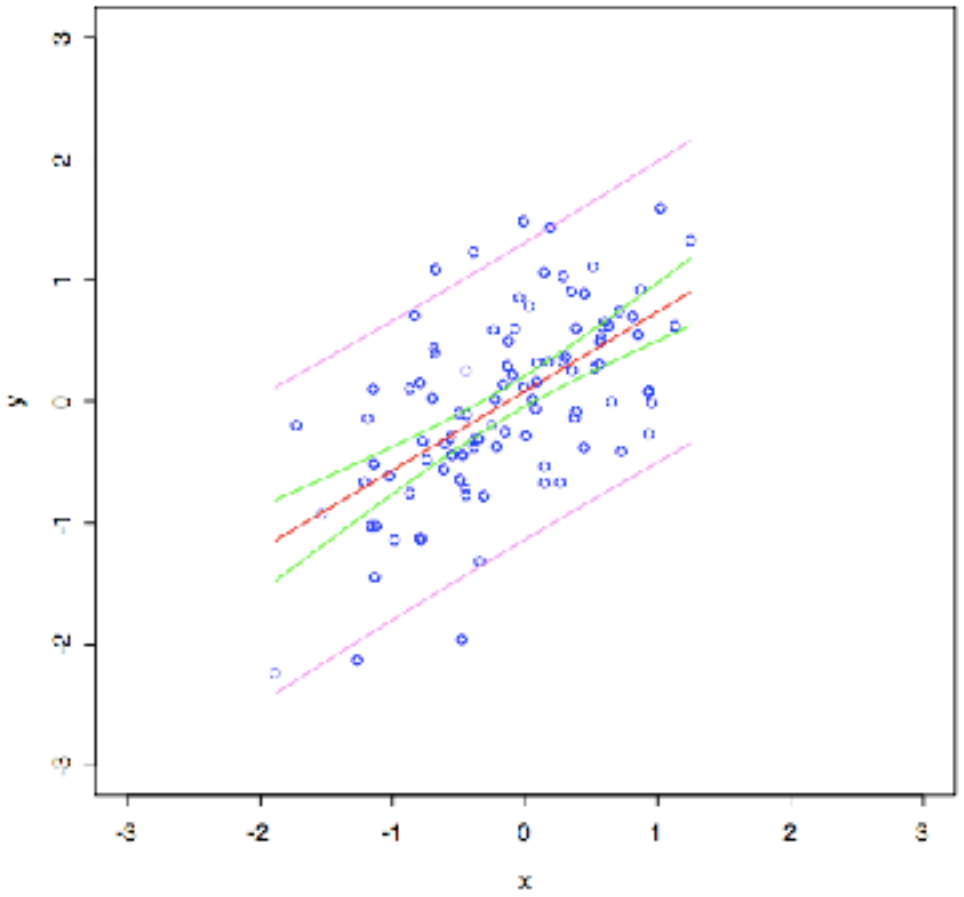
Gradient Descent

Finding the optimum relationship where the error is minimal.

Finding the intercept and coefficients value.

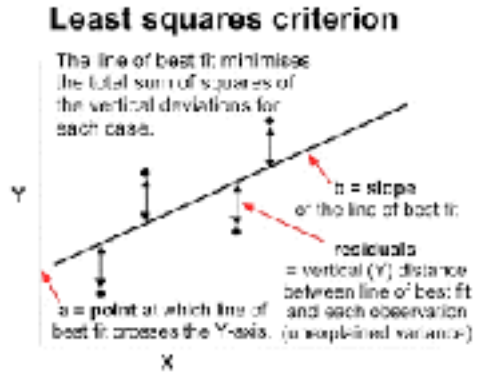
Find the solution?

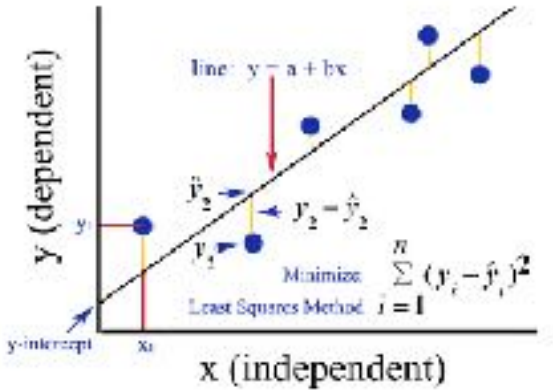
Any Suggestions?

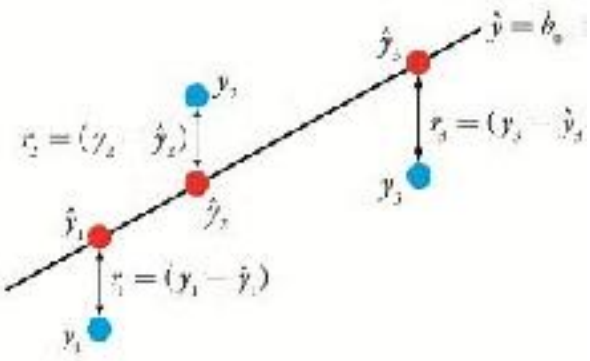


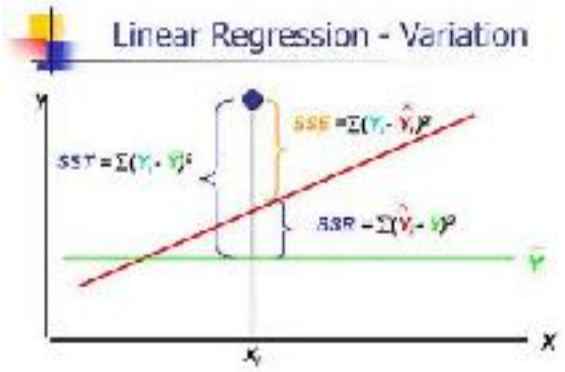
Line of best fit

Ordinary least square line









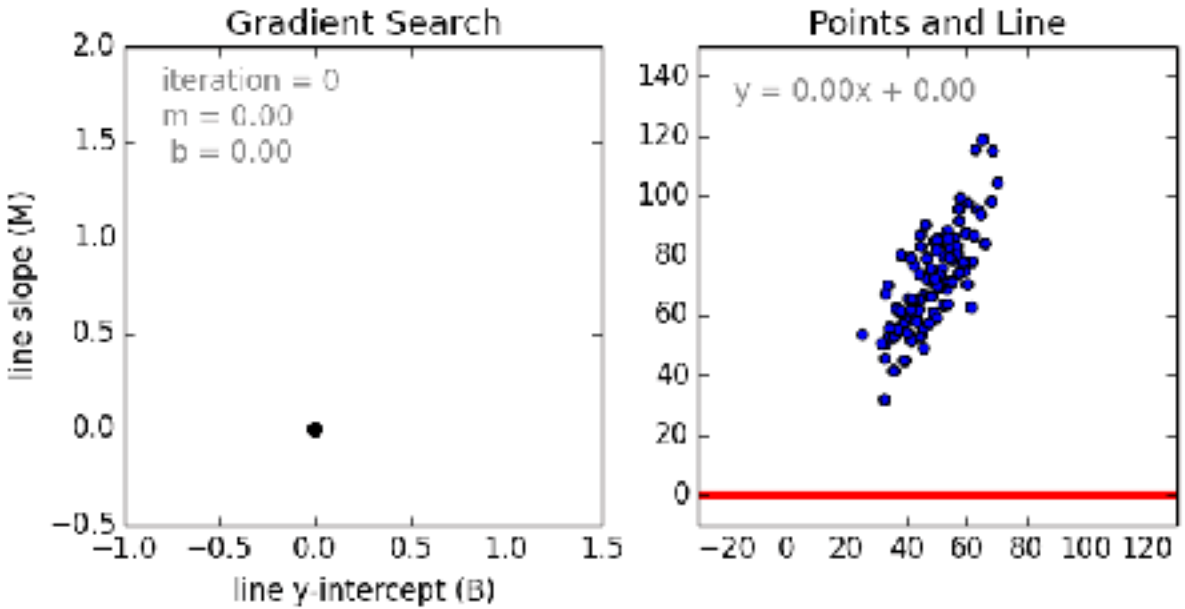
Cost Function

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

Gradient Descent

Learning Rate

Momentum



Partial Derivative

Finding the direction of coefficient and slope moves in.

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

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

Error Metrics for Regr

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

$$r^2 = 1 - \frac{\text{SS Error}}{\text{SS Total}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

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:

```
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.

```
height=[[4.0],[4.5],[5.0],[5.2],[5.4],[5.8],[6.1],[6.2],[6.4],[6.8]]
weight=[  42 ,  44 , 49, 55  , 53  , 58   , 60  , 64  ,  66 ,  69]

print("height weight")
for row in zip(height, weight):
    print(row[0][0],"->",row[1])
```

Step 3

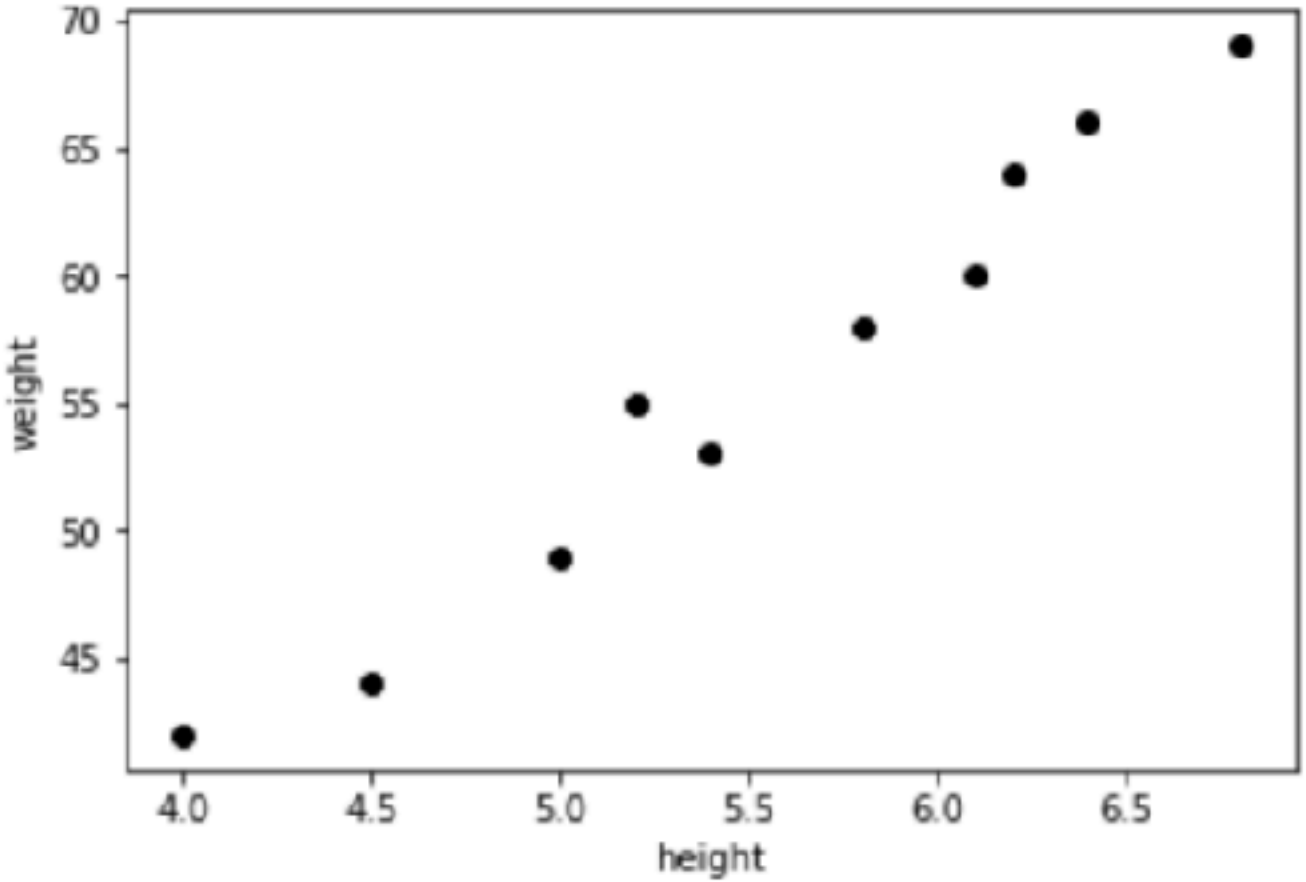
`import` statement to plot graph using `matplotlib` :

```
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:

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

Slope and intercept:

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

Output:

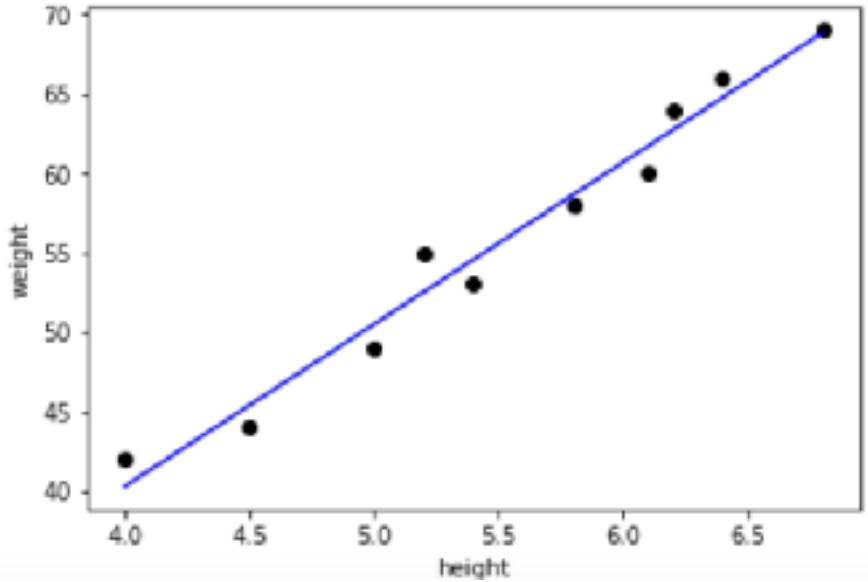
```
slope= 10.1936218679 intercept= -0.4726651480
```

Step 5

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

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

Output:



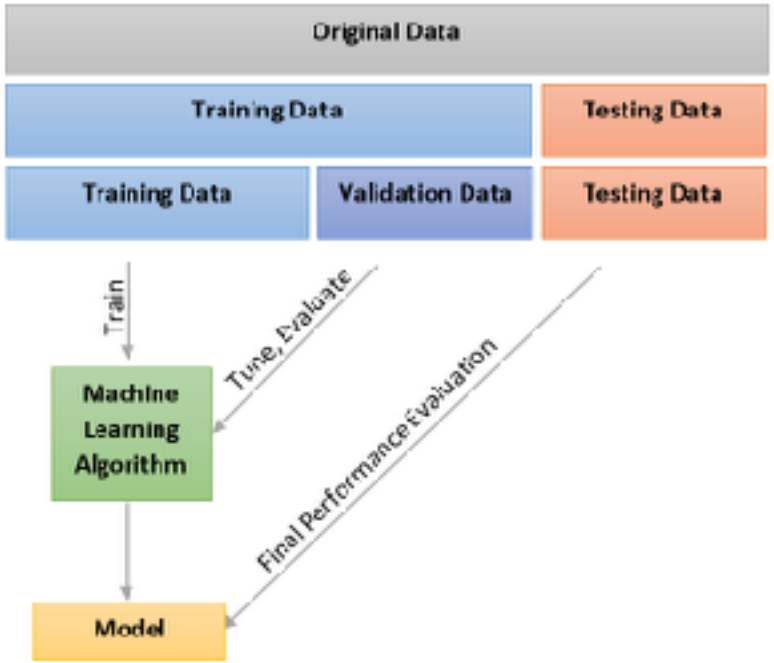
Advantage of Linear Regression

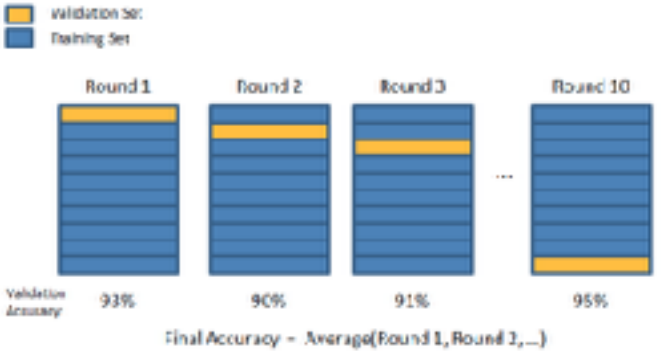
- Linear regression implements a statistical model that, when relationships between the independent variables and the dependent variable are almost linear, shows optimal results.
- Best place to understand the data analysis
- Easily Explicable

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)

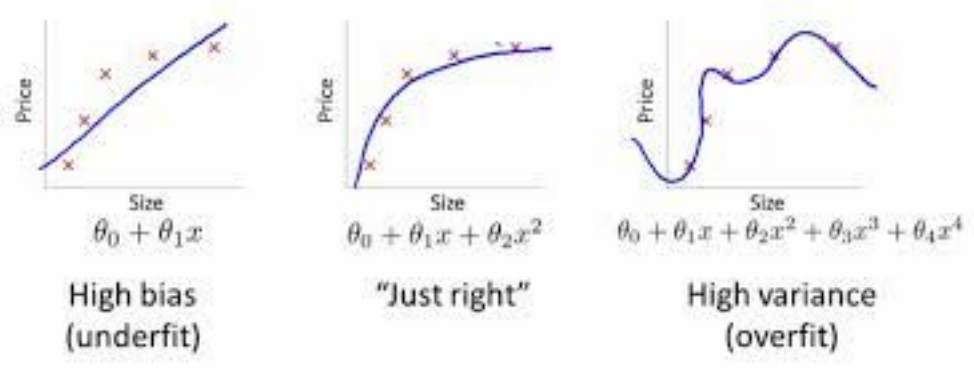
Testing Data (More Error)

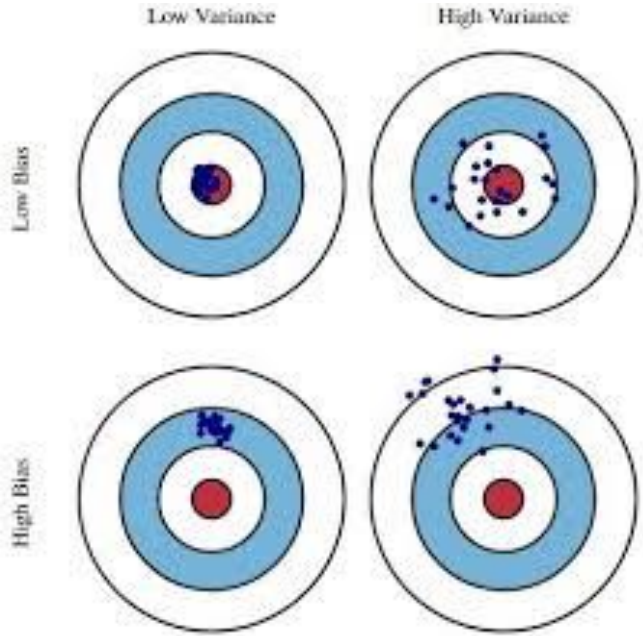
Overfitting vs Underfitting

Training Data (More Error)

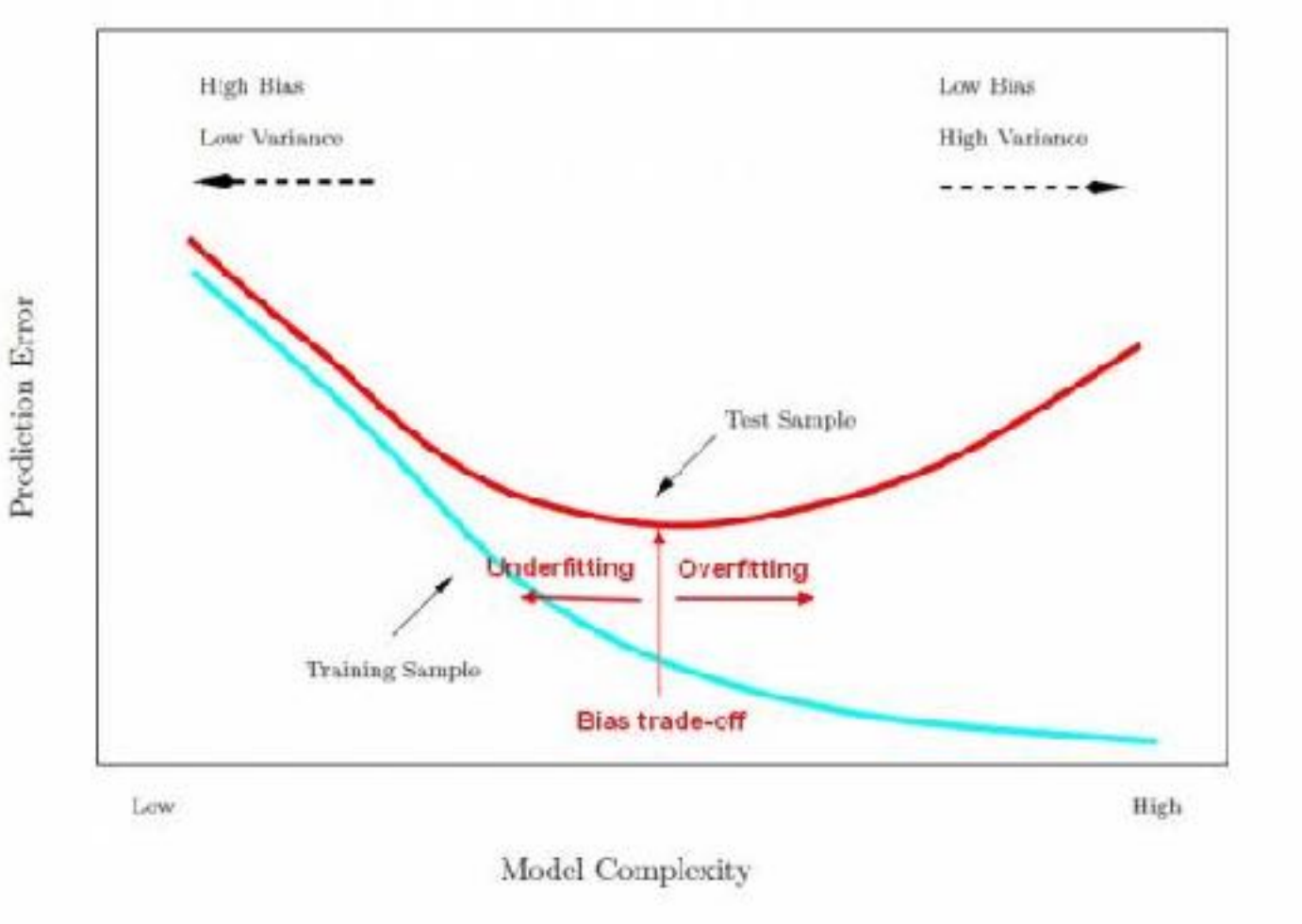
Testing (Still More Error)

Variance and Bias Trade off





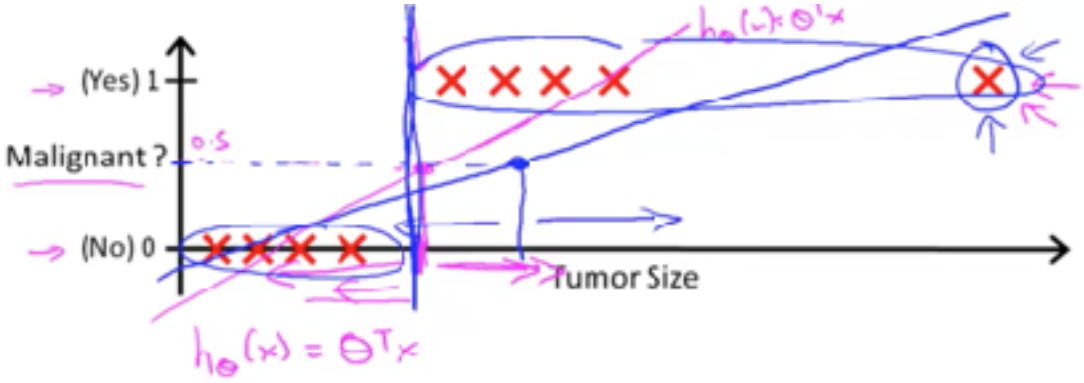
Ideal Model should have Low varinance and Low Bias


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Underfitting
Overfitting
Training Sample
Bias trade-off
Low
High
Model Complexity

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) = βo + β(Age)

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

p = exp(βo + β(Age)) = e^(βo + β(Age)) ------- (b)

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(βo + β(Age)) / exp(βo + β(Age)) + 1 = e^(βo + β(Age)) / e^(βo + β(Age)) + 1 ----- (c)

p = e^y/ 1 + e^y --- (d)

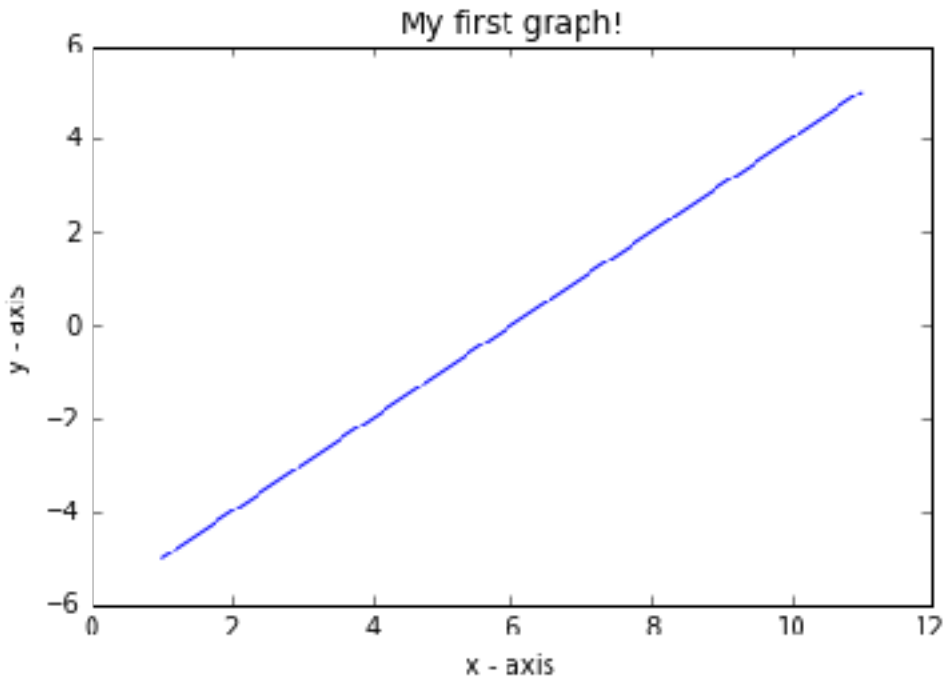
q = 1 - p = 1 - (e^y/ 1 + e^y) --- (e)

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

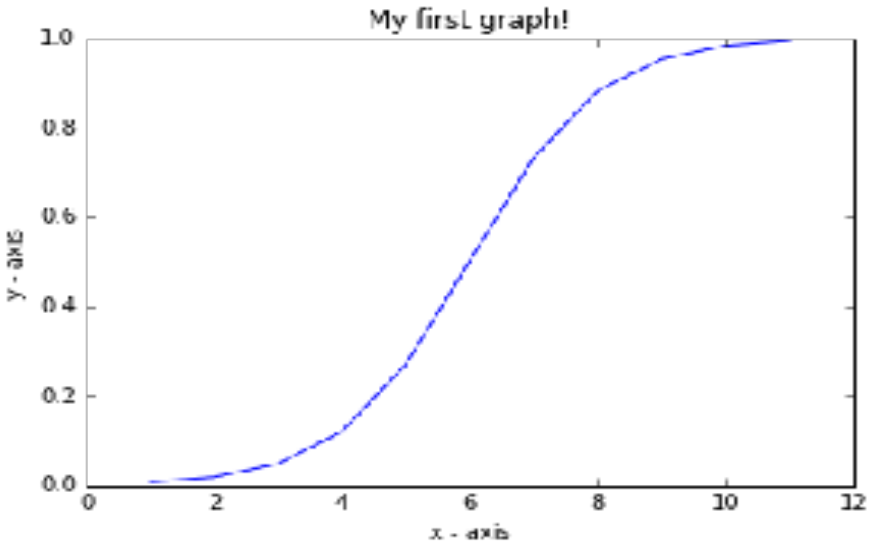
$$\log\left(\frac{p}{1-p}\right) = y$$

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

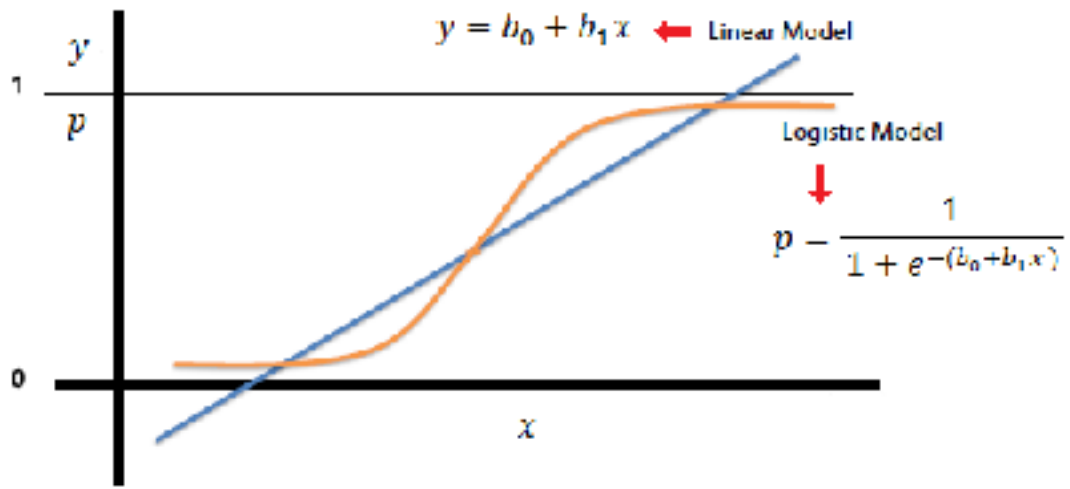
x	Y
1	-5
2	-4
3	-3
4	-2
5	-1
6	0
7	1
8	2
9	3
10	4
11	5



x	Sigmoid(Y)
1	0.006692850924
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 categorial 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)',
                  data, return_type="dataframe")
print X.columns
```

```
Index([u'Intercept', u'C(occupation)[T.2.0]', u'C(occupation)[T.3.0]', u'C(occupation)[T.4.0]', u'C(occupation)[T.
```

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.

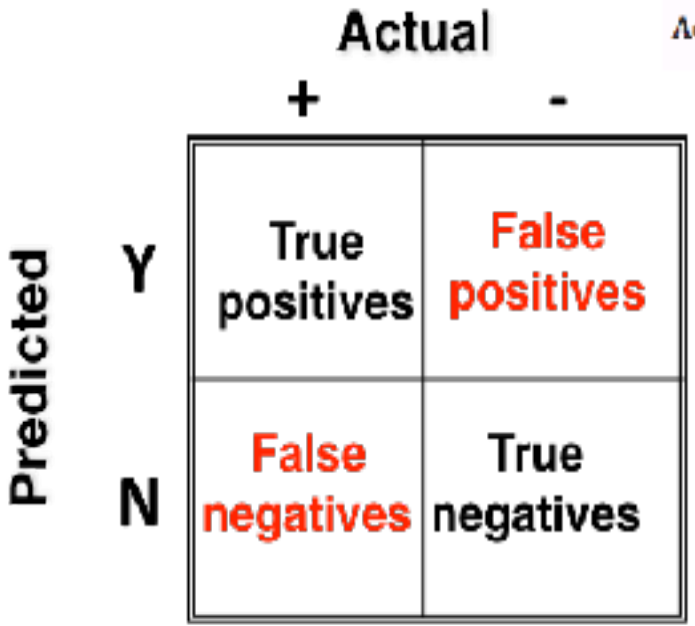
In [16]:
```
# evaluate the model by splitting into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

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



$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{\text{True positives} + \text{False positives} + \text{True negatives} + \text{False negatives}}$$

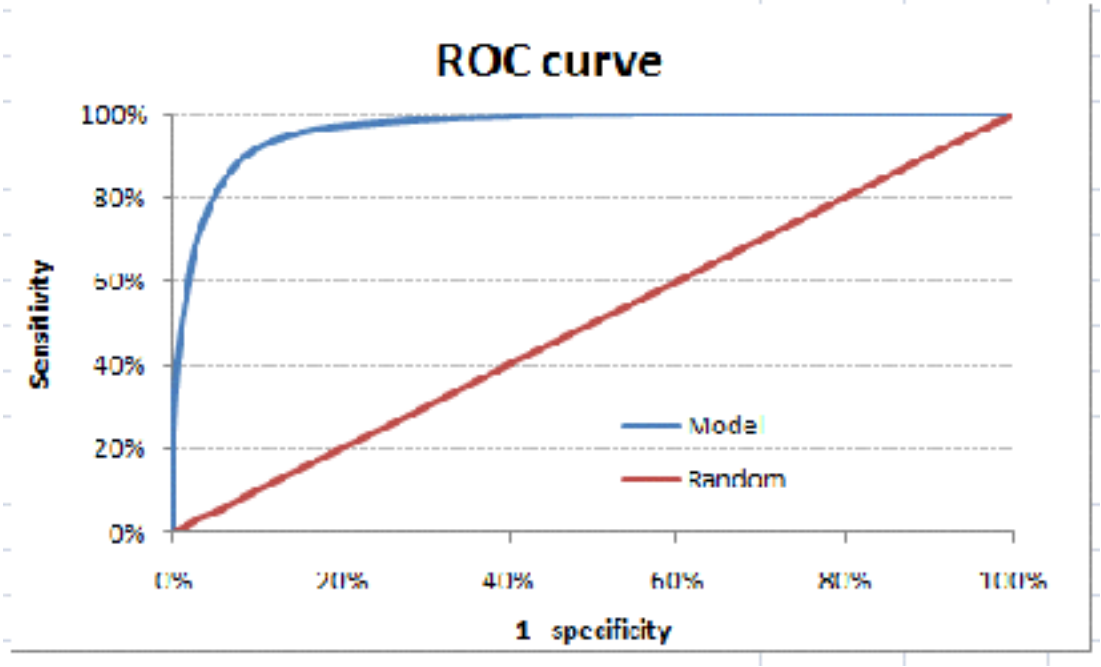
- **true positive rate** = TP/(TP+FN) = 1 – false negative rate
- **false positive rate** = FP/(FP+TN) = 1 – true negative rate
- **sensitivity** = true positive rate
- **specificity** = true negative rate
- **positive predictive value** = TP/(TP+FP)
- **recall** = TP / (TP+FN) = true positive rate
- **precision** = TP / (TP+FP)
- **F-score** is the harmonic mean of precision and recall:
- **G-score** is the geometric mean of precision and recall:

$$F' = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

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 regularization 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 kc in N that belong to c

4. Estimate P(cId) as kc/k

5. Choose as class argmaxc P(cId) [= majority class]

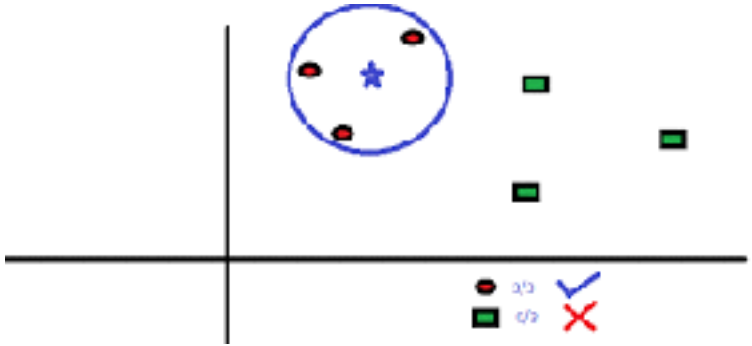
Finding similar rows

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 that 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 that 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 that 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

```
import numpy as np
from sklearn.preprocessing import Imputer
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Step 2 - Import Data

Step 3

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

Step 4

```
for K in range(25):
 K_value = K+1
 neigh = KNeighborsClassifier(n_neighbors = K_value, weights='uniform', algorithm='auto')
 neigh.fit(X_train, y_train)
 y_pred = neigh.predict(X_test)
 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

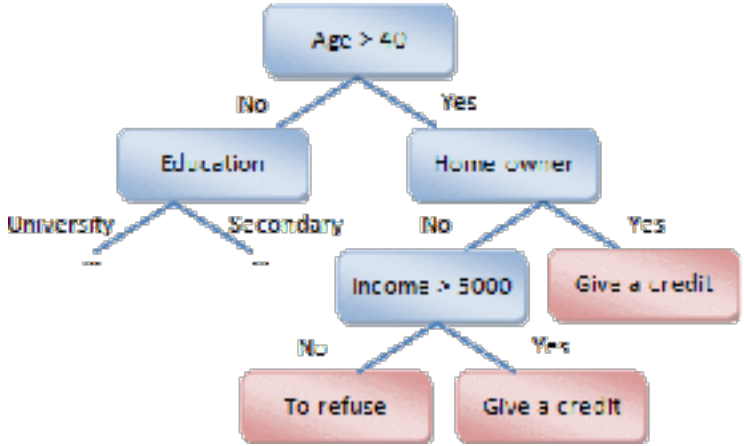
KNN Disadvantage

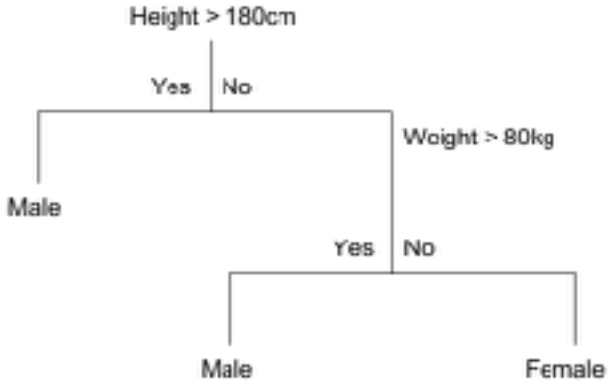
- Large search problem to find nearest neighbours
- Storage of data
- Must know we have a meaningful distance function

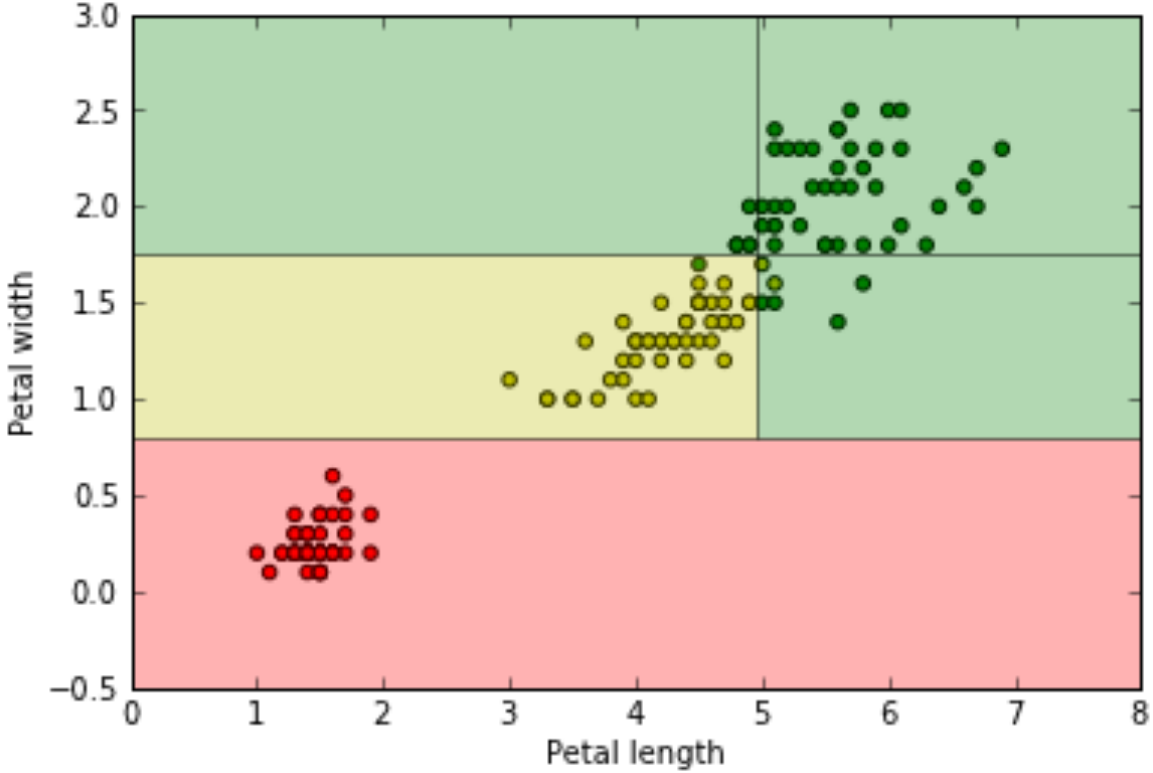
Decision Tree

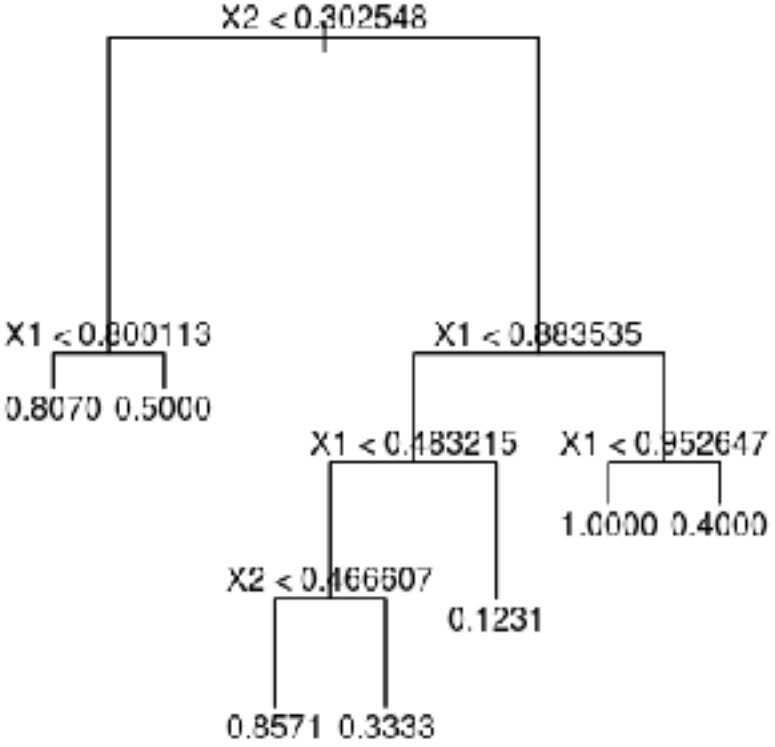
How it works?

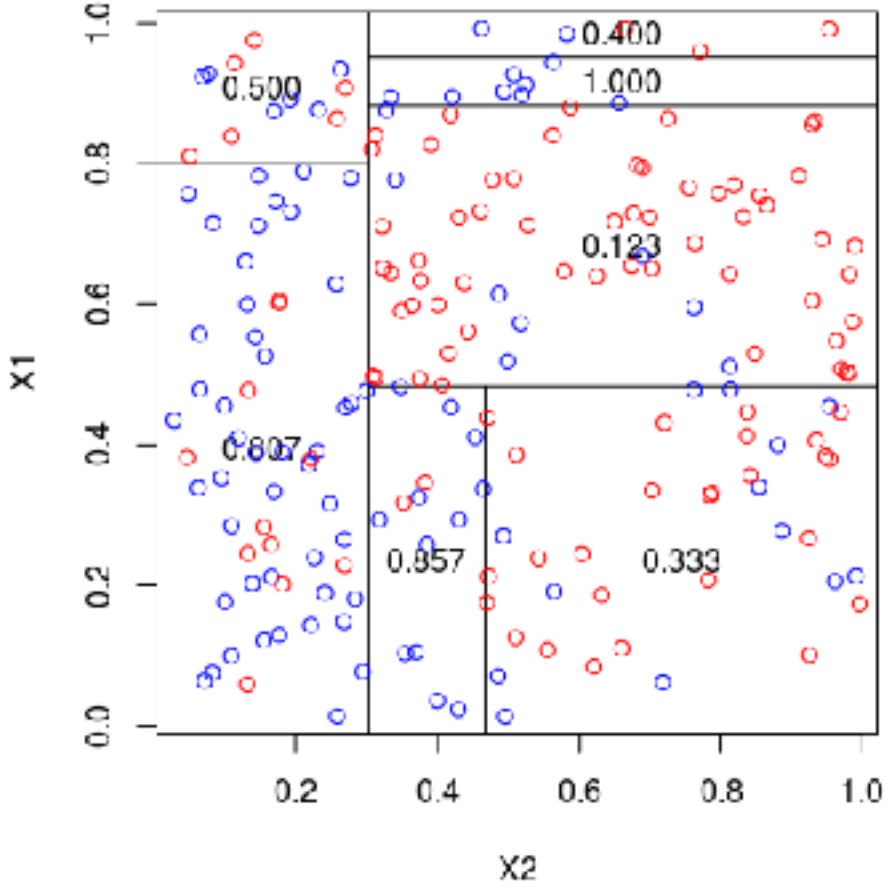
- Decision tree uses a tree structure to represent number of possible decision paths and an outcome for each path
- Decision Trees can be applied to both classification & regression problems
- Classification:
  Entropy,Information Gain
- Regression:
  Mean Square Error






3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
Petal width
0
1
2
3
4
5
6
7
8
Petal length


X2 < 0.302548
X1 < 0.000113
X1 < 0.883535
0.8070 0.5000
X1 < 0.183215
X1 < 0.952647
1.0000 0.4000
X2 < 0.166607
0.1231
0.8571 0.3333


0.500
0.400
1.000
0.123
0.807
0.857
0.333
X1
X2
0.0
0.2
0.4
0.6
0.8
1.0

— — —

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

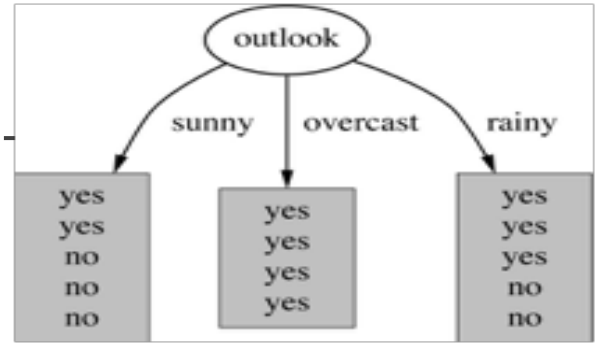
- Then: split instances into subsets

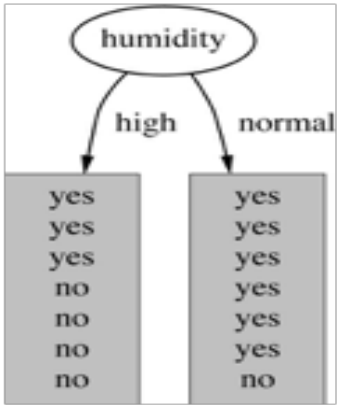
One for each branch extending from the node

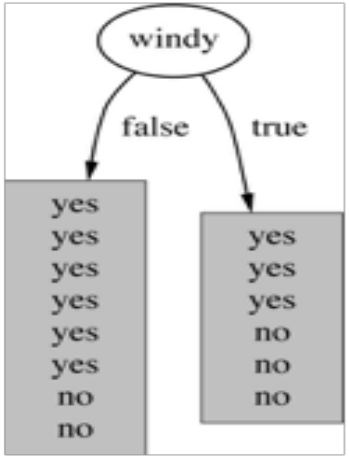
- Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

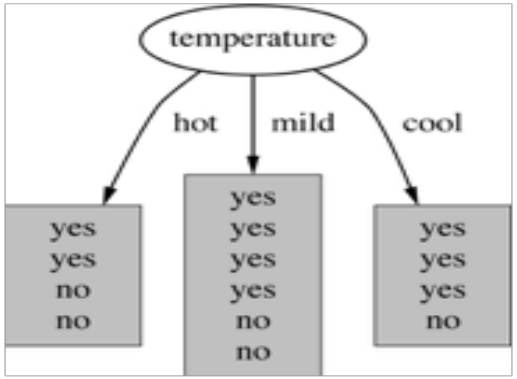
OUTLOOK	TEMP	HUMIDITY	WINDY	PLAY
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny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

Which attribute to select?

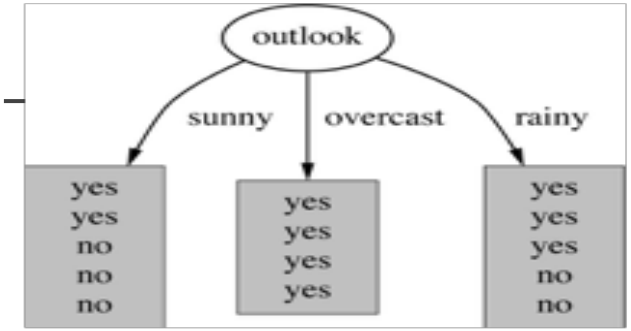


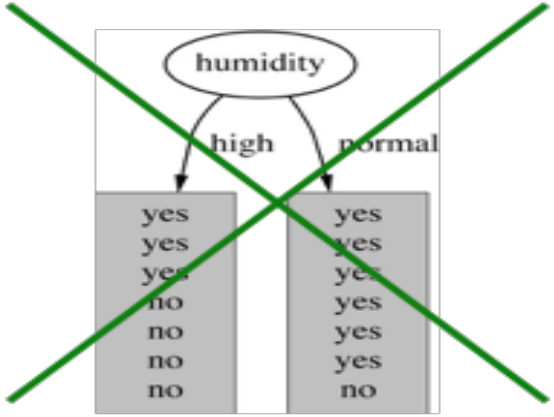


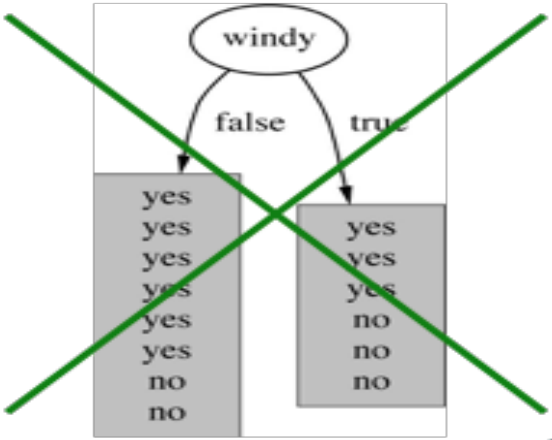


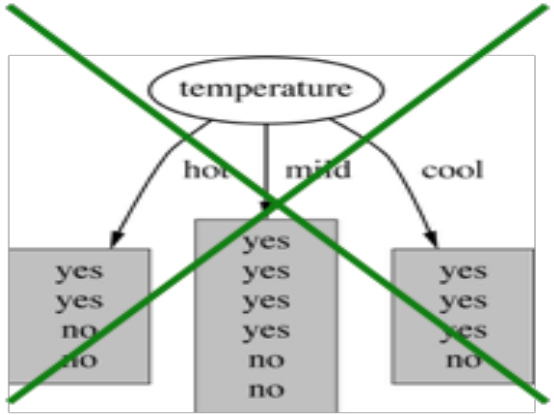


Which attribute to select?

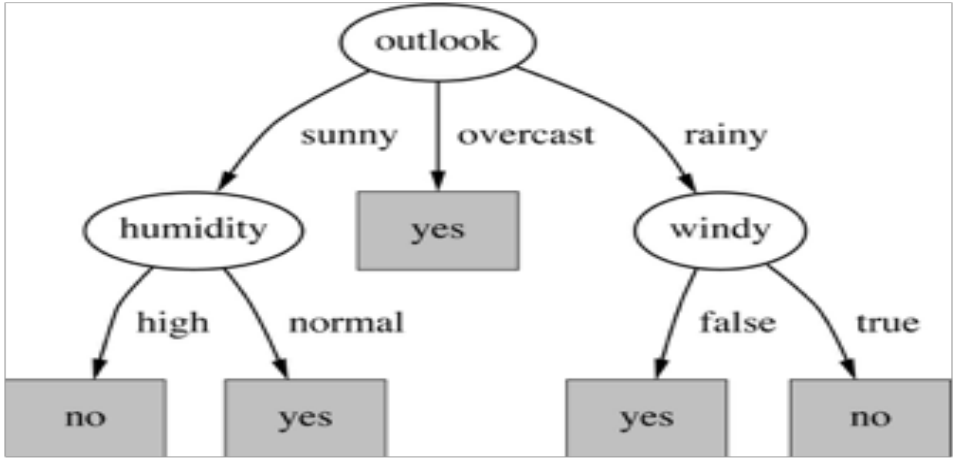








Final decision tree



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

Computing Information Gain

- Information gain: information before splitting – information after splitting

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

- Information gain for attributes from weather data:

gain(*Outlook*)	= 0.247 bits
gain(*Temperature*)	= 0.029 bits
gain(*Humidity*)	= 0.152 bits
gain(*Windy*)	= 0.048 bits

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

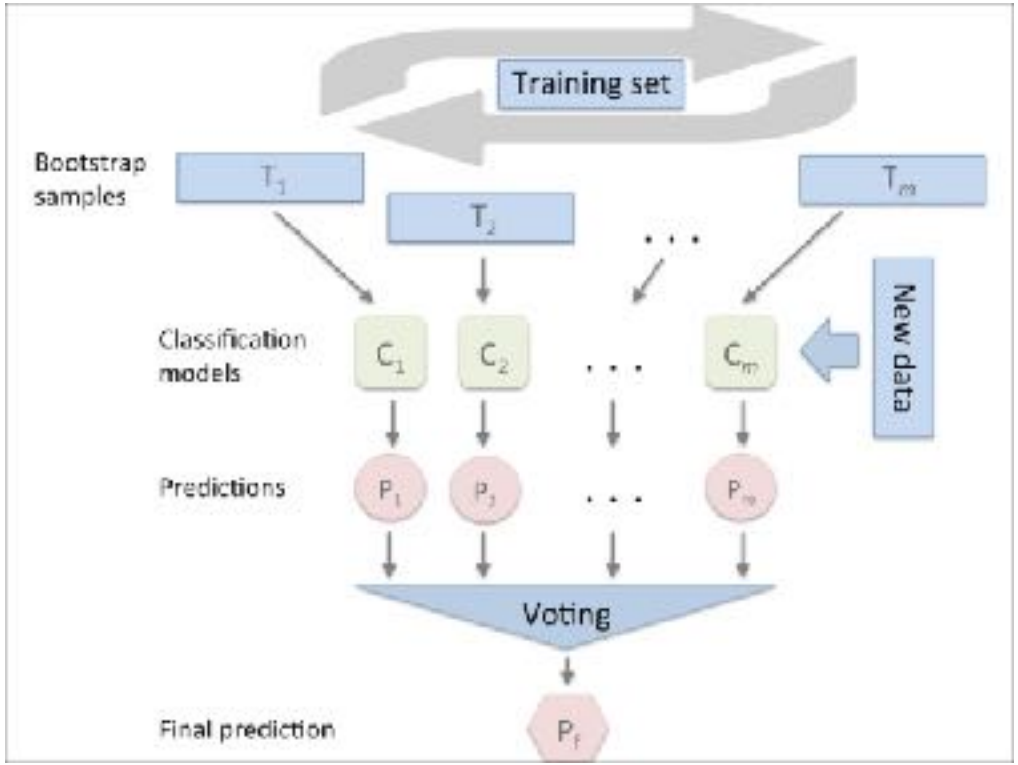
Decision Tree Cons

- 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

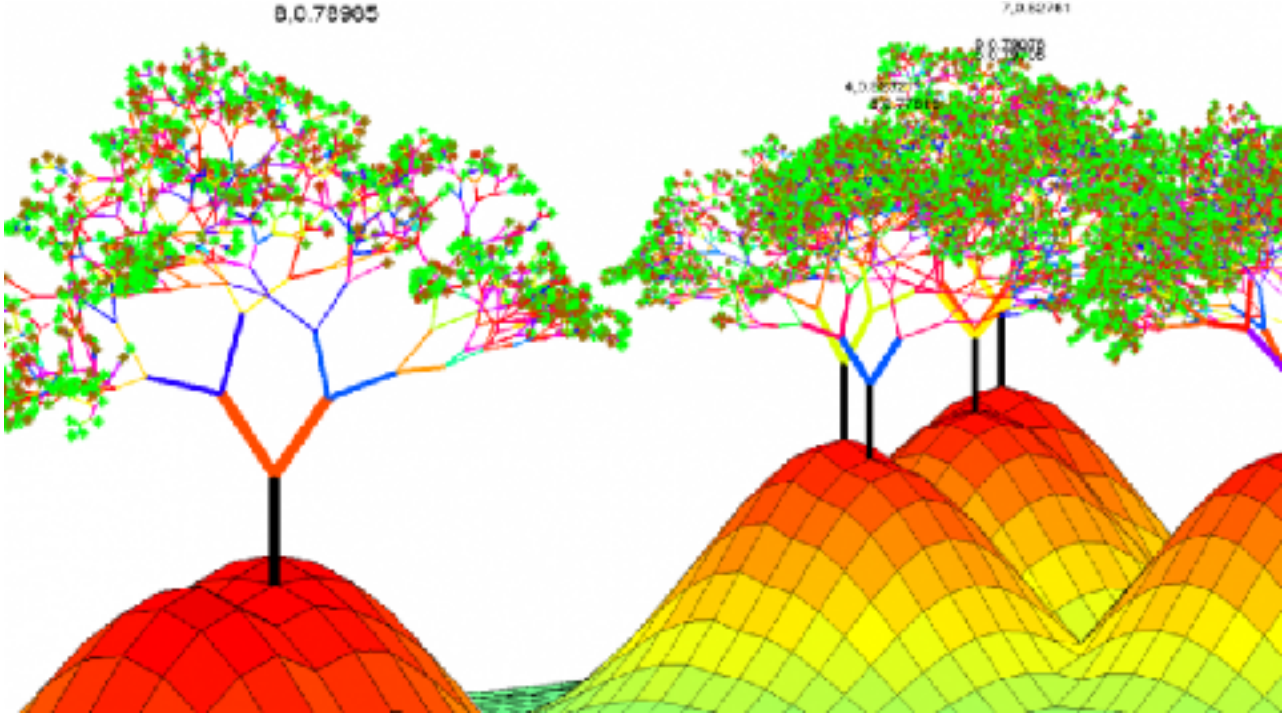
Most used algorithm- Bagging Technique

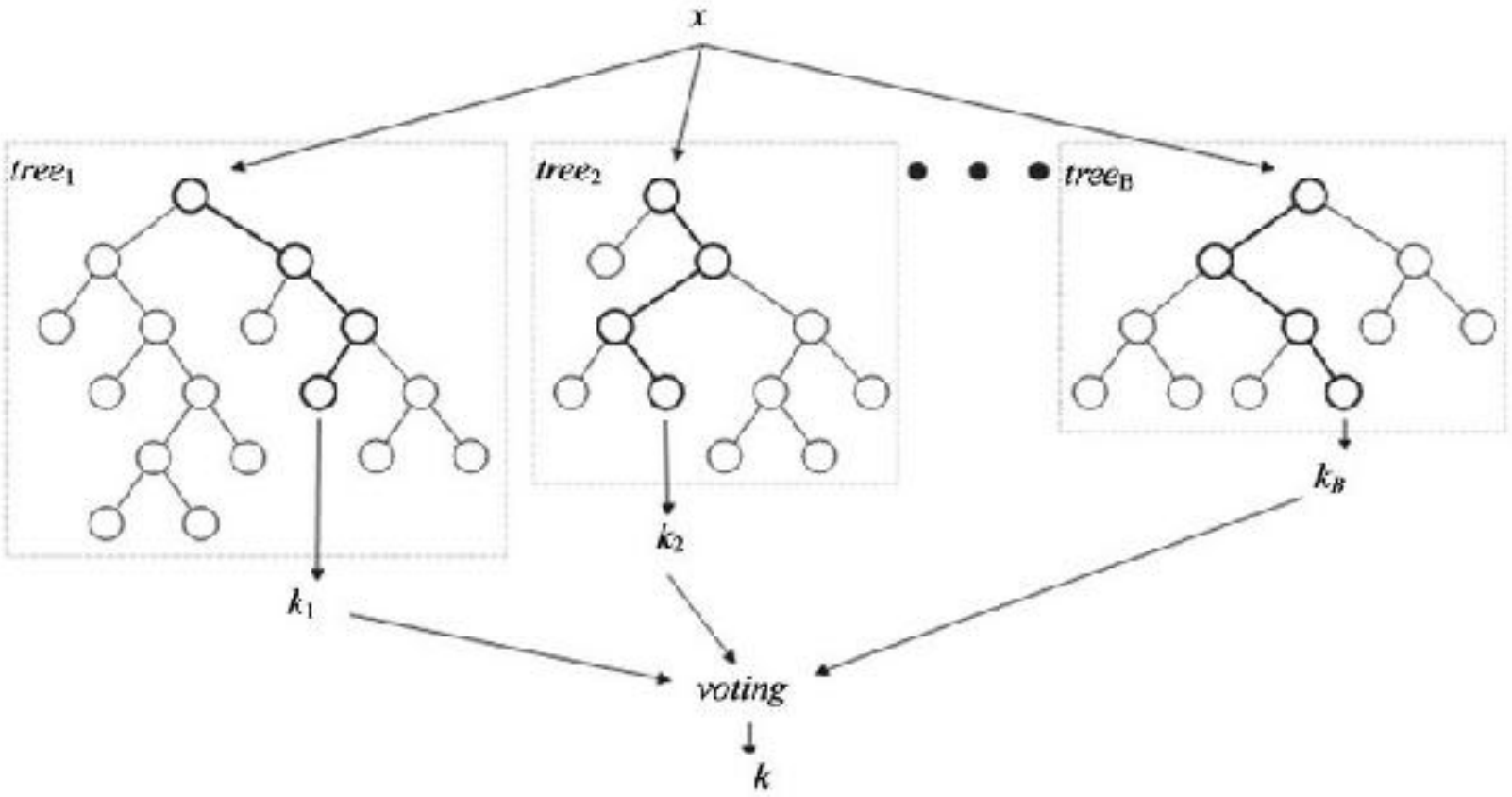
Bagging

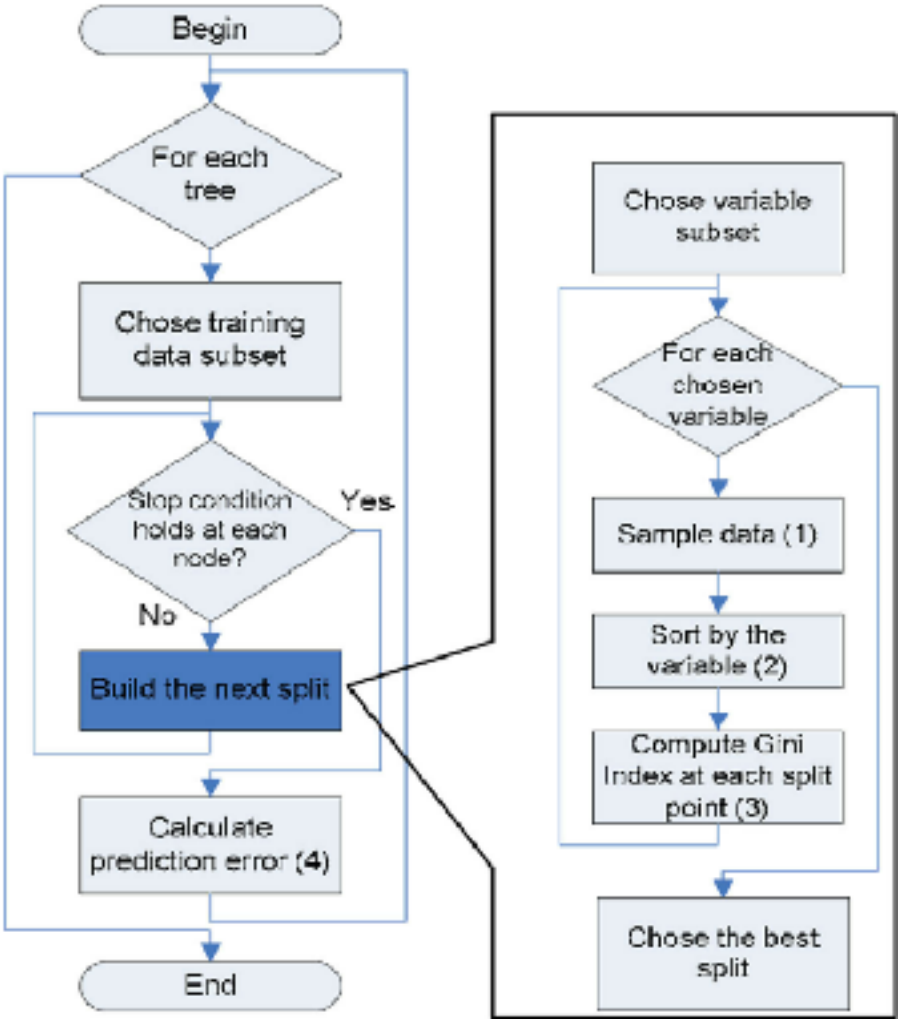
Bootstrap **agg**regat**ing**, also called **bagging**, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

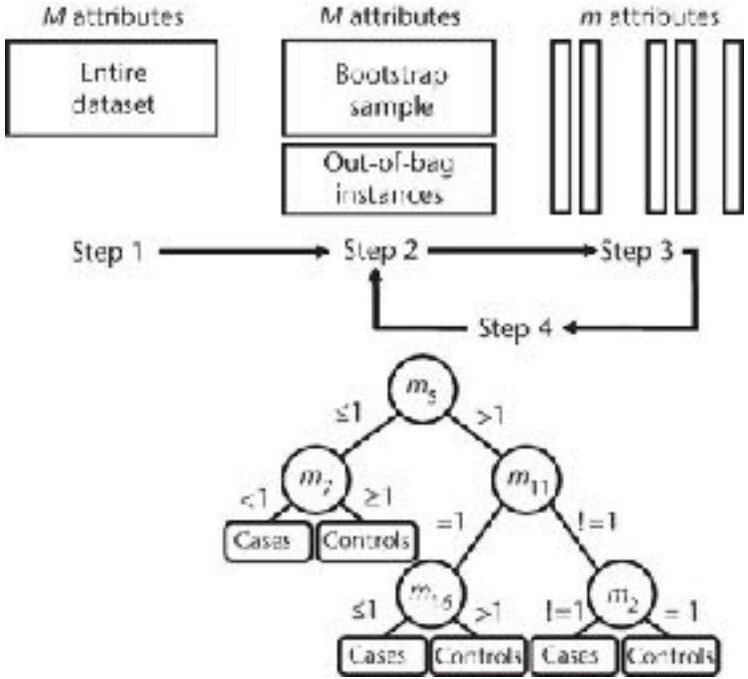
It also reduces variance and helps to avoid overfitting.

What insight can u get?




x
tree1
tree2
treeB
k1
k2
kB
voting
k


Begin
For each tree
Chose training data subset
Stop condition holds at each node?
Yes
No
Build the next split
Calculate prediction error (4)
End
Chose variable subset
For each chosen variable
Sample data (1)
Sort by the variable (2)
Compute Gini Index at each split point (3)
Chose the best split


M attributes
M attributes
m attributes
Entire dataset
Bootstrap sample
Out-of-bag instances
Step 1
Step 2
Step 3
Step 4
≤1
>1
<1
≥1
=1
!=1
Cases
Controls
≤1
>1
!=1
= 1
Cases
Controls
Cases
Controls

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.

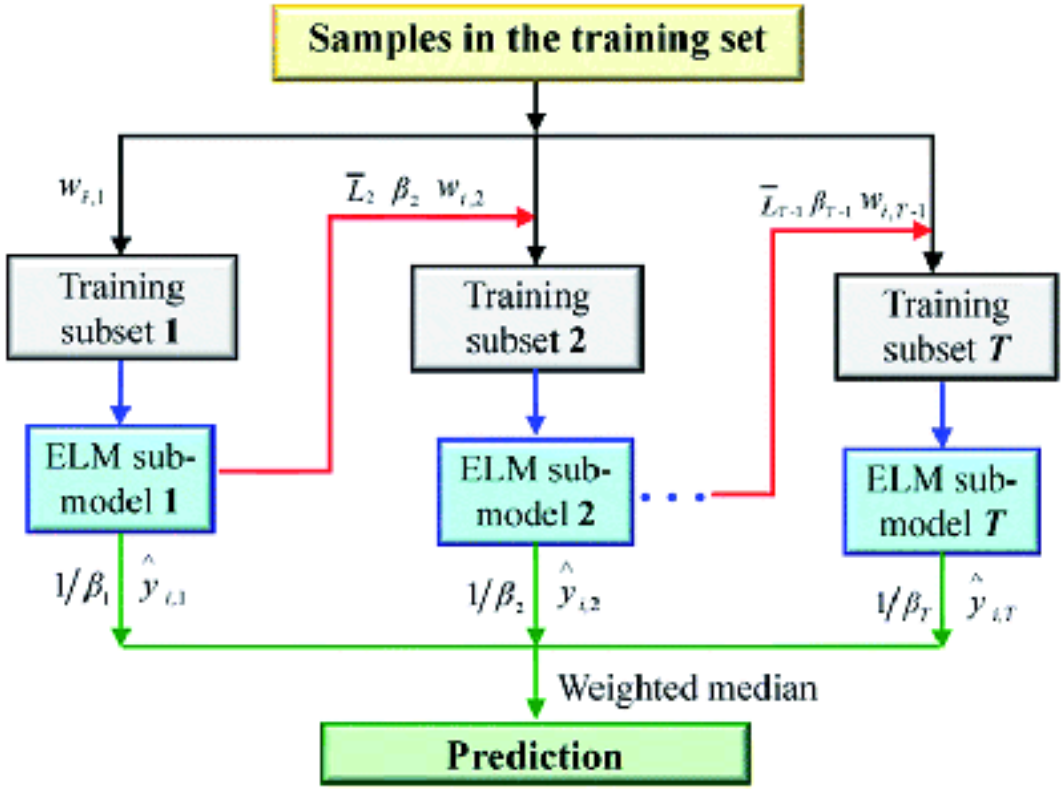
For many data sets, it produces a highly accurate classifier.

It runs efficiently on large databases.

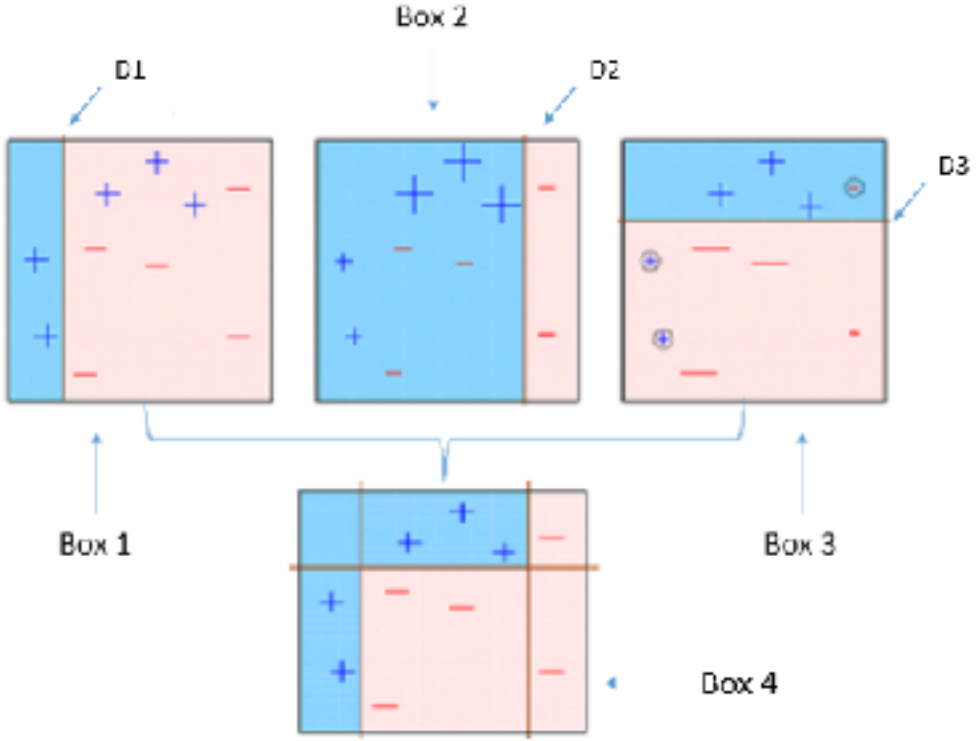
Boosting

Competition winning algorithm

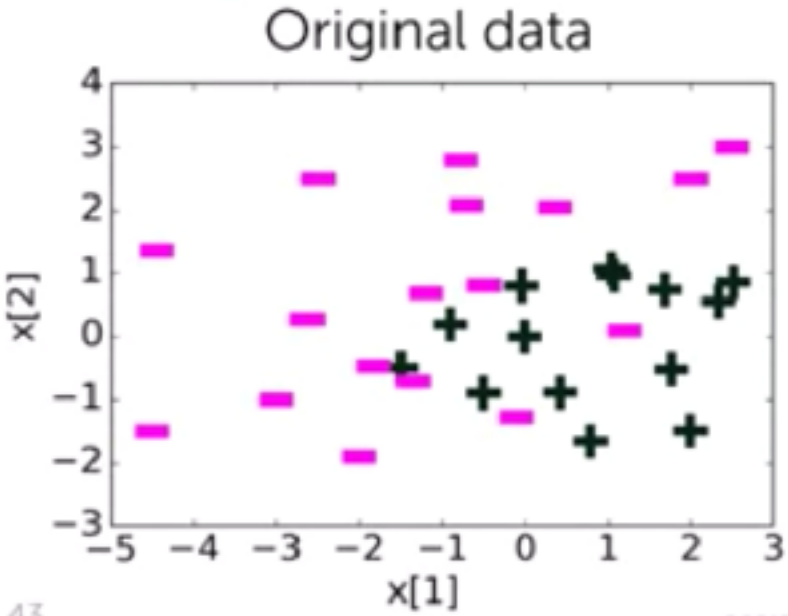
Boosting

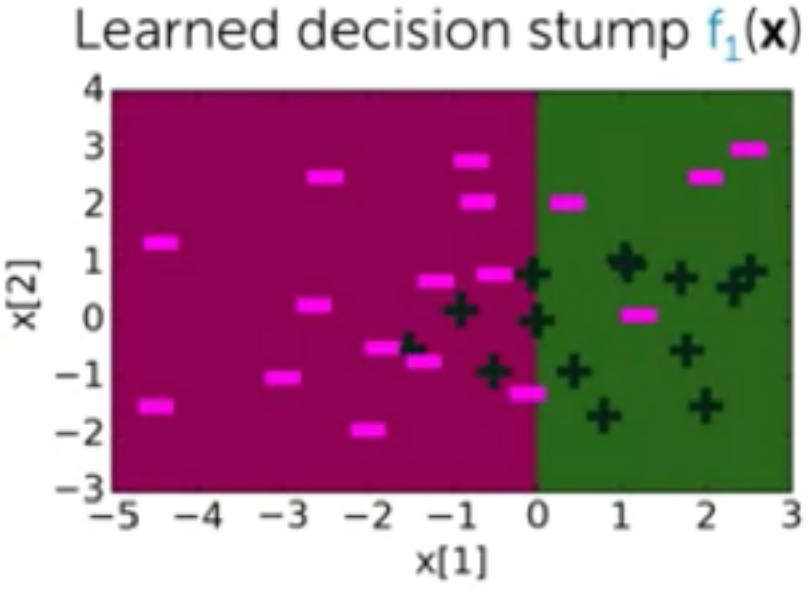

Samples in the training set
Training subset 1
Training subset 2
Training subset T
ELM sub-model 1
ELM sub-model 2
ELM sub-model T
Weighted median
Prediction

AdaBoost



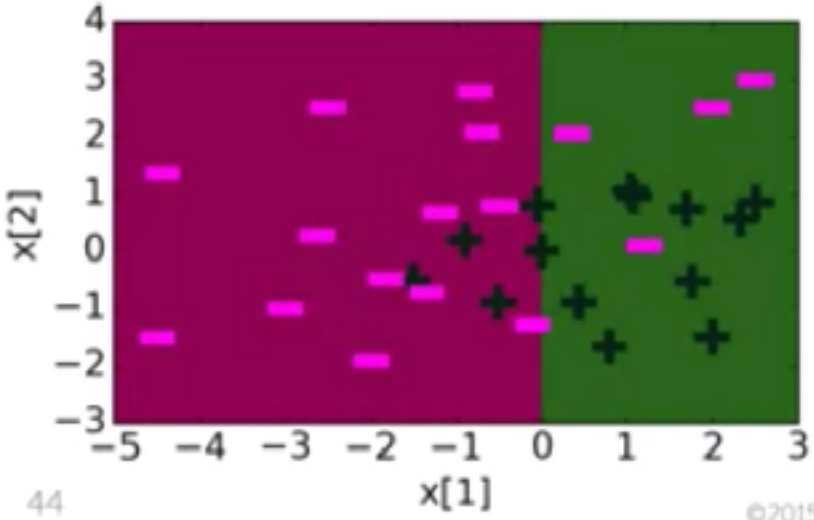
Iteration = 1



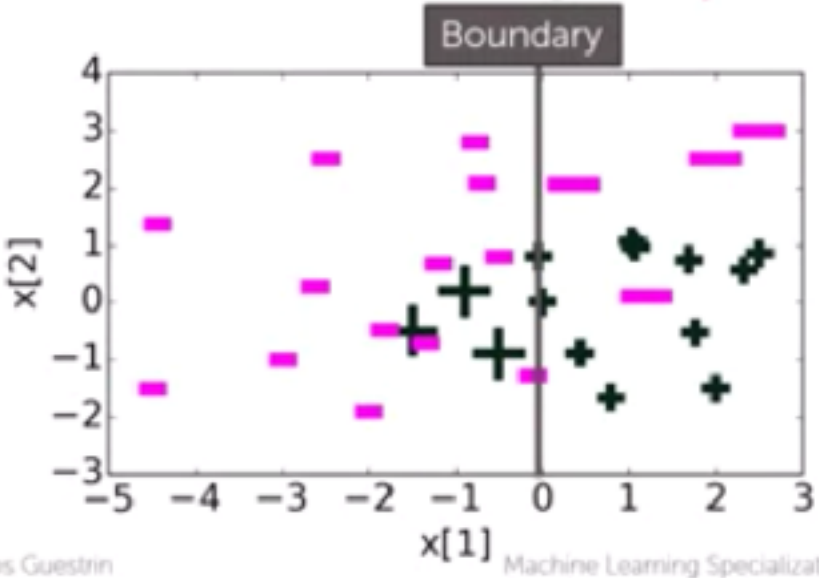


Updated Weights

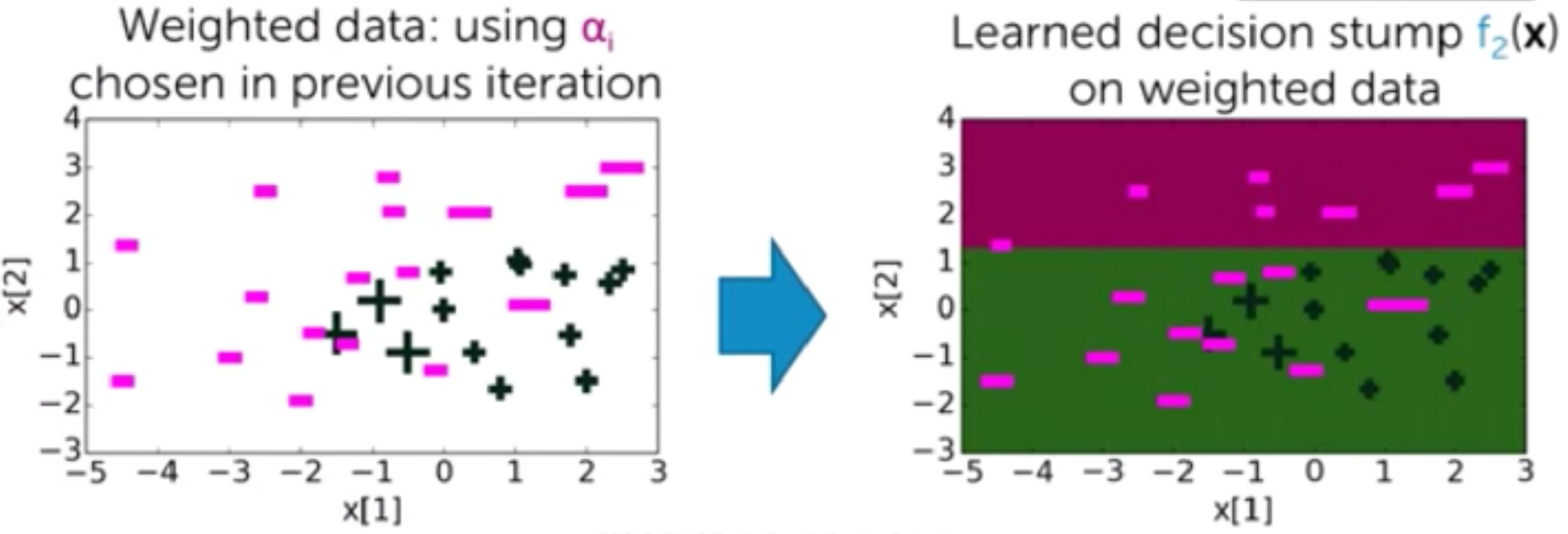
Learned decision stump $f_1(\mathbf{x})$



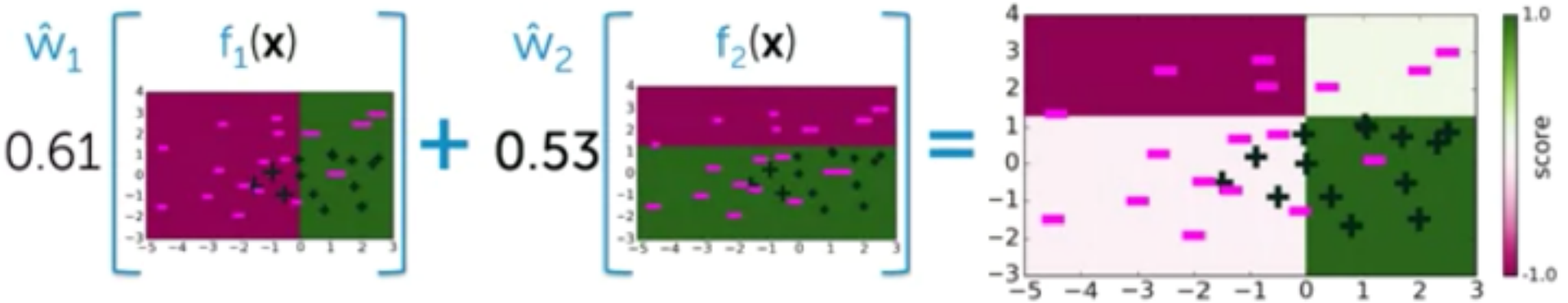
New data weights α_i



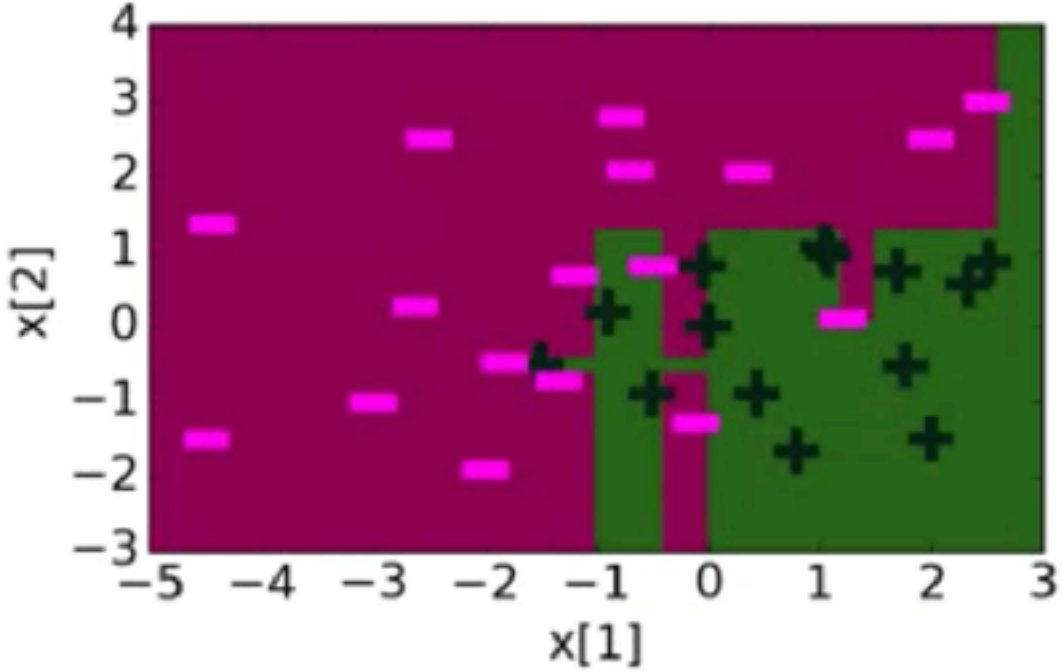
Iteration = 2



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: $\alpha_i = 1/N$

- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i
  - Normalize weights α_i

- Final model predicts by:

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

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

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

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

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,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:

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

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:

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

weighted_error = 0.3

weighted_error = 0.4

Finding best next decision stump

Consider splitting on each feature:

Finding best next decision stump

Consider splitting on each feature:

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

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,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:

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

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	ŷ
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

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

Credit	Income	y	ŷ	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

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

Credit	Income	y	ŷ	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

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

Credit	Income	y	ŷ	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 esti
mator if it ac# cepts 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.

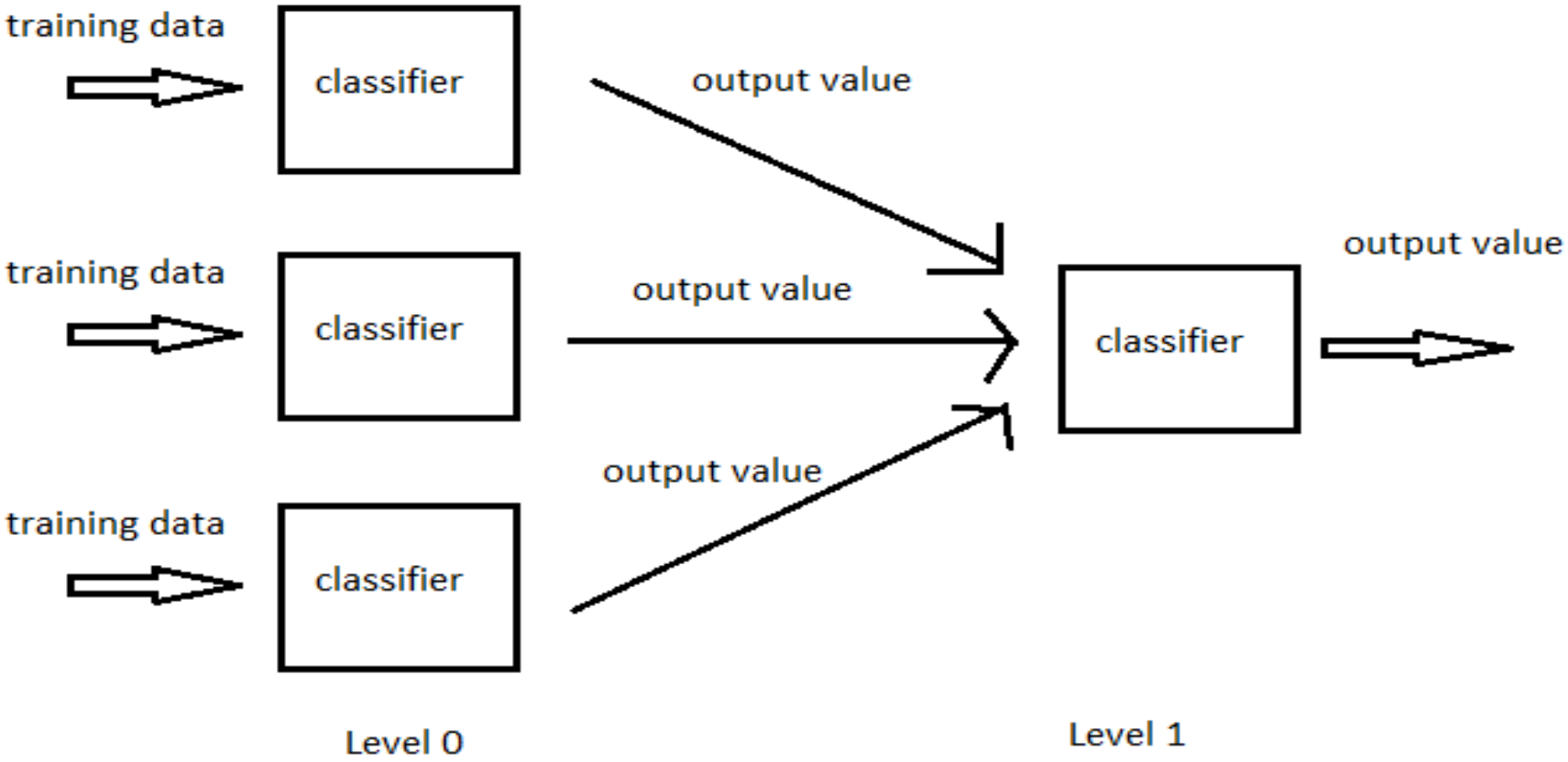
Hard to implement in real time platform.

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(S_train, y_train)

# Predict
y_pred = model.predict(S_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: [0.93103448]
    ----
    MEAN:   [0.94166667]

model 2: [XGBClassifier]
    fold 0: [0.83870968]
    fold 1: [0.93333333]
    fold 2: [1.00000000]
    fold 3: [0.93103448]
    ----
    MEAN:   [0.92500000]

Final prediction score: [0.95666667]
```

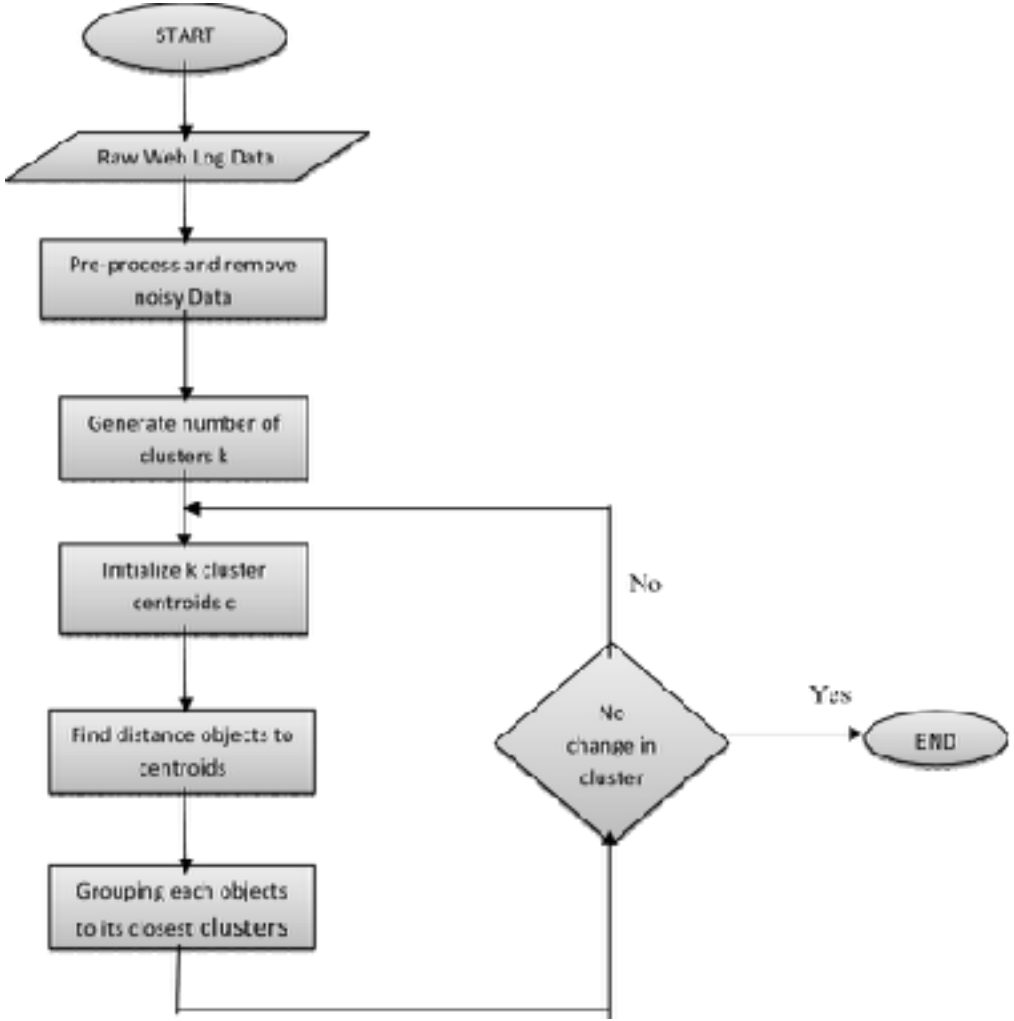
K-Means Clustering

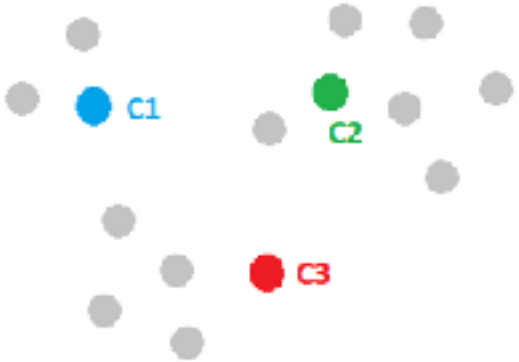
Step one: Initialize cluster centers

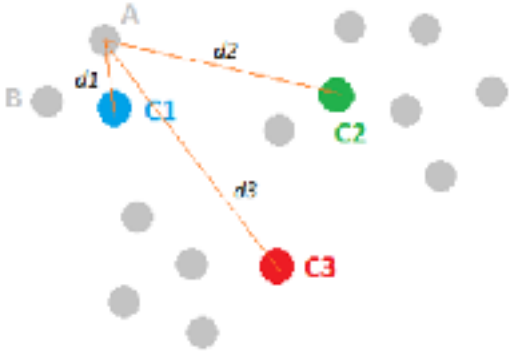
Step two: Assign observations to the closest cluster center

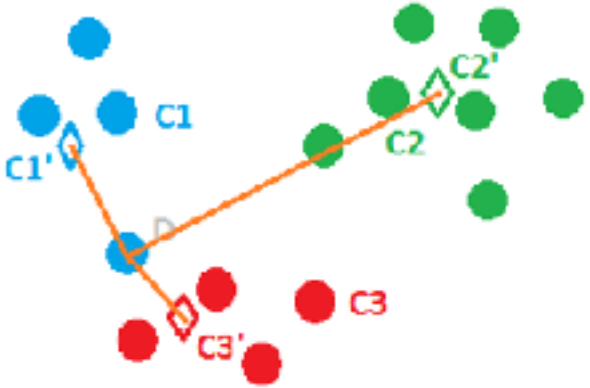
Step three: Revise cluster centers as mean of assigned observations

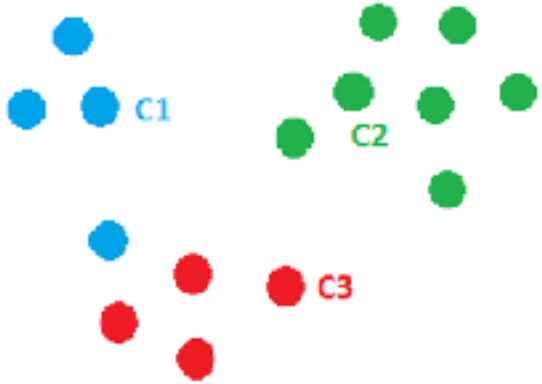
Step four: Repeat step 2 and step 3 until convergence

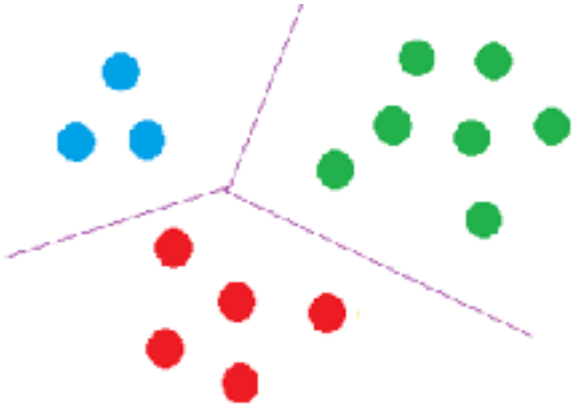



C1
C2
C3

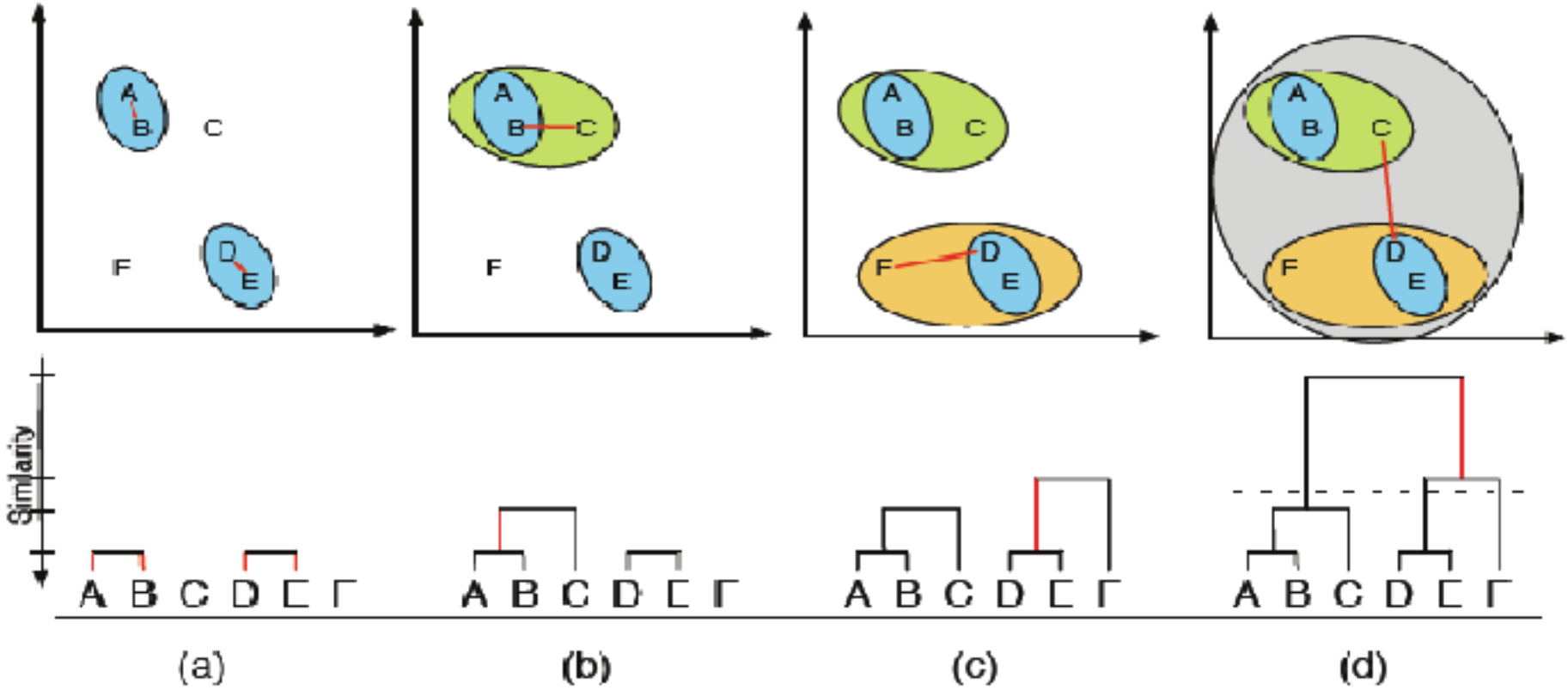

A
B
d1
d2
d3
C1
C2
C3


C1
C1'
C2
C2'
D
C3
C3'


C1
C2
C3



Example: Hierarchical Agglomerative Clustering



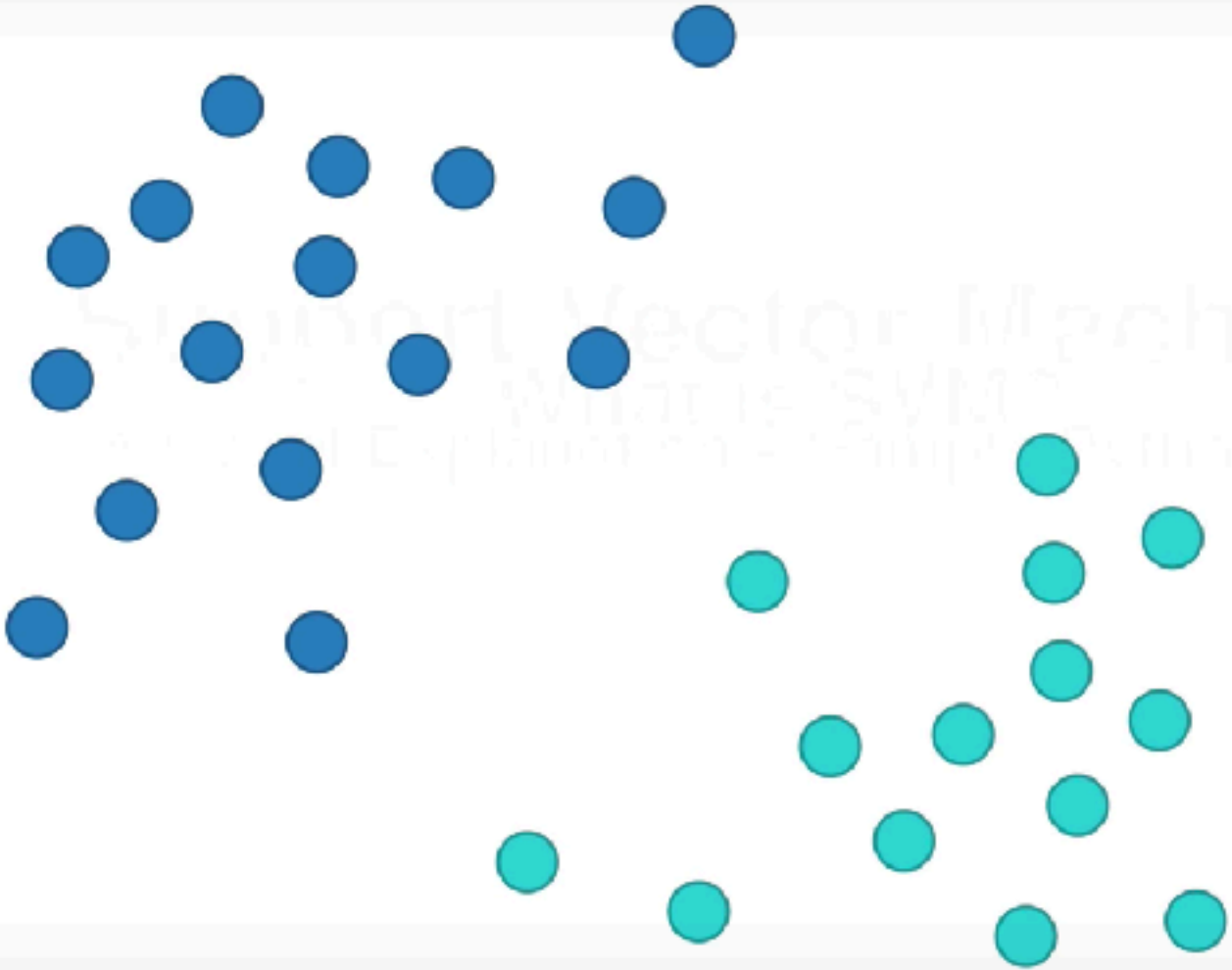
Support Vector Machine

Agenda

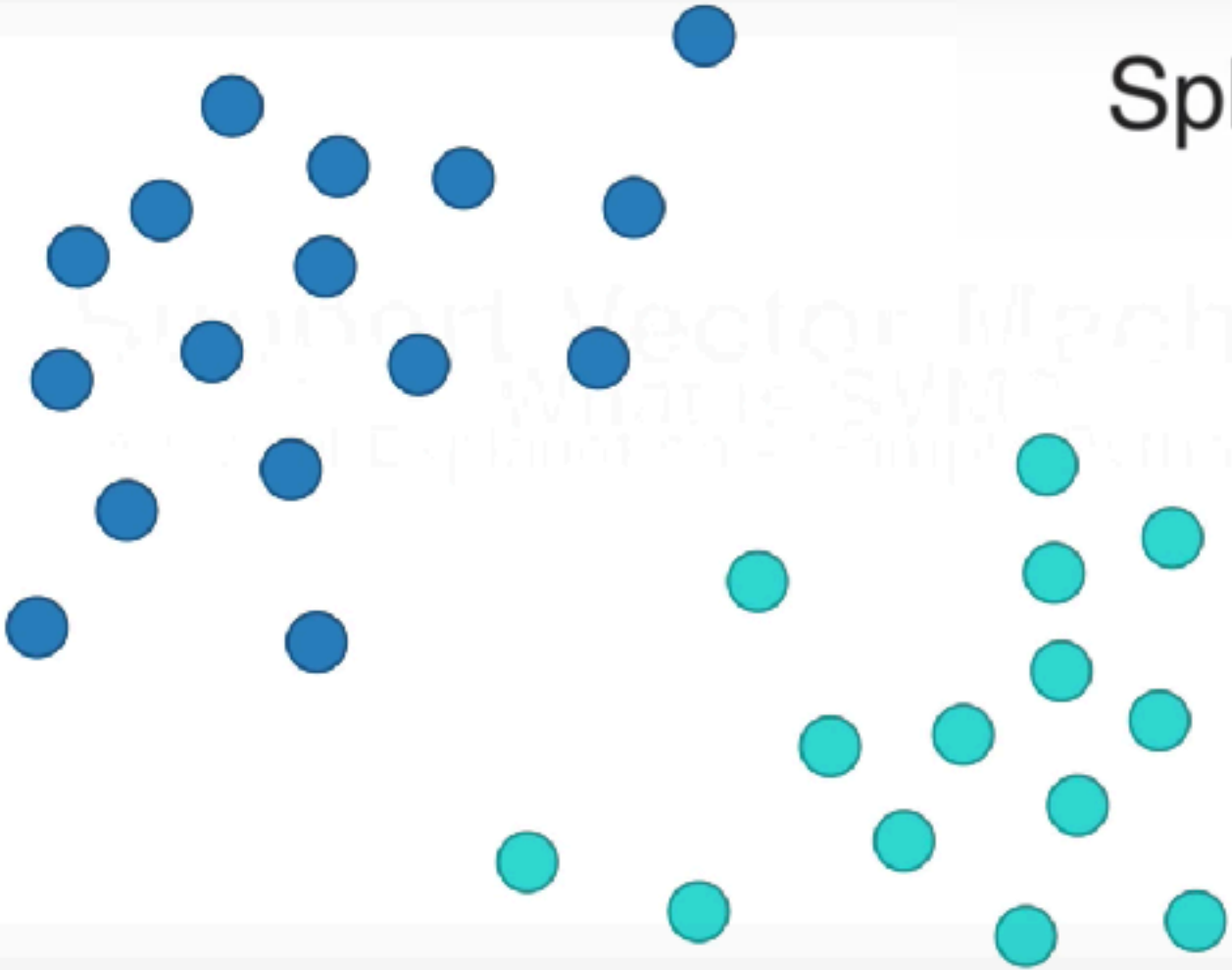
1. **SVM Basics**
   - Visual introduction
   - Example in Python
2. **Additional Complexities**
   - Higher dimensions
   - C parameter
   - Multiple classes
   - Kernel trick
3. **Closing Remarks**
   - Pros and cons
   - Other techniques

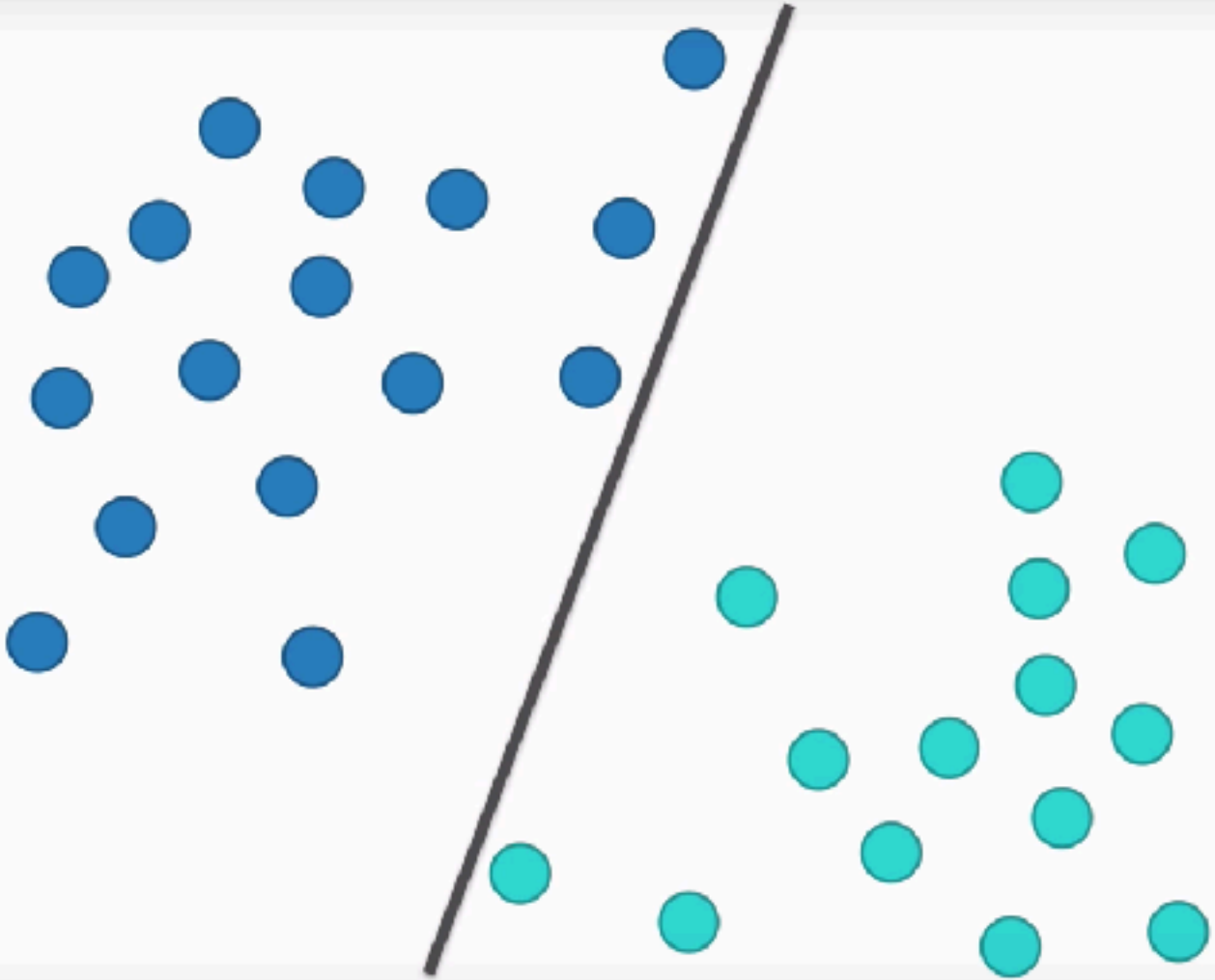
What is SVM?

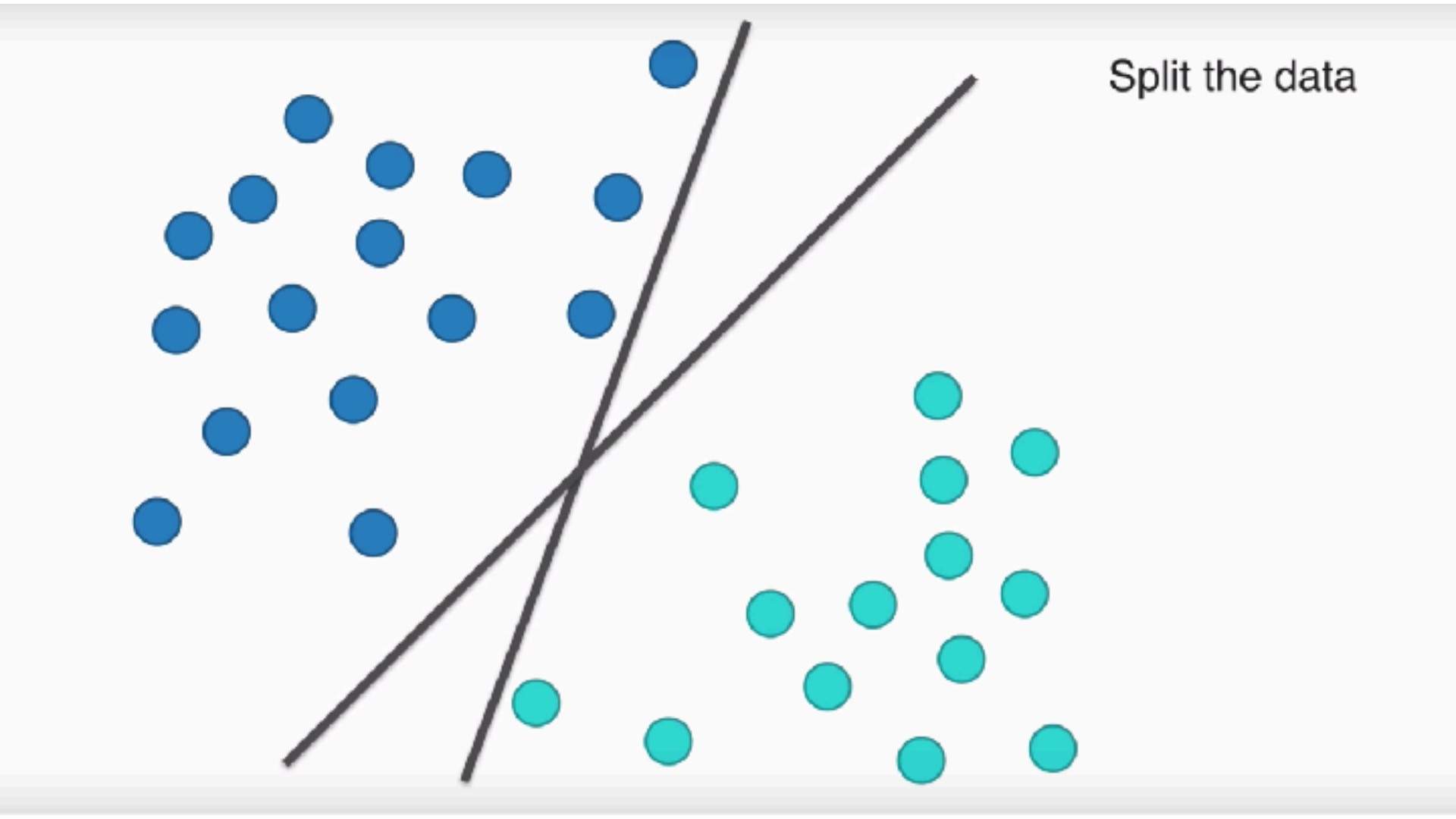
It’s a classification technique.

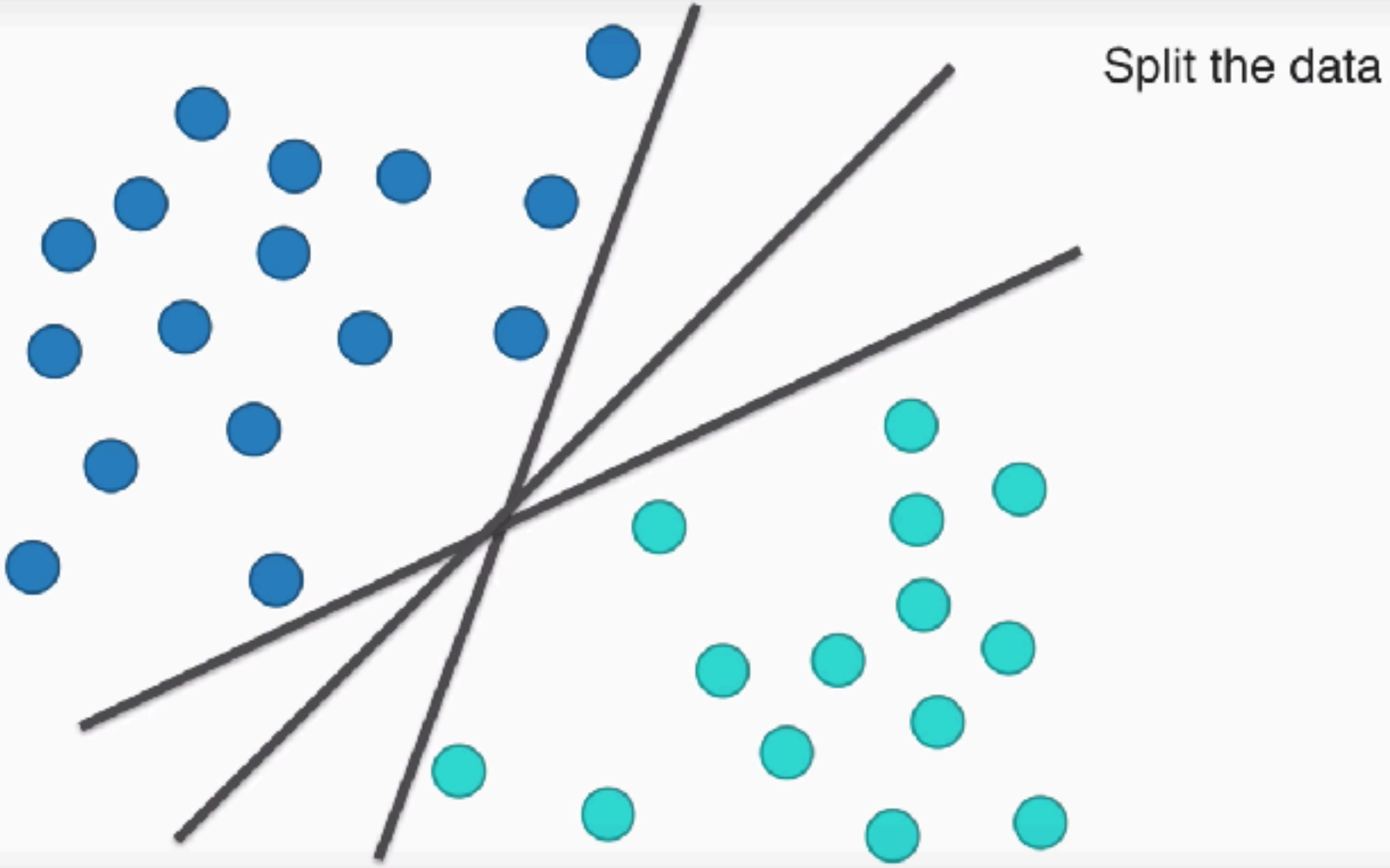


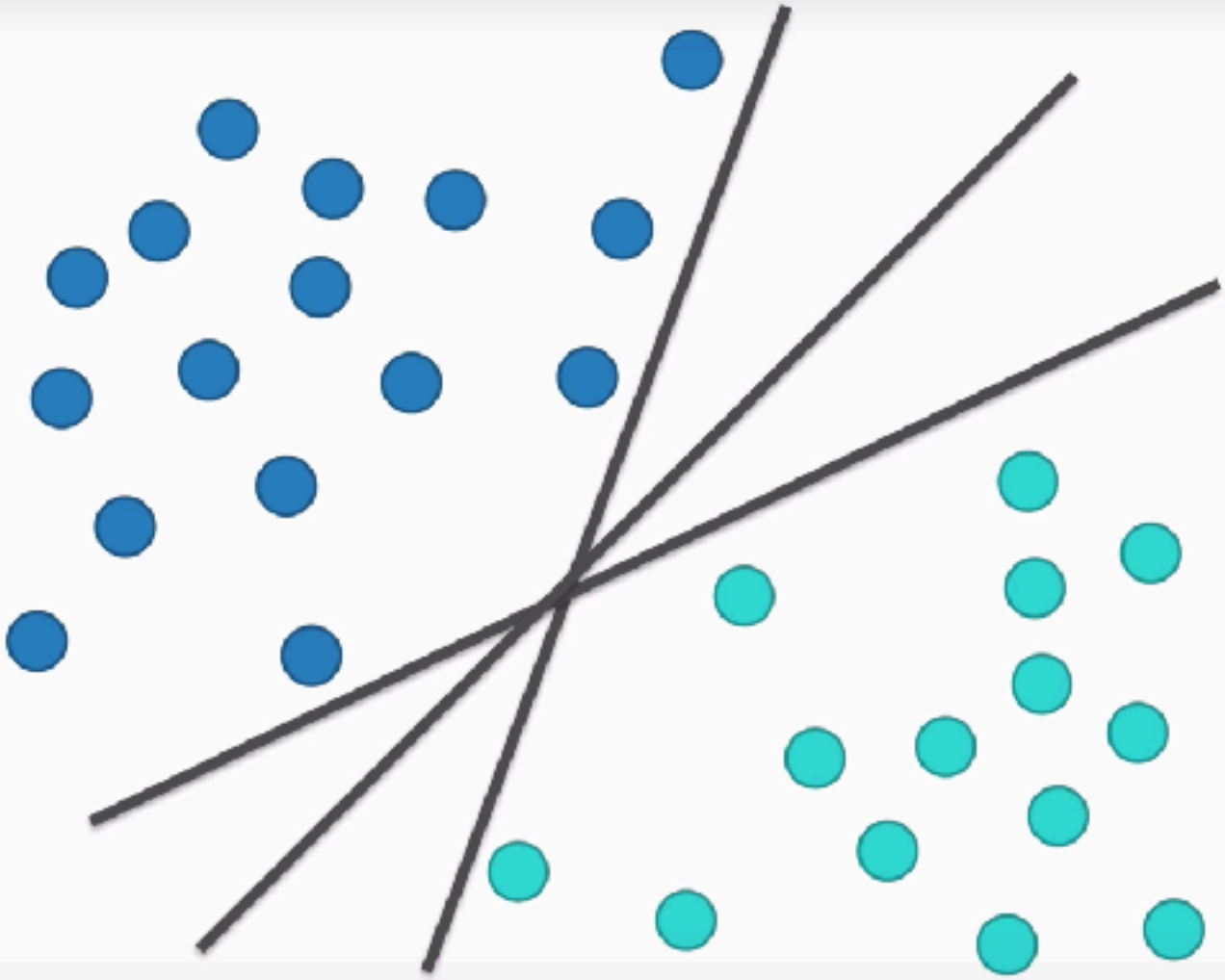
Split the data




Split the data

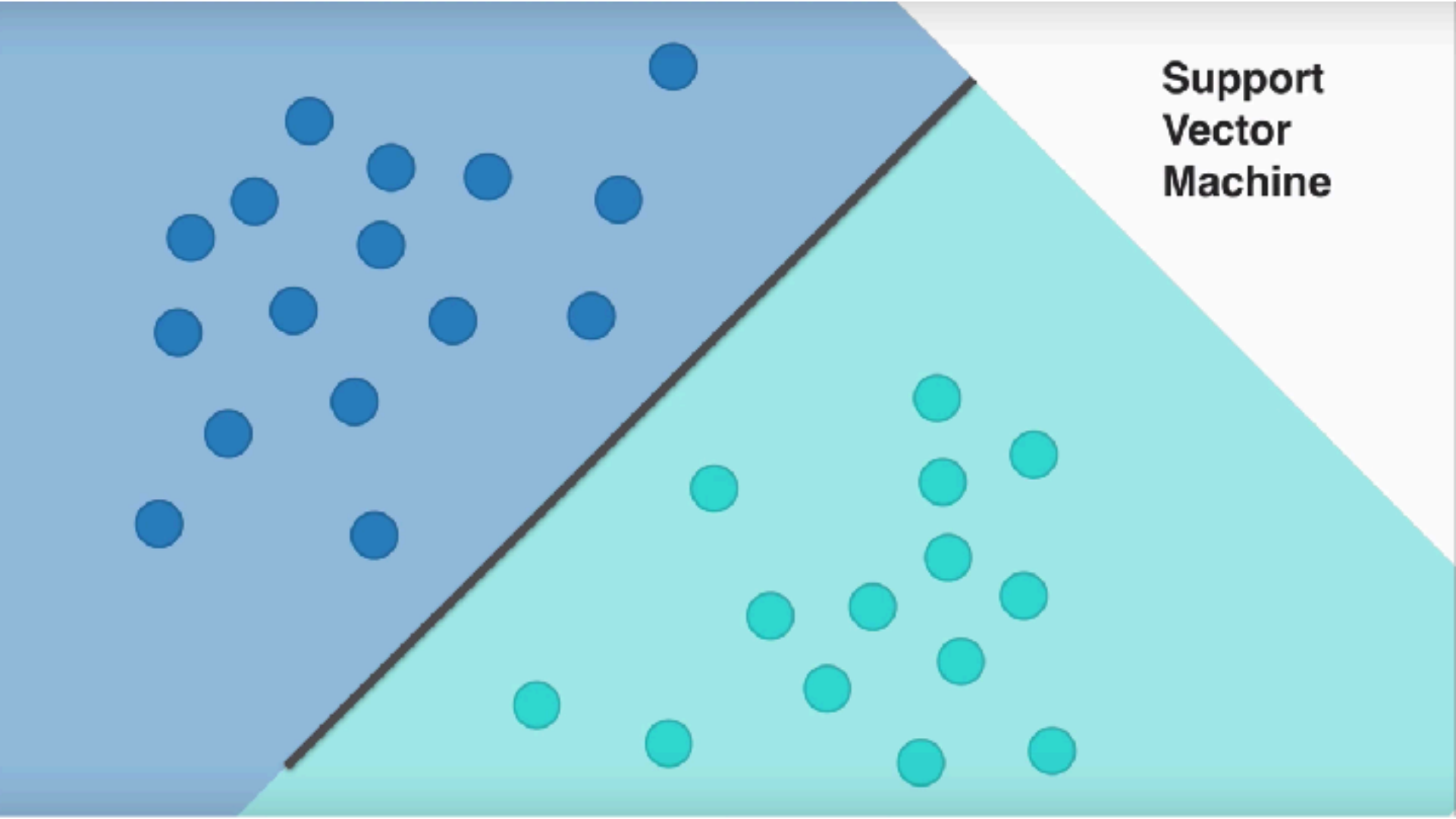

Split the data

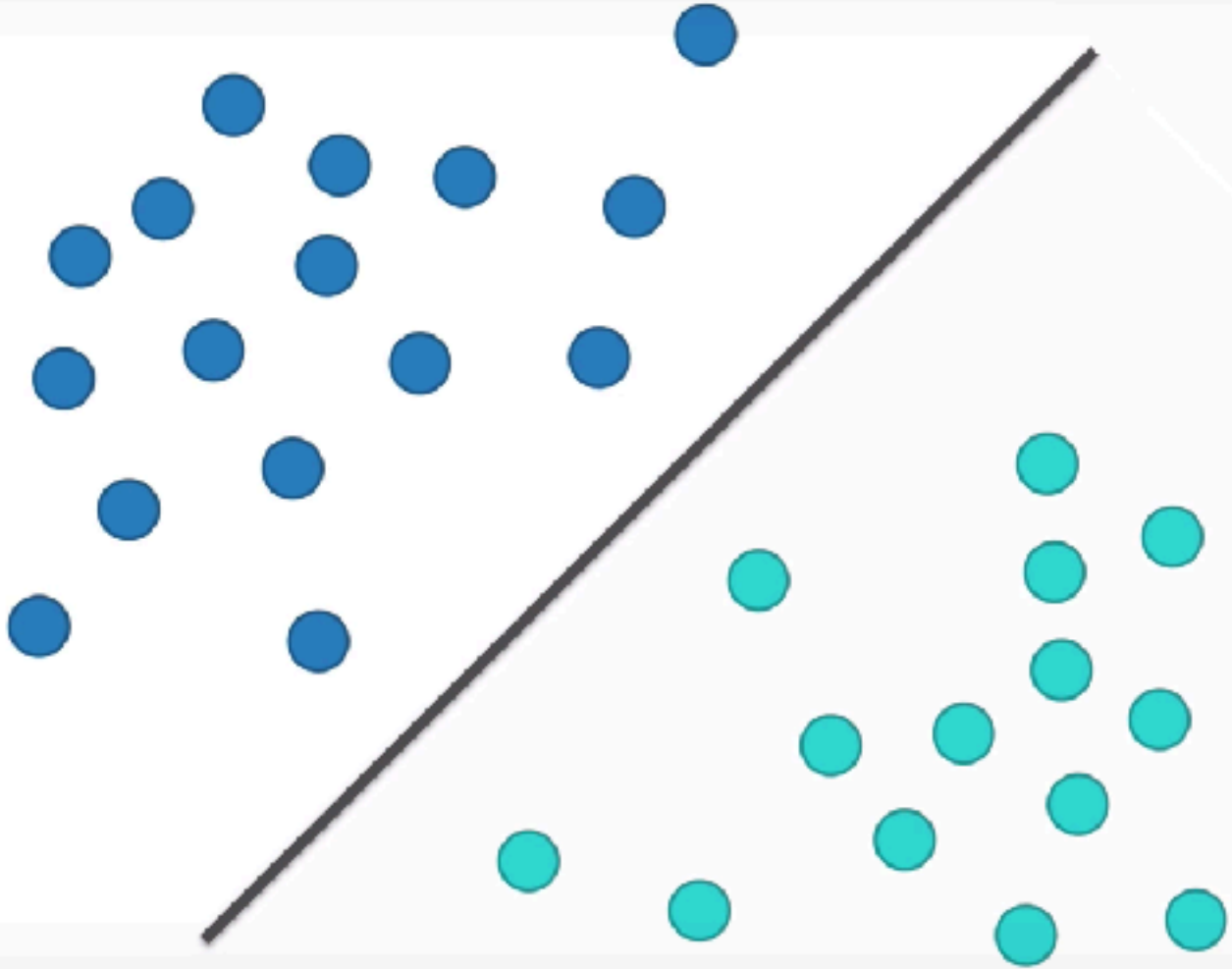

Split the data



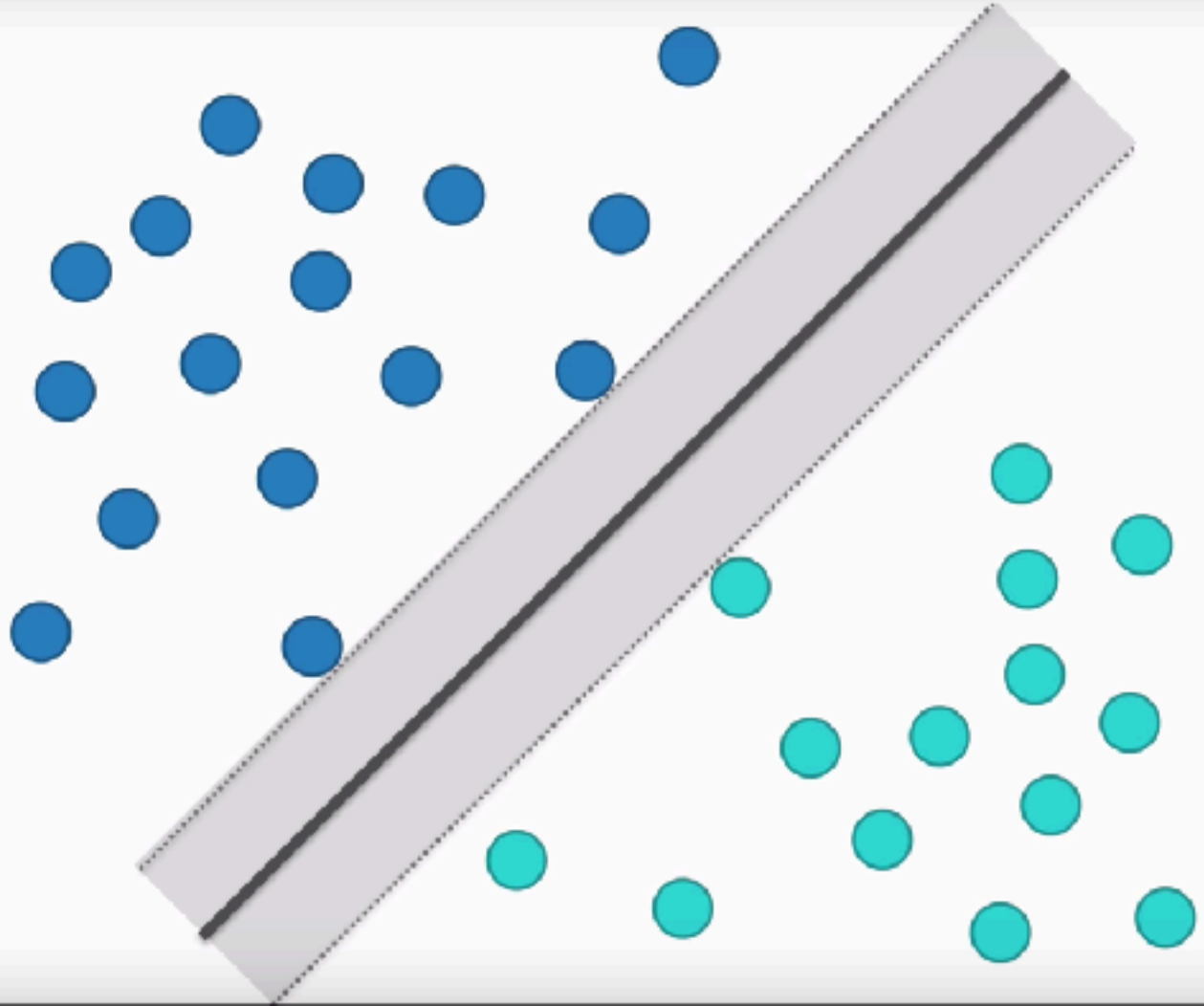
Split the data
in the best possible way


Split the data
in the best
possible way

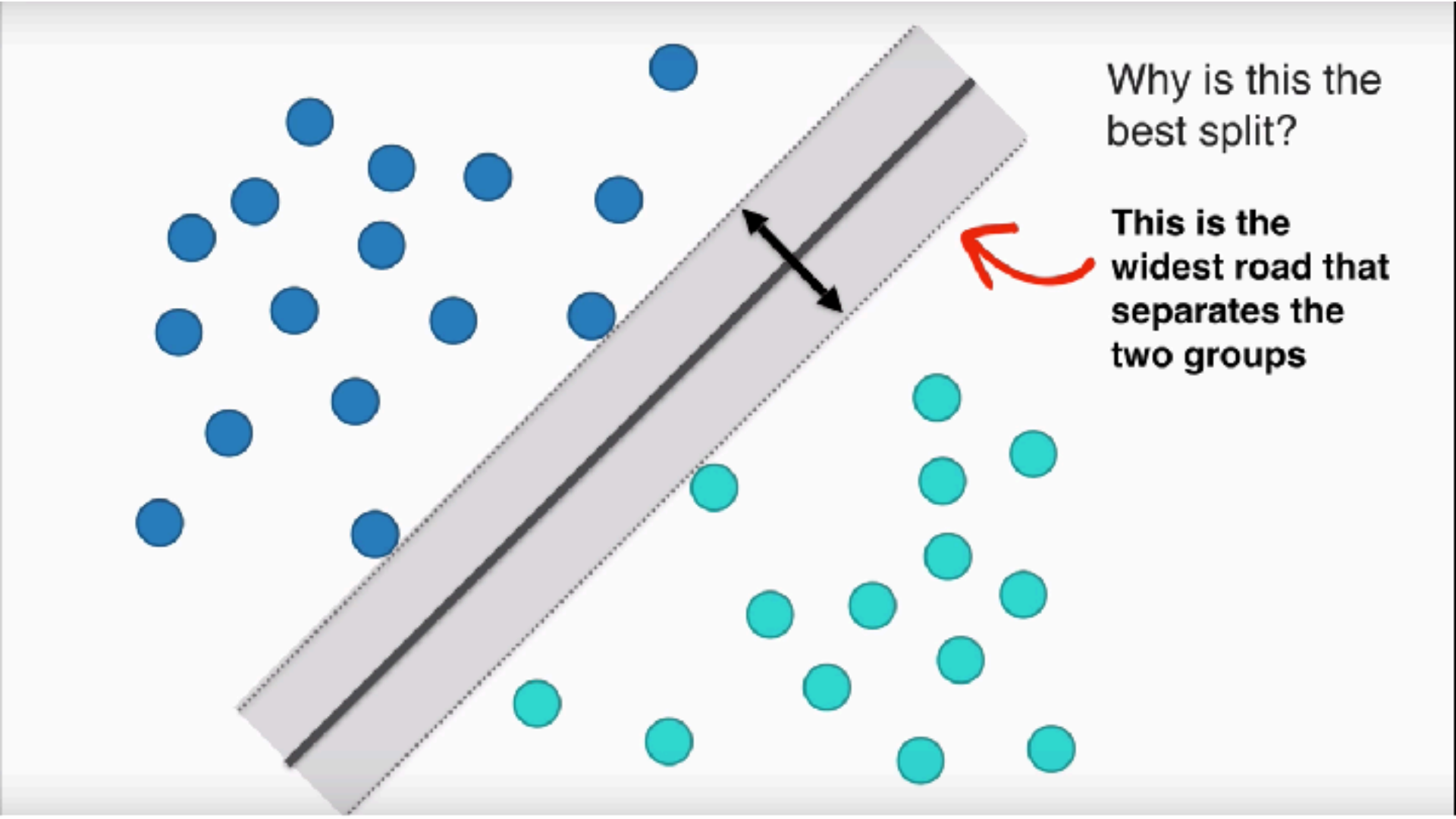

Support
Vector
Machine

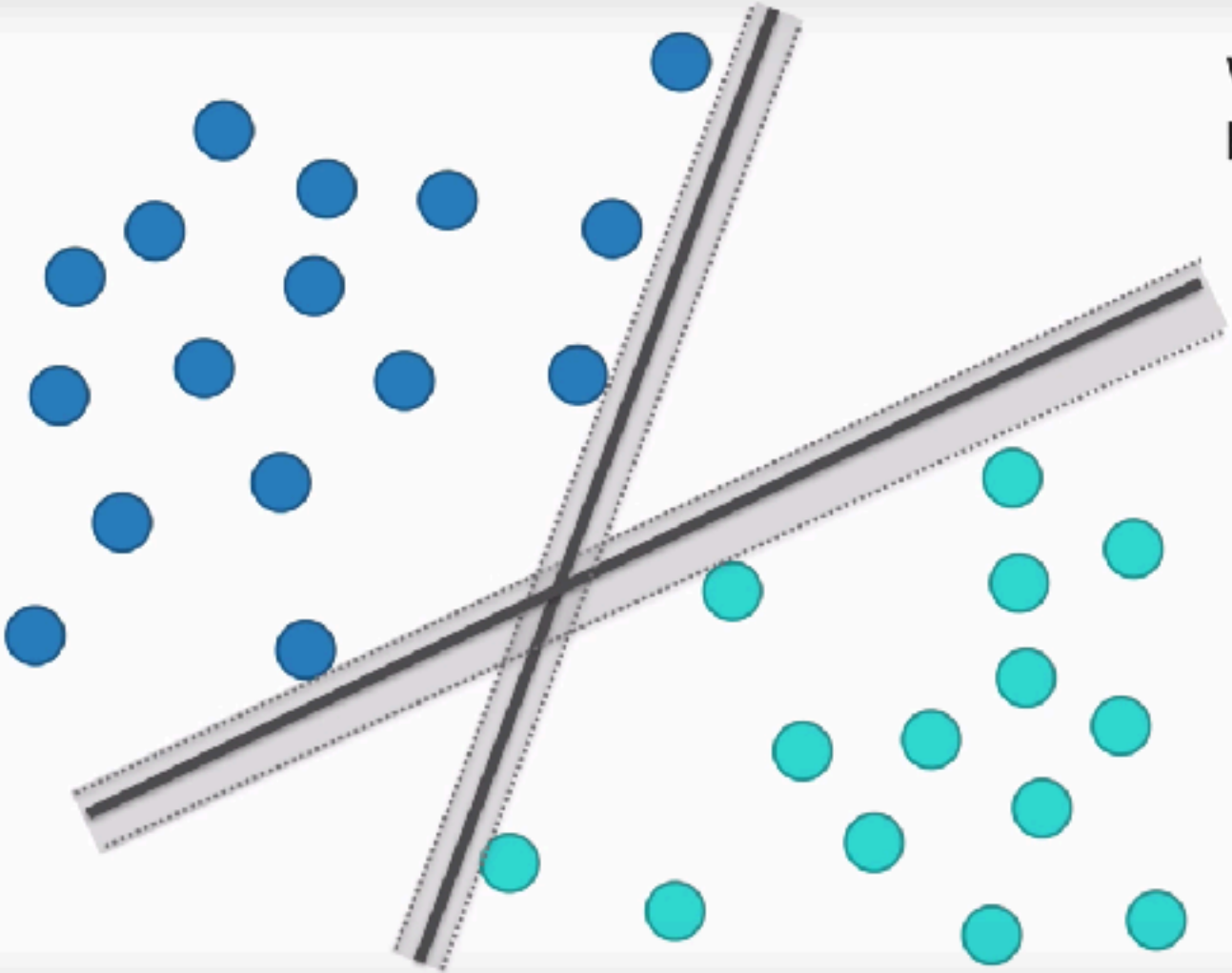


Why is this the best split?

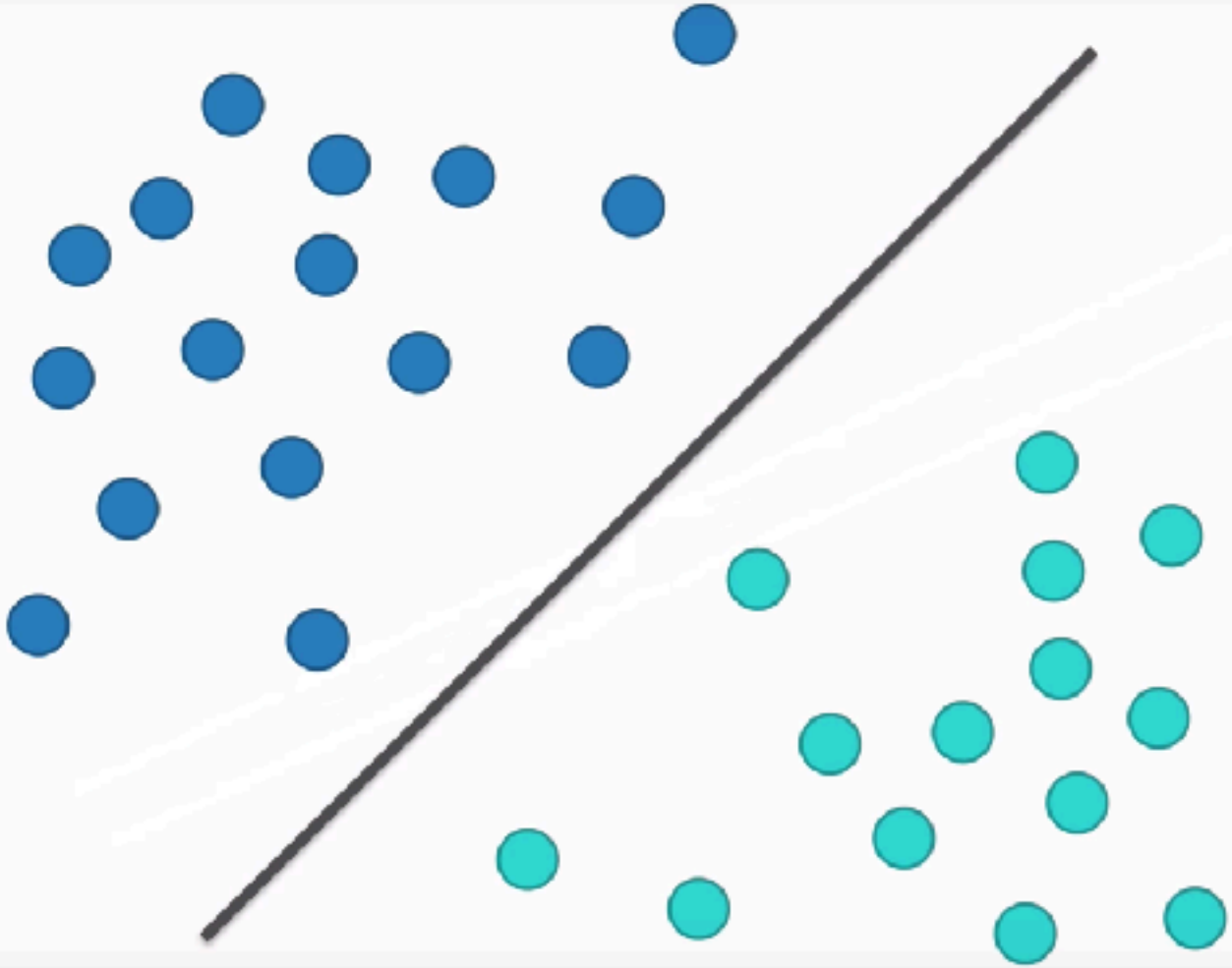


Why is this the best split?

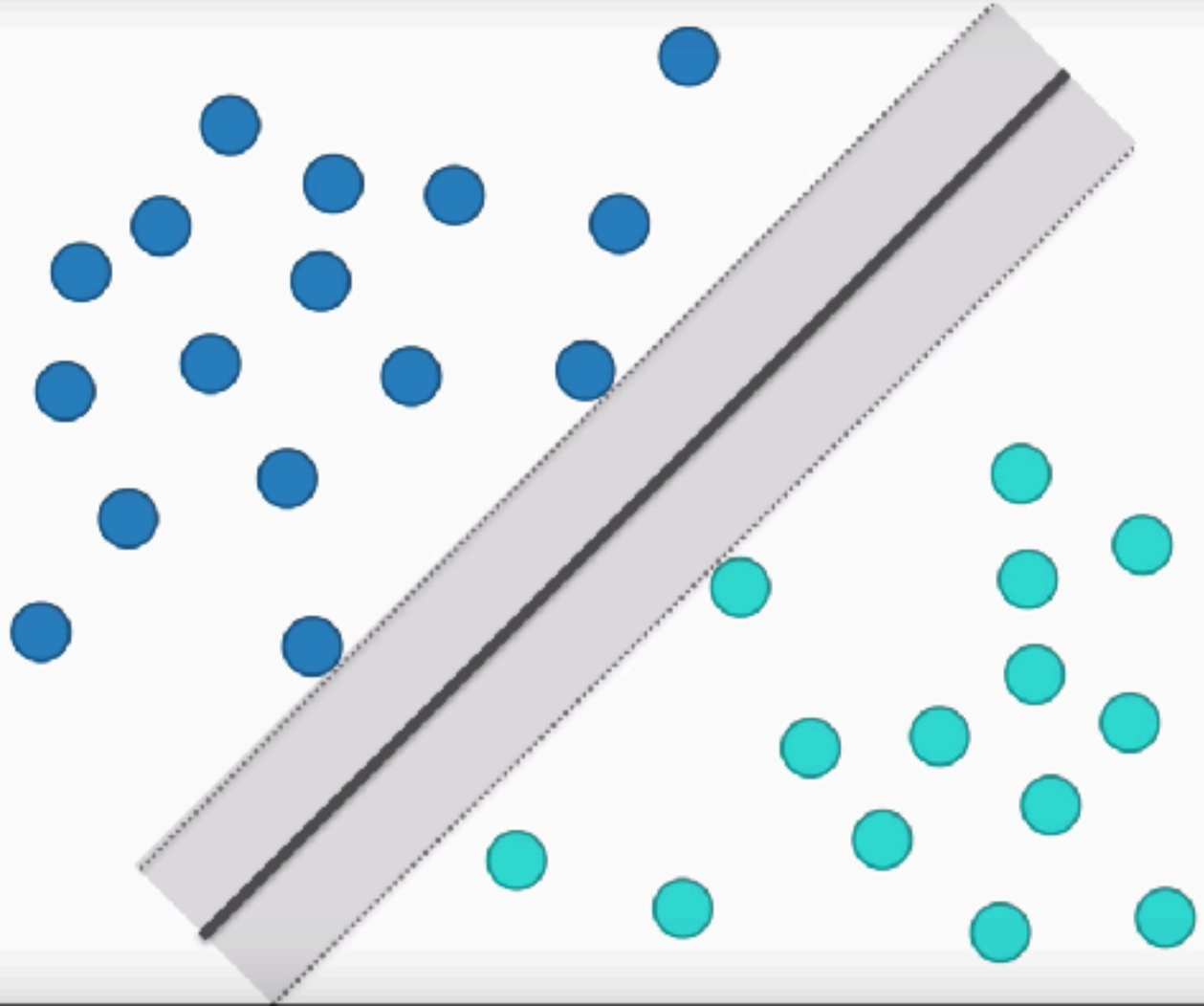

Why is this the best split?
This is the widest road that separates the two groups



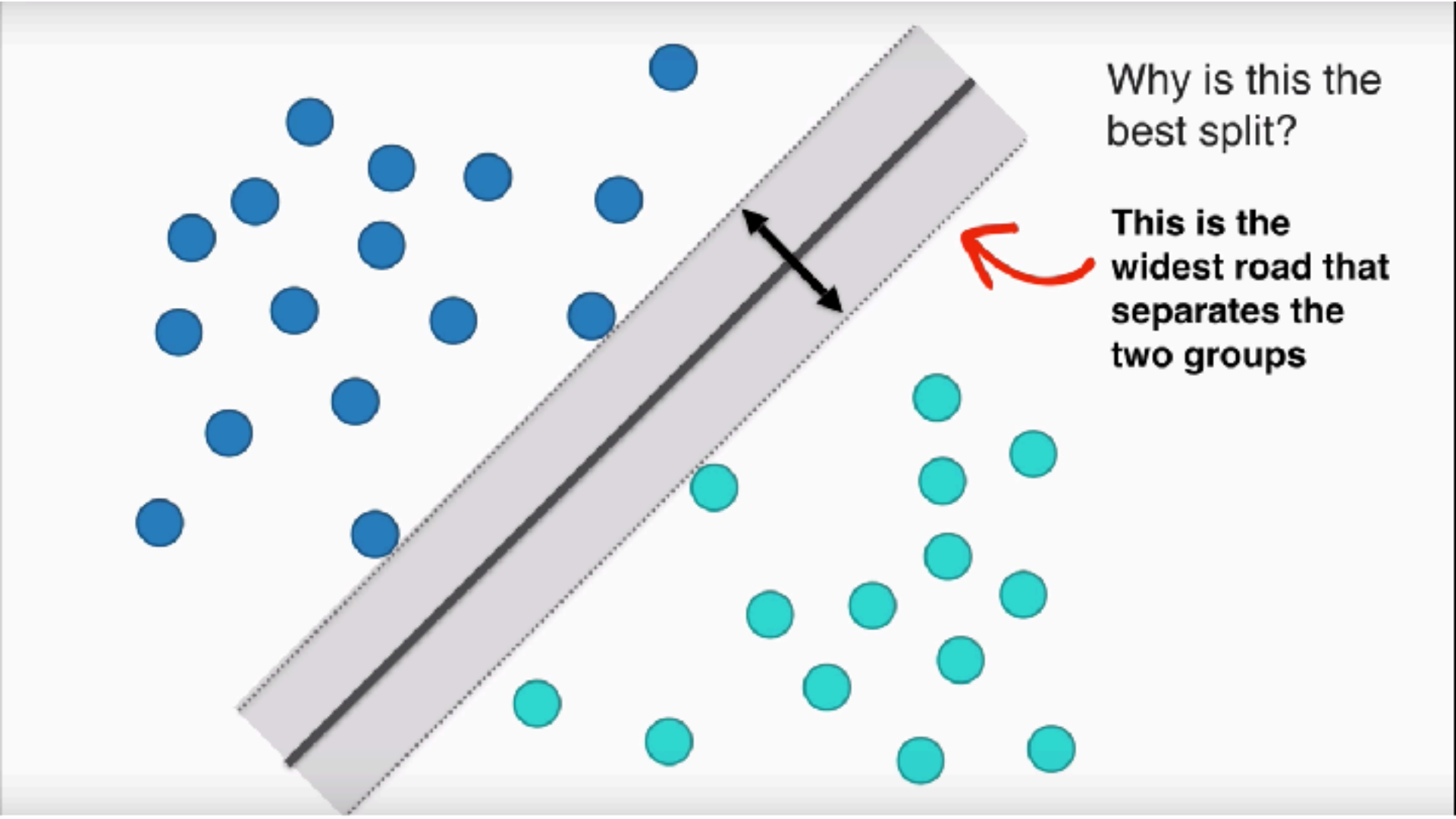
Why is this the best split?

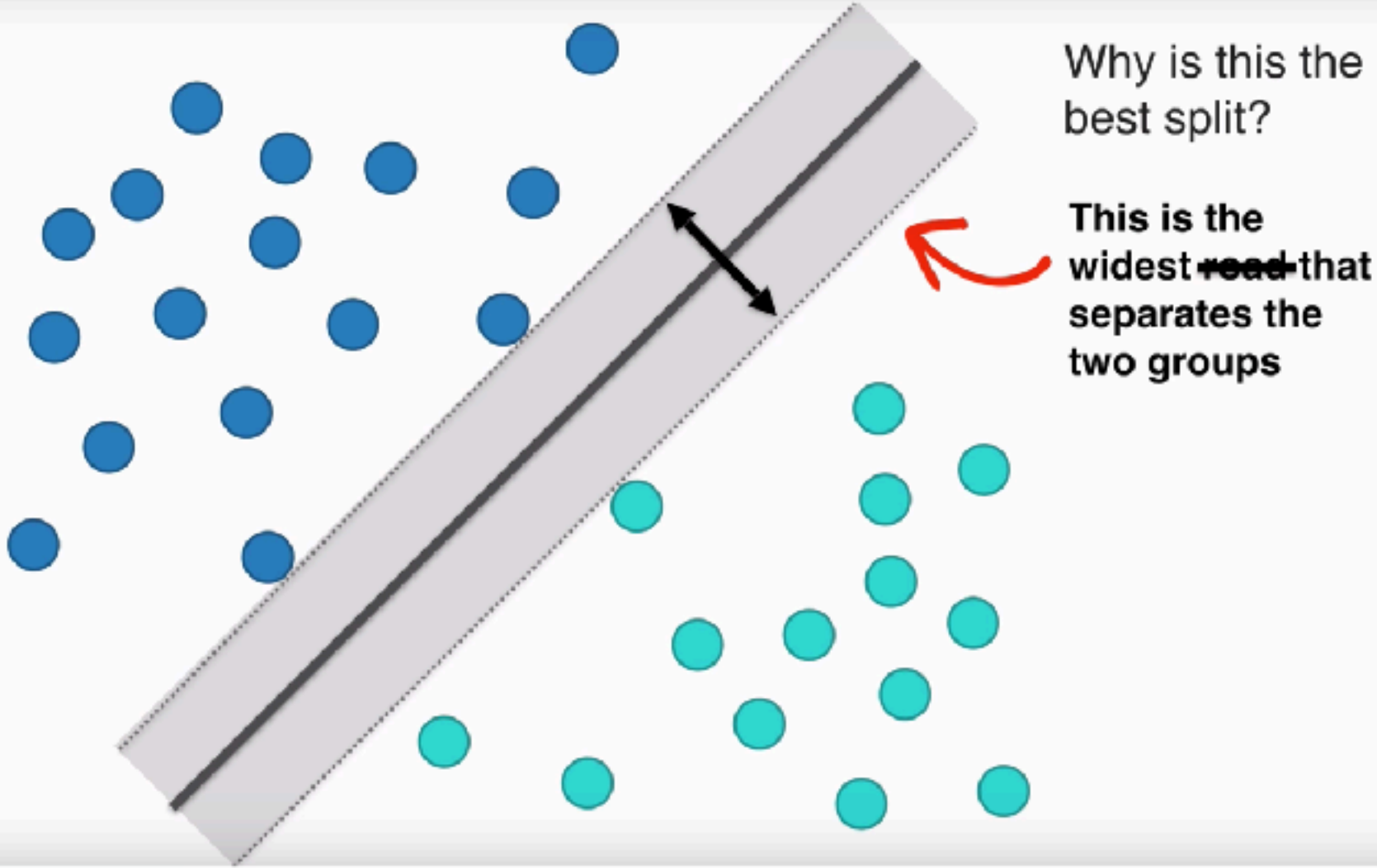


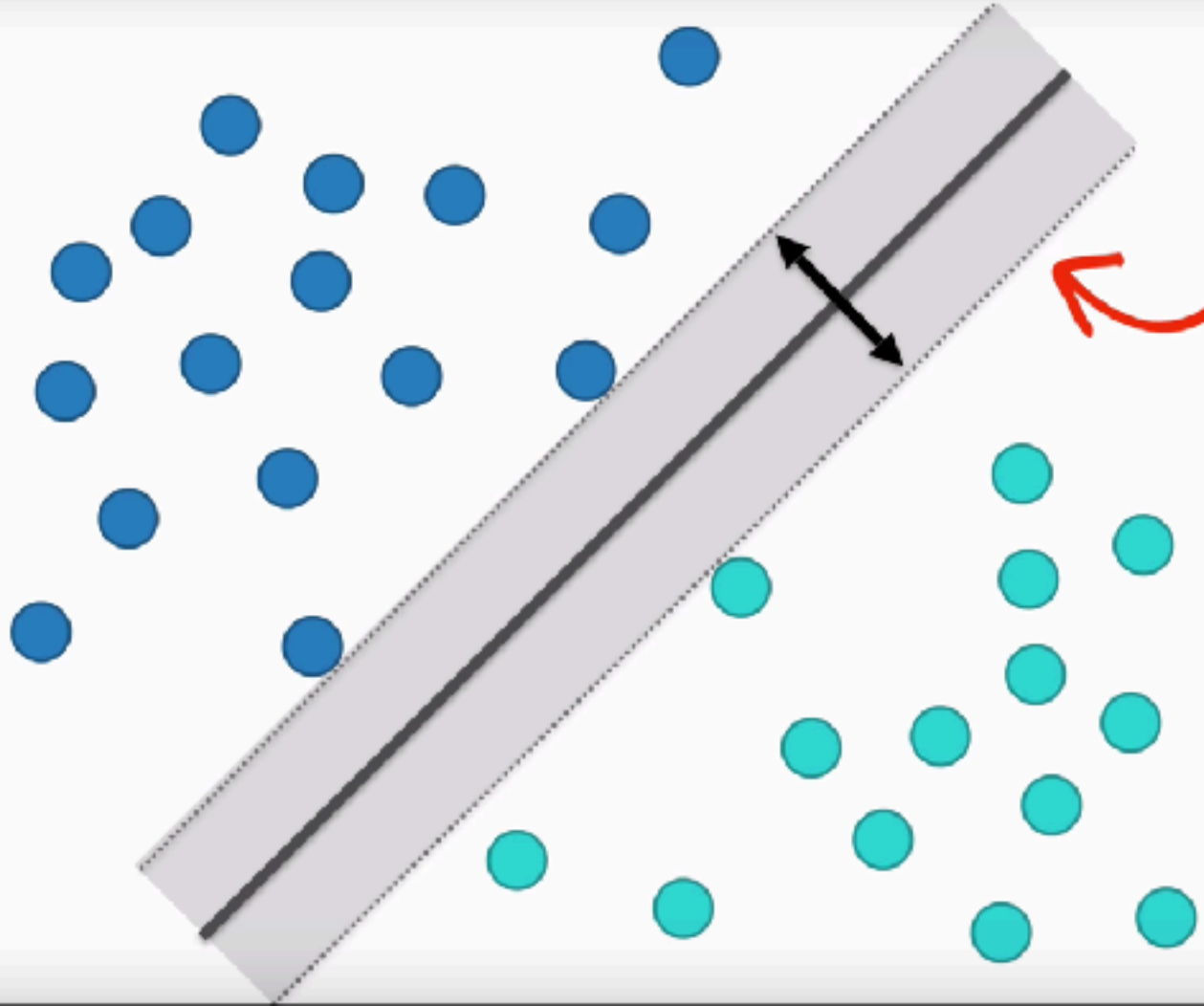
Why is this the best split?



Why is this the best split?

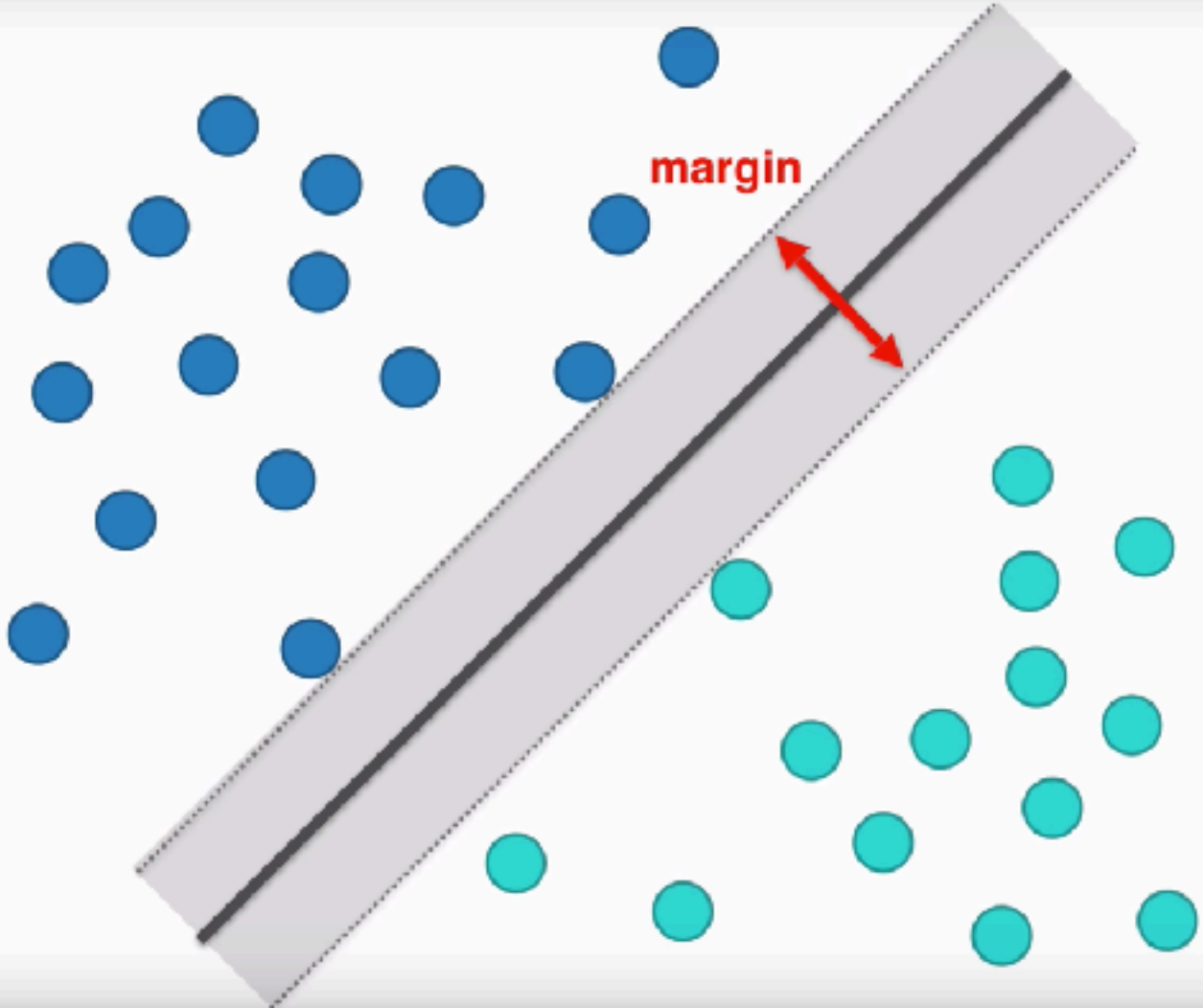

Why is this the best split?
This is the widest road that separates the two groups


Why is this the best split?
This is the widest ~~road~~ that separates the two groups

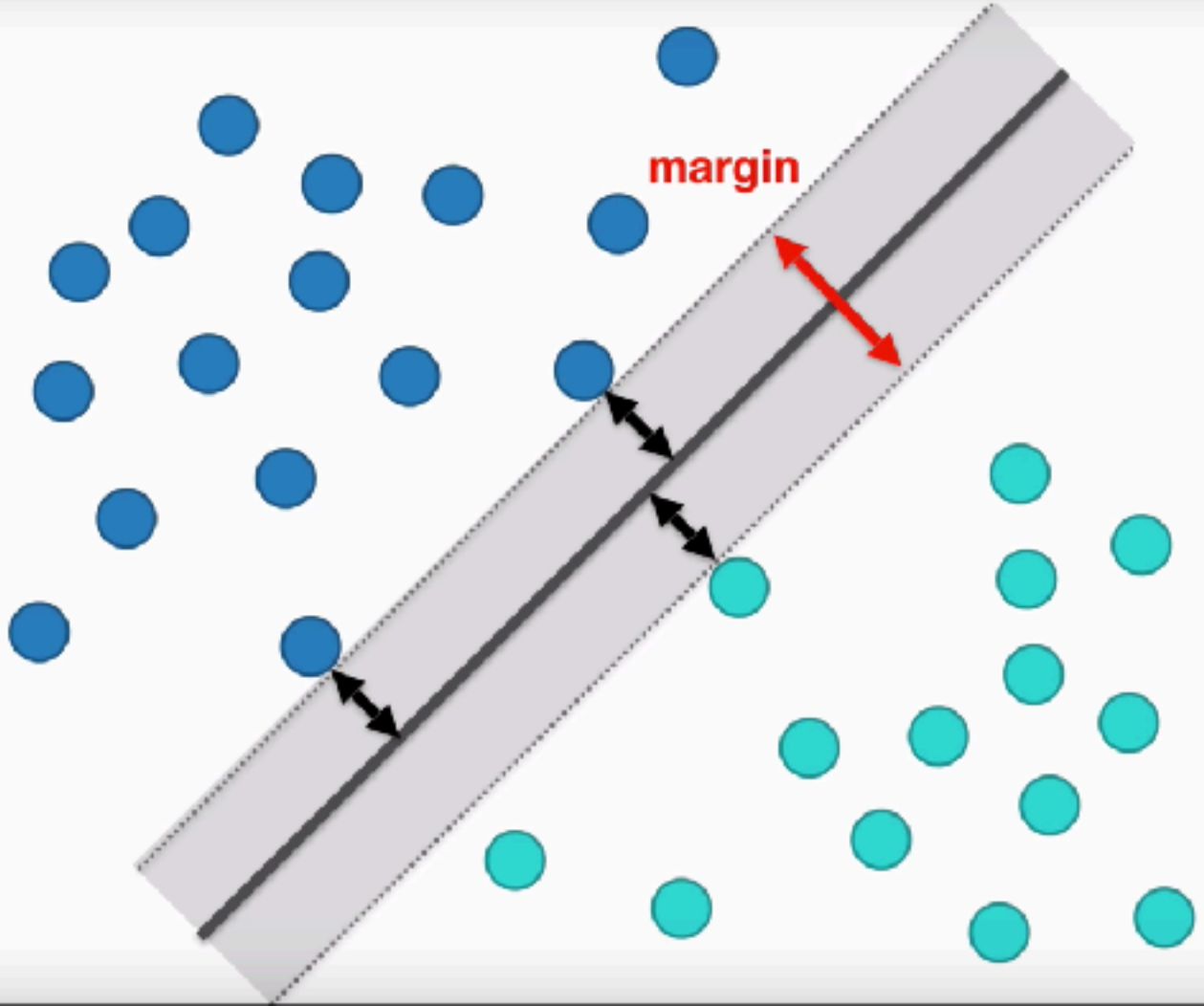


Why is this the best split?

This is the widest margin that separates the two groups

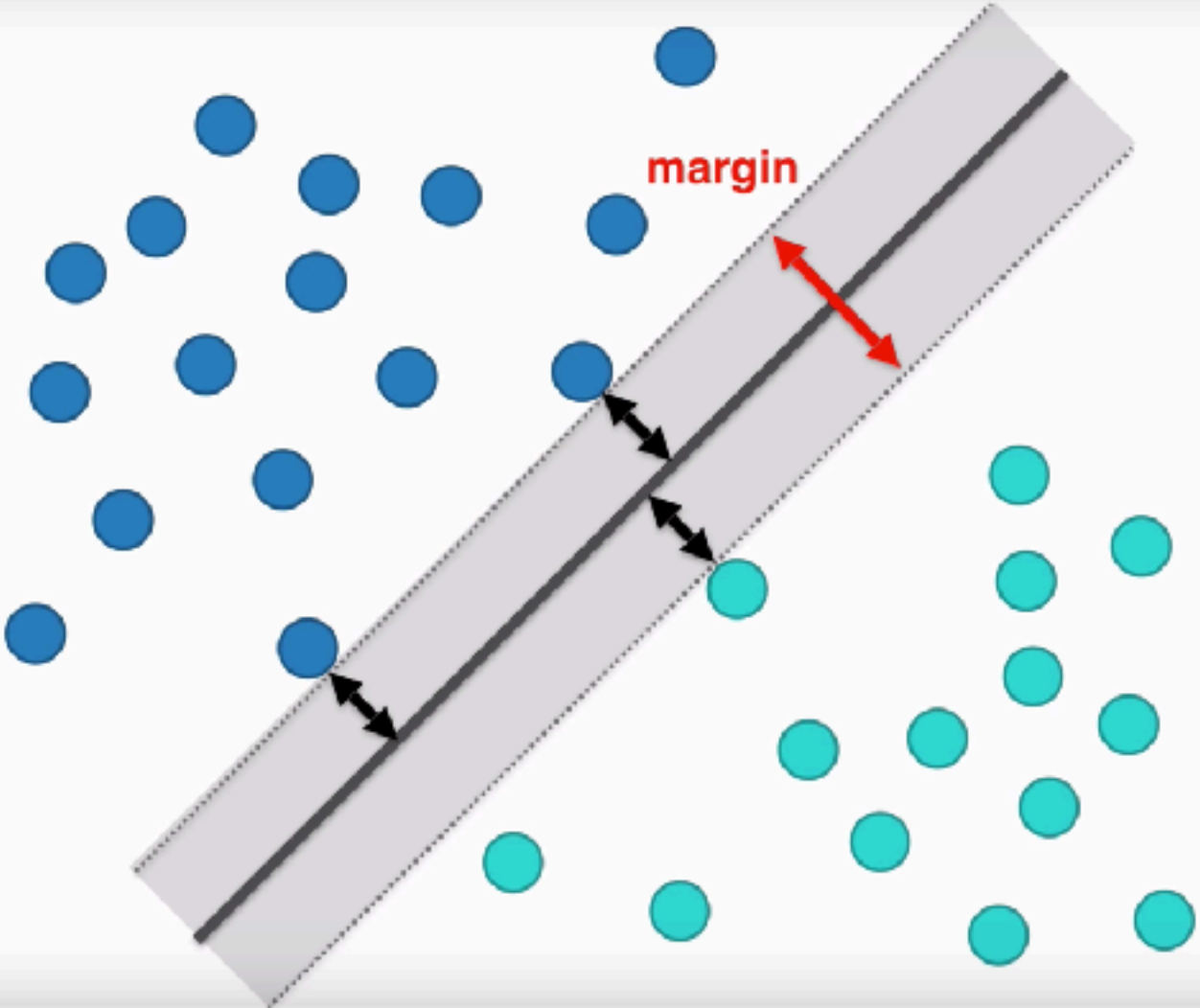


Why is this the best split?



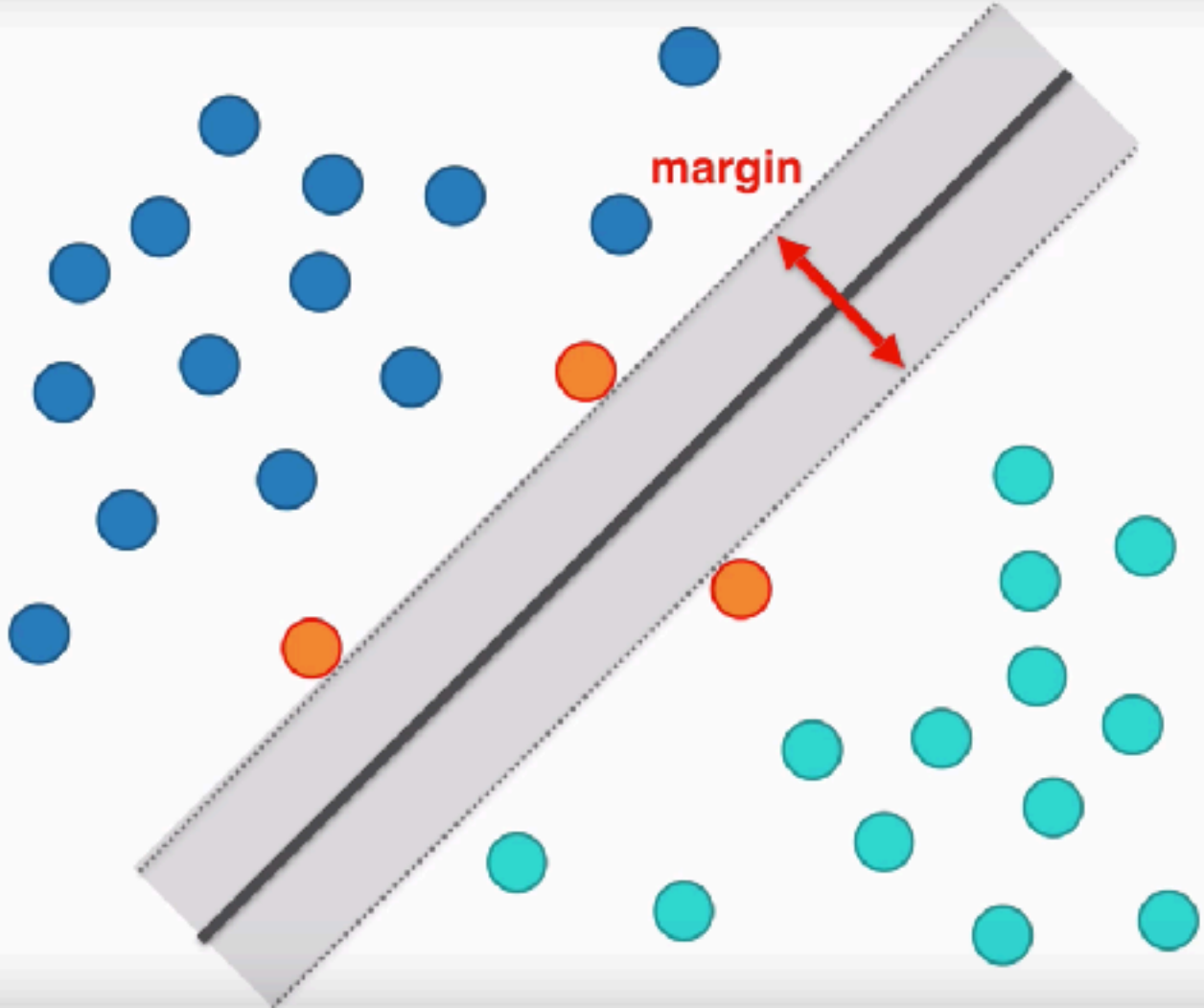
Why is this the best split?

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



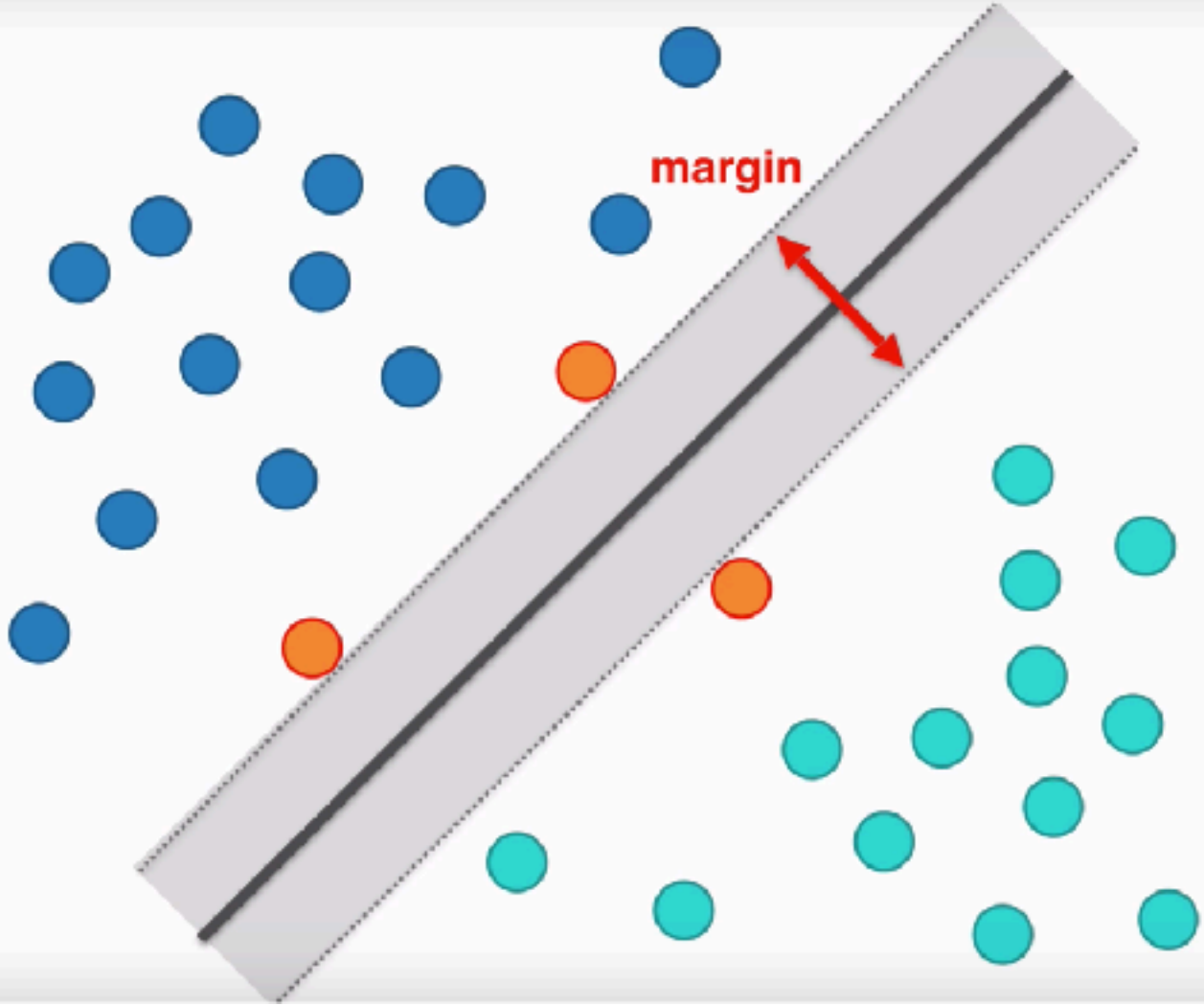
Why is this the best split?

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



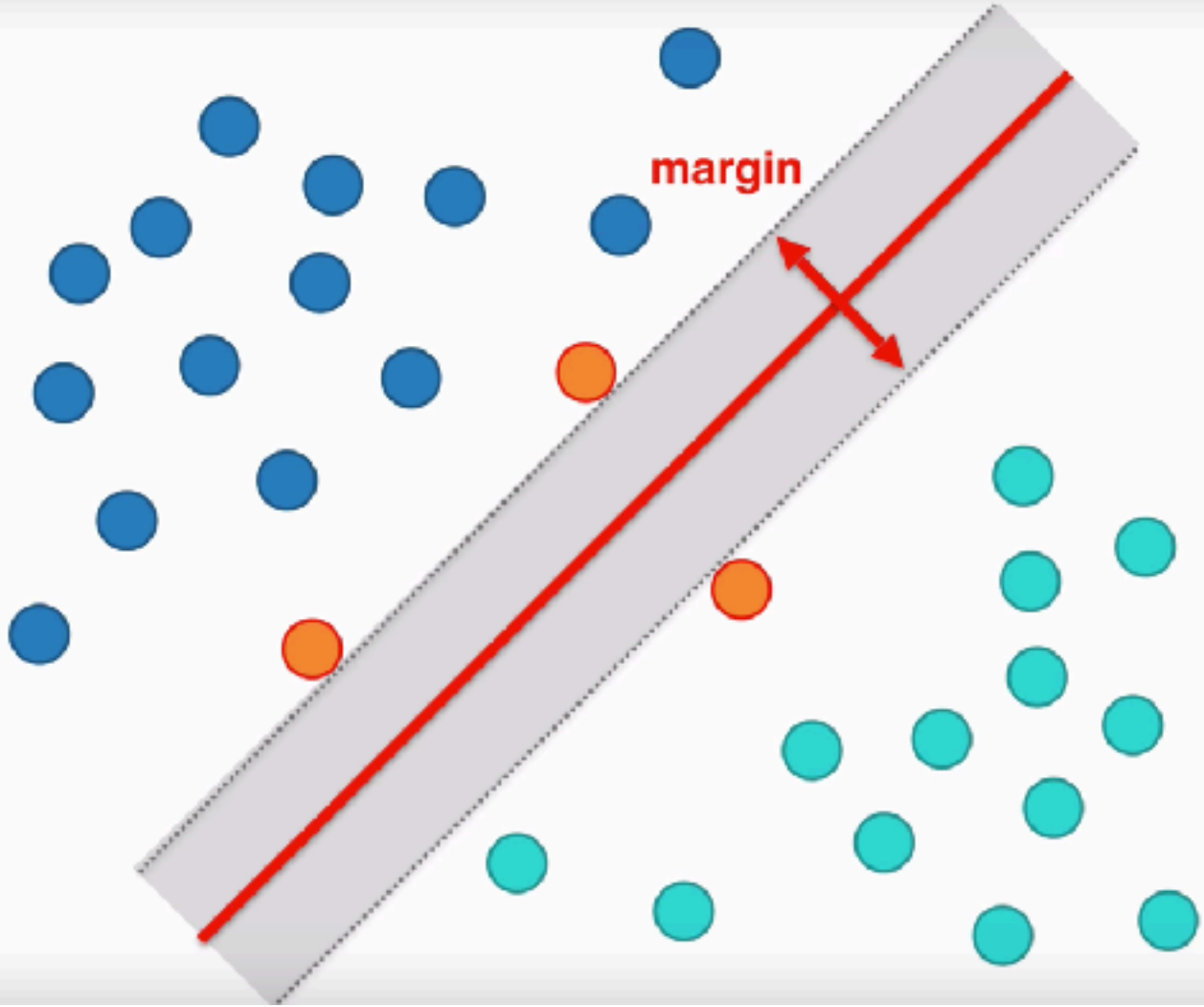
Why is this the best split?

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



Why is this the best split?

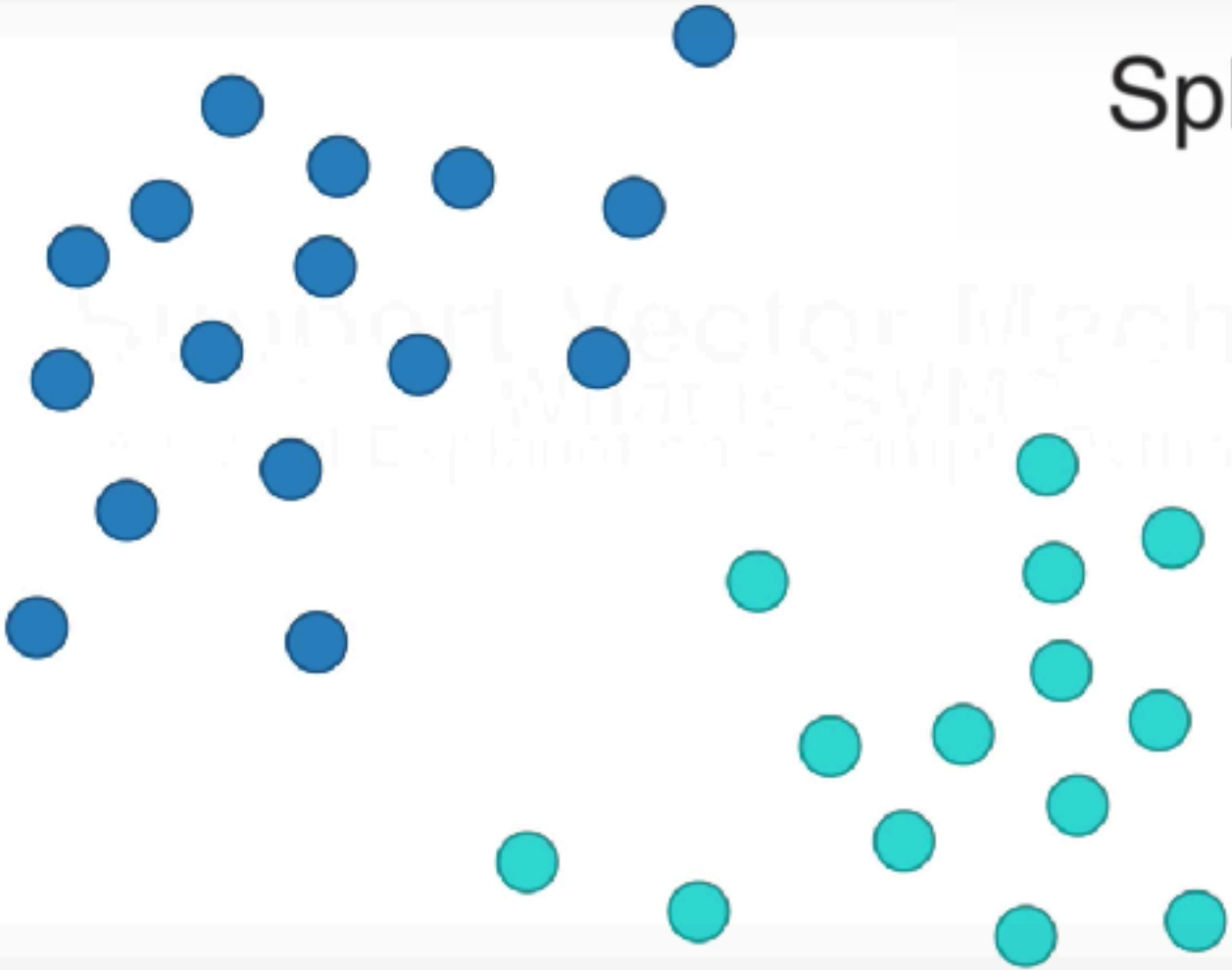
The distance between the support vectors and the ~~line~~ are as far as possible

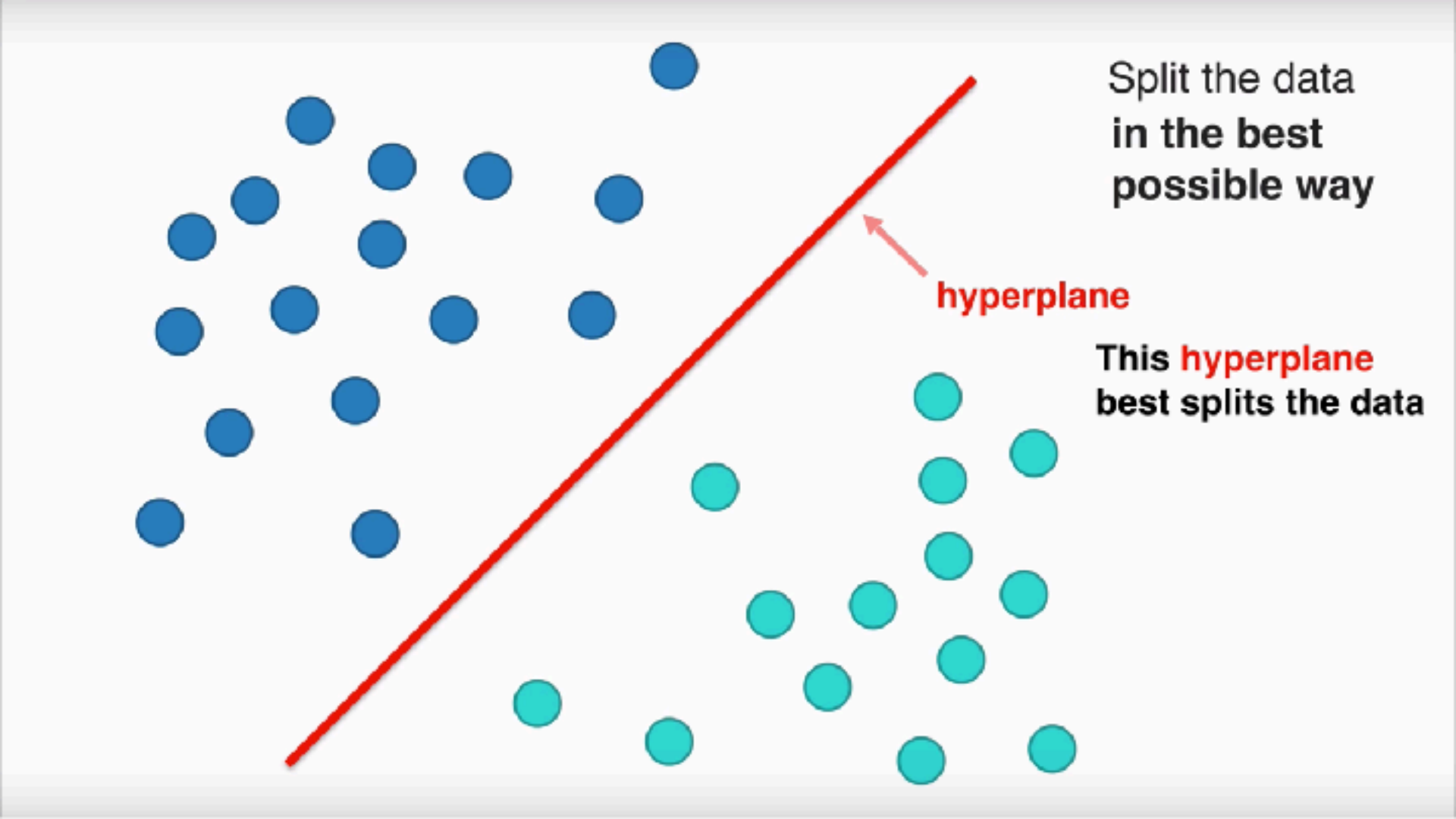


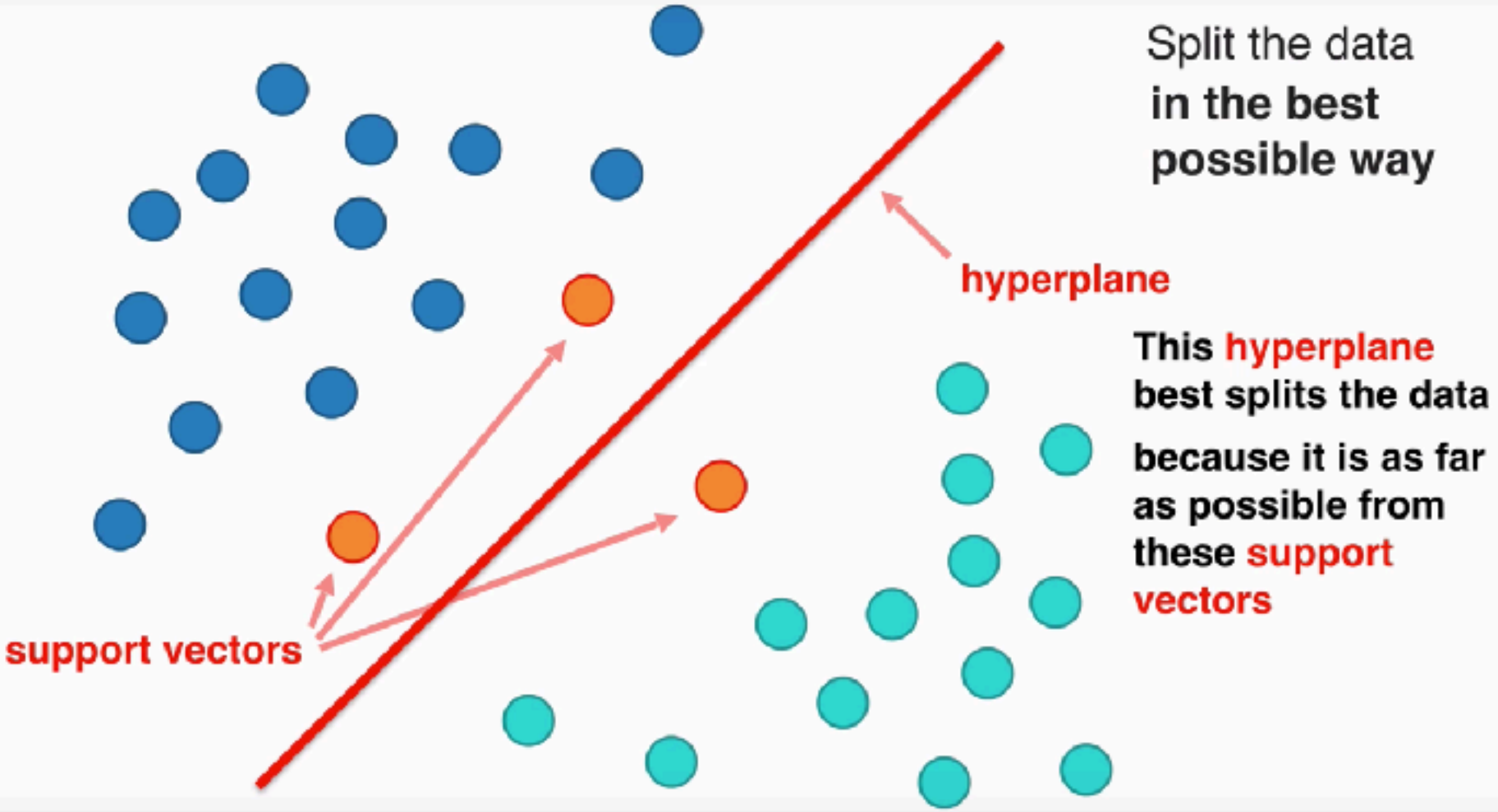
Why is this the best split?

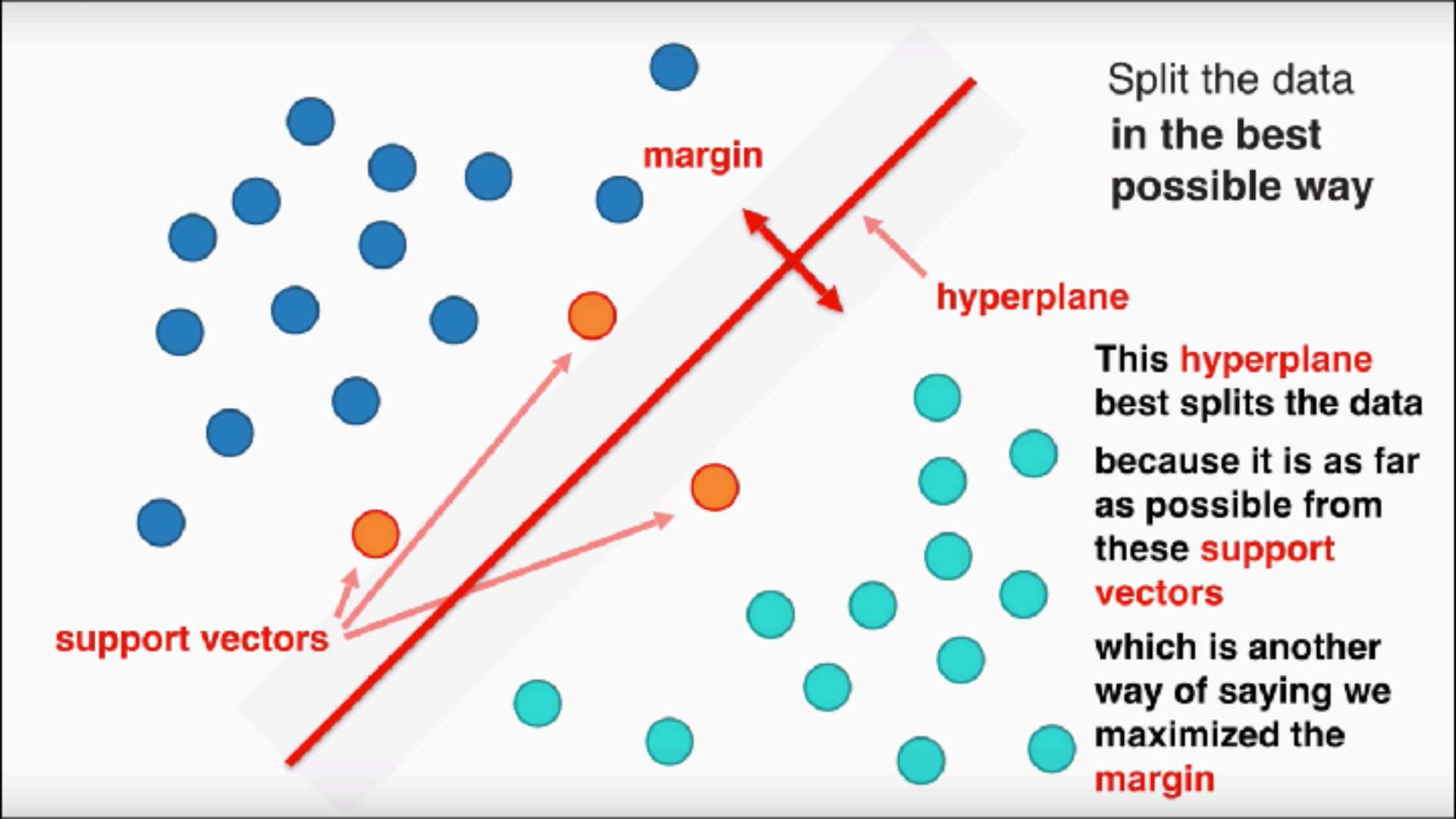
The distance between the support vectors and the hyperplane are as far as possible

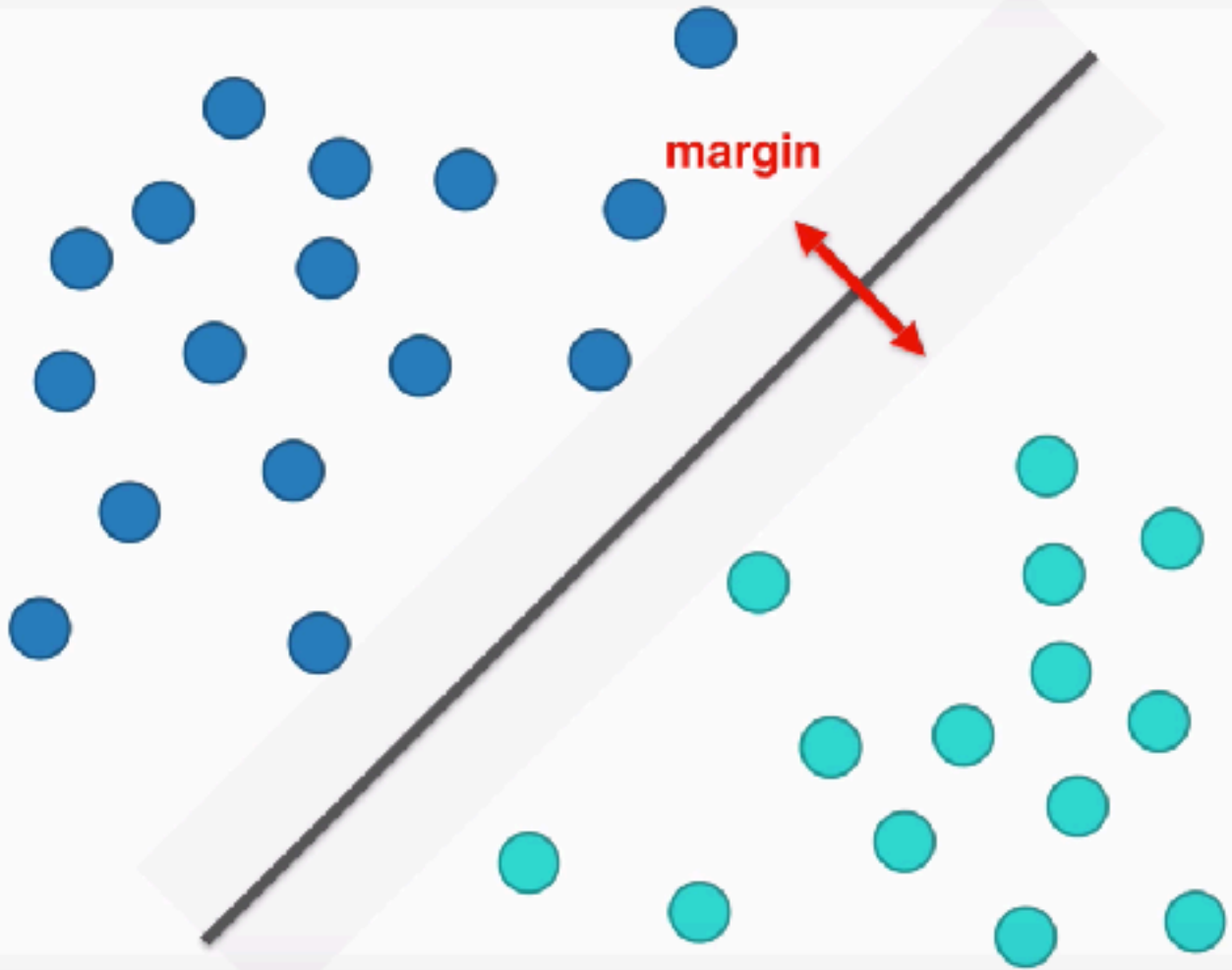
Split the data



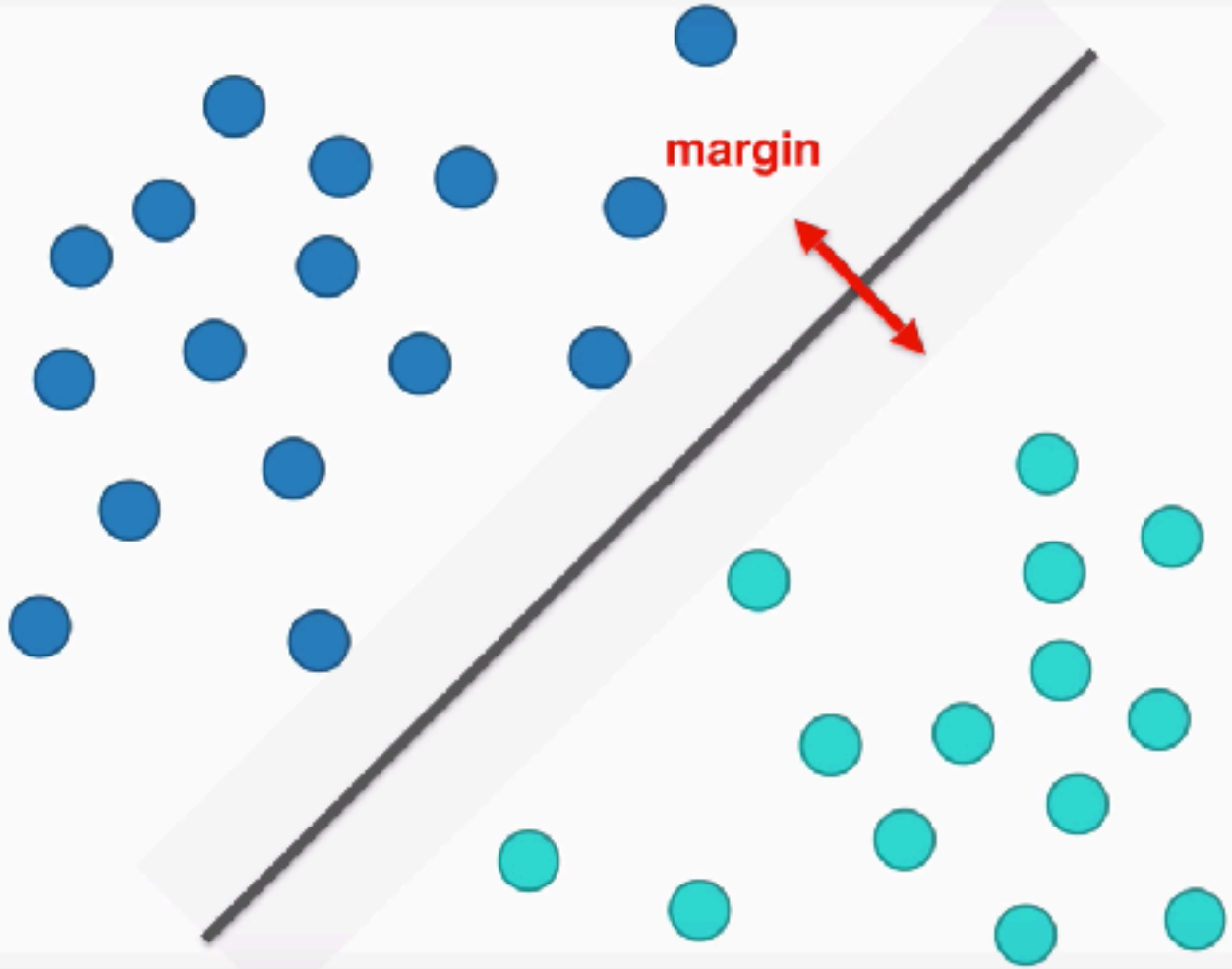

Split the data
in the best
possible way
hyperplane
This hyperplane
best splits the data


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


Split the data
in the best
possible way
margin
hyperplane
This hyperplane
best splits the data
because it is as far
as possible from
these support
vectors
which is another
way of saying we
maximized the
margin
support vectors

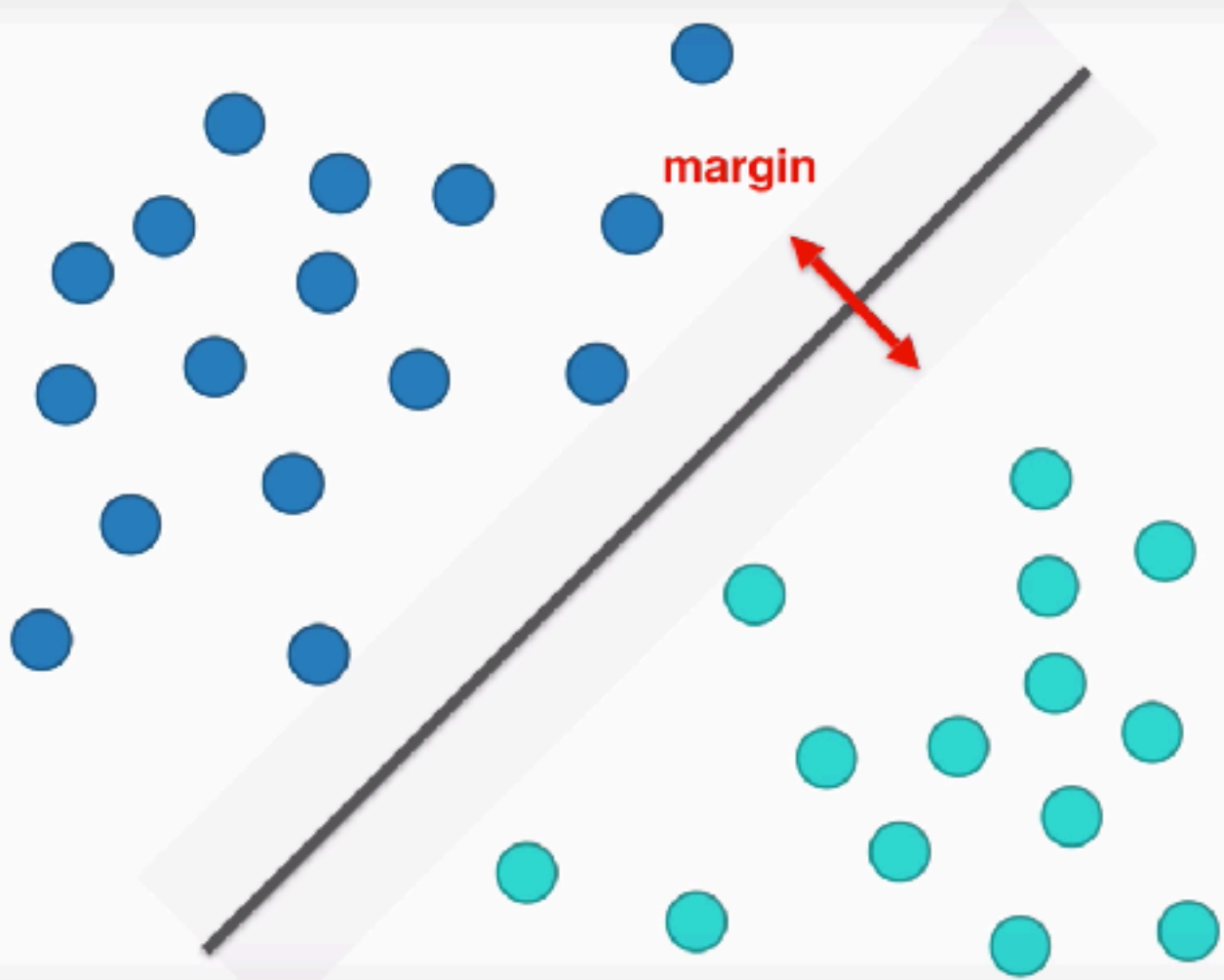


How do you maximize the margin?



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

versus

Muffins

"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

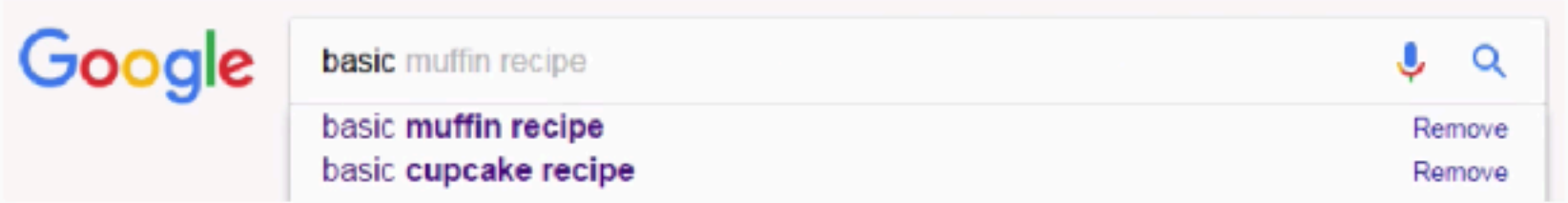
Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

The Steps

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

1. Find the data

1. Find the data



Recorded the top 10 muffin and top 10 cupcake recipes

Recipe	Flour	Sugar	Butter	Milk	Egg	Baking Powder	Salt	Vanilla
Muffin1	2 cups	1/2 cup	1/4 cup	1 cup	2 eggs	1 tbsp	1/2 tsp	1 tsp
Cupcake1	2 cups	3/4 cup	1/2 cup	1 cup	2 eggs	2 tsp	1/2 tsp	1 tsp
...	...	...	...	...	...	...	...	...

1. Find the data

Problem

Each recipe yields different amounts of batter

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

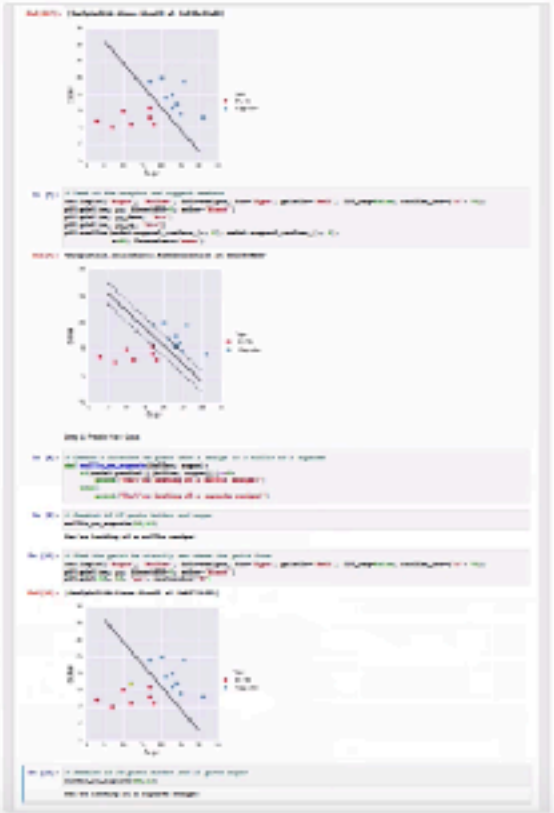
The Challenge

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

The Steps

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

Python Script



a. Import Libraries

b. Import Data

c. Prepare the Data

d. Fit the Model

e. Visualize Results

f. 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)
```

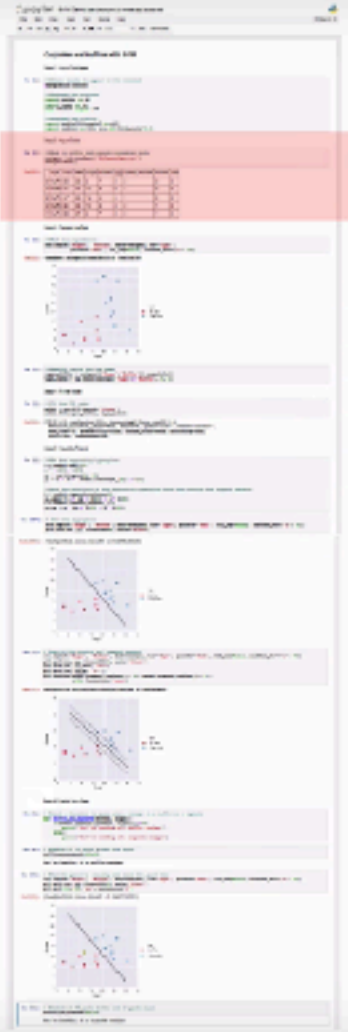
b. Import Data

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

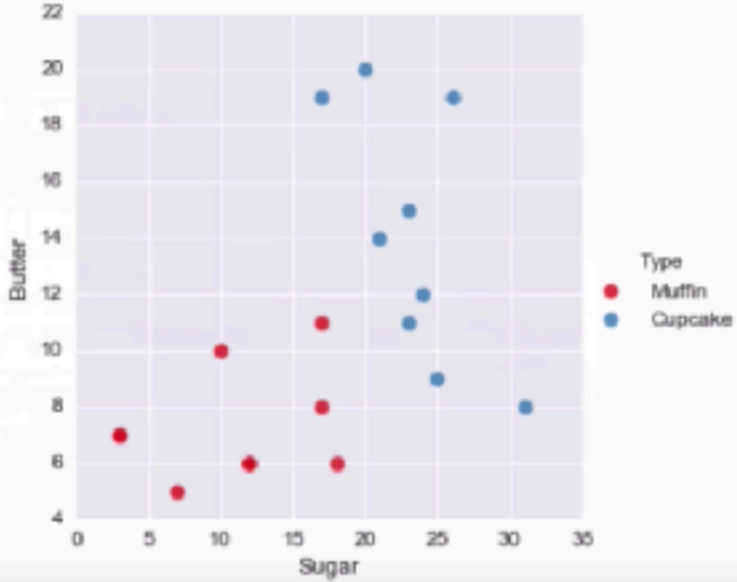


c. Prepare the Data

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)
```

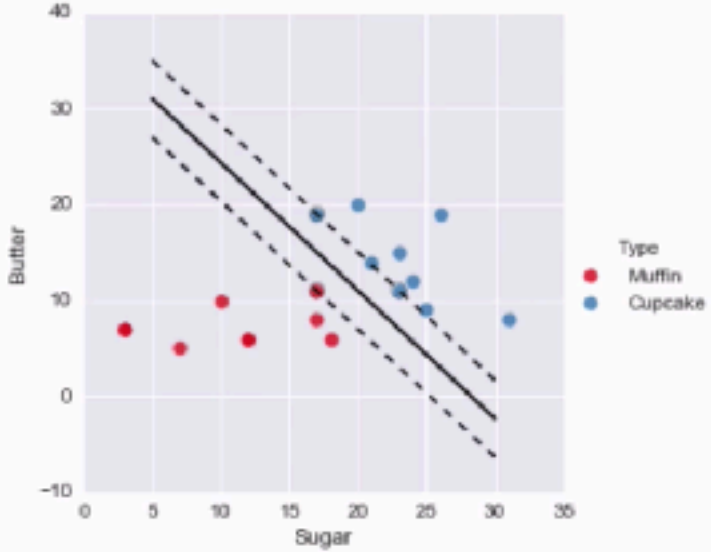
e. Visualize Results

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

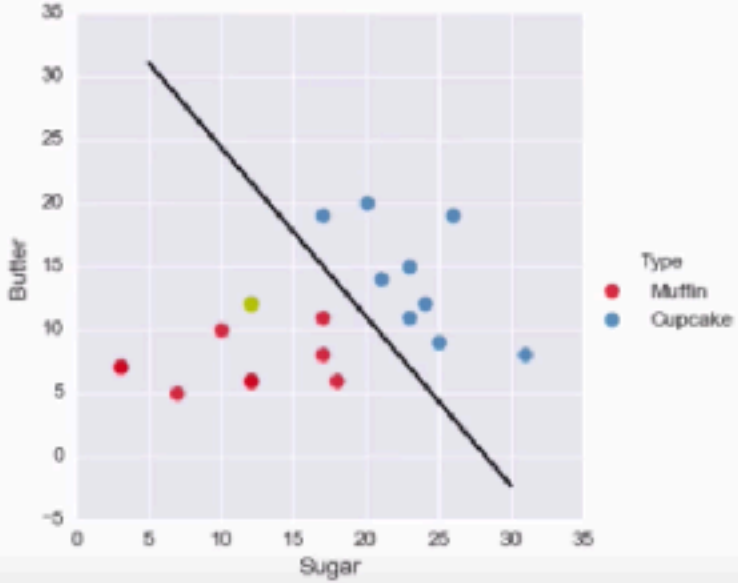
Classify recipes as cupcakes or muffins. ✓

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>]
```



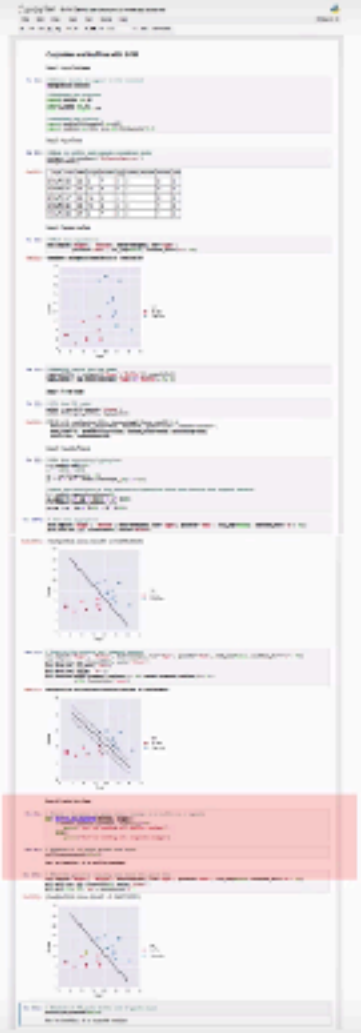
f. Predict New Case

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

versus

Muffins

Cupcakes and muffins can be classified using Support Vector Machines!

Basic case ✓

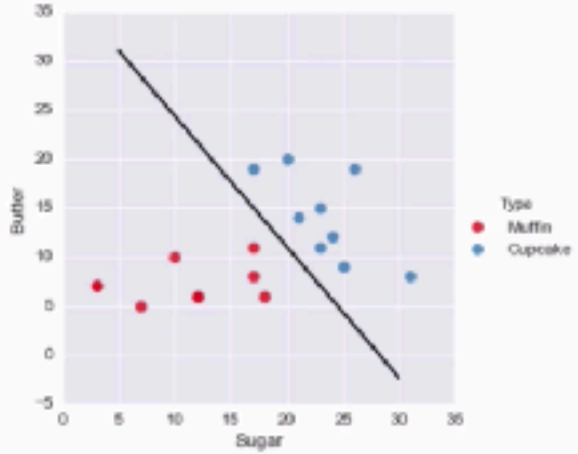
Up next: How to make SVM even more powerful

Agenda

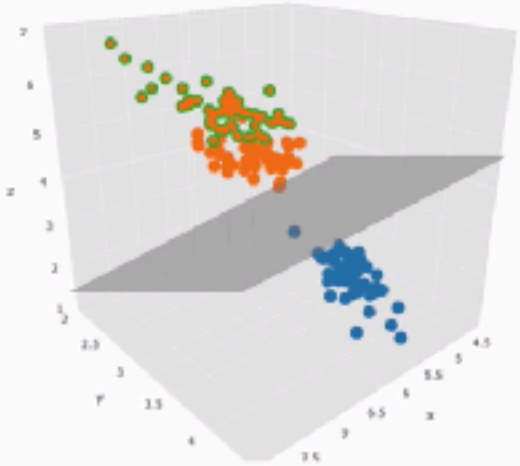
1. SVM Basics
   - Visual introduction
   - Example in Python
2. **Additional Complexities**
   - Higher dimensions
   - Multiple classes
   - C parameter
   - Kernel trick
3. Closing Remarks
   - Pros and cons
   - Other techniques

Higher Dimensions: Visual

2D: Separate with Line



3D: Separate with Plane



4D+: Separate with Hyperplane

Hard to Visualize

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

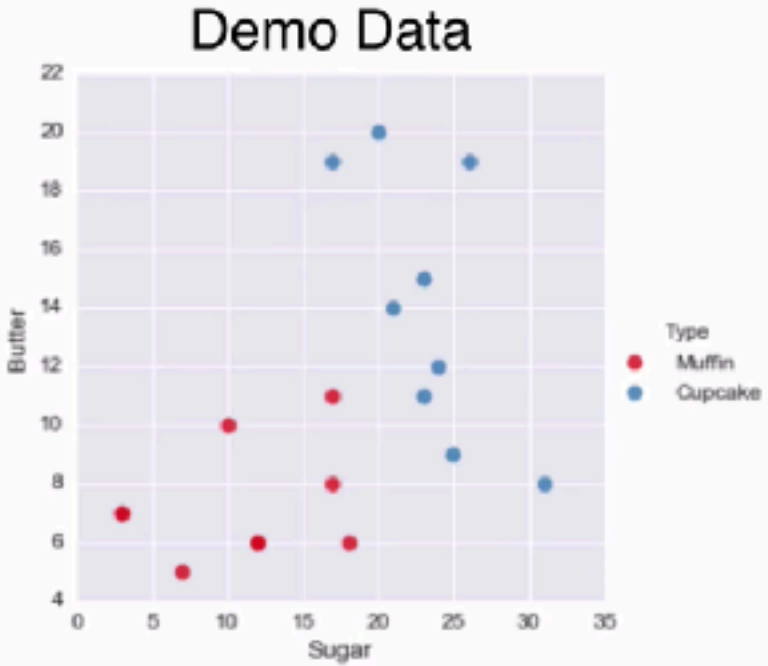
```
# Fit SVM model for sugar & butter
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

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

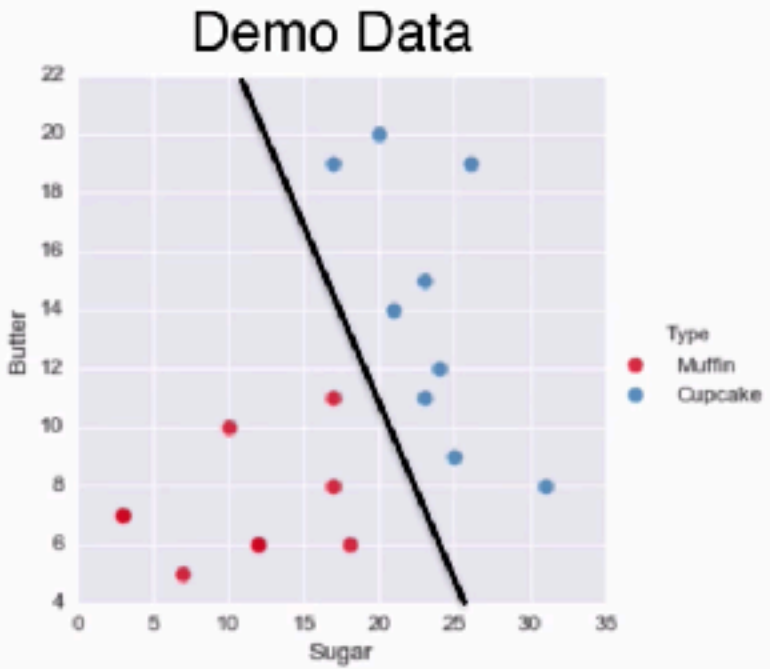
all_ingredients

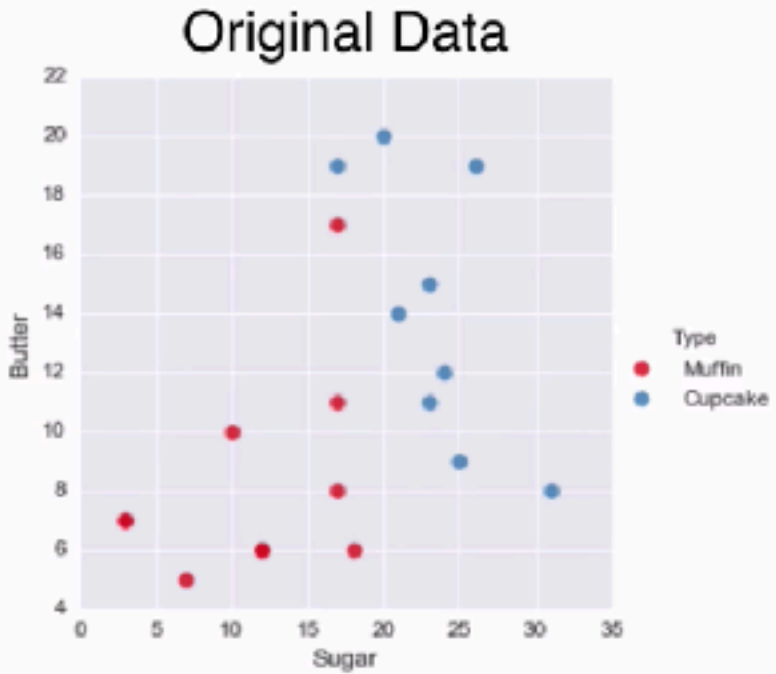
```
[55, 28,  3,  7,  5,  2,  0,  0],
[47, 24, 12,  6,  9,  1,  0,  0],
[47, 23, 18,  6,  4,  1,  0,  0],
[50, 25, 12,  6,  5,  2,  1,  0],
[55, 27,  3,  7,  5,  2,  1,  0], ...
```

C Parameter: Visual

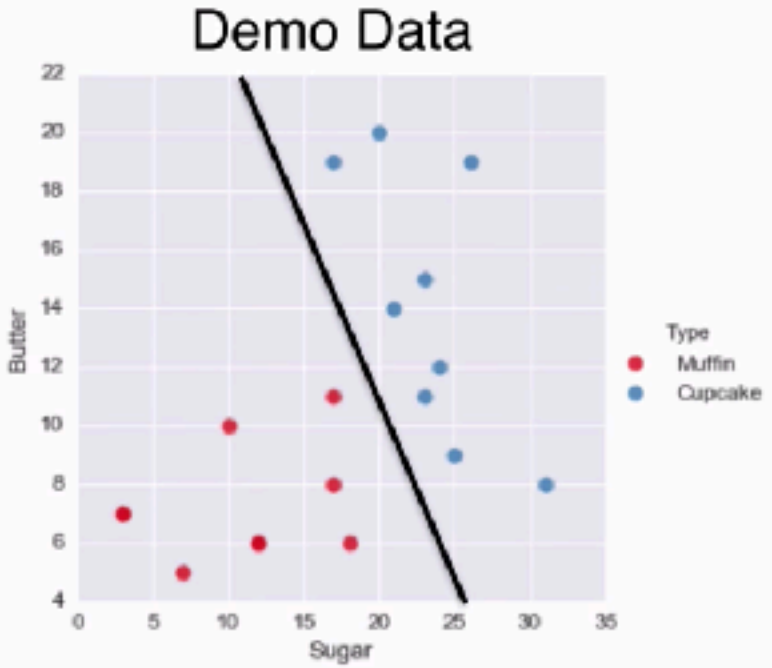


C Parameter: Visual





C Parameter: Visual



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

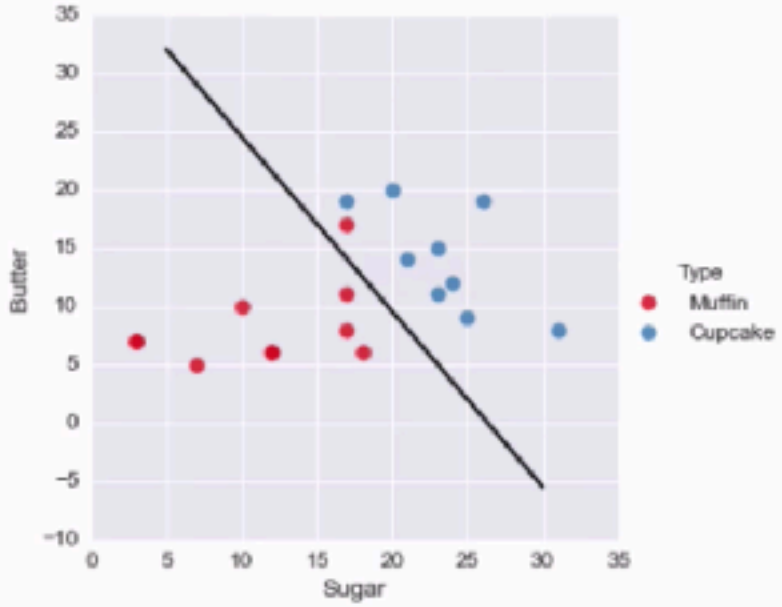
C Parameter: Comparison

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

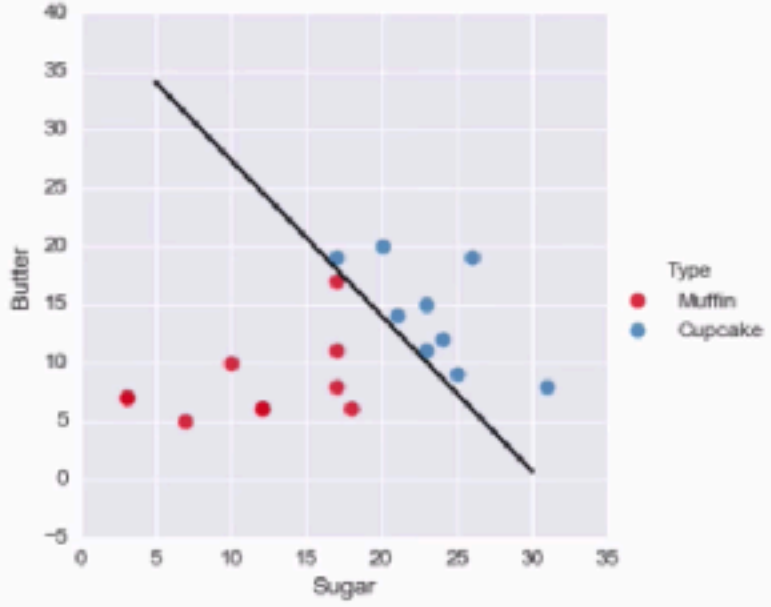
```
# Fit the SVM model with a HIGH C
model = svm.SVC(kernel='linear', C=2**5)
model.fit(sugar_butter, type_label)
```

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)
```



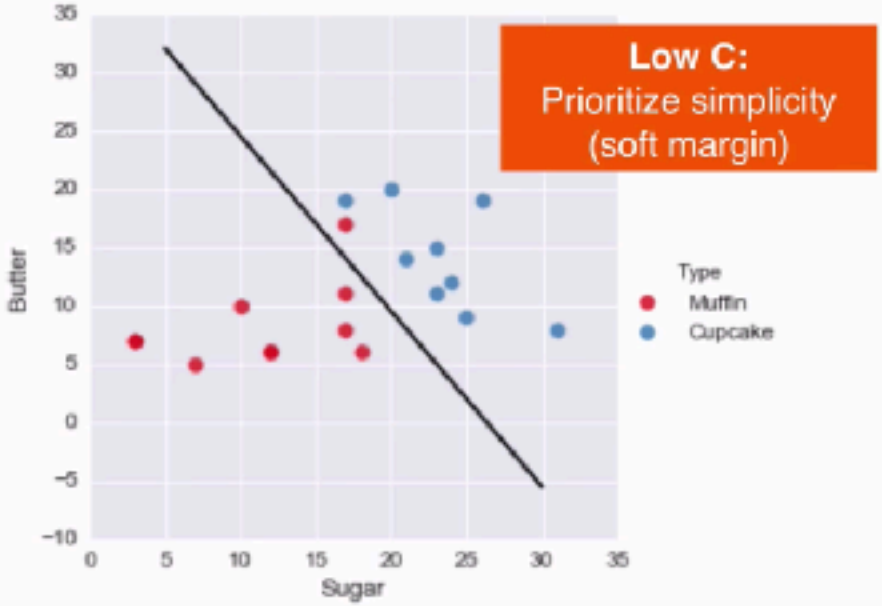

```
# Fit the SVM model with a HIGH C
model = svm.SVC(kernel='linear', C=2**5)
model.fit(sugar_butter, type_label)
```

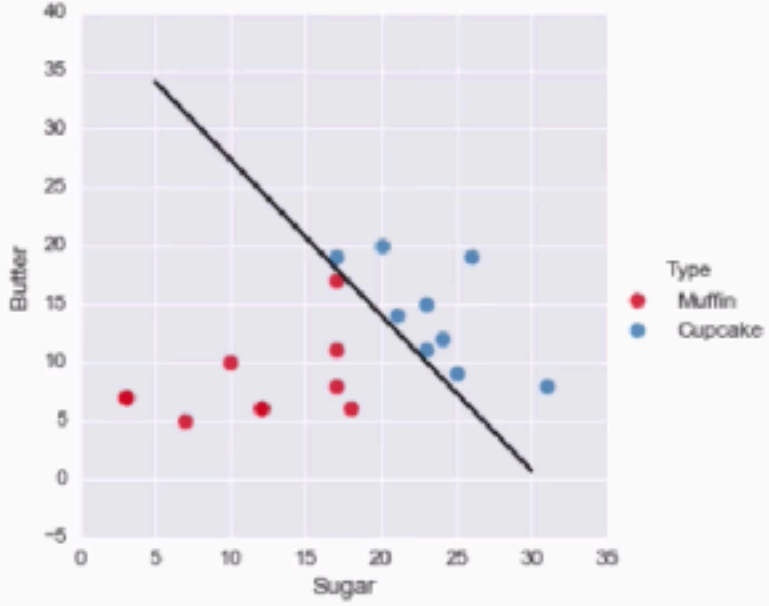


C Parameter: Comparison

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

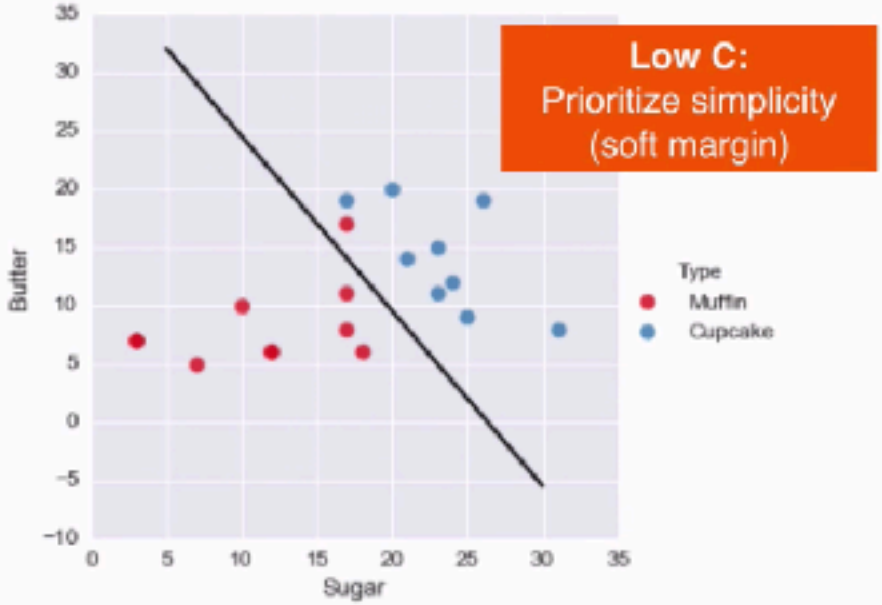
```
# Fit the SVM model with a HIGH C
model = svm.SVC(kernel='linear', C=2**5)
model.fit(sugar_butter, type_label)
```



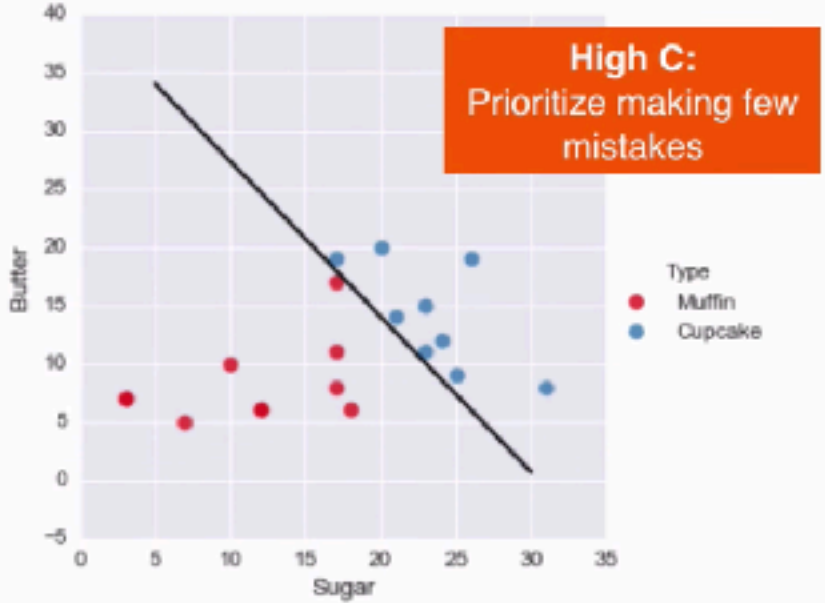


C Parameter: Comparison

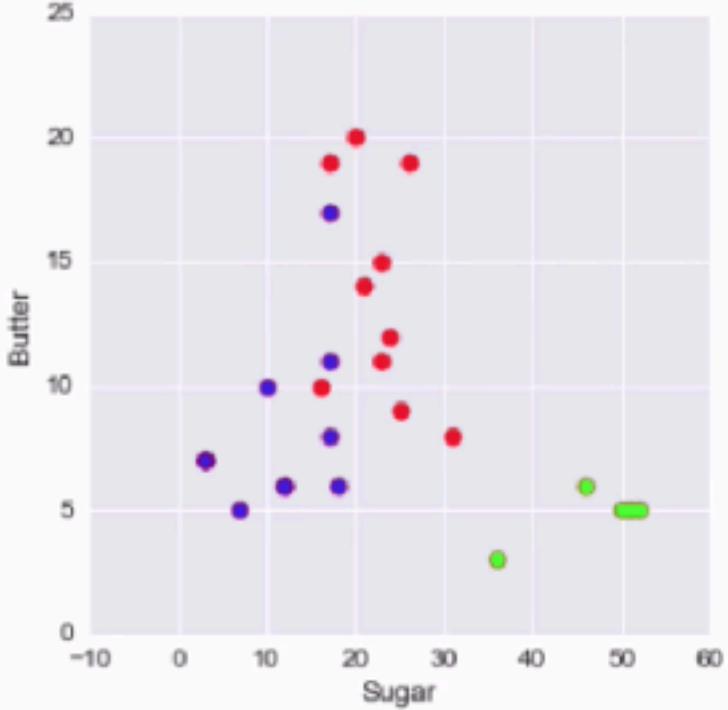
```
# Fit the SVM model with a LOW C
model = svm.SVC(kernel='linear', C=2**-5)
model.fit(sugar_butter, type_label)
```

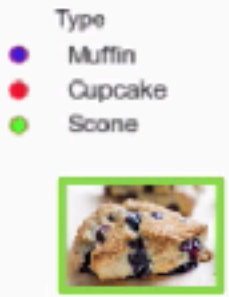


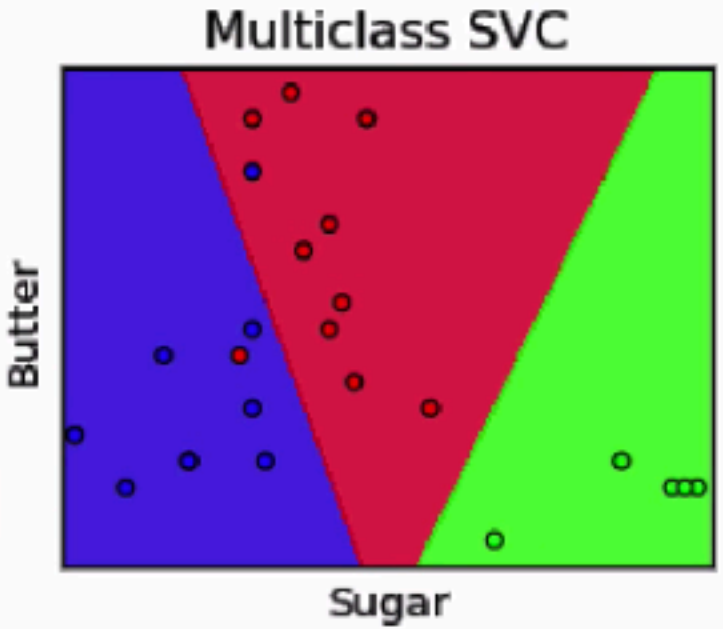

```
# Fit the SVM model with a HIGH C
model = svm.SVC(kernel='linear', C=2**5)
model.fit(sugar_butter, type_label)
```



Multiple Classes: Visual







Multiple Classes: Code

Original Code
(2 classes)

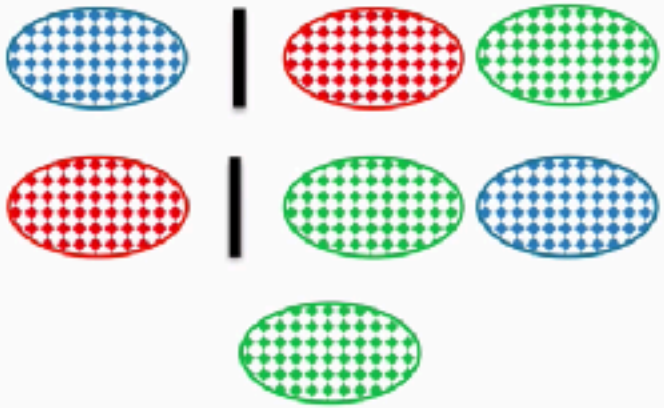
```
# Fit the SVM model
model = svm.SVC(kernel='linear')
model.fit(sugar_butter, type_label)
```

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)
```

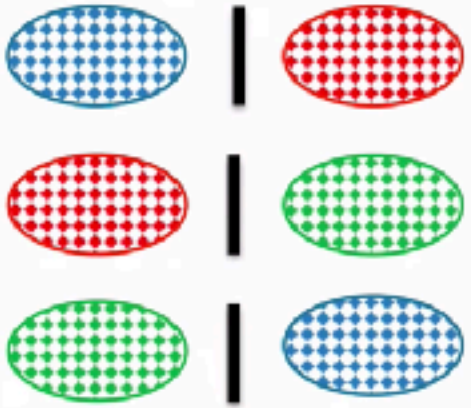
Multiple Classes: Comparison

OVR: One vs Rest



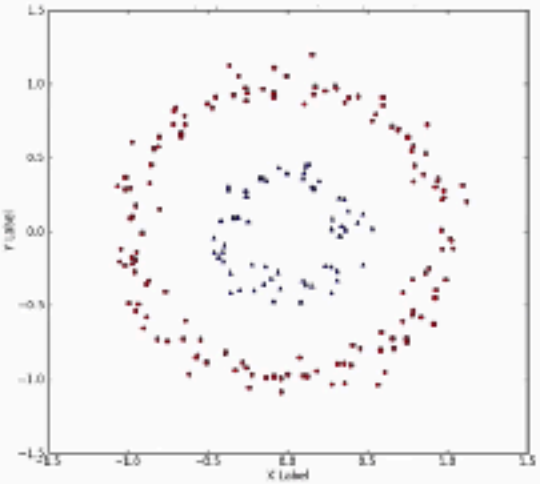
Pros: Fewer classifications
Cons: Classes may be imbalanced

OVO: One vs One

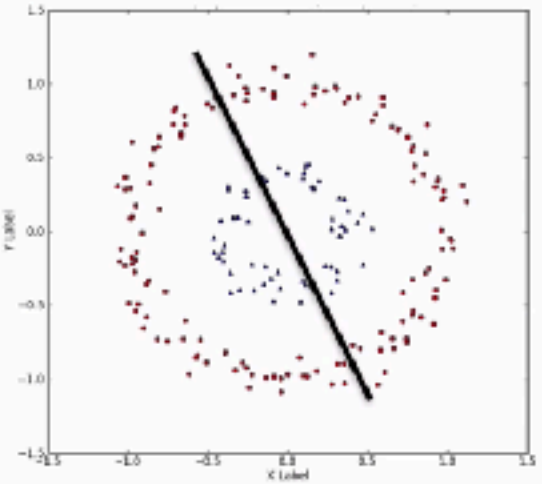


Pros: Less sensitive to imbalance
Cons: More classifications

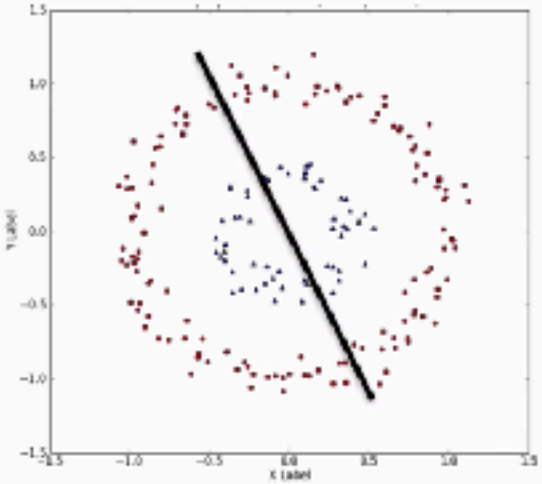
Kernel Trick: Visual

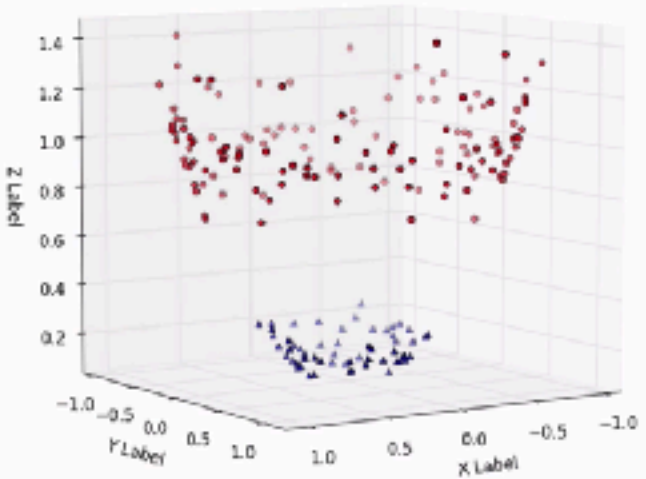


Kernel Trick: Visual

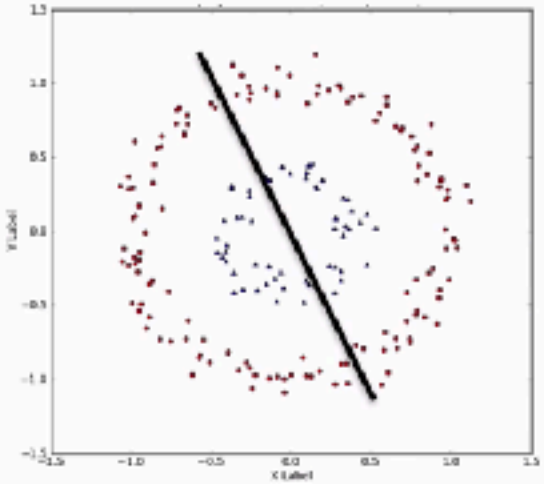


Kernel Trick: Visual

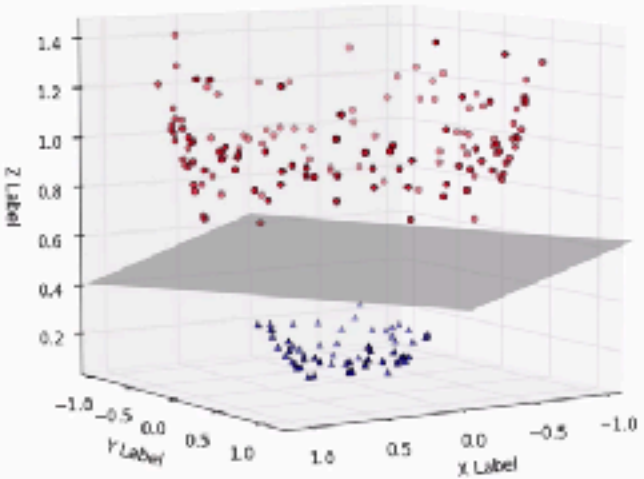




Kernel Trick: Visual



Kernel



Kernel Options

- Linear
- Radial Basis Function
- Polynomial
- Sigmoid

Kernel Trick: Code

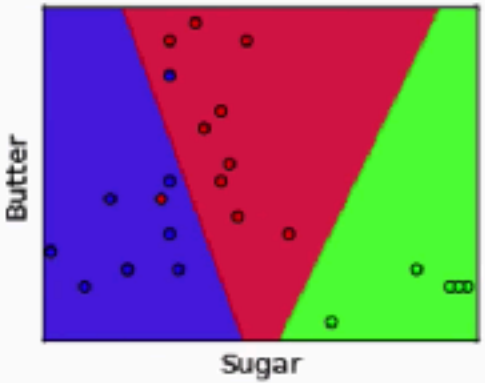
Original Code (linear)

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

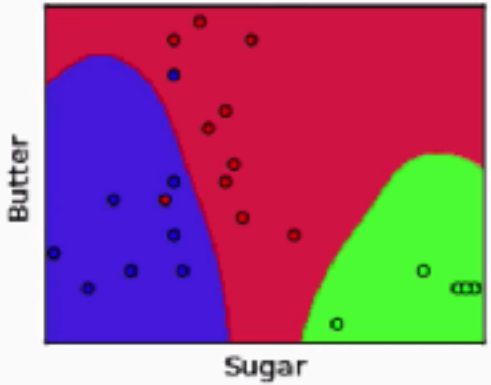
Updated Code (RBF)

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

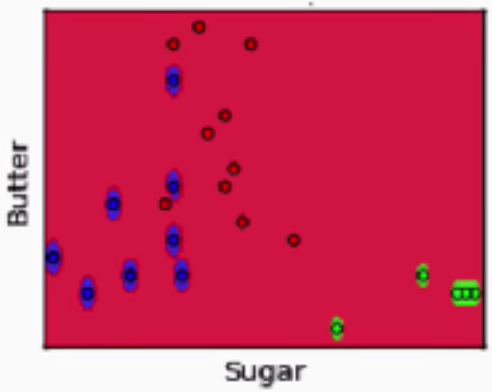
Kernel Trick: Comparison



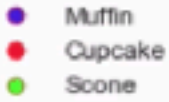
Kernel: Linear
C: 1



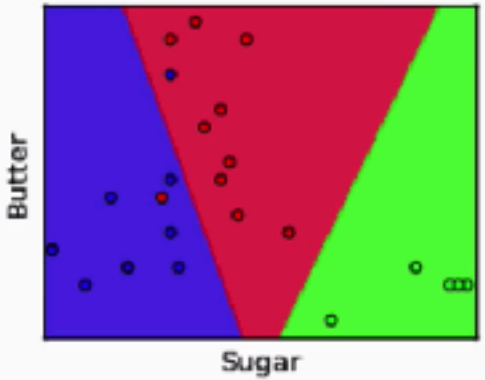
Kernel: RBF
C: 1
Gamma: 2^-5



Kernel: RBF
C: 1
Gamma: 2^1

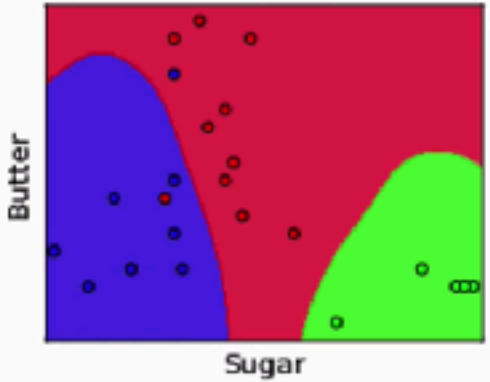


Kernel Trick: Comparison



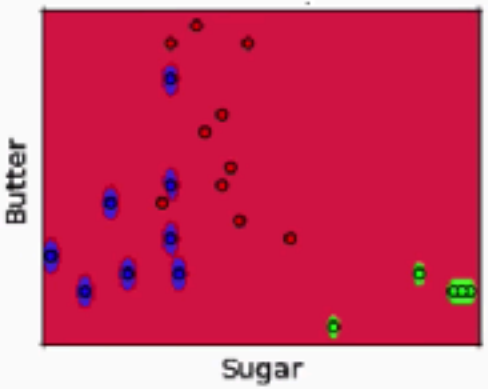
Kernel: Linear
C: 1

Muffin
Cupcake
Scone



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

Small Gamma:
Less complexity



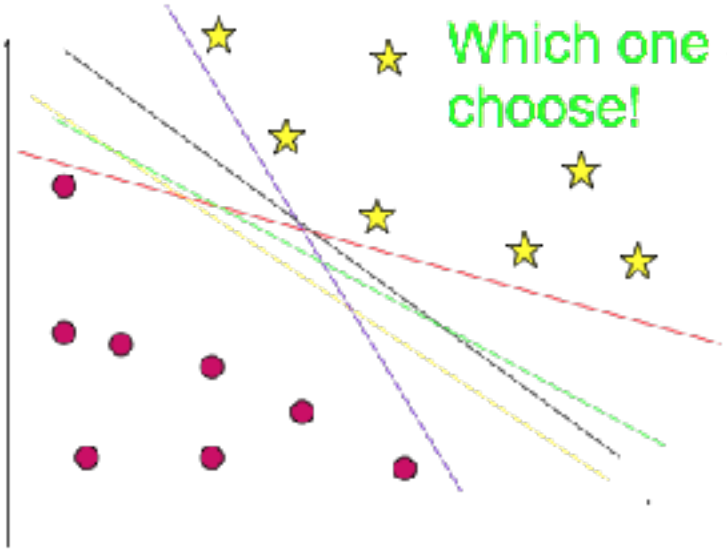
Kernel: RBF
C: 1
Gamma: 2^1

Large Gamma:
More complexity

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

$y_i = +1$
$y_i = -1$



Which one should we choose!

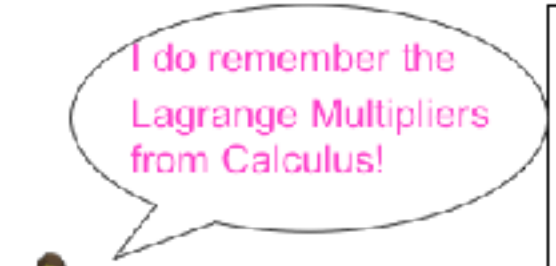
Yes, There are many possible separating hyperplanes

It could be this one or this or this or maybe….!

SVM : Linear separable case.

The optimization problem:

-Our optimization problem so far:



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

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

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

$$\text{minimize} \quad L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(x_i \cdot w + b) - 1)$$

$$s.t \quad \alpha_i \geq 0$$

SVM : Linear separable case.

The optimization problem cont':

So our primal optimization problem now:

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

$$s.t \quad \alpha_i \geq 0$$

We star solving this problem:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

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

SVM : Linear separable case.

Inroducing The Legrangin Dual Problem.

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

Lagrangian Dual Problem:

$$\text{maximaize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_i \alpha_j y_i y_j x_i^t x_j$$

$$s.t \ \ \alpha_i \geq 0 \ \ and \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

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

Summary

Supervised Learning

Linear Regression

Logistic Regression

SVM

Ensemble

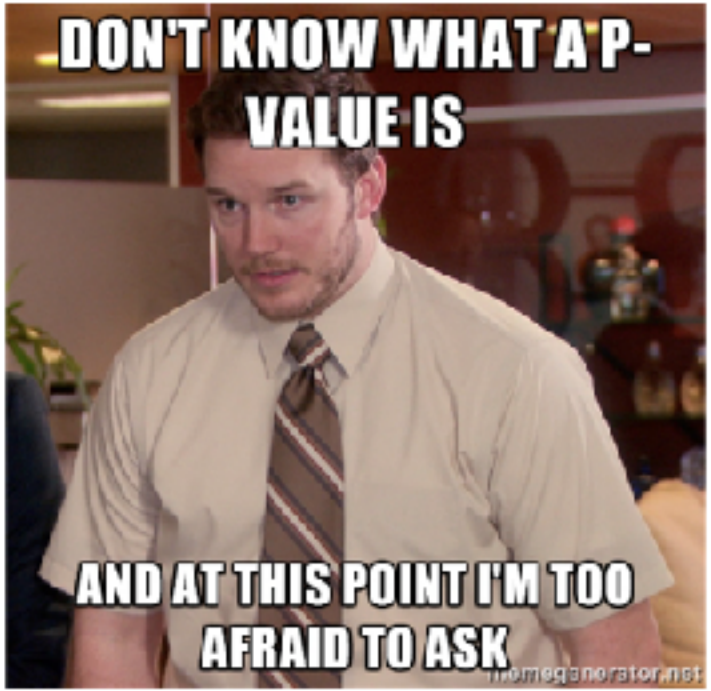
KNN

Unsupervised

K-means

Hierarchical

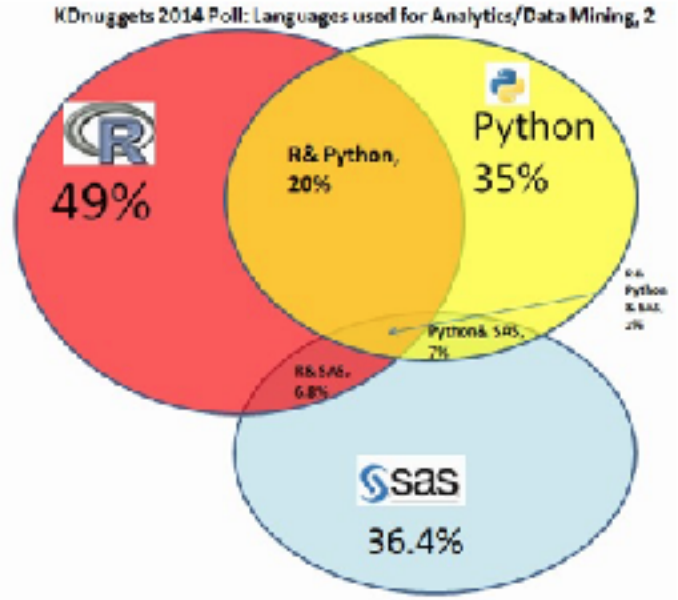
Revise Statistics and Probability



Learn algorithm how it works rather concentrating on output

HOW IT WORKS

Learn and practice a programming language



Big data and Data science are happily married

Practice all algorithm with each use case

Make use of Kaggle and Analytics vidhya

Contribute your work to Opensource

Write a blog

Make your resume attractive

MOHAMED NOORDEEN A
Data Scientist
Tiger Analytics
[redacted]@gmail.com
https://www.linkedin.com/in/nurenaaz/
https://github.com/nurenaaz/
www.nurenaaz.com
(+91) 9789-83[redacted]

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 Science | TIGER ANALYTICS

CONTACT

SKILLS

DATA ANALYTICS PROFESSIONAL

ACADEMIC

2001

2007

2011

CORE COMPETENCIES

CERTIFICATION

Contact

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

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

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

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