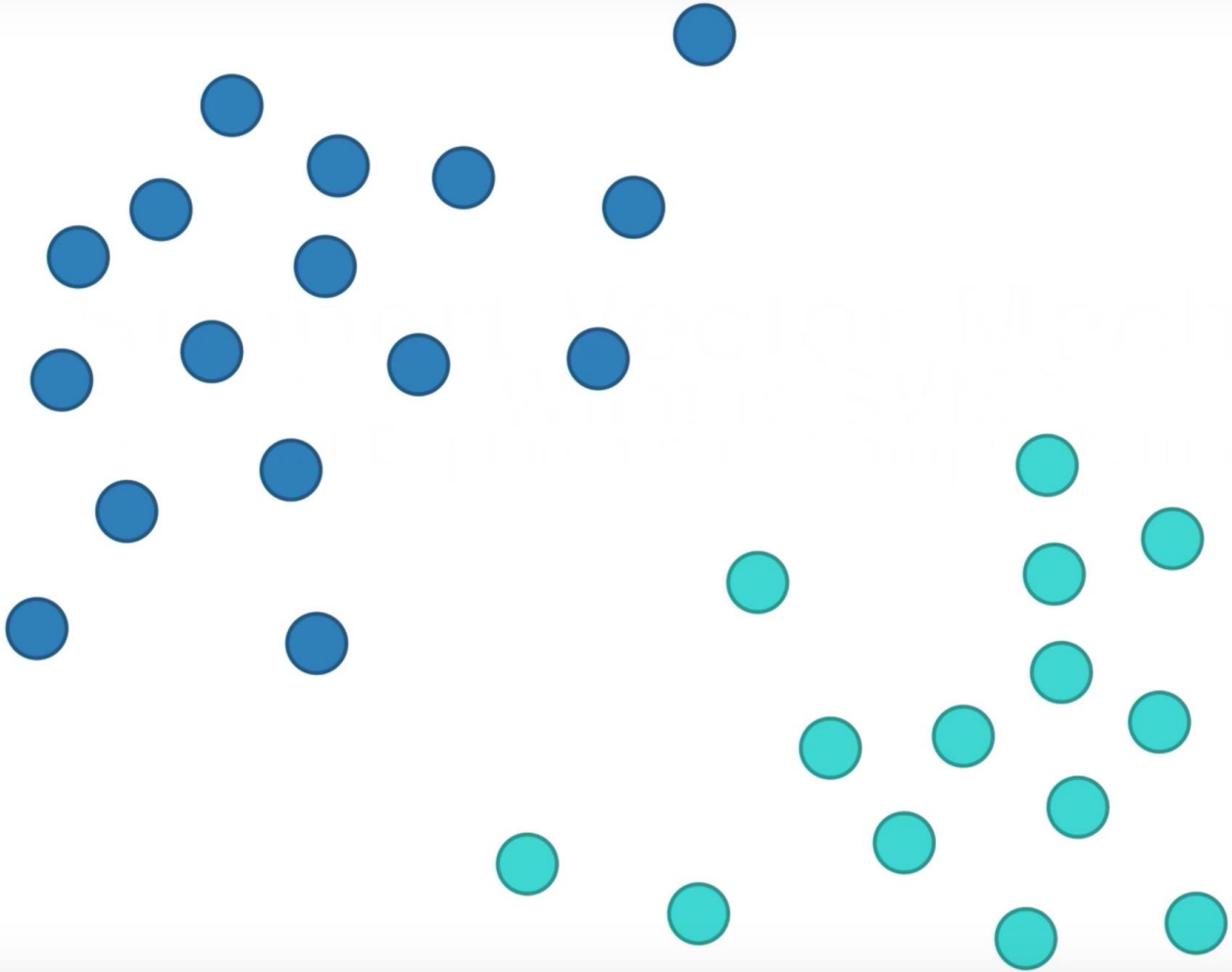
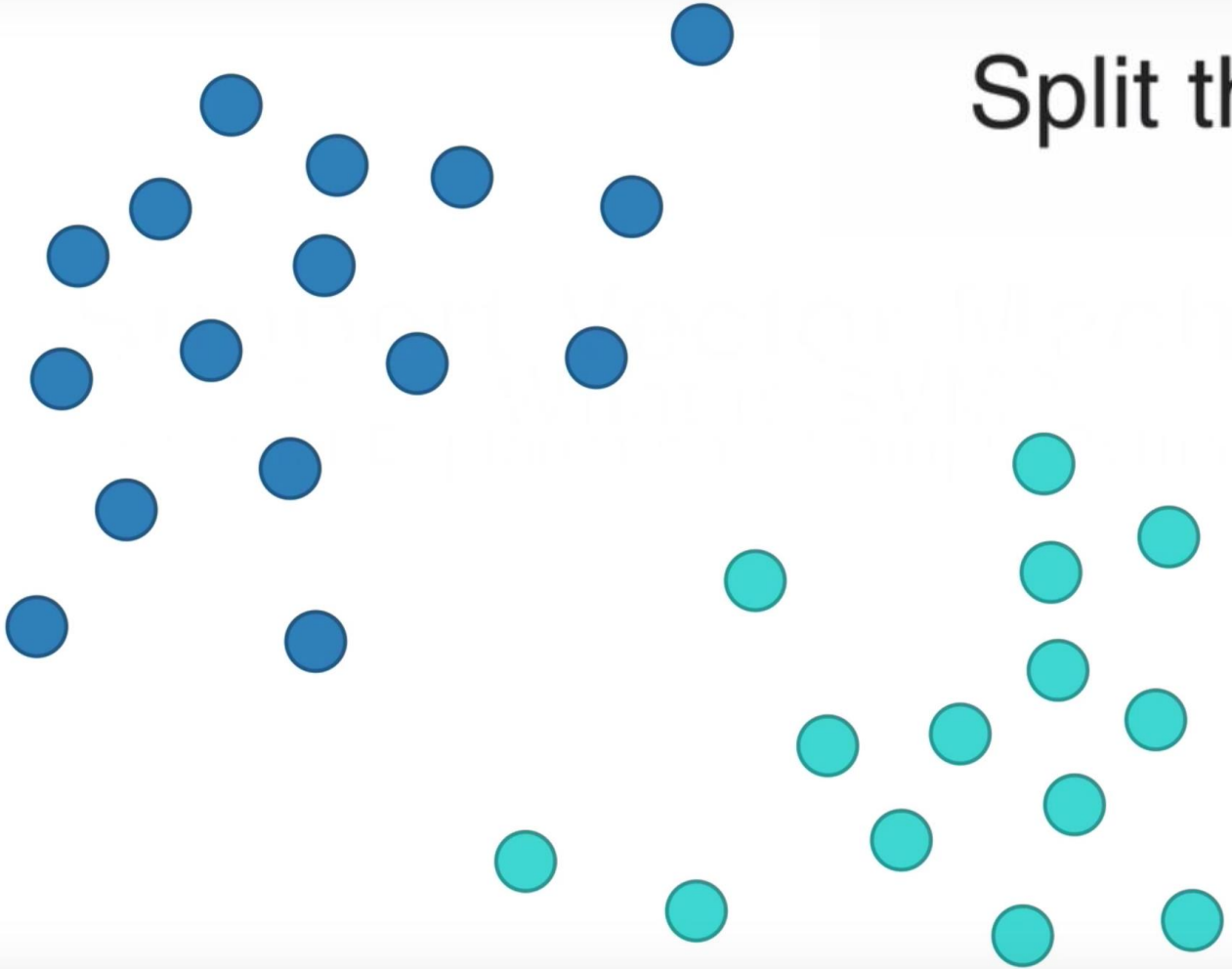
Support Vector Machine

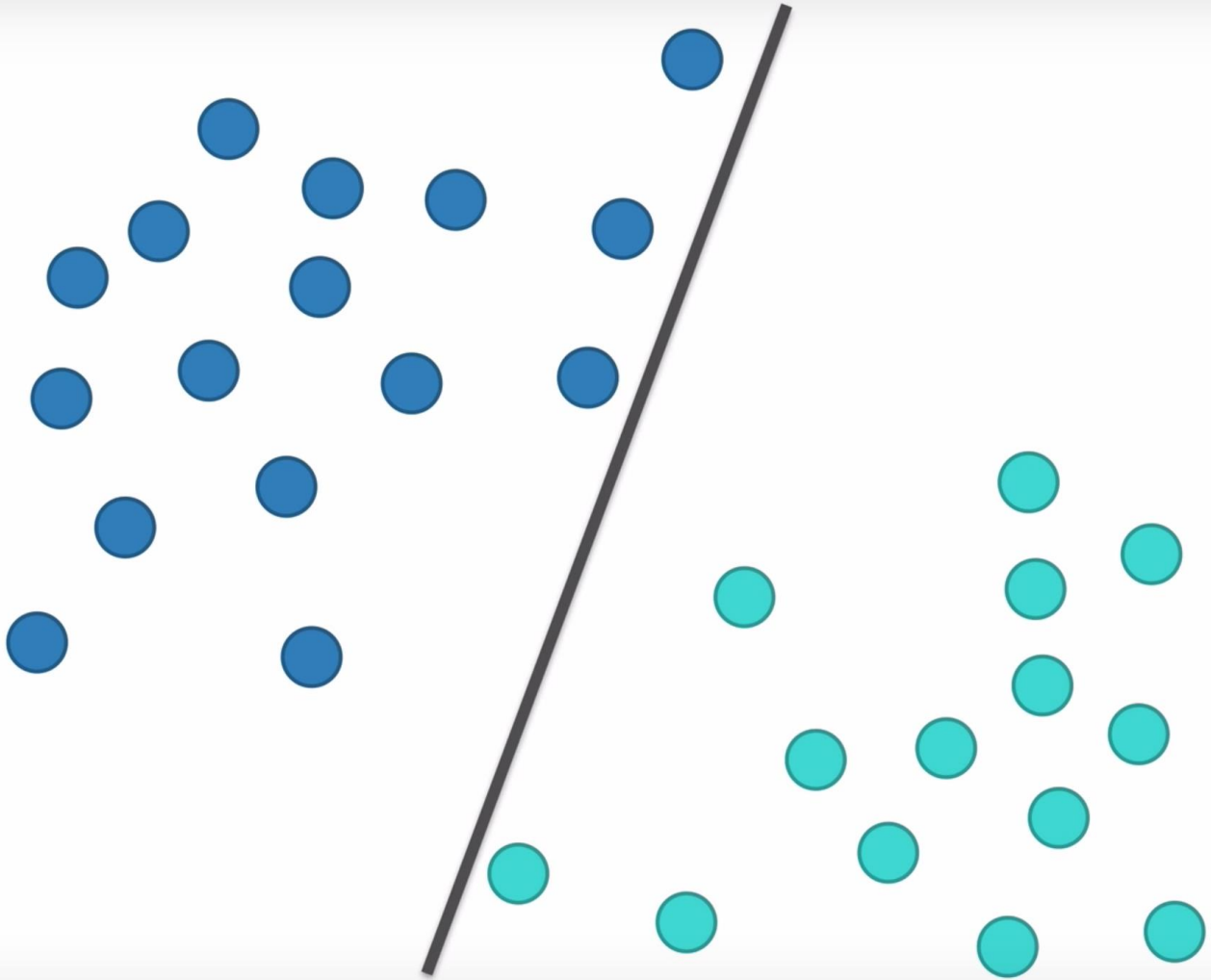
What is SVM?

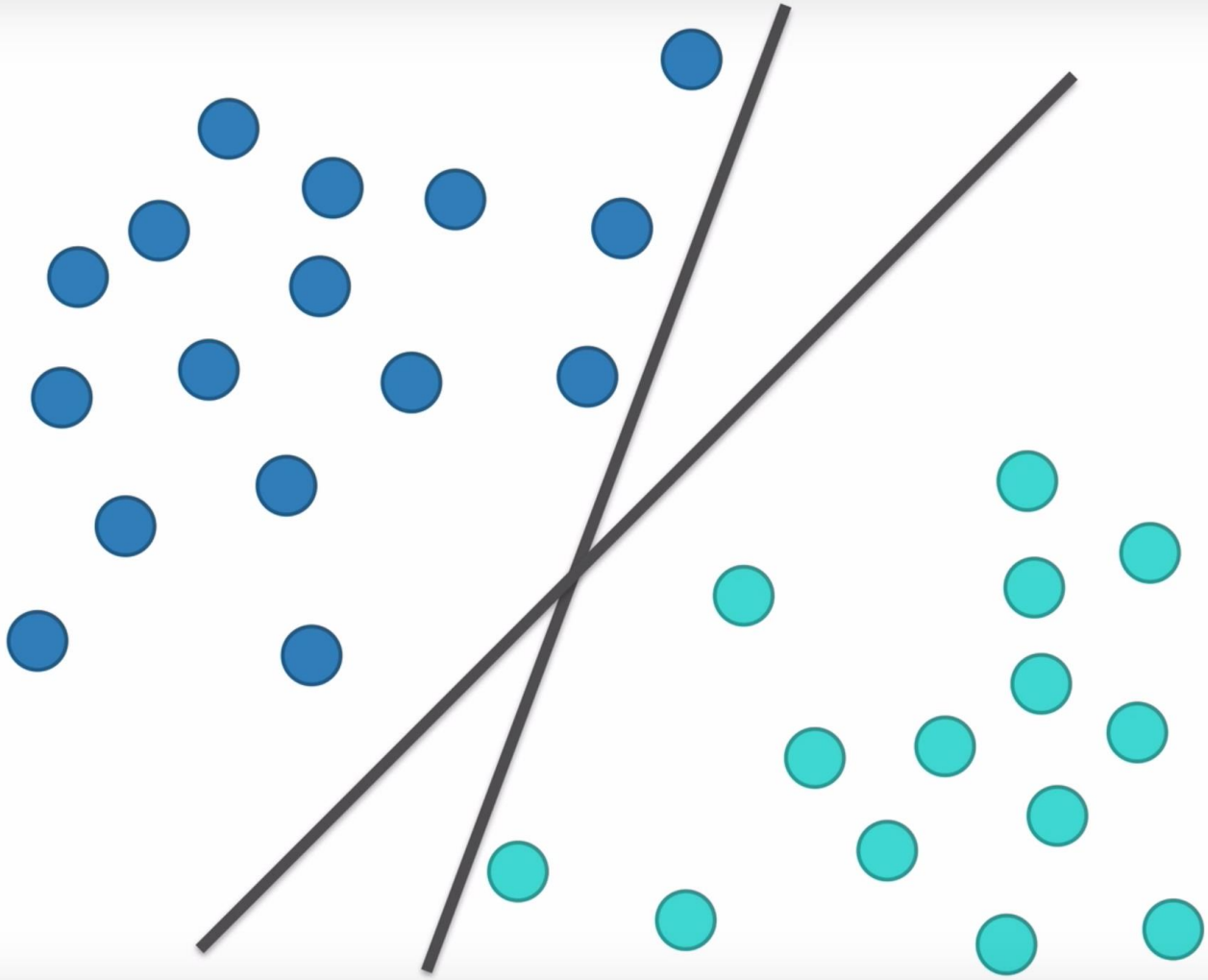
It’s a classification technique.

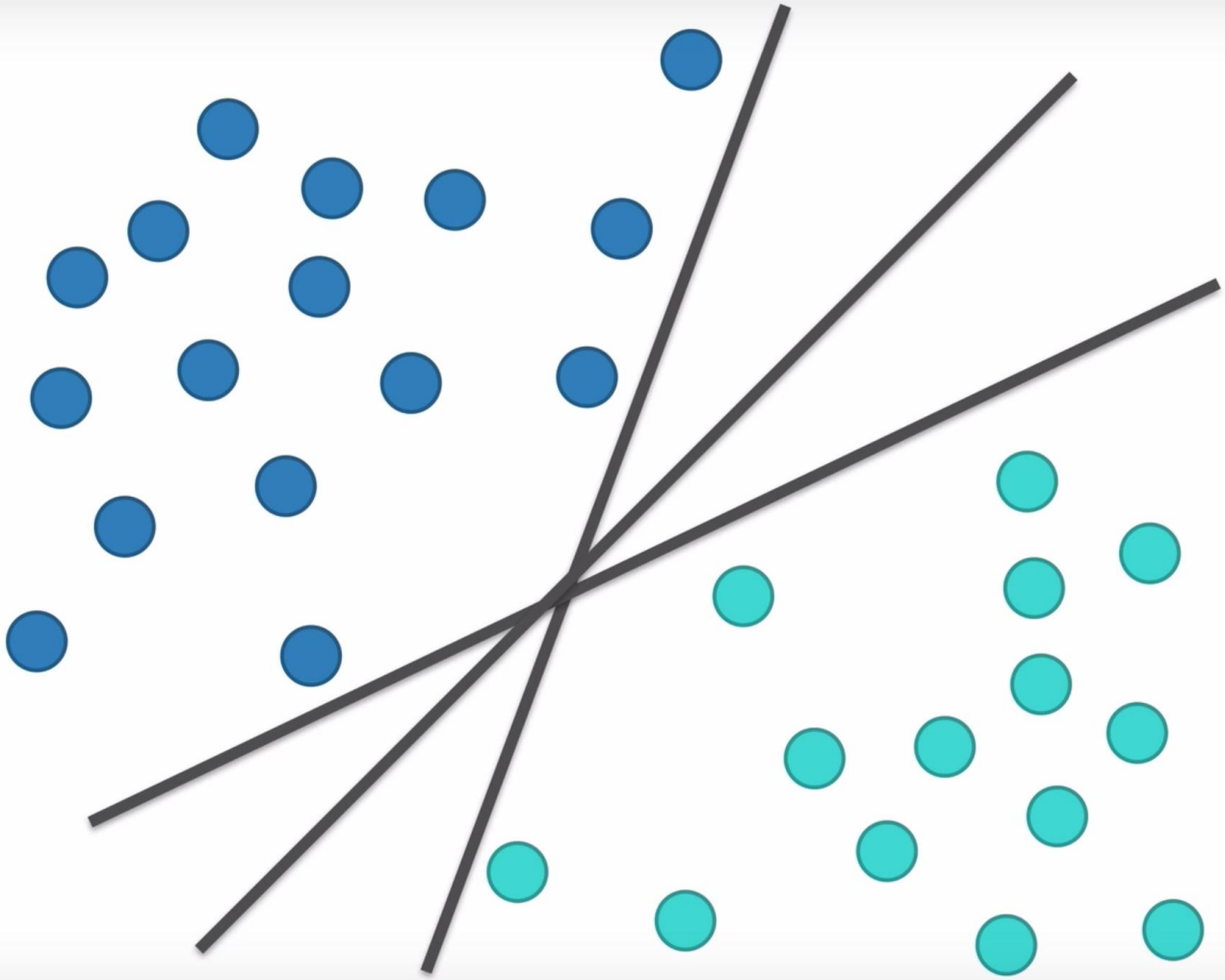


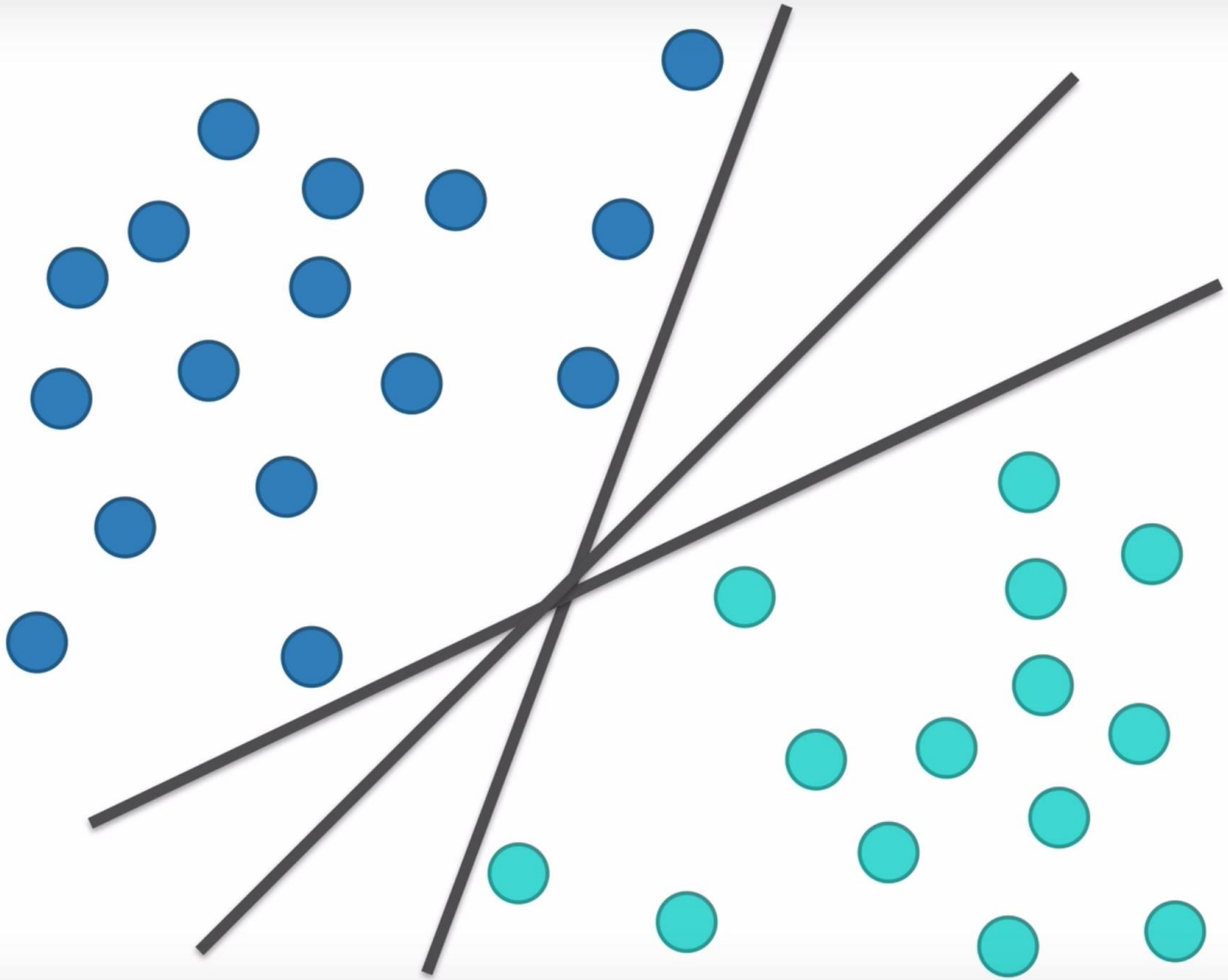
Split the data



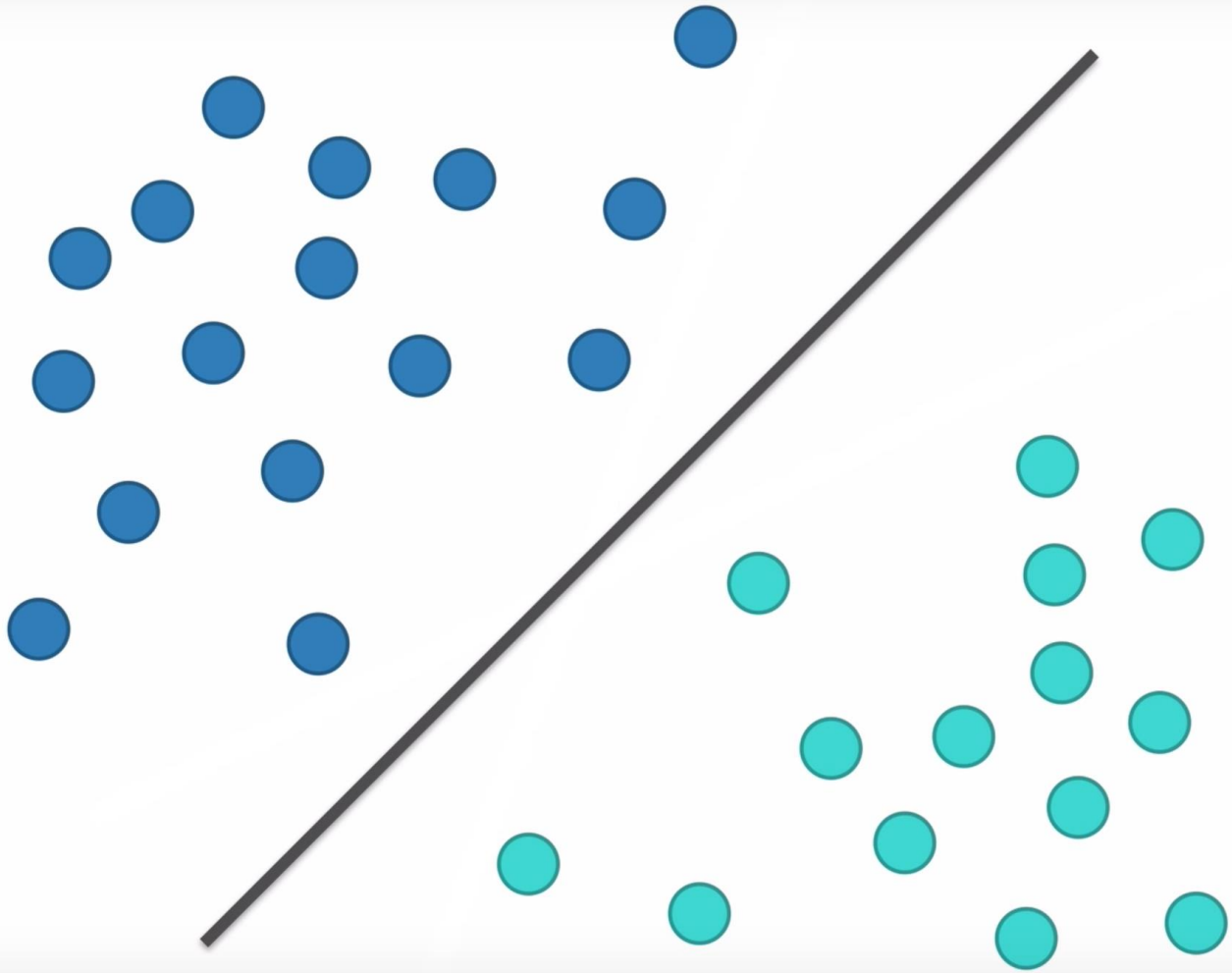

Split the data


Split the data

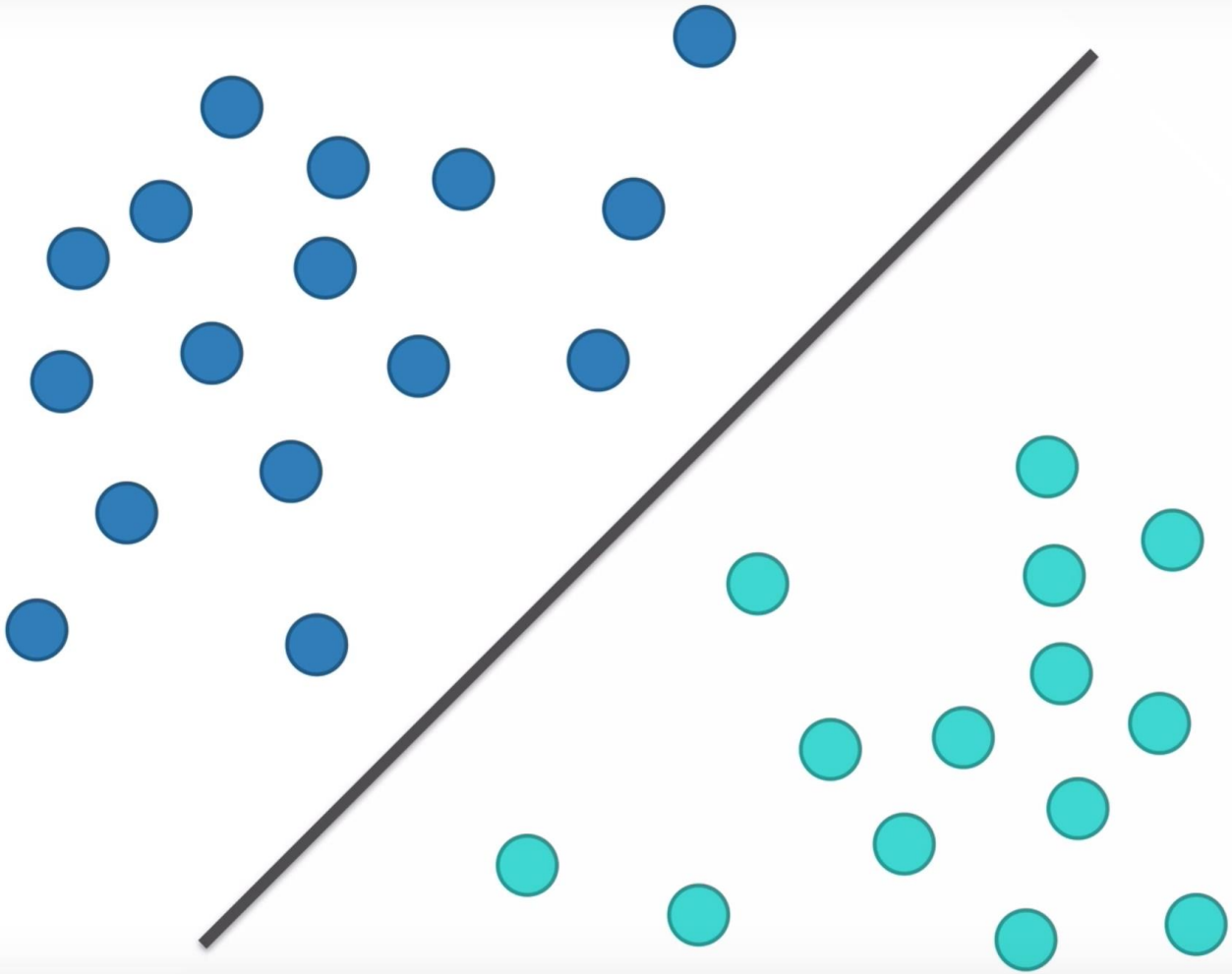

Split the data



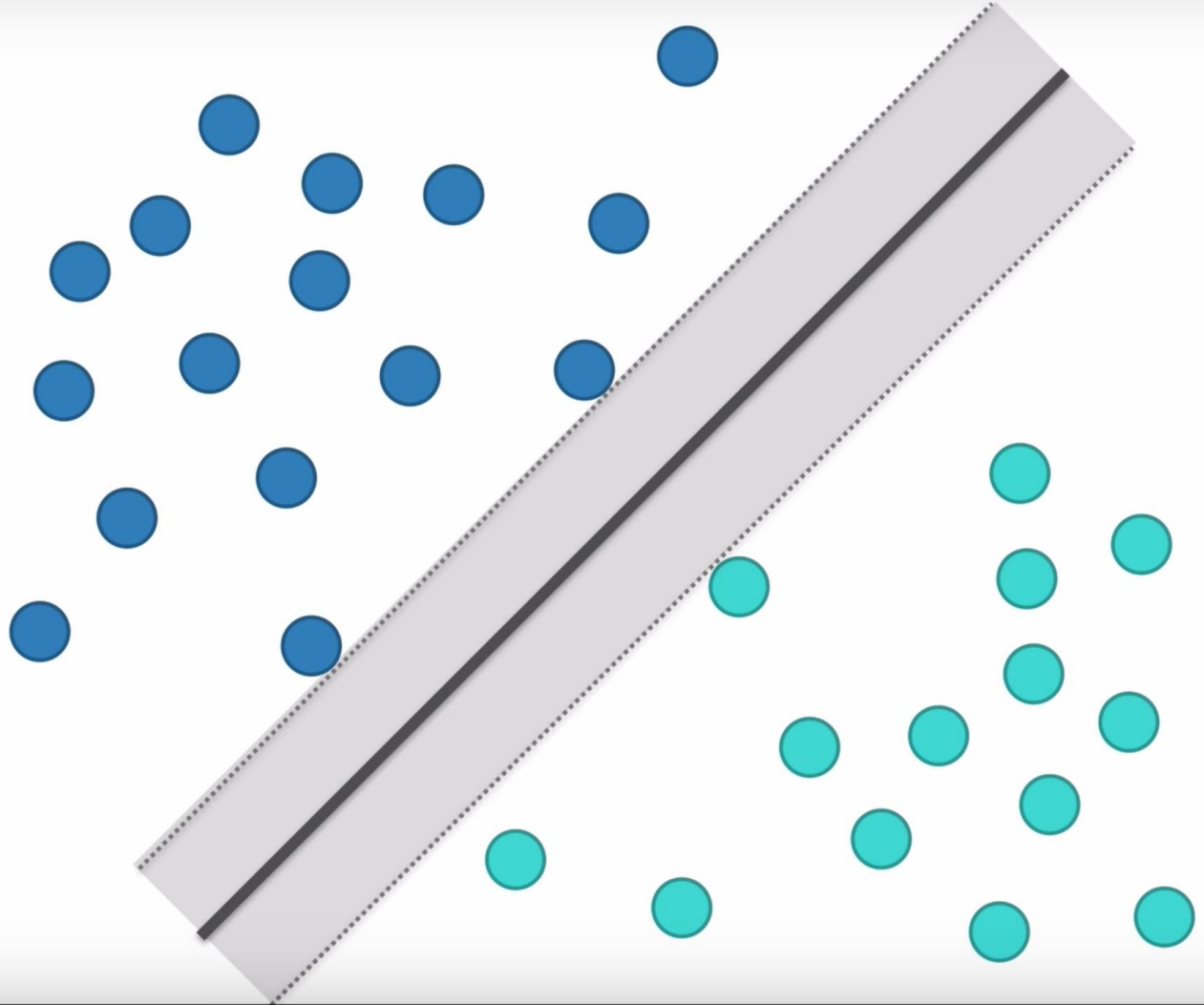
Split the data
in the best possible way



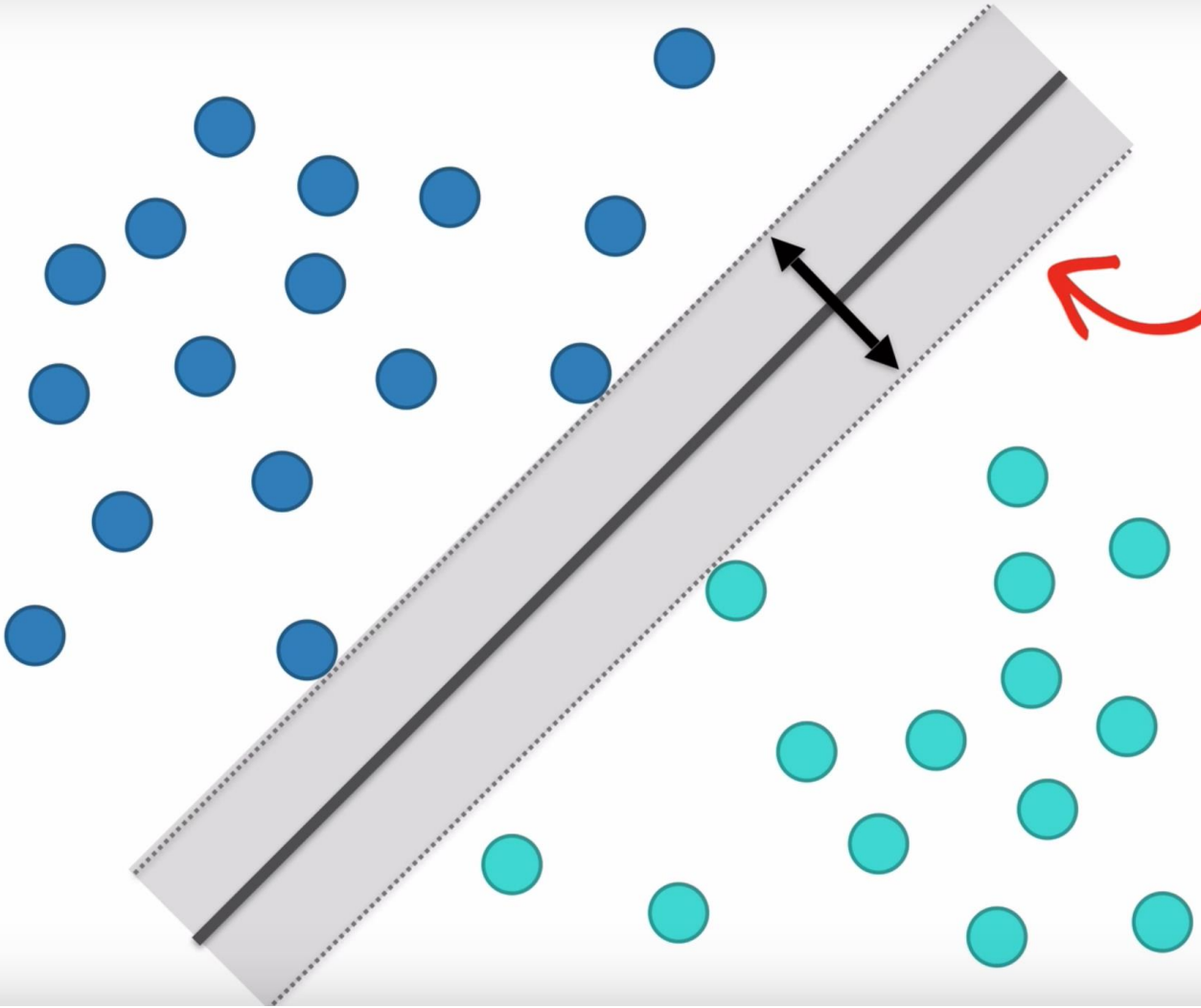
Split the data
in the best possible way



Why is this the best split?

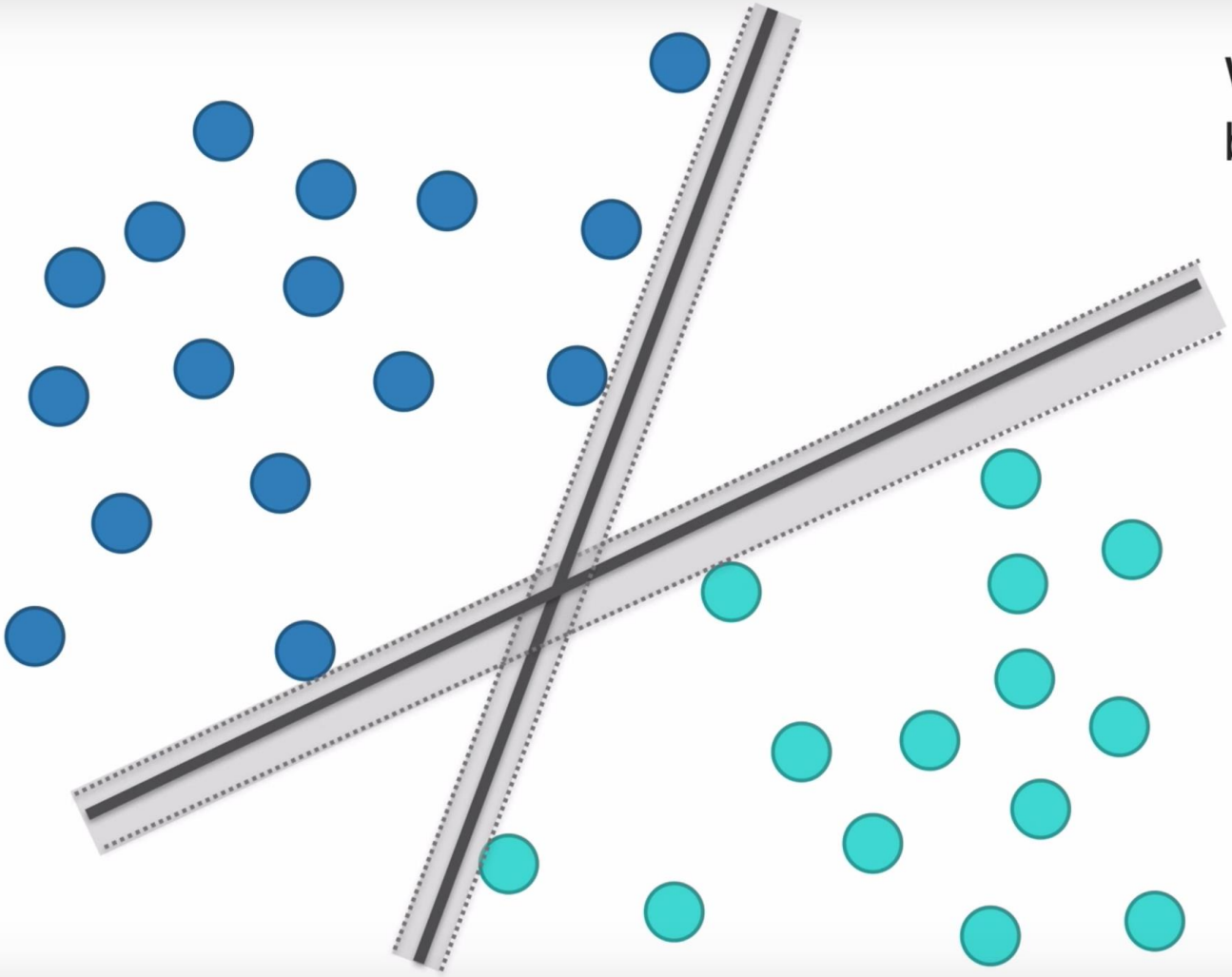


Why is this the best split?

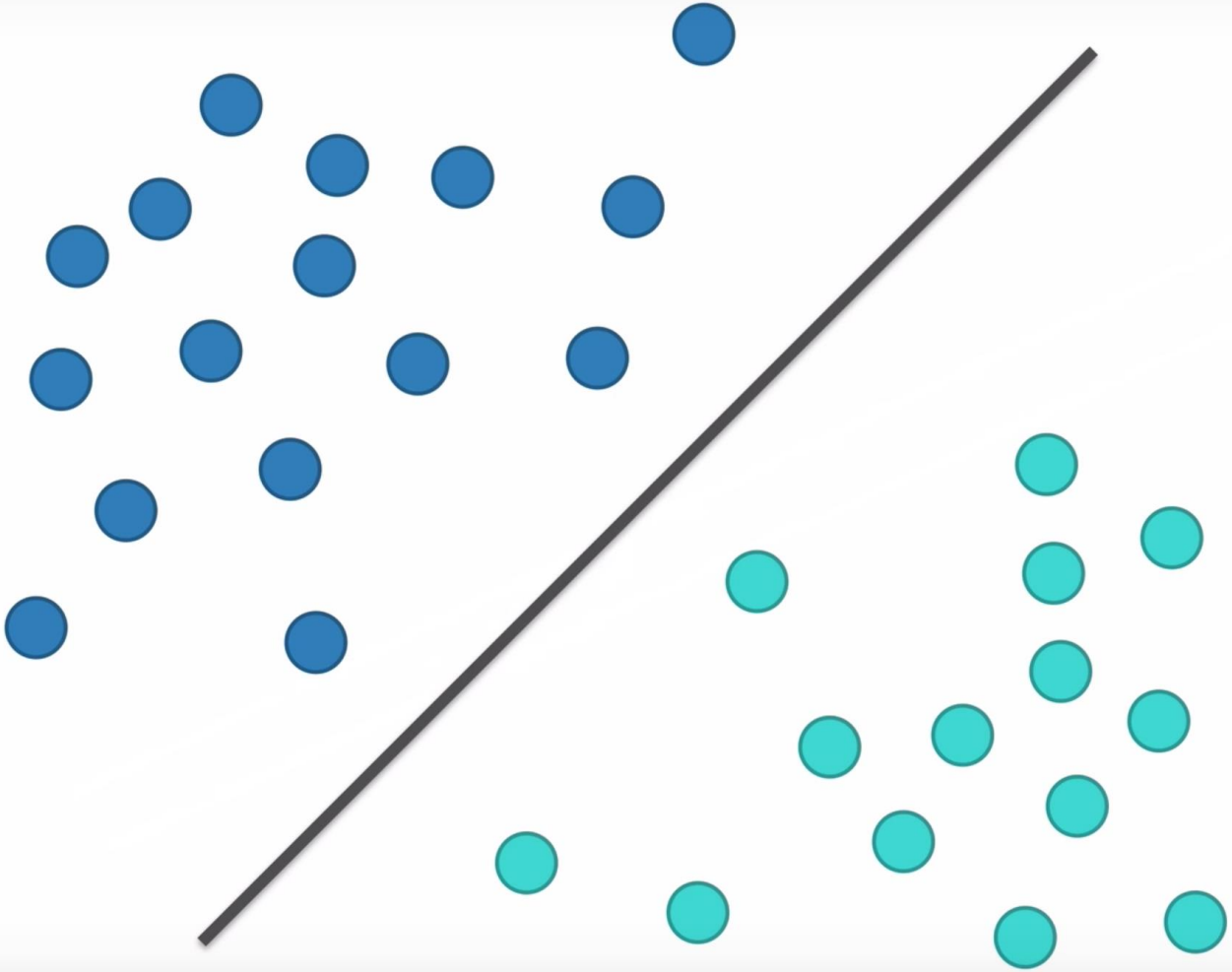


Why is this the best split?

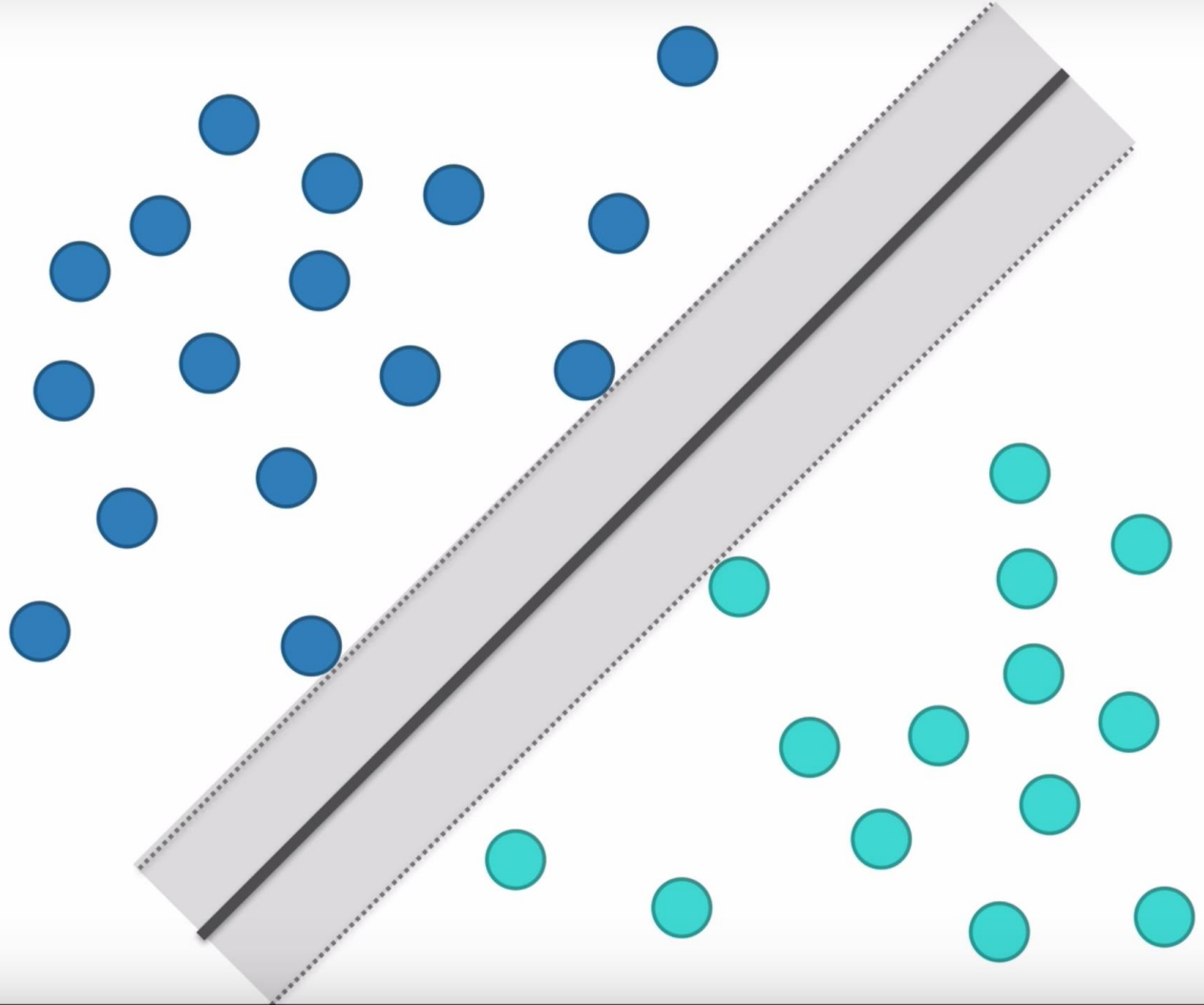
This is the widest road that separates the two groups



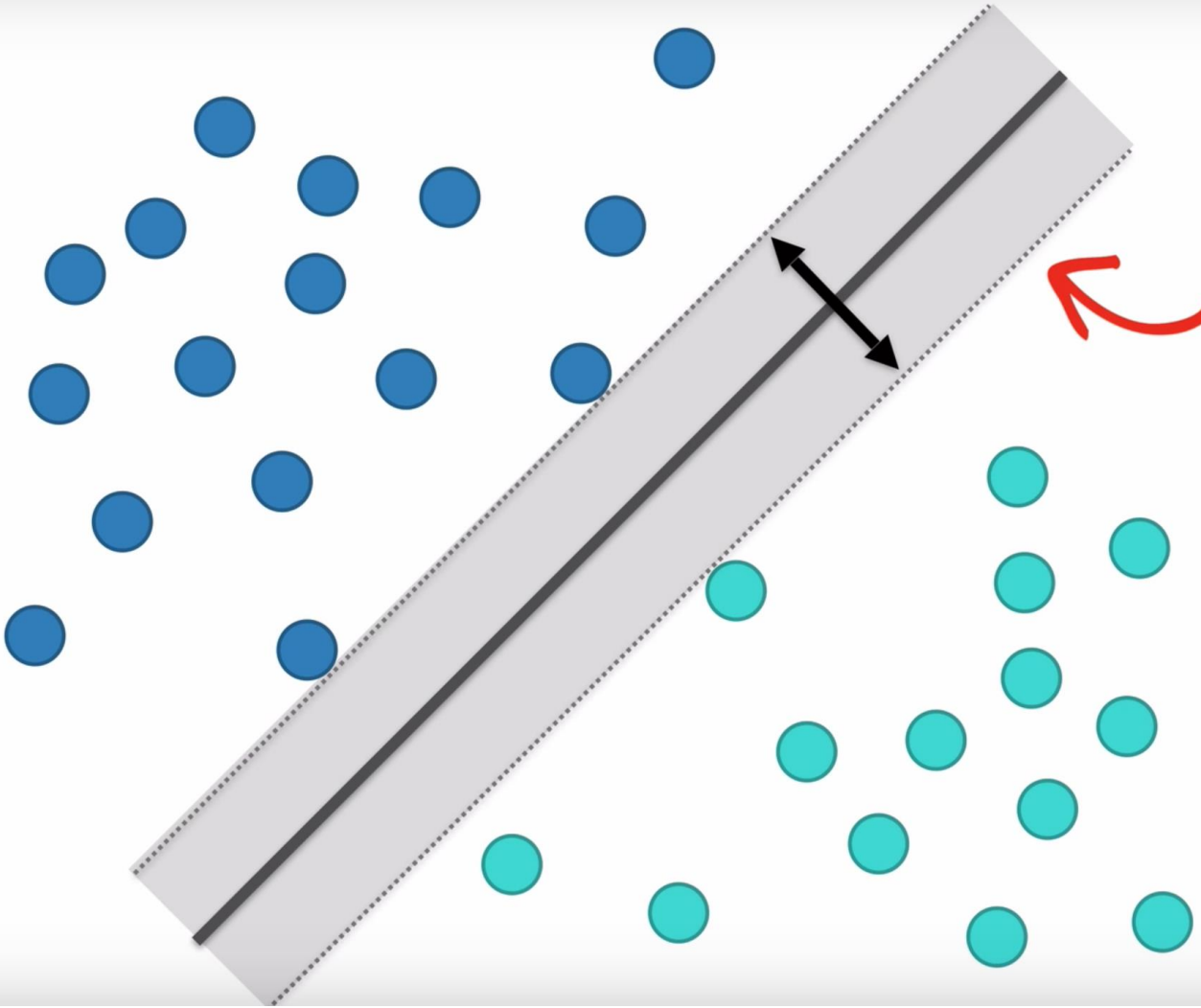
Why is this the best split?



Why is this the best split?

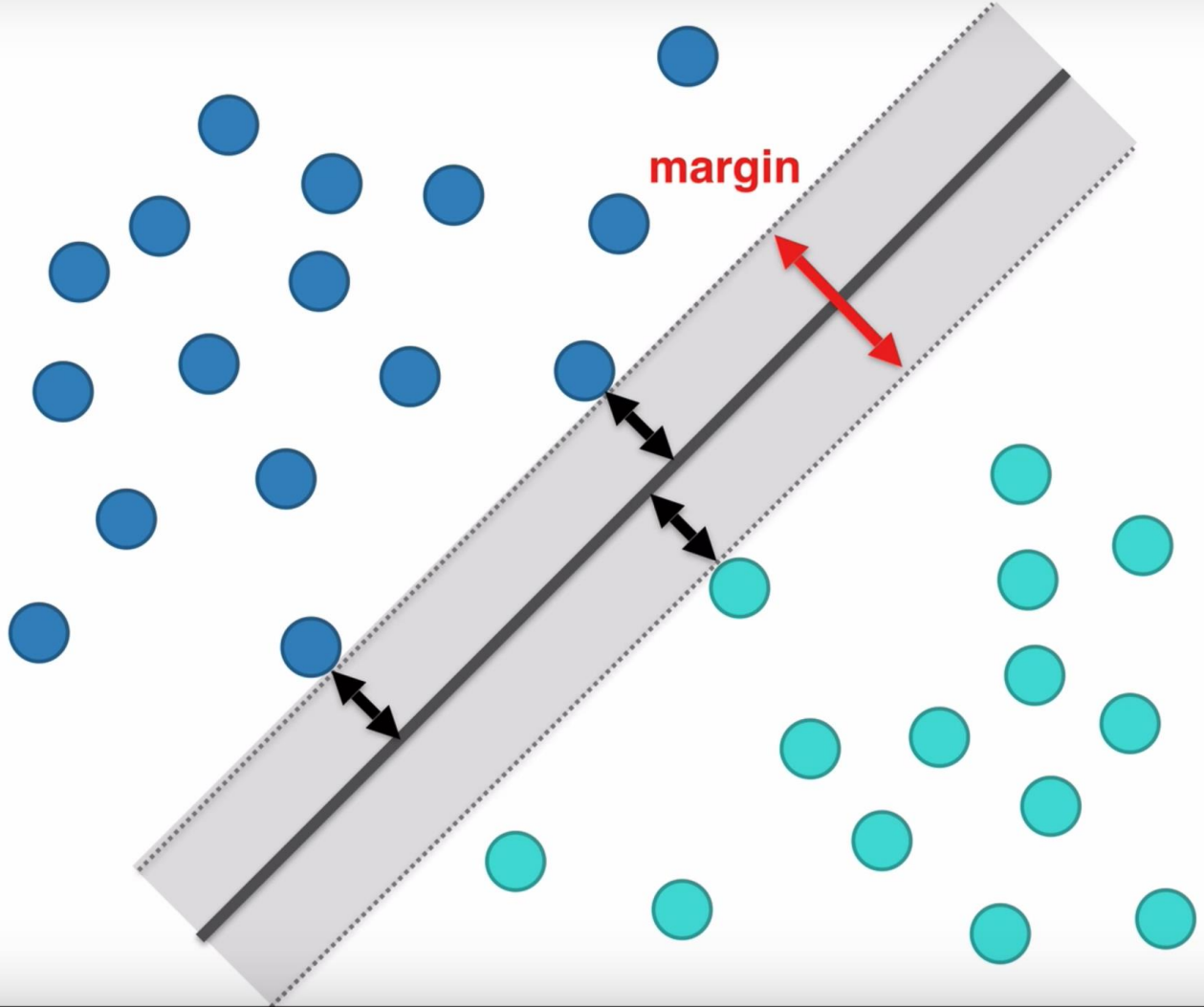


Why is this the best split?



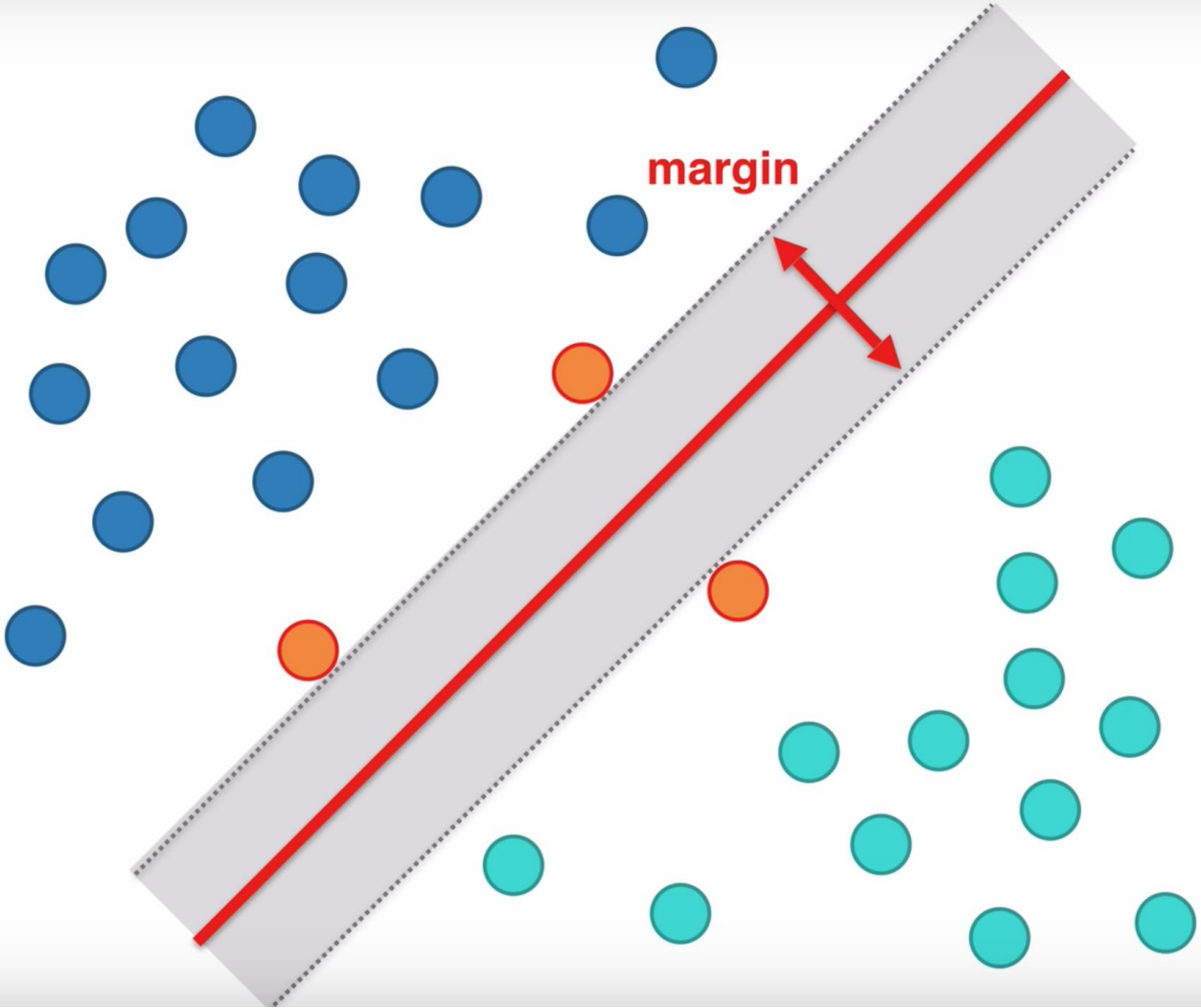
Why is this the best split?

This is the widest road that separates the two groups



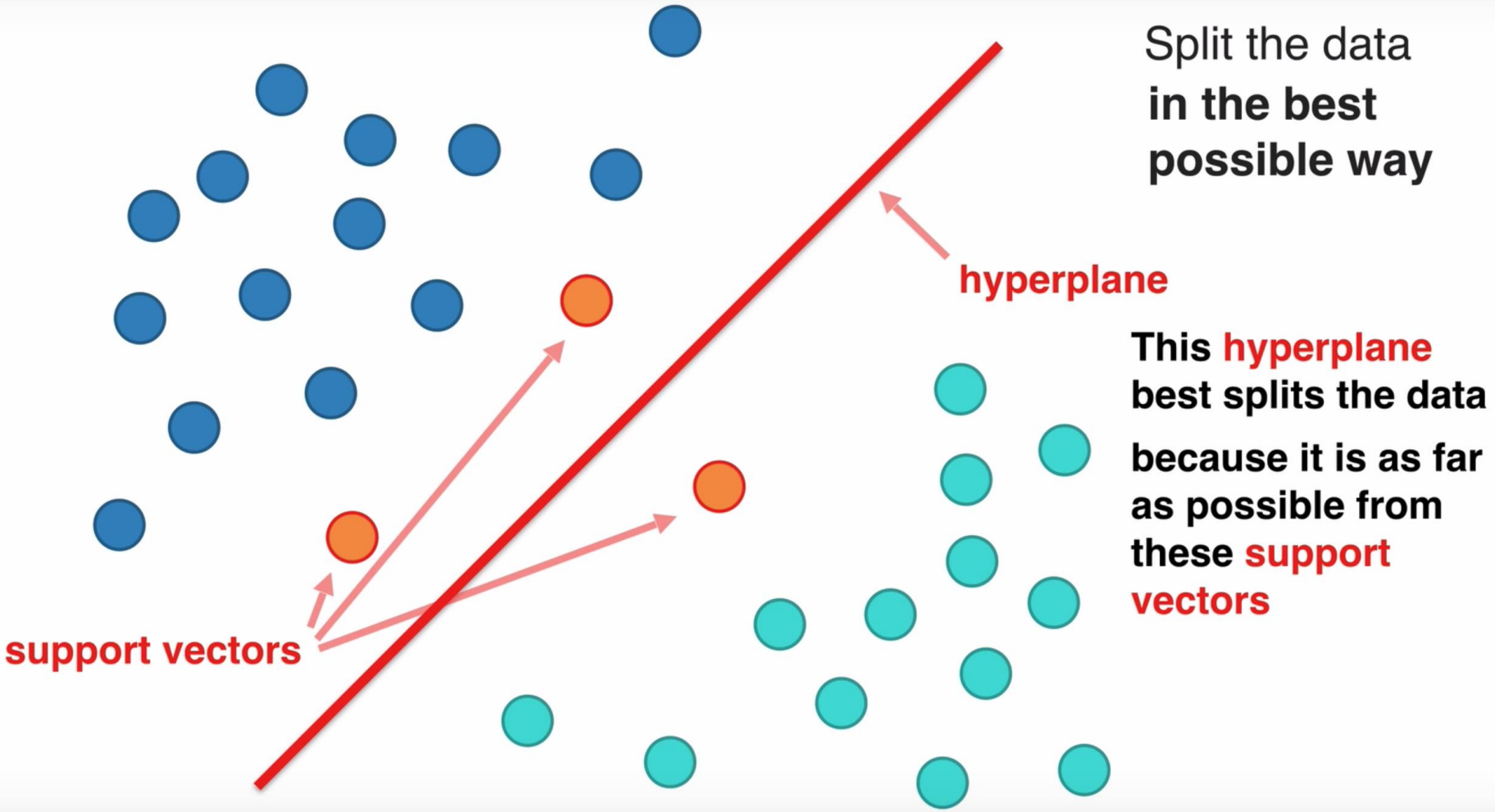
Why is this the best split?

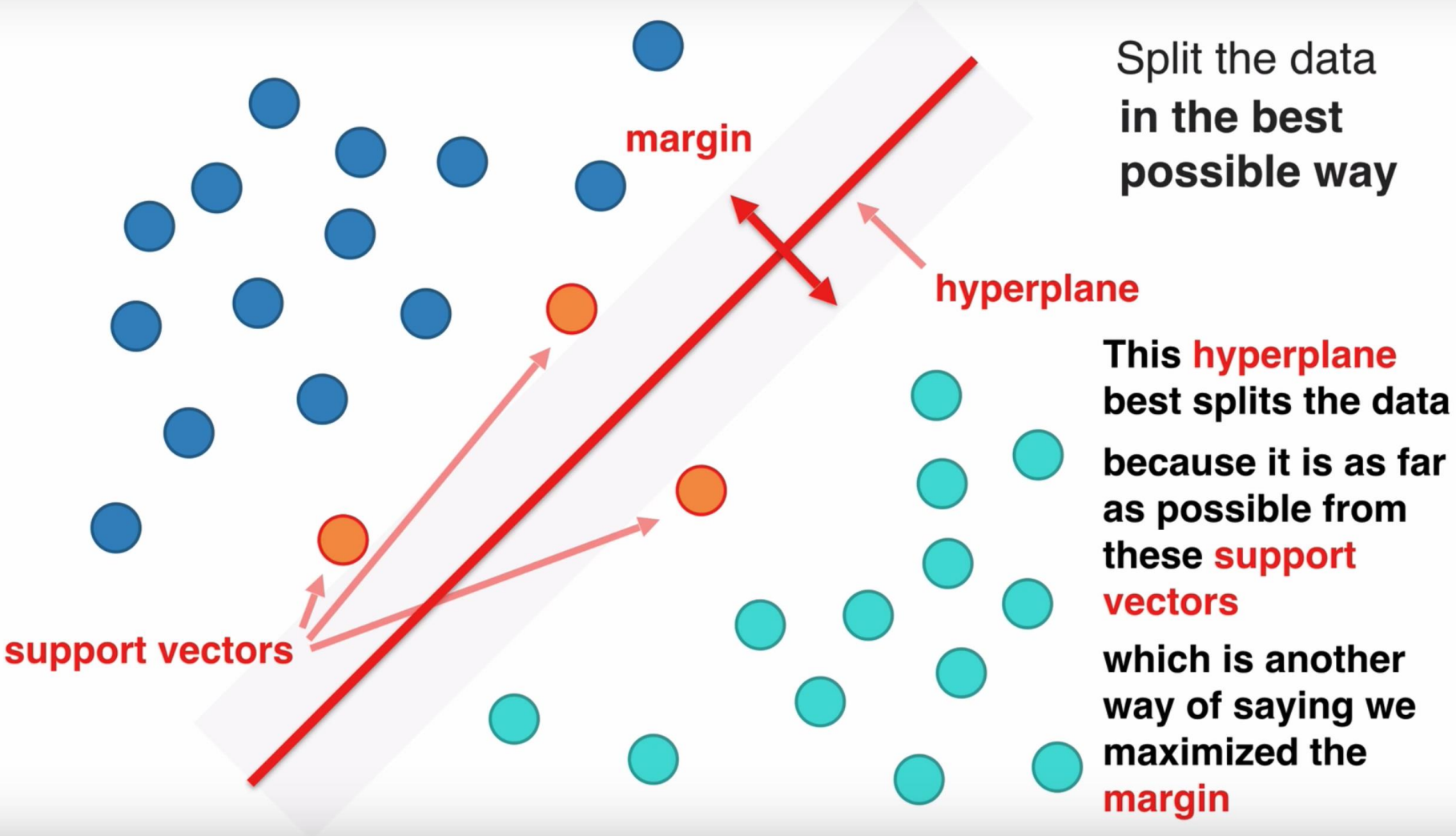
The distance between the points and the line are as far as possible

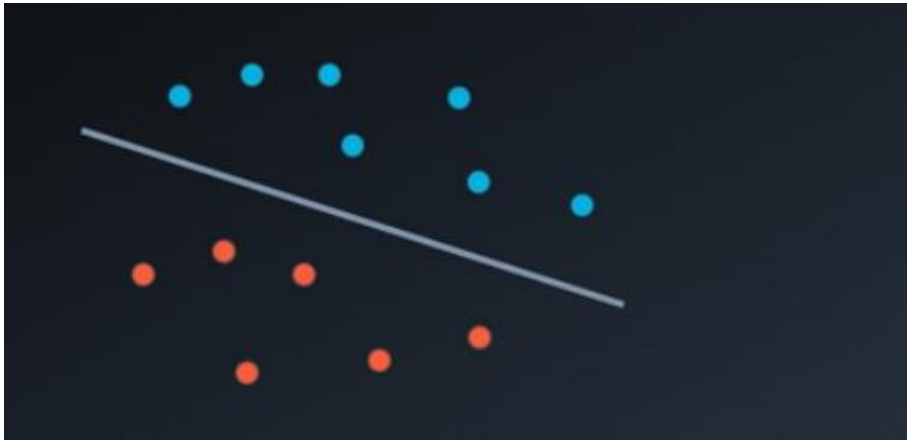


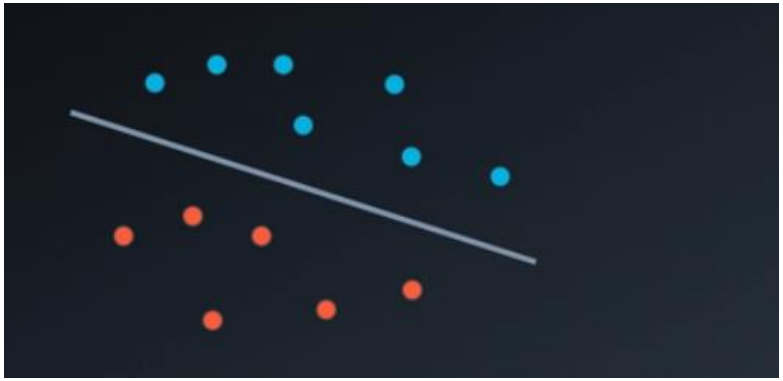
Why is this the best split?

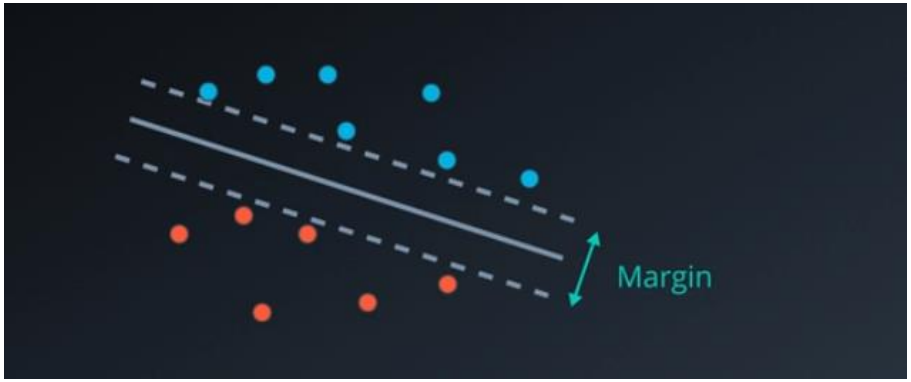
The distance between the support vectors and the hyperplane are as far as possible

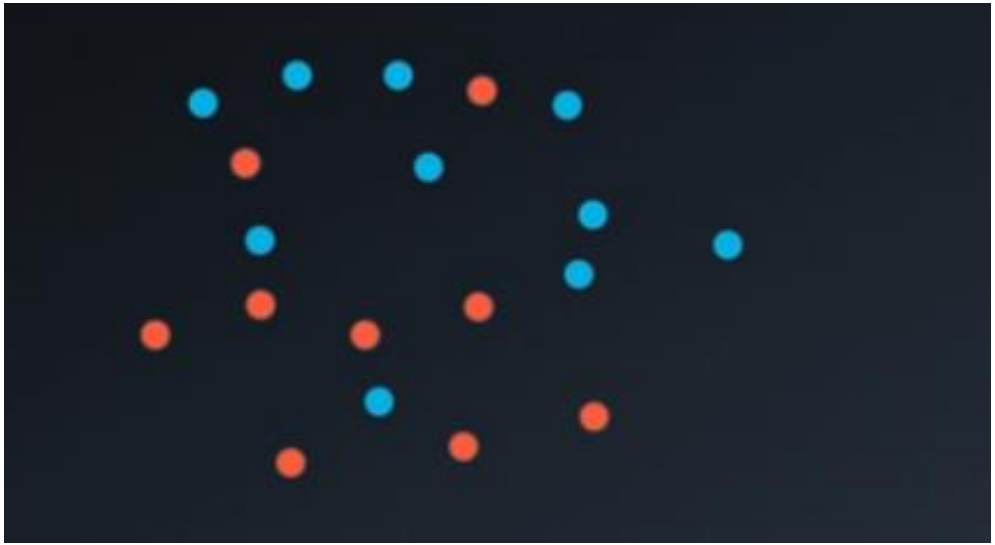

Split the data
in the best
possible way
hyperplane
This hyperplane
best splits the data
because it is as far
as possible from
these support
vectors
support vectors

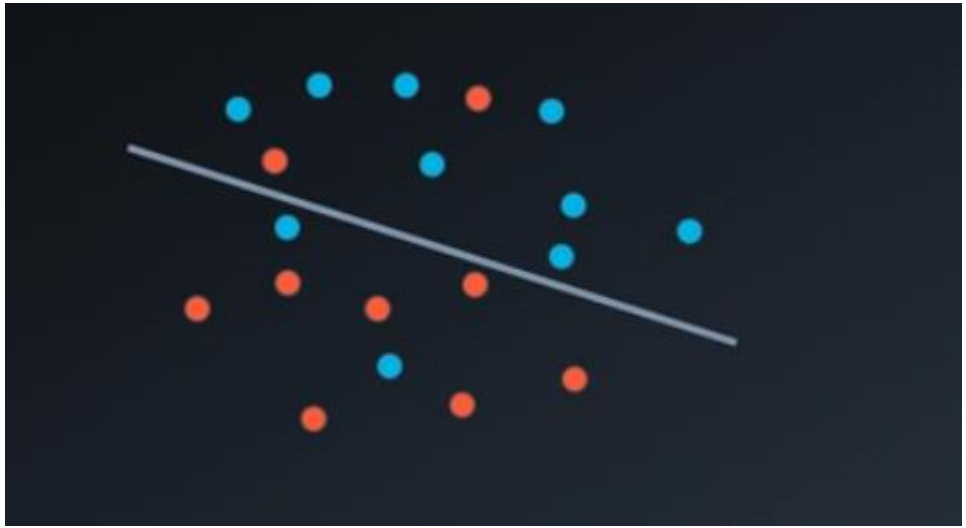

Split the data
in the best
possible way
margin
hyperplane
This hyperplane
best splits the data
because it is as far
as possible from
these support
vectors
which is another
way of saying we
maximized the
margin
support vectors

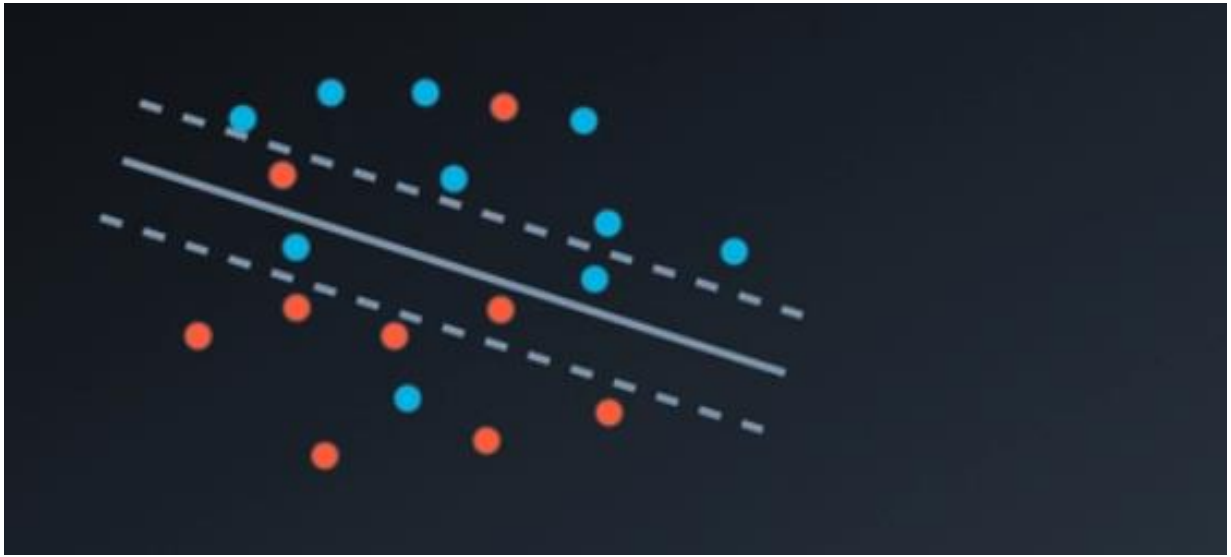


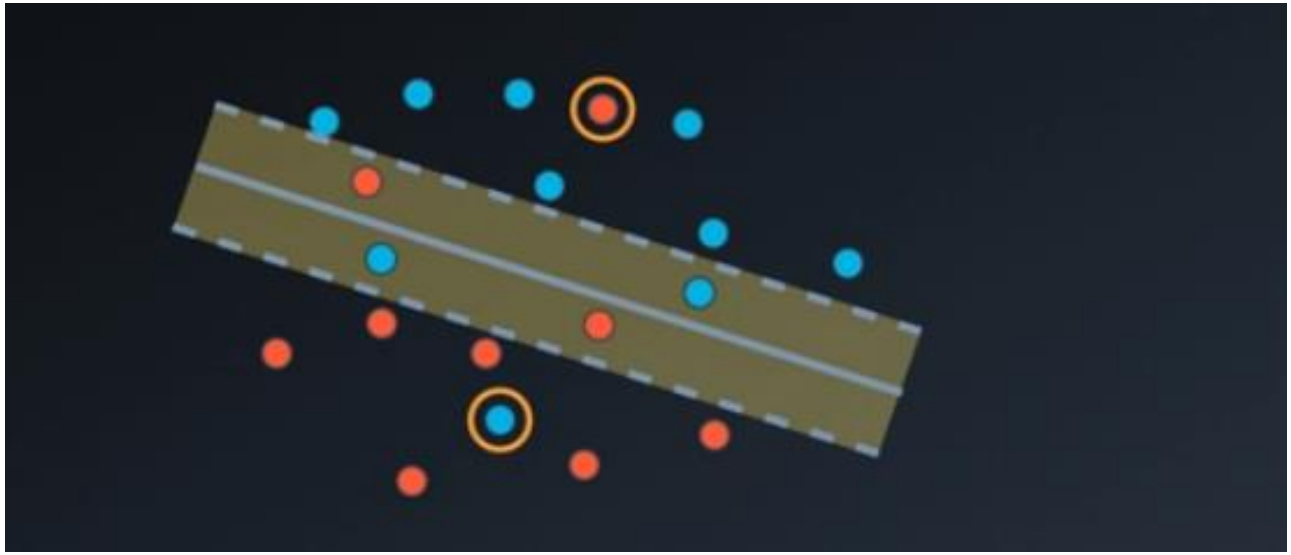


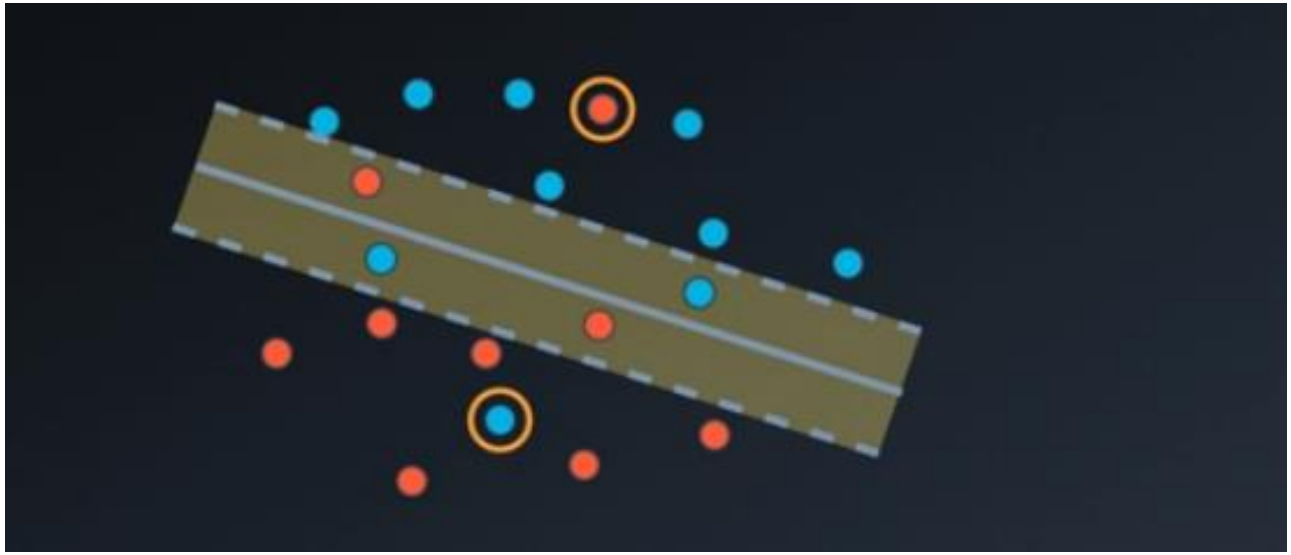

Margin





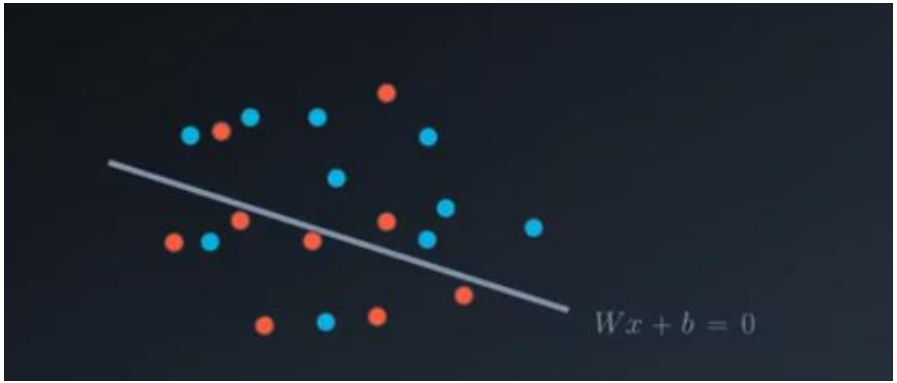




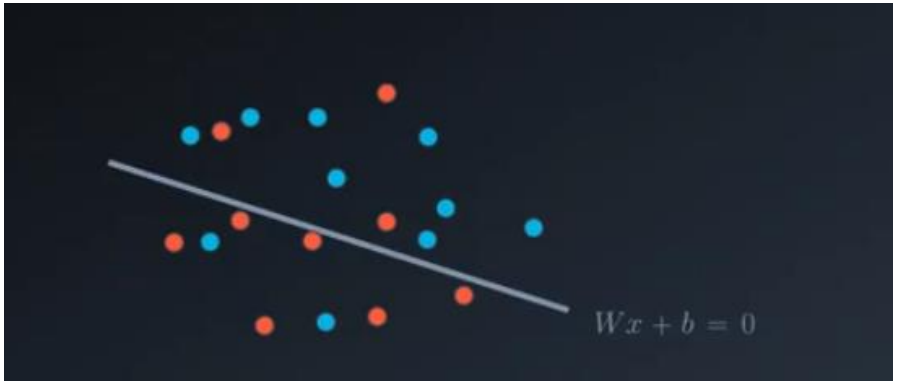


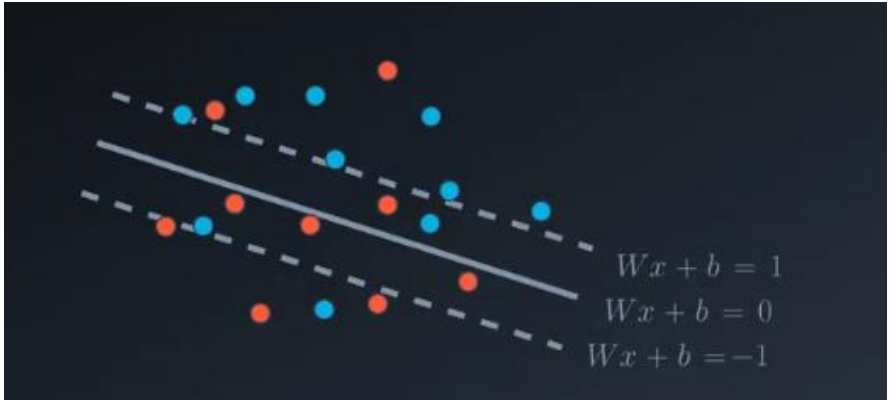
ERROR = CLASSIFICATION ERROR + MARGIN ERROR

Classification Error

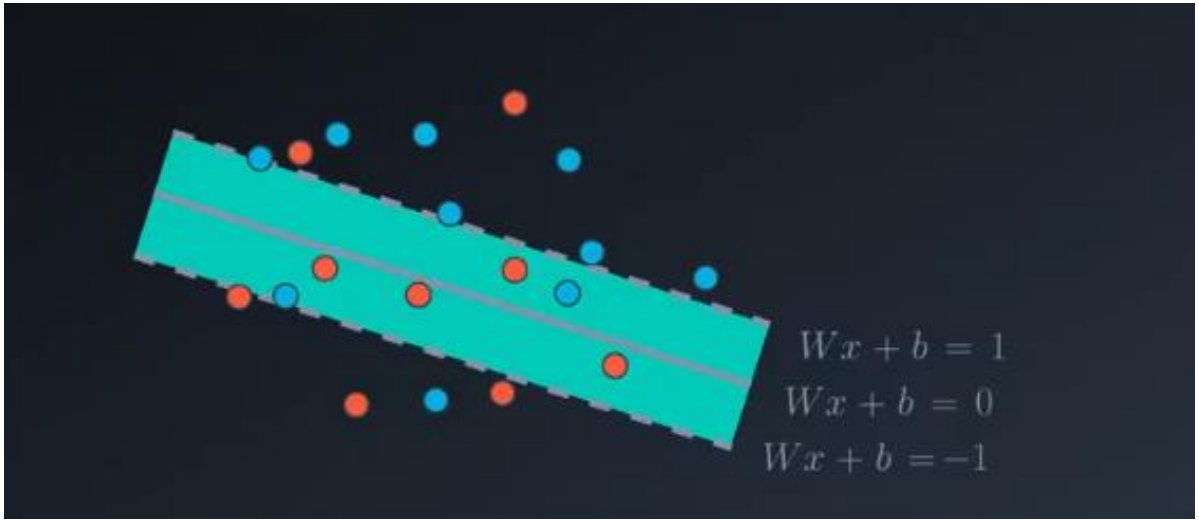


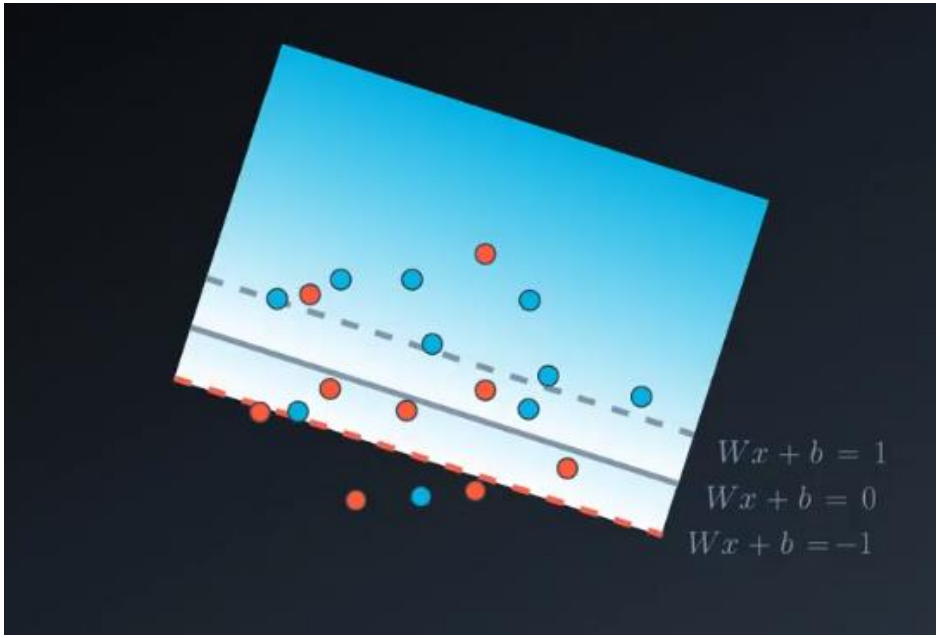
Classification Error

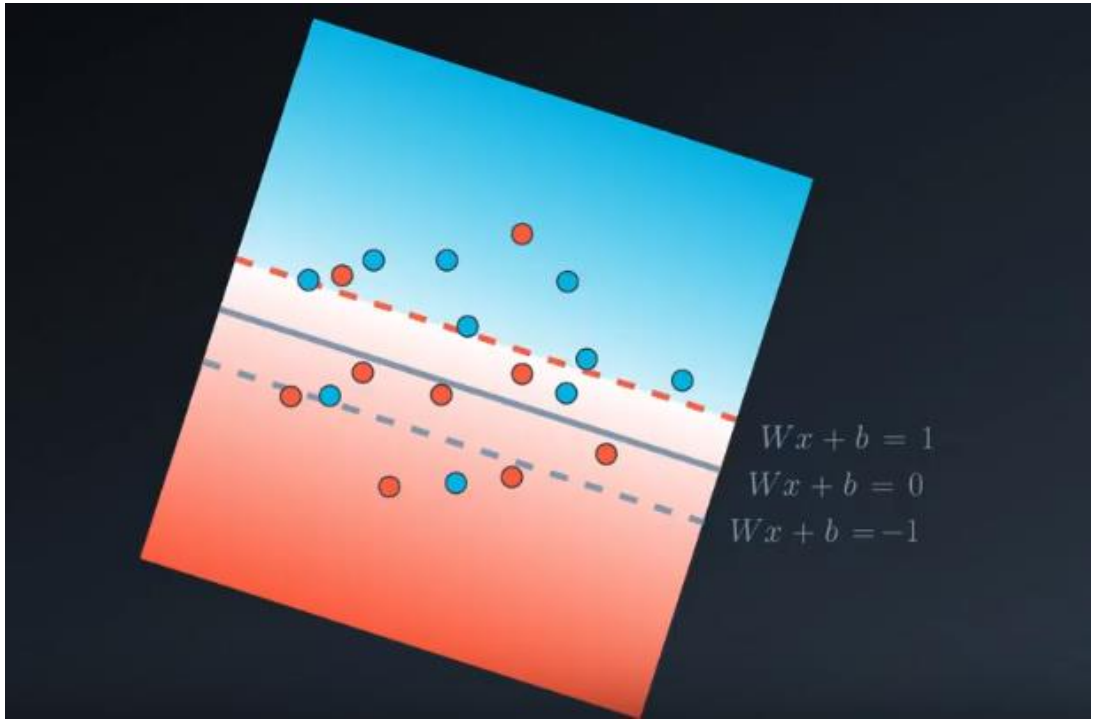


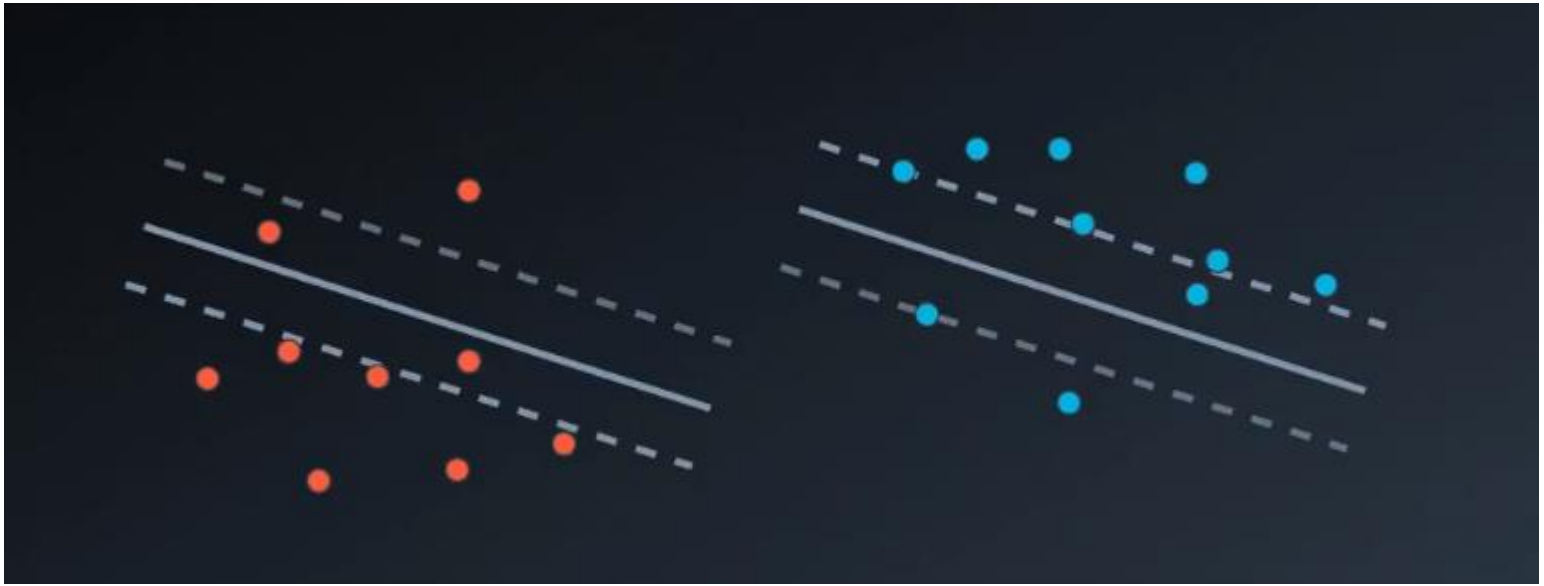


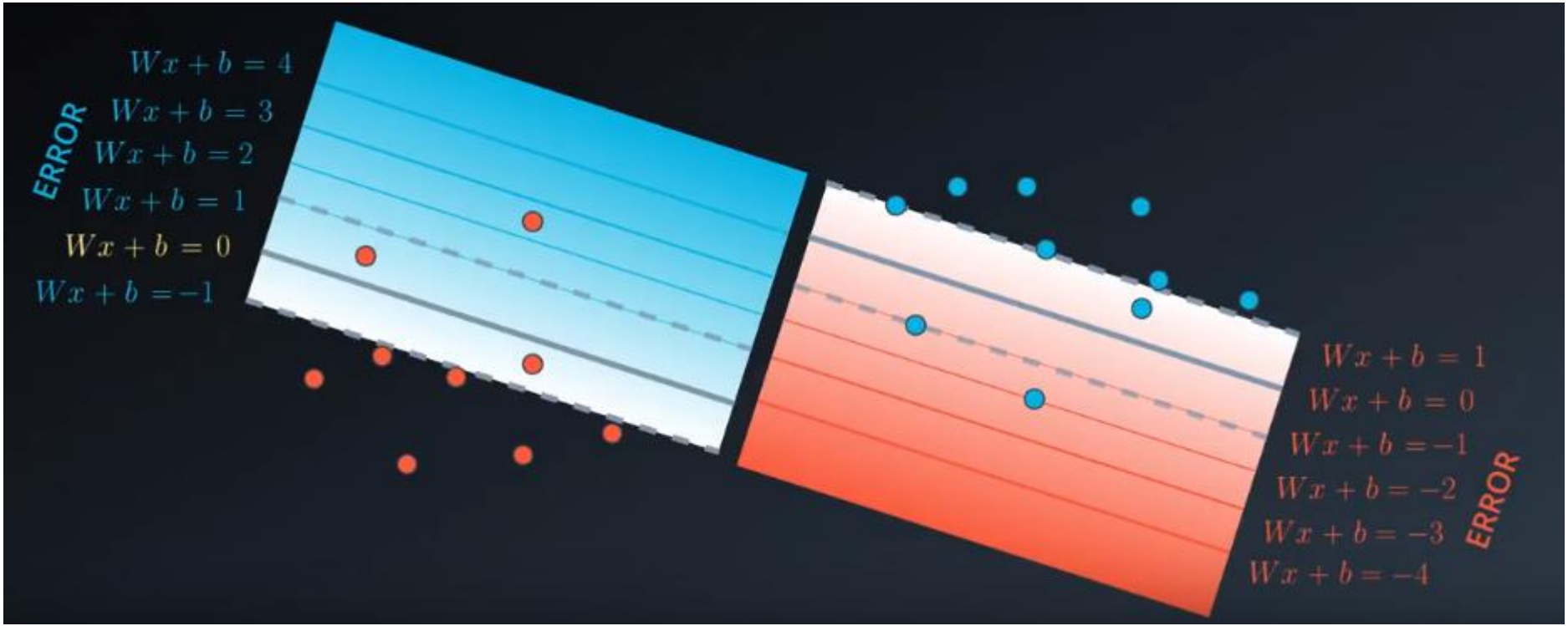
Classification Error

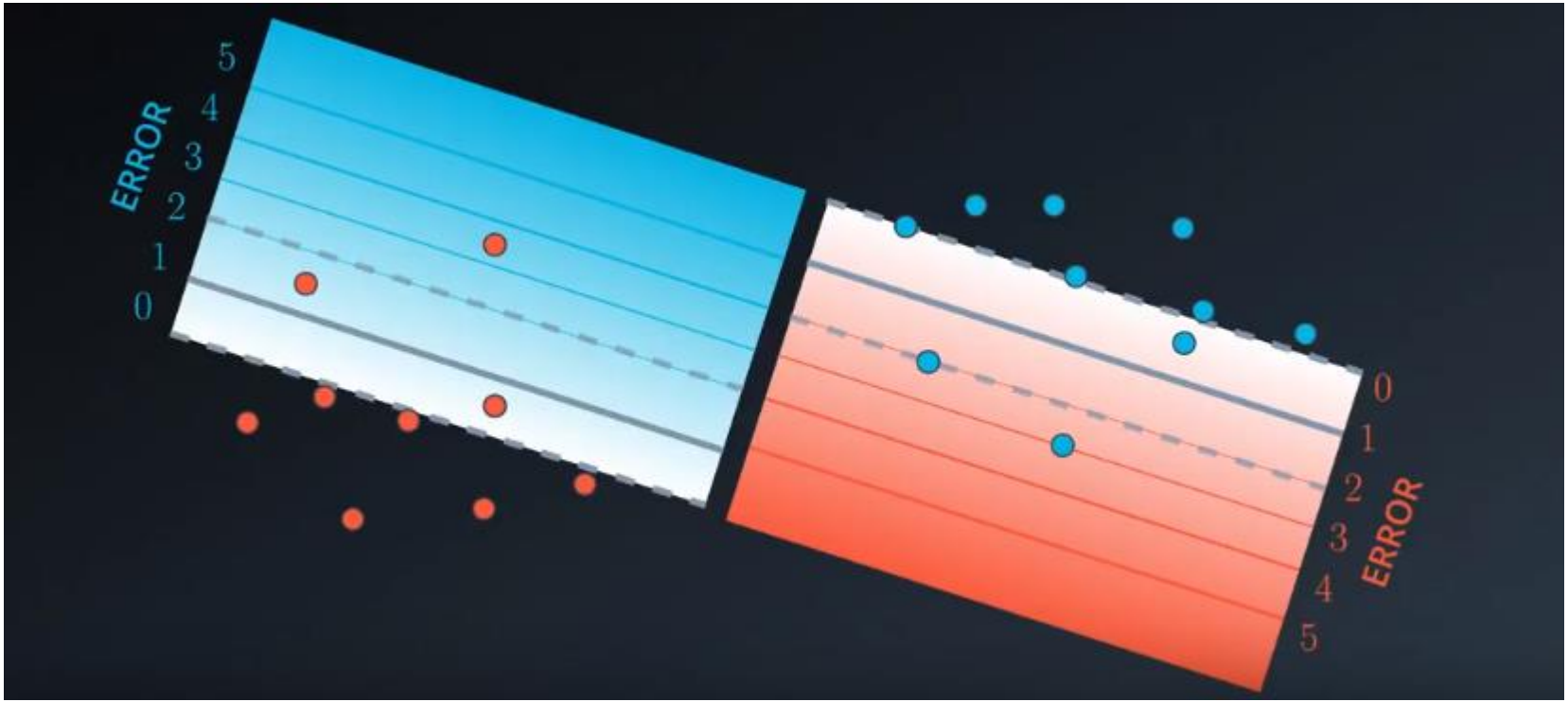


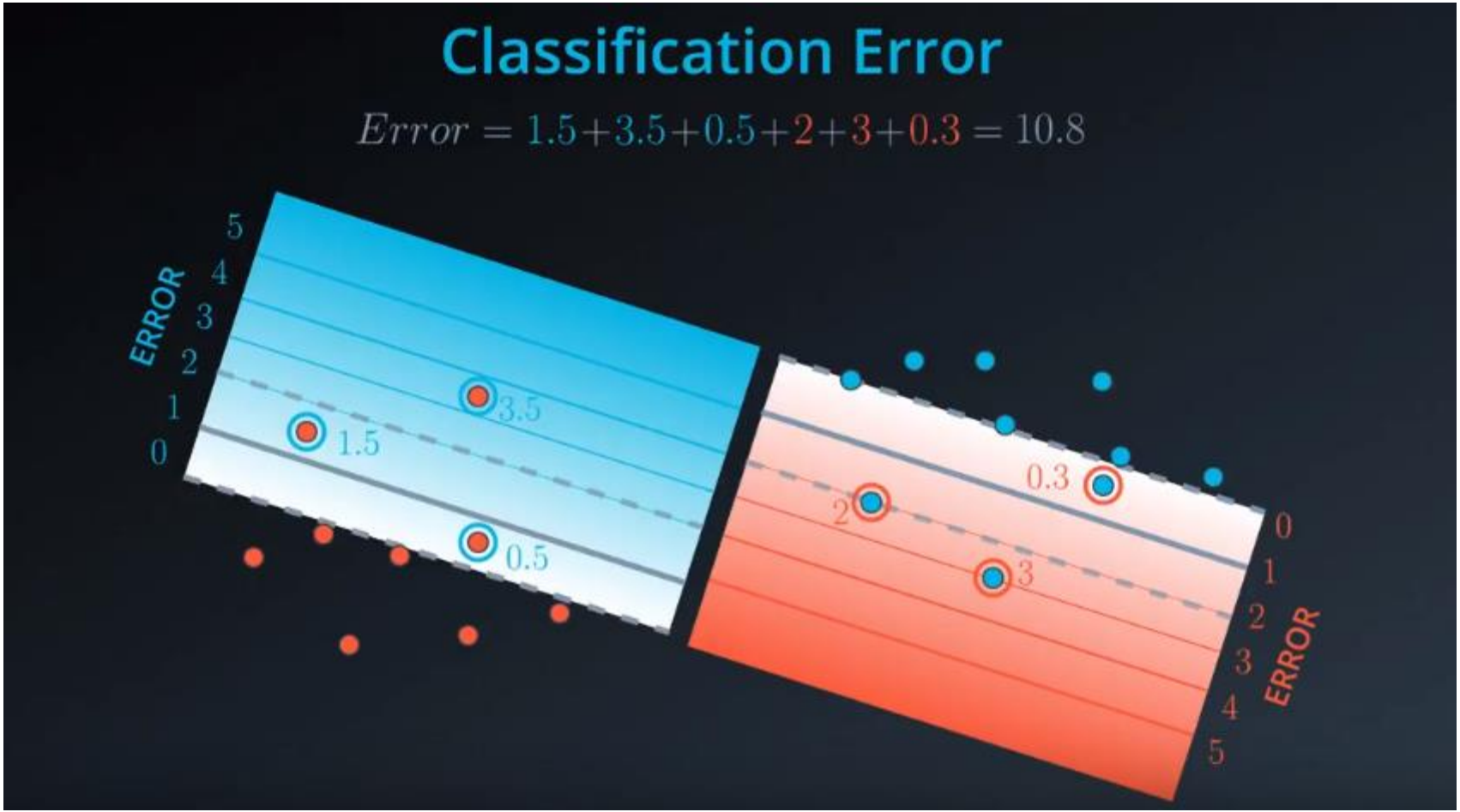

$Wx + b = 1$
$Wx + b = 0$
$Wx + b = -1$


$Wx + b = 1$
$Wx + b = 0$
$Wx + b = -1$

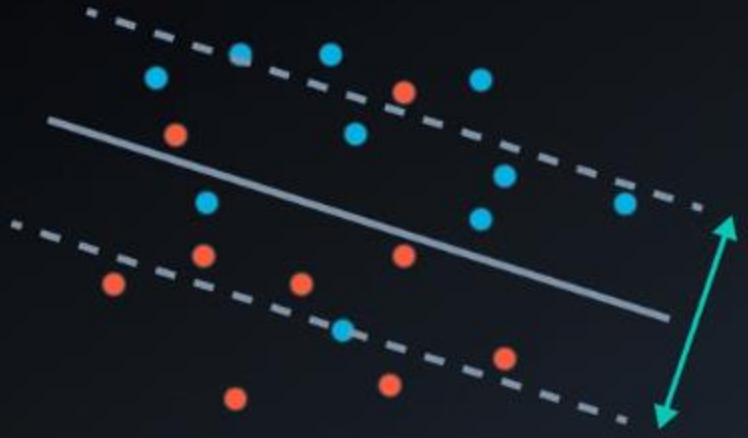



$Wx + b = 4$
$Wx + b = 3$
$Wx + b = 2$
$Wx + b = 1$
$Wx + b = 0$
$Wx + b = -1$
ERROR
$Wx + b = 1$
$Wx + b = 0$
$Wx + b = -1$
$Wx + b = -2$
$Wx + b = -3$
$Wx + b = -4$
ERROR


ERROR
5
4
3
2
1
0
0
1
2
3
4
5
ERROR


Classification Error
$Error = 1.5+3.5+0.5+2+3+0.3 = 10.8$
ERROR
0
1
2
3
4
5
1.5
3.5
0.5
2
3
0.3
ERROR
0
1
2
3
4
5

Margin Error



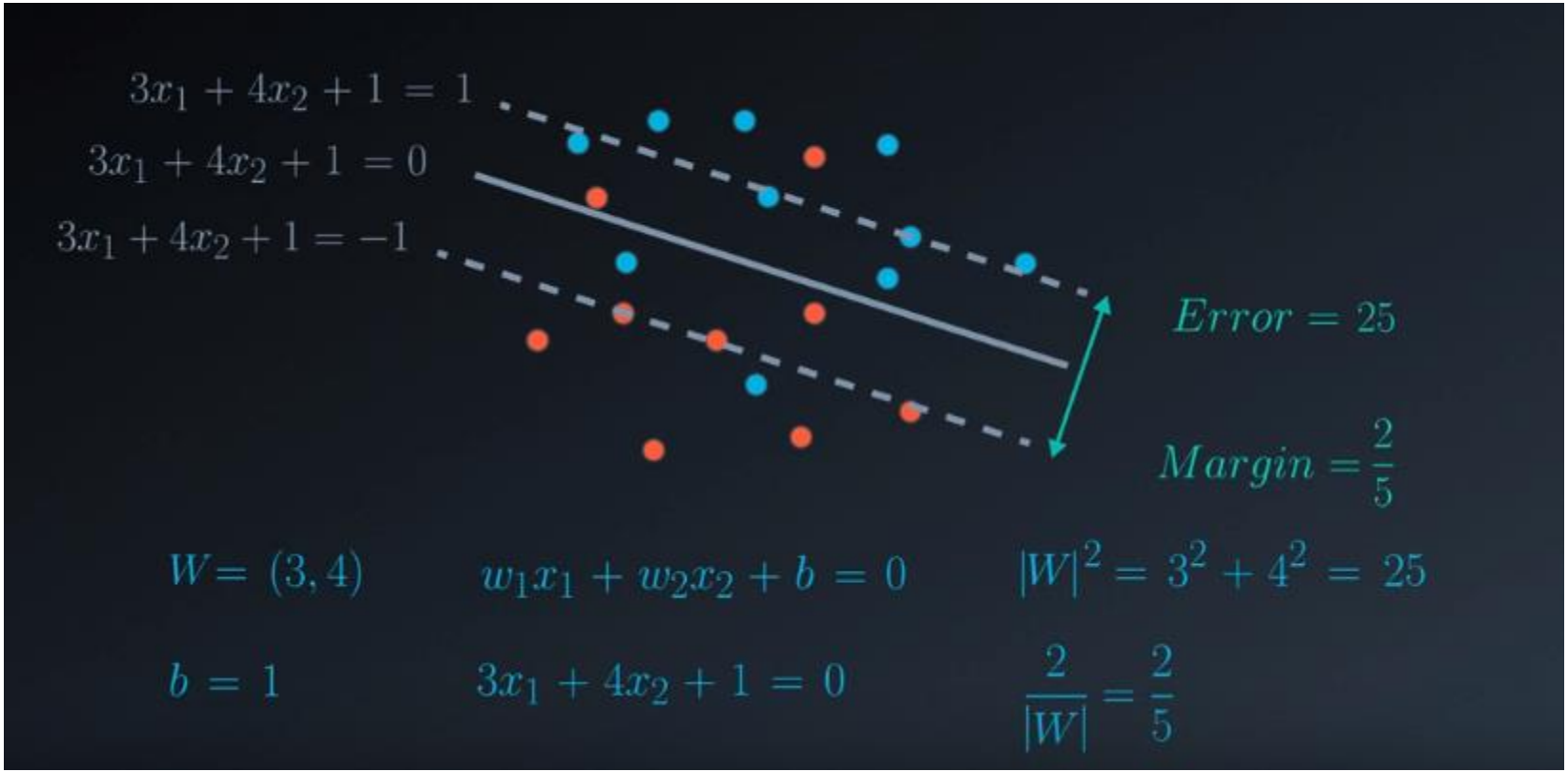
LARGE MARGIN
SMALL ERROR

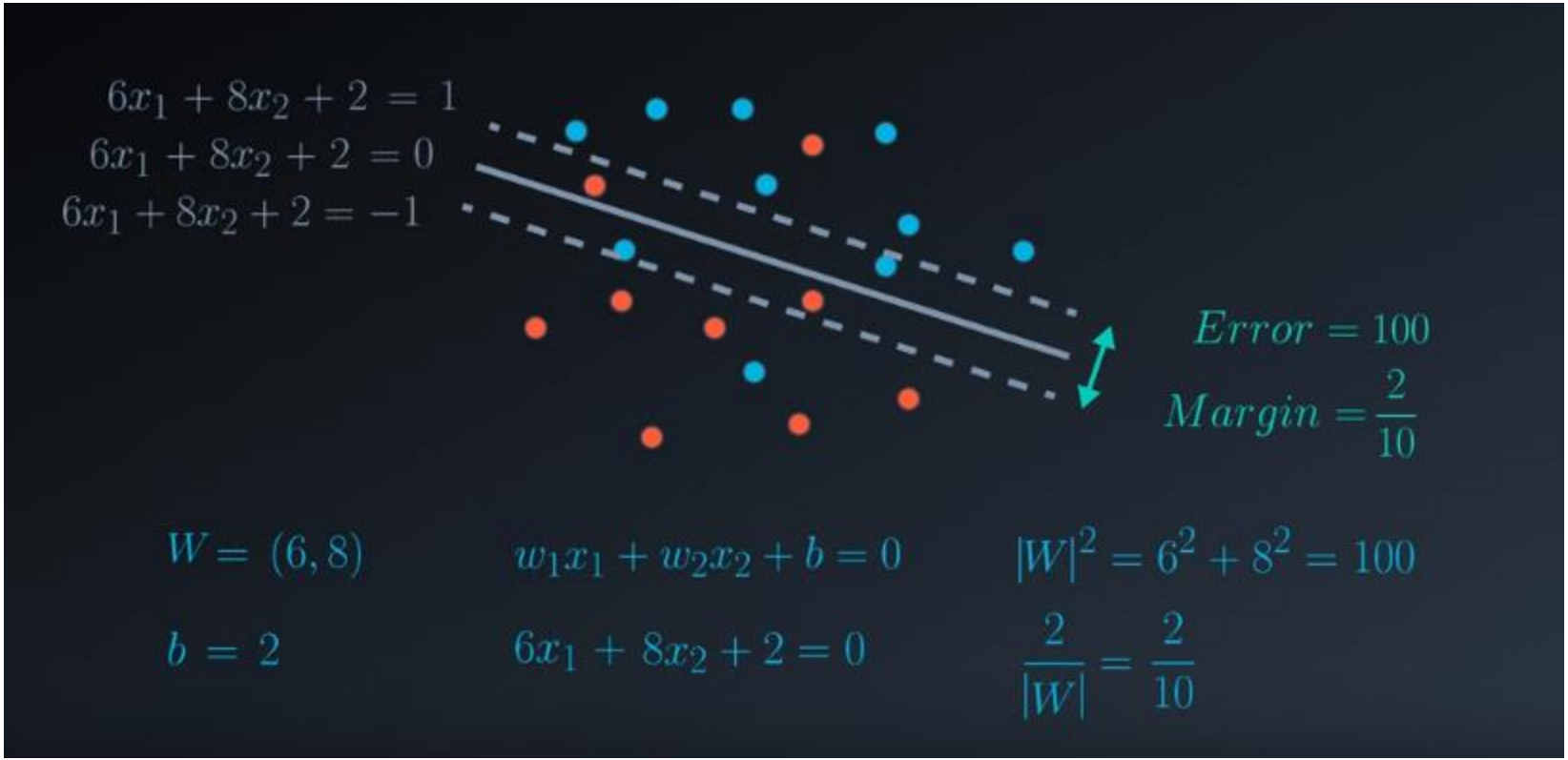
SMALL MARGIN
LARGE ERROR

$$Error = |W|^2$$

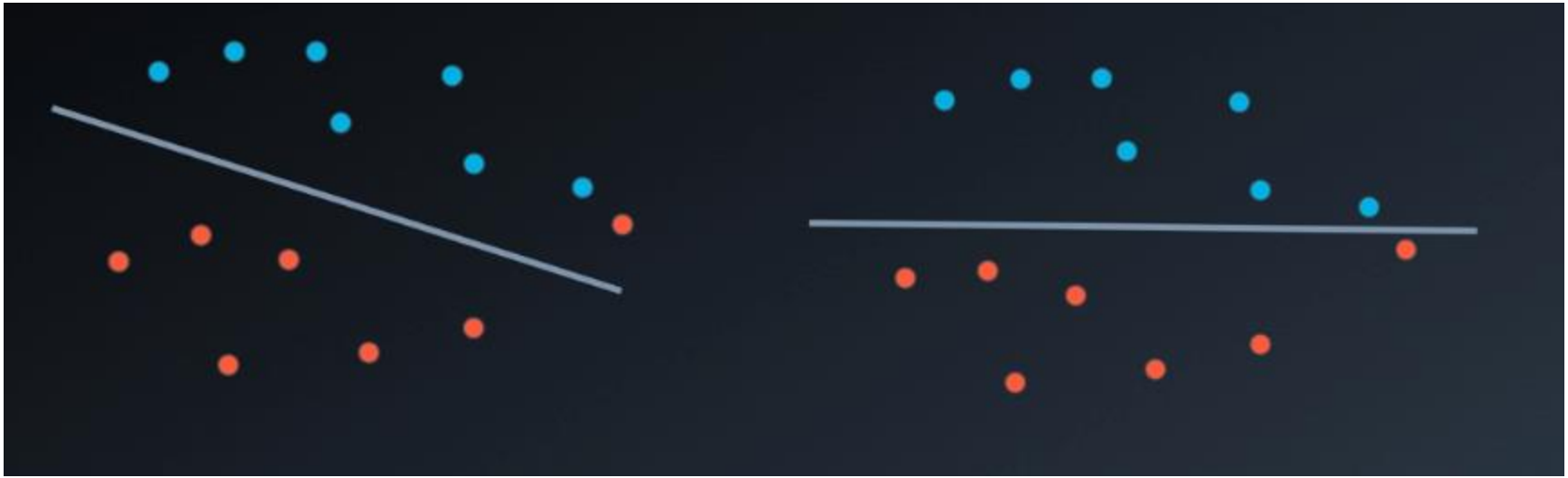
$$Margin = \frac{2}{|W|}$$

LARGE MARGIN ⟶ SMALL ERROR
SMALL MARGIN ⟶ LARGE ERROR


$3x_1 + 4x_2 + 1 = 1$
$3x_1 + 4x_2 + 1 = 0$
$3x_1 + 4x_2 + 1 = -1$
$Error = 25$
$Margin = \frac{2}{5}$
$W = (3, 4)$
$w_1x_1 + w_2x_2 + b = 0$
$|W|^2 = 3^2 + 4^2 = 25$
$b = 1$
$3x_1 + 4x_2 + 1 = 0$
$\frac{2}{|W|} = \frac{2}{5}$

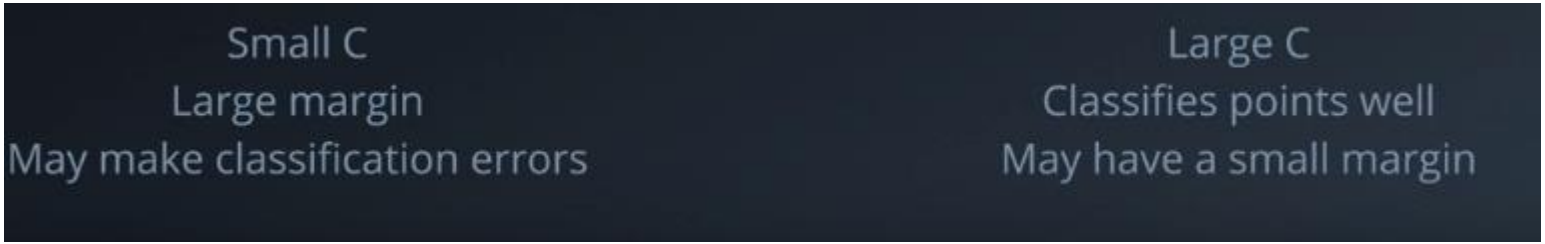

$6x_1 + 8x_2 + 2 = 1$
$6x_1 + 8x_2 + 2 = 0$
$6x_1 + 8x_2 + 2 = -1$
$Error = 100$
$Margin = \frac{2}{10}$
$W = (6, 8)$
$b = 2$
$w_1x_1 + w_2x_2 + b = 0$
$6x_1 + 8x_2 + 2 = 0$
$|W|^2 = 6^2 + 8^2 = 100$
$\frac{2}{|W|} = \frac{2}{10}$

The C Parameter



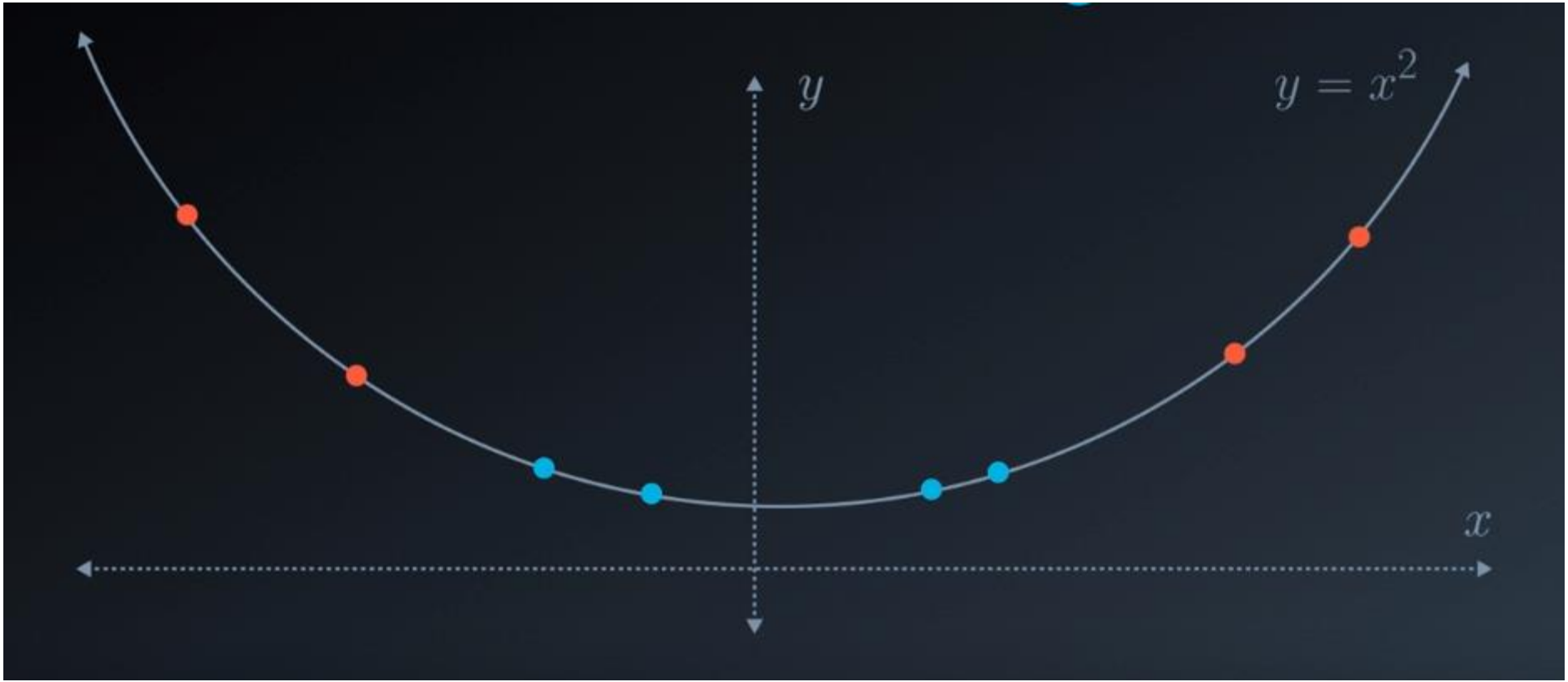
Error = C Classification Error + Margin Error

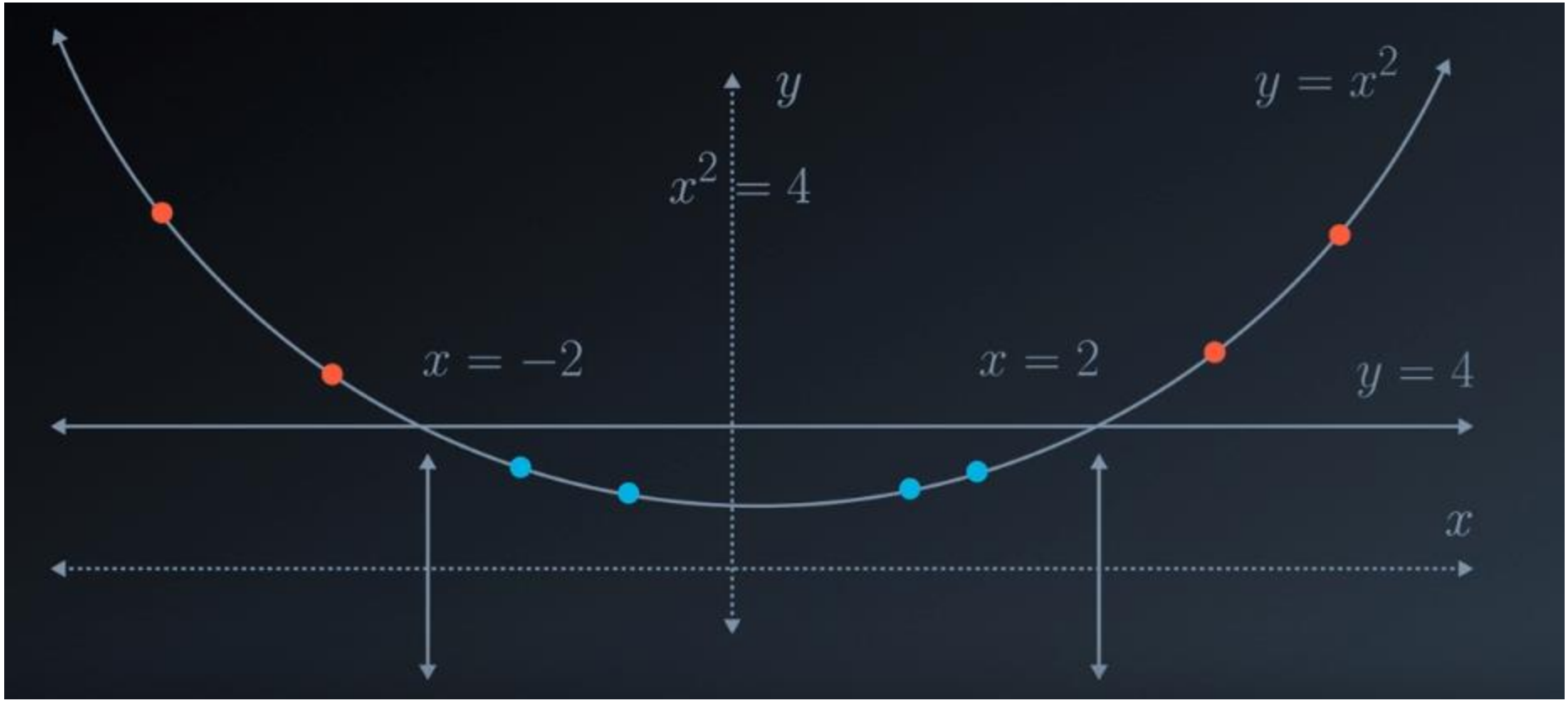
Cheat Code


Small C
Large margin
May make classification errors
Large C
Classifies points well
May have a small margin

Kernel Trick

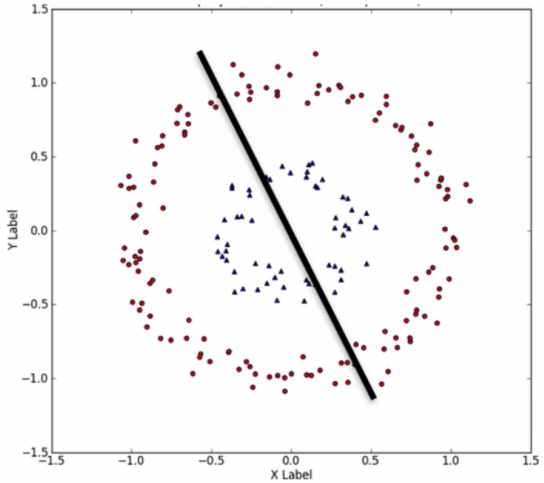
Kernel Trick


y
$y = x^2$
x

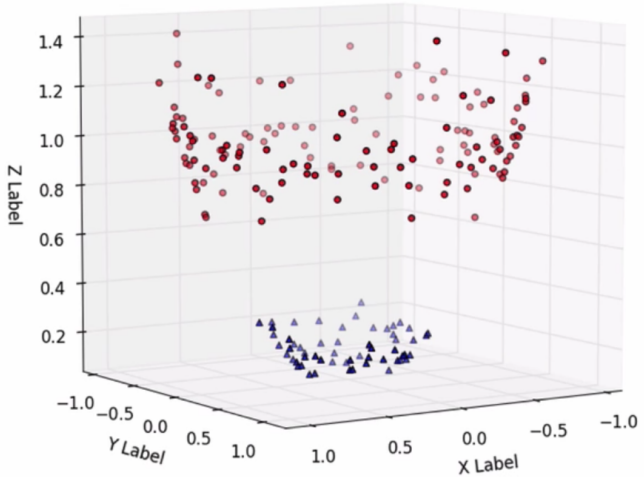

y
$y = x^2$
$x^2 = 4$
$x = -2$
$x = 2$
$y = 4$
x

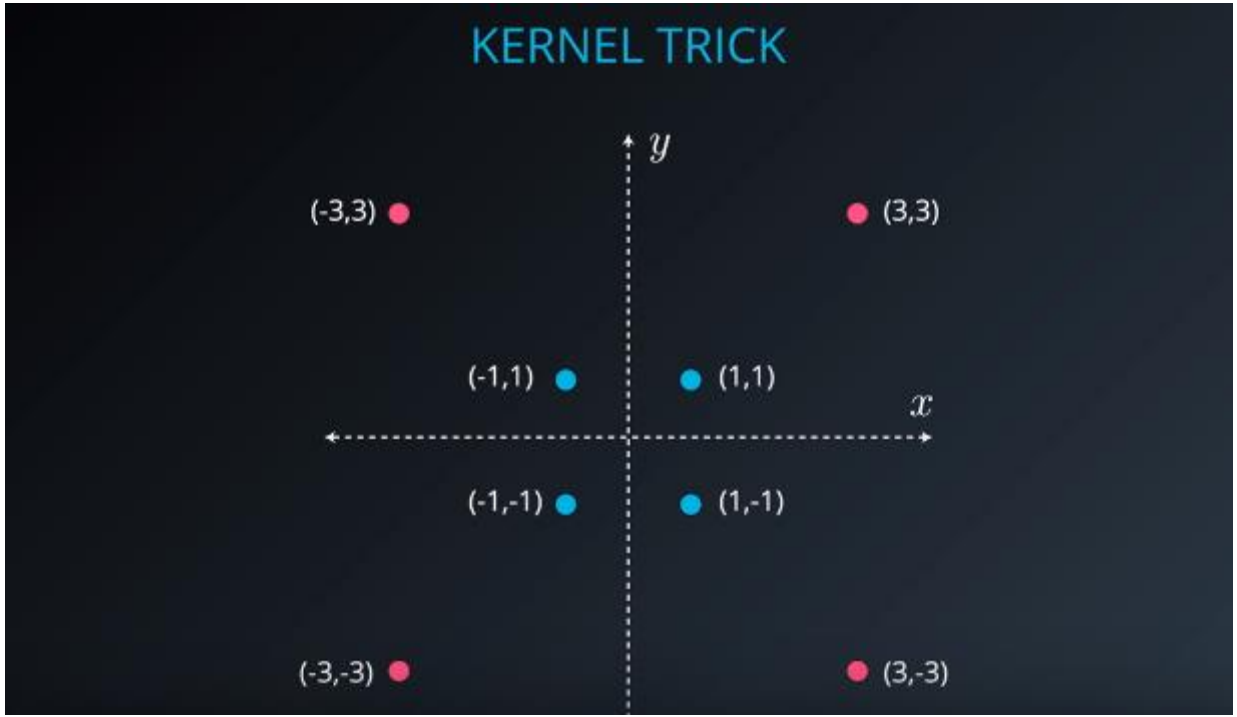

$x = -2$
$x = 2$

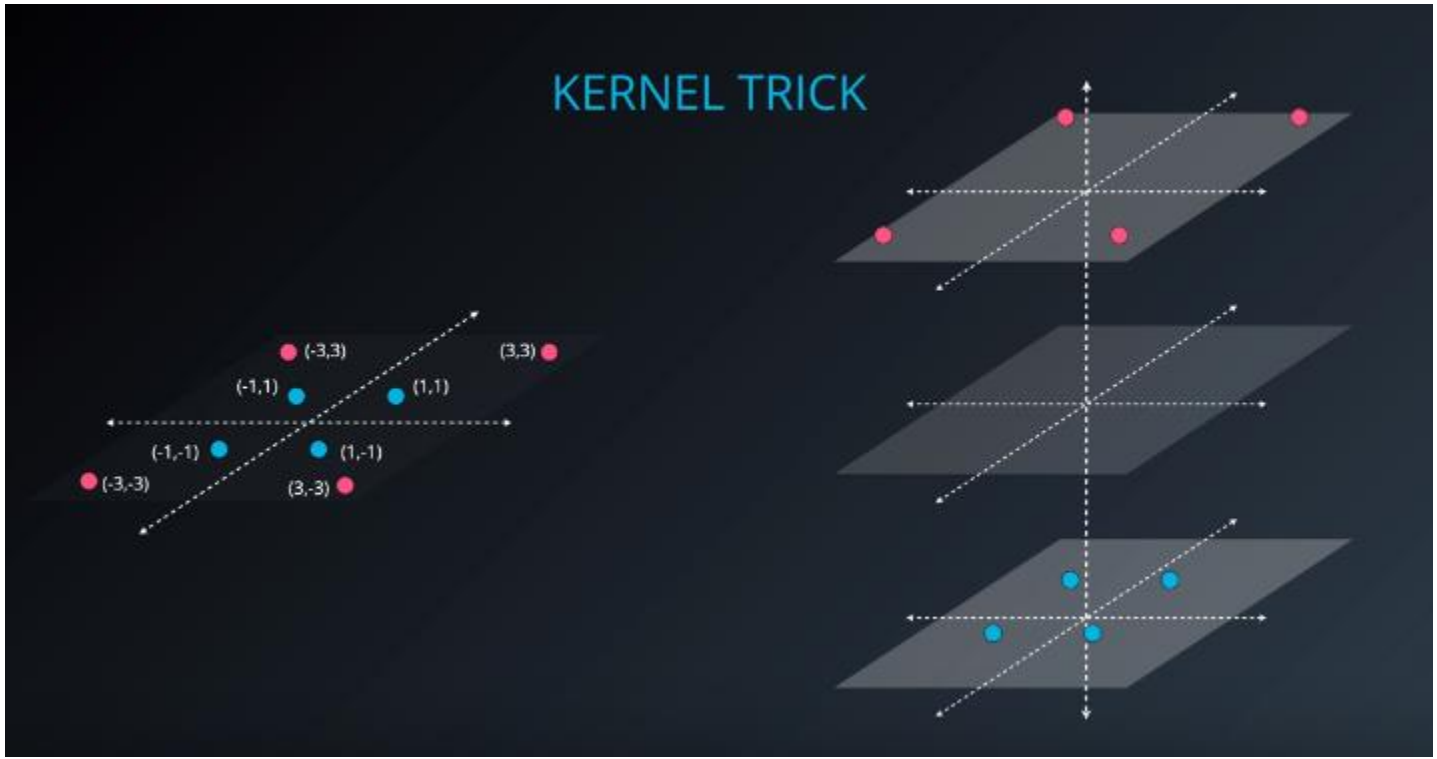
Kernel Trick: Visual

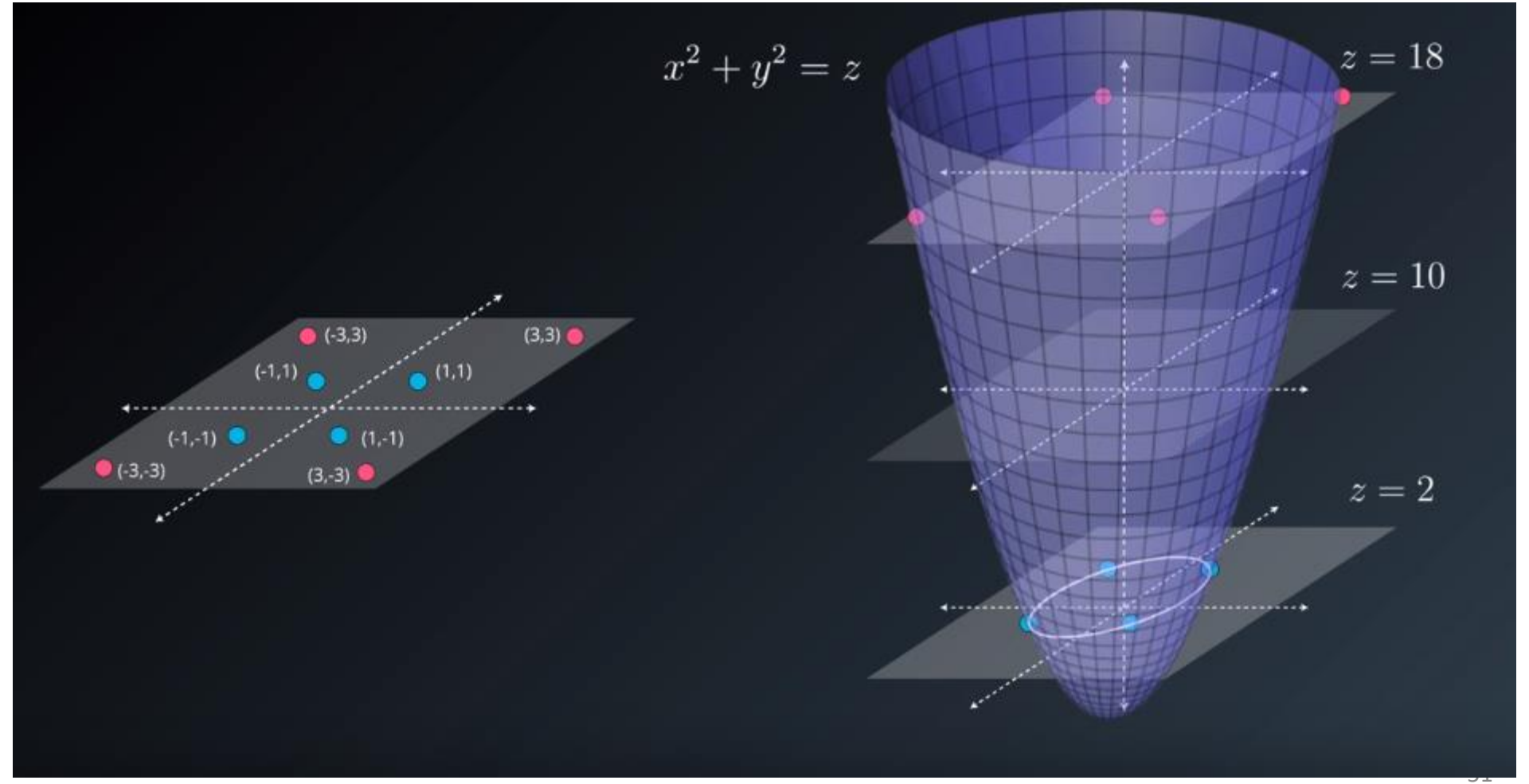


Kernel




KERNEL TRICK
y
x
(-3,3)
(3,3)
(-1,1)
(1,1)
(-1,-1)
(1,-1)
(-3,-3)
(3,-3)


KERNEL TRICK
(-3,3)
(3,3)
(-1,1)
(1,1)
(-1,-1)
(1,-1)
(-3,-3)
(3,-3)


$x^2 + y^2 = z$
$z = 18$
$z = 10$
$z = 2$
(-3,3)
(3,3)
(-1,1)
(1,1)
(-1,-1)
(1,-1)
(-3,-3)
(3,-3)

RBF KERNEL

RBF KERNEL

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

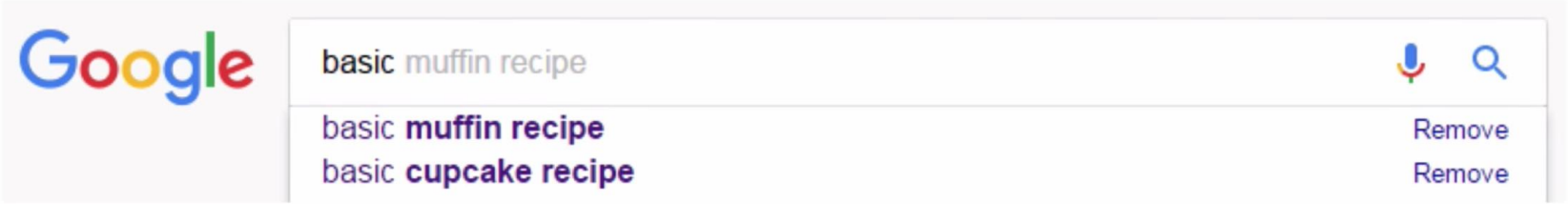
Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

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

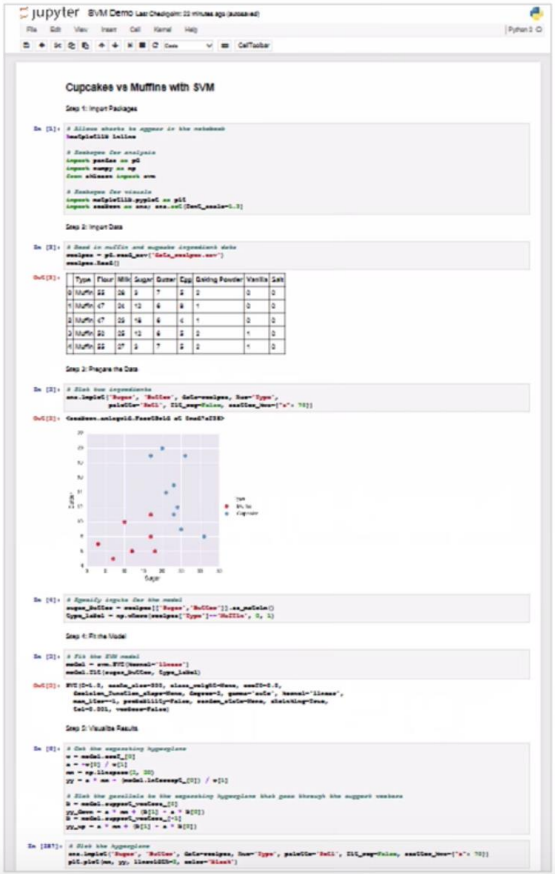
The Challenge

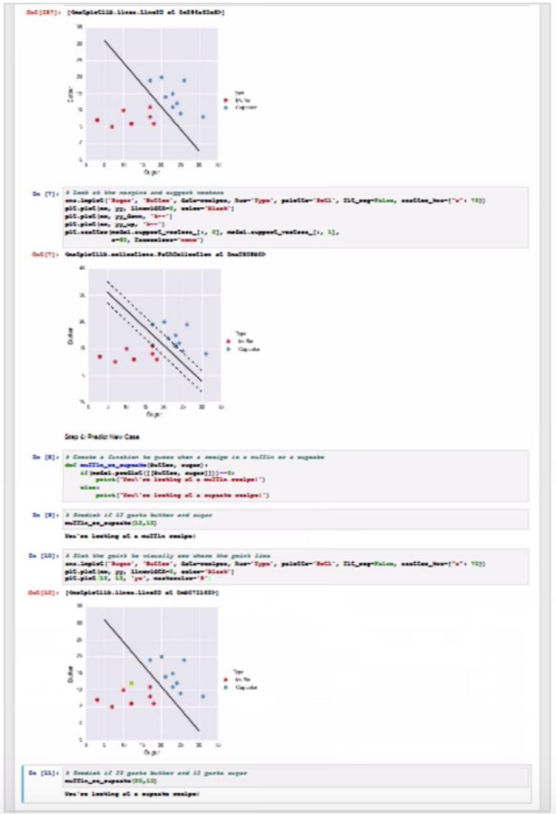
Classify recipes as cupcakes or muffins. When given a new recipe, determine if it's a cupcake or a muffin.

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

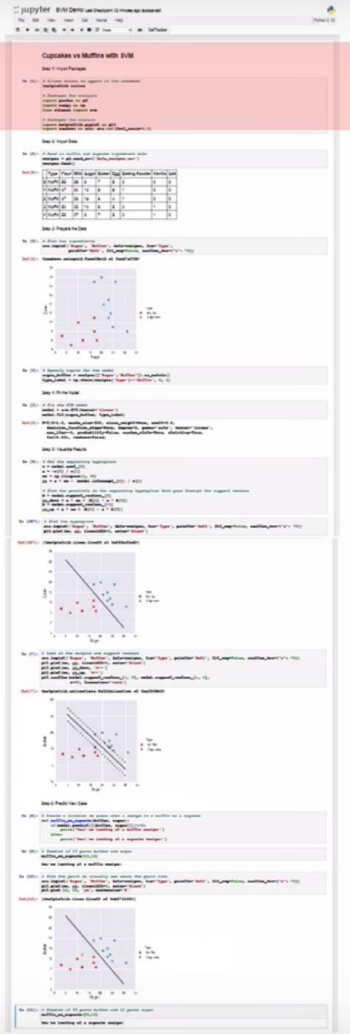
f. Predict New Case

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)
```



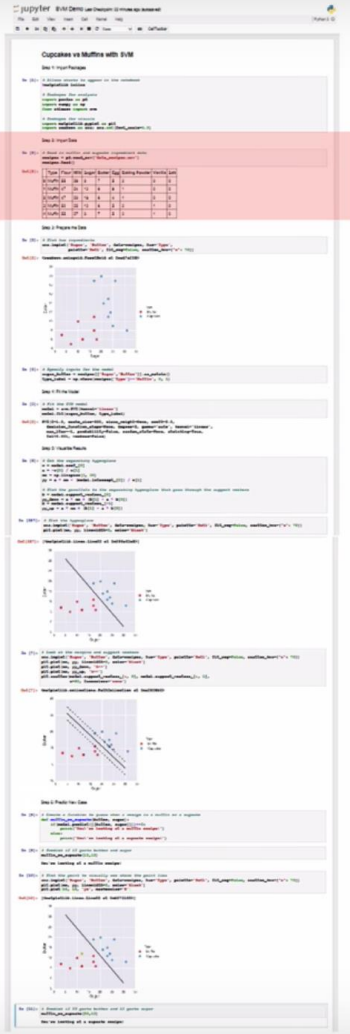
b. Import Data

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

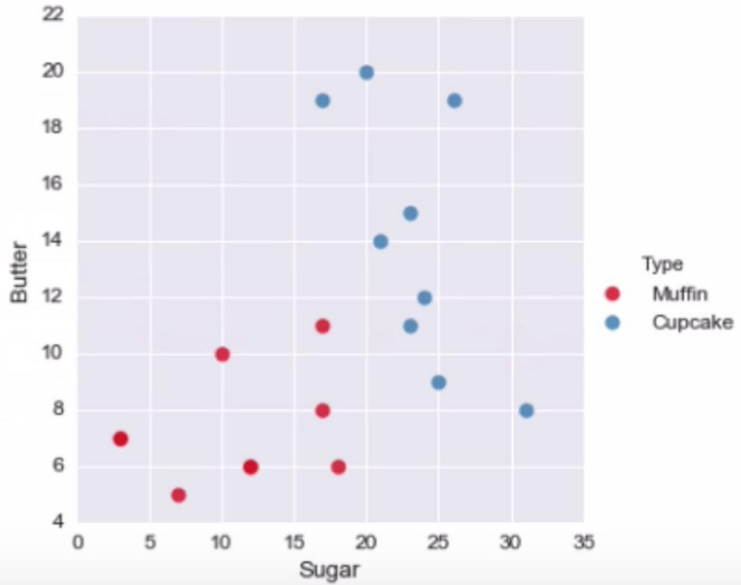


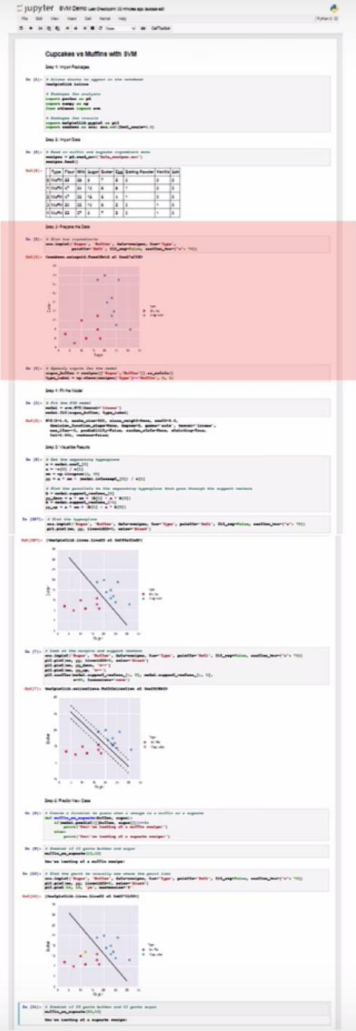
c. Prepare the Data

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>





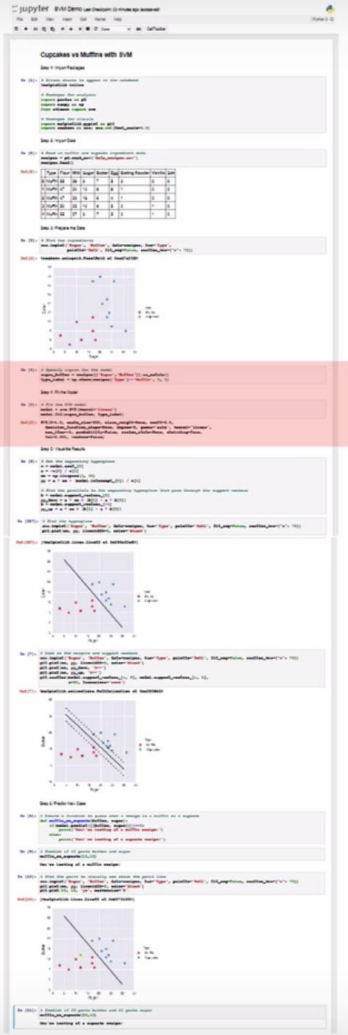
d. Fit the Model

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)
```



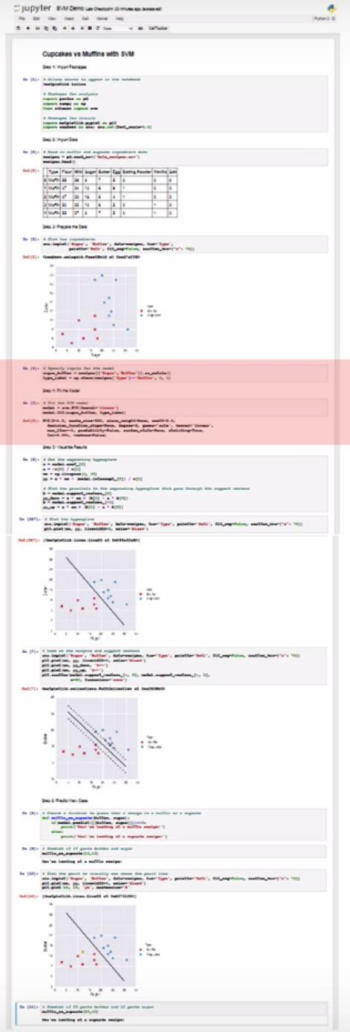
d. Fit the Model

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)
```

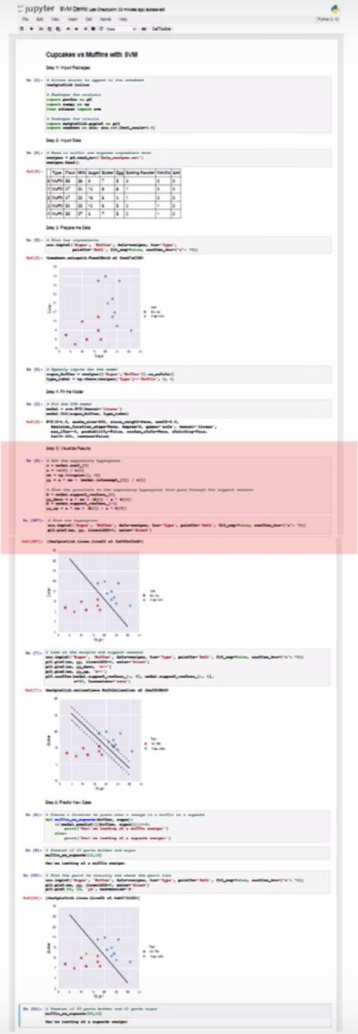
Out[5]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```



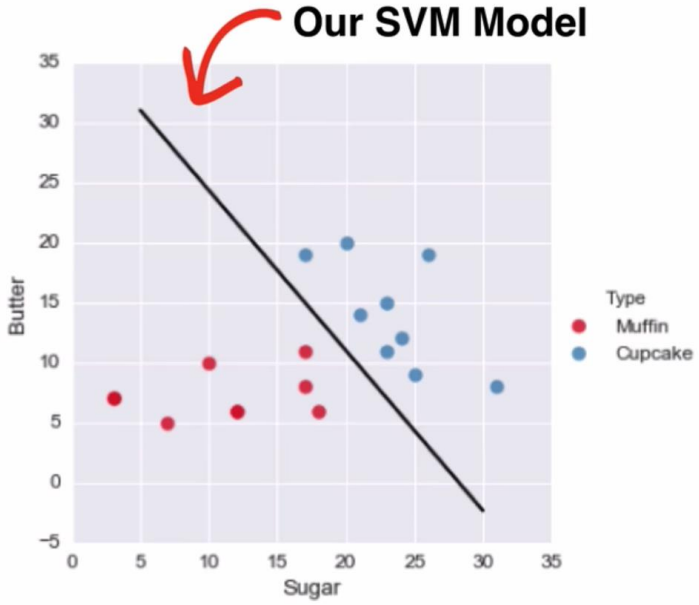
e. Visualize Results

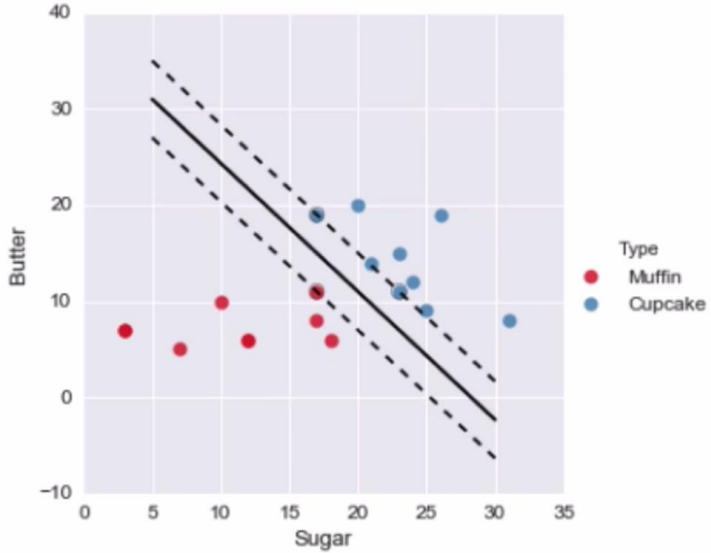
Step 5: Visualize Results

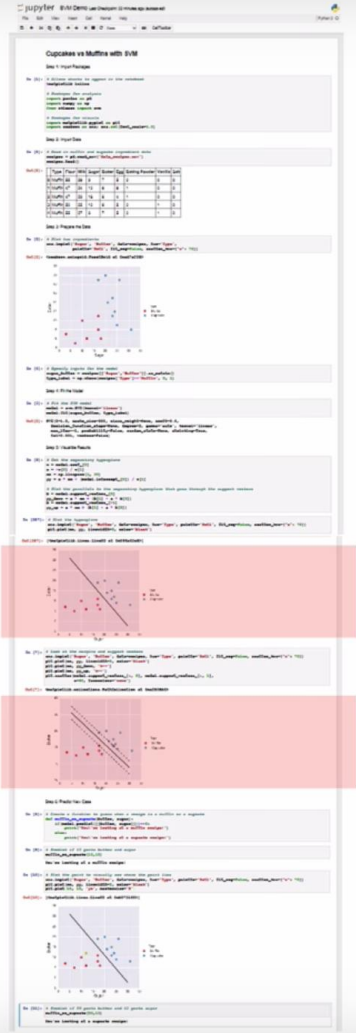
In [6]:
```
# Get the separating hyperplane
w = model.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(5, 30)
yy = a * xx - (model.intercept_[0]) / w[1]

# 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

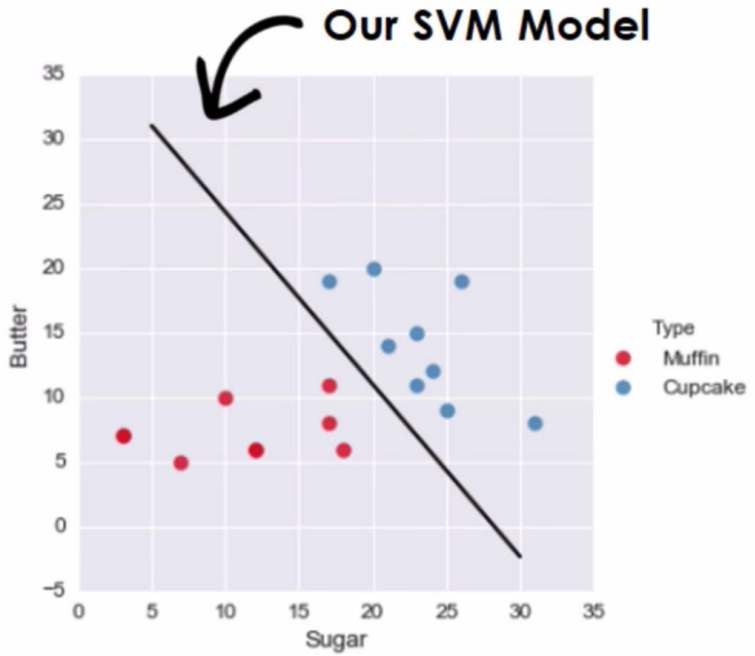
1. Find the data ✓
2. Apply a data science model ✓
3. Review the results

3. Review the Results

The Challenge

Classify recipes as cupcakes or muffins. ✓

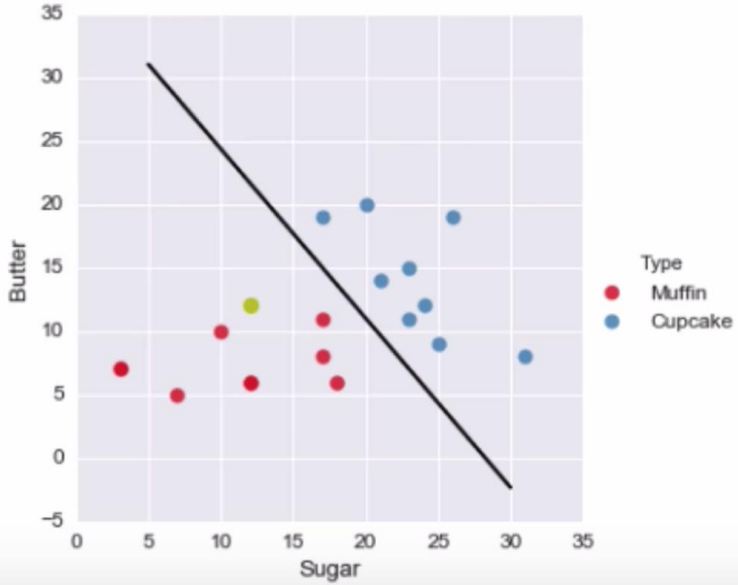
When given a new recipe, determine if it's a cupcake or a muffin.

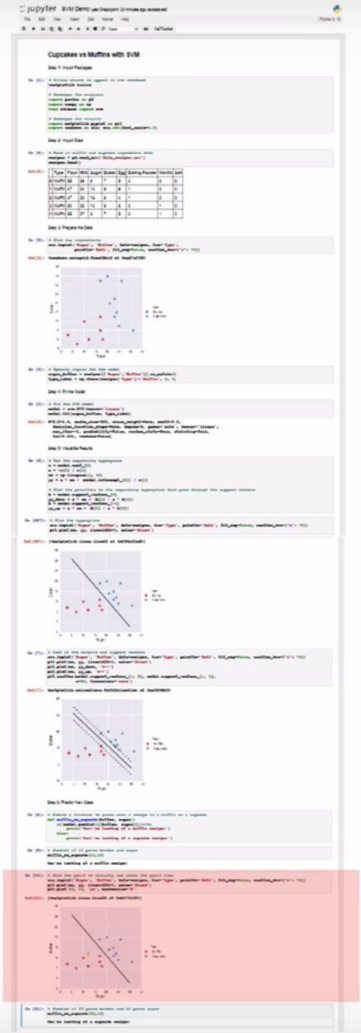


f. Predict New Case

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>]





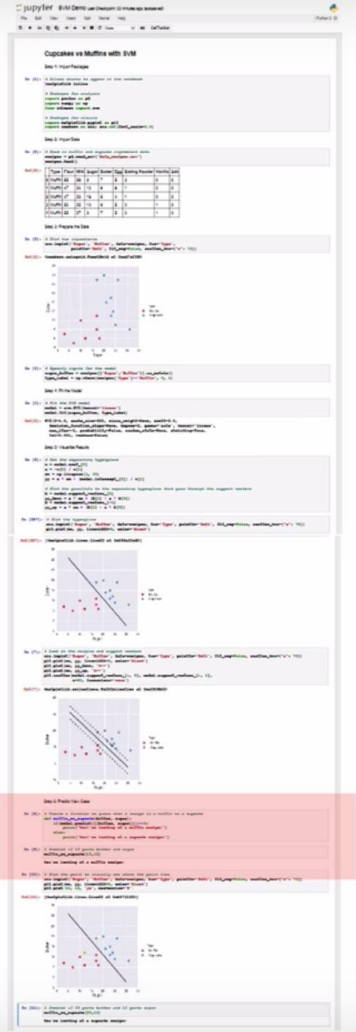
f. Predict New Case

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!
```



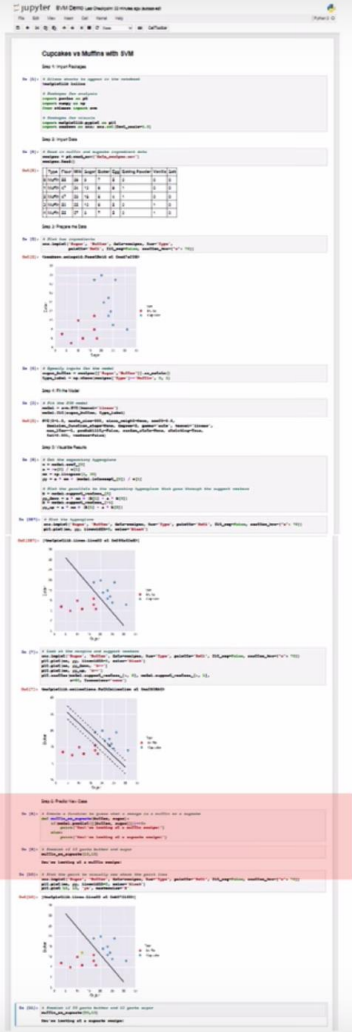
f. Predict New Case

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!

THANK YOU