ML Workshop

08 Dec 2018

By:

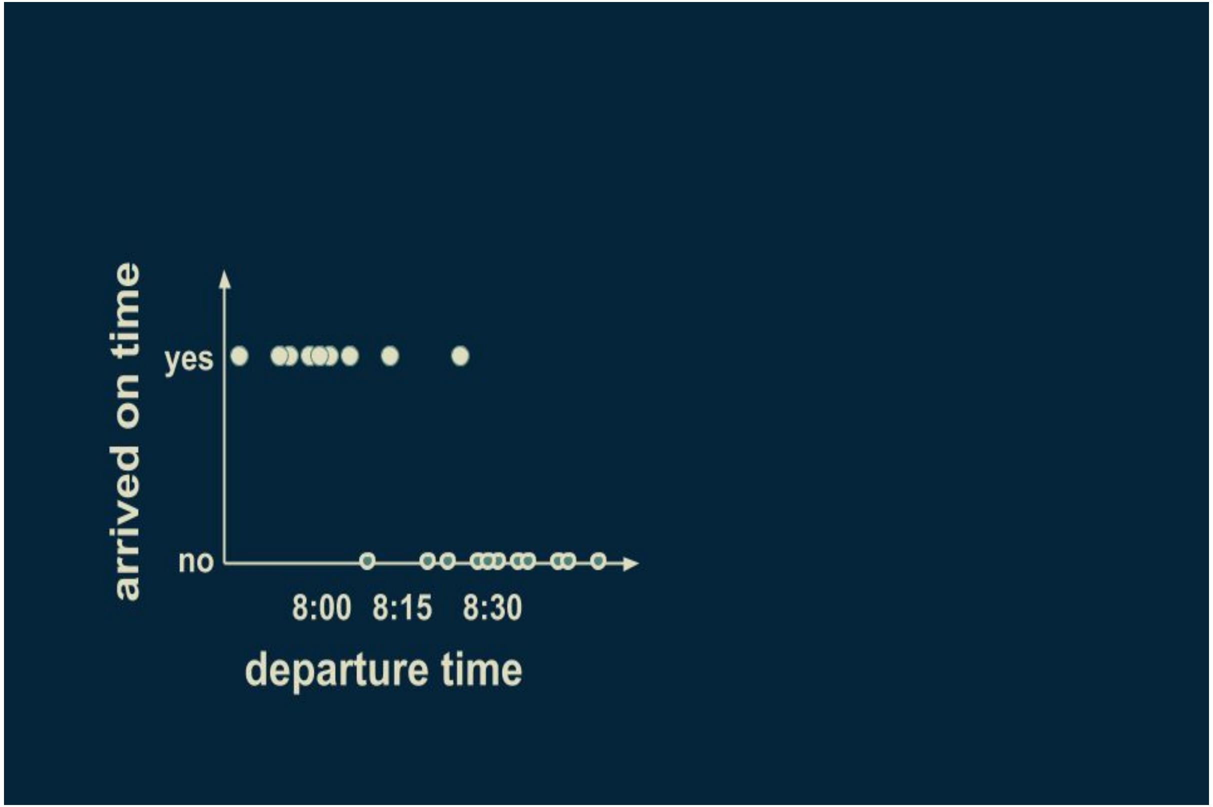
- Raj Dhyani (rddhyani@gmail.com)
    - iMarkServ, IIT Bombay
- Prashant K. Sharma (prashaantsharmaa@gmail.com)
    - IIT Bombay

Contents

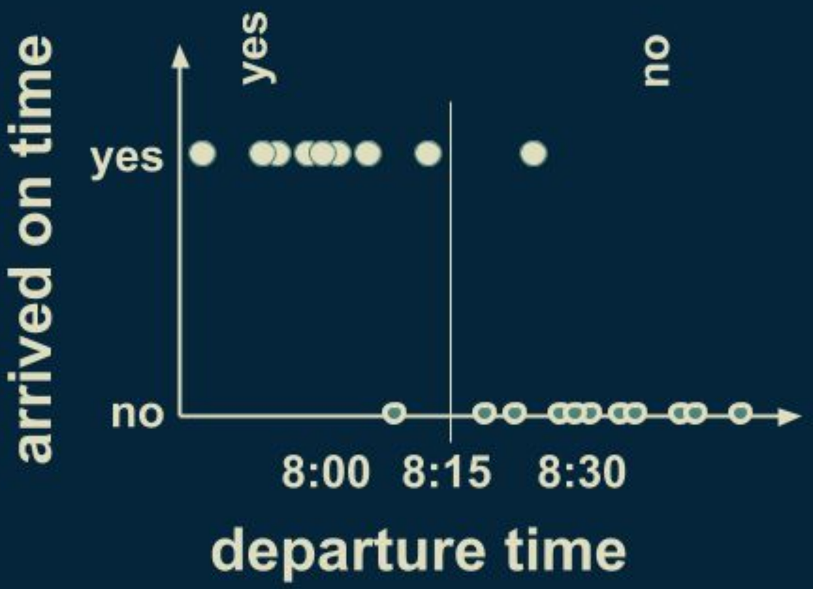
1. Decision Trees
    a. Random Forest
    b. XGboost
2. SVC/SVR
3. Naive Bayes
4. Logistic Regression

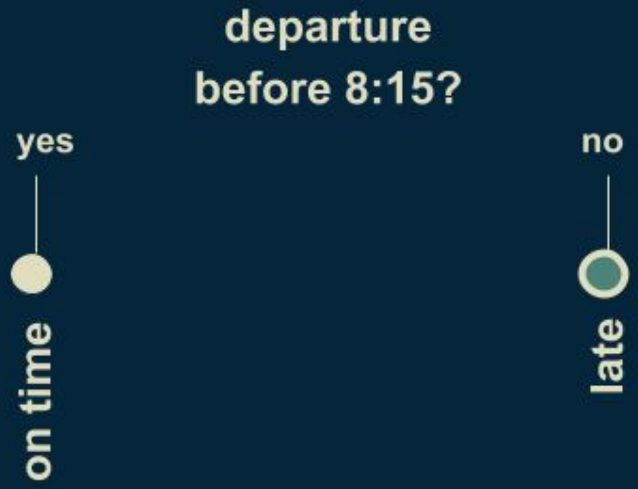
Decision Trees

A simple Decision Tree



Consider a dataset: Recording time of leaving the house, note whether you arrive on work on time.



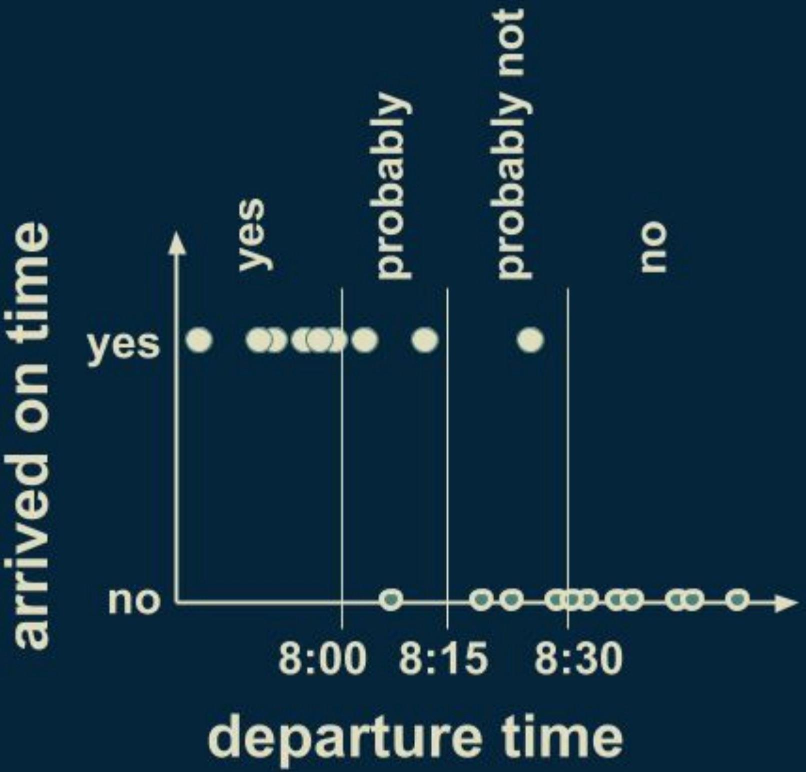

departure before 8:15?

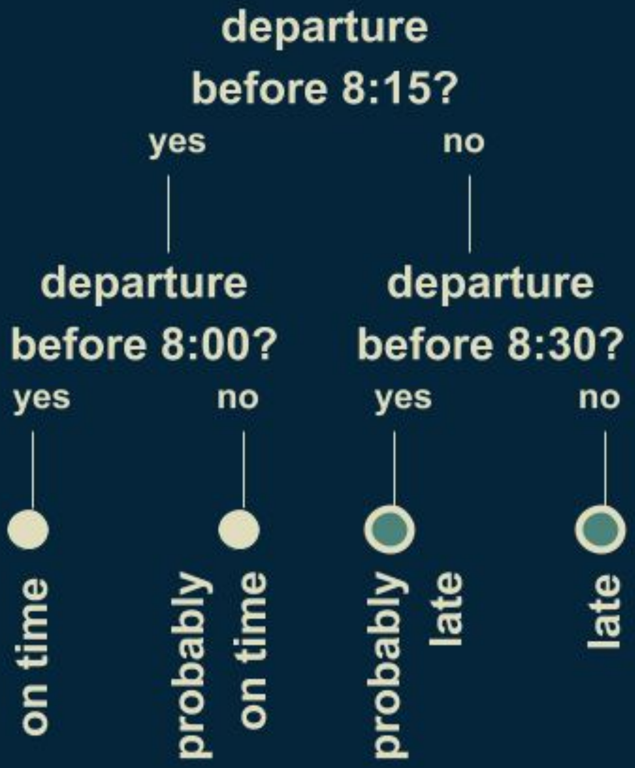
yes

no

on time

late





departure before 8:15?

yes

no

departure before 8:00?

yes

no

departure before 8:30?

yes

no

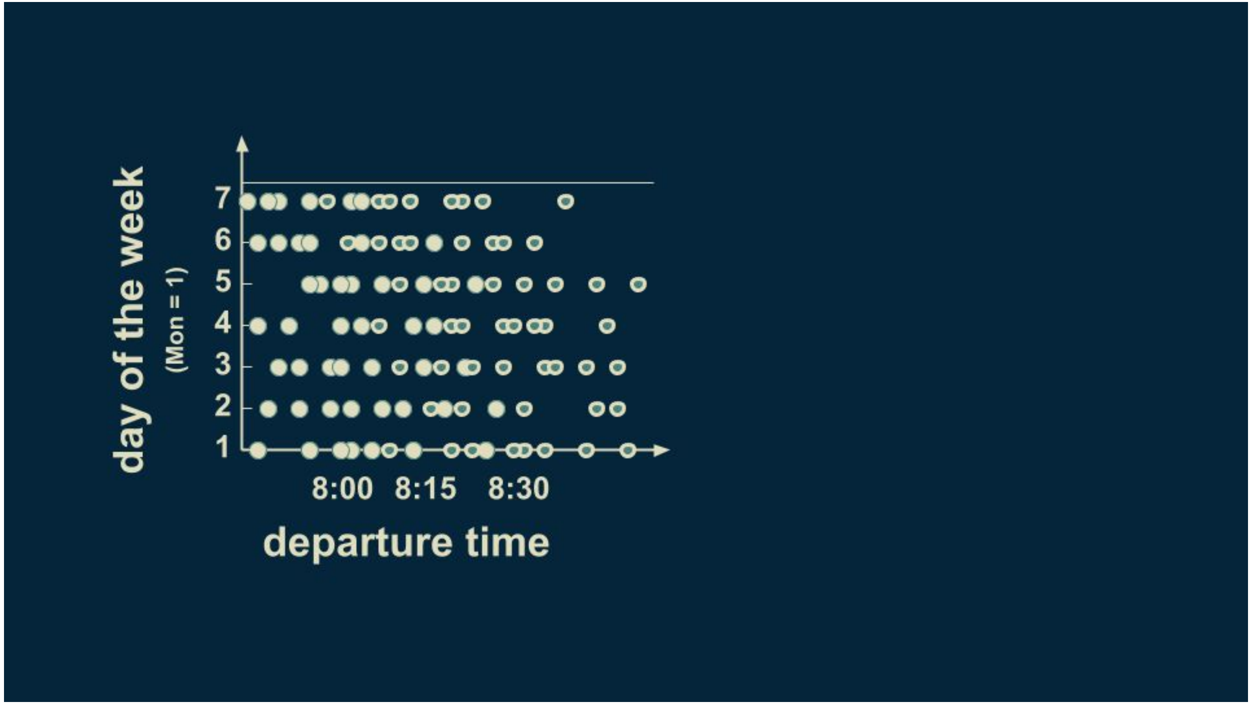
on time

probably on time

probably late

late

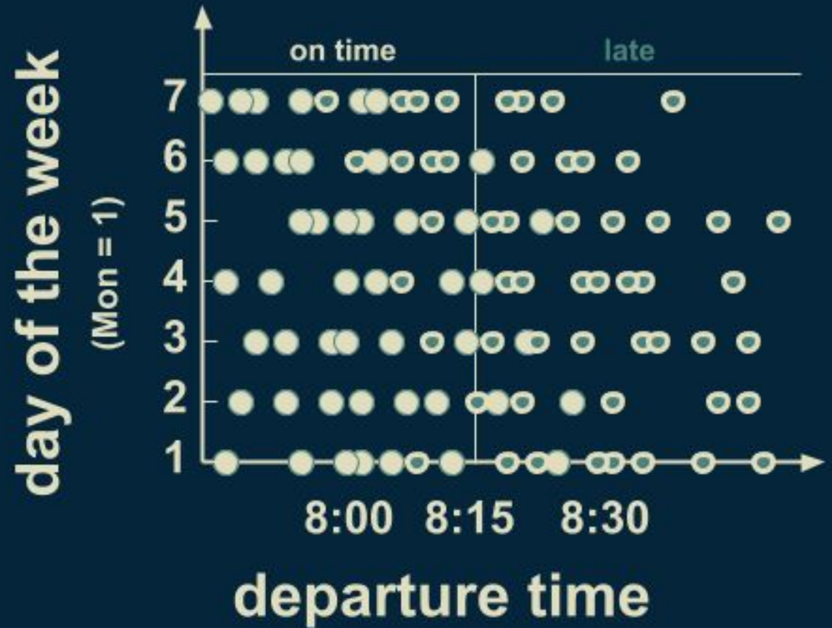
Decision Tree: Consider two variables/features

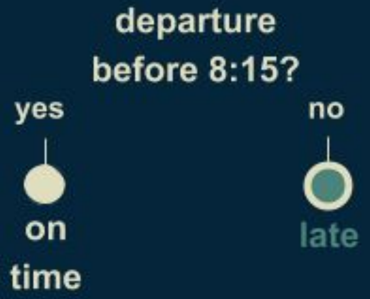


Consider:

1. Departure time
2. Day of week

Monday =1 , Saturday = 7





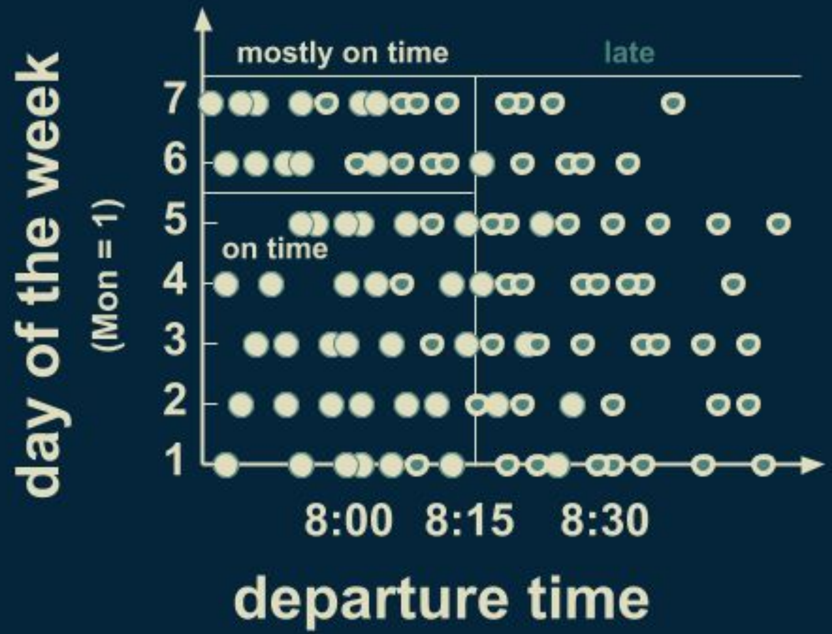
departure before 8:15?

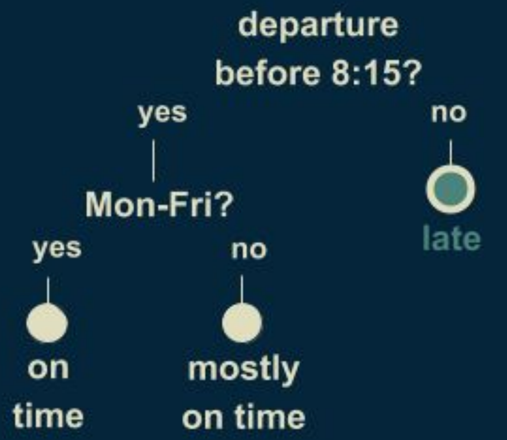
yes

no

on time

late





departure before 8:15?

yes

no

Mon-Fri?

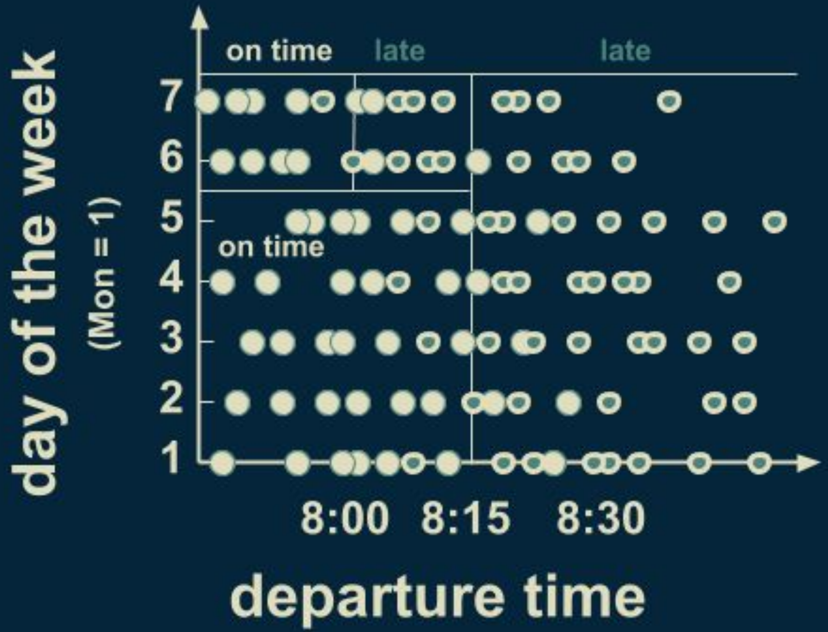
yes

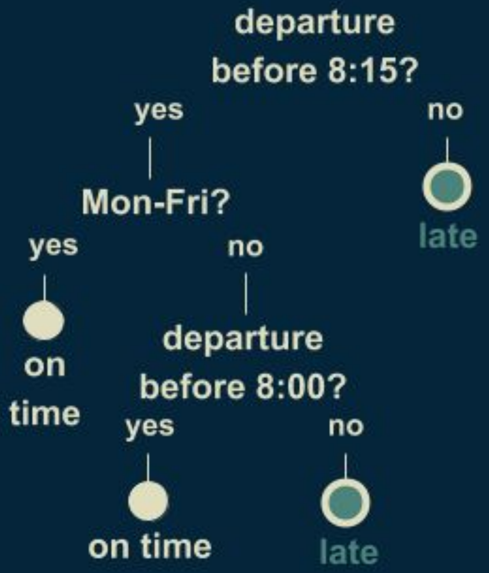
no

late

on time

mostly on time





departure before 8:15?

yes

no

late

Mon-Fri?

yes

no

on time

departure before 8:00?

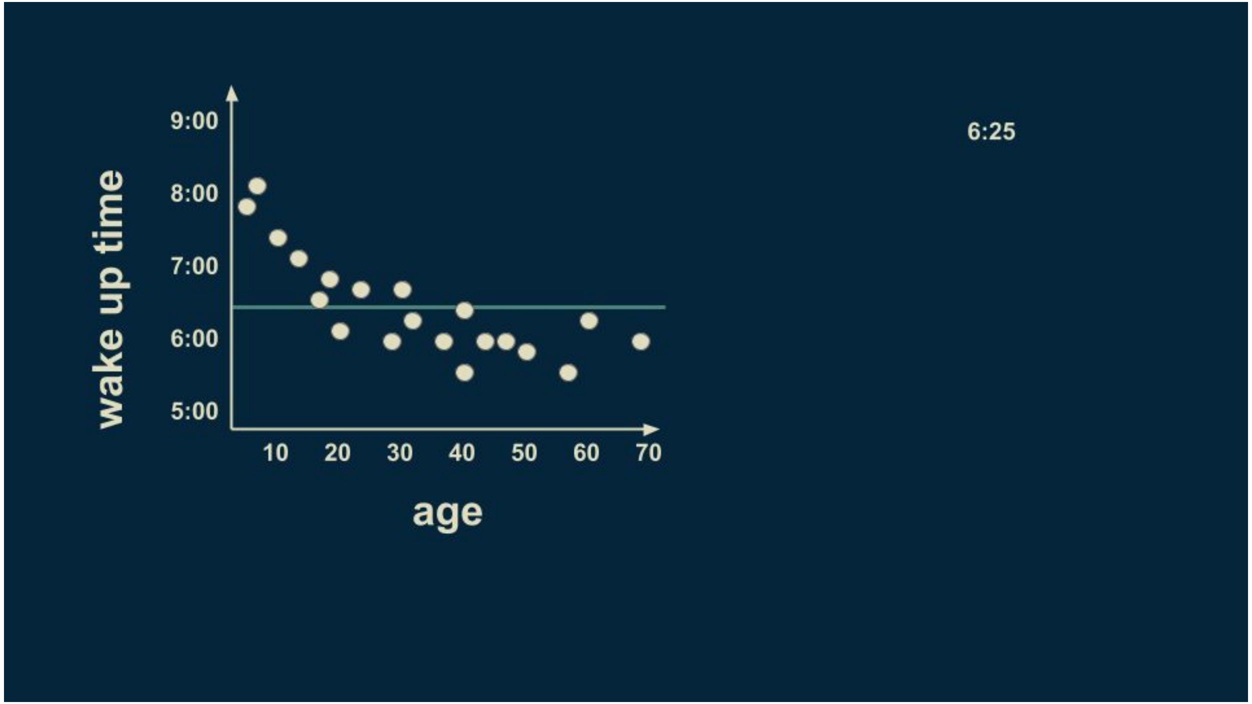
yes

no

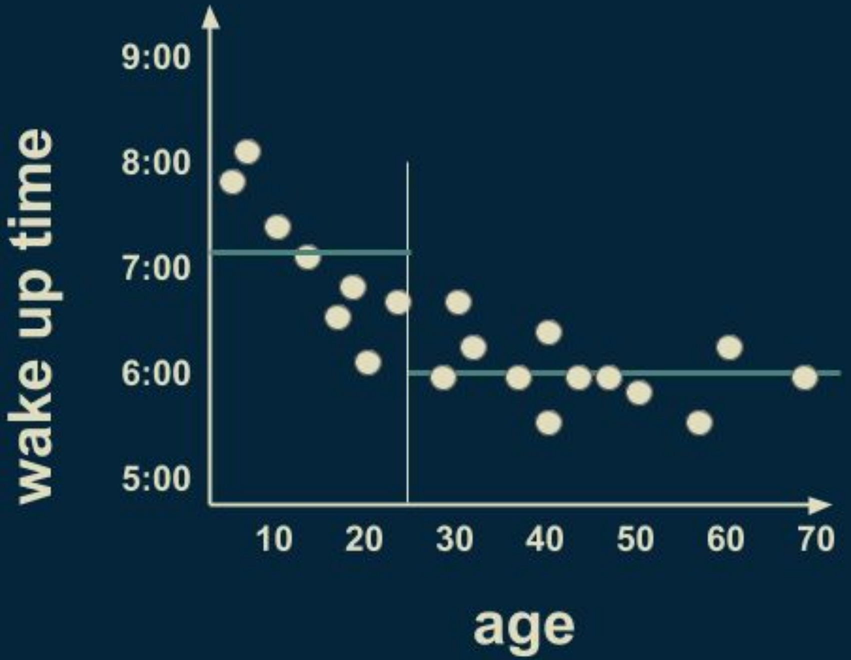
on time

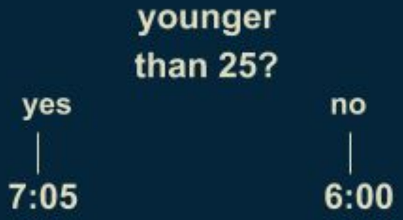
late

Decision Tree: Continuous Variable/Regression Tree



1. The root of regression tree is estimate for the entire dataset.





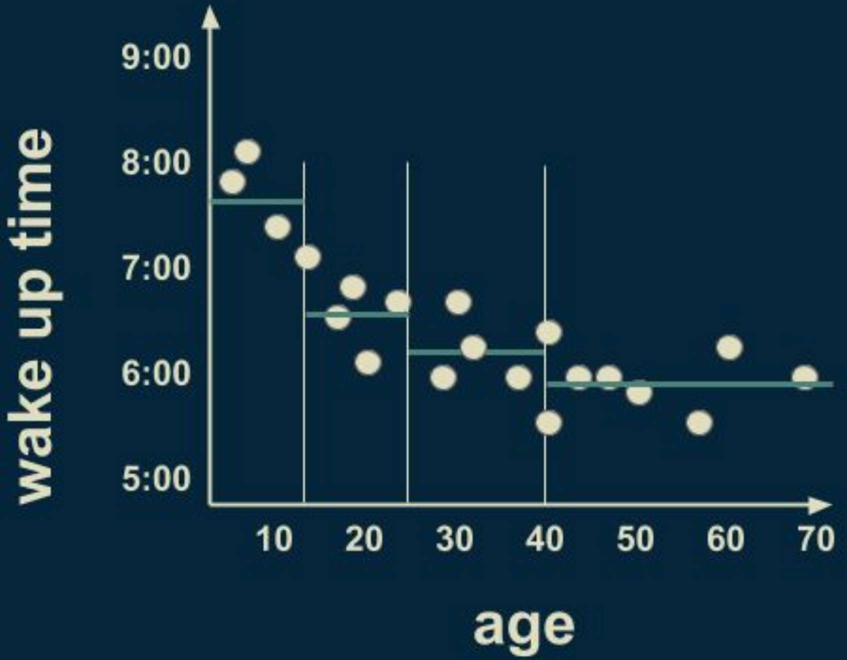
younger than 25?

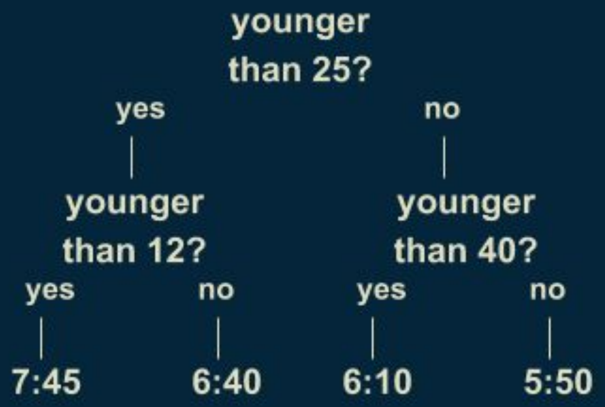
yes

7:05

no

6:00





younger than 25?

yes

younger than 12?

yes

7:45

no

6:40

no

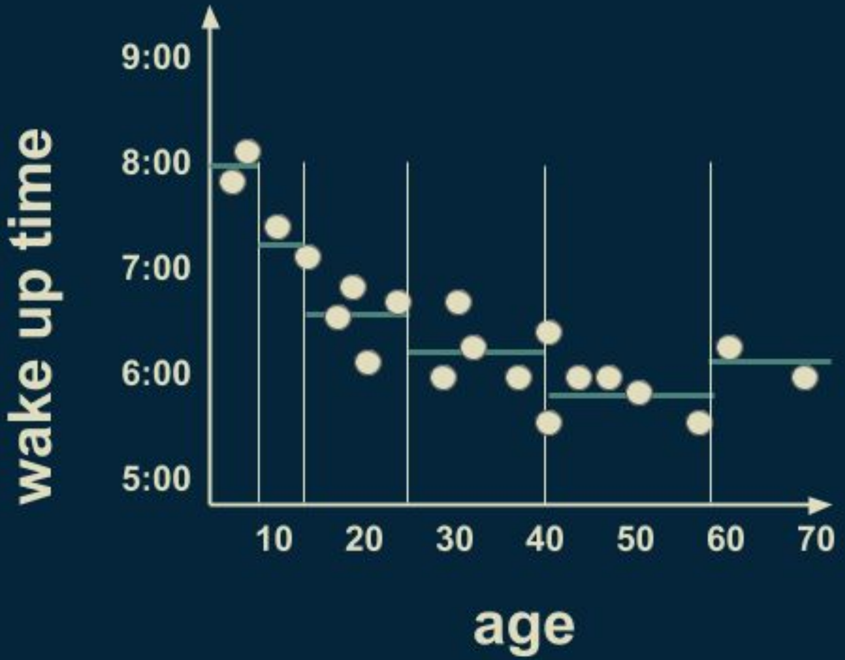
younger than 40?

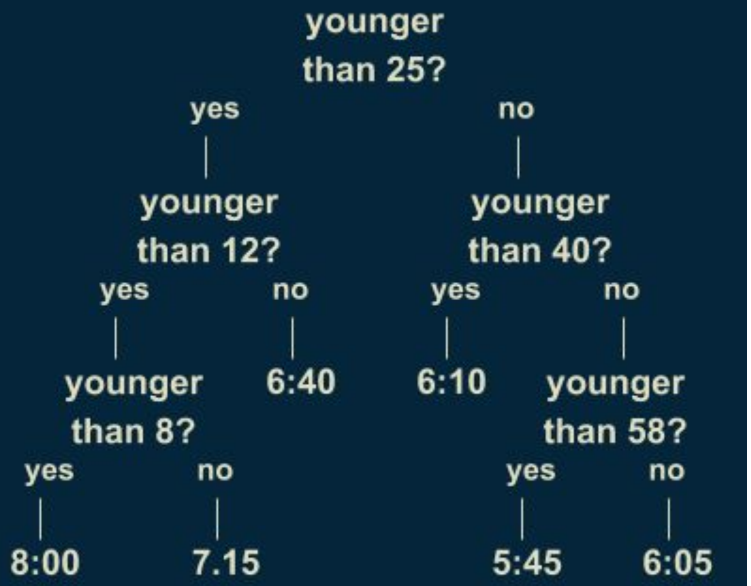
yes

6:10

no

5:50





younger than 25?

yes

younger than 12?

yes

younger than 8?

yes

8:00

no

7.15

no

6:40

no

younger than 40?

yes

6:10

no

younger than 58?

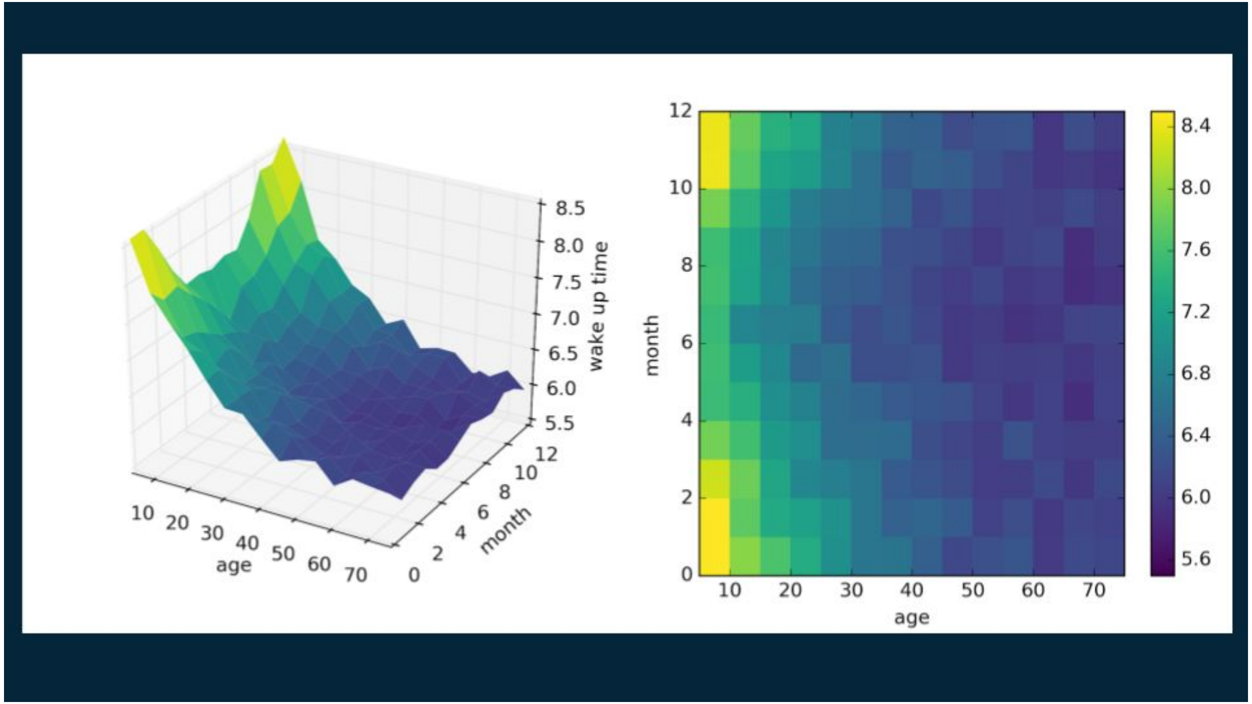
yes

5:45

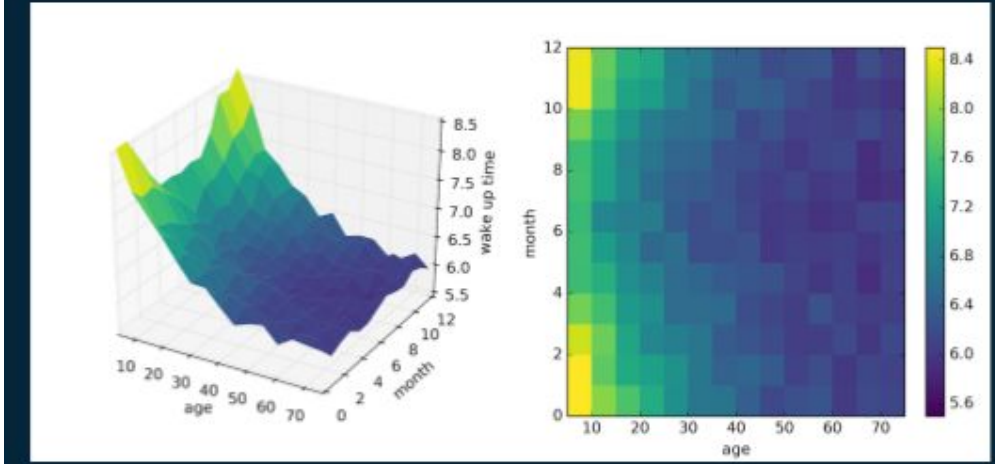
no

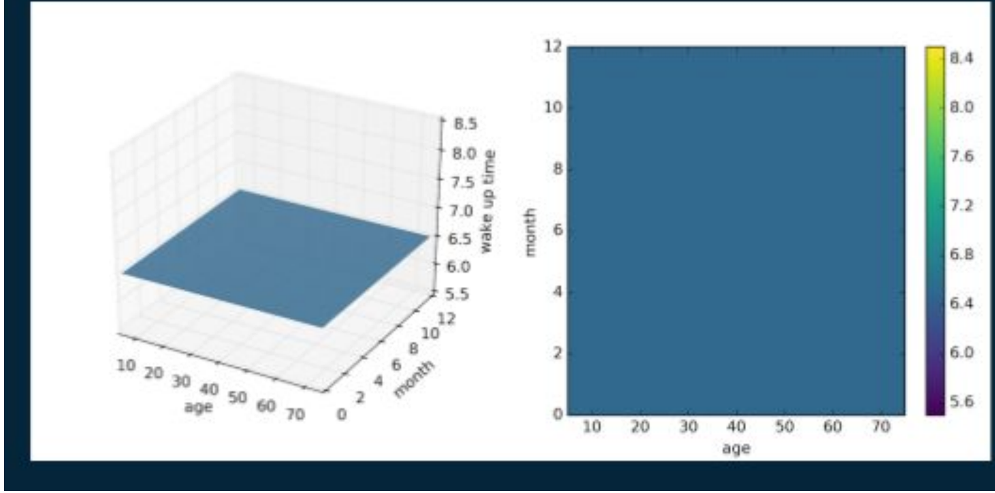
6:05

Decision Tree: 2 Variable Regression Tree

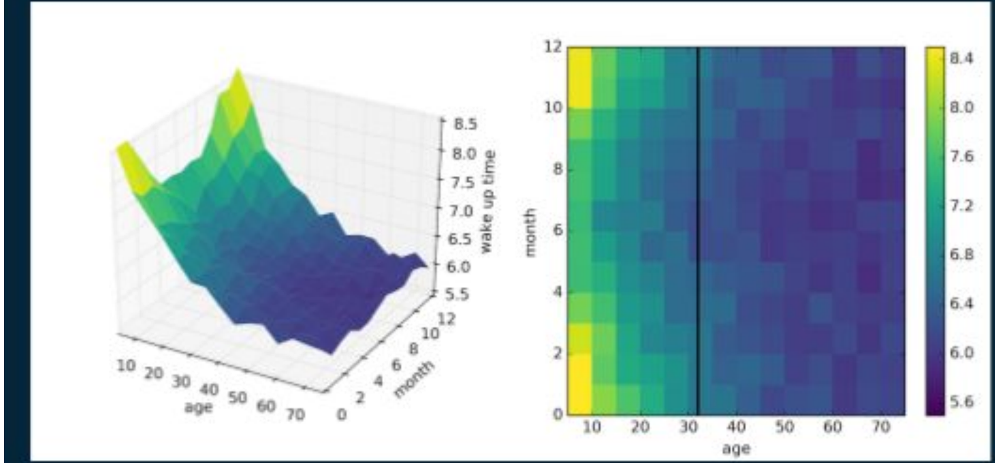


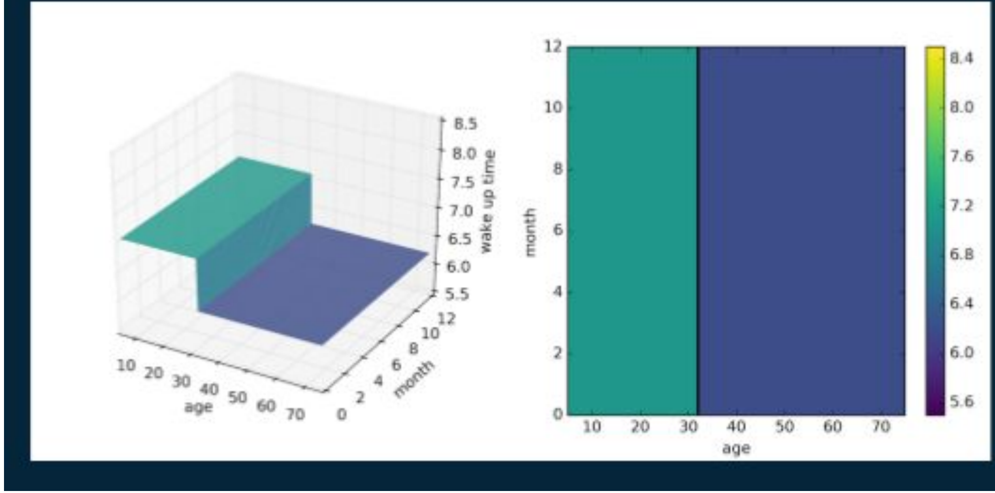
If we consider not only someone's age, but the month of the year as well, then we can find even richer patterns

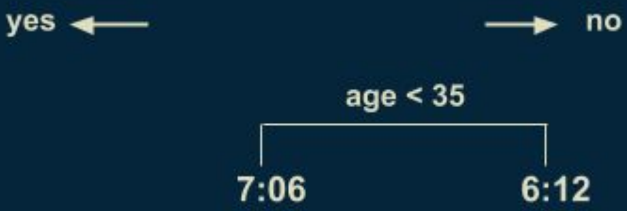

wake up time
8.5
8.0
7.5
7.0
6.5
6.0
5.5
10
20
30
40
50
60
70
age
0
2
4
6
8
10
12
month
12
10
8
6
4
2
0
month
10
20
30
40
50
60
70
age
8.4
8.0
7.6
7.2
6.8
6.4
6.0
5.6


wake up time
8.5
8.0
7.5
7.0
6.5
6.0
5.5
10
20
30
40
50
60
70
age
0
2
4
6
8
10
12
month
12
10
8
6
4
2
0
month
10
20
30
40
50
60
70
age
8.4
8.0
7.6
7.2
6.8
6.4
6.0
5.6

6:30







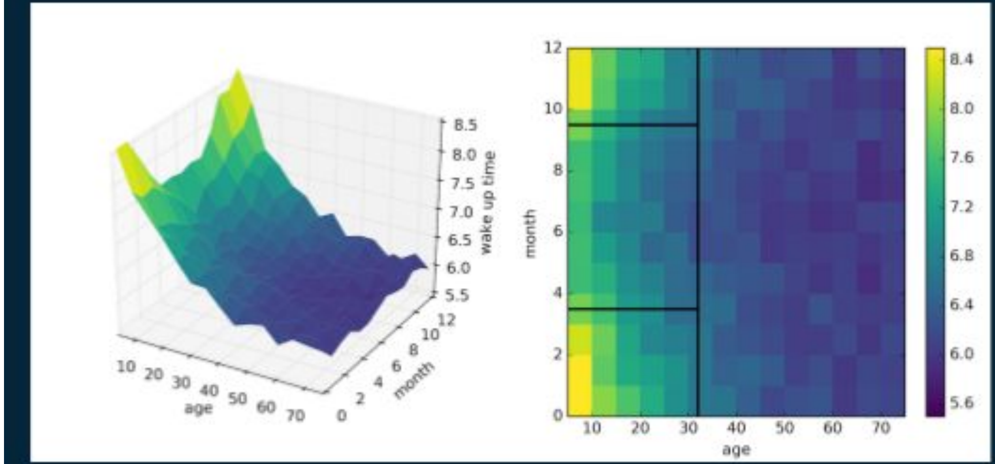
yes ←

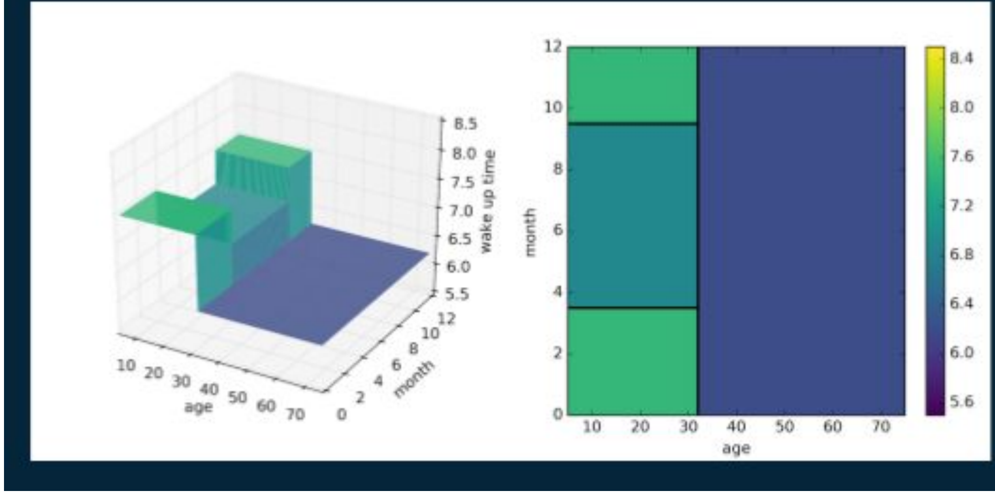
→ no

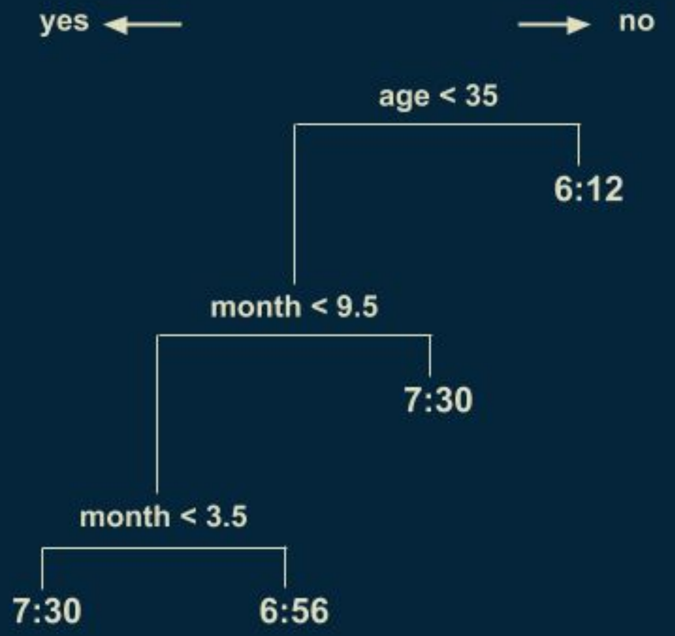
age < 35

7:06

6:12







yes ← → no

age < 35

6:12

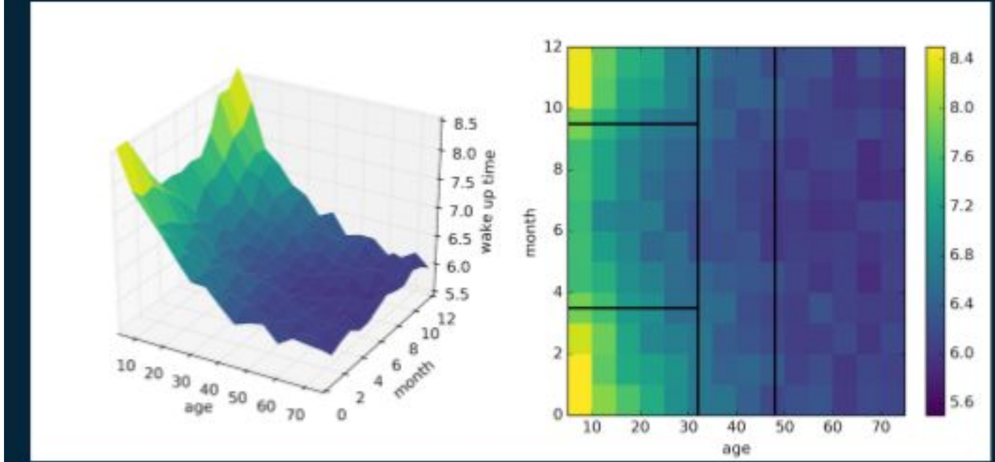
month < 9.5

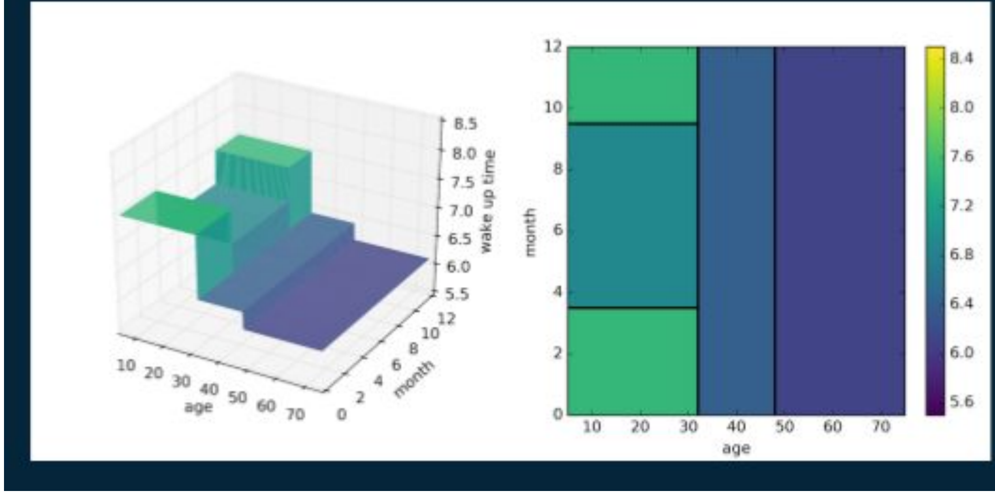
7:30

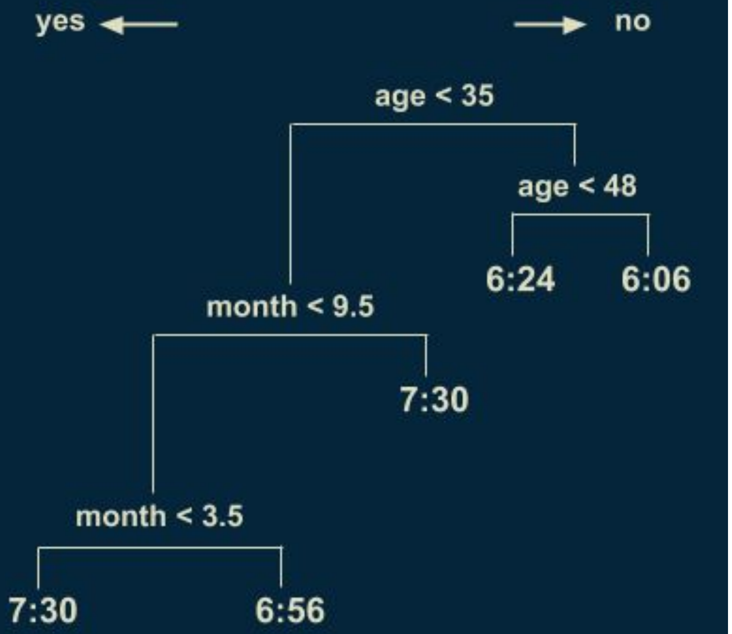
month < 3.5

7:30

6:56






yes ←

→ no

age < 35

age < 48

6:24

6:06

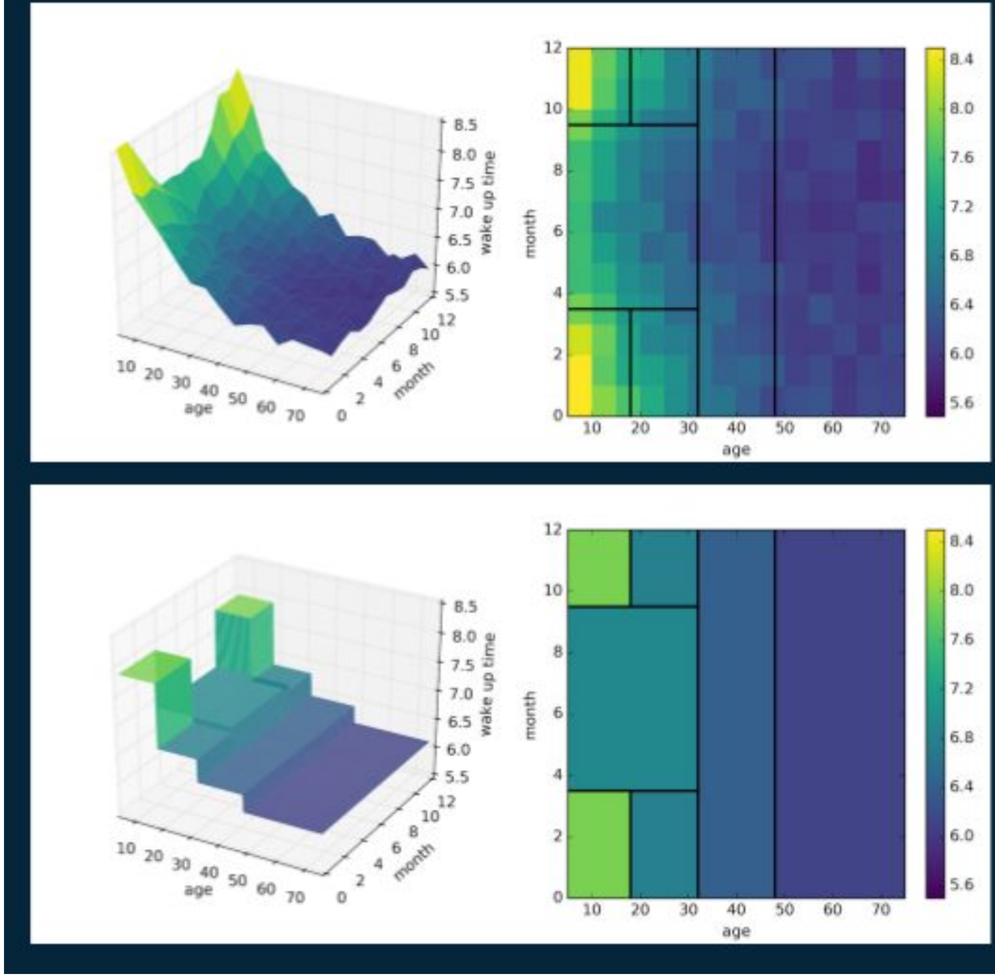
month < 9.5

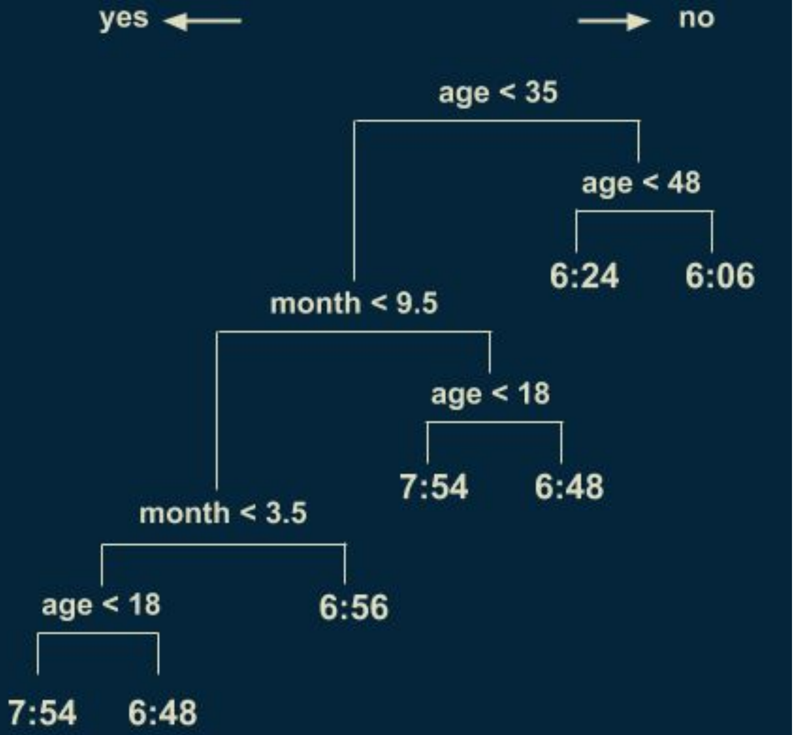
7:30

month < 3.5

7:30

6:56





yes ← → no

age < 35

age < 48

6:24

6:06

month < 9.5

age < 18

7:54

6:48

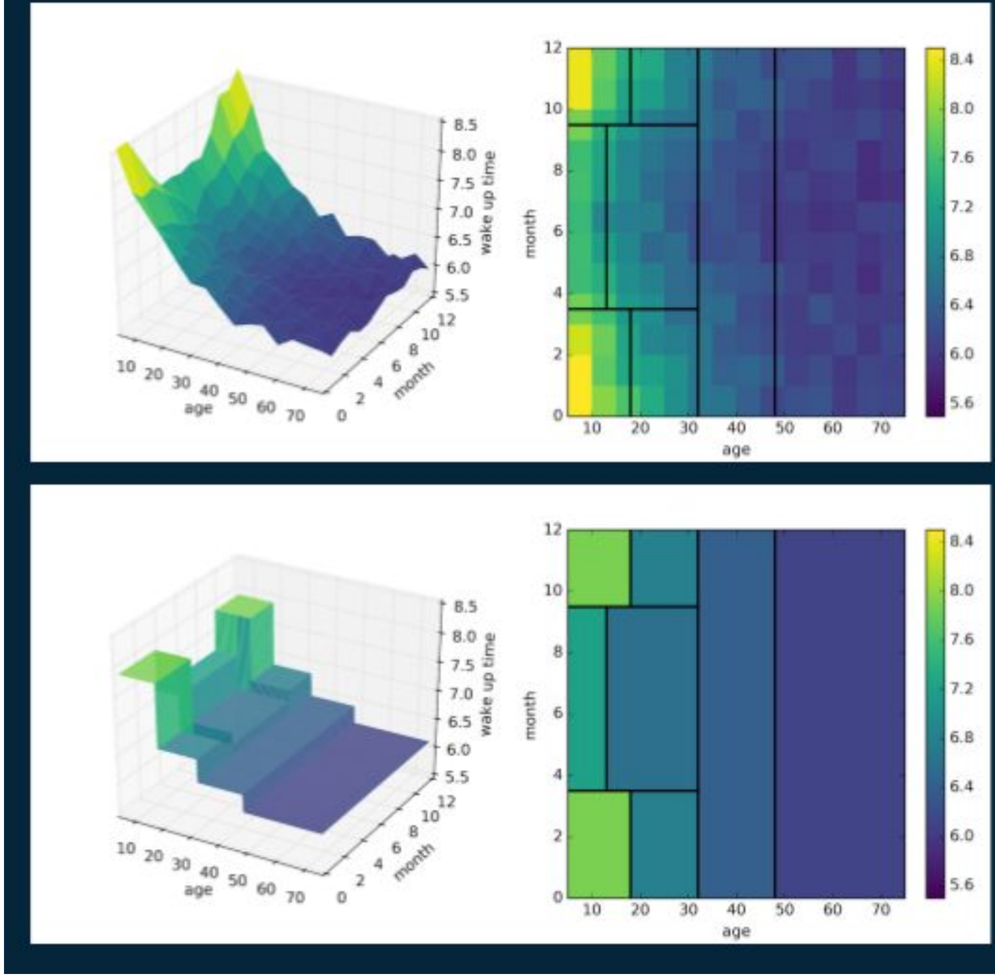
month < 3.5

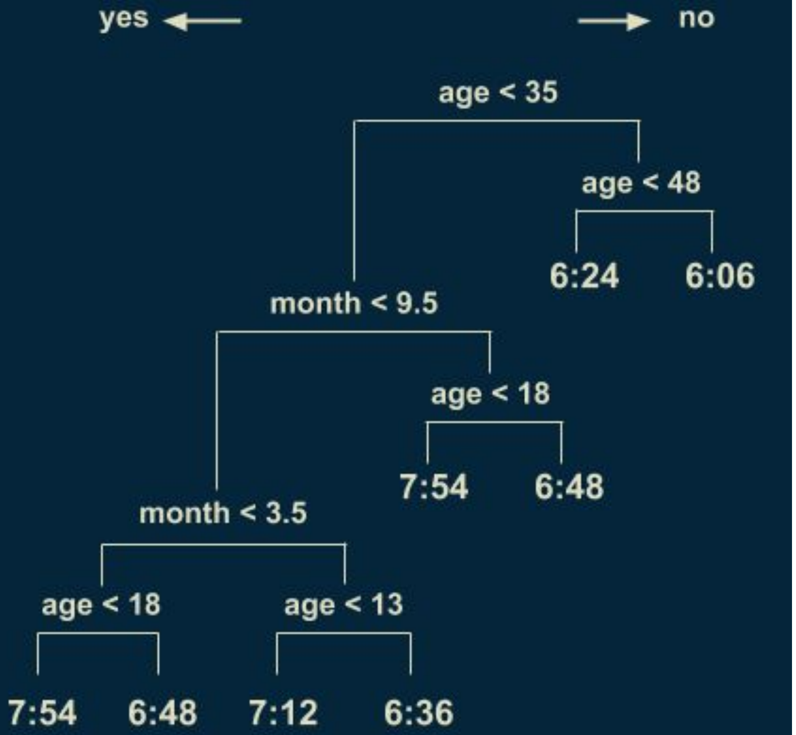
age < 18

6:56

7:54

6:48





yes ← → no

age < 35

age < 48

6:24 6:06

month < 9.5

age < 18

7:54 6:48

month < 3.5

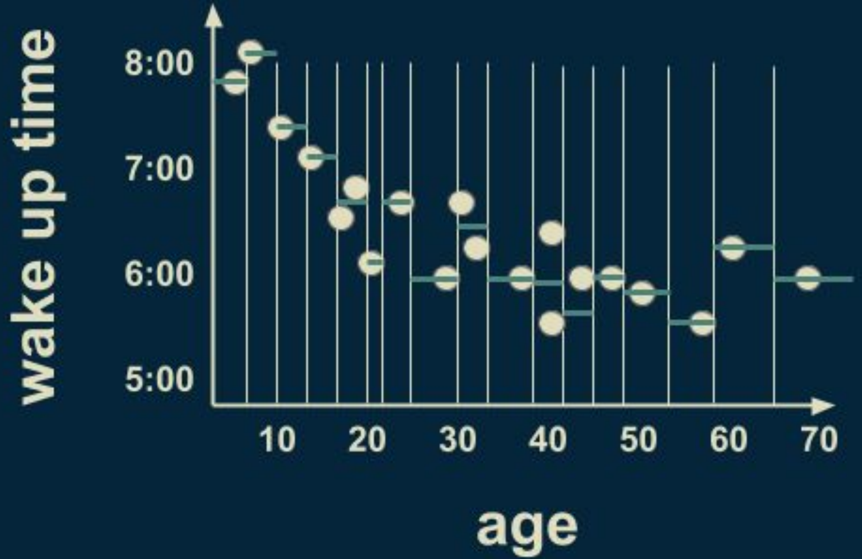
age < 18

age < 13

7:54 6:48 7:12 6:36

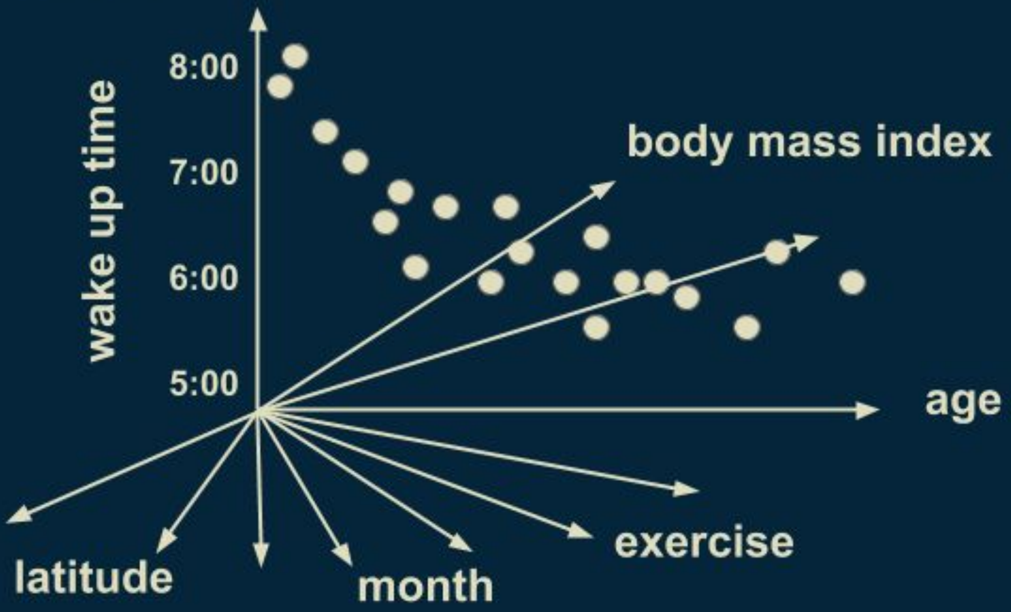
Watch out for

Overfitting / not enough data



Watch out for

Lots of variables



Decision Trees

There are couple of algorithms there to build a decision tree , we only talk about a few which are

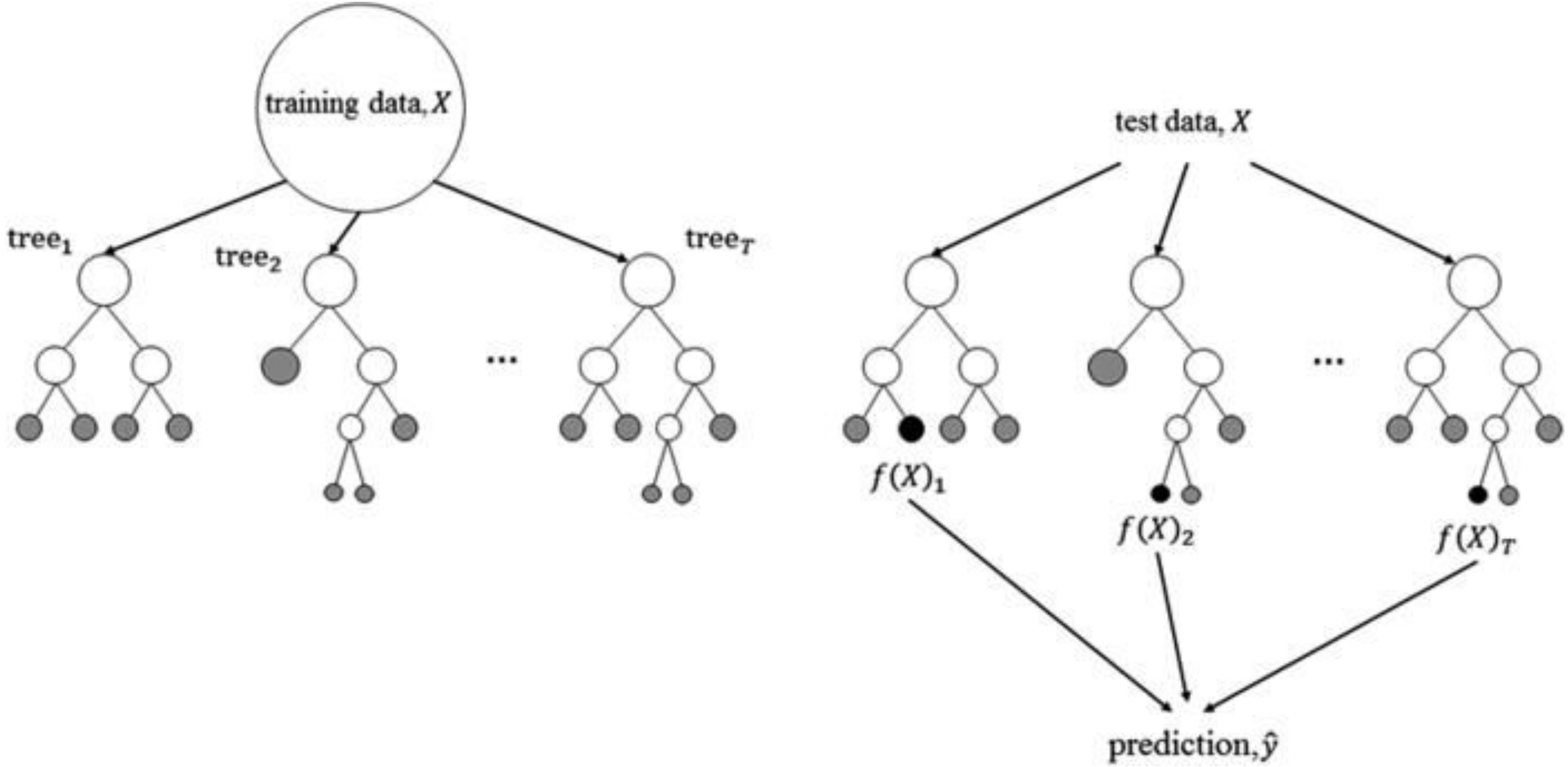
1. CART (Classification and Regression Trees) → uses ***Gini Index(Classification)*** as metric.
   a. The Gini coefficient is a measure of inequality of a distribution
   b. Gini index is Gini Coefficient expressed as percentage
2. ID3 (Iterative Dichotomiser 3) → uses ***Entropy function*** and ***Information gain*** as metrics.

Decision Trees

Two variants:

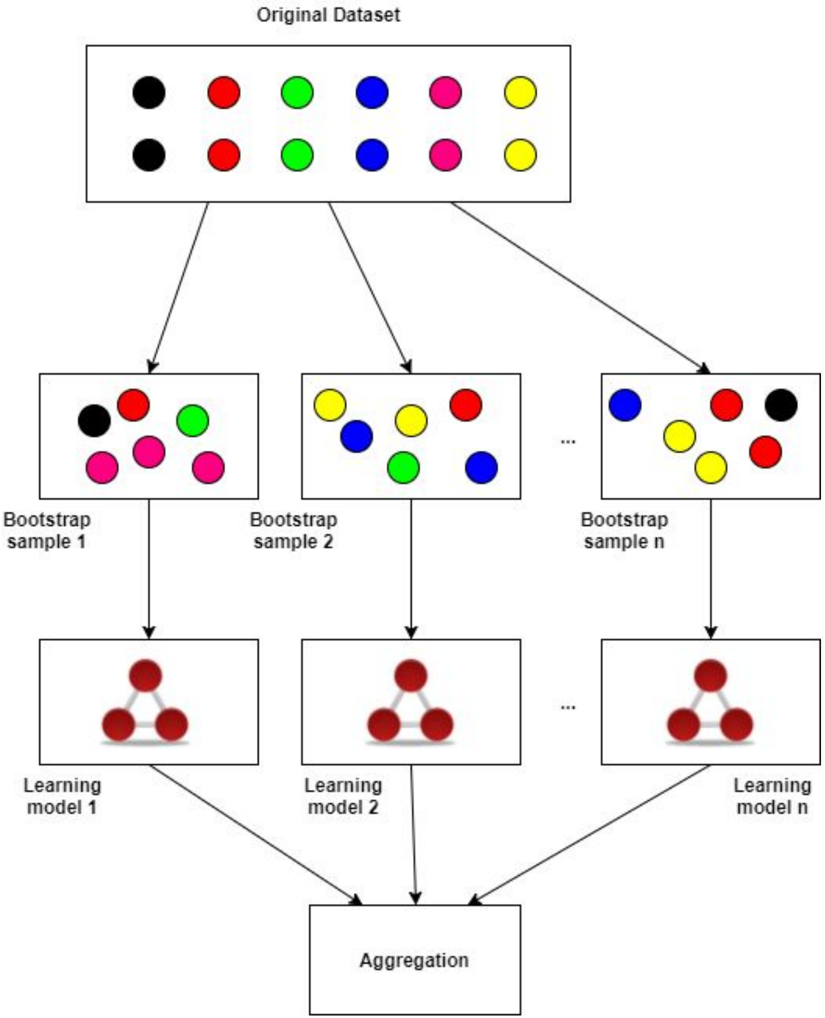
1. Random Forest
2. XgBoost

Random Forest


training data, X
tree$_1$
tree$_2$
...
tree$_T$
test data, X
...
$f(X)_1$
$f(X)_2$
$f(X)_T$
prediction, $\hat{y}$

Bootstrap Aggregation, or bagging is a powerful technique that reduces model variances (overfitting) and improves the outcome of learning on limited sample (i.e. small number of observations)

Bagging works by taking the original dataset and creating *M* subsets each with *n* samples per subset. The *n* individual samples are **uniformly sampled with replacement** from the original dataset



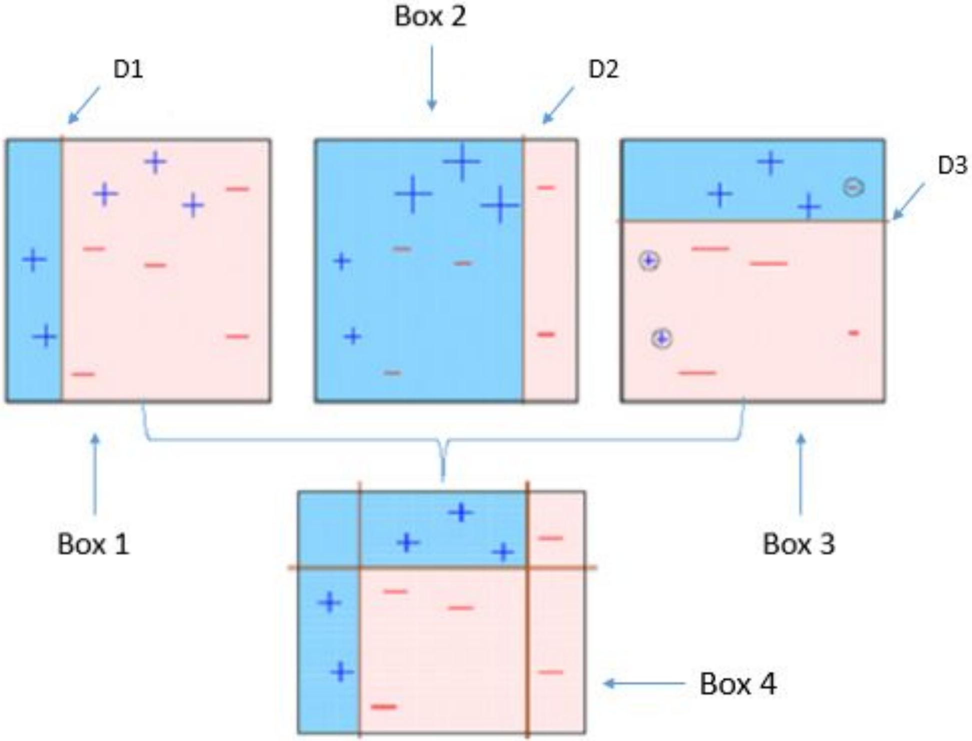
Random forest—a way of bagging trees.

So what is bagging? Bagging is an interesting idea which is what if we created five different models each of which was only somewhat predictive but the models gave predictions that were not correlated with each other. That would mean that the five models would have profound different insights into the relationships in the data. If you took the average of those five models, you are effectively bringing in the insights from each of them. So this idea of averaging models is a technique for **Ensembling**.

XGBoost

（Extreme Gradient Boosting）

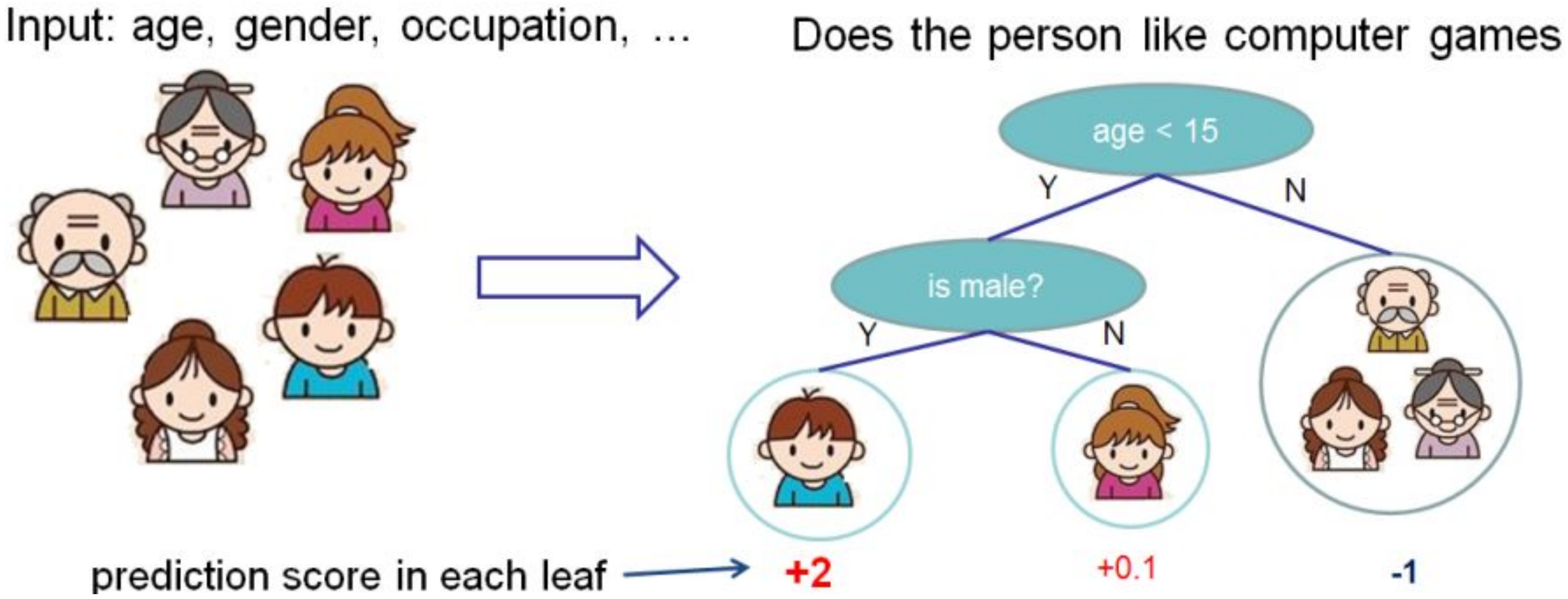
Boosting



Boosting

1. **Box 1**: The first classifier (usually a decision stump) creates a vertical line (split) at D1. It says anything to the left of D1 is + and anything to the right of D1 is -. However, this classifier misclassifies three + points.
2. **Box 2**: The second classifier gives more weight to the three + misclassified points (see the bigger size of +) and creates a vertical line at D2. Again it says, anything to the right of D2 is - and left is +. Still, it makes mistakes by incorrectly classifying three - points.
3. **Box 3**: Again, the third classifier gives more weight to the three - misclassified points and creates a horizontal line at D3. Still, this classifier fails to classify the points (in the circles) correctly.
4. **Box 4**: This is a weighted combination of the weak classifiers (Box 1,2 and 3). As you can see, it does a good job at classifying all the points correctly.

Decision Tree Ensembles

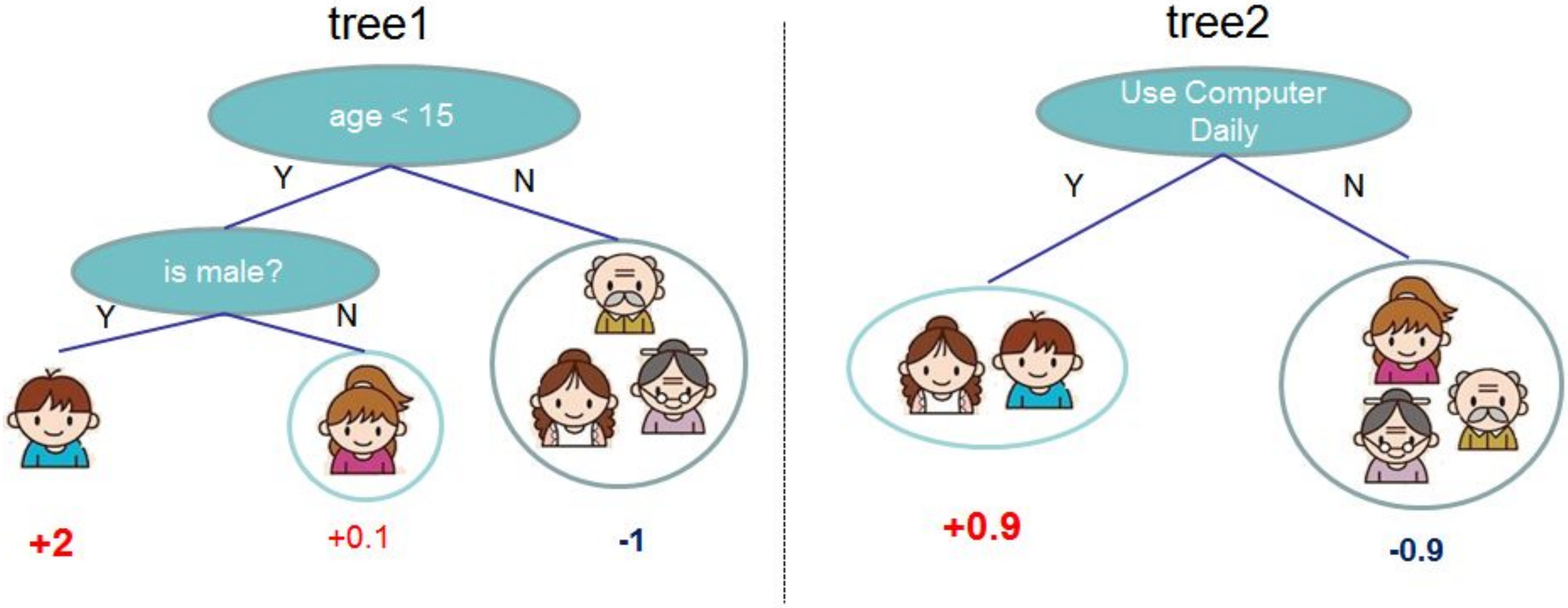


Input: age, gender, occupation, …

Does the person like computer games

prediction score in each leaf

Decision Tree Ensembles



f( ) = 2 + 0.9= 2.9

f( )= -1 - 0.9 = -1.9

Decision Tree Boosting

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

Where, K is the number of trees, f is a function in functional space F, and F is set of all possible CARTs. The objective function to be optimized is given by:

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

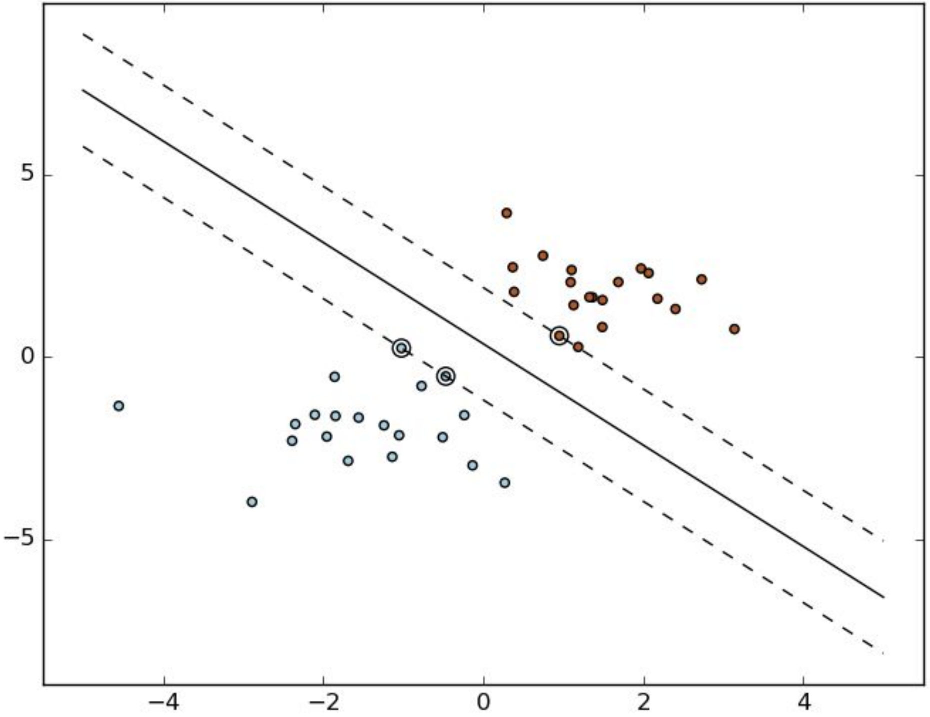
Tree Boosting

Now that we introduced the model, let us turn to training: How should we learn the trees? The answer is, as is always for all supervised learning models: *define an objective function and optimize it*!
Let the following be the objective function (remember it always needs to contain training loss and regularization):

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

SVM

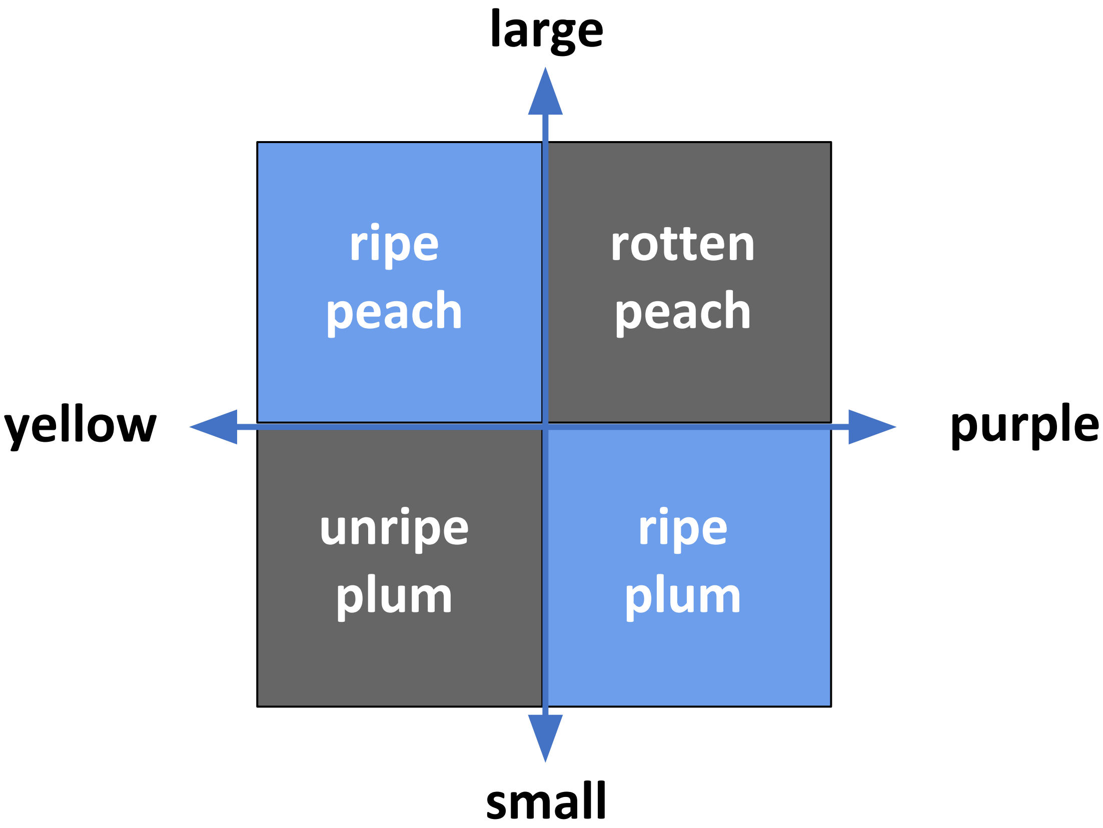
SVM: scikit learn docs



Example

1. A fruit is either
   a. small or large and
   b. yellow or purple.
2. A small yellow fruit is an unripe plum. It is not good to eat.
3. A small purple fruit is a ripe plum. It is good to eat.
4. A large yellow fruit is a ripe peach. It is good to eat.
5. A large purple fruit is a rotten peach. It is not good to eat.

Example



large

yellow

purple

small

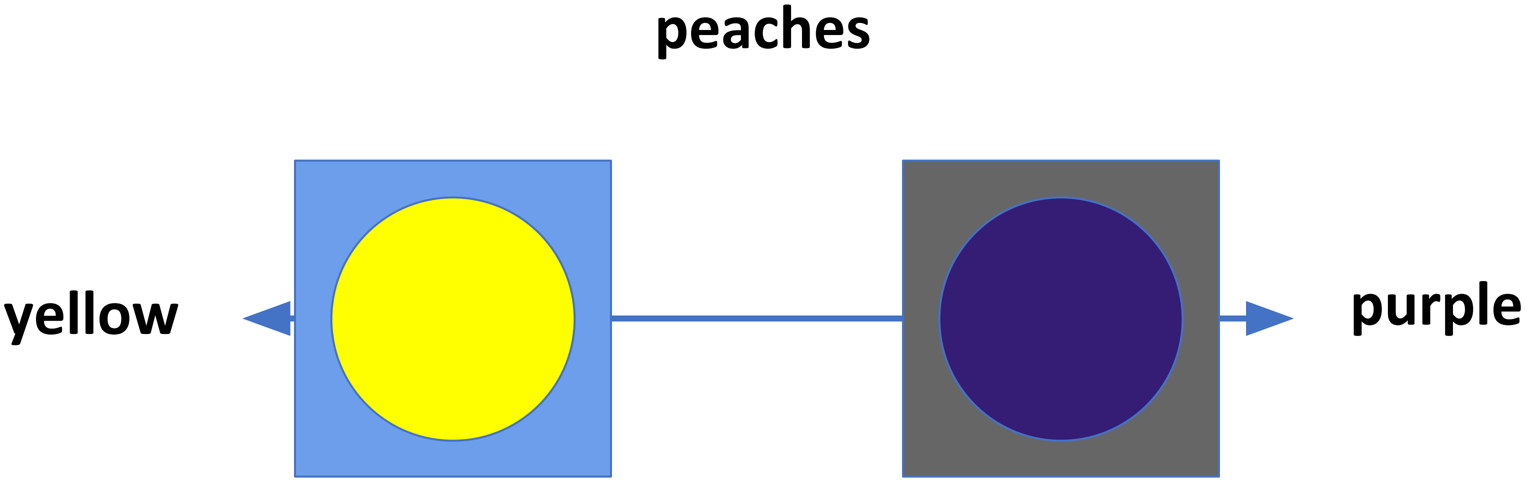
ripe peach

rotten peach

unripe plum

ripe plum

Example



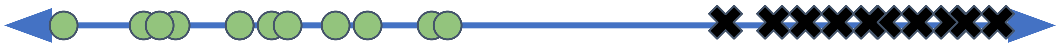
peaches

yellow

purple

Example

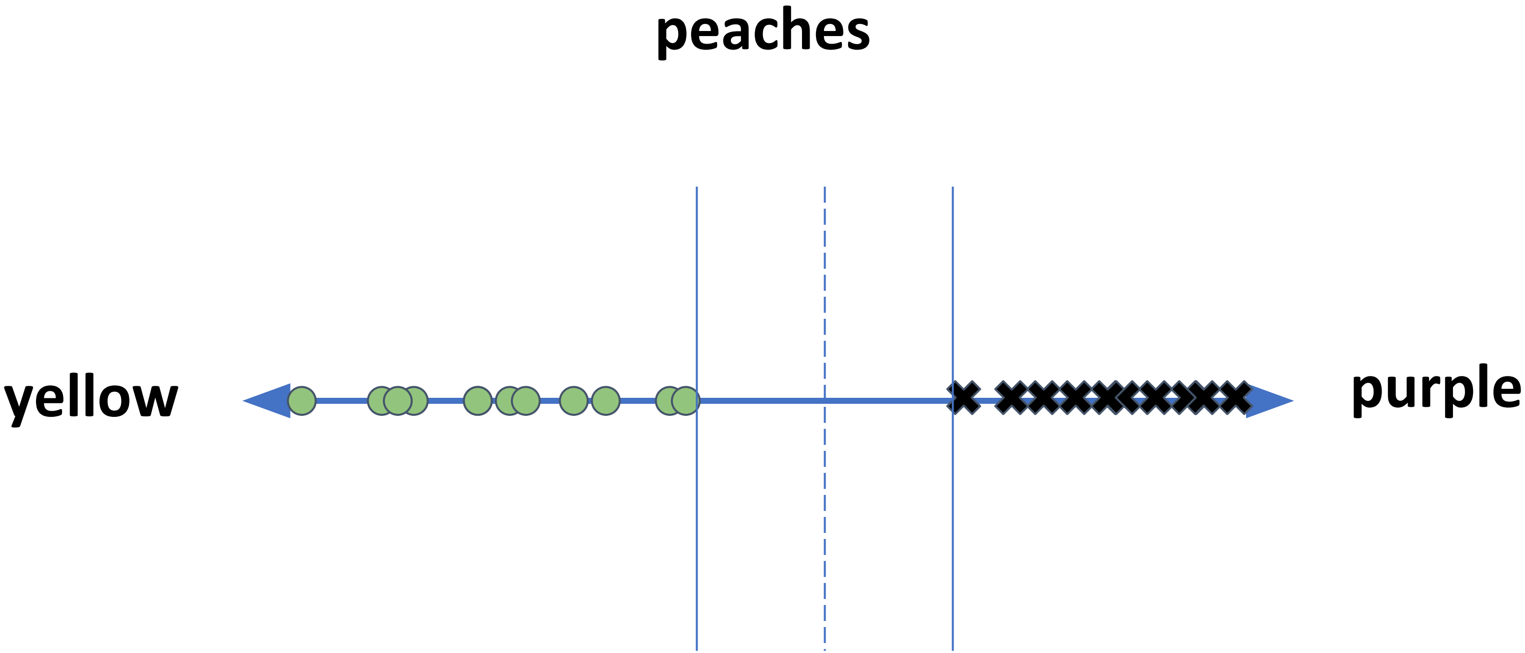
peaches



yellow

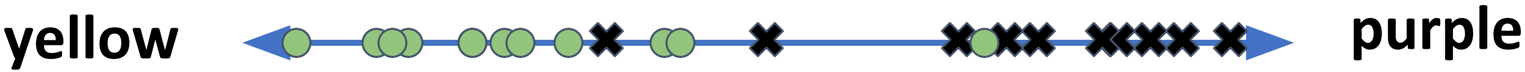
purple

Example



Example

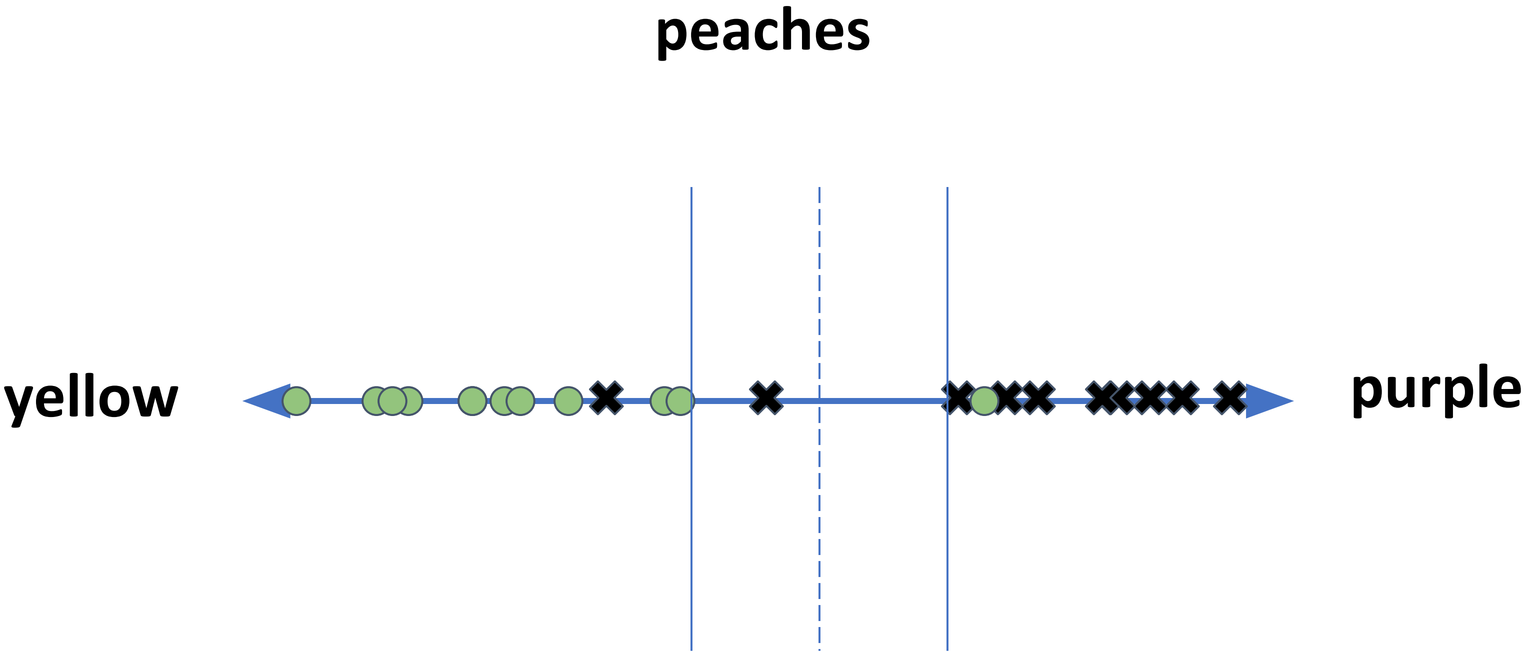
peaches



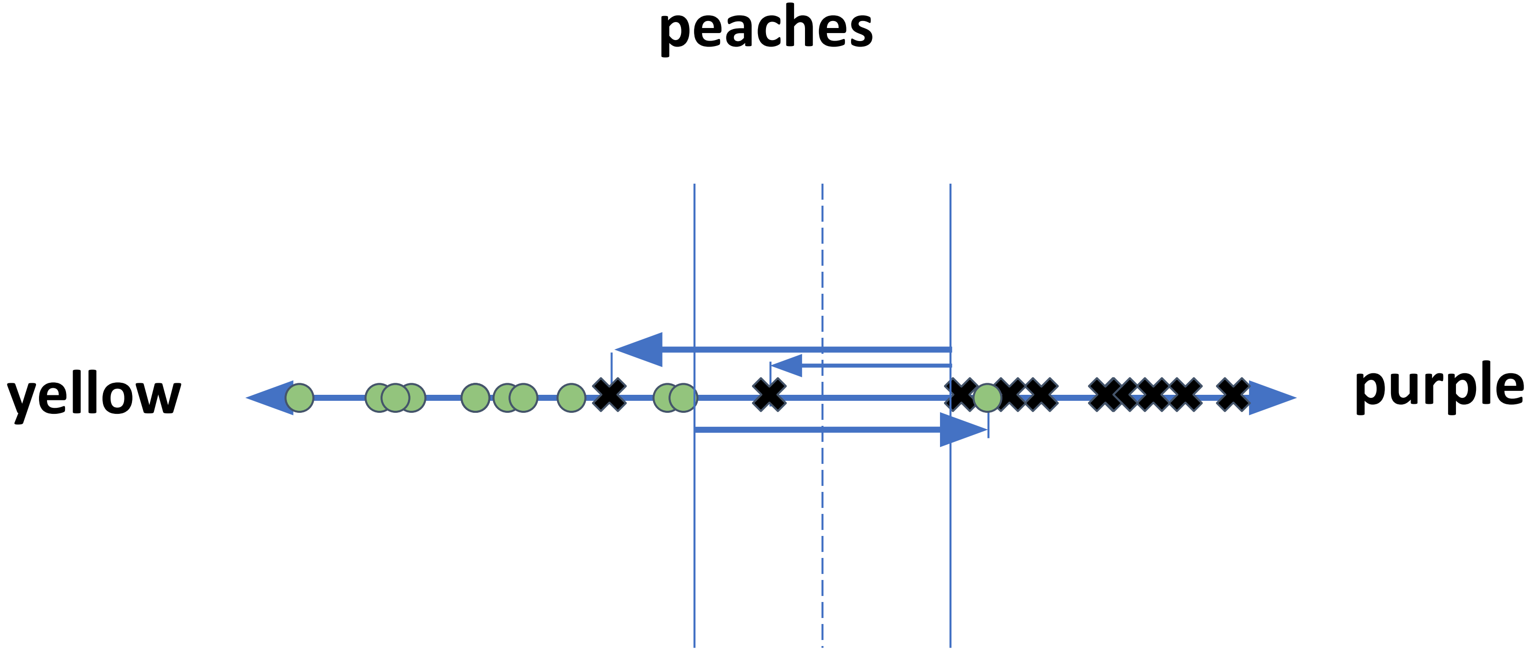
yellow

purple

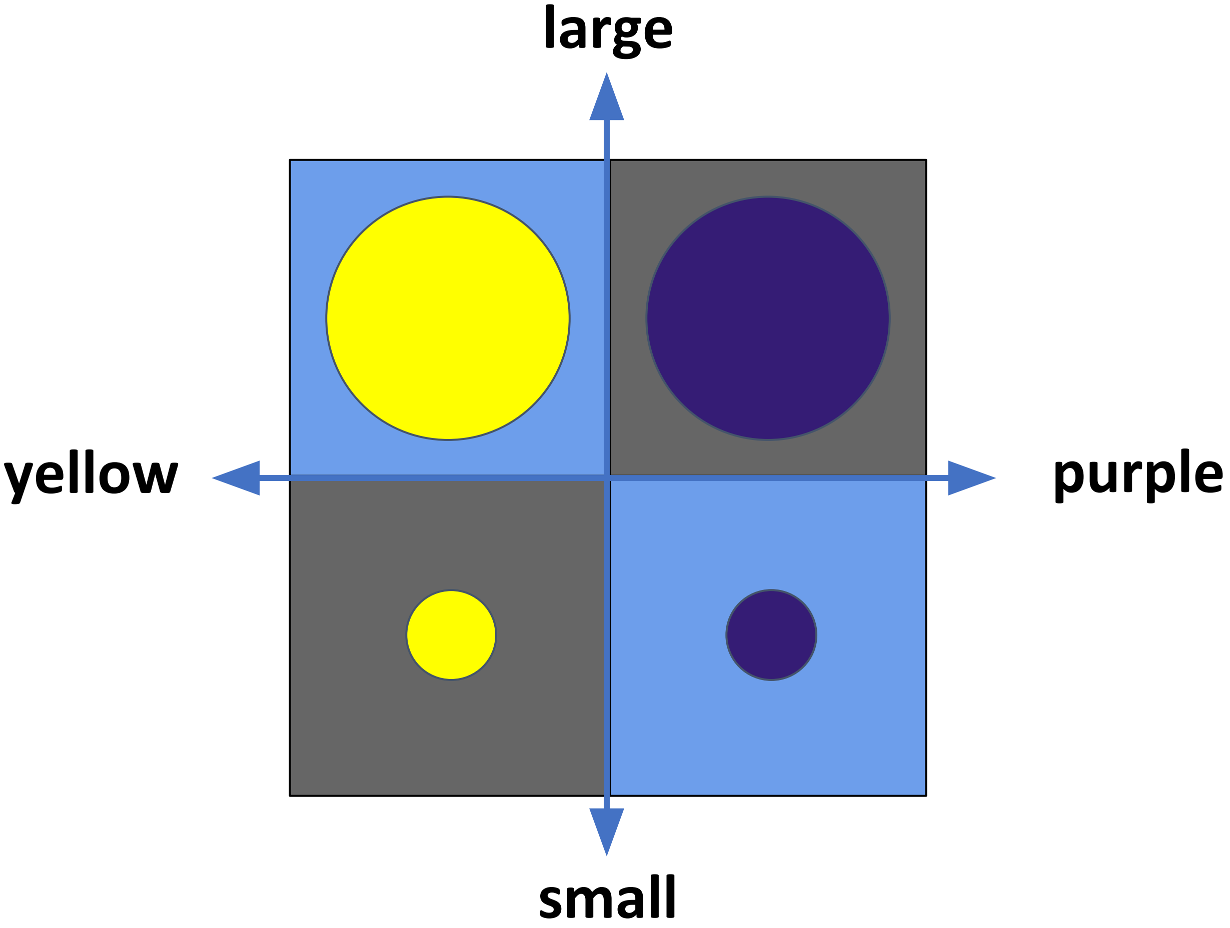
Example



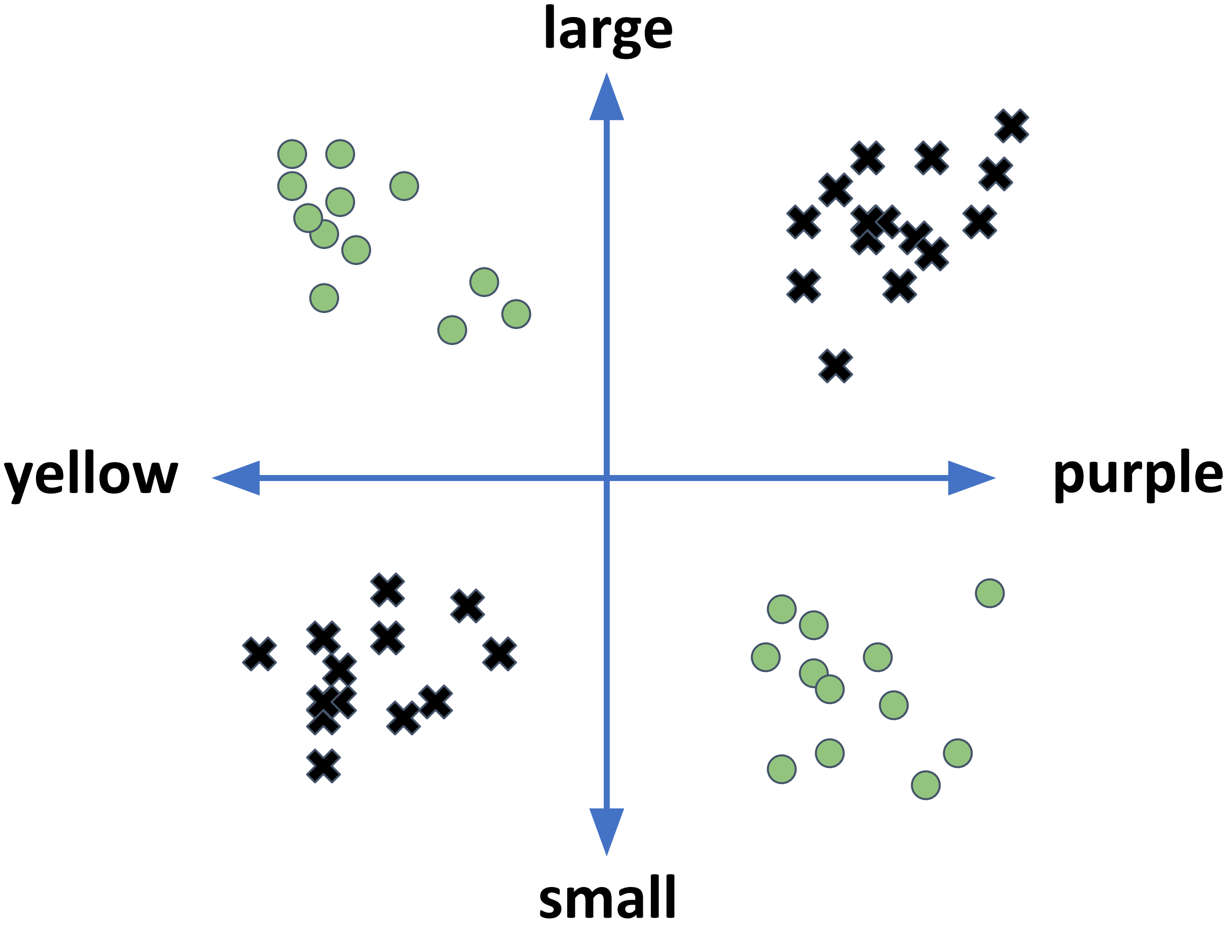
Example



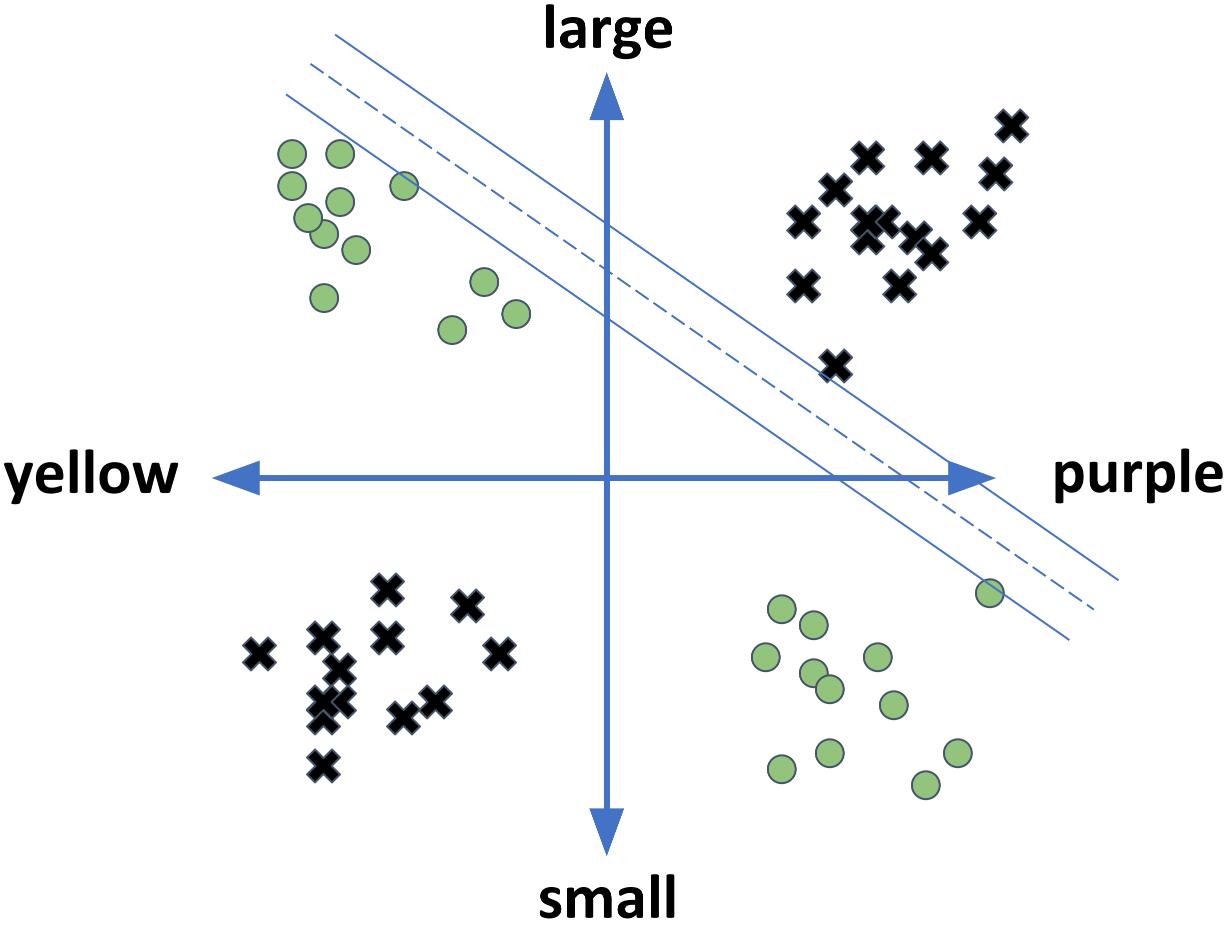
Example



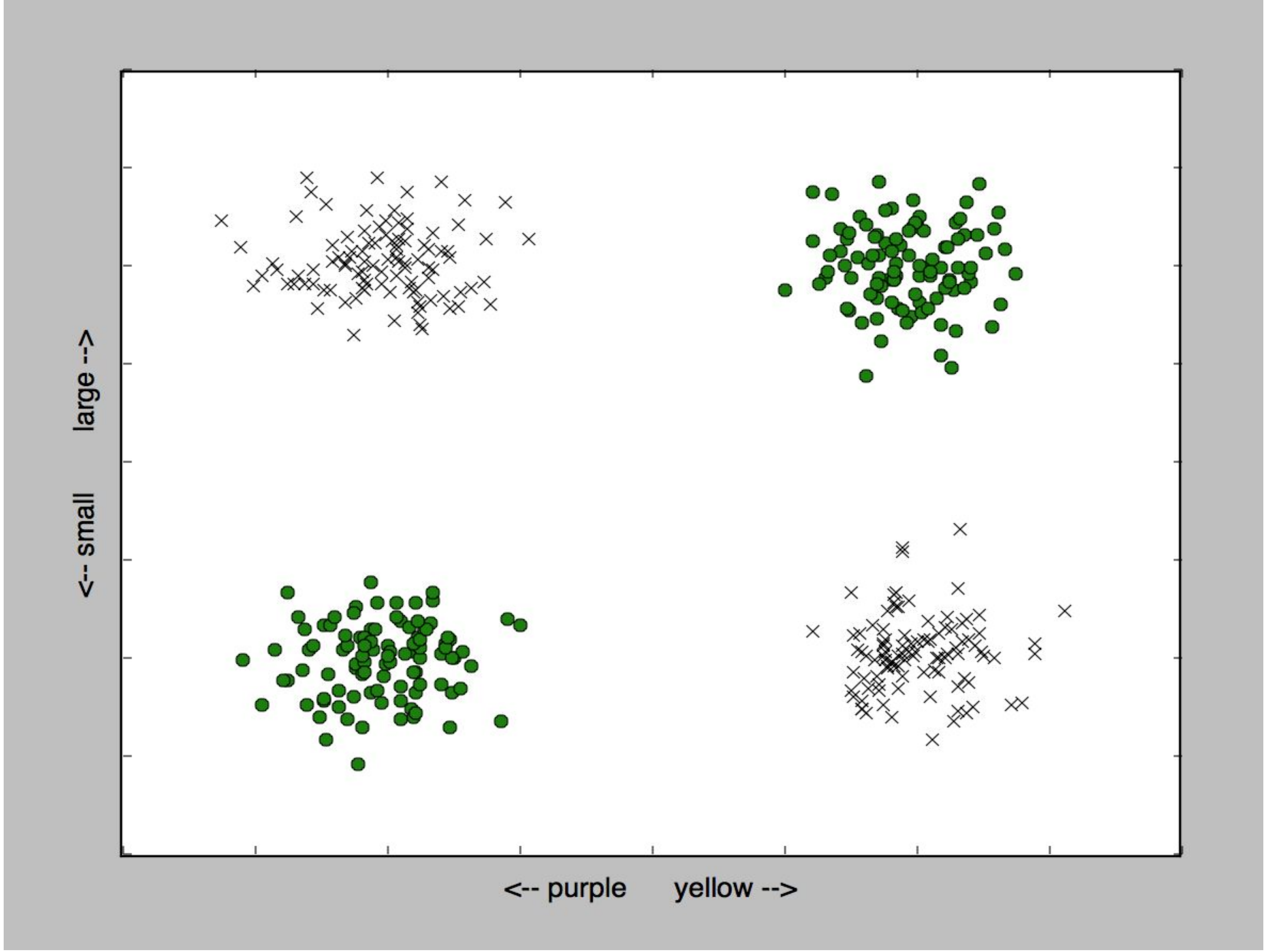
Example



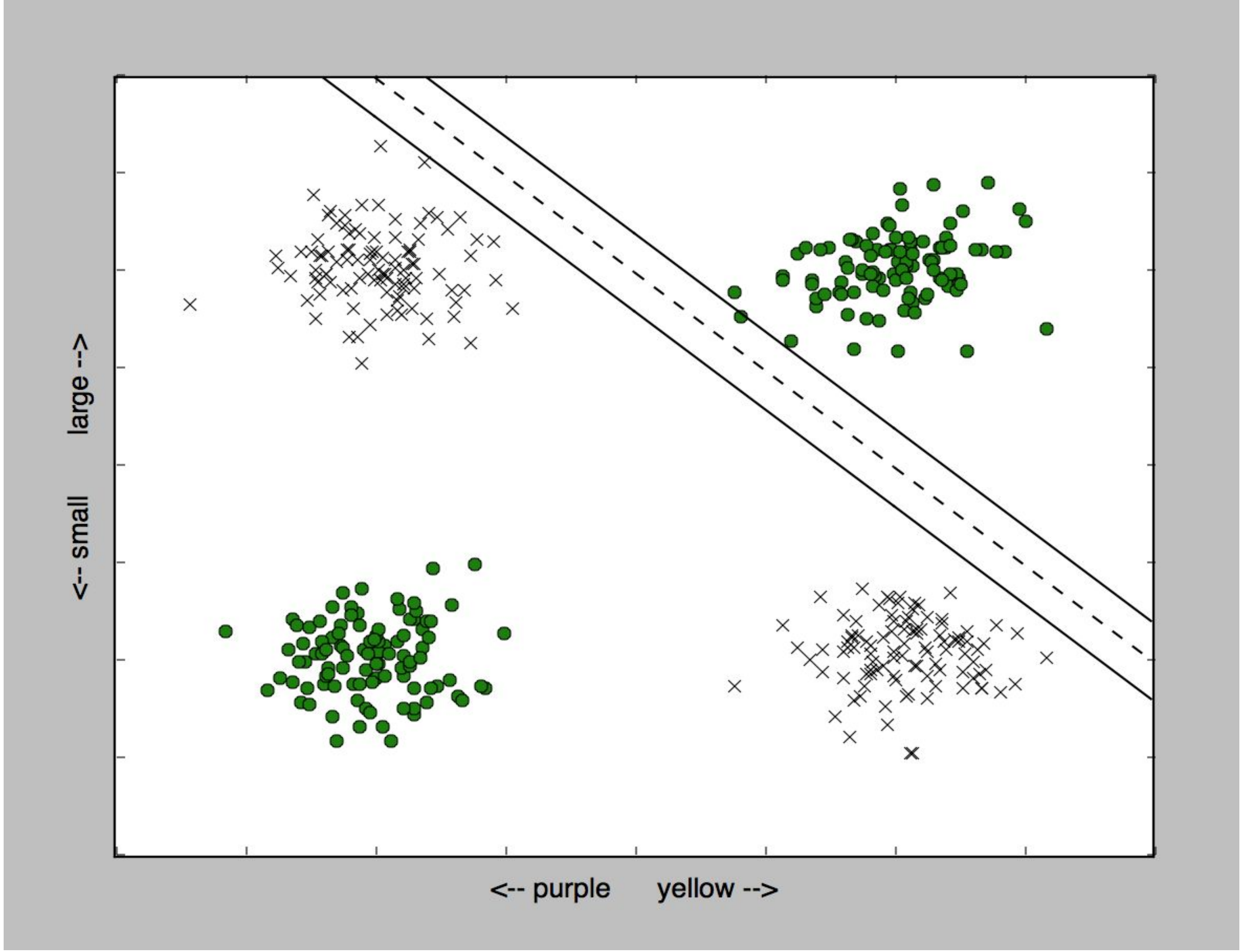
Example



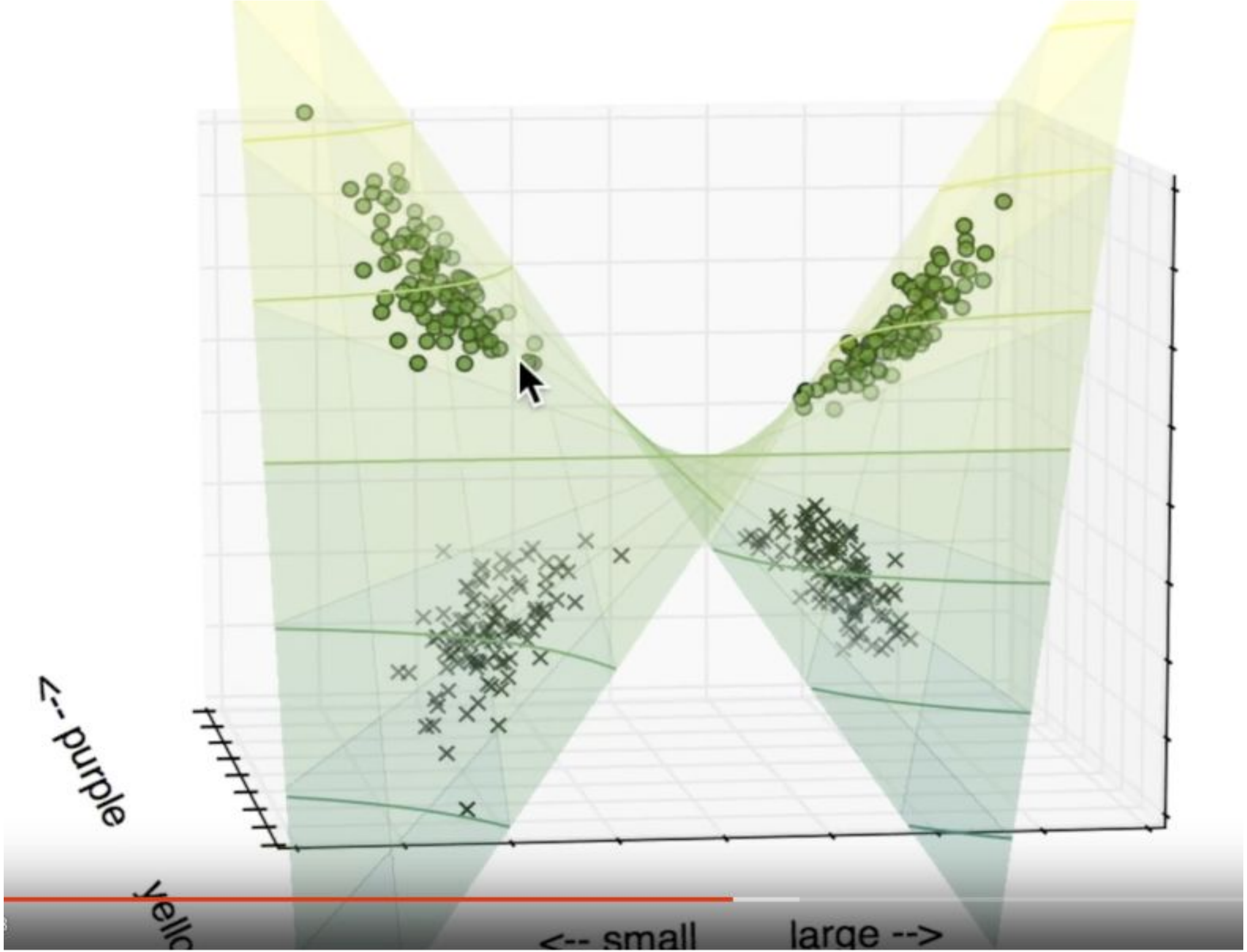
Example



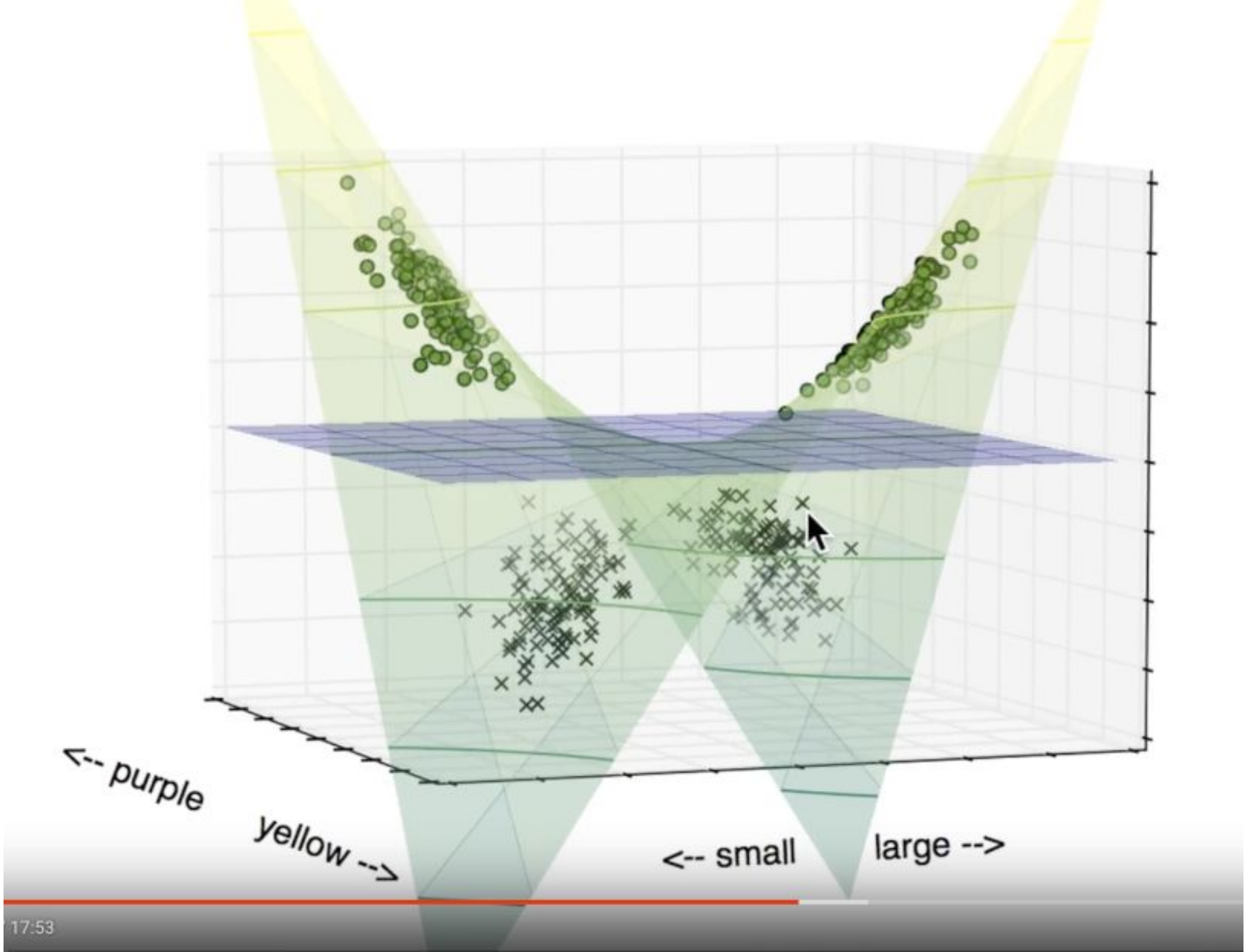
Example

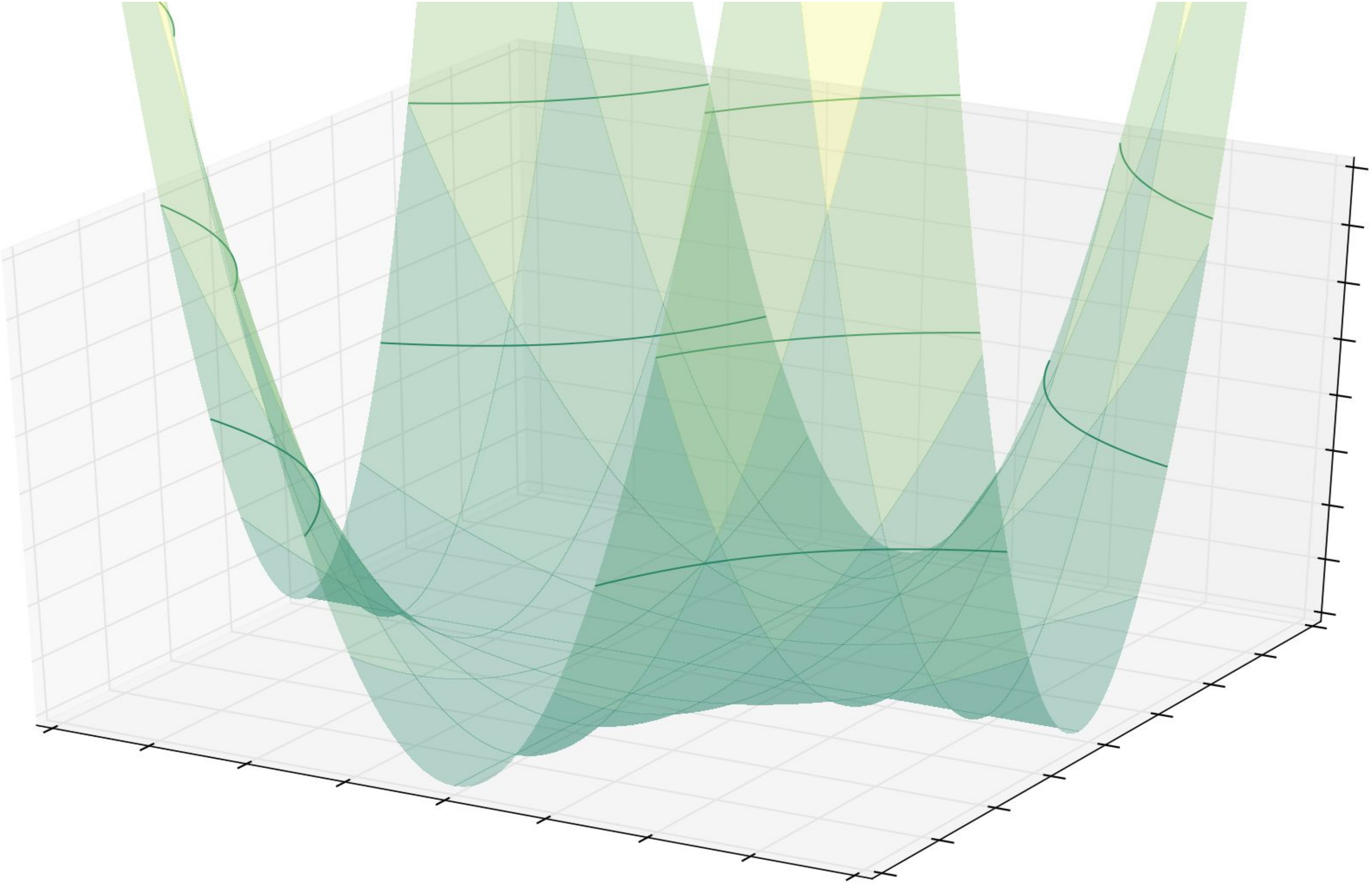


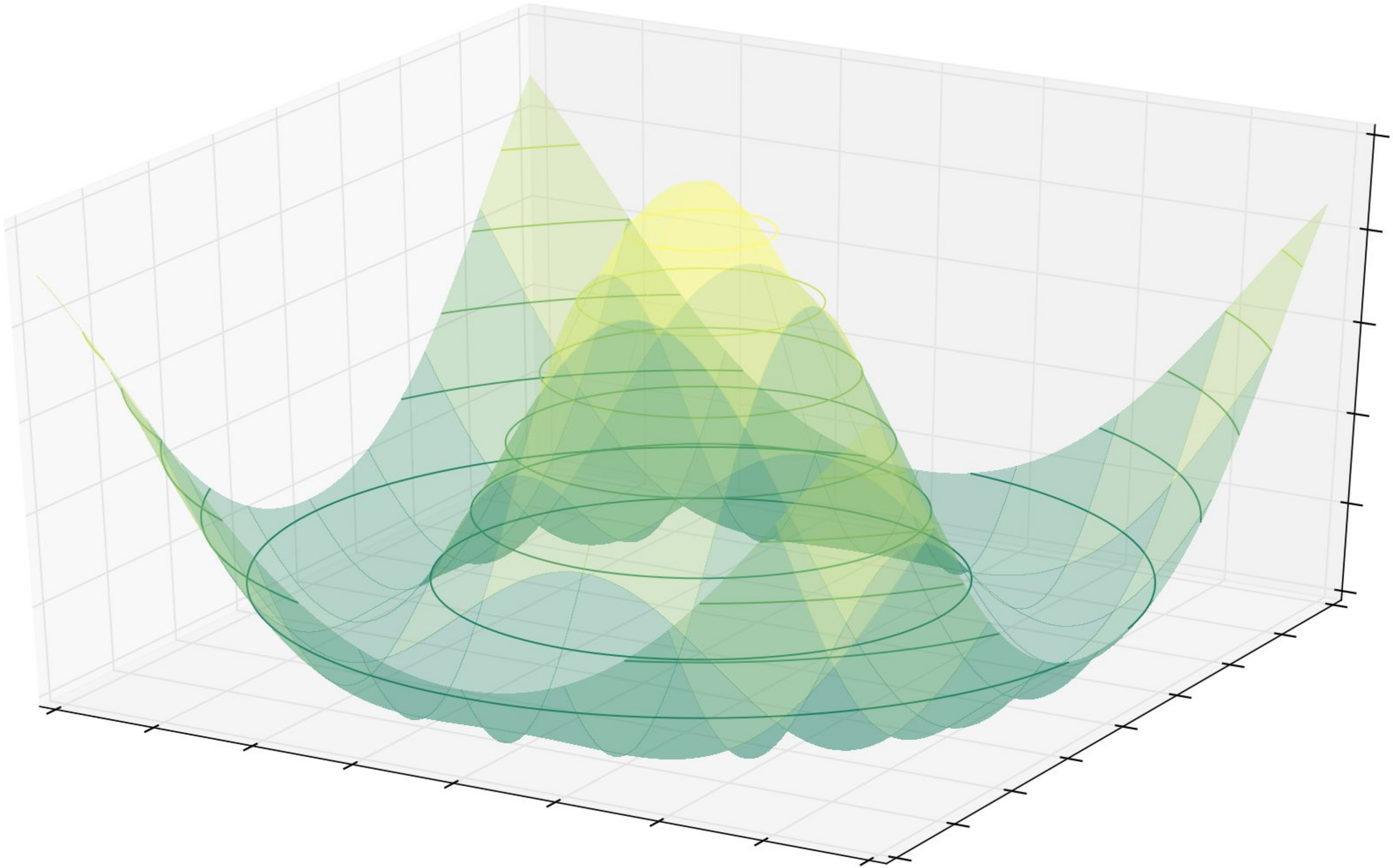
Example

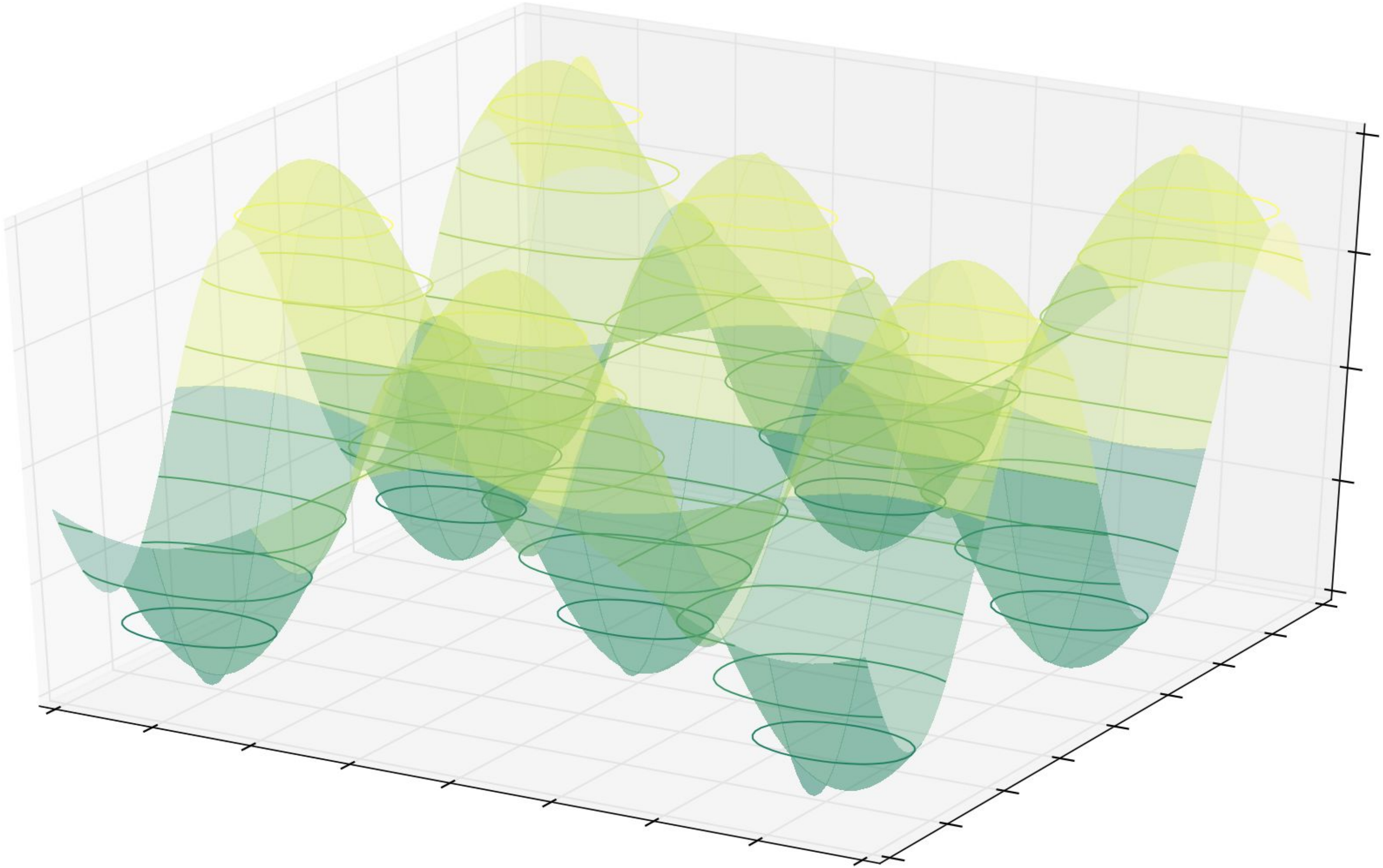


Example

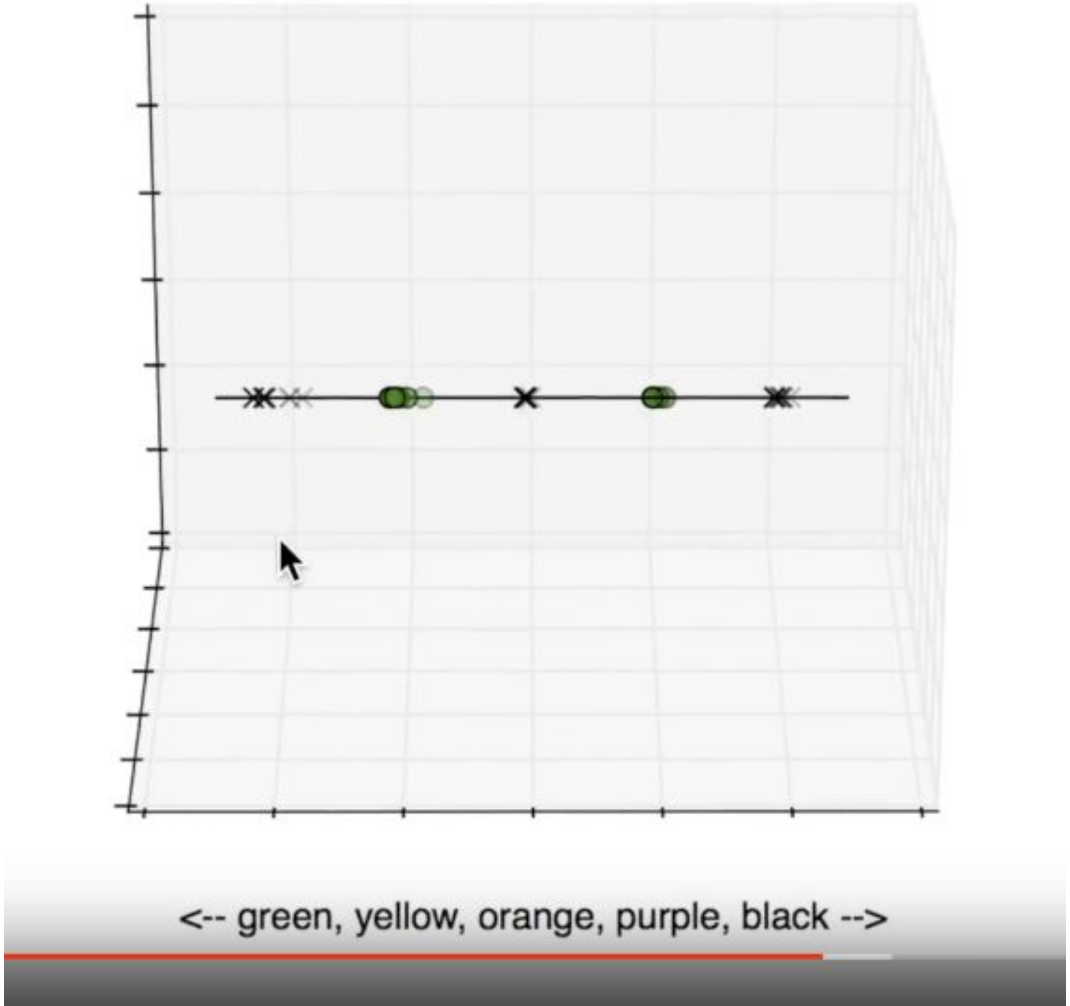




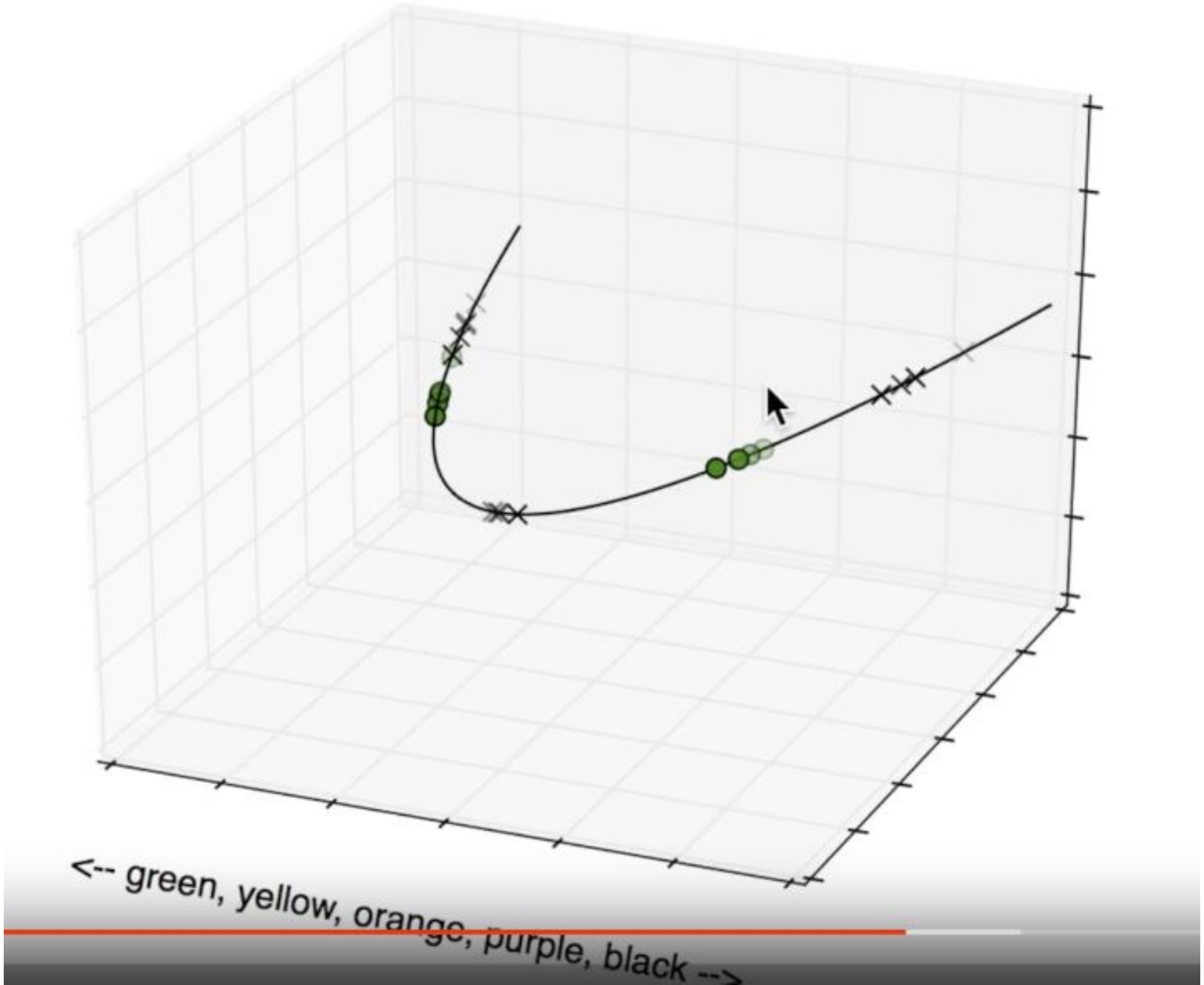




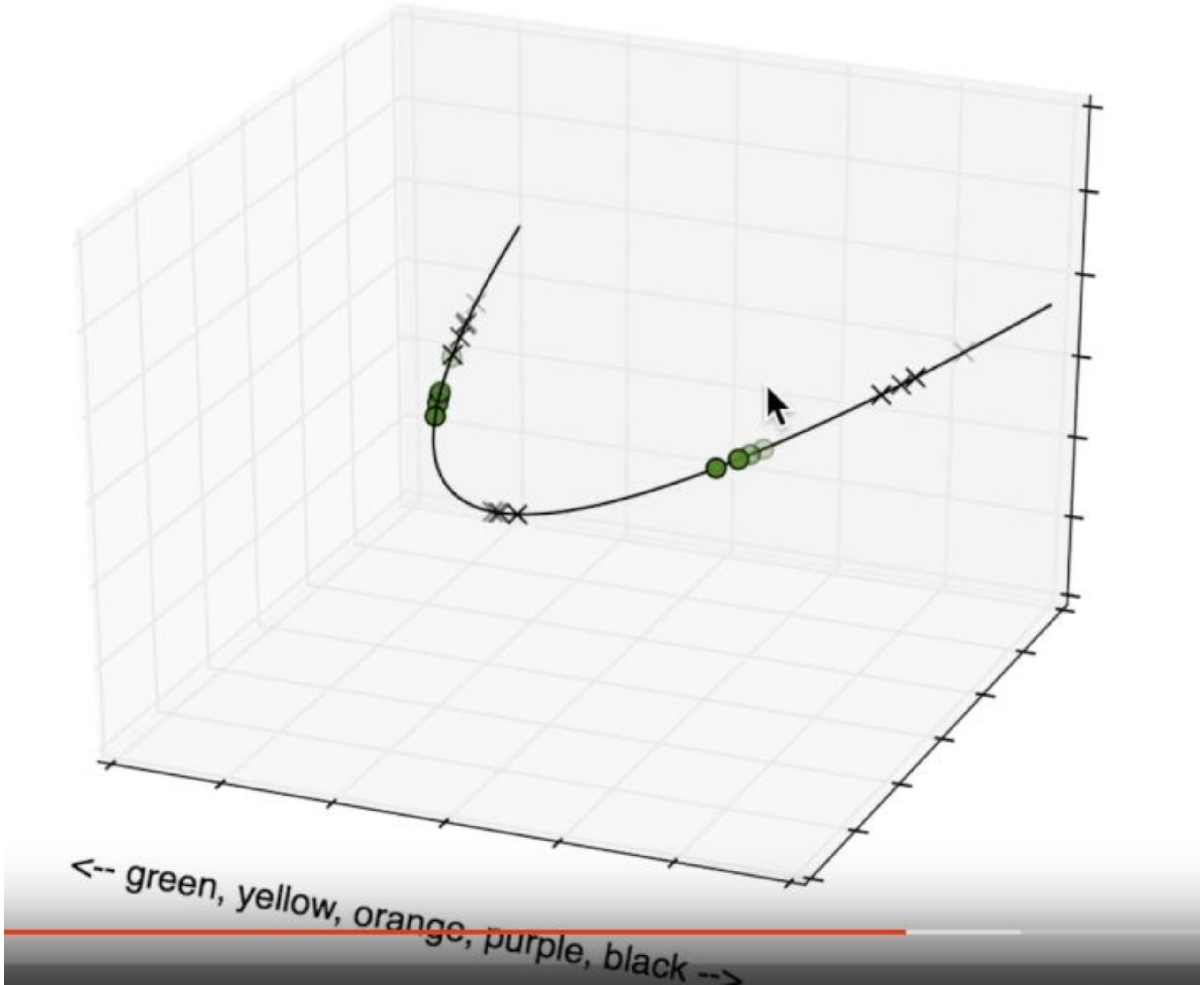
Example



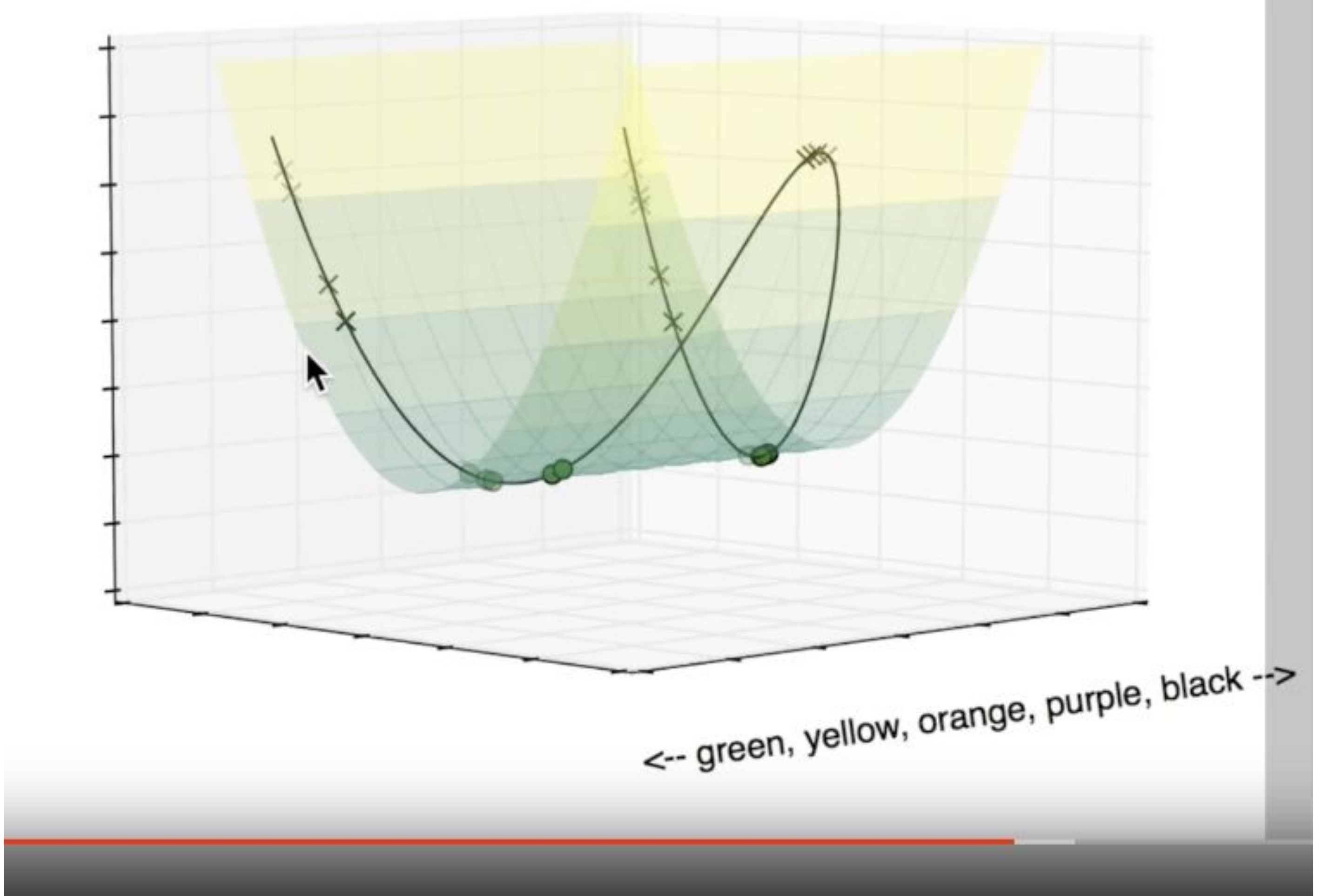
Example



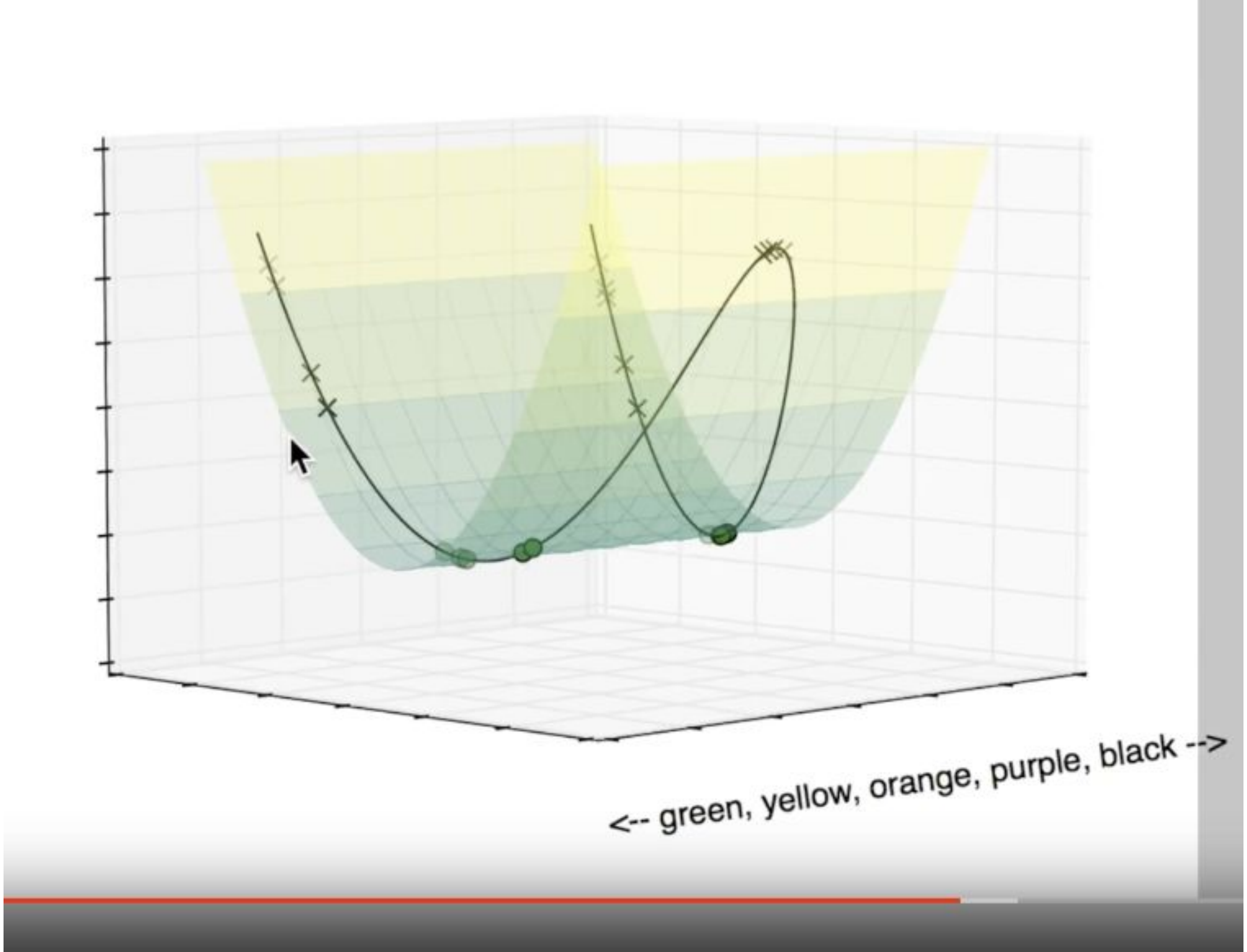
Example



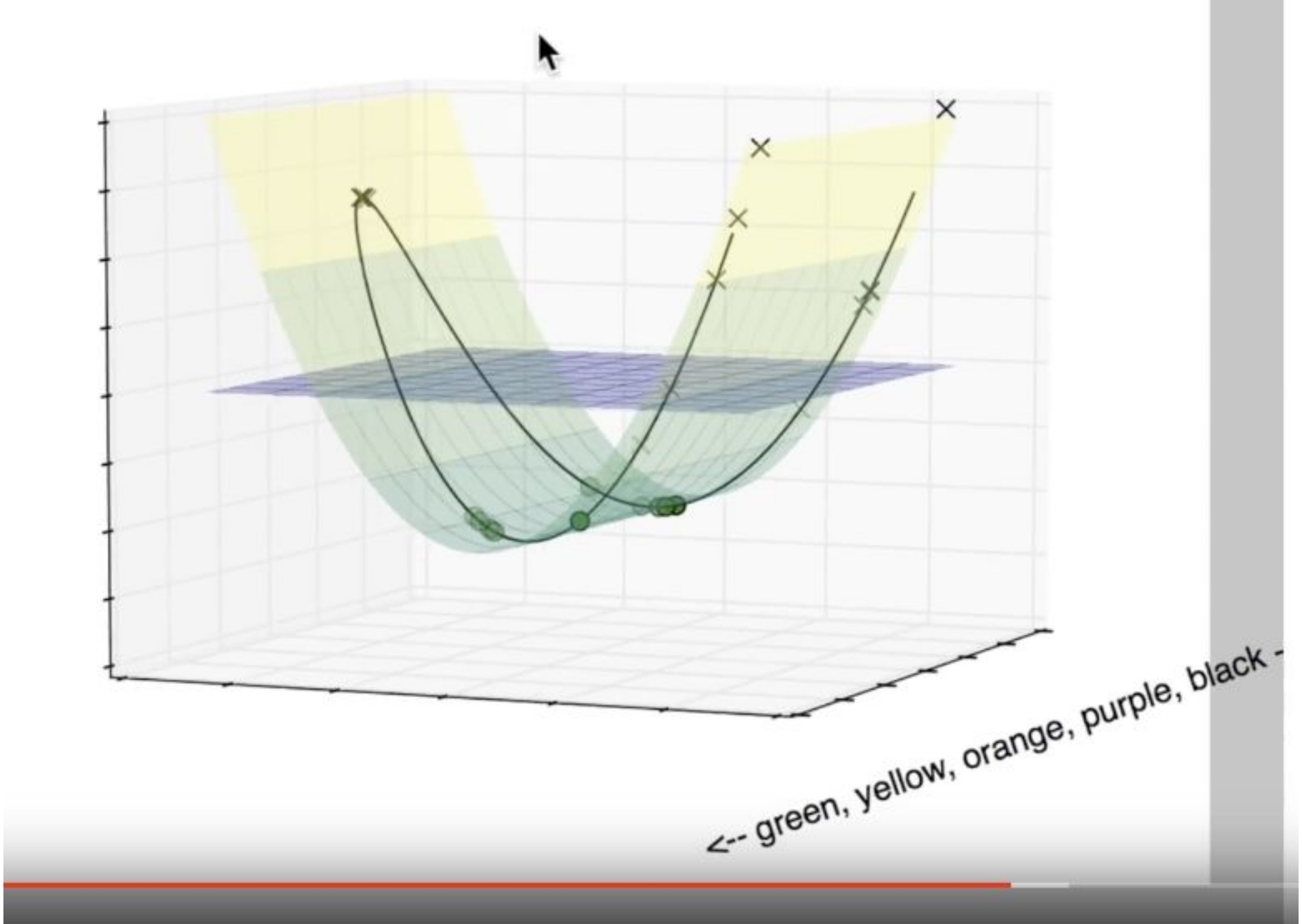
Example



Example



Example



SVM: Breaking Points

1. Data with lots of error.
   a. Discriminator location depends entirely on the few nearest data points.
2. Choosing the wrong kernel.
   a. Kernel selection is trial and error.
3. Large data sets.
   a. Calculating the kernel is expensive.
4. Each of these requires a human in the loop to make judgment calls.

Naive Bayes

Let’s start with bayes theorem: (for naive bayes, x is the input and y is the output)

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

For more than one feature, we can write Baye’s theorem as:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, x_2, \ldots, x_n)}$$

Since, we are making the assumption that X_i’s are conditionally independent given y, we can rewrite the above as

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, \ldots, x_n)}$$

We also know that P(X1, X2, ……. Xn) is a constant given the input

$$P(y|x_1,\ldots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \qquad (1)$$

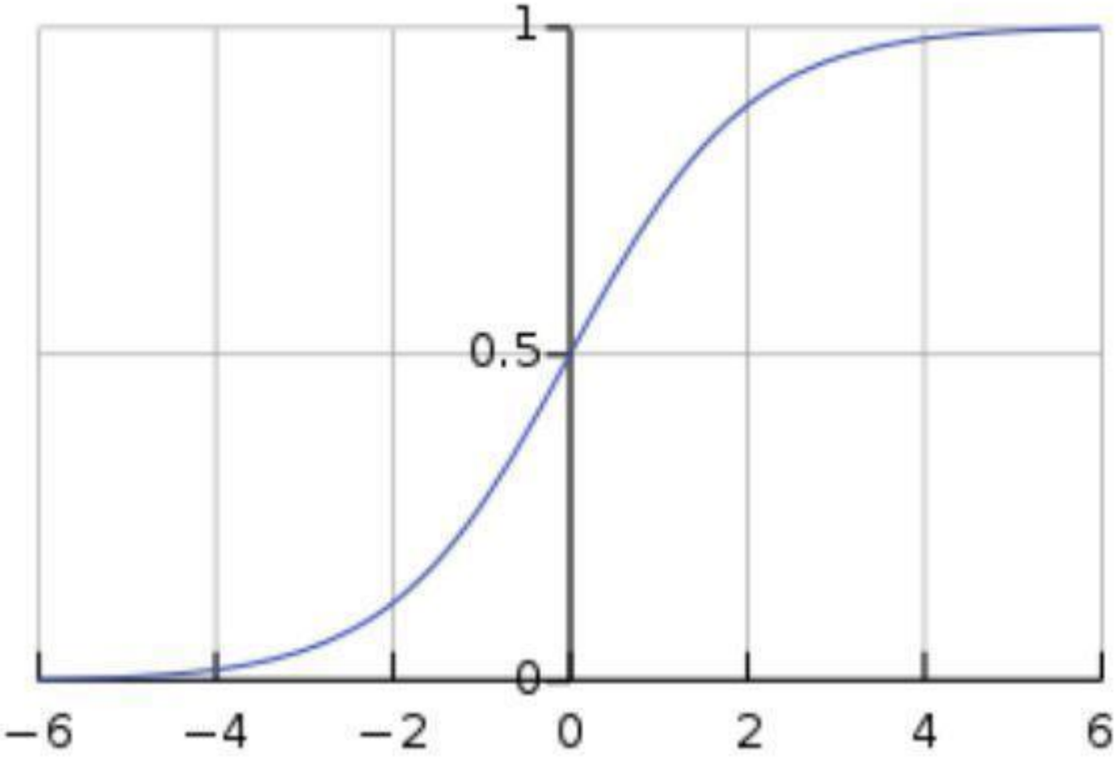
- LHS is the term we are interested in probability distribution of the output Y given input X
- P(y) can be estimated by counting the number of times each class y appears in our training data (this is called Maximum a Posteriori estimation)
- P(xi|y) can be estimated by counting the number of times each value of xi appears for each class y in our training data

Logistic Regression

Don’t be confused by the term “Regression” it’s a classification algorithm
(Variant of Linear Regression)

The logistic(or Sigmoid) Function

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



- The sigmoid function squashes the input value between [0,1].
- Since the range of output is between 0 and 1, we can interpret the output as a probability.
- The logistic function also has the desirable property that it is a differentiable function. Hence, we can train the machine learning model using gradient descent

Logistic Regression Model (for binary classification)

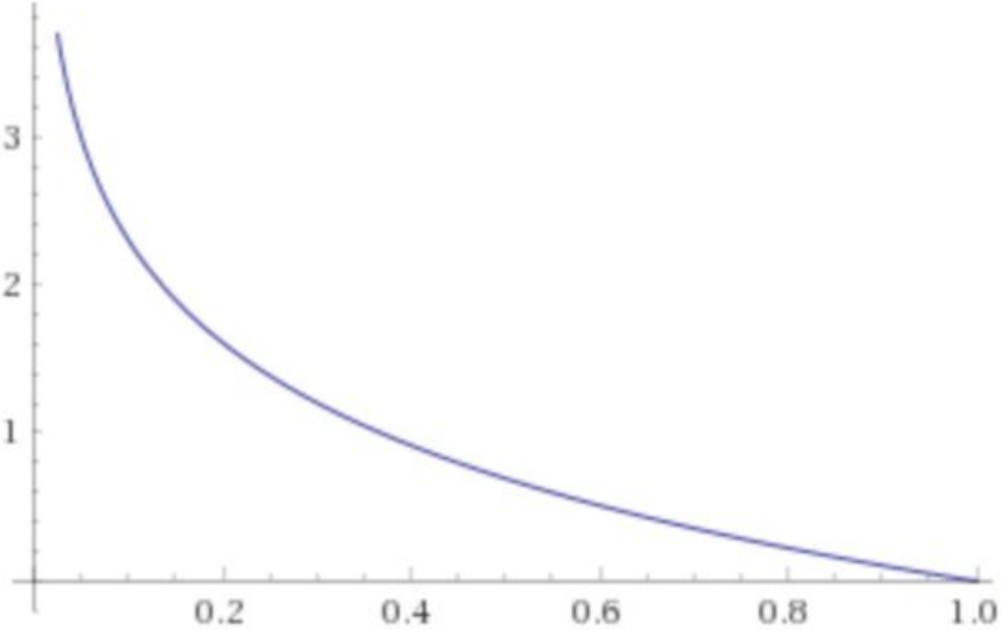
In logistic regression, the output $y_w(x)$ is squashed by a sigmoid function, i.e.

$$y_w(x) = \sigma(w^\top x) = \frac{\exp(w^\top x)}{\exp(w^\top x) + 1}$$

Cost Function

$$L(w) = -\frac{1}{n}\sum_i (y_{true}^{(i)} \log(y_w(x^{(i)})) + (1 - y_{true}^{(i)}) \log(1 - y_w(x^{(i)}))$$

The figure is a plot of negative log probability. As we can see, this cost is high when the target class is assigned a low probability, and is 0 if the assigned probability is 1.



Training the model

We use gradient descent to optimize the model. In fact, the cost function above is chosen so that the gradients **dL/dw** we get are meaningful.