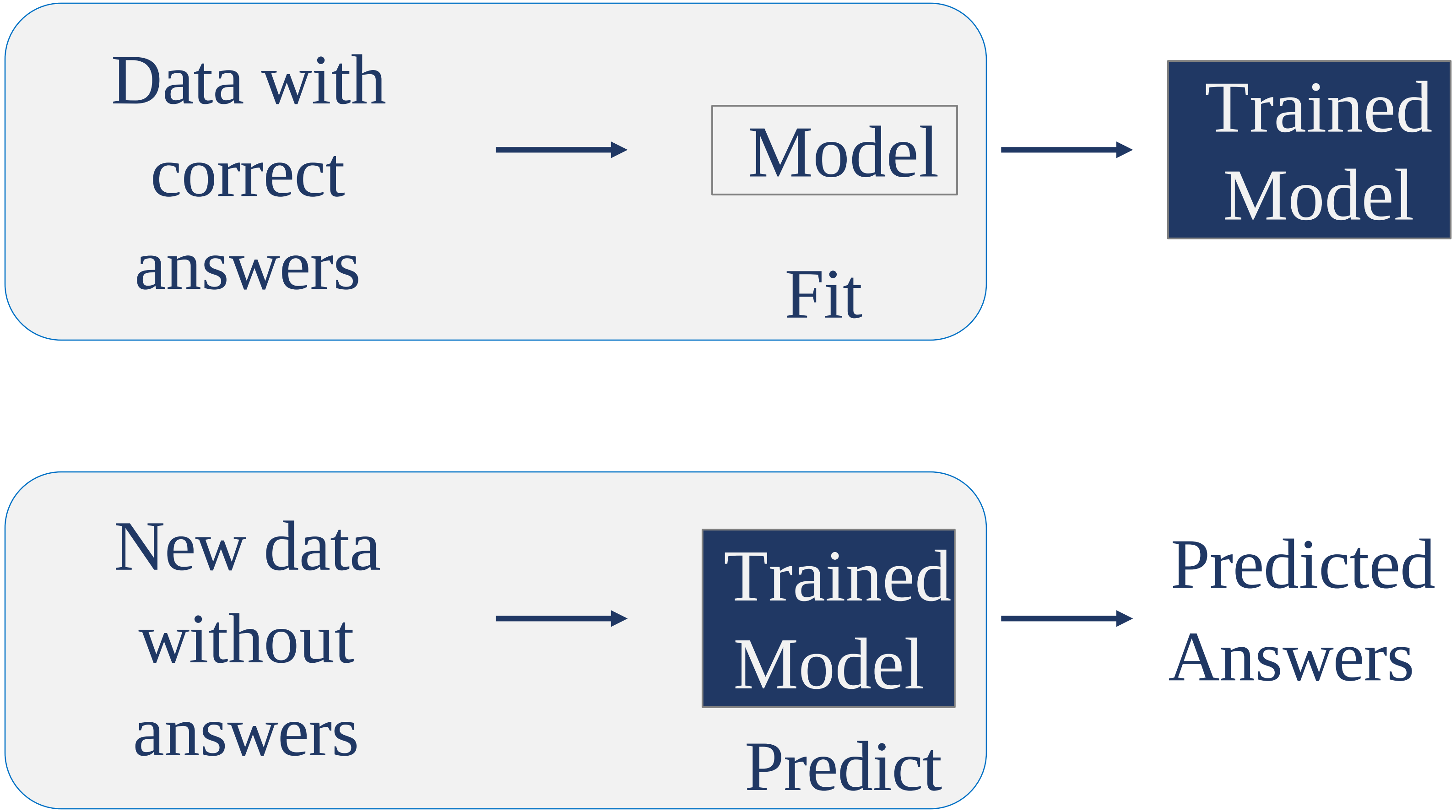
Intel AI Academy

LEARNING

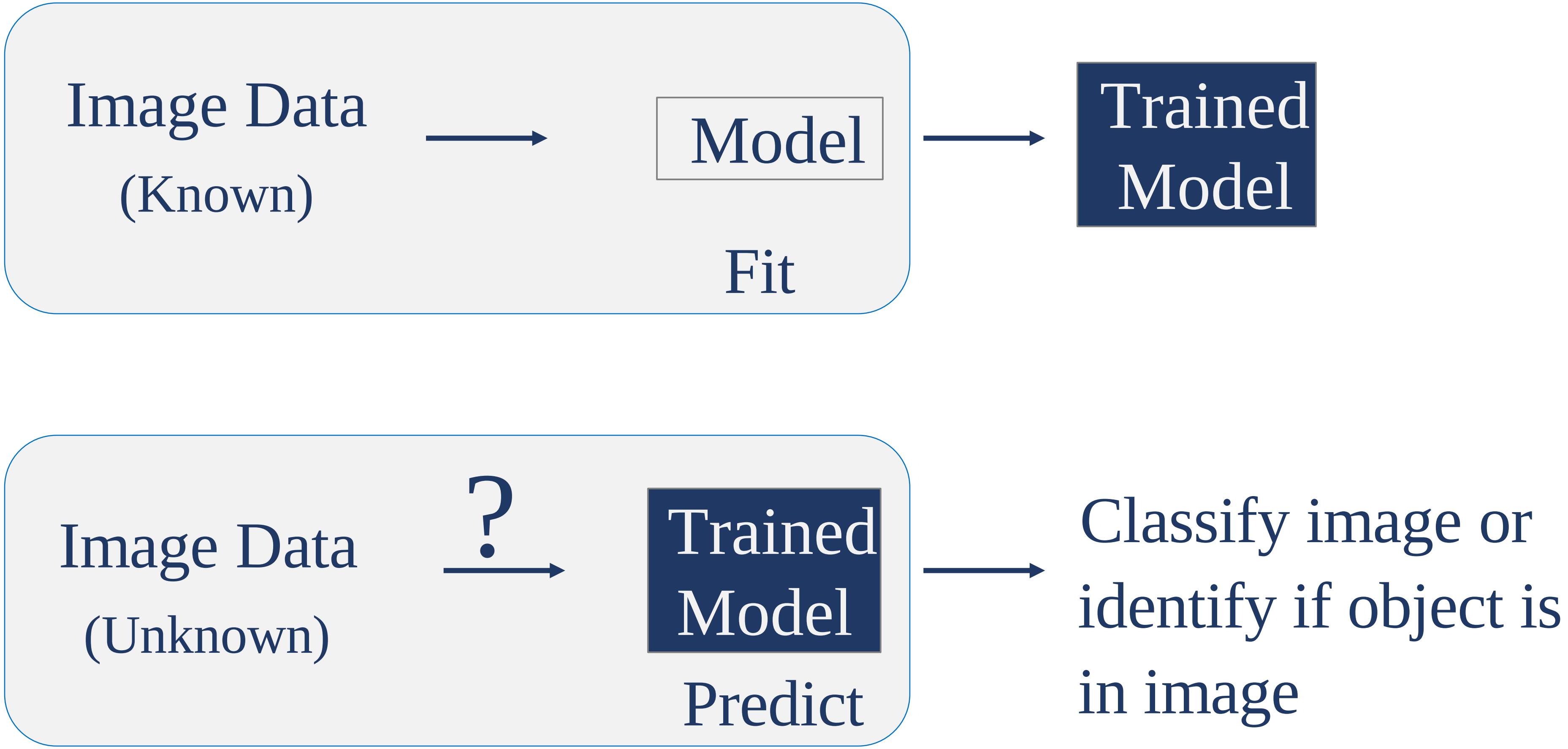
LEGAL NOTICES AND DISCLAIMERS

SIMPLE CLASSIFICATION

WHAT IS SUPERVISED LEARNING?



WHAT IS SUPERVISED LEARNING?



CLASSIFICATION: ANSWERS ARE CATEGORIES

Person

→ Model → Trained Model

Fit

? → Trained Model → Predict if there is a person in this photo (yes | no)

Predict

CLASSIFICATION: ANSWERS ARE CATEGORIES

Morning

Afternoon

Evening

Model

Fit

Trained Model

?

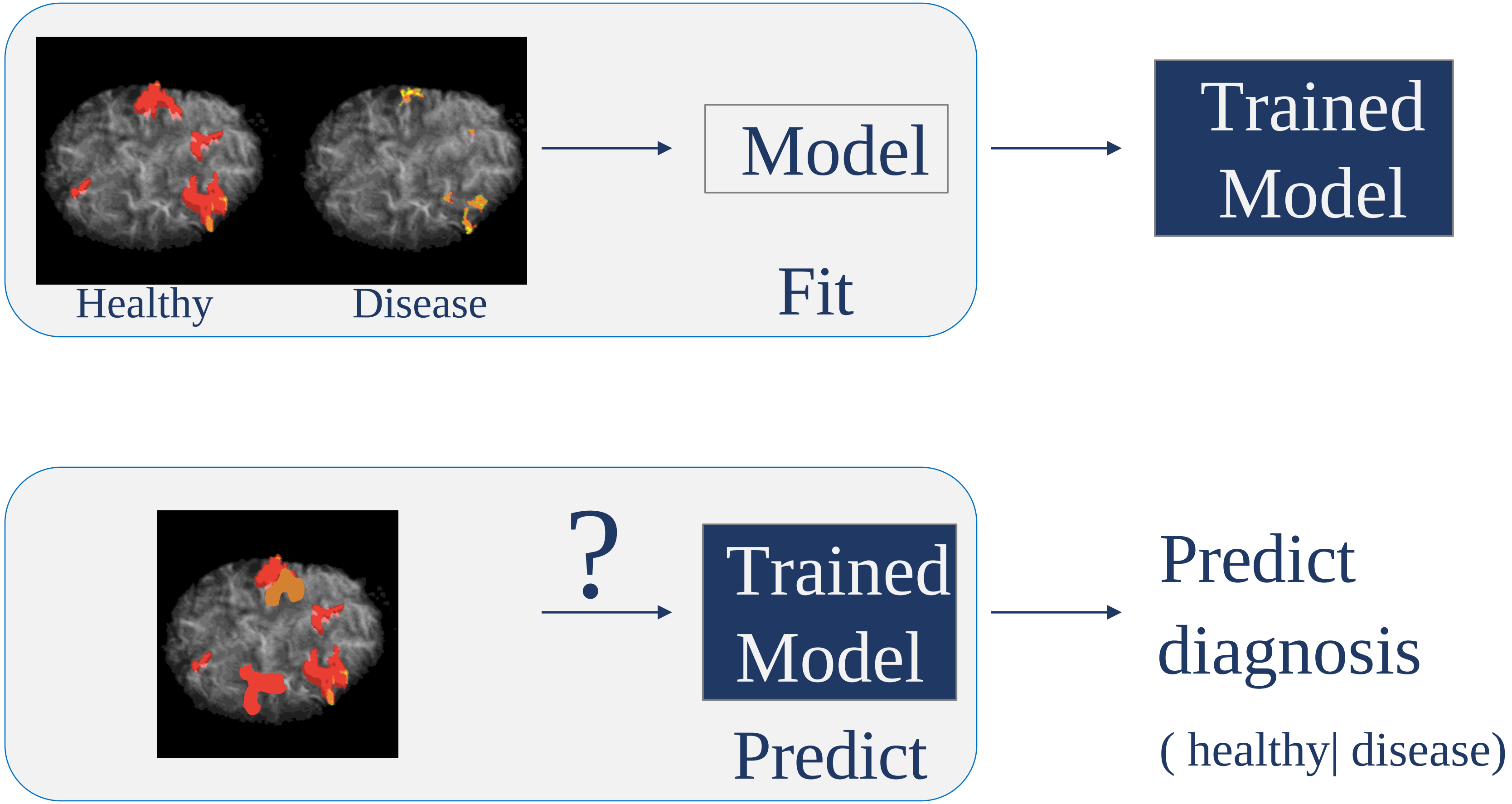
Trained Model

Predict

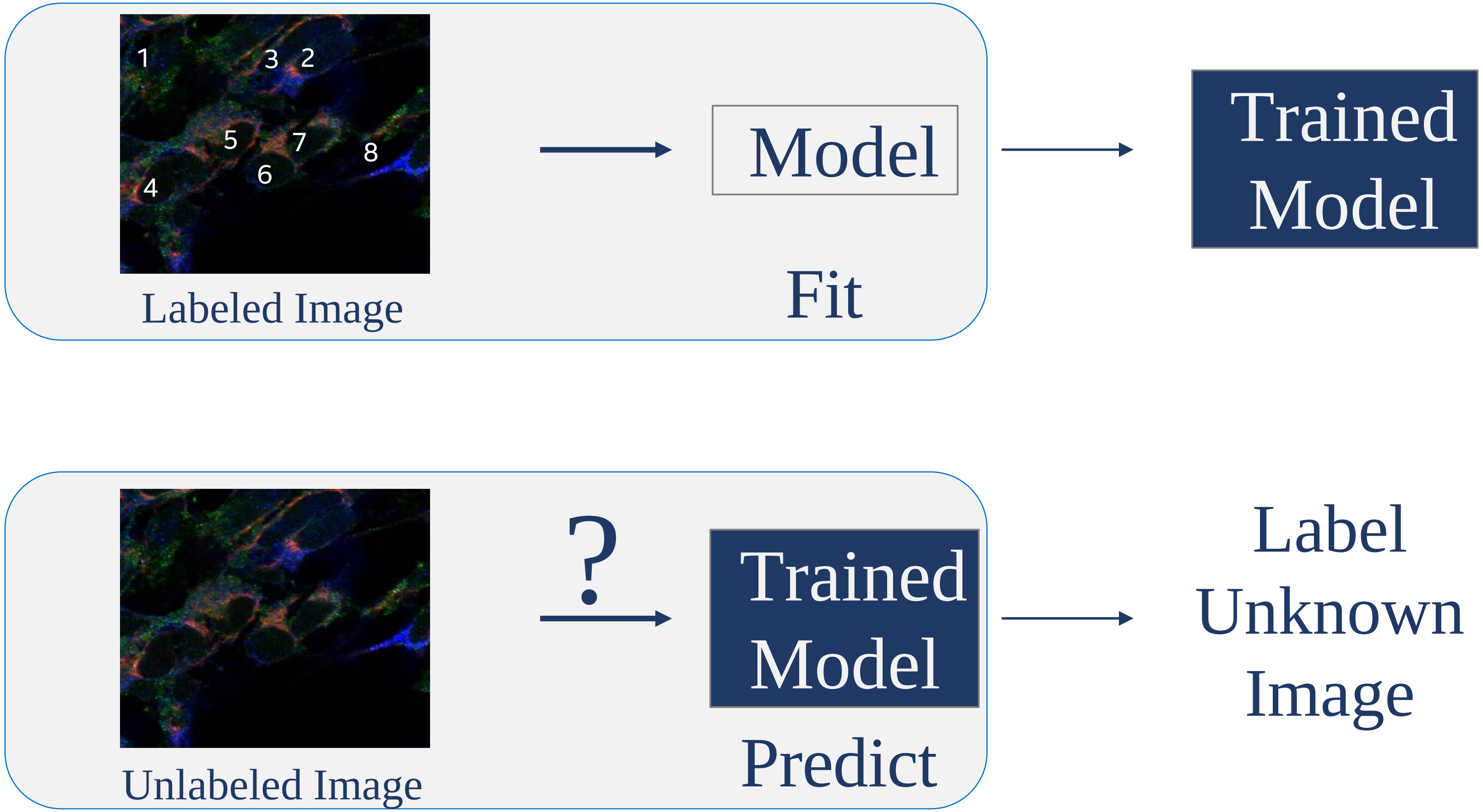
Predict what time of day it is

(Morning, Afternoon, Evening)

CLASSIFICATION: ANSWERS ARE CATEGORIES



CLASSIFICATION: ANSWERS ARE CATEGORIES



LEARNING TERMINOLOGY

Example Each data point (one row)

Feature A property of the point used for prediction (non-target columns in the model)

Target Predicted property (column to predict)

Label Target / category of the point (value of target column)

	Feature 1	Feature 2	Target
Example 1			Label 1
Example2			Label 2

LET'S REVISIT PREDICTING A PERSON

Person

Model

Fit

Model

?

Model

Predict

Predict if there is a person in this photo (yes | no)

LEARNING TERMINOLOGY

Example	1, 2, 3	**Feature**	Skin-like color
			People-like shapes
Target	Yes, Yes, No	**Label**	Yes
			No

	Skin-Like Color	People-Like Shapes	Target
Example 1			Yes
Example 2			Yes
Example 3			No

FEATURES AND LABELS

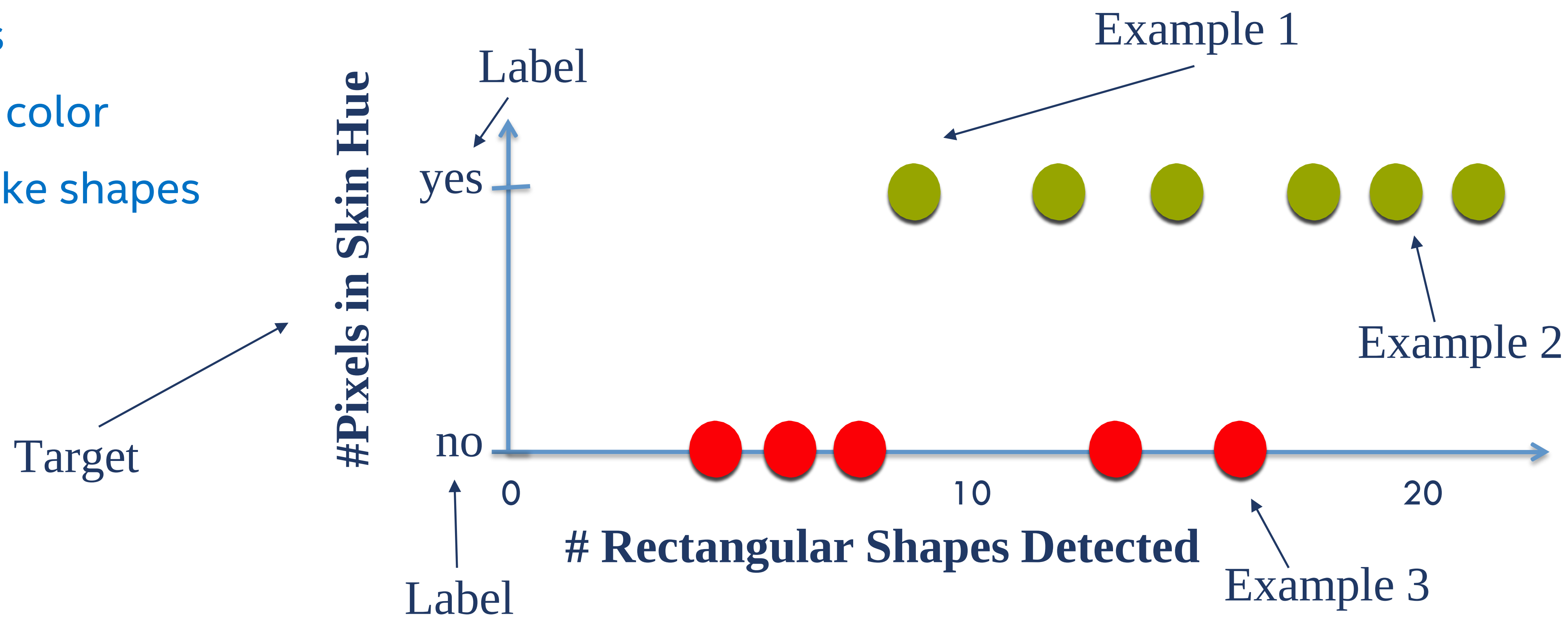
In this case we have:

Two features

- Skin-like color
- People-like shapes

Two labels

- yes
- no



FEATURES AND LABELS

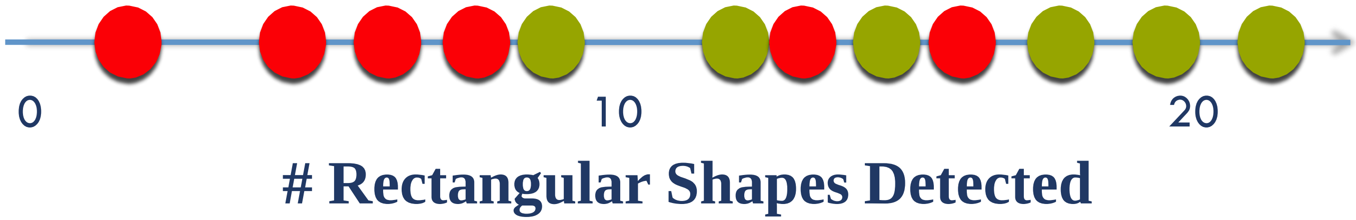
In this case we have:

Two features

- Skin-like color
- People-like shapes

Two labels

- Yes
- No



FEATURES AND LABELS

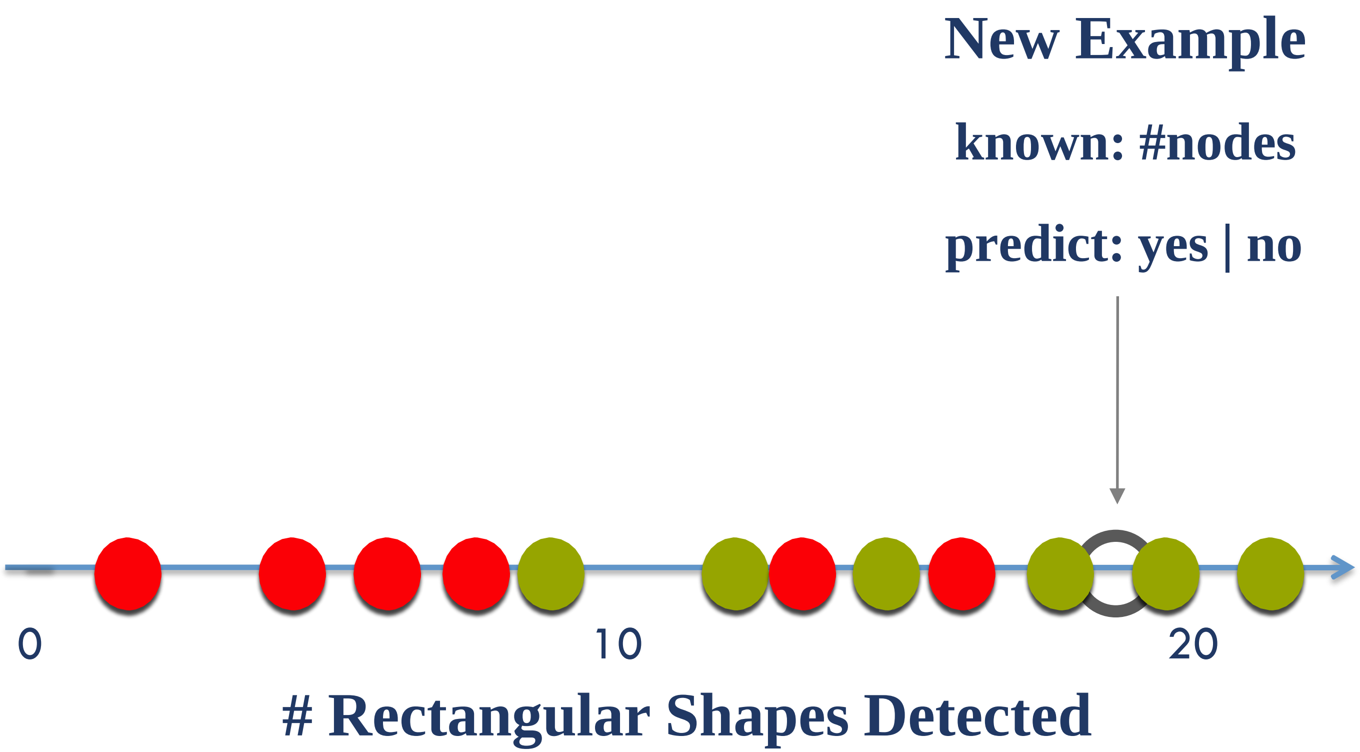
In this case we have:

Two features

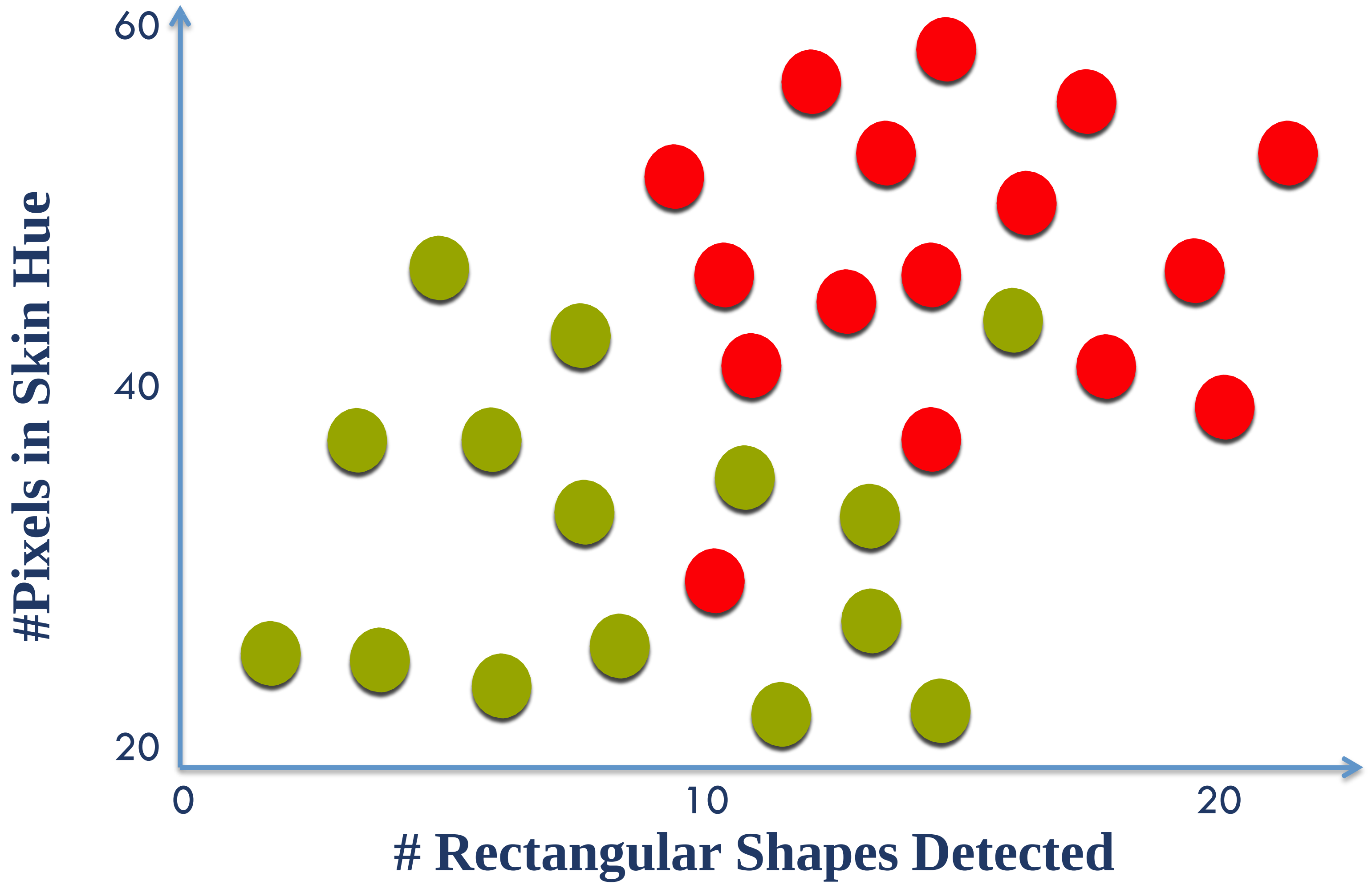
- Skin-like color
- People-like shapes

Two labels

- Yes
- No



FEATURES AND LABELS



FEATURES AND LABELS

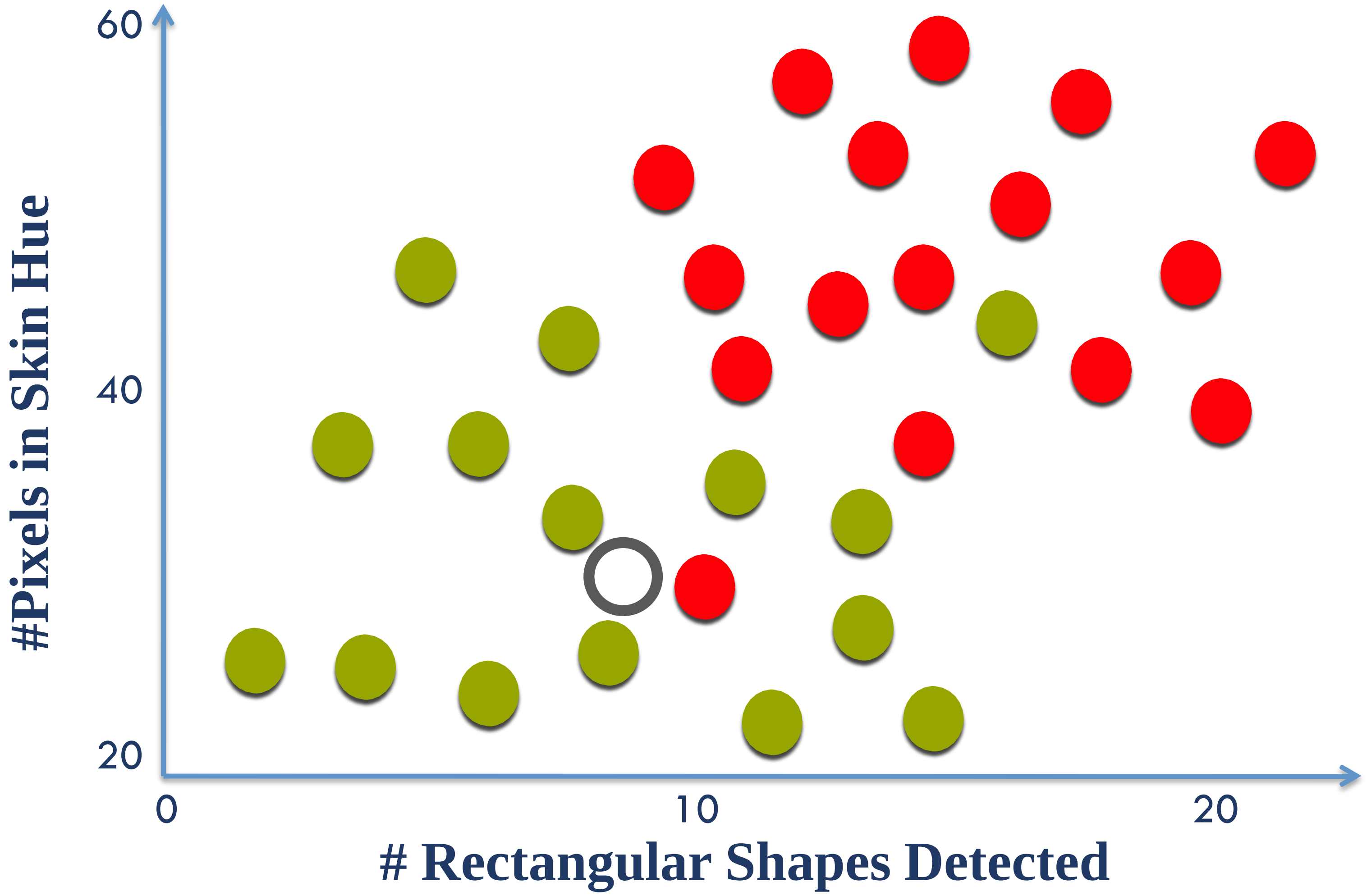
In this case we have:

Two features

- Skin-like color
- People-like shapes

Two labels

- Yes
- No



FEATURES AND LABELS

In this case we have:

Two features

- Skin-like color
- People-like shapes

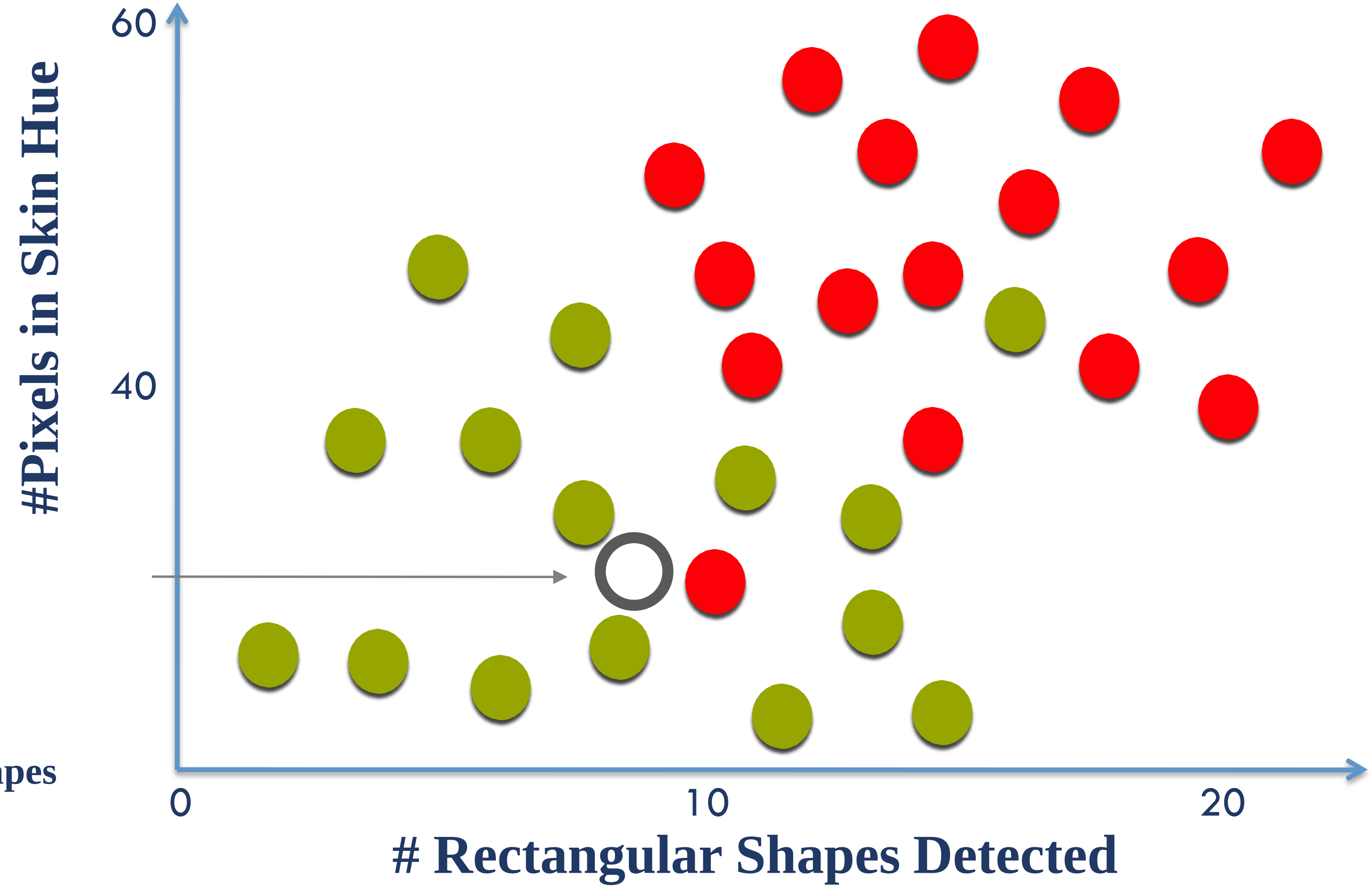
Two labels

- Yes
- No

New Example

known: #skin color, shapes

predict: yes | no



LET'S PREDICT CITY AND TIME OF DAY

Morning

Afternoon

Evening

Model

Fit

Model

?

Model

Predict

Is this city Pittsburgh?

What time of day is it?

(Morning, Afternoon, Evening)

FEATURES AND LABELS

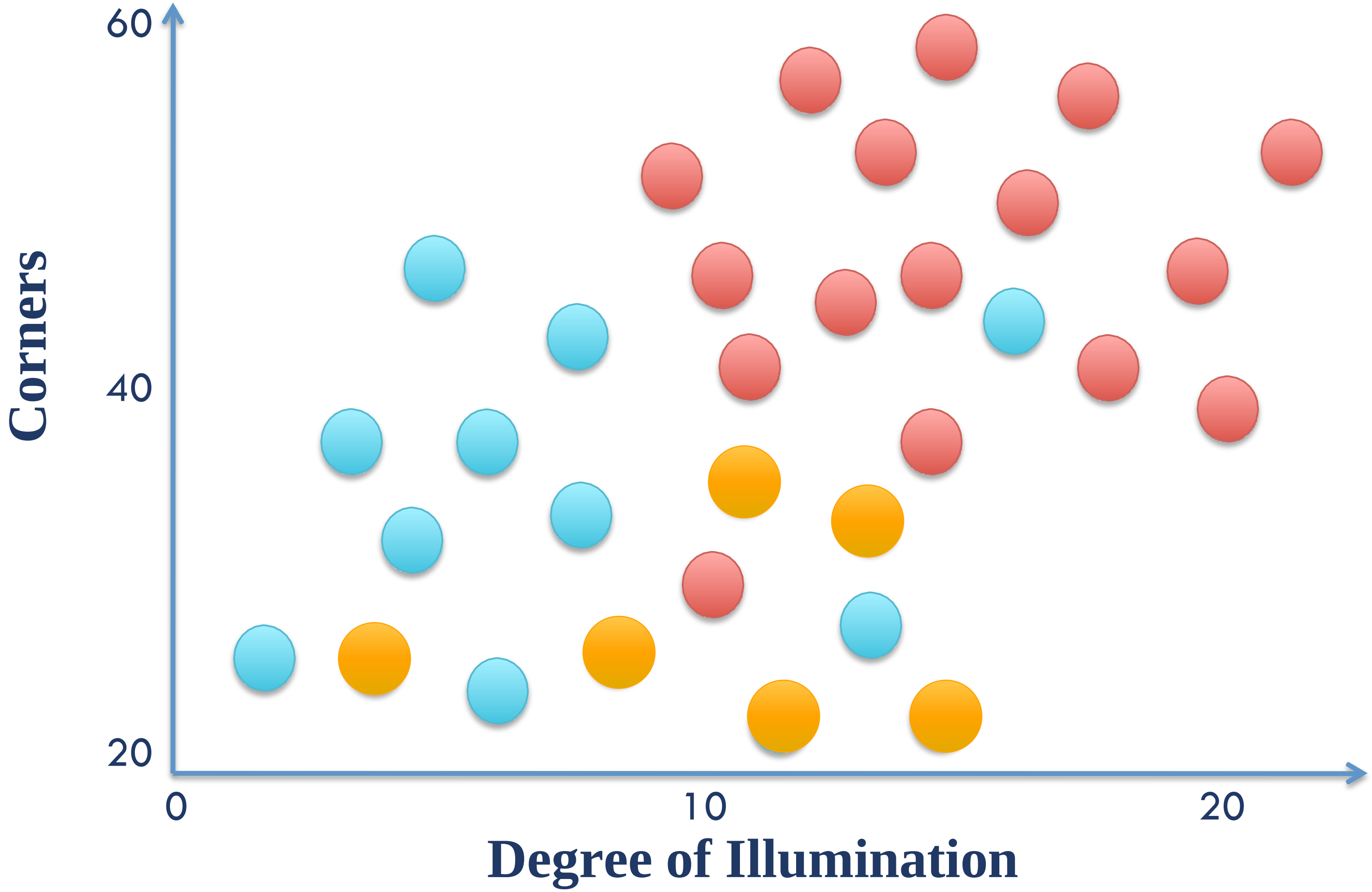
In this case we have:

Two features

- Illumination
- # corners detected against background

Three labels

- Morning
- Afternoon
- Evening



FEATURES AND LABELS

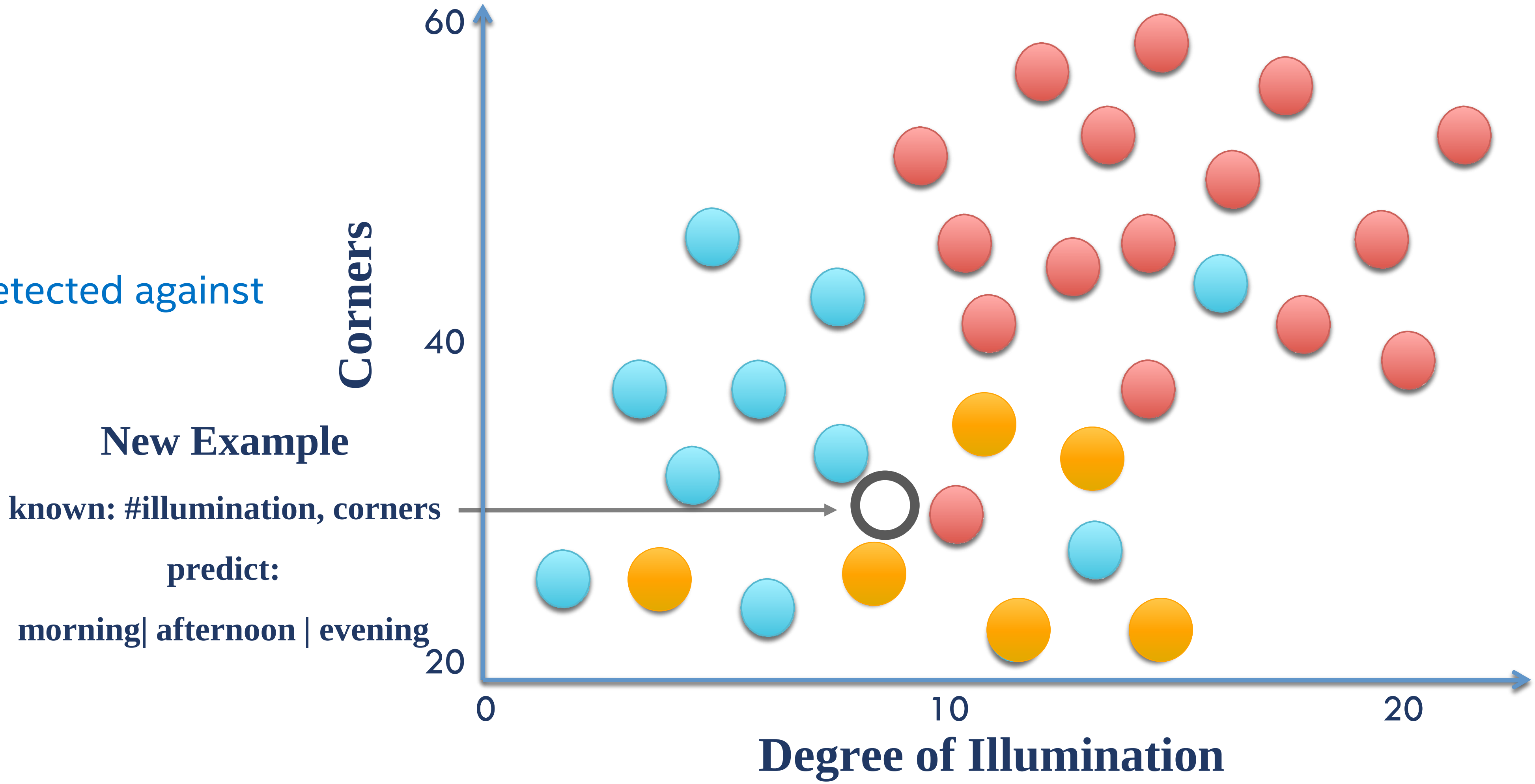
In this case we have:

Two features

- Illumination
- # corners detected against background

Three labels

- Morning
- Afternoon
- Evening



EVALUATION TECHNIQUES

EVALUATION TECHNIQUES

Last week we looked at evaluation metrics; now we'll look at effectively estimating these for a learner:

- Train test
- Cross validation
- LOO (leave one out)

TRAINING AND TESTS

Image #	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Target
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

Fold 1

Fold 2

Fold 3

TRAINING AND TEST SETS

Training set (actual)

- Fit the model

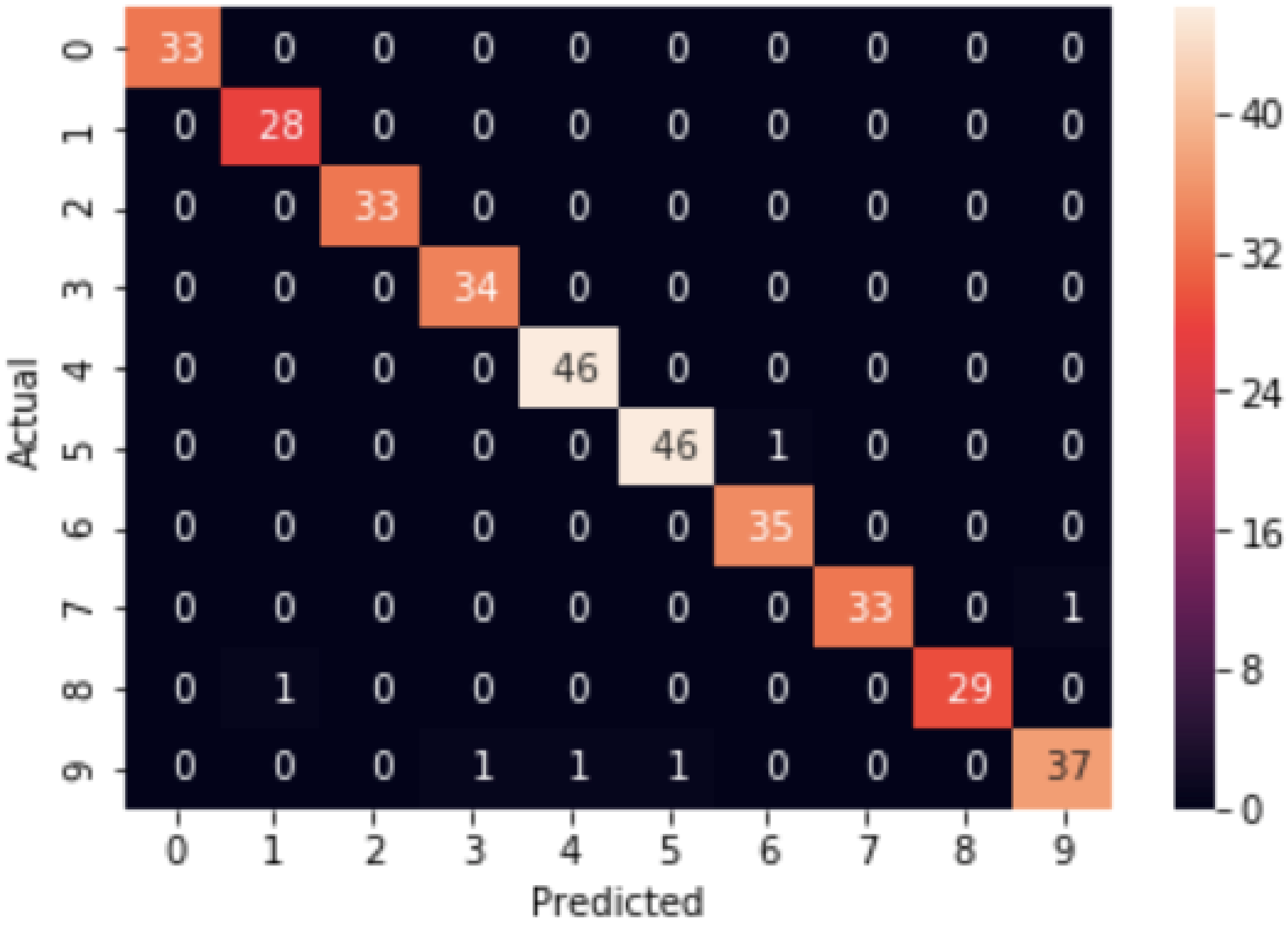
Test set (predicted)

- Measure performance

  Predict y with model

  Compare with actual y

  Measure error



TRAINING AND TEST SETS

From simple workflow in notebook:

```
import sklearn.model_selection as skms
(data_train, data_tst, tgts_train, tgts_tst) = skms.train_test_split(data, tgts, test_size=.2)


from sklearn import neighbors
# create and fit modelknn_classifier = neighbors.KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(data_train, tgts_train)


predicted = knn_classifier.predict(data_tst)


Accuracy = skms.accuracy(predicted, tgts_test)
```

CROSS-VALIDATION

Folds do not overlap

Each fold is the same size

Every example is in a fold

Image #	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Target
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

Fold 1

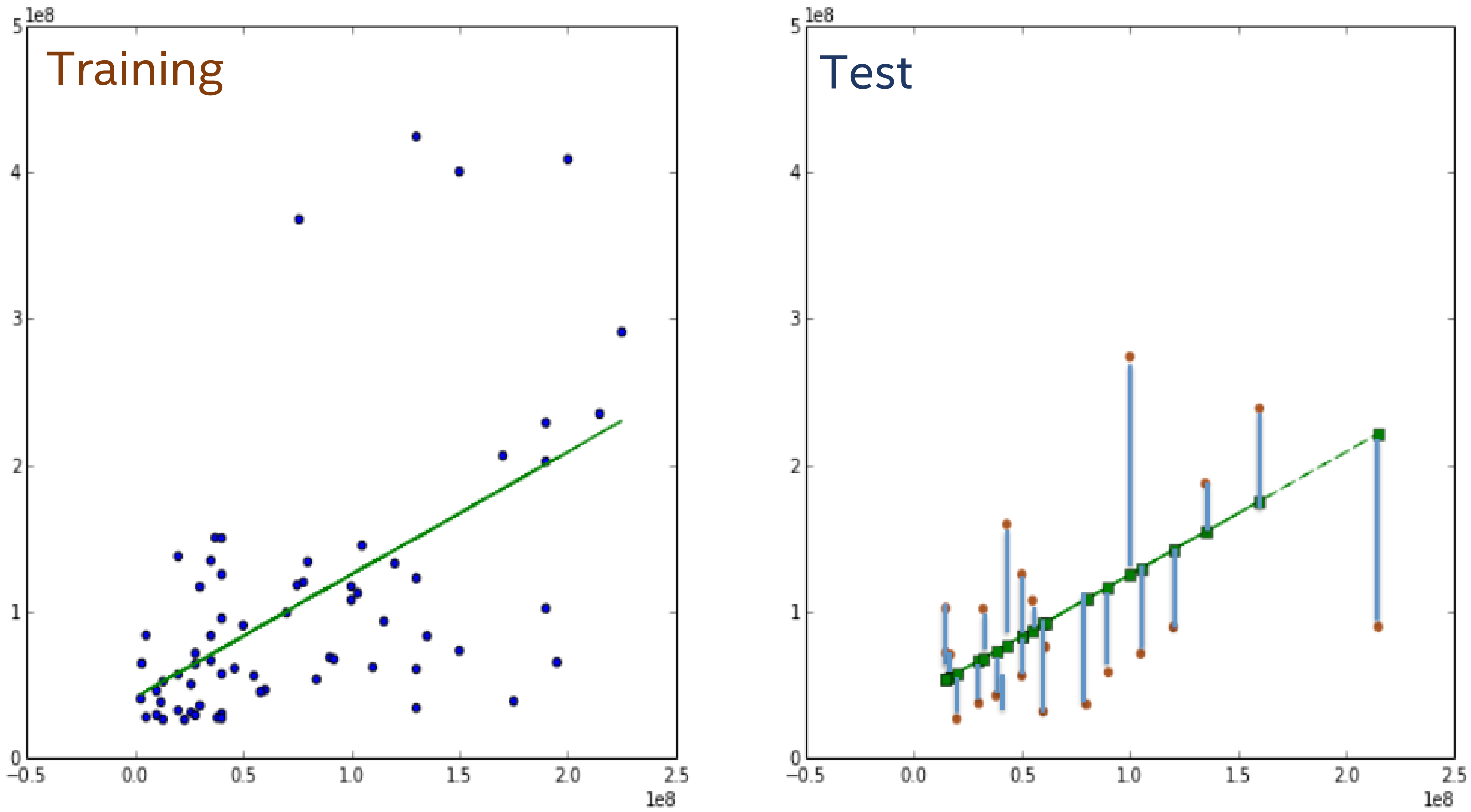
Fold 2

Fold 3

CROSS-VALIDATION

Best fit for Fold 1

What about Folds 2 and 3?



LEAVE ONE OUT CROSS-VALIDATION

Removes one datapoint from the dataset at a time.

Train from the N-1 datapoints

- N = number of datapoints
- In this example, N = 24

This is repeated for each datapoint in the set.

You can also think of the as N-fold cross validation

- That is, 24 datapoints is 24-folds

Image #	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Target
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

LEARNING METHODS

LEARNING METHODS

Supervised learning techniques

- K nearest neighbor (KNN)
- Support vector machines (SVM)

Unsupervised learning techniques

- Principal components analysis (PCA)
- Clustering

K NEAREST NEIGHBOR (KNN)

KNN

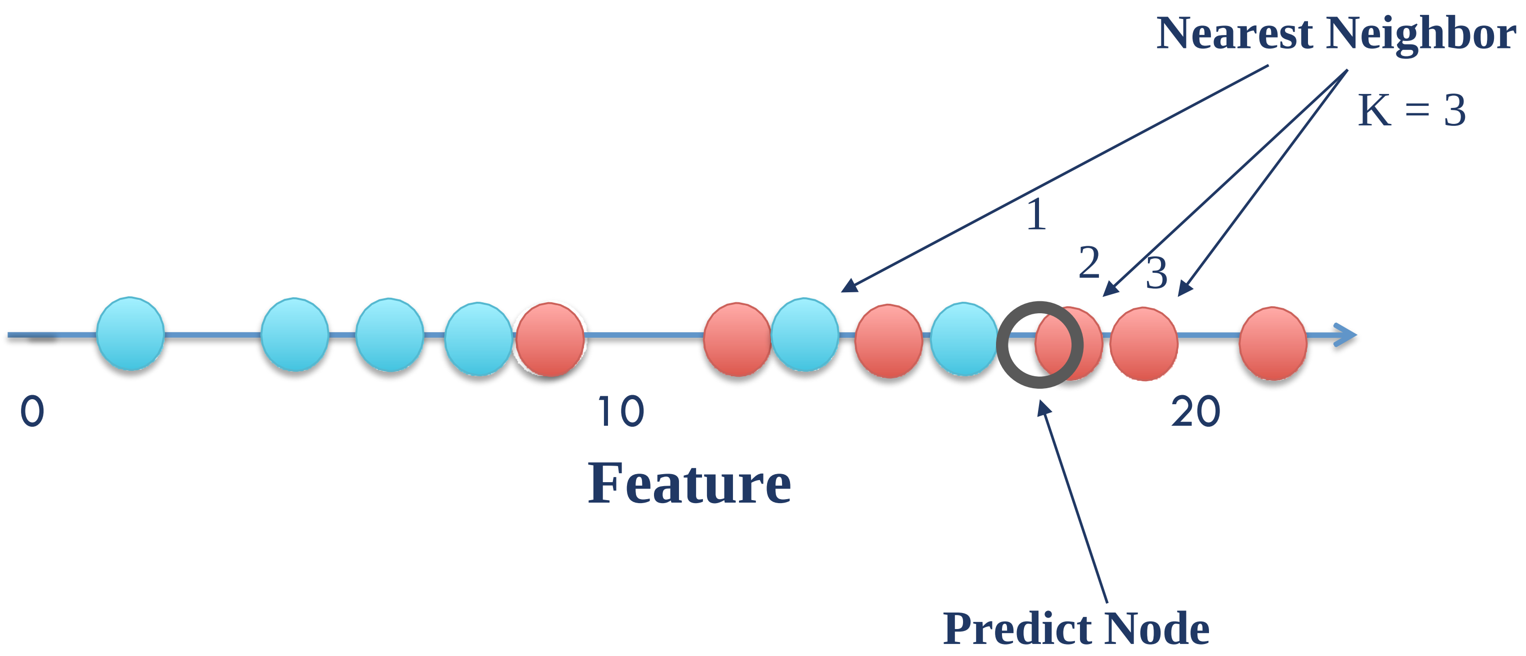
- Memorize the dataset

  For prediction, find example most like me

- Without preprocessing, fit is *instant* – but prediction takes more work.
- With preprocessing, spend some up-front cost to organize the data, then prediction is (relatively) faster.
- Requires significant memory because it saves some form of the entire dataset.

K NEAREST NEIGHBOR (KNN)

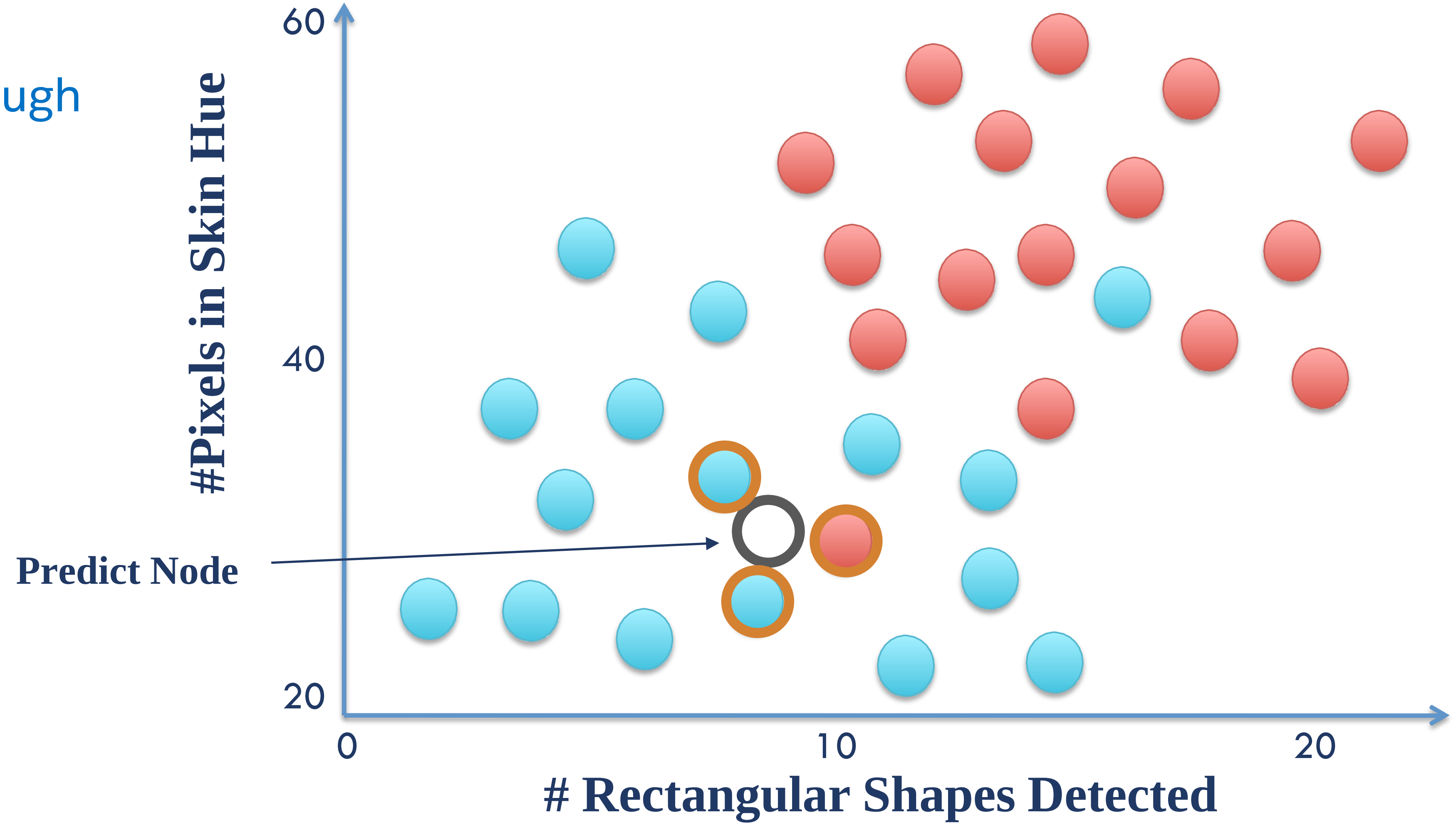
K= # nearest neighbors used for predictor



KNN

K = 3

Predict the node through its nearest neighbors



KNN

Use KNN to make a decision boundary

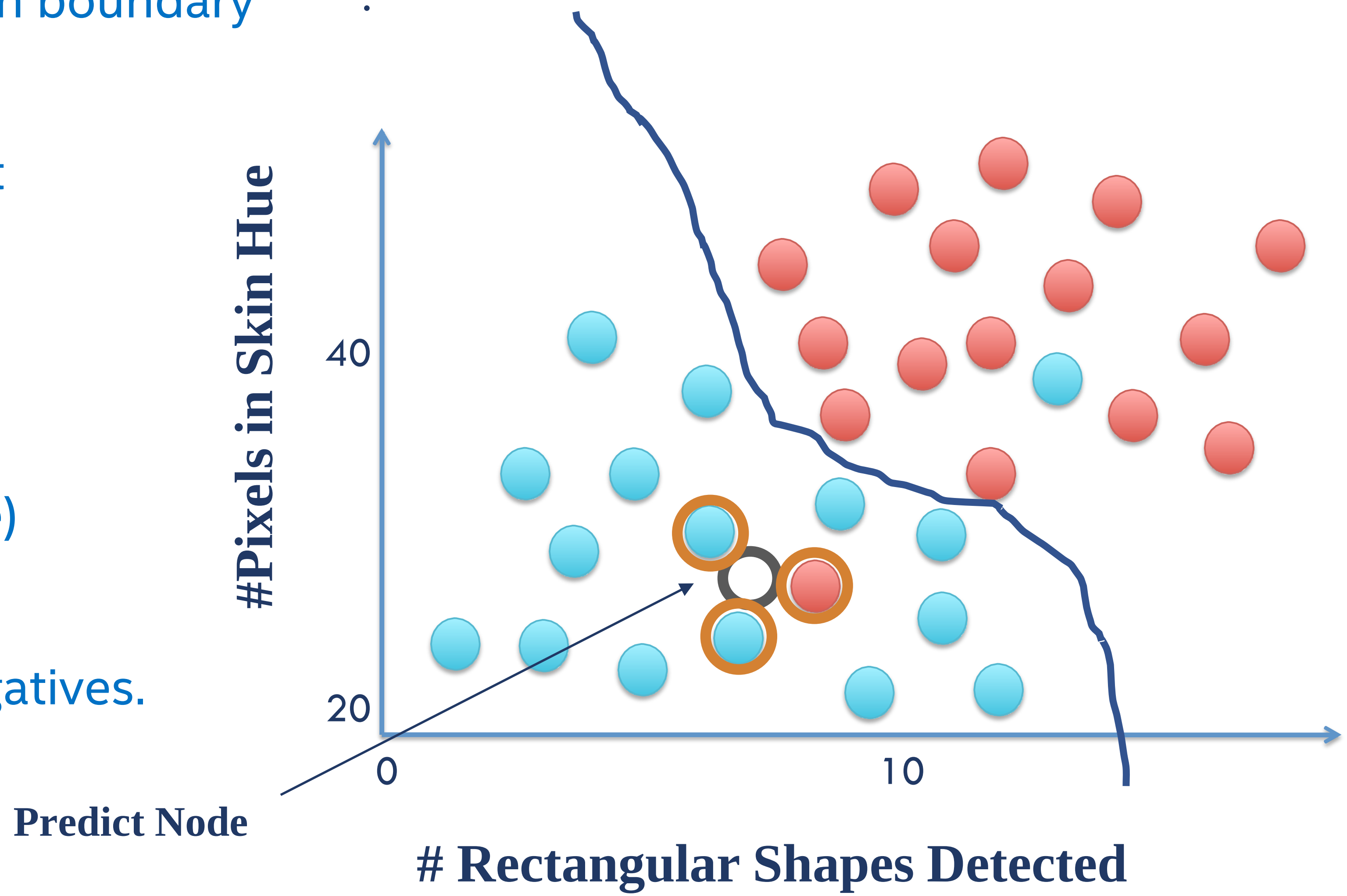
If the node falls to the right

- Decision = male (red)

If node falls to the left

- Decision = female (blue)

Note false positive and negatives.



KNN

Multiclass decision boundaries break the data into three or more subsets.

Scaling is critical for determining the best decision boundary.

- If data is scaled too closely together in one dimension, examples may be too similar distance-wise compared to actual class similarity.

The number of nearest neighbors used for analysis is also critical.

- Too small a K will give too wiggly a border.
  - Example of *overfitting* or following noise

- Too big K will give too flat a border.
  - Example of *underfitting* ignoring signal

SUPPORT VECTOR MACHINES

Used for discriminative classification

Divides classes of data into groups by *drawing a border between them.*

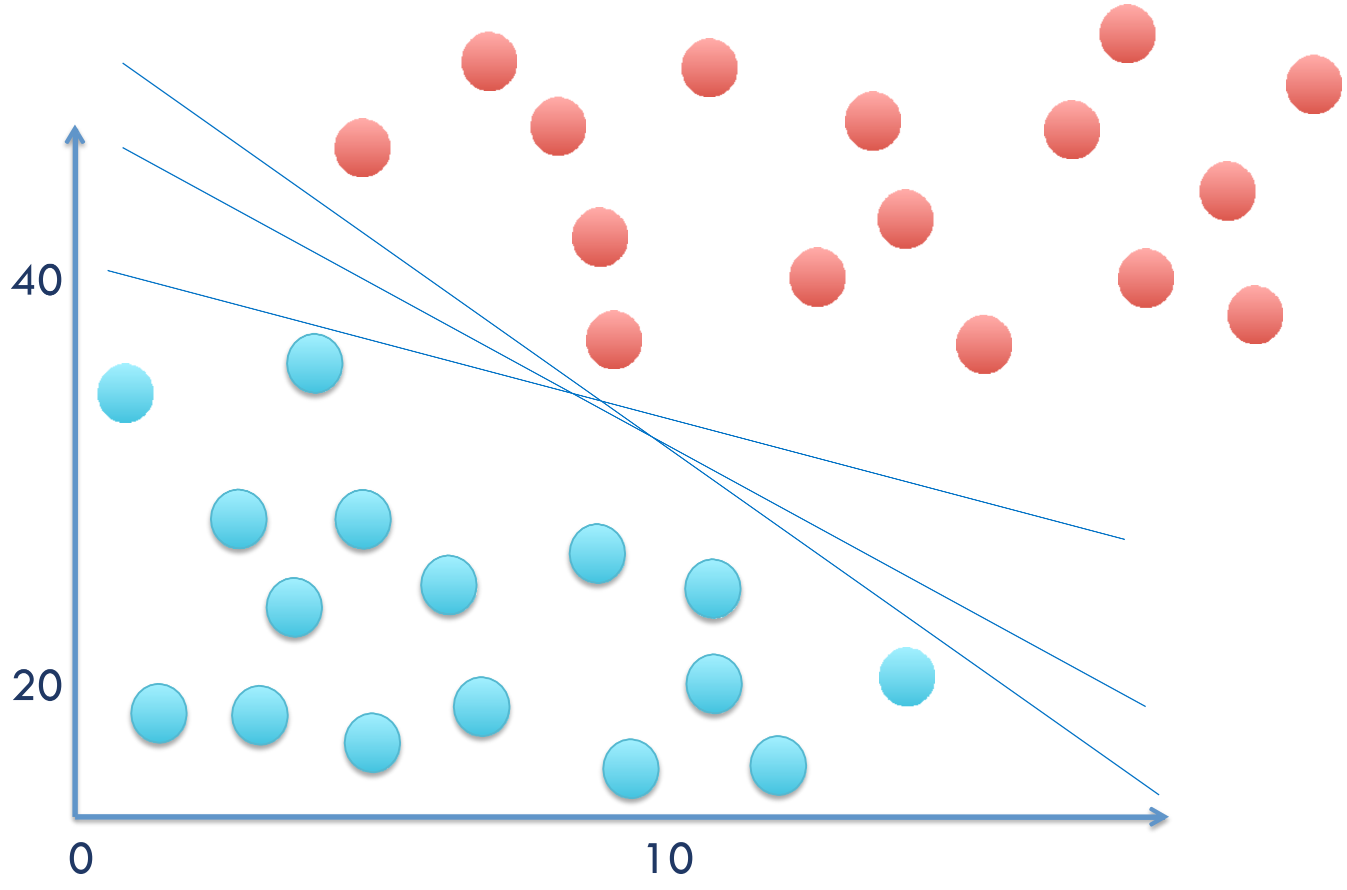
- Unlike KNN, it goes straight to the decision boundary
- Remembers boundary, throws out data

Is a maximum margin estimator.

- The border between classes that maximizes the distance to examples from that class

SVM: LINEAR BOUNDARIES

There are many (infinite) possibilities.



SVM: LINEAR BOUNDARIES

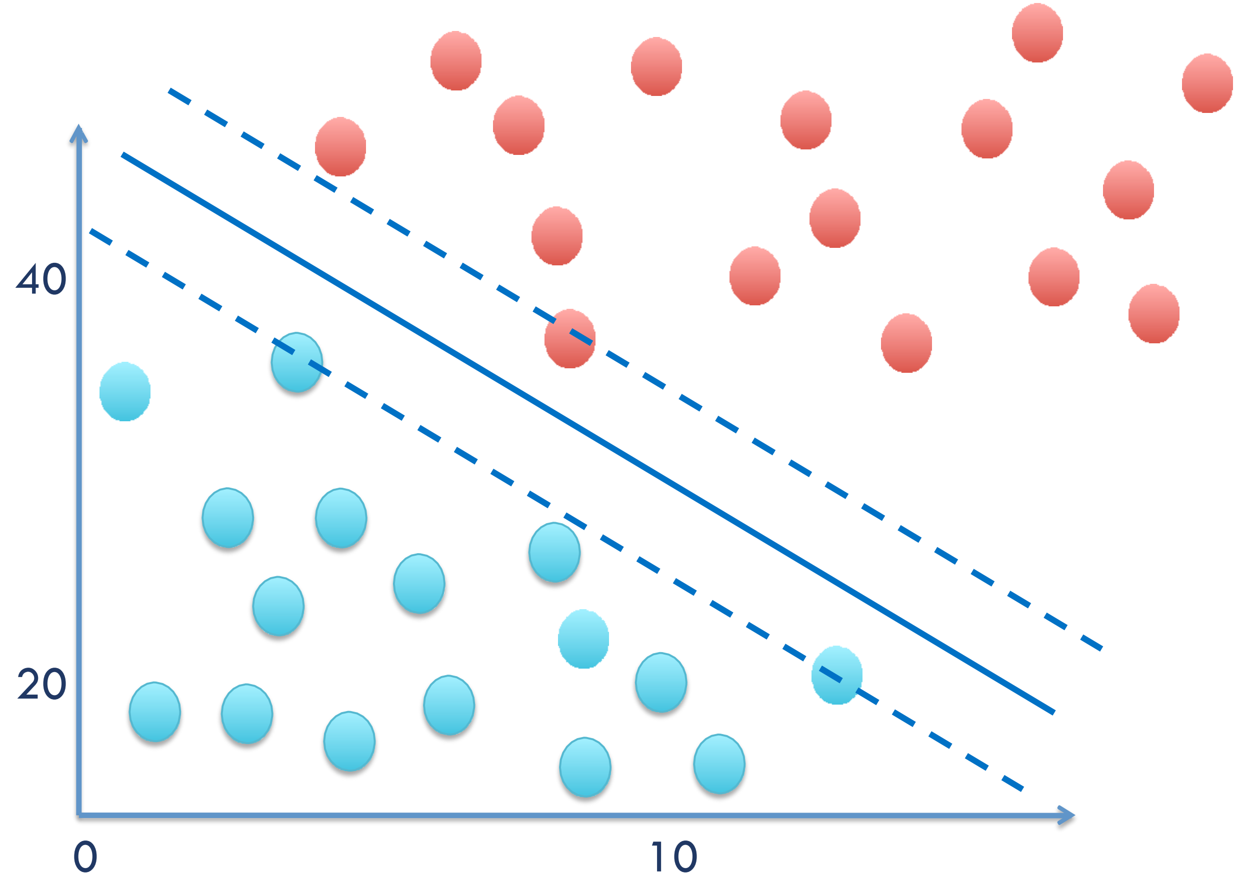
So, let's use a margin around the linear boundary.

- *Maximum-margin* is SVM's preferred line

The examples on the margin lines are support vectors.

Now, the only important examples to keep track of are the support vectors.

For new data, see which side of the line it falls on.



SVM: KERNEL

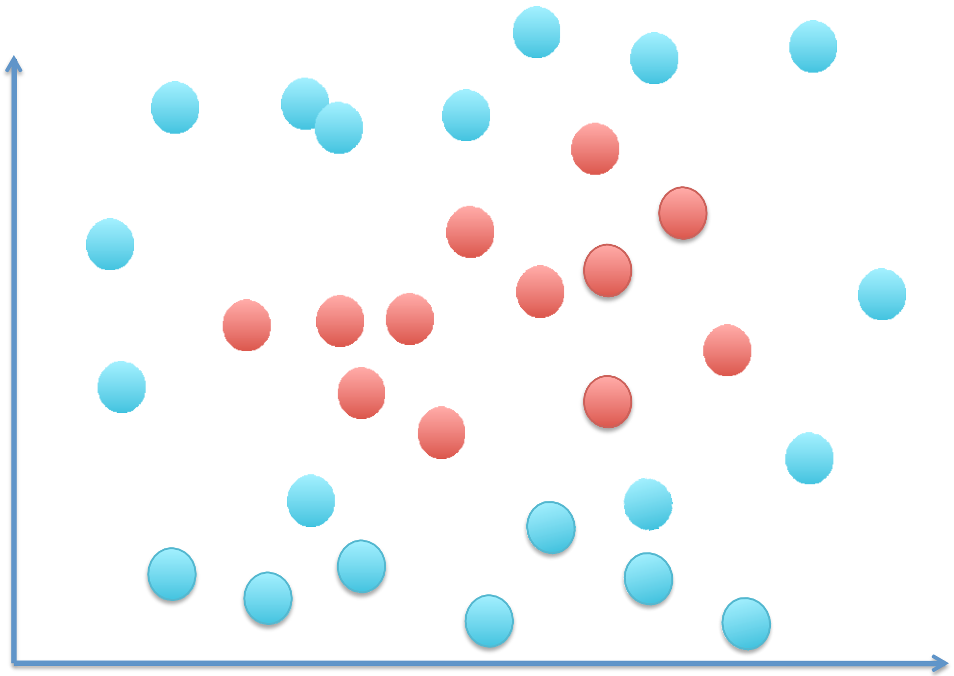
How would you separate this data?

Not with a line.

Need to get fancy.

Use linear algebra tricks to rewrite this data in more complicated terms:

- Polynomials of the inputs
- Distances from one input to another
- These are called *kernels*



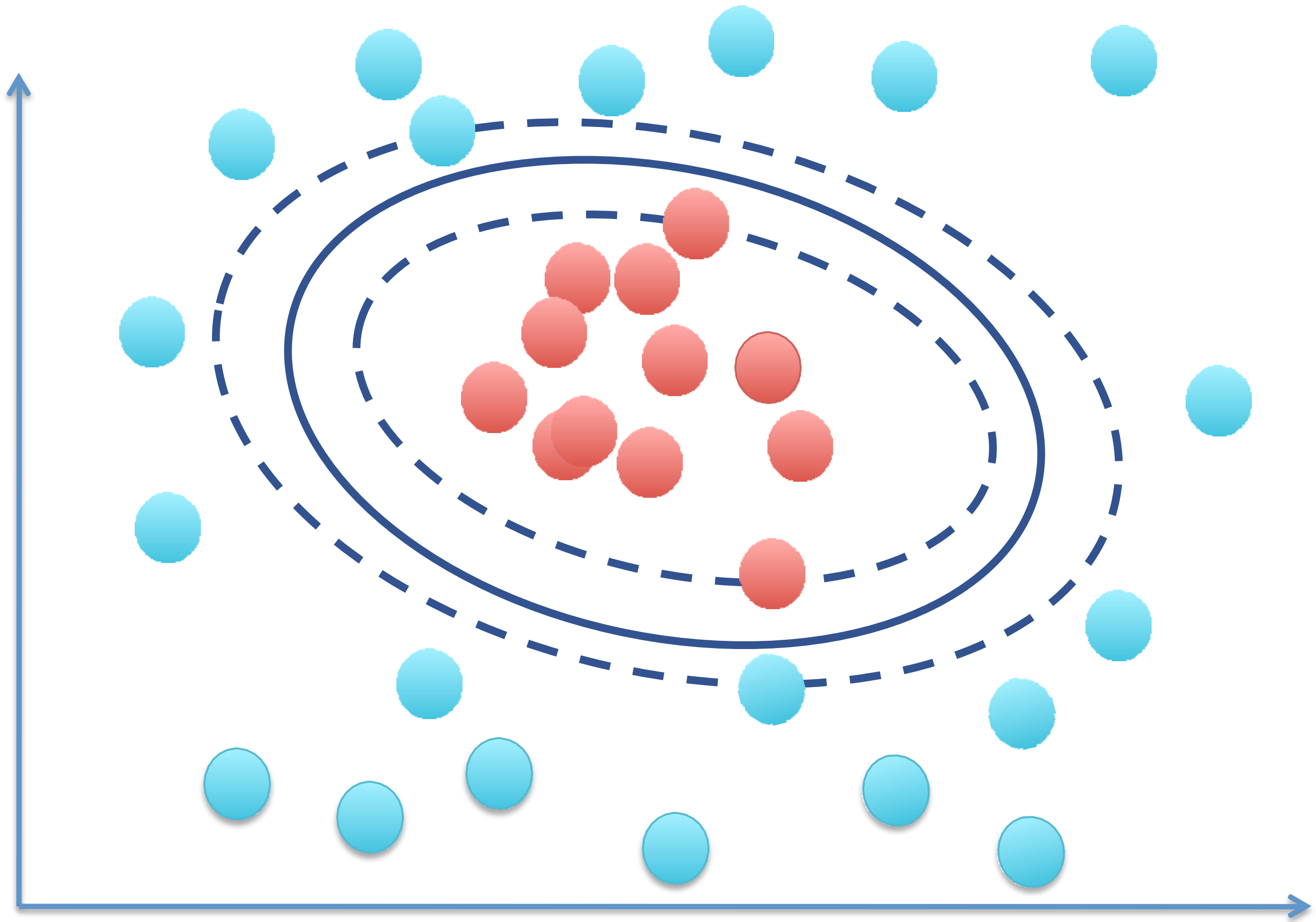
NON-LINEAR DATA

You must pick the *right* projection to separate the classes.

Kernel transformation computes the basis function for every example.

- Can be too cumbersome when you get large datasets
- Instead, we can use the *kernel trick*
  - Implicitly fit kernel-transformed examples
  - Does not build explicit (expensive) table

NON-LINEAR DATA



UNSUPERVISED METHODS

PRINCIPAL COMPONENTS ANALYSIS (PCA)

What are we trying to achieve?

- Improve clustering
- Improve classification
- Dealing with sparse features
- Visualizing high-dimensional 2D or 3D data
- Minimal loss data compression

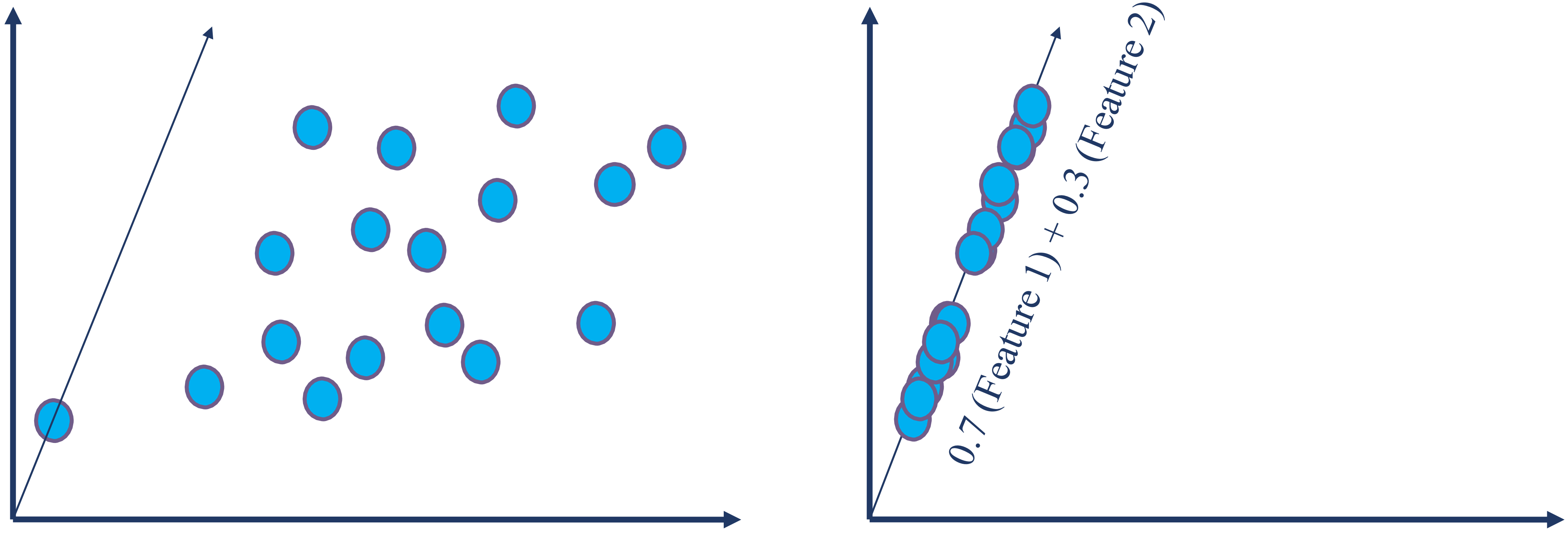
PCA: 2D TO 1D

You can weight your existing dimensions (2D) to give you 1D data.

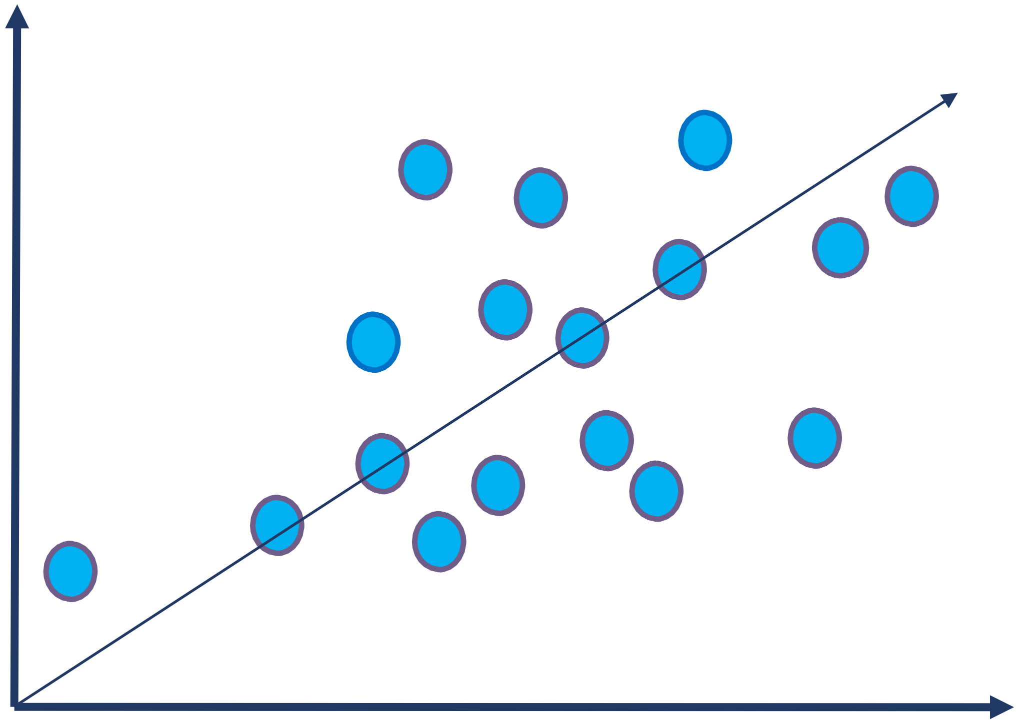
- Data as a point-cloud
- Fit ellipse
- Find perpendicular axes
  - Directions of maximum variation
- Describe data in terms of these directions
  - Orient the data in a new way

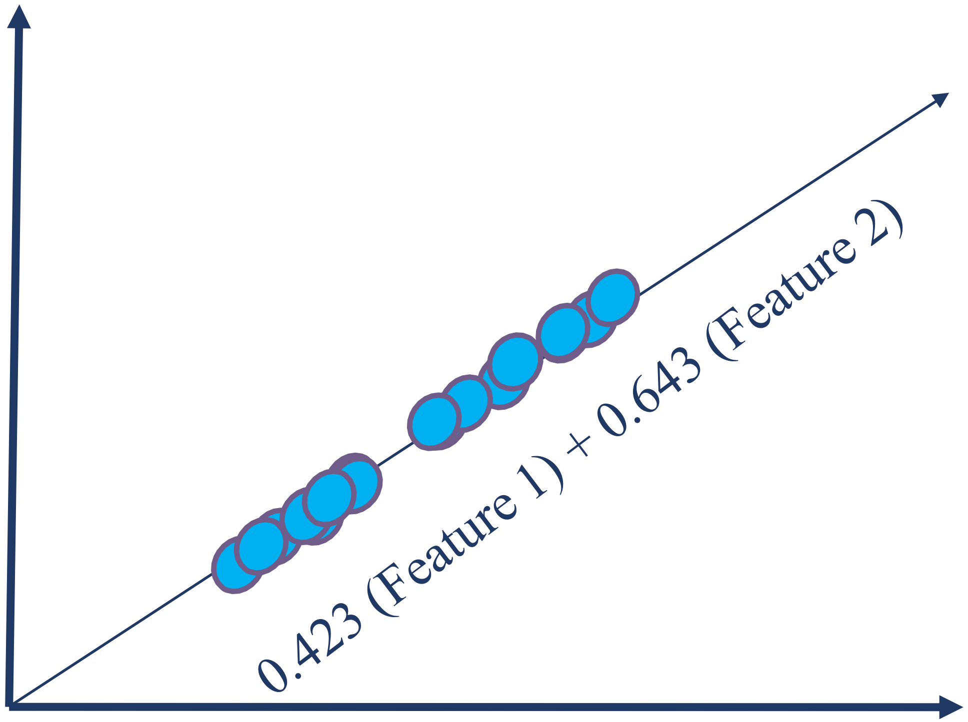
PCA: 2D TO 1D

You can weight your existing dimensions (2D) to give you 1D data.



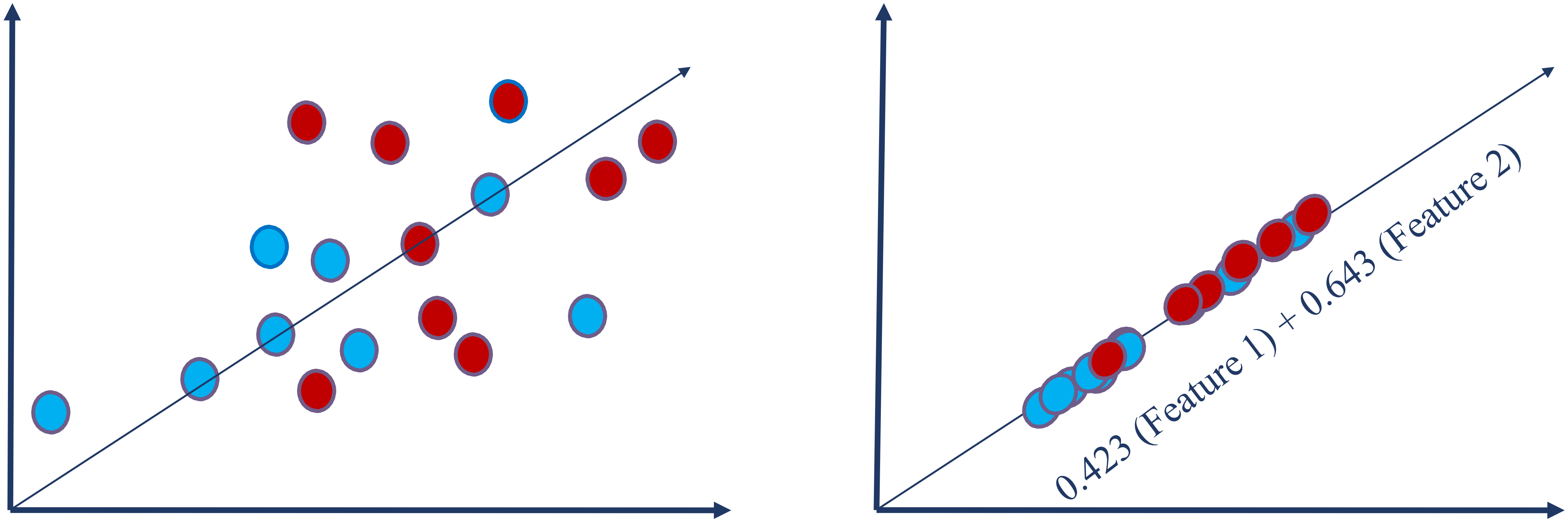
PCA: 2D TO 1D



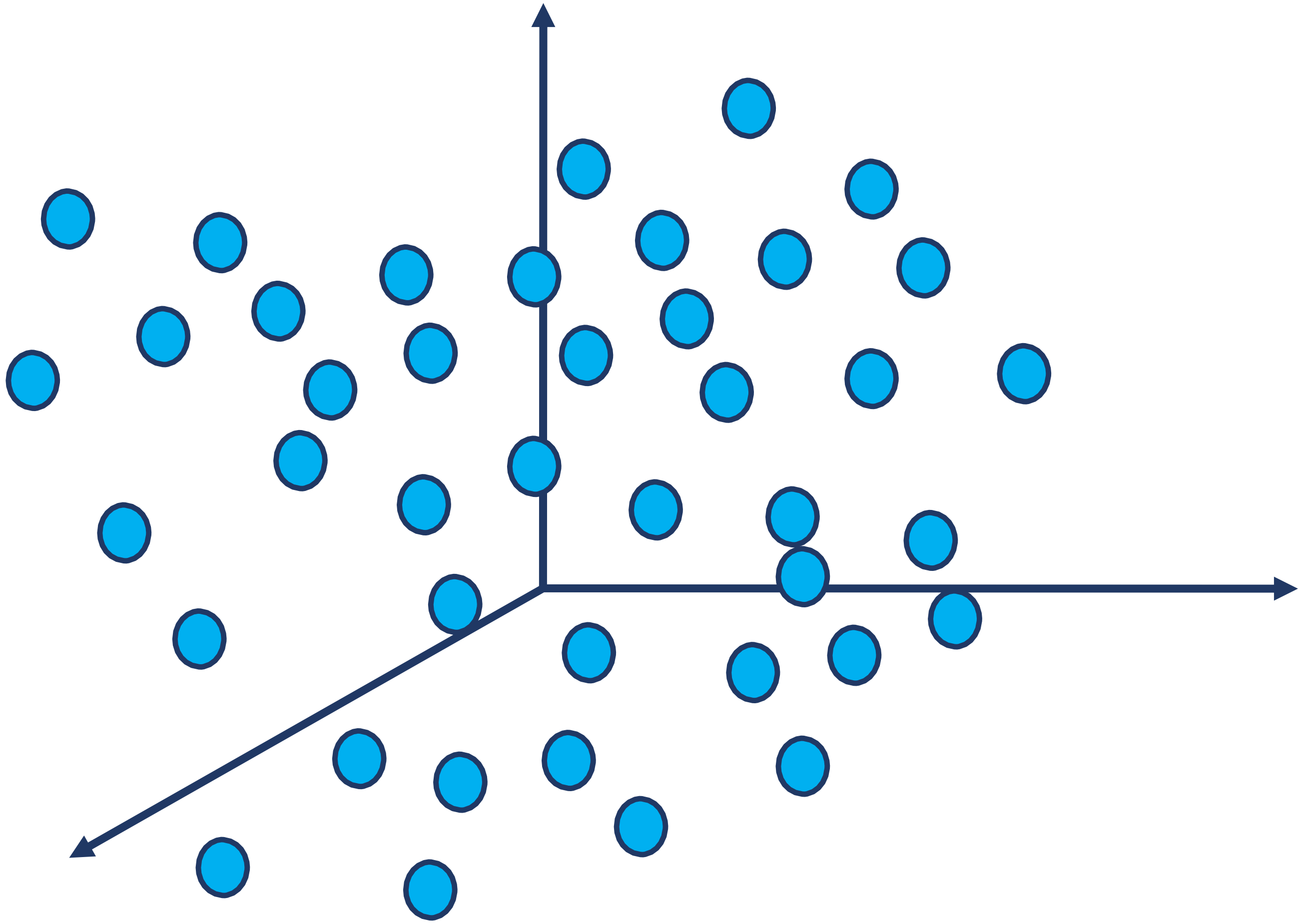


PCA: 2D TO 1D

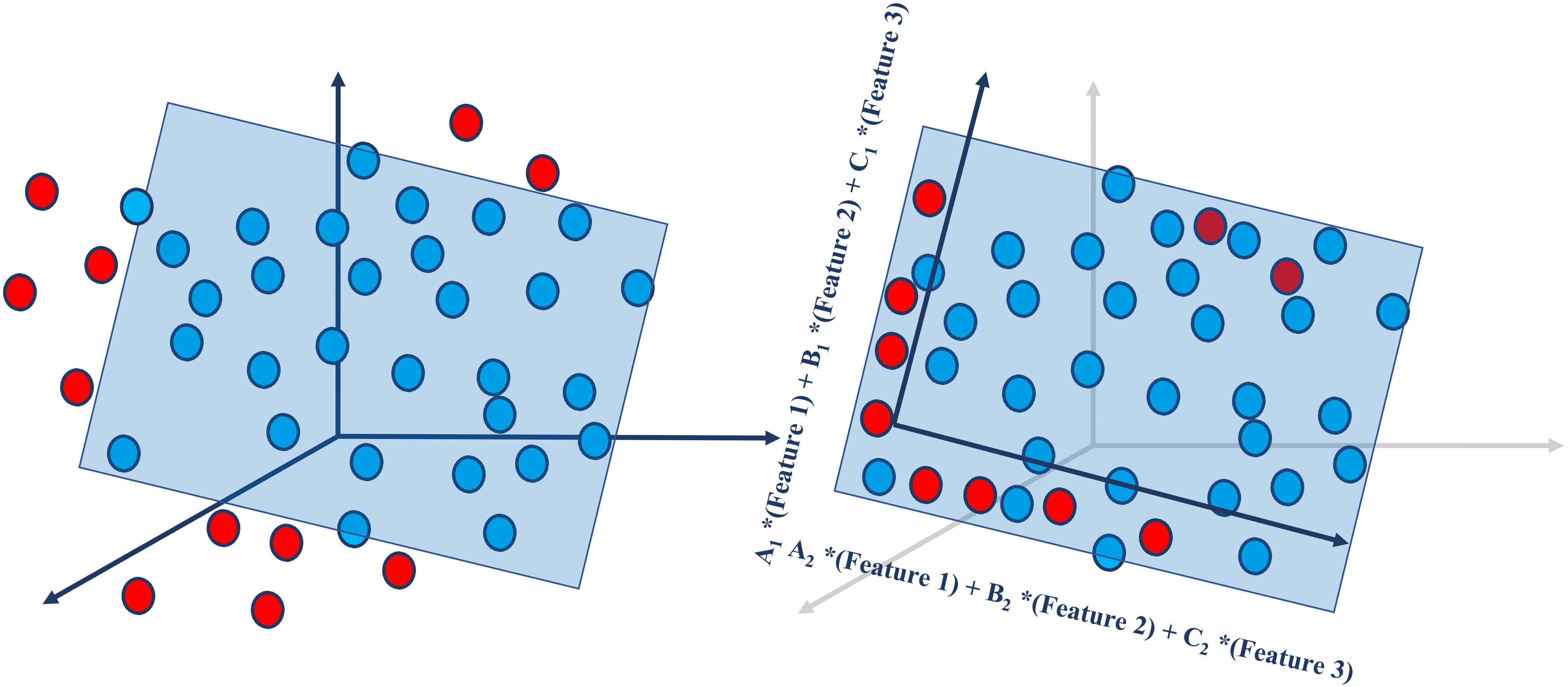
This may clarify or simplify the boundary between classes….



PCA: 3D TO 2D



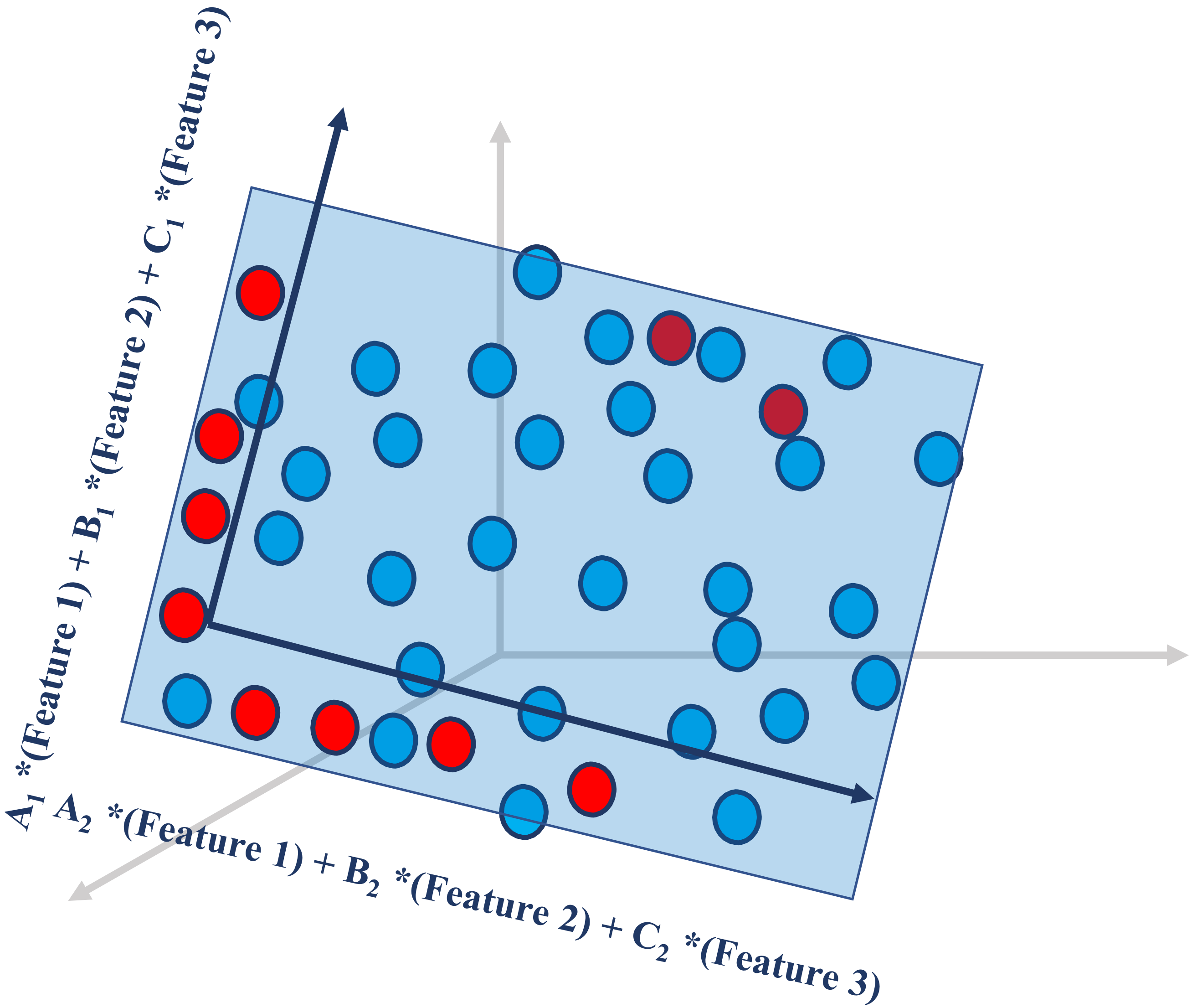
PCA: 3D TO 2D


A1 *(Feature 1) + B1 *(Feature 2) + C1 *(Feature 3)
A2 *(Feature 1) + B2 *(Feature 2) + C2 *(Feature 3)

PCA: 3D TO 2D

Math speak:

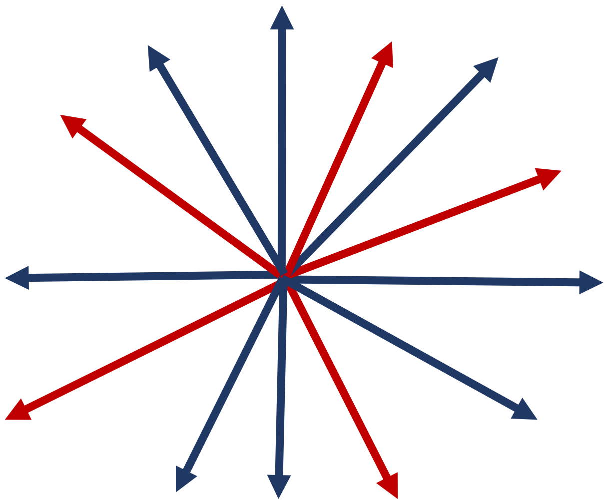
- The new axes are eigenvectors
- The new axes are calculated from covariance of the matrix features via a *singular value decomposition*



EIGENFACES: A PCA EXAMPLE

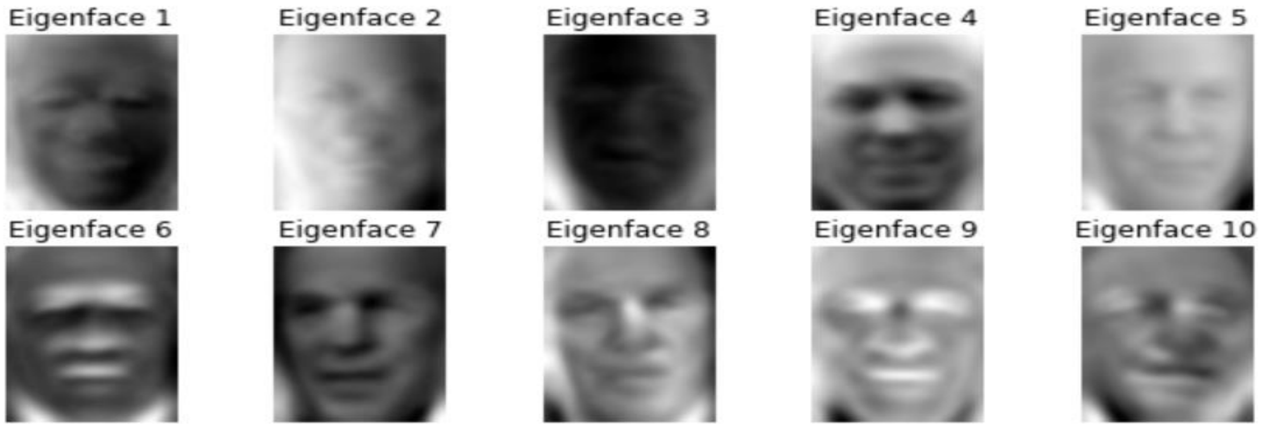
Original data

$f_{original}$ dimensional plot



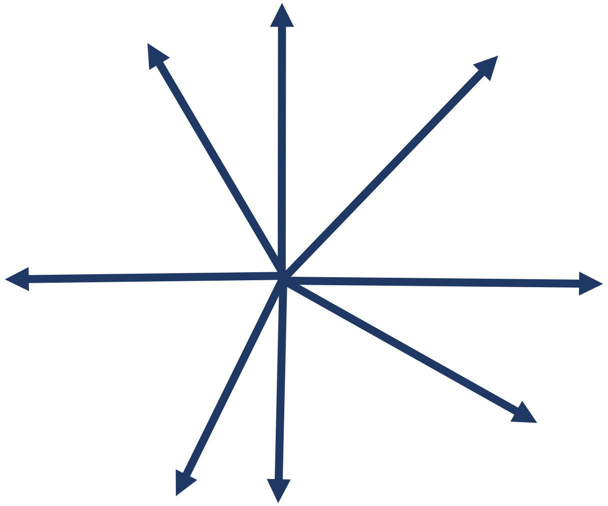
EIGENFACES: A PCA EXAMPLE

PCA data



New $f_{reduced}$ dimensional plot

- Removed some number of features from our representation
- Redescribe original faces in terms of these *primitive* faces that act like axes
- Similar to redescribing images in terms of circles (which we saw with Fourier transforms)



PCA

When choosing dimensions for your feature extraction:

- Think about combining features to reduce dimensionality.
  - That is, instead of analyzing pixel intensity at each color (RGB), can you analyze pixel intensity (grayscale)?
- You will need to perform hypothesis-driven trials and measure your model's performance.

PERFORMING PCA USING SCIKIT-LEARN*

```
from sklearn.decomposition import PCA

reducer = PCA( n_components = 20 )

reduced_X = reducer.fit_transform(X)

# “model” could be any trained model

model.fit(reduced_X, Y)
```

PERFORMING PCA USING SCIKIT-LEARN*

When you need to predict:

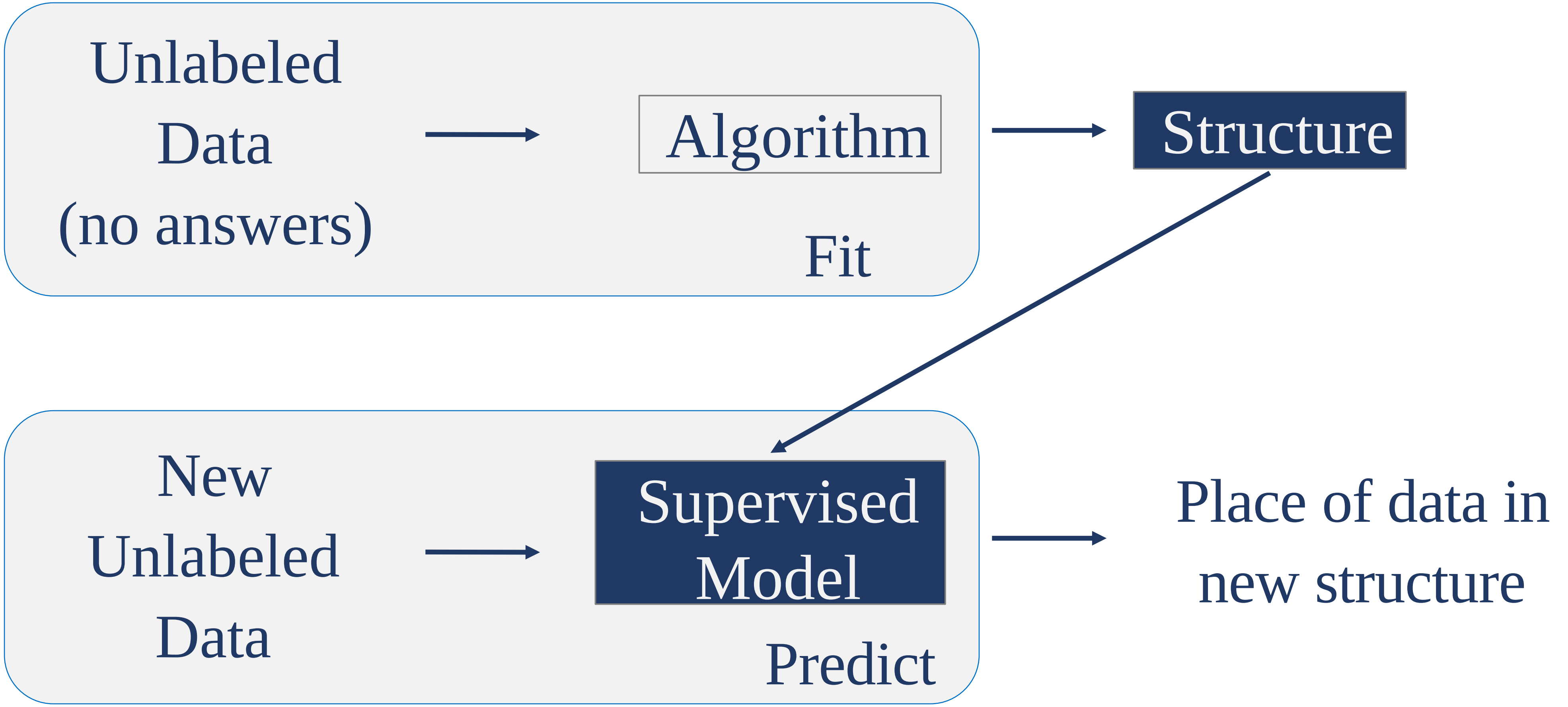
```
reduced_X_new = reducer.transform(X_new)

model.predict(reduced_X_new)

# “model” is the same model from the prior slide; could be any model.
```

CLUSTERING

WHAT IS UNSUPERVISED LEARNING?

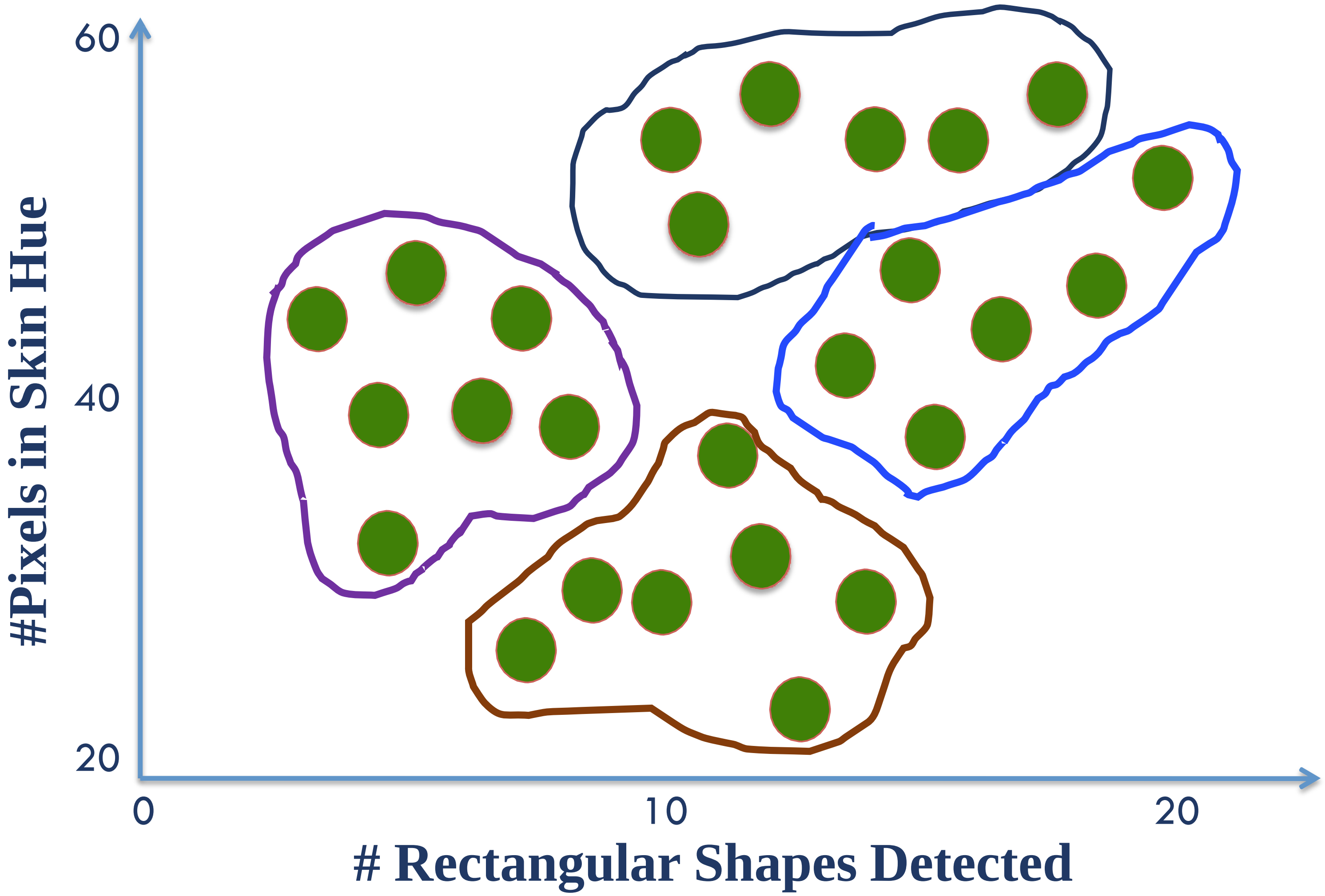


UNSUPERVISED LEARNING

Find structure in unlabeled data.

How?

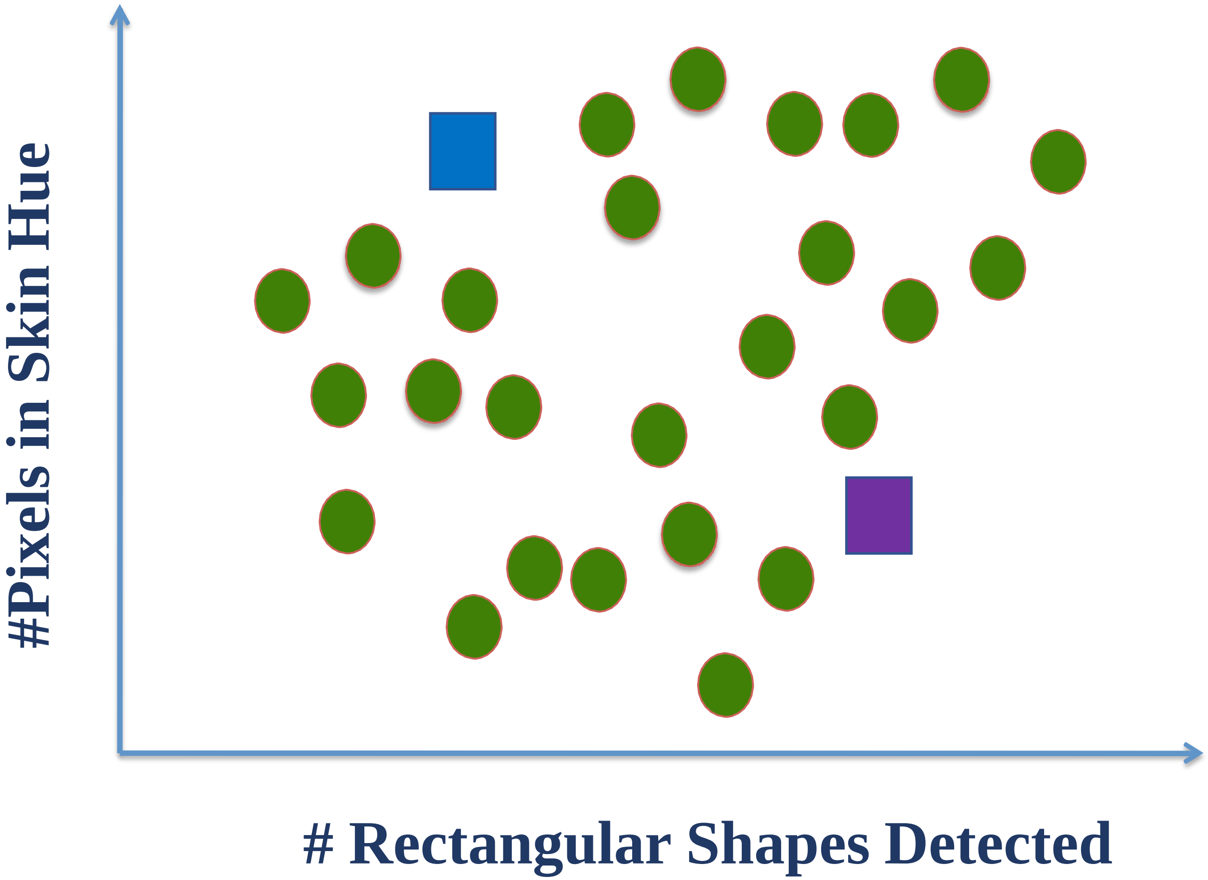
Use K-means



K-MEANS ALGORITHMS

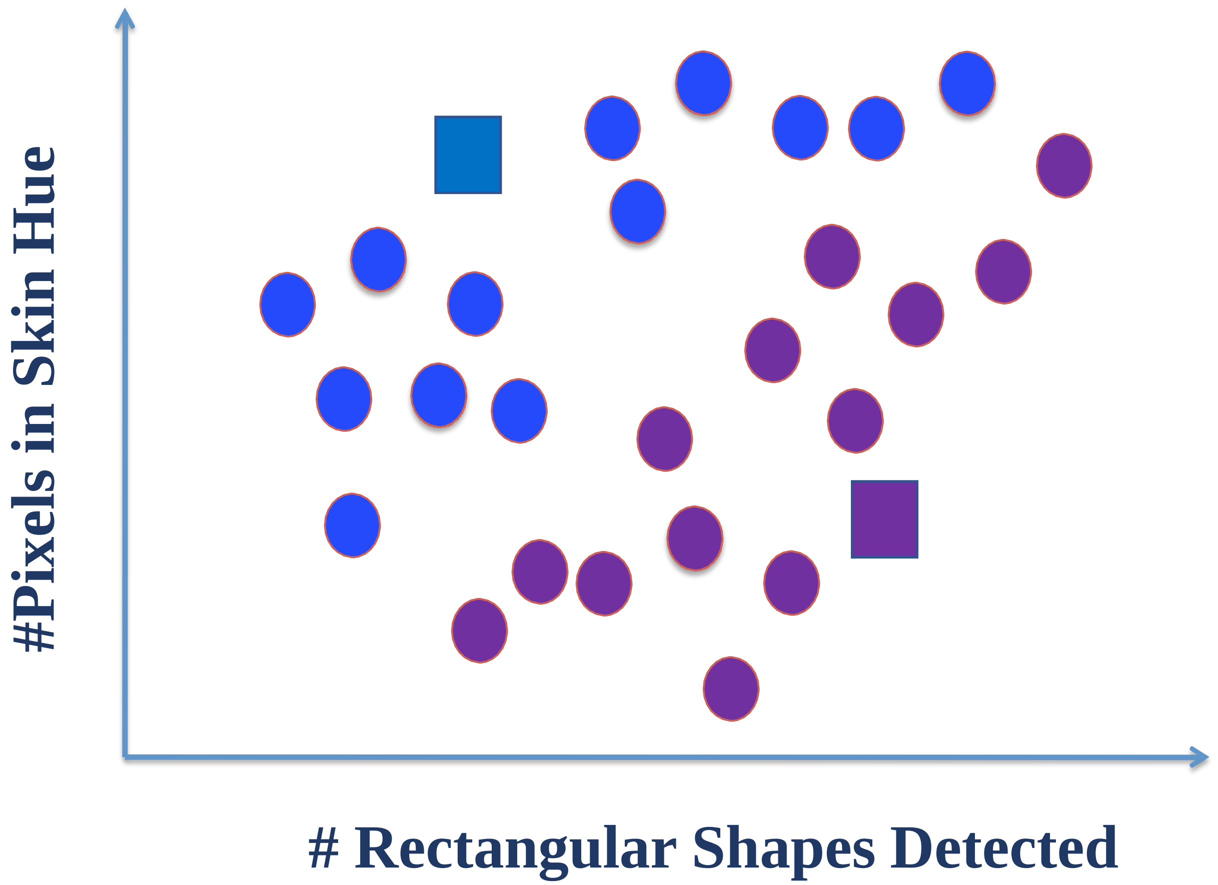
K = 2 (find two clusters)

Randomly assign two cluster centers.



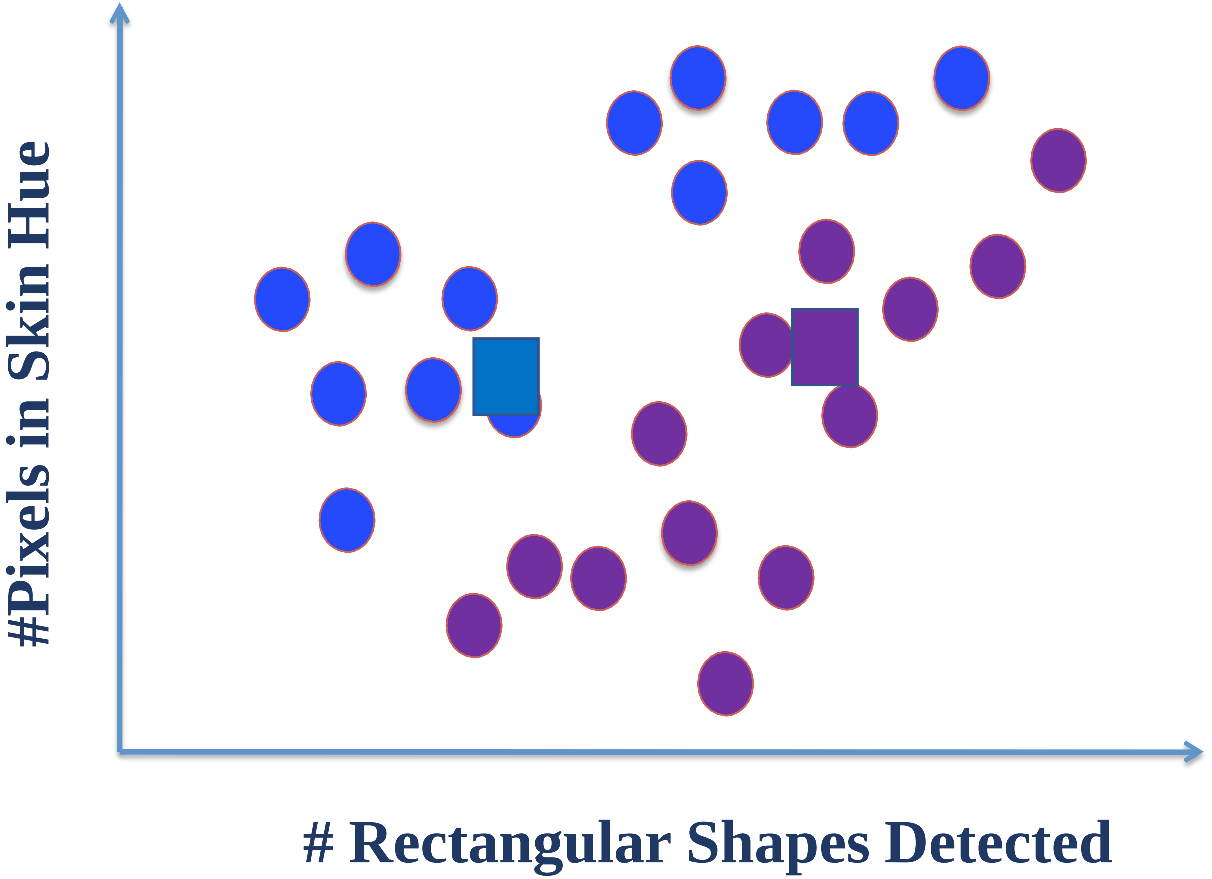
K-MEANS ALGORITHMS

Each point belongs to the closest center.



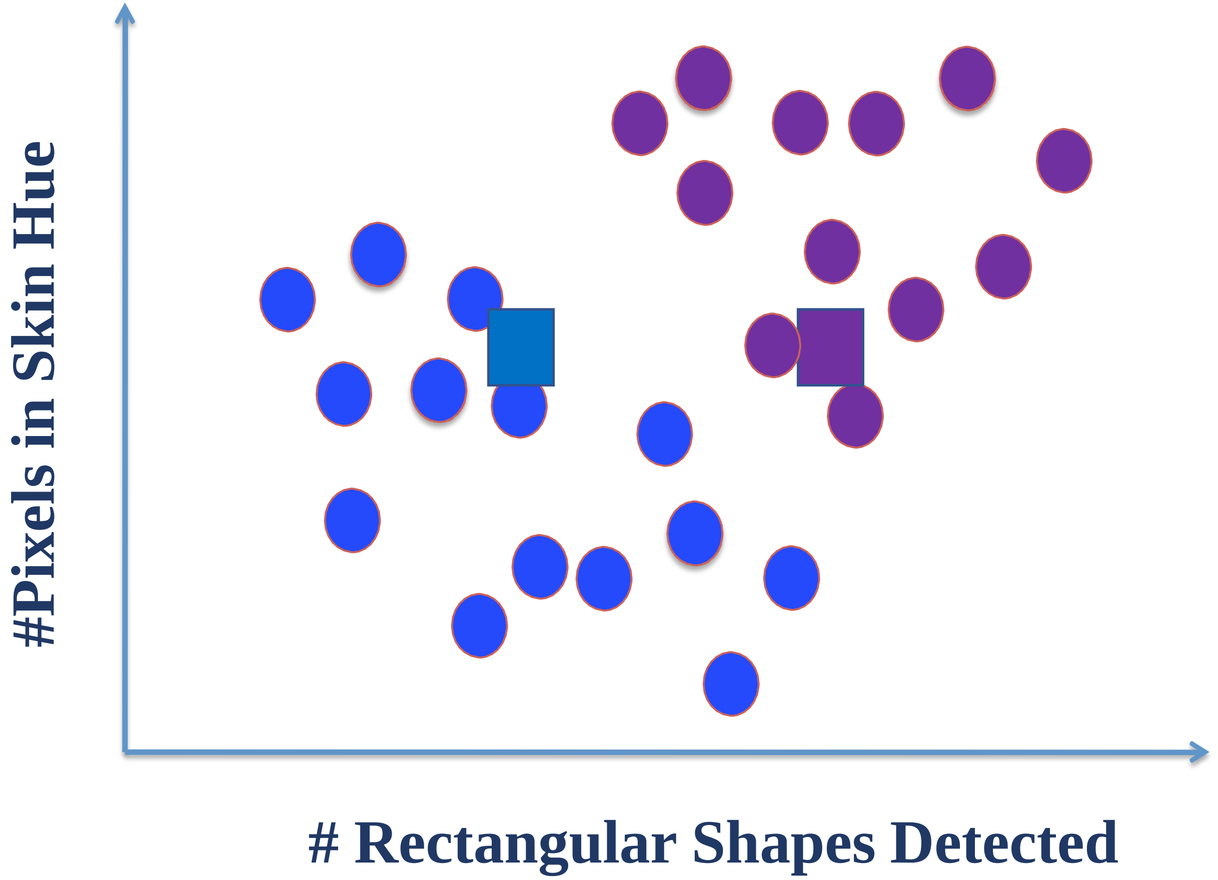
K-MEANS ALGORITHMS

Move each center to the cluster's mean.



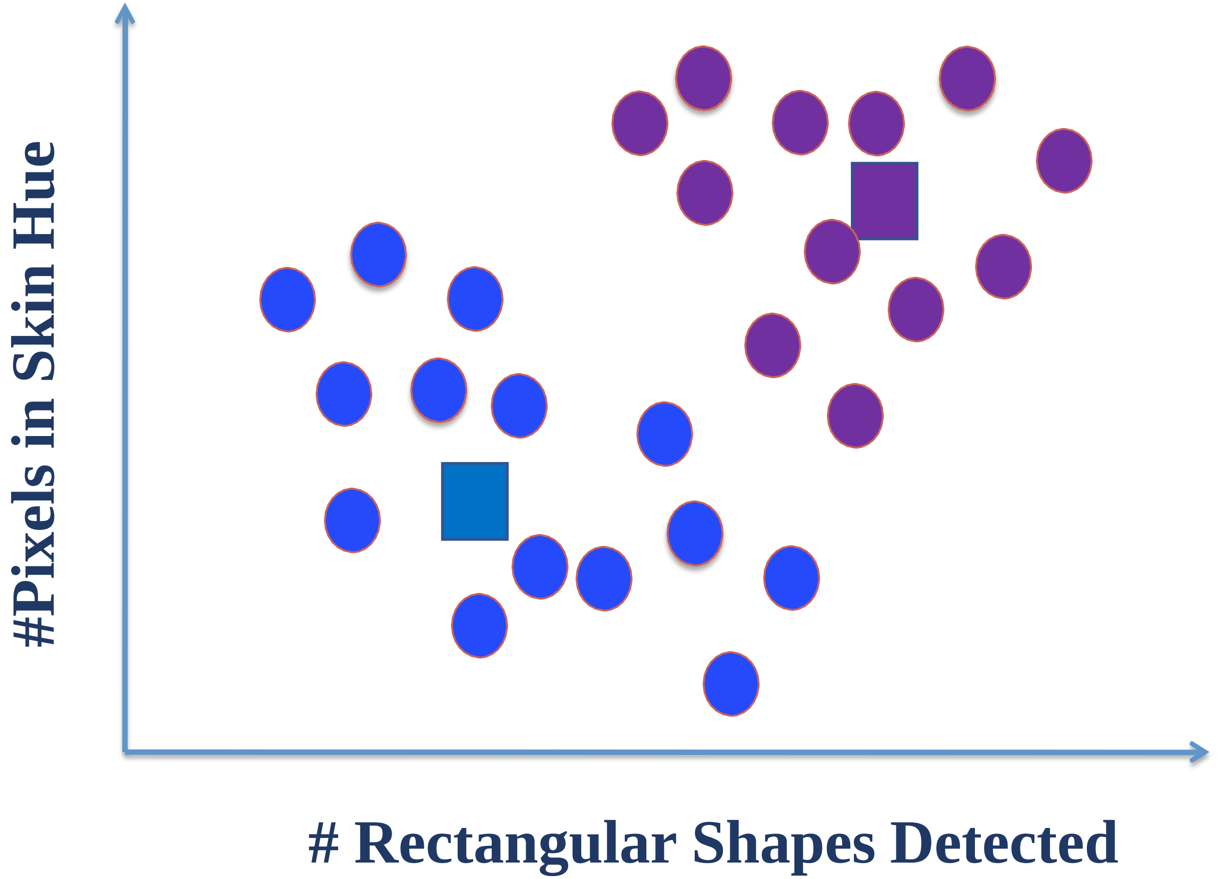
K-MEANS ALGORITHMS

Each point belongs to the *new* cluster's mean.



K-MEANS ALGORITHMS

Move each center to the *new* cluster's mean.

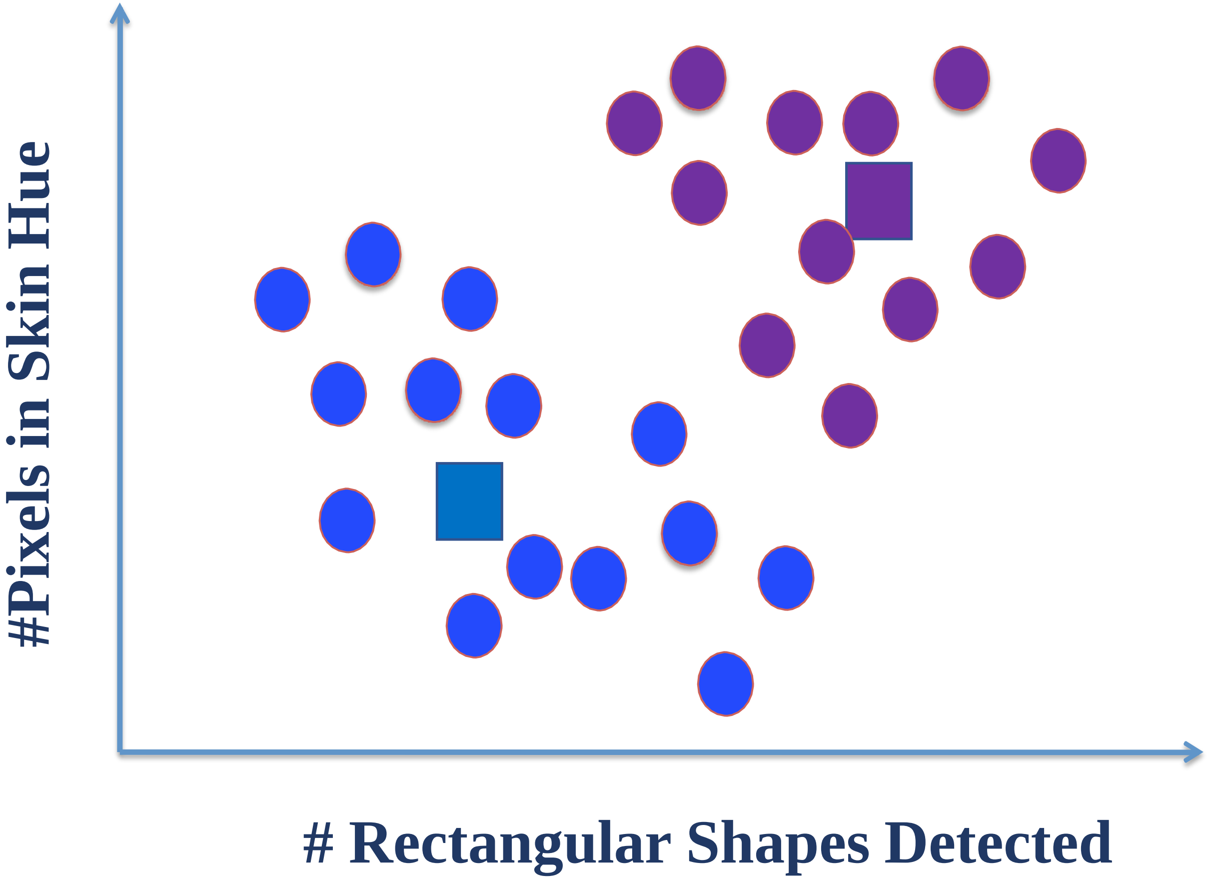


K-MEANS ALGORITHMS

Each point belongs to the *new* cluster's mean.

The points don't change anymore.

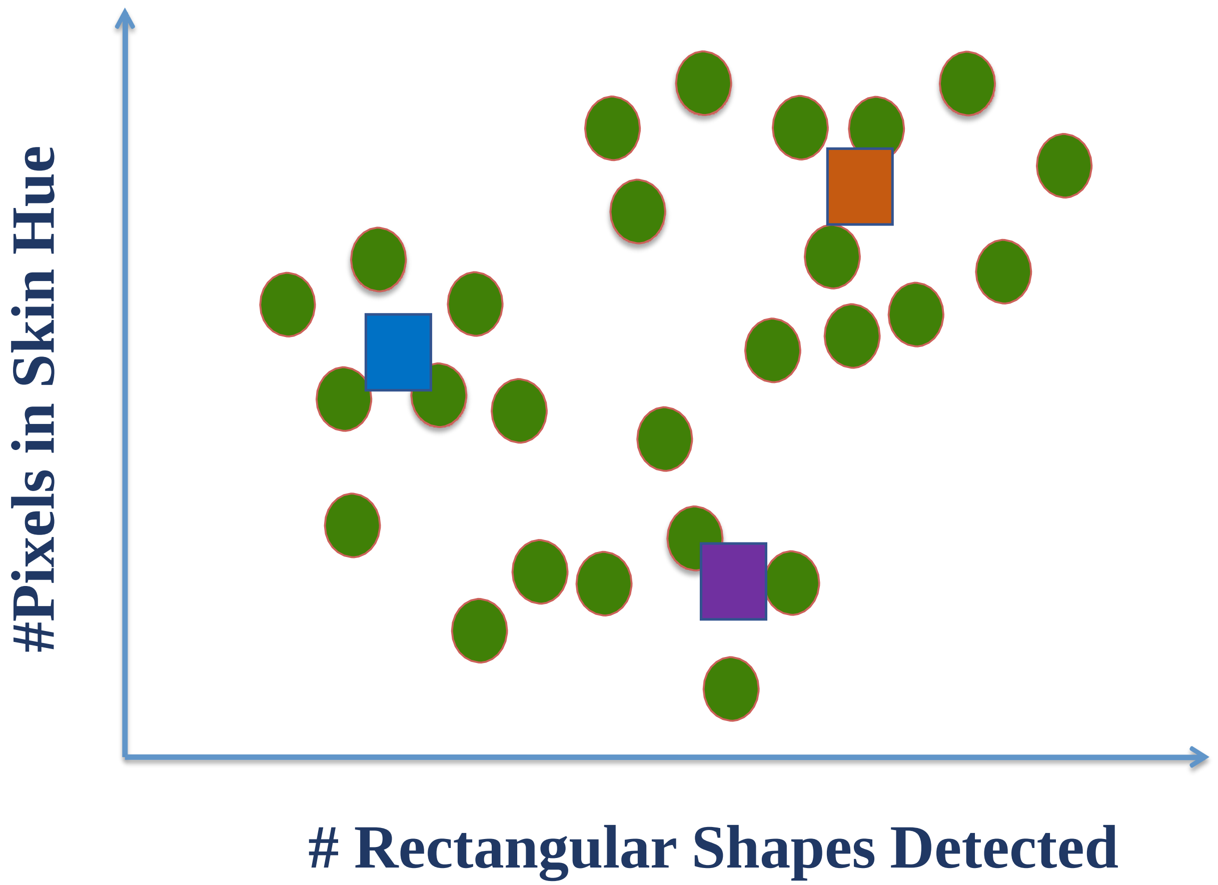
They are converged!



K-MEANS ALGORITHMS

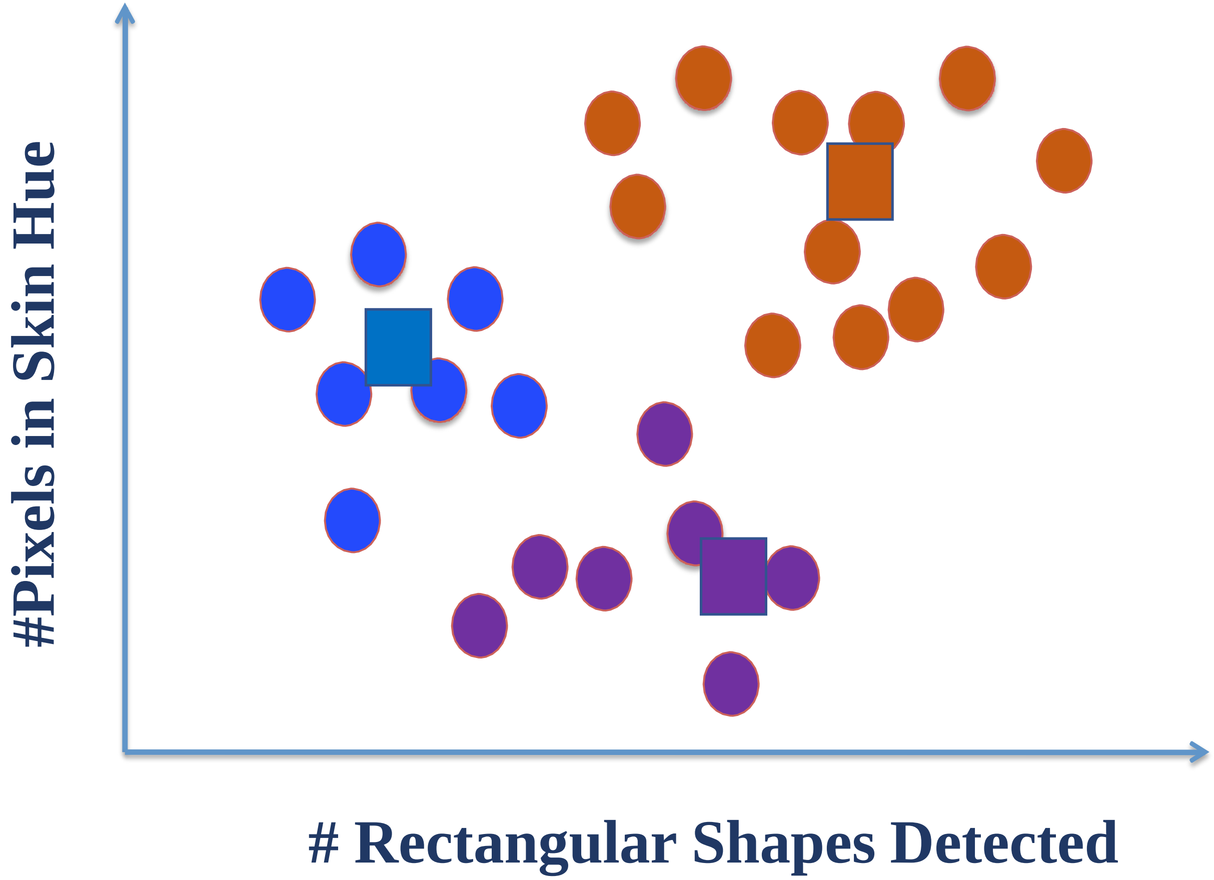
K = 3 (find three clusters)

Randomly assign three cluster centers.



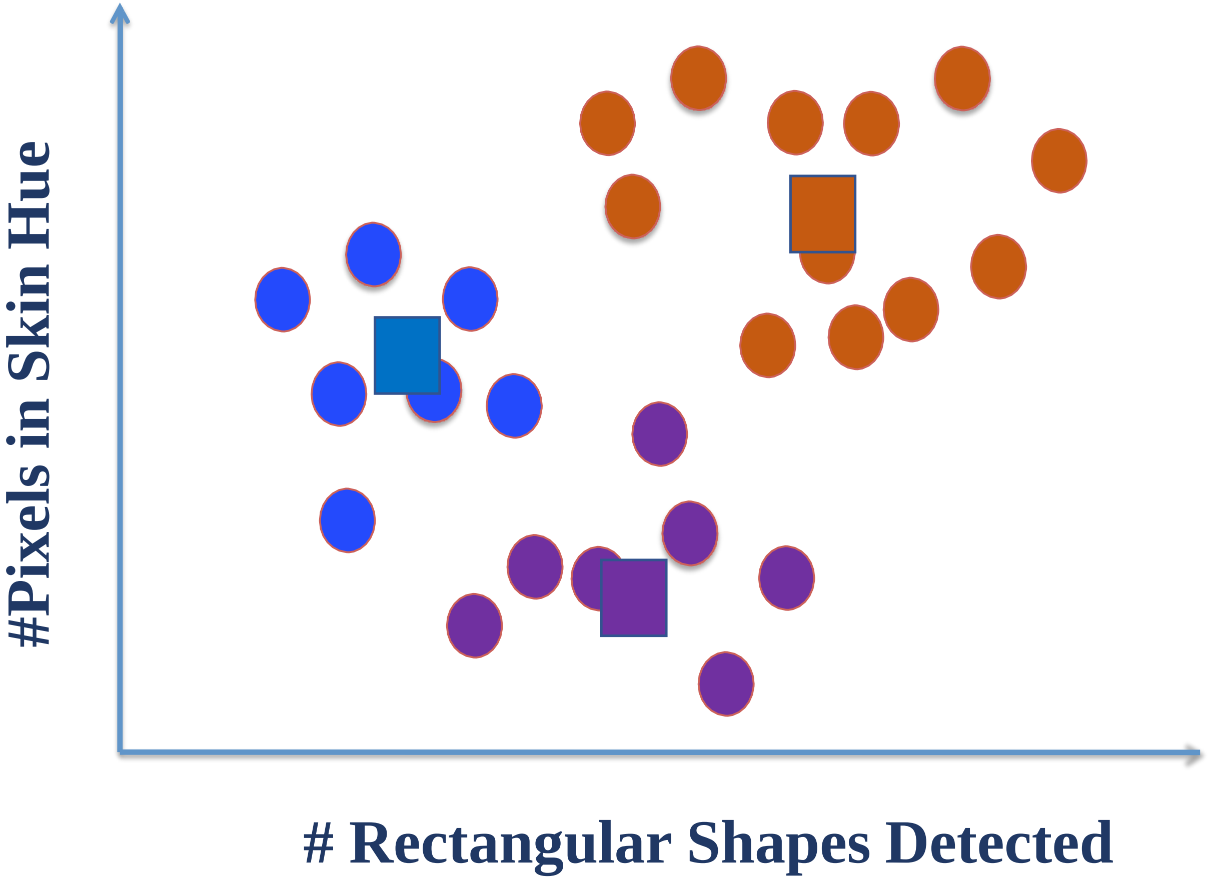
K-MEANS ALGORITHMS

Each point belongs to the closest center.



K-MEANS ALGORITHMS

Move each center to the cluster's mean.

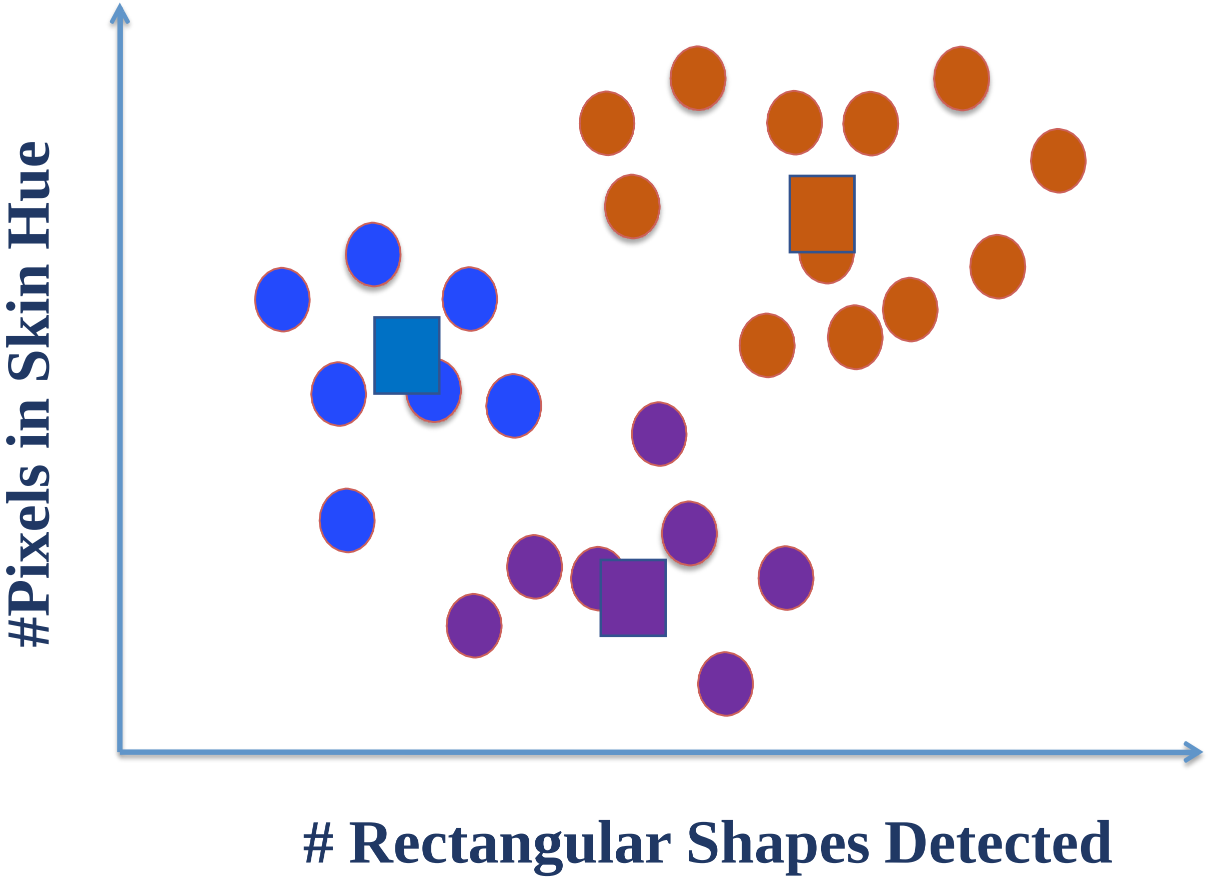


K-MEANS ALGORITHMS

Each point belongs to the *new* cluster's mean.

The points don't change anymore.

They are converged!



OTHER SCIKIT-LEARN* TOOLS

GridSearch

- Specify a space of parameters and evaluate model(s) at those params.

Pipelines

- Create a built/fit/predictable sequence of steps.
- *Fitting* w.ill fit each of the components in turn

BAG-OF-WORDS

BAG-OF-WORDS LEARNING

Image classification and recognition

Complicated architecture

Builds a vocabulary for an image based on its features.

BOW PROCESS: STEP 1

Take known image and find descriptors of keypoints.

Make a table with all information about keypoints.

- Number of columns are fixed between images.

  Columns are *how* we describe a keypoint

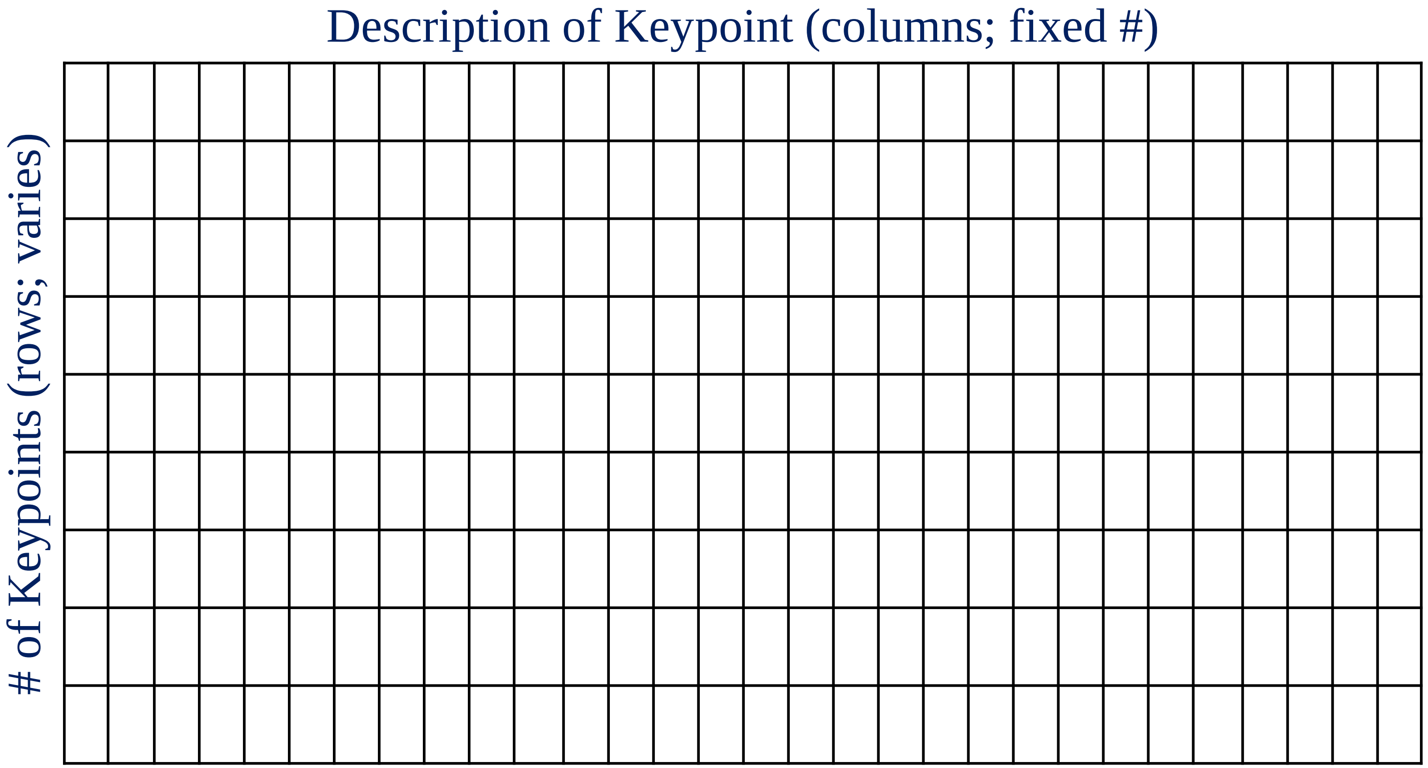
- Number of rows vary between images.

  Rows represent number of keypoints

BOW PROCESS: STEP 1

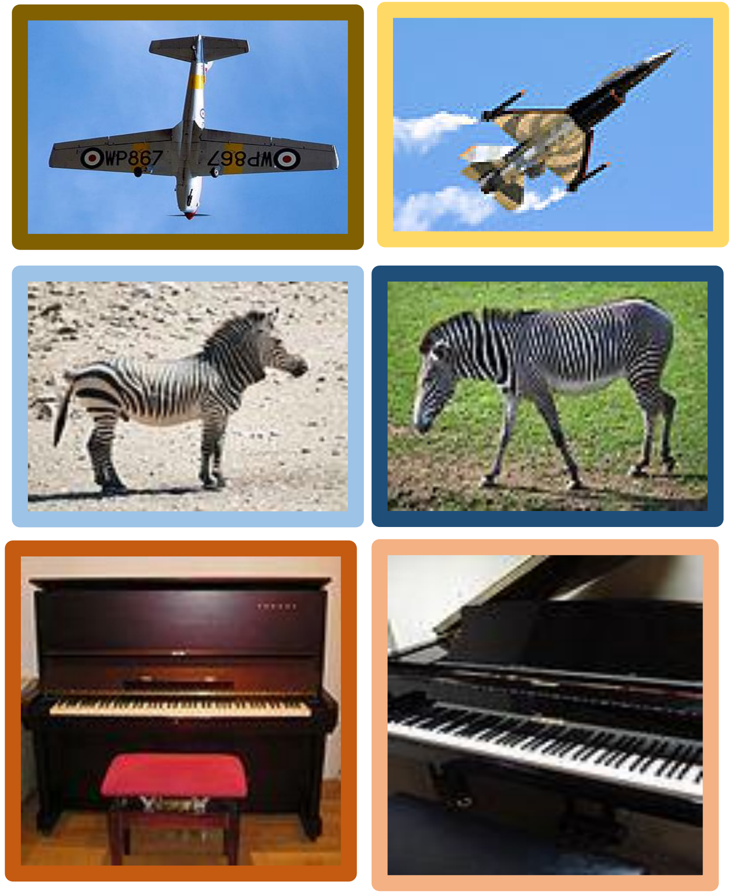
Take known image and find descriptors of keypoints.

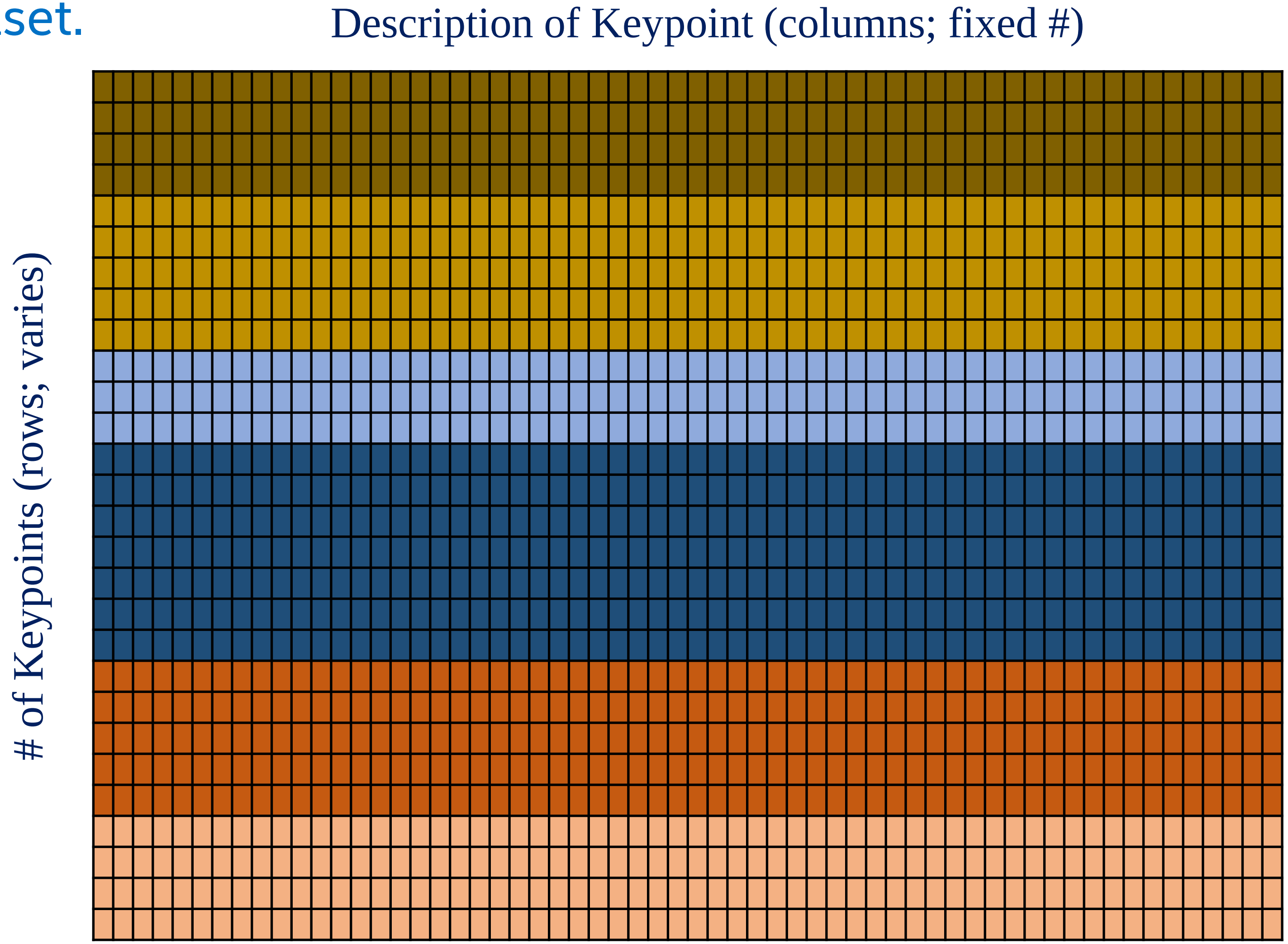
- Creating local words



BOW PROCESS: STEP1

Repeat for each image in your dataset.

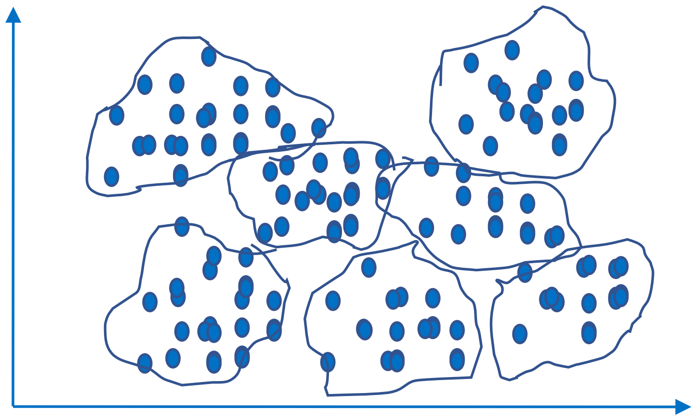




BOW PROCESS: STEP 2

Combine ALL individual descriptors.

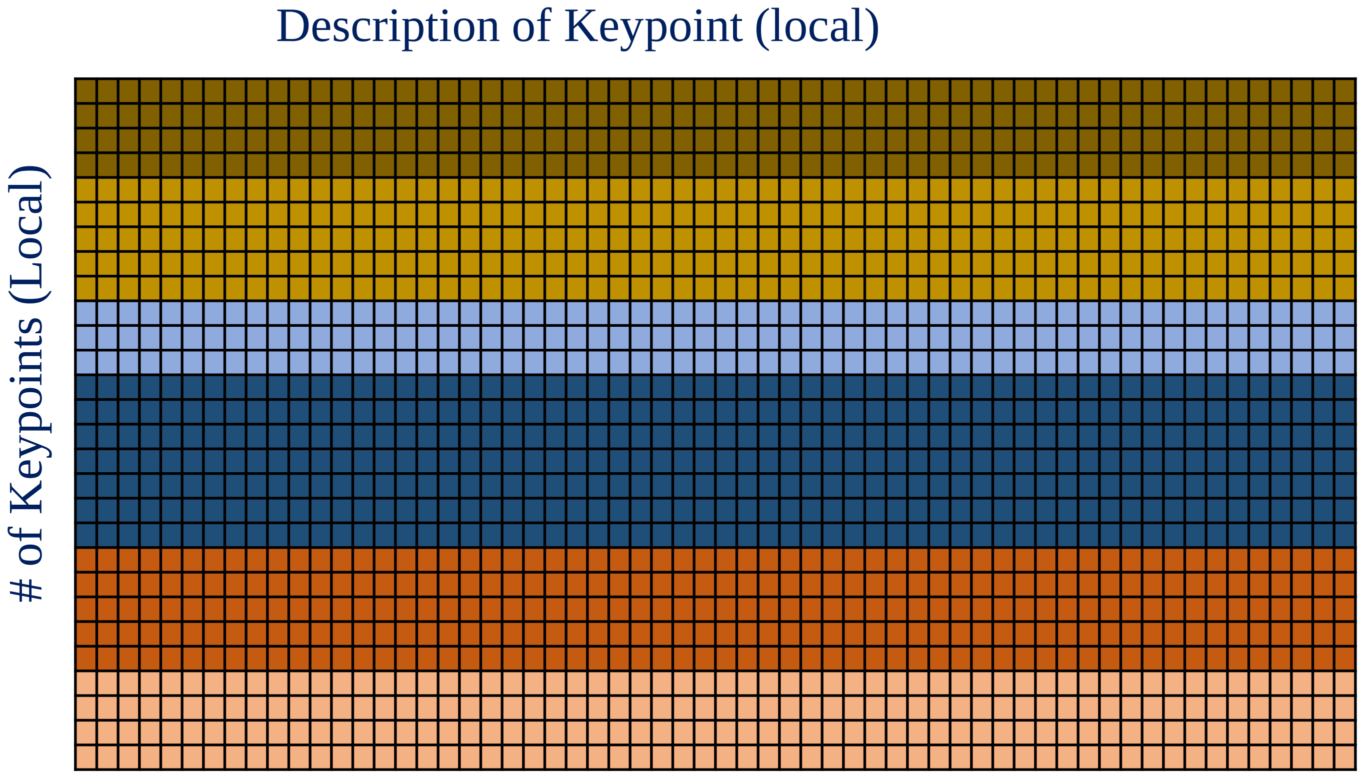
- Local vocabularies
- Put the rows in space and group clusters of data
  - Group similar descriptors together

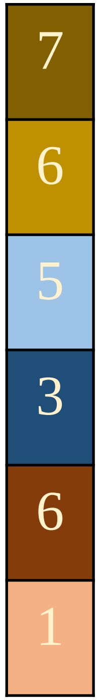


- # of clusters is the number of global words to be used for further analysis (current global vocabulary: 1, 2, 3, 4, 5, 6, 7).

BOW PROCESS: STEP 2

Convert all local words to same number of global words.



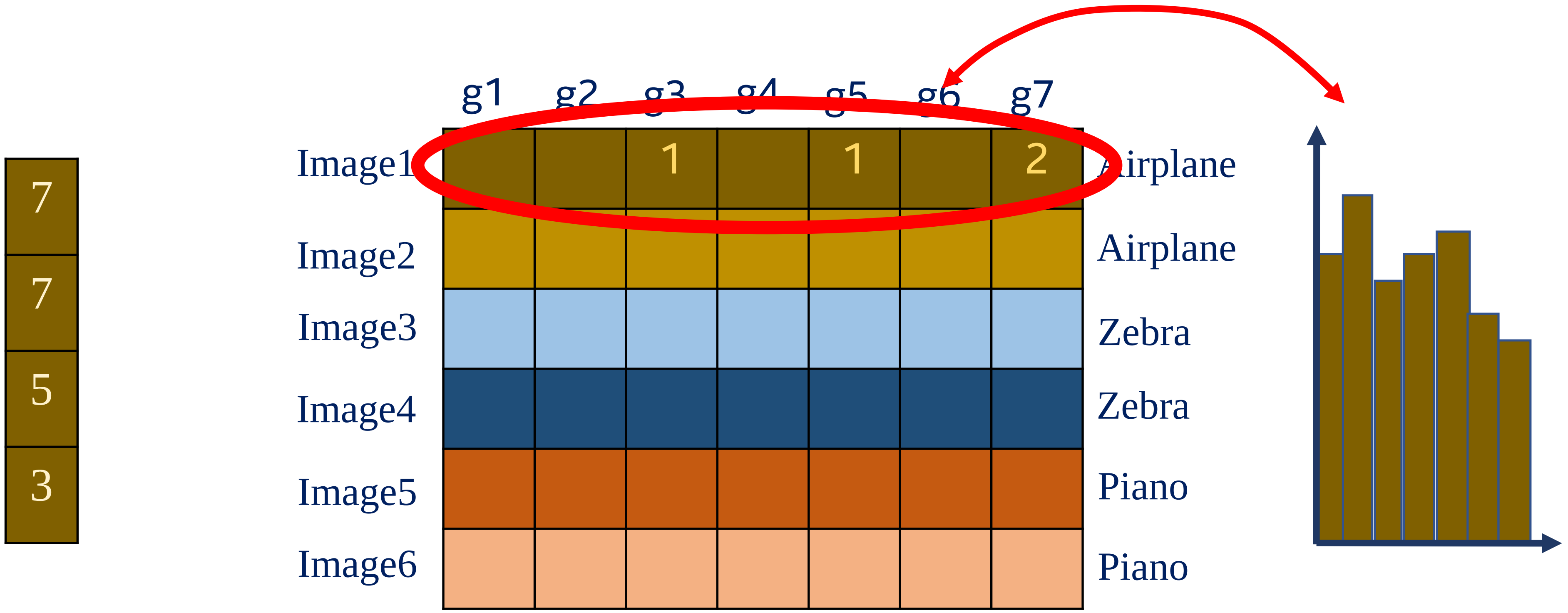


Represents ONE and only ONE row
But it represents the ENTIRE row

BOW PROCESS: STEP 3

Redescribe known images in terms of the global vocabulary.

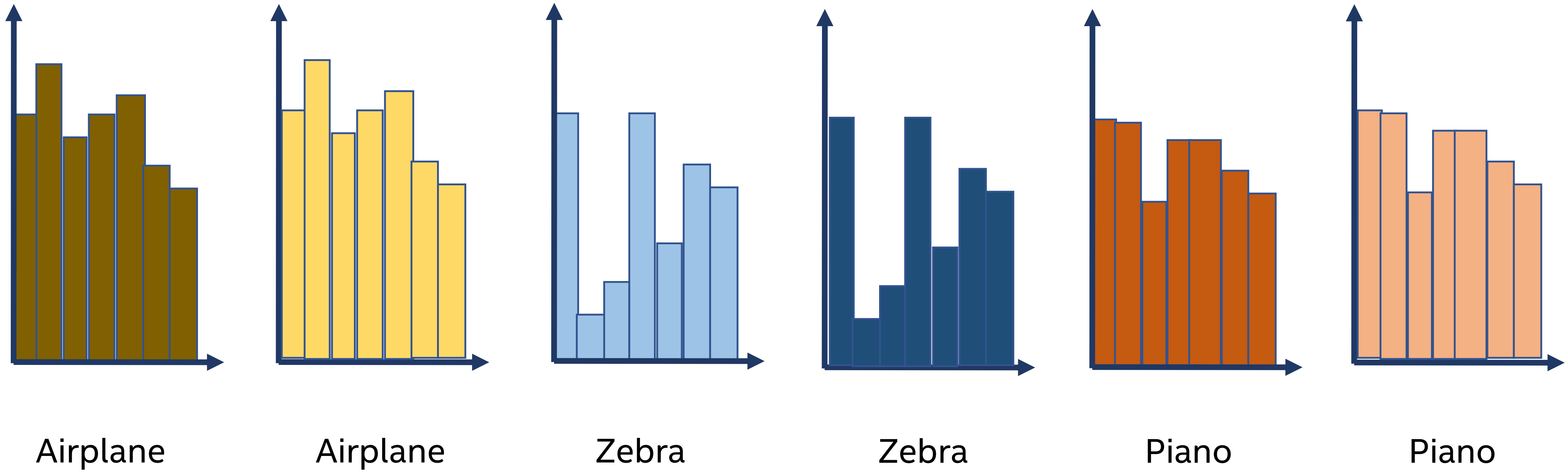
Generate histograms with counts of global words.



And likewise for other colors, empty cells are zero.

BOW PROCESS: STEP 4

Build an SVM model predicting from histograms (represented in rows) → object class



BOW PROCESS: STEP 5

With a new example:

- Describe with regional vocabulary
- Convert to global vocabulary
  Use the regional word *most similar* to the global word
- Create histogram representation
- Feed to SVM to make prediction about class

Imagine the brown zebra example without a known label.

BOW DETAILS

Local vocabularies come from feature descriptors

- SIFT, ORB, and so on

Global vocabulary comes from clustering the local vocabularies

- Convert regional to global by looking up cluster (global term) for a local descriptor
- K-means clustering