SI 568 Winter 2022 Introduction to Machine Learning

Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
University of Michigan

About your instructor

Kevyn Collins-Thompson

Associate Professor, School of Information

Associate Professor, Computer Science and Engineering

Born in London, UK – raised in Canada – career in U.S.A.

Ph.D. from School of Computer Science, Carnegie Mellon

10 years leading large-scale commercial software projects

5 years industry research (Microsoft Research)

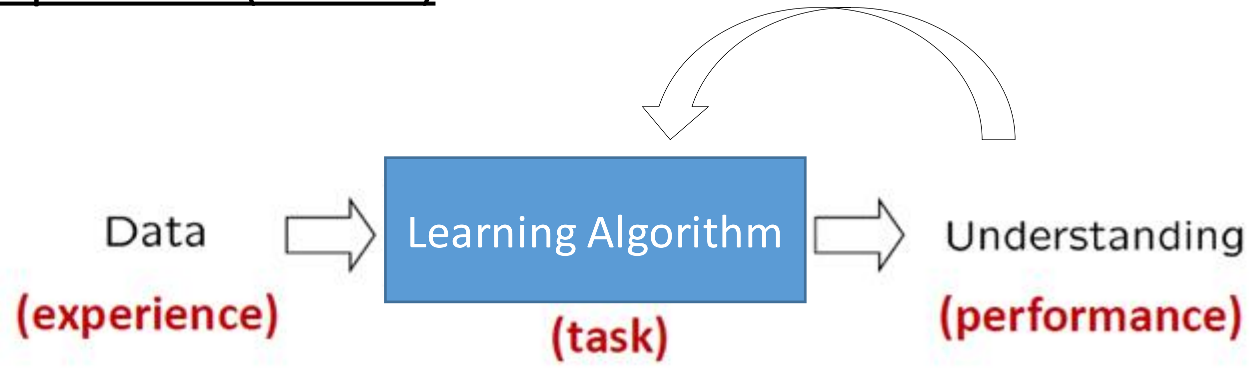
8 years academia (Michigan)

Research in connecting people with information: Personalized search, intelligent tutors for literacy, text difficulty prediction, risk-sensitive machine learning.

Applied machine learning, natural language processing, economics, psychology

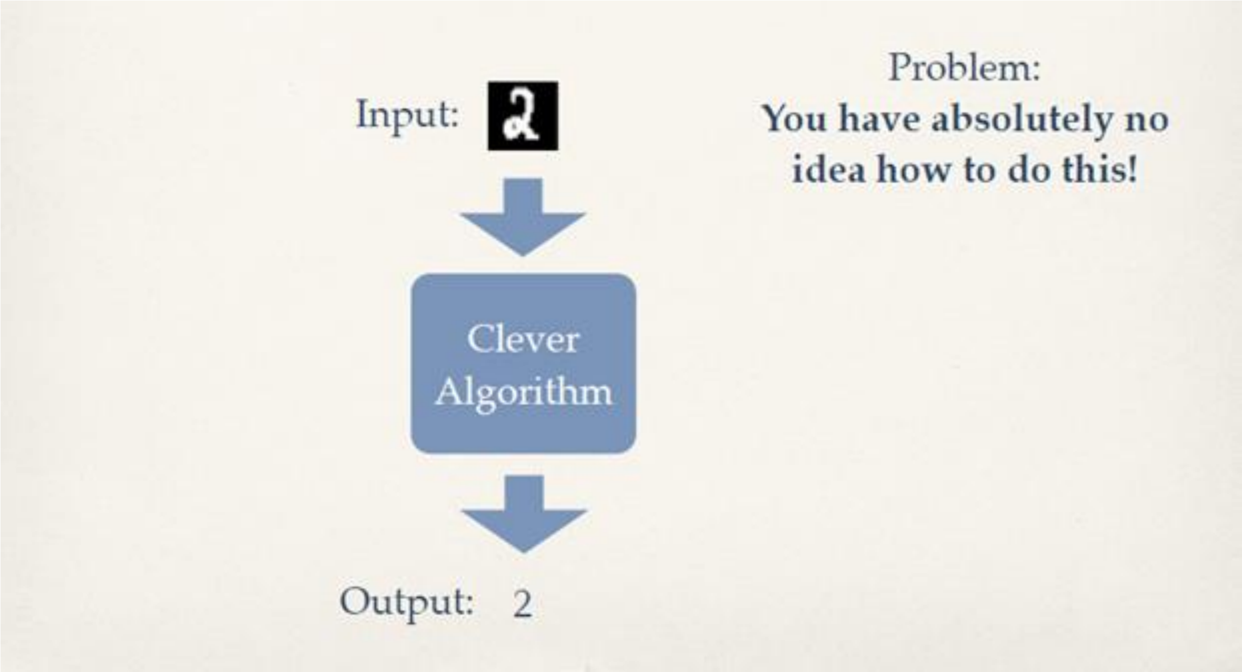
What is Machine Learning? An informal definition:

- Algorithms that improve their prediction performance at some task with experience (or data).



Example

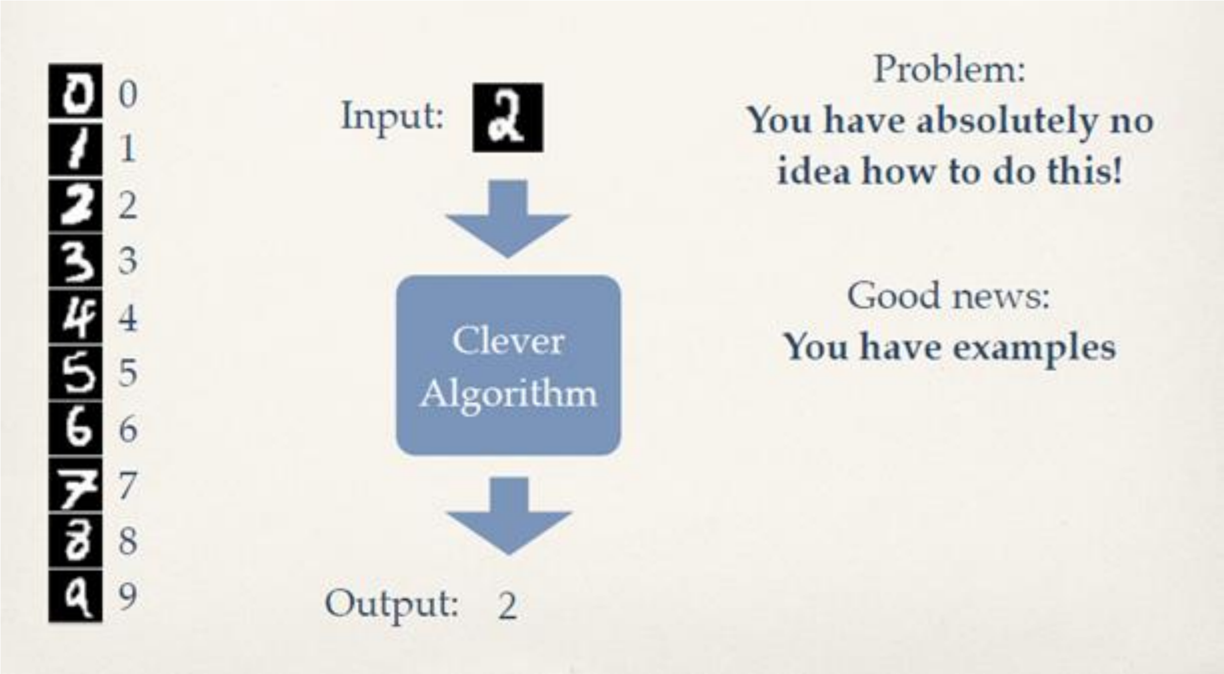
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

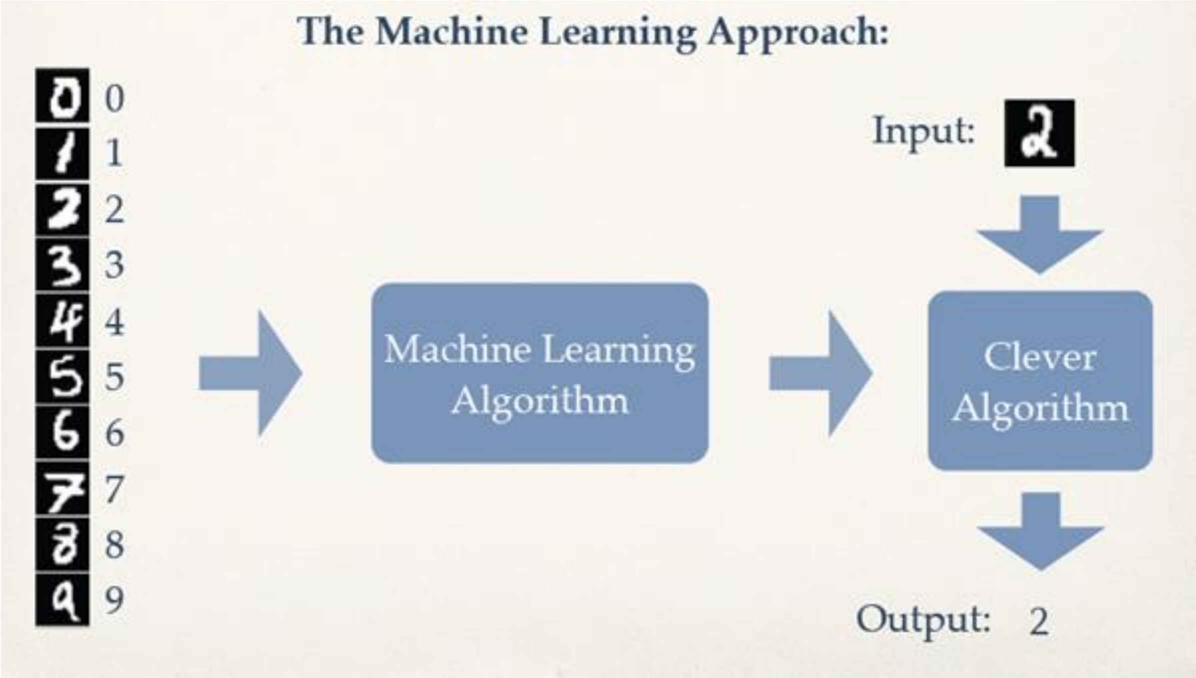
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

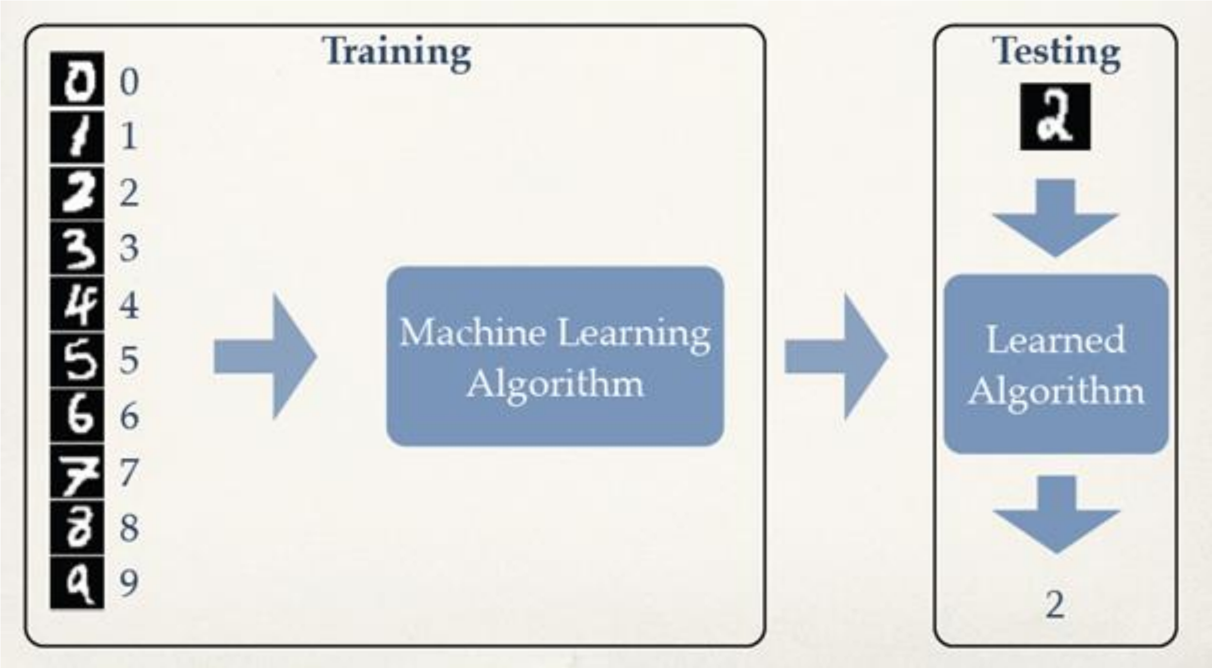
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

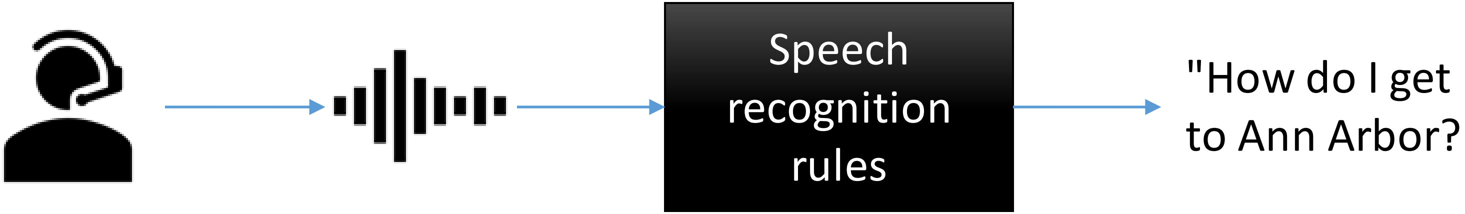
Example

- Problem: Given an image of a handwritten digit, what digit is it?

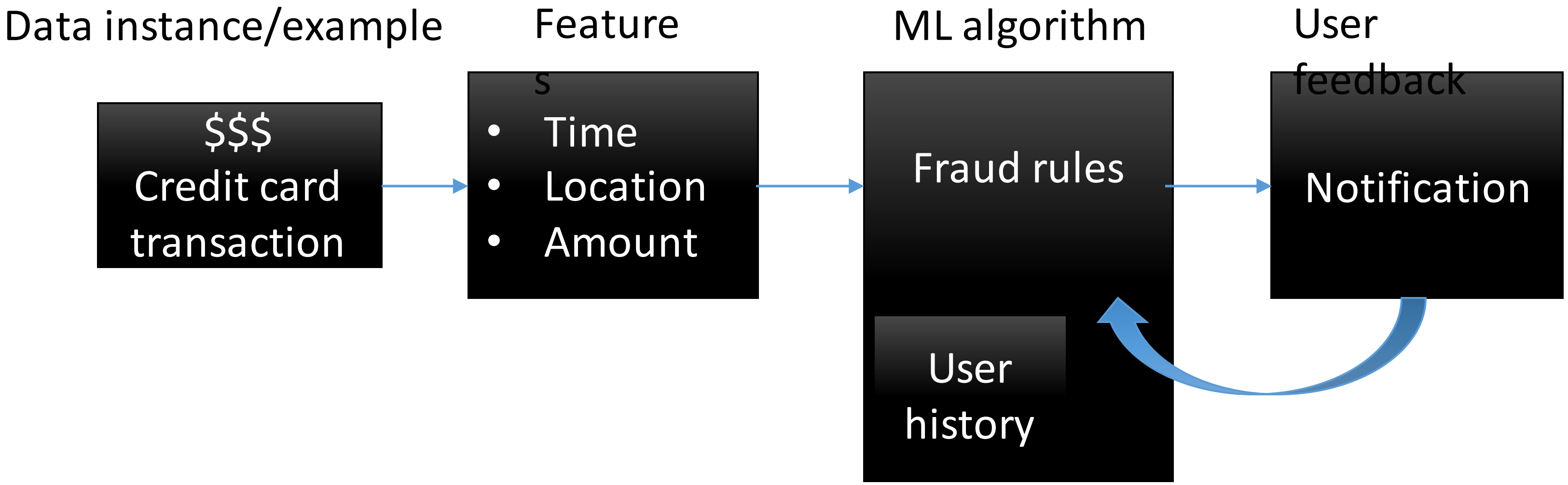


Slide credit: Kilian Weinberger

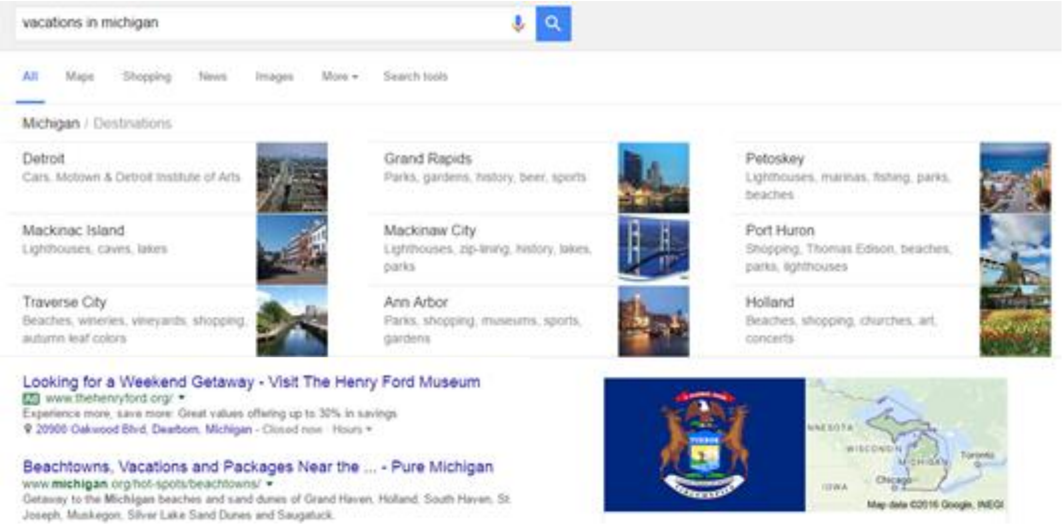
Speech Recognition



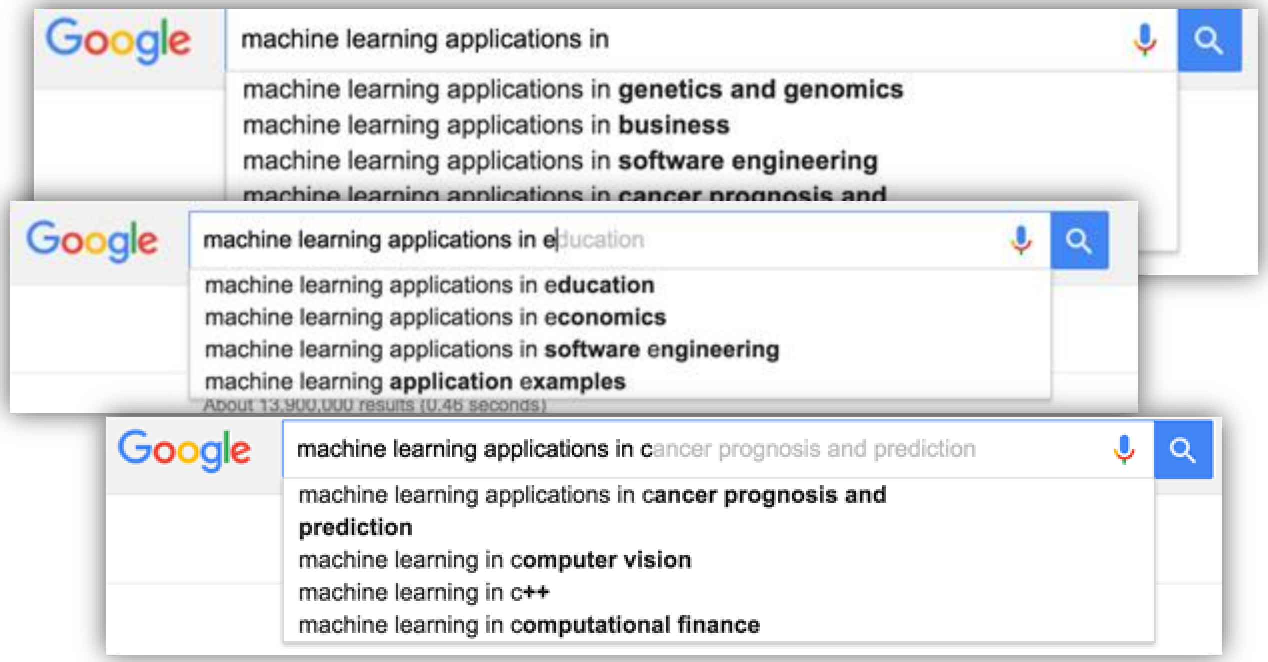
Machine Learning for fraud detection and credit scoring



Web search: query spell-checking, result ranking, content classification and selection, advertising placement



ML has huge numbers of Applications



Machine Learning algorithms are at the heart of the information economy

- Finance: fraud detection, credit scoring
- Web search results and social media feeds
- Speech recognition
- eCommerce: Product recommendations
- Email spam filtering
- Health applications: drug design and discovery
- Education: Automated essay scoring
- Judicial system: predict if someone is likely to re-offend

Discussion Question

- What are some applications for Machine Learning that most **interest** you?
- What are some applications for Machine Learning that most **scare** you?

ML Applications of Interest

Scary ML Applications

What is <u>Applied</u> Machine Learning?

- Understand basic ML **concepts** and **workflow**
- How to **properly** apply 'black-box' machine learning components and features
  - How to give the correct input (training data and labels, pre-processing)
  - How to correctly evaluate and interpret the output (test data, evaluation)
  - How to correctly adjust the “knobs” for the black box control settings (hyperparameter tuning)

- What is <u>excluded</u> by applied machine learning:
  - Underlying theory of statistical machine learning, proofs
  - Extensive low-level mathematical detail of how every ML component works

Key types of Machine Learning problems

Supervised machine learning

Learn to predict target values from labelled data.

Unsupervised machine learning

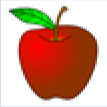
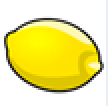
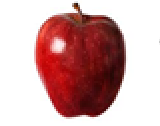
No labels! Find structure in *unlabeled data.*

Reinforcement machine learning

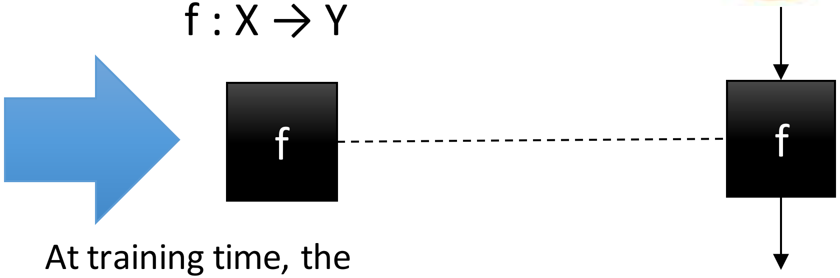
Take actions in an environment to maximize cumulative reward.

Supervised Learning (classification example)

Training set

X **Sample**		Y **Target Value (Label)**	
	x_1	Apple	y_1
	x_2	Lemon	y_2
	x_3	Apple	y_3
	x_4	Orange	y_4

Classifier
$f : X \rightarrow Y$



At training time, the classifier uses labelled examples to learn rules for recognizing each fruit type.

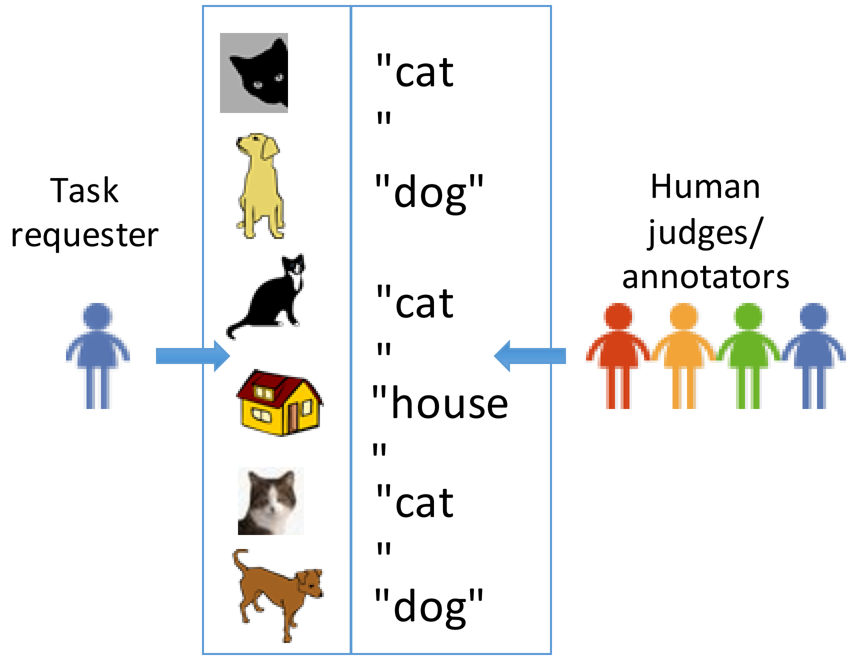
Future sample

Label: Orange

After training, at prediction time, the trained model is used to predict the fruit type for new instances using the learned rules.

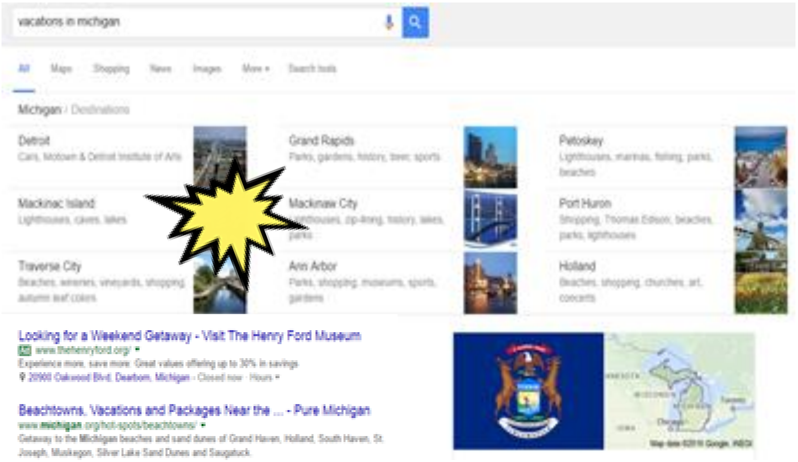
Examples of explicit and implicit label sources

Explicit labels



Crowdsourcing platform

Implicit labels

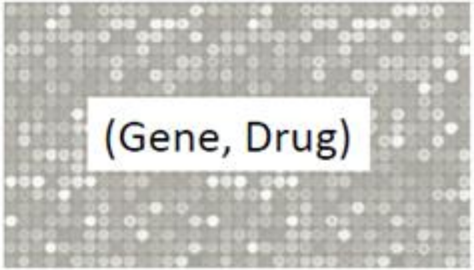


Clicking and reading the "Mackinac Island" result can be an implicit label for the search engine to learn that "Mackinac Island" is especially relevant for the query `[vacations in michigan]` for that specific user.

Supervised Learning - Regression

Feature Space $\mathcal{X}$ | **Label** Space $\mathcal{Y}$

Share Price
"$ 24.50"



Expression level
"0.01"

Continuous Labels

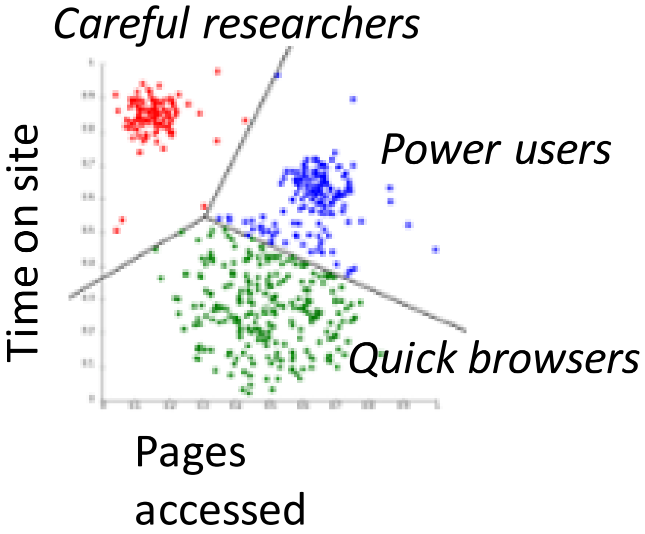
Slide credit: Aarti Singh

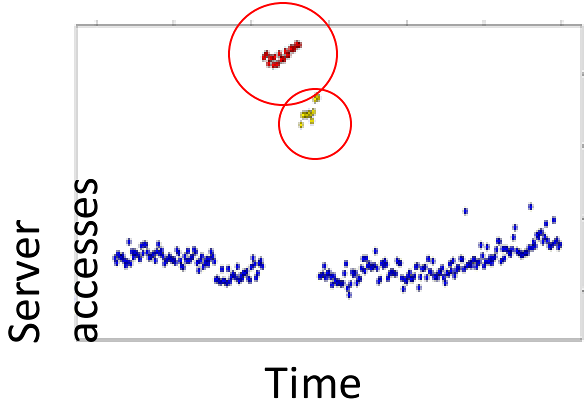
Unsupervised Learning

- Goal:
  - Given data X without any labels
  - Find interesting **structures in** the data
    - Clustering
    - Probability distribution (density estimation)
    - Embedding & neighborhood relations (e.g. community finding in network)
    - Outlier detection
- "Learning without a teacher"

Unsupervised learning: finding useful structure or knowledge in data when no labels are available

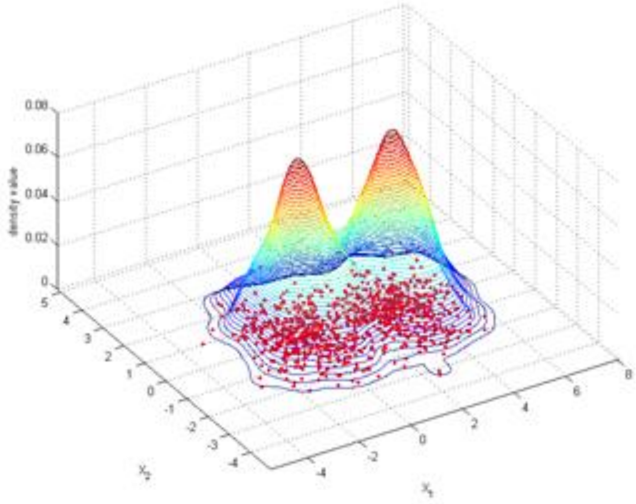
- Clustering
- Detecting abnormal server access patterns (unsupervised outlier detection)

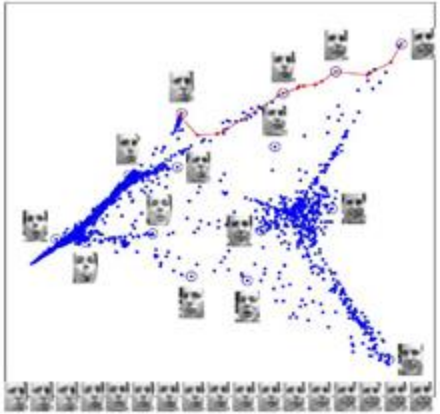




Unsupervised learning: finding useful structure or knowledge in data when no labels are available

- Density Estimation
- Reducing pixel images (several thousand pixels) into low dimensional coordinates





[Saul and Roweis, 03]

Reinforcement Learning

- Take actions in an environment to maximize cumulative reward:
- Large State Space (typically)
- Rewards in future, variable
- Exploration versus Exploitation
- Examples:
  - Games: Chess, etc.
  - Robot Navigation
  - Automatically customizing/optimizing website, HVAC, etc.
- Not covered in SI 670

Reinforcement Learning – learning to control

- Example: Robot walking
  - States: sensor inputs, joint angles
  - Action: servo commands for joints
  - Rewards:
    - 1 for reaching the goal
    - -1 for falling down
    - 0 otherwise
- Goal: How can we provide control inputs to maximize the expected future rewards?

Quiz

Finding the main verb in a sentence is an example of a:

- A) supervised learning problem
- B) unsupervised learning problem
- C) reinforcement learning problem

Cycle of Improvement:
Machine Learning Workflow

Represent / Train / Evaluate / Refine Cycle

Representation: Extract and select object features

Feature Representations

Feature representation

Email

```
To:  Chris Brooks
From:  Daniel Romero
Subject:  Next course offering
Hi Daniel,
Could you please send the outline for the
next course offering?  Thanks! -- Chris
```

Feature	Count
to	1
chris	2
brooks	1
from	1
daniel	2
romero	1
the	2
...	

A list of words with their frequency counts

Picture

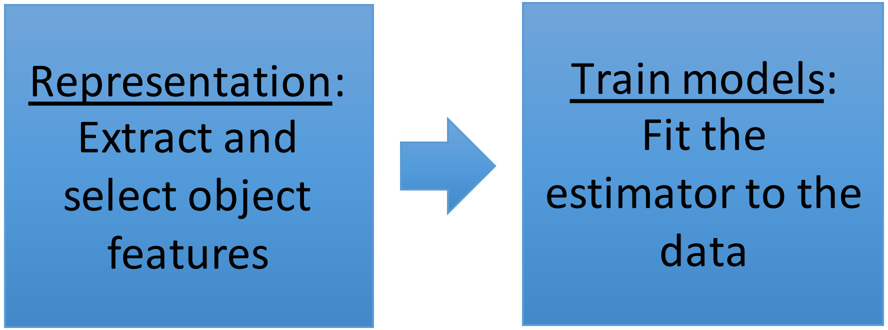
A matrix of color values (pixels)

Sea Creatures

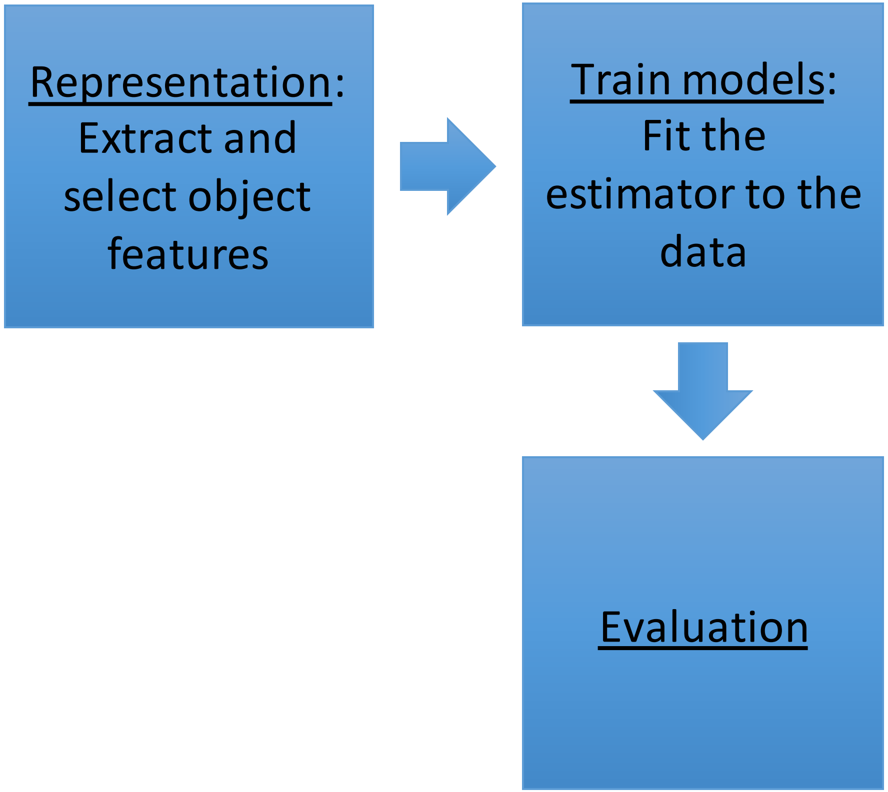
Feature	Value
DorsalFin	Yes
MainColor	Orange
Stripes	Yes
StripeColor1	White
StripeColor2	Black
Length	4.3 cm

A set of attribute values

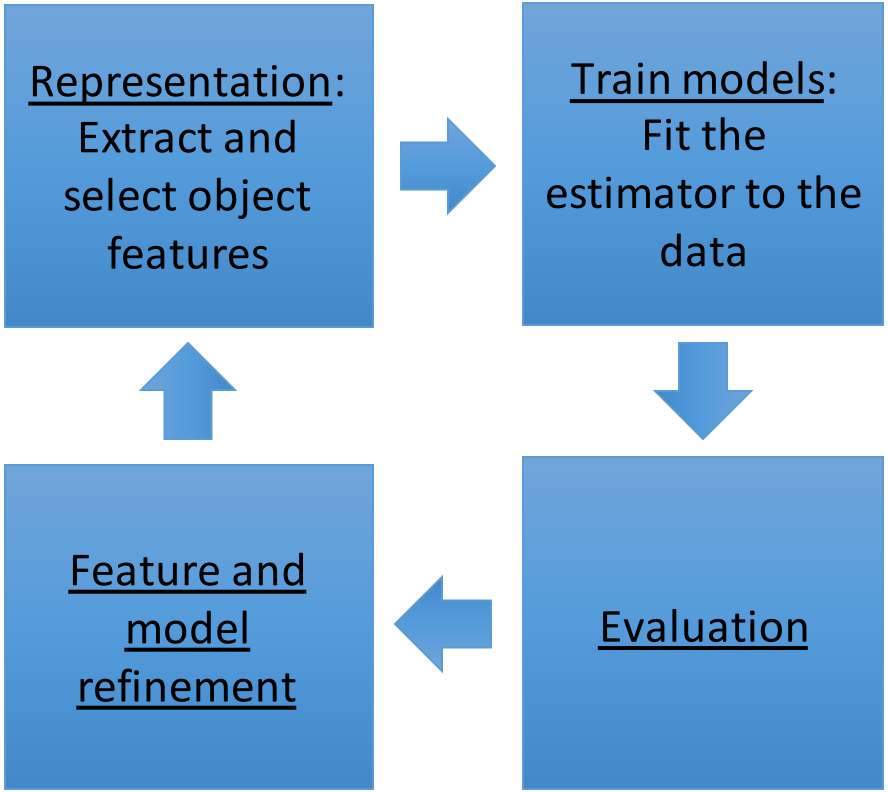
Represent / Train / Evaluate / Refine Cycle



Represent / Train / Evaluate / Refine Cycle

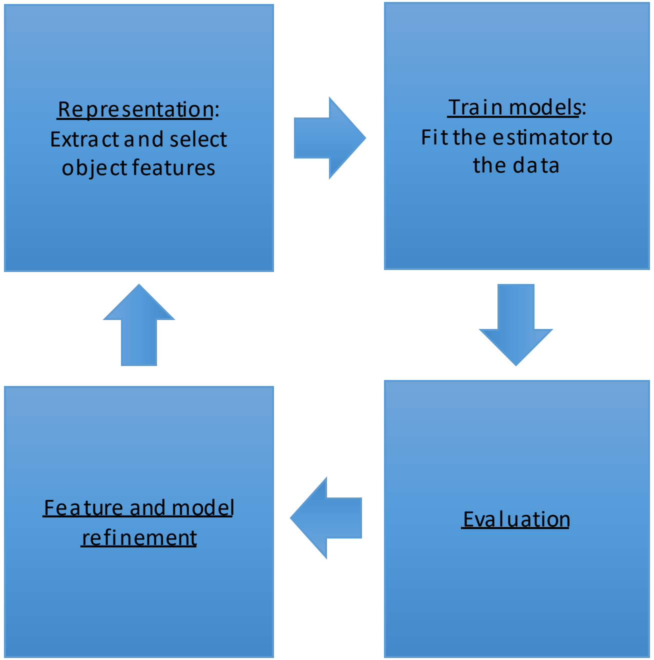


Represent / Train / Evaluate / Refine Cycle



Algorithm Knowledge

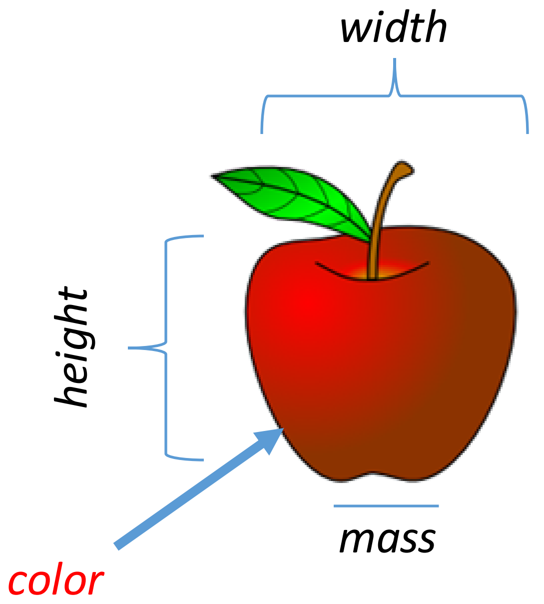
- Parameters to tune
- Transparency of operations
- Intuition to feature engineer



Workflow example: Fruit!

Representation

Representing a piece of fruit as an array of features (plus label information)



1. Feature representation

Label information
(available in training data only)

Feature
representation

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
18	1	apple	cripps_pink	162	7.5	7.1	0.83

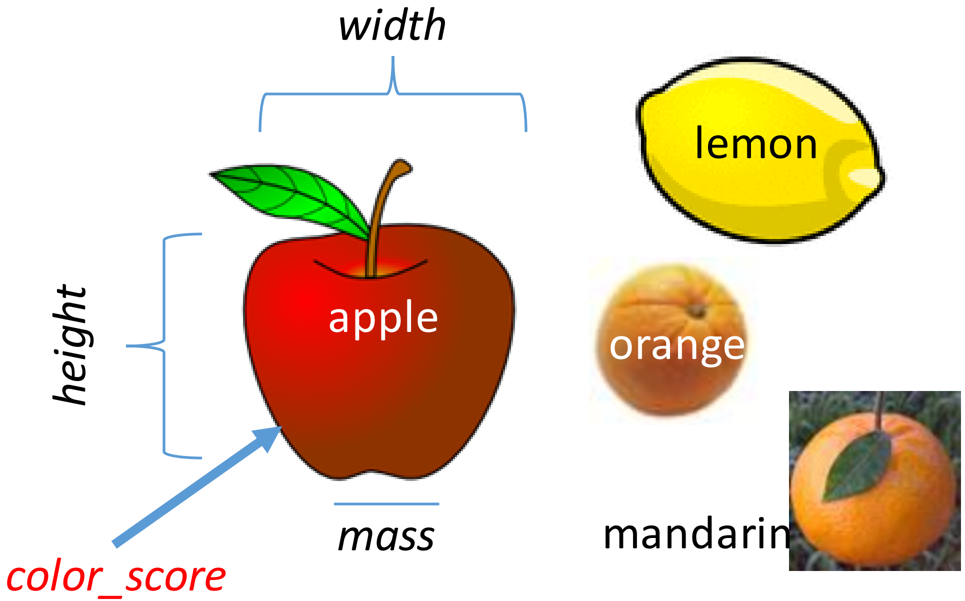
2. Learning model

Classifier

Predicted
class
(apple)

Creating Training and Testing Sets with `train_test_split`

The Fruit Dataset



	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

`fruit_data_with_colors.txt`

<u>Credit</u>: Original version of the fruit dataset created by Dr. Iain Murray, Univ. of Edinburgh

The input data as a table

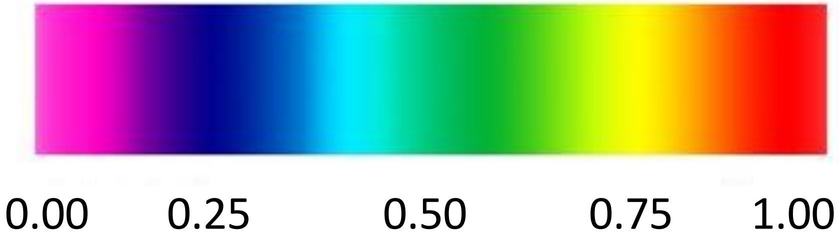
Each row corresponds to a single data instance (sample)

	fruit_label	ruit_name	fruit_subtype	mass	width	height	color_score
0	1	pple	granny_smith	192	8.4	7.3	0.55
1	1	pple	granny_smith	180	8.0	6.8	0.59
2	1	pple	granny_smith	176	7.4	7.2	0.60
3	2	andarin	mandarin	86	6.2	4.7	0.80
4	2	andarin	mandarin	84	6.0	4.6	0.79
5	2	andarin	mandarin	80	5.8	4.3	0.77
6	2	andarin	mandarin	80	5.9	4.3	0.81
7	2	andarin	mandarin	76	5.8	4.0	0.81
8	1	pple	braeburn	178	7.1	7.8	0.92
9	1	pple	braeburn	172	7.4	7.0	0.89
1	1	pple	braeburn	166	6.9	7.3	0.93
1	1	pple	braeburn	172	7.1	7.6	0.92
1	1	pple	braeburn	154	7.0	7.1	0.88
1	1	pple	golden_delicious	164	7.3	7.7	0.70
1	1	pple	golden_delicious	152	7.6	7.3	0.69
1	1	pple	golden_delicious	156	7.7	7.1	0.69
1	1	pple	golden_delicious	156	7.6	7.5	0.67
1	1	pple	golden_delicious	168	7.5	7.6	0.73
1	1	pple	cripps_pink	162	7.5	7.1	0.83
1	1	pple	cripps_pink	162	7.4	7.2	0.85

These four columns contain the features of each data instance (sample)

The `fruit_label` column contains the label for each data instance (sample)

The scale for the (simplistic) `color_score` feature used in the fruit dataset



Color category `color_score`		
	Red	0.85 - 1.00
	Orange	0.75 - 0.85
	Yellow	0.65 - 0.75
	Green	0.45 - 0.65

Our first classifier: k-Nearest Neighbor

The k-Nearest Neighbor (k-NN) Classifier Algorithm

Given a training set X_{train} with labels y_{train}, and given a new instance x_{test} to be classified:

1. Find the **most similar** instances (let's call them X_{NN}) to x_{test} that are in X_{train}.
2. Get the labels y_{NN} for the instances in X_{NN}
3. Predict the label for x_{test} by combining the labels y_{NN}
   e.g. simple majority vote

Training set

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

New object to be classified:

mass	width	height	color_score
167	6.8	7.2	0.92

1-nearest neighbor: Find label of most similar object seen in the training set, use that label as the prediction for the new object

Given a training set X_{train} with labels y_{train}, and given a new instance x_{test} to be classified:

1. Find the **most similar** instances (let's call them X_{NN}) to x_{test} that are in X_{train}.
2. Get the labels y_{NN} for the instances in X_{NN}
3. Predict the label for x_{test} by combining the labels y_{NN}
   e.g. simple majority vote

Training set

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

????

New object to be classified:

mass	width	height	color_score
167	6.8	7.2	0.92

1-nearest neighbor: Find label of most similar object seen in the training set, use that label as the prediction for the new object

Given a training set X_{train} with labels y_{train}, and given a new instance x_{test} to be classified:

1. Find the **most similar** instances (let's call them X_{NN}) to x_{test} that are in X_{train}.
2. Get the labels y_{NN} for the instances in X_{NN}
3. Predict the label for x_{test} by combining the labels y_{NN}
   e.g. simple majority vote

Training set

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

????

New object to be classified:

mass	width	height	color_score
167	6.8	7.2	0.92

1-nearest neighbor: Find label of most similar object seen in the training set, use that label as the prediction for the new object

Given a training set X_{train} with labels y_{train}, and given a new instance x_{test} to be classified:

1. Find the **most similar** instances (let's call them X_{NN}) to x_{test} that are in X_{train}.
2. Get the labels y_{NN} for the instances in X_{NN}
3. Predict the label for x_{test} by combining the labels y_{NN}
   e.g. simple majority vote

Training set

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

apple

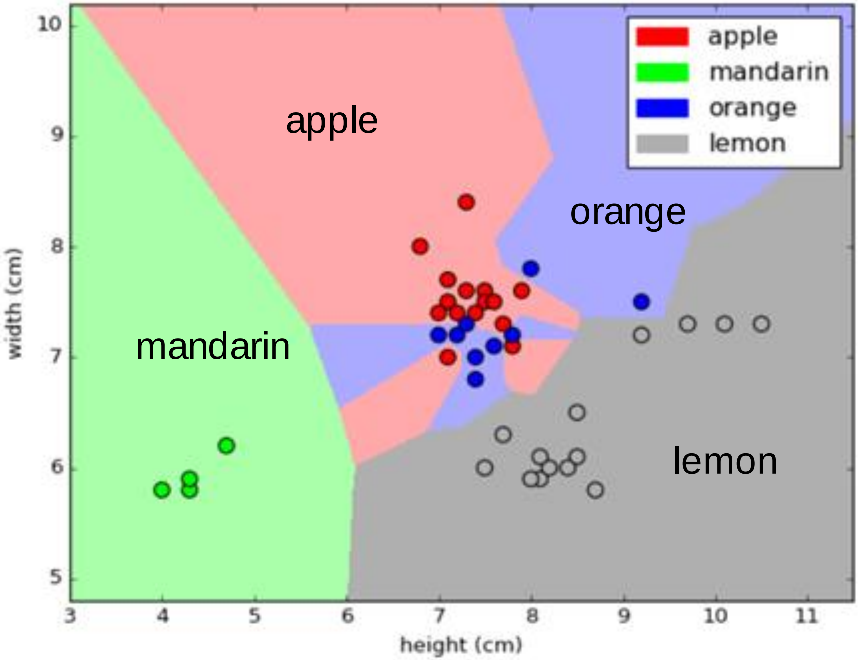
New object to be classified:

mass	width	height	color_score
167	6.8	7.2	0.92

Introducing a two-dimensional feature space showing a classifier’s decision boundaries

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79

- Each data point is a piece of fruit
- X-axis: the height of the fruit
- Y-axis: the width of the fruit
- Four possible fruit classes to predict
- For any point (x, y) in the feature space the color of (x, y) represents the fruit class the classifier would predict for a piece of fruit with height x, width y
- The boundaries between different colored regions are called decision boundaries between classes.

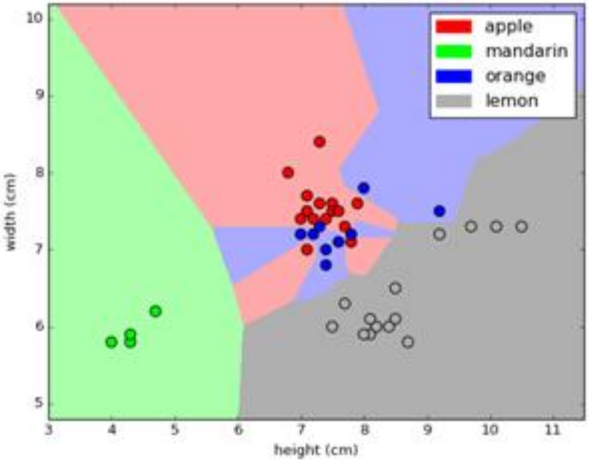


K=1

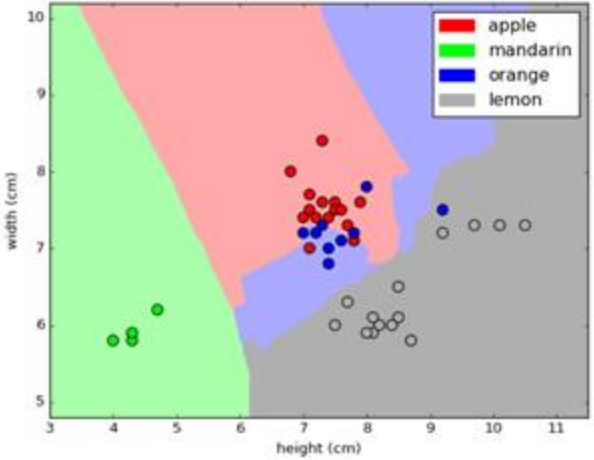
Feature space: plot the decision regions

Points show the training data

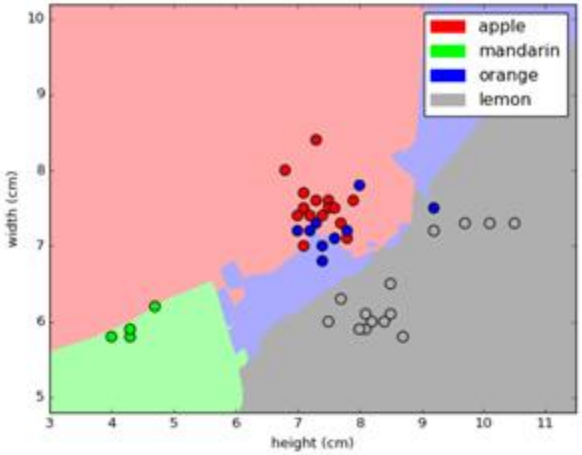
Pixels are colored according to how a test point at that location would be classified



K=1



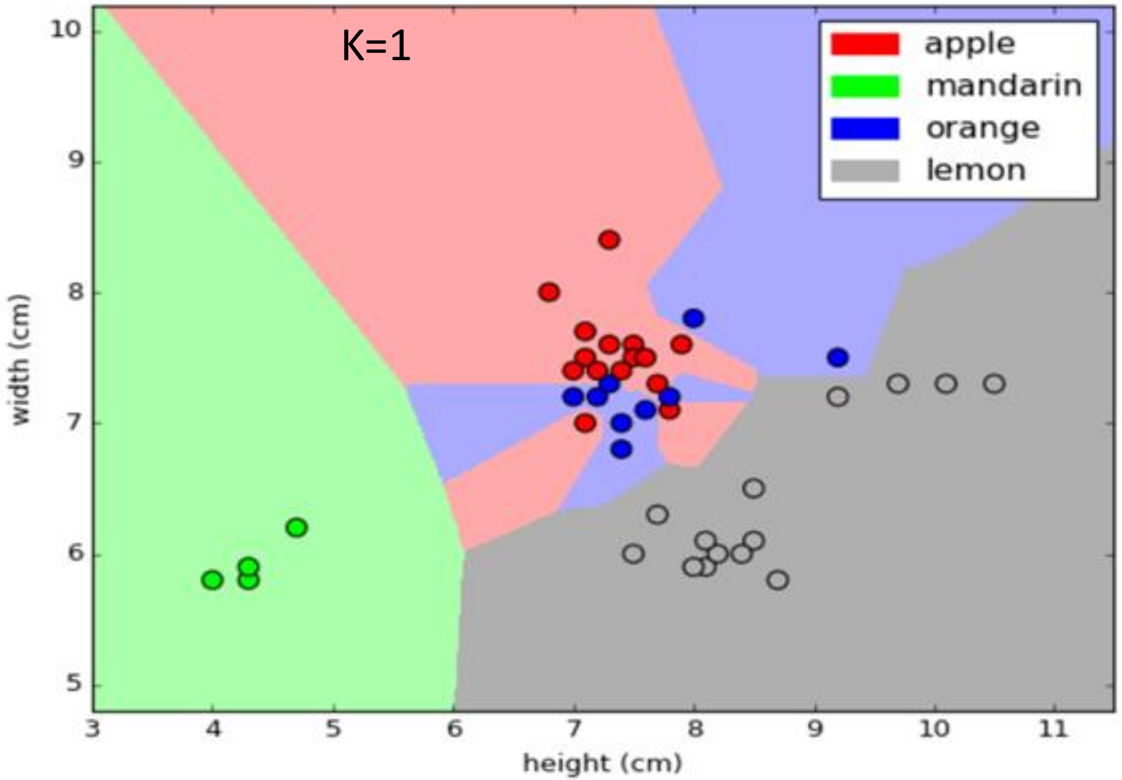
K=5



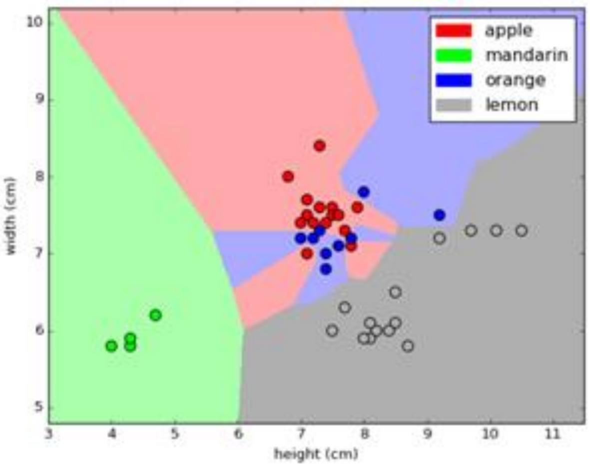
K=10

These were created with the plot_class_regions_for_classifier function in adspy_shared_utilities.py (lab 1)

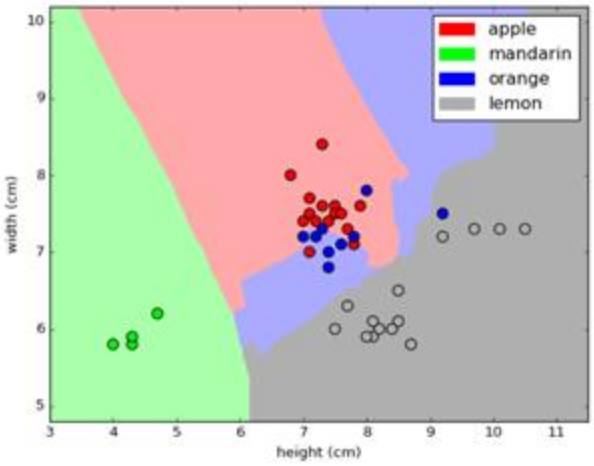
An intuitive definition of k-NN classification



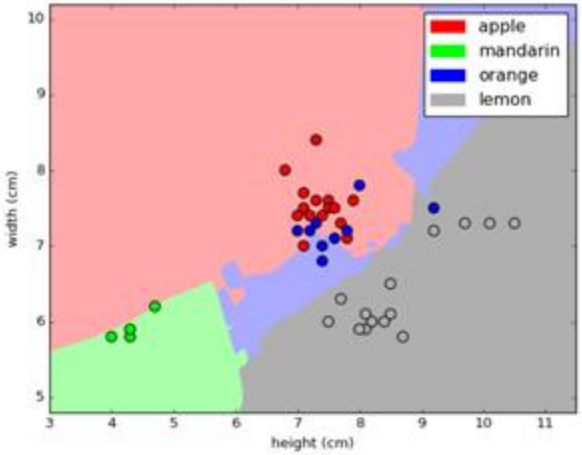
An intuitive definition of k-NN classification



K=1



K=5



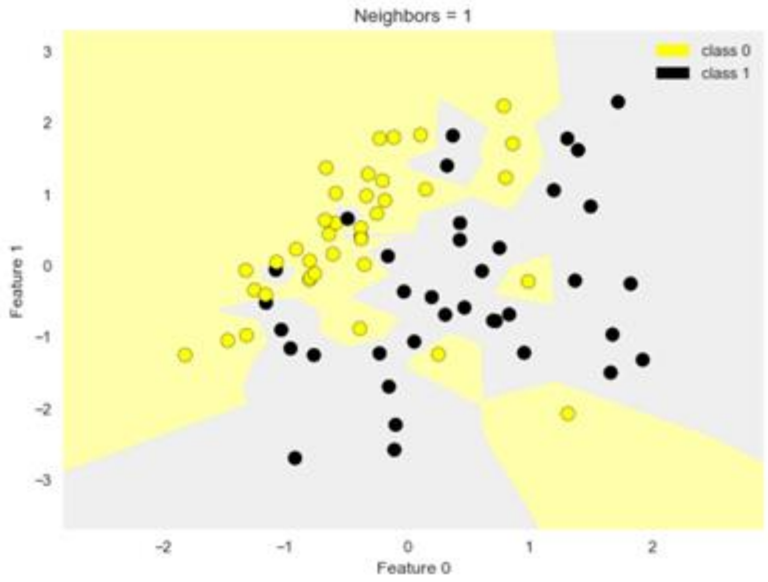
K=10

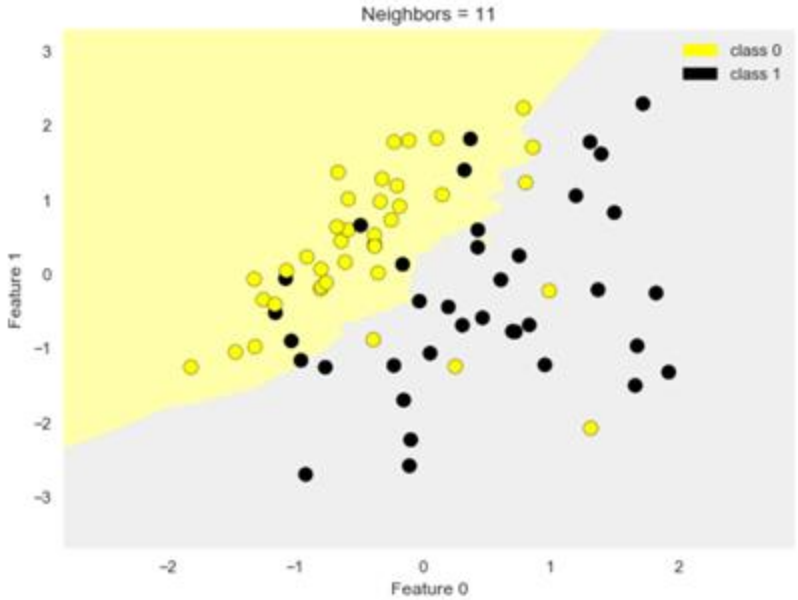
A nearest neighbor algorithm: parameters

1. A distance metric between objects
   Typically Euclidean (Fancy name: ‘Minkowski’ metric with p = 2)
2. How many 'nearest' neighbors to look at?
   e.g. five
3. Optional weighting function on the neighbor points
   Ignored (Could weight neighbor influence inversely by distance from query point)

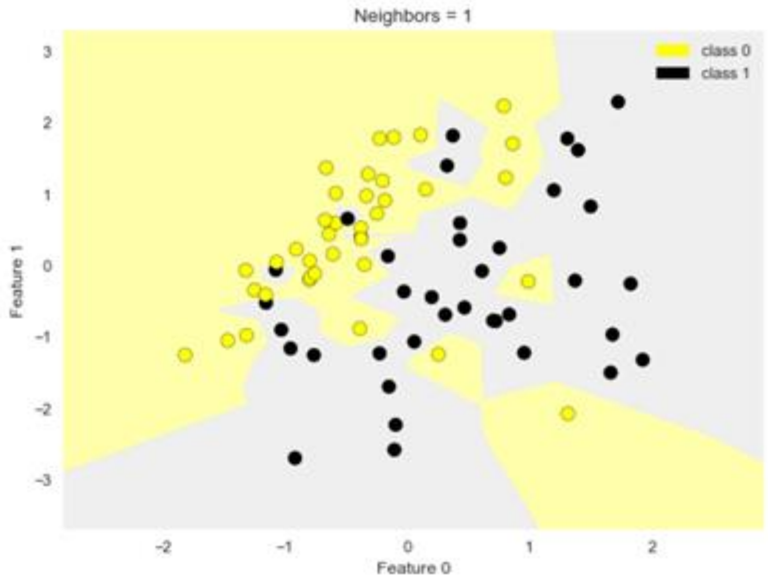
Quiz

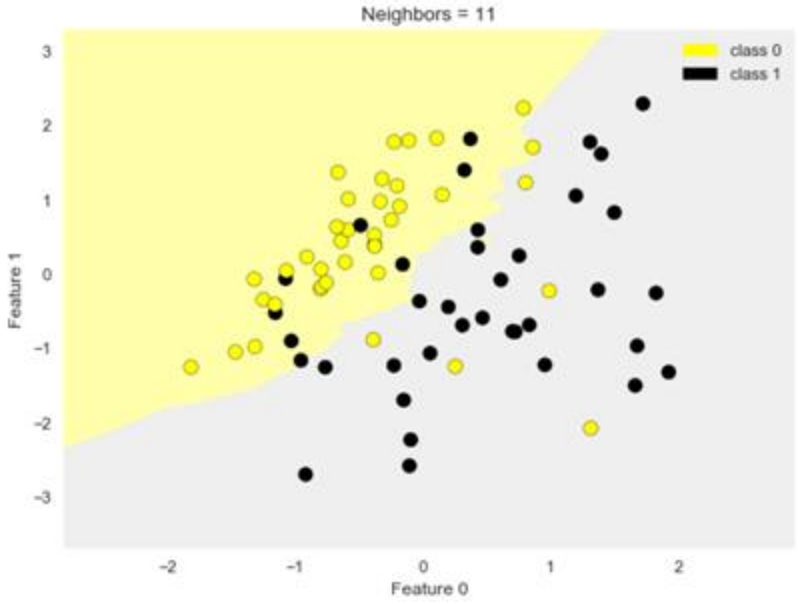
- Which diagram is 1-NN and which is 11-NN?





Nearest Neighbors Classification (k=1, 11)





Evaluation

How can we peek inside k-NN to see what is going on?

Evaluation: accuracy

$$\text{Accuracy} = \frac{\#\text{correct predictions}}{\#\text{total instances}}$$

We can compute:

- Training set accuracy
- Test set accuracy

Using the score function to compute classifier accuracy

```
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test =
 train_test_split(X_C1, y_C1, random_state = 0)

knnc = KNeighborsClassifier(n_neighbors = 5).fit(X_train, y_train)
knnc.predict(X_test)
knnc.score(X_test, y_test)
```

But just accuracy is not enough: we care about **classifier failures**.

A contingency table helps us understand the types of errors a classifier makes.

	Predicted negative	Predicted positive	
True negative	True Negative	False Positive	
True positive	False Negative	True Positive	

Recall, or True Positive Rate (TPR): what fraction of all positive instances does the classifier correctly identify as positive?

	Predicted negative	Predicted positive	
True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
			N = 450

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

$$= \frac{26}{26+17}$$

$$= 0.60$$

Recall is also known as:

- True Positive Rate (TPR)
- Sensitivity
- Probability of detection

Precision: what fraction of positive predictions are correct?

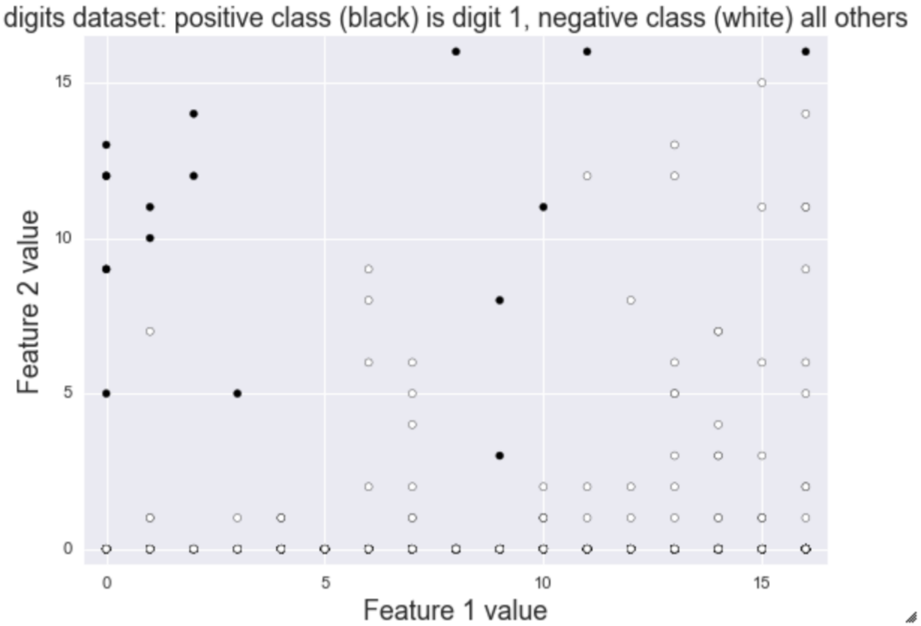
	Predicted negative	Predicted positive	
True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
			$N = 450$

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

$$= \frac{26}{26+7}$$

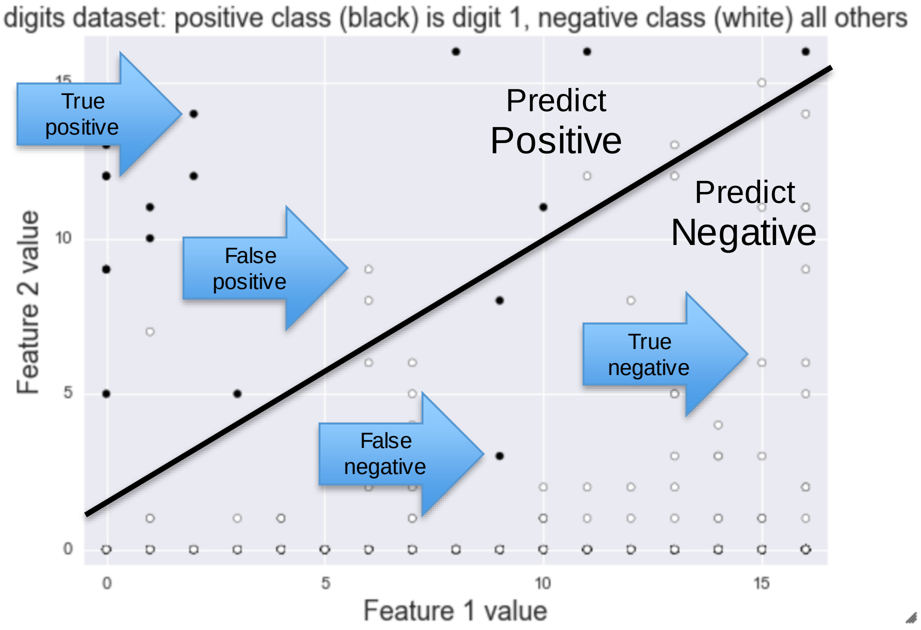
$$= 0.79$$

A Graphical Illustration of Precision & Recall



TN =	FP =
FN =	TP =

The Precision-Recall Tradeoff

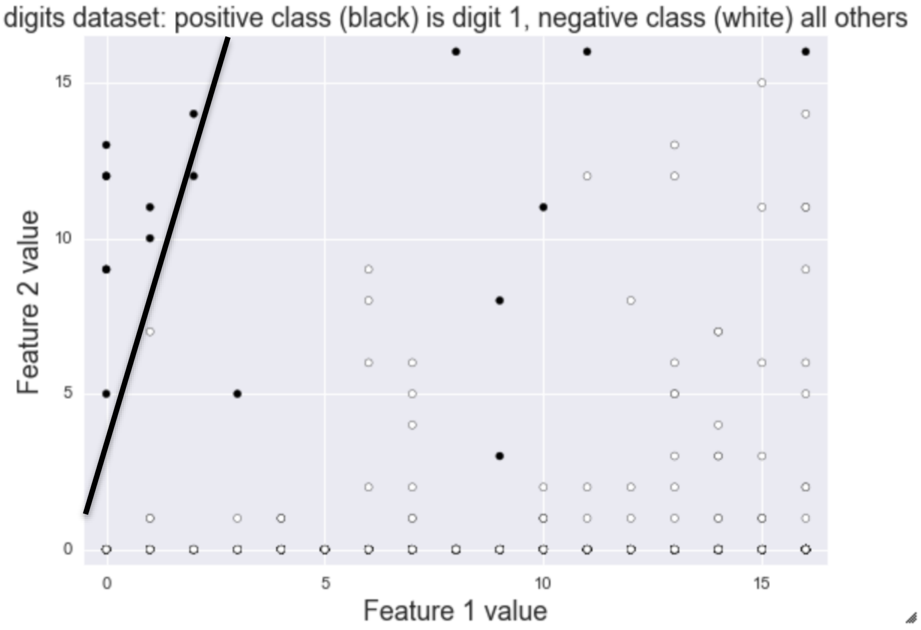


TN = 429	FP = 6
FN = 2	TP = 13

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{13}{19} = 0.68$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{13}{15} = 0.87$$

High Precision, Lower Recall

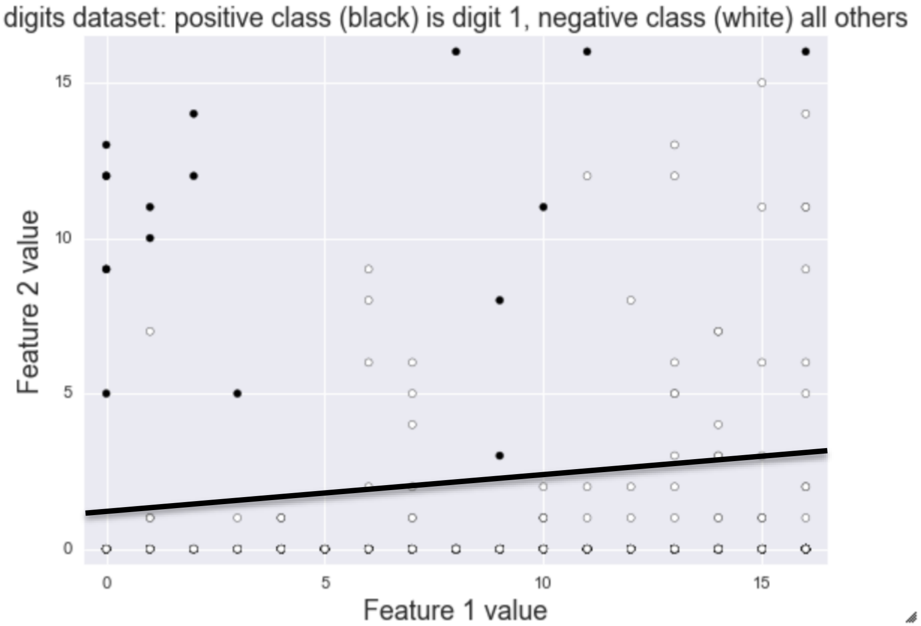


TN = 435	FP = 0
FN = 8	TP = 7

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7} = 1.00$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{7}{15} = 0.47$$

Low Precision, High Recall



TN = 408	FP = 27
FN = 0	TP = 15

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{15}{42} = 0.36$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{15}{15} = 1.00$$

Precision-Recall Curves

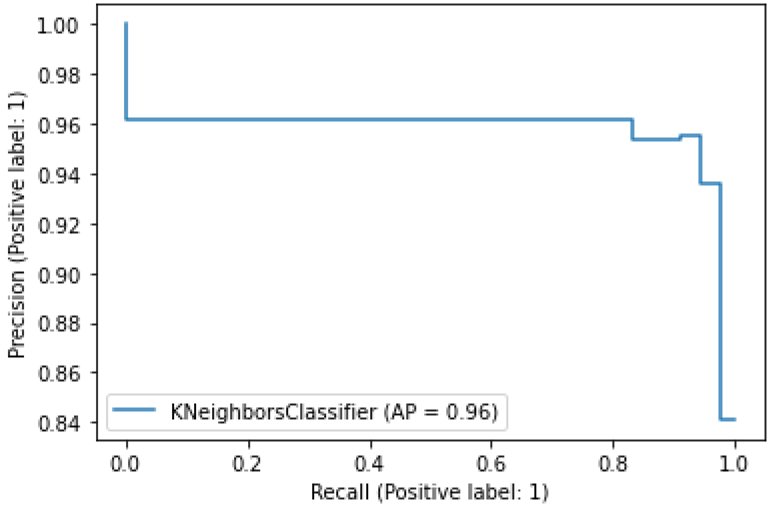
X-axis: Precision
Y-axis: Recall

Top right corner:

- **The "ideal" point**
- **Precision = 1.0**
- **Recall = 1.0**

There is often a tradeoff between precision and recall

- Recall-oriented tasks:
  - Search and information extraction in legal discovery
  - Tumor detection
  - Often paired with a human expert to filter out false positives
- Precision-oriented tasks:
  - Search engine ranking, query suggestion
  - Document classification
  - Many customer-facing tasks (users remember failures!)



How to decide what metric to apply

- Is it more important to avoid false positives, or false negatives?
- Precision is used as a metric when our objective is to minimize false positives
- Recall is used when the objective is to minimize false negatives.

How to decide what metric to apply

- Precision is used as a metric when our objective is to minimize false positives:

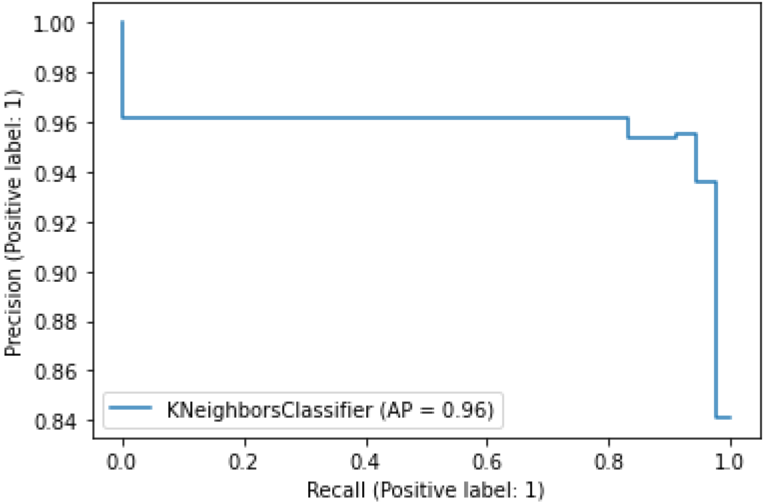
- The police use an algorithm for finding likely criminals in a given area. In this case, a false positive (arresting an innocent person) may be considered more damaging than a false negative (letting a potential criminal walk free).
- This is a precision-oriented task that should minimize false positives.

How to decide what metric to apply

- Recall is used when the objective is to minimize false negatives:

- A law firm is trying to find all emails that mention a certain event (discovery process). Missing even one email could omit valuable evidence. It’s OK to include false positives (emails that are actually not relevant) because we have experts who can filter them out later.
- This is a recall-oriented task that should minimize false negatives.

Cancer detection: what precision / recall tradeoff is best?

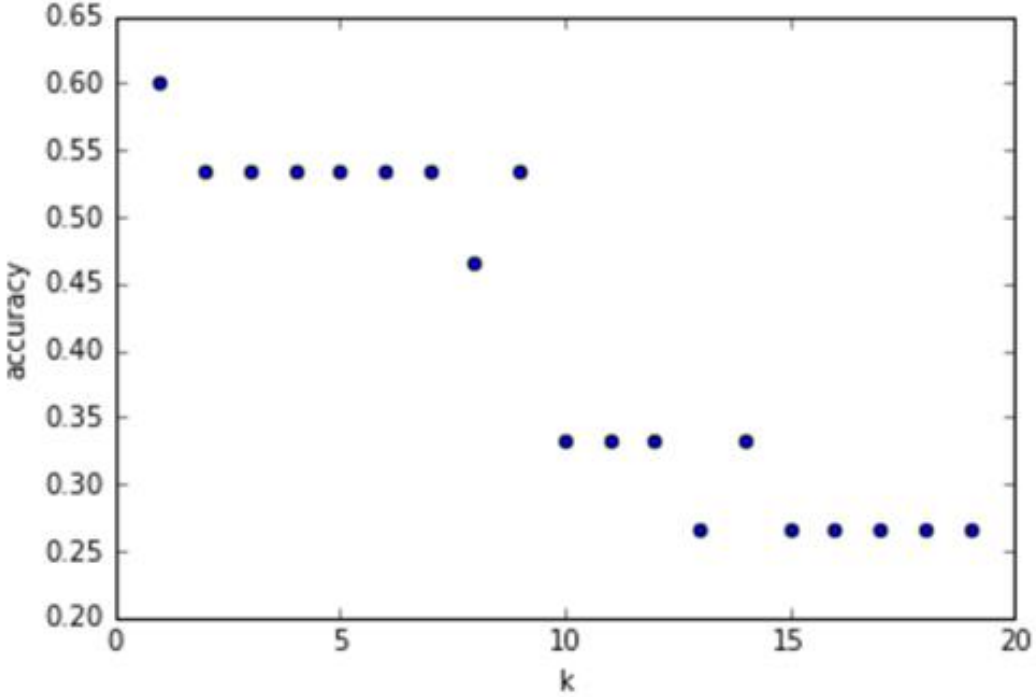


Example Python code to train and evaluate a classifier

1. Import the classifier you want (in this case, k-NN)
2. Split your dataset into training and test sets.
3. Create the classifier object with the correct parameters.
4. Fit the classifier using the training set.
5. Predict the class of each data point in the test set.
6. Score the accuracy of the test set predictions (step 5) against the true

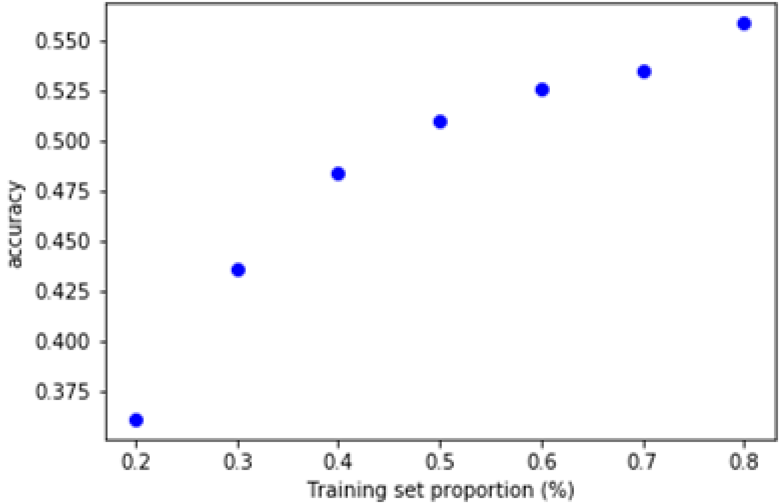
```
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train_true_label, y_test_true_label =
 train_test_split(X_C1, y_C1, random_state = 0)


knnc = KNeighborsClassifier(n_neighbors = 5)

knnc.fit(X_train, y_train_true_label)

y_test_predicted_label = knnc.predict(X_test)

knnc.score(X_test, y_test_true_label)
```

How sensitive is k-NN classifier accuracy to the choice of 'k' parameter?



Fruit dataset with 75%/25% train-test split

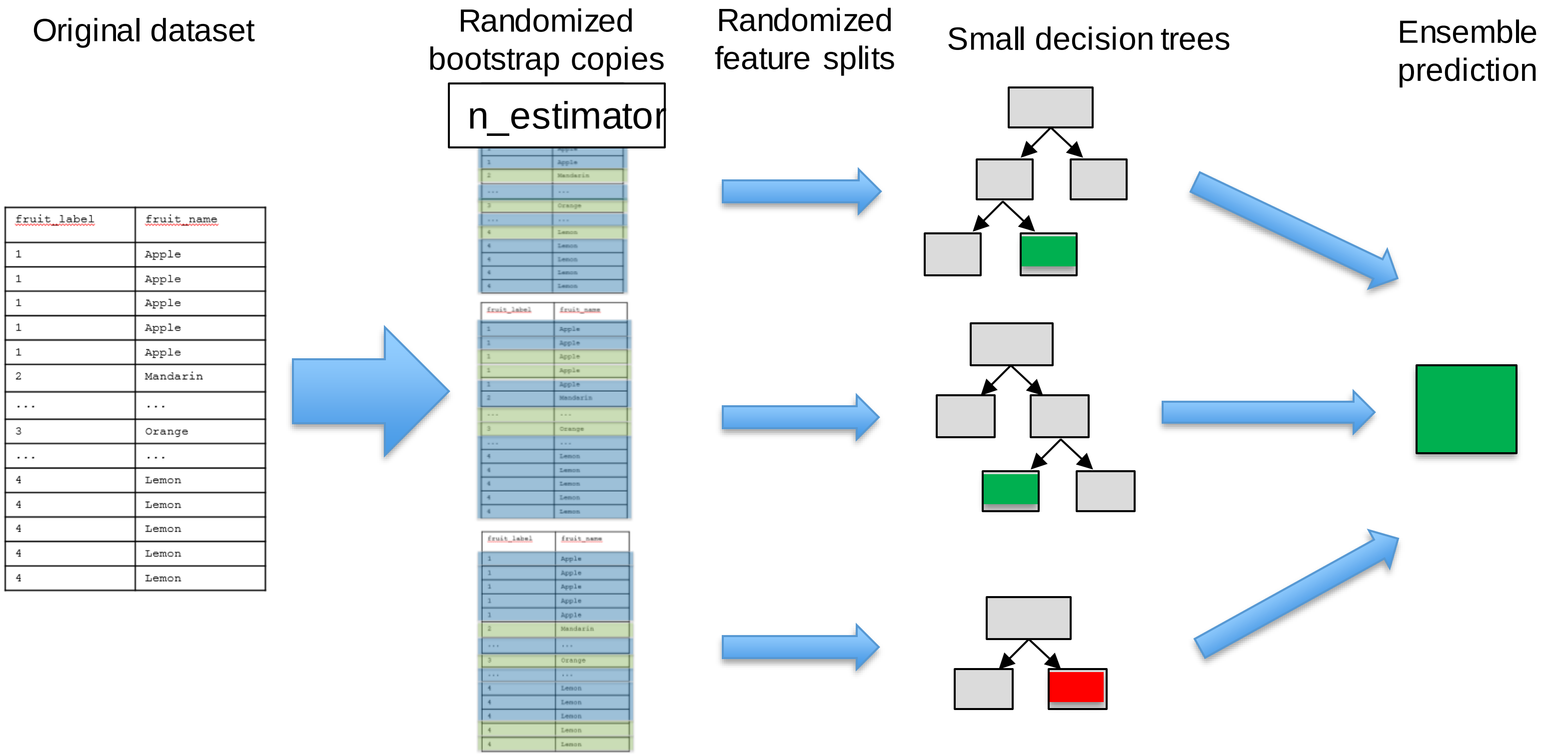
How sensitive is k-NN accuracy to the amount of training data?



But we'll be **suspicious** of very high classifier accuracy on a training set, which may be evidence of overfitting.

Random Forest Classifier

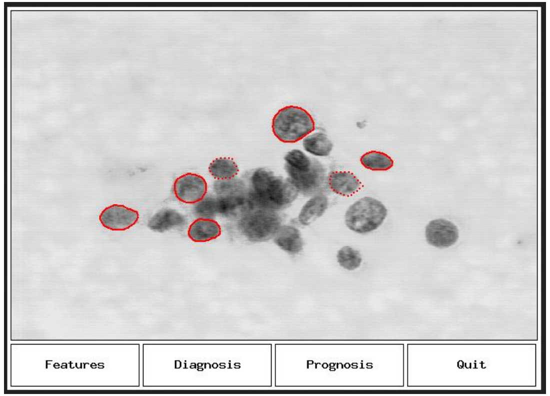
```
from sklearn.ensemble import RandomForestClassifier
```



Wisconsin Breast Cancer dataset

mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.



Two main types of features:

- **Radius** / area / perimeter
- **Texture** / smoothness / symmetry / fractal dimension (a type of smoothness)

Label: B if cell is benign (0), M if cell is malignant (1)

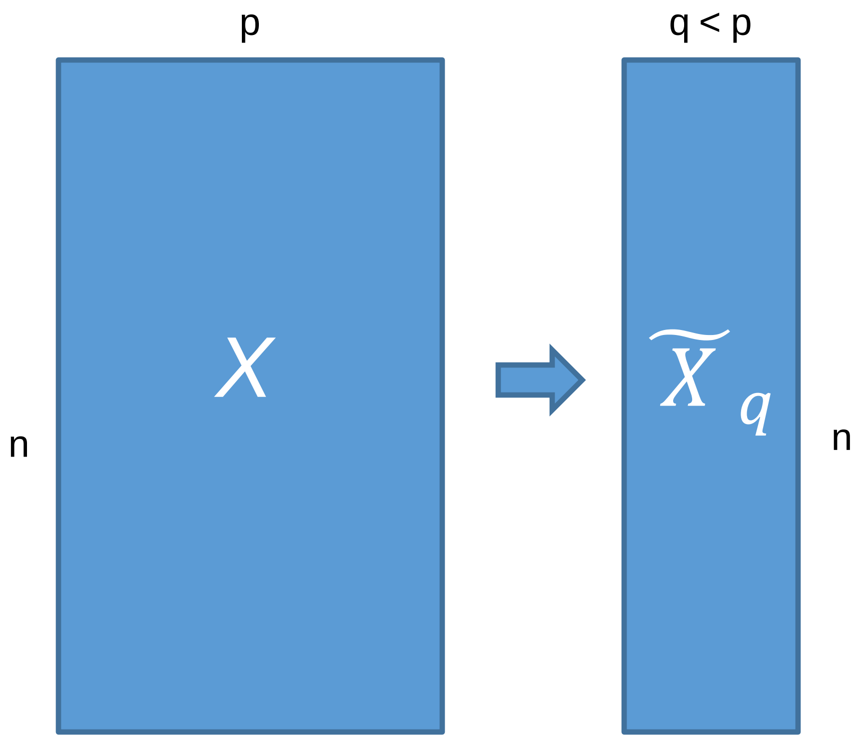
Image source: http://pages.cs.wisc.edu/~street/saves/xcyt1.gif

Supervised learning notebook walk-through

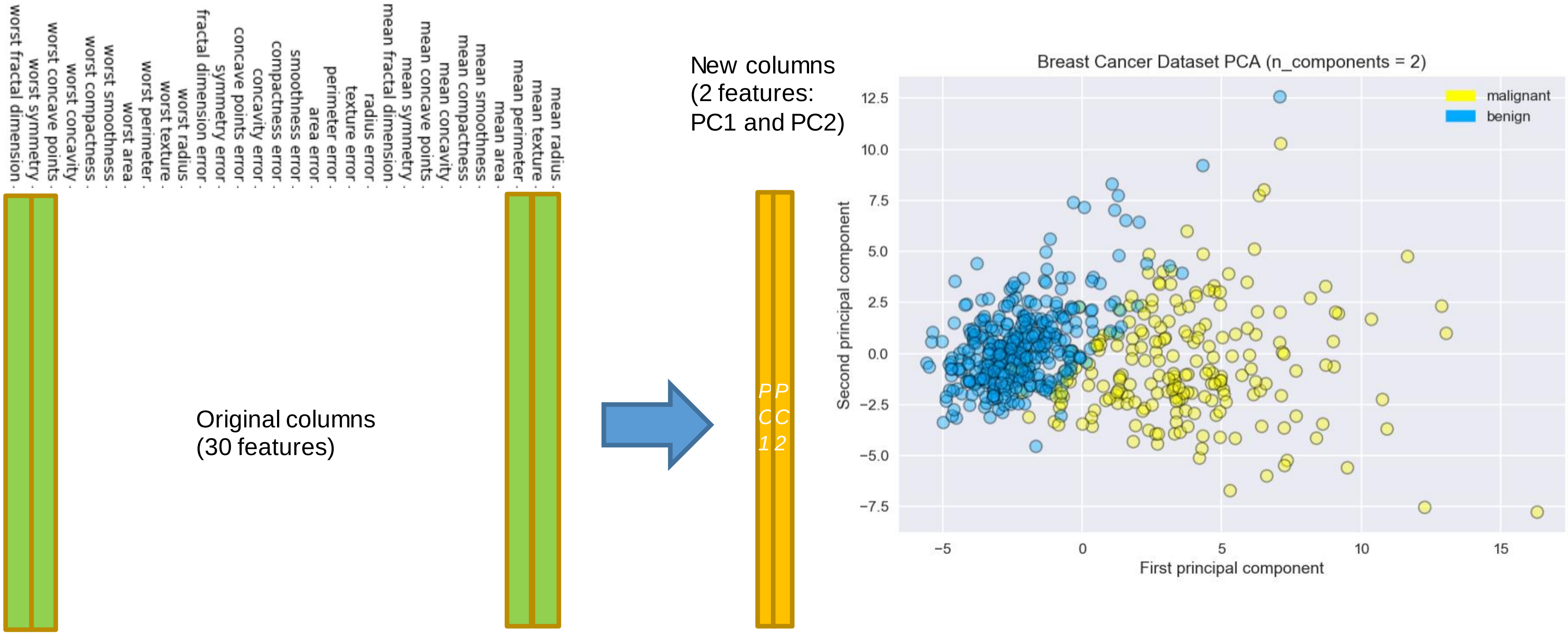
Unsupervised learning: no labels

- Example: dimensionality reduction using PCA

Dimensionality Reduction: Why do it?



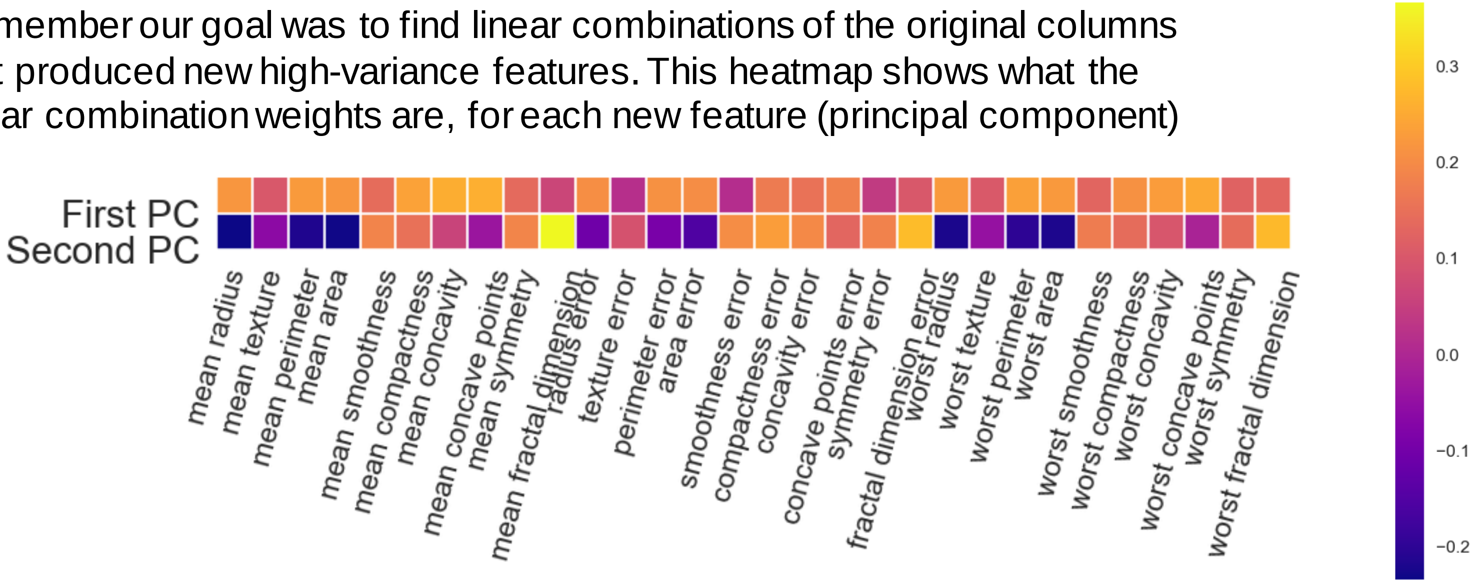
1. Explore a dataset.
2. Speed up machine learning algorithms by working with many fewer features.
3. Compress to save space.
4. Find interesting structure to improve accuracy of supervised learning predictions.




```
from adspy_shared_utilities import plot_labelled_scatter
plot_labelled_scatter(X_pca, y_cancer, ['malignant', 'benign'])

plt.xlabel('First principal component')
plt.ylabel('Second principal component')
plt.title("Breast Cancer Dataset PCA (n_components = 2)")
```

Visualizing PCA Components

Remember our goal was to find linear combinations of the original columns that produced new high-variance features. This heatmap shows what the linear combination weights are, for each new feature (principal component)

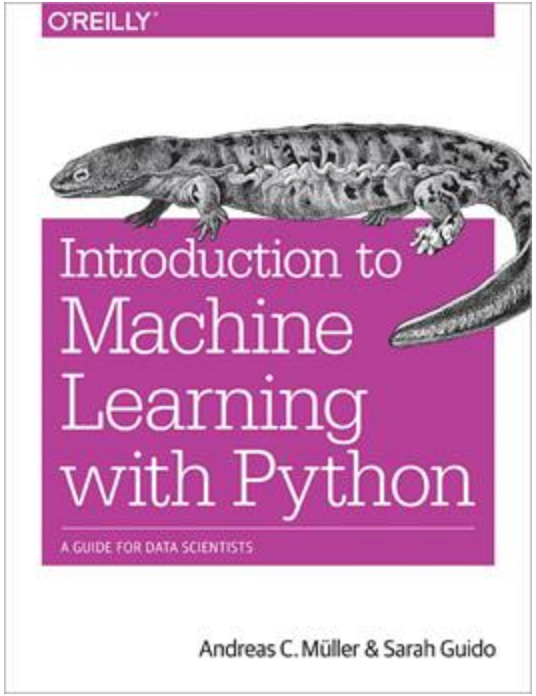


Unsupervised learning task: dimensionality reduction

Background needed for SI 670 Machine Learning

- Python programming
- Ideally some familiarity w/ pandas and NumPy libraries
  - e.g. basic Pandas DataFrame operations.
- Knowledge of basic statistics
  - Probability distributions
  - Bayes rule
- Some familiarity w/ basic vector / matrix operations
  - e.g. dot product of two vectors, how to multiply matrices
- The course includes short ‘refresher’ tutorial on numpy

To learn more… (and to get a head start on SI 670)



Introduction to Machine Learning with Python
A Guide for Data Scientists
By Andreas C. Müller and Sarah Guido

O'Reilly Media

Main textbook for SI 670
Available online for free to UM students: see syllabus.

Library references

scikit-learn: Python Machine Learning Library

- scikit-learn Homepage
  http://scikit-learn.org/
- scikit-learn User Guide
  http://scikit-learn.org/stable/user_guide.html
- scikit-learn API reference
  http://scikit-learn.org/stable/modules/classes.html
- In Python, we typically import classes and functions we need like this:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

SciPy Library: Scientific Computing Tools

http://www.scipy.org

- Provides a variety of useful scientific computing tools, including statistical distributions, optimization of functions, linear algebra, and a variety of specialized mathematical functions.
- With scikit-learn, it provides support for *sparse matrices*, a way to store large tables that consist mostly of zeros.
- Example import: `import scipy as sp`

NumPy: Scientific Computing Library

http://www.numpy.org/

- Provides fundamental data structures used by scikit-learn, particularly multi-dimensional arrays.
- Typically, data that is input to scikit-learn will be in the form of a NumPy array.
- Example import: `import numpy as np`

Pandas: Data Manipulation and Analysis

http://pandas.pydata.org/

- Provides key data structures like `DataFrame`
- Also, support for reading/writing data in different formats
- Example import: `import pandas as pd`

matplotlib and other plotting libraries

matplotlib http://matplotlib.org/

- We typically use matplotlib's **pyplot** module:

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

- We also sometimes use the **seaborn** visualization library (http://seaborn.pydata.org/)

```
import seaborn as sn
```

- And sometimes the **graphviz** plotting library:

```
import graphviz
```