SI 670 Applied Machine Learning

Grant Schoenebeck

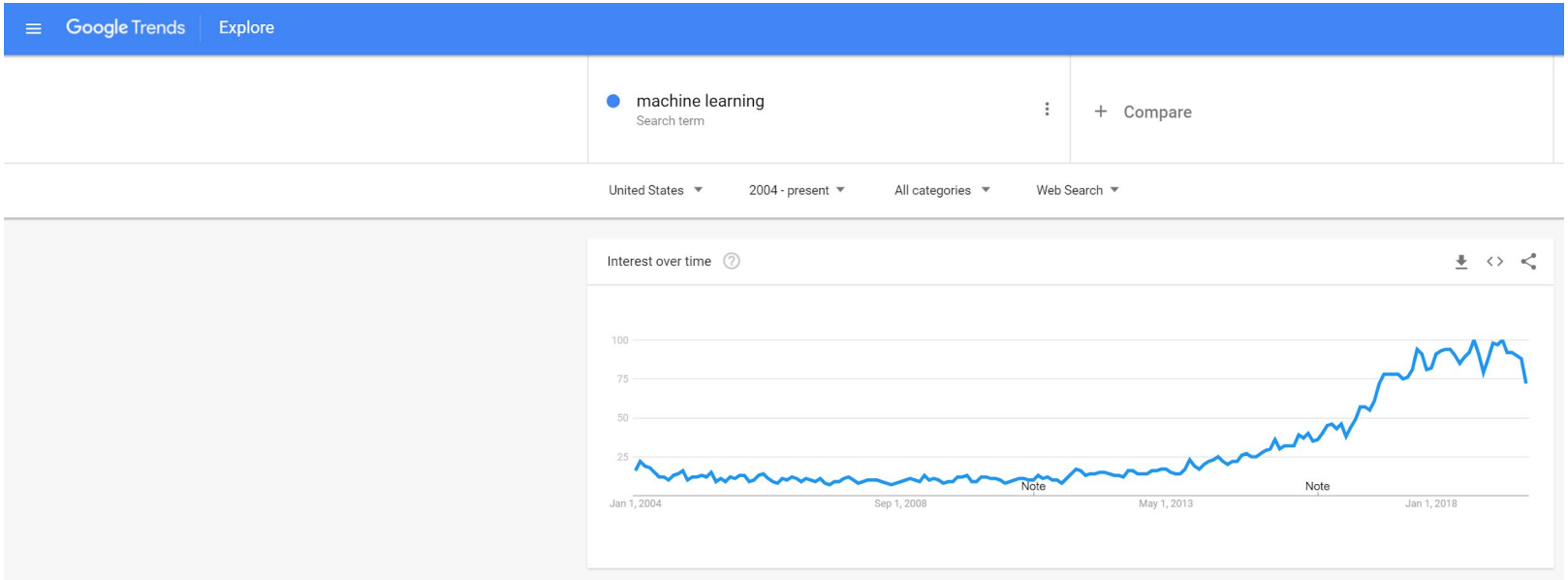
Zoom

- I am excited that there are a lot of you.
  - We cannot all speak at the same time.
- Please raise your hand to speak (under participants).
  - Feel free to interrupt me if it is clear that I have not noticed.
- Feel free to use the chat, the GSIs will monitor it and bring anything to my attention. (I cannot read the chat and engage otherwise at the same time.)
- Please mute yourself when you are not speaking.
- Please leave your video on!
  - We welcome dogs and kids.

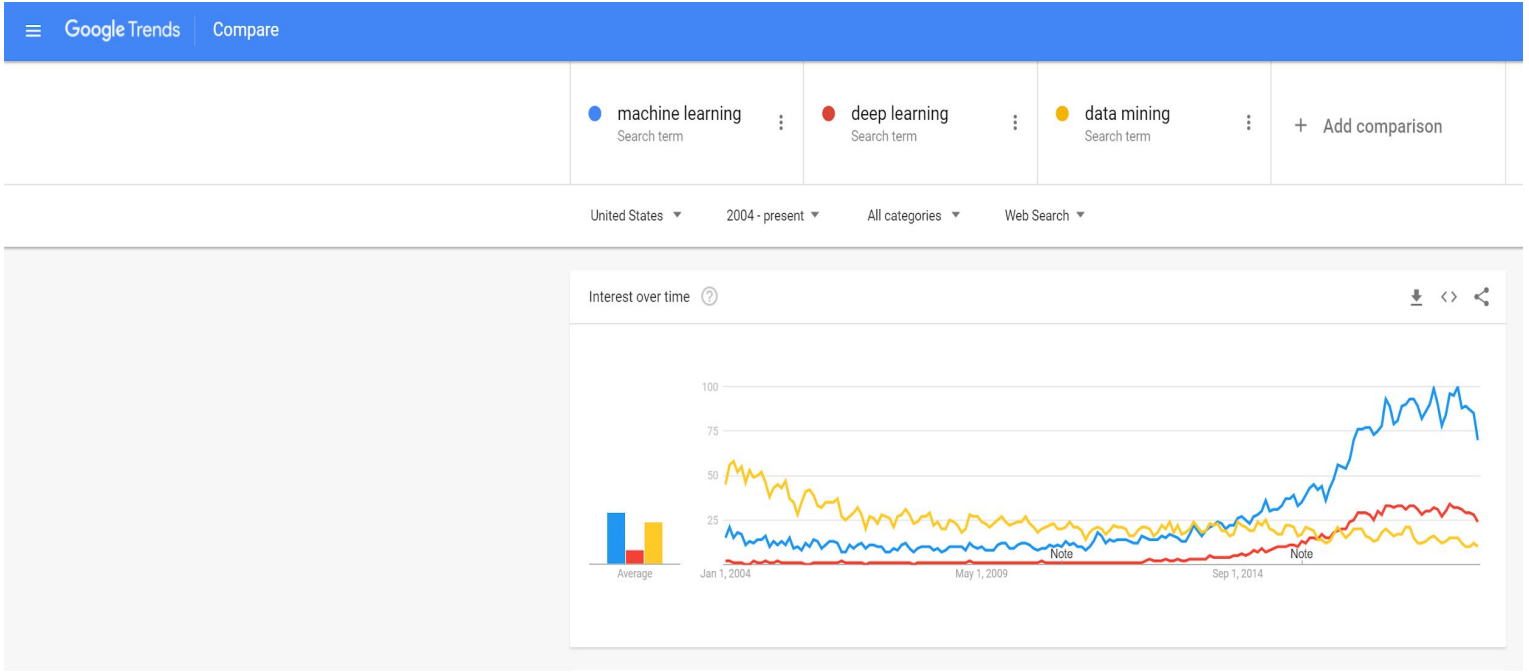
Introduction

What is Machine Learning (ML)?

Popular

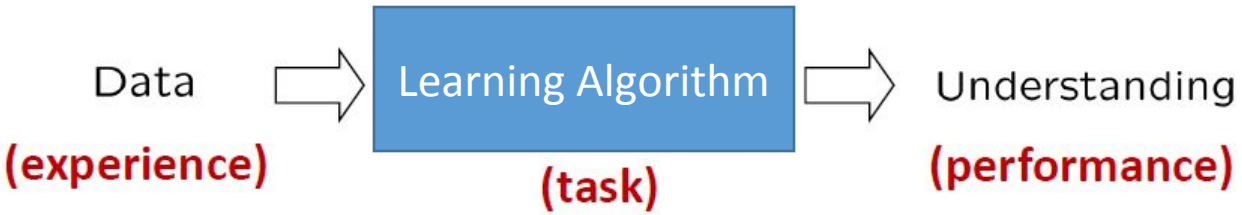

Google Trends
Explore
machine learning
Search term
Compare
United States
2004 - present
All categories
Web Search
Interest over time
100
75
50
25
Note
Note
Jan 1, 2004
Sep 1, 2008
May 1, 2013
Jan 1, 2018

Machine Learning versus Data Mining



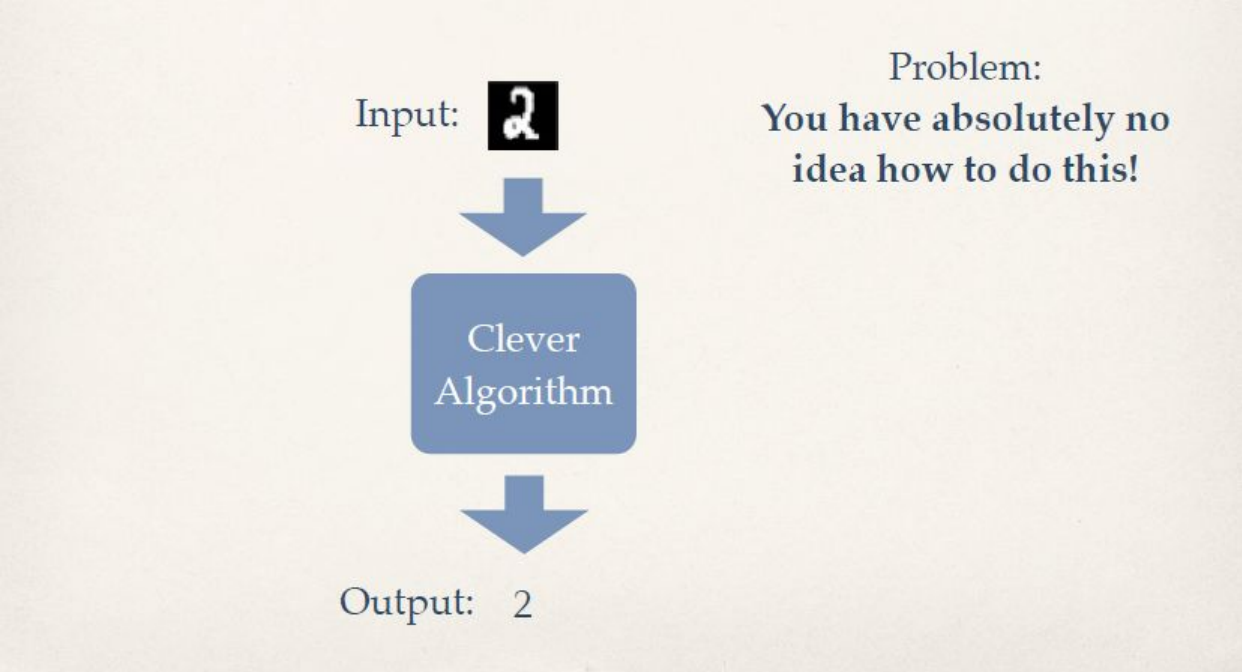
Informal definition

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



Example Definition

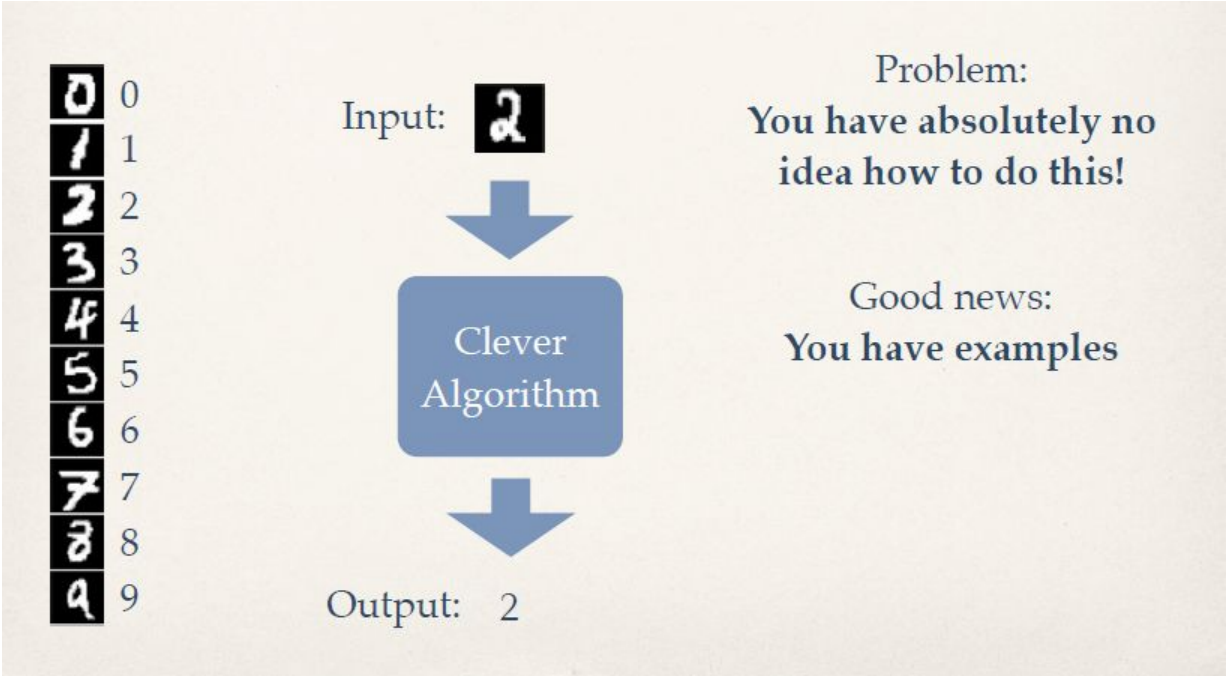
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

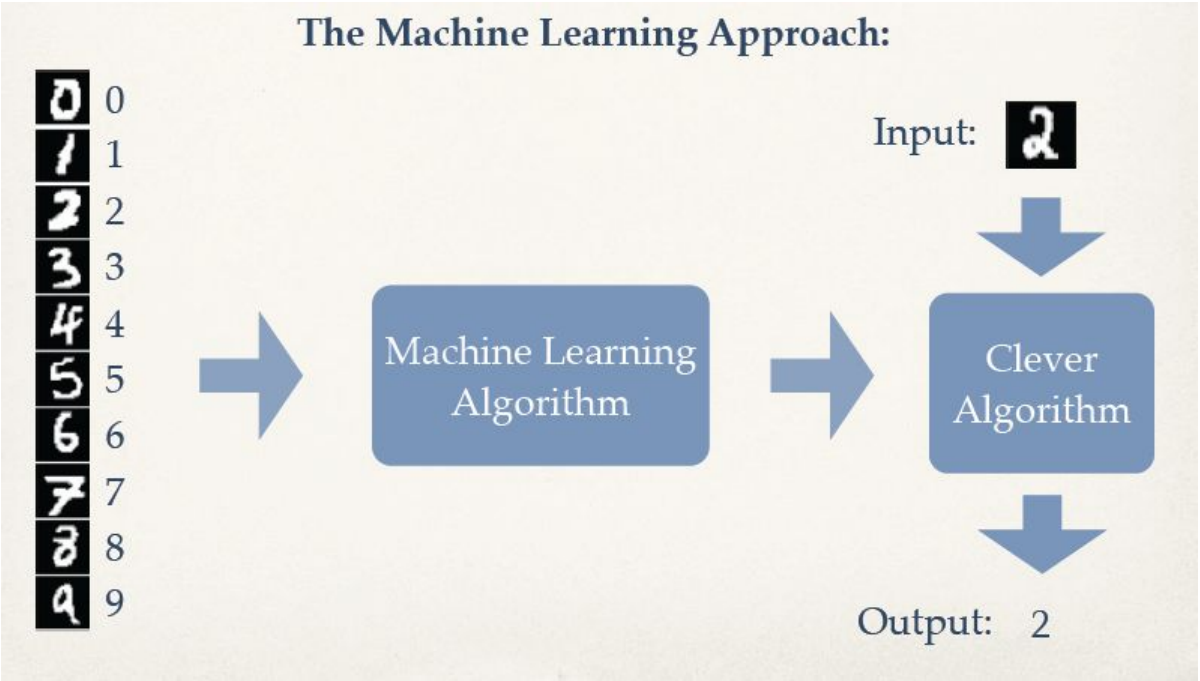
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

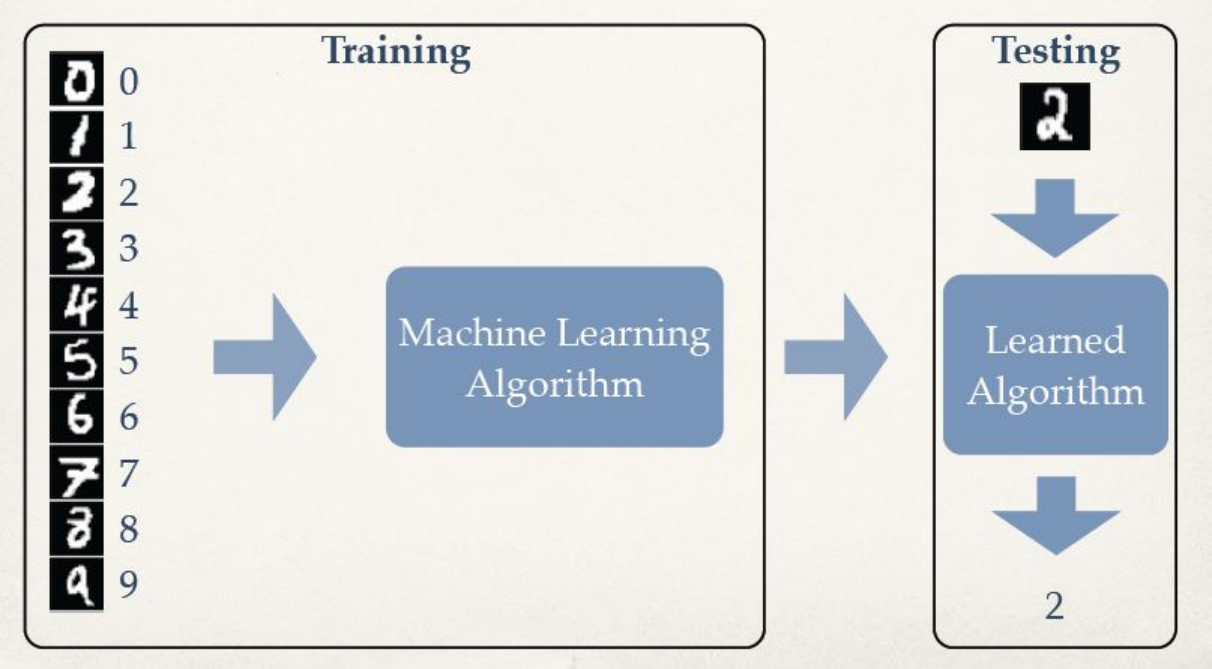
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

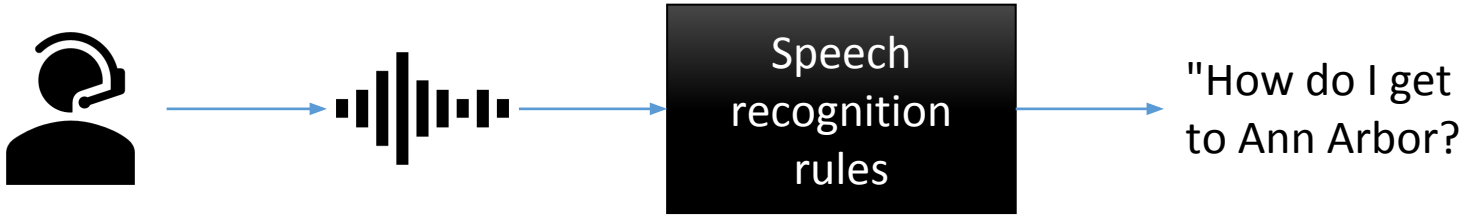
Example

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

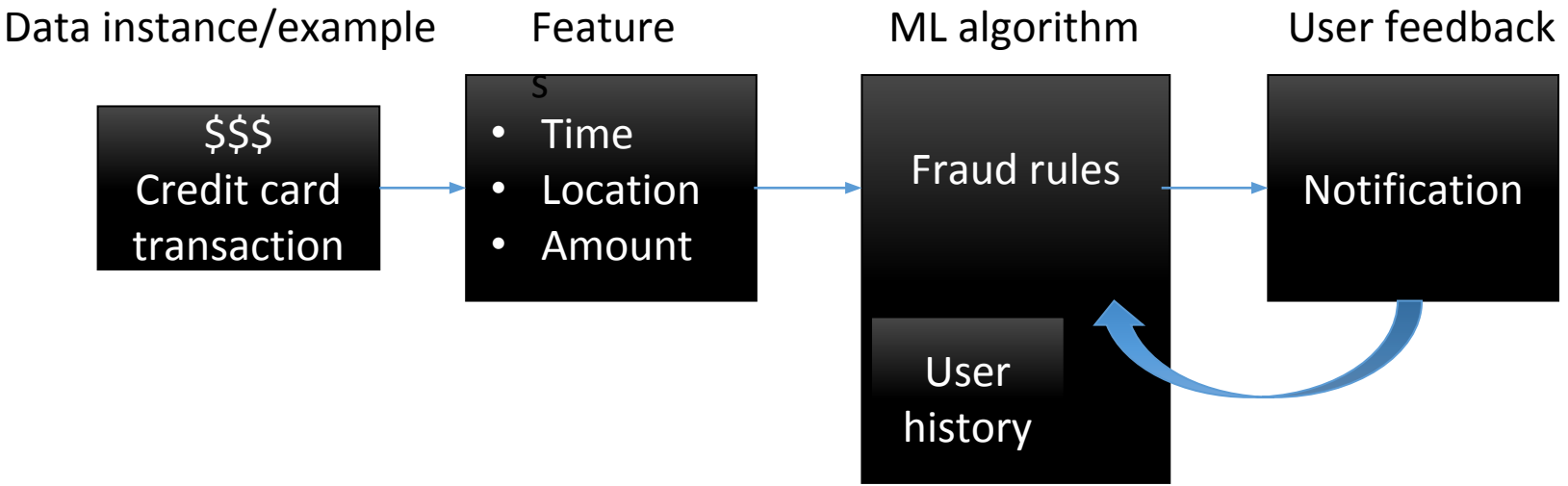


Slide credit: Kilian Weinberger

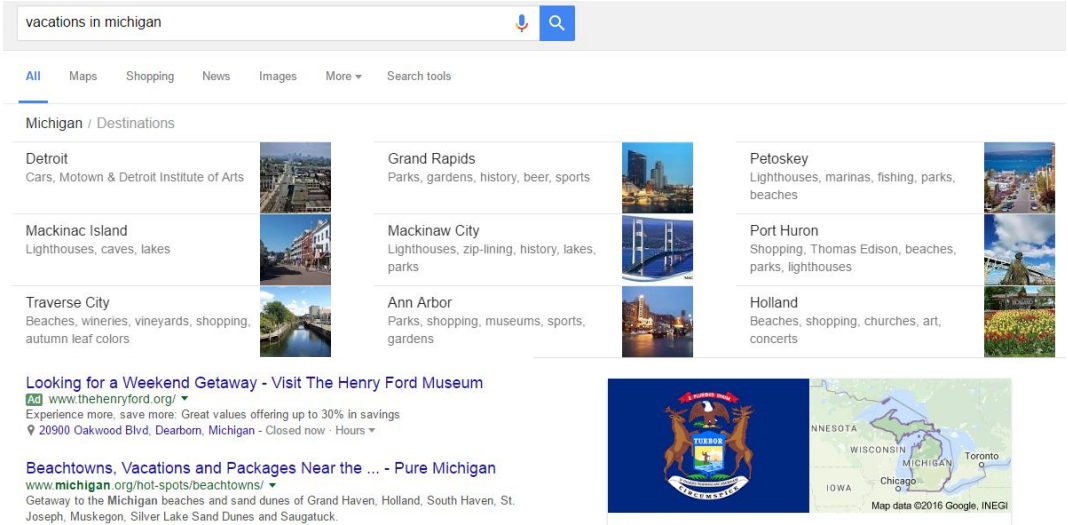
Speech Recognition



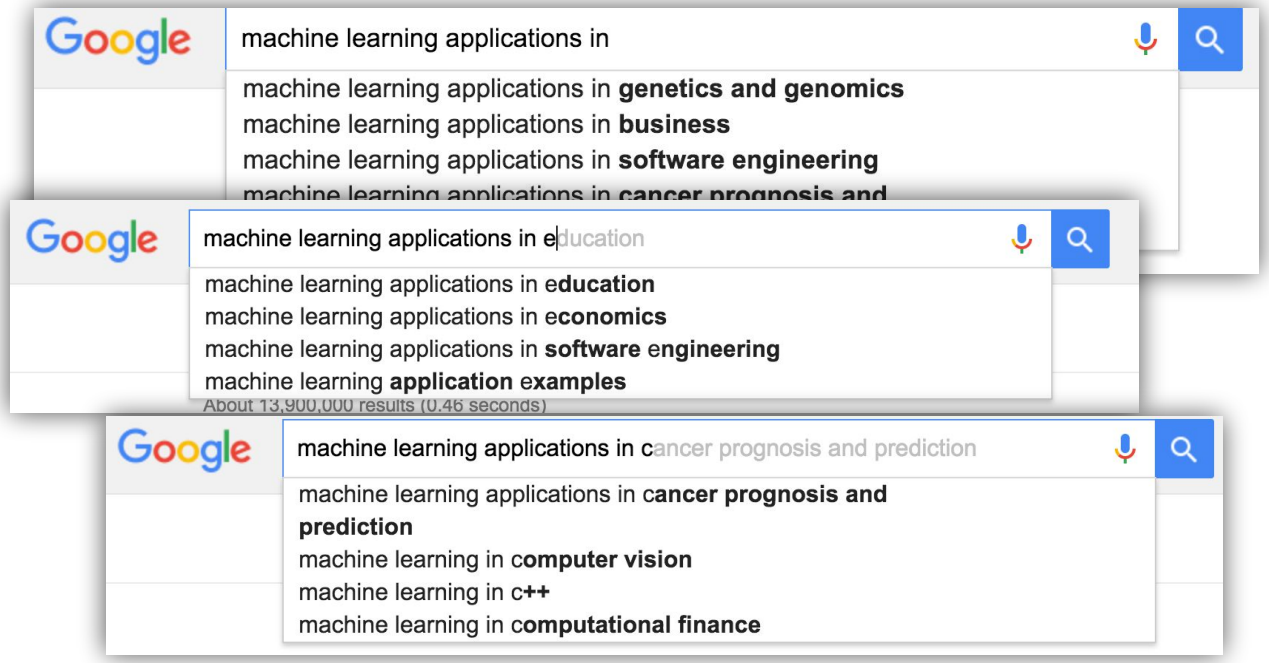
Machine Learning for fraud detection and credit scoring



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



ML □ Lots of Applications



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

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

Syllabus

What is Applied Machine Learning?

- Understand basic ML **concepts** and **workflow**
- How to **properly** apply 'black-box' machine learning components and features

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

Background needed for this course

- Python programming
- Some familiarity w/ pandas and NumPy libraries
  - e.g. basic 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

Course Staff

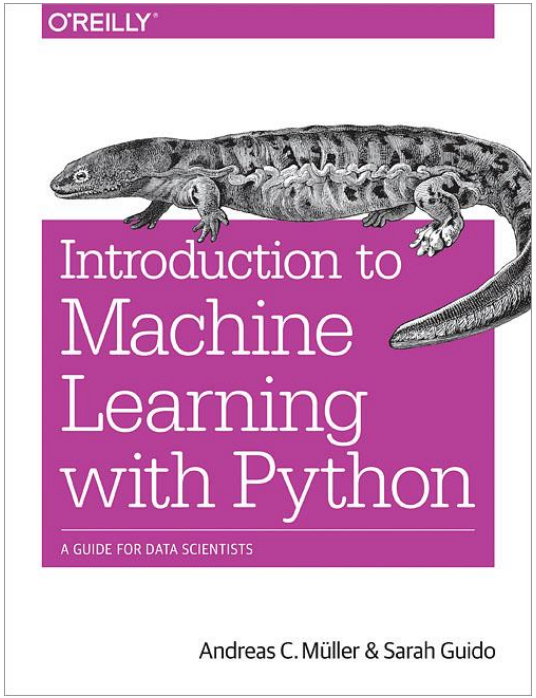
- Grant Schoenebeck
- Teng Ye
- Zhoufeng Wu

- Contacting Us
  - Piazza https://piazza.com/umich/fall2020/si670
  - SI670staff@umich.edu

Course syllabus

- Grade components
  - Homeworks: 45%
  - Kaggle 10%
  - Midterm: 15%
  - Final project: 30%
- Class Format
- Resources
  - Course Staff (email policy)
  - Texts
  - Internet
- Cheating: Don’t
- Class Format
- Questions?

Recommended text for this course

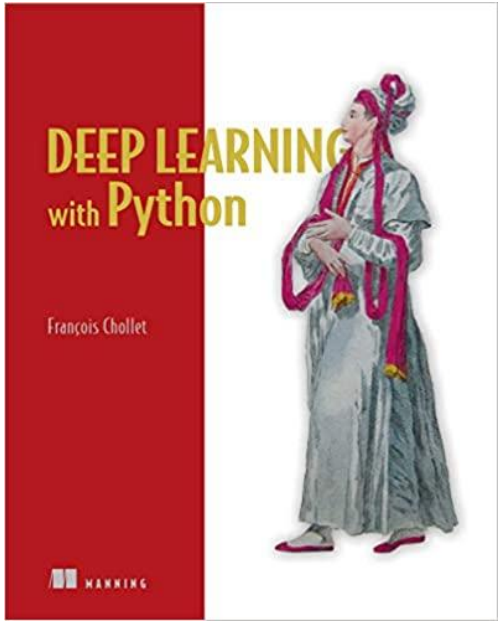


Introduction to Machine Learning with Python
A Guide for Data Scientists
By Andreas C. Müller and Sarah Guido

O'Reilly Media

Available online for free to UM students: see syllabus.

Recommended text for this course



Deep Learning with Python

by Francois Chollet
Manning Publications

Available online for free to UM students:
see syllabus.

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:

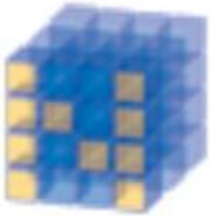
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

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

http://matplotlib.org/

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

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

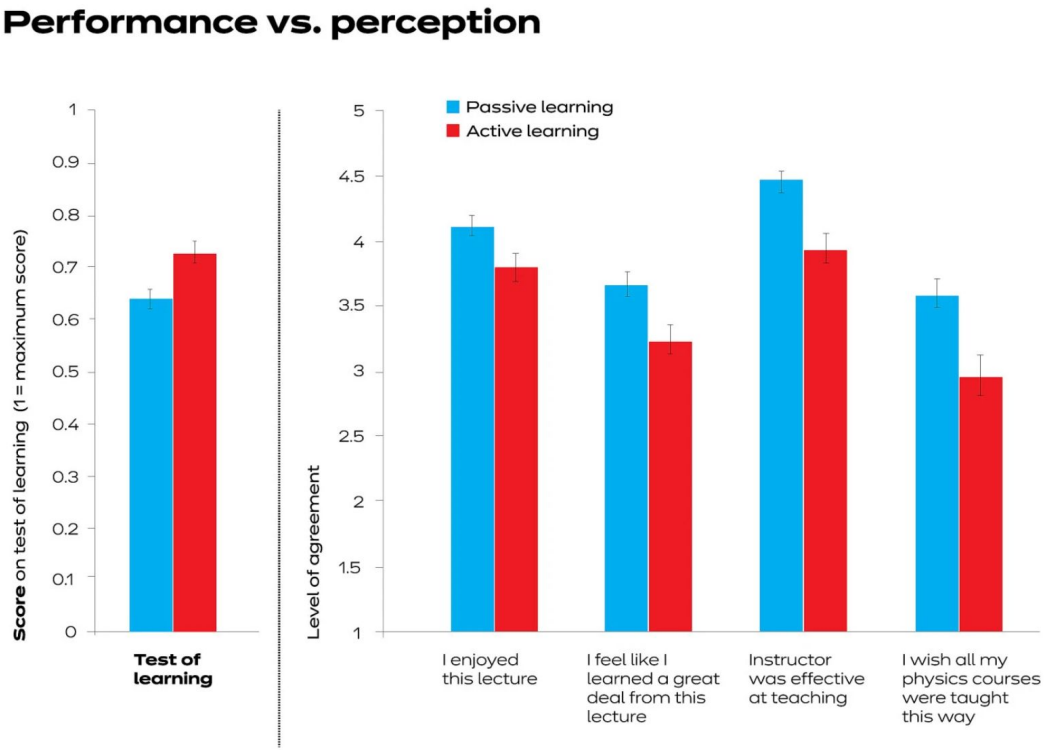
- We also sometimes use the **seaborn** visualization library (http://seaborn.pydata.org/)

```
import seaborn as sn
```

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

```
import graphviz
```

Pedagogy


Performance vs. perception
Score on test of learning (1 = maximum score)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Test of learning
Passive learning
Active learning
Level of agreement
5
4.5
4
3.5
3
2.5
2
1.5
1
I enjoyed this lecture
I feel like I learned a great deal from this lecture
Instructor was effective at teaching
I wish all my physics courses were taught this way

Course syllabus

- Do Help
- Don’t Cheat

Participation Pledge

- Designed to Help You!

Course syllabus

- Class Format

Course syllabus

- Questions?

Course syllabus

- Quiz!

Is it permitted to work with a small group of students on your problem sets if they come up with all the answers and tell you?

- A) Yes
- B) No

Types of ML

Key types of Machine Learning problems

Supervised machine learning: Learn to predict target values from labelled data.

Unsupervised machine learning: Find structure in *unlabeled data*

Reinforcement machine learning: take actions in an environment to maximize cumulative reward

Supervised Learning (classification example)

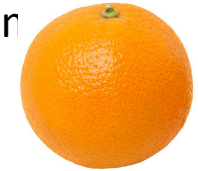
Training set

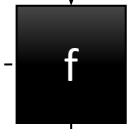
X Sample	Y Target Value (Label)
 x_1	Apple y_1
 x_2	Lemon y_2
 x_3	Apple y_3
 x_4	Orange y_4

Classifier
$f : X \rightarrow Y$

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

Future
sar



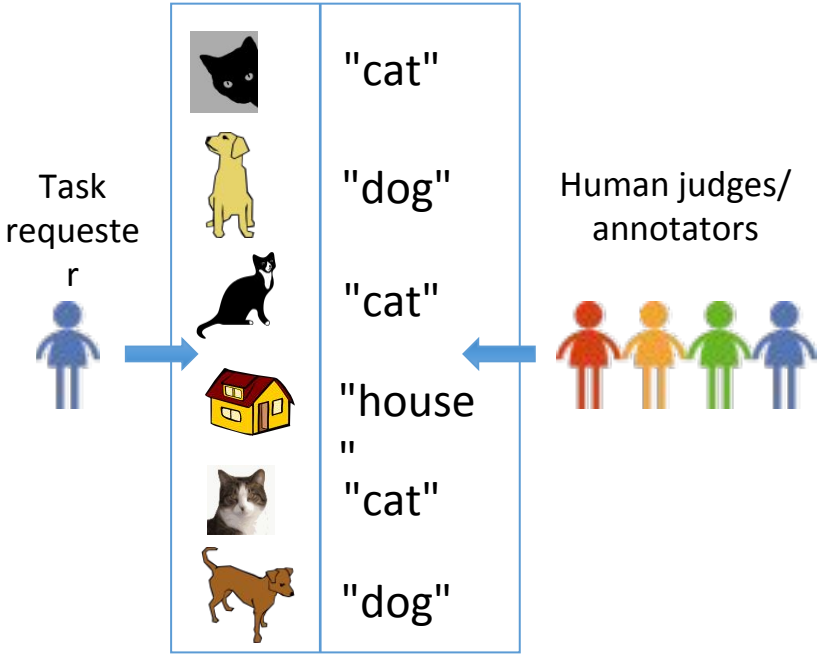


Label: Orange

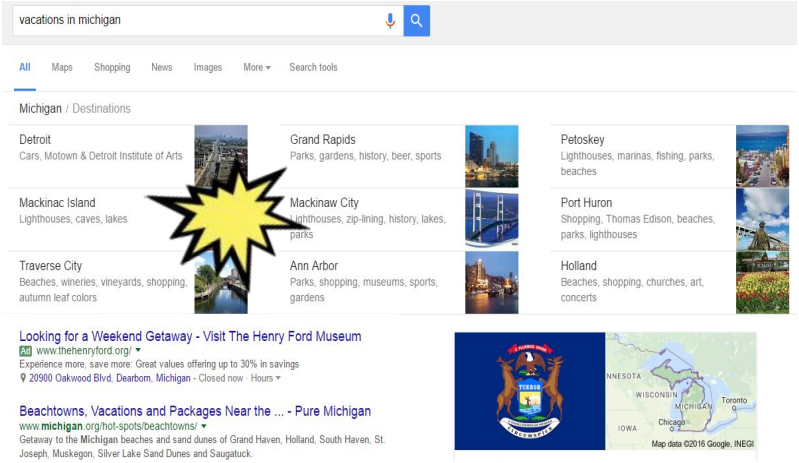
After training, at prediction time, the trained model is used to predict the fruit type for new instances using the learned rules.

Examples of explicit and implicit label sources

Explicit labels



Implicit labels

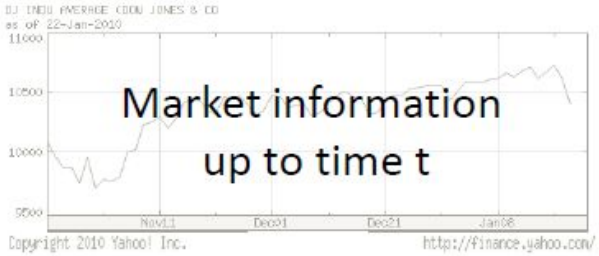


Clicking and reading the "Mackinac Island" result can be an implicit label for the search engine to learn that "Mackinac Island" is especially relevant for the query `[vacations in michigan]` for that specific user.

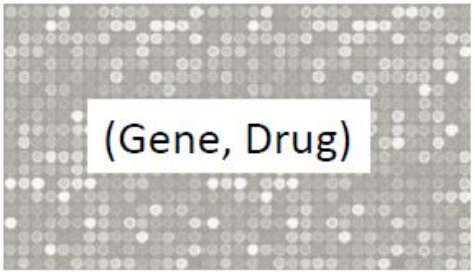
Supervised Learning - Regression

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$



Share Price
"$ 24.50"



Expression level
"0.01"

Continuous Labels

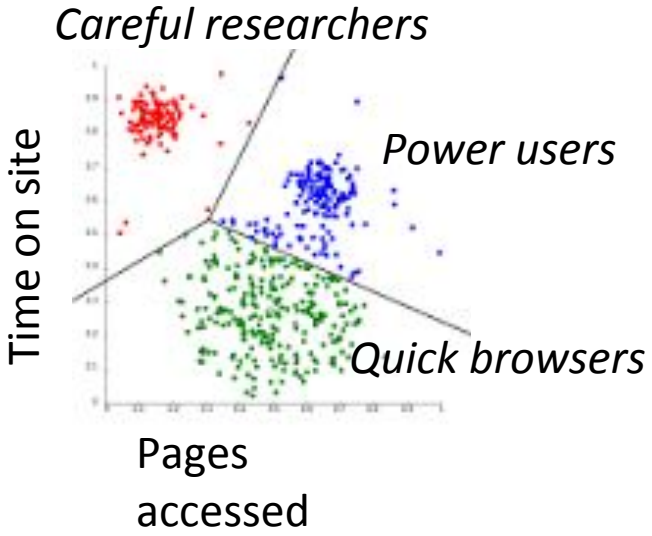
Slide credit: Aarti Singh

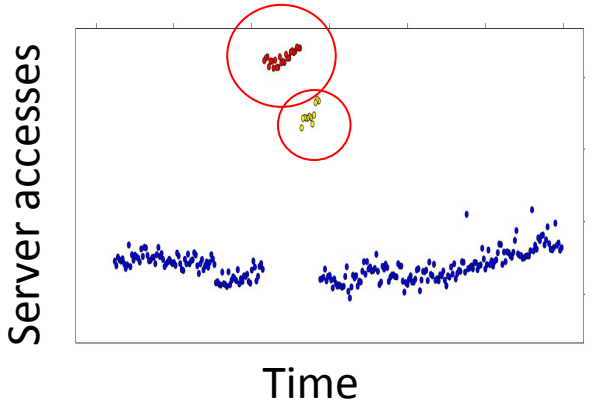
Unsupervised Learning

- Goal:
  - Given data X without any labels
  - Learn the **structures** of the data
    - Clustering
    - Probability distribution (density)
    - Embedding & neighborhood relations
    - Outlier detection
- "Learning without teacher"

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

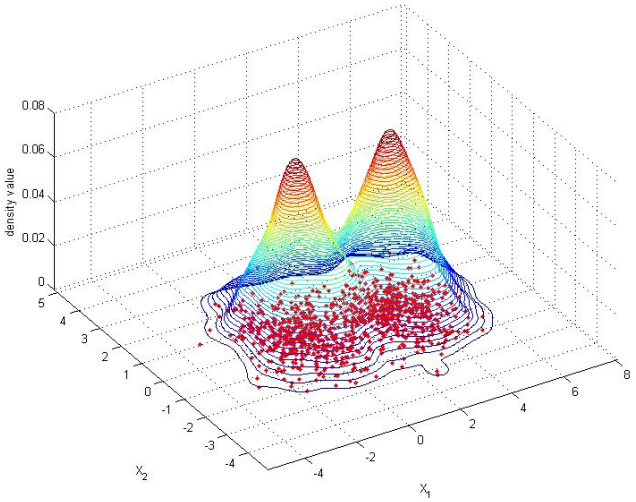
- Clustering
- Detecting abnormal server access patterns (unsupervised outlier detection)

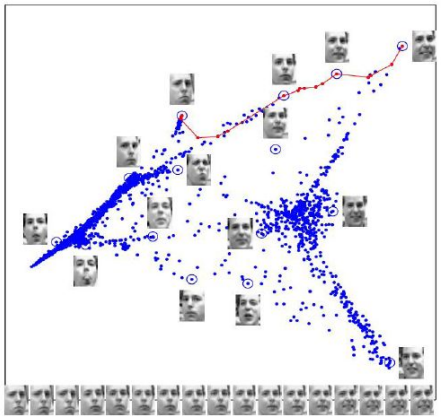




Unsupervised learning: finding useful structure or knowledge in data when no labels are available

- 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 this course

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

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?

Cycle of Improvement

Represent / Train / Evaluate / Refine Cycle

Representation: Extract and select object features

Feature Representations

Feature representation

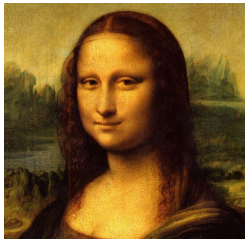
Email

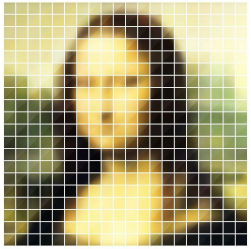
```
To:  Chris Brooks
From:  Daniel Romero
Subject:  Next course offering
Hi Daniel,
Could you please send the outline for
the
next course offering?  Thanks! -- Chris
```

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





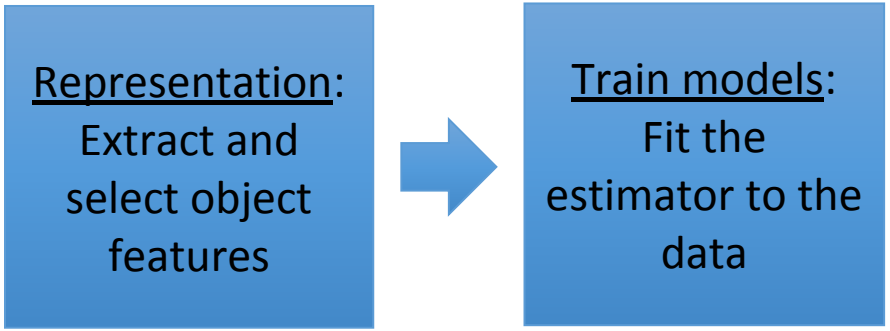
A matrix of color values (pixels)

Sea Creatures

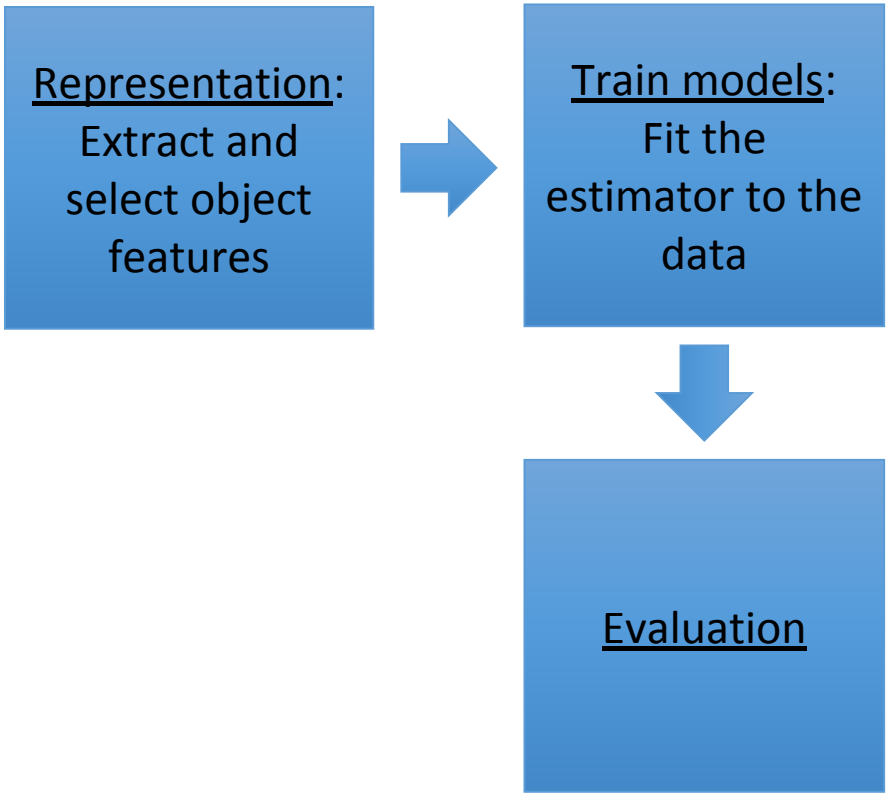
Feature	Value
DorsalFin	Yes
MainColor	Orange
Stripes	Yes
StripeColor1	White
StripeColor2	Black
Length	4.3 cm

A set of attribute values

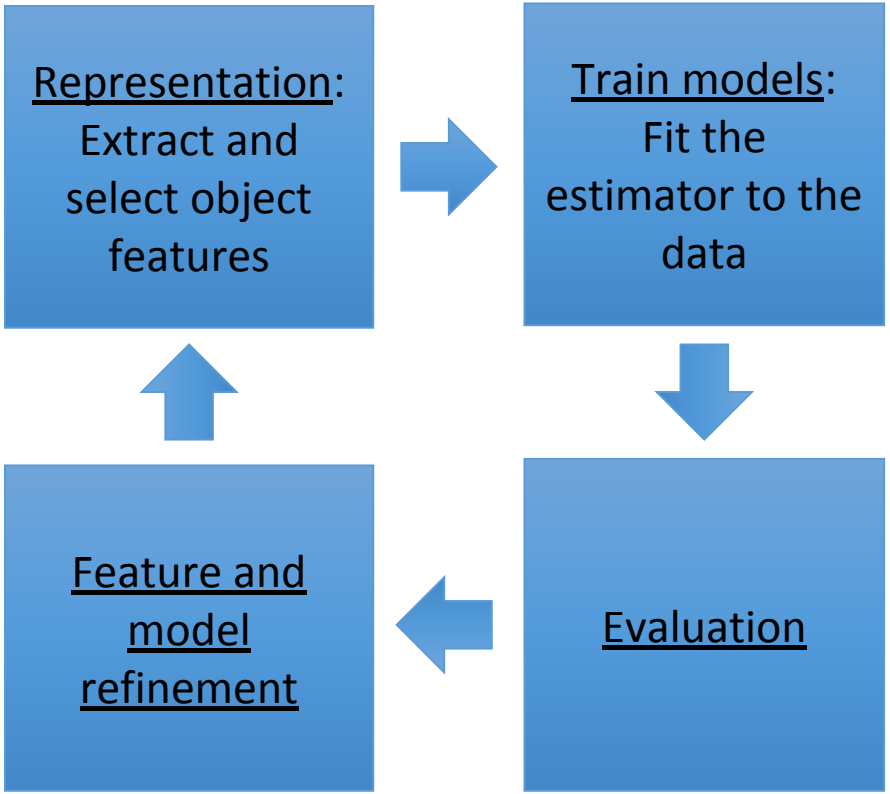
Represent / Train / Evaluate / Refine Cycle



Represent / Train / Evaluate / Refine Cycle



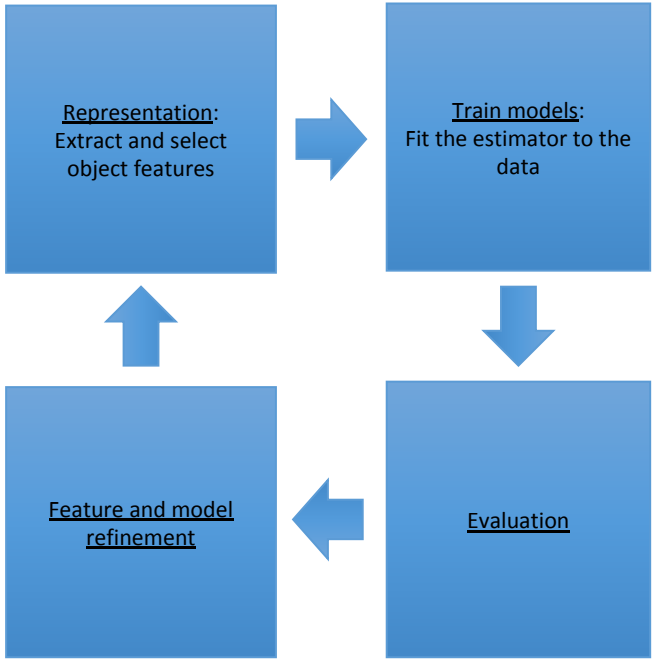
Represent / Train / Evaluate / Refine Cycle



Algorithm Knowledge

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

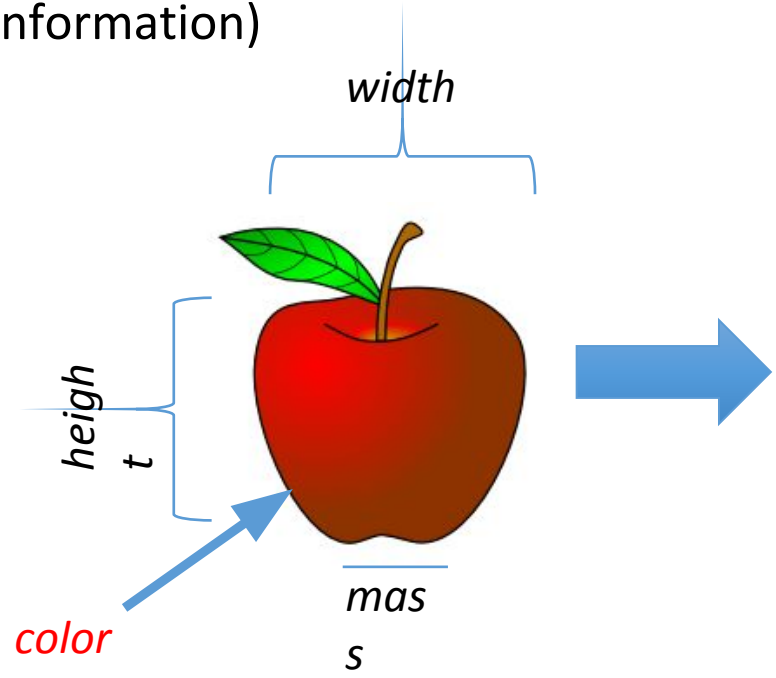
- THINK!



Work flow 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

X

	height	width	mass	color_score
0	7.3	8.4	192	0.55
1	6.8	8.0	180	0.59
2	7.2	7.4	176	0.60
3	4.7	6.2	86	0.80
4	4.6	6.0	84	0.79
5	4.3	5.8	80	0.77
6	4.3	5.9	80	0.81
7	4.0	5.8	76	0.81
8	7.8	7.1	178	0.92
9	7.0			
10	7.3			
11	7.6			
12	7.1			
13	7.7			
14	7.3	7.6	152	0.69
15	7.1	7.7	156	0.69
16	7.5	7.6	156	0.67
17	7.6	7.5	168	0.73
18	7.1	7.5	162	0.83
19	7.2	7.4	162	0.85

y

0	1
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	1
9	
10	
11	
12	
13	
14	1
15	1
16	1
17	1
18	1
19	1

Original data set

```
X_train, X_test, y_train, y_test
    = train_test_split(X, y)
```

X_train

	height	width	mass	color_score
42	7.2	7.2	154	0.82
48	10.1	7.3	174	0.72
7	4.0	5.8	76	0.81
14	7.3	7.6	152	0.69
32	7.0	7.2	164	0.80
49	8.7	5.8	132	0.73
29	7.4	7.0	160	0.81
37	7.3	7.3	15	0.79
				0.73
39	7.4	6.8	144	0.75
3	4.7	6.2	86	0.80
0	7.3	8.4	192	0.55
53	8.4	6.0	120	0.74
47	9.7	7.3	196	0.72
44	10.5	7.3	200	0.72

y_train

42	3
48	4
7	2
14	1
32	3
49	4
29	3
37	3
56	4
39	3
3	2
0	1
53	4
47	4
44	4

Training set

X_test

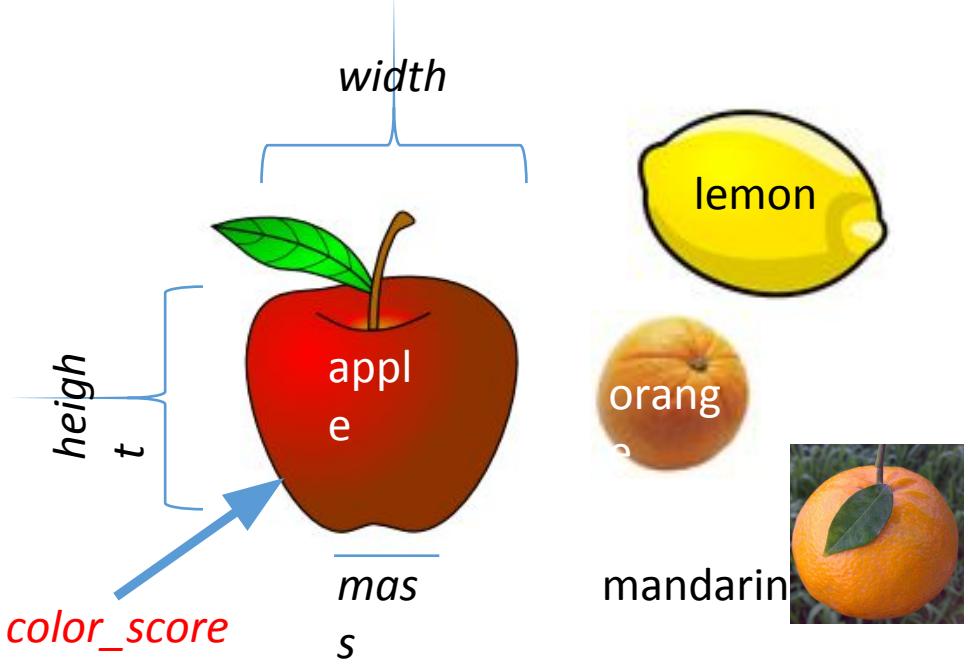
	height	width	mass	color_score
26	9.2	9.6	362	0.74
35	7.9	7.1	150	0.75
43	10.3	7.2	194	0.70
28	7.1	6.7	140	0.72
11	7.6	7.1	172	0.92
2	7.2	7.4	176	0.60
34	7.8	7.6	142	0.75
46	10.2	7.3	216	0.71
40	7.5	7.1	154	0.78
22	7.1	7.3	140	0.87
4	4.6	6.0	84	0.79
				0.93
				0.79
41	8.2	7.6	180	0.79
33	8.1	7.5	190	0.74

y_test

26	3
35	3
43	4
28	3
11	1
2	1
34	3
46	4
40	3
22	1
4	2
10	1
30	3
41	3
33	3

Test set

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`

Credit: 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)

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

These four columns contain the features of each 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	handarin	mandarin	86	6.2	4.7	0.80
4	2	handarin	mandarin	84	6.0	4.6	0.79
5	2	handarin	mandarin	80	5.8	4.3	0.77
6	2	handarin	mandarin	80	5.9	4.3	0.81
7	2	handarin	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
10	1	pple	braeburn	166	6.9	7.3	0.93
11	1	pple	braeburn	172	7.1	7.6	0.92
12	1	pple	braeburn	154	7.0	7.1	0.88
13	1	pple	golden_delicious	164	7.3	7.7	0.70
14	1	pple	golden_delicious	152	7.6	7.3	0.69
15	1	pple	golden_delicious	156	7.7	7.1	0.69
16	1	pple	golden_delicious	156	7.6	7.5	0.67
17	1	pple	golden_delicious	168	7.5	7.6	0.73
18	1	pple	cripps_pink	162	7.5	7.1	0.83
19	1	pple	cripps_pink	162	7.4	7.2	0.85
20	1	pple	cripps_pink	160	7.5	7.5	0.86

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

0.00 0.25 0.50 0.75 1.00

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

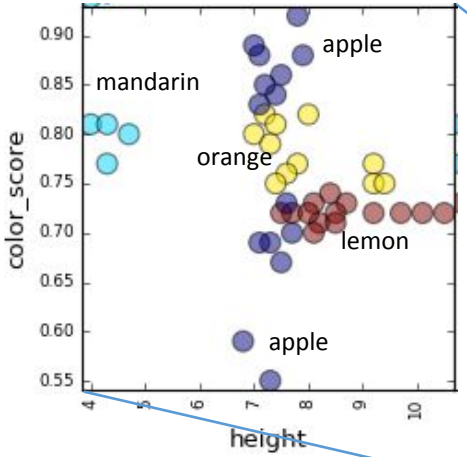
Some reasons why looking at the data initially is important!

- Inspecting feature values may help identify what cleaning or preprocessing still needs to be done once you can see the range or distribution of values that is typical for each attribute.
- You might notice missing or noisy data, or inconsistencies such as the wrong data type being used for a column, incorrect units of measurements for a particular column, or that there aren't enough examples of a particular class.
- You may realize that your problem is actually solvable without machine learning.

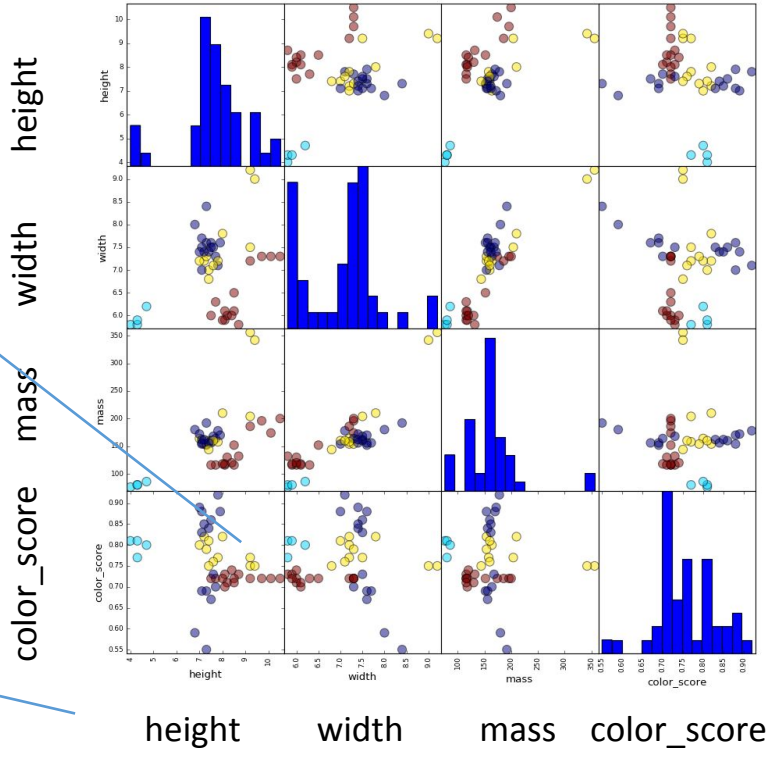
Examples of incorrect or missing feature values

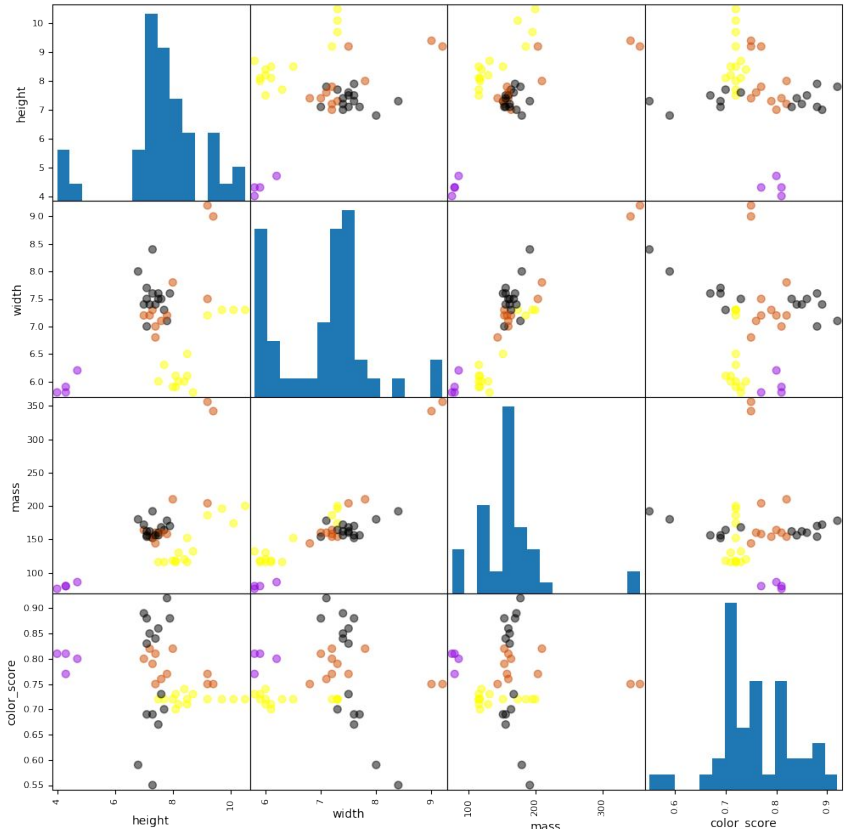
	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	192
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	apple	80	5.8	4.3	0.77
6	2	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		7.4	7.0	0.89
10	1	apple	braeburn		6.9	7.3	0.93
11	1	apple	braeburn		7.1	7.6	0.92
12	1	apple	braeburn		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

A pairwise feature scatterplot visualizes the data using all possible pairs of features, with one scatterplot per feature pair, and histograms for each feature along the diagonal.



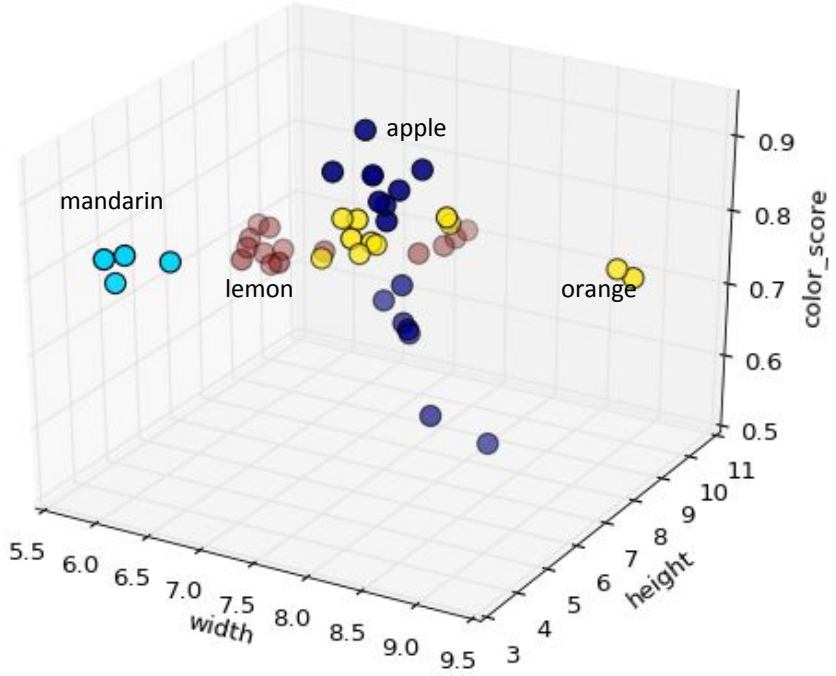
Individual scatterplot plotting all fruits by their **height** and **color_score**. Colors represent different fruit classes.



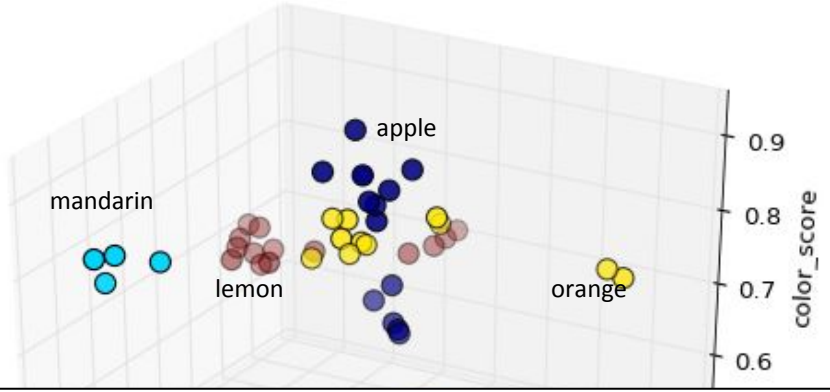



```
from matplotlib import cm
cmap = cm.get_cmap('gnuplot')
scatter = pd.scatter_matrix(X_train, c= y_train, marker = 'o', s=40, hist_kwds={'bins':15}, figsize=(12,12), cmap=cmap)
```

A three-dimensional feature scatterplot



A three-dimensional feature scatterplot




```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(X_train['width'], X_train['height'], X_train['color_score'], c = y_train, marker = 'o', s=100)
ax.set_xlabel('width')
ax.set_ylabel('height')
ax.set_zlabel('color_score')
plt.show()
```

Train Model: 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

A nearest neighbor algorithm: parameters

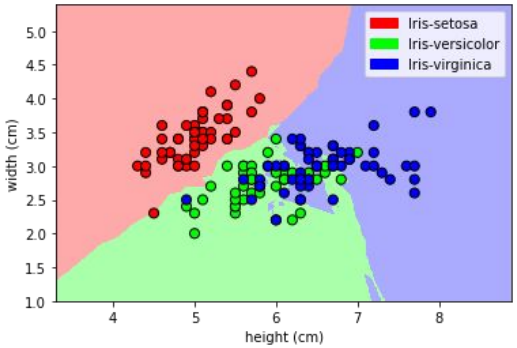
1. A distance metric
   Typically Euclidean (Minkowski with p = 2)
2. How many 'nearest' neighbors to look at?
   e.g. five
3. Optional weighting function on the neighbor points
   Ignored

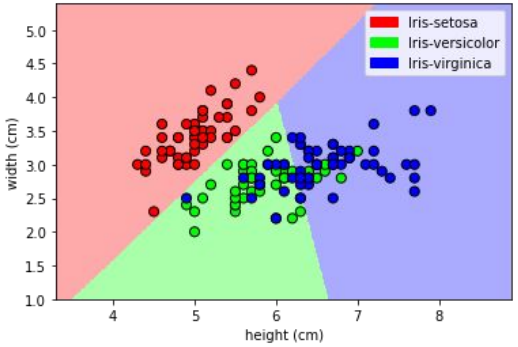
code k-NN classifier:

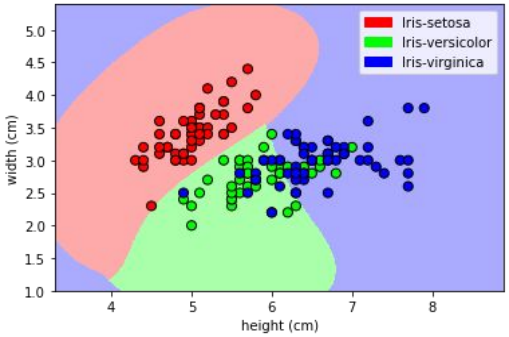
```
from sklearn.neighbors import KNeighborsClassifier


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)
```

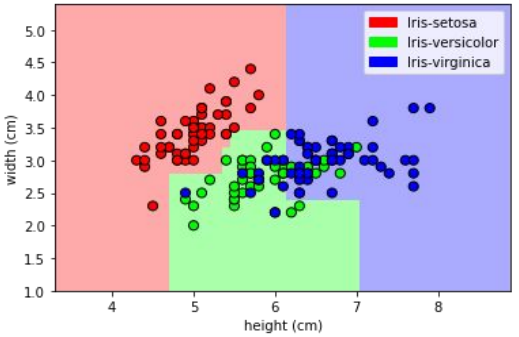
Think, Pair, Share

- Which of the following is from a k-NN classifier?





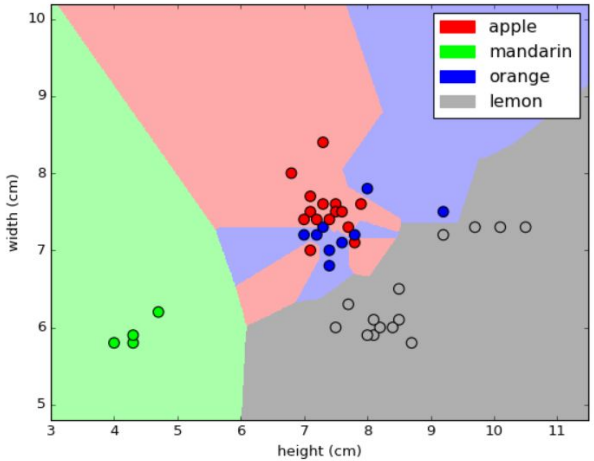




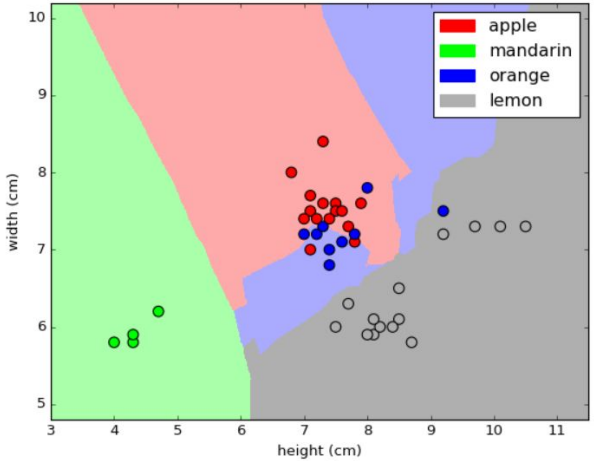
What is next thing to talk about?

Evaluation

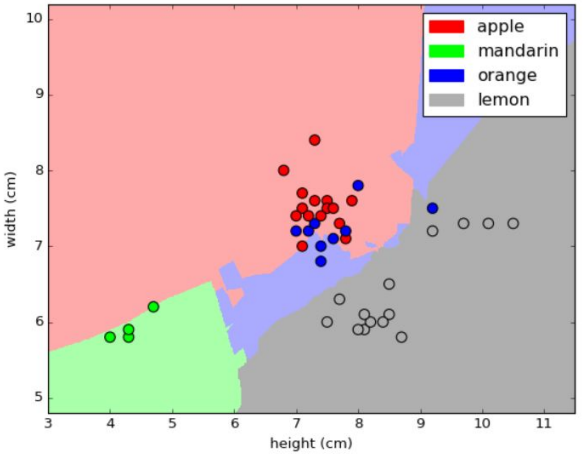
How can we peek inside k-NN to see what is going on?


apple
mandarin
orange
lemon
width (cm)
height (cm)

K=1


apple
mandarin
orange
lemon
width (cm)
height (cm)

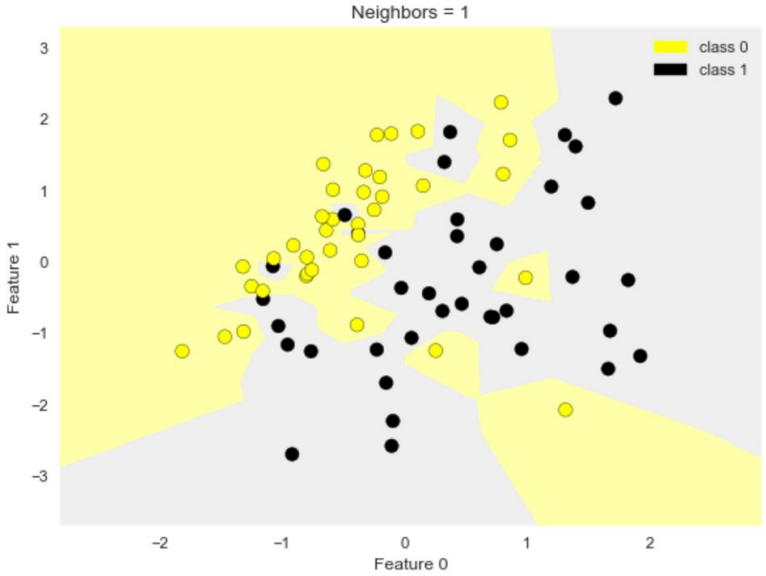
K=5

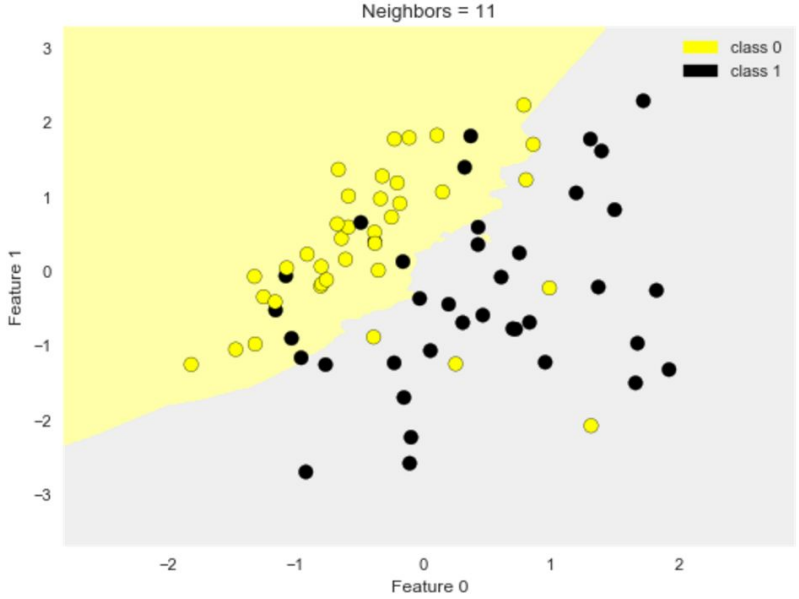

apple
mandarin
orange
lemon
width (cm)
height (cm)

K=10

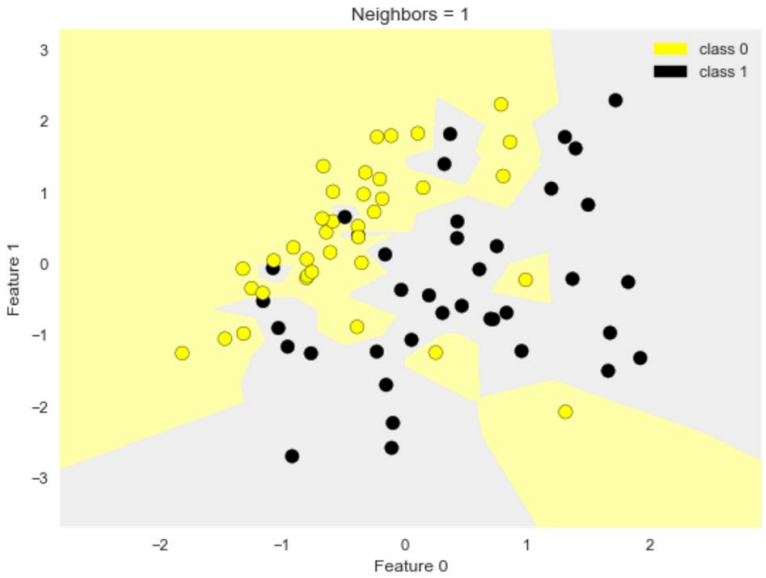
Think, Pair, Share:

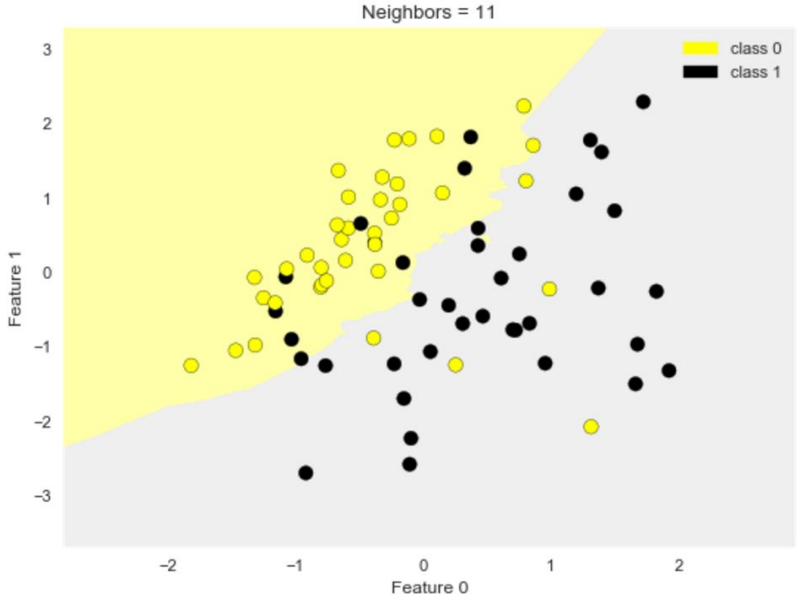
- Which diagram is 1-NN and which is 11-NN?



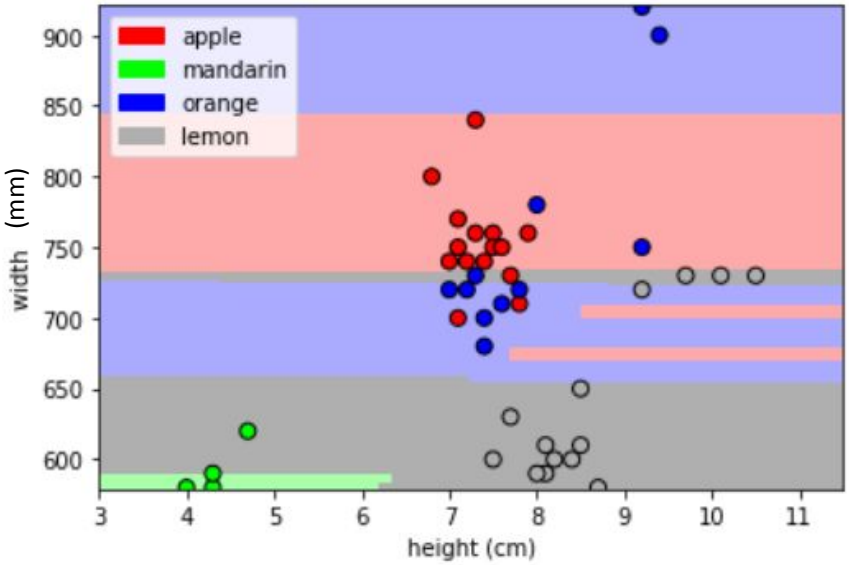


Nearest Neighbors Classification (k=1, 11)





What is going on?



K=5

Refinement: Feature Normalization

Our first encounter with the importance of feature normalization

- Important for some machine learning methods that all features are on the same scale (e.g. faster convergence in learning, more uniform or 'fair' influence for all weights)
  - e.g. regularized regression, k-NN, support vector machines, neural networks, …
- `StandardScaler` of the features:
  - Mean to 0
  - Standard Deviation to 1

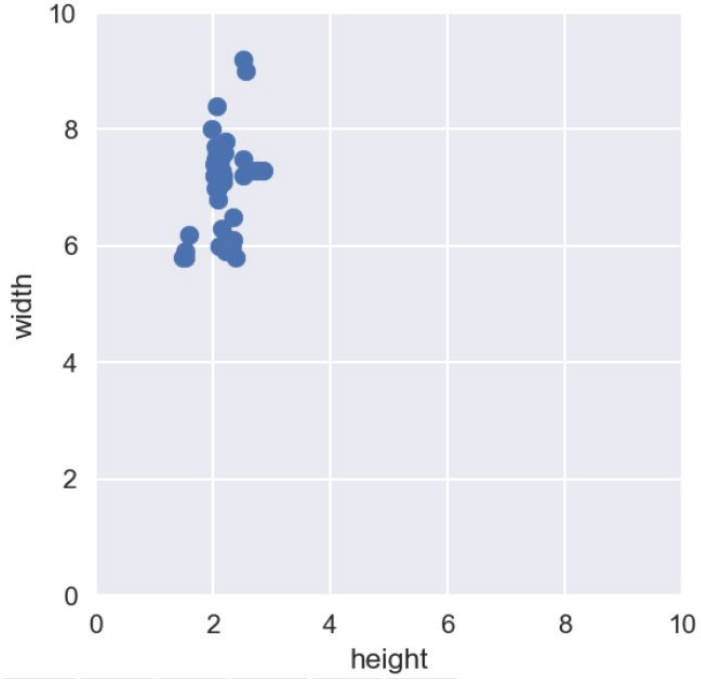
$$x_i' = (x_i - E[X])/\sigma(X)$$

Our first encounter with the importance of feature normalization

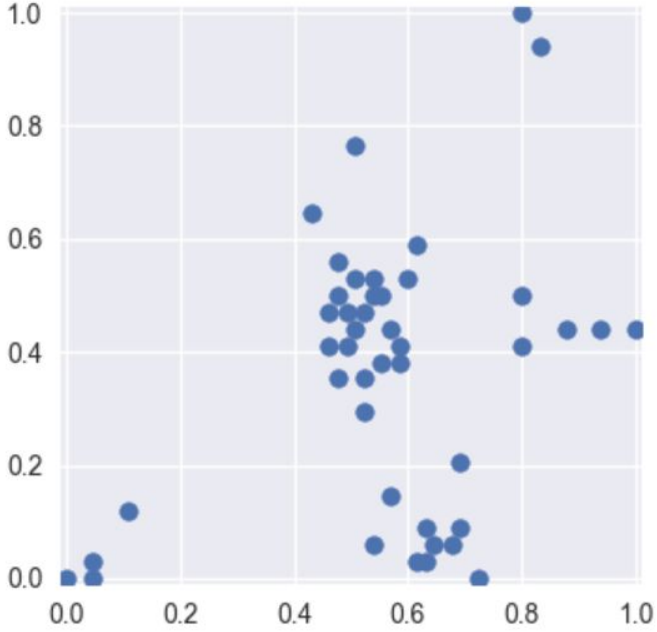
- MinMax scaling of the features:
  - For each feature x_i : compute the min value x_i^{MIN} and the max value x_i^{MAX} achieved across all instances in the training set.
  - For each feature: transform a given feature x_i value to a scaled version x_i' using the formula

$$x_i' = (x_i - x_i^{MIN})/(x_i^{MAX} - x_i^{MIN})$$

Feature Normalization with MinMaxScaler



Unnormalized data points



Normalized with MinMaxScaler

Feature Normalization: The test set must use identical scaling to the training set

- Fit the scaler using the training set, then apply the same scaler to transform the test set.
- Do not scale the training and test sets using different scalers: this could lead to random skew in the data.
- Do not fit the scaler using any part of the test data: referencing the test data can lead to a form of *data leakage.* More on this issue later in the course.

Evaluation: Accuracy

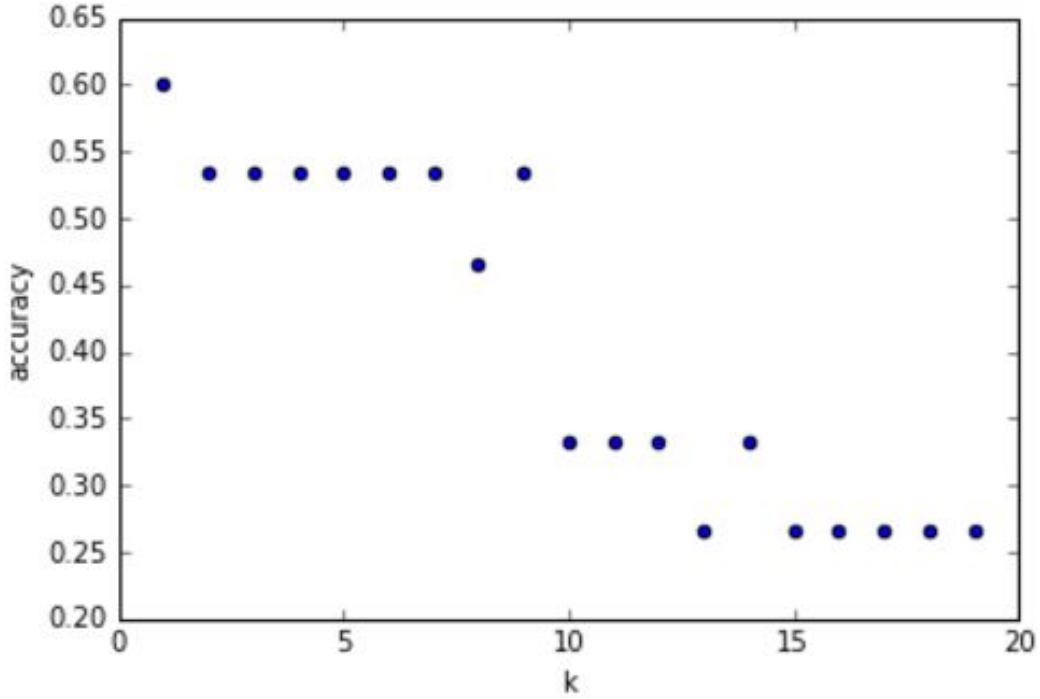
Accuracy

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

We can compute:

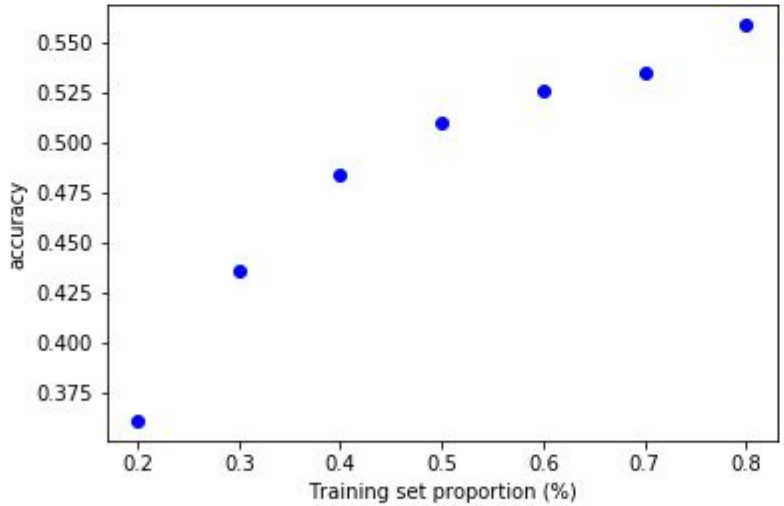
- Training set accuracy
- Test set accuracy

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?



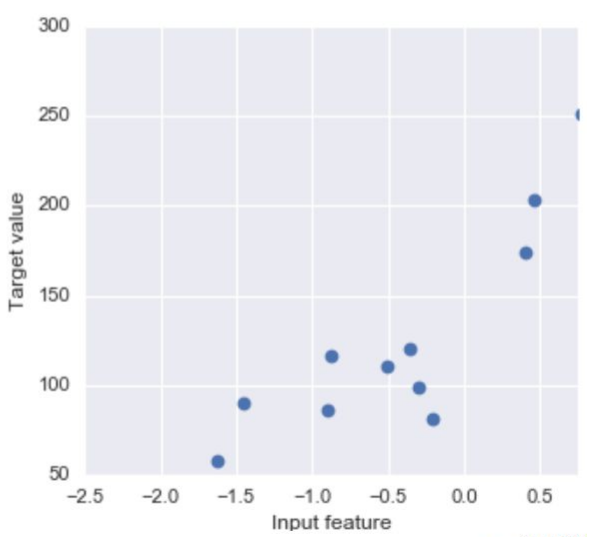
But we'll be **suspicious** of very high classifier accuracy on a training set, which may be evidence of overfitting. More on this soon!

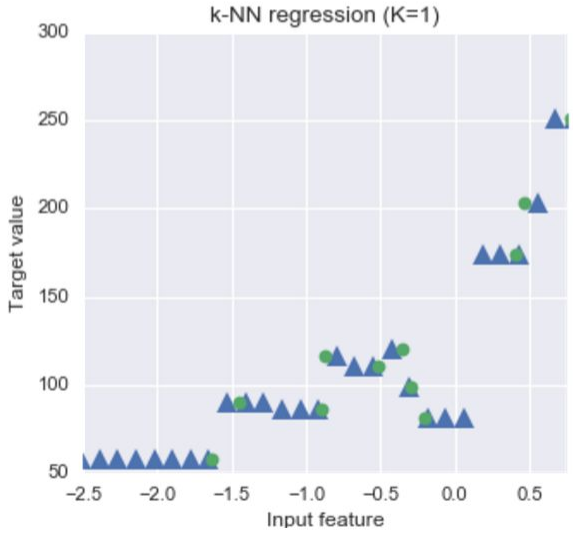
K-NN for Regression

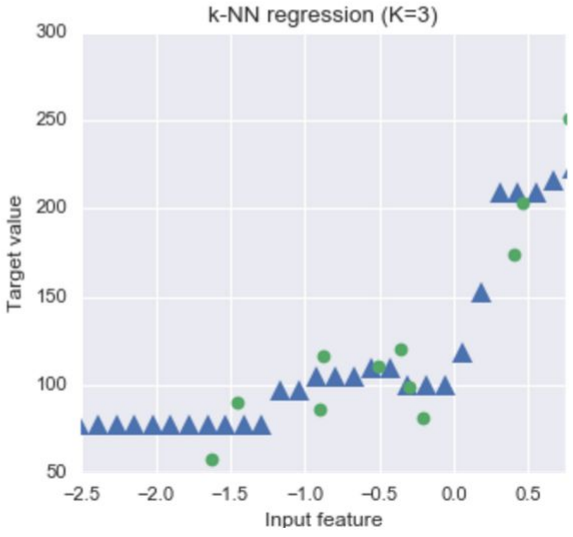
Recall the other main supervised learning task: regression

- What if we wanted to predict a continuous quantity given a fruit representation?
  - Say the height given the width.
- Could we somehow apply k-nearest neighbors to regression instead of classification?
- Yes! How?

k-Nearest Neighbors Regression







K-NN regression in scikit-learn

```
from sklearn.neighbors import KNeighborsRegressor


X_train, X_test, y_train, y_test =
    train_test_split(X_R1, y_R1, random_state = 0)


knnreg
 = KNeighborsRegressor(n_neighbors = 5).fit(X_train,
y_train)


print(knnreg.predict(X_test))
print('R-squared test score:
{:.3f}'.format(knnreg.score(X_test, y_test)
```

Compare k-NN classifier and regression:

```
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)

print(knnc.predict(X_test))

print('Accuracy test score: {:.3f}'.format(
    knnc.score(X_test, y_test))
```

```
from sklearn.neighbors import KNeighborsRegressor

X_train, X_test, y_train, y_test =
    train_test_split(X_R1, y_R1, random_state = 0)

knnr = KNeighborsRegressor(
   n_neighbors = 5).fit(X_train, y_train)

print(knnr.predict(X_test))

print('R-squared test score:
{:.3f}'.format(knnr.score(X_test, y_test)
```

The R^2 ("r-squared") Regression Score

- Measures how well a prediction model for regression fits the given data.
- The score is between 0 and 1:
  - A value of 0 corresponds to a constant model that predicts the mean value of all training target values.
  - A value of 1 corresponds to perfect prediction
- Also known as "coefficient of determination"

KNeighborsClassifier and KNeighborsRegressor: important parameters

Model complexity

- *n_neighbors :* number of nearest neighbors (k) to consider
  - D*efault = 5*

Model fitting

- *metric: distance function between data points*
  - Default: Minkowski distance with power parameter p = 2 (Euclidean)

Scikit-learn estimator class

1. Instantiate the object containing the algorithm and model that is learned from data

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

Some models have initialization parameters for regularization options, etc.

2. All estimators have a `fit(X_train, [y_train])` method

   Data *X* (NumPy array)

   Labels *y* *if a supervised algorithm* (1-d Numpy array)

   `logreg.fit(X, y)`

3. Use `predict(X_test)` or `transform(X_test)`

   All supervised model estimators also have a *score(X_test, y_test*) method