上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

《Python金融数据分析》

Hong Cheng（程宏）

School of Statistics and Mathematics

Shanghai LiXin University of Accounting and Finance

May 2022

- This course will introduce the learner to applied machine learning, **focusing more on the techniques and methods than on the statistics behind these methods**.
- The course will start with a discussion of **how machine learning is different than descriptive statistics,** and introduce the scikit learn toolkit through a tutorial.
- **The issue of dimensionality of data** will be discussed, and **the task of clustering data**, as well as **evaluating those clusters**, will be tackled.
- **Supervised approaches for creating predictive models will be described**, and learners will be able to apply the scikit learn predictive modelling methods while understanding process issues related to data generalizability (e.g. cross validation, overfitting).
- By the end of this course, students will be able to **identify the difference between a supervised (classification) and unsupervised (clustering) technique**, identify which technique they need to apply for a particular dataset and need, engineer features to meet that need, and **write python code to carry out an analysis**.

This week, you'll learn **what machine learning is** and **why it's important to data science**, **how machine learning is applied to key problems in our information economy**, and **how to set up your first machine learning application in Python**.

What is Machine Learning (ML)?

- **The study of computer programs (algorithms) that can learn by example**
- **ML algorithms can generalize from existing examples of a task**
  - *e.g. after seeing a training set of labeled images, an image classifier can figure out how to apply labels accurately to new, previously unseen images*

Speech Recognition

For example, how would you write down a set of rules in a programming language for accurately converting human speech to text?

The huge variety of **different pronunciations**, **vocabulary**, **accents**, and so forth.

Gargantuan Task

Inflexible and **not very robust** at recognizing different types of speech

Machine Learning models can learn by example

- Algorithms learn rules from labelled examples
- A set of labelled examples used for learning is called training data.
- The learned rules should also be able to generalize to correctly recognize or predict new examples not in the training set.

Audio signal	Output text
	How do I get to Ann Arbor?
	Hello!
	Please order me a pizza.

Machine Learning models learn from experience

- Labeled examples
  (Email spam detection)

- User feedback
  (Clicks on a search page)

- Surrounding environment
  (self-driving cars)

So **the basic problem of machine learning** is to explore how computers can program themselves to perform a task, and to improve their performance automatically as they gain more experience.

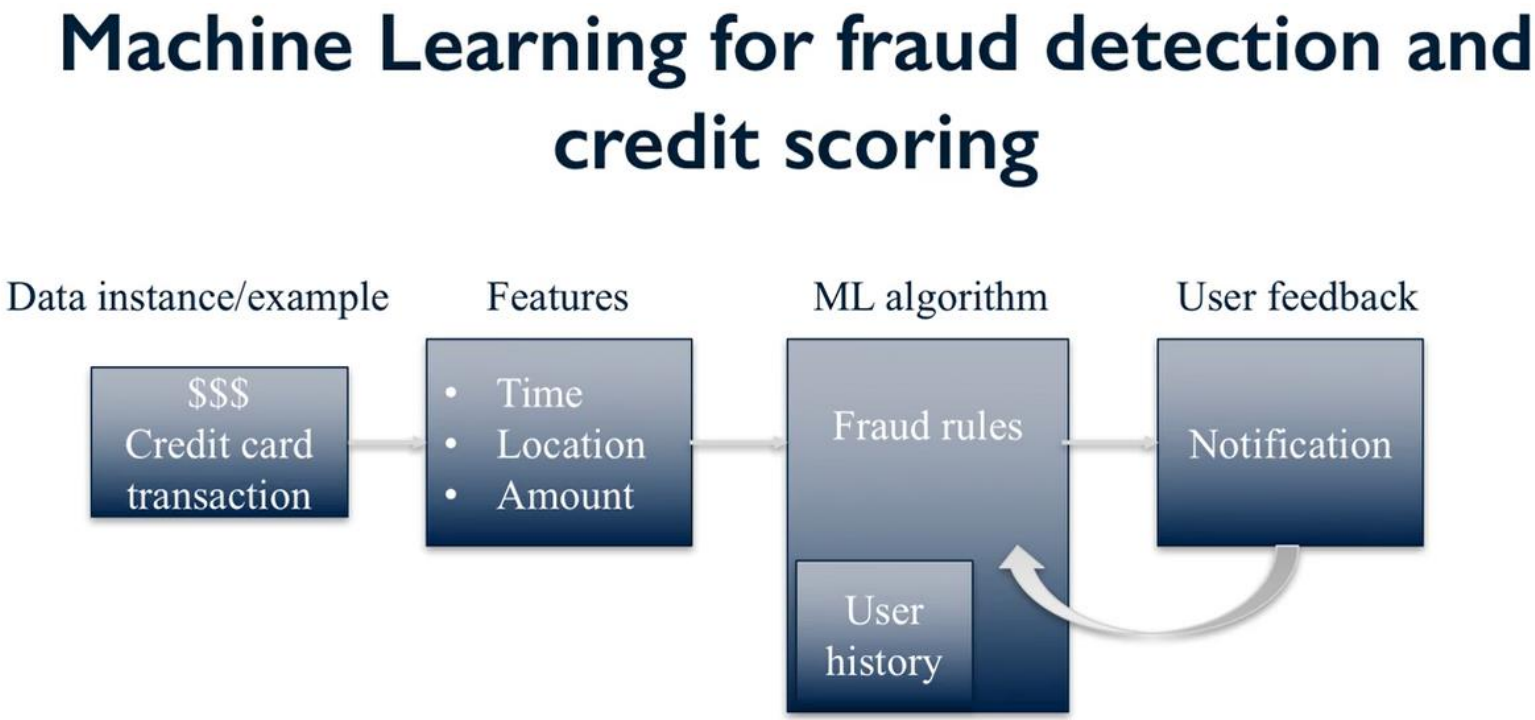
Machine Learning brings together statistics, computer science, and more..

- Statistical methods
  - *Infer conclusions from data*
  - *Estimate reliability of predictions*
- Computer science
  - *Large-scale computing architectures*
  - *Algorithms for capturing, manipulating, indexing, combining, retrieving and performing predictions on data*
  - *Software pipelines that manage the complexity of multiple subtasks*
- Economics, biology, psychology
  - *How can an individual or system efficiently improve their performance in a given environment?*
  - *What is learning and how can it be optimized?*

To accomplish this goal effectively and efficiently, Machine Learning draws heavily on statistics and computer science.

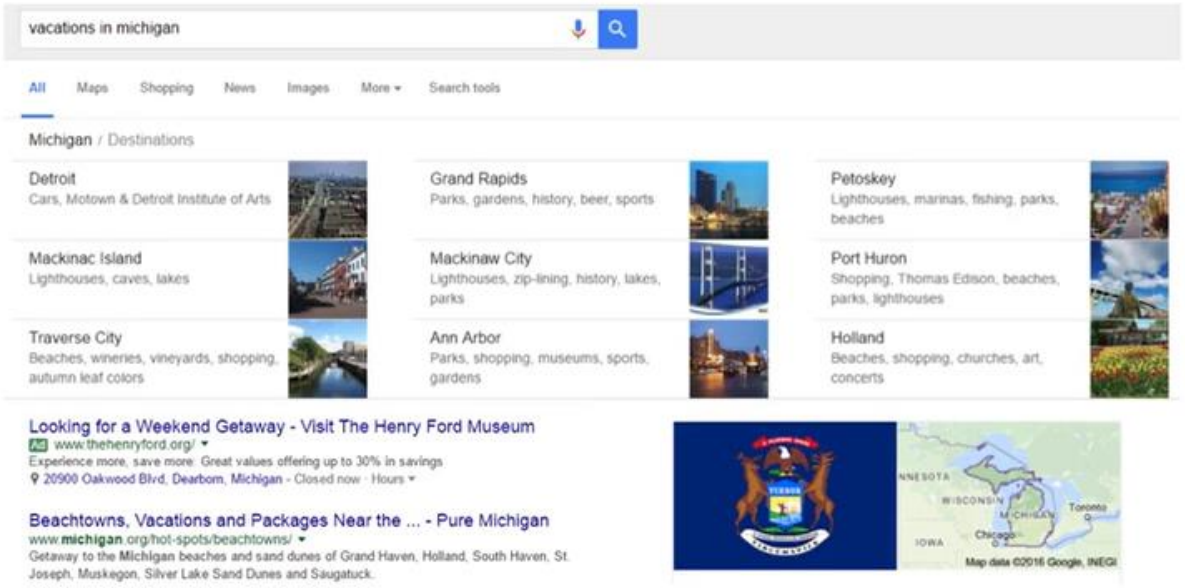
Machine Learning algorithms are now involved in more and more aspects of everyday life from what you read and watch, to how you shop, to who you meet and how you travel.

And here just a few examples, **for example**, **fraud detection**.



Search and recommendation systems are also a huge area of application for machine learning. And in fact, those **machine learning algorithms** are at the heart of how commercial search engines work, starting with the moment you begin typing in a query.

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



What is Applied Machine Learning?

- Understand basic ML concepts and workflow
- How to properly apply 'black-box' machine learning components and features
- Learn how to apply machine learning algorithms in Python using the scikit-learn package

- What is not covered in this course:
  - *Underlying theory of statistical machine learning*
  - *Lower-level details of how particular ML components work*
  - *In-depth material on more advanced concepts like deep learning*

Key Concepts in Machine Learning

Key types of Machine Learning problems

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

- **Classification (target values are discrete classes)**
- **Regression (target values are continuous values)**

So if the output is **a category a finite number of possibilities** such as a fraudulent or not fraudulent prediction for a credit card transaction.

We call this a classification problem within supervised learning, and **the function that we learn is called the classifier**.

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.

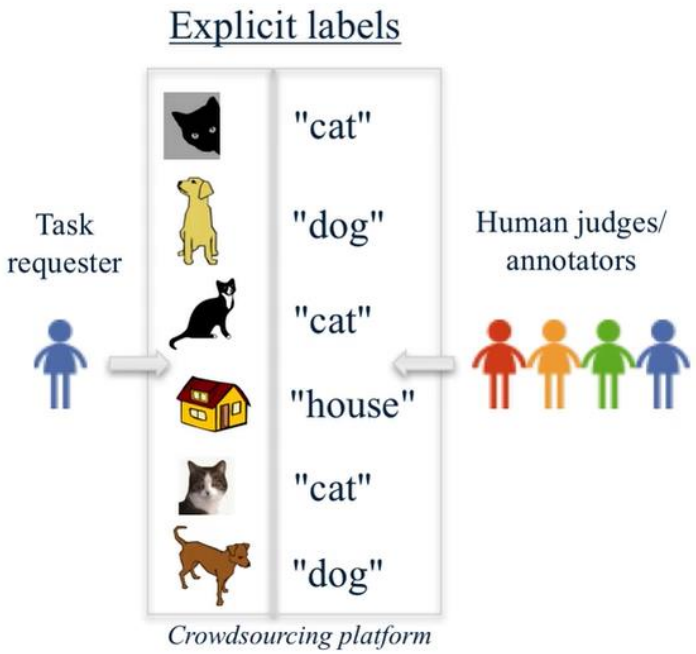
Now there're many algorithms that scientists have developed that can do supervise learning. That could be used to estimate this function F from the training data.

Supervised learning needs to have this **training set** with labeled objects in order to make its predictions.

But if the whole point is to predict these labels, **where does this initial set of label items come from?**

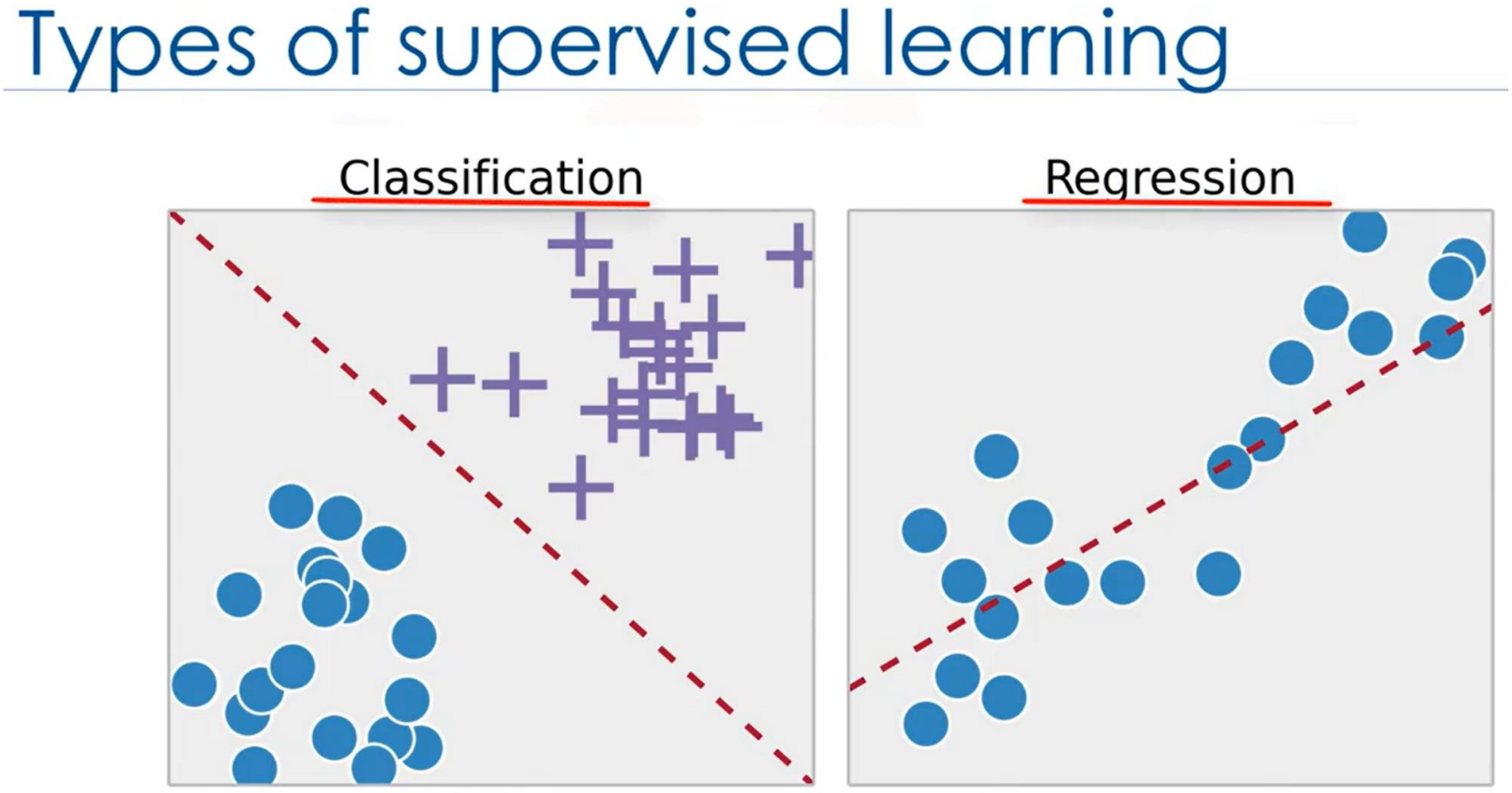
The answer is that **the training labels are typically provided by human judges**.

Examples of explicit and implicit label sources



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.

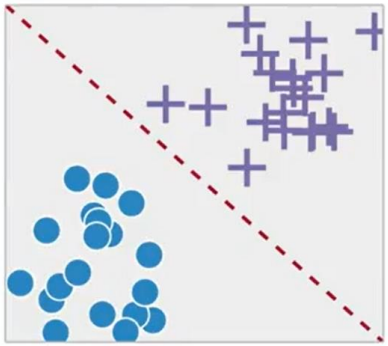
There are **two types of supervised learning techniques**. They are **classification**, and **regression**.



What is classification?

Classification is the process of predicting discrete class labels or categories.

ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000025	5	1	1	1	2	1	3	1	1	benign
1002945	5	4	4	5	7	10	3	2	1	benign
1015425	3	1	1	1	2	2	3	1	1	malignant
1016277	6	8	8	1	3	4	3	7	1	benign
1017023	4	1	1	3	2	1	3	1	1	benign
1017122	8	10	10	8	7	10		7	1	malignant
1018099	1	1	1	1	2	10	3	1	1	benign
1018561	2	1	2	H	2	1	3	1	1	benign
1033078	2	1	1	1	2	1	1	1	5	benign
1033078	4	2	1	1	2	1	2	1	1	benign



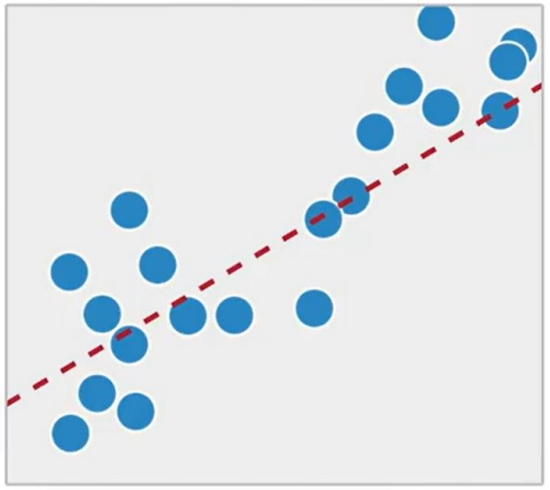
Classification is the process of predicting a discrete class label, or category.

What is regression?

Regression is the process of predicting continuous values.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Continuous Values



Regression is the process of predicting a continuous value as opposed to predicting a categorical value in classification.

Key types of Machine Learning problems

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

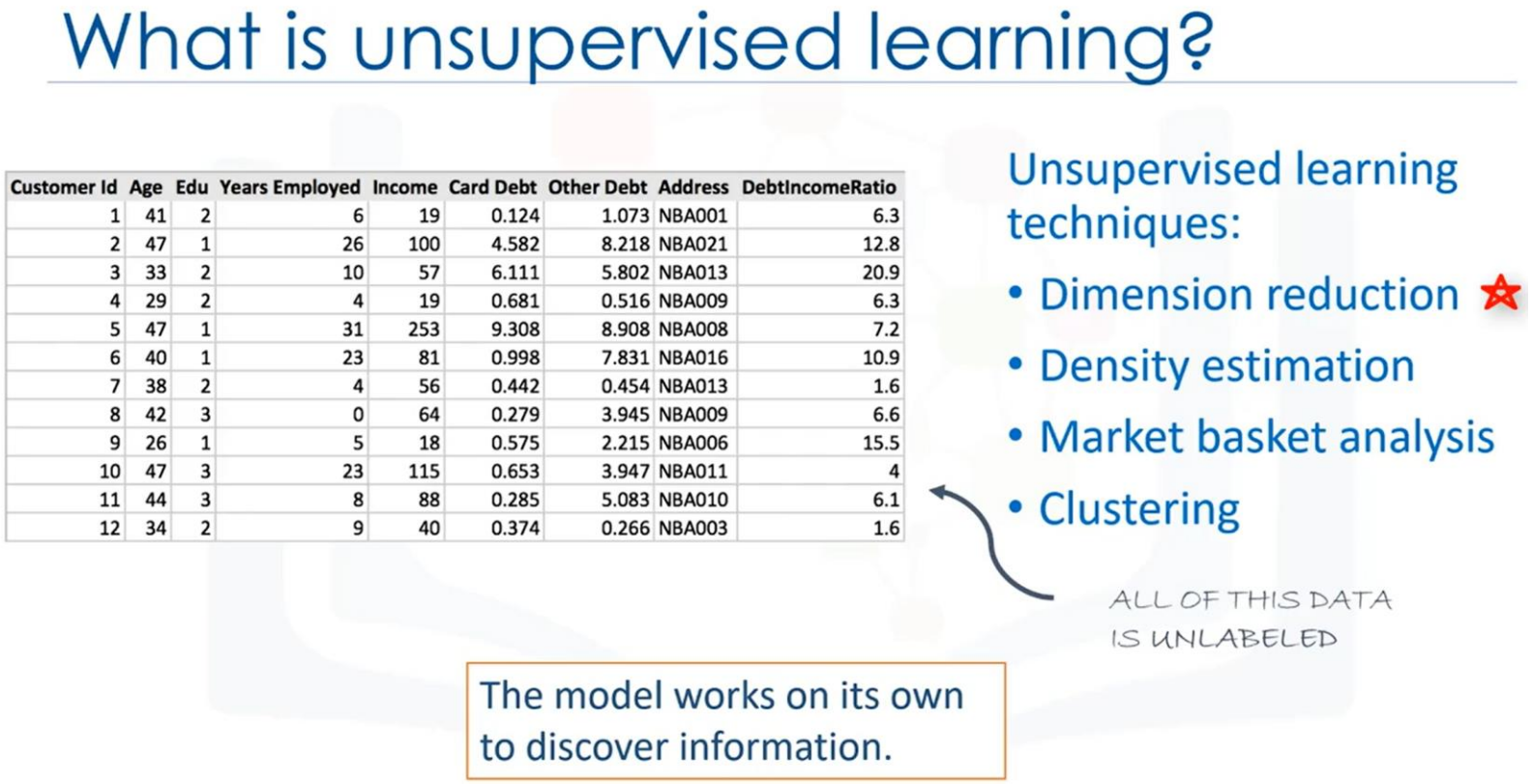
- Classification (target values are discrete classes)
- Regression (target values are continuous values)

Unsupervised machine learning: Find structure in *unlabeled data*

- **Find groups of similar instances in the data (clustering)**
- **Finding unusual patterns (outlier detection)**

In many cases we only have input data, we don't have any labels to go with the data.

what do you think unsupervised learning means?

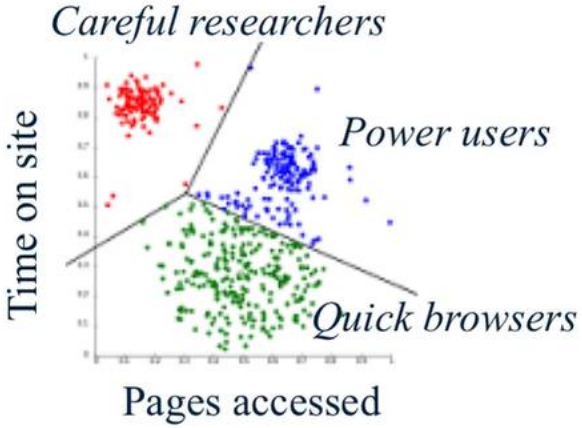


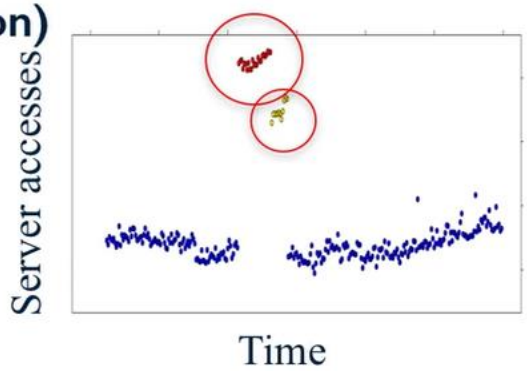
We do not supervise the model, but we let the model work on its own to discover information that may not be visible to the human eye. **It means**, the unsupervised algorithm trains on the dataset, and draws conclusions on unlabeled data.

Now what do I mean by structure? Well typically this means finding interesting clusters or groups within the data. So once we can discover this structure in the form of **clusters**, **groups** or **other interesting subsets**. The structure can be used for tasks like producing a useful summary of the input data maybe visualizing the structure.

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

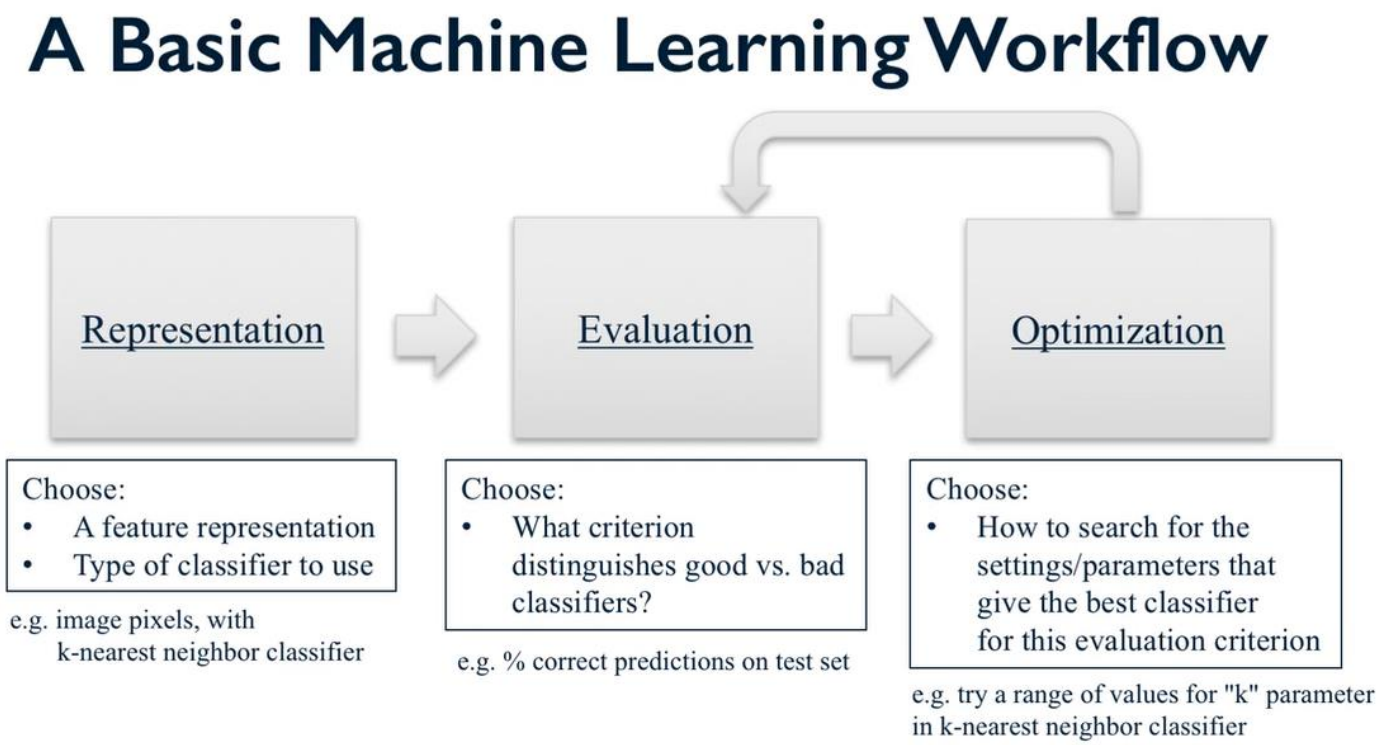
- Finding clusters of similar users (clustering)
- Detecting abnormal server access patterns (unsupervised outlier detection)





You can imagine then maybe tailing your site's offerings to each group so that prove the chance that a user from that group would maybe purchase a product or have a better experience using your site.

Suppose you have a situation where you think machine learning might be applicable, either using a supervised or an unsupervised approach. **How would you apply Machine Learning to solve your problem?** Well, there're **three basic steps** and I'm going to **use classification as my typical Machine Learning scenario.** So I'll often just use the term classifiers as an example of a machine learning task.

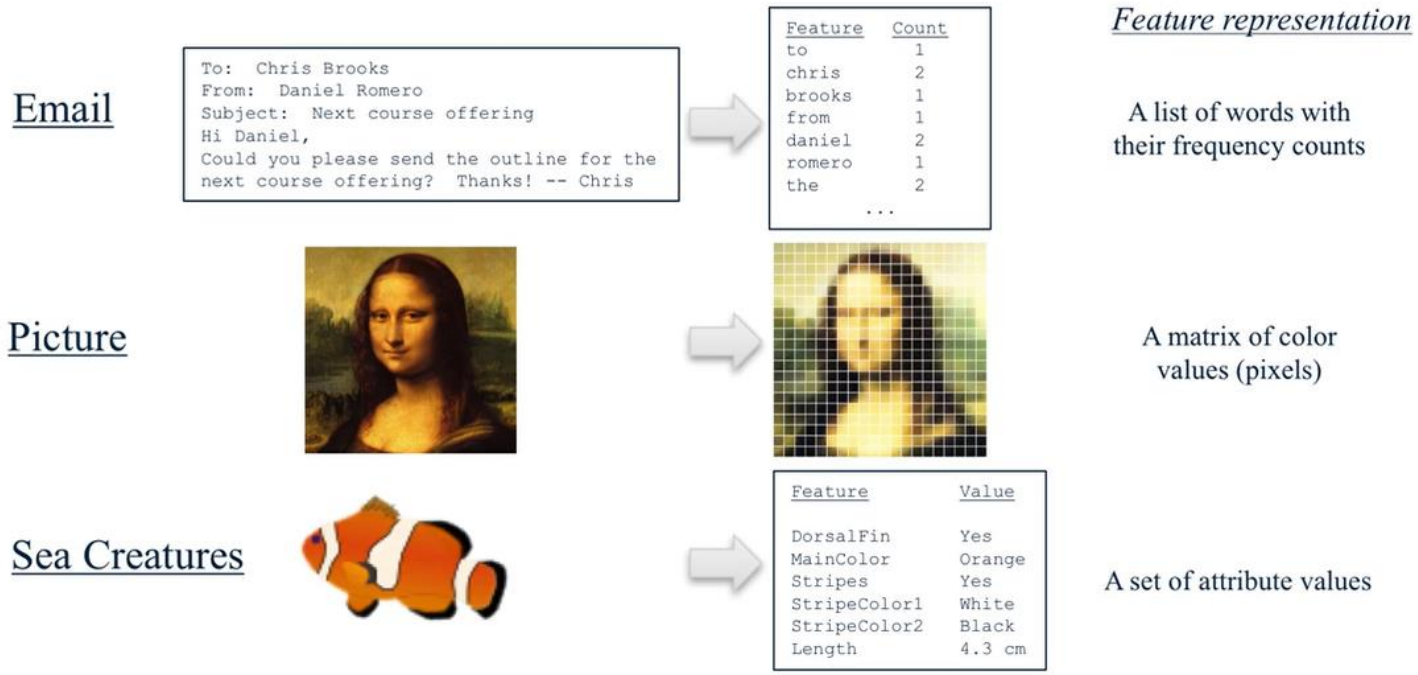


So let's go through these three three steps in a little more detail now. Let's first talk about what it means to convert the problem into a representation that a computer can deal with. **This involves two things**.

- You need to convert each input object, which we often call a sample, into a set of features that describe the object.
- Second, we need to pick a learning model, typically the type of classifier that you want the system to learn.

So let's look more closely at what we mean by a feature representation for an object.

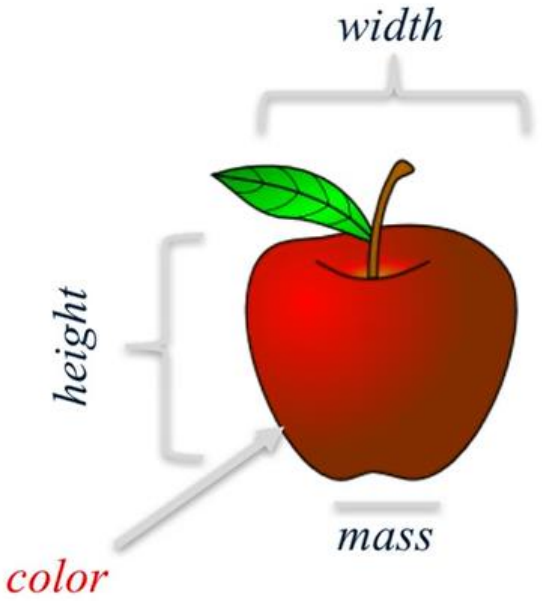
Feature Representations



A piece of fruit, like this apple could be represented by its color, its shape, its texture, and so forth.

So, these attribute values for an object are called **features**. You can think of the input data containing this feature representation, as the input to your function.

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)

So, to recap, **the biggest difference** between supervised and unsupervised learning is that supervised learning deals with labeled data while unsupervised learning deals with unlabeled data.

Supervised vs unsupervised learning

Supervised Learning

- **Classification:** Classifies labeled data
- **Regression:** Predicts trends using previous labeled data
- Has more evaluation methods than unsupervised learning
- Controlled environment

Unsupervised Learning

- **Clustering:** Finds patterns and groupings from unlabeled data
- Has fewer evaluation methods than supervised learning
- Less controlled environment

Python Tools for Machine Learning

We're going to make use of several important Python libraries that will support our work. **These include scikit-learn, SciPy, NumPy, pandas and matplotlib.**

We recommend installing all of these using the Anaconda Python distribution, since it comes with all the libraries we'll need in this part of the course.

The most important library we'll be using for machine learning is called scikit-learn.

scikit-learn: Python Machine Learning Library

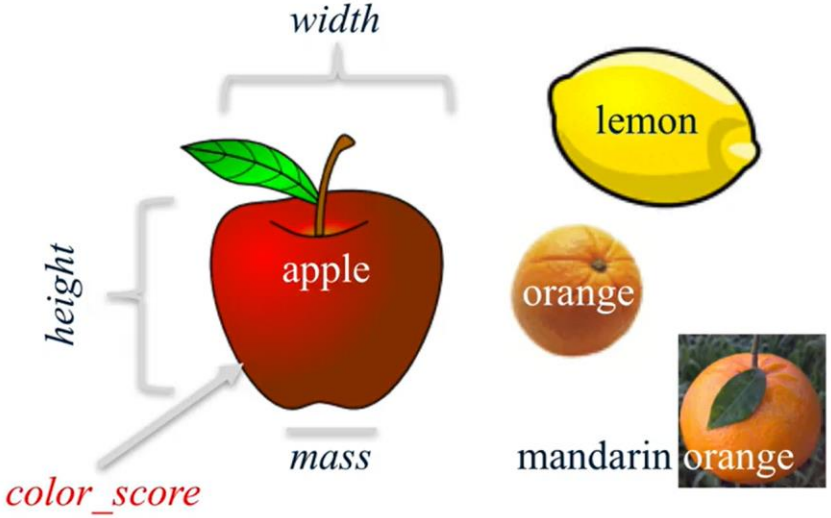
- scikit-learn Homepage
  http://scikit-learn.org/
- scikit-learn User Guide
  http://scikit-learn.org/stable/user_guide.html
- scikit-learn API reference
  http://scikit-learn.org/stable/modules/classes.html
- In Python, we typically import classes and functions we need like this:

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

An Example Machine Learning Problem

For our first machine learning exploration, **we're going to build an extremely simple form of object recognition system**. Now, although the example we'll use is very simple, it does reflect many of the same key machine learning concepts **that** go into building real-world commercial systems.

The Fruit Dataset



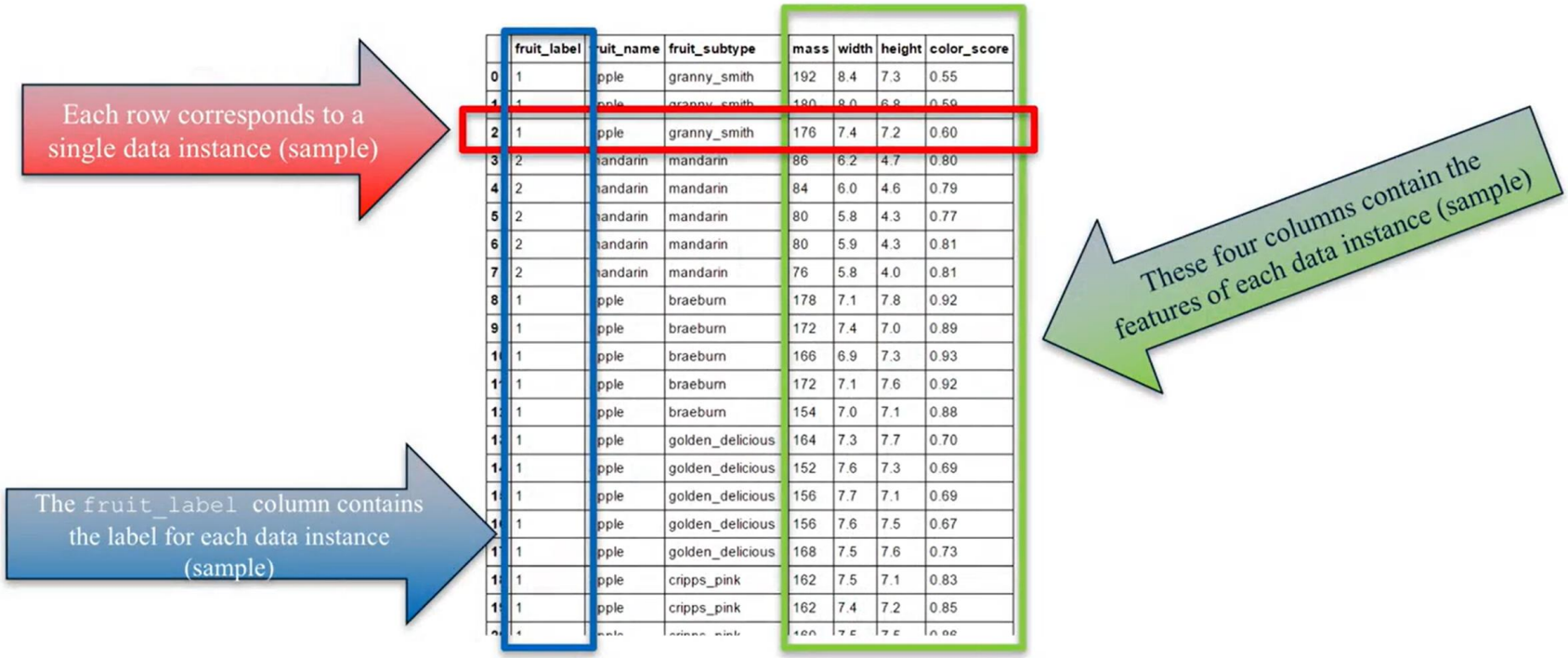
	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
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

This dataset is called **fruit data with colors.TXT**, and it's included in the folder of materials that you downloaded for this course.

The input data as a table



	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
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
17	1	apple	golden_delicious	168	7.5	7.6	0.73
18	1	apple	cripps_pink	162	7.5	7.1	0.83
19	1	apple	cripps_pink	162	7.4	7.2	0.85

Our goal here is to build a classifier from this data that can predict the correct type of fruit for any given observation of features, such as mass, height, width, and color score.

Applied Machine Learning, Module 1: A simple classification task

Import required modules and load data file

[1]:
```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split

fruits = pd.read_table('assets/fruit_data_with_colors.txt')
```

[2]:
```
fruits.head()
```

[2]:

	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

Finally, there's a feature stored in a column called **color score**.

That's meant to be a single number that captures a rough idea of the color of the fruit.

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

[3]:
```
# create a mapping from fruit label value to fruit name to make results easier to interpret
lookup_fruit_name = dict(zip(fruits.fruit_label.unique(), fruits.fruit_name.unique()))
lookup_fruit_name
```

[3]:
```
{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}
```

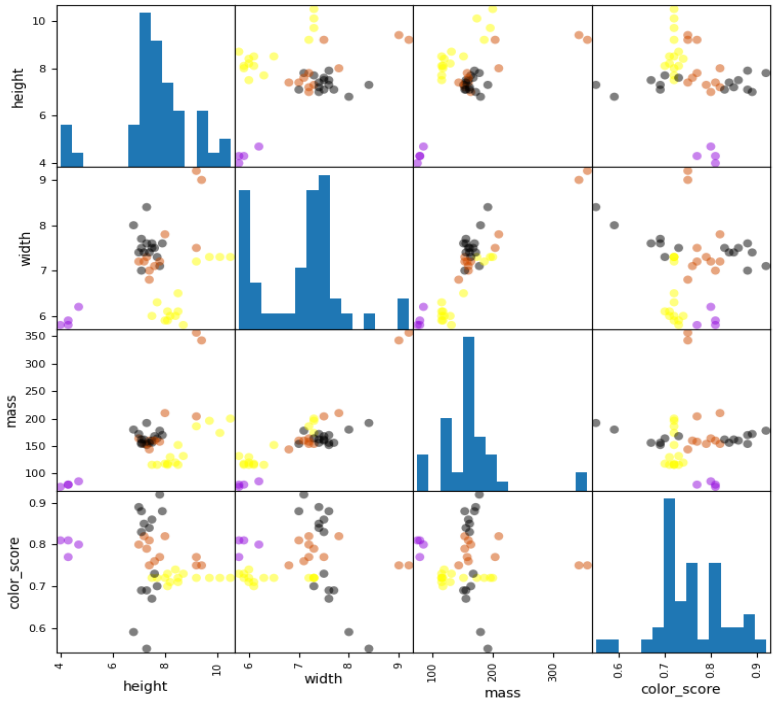
The file contains the mass, height, and width of a selection of oranges, lemons and apples. The heights were measured along the core of the fruit. The widths were the widest width perpendicular to the height.

Examining the data

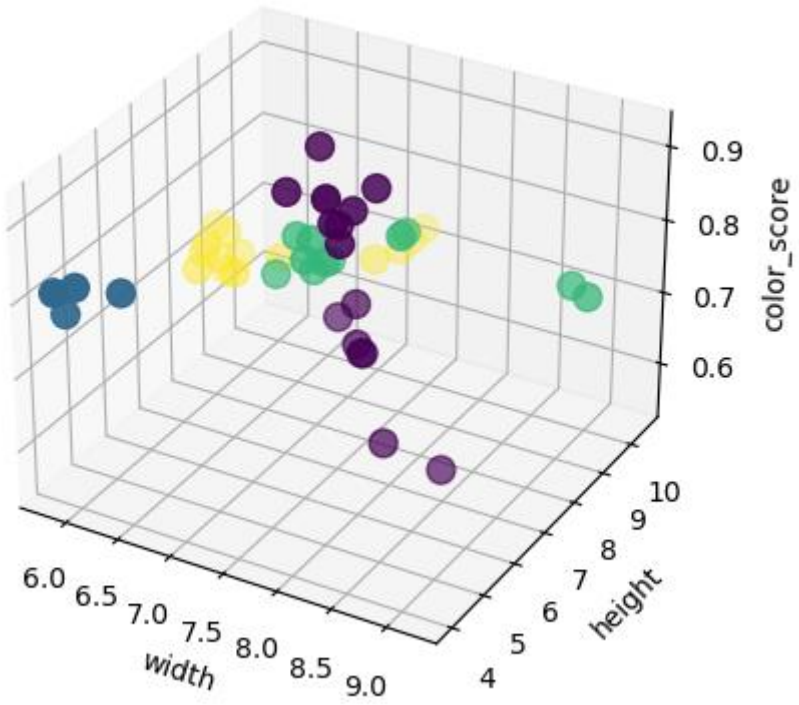
[5]:
```
pd.__version__
```

[5]:
```
'1.4.1'
```

[6]:
```
# plotting a scatter matrix
from matplotlib import cm

X = fruits[['height', 'width', 'mass', 'color_score']]
y = fruits['fruit_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

cmap = cm.get_cmap('gnuplot')
scatter = pd.plotting.scatter_matrix(X_train, c= y_train, marker = 'o', s=40, hist_kwds={'bins':15}, figsize=(9,9), cmap=cmap)
```



[7]:
```
# plotting a 3D scatter plot
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()
```

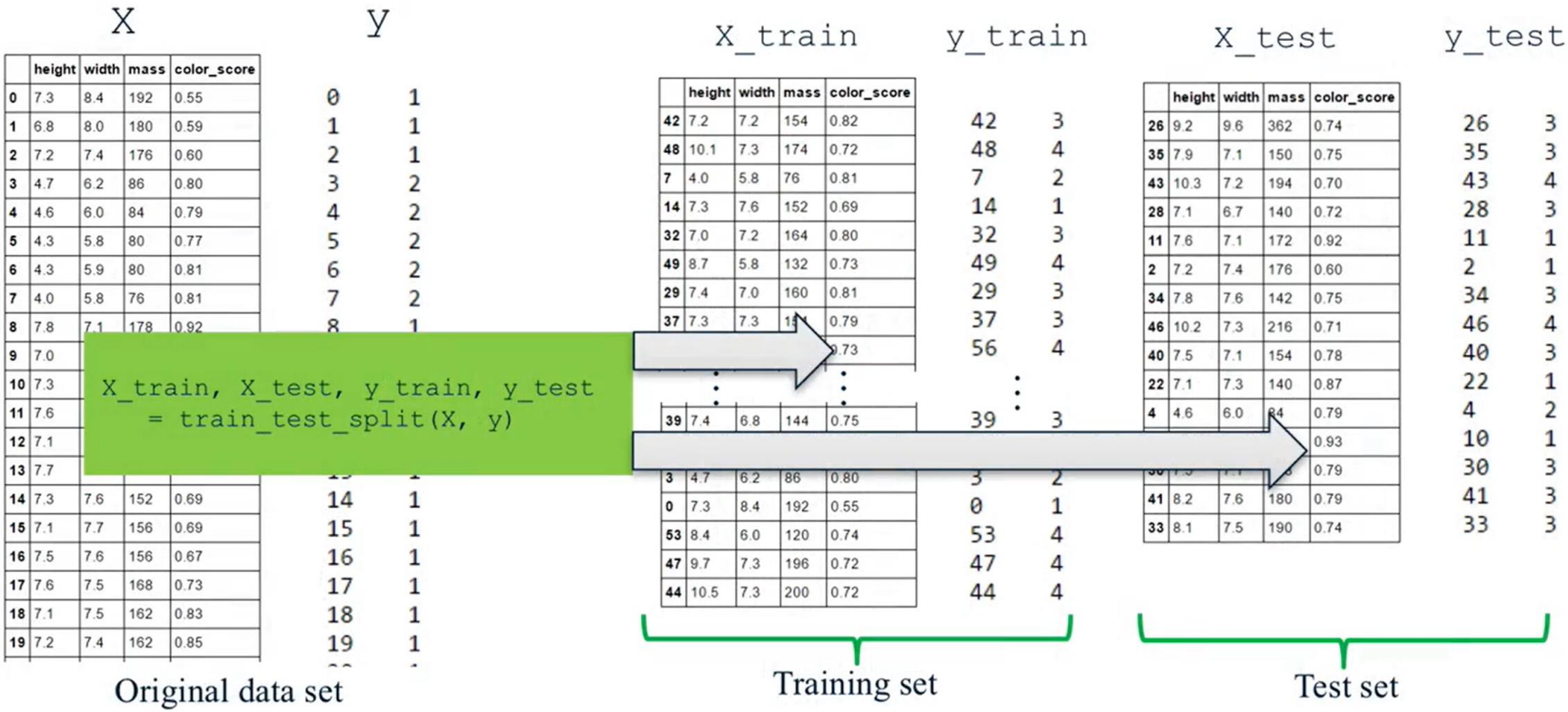


Now, assuming for the moment that we already had a classifier ready to go, **how would we know if its predictions were likely to be accurate?** Well, we could choose a fruit sample, called a test sample for which we already had a label.

We could feed the features of that piece of fruit into the classifier and then **compare the label that the classifier predicts with the actual true label of the fruit types**.

Since our only source of labeled data is the dataset we've been given, to estimate how well the classifier will do on future samples, what we'll do is **split the original dataset into two parts.**

Creating Training and Testing Sets



	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
14	1
15	1
16	1
17	1
18	1
19	1

	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

	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

To create training and test sets from an input dataset, scikit-learn provides a handy function that will do this split for us called not surprisingly, **train-test-split**, and here's an example of how we'll use it.

Create train-test split

```
[8]: # For this example, we use the mass, width, and height features of each fruit instance
X = fruits[['mass', 'width', 'height']]
y = fruits['fruit_label']

# default is 75% / 25% train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

If we want to get the same training and test split each time, we just make sure to pass in the same value of the **random state parameter** and **so here we're going to set that parameter to zero for all our examples**.

```
In [88]: fruits.shape
Out[88]: (59, 7)

In [92]: X_train.shape
Out[92]: (44, 4)

In [93]: X_test.shape
Out[93]: (15, 4)

In [94]: y_train.shape
Out[94]: (44,)

In [95]: y_test.shape
Out[95]: (15,)
```

In [96]: X_train

Out[96]:

	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	154	0.79
56	8.1	5.9	116	0.73
18	7.1	7.5	162	0.83
55	7.7	6.3	116	0.72
27	9.2	7.5	204	0.77
15	7.1	7.7	156	0.69
5	4.3	5.8	80	0.77
31	8.0	7.8	210	0.82
16	7.5	7.6	156	0.67

In [98]: y_train

Out[98]:

42	3
48	4
7	2
14	1
32	3
49	4
29	3
37	3
56	4
18	1
55	4
27	3
15	1
5	2
31	3
16	1
50	4
20	1
51	4
8	1
13	1
25	3
17	1
58	4
57	4
52	4
38	3
1	1
12	1
45	4
24	3
6	2

In [97]: X_test

Out[97]:

	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
10	7.3	6.9	166	0.93
30	7.5	7.1	158	0.79
41	8.2	7.6	180	0.79
33	8.1	7.5	190	0.74

In [99]: y_test

Out[99]:

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

Now that we have a training and a test set, we're ready for the next step and we'll look now into more depth at the data itself before we **proceed with giving it to a machine-learning algorithm.**

Create classifier object

[9]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5)
```

Train the classifier (fit the estimator) using the training data

[10]:
```
knn.fit(X_train, y_train)
```

[10]:
```
KNeighborsClassifier()
```

Estimate the accuracy of the classifier on future data, using the test data

[11]:
```
knn.score(X_test, y_test)
```

[11]:
```
0.5333333333333333
```

Use the trained k-NN classifier model to classify new, previously unseen objects

[12]:
```
# first example: a small fruit with mass 20g, width 4.3 cm, height 5.5 cm
fruit_prediction = knn.predict([[20, 4.3, 5.5]])
lookup_fruit_name[fruit_prediction[0]]
```

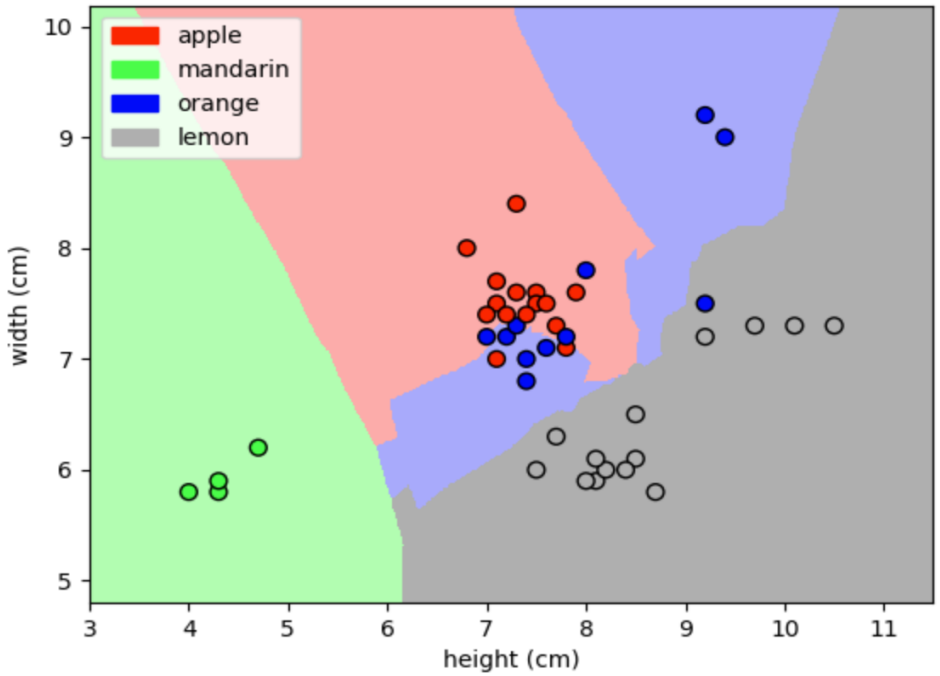
[12]:
```
'mandarin'
```

[13]:
```
# second example: a larger, elongated fruit with mass 100g, width 6.3 cm, height 8.5 cm
fruit_prediction = knn.predict([[100, 6.3, 8.5]])
lookup_fruit_name[fruit_prediction[0]]
```

Plot the decision boundaries of the k-NN classifier

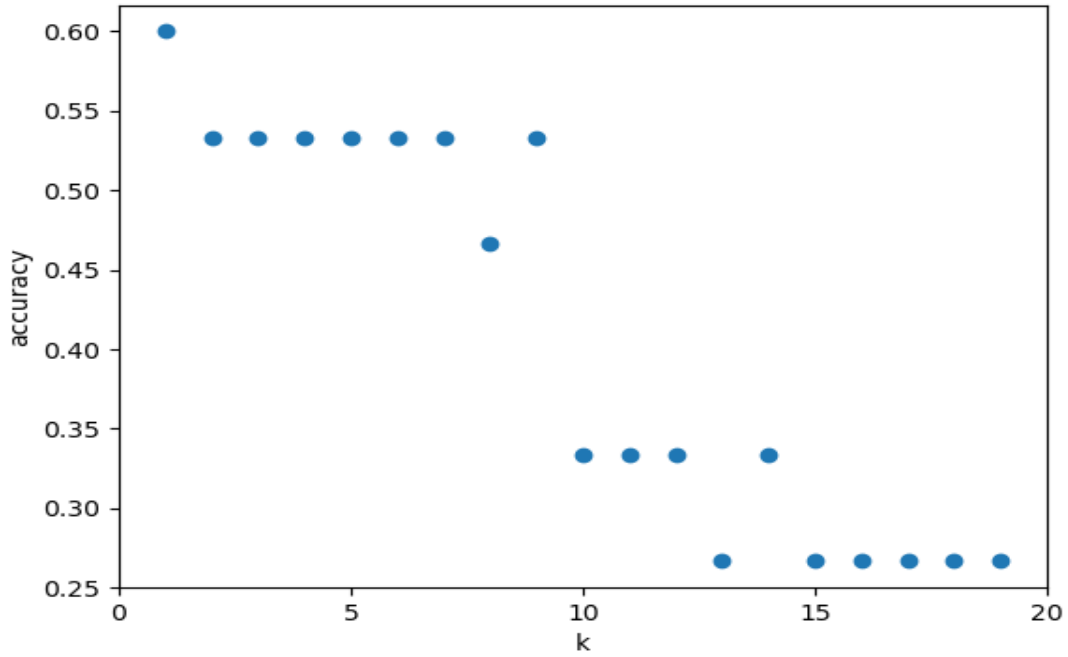
[14]:
```
from adspy_shared_utilities import plot_fruit_knn

plot_fruit_knn(X_train, y_train, 5, 'uniform')   # we choose 5 nearest neighbors
```

```
<IPython.core.display.Javascript object>
```



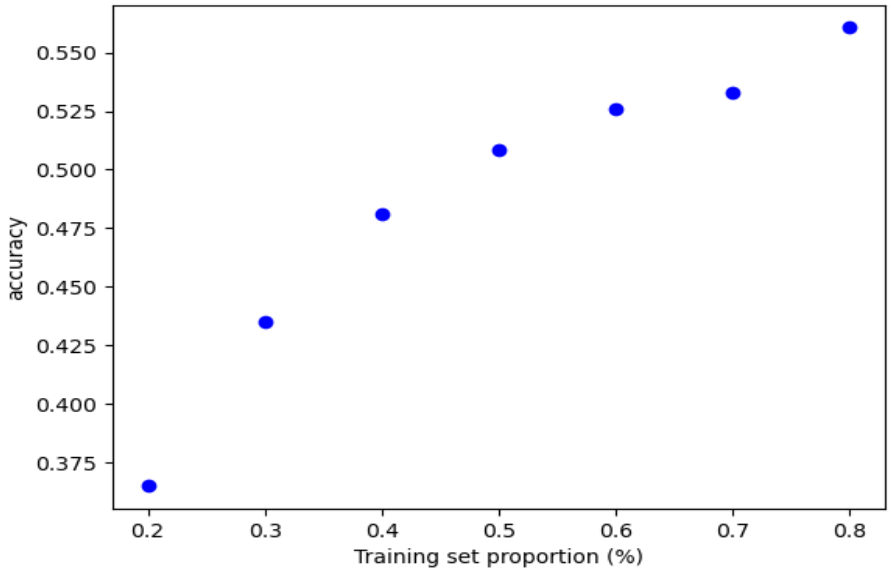
How sensitive is k-NN classification accuracy to the choice of the 'k' parameter?

[15]:
```
k_range = range(1,20)
scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))

plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20]);
```



How sensitive is k-NN classification accuracy to the train/test split proportion?

[16]:
```
t = [0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2]

knn = KNeighborsClassifier(n_neighbors = 5)

plt.figure()

for s in t:

    scores = []
    for i in range(1,1000):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1-s)
        knn.fit(X_train, y_train)
        scores.append(knn.score(X_test, y_test))
    plt.plot(s, np.mean(scores), 'bo')

plt.xlabel('Training set proportion (%)')
plt.ylabel('accuracy');
```



Thank You