Introduction to Supervised Learning

Dr. Qiwei Gan

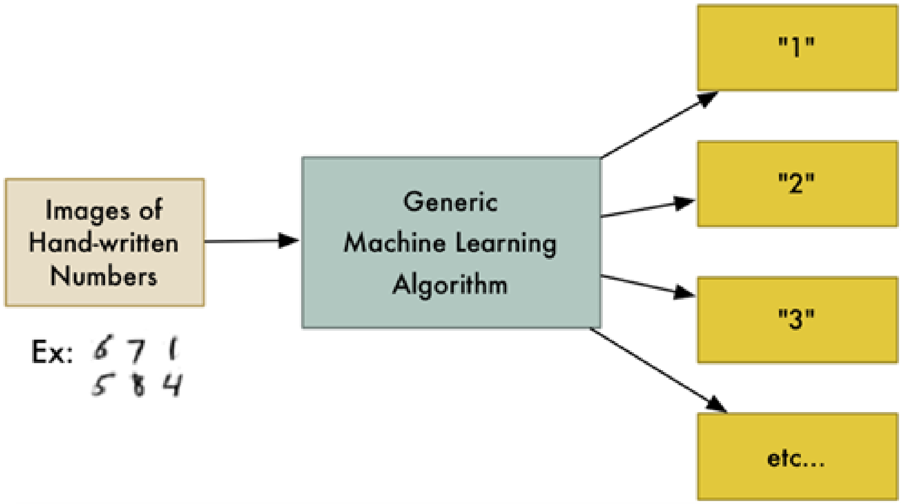
python™

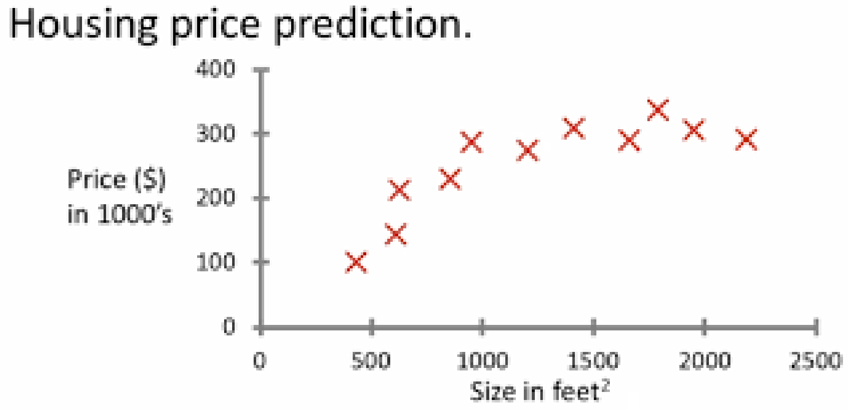
Recap: Libraries for Web Scrapping

- The Farm: Requests
- The Stew: Beautiful Soup 4
- The Salad: lxml
- The Restaurant: Selenium
- The Chef: Scrapy

Supervised Learning

- "Right answers" given
  - Learn to predict target values from labelled data.
- Classification
  - Output is discrete
- Regression
  - Output is continuous





A Basic Machine Learning Workflow

Representation → Evaluation → Optimization

Representation

Choose:
- A feature representation
- Type of classifier to use

e.g. image pixels, with
k-nearest neighbor classifier

Evaluation

Choose:
- What criterion distinguishes good vs. bad classifiers?

e.g. % correct predictions on test set

Optimization

Choose:
- How to search for the settings/parameters that give the best classifier for this evaluation criterion

e.g. try a range of values for "k" parameter in k-nearest neighbor classifier

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

Handwritten Digits Recognition

	1	2	3	4	5
1	0	0	0	0	0
2	0	1	0	1	0
3	0	1	1	1	0
4	0	0	0	1	0
5	0	0	0	0	0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	1	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0

Input Data as a Table

```
In [2]: labeled_images = pd.read_csv('C:\\data\\digits_train.csv')
```

```
In [4]: labeled_images.head()
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...
0	1	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	1	0	0	0	0	0	0	0	0	0	...
3	4	0	0	0	0	0	0	0	0	0	...
	0	0	0	0	0	0	0	0	0	0	...

Each row corresponds to a single data instance (sample)

The ‘label’ column contains the label for each data instance (sample)

Index

These columns contains the features of each data instance (sample)

Conventions

- Capital X: features matrix
- Lower y: label

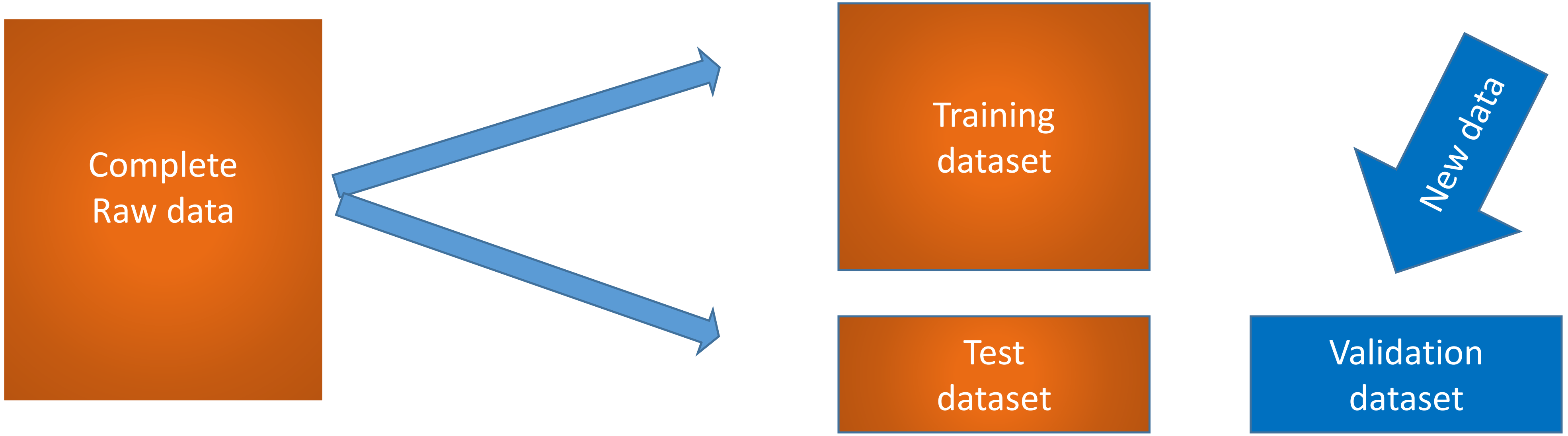
In [4]:
```
#select images features from the second column to the last column.
X = labeled_images.iloc[:,1:]

#select the first column which is the label, or the digit.
y = labeled_images.iloc[:,:1]

print('The original input dataset shape is: ', labeled_images.shape)
print('The X dataset shape is: ',X.shape)
print('The y dataset shape is: ', y.shape)
```

```
The original input dataset shape is:  (42000, 785)
The X dataset shape is:  (42000, 784)
The y dataset shape is:  (42000, 1)
```

Splitting Input Data into Train/Test



Data used to build the machine learning model, are called the *training data.*
The rest of the data will be used to assess how well the model works; these data are called *test data.*

Splitting Input Data into Train/Test

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X, y, train_size=0.8, random_state=0)
print('The X_train dataset shape is: ', X_train.shape)
print('The y_train dataset shape is: ', y_train.shape)
print('The X_test dataset shape is: ', X_test.shape)
print('The y_test dataset shape is: ', y_test.shape)
```

```
The X_train dataset shape is:  (33600, 784)
The y_train dataset shape is:  (33600, 1)
The X_test dataset shape is:  (8400, 784)
The y_test dataset shape is:  (8400, 1)
```

train_size specifies the proportion of train vs. test datasets, while ***random_state*** makes sure you get the same datasets every time.

Splitting Input Data into Train/Test

Original data set

```
X.head(20)
```

	pixel0	pixel1	pixel2	pixel3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0

```
y.head(20)
```

	label
0	1
1	0
2	1
3	4
4	0
5	0
6	7
7	3
8	5
9	3
10	8
11	9
12	1
13	3
14	3
15	1

Training data set

```
X_train.head(20)
```

	pixel0	pixel1	pixel2
39317	0	0	0
32837	0	0	0
16644	0	0	0
20005	0	0	0
1533	0	0	0
41842	0	0	0
7781	0	0	0
28433	0	0	0
5554	0	0	0
31233	0	0	0
40004	0	0	0
3094	0	0	0
28051	0	0	0
12857	0	0	0
32916	0	0	0
1153	0	0	0

```
y_train.head(20)
```

	label
39317	6
32837	6
16644	4
20005	4
1533	2
41842	2
7781	3
28433	1
5554	3
31233	3
40004	8
3094	5
28051	6
12857	7
32916	6
1153	8

Test data set

```
X_test.head(20)
```

	pixel0	pixel1	pixel2
16275	0	0	0
19204	0	0	0
18518	0	0	0
25780	0	0	0
16228	0	0	0
15824	0	0	0
29252	0	0	0
28482	0	0	0
13779	0	0	0
25912	0	0	0
27141	0	0	0
21848	0	0	0
32576	0	0	0
7975	0	0	0
10594	0	0	0
37445	0	0	0

```
y_test.head(20)
```

	label
16275	3
19204	6
18518	9
25780	5
16228	6
15824	5
29252	6
28482	0
13779	0
25912	1
27141	7
21848	1
32576	5
7975	7
10594	8
37445	1

Selection of Models Types

- K-Nearest Neighbors
- Linear Models
  - OLS regression, Ridge regression, Lasso regression, Logistic regression, etc.
- Naïve Bayes Classifiers
- Decision Trees
- Ensembles of Decision Trees
- Support Vector Machine
- Neural Networks

Training Models

Create classifier object

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

knn = KNeighborsClassifier(n_neighbors = 5)
```

Tuning the parameters

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

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

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

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

Use the trained classifier model to classify (predict) new, previously unseen data

In [ ]:
```
predictd_y=knn.predict(y_test)
```

Preprocessing Data/Feature Engineering

- Missing, inconsistent data
  - Remove, fill missing, correct, etc.
- Non-numerical data
  - Label encoding, Categorize, dummies, etc.
- Scale data
  - E.g., scale to (0,1), or scale to mean=0, standard deviation=1
- Dimensionality reduction
  - Reduce number of features, e.g., Principle Component Analysis
- y processing
  - Transform, oversampling, undersampling, SMOT, etc.

Model Evaluation

- Metrics
  - Accuracy rate, R squared, Confusion matrix, precision, recall, F measure, AUC
- Cross validation

Tuning Parameters and Chain Operations

- Grid search
- Pipelines

Questions?