Data Processing, Analysis and Visualization in Python

Basic Machine Learning I

1

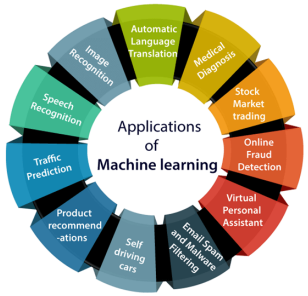
Outline

- What is ML?
  - AI vs ML vs DL
- The ML process
- Supervised vs unsupervised ML
- Introduction to Scikit-Learn

2

How is ML used?

- Image recognition
- Speech recognition
- Medical diagnosis
- Statistical arbitrage
- Predictive analytics
- Extraction

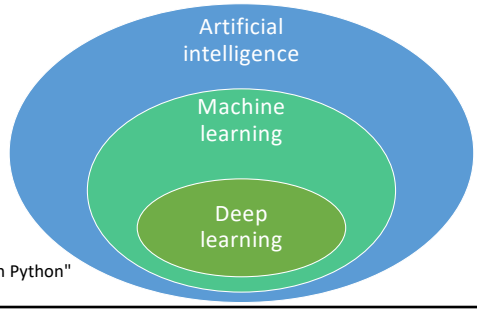


https://royalsociety.org/topics-policy/projects/machine-learning/videos-and-background-information/

3

AI vs ML vs DL?

- **AI**: the effort to automate intellectual tasks normally performed by humans.
  - AI is a general field that encompasses machine learning and deep learning, but that also includes many more approaches that don't involve any learning.
- **ML**: allows a computer go beyond automating intellectual tasks and can learn on its own how to perform a specified task. In ML, a computer can automatically learn rules by looking at data.

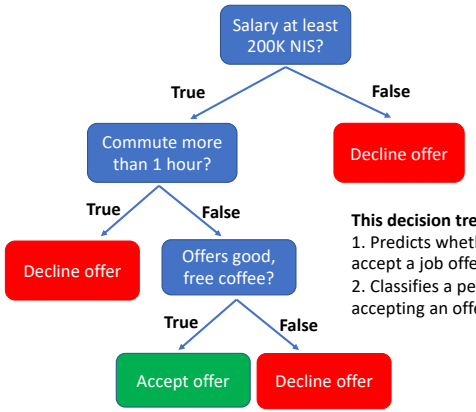


"Deep learning with Python"
By Francois Chollet

4

Some Simple ML Examples

- A decision tree (should I accept job offer):

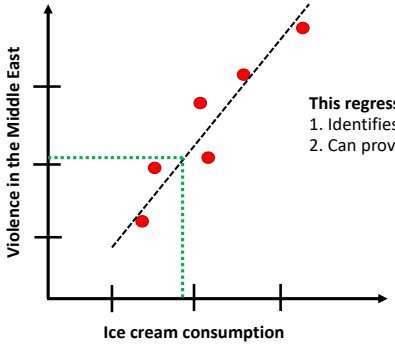


This decision tree:
1. Predicts whether a person will accept a job offer or not
2. Classifies a person as either accepting an offer or not

5

Some Simple ML Examples

- A regression analysis:

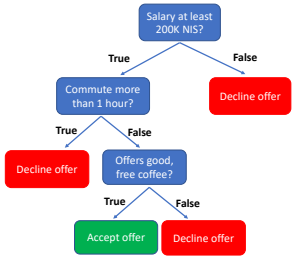


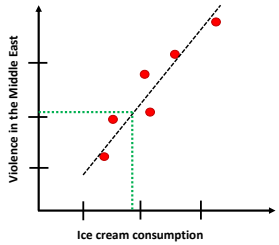
This regression analysis:
1. Identifies a trend
2. Can provide predictions

6

Some Simple ML Examples

- ML deals with making predictions and classifications

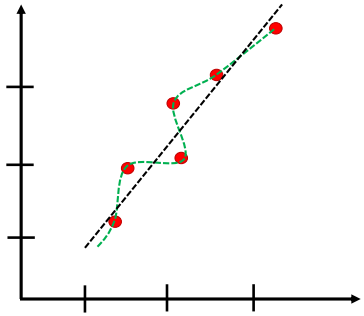




7

Some Simple ML Concepts

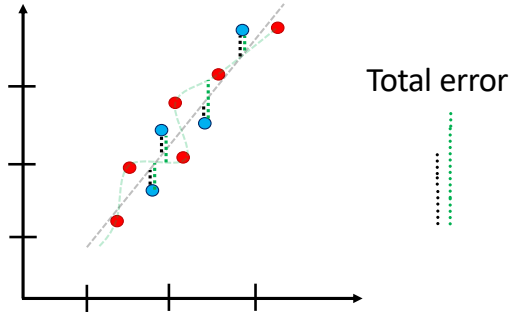
- Training data



8

Some Simple ML Concepts

- Testing data



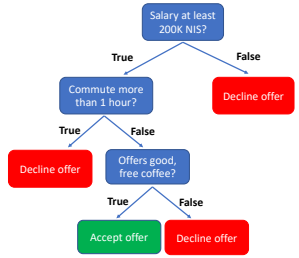
It's important not to be fooled by the good fit of training data.
Beware of the **bias-variance trade-off**!

9

Some Simple ML Examples

- Training data

	Salary	Commute	Coffee	Accept
1	True	False	True	True
2	False			False
3	True	False	False	False
...				

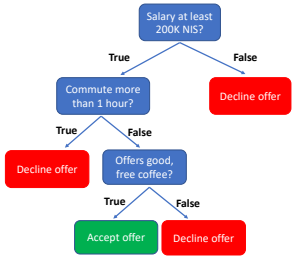


10

Some Simple ML Examples

- Testing data

	Salary	Commute	Coffee	Accept
1	True	True		False
2	True	False	True	True
3	True	False	True	False
...				



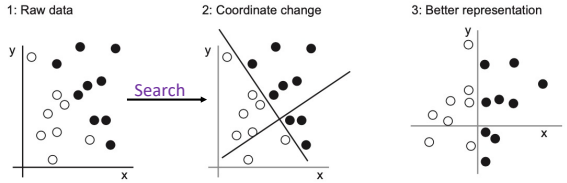
11

What is ML?

12

AI vs ML vs DL?

- An **ML** example: Let's say we want to develop an algorithm that can take the coordinates (x, y) of a point and output whether that point is likely to be black or to be white:
  - The inputs are the coordinates of our points.
  - The expected outputs are the colors of our points.
  - A measure whether our algorithm is doing a good job (e.g., % success).

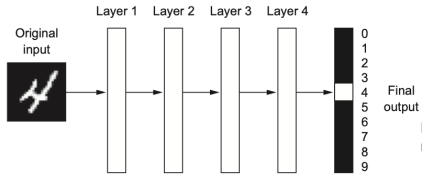


- Here, the black/white classification problem can be expressed as a simple rule: "Black points are such that x > 0" or "White points are such that x < 0".

13

AI vs ML vs DL?

- **ML** is also called shallow learning as it only involves 1-2 layers of learning.
- **DL** entails learning successive *layers* of increasingly meaningful representations.
  - The *deep* in *deep learning* isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for successive layers of representations. How many layers contribute to a model of the data is called the *depth* of the model.
  - DL usually uses neural networks as the layered architecture.



Network several layers deep transforms an image of a digit in order to recognize what the digit it is

- DL is a multistage way to learn data representations. It's a simple idea that when sufficiently scaled, can end up looking like magic.

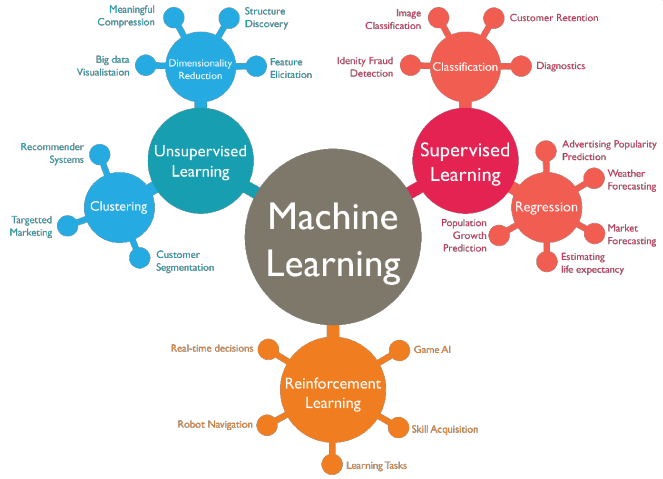
14

What is ML?

- Machine learning is a sub-field of computer science, giving a computer the ability to learn without being explicitly programmed.
- ML teaches computers to perform certain tasks using **data**, including **predictive analytics**.
- We can separate these learning problems into two broad categories – **supervised** and **unsupervised** learning:
  - In **supervised** learning, we have variables that can be divided into **input** variables (aka **features**) and **output** variable (aka the **target**).
    - The goal is to find an algorithm that learns the relationship between inputs and outputs so that when it is given new inputs, it can **predict** outputs.
  - In **unsupervised learning**, we only have **input** variables.
    - The goal is to find the underlying structure or distribution to draw inferences.

15

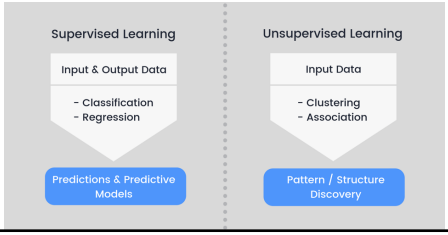
What is ML?



16

Supervised vs. Unsupervised ML

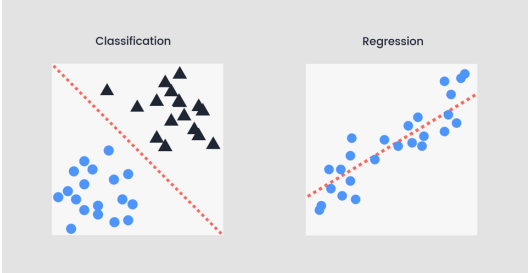
- In **supervised** learning, we have variables that can be divided into **input** variables (aka **features**) and **output** variable (aka the **target**).
  - The goal is to find an algorithm that learns the relationship between inputs and outputs so that when it is given new inputs, it can **predict** outputs.
- In **unsupervised learning**, we only have **input** variables.
  - The goal is to find the underlying structure or distribution to draw inferences.



17

Supervised ML

- Supervised learning problems can be divided into **regression** and **classification**:
  - If the **target** variable is **numerical**, then it is a **regression** problem.
  - If the **target** variable is **categorical**, then it is a **classification** problem.



18

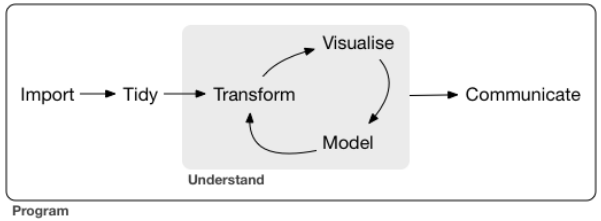
Unsupervised ML

- Unsupervised learning problems can be divided into **clustering** and **dimensionality reduction**:
  - Clustering assigns objects to similar groups (called clusters) while making sure that objects in different groups are not similar.
  - Dimensionality reduction reduces the number of features used to represent objects.

19

The ML Process

- First you must **import** your data into Python. This typically means that you take data stored in a file, database, or web API, and load it into a DataFrame in Python



Hadley Wickham's R for Data Science – Import: https://r4ds.had.co.nz/introduction.html

20

Data I/O

- Flat Files
  - .txt, .csv
- Files from Other Software
  - Excel, Matlab, SAS, Stata
- Relational Databases

21

txt Data I/O

- Working with flat files
  - Reading and writing text files
- File Extensions
  - .txt – Text file
  - .csv - Comma separated values
- Delimiters
  - Commas
  - Tabs
- Import flat files using NumPy
  - NumPy arrays are standard for storing numerical data
  - Essential for other packages, like scikit-learn
  - NumPy commands include:
    - `loadtxt()`, `genfromtxt()`

22

CSV and Excel Data I/O

- Import and export CSV files using Pandas

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

- Import and export Excel files using Pandas

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

- Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xlsx')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

23

MATLAB files I/O

- Matlab – "Matrix Laboratory"
- Industry standard in engineering and science
- Data is saved as .mat files
- `scipy.io.loadmat()` - read .mat files
- `scipy.io.savemat()` - write .mat files

```
>>> import scipy.io
>>> filename = 'workspace.mat'
>>> mat = scipy.io.loadmat(filename)
>>> print(type(mat))
<class 'dict'>
```

- keys = MATLAB variable names
- values = objects assigned to variables

```
>>> print(type(mat['x']))
<class 'numpy.ndarray'>
```

24

Database systems

- Relational Database Management Systems (RDBMS)
  - PostgreSQL
  - MySQL
  - SQLite
  - SQL = Structured Query Language

- We won't use RDBMBS in this course

25

Tidy Data Cleaning and Wrangling

- A significant amount of time is spent on **data preparation**: loading, cleaning, transforming, and rearranging.
- Such tasks are often reported to take up **80% or more** of an analyst's time.
- Sometimes the way that data is stored in files or databases **is not in the right format** for a particular task.
- Many researchers choose to do ***ad hoc* processing of data** from one form to another using a general-purpose programming language, like Python, Perl, R, or Java, or Unix text-processing tools like sed or awk.
- Pandas, along with the built-in Python language features, provides a high-level, flexible, and fast set of tools to enable one to manipulate data into the right form.

26

Tidy and Transform Data Cleaning and Wrangling

- Data Cleaning and Preparation
  - Handling Missing Data
  - Data Transformation
  - String Manipulation
- Data Wrangling: Join, Combine, and Reshape
  - Hierarchical Indexing
  - Combining and Merging Datasets
  - Reshaping and Pivoting

27

Introduction to Scikit-learn

- This project was started in 2007 as a Google Summer of Code project
- First public release in February 2010
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable

https://scikit-learn.org/stable/
Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

28

Introduction to Scikit-learn

- Data is represented by a two-dimensional grid, in which *the rows represent individual elements of the dataset*, and *the columns represent quantities related to each of these elements*.

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

- Here each row of the data refers to a single observed flower, and the number of rows is the total number of flowers in the dataset. In general, we will refer to the rows of the matrix as *samples*, and the number of rows as `n_samples`.
- Likewise, each column of the data refers to a particular quantitative piece of information that describes each sample. In general, we will refer to the columns of the matrix as *features*, and the number of columns as `n_features`.

29

Scikit-Learn – Data Representation – Features Matrix

- The **features matrix** is two-dimensional, with shape **`[n_samples, n_features]`**, and is most often contained in a NumPy array or a Pandas DataFrame, though some Scikit-Learn models also accept SciPy sparse matrices.
- The **samples** (i.e., rows) always refer to the individual objects described by the dataset. For example, the sample might be a flower, a person, a document, an image, a sound file, a video, an astronomical object, or anything else you can describe with a set of quantitative measurements.
- The **features** (i.e., columns) refer to the distinct observations that describe each sample in a quantitative manner. Features are generally real-valued, but may be Boolean or discrete-valued in some cases.
- By convention, the **features matrix** is often stored in a variable named **X**.

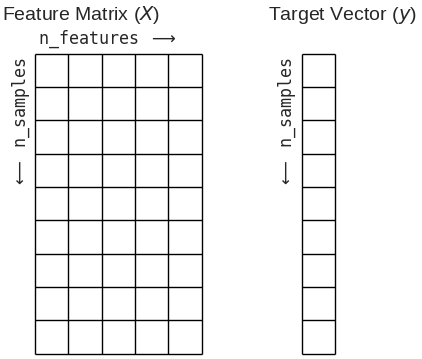
30

Scikit-Learn – Data Representation – Target Vector

- The **target array (vector)** is usually one dimensional, with length **`n_samples`**, and is generally contained in a NumPy array or Pandas Series.
- The **target array** may have continuous numerical values, or discrete classes/labels. (While some Scikit-Learn estimators do handle multiple target values in the form of a two-dimensional, `[n_samples, n_targets]` target array, we will primarily be working with the common case of a one-dimensional target array.)
- By convention, the **target vector** is often stored in a variable named **y**.

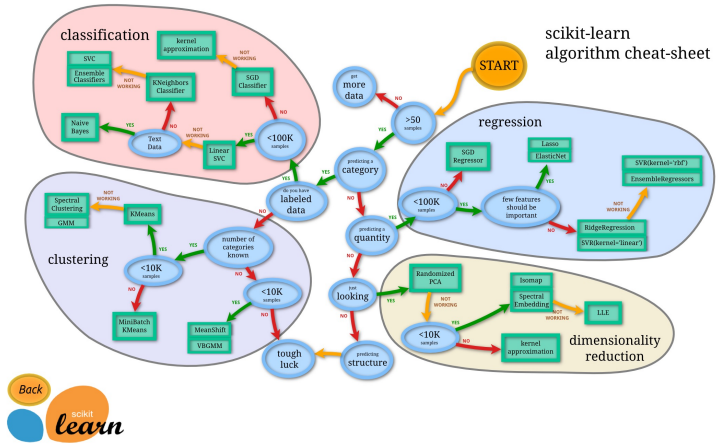
31

Scikit-Learn – Data Representation – Features Matrix & Target Vector



32

Scikit-Learn Algorithm Cheat-sheet



33

Scikit-Learn – Guiding Principles

The Scikit-Learn API is designed with the following guiding principles in mind, as outlined in the Scikit-Learn API paper:

- *Consistency*: All objects share a common interface drawn from a limited set of methods, with consistent documentation.
- *Inspection*: All specified parameter values are exposed as public attributes.
- *Limited object hierarchy*: Only algorithms are represented by Python classes; datasets are represented in standard formats (NumPy arrays, Pandas DataFrames, SciPy sparse matrices) and parameter names use standard Python strings.
- *Composition*: Many machine learning tasks can be expressed as sequences of more fundamental algorithms, and Scikit-Learn makes use of this wherever possible.
- *Sensible defaults*: When models require user-specified parameters, the library defines an appropriate default value.

https://scikit-learn.org/stable/

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

34

Scikit-Learn – Basics of the API

- The steps in using the Scikit-Learn API are as follows:

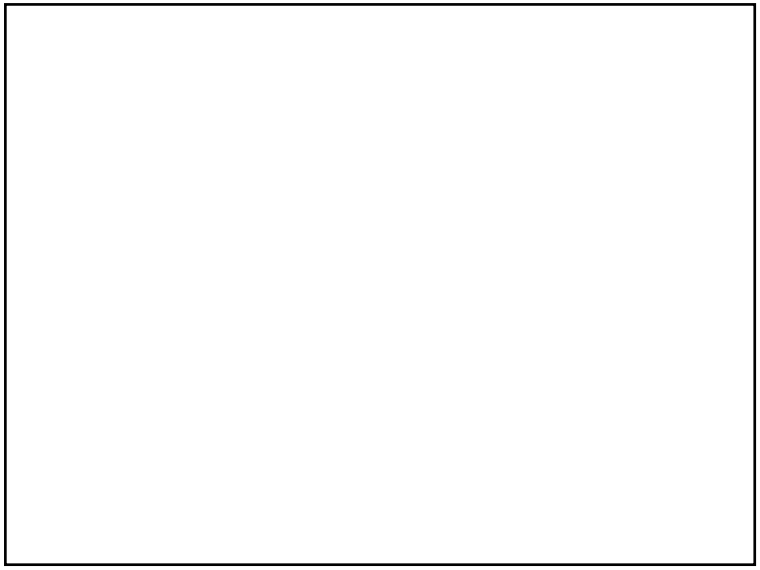
1. Choose a class of model by importing the appropriate estimator class from Scikit-Learn.
2. Choose model hyperparameters by instantiating this class with desired values.
3. Arrange data into a **features matrix** and **target vector** following the discussion above.
4. Fit the model to your data by calling the **`fit()`** method of the model instance.
5. Apply the Model to new data:
   1. For **supervised** learning, often we predict labels and values for unknown data using the **`predict()`** method.
   2. For **unsupervised** learning, we often transform or infer properties of the data using the **`transform()`** or **`predict()`** method.

35

Scikit-Learn – Basics of the API

```
>>> from sklearn import datasets
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.ensemble import RandomForestClassifier
```

36



37