Introduction to Scikit-Learn: Machine Learning with Python

Machine Learning Intro

Tony Yao-Jen Kuo

About Scikit-Learn

Scikit-Learn (http://scikit-learn.org/stable/) is a Python package designed to give access to **well-known** machine learning algorithms within Python code, through a **clean, well-thought-out API**. It has been built by hundreds of contributors from around the world, and is used across industry and academia.

Scikit-Learn is built upon Python's NumPy (Numerical Python) (http://numpy.org) and SciPy (Scientific Python) (http://scipy.org) libraries, which enable efficient in-core numerical and scientific computation within Python.

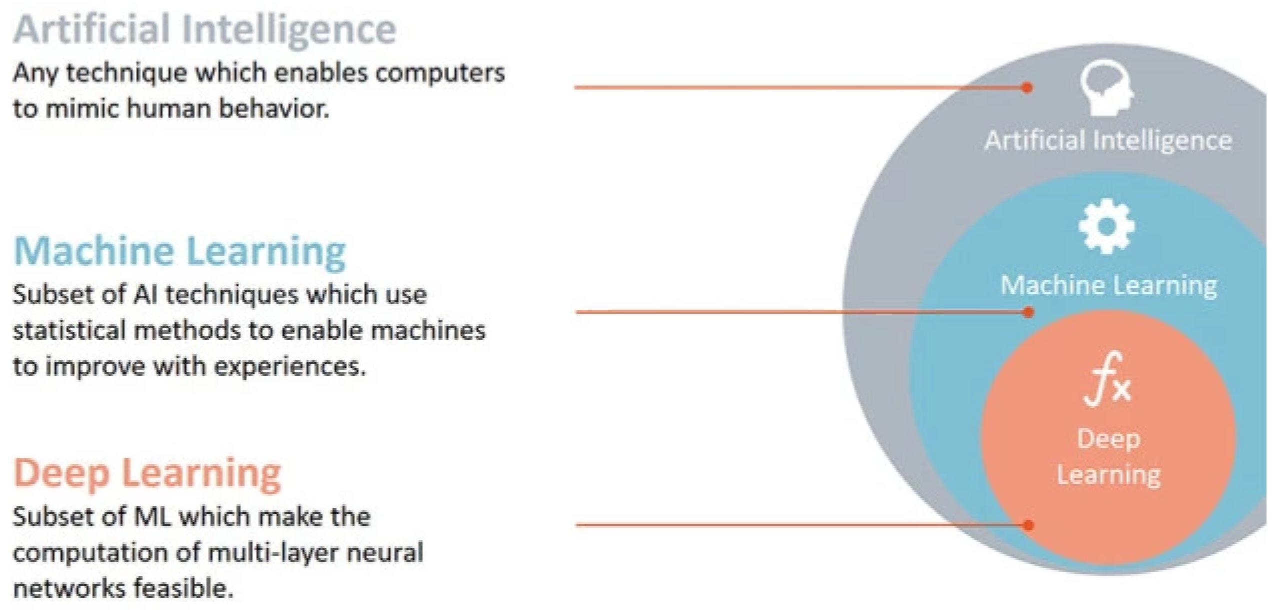
What is Machine Learning?

Machine Learning is about building programs with **tunable parameters** (typically an array of floating point values) that are adjusted automatically so as to improve their behavior by **adapting to previously seen data.**

Tom Mitchell (http://www.cs.cmu.edu/~tom/)

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P,*
> *if its performance at tasks in T, as measured by P, improves with experience E.*

Machine Learning can be considered a subfield of **Artificial Intelligence** since those algorithms can be seen as building blocks to make computers learn to behave more intelligently by somehow **generalizing** rather that just storing and retrieving data items like a database system would do.



Artificial Intelligence
Any technique which enables computers to mimic human behavior.

Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.

Source: rapidminer (https://rapidminer.com/artificial-intelligence-machine-learning-deep-learning/)

Machine Learning Topics

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Dimensionality Reduction
  - Clustering

Representation of Data in Scikit-learn

Machine learning is about

- Creating models from data
- How data can be represented in order to be understood by the computer?

Two-dimensional array or matrix

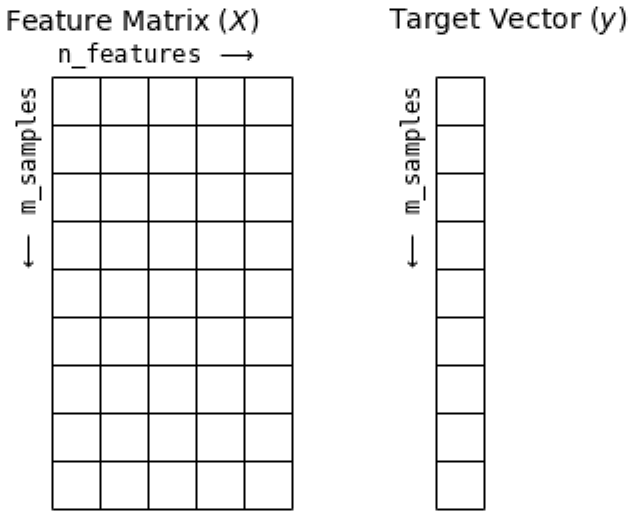
Most machine learning algorithms implemented in scikit-learn expect data to be stored in a **two-dimensional array or matrix**. The size of the array is expected to be `[m_samples, n_features]`

m x n in general

- **m_samples:** The number of samples: each sample is an item to process (e.g. classify). A sample can be a document, a picture, a sound, a video, an astronomical object, a row in database or CSV file, or whatever you can describe with a fixed set of quantitative traits.
- **n_features:** The number of features or distinct traits that can be used to describe each item in a quantitative manner. Features are generally real-valued, but may be boolean or discrete-valued in some cases.

In [2]:

```
# Figure from the Python Data Science Handbook
plt.show()
```



Simple Examples: Getting Started with Kaggle

We're going to take a look at the data hosted by Kaggle. Try extracting feature matrix and target vector from the following datasets:

House Prices: Advanced Regression Techniques

https://www.kaggle.com/c/house-prices-advanced-regression-techniques (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)

In [3]:

```
# Target Vector: SalePrice

train_url = "https://storage.googleapis.com/kaggle_datasets/House-Prices-Advanced-Regression-Techniques/train.csv"
```

Titanic: Machine Learning from Disaster

https://www.kaggle.com/c/titanic (https://www.kaggle.com/c/titanic)

In [4]:

```
# Target Vector: Survived

train_url = "https://storage.googleapis.com/kaggle_datasets/Titanic-Machine-Learning-from-Disaster/train.csv"
```

Digit Recognizer

https://www.kaggle.com/c/digit-recognizer (https://www.kaggle.com/c/digit-recognizer)

In [5]:

```
# Target Vector: label

train_url = "https://storage.googleapis.com/kaggle_datasets/Digit-Recognizer/train.csv"
```

The Scikit-learn Estimator Object

Every algorithm is exposed in scikit-learn via an ''Estimator'' object.

In [6]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Estimator parameters: All the parameters of an estimator can be set when it is instantiated, and have suitable default values:

In [7]:

```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()
print(model)
```

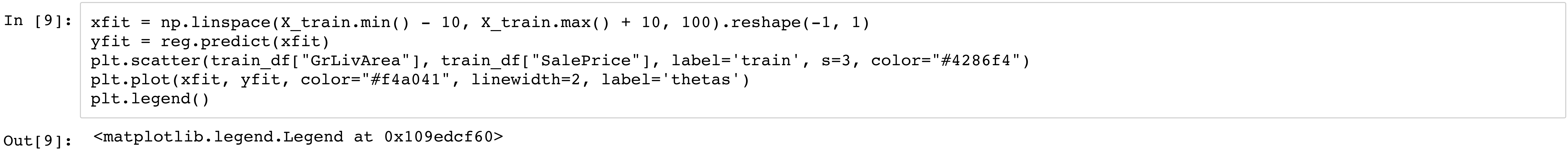
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Estimated Model parameters: When data is fit with an estimator, parameters are estimated from the data at hand. All the estimated parameters are attributes of the estimator object ending by an underscore:

In [8]:

```
train_url = "https://storage.googleapis.com/kaggle_datasets/House-Prices-Advanced-Regression-Techniques/train.csv"
train_df = pd.read_csv(train_url)
X_train = train_df["GrLivArea"].values.reshape(-1, 1)
y_train = train_df["SalePrice"].values.reshape(-1, 1)
reg = LinearRegression()
reg.fit(X_train, y_train)
print(reg.intercept_)
print(reg.coef_)
```

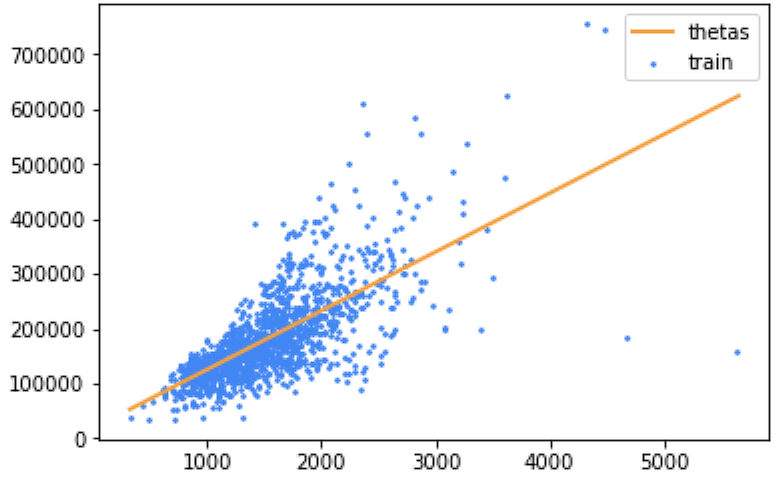
```
[ 18569.02585649]
[[ 107.13035897]]
```



```
In [9]: xfit = np.linspace(X_train.min() - 10, X_train.max() + 10, 100).reshape(-1, 1)
        yfit = reg.predict(xfit)
        plt.scatter(train_df["GrLivArea"], train_df["SalePrice"], label='train', s=3, color="#4286f4")
        plt.plot(xfit, yfit, color="#f4a041", linewidth=2, label='thetas')
        plt.legend()

Out[9]: <matplotlib.legend.Legend at 0x109edcf60>
```

```
In [10]: plt.show()
```


thetas
train