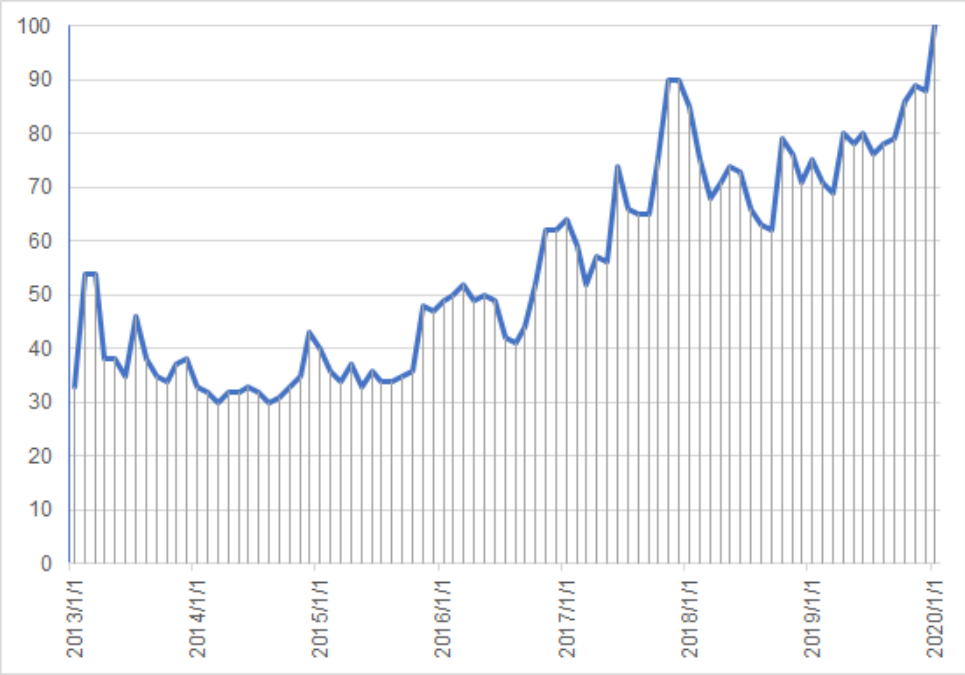
機械学習入門

Kronos

目次

1 AI



これは「AI」というトピックでのGoogle Trendsの結果です。2014年から徐々に増加しはじめて、2016年から2017年に掛けて増加の勢いが増しているのが見て取れます。

現在のAIブームは第3次AIブームと言われています。過去2度のブームは大きな社会の流れにはならずに終焉しました。しかし、今回ブームはこの傾向はまだ暫く続くだけでなく、社会を変革するトレンドになると多くのシンクタンクやメディア、政府、アナリストが予想しています。

それでは、これまでのブームと現在のAIは一体何が違うのでしょうか？現在のAIでは一体何が出来るようになったのでしょうか？

この講座ではハンズオンを交えながら技術的側面から、AIの現状を理解し、実践できるようになることを目標としています。AIは今後も変化・進歩の激しい分野ではありますが、本講座の学習内容が、AIの理解の第一歩となればと思っています。

時代	AIの特徴
第1次AIブーム 1950～1960年代	探索・推論アルゴリズムの時代 迷路やチェスのようなゲーム攻略（トイ・プロブレム）が中心
第2次AIブーム 1980年代	エキスパートシステムの時代 専門家の知識をAIに登録する
第3次AIブーム 2010年頃～	ディープラーニングによるブレイクスルー（子どものAI） 画像認識や音声認識技術の大幅な精度の向上

1.1 AIとは

AI（=Artificial Intelligence=人工知能）の定義については専門家によっても意見が分かれます。本講座では、AIは人間の知能を表現したソフトウェアとします。また知能とは、学習し抽象的な思考をし、環境に適応するもととなる能力、と考えることにします。それから本講座ではAIの開発手法として機械学習を使うものとします。

1.2 AIの分類

AIは非常に広い概念で、様々な分類・区分けがあります。その一つが「**汎用AI**」と「**特化型AI**」です。

汎用AI

人間のように、あらゆる状況を判断することの出来るAIを言います。人間と同等の学習、推論、反応を示すことを目標としています。汎用AIの研究は進んでいますが、実用化にはまだまだ時間のかかる研究分野です。

特化型AI

特定の課題を解決する目的で、学習、推論が出来るAIを言いいます。対象とする課題においては人間以上の成果を示すこともあります。現在ビジネスの場で活用されているAIはこの特化型AIを指します。本講座で学習するAIは、この特化型AIに分類されます。

大人のAI、子どものAI

> ➢ 「コンピュータ（AI）はチェスに強くても、花を見て花と認識したり、ものをつかんだり、といった、子どもにできることができない。」

これはモラベックのパラドックスと呼ばれるもので、従来のAIの得意なこと、苦手なことをうまく言い表しています。実は、現在のAIは花を見て花と認識することや、ものをつかむ、という動作を実現できる段階にあります。それでは従来のAIと現在のAIの違いとは一体何なのでしょうか。

AIを分類するときに大人のAI、子どものAIといった分け方があります。**大人のAI**とは人間が作り込んだAIを指します。これは人が設計したルールをコンピュータにプログラミングしたAIのことです。

一方の**子どものAI**は、コンピュータが子どものように自分でルールを学習していくプログラムです。子どものAIでは、AIの開発者が複雑なルールを考え、設計する必要はありません。膨大なデータからコンピュータに学習させて、コンピュータ自身に法則性を見つけさせるのです。

子どものAIを実現する手法として注目を集めているのがディープラーニングです。AIの専門家でなくても、誰もがAIを開発できる時代が来ているのです。

1.3 AIとプログラム

AIも結局はプログラムであることに変わりはありませんが、AIと呼ばれるプログラムと一般的なプログラムには違いがあります。

たとえば次の画像ファイルについて考えてみましょう。

上記のような画像ファイルを処理するプログラムは次のようなものがあるでしょう。

- 画像サイズ（400, 400）を（200, 200）に変更する
- 画像をグレースケールに変更する
- 200,200の位置のピクセルのカラーを取得する

これらの処理は従来のプログラムで実装可能です。画像処理の経験のあるプログラマーであれば実装方法が想像できるでしょう。

それでは以下の処理はどうでしょうか。

- この花はアサガオかどうか判定する
- この花は美しいかどうか判定する
- この花が枯れるのはいつか予測する

プログラマーであっても「アサガオかどうか判定する方法」が具体的ではないため、実装方法に戸惑うでしょう。同様に「美しい花」の定義が曖昧ですし、「いつ枯れるのか」のいう予測も、花の開き具合から判断するのか、そもそも花の開き具合を判定する方法はどのように測るのか、と考えていくと簡単ではないと気づくでしょう。

現在のAIが得意な分野はここにあります。AIの中にはルールがあります。たとえば囲碁を打つAIであれば、次の一手を決定するのがルールです。このルールを人が設計するのか、コンピュータ自身が設計するのか、ここに大きな違いがあります。現在話題になる多くのAIは機械学習と呼ばれる手法を用いて、コンピュータがデータからルールを設計します。このルールのことをモデルと呼ぶこともあります。

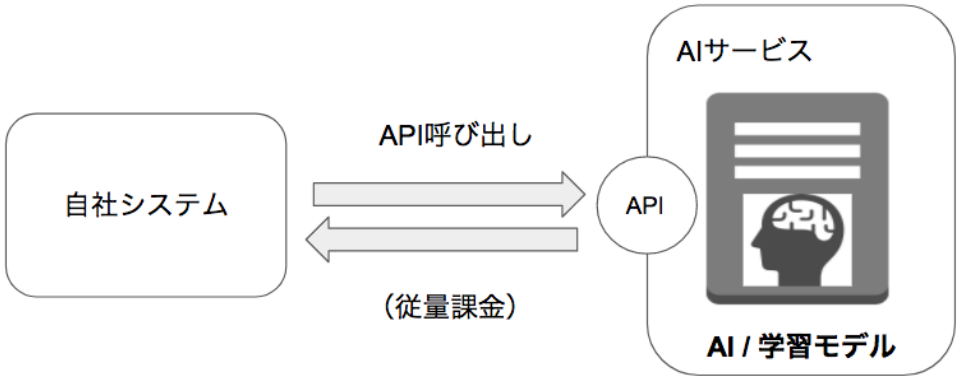
1.4 AIの活用方法

少し視点を変えて、現在のビジネスシーンにおいてAIを活用する方法を整理してみましょう。大きく2つにわけることができるでしょう。

- AIサービスの活用
- 機械学習ライブラリによる開発

AIサービスの活用

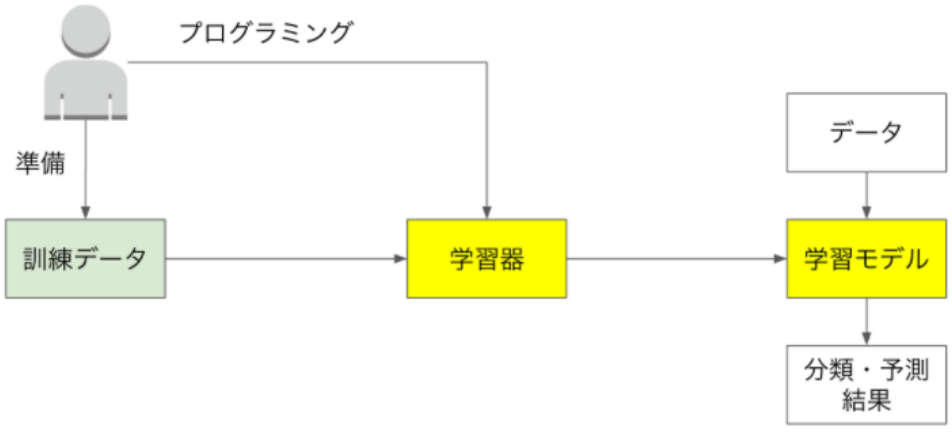
Googleのクラウドプラットフォーム（GCP）やIBMのコグニティブサービスWatsonでは、画像認識や音声認識、自然言語処理といった機能がサービスとして公開されています。これらのサービスは専用のアカウントを作成すれば、すぐに活用することができます。



機械学習ライブラリによる開発

Python言語を中心にTensorFlowなどの機械学習ライブラリが充実しています。これらのライブラリを活用することで自社のビジネスに特化したAIを開発することも可能です。

本講座では機械学習ライブラリの活用方法を紹介します。



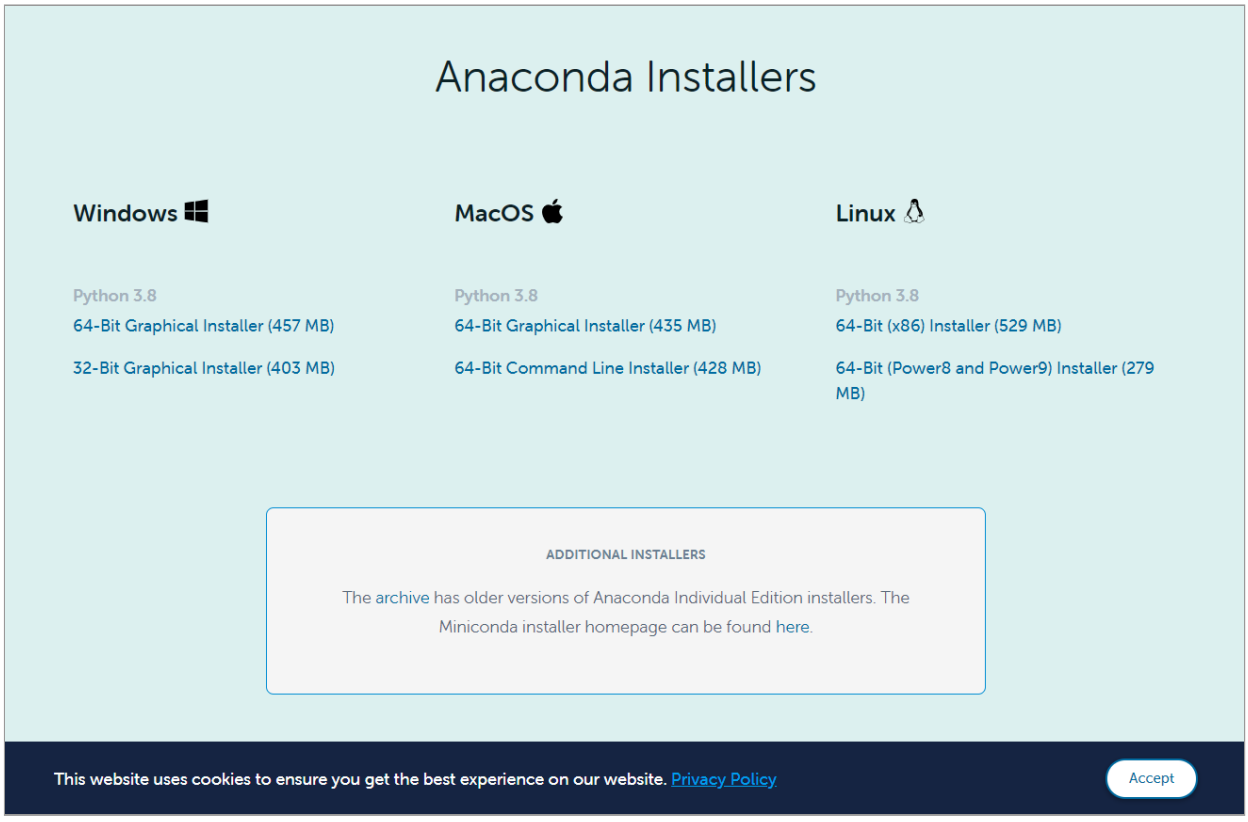
2 AI開発環境の構築

2.1 Anacondaディストリビューション

Pythonで機械学習を始めるにはAnaconda社（旧名Continuum Analytics）が配布しているAnacondaディストリビューションを活用すると良いでしょう。以下のURLからダウンロードすることができます。

https://www.anaconda.com/distribution/

AnacondaにはPython本体だけでなく、機械学習に必要なライブラリも梱包されているため、すぐに機械学習プログラミングを始めることができます。

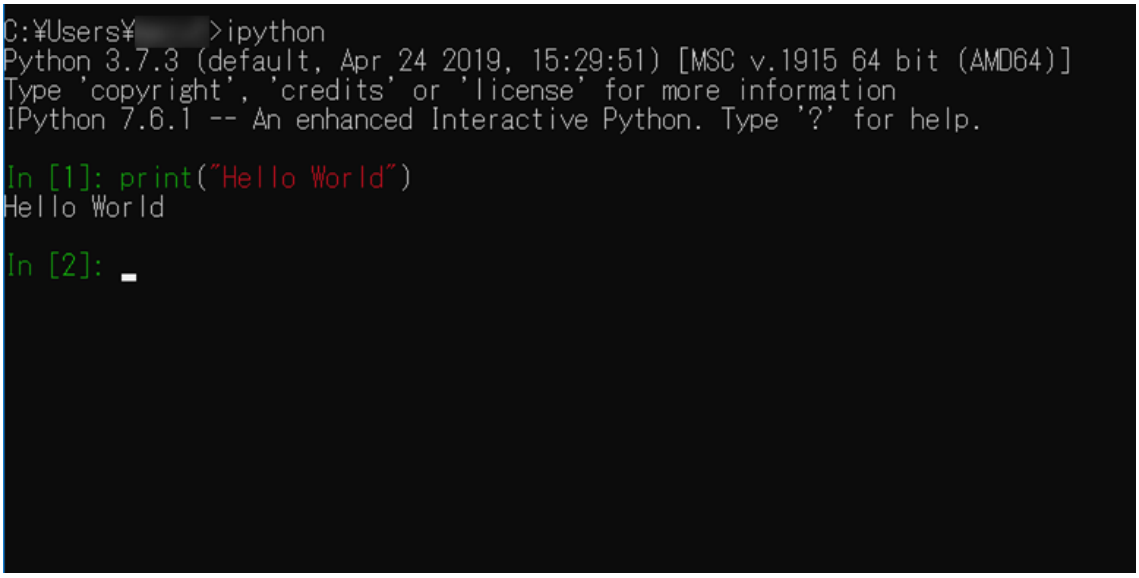


➢ 本講座のプログラムは Python 3.7 :: Anaconda 2019.03 で動作確認しています。

2.2 Python実行環境

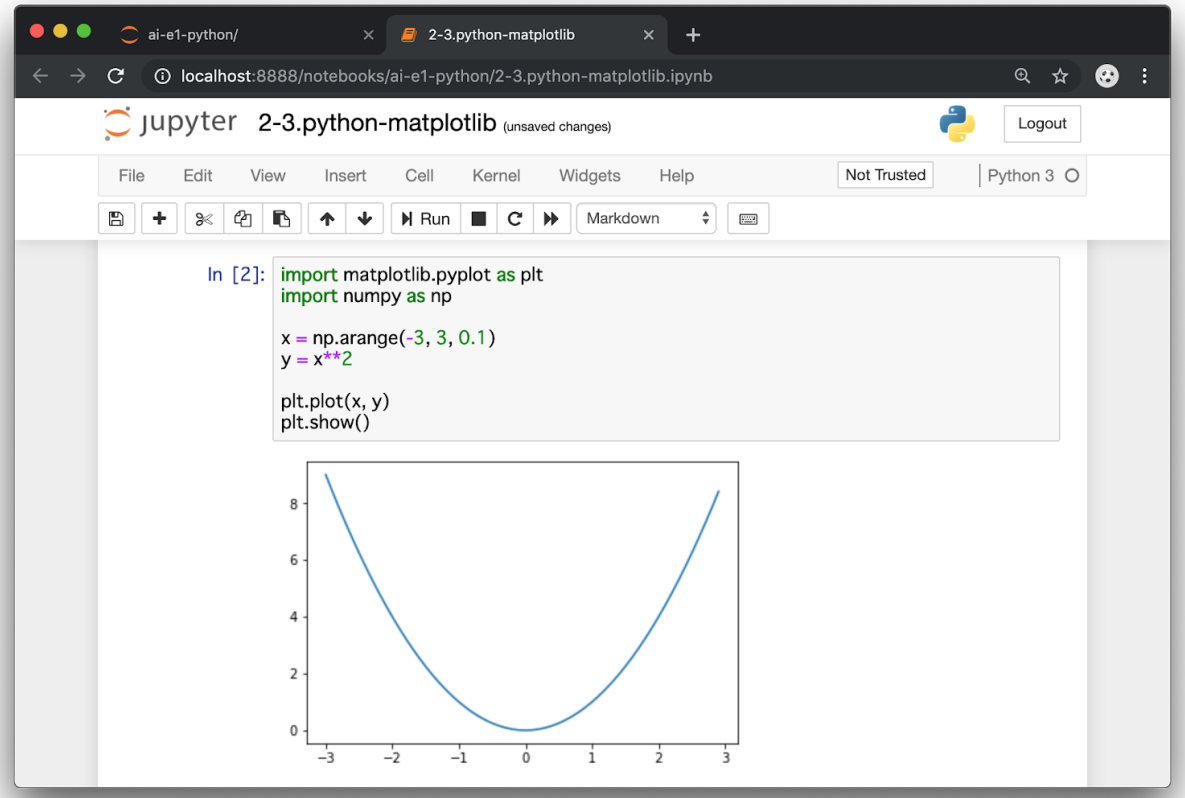
IPython

AnacondaにはIPythonというインタラクティブなPythonツールが含まれています。ターミナル（コマンドプロンプト）上で ipython コマンドを実行すると対話形式でPythonプログラムを実行できます。
標準の python コマンドも対話形式でプログラムを実行できますが、IPython はコードハイライトやコード補完機能が強化されています。



Jupyter Notebook

AnacondaにはJupyter Notebookというブラウザ上で動作するPython開発環境も付属しています。ターミナル（コマンドプロンプト）上で jupyter-notebook コマンドを実行するとローカルでサーバプログラムが起動し、ブラウザが起動します。



➢ デフォルトでは8888番ポートで起動します。

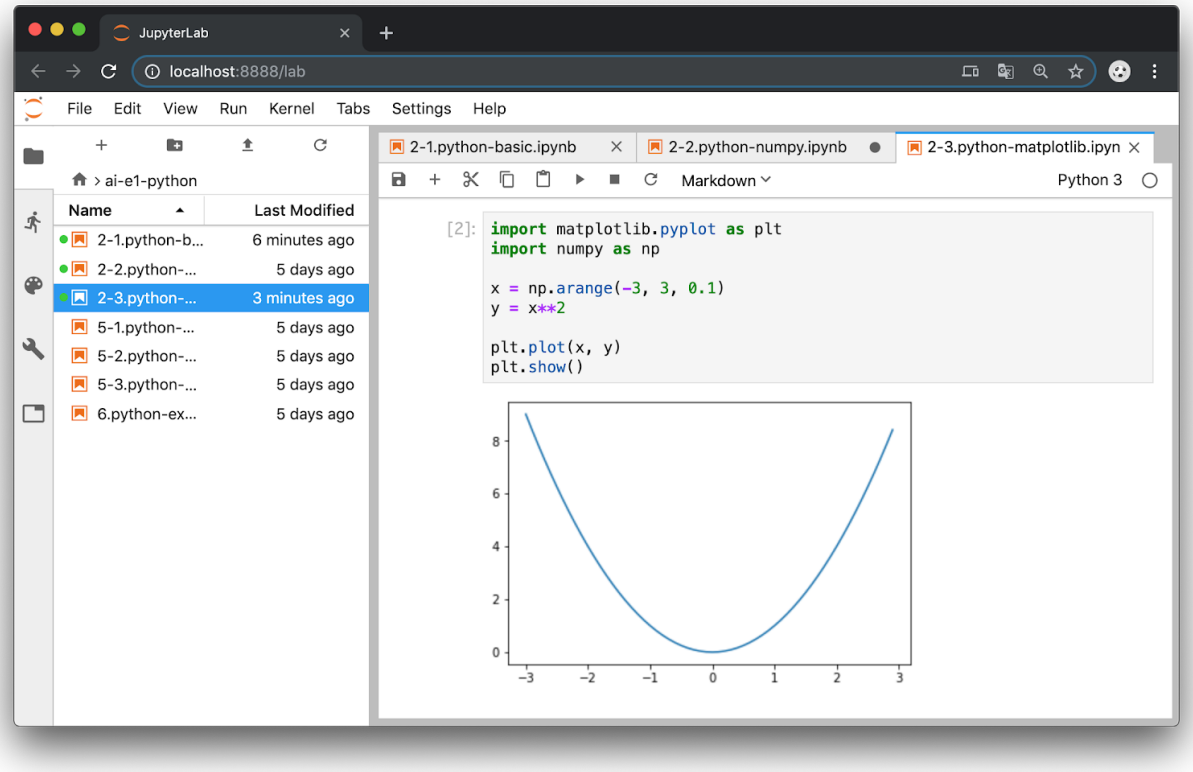
Jupyter Notebookでは、ノートブックという形式でファイルを作成できます。ノートブックにはPythonコードや実行結果だけでなく、Markdown形式で文章を挿入することもできるので、データ分析の評価レポートを作成しやすいようになっています。

➢ ノートブックファイルは拡張子.ipynbという形式保存されます。

Jupyter Notebookの実体はブラウザで利用するWebアプリケーションです。そのためサーバ上でJupyter Notebookを起動すれば、手元のパソコンのブラウザからサーバ上でPythonプログラムを実行することも可能です。複数の利用者がデータ解析を共有する場合にも便利でしょう。

Jupyter Lab

前述のJupyter Notebookの後継として登場したのがJupyter Labです。Jupyter Notebookと同様に、ブラウザ上で動作するPython開発環境で、jupyter-lab コマンドの実行でブラウザが起動します。
大きな変更点は画面構成です。Jupyter Notebookではディレクトリ画面とPython実行画面が別ページでしたが、Jupyter Labでは1つのページにまとめられました。ノートブックファイルを複数開いた場合もマルチタブで表示されるようになっています。



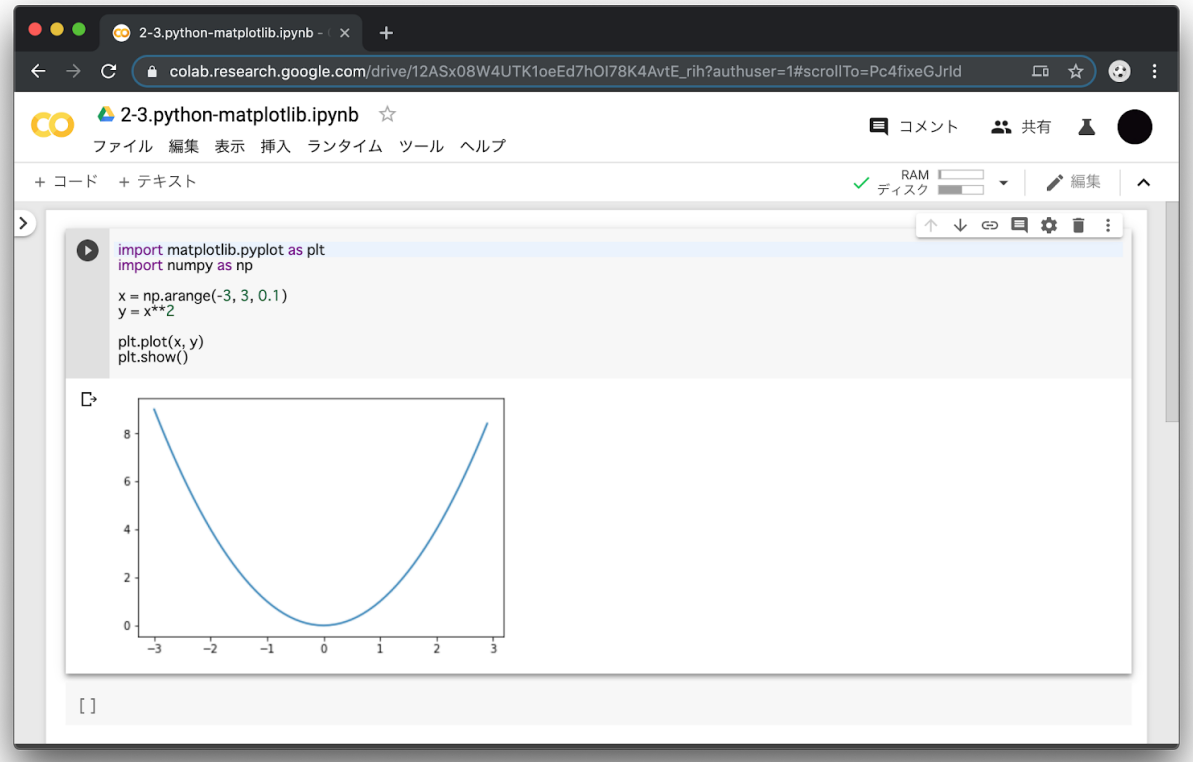
➢ デフォルトのポート番号やノートブックファイルの拡張子などはJupyter Notebookを踏襲しています。

Google Colaboratory

Googleは機械学習の教育や研究の促進を目的とした機械学習プラットフォームとしてColaboratoryを提供しています。

https://colab.research.google.com/

ColaboratoryはJupyter NotebookをベースにしたWebアプリケーションです。Googleアカウントがあればブラウザから上記のURLにアクセスすることで機械学習プログラミングをすぐに始めることができます。
Colaboratoryで作成したノートブックやアップロードしたファイルなどは、Googleドライブ上に保持されます。



3 機械学習

現在のAIブームを牽引するキーワードが機械学習です。機械学習は膨大なデータから規則性（モデル）を見つけ出し、コンピューターに学習させる技術です。そのため数学の統計・分析の手法が利用されます。

3.1 機械学習アルゴリズム

機械学習には様々な手法があります。

- 線形回帰
- ロジスティック回帰
- 決定木
- SVM（サポートベクターマシン）
- k-NN法（K近傍法）
- k-Means法（K平均法）
- ニューラルネットワーク

以前はこれらの機械学習を行うにはアルゴリズムを理解し、自分でプログラミングする必要がありました。そのため、高度な数学の知識、統計の知識とプログラミングのスキルが必要で、ごく限られた研究者以外には活用することができませんでした。

近年、プログラミング言語Pythonを中心に機械学習ライブラリの充実しています。機械学習の手法、特徴や使い方を理解して、プログラミングができれば簡単に機械学習に取り組むことができるようになっています。このような外部環境の変化も近年のAI・機械学習の爆発的な発展の一つの要因であると言えます。

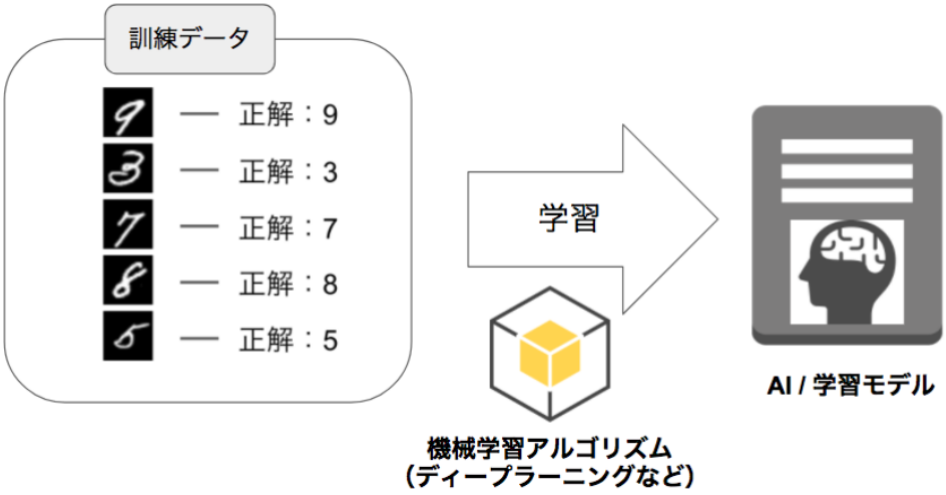
> ➢ ハードウェアの性能向上やインターネット、クラウドの登場もAI・機械学習ブームを支える一つの要因です。

3.2 教師あり学習と教師なし学習

さきほど紹介した機械学習の手法は学習に用いるデータによって、大きく2種類に分類することができます。
教師あり学習と**教師なし学習**です。

教師あり学習

教師あり学習では、学習する際に対象のデータとそのラベル（答え）を用います。機械学習の多くの場合でこの教師あり学習が使われます。



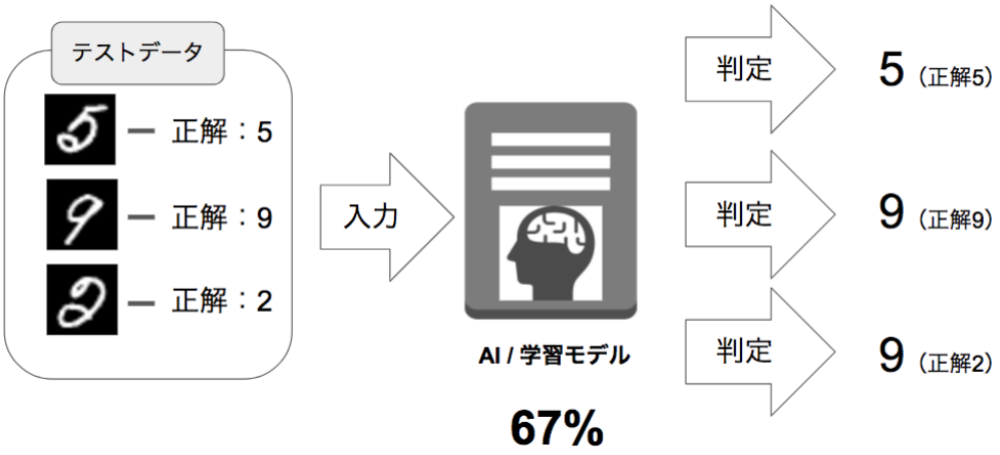
分類と回帰

教師あり学習の主な目的は「**分類（Classification）**」と「**回帰（Regression）**」です。たとえば、手書き数字を識別する場合やスパムメールを識別する機械学習の場合は「分類」を目的とするのに対して、株価の予測や気温の予測といった機械学習の場合は「回帰」を目的とします。これらの違いは、学習済みのモデルが出力する結果が**離散値**であるか、**連続値**であるかの違いです。分類は 0,1,2 のような離散値を出力するのに対して、回帰は23.45 や 29.87 のような連続値を出力します。

分類はさらに「**2値分類**」「**多クラス分類**」のように分けられます。前者はスパムメールのような「スパムメールである」「スパムメールでない」という2つの結果に分類するもので、後者は手書き数字認識のように複数の結果（0～9）に分類されるものです。

訓練データとテストデータ

教師あり学習では、訓練データとテストデータという2つのデータを用意します。訓練データを使ってモデルを訓練した後、テストデータを使って未知のデータに対する汎化性能を検証します。

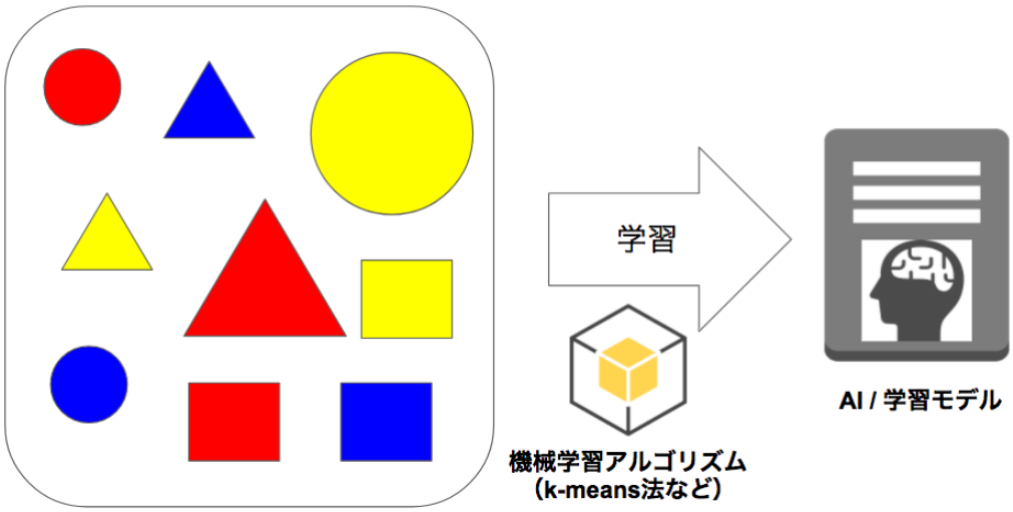


なぜこのようにデータを分けて検証を行うのでしょうか。もし訓練データだけで学習精度を測定した場合、モデルが訓練データの内容をすべて記録してしまえば正答率は100%になってしまいます。そのため、機械学習では既存のデータの学習結果を計測するのではなく、未知のデータへの認識能力を計測します。これは汎化能力などと呼ばれます。

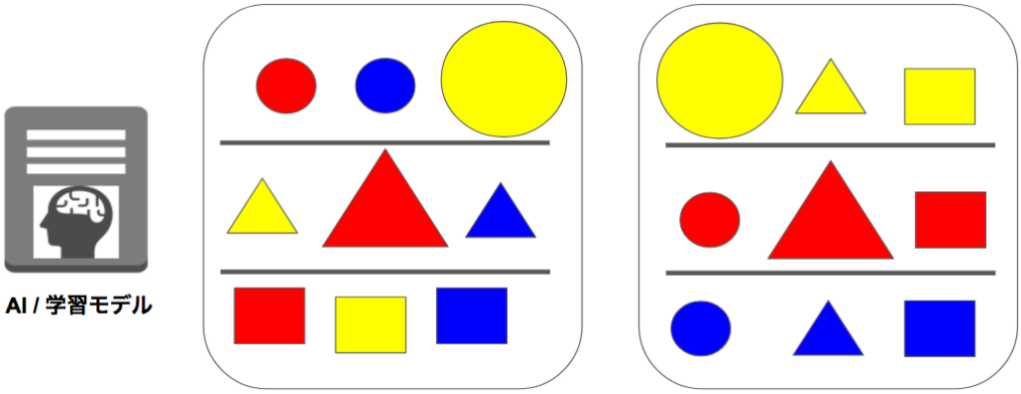
❖ 機械学習では、モデルが訓練データに最適化してしまうことを**オーバーフィッテング（過学習）**と呼びます。

教師なし学習

教師なし学習では学習対象となるデータのみを学習させます。データに正解ラベルはありません。



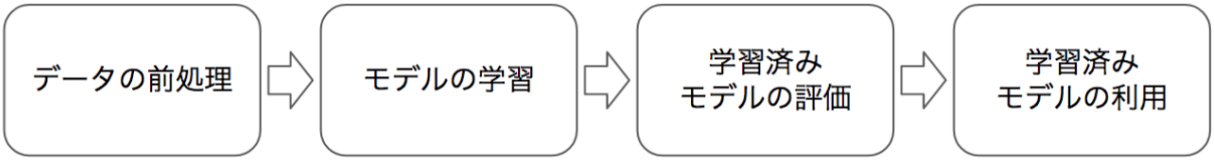
教師なし学習は主に**クラスタリング**に用いられます。



クラスタリングは教師あり学習の分類とよく似ています。クラスタリングは教師なし学習であるためデータに答えはありません。データの傾向からグループを作成する用途に使います。
教師なし学習の手法にはK-means（K平均法）などがあります。

4 機械学習プロセス

機械学習は以下の流れで作業を進めます。



4.1 データの前処理

機械学習に取り組むには学習に利用するデータを準備する必要があります。

1. データの収集
2. 欠損値処理
3. スケーリング処理
4. 訓練データとテストデータの分割

機械学習というとアルゴリズムに注目が集まりがちですが、実際の開発現場ではこれらの前処理に時間がかかるのが普通です。

データの前処理 - 1. データの収集

機械学習を始めるとき、多くの場合において、目的に合わせたデータを収集するところからスタートすることになるでしょう。企業の持つデータベースに対してSQLを発行してデータを収集するケースもあれば、システムのログファイルなどからデータを収集するケースもあるでしょう。

組織内にデータが存在しない場合は、インターネット上の学習データを活用することもできます。たとえば手書き数字データセットであるMNISTは、画像認識プログラムの題材として頻繁に利用されています。このようなインターネット上の共有データセットを使えば簡単に機械学習を始めることができます。

データセット	URL	概要
MNIST	http://yann.lecun.com/exdb/mnist/	手書き数字データセット。60,000枚の訓練データ、10,000枚のテストデータが配布されている。1枚の画像はグレースケールで、サイズは28x28ピクセル。
Iris	https://archive.ics.uci.edu/ml/datasets/iris	アヤメの品種データセット。花びらの縦、横サイズ、萼（がく）の縦、横サイズの4列と品種を含めた5列で構成される。
Boston Housing	http://lib.stat.cmu.edu/datasets/boston	ボストンの住宅価格データセット。住宅の平均部屋数や犯罪発生数、税率といった地域情報と住宅価格の関係を定義したもの。

- インターネット上には他にも機械学習で利用可能な大規模なデータベースも公開されています。スタンフォード大学が運営するImageNetには、インターネット上の画像を集めて分類したデータセットが公開されています。 http://www.image-net.org/

ここではIrisデータセットを例に見てみましょう。Irisデータセットは次の5つの属性(列)で構成されています。

1. sepal length in cm（がくの縦幅）
2. sepal width in cm（がくの横幅）
3. petal length in cm（花びらの縦幅）
4. petal width in cm（花びらの横幅）
5. class:（品種）
    - Iris Setosa
    - Iris Versicolor
    - Iris Virginica

具体的には次のようなファイルです。

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
...省略
```

各レコードの1列目はsepal length、2列目はsepal width、3列目はpetal length、4列目はpetal width、5列目はclassを意味しています。またIrisデータセットは各品種につき50件ずつ、合計150件のレコードで構成されています。

量的データと質的データ

機械学習でデータを扱うときは量的データ、質的データという観点で確認することも大切です。Irisデータセットの各列について見てみましょう。1列目のsepal lengthから4列目はpetal widthまではいずれも数量データとなっています。これらは量的データに分類されます。5列目のclassはIris Setosa、Iris Versicolor、Iris Virginicaの3つの値をとるカテゴリーデータです。これは質的データに分類されます。

機械学習で利用するデータは数値に置き換えて処理します。先述のIrisデータのclass列は文字データ（カテゴリーデータ）であるため、次のように置き換えて処理します。

- class:（品種）
    - Iris Setosa => 0
    - Iris Versicolor => 1
    - Iris Virginica => 2

```
sepal length,sepal width,petal length,petal width,class
5.3,3.7,1.5,0.2,0
5.0,3.3,1.4,0.2,0
7.0,3.2,4.7,1.4,1
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
```

データの前処理 - 2. 欠損値処理

収集したデータには不正なデータが紛れ込んでいたり、データが不足したりしているケースもあります。
次のケースを見てみましょう。

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,,1.4,Iris-versicolor
NaN,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
```

上記の3件目のレコードは3列目が空となっています。また4件目のレコードは先頭列がNaNという値になっています。このようなケースでは予め欠損するデータをどのように扱うか決めておく必要があります。たとえば次のような処理が考えられるでしょう。

1. 欠損値を含むレコードは学習データから除外する
2. 欠損値には、他のレコードの平均値を代入する
3. 欠損値には、指定の値（デフォルト値）を代入する

たとえば1のように欠損値を含むレコードを除外すると次のようになります。

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
6.9,3.1,4.9,1.5,Iris-versicolor
```

2のように欠損値に平均値を代入すると次のようになります。

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,3.0,1.4,Iris-versicolor
6.0,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
```

➢ 3行目の3列目（petal length）に平均値3.0、4行目の1列目（sepal length）に平均値6.0を代入しています。

また3のように指定の値として3行目の3列目（petal length）に3.3、4行目の1列目（sepal length）に6.6を指定すると次のようになります。

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,3.3,1.4,Iris-versicolor
6.6,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
```

データの前処理 - 3. スケーリング処理

欠損値処理以外にも、機械学習を効率よく進めるために既存のデータを整形することがあります。
このような作業を**スケーリング**と呼びます。

MNISTデータを例に考えてみましょう。MNISTデータは28x28ピクセルで構成されるグレースケールの画像データです。各画像ピクセルは0～255の数値をとります。

```
7,0,0,0,189,0,0,255,255,196,0,0...（※省略）
```

上記の例では先頭のデータは手書き数字（答え）を表しており、2つ目以降は画像の各ピクセルを表現しています。このようなケースでは値を0～1に置き換えることが一般的です。具体的には各ピクセルの値を255で割るようにします。データをスケーリングすると次のようになります。

```
7,0,0,0,0.74117647,0,0,1,1,0.76862745,0,0...（※省略）
```

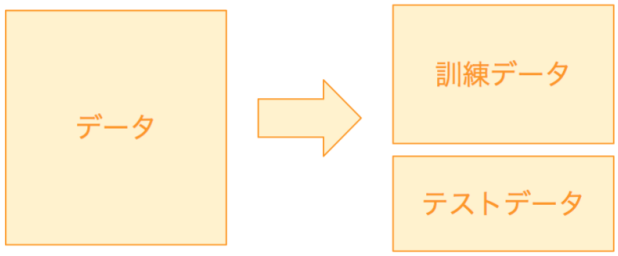
なおスケーリングは0～1に置き換える以外にも、データを 「(データ - 平均値) / 標準偏差」 で整形することもあります。この場合平均：0、標準偏差：1のデータにスケーリングされます。このような処理は**標準化**などと呼ばれます。

データの前処理 - 4. 訓練データとテストデータの準備

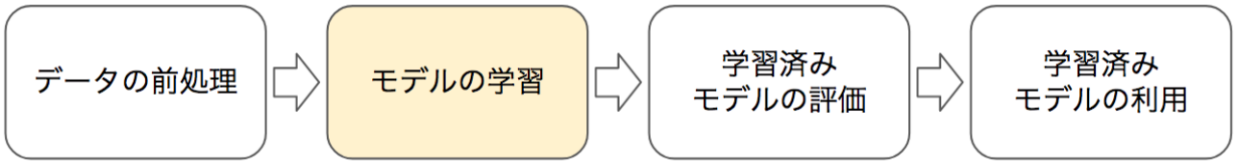
機械学習では訓練データだけで精度を測るのではなく、未知のデータに対する汎化能力を測定します。

データの欠損値処理やスケーリングを終えたら、訓練データとテストデータに分割します。これらのデータを事前に異なるファイルやデータベース（テーブル）に分割して用意しておくケースもあれば、機械学習プログラムの中でデータを分割するケースもあるでしょう。

具体的には機械学習用に10万件のレコードがある場合、5万件を訓練用、残りの5万件をテスト用という具合に分割します。分割する割合は、7:3、6:4、8:2など用途によって異なります。たとえばMNISTの場合は6万件の訓練用レコードと1万件のテスト用レコードが公開されています。



4.2 モデルの学習

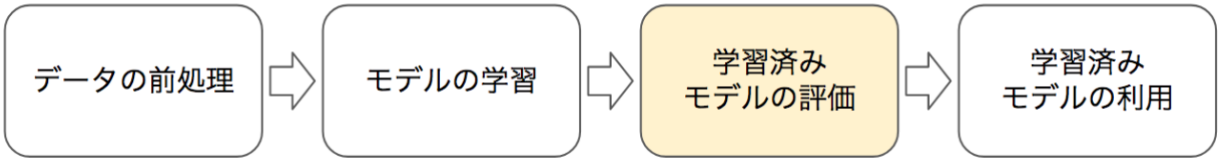


データの前処理を終えたら、機械学習アルゴリズムを決定してモデルの学習を始めます。ここでいうモデルとは機械学習後に利用可能な数式モデルを意味します。モデルは内部にパラメータを保持しており、学習によってパラメータが最適化されていきます。なお学習時には訓練用に準備したデータを使います。

➢ ここでは学習と呼びましたが、**訓練（Training）**と表現することもあります。

機械学習アルゴリズムには線形回帰やロジスティック回帰、決定木、ニューラルネットワークなど様々なものがあります。目的に合わせて、機械学習アルゴリズムを選択して、訓練データを順に処理していきます。訓練データを処理する度に期待値と実測値の誤差を求め、機械学習アルゴリズムの持つパラメータを更新していきます。

4.3 学習済みモデルの評価



モデルの学習が完了したら、学習済みのモデルを評価する必要があります。評価フェーズでは訓練用のデータとは別に用意したテストデータを使います。データの一部を訓練に使い、残りのデータの評価に使う手法を交差検証などと呼びます。

それではテストデータを使って評価するときにどのような指標を使えば良いでしょうか。扱う問題が分類問題の場合と回帰問題の場合で評価指標は異なります。

分類問題の評価指標

まず分類問題の評価指標です。ここではスパムメールの分類を考えてみましょう。分類問題の評価指標には**Accuracy（正確度、正答率）**を使います。具体的には、スパムメールとそうでないメールを含む10,000件のテストデータに対して、8,000件正解だった場合、Accuracyは80%という具合です。

また分類問題（2値分類）では評価時の正解、不正解のケースを次のように整理できます。

	正解ラベル - 真	**正解ラベル - 偽**
分類結果 - 真	True Positive	False Positive
分類結果 - 偽	False Negative	True Negative

➢ このような表は混同行列（Confusion Matrix）と呼ばれます。

スパムメールの分類では次のように考えます。

	正解：スパムメールである - 真	**正解：スパムメールでない - 偽**
予測：スパムメールである - 真	True Positive	False Positive
予測：スパムメールでない - 偽	False Negative	True Negative

ここでTrue Positiveに該当するのは、学習済みのモデルがスパムメールと予測して、結果もスパムメールだった場合です。同様にTrue Negativeに該当するのは、学習済みモデルがスパムメールでないと予測して、結果もスパムメールでなかった場合です。True PositiveもTrue Negativeも正解という意味では変わりありません。

次にはFalse Positiveに該当するのは、学習済みのモデルがスパムメールと予測したが、結果はスパムメールでなかった場合です。同様にFalse Negativeに該当するのは、学習済みのモデルがスパムメールでないと予測したが、結果はスパムメールであった場合です。False Positive、False Negativeのいずれも場合も不正解であることに変わりはないですが、間違え方が異なるのです。

具体的に数値を入れて2つのケースを比較してみましょう。

	正解：スパムメールである - 真	**正解：スパムメールでない - 偽**
予測：スパムメールである - 真	4,000件	2,000件
予測：スパムメールでない - 偽	0件	4,000件

上記のケース（※1）の場合、学習済みのモデルはすべてのスパムメールを検出できていますが、スパムメールでない正常なメールの一部も、スパムメールと誤って分類しているのがわかります。この学習済みモデルでスパムメールをフィルタリングした場合、正常なメールの一部もスパムメール（迷惑メール）として判定されてしまうでしょう。

	正解：スパムメールである - 真	**正解：スパムメールでない - 偽**
予測：スパムメールである - 真	4,000件	0件
予測：スパムメールでない - 偽	2,000件	4,000件

上記のケース（※2）の場合、学習済みのモデルは、スパムメールの一部をうまく検出できていないことがわかります。実際にこの学習済みモデルを使ってスパムメールをフィルタリングした場合、メールボックスにスパムメールが紛れこむケースが考えられるでしょう。

適合率、再現率、F値

正確度（Accuracy）に加えて、以下の指標についても覚えておくとよいでしょう。

- **適合率（Precision）**
    - Precision = TP / (TP + FP)
    - 真（Positive）と予測した中で、正解した割合

- **再現率（Recall）**
    - Recall = TP / (TP + FN)
    - 結果が真のデータの中で、正解した割合

さきほどのスパムメールのケース（※1）とケース（※2）の場合で考えてみましょう。

スパムメールのケース（※1）の場合、適合率は4000 / (4000 + 2000) = 66.7%、再現率は4000 / (4000 + 0) = 100%となります。ケース（※1）では過剰にスパムメールと判定していたため、適合率は低くなりましたが、取りこぼしはないため再現率は100%となっています。

一方で、スパムメールのケース（※2）の場合、適合率は4000 / (4000 + 0) = 100%ですが、再現率は4000 / (4000 + 2000) = 66.7%となります。スパムメールと判定してものは全て正解していたので適合率は100%ですが、取りこぼしたスパムメールが存在するため再現率は低くなっています。

適合率と再現率のどちらも高めることが理想ですが、一般的には両者はトレードオフの関係にあります。そのため適合率と再現率を入力としたF値と呼ばれる値を評価指標にすることができます。

- **F値（F1 Score）**
    - F1 Score= (2 * Precision * Recall) / (Precision + Recall)
    - 1に近いほど良い

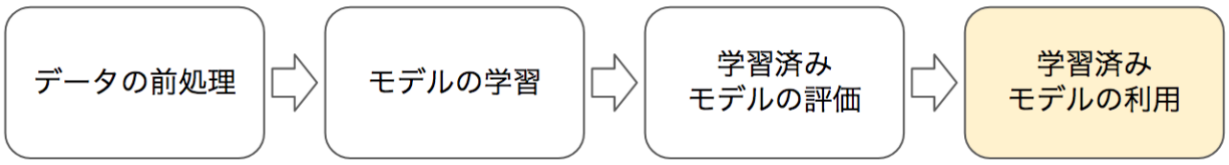
スパムメールのケース（※1）の場合のF値は(2 * 0.67 * 1) / (0.67 + 1) = 0.80、ケース（※2）の場合のF値も(2 * 1 * 0.67) / (1 + 0.67) = 0.80となります。

回帰問題の評価指標

回帰問題の場合は予測値と正解との誤差を評価指標にします。これにはいくつかの評価指標があります。

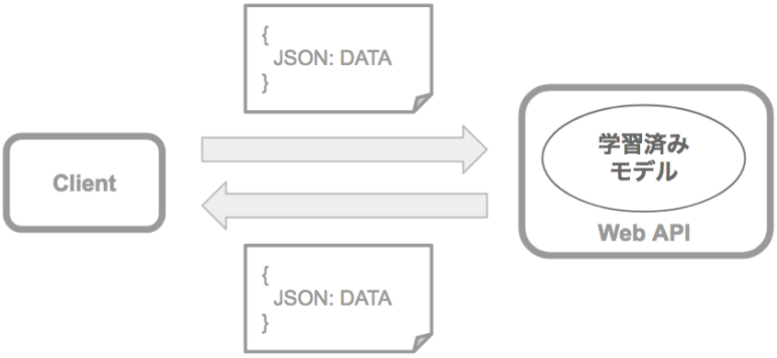
評価指標	日本語名	意味
Mean Absolute Error	平均絶対誤差	正解と予測値の差（絶対値）の平均値 0に近いほど精度は高い
Root Mean Squared Error	二乗平均平方根誤差	正解と予測値の差（二乗）の平均値の平方根 0に近いほど精度は高い
Relative Absolute Error	相対絶対誤差	平均絶対誤差を正規化したもの 0に近いほど精度は高い
Relative Squared Error	相対二乗誤差	二乗平均平方根誤差を正規化したもの 0に近いほど精度は高い
Coefficient of Determination	決定係数	期待値と実測値の差を0～1で表現したのもの 1に近いほど精度は高い

4.4 学習済みモデルの利用



評価を終えた学習済みモデルは実務の中で活用することになります。学習済みのモデルを実際に活用することを、モデルの推論（Predict）という呼び方をすることもあります。機械学習ではモデルの学習時にCPUやメモリといった大量のコンピュータリソースを必要としますが、推論時はそうではありません。そのため学習済みのモデルを他のコンピュータ上で動作させることができます。

学習済みモデルはコンポーネントとして、既存のシステムに組み込んで活用することができますし、Web APIを提供することで、多用途に学習済みモデルを活用することもできるでしょう。



5 scikit-learnによる機械学習プログラミング

scikit-learnはデータ解析やデータマイニングをシンプルに、かつ効果的に行えるライブラリです。線形回帰、決定木、SVMなど多くの機械学習手法が実装されているため、手軽に機械学習にチャレンジすることができます。

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

scikit-learnのサイトには以下のキーワードがあります。

- Classification（分類）
- Regression（回帰）
- Clustering（クラスタリング）
- Dimensionality reduction（次元削減）
- Model selection（モデル選定）
- Preprocessing（前処理）

分類や回帰は教師あり学習、クラスタリングや次元削減は教師なし学習の範囲になります。加えて、機械学習を行う上でのデータの前処理や、作成したモデルの検証など、機械学習に必要な様々な機能がライブラリとして用意されています。

演習1 教師あり学習（回帰問題）

ここではPythonの機械学習ライブラリscikit-learnを使って、教師あり学習プログラミングに取り組みます。まずは回帰問題を取り上げます。

線形回帰

まずはシンプルなプログラムを作成してみましょう。次のプログラムは線形回帰を行うものです。

> ➢ ここでは y = a * x + b という数式モデルの線形回帰問題を取り上げます。ここで a は係数、b は切片と呼びます。また y は 目的変数、 x は説明変数などと呼びます。与えられたデータから 係数 a と切片 b を見つけるのがプログラムの目的になります。

```
 1 import numpy as np
 2 from sklearn.linear_model import LinearRegression
 3
 4 x = np.array([0, 1, 2, 3, 4, 5]).reshape(-1, 1)
 5 y = np.array([5, 7, 9, 11, 13, 15]).reshape(-1, 1)
 6
 7 reg = LinearRegression()
 8 reg.fit(x, y)
 9
10 print(reg.predict([[6], [7], [8]]))
11
12 print(reg.coef_)
13 print(reg.intercept_)
```

実行結果は次のように表示されるでしょう。

```
[[ 17.]
 [ 19.]
 [ 21.]]
[[ 2.]]
[ 5.]
```

実行結果から x が 6, 7, 8 のとき、yの値はそれぞれ 17, 19, 21と予測しているのがわかります。また算出したモデルの係数、切片をそれぞれ 係数：2.0、切片 5.0と算出しています。

プログラムの詳細を見てみましょう。scikit-learnには様々な機械学習アルゴリズムが実装されているので、importによって利用することができます。

```
2 from sklearn.linear_model import LinearRegression
```

ここでは線形回帰（LinearRegression）クラスをimportとしています。

次にサンプルデータを準備しています。

```
4 x = np.array([0, 1, 2, 3, 4, 5]).reshape(-1, 1)
5 y = np.array([5, 7, 9, 11, 13, 15]).reshape(-1, 1)
```

ここではreshapeメソッドを使って、変数 x と y を2次元配列に変換しています。

- reshapeメソッドによって x は 二次元配列 [[0], [1], [2], [3], [4], [5]] となります。y も同様に[[5], [7], [9], [11], [13], [15]] となります。

次にLinearRegressionオブジェクトを生成し、reg.fitメソッドによってデータを学習させています。

```
7 reg = LinearRegression()
8 reg.fit(x, y)
```

学習はすぐに終わるので、未知のデータを投入して、予測（回帰）を確認してみましょう。

```
10 print(reg.predict([[6], [7], [8]])) #=>[[ 17.] [ 19.] [ 21.]]
```

また求めた係数や切片は以下のように確認できます。

```
12 print(reg.coef_)
13 print(reg.intercept_)
```

変数regのcoefは coefficient（係数）、intercept はintercept（切片）を意味します。scikit-learnでは学習済みのモデルから係数や切片の値を取得することができます。

- このサンプルではノイズのないデータを扱いましたが、実際には訓練データにノイズが含まれるのが普通です。

演習課題（教師あり学習-回帰問題）

次のデータは数学のテスト結果（math）と物理のテスト結果（physics）の結果をまとめたものです。

math	physics
80	90
82	95
65	71
45	42
72	88
66	72
68	76
90	94
83	83
77	82

上記のテスト結果を線形回帰で分析します。ここでは数学のテスト結果から物理のテスト結果を推論するものとします。回帰直線の回帰係数（coefficient）と切片（intercept）を求めてください。

```
1 import numpy as np
2 from sklearn.linear_model import ???
3
4 math = np.array([80, 82, 65, 45, 72, 66, 68, 90, 83, 77]).reshape(-1, 1)
5 physics = np.array([90, 95, 71, 42, 88, 72, 76, 94, 83, 82]).reshape(-1, 1)
6
7 ???
```

演習2 教師あり学習（分類問題）

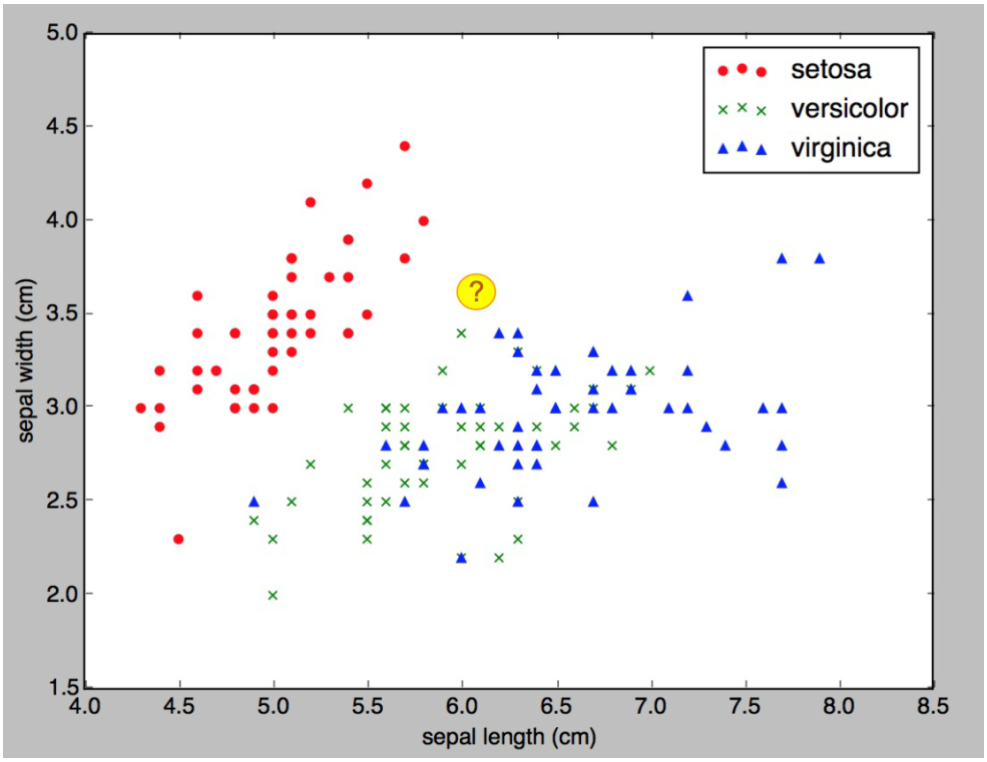
続いてデータの分類を行う機械学習プログラムを見てみましょう。
ここではk-NN法（k-nearest neighbor algorithm）を取り上げます。

k-NN法による分類

k-NN法はk近傍法とも呼ばれます。k近傍法は最寄りのデータを参考に分類を行うシンプルなアルゴリズムです。
続いてデータの分類を行う機械学習プログラムを見てみましょう。

➢ 教師あり学習で扱う問題は、回帰と分類に分かれます。

ここではk-NN法（k-nearest neighbor algorithm）を取り上げます。日本語ではk近傍法とも呼ばれます。k近傍法は最寄りのデータを参考に分類を行うシンプルなアルゴリズムです。



例えば、x0、x1 の2点で構成されるデータの場合を考えてみましょう。これは説明変数が2つというケースですが、このようなケースは2次元のグラフ上に表現できます。k-NN法では、予め訓練データをこのグラフ上に表現しておき、未知のデータを投入した際に、最も近くにある点（データ）を参考に分類を行います。

➢ 実際には最寄りの1点だけで決定するのではなく、最寄りの3点、5点とパラメータを調整することができます。

iris（アヤメ）データの分類 - 1 CSVファイルの分類

ここではテストデータとしてiris（アヤメ）データの分類にチャレンジします。具体的には次のようなCSVデータを訓練データに使います。

```
...
1,4.8,1.4
1,5.0,1.7
1,4.5,1.5
1,3.5,1.0
2,6.0,2.5
2,5.1,1.9
2,5.9,2.1
...
```

- ➢ ここでは60件の訓練データファイル（train01.csv）とテストデータファイル（test01.csv）を用意しています。

データは3列で構成しています。先頭列は品種を示します。1は'versicolor', 2は'virginica'という品種です。2列目、3列目はpetal（花びら）の縦、横幅です。

- ➢ 1,4.8,1.4 の場合、品種は 1:'versicolor' 花びらの縦幅 4.8cm、 横幅: 1.4cmという具合です。

それでは実際にirisデータの分類にチャレンジしてみましょう。

```
 1 import numpy as np
 2 from sklearn.neighbors import KNeighborsClassifier
 3
 4 train = np.loadtxt("datasets/train01.csv", delimiter=",", skiprows=1)
 5 test = np.loadtxt("datasets/test01.csv", delimiter=",", skiprows=1)
 6
 7 x_train = train[:,1:]
 8 y_train = train[:,0]
 9
10 x_test = test[:,1:]
11 y_test = test[:,0]
12
13 clf = KNeighborsClassifier()
14 clf.fit(x_train, y_train)
15
16 y_pred = clf.predict(x_test)
17 print(y_pred)
18 print(y_test)
19 print(clf.score(x_test, y_test))
```

実行結果は次のようになるでしょう。

```
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  2.  2.  2.  1.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.
  2.  2.  2.  2.]
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.
  2.  2.  2.  2.]
0.975
```

予測値と期待値はほぼ一致し、結果は97.5%という高い正答率となっています。

それではプログラムの詳細を見てみましょう。

scikit-learnでk-NN法による分類を行うには KNeighborsClassifier クラスをimportします。

```
 2 from sklearn.neighbors import KNeighborsClassifier
```

次にNumPyのloadtxtメソッドを使ってCSVファイルをロードし、訓練データとテストデータに分類しています。

```
 4 train = np.loadtxt("datasets/train01.csv", delimiter=",", skiprows=1)
 5 test = np.loadtxt("datasets/test01.csv", delimiter=",", skiprows=1)
 6
 7 x_train = train[:,1:]
 8 y_train = train[:,0]
 9
10 x_test = test[:,1:]
11 y_test = test[:,0]
```

CSVファイルをロードした後、先頭列（解答）とそれ以外の列を分離しています。

> ➢ 機械学習では訓練データを使って学習し、テストデータで正答率を検証します。これによってモデルの汎化性能を評価します。

次に KNeighborsClassifier オブジェクトを生成しています。このようなデータの分類を行うオブジェクトは分類器などと呼ばれます。

```
13 clf = KNeighborsClassifier()
14 clf.fit(x_train, y_train)
```

分類器を生成した後、clf.fitメソッドを呼び出して訓練データによる学習をスタートします。

学習を終えたら、テストデータで結果を検証してみましょう。ここでは予測値と期待値（y_test）を並べて出力しています。それからclf.scoreメソッドによって正答率も出力していています。

```
16 y_pred = clf.predict(x_test)
17 print(y_pred)
18 print(y_test)
19 print(clf.score(x_test, y_test))
```

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score

train = np.loadtxt("datasets/train01.csv", delimiter=",", skiprows=1)
test = np.loadtxt("datasets/test01.csv", delimiter=",", skiprows=1)

x_train = train[:,1:]
y_train = train[:,0] -1

x_test = test[:,1:]
y_test = test[:,0] - 1

clf = KNeighborsClassifier()
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)
print(y_pred)
print(y_test)
print(clf.score(x_test, y_test))

print("confusion_matrix:\n", confusion_matrix(y_test, y_pred))
print("accuracy_score:", accuracy_score(y_test, y_pred))
print("precision_score:", precision_score(y_test, y_pred))
print("recall_score:", recall_score(y_test, y_pred))
print("f1_score:", f1_score(y_test, y_pred))
```

プログラムを実行すると次のような結果を確認できるでしょう。

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
0.975
confusion_matrix:
 [[20  0]
 [ 1 19]]
accuracy_score: 0.975
precision_score: 1.0
recall_score: 0.95
f1_score: 0.9743589743589743
```

➢ いずれの関数も1を真（0を偽）として処理します。そのため x_train、y_trainの値を調整しています。

演習課題（教師あり学習-分類問題）

次のデータは、架空の400校の学習方法（learn）と、数学のテスト結果（math）と物理のテスト結果（physics）の結果をまとめたものです。

➢ 学習方法（learn）には4つの方法（0～3）が存在します。

learn	math	physics
0	44.87	38.16
0	…	…
1	48.61	47.85
1	…	…
2	55.47	60.26
2	…	…
3	66.78	69.49
3	…	…

➢ ここでは配布資料のex_class.csvファイルを使います。

ここではk-NN法を使って数学、物理のテスト結果から、いずれの学習方法が適用されたかを分類するプログラムを作成します。

➢ k-NN法では最寄りの5点から分類するものとします。

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from sklearn.model_selection import train_test_split
4  from sklearn.neighbors import KNeighborsClassifier
5
6  data = np.loadtxt("datasets/ex_class.csv", delimiter=',', skiprows=1)
7
8  x = data[:,1:]
9  y = data[:,0]
10
11 x_train, x_test, y_train, y_test = ???(x, y)
12
13 model = KNeighborsClassifier()
14 model.???(x_train, y_train)
15
16 model.???(x_test, y_test)
```

次のような実行結果（正答率）が表示されることを確認してください。

```
0.93999999999999995
```

参考：推論利用

学習済みモデルでテストデータを推論してみましょう。

```
18 y_pred = model.???(x_test)
19 print(y_pred)
20 # print(y_test)
```

次のような実行結果が表示されることを確認してください。

```
[ 2.  2.  1.  1.  3.  3.  1.  3.  2.  2.  0.  1.  2.  3.  2.  2.  2.  3.
  3.  2.  3.  1.  0.  2.  0.  3.  2.  1.  1.  2.  1.  0.  3.  3.  3.  0.
  2.  0.  3.  3.  3.  0.  2.  3.  0.  2.  0.  3.  1.  2.  0.  2.  0.  1.
  3.  3.  0.  2.  0.  2.  3.  0.  2.  2.  2.  1.  0.  0.  0.  0.  1.  0.
  0.  1.  3.  1.  2.  0.  3.  0.  0.  1.  2.  0.  3.  2.  0.  0.  0.  3.
  1.  1.  1.  1.  0.  1.  1.  1.  0.  2.]
```

➢ print(y_test)の結果と比較してみましょう。

Appendix - Pythonプログラミング

A.1 Python入門

本講座ではPython言語を活用して機械学習の理解を深めます。Pythonは多目的に活用できるシンプルで扱いやすい言語です。インデントによってif文やfor文などのスコープを表現するという特徴があります。また機械学習ライブラリが充実しているため近年注目を集めています。

それでは実際にPythonプログラムの概要を見てみましょう。

Hello World

画面に"Hello World"を表示してみましょう。

```
1 print("Hello World")
```

実行結果は次のように表示されるでしょう。

```
Hello World
```

変数とデータ型

Pythonの文字列、数値、実数、真偽値型、None型を確認してみましょう。

```
1  name = "John"
2  age = 20
3  height = 170.1
4  license = True # False
5  money = None
6
7  print(type(name))
8  print(type(age))
9  print(type(height))
10 print(type(license))
11 print(type(money))
```

実行結果は次のように表示されるでしょう。

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'NoneType'>
```

type関数を使えばデータ型を確認できます。

➢ Pythonプログラムの中でコメントは # で表現します。

if文

if文は次のようになります。論理積を表現するときは and キーワードでつなぎます。また他言語のように {} は使わずにインデントでスコープを表現します。

```
name = "John"
age = 20
license = True

if age >= 20 and license:
    print("OK {}".format(name))
else:
    print("NG")
```

実行結果は次のように表示されるでしょう。

```
OK John
```

➢ 論理和は or キーワードを使います。

for文

for文を使って繰り返し構造を定義できます。指定回数繰り返すにはrange関数を使います。

```
for i in range(3):
    print(i)

for i in range(1, 3):
    print(i)

for i in range(3, 1, -1):
    print(i)
```

実行結果は次のように表示されるでしょう。

```
0
1
2
1
2
3
2
```

リスト

Pythonは [] キーワードでリスト（配列）を生成できます。

```
fruits = ["apple", "banana", "cherry"]

print(fruits[1])

for fruit in fruits:
    print(fruit)

for i, fruit in enumerate(fruits):
    print(i, fruit)
```

実行結果は次のように表示されるでしょう。

```
banana
apple
banana
cherry
0 apple
1 banana
2 cherry
```

➢ enumerate関数を使えば、インデックスとリストの要素を取り出すことができます。

ディクショナリ

Pythonは {} キーワードでディクショナリを生成できます。ディクショナリとはキーと値をマッピングしたものです。

```
fruits = {"apple":100, "banana":200, "cherry":300}

print(fruits["apple"])

for key in fruits:
    print(key, fruits[key])
```

実行結果は次のように表示されるでしょう。

```
100
apple 100
banana 200
cherry 300
```

➢ keysメソッドですべてのキーを、valuesメソッドですべての値を、itemsメソッドですべてのキーと値の組み合わせを取り出すことができます。

関数

Pythonではdefキーワードを使って関数を定義します。

```
def sum(start, end):
    result = 0
    for i in range(start, end + 1):
        result = result + i
    return result

print(sum(1, 1))
print(sum(1, 5))
print(sum(1, 10))
```

実行結果は次のように表示されるでしょう。

```
1
15
55
```

クラス

Pythonはオブジェクト指向言語です。クラスを定義するには次のように実装します。

```
class Car:
    def __init__(self, name, gas):
        self.name = name
        self.gas = gas

    def move(self):
        if self.gas > 0:
            self.gas = self.gas - 1
            print("{}: move".format(self.name))
        else:
            print("{}: stop".format(self.name))

car1 = Car('kbox', 3)
car2 = Car('Kwagon', 5)

for i in range(5):
    car1.move()
    car2.move()
```

実行結果は次のように表示されるでしょう。

```
kbox: move
Kwagon: move
kbox: move
Kwagon: move
kbox: move
Kwagon: move
kbox: stop
Kwagon: move
kbox: stop
Kwagon: move
```

クラスの中でインスタンス自身を参照するにはselfキーワードを使います。またクラスに定義したメソッドの第1引数にはselfを指定する必要があります。また、クラスにコンストラクタを定義する場合は __init__メソッドを実装します。

- ➢ クラスからインスタンスを生成するときにnewキーワードは不要です。

A.2 NumPyライブラリ

NumPyは、Pythonの数値計算ライブラリです。ndarrayという多次元配列を使って、数値データを高速に、かつ柔軟に処理できることが特徴です。
本講座では主にテストデータの作成や、行列演算に利用します。

NumPyを使う場合は以下のようにimportします。

```
import numpy as np
```

NumPy配列の作成

NumPy配列を生成するにはいくつかの方法があります。

```
 1 import numpy as np
 2
 3 a = np.arange(3)
 4 print(a)
 5
 6 b = np.array([10, 20, 30])
 7 print(b)
 8
 9 c = np.random.rand(3)
10 print(c)
```

実行結果は次のように表示されるでしょう。

```
[0 1 2]
[10 20 30]
[ 0.20971528  0.86719696  0.70371265]
```

配列の要素の範囲を指定して生成する場合は np.arange メソッドを使います。

```
 3 a = np.arange(3)
```

上記の場合 [0 1 2] というNumPy配列が生成されます。

またnp.arrayメソッドを使えば通常の配列をNumPy配列に変換することができます。

```
 6 b = np.array([10, 20, 30])
```

上記の場合 [10 20 30] というNumPy配列が生成されます。

np.random.randメソッドを使えば、乱数で構成される配列を生成できます。

```
 9 c = np.random.rand(3)
```

乱数は [0.56886361 0.68599281 0.7017446] のように表示されるでしょう。

配列の操作

次は配列の操作について見てみましょう。

```
 1 import numpy as np
 2
 3 a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
 4 print(a)
 5 print(a.shape)
 6
 7 print(a[1])
 8 print(a[1:5])
 9 print(a[5:])
```

実行結果は次のように表示されるでしょう。

```
[1 2 3 4 5 6 7 8 9]
(9,)
2
[2 3 4 5]
[6 7 8 9]
```

NumPy配列は shape プロパティにアクセスすることで、配列の次元ごとの要素数をタプルで取得できます。上記の場合は (9,) と表示されます。

➢ タプルはリストと同じように扱うことができますが、内部の要素を変更することはできません。

PyhtonのリストやNumPy配列は要素へのアクセスを柔軟に指定できます。

```
 7 print(a[1])   # => 2
 8 print(a[1:5]) # => [2 3 4 5]
 9 print(a[5:])  # => [6 7 8 9]
```

➢ NumPy配列でなく通常のリストでも上記のようにアクセスできます。

NumPy配列は次元数を変更することもできます。次のプログラムを見てみましょう。

```
 1 import numpy as np
 2
 3 a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
 4 b = a.reshape(3, 3)
 5 print(b)
 6 print(b.shape)
 7
 8 print(b[1])
 9 print(b[1,1])
10
11 print(b[0:2])
12 print(b[0:2,1])
13
14 print(b[:,2])
```

実行結果は次のように表示されるでしょう。

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
[4 5 6]
5
[[1 2 3]
 [4 5 6]]
[2 5]
[3 6 9]
```

NumPy配列は reshape メソッドによって配列の次元数を変更できます。

```
 4 b = a.reshape(3, 3)
```

また多次元配列は次のようにカンマ区切りで要素にアクセスできます。

```
 9 print(b[1,1])
```

少し特殊な書き方ですが、次のように2次元目の要素番号だけを指定することもできます。

```
14 print(b[:,2])
```