Hands on Machine Learning with Scikit-Learn &TensolFlow

Chapter 1

Created by Yusuke FUJIMOTO

はじめに

- この資料は「Hands-On Machine Learning with Scikit-Learn and TensorFlow - O'Reilly Media」を読んだ際の（主にソースコードに関する）簡単な解説を残したものです。
- 全部を解説したわけではないので注意
- あとは一部訳を間違えているかもしれないので信用しすぎないこと
- 余裕があればソースコード周りの背景知識もまとめたい

Chapter 1

The Machine Learning Landscape

Setup

```
# Common imports
import numpy as np #  as 以降で名前簡略化

# to make this notebook's output stable across runs
rnd.seed(42) # 再現性のための乱数のシード設定
```

- 乱数のシードを設定しないと毎回random関数で違う値が出てくる → 再現できない

Load and prepare Life satisfaction data

```
# データフレームを扱う際はこれをよく使う
import pandas as pd

A = pd.read_csv(filepath) #  csv 読み取り

## A のCOLNAME列の値が"X"であるもののみ抽出
A = A[A["COLNAME"]=="X"]

## ピボットテーブルの作成
A = A.pivot(
        index="Country",      #  集計したい縦のキー
        columns="Indicator", #  集計したい横のキー
        values="Value"        #  集計したい項目
            )
A.head(2) #  A の先頭2行のみ表示
```

Load and prepare GDP per capita data

```
A.rename()          #  名前の変更
A.set_indiex("X") #  特定の列をインデックス（左端の列）にする

# A と B をマージする
pd.merge(
        left=A,
        right=B,
        left_index=True, #  index を考慮するか否か
        right_index=True
        )

# 列XとYの値に対し、インデックスがVALUEである行を返す
# VALUEには普通は数値が入るが、先ほどの処理で
# インデックスが変わったため
A[["X", "Y"]].loc["VALUE"]
```

loc, iloc, ix の違い

基本的にこれらは3つとも「行、列」を指定してデータを参照するという目的で使われる。

違いは以下の通り

	行、列の指定方法
loc	行ラベル、列ラベル
iloc	行の番号（0～）、列の番号（0～）
ix	行ラベル、列ラベル または 行の番号（0～）、列の番号（0～）

Load and prepare GDP per capita data（続き）

```
B = [0, 1, 6, 8, 33, 34, 35]

# 0 から 35 までの整数の集合から、Bの集合にある値を除いて
# リスト型に変換している
B = list(set(range(36)) - set(B))
```

- set 型 ：集合、同じ値は追加されない
- list() : リスト型に変換される
- range(x) : [0, … x-1]

Load and prepare GDP per capita data（続き）

```
# 各国とその情報（GDPと満足度）を図に書き足していく
for country, pos_text in position_text.items():
    # x は GDP、 y は Life satisfaction
    x, y = sample_data.loc[country]

    # if else 文の略表記
    country = "U.S." if country == "A" else country

    # 散布図の各要素に文字を付ける
    plt.annotate(
        country, xy=(x, y), xytext=pos_text,
        arrowprops=dict(
            facecolor='black', width=0.5,
            shrink=0.1, headwidth=5
        ))

    # 点を書き足す（赤色の〇を表す
    plt.plot(pos_data_x, pos_data_y, "ro")
```

Load and prepare GDP per capita data（続き）

```
import numpy as np #  よく使うライブラリ、行列演算が速くできる
# データ表示
sample_data.plot(
    kind='scatter',
    x="GDP per capita",
    y='Life satisfaction',
    figsize=(5,3)
)
plt.axis([0, 60000, 0, 10]) #  座標軸設定
X=np.linspace(0, 60000, 1000) #  0 から 60000 まで 1000ずつ
plt.plot(X, 2*X/100000, "r")  #  各点を計算して図に追加

#  tex 形式で数式を表現できる
plt.text(
    40000, 2.7, r"$\theta_0 = 0$",
    fontsize=14, color="r"
)
```

Load and prepare GDP per capita data（続き）

```
from sklearn import linear_model #  線形回帰モデル
lin1 = linear_model.LinearRegression()
#  np.c_ は横に連結、 np.r_ は縦に連結
Xsample = np.c_[sample_data["GDP per capita"]]
ysample = np.c_[sample_data["Life satisfaction"]]
lin1.fit(Xsample, ysample) #  モデルで学習
# t0 は 切片、 t1 は偏回帰係数
t0, t1 = lin1.intercept_[0], lin1.coef_[0][0]
```