Python チュートリアル(中級編)受講報告

はじめに

2023 年 11 月 27 日、12 月 4 日、12 月 11 日に分けて、大阪大学サーバーメディアセンターにより、表題の講習会が開催されました。その受講報告です。（11 月 27 日は、所要により参加出来なかったので、後日、録画版での受講になりました。）

Day 1 (11 月 27 日分)

１．プログラム学習における心得

・細かく覚えようとしない

「何ができるか」で、「どうやってできるか」は覚えなくて良い

・先人の知恵を活用する

自力で考えない。Google で検索する。検索力は非常に重要です。

・やりたいことドリブンで学ぶ

ドリブンとは、〇〇に基づいて　という意味です。やりたいことから決める。

・Done is better than perfect

「完璧を目指すより終わらせろ」ザッカーバーグ（Facebook の創始者）の言葉

2.　ライブラリを用いて効率的に開発できる

NumPy（数値計算　高速な行列計算、簡単に数列を生成できる）

Matplotlib　（簡単にデータから描画できる）

scikit-learn ,Tensorflow (機械学習)

が　紹介された。

3. 実践演習として、Web スクレイピング　を学びました。

Web サイトからのデータ取得の自動化を行います。

ライブラリは、BeautifulSoup で　使用します。

HTML のコードを取得するので、日時からの検索等が可能となります。

（個人的には、利用する機会はないように考えています）

Day 2 (12 月 4 日分)

Pandas を使ったデータ操作を学ぶ

1. Pandas とはデータ分析のためのライブラリです。

データの読み込み、加工分析を容易にします。

機械学習において、データの前処理のため用います。

以下の様々なデータが機械学習の対象となります。

・テーブルデータ

・時系列データ

・画像・動画・音声

・自然言語

データを加工して、モデルに組み込むため、Pandas を使います。

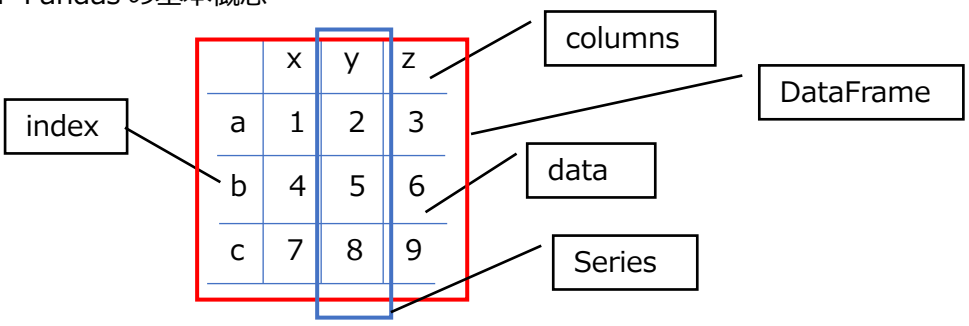
テーブルデータは、代表的に二つの変数があります。

・数値変数 （数値で表す変数）→ そのままモデルに入力できます。

・カテゴリカル変数 （文字列のようなデータの分類がしめされた変数）

→ そのままモデルに入力できません。数値に変換したりします。

2. Pandas の基本概念



Series は 1 次元であり、 DataFrame は 2 次元となります。

3. 基本的なコードの紹介

3.1　Pandas　への変換

```
Series = pd.Series(data=***)　#*** のリストから作成
df = pd.DataFrame(data=***, index=***, columns=***) #*** のリストから
df = pd.read_csv(********) #csv データから作成
```

3.2　基本操作

```
df = df.copy()　　#要素へのアクセス
df[“**”:”**”]　　df[2:4]　　#行のスライス
df[“**”]　　#列名
df.at[“**”,”**”] #単独要素へのアクセス
df.iat[2, 3]　　#単独要素へのアクセス
df.loc[“**”:”**”, “**”]　　#単独要素へのアクセス
df.iloc[2,3]　　#単独要素へのアクセス
```

3.3　欠損値処理

```
df.dropna(how='all')　#全ての値が欠損値である行を削除
df.fillna(0)　　#欠損値を 0 で置き換え
df.fillna(df.mean())　　#欠損値を平均値で置き換え
df.fillna(df.median())　#欠損値を中央値で置き換え
df.fillna(df.mode().iloc[0])　#欠損値を最頻値で置き換え
```

3.4　統計的分析

```
df.head()　　#先頭のみ出力
df.info()　　#データの要約のみ出力
df.insull().sum()　　#欠損値の個数を出力
df.describe() #データ数、平均値、標準偏差、最小値、第一四分位数、中央値、
                第三四分位数、最大値
df.hist() #ヒストグラムで可視化
```

Day 3 (12 月 11 日分)

AI プログラミングの初歩

1. 概略説明

人工知能(AI)については、確率した学術的な定義や合意はないです。主流は、‘人間が知能を使ってすることを機械にさせようとする’ になってます。

機械学習は、‘データから規則性や判断基準を学習し、それにもとづき未知のものを予測、判断する技術’ で、‘教師あり学習’、‘教師なし学習’、‘強化学習’ の 3 数類に大別されます。 教師あり学習は、回帰 と 分類 があります。

回帰：未知のデータに対応する数値(連続値)を予測します。

分類：未知のデータが属するクラス(離散値)を予測します。

データに関して、述べると、テーブルデータ、時系列データ、画像・動画・音声、自然言語等のデータが機械学習の対象となり、代表的な変数として、数値変数、カテゴリカル変数があり、モデルに理解しやすい特徴量に変換することがあります。

数値変数 ： 正規化、標準化、対数変換

カテゴリカル変数 ： 数値に変換（Label Encording）

2. 内容

今回は、‘タイタニック号沈没での、生存者を予測せよ’ というチュートリアルが行われました。タイタニック号の乗客について、生存したか/死亡したかを予測する
機械学習モデルを作成し、学習と予測結果の評価を行います。
併せて、決定木を作成します。

タイタニック号の事例は、初歩用の題材としてよく使います。とりあえず、ライブライリと主なコードを紹介し、全コードは、おまけの項に記載します。

2.1 タイタニック号の乗客のデータ ライブライリ seaborn にあります。

内容は、生存したか、旅客クラス、性別(カテゴリカル変数)、年齢、兄弟・配偶者の数、親・子供の数、乗船料金、出港地(カテゴリカル変数) についてのデータです。

2.2 データの分析・特徴量作成（データの前処理）

・Day2 で紹介したコードで、概要を把握します。(df.describe() , df.hist(),df.corr())

・ラベルエンコーディングを行います。（カテゴリカル変数を数値変数へ）

```
from sklearn.preprocessing import LabelEncoder　#ライブラリのインポート
le = preprocessing.LabelEncoder()
labels = ['posi', 'nega', 'posi']　　#posi と　nega の配列を数字に変換します。
labels_id = le.fit_transform(labels)　　#　0,1,0 の配列に変更されます。
```

・相関関係をヒートマップで表示します。

```
import matplotlib.pyplot as plt
_, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(df_le.corr(), annot=True, ax=ax)
```

・カラム別の生存率を表示します。

```
sns.catplot(x='survived', col='sex', kind='count', data=df)
```

・家族の人数毎の生存率を表示します。

```
sns.catplot(x='survived', col='sex', kind='count', data=df)
```

2.3　学習（機械学習（決定木）で予測していきます）

・データを学習データとテストデータに分割します。

```
from sklearn.model_selection import train_test_split　#ライブラリのインポート
X = df.drop('survived', axis=1) #  説明変数
y = df['survived'] #  正解ラベル
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,　＼
random_state=0, stratify=y)　#0.2 はテスト用数量の割合、random_state は
```

乱数シードを固定させます。stratify=y は y （正解ラベル）の比率を一定にします。

・決定木で学習します。

学習データでモデルを作ります。学習データのモデルの確率も算出します。

このモデルで、テストデータを予測します。予測精度を確認します。

```
from sklearn.tree import DecisionTreeClassifier　＃ライブラリのインポート
model = DecisionTreeClassifier(max_depth=3)　#モデルの定義、max_depth は
                                                決定木の最大の層の数を指定してます。
model.fit(X_train, y_train) #  学習データをモデルに入れます。
model.score(X_train, y_train)　＃精度の算出です。
```

テストデータで予測し、予測精度を確認します。

```
pred = model.predict(X_test)　 #テストデータをモデルに入れて予測します。
from sklearn.metrics import accuracy_score　　#ライブラリのインポート
accuracy_score(pred, y_test)　　#予測精度を算出します。
```

3 回の講座の感想

・講師が作成された　google colaboratory 内のファイルを、参加者が各自の PC にダウンロードして、講師の指導のなかで、コードを実行していくという内容でした。
視聴するだけで、後で試行するということではないので、よく理解できました。

・Python を用いた機械学習では、最適なライブラリの関数を使用することが肝心なようです。関数もブラックボックですので、内容を少しは理解してみようと、考えてます。

・今回の事例と類似した事象を念頭に、やりたいことを見つけるというのが、
スタートではないかと、思っています。（やりたいことドリブン）

おまけ（タイタニック号のモデル実装・モデル予測のコードを記載します）

1 ステップ毎に出力して、確認していたので、1 ステップを 1 枠としてます。

講座では　google colaboratory　で入力しています。

```
# タイタニック号のデータセットを読み込む
import seaborn as sns
df = sns.load_dataset('titanic')
required_columns = ['survived', 'pclass', 'sex', 'age', 'sibsp',
'parch', 'fare', 'embarked']
df = df[required_columns]
df
```

```
df.head()
```

```
df.describe()
```

```
df.hist(figsize=(12, 12))
```

```
# ラベルエンコーディング（カテゴリカル変数を数値変数に変換（欠損値は除く））
from sklearn.preprocessing import LabelEncoder
import pandas as pd

categorical_features = ['sex', 'embarked']

df_le = df.copy()

for c in categorical_features:
    le = LabelEncoder()
    not_null = df_le[c][df_le[c].notnull()] # 欠損値以外のデータをエンコーディングする
    df_le[c] = pd.Series(le.fit_transform(not_null),
index=not_null.index)

df_le.head()
```

```
# ラベルエンコーディング（カテゴリカル変数を数値変数に変換（欠損値は除く））
from sklearn.preprocessing import LabelEncoder
import pandas as pd

categorical_features = ['sex', 'embarked']

df_le = df.copy()

for c in categorical_features:
    le = LabelEncoder()
    not_null = df_le[c][df_le[c].notnull()] # 欠損値以外のデータをエンコーディングする
    df_le[c] = pd.Series(le.fit_transform(not_null), index=not_null.index)

df_le.head()
```

```
# 各変数の相関係数をヒートマップで確認
import matplotlib.pyplot as plt
_, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(df_le.corr(), annot=True, ax=ax)
```

```
# 性別ごとの生存率を確認
sns.catplot(x='survived', col='sex', kind='count', data=df)
```

```
# 旅客クラスごとの生存率を確認
sns.catplot(x='survived', col='pclass', kind='count', data=df)
```

```
# 新しい特徴量として家族の人数を追加
for d in [df, df_le]:
  d['family'] = d['sibsp'] + d['parch']

df.head()
```

```
# 家族の人数ごとの生存率を確認
sns.pointplot(x='family', y='survived', data=df)
```

```
# 各列のユニークな要素の個数を出力
df.nunique()
```

```
# 各カラムの欠損値を出力
df.isnull().sum()
```

```
# 欠損値を埋める
df['age'].fillna(df['age'].mean(), inplace=True)
df['embarked'].fillna(df['embarked'].mode()[0], inplace=True)
```

```
# 欠損値が埋まったことを確認
df.isnull().sum()
```

```
# カテゴリカル変数をラベルエンコード
categorical_features = ['sex', 'embarked']

for c in categorical_features:
    le = LabelEncoder()
    le.fit(df[c])
    df[c] = le.transform(df[c])

df.head()
```

```
df.dtypes
```

```
# データセットを学習データとテストデータに分割
from sklearn.model_selection import train_test_split

X = df.drop('survived', axis=1) # 説明変数
y = df['survived'] # 正解ラベル
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0, stratify=y)
```

```
# 学習データの説明変数
X_train
```

```
# 学習データの説明変数
X_train
```

```
# テストデータの説明変数
X_test
```

```
# テストデータの正解ラベル
y_test
```

```
# 決定木で学習
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=3)
model.fit(X_train, y_train) # 学習
model.score(X_train, y_train)
```

```
# テストデータで予測
pred = model.predict(X_test)

pred
```

```
len(pred)
```

```
# 予測精度を確認
# sum(pred == y_test) / len(y_test) と同じ
from sklearn.metrics import accuracy_score

accuracy_score(pred, y_test)
```

```
# 決定木を可視化
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(15, 10))
plot_tree(model, feature_names=X_test.columns,
class_names=['dead', 'survived'], filled=True)
plt.show()
```