# 深層学習入門 第2回

東海大学 情報通信学研究科 情報通信学専攻 南 直輝

# 今日の内容

・タイタニック号の乗客者の年齢や性別などの項目がある表形式データセットを用いて、生存予測をする。(PyTorch を用いる)

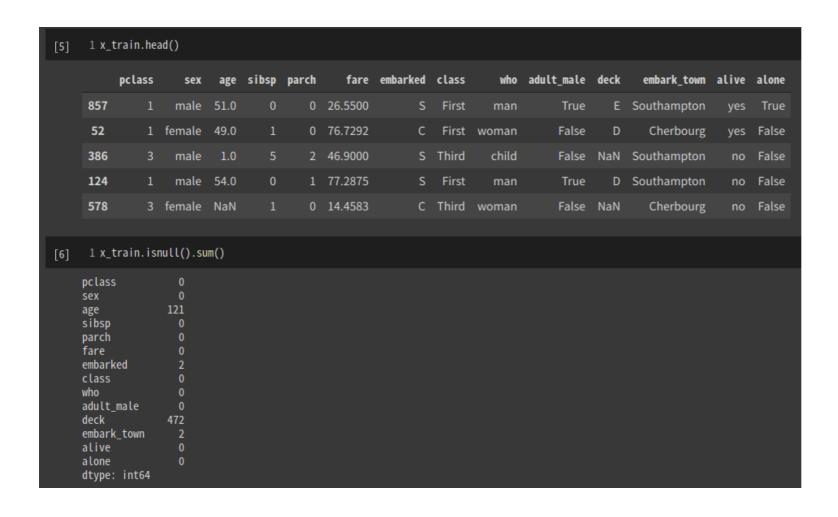
#### データセットをダウンロードし内容を確認する

- ・support.py を取得し、インポートする。
- ・support.py にある fetch\_titanic 関数を用い 、訓練用データとテスト用データに分割す る。
- ・x\_train の情報(オブジェクトの型や Columnなど)を確認する。

```
1 !curl -O https://raw.githubusercontent.com/khirotaka/tigerfish/master/utils/support.py
 % Total % Received % Xferd Average Speed Time Time
                                                              Time Current
                               Dload Upload
                                             Total Spent
                                                              Left Speed
1 import support
1 (x_train, y_train), x_test = support.fetch_titanic()
1 x_train.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 623 entries, 857 to 684
Data columns (total 14 columns):
    Column
                 Non-Null Count Dtype
    pclass
                623 non-null
                                int64
                623 non-null
    sex
                               object
    age
                502 non-null
                               float64
    sibsp
                623 non-null
                                int64
    parch
                623 non-null
                               int64
    fare
                623 non-null
                              float64
    embarked 621 non-null
                               object
    class
                623 non-null
                               category
    who
                 623 non-null
                               object
    adult male 623 non-null
                               bool
 10 deck
                 151 non-null
                             category
 11 embark_town 621 non-null
                               object
 12 alive
                 623 non-null
                               object
 13 alone
                623 non-null
                               bool
dtypes: bool(2), category(2), float64(2), int64(3), object(5)
memory usage: 56.4+ KB
```

# データセットをダウンロードし内容を確認する 続き

- ・先頭の5行を確認する。x\_trainは、DataFrame型なので.head()が使える。
- ・カラム内にあるnullの合計を確認する。



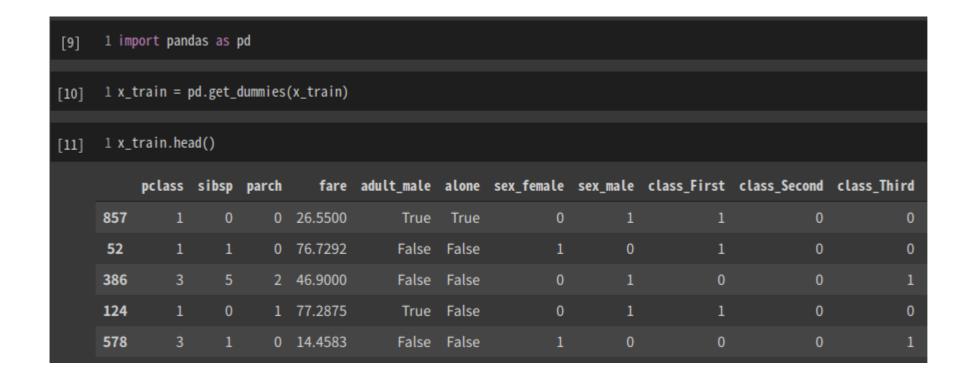
### データの前処理

- ・4つのカラムでnullがあったので、そのカラムを削除する。
- ・カラムが削除されているか確認する。

```
1 x_train = x_train.drop(["age", "embarked", "deck", "embark_town"], axis=1)
[7]
     1 x_train.isnull().sum()
[8]
    pclass
    sex
    sibsp
    parch
    fare
    class
    who
    adult_male
    alive
    alone
    dtype: int64
```

# データ前処理 続き

- ・pandas のメソッド、 get\_dummies() で、文字列でカテゴリー分けされた性別などのデータを、男1 女0 のように変換する。
- ・変換されたことを確認する。



#### ライブラリのインポート

- ・PyTorch ライブラリのインポートする。
- •x\_train と y\_train を tensor型に変換する。

```
[12] 1 import torch
     2 import torch.nn as nn
     3 import torch.optim as optim
     4 from torch.utils.data import Dataset, DataLoader
      6 import numpy as np
     1 data = torch.tensor(x_train.values.astype(np.float32))
     2 print(data[:5])
     3 print(data.shape)
     4 print("-" * 50)
     5 targets = torch.tensor(y_train.values.astype(np.int64))
     6 print(targets[:5])
     7 print(targets.shape)
    tensor([[ 1.0000, 0.0000, 0.0000, 26.5500, 1.0000, 1.0000, 0.0000, 1.0000,
              1.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 1.0000]
            [1.0000, 1.0000, 0.0000, 76.7292, 0.0000, 0.0000, 1.0000, 0.0000,
             1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 1.0000
            [3.0000, 5.0000, 2.0000, 46.9000, 0.0000, 0.0000, 0.0000, 1.0000,
             0.0000, 0.0000, 1.0000, 1.0000, 0.0000, 0.0000, 1.0000, 0.0000
            [1.0000, 0.0000, 1.0000, 77.2875, 1.0000, 0.0000, 0.0000, 1.0000,
             [1.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 1.0000, 0.0000]
            [3.0000, 1.0000, 0.0000, 14.4583, 0.0000, 0.0000, 1.0000, 0.0000,
             0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 1.0000, 1.0000, 0.0000]]
     torch.Size([623, 16])
    tensor([1, 1, 0, 0, 0])
    torch.Size([623])
```

# Datasetとネットワークの作成

- Dataset を作成する。
- ・ネットワークを作成する。自作してみましょう!

\_\_init\_\_\_ : 使う層を定義する

forward:データの流れを定義する

```
[14] 1 class TitanicDataset(Dataset):
      2 def __init__(self, x, y):
        self.data = x
           self.targets = y
      6 def len (self):
           return len(self.targets)
         def __getitem__(self, item):
     10 x = self.data[item]
     11  y = self.targets[item]
           return x, y
     1 train_ds = TitanicDataset(data, targets)
      2 train_loader = DataLoader(train_ds, batch_size=32, shuffle=True)
     1 # class Model(nn.Module):
[16]
             def __init__(self, in_features, mid_features, n_class):
             def forward(self, x):
```

入力層1層、中間層2層、出力層1層 の合計4層のネットワークを実装する。

#### モデルの学習

- 作成したネットワークのインスタンスを生成し、最適化アルゴリズムと損失関数を設定する。
- ・モデルの学習をする。

```
1 model = Model(16, 10, 2)
      2 optimizier = optim.Adam(model.parameters()) # 最適化アルゴリズム
      3 criterion = nn.CrossEntropyLoss() # 損失関数
     1 from tqdm.auto import tqdm
     1 epochs = 10
[19]
      3 with tqdm(total=epochs) as pbar:
        for epoch in range(epochs):
           for batch in train_loader:
         a, b = batch
             optimizier.zero_grad()
             out = model(a)
             loss = criterion(out, b)
             loss.backward()
     11
             optimizier.step()
     12
             pbar.set_description("loss: {:.4f}".format(loss.detach().item()))
     13
           pbar.update(1)
```

#### 学習したモデルを評価

・学習済みのモデルを、テストデータを用いて、f1\_score で評価する。

```
[20] 1 x_test = x_test.drop(["age", "embarked", "deck", "embark_town"], axis=1)
2 x_test = pd.get_dummies(x_test)
3 x_test = torch.tensor(x_test.values.astype(np.float32))
4 with torch.no_grad():
5 result = model(x_test)
6 predicts = torch.max(result, 1)[1]

[21] 1 support.calc_score(predicts) # f1_score, 最大值1、最小值0

0.8323699421965317
```