③ロジスティック回帰_実装演習

In [1]:

```
#from モジュール名 import クラス名（もしくは関数名や変数名）
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

・タイタニックのデータを使用

In [2]:

```
# titanic data csvファイルの読み込み
titanic_df = pd.read_csv('C:/Users/Kadoya Toshiki/Desktop/2.機械学習/機械学習_実習演習用コード/study_ai_ml_google/data/titanic_train.csv')
```

In [3]:

```
# ファイルの先頭部を表示し、データセットを確認する
titanic_df.head(5)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

==========

0.データ前処理

・不要なデータを削除

In [4]:

```
#予測に不要と考えるからうをドロップ（本来はここの情報も使うべき）
titanic_df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)

#一部カラムをドロップしたデータを表示
titanic_df.head()
```

Out[4]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

・nullを含んでいるデータの補完

In [5]:

```
#nullを含んでいる行を表示
titanic_df[titanic_df.isnull().any(1)].head(10)
```

Out[5]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
5	0	3	male	NaN	0	0	8.4583	Q
17	1	2	male	NaN	0	0	13.0000	S
19	1	3	female	NaN	0	0	7.2250	C
26	0	3	male	NaN	0	0	7.2250	C
28	1	3	female	NaN	0	0	7.8792	Q
29	0	3	male	NaN	0	0	7.8958	S
31	1	1	female	NaN	1	0	146.5208	C
32	1	3	female	NaN	0	0	7.7500	Q
36	1	3	male	NaN	0	0	7.2292	C
42	0	3	male	NaN	0	0	7.8958	C

In [6]:

```
#Ageカラムのnullを中央値で補完

titanic_df['AgeFill'] = titanic_df['Age'].fillna(titanic_df['Age'].mean())

#再度nullを含んでいる行を表示 (Ageのnullは補完されている)
titanic_df[titanic_df.isnull().any(1)]
```

Out[6]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	AgeFill
5	0	3	male	NaN	0	0	8.4583	Q	29.699118
17	1	2	male	NaN	0	0	13.0000	S	29.699118
19	1	3	female	NaN	0	0	7.2250	C	29.699118
26	0	3	male	NaN	0	0	7.2250	C	29.699118
28	1	3	female	NaN	0	0	7.8792	Q	29.699118
...	...	...	...	...	...	...	...	...	...
859	0	3	male	NaN	0	0	7.2292	C	29.699118
863	0	3	female	NaN	8	2	69.5500	S	29.699118
868	0	3	male	NaN	0	0	9.5000	S	29.699118
878	0	3	male	NaN	0	0	7.8958	S	29.699118
888	0	3	female	NaN	1	2	23.4500	S	29.699118

179 rows × 9 columns

==========

1.ロジスティック回帰(1変数)

・チケット価格(1変数)から乗客の生死を判別する

In [7]:

```
#運賃だけのリストを作成(説明変数)
data1 = titanic_df.loc[:, ["Fare"]].values
```

In [8]:

```
#生死フラグのみのリストを作成(目的変数)
label1 =  titanic_df.loc[:,["Survived"]].values
```

In [11]:

```
from sklearn.linear_model import LogisticRegression

model=LogisticRegression()

#学習
model.fit(data1, label1)
#予測
model.predict([[61]])
model.predict_proba([[62]])
```

```
C:¥Users¥Kadoya Toshiki¥anaconda3¥lib¥site-packages¥sklearn¥utils¥validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[11]:

```
array([[0.49978123, 0.50021877]])
```

In [12]:

```
print (model.intercept_)

print (model.coef_)
```

```
[-0.94131796]
[[0.01519666]]
```

==========

2.ロジスティック回帰(2変数)

・性別を扱えるようにカテゴリ変数をエンコード

・新しい特徴量(Pclass_Gender)を追加

In [15]:

```
titanic_df['Gender'] = titanic_df['Sex'].map({'female': 0, 'male': 1}).astype(int)
titanic_df['Pclass_Gender'] = titanic_df['Pclass'] + titanic_df['Gender']
titanic_df.head(3)
```

Out[15]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	AgeFill	Gender	Pclas
0	0	3	male	22.0	1	0	7.2500	S	22.0	1	
1	1	1	female	38.0	1	0	71.2833	C	38.0	0	
2	1	3	female	26.0	0	0	7.9250	S	26.0	0	

・使用しない特徴量を削除

In [16]:

```
titanic_df = titanic_df.drop(['Pclass', 'Sex', 'Gender','Age'], axis=1)
titanic_df.head()
```

Out[16]:

	Survived	SibSp	Parch	Fare	Embarked	AgeFill	Pclass_Gender
0	0	1	0	7.2500	S	22.0	4
1	1	1	0	71.2833	C	38.0	1
2	1	0	0	7.9250	S	26.0	3
3	1	1	0	53.1000	S	35.0	1
4	0	0	0	8.0500	S	35.0	4

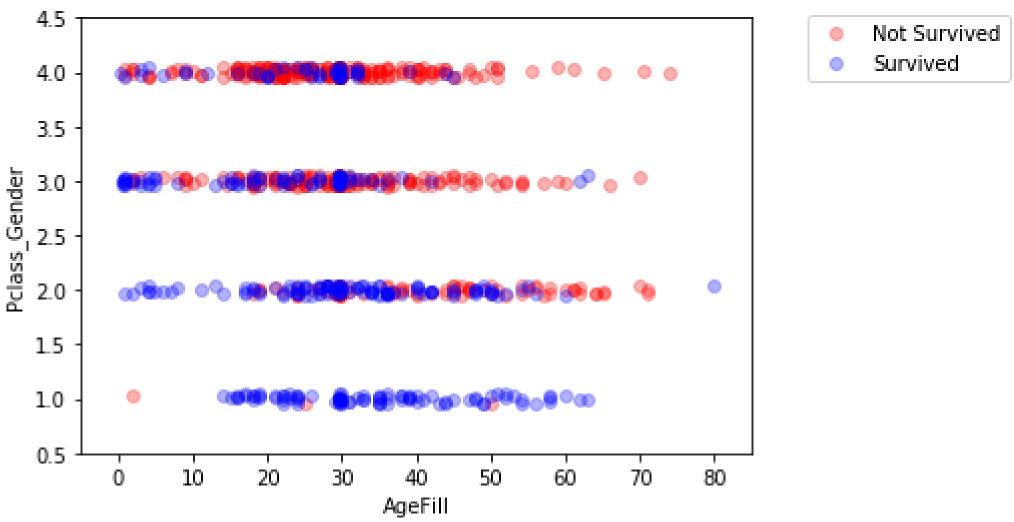
・データを可視化し、関係をみる(AgeFillとPclass_Genderおよび生死)

In [17]:

```
np.random.seed = 0

xmin, xmax = -5, 85
ymin, ymax = 0.5, 4.5

index_survived = titanic_df[titanic_df["Survived"]==0].index
index_notsurvived = titanic_df[titanic_df["Survived"]==1].index

from matplotlib.colors import ListedColormap
fig, ax = plt.subplots()
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
sc = ax.scatter(titanic_df.loc[index_survived, 'AgeFill'],
                titanic_df.loc[index_survived, 'Pclass_Gender']+(np.random.rand(len(index_survived))-0.5)*0.1,
                color='r', label='Not Survived', alpha=0.3)
sc = ax.scatter(titanic_df.loc[index_notsurvived, 'AgeFill'],
                titanic_df.loc[index_notsurvived, 'Pclass_Gender']+(np.random.rand(len(index_notsurvived))-0.5)*0.1,
                color='b', label='Survived', alpha=0.3)
ax.set_xlabel('AgeFill')
ax.set_ylabel('Pclass_Gender')
ax.set_xlim(xmin, xmax)
ax.set_ylim(ymin, ymax)
ax.legend(bbox_to_anchor=(1.4, 1.03))
```

Out[17]:

```
<matplotlib.legend.Legend at 0x17448aa1388>
```



In [18]:

```
#運賃だけのリストを作成
data2 = titanic_df.loc[:, ["AgeFill", "Pclass_Gender"]].values
#生死フラグのみのリストを作成
label2 =  titanic_df.loc[:,["Survived"]].values

model2 = LogisticRegression()

#学習
model2.fit(data2, label2)
#予測
model2.predict([[10,1]])
model2.predict_proba([[10,1]])
```

```
C:¥Users¥Kadoya Toshiki¥anaconda3¥lib¥site-packages¥sklearn¥utils¥validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[18]:

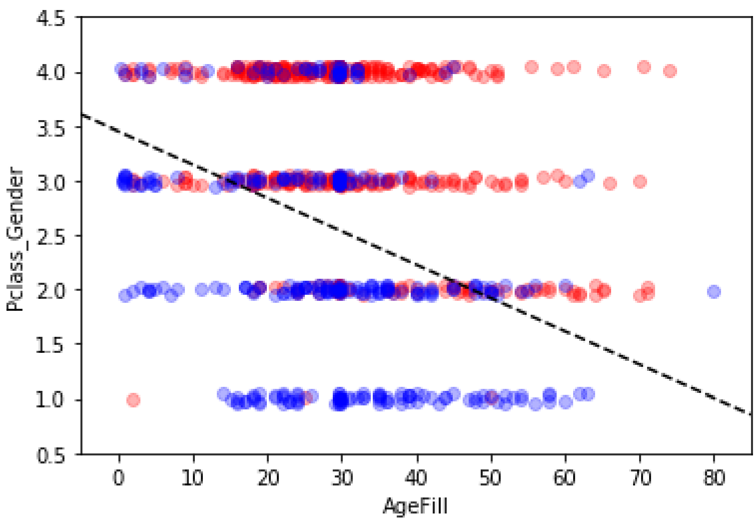
```
array([[0.03754749, 0.96245251]])
```

In [19]:

```
h = 0.02
xmin, xmax = -5, 85
ymin, ymax = 0.5, 4.5
xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))
Z = model2.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
Z = Z.reshape(xx.shape)

fig, ax = plt.subplots()
levels = np.linspace(0, 1.0)
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
#contour = ax.contourf(xx, yy, Z, cmap=cm, levels=levels, alpha=0.5)

sc = ax.scatter(titanic_df.loc[index_survived, 'AgeFill'],
                titanic_df.loc[index_survived, 'Pclass_Gender']+(np.random.rand(len(index_surviv
ed))-0.5)*0.1,
                color='r', label='Not Survived', alpha=0.3)
sc = ax.scatter(titanic_df.loc[index_notsurvived, 'AgeFill'],
                titanic_df.loc[index_notsurvived, 'Pclass_Gender']+(np.random.rand(len(index_not
survived))-0.5)*0.1,
                color='b', label='Survived', alpha=0.3)

ax.set_xlabel('AgeFill')
ax.set_ylabel('Pclass_Gender')
ax.set_xlim(xmin, xmax)
ax.set_ylim(ymin, ymax)
#fig.colorbar(contour)

x1 = xmin
x2 = xmax
y1 = -1*(model2.intercept_[0]+model2.coef_[0][0]*xmin)/model2.coef_[0][1]
y2 = -1*(model2.intercept_[0]+model2.coef_[0][0]*xmax)/model2.coef_[0][1]
ax.plot([x1, x2] ,[y1, y2], 'k--')
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x17448b6ec08>]
```



==========

3.モデル評価

・混同行列と交差検証(クロスバリデーション)

In [20]:

```
from sklearn.model_selection import train_test_split
traindata1, testdata1, trainlabel1, testlabel1 = train_test_split(data1, label1, test_size=0.2)
traindata1.shape
trainlabel1.shape
```

Out[20]:

```
(712, 1)
```

In [21]:

```
traindata2, testdata2, trainlabel2, testlabel2 = train_test_split(data2, label2, test_size=0.2)
traindata2.shape
trainlabel2.shape
#本来は同じデータセットを分割しなければいけない。(簡易的に別々に分割している。)
```

Out[21]:

```
(712, 1)
```

In [22]:

```
data = titanic_df.loc[:, ].values
label =  titanic_df.loc[:,["Survived"]].values
traindata, testdata, trainlabel, testlabel = train_test_split(data, label, test_size=0.2)
traindata.shape
trainlabel.shape
```

Out[22]:

```
(712, 1)
```

In [23]:

```
eval_model1=LogisticRegression()
eval_model2=LogisticRegression()
#eval_model=LogisticRegression()

predictor_eval1=eval_model1.fit(traindata1, trainlabel1).predict(testdata1)
predictor_eval2=eval_model2.fit(traindata2, trainlabel2).predict(testdata2)
#predictor_eval=eval_model.fit(traindata, trainlabel).predict(testdata)
```

```
C:¥Users¥Kadoya Toshiki¥anaconda3¥lib¥site-packages¥sklearn¥utils¥validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:¥Users¥Kadoya Toshiki¥anaconda3¥lib¥site-packages¥sklearn¥utils¥validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:

```
eval_model1.score(traindata1, trainlabel1)
```

Out[25]:

```
0.6629213483146067
```

In [26]:

```
eval_model1.score(testdata1,testlabel1)
```

Out[26]:

```
0.664804469273743
```

In [27]:

```
eval_model2.score(traindata2, trainlabel2)
```

Out[27]:

```
0.7808988764044944
```

In [28]:

```
eval_model2.score(testdata2,testlabel2)
```

Out[28]:

```
0.7430167597765364
```

In [29]:

```
from sklearn import metrics
print(metrics.classification_report(testlabel1, predictor_eval1))
print(metrics.classification_report(testlabel2, predictor_eval2))
```

```
              precision    recall  f1-score   support

           0       0.67      0.92      0.77       112
           1       0.64      0.24      0.35        67

    accuracy                           0.66       179
   macro avg       0.65      0.58      0.56       179
weighted avg       0.66      0.66      0.61       179

              precision    recall  f1-score   support

           0       0.73      0.89      0.80       104
           1       0.78      0.53      0.63        75

    accuracy                           0.74       179
   macro avg       0.76      0.71      0.72       179
weighted avg       0.75      0.74      0.73       179
```

混同行列

・正解率

・適合率

・再現率

・F値

In [30]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix1=confusion_matrix(testlabel1, predictor_eval1)
confusion_matrix2=confusion_matrix(testlabel2, predictor_eval2)
```

In [31]:

```
confusion_matrix1
```

Out[31]:

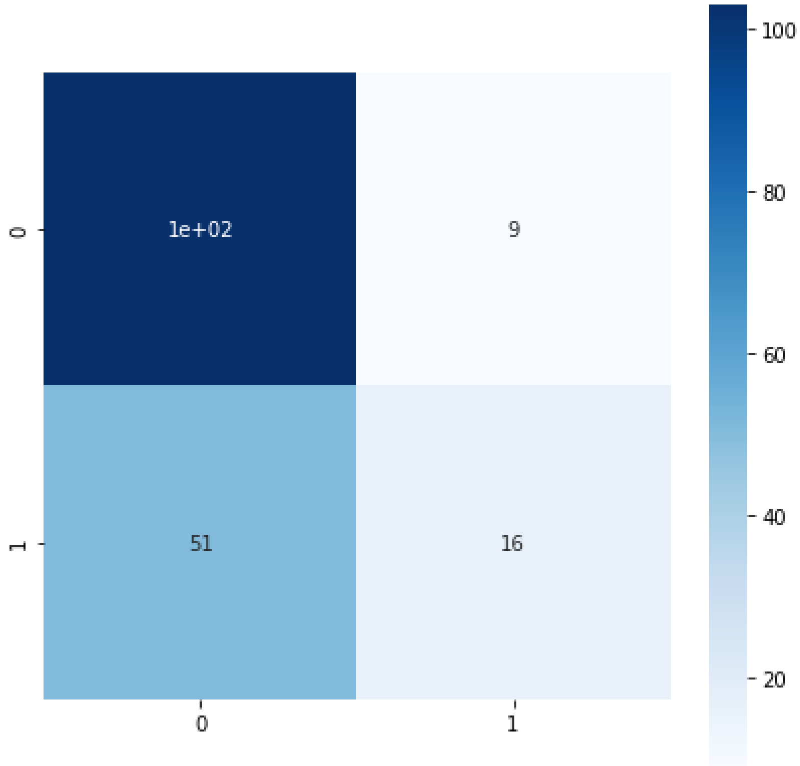
```
array([[103,   9],
       [ 51,  16]], dtype=int64)
```

In [33]:

```
fig = plt.figure(figsize = (7,7))
#plt.title(title)
sns.heatmap(
    confusion_matrix1,
    vmin=None,
    vmax=None,
    cmap="Blues",
    center=None,
    robust=False,
    annot=True, fmt='.2g',
    annot_kws=None,
    linewidths=0,
    linecolor='white',
    cbar=True,
    cbar_kws=None,
    cbar_ax=None,
    square=True, ax=None,
    #xticklabels=columns,
    #yticklabels=columns,
    mask=None)
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x174499c1a88>
```



In [32]:

```
confusion_matrix2
```

Out[32]:

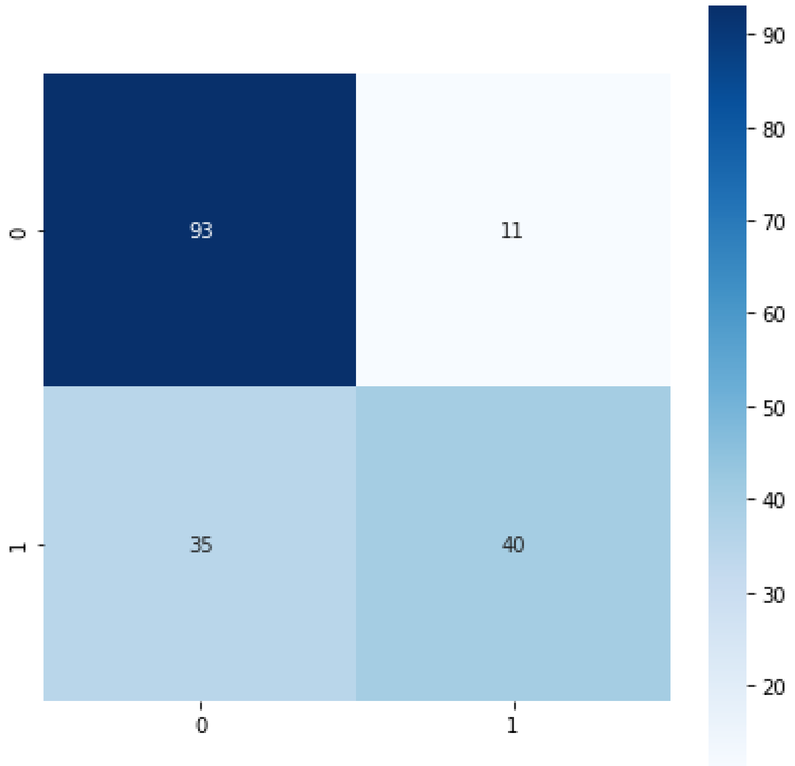
```
array([[93, 11],
       [35, 40]], dtype=int64)
```

In [34]:

```
fig = plt.figure(figsize = (7,7))
#plt.title(title)
sns.heatmap(
    confusion_matrix2,
    vmin=None,
    vmax=None,
    cmap="Blues",
    center=None,
    robust=False,
    annot=True, fmt='.2g',
    annot_kws=None,
    linewidths=0,
    linecolor='white',
    cbar=True,
    cbar_kws=None,
    cbar_ax=None,
    square=True, ax=None,
    #xticklabels=columns,
    #yticklabels=columns,
    mask=None)
```

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x17449a4c4c8>



==========

・学習の前に、データを確認。 ・データの前処理を実施。不要なデータの削除や欠損値の補完。 ・データを視覚化し、関係性を把握。 ・ロジスティック回帰(特徴量：1 or 2つの場合を実施。) ・モデル評価 →混同行列で評価。どの指標を重視するかは、目的に合わせて変化。目的別に最適な指標を選択。 ・交差検証 →訓練データ、テストデータの分割の組み合わせを複数パターンで実施。学習するデータの偏りによる影響を取り除く。

In [ ]:

```

```