

機械学習

線形回帰モデル

回帰問題

- ・回帰で扱うデータ
入力ベクトルの各要素…説明変数（または特徴量）
出力（スカラー値）…目的変数

回帰問題を解くための機械学習モデル

- ・線形回帰モデル
教師あり学習（正解付きデータから学習）
入力とm次元パラメータの線形結合を出力

$$\mathbf{w} = (w_1, w_2, \dots, w_m)^T \in \mathbb{R}^m$$

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b = \sum_{j=1}^m w_j x_j + b$$

線形結合

- ・入力ベクトルとパラメータの内積をとり切片を足し合わせたもの

線形回帰モデルのパラメータ

- ・モデルのパラメータ
特徴量が予測値に与える影響を決定する重みの集合
重みが大きければ予測に大きな影響を与え、重みが0なら予測への影響も0

線形単回帰モデル（m = 1）

- ・データへの仮定
データは、回帰直線に誤差が加わった形で観測されていると仮定する
 $y = (w_0 + w_1 x_1 + \varepsilon)$
切片 w_0 、回帰係数 w_1 を学習で決める
未知パラメータは最小二乗法により推定→誤差は正規分布を仮定しなくてもよい。
最尤法を使う場合は正規分布を仮定するとより詳細な分析が可能となる。

線形重回帰モデル（m = 多次元）

- ・データへの仮定
データは、回帰平面に誤差が加わった形で観測されていると仮定する。
 $y = (w_0 + w_1 x_1 + w_2 x_2 + \varepsilon)$
切片 w_0 、回帰係数 w_1 、 w_2 を学習で決める

データの分割

- ・データを学習用と検証用に分割
→学習用を80%、検証用を20%など。

パラメータの推定

- ・平均二乗誤差（Mean Square Error）
データとモデル出力の二乗誤差
- ・最小二乗法
学習データの平均二乗誤差を最小とするパラメータを探索
→勾配が0となる点を求める

非線形回帰モデル

- ・現実問題、線形なデータは少ない
→非線形回帰モデル
- ・非線形回帰モデル
多項式…次数の高い多項式を考えると非線形のデータをとらえることができる
ガウス型基底

正則化法

- ・未学習
学習データに対して十分小さい誤差を得られない場合
- ・過学習
小さい誤差を得られたが、テスト集合との誤差が大きい場合
→正則化で回避する
- ・汎化性能

新たな入力に対する予測性能

汎化誤差が小さいものが良い性能を持ったモデルといえる。

- 正則化法

モデルの複雑さに伴ってその値が大きくなるペナルティ項（あるいは正則化項）を課した関数の最小化を考える。

正則化パラメータは、モデルの曲線の滑らかさを調節し、推定量の安定に寄与する。

- ペナルティ項

RIDGE推定量：L2ノルムを利用。パラメータを0に近づける（縮小推定）

LASSO推定量：L1ノルムを利用。いくつかのパラメータを0と推定（スパース推定）

モデル選択

- ホールドアウト法

手元のデータを2つに分割。

一方を学習に使用、もう一方をテストに使用し予測精度や誤り率を推定。

- クロスバリデーション

手元の各クラスのデータをm個に分割。

m-1個のデータを使用して識別器を学習、1つのグループのデータでテストを実行。

これをm回繰り返し、それらの誤り率（平均二乗誤差）の平均を予測値とする。

→平均を取るので1回のみよりも結果が安定する。

ロジスティック回帰

- ロジスティック回帰

入力からクラスに分類する問題。

入力はm次元のベクトル、出力は0または1を想定。

- ロジスティック線形回帰モデル

分類問題を解くための機械学習モデル。

入力からそのラベルを予測するシステムを構築。

入力とパラメータの線形結合をシグモイド関数に入力する。出力はy=1となる確率となる。

- シグモイド関数

$$\sigma(x) = \frac{1}{1 + \exp(-ax)}$$

aを増加させるとx=0付近での曲線の勾配が増加する。

シグモイド関数の微分はシグモイド関数自身で表現することが可能。

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x) (1 - \sigma(x))$$

最尤推定

- 尤度

あるデータを得たときに、分布のパラメータが特定の値であることがどれほどありえそうか（もっともらしいか）を表現。

尤度（データを固定してパラメータが変化）⇔確率（パラメータを固定してデータが変化）

- 尤度関数

確率変数はベルヌーイ試行に従う。

$$P(Y=1|x) = p$$

$$P(Y=0|x) = 1 - P(Y=1|x) = 1 - p$$

としたとき、Y=tとなる確率は以下のように表される。

$$P(Y=t|x) = P(Y=1|x)^t P(Y=0|x)^{1-t} \\ = p^t (1-p)^{1-t}$$

- 同時確率

観測されたデータ（学習データ）を発生させるもっともらしい確率分布を求める。

$$P(Y=y_1|x_1) P(Y=y_2|x_2) \cdots P(Y=y_n|x_n) \\ = p_1^{y_1} (1-p_1)^{1-y_1} p_2^{y_2} (1-p_2)^{1-y_2} \cdots p_n^{y_n} (1-p_n)^{1-y_n} \\ = \prod_{i=1}^n p_i^{y_i} (1-p_i)^{1-y_i}$$

$$= L(w_0, w_1, \dots, w_m) \cdots (*)$$

(*) ロジスティック回帰モデルより

$$P(Y=1|x_1) = p_1 = \sigma(w_0 + w_1 x_{11} + \cdots + w_m x_{1m})$$

$$P(Y=2|x_2) = p_2 = \sigma(w_0 + w_1 x_{21} + \cdots + w_m x_{2m})$$

⋮

$$P(Y=n|x_n) = p_n = \sigma(w_0 + w_1 x_{n1} + \cdots + w_m x_{nm})$$

- 対数尤度関数

計算が面倒なので対数で書く。

平均二乗誤差は「最小化」、尤度関数は「最大化」では話がややこしい

→対数尤度関数にマイナスをかけて「最小化」で統一

$$\begin{aligned}
& E(w_0, w_1, \dots, w_m) \\
&= -\log L(w_0, w_1, \dots, w_m) \\
&= \sum_{i=1}^n \{ t_i \log p_i + (1 - t_i) \log (1 - p_i) \}
\end{aligned}$$

勾配降下法

- 勾配降下法の必要性
高次元となると、損失関数を微分して0になる値を解析的に求めることが難しい
→近似的に解を求める必要
- 勾配降下法

$$\begin{aligned}
w^{k+1} &= w^k - \eta \frac{\partial E(w, b)}{\partial w} \\
b^{k+1} &= b^k - \eta \frac{\partial E(w, b)}{\partial b}
\end{aligned}$$

η : 学習率 (ハイパーパラメータ)

対数尤度関数を最適化するため、係数 (w) とバイアス (b) に関して微分する。

$$\begin{aligned}
\frac{\partial E(w, b)}{\partial w} &= \sum_{i=1}^n \frac{\partial E_i}{\partial p_i} \frac{\partial p_i}{\partial w} \\
&= - \sum_{i=1}^n \left(\frac{t_i}{p_i} - \frac{1-t_i}{1-p_i} \right) \frac{\partial p_i}{\partial w} \\
&= - \sum_{i=1}^n \left(\frac{t_i}{p_i} - \frac{1-t_i}{1-p_i} \right) p_i (1-p_i) x_i \cdots \text{シグモイド関数の微分} \\
&= - \sum_{i=1}^n \{ t_i (1-p_i) - p_i (1-t_i) \} x_i \\
&= - \sum_{i=1}^n (t_i - p_i) x_i \\
\frac{\partial E(w, b)}{\partial b} &= - \sum_{i=1}^n (t_i - p_i)
\end{aligned}$$

勾配を逐次的に求める式は以下の通り。

$$\begin{aligned}
w^{k+1} &= w^k + \eta \sum_{i=1}^n (t_i - p_i) x_i \\
b^{k+1} &= b^k + \eta \sum_{i=1}^n (t_i - p_i)
\end{aligned}$$

計算量大→勾配降下法のデメリット
ディープラーニングでは確率的勾配降下法 (SGD) で回避

モデルの評価

- 混同行列

		検証用データの結果	
		生存	死亡
モデルの予測結果	生存	True Positive	False Positive
	死亡	True Negative	False Negative

- 正解率

$$\frac{TP + TN}{TP + FN + FP + TN}$$

- 適合率

見逃しが多くても正確な予測がしたい場合に使用。

$$\frac{TP}{TP + FP}$$

- 再現率

誤りが多少多くても抜け漏れを少なくしたい場合。

$$\frac{TP}{TP + FN}$$

- F値

適合率と再現率の調和平均。

ハンズオン

0. データ表示 ¶

```
In [15]: #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

#matplotlibをinlineで表示するためのおまじない (plt.show()しなくていい)
%matplotlib inline
```

```
In [16]: # titanic data csvファイルの読み込み
titanic_df = pd.read_csv('./data/titanic_train.csv')
```

```
In [17]: # 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(3)
```

```
Out[17]:
```

	PassengerId	Survived	Pclass		Name	Sex	Age	SibSp	Parch		Ticket	Fare	Cabin	Embarked
0	1	0	3		Braund, Mr. Owen Harris	male	22.0	1	0		A/5 21171	7.2500	NaN	S
1	2	1	1		Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0		PC 17599	71.2833	C85	C
2	3	1	3		Heikkinen, Miss. Laina	female	26.0	0	0		STON/O2. 3101282	7.9250	NaN	S

1. ロジスティック回帰

不要なデータの削除・欠損値の補完

```
In [18]: #予測に不要と考えるからうをドロップ (本当はこの情報もしっかり使うべきだと思います)
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[18]:
```

	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

```
In [19]: #nullを含んでいる行を表示
titanic_df[titanic_df.isnull().any(1)].head(10)
```

```
Out[19]:
```

	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 [20]: #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)]

#titanic_df.dtypes
```

Out[20]:

	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
29	0	3	male	NaN	0	0	7.8958	S	29.699118
31	1	1	female	NaN	1	0	146.5208	C	29.699118
32	1	3	female	NaN	0	0	7.7500	Q	29.699118
36	1	3	male	NaN	0	0	7.2292	C	29.699118
42	0	3	male	NaN	0	0	7.8958	C	29.699118
45	0	3	male	NaN	0	0	8.0500	S	29.699118
46	0	3	male	NaN	1	0	15.5000	Q	29.699118
47	1	3	female	NaN	0	0	7.7500	Q	29.699118
48	0	3	male	NaN	2	0	21.6792	C	29.699118
55	1	1	male	NaN	0	0	35.5000	S	29.699118
61	1	1	female	38.0	0	0	80.0000	NaN	38.000000
64	0	1	male	NaN	0	0	27.7208	C	29.699118
65	1	3	male	NaN	1	1	15.2458	C	29.699118
76	0	3	male	NaN	0	0	7.8958	S	29.699118
77	0	3	male	NaN	0	0	8.0500	S	29.699118
82	1	3	female	NaN	0	0	7.7875	Q	29.699118
87	0	3	male	NaN	0	0	8.0500	S	29.699118
95	0	3	male	NaN	0	0	8.0500	S	29.699118
101	0	3	male	NaN	0	0	7.8958	S	29.699118
107	1	3	male	NaN	0	0	7.7750	S	29.699118
109	1	3	female	NaN	1	0	24.1500	Q	29.699118
121	0	3	male	NaN	0	0	8.0500	S	29.699118
126	0	3	male	NaN	0	0	7.7500	Q	29.699118
128	1	3	female	NaN	1	1	22.3583	C	29.699118
140	0	3	female	NaN	0	2	15.2458	C	29.699118
...
727	1	3	female	NaN	0	0	7.7375	Q	29.699118
732	0	2	male	NaN	0	0	0.0000	S	29.699118
738	0	3	male	NaN	0	0	7.8958	S	29.699118
739	0	3	male	NaN	0	0	7.8958	S	29.699118
740	1	1	male	NaN	0	0	30.0000	S	29.699118

主成分分析

- 多変量データの持つ構造をより小数個の指標にまとめる（大きな次元のものを低次元に圧縮）
変量の個数を減らすことに伴う情報の損失はなるべく少なくする必要
→学習データの分散が最大となる方向への線形変換を求める
- 係数ベクトルを変えると線形変換後の値が変わる

$$s_j = (s_{1j}, s_{2j}, \dots, s_{nj})^T = \bar{X} a_j \text{ とおくと分散は}$$

$$\text{Var}(s_j) = \frac{1}{n} s_j^T s_j$$

$$= \frac{1}{n} (\bar{X} a_j)^T (\bar{X} a_j)$$

$$= \frac{1}{n} a_j^T \bar{X} \bar{X} a_j$$

$$= a_j^T \text{Var}(\bar{X}) a_j$$

以下の最適化問題となる。

$$\underset{a \in \mathbb{R}^m}{\text{argmax}} a_j^T \text{Var}(\bar{X}) a_j$$

$$\text{subject to } a_j^T a_j = 1$$

この問題を解くために、以下のラグランジュ関数を置き、係数ベクトル a_j で微分する。

$$E(a_j) = a_j^T \text{Var}(\bar{X}) a_j - \lambda(a_j^T a_j - 1)$$

$$\frac{\partial E(a_j)}{\partial a_j} = 2 \text{Var}(\bar{X}) a_j - 2\lambda a_j = 0$$

$\text{Var}(\bar{X}) a_j = \lambda a_j$... 解は分散共分散行列 $\text{Var}(\bar{X}) = \frac{1}{n} \bar{X} \bar{X}^T$ の固有ベクトル

分散共分散行列は実対象行列であるため、固有ベクトルはすべて直交となる。

- 主成分
 - 第1主成分：最大固有値に対応する固有ベクトルで線形変換された特徴量
 - 第k主成分：k番目の固有値に対応する固有ベクトルで線形変換された特徴量
- 寄与率
 - 第k主成分の寄与率：第k主成分の分散の全分散に対応する割合

k近傍 (kNN) 法

- 分類問題のための機械学習法。
- データから近い順にk個のデータを見て多数決で所属クラスを決定する。
- kを変化させると結果も変わる。kを大きくすると決定境界が滑らかになる。

k平均 (k-means) 法

- 教師なし学習のクラスタリング手法。与えられたデータをk個のクラスタに分類。
- クラスタの中心をランダムに生成。
 - 各データと各クラスタ中心との距離を計算し最も距離が近いクラスタに各データを所属させる。
 - 各クラスタの平均ベクトル (中心) を計算して中心を重心の位置にずらす。
 - クラスタの再割り当てを行い中心の更新を行う。
 - 以上をクラスタの中心が変化しなくなるまで行う。

サポートベクターマシン

- サポートベクターマシン (SVM)
 - 2クラス分類のための機械学習法
 - マージンを最大化するための決定境界 (識別面) を求める
 - 線形モデルの正負で2値分類 (決定境界 $w^T x + b = 0$)
- 目的関数の導出
 - 各点と決定境界との距離

$$\frac{|w^T x_i + b|}{\|w\|} = \frac{t_i (w^T x_i + b)}{\|w\|}$$

マージンは決定境界と最も距離の近い点との距離

$$\min_i \frac{t_i (w^T x_i + b)}{\|w\|}$$

SVMの目標はマージンの最大化

$$\max_{w, b} \left[\min_i \frac{t_i (w^T x_i + b)}{\|w\|} \right]$$

マージン上の点において $t_i (w^T x_i + b) = 1$ が成り立つと仮定すると

すべての点において $t_i (w^T x_i + b) \geq 1$ が成り立つため、目的関数は以下となる

$$\max_{w, b} \frac{1}{\|w\|}$$

SVMの主問題

マージン $1 / \|w\|$ の最大化のため

目的関数：

$$\min_{w, b} L(w, b) = \frac{1}{2} \|w\|^2$$

制約条件：

$$t_i (w^T x_i + b) - 1 \geq 0$$

相対問題の補問題として、ラグランジュ未定乗数を用いて定義したラグランジュ関数を最大化する問題を考える

ラグランジュ関数として以下を定義

$$L(w, b, \lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i (t_i (w^T x_i + b) - 1)$$

$\frac{\partial L}{\partial b} = 0, \frac{\partial L}{\partial w} = 0$ を解くと

$$\sum_{i=1}^n \lambda_i t_i = 0, w = \sum_{i=1}^n \lambda_i t_i x_i \text{ が求まる}$$

この結果をラグランジュ関数に代入

$$\begin{aligned} L(w, b, \lambda) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i (t_i (w^T x_i + b) - 1) \\ &= \frac{1}{2} \left(\sum_{i=1}^n \lambda_i t_i x_i \right)^T \left(\sum_{j=1}^n \lambda_j t_j x_j \right) - \sum_{i=1}^n \lambda_i t_i \left(\sum_{j=1}^n \lambda_j t_j x_j \right)^T x_i \\ &\quad - b \sum_{i=1}^n \lambda_i t_i + \sum_{i=1}^n \lambda_i \\ &= \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j t_i t_j x_i^T x_j \end{aligned}$$

この関数がKKT条件を満たすと考え、制約条件は

$$\lambda_i \geq 0 \quad (i = 1, 2, \dots, n)$$

$$\sum_{i=1}^n \lambda_i t_i = 0$$

補問題まとめ

目的関数：

$$\max_{\lambda} L(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j t_i t_j x_i^T x_j$$

制約条件：

$$\lambda_i \geq 0 \quad (i = 1, 2, \dots, n)$$

$$\sum_{i=1}^n \lambda_i t_i = 0$$

- サポートベクター
分離超平面を構成する学習データはサポートベクターのみ（残りのデータは不要）
- ソフトマージンSVM
サンプルを線形分離できない場合に使用
誤差を許容しペナルティを与える
マージン内に入るデータや誤分類されたデータに対して誤差を表す変数 ξ_i を与える
 $\xi_i = 1 - t_i (w^T x_i + b) > 0$

- ソフトマージンSVMの目的関数と制約条件

マージン1 / $\|w\|$ の最大化のため

目的関数：

$$\min_{w, b} L(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad C \dots \text{トレードオフを制御するパラメータ}$$

制約条件：

$$t_i (w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

相対問題の補問題とした場合

目的関数：

$$\max_{\lambda} L(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j t_i t_j x_i^T x_j$$

制約条件：

$$0 \leq \lambda_i \leq C \quad (i = 1, 2, \dots, n)$$

$$\sum_{i=1}^n \lambda_i t_i = 0$$

パラメータCの大小で決定境界が変化
Cが小さいほど誤差を許容し、大きいほど許容しない

- 非線形分離

線形分離できない場合に使用

特徴空間に写像しその空間で線形に分離する

- カーネルトリック

目的関数が変わる

目的関数：

$$\max_{\lambda} L(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j t_i t_j \phi(x_i)^T \phi(x_j)$$

高次元ベクトルの内積をスカラー関数で表現

特徴空間が高次元でも計算コストを抑えられる

ハンズオン（ボストン住宅価格データ）

線形単回帰分析

```
In [11]: #カラムを指定してデータを表示
df[['RM']].head()
```

```
Out[11]:
```

	RM
0	6.575
1	6.421
2	7.185
3	6.998
4	7.147

```
In [12]: #説明変数
data = df.loc[:, ['RM']].values
```

```
In [13]: #dataリストの表示(1-5)
data[0:5]
```

```
Out[13]: array([[6.575],
               [6.421],
               [7.185],
               [6.998],
               [7.147]])
```

```
In [14]: # 目的変数
target = df.loc[:, 'PRICE'].values
```

```
In [15]: target[0:5]
```

```
Out[15]: array([24. , 21.6, 34.7, 33.4, 36.2])
```

```
In [16]: ## sklearnモジュールからLinearRegressionをインポート
from sklearn.linear_model import LinearRegression
```

```
In [17]: # オブジェクト生成
model = LinearRegression()
#model.get_params()
#model = LinearRegression(fit_intercept = True, normalize = False, copy_X = True, n_jobs = 1)
```

```
In [18]: # fit関数でパラメータ推定
model.fit(data, target)
```

```
Out[18]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [19]: #予測
model.predict(1)
```

```
Out[19]: array([-25.5685118])
```


重回帰分析(2変数)

```
In [20]: #カラムを指定してデータを表示
df[['CRIM', 'RM']].head()

Out[20]:
```

	CRIM	RM
0	0.00632	6.575
1	0.02731	6.421
2	0.02729	7.185
3	0.03237	6.998
4	0.06905	7.147

```
In [21]: #説明変数
data2 = df.loc[:, ['CRIM', 'RM']].values
#目的変数
target2 = df.loc[:, 'PRICE'].values

In [22]: #オブジェクト生成
model2 = LinearRegression(fit_intercept = True, normalize = False, copy_X = True, n_jobs = 1)

In [23]: #関数でパラメータ推定
model2.fit(data2, target2)

Out[23]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [24]: model2.predict([[2, 3]])

Out[24]: array([-4.63273203])
```

回帰係数と切片の値を確認

```
In [25]: #単回帰の回帰係数と切片を出力
print('推定された回帰係数: %.3f, 推定された切片: %.3f' % (model.coef_, model.intercept_))

推定された回帰係数: 9.102, 推定された切片: -34.671

In [26]: #重回帰の回帰係数と切片を出力
print(model2.coef_)
print(model2.intercept_)

[-0.2618229  8.3975317]
-29.301681346741116
```

ハンズオン（タイタニック）

0. データ表示

```
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

#matplotlibをinlineで表示するためのおまじない (plt.show()しなくていい)
%matplotlib inline

In [2]: #titanic data csvファイルの読み込み
titanic_df = pd.read_csv('./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(3)

Out[3]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

1. ロジスティック回帰

不要なデータの削除・欠損値の補完

In [4]:

```
#予測に不要と考えるからうをドロップ (本当はこの情報もしっかり使うべきだと思います)
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

In [5]:

```
#nullを含んでいる行を表示
titanic_df[titanic_df.isnull().any(1)].head(3)
```

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

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)]

#titanic_df.dtypes
```

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	7.0792	Q	29.699118
29	0	3	male	NaN	0	0	7.8958	S	29.699118
31	1	1	female	NaN	1	0	146.5208	C	29.699118
32	1	3	female	NaN	0	0	7.7500	Q	29.699118
36	1	3	male	NaN	0	0	7.2292	C	29.699118
42	0	3	male	NaN	0	0	7.8958	C	29.699118
45	0	3	male	NaN	0	0	8.0500	S	29.699118
46	0	3	male	NaN	1	0	15.5000	Q	29.699118
47	1	3	female	NaN	0	0	7.7500	Q	29.699118
48	0	3	male	NaN	2	0	21.6792	C	29.699118
55	1	1	male	NaN	0	0	35.5000	S	29.699118
61	1	1	female	38.0	0	0	80.0000	NaN	38.000000

64	0	1	male	NaN	0	0	27.7208	C	29.699118
65	1	3	male	NaN	1	1	15.2458	C	29.699118
76	0	3	male	NaN	0	0	7.8958	S	29.699118
77	0	3	male	NaN	0	0	8.0500	S	29.699118
82	1	3	female	NaN	0	0	7.7875	Q	29.699118
87	0	3	male	NaN	0	0	8.0500	S	29.699118
95	0	3	male	NaN	0	0	8.0500	S	29.699118
101	0	3	male	NaN	0	0	7.8958	S	29.699118
107	1	3	male	NaN	0	0	7.7750	S	29.699118
109	1	3	female	NaN	1	0	24.1500	Q	29.699118
121	0	3	male	NaN	0	0	8.0500	S	29.699118
126	0	3	male	NaN	0	0	7.7500	Q	29.699118
128	1	3	female	NaN	1	1	22.3583	C	29.699118
140	0	3	female	NaN	0	2	15.2458	C	29.699118
...
727	1	3	female	NaN	0	0	7.7375	Q	29.699118
732	0	2	male	NaN	0	0	0.0000	S	29.699118
738	0	3	male	NaN	0	0	7.8958	S	29.699118
739	0	3	male	NaN	0	0	7.8958	S	29.699118
740	1	1	male	NaN	0	0	30.0000	S	29.699118
760	0	3	male	NaN	0	0	14.5000	S	29.699118
766	0	1	male	NaN	0	0	39.6000	C	29.699118
768	0	3	male	NaN	1	0	24.1500	Q	29.699118
773	0	3	male	NaN	0	0	7.2250	C	29.699118

773	0	3	male	NaN	0	0	7.2250	C	29.699118
776	0	3	male	NaN	0	0	7.7500	Q	29.699118
778	0	3	male	NaN	0	0	7.7375	Q	29.699118
783	0	3	male	NaN	1	2	23.4500	S	29.699118
790	0	3	male	NaN	0	0	7.7500	Q	29.699118
792	0	3	female	NaN	8	2	69.5500	S	29.699118
793	0	1	male	NaN	0	0	30.6958	C	29.699118
815	0	1	male	NaN	0	0	0.0000	S	29.699118
825	0	3	male	NaN	0	0	6.9500	Q	29.699118
826	0	3	male	NaN	0	0	56.4958	S	29.699118
828	1	3	male	NaN	0	0	7.7500	Q	29.699118
829	1	1	female	62.0	0	0	80.0000	NaN	62.000000
832	0	3	male	NaN	0	0	7.2292	C	29.699118
837	0	3	male	NaN	0	0	8.0500	S	29.699118
839	1	1	male	NaN	0	0	29.7000	C	29.699118
846	0	3	male	NaN	8	2	69.5500	S	29.699118
849	1	1	female	NaN	1	0	89.1042	C	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

実装(チケット価格から生死を判別)

[illegible]

```
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
dtype=int64)
```

```
In [13]: #この絵を描く!!!!!!
model.predict_proba(data1)
```

```
Out[13]: array([[0.69501393, 0.30498607],
 [0.4647799 , 0.5352201 ],
 [0.69285391, 0.30714609],
 ...,
 [0.64097142, 0.35902858],
 [0.6179592 , 0.3820408 ],
 [0.69341474, 0.30658526]])
```

```
In [14]: X_test_value = model.decision_function(data1)
```

```
In [18]: # 決定関数値 (絶対値が大きいほど識別境界から離れている)
#X_test_value = clf.decision_function(X_test)
# 決定関数値をシグモイド関数で確率に変換
X_test_prob = sigmoid(X_test_value)
```

```
In [19]: print(model.intercept_)
print(model.coef_)
```

```
[-0.93290045]
[[0.01506685]]
```

```
In [20]: w_0 = model.intercept_[0]
w_1 = model.coef_[0,0]

# def normal_sigmoid(x):
#     return 1 / (1+np.exp(-x))

def sigmoid(x):
    return 1 / (1+np.exp(-(w_1*x+w_0)))

x_range = np.linspace(-1, 500, 3000)

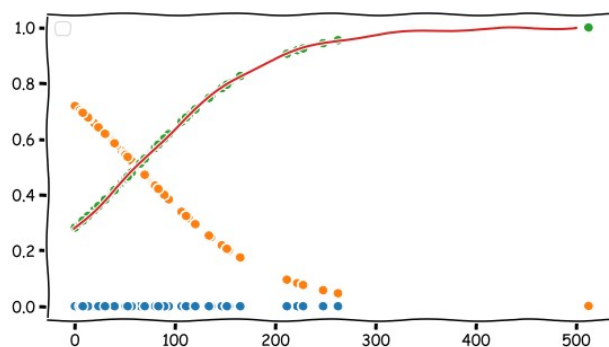
plt.figure(figsize=(9,5))
plt.xticks()
plt.legend(loc=2)

# plt.ylim(-0.1, 1.1)
# plt.xlim(-10, 10)

# plt.plot([-10, 10], [0, 0], "k", lw=1)
# plt.plot([0, 0], [-1, 1.5], "k", lw=1)
plt.plot(data1, np.zeros(len(data1)), 'o')
plt.plot(data1, model.predict_proba(data1), 'o')
plt.plot(x_range, sigmoid(x_range), '-')
# plt.plot(x_range, normal_sigmoid(x_range), '-')
#
```

No handles with labels found to put in legend.

```
Out[20]: [<matplotlib.lines.Line2D at 0x1dff0a6ffd0>]
```



1. ロジスティック回帰

実装(2変数から生死を判別)

```
In [21]: #AgeFillの欠損値を埋めたので
#titanic_df = titanic_df.drop(['Age'], axis=1)
```

```
In [22]: titanic_df['Gender'] = titanic_df['Sex'].map({'female': 0, 'male': 1}).astype(int)
```

```
In [23]: titanic_df.head(3)
```

```
Out[23]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	AgeFill	Gender
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

```
In [24]: titanic_df['Pclass_Gender'] = titanic_df['Pclass'] + titanic_df['Gender']
```

```
In [25]: titanic_df.head()
```

```
Out[25]:
```

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

```
In [26]: titanic_df = titanic_df.drop(['Pclass', 'Sex', 'Gender', 'Age'], axis=1)
```

```
In [27]: titanic_df.head()
```

```
Out[27]:
```

	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

```
In [28]: # 重要だよ!!!
# 境界線の式
#  $w_1 \cdot x + w_2 \cdot y + w_0 = 0$ 
#  $\Rightarrow y = (-w_1 \cdot x - w_0) / w_2$ 

## 境界線 プロット
# plt.plot([-2,2], map(lambda x: (-w_1 * x - w_0)/w_2, [-2,2]))

## データを重ねる
# plt.scatter(X_train_std[y_train==0, 0], X_train_std[y_train==0, 1], c='red', marker='x', label='train 0')
# plt.scatter(X_train_std[y_train==1, 0], X_train_std[y_train==1, 1], c='blue', marker='x', label='train 1')
# plt.scatter(X_test_std[y_test==0, 0], X_test_std[y_test==0, 1], c='red', marker='o', s=60, label='test 0')
# plt.scatter(X_test_std[y_test==1, 0], X_test_std[y_test==1, 1], c='blue', marker='o', s=60, label='test 1')
```

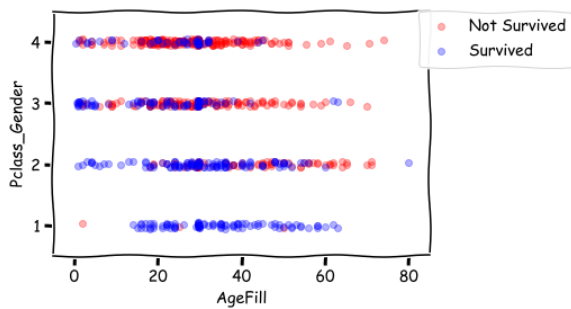
```
In [29]: np.random.seed = 0
```

```
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[29]: <matplotlib.legend.Legend at 0x1dff0b3cd68>



```
In [30]: #運賃だけのリストを作成
data2 = titanic_df.loc[:, ["AgeFill", "Pclass_Gender"]].values
```

```
In [31]: #生死フラグのみのリストを作成
label2 = titanic_df.loc[:,["Survived"]].values
```

```
In [32]: model2 = LogisticRegression()
```

```
Out[33]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)
```

```
In [34]: model2.predict(data2)
```

[illegible]

```
1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, ], dtype=int64)
```

```
In [35]: titanic_df.head(3)
```

Out[35]:

	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

```

In [36]: 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)

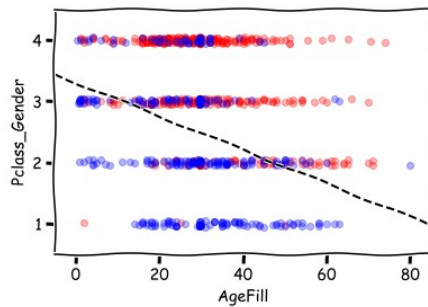
sc = ax.scatter(titanic_df.loc[index_survived, 'AgeFill'],
               titanic_df.loc[index_survived, 'Pclass_Gender']+(np.random.rand(len(index_survived))-0.5)*0.1,
               color='r', label='Not Survived', alpha=0.3)
sc = ax.scatter(titanic_df.loc[index_not_survived, 'AgeFill'],
               titanic_df.loc[index_not_survived, '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[36]: [<matplotlib.lines.Line2D at 0x1dffb0af3588>]



2. モデル評価

混同行列とクロスバリデーション

```
In [37]: from sklearn.model_selection import train_test_split
```

```
In [38]: traindata1, testdata1, trainlabel1, testlabel1 = train_test_split(data1, label1, test_size=0.2)
traindata1.shape
trainlabel1.shape
```

Out[38]: (712, 1)

```
In [39]: 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[39]: (712, 1)

```
In [40]: 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[40]: (712, 1)

```
In [41]: eval_model1=LogisticRegression()
eval_model2=LogisticRegression()
#eval_model=LogisticRegression()
```



```
In [42]: predictor_eval1=eval_model1.fit(traindata1, trainlabel1).predict(testdata1)
        predictor_eval2=eval_model2.fit(traindata2, trainlabel2).predict(testdata2)
        #predictor_eval=eval_model1.fit(traindata, trainlabel).predict(testdata)
```

```
C:\Users\yohei\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\yohei\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\yohei\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\yohei\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
In [43]: eval_model1.score(traindata1, trainlabel1)
```

```
Out[43]: 0.6544943820224719
```

```
In [44]: eval_model1.score(testdata1, testlabel1)
```

```
Out[44]: 0.7039106145251397
```

```
In [45]: eval_model2.score(traindata2, trainlabel2)
```

```
Out[45]: 0.7724719101123596
```

```
In [46]: eval_model2.score(testdata2, testlabel2)
```

```
Out[46]: 0.7821229050279329
```

```
In [47]: from sklearn import metrics
        print(metrics.classification_report(testlabel1, predictor_eval1))
        print(metrics.classification_report(testlabel2, predictor_eval2))
```

```

      precision    recall  f1-score   support

     0       0.70      0.93      0.80      115
     1       0.70      0.30      0.42       64

   micro avg       0.70      0.70      0.70      179
   macro avg       0.70      0.61      0.61      179
weighted avg       0.70      0.70      0.66      179

      precision    recall  f1-score   support

     0       0.80      0.88      0.84      114
     1       0.74      0.62      0.67       65

   micro avg       0.78      0.78      0.78      179
   macro avg       0.77      0.75      0.75      179
weighted avg       0.78      0.78      0.78      179
```

```
In [48]: from sklearn.metrics import confusion_matrix
        confusion_matrix1=confusion_matrix(testlabel1, predictor_eval1)
        confusion_matrix2=confusion_matrix(testlabel2, predictor_eval2)
```

```
In [49]: confusion_matrix1
```

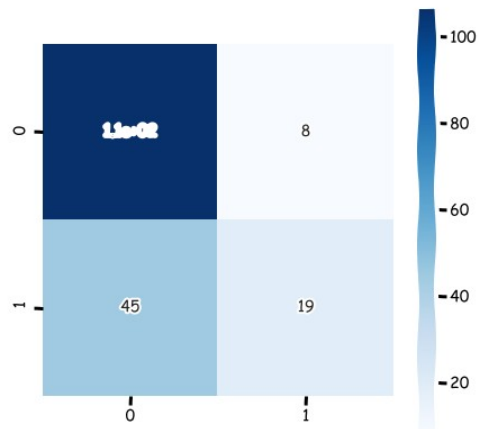
```
Out[49]: array([[107,  8],
               [ 45, 19]], dtype=int64)
```

```
In [50]: confusion_matrix2
```

```
Out[50]: array([[100, 14],
               [ 25, 40]], dtype=int64)
```

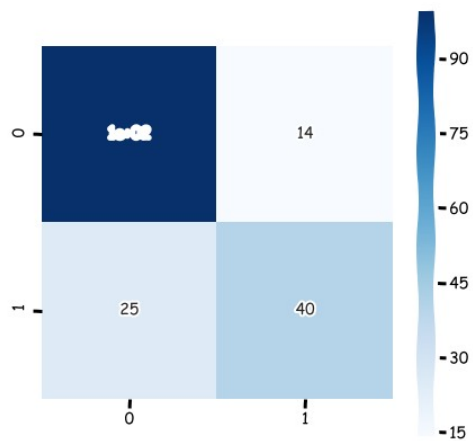
```
In [51]: fig = plt.figure(figsize = (7,7))
        #plt.title(title)
        sns.heatmap(
            confusion_matrix1,
            vmin=None,
            vmax=None,
            cmap="Blues",
            center=None,
            robust=False,
            annot=True, fmt='.2g',
            annot_kws=None,
            linewidths=0,
            linecolor='white',
            cbar=True,
            cbar_kws=None,
            cbar_ax=None,
            square=True, ax=None,
            #ticklabels=columns,
            #yticklabels=columns,
            mask=None)
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1dff19dd0b8>



```
In [52]: 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,
    #ticklabels=columns,
    #yticklabels=columns,
    mask=None)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1dff1a514e0>



In [53]: *#Paired categorical plots*

```
import seaborn as sns
sns.set(style="whitegrid")

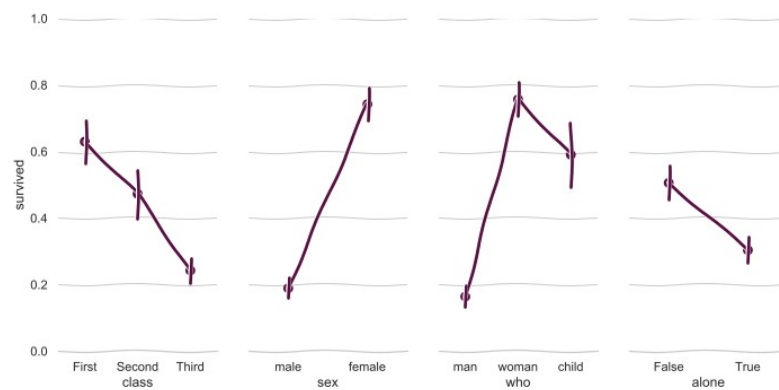
# Load the example Titanic dataset
titanic = sns.load_dataset("titanic")

# Set up a grid to plot survival probability against several variables
g = sns.PairGrid(titanic, y_vars="survived",
                 x_vars=["class", "sex", "who", "alone"],
                 size=5, aspect=.5)

# Draw a seaborn pointplot onto each Axes
g.map(sns.pointplot, color=sns.xkcd_rgb["plum"])
g.set(ylim=(0, 1))
sns.despine(fig=g.fig, left=True)

plt.show()
```

C:\Users\yohel\Anaconda3\lib\site-packages\seaborn\axisgrid.py:1241: UserWarning: The 'size' paramter has been renamed to 'height'; please update your code.
warnings.warn(UserWarning(msg))



In [54]: *#Faceted logistic regression*

```
import seaborn as sns
sns.set(style="darkgrid")

# Load the example titanic dataset
df = sns.load_dataset("titanic")

# Make a custom palette with gendered colors
pal = dict(male="#6495ED", female="#F08080")

# Show the survival probability as a function of age and sex
g = sns.lmplot(x="age", y="survived", col="sex", hue="sex", data=df,
               palette=pal, y_jitter=.02, logistic=True)
g.set(xlim=(0, 80), ylim=(-.05, 1.05))
plt.show()
```

