Логістична регресія побудована на основі даних про смертність на Титаніку

Виконали:
студентка групи ПМ-42
Хорощук Дарія
студент групи ПМ-41
Федишин Любомир

Опис завдання

Отримано дані смертності на Титаніку. Потрібно побудувати логістичну регресію для оцінки виживання пасажира в залежності від класу кабіни, статі, віку, кількості дітей та ціни, яку пасажир заплатив за квиток. Побудувати ROC-криві і оцінити порогове значення.

Оцінити шанси на виживання жінки третього класу без дітей, віком до 25 років. Як зміняться шанси на виживання, якщо жінка буде мати 1 дитину?

Id	вижив	клас	стать	вік	К-сть дітей	Fare
1	0	3	female	18	1	17.8
2	0	3	male	7	4	39.6875
3	0	3	male	21	0	7.8

Аналіз даних

Проаналізувавши дані, ми помітили, що деякі дані пропущено, або вони не коректні. Це стосується віку людей. Оскільки кількість людей з не зазначеним віком - 10, загальна кількість людей 49, 10/49 ≈ 20% даних є неповними або не коректними. Це досить велика частина даних, тому ми заповнили ці пропуски на основі наявних даних.

Id	вижив	клас	стать	вік	К-сть дітей	Fare
7	1	1	male		2	35,50
9	0	3	male	28.5	0	7,23
16	0	1	male		0	27,72
17	1	3	male		1	15,25
28	1	3	male		0	7,90
29	0	3	male		1	8,05
30	1	2	male	0.83	0	29
34	1	3	female		0	7,79
39	0	3	male		0	8,05
47	0	3	male		0	8,05

Аналіз даних

Також було виявлено, що у дітей віком 4, 5, 7, 11 років було вказано кількість дітей, часто навіть більше ніж 1. Такі дані також вважались не коректними, та були змінені.

Id	вижив	клас	стать	вік	К-сть дітей	Fare
2	0	3	male	7	4	39,69
10	1	2	female	5	1	27,75
11	0	3	male	11	5	46,90
15	0	3	male	4	3	27,90

Матриця плану. МНК-оцінка

Для початку задаємо матрицю спостережень, а також вектор результатів. Задаємо матрицю плану, знаходимо МНК-оцінку.

```
features = ['клас', 'стать_binarize', 'вік', 'к-сть_дітей', 'Fare']
X = df[features].to_numpy()
Y = df['вижив'].to_numpy()
```

```
F = np.insert(X, 0, 1, axis=1)
least_squares_value = np.dot(np.dot(np.linalg.inv(np.dot(F.T, F)), F.T), Y)
least_squares_value
```

```
array([ 6.20821083e-01, -1.26336286e-01,  6.13792337e-01, -1.68437210e-03,
       -4.06127757e-02,  1.13685089e-04])
```

Передбачуваний Y. Коефіцієнт детермінації

Обраховуємо передбачувані результати за знайденою МНК-оцінкою та коефіцієнт детермінації.

```
Y_pred = np.dot(F, least_squares_value)
```

```
R2 = 1 - np.sum((Y_pred - Y) ** 2) / np.sum((np.mean(Y) - Y)**2)
print("Коефіцієнт детермінації:", R2, "\n")
```

```
Коефіцієнт детермінації: 0.3940402713388441
```

Коефіцієнт детермінації є досить низьким, що вказує на те, що точність моделі не є високою.

Логістична регресія

Отриманий вектор результатів підставляємо в формулу логістичної регресії.

$$P = \frac{1}{1 + e^{-y}}$$

```
posibility = 1 / (1 + np.exp(-Y_pred))
posibility
```

На основі цих ймовірностей можемо побудувати ROC-криві.

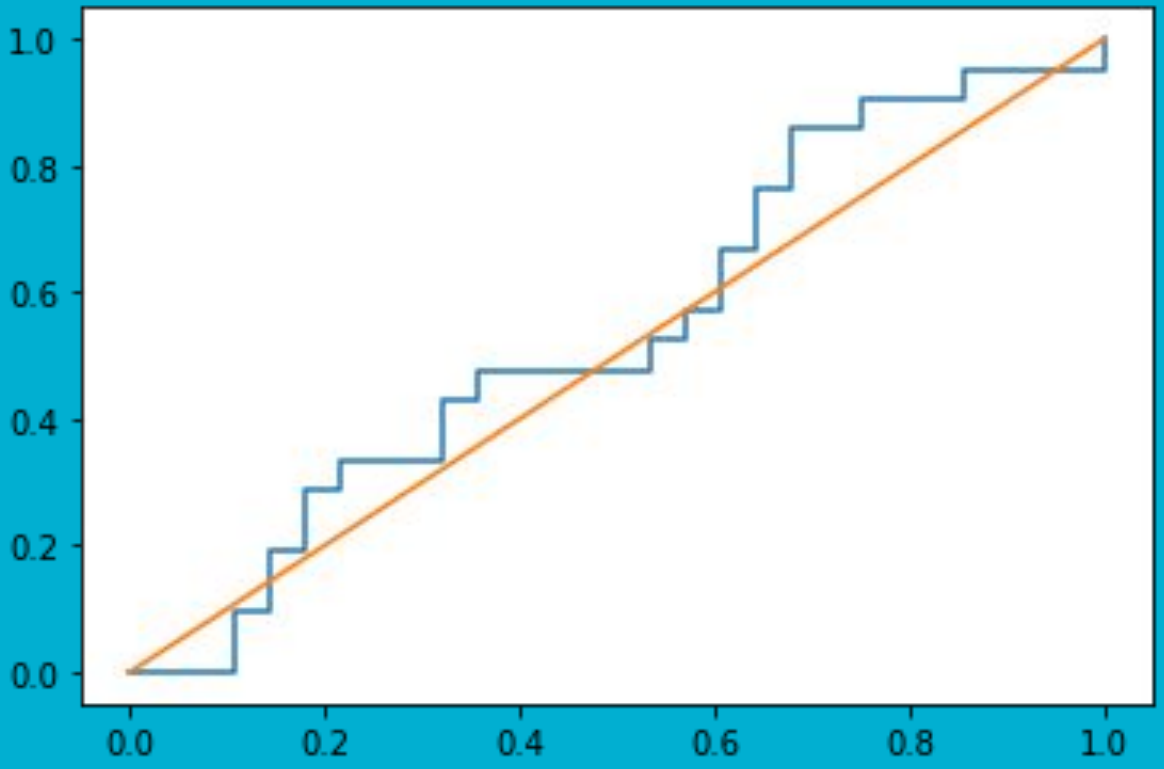
ROC-крива

Побудуємо ROC-криву. На одиничному квадраті проходимось по відсортованих даних, якщо значення більше 0.5, та Y = 1 то робимо крок вгору, якщо ні, то робимо крок вправо.

```
order = np.argsort(posibility)
```

```
n = len(Y)
points_x = np.array([])
points_y = np.array([])
points_x = np.append(points_x, 0)
points_y = np.append(points_y, 0)
for i in range(n):
    if((posibility[i] > 0.5) == (Y[i] == 1)):
        points_x = np.append(points_x, points_x[i])
        points_y = np.append(points_y, points_y[i]+1)
    else:
        points_x = np.append(points_x, points_x[i]+1)
        points_y = np.append(points_y, points_y[i])
points_x = (points_x-points_x.min())/(points_x.max()-points_x.min())
points_y = (points_y-points_y.min())/(points_y.max()-points_y.min())
```

ROC-крива



AUC

Обчислимо AUC.

```
AUC = 0
for i in range(n):
    AUC += (points_x[i+1] - points_x[i]) * (points_y[i])
print(AUC)
```

```
0.5408163265306123
```

AUC ≈ 0.54 < 0.6. Це означає, що дана модель є незадовільною

Порогове значення

При обчисленні порогового значення, вважається, що воно найкраще якщо в нього найбільший AUC.

```
thresholds = np.linspace(0,1,n)
max_auc = 0
better_thresholds = 0
for i in thresholds:
    points_x = np.array([])
    points_y = np.array([])
    points_x = np.append(points_x, 0)
    points_y = np.append(points_y, 0)
    for j in range(n):
        if((posibility[j] > i) == (Y[j] == 1)):
            points_x = np.append(points_x, points_x[j])
            points_y = np.append(points_y, points_y[j]+1)
        else:
            points_x = np.append(points_x, points_x[j]+1)
            points_y = np.append(points_y, points_y[j])
    points_x = (points_x-points_x.min())/(points_x.max()-points_x.min())
    points_y = (points_y-points_y.min())/(points_y.max()-points_y.min())
    AUC = 0
    for j in range(n):
        AUC += (points_x[j+1]-points_x[j])*(points_y[j+1])
    if(abs(AUC) > max_auc):
        max_auc = abs(AUC)
        better_thresholds = i
```

better_thresholds
0.7083333333333333

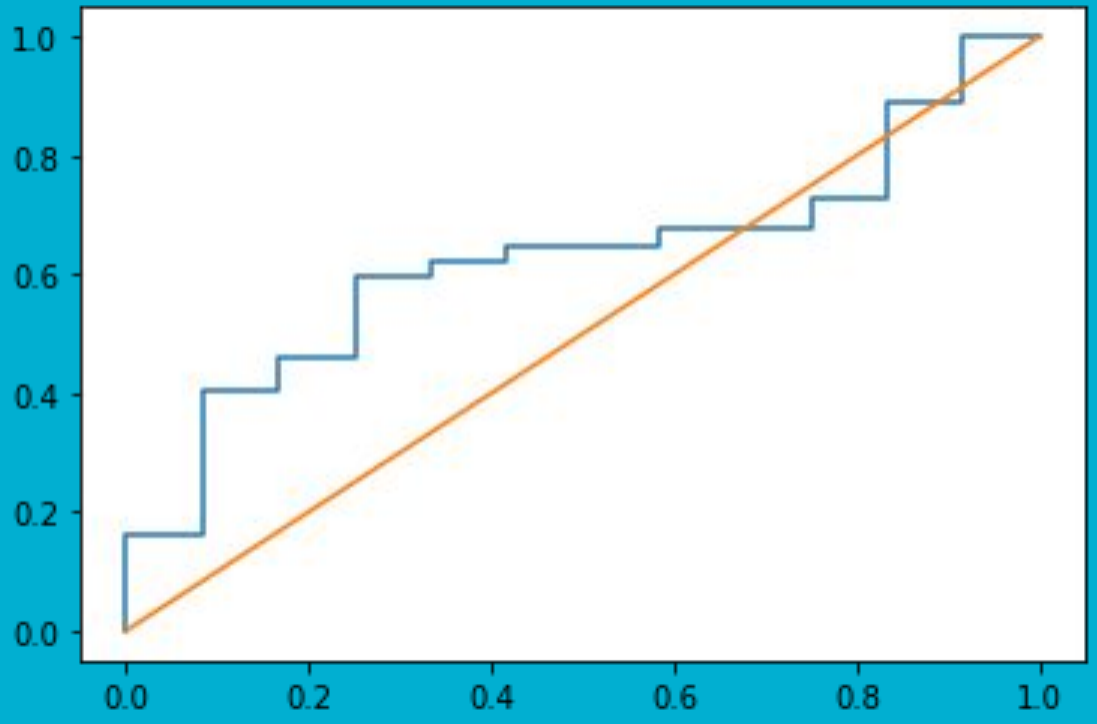
ROC-крива з новим пороговим значенням

Побудуємо ROC-криву за тим же методом, що й минулого разу, проте з новим пороговим значенням.

```
n = len(Y)
points_x = np.array([])
points_y = np.array([])
points_x = np.append(points_x, 0)
points_y = np.append(points_y, 0)
for i in range(n):
    if((posibility[i] > better_thresholds) == (Y[i] == 1)):
        points_x = np.append(points_x, points_x[i])
        points_y = np.append(points_y, points_y[i]+1)
    else:
        points_x = np.append(points_x, points_x[i]+1)
        points_y = np.append(points_y, points_y[i])
points_x = (points_x-points_x.min())/(points_x.max()-points_x.min())
points_y = (points_y-points_y.min())/(points_y.max()-points_y.min())
```

```
plt.plot(points_x, points_y)
plt.plot([0,1],[0,1])
```

ROC-крива з новим пороговим значенням



AUC нової ROC-кривої

Обчислимо AUC.

```
AUC = 0
for i in range(n):
    AUC += (points_x[i+1]-points_x[i])*(points_y[i])
print(AUC)
```

```
0.6261261261261262
```

AUC ≈ 0.62.

Результат є кращим ніж при пороговому значенні 0.5, отже для нашої моделі краще використовувати поріг, що дорівнює 0.708.

При цьому 0.6<0.62<0.7. Це означає, що дана модель є середньою.

Оцінка шансу

Оцінимо шанси на виживання жінки третього класу без дітей, віком до 25 років.

```
X_test1 = np.array([1, 3, 1, 25, 0, meanFare])
```

```
test1 = 1 / (1 + np.exp(-np.dot(X_test1, least_squares_value)))
test1
```

```
0.6932511305074461
```

Ймовірність становить 0.69. Це означає, що в неї відносно високі шанси на виживання, 69%.

Оцінка шансу

Якщо ця ж жінка матиме дитину.

```
X_test2 = np.array([1, 3, 1, 25, 1, meanFare])
```

```
test2 = 1 / (1 + np.exp(-np.dot(X_test2, least_squares_value)))
test2
```

```
0.684547547783276
```

Ймовірність становить 0.68, тобто 68%, що на 1% нище ніж якби у неї не було дитини. Різниця між ймовірностями мала.

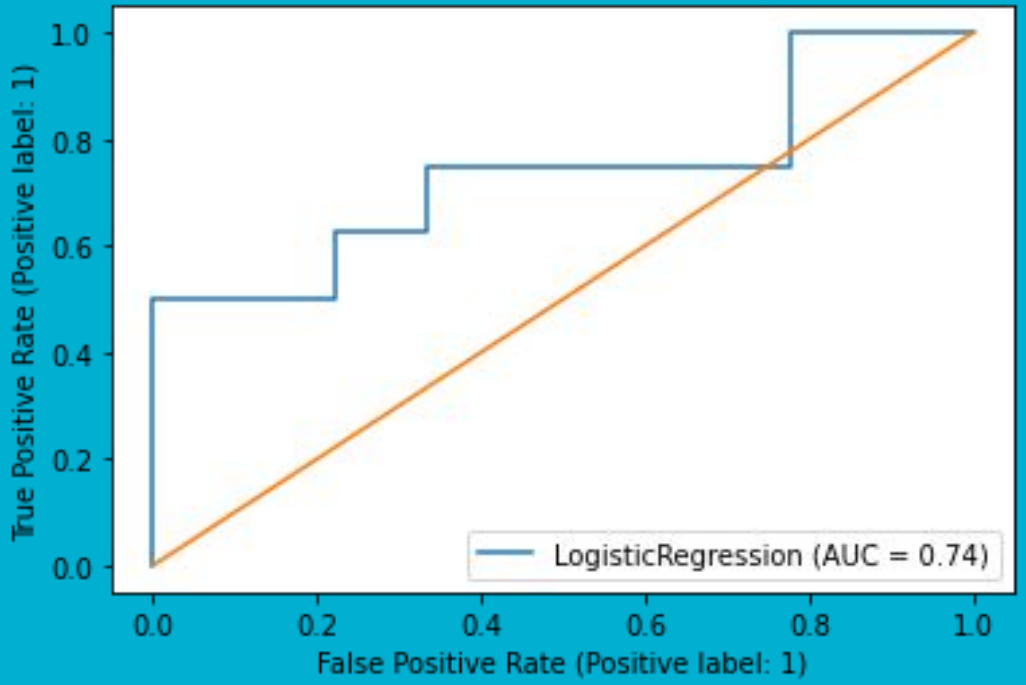
Логістична регресія

Модель, що ми отримали є незадовільною, спробуємо побудувати модель за допомогою інструментів бібліотеки sklearn.linear_model.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import plot_roc_curve

X_train, X_test, y_train, y_test = train_test_split(df[features], df['вижив'],
                                                    test_size=0.33, random_state=42)

clf = LogisticRegression()
clf.fit(X_train,y_train)

plot_roc_curve(clf, X_test, y_test)
plt.plot([0,1],[0,1])
plt.show()
```

Логістична регресія

В результаті отримаємо:



Висновки

Точність нашої моделі не є задовільною. Порівнявши результати отримані нашою моделлю з результатами отриманими моделлю побудованою за допомогою інструментів бібліотеки sklearn.linear_model, можемо зробити висновок, що для логістичної регресії краще використовувати оцінку максимальної правдоподібності, яку використовує функція LogisticRegression(), замість оцінки методом найменших квадратів. Хорошою, вважають моделі з AUC > 0.9, тому для покращення нашої моделі варто розширити набір даних. Або ж наші дані є не коректні і не підходять для побудови подібної моделі.

Дякуємо за увагу!