

Python大数据分析原理与应用 Assignment 1

作者: 付大为 学号: 2201110122

1. 读取患者的检查样本数据，并补全缺失数据（均值填充即可），之后划分训练样本和测试样本，搭建逻辑回归模型，并计算在测试集上预测的准确率。

导入第三方库并读取样本数据:

```
1 import pandas as pd
2 import numpy as np
3 from sklearn import linear_model
4 from sklearn.metrics import accuracy_score, confusion_matrix
5 from sklearn.model_selection import train_test_split
6
7 df = pd.read_csv('./breast_cancer.csv')
```

补全缺失数据(均值填充):

```
1 df.fillna({k: df.mean(skipna=True)[k] for k in df.columns if
            df[k].isnull().any()}, inplace=True)
```

获取feature数据与label数据, 分别记为X和y:

```
1 X = df.iloc[:, :-1].values
2 y = df.iloc[:, -1].values
```

按照80% : 20%比例划分训练样本和测试样本:

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y,
            test_size=0.2)
```

搭建逻辑回归模型：

```
1 model = linear_model.LogisticRegression()
```

拿训练集进行拟合：

```
1 model.fit(X_train, y_train)
```

计算在测试集上预测的准确率：

```
1 y_predict = model.predict(X_test)
2 print(accuracy_score(y_true=y_test, y_pred=y_predict))
3 # 0.9824561403508771
```

2. 计算测试集上预测结果的混淆矩阵

计算测试集上的混淆矩阵：

```
1 cm = confusion_matrix(y_true=y_test, y_pred=y_predict)
2 pd.DataFrame(data={'预测不患癌症': cm[:, 0], '预测患癌症': cm[:, 1]},
               index=['实际患癌症', '实际不患癌症'])
```

结果如下

	预测不患癌症	预测患癌症
实际患癌症	45	0
实际不患癌症	2	67

3. 输出逻辑回归模型的参数 k_0 - k_{30} ，对每一个测试样本计算对应的 y 和 $f(y)$ 值，画出 y 与 $f(y)$ 的散点图，其中正样本以红色表示，负样本以蓝色表示。（正/负样本指数据集中的真实正/负样本）

$$y = k_0 + k_1x_1 + k_2x_2 + \cdots + k_{30}x_{30}$$

$$f(y) = \frac{1}{1 + e^{-y}}$$

首先查看fitting后的model参数

```
1 print('k_0:', model.intercept_)
2 print('k_1 ~ k_30:', model.coef_)
```

```
k_0: [0.21647276]
k_1 ~ k_30: [[ 1.08593442  0.29773749  0.19279967 -0.01243578 -0.04737156 -0.20032599
 -0.28848246 -0.12606566 -0.05770798 -0.01090269  0.05240462  0.55950665
  0.28184453 -0.08787419 -0.00439637 -0.03840022 -0.06108633 -0.01691103
 -0.01742445 -0.00248244  1.149355   -0.43558281 -0.21776817 -0.01599967
 -0.08216659 -0.5775676  -0.75738065 -0.23796901 -0.19666855 -0.04783856]]
```

然后在测试集上计算 y 和 $f(y)$

```
1 y = np.dot(X_test, model.coef_.reshape(-1))
2 f_y = 1 / (1 + np.exp(-y))
```

接下来准备可视化

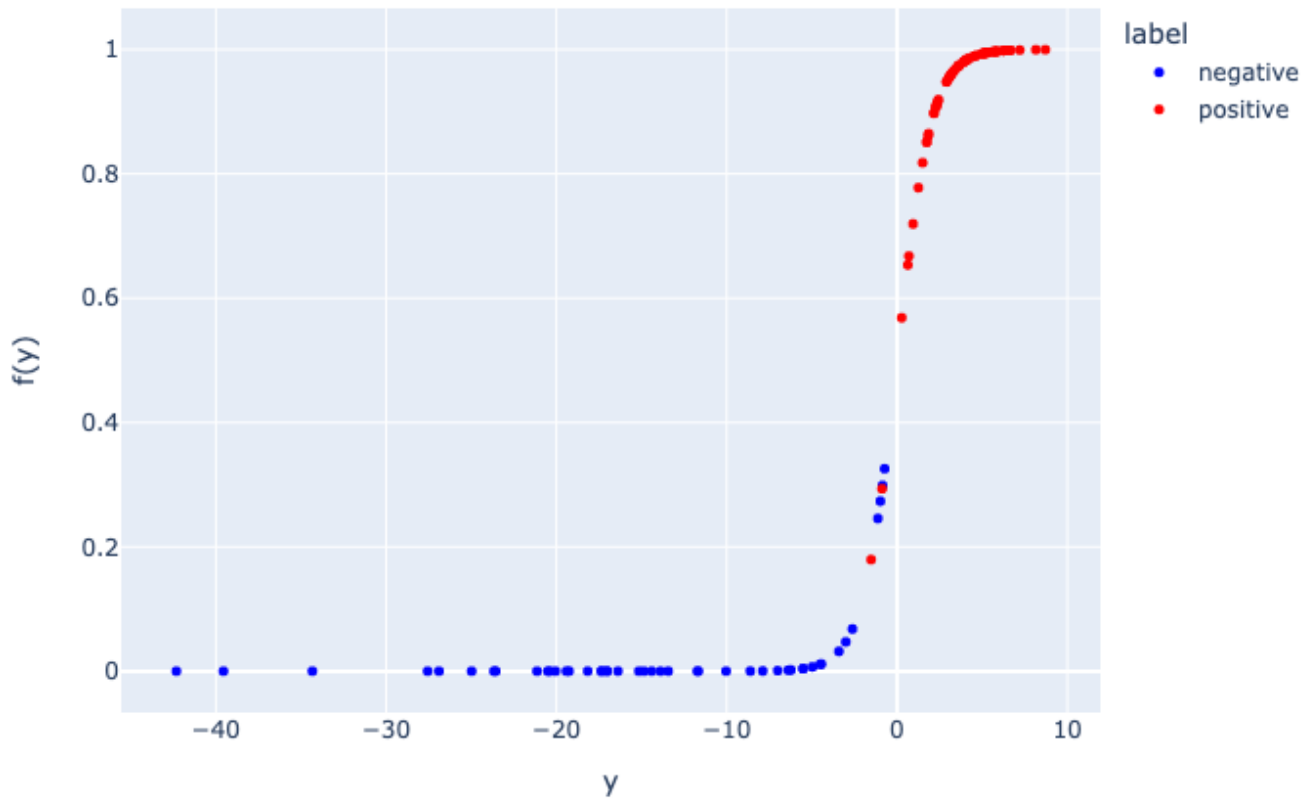
```
1 df_scatter = pd.DataFrame({'y': y, 'f(y)': f_y, 'label':
    ['positive' if label==1 else 'negative' for label in y_test]})
```

我选择用第三方库plotly画图

```

1 import plotly.express as px
2 fig = px.scatter(df_scatter, x="y", y="f(y)", color='label',
3                 color_discrete_sequence=['red', 'blue'])
4 fig.update_traces(marker=dict(size=5))
5 fig.show()
6 fig.write_image('./scatter.pdf')

```



可以看到 $f(y)$ - y 曲线很好地符合了logistic regression的特点, 并且存在测试集中两个正样本被误标记为负样本的问题

完整的程序参考见下面附加的pdf

```
In [1]: import pandas as pd
import numpy as np
```

```
In [2]: df = pd.read_csv('./breast_cancer.csv')
df.fillna({k: df.mean(skipna=True)[k] for k in df.columns if df[k].isnull().any()}, inplace=True)
```

```
In [3]: X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

```
In [4]: from sklearn import linear_model
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

/Users/DaweiFu/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >= 1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

```
In [5]: model = linear_model.LogisticRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print(accuracy_score(y_true=y_test, y_pred=y_predict))
```

0.9824561403508771
/Users/DaweiFu/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(

```
In [6]: cm = confusion_matrix(y_true=y_test, y_pred=y_predict)
pd.DataFrame(data={'预测不患癌症': cm[:, 0], '预测患癌症': cm[:, 1]}, index=['实际患癌症', '实际不患癌症'])
```

Out[6]:

	预测不患癌症	预测患癌症
实际患癌症	45	0
实际不患癌症	2	67

```
In [7]: print('k_0:', model.intercept_)
print('k_1 ~ k_30:', model.coef_)
```

k_0: [0.21647276]
k_1 ~ k_30: [[1.08593442 0.29773749 0.19279967 -0.01243578 -0.04737156 -0.20032599
-0.28848246 -0.12606566 -0.05770798 -0.01090269 0.05240462 0.55950665
0.28184453 -0.08787419 -0.00439637 -0.03840022 -0.06108633 -0.01691103
-0.01742445 -0.00248244 1.149355 -0.43558281 -0.21776817 -0.01599967
-0.08216659 -0.5775676 -0.75738065 -0.23796901 -0.19666855 -0.04783856]]

```
In [8]: model.intercept_ + X_test * model.coef_
```

Out[8]: array([[15.10463364, 6.41834466, 17.6070034 , ..., 0.17944478,
0.15361749, 0.21096654],
[13.51916939, 6.89770202, 15.28955132, ..., 0.20139742,
0.15550551, 0.21254856],
[14.31190152, 5.97769317, 16.51190125, ..., 0.19301377,
0.14575075, 0.21208787],
...,
[15.65846019, 8.50846183, 18.06008264, ..., 0.19691408,
0.1793024 , 0.21274326],
[12.45495366, 4.83140384, 14.36411288, ..., 0.19678796,
0.17405135, 0.21148319],
[15.7344756 , 5.22441733, 17.62628337, ..., 0.20854125,
0.16813163, 0.21354504]])

```
In [9]: np.shape(X_test), np.shape(model.coef_), np.shape(np.dot(X_test, model.coef_.reshape(-1)))
```

Out[9]: ((114, 30), (1, 30), (114,))

```
In [10]: y = np.dot(X_test, model.coef_.reshape(-1))
```

```
In [11]: f_y = 1 / (1 + np.exp(-y))
```

```
In [12]: df_scatter = pd.DataFrame({'y': y, 'f(y)': f_y, 'label': ['positive' if label==1 else 'negative' for label in y]})
```

```
In [14]: import plotly.express as px
fig = px.scatter(df_scatter, x="y", y="f(y)", color='label', color_discrete_sequence=['blue', 'red'])
fig.update_traces(marker=dict(size=5))
fig.show()
fig.write_image('./scatter.png')
```

```
In [ ]:
```