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:

```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split

df = pd.read_csv('./breast_cancer.csv')
```

补全缺失数据(均值填充):

```
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:

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

按照80% : 20%比例划分训练样本和测试样本:

```
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:

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

拿训练集进行拟合:

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

计算在测试集上预测的准确率:

```
y_predict = model.predict(X_test)
print(accuracy_score(y_true=y_test, y_pred=y_predict))
# 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:

```
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'])
```

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k0-k30，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

```
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]]
```

然后在测试集上计算 y 和 f(y)

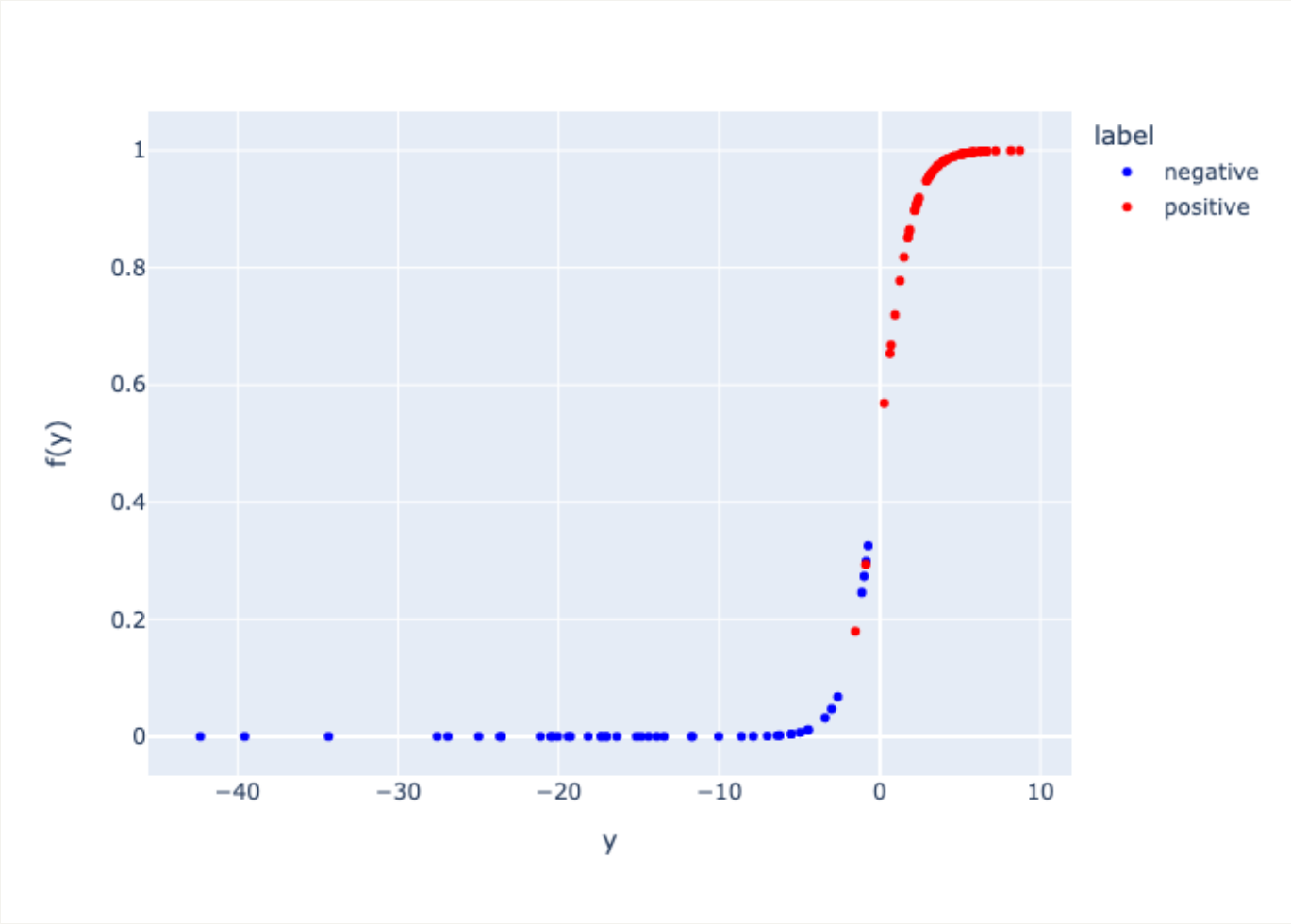
```
y = np.dot(X_test, model.coef_.reshape(-1))
f_y = 1 / (1 + np.exp(-y))
```

接下来准备可视化

```
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

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