logistic regression

March 20, 2024

[6]:

```
import numpy as np
import pandas as pd
```

[7]:

```
data=pd.read_csv("C:\\Users\\user\\Downloads\\archive\\Training.csv")
data
```

[7]:

```
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
…           …      …            …            …      …   …
2455            3      126             88             41      235  39.3
2456            4      123             62              0        0  32.0
2457            1       80             74             11       60  30.0
2458            1       96             64             27       87  33.2
2459            6      105             70             32       68  30.8

      DiabetesPedigreeFunction  Age  Outcome
0                        0.627   50        1
1                        0.351   31        0
2                        0.672   32        1
3                        0.167   21        0
4                        2.288   33        1
…                        …  …      …
2455                     0.704   27        0
2456                     0.226   35        1
2457                     0.527   22        0
2458                     0.289   21        0
2459                     0.122   37        0

[2460 rows x 9 columns]
```

Assigning x and y

[8]:

```
x=data.drop(['Outcome'],axis=1)
x
```

[8]:

```
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
…             …      …             …             …      …    …
2455            3      126             88             41      235  39.3
2456            4      123             62              0        0  32.0
2457            1       80             74             11       60  30.0
2458            1       96             64             27       87  33.2
2459            6      105             70             32       68  30.8

      DiabetesPedigreeFunction  Age
0                        0.627   50
1                        0.351   31
2                        0.672   32
3                        0.167   21
4                        2.288   33
…                        …   …
2455                     0.704   27
2456                     0.226   35
2457                     0.527   22
2458                     0.289   21
2459                     0.122   37

[2460 rows x 8 columns]
```

[9]:

```
y=data['Outcome']
y
```

[9]:

```
0       1
1       0
2       1
3       0
4       1
       ..
2455    0
2456    1
2457    0
2458    0
2459    0
Name: Outcome, Length: 2460, dtype: int64
```

splitting the data set

[10]:

```
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,random_state=42)
```

standardize

[11]:
```
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# x_train_scaled = scaler.fit_transform(x_train)
# x_test_scaled = scaler.transform(x_test)
```

choosing and building the model

[12]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

[12]:
```
array([0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 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, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

[13]:
```
from sklearn.metrics import␣
 ↪mean_absolute_error,mean_squared_error,r2_score,accuracy_score
print(mean_absolute_error(y_test,y_pred))
```

```
0.2703252032520325
```

[14]:
```
print(r2_score(y_test,y_pred))
```

```
-0.14729552029455606
```

[15]:

```
print(mean_squared_error(y_test,y_pred))
```

```
0.2703252032520325
```

[16]:

```
#accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.7296747967479674
```

MODEL OPTIMIZATION

[17]:

```
from sklearn.model_selection import GridSearchCV
model = LogisticRegression(max_iter=1000)
param_grid = {'C':[3.0,10.5]}
```

[18]:

```
#fitting parameters into the grid-seacrh
data= GridSearchCV(model,param_grid,cv=5,)
best_data = data.fit(x_train,y_train)
```

[19]:

```
#getting th best parameters
best_params = data.best_params_
best_params
```

[19]:

```
{'C': 3.0}
```

[20]:

```
#training the model using the best parameters
model=LogisticRegression(** best_params)
model.fit(x_train,y_train)
```

```
D:\tendomatic\Lib\site-packages\sklearn\linear_model\_logistic.py:460:
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(
```

[20]:

```
LogisticRegression(C=3.0)
```

[21]:

```
#making predictions
y_pred_best = model.predict(x_test)
y_pred_best
```

[21]:
```
array([0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 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, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

[22]:
```
#evaluating model performance
print(mean_absolute_error(y_test,y_pred))
```

```
0.2703252032520325
```

[23]:
```
print(r2_score(y_test,y_pred))
```

```
-0.14729552029455606
```

[24]:
```
print(mean_squared_error(y_test,y_pred))
```

```
0.2703252032520325
```

[25]:
```
#finding accuracy on train values
model.score(x_train,y_train)
```

[25]:
```
0.7449186991869918
```

[27]:
```
#finding accuracy on test values
model.score(x_test,y_test)
```

[27]:
```
0.7276422764227642
```