logistic-heart

March 19, 2024

[1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv("heart.csv")
df
```

[2]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0      2.3
1      37    1   2       130   250    0        1      187      0      3.5
2      41    0   1       130   204    0        0      172      0      1.4
3      56    1   1       120   236    0        1      178      0      0.8
4      57    0   0       120   354    0        1      163      1      0.6
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298    57    0   0       140   241    0        1      123      1      0.2
299    45    1   3       110   264    0        1      132      0      1.2
300    68    1   0       144   193    1        1      141      0      3.4
301    57    1   0       130   131    0        1      115      1      1.2
302    57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

[ ]:
```

```

[3]:
```
df.shape
```

[3]:
```
(303, 14)
```

[4]:
```
df.isnull().sum()
```

[4]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[5]:
```
df['target'].value_counts()
```

[5]:
```
target
1    165
0    138
Name: count, dtype: int64
```

[6]:
```
x = df.drop("target", axis=1)
x
```

[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
..	…	…	..	…	…	…	…	…	…	…	…	…	…
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

```
4        2   0     2
..     …  ..   …
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

[7]:
```
y = df["target"]
```

[8]:
```
y
```

[8]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

[9]:
```
df.isnull().sum()
```

[9]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[10]:
```
from sklearn.model_selection import train_test_split
```

[11]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42, stratify=y)
```

[12]:
```
from sklearn.linear_model import LogisticRegression
```

[13]:
```
model = LogisticRegression(max_iter=1000).fit(x_train, y_train)
model
```

[13]:
```
LogisticRegression(max_iter=1000)
```

[14]:
```
y_pred = model.predict(x_test)
y_pred
```

[14]:
```
array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

[15]:
```
from sklearn.metrics import accuracy_score, mean_absolute_error,␣
↪mean_squared_error, r2_score
```

[16]:
```
accuracy = accuracy_score(y_test, y_pred)
```

[17]:
```
print(f"Accuracy = ", accuracy)
```

```
Accuracy =  0.8032786885245902
```

OPTIMIZATION

[18]:
```
from sklearn.model_selection import GridSearchCV
```

[19]:
```
model = LogisticRegression()
model
```

[19]:
```
LogisticRegression()
```

[20]:
```
param_grid = {
    'penalty' :['l2', None],
    'solver':['liblinear','newton-cg', 'newton-cholesky', 'sag', 'saga'],
    'C':[1.0, 1.5]
}
```

[21]:
```
grid_search = GridSearchCV(model, param_grid, cv=5, n_jobs=-1)
grid_search.fit(x_train, y_train)
```

[21]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'C': [1.0, 1.5], 'penalty': ['l2', None],
                         'solver': ['liblinear', 'newton-cg', 'newton-cholesky',
                                    'sag', 'saga']})
```

[23]:
```
best_params = grid_search.best_params_
print("Best Parameters :", best_params)
```

```
Best Parameters : {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
```

[24]:
```
best_model = LogisticRegression(**best_params)
best_model.fit(x_train, y_train)
best_model
```

[24]:
```
LogisticRegression(solver='newton-cg')
```

[25]:
```
y_pred = best_model.predict(x_test)
y_pred
```

[25]:
```
array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

[26]:
```
accuracy = accuracy_score(y_test, y_pred)
```

[27]:
```
print("Best Parameters :", best_params)
print(f"Accuracy = ", accuracy)
```

```
Best Parameters : {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy =  0.8032786885245902
```

[ ]:
```

```

[ ]:
```

```