dsbda5

March 13, 2024

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

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

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

[3]:
```
(1025, 14)
```

[4]:
```
df.head()
```

[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
```

```
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

[6]:
```
#1. Data Cleaning
```

[19]:
```
# Check for missing values
df.isnull().sum()
```

[19]:
```
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
```

[21]:
```
# Remove columns with more than 50% missing values
df.dropna(thresh=0.5*len(df), axis=1, inplace=True)
```

[22]:
```
# Check for duplicate data
df.duplicated().sum()
```

[22]:
```
0
```

[26]:
```
df.drop_duplicates(inplace=True)
```

[27]:
```
df.shape
```

[27]:
```
(302, 14)
```

[28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 302 entries, 0 to 878
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       302 non-null    int64
 1   sex       302 non-null    int64
 2   cp        302 non-null    int64
 3   trestbps  302 non-null    int64
 4   chol      302 non-null    int64
 5   fbs       302 non-null    int64
 6   restecg   302 non-null    int64
 7   thalach   302 non-null    int64
 8   exang     302 non-null    int64
 9   oldpeak   302 non-null    float64
 10  slope     302 non-null    int64
 11  ca        302 non-null    int64
 12  thal      302 non-null    int64
 13  target    302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 43.5 KB
```

[11]:
```
# 2.Data Integration (first split dataset and then merge )
```

[12]:
```
#subset1
subset1= df[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs']].loc[0:]
subset1
```

[12]:

	age	sex	cp	trestbps	chol	fbs
0	52	1	0	125	212	0
1	53	1	0	140	203	1
2	70	1	0	145	174	0
3	61	1	0	148	203	0
4	62	0	0	138	294	1
..	…	…	..	…	…	…
723	68	0	2	120	211	0
733	44	0	2	108	141	0
739	52	1	0	128	255	0
843	59	1	3	160	273	0
878	54	1	0	120	188	0

[302 rows x 6 columns]

[13]:
```
#Subset 2
subset2= df[['age', 'restecg', 'thalach', 'exang', 'oldpeak', 'ca']].loc[0:]
subset2
```

[13]:
```
      age  restecg  thalach  exang  oldpeak  ca
0      52        1      168      0      1.0   2
1      53        0      155      1      3.1   0
2      70        1      125      1      2.6   0
3      61        1      161      0      0.0   1
4      62        1      106      0      1.9   3
..    ...      ...      ...    ...      ...  ..
723    68        0      115      0      1.5   0
733    44        1      175      0      0.6   0
739    52        1      161      1      0.0   1
843    59        0      125      0      0.0   0
878    54        1      113      0      1.4   1

[302 rows x 6 columns]
```

[14]:
```
merged_df = pd.merge(subset1, subset2, on='age')
```

[15]:
```
merged_df
```

[15]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  ca
0      52    1   0       125   212    0        1      168      0      1.0   2
1      52    1   0       125   212    0        1      156      1      1.0   0
2      52    1   0       125   212    0        1      158      0      0.8   1
3      52    1   0       125   212    0        0      190      0      0.0   0
4      52    1   0       125   212    0        1      169      0      0.0   4
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...  ..
3057   35    1   0       126   282    0        1      130      1      1.6   0
3058   35    1   0       126   282    0        1      182      0      1.4   0
3059   35    1   0       126   282    0        1      174      0      0.0   0
3060   35    1   0       126   282    0        0      156      1      0.0   0
3061   74    0   1       120   269    0        0      121      1      0.2   1

[3062 rows x 11 columns]
```

[16]:
```
#Data Transformation(converting target column to binary values 0 & 1)
```

[17]:
```
df['target'] = df['target'].apply(lambda x: 1 if x>0 else 0)
```

[18]:
```
df.tail()
```

[18]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
723    68    0   2       120   211    0        0      115      0      1.5
733    44    0   2       108   141    0        1      175      0      0.6
739    52    1   0       128   255    0        1      161      1      0.0
843    59    1   3       160   273    0        0      125      0      0.0
878    54    1   0       120   188    0        1      113      0      1.4
```

	slope	ca	thal	target
723	1	0	2	1
733	1	0	2	1
739	2	1	3	0
843	2	0	2	0
878	1	1	3	0

[ ]:
```
#Error correcting
```

[34]:
```
df[df['age']<0] = df['age'].mean()
```

[ ]:
```
#Data Modelling
```

[35]:
```
from sklearn.model_selection import train_test_split

X = df.drop(['target'], axis=1)
y = df['target']
```

[36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[39]:
```
X_train.shape
```

[39]:
```
(241, 13)
```

[41]:
```
X_test.shape
```

[41]:
```
(61, 13)
```

[44]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\Users\namya\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n
2kfra8p0\LocalCache\local-packages\Python311\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(
```

[44]:
```
LogisticRegression()
```

[45]:
```
y_pred = logreg.predict(X_test)
```

[46]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

[47]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[25  5]
 [ 1 30]]
              precision    recall  f1-score   support

           0       0.96      0.83      0.89        30
           1       0.86      0.97      0.91        31

    accuracy                           0.90        61
   macro avg       0.91      0.90      0.90        61
weighted avg       0.91      0.90      0.90        61
```

[ ]:
```

```