Healthcare (ML) project

January 20, 2023

1 Import Libraries

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[2]:
```
heart_data= pd.read_csv('cep1_dataset.csv')
```

[3]:
```
heart_data.head()
```

[3]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

[4]:
```
heart_data.shape
```

[4]:
```
(303, 14)
```

[5]:
```
heart_data.info
```

[5]:
```
<bound method DataFrame.info of      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]>
```

[6]:
```
heart_data.isnull().sum()
```

[6]:
```
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
```

[7]:
```
heart_data.describe()
```

```
[7]:               age         sex          cp    trestbps        chol         fbs  \
       count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
       mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
       std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
       min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
       25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
       50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
       75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
       max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

                 restecg     thalach       exang     oldpeak       slope          ca  \
       count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
       mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
       std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
       min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
       25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
       50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
       75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
       max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

                    thal      target
       count  303.000000  303.000000
       mean     2.313531    0.544554
       std      0.612277    0.498835
       min      0.000000    0.000000
       25%      2.000000    0.000000
       50%      2.000000    1.000000
       75%      3.000000    1.000000
       max      3.000000    1.000000
```

2 Checking distribution of the target variable

```
[8]: heart_data['target'].value_counts()
```

```
[8]: 1    165
     0    138
     Name: target, dtype: int64
```

```
[9]: X= heart_data.drop(columns='target', axis=1)
     Y= heart_data['target']
```

```
[10]: print(X)
```

```
          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
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     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]
```

[11]:

```python
print(Y)
```

```
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
```

3 Splitting the data into train and test data

[12]:
```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.
 ↪2,stratify=Y, random_state=2)
```

[13]:
```
print(X.shape)
```

```
(303, 13)
```

[14]:
```
print(X_train.shape)
```

```
(242, 13)
```

[15]:
```
print(X_test.shape)
```

```
(61, 13)
```

4 Model Training Logistic Regression

[16]:
```
model= LogisticRegression()
```

[17]:
```
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818:
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

[17]:
```
LogisticRegression()
```

5 Model Evaluation

5.1 Accuracy Score

[18]:
```
# Accuracy on train data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction, Y_train)
```

[19]:

```
print('Accuracy on Training data:', training_data_accuracy)
```

```
Accuracy on Training data: 0.8512396694214877
```

[20]:

```
# Accuracy on test data
X_test_prediction=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction, Y_test)
```

[21]:

```
print('Accuracy on Test data:', test_data_accuracy)
```

```
Accuracy on Test data: 0.819672131147541
```

5.2 Builiding a Predication System

[22]:

```
input_data=(63, 1, 3, 145, 233, 1, 0, 150, 0, 2.3, 0, 0, 1)
```

[23]:

```
# change the input data to a numpy array
input_data_as_numpy_array=np.asarray(input_data)
print(input_data_as_numpy_array)
```

```
[ 63.    1.    3.  145.  233.    1.    0.  150.    0.    2.3   0.    0.
   1. ]
```

[24]:

```
# reshaping the numpy array as we are predicating for only on instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
```

[25]:

```
print(input_data_reshaped)
```

```
[[ 63.    1.    3.  145.  233.    1.    0.  150.    0.    2.3   0.    0.
    1. ]]
```

[26]:

```
predication=model.predict(input_data_reshaped)
print(predication)
```

```
[1]
```

```
/usr/local/lib/python3.7/site-packages/sklearn/base.py:451: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  "X does not have valid feature names, but"
```

[27]:

```
if (predication[0]== 0):
    print('The Person does not have a Heart Disease')
else:
    print('The Person has Heart Disease')
```

```
The Person has Heart Disease
```

[28]:

```
input_data= (60,1,0,117,230,1,1,160,1,1.4,2,2,3)
input_data_as_numpy_array=np.asarray(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
predication=model.predict(input_data_reshaped)
print(predication)
if (predication[0]== 0):
    print('The Person does not have a Heart Disease')
else:
    print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

```
/usr/local/lib/python3.7/site-packages/sklearn/base.py:451: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  "X does not have valid feature names, but"
```

[ ]:

```

```