Health care – Heart Attack Possibility

Batch-2(188W1A0544, 188W1A0536, 198W5A0506, 198W5A0503)

About data set

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date.The " target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no/less chance of heart attack and 1 = more chance of heart attack.

Attribute Information:

1) age

2) sex

3) chest pain type (4 values)

4) resting blood pressure

5) serum cholestoral in mg/dl

6)fasting blood sugar > 120 mg/dl

7) resting electrocardiographic results (values 0,1,2)

8) maximum heart rate achieved

9) exercise induced angina

10) oldpeak = ST depression induced by exercise relative to rest

11)the slope of the peak exercise ST segment

12) number of major vessels (0-3) colored by flourosopy

13) thal: 0 = normal; 1 = fixed defect; 2 = reversable defect

14) target: 0= less chance of heart attack 1= more chance of heart attack

Dataset2-https://drive.google.com/drive/folders/1Vj9ZbqRcUhDPc1GGYYscLiMTizn8bW6U?usp=sharing

Tasks:

1. Define the problem in structured terms.
2. Read the data and view the data present.
3. Check for Null values in data and process the data.
4. Separate the independent and dependent values.
5. Divide test and train datasets.
6. Build the Model.
7. Check for the accuracy.

Description:

- A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.
- **pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
- **Seaborn** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines

Importing the libraries

```
# importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing the dataset

```
hdata=pd.read_csv('heart.csv')
hdata.head()
```

```
In [4]: hdata.head()
```

Out[4]:

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

```
In [5]: hdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Taking care of missing data

```
In [6]: hdata.isnull().sum()
```

```
Out[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
```

Encoding the Independent Variable

```
In [10]: # Putting feature variable to X
         X = hdata.drop('target',axis=1)

         # Putting response variable to y
         y = hdata['target']
         X.head()
```

Out[10]:

	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

Encoding the Dependent Variable

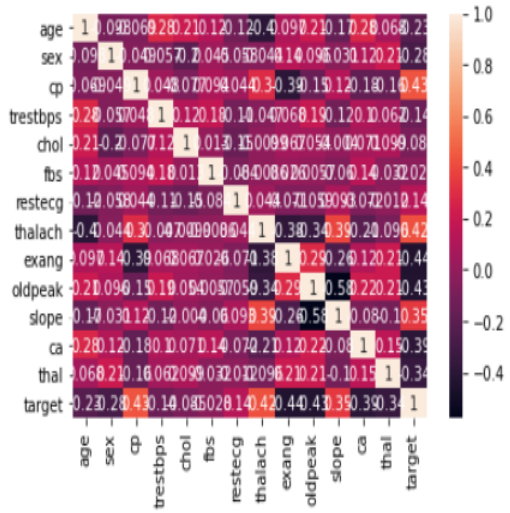
```
In [11]: y.head()
```

```
Out[11]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: target, dtype: int64
```

Data Visualization

In [8]:
```
sns.heatmap(hdata.corr(),annot=True)
```

Out[8]: <AxesSubplot:>



Splitting the dataset into the Training set and Test set

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)
```

In [14]:
```
X_train.shape
```

Out[14]: (212, 13)

In [15]:
```
y_train.shape
```

Out[15]: (212,)

In [16]:
```
X_test.shape
```

Out[16]: (91, 13)

In [17]:
```
y_test.shape
```

Out[17]: (91,)

Feature Scaling

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [19]:
```
d_tree = DecisionTreeClassifier(max_depth=3)
```

In [20]:
```
d_tree.fit(X_train,y_train)
```

Out[20]: DecisionTreeClassifier(max_depth=3)

In [21]:
```
from sklearn.metrics import roc_auc_score,accuracy_score #to check accuracy
```

In [22]:
```
y_predict=d_tree.predict(X_test)
y_predict
```

Out[22]:
```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 1, 0], dtype=int64)
```

▼Testing Accuracy

```
In [23]: accuracy_score(y_test,y_predict)
Out[23]: 0.8351648351648352

In [24]: roc_auc_score(y_test,y_predict)
Out[24]: 0.8355072463768116

In [25]: y_predict_train=d_tree.predict(X_train)

In [26]: y_predict_train
Out[26]: array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1,
                1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
                1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
                1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
                0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
                1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
                0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
                0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
                1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
                1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1], dtype=int64)

In [27]: accuracy_score(y_train,y_predict_train)
Out[27]: 0.8537735849056604

In [28]: roc_auc_score(y_train,y_predict_train)
Out[28]: 0.8454710144927535
```

since the difference between the accuracy scores of test and train datasets is less(almost negligible) the model duild by us is a good fit.

Result: The program is implemented in python and the output is observed