HEART DISEASE PREDICTION USING ML ALGORITHMS

SPRINT-3(II)

Date	16 November 2022
Team ID	PNT2022TMID21605
Project Name	Visualizing and Predicting Heart Diseases with an Interactive Dashboard

1.Import the dataset into Google Golab

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  Heart_Dise...rediction.csv
• Heart_Disease_Prediction.csv(text/csv) - 11930 bytes, last modified: 11/17/2022 - 100% done
Saving Heart_Disease_Prediction.csv to Heart_Disease_Prediction (1).csv
```

2.Import all the package which are required

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import numpy as np
```

3.Read the dataset

```
df = pd.read_csv('Heart_Disease_Prediction.csv')
df.dtypes
```

```
Age                       int64
Sex                       int64
Chest pain type           int64
BP                        int64
Cholesterol               int64
FBS over 120              int64
EKG results               int64
Max HR                    int64
Exercise angina           int64
ST depression           float64
Slope of ST               int64
Number of vessels fluro   int64
Thallium                  int64
Heart Disease            object
dtype: object
```

4.Display first five data details from our dataset

```
[61]
df.head()
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

5.Check duplicates in the dataset

```
df.isnull().sum()
```

```
Age                       0
Sex                       0
Chest pain type           0
BP                        0
Cholesterol               0
FBS over 120              0
EKG results               0
Max HR                    0
Exercise angina           0
ST depression             0
Slope of ST               0
Number of vessels fluro   0
Thallium                  0
Heart Disease             0
dtype: int64
```

6.Describe the some specific data from dataset

```
name = df.columns
num_var = ['Age', 'BP', 'Cholesterol', 'Max HR', 'Heart Disease']
cat_var = [item for item in name if item not in num_var]

num_var_data = df[df.columns & num_var]
num_var_data.describe()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning:

Index.__and__ operating as a set operation is deprecated, in the future this will be a logical operation matching Series.__and__.  Use index.intersection(other) instead
```

	Age	BP	Cholesterol	Max HR
count	270.000000	270.000000	270.000000	270.000000
mean	54.433333	131.344444	249.659259	149.677778
std	9.109067	17.861608	51.686237	23.165717
min	29.000000	94.000000	126.000000	71.000000
25%	48.000000	120.000000	213.000000	133.000000
50%	55.000000	130.000000	245.000000	153.500000
75%	61.000000	140.000000	280.000000	166.000000
max	77.000000	200.000000	564.000000	202.000000

7.Calculates the relationship between each column in your data set

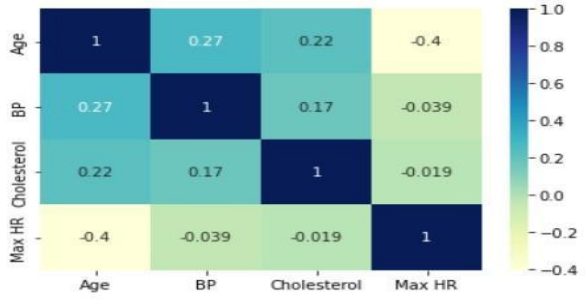
```
[65] num_var_data.corr()
```

	Age	BP	Cholesterol	Max HR
Age	1.000000	0.273053	0.220056	-0.402215
BP	0.273053	1.000000	0.173019	-0.039136
Cholesterol	0.220056	0.173019	1.000000	-0.018739
Max HR	-0.402215	-0.039136	-0.018739	1.000000

8.Display the Heatmap with colour

```
sns.heatmap(num_var_data.corr(), cmap="YlGnBu", annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72d9a91f50>
```



9.Display the different types of models

(i) Using Logistic Regression algorithm

Logistic regression is an example of supervised learning. It is used to calculate or predict the

probability of a binary (yes/no) event occurring.

```
[69] y=df['Heart Disease']
     x=df.drop('Heart Disease',axis=1)
```

```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20)
model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-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

LogisticRegression()
```

```
y=model.predict(X_test)
ya=np.array(Y_test)
ya
```

```
array(['Presence', 'Presence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Absence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Absence', 'Presence', 'Absence', 'Presence', 'Presence',
       'Absence', 'Presence', 'Absence', 'Presence', 'Presence',
       'Presence', 'Presence', 'Absence', 'Absence', 'Presence',
       'Absence', 'Presence', 'Presence', 'Presence', 'Absence',
       'Absence', 'Presence', 'Presence', 'Presence', 'Presence',
       'Presence', 'Presence', 'Presence', 'Presence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Presence', 'Absence', 'Presence', 'Absence', 'Absence', 'Absence',
       'Absence'], dtype=object)
```

Accuracy score of Logistic Regression:

```
[ ]  scLR=model.score(X_test,Y_test)
     scLR
```

```
0.8333333333333334
```

(ii) Using Support vector machine algorithm

SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

Accuracy score of SVM:

```
[ ]  model1=SVC()
     model1.fit(X_train,Y_train)
     sc=model1.score(X_test,Y_test)
     sc
```

```
0.6481481481481481
```

(iii)Using Random Forest algorithm

Random forest is a supervised machine learning algorithm that is used widely in classification and regression problem. It builds decision tree on different samples and takes their majority vote for classification and average in case of regression

Accuracy score of Random forest

```
model2=RandomForestClassifier()
model2.fit(X_train,Y_train)
scrfc=model2.score(X_test,Y_test)
scrfc
```

```
0.8703703703703703
```

(iv) Using K Nearest Neighbors alogithm:

This algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-NN algorithm

Accuracy score of KNN algorithm

```
[ ] model3=KNeighborsClassifier()
    model3.fit(X_train,Y_train)
    sckn=model3.score(X_test,Y_test)
    sckn
```

```
0.6111111111111112
```

(v) Using Gaussian Naïve Bayes algorithm

Gaussian naïve bayes algorithm based in applying bayes's theorem with strong independence assumptions.

Accuracy score of GaussianNB

```
model4=GaussianNB()
model4.fit(X_train,Y_train)
scnb=model4.score(X_test,Y_test)
scnb
```

```
0.8888888888888888
```

(vi) Using Linear Discriminant Analysis

Linear discriminant analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling difference in groups. i.e separating two or more classes.

Accuracy score of Linear Discriminant analysis

```
[ ]  model5=LinearDiscriminantAnalysis()
     model5.fit(X_train,Y_train)
     sclda=model5.score(X_test,Y_test)
     sclda

     0.8518518518518519
```

(vii) Using Ada Boosting algorithm

Ada Boosting is an ensemble learning method which was initially created to increase the efficiency of binary classifiers.

Accuracy score of Ada Boosting

```
[ ]  model6=AdaBoostClassifier()
     model6.fit(X_train,Y_train)
     scabc=model6.score(X_test,Y_test)
     scabc

     0.8333333333333334
```

10)The dataset is tested with all above models out of which random forest has given the good

accuracy 87% and this it is better than all the model for this

dataset. 1.import the Random forest package from ensemble

```
[ ]  from sklearn.ensemble import RandomForestClassifier

[ ]  rf=RandomForestClassifier()

[ ]  rf.fit(X_train,Y_train)

     RandomForestClassifier()

[ ]  y_pred5=rf.predict(X_test)
```

2.Create a data for testing

```
new_data=pd.DataFrame({
    'Age':70,
    'Sex':1,
    'Chest pain type':4,
    'BP':130,
    'Cholesterol':322,
    'FBS over 120':0,
    'EKG results':2,
    'Max HR':109,
    'Exercise angina':0,
    'ST depression':2.4,
    'Slope of ST':2,
    'Number of vessels fluro':3,
    'Thallium':3,
},index=[0])
```

```
[ ]  new_data
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3

3. The above data is specifically tested to check the different testcases

```
p =rf.predict(new_data)
if p[0]==0:
  print("No disease")
else:
  print("disease")
```

```
disease
```

4.No Disease Prediction

```
new_data1=pd.DataFrame({
    'Age':57,
    'Sex':1,
    'Chest pain type':4,
    'BP':140,
    'Cholesterol':192,
    'FBS over 120':0,
    'EKG results':0,
    'Max HR':148,
    'Exercise angina':0,
    'ST depression':0.4,
    'Slope of ST':2,
    'Number of vessels fluro':0,
    'Thallium':6,
},index=[0])
```

```
[44] new_data1
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
0	67	0	3	115	564	0	2	160	0	1.6	2	0	7

```
p1 =rf.predict(new_data1)
if p1[0]==0:
  print("disease")
else:
  print("No Disease")
```

```
No Disease
```