Experiment 2

Aim: Medical diagnosis for disease classification using decision tree.

Decision Tree Classifier: Decision tree is a supervised learning algorithm applied for classification and regression problem. General concept of creating decision tree has to train the model, which can be used predict target class using decision rules. Decision rules can be created using prior (training) data and tree structure. Decision tree is a tree structure, where each internal node denotes an attribute, branch represent outcome and decision rule and leaf node represent the class label. Root node is the topmost node in the tree. Attributes for root node and internal node has been decided according to splitting criterion. Attribute selection measure like information gain, gain ratio, and gini index has been applied for splitting of the node.

Decision Tree Algorithm:

Step 1: Select the best attribute to split the dataset, using one of the attribute selection measures.

Step 2: Consider the attribute as decision node obtained in step 1, then repeat step 1 for the smaller subset of the dataset (subtree)

Step 3: Repeat step 1 and step 2 until leaf node has been obtained with one of the following condition.

(a) All the instances belong to respective label.

(b) There are no more attribute remaining

(c) There are no more instances remaining.

Attribute Selection Measure:

Attribute selection measure enables the selection and splitting criterion for the given dataset. Attribute selection measure provides ranking for each attribute, which are described in training tuples. Attribute with best score has been criterion for splitting the tree.

Information Gain: Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. Entropy is the measure of the impurity of the input set.

$$\boldsymbol{Info(D)} = -\sum_{i=1}^{m} pi \log2. pi$$

$$\boldsymbol{Info\ A(D)} = \sum_{j=1}^{V} \frac{|Dj|}{|D|}\ Info\ (Dj)$$

$$\boldsymbol{Gain(A)} = Info(D) - InfoA(D)$$

Where, pi is the probability that an arbitrary tuple in D belongs to class Ci. Info(D) is the average amount of information needed to identify the class label of a tuple in D. |Dj|/|D| acts as the weight of the jth partition. InfoA(D) is the expected information required to classify a tuple from D based on the partitioning by A. The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N.

Gain Ratio: Information gain has been biased when test with many outcomes. To avoid the bias, select the attributes having a large number of values, which can be handled by gain ratio.

$$\boldsymbol{SplitInfoA(D)} = -\sum_{j=1}^{v} \frac{|Dj|}{|D|} \text{ x } \log 2\ \frac{|Dj|}{|D|}$$

Where, |Dj|/|D| acts as the weight of the jth partition. v is the number of discrete values in attribute A. The gain ratio can be defined as

$$\boldsymbol{GainRatio} = \frac{Gain\ (A)}{SplitInfoA(D)}$$

Attribute with the highest gain ratio among the all attributes from the dataset has been chosen as the splitting (node) attribute.

Dataset:

For this experiment we are using the Heart dataset in which following 13 features (1 to 13) and one target/class (S.No. 14) columns are given.

```
S.No.Column    Non-Null Count  Dtype
---  ------    --------------  -----
 1   age       303 non-null    int64
 2   sex       303 non-null    int64
 3   cp        303 non-null    int64
 4   trtbps    303 non-null    int64
 5   chol      303 non-null    int64
 6   fbs       303 non-null    int64
 7   restecg   303 non-null    int64
 8   thalachh  303 non-null    int64
 9   exng      303 non-null    int64
 10  oldpeak   303 non-null    float64
 11  slp       303 non-null    int64
 12  caa       303 non-null    int64
 13  thall     303 non-null    int64
 14  output    303 non-null    int64
```

Python Code

Importing required libraries

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

Fetching data

```
data=pd.read_csv('heart.csv')
```

Checking the information of the data

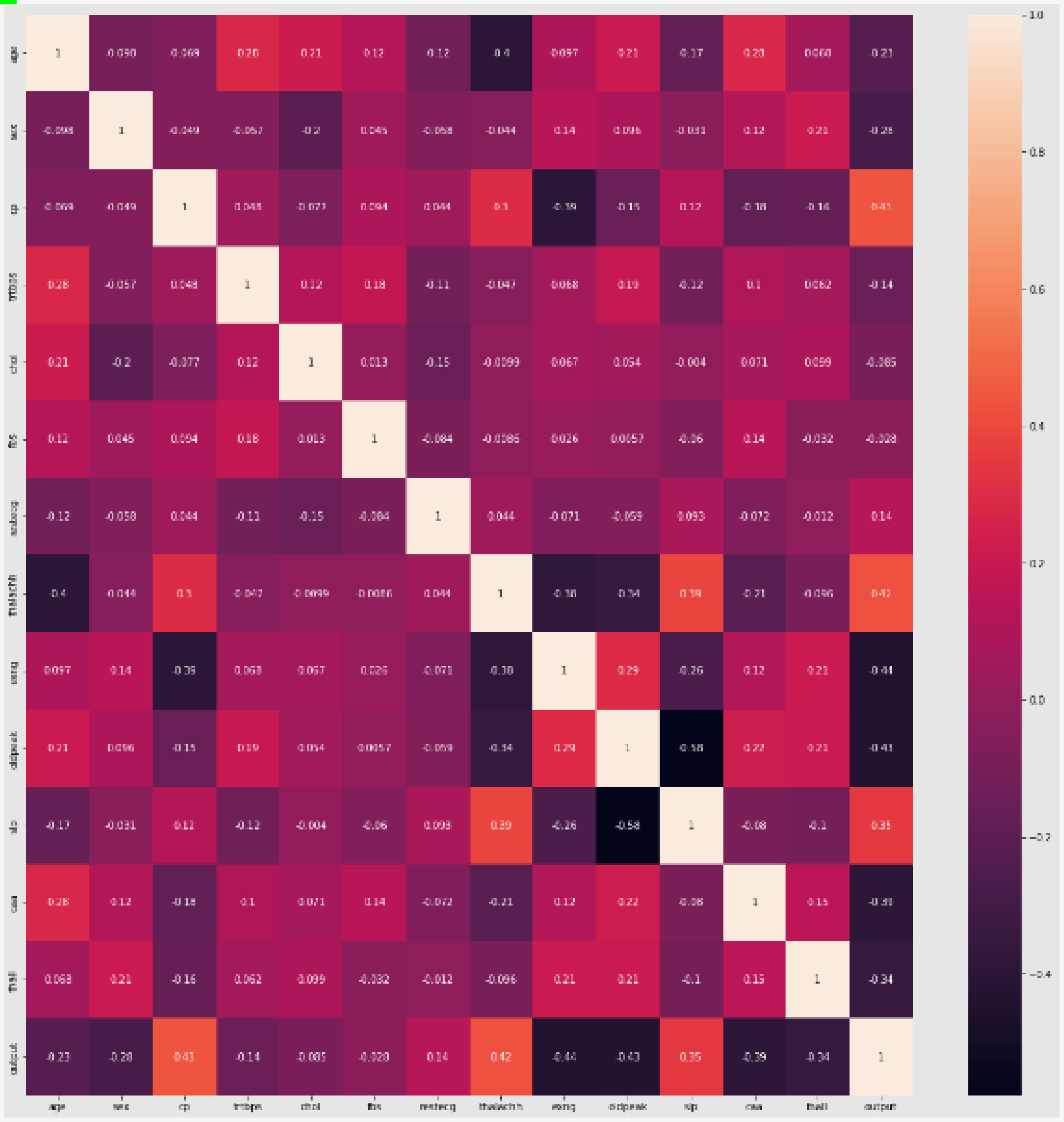
```
data.info()
```

Correlation of variables and Plot the Heat-map

```
data_corr=data.corr()
data_corr   #correalation of variables


plt.figure(figsize=(20,20))
sns.heatmap(data=data_corr,annot=True)
```

Output:



Preprocessing with Feature Correlation and Feature Selection

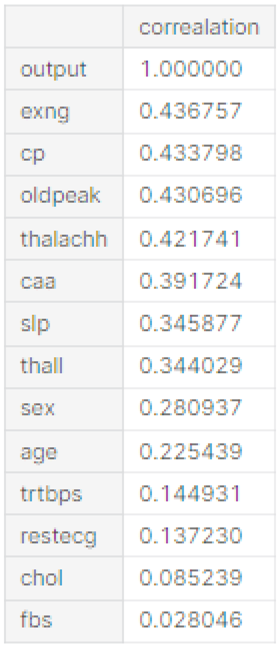
```
feature_value=np.array(data_corr['output'])
for i in range(len(feature_value)):
    if feature_value[i]<0:
        feature_value[i]=-feature_value[i]
feature_value
```

Output:

```
array([0.22543872, 0.28093658, 0.43379826, 0.14493113, 0.08523911,
       0.02804576, 0.1372295 , 0.42174093, 0.43675708, 0.430696  ,
       0.34587708, 0.39172399, 0.34402927, 1.        ])
```

```
features_corr=pd.DataFrame(feature_value,index=data_corr['output'].index,columns=
['correalation'])
feature_sorted=features_corr.sort_values(by=['correalation'],ascending=False)
feature_sorted
```

Output:



	correalation
output	1.000000
exng	0.436757
cp	0.433798
oldpeak	0.430696
thalachh	0.421741
caa	0.391724
slp	0.345877
thall	0.344029
sex	0.280937
age	0.225439
trtbps	0.144931
restecg	0.137230
chol	0.085239
fbs	0.028046

```
feature_selected=feature_sorted.index
feature_selected
```

Output:

```
Index(['output', 'exng', 'cp', 'oldpeak', 'thalachh', 'caa', 'slp', 'thall',
       'sex', 'age', 'trtbps', 'restecg', 'chol', 'fbs'],
      dtype='object')
```

```
clean_data=data[feature_selected]
```

Classification using Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

#making input and output dataset
X=clean_data.iloc[:,1:]
Y=clean_data['output']

x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=0)

# feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)

#training our model
dt=DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt.fit(x_train,y_train)

#predicting the value on testing data
y_pred=dt.predict(x_test)

#ploting the data
from sklearn.metrics import confusion_matrix
conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
accuracy=dt.score(x_test,y_test)
print("\nThe accuracy of decisiontreelassifier on Heart disease prediction datase
t is "+str(round(accuracy*100,2))+"%")
```

Output:

```
[[24  9]
 [ 3 40]]

The accuracy of decision tree classifier on Heart disease prediction dataset is84.21%
```