Data ware house data mining

Name: Jahidul Islam Noor

ID: 19-39306-1

Report Name: Predicting heart disease by machine learning.

Introduction:

In this report I am going to build a model where my model will predict a person heart disease by machine learning. For this I need some data to build my model. I have taken my data from Kaggle. Link of the data set is given below,

Dataset link: https://www.kaggle.com/ronitf/heart-disease-uci

Dataset:

In the dataset there are 303 rows and 14 columns. In the 14 columns 13 columns are feature columns and remaining one is predicted variable column.

Features description:

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: 3 = normal; 6 = fixed defect; 7 = reversable defect

Another remaining column is *target* where it is defined that if a person has disease or not.

0 → No heart disease

1 → Heart disease

Choosing algorithm:

In my model I am providing the data, and my model will learn from that data. Then the model will predict heart disease for new instance. So, my model will use supervised algorithm. Also, my model will only predict *Yes* or *No*. So, it is classification problem. For this particular problem **Decision tree classification** algorithm works well, but **Random Forest classification** can also perform well in this kind of problem, because random forest is built up from many decision trees. We will decide which algorithm is best suited for our model after getting the accuracy from the both Algorithm.

Platform:

I am going to use **python** for my model and **Spyder** IDE, to make work easy.

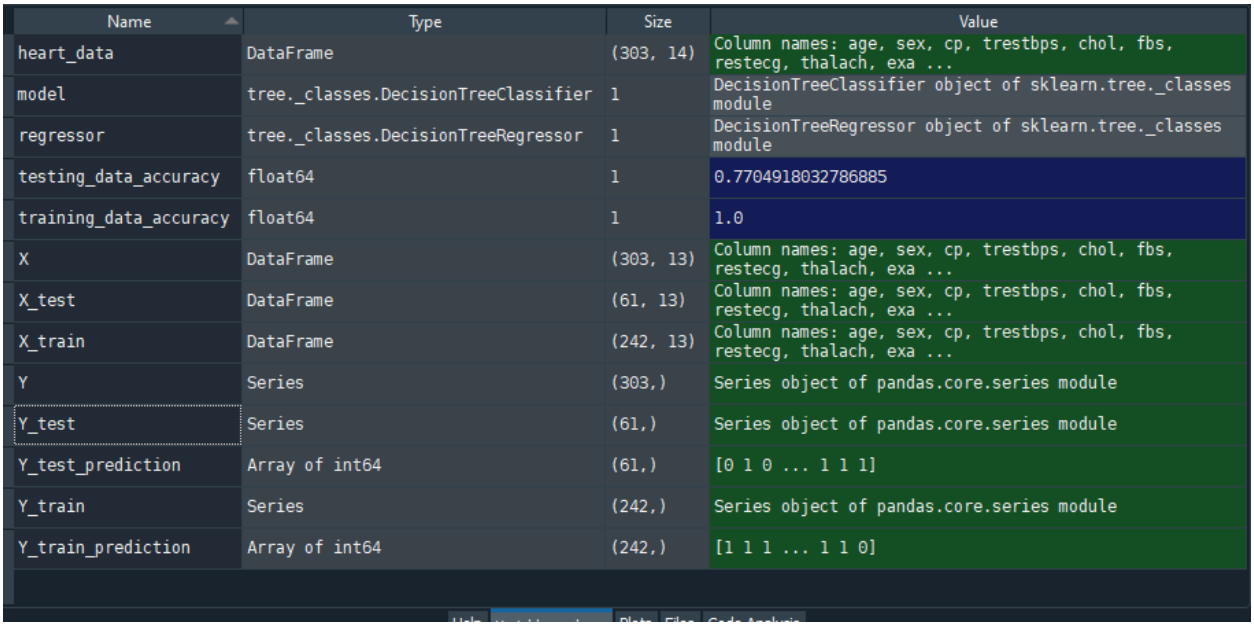
Code for Decision tree classification:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score


heart_data = pd.read_csv("heart.csv")



"""heart_data.head()"""


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


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=0)


model = DecisionTreeClassifier()


model.fit(X_train, Y_train)


Y_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train_prediction, Y_train)


print("Accuracy on training data: " , training_data_accuracy)


Y_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(Y_test_prediction, Y_test)


print("Accuracy on testing data: " , testing_data_accuracy)
```

Code explanation:

1. I have imported *panda* library to read my **heart.csv** file, which is located in my same working directory of my programing file.
2. I have imported *train_test_split* from *sklearn* to divide my data set into two groups one for training my model and another for testing my model. 80% of the data set is given to training and another 20% is given to testing.
3. Then I have imported my desired algorithm *DecisionTreeClassifier* from *sklearn.tree.*
4. I have taken all my feature column to **X** and target column to **Y**.
5. Then I have created my model and trained or fitted my model from training data.
6. After that I have checked my model on training data and calculated it accuracy.
7. Also, I have checked my model with testing data, which my model never seen and calculated accuracy.

Variable explorer:



Name	Type	Size	Value
heart_data	DataFrame	(303, 14)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
model	tree._classes.DecisionTreeClassifier	1	DecisionTreeClassifier object of sklearn.tree._classes module
regressor	tree._classes.DecisionTreeRegressor	1	DecisionTreeRegressor object of sklearn.tree._classes module
testing_data_accuracy	float64	1	0.7704918032786885
training_data_accuracy	float64	1	1.0
X	DataFrame	(303, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
X_test	DataFrame	(61, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
X_train	DataFrame	(242, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
Y	Series	(303,)	Series object of pandas.core.series module
Y_test	Series	(61,)	Series object of pandas.core.series module
Y_test_prediction	Array of int64	(61,)	[0 1 0 ... 1 1 1]
Y_train	Series	(242,)	Series object of pandas.core.series module
Y_train_prediction	Array of int64	(242,)	[1 1 1 ... 1 1 0]

Figure 1: State of the variables on Decision tree regressor

Model accuracy:

On training data: 1.0 [100% accuracy because model already know the datasets]

On testing data: 0.77 [77%, Good accuracy considering the risk of data overfit]

Code for Random Forest tree classification:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


heart_data = pd.read_csv("heart.csv")



"""heart_data.head()"""


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


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=0)


model = RandomForestClassifier()


model.fit(X_train, Y_train)


Y_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train_prediction, Y_train)


print("Accuracy on training data: " , training_data_accuracy)


Y_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(Y_test_prediction, Y_test)


print("Accuracy on testing data: " , testing_data_accuracy)
```

Code explanation:

1. All things are same as previous code, only I have imported *Random Forest Regressor* algorithm from *sklearn.ensemble* and build the new model on it.

Variable explorer:

Name	Type	Size	Value
heart_data	DataFrame	(303, 14)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
model	ensemble._forest.RandomForestClassifier	100	RandomForestClassifier object of sklearn.ensemble._forest module
regressor	tree._classes.DecisionTreeRegressor	1	DecisionTreeRegressor object of sklearn.tree._classes module
testing_data_accuracy	float64	1	0.819672131147541
training_data_accuracy	float64	1	1.0
X	DataFrame	(303, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
X_test	DataFrame	(61, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
X_train	DataFrame	(242, 13)	Column names: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exa ...
Y	Series	(303,)	Series object of pandas.core.series module
Y_test	Series	(61,)	Series object of pandas.core.series module
Y_test_prediction	Array of int64	(61,)	[0 1 1 ... 1 1 1]
Y_train	Series	(242,)	Series object of pandas.core.series module
Y_train_prediction	Array of int64	(242,)	[1 1 1 ... 1 1 0]

Figure 2: State of the variables on Random forest regressor

Model accuracy:

On training data: 1.0 [100% accuracy because model already know the datasets]

On testing data: 0.82 [82% better accuracy than decision tree regressor]

Final decision:

As we can see random forest regressor gives 5% more accuracy than decision tree regressor, So I will choose Random forest regressor for my model.

Future improvements:

To reduce the underfit problem and reduce the time complexity, I can pass max leaf nodes as parameter in the regressor and using some random leaf nodes I can determine which max leaf nodes is giving more accurate value, but I have to also remember about overfitting the model.

There are many classification algorithms which will perform better than random forest, but for now I know this much about algorithm. In future when I will learn more algorithm, I would be able to improve this model.

Project github link: https://github.com/jahidul39306/Heart-Disease-Prediction-using-ML