2021

Random Forest

S14490

D.S.P.A.U.M. PEIRIS

Introduction

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages.

➢ 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.

➢ Problem

Identify, if the patient is suffering from heart disease

➢ Approach

Matching data set for random forest and model the random forest model. Then test the accuracy of the data set and take necessary decisions.

Content

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

➢ Steps Taken

Link for the data set;
https://raw.githubusercontent.com/anjula510/Heart/main/heart.csv

Link for the Google colab;
https://colab.research.google.com/drive/1oecCO8V0HUPE1WQqf1rgfV9kyNOXUIBs?usp=sharing

❖ First, we want to import essential libraries.

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action="ignore")
```

❖ Import data set

```
df=pd.read_csv("https://raw.githubusercontent.com/anjula510/Heart/main/heart.csv")
```

```
df.head()
```

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

❖ Checking variable types

```
#checking the variable type
df.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   oldpeak   303 non-null    float64
 9   slope     303 non-null    int64  
 10  ca        303 non-null    int64  
 11  thal      303 non-null    int64  
 12  target    303 non-null    int64  
 13  exang     303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

❖ Identifying Missing values.

```
#Check missing values
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
oldpeak     0
slope       0
ca          0
thal        0
target      0
exang       0
dtype: int64
```

❖ Dropping Unnecessary Columns.

```
[320] #column names
      df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'oldpeak', 'slope', 'ca', 'thal', 'target', 'exang'],
      dtype='object')
```

```
[321] df['oldpeak'].unique()
```

```
array([2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0. , 0.5, 1.6, 1.2, 0.2, 1.8,
       1. , 2.6, 1.5, 3. , 2.4, 0.1, 1.9, 4.2, 1.1, 2. , 0.7, 0.3, 0.9,
       3.6, 3.1, 3.2, 2.5, 2.2, 2.8, 3.4, 6.2, 4. , 5.6, 2.9, 2.1, 3.8,
       4.4])
```

```
[322] df['thal'].unique()
```

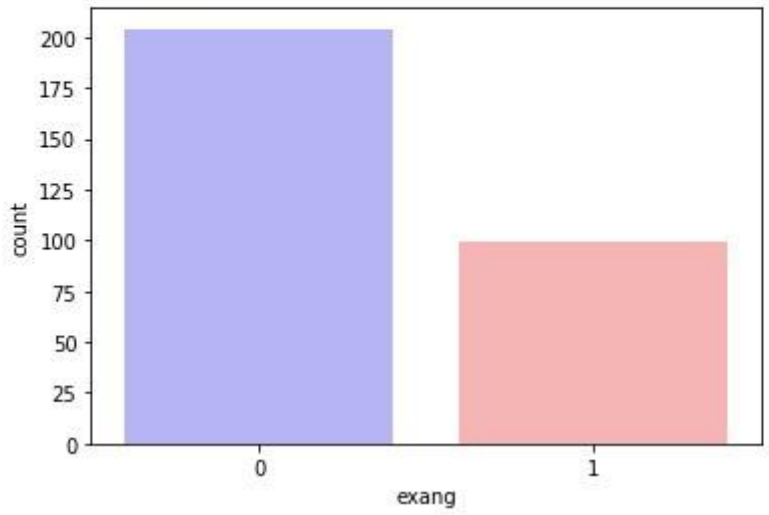
```
array([1, 2, 3, 0])
```

```
df=df.drop(['cp','target','ca','slope','restecg','fbs'], axis=1)
df.head()
```

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

❖ Data distribution

```
[166] df.exang.value_counts()
      sns.countplot( x="exang",  data=df ,  palette="bwr")
      plt.show()
```



0 is twis as 1 there for this will get bias. It affects the accuracy of the data. We need to under sample or over sample data.

❖ Dummy variables

```
[325] #dumy values
      sex = pd.get_dummies(df['sex'],drop_first=True,prefix='Sex')
      thal = pd.get_dummies(df['thal'],drop_first=True,prefix='Thal')
```

```
df= pd.concat([df,sex,thal],axis=1)
df = df.drop(columns =['sex','thal'])
df.head()
```

	age	trestbps	chol	thalach	oldpeak	exang	Sex_1	Thal_1	Thal_2	Thal_3
0	63	145	233	150	2.3	0	1	1	0	0
1	37	130	250	187	3.5	0	1	0	1	0
2	41	130	204	172	1.4	0	0	0	1	0
3	56	120	236	178	0.8	0	1	0	1	0
4	57	120	354	163	0.6	1	0	0	1	0

Adding variables to graph

❖ Training data set

```
[169] from sklearn.model_selection import train_test_split
      y =df.exang.values
      x_data =df.drop(['exang'], axis = 1)

      x_train,x_test,y_train,y_test = train_test_split(x_data,y,test_size = 0.2,random_state=0)
```

Splitting data to parts to train and test

Training 80% and Test 20%

```
[328] from sklearn.tree import DecisionTreeClassifier
      dtc = DecisionTreeClassifier()
      dtc.fit(x_train, y_train)

      DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                             max_depth=None, max_features=None, max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, presort='deprecated',
                             random_state=None, splitter='best')
```

```
[329] dtc.score(x_test, y_test)*100

      47.540983606557376
```

❖ Random Forest

```
[172] from sklearn.ensemble import RandomForestClassifier
      rf=RandomForestClassifier()
      rf.fit(x_train, y_train)

      RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                             criterion='gini', max_depth=None, max_features='auto',
                             max_leaf_nodes=None, max_samples=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=100,
                             n_jobs=None, oob_score=False, random_state=None,
                             verbose=0, warm_start=False)
```

➢ Inference

```
[331] #Random forest accuracy
      rf.score(x_test, y_test)*100

      75.40983606557377
```

Model accuracy we get 75.40%

➢ Conclusion

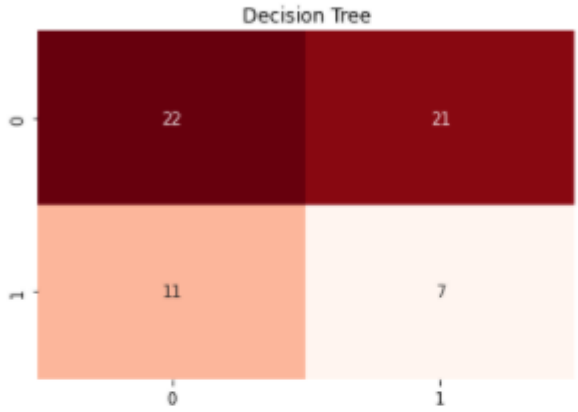
- Accuracy increased when the no. of decision trees increases

- We can conclude that confusion matrix is giving good model performance.

- **Confusion Matrix**

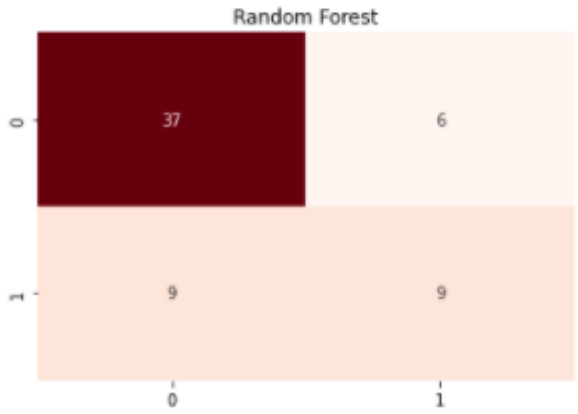
```
[333] cm_dtc = confusion_matrix(y_test,y_dtc)
      cm_rf = confusion_matrix(y_test,y_rf)
```

```
[334] plt.subplot()
      plt.title("Decision Tree")
      sns.heatmap(cm_dtc,annot=True,cmap="Reds",fmt="d",cbar=False)
      plt.show()
```




```
[335] plt.subplot()
      plt.title("Random Forest")
      sns.heatmap(cm_rf,annot=True,cmap="Reds",fmt="d",cbar=False)
      plt.show
```

```
<function matplotlib.pyplot.show>
```



➢ Reference

- https://www.kaggle.com/ronitf/heart-disease-uci
- https://raw.githubusercontent.com/anjula510/Heart/main/heart.csv

- [Sklearn Random Forest Classifiers in Python - DataCamp]
-