1. Постановка задачи

Провести серию экспериментов с построением и тестированием деревьев решений (используя DecisionTreeClassifier и RandomForestClassifier), переразбивая исходное множество данных, заданное в варианте, следующим образом:

Номер эксперимента	Размер обучающей выборки	Размер тестовой выборки
1	60%	40%
2	70%	30%
3	80%	20%
4	90%	10%

2. Исходные данные

Датасет: http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29
Предметная область: медицина
Задача: определить, присутствует ли сердечная болезнь или нет
Количество записей: 270
Количество атрибутов: 13
Атрибуты:
 -- 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
Классы:
 -- 14. Absence (1) or presence (2) of heart disease

3. Ход работы

Тестирование алгоритма DecisionTreeClassifier

```
import pandas as pd
import math
import numpy as np
import operator
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

def load_dataset():
    dataset = pd.read_csv('heart.dat', header=None).values
    return dataset
# разделение датасета на тестовую и обучающую выборку
def split_dataset(dataset,test_size):
    attr = dataset[:, 0:-1] # атрибуты
    heart_class =  (dataset[:, -1]).astype(np.int64, copy=False) # классы
    data_train, data_test, class_train, class_test =
train_test_split(attr, heart_class, test_size=test_size, random_state=55)
    return data_train, class_train, data_test, class_test

test_sizes=[0.4, 0.3, 0.2, 0.1]
dataset=load_dataset()
```

```
for test_size in test_sizes:
    data_train, class_train,data_test, class_test =
split_dataset(dataset,test_size)
    dtc = DecisionTreeClassifier()
    dtc.fit(data_train, class_train)
    print('Test-Size: ',test_size*100, '% Result: ', dtc.score(data_test,
class_test))
```

Тестирование алгоритма RandomForestClassifier

```
import pandas as pd
import math
import numpy as np
import operator
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

def load_dataset():
    dataset = pd.read_csv('heart.dat', header=None).values
    return dataset
# разделение датасета на тестовую и обучающую выборку
def split_dataset(dataset,test_size):
    attr = dataset[:, 0:-1] # атрибуты
    heart_class =  (dataset[:, -1]).astype(np.int64, copy=False) # классы
    data_train, data_test, class_train, class_test =
train_test_split(attr, heart_class, test_size=test_size, random_state=55)
    return data_train, class_train, data_test, class_test

test_sizes=[0.4, 0.3, 0.2, 0.1]
dataset=load_dataset()
for test_size in test_sizes:
    data_train, class_train,data_test, class_test =
split_dataset(dataset,test_size)
    rfc = RandomForestClassifier(n_estimators=100)
    rfc.fit(data_train, class_train)
    print('Test-Size: ',test_size*100, '% Result: ', rfc.score(data_test,
class_test))
```

Результаты:

Результаты тестирования алгоритма DecisionTreeClassifier

```
Test-Size:  40.0 % Result:  0.740740740741
Test-Size:  30.0 % Result:  0.79012345679
Test-Size:  20.0 % Result:  0.703703703704
Test-Size:  10.0 % Result:  0.62962962963
```

Результаты тестирования алгоритма RandomForestClassifier

```
Test-Size:  40.0 % Result:  0.833333333333
Test-Size:  30.0 % Result:  0.851851851852
Test-Size:  20.0 % Result:  0.851851851852
Test-Size:  10.0 % Result:  0.851851851852
```

Результаты алгоритмов с лаб 1:

```
Naive Bayes library algo Result: 0.83950617284
Naive Bayes algo Result: 0.83950617284
KNeighborsClassifier library algo Result: 0.679012345679
KNeighborsClassifier algo Result: 0.666666666667
```

Из результатов видно, что результаты алгоритма **RandomForestClassifier** показали результаты схожи с результатом алгоритма **Naive Bayes.** А результаты алгоритма **DecisionTreeClassifier** близки к результату алгоритма **KNeighborsClassifier.**