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

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

2. Исходные данные

- Датасет: https://archive.ics.uci.edu/ml/datasets/Wine
- Предметная область: Состав вина разного географического происхождения
- Задача: определить, в какой из 3 областей произведено вино
- Количество записей: 178
- Количество атрибутов: 13
- Атрибуты:
    1. Алкоголь
    2. Малиновая кислота
    3. Зола
    4. Алкалиния золы
    5. Магний
    6. Всего фенолов
    7. Флаваноиды
    8. Нефлаваноидные фенолы
    9. Проантоцианы
    10. Интенсивность цвета
    11. Оттенок
    12. OD280 / OD315 разведенных вин
    13. пролин

3. Ход работы

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# loading file with data
def load_data(filename):
      return pd.read_csv(filename, header=None).values

# splitting data on training and test sets
def split_dataset(test_size):
      dataset = load_data('data.csv')
      occ_class = dataset[:, 0]
      occ_attr = dataset[:, 1:]
      occ_class = occ_class.astype(np.float)
      occ_attr = occ_attr.astype(np.float)
      data_train, data_test, class_train, class_test = train_test_split(occ_attr,
occ_class, test_size=test_size, random_state=55)
      return data_train, class_train, data_test, class_test

def main():
    for size in np.arange(0.1, 0.4, 0.1):
        data_train, class_train, data_test, class_test = split_dataset(size)
        print('Size: ', round(size,1))
        decisionForest = DecisionTreeClassifier()
        decisionForest = decisionForest.fit(data_train, class_train)
        decisionAcc = decisionForest.score(data_test, class_test)
        print('Decision Tree accuracy: ', round(decisionAcc,10))
        randonForest = RandomForestClassifier()
        randonForest = randonForest.fit(data_train, class_train)
        randomAcc = randonForest.score(data_test, class_test)
        print('Random Tree accuracy: ', round(randomAcc,10))
        print('_____________________________')
main()


Lab1 Results:

Naive Bayes
('myNBClass ', 'Accuracy: ', 0.9491525423728814)
('sklNBClass ', 'Accuracy: ', 0.94915254237288138)

K Nearest Neighbours
('myKNClass', 'Accuracy: ', 0.66666666666666663)
('sklKNClass', 'Accuracy: ', 0.73015873015873012)


Results:

('Size: ', 0.1)
('Decision Tree accuracy: ', 0.833333333
('Random Tree accuracy: ', 0.9444444444)
_____________________________
```

```
('Size: ', 0.2)
('Decision Tree accuracy: ', 0.861111111
('Random Tree accuracy: ', 0.8888888889)
_________________________________
('Size: ', 0.3)
('Decision Tree accuracy: ', 0.907407407
('Random Tree accuracy: ', 0.9444444444)
_________________________________
('Size: ', 0.4)
('Decision Tree accuracy: ', 0.902777777
('Random Tree accuracy: ', 0.9583333333)
```

В ходе проделанной работы были получены приведенные выше результаты. Результат RandomForestClassifier схож с результатом, получаемым алгоритмом Naïve Bayes, а результат DecisionTreeClassifier ближе к значениям, полученным K Nearest Neighbours алгоритмом.