Задание 1.
Исходные данные:
Сформировать с помощью sklearn.make_classification датасет из 100 объектов с двумя признаками, обучить случайный лес из 1, 3, 10 и 50 деревьев и визуализировать их разделяющие гиперплоскости на графиках (по подобию визуализации деревьев из предыдущего урока, необходимо только заменить вызов функции predict на tree_vote).
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
>>> from sklearn import datasets
>>> from sklearn.datasets import make_classification
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.tree import DecisionTreeClassifier
>>> from sklearn.ensemble import RandomForestClassifier
>>> classification_data, classification_labels = make_classification(n_samples=100, n_features=1,
n_informative=1, n_classes=2, n_redundant=0, n_clusters_per_class=1)
>>> X_train, X_test, y_train, y_test = train_test_split(classification_data, classification_labels,
test_size=0.5)
>>> clf = DecisionTreeClassifier(random_state=0)
>>> rfc = RandomForestClassifier(random_state=0)
>>> clf = clf.fit(X_train, y_train)
>>> rfc = rfc.fit(X_train, y_train)
>>> score_c = clf.score(X_test, y_test)
>>> score_r = rfc.score(X_test, y_test)
>>> print("Single Tree:{}".format(score_c)
...       ,"Random Forest:{}".format(score_r)
...      )
Single Tree:0.72 Random Forest:0.72
>>> classification_data, classification_labels = make_classification(n_samples=100, n_features=3,
n_informative=1, n_classes=2, n_redundant=0, n_clusters_per_class=1)
>>> X_train, X_test, y_train, y_test = train_test_split(classification_data, classification_labels,
test_size=0.5)
>>> clf_1 = DecisionTreeClassifier(random_state=0)
>>> rfc_1 = RandomForestClassifier(random_state=0)
>>> clf_1 = clf_1.fit(X_train, y_train)
>>> rfc_1 = rfc_1.fit(X_train, y_train)
>>> score_c_1 = clf_1.score(X_test, y_test)
>>> score_r_1 = rfc_1.score(X_test, y_test)
>>> print("Single Tree:{}".format(score_c_1)
...       ,"Random Forest:{}".format(score_r_1)
...      )
Single Tree:0.94 Random Forest:0.96
>>> classification_data, classification_labels = make_classification(n_samples=100, n_features=10,
n_informative=1, n_classes=2, n_redundant=0, n_clusters_per_class=1)
```

```
>>> X_train, X_test, y_train, y_test = train_test_split(classification_data, classification_labels,
test_size=0.5)
>>> clf_2 = DecisionTreeClassifier(random_state=0)
>>> rfc_2 = RandomForestClassifier(random_state=0)
>>> clf_2 = clf_2.fit(X_train, y_train)
>>> rfc_2 = rfc_2.fit(X_train, y_train)
>>> score_c_2 = clf_2.score(X_test, y_test)
>>> score_r_2 = rfc_2.score(X_test, y_test)
>>> print("Single Tree:{}".format(score_c_2)
...       ,"Random Forest:{}".format(score_r_2)
...      )
Single Tree:1.0 Random Forest:1.0
>>> classification_data, classification_labels = make_classification(n_samples=100, n_features=50,
n_informative=1, n_classes=2, n_redundant=0, n_clusters_per_class=1)
>>> X_train, X_test, y_train, y_test = train_test_split(classification_data, classification_labels,
test_size=0.5)
>>> clf_3 = DecisionTreeClassifier(random_state=0)
>>> rfc_3 = RandomForestClassifier(random_state=0)
>>> clf_3 = clf_3.fit(X_train, y_train)
>>> rfc_3 = rfc_3.fit(X_train, y_train)
>>> score_c_3 = clf_3.score(X_test, y_test)
>>> score_r_3 = rfc_3.score(X_test, y_test)
>>> print("Single Tree:{}".format(score_c_3)
...       ,"Random Forest:{}".format(score_r_3)
...      )
Single Tree:1.0 Random Forest:1.0
>>> plt.scatter(score_c, score_r)
<matplotlib.collections.PathCollection object at 0x7f6596066940>
>>> plt.scatter(score_c_1, score_r_1)
<matplotlib.collections.PathCollection object at 0x7f6596066c40>
>>> plt.scatter(score_c_2, score_r_2)
<matplotlib.collections.PathCollection object at 0x7f6596066df0>
>>> plt.scatter(score_c_3, score_r_3)
<matplotlib.collections.PathCollection object at 0x7f659607c520>
>>> plt.show()
>>>
```

Задание 2.
Исходные данные:
Сделать выводы о получаемой сложности гиперплоскости и недообучении или переобучении случайного леса в зависимости от количества деревьев в нем
Решение:
Анализируя полученные результаты можно отметить несколько моментов: во-первых когда мы строим единичное дерево, то сказать об том, что оно переобученно или недообучено, так как у нас слишком мало исходных данных и нам сложно сказать, где будет ошибка и дерево переобучется или недообучется, так как среди этих данных могут быть, как ошибочные, так и нулевые данные или данные другой размерности, которые мы не масштабировали, так как эти данные мы используем обезличенно.
Далее рассматривая вариант с тремя деревьями, вероятность недообученности снижается в три раза, так как мы можем получить уже данные от трёх деревьев и получить чуть чище результат чем от одного дерева, точно также обстоит дело и с переобученностью.

Далее мы берём уже десять деревьев и смотрим как модель ведёт себя уже с ними и здесь вероятность снижается ещё больше из-за количества деревьев, тоже самое обстоит и с переобученностью.
Далее мы берём уже 50 деревьев и соотвественно вероятность недообученности снижается ещё больше, так как и переобученность.