Лабораторная работа № 2
Ансамблевые модели

1. Классификатор на основе дерева принятия решений (**Decision Tree Classifier**)
    1. Загрузить данные из файла data_decision_trees.txt, где первые 2 столбца — входные данные, а последний столбец соответствуют целевым меткам. Визуализировать распределение классов набора данных.
    2. Разделить выборку на тренировочную и тестовую. Обучить модель Random Forest Classifier. Подобрать оптимальные значения параметров *max_depth, min_samples_split, min_samples_leaf*.
    3. Получить метрики качества модели (accuracy precision, recall, f1-score). Использовать для этого функцию sklearn.metrics.classification_report(). Визуализировать результат классификации, используя функцию visualize_classifier() из utilities.py
2. Классификатор на основе случайных лесов (**Random Forest Classifier**) и предельно случайных лесов (**Extra Trees Classifier**)
    1. Загрузить данные из файла data_random_forests.txt, где первые 2 столбца — входные данные, а последний столбец соответствуют целевым меткам. Визуализировать распределение классов набора данных.
    2. Разделить выборку на тренировочную и тестовую. Обучить модели Random Forest Classifier и Extra Trees Classifier. Подобрать оптимальные значения параметров *n_estimators, max_depth*. Визуализировать результат классификации, используя функцию visualize_classifier() из utilities.py
3. Решение проблемы дисбаланса классов
    1. Используя данные из файла data_imbalance.txt обучить классификатор Extra Tree Classifier на тренировочных данных и получить метрики на тестовых.
    2. Визуализировать результат классификации, используя функцию visualize_classifier() из utilities.py. Почему классификатор так плохо справился со своей задачей?
    3. Повторить пункты 3.1-3.2, но в параметры классификатора указать параметр class_weight='balanced'.
4. Поиск оптимальных обучающих параметров с помощью сеточного поиска
    1. Исходные данные в файле data_random_forests.txt.
    2. Изучить работу **GridSearchCV**. Обучить модель **Random Forest Classifier**, используя сеточный поиск оптимальных параметров.
    3. Используя метод cv_results_ вывести метрику (`precision_weighted`) для каждого из наборов параметров.
    4. Получить отчет о тестировании модели с помощью функции classification_report().