Лабораторная работа №5.1

**Применение модели решающих деревьев в задачах классификации**

Для выполнения данной работы мы будем использовать набор данных (датасет) с информацией о медицинских параметрах больных диабетом. Датасет прилагается к заданию работы и может быть загружен с Kaggle <https://www.kaggle.com/datasets/saurabh00007/diabetescsv>

Задачи к лабораторной работе:

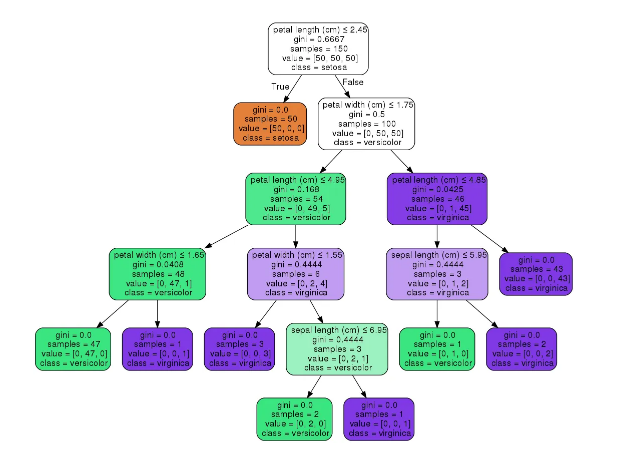
1. Решить задачу классификации больных методом логистической регрессии и решающих деревьев используя стандартные настройки моделей. Вывести стандартные метрики. Сделать вывод на основе анализа метрик о том, какая из рассмотренных моделей более подходит для исследуемого датасета.
2. Исследовать значение выбранной самостоятельно метрики (выбор обосновать) в зависимости от глубины решающего дерева. Построить график зависимости.
3. Для модели с оптимальной глубиной полученной в задании 2, отрисовать получившееся дерево (см. пример далее), важность признаков (feature importances) в виде столбчатой диаграммы, а также PR и ROC кривые
4. Опциональное задание: аналогично п.2 исследовать зависимость метрики от других параметров модели решающего дерева, например **max\_features.**

# **Пример в Scikit-learn**

from sklearn.datasets import load\_iris  
from sklearn import tree  
  
# Load in our dataset  
iris\_data = load\_iris()  
  
# Initialize our decision tree object  
classification\_tree = tree.DecisionTreeClassifier()  
  
# Train our decision tree (tree induction + pruning)  
classification\_tree = classification\_tree.fit(iris\_data.data, iris\_data.target)

Scikit-learn также позволяет визуализировать дерево с помощью библиотеки graphviz, в которой есть несколько очень полезных опций для визуализации узлов решений и разбиений, выученных моделью. Ниже обозначим узлы разными цветами, отталкиваясь от признаков имен, и отобразим класс и признак каждого узла.

import graphviz   
dot\_data = tree.export\_graphviz(classification\_tree, out\_file=None,   
 feature\_names=iris.feature\_names,   
 class\_names=iris.target\_names,   
 filled=True, rounded=True,   
 special\_characters=True)   
graph = graphviz.Source(dot\_data)   
graph.render("iris")



Кроме того, в Scikit-learn можно указать несколько параметров для модели дерева решений. Ниже приведем некоторые из таких параметров, позволяющих получить лучший результат:

**max\_depth:** максимальная глубина дерева — точка, на которой останавливается разбиение узлов. Меньшее количество сделает модель быстрой, но не точной. Большее количество увеличивает точность, но создает риски переобучения и замедляет процесс.

**min\_samples\_split:**необходимое минимальное количество выборок для разбиения узлов.

**max\_features:**число признаков для поиска лучшей точки для разбиения. Чем больше число, тем лучше результат. Но в этом случае обучение займет больше времени.

**min\_impurity\_split:**порог для ранней остановки роста дерева. Узел разобьется только в том случае, если его точность будет выше указанного порога. Такой метод может служить в качестве компромисса между минимизацией переобучения (высокое значение, маленькое дерево) и высокой точностью (низкое значение, большое дерево).

**presort:** выбор того, нужно ли предварительно сортировать данные для ускорения поиска наилучшего разбиения при подборе. Если данные заранее отсортируются по каждому признаку, то алгоритму обучения будет гораздо проще найти хорошие значения для разбиения.