

Параллельная сегментация точечных изображений деревьев: сравнительный анализ и оптимизация методов машинного обучения для эффективного разделения кроны и листвы

Курсовая работа
Майстренко Марат 323, СКИ ВМК МГУ

Введение

Облако точек – набор точек данных в трёхмерной системе координат.

Кластеризация – процесс поиска сходства с целью объединения в группы.

Сегментация – процедура классификации объектов в однородные группы согласно их схожим характеристикам.

Проблематика и цель работы

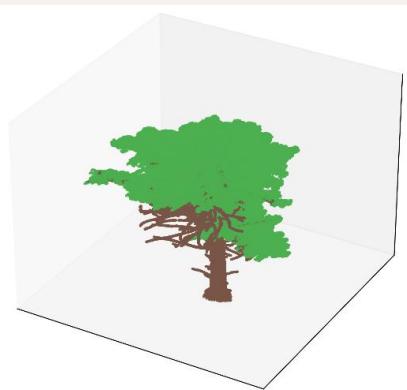
Проблема – большие объемы генерируемых данных

Цель работы - оптимизировать и ускорить сегментацию данных, полученных лазерным сканированием, с помощью машинного обучения и параллельных вычислений.

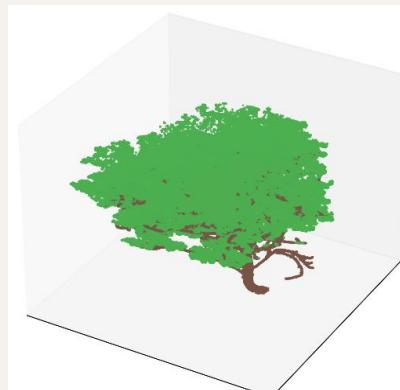
Данные

В датасете 9 деревьев различных пород из центральноевропейских лесов.

Коллекция собрана университетом Гейдельберг с помощью наземного сканирования.



Визуализация файла 1.laz



Визуализация файла 8.laz

laz-файла	Объем файла	Количество точек
1.laz	78.23 мб	6 382 773 шт
2.laz	127.32 мб	10 627 158 шт
3.laz	35.47 мб	2 611 685 шт
4.laz	19.73 мб	1 710 699 шт
5.laz	112.19 мб	9 707 837 шт
6.laz	39.63 мб	3 936 054 шт
7.laz	15.76 мб	1 322 666 шт
8.laz	67.93 мб	5 852 254 шт
9.laz	6.33 мб	479 034 шт

Характеристики laz-файлов в датасете

Конвейер сегментации облаков точек



Предобработка

- **Линейный индекс** $L_\lambda = \frac{\lambda_2 - \lambda_1}{\lambda_2}$
- **Планарный индекс** $P_\lambda = \frac{\lambda_1 - \lambda_0}{\lambda_2}$
- **Рассеянный индекс** $S_\lambda = \frac{\lambda_0}{\lambda_2}$
- **Кривизна** $C_\lambda = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$
- **Отношение с. з.** $R_\lambda = \frac{\lambda_1}{\lambda_2}$

Предобработка:

Координаты – MinMaxScaler()

Признаки – StandartScaler()

Метрики оценки качества сегментирования

- верно-положительные (TP)
 - ложно-положительные (FP)
- верно-отрицательные (TN)
 - ложно-отрицательный (FN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F-score} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * TP}{TP + \frac{(FN + FP)}{2}}$$

Распараллеливание

1) NumPy:

Автоматическое, неэффективное в контексте нашей задачи

2) Joblib:

Отдельная библиотека, распараллеливание цикла for

```
features = Parallel(n_jobs=-1)(delayed(compute_features_for_point)(i, points,  
indices) for i in range(len(points)))
```

3) Sklearn:

Гипер-параметр n_jobs при инициализации метода

Условия вычислений

Временные эксперименты –

12th Gen Intel(R) Core (TM) i5-1240P 1.70 GHz:

4 высокопроизводительных и 8 энергоэффективных ядер, 16 потоков

Качественные эксперименты –

Intel Xeon Platinum 8168:

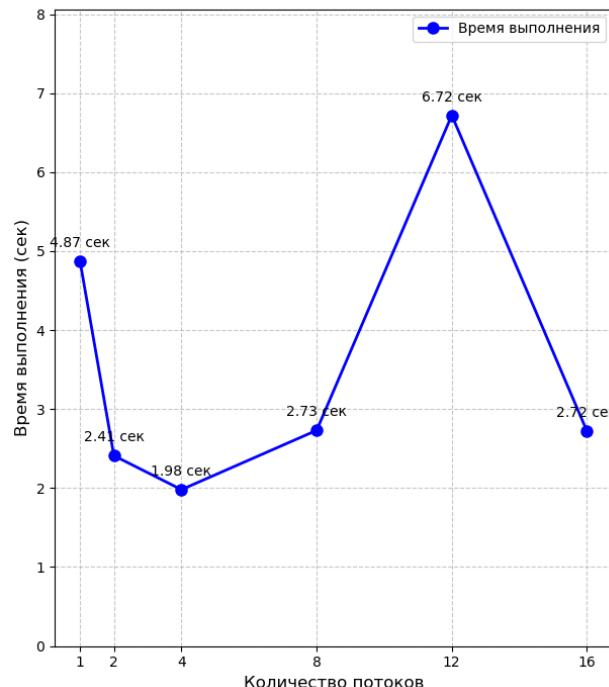
48 физических ядер, 96 логических потоков

Версии библиотек –

Python: 3.12.9 NumPy: 2.2.2 Joblib: 1.4.2 Sklearn: 1.4.1

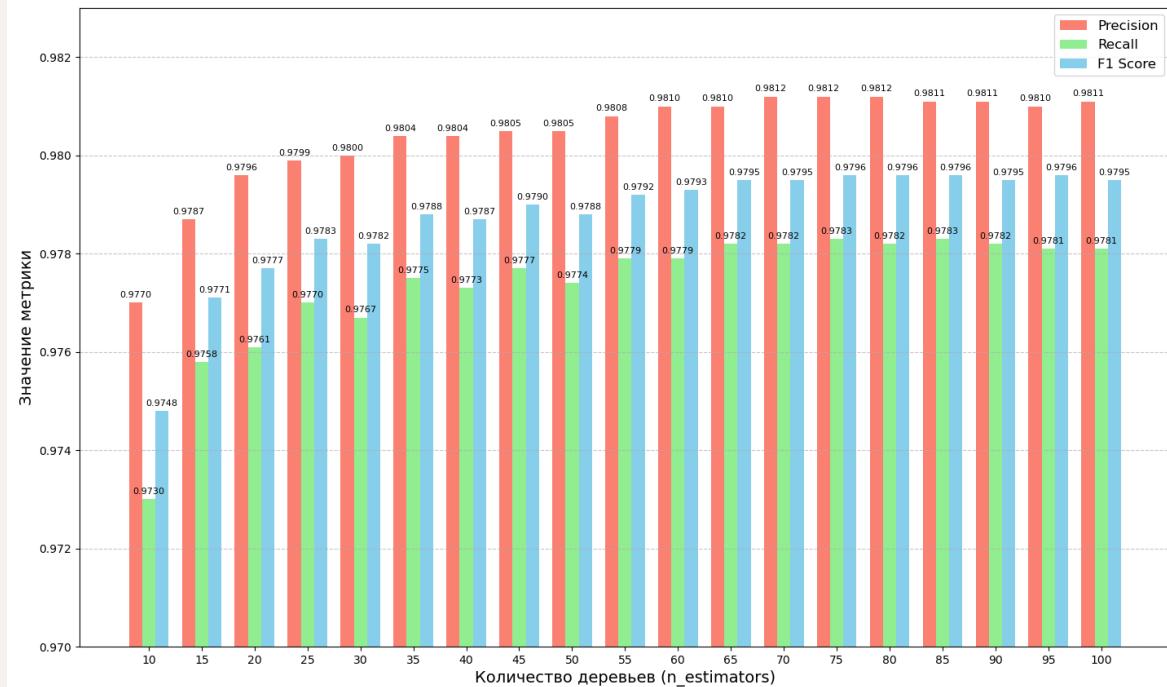
Результаты экспериментов

Зависимость времени выполнения от количества потоков



Зависимость времени выполнения от количества потоков (Joblib). Файл: 9.laz

Метрики RandomForest в зависимости от количества деревьев



Зависимость метрик качества, полученных на файле 9.laz, от количества базовых моделей в ансамбле

Выводы

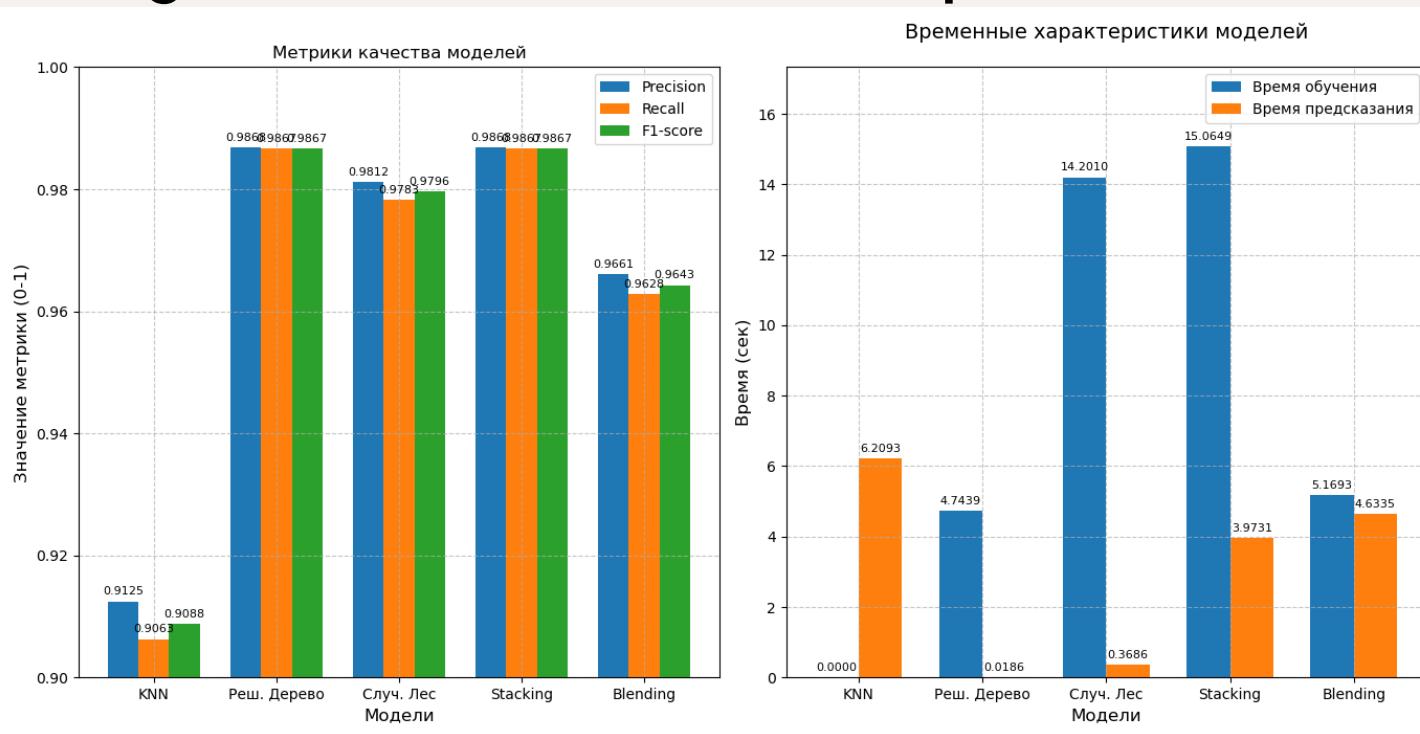
Оптимальные параметры –

- 50 соседей для вычисления ковариационной матрицы признаков
- 8 соседей для KNN
- 75 деревьев для Случайного Леса.

Распараллеливание –

- NumPy неэффективен
- Joblib (n_jobs=4)
- SkLearn (n_jobs=-1)

Результаты экспериментов



Результаты метрик качества и времени алгоритмов на файле 9.laz из датасета

Результаты

Файлы	Точность			Время	
	Precision	Recall	F1-score	Обучения	Предсказания
1.laz	0.9825	0.9823	0.9824	103.8032 сек.	0.4371 сек.
2.laz	0.9802	0.9806	0.9804	180.9451 сек.	5.0195 сек.
3.laz	0.9872	0.9873	0.9872	36.6573 сек.	0.1253 сек.
4.laz	0.9773	0.9775	0.9774	23.5695 сек.	0.0847 сек.
5.laz	0.9878	0.9881	0.9879	146.4735 сек.	0.4414 сек.
6.laz	0.9868	0.9870	0.9869	50.6510 сек.	0.1826 сек.
7.laz	0.9866	0.9868	0.9867	14.9511 сек.	0.0543 сек.
8.laz	0.9900	0.9899	0.9900	87.4916 сек.	0.2618 сек.
9.laz	0.9868	0.9867	0.9867	4.7439 сек.	0.0186сек.

Результаты метрик качества и времени алгоритма Решающее Дерево на файлах в датасете

Основные результаты

1. Распараллеливание вычислений

Определены эффективные способы ускорения вычислений: Joblib даёт прирост в 4 раза, Sklearn (`n_jobs = -1`) — до 12 раз при обучении моделей.

2. Классификация точек облаков

С помощью вычислительных экспериментов выявлены наилучшие значения гипер-параметров для всех исследуемых моделей.

3. Сравнение параметров и оптимизация гиперпараметров

Проведено сравнение моделей с оптимальным значением параметров. Показано, что лучшая модель – Решающее дерево.

Спасибо за внимание!