Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Кластеризация данных LiDAR

Студент ___ИУ5-61Б__ _____ __**Коваленко А.В .**__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ _____**Канев А. И.**___
(Подпись, дата) (И.О.Фамилия)

Москва, 2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ___ИУ5___
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме _____Кластеризация данных LiDAR_____

_____

_____

Студент группы ____ИУ5-61Б____

_____Коваленко Алексей Викторович_ _____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____Исследовательская_____

Источник тематики (кафедра, предприятие, НИР) _____НИР_____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание _Исследовать использование методов машинного обучения для решения задачи сегментации деревьев из облака точек

_____

_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _28_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания «07» февраля 2023 г.

Руководитель НИР _____ **Канев А.И.**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **Коваленко А.В.**
(Подпись, дата) (И.О.Фамилия)

Содержание

ВВЕДЕНИЕ

LiDAR – технология, предназначенная для определения расстояния до какой-либо точки от точки измерения. Она реализована при помощи замера времени, которое необходимо световому лучу для достижения точки и возвращения обратно в сенсор лидара, после чего происходит расчёт расстояния на основе полученных данных. Данная технология имеет широкий спектр применения, который включает в себя геодезию, изучение атмосферы, мониторинг лесов, строительство и архитектуру, также при работе в подводных условиях в мирных и военных целях, а также в робототехнике и машинном зрении.

По итогам своей работы лидар возвращает облако точек, которое соответствует точкам пространства, которые были засняты в ходе работы устройства. Само по себе облако является сплошным и не различает отдельные объекты, чтобы различить объекты или разделить облако на различные объекты применяется сегментация. Данная операция, в общем, является широко используемой в различных областях, таких как машинное обучение, анализ данных, обработка изображений и других.

Целью этой работы является исследование методов применения машинного обучения для выделения деревьев из облака точек, а затем дальнейшее обучение модели для анализа полученных сегментов. Задачами данной работы являются сегментация облака точек при помощи методов машинного обучения для получения деревьев, а также обучение модели машинного обучения для определения является ли полученное облако точек деревом.

1 Постановка задачи

В результате наземной съемки при помощи лидара было получено трехмерное облако точек. Это облако необходимо сегментировать на отдельные кластеры, которые будут представлять собой отдельное дерево, а после обучить модель на полученных результатах.

Современный мир требует высокой заботы об окружающей среде, в результате чего возрастает необходимость в средствах для отслеживая состояния оной.

Особенно данная тенденция является важной в сфере лесного хозяйства, так как деревья играют огромнейшую роль в работе экосистем на нашей планете. Автоматизация работы над отслеживанием лесов является довольно перспективным направлением в данной области. Необходимо отслеживать, не только здоровье самих деревьев, но и их разнообразие, так же отчасти в природе необходимо наличии и плохих деревьев для развития экосистем, однако они должны быть в малых количествах. Поэтому необходимо отслеживать и анализировать состояние леса. Из-за неэффективности и высоких затрат на ручной анализ данных было принято решение осуществить анализ выбранный лес при помощи дистанционной съемки, а затем привести анализ полученных данных при помощи методов машинного обучения [1].

Применение лидара для анализа подобных данных позволяет осуществить быстрый и качественный анализ леса на определенной площади. Среди того, что может предоставить лидар есть: анализ различных высотных уровней представленного пространства, анализ густоты представленного массива, анализ состояния леса в целом и много другое, что может быть использовано для анализа [2].

Автоматизированная система по анализу деревьев в различных ситуациях является важной задачей для лесного и сельского хозяйства. Например, при валке леса, на выбор дерева влияют факторы, описывающие само дерево, среди которых его высота и толщина, род дерева, возраст, и здоровье дерева. Все эти детали делают лес неоднородным, что усложняет задачу анализа, и на данный

момент самым надежным способом анализа деревьев является использование человека-оператора, который будет анализировать и принимать решения. Но с развитием моделей машинного обучения и технологий с ними связанных человек получит способ упростить себе работу с представленными задачами [3].

Данные о деревьях, такие как здоровье, физические показатели и местоположение, имеют большое значение для работы коммерческих лесоводств. Использование автоматизированных систем позволит им использовать в разы меньше ресурсов для лесозаготовительного процесса [4][5].

Также большую роль сегментация придает в задачах изучения антропогенного влияния на лесной массив, а также в задачах улучшения антропогенного взаимодействия с природой, такой как прокладывание маршрутов через лесополосу, для развития логистики. Влияние же человека рассматривается в ситуации негативных изменений в виде эрозии, болезней, засух и других негативных последствий. И опять же сегментация позволяет получить широкие данные о состоянии и показателях отдельно взятого дерева, после проведения анализа, что имеет широкое применение и положительное влияние на ведение лесозаготовительное деятельности [5].

2 Описание данных, используемых методов и метрик

Используемые данные представляют собой двумерный массив координат 10000 точек (Рисунок 2.1), полученных в результате съемки лидаром трех близких друг к другу деревьев (вариант 0). Точки, принадлежащие стволу, распределены с более высокой плотностью, чем точки, принадлежащие листве деревьев, что может сказаться на точности методов машинного обучения.

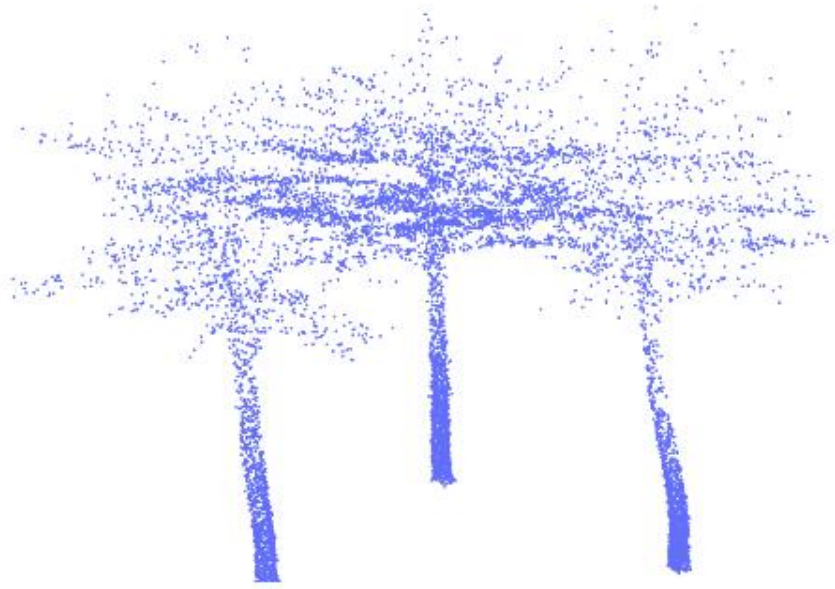

Рисунок 2.1 – облако точек

Для анализа представленных данных будет использован метод DBSCAN. Он представляет собой плотностный алгоритм пространственной кластеризации с присутствием шума. Для проведения анализа необходимо представить 2 параметра: eps – радиус ε-окрестности и количество соседних точек min_samples. Алгоритм сначала выбирает корневую точку на основе того, что в ε-окрестности есть min_samples количество точек. Затем алгоритм начинает проходить по соседим точкам. Если соседняя точка является корневой, то она добавляется в обход, иначе – нет. По такому методу все точки, которые не являются корневыми, будут считаться выбросами. Для оценки качества результатов кластеризации выбрана метрика Silhouette Score, так же визуальный анализ на ранней стадии анализа.

3 Сегментация деревьев из облака точек

Для сегментации точек из облака был использован метод DBSCAN для кластеризации и методы для чтения pcd файлов и представления полученных результатов. Была проведена первоначальная сегментация облака с параметрами eps = 0,5 и min_samples= 140. Полученный результат был крайне

неудовлетворительным, так как он распознал всего лишь 1 дерево из 3 (Рисунок 3.1).

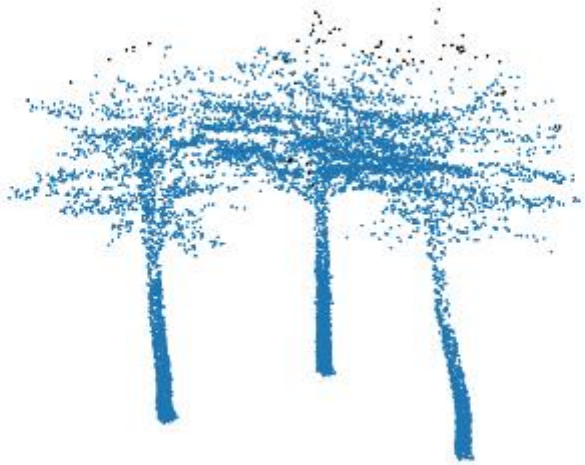

Рисунок 3.1 – результат сегментации при eps = 0,5, min_samples = 140

Было принято решение провести grid search анализ для поиска подходящих параметров. Для этого eps будет перебираться от 0,02 до 0,55 с шагом 0,01, а min_samples будет перебираться от 10 до 300 с шагом 300. В ходе перебора мы создадим DataFrame, в котором будет сохранять только те параметры, при которых было получено только 3 дерева, или же всего 4 кластера (3 дерева и выбросы) (Рисунок 3.2, Рисунок 3.3).

```
Ввод [64]: %%time
eps_list = np.arange(0.02,0.55,0.01)
min_pts_list = np.arange(10,300,5)
res = []

for e in eps_list:
    for m in min_pts_list:
        pca_dbscan_cluster = DBSCAN(eps=e, min_samples=m).fit(X)
        if len(np.unique(pca_dbscan_cluster.labels_)) == 4:
            res.append((e,m,4))
    print (f"ended working with eps={e} with resust as {len(res)}")

ended working with eps=0.02 with resust as 0
ended working with eps=0.03 with resust as 0
ended working with eps=0.039999999999999994 with resust as 0
```

Рисунок 3.2 – код создания grid search для поиска параметров

Ввод [66]:
```
newdf = pd.DataFrame(res, columns=["eps","min_pts","size"])
newdf
```

Out[66]:

	eps	min_pts	size
0	0.05	40	4
1	0.05	45	4
2	0.05	50	4
3	0.05	65	4
4	0.06	45	4
...	...	...	...
1018	0.53	295	4
1019	0.54	280	4
1020	0.54	285	4
1021	0.54	290	4
1022	0.54	295	4

1023 rows × 3 columns

Рисунок 3.3 – Код переформатирования массива в датафрейм

При малых eps и min_samples кластеры содержат малое количество тесно сгруппированных точек, подавляющее большинство точек определяется как шумы (Рисунок 3.4).

Ввод [13]:
```
eps = 0.05
min_pts = 45


pcd, colors = add_color(pcd, colors)
X = np.asarray(pcd.points)
max_label, obj_points = None, None

new_pcd, colors, max_label, obj_points = segment_pcd_old(X, pcd, points, eps, min_pts)
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

```
3
```

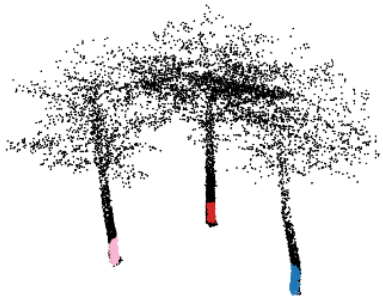

Рисунок 3.4 – результат сегментации при малых eps и min_samples

При увеличении eps и min_samples все больше точек попадают в кластеры, и начинают охватывать большие объемы ствола. Однако вся листва все еще распознавалась как выброс (Рисунок 3.5).

```
Ввод [14]: eps = 0.21
           min_pts = 200


           pcd, colors = add_color(pcd, colors)
           X = np.asarray(pcd.points)
           max_label, obj_points = None, None

           new_pcd, colors, max_label, obj_points = segment_pcd_old(X, pcd, points, eps, min_pts)
           points = np.asarray(pcd.points)
           new_pcd, colors = add_color(new_pcd, colors)

           draw_plot(points, colors)

           3
```

Рисунок 3.5 – результаты сегментации при eps = 0,21, min_samples = 200

При eps=0,41 и min_samples=160 наблюдается довольно качественная кластеризация, где большая часть листвы попала в соответствующие кластеры. А также при увеличении параметров кластеры начинали перекрывать друг друга, в результате чего, крона одного дерева сегментировалась в соседнее дерево. (Рисунок 3.6).

```
n_clusters = 4

eps = 0.41
min_pts = 160
```

Ввод [16]:
```
pcd, colors = add_color(pcd, colors)
X = np.asarray(pcd.points)
max_label, obj_points = None, None

new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps, min_pts, n_clusters, True)
print(f"Распознано {max_label} объектов в облаке точек")
```

```
Распознано 3 объектов в облаке точек
```

Ввод [17]:
```
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

Рисунок 3.6 – результаты сегментации при eps = 0,41, min_samples = 160

Для улучшения результатов кластеризации была проведена операция перебора в окрестностях eps = 0,41, min_samples = 160. Для поиска наиболее успешных параметров была использована метрика silhouette_score (Рисунок 3.7).

Ввод [81]:
```
%%time
eps_list = np.arange(0.39,0.45,0.0025)
min_pts_list = np.arange(120,200,1)
res = []

for e in eps_list:
    for m in min_pts_list:
        pca_dbscan_cluster = DBSCAN(eps=e, min_samples=m).fit(X)
        if len(np.unique(pca_dbscan_cluster.labels_)) == 4:
            sil = metrics.silhouette_score(X, pca_dbscan_cluster.labels_)
            res.append((e,m,3,sil))
    print (f"ended working with eps={e} with resust as {len(res)}")
increaseddf = pd.DataFrame(res, columns=["eps","min_pts","size","sil_score"])
increaseddf
```

```
ended working with eps=0.39 with resust as 14
ended working with eps=0.3925 with resust as 32
```

Рисунок 3.7 – Поиск параметров в окрестности eps = 0,41, min_samples = 160

Наилучшие результаты были достигнуты при eps=0,39 и min_samples=149 с silhouette_score = 0. 346324 (Рисунок 3.8, Рисунок 3.9).

```
Ввод [82]: sorted = increaseddf.sort_values('sil_score', ascending = False)
           sorted
```

Out[82]:

	eps	min_pts	size	sil_score
12	0.3900	149	3	0.346324
13	0.3900	150	3	0.346321
31	0.3925	152	3	0.346038
10	0.3900	147	3	0.345150
11	0.3900	148	3	0.344790
...	...	...	...	...
35	0.3950	139	3	0.108387
51	0.3975	141	3	0.108383
36	0.3950	140	3	0.108295
73	0.4025	145	3	0.103430
63	0.4000	143	3	0.102898

210 rows × 4 columns

Рисунок 3.8 – результаты перебора в окрестности eps = 0,41, min_samples = 160, отсортированные по наилучшему силуэту

```
Ввод [30]: eps = 0.39
           min_pts = 149


           pcd, colors = add_color(pcd, colors)
           X = np.asarray(pcd.points)
           max_label, obj_points = None, None

           new_pcd, colors, max_label, obj_points = segment_pcd_old(X, pcd, points, eps, min_pts)
           points = np.asarray(new_pcd.points)
           new_pcd, colors = add_color(new_pcd, colors)

           draw_plot(points, colors)
```

Рисунок 3.9 – результаты сегментации при eps=0,39 и min_samples=132

4 Сегментация облака точек большой размерности

Следующей нашей задачей является сегментация облака большой размерности, чтобы получить деревья для дальнейшего создания модели. Для этого изначальное было разделено на сегменты, а после проанализировано. Но начнем сначала.

Имеется облако точек, которое состоит из 46744308 точек и представлено в виде двумерного массива (Рисунок 4.1), полученное в результате съемки лидаром множества деревьев.

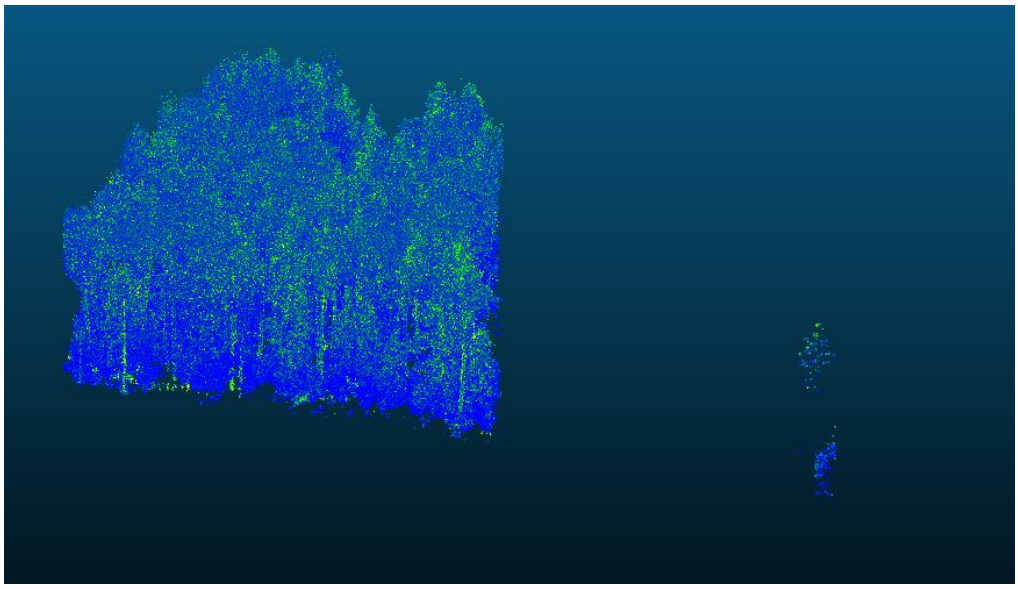

Рисунок 4.1 – облако точек большой размерности

Как видно на изображении имеется выброс, который мы удалим, также выровняв вместе в ними уровень почвы (Рисунок 4.2). Также вместе с этим уменьшилось количество точек – до 45863756.

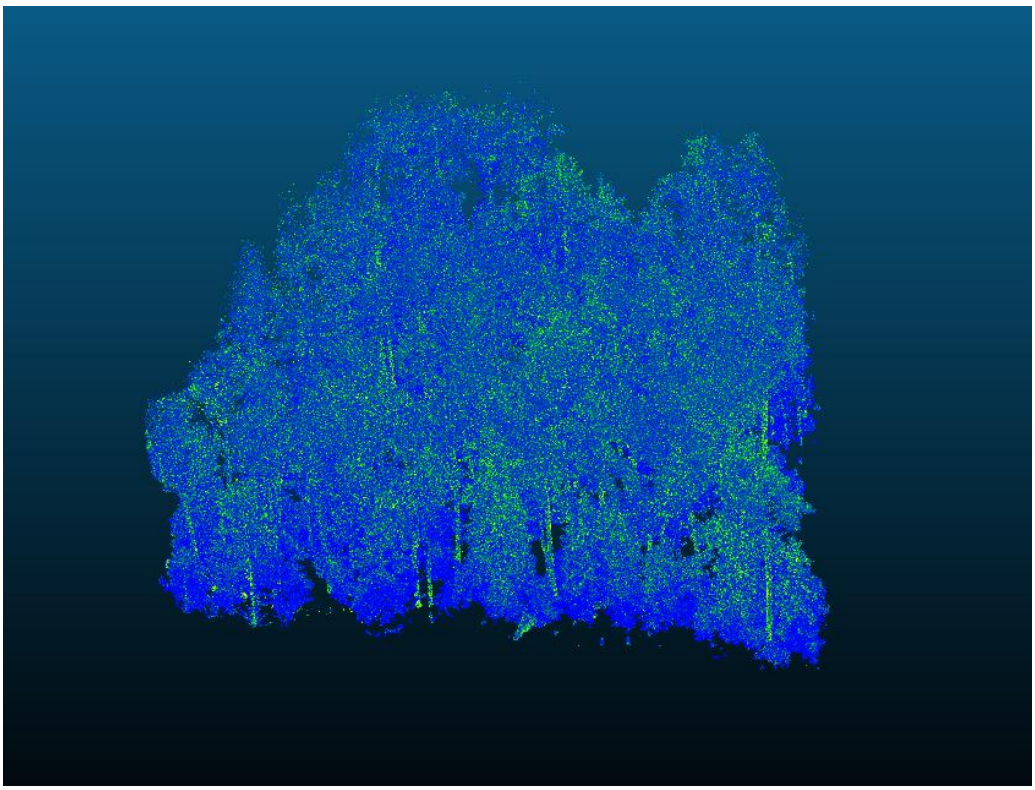

Рисунок 4.2 – очищенное от выбросов облако точек

В силу специфики нашей задачи разделим облако на несколько небольших сегментов по несколько деревьев в каждом. Каждый такой сегмент состоит примерно из 4-5млн точек. В результате создания таких фрагментов, получилось 13 небольших сегментов (Рисунок 4.3, Рисунок 4.4).

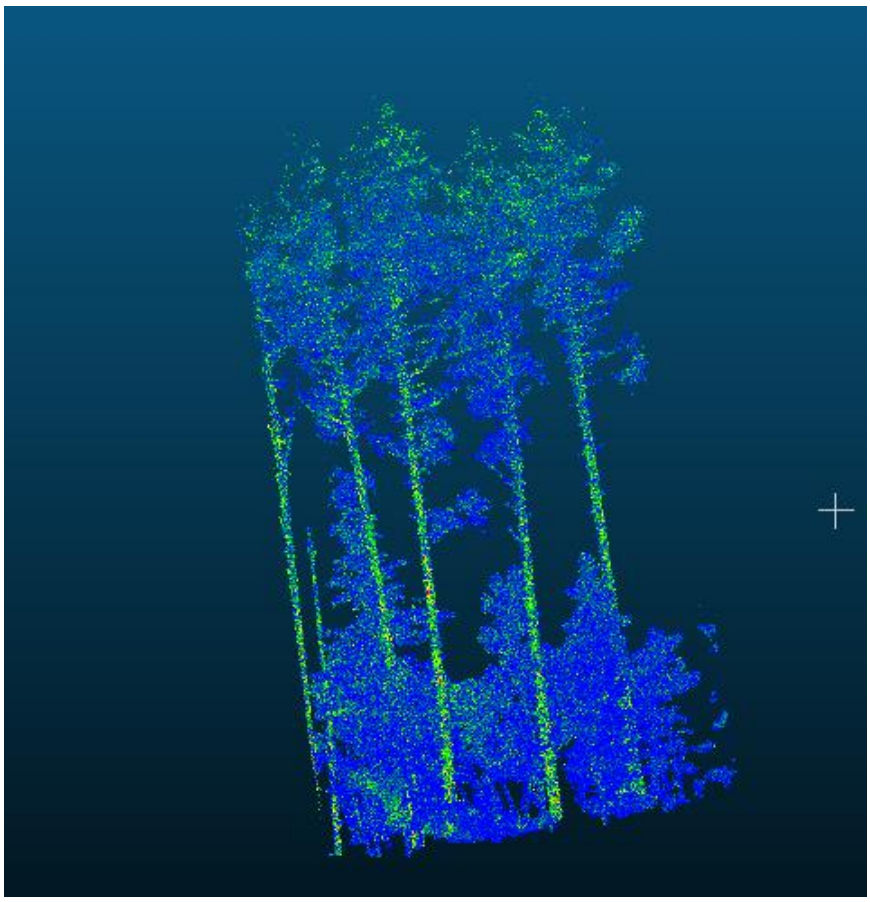

Рисунок 4.3 – Пример одного из фраментов

1.pcd	01.06.2023 17:18	Файл "PCD"	115 391 КБ
2.pcd	01.06.2023 17:18	Файл "PCD"	72 800 КБ
3.pcd	01.06.2023 17:19	Файл "PCD"	48 173 КБ
4.pcd	01.06.2023 17:19	Файл "PCD"	50 486 КБ
5.pcd	01.06.2023 17:19	Файл "PCD"	60 910 КБ
6.pcd	01.06.2023 17:19	Файл "PCD"	110 344 КБ
7.pcd	01.06.2023 17:19	Файл "PCD"	43 496 КБ
8.pcd	01.06.2023 17:19	Файл "PCD"	136 953 КБ
9.pcd	01.06.2023 17:19	Файл "PCD"	98 531 КБ
10.pcd	01.06.2023 17:19	Файл "PCD"	59 412 КБ
11.pcd	01.06.2023 17:19	Файл "PCD"	66 368 КБ
12.pcd	01.06.2023 17:19	Файл "PCD"	98 142 КБ
13.pcd	01.06.2023 17:19	Файл "PCD"	61 601 КБ

Рисунок 4.4 – Все полученные фрагменты

Также был применен встроенный алгоритм CloudCompare по уменьшению размерности облака, на основе минимального расстояния между точками, что сильно проредило основания у стволов деревьев, уменьшив количество точек в ~10 раз у каждого набора деревьев. Полученные фрагменты будут являться запасными если работа с изначальными облаками окажется неудачной.

Для дальнейшей сегментации был применен DBSCAN для сегментации разряженного облака точек. Для него были выбраны произвольные параметры eps = 0.42 и min_samples = 200 (Рисунок 4.5). Однако и при дальнейшем переборе параметров не было получено удовлетворительных результатов (Рисунок 4.6).

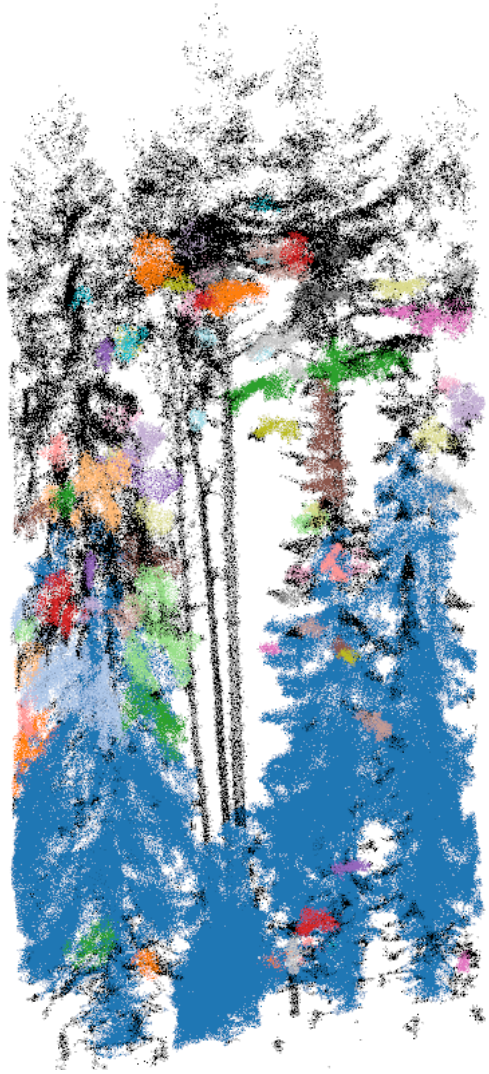

Рисунок 4.5 – Сегмент 1 при eps = 0.42 и min_samples = 200

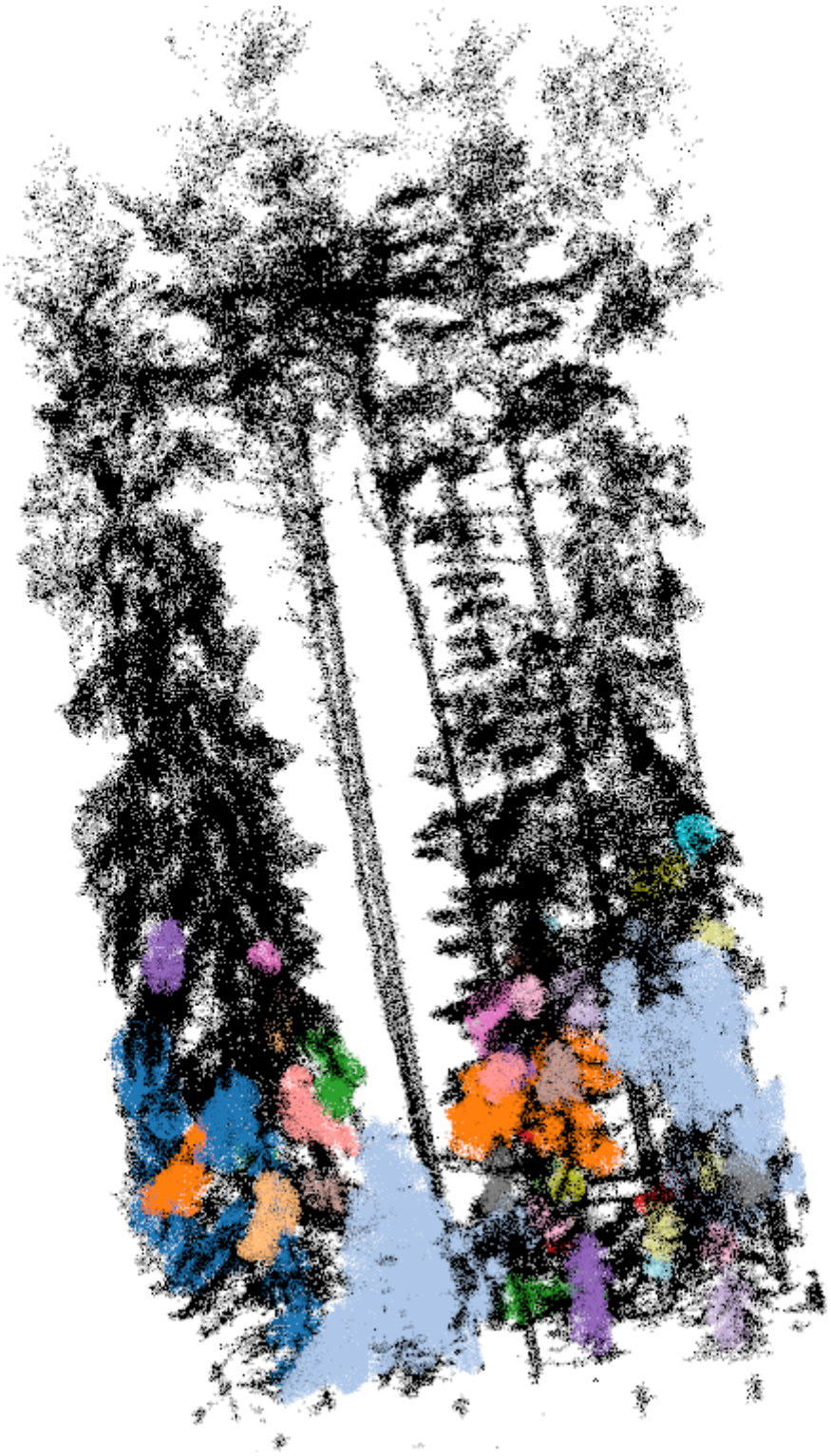

Рисунок 4.6 – Сегмент 1 при eps = 0.35 и min_samples = 150

В силу неуспешности попыток сегментации было принято решение искать другой способ сегментации деревьев. Одним из решений был найден способ сегментации через вокселизацию леса, а затем дальнейшее построение графа, на основе которого будет уже создан сегментированный лес. В дальнейшей работе будет использован метод сегментации, разработанный

Юлией Рыжковой в схожей научной работе, который был разрешен мне к применению в моей научной работе.

Для работы с вокселями и точками был применён алгоритм уменьшения размерности, который уменьшает количество точек до желаемого размера с учетом их высоты (Рисунок 4.7). Таким образом, можно будет использовать алгоритм вокселизации, так как иначе он являлся слишком ресурсоемким из-за количества точек.

```python
# до ??? точек
def down_pcd1(data, num_points):
    if data.shape[0] > num_points:
        factor = data.shape[0] // num_points
    else:
        factor = 1
    down_data = data[::factor]

    z_list=[d[2] for d in down_data]
    z_max=max(z_list)



    res_data=[]
    for point in down_data:
        if z_max-point[2] < random.random() * z_max:
            res_data.append(point)
    print(f'amount of points: {len(res_data)}')



    return np.asarray(res_data, dtype=np.float32)
```

Рисунок 4.7 – Алгоритм уменьшения размерности

Одним из минусов текущей версии алгоритма сегментации через воксели является высокая зависимость от ровности земли, отчего в ходе сегментации возникали проблемы из-за кривизны земли в наборах данных даже после их выравнивания, в результате чего случались ошибки в сегментации. Однако результаты, которые были получены таким образом, оказались в разы лучше сегментации DBSCAN (Рисунок 4.8, Рисунок 4.9).

Рисунок 4.8 – Фрагмент 2 в случае сегментации через воксели

Рисунок 4.9 – Фрагмент 8 в случае сегментации через воксели

После сегментации каждого участка, полученные сегменты были сохранены отдельно в виде .pcd файла и были объединены в одну группу под заголовком сегмента (Рисунок 4.10).

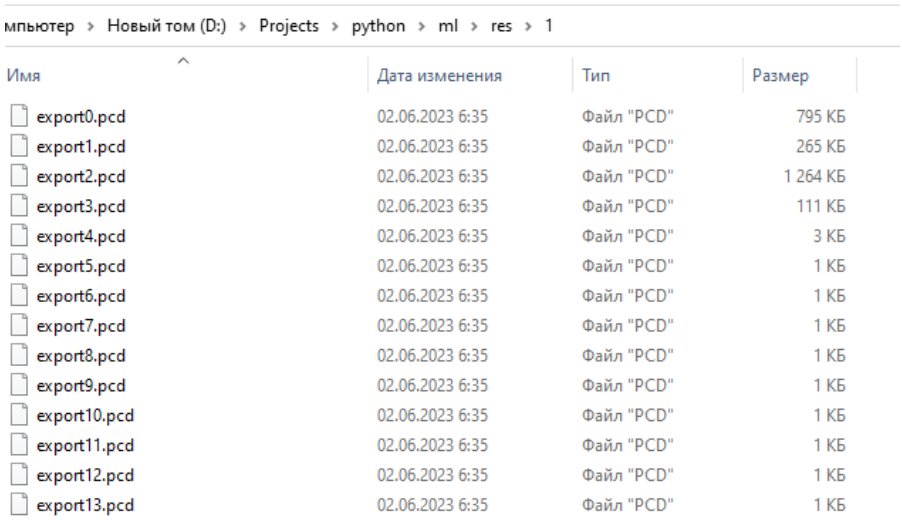

Рисунок 4.10 – Результаты сегментации 1-го участка

После анализа всех участков был проведен визуальный анализ и сортировка полученных результатов после сегментации. Конечно, из-за кривизны почвы даже после очистки, сегментация не всегда получалась удачной, однако полученных данных было достаточно для создания набора данных для обучения модели, о чем и пойдет речь далее.

5 Создание модели на основе полученных данных

Для этого все полученные сегменты были разделены на 2 группы “tree” (“дерево”) и “not tree” (“не дерево”) (Рисунок 5.1, Рисунок 5.2, Рисунок 5.3).

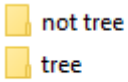

Рисунок 5.1 – Наборы данных

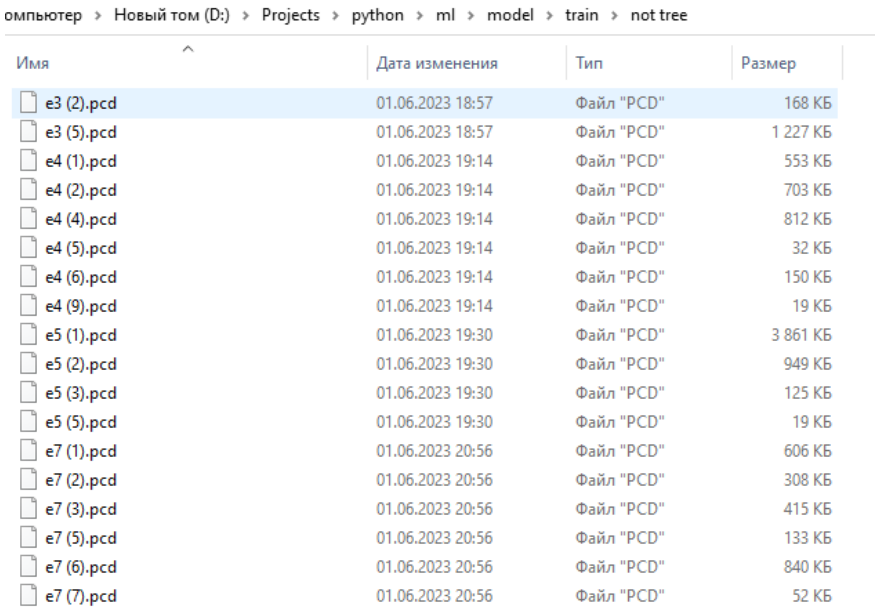

Рисунок 5.2 – Набор данных “Не деревья”

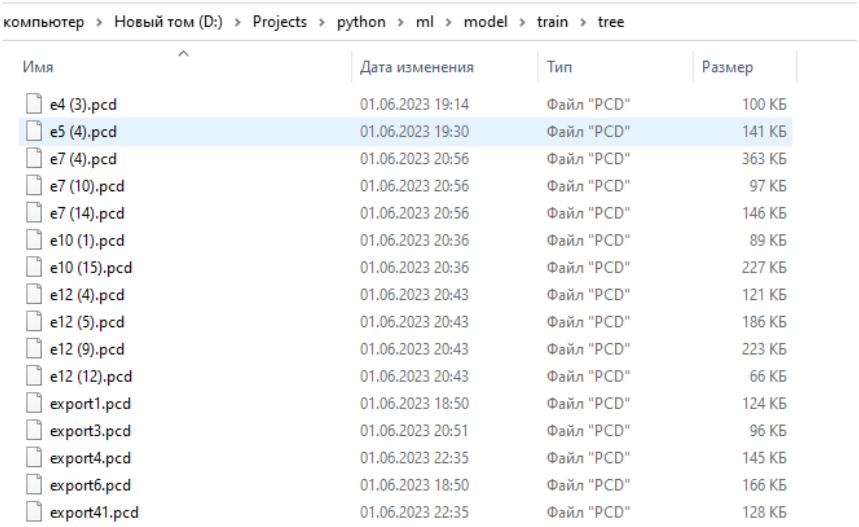

Рисунок 5.3 – Набор данных “Деревья”

Затем было проведено разделение на обучающую и тестовую выборки. Для этого из полученных данных были произвольно выбраны 20% данных и перенесены в набор для проверки, оставшиеся данные будут же использованы для обучения модели.

Модель, которую я буду обучать работает с треугольными mesh объектами, что отличаются от облака точек, поэтому была создана функция, которая преобразует облака точек в mesh объект (Рисунок 5.4).

```
Ввод [40]: def to_trimesh(path):
               pcd = o3d.io.read_point_cloud(path)
               pcd.estimate_normals()
               distances = pcd.compute_nearest_neighbor_distance()
               avg_dist = np.mean(distances)
               radius = 1.5 * avg_dist

               mesh = o3d.geometry.TriangleMesh.create_from_point_cloud_ball_pivoting(
                         pcd,
                         o3d.utility.DoubleVector([radius, radius * 2]))

               # create the triangular mesh with the vertices and faces from open3d
               mesh = trimesh.Trimesh(np.asarray(mesh.vertices), np.asarray(mesh.triangles),
                                      vertex_normals=np.asarray(mesh.vertex_normals))

               trimesh.convex.is_convex(mesh)
               return mesh
```

Рисунок 5.4 – Алгоритм трансформации облака точек в mesh-объект.

Для упрощения и облегчения создания модели было уменьшено количество точек в каждом объекте до 2048, что было сделано при загрузке деревьев в выборки(Рисунок 5.5).

```
Ввод [50]: def parse_dataset(num_points=2048):
               train_points = []
               train_labels = []
               test_points = []
               test_labels = []
               folders = ['tree', "not tree"]

               for i, folder in enumerate(folders):
                   print("processing class: {}".format(os.path.basename(folder)))
                   # store folder name with ID so we can retrieve later
                   # gather all files

                   for f in os.listdir(f".{DATA_DIR}/train/{folder}"):
                       trm = to_trimesh(f".{DATA_DIR}/train/{folder}/{f}")
                       train_points.append(trm.sample(num_points))
                       train_labels.append(i)

                   for f in os.listdir(f".{DATA_DIR}/test/{folder}"):
                       trm = to_trimesh(f".{DATA_DIR}/test/{folder}/{f}")
                       test_points.append(trm.sample(num_points))
                       test_labels.append(i)

               return (
                   np.array(train_points),
                   np.array(test_points),
                   np.array(train_labels),
                   np.array(test_labels),
               )
```

Рисунок 5.5 – Алгоритм загрузки данных и их сжатия

Затем были выбраны гиперпараметры для обучения модели и запущено само обучение (Рисунок 5.6, Рисунок 5.7). Обучение происходило через центральный процессор, так как последняя версия tensorflow на Windows не может работать с графическими ускорителями.

Ввод [57]:
```
inputs = keras.Input(shape=(NUM_POINTS, 3))

x = tnet(inputs, 3)
x = conv_bn(x, 32)
x = conv_bn(x, 32)
x = tnet(x, 32)
x = conv_bn(x, 32)
x = conv_bn(x, 64)
x = conv_bn(x, 512)
x = layers.GlobalMaxPooling1D()(x)
x = dense_bn(x, 256)
x = layers.Dropout(0.3)(x)
x = dense_bn(x, 128)
x = layers.Dropout(0.3)(x)

outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs, name="pointnet")
model.summary()
```

```
dropout (Dropout)              (None, 256)          0           ['activation_15[0][0]']

dense_7 (Dense)                (None, 128)          32896       ['dropout[0][0]']

batch_normalization_16 (BatchN  (None, 128)         512         ['dense_7[0][0]']
ormalization)
```

Рисунок 5.6 – Параметры обучаемой модели

Ввод [60]:
```
model.compile(
    loss="sparse_categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=["sparse_categorical_accuracy"],
)

history = model.fit(train_dataset, epochs=100, validation_data=test_dataset)
```

```
al_sparse_categorical_accuracy: 0.7895
Epoch 95/100
3/3 [==============================] - 5s 1s/step - loss: 1.4669 - sparse_categorical_accuracy: 0.7468 - val_los
al_sparse_categorical_accuracy: 0.7895
Epoch 96/100
3/3 [==============================] - 5s 2s/step - loss: 1.4603 - sparse_categorical_accuracy: 0.8101 - val_los
al_sparse_categorical_accuracy: 0.7895
Epoch 97/100
3/3 [==============================] - 5s 1s/step - loss: 1.4567 - sparse_categorical_accuracy: 0.8228 - val_los
al_sparse_categorical_accuracy: 0.7368
Epoch 98/100
3/3 [==============================] - 5s 1s/step - loss: 1.4880 - sparse_categorical_accuracy: 0.7975 - val_los
al_sparse_categorical_accuracy: 0.7368
Epoch 99/100
3/3 [==============================] - 5s 1s/step - loss: 1.4507 - sparse_categorical_accuracy: 0.7722 - val_los
al_sparse_categorical_accuracy: 0.7895
Epoch 100/100
3/3 [==============================] - 5s 2s/step - loss: 1.5688 - sparse_categorical_accuracy: 0.7215 - val_los
al_sparse_categorical_accuracy: 0.7368
```

Рисунок 5.7 – Обучение модели

Затем была проанализирована деятельность модели в ходе обучения (Рисунок 5.8, Рисунок 5.9). Можно сказать, что по ходу обучения модель оказалась достаточно точной для представленных данных, а также в ходе её обучения случился всего 1 прецедент ошибки.

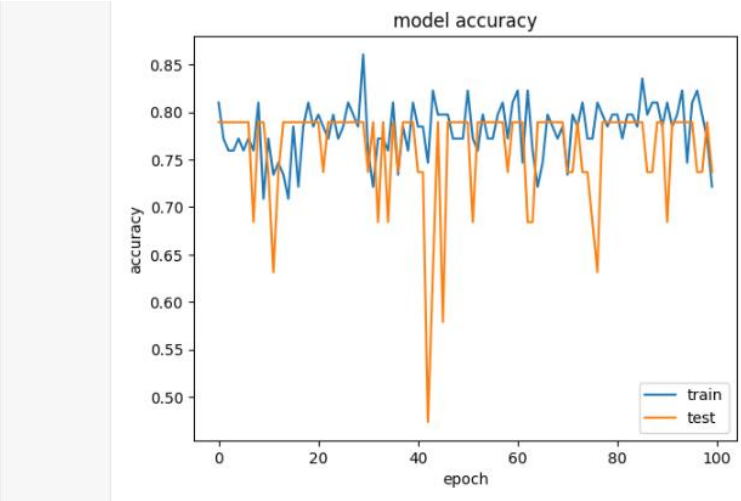

Рисунок 5.8 – Точность модели в процессе обучения

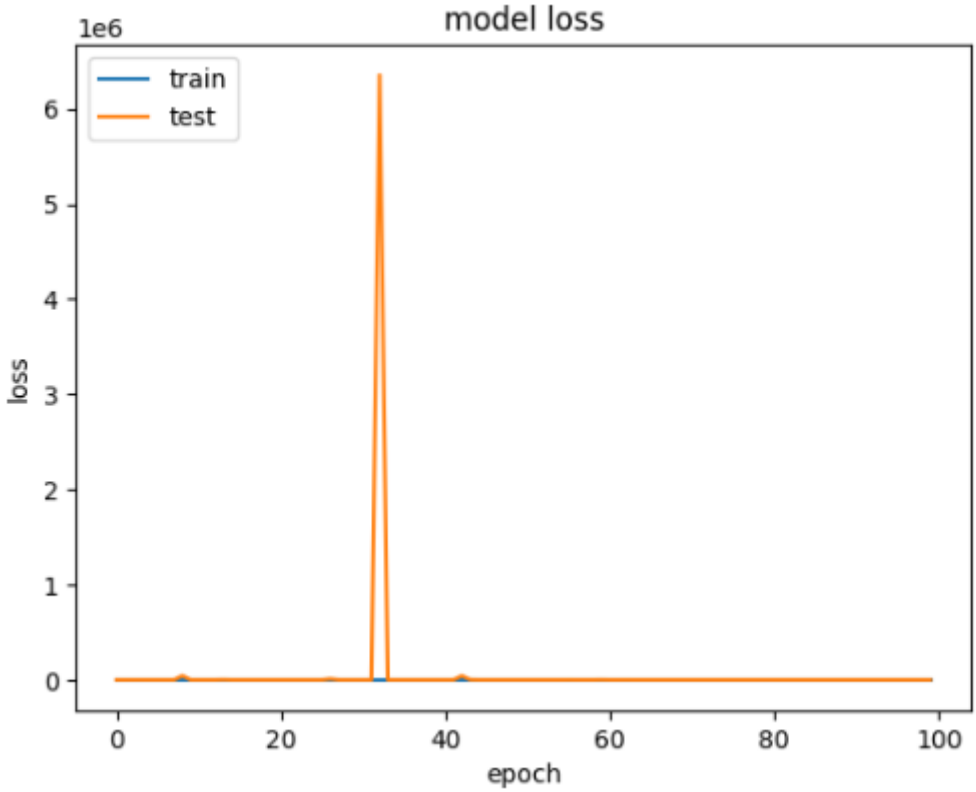

Рисунок 5.9 – Ошибки в процессе обучения

И наконец была создана корреляционная матрица для проведения корреляционного анализа (Рисунок 5.10). На ней видно, что полученные результаты оказались не очень хорошими. Модель предугадала всего 1 дерево из 4, однако она смогла выбрать 13 не деревьев из 15 имеющихся в наборе данных. Я связываю причину такого результата с тем, какой набор данных был использован при обучении модели. В нем было 16 деревьев и 63 не дерева, что сказалось на полученном результате. Также качество сегментации было не на

высоте, из-за чего случались ситуации 2 склеенных деревьев, которые выглядят как одно цельное, что также оказывает влияние на полученную модель (Рисунок 5.11). Причины такой сегментации в проблемах в неровной почве, а также в слишком плотном расположении деревьев, а также в их разновидности, так как по полученной сегментации видно, что есть как и лиственные деревья, так и сосны с елками. Также причина может быть в алгоритме очистки данных от лишних точек.

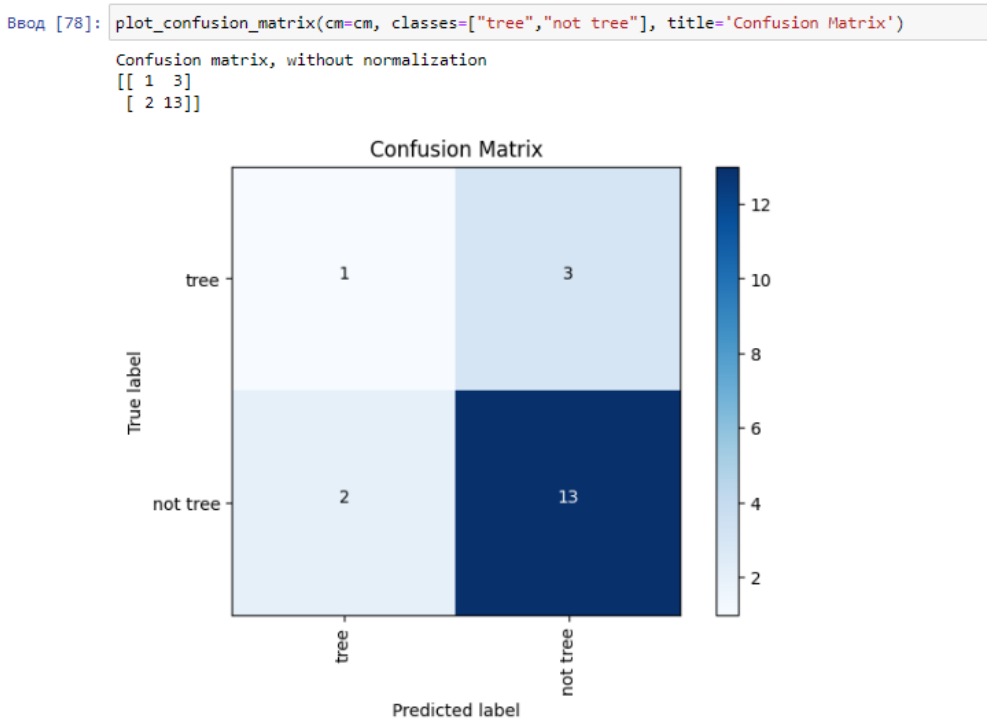

Рисунок 5.10 – Корреляционная матрица

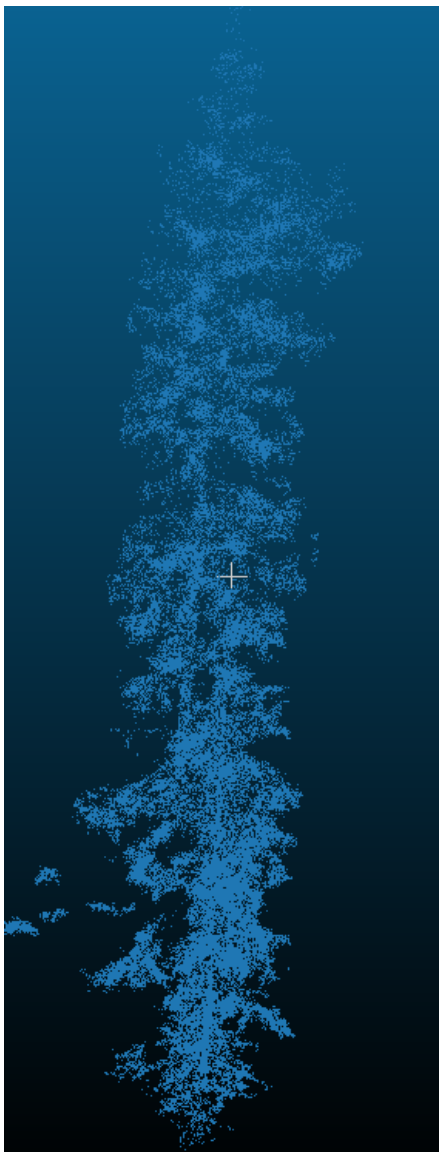

Рисунок 5.11 – “Склеенное дерево” из набора 4

ЗАКЛЮЧЕНИЕ

В результате выполнения научно-исследовательской работы были выполнены следующие задачи:

1. С использованием метода машинного обучения DBSCAN была произведена сегментация деревьев из облака точек. В результате чего было получено три кластера, которые соответствуют 3 деревьям в облаке точек.
2. Была произведена оценка качества кластеризации с использованием метрики Silhouette Score. Наивысшее значение sil_score = 0. 346324, что соответствует eps = 0.39 и min_samples = 149.
3. Была произведена ручная сегментация, а затем каждый из созданных сегментов был впоследствии кластеризирован при помощи метода сегментации через воксели и графы. Это дало значительный прирост в качестве по сравнению с DBSCAN.
4. Была создана вручную выборка из полученных данных сегментации для обучения модели.
5. Была обучена модель бинарной классификации деревьев и не деревьев, на основе собранных данных, а также сделаны выводы о её качестве и причинах такого результата.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Tsz-Chung Wong, Abubakar Sani-Mohhamed, Wei Yao, Marco Heurich. Automatic Classification of Single Tree Decay Stages from Combined ALS Data and Aerial Imagery using Machine Learning: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 04.01.2023. URL: https://arxiv.org/abs/2301.01841 (Дата обращения: 30.04.2023).

2. Ekaterina Kalinicheva, Loic Landrieu, Clément Mallet, Nesrine Chehata. Multi-Layer Modeling of Dense Vegetation from Aerial LiDAR Scans: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 25.04.2022. URL: https://arxiv.org/abs/2204.11620 (Дата обращения: 30.04.2023).

3. Vincent Grondin, Jean-Michel Fortin, Francois Pomerleau, Philippe Giguere. Tree Detection and Diameter Estimation Based on Deep Learning: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 31.10.2022. URL: https://arxiv.org/abs/2210.17424 (Дата обращения: 30.04.2023).

4. Jonathan Williams, Carola-Bibiane Schonlieb, Tom Swinfield, Juheon Lee, Xiaohao Cai, Lan Qie, David A. Coomes. Three-dimensional Segmentation of Trees Through a Flexible Multi-Class Graph Cut Algorithm (MCGC): [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 20.03.2019. URL: https://arxiv.org/abs/1903.08481 (Дата обращения: 30.04.2023).

5. Lloyd Windrim, Mitch Bryson. Forest Tree Detection and Segmentation using High Resolution Airborne LiDAR: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 30.10.2018. URL: https://arxiv.org/abs/1810.12536 (Дата обращения: 30.04.2023).