Projekt uczenia maszynowego: Klasyfikator chmury punktów

Opracowanie:
Patryk Bachanek, Szymon Bieniaszewski,
Paweł Hryń, Łukasz Bochno, Jakub Wróblewski

Spis treści

Wstęp

W dzisiejszym świecie technologii, zbieranie danych o otaczającym nas środowisku jest kluczowym elementem dla między innymi urbanistyki czy geologii. Jednym z popularniejszych i wszechstronnych narzędzi do tego celu są skanery laserowe LIDAR (Light Detection and Ranging), które pozwalają na szybkie i precyzyjne gromadzenie informacji o trójwymiarowej strukturze terenu oraz obiektów znajdujących się na jego powierzchni.

Celem naszego projektu badawczego jest opracowanie i zaimplementowanie klasyfikatora chmur punktów pochodzących z plików LIDAR. Chmury punktów uzyskiwane za pomocą pomiarów LIDAR, reprezentują punkty w trójwymiarowej przestrzeni, które odzwierciedlają powierzchnię terenu oraz obiekty na niej znajdujące się. Klasyfikacja tych chmur punktów ma kluczowe zastosowanie między innymi, w analizie terenu, planowaniu dróg bądź rurociągów czy wykrywaniu linii wysokiego napięcia. Skupimy się na dobrej klasyfikacji gruntu, roślinności(głównie drzew) oraz budynków.

Klasyfikacja chmur punktów pozwala na automatyczne rozpoznawanie różnych typów obiektów na podstawie ich cech geometrycznych, intensywności sygnału oraz innych atrybutów dostępnych w danych LIDAR lub implementacji własnych cech opartych o zgromadzone dane. Wraz z postępem technologicznym oraz rosnącą popularnością metody LIDAR, rośnie potrzeba analizy i interpretacji tych danych, co sprawia, że klasyfikacja staje się istotnym obszarem badań w dziedzinie przetwarzania danych przestrzennych.

W niniejszej dokumentacji przedstawimy proces opracowania klasyfikatora chmur punktów z plików LIDAR, w tym etapy przetwarzania danych, wybór odpowiednich cech oraz proces wyboru i trenowania modelu uczenia maszynowego. Ponadto omówimy napotkane problemy oraz dokładność naszego modelu. Do implementacji klasyfikatora użyjemy języka Python.

Zapoznanie się z danymi oraz ich przetwarzanie

Naszym pierwszym krokiem w celu zrozumienia problemu było zapoznanie się z danymi na których mieliśmy opracować nasz klasyfikator. Są to dane typu LAS format ten opracowany przez ASPRs został specjalnie stworzony do przechowywanie danych chmury punktów LIDAR. Plik tego typu zawiera nagłówek, zmienne rekordy długości (VLR) oraz rekordy punktowe. Nagłówek zawiera informacje o danych, takie jak ich wersja, format punktów (który określa różne wymiary przechowywane dla każdego punktu). Po nagłówku pliki LAS mogą zawierać VLRs (Zmienne Rekordy o Zmiennej Długości). VLRs służą do przechowywania dodatkowych informacji, takich jak SRS (System Odniesienia Przestrzennego), opis dodatkowych wymiarów dodanych do punktów. Rekordy punktowe zawierają już dane chmury punktów które nas interesują. Specyfikacja LAS określa 10 formatów punktów. Nasze dane są zapisane w formacie 2 który wygląda następująco:

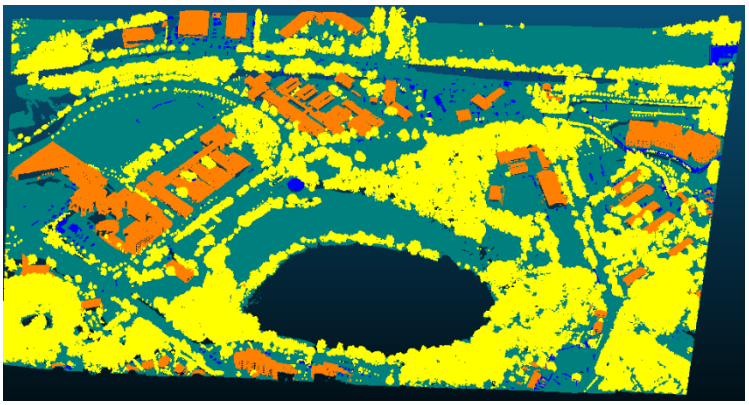
Item	Format	Size	Required
X	long	4 bytes	yes
Y	long	4 bytes	yes
Z	long	4 bytes	yes
Intensity	unsigned short	2 bytes	no
Return Number	3 bits (bits 0-2)	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)	3 bits	yes
Scan Direction Flag	1 bit (bit 6)	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	1 bit	yes
Classification	unsigned char	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	signed char	1 byte	yes
User Data	unsigned char	1 byte	no
Point Source ID	unsigned short	2 bytes	yes
Red	unsigned short	2 bytes	yes
Green	unsigned short	2 bytes	yes
Blue	unsigned short	2 bytes	yes
Minimum PDRF Size		*26 bytes*	

Rys. 1 Tabela formatu pliku las

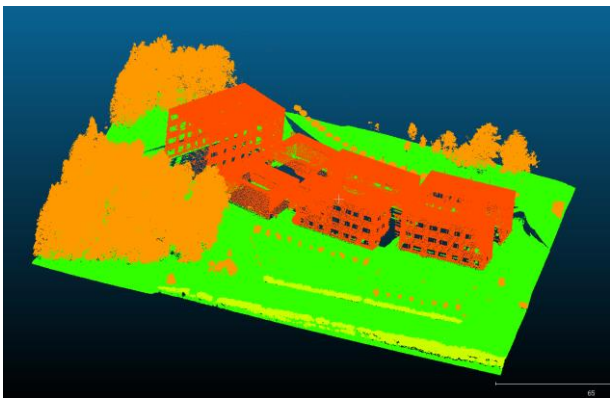
Przetwarzanie danych

W pierwszej kolejności sprawdziliśmy wykorzystując język Python i bibliotekę Pandas czy występują brakujące dane użyliśmy metody any oraz isnull co pokazało ze nasze dane nie są wybrakowane. Zauważyliśmy ze nasze pliki nie są sklasyfikowane użyliśmy do tego programu

CloudCompare i ręcznie sklasyfikowaliśmy chmury punktów odróżniając grunt, roślinność, budynki.



Rys 2. Kortowo plik



Rys. 3 WMII plik

Początkowy stack technologiczny: Python - biblioteki: Pandas, Numpy, laspy(wykorzystywany do obsługi plików LAS)

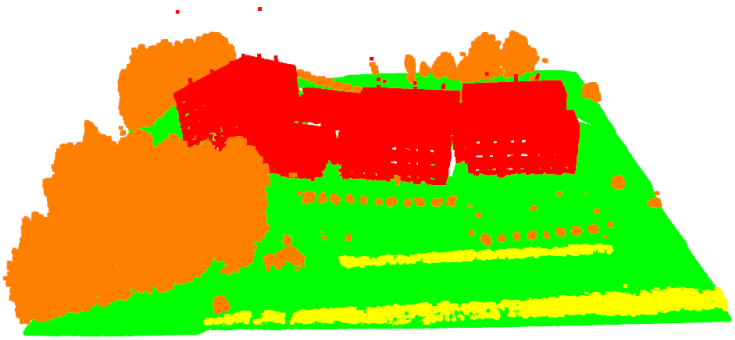
Pierwsze próby uczenia maszynowego

Zaczęliśmy od poszukiwania algorytmów uczenia oraz sieci neuronowej które moglibyśmy wykorzystać. Pierwszym z algorytmów był KNN (k najbliższych sąsiadów) polegający na przydzieleniu klasy decyzyjnej, którą obiekty w określonym zasięgu najliczniej reprezentują.

Posłużyliśmy się gotową implementacją klasyfikatora KNN z biblioteki scikit-learn KNeighborsClassifier z parametrami n_neighbors=50, algorithm='kd_tree' model był trenowany na danych pliku WMII podzielony 80% do treningu 20% cechy które wybraliśmy to x, y, z do testu predykcji. W programie CloudCompare zaklasyfikowany został grunt (2) budynki (6) i roślinność(wysoka (5) i niska(3)). Do miary dokładności będziemy się posługiwać dokładnością - (Accuracy) jest podstawową miarą oceny klasyfikatorów w uczeniu maszynowym. Jest definiowana jako stosunek liczby poprawnie sklasyfikowanych przykładów do całkowitej liczby przykładów, dokładnością zbalansowaną (Balanced Accuracy) to miara stosowana w przypadku, gdy dane są niezbalansowane, tj. liczba przykładów jednej klasy jest znacznie większa niż liczba przykładów innej klasy, oraz metody classification_report, która zawiera: Precision (Precyzja): Procent przykładów, które zostały poprawnie zidentyfikowane jako należące do danej klasy. Recall (Czułość): Procent przykładów należących do danej klasy, które zostały poprawnie zidentyfikowane F1-score: Harmoniczna średnia precyzji i czułości, dająca pojedynczą miarę dla każdej klasy. Support: Liczba rzeczywistych przykładów danej klasy w zbiorze danych. Dzięki temu raportowi w przypadku wieloklasowych problemów klasyfikacji jesteśmy w stanie zobaczyć, jak dobrze model radzi sobie z poszczególnymi klasami.

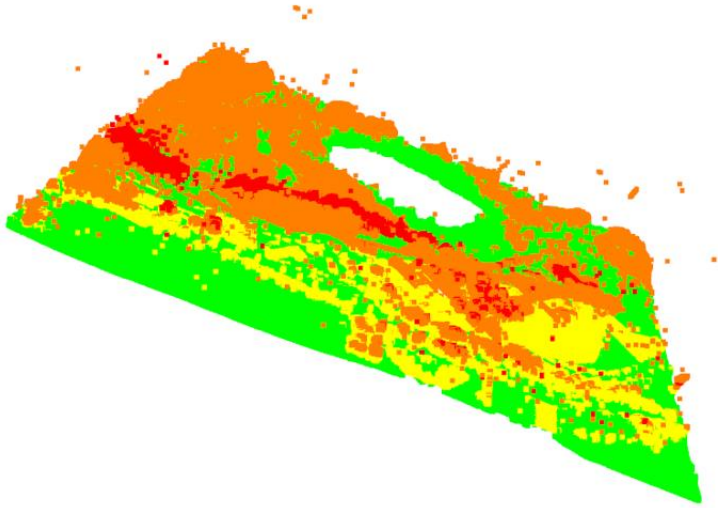
```
Dokładność klasyfikacji: 1.00
Raport klasyfikacji:
              precision    recall  f1-score   support

           2       1.00      1.00      1.00    710947
           3       0.98      0.99      0.99     22846
           5       1.00      1.00      1.00    262511
           6       1.00      1.00      1.00    202277

    accuracy                           1.00   1198581
   macro avg       0.99      1.00      1.00   1198581
weighted avg       1.00      1.00      1.00   1198581

[2, 3, 5, 6]
```

Rys. 4 Wynik pierwszej klasyfikacji WMII



Rys. 5 Wizualizacja klasyfikacji WMII

Dokładność jest tak wysoko ponieważ model był trenowany na tych danych, gdy próbowaliśmy wykonać predykcję na Kortowie przez około 10 min wykonywała się predykcja z takim rezultatem.



Rys. 6. Pierwsza klasyfikacja kortowa

Budynki zostały sklasyfikowane jako roślinność, mała część roślinności wysokiej czyli drzew jako budynki jedynie grunt był dobrze sklasyfikowany. Długa predykcja i działanie algorytmu KNN(jeśli mamy duże zbiory danych samo wmii ma ponad 3 miliony punktów stwierdziliśmy ze trudno będzie nam zoptymalizować ilość sąsiadów po których mamy szukać) skłoniło nas do poszukiwania innego rozwiązania. Znaleźliśmy sieć neuronową PointNet która została stworzona do działania na chmurach punktów 3d lecz napotkaliśmy problemy z konfiguracją i instalacją CUDNN na wsl. Ostatecznie wybraliśmy las losowy (Random Forest). Składa się z wielu drzew decyzyjnych, które są trenowane niezależnie i łączone w celu uzyskania bardziej stabilnych i dokładnych predykcji. Tutaj również użyliśmy biblioteki scikit-learn. RandomForsetClassifier z parametrami n_estimators=100 oraz cechami x, y, z, trenowany tak samo na pliku WMII. Dzieki wykorzystaniu lasu losowego trening trwał dłużej lecz zależało nam na czasie predykcji który w tym przypadku skrócił się. Robiąc klasyfikacje na Kortowie znów otrzymaliśmy wynik podobny do tego z KNN. Zaczęliśmy szukać przyczyny, oczywistym powodem była mała liczba cech, dodaliśmy więc na próbę red, green, blue odpowiadające za kolor punktu, lecz wycofaliśmy się z tego pomysłu ponieważ problemem byłby dane z pomiaru wykonanego zimą większa część powierzchni byłaby przykryta śniegiem co mogłoby prowadzić do błędnej klasyfikacji. Kolejny problem wystąpił, gdy zdaliśmy sobie

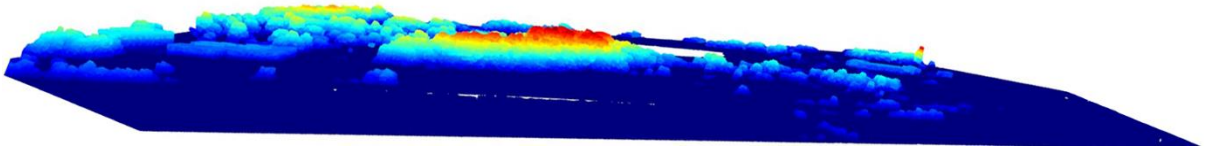
sprawę z tego ze nasza cecha „z” odpowiedzialna za wysokość jest wysokością nad poziomem morza sprawia to problem, trenując w ten sposób model na danym ukształtowaniu terenu, będziemy wykonywać predykcje na terenu z zupełnie różniącym się ukształtowaniem mamy pewność błędnej klasyfikacji chmury punktów. Wyjścia były dwa usunięcie cechy „z” lub próba normalizacji wysokości chmury punktów. Podczas pracy nad powyższymi problemami udało się usunąć szum z chmury punktów, który można było zauważyć na wcześniejszych zdjęciach. Następne wizualizacje będą już z usuniętym szumem.

Podejście z normalizacją wysokości chmury punktów

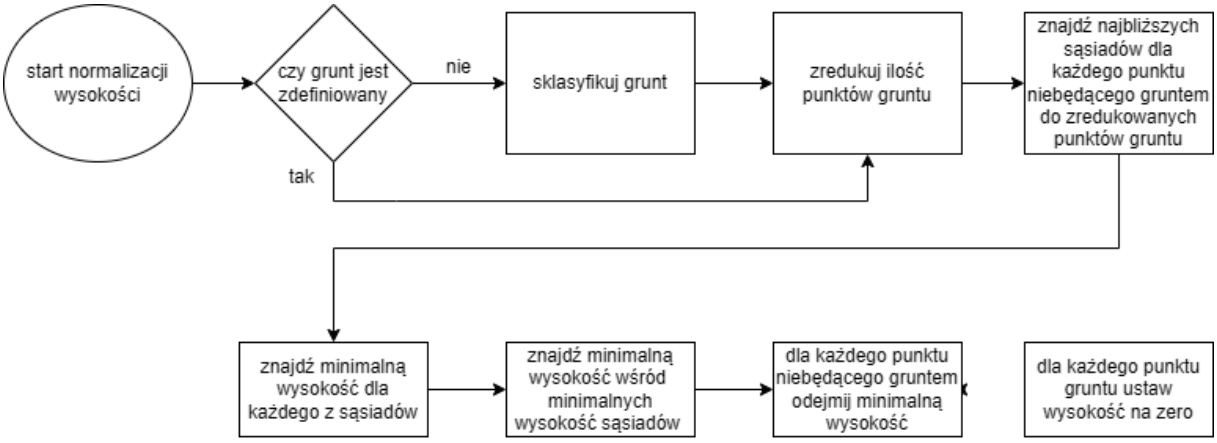
Pierwszym sposobem było obniżanie punktów za pomocą odjęcia wartości średniej lub maksymalnej wysokości dla każdego punktu, te podejście było słabym rozwiązaniem, powodem nieskuteczności rozwiązania jest różne ukształtowanie terenu jeśli nauczymy model na terenie nizinnym gdzie uda nam się w miarę dokładnie obniżyć grunt to będzie działać, ale jeśli teren jest mocno zróżnicowany to może wystąpić spłaszczenie całych obszarów do wartości zero.

Rys. 7 WMII udane przekształcenie

Rys 8. Kortowo złe przekształcenie w większości zostały same korony drzew



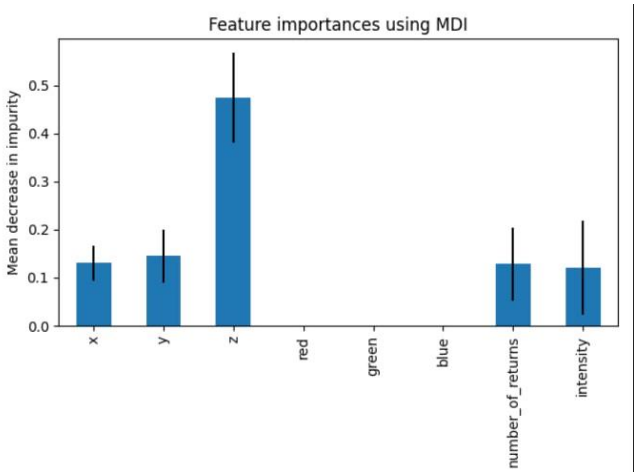
udało się zaimplementować poprawną normalizację gruntu oto schemat blokowy rozwiązania:



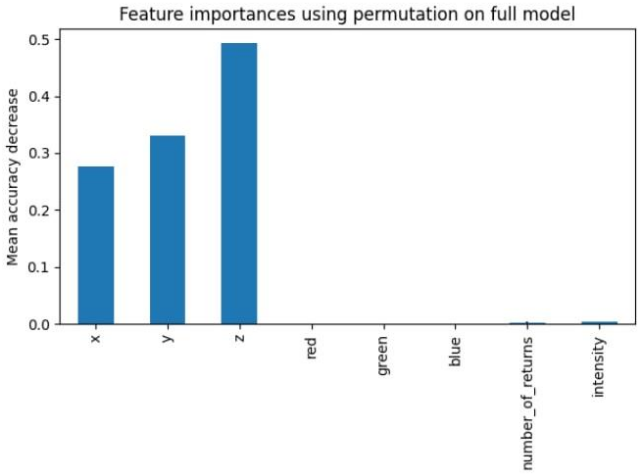
Rys.9 Schemat algorytmu obniżania gruntu

Rys. 10 Wizualizacja znormalizowanej chmury punktów

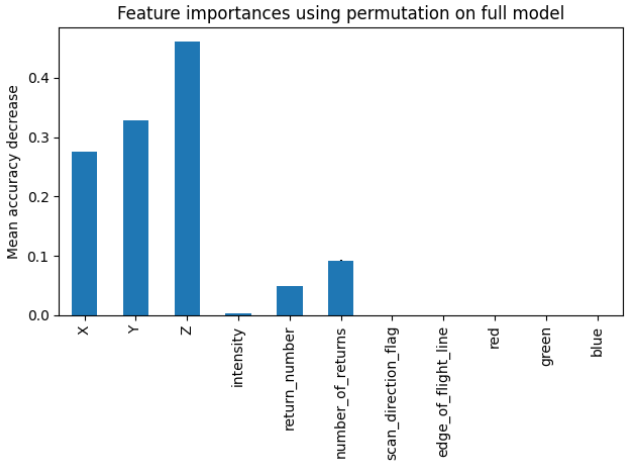
Następnym krokiem było szukanie cech które możemy dodać do modelu robiliśmy to za pomocą atrybuty „„feature_importances_” (MDI) w klasie RandomForestClassifier która odnosi się do istotności cech w modelu lasu losowego oraz za pomocą istotności bazującej na permutacji (MDA) z biblioteki scikit-learn. Różnica między MDA a MDI polega na tym, że MDA oceniacechę na podstawie jej wpływu na dokładność predykcji modelu, podczas gdy MDI ocenia ją na podstawie zmniejszenia nieczystości w węźle drzewa. Postanowiliśmy sprawdzić cechy które już używaliśmy oraz po przeczytaniu dokumentacji formatu LAS wybraliśmy dwie cechy te które mogą poprawić nas model.



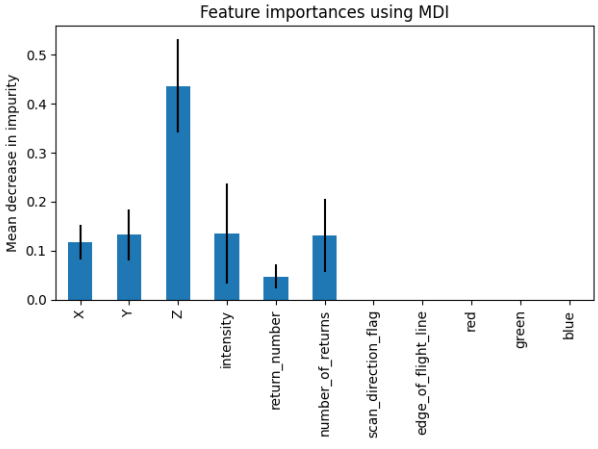
Rys. 11 istotność cech MDI



Rys. 12 istotność cech MDA



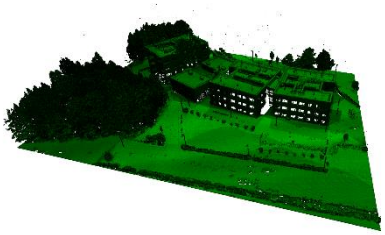
Rys.13 istotność cech MDA



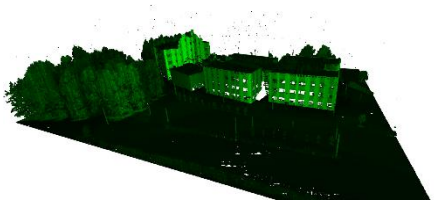
Rys.14 istotność cech MDI

Otrzymane wyniki uświadomiły nas, że wybrane cechy w niskim stopniu wpływają na dokładność modelu. Zmusiło nas to do stworzenia własnych cech, które są następujące Ball_density - to ilość punktów w kuli o współrzędnych punktu dla którego jest obliczane Ball_density i promieniu 0.2, cylinder_density - to ilość punktów w walcu o współrzędnych punktu dla którego jest obliczane cylinder_density, promieniu 0.05 i nieskończonej wysokości, Phi_angles_of_normal_vectors to kąt między rzutem wektora normalnego na powierzchnię

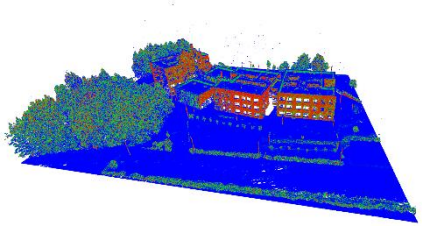
a osią x wzór: φ=arctan(y/x), theta_angles_of_normal_vectors to kąt między wektorem normalnym a osią z wzór: θ=arccos(z/r), Min_height, max_height - to lokalna minimalna i maksymalna wysokość w obrębie danego punktu która jest wyszukiwana w walcu o współrzędnych punktu dla którego jest obliczane min_height lub max_heigh, o promieniu 0.05 i nieskończonej wysokości, Mean_height jest to średnia wysokość obliczana na podstawie ilorazu sumy wysokości punktów w obrębie danego punktu która jest liczona w walcu o współrzędnych punktu dla którego jest obliczane Mean_height, o promieniu 0.05 i nieskończonej wysokości



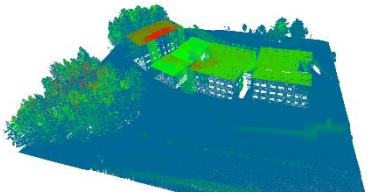
Rys. 15a Wizualizacja ball_density



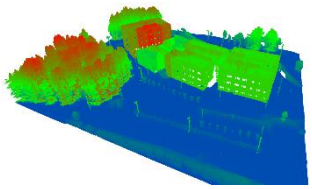
Rys. 15b Wizualizacja cylinder_density



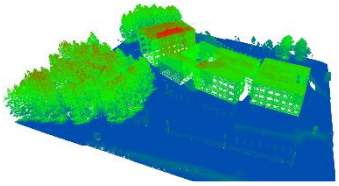
Rys. 15c Wizualizacja kątów wektora normlanego



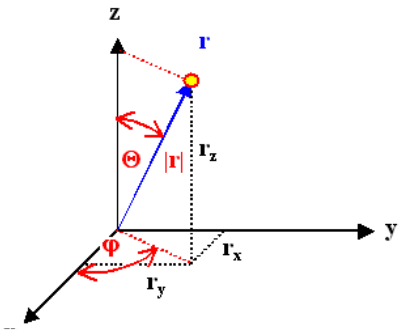
Rys. 15d Wizualizacja min_height



Rys. 15e Wizualizacja max_height

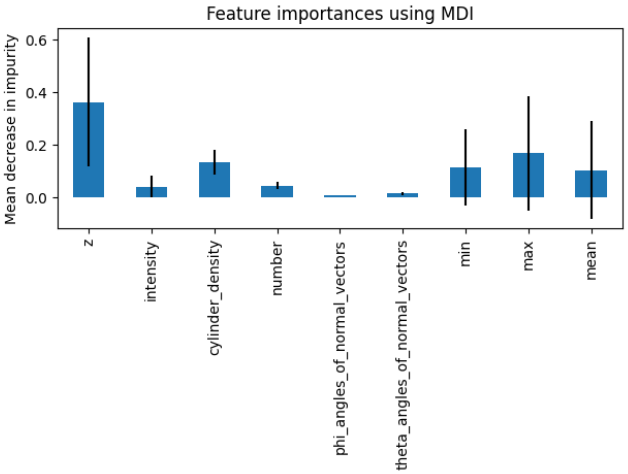


Rys. 15f Wizualizacja mean_height

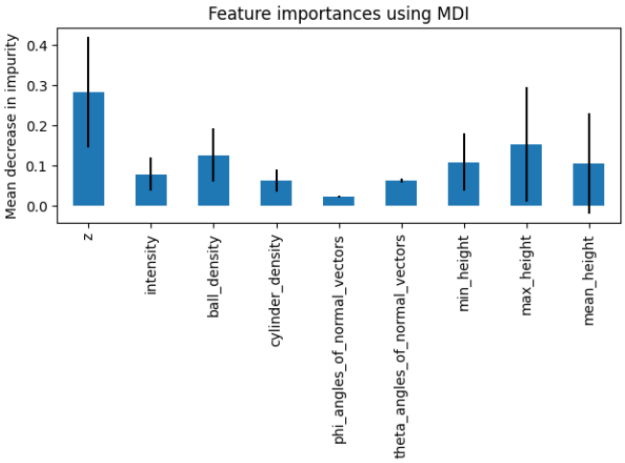


Rys. 15g kąty phi i theta

Postanowilismy również usunąć x i y oraz number of returns nie wprowadzały żadnej różnicy w klasyfikacji, a dzięki temu skróciliśmy czas wykonywania terningu oraz model zajmował mniej miejsca na dysku.



Rys.16. istotność cech własne MDI



Rys.17 istotność cech własne MDI

Następujące kombinacje cech okazały się nie wystarczające dostawaliśmy dokładności modelu rządu 40-50 %. Postanowiliśmy wykonać optymalizacje hiper parametrów za pomocą GridSearchCV sprawdzaliśmy n_estimator, max_depth, oraz max_fatures przy próbach większej ilości parametrów czas wykonywania operacji był zbyt długi by je kontynuować (przez 8 godzin nie została sprawdzona nawet połowa kombinacji, oraz koszty obliczeniowe były zbyt wysokie na nasz sprzęt). Po optymalizacji najlepsze parametry były domyślnymi wartościami których już używaliśmy w naszym modelu. Następna próbą było dodanie cech number_of_returns i return_number oraz usunięcie ball_density, które uniemożliwiało używanie pliku kortowa przez zbyt duże koszty obliczeniowe. Otrzymane wyniki poprawiły się.

Rys. 18 Wizualizacja predykcji

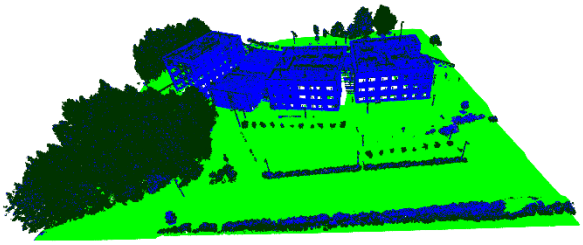
	precision	recall	f1-score	support
1	0.12	0.09	0.11	64354
2	0.94	0.98	0.96	3397356
5	0.94	0.73	0.82	2037147
6	0.60	0.83	0.70	874416
accuracy			0.87	6373273
macro avg	0.65	0.66	0.65	6373273
weighted avg	0.89	0.87	0.87	6373273

0.6591769733339825

Rys. 19 Wyniki predykcji

Dokładność wynosiła teraz 87% dokładność zbalansowana 65%, na dokładność zbalansowaną duży wpływ ma klasa 1, gdzie obiekty te są sklasyfikowane jako niezaklasyfikowany. Klasa ta została w pliku Kortowo gdzie zawiera obiekty które zostały pominięte przy ręcznym

klasyfikowaniu może to być np. samochód, murek. Naszym celem było skupienie się na klasyfikacji gruntu(2), roślinności(5), budynków(6). Następnie sprawdziliśmy model usuwając kąty między osiami a wektorami normlanymi, dodając za to wektory normalne.



Rys. 20 Wizualizacja predykcji

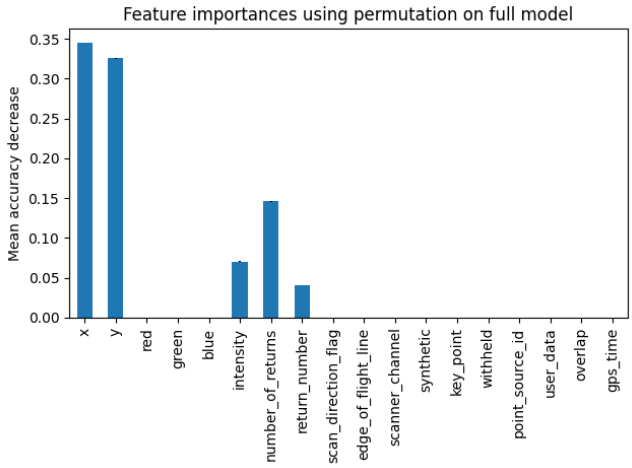
Balanced accuracy: 0.65

	precision	recall	f1-score	support
1	0.11	0.02	0.03	50932
2	0.98	0.94	0.96	3542566
5	0.78	0.93	0.85	1580940
6	0.80	0.73	0.76	1198835
accuracy			0.89	6373273
macro avg	0.67	0.65	0.65	6373273
weighted avg	0.89	0.89	0.89	6373273

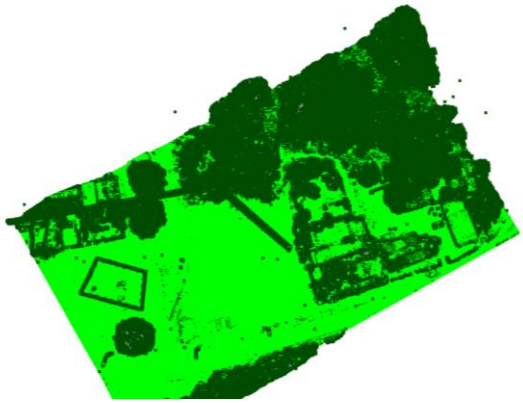
Rys. 21 Wyniki predykcji

Podejście z usunięciem cechy „z”

Po sprawdzeniu istotności cech bez „z” wytrenowaliśmy model z cechami x, y, intensity oraz number of returns, reszta parametrów modelu bez zmian. Model sklasyfikował dobrze grunt resztę wykrył jako roślinność próba została powtórzona z cechą ball_density wyniki były podobne znów pojawił się ten sam problem



Rys. 22 istotność cech MDA

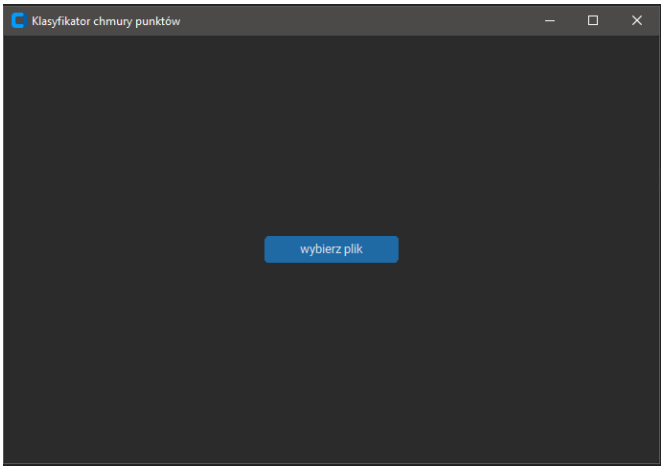


Rys. 23 Wizualizacja próby klasyfikacji bez „z”

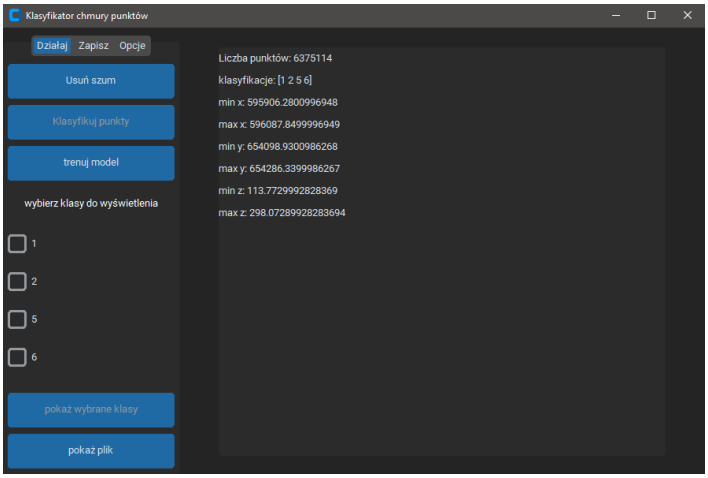
Po tym zdecydowaliśmy się porzucić to podejście. Normalizowanie wysokości wyglądało bardziej obiecująco pod względem poprawy dokładności gdyż „z” odrywa duża rolę w istotności cech.

Tworzenie GUI

Gui zostało wykonane z pomocą biblioteki CustomTkinter. Podczas procesu tworzenia aplikacji napotkaliśmy problem z długim czasem wykonywania niektórych obliczeń, co powodowało, że aplikacja nie odpowiadała przez dłuższy czas po naciśnięciu pewnych przycisków. Aby rozwiązać ten problem, wprowadziliśmy wielowątkowość do poszczególnych przycisków. Ponadto, aby uniknąć przypadkowego uruchamiania wielu wątków jednocześnie i potencjalnego przeciążenia systemu, dodaliśmy mechanizm blokowania wszystkich możliwych przycisków przed rozpoczęciem wykonywania funkcji. Po zakończeniu obliczeń przyciski były ponownie odblokowywane, co zapewniało płynność i stabilność działania aplikacji.



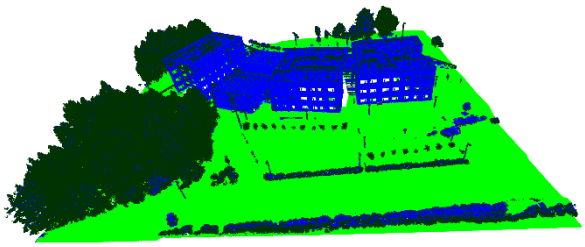
Rys. 24a wygląd GUI



Rys. 24b wygląd GUI

Finałowa wersja modelu

Po wyżej wymienionych próbach i eksperymentach wybraliśmy model lasu losowego z parametrami n_estimators=100, max_depth=5, max_features='sqrt', dzięki max_depth ilość zajmowanego miejsca przez zapisany model w dużym stopniu zmalała z 11 GB do zaledwie 576 KB nie powodując straty dokładności modelu. Ostatecznie cechami które wybraliśmy było: 'z', 'intensity', 'number_of_returns', 'return_number', 'cylinder_density', 'normal_vectors_x', 'normal_vectors_y', 'normal_vectors_z', 'min_height', 'max_height', 'mean_height'. Dokładność modelu wygląda następująco:

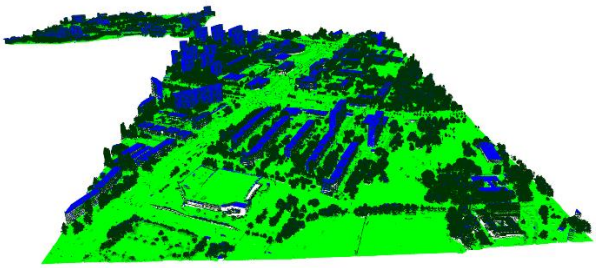


Rys. 25 Wynik dokładności predykcji WMII

Balanced accuracy: 0.65

	precision	recall	f1-score	support
1	0.11	0.02	0.03	50932
2	0.98	0.94	0.96	3542566
5	0.78	0.93	0.85	1580940
6	0.80	0.73	0.76	1198835
accuracy			0.89	6373273
macro avg	0.67	0.65	0.65	6373273
weighted avg	0.89	0.89	0.89	6373273

Rys. 26 Wizualizacja predykcji WMII



Rys. 27 Wynik dokładności predykcji Olsztyn wycinek

	precision	recall	f1-score	support
0	0.07	0.00	0.00	282629
2	0.92	0.97	0.95	8437408
3	0.17	0.07	0.10	165350
4	0.48	0.78	0.59	328303
5	0.80	0.97	0.87	3965492
6	0.86	0.53	0.65	2146530
7	0.00	0.00	0.00	1709
12	0.06	0.02	0.03	423316
accuracy			0.85	15750737
macro avg	0.42	0.42	0.40	15750737
weighted avg	0.83	0.85	0.83	15750737

Rys. 28 Wizualizacja predykcji Olsztyn wycinek

Z geoportalu udało się pobrać wycinek Olsztyna na którym wyniki interesujących nas klas wyglądają następująco grunt 92% roślinność 80% budynki 86%. Podsumowując, nasze założenia dotyczące klasyfikatora oraz zastosowanego modelu przyniosły zadowalające wyniki, spełniając oczekiwania i wymagania postawione na początku projektu.