Hochschule München University of Applied Sciences

Bachelorarbeit

Softwarealternativen zur Bearbeitung von Ausreisserpunkten und automatisierte Ausreisserdetektion mittels überwachter Klassifizierung für Digitale Oberflächenmodelle (DOM)

angefertigt von:
Hubertus Carlos Antonio Vier

Studiengang:
Angewandte Geodäsie und Geoinformatik

Betreuer:
Prof. Dr.-Ing. Sebastian Briechle

Die Arbeit wurde angefertigt in Kooperation mit:
Landesamt für Digitalisierung, Breitband und Vermessung - Referat 85,
Ansprechpartnerin: Isabell Riesinger

Wintersemester 2023/ 2024

München, den 01.03.2024

Kurzfassung

Das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) stellt für ganz Bayern aktuelle digitale Oberflächenmodelle (DOM) zur Verfügung. Bei der Berechnung der Oberflächenmodelle in Form von 1km² Kacheln treten Punktausreißer auf, die ihm Nachhinein manuell korrigiert werden müssen. Das aktuell verwendete Programm zur Korrektur erfüllt nicht mehr die Performance Anforderungen und soll ersetzt werden. Zusätzlich wird nach neuen Methoden gesucht den Arbeitsablauf zu beschleunigen.

In dieser Arbeit wird in Kooperation mit dem LDBV dafür eine Softwarealternative ermittelt. Zusätzlich wird mittels überwachter Klassifizierung/ Random Forest versucht Ausreißer auf Äckern und in Bereichen von Solarparks automatisch zu detektieren, um diese vor der manuellen Bearbeitung beispielsweise kennzeichnen zu können, um den Arbeitsablauf weiter zu beschleunigen.

Um eine geeignete Softwarealternative zu finden, wurden aus den Rahmenbedingungen Recherche-Kriterien ermittelt und entsprechende Softwareprodukte gesucht. Von der recherchierten Software wurde ein kostenloses und ein kostenpflichtiges Programm getestet. Das kostenpflichtige bestand aus der Kombination von Terra Scan und Terra Modeller von Terra Solid und das kostenlose die Freeware CloudCompare. Beide erfüllten dabei die Anforderungen zu Korrektur von Ausreißern.

Für die überwachte Klassifizierung wurden DOM-Kacheln mit Ausreißern zu Erstellung von Trainings- / Validierungsbeispielen und Testkacheln verwendet. Um weitere Merkmale für die Vorhersage zu erhalten, wurden zusätzliche Eigenschaften wie Dichte und Geometrie für verschiedene Radien auf den Referenzdaten berechnet. Die besten 10 Merkmale wurden für das Training des Klassifikators verwendet. Beim Test des Klassifikators konnte jedes mal Ausreißer bestimmt werden. Der F1-Wert für die Detektion von Ackerausreißern lag bei den Tests zwischen 0,46 und 0,99. Der F1-Wert für die Solarparkausreißern lag bei 0,79.

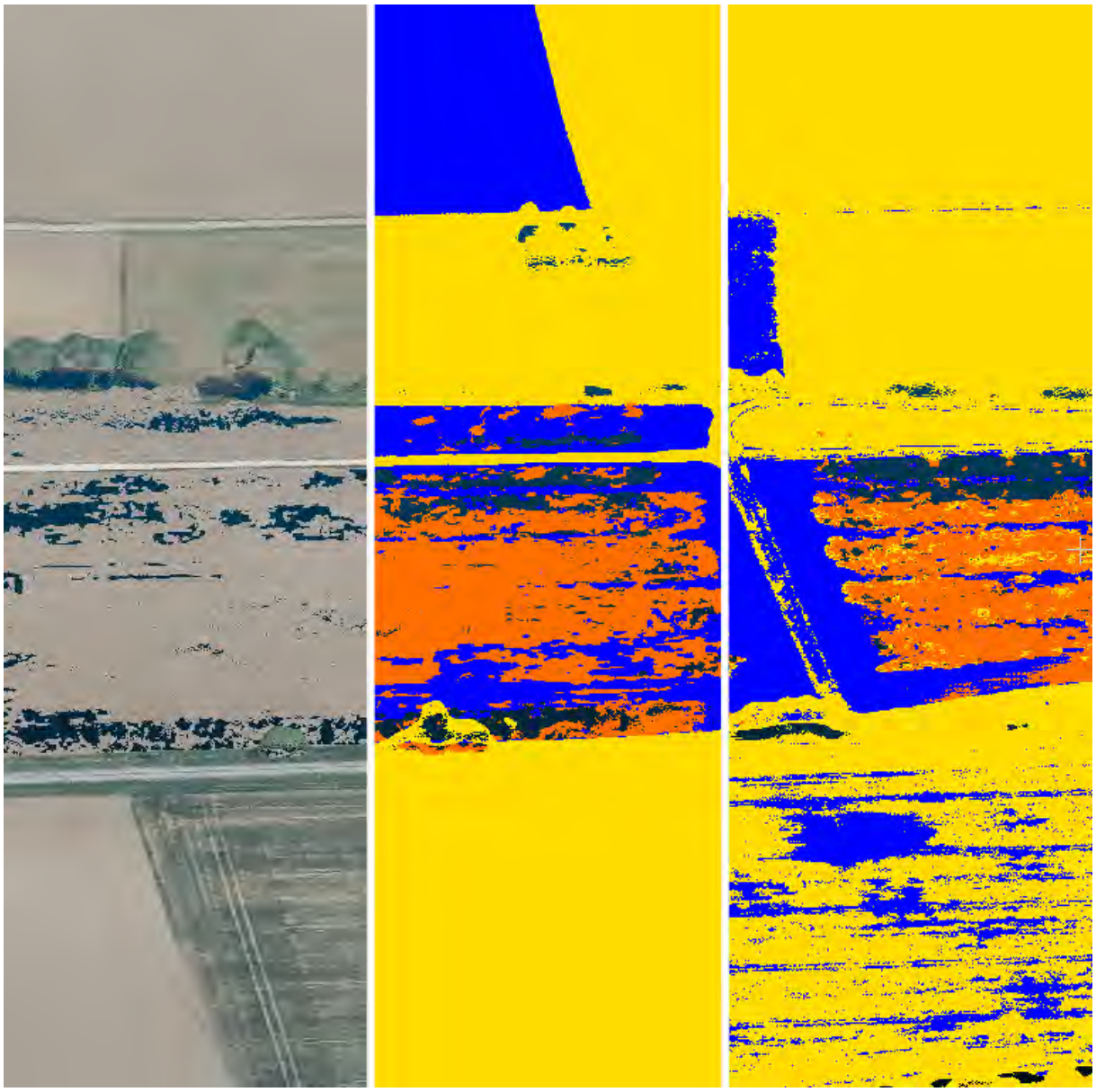

Abbildung 1: Allgemeine Visualisierung von Ackerausreißern in Echtfarbe (links), klassifiziert (Mitte) und mit Klassifikator vorhergesagt(rechts)

Abstract

The Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) provides up-to-date digital surface models (DSM) for the entire state of Bavaria. The calculation of these surface models in the form of 1km² tiles is prone to produce outliers, which must be manually corrected afterwards. The current program used for corrections no longer meets recent performance requirements and is to be replaced. In addition, new methods are being sought to speed up the workflow.

In this project, in cooperation with the LDBV, an alternative software solution is being searched. Additionally, an attempt is being made to automatically detect outliers in empty fields or in areas of solar parks through supervised classification/Random Forest before manual processing, to mark them and thus further accelerate the workflow.

To find a suitable software alternative, research criteria were drawn from the framework conditions and corresponding software products were sought out. Out of the researched software, one free and one payable program were tested. The payable option was the combination of TerraScan and TerraModeller by TerraSolid, and the free software was the freeware CloudCompare. Both fullfilled the requirements for outlier correction.

For the supervised classification, DSM tiles with outliers were used to create training/validation and test tiles. To obtain further features for prediction, properties such as density and geometry for different radius values were calculated on the reference data. The best 10 features were then used to train the classifier. During classifier testing, outliers could be identified each time. The F1 score for the detection of field outliers varied between 0.46 and 0.99 in the tests. The F1 score for solar park outliers was 0.79.

Erklärung

gemäß § 16 Abs. 10 APO in Zusammenhang mit § 35 Abs. 7 RaPO

Semester: Wintersemester 23/24

Betreuer: Prof. Dr.-Ing. Sebastian Briechle

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München 25.2.2024

Ort, Datum Unterschrift

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Erstellung von flächendeckenden digitalen Oberflächen Modellen (DOM) und Orthophotos (DOP) ist eine der Kernaufgaben des Referat 85 am Landesamt für Digitalisierung, Breitband und Vermessung (LDBV). Sie werden aus den Luftbildern der Bayernbefliegung abgeleitet, welche im zwei-Jahre-Turns für ganz Bayern durchgeführt wird, (LDBV, 2024a). Beide Produkte durchlaufen bei der Produktion einen umfassenden Prozess der Qualitätssicherung. Ein Arbeitsschritt im Prozess der Qualitätssicherung ist die Detektion und Behebung von Ausreißerpunkten. Diese entstehen bei der Berechnung von Punktwolken mittels Dichte Bildzuordnung und treten meist an Stellen mit sich wiederholenden Strukturen oder Flächen mit beinahe homogenem Farbverlauf auf (AdV, 2019). Entsprechend oft findet man sie so auf Wasserflächen, Äckern oder großflächigen Solarparks. Die Detektion und Behebung der Fehler ist der zeitaufwändigste Teil im Prozess der Qualitätssicherung. Um diesen zu beschleunigen, müssen aktuelle Softwareprodukte mit eingebunden und weitere Prozesse beschleunigt werden.

Ein Ziel dieser Arbeit ist es, aktuelle Softwarealternativen zur Bearbeitung von Ausreißerpunkten zu recherchieren und zwei Produkte dem QS-Arbeitsablauf entsprechend zu testen. Das andere Ziel dieser Arbeit ist es zu prüfen, ob Ausreißerpunkte mittels einer Methode der überwachten Klassifizierung automatisiert detektiert werden können, um den Arbeitsaufwand weiter zu reduzieren. Dafür wird beispielhaft versucht, Acker- und Solarparkausreißer zu detektieren.

1.1 Erstellung von DOM und DGM

Im Folgendem wird einleitend ein kurzer Überblick über die Erstellung von digitalen Oberflächenmodellen und digitalen digitalen Geländemodellen gegeben. Danach wird der aktuelle Arbeitsablauf bei der Erstellung von DOMs und DOPs zusammengefasst dargestellt.

1.1.1 Aerotriangulation

Durch die Aerotriangulation wird die äußere Orientierungen aller Luftbilder berechnet. Bei der äußeren Orientierung handelt es sich um die Lage (RW, HW, h) sowie die Verdrehung (ω ,ϕ , κ) der Kamera-Projektionszentren im Bezugssystem. Für die Berechnungen werden im Allgemeinen Positionsdaten aus Navigationssystemen (GNSS+IMU) und Passpunkte verwendet.

1.1.2 Digitale Oberflächenmodelle

Durch präzise orientierte Luftbilder kann per dichten Bildzuordnung (Dense Matching) die Oberfläche des Geländes rekonstruiert werden. Ergebnisse der Berechnung können 3D-Punktwolken oder digitale Oberflächenmodelle (DOM) sein. Das digitale Oberflächenmodel beschreibt hierbei die Erdoberfläche mit allen darauf befindlichen Objekten (z.B. Gebäude und Vegetation)(LDBV, 2024c). Die Darstellung erfolgt in Gitterform, wobei jedem Punkt ein Höhenwert zu geordnet wird. Durch die Erweiterung eines zweidimensionalen Gitters mit einem Höhenwert, spricht man bei einem DOM von einer 2,5-D Darstellung.

1.1.3 Digitale Orthophotos

Mit Hilfe der georeferenzierten und orientierten Luftbildern wird eine verzerrungsfreie und maßstabsgetreue orthographische Darstellung der Geländeoberfläche berechnet. Das Produkt der Berechnung nennt man Orthophoto. Bezieht man ein digitales Geländemodell oder eine Punktwolke bei der Berechnung mit ein, werden Gebäudeverkippungen eliminiert und man erhält ein true Orthophoto.

1.2 Qualitätssicherung bei der Erstellung von DOM und DOP am LDBV-Referat 85

Die Qualitätssicherung bei der DOM und DOP-Erstellung am LDBV basiert auf dem Leitfaden zur Qualitätssicherung von True Orthophotos (true DOP), welcher von der Projektgruppe ATKIS-DOP im AdV-Arbeitskreis Geotopographie erstellt wurde (Version 1.0)(AdV, 2019).

Am LDBV-Referat 85 wird die Qualitätssicherung nach folgendem Arbeitsablauf durchgeführt:

1. Berechnung der digitalen Oberflächen Modelle und true Orthophotos
2. Vollständigkeitsprüfung:
   Überprüfung der Kachelanzahl pro Projektfläche, Überprüfung der Punktanzahl pro Kachel, Überprüfung des Infrarot-Kanals, Überprüfung der True DOPs
3. Lage- und Höhenkontrolle:
   Lagekontrolle an Giebelpunkte, Sichtprüfung auf Projektflächengrenzen, Höhenkontrolle über Referenzpunkte und -flächen
4. Korrektur von Gewässerflächen:
   Automatische Klassifikation von Gewässermasken über true DOP, manuelle Korrektur der Gewässermasken, Höhenkorrektur/ Interpolation aller Punkte innerhalb der Gewässermasken auf Höhenniveau am Rand

5. **Ausreißerbearbeitung:**

   Ausreißererkennung über statistischen Test (Perzentile) und Erzeugung von Seitenansichten für fehlerhafte Kacheln, Sichtung der Seitenansichten und Kennzeichnung von tatsächlich fehlerhaften Kacheln, Überprüfung der fehlerhaften Kacheln im Punktwolken-Viewer (Programm A), Korrektur der Ausreißer (Programm B)

6. Prüfung auf Luftbildmängel:

   Sichtung der DOPs und Kennzeichnung von Mängeln (Reflexionen, Wolken und Beleuchtung)

2 Daten

Als Grundlage für den Test der Softwarealternativen sowie der automatisierten Ausreißerdetektion dienen ausgesuchte 2,5D DOM-Kacheln mit Punktausreißern auf Acker- sowie Solarparkflächen. Diese stammen aus den Befliegungsjahren 2022/ 2023 und wurden vom LDBV für diese Arbeit zur Verfügung gestellt.

Die Eigenschaften der zur Verfügung gestellten Kacheln:

Eigenschaften :	
Abgabeformat	LAZ 1.2 (mit Farbwerten: RGBI)
Bodenpixelgröße / Gitterweite	0,2 m (entspricht 6,25 Pkt./m²)
Abgabeeinheit	1-km²-Kacheln
Anzahl abgebildeter Punkte	25.000.000
Auflösung	meistens +-1,5 Meter.
Höhenbezugssystem	DHHN2016
Bezugsellipsoid	GRS80
Geodätisches Datum	ETRS89

Tabelle 2.1: Eigenschaften der übergebenen Beispiel Kacheln (LDBV, 2024b)

Die verwendeten Kacheln befinden sich in der auf mittlerer Höhe Bayerns und liegen im nördlich sowie südlichen Befliegungsraum. Sie liegen in den Landkreisen Donauwörth, Günzbrug, Kelheim, Regensburg, Straubing und Vilshofen an der Donau (siehe Abb.2.1).

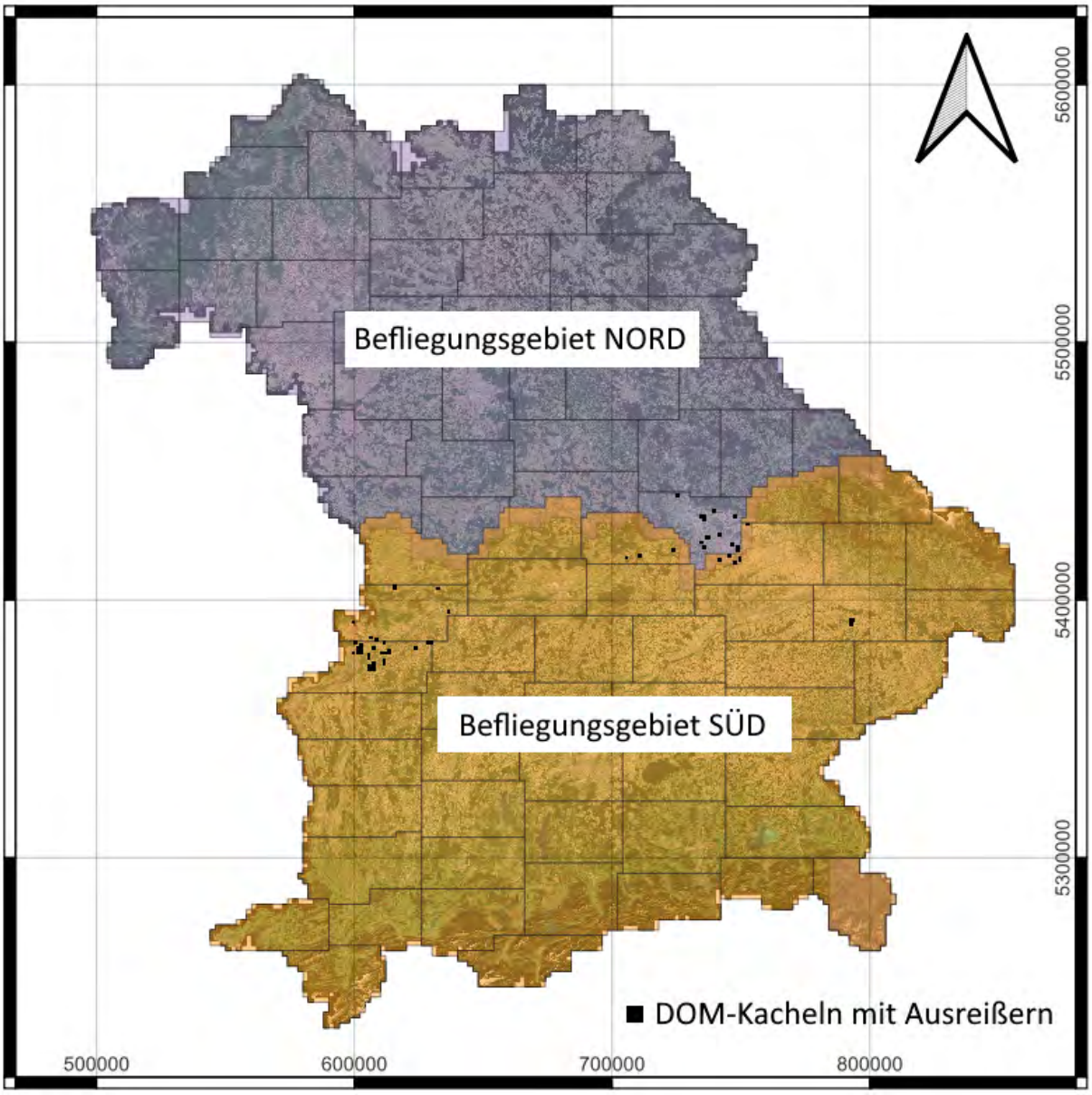

Abbildung 2.1: Lageübersicht über die übergebenen Kacheln mit Ausreißer

3 Software

3.1 Recherche und Test

Bei der Recherche wurden folgende Softwareprodukte zur Bearbeitung von Punktwolken gefunden und die mit *-markierten Produkte getestet:

- Metashape - Agisoft
- Point-Tools - Bentley
- LIMON - DEPHOS Group
- Lidar360 - Green Valley International
- **Terra Modeller, Terra Scan - Terra Solid***
- **Cloud Compare (v2.13 beta)***

3.2 Automatisierte Ausreißerdetektion

Python-Bibliotheken:

- **NumPy (1.24.3):**
  Handhabung von Matrizen und Vektoren
- **pandas (2.0.3):**
  Verarbeitung, Analyse und Darstellung von Daten
- **scikit-learn (1.2.2)**
  Machine Learning
- **matplotlib (3.7.2):**
  Darstellung Ergebnisse
- **imbalanced-learn (0.10.1):**
  Under- und Oversampling unausgegleichener Klassen

3.3 Sonstige

- **QGIS (3.28):**
  Visualisierung von Geodaten

4 Vorgehensweise

4.1 Softwarealternativen

4.1.1 Recherche

Um Kriterien für die Recherche nach alternativen Softwareprodukten zu erhalten, wird zunächst der allgemeine Arbeitsablauf betrachtet, in der die aktuell eingesetzte Software verwendet wird. Dort werden die an die Software gestellten Anforderungen bestimmt und aktuelle Beschränkungen festgestellt. Zusätzlich werden zukünftige Anforderungen mit einbezogen.

4.1.2 Test

Nach der Recherche wird ein kommerzielles, sowie ein kostenlos erhältliches Software-Produkt auf die Kriterien der Software Recherche getestet. Ziel ist es einen Überblick über die vorhanden Funktionen und der Verwendbarkeit im QS-Workflow bei der DOP/DOM-Erstellung zu geben.

4.2 Automatisierte Ausreißerdetektion

Im Folgendem werden die Teilschritte der Datenvorbereitung und die Detektion von Ausreißerpunkten auf Acker- und Solarparkflächen mittels überwachter Klassifizierung genauer beschrieben. Am Schluss erfolgt eine Zusammenfassung des Arbeitsablaufs in einem Flussdiagramm.

4.2.1 Datenvorbereitung

Bei der Datenvorbereitung werden aus den vorhanden Trainingsdaten klassifizierte Beispieldaten für Training, Validierung und den finalen Test auf fremden Daten erzeugt. Die Datengrundlage hierfür bilden DOM-Kacheln, welche Umgebungs-, fehlerfreie Acker- und Solarpark- und Ausreißerpunkte besitzen. Dafür werden Referenzdaten vorbereitet, zusätzliche Features abgeleitet und unausgeglichene Beispielverteilungen behoben.

Referenzdaten

Damit später der Klassifikator trainiert, validiert und getestet werden kann, müssen zunächst Referenzdaten aufbereitet werden. Bei der Aufbereitung werden Beispielpunkte für die jeweiligen Klassen ausgesucht und entsprechend klassifiziert. Die Auswahl der benötigten Klassen muss dabei vor der Klassifizierung erfolgen, da das nachträgliche Ergänzen von Klassen großen Aufwand mit sich bringt. In dieser Arbeit wird in fünf unterschiedliche Punktklassen unterschieden. Dabei handelt es sich um richtige und fehlerhafte Ackerpunkte so wie richtige und falsche Solarparkpunkte. Um eine Unterscheidung zur Umgebung zu gewährleisten, wird auch für diese eine Klasse angelegt. Wichtig ist eine strikte Unterscheidung und eine klare Trennung zwischen den jeweiligen Klassen. Keine Klasse darf Beispiele der anderen Klasse enthalten, da dadurch das Endergebnis verfälscht wird. Zu Beginn der Klassifizierung werden die Daten in einem Punktwolken-Viewer geladen. Um Ausreißerpunkte zu finden, bieten sich zwei Möglichkeiten. Die eine Möglichkeit besteht darin, in die perspektivische Ansicht zu wechseln und nach Löchern in der Oberfläche suchen. Die andere ist die Darstellung der synthetic flag-Eigenschaft, welche meist auf fehlerhafte Punkte oder Bereiche hinweist. Sind die fehlerhaften Bereiche gefunden, kann durch diese vertikale Schnitte gelegt werden, um dort die fehlerhaften Punkte zu klassifizieren. Dabei kann die Schnittbreite variiert werden. Ist das Gelände eben, kann eine große Schnittbreite gesetzt werden, besitzt das Gelände eine Starke Wölbung oder andere Unebenheiten, empfiehlt es sich eine geringere Schnittbreite zu wählen. Besitzt das verwendete Programm keine Möglichkeit vertikale Schnitte zu setzen, jedoch eine Segmentierungsfunktion, kann auch damit klassifiziert werden.

Berechnung zusätzlicher Merkmale

Zusätzlich zu den Farbwerten und den synthetischen Punktkennzeichnung werden weitere Unterscheidungsmerkmale bestimmt. Wichtig ist hierbei, dass die Merkmale mit vollständigen Kacheln und nicht auf bereits segmentierte Bespiele bestimmt werden. Eine vorherige Segmentierung würde die Ergebnisse an Randbereichen verfälschen. Mögliche zusätzliche Merkmale sind hierbei die Berechnung von Rauheit, Krümmung, Dichte, stochastisches Moment, geometrische Merkmale und die Brechnung von Abstandswerten zwischen DOM und DGM. Alle Merkmale werden Radius abhängig bestimmt. Der Radius bestimmt hierbei eine dreidimensionale Auswahl von Punkten innerhalb einer Kugel.
Die Rauheit kann gesamt oder aufgeteilt auf die drei Achsen des lokal vorhanden Koordinatensystems bestimmt werden. Da die in den DOM-Kacheln abgebildeten Landschaft-Strukturen naturgemäß zufällig ausgerichtet sind, wird die Rauheit nur gesamt betrachtet. Die Krümmung kann mit der Berechnung eines Durchschnittswerts, der gaußschen Krümmung und der lokalen Normalenänderungsrate erfolgen. Zur Betrachtung der Dichte kann die Anzahl der Nachbarn sowie die Oberflächen- und Volumendichte berechnet werden. Als geometrische Merkmale können Planarität, Vertikalität, Summe der Eigenvektoren, Oberflächenvariation und weitere abgleitet werden. Da der verwendete Radius Einfluss auf das Ergebnis der zusätzlich bestimmten Eigenschaften hat, werden zur Bestimmung des

besten Radius oder der Ableitung von Radius abhängigen Trends alle Eigenschaften für verschiedene Radien berechnet.
Die Berechnung der Höhendifferenz zwischen DOM und DGM erfolgt in der Punktwolken Software CloudCompare, mittels Cloud to Cloud-Vergleich. Werden vor dem Vergleich keine Normalen berechnet, werden nur absolute Distanzen ausgegeben. Alle Ergebnisse zeigen hierbei nur die Distanzen und haben somit kein Vorzeichen. Werden Normalen mit abgeleitet, können positive sowie negative Differenzen berechnet werden.

Sampling

Zum Erstellung von Trainingsbeispielen (Samples) werden die Beispiele mit zusätzlichen Eigenschaften zunächst extrahiert und in einer Gesamtdatei zusammengefasst. Danach werden Merkmale, die die Entscheidung des Klassifikators verfälschen, entfernt. Dafür werden Beispiele mit NaN-Werten und die Koordinaten der Bespiele weggelassen. Um Ungleichheiten in der Anzahl der vorhandenen Klassen auszugleichen, können die Klassen durch Under- oder Oversampling ausgeglichen werden. Undersampling beschreibt dabei die Reduzierung der Beispielanzahl der jeweiligen Klassen auf die Anzahl der Klasse mit den geringsten Beispielen. Beim Oversampling wird die Anzahl der Beispiele auf die Anzahl der beispielsstärksten Klasse angeglichen. Under- sowie Oversampling haben hierbei einen direkten Einfluss auf den Speicherverbrauch. Beide Methoden müssen so angewendet werden, dass der Trainingsdatensatz stets möglichst viele Beispiele für das spätere Training enthält.

4.2.2 Klassifikation

Überwachte Klassifizierung - Random Forest

Als Klassifizierungsmethode wird der Random Forest Algorithmus verwendet. Der Random Forest Algorithmus ist eine Möglichkeit zur überwachten Klassifizierung von Daten. Der Nutzer legt dabei zu Beginn die zu bestimmenden Klassen und deren Eigenschaften fest. Durch die Vorgabe der Klassen kann der Nutzer direkt bestimmen, welche Klassen vorhergesagt werden sollen. Spätere Vorhersagen können nur auf Datensätzen mit identischen Klassen und Eigenschaften erfolgen.
Für das Training des Klassifikators werden diesem klassifizierte Beispieldaten mit deren Eigenschaften übergeben. Auf Grundlage der übergebenen Daten werden dann Entscheidungsbäume erstellt und bilden somit das Model. Die Anzahl und Struktur der Entscheidungsbäume werden über Hyperparameter festgelegt. Neben der Anzahl der Entscheidungsbäume können beispielsweise die maximale Tiefe oder die maximale Anzahl an Features pro Baum festgelegt werden.
Bei der Vorhersage durchlaufen die Eigenschaften diese Entscheidungsbäume. Die endgültige Vorhersage erfolgt dann über einen Mehrheitsentscheid aus der Menge der Entscheidungen der einzelnen Entscheidungsbäume.

Training

Um die Features zu identifizieren, die am besten für das Training des Klassifikators geeignet sind, wird der Klassifikator auf allen vorhandenen Features trainiert. Nach dem Training wird ein Feature-Ranking basierend auf dem Mean-Decrease-Impurity-Wert (MDI-Wert) eines jeden Features erstellt. Der MDI-Wert zeigt an, wie stark ein Feature zur Verringerung der Entscheidungsunreinheit beiträgt. Je höher der MDI-Wert ist, desto eindeutiger können Entscheidungen getroffen werden. Zusätzlich wird eine Standardabweichung berechnet, die einen Überblick darüber gibt, wie stark die Variation in den jeweiligen Eigenschaften ist. Basierend auf diesem Feature-Ranking werden die besten Eigenschaften für das Training des Klassifikators ausgewählt.

Validierung und Test

Beim Training des Modells werden die Parameter immer wieder angepasst und die Ergebnisse validiert. Ziel hierbei ist es, die besten Hyperparameter für den Algorithmus zufinden. Dafür wird der Trainingsdatensatz in einen Trainings- sowie Validierungsteil aufgeteilt. Der Klassifikator wird auf dem Trainingsteil trainiert und auf den Testteil angewendet. Um eine Aussage darüber zu treffen, wie gut der Klassifikator arbeitet, werden verschiedene Metriken verwendet. Die Grundlage der hier verwendeten Metriken ist eine Konfusionsmatrix. Diese zeigt die Anzahl der korrekt und fehlerhaft klassifizierten Punkte. Aus ihr können sich weitere Aussagen berechnen lassen. Ein Beispiel hierfür ist die Wiedererkennungsrate (Recall). Sie zeigt das Verhältnis zwischen der Anzahl der richtig klassifizierten Punkte und der tatsächlichen Gesamtanzahl in der jeweiligen Klasse. Ein hoher Wert zeigt, dass fast alle in einer Klasse vorhandenen Punkte gefunden worden sind. Eine weitere Möglichkeit ist die Betrachtung des Genauigkeitswerts (Precision). Diese gibt das Verhältnis zwischen den richtig klassifizierten Punkten (true positivs) und den als richtig klassifizierten Punkten an (false positivs). Ein hoher Wert zeigt, dass nur wenige Punkte falsch klassifiziert werden. Kombiniert man beide Werte miteinander, kann man den sogenannten F1-Score berechnen. Mit diesem kann man die Leistung eines Modells bewerten. Ist der F1-Score bei 1, funktioniert das Modell fehlerfrei, ist der Wert bei 0, funktioniert das Modell überhaupt nicht. Durch den Einbezug von Recall und Precision erhält man eine ganzheitliche Bewertung. Ziel ist es das Modell mit dem höchsten F1-Score zu finden.
Ein Ansatz für das Modell-Tuning ist die Kreuzvalidierung (Cross-Validation). Der Datensatz wird dabei in n gleichgroße Teile aufgeteilt, das Modell auf einen Teil traniert und auf den restlichen Teilen validiert. Dieser Ansatz verläuft meist automatisiert unter Einbezug verschiedener Kombinationen von Hyperparametern. Als Ergebnis erhält man verschiedene Gesamtleistungen, die dann miteinander verglichen werden. Die Parameter Kombination mit der besten Gesamtleistung wird dann für das Training des finalen Klassifikators verwendet.
Nachdem die besten Features und die besten Hyperparameter bestimmt worden sind, muss das Modell auf einen neuen Datensatz getestet werden. Ziel ist es die tatsächlich Leistung

des Modells zu bestimmen. Anders zu den Validierungsdaten, hat das Modell diese Daten noch nicht während des Trainings gesehen. Das Ergebnis kann dabei zeigen, wie gut ein Modell funktioniert und ob eine sogenannte Überanpassung vorliegt oder nicht. Bei einer Überanpassung hat das Modell die Trainingsdaten auswendig gelernt und generalisiert damit schlecht auf neuen Daten.

4.2.3 Zusammenfassung Arbeitsablauf

Die zuvor beschriebene Vorgehensweise wird in Abbildung 4.1 zusammengefasst dargestellt.

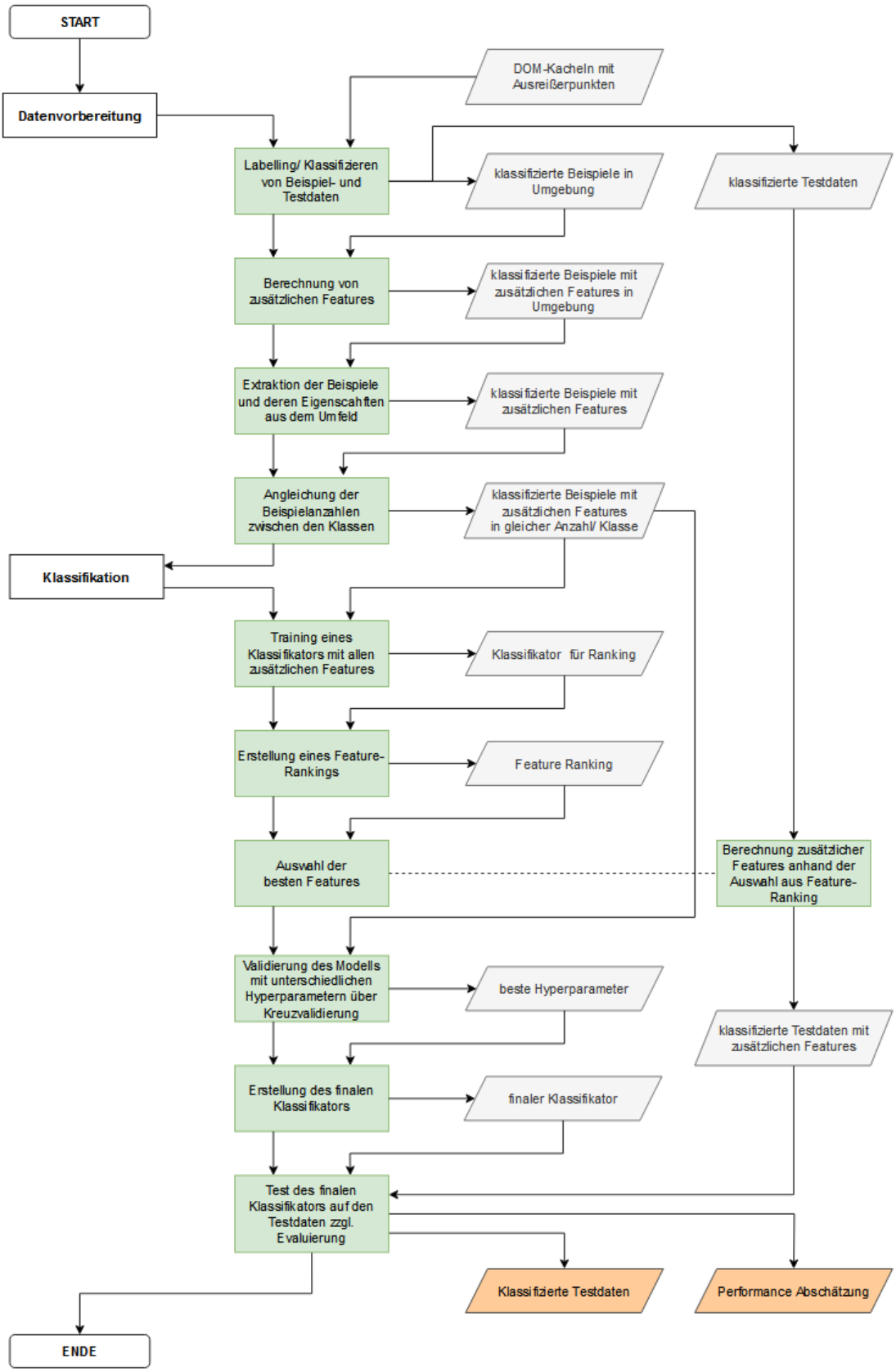

Abbildung 4.1: Gesamtübersicht über den Arbeitsablauf der Ausreißerdetektion

5 Experimente und Ergebnisse

5.1 Softwarealternativen

5.1.1 Kriterien für die Software-Recherche

Zu Beginn der Software-Recherche wurde der in Kapitel 1.2 vorgestellte QS-Ablauf am LDBV einmal im Ganzen mit Schwerpunkt auf der Ausreißerbearbeitung durchgeführt, um die Rahmenbedingungen der zu ersetzenden Software zu ermitteln. Aus den Rahmenbedingungen werden dann, die Kriterien für die Recherche abgeleitet. Die zu ersetzende Software ist DTMaster vom Entwickler INPHO (Trimble). Der QS-Workflow erfolgt immer für ein Befliegungslos. Jedes Befliegungslos besteht aus 1km²-DOM-Kacheln.

Die Ausreißerbearbeitung erfolgt im fünften Schritt des Arbeitsablaufs. Zur Detektion von Kacheln mit Punktausreißern werden alle Kacheln mit statistischen Tests über einen festgelegten Perzentilbereich überprüft. Dabei wird die Summe aller Punkte außerhalb des Perzentilbereichs je Kachel ermittelt und ab einem Schwellwert als Kachel mit Ausreißern gekennzeichnet. Da Ausreißer in Lage immer richtig liegen, aber in der Höhe falsch, wird zur schnelleren Beurteilung je betroffener Kachel eine Seitenansicht generiert. Diese erlaubt einen ersten Überblick, ohne dabei die Kacheln in eine Punktwolkensoftware importieren zu müssen. Dieser Schritt dient der Reduzierung des Arbeitsaufwandes, da z. B. Hochspannungsleitungen oder Masten als Ausreißer den Test nicht erfüllen, aber nicht als zu korrigierende Ausreißer betrachtet werden. Die entsprechenden Kacheln werden dann als fehlerfrei gekennzeichnet und die Auswahl an vermeintlich fehlerhaften Kacheln für die weitere Bearbeitung entsprechend reduziert. Sind Ausreißer in der Seitenansicht klar oder nicht eindeutig zu erkennen, werden die betroffenen Kacheln in die Software CloudCompare (Programm A) geladen.

In CloudCompare werden die Punktwolken optisch in Farbe oder durch Darstellung der synthetischen Punkteigenschaft auf Unregelmäßigkeiten geprüft. Bei der optischen Kontrolle fallen meist Löcher oder starkes Rauschen in der Punktwolke auf. Die Punkte sind dann synthetisch, wenn an den entsprechenden Stellen keine Punkte berechnet werden konnten und deswegen interpoliert werden mussten. Damit können synthetische Punkte zusätzlich auf fehlerhafte Bereiche hindeuten. Wird in CloudCompare festgestellt, dass eine Kachel tatsächlich Ausreißer beinhaltet, wird sie zur Korrektur in DTMaster (Programm B) geladen.

In DTMaster werden die Ausreißer zunächst ausgewählt und einer anderen Klasse zugewiesen, um diese für den späteren Verlauf als synthetisch zu kennzeichnen. Die Software DTMaster bietet hierbei keine Möglichkeit der direkten Veränderung dieser Eigenschaft.

Die Eigenschaft dient der Dokumentation, welche Punkte nicht berechnet oder bearbeitet worden sind. Nach der Klassifizierung werden die Punkte wieder ausgewählt und per “Reinterpolate Selected Points“-Funktion auf Umgebungshöhe korrigiert/ interpoliert. Die Auswahl erfolgt hierbei über definierbare Schnitt-Ansichten. Nachdem die Kacheln mit Punktausreißern bearbeitet worden sind, erfolgt nach dem Export die Umwandlung der klassifizierten Ausreißerpunkte in normale Punkte mit der Kennzeichnung als synthetisch. Die Umwandlung erfolgt per Batch-File, welches auf die Funktionen der Software LAStools zugreift und für alle bearbeiteten Kacheln in einem durchgeführt wird.

Bei der Ausreißerbearbeitung mit DTMaster ist aufgefallen, dass die Software lange Import- und Exportzeiten hat. Des weiteren ergaben sich auch bei der Viewer-Funktion lange Wartezeiten bei der Nutzung. Unteranderem konnten zeitweilig nur alle Punkte in einer bestimmten Zoom-Stufe betrachtet werden, während sie in allen restlichen nur zum Teil dargestellt wurden.

Zusätzlich zu den beschriebenen Rahmenbedingungen, ergeben sich in Hinblick auf die Weiterentwicklung des Produktportfolios und der Beschleunigung des Arbeitsablaufs weitere Kriterien. So ist in Zukunft das Angebot von Vermaschungen geplant. Deswegen sollte die Alternative Vermaschungen erzeugen und bearbeiten können. Die Möglichkeit einer Bearbeitungswarteschlange (Queue) könnte den Arbeitsablauf weiter beschleunigen.

Betrachtet man die gerade beschriebenen Rahmenbedingungen ergeben sich folgende Recherche-Kriterien:

- Möglichkeit der Queue-Bearbeitung
- Software sollte performant sein
- Bearbeitung von großen Datenmengen (25 Millionenpunkte)
- Geeignete Korrektur- und Interpolationsmethoden (Höhe)*
- Möglichkeit der Klassifizierung
- Import/ Export von aktuellen Datenformaten (LAZ 1.0-1.4, TIFF)
- Vollständige Visualisierung der in den Daten abgespeicherten Eigenschaften (synthetische Punkte abbildbar)
- Möglichkeit der Erstellung und Bearbeitung von Meshs
  (Folgeprodukt ist noch in Entwicklung)
- Informationen über Kosten zur finanziellen Abschätzung

*Nach der ursprünglichen Berechnung der Oberflächenmodelle mit der Software “SURE“von Esri, durchlaufen die Modelle zwei weitere Male die Berechnungssoftware. Das erste Mal nach der Übergabe der Gewässerpolygone und ein zweites Mal nach der restlichen Ausreißerbearbeitung. Die weiteren Durchläufe lassen sich damit begründen, dass eine einheitliche Rasterbreite und eine korrekte Farbgebung gewährleistet werden sollen.

So werden alle Raster erneut, auf Grundlage der übergebenen Raster und Orthophotos, berechnet. Bei diesem Schritt werden die Oberflächen erneut berechnet und damit auch interpoliert.
Es wurde festgestellt, dass Ausreißerpunkte somit einfach gelöscht werden können. Entstandene Löcher werden bei der erneuten Berechnung geschlossen und automatisch als synthetisch gekennzeichnet. Damit müssen Ausreißer nicht zwingend klassifiziert werden und auch der Schritt der Umwandlung der Klasse zur Eigenschaft fällt damit weg.

5.1.2 Rechercheergebnisse

Bei der Recherche nach den herausgearbeiteten Kriterien wurden folgende Software Produkte gefunden:

Software-Name:	**Entwickler:**	**Herkunftsland**
Metashape Professional Edition	Agisoft	Russland
Point-Tools	Bentley	USA
LIMON UAV	DEPHOS Group	Polen
TerraScan/TerraModeler	Terra Solid	Finnland
Lidar360	GreenValley Int.	USA
CloudCompare	CloudCompare	Frankreich

Tabelle 5.1: Übersicht der Software und Entwickler

Im Folgendem werden die in der Recherche ermittelten Software Produkte mit einer Funktionsumfangstabelle beschrieben. Die Beschreibung erfolgt nach den online verfügbaren Informationen und Dokumentationen der jeweiligen Software.

5.1.3 Funktionsumfangsdiagramm

Tabelle 5.2: Funktionsumfangsdiagramm der recherchierten Softwareprodukte

Software:	Metashape Professional Edition	Point-Tools	LIMON UAV	Terra Scan/ Terra Modeller	Lidar 360	CloudCompare
Preis:	3499€ einmalig	nicht mehr verfügbar	2899€ mit opt. Update Abonnment	TS: 1620 €/Jahr, TM 1080 €/Jahr	1536€ einmalig	kostenlos
Queue Bearbeitung:	nein	keine Angabe	keine Angabe	nein	ja	ja
Limitierung bei Datenmenge:	nein	keine Angabe	keine Angabe	nein	-	nein
Klassifizierung möglich:	ja	ja	ja	ja	ja	ja
Punktwolkenkorrektur möglich:	nur löschen	ja	ja	ja	ja	ja
Interpolation möglich:	nein	keine Angabe	keine Angabe	ja	ja	ja
LAS Import/ Export:	ja	ja	ja	ja	ja	ja
Visualisierung zusätzlicher Eigenschaften:	nur Klassen	nur Klassen	nur Klassen	ja - Darstellung in eigenen Punkt -Layern	ja	ja
Mesh-Erstellung und Bearbeitung:	ja	keine Angabe	ja	ja	ja	ja

5.1.4 Softwaretest

Für den Softwaretest wurden mit TerraScan + Modeler ein kostenpflichtiges und mit CloudCompare ein kostenloses Softwareprodukt ausgewählt. Beide Programme werden im Folgenden getestet. Der Test beinhaltet den Import sowie Export von LAZ-Datein (LAZ 1.2), die Möglichkeit der Punktwolkenbearbeitung, die vorhandenen Klassifizierungsmöglichkeiten, die Visualisierung von Punkteigenschaften, sowie das Erstellen und Bearbeiten von Meshs.

TerraScan (v023.027) und TerraModeler(v023.13)

Die Kombination von Terrasolid Scan und Modeler ist ursprünglich zur Bearbeitung von Laserscan-Daten gedacht, kann aber auch zur Verarbeitung von Punktwolken aus bildbasierten Verfahren verwendet werden. Beide Software Produkte benötigen als Grundlage ein CAD-Programm. Zur Auswahl stehen “Spatix“oder “Bentley“. Der Test wurde mit Spatix durchgeführt. Spatix bietet die Möglichkeit verschiedene Ansichtsfenster zu erstellen, die dann von der Terrasolid Software direkt benutzt werden kann. Zur Bearbeitung von Punktwolken wird Software TerraScan benutzt. Mesh-Produkte können mit TerraModeler und TerraScan erzeugt aber nur mit TerraModeler bearbeitet werden.

Datenimport und -export:
Nach der Auswahl des Beispieldatensatzes erscheint ein Import-Dialog Fenster (Abb.5.1), Terra Scan bietet dabei die Möglichkeit den Punktwolkenursprung anzugeben (Laserscanning/ Foto). Wählt man als Ursprung Foto aus, so kann auch die Software-Quelle ausgewählt werden, mit der die Punktwolke berechnet worden ist. Zur Auswahl steht hierbei “Metashape“, “Pix4D“und die Option “Andere“. Wählt man als Ursprung Laser-Scanning, kann zwischen verschiedenen Aufnahmeplatformen als Quelle ausgewählt werden. Die Software erkennt das Dateiformat (LAZ 1.2) und das Bezugssystem automatisch. Das Bezugssystem kann danach manuell umgestellt oder durch eine Transformation in ein anderes gewechselt werden. Beim Import stehen zudem drei verschiedene Filter-Funktionen zur Verfügung. Die erste Option beschreibt einen Import über “Sampling by Step“, bei der nur jeder n-te Punkt importiert wird. Bei der zweiten Funktion kann eine Auswahl getroffen werden, welche Klassen importiert werden sollen. Die dritte Option bietet die Möglichkeit nur innerhalb eines ausgewählten Bereichs Punkte zu importieren. Neben der Filterung der Punkte können die zu importierenden Punkteigenschaften, dem Format entsprechend, ausgewählt werden. Die “Synthetic Flag“-Eigenschaft wird hierbei nicht aufgeführt. Nach dem Import wird die Eigenschaft in einem temporären Klassenlayer dargestellt.
Der Export bietet die Möglichkeit, die Daten in allen bisher verfügbaren LAS/ LAZ-Versionen auszugeben. Eine genauere Auswahl des Punktformats ist hierbei aber nicht möglich. Laut Dokumentation sind alle Punktformate, der LAS/LAZ-Version entsprechend verfügbar und werden, vermutlich je nach Struktur der Eingangsdaten, automatisch erkannt und ausgewählt. Neben dem Dateiformat kann eine Auswahl über die zu exportierenden Klassen getroffen werden. Optional kann eine Begrenzung (Fence) angegeben werden,

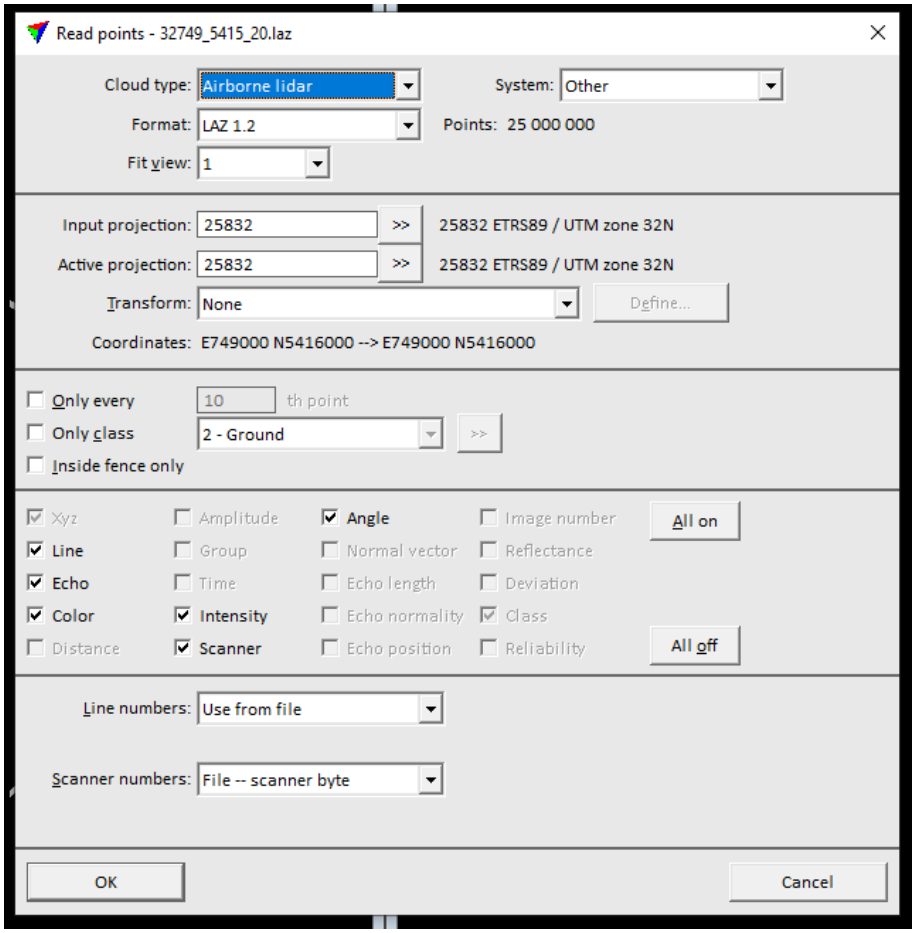

Abbildung 5.1: Import Dialog in Terra Scan

wobei nur die innenliegenden Punkte ausgegeben werden. Will man die Punktwolke in einem anderen Bezugssystem ausgeben, steht auch hier eine Transformations-Option zur Verfügung.

Bearbeitung von großen Datenmengen:
Bei der Bearbeitung von Beispieldaten konnten mehrere Kacheln mit je 25.000.000 Punkten importiert werden. Die jeweilige Systemleistung ist hier der begrenzende Faktor.

Möglichkeit der Queue-Bearbeitung:
Es gibt keine direkte Möglichkeit der Queue-Bearbeitung. Es können mehrere Kacheln ausgewählt und mit individueller Nummer importiert und exportiert werden. Für die Ausgabe muss man sich aber die Nummer und die entsprechende Kachelbenennung sichern, bei der Ausgabe entsprechend auswählen und den Dateinamen wieder eingeben.

Korrektur- und Interpolationsmöglichkeiten:
Zur Korrektur von fehlerhaften Punkten oder Bereichen gibt es zwei Möglichkeiten. Die erste Möglichkeit ist das Verschieben/ Rotieren von Punkten über eine Auswahl. Die

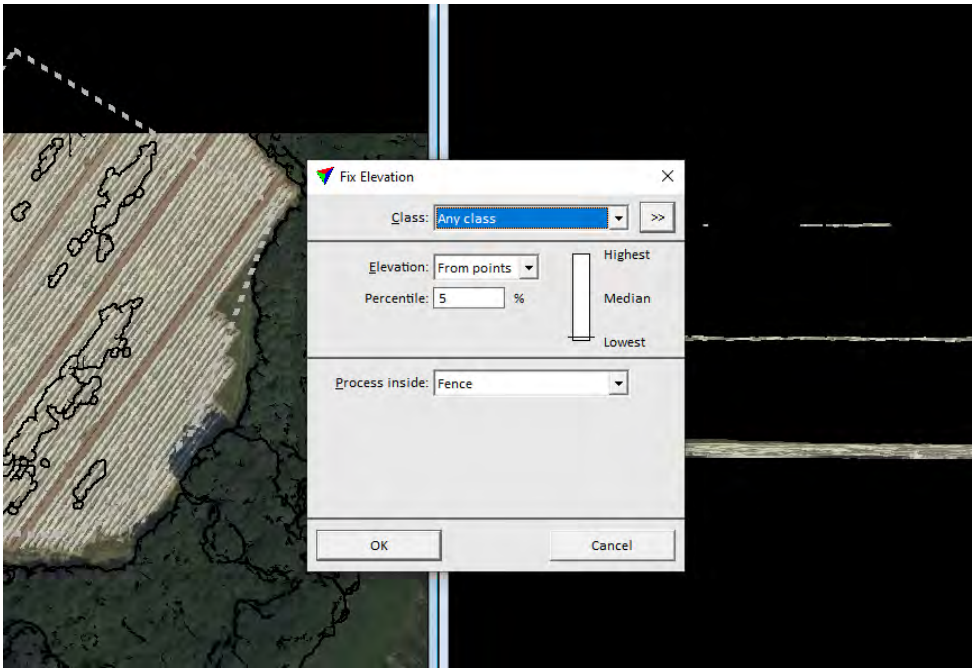

Abbildung 5.2: Fix Elevation Funktion in Terra Scan

zweite Möglichkeit ist die Höhenkorrektur von Ausreißerpunkten über die Funktion “Fix Elevation“(Abb5.2). Bei diesen wählt man in der Grundrissansicht den fehlerhaften Bereich und seine direkte Umgebung aus. Dann wählt man, welche Klasse in der Höhe verschoben werden soll. Danach muss eine Höhenreferenz angeben werden. Dazu kann eine Referenzfläche in Form einer Dreiecksvermaschung oder die Verteilung der in der Auswahl befindlichen Punkthöhen angegeben werden. Bei der Bearbeitung des Beispieldatensatzes werden die Punkthöhen beispielhaft als Referenz ausgewählt. Da man bei allen Punkthöhen eine große Spanne an vorhandenen Höhen hat, wählt man, auf die Häufigkeitsverteilung bezogen, den passenden Perzentilwert aus. Als optische Referenz zeigt eine Anzeige die Position des Perzentilwerts in der Höhe. Wählt man 50% so erhält man den Höhen-Median aller in der Auswahl befindlichen Punkte, welche dann auf alle Punkte in der Auswahl angewendet wird.

Möglichkeiten der Klassifizierung:
Zur Klassifizierung von Punkten können Schnitte mit variabler oder zuvor festgelegter Breite angewendet werden. Als Klassifizierungswerkzeuge (Abb.5.3) stehen hier Pinsel, Auswahl über Polygon, Auswahl überhalb einer Line, Auswahl unterhalb einer Linie oder die Auswahl zwischen zwei Linien zu Verfügung. Die Auswahl unter und oberhalb einer Linie waren im Test die schnellsten Werkzeuge zur Klassifizierung von Ausreißern auf Acker- oder Solarparkflächen. Nach der Klassifizierung der Ausreißerpunkte ist aufgefallen, dass es nicht möglich ist, die in dem temporären Layer als Punkte dargestellte “Synthetic-Flag“-Eigenschaft beizubehalten. Klassifiziert man synthetische Punkte, so werden die in der entsprechenden Klasse angezeigt, verlieren dabei aber ihre Eigenschaft.

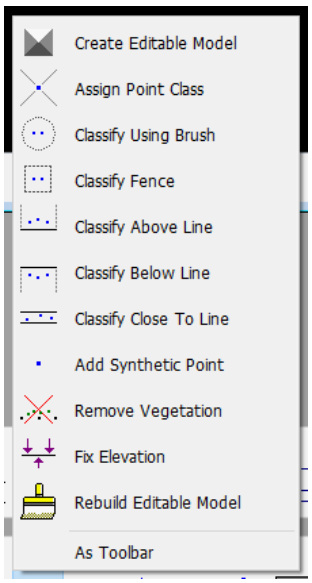

Abbildung 5.3: Klassifizierungswerkzeuge in Terra Scan

Vollständige Visualisierung der in den Daten abgespeicherten Eigenschaften:
Es können alle im LAZ/LAS-Format standardmäßig vorhandenen Eigenschaften in verschiedenen Farbverläufen oder ausgewählten Farben dargestellt werden. Desweitern können einzelne Klassen ein- und ausgeblendet werden. Synthetische Punkte liegen, wie schon beschrieben, in einem temporären Layer und werden nicht als Eigenschaft dargestellt. Färbt man die Punktwolke nach Klassen ein, können sie durch die unterschiedliche Einfärbung detektiert werden.

Möglichkeit der Mesh-Erzeugung und -Bearbeitung:
Sowohl mit Terra Scan als auch Terra Modeller können Dreiecksvermaschungen erzeugt werden. Während bei Terra Scan nach dem Import der LAZ-Datei direkt eine Vermaschung erzeugt werden kann, müssen die Daten für den Import in Terra Modeller zunächst in eine XYZ-Datei (Text- oder Binärformat) umgewandelt werden. In Terra Modeller können auch bestehenden Vermaschungen importiert werden. Importierbare Datenformate sind hierbei: Lattice (Terra Solid), InfraModel, LandXML und TIN.
Bei der Erstellung eines Meshs müssen zu Beginn die miteinzubeziehenden Klassen ausgewählt werden. Danach können weitere fünf Parameter für die Mesh-Berechnung festgelegt werden. Der erste Parameter beschreibt den Umgang mit den Außenkanten der Kachel und muss immer angegeben werden. Alle weiteren Parameter sind optional. Der zweite Parameter legt die maximale Dreieckslänge fest. Der dritte Parameter bestimmt den minimalen Abstand zwischen Punkten, die bei der Vermaschung mit einbezogen werden. Der vierte Parameter bietet einen Abstandswert, der bestimmt, in welchem Intervall entlang einer Bruchkante Vermaschungen erstellt werden soll. Der fünfte Parameter bietet die optionale Filterung von Ausreißerpunkten die bei der Erstellung ignoriert werden können (TerraSolid, 2023a).
Zur Beabreitung von Meshs gibt es die “Edit Area Toolbox“. Mit dieser können Berei-

che geplättet, angehoben oder abgesenkt, Vermaschungen gelöscht und Löcher in der Vermaschung geschlossen werden. Beim Test auf eine ganze Kachel ergaben sich hierbei starke Performance Probleme. Das Verschieben der Ansicht oder die Bearbeitung der Vermaschung verursachten sehr lange Wartezeiten ohne dabei das System auszulasten.
Zur besseren Visualisierung der Vermaschungen können diese nach Höhe oder Steigung, mit zuvor definierten Farbverläufen, eingefärbt werden (TerraSolid, 2023b).
Erzeugte oder bearbeitete Vermaschungen können in folgenden Formaten exportiert werden: Lattice, InfraModel, LandXML, TIN

CloudCompare (v2.13 beta)

Datenimport und -export:
Beim Import des Beispieldatensatzes traten keine Fehler auf. Die Datei in der LAS 1.2 Version und dem Punktformat 2 konnte problemlos eingelesen werden.
Der Export bietet die Möglichkeit LAZ-Dateien in allen bisher verfügbaren LAS/ LAZ-Version und den dazugehörigen Punktformaten (PF 0-10) abzuspeichern. Zudem kann die Anzahl der Nachkommastellen festgelegt werden, was direkten Einfluss auf die Dateigröße hat, da mehr Infromationen pro Punkt abgespeichert werden müssen. Zusätzliche Eigenschaften wie Koordinatensystem oder berechnete geometrische Features, welche nicht von den Standardeigenschaften abgedeckt werden, können in den Versionen 1.0-1.2 in "Variable length records (VLR)"und in den Versionen 1.3-1.4 in "Extended variable length records (EVLR)"abgespeichert werden. Eine Wahl des Referenzsystems oder eine damit einhergehende Transformation ist weder beim Import noch beim Export möglich.

Bearbeitung von großen Datenmengen:
Bei der Bearbeitung von Beispieldaten begrenzte der verfügbare Arbeitsspeicher die maximale Anzahl an zubearbeitenden Kacheln. Trotz umfangreichen Datenmengen konnten alle Funktionen ohne Verzögerung benutzt werden.

Möglichkeit der Queue-Bearbeitung:
Es gibt keine direkte Möglichkeit der Queue-Bearbeitung, jedoch können mehrere Datensätze gleichzeitig importiert und nacheinander bearbeitet werden.

Korrektur- und Interpolationsmöglichkeiten:
Zur Korrektur von fehlerhaften Punkten oder Bereichen gibt es zwei Möglichkeiten. Die erste Möglichkeit ist das Segmentieren und Verschieben/ Rotieren von Punkten (siehe Abbildung 5.4). Die zweite Möglichkeit ist das löschen von Punkten und die erneute Berechnung eines Rasters über die "Rasterize-Funktion".

Möglichkeiten der Klassifizierung:
Einzele Punkte können nach vorangegangener Segmentierung klassifiziert werden. Segmentierte Bereiche können dafür einzeln ausgewählt werden und dann mit Edit/ Scalar fields/ Arithmetic über ein Eingabedialog klassifiziert werden. Dabei wird die Klassifizierung des gesamten Segments bearbeitet (siehe Abbildung 5.5).

Abbildung 5.4: Verschiebung und Rotation von segementierten Punkten

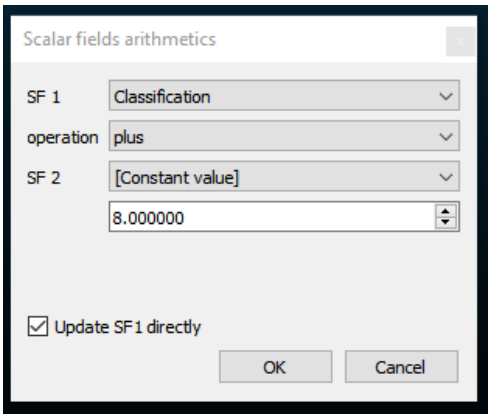

Abbildung 5.5: Klassifizierung in CloudCompare

Vollständige Visualisierung der in den Daten abgespeicherten Eigenschaften:
Es können alle Eigenschaften dargestellt werden. Das Programm bietet die Möglichkeit jede Eigenschaft in Farbe darzustellen und dabei sogar Farbverläufe oder Abstufung festzulegen. So können Standard LAS-Eigenschaften wie “Synthetic Flag“(Abbildung 5.6 oder berechnete geometrische Eigenschaften (Abb.5.22) dargestellt werden.

Möglichkeit der Mesh-Erzeugung und -Bearbeitung:
Bei der Mesherstellung wird zwischen Oberflächen und dreidimensionalen Strukturen unterschieden. Für die Erstellung von Vermaschungen für 3D-Strukturen wird das “Poisson Surface Reconstruction“-Plugin (CloudCompare, 2024) verwendet. Oberflächen können mit einer integrierten Delauny 2,5D Vermaschungsfunktion erzeugt werden.
Zur Bearbeitung von Meshs gibt es lediglich die Option, diese zu glätten. Die Navigation durch das Mesh erfolgte ohne merkliche Ladezeiten.
Zur besseren Visualisierung der Vermaschungen können diese genauso wie die Punktwolke eingefärbt werden. Erzeugte oder bearbeitete Vermaschungen können in folgenden Formaten exportiert werden: OBJ, PLY, STL, FBX

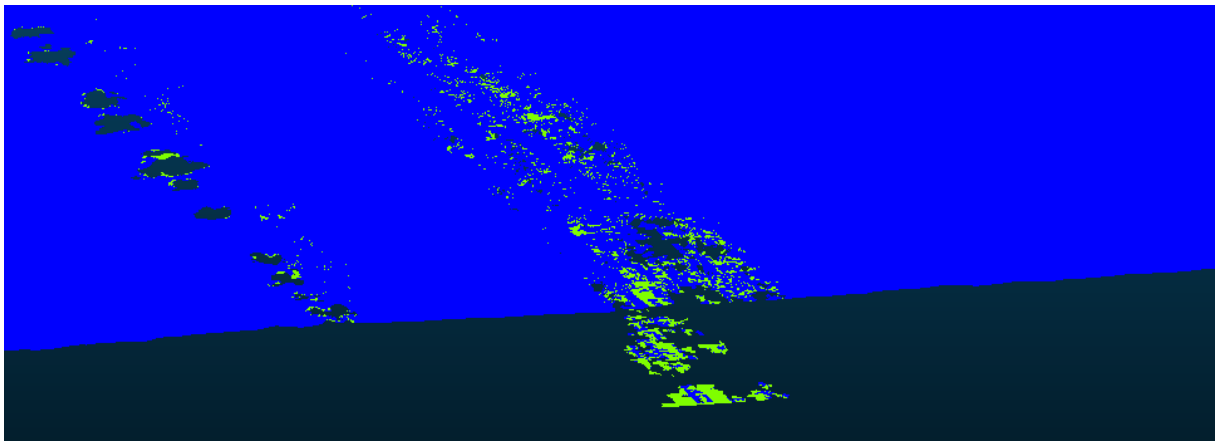

Abbildung 5.6: Darstellung der LAS-Synthetic Flag in CloudCompare

5.2 Automatisierte Ausreißerdetektion mittels überwachter Klassifizierung

Im Rahmen des Experiments zur automatisierten Ausreißerdetektion wurden die zwei häufigsten Ausreißer-Typen für die überwachte Klassifizierung mittels Random Forest ausgewählt. Dabei handelt es sich um Ausreißer im Bereich von Ackerflächen und Solarparks. Dafür wurden zunächst Referenzdaten auf Grundlage von Kacheln mit Ausreißern erzeugt. Danach wurden für alle Referenzdaten zusätzliche Merkmale (Features) berechnet, die klassifizierten Punkte mit zusätzlichen Features exportiert und zu einem Datensatz zusammengefügt. Es wurden hierfür für Acker- und Solarparkausreißer Datensätze erzeugt. Darauf folgte die Angleichung der Beispielanzahlen für jede Klasse. Nach der Angleichung erfolgte die Bestimmung der besten Features und das Training/ Validierung der Klassifikatoren auf den gesamten Datensätzen. Abschließend wurden die erstellten Klassifikatoren auf unbekannten klassifizierten Kacheln getestet, welche die entsprechenden Fehler beinhalten.

5.2.1 Datenvorbereitung

Referenzdaten

Bevor mit der Klassifizierung begonnen werden konnte, mussten zu erst Referenzdaten für Training, Validierung und Test vorbereitet werden. Dafür wurden alle vom LDBV übergebenen Fehler-Beispiele gesichtet und nach Fehlerart sortiert. Die Sortierung ergab eine große Anzahl an Beispielen für Ackerausreißer und eine sehr geringe Anzahl (4x) an Beispielen für Solarpark-Fehler. Betrachtet man die Verteilung der übergebenen Datensätze in Abbildung 2.1, befinden sie sich in der Mitte Bayerns. Sie zeigen Häufungen im Bereich Donauwörth-Günzburg und Kelheim-Regensburg-Straubing. Wenige Bespiele befinden sich noch zusätzlich im Bereich von Vilshofen an der Donau.

Für den Trainings-/ Validierungsdatensatz (T/V-Datensatz) für die Detektion von Ackerausreißer wurden 15 Datensätze exemplarisch ausgewählt und drauf geachtet, dass beide Häufungsgebiete abgedeckt werden. Bei der Auswahl der Kacheln wurde zudem darauf geachtet, dass diese möglichst unterschiedliche Beispiele beinhalten. Bei der Klassifizierung wurden dann Ausreißer und Beispiele für fehlerfreie Punkte sowie der allgemeinen Umgebung klassifiziert. Die Klassifizierung der Ausreißer erfolgte knapp über der tatsächlichen Ackeroberfläche. Die allgemeine Umgebung umfasst dabei verschiedenste Punkte, die nicht auf Ackerflächen liegen. Nach der Klassifizierung der Beispiele in allen Kacheln wurden diese dann extrahiert und in einem Gesamtdatensatz für die weitere Bearbeitung zusammengefasst.
Für den Test auf unbekannten Beispielen wurden drei Kacheln für die Detektion von Ackerausreißern ausgewählt und im ganzen klassifiziert. Alle drei Kacheln weisen unterschiedliche Umgebungszusammensetzungen auf.

Für den T/V-Datensatz für die Detektion von Solarpark-Fehlern wurden drei der vier

Kacheln zur Erstellung des Trainingsdatensatzes verwendet. Bei der Klassifizierung wurden Ausreißer und Beispiele für fehlerfreie Solarpark-Flächen sowie der allgemeinen Umgebung klassifiziert. Da die beiden Ausreißer-Typen zunächst getrennt von einander untersucht werden, wurden bei der Klasse Umgebungspunkte auch Punkte auf Ackerflächen miteinbezogen. Nach der Klassifizierung der Beispiel wurden auch hier alle Beispiele extrahiert und in einem Gesamtdatensatz für die weitere Bearbeitung zusammengefasst.
Für den Test auf unbekannten Beispielen wurde die verbleibende Kachel im ganzen klassifiziert.

Zusätzlich zur Erstellung der Referenzdaten wurde zu jeder verwendeten Kachel das digitale Geländemodell gesucht und heruntergeladen. Dies diente der späteren Berechnung von Höhendifferenzen zwischen DOM- und DGM-Punkten.
Die Sichtung, Klassifizierung sowie Extraktion aller Beispiele erfolgte mit den zuvor getesteten Software Produkten Terra Solid (Modeller und Scan) und CloudCompare.

Übersicht der Testgebiete für die Detektion von Ackerausreißer:

Im Folgendem werden die für den Test der **Acker-Ausreißer-Klassifikation** ausgewählten Test-Kacheln und den in ihnen enthaltenen Fehler beschrieben sowie in RGB-Farben und nach Klassen eingefärbt visualisiert.

Testgebiet 1 beinhaltet Acker-Ausreißer, fehlerfreie Ackerflächen und Umgebungspunkte in Form vom Wäldern, Feldwegen und einen kleinen Ausschnitt einer asphaltierten Straße (Ost-Nord-Ost). Dieses Testgebiet hat den größten Anteil an Ackerflächen, welche verschiedene Brauntöne aufweisen. Des weiteren weisen die Ackerausreißer im rot markierten Gebiet nicht alle eine braune Färbung auf. Die Ausreißer liegen in einem Gebiet, wo sich grüne und hellbraune Streifen abwechseln. Die Feldwege ähneln zudem der Farbe der Äcker. Die im Testgebiet enthaltenen Fehler heben sich in mehreren einzelnen Ebenen von der eigentlichen Oberfläche ab, wie in Abbildung 5.8 und 5.9 zusehen ist. Die Ausreißerpunkte liegen hier ober- und unterhalb der eigentlichen Oberfläche.

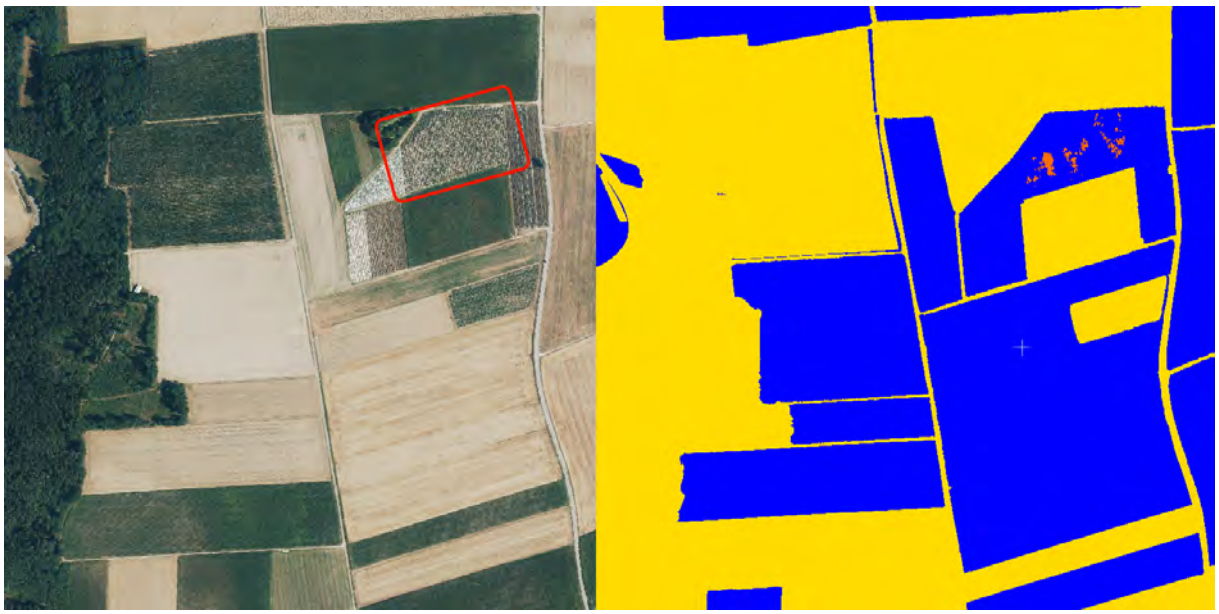

Abbildung 5.7: Detektion von Ackerausreißern - Testgebiet 1 - RGB-Darstellung links, Klassifizierungs-Darstellung rechts

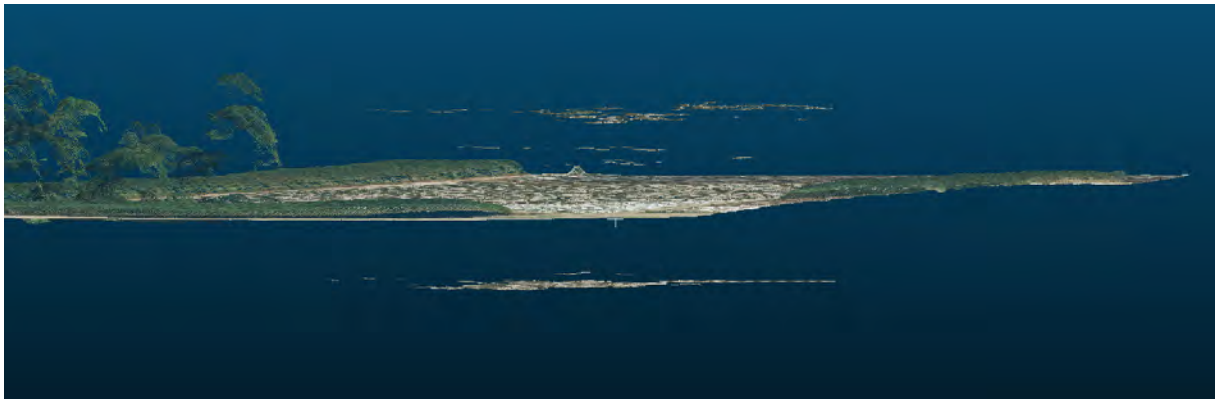

Abbildung 5.8: Detektion von Ackerausreißern - Testgebiet 1 (RGB) - Ausreißer im Detail

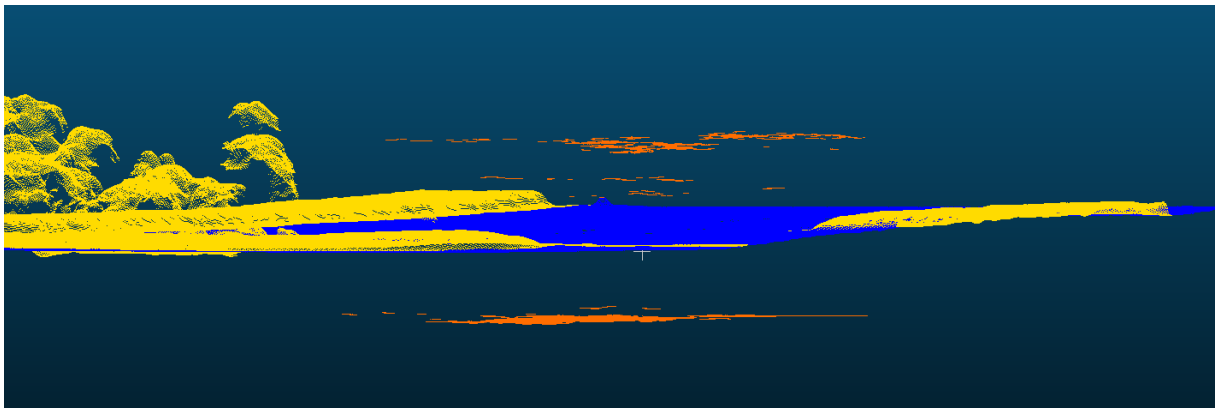

Abbildung 5.9: Detektion von Ackerausreißern - Testgebiet 1 (Klassifizierung) - Ausreißer im Detail

Testgebiet 2 beinhaltet die selben Klassen, hat jedoch weit weniger Ackerflächen als Testgebiet 1. Dafür beinhaltet es Gebäude, mehr Wald und mehr asphaltierte Straßen als Feldwege. Die Ackerausreißer im rot markierten Bereich (Abbildung 5.10, links) befinden sich auf einem Feld mit dunkel und hell brauner Farbe. Die im Testgebiet enthaltenen Fehler heben sich in mehreren einzelnen Ebenen von der eigentlichen Oberfläche ab, wie in Abbildung 5.12 zusehen ist. Die Ausreißerpunkte liegen hier ober- und unterhalb der eigentlichen Oberfläche.

Abbildung 5.10: Detektion von Ackerausreißern - Testgebiet 2 - RGB-Darstellung links, Klassifizierungs-Darstellung rechts

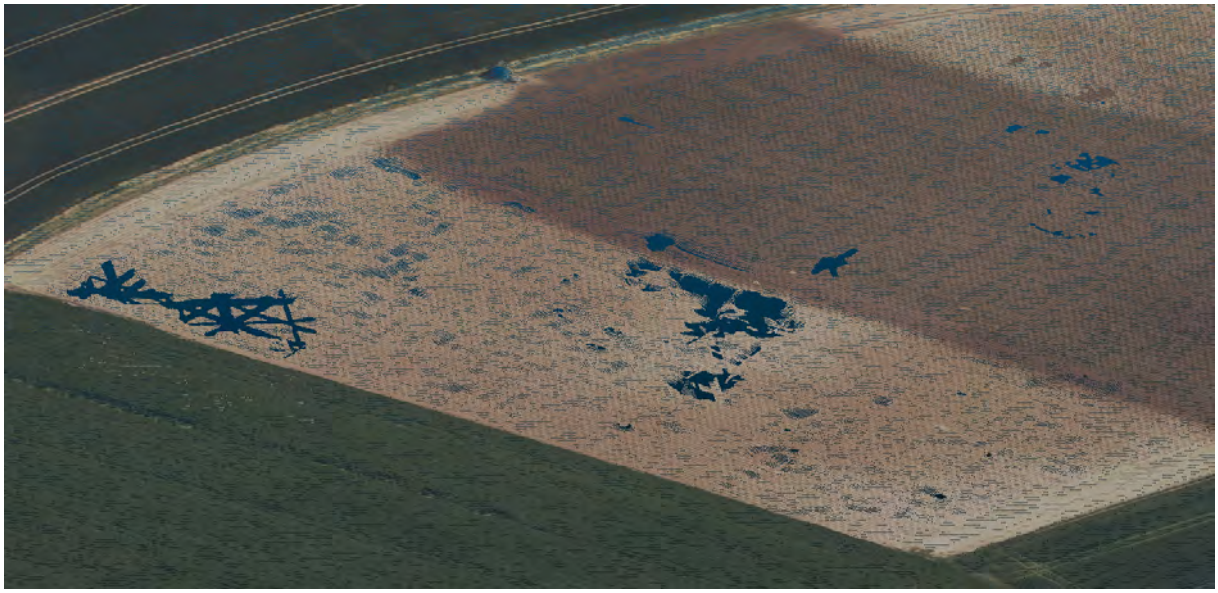

Abbildung 5.11: Detektion von Ackerausreißern - Testgebiet 2 (RGB) - Ausreißer im Detail

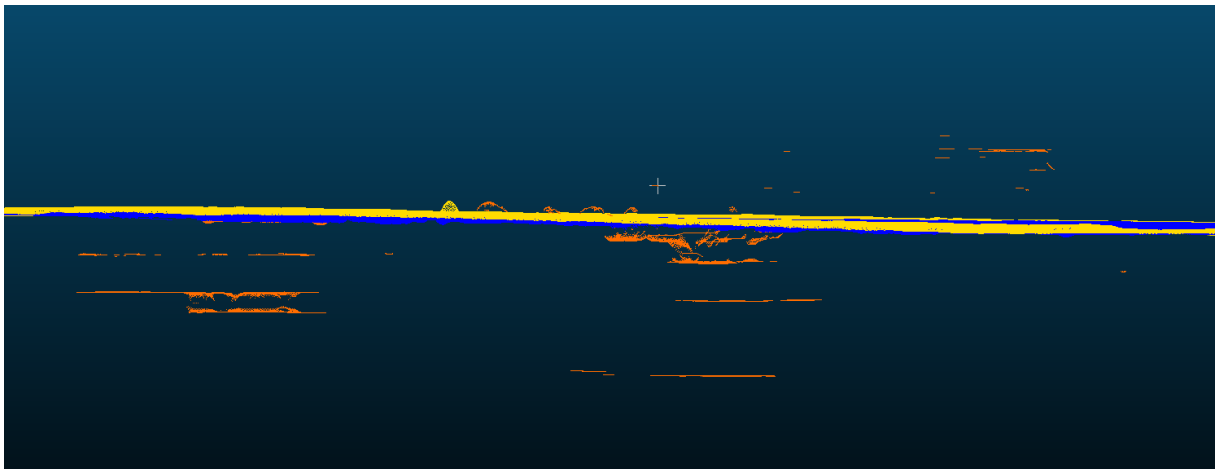

Abbildung 5.12: Detektion von Ackerausreißern - Testgebiet 2 (Klassifizierung) - Ausreißer im Detail

Testgebiet 3 beinhaltet Acker-Ausreißer, fehlerfreie Ackerflächen und Umgebungspunkte. Waldflächen decken den Großteil der abgebildeten Fläche ab. Das Gebiet beinhaltet zudem die geringste Abwechslung in der Flächenzusammenssetzung. Die im Testgebiet enthaltenen Fehler heben sich in mehreren einzelnen Ebenen von der eigentlichen Oberfläche ab, wie in Abbildung 5.14 und Abbildung 5.15 zusehen ist. Die Ausreißerpunkte liegen hier ober- und unterhalb der eigentlichen Oberfläche.

Abbildung 5.13: Detektion von Ackerausreißern - Testgebiet 3 - RGB-Darstellung links, Klassifizierungs-Darstellung rechts

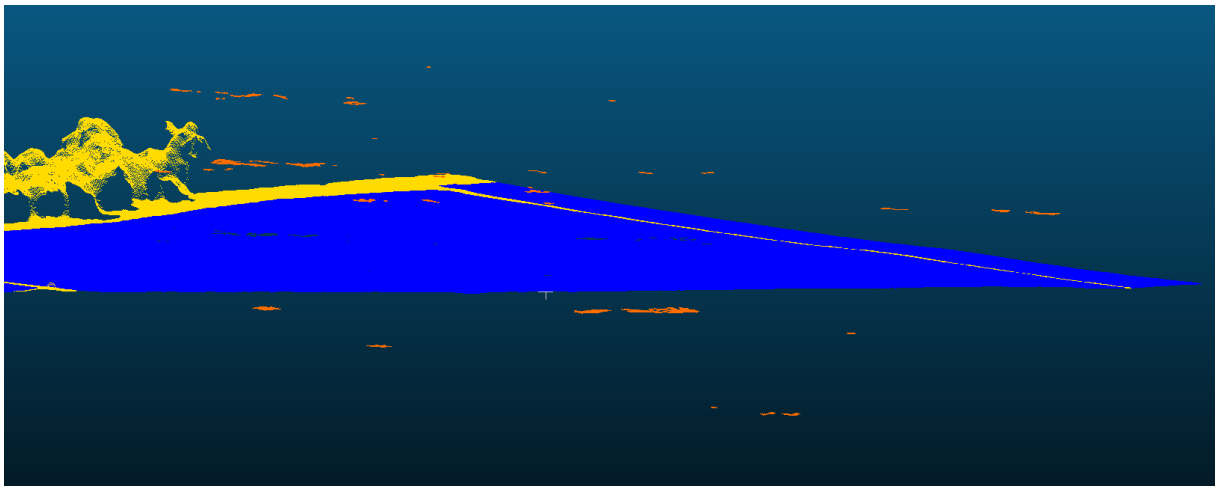

Abbildung 5.14: Detektion von Ackerausreißern - Testgebiet 3 (RGB) - Ausreißer im Detail

Abbildung 5.15: Detektion von Ackerausreißern - Testgebiet 3 (Klassifizierung) - Ausreißer im Detail

Übersicht über das Testgebiet für die Detektion von Solarparkausreißern:

Im Folgendem wird die für den Test der **Solarpark-Ausreißer-Klassifikation** ausgewählte Test-Kachel beschrieben sowie in RGB-Farben und nach Klassen eingefärbt visualisiert. **Testgebiet 4** beinhaltet Solarpark-Fehler, fehlerfreie Solarpark-Flächen und einen sehr hohen Anteil an als Umgebung klassifizierte Flächen. Obwohl es in den Umgebungs-Siedlungsfläche mehrere kleine Solar-Flächen auf den Dächern der Gebäude gibt, wurden diese nicht klassifiziert, um zu prüfen ob diese bei der automatisierten Detektion als Ausreißer oder fehlerfreie Solarparkpunkte erkannt werden. Die im Testgebiet enthaltenen Fehler liegen zum Großteil in einer Ebene über der eigentlichen Oberfläche ab, wie in Abbildung 5.17 und Abbildung 5.18 zusehen ist.

Abbildung 5.16: Detektion von Solarparkausreißern - Testgebiet 4 - RGB-Darstellung links, Klassifizierungs-Darstellung recht

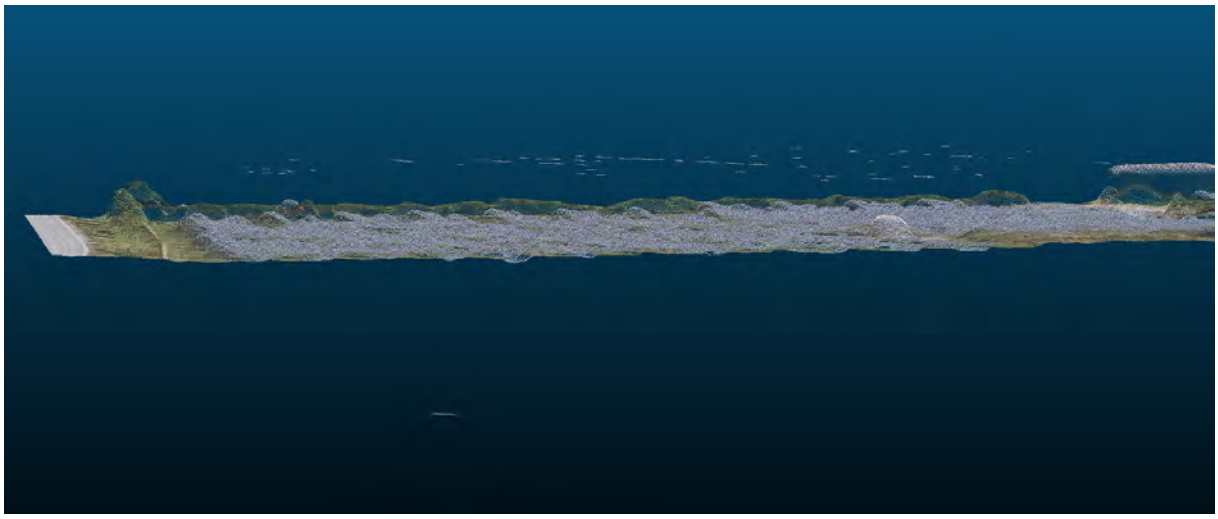

Abbildung 5.17: Detektion von Solarparkausreißern - Testgebiet 4 (RGB) - Ausreißer im Detail

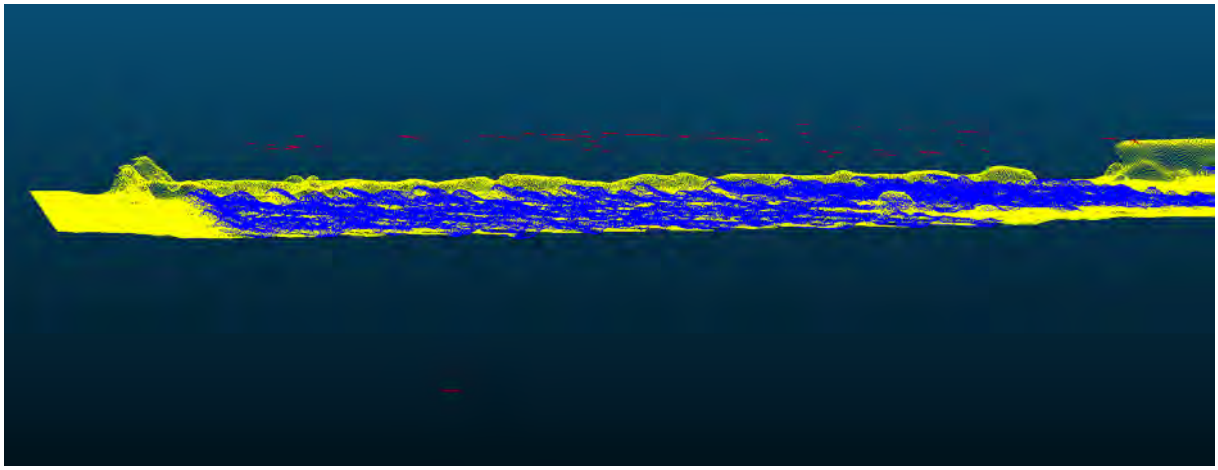

Abbildung 5.18: Detektion von Solarparkausreißern - Testgebiet 4 (Klassifizierung) - Ausreißer im Detail

Übersicht über das Testgebiet für die Detektion von Acker- und Solarparkausreißern:

Testgebiet 5 dient zum Test gleichzeitigen Vorhersage von Acker- und Solarparkausreißern. In dieser Kachel müssen alle entsprechenden Klassen vorliegen. Die Kachel besitzt eine große Anzahl an Ackerausreißern (rot umrandete Bereiche) und große Flächen von fehlerfreien Ackerflächen. Sie besitzt auch zwei Solarpark-Flächen (türkis umrandete Bereiche), welche aber keine Fehler aufwiesen. Deswegen wurde versucht, Ausreißerpunkte zu simulieren, in dem unregelmäßige Segmente aus den Solarparkflächen segmentiert und in ihrer Höhe zufällig verschoben wurden. Die Verschiebung erfolgt hier sowohl über- als auch unterhalb der eigentlichen Oberfläche. Die Modifikation der fehlerfreien Solarparkpunkte zu Solarparkausreißern erfolgt in Cloudcompare (v2.13 beta) durch Segmentierung, Klassifikation und räumlicher Verschiebung.

Abbildung 5.19: Detektion von Acker und Soalrparkausreißern - Testgebiet 5 - RGB-Darstellung links, Klassifizierungs-Darstellung rechts

Abbildung 5.20: Detektion von Acker- und Solarparkausreißern - Testgebiet 5 (RGB) - Ausreißer im Detail

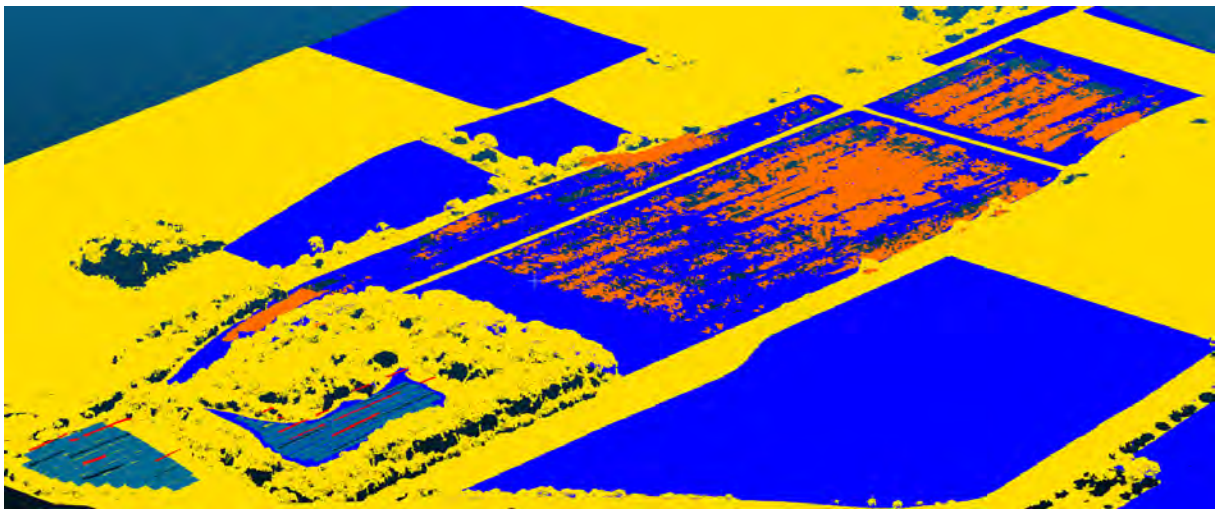

Abbildung 5.21: Detektion von Acker- und Solarparkausreißern - Testgebiet 5 (Klassifizierung) - Ausreißer im Detail

Berechnung zusätzlicher Merkmale

Um für die spätere Klassifizierung weiter Merkmale zu erhalten, wurden zusätzliche Merkmale in CloudCompare berechnet. Dafür wurden zu nächste Abstandswerte zwischen den DOM-Kacheln und den dazu passenden DGM-Kacheln berechnet. Die Berechnung erfolgte über Cloud-2-Cloud Vergleiche ohne zuvor ermittelte Normalen (siehe Beispiel in Abb.5.23). Danach wurden Rauheit, Krümmung, Dichte (siehe Beispiel in Abb.5.24), Moment und geometrische Merkmale für die Radien 1,2m, 1,5m und 1,8m berechnet. Die verschiedenen Radien zur Ermittelung des geeignetsten Radius für das jeweilige Merkmal und zur Erkennung von Radius abhängigen Trends. Nach der Berechnung erfolgte der Export des gesamten Datensatzes inklusive aller Features im xyz-Format.

Zur Berechnung der zusätzlichen Merkmale, ausgeschlossen der Abstandswerte, wurde der Vorgang über ein Batch-File automatisiert (siehe Anhang A.1).Das Batch-File greift hierbei auf die Commandline-Funktion von CloudCompare zu (CloudCompare, 2023). Die Berechnung der Abstandswerte hätte hier auch integriert werden können, wurde aber nicht durchgeführt, da dazu alle DGMs einzeln hätten umbenannt werden müssen.

Zusatz:
Die Detektion der Ackerausreißer ist als erstes Experiment durchgeführt worden. Zu diesem Zeitpunkt gab es noch keine aussagekräftigen Erkenntnisse über den Einfluss der Radien, weshalb hier für zusätzlich die Radien 0,6m und 0,9m berechnet wurden.

Alle in CloudCompare (v2.13 beta) berecheneten Merkmale sind in Abbildung 5.22 dargestellt.

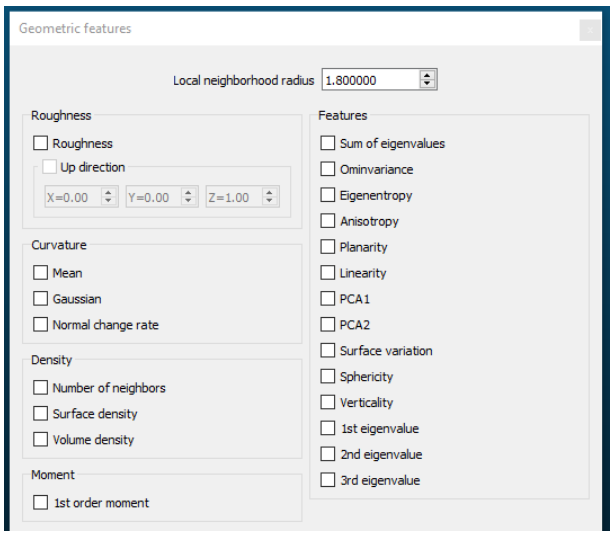

Abbildung 5.22: Berechnung von geometrischen Features in CloudCompare

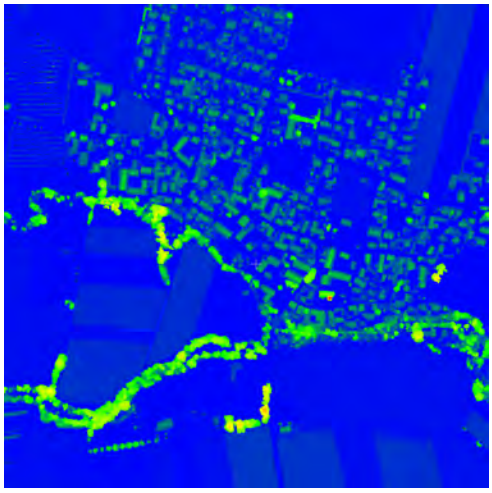

Abbildung 5.23: Beispiel-Darstellung von Hoehendifferenzen ziwschen DOM-DGM für Testgebiet 4

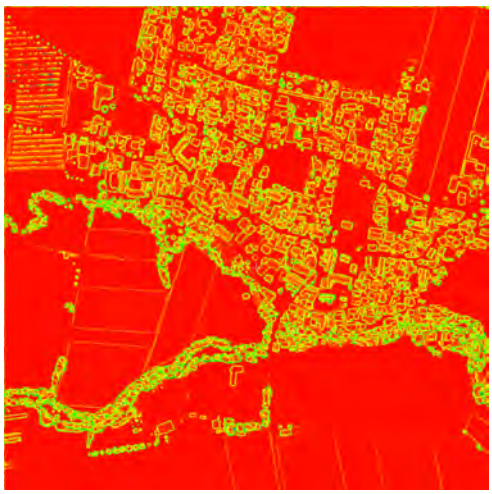

Abbildung 5.24: Beispiel-Darstellung von Nachbaranzahl (R=1.8) für Testgebiet 4

Sampling

Nach der Zusammenstellung der T/V-Datensätze und der Berechnung der geometrischen Features erfolgte das finale Sampling. Dafür sind die jeweiligen Datensätz in Python mittels CSV-Reader importiert worden. Nach dem Import wurden die Koordinaten eines jeden Beispiels gelöscht, um lageabhängige Entscheidungen zu verhindern. Danach wurden Beispiele mit unvollständigen Eigenschaften vom Datensatz entfernt(NaN-Handling), da diese das Ergebnis die Leistung des Klassifikators-Modells verzerren oder verfälschen könnten. Das NaN-Handling verbessert zudem zusätzlich das erlernen genauer Muster. Am Schluss erfolgte die Angleichung der Beispielanzahlen. Die Angleichung der Beispiel-Anzahlen erfolgte mit der Python-Bibliothek "imblearn". Diese bietet die Möglichkeit der zufälligen Reduktion per Undersampling (imbalanced learn, 2023b) und Erhöhung der Beispiele per Oversampling (imbalanced learn, 2023a). Die jeweiligen Datensätze sind wie folgt angeglichen worden:

- Für die Erstellung der T/V-Samples für die **Detektion der Ackerausreißer**, wurde als Angleichungsmethode Undersampling gewählt. Diese Methode wurde gewählt, um die Datenmenge und die Berechnungszeiten zu reduzieren.

  Nach dem Undersampling waren die Klassen in folgendem Verhältnis vorhanden:
  3 x Klassen à 33,3%
  Gesamtanzahl Samples: 7127916

- Für die Erstellung der T/V-Samples für die **Detektion von Solarpark-Fehlern**, wurde die Angleichungsmethode Oversampling gewählt, da der Datensatz vergleichsweise klein war.

  Nach dem Oversampling waren die Klassen in folgendem Verhältnis vorhanden:
  3 x Klassen à 33,3 %
  Gesamtanzahl Samples: 6165006

- Für den **finalen gemeinsamen Test** wurden beide Datensätze zusammengefügt. Dabei wurden die fehlerfreie Solarparkpunkte und Solarparkausreißerpunkte aus dem Solarpark-Datensatz extrahiert und mit dem Acker-Datensatz vereint. Die Extraktion der Klassen erfolgte, damit keine als Umgebungspunkte klassifizierten Ackerpunkte das Ergebnis verfälschen. Nach dem Zusammenfügen folgte eine Angleichung über Oversampling.

  Nach dem Oversampling waren die Klassen in folgendem Verhältnis vorhanden:
  5 x Klassen à 20%
  Gesamtanzahl Samples: 11879860

5.2.2 Klassifikation

Random Forest

Für die Implementierung des Random Forest Algorithmus wurde die freie Python-Bibliothek scikit-learn verwendet (scikit learn, 2023). Die dort beinhaltete Random Forest Funktion bietet alle gängigen Anpassungsmöglichkeiten über Hyperparameter. So können beispielsweise Anzahl der Bäume (n-estimators)und die maximale Tiefe der Entscheidungsbäume angepasst werden. Desweiteren kann die Anzahl der verwendeten Prozessorkerne variert werden, um den Prozess entsprechend der System-Leistung anzupassen oder die Laufzeit zu beeinflussen.
Für die Erstellung des Feature-Rankings wurde die Anzahl der zu bildenden Entscheidungsbäume auf 100 festgelegt. Für die maximale Tiefe des Entscheidungsbaums wurde der Standardwert (default=None) festgelegt.

Training

Für das Training des Klassifikators ist für jeden Datensatz (Acker, Solarpark, Acker und Solarpark) ein Feature-Ranking erstellt worden. Das Feature-Ranking beinhaltet alle Features geordnet nach ihren jeweiligen MDI-Werten. Zusätzlich wurde noch die MDI-Standardabweichung berechnet. Aus diesem Ranking wurden dann die zehn besten Features für das Training und die anschließende Validierung ausgewählt. Bei der Auswahl wurde zudem darauf geachtet, dass jedes Feature nur einmal mit dem besten Radius vorkommt.
Nachfolgend werden die ersten 15 Ergebnisse mit den höchsten MDI-Werten/ Relevanzen für alle drei Datensätze aufgeführt und die verwendeten Features genannt. Die resultierenden MDI-Werte sind hierbei nur innerhalb eines Rankings aussagekräftig und hängen von der Gesamtzahl an Features ab.

Ranking Ackerausreißer Trainingsdatensatz:

Tabelle 5.3: Top 15 Feature Ranking - Acker

Feature Namen	**Feature Wichtigkeit**	**Standardabweichung**
R	0,13	0,09
C2C_absolute_distances[<50]	0,12	0,08
B	0,12	0,08
G	0,08	0,07
Intensity	0,06	0,02
Surface_variation_(1.8)	0,03	0,06
Synthetic_Flag	0,03	0,02
Normal_change_rate_(1.8)	0,02	0,05
Sphericity_(1.8)	0,02	0,04
3rd_eigenvalue_(1.8)	0,02	0,04
Anisotropy_(1.8)	0,02	0,04
Sphericity_(1.5)	0,02	0,05
Normal_change_rate_(1.2)	0,02	0,05
Gaussian_curvature_(1.8)	0,01	0,04

Für das Training des Klassifikators auf dem gesamten Ackerausreißer-Datensatz wurden entsprechend dem Feature-Ranking folgende Features verwendet:

C2C_absolute_distances[<50], R, G, B, Intensity, Synthetic_Flag, Surface_variation_(1.8), Sphericity_(1.8), Anisotropy_(1.8), Normal_change_rate_(1.8)

Um die Abhängigkeit der Vorhersage von den Farbwerten zu überprüfen, wurden Klassifikatoren mit unterschiedlichen Kombinationen an Farbwert-Features erzeugt und auf Testgebiet 1 getestet. Nicht verwendete Farbwerte wurden weggelassen und die verwendeten mit den zusätzlichen Features zusammen getestet. Zur Vergleichbarkeit wurden keine neuen Features als Ersatz hinzugefügt. Dabei wurden die zusätzlichen Features zusammen mit Infrarot, dann mit Infrarot+ Grün und mit allen Farbwerten kombiniert.

Ranking Solarparkausreißer-Trainingsdatensatz:

Tabelle 5.4: Top 16 Feature Ranking - Solarpark

Feature Namen	**Feature Wichtigkeit**	**Standardabweichung**
C2C_absolute_distances[<50]	0,23	0,11
B	0,05	0,05
Number_of_neighbors_(r=1.8)	0,05	0,08
Surface_density_(r=1.8)	0,04	0,08
Verticality_(1.8)	0,04	0,05
Intensity	0,03	0,04
Verticality_(1.5)	0,03	0,04
Volume_density_(r=1.8)	0,03	0,07
Verticality_(1.2)	0,03	0,04
2nd_eigenvalue_(1.8)	0,03	0,07
Surface_density_(r=1.2)	0,03	0,06
Number_of_neighbors_(r=1.2)	0,03	0,06
PCA1_(1.8)	0,02	0,06
Volume_density_(r=1.2)	0,02	0,05
G	0,02	0,05

Für das Training des Klassifikators auf dem gesamten Solarparkausreißer-Datensatz wurden entsprechend dem Feature-Ranking folgende Features verwendet:

C2C_absolute_distances[<50], B, Number_of_neighbors_(r=1.8), Surface_density_(r=1.8), Verticality_(1.8), Intensity, Volume_density_(r=1.8), 2nd_eigenvalue_(1.8), PCA1_(1.8), G

Ranking Acker- und Solarparkausreißer Trainingsdatensatz:

Tabelle 5.5: Top 15 Feature Ranking - Acker und Solarpark

Feature Namen	**Feature Wichtigkeit**	**Standardabweichung**
C2C_absolute_distances[<50]	0,13	0,08
R	0,12	0,08
B	0,11	0,08
G	0,09	0,07
Intensity	0,07	0,02
Surface_variation_(1.8)	0,02	0,05
Synthetic_Flag	0,02	0,02
Anisotropy_(1.8)	0,02	0,05
Normal_change_rate_(1.8)	0,02	0,05
Surface_variation_(1.5)	0,02	0,05
Normal_change_rate_(1.5)	0,01	0,05
Sphericity_(1.8)	0,01	0,04
3rd_eigenvalue_(1.8)	0,01	0,03
Sphericity_(1.5)	0,01	0,04

Für das Training des Klassifikators auf dem gesamten Acker- und Solarparkausreißer-Datensatz wurden entsprechend dem Feature-Ranking folgende Features verwendet:

C2C_absolute_distances[<50], R, B, G, Intensity, Surface_variation_(1.8), Synthetic_Flag , Anisotropy_(1.8), Normal_change_rate_(1.8), Sphericity_(1.8)

Validierung

Um für die jeweiligen Trainingsdaten, nach der Auswahl der besten Features, die besten Hyperparameter zu detektieren, wurde die Funktion GridSeachCV aus der skikit-learn Bibliothek verwendet (scikit learn, 2024). Mit dieser Funktion werden mehrere Kreuzvalidierungen mit unterschiedlichen Hyperparameterkombinationen durchgeführt. Durch diese Funktion werden Validierung und Hyperparametertuning in einem durchgeführt. Der Funktion wurden dafür verschiedene Anzahlen an zu bildenden Entscheidungsbäume und Werte für maximale Baumtiefen übergeben. Der Funktion wurden folgende Parameter zum testen übergeben:

- Anzahl der Entscheidungsbäume:
  10, 20, 50
- Maximale Tiefe:
  5, 10, “None“

Das Ergebnis der Funktion war bei jedem Trainingsdatensatz 50 Entscheidungsbäume und None als maximale Tiefe. Dabei erreichte die Kombination einen Wert von 0.994 bei der Kreuzvalidierung. Der beste zu erreichende Wert bei der Funktion ist 1 und der schlechteste 0. Da sich das Ergebnis zwischen 20 und 50 Entscheidungsbäumen nur um 0,1 Prozent verbessert hat, wurde zur Reduzierung der Laufzeit 20 als Anzahl festgelegt.

Ergebnisse: Vorhersage von Ackerausreißern

Ackerausreißer Testgebiet 1 - Infrarot

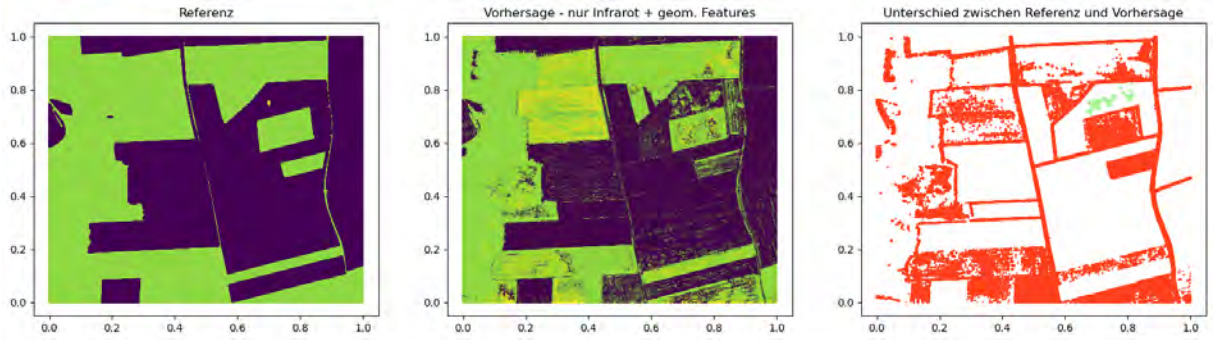

Abbildung 5.25: Ackerausreißer Testgebiet 1 - Infrarot - Visualisierung der Ergebnisse

Tabelle 5.6: Ackerausreißer Testgebiet 1 - Infrarot - Konfusionsmatrix

Vorhersage/ **Referenz**	**Acker**	**Ackerausreißer**	**Umgebung**	Summe
Acker	9941787	6	1297987	11239780
Ackerausreißer	67943	16866	1601410	1686219
Umgebung	1857788	14483	10201524	12073795
Summe	11867518	31355	13100921	24999794

Tabelle 5.7: Ackerausreißer Testgebiet 1 - Infrarot - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.88	0.84	0.86	11867518
Umgebung	0.84	0.78	0.81	13100921
Ackerausreißer	0.01	0.54	0.02	31355
Genauigkeit			0.81	24999794
∅ **ungew.**	0.58	0.72	0.56	24999794
∅ **gew.**	0.86	0.81	0.83	24,999,794

Ackerausreißer Testgebiet 1 - Grün und Infrarot

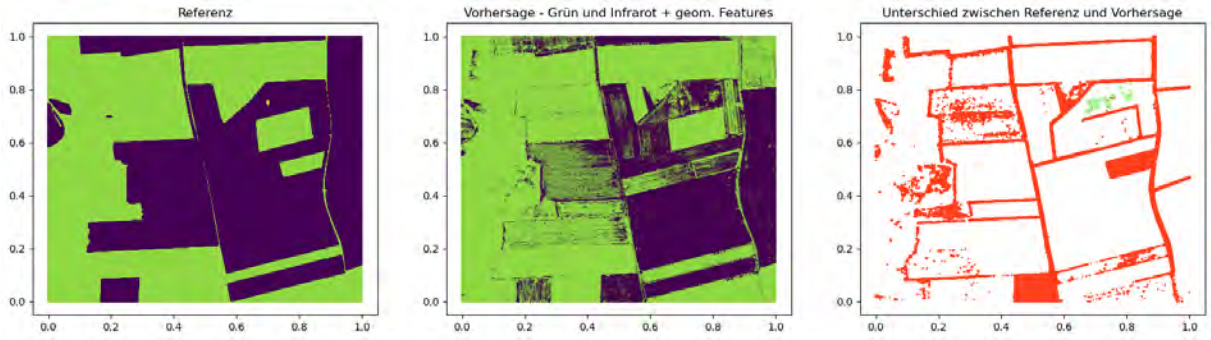

Abbildung 5.26: Ackerausreißer Testgebiet 1 - Grün und Infrarot - Visualisierung der Ergebnisse

Tabelle 5.8: Ackerausreißer Testgebiet 1 - Grün und Infrarot - Konfusionsmatrix

Vorhersage/ **Referenz**	**Acker**	**Ackerausreißer**	**Umgebung**	Summe
Acker	7448082	37	888257	8336376
Ackerausreißer	19981	19695	15464	55140
Umgebung	4399455	11623	1219,200	16608278
Summe	11867518	31355	13100921	24999794

Tabelle 5.9: Ackerausreißer Testgebiet 1 - Grün und Infrarot - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.89	0.63	0.74	11867518
Umgebung	0.73	0.93	0.82	13100921
Ackerausreißer	0.36	0.63	0.46	31355
Genauigkeit	-	-	0.79	24999794
∅ **ungew.**	0.66	0.73	0.67	24999794
∅ **gew.**	0.81	0.79	0.78	24999794

Ackerausreißer Testgebiet 1 - alle Features

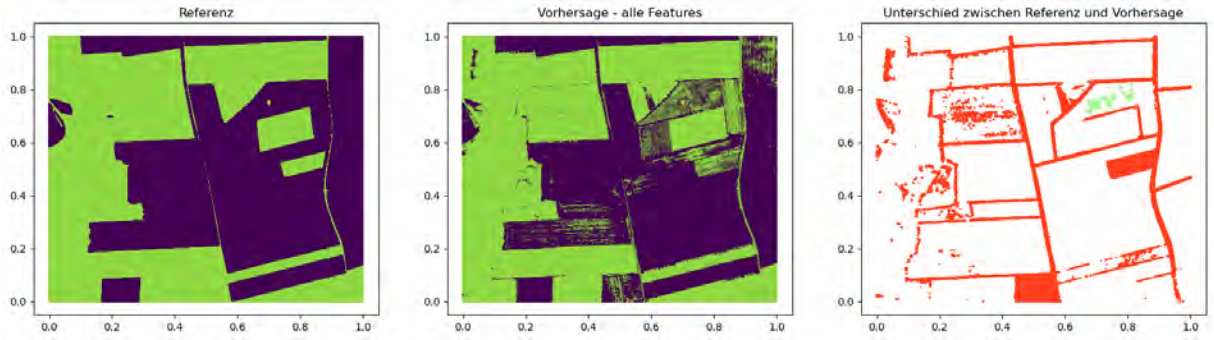

Abbildung 5.27: Ackerausreißer Testgebiet 1 - alle Features (RGBI) - Visualisierung der Ergebnisse

Tabelle 5.10: Ackerausreißer Testgebiet 1 - alle Features (RGBI) - Konfusionsmatrix

Vorhersage/ **Referenz**	**Acker**	**Ackerausreißer**	**Umgebung**	Summe
Acker	10352470	10	791780	11144260
Ackerausreißer	23931	19081	7930	50942
Umgebung	1491117	12264	12301211	13804592
Summe	11867518	31355	13100921	24999794

Tabelle 5.11: Ackerausreißer Testgebiet 1 - alle Features (RGBI) - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.93	0.87	0.90	11867518
Umgebung	0.89	0.94	0.91	13100921
Ackerausreißer	0.37	0.61	0.46	31355
Genauigkeit			0.91	24999794
∅ **ungew.**	0.73	0.81	0.76	24999794
∅ **gew.**	0.91	0.91	0.91	24999794

Ackerausreißer Testgebiet 2 - alle Features

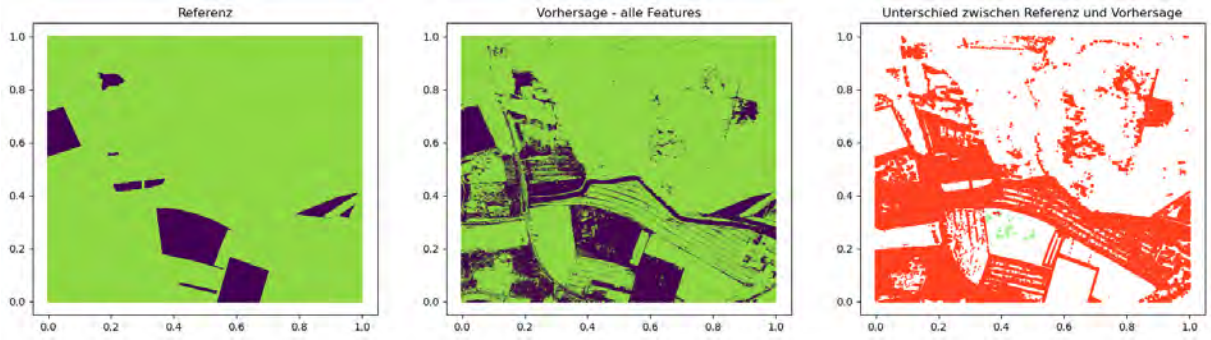

Abbildung 5.28: Ackerausreißer Testgebiet 2 - alle Features (RGBI) - Visualisierung der Ergebnisse

Tabelle 5.12: Ackerausreißer Testgebiet 2 - alle Features (RGBI) - Konfusionsmatrix

Vorhersage/ **Referenz**	**Acker**	**Ackerausreißer**	**Umgebung**	Summe
Acker	2189416	153	2899052	5088621
Ackerausreißer	3521	20236	3965	27722
Umgebung	170168	4123	19708916	19883207
Summe	2363105	24512	22611933	24999550

Tabelle 5.13: Ackerausreißer Testgebiet 2 - alle Features (RGBI) - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.43	0.93	0.59	2363105
Umgebung	0.99	0.87	0.93	22611933
Ackerausreißer	0.73	0.83	0.77	24512
Genauigkeit			0.88	24999550
∅ **ungew.**	0.72	0.87	0.76	24999550
∅ **gew.**	0.94	0.88	0.90	24999550

Ackerausreißer Testgebiet 3 - alle Features

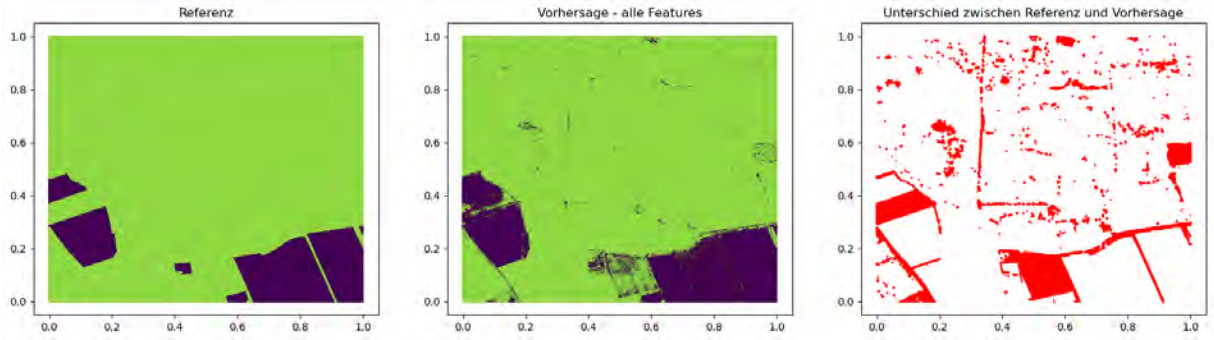

Abbildung 5.29: Ackerausreißer Testgebiet 3 - alle Features (RGBI) - Visualisierung der Ergebnisse

Tabelle 5.14: Ackerausreißer Testgebiet 3 - alle Features (RGBI) - Konfusionsmatrix

Vorhersage/ **Referenz**	**Acker**	**Ackerausreißer**	**Umgebung**	Summe
Acker	3248338	0	499085	3747423
Ackerausreißer	0	10091	127	10218
Umgebung	175060	0	21066413	21241473
Summe	3423398	10091	21565625	24999114

Tabelle 5.15: Ackerausreißer Testgebiet 3 - alle Features (RGBI) - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.87	0.95	0.91	3423398
Umgebung	0.99	0.98	0.98	21565625
Ackerausreißer	0.99	1.00	0.99	10091
Genauigkeit			0.97	24999114
∅ **ungew.**	0.95	0.98	0.96	24999114
∅ **gew.**	0.97	0.97	0.97	24999114

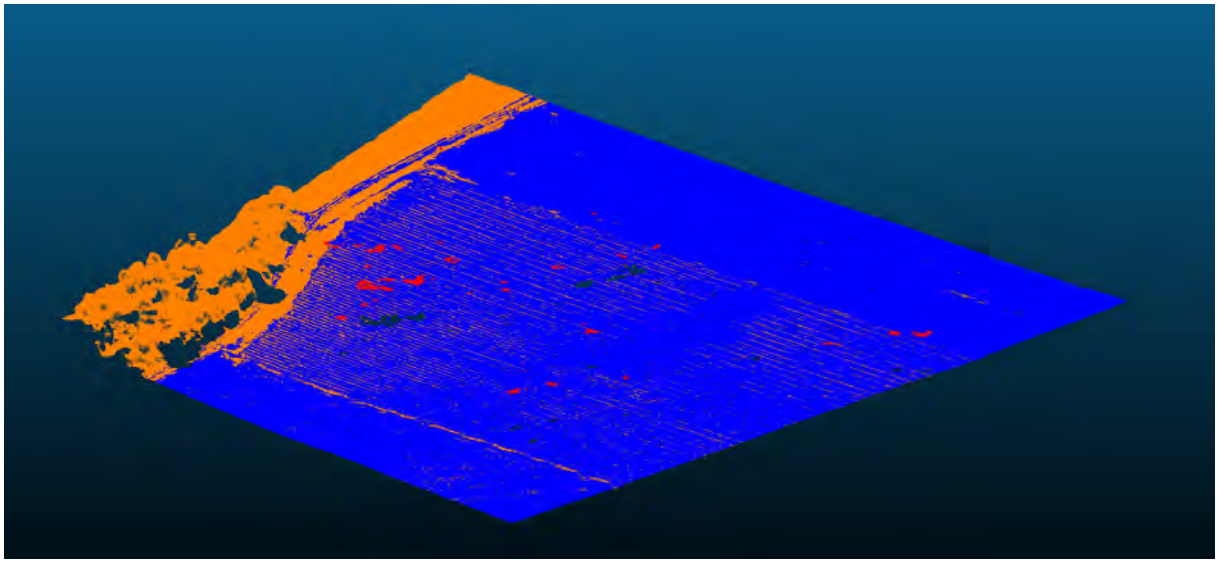

Abbildung 5.30: Ackerausreißer Testgebiet 3 - alle Features (RGBI) - Vorhersage Punktwolke Bild 1

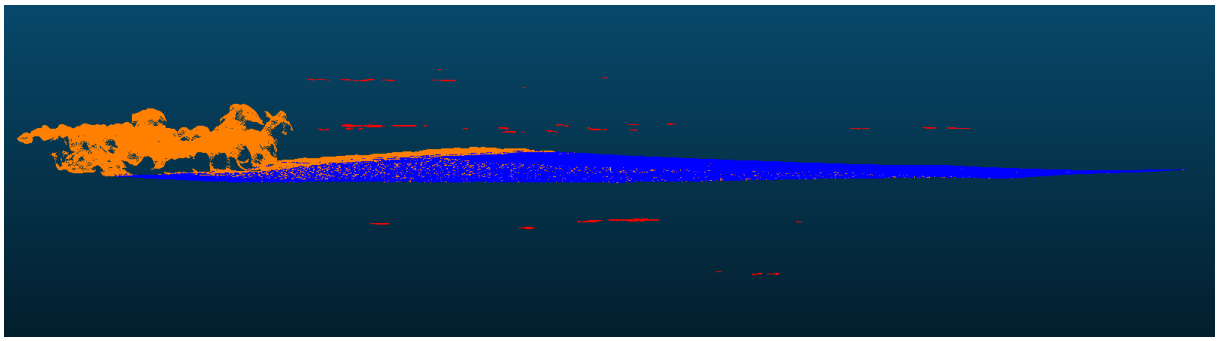

Abbildung 5.31: Ackerausreißer Testgebiet 3 - alle Features (RGBI) - Vorhersage Punktwolke Bild 2

Vorhersage von Solarparkausreißern

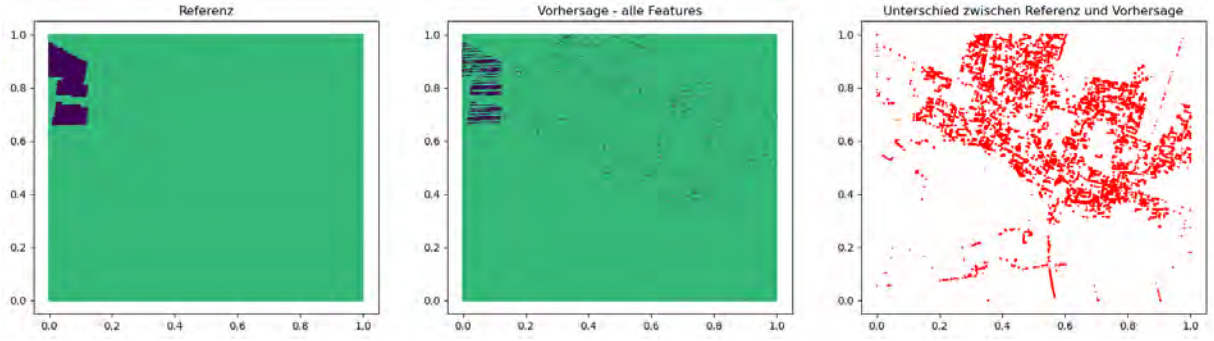

Abbildung 5.32: Solarparkausreißer Testgebiet 4 - alle Features (RGBI) - Visualisierung der Ergebnisse

Tabelle 5.16: Solarparkausreißer Testgebiet 4 - Konfusionsmatrix

Vorhersage/ **Referenz**	**PV-Soll**	**PV-Fehler**	**Umgebung**	Summe
PV-Soll	175197	0	161926	337123
PV-Fehler	0	1900	708	2608
Umgebung	86036	306	24573878	24660220
Summe	261233	2206	24736512	24999951

Tabelle 5.17: Solarparkausreißer Testgebiet 4 - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
PV-Soll	0.52	0.67	0.59	261233
Umgebung	1.00	0.99	0.99	24736512
PV-Fehler	0.73	0.86	0.79	2206
Genauigkeit			0.99	24999951
∅ **ungew.**	0.75	0.84	0.79	24999951
∅ **gew.**	0.99	0.99	0.99	24999951

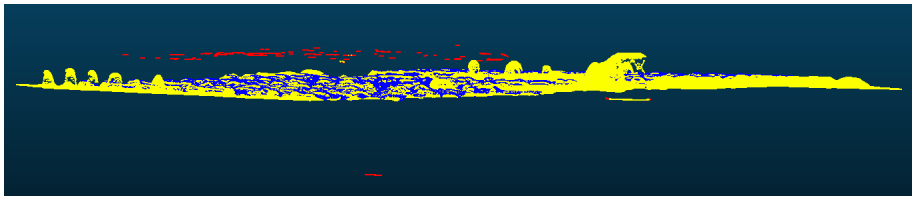

Abbildung 5.33: Solarparkausreißer Testgebiet 4 - Vorhersage - klassifizierte Punktwolke Bild 1

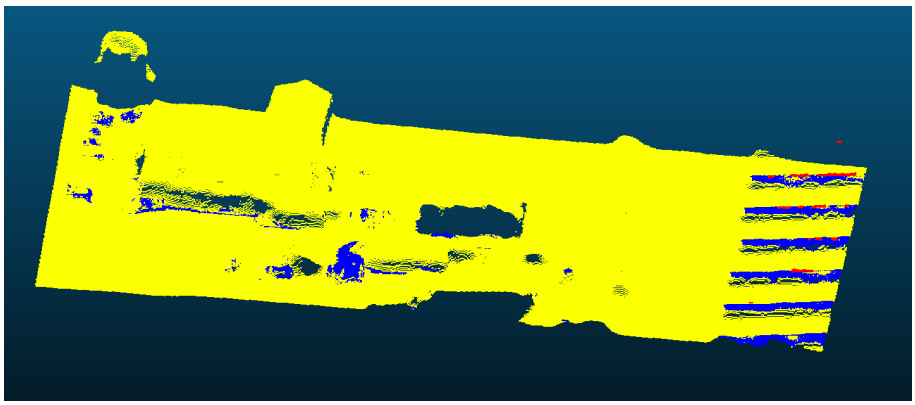

Abbildung 5.34: Solarparkausreißer Testgebiet 4 - Vorhersage - klassifizierte Punktwolke Bild 2

Abbildung 5.35: Solarparkausreißer Testgebiet 4 - Vorhersage - RGB Punktwolke Bild 3

Vorhersage von Acker- und Solarparkausreißern

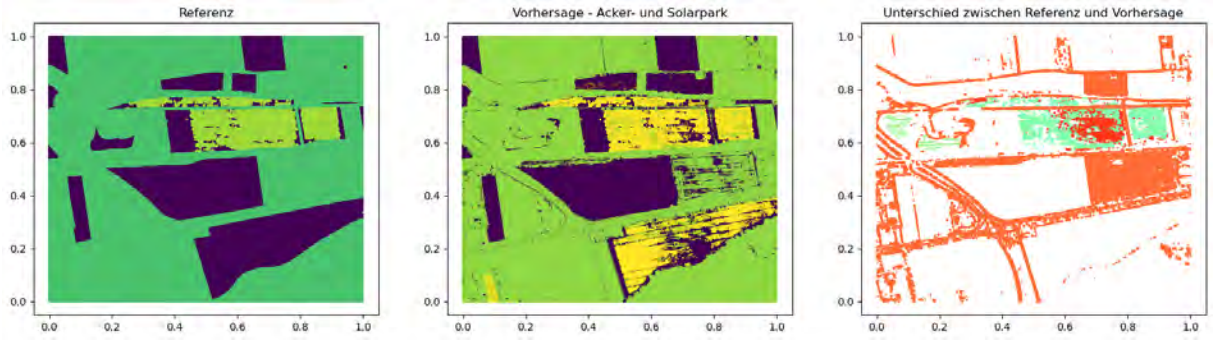

Abbildung 5.36: Detektion von Acker- und Solarparkausreißern - Testgebiet 5 - Visualisierung der Ergebnisse

Tabelle 5.18: Acker- und Solarparkausreißern Testgebiet 5 - Konfusionsmatrix

Vorhersage/ **Referenz**	**Ackerausreißer**	**Acker**	**Solarparkausreißer**	**Solarpark**	**Umgebung**	Summe
Ackerausreißer	930609	14778368	0	83	88125	2497185
Acker	122757	5443245	3	411	798707	6365123
Umgebung	138902	637702	12728	183777	15161324	16134433
Summe	1192268	7559315	12731	184271	16048156	24996741

Tabelle 5.19: Acker- und Solarparkausreißern Testgebiet 5 - Precision, Recall, F1-Score

Klasse	**Precision**	**Recall**	**F1-Score**	**Support**
Acker	0.86	0.72	0.78	7559315
Solarpark	0.00	0.00	0.00	261233
Umgebung	0.94	0.94	0.94	16048156
Ackerausreißer	0.37	0.78	0.50	1192268
Solarparkausreißer	0.00	0.00	0.00	12731
Genauigkeit			0.86	24996741
∅ **ungew.**	0.43	0.49	0.45	24996741
∅ **gew.**	0.88	0.86	0.87	24996741

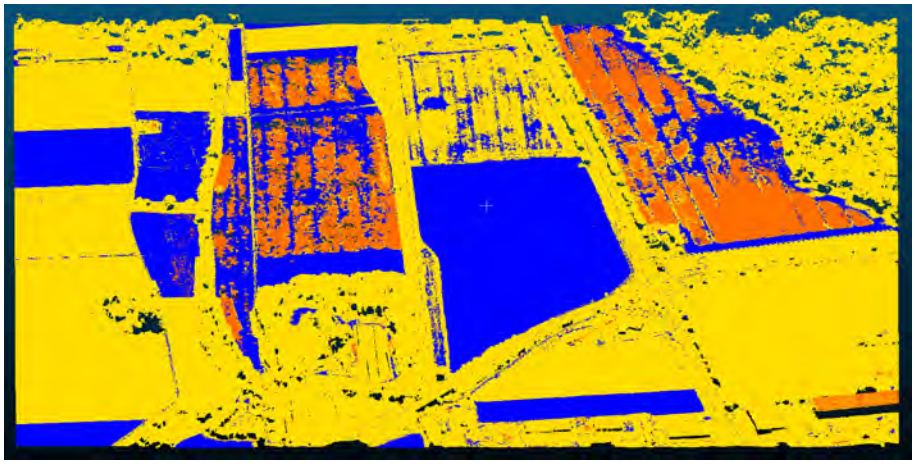

Abbildung 5.37: Acker- und Solarparkausreißern Testgebiet 5 - Vorhersage - Ost-Ansicht perspektivisch

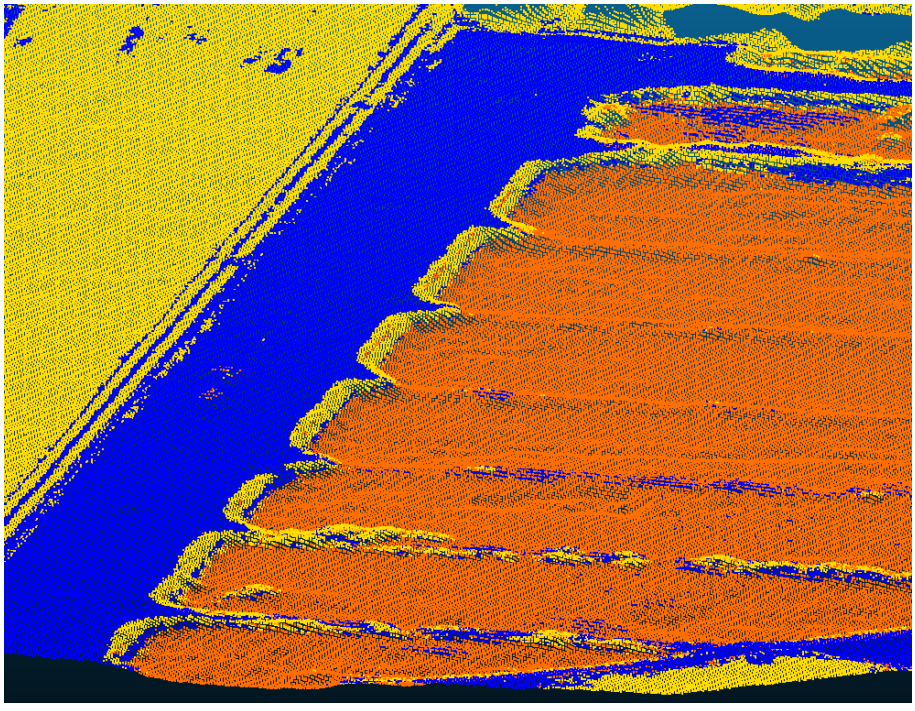

Abbildung 5.38: Acker- und Solarparkausreißern Testgebiet 5 - Vorhersage zuvor nicht klassifizierte Fehler

6 Diskussion

6.1 Softwarealternativen

Durch die in Kapitel 5.1.1 erfassten Rahmenbedingungen konnten klare Kriterien für eine alternative Software herausgearbeitet werden. Die beiden zum Test ausgewählten Produkte erfüllen alle Anforderungen für den aktuellen Arbeitsablauf der Ausreißerbearbeitung. Beide Programm können alle vorhandenen LAS/ LAZ-Formate öffnen. Terra Solid bietet hier aber zusätzlich Möglichkeit der Verwendung von Bezugssystemen und auch die Transformation in ein anderes Bezugssytem. CloudCompare kann hier nur die Koordinaten nutzen, die übergeben werden. Beim Daten Export können bei beiden Produkten alle vorhanden LAS-/ LAZ-Formate ausgegeben werden. Hier kann bei Terra Solid lediglich die Version ausgewählt werden, während bei CloudCompare die Version und das Punktformat angegeben werden können. Zusätzlich können in CloudCompare erweiterte Eigenschaften in VLR und EVLRs ausgegeben werden.

Beide Programm können große Datenmengen verarbeiten und sind nur durch die Hardware limitiert. Bei CloudCompare können mehrere Kacheln separiert voneinander eingeladen werden, was eine indirekte Queue-Bearbeitung ermöglicht. Terra Solid kann auch mehrere Kacheln und importieren, jedoch ist hierbei die Bearbeitung umständlicher. Beim Import muss jeder Kachel eine Nummer zugewiesen werden und beim Export entsprechend angegeben werden. Die Zuordnung beim Export ist hier umständlich und stellt eine große Fehlerquelle bei der Bearbeitung großer Datenmenge dar.

Bei der Bearbeitung von Punktwolken bieten beide Produkte die Möglichkeit der räumlichen Transformation mittels Verschieben und Rotieren. Terra Solid bietet hier zusätzlich die Möglichkeit der Korrektur von Höhen bezogen auf die vorhandenen Methoden. Damit besitzt es eine ähnliche Funktion wie DTMaster. Der Unterschied besteht aber darin, dass bei DTMaster Höhen tatsächlich auf eine gerechnete Oberfläche gerechnet werden, während bei Terra Solid lediglich eine feste Höhe über Perzentile angegeben werden kann.

Die Klassifizierungs-Möglichkeiten von Terra Solid sind CloudCompare klar überlegen. Während bei Terra Solid Schnitte angelegt werden können, in denen nur die Sichtbaren Punkte editiert werden können, muss man diese in CloudCompare zuvor segmentieren oder eine LimitBox verwenden. Die in Abbildung 5.3 dargestellten Klassifizierungswerkzeuge bieten zudem eine flexiblere Klassifizierung von Punkten, ohne vorher segmentieren zu müssen.

Mit beiden Programmen können Vermaschungen erzeugt und bearbeitet werden. Während bei CloudCompare Vermaschungen nur geglättet werden können, bietet Terra Solid eine Toolbox mit Bearbeitungswerkzeugen. Die Bearbeitung von ganzen Kacheln führt bei

Terra Solid jedoch zu langen Wartezeiten, ohne das System merklich auszulasten. Beide Programme bieten den Export in Vermaschungs-Datenformaten.

6.2 Automatisierte Ausreißerdetektion

6.2.1 Detektion von Ackerausreißer

Das Feature-Ranking in Tabelle 5.3 zeigt, dass die Standardfarbwerte (RGB) der Punkte einen hohen Einfluss auf die Entscheidungen der Vorhersage haben. Der MDI-Wert der Farben liegt dabei zwischen 0,08 und 0,13. Die dazugehörigen Standardabweichungen liegen zwischen 0,07 und 0,09. Die Standardabweichungen sind im Vergleich zum MDI sehr groß. Ein Grund hierfür könnte die große Streuung an an Eingangswerten sein, die dem Klassifiaktor beim Training übergeben werden. Die berechnete Geländehöhe ist die zweitbeste Eigenschaft für die Vorhersage. Der MDI-Wert und die Standardabweichung ähnelt den Wert für der Farbwerte. Absteigend folgt die Intensität/ der Infrarot-Kanal mit einem MDI-Wert von 0,06 und einer Standardabweichung von 0,02. Die Standardabweichung des Infrarot-Kanals ist im Vergleich zu den anderen Farbkanälen signifikant besser. Nach den zuvor genannten Eigenschaften ist ein signifikanter Abfall der Wichtigkeiten zu erkennen. Es folgen die "Synthetic Flag"-Eigenschaft und die berechneten geometrischen Features. Die Standardabweichungen der Features ähneln oder übertreffen dabei die MDI-Werte. Zudem sieht man, dass die Features mit den größten Umfangsradius, die besten Ergebnisse erzielt haben. Betrachtet man das komplette Ranking über alle Features von 0,6-1,8m, sieht man einen fast linearen Anstieg der MDI-Werte. Hier könnte man weiter testen, ob noch größere Radien bessere Ergebnisse geliefert hätten. Im Vergleich zu den Farbwerten haben die zusätzliche berechneten Features bisher nur eine geringe Aussagekraft.

Die Detektion von Ausreißerpunkten mit verschiedenen Features auf Testgebiet 1 hat bei der Detektion mit allen Farben die besten Ergebnisse zur Detektion von Ackerausreißern hervorgebracht. Dabei konnte mit der Kombination von Grün und Infrarot fast identische Ergebnisse für die Detektion von Ausreißern erzielt werden. Die resultierende Gesamtgenauigkeit ist aber mit 0,79, der der Vorhersage mit allen Farben mit 0,91 unterlegen. Aus diesem Grund wurden alle weiteren Vorhersagen mit allen Farben vorgenommen.

Die Detektion von Ausreißerpunkten mit allen Features hat auf den drei Testgebieten unterschiedlich gut funktioniert. Betrachtet man die F1-Werte für die Vorhersage von Ackerausreißer mit allen Features von Tabelle 5.10 bis 5.14, so liegt der schlechteste Wert bei 0,46 und der beste bei 0,99. Über alle drei Ergebnisse hinweg liegt der Recall-Wert bei minimal 0,61 und maximal 1,00. Der Precision-Wert liegt bei minimal bei 0,37 und maximal bei 0,99.

Die Vorhersage auf **Testgebiet 1** zeigt bei Ackerausreißern eine schlechte Präzision mit

nur 0,37 (Tabelle 5.11). Es wurden viele fehlerfreie Acker- und Umgebungspunkte fälschlicherweise als Ackerausreißer klassifiziert (Tabelle 5.10). Die meisten dieser Punkte liegen in unmittelbarer nähe zum tatsächlichen Häufungsbereich der Ackerausreißer und zeigen die selben Geländestrukturen wie der Häufungsbereich auf. Das Gelände an diesen Stellen beinhaltet linienförmige Haufen an denen sich hellbraune und grüne Streifen abwechseln. Die fehlerhaften Punkte sind hierbei an Stellen, wo es größere Differenzen zum DGM gibt und die Farbe mit den Ackerausreißern identisch ist. Für die restlichen falsch vorhergesagten Punkte, außerhalb des Bereichs, ergibt sich die selbe Beobachtung. Damit lässt sich sagen, dass die Differenz zum DGM und die Farbe der Ausreißer einen großen Einfluss auf die Entscheidungen des Modells haben. Dies bestätigt das Ergebnis des zuvor ermittelten Feature Rankings. Weiter ist anzumerken, dass damit auch schon die Höhe, ab der Ackerausreißer in den Referenzdaten klassifiziert werden, das Ergebnis beeinflussen kann.

Der resultierende Recall-Wert für Testgebiet 1 bei Ackerausreißern ist mit 0,65 befriedigend. Sieht man sich die tatsächlichen Ausreißer an, sieht man auf den Ausreißerpunkten ein Streifenmuster in der Klassifizierung. Dieses Muster entspricht in Lage den Farben der Punkte. Grüne Punkte wurden hier als Umgebung und hellbraune Punkte als Ackerausreißer klassifiziert. Auch hier ist der Einfluss der Farbe klar in den Ergebnissen zu erkennen und beinflusst das Ergebnis in diesem Fall negativ.

Bei allen anderen Punkten im Testgebiet ist ein ähnliches Muster zu erkennen. Grüne Streifen in Äckern werden als Umgebung erkannt. In Abbildung 5.27 sieht man, dass Feldwege mit Acker ähnlichen Farbeigenschaften, als Acker erkannt werden. Zusätzlich führen Randbereiche neben den Feldwegen, welche als Umgebung klassifiziert aber als Acker vorhergesagt worden sind, zu einer fast vollständigen Visualisierung der Wegestruktur im Testgebiet bei der Differenzdarstellung. Trotz Unsicherheiten in der manuellen Klassifizierung und dem Hohen Einfluss der Farbgebung, konnte für beide Klassen mit einem F1-Wert von >=0,9 ein gutes Ergebniss erzielt werden. Eine Unterscheidung zwischen Acker und nicht Acker ist somit gegeben und unterstützt weiter die Unterscheidung zu Ackerausreißern.

Die Vorhersage auf **Testgebiet 2** zeigt bei der Detektion der Ackerausreißern eine besseres Ergebnis (Konfusionsmatrix in Tabelle 5.13) als bei Testgebiet 1. Das Ergebnis für die Präzission ist mit 0.73 um 0.36 besser. Der Recall-Wert ist mit 0,83 um 0,22 besser. Der resultierende F1-Score verbessert sich um 0,31 auf 0,77. Im Vergleich zu Testgebiet 1 gibt es keine auffälligen Geländestrukturen oder Unterschiede in der Farbgebung bei den fehlerfreien Ackerpunkten sowie bei den Ackerausreißern. Daraus kann geschlossen werden, dass mit geringerer Komplexität der vorhandenen Ackerfehler, diese auch besser erkannt werden können. Auch hier bestätigen sich die Erkenntnisse von Testgebiet 1. Dadurch das bei der Klassifikation der Referenzdaten nicht nach Farbwerten sondern nach tatsächlichen Klassen unterschieden wird, treten bei den anderen beiden Klassen die selben Differenzen auf wie auch schon bei Testgebiet 1. In Abbildung 5.28 sieht man zwischen Referenz und Vorhersage starke Unterschiede. Man sieht das viele Bereiche mit Acker ähnlichen Farben als solches klassifiziert werden. Resultierend ist die Präzision für die Klasse der fehlerfreien Ackerpunkte mit 0,43 im Vergleich zum vorherigen Beispiel wesentlich schlechter. Der

Recall-Wert von 0,93 zeigt jedoch, dass ein Großteil der zur Unterscheidung notwendigen fehlerfreien Ackerpunkte dennoch gefunden wurde.

Die Vohersage auf **Testgebiet 3** erzielte das beste Ergebnis bei der Detektion der Ackerausreißer. Die Präzision und der Recall-Wert sind mit 0,99 und 1,00 die besten Ergebnisse der drei Tests. Der resultierende F1-Score ist mit 0,99 auch das beste erzielte Ergebnis. Auch hier gibt es keine auffälligen Geländestrukturen oder Unterschiede in der Farbgebung bei fehlerfreien Ackerpunkten sowie bei den Ackerausreißern. Da die Kachel eine sehr abwechslungsarme Flächenzusammensetzung hat, reduziert sich auch die Anzahl von Vorhersagefehlern aufgrund von sich ähnendeln Farben.

Die Vorhersage für Umgebungspunkte erzielte eine Präzision von 0.99 und einen Recall-Wert von 0,98. Die Vorhersage der fehlerfreien Ackerpunkte erzielte eine Präzision von 0,87 und einem Recall-Wert von 0,95. Die hohen Werte für Präzision und Recall bei der Detektion von Umgebungspunkten sind auf die hohe Anzahl an Punkten zurückzuführen. Betrachtet man Tabelle 5.15, sind 21,5 von 25 Millionen Punkte in der Umgebungs-Klasse. Fehlerhafte Vorhersagen sind somit im Verhältnis zur Gesamtpunktzahl relativ gering. Der Präzisions-Wert der fehlerfreien Ackerpunkte erklärt sich wieder durch die fehlerhafte Klassifikation von Punkten mit Acker ähnlicher Farbe. Der Recall-Wert zeigt, dass die meisten fehlerfreien Ackerpunkte gefunden worden sind.

6.2.2 Detektion von Solarparkausreißer

Das Feature-Ranking in Tabelle 5.4 zeigt, dass die berechnete Geländehöhe die höchste Relevanz bei der Vorhersage hat. Die Relevanz ist dabei mit 0,23 fast vier mal höher, als die des zweit relevantesten Features, der Farbe Blau mit 0,05. Alle nachfolgenden Features haben eine Relevanz von 0,04-0,02. Die Standardabweichung der Features ist bei der Geländehöhe noch geringer als der MDI-Wert, bei Blau gleich und bei allen anderen Features größer als der zugehörige MDI-Wert. Betrachtet man die Radien der Features, so erkennt man, dass auch hier wieder die Features mit den größten Radien die höchste Relevanz besitzen.

Die Vorhersage der Solarparkausreißer erreicht eine Präzision von 0,73 und einen Recall-Wert von 0,86 (Tabelle 5.16). Die Vorhersage der fehlerfreien Solarparkflächen erreicht eine Präzision von 0,52 und einen Recall-Wert von 0.67. Die Umgebung erzielt eine Präzision von 1,00 und einen Recall von 0.99.

Betrachtet man den Recall-Wert der Vorhersage der Ausreißer, so wurde ein Großteil der vorhandenen Punkte gefunden. Auch die Umgebungspunkte wurden zuverlässig vorhergesagt. Die Vorhersage auf fehlerfreie Solarparkpunkte erzielte das schlechteste Ergebnis. Betrachtet man Abbildung 5.32, sieht man viele als fehlerfrei vorhergesagte Solarparkpunkte im Bereich der in Testgebiet 4 vorhandenen Siedlungsstruktur (RGB-Visualisierung in Abb.5.16). Betrachtet man dazu Abbildung 5.18 und 5.17, erkennt man das viele falsch klassifizierte Punkte in Bereichen liegen, die eine ähnliche blaue Färbung aufweisen und auf der selben Höhe liegen wie die Solarpanele. Durch die hohe Relevanz der Farbe Blau

und der Geländehöhe, scheint es hierbei Schwierigkeiten bei der Trennung zu geben. Trotz Schwierigkeiten bei der Trennung der Punkte auf Bodenhöhe, wurden keine Solaranlagen auf Dächern fälschlicherweise als Solarparkflächen erkannt.

6.2.3 Detektion von Acker- und Solarparkausreißer

Das Feature-Ranking des kombinierten Trainingsdatensatzes in Tabelle 5.5 ähnelt stark dem Ranking des Ackerausreißerdatensatzes. Die wichtigsten Features sind wieder Geländehöhe und die Farbwerte. Auch hier sind die besten zusätzlichen Features, die mit den größten Radien.

Die Vorhersage auf beide Ausreißertypen erzielt für Ackerausreißer eine Präzision von 0,37 und einen Recall von 0,78 (Tabelle 5.18). Bei der Vorhersage der Solarpark und dessen Ausreißer wurden keine Punkte gefunden. Die Vorhersage der Umgebung erreichte eine Präzision von 0,94 und einen Recall von 0,94. Ackerpunkte konnten mit einer Präzision von 0,86 und einem Recall von 0,72 vorhergesagt werden.

Das schlechte Ergebnis für die Präzision bei den Ackerausreißern und den Recall bei den fehlerfreien Ackerpunkten lässt sich anhand der Visualisierung der Ergebnisse in Abbildung 5.36 erklären. Beim Vergleich von Referenz und Vorhersage fällt auf, dass im südlichen Teil des Testgebiet Ackerfehler erkannt wurden, die bei der Klassifizierung der Test-Kachel nicht erkannt worden sind. In Abbildung 5.38 werden diese von der Seite betrachtet. Der betroffene Acker hat keine Löcher, wie die anderen Äcker mit Ausreißern. Auch eine Prüfung der synthetischen Punkte hat in diesem Bereich keine Hinweise auf Fehler gegeben. Die Ausreißer treten hier nicht in Stufen, sondern in Wannen und Wölbungen auf. Die Menge der neu gefundenen Ackerausreißer entspricht ungefähr der Menge der zuvor klassifizierten Ackerausreißer, was die Präzision negativ verfälscht. Der Recall-Wert der Ackerklasse wird auch entsprechend negativ verfälscht, da die Ausreißer die Anzahl an wiedergefundenen Ackerpunkten reduziert.

Um den Grund dafür zu finden, warum die Vorhersage für die Solarparkflächen versagt hat, wird die Konfusionsmatrix in Tabelle 5.18 betrachtet. Betrachtet man die Klassifizierung der Referenz, so sind für beide Klassen Punkte vorhanden, was eine fehlerhafte Klassifizierung ausschließt. Die Konfusionsmatrix zeigt, dass beide Klassen fast gänzlich der Umgebungsklasse zugeordnet worden sind. Ein Grund dafür könnten ähnliche Punkte in der Umgebungsklasse oder die Geländehöhe sein. Durch das Oversampling wurden zwar die Anzahl der Trainingsbeispiele erhöht, aber nicht die Abwechslung in den Beispielen für beide Klassen. Da die Beispiele für die Umgebung im Trainingsdatensatz per zufälligem Undersampling reduziert wurden, wurde eine große Variation in den Beispielen für diese Klasse beibehalten. Damit ergibt sich für die Solarpark Klassen nur ein kleines Fenster an Eigenschaften, in dem Punkte als solches klassifiziert werden. Liegen die Eigenschaften, außerhalb dieses Fenster, werden sie der Umgebung zu geordnet.

6.3 Empfehlung für die Praxis

6.3.1 Softwarealternativen

Beide Programme sind für die Bearbeitung von Ausreißern geeignet. Da CloudCompare bereits bei der Sichtung von Ausreißern als Viewer verwendet wird, ist die Empfehlung es auch zur Bearbeitung der Ausreißerpunkten zu verwenden. Durch die Feststellung das Punkte einfach gelöscht werden können, besteht keine Notwendigkeit der Klassifizierung zum Erhalt der synthetischen Punkteigenschaft. Abschließend ist anzuführen, dass die Software kostenlos erhältlich ist und durch ihre Verwendung Lizenzkosten gespart werden können.

6.3.2 Automatisierte Ausreißerdetektion

Um die Relevanz der zusätzlichen Features zu steigern, wird empfohlen, für die Trainingsdaten weiter größere Radien zu berechnen, um die besten Radien für das jeweilige Feature zu finden. Die im Experiment verwendeten Radien zeigen hier deutlich, dass bei der verwendeten Radien noch kein signifikant relevanter Radius ermittelt worden ist. Eine vorherige Vermaschung des DGMs und ein darauf folgender Mesh to Cloud (M2C-Vergleich) ermöglicht Höhendifferenzen mit Vorzeichen und könnte damit das Ergebnis weiter verbessern. Punkte die unter der Geländeoberfläche liegen, wären damit direkt als Ausreißer ausgemacht. Im Laufe der Bearbeitung ist zudem aufgefallen, dass es verschiedene Arten von Ausreißern gibt. Da verschiedene Ausreißerformen verschiedene Eigenschaften besitzen, könnte man für jeden Fehlertyp ein eigenes Feature-Set bestimmen. Eine Begrenzung der Vorhersagegebiete über Polygone aus der Tatsächlichen Nutzung könnten die Ergebnisse weiter verbessern.

Um den aktuellen Arbeitsablauf zu beschleunigen, könnte die automatisierte Ausreißerdetektion als Visualisierungshilfe dabei helfen, Ausreißer schneller zu identifizieren. Des weiteren könnte sie dazu beitragen Fehler zu erkennen, die bei der optischen Kontrolle nicht sichtbar sind. Ein Beispiel ist die Detektion von zuvor nicht gesehenen Ausreißern auf Testgebiet 5 (Abb.5.38).

7 Zusammenfassung

Die Ermittlung der Rahmenbedingungen für den Einsatz einer alternativen Software zur Ausreißerbearbeitung hat klare Kriterien für die Recherche hervorgebracht. Anhand dieser konnten während der Recherche mehrere Softwareprodukte ausgemacht werden, die die aktuelle Software ersetzen können. Darauf wurden zwei vielversprechende Produkte anhand der erarbeiteten Kriterien auf ihre Funktionalität getestet. Die beiden getesteten Produkte erfüllten alle Anforderung zur Nutzung im Arbeitsablauf der Ausreißerbearbeitung. Jede bietet dabei individuelle Vorteile und Nachteile, die bei der finalen Auswahl abgewogen werden müssen.

Zur automatisierten Detektion von Ausreißer wurden zunächst Referenzdaten für das Training, Validieren und Testen des Klassifikators erzeugt. Die Referenzdaten wurden dabei aus DOM-Kacheln mit Ausreißern erzeugt, welche vom LDBV Referat 85 übergeben wurden. Nach der Vorbereitung der Referenzdaten erfolgte die Berechnung zusätzlicher Features mit unterschiedlichen Radien und die Extraktion sowie Zusammenfassung der Beispiele und deren Eigenschaften zu Traings-/Validierungsdatensätzen. Die Beispielanzahlen der T/V-Datensätze wurden danach mit Under- und Oversampling angeglichen, so dass für jede Klasse die selbe Anzahl an Beispielen vorhanden waren. Mit den angeglichenen Datensätzen wurden dann Klassifikatoren trainiert und Feature-Rankings erstellt. Anhand dieses Feature Rankings wurden Kreuzvalidierung mit verschiedenen Hyperparametern auf den Trainingsdatensätze mit den besten 10 Features durchgeführt, um die besten Hyperparameter für das Trainieren der finalen Klassifikatoren zu erhalten. Nach dem Training der Klassifikatoren mit den besten Features und Hyperparametern wurden die Klassifikatoren auf die zuvor erstellten Testkacheln getestet. Um die Vorhersagen beurteilen zu können, wurden Konfusionsmatrizen und die daraus berechenbaren Werte für Precision, Recall und der daraus resultierenden F1-Wert abgeleitet.

Die Ergebnisse der Feature-Rankings zeigen, dass die Farbwerte und die Höhendifferenz zwischen DOM und DGM, die höchste Relevanz haben. Alle zusätzlichen Features haben dazu im Vergleich nur eine geringe Relevanz. Betrachtet man die zusätzlichen Features mit der höchsten Relevanz, erkennt man, das diese meist auch den größten berechneten Radius haben.
Fasst man die Ergebnisse der Detektion von Ackerausreißer zusammen, kann man den Einfluss der relevantesten Features erkennen. Ausreißer, die keine hellbraune Farbe haben, werden nicht erkannt. Bereiche ohne Ausreißer, die sich stärker vom DGM unterscheiden und eine hellbraune Farbe haben, werden als Ausreißer vorhergesagt.
Betrachtet man die Ergebnisse der Detektion von Solarparkausreißern, kann man denselben Einfluss erkennen. Umgebungspunkte, die auf derselben Höhe wie die fehlerfreien Solarparkpunkte liegen und derselben Farbe entsprechen, werden als Solarparkpunkte

erkannt.
Bei der Kombination beider Datensätze ist aufgefallen, dass die Performance einer Klasse von der Abwechslung in deren Beispiele abhängen kann. Eine zufällige Vervielfältigung erhöhte zwar die Beispielanzahl, ändert aber nichts an dem Spektrum, welches abgedeckt wird. Dadurch werden Punkte, die eigentlich zu einer Klasse gehören, aber nicht genau im Spektrum liegen, einer Klasse zugewiesen, die dieses Spektrum abdeckt.

Abschließend kann man feststellen, dass Ausreißerpunkte mit Hilfe einer überwachten Klassifizierung vorhergesagt werden können. Zur Verbesserung der Performance könnten in weiteren Forschungen die in Kapitel 6.3.2 gegeben Empfehlungen getestet und evaluiert werden.

Literaturverzeichnis

[AdV 2019] AdV: *Leitfaden zur Qualitätssicherung von True Orthophotos (TrueDOP)*. https://www.adv-online.de/AdV-Produkte/Standards-und-Produktblaetter/Standards-der-Geotopographie/binarywriterservlet?imgUid=1e220307-0b71-ee71-7657-80b6a757628a&uBasVariant=11111111-1111-1111-1111-111111111111. 2019. – Aufgerufen: 10.10.2023

[CloudCompare 2023] CloudCompare: *Command line mode*. https://www.cloudcompare.org/doc/wiki/index.php/Command_line_mode. 2023. – Aufgerufen: 20.10.2023

[CloudCompare 2024] CloudCompare: *Poisson Surface Reconstruction (plugin)*. https://www.cloudcompare.org/doc/wiki/index.php/Poisson_Surface_Reconstruction_(plugin). 2024. – Aufgerufen: 27.02.2024

[LDBV 2024a] LDBV: *Bayernbefliegung*. https://www.ldbv.bayern.de/produkte/luftbild/bayernbefliegung.html. 2024. – Aufgerufen: 20.01.2024

[LDBV 2024b] LDBV: *Digitale Oberflächenmodelle*. https://www.ldbv.bayern.de/produkte/3dprodukte/dom.html. 2024. – Aufgerufen: 26.01.2024

[LDBV 2024c] LDBV: *Das Luftbild wird dreidimensional*. https://www.ldbv.bayern.de/produkte/3dprodukte/dom.html. 2024. – Aufgerufen: 20.01.2024

[imbalanced learn 2023a] learn imbalanced: *Over-sampling*. https://imbalanced-learn.org/stable/over_sampling.html. 2023. – Aufgerufen: 15.11.2023

[imbalanced learn 2023b] learn imbalanced: *Under-sampling*. https://imbalanced-learn.org/stable/under_sampling.html. 2023. – Aufgerufen: 15.11.2023

[scikit learn 2023] learn scikit: *RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. 2023. – Aufgerufen: 20.10.2023

[scikit learn 2024] learn scikit: *sklearn model selection GridSearchCV*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. 2024. – Aufgerufen: 10.1.2024

[TerraSolid 2023a] TerraSolid: *Creating a Surface Model*. https://terrasolid.com/guides/tmodel/create-surface.html. 2023. – Aufgerufen: 10.10.2023

[TerraSolid 2023b] TerraSolid: *Display Surface toolbox*. `https://terrasolid.com/guides/tmodel/tbox_displaysurface.html`. 2023. – Aufgerufen: 10.10.2023

A Quellcode

A.1 Batchdatei-Code zur Berechnung zusätzlicher Features CloudCompare Commandline Befehle

```
1
2 @echo off
3 set local EnableDelayedExpansion
4 set Input=X:\...
5 set Output=X:\...
6
7
8 for %%f in ( "%Input%"\* ) do ("F:\Programme\CloudCompare\cloudcompare
    .exe" -SILENT -AUTO_SAVE OFF -C_EXPORT_FMT ASC -ADD_HEADER -O -
    GLOBAL_SHIFT AUTO %Input%\%%~nxf -FEATURE SUM_OF_EIGENVALUES 1 -
    FEATURE OMNIVARIANCE 1 -FEATURE EIGENTROPY 1 -FEATURE ANISOTROPY 1
    -FEATURE PLANARITY 1 -FEATURE LINEARITY 1 -FEATURE PCA1 1 -FEATURE
    PCA2 1 -FEATURE SURFACE_VARIATION 1 -FEATURE SPHERICITY 1 -FEATURE
    VERTICALITY 1 -FEATURE EIGENVALUE1 1 -FEATURE EIGENVALUE2 1 -
    FEATURE EIGENVALUE3 1 -DENSITY 1 -TYPE VOLUME -DENSITY 1 -TYPE KNN
    -CURV NORMAL_CHANGE 1 -FEATURE SUM_OF_EIGENVALUES 2 -FEATURE
    OMNIVARIANCE 2 -FEATURE EIGENTROPY 2 -FEATURE ANISOTROPY 2 -FEATURE
     PLANARITY 2 -FEATURE LINEARITY 2 -FEATURE PCA1 2 -FEATURE PCA2 2 -
    FEATURE SURFACE_VARIATION 2 -FEATURE SPHERICITY 2 -FEATURE
    VERTICALITY 2 -FEATURE EIGENVALUE1 2 -FEATURE EIGENVALUE2 2 -
    FEATURE EIGENVALUE3 2 -DENSITY 2 -TYPE VOLUME -DENSITY 2 -TYPE KNN
    -CURV NORMAL_CHANGE 2 -FEATURE SUM_OF_EIGENVALUES 3 -FEATURE
    OMNIVARIANCE 3 -FEATURE EIGENTROPY 3 -FEATURE ANISOTROPY 3 -FEATURE
     PLANARITY 3 -FEATURE LINEARITY 3 -FEATURE PCA1 3 -FEATURE PCA2 3 -
    FEATURE SURFACE_VARIATION 3 -FEATURE SPHERICITY 3 -FEATURE
    VERTICALITY 3 -FEATURE EIGENVALUE1 3 -FEATURE EIGENVALUE2 3 -
    FEATURE EIGENVALUE3 3 -DENSITY 3 -TYPE VOLUME -DENSITY 3 -TYPE KNN
    -CURV NORMAL_CHANGE 3 -SET_ACTIVE_SF 3 -FILTER_SF 1 7 -SAVE_CLOUDS
    FILE %Output%\konvertiert_%%~nf)
9
10
11 rem change the data type from asc to xyz for later import into python
12
13 pushd X:\Studium\Bachelor-Arbeit\_aktuell\Training_Test_Validierung\
    Training\ACKER_Auswahl_Trainings_Daten\BATCH\output
14 for /f "delims=" %%i in ('dir /a-d/b/s *.asc') do ren "%%i" "%%~ni.xyz
    "
15 popd
16
17 rem artificial pause to be able to read the log
18
```

```
19 pause
```

A.2 Python-Beispiel-Code zur automatisierten Ausreißerdetektion

A.2.1 Import Trainingsdaten, NaN-Handling, Undersampling, Export resultierender Datensatz

```
#Automatisierte Ausreisserdetektion fuer digitale Oberflaechenmodelle
#Author: Hubertus Vier

#Basierend auf den Code von: Ph.D. Florent Poux
#Titel der Veroeffentlichung: 3D Machine Learning 201 Guide: Point Cloud Semantic Segmentation
#Datum der Veroeffentlichung: 29.06.2022
#URL: https://towardsdatascience.com/3d-machine-learning-course-point-cloud-semantic-segmentation-9b32618ca5df



#Import der benoetigten Python-Bibliotheken
print('-Import der benoetigten Python-Bibliotheken')

import sys
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler

import matplotlib
import matplotlib.pyplot as plt

from imblearn.under_sampling import RandomUnderSampler

print('- alle notwendigen Bibliotheken wurden importiert')
print('================================')


#1. Import der Trainingsdaten, NaN-Handling, Undersampling, Export des resultierenden Datensatzes

#Import des Trainingsdatensatzes mit
print('Import der Trainingsdaten')

#Angabe Dateipfad
data_folder='DATA/'
#Dateiname
dataset1="Trainingsdaten_Acker.xyz"

#Import der PW datei um CSV Parser
pcd1=pd.read_csv(data_folder+dataset1, delimiter=' ')
print("- Datensatz eingelesen")
```

```
45
46 #Einfacher Check, ob auch wirklich Daten geladen worden sind
47 print("- Speichergroesse des eigenlesenen Trainings Datensatzes: ",
       sys.getsizeof(pcd1)/(1024*1024),'mb')
48
49 #NAN-Handling/ Entfernen von fehlenden Daten/ fehlerhaften
       Datensaetzen
50 pcd1.dropna(inplace=True)
51 print("- fehlerhafte Datensaetze eleminiert (NaN-Handling)")
52
53 #Extraktion der vorhandenen Klassen
54 labels1=pcd1['Classification']
55 print("- Labels extrahiert")
56
57 #Feature-Extraktion mit Koordinaten
58 features1=pcd1
59
60 #Feautres die nicht fuer das Training des Klassfikators verwendet
       werden sollen
61 columns_to_exclude = ['//X','Y','Z']
62 print("- ueberblick ueber die eingelesen Features:")
63 pcd1_t = pcd1.drop(columns=columns_to_exclude)
64 pcd1_t.columns.to_list()
65 #print(pcd1_t)
66
67 print('================================')
68
69 #Anzahlsausgleich der Klassen ueber Undersampling
70 print('Anzahlsausgleich der Klassen ueber Undersampling')
71
72 X=pcd1_t
73 y=labels1
74
75 print("- Start Undersampling")
76 rus = RandomUnderSampler(random_state=0)
77 X_resampled, y_resampled = rus.fit_resample(X, y)
78 print('- Ende Undersampling')
79
80
81 #Export the labeled dataset
82 print('- Start Export Datensatz aus Undersampling')
83 X_resampled['Classification']=y_resampled
84 result_folder="./DATA/RESULTS/"
85 X_resampled.to_csv(result_folder+dataset1.split(".")[0]+"
       _Klassen_ausgeglichen.xyz", index=None, sep=' ')
86 print('-Ende Export Datensatz aus Undersampling')
87
88
89 #Ausgabe der Punktanzahl der ausgeglichenen Klassen
90 print('-Anzahl der Beispiele fuer die vorhandenen Klassen:')
91 print(X_resampled["Classification"].value_counts())
92
93 #Graphische Visualisierung der Ausgeglichenen Klassen
```

```
X_resampled.groupby('Classification').size().plot(kind='pie',
                                        y = "v1",
                                        label = "Type",
                                        autopct='%1.1f%%')

print('================================')
```

A.2.2 Feature-Ranking

```
#2.Erstellung eines Feature-Rankings fuer die Trainingsdaten um zu sehen, welche Features den groessten Einfluss
# auf das Ergebnis haben oder die am besten zur Klassifizierung beitragen.

import sys
import pandas as pd
import numpy as np

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import matplotlib.pyplot as plt

import pickle

from imblearn.under_sampling import RandomUnderSampler

# Angabe der Dateipfade
data_folder = 'DATA/RESULTS/'
result_folder="./DATA/RESULTS/"

# Dateiname
dataset1 = "Trainingsdaten_Acker_Klassen_ausgeglichen.xyz"

# Import der undersampeled Trainingsdaten
pcd1 = pd.read_csv(data_folder + dataset1, delimiter=' ')
print("- Datensaetze eingelesen")

# Einfacher Check, ob auch wirklich Daten geladen worden sind
print("- Groesse des eingelesenen Datensatzes: ", sys.getsizeof(pcd1) / (1024 * 1024), 'mb')

# NAN-Handling/ Entfernen von fehlenden Daten/ fehlerhaften Datensaetzen
pcd1.dropna(inplace=True)
print("- fehlerhafte Datensaetze eliminiert")

# Zuweisung der vorhandenen Klassen
labels1 = pcd1['Classification']
```

```
39 print("- Labels extrahiert")
40
41 # Festlegung welche Features fuers trainieren verwendet werden
42 features1 = pcd1.drop(columns='Classification')
43 print("- Features extrahiert")
44
45
46 # Erstellung eines Klassifikators mit den Trainingsdaten
47 print('- Start Training des RF-Klassifikators')
48 forest = RandomForestClassifier(n_estimators=100, n_jobs=-1)
49 forest.fit(features1, labels1)
50 print('- Ende Training des RF-Klassifikators')
51
52 #Export des Klassifikators
53 print('- Export des Feature-Ranking-Klassifikators')
54 pickle.dump(forest, open(result_folder+"rf_classifier_FR.vier", 'wb'))
55 print("- Klassifikator gespeichert")
56
57 # Abruf der Feature-Relevanzen
58 feature_importances = forest.feature_importances_
59
60 # Sortierung der Feature-Wichtigkeit in absteigender Reihenfolge
61 indices = (-feature_importances).argsort()
62
63 feature_names = features1.columns.to_list()
64 feature_importances = dict(zip(feature_names, feature_importances))
65
66 # Berechnung der Standardabweichung der einzelnen Features
67 std = np.std([tree.feature_importances_ for tree in forest.estimators_
    ], axis=0)
68
69 # Sortierung der Feature-Relevanzen
70 sortierte_Relevanz = sorted(feature_importances.items(), key=lambda x:
     x[1], reverse=True)
71
72 # Extraktion der Features mit ihren Relvanzwerten
73 sortierte_Namen, sortiertze_Relevanz = zip(*sortierte_Relevanz)
74
75 # Darstellung - Geordnete Feature-Wichtigkeit mit dazugehoeriger
    Standardabweichung
76 print('- Erstelltes Feature-Ranking:')
77 plt.figure(figsize=(20, 6))
78 plt.bar(range(len(sortiertze_Relevanz)), sortiertze_Relevanz, align='
    center', yerr=std[indices])
79 plt.xticks(range(len(sortiertze_Relevanz)), sortierte_Namen, rotation
    =90)  # Rotate x-axis labels for better visibility
80 plt.xlabel('Feature Namen')
81 plt.ylabel('Feature Wichtigkeit')
82 plt.title('Geordnete Feature-Wichtigkeit mit dazugehoeriger
    Standardabweichung')
83 plt.tight_layout()
84 plt.show()
85
```

```
top_k = 20  # Anzahl der besten Features, die angezeigt werden sollen
print("Die besten", top_k, "Features sind:")
for feature in sortierte_Namen[:top_k]:
    print("-", feature)

# Erstellen eines DataFrames mit den sortierten Feature-Namen und deren Wichtigkeiten sowie der Standardabweichung
feature_table = pd.DataFrame({
    'Feature Namen': sortierte_Namen,
    'Feature Wichtigkeit': sortiertze_Relevanz,
    'Standardabweichung': std[indices]
})

# Ausgabe des DataFrames als Tabelle
print(feature_table)

result_folder = "./DATA/RESULTS/"
file_path = result_folder + "Feature-Ranking_Acker.xlsx"

# Speichern des DataFrames als Excel-Datei
feature_table.to_excel(file_path, index=False)
```

A.2.3 Hyperparameter Tuning und Training des Klassifikators

```

#3.Training des Klassifikators mit den besten Features und Hyperparameter Tuning mit Validierungsdaten + Klassifikator Erstellung

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sys
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
import pickle

#Start der Stoppuhr
start_time = time.time()

# Angabe der Dateipfade
data_folder = 'DATA/RESULTS/'
result_folder = "./DATA/RESULTS/"

# Dateiname
dataset1 = "Trainingsdaten_Acker_Klassen_ausgeglichen.xyz"

# Import der undersampeled Trainingsdaten
pcd1 = pd.read_csv(data_folder + dataset1, delimiter=' ')
```

```
26 print("- Datensaetze eingelesen")
27
28 # Einfacher Check, ob auch wirklich Daten geladen worden sind
29 print("- Groesse des eingelesenen Datensatzes: ", sys.getsizeof(pcd1)
       / (1024 * 1024), 'mb')
30
31 # NAN-Handling/ Entfernen von fehlenden Daten/ fehlerhaften
       Datensaetzen
32 pcd1.dropna(inplace=True)
33 print("- fehlerhafte Datensaetze eliminiert")
34
35 # Zuweisung der vorhandenen Klassen
36 labels1 = pcd1['Classification']
37 print("- Labels extrahiert")
38
39 # Festlegung welche Features fuers trainieren verwendet werden
40 features1 = pcd1[['Intensity','C2C_absolute_distances[<50]','
       Normal_change_rate_(1.8)', 'Surface_variation_(1.8)', 'Sphericity_
       (1.8)' ,'3rd_eigenvalue_(1.8)','Synthetic_Flag']]
41 print("- Features extrahiert")
42
43 # Hyperparameter-Tuning mit GridSearch (sehr zeitaufwendig)
44 param_grid = {
45     'n_estimators': [10, 20, 50],
46     'max_depth': [None, 10, 20],
47     #'min_samples_split': [2, 5, 10],
48     #'min_samples_leaf': [1, 2, 4]
49 }
50
51 print('-Start GridSearch')
52 rf = RandomForestClassifier()
53 grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=2,
       n_jobs=-1)
54 grid_search.fit(X_train, y_train)
55 print('-Ende GridSearch')
56
57 # Ergebnisse der Kreuzvalidierung ausgeben
58 print("GridSearchCV Ergebnisse:")
59 cv_results = grid_search.cv_results_
60 for mean_score, params in zip(cv_results["mean_test_score"],
       cv_results["params"]):
61     print(mean_score, params)
62
63 # Beste Parameter
64 beste_Parameter = grid_search.best_params_
65 n = beste_Parameter['n_estimators']
66 md = beste_Parameter['max_depth']
67
68 # Ergebnisse der Kreuzvalidierung als Excel ausgeben
69 GS_CV_results = grid_search.cv_results_
70 Ergebnisse = pd.DataFrame(GS_CV_results)
71
72 # Ergebnisse als Excel-Datei speichern
```

```
Ergebnisse.to_excel("grid_search_results.xlsx", index=False)
print("GridSearchCV Ergebnisse als Excel-Tabelle ausgegeben.")

#Stop der Stoppuhr
elapsed_time = time.time() - start_time

print(f"Fuer die Kreuzvalidierung mit GirdSeachCV wurde folgende Zeit gebraucht : {elapsed_time/60:.3f} minutes")


#Training des Klassifikators mit der besten Hyperparameter Kombination
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sys
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
import pickle

# Angabe der Dateipfade
data_folder = 'DATA/RESULTS/'
result_folder = "./DATA/RESULTS/"

# Dateiname
dataset1 = "Trainingsdaten_Acker_Klassen_ausgeglichen.xyz"

# Import der undersampeled Trainingsdaten
pcd1 = pd.read_csv(data_folder + dataset1, delimiter=' ')
print("- Datensaetze eingelesen")

# Einfacher Check, ob auch wirklich Daten geladen worden sind
print("- Groesse des eingelesenen Datensatzes: ", sys.getsizeof(pcd1) / (1024 * 1024), 'mb')

# NAN-Handling/ Entfernen von fehlenden Daten/ fehlerhaften Datensaetzen
pcd1.dropna(inplace=True)
print("- fehlerhafte Datensaetze eliminiert")

# Zuweisung der vorhandenen Klassen
labels1=pcd1['Classification']
print("-Labels extrahiert")

#Festlegung welche Features fuers trainieren verwendet werden
features1=pcd1[['Intensity','C2C_absolute_distances[<50]','Normal_change_rate_(1.8)', 'Surface_variation_(1.8)', 'Sphericity_(1.8)' ,'3rd_eigenvalue_(1.8)','Synthetic_Flag']] #,'Volume_density_(r=1.5)','Surface_density_(r=1.5)','Verticality_(1.5)','Number_of_neighbors_(r=1.5)'
print("-Features extrahiert")

```

```
#['R','G','B','Intensity','Return_Number','Number_Of_Returns','
    Scan_Direction_Flag','EdgeOfFlightLine','Classification','
    Synthetic_Flag','Keypoint_Flag','Withheld_Flag','Scan_Angle_Rank','
    User_Data','Point_Source_ID','C2C_absolute_distances','Roughness_
    (1.2)','1st_order_moment_(1.2)','Mean_curvature_(1.2)','
    Gaussian_curvature_(1.2)','Normal_change_rate_(1.2)','
    Number_of_neighbors_(r=1.2)','Surface_density_(r=1.2)','
    Volume_density_(r=1.2)','Eigenvalues_sum_(1.2)','Sphericity_(1.5)
    ','1st_eigenvalue_(1.5)','2nd_eigenvalue_(1.5)','3rd_eigenvalue_
    (1.5)']

print("-Start: Klassifikator wird erstellt")
forest = RandomForestClassifier(n_estimators = n, max_depth=md n_jobs
    =-1)
forest.fit(features1, labels1)
print("-Ende:Klassifikator ist erstellt")

print("-Start: Export Klassifikator")
result_folder="./DATA/RESULTS/"

#Export des Klassifikators
pickle.dump(forest, open(result_folder+"Klassifikator_Acker.vier", 'wb
    '))
print("-Ende: Klassifikator exportiert")
```

A.2.4 Vorhersage auf Testdaten

```
#4. Vorhersage auf Testdaten
import pickle
import sys
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
import matplotlib
import matplotlib.pyplot as plt

#Start der Stoppuhr
start_time = time.time()

#Angabe Dateipfad Daten
data_folder='DATA/'

#Angabe Dateipfad Resultate
result_folder="./DATA/RESULTS/"

#Dateiname
dataset2="32725_5416_20_Test.xyz"
pcd2=pd.read_csv(data_folder+dataset2,delimiter=' ')
pcd2.dropna(inplace=True)
```

```
25
26
27 #Feature Auswahl
28 features2=pcd2[['Intensity','C2C_absolute_distances[<50]','
      Normal_change_rate_(1.8)', 'Surface_variation_(1.8)', 'Sphericity_
      (1.8)' ,'3rd_eigenvalue_(1.8)','Synthetic_Flag']] #,'
      Volume_density_(r=1.5)','Surface_density_(r=1.5)','Verticality_
      (1.5)','Number_of_neighbors_(r=1.5)'
29 features_scaled2 = MinMaxScaler().fit_transform(features2)
30
31 labels2=pcd2[['Classification']]
32
33 #Extraktion der Koordianten fuer Plot
34 coo_temp=pcd2[['//X','Y']]
35 coo_temp=MinMaxScaler().fit_transform(coo_temp)
36
37
38
39 model_name="Klassifikator_Acker.vier"
40 loaded_model = pickle.load(open(result_folder+model_name, 'rb'))
41 print("-Start Vorhersage")
42 #rf_predictions = loaded_model.predict(features_scaled2)
43 rf_predictions = loaded_model.predict(features2)
44 print("-Ende Vorhersage")
45
46
47 print(classification_report(labels2, rf_predictions, target_names=['
      Acker','Umgebung','Ackerausreisser']))
48
49
50 pcd2['Classification']=rf_predictions
51 result_folder="./DATA/RESULTS/"
52 pcd2[['//X','Y','Z','R','G','B','Classification']].to_csv(
      result_folder+dataset2.split(".")[0]+"_Klassifiziert.xyz", index=
      None, sep=';')
53
54 # Beispielwerte fuer labels_soll und labels_ist
55 labels_soll = labels2['Classification'].values.ravel()
56 labels_ist = rf_predictions.ravel()
57
58 # Die Klassennamen aendern
59 class_mapping = {1: 'Acker', 6: 'Umgebung', 7: 'Ackerausreisser'}
60 labels_soll_mapped = [class_mapping.get(label, label) for label in
      labels_soll]
61 labels_ist_mapped = [class_mapping.get(label, label) for label in
      labels_ist]
62
63 # Die pd.crosstab-Funktion verwenden
64 cm = pd.crosstab(labels_soll_mapped, labels_ist_mapped, margins=True).
      T  # Invertieren von Zeilen und Spalten
65
66 # Die Spalten und Zeilen der Tabelle umbenennen
67 cm.rename(columns={'All': 'c_sum'}, index={'All': 'r_sum'}, inplace=
```

```
    True)
print(cm)

#Visualisierung der Ergebnisse
#1 Darstellung der Referenz
#2 Darstellung der Vorhersage
#3 Darstellung der Differenz Referenz und Vorhersage

from sklearn.preprocessing import MinMaxScaler
labels_scaled = MinMaxScaler().fit_transform(labels2)
predictions=pcd2[['Classification']]
rf_predictions_scaled = MinMaxScaler().fit_transform(predictions)


#Erstellung einer Figure/ Darstellung mit einer Zeile und 3
    Darstellungen
print("Start Ploterstellung")
fig, axs = plt.subplots(1, 3, figsize=(20,5))

axs[0].scatter(coo_temp[:,0], coo_temp[:,1], c = labels_scaled, s
    =0.05)
axs[0].set_title('Referenz')
axs[1].scatter(coo_temp[:,0], coo_temp[:,1], c = rf_predictions_scaled
    , s=0.05)
axs[1].set_title('Vorhersage - alle Features ')
axs[2].scatter(coo_temp[:,0], coo_temp[:,1], c = labels_scaled-
    rf_predictions_scaled, cmap = plt.cm.rainbow, s=0.5*(labels_scaled-
    rf_predictions_scaled))
axs[2].set_title('Unterschied zwischen Referenz und Vorhersage')

#Stop der Stoppuhr
elapsed_time = time.time() - start_time

print(f"Fuer die Klassifizierung, Ausgabe und Visualisierung der
    Ergebnisse wurde folgende Zeit gebraucht : {elapsed_time/60:.3f}
    minutes")
```