

Multivariate Zeitreihenanalyse für Klassifikation und Anomalieerkennung auf SITS-Daten

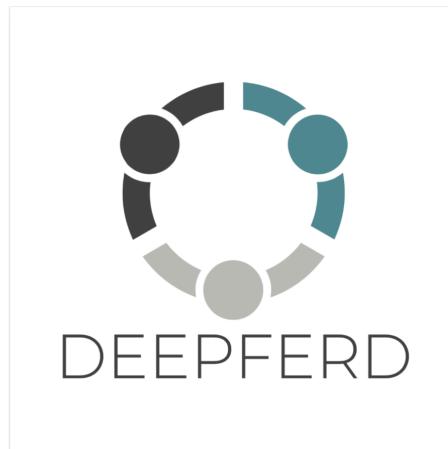


Abbildung 1: Logo von DeepFerd

Hochschule Karlsruhe
Anwendungsprojekt 2

Verfasst durch:
Evelyn Rempel
Alexandru Cozma
Rafael Riesle

31. Oktober 2025

Inhaltsverzeichnis

1 Einleitung	3
1.1 Projektvorstellung	3
1.2 Hintergrund und Motivation	3
1.3 Zielsetzung des Projekts	4
2 Projektbeschreibung	5
2.1 Beschreibung des Themas	5
2.2 Ausgangssituation oder Problemstellung	5
2.3 Relevanz des Projekts	6
3 Projektdurchführung	7
3.1 Ressourcenplanung	7
3.2 Arbeitsablauf	7
3.3 Methoden und Vorgehensweise	7
4 Datenanalyse	8
4.1 Explorative Datenanalyse Trainingsdatensatz	8
4.2 Explorative Datenanalyse Validierungsdatensatz	10
4.3 Erkennen von kranken Bäumen	11
5 Prozessstruktur und Datenverarbeitungsschritte	14
5.1 Beschreibung der Prozessstruktur	14
5.2 Datenvorverarbeitung	15
5.3 Datenverarbeitung	15
5.4 Data Augmentation zur Verbesserung der Trainingsdaten	17
5.4.1 Problemstellung und Motivation	17
5.4.2 Methodik der Augmentierung	18
5.4.3 Ergebnisse	20
5.4.4 Zusammenfassung	21
6 Modell Ergebnisse	22
6.1 Generelles Vorgehen	22
6.2 Modell Vergleich	22
6.3 Baseline Modell	23
6.4 Ensemble-Modell	23
6.5 LSTM Modell	24
6.6 PYTS-Modell	25

6.7	Einfluss der Verarbeitungsschritte auf das Modellergebnis	26
6.8	Feature-Analyse	28
6.8.1	SHAP-Analyse für das Baseline-Modell	28
6.8.2	Feature Analyse für LSTM (mit Integrated Gradients)	30
6.9	Datenreduktion durch Analyse von Zeitintervallen	31
6.9.1	Jahresbasierte Datenreduktion	31
6.9.2	Monatsbasierte Datenreduktion	32
6.9.3	Analyse um Disturbance Year	33
7	Probleme und Lösungen	35
7.1	Aufgetretene Schwierigkeiten	35
8	Fazit und Ausblick	36
8.1	Zusammenfassung der wichtigsten Erkenntnisse	36
8.2	Reflexion über den Projektverlauf	36
8.3	Verbesserungsvorschläge	36
9	Anhang	37
9.1	Abbildungen	37
9.2	Leistungsbeschreibung	41
9.2.1	Evelyn Rempel	41
9.2.2	Alexandru Cozma	42
9.2.3	Rafael Riesle	42
9.2.4	Gesamt	43
9.3	Quellen und Literatur	43

Kapitel 1

Einleitung

Die fortschreitende Verfügbarkeit hochauflösender Satellitendaten eröffnet neue Möglichkeiten zur detaillierten Analyse von Vegetationsdynamiken. Insbesondere multispektrale Sensoren wie jene der Sentinel-2-Mission liefern umfangreiche Daten im sichtbaren und infraroten Spektralbereich, die Rückschlüsse auf physiologische und strukturelle Eigenschaften von Vegetation erlauben. Die Untersuchung der spektralen Signaturen über die Zeit hinweg. Satellite Image Time Series (SITS) ermöglicht eine präzisere Erfassung und Klassifikation von Baumarten sowie eine bessere Beurteilung von Vegetationszuständen.

1.1 Projektvorstellung

Das vorliegende Projekt befasst sich mit der automatisierten Auswertung von Satelliten-Zeitreihen zur Identifikation und Differenzierung von Baumarten in deutschen Waldgebieten. Grundlage bilden Sentinel-2-Aufnahmen mit einer räumlichen Auflösung von 10 Metern und einer zeitlichen Wiederholrate von etwa acht Tagen. In Kombination mit Referenzdaten aus der Bundeswaldinventur (BWI) 2012 und ergänzenden Informationen der BWI 2022 werden zeitlich aufgelöste Vegetationsprofile für einzelne Baumstandorte erstellt, um daraus robuste Klassifikationsmodelle zu entwickeln.

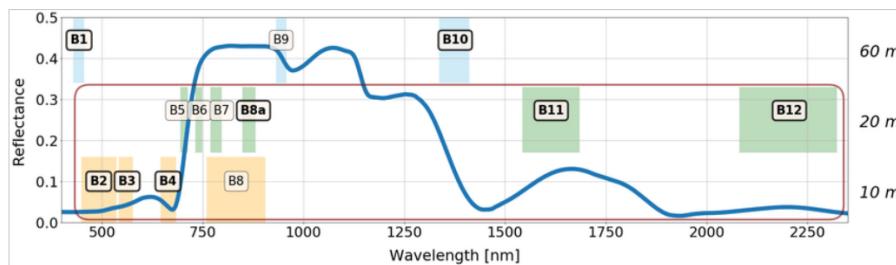


Abbildung 1.1: Spektralbänder der Sentinel-2-Daten

1.2 Hintergrund und Motivation

Wälder stellen ein zentrales ökologisches und ökonomisches Gut dar. Ihre nachhaltige Bewirtschaftung erfordert genaue Kenntnisse über Baumartenzusammensetzung,

Vitalität und Veränderungen im Zeitverlauf. Traditionelle Erhebungsmethoden wie terrestrische Kartierungen oder Luftbildinterpretationen sind jedoch zeit- und kostenintensiv. Fernerkundungsbasierte Verfahren bieten hier eine effiziente Alternative, da sie großflächige, wiederkehrende und objektive Beobachtungen ermöglichen. Die Kombination von Satelliten-Zeitreihen und forstlichen Inventurdaten verspricht daher eine deutliche Verbesserung der automatisierten Waldkartierung und des Monitorings.

1.3 Zielsetzung des Projekts

Ziel des Projekts ist die Entwicklung und Evaluierung eines datengetriebenen Ansatzes zur Klassifikation von Baumarten anhand von Satellite Image Time Series (SITS). Dabei sollen spektrale und zeitliche Merkmale aus den Sentinel-2-Bändern (1.1), insbesondere den Red-Edge- und SWIR-Bereichen, genutzt werden, um Unterschiede zwischen Nadel- und Laubbaumarten zu identifizieren. Gleichzeitig wird untersucht, inwiefern Datenlücken durch Wolken, Schnee oder Schatten die Modellgüte beeinflussen und welche Vorverarbeitungsschritte erforderlich sind, um stabile Zeitreihenmodelle zu erzeugen.

Kapitel 2

Projektbeschreibung

Das Projekt umfasst mehrere aufeinander aufbauende Schritte:

1. **Datenaufbereitung:** Extraktion und Vorverarbeitung der Sentinel-2-Zeitreihen (Bänder B5–B8A, B11, B12) pro Baumposition aus der BWI.
2. **Explorative Datenanalyse:** Visualisierung der Kernelemente des Datensatzes sowie Analyse der Merkmale zur Gewinnung von Erkenntnissen.
3. **Feature Engineering:** Berechnung spektraler Indizes (z. B. NDVI, NDWI) und Ableitung zeitlicher Merkmale wie Saisonalität und Phänologie.
4. **Modellierung:** Anwendung und Vergleich verschiedener Klassifikationsalgorithmen (z. B. Random Forest, Gradient Boosting, Deep Learning).
5. **Evaluation:** Analyse der Modellleistung unter besonderer Berücksichtigung unausgeglichenener Klassenverteilungen und Datenlücken.

2.1 Beschreibung des Themas

Das zentrale Untersuchungsobjekt sind die aus Sentinel-2-Daten generierten Pixel-Zeitreihen, die für jeden Baumstandort eine kontinuierliche spektrale Entwicklung abbilden. Diese SITS repräsentieren charakteristische saisonale Muster und reflektieren Zustandsänderungen wie Blattentwicklung, Laubfall oder Trockenstress. Durch die Verknüpfung dieser Zeitreihen mit den Baumartendaten der Bundeswaldinventur entsteht eine Grundlage zur datenbasierten Differenzierung zwischen verschiedenen Vegetationstypen und Baumarten.

2.2 Ausgangssituation oder Problemstellung

Die Ausgangslage ist durch mehrere Herausforderungen gekennzeichnet:

- Die **ungleiche Verteilung** der Baumarten (z. B. Überrepräsentation von Fichte und Kiefer) führt zu **unausgeglichenen Trainingsdaten**.

- **Fehlerquellen** in den BWI-Referenzen (z. B. GNSS-Ungenauigkeiten, Mischpixel, Kartierungsfehler) erschweren die eindeutige Zuordnung.
- **Zeitreihenlücken** infolge von Wolken, Schnee oder Schatten beeinträchtigen die Datenqualität und erfordern komplexe Interpolations- oder Filterverfahren.

Diese Faktoren machen die zuverlässige Klassifikation zu einer anspruchsvollen Aufgabe, die robuste Datenverarbeitungs- und Modellierungsstrategien erfordert.

2.3 Relevanz des Projekts

Das Projekt besitzt hohe wissenschaftliche und praktische Relevanz. Einerseits leistet es einen Beitrag zur Weiterentwicklung von Methoden zur Fernerkundungsbasierten Vegetationsanalyse. Andererseits liefert es konkrete Ansätze für das forstliche Monitoring und die ökologische Zustandsbewertung. Durch die Automatisierung der Baumartenklassifikation können Inventurkosten reduziert, Aktualisierungszyklen verkürzt und die Grundlage für ein flächendeckendes Waldbeobachtungssystem geschaffen werden.

Kapitel 3

Projektdurchführung

3.1 Ressourcenplanung

Das Projekt wurde in Visual Studio Code (VS Code) unter Verwendung einer virtuellen Entwicklungsumgebung umgesetzt. Als Programmiersprache kam Python zum Einsatz. Für die Datenverarbeitung und -analyse wurden Bibliotheken wie Pandas und NumPy genutzt, während die Datenvisualisierung mithilfe von Matplotlib erfolgte. Darüber hinaus kamen Machine-Learning-Frameworks wie scikit-learn und PyTorch zur Anwendung.

3.2 Arbeitsablauf

Zu Beginn stand die Erkenntnisgewinnung im Vordergrund, die als Grundlage für nachgelagerte Modellierungs- und Datenreduktionsschritte diente. Die Explorative Datenanalyse (EDA) wurde durchgeführt, um ein tiefes Verständnis der Datengrundlage zu erlangen und darauf aufbauend geeignete Preprocessing- und Processing-Schritte zu entwickeln.

Im weiteren Verlauf wurde die Analyse der Modellergebnisse verfeinert, unter anderem durch SHAP-Analysen zur Erklärbarkeit der Modelle, um die bedeutendsten Eingangsvariablen zu identifizieren. Diese Schritte trugen wesentlich zur Verbesserung der Modellinterpretation und -validierung bei.

3.3 Methoden und Vorgehensweise

Aufgrund der dynamischen Projektentwicklung wurden regelmäßig Status-Meetings im Team abgehalten. Jedes Teammitglied erhielt klar definierte Arbeitspakete, die individuell bearbeitet und im Meeting präsentiert wurden. In den wöchentlichen Sitzungen wurden Zwischenergebnisse diskutiert, das weitere Vorgehen abgestimmt und Herausforderungen gemeinsam gelöst.

Zur strukturierten Organisation der Aufgaben kam ein Kanban-Board zum Einsatz, wodurch der aktuelle Bearbeitungsstand jederzeit transparent und nachvollziehbar war. Dieses Vorgehen förderte die effiziente Zusammenarbeit, verbesserte die Kommunikation und unterstützte die Projektkoordination.

Kapitel 4

Datenanalyse

4.1 Explorative Datenanalyse Trainingsdatensatz

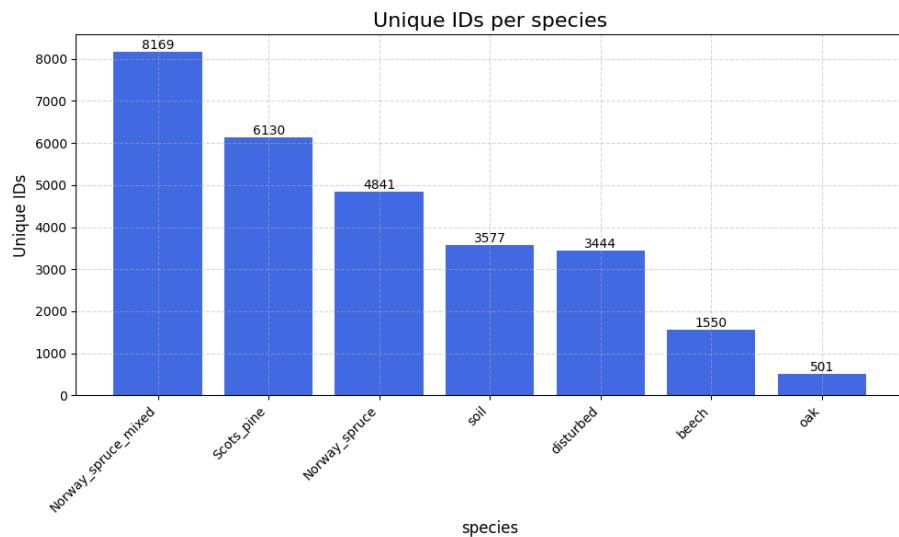


Abbildung 4.1: Anzahl eindeutiger IDs pro Baumart

Der zugrunde liegende Datensatz umfasst sieben verschiedene Klassen (4.1), die sich überwiegend in Laub- und Nadelbaumarten unterteilen. Zusätzlich sind „disturbed“-Flächen (gestörte Vegetation) sowie ein Label für Bodenwerte (Soil) enthalten.

Die Klassenverteilung zeigt eine deutliche Ungleichverteilung der Beobachtungen, wobei Nadelbäume signifikant häufiger vertreten sind als Laubbäume. Dieses Klassenungleichgewicht erforderte den Einsatz von Data-Augmentation-Strategien, um die Modellrobustheit zu erhöhen und eine Verzerrung zugunsten der häufigeren Klassen zu vermeiden.

Darüber hinaus unterscheiden sich die Anzahl der Zeitreiheneinträge pro ID stark (4.2) und entsprechen keiner Normalverteilung. Datensätze mit sehr wenigen Beobachtungen weisen einen geringen Informationsgehalt auf, wodurch keine verlässlichen Aussagen über zeitliche Muster getroffen werden können. Umgekehrt zeigten Datensätze mit sehr vielen Einträgen teilweise unregelmäßige Zeitschlüsse („Gaps“)

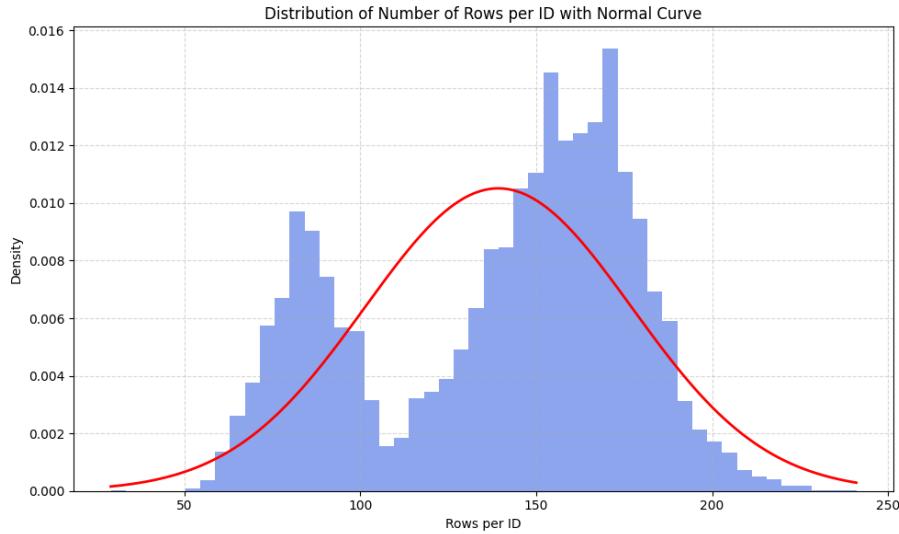


Abbildung 4.2: Verteilung: Anzahl Zeilen pro ID

und Cluster von Beobachtungen, die ebenfalls eine Verzerrung im Modelltraining verursachen können.

Zur Harmonisierung der Datenbasis wurden die Zeitreihen mithilfe eines Algorithmus auf ein Aggregationsintervall von zwei Wochen zusammengefasst. Datensätze mit zu wenigen Informationen wurden ausgeschlossen, um die Vergleichbarkeit und Datenkonsistenz sicherzustellen. Dadurch weisen die resultierenden Zeitreihen nun eine weitgehend einheitliche zeitliche Abdeckung auf und bilden eine robuste Grundlage für die nachfolgende Modellierung.

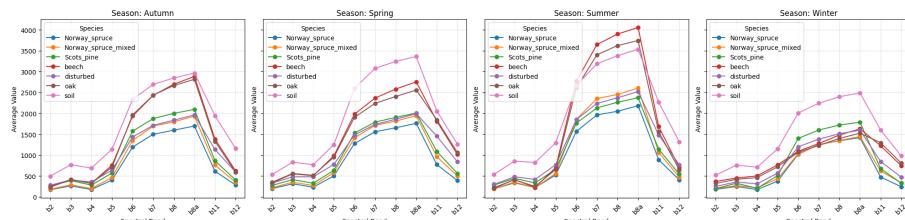


Abbildung 4.3: Durchschnittswerte der Bänder pro Jahreszeit

Zur Identifikation von Mustern in den Spektralbändern wurden verschiedene Visualisierungen erstellt. Die saisonalen Verläufe der Bänder (4.3) zeigen deutliche Unterschiede der Reflexionswerte zwischen Vegetationsperioden und liefern Hinweise auf vegetationsspezifische Veränderungen. Erkennbar ist, dass sich vor allem die Bodenwerte deutlich von den Baumwerten unterscheiden und dass insbesondere im Sommer die Spektralbänder der verschiedenen Labels klare Unterschiede aufweisen.

Die saisonale Komponente (4.4) verdeutlicht den periodischen Verlauf des Bandes B11 über das Jahr. Wird die Zeitreihe um ein Jahr verschoben, schwankt die Differenz zwischen Original und verschobener Reihe um null, ein Hinweis auf stark wiederkehrende Muster. Damit zeigt die Grafik, dass sich die spektralen Signale über die Jahre ähneln und somit redundante, aber stabile saisonale Informationen enthalten.

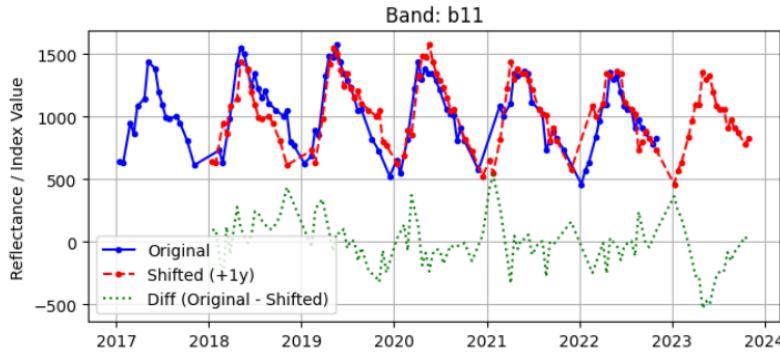


Abbildung 4.4: Verschiebung der Zeitreihe für b11

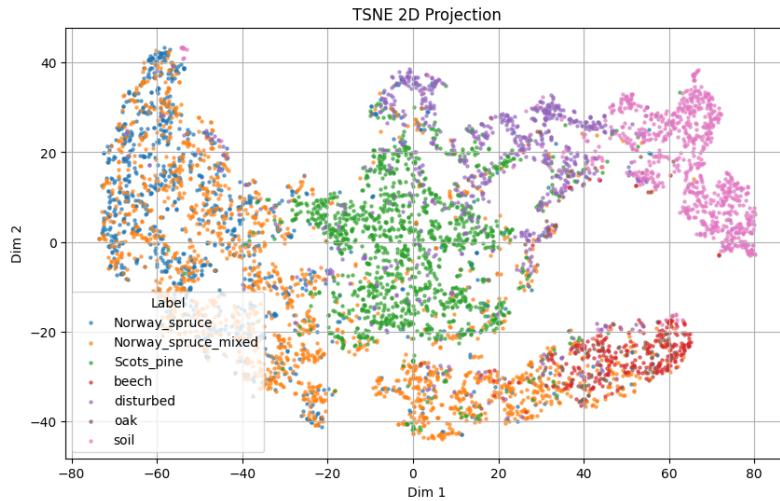


Abbildung 4.5: t-SNE Analyse

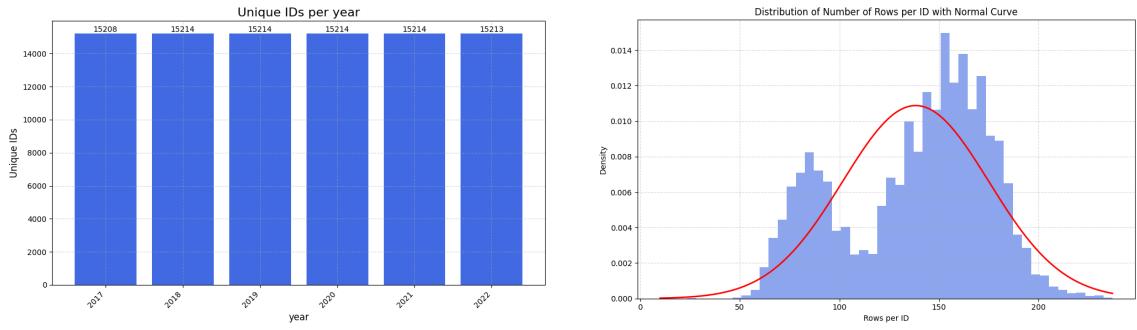
Um zu prüfen, ob sich ein Clusterverfahren als ML-Modell eignet, wurde eine t-SNE-Analyse durchgeführt (4.5). In der Darstellung ist erkennbar, dass sich vor allem das Label Soil deutlich von den anderen Klassen abgrenzt. Eine gewisse Trennung zeigt sich zudem bei Disturbed und Scots Pine, während die übrigen Baumarten stark überlappen. Dies weist darauf hin, dass die Datenstruktur nach der Dimensionsreduktion nur begrenzt clusterfähig ist und ein unüberwachtes Verfahren daher wenig geeignet erscheint.

4.2 Explorative Datenanalyse Validierungsdatensatz

Der Validierungsdatensatz wurde am Ende des Projekts ohne Labels bereitgestellt, um darauf das trainierte Modell anzuwenden. Die Anzahl der Beobachtungen und die zeitlichen Lücken wurden analysiert, um sicherzustellen, dass sich der Validierungsdatensatz nicht wesentlich vom Trainingsdatensatz unterscheidet.

Abbildung 4.6a zeigt, dass im Validierungsdatensatz für sechs IDs keine Daten im Jahr 2017 und für eine ID keine Daten im Jahr 2022 vorliegen. Insgesamt ist die zeitliche Abdeckung jedoch weitgehend vollständig.

Die Verteilung der Zeilen pro ID (4.6b) entspricht weitgehend der des Trainings-



(a) Eindeutige IDs pro Jahr im Validierungsdatensatz
 (b) Verteilung: Anzahl Zeilen pro ID im Validierungsdatensatz

Abbildung 4.6: Übersicht der ID-Verteilung im Validierungsdatensatz

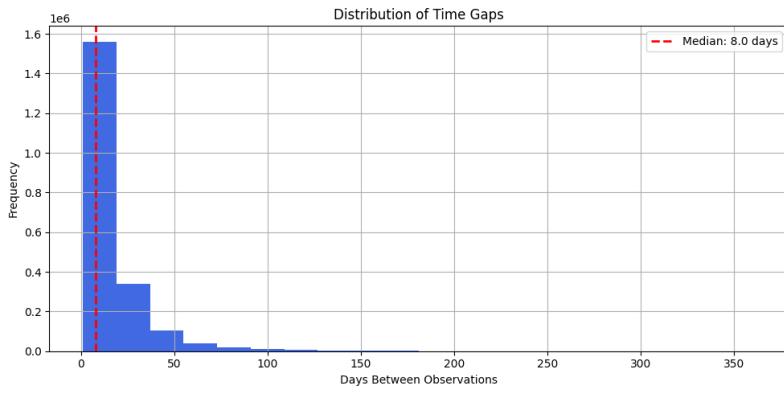


Abbildung 4.7: Häufigkeitsverteilung der Zeitlücken im Validierungsdatensatz

datensatzes. Es gibt einige IDs mit deutlich weniger Beobachtungen, und die Verteilung weist keine Normalverteilung auf.

Die Analyse der Zeitlücken (4.7) verdeutlicht, dass die meisten Zeitabstände zwischen den Beobachtungen gering sind, der Median liegt bei etwa acht Tagen. Nur in wenigen Fällen treten größere Lücken von über 300 Tagen auf, was auf unregelmäßige oder ausgefallene Aufnahmen hindeutet.

4.3 Erkennen von kranken Bäumen

Zur Erkennung potenziell kranker Bäume, die nicht als „disturbed“ gelabelt sind, wurden zunächst Merkmale identifiziert, die auf Baumkrankheiten hinweisen können. Dazu wurden die Daten pro Baumart normalisiert, um unterschiedliche Wertebereiche auszugleichen, und pro Jahr aggregiert, um saisonale Schwankungen zu reduzieren. Für jeden Baum wurden aus den Jahresmitteln die Steigung und die Standardabweichung berechnet, während die Differenz zwischen dem ersten und letzten Jahr sich als wenig aussagekräftig erwies. Anschließend wurden die Merkmale auf ihre Korrelation mit dem Label `is_disturbed` untersucht, und Bäume mit hohen Merkmalswerten visuell überprüft, um die wichtigsten Indikatoren zu bestätigen.

In 4.8 sind die Top Features, die mit dem Label `is_disturbed` korrelieren, zu

erkennen. Vor allem die Standardabweichung der Indizes ndmi und nbr scheint ein starker Indikator dafür zu sein, kranke Bäume zu erkennen.

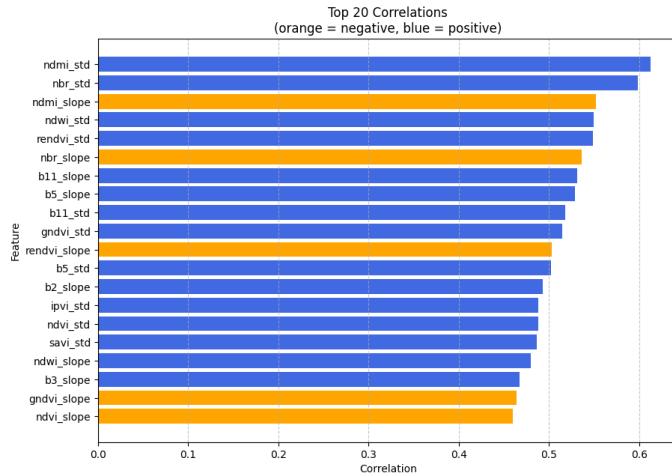


Abbildung 4.8: Korrelationen mit Disturbance

Eine Analyse von Bäumen, die ursprünglich als gesund gelabelt wurden, zeigte, dass extrem hohe oder niedrige Messwerte (je nach Art des Einflusses positiv oder negativ) häufig auf falsch gelabelte, tatsächlich gestörte Bäume hindeuten. In den untersuchten Zeitreihen waren dabei deutliche Veränderungen erkennbar: Ab einem bestimmten Zeitpunkt stiegen oder fielen die Werte abrupt, was auf eine Störung im Wachstumsverlauf schließen lässt. Zur Veranschaulichung wurden mehrere Beispiele grafisch dargestellt, deren Label fälschlicherweise „nicht gestört“ lautet (siehe Abbildung 4.9). Die Abbildungen zeigen die IDs mit dem höchsten, dem 300., dems 400. und dem 550. höchsten b11_slope-Wert. Während die ersten Beispiele klar auf eine Störung hindeuten, ist bei der letzten ID nicht mehr eindeutig zu erkennen, ob sie tatsächlich als „disturbed“ klassifiziert werden sollte.

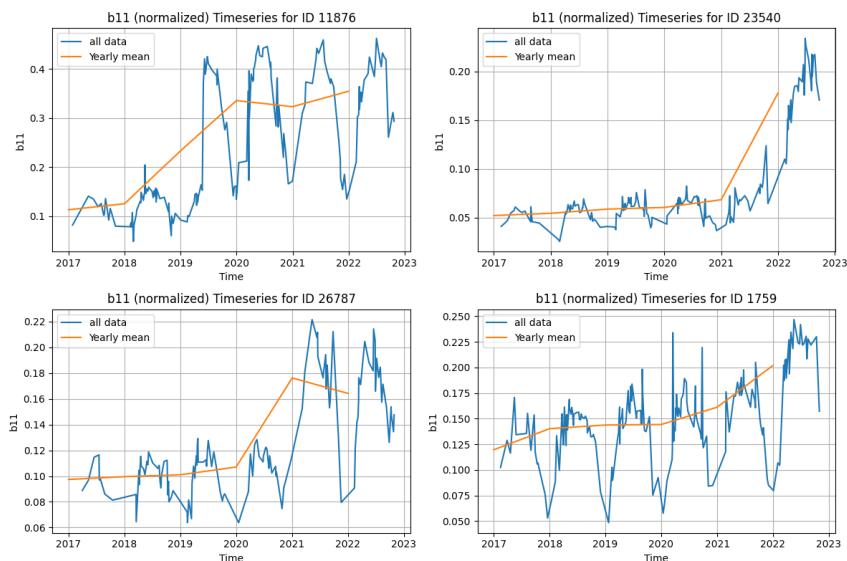


Abbildung 4.9: Vier Beispiele mit hoher b11-Steigung

Auf Basis dieser Merkmale wurde ein ML-Modell trainiert: Als Trainingsdaten dienten alle als disturbed gelabelten Bäume sowie eine Stichprobe vermutlich gesunder Bäume. Der Klassifikator lernte, gestörte von gesunden Bäumen zu unterscheiden und konnte anschließend auf die als gesund geltenden Bäume angewendet werden, um potenziell kranke Exemplare zu identifizieren. Zusätzlich wurde ein Algorithmus implementiert, welcher diese gefundenen Bäume entfernt. In Abschnitt 5.3 können hierzu weitere Informationen zum Algorithmus mit dem Namen "detect disturbed trees" nachgelesen werden.

Kapitel 5

Prozessstruktur und Datenverarbeitungsschritte

5.1 Beschreibung der Prozessstruktur

Das Flowchart (5.1) veranschaulicht den vollständigen Workflow der SITS Multi-Tree-Classification-Pipeline, von der Datenvorverarbeitung bis hin zur abschließenden Modelldiagnose. Der gesamte Prozess ist in vier Hauptphasen gegliedert: Preprocessing, Processing, Data Reduction, Model Training sowie Model Analysis.

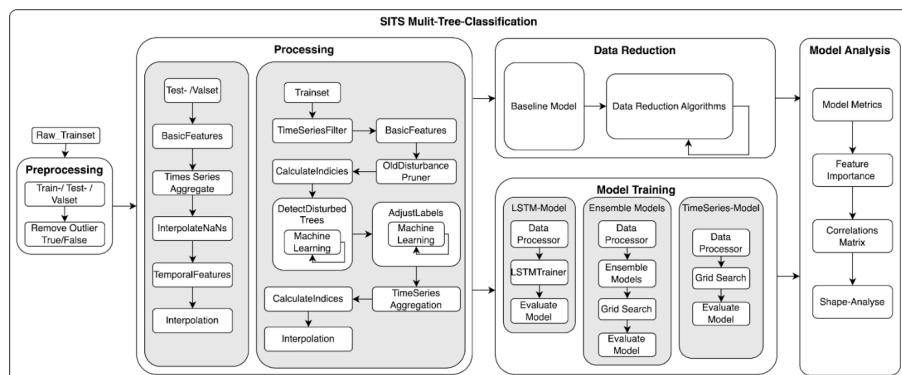


Abbildung 5.1: Flowchart (Pipeline Struktur)

Zum einfachen Einsatz der Pipelines wurde entschieden, die grundlegende Verarbeitung, einschließlich des Aufteilens in Trainings-, Validierungs- und Testdatensatz sowie der Anomalieerkennung, als Preprocessing zu bezeichnen. Die nachfolgenden Schritte, die detailliertere Datenverarbeitungen durchführen, wurden als Processing benannt. Diese Unterscheidung ermöglicht eine klare Trennung der beiden Pipeline-Typen und sorgt für eine übersichtlichere Code-Struktur.

Diese strukturierte Vorgehensweise stellt sicher, dass die Daten konsistent, reproduzierbar und nachvollziehbar verarbeitet werden, wodurch die Grundlage für eine valide Modellbewertung geschaffen wird.

5.2 Datenvorverarbeitung

Die Vorverarbeitungsphase bildet den ersten und grundlegenden Schritt der Pipeline. Ziel ist es, eine saubere und konsistente Datenbasis zu schaffen, um die Qualität und Generalisierbarkeit der nachfolgenden Machine-Learning-Modelle sicherzustellen.

Ein zentrales Problem beim Training von Machine-Learning-Modellen ist das sogenannte Data Leakage, welches unbedingt vermieden werden muss. Data Leakage tritt auf, wenn Informationen aus Test- oder Validierungsdaten unbeabsichtigt in den Trainingsdatensatz einfließen. Dies kann zu einer überschätzten Modellleistung führen, da das Modell auf Daten trainiert wird, die es eigentlich später vorhersagen soll. Um dies zu verhindern, wurde der Datensatz vor der weiteren Verarbeitung in Trainings-, Test- und Validierungsdaten im Verhältnis 70/20/10 aufgeteilt. Diese Vorgehensweise trägt dazu bei, Overfitting auf den Trainingsdatensatz zu reduzieren und ein generalisierbares Modell zu entwickeln.

Der Datensatz wurde bereits vor seiner Bereitstellung einer Anomalieerkennung unterzogen (??), um offensichtliche Ausreißer und fehlerhafte Beobachtungen zu identifizieren. Bei der anschließenden Durchsicht der Daten und insbesondere der spektralen Verläufe konnten jedoch weiterhin Ausreißer festgestellt werden. Um die Modellgüte zu erhöhen, wurden diese Anomalien vor der Verarbeitung in der Pipeline für jede ID individuell entfernt. Durch diese gezielte Bereinigung wird erwartet, dass das Modell robuster gegenüber extremen Werten wird und eine höhere Vorher sagegenauigkeit erzielt.

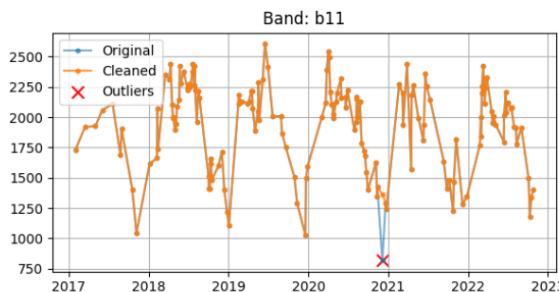


Abbildung 5.2: Ausreißer Erkennung

Das Ergebnis des Preprocessing-Schritts ist ein validierter, aufbereiteter Datensatz, der als Grundlage für die nachfolgenden Verarbeitungsschritte dient.

5.3 Datenverarbeitung

Die Verarbeitungsphase (5.3) ist in zwei parallele Stränge unterteilt: einen für die Trainingsdaten und einen für die Validierungs- und Testdaten.

Für die Validierungs- und Testdaten liegt der Fokus auf der Extraktion grundlegender Merkmale, der zeitlichen Aggregation der Daten sowie der Handhabung fehlender Werte. Diese Schritte sorgen dafür, dass die Daten in einer konsistenten, sauberen Form für die Modellbewertung vorliegen.

Die Trainingspipeline enthält zusätzlich Schritte, die speziell auf die Vorbereitung der Daten für das Modelltraining abzielen, etwa das Filtern kurzer oder weniger

```

test_steps = [
    BasicFeatures(on=True),
    TimeSeriesAggregate(on=True, freq=2, method="mean"),
    InterpolateNaNs(on=True, method="linear"),
    Smooth(on=True, overwrite=True),
    CalculateIndices(on=True),
    TemporalFeatures(on=True),
    Interpolation(on=True),
]
train_steps = [
    TimeSeriesFilter(on=True, max_median_diff_days=25),
    BasicFeatures(on=True),
    OldDisturbancePruner(on=False),
    CalculateIndices(on=True),
    DetectDisturbedTrees(on=False),
    AdjustLabels(on=False),
    DataAugmentation(on=False),
    TimeSeriesAggregate(on=True, freq=2, method="mean"),
    InterpolateNaNs(on=True, method="linear"),
    Smooth(on=True, overwrite=True),
    Interpolation(on=True),
    CalculateIndices(on=True),
    TemporalFeatures(on=True),
]

```

Abbildung 5.3: Train- und Test-Verarbeitungsschritte

informativer Sequenzen sowie die Anpassung von Labels oder die Identifikation von kranken Bäumen. Damit unterscheidet sich die Trainingspipeline von der Test- und Validierungspipeline, deren Ziel primär die konsistente Darstellung der Daten zur Evaluation ist.

Im Allgemeinen lassen sich die Verarbeitungsoperationen in drei Kategorien einteilen:

- Erstellung und Berechnung von Features
- Reduktion und Auswahl von Daten, um die Modellkomplexität zu verringern
- Verarbeitungsschritte wie Interpolation, Glättung oder zeitliche Aggregation

Eine tabellarische Übersicht aller Verarbeitungsschritte ist im Folgenden zu finden (5.1):

Tabelle 5.1: Übersicht aller Verarbeitungsschritte

Processing Class	Category	Description	Depends on class	Uses columns	Generates/Deletes columns	Deletes Rows	Changes values
TimeSeries Filter	Data Reduction	Deletes IDs that don't have enough timestamps		Time, id		Where median_time_diff_days > 18	
BasicFeatures	Features	Generates basic columns		disturbance_year	Is_disturbed		
OldDisturbance Pruner	Data Reduction	Deletes "disturbed" species if disturbance_year too long ago		disturbance_year, species		Where disturbance_year < 2017 for col "disturbed"	
Calculate Indices	Features	Calculate indices		Spectral Bands	Indices (NDVI, REIP, ...)		
Detect DisturbedTrees	Processing Step	Deletes IDs that may be disturbed (and are mislabeled)	CalculateIndices	Bands & Indices, species, id, year, disturbance_year	Is_disturbed, Is_disturbed_pred	Where is_disturbed for a "healthy" id was predicted	
AdjustLabels	Processing Step	Specify label disturbed	CalculateIndices	Disturbance_year, species, id			Changes "soil" to soil_disturbedland "disturbed" to "Norway_spruce_disturbedbedrock_pine_disturbed"
Data Augmentation	Processing Step	Generates new data points so that the ids have 150 timestamps. Generates new ids for underrepresented classes.	DetectDisturbedTrees, AdjustLabels	Time, species, id			New ids with "aug" and new feature values
TimeSeries Aggregate	Processing Step	Aggregates data so that all timestamps fall on Mondays	DataAugmentation	Time, id			Calculates mean/median if aggregated
Interpolate NaNs	Processing Step	Quadratic interpolation for the few remaining NaNs	TimeSeriesAggregate	Id			Interpolates if feature column is null
Smooth	Processing Step	Reduces noise		Spectral Bands, id, time			Smooths spectral bands
Interpolation	Processing Step	Interpolates b4 if b4=0		B4, id, time			Interpolation if b4=0
Calculate Indices	Features	Calculate indices	DataAugmentation, Interpolation	Spectral Bands	Indices (NDVI, REIP, ...)		
Temporal Features	Features	Create new features using time	Data Augmentation	Time	Month (num, sin, cos), year, season, datediff, is_growing_season		

5.4 Data Augmentation zur Verbesserung der Trainingsdaten

Die Qualität und Quantität der Trainingsdaten sind entscheidende Faktoren für die Leistungsfähigkeit von Machine-Learning-Modellen. In vielen realen Anwendungsfällen, insbesondere bei der Arbeit mit Zeitreihendaten aus der Fernerkundung, sind die verfügbaren Datensätze oft unvollständig oder unausgeglichen. Um diese Herausforderungen zu bewältigen und die Robustheit unseres Modells zu verbessern, wurde ein dedizierter Datenaugmentierungsprozess implementiert.

5.4.1 Problemstellung und Motivation

Der ursprüngliche Trainingsdatensatz wies zwei wesentliche Probleme auf, die die Modellleistung beeinträchtigen könnten:

1. **Unausgeglichene Klassen (Imbalance):** Wie in Abbildung 5.4 dargestellt, war die Anzahl der eindeutigen Baum-IDs pro Baumart sehr unterschiedlich. Einige Arten, wie die Fichte (*Norway spruce*), waren stark vertreten, während andere, wie Eiche (*oak*) oder Buche (*beech*), deutlich seltener vorkamen. Ein solches Ungleichgewicht kann dazu führen, dass das Modell eine Tendenz zur Vorhersage der Mehrheitsklassen entwickelt und bei den selteneren Klassen eine schlechtere Leistung zeigt.

2. Irreguläre und lückenhafte Zeitreihen: Die Zeitreihen der Spektralbänder für die einzelnen Bäume wurden in unregelmäßigen Abständen und mit variierender Häufigkeit erfasst. Die Analyse der Zeitdifferenzen zwischen aufeinanderfolgenden Messpunkten ergab durchschnittliche Abstände von 13 bis 15 Tagen, jedoch mit hoher Varianz. Zudem wiesen viele Zeitreihen Lücken auf oder waren deutlich kürzer als der gesamte Beobachtungszeitraum, wie das Beispiel in Abbildung 5.5 zeigt. Dies erschwert das Training von Modellen, die eine feste Eingabelänge oder eine regelmäßige Abtastung erwarten, wie z.B. rekurrente neuronale Netze (RNNs) oder Transformer.

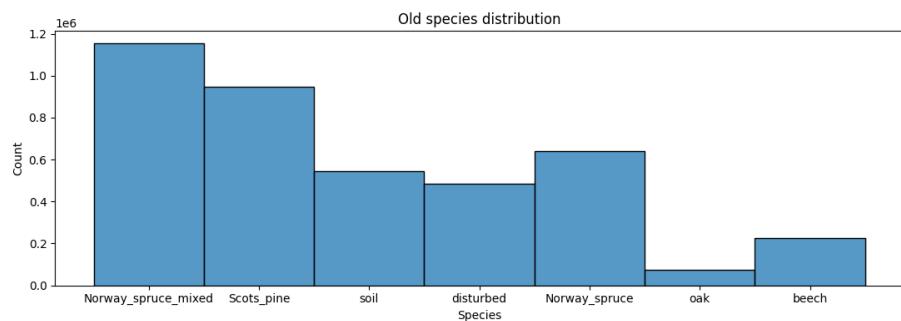


Abbildung 5.4: Verteilung der Baumarten im ursprünglichen Datensatz. Die unausgeglichene Anzahl von IDs pro Spezies ist deutlich erkennbar.

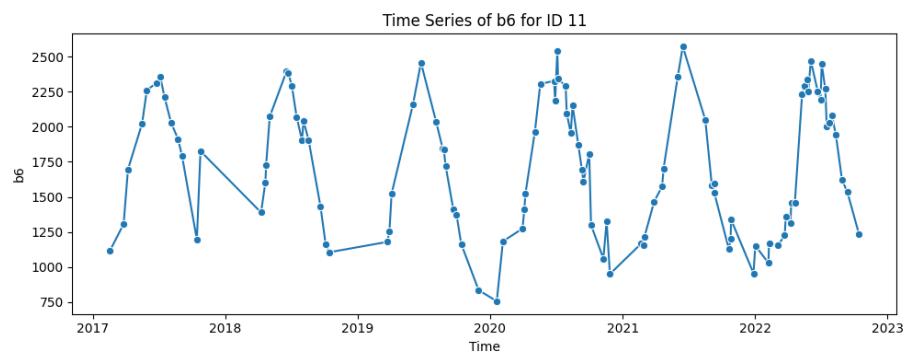


Abbildung 5.5: Beispiel einer lückenhaften Zeitreihe für die Baum-ID 11 vor der Augmentierung.

Das Ziel der Datenaugmentierung war es daher, einen neuen, angereicherten Trainingsdatensatz zu erstellen, der (1) eine ausgeglichene Verteilung der Baumarten aufweist und (2) für jede ID eine dichte, regelmäßig abgetastete Zeitreihe über den gesamten Zeitraum enthält.

5.4.2 Methodik der Augmentierung

Der Augmentierungsprozess wurde in einer Python-Klasse `DataAugmentation` gekapselt und lässt sich in drei Hauptschritte unterteilen:

Resampling und Interpolation

Zunächst werden alle Zeitreihen auf eine einheitliche Länge und eine gemeinsame, feste Zeitachse gebracht. Basierend auf dem gesamten Beobachtungszeitraum und der durchschnittlichen Messfrequenz wurde eine Zieldauer von 152 Datenpunkten mit einem festen Intervall von 14 Tagen gewählt.

Für jede Baum-ID werden die vorhandenen Messpunkte auf diese neue Zeitachse projiziert. Fehlende Werte, die durch dieses Resampling entstehen, werden mittels linearer Interpolation gefüllt. Dieser Schritt stellt sicher, dass alle Zeitreihen dieselbe Struktur aufweisen und lückenlos sind.

Anwendung von Augmentierungstechniken

Nach der Vereinheitlichung werden auf die Zeitreihen der zu augmentierenden Klassen (Minderheitsklassen) künstliche Variationen angewendet. Hierfür wurde die Python-Bibliothek `tsaug` verwendet, die speziell für die Augmentierung von Zeitreihen entwickelt wurde. Zwei Techniken kamen zur Anwendung:

- **Drift:** Diese Technik fügt der Zeitreihe eine langsame, zufällige Veränderung (Drift) hinzu. Die Werte werden im Zeitverlauf allmählich leicht nach oben oder unten verschoben. Dies simuliert natürliche, langsame Veränderungen in den Spektraleigenschaften, beispielsweise durch unterschiedliche Witterungsbedingungen über die Jahreszeiten.
- **AddNoise:** Hierbei wird ein zufälliges Rauschen mit geringer Amplitude auf die Zeitreihe addiert. Dies modelliert kleinere, kurzfristige Schwankungen oder Messungenauigkeiten in den Satellitendaten.

Diese Augmentierungen werden auf die Original-Zeitreihe angewendet, *bevor* die Interpolation auf die neue, dichte Zeitachse erfolgt. Dadurch wird sichergestellt, dass die erzeugten Variationen realistisch über die Zeitachse verteilt werden.

Oversampling zur Klassenausgleichung

Der Kern der Augmentierung ist das Oversampling der Minderheitsklassen. Die Strategie ist wie folgt:

1. Zuerst wird die Anzahl der IDs in der größten Klasse (Mehrheitsklasse) ermittelt.
2. Für jede Minderheitsklasse wird die Differenz zur Größe der Mehrheitsklasse berechnet.
3. Um diese Differenz auszugleichen, werden zufällig IDs aus der jeweiligen Minderheitsklasse ausgewählt (mit Zurücklegen).
4. Von diesen ausgewählten IDs werden augmentierte Kopien erstellt, indem die oben beschriebenen Drift- und Rausch-Transformationen mit unterschiedlichen Zufalls-Seeds angewendet werden.

- Diese neu generierten, augmentierten Zeitreihen werden dem Datensatz hinzugefügt, bis jede Klasse die gleiche Anzahl an IDs wie die ursprüngliche Mehrheitsklasse aufweist.

Durch diesen Prozess entsteht ein ausbalancierter Datensatz, in dem jede Baumart gleich stark repräsentiert ist.

5.4.3 Ergebnisse

Die Anwendung des Augmentierungsprozesses führte zu einem deutlich verbesserten Datensatz für das Modelltraining.

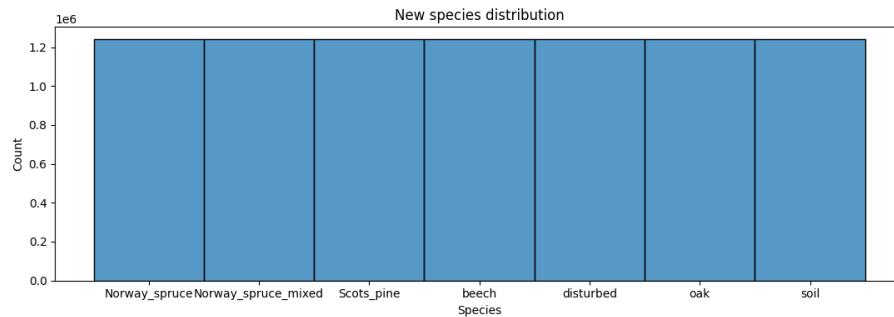


Abbildung 5.6: Verteilung der Baumarten nach der Augmentierung. Alle Klassen weisen nun die gleiche Anzahl von IDs auf.

Abbildung 5.6 zeigt die Verteilung der Baumarten nach dem Oversampling. Es ist klar ersichtlich, dass alle Klassen nun die gleiche Anzahl an Instanzen haben, wodurch das Problem der Klassen-Imbalance behoben ist.

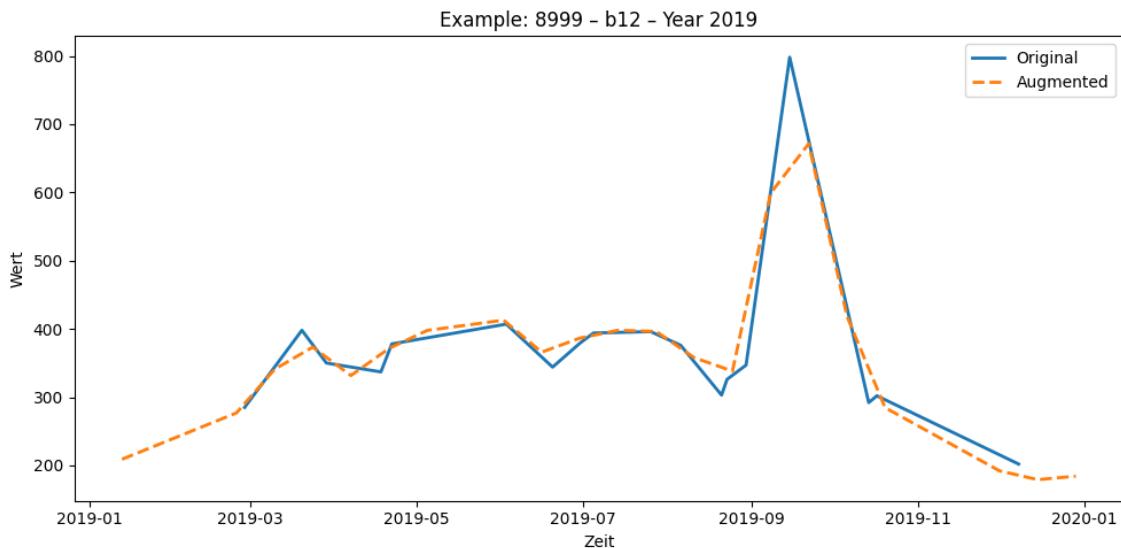


Abbildung 5.7: Vergleich einer originalen (durchgezogene Linie) und einer augmentierten (gestrichelte Linie) Zeitreihe für das Spektralband B12 der Baum-ID 8999.

Abbildung 5.7 visualisiert den Effekt der Augmentierung auf eine einzelne Zeitreihe. Die augmentierte Kurve folgt dem allgemeinen Trend der Originaldaten, weist

jedoch leichte Abweichungen auf, die durch die Drift- und Rausch-Methoden eingeführt wurden. Die Punkte zeigen die ursprünglichen Messungen (blau) und die neuen, interpolierten und augmentierten Datenpunkte (orange) auf der regelmäßigen 14-Tage-Achse.

Schließlich wurde überprüft, ob die Augmentierung die statistischen Eigenschaften der Daten signifikant verändert. Abbildung 9.1 zeigt den Vergleich der Werteverteilungen für alle Spektralbänder vor und nach der Augmentierung. Die Boxplots verdeutlichen, dass die grundlegenden statistischen Kennzahlen wie Median, Quartile und die Streuung der Daten weitgehend erhalten bleiben. Dies ist wichtig, um sicherzustellen, dass die augmentierten Daten die ursprünglichen Charakteristika der jeweiligen Baumart nicht verfälschen.

5.4.4 Zusammenfassung

Die Datenaugmentierung war ein entscheidender Schritt in der Vorverarbeitung. Durch die Kombination aus Resampling, Interpolation, leichten augmentativen Veränderungen und einem gezielten Oversampling-Ansatz konnte ein umfangreicher, dichter und klassen-ausgeglichener Trainingsdatensatz erzeugt werden. Dies legt eine solide Grundlage für das Training eines robusteren und generalisierbareren Klassifikationsmodells.

Kapitel 6

Modell Ergebnisse

6.1 Generelles Vorgehen

Nach der Datenverarbeitung wird zunächst ein Baseline-Modell erstellt, das als Referenzpunkt für die Bewertung der Modellverbesserungen dient.

Im anschließenden Modell Training werden verschiedene Modelltypen trainiert, darunter:

- LSTM-Modelle für sequenzielle Zeitreihenanalysen,
- Ensemble-Modelle (z. B. Random Forest) zur robusten Klassifikation, sowie TimeSeries-Modelle
- PYTS Classification

Nach dem Training werden die Modelle anhand standardisierter Metriken evaluiert, um die Leistungsfähigkeit objektiv zu vergleichen.

In der abschließenden Analysephase werden die Modellmetriken ausgewertet, um die Performance der verschiedenen Modellansätze zu beurteilen. Anschließend wird eine Feature-Importance-Analyse durchgeführt, um die Bedeutung einzelner Eingangsvariablen für die Modellentscheidungen zu bestimmen.

Darüber hinaus wird eine Korrelationsmatrix erstellt, um Zusammenhänge zwischen den Merkmalen zu identifizieren. Abschließend erfolgt eine Shape-Analyse, welche die zeitlichen und strukturellen Muster innerhalb der Daten untersucht und zur Interpretation der Modellergebnisse beiträgt.

6.2 Modell Vergleich

Im Projekt wurden verschiedene Modelle umgesetzt (6.1), beginnend bei einfachen Baseline-Modellen bis hin zu einem bidirektionalen LSTM. Jedes Modell wurde iterativ weiterentwickelt, um schrittweise die Leistungsfähigkeit zu verbessern. Am Ende wurde das beste Modell anhand der gewichteten Accuracy ausgewählt und für die Vorhersagen eingesetzt.

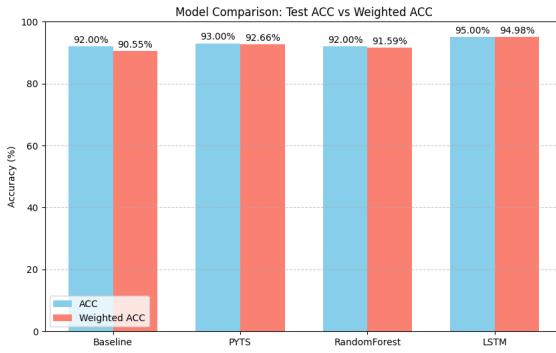


Abbildung 6.1: Accuracy Modell Vergleich

6.3 Baseline Modell

Für das Baseline-Modell wurde bewusst ein möglichst einfaches Vorgehen gewählt, das nur minimale Datenvorverarbeitung erfordert. Zu diesem Zweck wurden die Daten nach `id` gruppiert und anschließend für jede `id` statistische Kennzahlen berechnet. Für jeden Band und Index wurden dabei monatlich die Werte `max`, `min`, `mean` und `std` bestimmt. Der resultierende Datensatz hatte somit eine Struktur wie in Tabelle 6.1 dargestellt.

Tabelle 6.1: Beispielhafter Aufbau des Baseline-Datensatzes

<code>id</code>	<code>b1_min_January</code>	<code>b1_max_January</code>	<code>b1_mean_January</code>	<code>b1_std_January</code>	...
123	0.1	101	56	2.3	...

Der Vorteil dieses Ansatzes liegt darin, dass anstelle der ursprünglichen Zeitreihen eine vereinfachte, tabellarische Repräsentation entsteht, die direkt für eine Klassifikation genutzt werden kann. Die resultierenden Merkmale sind leicht interpretierbar und eignen sich gut für weiterführende Analysen, beispielsweise im Rahmen einer Feature-Analyse.

Als Modell wurde XGBoost eingesetzt, mit den folgenden Ergebnissen:

- **Accuracy:** 92.00 %
- **Weighted Accuracy:** 90.55 %

Es zeigt sich, dass bereits mit einem einfachen Modell sehr gute Resultate erzielt werden können. Dies unterstreicht, dass die gewählten Merkmale eine hohe Aussagekraft für die Klassifikation besitzen und generell der Datensatz schon gut bereinigt ist und die Bäume sich signifikant voneinander unterscheiden.

6.4 Ensemble-Modell

Für das Ensemble-Modell wurden die zeitabhängigen Daten zunächst systematisch aufbereitet, um sie für klassische Machine-Learning-Algorithmen wie den Random Forest nutzbar zu machen. Hierzu kam die Klasse `TimeSeriesAggregator` zum Einsatz, die die Daten in überlappende Sliding Windows segmentiert. Innerhalb jedes

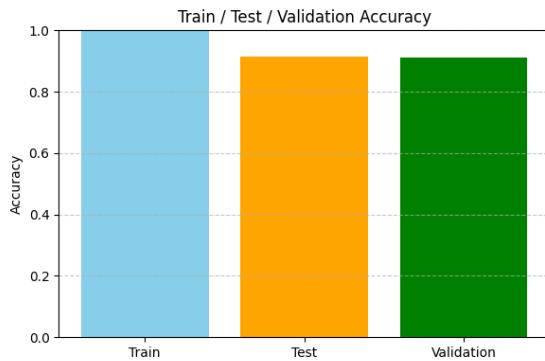


Abbildung 6.2: Ensemble Model Performance

Fensters wurden für alle relevanten Merkmale, einschließlich der Spektralbänder und der daraus abgeleiteten Indizes, verschiedene statistische Kennzahlen berechnet:

- **Mittelwert, Standardabweichung, Minimum und Maximum** jedes Features, um zentrale Tendenzen und Variabilität abzubilden,
- **Trendberechnung** innerhalb des Fensters zur Erfassung zeitlicher Entwicklungen,
- Aggregation der Fenster zu einem einzelnen Datensatz, wobei die Merkmale für jedes Fenster separat erhalten bleiben, um den zeitlichen Verlauf zu repräsentieren.

Auf Basis dieser aggregierten Features konnte eine robuste, interpretierbare Eingabe für das Ensemble-Modell erstellt werden. Durch die Kombination aus statistischen Kennzahlen und trendbasierten Features war das Modell in der Lage, komplexe zeitliche Muster zu erkennen, während zusätzliche Merkmale wie z.B. `is_growing_season` keinen nennenswerten Einfluss auf die Modellleistung hatten.

Das Ensemble-Modell selbst basiert auf einem Random Forest Classifier, der in einer modularen Pipeline trainiert, validiert und für Vorhersagen eingesetzt wurde. Die sorgfältige Feature-Aufbereitung mit Sliding Windows bildete dabei die Grundlage für die stabile und zuverlässige Performance des Modells.

6.5 LSTM Modell

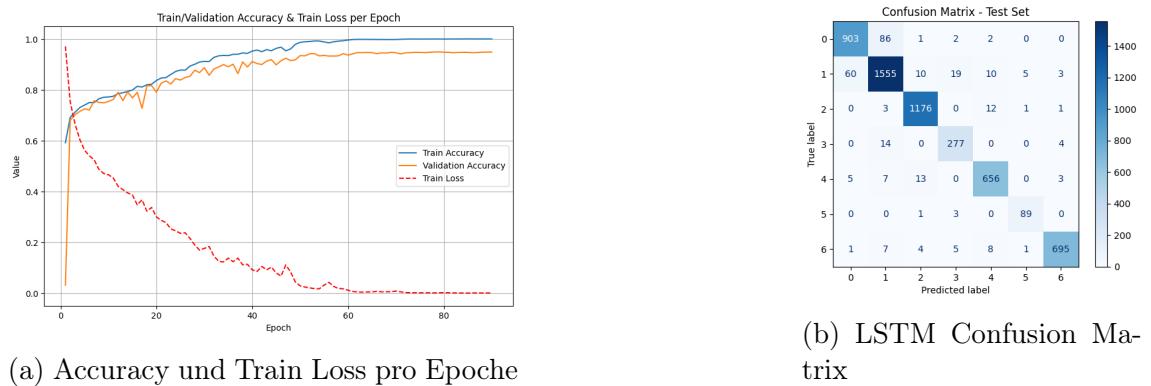
Im Projekt wurde ein mehrschichtiges, bidirektionales LSTM-Modell (Long Short-Term Memory) entwickelt und implementiert, das speziell zur Verarbeitung von zeitabhängigen Sequenzdaten konzipiert ist. Das Modell besteht aus zwei LSTM-Schichten mit jeweils 256 Neuronen und nutzt Dropout zur Regularisierung, um Overfitting zu vermeiden. Durch die Bidirektionalität werden sowohl vergangene als auch zukünftige Kontextinformationen berücksichtigt, was zu einer verbesserten Modellgenauigkeit führt. Die extrahierten Merkmalsrepräsentationen werden anschließend über vollständig verbundene Schichten weiterverarbeitet, um eine präzise Klassifikation der Zielvariable zu ermöglichen.

Tabelle 6.2: Classification Report: Validation vs Test Set

Class	Validation			Test		
	Precision	Recall	F1-score	Precision	Recall	F1-score
0	0.90	0.89	0.90	0.93	0.91	0.92
1	0.92	0.91	0.92	0.93	0.94	0.93
2	0.98	0.99	0.98	0.98	0.99	0.98
3	0.91	0.96	0.94	0.91	0.94	0.92
4	0.96	0.96	0.96	0.95	0.96	0.96
5	0.87	0.91	0.89	0.93	0.96	0.94
6	0.98	0.97	0.98	0.98	0.96	0.97
Accuracy		0.94			0.95	
Macro Avg	0.93	0.94	0.94	0.94	0.95	0.95
Weighted Avg	0.94	0.94	0.94	0.95	0.95	0.95

Bemerkung: Die Zuordnung der Klassen 0–6 zu den Species-Namen ist wie folgt: 0 = Norway_spruce, 1 = Norway_spruce_mixed, 2 = Scots_pine, 3 = beech, 4 = disturbed, 5 = oak, 6 = soil.

Das entwickelte bidirektionale LSTM-Modell zeigte in den Validierungsmessungen hervorragende Ergebnisse und wurde daher als zentrales Vorhersagemodell im Projekt eingesetzt. Trotz seiner hohen Rechenintensität und langen Trainingslaufzeit überzeugte das Modell durch stabile und präzise Vorhersagen, was seine Eignung für komplexe zeitabhängige Datenanalysen deutlich unterstrich.



(a) Accuracy und Train Loss pro Epoche

(b) LSTM Confusion Matrix

Abbildung 6.3: Vergleich der Trainingsmetriken und der LSTM-Konfusionsmatrix

6.6 PYTS-Modell

Ergebnisse mit Anomalieerkennung:

- Das Modell erreicht auf dem Testdatensatz eine Accuracy von 90,66% und einen gewichteten F1-Score von 90,56%, was auf eine sehr gute Gesamtleistung hinweist.

- Auf dem Validierungsdatensatz sind die Werte mit 90,79% Accuracy und 90,67% gewichteten F1 nahezu identisch, was auf ein gutes Generalisierungsvermögen hinweist.
- Hohe F1-Scores: Scots_pine (0.95), soil (0.98), beech (0.91) – das Modell unterscheidet diese Klassen sehr zuverlässig.
- Etwas niedrigere Recall-Werte: Norway_spruce (0.80) und disturbed (0.81) – diese Klassen werden gelegentlich falsch klassifiziert, z.B. als Norway_spruce_mixed oder Scots_pine.
- Geringe Support-Klassen: oak (93 Testproben) zeigt eine leicht reduzierte Recall-Rate, vermutlich aufgrund der geringen Datenmenge.

Ergebnisse ohne Anomalieerkennung: (optional: hier Tabelle)

- Das Modell erreicht auf dem Testdatensatz eine Accuracy von 91,47 % und einen gewichteten F1-Score von 91,40 %.
- Auf dem Validierungsdatensatz liegt die Accuracy bei 91,25 % und der gewichtete F1-Score bei 91,16 %. Dies zeigt gute Generalisierung, da die Ergebnisse auf beiden Datensätzen sehr ähnlich sind.
- Die höchsten F1-Werte erreichen die Klassen soil (0.98) und Scots_pine (0.95–0.96).
- Etwas schwächere Ergebnisse zeigen Norway_spruce und oak, hauptsächlich durch leicht geringeren Recall.

6.7 Einfluss der Verarbeitungsschritte auf das Modellergebnis

Durch die Durchführung mehrerer Trainingsläufe wurden die jeweiligen Ergebnisse miteinander verglichen. Dabei zeigte sich, dass die Anwendung der Outlier Detection tendenziell zu einer Verschlechterung der Modellmetriken führte. Die Leistungunterschiede zwischen den Modellen mit und ohne Outlier Detection betrugen dabei etwa 1%.

Dieses Ergebnis lässt sich dadurch erklären, dass der Datensatz bereits vorab bereinigt wurde und somit nur noch eine sehr geringe Kontamination aufwies. Die zusätzliche Anwendung einer Anomalieerkennung führte daher vermutlich zur Entfernung von Datenpunkten, die zwar ungewöhnlich, aber dennoch relevant für die Modellbildung waren. In Datensätzen mit höherem Rauschanteil oder stärkerer Kontamination wäre hingegen zu erwarten, dass eine Outlier Detection einen signifikanten positiven Einfluss auf die Modellgüte hätte.

Zudem wurde, um herauszufinden, welche Verarbeitungsschritte die Modellqualität verbessern, schrittweise jeweils ein weiterer Verarbeitungsschritt ergänzt. Anschließend wurde das Modellergebnis mit dem `pyts`-Modell getestet (nicht mit dem LSTM, aufgrund der längeren Laufzeit) und die Ergebnisse wurden dokumentiert.

Wenn ein Schritt das Modellergebnis verschlechterte, wurde entschieden, diesen nicht weiter zu verwenden.

In Abbildung 6.4 ist eine Visualisierung dargestellt, die das Vorgehen und die jeweiligen Ergebnisse veranschaulicht. Zu beachten ist, dass die y-Achse abgeschnitten wurde, um die Unterschiede deutlicher sichtbar zu machen.

Auffällig ist, dass die Aggregation einen stark positiven Einfluss auf das Modell hat. Dadurch werden die Werte besser vergleichbar, da nun alle Bäume identische Zeitstempel besitzen. Der `OldDisturbancePruner`, welcher alle Bäume mit einem `disturbance_year < 2017` entfernt, verschlechtert hingegen das Modellergebnis. Vermutlich ist es wichtiger, ausreichend viele Daten mit dem Label `disturbed` zu behalten, als sicherzustellen, dass alle `disturbed`-Daten einen `disturbance_year` innerhalb des Messzeitraums aufweisen.

Die Verarbeitungsschritte `DetectDisturbedTrees` und `AdjustLabels` zeigen ebenfalls einen leichten negativen Einfluss. Dies könnte darauf zurückzuführen sein, dass durch `DetectDisturbedTrees` relevante Daten entfernt werden, die trotz der Störung hilfreiche Informationen für das Modell enthalten. Es wurde zudem getestet, die Funktion so anzupassen, dass nur Bäume mit weniger als 100 Datenpunkten entfernt werden, obwohl laut Analyse mindestens 400 falsch gelabelte Bäume im Datensatz vorhanden sind. Dennoch führte dies zu keiner Verbesserung des Modellergebnisses.

Bei `AdjustLabels` lässt sich vermuten, dass durch die Umbenennung der Labels in spezifischere Klassen wie `soil_disturbed` oder `scots_pine_disturbed` der Datensatz zu komplex wurde. Möglicherweise waren die ursprünglichen Labels `soil` und `disturbed` bereits ausreichend aussagekräftig. Warum die Hinzunahme von temporalen Merkmalen (`Temporal Features`) das Modellergebnis verschlechtert hat, ist nicht eindeutig erklärbar. Vermutlich führte die starke Erweiterung der Spaltenanzahl zu einer erhöhten Modellkomplexität, die sich negativ auf die Leistung auswirkte.

Insgesamt verbessern die Verarbeitungsschritte das Modell um ganze 8%, was ein sehr gutes Ergebnis darstellt und die Bedeutung einer sorgfältigen Datenvorverarbeitung deutlich unterstreicht.

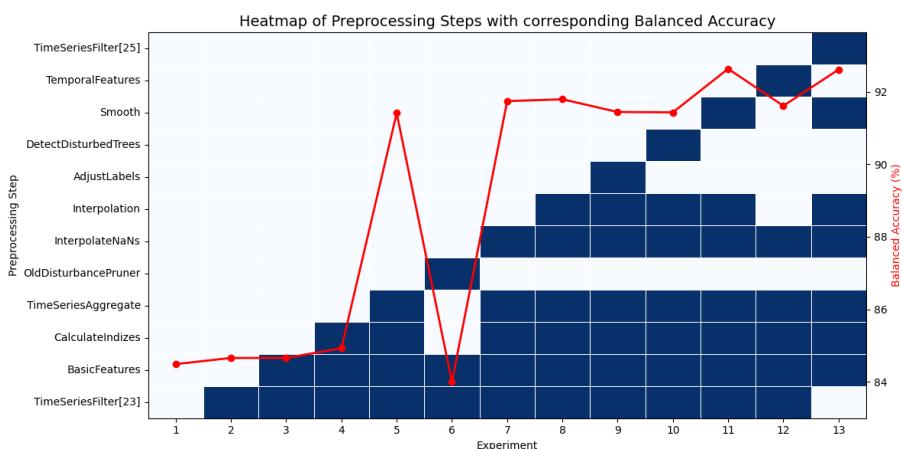


Abbildung 6.4: Einfluss der Verarbeitungsschritte auf das Modellergebnis

6.8 Feature-Analyse

6.8.1 SHAP-Analyse für das Baseline-Modell

In Abbildung 6.5 ist zu erkennen, dass `b3_mean_July` den größten Einfluss auf das Modell hat. Auffällig ist, dass die fünf wichtigsten Merkmale alle aus den Sommer- und Frühlingsmonaten stammen. Erst an sechster Stelle folgt ein Merkmal aus dem Herbst, nämlich `nbr_mean_October`.

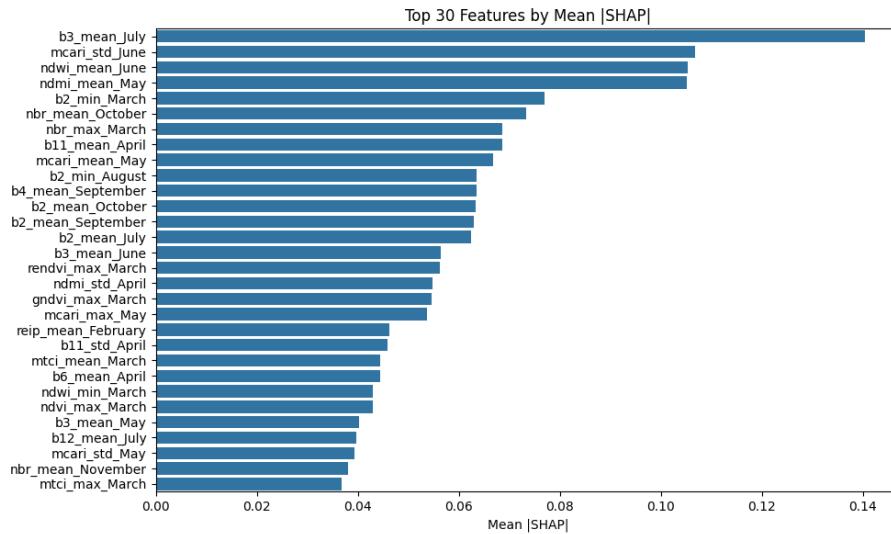


Abbildung 6.5: SHAP: Feature Importance

In Abbildung 6.6 ist im Gegensatz zu Abbildung 6.5 die Feature-Importance nach Monat, Band & Index und Statistik gruppiert dargestellt. Dies ermöglicht einen besseren Vergleich der einzelnen Kategorien untereinander. Auch hier ist erkennbar, dass der Frühling im Baseline-Modell den größten Einfluss hat, gefolgt von Sommer und Herbst.

Bei den Bändern und Indizes zeigt sich, dass `b2` den höchsten Einfluss besitzt, gefolgt von `ndmi`, `mcari` und `nbr`. Unter den Statistiken ist der Mittelwert (`mean`) am relevantesten. Dies ist plausibel, da sich die Mittelwerte zwischen verschiedenen Bäumen in einem Monat stark unterscheiden können, während Kennzahlen wie `min` häufig näher beieinanderliegen und stärker von Ausreißern beeinflusst sein können.

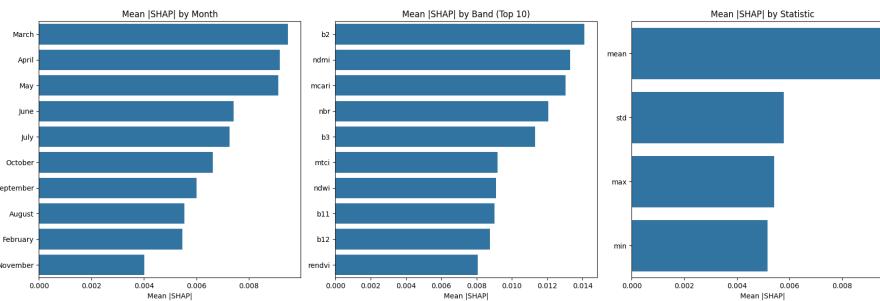


Abbildung 6.6: SHAP: Feature Importance, gruppiert nach Kategorie

Die Heatmap in Abbildung 6.7 bietet den Vorteil, Kombinationen aus zwei Kategorien gleichzeitig zu analysieren. Auffällig ist beispielsweise, dass **b3** insbesondere im Sommer eine hohe Relevanz aufweist. Dies deutet darauf hin, dass **b3** Werte erfasst, die vor allem in dieser Jahreszeit deutliche Unterschiede zwischen den Bäumen zeigen. Der Index **mcari** liefert vor allem im Mai und Juni besonders aussagekräftige Werte, während **b11** fast ausschließlich im April einen deutlichen Einfluss zeigt.

In der zweiten Grafik ist ebenfalls zu erkennen, dass der Mittelwert im Allgemeinen die größte Bedeutung hat, insbesondere der Mittelwert von **b3**. Die Kennzahlen **max**, **min** und **std** von **b3** spielen hingegen kaum eine Rolle. Bei **b2** sind vor allem **mean** und **min** relevant. Generell hat **min** nur eine geringe Bedeutung, jedoch scheint der **min**-Wert von **b2** eine Ausnahme darzustellen.

In der dritten Grafik ist zu sehen, dass die Kennzahl **max** insgesamt nur eine geringe Relevanz besitzt, im März jedoch besonders wichtig ist. Zudem stechen die Mittelwerte des Mai und Juli besonders hervor.

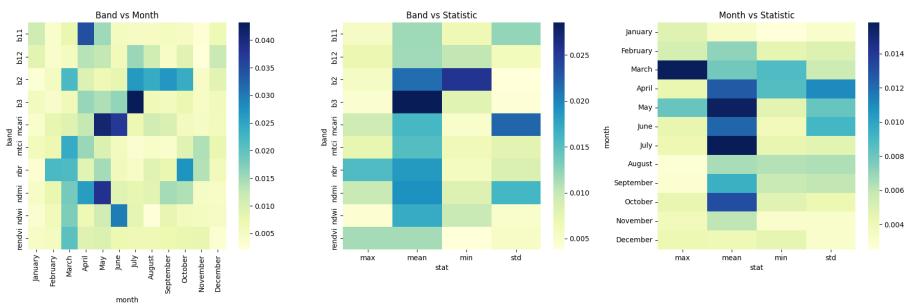


Abbildung 6.7: SHAP: Feature Heatmap

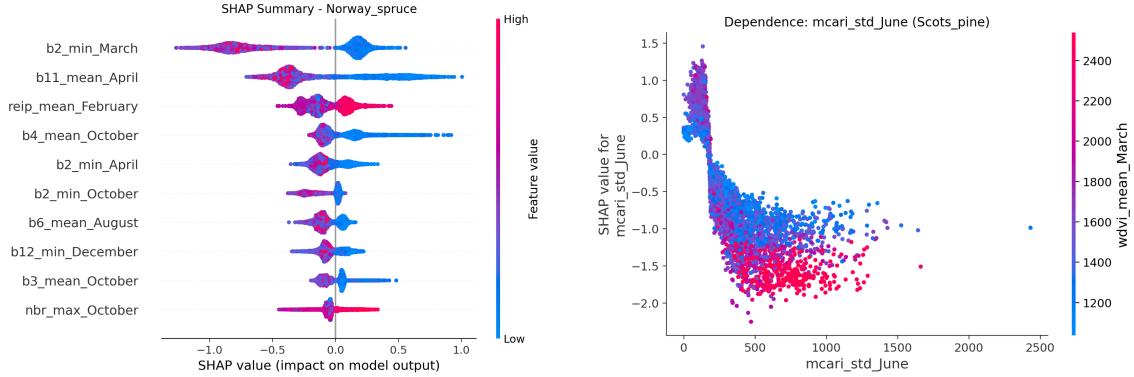
Bemerkung: Für jede Species sind ein Summary-Plot und ein Dependency-Plot im Anhang dargestellt (9.2 und 9.3); in Abbildung 6.8 werden beispielhaft jeweils nur die Ergebnisse für eine Species gezeigt.

In der Feature-Importance-Darstellung für *Norway Spruce* (6.8a) ist insgesamt zu erkennen, dass vor allem geringere Werte der Merkmale einen eindeutigen Einfluss auf das Modell haben. Der SHAP-Wert ist überwiegend positiv, wenn die Merkmale blau dargestellt sind, also niedrige Werte aufweisen. Ein negativer Einfluss zeigt sich hingegen meist bei mittleren bis hohen Werten.

Das wichtigste Merkmal zur Identifikation von *Norway Spruce* ist **b2_min_March**. Dieses weist bei niedrigen Werten einen leicht positiven Einfluss auf das Modell auf, während hohe bis mittelhohe Werte einen stark negativen Effekt haben.

In den Dependency-Plots (6.8b) wurde für jede Species das jeweils wichtigste Merkmal extrahiert und jenem Merkmal gegenübergestellt, mit dem es am stärksten korreliert. Für *Scots Pine* ist **mcari_std_June** das wichtigste Merkmal, welches eine hohe Korrelation mit **wdvi_mean_March** aufweist. Diese Korrelation lässt sich dadurch erklären, dass beide Kennzahlen unter anderem mit **b4** berechnet werden.

Erkennbar ist, dass der Einfluss ab einem **mcari_std_June**-Wert von etwa 200 unmittelbar negativ wird. Zudem zeigt sich, dass bei negativem Einfluss der **wdvi_mean_March** sowohl hohe als auch niedrige Werte annehmen kann; ist **wdvi_mean_March** jedoch hoch, so verstärkt sich der negative Einfluss zusätzlich.



(a) Feature Importance for Norway Spruce

(b) Dependence Plot for Scots Pine

Abbildung 6.8: SHAP-Ergebnisse für verschiedene Baumarten

6.8.2 Feature Analyse für LSTM (mit Integrated Gradients)

Die anschließende Feature-Analyse wurde mithilfe der Feature Importance sowie der Integrated Gradients durchgeführt. Dabei zeigte sich (6.10), dass insbesondere die Spektralbänder und die daraus berechneten spektralen Indizes die wichtigsten Einflussgrößen für das Modell darstellen. Zusätzliche Merkmale, wie beispielsweise Monate, hatten hingegen keinen Einfluss auf die Modellperformance.

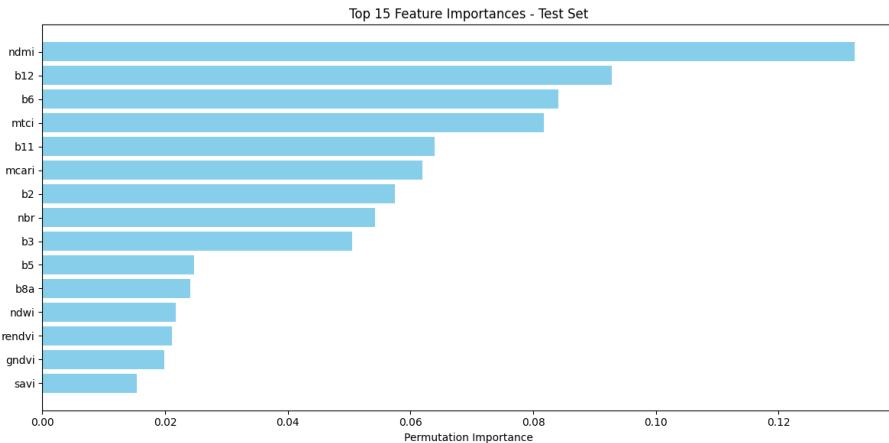


Abbildung 6.9: LSTM Feature Importance

Bei der Anwendung der Integrated Gradients wurde untersucht, welche EingabevARIABLEN über die jeweilige Sequenzlänge hinweg den größten Einfluss auf die Modellentscheidungen haben. Dadurch konnte die zeitliche Bedeutung einzelner Features innerhalb der Sequenzen sichtbar gemacht und die Interpretierbarkeit des LSTM-Modells deutlich verbessert werden.

Zur Veranschaulichung der Ergebnisse wurde bei der Integrated-Gradients-Analyse (IG) für jedes Label ein Beispiel generiert, das die entsprechenden Einflussverteilungen der EingabevARIABLEN zeigt (9.4).

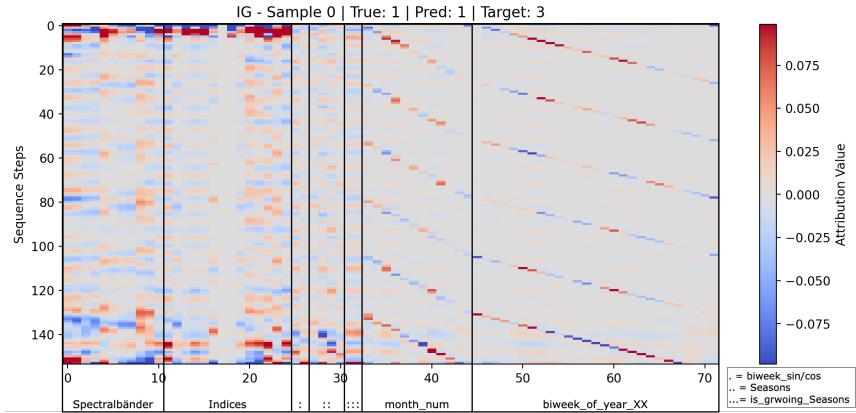


Abbildung 6.10: Aufbau der IG-Heatmap

6.9 Datenreduktion durch Analyse von Zeitintervallen

In diesem Kapitel werden Strategien zur Datenreduktion untersucht, mit dem Ziel, einen optimalen Kompromiss zwischen der Reduzierung des Datenumfangs und der Leistungsfähigkeit des Modells zu finden. Die primäre Motivation liegt in der Optimierung rechenintensiver Prozesse, insbesondere bei der Verarbeitung von Zeitreihen-Pixeldaten. Durch die gezielte Auswahl von relevanten Zeitintervallen soll die Trainings- und Verarbeitungszeit reduziert werden, ohne die Genauigkeit des Klassifikationsmodells signifikant zu beeinträchtigen.

Die Analyse gliedert sich in zwei Hauptuntersuchungen:

- 1. Jahres- und monatsbasierte Analyse:** Hier wird untersucht, wie sich die Modellleistung verhält, wenn nur Daten aus einzelnen Jahren, kumulierten Jahren oder spezifischen Monaten für das Training verwendet werden.
- 2. Analyse gestörter Bäume:** Diese Untersuchung konzentriert sich auf Bäume mit einem bekannten Störungseignis und evaluiert, welches Zeitfenster um das Störungsjahr herum die besten Klassifikationsergebnisse liefert.

Als Basismodell für alle Experimente dient ein XGBoost-Klassifikator.

6.9.1 Jahresbasierte Datenreduktion

Zunächst wurde die Modellleistung auf Basis einzelner und kumulierter Jahre bewertet. (6.11) zeigt die Genauigkeit (Accuracy) eines Modells, das jeweils nur mit den Daten eines einzelnen Jahres trainiert wurde. Es ist deutlich zu erkennen, dass die Leistung stark vom jeweiligen Jahr abhängt. Das Jahr 2020 liefert mit Abstand die besten Ergebnisse, während die Jahre 2017 und 2021 eine merklich geringere Genauigkeit aufweisen.

In einem zweiten Schritt wurden die Jahre kumulativ zum Training herangezogen. Wie in (6.12) dargestellt, verbessert sich die Modellleistung stetig mit der Hinzunahme weiterer Jahre. Der größte Leistungszuwachs wird bei der Erweiterung

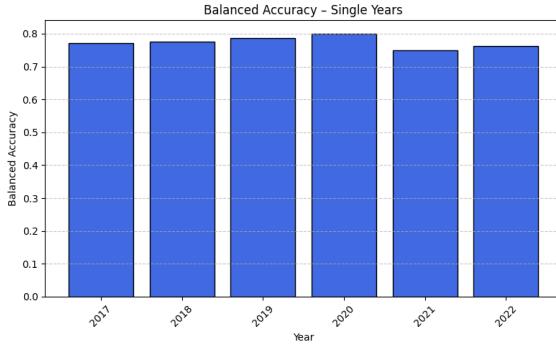


Abbildung 6.11: Genauigkeit bei Training mit einzelnen Jahren.

von 2017 auf die Jahre 2017-2019 beobachtet. Nach der Hinzunahme der Daten aus dem Jahr 2020 flacht der Zuwachs jedoch deutlich ab. Dies deutet darauf hin, dass die Hinzunahme weiterer Daten ab diesem Punkt nur noch einen marginalen Mehrwert liefert („diminishing returns“).

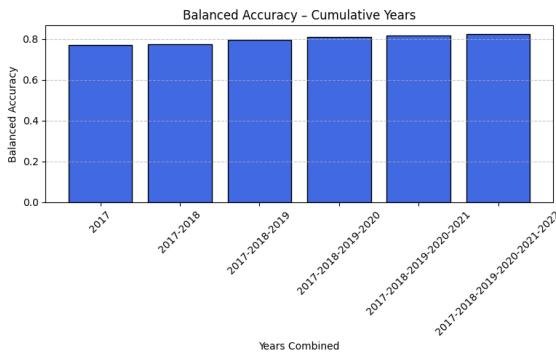


Abbildung 6.12: Genauigkeit bei Training mit kumulierten Jahren.

6.9.2 Monatsbasierte Datenreduktion

Um die Datenmenge weiter zu reduzieren, wurde untersucht, welche monatlichen Intervalle innerhalb eines Jahres den größten Informationsgehalt für das Modell bieten. Für das Jahr 2020 wurde für jede mögliche Intervalllänge (von 1 bis 12 Monaten) das beste zusammenhängende monatliche Intervall ermittelt.

(6.13) visualisiert die Ergebnisse. Die Monate März bis Mai erweisen sich als besonders aussagekräftig und werden in den besten Intervallen fast durchgängig ausgewählt. Die Genauigkeit steigt mit zunehmender Anzahl an Monaten, jedoch wird bereits mit einem Bruchteil der Daten (z.B. 4-5 Monate, ca. 33-42% der Daten) eine hohe Genauigkeit erreicht. Die Leistung stagniert ab etwa 8-9 Monaten, was darauf hindeutet, dass die zusätzlichen Monate nur wenig neue, relevante Informationen enthalten. Die späten Monate (Oktober bis Dezember) scheinen durchweg eine geringere Aussagekraft zu besitzen.

Der marginale Genauigkeitsgewinn, also der zusätzliche Leistungszuwachs pro hinzugenommenem Jahr in einem Intervall, bestätigt diesen Trend (6.14). Während die ersten Jahre einen deutlichen Zuwachs bringen, wird der Gewinn im weiteren

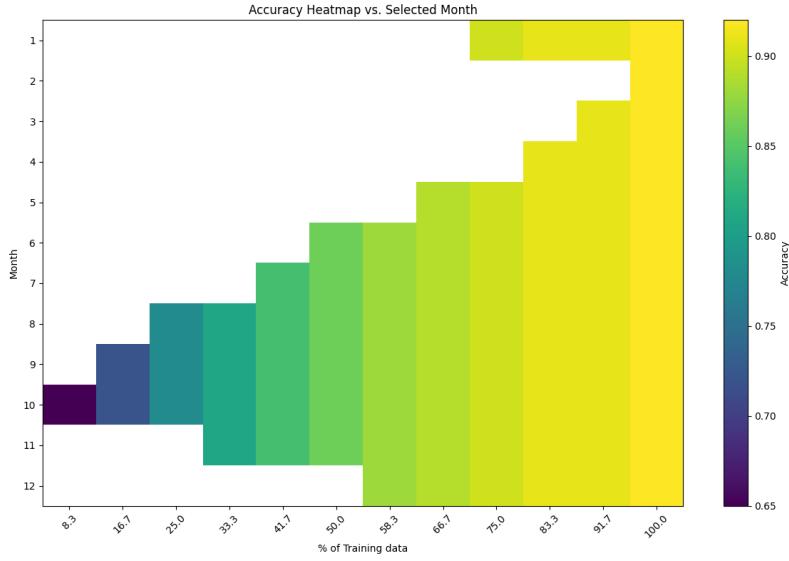


Abbildung 6.13: Genauigkeits-Heatmap für die besten Monatsintervalle (Jahr 2020).

Verlauf minimal. Dies unterstreicht das Potenzial der Datenreduktion: Eine sorgfältig ausgewählte Untermenge der Daten kann eine vergleichbare Leistung bei deutlich geringerem Rechenaufwand erzielen.

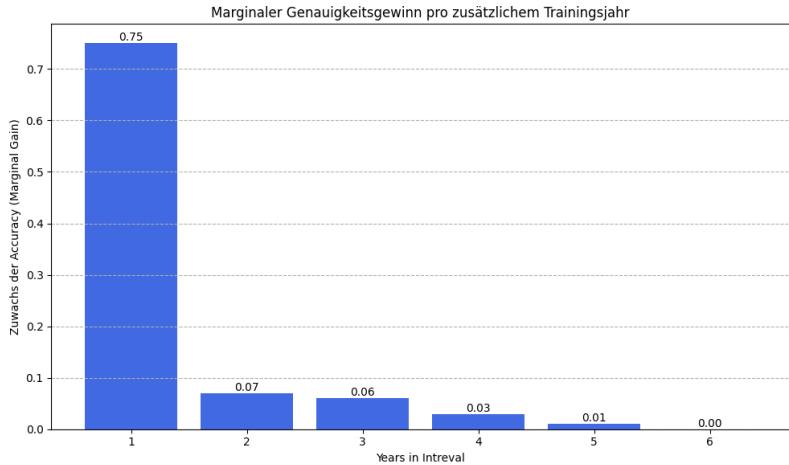


Abbildung 6.14: Marginaler Genauigkeitsgewinn pro zusätzlichem Trainingsjahr.

6.9.3 Analyse um Disturbance Year

Bei der Reduzierung der Datenmengen gilt es, die Daten um das Disturbance Year gezielt zu betrachten, um zu analysieren, wie dieser Zeitraum die Modellergebnisse beeinflusst und welche Effekte sich durch die Extraktion der Daten rund um das Störungsjahr zeigen. Aus dem Plot lässt sich erkennen, dass die Balanced Accuracy in den Jahren vor dem Disturbance Year leicht schwankt und tendenziell höhere Werte erreicht als im eigentlichen Störungsjahr. Dies deutet darauf hin, dass die

Modelle in den Zeiträumen vor der Störung stabilere oder besser erkennbare Muster erfassen konnten.

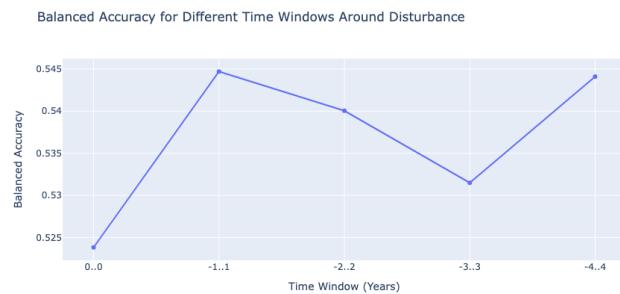


Abbildung 6.15: Modell Performance um disturbance Year

Kapitel 7

Probleme und Lösungen

7.1 Aufgetretene Schwierigkeiten

Eine der Herausforderungen während des Projekts war das Update von VSCode, das zunächst ein zeitintensives Problem darstellte. Die dadurch entstehende Verzögerung konnte jedoch produktiv genutzt werden, um andere Projektaufgaben voranzutreiben.

Die gesamte Vorverarbeitung, einschließlich der Ausreißererkennung, der Processing-Pipeline sowie der Berechnung von Sliding Windows, war auf dem vollständigen Datensatz sehr zeitintensiv. Daher wurde zunächst stets mit einem Daten-Sample gearbeitet, um die Funktionalität der einzelnen Schritte effizient zu testen und zu validieren, bevor sie auf den gesamten Datensatz angewendet wurden.

Kapitel 8

Fazit und Ausblick

8.1 Zusammenfassung der wichtigsten Erkenntnisse

Im Rahmen dieses Projekts wurde ein umfassender Workflow zur Verarbeitung, Analyse und Modellierung von zeitabhängigen SITS-Daten entwickelt. Dabei konnte gezeigt werden, dass insbesondere das bidirektionale LSTM-Modell die besten Validierungsergebnisse lieferte und sich aufgrund seiner hohen Genauigkeit als zentrales Vorhersagemodell eignete.

Die anschließende Feature-Analyse mittels *Feature Importance* und *Integrated Gradients* verdeutlichte, dass vor allem die Spektralbänder und die daraus abgeleiteten Indizes den größten Einfluss auf die Modellvorhersagen haben. Durch die Anwendung interpretierbarer Methoden konnte somit ein tieferes Verständnis über die Einflussfaktoren des Modells gewonnen werden.

8.2 Reflexion über den Projektverlauf

Der Projektverlauf zeichnete sich durch eine strukturierte und teamorientierte Arbeitsweise aus. Innerhalb des Teams wurden Aufgaben gezielt verteilt, sodass alle Mitglieder ihre individuellen Stärken einbringen konnten. Die Kommunikation und Zusammenarbeit verliefen effektiv, wodurch Herausforderungen gemeinsam gelöst und Entscheidungen fundiert getroffen werden konnten.

8.3 Verbesserungsvorschläge

Allgemein:

- Zugriff auf eine GPU für alle Teams
- Mindestens eine Woche früher anfangen, um Ende Oktober auch die Präsentation abgeschlossen zu haben.

Kapitel 9

Anhang

9.1 Abbildungen

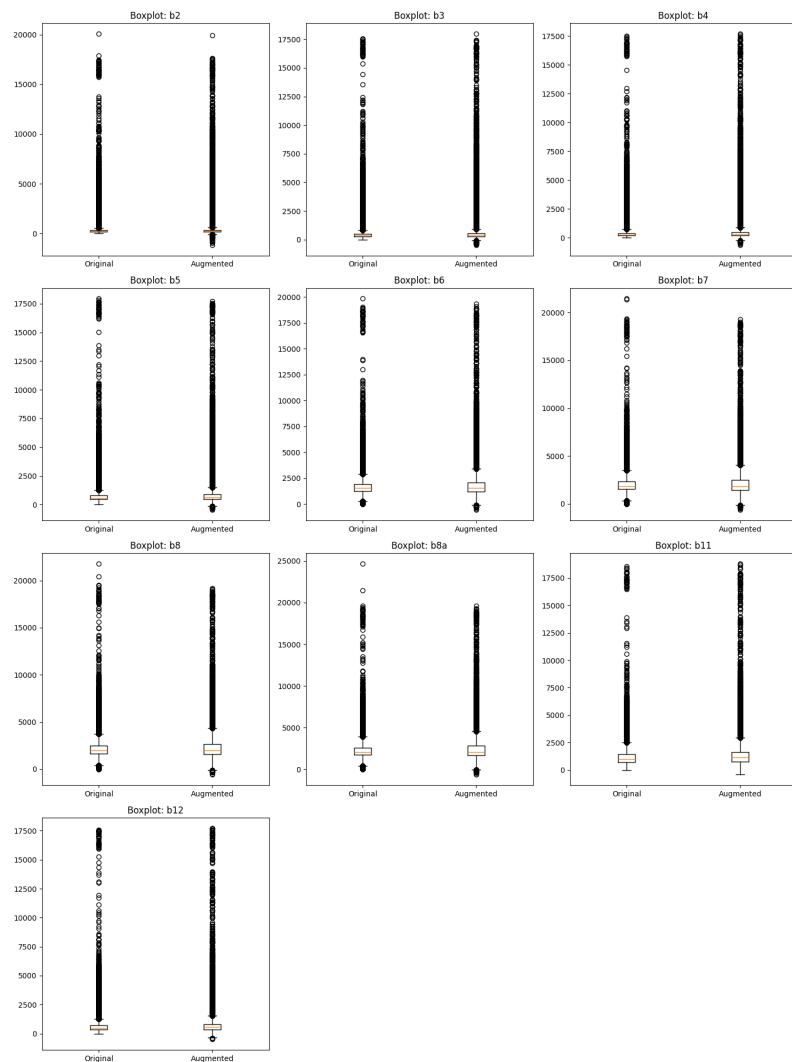


Abbildung 9.1: Boxplots zum Vergleich der Werteverteilungen der einzelnen Spektralbänder vor (Original) und nach der Augmentierung (Augmented).

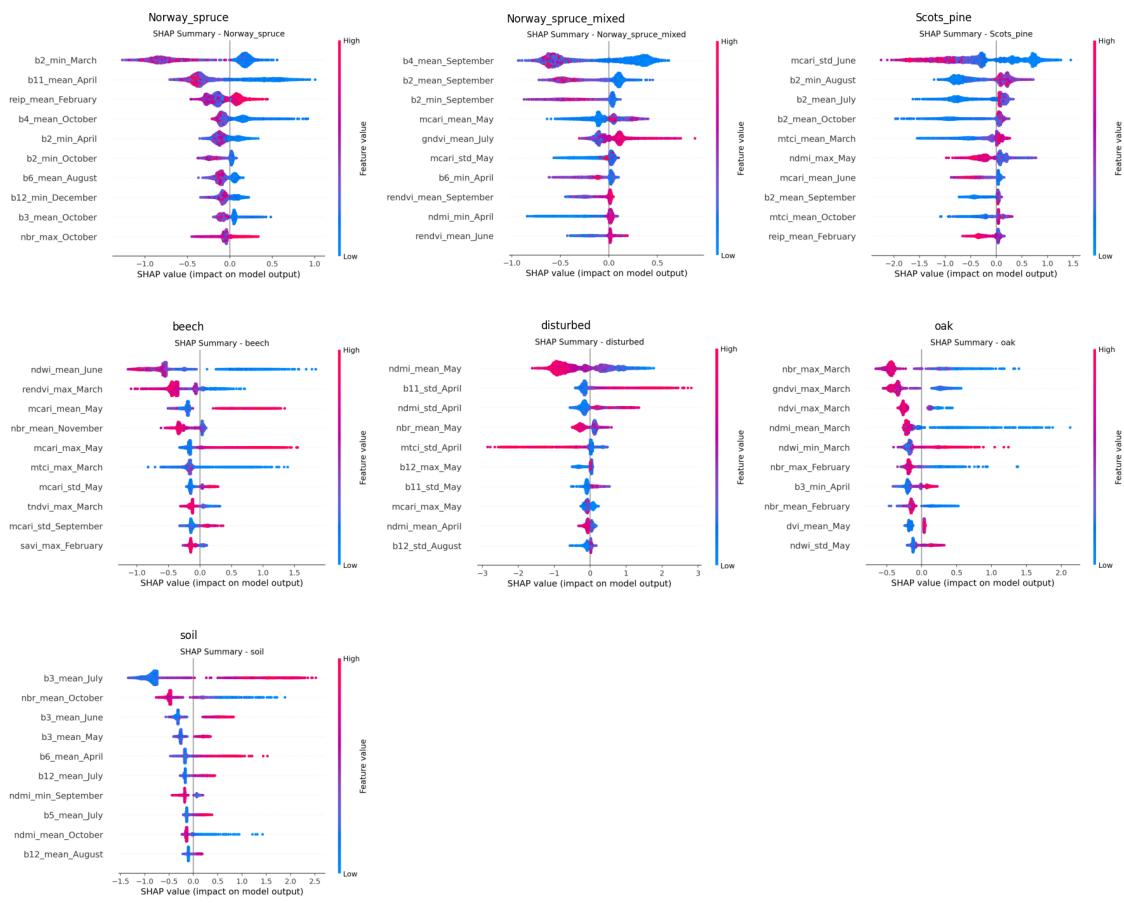


Abbildung 9.2: SHAP: Summary per Species

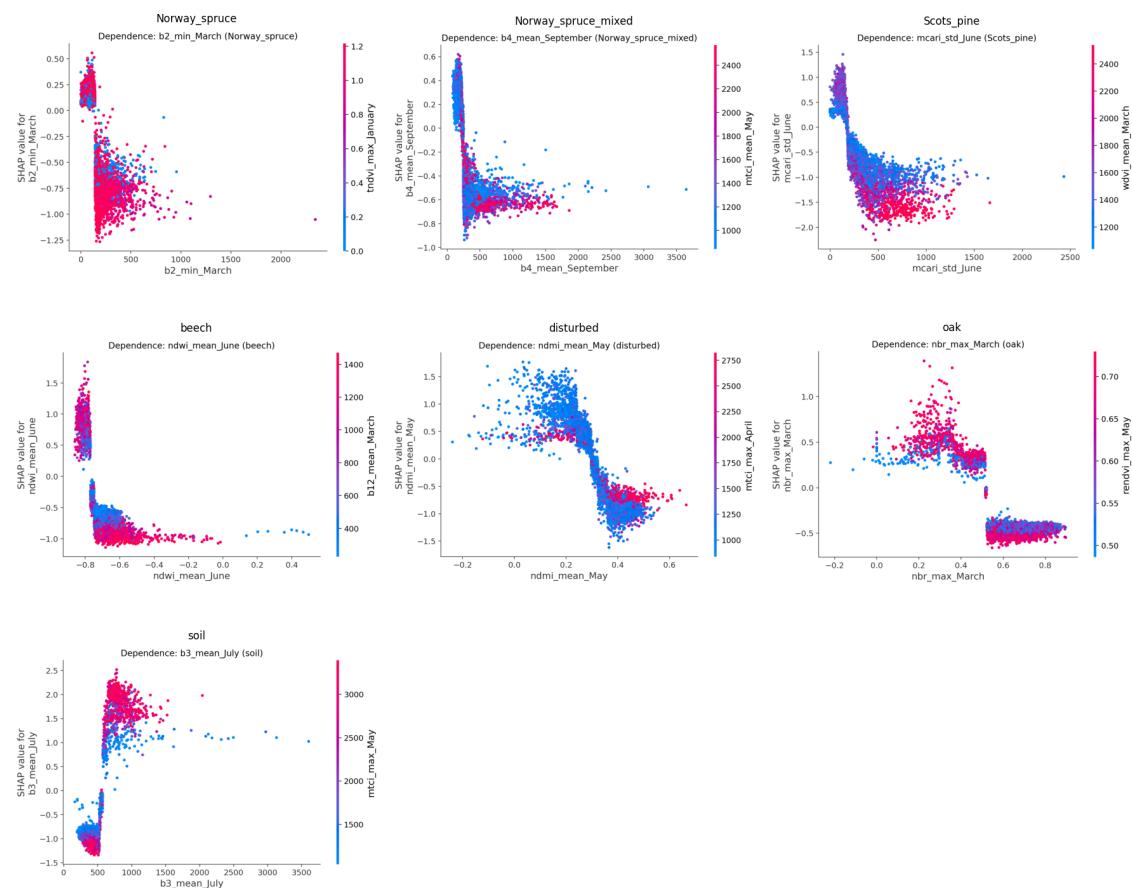


Abbildung 9.3: SHAP: Dependency Plots

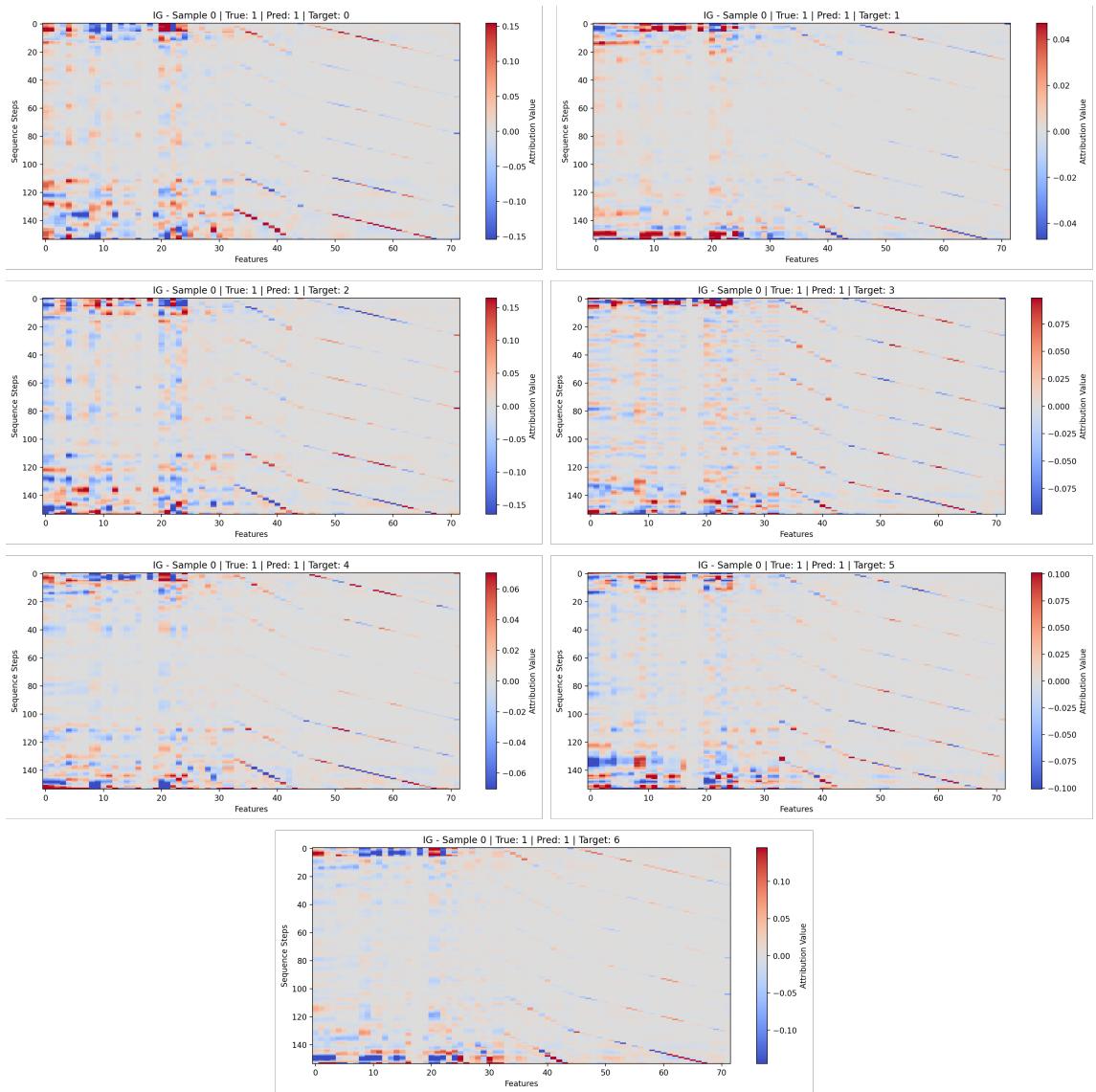


Abbildung 9.4: Integrated Gradients Plots

9.2 Leistungsbeschreibung

9.2.1 Evelyn Rempel

Projektorganisation:

- Strukturierung des Trello-Boards mit Aufgaben basierend auf der Aufgabenstellung
- Einrichtung und Pflege der Gruppenorganisation mit GitHub
- Dokumentation von Ergebnissen und Erkenntnissen in Trello und Google Docs
- Vorbereitung und Nachbereitung von Statusmeetings

Datenanalyse und Data Understanding:

- Durchführung von Data Understanding und Exploratory Data Analysis (EDA)
- Erstellung von Plots und Analyse der Datenqualität
- Untersuchung fehlerhafter Datenpunkte (finden kranker Bäume und finden generell falsch gelabelter Bäume)

Preprocessing und Pipeline-Entwicklung:

- Entwicklung einer Preprocessing Pipeline Klasse zur strukturierten Datenverarbeitung
- Integration einiger Preprocessing Schritte in die Pipeline
- Implementierung einer Aggregation (vereinheitlichte Zeitstempel pro ID), so dass alle Zeitstempel auf den Montag fallen
- Detaillierte Dokumentation der einzelnen Verarbeitungsschritte

Machine Learning:

- Erstellung des Baseline-Modelles, für späteren Vergleich mit komplizierteren Modellen
- Modell zur Erkennung von „disturbed trees“, die aber als gesund gelabelt sind
- Recherche zu Cleanlab -> Entschieden nicht zu verwenden, da bisher das zu viele "cleanen" Modell nicht verbessert
- Feature Importance des Baseline Modells mit SHAP

9.2.2 Alexandru Cozma

Mehrere zusammenhängende Beiträge zum Projekt haben die Datenaufbereitung, Analyse und Modellstabilität merklich verbessert. Zunächst wurden eine Reihe von Analyse-Notebooks erstellt, die explorative Analysen, Visualisierungen und reproduzierbare Vorverarbeitungsschritte dokumentieren. Diese Notebooks umfassen Experimente zur Datenreduktion, verschiedene Glättungsansätze, Analysen zur Augmentation und Hilfsdatensätze zur Pipeline-Entwicklung.

Ein zentraler Beitrag besteht in der Datenreduktionsanalyse, bei der untersucht wurde, wie sich die Leistung eines XGBoost-Modells über Monate und Jahre verändert. Dazu kamen zeitlich geordnete Trainings- und Test-Splits, rollende Fenster und aggregierte Metriken zum Einsatz. Die Auswertungen zeigen Performance-Trends über Zeiträume hinweg (z. B. Genauigkeit und F1-Score im Zeitverlauf) und machen saisonale Effekte sowie langfristige Verteilungen sichtbar – wertvolle Erkenntnisse für Entscheidungen über Retrain-Intervalle und Modellpflege.

Im Bereich Data Augmentation wurden mehrere zeitreihenspezifische Methoden implementiert, darunter kontrolliertes Hinzufügen von Messrauschen, zeitliche Verschiebung und Cropping sowie gezieltes Oversampling seltener Klassen. Diese Maßnahmen reduzieren Klassenungleichgewichte und erhöhen die Robustheit gegenüber Mess- und Umweltvariationen. Alle Methoden sind in die Vorverarbeitungs-Pipeline integrierbar und reproduzierbar dokumentiert.

Neben den größeren Analysen entstanden zahlreiche kleinere, aber wichtige Funktionen: verschiedene Glättungsalgorithmen (gleitender Durchschnitt, exponentielle Glättung), Komponenten des Feature-Engineerings für zeitliche Merkmale (z. B. saisonale Indikatoren und aggregierte Fenster-Features) sowie automatisierte Stationaritätsprüfungen. Diese Bausteine verbessern die Stabilität der Modelle, unterstützen datengetriebene Feature-Entscheidungen und sorgen für eine höhere Datenqualität innerhalb der Pipeline.

Insgesamt tragen die Arbeiten zu besserer Reproduzierbarkeit, erhöhter Modellrobustheit und klaren Empfehlungen für den Betrieb bei – einschließlich geeigneter Retrain-Frequenzen, sinnvoller Augmentationsstrategien und notwendiger Vorverarbeitungsschritte.

9.2.3 Rafael Riesle

Im Rahmen des Projekts wurde zunächst eine umfassende Basisanalyse der Daten durchgeführt, um ein besseres Verständnis der Datengrundlage zu gewinnen und erste Erkenntnisse zu erlangen. Auf Grundlage dieser Analyse wurde eine Ausreißerkennung entwickelt und erfolgreich in die Preprocessing-Pipeline integriert.

Zur strukturierten Datenverarbeitung wurde ein Train/Test/Split eingerichtet, um die Daten getrennt zu bearbeiten und eine saubere Validierung zu gewährleisten. Parallel dazu wurde das Data-Analysis-Notebook erstellt sowie die Analyse des Validierungsdatensatzes weiter vorangetrieben.

Darauf aufbauend erfolgte die Entwicklung und Validierung verschiedener Modellarchitekturen, darunter LSTM-Modelle, Ensemble-Modelle sowie das zuletzt ergänzte PYTS-Modell. Die Arbeitsschritte umfassten den Modellaufbau, das Training sowie die Evaluation anhand unterschiedlicher Kennzahlen und Visualisierungen .

Diese Modelle wurden iterativ verbessert und in unterschiedlichen Pipelines implementiert, die anschließend zu einer gemeinsamen Trainingspipeline zusammengeführt wurden.

Ein weiterer Schwerpunkt lag auf der Hyperparameter-Optimierung mittels Grid Search sowie der intensiven Validierung der Preprocessing-Pipeline in der letzten Projektphase.

Ein bedeutender Einfluss meiner Arbeit lag in der objektorientierten Programmierung (OOP). Durch die Einführung und konsequente Anwendung objektororientierter Prinzipien konnte die Modularität, Wiederverwendbarkeit und Wartbarkeit des Codes erheblich verbessert werden. Dies ermöglichte eine klare Strukturierung der einzelnen Komponenten und eine effiziente Integration der verschiedenen Modelle und Pipelines in ein konsistentes Gesamtsystem.

Zur besseren Nachvollziehbarkeit wurde zudem ein Ablaufdiagramm erstellt, das den Datenfluss und die Struktur des Projekts übersichtlich visualisiert und somit zur Klarheit und Dokumentation des Gesamtprozesses beiträgt.

9.2.4 Gesamt

Wir als Team haben uns hervorragend ergänzt und gegenseitig unterstützt, sodass wir das Projekt gemeinsam erfolgreich abschließen konnten. Dabei konnte jedes Teammitglied seine individuellen Stärken einbringen und zum Gesamterfolg beitragen. Die Arbeitspakete wurden so gestaltet, dass die Workload stets ausgewogen verteilt war und alle Teammitglieder effizient und zielgerichtet zusammenarbeiten konnten. Projektorganisation

9.3 Quellen und Literatur

- PYTS Documentation: <https://pyts.readthedocs.io/en/stable/index.html>
- PyTorch Documentation: <https://docs.pytorch.org/docs/stable/index.html>
- Scikit-learn User Guide: https://scikit-learn.org/stable/user_guide.html
- Sentinel-2 Custom Scripts Index Database: <https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/indexdb/>
- Captum Integrated Gradients Documentation: https://captum.ai/docs/extension/integrated_gradients
- SHAP Documentation: <https://shap.readthedocs.io/en/latest/>