Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

CAI Centre for Artificial Intelligence

Bachelorarbeit (Informatik)

Automatische Erkennung schweizerdeutscher Dialekte anhand von Audiodaten via Phonemtranskriptionen

Autoren	Laura Bolliger Safiyya Waldburger
Hauptbetreuung	Jasmina Bogojeska Mark Cieliebak

Datum	07.06.2024

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)
Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	**Name Studierende:**
Zürich, 7.6.2024	Safiyya Waldburger
Zürich, 7.6.2024	Laura Bolliger

Zusammenfassung

Dialekte sind hauptsächlich ein mündliches Phänomen, sind sich meist sehr ähnlich und es stehen nur beschränkt Ressourcen für das Training eines Natural-Language-Processing-Modells zur Verfügung. Dies sind alles Hürden, die es für die automatische Dialekterkennung zu überwinden gilt. Die vorliegende Bachelorarbeit nahm sich den Herausforderungen an und prüfte einen neuen Ansatz, um schweizerdeutsche Dialekte anhand von Audiodaten zu erkennen. In einem ersten Schritt wurden Audiodaten aus dem SDS-200- und dem STT4SG-350-Korpus mit den neusten Phoneme-Recognizer-Modellen Wav2Vec2Phoneme und MultIPA zu Phonemsequenzen transkribiert. Anschliessend wurden einfache, traditionelle Klassifizierungsalgorithmen mit n-Gramm-Repräsentationen der generierten Phonemsequenzen trainiert. Den grössten Einfluss auf die Erkennungsrate der Modelle hatte eine Konkatenation mehrerer Samples zu längeren Phonemsequenzen. Zudem erwies sich eine n-Gramm-Länge von zwei bis fünf als ideal. Eine zusätzliche Tf-idf-Gewichtung war nur teilweise gewinnbringend. Unter den Classifiern und den Phoneme-Recognizern setzten sich Logistic-Regression und Support-Vector-Machine durch, unter den Phoneme-Recognizern MultIPA. Der beste Classifier – eine Logistic-Regression – erzielte einen F1-Macro-Score von bis zu 94 %. Der Ansatz überzeugte jedoch nicht nur mit hohen Scores, sondern auch mit einer grossen Trainingseffizienz. Die Phonemtranskriptionen mussten nur einmalig erstellt werden und die Classifier waren auf handelsüblichen Laptops innert weniger Minuten trainiert. Zudem ermöglicht der Ansatz mittels klassischer Algorithmen, die Feature-Importance zu untersuchen. Eine erste grobe Analyse ergab, dass hauptsächlich Features, die in einem bestimmten Dialekt klar häufiger oder seltener als in anderen Dialekten vorkommen, für die Identifizierung eines Dialekts relevant sind.

Abstract

Dialects are mainly an oral phenomenon, most of them are very similar and there are only limited resources available to train a natural language processing model. These are obstacles that need to be overcome for the automatic dialect identification. This bachelor's thesis addressed these challenges and tested a new approach to recognize Swiss German dialects from audio data. First audio samples from the SDS-200 and the STT4SG-350 corpus were transcribed into phoneme sequences, using the state-of-the-art phoneme recognizer models Wav2Vec2Phoneme and MultIPA. In a second step simple traditional classification algorithms were trained with n-gram representations of the generated phoneme sequences. The greatest impact on the model performance had the concatenation of several samples into longer phoneme sequences. Also, an n-gram length of two up to five proved to be ideal. An additional tf-idf was only partially beneficial. Among the classifiers logistic regression and support vector machine performed best, and MultIPA prevailed among the phoneme recognizers. The best classifier – a logistic regression – achieved an F1 macro score of up to 94%. However, the approach was not only compelling due to high scores, but also because of its great training efficiency. The phoneme transcriptions only had to be created once and the classifiers were trained on standard laptops within few minutes. Furthermore, the approach using the classical algorithms makes it possible to examine the feature importance. A first rough analysis showed that mostly features that are clearly more frequent or rarer in a certain dialect than in others are relevant for identifying a dialect.

Inhaltsverzeichnis

1. Einleitung

Gesprochene Sprache ist der bevorzugte Kommunikationsmodus der Menschen, wie es Malik in einer Erhebung zum Thema automatische Spracherkennung (Automatic-Speech-Recognition oder ASR) treffend ausdrückt [1]. So ist es von Vorteil, wenn Maschinen mit dieser Kommunikationsform umgehen können. Ein Weg, wie Maschinen gesprochene Sprache verstehen können, führt über ASR, das sich mit der Umwandlung von gesprochener Sprache zu Text beschäftigt. Die Dialekterkennung ist ein Teilgebiet von ASR und eine schwierigere Form der Sprachidentifizierung[1], die die Umwandlung von gesprochener Sprache zu Text unterstützen und verbessern kann [2]. Während Sprachen oder genauer Standardsprachen über grössere Regionen hinweg genutzt werden, zeichnen sich Dialekte durch eine «relativ geringe regionale Ausdehnung» [3] aus. Dies bedeutet, dass ein Dialekt von vergleichsweise weniger Menschen gesprochen wird und nur beschränkt Ressourcen für das Training von Natural-Language-Processing-Modellen vorhanden sind. Zudem sind Dialekte eine hauptsächlich mündliche Form der Kommunikation, Standardsprache wird hingegen auch als Schriftsprache verwendet [3]. Sowohl die geringen Ressourcen als auch die Mündlichkeit machen die automatische Dialekterkennung zu einer herausfordernderen Aufgabe als die Sprachidentifizierung.

Diese Bachelorarbeit widmet sich der automatischen Erkennung schweizerdeutscher Dialekte anhand von Audiodaten mittels Maschine-Learning-Modellen. Dieses Thema wurde bereits in einer vorangehenden Arbeit der Autorinnen untersucht [4]. Der darin verwendete Ansatz über das Finetuning multilingualer Pretrained-Models Whisper und Wav2Vec lieferte jedoch noch keine zufriedenstellenden Resultate. Das Training der Modelle war äusserst rechenintensiv und schien an den erwähnten beschränkten Ressourcen zu scheitern. Es wurde vermutet, dass zu wenig Daten vorhanden waren, damit die grossen Deep-Learning-Modelle wirklich hätten lernen können, dialektspezifische Merkmale von anderen für die Dialekterkennung irrelevanten Merkmalen zu trennen. Somit war es Ziel dieser Arbeit, einen Ansatz zu finden, der besser mit den verfügbaren Ressourcen auskommt und Dialekte besser identifiziert. Dazu wurde die Idee verfolgt, gesprochene Sprache zuerst mit den neusten multilingualen Phoneme-Recognizer-Modellen zu Phonemsequenzen zu transkribieren und erst dann ein möglichst einfaches Dialekterkennungsmodell mit den generierten Transkriptionen zu trainieren. So könnten linguistisch unbedeutende Merkmale wie etwa die Stimme der Sprecher:innen dank des Transkribierens vorgängig herausgefiltert werden, was das eigentliche Klassifizieren vereinfachen könnte.

In den folgenden Kapiteln wird der Ansatz und dessen Potenzial genau erläutert. Im ersten Kapitel wird ein Überblick über die bisherige Forschung zum Thema «Dialekterkennung» gegeben und im darauffolgenden Kapitel die genutzten Schweizerdeutsch-Korpora vorgestellt. Im vierten Kapitel werden sämtliche Komponenten der Methoden dargelegt. Die Kapitel 5 bis 7 widmen sich der konkreten Umsetzung der Methode und der damit durchgeführten Experimente. Im Kapitel 8 wird schliesslich beim besten trainierten Classifier überprüft, welche Merkmale für die Dialekterkennung relevant sind. Zuletzt werden in Kapitel 9 nochmals die Umsetzung und die Resultate verifiziert und in Kapitel 10 folgt die Diskussion der Ergebnisse und ein Ausblick.

[1] Es sei hier von Sprachidentifizierung die Rede, wenn es um die Unterscheidung von Sprachen geht. Spracherkennung wird allgemeiner als die Verarbeitung von Sprache verstanden. Dialekterkennung wird jedoch synonym zu Dialektidentifizierung genutzt. Beides meint die Unterscheidung von Dialekten.

2. Bestehende Arbeiten

Wie bereits in der Einleitung angedeutet, birgt die Dialekterkennung des Schweizerdeutschen an Audiodaten mehrere Herausforderungen. Erstens gilt es, gesprochene Sprache zu verarbeiten. Gesprochene Sprache enthält im Vergleich zu Text zwar zusätzliche Informationen, doch diese machen auch die Verarbeitung komplexer. Zweitens sollen Dialekte unterschieden werden, also Varietäten von Sprache, die nahe beieinander liegen und schwieriger zu unterscheiden sind als beispielsweise Standardsprachen. Zudem sind schweizerdeutsche Dialekte besonders ressourcenarm, da sie von relativ wenigen Menschen gesprochen werden. Das heisst, entsprechende Machine-Learning-Modelle müssen mit wenig Trainingsdaten auskommen.

In der bisherigen Forschung gibt es hauptsächlich drei Ansätze, an diese Herausforderungen heranzugehen. Der Dialekt wird anhand vorhandener Transkriptionen der Audiosamples identifiziert, die Audiodaten werden direkt klassifiziert oder es werden akustische Informationen mit vorhandenen Transkriptionen kombiniert und dann klassifiziert. Mit einem weniger erforschten vierten Ansatz werden die Audiosamples zuerst maschinell transkribiert und dann nur die Transkriptionen zur Klassifizierung verwendet. In den folgenden Abschnitten wird näher auf verschiedene Publikationen zu den vier Ansätzen eingegangen, die für die Methoden dieser Arbeit grundlegend sind.

2.1 Dialekterkennung an vorhandenen Transkriptionen

Nahverwandte Sprachen können mit klassischen Algorithmen wie Support-Vector-Machine (SVM) oder Multinomial-Naive-Bayes bei Text schon länger gut unterschieden werden. Dies zeigen beispielsweise die Resultate des Shared-Tasks «Discriminating between Similar Languages» (DSL) der VarDial Workshops 2016 und 2017 mit einem Weighted-F1-Score von bis zu 92.7 % [5] [6]. Die Unterscheidung von Dialekten anhand von Text beinhaltet zusätzliche Schwierigkeiten. Dialekte sind nicht nur ähnlicher, sondern müssen zuerst möglichst akkurat und einheitlich transkribiert werden, um Text zu erhalten. Im VarDial Workshop 2016 wurde ein zweiter Shared-Task für die Unterscheidung von vier arabischen Dialekten bearbeitet [5]. Die Datensets wurden aus transkribierten Debatten und Diskussionen vom Sender «Al Jazeera» zusammengestellt, wobei die Transkriptionen mit Arabic-Large-Vocabulary-Speech-Recognition generiert wurden. Im Gegensatz zum DSL-Task lag der beste Weighted-F1-Score mit 51.3 % deutlich tiefer [6]. Ein Jahr später wurde im VarDial Workshop 2017 ein neuer Shared-Task hinzugefügt: «German Dialect Identification» (GDI) für manuelle Transkriptionen des ArchiMob-Korpus in vier Dialekten [6] [7]. Es handelt sich dabei um Transkriptionen längerer Interviewsequenzen, die von vier Transkribierer:innen in die «Schwyzerdütschi Dialäktschrift» von Dieth übertragen wurden [8]. Diese Dialektschrift ist nicht als Standardschrift anzusehen und, wie Zampieri et al. in der Zusammenfassung des Workshops anmerken, weniger präzise als phonetische Schrift [6]. Dadurch entsteht ein gewisser Spielraum beim Transkribieren. Die besten Teams erreichten beim Klassifizieren dieser Transkriptionen mit probabilistischen, SVM- oder Multinomial-Naive-Bayes-Classifiern Weighted-F1-Scores von bis zu 66.2 %. Also bessere Ergebnisse als für die arabischen Dialekte im Jahr zuvor. Im VarDial Workshop 2018 wurden diese Ergebnisse an einem aktualisierten Datenset und einem zusätzlichen fünften Überraschungsdialekt verbessert [9]. Mithilfe der HeLI-Methode und 4-Gramm-Repräsentationen erreichte das beste Team einen Macro-F1-Score von 68.6 %.[2]

[2] Es ist anzumerken, dass bei den VarDial-Workshops eingereichte Deep-Learning-Modelle, die hier nicht speziell erwähnt wurden, eher schlechter abschnitten.

2.2 Dialekterkennung an akustischen Features und Transkriptionen

Schon im VarDial Workshop 2017 wurde der Task für das Arabische weiterentwickelt und es wurden zusätzlich zu den Transkriptionen akustische Features zur Verfügung gestellt. Damit erreichte das beste Team mit einer Kernel-Ridge-Regression und n-Gramm-Repräsentationen einen Weighted-F1-Score von 76.3 %. Auch die Teams auf den Plätzen zwei und drei mit einem Voting-Ensemble und einer linearen SVM erreichten F1-Scores von 71.7 % und 69.7 % [6]. Erst im VarDial Workshop 2019 wurde auch das Datenset des Tasks für die schweizerdeutschen Dialekte mit akustischen Features erweitert [10]. Damit gelangen den besten Teams mit SVM-Classifiern oder der HeLi-Methode und n-Grammen mit 74.6 bis 75.9 % ähnlich gute Macro-F1-Scores wie für das Arabische. Daraus lässt sich schliessen, dass zusätzliche akustische Features gewinnbringend sind.

2.3 Dialekterkennung an Audio

Doch was, wenn keine Transkriptionen für das Training zur Verfügung stehen? Entweder werden die Audiodaten direkt verarbeitet oder es werden als Teil der Architektur zuerst Transkriptionen generiert. Ersteres erscheint zunächst eleganter.

Bereits 2008 publizierten Mingliang et al. für chinesische Dialekte einen Ansatz mit einer Convolutional-Support-Vector-Machine, der für chinesische Dialekte Accuracies von 72.78 % bis 92.5 % erreichten [11]. Mit einem Gaussian-Mixture-Model konnten Nour-Eddine und Abdelkader 2015 arabische Dialekte des Maghrebs mit einer Präzision von 80.49 % erkennen [12]. In den jüngeren Jahren zeigte sich auch, dass Dialekte bei Audiosamples auch sehr gut mithilfe von Deep-Learning-Modellen erkannt werden können. Hier können Beispiele genannt werden wie die Convolutional-Neural-Network-Architektur (CNN) von Ahmet et al. 2019 mit einer Accuracy von 90 % für englische Dialekte und 71 % für arabische Dialekte sowie weitere CNN-RNN[3]-Modellen oder reine CNN-Modelle von Azim et al. 2022, Khaled et al. 2022 oder Alrehaili et al. 2023 mit Accuracies von 83 % bis über 90 % für verschiedene arabische Dialekte [13] [14] [15] [16].

2.4 Dialekterkennung an Audiodaten via generierter Transkriptionen

Trotz der Eleganz, Audiodaten ohne «Umweg» zu verarbeiten, lohnt es sich, einen Blick auf Publikationen zu werfen, in denen Transkriptionen als Zwischenschritt erstellt werden und der Dialekt an den Transkriptionen erkannt wird. Tatsächlich ist die Idee, zuerst akustische Signale in schriftliche Symbole umzuwandeln, relativ alt und beispielsweise bei Zissman (1996) zu finden. Dieser beschreibt Techniken, wie Audiodaten in Phonemsequenzen umgewandelt werden können, um daran verschiedene Sprachen zu identifizieren [17]. Obwohl die beschriebenen Techniken selbst veraltet sind, taucht die Idee an sich auch in neuerer Forschung auf. Im Abschnitt 2.1 wurde bereits erwähnt, dass für den Shared-Task zu den arabischen Dialekten im VarDial Workshop 2016 ein Korpus mit generierten Transkriptionen verwendet wurde. Nur war dies noch kein integraler Bestandteil der Dialekterkennungsmodelle.

Işık und Artuner präsentieren 2020 in ihrem Paper ein Modell, mit dem die Audiodaten zuerst mithilfe von Phoneme-Recognizern in Phonemsequenzen transkribiert werden [18]. Mit den Phonemsequenzen wird anschliessend für jeden vorhanden Dialekt ein Long-Short-Term-Memory-Model (LSTM) trainiert, das lernt, Phonemsequenzen seines Dialekts vorherzusagen. Die eigentliche Dialekterkennung wird mit

[3] RNN steht für Recurrent-Neural-Network.

einer Softmax-Regression durchgeführt, die die Performance der einzelnen LSTM für ein Audiosample vergleicht. Der gesuchte Dialekt ist der Dialekt des LSTMs, das die Phonemsequenz für ein Audiosample am besten vorhergesagt hat. Bei einem Korpus mit türkischen Dialekten wurden je nach Dialekt Accuracies von 84.4 % bis 85.5 % erzielt. Ihr Baseline-LogMel-CNN-Modell lag bei Accuracies von 83.4 bis 84 %. Der Umweg über die Phonemsequenzen stellte also eine Verbesserung dar.

Auch Imaizumi et al. konnten sich 2022 ein Weg über Transkriptionen für die Erkennung japanischer Dialekte zu Nutze machen [19]. Ihr Ziel war, ein End-to-End-System zu designen, das sowohl Dialekte erkennt als auch transkribieren kann. Der Vorteil einer Kombination der beiden Tasks «Dialekterkennung» und «Transkription» in einem System sei, dass diese voneinander abhängig seien und voneinander profitieren könnten. Dafür haben die Forscher mit verschiedenen Deep-Learning-Konstellationen aus Transformer-Speech-Encoder, Transformer-Text-Decoder und Dialect-Identification-Layern experimentiert. Das DID2ASR-Modell, das zuerst den Dialekt an einem Audiosample erkennt und mithilfe des erkannten Dialekts das Sample transkribiert, erreichte eine Accuracy von 83.1 % für die Dialekterkennung. Das ASR2DID-Modell, das umgekehrt arbeitet und den Dialekt an den Transkriptionen erkennt, erreichte eine bessere Accuracy von 86.5 %. Das dritte «DID + ASR»-Modell, das simultan arbeitet, schnitt mit 81.8 % am schlechtesten ab. Folglich ist es am gewinnbringendsten für die Dialekterkennung, wenn zuerst eine Transkription aus der gesprochenen Sprache erstellt wird, um daran den Dialekt zu erkennen.

2.5 Wahl des Ansatzes für diese Bachelorarbeit

Für das vorliegende Thema dieser Bachelorarbeit, die Dialekterkennung beim gesprochenen Schweizerdeutsch zu erkennen, sind sowohl die direkte Verarbeitung der Audiodaten als auch der Weg über die Transkriptionen valide. Wie die Analyse der Forschungsliteratur zeigt, haben beide Ansätze Potenzial.

In Vorgängerarbeiten an der ZHAW School of Engineering wurde in den letzten Jahren vor allem untersucht, wie mit den Pretrained-Models Wav2Vec oder Whisper und einem Finetuning für Schweizerdeutsch direkt Audio klassifiziert werden können. Die besten Ergebnisse erreichten Frei und Schneider 2023 mit einem Micro-F1-Score von 64.72 % [20]. Dazu verwendeten sie Wav2Vec und machten ein Finetuning mit den beiden Korpora SDS-200 und STT4SG-250, auf die in Kapitel 3 näher eingegangen wird. In der Projektarbeit der Autorinnen dieser Arbeit wurde das Setting von Frei und Schneider übernommen und mit verschiedenen Zusammenstellungen der Trainingsdatensets experimentiert [4]. Der Score konnte jedoch nicht weiter verbessert werden und kam nicht an die besten Resultate des erwähnten VarDial Workshops 2019 am ArchiMob-Korpus heran (75.9 %), für die Transkriptionen und akustische Features verwendet wurden. In der Bachelorarbeit von Frei und Schneider und in der vorangehenden Arbeit der Autorinnen wurde vermutet, dass zu wenig Daten für ein besseres Finetuning vorhanden sind und die Modelle von nicht-dialektspezifischen Features wie der Stimme der Sprecher:innen abgelenkt werden. Zudem hat der Ansatz mit dem Finetuning von Wav2Vec oder Whisper den grossen Nachteil, dass das Training sehr rechenintensiv ist.

Diese Bachelorarbeit verfolgt nun einen Ansatz über Phonemtranskriptionen mithilfe von vortrainierten Phoneme-Recognizer-Modellen der neusten Forschung in Kombination mit traditionellen Klassifizierungsalgorithmen. Erstens können dank des Transkribierens einige für die Dialekte irrelevante Sprecher:innenfeatures eliminiert werden, was potenziell zu besseren Ergebnissen führen könnte. Zweitens müssen die Transkriptionen nur einmal gemacht werden und nicht für jedes Training und Experiment wiederholt werden. Die Transkriptionen wiederum sind für das Training eines Classifiers einfacher zu

verarbeiten. Des Weiteren haben sich die traditionellen Klassifizierungsalgorithmen für die Dialekterkennung nicht nur bereits bewährt, wie die Abschnitte oben zeigen, sondern lassen sich auch schneller trainieren als grosse Deep-Learning-Modelle. Es werden also sowohl eine Verbesserung der Ergebnisse als auch eine Effizienzsteigerung erwartet.

3. Korpora für gesprochenes Schweizerdeutsch

Es gibt insgesamt fünf grössere Korpora mit Audiosamples für gesprochene schweizerdeutsche Dialekte. Darunter befinden sich zwei Korpora mit je einem Dialekt. Der Radio Rottu Oberwallis Korpus (RRO) enthält insgesamt etwas mehr als 8 Stunden Audiodaten für Walliserdeutsch aus Aufnahmen einer täglichen Newssendung des Radio Rottu Oberwallis [21]. Zwei Walliserdeutschsprecher:innen haben die Aufnahmen transkribiert und sich so gut wie möglich an den im Buch «Wallissertitschi Weerter» [22] definierten Standard für eine Schreibweise des Walliserdeutschen gehalten. Für das Berndeutsche wurde 2021 der Swiss Parliaments Corpus (SPS) publiziert [23]. Dieser enthält 293 Stunden Aufnahmen und hochdeutsche Transkriptionen des Parlaments «Grosser Rat Kanton Bern». Für die Dialekterkennung sind jedoch die Korpora interessanter, die mehrere Dialekte enthalten. Der älteste Korpus, der dieses Kriterium erfüllt, ist der in Abschnitt 2.1 erwähnte ArchiMod-Korpus aus dem Jahr 2016, der 2019 nochmals aktualisiert wurde [7]. Die erste Version enthält 34 Aufnahmen, die zweite 43 Aufnahmen mit ein- bis zweistündigen Interviews über Politik, Liebe und Krieg mit je eine:r Informant:in eines Jahrgangs zwischen 1910 und 1930. Dies sind insgesamt 69 Stunden Audio in der neueren Version. Die Aufnahmen wurden, wie in Abschnitt 2.1 erwähnt, von vier Transkribierer:innen in die Schwyzerdütschi Dialäktschrift übertragen. Zusätzlich zu den Transkriptionen werden ausserdem normalisierte Formen der Wörter meist im Standarddeutschen, Part-of-Speech-Tags und demografische Informationen, darunter auch die Angaben zum Herkunftsort beziehungsweise dem Dialekt, zur Verfügung gestellt. Ein weiterer Korpus ist SwissDial aus dem Jahr 2021, der Audiosamples aus acht grossen Dialektregionen (Aargau, Bern, Basel, Luzern, St. Gallen, Graubünden, Wallis und Zürich) enthält [24]. Der Korpus besteht aus über 26 Stunden Audiodaten mit Sätzen aus Schweizer Zeitungen und Wikipedia inklusive schweizerdeutscher und hochdeutscher Transkriptionen. In den letzten beiden Jahren wurden zudem die beiden Korpora SDS-200 und STT4SG-350 veröffentlich, die mit je 200 und 343 Stunden im Vergleich zum ArchiMob- und SwissDial-Korpus nochmals deutlich mehr Daten enthalten [25] [26]. Gerade wegen der grossen Datenmenge und weil die Korpora bereits in den Vorgängerarbeiten verwendet wurden, kommen letztere in dieser Arbeit zum Einsatz. In den folgenden Abschnitten wird im Detail auf die Korpora eingegangen.

3.1 SDS-200-Korpus

Die 200 Stunden Audiodaten des SDS-200-Korpus setzen sich aus 152'251 Audioclips von 3'816 Sprecher:innen aus der deutschsprachigen Schweiz zusammen [25]. Durchschnittlich sind die Clips 4.8 Sekunden lang und es sind 138'553 verschiedene Sätze im Korpus enthalten. Es wurden also nur wenige Sätze mehrmals gesprochen. Die Clips wurden im Rahmen eines Crowdsourcings mithilfe eines Onlinetools gesammelt. Die Sprecher:innen erhielten jeweils einen hochdeutschen Satz aus einer Zeitung oder dem German-Common-Voice-Korpus und wurden dazu aufgefordert, den Satz in ihren Dialekt zu übersetzen und mit dem Tool aufzunehmen. Zusätzlich mussten die Sprecher:innen Herkunftskanton, Postleitzahl des Herkunftsorts, Alter und Geschlecht angeben. Die Sprecher:innen wurden über

verschiedene Medien akquiriert und nicht speziell selektiert. Ausserdem wurden sie über Wettbewerbe dazu animiert, möglichst viele Samples aufzunehmen. Dabei wurde nicht begrenzt, wie viele Samples ein:e Sprecher:in aufnehmen darf. Um eine gewisse Datenqualität garantieren zu können, mussten die Sprecher:innen ihre Samples jeweils gegenseitig validieren. In einem Postprozessierungsschritt wurden die Daten nochmals gefiltert. In Abbildung 1 wird die Verteilung der Samples unter drei Aspekten gezeigt. Es zeigt sich, dass die Anzahl Samples pro Kanton, Geschlecht und Alter nicht ausgeglichen ist, was beim Experimentieren mit den Daten unbedingt beachtet werden muss.

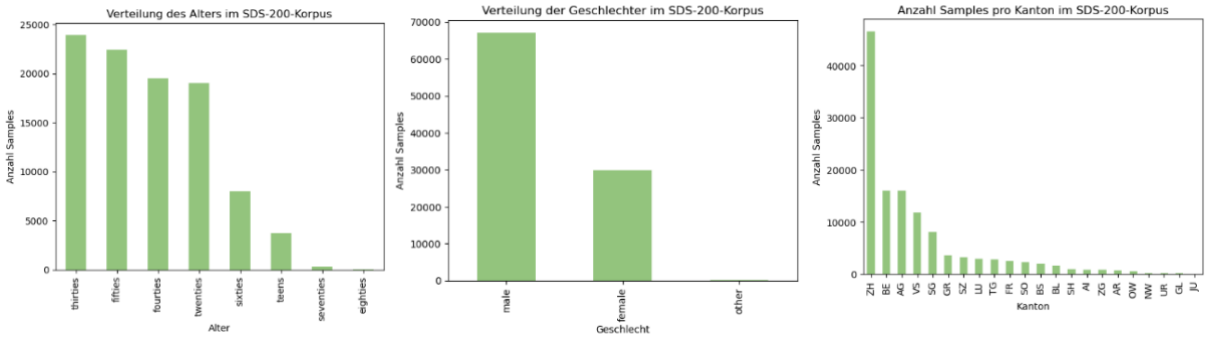

Abbildung 1: Anzahl Samples nach Alter, Geschlecht und Kanton im SDS-200-Korpus

3.2 STT4SG-350

Mit 343 Stunden Audio ist der STT4SG-350-Korpus noch grösser als der SDS-200-Korpus [26]. Es sind 247'527 Samples mit einer durchschnittlichen Länge von circa 5 Sekunden von 316 Sprecher:innen. Unter diesen Samples sind 217'687 einzigartigen Sätze. Es wurden also nur wenige mehrmals gesprochen. Die Daten wurden über das gleiche Verfahren wie beim SDS-200-Korpus gesammelt. Allerdings wurden sie nicht im MP3-Format abgespeichert wie beim SDS-200, sondern als verlustlose FLAC-Dateien. Für die Metadaten wurden Alter, Geschlecht, Postleitzahl des Dialektherkunftsorts und eine von den sieben Dialektregionen Zürich, Ostschweiz, Graubünden, Innerschweiz, Bern, Basel und Wallis aufgenommen. Die Datenqualität wurde am Schluss an zehn zufällig ausgewählten Samples pro Sprecher:in manuell überprüft. Der Korpus wird in Form eines vordefinierten Splits mit einem Trainings-, Validierungs- und Testingsplit zur Verfügung gestellt. Die Splits enthalten keine überschneidenden Sprecher:innen und sind ausgeglichen in der Anzahl Samples pro Dialektregion (siehe Abbildung 2). Die Anzahl Samples pro Sprecher:in wurde im Testdatenset auf 368 und im Trainingsdatenset auf 1'112 limitiert. Es ist ausserdem zu erwähnen, dass im Testdatenset in allen Dialekten die gleichen 3'515 Sätze gesprochen werden. In Abbildung 2 wird die Anzahl Samples pro Merkmal dargestellt. Dies unterstreicht, dass der STT4SG-350-Korpus in Anzahl Samples pro Dialektregion und Geschlecht ausgeglichener ist als der SDS-200. Hier nicht abgebildet ist, dass auch die Anzahl Sprecher:innen pro Dialektregion mit 45 bei allen Regionen ausgeglichen ist.

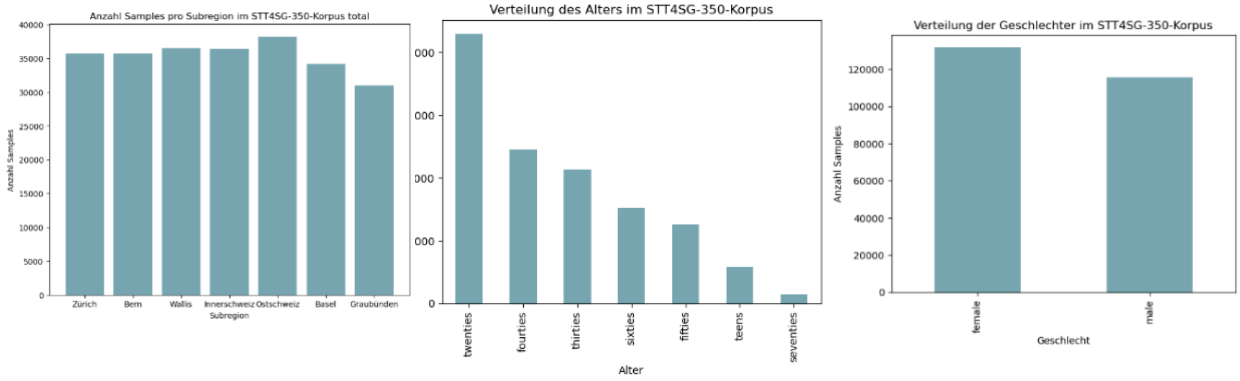

Abbildung 2: Anzahl Samples nach Alter, Geschlecht und Dialektregion im STT-350-Korpus

4. Methode

Ein schematischer Überblick über die verwendete Architektur dieser Arbeit ist in der Abbildung 3 illustriert. Als Erstes werden bereinigte und normalisierte Audiosamples in roher Wellenform von einem von drei Phoneme-Recognizer-Modellen in Phonemsequenzen transkribiert, wie in Abbildung 3 unter «Phoneme-Recognizer» dargestellt. Auf Basis dieser Phonemtranskriptionen wird eine Merkmalrepräsentation in Form von n-Grammen erstellt. Dabei handelt es sich um eine Sequenz von n aufeinanderfolgenden Phonemen (mehr dazu in Abschnitt 4.3). Die n-Gramme werden einem der Classifier übergeben, die in Abbildung 3 unter «Classifier» aufgeführt sind. Dieser berechnet die Wahrscheinlichkeiten, dass die als n-Gramme repräsentierte Phonemtranskription den möglichen Dialekten zugehören respektive teilt die Transkriptionen einem Dialekt zu.

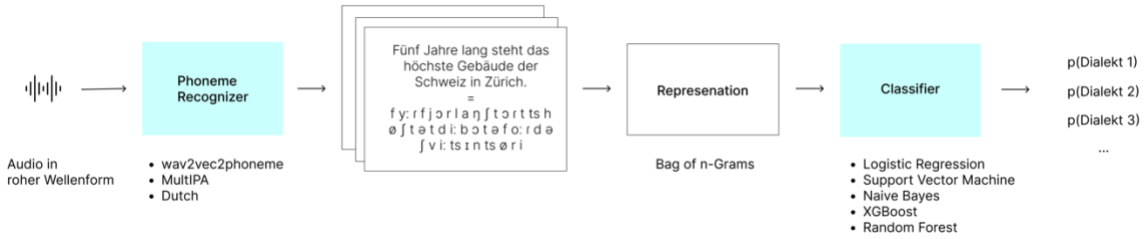

Abbildung 3: Schematische Darstellung der verwendeten Architektur zur automatischen Phonemtranskription und Dialektklassifizierung

Die einzelnen Bestandteile beziehungsweise Komponenten dieser Architektur werden in den folgenden Unterkapiteln konzeptuell erläutert.

4.1 Phoneme-Recognizer

Phoneme-Recognizer erkennen, wie der Name bereits suggeriert, und extrahieren Phoneme aus gesprochener Sprache. Ein Phonem ist die «kleinste bedeutungsunterscheidende sprachliche Einheit» [27] in der gesprochenen Sprache. Diese sind von Phonen (Lauten) zu unterscheiden, die die kleinsten realen Einheiten einer Sprache darstellen, jedoch nicht notwendigerweise bedeutungsunterscheidend sind [28]. So differenziert sich das Wort «rund» durch das Phonem /r/ von «Hund». Das «r» in «rund» kann jedoch unterschiedlich ausgesprochen werden, etwa vorne mit der Zungenspitze oder hinten im Gaumen. Diese unterschiedlichen physikalisch messbaren Laute, die die Bedeutung des Wortes nicht verändern, werden Phone genannt [29] [30]. Die Aussprache eines Wortes kann demnach als Abfolge von einzelnen Phonemen repräsentiert werden, wobei diese Phoneme durch verschiedene Phone realisiert werden können [28] [30].

In dieser Arbeit wurden zur Erkennung und Generierung der Phonemfolgen drei verschiedene Phoneme-Recognizer-Modelle eingesetzt.[4] Alle Modelle basieren auf der Wav2Vec-2.0-Architektur [4] [31] [32] [33]. Um ein tieferes Verständnis für die Phoneme-Recognizer zu entwickeln, wird daher zunächst das zugrunde liegende Wav2Vec-2.0-Modell erläutert. Im Anschluss werden die verwendeten Phoneme-Recognizer beschrieben.

4.1.1 Wav2Vec 2.0

Wav2Vec 2.0 ist ein self-supervised Framework, das latente Repräsentationen aus gesprochener Sprache generieren kann. Die latenten Repräsentationen lernt das Modell während des Pretrainings aus ungelabelten Daten. Das Modell kann danach auf einzelne Sprachen mit gelabelten Daten finegetunt werden, um spezifische Merkmale dieser Sprache zu erfassen.

Die Wav2Vec-2.0-Architektur setzt sich im Wesentlichen aus zwei Teilen zusammen: aus dem Feature-Encoder und dem Transformer-Netzwerk. Die rohen Audiodaten X werden dem Feature-Encoder übergeben und mittels eines mehrschichtigen Convolutional-Networks in latente Repräsentationen Z umgewandelt werden. Die latenten Repräsentationen Z erfassen die grundlegenden Merkmale des Audiosignals. Dabei entspricht jede Repräsentation einem Zeitschritt und deckt ungefähr 25 Millisekunden des Audiosignals ab. Bevor die Repräsentationen dem Transformer-Netzwerk übergeben werden, werden sie quantisiert. Dabei wird die gesprochene Sprache, also das kontinuierliche Signal, in diskrete Zustände umgewandelt.

Die quantisierten latenten Repräsentationen Z werden nun dem Transformer-Netzwerk übergeben. Je nach Version des Modells wird eine bestimmte Anzahl dieser Transformer-Blöcke eingesetzt. Der Transformer erstellt die für das Finetuning relevanten kontextualisierten Repräsentationen C, die kontextuelle Informationen aus der gesamten Eingabesequenz enthalten, wie beispielsweise die Position eines Wortes im Satz oder die Abhängigkeit zwischen verschiedenen Wörtern [34] [35]. Ein detaillierter Beschrieb der Transformer-Architektur kann in der vorherigen Arbeit «Automatische Erkennung von Dialekten (Schweizerdeutsch und Englisch)» in Abschnitt 2.1.2 Transformer-Modelle nachgelesen werden [4].

Ein wesentlicher Bestandteil des Pretrainings in Wav2Vec-2.0 ist das Contrastive-Learning, bei dem ein Teil der Zeitschritte der latenten Repräsentationen maskiert wird. Das Ziel des Contrastive-Learning besteht darin, dass das Modell die korrekte quantisierte latente Repräsentation aus den umgebenden, nicht

[4] Die Modelle wurden unverändert vom Hugging-Face-Hub übernommen; eine Plattform für vortrainierte Modelle und weitere Tools [89].

maskierten Zeitschritten vorhersagen kann [35]. Auf diese Weise lernt das Modell selbstständig aus den umgebenden Daten, ohne auf gelabelte Daten angewiesen zu sein, was als self-supervised bezeichnet wird.

Bei der Modellvariante Wav2Vec-2.0 XLSR-53, die von den Phoneme-Recognizern Wav2Vec2Phoneme und MultIPA verwendet wird, wurden Trainingsdaten aus insgesamt 53 Sprachen mit 56'000 h Audio eingesetzt [34] [31]. Diese Modellvariante verfügt somit über sprachübergreifende latente Repräsentationen [36].

4.1.2 Wav2Vec2Phoneme

Damit das Modell für die Aufgabe der Phonemerkennung finegetunt werden kann, wird dem Modell ein zusätzlicher Klassifikationslayer hinzugefügt. Das Finetuning erfolgt auf Phonemtranskriptionen, mit dem Ziel Phoneme aus den Trainingssprachen zu erkennen. Dabei werden die aus dem Pretraining des Modells XLSR-53 gewonnenen latenten Repräsentationen genutzt. Während des Finetunings werden die Gewichte des Feature Encoders eingefroren, während die Gewichte des Transformer-Netzwerks nach 10'000 Updates angepasst werden. Dies unterstützt das Modell bei der Erfassung phonologischer Merkmale [31].

Eine wesentliche Herausforderung während des Trainings besteht darin, dass die Eingabelänge eines Audiosamples nicht der Phonemtranskription entspricht. Dieses Problem wird mithilfe des Connectionist-Temporal-Classification-Loss, abgekürzt CTC-Loss, überbrückt. Während des Trainings berechnet der CTC-Loss für jede vorhergesagte Phonemtranskription die Wahrscheinlichkeit, dass es sich um die tatsächliche Phonemtranskription handelt. Das Modell lernt dabei mit variablen Zuordnungen umzugehen [31] [37].

Eine weitere Herausforderung ist es, die Phoneme einer Zielsprache basierend auf allen im Training gelernten Phonemen zu erkennen. Dafür wird ein sprachenübergreifendes Phonemvokabular eingesetzt. Dies beinhaltet sogenannte globale Phoneme – phonologische Einheiten – die unabhängig von spezifischen Sprachen sind. Um die gelernten Phoneme aus den Trainingssprachen auf die Zielsprache abzubilden, verwendet das Modell artikulatorische Merkmale. Diese Merkmale bestehen aus Eigenschaften von Lauten wie beispielsweise Nasalität und tragen zu einer sprachübergreifenden Vergleichbarkeit der Phoneme bei. Mithilfe dieser Merkmale werden die gelernten Phoneme mit einem Mapping-Mechanismus auf die Zielsprache zugeordnet. Nach dieser Zuordnung wird ein Sprachmodell genutzt, um die Phonemsequenzen zu generieren. Dabei werden relevante Informationen der Zielsprache einbezogen, um eine kohärente Sequenz von Phonemen in Relation zum Kontext der Sprache sicherzustellen [31].

Das Modell wurde auf Audiosamples aus den Korpora Multilingual LibriSpeech, CommonVoice und BABEL trainiert (siehe Abbildung 4). Die entsprechenden Phonemtranskriptionen zu den Audiosamples wurden durch automatisierte Graphem-to-Phoneme-Tools (g2p-Tools) wie ESpeak und Phonetisaurus generiert [31] [38] [39]. Diese wurden basierend auf den gegebenen Texttranskriptionen der Audiosamples erzeugt. Für die Symbole der Phoneme wird das standardisierte International Phonetic Alphabet (IPA) verwendet [31].

An dieser Stelle sei hervorzuheben, dass das die verwendete Version des Modells in dieser Arbeit nur auf dem CommonVoice-Datensatz finegetunt wurde und es sich um einen Checkpoint des Wav2Vec2Phoneme-Modells handelt [40]. Zum einen ist keine weitere öffentlich Version zugänglich und zum anderen ist der Einsatz dieser Version insofern legitim, als dass der CommonVoice-Datensatz die grösste Anzahl Sprachen umfasst, die geographisch nahe beim Schweizerdeutschen liegen.

Split	Languages
CommonVoice (CV)	
train	Esperanto(eo), Lithuanian(lt), Welsh(cy), Tamil(ta), Swedish(sv-SE), German(de), English(en), Oriya(or), Hindi(hi), Persian(fa), Japanese(ja), Assamese(as), Indonesian(id), Catalan(ca), Spanish(es), French(fr), Portuguese(pt), Arabic(ar), Chinese(zh-CN), Chinese(zh-TW), Turkish(tr), Estonian(et), Hungarian(hu), Russian(ru), Czech(cs)
dev	Italian(it)
test	Basque(eu), Interlingua(ia), Latvian(lv), Georgian(ka), Irish(ga-IE), Dutch(nl), Greek(el), Punjabi(pa-IN), Romanian(ro), Maltese(mt), Chinese(zh-HK), Tatar(tt), Finnish(fi), Slovenian(sl), Polish(pl), Kirghiz(ky)
BABEL (BB)	
train	**Amharic(am)**, **Bengali(bn)**, Cebuano(ceb), Igbo(ig), Haitian(ht), **Javanese(jv)**, Mongolian(mn), Swahili(sw), Tamil(ta), **Vietnamese(vi)**, Assamese(as), Dholuo(luo), Guarani(gn), Kazakh(kk), Pashto(ps), **Georgian(ka)**, Tagalog(tl), Telugu(te), Turkish(tr), **Zulu(zu)**
dev	CV-Italian(it)
test	Cantonese(yue), Lao(lo)

Abbildung 4: Für das Finetuning verwendete Sprachen aus den Korpora CommonVoice (CV), BABEL und Multilingual LibriSpeech (MLS) (fettgedruckt) [31]

4.1.3 MultIPA

MultIPA basiert wie Wav2Vec2Phoneme auf der XLSR-53-Variante des Wav2Vec 2.0-Modells und ist eine Weiterentwicklung von Wav2Vec2Phoneme. Im Gegensatz zu Wav2Vec2Phoneme, das auf einer grossen Menge automatisch generierter Transkriptionen trainiert wurde, wurde für MultIPA ein kleineres, aber qualitativ hochwertigeres Trainingsset verwendet. Dieses Trainingsset besteht aus sieben Sprachen aus dem CommonVoice 11.0-Korpus: Japanisch, Polnisch, Maltesisch, Ungarisch, Finnisch, Griechisch und Tamil. Diese Sprachen wurden ausgewählt, weil sie eine konsistente Zuordnung von Orthografie zu Aussprache aufweisen. Diese Regelmässigkeit in der Schreibweise der Wörter und deren Aussprache führt zu einer höheren Genauigkeit der Transkriptionen. Dies ist insofern relevant, da basierend auf den Transkriptionen die Phonemtranskriptionen semiautomatisch mit dem Internationalen Phonetische Alphabet (IPA) erstellt werden. Mit semiautomatisch ist gemeint, dass auch manuelle Überprüfungen und Korrekturen vorgenommen wurden. Im Gegensatz zu Wav2Vec2Phoneme werden die komplette Liste der IPA-Zeichen sowie deren möglichen Kombinationen verwendet. Diese Kombinationen beinhalten unter anderem mehrbuchstabige Laute wie beispielsweise «au» oder «eu». Zudem wurden Wav2Vec2Phoneme eigens entwickelte Tools sowie ein Tool namens Epitran eingesetzt. Deren Qualität wurde sorgfältig hinsichtlich ihrer Verlässlichkeit geprüft. So konnte MultIPA Ergebnisse erzielen, die mit denen von Wav2Vec2Phoneme vergleichbar oder teils besser sind, obwohl deutlich weniger Trainingsdaten verwendet wurden [32] [41].

4.1.4 Dutch

Das Modell Wav2Vec2-base-960h-phoneme-reco-dutch, in dieser Arbeit abgekürzt als Dutch, wurde ausschliesslich in der Voranalyse der Daten eingesetzt (siehe Unterkapitel 7.1).

Der Phoneme-Recognizer Dutch basiert, auf einer kleineren Ausführung des Wav2Vec-2.0-Modells, die halb so viele Transformer-Blöcke wie die Variante XLSR-53 enthält und mit 960 Stunden englischen

Audiodaten aus dem Korpus Librispeech vortrainiert wurde [40] [33]. Das Modell wurde im Anschluss auf niederländischen Audiodaten von Common Voice auf die Aufgabe der Phonemerkennung finegetunt [33]. Im Gegensatz zu Dutch, wurden Wav2Vec2Phoneme und MultIPA wurden mit mehreren Sprachen finegetunt.

4.2 Classifier

In diesem Abschnitt werden die in dieser Arbeit eingesetzten Classifier konzeptuell beschrieben. Es wurden klassische erprobte Classifier gewählt [42]. Insbesondere die ersten drei Classifier Multinomial-Naive-Bayes, Logistic-Regression und Support-Vector-Machine haben sich im Bereich der Textverarbeitung bewährt. Die Beschreibung der Funktionsweise der Classifier beschränkt sich auf die Aspekte, die für die Multiklassenklassifizierung, wie sie in dieser Arbeit durchgeführt wurde, relevant sind.

4.2.1 Multinomial-Naive-Bayes

Multinomial-Naive-Bayes, im Weiteren abgekürzt als NB, ist ein wahrscheinlichkeitsbasierter Algorithmus, der in zahlreichen Anwendungen für die Textklassifizierung eingesetzt wird [42]. Er berechnet die Wahrscheinlichkeitsverteilung $p(X, y_k)$ für die Daten und geht von der Annahme aus, dass alle Merkmale (in dieser Arbeit n-Gramme) einer Klasse unabhängig voneinander sind. Die Merkmale werden folglich als ein sogenannter «Bag of Features» behandelt. Diese Annahme wird als «naiv» bezeichnet, da sie in der Realität oft nicht zutrifft, insbesondere was die natürliche Sprache betrifft. Ziel des Algorithmus ist es, die Wahrscheinlichkeit der Klasse y_k für gegebene Datenmerkmale X, also die bedingte Wahrscheinlichkeit $p(y_k|X)$, zu berechnen. Für die Berechnung wird das Bayes'sche Theorem herangezogen:

$$p(y_k|X) = \frac{p(y_k)\, p(X|y_k)}{p(X)}$$

Formel 1: Bays'sches Theorem. Angepasst aus [42].

Wobei [5]:

- $p(y_k|X)$ ist die Wahrscheinlichkeit einer Klasse y_k gegeben die Merkmale X.
- $p(X|y_k)$ ist die Wahrscheinlichkeit für die Merkmale X gegeben die Klasse y_k.
- $p(y_k)$ ist die Wahrscheinlichkeit der Klasse y_k. Für die Dialektklassifizierung, bei der jeder Dialekt gleichwahrscheinlich ist, gilt $p(y_k) = \frac{1}{K}$, wobei K für die Anzahl Dialekte steht. Aus diesem Grund handelt es sich auch um eine multinomiale Verteilung.
- $p(X)$ ist die Wahrscheinlichkeit des Auftretens von X in allen Klassen und dient als Normierungsfaktor.

[5] Für eine mathematisch exakte Darstellung müssten die Definitionsbereiche der jeweiligen Variablen definiert werden. In dieser Arbeit wird jedoch darauf verzichtet, da es in erster Linie um das konzeptuelle Verständnis geht.

Mit der Annahme, dass die Merkmale X einer Klasse unabhängig voneinander sind, kann demnach $p(X|y_k)$ als Produkt der Einzelwahrscheinlichkeiten der Merkmale berechnet werden:

$$p(X|y_k) = \prod_i p(x_i|y_k)$$

Formel 2: Produkt der Einzelwahrscheinlichkeiten. Angepasst aus [42].

Hierbei steht x_i für die relative Häufigkeit dieses Merkmals in der Klasse y_k.

Kommt ein Merkmal in einer Klasse während des Trainings nicht vor, erhält es die Wahrscheinlicht 0. Dies führt dazu, dass die gesamte Wahrscheinlichkeit als Produkt aller Merkmale ebenfalls auf 0 gesetzt werden würde. Dies kann umgangen werden, indem ein kleiner konstanter Wert a zu allen Wahrscheinlichkeiten addiert wird. Dies ist eine sogenannten Glättungstechnik gesprochen, die sicherstellt, dass stets eine kleine positive Wahrscheinlichkeit resultiert [42] [43].

Angewendet auf die Dialekterkennung berechnet der NB die Wahrscheinlichkeiten, dass eine Phonemtranskription einem Dialekt angehört. Dafür werden die gelernten Wahrscheinlichkeiten der Features respektive n-Grammen multipliziert. Der Dialekt mit der höchsten Wahrscheinlichkeit wird als Vorhersage zurückgegeben.

4.2.2 Multinomiale-Logistic-Regression

Die multinomiale logistische Regression, im Weiteren als LR abgekürzt und auch als Softmax-Regression bekannt, ist eine Erweiterung der klassischen binären logistischen Regression, die für die Klassifikation mehrerer Klassen eingesetzt wird und auch im Bereich der Textklassifikation erprobt ist [42]. Sie berechnet die Wahrscheinlichkeiten, dass n Merkmale zu einer Klasse j gehören.

Für k Klassen werden zunächst lineare Kombinationen der eingegebenen Merkmale X berechnet, wobei n für die Anzahl Merkmale steht:

$$\begin{aligned} s^1 &= b^1 + \omega_1^1 x_1 + \omega_2^1 x_2 + \cdots + \omega_n^1 x_n \\ s^2 &= b^2 + \omega_1^2 x_1 + \omega_2^2 x_2 + \cdots + \omega_n^2 x_n \\ &\vdots \qquad\qquad \vdots \\ s^k &= b^k + \omega_1^k x_1 + \omega_2^k x_2 + \cdots + \omega_n^k x_n \end{aligned}$$

Formel 3: Scores aus linearen Kombinationen der Merkmale. Angepasst aus [42].

Hierbei repräsentieren ω_n^k die Koeffizienten und b^k die Biases für jede Klasse. Diese linearen Kombinationen s^k repräsentieren die nicht normalisierten Wahrscheinlichkeiten für jede Klasse, auch als Logits oder Scores bezeichnet. Jede Klasse erhält somit einen eigenen Satz von Koeffizienten.

Anschliessend wird jeder Score s^k mithilfe der folgenden Softmax-Funktion in Wahrscheinlichkeiten transformiert:

$$\sigma(s^j) = \frac{e^{s^j}}{e^{s^1} + e^{s^2} + \cdots + e^{s^k}}$$

Formel 4: Softmax-Funktion für die Transformation der Scores in Wahrscheinlichkeiten. Formel aus [42].

Hierbei ist s^j der zuvor berechnete Score einer spezifischen Klasse j. Das Exponenzieren der Logits führt dazu, dass die berechneten Wahrscheinlichkeiten keine negativen Werte annehmen, was eine erforderliche Bedingung für die Wahrscheinlichkeitsberechnungen ist. Ausserdem werden im Nenner alle zuvor berechneten und exponenzierten Scores aufsummiert, sodass die Summe aller Wahrscheinlichkeiten stets 1 ergibt [42].

Während des Trainings werden die Koeffizienten für jede Klasse angepasst, sodass die vorhergesagten Wahrscheinlichkeiten möglichst genau den tatsächlichen Werten, also den Klassenlabels entsprechen. Dies wird durch die Maximierung der Wahrscheinlichkeiten für die jeweilige Klasse j erreicht:

$y = j$ sodass $\sigma(b^j + \omega_1^j x_1 + \cdots + \omega_n^j x_n)$ maximiert wird [42].

Um dieses Ziel zu erreichen, wird während des Trainings der Cross-Entropy-Loss verwendet. Dieser wird wie folgt definiert:

$$loss = \left.\begin{array}{ll} -log\big(\sigma(s^1)\big) & \text{wenn } y_{true,1} = 1 \\ -log\big(\sigma(s^2)\big) & \text{wenn } y_{true,2} = 1 \\ -log\big(\sigma(s^3)\big) & \text{wenn } y_{true,3} = 1 \\ \vdots & \\ -log\left(\sigma\big(s^k\big)\right) & \text{wenn } y_{true,k} = 1 \end{array}\right\} = -y_{true,1} log(\sigma(s^1)) - \cdots - y_{true,k} log(\sigma(s^k))$$

Formel 5: Cross-Entropy-Loss ausführlich. Angepasst aus [42].

Dies lässt sich zusammenfassen als:

$$loss = -\sum_{k=1}^{K} y_{true,k}\, log(\sigma(s^k))$$

Formel 6: Cross-Entropy-Loss kompakt. Angepasst aus [42].

Wobei gilt: $y_{true,j} = 1$, wenn die Klasse j die wahre Klasse ist, sonst 0.

Der Cross-Entropy-Loss bestraft Fehlklassifikationen stark. Wenn das Modell eine geringe Wahrscheinlichkeit für eine tatsächliche Klasse vorhersagt, wird der Loss erhöht. Dies sorgt für eine schnelle Konvergenz, da das Modell stark für sichere, aber falsche Vorhersagen bestraft wird [44].

Für den gesamten Loss wird der Durchschnitt des Losses über alle m Trainingsdaten berechnet:

$$loss = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} {y^k}_{true,i}\ log(\sigma({s_i}^k))$$

Formel 7: Cross-Entropy-Loss über alle Trainingsdaten hinweg. Angepasst aus [42].

Im Trainingsprozess werden die Modellparameter ω iterativ durch Optimierungsalgorithmen wie den Gradient Descent oder den Newton-Conjugate-Gradient angepasst, um den durchschnittlichen Cross-Entropy-Loss über alle Trainingsdaten zu minimieren. Die Minimierung dieser Loss-Funktion wird als Optimierungsproblem bezeichnet [42] [45] [46]. Auf die Details der iterativen Optimierungsalgorithmen, von denen es einige gibt, wird an dieser Stelle nicht weiter eingegangen.

4.2.3 Support-Vector-Machine

Support-Vector-Machine, im Weiteren als SVM abgekürzt, wurde bereits mehrfach erfolgreich für die Dialekterkennung eingesetzt, wie Abschnitt 2.1 zeigt. Das Ziel von SVM für eine Klassifikationsaufgabe ist es, eine sogenannte Trennlinie zu finden, die die Datenpunkte zweier Klassen mit grösstmöglichem Abstand voneinander trennt. Diese Trennlinie wird als Hyperplane bezeichnet. Ein grösserer Abstand zwischen den Klassen ist erstrebenswert, da er zu einer robusteren Klassifikation beiträgt und folglich die Fähigkeit des Modells erhöht, auf neue Daten zu generalisieren.

Die Hyperplane trennt einen n-dimensionalen Raum in zwei Hälften, wobei n der Anzahl Merkmale entspricht. Im Falle eines zweidimensionalen Raums reduziert sich die Hyperplane zu einer einfachen Linie [47]. Die Gleichung einer Hyperplane wird definiert als:

$$w^T x + b = 0$$

Formel 8: Hyperplane. Übernommen aus [48].

wobei,

- w der Gewichtungsvektor ist, der senkrecht zur Hyperplane steht und den es im Trainingsprozess zu lernen gilt,

- x der Vektor ist, der die Merkmale bezeichnet und der die Dimension des Raums bestimmt,
- b der Bias ist, der für die Verschiebung der Hyperplane verantwortlich ist.

Um die Datenpunkten basierend auf ihrer Position relativ zur Hyperplane in zwei Klassen einzuteilen, wird folgende Regel festgelegt:

$$y = \begin{cases} +1 \text{ wenn } w^T x + b \geq 0 \\ -1 \text{ wenn } w^T x + b < 0 \end{cases}$$

Formel 9: Klassifikationsregel basierend auf Hyperplane-Gleichung. Übernommen aus [48].

Ein Punkt wird der Klasse +1 zugeordnet, wenn er auf oder oberhalb der Hyperplane liegt. Kommt er unterhalb der Hyperplane zu liegen, wird er der Klasse -1 zugeordnet (siehe in Abbildung 5 blaue beziehungsweise rote Punkte).

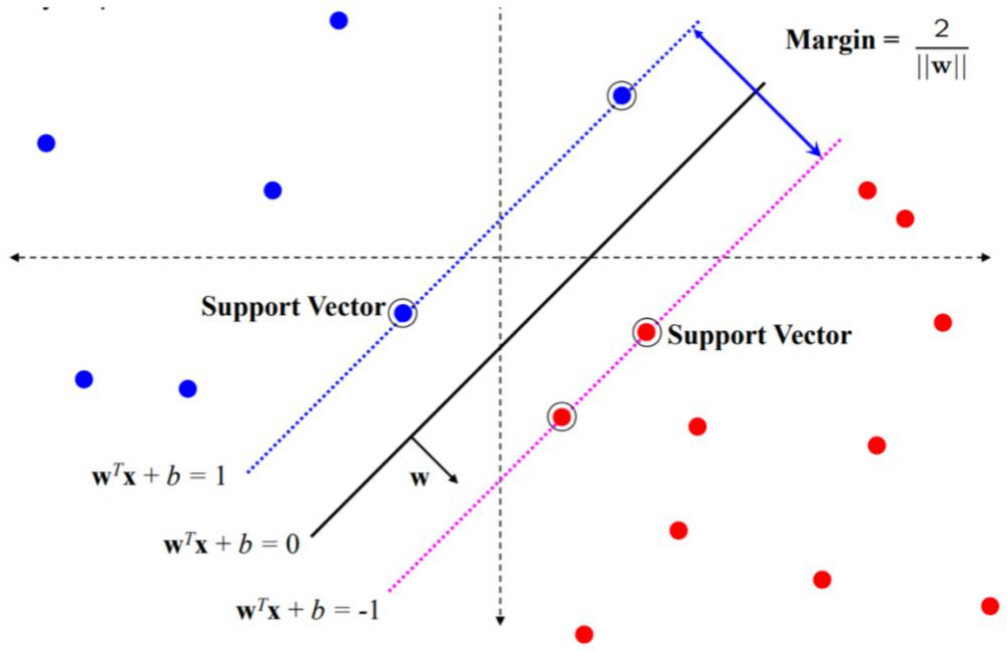

Abbildung 5: Zweidimensionale Illustration der SVM. Übernommen aus [49].

Mit dem Wissen, dass w senkrecht zur Hyperplane steht, kann der Abstand eines Punktes zur Hyperplane wie folgt definiert werden:

$$Distanz\ von\ Punkt\ x_i\ zu\ Hyperplane = \frac{|w^T\ x_i + b|}{\|w\|_2}$$

Formel 10: Distanz von Punkt zu Hyperplane. Übernommen aus [48].

Die Distanz kann ermittelt werden, indem der Punkt x_i auf den Normalenvektor w projiziert wird, also das Skalarprodukts $w^T x_i$ berechnet wird. Die Projektion wird dann durch die L2-Norm des Normalenvektors $\|w\|_2$ normiert. Daraus resultiert die tatsächliche orthogonale Distanz des Punktes x_i zur Hyperplane. Das Ziel ist es, diesen Abstand für alle Punkte zu maximieren. Der Abstand zu den nächstgelegenen Punkten der beiden Klassen wird als Margin und die Ränder des Margins, die parallel zur Hyperplane liegen, werden als Gutter bezeichnet (siehe in Abbildung 5, blaue beziehungsweise pink gestrichelte Linie). Auf den Guttern kommen die nächsten Punkte der beiden Klassen zu liegen, die sogenannten Support-Vektoren (siehe in Abbildung 5 «Support Vector»). Da der Margin theoretisch unendlich gross sein könnte, sind weitere Einschränkungen nötig, um die optimalen w-s und b-s zu berechnen. Für die beiden Gutter gilt neu:

$$w^T x + b = 1$$
$$w^T x + b = -1$$

Formel 11: Bedingungen für Gutter. Übernommen aus [48].

Das Standardisieren der Abstände der Support-Vektoren zur Hyperplane ist eine mathematische Konvention, die primär dazu dient, weitere Berechnungen zu vereinfachen. Die Standardisierung verlangt, dass w und b so skaliert werden, dass der Abstand genau 1 beträgt. Zudem garantiert es, dass die Abstände gleich gross auf beiden Seiten der Hyperplane sind. Für alle Punkte der Klasse, die oberhalb der Hyperplane liegen, also $y_i = 1$ (siehe in Abbildung 5 die Punkte der blauen Klasse) gilt folglich:

$$w^T x + b \geq 1$$

Formel 12: Klassifizierungsformel für oberhalb der Hyperplane liegende Punkte. Übernommen aus [48].

Für alle Punkte, die unterhalb der Hyperplane liegen, also $y_i = -1$, gilt umgekehrt:

$$w^T x + b\ \leq -1$$

Formel 13: Klassifizierungsformel für oberhalb der Hyperplane liegende Punkte. Übernommen aus [48].

Basierend auf den obigen Formeln und mit dem Ziel, eine einheitliche klassenunabhängige Gleichung für die Hyperplane zu erhalten, wird die Gleichung auf beiden Seiten mit y_i multipliziert. Es resultiert eine Gleichung für beide Klassen:

$$y_i(w^T x_i + b) \geq 1$$

Formel 14: Klassifizierungsformel für unterhalb der Hyperplane liegende Punkte. Übernommen aus [48].

Folglich gilt für die Support-Vektoren, die zur oberen Klasse beziehungsweise der unteren Klasse gehören:

$$\frac{|w^T x_i + b|}{\|w\|_2} = \frac{|\,1\,|}{\|w\|_2}$$

$$\frac{|w^T x_i + b|}{\|w\|_2} = \frac{|\,-1\,|}{\|w\|_2}$$

Formel 15: Distanz für Support-Vektoren. Übernommen aus [48].

Indem der minimale Abstand zur Hyperplane für jede Klasse nun auf mindestens 1 gesetzt wurde, lässt sich nun der Margin durch den Vektor w wie folgt beschreiben:

$$\frac{|\,-1\,|}{\|w\|_2} + \frac{1}{\|w\|_2} = \frac{2}{\|w\|_2}$$

Formel 16: Formel für Margin. Übernommen aus [48].

Diesen Abstand gilt es zu maximieren oder anders betrachtet, den Nenner $\|w\|_2$ zu minimieren:

$$min \frac{1}{2} \|w\|_2^2$$

Formel 17: Abstand im Nenner. Übernommen aus [48].

Wobei der Faktor $\frac{1}{2}$ und die Quadrierung für Ableitungszwecke eingeführt werden [47].
Das Optimierungsproblem für die Hyperplane lässt sich somit formulieren als:

$$\min \frac{1}{2} \|w\|_2^2 \;\; sodass \;\; y_i (w^T + b) \geq 1$$

Formel 18: Optimierungsproblem für SVM. Übernommen aus [48]

Der Bedingung im zweiten Teil der obigen Formel 18 dient dazu, die Punkte korrekt zu klassifizieren [50] [48].

Der bisher beschriebene Ansatz von SVM ist restriktiv, da keine Missklassifikationen innerhalb des Margins toleriert werden. In der Realität sind Daten oft nicht perfekt linear trennbar. Um die Flexibilität des Modells zu erhöhen, werden in einer Soft Margin SVM sogenannte Slackvariablen ξ eingeführt, die gewisse Missklassifikationen zulassen.

Das Optimierungsproblem für die Soft-Margin-SVM wird wie folgt definiert:

$$min_{w,b\,\xi} \;\; \frac{1}{2}\|w\|_2^2 \; + \; C \sum_{i=1}^{n} \xi_i$$

Formel 19: Optimierungsproblem für Soft-Margin-SVM. Übernommen aus [50] [48].

Hierbei stellt der zweite Term eine Strafsumme für die vorkommenden Missklassifikationen dar. Der Regulierungsparameter C kontrolliert die Gewichtung dieser Strafsumme im Verhältnis zur Maximierung des Margins und sorgt für das Gleichgewicht zwischen den beiden Termen [50] [48].

Es gelten zudem folgende Nebenbedingungen für das Soft-Margin-SVM-Optimierungsproblem:

$$y_i (w \cdot x_i - b) \geq 1 - \xi_i \;\; und \; \xi_i \geq 0$$

Formel 20: Soft-Margin-SVM-Optimierungsproblem und Nebenbedingung. Übernommen aus [50] [48].

Die Slackvariablen ξ dürfen nur positive Werte annehmen. Nimmt die Slackvariable den Wert 0 an, so ist der Punkt korrekt klassifiziert und liegt höchstens auf den Guttern, was ihn dann zu einem Support-Vektor macht. Weist die Slackvariable einen grösseren Wert als 1 auf, so handelt es sich um einen falsch klassifizierten Punkt, wie er in der Abbildung 6 als «Missclassified point» dargestellt wird [50] [49].

SVM verfügt zudem über den sogenannten Kernel-Trick, bei dem nicht linear trennbare Daten durch die Transformation in höhere Dimensionen separiert werden können. Da der Kernel-Trick in dieser Arbeit nicht verwendet wurde, wird auf dessen Beschreibung verzichtet [50] [48].

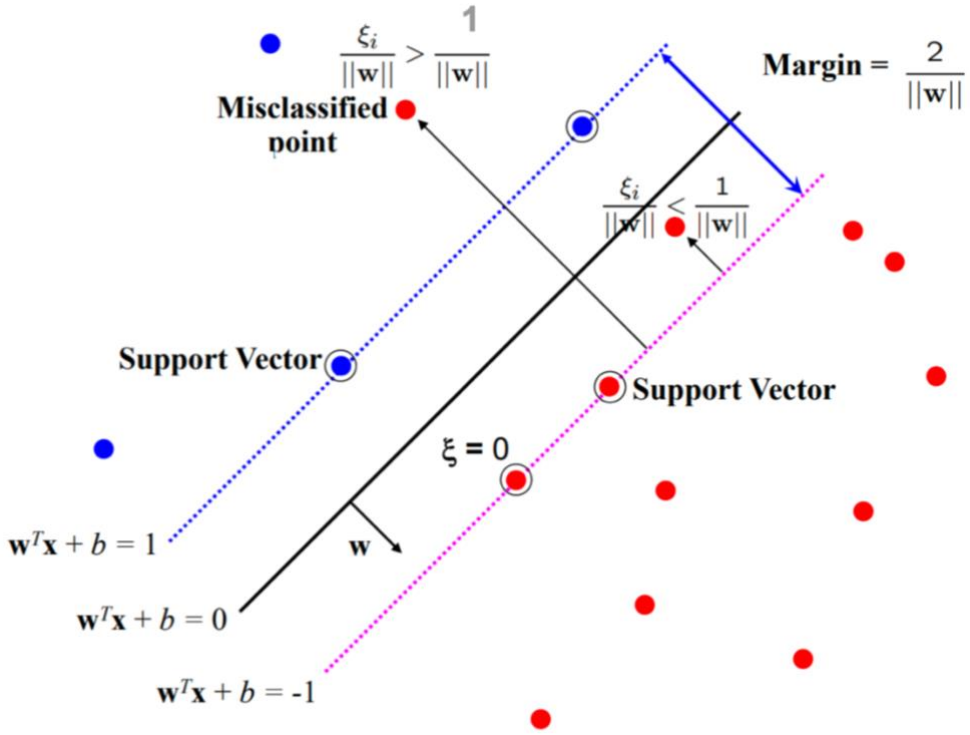

Abbildung 6: Zweidimensionale Illustration der Soft Margin SVM. Übernommen aus [49].

Für die Klassifikation mehrerer Klassen, wie sie in dieser Arbeit nötig ist, wird der Ansatz der oben beschriebenen binären SVM beziehungsweise Soft-Margin-SVM durch das One-vs-One-Verfahren erweitert, bei dem für jede mögliche Klassenkombination ein separates SVM-Modell trainiert wird. Die Klassifizierung eines Samples wird durch den Mehrheitsentscheid der Modelle bestimmt [51].

4.2.4 Random-Forest

Random-Forests, im Weiteren abgekürzt als RF, gehören – wie die drei zuvor beschriebenen Algorithmen – zu den gängigsten maschinellen Lernalgorithmen. Sie basieren auf einer Menge von sogenannten Entscheidungsbäumen. Die Funktion von RFs in Bezug auf die Klassifikation wird im Folgenden betrachtet [52].

Ein Entscheidungsbaum ist ein Modell, das eine hierarchische und baumartige Struktur besitzt [53]. Er besteht aus:

- einem Wurzelknoten, dem obersten Knoten, der für die gesamte Datenmenge steht (siehe in Abbildung 7 dunkelblauer Knoten)
- mehreren inneren Knoten, die Entscheidungen treffen und die Daten in zwei Teilmengen aufteilen (siehe in Abbildung 7 hellblaue Knoten)
- mehreren Blattknoten, die die endgültige Vorhersage beziehungsweise Klassifikation liefern (siehe in Abbildung 7 orangefarbene Knoten)

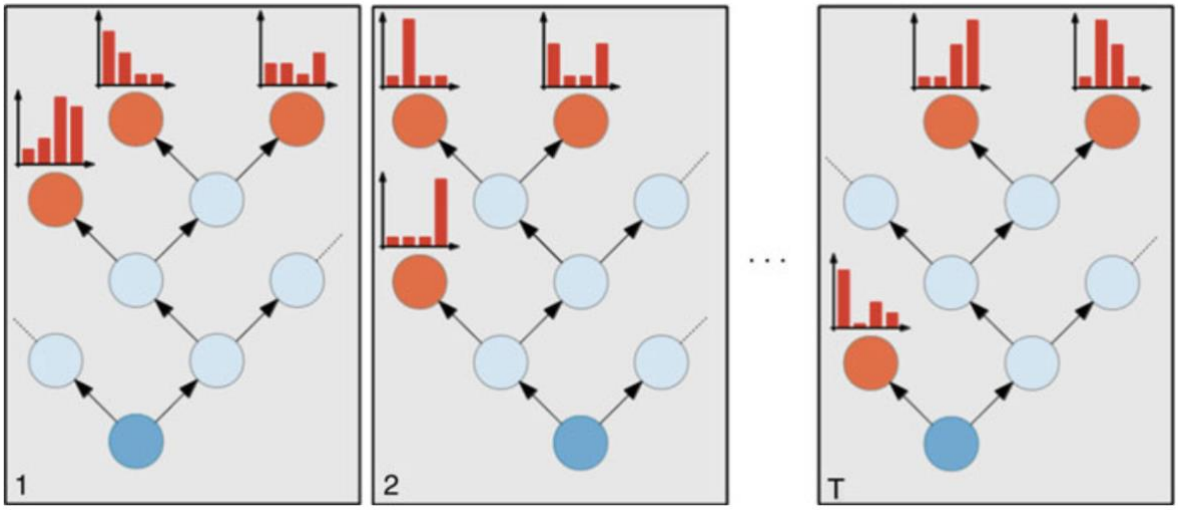

Abbildung 7: «RFs als Menge von binären Entscheidungsbäumen» [52]. Abbildung aus [52].

Die Aufteilung der Daten beginnt bei einem Entscheidungsbaum am Wurzelknoten (siehe in Abbildung 7 dunkelblauer Knoten) und erfolgt basierend auf einer sogenannten Schwellwertfunktion. Ein einfaches Beispiel einer Schwellwertfunktion ist:

$$x_i < \Theta$$

Formel 21: Beispiel für Schwellwertfunktion. Übernommen aus [52].

Diese Funktion vergleicht ein bestimmtes Merkmal x_i des Eingabevektors X mit einem Schwellwert Θ. Die Wahl des geeigneten Merkmals x_i und des Schwellwertes Θ erfolgt durch die Maximierung des Informationsgewinns. Der Informationsgewinn gibt an, wie geeignet ein Merkmal für die Datentrennung ist und wird wiederum durch ein Unreinheitsmass bestimmt. Ein häufig verwendetes Unreinheitsmass ist der Gini-Index, der auch in dieser Arbeit eingesetzt wurde:

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2$$

Formel 22: Formel zur Berechnung des Gini-Index. Übernommen aus [54]

Dabei ist p_i die relative Häufigkeit der Klasse i im Datensatz D und c steht für die Anzahl Klassen. Mit dem Gini-Index wird die Wahrscheinlichkeit dafür berechnet, dass ein zufällig ausgewähltes Element aus dem Datensatz falsch klassifiziert werden würde, wenn es gemäss der Verteilung der Klassen einer Klasse

zugeordnet wird. Mit anderen Worten: Der Gini-Index gibt die Wahrscheinlichkeit einer Fehlklassifizierung eines zufälligen Elements an [55].

Ein hoher Gini-Wert weist auf eine hohe Unreinheit im (Blatt-)Knoten hin. Das heisst, keine Klasse dominiert und die Wahrscheinlichkeit einer Fehlklassifizierung ist hoch. Ein niedriger Gini-Wert bedeutet hingegen, dass die Mehrheit der Elemente in dem betreffenden (Blatt-)Knoten einer Klasse angehören. Erwünscht ist ein niedriger Gini-Wert, da das Blatt dadurch homogener ist und die Wahrscheinlichkeit hoch ist, dass ein zufällig ausgewähltes Element dieser Klasse angehört.

Die Auswahl eines Merkmals m und einem Schwellwert Θ erfolgt basierend auf dem höchsten Informationsgewinn. Der Informationsgewinn gibt an, wie viel Unreinheit des Datensatzes durch die jeweilige Aufteilung reduziert wird. Als Erstes wird der Gini-Index des Datensatzes D im aktuellen Knoten berechnet. Danach werden die Gini-Indizes für die nach dem Split erzeugten Teilmengen D_1 und D_2 berechnet und gewichtet. Die Differenz zwischen den beiden steht für den Informationsgewinn, der wiederum ein Mass für die Güte der Aufteilung ist [52].

Die Formel für den Informationsgewinn basierend auf dem Gini-Index lautet:

$$I(m) = Gini(D) - (\frac{D_1}{|D|}\ Gini(D_1) + \frac{D_2}{|D|} Gini(D2))$$

Formel 23: Informationsgewinn basierend auf dem Gini-Index [54]

$|D|$ repräsentiert die Grösse des Datensatzes am aktuellen Knoten. Die Teilmengen D_1 und D_2, die durch das Merkmal m und den Schwellwert Θ erzeugt wurden, bilden die neuen Kindknoten des Baums [56].

Das Merkmal sowie der Schwellenwert mit dem höchsten Informationsgewinn werden ausgewählt, um den Datensatz in möglichst homogene Teilmengen aufzuteilen. Die resultierenden Teilmengen D_1 und D_2 werden dann als neue Knoten im Baum eingesetzt. Für jeden dieser neu eingesetzten Knoten wird der optimale Split erneut gesucht. Dieser Prozess wird fortlaufend wiederholt, wodurch der Baum kontinuierlich wächst [52].

Damit der Baum nicht übermässig wächst, wird der Prozess der Datenunterteilung für alle Knoten so lange durchgeführt, bis ein bestimmtes Abbruchkriterium erreicht wird:

- Maximale Tiefe des Baums: Der Baum erreicht eine vordefinierte maximale Anzahl von Ebenen.
- Minimale Anzahl von Elementen in einem Knoten: Ein Knoten unterschreitet diese Anzahl.
- Homogenität im Knoten: Alle Datenpunkte in einem Knoten gehören zu derselben Klasse. Es besteht keine Notwendigkeit mehr, diesen Knoten zu splitten beziehungsweise er ist der optimale Blattknoten.

Ein Nachteil von Bäumen ist, dass sie häufig zu Overfitting neigen, insbesondere wenn die Daten komplex sind oder aus vielen Merkmalen bestehen. Dies führt dazu, dass sie auf neuen Daten schlecht generalisieren [57].

Mithilfe des Ensemble-Lernens von RF wird dieses Risiko minimiert. Dabei werden mehrere schwache Modelle – in diesem Fall Entscheidungsbäume – zu einem robusteren Modell kombiniert [52].

Für die Klassifizierung eines Samples X wandert der Datenpunkt durch alle Entscheidungsbäume des RFs. Jedes Sample endet dabei genau in einem einzigen Blatt pro Baum. Das Sample wird dieser Klasse mit der höchsten Häufigkeit im Blatt zugeteilt. Die endgültige Klassifikation basiert auf dem Mehrheitsentscheid aller Bäume. Dadurch wird die Generalisierungsfähigkeit des Modells verbessert und das Risiko für Overfitting verringert [58] [59].

Um die Korrelation zwischen den Bäumen zu reduzieren, werden zwei Methoden zur Randomisierung verwendet: Bagging und die zufällige Auswahl von Merkmalen [57] [59].

Beim Bagging wird jeder Entscheidungsbaum mit einer Teilmenge aus dem Trainingsdatensatz trainiert. Diese Teilmenge wird durch zufälliges Ziehen mit Zurücklegen erstellt. Dadurch können einzelne Datenpunkte mehrmals in einem Set erscheinen, während andere Datenpunkte möglicherweise nie in einem Set enthalten sind [59].

Bei jedem Split wird eine zufällige Teilmenge der Merkmale ausgewählt. Der beste Split wird dann basierend auf dieser Teilmenge konstruiert. Der Split mit der grössten Reduktion der Unreinheit beziehungsweise dem höchsten Informationsgewinn gemessen am Gini-Index wird ausgewählt [52] [57] [59].

4.2.5 XGBoost

Wie RF basiert XGBoost, im Weiteren als XGB abgekürzt, auf der Methode des Ensemble-Lernens. Dafür kommen binäre Entscheidungsbäume zum Einsatz. Während RF seine Bäume auf dem Prinzip des Baggings und der zufälligen Auswahl an Merkmalen konstruiert, basiert XGB auf dem Gradient-Boosting-Prinzip. Dabei wird eine Reihe von binären Entscheidungsbäumen sequenziell aufgebaut. Jeder neu aufgebaute Baum zielt darauf ab, die Fehler des vorherigen Baums zu korrigieren.

Gradient-Boosting beschreibt ein iteratives Verfahren zur Erstellung eines starken Modells durch die Kombination vieler schwacher Modelle. Begonnen wird mit einem einfachen Modell, wie beispielsweise einem Entscheidungsbaum, zu dem schrittweise neue Bäume hinzugefügt werden. Die neuen Bäume zielen darauf ab, die Residuen des aktuellen Modells zu korrigieren [60] [61].

Der iterative Gradient-Boosting-Algorithmus kann wie folgt zusammengefasst werden:

1. Es wird ein einfaches Modell $f_1\ (X, \theta_1)$ initialisiert (siehe Abbildung 8), das eine anfängliche Vorhersage für alle Daten liefert. Das kann beispielweise für die Klassifikation, die am häufigsten vorkommende Klasse sein.

2. Für jeden Datenpunkt im Trainingsdatensatz werden die Residuen berechnet. Residuen stehen für die Differenz zwischen den tatsächlichen Werten y und den vorhergesagten Werten $\hat{y}$. Diese Residuen entsprechen dem negativen Gradienten der jeweils verwendeten Loss-Funktion. Hervorzuheben ist an dieser Stelle, dass hier die Rede von negativen Gradienten ist, da diese dazu dienen, die Differenz zwischen den tatsächlichen und vorhergesagten Werten zu korrigieren.

3. Neue Entscheidungsbäume werden trainiert, um die zuvor berechneten negativen Gradienten zu approximieren.

4. Die neuen Entscheidungsbäume werden mit einer Lernrate gewichtet und dem anfänglichen Modell hinzugefügt, mit dem Ziel, die Vorhersage zu verbessern.
5. Die Schritte 2 bis 4 werden wiederholt, bis eine Konvergenzbedingung erreicht ist. Das kann beispielsweise eine festgelegte Anzahl von Iterationen oder das Unterschreiten eines Schwellenwertes sein [60] [61].

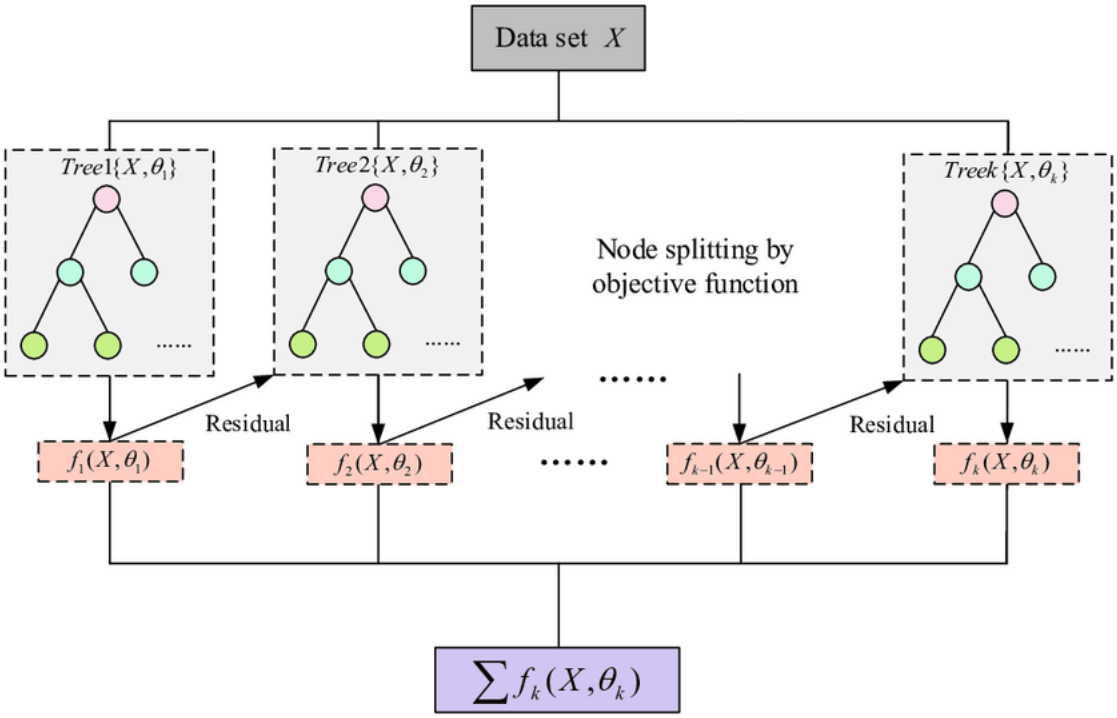

Abbildung 8: Illustration für XGB. Übernommen aus [62]

Das endgültige Modell $F(X)$ ist die gewichtete Summe aller Bäume, wobei θ_k den Gewichten des jeweiligen Baumes entspricht:

$$F(X) = \sum_{m=k}^{K} \alpha_k \, f_k(X, \, \theta_k)$$

Formel 24: Vollständiges Gradient-Boosting-Modell. Angepasst aus [61].

XGB erweitert das Prinzip des Gradient-Boostings durch Optimierungen und Regularisierungen. Dadurch gehört er zu den leistungsstärksten Algorithmen. Eine solche Optimierung ist die Erweiterung der Gradientenberechnung mit Hesse-Matrizen, die die zweite Ableitung der Loss-Funktion berechnen. Indem

die Hesse-Matrizen in die Gradientenberechnung einbezogen werden, kann das Modell genauer angepasst werden.

Für Klassifikationsaufgaben werden als Erstes die Ausgaben der Bäume – Logits – mit der Softmax-Funktion in Wahrscheinlichkeiten transformiert, wie bereits in Abschnitt 4.2.2 zu Multinomial-Logistic-Regression beschrieben. Für den Loss wird der ebenfalls in Abschnitt 4.2.2 behandelte Cross-Entropy-Loss verwendet, um die Differenz zwischen den tatsächlichen Klassenlabels und den vorhergesagten Wahrscheinlichkeiten zu berechnen. Der Gradient der Loss-Funktion g_i berechnet die erste Ableitung bezüglich der Vorhersagen:

$$g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} = \hat{p}_i - y_i$$

Formel 25: Erste Ableitung der Cross-Entropy-Loss-Funktion. Angepasst aus [63].

$\hat{p}_i$ steht dabei für die vorhergesagte Wahrscheinlichkeit der Klasse i, die durch die zuvor angewendete Softmax-Funktion berechnet wurde [64] [63].

Die zweite Ableitung der Loss-Funktion, die sogenannte Hesse-Matrix, wird ebenfalls für jedes Sample berechnet:

$$h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2} = \hat{p}_i(1 - \hat{p}_i)$$

Formel 26: Zweite Ableitung der Loss-Funktion. Angepasst aus [63].

Die zweite Ableitung bezieht die Krümmung der Funktion in die Loss-Funktion ein, was die Komplexität reduziert [64] [63].

Basierend auf den berechneten Gradienten und Hesse-Matrizen, werden neue Entscheidungsbäume trainiert, um die berechneten Gradienten der Loss-Funktion zu modellieren. Mit den neuen Bäumen werden die Fehler des aktuellen Modells direkt angegangen.

Im Vergleich zu RF werden für das Splitten statt des Gini-Index wie in Abschnitt 4.2.4 die Gradienten und Hesse-Matrizen genutzt. Die Entscheidung für einen Split basiert auf dem sogenannten «Gain». Dieser berechnet den Unterschied in der Loss-Funktion vor und nach dem Split. Dies führt zu einer weiteren Optimierung, da in die Entwicklung der Bäume direkt die Informationen aus den Gradienten- und Hesse-Werten miteinfliessen.

Zusätzlich werden für die Blätter, die die endgültige Vorhersage beziehungsweise Klassifikation durchführen, Gewichte basierend auf den Gradienten und Hesse-Matrizen berechnet:

$$w_j = -\frac{\sum_{i \in j} g_i}{\sum_{i \in j} h_i + \lambda}$$

Formel 27: Gewichte für Blätter basierend auf 1. und 2. Ableitungswerten. Übernommen aus [61].

Der Regulierungsterm $\lambda > 0$ bestraft grosse Gewichte und reduziert das Risiko für Overfitting, während j für den Index des jeweiligen Blattes steht.

Die Summe der Gradienten im Zähler steht für die Abweichung zwischen den aktuellen Vorhersagen in diesem Blatt und den tatsächlichen Werten. Die Krümmung der Loss-Funktion wird im Nenner durch die Summe der Hesse-Werte dargestellt. Sie dient dazu, die Stärke der Anpassung zu regulieren. Denn ein hoher Wert für die Krümmung deutet darauf hin, dass kleine Änderungen in der Vorhersage eine signifikante Änderung in der Loss-Funktion bewirken können.

Nachdem die Bäume konstruiert und die Blattgewichte entsprechend angepasst wurden, wird das Modell aktualisiert, indem neue Vorhersagen zu den bisherigen addiert werden (siehe Abbildung 8). Dies geschieht so lange, bis eine Abbruchbedingung erreicht wird, wie im Prozess des Gradient-Boostings eingangs beschrieben.

Der iterative Prozess zur Minimierung der Loss-Funktion bei gleichzeitiger Reduzierung der Modellkomplexität wird in der Zielfunktion von XGB wie folgt zusammengefasst:

$$Obj(\Theta) = \sum_{i=1}^{n} L\,(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega\,(f_k)$$

Formel 28: Zielfunktion für XGB. Übernommen aus [65]

Die Zielfunktion von XGB ist eine Kombination aus einer Loss-Funktion L und einer Regularisierungsfunktion Ω, wobei n für die Anzahl der Trainingsdaten steht, K die Anzahl der Bäume und y_i der tatsächliche Wert, $\hat{y}_i$ der vorhergesagte Wert und f_k die Funktion des k-ten Baums ist [61] [66] [67].

XGBs Stärke liegt genau in dieser Zielfunktion, bei der sowohl die Vorhersagegenauigkeit optimiert wird als auch mit dem Regulierungsterm auf die Komplexität der einzelnen Bäume eingegangen wird [61] [66].

4.3 Repräsentationen

Für das maschinelle Lernen ist es essenziell, in welcher Form die Daten vorliegen und wie sie repräsentiert werden. In dieser Arbeit wurden die von den Phoneme-Recognizern generierten Phonemtranskriptionen den Classifiern als n-Gramme bereitgestellt. Wie in den in Kapitel 2 vorgestellten Forschungsarbeiten zur Dialekterkennung, wurden n-Gramme als Merkmalsrepräsentationen eingesetzt. Hier sei hervorgehoben, dass die n-Gramme auf Zeichenbasis und nicht auf Wortbasis erstellt werden. Jedes n-Gramm repräsentiert eine zusammenhängende Folge von n Phonemen aus einer gegebenen Phonemtranskription. Solche

zeichenbasierten n-Gramme werden neben der Dialekterkennung für viele weitere Aufgaben des Natural Language Processing eingesetzt [68]. Sie weisen eine grössere Granularität auf und bieten das Potenzial, charakteristische Muster für einzelne Dialekte auf Phonembasis zu erfassen [69]. Es wird die Häufigkeit der vorkommenden n-Gramme in einem Trainingssample gezählt, wobei das Sample sowohl aus einem Satz als auch aus mehreren Sätzen bestehen kann. Dem jeweiligen Classifier wird demnach für ein Trainingssample ein Vektor übergeben, dessen Länge der Anzahl vorkommender n-Gramme im Trainingsset entspricht. Die Häufigkeit der effektiv auftretenden n-Gramme in diesem Sample wird erfasst, während allen übrigen n-Grammen im Vektor der Wert 0 zugewiesen wird. Infolgedessen resultiert zum Beispiel ein kurzes Sample in einem spärlicher besetzten Merkmalsvektor [69].

4.4 Normalisierung

Um die Vergleichbarkeit zwischen den verschiedenen Transkriptionen sicherzustellen, bedarf es der Normalisierung der Häufigkeiten im zuvor beschriebenen Merkmalsvektor.

Dazu wurde die L2-Norm, auch als euklidische Norm bezeichnet, verwendet. Diese ist eine gängige Methode zur Normalisierung. Die L2-Norm wird definiert als:

$$\|X\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2}$$

Formel 29: Formel für L2-Norm. Übernommen aus [70].

Wobei n für die Anzahl Komponenten im Vektor steht und x_k die k-te Komponente ist.

Jede Vektorkomponente des Merkmalvektors wird durch die L2-Norm geteilt. Die Häufigkeiten in den Vektoren werden dadurch skaliert, sodass alle Vektoren eine einheitliche Länge von 1 erhalten. Damit wird der Einfluss der absoluten Häufigkeiten der n-Gramme reduziert, da längere Transkriptionen dazu tendieren, grössere absolute Werte für häufig auftretende Features zu erhalten. Dies führt zu einer besseren Vergleichbarkeit zwischen den verschieden langen Transkriptionen [71] [42].

4.5 Tf-idf

Mit der Tf-idf-Gewichtung können die Merkmalsvektoren gewichtet werden. Dabei wird die Relevanz der einzelnen Merkmale, also der n-Gramme in einer Transkription gemessen. TF steht für Term-Frequency und bezeichnet, wie oft ein Term in einem Dokument vorkommt [72]. Dabei wird die relative Häufigkeit des Auftretens des Terms, in diesem Fall n-Gramm, im Verhältnis zur Länge der Transkription berechnet, wie in der folgenden Formel dargestellt:

$$tf\ (t,d) = \frac{Anzahl\ Vorkommnisse\ des\ Terms\ t\ in\ Dokument\ d}{Gesamtanzahl\ aller\ Terme\ im\ Dokument\ d}$$

Formel 30: Tf-Formel. Angepasst aus [73].

IDF, abgekürzt für Inverse-Document-Frequency, ist ein Mass für die Ausprägung (Wichtigkeit) eines Terms für ein bestimmtes Dokument [72].

$$idf(t) = \log(\frac{Anzahl\ Dokumente}{1 + Anzahl\ Dokumente, die\ Term\ t\ enthalten})$$

Formel 31: Idf-Formel. Angepasst aus [73].

Die 1 im Nenner vermeidet eine Division durch null, falls der Term in keinem Dokument vorkommt.

Die Kombination von tf und idf ergibt den Tf-idf-Wert, der durch folgende Formel beschrieben werden kann:

$$tf\text{-}idf\ =\ tf(t,d) * idf(t)$$

Formel 32: Tf-idf Formel. Übernommen aus [73].

Seltene Terme, in diesem Fall n-Gramme, die häufig in einem bestimmten Dokument (Transkription) vorkommen, aber selten in allen anderen Transkriptionen, erhalten mit der Tf-idf-Formel eine stärkere Gewichtung. In diesem Zusammenhang wird davon ausgegangen, dass häufig vorkommende n-Gramme, die in allen übrigen Transkriptionen weniger häufig vorkommen, wichtig für die Identifizierung der jeweiligen Transkription sind [72] [74] [75].

Inwiefern die Tf-idf-Gewichtung dazu beitragen kann, Dialekte zu identifizieren, wird sich später in den Experimenten im Kapitel 7 zeigen.

4.6 Metriken

Für die Bewertung der Leistung der eingesetzten Classifier wurden die folgenden beiden Metriken verwendet.

4.6.1 Accuracy

Mit der Accuracy wird der Anteil der korrekt klassifizierten Vorhersagen gemessen:

$$accuracy(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} 1(\hat{y}_i = y_i)$$

Formel 33: Formel für Accuracy. Übernommen aus [76].

Wobei n für die Anzahl Samples steht und $1(\hat{y}_i = y_i)$ eine sogenannte Indikatorfunktion ist, die 1 zurückgibt, wenn $\hat{y}_i$ mit y_i übereinstimmt und sonst 0 [76].

Eine hohe Accuracy zeigt, dass das Modell die Merkmale der verschiedenen Dialekte korrekt klassifizieren und folglich gut unterscheiden kann [76]. Dies bedingt jedoch eine Gleichverteilung der Klassen, da die Accuracy die Gesamtzahl der korrekten Vorhersagen misst, ohne auf die Verteilung der Klassen Rücksicht zu nehmen [77].

Die Accuracy wurde ausschliesslich für das Hyperparameter-Tuning der Modelle verwendet.

4.6.2 F1-Macro

Für den F1-Macro-Score wird der F1-Score für jede Klasse berechnet und dann der Durchschnitt dieser Scores gebildet:

$$F1\text{-}Score = 2 * \frac{Precision + Recall}{Precision * Recall}$$

Formel 34: Formel für F1-Score. Übernommen aus [76].

$$F1\text{-}Macro\text{-}Score = \frac{1}{K} \sum_{k=1}^{K} F1\text{-}Score_k$$

Formel 35: Formel für F1-Macro-Score.

Der F1-Score bildet das sogenannte harmonische Mittel aus Precision und Recall. Die Precision beschreibt die Genauigkeit der Vorhersagen für eine bestimmte Klasse und der Recall gibt den Anteil der korrekt identifizierten Instanzen einer bestimmten Klasse an [76] [77].

Der F1-Macro-Score ist insbesondere für Klassifikationsaufgaben geeignet, bei der alle Klassen unabhängig von ihrer Häufigkeit gleich gut erkannt werden sollen. Im Gegensatz zur Accuracy ist der F1-Macro-Score insbesondere bei ungleichmässig verteilten Klassen robuster [77]. Aus diesem Grund wurde der F1-Macro-Score für die abschliessende Leistungsbewertung der Modelle auf dem Testingset eingesetzt.

4.7 Validierung (Cross-Validation) und Testing

Um eine Leistungsschätzung für das Hyperparametertuning des jeweiligen Classifiers zu ermitteln, wurde die Cross-Validation eingesetzt. Die Cross-Validation teilt den Datensatz, mit Ausnahme des Testingset, in gleich grosse Teile, sogenannte Folds. Das Modell wird wiederholt für die Anzahl Folds trainiert. Dabei dient jeweils ein Fold als Validierungsset, während die anderen Folds zusammen als Trainingsset dienen. Zum Schluss wird die durchschnittliche Leistung über alle Folds hinweg berechnet. Dieses Verfahren wird als K-Fold-Cross-Validation bezeichnet, wobei K für die Anzahl Folds steht [78] [79] [80].

Mit dem Einsatz der Cross-Validation wird die Varianz der Leistungsschätzung reduziert, da auf verschiedenen Teilmengen der Daten trainiert und validiert wird. Je nach Ergebnis der Cross-Validation wird das Modell angepasst, indem die Hyperparameter optimiert werden. Ganz zum Schluss wird anhand eines unangetasteten und ungesehenen Testingsets, das zu Beginn beiseitegelegt wurde, die Leistung des Modells gemessen [78] [79].

In dieser Arbeit wurde eine Variante der K-Fold-Cross-Validation eingesetzt. Diese stellt sicher, dass die Klassenverteilung in den Folds der Klassenverteilung im gesamten Datensatz entspricht. Diese Methode wird als Stratified-K-Fold-Cross-Validation bezeichnet [78] [79].

An dieser Stelle sei zu erwähnen, dass die gleiche Herausforderung bei der Generierung der Folds wie bei der Generierung der Splits (siehe Abschnitt 6.2) bestand. Die Transkriptionen sind an die Sprecher:innen geknüpft und für den Korpus SDS-200 weisen die Sprecher:innen eine ungleichmässige Anzahl der Samples auf. Die gleichzeitige Berücksichtigung der Klassenverteilung in den Folds, die Erstellung gleich grosser Folds und die Gruppierung nach Sprecher:innen ist enorm herausfordernd, wenn nicht unmöglich.

Grouped-Stratified-K-Fold-Cross-Validation ist eine Variante der Cross-Validation, die die Samples nach Sprecher:innen gruppiert, sodass keine Überschneidung der Sprecher:innen im Trainings- und Validierungsset enthalten sind. Diese wurde implementiert, erbrachte jedoch keine guten Ergebnisse und führte zu unausgeglichenen Folds. Daher wurde trotz der Überschneidung der Sprecher:innen in den Folds weiterhin die Stratified-K-Fold-Cross-Validation verwendet [80] [78] [79].

Aufgrund der teils hohen Laufzeit des Classifiers SVM wurde klassisch ein separates Validierungsset verwendet und nicht mit Cross-Validation vorgegangen.

5. Code und Infrastruktur

In diesem Kapitel wird genauer beschrieben, wie die Codebasis für die Experimente implementiert wurde und welche Infrastruktur genutzt wurde, um die Experimente durchzuführen und zu dokumentieren.

5.1 Programmiersprache und Libraries

Wie bereits in der vorangehenden Projektarbeit wird Python als Programmiersprache verwendet [4] [81]. Die Sprache ist äusserst vielseitig einsetzbar und kann sowohl für einzelne Skripte prozedural als auch für grössere Konstrukte objektorientiert eingesetzt werden. Gerade für Machine-Learning eignet sich Python besonders gut, da viele bewährte Libraries für die Datenverarbeitung und für die Umsetzung von Machine-Learning-Modellen zur Verfügung stehen. Nicht zu unterschätzen ist zudem, dass sie die bevorzugte Wahl der meisten Machine-Learning-Ingenieuren ist und entsprechend viele hilfreiche Ressourcen vorhanden sind.

Für die Verarbeitung der Metadaten und Transkriptionen wurden in dieser Arbeit hauptsächlich Pandas und Numpy verwendet [82] [83]. Mit Pandas können tabellarische Daten schnell angepasst, durchsucht und neu zusammengestellt werden, während Numpy den Umgang mit mehrdimensionalen Arrays vereinfacht. Die Classifier wurden mit der beliebten Machine-Learning-Library scikit-learn implementiert [84]. scikit-learn bietet alle Klassifizierungsalgorithmen, die für die Arbeit benötigt werden, Klassen und Funktionen, um Features aus den Transkriptionen zu extrahieren und zu gewichten, sowie Metriken, um die Model-Performance zu evaluieren. Für die Analyse und Darstellung der Ergebnisse kam neben Pandas matplotlib zum Einsatz, das mit Pandas kompatibel ist [85]. Es bietet alle gängigen statistischen Plots, die je nach Bedürfnis angepasst werden können.

5.2 Codearchitektur

Die Codearchitektur setzt sich aus drei autonomen Teilen zusammen. Ein Teil ist das Preprocessing der Audiodaten in einem ersten Schritt und der generierten Transkriptionen in einem zweiten Schritt.[6] Für jeden abgrenzbaren Preprocessingschritt wurde ein einzelnes Skript implementiert, das separat lauffähig ist, solange die entsprechenden Daten in der benötigten Form vorhanden sind. Dies sind meist Datenformen, die bei vorangehenden Preprocessingschritten erstellt wurden. Der zweite Teil umfasst die Phonemtranskription mithilfe verschiedener Phoneme-Recognizer, wie sie in Abschnitt 4.1 zur Model-Architektur beschrieben wird. Abbildung 9 zeigt das entsprechende Klassendiagramm. Jeder Phoneme-Recognizer wurde in einer eigenen Klasse implementiert. Die Struktur für diese Klassen wird im Interface «phoneme_recognizer.py» vorgegeben, sodass jeder Phoneme-Recognizer gleich integriert wird und austauschbar genutzt werden kann. Die Transkription kann für alle Recognizer über das gleiche Main-Skript «transcribe.py» gestartet werden.

[6] Auf die Schritte des Preprocessing wird in Abschnitt 6.1 genauer eingegangen.

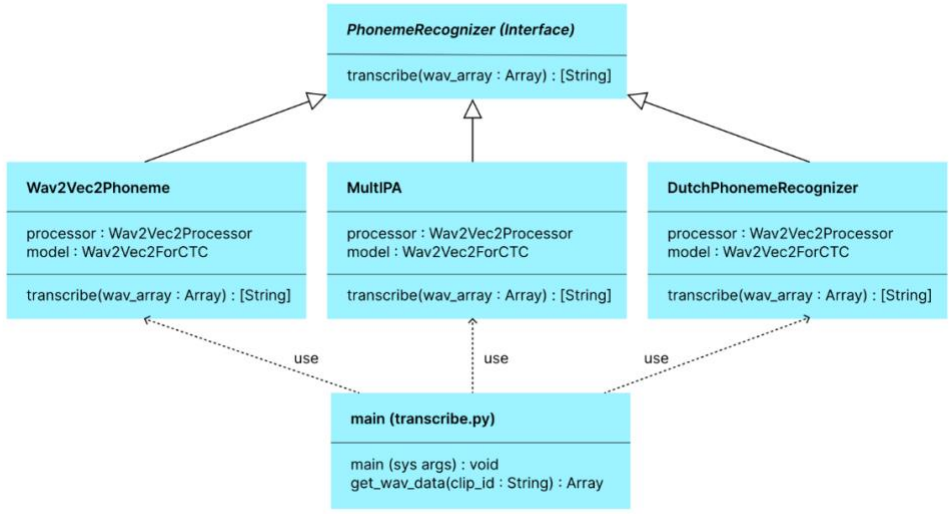

Abbildung 9: Klassendiagramm für die Phonemtranskription

Der dritte Teil enthält das Training und Tuning der Classifier. Das dazugehörige Klassendiagramm ist in Abbildung 10 zu sehen. Wie bei den Phoneme-Recognizern wurde jeder Classifier in einer eigenen Klasse implementiert. Gemeinsame Elemente der Classifier wurden in der Parent-Class «phoneme_classifier.py» implementiert, die gleichzeitig die Struktur für die Child-Classes vorgibt. Die main-Skripte, um das Training oder Tuning zu starten, sind das «train_phoneme_classifer.py» beziehungsweise das «tune_phoneme_classifer.py».

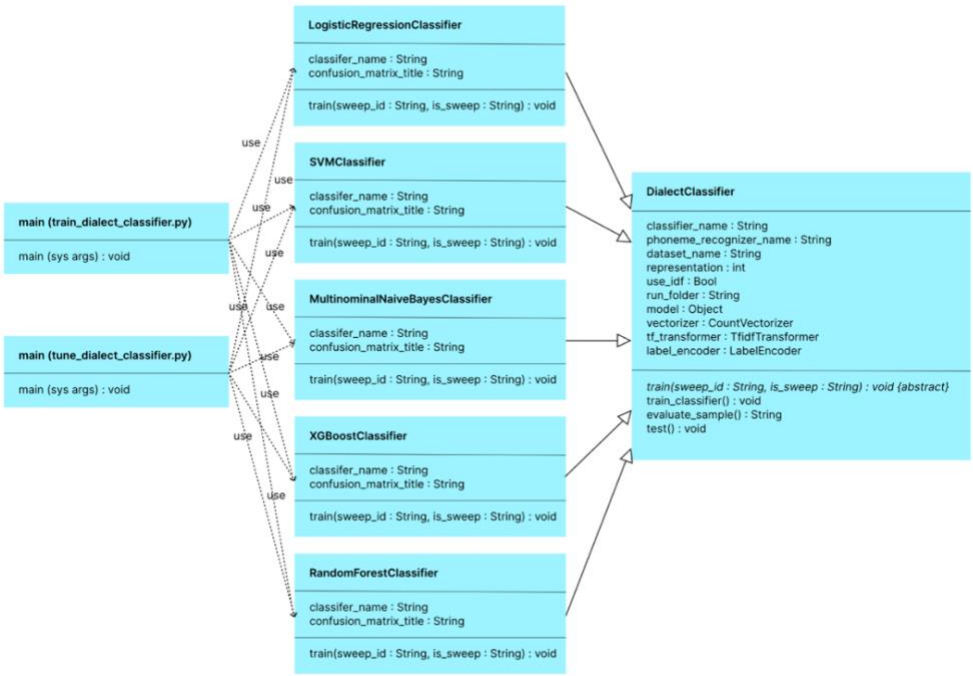

Abbildung 10: Klassendiagramm für das Training und Tuning

5.3 Logging und Hyperparameter-Tuning mit Weights & Biases

Weights & Biases ist eine vielseitige Online-Plattform für AI-Entwickler [86]. In erster Linie können der Verlauf und die Ergebnisse von Experimenten aufgezeichnet und visualisiert werden. Ausserdem werden Funktionalitäten bereitgestellt, um Hyperparameter einfacher zu tunen oder Datensets, trainierte Modelle und andere Artefakte zu speichern und zu versionieren. Weil für diese Arbeit Hunderte von Experimenten auf verschiedenen Maschinen durchgeführt werden sollten und ein ausführliches Hyperparameter-Tuning für jeden Classifier vorgenommen werden sollte, bot sich Weights & Biases an. Mithilfe der zur Plattform gehörigen Python-Library werden direkt aus dem Code für jeden Trainingsrun die Hyperparameter, Metriken und Confusion-Matrix geloggt. Das Hyperparameter-Tuning wird mithilfe derselben Library direkt von Weights & Biases gesteuert. Im Code wird angegeben, welche Hyperparameter ausprobiert werden sollen, welche Metrik maximiert werden soll und welche Hyperparameter-Tuning-Technik (zum Beispiel Grid-Search, Random-Search oder Bayesian-Optimization) angewendet werden soll. Weights & Biases startet die nötigen Trainingsdurchläufe automatisch und loggt die einzelnen Durchläufe sowie den Verlauf des Tunings auf der Plattform.

5.4 Genutzte Hardware und Virtual Machines

Insgesamt wurden für die Umsetzung drei verschiedene Cloudplattformen und die persönlichen Computer eingesetzt. Für das Preprocessing der Audiodaten wurde der Openstack Cluster APU genutzt, der von der ZHAW kostenlos bereitgestellt wurde. In einer persönlichen Virtual-Machine (VM) standen 650 GB Speicher, eine CPU mit 16 GB RAM und eine 92-nVidia-Tesla-T4-Grafikkarte zur Verfügung. Das Preprocessing profitiert jedoch nicht von der GPU, da es sich um simple Datenverarbeitung handelt, und wurde deshalb auf der CPU ausgeführt. Erst für das Transkribieren wurde die GPU verwendet, da die in Abschnitt 4.1 erklärten Phoneme-Recognizer auf neuronalen Netzwerken basieren. Für das Tuning und Training der Classifier war keine speziell leistungsstarke Hardware nötig. Ein Experiment mit traditionellen Klassifizierungsalgorithmen dauert nur wenige Minuten bis selten wenige Stunden und kann auch auf einem lokalen Computer durchgeführt werden. Allerdings sollten, wie bereits erwähnt, mehrere Hundert Experimente durchgeführt werden. Daher wurden zwei weitere Plattformen hinzugezogen: ein virtueller Server mit einer 16-GB-RAM-CPU und 96 GB Speicher und eine Compute-Engine auf der Google Cloudplattform mit einer 32-GB-RAM-CPU.

Einzig die APU-VM musste für das Preprocessing eingerichtet werden. Dafür wurden die 150 GB originalen Daten der beiden vorgestellten Korpora in die VM geladen. Auf den restlichen beiden VMs mussten nur die in der APU-VM erstellten Transkriptionen hochgeladen werden. Die Libraries wurden nicht direkt in den VMs installiert. Stattdessen wurde Docker installiert, um die Skripte isoliert von der VM in einem Container auszuführen, der alle nötigen Abhängigkeiten enthält [87]. Es wurde ein Docker-Image mit allen nötigen Abhängigkeiten basierend auf einem Dockerfile erstellt und sämtliche Prozesse in einem neuen Docker-Container abgewickelt. Dies hat den Vorteil, dass Libraries nicht auf jeder VM neu installiert werden müssen und nicht jedes Mal von Neuem auf Versionskonflikte geachtet werden muss.

6. Datenaufbereitung für die Experimente

Wie in Kapitel 3 erwähnt, werden die beiden vorgestellten Korpora STT4SG-350 und SDS-200 als Datengrundlage für die Experimente verwendet. In der vorangehenden Projektarbeit hat ein Mix der Korpora zu besseren Ergebnissen geführt als der neuere STT4SG-350 allein [4]. Es wird davon ausgegangen, dass dies auch für den neuen Ansatz gilt, zumal dadurch eine grössere Diversität entsteht. Zudem können die neuen Resultate besser mit denen der vorangehenden Arbeit verglichen werden, wenn die gleichen Daten verwendet werden. In den folgenden Abschnitten wird beschrieben, wie die Audiodaten aus den Korpora aufbereitet werden, um damit Classifier-Modelle trainieren zu können.

6.1 Preprocessing der Audio- und Metadaten

Das Preprocessing konnte aus der vorangehenden Projektarbeit übernommen werden, wo es sich wiederum an der Bachelorarbeit von Frei und Schneider orientiert hat [20] [4]. Abbildung 11 stellt die einzelnen Schritte dar. Für beide Korpora liegen sowohl Audiodateien als auch dazugehörige Metadaten in tabellarischer Form vor. Die originalen Audiodaten des STT4SG-350-Korpus sind im FLAC-Format. Diese mussten zuerst neu codiert werden, weil Informationen im Dateiheader fehlten, die benötigt werden, um die Dateien in gängigen Mediaplayern abzuspielen. Beim SDS-200-Korpus, der MP3-Dateien enthält, ist dies nicht nötig. In einem weiteren Schritt wurden für beide Korpora die rohen Wellenformen als Arrays aus den Audiodaten extrahiert und unter den in den Metadaten definierten IDs in einer HDF5-Datei gespeichert.[7] Die HDF5-Datei ist vergleichbar mit einer Art Minidatenbank in Dateiform, in der grosse Datenmengen effizient und strukturiert gespeichert werden können und auf die einfach zugegriffen werden kann [88]. Um auf ein bestimmtes Sample zuzugreifen, wird einzig die ID benötigt, unter der es gespeichert wurde.

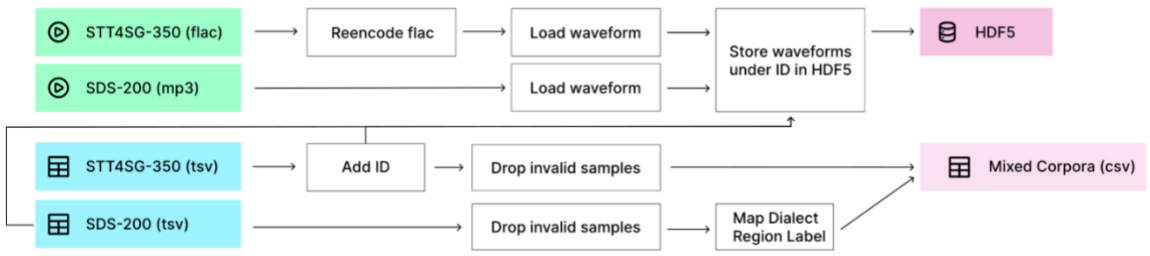

Abbildung 11: Schritte des Preprocessings der Audio- und Metadaten

Parallel zu den Audiodaten wurden die Metadaten der Korpora aufbereitet. Die Metadaten wurden im TSV-Format zusammen mit den Audiodateien mitgeliefert. Für den SDS-200 stand eine einzige TSV-Datei für alle Samples zur Verfügung. Mit dem STT4SG-350 wurden vier TSV-Dateien bereitgestellt: je eine Datei für die vordefinierten Testing- und Validierungssets, eine für ein über die Dialektregionen hinweg

[7] Wie später im Abschnitt noch aufgegriffen wird, musste für den STT4SG-350-Korpus zuerst noch eine ID erzeugt werden.

unbalanciertes Trainingsset und eine für ein ausbalanciertes Trainingsset. Dabei ist Letzteres eine Teilmenge des unbalancierten Trainingssets. In dieser Arbeit wurde nur das balancierte Trainingsdatenset verwendet.

Als Erstes wurde im Preprocessing der Metadaten beim STT4SG-350 eine ID hinzugefügt, die sich aus der Client-ID und dem Dateinamen zusammensetzt. Dies wurde so gewählt, dass sie sich sicher von den IDs im SDS-200-Korpus unterscheiden und später beide Korpora gemischt werden konnten. Beim SDS-200-Korpus waren bereits IDs in Form einer fortlaufenden ganzen Zahl vorhanden. Im nächsten Schritt wurden als ungültig gekennzeichnete Samples entfernt. Da in den Experimenten die im STT4SG-350 definierten Dialektregionen als Label verwendet werden sollten, mussten diese als Nächstes im SDS-200-Korpus hinzugefügt werden. Die Dialektregionen konnten mithilfe der im SDS-200-Korpus vorhandenen Angaben zur Postleitzahl hergeleitet werden. So wurde für Samples mit der gleichen Postleitzahl die Dialektregion übernommen. Kamen im STT4SG-Korpus für die gleiche Postleitzahl mehrere Dialektregionen vor, wurden Samples im SDS-200-Korpus mit dieser Postleitzahl verworfen. Die bereinigten Metadaten konnten schliesslich in einem einzigen CSV abgespeichert werden, um die Korpora zu mischen.

6.2 Transkriptionen und deren Aufbereitung für das Training

Alle im HDF5 enthaltenen Audiosamples wurden mit jedem der drei in Abschnitt 4.1 vorgestellten Phoneme-Recognizern je einmal transkribiert. Dies dauerte mit der in Abschnitt 5.4 erwähnten APU-VM etwa 20 Stunden pro Phoneme-Recognizer und musste genau einmal für das ganze Projekt gemacht werden. Die Phoneme-Recognizer wurden von Hugging Face bezogen, der beliebten Austauschplattform für Machine-Learning-Modelle [89]. Der Reihe nach wurde mithilfe der ID in den Metadaten auf ein Audiosample im HDF5 zugegriffen. Ein Sample wurde zuerst einem Processor für die Tokenisierung übergeben, dem eigentlichen Phoneme-Recognizer-Modell, um die Phoneme vorherzusagen, und dann vom zum Processor gehörigen Decoder zu einer finalen Transkription umgewandelt. Sämtliche Transkriptionen eines Phoneme-Recognizers wurden in einer neuen CSV-Datei zusammen mit den wichtigsten Metadaten inklusive Dialekt-Label (clip_id, client_id, sentence, dialect_region) abgespeichert. Sobald die Transkriptionen erstellt waren, musste nicht mehr mit den ursprünglichen Audiodaten gearbeitet werden. Die Klassifizierungsexperimente konnte direkt mit den Transkriptionen durchgeführt werden.

Bevor ein Classifier trainiert werden konnte, musste ein über die Dialektregionen ausgeglichenes Datenset aus den kombinierten Korpora zusammengestellt werden und dieses in ein Trainings-, Validierungs- und Testingset unterteilt werden. Um die Ergebnisse dieser Bachelorarbeit mit der vorangehenden Projektarbeit vergleichen zu können, wurde das Datenset beziehungsweise der Split, der zu den besten Ergebnissen führte, für sämtliche Experimente übernommen [4]. Im Weiteren wird dieses übernommene Datenset «mixed-PA» genannt für «gemischte Korpora» und «Projektarbeit». Die Verteilungen der Samples und Sprecher:innen ist in der Abbildung 12 zu sehen.

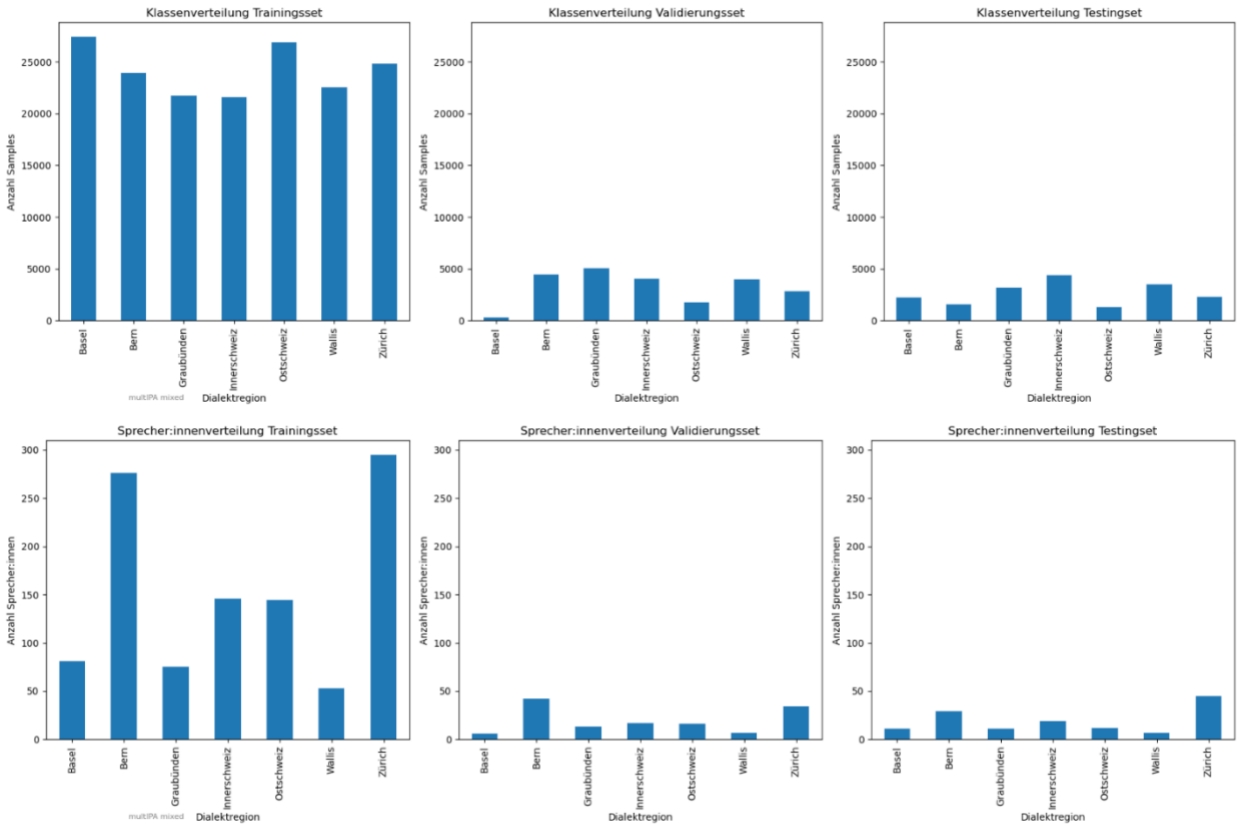

Abbildung 12: Klassen- und Sprecher:innenverteilungen des Splits

Erst für die Verifizierung der Ergebnisse wurde nochmals ein neues Datenset generiert (siehe Abschnitt 6.1). Die Technik, um Datenset und Splits zu generieren, wurde aus der Projektarbeit übernommen und so angepasst, dass diese direkt auf die Transkriptionsdateien angewendet werden konnte [4]. Zunächst wurden die Anzahl Samples pro Dialektregion im Datenset ausgeglichen, sodass die Modelle gleichmässig für alle Dialektregionen trainiert werden konnten.[8] Es wurden alle Dialektregionen auf 30'000 Samples reduziert. Dies ist etwas weniger als 34'404, die Anzahl Samples Graubündens, die die Dialektregion mit den wenigsten Samples ist. Entfernt wurden nur Samples von den Sprecher:innen mit den meisten Samples, um das Datenset bezüglich Sprecher:innen möglichst divers zu halten. Das resultierende Datenset wurde im Verhältnis 80:10:10 in Trainings-, Validierungs-[9] und Testingset aufgeteilt. Beim Splitverfahren wurde speziell darauf geachtet, dass die Ausgeglichenheit der Anzahl Samples pro Dialektregion erhalten bleibt, sich aber keine Sprecher:innen über den Split hinweg überlappen. Weder die verwendete Library scikit-learn noch andere Libraries bieten eine Möglichkeit für einen dreiteiligen Split. Daher wurde der Split in einem selbstentwickelten iterativen Verfahren ermittelt, der die Bedingungen am besten erfüllt.

[8] Die Anzahl Samples pro Dialektregion ist im STT4SG-350-Korpus ausgeglichen, jedoch nicht im dazugenommenen SDS-200-Korpus, wie in den Abschnitten 3.1 und 3.2 erläutert wurde.
[9] Wie bereits in Abschnitt 4.7 erwähnt, wird das Validierungsset nur für die SVM verwendet. Ansonsten werden Trainings- und Validierungsset für die Cross-Validation konkateniert.

7. Experimente

Ziel der Experimente war es, das Potenzial der in Kapitel 4 vorgestellten Methode zu überprüfen. Dazu wurden verschiedene Kombinationen folgender Faktoren überprüft: Classifier, Phoneme-Recognizer, Länge der n-Gramme, Tf-idf-Gewichtung und Anzahl konkatenierter Transkriptionen. Es sollte nicht nur ermittelt werden, welche Kombinationen die besten Resultate ergeben, sondern auch wie sehr die einzelnen Faktoren die Resultate beeinflussen. Bei der Reihenfolge der Experimente wurde iterativ und selektiv vorgegangen. In jeder Experimentenreihe[10] wurden einzelne Faktoren variiert, während andere konstant gehalten wurden. Dabei wurden die bisher aussichtsreichsten Ausprägungen als konstante Faktoren übernommen. Da es bei keinem Faktor klar bessere Ausprägungen gab, musste in mehreren Iterationen die Menge der besten Ausprägungen immer weiter verkleinert werden. Bei Experimenten mit längeren Laufzeiten wurden aus Zeitgründen die vielversprechendsten Kombinationen selektiert und andere nicht weiterverfolgt.

In den folgenden Abschnitten wird zuerst in einer Voranalyse ein erstes Bild der Phonemtranskriptionen vermittelt. Im nächsten Abschnitt werden erste Probeexperimente beschrieben, wie das Hyperparameter-Tuning der Classifier daraus abgeleitet wurde und welche Hyperparameter für die Experimente ausgewählt wurden. Die darauffolgenden Abschnitte gelten den verschiedenen Experimentenreihen und im letzten Abschnitt wird ein Fazit zum Potenzial der Methode gezogen.

7.1 Voranalyse der Daten

In diesem Unterkapitel werden die von den drei Phoneme-Recognizern Wav2Vec2Phoneme, MultIPA und Dutch generierten Phonemtranskriptionen verglichen und analysiert. Die Voranalyse hat zum Ziel, vor Beginn der Experimente die Phonemtranskriptionen auf ihre Qualität und Konsistenz hin genauer zu betrachten und ihre Eignung für die Aufgabe der Klassifikation schweizerdeutscher Dialekte zu prüfen.

7.1.1 Betrachtung eines Satzbeispiels

Die Phonemtranskriptionen in Tabelle 1 für den Satz «Dafür haben wir Verständnis.» in Bündnerdeutsch sind grundsätzlich gut lesbar. Allerdings ist der Anfangsteil v ɪː r d aː r z in der vom Dutch erzeugten Transkription schwerer zu lesen. Ohne die anderen beiden Phonemtranskriptionen von Wav2Vec2Phoneme und MultIPA wäre nicht offensichtlich, dass das hochdeutsche Wort «dafür» ins Schweizerdeutsche «für das» übersetzt wurde. Dies veranschaulicht, wie die Sprecher:innen die vorgegebenen Sätze aus dem Hochdeutschen in ihren Dialekt übersetzt haben, wie in den Abschnitten 3.1 und 3.2 beschrieben.

[10] Die Abschnitte 0 - 7.6 entsprechen je einer Experimentenreihe. Es ist ausserdem zu beachten, dass die Experimentenreihen der Abschnitte 0 und 7.5 parallel stattgefunden haben.

Satz:	Dafür haben wir Verständnis
Transkription mit Dutch:	v ɪː r d aː r z aː n m ɛ r v ə r s t aː n n ɪ s
Transkription mit Wav2Vec2phoneme:	f y r d a s h a m m ə r f ɛ r ʃ t a n t n ɪ ts
Transkription mit MultIPA:	fyrdasammurfɛrɕtannis

Tabelle 1: Transkriptionen für einen zufälligen Satz aus dem Datensatz

Beim Vergleich der Transkriptionen fällt auf, dass der erste Teil mit f y r d a s bei MultIPA und Wav2Vec2Phoneme mit denselben Phonemen übersetzt wurde. Weiter fällt auf, dass bei diesem Satzbeispiel Wav2Vec2Phoneme mehr phonemische Details als Dutch und MultIPA erfassen konnte. Dem ausgesprochenen «d» im Wort «Verständnis» wird das Phonem /t/ zwischen den beiden /n/ zugewiesen. Dieses «d» wird sowohl von Dutch als auch MultIPA nicht erkannt.

7.1.2 Gleicher Satz und verschiedene Dialektregionen

Ein weiterer Vergleich der Phonemtranskriptionen über verschiedene Dialektregionen hinweg zeigt, dass alle drei Phoneme-Recognizer erwartungsgemäss, wie in untenstehender Tabelle 2 ersichtlich ist, unterschiedliche Transkriptionen liefern:

Phoneme-Recognizer	**Transkription für Satz:** **«Zürich ist eine wunderschöne Stadt»**	**Dialektregion**
Dutch	t ɵ d i s ə ʊ u n t ə r s ɵ n ɪ s t ɑ p t	Graubünden
	t y r aː r ɪ r s ə ʊ ɔ n d ə r s øː r n ɪ ŋ s t ɑ t	Basel
	z ɪː r r ɪ x i z ʌʊ ʊ ɔ n t ə n z ʊ ɛ m ɪ s t ɑ t	Ostschweiz
	s y r i ɪ s ə n ʊ ɔ n t ɛ ʃ øː n i s t ɑ t	Innerschweiz
	t ɛ r ɪ r s ə r v ə r ɔ n d ə r t ə r s p ɛ r n ə ɪ ŋ s t aː t	Bern
	z ɪː r ə x ɪ s ə n t ʊ u r n d ɛ ʃ eː n i s t ɵ t	Wallis
	t r y r i ɪ s h ə r ʊ u n d ə r s p ɵ n ə s t ɑ t	Zürich
Wav2Vec2Phoneme	ts uː r ɪ ʃ ə v oː n t ə ʃ œ n ə ʃ t a t	Graubünden
	ts ɪ r eː ɪ ʃ ə v oː ə n t ə ʃ ɜː n ɪ ʃ t ɑː t	Basel
	ts ɪ r ɪ x ɪ ʃ ɛ v oː n t ə ʃ v ɛ n ɪ ʃ t a t	Ostschweiz
	ts uː ɾ iː ɪ s t ɛ n v oː n d ə ʃ ø n ɪ ʃ t a t	Innerschweiz
	ts ø r iː ʃ eː v h oː ə t r ʃ m ɛ t n iː ʃ t a	Bern
	ts y r ɪ x ɪ s ʊ n v ʊ n d ɜ ʃ ɜː n ɪ ʃ t oː t	Wallis
	ts yː r ɪ ɪ ç ɛ v ʊ n d ə ʃ ø n ɪ ʃ ʃ t a t	Zürich
MultIPA	t͡sɯriiʂɐ̃vuntɯɯniːɕtaätː	Graubünden
	t͡sɨjarjɛ̃ːʂɐbuwtʂɨrɲizɖɔt	Basel
	psivɨxiɕɛ̃mɔnt͡sɛ̃ʂvɐjɲiɕta	Ostschweiz

	t͡sɯriizɛnbo̥ndəʂɨɲiʂtɔt:	Innerschweiz
	siɾi:ʃɛkumtrʃʃɛdniʃta	Bern
	t͡sɨiɾixljiʃɛnvundaʃiɾniʃtot:	Wallis
	t͡sy:ɾiiʃɛvundaʃwiniʃtat:	Zürich

Tabelle 2: Beispiel von Transkriptionen für den gleichen Satz aus verschiedenen Regionen

Es fällt auf, dass die Phoneme in der von MultIPA generierten Phonemtranskription nicht durch Leerzeichen getrennt sind. Darauf muss später in der Verarbeitung Rücksicht genommen werden.

Weiter zu beobachten ist, dass Wav2Vec2Phoneme durchgängig das gleiche Phonem /ts/ zu Beginn über alle Dialektregionen hinweg verwendet.

Dutch zeigt grössere Schwankungen in der Länge der Phonemtranskriptionen als Wav2Vec2Phoneme und MultIPA. Die Transkription für den Dialekt Bern ist bei Dutch auffällig länger.

7.1.3 Gleicher Satz und gleiche Dialektregion

Gleiche Sätze aus der gleichen Dialektregion, jedoch von unterschiedlichen Sprecher:innen, werden von allen drei Phoneme-Recognizern unterschiedlich transkribiert, wie in der Tabelle 3 erkennbar ist.

Satz	**Phonemtranskription**	**Region**	**Phoneme-Recognizer**
Ihre Stimme hat Power und kratzt.	ɪ ɾ r ɪ s t i m h ə t p ʌʊ ə r k ɔ n ɣ a: r ɑ t s t	Ostschweiz	Dutch
Ihre Stimme hat Power und kratzt.	i r ɪ s t i m h ə t p a: l b ə ə n ɣ r a: t s t	Ostschweiz	Dutch
Übertreiben Sie da nicht ein wenig?	y p ə t r i: b ə ts i: d aʊ n y t ɛ s p i: ts ə l	Zürich	Wav2Vec2Phoneme
Übertreiben Sie da nicht ein wenig?	y b ə t r i: b ə t z i: d ɔ n ə d ɛ s p i: ts l i	Zürich	Wav2Vec2Phoneme
Über den Aufwand müsse man jedoch weiterhin diskutieren.	bɾɛwfɔmtmijusmɛjatɔxvittɾindiskut:i:ɛɾɛ	Bern	MultIPA

Über den Aufwand müsse man jedoch weiterhin diskutieren.	øprɐu:famtmywɨsnaɒbrbit:rhindiskot:ije:rɛ	Bern	MultIPA

Tabelle 3: Beispiel von Transkriptionen für den gleichen Satz aus gleicher Region

Die Unterschiede sind damit zu begründen, dass in der gleichen Dialektregion Variationen in demselben Dialekt, aber auch individuelle Unterschiede in der Aussprache der Sprecher:innen vorkommen. Dies verdeutlicht nochmals die allgemeine Herausforderung der Phonemtranskription und folglich auch der Dialekterkennung.

Es gibt nur sehr wenige Ausnahmen, die zur gleichen Phonemtranskription führen. Diese Ausnahmen sind sehr kurze Sätze wie beispielsweise «Das ist so», die mit Wav2Vec2Phoneme für die Dialektregionen Ostschweiz und Innerschweiz die gleiche Transkription « d a s ɪ ʃ s oː» ergeben. Im Gegensatz dazu liefert MultIPA für denselben Satz unterschiedliche Ergebnisse: «dasjiʃsu» für die Ostschweiz und «taseʃsu» für die Innerschweiz. Für Wav2Vec2Phoneme wurden zehn Sätze gefunden, die die gleiche Transkription aufweisen, während es bei Dutch nur ein Satz ist. MultIPA hingegen zeigt für keinen Satz die gleiche Phonemtranskription und scheint mehr Feinheiten einfangen zu können als Wav2Vec2Phoneme und Dutch.

7.1.4 Stabilität der Phonemtranskriptionen

Um die Verlässlichkeit der Phonemtranskriptionen für die einzelnen Phoneme-Recognizer zu überprüfen, wurden die Transkriptionen für jede Region halbiert und die relative Häufigkeit der Phoneme je für die Hälften berechnet. Wie in den Abbildung 13 beispielhaft für die Regionen Wallis und Basel ersichtlich, zeigen beide Hälften hohe Übereinstimmungen in den Verteilungen. Alle drei Phoneme-Recognizer scheinen konsistent und zuverlässig ihre Transkriptionen zu erzeugen.

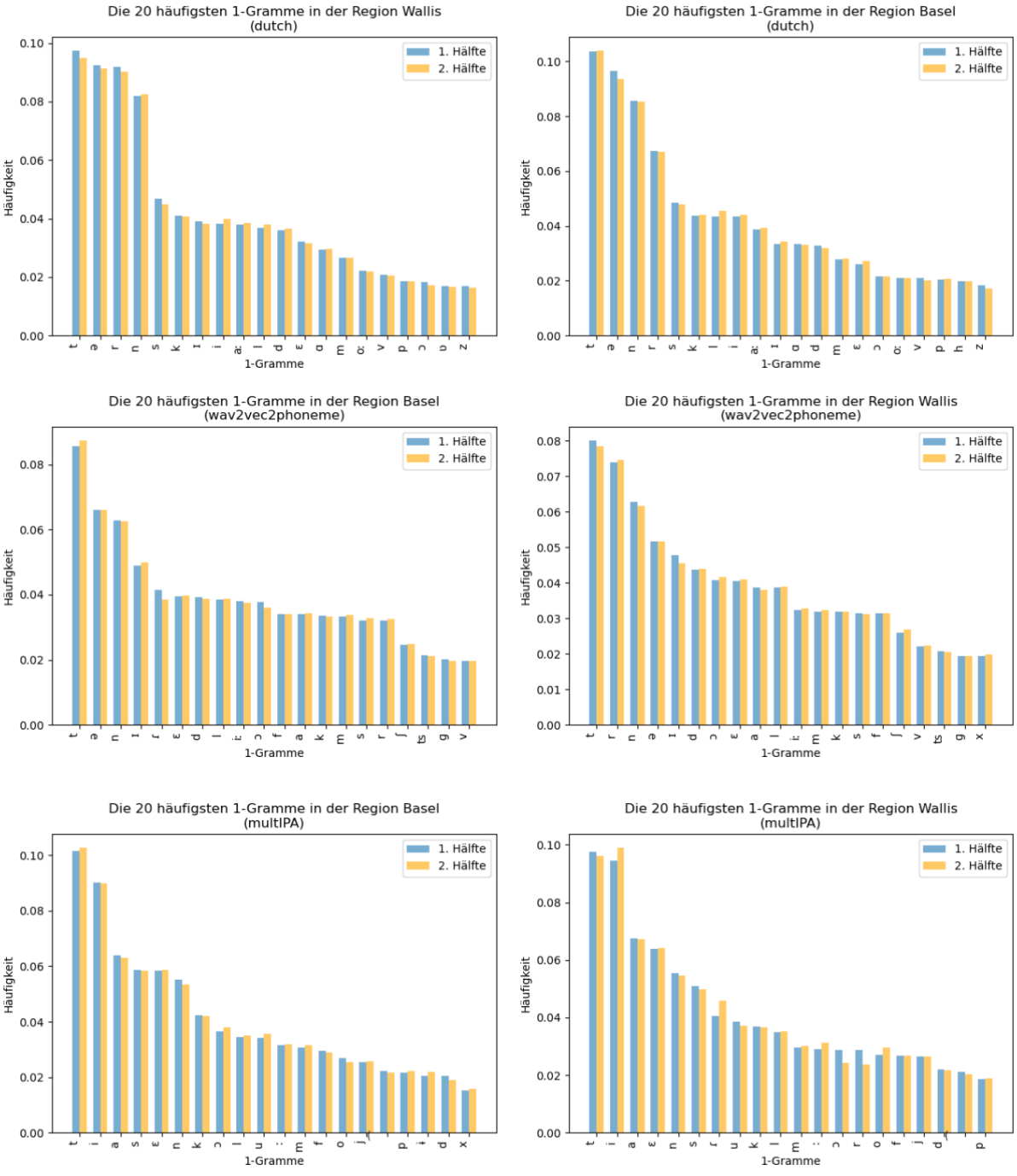

Abbildung 13: Vergleich der relativen Häufigkeiten von Phonemen über zwei Hälften am Beispiel der Dialektregionen Wallis und Basel

7.1.5 Unterschiede in den Phonemtranskriptionen

Es ist davon auszugehen, dass die beobachteten Unterschiede in den Phonemtranskriptionen zwischen den Phoneme-Recognizern mit der unterschiedlichen Entwicklung beziehungsweise dem Training der Modelle zusammenhängen. Wie im Abschnitt 4.1.2 bereits beschrieben, wurde Wav2Vec2Phoneme auf automatisch generierten Phonemen aus einer Vielzahl an Sprachen trainiert. Das für die Phonemtranskriptionen eingesetzte Modell verfügt mit 388 Phonemen über das grösste Vokabular [90]. Das Modell MultIPA hingegen verfügt über 311 Phoneme und wurde mit semiautomatisch generierten Phonemen aus weniger Sprachen trainiert [91]. Der Fokus lag beim Pretraining auf der Qualität der Phonemgenerierung, indem Sprachen mit einer konsistent orthografischen Zuordnung ausgewählt wurden. Der Ansatz zielte darauf ab, präzisere Transkriptionen zu ermöglichen, was sich in den obigen Beobachtungen widerspiegelt. Dutch, das speziell für die niederländische Sprache finegetunt wurde, weist ein kleines Vokabular von 53 Phonemen auf [92].

7.1.6 Phonemverteilungen über Dialektregionen

Die Abbildung 14 - Abbildung 16 zeigen die relativen Phonemhäufigkeiten für jede Dialektregion, wobei jeder Farbbalken ein spezifisches Phonem repräsentiert. Für Dutch sind die relativen Häufigkeiten der einzelnen Phoneme über alle Regionen hinweg ähnlich verteilt. Grössere Unterschiede zwischen den Phonemverteilungen in den Regionen sind hingegen für Wav2Vec2Phoneme und MultIPA zu verzeichnen.

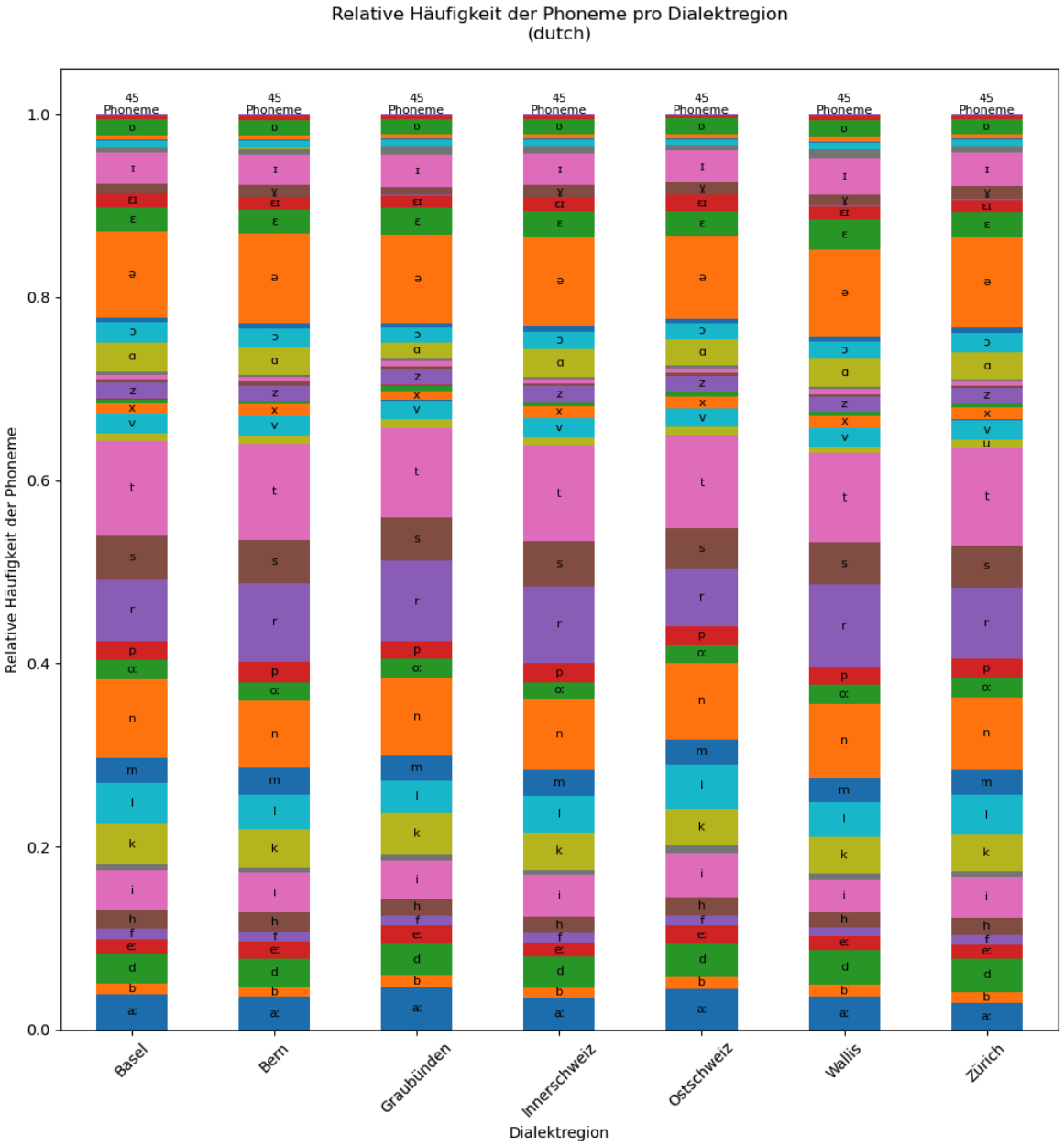

Abbildung 14: Verteilung der Phonemhäufigkeiten für Dutch

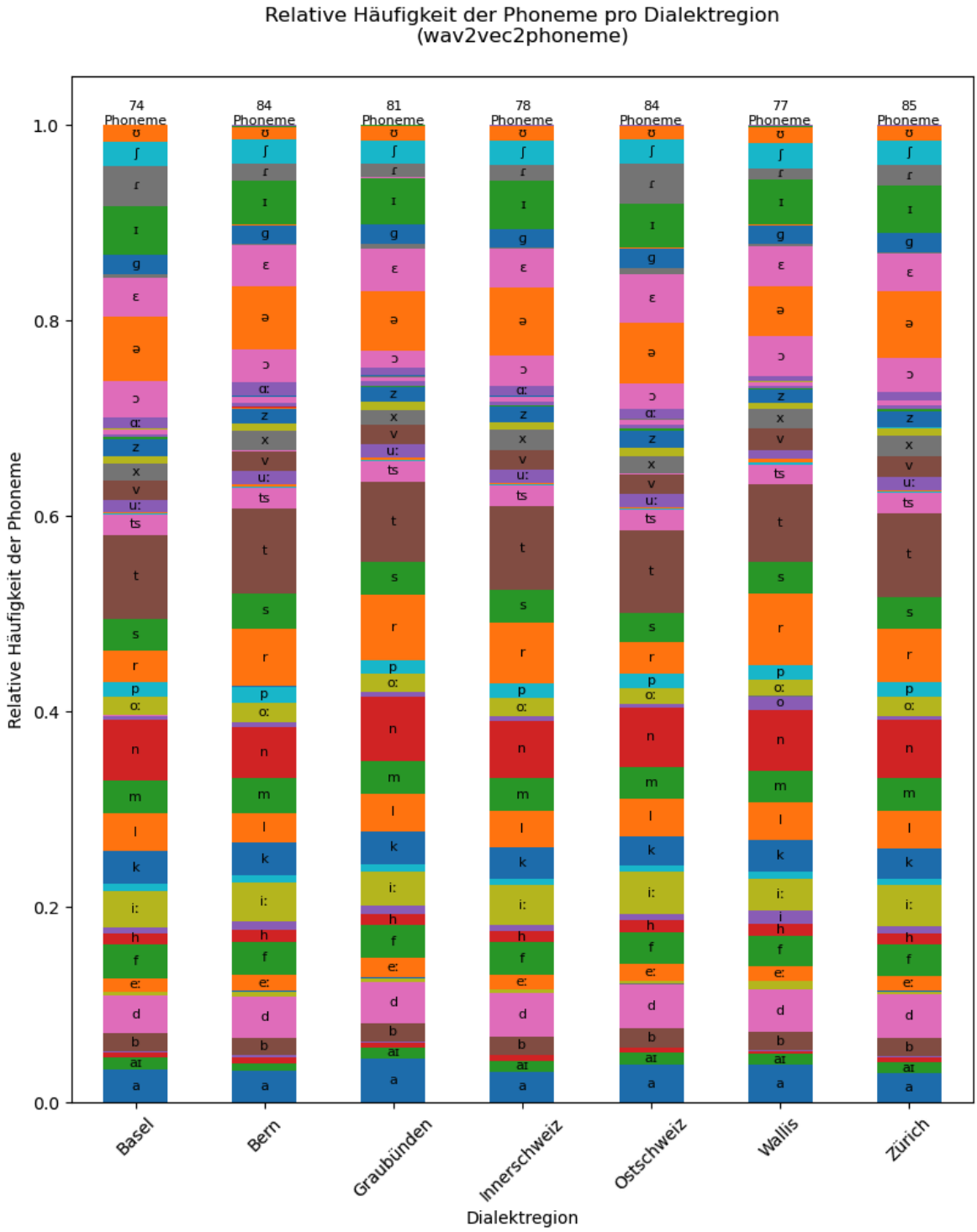

Abbildung 15: Verteilung der Phonemhäufigkeiten für Wav2Vec2Phoneme

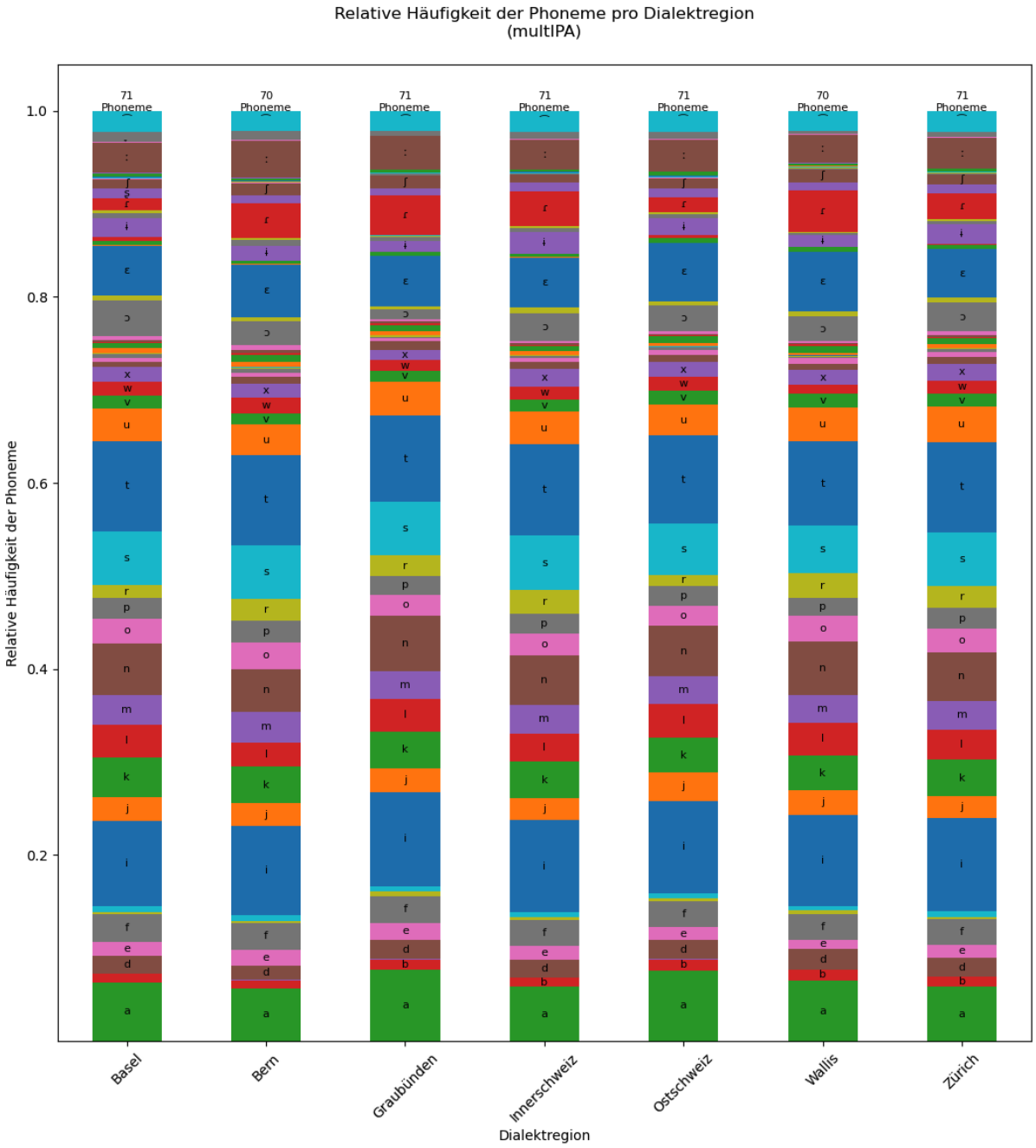

Abbildung 16: Verteilungen der Phonemhäufigkeiten für MultIPA

Oberhalb jedes Balkens ist in den Abbildung 14 - Abbildung 16 die Anzahl unterschiedlicher Phoneme für den jeweiligen Dialekt vermerkt. Wav2Vec2Phoneme weist mit 74 unterschiedlichen Phonemen für Basel und 85 für Zürich die grösste Spanne auf. Bei MultIPA bewegen sich die unterschiedlichen Phoneme pro Dialekt zwischen 70 und 71. Allerdings wurden aufgrund des Formats der Transkriptionen, wie zu Beginn des Kapitels unter Abschnitt 7.1.2 beschrieben, nur die unterschiedlichen Zeichen für MultIPA gezählt und

nicht die tatsächlichen Phoneme. Bei Dutch erhält jeder Dialekt die gleiche Anzahl von Phonemen, nämlich 45.

Die Verteilungen der Häufigkeiten für Wav2Vec2Phoneme und MultIPA zeigen jetzt schon vielversprechende Unterschiede, die eine Dialekterkennung ermöglichen können.

Beispielsweise kommt das Phonem /o/ bei Wav2Vec2Phoneme im Wallis mit einer grösseren relativen Häufigkeit im Vergleich zu den anderen Dialekten vor (siehe in Abbildung 15 violett /o/). Auch in der Abbildung 16 für MultIPA ist beispielsweise /ɔ/ in Graubünden klar unterdurchschnittlich vertreten im Vergleich zu den anderen Dialektregionen (siehe in Abbildung 16 graufarbig /ɔ/).

7.1.7 Eignung für Dialekterkennung

Der Ansatz, die Dialekterkennung über die Phonemtranskriptionen anzugehen, scheint vielversprechend. Es bestehen Unterschiede in den Transkriptionen gleicher Sätze sowie in der Verteilung der Phonemhäufigkeiten zwischen den Dialekten. Ausserdem sind die erzeugten Phonemtranskriptionen bei allen Phoneme-Recognizern beständig. Je nach Phoneme-Recognizer resultieren unterschiedliche Phonemtranskriptionen. Dutch fällt im Vergleich zu MultIPA und Wav2Vec2Phoneme qualitativ ab, da die Verteilungen über die Regionen hinweg zu gleichmässig sind und die Transkriptionen teilweise schwer lesbar sind. Aus diesen Gründen werden die Transkriptionen von Dutch nicht für die Experimente eingesetzt.
Offen bleibt, ob Wav2Vec2Phoneme oder MultIPA die schweizerdeutschen Phoneme besser erfassen kann. Ein grober Abgleich mit den schweizerdeutschen Phonemen in Phoible, einer phonetischen Datenbank für unzählige Sprachen, hat ergeben, dass sowohl Wav2Vec2Phoneme als auch MultIPA teilweise andere Phoneme verwenden [93].

Die Qualität der Phonemtranskriptionen hinsichtlich ihrer linguistischen Eignung zur Darstellung von schweizerdeutschen Dialekten lässt sich zwar mit dieser Analyse nicht beurteilen. Es lassen sich jedoch ausreichend Unterschiede in den Transkriptionen und Verteilungen der Phonemhäufigkeiten zwischen den Dialekten feststellen, sodass damit Experimente zur Dialekterkennung durchgeführt werden können.

Wie geeignet die Phonemtranskriptionen für die schweizerdeutsche Dialekterkennung sind, wird sich während den Experimenten zeigen.

7.2 Probeexperimente und Hyperparameter-Tuning

Um bei den Experimenten das Beste aus den Classifiern herausholen zu können, mussten zunächst deren Hyperparameter getunt werden. Dafür wurden wiederum erste Probeexperimente nötig, sodass eingeschätzt werden konnte, welche Ausprägungen der wichtigsten Faktoren wie n-Gramm-Länge, Tf-idf-Gewichtung und Sampellänge sinnvoll sind.

Als erste Probeexperimente wurden Kombinationen mit n-Grammen und Tf-idf-Gewichtung zufällig mit Wav2Vec2Phoneme oder MultIPA und allen Classifiern getestet. Daraus zeichnete sich ab, dass n-Gramme für $2 < n < 5$ bessere Resultate liefern. Mit diesem Wissen wurde der NB-Classifier mit 3-Grammen über einen Grid-Search am Validierungsset getunt.

Der NB hat einen Hyperparameter: den in Abschnitt 4.2.1 erklärten Glättungsparameter α, der in skit-learn als alpha bezeichnet wird. Es zeigte sich, dass dieser kaum einen Einfluss auf das Resultat hat, solange er in

einem sinnvollen Bereich von 0.1-1.0 gewählt wurde. Aus diesem Grund wurde später verzichtet, ein weiteres Tuning mit Cross-Validation vorzunehmen.

Für die restlichen Classifier wurde eine weitere Serie Probeexperimente mit einer Verlängerung der Samples durchgeführt. (In Abschnitt 0 wird nochmals genauer erklärt, was damit gemeint ist.) Dies zeigte, dass längere Samples höhere Macro-F1-Scores erreichen. Aus diesem Grund wurden die restlichen Classifier mit einer zufällig gewählten Verlängerung von 10 und 15 aneinandergehängten Transkriptionen und 3-Grammen über einen Grid-Search getunt. Dabei wurde mit einer Cross-Validation die durchschnittliche Validation-Accuracy über alle Folds hinweg maximiert. Für LR wurden die folgenden Hyperparameter der skit-learn-Implementierung geprüft: Solver, C, max_iter und Penalty [84]. Der Solver beschreibt den in Abschnitt 4.2.2 erwähnten Optimierungsalgorithmus für den Trainingsprozess und max_iter, wie oft der Solver maximal iterieren darf, um zu konvergieren. Die Regularisierungsstärke C verhindert, dass die Koeffizienten zu klein werden und es zu einem Overfitting kommt. Die Penalty schränkt schliesslich die Koeffizienten auf einen bestimmten Bereich ein. Beim Tuning hat sich schwächere Regularisierung (grösseres C) in Kombination mit der gängigen L2-Regularisierungstechnik bewährt. Für die restlichen Parameter kann keine klare Aussage getroffen werden, wobei bei den Solvern nur solche ausprobiert wurden, die sich für grössere Datensätze eignen. Für den SVM wurden drei Parameter getestet: der Regularisierungsparameter C, der Kernel und der Kernel-Koeffizient gamma. C steuert den Einfluss der in Abschnitt 4.2.3 erläuterten Slackvariablen. Der Kernel bestimmt die Beschaffenheit der in Abschnitt 4.2.3 vorgestellten Hyperplane, die linear oder nicht linear mit zum Beispiel einem polynomialen oder Radial-Bias-Kernel sein kann. gamma ist ein Parameter für nicht-lineare Kernel, der bestimmt, wie viel Gewicht ein Trainingssample hat. Beim Tuning des SVM hat sich vor allem der lineare Kernel als besser herauskristallisiert, wobei einige wenige Durchläufe mit einem Radial-Basis-Kernel ähnlich gut ausfielen. Bei Letzterem machte gamma den Unterschied. C schien generell nicht entscheidend. RF und XGB haben zahlreiche Hyperparameter, wobei für das Tuning die wichtigeren ausgewählt wurden. Diese sind beim RF folgende Hyperparameter, wie sie in Abschnitt 4.2.4 erklärt wurden: die Anzahl Entscheidungsbäume (n_estimators), die maximale Tiefe eines Baumes (max_depth), die minimale Anzahl Samples in einem Split (min_samples_split) und in einem Blatt (min_samples_leaf). Als Mass für die Aufteilungsgüte wurde konstant Gini verwendet. Bei XGB wurden verschiedene Werte für die Anzahl Bäume (n_estimators), die maximale Tiefe eines Baumes (max_depth), Gewichte von Kindsbäumen (min_child_weight) und Regularisierungsparameter (reg_lambda, reg_alpha) getestet, wie sie in Abschnitt 4.2.5 dargelegt wurden. Bei RF und XGB war der Bereich der erreichten Average-Validation-Accuracy deutlich kleiner als bei den anderen Classifiern (abgesehen vom NB). Bei beiden war eine grössere Anzahl Bäume besser. Zu den weiteren Parameter kann keine genaue Aussage getroffen werden. Wegen der teils langen Laufzeit der Entscheidungsbaumalgorithmen wurde jedoch auf ein weiteres Tuning verzichtet, das mehr Einsicht hätte bringen können.

Die resultierten besten Hyperparameter-Konfigurationen, die in Tabelle 4 dargestellt sind, wurden für sämtliche Experimente verwendet, um diese vergleichen zu können. Auf weiteres Tuning musste aus Zeitgründen verzichtet werden. Die genauen Werte, die im Hyperparameter-Tuning genutzt wurden sowie die entsprechenden Resultate sind in den Abbildungen im Anhang unter Abschnitt 12.3.2 zu finden.

Classifier	Phoneme-Recognizer	Anzahl aneinander-gehängter Transkriptionen	n-Gramm-Länge	Beste Konfiguration
NB	MultIPA	1	3	alpha: 0.8
LR	Wav2Vec2Phoneme	15	3	solver: saga C: 100 max_iter: 200 penalty: l2
SVM	Wav2Vec2Phoneme	10	3	C: 1.0 kernel: linear
RF	Wav2Vec2Phoneme	10	3	n_estimators: 1000 max_depth: null min_samples_split: 5 min_samples_leaf: 2 criterion: gini random_state:42
XGB	Wav2Vec2Phoneme	10	3	n_estimators: 1000 max_depth: 3 min_child_weight: 2 learning_rate: 0.5 reg_lambda: 1 reg_alpha: 7

Tabelle 4: Die besten Hyperparameter aus dem Tuning

7.3 Experimentenserie 1: Konkatenation von Transkriptionen

In dieser Experimentenserie wurden mehrere Transkriptionen beziehungsweise Sätze[11] der:des gleichen Sprecher:in zu neuen, längeren Samples konkateniert. Obwohl davon ausgegangen wird, dass dank der Phonemtranskription der Audiodaten sprecher:innenspezifische Eigenschaften deutlich reduziert werden, ist dennoch damit zu rechnen, dass sich Sprecher:innen nach wie vor unterscheiden könnten. Daher sollten die Samples nach Sprecher:innen differenzierbar bleiben. Übrigbleibende Transkriptionen, die nicht mehr für die gewünschte Anzahl zu konkatenierenden Transkriptionen ausreichten, wurden verworfen. Insgesamt bleibt die Menge an Transkriptionsdaten also in etwa gleich, während sich die Anzahl Samples mit einer steigenden Anzahl konkatenierter Sätze verringert.

Im ersten Durchlauf wurden die Samples im Trainings- und Testingset zu einer gleichen Anzahl Sätze verlängert. Als Feature wurde für alle fünf Classifier gemäss ersten Erkenntnissen aus den beschriebenen Probeexperimenten 3-Gramme gewählt. Abbildung 17 zeigt, dass der F1-Macro-Score für beide Phoneme-Recognizer und alle Classifier mit einer steigenden Anzahl Sätze grösser wird. NB und RF schneiden mit

[11] Im Weiteren wird für die Verlängerung von Samples von der Anzahl Sätze gesprochen. Im ursprünglichen Datenset ohne Sampleverlängerung entspricht jedes Sample einer Transkription und genau einem Satz.

Wav2Vec2Phoneme besser ab. LR, SVM und XGB performen besser mit MultIPA. Werden die Classifier untereinander verglichen, fällt vor allem die Leistung von RF ab.

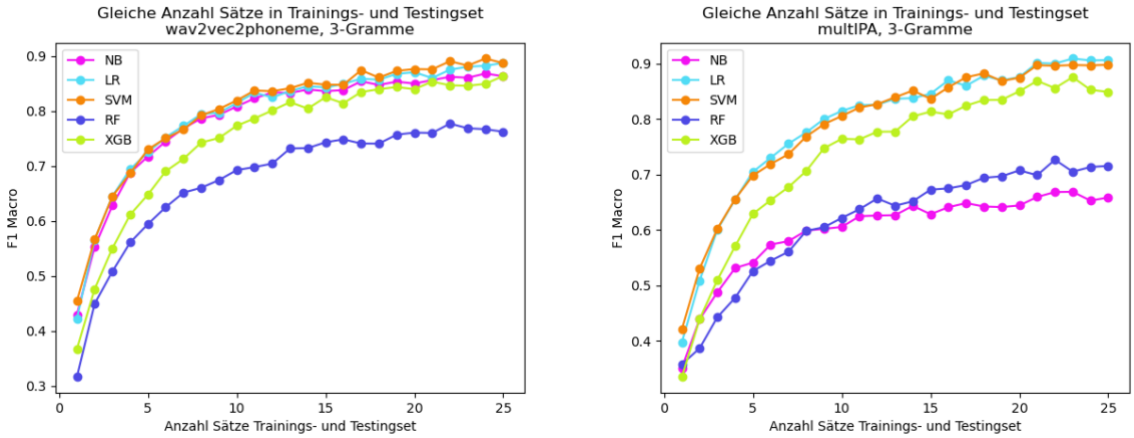

Abbildung 17: F1-Macro-Score mit einer zunehmenden Anzahl Sätze und Wav2Vec2Phoneme (links) oder MultIPA (rechts)

Im zweiten Durchlauf wurde bei den schnellsten beiden Classifier LR und NB überprüft, wie sich eine konstante Anzahl Sätze im Testing- beziehungsweise Trainingsset auswirkt, wenn die Anzahl Sätze im jeweils anderen Set variiert wird. Zunächst wurde eine Konstante von 20 Sätzen gewählt. In Abbildung 18 ist zu sehen, dass der Macro-F1-Score mit mehr Sätzen im Testingset steigt. Wird hingegen die Anzahl Sätze im Testingset konstant gehalten und die Anzahl Sätze im Trainingsset variiert, bewegt sich der Macro-F1-Score in einem sehr kleinen Bereich. Dies deutet darauf hin, dass es weniger relevant ist, wie lang die Samples im Trainingsset sind. Viel mehr kommt es auf die Samplelänge im Testingset an.

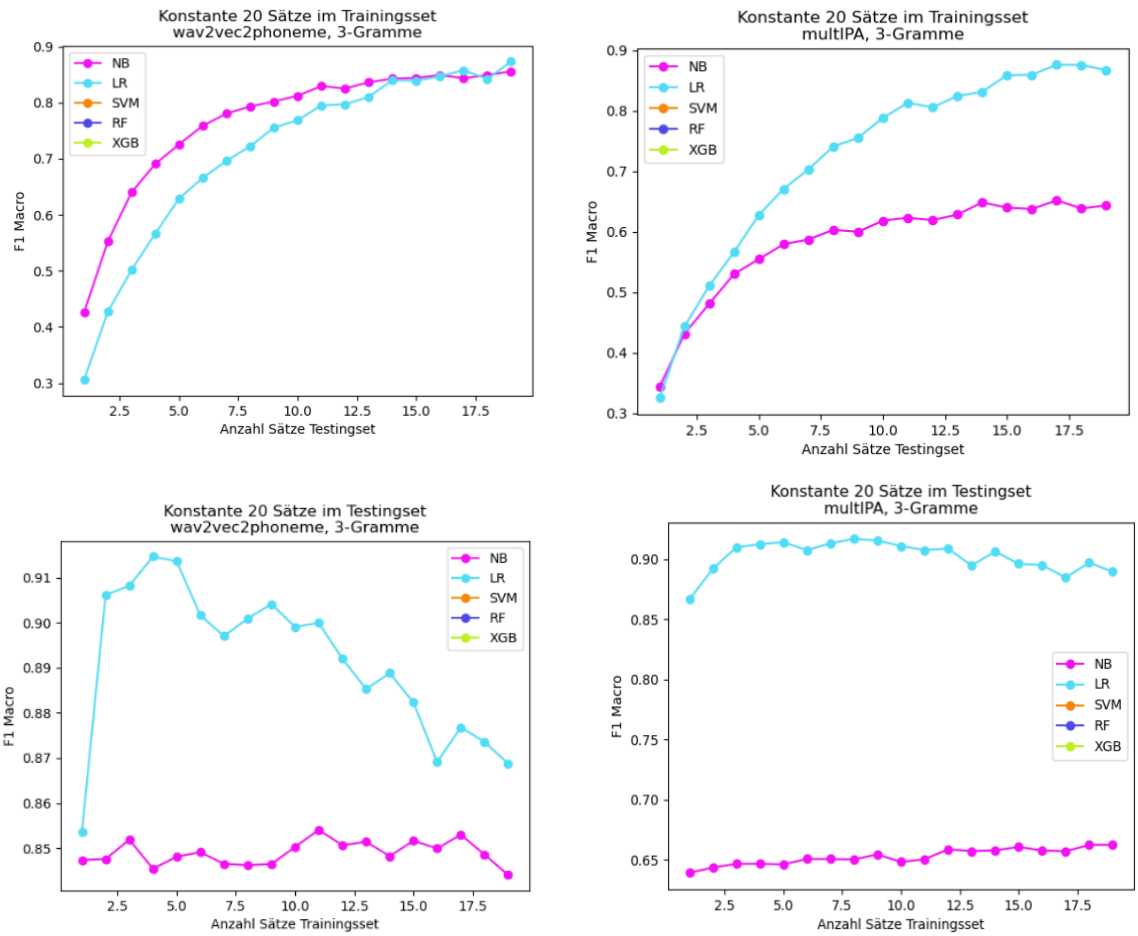

Abbildung 18: Konstante Anzahl Sätze im Trainingsset (oben) oder im Testingset (unten) bei MultIPA

Um diese Zusammenhänge näher zu untersuchen, wurden im dritten Durchlauf mit weiteren Kombinationen aus Anzahl Sätzen im Trainings- und Testingset experimentiert. Mit LR wurden zusätzlich zu den bestehenden Experimenten alle Kombinationen bis und mit 20 Sätzen getestet. Bei NB wurde auf 4-Gramme umgestellt, weil sich in parallelen Experimenten mit 4-Grammen besser herausstellten, und ebenfalls alle Kombinationen mit bis zu 20 Sätzen getestet. Da SVM mit mehr Samples eine exponentiell längere Laufzeit hat, wurden alle Kombinationen von 10 bis 20 Sätzen im Trainingsset und 1 bis 20 Sätzen im Testingset dazugenommen. Es wurden alle Experimente nur mit MultIPA durchgeführt, da sich bereits herauskristallisiert hatte, dass dieser (bis auf NB) zu besseren Scores führt. Mit XGB und RF wurden keine weiteren Experimente mit Verlängerungen gemacht, weil diese im Vergleich zu den anderen Classifiern eine längere Laufzeit haben und bisher keine besseren Scores erzielt hatten.

Die Resultate der Experimente werden in Abbildung 19 dargestellt. Die Vermutung aus dem zweiten Durchlauf bestätigt sich: Die Anzahl Sätze im Trainingsset spielt weniger eine Rolle, wohingegen eine grössere Anzahl Sätze im Testingset entscheidend ist. Bei LR und SVM ist zudem eine Tendenz zu einer mittleren Anzahl Sätzen im Trainingsset zu erkennen.

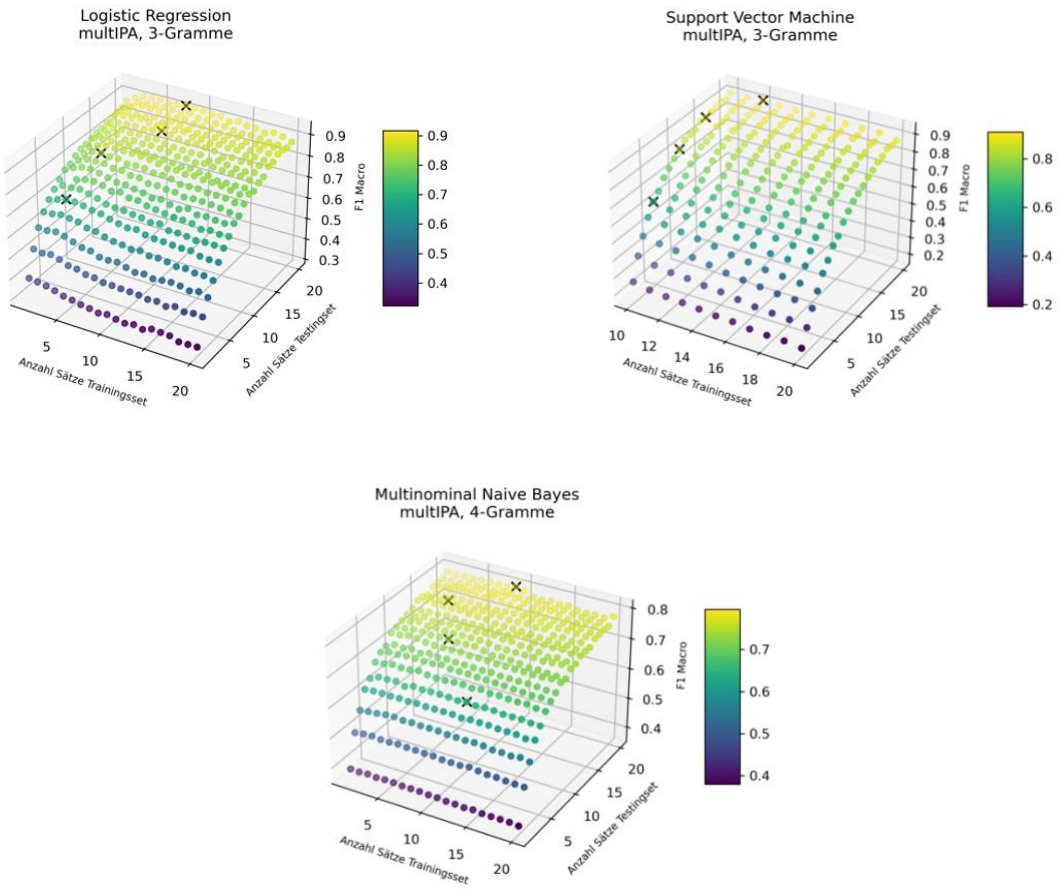

Abbildung 19: F1-Macro-Score mit sämtlichen Kombinationen der Anzahl Sätze im Trainings- und Testingset für drei Classifier

Im realen Anwendungsfall bedeutet dies, dass ein Classifier die Dialektregion bei längeren Audiobeispielen besser voraussagen kann als bei kürzeren.[12] Dabei lohnt es sich, zu überprüfen, wie hoch ein Score maximal sein kann bei einer gewissen Anzahl Sätze im Testingset. Denn je nach Anwendungsfall stehen nicht unbedingt genug lange Samples für Topscores zur Verfügung. Die besten Scores sind in der Abbildung 19 mit X markiert. Dabei wird nicht nur der insgesamt beste Score markiert, sondern auch die besten Scores für bis zu 15, 10 oder 5 Sätze im Testingset.

Die konkreten F1-Macro-Scores sind nochmals in der Tabelle 5 aufgelistet. Daraus ist abzulesen, dass jeweils die obere Grenze für die Anzahl Sätze im Testingset zu den besten Resultaten führt. Die dazugehörige ideale Anzahl Sätze im Trainingsset variiert von Classifier zu Classifier. Tendenziell liegen sie eher in der Mitte zwischen 1 und 20 Sätzen. Ausserdem ist festzustellen, dass auch für 10 bis 15 Sätze im Testingset noch akzeptable F1-Macro-Scores erreicht werden.

Wird mit der in den Abschnitten 3.1 und 3.2 erwähnten durchschnittlichen Dauer von 4.8 bis 5 Sekunden pro Sample gerechnet, wären dies umgerechnet 50- bis 75-sekündige Samples. Ob dies akzeptable Samplelängen sind und ob eine bestimmte Vorhersagegenauigkeit genügt, muss für den jeweiligen Anwendungsfall geprüft werden.

[12] Dies lässt sich intuitiv nachvollziehen.

Classifier	Obere Grenze der Anzahl Sätze im Testingset	Anzahl Sätze im Trainings- und Testingset	F1-Macro
LR	20	8/20	**0.916817**
	15	8/15	0.890394
	10	4/10	0.839097
	5	3/5	0.711225
SVM	20	12/20	0.90988
	15	10/15	0.872716
	10	10/10	0.805397
	5	5/5[13]	0.698133
NB	20	9/20	0.795407
	15	4/15	0.776624
	10	7/10	0.740461
	5	12/5	0.649105

Tabelle 5: Beste F1-Macro-Scores bis zu einer oberen Grenze der Anzahl Sätze im Testingset

7.4 Experimentenserie 2: Gleichzeitige Konkatenation und Augmentation der Transkriptionen

Als Alternative zur im vorherigen Abschnitt vorgestellten Transkriptions-Konkatenation, wurde in einer weiteren Experimentenserie gleichzeitig mit der Verlängerung eine Augmentation vorgenommen. Es wurde eine bestimmte Anzahl Transkriptionen zufällig aus allen vorhandenen Transkriptionen eine:r Sprecher:in ausgewählt und konkateniert. Pro Sprecher:in wurden so viele solche Samples generiert, wie die:der Sprecher:in im originalen Datensatz Samples hat. Die gleichen Transkriptionen beziehungsweise Sätze können also mehrmals ausgewählt und zusammen mit anderen Sätzen in einer anderen Reihenfolge zusammengefügt werden.

Die Experimente wurden mit den schnellsten Algorithmen LR und NB ausgeführt. Da es viel Zeit beansprucht, die neuen vergrösserten Datensets zu erstellen, beschränken sich die Experimente auf Konkatenationen aus 15 bis 23 Sätzen in Trainings- und Testingset. Die Abbildung 20 zeigt, dass für LR mit Augmentation bessere F1-Macro-Scores erreicht werden als in den entsprechenden Experimenten vom vorherigen Abschnitt 0. Für NB ist hingegen das Gegenteil der Fall. Auf weitere Experimente mit Augmentation wird jedoch verzichtet, da es zu zeit- und datenintensiv gewesen wäre, die benötigten

[13] Wie beschrieben, wurden bei SVM erst ab zehn Samples alle Kombinationen aus der Anzahl aneinandergehängter Sätze im Trainings- und Testingset ausprobiert.

Datensets zu erstellen. Es sei jedoch darauf hingewiesen, dass Augmentation potenziell bei einzelnen Classifiern zu noch besseren Resultaten führen könnte als die in Abschnitt 7.7 gesammelten besten Resultate aller durchgeführten Experimente.

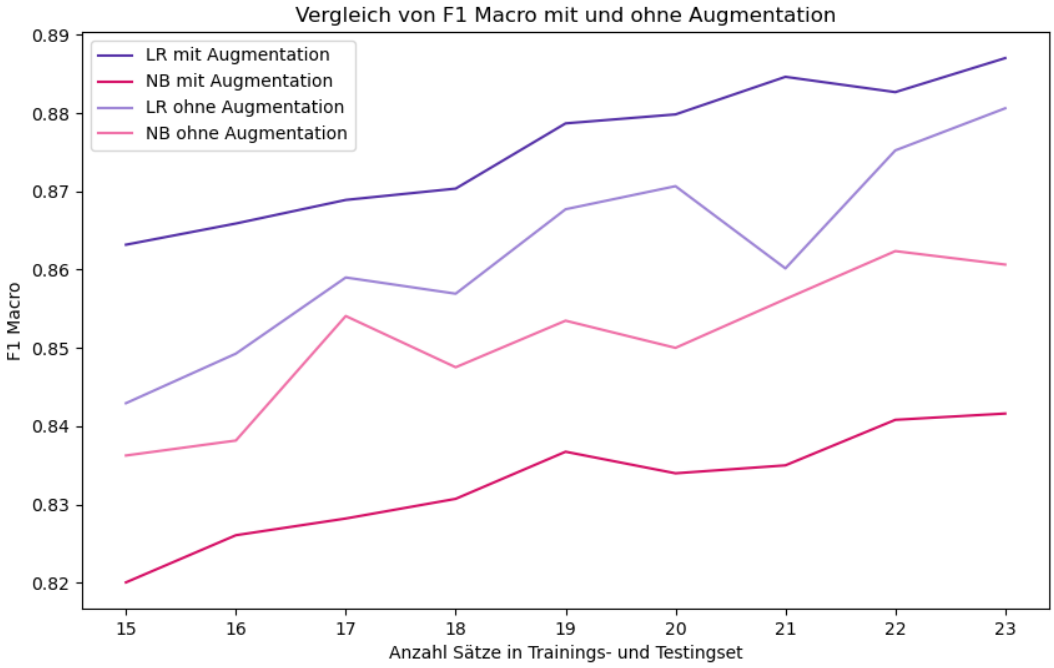

Abbildung 20: Vergleich der F1-Macro-Score mit und ohne Augmentation mit steigender Anzahl Sätze im Trainings- und Testingset

7.5 Experimentenserie 3: n-Gramme und Tf-idf

In dieser Experimentenserie wurden unterschiedliche n-Gramm-Längen mit Tf-idf-Gewichtungen kombiniert. Der erste Durchlauf wurde auf nur einer Transkription durchgeführt. Wie zu Beginn des Kapitels 7 erwähnt, lief parallel zu dieser Experimentenserie die Serie «Konkatenation von Transkriptionen», in der festgestellt wurde, dass die Classifier SVM und RF bei nur einer Transkription sehr lange Laufzeiten aufwiesen. Aufgrund dessen wurden in diesem Durchlauf nur die drei übrigen Classifier NB, LR und XGB eingesetzt.

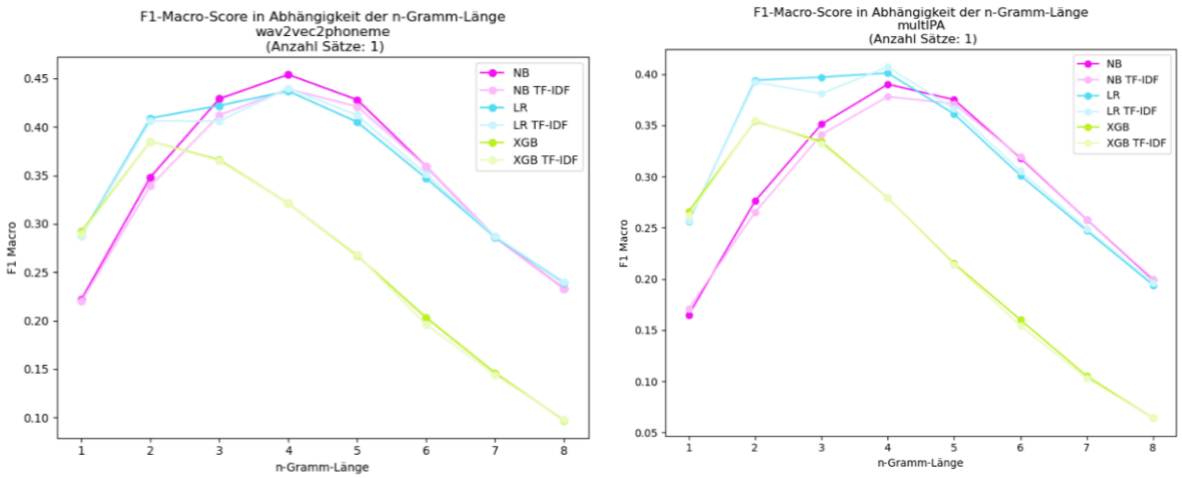

Abbildung 21: F1-Score in Abhängigkeit der n-Gramm-Länge für Wav2Vec2Phoneme (links) und MultIPA (rechts)

In der obigen Abbildung 21 sind die Resultate der drei Classifier auf einer Transkription dargestellt, einmal mit dem Phoneme-Recognizer Wav2Vec2Phoneme und einmal mit MultIPA.

In den Probeexperimenten und dem Hyperparameter-Tuning unter Abschnitt 7.2 wurde bereits festgestellt, dass sich die optimale n-Gramm-Länge zwischen zwei bis fünf bewegt. Dies ist in den obigen Abbildung 21 ebenfalls beobachtbar: XGB erreicht den besten F1-Macro-Score bereits bei einer n-Gramm-Länge von n=2 und flacht mit zunehmendem n stetig ab. NB und LR zeigen hingegen ihre Höchstleistung bei einer n-Gramm-Länge von n=4. NB erreicht von allen Classifiern den höchsten F1-Macro-Score mit 0.45 basierend auf den Wav2Vec2Phoneme-Transkriptionen und ohne Tf-idf-Gewichtung.

Unabhängig von den Phoneme-Recognizern ist für alle Classifier in den obigen Abbildung 21 ein steiler, fast doppelt so grosser Anstieg, im F1-Macro-Score von der n-Gramm-Länge n=1 auf n=2 zu verzeichnen. Dies zeigt, dass die 2-Gramme die Charakteristika der Dialekte besser erfassen.

Die Tf-idf-Gewichtung zeigt keine signifikante Wirkung auf die Ergebnisse und variiert je nach verwendetem Classifier und Phoneme-Recognizer. Für NB zeigt die Tf-idf-Gewichtung sogar eine Leistungseinbusse. An dieser Stelle sei zu erwähnen, dass Tf-idf die Annahme trifft, dass seltener vorkommende n-Gramme ausschlaggebend für die Dialekterkennung sind, das scheint hier nicht zwingend für alle seltenen Merkmale zu gelten. Hervorzuheben ist, dass die Leistungskurve mit der Tf-idf-Gewichtung tendenziell parallel zur Leistung ohne Gewichtung verläuft.

Auch Phoneme-Recognizer übergreifend sind ähnliche Leistungskurven mit einer Transkription zu beobachten, wobei der F1-Macro-Score für Wav2Vec2Phoneme um ungefähr fünf Prozentpunkte höher liegt. Dass NB besser mit Wav2Vec2Phoneme abschneidet, wurde bereits in bei der Experimentenserie zur Transkriptionskonkatenation beobachtet. Diese Beobachtung wurde auch im ersten Lauf in der der Experimentenserie 0 gemacht.

Die optimalen n-Gramm-Längen variieren für die unterschiedlichen Classifier und Phoneme-Recognizer. Klar lässt sich sagen, dass die n-Gramm-Länge von n=1 nicht für die Dialekterkennung geeignet ist.

Als Nächstes wurde mit einem Zwischenstand der Experimentenserie 0 ein zweiter Durchlauf mit aneinandergehängten Transkriptionen durchgeführt, wobei diesmal alle Classifier zum Zug kamen und je

nach Classifier und Art der Phonemtranskription (Wav2Vec2Phoneme oder MultIPA) eine unterschiedliche Anzahl von Transkriptionen konkateniert wurden. Die Anzahl konkatenierte Transkriptionen im Trainings- und Testingset war dabei gleich.

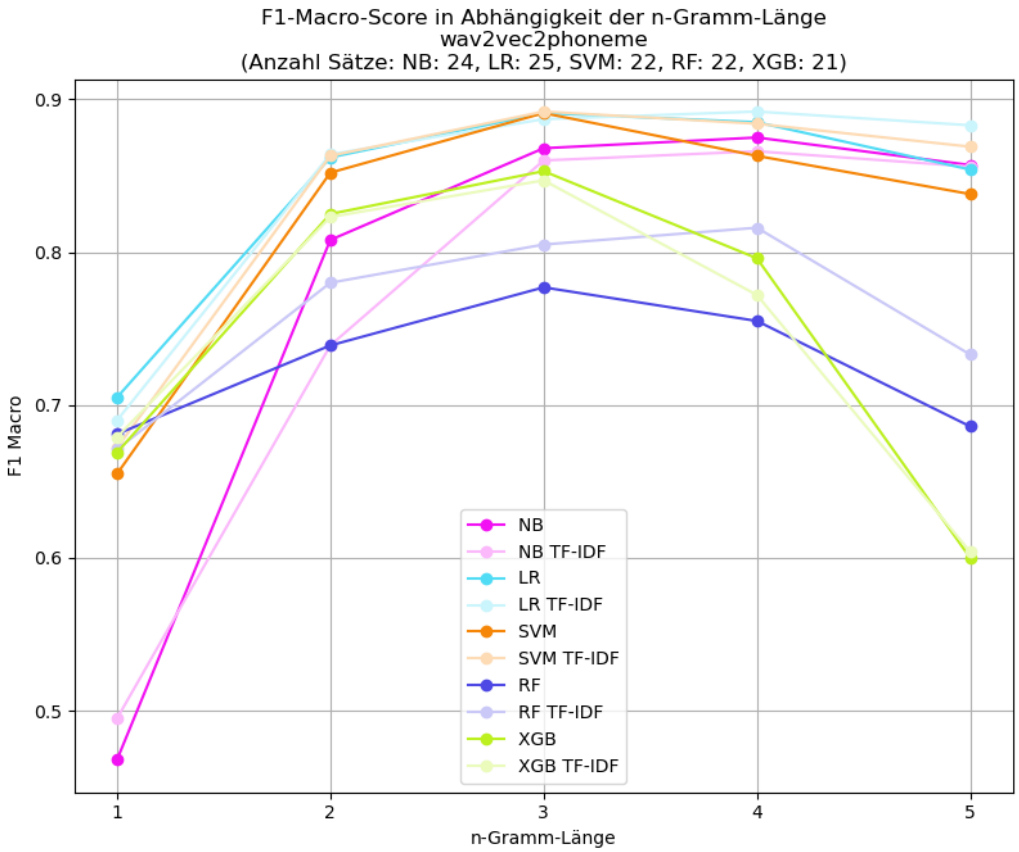

Abbildung 22: F1-Macro-Score in Abhängigkeit der n-Gramm-Länge auf konkatenierten Transkriptionen für Wav2Vec2Phoneme

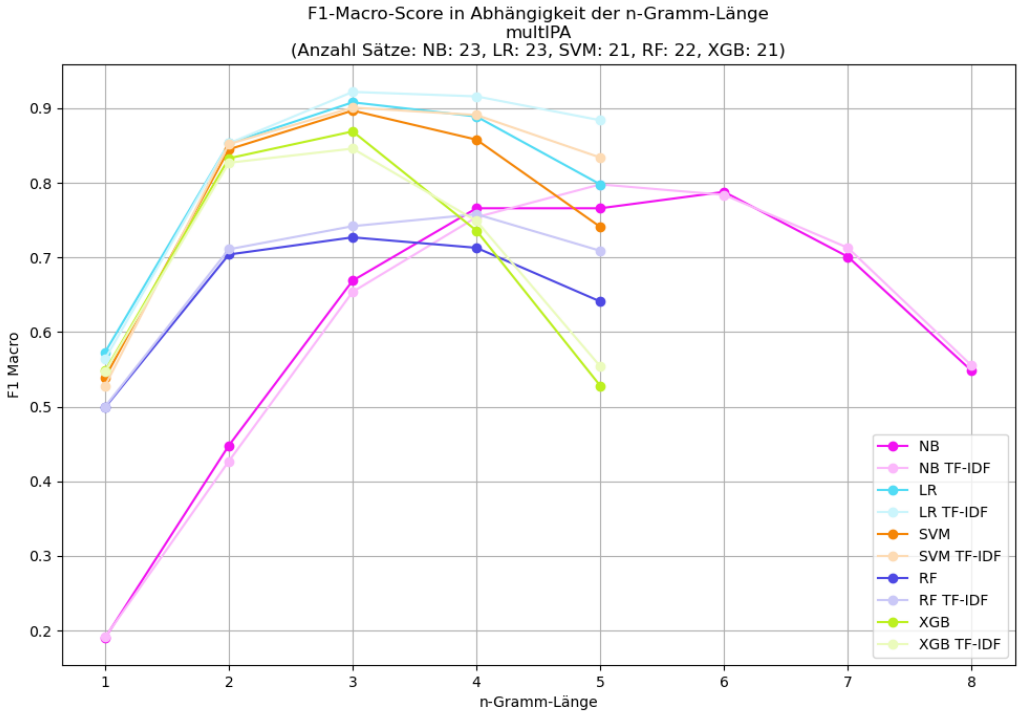

Abbildung 23: F1-Macro-Score in Abhängigkeit der n-Gramm-Länge auf konkatenierten Transkriptionen für MultIPA

Im Vergleich zu den Experimenten mit einer Transkription erzielte dieser Durchlauf mit aneinandergehängten Transkriptionen nahezu doppelt so gute F1-Macro-Scores, wie in den Abbildung 22 und Abbildung 23ersichtlich.

Die optimalen n-Gramm-Längen variieren weiterhin je nach Classifier und Phoneme-Recognizer. Für LR, SVM und XGB beträgt die optimale Länge neu n=3. Für RF erweist sich eine Länge von n=4 in Kombination mit Tf-idf-Gewichtung als optimal, während für NB die optimale Länge bei n=5 liegt, wenn MultIPA und Tf-idf verwendet wurde, und bei n=4, wenn Wav2Vec2Phoneme ohne Tf-idf verwendet wurde. Für XGB und NB hat sich somit die optimale Länge verschoben. Folglich ist die n-Gramm-Länge nicht nur vom Classifier abhängig, sondern auch von der Art der Transkription und der Anzahl Transkriptionen.

NB ist nicht länger der beste Classifier. Stattdessen erzielt LR auf MultiPA-Transkriptionen mit 23 zusammengefügten Sätzen im Trainings- und Testingset, 3-Grammen und Tf-idf-Gewichtung einen beachtlichen Score von 92 %. RF schneidet bei beiden Transkriptionen am schwächsten ab, während LR und SVM die stärksten Leistungen erbringen.

Der Einfluss der Tf-idf-Gewichtung zeigt sich bei der Verwendung von konkatenierten MultIPA-Transkriptionen eher positiv, wovon RF, SVM und LR profitieren.

Für SVM, RF und XGB zeigen die Leistungskurven bei MultIPA nach Erreichen des Höhepunkts einen steileren Leistungsabfall. Dies deutet darauf hin, dass diese Modelle empfindlicher auf die grössere Merkmalsdimensionen reagieren.

Die Art der Transkription spielt eine wesentliche Rolle in der Leistung: NB und RF erzielen bessere Leistungen mit den Transkriptionen von Wav2Vec2Phoneme, während alle übrigen Classifier mit MultIPA besser abschneiden.

Die Analyse zeigt, dass eine n-Gramm-Länge von n=1 als alleinige Merkmalsrepräsentation wenig wirksam für die Dialekterkennung ist. Im Gegensatz dazu erfasst eine minimale Länge von n= 2 signifikante Merkmale, die für die Dialektunterscheidung relevant sind. Die Längen n=3, n=4 und n=5 beinhalten relevante Merkmale, da sie je nach Kombination aus Phoneme-Recognizer und Anzahl konkatenierter Transkriptionen für bestimmte Classifier zu Bestleistungen führten. Mit zunehmender n-Gramm-Länge steigt aber auch die Anzahl der generierten n-Gramme, was zu einem Wachstum der Dimension und der Anzahl irrelevanter Merkmale führt. Dies kann sich negativ auf die Leistung der Classifier auswirken. Eine sorgfältige Auswahl der n-Gramm-Länge ist daher entscheidend für die Dialekterkennung.

Des Weiteren könnte sich eine Kombination unterschiedlicher n-Gramm-Längen sich positiv auf die Erkennung der Dialekte auswirken und die bisherige Leistung sogar verbessern. Im nächsten Experiment wurden daher selektierte Kombinationen verschiedener n-Gramm-Längen durchgeführt.

Längerfristig ist es sinnvoll, die Relevanz der n-Gramme aus linguistischer Sicht zu analysieren, um die Merkmalsauswahl zu optimieren und auch entsprechend gewichten zu können. Dadurch, dass Tf-idf wenig bis kaum einen positiven Effekt zeigen konnte, müssen es also noch andere Merkmale sein als nur die seltenen, die charakteristisch für die Erkennung sind. Dies impliziert, dass die ausschlaggebenden Merkmale für die Dialekterkennung erkannt und entsprechend gewichtet werden müssen, um die Erkennung der Dialekte durch den Fokus auf relevante Features zu verbessern.

7.6 Experimentenserie 4: n-Gramm-Mix

Diese Experimentenserie basiert auf den besten Ergebnissen der Konkatenation von Transkriptionen, bei der die Anzahl Sätze im Trainings- und im Testingset variiert wurden. Für die vier besten Classifier NB, LR, SVM und XGB wurde mit verschiedenen Kombinationen von n-Gramm-Längen experimentiert. Da die bisherige Leistung von RF, verglichen mit den anderen Classifier am schwächsten war, wurde er für dieses Experiment ausgeschlossen.

Um die Auswirkungen multipler n-Gramm-Längen zu untersuchen, wurden zur jeweiligen aktuellen n-Gramm-Länge jeweils eine oder zwei kleinere sowie eine oder zwei grössere Zahlen hinzugefügt. Mehr als drei unterschiedliche n-Gramm-Längen wurden nicht kombiniert, da eine Vergrösserung des Merkmalsraums ohne Selektion der relevanten Features sich in den bisherigen Experimenten als kontraproduktiv erwies. Alle anderen Parameter wurden während dieses Experiments konstant gehalten.

Classifier und Parameter	n-Gramm-Längen	F1-Macro-Score
NB - Sätze im Trainings-/ Testingset: 24 /24 - Wav2Vec2Phoneme	4	0.8753
	2,3	0.8298
	3,4	0.8632
	1,2,3	0.6597
	2,3,4	0.8235
	3,4,5	0.8513
LR - Tf-idf - Sätze im Trainings-/ Testingset: 8/20 - MultIPA	3	0.9241
	3,4	**0.9341**
	4,5	0.9144
	2,3,4	**0.9312**
	3,4,5	**0.9327**
	4,5,6	0.8995
SVM - Tf-idf - Sätze im Trainings-/ Testingset: 12/20 - MultIPA	3	0.9226
	2,3	0.9103
	3,4	0.9183
	1,2,3	0.8850
	2,3,4	0.9115
	3,4,5	0.9174
XGB - Sätze im Trainings-/ Testingset: 23/23 - MultIPA	3	0.8756
	2,3	0.8734
	3,4	**0.8758**
	1,2,3	0.8605
	2,3,4	0.8802
	3,4,5	0.8641

Tabelle 6: Beste Ergebnisse aus der Konkatenation von Transkriptionen und der Kombination aus mehreren n-Gramm-Längen, ohne RF

Ein Vergleich des F1-Macro-Scores mit einer n-Gramm-Länge und den Ergebnissen mehrerer n-Gramm-Längen in der Tabelle 6 zeigt, dass nur für LR die Kombination von n-Gramm-Längen eine offensichtliche Verbesserung ergab. Für XGB konnte eine minimale Verbesserung im F1-Macro-Score erzielt werden. Dies spricht für die Notwendigkeit, spezifische Merkmale auch aus linguistischer Sicht weiter zu untersuchen, um eine gezieltere Kombination der Merkmale abzuleiten und dadurch eine Verbesserung der Leistung zu erwirken.

7.7 Analyse der besten Resultate

Die besten Resultate über alle Experimente werden in Tabelle 7 **Error! Reference source not found.**und Tabelle 8 aufgelistet. Für drei von fünf Classifiern – LR, SVM und XGB – führen die MultIPA-Transkriptionen zu besseren F1-Macro-Scores als Wav2Vec2Phoneme. Der Höchstscore wird auch mit MultIPA erreicht. Bei den Classifiern ergibt sich für beide Phoneme-Recognizer fast die gleiche Rangliste. Lediglich XGB und NB tauschen die Plätze, was aber damit zusammenhängt, dass XGB mit MultIPA und NB mit Wav2Vec2Phoneme besser harmoniert. RF fällt verglichen mit den anderen Classifiern stark ab.

MultIPA (Cross-Validation)					
Classifier	**n (n-Gramm)**	**Tf-idf**	**Anzahl Sätze Trainingsset**	**Anzahl Sätze Testingset**	**F1-Macro**
LR	3,4	true	8	20	**0.9341**
SVM	3	true	12	20	0.9226
XGB	3	false	23	23	0.8756
NB	5	false	9	20	0.8379
RF	3	false	22	22	0.7266

Tabelle 7: Beste F1-Macro-Scores für MultIPA

Wav2Vec2Phoneme (Cross-Validation)					
Classifier	**n (n-Gramm)**	**Tf-idf**	**Anzahl Sätze Trainingsset**	**Anzahl Sätze Testingset**	**F1-Macro**
LR	3	false	4	20	0.9147
SVM	3	false	24	24	0.8998
NB	4	false	24	24	0.8753
XGB	3	false	25	25	0.8559
RF	3	false	22	22	0.7771

Tabelle 8: Beste F1-Macro-Scores für Wav2Vec2Phoneme

Betrachtet man die Confusion-Matrizen in Abbildung 24 zu den besten vier Classifier aus Tabelle 7 und Tabelle 8 werden Basel, Bern, Wallis und Ostschweiz am besten erkannt. Graubünden und Innerschweiz werden schlechter vorhergesagt, wobei Innerschweiz am meisten mit Zürich verwechselt wird. Umgekehrt wird Zürich bei LR und SVM schlechter identifiziert und am häufigsten mit Innerschweiz verwechselt. Dies könnte für gewisse Ähnlichkeiten zwischen diesen Dialekten sprechen. Bei Graubünden kann kein bestimmter Dialekt identifiziert werden, mit dem es besonders häufig verwechselt wurde.

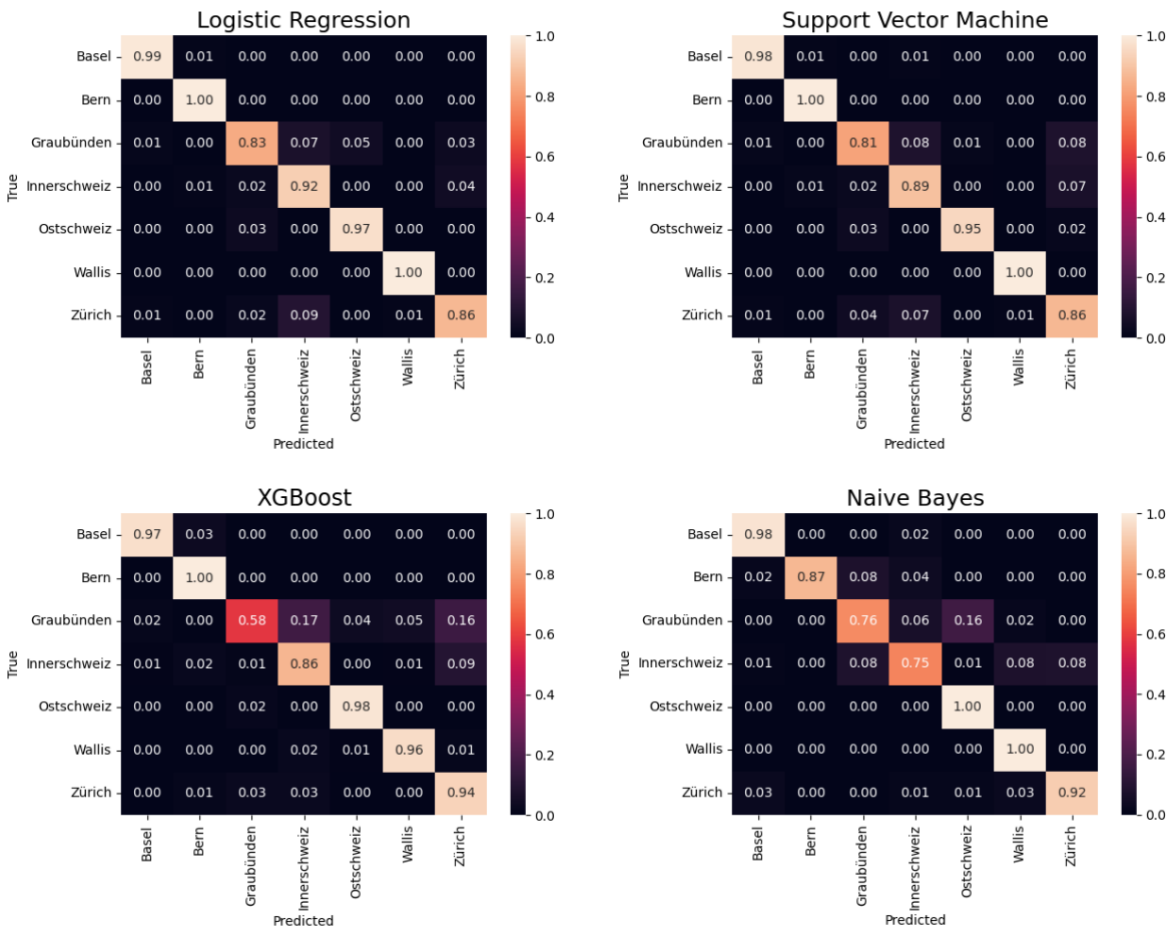

Abbildung 24: Confusion-Matrizen für die vier besten Classifier-Konfigurationen gemäss Tabelle 7 und Tabelle 8

Um diese Beobachtungen weiter zu untersuchen, lohnt sich ein Blick auf die falsch vorhergesagten Samples und deren Sprecher:innen. Mit LR, dem besten Classifier, wurden insgesamt 56 Samples von 857 im Testingset falsch identifiziert. Diese falschen Vorhersagen verteilen sich auf gerade mal 18 von 65 Sprecher:innen, also 28 %, im Testingset. Abbildung 25 zeigt links, wie viel Samples dieser 18 Sprecher:innen korrekt und falsch identifiziert wurden. Rechts ist zu sehen, welche Dialekte vorhergesagt wurden. Es ist zu erkennen, dass ein Grossteil der falschen Vorhersagen auf zwei Sprecher:innen zurückzuführen ist. Insbesondere die falschen Vorhersagen für Graubünden gehören zu einer einzigen Bündner Sprecherin mit der ID aece75d7-5d2b-47a4-9f87-24962dfd2e38. Hörproben zeigen, dass die Sprecherin Wörter verschluckt und sehr schnell spricht, aber durchaus nach Graubünden klingt. Entsprechend sind die gesprochenen Sätze in den Transkriptionen kaum wiederzuerkennen. Ein Beispiel wäre die Transkription «rafɨbrajɛxtɛnlɨtailnɛbrinbeːrndɐ̃zdedsɛbɨgeːrtiruːsɛ» für den längeren Satz «Rafael überreicht einer Teilnehmerin während eines Dates eine der begehrten Rosen». Im Beispiel sind nur wenige Wörter wie Teil (tail) oder Rose (ruːsɛ') zu erkennen. Gerade bei längeren n-Grammen, wie es hier der Fall ist, ist es naheliegend, dass verschluckte Silben und Wörter problematisch sind. Übrigbleibende Phoneme werden zu unüblichen 4-Grammen zusammengenommen, die wohl kaum dialektspezifisch sind. Die schlechte Samplequalität wirkt sich bei der Sprecher:in also klar auf die Dialektidentifizierung aus. Dass

vier andere Dialekte erkannt wurden und nicht ein oder zwei bestimmte, spricht ebenso dafür, dass es an der Audioqualität scheitert und nicht am Dialekt.

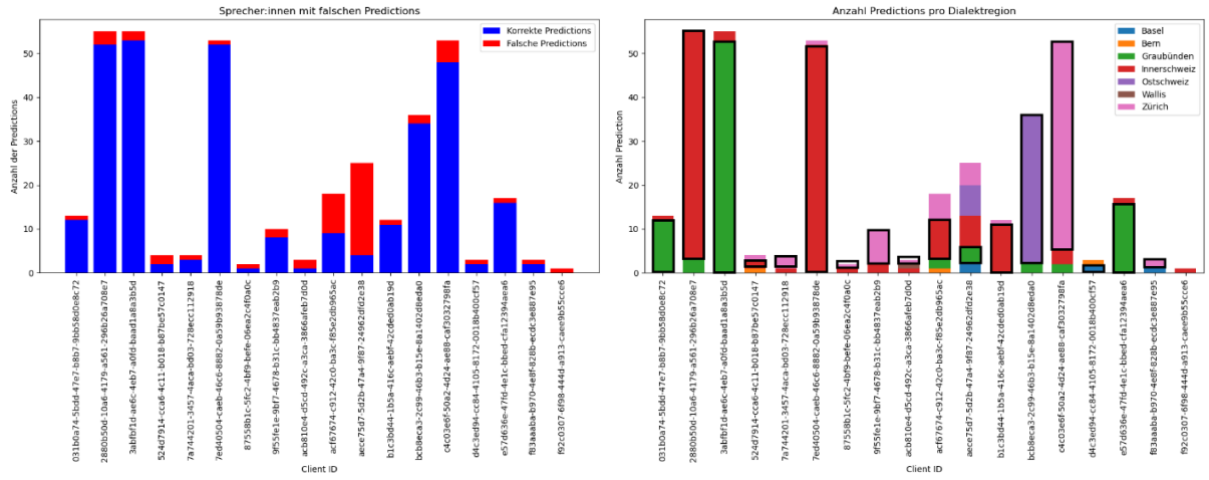

Abbildung 25: Sprecher:innen mit falschen Predictions des besten LR-Classifiers mit einem 3- und 4-Gramm-Mix, MultIPA und Tf-idf

Auch auffällig ist der Innerschweizer Sprecher mit der ID acf67674-c912-42c0-ba3c-f85e2db965ac. Gut die Hälfte seiner Samples wird falsch vorhergesagt. Stichproben zeigen, dass dies jedoch nicht an der Audioqualität liegt. Auch die Transkriptionen scheinen unauffällig. Hört man sich Samples an, die als «Zürich» vorhergesagt wurden, finden sich Beispiele darunter, die tatsächlich nach «Zürich» klingen könnte, andere aber nicht. Ob diese nach Innerschweiz klingen, konnten die Autorinnen nicht verifizieren. Es ist jedoch nicht auszuschliessen, dass der Dialekt des Sprechers tatsächlich nicht klar als Innerschweiz klassifiziert werden kann. Der Sprecher kommt aus Näfels (8752), was gemäss Abbildung 26 eher am Rand des Gebiets mit Innerschweizer Samples liegt. So könnte er potenziell einen weniger klaren Innerschweizer Dialekt sprechen. Ob dies wirklich ein Grund sein könnte, dass Samples falsch vorhergesagt werden, wie es in der vorangehenden Projektarbeit vermutet wurde [4], kann jedoch anhand nur eines Sprechers nicht festgestellt werden. Dazu müssten weitere Testdaten hinzugezogen werden und die Wahrscheinlichkeiten betrachtet werden, die die Modelle für die verschiedenen Dialekte ausgeben. Wenn dies eine Rolle spielt, scheint es bei einem von 65 Sprecher:innen doch eher eine Ausnahme zu sein.

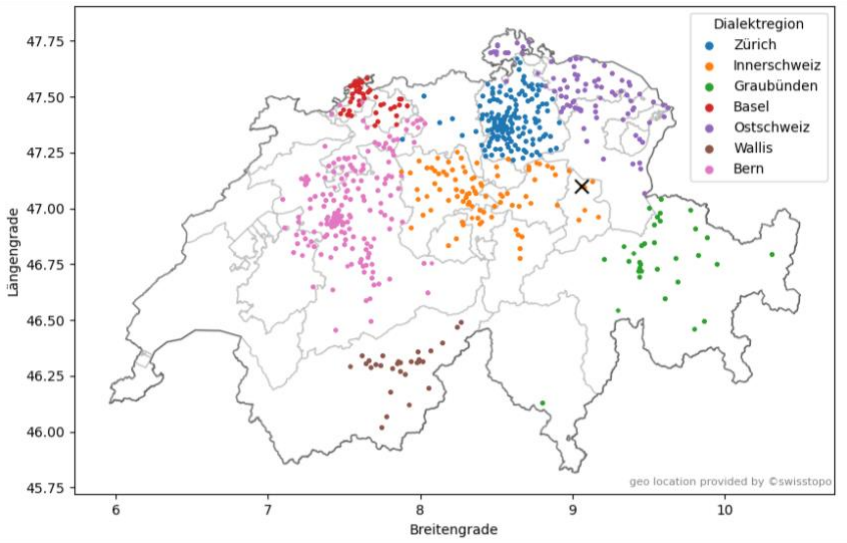

Abbildung 26: Geographische Lage der Samples der gemischten Korpora und Markierung des Innerschweizer Sprechers acf67674-c912-42c0-ba3c-f85e2db965ac

Die restlichen Sprecher:innen sind unauffällig, da deren meisten Samples korrekt klassifiziert wurden. Weitere Stichproben deuten alle auf eine zu schlechte Audioqualität hin. Die Samples weisen Störgeräusche auf oder die Sprecher:innen sprechen undeutlich. Insgesamt ist also hauptsächlich die Audioqualität ein Grund für falsche Vorhersagen. Ein Dialektemix könnte ein weiterer Grund sein, müsste aber nochmals mit mehr Testdaten überprüft werden.

7.8 Fazit

Aus den Experimenten lässt sich ableiten, dass die Verlängerungen der Samples im Testingset auf bis zu 20 Sätze einen besonders positiven Effekt auf den Macro-F1-Score haben. Auch die Samples im Trainingsset sollten auf eine mittlere Anzahl Sätze konkateniert werden. Tatsächlich ist es die Sampleverlängerung, die hauptsächlich die guten Ergebnisse ausmacht. Eine zusätzliche Augmentation könnte potenziell zu noch besseren Ergebnissen führen. Angesichts der guten Resultate ist jedoch anzuzweifeln, ob diese mit Augmentation noch übertroffen werden könnten. Für die n-Gramm-Länge hat sich je nach Classifier und Phoneme-Recognizer ein $1 < n < 6$ als signifikanter erwiesen. Tf-idf macht den kleinsten Unterschied, hat sich jedoch bei den besten beiden Modellen bewährt.

Unter den Classifiern haben sich LR und SVM durchgesetzt. XGB und NB liefern gute Ergebnisse. RF fällt ab. Bei den Phoneme-Recognizern setzt sich MultIPA durch. Dies entspricht den Erwartungen, da MultIPA, wie in Abschnitt 4.1.3 erklärt wurde, eine erweiterte Version des Wav2Vec2Phoneme ist. Allerdings schneiden NB und RF mit Wav2Vec2Phoneme besser ab. Möglicherweise sind diese Algorithmen gewissermassen mit den zusätzlichen Feinheiten, die MultIPA bietet, «überfordert».

Im Ganzen haben die Macro-F1-Scores der besten Classifier mit über 90 % die Erwartungen übertroffen. Der Ansatz überragt den bisher besten Score aus vorangehenden Arbeiten von circa 64 % deutlich [20]. Doch sind nicht nur die Ergebnisse überzeugend, sondern auch die Effizienz des Trainingsprozesses. Die Audiodaten müssen einmal zu Phonemsequenzen transkribiert werden und die Classifier können innert

weniger Minuten trainiert werden. Ganz im Kontrast zu den Pretrained-Models, die in den vorhergehenden Arbeiten mehrere Stunden trainiert werden mussten. Die klassischen Algorithmen haben einen weiteren erheblichen Vorteil gegenüber Deep-Learning-Modellen: Es kann anhand der Feature-Importance analysiert werden, welche Merkmale inwiefern für die Dialekterkennung von Bedeutung sind. (Darauf wird im nächsten Kapitel eingegangen.) Gerade dass die klassischen Algorithmen so gut funktionieren, könnte darauf hindeuten, dass durchaus einfache Zusammenhänge der Merkmale entscheidend sind für die Dialekterkennung.

8. Feature-Importance

Die sogenannte Feature-Importance ist ein Mass dafür, welche Features ein Classifier als wichtig und weniger wichtig für die Dialekterkennung einstuft. Wird die Feature-Importance analysiert, kann potenziell evaluiert werden, wie sich Dialekte unterscheiden. In dieser Arbeit sind die Features die n-Gramme. Bei all den gewählten Klassifizierungsalgorithmen kann eine Feature Importance ausgegeben werden. Hier soll nur auf den besten, aussagekräftigsten Classifier eingegangen werden, also LR mit einem 3- und 4-Gramm-Mix. Bei LR lässt sich die Feature Importance direkt aus den in Abschnitt 4.2.2 erklärten Koeffizienten ableiten. Positive Koeffizienten gewichten Merkmale respektive Features, die für einen Dialekt sprechen. Negative Koeffizienten gewichten Features, die gegen einen Dialekt sprechen. Das heisst, kommt ein bestimmtes Feature bei einem Sample häufig vor, wird der Logit bei den Klassen grösser, bei der das Feature einen positiven Koeffizienten hat, und bei den Klassen kleiner, bei denen das Feature einen negativen Koeffizienten hat. Je grösser der Betrag des Koeffizienten, desto stärker wird ein Feature gewichtet und desto stärker ein Logit beeinflusst.

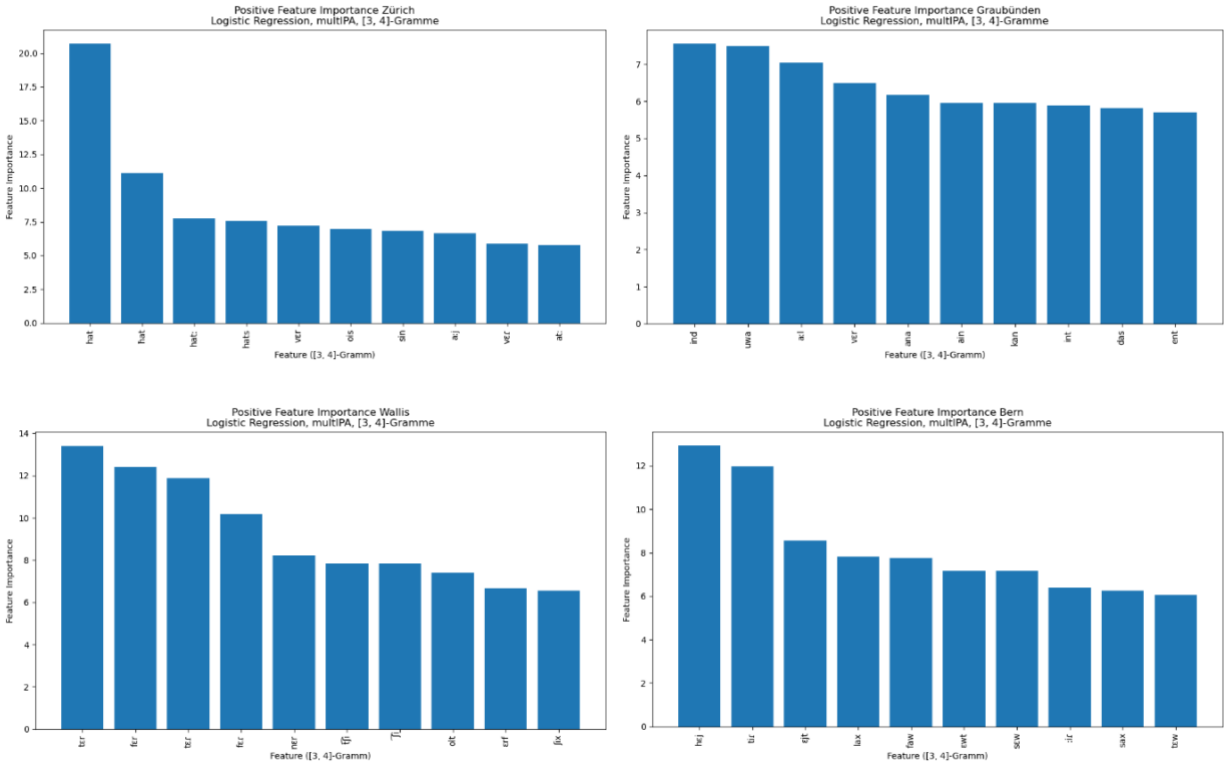

Abbildung 27: Die zehn Features mit der höchsten positiven Feature-Importance für vier beispielhafte Dialekte

scikit-learn bietet ein Attribut, aus dem die Koeffizienten und damit die Feature-Importance direkt nach dem Training des Classifiers ausgegeben werden konnten. Die Abbildung 27 stellen die wichtigsten zehn positiven Features für vier beispielhafte Dialekte dar.[14] Der Wert der Feature-Importance entspricht den Koeffizientenwerten. Nur bei Zürich hebt sich das wichtigste Feature klar von den anderen ab. Bei den restlichen Dialekten sind die Unterschiede zwischen den Koeffizienten der Features feiner abgestuft. Ein ähnliches Bild zeigt sich bei den zehn wichtigsten negativen Features in Abbildung 28. Insofern sind nicht nur wenige Features entscheidend, sondern es ist die richtige Mischung, die es ausmacht. Ausserdem ist zu beobachten, dass unter den zehn wichtigsten Features kaum 4-Gramme vorkommen, obwohl es sich um eine Mischung aus 3- und 4-Grammen handelt. Allerdings liegt der beste F1-Macro-Score von LR mit 3-Grammen von 92.19 % auch nur knapp unter den 93.41 % des n-Gramm-Mixes.

[14] Dialekte, die in der Analyse nicht dargestellt sind, sind im Anhang zu finden. Dies gilt für alle in diesem Kapitel beispielhaft ausgewählten Diagramme.

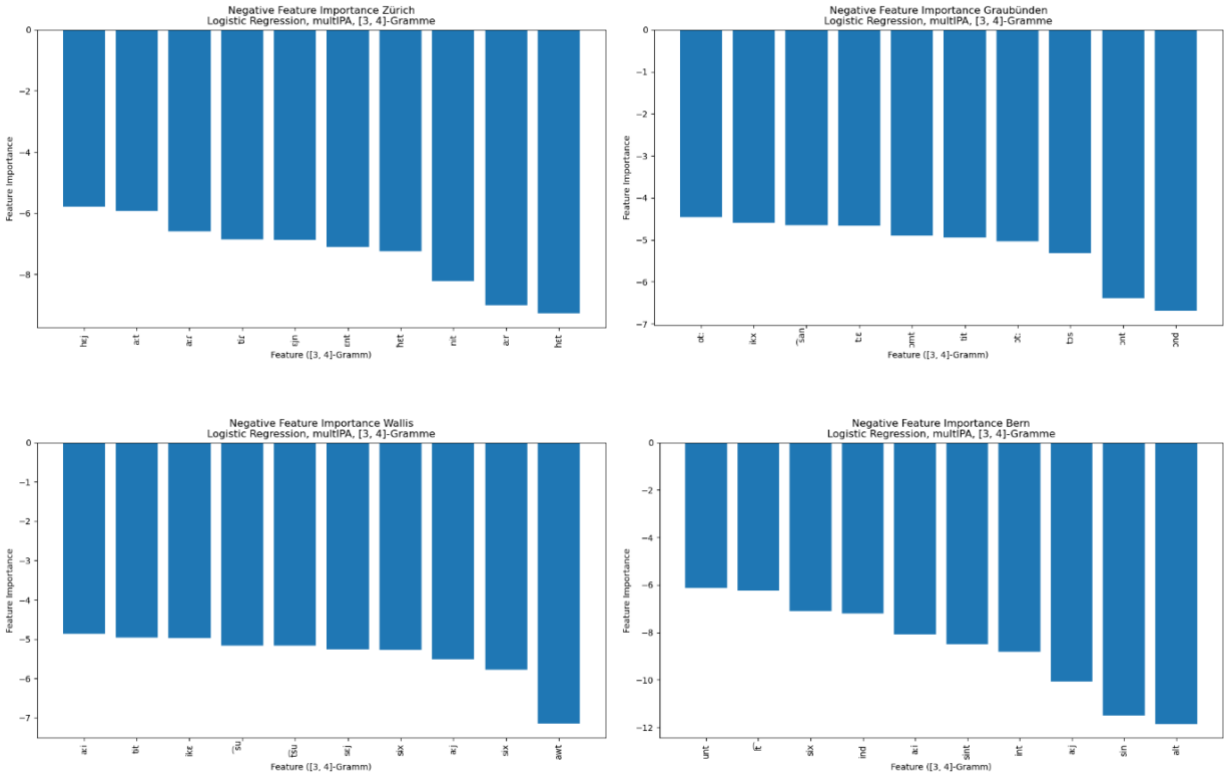

Abbildung 28: Die zehn Features mit der höchsten negativen Feature-Importance für vier beispielhafte Dialekte

Doch was macht nun ein Feature zu einem wichtigen Feature? In der Voranalyse in Abschnitt 7.1 wurde untersucht, wie häufig Phoneme vorkommen, und es wurde festgestellt, dass die Häufigkeit je nach Dialekt variiert. Inwiefern die Häufigkeit für die Dialekterkennung eine Rolle spielt, kann jetzt überprüft werden. In den Abbildung 29 wird dargestellt, wie häufig die zehn wichtigsten positiven Features vierer beispielhafter Dialekte in allen Dialekten vorkommen. Die absolute Häufigkeit der zehn wichtigsten Features variiert von wenigen 100 in Zürich bis über 12'000 in Basel. Es können also seltenere und häufigere Features wichtig sein. Bei der Mehrheit der Top-10-Features kommt das Feature jedoch im Zieldialekt häufiger vor als in den anderen Dialekten. Somit könnte dies ein wichtiges Kriterium für die Dialekterkennung sein.

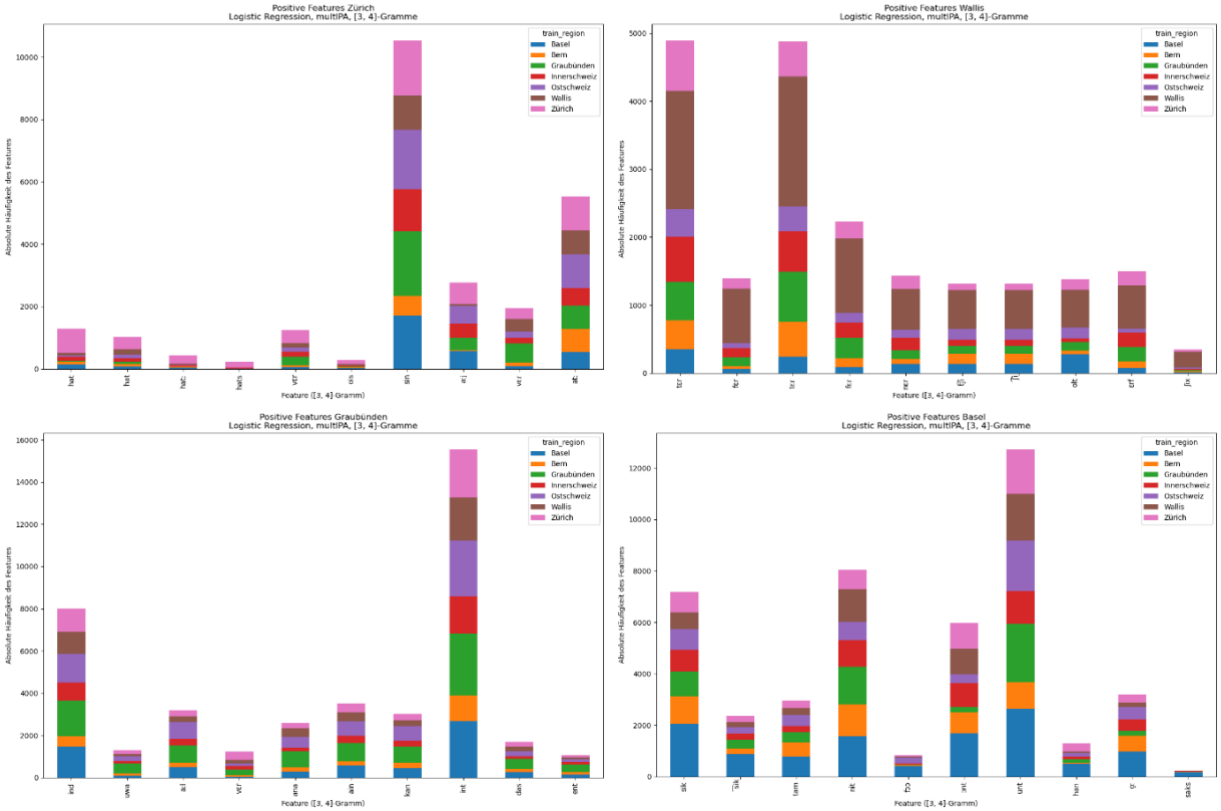

Abbildung 29: Häufigkeit der zehn wichtigsten positiven in den einzelnen Dialektregionen für vier beispielhafte Dialekte

Einzig die Innerschweiz in Abbildung 30 hat unter den zehn wichtigsten Features ungewöhnlich viele Features, die eigentlich in anderen Dialekten häufiger wären. Was der These nicht widersprechen muss, zumal Innerschweiz etwas schlechter vorhergesagt wird und zum Beispiel beim erwähnten Sprecher noch kein klarer Grund dafür evaluiert werden konnte. Es ist daher auch in Erwägung zu ziehen, dass das Modell schlechter gelernt hat, Innerschweiz zu erkennen, als andere Dialekt, und das hier sichtbar wird.

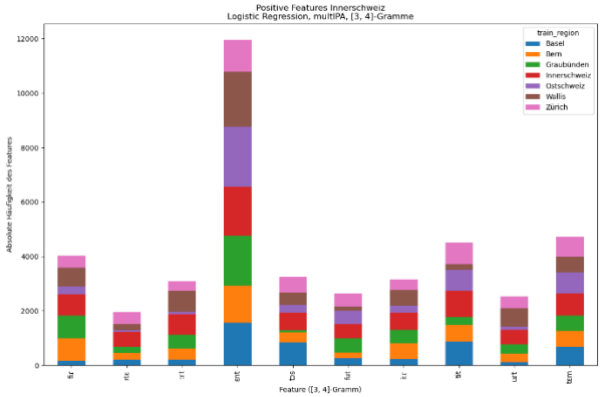

Abbildung 30: Häufigkeit der zehn wichtigsten positiven in den einzelnen Dialektregionen für Innerschweiz

Werden die wichtigsten positiven Features mit den zehn wichtigsten negativen Features in Abbildung 31 verglichen, wird die These nochmals bestätigt. Die Top-10 der negativen Features kommen im Zieldialekt seltener vor als in den anderen Dialekten und fungieren dadurch als Ausschlusskriterium.

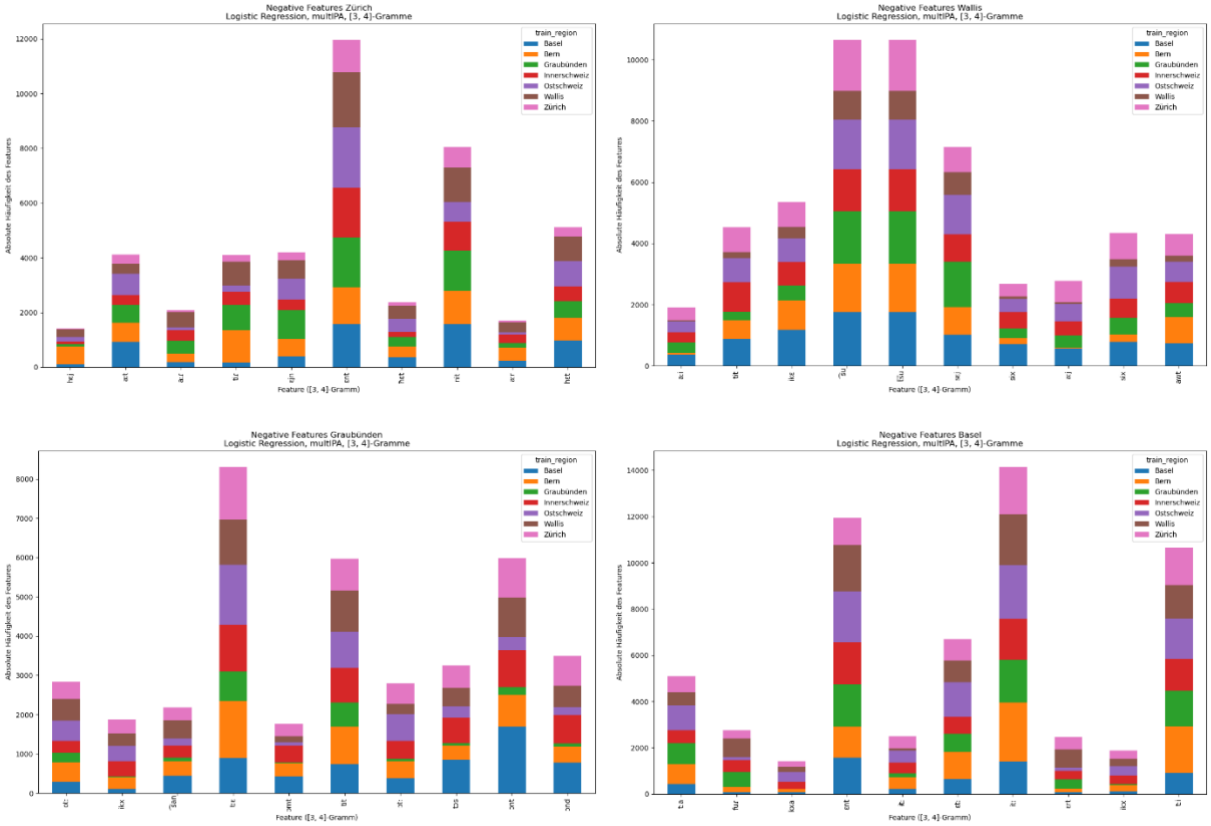

Abbildung 31: Häufigkeit der zehn wichtigsten negativen in den einzelnen Dialektregionen für vier beispielhafte Dialekte

Da Features bei den einen Dialekten unter den positiven Features und gleichzeitig bei anderen unten den negativen Features auftauchen können, wurden nochmals die Mengen der zehn wichtigsten positiven und negativen Features der Dialekte genau untereinander abgeglichen. Dies wird in Abbildung 32 in Form von Schnittmengen gezeigt. Grün sind jeweils die Dialekte, für die die Features unter den zehn wichtigsten positiven Features vorkommen, und orange sind die Dialekte, unter denen die gleichen Features zu den zehn wichtigsten negativen Features gehören. (Die restlichen Features der Mengen werden nicht dargestellt.).

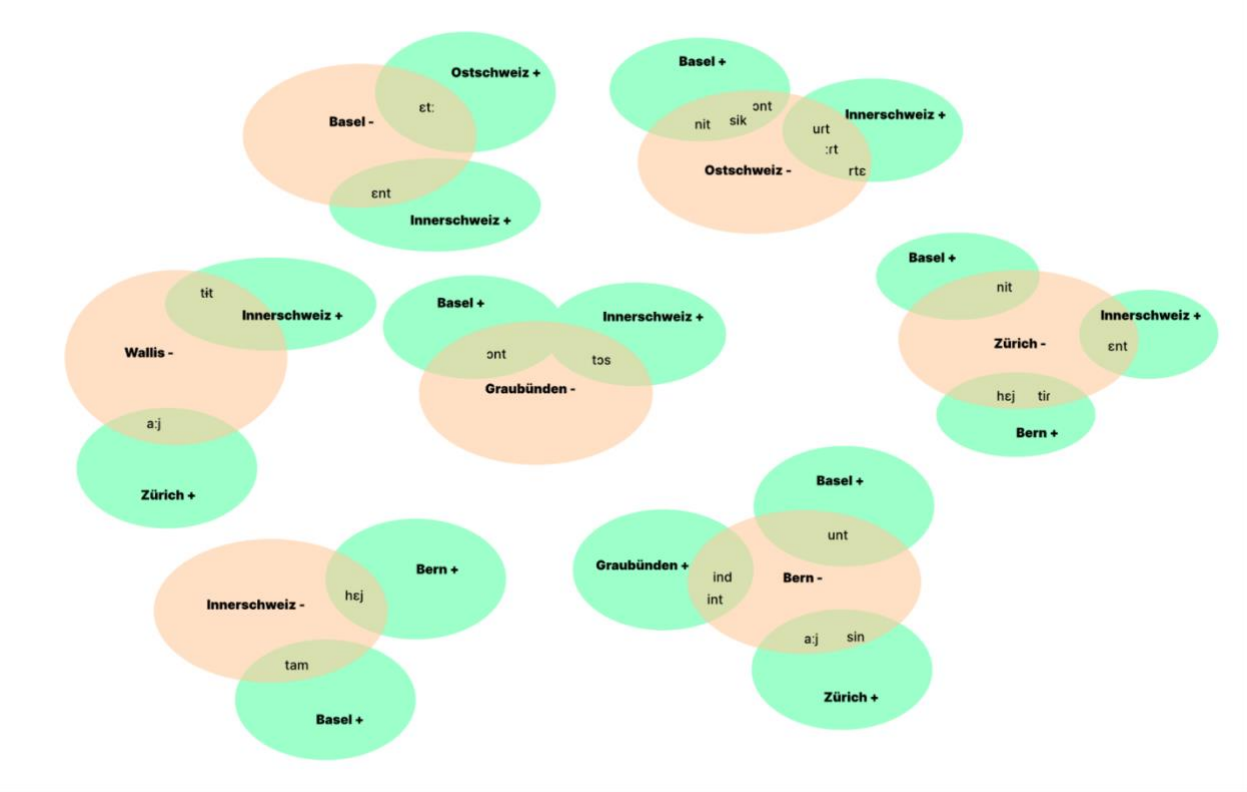

Abbildung 32: Schnittmengen für Features die bei den orange eingefärbten Dialekten zu den wichtigsten negativen Features gehören und bei den grün eingefärbten Dialekten zu den zehn wichtigsten positiven Features

Diese Darstellung deutet an, von welchen Dialekten sich ein roter Dialekt durch die Feature-Gewichtung besonders abgrenzt. Denn die Features in den Schnittmengen sind ein wichtiges Einschlusskriterium der grünen Dialekte und gleichzeitig ein wichtiges Ausschlusskriterium der orangen Dialekte. Es ist allerdings anzumerken, dass dies nur einen ersten Eindruck vermittelt. Um Zusammenhänge und Abgrenzung zwischen den Dialekten abzuleiten, wäre eine komplexere Analyse notwendig, die mehr Features einbezieht.

Zusammengefasst kann gesagt werden, dass Features, die in einem Dialekt häufiger vorkommen als in anderen, tendenziell wichtigere Einschlusskriterien sind und gleichzeitig Features, die in einem Dialekt seltener vorkommen als in anderen, tendenziell wichtigere Ausschlusskriterien sind. Einige dieser Features könnten dazu dienen, von bestimmten anderen Dialekten abzugrenzen.

9. Verifizierung der Resultate

Die in Tabelle 7 und Tabelle 8 aufgezählten Resultate sind deutlich besser als diejenigen früherer Arbeiten und Projekte, wie bereits in Abschnitt 7.8 festgestellt wurde. Damit sichergestellt werden kann, dass keine Fehler im Preprocessing und den Experimenten entstanden sind, wurden verschiedene Tests durchgeführt. Diese werden in den folgenden Abschnitten beschrieben.

9.1 Unit-Tests für die Datensplits

Indem verschiedene Aspekte bei Trainings-, Validierungs- und Testingset überprüft werden, kann gleichzeitig verifiziert werden, dass das Preprocessing wie erwartet funktioniert. Dafür wurden die in der Tabelle 9 ersichtlichen Unit-Tests durchgeführt; sowohl beim genutzten Split für das «mixed-PA»-Datenset als auch bei dessen bearbeiteten Varianten mit verlängerten Transkriptionen. In den ersten Tests werden Samples gesucht, die nicht nur in einem der drei Sets eines Splits vorkommen. Insbesondere Samples, die sowohl im Trainings- als auch im Testingset vorkommen, würden zu vermeintlich guten Testresultaten führen, da das Modell die Samples im Training gesehen hat und darum im Testing leichter identifizieren kann. Weil sprecher:innenspezifische Merkmale, wie bereits erwähnt, nicht ganz ausgeschlossen werden können, wird ebenfalls getestet, ob überlappende Sprecher:innen zwischen den drei Sets vorkommen. Denn so kann ausgeschlossen werden, dass das Modell Sprecher:innen beim Testing wiedererkennt und deshalb besser klassifiziert. In einem letzten Test wird der Vollständigkeit halber geprüft, ob alle Dialektregionen in allen Sets vorkommen. Dass der Split für das «mixed-PA»-Datenset in der Anzahl Samples ausgeglichen ist und alle Dialektregionen vorkommen, ist dank der Diagramme in Abbildung 12 in Abschnitt 6.2 eigentlich bereits ersichtlich. Sämtliche Tests wurden bestanden und folglich konnten grobe Fehler in den Daten ausgeschlossen werden.

Test	Ergebnis
Gemeinsame Samples im Trainings- und Testingset	✅
Gemeinsame Samples im Trainings- und Validierungsset	✅
Gemeinsame Samples im Testing- und Validierungsset	✅
Überlappende Sprecher:innen im Trainings- und Testingset	✅
Überlappende Sprecher:innen im Trainings- und Validierungsset	✅
Überlappende Sprecher:innen im Testing- und Validierungsset	✅
Alle Dialektregionen im Trainings-, Validierungs- und Testingset	✅

Tabelle 9: Tests von Datensplits

9.2 Tests mit anderen Datensets und Splits

Da sämtliche Experimente mit dem gleichen Datenset «mixed-PA» und dem gleichen Split durchgeführt wurden, muss untersucht werden, wie gut ein Classifier abschneidet, wenn er mit anders zusammengestellten Datensets und Splits trainiert und getestet wird. Deshalb wurden mithilfe des in Abschnitt 6.2 erklärten Verfahrens neue Datensets und Splits erstellt. In einem ersten Schritt wurden vier Verifizierungssplits mit der gleichen Anzahl Samples pro Dialektregion wie im «mixed-PA»-Datenset erzeugt. Mit jedem dieser Splits wurde der beste Classifier (LR mit verlängerten Samples, Tf-idf und 3- und 4-Grammen) nochmals trainiert. Die resultierenden F1-Macro-Scores sind in der Tabelle 10 sehen. Im Vergleich zum Score der Experimente weichen diese um bis zu 0.0756 ab. Folglich sind die Ergebnisse bis zu einem gewissen Masse abhängig davon, wie die Samples beim Splitten zufällig verteilt wurden. Einerseits spielt es eine Rolle, wie geeignet die Samples im Trainingsset für das Training sind und andererseits können die Samples im Testingset repräsentativer oder weniger repräsentativ sein. Wie in Abschnitt 7.7 zum Beispiel gezeigt wurde, können falsche Vorhersagen stark von schlechter Samplequalität dominiert werden. Sind zu viele qualitativ schlechte Samples im Testingset, fallen die Ergebnisse schlechter aus als repräsentativ wäre. Wird das Modell umgekehrt mit qualitativ schlechten oder weniger aussagekräftigen Samples trainiert, vermag beim Training nicht das volle Potenzial des Klassifizierungsalgorithmus ausgeschöpft werden. Um das vorgestellte Dialekterkennungsmodell produktiv anzuwenden, würde es sich lohnen, eine möglichst ideale Zusammensetzung des Trainingssets zu evaluieren. Für das Testing könnten zudem mehr Daten beispielsweise aus anderen Korpora hinzugezogen werden, um noch repräsentativere Testergebnisse zu erhalten.

F1-Macro (30'000 Samples pro Region)				
Split Experimente	Verifizierungs-split 1	Verifizierungs-split 2	Verifizierungs-split 3	Verifizierungs-split 4
0.9341	0.8904	0.88072	0.8585	0.9238

Tabelle 10: F1-Macro-Scores für vier neue Verifizierungssplits im Vergleich zum in den Experimenten genutzten Split

In einem zweiten Schritt wurde die Anzahl Samples pro Dialektregion variiert. Aus den Ergebnissen in Tabelle 11 lässt sich jedoch kein klarer Zusammenhang zwischen der ansteigenden Anzahl Samples und dem F1-Macro-Score feststellen, obwohl ein neues Bestergebnis von 94.07 % mit der maximal möglichen Anzahl Samples erreicht wurde. Gerade wenn mit den beschriebenen Schwankungen in Tabelle 10 zu rechnen ist, müssten weitere Experimente durchgeführt werden, um klare Aussagen treffen zu können. Doch ist zu erwägen, dass Qualität und Aussagekraft der Samples im zusammengestellten Datenset genauso entscheidend für die Ergebnisse sein können wie die Datenmenge. Für diese Arbeit zeigen die Scores mit anderen Splits nach wie vor, dass die vorgestellte Dialekterkennungsmethode gut funktioniert.

F1-Macro für Anzahl Samples pro Dialektregion					
10'000	15'000	20'000	25'000	30'000	34'404 (max)
0.8838	0.8693	0.9210	0.8461	0.9341	0.9407

Tabelle 11: F1-Macro-Scores für verschieden grosse, neue Datensets

10. Diskussion

Der beste F1-Macro-Score von 94.07 % mit Multinomial-Logistic-Regression aus Abschnitt 9.2 zeigt das grosse Potenzial des gewählten Ansatzes zur Dialekterkennung. Der Score übertrifft nicht nur die Scores der vorangehenden Projekt- und Bachelorarbeiten deutlich, sondern kommt auch an die in Kapitel 2 vorgestellten Ansätze der bisherigen Forschung heran oder übertrifft diese sogar. Ausschlaggebend für die hervorragenden Ergebnisse war nicht nur die Qualität der Phonemtranskriptionen, sondern auch die Konkatenation der Transkriptionen für das Training der Classifier. Ob der Ansatz der vorhergehenden Arbeiten über das Finetuning eines Pretrained-Models mit verlängerten Samples ähnliche Scores liefern würde, müsste reevaluiert werden.

Neben der Sampleverlängerung machten die n-Gramm-Längen den zweitgrössten Unterschied in den Scores. Zu kurze n-Gramme scheinen zu wenig aussagekräftig zu sein, zu lange hingegen zu spezifisch und selten. Damit die Modelle noch besser und schneller lernen, könnte in einem nächsten Schritt eine Feature-Selection vorgenommen werden und irrelevante n-Gramme für das Training verworfen werden.

Tf-idf hatte den geringsten Einfluss auf die Resultate. Die Gewichtung hat zwar bei Logistic-Regression und SVM die Ergebnisse nochmals verbessert, doch nur minimal. Bei den übrigen Classifiern hatte Tf-idf sogar einen negativen Effekt. Somit müssen seltenere n-Gramme, wie sie die Tf-idf-Gewichtung betont, nicht unbedingt ausschlaggebend sein. Stattdessen scheint es bei einigen Classifiern gewinnbringender zu sein, wenn diese die Gewichtung frei ohne Tf-idf lernen können.

Unter den Classifiern erreichte Multinomial-Logistic-Regression die besten F1-Macro-Scores, direkt gefolgt von SVM. Naive-Bayes und XGBoost bewegten sich im Mittelfeld, während Random-Forest am schlechtesten abschnitt. Bei den Phoneme-Recognizern gewinnt MultIPA mit seinem feineren Phonemvokabular. Allerdings kommen Naive-Bayes und Random-Forest besser mit Wav2Vec2Phoneme zurecht.

Obwohl noch nicht sicher gesagt werden kann, dass der vorgestellte Ansatz Dialekte besser erkennt als der Ansatz der vorangehenden Arbeiten, solange letzterer nicht mit Sampleverlängerung getestet wurde, besticht ersterer jedoch mit anderen Vorteilen. Zum einen ist dies die Effizienz beim Training. Es müssen lediglich einmal Transkriptionen der Audiodaten erstellt werden. Die Classifier sind auf einem durchschnittlichen persönlichen Computer in nur wenigen Minuten trainiert. Zum anderen kann bei den gewählten, einfachen Algorithmen analysiert werden, welche Features als wichtiger und weniger wichtig für bestimmte Dialekte eingestuft werden. Eine erste grobe Analyse hat gezeigt, dass n-Gramme, die in einem Dialekt deutlich häufiger vorkommen als in anderen, für diesen Dialekt sprechen und n-Gramme, die deutlich seltener bei

einem Dialekt vorkommen als bei anderen, gegen diesen Dialekt sprechen. Eine detailliertere Analyse könnte weitere Erkenntnisse bringen.

Zusammengefasst ist der Ansatz also einfach, effizient, aussagekräftig und erkennt schweizerdeutsche Dialekte sehr gut. Einzig die Samplequalität und eine ungünstige Auswahl der Trainings- und Testingsamples scheint die Dialekterkennung zu verschlechtern. Um den Ansatz produktiv nutzen zu können, würde es sich also lohnen, Samples schlechter Qualität für das Training auszusortieren. Ob der Dialekt bei einem kleinen Anteil der Sprecher:innen aus anderen Gründen schlechter erkannt wird, bleibt offen und könnte mit mehr Testdaten überprüft werden.

Danksagung

An dieser Stelle danken wir unseren Betreuenden Jasmina und Mark für die wertvolle Beratung bei der Themenfindung sowie die hilfreichen Anregungen bei der Durchführung von Experimenten. Geschätzt haben wir auch den Freiraum für eigene Ideen und Umsetzungen.

Des Weiteren danken wir der ZHAW für die Bereitstellung der notwendigen Ressourcen beziehungsweise den Zugang zum APU-Cluster. Ohne diesen wäre die Generierung von Phonem-Transkriptionen sowie die Durchführung von Experimenten nicht möglich gewesen.

Nicht zuletzt gilt unser Dank unserem engsten Umfeld, das uns stets mit viel Geduld und Vertrauen unterstützt hat.

11. Verzeichnisse

11.1 Literaturverzeichnis

[1] M. Malik, M.K. Malik, K. Mehmood und I. Makhdoom, «Automatic speech recognition: a survey», *Multimedia Tools and Applications,* Bd. 80, S. 9411–9457, 2021.

[2] A. Etman und A. A. L. Beex, «Language and Dialect Identification: A survey,» in *2015 SAI Intelligent Systems Conference (IntelliSys)*, London, 2015.

[3] A. Linke, M. Nussbaumer, P. K. Portmann, *Studienbuch Linguistik*. Tübingen: Max Niemeyer Verlag, 2004.

[4] S. Waldburger und L. Bolliger. (22 12 2023). Automatische Erkennung von Dialekten (Schweizerdeutsch und Englisch) [Online]. URL: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/studentische_arbeiten/Herbst_2023/PA23_ciel_Waldburger_Bolliger_Dialect_Recogniti on.pdf [Stand am 30.5.2024].

[5] S. Malmasi, M. Zampieri, N. Ljubešić, P. Nakov, A. Ali, J. Tiedemann, «Discriminating between Similar Languages and Arabic Dialect Identification: A Report on the Third DSL Shared Task», *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, Osaka, 2016.

[6] M. Zampieri, S. Malmasi, N. Ljubešić, P. Nakov, A. Ali, J. Tiedemann, Y. Scherrer, N. Aepli, «Findings of the VarDial Evaluation Campaign 2017», *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, 2017.

[7] Y. Scherrer, T. Samardžić, E. Glaser, «ArchiMob: Ein multidialektales Korpus schweizerdeutscher Spontansprache», *Linguistik Online,* Bd. 98, Nr. 5, S. 425-454, 2019.

[8] E. Dieth, *Schwyzertütschi Dialäktschrift*. Aarau: Sauerländer Verlag, 1986.

[9] M. Zampieri, S. Malmasi, P. Nakov, A. Ali, S. Shon, J. Glass, Y. Scherrer, T. Samardžić, N. Ljubešić, J. Tiedemann, C. van der Lee, S. Grondelaers, N. Oostdijk, D. Speelman, A. van den Bosch, R. Kumar, B. Lahiri, M. Jain, «Language Identification and Morphosyntactic Tagging: The Second VarDial Evaluation Campaign», *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, Santa Fe, New Mexico, 2018.

[10] M. Zampieri, S. Malmasi, Y. Scherrer, T. Samardžić, F. Tyers, M. Silfverberg, N. Klyueva, T. Pan, C. Huang, R. Tudor Ionescu, A. M. Butnaru, T. Jauhiainen, «A Report on the Third VarDial Evaluation Campaign», *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, Ann Arbor, 2019.

[11] G. Mingliang und X. Yuguo, «Chinese dialect identification using clustered support vector machine», *2008 International Conference on Neural Networks and Signal Processing*, Nanjing, 2008.

[12] L. Nour-Eddine und A. Abdelkader, «GMM-Based Maghreb Dialect IdentificationSystem», *Journal of Information Processing Systems,* Bd. 11, Nr. 1, S. 22-38, 2015.

[13] A. Ahmed, P. Tangri, A. Panda, D. Ramani und S. Karmakar, «VFNet: A Convolutional Architecture for Accent Classification», *2019 IEEE 16th India Council International Conference (INDICON)*, Rajkot, 2019.

[14] M. Abdelazim, W. Hussein und N. Badr, «Automatic Dialect identification of Spoken Arabic Speech using Deep Neural Networks», *International Journal of Intelligent Computing and Information Sciences,* Bd. 22, Nr. 4, S. 25-34, 2022.

[15] K. Lounnas, M. Lichouri, M. Abbas, T. Chahboub und S. Salmi, «Towards an Automatic Dialect Identification System using Algerian Youtube Videos» *Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)*, Trento, 2022.

[16] M. Alrehaili, T. Alasmari und A. Aoalshutayri, «Arabic Speech Dialect Classification using Deep Learning», *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*, Jeddah, 2023.

[17] M. A. Zissman, «Comparison of four approaches to automatic language identification of telephone speech», *IEEE Transactions on Speech and Audio Processing,* Bd. 4, Nr. 1, S. 31-44, 1996.

[18] G. Işık und H. Artuner, «Turkish Dialect Recognition Using Acoustic and Phonotactic Features in Deep Learning Architectures», *Bilişim Teknolojileri Dergisi,* Bd. 13, Nr. 3, S. 207-216, 2020.

[19] R. Imaizumi, R. Masumura, S. Shiota und H. Kiya, «End-to-end Japanese Multi-dialect Speech Recognition and Dialect Identification with Multi-task Learning», *APSIPA Transactions on Signal and Information Processing,* Bd. 11, Nr. 1, e4, S. 1-23, 2022.

[20] C. Frei, P. Schneider. (8 6 2023). Automatic Identification of Swiss German Dialects using Large Language Models [Online]. URL: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/studentische_arbeiten/Spring_2023/Spring23_BA_ciel_Dialect_Recognition_Swiss_German_Schneider_Frei.pdf [Stand: 30.5.2024].

[21] P. N. Garner, D. Imseng und T. Meyer, «Automatic speech recognition and translation of a Swiss German dialect: Walliserdeutsch», *Proceedings of Interspeech*, Singapore, 2014.

[22] A. Grichting, *Wallissertitschi Weerter*. Visp: Rotten-Verlag, Radio Rottu Oberwallis, Walliser Bote, 1998.

[23] M. Plüss, L. Neukom und M. Vogel, «Swiss Parliaments Corpus, an Automatically Aligned Swiss German Speech to Standard German Text Corpus», *Proceedings of the Swiss Text Analytics Conference 2021*, Winterthur, 2021.

[24] P. Dogan-Schönberger, J. Mäder und T. Hofmann. (9 6 2021). SwissDial: Parallel Multidialectal Corpus of Spoken Swiss German [Online]. URL: https://arxiv.org/abs/2010.02810 [Stand: 30.5.2024].

[25] M. Plüss, M. Hürlimann, M. Cuny, A. Stöckli, N. Kapotis, J. Hartmann, M. A. Ulasik, C. Scheller, Y. Schraner, A. Jain, J. Deriu, M. Cieliebak und M. Vogel, «SDS-200: A Swiss German Speech to Standard German Text Corpus», *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, 2022.

[26] M. Plüss, J. Deriu, Y. Schraner, C. Paonessa, J. Hartmann, L. Schmidt, C. Scheller, M. Hürlimann, T. Samardžić, M. Vogel und M. Cieliebak, «STT4SG-350: A Speech Corpus for All Swiss German Dialect Regions», *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Toronto, 2023.

[27] Duden. (2024). Phonem. Duden [Online]. URL: https://www.duden.de/rechtschreibung/Phonem [Stand am 27 Mai 2024].

[28] D. Jurafsky und J. H. Martin. (3.2.2024). Speech and Language Processing. Phonetics [Online]. URL: https://web.stanford.edu/~jurafsky/slp3/H.pdf [Stand: 1.6. 2024].

[29] Kim. (11.5.2024). r-Laute: geRollt, geschnaRRt, vokalisieRt [Online]. URL: https://radic.es/2017/05/11/gerollt-geschnarrt-vokalisiert/ [Stand 28.5. 2024].

[30] C. Sappok und D. Nerius. (2000). Die Beziehung von graphischer und phonologischer Ebene [Online]. URL: https://idsl2.phil-fak.uni-koeln.de/sites/idslII/Lehrende/Sappok/nerius-2000-2-a.pdf [Stand 25.5. 2024].

[31] Q. Xu, A. Baevski, M. Auli, «Simple and Effective Zero-shot Cross-lingual Phoneme Recognition», *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2022, S. 2113-2117.

[32] C. Taguchi, Y. Sakai, P. Haghani, D. Chiang, «MultIPA: Universal Automatic Phonetic Transcription into the International Phonetic Alphabet», *Universal Automatic Phonetic Transcription into the International Phonetic Alphabet. Proc. INTERSPEECH 2023. 2548-2552.*

[33] C. Apavou. (2024). Wav2Vec2-base-960h-phoneme-reco-dutch: A Model for Phoneme Recognition in Dutch, Hugging Face [Online]. URL: https://huggingface.co/Clementapa/wav2vec2-base-960h-phoneme-reco-dutch [Stand: 25.5.2024].

[34] A. Baevski, H. Zhou, A. Mohamed, M. Auli, «wav2vec 2.0: A framework for self-supervised learning of speech representations», *Advances in neural information processing systems 33*, 2020.

[35] J. Boigne. (30.9.2021). An Illustrated Tour of Wav2vec 2.0 [Online]. URL: https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html [Stand: 25.5.2024].

[36] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, M. Auli, «Unsupervised Cross-lingual Representation Learning for Speech Recognition», *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Brno, Czechia, 2021.

[37] Hugging Face. (2024). Hugging Face. CTC architectures [Online]. URL: https://huggingface.co/learn/audio-course/chapter3/ctc [Stand: 25.5.2024].

[38] R. H. Dunn. (4.5.2024). eSpeak NG Text-to-Speech [Online]. URL: https://github.com/espeak-ng/espeak-ng [Stand: 24.5.2024].

[39] A. V. Kleist. (3.10.2021). Phonetisaurus: Open Source Grapheme-to-Phoneme (G2P) Conversion Toolkit [Online]. URL: https://github.com/AdolfVonKleist/Phonetisaurus [Stand: 24.5. 2024].

[40] Facebook AI. (10.11.2021). Wav2Vec2-XLSR-53-espeak-cv-ft: A Model for Phoneme Recognition, Hugging Face [Online]. URL: https://huggingface.co/facebook/wav2vec2-xlsr-53-espeak-cv-ft [Stand: 25.5. 2024].

[41] C. Taguchi. (24.8.2024). multipa [Online]. URL: https://github.com/ctaguchi/multipa [Stand: 25.5. 2024].

[42] H. Nelson, *Essential Math for AI*. Sebastopol: O'Reilly Media, 2023.

[43] GeeksforGeeks. (28.1.2024). Multinomial Naive Bayesv[Online]. URL: https://www.geeksforgeeks.org/multinomial-naive-bayes/ [Stand 28.5. 2024].

[44] K. Chen. (24.9.2021). Softmax Regression [Online]. URL: https://kinder-chen.medium.com/softmax-regression-10933c8429f [Zugriff am 27 Mai 2024].

[45] J. Brownlee. (1.9.2020). Multinomial Logistic Regression With Python [Online]. URL: https://machinelearningmastery.com/multinomial-logistic-regression-with-python/ [Stand: 20.5.2024].

[46] S. Raschka. (25.2.2021). L8.6 Multinomial Logistic Regression / Softmax Regression[Online]. URL: https://www.youtube.com/watch?v=L0FU8NFpx4E [Stand: 25.5.2024].

[47] Y. Abu-Mostafa. (17.5.2024). Lecture 14 - Support Vector Machines [Online]. Available: https://www.youtube.com/watch?v=eHsErlPJWUU [Stand: 25.5.2024].

[48] P. Winston. (10.2.2014). 16. Learning: Support Vector Machines [Online]. URL: https://www.youtube.com/watch?v=_PwhiWxHK8o [Stand: 26.5.2024].

[49] M. Cieliebak. (2022). Vorlesung Machine Learning and Data Mining Lektion 06: Learning curve analysis data collection SVM's [PowerPoint]. Zürich: ZHAW.

[50] TileStats. (15.8.2022). Support Vector Machines (SVM) - the basics | simply explained [Online]. URL: https://www.youtube.com/watch?v=gUzEN2TxnxE [Stand 25.5.2024].

[51] J. Brownlee. (27.4.2021). Machine Learning Mastery. One-vs-Rest and One-vs-One for Multi-Class Classification [Online]. URL: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/ [Stand: 30.5.2024].

[52] R. Hänsch, O. Hellwich, *Random Forests* (Handbuch der Geodäsie). Heidelberg: SpringerSpektrum, 2015, S. 5-32 [Online]. URL: https://doi.org/10.1007/978-3-662-46900-2_46-1 [Stand: 30.5.2024].

[53] IBM. (o. A.).Was ist ein Entscheidungsbaum? [Online]. URL: https://www.ibm.com/de-de/topics/decision-trees [Stand: 25.5.2024].

[54] R. Vamsi. (14.7.2020). Decision Tree – Implementation From Scratch in Python [Online]. URL: https://medium.com/swlh/decision-tree-implementation-from-scratch-in-python-1cff4c00c71f [Stand: 20.5.2024].

[55] D. Kumar. (o. A.). CS460 Lecture note Decision Tree (Gini Impurity) [Online]. URL: https://www.niser.ac.in/~smishra/teach/cs460/2020/lectures/lec11/ [Stand: 22.5.2024].

[56] T. D. (20.11.2021). An Introduction to Classification And Regression Trees (CART) in Python Part I: Understanding the Classification and Regression Trees (CART) algorithm in Python [Online]. URL: https://tenisha-d.medium.com/cart-classification-and-regression-trees-part-i-20edc81f8296 [Stand: 20.5.2024].

[57] G. James, D. Witten, T. Hastie, R. Tibshirani. (2013). An Introduction to Statistical Learning with Applications in R [Online]. URL: https://www.stat.berkeley.edu/~rabbee/s154/ISLR_First_Printing.pdf [Stand: 26.5.2024].

[58] StudySmarter. (o. A.). Random Forests [Online]. URL: https://www.studysmarter.de/studium/informatik-studium/kuenstliche-intelligenz-studium/random-forests/ [Stand: 20.5.2024].

[59] GeeksforGeeks. (22.2.2024). Random Forest Algorithm in Machine Learning [Online]. URL: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/ [Stand: 27.5.2024].

[60] ritvikmath. (7.4.2016). XGBoost: A Scalable Tree Boosting System [Online]. URL: https://www.youtube.com/watch?v=en2bmeB4QUo [Stand: 25.5.2024].

[61] xgboost developers. (2022). Introduction to Boosted Trees. XGboost Documentation [Online]. URL: https://xgboost.readthedocs.io/en/stable/tutorials/model.html [Stand: 25.5.2024].

[62] R. Guo, Z. Zhao, T. Wang, G. Liu, J. Zhao, D. Gao «Degradation state recognition of piston pump based on ICEEMDAN and XGBoost», *Applied Sciences*, Bd. 10, Nr. 6593, Sep. 2020.

[63] Z. Zhang. (9.11.2018). Understanding Gradient Boosting Tree for Binary Classification [Online]. URL: https://zpz.github.io/blog/gradient-boosting-tree-for-binary-classification/ [Stand: 25.5.2024].

[64] T. Masiu. (7.2.2022). All You Need to Know about Gradient Boosting Algorithm − Part 2. Classification [Online]. URL: https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e [Stand: 20.5.2024].

[65] K. Chouhbi. (26.12.2019). 2. Getting Started With XGBoost [Online]. URL: https://medium.com/xgboost-all-you-need-to-know/2-getting-started-with-xgboost-ebf67b86136 [Stand: 6.6.2024].

[66] G. Tseng. (13.4.2018). Gradient Boosting and XGBoost. Medium [Online]. URL: https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5 [Stand: 25.5.2024].

[67] C. Tianqi, C. Guestrin, «XGBoost: A Scalable Tree Boosting System», in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016.

[68] D. Jurafsky und J. H. Martin, Speech and Language Processing. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Standford: Stanford University, 2024, S. 32-81.

[69] S. Vajjala, B. Majumder, A. Gupta und H. Surana, *Practical Natural Language Processing. Sebastopol*: O'Reilly Media, Inc., 2020.

[70] E. W. Weisstein. (o. A.). L^2-Norm [Online]. URL: https://mathworld.wolfram.com/L2-Norm.html [Stand: 20.5.2024].

[71] D. Wei. (5.2.2024). Demystifying Data Normalization in Machine Learning [Online]. URL: https://medium.com/@weidagang/demystifying-machine-learning-normalization-0cdb8b281234 [Stand: 26.5.2024].

[72] M. Braschler. Information Engineering I, Thema: «2: Theorie: Indexierung/Vergleich.» ZHAW School of Engineering, Zürich, 2023.

[73] F. Karabiber. (o. A.). TF-IDF – Term Frequency-Inverse Document Frequency [Online]. URL: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ [Stand: 20.5.2024].

[74] F. Chiusano. (20.1.2022). Two minutes NLP – Learn TF-IDF with easy examples Term Frequency, Inverse Document Frequency, and Information Retrieval [Online]. URL: https://medium.com/nlplanet/two-minutes-nlp-learn-tf-idf-with-easy-examples-7c15957b4cb3 [Stand: 20.5.2024].

[75] scikit-learn developers. (o. A.). TfidfTransformer [Online]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html [Stand: 20.5.2024].

[76] scikit-learn developers. (o. A.). Accuracy score [Online]. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score [Stand am 23.5.2024].

[77] P. Huilgol. (24.8.2019). Accuracy vs. F1-Score [Online]. URL: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2 [Stand: 24.5.2024].

[78] scikit-learn developers. (o. A.). 3.1. Cross-validation: evaluating estimator performance [Online]. URL: https://scikit-learn.org/stable/modules/cross_validation.html [Stand: 24.5.2024].

[79] A. Jha und V. Lyashenko. (12.4.2024). MLOps Blog Cross-Validation in Machine Learning: How to Do It Right [Online]. URL: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right [Stand: 30.5.2024].

[80] J. Brownlee. (4.10.2023). A Gentle Introduction to k-fold Cross-Validation [Online]. URL: https://machinelearningmastery.com/k-fold-cross-validation/ [Stand: 23.5.2024].

[81] Python Software Foundation. (2024). python [Online]. URL: https://www.python.org/ [Stand: 30.5.2024].

[82] NumFOCUS Inc. (2024). pandas [Online]. URL: https://pandas.pydata.org/ [Stand: 30.5.2024].

[83] NumPy team. (2024). NumPy [Online]. URL: https://numpy.org/ [Stand: 30.5.2024].

[84] scikit-learn developers. (2024). scikit-learn [Online]. URL: https://scikit-learn.org/ [Stand: 30.5.2024].

[85] Matplotlib development team. (2024). matplotlib [Online]. URL: https://matplotlib.org/ [Stand: 30.5.2024].

[86] Weights & Biases. (2024). Weights & Biases [Online]. URL: https://wandb.ai/ [Stand: 30.5.2024].

[87] Docker Inc. (2024). Docker [Online]. URL: https://www.docker.com/ [Stand 30.5.2024].

[88] The HDF Group. (2006). HDF5® [Online]. URL: https://www.hdfgroup.org/solutions/hdf5/ [Stand: 30.5.2024].

[89] Hugging Face. (2024). Hugging Face [Online]. URL: https://huggingface.co/ [Stand: 30.5.2024].

[90] Patrick von Planten. (10.11.2021). Wav2vec2Phoneme Vocab [Online]. URL: https://huggingface.co/facebook/wav2vec2-xlsr-53-espeak-cv-ft/blob/main/vocab.json [Stand: 28.5.2024].

[91] C. Taguchi. (16.1.2023). MultIPA Vocab [Online]. URL: https://huggingface.co/ctaguchi/wav2vec2-large-xlsr-japlmthufielta-ipa1000-ns/blob/main/vocab.json [Stand: 28.5.2024].

[92] C. Apavou. (11.10.2022). wav2vec2-base-960h-phoneme-reco-dutch Vocab [Online]. URL: https://huggingface.co/Clementapa/wav2vec2-base-960h-phoneme-reco-dutch/blob/main/vocab.json [Stand: 28.5.2024].

[93] S. Moran. (2019). PHOIBLE 2.0. Swiss German sound inventory (UZ 2185) [Online]. URL: https://phoible.org/inventories/view/2185 [Stand: 25.5.2024].

11.2 Abbildungsverzeichnis

11.3 Tabellenverzeichnis

11.4 Formelverzeichnis

12. Anhang

12.1 Quellcode und technische Dokumentation

Der Quellcode und die technische Dokumentation sind mit den nötigen Zugriffberechtigungen unter folgendem Link abrufbar:

https://github.zhaw.ch/waldbsaf/DialectRecognitionBA

12.2 Experimentenübersicht

Eine Übersicht über sämtliche Experimente ist mit den nötigen Zugriffberechtigungen unter folgendem Link abrufbar:
https://github.zhaw.ch/waldbsaf/DialectRecognitionBA/blob/main/uebersicht_alle_experimente.xlsx

12.3 Ergänzende Abbildungen

12.3.1 Voranalyse Verteilungen Phoneme

Die Abbildung 33 - Abbildung 35 zeigen die Verteilung der Phonemhäufigkeiten über zwei Dialektregionshälften für die nicht gezeigte Dialektregionen in Abschnitt 7.1.4.

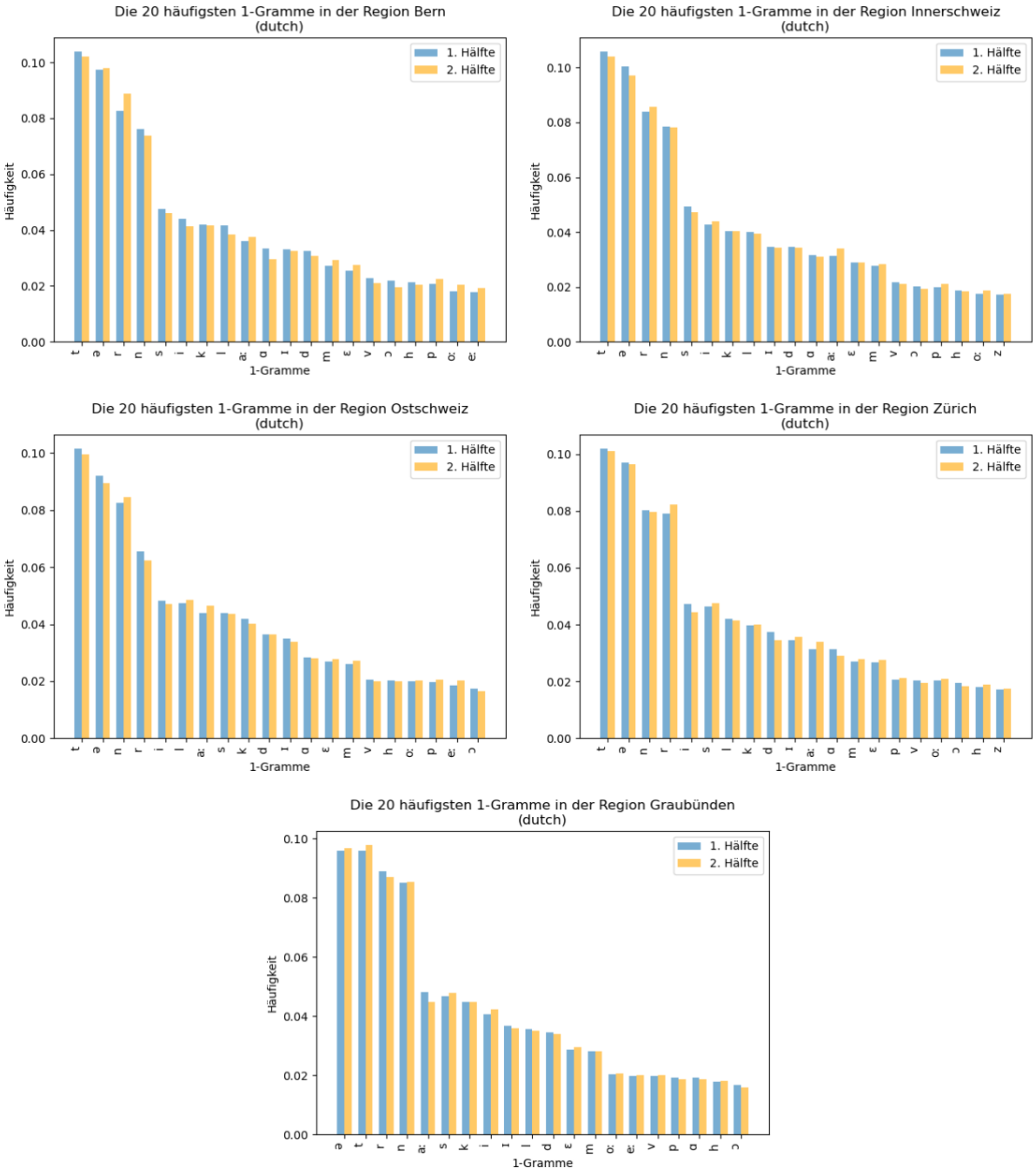

Abbildung 33: Vergleich der relativen Häufigkeiten von Dutch generierten Phonemen über zwei Hälften der Dialektregionen Bern, Innerschweiz, Ostschweiz, Zürich und Graubünden

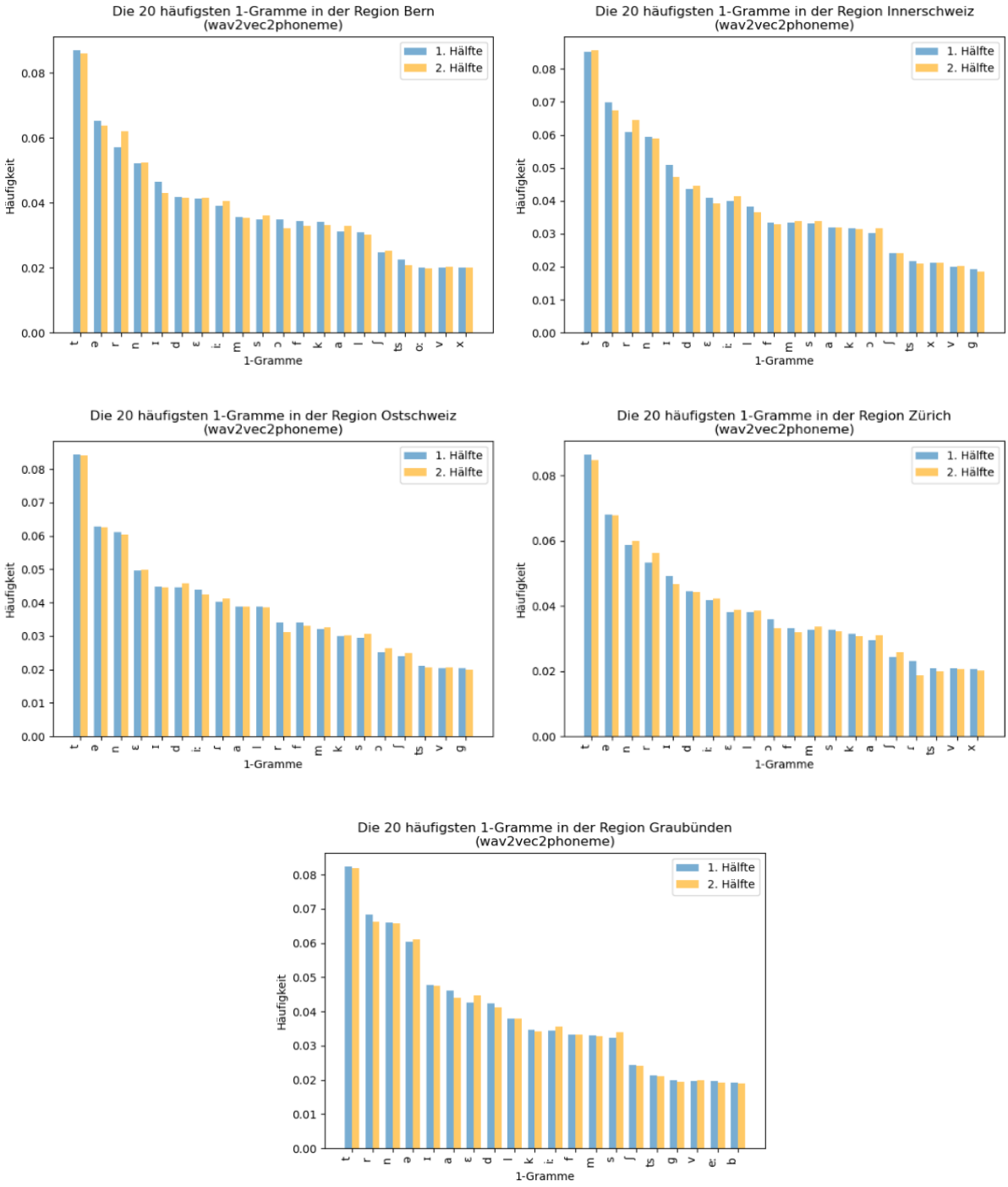

Abbildung 34: Vergleich der relativen Häufigkeiten von Wav2Vec2Phoneme generierten Phonemen über zwei Hälften der Dialektregionen Bern, Innerschweiz, Ostschweiz, Zürich und Graubünden

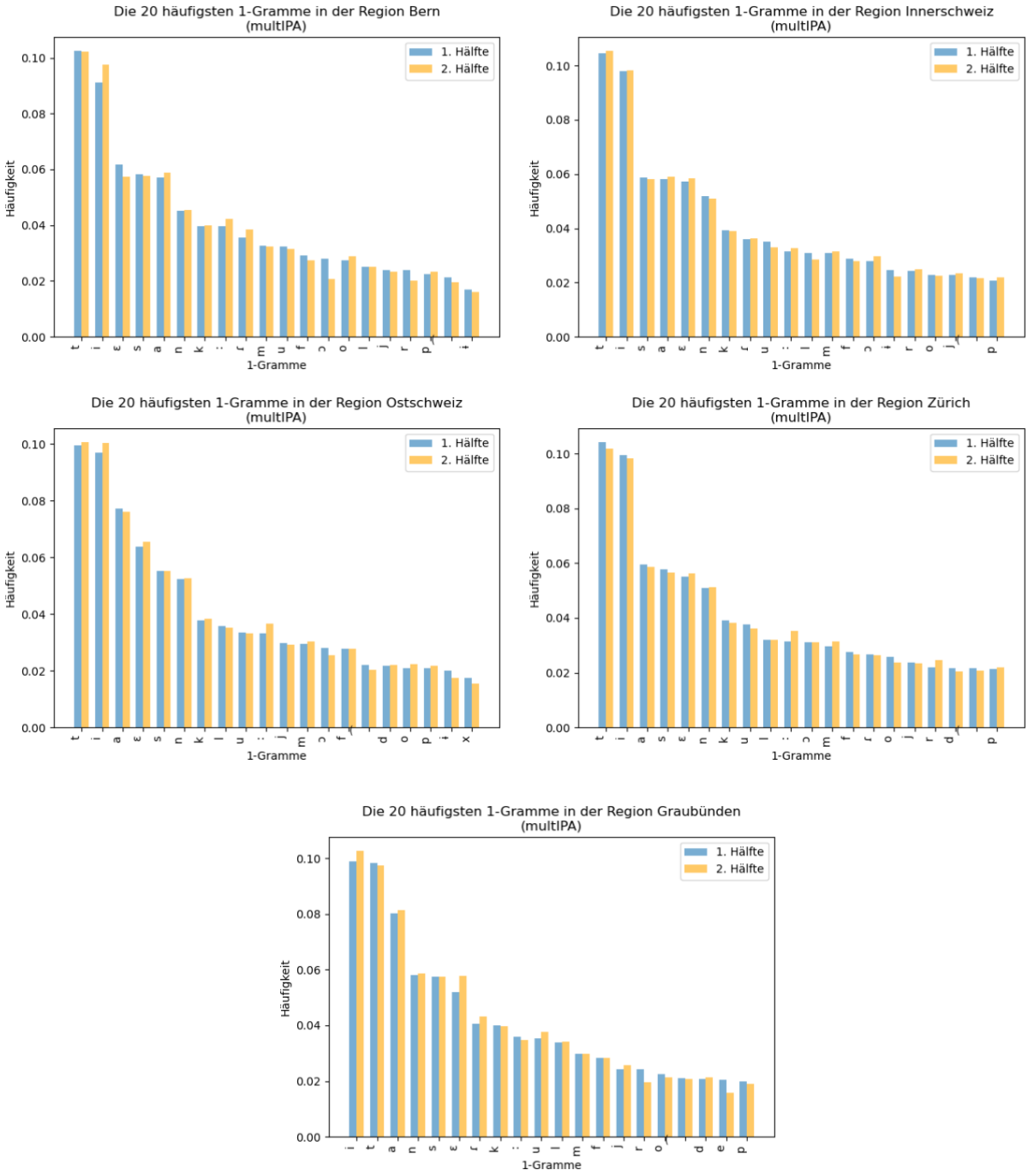

Abbildung 35: Vergleich der relativen Häufigkeiten von MultIPA generierten Phonemen über zwei Hälften der Dialektregionen Bern, Innerschweiz, Ostschweiz, Zürich und Graubünden

12.3.2 Hyperparameter-Tuning

In den Abbildung 36 - Abbildung 40 wird dargestellt, welche Hyperparameter beim Tuning getestet wurden und welche Validation-Accuracy beziehungsweise Average-Validation-Accuracy die Kombinationen dieser Hyperparameter ergeben haben.

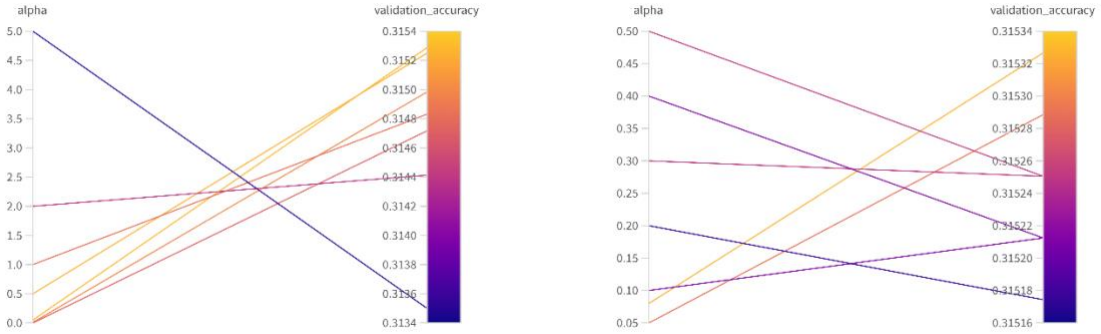

Abbildung 36: Getestete Werte und resultierende Validation-Accuracy für das Hyperparamater-Tuning des NB-Classifiers

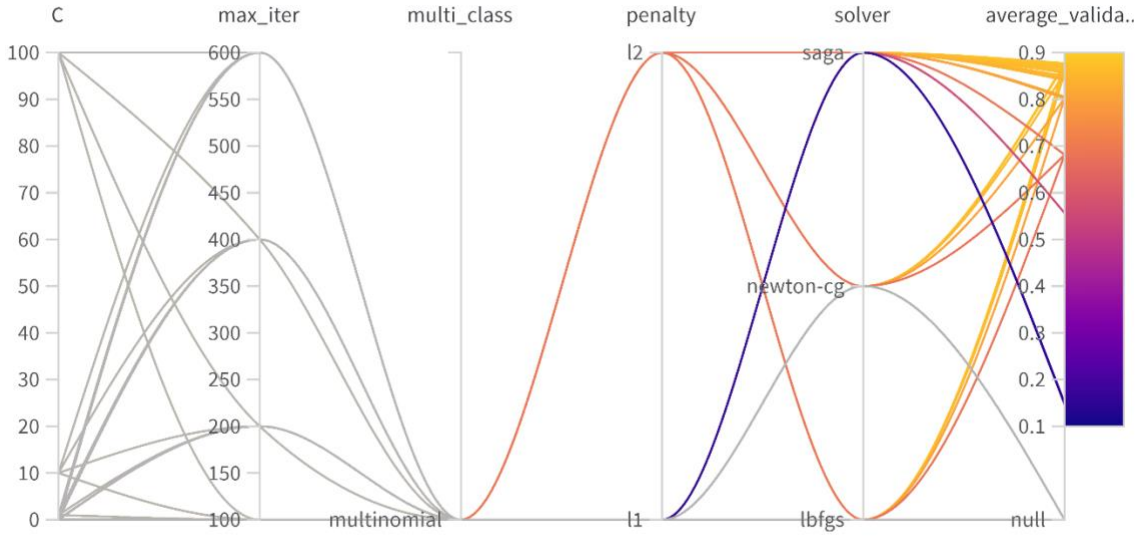

Abbildung 37: Getestete Werte und resultierende Average-Validation-Accuracy für das Hyperparamater-Tuning des LR-Classifiers

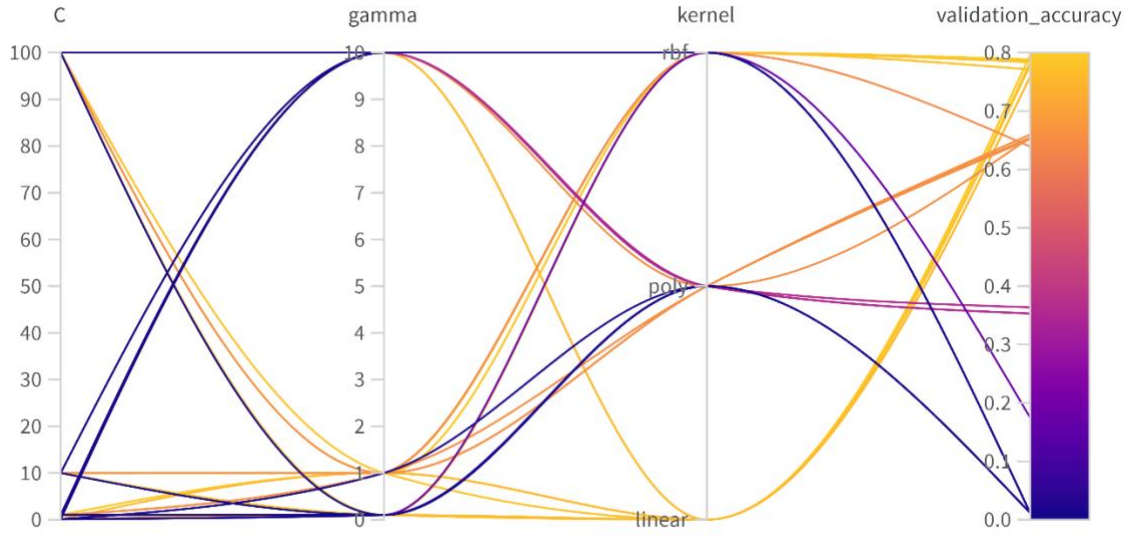

Abbildung 38: Getestete Werte und resultierende Validation-Accuracy für das Hyperparamater-Tuning des SVM-Classifiers

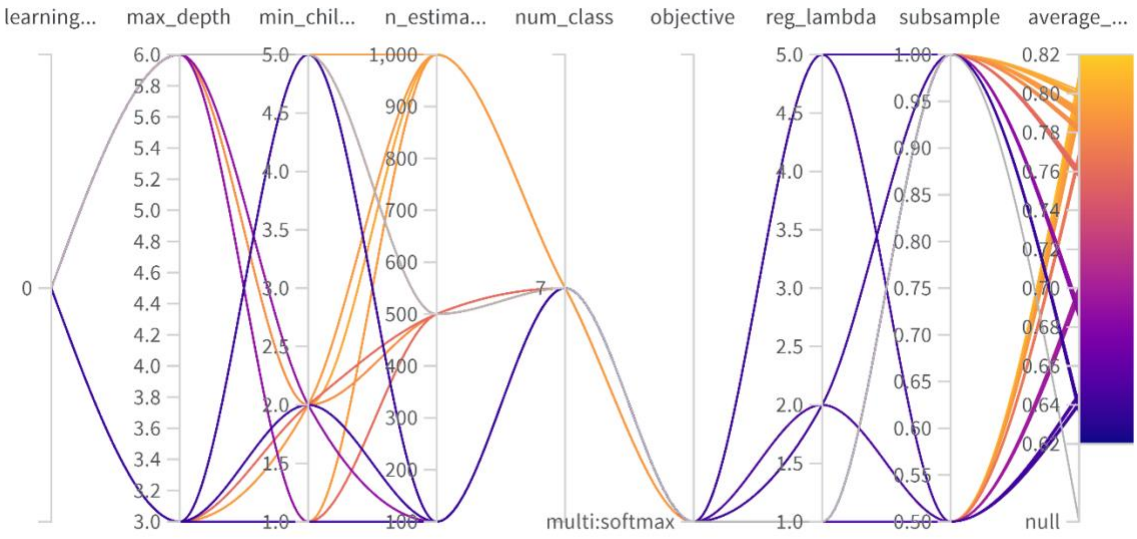

Abbildung 39: Getestete Werte und resultierende Average-Validation-Accuracy für das Hyperparamater-Tuning des XGB-Classifiers

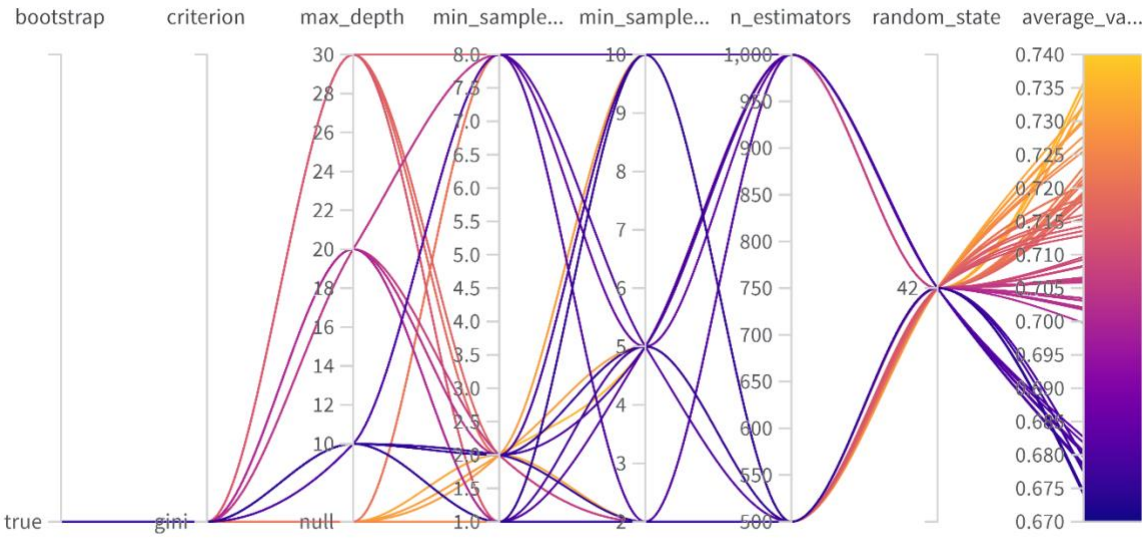

Abbildung 40: Getestete Werte und resultierende Average-Validation-Accuracy für das Hyperparamater-Tuning RF-Classifiers

12.3.3 Feature Importance

In den Abbildung 41 - Abbildung 44 werden die ergänzenden Darstellungen für Kapitel 8 aufgeführt.

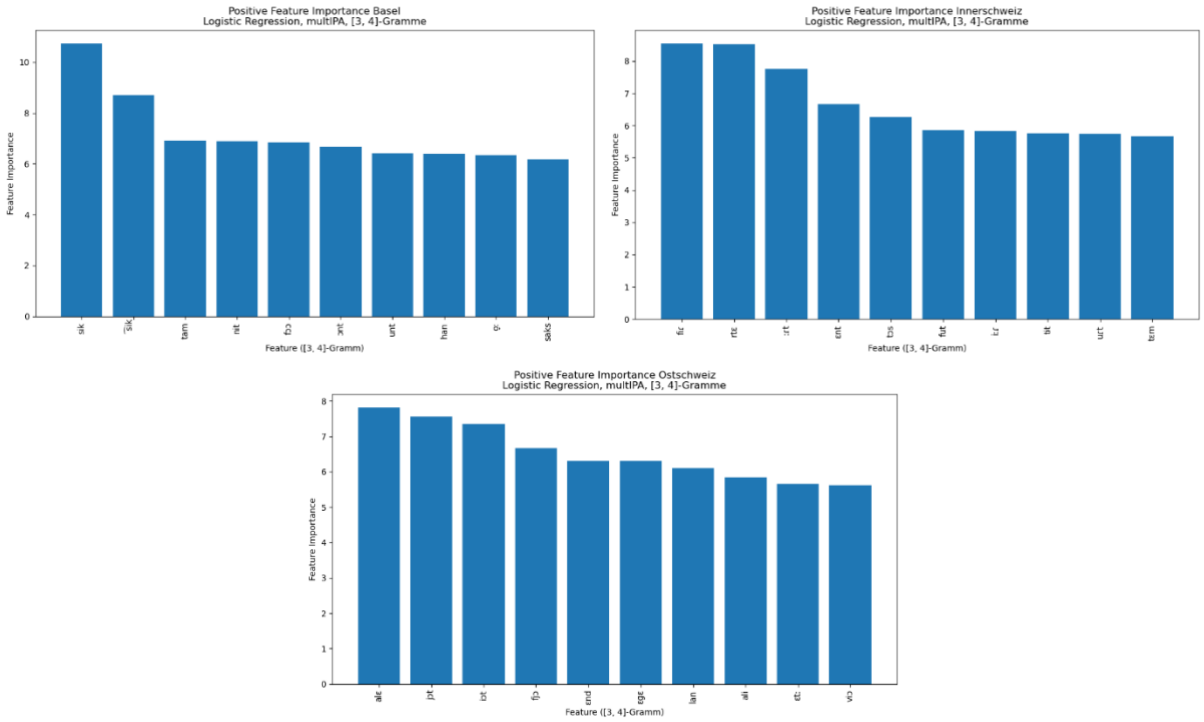

Abbildung 41: Die zehn Features mit der höchsten positiven Feature-Importance für die in Kapitel 8 nicht dargestellten Dialekte

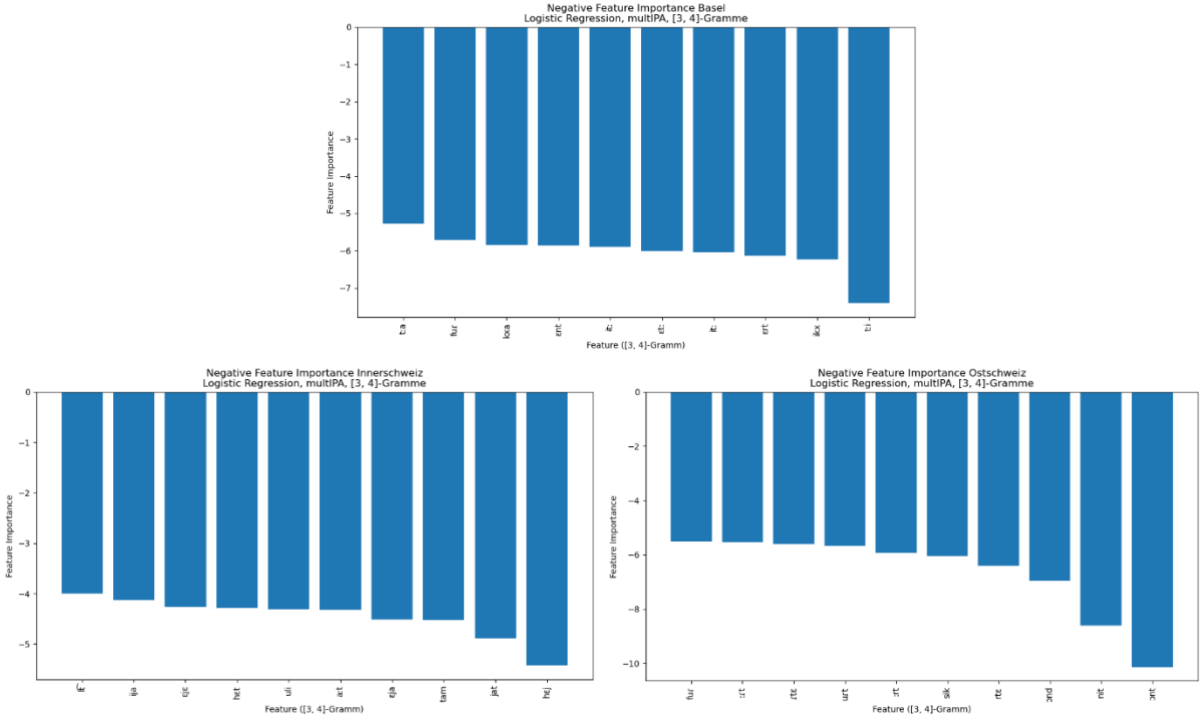

Abbildung 42: Die zehn Features mit der höchsten negativen Feature-Importance für die in Kapitel 8 nicht dargestellten Dialekte

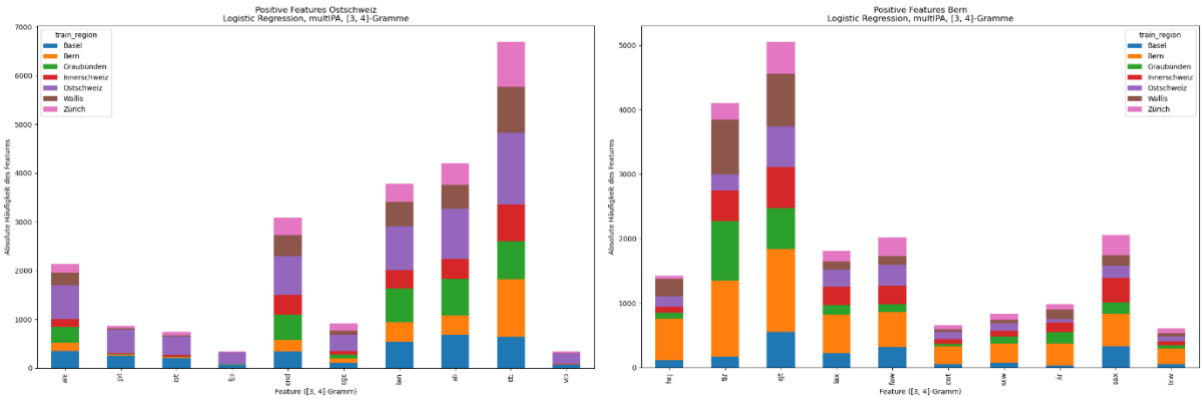

Abbildung 43: Häufigkeit der zehn wichtigsten positiven in den einzelnen Dialektregionen für die in Kapitel 8 nicht dargestellten Dialekte

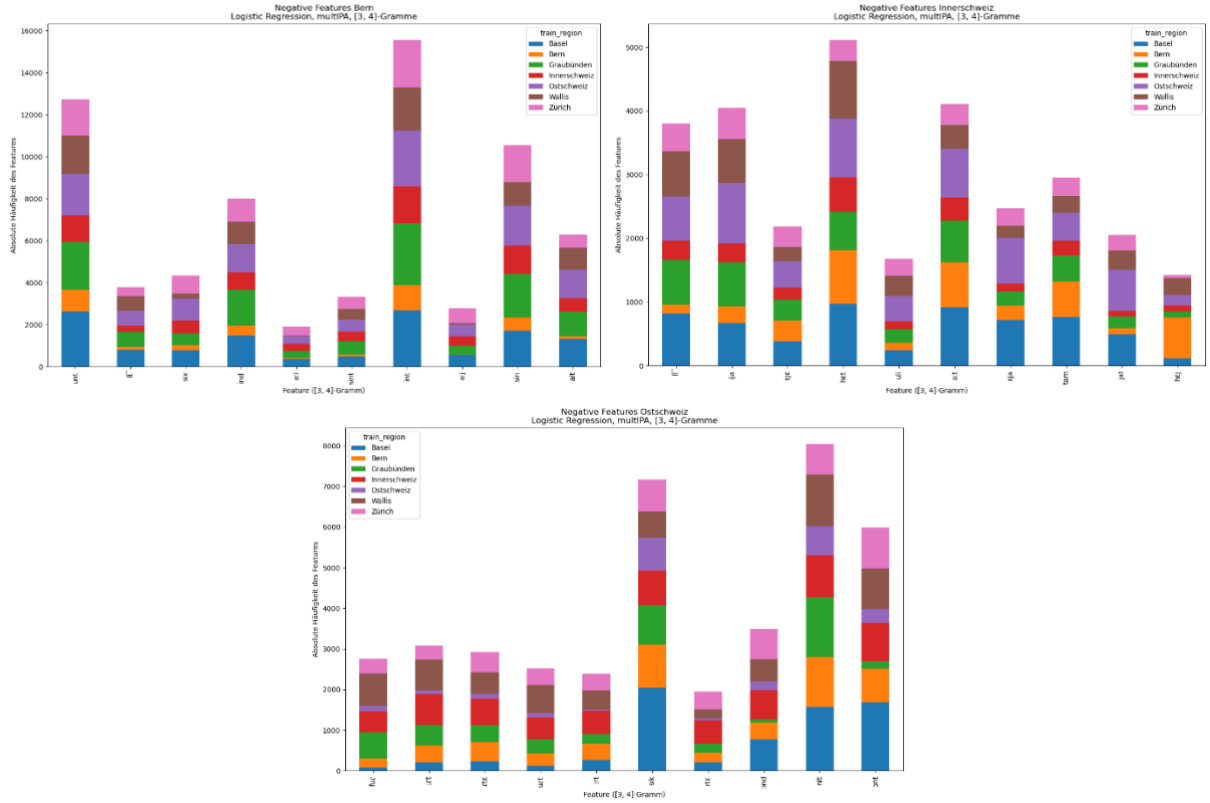

Abbildung 44: Häufigkeit der zehn wichtigsten negativen in den einzelnen Dialektregionen für die in Kapitel 8 nicht dargestellten Dialekte