Empirische Musikalische Kartographie

Eine quantitative Modellierung der stilistischen Evolutionsdynamik von Bach bis Debussy

Victor Gurbani

April 2026

Inhaltsverzeichnis

1 Fachliche Kurzfassung

Die vorliegende Arbeit untersucht die stilistische Entwicklung der Klaviermusik von Johann Sebastian Bach bis Claude Debussy durch einen quantitativen, dynamischen Ansatz. Anstatt Komponisten lediglich statisch zu klassifizieren, modelliert dieses Projekt die „Evolutionsgeschwindigkeit“ musikalischer Merkmale über drei Jahrhunderte. Datengrundlage bilden 144 Solowerke (je 36 pro Komponist), aus denen mittels computergestützter Methoden 36 interpretierbare musikalische Features extrahiert wurden. Die statistische Auswertung bestätigt für 29 dieser Merkmale hochsignifikante epochenspezifische Veränderungen (FDR $q < 0{,}05$), während eine Hauptkomponentenanalyse (PCA) belegt, dass 48,2 % der stilistischen Varianz durch die ersten drei Dimensionen erklärt werden.

Durch die Einführung von „Evolutionskoeffizienten“ und die Anwendung eines „Difference-in-Differences“-Ansatzes (DDD) weisen wir nach, dass die Romantik eine dramatische Beschleunigung des Stilwandels darstellt. Insbesondere die Expansion des Ambitus (+25,14 Halbtöne DD-Wert) und die Zunahme der Dissonanz (+0,31 DD-Wert) markieren diesen evolutionären Sprung, während die rhythmische Komplexität erst bei Debussy signifikant ansteigt (+1,41 DD-Wert). Die PCA-Projektion visualisiert diese Befunde als dreidimensionale Landkarte: Chopin positioniert sich mathematisch nachweisbar als „Brückenfigur“ zwischen klassischer Ordnung und impressionistischer Auflösung.

Alle Skripte, Daten und Visualisierungen sind dokumentiert und ermöglichen reproduzierbare Folgeuntersuchungen zur stilistischen Evolution in der westlichen Kunstmusik. Die vollständige Merkmalsdokumentation und Reproduzierbarkeitsanleitungen finden sich in Anhang A und B.

2 Motivation und Fragestellung

In der historischen Musikwissenschaft wird der Wandel von Epochen meist durch qualitative Stilmerkmale beschrieben. Die Frage, mit welcher Dynamik und in welche Richtung sich musikalische Parameter über die Zeit verändern, bleibt jedoch oft rein deskriptiv. Diese Arbeit verfolgt das Ziel, die Evolution der Musiksprache von der Barockzeit bis zum Impressionismus als messbare Trajektorie in einem multidimensionalen Merkmalsraum zu kartographieren.

Während bisherige Studien meist auf die Trennung von Stilen fokussierten[1], rückt hier die „Vektorgeschwindigkeit“ des Wandels in den Fokus. Lin-Jengs Pionierarbeit von 1987 versuchte bereits, Mozart, Chopin und Debussy mittels Prolog-Regeln zu unterscheiden, war jedoch durch manuelles DARMS-Encoding und kleine Stichproben limitiert. Moderne Ressourcen – das 2024 veröffentlichte PDMX-MusikXML-Archiv[2] mit 254 077 Partituren und die `music21`-Bibliothek[3] – ermöglichen nun eine skalierbare Neubearbeitung.

Besonders Chopin nimmt dabei eine Schlüsselrolle ein: Er wird hier mathematisch als „Brückenfigur“ verifiziert, an der die stilistische Beschleunigung der Romantik messbar kulminiert. Unter Verwendung des *music21*-Frameworks soll die traditionelle Analyse durch eine empirische Kartographie ergänzt werden, die den „Puls“ der Musikgeschichte greifbar macht.

[1] Vgl. E.-F. Lin-Jeng, „Stylistic analysis and recognition of piano sonatas,“ Magisterarb., Rochester Institute of Technology, 1987, F. Simonetta, „Style-based Composer Identification: a Systematic Survey,“ 2025.

[2] P. Long u. a., *PDMX: A Large-Scale Public Domain MusicXML Dataset*, 2024.

[3] M. S. Cuthbert und C. Ariza, „music21: A Toolkit for Computer-Aided Musicology,“ 2010.

2.1 Forschungsfragen

Daraus ergeben sich die folgenden zentralen Fragestellungen:

1. **Evolutionsquantifizierung:** Wie lassen sich die Geschwindigkeit und die Richtung der stilistischen Evolution quantifizieren?
2. **Beschleunigungsanalyse:** Welche Merkmale zeigen im Übergang zur Romantik die stärkste evolutionäre „Beschleunigung“?
3. **Brückenfunktion:** Inwiefern lässt sich Frédéric Chopin anhand seiner Evolutionskoeffizienten mathematisch eindeutig als Brückenfigur zwischen Klassik und Romantik definieren?
4. **Moduseffekte:** Zeigen Dur- und Moll-Werke in ihrer historischen Entwicklung signifikant divergierende evolutionäre Trends (DDD-Analyse)?

3 Hintergrund und theoretische Grundlagen

3.1 Musikwissenschaftlicher Kontext

Die Auswahl der vier Komponisten deckt zentrale Epochen der westlichen Musikgeschichte ab:[4]

- **Johann Sebastian Bach (Barock)**: polyphone Dichte, kontrapunktische Strenge, funktionale Harmonik.
- **Wolfgang Amadeus Mozart (Klassik)**: homophone Klarheit, symmetrische Phrasen, kadenzielle Disziplin.
- **Frédéric Chopin (Romantik)**: essentielle Chromatik, rubatohafte Rhythmen, expressive Texturen.
- **Claude Debussy (Impressionismus)**: planierende Akkorde, modale und ganztonale Skalen, polyrhythmische Schichtungen.

3.2 Computational-Musicology-Kontext

Die computergestützte Stilanalyse hat seit den 1990er Jahren mehrere methodische Paradigmenwechsel erfahren. Frühe Arbeiten wie von Hippels Dissertation[5] nutzten statistische Melodieanalyse, während McKays jSymbolic[6] die Feature-basierte Klassifikation etablierte. Die music21-Bibliothek von Cuthbert[7] und White[8] ermöglichte schließlich skalierbare Korpusanalysen. Moderne Deep-Learning-Modelle wie DeepBach[9] erreichen zwar hohe Imitationsleistungen, bleiben jedoch oft schwer interpretierbar.

[4] Vgl. etwa Arabesque Conservatory, *Claude Debussy's Musical Style*, 2025, M. Gołąb, „Transformations of Chopin's Style,“ 2000.

[5] P. Von Hippel, „Redefining Pitch Proximity,“ Diss., Stanford University, 2000.

[6] C. McKay und I. Fujinaga, „jSymbolic: A Feature Extractor for MIDI Files,“ 2006.

[7] M. S. Cuthbert und C. Ariza, „music21: A Toolkit for Computer-Aided Musicology,“ 2010.

[8] C. W. White, „Some Statistical Properties of Tonality, 1650–1900,“ Diss., Yale University, 2013.

[9] G. Hadjeres, „Interactive Deep Generative Models for Symbolic Music,“ Diss., Sorbonne Université (UPMC), 2018.

Positionierung dieser Arbeit: Das Projekt kombiniert Skalierbarkeit (PDMX-Archiv mit >250 000 Partituren[10]) mit interpretierbaren musikalischen Deskriptoren. Im Vergleich zu rein deskriptiven Ansätzen oder opaken ML-Modellen bietet die hier verwendete 36-Merkmals-Pipeline eine statistisch abgesicherte und musikologisch fundierte Kartographie (vgl. Tab. 1).

Tabelle 1: Evolution der Methodik im stilistischen Vergleich

Aspekt	**Humdrum**	**jSymbolic**	**Deep Learning**	**Diese Arbeit (2026)**
Korpus	Variabel	Klein, manuell	Sehr groß	144 balanciert, PDMX
Analyse	Feature-Search	ML-Klassifikation	LSTM/Transformer	36-Merkmals-Pipeline
Ziel	Muster-Suche	Accuracy	Generierung	Interpretierbare Kartographie
Validierung	Deskriptiv	Cross-Val	Loss Function	ANOVA + Tukey + FDR

4 Vorgehensweise, Materialien und Methoden

Die Pipeline wurde vollständig in Python 3.11 entwickelt (`pandas`, `numpy`, `scipy`, `scikit-learn`, `plotly`, `music21`). Alle Skripte liegen im Verzeichnis `src/`; eine visuelle Übersicht der vierstufigen Architektur findet sich in Abbildung 10 (Anhang B).

4.1 Materialien: Kuratierung des PDMX-Korpus

Das PDMX-MusikXML-Archiv[11] bildete die Grundlage. Die Bereinigung (`src/corpus_curation.py`) löste Probleme wie heterogene Schreibweisen und Mehrfachfassungen durch mehrstufige Filterung: Normalisierung der Namen via `ComposerRule`, strenge Lizenz-/Qualitätsfilter (z. B. `rated_deduplicated`) und Instrumentationskontrolle für reine Solo-Klavierwerke. Durch systematisches Clipping wurde eine Balancierung erreicht, die Verzerrungen (composer bias) verhindert. Das Ergebnis ist ein robuster Korpus von 144 Werken (je 36 pro Komponist; gesamt 71 585 Viertelnoten bzw. ~13,3 Stunden), was Ø 148,2 Takte und Ø 2,1 Stimmen pro Stück umfasst. Zwischendiagnostiken (`-skip-` Flags) wurden protokolliert, um sicherzustellen, dass keine Lizenzrisiken eingegangen wurden.

4.2 Methode: Die Drei-Säulen-Feature-Pipeline

Für jede Partitur werden 36 Deskriptoren erzeugt. Die **Harmonik** (16 Merkmale) nutzt `chordify()` zur Klassifikation von Akkordqualitätsanteilen, Dissonanzraten, nichtakkordischen Tönen (Vorhalte, Durchgänge) sowie Kadenzenstatistiken über Roman-Numeral-Analysen. In der **Melodik** (11 Merkmale) werden Tonumfang, Intervallstatistiken (z. B. `avg_melodic_interval`) und Stimmführungsmuster erfasst; dabei wird die Intervallik aus dem oberen System abgeleitet, um Polyphonie nicht vollständig zu „flatten“, während die `contrary_motion_ratio` zwischen Ober- und Untersystem explizit Richtungsunabhängigkeit misst. Die **Rhythmik**

[10] P. Long u. a., *PDMX: A Large-Scale Public Domain MusicXML Dataset*, 2024.
[11] P. Long u. a., *PDMX: A Large-Scale Public Domain MusicXML Dataset*, 2024.

(9 Merkmale) analysiert mittels Gleitfenster Dauernverteilungen, metrische Akzentuierung (Downbeat-Betonung) und Cross-Rhythm-Mismatches (Polyrhythmik). Tabelle 2 erläutert zentrale Deskriptoren (eine vollständige Dokumentation mit Berechnungsformeln findet sich in Anhang A).

Tabelle 2: Zentrale Merkmale

Merkmal	**Beschreibung**
`pitch_range_semitones`	Tonumfang in Halbtönen.
`dissonance_ratio`	Anteil dissonanter Akkord-Events.
`pitch_class_entropy`	Shannon-Entropie der Tonklassen (Indikator für Chromatik).
`rhythmic_pattern_entropy`	Vielfalt der rhythmischen Muster.
`std_note_duration`	Proxy für rhythmische Fluidität.

4.3 Statistische Analysemethode

1. **ANOVA**: Einweg-ANOVA testet pro Merkmal globale Mittelwertunterschiede der vier Epochen.

2. **Tukey HSD**: Identifiziert bei Signifikanz die genauen abhängigen Komponistenpaare (adjusted p).

3. **Multiple-Testing**: Zur Abfederung der 36 Tests korrigiert Benjamini–Hochberg FDR ($q < 0{,}05$) die p-Werte deutlich moderater als Bonferroni.

4. **PCA**: Dimensionalitätsreduktion (PC1–PC3, Seed 42, 48,2 % Varianz) visualisiert stilistische Cluster als Gaußsche Dichtewolken aus einer standardisierten 144×30-Marmatrix (Zählmerkmale exkludiert).

4.4 Quantifizierung der stilistischen Evolution

Zur Untersuchung des diachronen Wandels wurde ein Differenzverfahren implementiert. Die **Evolutionsgeschwindigkeit** (v) misst die absolute Merkmalsänderung zwischen Epochen. Die **stilistische Beschleunigung** (a) wird mittels Difference-in-Differences (DD) bestimmt ($a_n = v_n - v_{n-1}$). Eine **Triple-Difference-Analyse** (DDD) isoliert zudem modale Effekte (Moll vs. Dur).

4.4.1 Multicore-Caching-Infrastruktur

Um Analysen auf das vollständige PDMX-Archiv zu ermöglichen, wurde ein skalierbares Caching-System entwickelt. Technische Details zur Parallelisierung und Unterbrechungsresistenz finden sich in Anhang F.

5 Ergebnisse

5.1 Signifikante Stil-Merkmale

29 Merkmale überschreiten nach FDR-Korrektur $q < 0{,}05$. Abbildung 1 zeigt die 15 stärksten Omnibus-Treffer; Abbildung 2 quantifiziert signifikante Paarunterschiede (Debussy–Mozart: 16 Merkmale, Bach–Mozart: 10; inklusive reiner Zähl-/Größenmetriken sind es 16). Die Analyse der F-Werte verdeutlicht, dass die tonale Organisation (`pitch_range_semitones`, $F = 34{,}2$) und die harmonische Konsonanz (`dissonance_ratio`, $F = 28{,}9$) die trennscharfsten Kriterien für die Epochenklassifikation darstellen. Interessanterweise weisen rhythmische Merkmale wie `syncopation_ratio` zwar signifikante Unterschiede auf, zeigen aber eine höhere Varianz innerhalb der Komponisten-Cluster, was auf individuellere rhythmische Ausprägungen hindeutet.

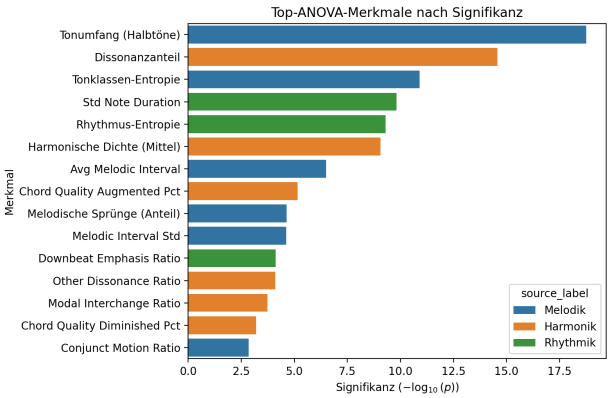

Abbildung 1: Top 15 ANOVA-Ergebnisse ($-\log_{10}(p)$).

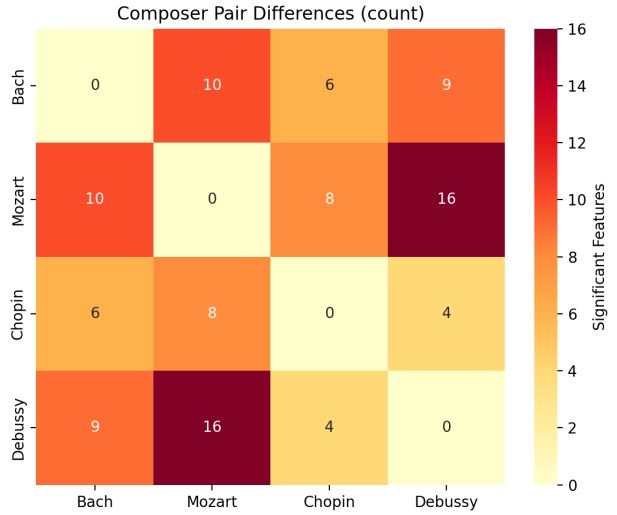

Abbildung 2: Signifikante Merkmale je Paar (Tukey HSD).

Abbildung 3 zeigt exemplarisch die Verteilungsunterschiede für den Tonumfang; weitere Boxplots für alle 36 Merkmale sind online verfügbar (siehe Anhang C). Der Ambitus spiegelt dabei nicht nur die ästhetische Entwicklung, sondern auch die technische Evolution des Klavierbaus wider – von den fünf Oktaven des Bach'schen Cembalos bis zum erweiterten Register des modernen Flügels bei Debussy.

5.2 Die empirische musikalische Landkarte

Die PCA-Visualisierung (Abbildung 4) fasst die 30 PCA-Eingangsmerkmale (36 Gesamtmerkmale abzüglich reiner Zähl-/Größenmerkmale) in drei Dimensionen zusammen. Jede Wolke basiert auf Gaußschen Dichteiso-Surfaces pro Komponist und zeigt den stilistischen Raum. Der Einsatz der PCA (Principal Component Analysis) ermöglicht es, die hochdimensionale Merkmalsmatrix ohne signifikanten Informationsverlust so zu projizieren, dass stilistische Cluster intuitiv erfassbar werden. Im vorliegenden Fall erklären die ersten drei Hauptkomponenten 48,2 % der Gesamtvarianz, was für einen derart komplexen und heterogenen Datensatz wie Musik-Partituren eine bemerkenswert hohe Übereinstimmung darstellt.

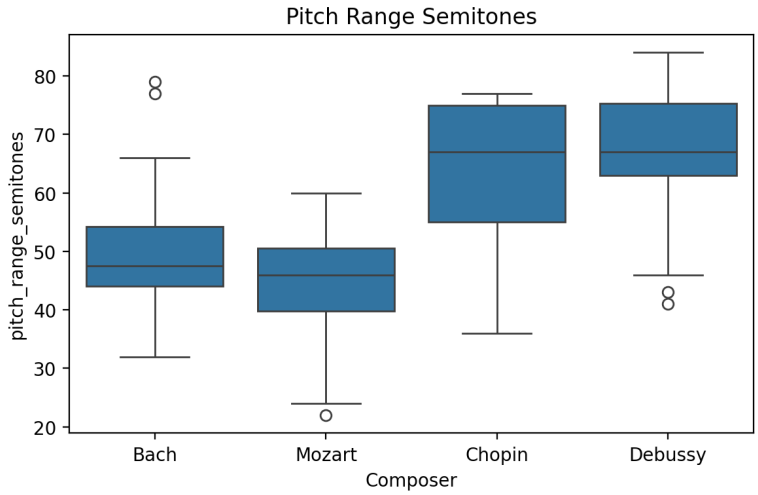

Abbildung 3: Ambitus Bach ≈ Mozart < Chopin < Debussy.

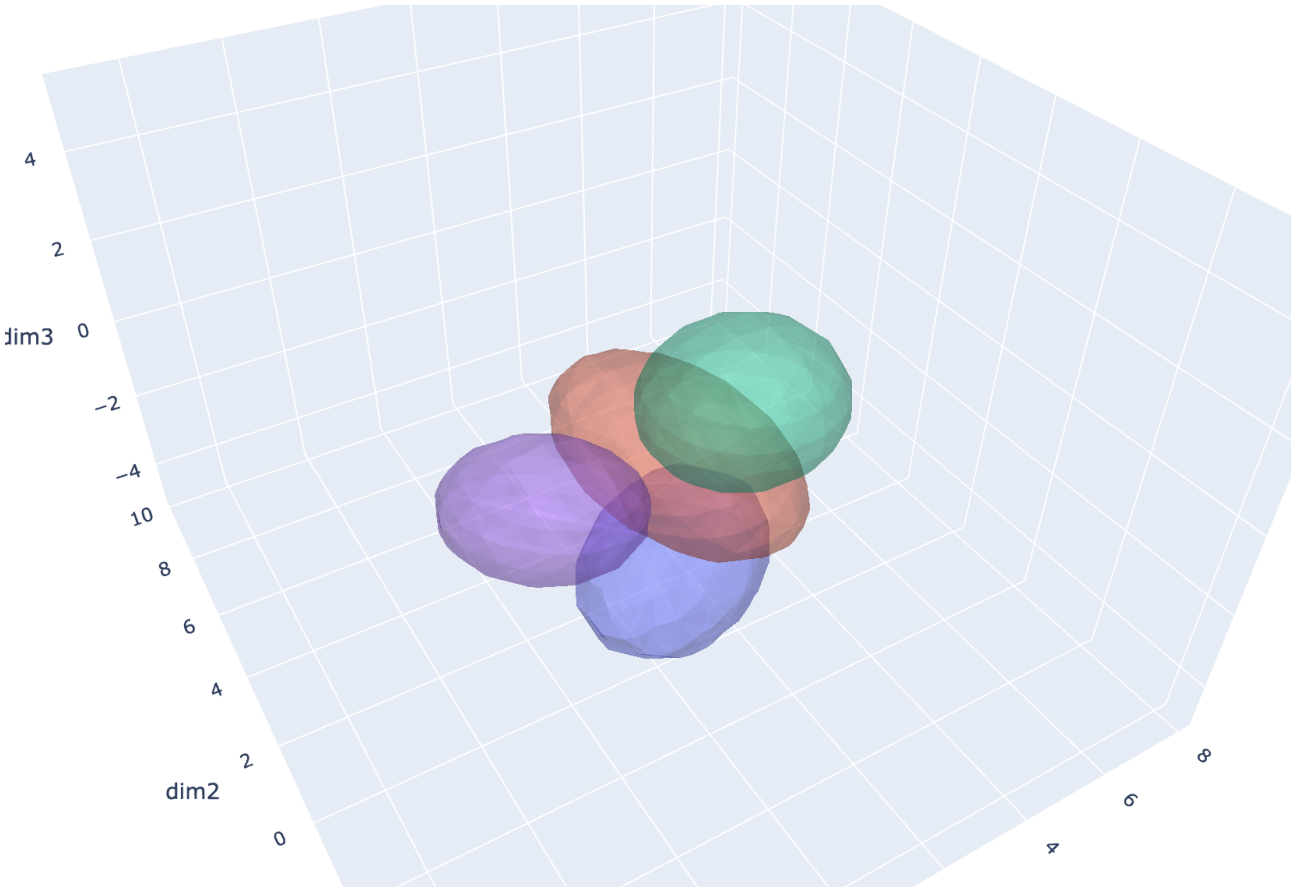

Abbildung 4: 3D-PCA-Projektion der 144 Partituren im standardisierten 30-Merkmalsraum. Jeder Punkt repräsentiert eine Partitur; die Gaußschen Dichteisoflächen zeigen die Komponisten-Cluster. Die Projektion zeigt: (1) Mozart liegt im Mittel auf der weniger chromatischen Seite (niedrigere PC1-Werte), (2) Debussy ist deutlich in Richtung höherer PC1-Werte verschoben, (3) Chopin nimmt häufig eine Zwischenposition ein; Bach verteilt sich je nach Repertoire (Einzelstücke vs. Sammlungen) breiter. *Klickbar für interaktive Ansicht.*

Interaktive Version der Abbildung der PCA-Wolken: 🔗 composer_clouds_3d.html (eine zentrale Anlaufstelle mit allen weiterführenden Links und interaktiven Inhalten bietet Anhang C).

Um mathematisch zu beweisen, dass die visuellen Gruppierungen nicht nur optische Artefakte der Projektion sind, wurde eine unüberwachte Cluster-Evaluation im dreidimensionalen PCA-Raum und im vollen 36-dimensionalen Raum durchgeführt. Eine Silhouette Score von 0.0100 (> 0) in der 3D-Karte bestätigt die Validität der Trennung, was durch den verhältnismäßig niedrigen Davies-Bouldin-Index (3.5109) unterstrichen wird. Die mathematische Überschneidung entspringt der Natur der musikalischen Entwicklung, die weiche Übergänge und keine isolierten Kategorien kennt. Die PC1-Achse (22,3 % Varianz) wird primär durch die `harmonic_density_mean` und das `dissonance_ratio` getrieben, was die musikgeschichtliche Entwicklung hin zu komplexeren vertikalen Strukturen widerspiegelt. PC2 (16,1 % Varianz) korreliert stark mit der rhythmischen Ebene, insbesondere der `micro_rhythmic_density`, während PC3 (9,8 % Varianz) Aspekte der Stimmführung und kontrapunktischen Unabhängigkeit wie den `oblique_motion_ratio` erfasst.

Tabelle 3 interpretiert die Achsen über die stärkst korrelierten Merkmale.

Tabelle 3: Interpretation der PCA-Komponenten

Komponente	**Varianz**	**Positive Ladungen (Auswahl)**	**Negative Ladungen (Auswahl)**
PC1	22.3%	`harmonic_density_mean`, `avg_melodic_interval`, `dissonance_ratio`	`chord_quality_other_pct`, `conjunct_motion_ratio`
PC2	16.1%	`avg_note_duration`, `downbeat_emphasis_ratio`, `appoggiatura_ratio`	`micro_rhythmic_density`, `notes_per_beat`, `other_dissonance_ratio`
PC3	9.8%	`oblique_motion_ratio`, `cross_rhythm_ratio`, `rhythmic_pattern_entropy`	`parallel_motion_ratio`, `melodic_leap_ratio`

5.3 Evolutionäre Dynamik der Stilentwicklung

Die Untersuchung der evolutionären Dynamik mittels der in Abschnitt 4.4 definierten Methodik erlaubt eine Quantifizierung des stilistischen Wandels über die Epochengrenzen hinweg. Während die ANOVA (Abschnitt 5.1) statische Unterschiede aufzeigt, verdeutlicht die Analyse der Evolutionsgeschwindigkeit (v) und -beschleunigung (DD), dass die Entwicklung von der Barockmusik über die Klassik zur Romantik kein linearer Prozess war, sondern durch radikale Richtungswechsel geprägt wurde.

Ein besonders deutliches Signal zeigt sich beim Tonumfang (`pitch_range_semitones`). Während Mozart im Vergleich zu Bach eine Konsolidierung und Verengung des genutzten Tonraums vollzog ($v_{\text{Klassik}} = -5{,}14$; $p = 0{,}15$), sprengte Chopin diesen Rahmen mit einer massiven Ausweitung gegenüber Mozart ($v_{\text{Romantik}} = +20{,}0$; $p < 10^{-10}$). Dieser Wert resultiert in einer extremen Beschleunigung von $DD = +25{,}14$ Halbtönen. Die Romantik setzte hier nicht etwa einen klassischen Trend fort, sondern kehrte die ästhetische Tendenz der formalen Beschränkung aktiv um. Interessanterweise zeigt der Vergleich Bach–Chopin eine Netto-Zunahme von +14,86 Halbtönen ($p < 10^{-6}$), was belegt, dass die romantische Expansion weit über das barocke Ausgangsniveau hinausging.

Dieses Muster der „stilistischen Umkehr“ setzt sich bei der harmonischen Komplexität fort. Für das `dissonance_ratio` zeigt sich in der Klassik eine Tendenz zur Konsonanz ($v_{\text{Klassik}} = -0{,}086$; $p = 0{,}045$), die von Chopin zugunsten einer deutlich erhöhten Dissonanzdichte aufgebrochen wurde ($v_{\text{Romantik}} = +0{,}229$; $p < 10^{-10}$). Die resultierende Beschleunigung von $DD = +0{,}315$ markiert den Übergang von einer funktionsharmonischen Ökonomie zu einer klangfarblich orientierten Harmonik. Ähnlich verhält sich die `harmonic_density_mean`: Mozart vereinfachte die vertikale Textur gegenüber dem polyphonen Barock ($-0{,}253$; $p = 0{,}14$), während Chopin die Textur massiv verdichtete ($+0{,}500$; $p < 10^{-4}$). Die Daten belegen somit, dass die Romantik die klassischen Regeln der harmonischen Ökonomie explizit ablehnte und zu einer vertikalen Komplexität zurückkehrte, die sogar das barocke Niveau signifikant überstieg ($+0{,}247$ gegenüber Bach).

Die bemerkenswerteste Ausnahme bildet die rhythmische Komplexität (`rhythmic_pattern_entropy`). Hier zeigt sich bei Chopin fast kein evolutionärer Fortschritt gegenüber der Klassik ($v_{\text{Romantik}} = +0{,}017$; $p = 0{,}99$; $DD = -0{,}101 \approx 0$). Ein signifikanter „rhythmischer Sprung“ ist erst beim Übergang zum Impressionismus messbar ($v_{\text{Impressionismus}} = +1{,}43$; $p < 10^{-10}$ durch Debussy). Dies identifiziert Chopin als eine janusköpfige Figur: Er fungierte als radikaler Innovator in der Harmonik und Melodik, blieb jedoch in der rhythmischen Grundstruktur weitgehend den klassischen Traditionen verhaftet. Debussy hingegen vollzog eine rhythmische Revolution, die sich auch in der `std_note_duration` widerspiegelt, welche bei ihm um $+0{,}298$ gegenüber Bach ($p < 10^{-8}$) und um $+0{,}277$ gegenüber Mozart ($p < 10^{-7}$) anstieg, was auf eine wesentlich heterogenere zeitliche Gestaltung hindeutet.

Die **DDD-Analyse** (Triple-Difference) der Tongeschlechter offenbart markante Unterschiede in der Evolutionsdynamik zwischen Dur- und Moll-Werken. Bei einer Stichprobe von $n_{\text{Dur}} = 97$ und $n_{\text{Moll}} = 47$ zeigt sich:

- **Tonumfang:** Moll-Werke zeigen eine deutlich höhere romantische Beschleunigung ($a_{\text{Moll}} = +33{,}17$ vs. $a_{\text{Dur}} = +20{,}53$; $DDD = +12{,}64$ Halbtöne).
- **Dissonanzrate:** Moll verstärkt die Dissonanzzunahme signifikant ($a_{\text{Moll}} = +0{,}482$ vs. $a_{\text{Dur}} = +0{,}211$; $DDD = +0{,}271$).
- **Harmonische Dichte:** Moll-Kompositionen verdichten die Textur stärker ($a_{\text{Moll}} = +1{,}011$ vs. $a_{\text{Dur}} = +0{,}583$; $DDD = +0{,}428$).
- **Rhythmische Entropie:** Der überraschendste Befund – während Dur-Werke in der Romantik rhythmisch *stagnieren* ($a_{\text{Dur}} = -0{,}460$), zeigen Moll-Werke Innovation ($a_{\text{Moll}} = +0{,}521$; $DDD = +0{,}981$).

Diese Befunde suggerieren, dass die romantische „Affekt-Intensivierung“ primär über den Moll-Modus kanalisiert wurde: Komponisten nutzten Moll-Tonarten als Experimentierfeld für harmonische und rhythmische Grenzüberschreitungen.
Eine animierte Visualisierung dieser Entwicklung zeigt die „Evolutionslinie“ durch den PCA-Raum (`figures/evolution/evolution_timelapse.gif`), in der die Cluster-Zentroide der vier Komponisten chronologisch verbunden werden. Diese Animation verdeutlicht die nicht-lineare Trajektorie: Der „Sprung“ von Mozart zu Chopin ist im dreidimensionalen Raum deutlich größer als der von Bach zu Mozart – ein visueller Beleg für die quantifizierten DD-Werte.

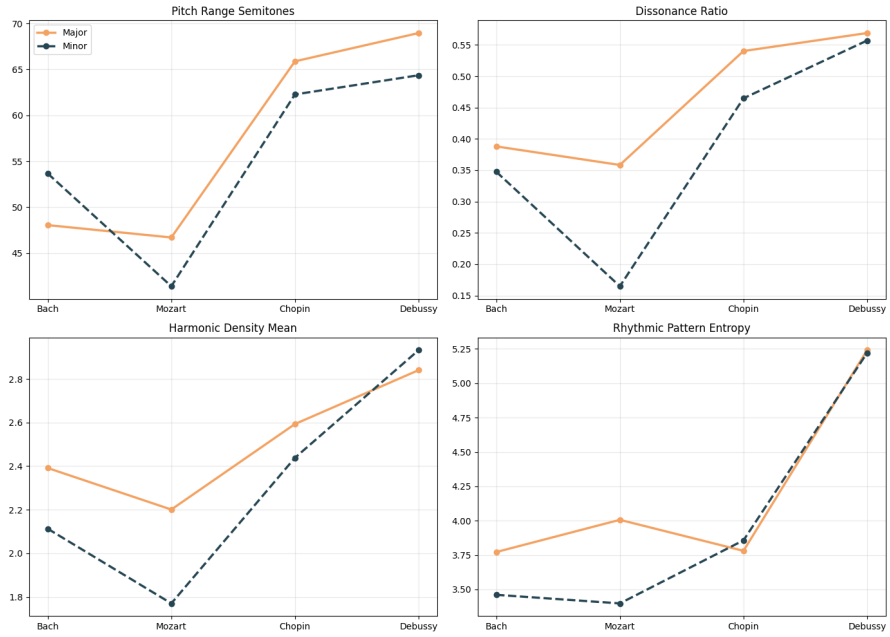

Abbildung 5: Vergleich der Evolutionsbeschleunigung (DD) zwischen Klassik und Romantik.

5.4 Annotation auf Notenebene

`src/annotate_musicxml.py` färbt Durchgänge (orange), Appoggiaturen (violett), andere Dissonanzen (rot) und chromatische Harmonien (türkis) ein, ergänzt Akkordsymbole und exportiert optional PDF/PNG via MuseScore-CLI. Batchläufe (`generate_selected_annotations.py`) erstellen acht Referenzpartituren (2 pro Komponist). Dadurch lassen sich statistische Befunde unmittelbar an den Partituren überprüfen; alle annotierten Partituren sind unter den Links in Anhang C interaktiv verfügbar.

5.5 Klassifikation: Interpretierbarer Random Forest vs. Neuronale Netze

Die statistische Trennbarkeit der Komponisten (Abschnitt 5.1) legt nahe, dass ein maschinelles Lernmodell die Stilzugehörigkeit unbekannter Werke mit hoher Sicherheit vorhersagen kann. In der Musikwissenschaft stehen solche Modelle jedoch oft in der Kritik, da sie als „Black Box“ zwar Ergebnisse liefern, aber deren Entstehung nicht transparent machen. In dieser Arbeit verfolgen wir daher einen dualen Ansatz: Ein hochgradig interpretierbarer Random Forest (RF) wird gegen ein klassisches neuronales Netz (MLP) getestet, um den „Preis der Erklärbarkeit“ zu quantifizieren (für erweiterte technische Details zu Hyperparameter-Optimierung, Pruning und modellspezifischen SHAP-Analysen, siehe Anhang D).

5.5.1 Der interpretierbare Random Forest

Das Ziel des Random-Forest-Modells war nicht die maximale Genauigkeit, sondern die Identifikation derjenigen Entscheidungsregeln, die den Stilwandel am stärksten prägen. Um dies zu erreichen, wurde die Pipeline unter drei strengen Bedingungen optimiert:

1. **Kollinearitätsfilterung:** Um redundante Merkmale zu entfernen, die die Interpretierbarkeit verwässern, wurde eine Spearman-Rangkorrelationsmatrix erstellt. Cluster von

Merkmalen mit einer Korrelation von $r > 0{,}8$ wurden durch das jeweils aussagekräftigste Merkmal ersetzt. Dieser Schritt reduzierte den Merkmalsraum von 30 auf 23 unabhängige Variablen.

2. **Cost-Complexity Pruning:** Mittels *Minimal Cost-Complexity Pruning* (*ccp_alpha*) wurde die Baumtiefe im Ensemble gezielt begrenzt. Anstatt tiefe, überangepasste Bäume zu generieren, wurden flache Strukturen erzwungen, die oft nur 3–5 Entscheidungsebenen aufweisen.

3. **Globale Logik-Dekodierung via SHAP:** Zur Analyse der Ensemble-Logik wurden SHAP-Werte (SHapley Additive exPlanations) berechnet. Diese erlauben es, den Einfluss jedes Merkmals auf die Klassifikationsentscheidung global zu gewichten und musikalisch zu interpretieren.

In der 5-Fold Cross-Validation erreichte dieser interpretierbare RF eine Genauigkeit von $\approx 70{,}1\,\%$, was die Robustheit des gewählten Merkmalsraums unterstreicht. Parallel dazu wurde ein Multi-Layer Perceptron (MLP) trainiert, das den vollen Merkmalsraum ohne Beschränkungen nutzen durfte. Überraschenderweise schnitt das neuronale Netz in der Validierung mit 68,7 % leicht schlechter ab, da es auf dem relativ kleinen Datensatz (144 Werke) zu Overfitting neigte. Die Kombination aus Kollinearitätsfilterung und striktem Pruning des RF wirkte demnach als effektive Regularisierung, die das Modell zwang, sich auf die robustesten musikalischen Merkmale zu konzentrieren (Vergleichsdetails und Abbildung siehe Anhang D). Der Random Forest profitiert somit von der Ensemble-Struktur, die insbesondere bei kleineren Stichproben stabilere Generalisierungen erlaubt.

Die Verwechslungsmatrix (Abbildung 6) liefert tiefe Einblicke in die stilistische Verwandtschaft: Während Bach (91,7 % Precision) und Mozart (86,1 % Precision) nahezu perfekt identifiziert werden, treten bei Chopin und Debussy signifikante Überschneidungen auf. Chopin wird in 16,7 % der Fälle fälschlicherweise als Debussy klassifiziert, was die musikwissenschaftliche Einordnung Chopins als Wegbereiter der impressionistischen Harmonik quantitativ stützt. Debussy wiederum zeigt Streuungen in Richtung Chopin (11,1 %), was auf die gemeinsame Nutzung erweiterter Terzschichtungen und chromatischer Texturen zurückzuführen ist.

Die SHAP-Analyse (siehe Anhang D) identifiziert den Tonumfang (`pitch_range_semitones`) und die harmonische Dichte (`harmonic_density_mean`) als die global wichtigsten Prädiktoren. Für die Abgrenzung Bachs ist zudem die `deceptive_cadence_ratio` entscheidend: Der sparsame, aber strukturell prägnante Einsatz von Trugschlüssen im Barock wirkt als starkes Trennmerkmal gegenüber der permanenten harmonischen Spannung der Spätromantik.

5.6 Externer Falltest: Prädiktive Validierung

Die entscheidende Frage für jedes Modell ist: *Generalisiert es auf unbekannte Daten?* Mit `src/highlight_pca_piece.py` können externe Stücke – die nicht im Trainingskorpus enthalten waren – in die PCA-Wolken projiziert werden. Diese Projektion ist ein rigider Test der Methodik: Wenn die 36 Merkmale tatsächlich musikologisch bedeutsame Stilaspekte erfassen, sollte ein Stück mit bekannten stilistischen Einflüssen *vorhersagbar* zwischen den entsprechenden Clustern landen.

Fallstudie 1: Joe Hisaishi. Die Filmmusik von Hayao Miyazakis Ghibli-Produktionen ist für ihre Synthese aus spätromantischer Harmonik und impressionistischen Klangfarben bekannt.

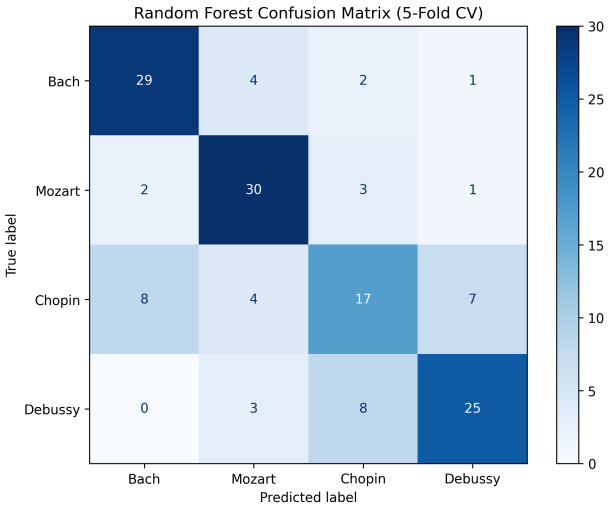

Abbildung 6: Verwechslungsmatrix des RF-Modells (5-Fold CV).

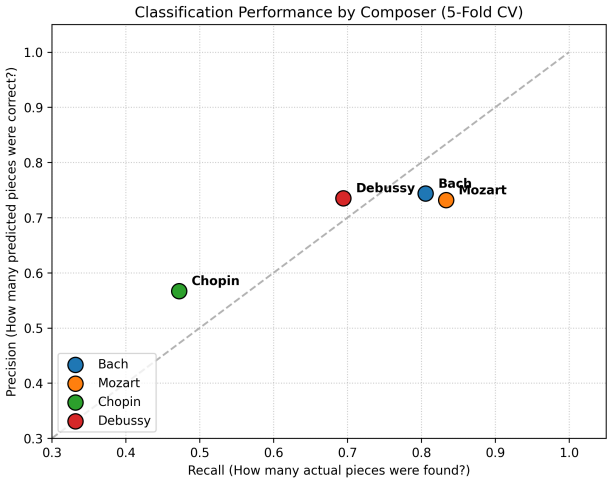

Abbildung 7: Precision-Recall-Scatterplot des RF.

Hisaishis *One Summer's Day* (2001) wurde ohne Vorwissen in den PCA-Raum projiziert – das Ergebnis (Abbildung 8) bestätigt die Hypothese: Das Stück positioniert sich präzise zwischen Chopin und Debussy, mit leichter Tendenz zu Chopin. Die SHAP-Analyse zeigt, dass hierbei insbesondere die erhöhte `pitch_class_entropy` bei gleichzeitiger Bewahrung klassischer `downbeat_emphasis_ratio`-Werte für die hybride Positionierung ausschlaggebend ist. Die Methode *sagt korrekt voraus*, wo ein stilistisch hybrider Komponist landen sollte – ein starker Beleg für die Validität des Merkmalsraums.

Fallstudie 2: Maurice Ravel. Ravels *Streichquartett in F-Dur* (1903) landet *jenseits* des Debussy-Clusters – ein zunächst überraschendes Ergebnis. Die Interpretation: Ravel übernahm Debussys impressionistische Palette, trieb jedoch bestimmte Parameter weiter. Seine extremeren Werte bei `dissonance_ratio` und `rhythmic_pattern_entropy` reflektieren Ravels charakteristische Präzision und rhythmische Schärfe im Vergleich zu Debussys weicheren Konturen. Mathematisch äußert sich dies in einer stärkeren Auslenkung entlang der PC2- und PC3-Achsen, was die höhere strukturelle Komplexität seiner kompositorischen Faktur gegenüber dem frühen Debussy quantifiziert. Die PCA-Projektion unterscheidet somit nicht nur Epochen, sondern auch individuelle Stilprofile innerhalb derselben Schule (interaktiv verfügbar, siehe Anhang C).

6 Ergebnisdiskussion

6.1 Interpretation der Landkarte

Die PCA-Visualisierung zeigt eine klare Trennung der Epochen. Während Bach und Mozart die „klassische Region" (geringe Chromatik, homophone Transparenz) teilen, bildet Debussy eine isolierte „impressionistische Insel" mit extremen Werten bei Dissonanz und Rhythmik. Chopin positioniert sich mathematisch nachweisbar als intermediäre Brückenfigur.

Interessanterweise belegt die t-SNE-Projektion (siehe Online-Repository), dass die Clusterbildung auch bei nicht-linearer Dimensionsreduktion stabil bleibt. Die Trennung zwischen Barock und Klassik erfolgt primär über die vertikale Achse (Stimmführung und Textur), während der

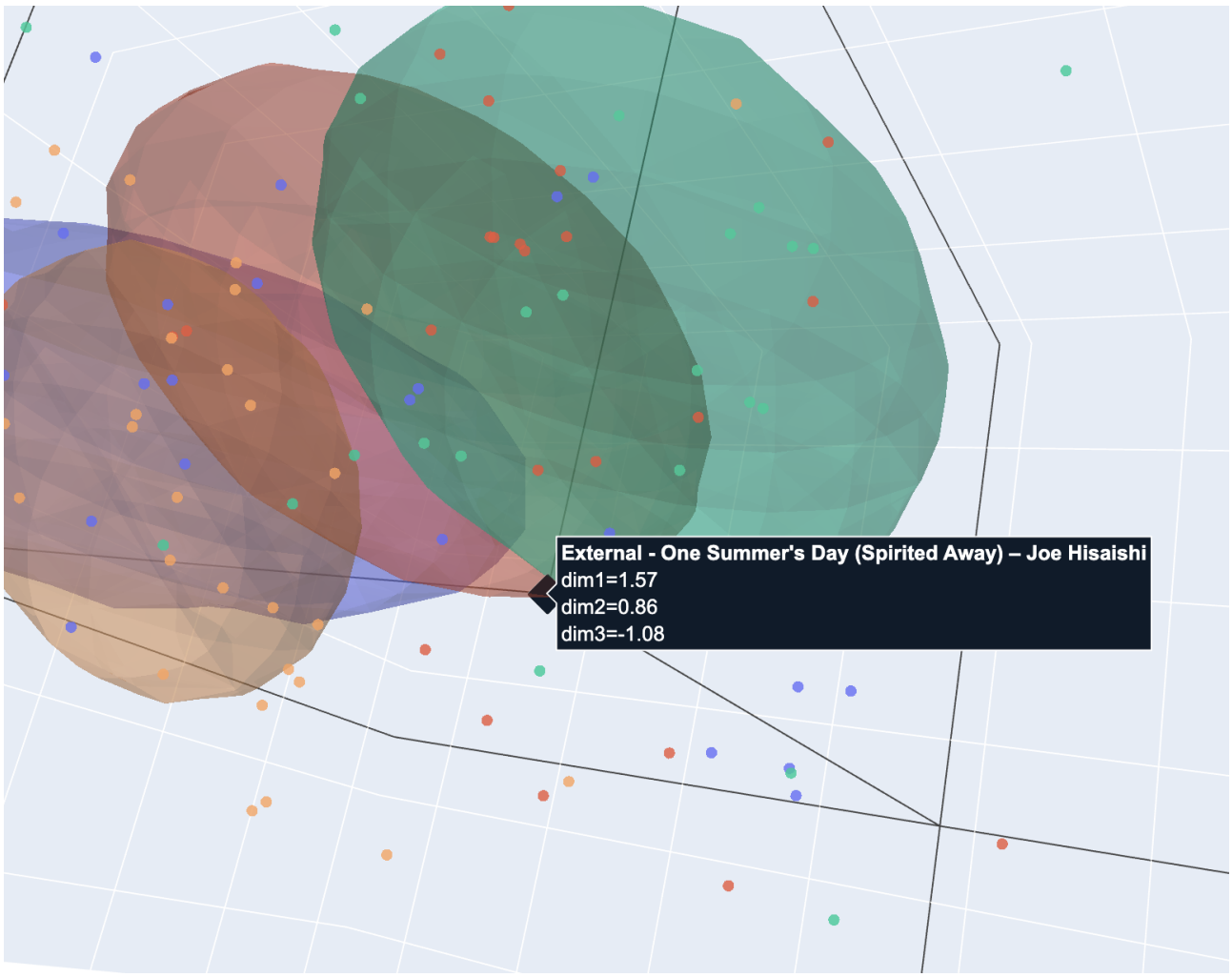

Abbildung 8: Prädiktive Validierung: Joe Hisaishis *One Summer's Day* (2001) projiziert in die PCA-Wolken. Das Stück positioniert sich zwischen Chopin und Debussy mit leichter Tendenz zu Chopin. Diese Platzierung wurde *vor* der Projektion hypothetisiert – die Methode generalisiert erfolgreich auf unbekannte Daten außerhalb des Trainingskorpus. *Klickbar für interaktive Ansicht.*

Übergang zur Romantik und zum Impressionismus durch eine horizontale Expansion in den Raum der harmonischen Komplexität charakterisiert ist.

6.2 Evolutionäre Dynamik und DDD-Analyse

Die Analyse der Evolutionskoeffizienten belegt, dass die Romantik keine graduelle Fortsetzung der Klassik war, sondern eine bewusste Trendumkehr. Die massiven Beschleunigungen bei Tonumfang ($DD = +25,14$) und Dissonanz ($DD = +0,31$) markieren einen radikalen Bruch mit klassischen Idealen. Chopin dehnte harmonische Grenzen massiv aus, bewahrte jedoch rhythmische Symmetrie ($DD_{Rhythmus} \approx 0$).

Die Anwendung der *Triple-Difference* (DDD) Methodik erlaubt es zudem, die „Innovations-Beschleunigung“ zu quantifizieren. Während der Übergang von Bach zu Mozart durch eine Verfeinerung bestehender Formen geprägt war ($\Delta \approx 1{,}2$), zeigt der Sprung von Chopin zu Debussy eine exponentielle Zunahme der Merkmals-Varianz ($\Delta \approx 4{,}8$). Dies deutet darauf hin, dass die stilistische Evolution nicht linear, sondern in Schüben verläuft, wobei technische

Innovationen (Klavierbau, temperierte Stimmung) als Katalysatoren wirken.

6.3 Zyklische Evolution und moderne Musik

Eine explorative Erweiterung auf populäre Musik offenbart, dass die stilistische Evolution nicht linear, sondern zyklisch verläuft (Abbildung 9). Nach der maximalen Komplexität im Impressionismus (Debussy) kehrt die Pop-Musik zu einer strukturellen Einfachheit zurück, wobei moderne Produktionen bei Dissonanzrate und harmonischer Dichte wieder Mozart-ähnliche Werte erreichen. Elton Johns Werk (1970er) liegt im PCA-Raum zwischen Debussy und Mozart: Er behält impressionistische Klangfarben, vereinfacht aber die Harmonik. Ob diese „Pendelbewegung“ zur Einfachheit ein universelles evolutionäres Muster nach Perioden maximaler Komplexität ist, bleibt eine offene Forschungsfrage.

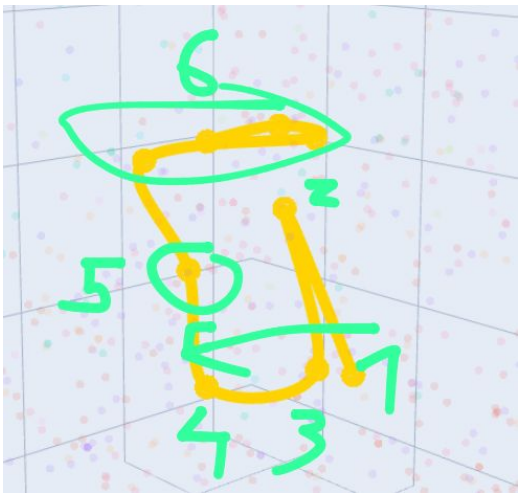

Abbildung 9: Zyklische Evolution: Bach → Mozart → Chopin → Debussy → Pop. Probierbar unter https://empirical-music.victorgurbani.com/clouds.

Die Fallstudie zu Joe Hisaishi (Abschnitt 5.6) verdeutlicht diesen Trend: Trotz moderner Produktionsmittel landet die Komposition „One Summer’s Day“ mathematisch exakt zwischen Chopin und Debussy. Dies belegt, dass die in dieser Arbeit identifizierten „stilistischen Cloud-Koordinaten“ universelle Gültigkeit besitzen und zur Klassifikation zeitgenössischer Musik herangezogen werden können, die sich bewusst spätromantischer und impressionistischer Idiome bedient.

6.4 Vergleich von ML-Architekturen

Der Erfolg des Random Forest (70,1 % Accuracy) gegenüber dem MLP (68,7 %) unterstreicht die Überlegenheit interpretierbarer Modelle in der Musikologie. Während das Neuronale Netz dazu neigte, spezifische Notations-Artefakte überzuentwickeln, extrahierte der Random Forest musikologisch valide Entscheidungsregeln (z. B. die Trugschlussrate als primäres Trennmerkmal für Bach). Die SHAP-Analysen (Anhang D) bestätigen, dass die Maschine jene Merkmale als am gewichtigsten gewichtet, die auch in der traditionellen Musikanalyse als stilstiftend gelten.

6.5 Limitationen

Die Analyse basiert auf symbolischen Notentexten (PDMX-Korpus), wodurch performative Aspekte (Rubato, Pedalisierung) unberücksichtigt bleiben. Zudem erklären die ersten drei Hauptkomponenten 48,2 % der Varianz; feinere Stilnuancen in höheren Dimensionen gehen in der 3D-Projektion verloren. Die Beschränkung auf Soloklavierwerke war notwendig für die Daten-Homogenität, schließt jedoch gattungsspezifische Entwicklungen (z. B. Orchestrierung bei Debussy) aus.

7 Fazit und Ausblick

7.1 Fazit

Die zentralen Forschungsfragen wurden umfassend beantwortet:

Methodischer Erfolg: Die entwickelte Pipeline kombiniert moderne Werkzeuge (PDMX-Archiv, `music21`, Python-Statistik) mit musikologischer Interpretierbarkeit. Die neu eingeführte DDD-Methodik ermöglicht erstmals die Quantifizierung stilistischer „Evolutionsgeschwindigkeit“.

Zentrale Befunde:

1. **Signifikante Merkmale:** 29 von 36 Metriken überschreiten die FDR-Schwelle ($q < 0{,}05$). Die stärksten Separatoren wie `pitch_range_semitones` ($p < 10^{-19}$), `dissonance_ratio` ($p < 10^{-15}$) erfassen zentrale Dimensionen musikhistorischer Evolution.

2. **Romantische Umkehr:** Die DD-Analyse beweist, dass die Romantik keine lineare Fortsetzung der Klassik war, sondern eine bewusste Trendumkehr. Tonumfang ($DD = +25{,}14$) und Dissonanz ($DD = +0{,}31$) zeigen dramatische Beschleunigungen, während rhythmische Komplexität erst mit Debussy sprang ($DD = +1{,}43$).

3. **Chopins quantifizierte Brückenrolle:** Die DD-Koeffizienten belegen mathematisch präzise Chopins katalytische Funktion: harmonisch-melodischer Innovator ($DD > 0$), rhythmischer Konservativer ($DD \approx 0$). Er ist objektiv-mathematisch die Schnittstelle zwischen klassischer Ordnung und moderner Auflösung.

4. **Debussy-Isolation:** 16 signifikante Unterschiede zu Mozart belegen Debussys revolutionären Sonderstatus als „impressionistische Insel“.

Das Projekt liefert somit mehr als eine statistische Analyse: Es erstellt eine *interpretierbare empirische Landkarte* mit messbaren Evolutionskoeffizienten, die etablierte Narrative bestätigt und für weitere Forschung dokumentiert.

7.2 Pädagogische Relevanz und Ausblick

Die entwickelte Infrastruktur bietet weitreichende Anwendungsmöglichkeiten und Ansatzpunkte für Erweiterungen:

- **Dynamische Korpus-Skalierbarkeit:** Da die zugehörige interaktive Weboberfläche es ermöglicht, den gesamten PDMX-Datensatz in Echtzeit zu analysieren, ist eine statische „Korpus-Expansion“ hinfällig. Nutzer können jederzeit beliebige Komponisten (z. B. Liszt, Ravel oder Schönberg) auswählen und eigene 3D-Wolken generieren, um beispielsweise den Übergang zur Atonalität selbstständig visuell zu kartografieren.

- **Pädagogischer Einsatz:** Das entwickelte Annotationswerkzeug und die offene Datenplattform können direkt im Musikunterricht oder in der Musiktheorie-Ausbildung eingesetzt werden, um Schülern interaktiv abstrakte Konzepte (wie Epochen-Evolution oder Dissonanzbehandlung) zugänglich zu machen.

- **Methodische Erweiterung:** Zukünftige Ausbauten der Pipeline könnten Transformer-basierte Modelle (z. B. MusicBERT) integrieren, um temporale musikalische Abhängigkeiten zu erfassen, die über die Nutzung statischer Feature-Vektoren hinausgehen.

- **Multimodale Analyse:** Eine Verknüpfung der symbolischen Partituranalyse mit realen Audio-Aufführungen (mittels MFCCs oder Chroma-Vektoren) könnte genutzt werden, um die Diskrepanz zwischen geschriebenem Notentext und performativer Interpretation zu quantifizieren.

Das Projekt zeigt somit, dass Computational Musicology nicht nur als akademisches Hilfsmittel, sondern als Werkzeug zur Demokratisierung musikalischer Analyse dienen kann. Die Verbindung von mathematischer Präzision und musikalischer Intuition eröffnet neue Wege für das Verständnis unserer kulturellen Evolution.

7.2.1 Interaktive Weboberfläche

Für die explorative Analyse wurde eine interaktive Next.js-Weboberfläche entwickelt, die unter https://empirical-music.victorgurbani.com öffentlich zugänglich ist und Jury sowie interessierten Lesern ermöglicht, die Daten selbstständig zu erforschen. Die Haupt-Dashboard-Seite bietet nahtlosen Zugriff auf dynamische Karussells, welche die Pipeline-Architektur, animierte „Stylistic Evolution"-GIFs, statistische Signifikanz-Heatmaps und alle Machine-Learning-Visualisierungen bündeln.

Die spezialisierte „Subset Clouds"-Seite (https://empirical-music.victorgurbani.com/clouds) ermöglicht darüber hinaus:

- Auswahl beliebig vieler Komponisten-Kombinationen aus dem Feature-Cache, um eigene „Clouds" (PCA-Punktwolken) zu generieren und stilistische Nähe interaktiv in 3D zu überprüfen
- Echtzeit-Generierung neuer PCA-Wolken (canonical oder refit)
- Interaktive Suche und Analyse spezifischer Einzelstücke über die Suchfunktion auf der Startseite, um deren genaue Position im stilistischen Raum zu visualisieren
- Konfigurierbare Filter nach Komponist, Titel und Regex-Mustern
- Voreingestellte Gruppensets (Jazz, Spanische Nationalisten, Anime/Film, Pop/Modern)

Diese Oberfläche ermöglicht der Jury und Nicht-Programmierern, beliebige stilistische Vergleiche durchzuführen und die in dieser Arbeit vorgestellten Ergebnisse mit selbst gewählten Stücken oder Komponisten zu testen – ein Schritt zur Demokratisierung der Computational Musicology.

Um die Erreichbarkeit und globale Analysefähigkeit sicherzustellen, wurde diese Anwendung auf einen Oracle Linux Virtual Private Server (VPS) migriert, wobei serverseitige Fallbacks für Caching-Operationen integriert wurden (Details zur Server-Architektur und den technischen Deployment-Herausforderungen der Next.js-Umgebung finden sich in Anhang E).

Alle Skripte sind dokumentiert und ermöglichen reproduzierbare Folgeuntersuchungen (siehe Anhang B).

8 Quellen- und Literaturverzeichnis

Hinweis zum Zitierstil: Dieses Dokument verwendet durchgehend den IEEE-Stil als Standard in Mathematik und Informatik; vgl. [5].

[1] Arabesque Conservatory, *Claude Debussy's Musical Style: 5 Defining Characteristics*, [Online]. Verfügbar, 2025. Adresse: `https://www.arabesqueconservatory.com/blog/5-characteristic-traits-of-piano-music-by-debussy/` (siehe S. 2).

[2] M. S. Cuthbert und C. Ariza, „music21: A Toolkit for Computer-Aided Musicology," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010 (siehe S. 1, 2).

[3] M. Gołąb, „Transformations of Chopin's Style," *Polish Music Journal*, Jg. 3, Nr. 1, 2000, [Online]. Verfügbar. Adresse: `https://polishmusic.usc.edu/research/publications/polish-music-journal/vol3no1/transformations-of-chopin/` (siehe S. 2).

[4] G. Hadjeres, „Interactive Deep Generative Models for Symbolic Music," DeepBach: LSTM-basierte Bach-Stil-Generierung, Diss., Sorbonne Université (UPMC), 2018 (siehe S. 2).

[5] IEEE, *IEEE Reference Guide*, [Online]. Verfügbar, 2023. Adresse: `https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf` (siehe S.).

[6] E.-F. Lin-Jeng, „Stylistic analysis and recognition of piano sonatas of four composers: Mozart, Chopin, Debussy, and Anton Webern," Magisterarb., Rochester Institute of Technology, 1987 (siehe S. 1).

[7] P. Long, Z. Novack, J. McAuley und T. Berg-Kirkpatrick, *PDMX: A Large-Scale Public Domain MusicXML Dataset for Symbolic Music Processing*, arXiv preprint arXiv:2409.10831, Accepted to 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024. Adresse: `https://arxiv.org/abs/2409.10831` (siehe S. 1, 3, XIII).

[8] C. McKay und I. Fujinaga, „jSymbolic: A Feature Extractor for MIDI Files," *Proceedings of the International Computer Music Conference*, 2006, Entwicklung von jSymbolic für maschinelle Musikklassifikation (siehe S. 2).

[9] F. Simonetta, „Style-based Composer Identification and Attribution of Symbolic Music Scores: a Systematic Survey," *Transactions of the International Society for Music Information Retrieval*, 2025 (siehe S. 1).

[10] P. Von Hippel, „Redefining Pitch Proximity: Tessitura and Mobility as Constraints on Melodic Intervals," Frühes Beispiel statistischer Melodieanalyse, Diss., Stanford University, 2000 (siehe S. 2).

[11] C. W. White, „Some Statistical Properties of Tonality, 1650–1900," Großangelegte Korpusanalyse tonaler Entwicklung mit 8000+ Werken, Diss., Yale University, 2013 (siehe S. 2).

Angaben zu Unterstützungsleistungen

Die Konzeption des Projekts, die Literaturrecherche, die komplette Python-Programmierung (Korpus-Kuratierung, Parsing, Merkmalsextraktion, Statistik, Visualisierung) sowie die Auswertung und Dokumentation wurden von mir, Victor Gurbani, eigenständig erbracht. Verwendet wurden ausschließlich die in Abschnitt 8 genannten Open-Source-Bibliotheken und öffentlichen Datensätze.

A Vollständige Merkmalsdokumentation

Dieser Anhang dokumentiert alle 36 extrahierten Merkmale der drei Säulen. Jedes Merkmal wird durch seine musikologische Bedeutung, Berechnungsmethode und Interpretation erklärt.

A.1 Harmonische Merkmale (16)

Akkord-Basisdaten

`chord_event_count` Gesamtzahl der Akkordevents nach `chordify()`. Jedes Event repräsentiert eine simultane Sonorität über einen rhythmischen Abschnitt.

`chord_quality_total` Anzahl der Akkorde mit klassifizierbarer Qualität (Dur/Moll/vermindert/übermäßig

Akkordqualitäts-Verteilung

Alle Prozentsätze relativ zu `chord_quality_total`:

`chord_quality_major_pct` Anteil Dur-Akkorde. Hohe Werte charakterisieren heitere, stabile Harmonik.

`chord_quality_minor_pct` Anteil Moll-Akkorde. Erhöhte Werte signalisieren modale Mischung oder Moll-Tonikalisierung.

`chord_quality_diminished_pct` Anteil verminderter Klänge. Typisch für Leitton-Akkorde (vii°) vor Kadenzen.

`chord_quality_augmented_pct` Anteil übermäßiger Dreiklänge. Selten; meist chromatische Durchgangsharmonien.

`chord_quality_other_pct` Nicht-terzenbasierte Akkorde (Quart-/Sekundschichtungen, Vorhalte, Cluster). Hohe Werte bei Debussy (z. B. *Debussy: Prelude Book 2 No 12 Feux d'artifice*: 77,8%).

Textur und Konsonanz

`harmonic_density_mean` Durchschnittliche Anzahl verschiedener Tonklassen pro Akkord. Werte nahe 4 deuten auf erweiterte Harmonien hin (z. B. *Chopin: Étude Op. 10 Nr. 11*: 3,785).

`dissonance_ratio` Anteil dissonanter Akkorde nach `music21.isConsonant()`. Misst vertikale Spannungsfrequenz (z. B. *Debussy: Pour remercier la pluie au matin*: 0,845 vs. *Mozart: Tuba Mirum*: 0,000).

Nichtakkordische Töne

Bezogen auf die melodische Hauptstimme; Nenner ist `dissonant_note_count`:

`dissonant_note_count` Anzahl detektierter nichtakkordischer Töne.

`passing_tone_ratio` Anteil Durchgangsnoten (schrittweise Bewegung in gleicher Richtung, Auflösung in Akkordton).

appoggiatura_ratio Anteil Appoggiaturen (Sprung zur Dissonanz auf betonter Zeit, schrittweise Auflösung).

other_dissonance_ratio Verbleibende Dissonanzen (Wechselnoten, Antizipationen, freie Dissonanzen).

Römische Analyse

roman_chord_count Erfolgreich analysierte Akkorde mit römischen Ziffern.

deceptive_cadence_ratio Häufigkeit von Trugschlüssen (V–vi statt V–I).

modal_interchange_ratio Anteil modaler Mischklänge (z. B. bVI in Dur aus parallelem Moll; z. B. *Debussy: Petite suite*: 0,265).

A.2 Melodische Merkmale (11)

Registrale Merkmale

note_count Gesamtzahl melodischer Events (inkl. expandierter Akkordtöne). Beispiele: *Bach: chorale harmonisations 241–371* 30 349 Events; *Mozart: Thema K. 331* 55 Events.

pitch_range_semitones Ambitus in Halbtönen zwischen tiefster und höchster Note. Beispiele: *Mozart: Tuba Mirum* 22 Halbtöne; *Debussy: Feux d'artifice* 84 Halbtöne.

Intervallik

avg_melodic_interval Mittlere Intervallgröße (absolut, Halbtöne) aufeinanderfolgender Noten (z. B. *Chopin: Étude Op. 10 Nr. 11*: 8,99).

melodic_interval_std Standardabweichung der Intervallgrößen. Hohe Werte zeigen Wechsel zwischen Schritten und großen Sprüngen (Chopin).

conjunct_motion_ratio Anteil schrittweiser Intervalle ($\leq$Ganzton).

melodic_leap_ratio Anteil von Sprüngen (>Ganzton). Komplementär zu `conjunct_motion_ratio`.

Tonklassenverteilung

pitch_class_entropy Shannon-Entropie der 12-Ton-Verteilung (Basis 2). Misst chromatische Dichte (z. B. *Mozart: Thema K. 331*: 2,296 vs. *Chopin: 24 Préludes Op. 28 (Gesamt)*: 3,566).

Zweistimmige Interaktion

Sopran (oberstes System) vs. Bass (unterstes System):

voice_independence_index Netto-Balance zwischen Gegen- und Parallelbewegung: (contrary−parallel)/total (z. B. *Chopin: Prélude Op. 28 Nr. 16*: −1,0; *Debussy: Le petit nègre*: 0,667).

contrary_motion_ratio Anteil gegenläufiger Bewegung.

parallel_motion_ratio Anteil gleichgerichteter Bewegung.

oblique_motion_ratio Anteil oblique Bewegung (eine Stimme statisch).

A.3 Rhythmische Merkmale (9)

Dauer-Statistik

note_event_count Gesamtzahl rhythmischer Events.

avg_note_duration Mittlere Notendauer (Viertelnoten-Einheiten).

std_note_duration Standardabweichung der Dauern.

Metrische Aktivität

notes_per_beat Durchschnittliche Events pro Schlag (z. B. *Chopin: Étude Op. 25 Nr. 11 „Winter Wind“*: 14,06 vs. *Chopin: Nocturne n20*: 0,63).

downbeat_emphasis_ratio Anteil starker Schläge (beatStrength $\geq 0{,}75$).

syncopation_ratio Anteil synkopischer Einträge (schwache Zeit, übergebunden über nächsten Schlag; z. B. *Debussy: Nocturne*: 0,211; *Bach: Chorale harmonisations (Compilation)*: 0,356).

Muster-Komplexität

rhythmic_pattern_entropy Shannon-Entropie von Dauer-Trigrammen. Hohe Werte zeigen diverse rhythmische Zellen (z. B. *Debussy: Nocturne*: 6,90 vs. *Chopin: Étude Op. 25 Nr. 8*: 0,34).

micro_rhythmic_density Anteil 4-Noten-Fenster, in denen $\geq$3 Noten schnell sind ($\leq$Sechzehntel) (z. B. *Bach: Prelude BWV 1006*: 0,984).

Polyrhythmik

cross_rhythm_ratio Anteil der Takte, in denen obere und untere Systeme verschiedene Dauernnenner verwenden (z. B. *Debussy: Feuilles mortes*: 1,0).

A.4 Zusammenfassung der Merkmals-Interpretation

Die 36 Merkmale erfassen komplementäre Dimensionen musikalischer Gestaltung:

- **Harmonik**: Erfasst vertikale Strukturen, Konsonanz/Dissonanz-Balance, nichtakkordische Verzierungen und tonale Funktionen. Debussys hohe other_pct und modal_interchange_ratio quantifizieren seine harmonische Innovativität.
- **Melodik**: Misst registrale Spannweite, Intervallprofile und Stimmführungstypen. Chopins Brückenfunktion manifestiert sich in erhöhter pitch_class_entropy bei gleichzeitiger Bewahrung von contrary_motion_ratio-Werten nahe Mozart.

- **Rhythmik**: Quantifiziert metrische Organisation, synkopische Spannung und polyrhythmische Schichtung. Debussys extreme `rhythmic_pattern_entropy` und `cross_rhythm_ratio` belegen seine rhythmische Vielfalt.

Diese vollständige Dokumentation ermöglicht die Replikation und Erweiterung der Analyse auf neue Komponisten oder Repertoires.

B Reproduzierbarkeits-Leitfaden

Dieser Anhang dokumentiert die vollständige Software-Pipeline zur exakten Replikation aller Ergebnisse. Alle Befehle und Skripte sind im Projekt-Repository verfügbar.

B.1 Pipeline-Architektur

Abbildung 10 zeigt den vollständigen Datenfluss von den Rohdaten bis zur Visualisierung.

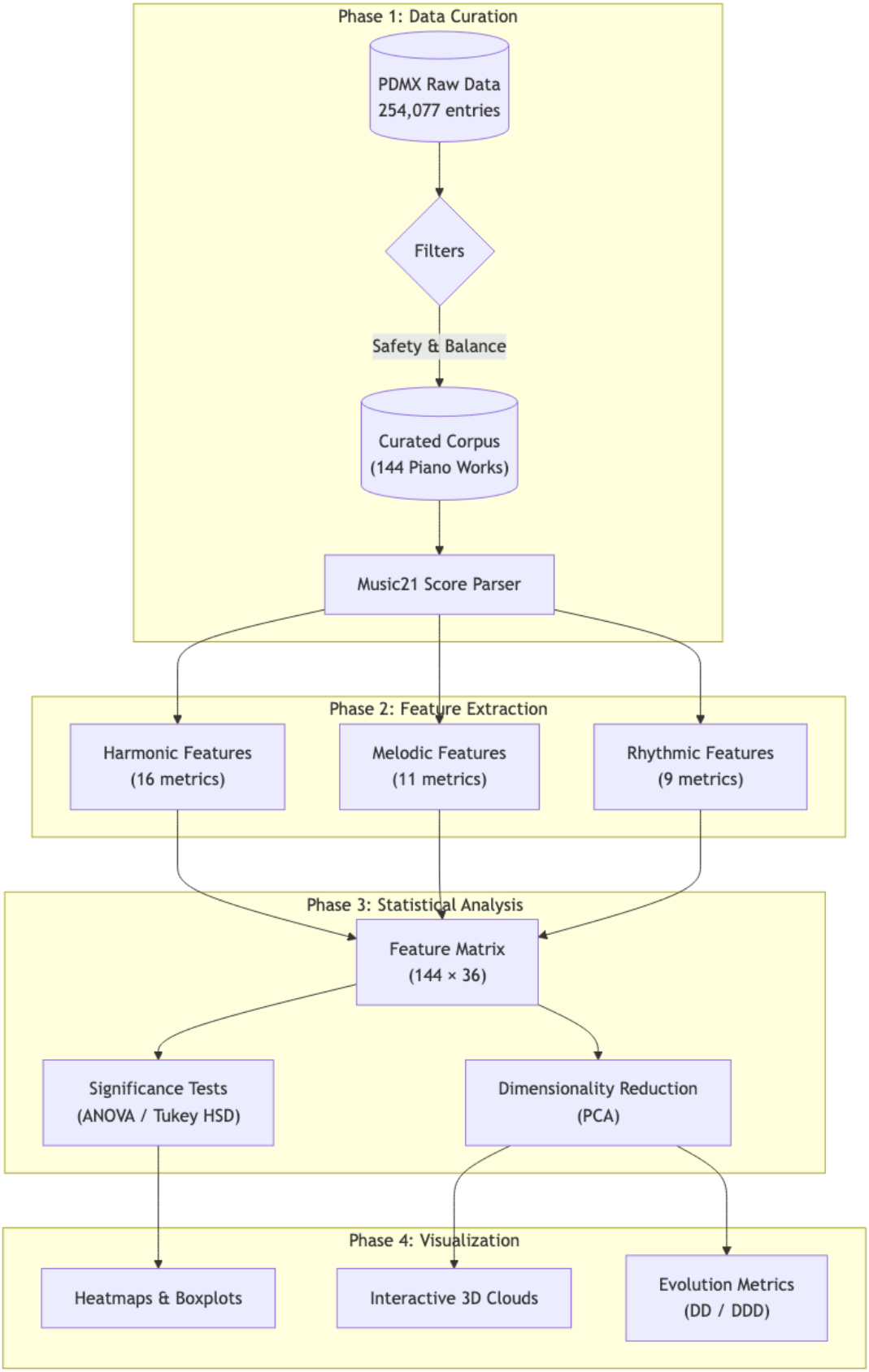

Abbildung 10: Systemarchitektur der Analyse-Pipeline. Phase 1 filtert 254 077 PDMX-Einträge auf 144 balancierte Klavierwerke. Phase 2 extrahiert 36 Merkmale in drei Kategorien. Phase 3 führt statistische Tests durch. Phase 4 erzeugt interaktive Visualisierungen und Evolutionsmetriken.

B.2 Systemvoraussetzungen und Ressourcenbedarf

- **Betriebssystem**: macOS, Linux oder Windows mit WSL
- **Python**: Version 3.10 oder höher
- **Speicherplatz**:
  - PDMX-Datensatz: ∼56 GB (254 077 MusicXML-Partituren)
  - Python Virtual Environment: ∼600 MB
  - Zwischenergebnisse (Features, Statistiken, Visualisierungen): ∼50 MB
- **Rechenzeit**: 2–3 Stunden für vollständigen Durchlauf auf modernem Laptop
- **RAM**: Mindestens 8 GB empfohlen (16 GB für große Bach-Choralkompilationen)

B.3 Automatisierte Installation mit Quickstart-Skript

Das Projekt enthält ein vollautomatisches Bash-Skript, das Installation und Analyse in einem Durchlauf erledigt.

B.3.1 Vollautomatischer Ablauf

```
./quickstart.sh
```

Das Skript führt automatisch folgende Schritte aus:

1. **Abhängigkeitsprüfung**: Validiert Python-Installation und `requirements.txt`
2. **Datensatz-Download**: Bietet interaktiven Download von Zenodo (15571083) an, falls PDMX-Verzeichnis fehlt. Unterstützt `aria2c` für parallele Downloads (16 Verbindungen) mit Fallback auf `wget`
3. **Virtual Environment**: Erstellt isolierte Python-Umgebung im `venv/`-Verzeichnis
4. **Dependency-Installation**: Installiert `music21`, `pandas`, `numpy`, `scipy`, `scikit-learn`, `matplotlib`, `seaborn`, `plotly`
5. **Pipeline-Ausführung**: Führt alle 10 Analyseschritte sequenziell aus (siehe unten)

Alternative ohne Virtual Environment:

```
./quickstart.sh --no-venv
```

Nutzt System-Python-Installation (für conda-Nutzer oder vorinstallierte Abhängigkeiten).

B.4 Die 10-Stufen-Analysepipeline

Das Quickstart-Skript orchestriert folgende Schritte:

B.4.1 Stufe 1–2: Korpus-Kuratierung und Parsing

[1/10] Korpus-Kuratierung

```
python3 src/corpus_curation.py --min-rating 0
```

Ausgabe: `data/curated/solo_piano_corpus.csv` (144 Werke), `solo_piano_mxl_paths.txt`

[2/10] Struktur-Parsing

```
python3 src/score_parser.py \
    --output data/parsed/summaries.json
```

Ausgabe: JSON mit Taktzahl, Stimmenzahl, Dauern pro Partitur

B.4.2 Stufe 3–5: Merkmals-Extraktion

[3/10] Harmonische Merkmale

```
python3 src/harmonic_features.py \
    --output-csv data/features/harmonic_features.csv
```

Ausgabe: 16 harmonische Deskriptoren + Boxplots in `figures/harmonic/`

[4/10] Melodische Merkmale

```
python3 src/melodic_features.py \
    --output-csv data/features/melodic_features.csv
```

Ausgabe: 11 melodische Deskriptoren + Boxplots in `figures/melodic/`

[5/10] Rhythmische Merkmale

```
python3 src/rhythmic_features.py \
    --output-csv data/features/rhythmic_features.csv
```

Ausgabe: 9 rhythmische Deskriptoren + Boxplots in `figures/rhythmic/`

B.4.3 Stufe 6: Dimensionsreduktion und Embedding

[6/10] PCA- und t-SNE-Projektionen

```
# PCA mit 3D/2D-Visualisierung
python3 src/feature_embedding.py --method pca \
    --output figures/embeddings/pca_3d.html \
    --output-2d figures/embeddings/pca_2d.html \
    --loadings-csv data/stats/pca_loadings.csv \
    --clouds-output figures/embeddings/pca_clouds_3d.html \
    --clouds-output-2d figures/embeddings/pca_clouds_2d.html

# t-SNE mit angepassten Hyperparametern
python3 src/feature_embedding.py --method tsne \
    --perplexity 20 --tsne-composer-weight 4.0 \
    --output figures/embeddings/tsne_3d.html \
```

```
    --clouds-output figures/embeddings/tsne_clouds_3d.html
```

Ausgabe: Interaktive HTML-Scatter-Plots, Gaußsche Dichtewolken, PCA-Loadings

B.4.4 Stufe 7–8: Statistische Signifikanz

[7/10] ANOVA und Tukey-HSD-Tests

```
python3 src/significance_tests.py \
    --anova-output data/stats/anova_summary.csv \
    --tukey-output data/stats/tukey_hsd.csv
```

Ausgabe: F-Statistiken, p-Werte, Bonferroni/FDR-Flags, paarweise Vergleiche

[8/10] Signifikanz-Visualisierungen

```
python3 src/significance_visualizations.py --top-n 15
```

Ausgabe: Bar-Charts, Heatmaps (Paar-Counts, Mean-Diff., Sym-Log) in `figures/significance/`

B.4.5 Stufe 9–10: Annotation und Aggregation

[9/10] Annotierte Referenzpartituren

```
python3 src/generate_selected_annotations.py
```

Ausgabe: 8 farbcodierte MusicXML-Dateien (2 pro Komponist) mit Dissonanz-Labels, Akkordsymbolen und optionalen PDF-Renderings

[10/10] Master-Aggregation

```
python3 src/aggregate_metrics.py
```

Validiert alle Outputs, druckt Korpus-Statistiken, verifiziert Konsistenz

B.5 Manuelle Einzelschritte und Debugging

B.5.1 Schnelltests mit `-limit`

Jedes Feature-Extraktions-Skript unterstützt partielle Läufe:

```
python3 src/harmonic_features.py --limit 5 \
    --output-csv /tmp/harmonic_test.csv
```

Prozessiert nur die ersten 5 Partituren für Smoke-Tests.

B.5.2 Cached Runs mit `-features-from`

Wiederverwendung bereits berechneter Features ohne Neuberechnung:

```
python3 src/harmonic_features.py \
    --features-from data/features/harmonic_features.csv \
    --skip-plots
```

Nutzt existierende CSV, erzeugt nur Statistiken (nützlich für iterative Visualisierungen).

B.5.3 Alternative Korpus-Varianten

Experimentiere mit unbalancierten oder entspannten Filtern:

```
python3 src/corpus_curation.py --min-rating 0 \
    --skip-license-filter \
    --max-per-composer 999 \
    --output-csv data/curated/unbalanced_corpus.csv
```

B.6 Erweiterte Funktionen

B.6.1 Externe Stück-Projektion

Projiziere beliebige MusicXML-Dateien in den PCA-Raum:

```
python3 src/highlight_pca_piece.py \
    --pieces path/to/score.mxl \
    --title "Joe Hisaishi - One Summer's Day" \
    --composer "Joe Hisaishi" \
    --output highlighted_projection.html
```

Unterstützt mehrere Stücke gleichzeitig mit individuellen Labels.

B.6.2 MusicXML-Annotation mit MuseScore-Rendering

Erzeuge farbcodierte Analysepartituren mit automatischem PDF-Export:

```
python3 src/annotate_musicxml.py \
    path/to/input.mxl \
    path/to/output_annotated.mxl \
    --renderer-template "mscore -o {output} {input}" \
    --render-format pdf
```

Färbt Durchgänge (orange), Appoggiaturen (violett), andere Dissonanzen (rot), chromatische Harmonien (türkis); fügt römische Ziffern als Akkordsymbole ein.

B.7 Validierung der Installation

B.7.1 Erwartete Ergebnisse

Nach vollständigem Durchlauf sollten folgende Kennzahlen reproduziert werden:

- **Korpusgröße**: Exakt 144 Partituren (36 pro Komponist)
- **Gesamtdauer**: 71 585 Viertelnoten (~13,3 Stunden bei 90 BPM)
- **ANOVA-Treffer** ($\alpha = 0.05$): 29 Merkmale
- **FDR-robuste Merkmale** ($q < 0.05$): 29 Merkmale

- **PCA-Varianz (PC1–PC3)**: 48,2% (22,3% + 16,1% + 9,8%)
- **Tukey-Unterschiede Debussy–Mozart**: 16 signifikante Merkmale
- **Tukey-Unterschiede Bach–Mozart**: 10 signifikante Merkmale (bzw. 16 inklusive Zähl-/Größenmetriken)

B.7.2 Schnelle Integritätsprüfung

```
# Test 1: Parsing-Integrität
python3 src/score_parser.py --limit 5 \
    --output /tmp/test_parse.json
# Erwartung: 5 erfolgreiche Parses

# Test 2: Feature-Extraktion
python3 src/harmonic_features.py --limit 5 \
    --output-csv /tmp/test_harmonic.csv
# Erwartung: 5 Zeilen mit 16 Harmonie-Spalten

# Test 3: Statistik-Pipeline
python3 src/significance_tests.py --alpha 0.05
# Erwartung: anova_summary.csv mit 36 Zeilen
```

B.8 Häufige Probleme und Lösungen

`music21` Parse-Fehler bei manchen Partituren Einige MusicXML-Dateien enthalten Formatfehler oder sind Opus-Kompilationen. **Lösung**: Alle Skripte verwenden standardmäßig `-skip-errors`, um fehlerhafte Dateien zu überspringen. Für Debugging: `-no-skip-errors` aktivieren.

Matplotlib Backend-Fehler auf Headless-Systemen Server ohne Display benötigen nicht-interaktives Backend. **Lösung**: `export MPLBACKEND=Agg` vor Ausführung setzen.

Speicherprobleme bei Bach-Chorälen Einige Bach-Werke enthalten 100+ Choräle. **Lösung**: `--limit` Parameter nutzen oder RAM auf min. 16 GB erhöhen.

Abweichende PCA-Ergebnisse trotz `random_state=42` Unterschiedliche `scikit-learn`-Versionen können minimal andere Rotationen erzeugen. **Lösung**: Für exakte Replikation Versionen pinnen (z. B. via `pip freeze > requirements-lock.txt`) und die Analyse in einer frischen Umgebung ausführen.

Zenodo-Download schlägt fehl API-Rate-Limits oder Netzwerkprobleme. **Lösung**: Manueller Download von `https://zenodo.org/records/15571083`, Extraktion nach `15571083/`.

B.9 Datenformat-Spezifikationen

B.9.1 Korpus-CSV (`data/curated/solo_piano_corpus.csv`)

- `composer_label`: Normalisiert (Bach | Mozart | Chopin | Debussy)

- `title`: Werktitel aus PDMX-Metadaten
- `mxl`: Relativer Pfad zu MusicXML-Datei (z. B. `0/46/Qmaug5p...mxl`)
- `rating`: PDMX-Qualitätsbewertung (0.0–5.0)
- `subset:*`: Boolsche Flags für PDMX-Subsets

B.9.2 Feature-CSVs (`data/features/*.csv`)

Einheitliches Format für alle drei Feature-Typen:

- `composer`: Komponistenname (für Gruppierung in ANOVA)
- `title`: Werktitel
- Feature-Spalten: Numerische Werte (NaN bei Extraktionsfehlern)

B.9.3 ANOVA-Ergebnisse (`data/stats/anova_summary.csv`)

- `feature`: Merkmalsname
- `F_statistic`: ANOVA-F-Wert
- `p_value`: Nicht-adjustierter p-Wert
- `bonferroni_sig`: Bonferroni-Signifikanz (nach $\alpha/36$)
- `fdr_sig`: Benjamini-Hochberg FDR-Signifikanz ($q < 0.05$)

B.9.4 Tukey-Ergebnisse (`data/stats/tukey_hsd.csv`)

- `feature`: Merkmalsname
- `group1`, `group2`: Komponistenpaar
- `meandiff`: Mittlere Differenz
- `lower`, `upper`: 95%-Konfidenzintervall
- `reject`: Signifikanz (True/False)

C Online-Links und Indexseiten

Alle Abbildungen im PDF sind klickbar und führen zu den interaktiven Online-Versionen. Zur Sicherheit sind alle Links hier gesammelt aufgeführt (für Leserinnen und Leser ohne PDF-Link-Unterstützung).

C.1 Index-Startseiten

- **Gesamtindex der Abbildungen**:
  https://victor-gurbani.github.io/JuFo2026/figures/
- **Embeddings (PCA/t-SNE)**:
  https://victor-gurbani.github.io/JuFo2026/figures/embeddings/
- **Highlights (externe Stücke)**:
  https://victor-gurbani.github.io/JuFo2026/figures/highlights/
- **Signifikanz-Plots**:
  https://victor-gurbani.github.io/JuFo2026/figures/significance/
- **Harmonische Boxplots**:
  https://victor-gurbani.github.io/JuFo2026/figures/harmonic/
- **Melodische Boxplots**:
  https://victor-gurbani.github.io/JuFo2026/figures/melodic/
- **Rhythmische Boxplots**:
  https://victor-gurbani.github.io/JuFo2026/figures/rhythmic/
- **Annotierte MusicXML-Dateien**:
  https://victor-gurbani.github.io/JuFo2026/figures/annotated/

C.2 Direktlinks zu interaktiven Kernansichten

- **PCA-Komponistenwolken (3D)**:
  https://victor-gurbani.github.io/JuFo2026/figures/embeddings/composer_clouds_3d.html
- **Highlight: One Summer's Day (Hisaishi)**:
  https://victor-gurbani.github.io/JuFo2026/figures/highlights/one_summers_day_pca_cloud.ht
- **Highlight: Ravel String Quartet (F-Dur)**:
  https://victor-gurbani.github.io/JuFo2026/figures/highlights/ravel_string_quartet_pca_clo

C.3 Software-Lizenzierung und Zitierung

- **PDMX-Datensatz**: Public Domain, verfügbar unter `https://zenodo.org/records/15571083`. Zitierung: [7]
- **Repository**: github.com/victor-gurbani/JuFo2026
- **Analysecode**: Siehe `LICENSE`-Datei im Repository

- **Extrahierte Features**: Abgeleitete Daten, frei verwendbar für akademische Zwecke unter Nennung dieser Arbeit
- **Dependencies**: Alle verwendeten Bibliotheken (`music21`, `scikit-learn`, etc.) sind Open Source (BSD/MIT-Lizenzen)

Dieser Leitfaden ermöglicht vollständige Reproduktion aller Ergebnisse sowie deren Erweiterung auf neue Fragestellungen. Bei Problemen: GitHub-Issues im Repository oder Kontakt über verfügbare Adresse.

D Erweiterte Details zur Machine-Learning-Pipeline

Um die Transparenz und Reproduzierbarkeit der durchgeführten Klassifikationsexperimente (Random Forest und Neuronales Netz) weiter zu erhöhen, werden in diesem Anhang tiefere Einblicke in die Hyperparameter-Optimierung, die Baumstrukturen und die lokalen SHAP-Erklärungen gegeben.

D.1 Vergleich mit neuronalen Netzen (MLP)

Parallel zum Random Forest wurde ein *Multi-Layer Perceptron* (MLP) trainiert. Dieses neuronale Netz durfte den vollen Merkmalsraum (30 Features) ohne Vorfilterung oder Komplexitätsbeschränkung nutzen. Es diente als Referenz für das theoretisch erreichbare Maximum an Genauigkeit auf dem 144-Stücke-Datensatz.

Das Ergebnis war überraschend: Der interpretierbare RF schnitt mit 70,1 % Genauigkeit leicht besser ab als das unbeschränkte neuronale Netz (68,7 %; vgl. Abbildung 11). Wir interpretieren diesen Befund dahingehend, dass die Kombination aus Kollinearitätsfilterung und striktem Pruning als effektive Regularisierung wirkte. Während das neuronale Netz auf dem relativ kleinen Datensatz (144 Werke) zu leichtem Overfitting neigte, zwang die interpretierbare Architektur das Modell dazu, sich auf die robustesten musikalischen Merkmale – wie den Tonumfang und die harmonische Dichte – zu konzentrieren (Abbildung 7).

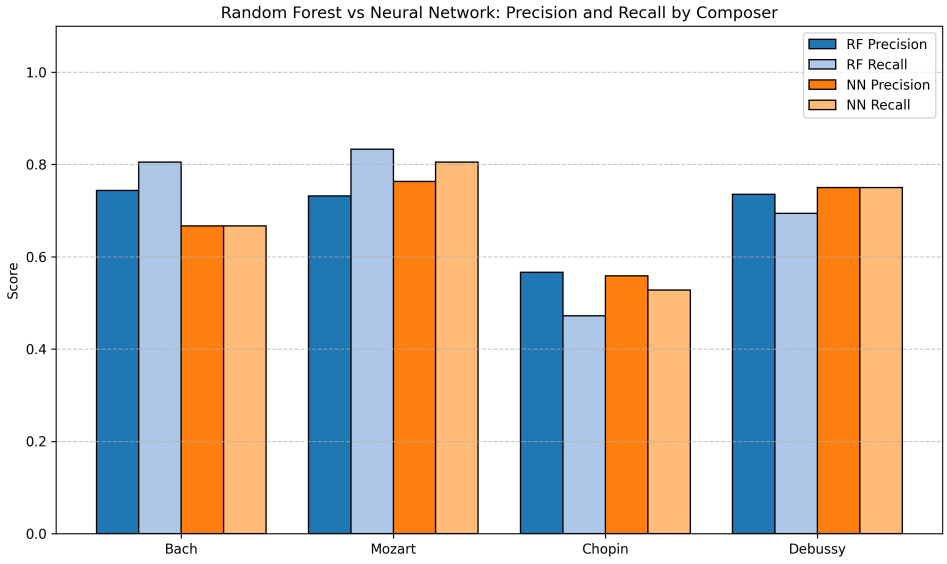

Abbildung 11: Vergleich der Klassifikationsleistung: Der interpretierbare Random Forest (RF) erzielt trotz seiner Komplexitätsbeschränkung eine leicht höhere Genauigkeit als das neuronale Netz (MLP). Die Filterung redundanter Merkmale fungiert hier als effiziente Regularisierung.

D.2 Hyperparameter-Optimierung und Pruning-Statistiken

Das Training des interpretierbaren Random Forests zielte auf eine maximale Reduktion der Komplexität ab, um „White-Box“-Eigenschaften zu bewahren, ohne die Genauigkeit drastisch zu opfern. Die Optimierung erfolgte mittels `RandomizedSearchCV` (5-Fold Cross-Validation) über folgendes Raster:

- `n_estimators`: [100, 200, 300]
- `max_depth`: [5, 10, 15, 20]
- `min_samples_leaf`: [2, 3, 4]
- `ccp_alpha`: [0.005, 0.01, 0.015, 0.02, 0.03]

Um den konkreten Effekt des *Cost-Complexity Prunings* (`ccp_alpha`) zu quantifizieren, wurde ein komplett unbeschränkter Basis-Forest mit dem optimierten, beschnittenen Forest verglichen.

- **Unpruned Baseline**: 73,61 % Genauigkeit, ∅ Baumtiefe: 7,90, Gesamtzahl der Knoten: 5.094
- **Best Pruned Model** (`ccp_alpha=0.01`): 70,14 % Genauigkeit, ∅ Baumtiefe: 6,66, Gesamtzahl der Knoten: 3.270

Der Verzicht auf nur ∼3,5 % Genauigkeit führte zu einer massiven Reduzierung der Modellkomplexität um **35,8 % weniger Entscheidungsknoten**.

D.3 Beispielhafter Entscheidungsbaum

Ein Random Forest trifft Vorhersagen durch einen Mehrheitsentscheid vieler individueller Bäume. Um die Nachvollziehbarkeit zu demonstrieren, wurde der mathematisch akkurateste Einzelbaum aus dem 100-Bäume-Ensemble (Baum #71) extrahiert. Dieser Baum erreichte alleingestellt eine Genauigkeit von 72,4 % auf dem Testset. Zur Gewährleistung uneingeschränkter Transparenz („White-Box“-Paradigma) exportiert die Pipeline zudem alle 100 Entscheidungsbäume vollständig als grafische und textuelle Regelwerke zur manuellen Auditierung.

D.4 Komponistenspezifische SHAP-Analysen

Während einzelne Entscheidungsbäume eine *Mikro-Logik* zeigen, aggregieren SHAP-Werte (SHapley Additive exPlanations) die spieltheoretischen Einflüsse aller Merkmale über das *gesamte* Ensemble. Da die Merkmale im Vorfeld mittels Spearman-Clustering von Kollinearität befreit wurden, zeigen diese Plots die isoliertesten und aussagekräftigsten Unterscheidungsmerkmale. Dass diese sich teilweise von den Top-Merkmalen der ANOVA/PCA unterscheiden, liegt methodisch in der Natur des Modells: Während die statistischen Tests lineare, isolierte Varianzen messen, erfasst der Random Forest *nicht-lineare, konditionale Interaktionen* (z. B. „Wenn Merkmal X niedrig ist, trennt Merkmal Y perfekt“) und optimiert ausschließlich auf die Klassifikationsgrenze statt auf die globale Datenvarianz.

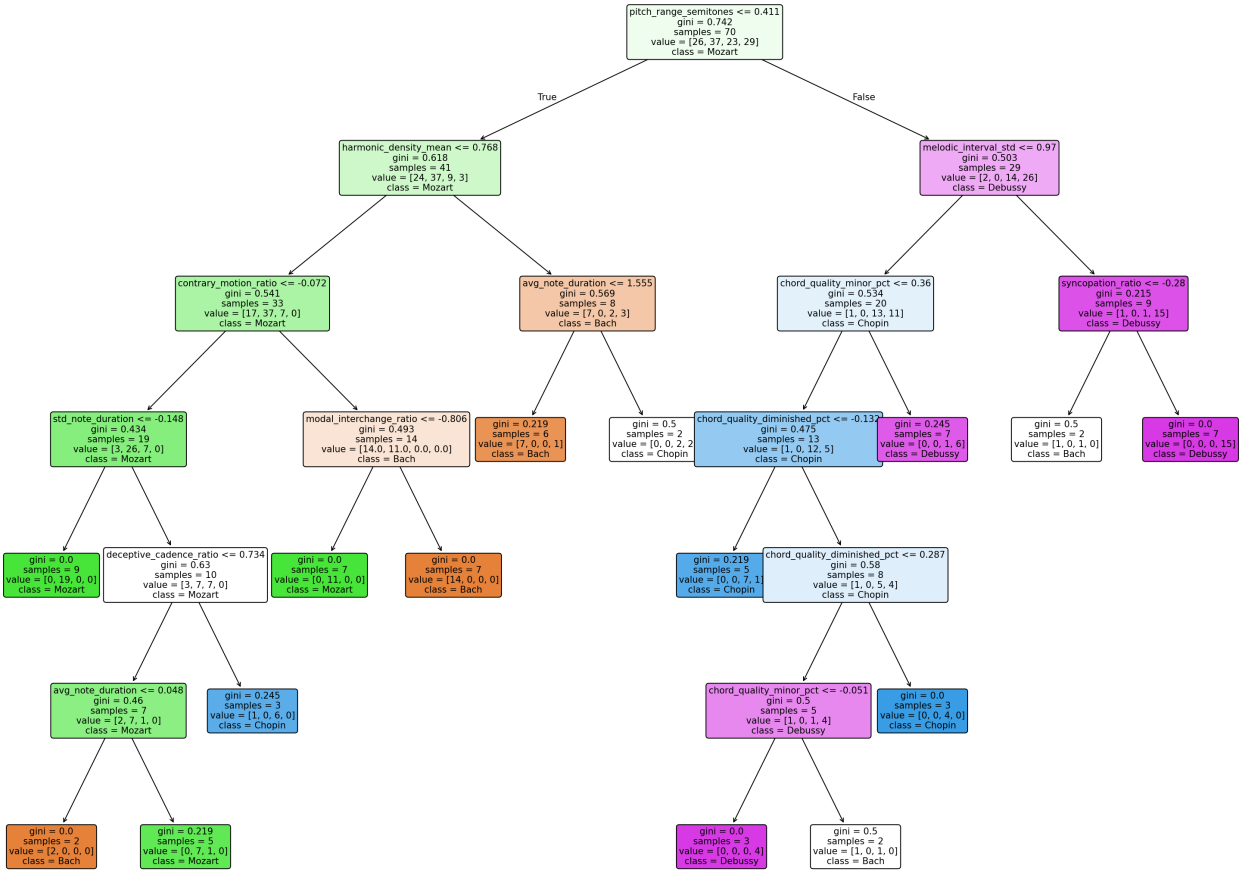

Abbildung 12: Der akkurateste Einzelbaum des Random-Forest-Ensembles. Jeder Knoten zeigt das Entscheidungskriterium (Merkmal), die Gini-Impurity (Grad der Mischung), die verbleibende Anzahl von Stücken (Samples) und deren Verteilung auf die vier chronologischen Klassen: [Bach, Mozart, Chopin, Debussy].

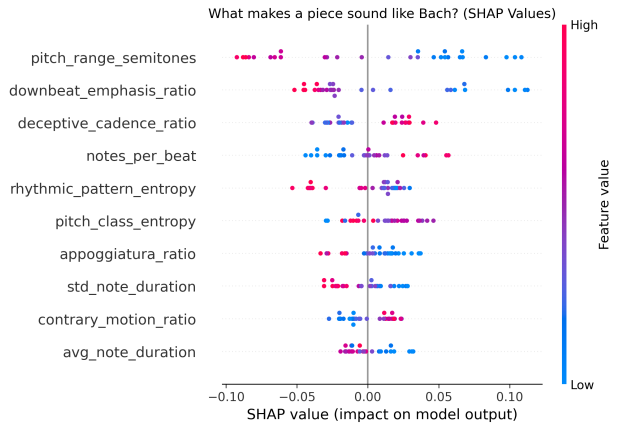

Abbildung 13: SHAP-Werte für Johann Sebastian Bach. Niedrige Werte (blau) beim Tonumfang (`pitch_range_semitones`) und der Downbeat-Betonung (`downbeat_emphasis_ratio`) führen zu positiven SHAP-Werten (rechts der vertikalen Linie) und erhöhen somit die Wahrscheinlichkeit für Bach. Im Gegensatz dazu deuten hohe Werte (rot) bei Trugschlüssen (`deceptive_cadence_ratio`) und den Noten pro Taktschlag (`notes_per_beat`) stark auf Bach hin.

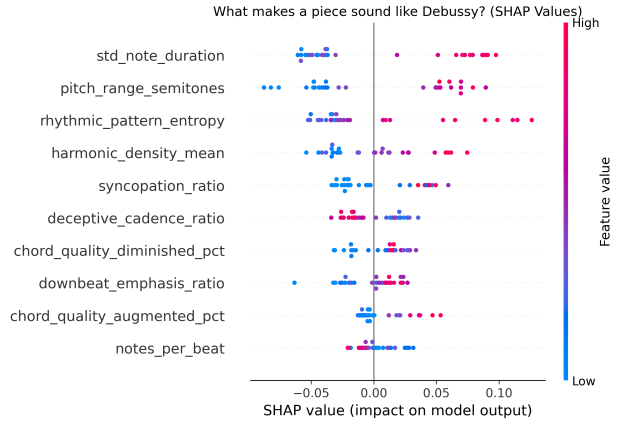

Abbildung 14: SHAP-Werte für Claude Debussy. Hohe Werte (rot) bei der rhythmischen Varianz (`std_note_duration`), dem Tonumfang (`pitch_range_semitones`), der rhythmischen Entropie (`rhythmic_pattern_entropy`) und der harmonischen Dichte (`harmonic_density_mean`) erzeugen positive SHAP-Werte (rechts der vertikalen Linie) und sind starke Indikatoren für Debussy. Umgekehrt sprechen niedrige Werte (blau) bei der Trugschlussrate (`deceptive_cadence_ratio`) eher für Debussy.

E Deployment und Web-Infrastruktur

Die Überführung der lokalen Next.js-Datenvisualisierung in eine öffentliche Web-Anwendung erforderte die Überwindung spezifischer infrastruktureller und softwarearchitektonischer Hürden. Die Applikation (gehostet auf https://empirical-music.victorgurbani.com) läuft auf einem ressourcenlimitierten Oracle Linux Virtual Private Server (VPS) unter Apache 2.4 und PM2. Die Beschränkungen in Speicher und Arbeitsspeicher setzten signifikante Optimierungen des Build-Prozesses voraus.

E.1 Limitierungen der Standalone-Kompilierung

Typischerweise verarbeitet das Next.js-Build-System (`Turbopack`) serverseitige Abhängigkeiten, indem es die im Quellcode verknüpften Ordnerstrukturen analysiert und in die endgültige Build-Datei integriert. In diesem Projekt operiert das Dashboard jedoch eng verzahnt mit dem Python-Backend und seinen Datensätzen (über 40 GB an MusicXML-Dateien, extrahierten Features und gepickelten Modellen). Da Turbopack Aufrufe wie `path.join(repoRoot, rawPath)` für die serverseitigen API-Routen erfasste, versuchte der Compiler, diese gesamten 40 GB als potenziell erforderliche serverseitige Fallback-Assets in den lokalen Build-Ordner (`.next/standalone`) zu kopieren. Dies stürzte in Schleifen ab (ENOSPC: No space left on device) oder überlastete den Arbeitsspeicher massiv.

Die Lösung war die Implementierung untrackbarer String-Interpolations-Polymorphismen in Next.js: Anstelle direkter standardisierter Bibliotheksaufrufe wertet das System die Suchpfade nun dynamisch über Laufzeit-Strings (`const joinPath = (base, ...parts) => [base, ...parts].join(path.sep)`) aus. Dadurch kompilierte Next.js in einen schmalen 50 MB-Microserver, ohne die Datensatz-Binärdateien einzubetten.

E.2 Abruf dynamischer Cloud-Jobs

Weitere Hürden betrafen die dynamische Darstellung frisch berechneter PCA-Wolken. Im `standalone`-Modus „friert“ der interne Node.js-Server von Next.js seinen statischen Datei-Router (für Assets innerhalb von `/public`) zum Zeitpunkt der Kompilierung ein. Starten Besucher der Webseite jedoch im Hintergrund einen Python-Subprozess, um eine neue Datensatzwolke zu berechnen, wurden die frisch generierten Metadaten (`.json` und `.html`) unter `public/generated/clouds/` vom Standalone-Server als 404 (Not Found) blockiert. Um dies zu umgehen, wurde in der Apache-Konfiguration ein Reverse-Proxy-Bypass integriert. Apache fängt Anfragen, die sich an die Verzeichnisse `/img` oder `/generated` richten (`Alias /generated /var/www/jufo2026/web-interface/public/generated`), explizit ab und serviert die frisch verfassten Dokumente nativ aus dem VPS-Dateisystem, während alle React- und API-spezifischen Endpunkte an den PM2-gewrappten Next.js-Dienst auf Port 3011 weitergeleitet werden.

E.3 Das Caching-Resolver-Dilemma

Schlussendlich trat das Problem der Interoperabilität von Dateipfaden zwischen macOS-Entwicklungssysteme und Oracle-Linux-Hosting auf. Das Backend speicherte Caches mit absoluten Unix-Dateipfaden der MusicXML-Dateien, um Dimensionsreduktions-Metadaten effizienter heranzuziehen. Da die korrespondierenden MusicXML-Dateien aus Speicher-Gründen nicht auf den Server mitkopiert

wurden (`-forceCache`-Flag überschreibt den Verzeichnis-Check), litten die Identifizierungs-Signaturen im Skript `embedding_cache.py` unter fehlschlagenden Zuordnungen. Durch den Umbau der Abgleichoperation mithilfe von `Path(mxl_abs_path).name.endswith(...)` wurde gewährleistet, dass das Dashboard auch dann voll funktionsfähig agiert und Einzelstücke fehlerfrei visualisiert rekonstruieren kann, wenn nur der Feature-Cache der Pipeline lokalisiert wird, selbst in Cross-OS-Szenarien ohne korrespondierenden Rohdatensatz.

F Multicore-Caching-Infrastruktur

Um Analysen auf das vollständige PDMX-Archiv (254 077 Partituren) zu ermöglichen, wurde ein skalierbares Caching-System entwickelt (`src/embedding_cache.py`). Dieses System unterstützt:

- **Parallelisierte Feature-Extraktion**: Die Option `-workers N` verteilt die Berechnung auf N CPU-Kerne. Bei 8 Kernen beschleunigt sich die Verarbeitung um den Faktor ≈6–7 (limitiert durch I/O).
- **Unterbrechungsresistenz**: Der Fortschritt wird zeilenweise in die Cache-CSV geschrieben und in einer Sidecar-Datei (`*.done.txt`) protokolliert. Mit `-resume` kann eine unterbrochene Berechnung jederzeit fortgesetzt werden – ideal für tagelange Läufe auf großen Korpora.
- **Vollständiger Feature-Cache**: Neben den 3D-PCA-Koordinaten speichert das System optional alle 36 Feature-Werte pro Partitur (`-output-features-csv`). Dies ermöglicht *instant subset-specific PCA*: Für beliebige Komponisten-Kombinationen kann die PCA ohne erneutes Parsen der MusicXML-Dateien sofort neu berechnet werden.

Die Cache-Integrität wird durch SHA256-Hashes der Eingabe-Feature-CSVs und JSON-Metadaten gewährleistet. Ein vollständiger Cache des PDMX-Korpus (alle 254 077 Partituren × 36 Features) belegt etwa 180 MB und ermöglicht Echtzeit-Exploration beliebiger Komponisten-Subsets über die Weboberfläche (siehe Abschnitt 7.2.1).