

# Data Science Projekt – Universität Stuttgart

*Hristo Baldzhiyski*

## Einleitung und Motivation des Projekts

Digitale Musikplattformen wie Spotify generieren sehr große, heterogene und hochdimensionale Datensätze, die Informationen über Musikstücke, Künstler, Alben sowie akustische Eigenschaften enthalten. Diese Daten bieten ein erhebliches Potenzial für datengetriebene Analysen, etwa zur Untersuchung von Popularität, musikalischen Trends oder zur Entwicklung prädiktiver Modelle. Gleichzeitig stellen sie hohe Anforderungen an Datenqualität, Reproduzierbarkeit und methodische Sauberkeit, da fehlerhafte Annahmen oder inkonsistente Datenverarbeitung zu verzerrten Ergebnissen führen können.

Ziel dieses Projekts ist es, einen großen Spotify-Datensatz systematisch zu analysieren und daraus **robuste, reproduzierbare und modellierbare Datenstrukturen** zu erzeugen. Der Fokus liegt dabei nicht allein auf der Modellleistung, sondern explizit auf einem **wissenschaftlich sauberen Analyseprozess**, der von der Datenauswahl über die Bereinigung bis hin zur Modellierung und Feinabstimmung nachvollziehbar dokumentiert ist.

Ein zentrales methodisches Problem besteht in der Größe des Datensatzes: Eine vollständige Analyse aller verfügbaren Daten wäre für interaktive Exploration ineffizient und schwer reproduzierbar. Gleichzeitig sollen die gewonnenen Erkenntnisse nicht zufällig oder nur für eine einzelne Stichprobe gelten. Dieses Spannungsfeld zwischen Skalierbarkeit, Reproduzierbarkeit und inhaltlicher Aussagekraft prägt den Aufbau der gesamten Analysepipeline.

## Gesamtstruktur der Analysepipeline

Um diesen Anforderungen gerecht zu werden, ist das Projekt in **sechs logisch aufeinander aufbauende Notebooks** gegliedert. Jedes Notebook hat ein klar abgegrenztes Ziel und erzeugt definierte Artefakte, die als Input für die folgenden Schritte dienen. Diese modulare Struktur reduziert Komplexität, erhöht Transparenz und erleichtert die wissenschaftliche Nachvollziehbarkeit.

Die sechs Notebooks bilden gemeinsam eine durchgängige Pipeline von der Rohdatenexploration bis zur robusten Modellbewertung.

## Notebook 01 — Schema & Explorative Datenanalyse (EDA)

### Einordnung und Zielsetzung

Notebook 01 bildet den methodischen Einstieg in die Analysepipeline. Ziel ist es, ein fundiertes Verständnis der Datenstruktur, der Datenqualität sowie zentraler statistischer Eigenschaften des Spotify-Datensatzes zu gewinnen, bevor Bereinigung, Feature Engineering oder Modellierung erfolgen. Der Fokus liegt bewusst auf Exploration, Validierung und der Identifikation potenzieller Risiken, die spätere Analysen beeinflussen könnten.

### Methodisches Vorgehen

Zu Beginn wird ein deterministisches Sampling definiert, bei dem ein reproduzierbarer Daten-Slice über ein modulares RowID-Verfahren ausgewählt wird. Diese Vorgehensweise ermöglicht effizientes Arbeiten auf großen Datenmengen und stellt gleichzeitig sicher, dass alle Analysen über verschiedene Notebooks hinweg

auf exakt derselben Datenbasis beruhen. Die aktive Slice-Konfiguration wird explizit gespeichert, um stille Inkonsistenzen zwischen Analyse-Schritten zu vermeiden.

Anschließend erfolgt eine konzeptionelle Analyse des relationalen Datenmodells. Die zentralen Entitäten (Tracks, Artists, Albums, Audio-Features) sowie deren Beziehungstabellen werden untersucht, um fehlerhafte Joins und Missinterpretationen frühzeitig auszuschließen. Darauf aufbauend wird das Schema direkt in SQLite inspiert, inklusive Tabellenumfang, Datentypen und Schlüsselstrukturen, ergänzt durch kurze Previews zur Erkennung typischer Formatprobleme.

Der aktive Slice wird anschließend als CSV-Snapshot exportiert und in Pandas geladen. Diese Trennung von Datenextraktion und Analyse erlaubt flexible explorative Auswertungen. Ein besonderer Fokus liegt auf der Normalisierung der Release-Daten, da diese unterschiedliche zeitliche Granularitäten aufweisen. Durch robustes Parsing und ein Granularitäts-Audit wird Transparenz über die zeitliche Auflösung geschaffen, was Voraussetzung für valide Zeitanalysen ist.

Die eigentliche EDA umfasst univariate, bivariate und multivariate Analysen sowie einen expliziten Datenqualitätsblock. Hier werden Verteilungen, Korrelationen, Quantilvergleiche, Missingness, Ausreißer, Regelverletzungen und Integritätsaspekte systematisch geprüft, bevor inhaltliche Interpretationen vorgenommen werden.

## Zentrale Beobachtungen

Die Audio-Features zeigen insgesamt plausible Wertebereiche und hohe interne Konsistenz. Gleichzeitig bestehen starke Abhängigkeiten zwischen einzelnen Merkmalen, insbesondere zwischen Energy und Loudness sowie zwischen Acousticness und Energy. Dies deutet auf teilweise redundante Feature-Strukturen hin und legt für spätere Modelle ein erhöhtes Multikollinearitätsrisiko nahe.

Die Popularität von Tracks und Artists ist extrem ungleich verteilt und folgt einer ausgeprägten Long-Tail-Struktur. Der Median liegt deutlich unter dem Mittelwert, was Durchschnittswerte nur eingeschränkt aussagekräftig macht. Der mit Abstand stärkste Zusammenhang mit Popularität besteht bei der Anzahl der Artist-Follower, während Audio-Features und Track-Metadaten nur schwache Effekte zeigen.

Künstlerbekanntheit erweist sich damit als dominanter, jedoch nicht allein ausreichender Erfolgsfaktor.

Zeitliche Analysen zeigen einen langfristigen Anstieg von Loudness und Energy, konsistent mit dem bekannten „Loudness War“. Andere Merkmale wie Tempo bleiben über Jahrzehnte hinweg weitgehend stabil, während Valence leicht rückläufige Tendenzen aufweist. Die Interpretation dieser Trends ist vor allem für moderne Musik valide, da frühere Jahrzehnte im Datensatz deutlich unterrepräsentiert sind.

Genre-spezifische Auswertungen zeigen klar unterscheidbare Charakteristika, etwa hohe Danceability und Energy in tanzorientierten Genres und niedrige Werte in klassischer Musik. Gleichzeitig ist die Streuung innerhalb der Genres groß, sodass Genre-Zugehörigkeit musikalische Eigenschaften nur teilweise erklärt.

Die Datenqualität ist insgesamt hoch, insbesondere bei zentralen Audio-Features. Auffällig sind jedoch einzelne Problemfelder wie hohe Ausreißeranteile bei Duration, Time Signature und Speechiness sowie extreme Werte bei Loudness. Zudem weisen einzelne Variablen eine sehr hohe Missingness auf, was ihre analytische Nutzbarkeit einschränkt.

## Robustheit der Ergebnisse

Ein Vergleich zweier unabhängig gezogener, deterministischer Daten-Slices zeigt eine sehr hohe Übereinstimmung hinsichtlich Verteilungen, Korrelationen und identifizierter Datenqualitätsprobleme. Die beobachteten Muster sind daher als robuste Eigenschaften des Datensatzes zu interpretieren und nicht als Stichprobeneffekte.

## Gesamteinordnung

Notebook 01 liefert ein kompaktes, belastbares Gesamtbild der Spotify-Daten. Es identifiziert strukturelle Eigenschaften, zentrale Einflussfaktoren und relevante Qualitätsprobleme und schafft damit die empirische Grundlage für die nachfolgenden Schritte der regelbasierten Datenbereinigung, Feature-Konstruktion und Modellierung.

## Notebook 02 — Data Cleaning & Rules

### Einordnung und Ziel

Notebook 02 bildet die Brücke zwischen der explorativen Analyse aus Notebook 01 und einer verlässlichen Grundlage für Feature Engineering und Modellierung. Ziel ist es, aus den exportierten CSV-Snapshots eines definierten Samples eine **Clean Layer** zu erzeugen, die einen stabilen Datenvertrag erfüllt. Stabil bedeutet in diesem Kontext konsistente Datentypen, klar definierte Wertebereiche, kontrollierten Umgang mit Ausreißern sowie eine saubere referenzielle Struktur der relationalen Tabellen, sodass spätere Joins keine unbeabsichtigten Duplikate oder Verzerrungen erzeugen.

### Laden der CSV-Exports und strukturelle Standardisierung

Zu Beginn werden die erwarteten CSV-Dateien des aktiven Samples geladen. In diesem Schritt erfolgt bewusst noch keine inhaltliche Bereinigung; der Fokus liegt auf einem sauberen strukturellen Einlesen der Daten. Alle Spaltennamen werden direkt in ein einheitliches `snake_case`-Format überführt, um sicherzustellen, dass nachfolgende Cleaning-Regeln zuverlässig greifen. Diese frühe Standardisierung ist notwendig, da selbst geringe Abweichungen in Spaltennamen dazu führen könnten, dass Regeln stillschweigend nicht angewendet werden.

### Frühe Reduktion und Signalkompression

Noch vor der eigentlichen Cleaning-Pipeline werden einzelne Felder entfernt oder vereinfacht, die entweder keinen stabilen Mehrwert liefern oder technisch ungünstig sind. Konkret werden in der Tabelle `tracks` die Spalte `is_playable` und in `albums` die Spalte `album_group` entfernt. Zusätzlich wird die URL-Spalte `preview_url` durch ein binäres Indikator-Feature `has_preview` ersetzt, das lediglich erfasst, ob ein Preview vorhanden ist. Die URL selbst wird anschließend verworfen. Dadurch wird hochdimensionierter Text reduziert, ohne das enthaltene Signal vollständig zu verlieren.

### Tabellenspezifisches Cleaning als deterministische Pipeline

Das eigentliche Cleaning erfolgt über eine deterministische Pipeline (`run_cleaning_pipeline`), die eine feste Reihenfolge spezialisierter Cleaner auf jede vorhandene Tabelle anwendet. Jede Tabelle wird nur dann bereinigt, wenn sie im aktuellen Sample existiert, wodurch der Prozess robust gegenüber unvollständigen Exports bleibt.

Für `tracks` wird sichergestellt, dass ein konsistenter Primärschlüssel (`track_id`) existiert, boolesche Felder robust geparsst werden und die Track-Dauer ausschließlich positive Werte annimmt. Extreme Ausreißer der Dauer werden über ein hohes Quantil (Standard: 0.999) nach oben begrenzt. Die Popularität wird strikt auf den Spotify-definierten Bereich von 0 bis 100 gekappt.

In `audio_features` werden alle kontinuierlichen Merkmale in den Bereich [0,1] gezwungen, Tempo-Werte auf positive Bereiche beschränkt und ebenfalls quantilbasiert begrenzt. Lautstärke wird auf einen festen Hard-Range begrenzt, und Schlüsselmerkmale wie `key` und `mode` werden domänenentreu normalisiert.

Für `albums` wird das Veröffentlichungsdatum mit einer robusten Parsing-Funktion in ein einheitliches Datumsformat überführt. Alben mit nicht parsbaren Datumsangaben werden entfernt, da zeitliche

Information später zentral für Analysen und Splits ist. Album-Typen werden normalisiert, Popularität erneut auf [0,100] begrenzt.

In `artists` werden IDs vereinheitlicht, Follower-Zahlen numerisch und nicht-negativ gemacht sowie Popularitätswerte domänentreu gekappt.

Für alle Beziehungstabellen stellt ein eigener Cleaner sicher, dass zusammengesetzte Schlüssel vollständig und eindeutig sind. Duplikate werden nicht zufällig entfernt, sondern über einen deterministischen Mechanismus aufgelöst, der jeweils die inhaltlich vollständigste Zeile behält. Dadurch wird Informationsverlust durch Exportartefakte vermieden.

## **Erzwingen referenzieller Integrität**

Nach dem tabellenspezifischen Cleaning wird die referenzielle Struktur explizit stabilisiert. Dazu werden Mengen gültiger IDs aufgebaut und genutzt, um Beziehungstabellen von verwaisten Einträgen zu bereinigen. Zeilen, die auf nicht existierende Entitäten verweisen, werden konsequent entfernt. Bestimmte Fremdschlüssel, etwa die Referenz von Tracks auf Audio-Features, werden jedoch bewusst als optional behandelt. In diesen Fällen wird der Fremdschlüssel auf NA gesetzt, anstatt den gesamten Track zu verwerfen, um wertvolle Beobachtungen nicht unnötig zu verlieren.

## **Regelbasierte Outlier-Behandlung mit Signal-Erhalt**

Im Anschluss an das Basis-Cleaning werden zusätzliche regelbasierte Outlier-Stages angewandt. Dabei werden Ausreißer nicht pauschal entfernt, da Musikdaten typischerweise Heavy-Tail-Verteilungen und reale Sonderfälle enthalten. Stattdessen werden unmögliche Werte invalidiert, extreme Tails quantilbasiert gekappt und ergänzende Indikator-Features erzeugt, die Extremfälle explizit markieren. Beispiele sind Flags für ungewöhnlich lange Tracks, extreme Track- oder Disc-Nummern, seltene Taktarten, sehr hohe Sprachanteile oder extrem hohe Follower-Zahlen. Auf diese Weise bleibt das Signal erhalten, ohne die Modelle durch extreme Werte zu destabilisieren.

## **Qualitätsprüfungen und Persistierung**

Vor der Persistierung werden harte Quality Gates angewandt. Diese prüfen unter anderem die Eindeutigkeit und Nicht-Null-Bedingungen der Primärschlüssel, die Einhaltung definierter Wertebereiche sowie optional die referenzielle Integrität der Beziehungstabellen. Bei Verletzung zentraler Varianten bricht das Notebook bewusst ab, um fehlerhafte Daten nicht weiterzugeben.

Die bereinigten Tabellen werden anschließend als Parquet-Dateien gespeichert, da dieses Format für spätere Modellschritte spechereffizient und performant ist. Ergänzend wird ein strukturierter Audit-Report erzeugt, der Profilstatistiken vor und nach dem Cleaning sowie die Nettoveränderungen der Zeilenanzahlen dokumentiert. Dadurch bleibt der gesamte Bereinigungsschritt transparent und versionierbar.

# **Notebook 03 — Modeling Tables Build**

## **Einordnung und Zielsetzung**

Dieses Notebook überführt die bereinigten Parquet-Tabellen aus dem Clean-Layer in **modellierungsfertige, denormalisierte Feature-Tabellen**. Ziel ist es, eine konsistente Datenbasis zu erzeugen, in der jede Beobachtungseinheit eindeutig definiert ist und ohne weitere Joins direkt in ML-Pipelines verwendet werden kann. Dafür werden drei getrennte Ebenen aufgebaut: Track, Album und Artist. Diese Trennung ist methodisch sinnvoll, da sich spätere Analyse- und Modellierungsaufgaben auf unterschiedliche Entitäten beziehen und jeweils eigene Feature-Logiken erfordern.

## **Setup, Reproduzierbarkeit und Konfiguration**

Zu Beginn lädt das Notebook die zentralen Projektparameter, darunter den Zufallsseed, Konfigurationsschalter zur Vermeidung von Feature-Leakage sowie die Strategie zur Zuordnung eines Hauptalbums pro Track. Anschließend werden alle Pfade sample-spezifisch erzeugt, sodass sämtliche Artefakte reproduzierbar abgelegt werden. Ergänzend wird eine Run-Metadatenstruktur erzeugt, die dokumentiert, mit welchen Einstellungen die Modeling-Tables erstellt wurden. Dadurch bleiben spätere Modellresultate eindeutig einer konkreten Datenableitung zuordenbar.

## **Laden des Clean-Layers über einen Table Catalog**

Die bereinigten Daten werden nicht manuell geladen, sondern über einen zentralen Table-Catalog, der die definierten Entitäts- und Bridge-Tabellen aus dem Clean-Parquet-Verzeichnis verwaltet. Das Laden erfolgt bewusst nicht strikt, sodass das Notebook auch bei unvollständigen Samples ausführbar bleibt. Diese Designentscheidung erhöht die Robustheit der Pipeline gegenüber variierenden Datenabdeckungen.

## **Track-Level Dataset: 1 Zeile = 1 Track**

Im ersten Hauptschritt wird ein Track-zentriertes Dataset aufgebaut, bei dem die Beobachtungseinheit strikt als einzelner Track definiert ist. Track-Metadaten werden mit Audio-Features und relationalem Kontext zusammengeführt. Der Join zu Audio-Features erfolgt als Left-Join, sodass Tracks ohne Audio-Features erhalten bleiben und Missingness explizit als Signal bestehen bleibt. Die potenziell mehrdeutige Track-Album-Beziehung wird über eine deterministische Main-Album-Strategie auf eine 1:1-Zuordnung reduziert, wodurch stabile Zeit- und Albumkontext-Features entstehen. Beziehungen zu Artists werden aggregiert, sodass Kollaborationen und Reichweitenmerkmale modellierbar werden. Genre-Informationen werden konsistent über die beteiligten Artists abgeleitet, um strukturelle Inkonsistenzen zu vermeiden.

In einer zweiten Stufe werden auf dem Track-Dataset gezielt modellfreundliche Features erzeugt, darunter einfache Metasignale, robuste Transformationen und binäre Flags. Die Trennung zwischen strukturellem Aufbau und Feature Engineering sorgt dafür, dass die relationale Logik klar von statistischen Anpassungen getrennt bleibt.

## **Album-Level Dataset: 1 Zeile = 1 Album**

Im zweiten Block wird ein Album-zentriertes Dataset erzeugt, bei dem Track-Informationen ausschließlich in aggregierter Form einfließen. Jedes Album wird durch Metadaten, Zeitinformationen, strukturelle Merkmale wie Track-Anzahl sowie aggregierte Audio-Signaturen beschrieben. Zusätzlich werden die beteiligten Artists und deren Reichweitenmerkmale zusammengefasst. Genres werden analog zum Track-Dataset über Artist-Genres abgeleitet. Der anschließende Engineering-Schritt stabilisiert Skalen und ergänzt strukturierende Flags, sodass unterschiedliche Albumtypen vergleichbar bleiben.

## **Artist-Level Dataset: 1 Zeile = 1 Artist**

Im dritten Block entsteht ein Artist-zentriertes Dataset, bei dem Artist-Features primär durch Aggregation über Track-Daten abgeleitet werden. Jeder Artist erhält ein kompaktes Profil aus durchschnittlichen Audio-Eigenschaften, Produktivitätsmaßen und Track-basierten Signalen wie mittlerer Popularität oder Explicit-Anteil. Genre-Informationen werden als Mehrfachzuordnung beibehalten. Im Engineering-Schritt liegt der Fokus auf der Stabilisierung von Heavy-Tail-Verteilungen sowie auf der Ableitung relativer und stilistischer Merkmale, die den musikalischen Charakter eines Artists beschreiben.

## Persistierung der Modeling Tables

Zum Abschluss werden alle drei Modeling-Tables als Parquet-Dateien persistiert. Diese Tabellen bilden die zentrale Schnittstelle zu den folgenden Notebooks für Target-Erstellung, Feature-Matrizen und Modelltraining. Die Wahl des Parquet-Formats ermöglicht eine effiziente Speicherung und schnelle Weiterverarbeitung großer Tabellen.

## Notebook 04 — Targets und Feature-Matrizen

### Einordnung und Zielsetzung

Notebook 04 ist der Schritt, in dem aus den modellierungsfertigen Tabellen aus Notebook 03 tatsächlich **lernreife Trainingsdaten** entstehen. Ziel ist es, einerseits konsistente Zielvariablen für unterschiedliche Aufgabenstellungen zu erzeugen und andererseits dazu passende, leakage-sichere Feature-Matrizen aufzubauen. Das Ergebnis dieses Notebooks ist ein geschlossenes Paket aus Feature-Matrizen und Targets, das in den folgenden Trainingsnotebooks unverändert wiederverwendet wird. Dadurch ist sichergestellt, dass Training und spätere Inference exakt auf derselben Feature-Logik basieren.

### Setup, Sample-Kontext und reproduzierbare Struktur

Zu Beginn wird der aktive Sample-Kontext geladen und daraus eine einheitliche, sample-spezifische Ordnerstruktur erzeugt. Alle Ausgabepfade werden explizit angelegt, sodass Artefakte reproduzierbar und eindeutig abgelegt werden. Zusätzlich werden zentrale Utility-Module bewusst neu geladen, um sicherzustellen, dass Änderungen an Target-Definitionen oder Feature-Schemata ohne versteckte Altimporte wirksam werden. Dieser Schritt ist wichtig, um stille Inkonsistenzen zwischen Code und erzeugten Trainingsdaten zu vermeiden.

### Laden der Modeling Tables als Ausgangsbasis

Als Ausgangsbasis dienen die drei Modeling Tables auf Track-, Album- und Artist-Ebene. Diese werden über einen zentralen Table-Katalog geladen, der das Notebook robust gegenüber leicht variierenden Samples hält. Bestimmte Spalten, etwa die Track-Dauer, werden dynamisch referenziert, sodass unterschiedliche Exportvarianten kompatibel bleiben, ohne harte Annahmen über Spaltennamen zu treffen.

### Konstruktion der Targets auf Track-Ebene

Die Zielvariablen werden konsequent auf Track-Ebene konstruiert, da der Track die sauberste Ereigniseinheit für zeit- und erfolgsbezogene Definitionen darstellt. Die Target-Erzeugung erfolgt über einen einheitlichen Builder, der mehrere Aspekte kombiniert: eine kohortenbasierte Logik, eine perzentilbasierte Hit-Definition, schwach supervisierte Mood-Tags sowie eine zeitlich gerichtete Artist-Trajectory-Definition.

Das zentrale Success-Target wird relativ zur jeweiligen Release-Kohorte definiert, um zeitliche Verschiebungen im Katalog zu kontrollieren. Ergänzend wird ein Residual-Target berechnet, das Über- oder Unterperformance innerhalb der Kohorte abbildet. Das Hit-Target basiert auf einer festen Perzentilschwelle und erzwingt zusätzlich eine stabile Positivrate pro Jahr, damit das Label über Zeit vergleichbar bleibt. Die Mood-Tags entstehen deterministisch aus Audio-Features über Quantilschwellen und bilden ein reproduzierbares Multi-Label-Target ohne externe Annotation. Für Artists wird zusätzlich ein Trajectory-Target erzeugt, das auf einem festen Vergangenheits- und Zukunftsfenster basiert und sowohl kontinuierliches Wachstum als auch einen Breakout-Indikator liefert.

Alle Targets werden explizit typisiert, um Speicherverbrauch und Konsistenz in späteren Pipelines zu sichern. Neben den Zielvariablen werden auch erweiterte Zwischenstrukturen erzeugt, die die zeitlich

gerichteten Aggregationen kapseln und die Logik der Target-Erzeugung klar vom restlichen Notebook trennen.

## Genre-Repräsentation und kontrollierte Dimensionalität

Genres liegen als Listen vor und werden daher in eine feste numerische Repräsentation überführt. Dazu wird zunächst ein Vokabular der häufigsten Genres bestimmt, wodurch die Feature-Dimension bewusst begrenzt wird. Anschließend werden konsistente Multi-Hot-Matrizen für Track-, Album- und Artist-Ebene erzeugt. Die Ausrichtung erfolgt jeweils über stabile IDs, sodass die Genre-Spalten in allen Matrizen semantisch identisch bleiben und wiederverwendet werden können.

## Task-spezifische Feature-Matrizen mit Leakage-Schutz

Die eigentlichen Feature-Matrizen werden über ein formales Feature-Schema erzeugt, das explizit festlegt, welche Feature-Gruppen für welche Aufgabe zulässig sind. Dadurch werden ID-Leaks, URL-Leaks und indirekte Popularitäts-Proxies systematisch ausgeschlossen, ohne dass im Notebook manuell Spalten entfernt werden müssen. Unterschiedliche Aufgaben erhalten bewusst unterschiedliche Feature-Sets: Hit- und Success-Modelle arbeiten mit streng kontrollierten Features, während Artist-Modelle zusätzliche Reichweiteninformationen nutzen dürfen. Für zeitgerichtete Trajectory-Modelle werden ausschließlich vergangenheitsbasierte Features verwendet, sodass die zeitliche Kausalrichtung bereits auf Feature-Ebene abgesichert ist.

Dieses Schema-basierte Vorgehen stellt sicher, dass Training und Inference dieselbe Spaltenlogik verwenden und dass Feature-Leakage nicht implizit durch spätere Änderungen eingeführt wird.

## Persistierung der Trainingsartefakte

Zum Abschluss werden alle erzeugten Datasets, Feature-Matrizen, Targets und Genre-Repräsentationen als versionierte Artefakte gespeichert. Der Export wird explizit als abgeschlossener Schritt markiert, sodass die folgenden Trainingsnotebooks exakt diese Inputs laden können, ohne die Ableitung erneut auszuführen. Damit trennt Notebook 04 sauber die **Daten- und Feature-Ableitung** vom eigentlichen **Modelltraining** und bildet die verbindliche Schnittstelle zwischen Datenaufbereitung und Lernen.

# Notebook 05 — Base ML Model Training und Berichte

## Zielsetzung des Notebooks

Ziel dieses Notebooks ist es, auf Basis der in Notebook 04 erzeugten, leakage-sicheren Feature-Matrizen und Targets eine **robuste Baseline** für alle zentralen Modellierungsaufgaben des Projekts zu etablieren. Der Fokus liegt dabei nicht auf maximaler Modellleistung, sondern auf **Reproduzierbarkeit, Vergleichbarkeit und methodischer Sauberkeit**. Die hier trainierten Modelle dienen als Referenzpunkt für spätere Fine-Tuning- und Robustheitsanalysen und sollen zeigen, welche Signale bereits mit einfachen, gut kontrollierten Modellen lernbar sind.

## Ausgangshypothesen

Die Modellierung basiert auf mehreren klar formulierten Hypothesen, die sich direkt aus der explorativen Analyse ableiten:

Erstens wird angenommen, dass **Track-Erfolg stark zeitabhängig ist** und absolute Popularität über verschiedene Release-Jahre hinweg nicht vergleichbar ist. Daher sollten kohortenbasierte Zielgrößen stabilere Lernsignale liefern als rohe Popularitätswerte.

Zweitens wird angenommen, dass es **Overperformer innerhalb einer Kohorte** gibt, also Tracks, die relativ zu ihrem zeitlichen Kontext besser abschneiden als erwartet, auch wenn sie absolut keine globalen Hits sind.

Drittens wird angenommen, dass **Hit-Erfolg ein seltenes Ereignis** mit starker Klassenungleichverteilung ist und daher als Klassifikationsproblem mit expliziter Threshold-Optimierung modelliert werden muss.

Viertens wird angenommen, dass **Empfehlungsszenarien primär Ranking-Probleme** sind, bei denen relative Ordnung wichtiger ist als absolute Scores.

Fünftens wird angenommen, dass **Artist-Erfolg eine zeitlich strukturierte Dynamik** besitzt, die aus vergangenen Veröffentlichungen ableitbar ist und sowohl kontinuierliches Wachstum als auch plötzliche Breakouts umfasst.

Schließlich wird angenommen, dass **musikalische Stimmungen (Mood)** als schwach supervisierte Multi-Label-Ziele aus Audio-Features deterministisch ableitbar und lernbar sind.

## Modellwahl und Implementierung

Die eingesetzten Modelle sind bewusst als **Baseline-Modelle** gewählt. Für jede Hypothese wird die jeweils **minimal passende Modellklasse** verwendet, um unnötige Komplexität zu vermeiden und die Ergebnisse gut interpretierbar zu halten.

Für den Track-Erfolg werden zwei Regressionsmodelle trainiert: eines für das **kohortenbasierte Erfolgsperzentil** und eines für das **Success-Residual**, das die Abweichung vom kohorteninternen Erwartungswert modelliert. Beide Modelle werden über standardisierte Trainer-Klassen trainiert und evaluiert, die Training, Validierung und Metrikberechnung kapseln.

Die **Hit Prediction** wird als binäre Klassifikation umgesetzt. Neben klassischen Metriken wird explizit ein optimaler Entscheidungs-Threshold bestimmt, da ein fixer Threshold von 0.5 bei stark unausgeglichenen Klassen in der Regel ungeeignet ist.

Für Empfehlungsszenarien wird ein **Learning-to-Rank-Modell** trainiert, das innerhalb kohortenbasierter Gruppen optimiert. Damit wird direkt die relative Ordnung der Tracks gelernt, was näher an realen Empfehlungssystemen liegt als punktweise Regression.

Die **Artist-Trajectory-Modelle** arbeiten auf einer Panel-Struktur, die ausschließlich vergangenheitsbasierte Features enthält. Dadurch können sowohl zukünftiges Wachstum (Regression) als auch Breakout-Ereignisse (binäre Klassifikation) modelliert werden, ohne zeitliches Leakage zu riskieren.

Ergänzend werden zwei **unüberwachte Modelle** trainiert: ein Artist-Clustering zur explorativen Segmentierung sowie ein Track-Similarity-Modell auf Basis von Audio-Feature-Embeddings, das „Songs-wie-dieser“-Abfragen ermöglicht.

Für die **Mood-Vorhersage** wird ein Multi-Label-Modell mit label-spezifischen Thresholds trainiert, da die einzelnen Mood-Dimensionen sehr unterschiedliche Score-Verteilungen aufweisen.

## Ergebnisse und Sample-Vergleich

Alle Modelle wurden auf zwei unabhängigen, deterministisch gezogenen Samples (**schema\_000** und **schema\_001**) trainiert, um die Robustheit der Ergebnisse zu überprüfen.

Die **kohortenbasierten Regressionsmodelle** zeigen in beiden Samples nahezu identische Ergebnisse. Das Success-Percentile-Modell erreicht einen MAE von etwa 21–22 und ein R<sup>2</sup>R<sup>2</sup> von ca. 0.18–0.20, während das Residual-Modell einen MAE von etwa 12–13 und ein R<sup>2</sup>R<sup>2</sup> von rund 0.21–0.22 erzielt. Die geringe erklärte Varianz bestätigt die Hypothese, dass Track-Erfolg nur teilweise aus beobachtbaren Features erklärbar ist.

Die **Artist-Trajectory-Modelle** sind über beide Samples hinweg außergewöhnlich stabil. Das Growth-Modell erreicht ein R<sub>2</sub>R<sup>2</sup>R von etwa 0.985, während die Breakout-Klassifikation eine ROC-AUC nahe 0.99 und eine PR-AUC über 0.9 erzielt. Diese Ergebnisse zeigen, dass zeitliche Artist-Signale stark strukturiert und gut vorhersagbar sind.

Für die **Hit Prediction** liegt die ROC-AUC in beiden Samples bei etwa 0.82–0.83. PR-AUC und F1-Score schwanken leicht (PR-AUC ca. 0.34–0.40), was konsistent mit der starken Klassenungleichverteilung ist, aber keine strukturelle Instabilität erkennen lässt.

Das **Learning-to-Rank-Modell** erreicht in beiden Samples eine nahezu identische NDCG@10 von etwa 0.74, was eine stabile Ranking-Qualität zeigt. Auch das **Multi-Label-Mood-Modell** liefert sehr konsistente Ergebnisse mit Micro- und Macro-F1-Werten um ca. 0.69–0.70.

Insgesamt zeigen die Metriken **keine wesentlichen Unterschiede zwischen den beiden Samples**. Abweichungen liegen im erwartbaren Bereich zufälliger Stichprobenvariation.

## Gesamteinordnung

Notebook 05 zeigt, dass bereits mit sauberen Targets, kontrollierten Features und einfachen Baseline-Modellen **stabile und reproduzierbare Ergebnisse** erzielt werden können. Die Ergebnisse bestätigen zentrale Hypothesen über die Struktur von Erfolg, Ranking und Artist-Entwicklung im Spotify-Katalog und machen gleichzeitig die Grenzen der Vorhersagbarkeit deutlich. Damit bildet dieses Notebook eine belastbare Referenz für gezieltes Fine-Tuning und Robustheitsanalysen im nächsten Schritt.

## Notebook 06 — Fine Tuning und Robustheit

### Zielsetzung des Notebooks

Notebook 06 dient der **systematischen Optimierung und Absicherung** der in Notebook 05 etablierten Baseline-Modelle. Ziel ist es, pro Task durch gezieltes Hyperparameter-Tuning die Generalisierung im kohortenbasierten Zeitsplit zu verbessern und gleichzeitig sicherzustellen, dass die gefundenen Verbesserungen **reproduzierbar und robust** sind. Methodisch trennt dieses Notebook strikt zwischen **Suchphase** (Optuna-Tuning) und **finalem Modell** (Retrain mit Best-Parametern), sodass die Ergebnisse klar nachvollziehbar bleiben.

### Vorgehen und Implementierung

Alle Tuning-Läufe basieren ausschließlich auf den in Notebook 04 erzeugten Feature-Matrizen und Targets. Pro Task wird ein konsistentes Task-Dataset erzeugt, das einen kohortenbasierten Train/Validation-Split verwendet. Dadurch wird sichergestellt, dass das Tuning reale zeitliche Generalisierung optimiert und kein Future-Leakage entsteht.

Für jede Aufgabe wird eine **task-adäquate Optimierungsmetrik** gewählt: MAE für Regressionsaufgaben, PR-AUC für stark unausgeglichene Klassifikation, Micro-F1 für Multi-Label-Mood und NDCG@10 für Ranking. Die Suche erfolgt über Optuna mit begrenzter Trial-Zahl, Early Stopping und expliziter Regularisierung, um Überanpassung zu vermeiden. Nach Abschluss der Suche werden alle Modelle mit den besten Parametern **neu trainiert** und als finale Artefakte gespeichert.

### Kohortenbasierter Erfolg (Regression)

Für das **Success-Percentile-Modell** reduziert das Tuning den Validierungs-MAE auf **20.29**, mit einem besten Trial-R<sub>2</sub>R<sup>2</sup>R von **0.26**. Im finalen Test liegt der MAE bei **23.02** und das R<sub>2</sub>R<sup>2</sup>R bei **0.14**. Im Vergleich zur Baseline zeigt sich eine **moderate, aber konsistente Verbesserung**, wobei die begrenzte erklärbare Varianz bestätigt, dass Track-Erfolg nur teilweise aus beobachtbaren Features ableitbar ist.

Das **Success-Residual-Modell** profitiert stärker vom Tuning. Der Validierungs-MAE sinkt auf **10.12**, und im Test erreicht das Modell ein R<sup>2</sup>R<sup>2</sup> von **0.20**. Damit wird Overperformance innerhalb der Kohorte robuster modelliert als mit der Baseline.

### Hit Prediction (binäre Klassifikation)

Für die **Hit Prediction** wird PR-AUC optimiert. Das getunte Modell erreicht eine **PR-AUC von 0.335** und eine **ROC-AUC von 0.80** im Testset. Der optimale Entscheidungs-Threshold liegt bei etwa **0.55**, mit einem F1-Score von **0.38**. Gegenüber der Baseline ist die Rangqualität stabil, während die absolute Klassifikationsleistung erwartungsgemäß durch die starke Klassenungleichverteilung begrenzt bleibt.

### Mood Prediction (Multi-Label)

Das **Mood-Modell** zeigt nach Tuning eine **Micro-F1 von 0.687** und eine **Macro-F1 von 0.700**. Besonders stabile Labels sind *instrumental* ( $F1 \approx 0.95$ ) und *chill* ( $F1 \approx 0.77$ ), während abstraktere Stimmungen wie *danceable* erwartungsgemäß schwieriger bleiben. Insgesamt bestätigt das Tuning, dass Mood-Tags aus Audio-Features zuverlässig und reproduzierbar lernbar sind.

### Learning-to-Rank

Das getunte **Ranking-Modell** erreicht eine **mean NDCG@10 von 0.733**, was praktisch identisch zur Baseline ist. Das zeigt, dass die Baseline bereits nahe an einem lokalen Optimum lag und weiteres Tuning nur begrenzte Gewinne bringt. Wichtig ist hier weniger der absolute Zugewinn als die bestätigte **Stabilität des Rankings**.

### Unüberwachte Modelle

Für das **Artist-Clustering** wird die optimale Clusterzahl auf **k = 34** mit **9 PCA-Komponenten** bestimmt. Der Silhouette-Score liegt bei etwa **0.20**, was für hochdimensionale, heterogene Musikdaten typisch ist. Die Cluster sind kompakt und stabil, aber nicht scharf trennbar, was die kontinuierliche Natur musikalischer Stile widerspiegelt.

Die **Track-Similarity-Embeddings** werden erfolgreich auf dem vollständigen Datensatz neu trainiert. Plausibilitätsabfragen liefern konsistente „Songs-wie-dieser“-Ergebnisse, was die Robustheit der Similarity-Pipeline bestätigt.

### Robustheit und Vergleich der Samples

Die hier dargestellten Ergebnisse basieren auf **schema 001**. Aufgrund der bereits in Notebook 05 beobachteten hohen Übereinstimmung zwischen **schema\_000** und **schema\_001** ist davon auszugehen, dass die getunten Metriken für **schema\_000 sehr ähnlich ausfallen**. Das Tuning verbessert einzelne Tasks moderat, verändert jedoch nicht die grundsätzlichen Leistungsgrenzen oder Rangfolgen der Modelle.

### Gesamteinordnung

Notebook 06 zeigt, dass gezieltes Hyperparameter-Tuning die Baseline-Modelle punktuell verbessert, ohne deren grundlegende Charakteristika zu verändern. Besonders bei Residual-Regression und Multi-Label-Mood ergeben sich messbare Gewinne, während Ranking und Hit-Prediction primär durch die Datenstruktur begrenzt bleiben. Durch die saubere Trennung von Suche und finalem Retrain entsteht ein reproduzierbarer, robuster Modell-Snapshot, der den Abschluss der Modellierungs-Pipeline bildet.