Visualisierung von Musikdaten mittels UMAP und t-SNE

Data Management Fundamentals

an der Fakultät für Wirtschaft
im Studiengang Wirtschaftsinformatik

an der
DHBW Ravensburg

Verfasser:	Anton Geiger
Ausbildungsbetrieb:	Festo SE & Co KG
Anschrift:	Ruiter Straße 82
	73734 Esslingen Berkheim
Dozent:	Herr Dr. Andreas Buckenhofer
Abgabedatum:	28.3.2025

Inhaltsverzeichnis

Abkürzungsverzeichnis

CSV ..... Comma Separated Values
GUI ..... Graphical User Interface
t-SNE ..... t-Distributed Stochastic Neghbour Embedding
UMAP ..... Unifold Approximation and Projection

Abbildungsverzeichnis

Tabellenverzeichnis

Formeln

1 Grundlagen

In dieser Arbeit geht es um die Umsetzung einer Dimensionsreduktion von der Uniform Manifold Approximation and Projection (UMAP) und t-Distributed Stochastic Neighbour Embedding (t-SNE) anhand eines hochdimensionalen Datensatzes, welcher Informationen über Musikstücke von Spotify fasst.

1.1 Mathematische Funktionsweise von UMAP

UMAP reduziert die Dimensionalität von Daten, indem es zunächst die Abstände zwischen den nächsten Nachbarn berechnet. Anschließend projiziert es diese Abstände in einen niedrigerdimensionalen Raum und optimiert die resultierende Verteilung durch Minimierung der Cross-Entropy.

Bestimmung von Kugeln mit K Nachbarn für jeden Punkt

Im Detail müssen erst die Abstände zwischen den Datenpunkten definiert und bestimmt werden. Da die eingegebenen Daten nicht gleichmäßig verteilt sind, wird die geodätische Distanz zwischen zwei Datenpunkten auf der lokalen Mannigfalitkeit berechnet, wo eine Gleichverteilung vorliegt. Dabei wird um jeden Punkt basierend auf der Nähe der k Nachbarknoten ein Ball geformt und die Distanz zu den anderen Punkten basierend auf der geodätischen Distanz auf dem Ball bestimmt. Um den Radius des Balles und die Nachbarn eines Punktes zu erhalten, wird (**Form. 1**) verwendet, welche die Wahrscheinlichkeit errechnet, dass ein potenzieller Nachbarpunkt u ebenfalls innerhalb des kleineren Balls mit der Hälfte des Radius Bestandteil ist (vgl. Cheng et al., 2011, o.S.). Betitelt wird dieser Algorithmus als Nearest Neighbour Descent (vgl. McInnes et al., 2020, S.5). Dabei wird mit einem hohen Radius gestartet, welcher viele Punkte beeinhaltet. Basierend auf der Idee, dass Nachbarpunkte nah beieinander liegen, wird nun stückweise die Wahrscheinlichkeit abgeschätzt, dass ein Punkt u immernoch in einer halbierten Kugel liegt. $B[v]$ ist die aktuell betrachtete Menge der Punkte im im Ball. $B'[v]$ ist dabei die neue Kugelmenge mit kleinerem Radius. Der Radius wird solange halbiert, wie diese Wahrscheinlichkeit hoch bleibt, um die tatsächlichen Nachbarn zu finden (vgl. Cheng et al., 2011, o.S).

$$Pr\{u \in B'[v]\} \approx \frac{K}{\left|B_{\frac{r}{2}}[v]\right|} \tag{1}$$

Formel W'keit u in kleineren Kreis

Skalierung der Abstände

Mit dem nun kalkulierbaren Radius des Balles um einen Punkt p soll nun die geodätische Distanz zwischen Punkten p und q mit der Formel (**Form. 2**) d_M berechnet werden. d_R ist dabei die euklidische Distanz zwischen zwei Punkten auf der Mannigfalitkeit M (Kugeln), wo die

Punkte näherungsweise normalverteilt vorliegen.

$$d_M(p,q) \approx \frac{1}{r} d_R(p,q) \tag{2}$$

Formel geodätische Distanz

Mit den verschiedenen Bällen mit aus K Nachbarn wird nun die Distanz gemäß des Radiusses zu den Nachbarn normalisiert. Diese Normalisierung der euklidischen Distanzen zur geodätische Distanz führt dazu, dass der krümmungsbeschreibende Parameter g der Mannigfalitkeit M, auf der die Daten liegen über alle Punkte näherungsweise konstant wird und dadurch nun von einer Gleichverteilung der Daten ausgegangen werden kann (vgl. McInnes et al., 2020, S.5).
Dadurch, dass jeder Punkt einen anderen Ball und so unterschiedliche normalisierte Distanzen zu den Nachbarn hat, folgt deshalb, dass der Abstand von Punkt P_1 zu P_2 anders ist als jener von P_2 zu P_1 (vgl. McInnes et al., 2020, S.5). Dadurch wird erreicht, dass die Distanzen der nicht dicht besiedelten Punkte nicht mit dichtbesiedelten Punkten direkt über die euklidische Distanz verglichen werden können. Vielmehr hängt nun die Distanz zu den Punkten auch davon ab, wie viele Punkte andere in der Nähe des Punkten vorhanden sind. Das zeigt eine Visualisierung der Kugelgrößen am folgenden Bespiel abgebildet in (**Abb. 1.1**), da dicht besiedelte Punkte kleinere Kugeln haben als die dicht besiedelten.

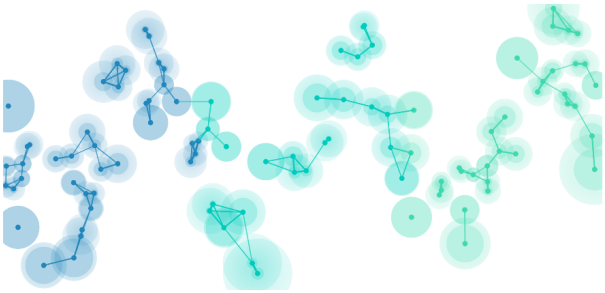


Abb. 1.1: Kugelgrößen bei UMAP

Bestimmung der Ähnlichkeitswerte zwischen den Punkten im hochdimensionalen Raum
Mit den nun normalisierten Distanzen d_M vom Ausgangspunkt wird dann der Ähnlichkeitswert der Punktepaare mit der Formel (**Form. 3**) bestimmt, indem der Parameter σ angepasst wird. Ein höheres σ bedeutet, wie bei einer Normalverteilung auch, dass die Verteilung in die Breite gezogen wird, und so Punkte die weiter weg voneinander liegen ein höheres Gewicht erhalten. p_i ist dabei ein Parameter der dafür sorgt, dass die Gewichte von einem Punkt zu einem anderen Punkt genau Eins wird (vgl. McInnes et al., 2020, S.14).

$$\sum_{j=1}^{k} e^{\left(\frac{-max(0,d(x_i,x_{ij})-p_i)}{\sigma_i}\right)} = log_2(k) = \sum_{j=1}^{k} w((x_i,x_{ij})) \tag{3}$$

Gewichtung UMAP Distanzen

Wobei jeder Anteil der oben genannten Formel zwischen zwei Punkten als Gewicht $w((x_i, x_{ij}))$ aufgefasst wird und aufsummiert genau den Logarithmus von von K Nachbarn zur Basis 2 gibt.

Optimierung der Abstände im niedrigdimensionalen Raum

Nachdem mit den Gewichten die Adjazenzmatrix G aufgestellt wurde, beginnt die Optimierung und das Auseinanderzurren der Punkte im niedrigdimensionalen Raum. So werden iterativ anziehende und abstoßende Funktionen zwischen 2 Punkten mit den niedrigdimensionalen Koordinaten y_i und y_j bestimmt. Dabei gilt folgende Formel (**Form. 4**) für die Anziehungskraft beider Punkte mit den Hyperparametern a und b (vgl. McInnes et al., 2020, S.14ff).

$$\frac{-2ab\left|\left|y_i - y_j\right|\right|_2^{2(b-1)}}{1+\left|\left|y_i - y_j\right|\right|_2^2} * w((x_i, x_j)) * (y_i - y_j) \tag{4}$$

Anziehung UMAP im niedrigdimensionalen Raum

und für die Abstoßungskraft (**Form. 5**):

$$\frac{2b}{(\varepsilon + ||y_i - yj||_2^2)(1+\alpha\,||y_i - yj||_2^{2b})} * (1 - w((x_i, x_j))) * (y_i - y_j) \tag{5}$$

Abstoßung UMAP im niedrigdimensionalen Raum

Die beiden Formeln stammen vom Gradienten, der bei der Minimierung der Lossfunktion, der Cross Entropy, entsteht. Die Cross Entropy wird für die Adjazenmatrix G und der durch die y Werte berechenbaren Matrix H bestimmt und es wird versucht die y Werte so zu wählen, dass der Unterschied zwischen der kalkulierten Matrix H so nah wie möglich an die Matrix G gering wird. Dies wird durch die Optimierung der Cross Entropy erreicht (vgl. McInnes et al., 2020, S.14ff).

1.2 Mathematische Funktiosweise von t-SNE

Der andere zu betrachtene Algorithmus ist der t-SNE Algorithmus, welcher ebenfalls zwei Verteilungen definiert und dann versucht iterativ den Unterschied beider Verteilungen zu minimieren (vgl. Cai und Ma, 2022, S.1f). Im Kern handelt es sich bei t-SNE wieder um ein Minimierungsproblem der Divergenz der Verteilung und Relation der Datenpunkten im hohen sowie im niedrigdimensionalen Raum (vgl. van der Maaten, 2014, S.4f).

Bestimmung der hochdimensionalen Ähnlichkeitswerte durch Normalverteilung

Als Erstes werden für alle Kombinationen zweier Punkte alle möglichen Wahrscheinlichkeiten, dass Punkt j Nachbar von meinem Punkt i ist mit der Formel (**Form. 6**) errechnet. x_i - x_j ist dabei die euklidische Distanz beider Datenpunkte und σ ein zu optimierender Parameter, welcher im weiteren Verlauf genauer erklärt wird.

$$p_{j|i} = \frac{e^{(\frac{-||x_i - x_j||^2}{2\sigma_i^2})}}{\sum_{k \neq i} e^{(\frac{-||x_i - x_k||^2}{2\sigma_i^2})}} \tag{6}$$

Wahrscheinlichkeit von j Nachbar von i

Diese Formel ergibt sich, durch die bedingten Wahrscheinlichkeiten nach Bayes (**Form. 7**).

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} = \frac{P(B|A) * P(A_i)}{\sum_{k \neq i} P(B|A_i) * P(A_i)} \tag{7}$$

Formel der bedingten Wahrscheinlichkeiten nach Bayes

An der Form ist erkenntlich, dass die Verteilung der Wahrscheinlichkeiten eine Normalverteilung der Punkte nach Gauss angenommen wird und keine normalisierte geodätische Distanz zwischen den Punkten bestimmt wird. σ ist hier ein Parameter, welcher die Breite der GaussKurve so anpasst, dass die Perplexity Bedingungen eingehalten werden.

Anpassung der Globalität der Verteilungen durch Perplexity

Im Detail wird σ in (**Form. 8**) so angepasst, dass die Bedingung mit der übergebenden Perplexity PP übereinstimmt. Dr. Dan Jurafsky aus (vgl. Snyder, 2025, o.S).

$$PP = \sqrt[N]{\prod_{i=i}^{N} \frac{1}{P(x_i|x_{i+1}...x_n)}} \tag{8}$$

Perplexity Begrenzung t-SNE

Wenn σ steigt, wird die Verteilung breiter und die Perplexity Bedingung wird breiter. Daraus lässt sich ableiten, dass die Bedingungen globaler werden und sich nicht auf kleinere Teilprobleme versteifen, wenn die Perplexity PP steigt. Bei dichteren Bereichen wird σ zudem kleiner als bei weniger dichten (vgl. van der Maaten, 2014, S.5).

Um die Verteilungsmatrix zu erhalten werden nun paarweise alle Distanzen mittels der Formel (**Form. 9**) errechnet.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{9}$$

Gleichheitsmaß i und j t-SNE

Optimierung der Abstände im niedrigdimensionalen Raum

Wie bei UMAP sind die Datenpunkte in der niedrigen Zieldimension mit der Menge Y beschrieben. Ziel ist es wieder zu versuchen die Verteilung der Wahrscheinlichkeiten p_{ij} mit einer anderen Verteilung basierend auf den Y Werten bestmöglich nachzubilden.

Dabei wird bei t-SNE folgende Verteilung (**Form. 10**) verwendet, um die Wahrscheinlichkeit von j als Nachbarn von i auf Basis von den Y Koordinaten zu errechnen, welche auf einer breiten t-Verteilung aufbaut (vgl. van der Maaten, 2014, S.5).

$$q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}} \tag{10}$$

Verteilung auf niedriger Dimension

Mithilfe dieser Verteilung kann wiederum die Kullback Leibler Divergence (**Form. 11**) berechnet werden, welche die Unterschiede von zwei Verteilungen bestimmt und hier als Lossfunktion herangezogen wird. Die Formel hierführ ist:

$$C = \sum_{i \neq j} p_{ij} log(\frac{p_{ij}}{q_{ij}}) \tag{11}$$

Kullback Leibler Divergenz

Mit der partiellen Ableitung dieser Funktion nach $\frac{\delta C}{\delta y_i}$ wird dann y_i iterativ so optimiert, dass die Verteilung Q im niedrigdimensionalen Raum bestmöglich die Ursprünglich Verteilung P nachbildet (vgl. van der Maaten, 2014, S.5).

2 Installation

Hinweis: Die Shell Script Befehle sind in der Readme des Repostitories besser zu kopieren als von diesem Dokument.
Für die Umsetzung in PG Vector muss angeführt werden, dass das Programm auf einen Linux basierten Computer angefertigt wurde und daher diese Anleitung für Linux systeme geeignet ist.

Installation von Postgres

Für die Installation von Postgres wurde der folgender cli Command durchgeführt, um das Postgres package aus dem Manager APT herunterzuladen:

```
$ sudo apt-get install postgresql
```

Installation von PGVector

Um die Erweiterung PGVector für postgres herunterzuladen, wird erst ein Docker Image heruntergeladen und dann ein Docker container erstellt und gestartet.

```
$ docker pull ankane/pgvector
$ sudo ss -lptn sport = :5432 # Kill Process if port already blocked
$ sudo docker run -e POSTGRES_PASSWORD=PW -p 5432:5432 ankane/pgvector
```

Dannach kann in PGAdmin die Extension hinzugefügt werden.

Klonen des GitHub Repostitories

Das GitHub Repository kann mittels folgenden Command geklont werden.

```
$ git clone https://github.com/ToniGGR/Data_Management_UMAP_TSNE.git
```

Installation von PGAdmin

Den GUI Managemer für Postgres (PGAdmin) herunterzuladen wurde ein virtuelles environment in Python erstellt und dann aktiviert.

```
$ sudo mkdir /var/lib/pgadmin
$ sudo mkdir /var/log/pgadmin
$ sudo chown \$USER /var/lib/pgadmin
$ sudo chown \$USER /var/log/pgadmin
$ python3 -m venv pgadmin4
$ source pgadmin4/bin/activate
$ pip install pgadmin4
$ pgadmin4
```

Einrichten eines virtuellen Environmentes

Mit dem Command

```
$ python -m venv venv
```

wird das venv im directory ./venv angelegt. Dieses kann durch den Command

```
$ source venv/bin/activate
```

gestartet werden.

Installation der Biblitoheken

Die Bibliotheken können durch die requirements txt mit dem Command:

```
$ pip install -r /path/to/requirements.txt
```

installiert werden.

Einrichten der User Credentials

Die .env Datei muss erstellt werden und folgende Struktur aufweisen.

postgres_user='your_user'
postgres_pw='your_pw'

Datengrundlage

Die Datengrundlage ist ein von Kaggle heruntergeladener Datensatz (siehe Vergnou), welcher

in eine CSV Datei entpackt worden ist und im dem Directory /data abgelegt wurde.

Datenbankkonfiguration
Für die Umsetzung der Datenbank muss einerseits die PGVector Erweiterung heruntergeladen und über PGAdmin aktiviert sein. In der erstellten Datenbank 'project_vector' werden alle aufbauenden Relationen im Code erstellt.

```
$ python code/fill_table.py
```

Jupyter Notebooks
In dem Directory code/ können dannach die Notebooks UMAP.ipynb und TSNE.ipynb gestartet werden. Erst nach dem Durchlaufen beider Notebooks können die Daten im Analysis.ipynb Notebook analysiert werden.

3 Umsetzung Beispiel Musikdaten

3.1 Dimensionsreduktion mit UMAP und t-SNE

Der vorliegende Datensatz für die praktische Umsetzung stammt von der Plattform Kaggle und umfasst Daten und die Charakteristiken verschiedener Musikstücke. Der Aufbau des Datensatzes ist dabei in der Tabellen (**Tab. 3.1**) (**Tab. 3.2**) zu sehen.

Tab. 3.1: Datenbasis - Teil 1

dance	energy	key	loudness	mode	speech	acoustic	instrumental
803	624	7	-6764	0	0.0477	451	0.000734
762	703	10	-7951	0	306	206	0.0
261	0.0149	1	-27528	1	0.0419	992	897

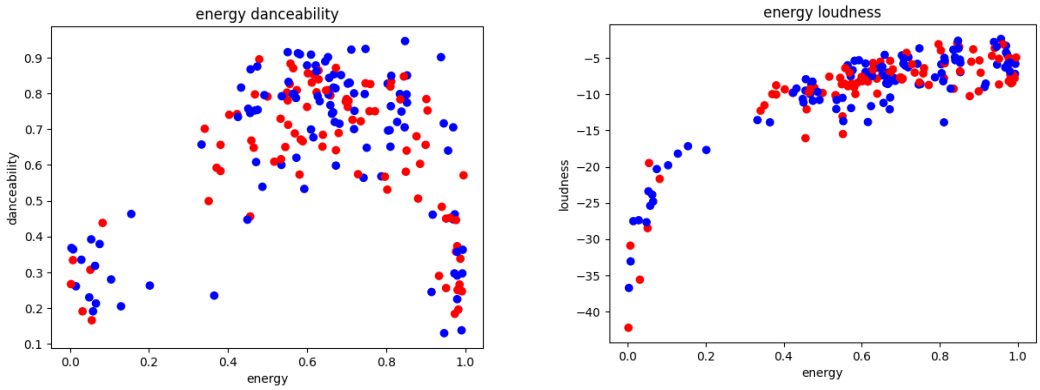
Tab. 3.2: Datenbasis Sample - Teil 2

liveness	valence	tempo	duration_ms	time_signatur	liked
0.1	628	95968	304524	4	0
0.0912	519	151329	247178	4	1
102	0.0382	75296	286987	4	0

Alle Features bestehen dabei wie in der Tabelle abzulesen ist, aus numerischen Features.

Ziel ist es basierend auf den oben vorgestellten Algorithmen die Daten von diesen 14 Features auf weniger Dimensionen so zu reduzieren, dass die Daten visualisiert werden und Informationen über Cluster und die Unterschiede dieser herausgefunden werden können.
Für den ersten Überblick über die Daten werden alle möglichen Kombinationen der Features in Scatterplots untersucht. Die y und x-Achsen sind die in dem Scatterplot verglichenen Features und die Farbe zeigt die Tonart Dur (blau) oder Moll (rot). So sind also drei Dimensionen

in den Scatterplots vertreten. In den allermeisten Scatterplots sind keine markanten Gruppierungen in den Daten erkennbar, weswegen zu vermuten ist, dass es einer Dimensionreduktion erfordert, um die Songs in den Daten komplett zu verstehen. Exemplarisch sind in (**Abb. 3.1**) zwei Scatterplots zu sehen, welche eine leichte Gruppierung der Daten erahnen könnten.



(a) Energy Danceability **(b)** Energy Loudiness

Abb. 3.1: Musikdaten nach Energy

Mit diesen beiden Scatterplots könnte zu erahnen sein, dass es in den Songs eine Gruppe an Songs gibt, die dem klassischen Genre angehören, und leiser sind, weniger Energie versprühen und weniger zum Tanzen anregen, da die Beats und das Tempo womöglich verlangsamt ist. Trotz diesen Hypothesen ist ohne eine Dimensionreduktion keine komplette Analyse der Daten möglich.

Im konkreten wird der t-SNE Algorithmus mit der Library Scikit Learn angewendet und der UMAP Algoritmus mittels der Bibliothek umap-learn implementiert, um die Dimensionsreduktion durchzuführen. Nachdem die Rohdaten aus der Vektordatenbank geladen werden, wird der Vektor, welcher alle Daten umfasst, in ein Dataframe umgewandelt und wieder die urprünglichen Spaltennamen mitgegeben. So wird jede numerische Spalte des Dataframes mit folgender Formel (**Form. 12**) skaliert (vgl. Scikit-Learn, o.S).

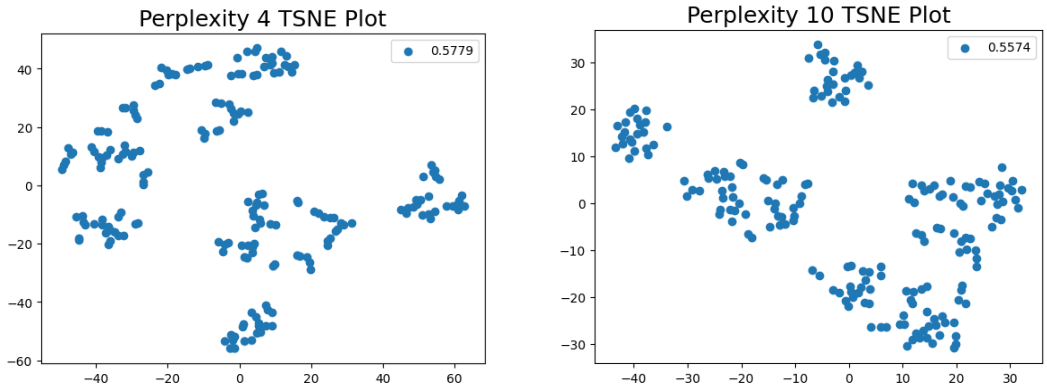
$$z = (x - \mu) * \sigma \tag{12}$$

Skalierung des Standardscalers Scikit Learn

t-SNE

Die Perplexity wird jeweils von 2 auf 26 erhöht und mit der KL Divergenz als Legende das 2-dimensionale Scatterplot ausgegeben, sodass evaluiert werden kann, wie die Parameter bestmöglich eingestelllt werden sollen. Die maximale KL Divergenz wird bei t-SNE bei einer Perplexity von vier erreicht (Wert 0,5779) und mit höherer Perplexity sinkt der Wert stetig weiter. Jedoch wirkt es so als ob der Algorithmus bei einer Perplexity von 4 den Fokus stark auf lokale Cluster legt, was bei einer Perplexity von 10 deutlich globaler ausschaut(**Abb. 3.2**).

Bei einer Perplexity von 10 können im Groben 5 Klassen relativ klar unterschieden werden. Mit



(a) t-SNE mit Perplexity 4 **(b)** t-SNE mit Perplexity 10

Abb. 3.2: t-SNE Reduktion nach Perplexity (Legende = KL Div)

dieser Verteilung kann beispielsweise der KMeans Clusteralgorithmus angwendet werden und die Punkte in die 5 Cluster eingeteilt werden(**Abb. 3.3**).

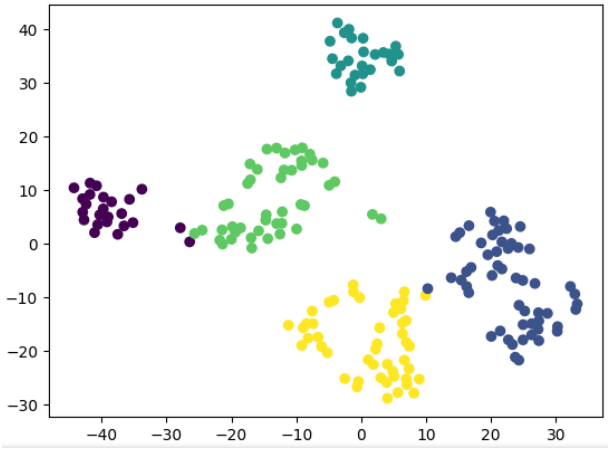


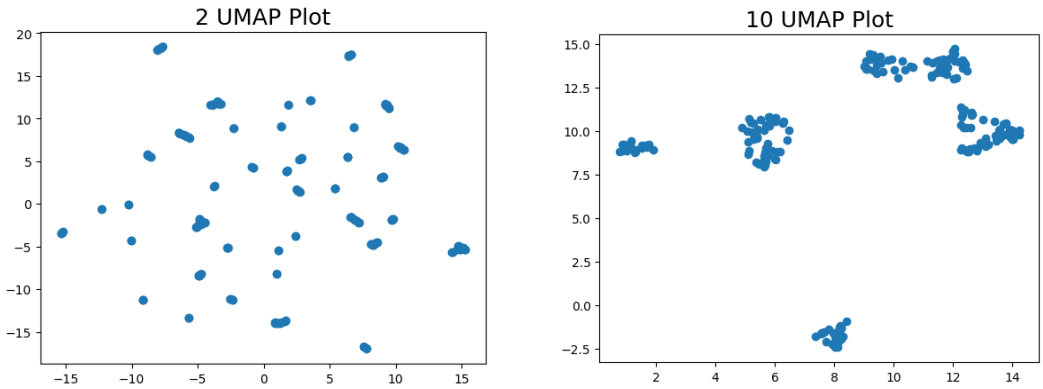
Abb. 3.3: 5 KMeans Cluster bei t-SNE mit Perplexity 10

Nachdem die Einteilung eingenommen wird, wird in postgres eine neue Relation erstellt und die Klassen mit den jeweiligen Song IDs hineingeschrieben, um präzisere Auswertung durchzuführen.

UMAP

Für UMAP wird eine ähnliche Vorgehensweise durchgeführt, nur, dass bei UMAP nicht die Perplexity erhöht wird, sondern die betrachtete Neighbour size. Hier ergeben sich ähnliche Muster wie bei t-SNE, da durch eine niedrige Neighbour Size eher auch viele kleine, lokale Cluster entstehen und mit höhrere die Cluster immer gloabler und größer werden, wie in (**Abb. 3.4**) zu sehen ist.

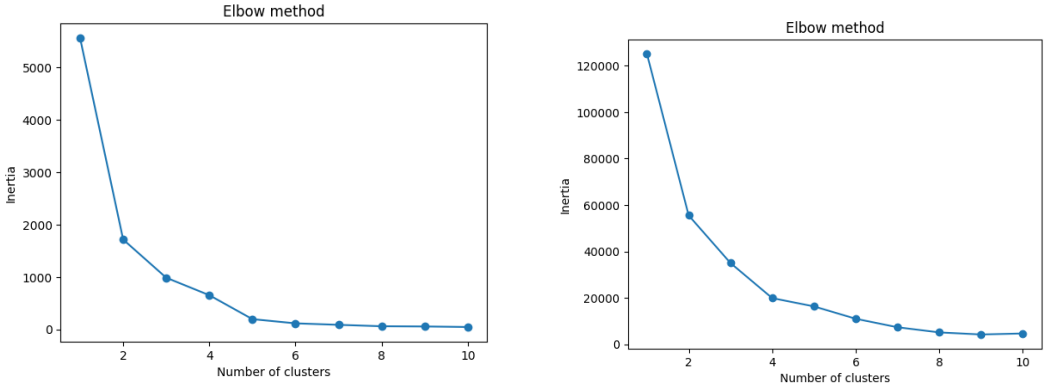
Um die Anzahl an Clustern zu bestimmen wird die Ellbow Methode mit einer steigenden KMeans Cluster Größe verwendet und die passende Kurven aufgezeichnet (**Abb. 3.5**). Um die Clustergröße bei beiden Algorithmen gleich zu behalten muss eine gute Lösunge für beide Algorithmen gefunden werden. Bei UMAP liegt die optimale Clustergröße bei 5 während bei t-SNE ein Wert zwischen 4 und 5 optimal wäre. Für die weitere Analyse wurde daher entschieden



(a) UMAP mit Neighbour Size 2 **(b)** UMAP mit Neighbour Size 10

Abb. 3.4: UMAP Reduktion nach Neighbour size

bei beiden Algorithmen 5 Cluster zu bestimmen.



(a) Ellbowkurve UMAP mit Neghbour Size 10 **(b)** Ellbowkurve t-SNE mit Perplexity 10

Abb. 3.5: UMAP Reduktion nach Neighbour size

Nach der Anwendung des KMeans Clusters mit 5 Gruppen ergeben sich folgende Cluster (**Abb. 3.6**):

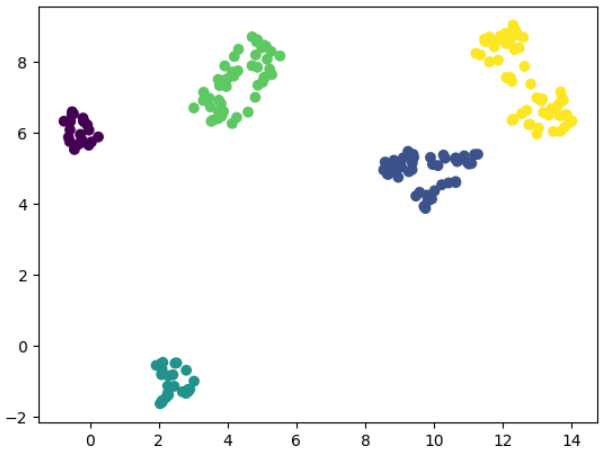
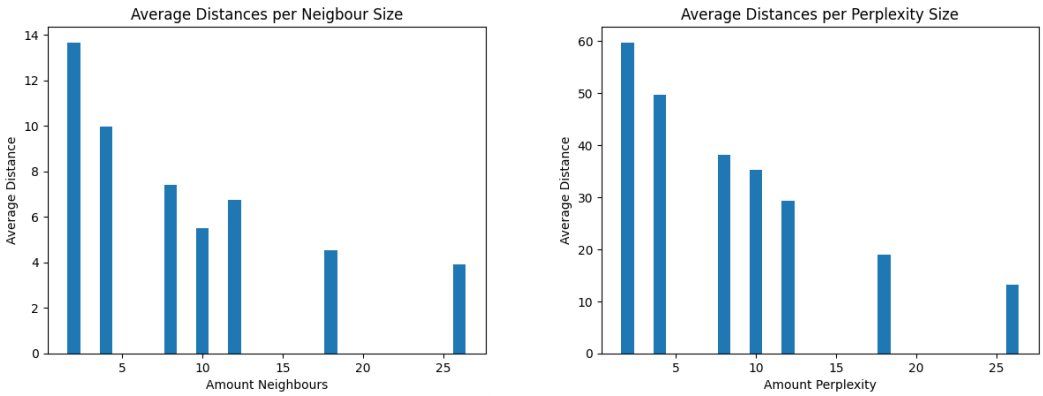


Abb. 3.6: KMeans Cluster bei UMAP mit Neighbour size 10

Auswirkungen der Parameter

Dass bei UMAP und t-SNE die Cluster mit steigender Perplexity (t-SNE) und Neighbour Size (UMAP) zu globaleren Clustern führen kann identifiziert werden, indem die durchschnittliche Distanz der Punkte zueinander für eine 2 dimensionale Reduktion pro Parameterwert geloggt wird. Die dabei resultierende Grafiken sind in (**Abb. 3.7**) zu sehen.



(a) UMAP Neighbour Size Distanzen **(b)** t-SNE Perplexity Distanzen

Abb. 3.7: Durchschnisstabstände pro Parameter

Dabei sinken bei beiden Algorithmen die Distanzen mit zunehmender Perplexity und Neighbour size näherungsweise vergleichbar mit einer Potenzfunktion mit einem negativen Exponenten. Bei steigenden Paramater rücken also die Punkte immer näher zusammen.

3.2 Auswertung der Cluster

Um genauer zu analysieren, welche verschiedenen Cluster es in den Daten gibt, werden die Mittelwerte der Features je nach eingeordneter Klasse untersucht. Die komplette Auswertung der Features ist im Anhang für t-SNE unter (**Abb. 5.1**) und für UMAP in (**Abb. 5.2**) zu sehen. Zusammengefasst haben die Cluster bei t-SNE folgende Merkmale

Cluster nach T-SNE

Cluster 0 : Annahme: klassische Musik

- Farbe Rot
- kaum Tanzbarkeit
- wenig Energie
- viel Akustik
- viele Instrumente
- Valenz niedrig (klingt negativ)
- kaum Gesang
- leise

Cluster 1 : Annahme: Pop

- Farbe Blau

- Tanzbarerkeit hoch
- laut
- viel Gesang
- schnell
- Nur Dur
- Dauer kurz

Cluster 2: Annahme: Rock

- Farbe Grün
- Tanzbarkeit niedrig
- laut
- viel Energie
- viele Instrumente
- keine Aktustik
- mittel Gesang
- Liveiness Hoch

Cluster 3: Annahme: Soft Rock

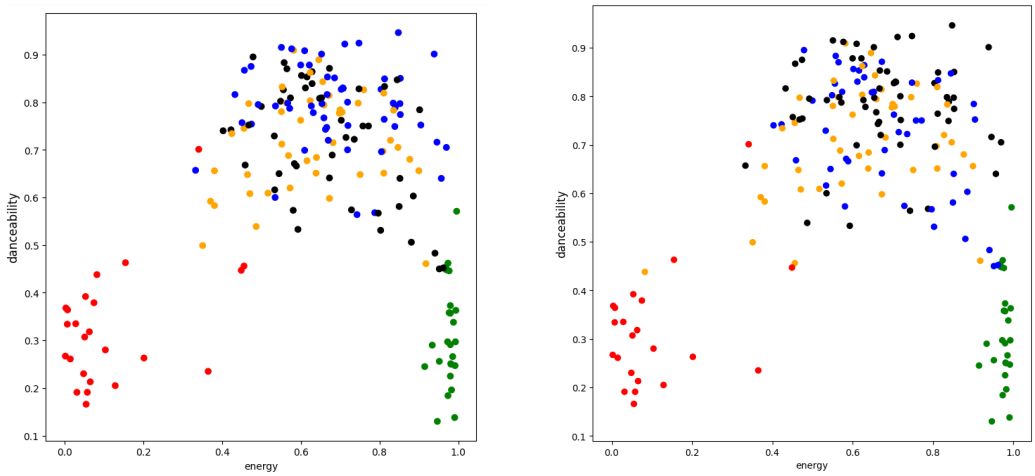
- Farbe Orange
- Tanzbarkeit hoch
- Kaum Gesang
- laut
- Sehr postive Stimmung

Cluster 4: Annahme: Rap

- Farbe Schwarz
- Hohe Tanzbarkeit
- laut
- hoher Sprachanteil
- kaum Akustik
- schnell
- keine Instrumente
- Moll Lastig

Durch die t-SNE Dimensionsreduktion lassen sich nun also verschiedene Musikrichtungen ableiten, die auf den ersten Blick nicht erkennbar waren. UMAP findet ein sehr ähnliches Ergebnis, wie t-SNE, weshalb auf die explizite Beschreibung der 5 UMAP Cluster verzichtet wird. Im Anhang sind die UMAP Cluster detailliert unter (**Abb. 5.2**) zun finden.

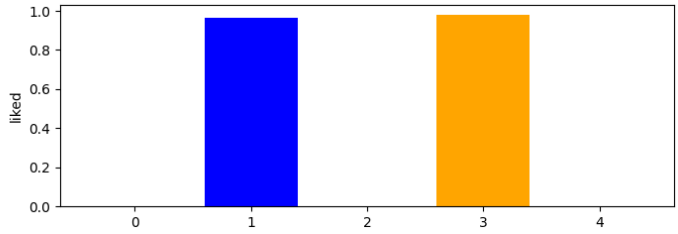
Wird nun das anfangs gezeichnete Scatterplot mit den Klassen eingezeichnet, können einerseits die vorher bestimmten Hypothesen erkannt werden. Die Cluster mit niedriger Energie und niedriger Tanzbarkeit gehören einer anderer Musikrichtung an. Es kann auch gesehen, dass die Blaue, orangene und schwarze Klasse sich zwischen Tanzbarkeit und Energie nicht unterscheiden, wie in(**Abb. 3.8**) zu sehen ist.



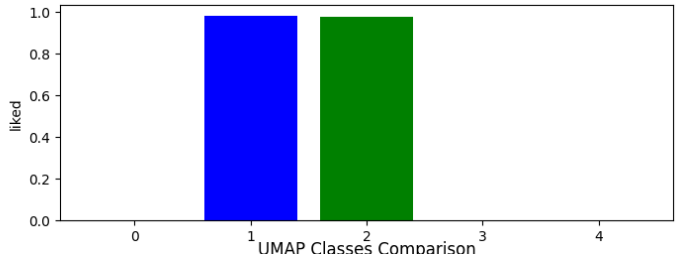
(a) t-SNE Cluster Danceability und Energy **(b)** UMAP Cluster Danceability und Energy

Abb. 3.8: Energie und Danceability nach UMAP und t-SNE Klassen

Beim Vergleich der beiden Abbildungen in (**Abb. 3.8**) kann festgestellt werden, dass UMAP und t-SNE auf sehr ähnliche Cluster kommen und sich dementsprechend kaum unterscheiden. Beide finden beispielsweise Cluster, welche fast alle gelikten Songs enhtalten, wie in folgenden Grafiken (**Abb. 3.9**) erkenntlich ist, wo auf der y-Achse der durschnittliche Like Wert (0,1) im Cluster und auf der x-Achse die fünf Cluster visualisiert werden. Bei beiden Clustern ist der durchschnittliche Wert der liked Variable nahe Eins und bei den anderen Clustern Null was bedeutet, dass der User möglicherweise nach Genre Songs geliked hat. Die beiden Cluster gehören zu den igenständig gelabelten Musikgenres Soft-Rock und Pop.
Eine Analyse der Klassen auf den durch die Algorithmen entstehenden Dimensionen wird in folgenden Grafiken für UMAP und für t-SNE angeführt (**Abb. 3.10**) und (**Abb. 3.11**). Die Beschriftung der Cluster basiert auf den Merkmalsausprägungen der Songs, die oben für t-SNE Cluster bestimmt wurden und diese wurden intuitiv den Clustern zugeordnet.

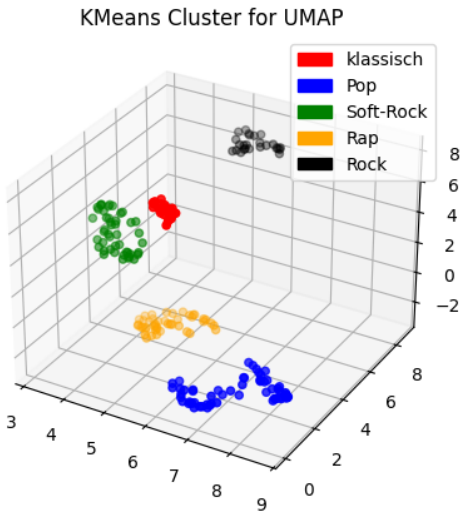


(a) Avrg Like per Cluster (t-SNE)

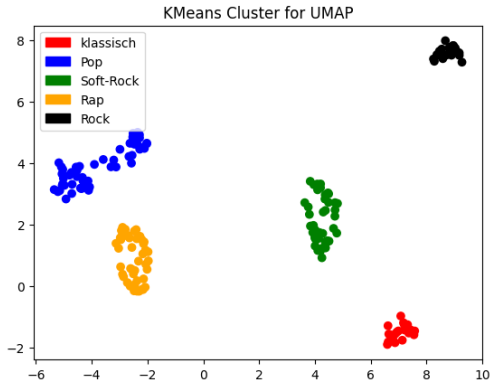


(b) Avrg Like per Cluster (UMAP)

Abb. 3.9: Energie und Danceability nach UMAP und t-SNE Klassen



(a) Clusters in UMAP 3 Dimensional Scatter-plot

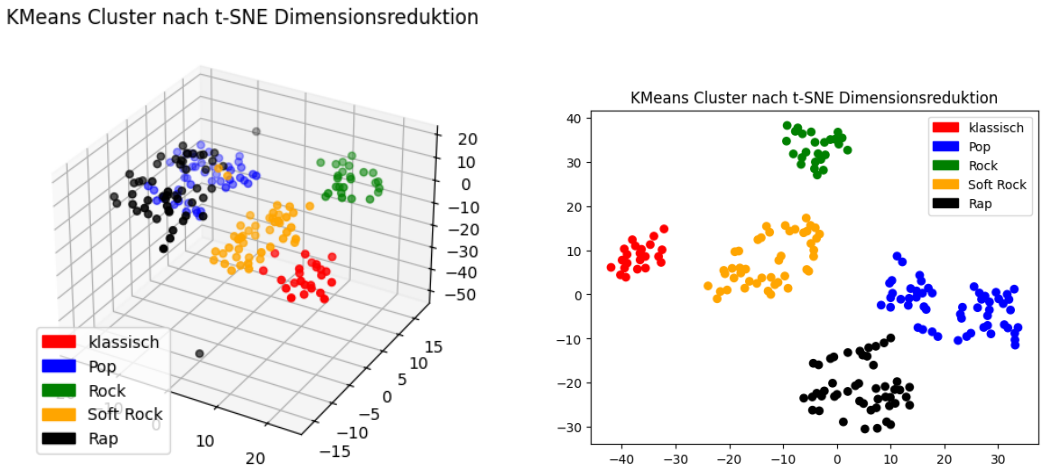


(b) Clusters in UMAP 2 Dimensional Scatter-plot

Abb. 3.10: Energie und Danceability nach UMAP und t-SNE Klassen

4 Fazit

Zusammenzufassend kann angeführt werden, dass es mit beiden Algorithmen möglich ist, die den gegebenen Datensatz in eine niedrigere Dimension herunterzubrechen und daraus Erkenntnisse über die Songs zu erhalten. Bei beiden Algorithmen treten am Ende ähnliche Cluster auf, die ähnliche Attribute hatten und so zu Musikgenres zugeordnet werden konnten. In beiden Algorithmen werden beispielsweise die Pop und Rap Cluster relativ nah aneinandergezogen. Gleiches gilt bei SoftRock und dem Cluster der klassischen Musik. Unterschiede, die sich beim aktuellen Vergleich zeigen, ist dass die Cluster von der aktuellen UMAP Integration mit einer Neighbours Size von 10 die Cluster klarer zu trennen scheint als t-SNE mit einer Perplexity



(a) Clusters in t-SNE 3 Dimensional Scatterplot

(b) Clusters in t-SNE 2 Dimensional Scatterplot

Abb. 3.11: Energie und Danceability nach UMAP und t-SNE Klassen

10. Da könnte der Parameter Perplexity ein bisschen nach unten gesetzt werden, um die Cluster ähnlich wie UMAP stark voneinander abzugliedern. Ein Grund für die klarere Trennung von UMAP kann auch sein, dass die Datenpunkte weit auseinander waren und so die geodätische Distanz einen bessere Repräsentation der wirklichen Datenpunkte geschaffen hat als die euklidische Distanz bei t-SNE.

5 Anhang

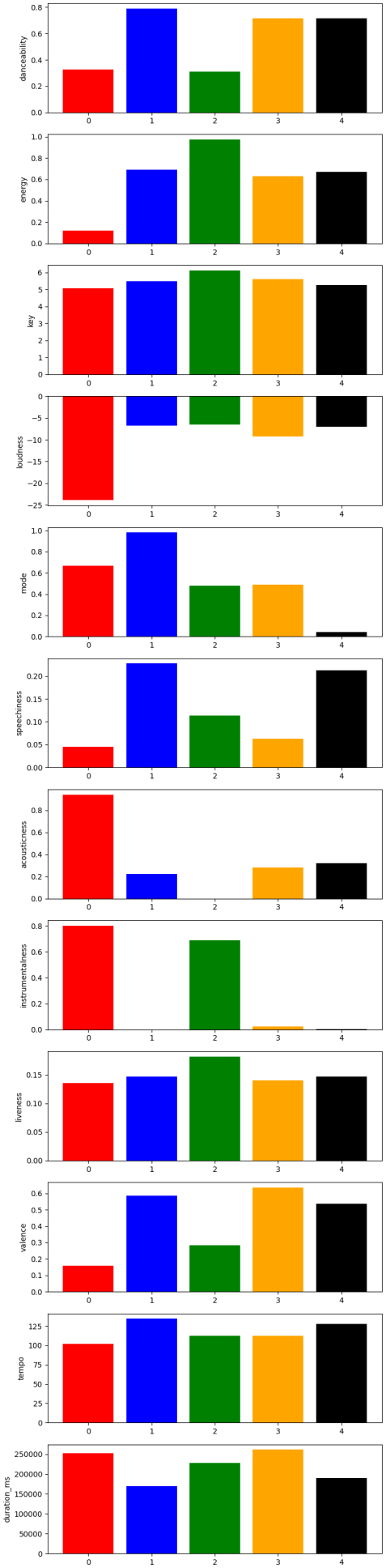


Abb. 5.1: t-SNE Cluster

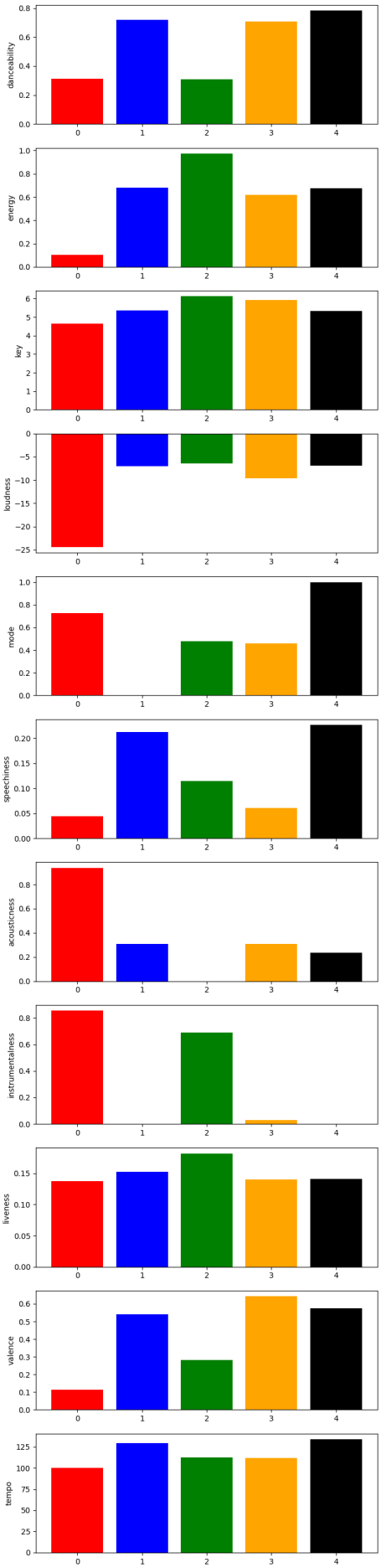


Abb. 5.2: UMAP Cluster

Literatur

[Cai und Ma 2022] CAI, T. T. ; MA, Rong: *Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data*. November 2022. – URL http://arxiv.org/abs/2105.07536. – Zugriffsdatum: 2025-03-02. – arXiv:2105.07536 [stat]

[Cheng et al. 2011] CHENG, Zhe ; FREEDMAN, Michael J. ; REXFORD, Jennifer: CASS: Efficient Similarity Search for Large-Scale Data. In: *Proceedings of the 20th International Conference on World Wide Web*, URL https://www.cs.princeton.edu/cass/papers/www11.pdf, 2011, S. 1–10

[van der Maaten 2014] MAATEN, Laurens van der: Accelerating t-SNE using Tree-Based Algorithms. In: *Journal of Machine Learning Research* 15 (2014), S. 3221–3245. – URL https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf

[McInnes et al. 2020] MCINNES, Leland ; HEALY, John ; MELVILLE, James: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. September 2020. – URL http://arxiv.org/abs/1802.03426. – Zugriffsdatum: 2025-03-02. – arXiv:1802.03426 [stat]

[Scikit-Learn] SCIKIT-LEARN, developers: *Scikit Learn Documentation - Standardscaler*. – URL https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. – Zugriffsdatum: 2025-03-03

[Snyder 2025] SNYDER, Wayne: *Perplexity and Its Applications in Natural Language Processing*. 2025. – URL https://www.cs.bu.edu/fac/snyder/cs505/PerplexityPosts.pdf

[Vergnou] VERGNOU, Brice: *Spotify Recommendation*. – URL https://www.kaggle.com/datasets/bricevergnou/spotify-recommendation. – Zugriffsdatum: 2025-03-03

KI Nutzung

KI basiertes Hilfsmittel	Einsatzform	Betroffene Teile der Arbeit
ChatGPT 4.o	Erstellung von Code für KL Divergenz	Kapitel 3

Selbständigkeitserklärung Anton Geiger

Ich versichere hiermit, dass ich, Anton Geiger, meine Seminararbeit

Visualisierung von Musikdaten mittels UMAP und t-SNE

selbständig verfasst, die digitale Version mit der ausgedruckten
Version übereinstimmt und keine anderen als die angegebenen
Quellen und Hilfsmittel benutzt habe.

Ort, Datum

Unterschrift