# HTL - Perg
## Höhere Abteilung für Informatik

# Diplomarbeit

## Communicating Emotion by Playing the Piano


Projektteam: Christina Brandstetter
Katharina Praher
Projektbetreuer: Prof. Ing. Patrick Praher, MSc, BSc

In Zusammenarbeit mit dem Institut für Computational Perception der JKU
Verena Praher, MSc
Dr. Emilia Parada-Cabaleiro
Dr. Shreyan Chowdhury


Bearbeitungszeitraum: 12.12.2022 – 01.04.2023

# 1 Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

Christina Brandstetter

Perg, _____ Unterschrift _____

Katharina Praher

# 2 Gendererklärung

Im Sinne der besseren Lesbarkeit werden in dieser Diplomarbeit personenbezogene Bezeichnungen, die sich zugleich auf Frauen und Männer beziehen, generell nur in der im Deutschen üblichen maskulinen Form angeführt. Dies soll jedoch keinesfalls eine Geschlechterdiskriminierung oder eine Verletzung des Gleichheitsgrundsatzes zum Ausdruck bringen.

Perg, _____ Unterschrift _____

Christina Brandstetter

Perg, _____ Unterschrift _____

Katharina Praher

# 3 Danksagung

Wir möchten uns bei sämtlichen Personen und Organisationen, die uns diese Diplomarbeit ermöglicht haben, bedanken.

Besonders erwähnen möchten wir hier unseren Diplomarbeitsbetreuer Herrn Professor Ing. Patrick Praher, MSc, BSc der uns stets mit Rat und Tat zur Seite gestanden ist. Als treibende Kraft hinter der Diplomarbeit ist er es gewesen, der uns auf die Möglichkeit einer Zusammenarbeit mit der JKU aufmerksam gemacht hat. Durch seinen Hintergrund in der Bioinformatik hat er stets den perfekten Ansprechpartner für das Thema Machine Learning geboten und hat uns so immer weiterhelfen können.

Weiters bedanken wir uns sehr herzlich bei unseren Betreuern und Unterstützern vom Institute for Computational Perception. Hier gilt unser Dank besonders Frau Verena Praher, MSc, die uns vor allem in der Anfangsphase der Zusammenarbeit sehr gut betreut hat. Während unseres Praktikums sind uns zudem Frau Dr. Emilia Parada-Cabaleiro und Herr Dr. Shreyan Chowdhury stets zur Verfügung gestanden, denen wir für die umfassende Unterstützung hier nochmals ausdrücklich danken möchten.

Unser Dank gilt in weiterer Folge allen Freunden und Kollegen, die bereit gewesen sind, für uns etwas am Klavier einzuspielen und somit zur Erstellung unseres Datensets beigetragen haben.

Abschließend gilt unser großer Dank allen Bekannten, Verwandten und Freunden, die uns bei der Korrektur der Diplomarbeit unterstützt haben.

# 4 Kurzfassung

CEPP steht für Communicating Emotion by Playing the Piano, was mit Kommunikation von Emotionen beim Spielen des Klaviers übersetzt wird. Ziel ist eine Anwendung, die Benutzereingaben über eine Klaviatur verarbeitet und dabei feststellt welche Emotionen der Benutzer ausdrücken möchte.

Zur Realisierung werden dabei drei grundlegende Komponenten benötigt:

Die **Kommunikation** zwischen Computer und E-Piano mittels MIDI stellt sicher, dass Benutzereingaben verarbeitet werden können. Die Anwendung teilt den eingehenden MIDI-Stream in kleine Abschnitte, welche anschließend analysiert werden.

Durch ein **Machine Learning-Modell**, das mit einem öffentlichen Datenset trainiert wurde, wird diesen Informationen eine Emotion zugeordnet. Im Rahmen der Arbeit wurden verschiedene Modelle getestet und verglichen. Zur Auswahl stehen dabei vier Emotionen (Happy, Angry, Sad, Tender), da das genutzte Datenset in dementsprechende Quadranten unterteilt ist. Die Untersuchung und die Bewertung dieser bestehenden Daten stellen einen weiteren zentralen Punkt der Arbeit dar. Besonders die Festlegung und Gewichtung von sogenannten Features, also Eigenschaften der MIDI-Dateien, hat einen großen Einfluss auf das spätere Modell.

Das Modell klassifiziert die aufgenommenen Dateien und liefert eine Kategorie, welche über eine **grafische Benutzeroberfläche** angezeigt wird. Dabei erscheint ein farblich gekennzeichnetes Label, welches die festgestellte Emotion beschreibt.

Das Ergebnis ist eine Gesamtanwendung, die sich aus Python Modulen und PowerShell Skripten zusammensetzt. Das optimierte Modell stellt einen wichtigen Teil dieser Anwendung dar. Weiters wurden zur Erweiterung und zum Vergleich mit dem verwendeten Datenset eigene Daten aufgenommen, die zum Training verschiedener Modelle zum Einsatz gekommen sind.

# 5 Abstract

CEPP is short for Communicating Emotion by Playing the Piano. The goal of the project is to recognise the intended emotion of someone playing the piano by analysing the generated user input.

The project is made up of three main components:

**Communication** between the Computer and the E-Piano via MIDI ensures, that user input can be processed. The application cuts the incoming MIDI-data into small segments, which are then further analysed.

A **machine learning model** that was trained with a public dataset determines an emotion for these segments. Part of the project was comparing and testing different model classes. There are four emotions available, namely Happy, Angry, Sad and Tender. These four were chosen, since the used dataset is split into corresponding quadrants. The exploration and assessment of the existing data prove to be another central part of the project. Especially determining and evaluating different features for the available MIDI-data influences the resulting model greatly.

The model classifies the recorded data and provides a category, which is displayed over a **graphical user interface**. The depiction consists of a colour-coded label and describes the determined emotion.

The end result consists of the full application, which is made up of python modules and PowerShell scripts. The optimised model is an important part of this application. Furthermore, additional data was recorded to enhance the existing dataset and train different models.

# 6 Inhaltsverzeichnis

# 7 Einleitung

## 7.1 Motivation

Das CP-Institut der JKU beschäftigt sich mit künstlicher Intelligenz und Machine Learning, wobei der Schwerpunkt der Forschung des Instituts auf intelligenter Audio- und Musikverarbeitung liegt. Daraus entstand auch die Idee für die Diplomarbeit. Diese soll es einem Computer ermöglichen, mit Hilfe von Machine Learning in Echtzeit festzustellen, welche Emotion ein Klavierspieler spielt bzw. versucht zu spielen.

## 7.2 Zielsetzung

Ziel ist es, eine Applikation zu entwickeln, die Userinput über eine Klaviatur verarbeitet und dabei feststellt welche Emotion der User ausdrücken möchte.

Mit dem Projekt werden keine unternehmerischen oder kommerziellen Ziele verfolgt. Es handelt sich dabei um eine Forschungsarbeit, mit dem Ziel eine betriebsbereite Anwendung zu entwickeln, die als interaktives Schauobjekt, Proof-of-Concept, Prototyp oder Referenz für ähnliche Projekte in der Zukunft verwendet werden kann.

## 7.3 Projektinhalt – Überblick

Das Projektergebnis im Sinne der Anwendung kann in drei wichtige Komponenten unterteilt werden:

### 7.3.1 Kommunikation zwischen Computer und Klavier mittels MIDI

Zwischen dem Computer, auf dem das Projekt läuft und dem angeschlossenen Klavier (bzw. digitalem MIDI-Keyboard) wird eine Verbindung hergestellt, sodass die Daten live übertragen und verarbeitet werden können.

Aufgrund der notwendigen Analyse des MIDI-Materials kommt es dabei zwangsläufig zu Verzögerungen. Allerdings sollen diese möglichst geringgehalten werden und vom Anwender nicht störend empfunden werden.

### 7.3.2 Emotion Prediction

Die Vorhersage bzw. die Bestimmung von Emotionen basierend auf Daten, die aus MIDI-Dateien entnommen werden können, stellt die Kernfunktion des Projekts dar.

Dabei werden verschiedene Ansätze erprobt, bewertet und angewendet. Dazu zählen Versuche mit sogenannten „Handcrafted Rules", also manuell festgelegten Regeln und der datengetriebene Ansatz, der sich die Analyse durch Machine Learning zu Nutze macht.

Zusätzlich werden unterschiedliche Arten von Modellen auf Effizienz, Treffsicherheit und Verständlichkeit überprüft und getestet, um das optimale Modell zu finden und zu trainieren.

#### *7.3.2.1 Welche Emotionen sollen dabei erkannt werden?*

Aufgrund des verwendeten Datensets, das die Daten in vier Gruppen aufteilt, wird die Anwendung auf vier Emotionen eingeschränkt:

- Happy (Fröhlich)
- Angry (Wütend)
- Sad (Traurig)
- Tender (Sanft)

#### *7.3.2.2 Welche Machine Learning Algorithmen (Modelle) werden evaluiert?*

Folgende Arten von Modellen werden für die Anwendung getestet und bewertet:

- Support Vector Machine (SVM)
- Random Forrest (RF)
- K-nearest Neighbour (k-NN)
- Multilayer Perceptron (MLP)

### 7.3.3 User Interface

Die Benutzeroberfläche zeigt die bestimmte Emotion für den Benutzer an. Dabei kommen ein Label sowie eine zur Emotion passende, vorbestimmte Farbe zum Einsatz.

Die Anzeige aktualisiert sich automatisch mit neuem Input und reagiert somit in Echtzeit auf die Eingaben des Benutzers.

### 7.3.4 Datenerhebung

Ein weiteres Projektergebnis stellt die Datenerhebung dar. Um ein Machine Learning-Modell zu trainieren, benötigt es sehr viele Datensätze, dazu werden im Zuge des Projekts auch eigene Datensätze aufgenommen. Es handelt sich dabei um MIDI-Files, die auf einem E-Piano von Freiwilligen eingespielt, gelabelt und abgespeichert werden. Diese „hausgemachten" Datensätze bieten einen Kontrast zu den professionell aufgenommenen Daten des verwendeten EMOPIA Datensets und kommen beim Test von Modellen zum Einsatz.

## 7.4 Projektumfeld

### 7.4.1 Projektteam

Das Projektteam besteht aus Christina Brandstetter und Katharina Praher, zwei Schülerinnen der HTL Perg. Da Christina Brandstetter schon seit mehr als acht Jahre Klavier spielt, ist sie für den musiktheoretischen Teil der Diplomarbeit zuständig gewesen. Katharina Praher hingegen ist mit ihrem großen Interesse für Machine Learning für das Trainieren und Vergleichen unterschiedlicher Modellklassen hauptverantwortlich gewesen.

*Abb. 1: Katharina Praher (links) und Christina Brandstetter (rechts)*

### 7.4.2 Betreuung

Betreut worden ist die Diplomarbeit von Professor Patrick Praher. Er hat das Projektteam vor allem mit seinem fachlichen Know-how im Bereich Machine Learning zusätzlich unterstützen können.

### 7.4.3 Auftraggeber

Auftraggeber ist das Institute für Computational Perception der Johannes Kepler Universität. Ein Schwerpunkt des Instituts liegt dabei auf intelligenter Audio- und Musikverarbeitung mithilfe von Machine Learning.

INSTITUTE OF COMPUTATIONAL PERCEPTION

*Abb. 2: Logo des Instituts für Computational Perception*

Am Institut selbst ist die Ansprechpartnerin für das Projektteam Verena Praher, eine Universitätsassistentin des Instituts, gewesen. Während des Sommerpraktikums im Juli 2022 ist das Projektteam zusätzlich von Shreyan Chowdhury und Emilia Parada-Cabaleiro betreut worden, wobei Emilia Parada-Cabaleiro das Projektteam im Bereich Music Information Retrieval unterstützt hat und Shreyan Chowdhury sein Wissen über verschiedenste Machine Learning-Techniken dem Projektteam nähergebracht hat.

# 8 Theoretische und fachpraktische Grundlagen und Methoden

## 8.1 Verwendete Technologien

### 8.1.1 Python

Für den größten Teil des Projekts ist Python als Programmiersprache verwendet worden. Python ist objektorientiert, interpretiert und umfasst Konzepte wie Module, Exceptiones und dynamische Datentypen. Ein Merkmal von Python ist, dass es eine sehr kompakte Syntax hat und somit Programme kurz, aber trotzdem gut lesbar sind. Ein Beispiel dafür ist, dass in Python ein Statement durch Einrückungen und nicht durch Klammern am Beginn und am Ende des Statements gruppiert wird. [1]

Der Hauptgrund für die Verwendung von Python im Projekt ist, dass es in Python viele verschiedene Module rund um Machine Learning und Data Science gibt und Python somit in diesen Bereich eine häufig verwendete Sprache ist.

### 8.1.2 Project Jupyter

Project Jupyter[1] ist ein gemeinnütziges Open-Source-Projekt und es ist entwickelt worden, um interaktives Arbeiten im Bereich Data Science und Scientific Computing zu unterstützen. Konkret ist im Projekt die Technologie der Jupyter Notebooks verwendet worden, so sind die einzelnen Machine Learning-Modelle jeweils in Jupyter Notebooks programmiert und analysiert worden. Jupyter Notebooks sind strukturierte Dateien in denen man Code, Metadaten, Text-Inhalte und Codeoutput gleichzeitig als .ipynb-Datei im JSON-Format speichern kann. Zusätzlich ist Jupyter Notebook auch ein Interface, mit dem das Notebook visualisiert, sein Code ausgeführt und zusätzlich der Output des Codes gespeichert werden kann. [2]

[11] https://jupyter.org

### 8.1.3 Flask

Flask ist ein in Python programmiertes Micro-Webframework. Es wird als ‚micro' bezeichnet, da es keine speziellen Tools oder Bibliotheken voraussetzt und selbst keine Datenbankabstraktionsschicht, Formularvalidierung etc. mitbringt. Jedoch existieren Erweiterungen für diese für Webframeworks gängigen Tools. [3] Die im Zuge der Diplomarbeit entwickelte Webapplikation baut auf Flask auf, da diese weder aus einer Datenbank besteht, noch Unterstützung für Formularverarbeitung notwendig ist.

### 8.1.4 Conda

Conda[2] ist ein open-source Package- und Environment-Managementsystem, welches zwar speziell für Python entwickelt worden ist, aber auch für viele andere Programmiersprachen genutzt werden kann. Mit Conda lassen sich Python Environments erstellen, sodass man in den verschiedenen Environments verschiedene Versionen von Python und unterschiedliche Packages installieren kann. Da im Projekt sehr viele verschiedene, zusätzliche Python-Packages benötigt worden sind, ist Conda verwendet worden, um ein eigenes Environment für das Projekt zu erstellen.

### 8.1.5 Git und GitLab

Git ist eine Open-Source-Software zur Versionsverwaltung von sowohl kleinen als auch sehr großen Softwareprojekten. Die Dateien werden dabei in einem Git-Repository gespeichert und ihr Versionsverlauf wird darin zusätzlich erfasst. Da Git ein verteiltes Versionsverwaltungstool ist, wird immer das vollständige Repository geklont und nicht, wie bei zentralen Versionsverwaltungen, ein aktueller Snapshot der Dateien am zentralen Server erstellt, wenn man als Benutzer die Dateien lokal bearbeiten will. Der Vorteil ist dabei, dass man immer noch das vollständige lokale Repository hat, sollte der Server kaputt gehen. [4]

GitLab ist eine Webapplikation zur Git-basierten Versionsverwaltung. Zusätzlich unterstützt GitLab DevOps-Prozesse, wie Issue-Tracking, Continuous Integration und Continuous Delivery. [5]

Im Zuge des Projektes ist sowohl das von der HTL Perg gehostete GitLab als auch das GitLab des Instituts für Computational Perception verwendet worden,

[2] https://docs.conda.io

wobei auf dem GitLab der HTL Perg das Repository mit dem Code der CEPP-Applikation und der Jupyter Notebooks liegt.

### 8.1.6 jSymbolic

Bei jSymbolic handelt es sich um einen Teil der Software Suite jMIR, der im Projekt zur Extraktion der Features verwendet wurde. Probleme bei der Anwendung sind unter anderem dadurch entstanden, dass jSymbolic nicht mit einem europäischen Zeit- und Datumsformat arbeiten kann. Um die Software verwenden zu können, wurden daher alle Systeme auf das amerikanische Datumsformat umgestellt. Wie die Software im Projekt genau eingesetzt wird, ist im Kapitel zur Feature Extraction 9.1.6 genauer beschrieben.

## 8.2 Verwendete Entwicklungssysteme

### 8.2.1 PyCharm

PyCharm[3] ist eine moderne Entwicklungsumgebung von JetBrains zur Entwicklung von Python-Programmen. Funktionalitäten von PyCharm sind unter anderem Code-Completion, Code-Inspection, Developer Tools, wie ein Debugger für Python und eine integrierte Versionskontrolle.

Im Projekt ist PyCharm für die Entwicklung der Applikation verwendet worden.

### 8.2.2 DataSpell

DataSpell[4] ist eine weitere Entwicklungsumgebung von JetBrains, welche speziell für Data Scientists entwickelt worden ist. Für das Projekt relevant ist dabei vorwiegend die Unterstützung von Jupyter Notebooks, da DataSpell für das Arbeiten mit Jupyter Notebooks verwendet worden ist.

[3] https://www.jetbrains.com/pycharm/
[4] https://www.jetbrains.com/dataspell/

## 8.3 Verwendete Bibliotheken und Plug-Ins

### 8.3.1 pandas

pandas selbst ist eine Library zur Analyse und Manipulation von Daten. Häufig genutzt wird dabei das ‚Data Frame'-Objekt von pandas. Ein Data Frame ist eine zweidimensionale Datenstruktur und somit vergleichbar mit einem zweidimensionalen Array oder einer Tabelle, die aus Zeilen und Spalten besteht. Der entscheidende Unterschied ist aber, dass, anders als bei zum Beispiel einem Numpy Array, welchem man nur einen Datentyp über das gesamte Array hinweg zuweisen kann, auch einzelne Spalten eines Dataframes unterschiedliche Datentypen zugewiesen werden können. Auf einem Data Frame können zusätzlich verschiedenste Datenmanipulations- und Datenselektionsmethoden ausgeführt werden. [6]

### 8.3.2 Matplotlib

Matplotlib ist eine umfangreiche Python-Bibliothek zur Visualisierung von Daten. Dabei können statistische Grafen, dynamische Animationen oder interaktive Diagramme in publikationstauglicher Qualität produziert werden.

Die verfügbaren Plots erstrecken sich dabei von sehr einfachen Diagrammen, die Werte auf einer X- und Y-Achse abbilden, bis hin zu komplexen, statistischen Darstellungsmöglichkeiten oder 3D-Abbildungen.

Besonders wichtig für das Projekt ist das Pyplot-Modul der Matplotlib. Es handelt sich um eine Sammlung von Funktionen, die ermöglichen, dass der Anwender mit der Matplotlib arbeitet wie mit MATLAB. Ein Plot verhält sich dabei wie ein Objekt und behält seine zugewiesenen Werte, welche zur Programmlaufzeit durch verschiedene Operationen verändert werden können. [7]

Bei MATLAB handelt es sich um eine kommerzielle Software, die zur Lösung von mathematischen Problemen und der grafischen Darstellung der Ergebnisse verwendet wird. Das Programm ist dabei vor allem für numerische Berechnungen mit Matrizen ausgelegt. [8]

### 8.3.3 Seaborn

Seaborn ist ebenfalls eine Library zur statistischen Datenvisualisierung und basiert auf Matplotlib- und pandas-Datenstrukturen. Seaborn ist ein Tool zum Auswerten und Verstehen von Daten.

Als eine High-Level API für statistische Grafen bietet Seaborn zahlreiche Darstellungsmöglichkeiten für Daten und die Beziehungen zwischen diesen Daten. Die Auswahl beschränkt sich dabei auf 2D Darstellungen wie beispielsweise Scatterplots, Boxplots, Joint Grids oder Clustermaps. Für das Projekt sind dabei besonders die Pair Plots relevant. [9]

### 8.3.4 Scikit-learn

Scikit-learn ist eine Python Library in der eine große Anzahl an Machine Learning-Algorithmen für mittelgroße Supervised und Unsupervised Learning-Probleme implementiert sind. Augenmerk liegt dabei auf einer einfachen Handhabung, guter Performance, weitreichender Dokumentation und einer einheitlichen API, sodass auch Nicht-Machine Learning-Spezialisten Scikit-learn benutzen können. Ein Vorteil von Scikit-learn ist, dass es Open Source ist und somit sowohl für akademische als auch für kommerzielle Zwecke genutzt werden kann. [10]

Im Projekt sind vor allem die verschiedenen Implementationen von Modellklassen für Klassifikationsprobleme benutzt worden. Zusätzlich ist Scikit-learn zum Skalieren der Daten benutzt worden und zur Evaluierung der Machine Learning-Modelle.

### 8.3.5 Music21

Music21 ist eine Python-Library und bietet eine Vielzahl an Möglichkeiten für Analyse und Transformation von Musik in ‚symbolic' Form. ‚Symbolic' bedeutet, dass Musik nicht als 'normale' Audiodatei (z.B. MP3-Datei), sondern als die einzelnen Noten mit Tonhöhe, Zuordnung zu Musikinstrument etc. gespeichert (z.B. als Datei im MIDI-Format), vorliegt. [11]

Anwendung hat music21 im Zuge der Extraktion der Features gefunden, da man mit music21 auch MIDI-Dateien verarbeiten kann.

### 8.3.6 NumPy

NumPy ist die wichtigste Programmierbibliothek für Arrays in Python, da sie eine leistungsstarke und ausdrucksstarke Syntax für den Zugriff und die Bearbeitung von Daten in Vektoren, Matrizen und höherdimensionalen Arrays mitbringt. Somit spielt NumPy eine wesentliche Rolle bei Forschung und Analyse in unterschiedlichen Bereichen wie Physik, Chemie, Astronomie, Geowissenschaften, Biologie, Psychologie, Materialwissenschaften, Ingenieurwesen, Finanzen und Wirtschaft. [12]

Im Projekt sind NumPy Arrays verwendet worden, mit denen eine große Menge an wissenschaftlichen Berechnungen möglich sind. [12]

## 8.4 Sonstige verwendete Software

### 8.4.1 Div Slomins MIDI Utilities

David G. Slomin ist ein Princeton Alumni, der sich mit Softwareentwicklung und den darstellenden Künsten auseinandersetzt. Eines seiner Projekte umfasst zahlreiche MIDI Utilities, die er zur eigenen Verwendung entwickelt hat, aber zum allgemeinen Gebrauch zur Verfügung stellt.

Besonders relevant für das Projekt sind die Realtime MIDI Utilities. Mit den *lsmidiins* und *lsmidiouts* können MIDI In- und Outputs abgefragt werden. So kann beispielsweise überprüft werden, ob eine aktive Verbindung mit einem MIDI-fähigen Endgerät herrscht.

Am wichtigsten ist allerdings *brainstorm*, das als „Diktiergerät" für MIDI konzipiert wurde. Es zeichnet den MIDI-Stream auf und speichert das Ergebnis, sobald eine Pause über mehrere Sekunden lang andauert. Die Dateinamen werden dabei automatisch generiert und setzen sich aus einem aktuellen Zeitstempel und einem frei wählbaren Prefix zusammen. So können beispielsweise Aufnahmen für unterschiedliche Zwecke auseinandergehalten werden. Der Name „Brainstorm" leitet sich vom Verwendungszweck ab, da das Tool vom Entwickler für Improvisations- und Brainstorming-Sessions verwendet wird. [13]

### 8.4.2 MidiEditor

Der MidiEditor ist eine kostenfreie Software zum Bearbeiten, Aufnehmen und Abspielen von MIDI-Daten. Bereits bestehende MIDI-Dateien können geöffnet werden, um sie abzuspielen oder den Inhalt zu bearbeiten. Neue Files können erstellt werden, sobald ein Benutzer Daten über ein verbundenes MIDI-Gerät einspielt. Auch einzelne Noten oder andere MIDI-Events können händisch kreiert werden, um ein File zu erstellen.

Die Software wurde von Markus Schwenk in C++ entwickelt und läuft sowohl auf Windows als auch auf Linux. Das Projekt wird laufend weiterentwickelt und von freiwilligen Spenden unterstützt – auch User Feedback ist erwünscht.

## 8.5 Verwendete Hardware

### 8.5.1 Klavier

Um mit der fertigen CEPP-Applikation interagieren zu können, benötigt es ein Klavier auf dem versucht werden kann, eine bestimmte Emotion zu spielen. Da dabei der MIDI-Input des Klaviers verarbeitet wird, muss es sich zwingendermaßen um ein elektronisches Klavier mit MIDI-Schnittstelle handeln. Wobei es auch eine Android App gibt, welche ein Klavier mit MIDI-Schnittstelle emuliert, sodass man anstatt des Klaviers einfach sein Handy an den PC anschließen kann. Diese ist auch für Testzwecke verwendet worden, da nicht immer ein richtiges Klavier zur Hand gewesen ist. Zusätzlich ist ein Klavier im Projekt dazu verwendet worden, um eigene Datensätze aufzunehmen.

# 9 Implementierung

## 9.1 Data Exploration

### 9.1.1 Einleitung

Ziel der endgültigen Anwendung ist die Vorhersage bzw. Erkennung von Emotionen basierend auf dem MIDI-Input, der durch den Benutzer erzeugt wird. Bis dahin ist es allerdings ein weiter Weg, und am Anfang stehen die Daten. Aus Daten können viele Informationen gewonnen werden, diese Daten richtig zu nutzen und zu verstehen ist daher essenziell für das gesamte Projekt. Die Data Exploration dient dazu bestehende Daten auszuwerten, zu analysieren und daraus Schlüsse zu ziehen.

### 9.1.2 EMOPIA

EMOPIA ist ein Datenset bestehend aus gelabelten MIDI-Files, speziell zur Verwendung in der Emotion Recognition oder der emotionsbasierten Musikkreation. Insgesamt sind 1057 Audioclips enthalten, zusammengestellt aus 387 verschiedenen Musikstücken. Dabei handelt es sich zum größten Teil um Stücke aus den Soundtracks von Videospielen. [14]

### 9.1.3 Quadranten

Die Daten aus dem EMOPIA Datenset sind bereits gelabelt, d.h. ihnen wurde bereits eine Kategorie zugewiesen. Mithilfe eines eigens konzipierten Classifiers wurden die Daten in vier verschiedene Quadranten unterteilt. Dieser Algorithmus teilt die Daten ein, er klassifiziert sie. Der Classifier ermittelt dabei "Arousal" und „Valence" eines MIDI-Files und ordnet es entsprechend ein. [14]

#### *9.1.3.1 Valence und Arousal*

Valence und Arousal werden im Zusammenhang mit Musik oder Sprache oft verwendet und drücken ähnliche, aber unterschiedliche Wahrnehmungen aus. Die deutsche Übersetzung von Wertigkeit (bzw. Valenz) und Erregung gibt bereits einen Hinweis auf den Unterschied.

Psychologisch gesehen beschreibt „Valence" wie die Befindlichkeit beeinflusst wird. Konkret also, ob sich etwas angenehm oder unangenehm auswirkt bzw. im Falle von Musik anhört.

Das „Arousal" gibt Auskunft über die ausgelöste Erregung, also ob etwas als aufregend oder als beruhigend wahrgenommen wird. [15]

Daraus ergeben sich vier mögliche Kombinationen, die zweidimensional als Quadranten dargestellt werden können. Abb. 3 zeigt ein Koordinatensystem mit zwei Achsen für Valence und Arousal, wobei jedem Quadranten entsprechende Emotionen zugeordnet sind.

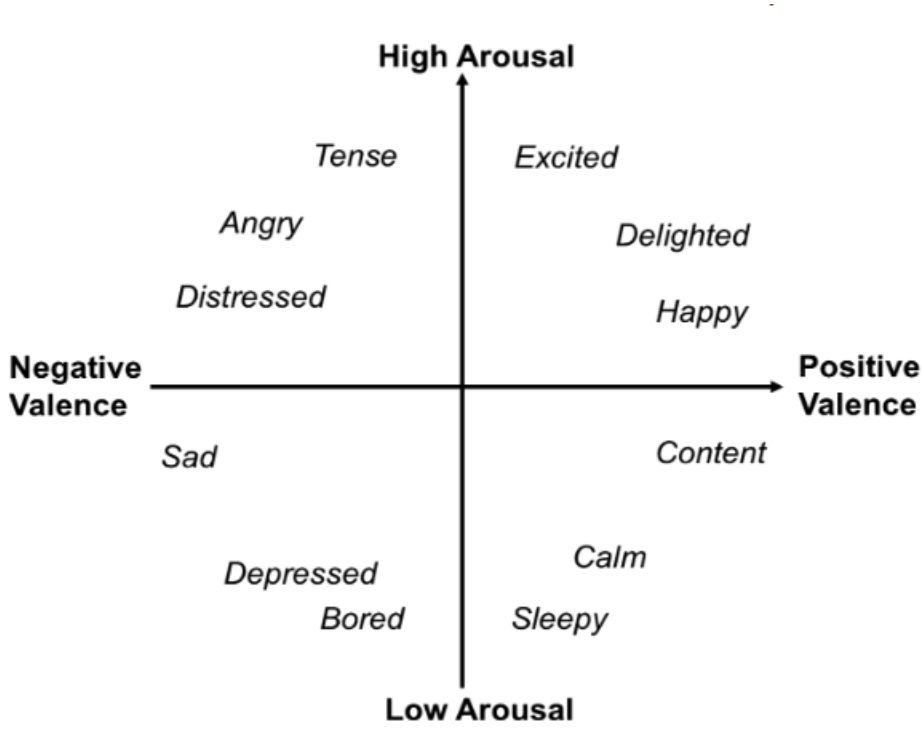



*Abb. 3: Koordinatensystem mit Achsen für „Valence" und „ Arousal"* [16]

Die Kombination von positiver Valence und hohem Arousal entspricht daher Emotionen wie (positiver) Aufregung, Begeisterung oder einfach genereller Freude. Es geht also um positive, energiegeladene Empfindungen. [16]

Eine positive Valence in Kombination mit niedrigem Arousal vermittelt Gefühle wie Zufriedenheit, Ruhe oder Schläfrigkeit. Hierbei handelt es sich um Emotionen mit einem geringen Energielevel, die aber als positiv wahrgenommen werden.

Niedriges Arousal und negative Valence drücken Gefühle aus, die als negativ wahrgenommen werden und über wenig Energie oder Aktivität verfügen. Diese Kombination wird daher als Trauer, Depression oder Langeweile wahrgenommen.

Hohes Arousal und niedrige Valence vermitteln starke, negative Emotionen wie Ärger, (negative) Aufregung und Gereiztheit oder Verzweiflung. [15]

### *9.1.3.2 Emotionen einordnen*

Besonders in der Musik wird bestimmten Stücken oft eine bestimmte Emotion zugeschrieben. Stellt man fest, wo sich das Stück im Valence-Arousal Graph bewegt, kann man darauf basierend die Emotion eingrenzen oder bestimmen.

In diesem Diagramm aus Patrick N. Juslins „Musical Emotions Explained" sind die musikalisch relevantesten Emotionen aufgetragen. Da die Daten des EMOPIA Datensets nicht nach konkreten Emotionen, sondern nach Quadranten gelabelt sind, muss ein Überbegriff für jeden Quadranten gefunden werden. Fear (Angst) wird daher mit Anger (Ärger) zusammengefasst, um die Abgrenzung von Happiness (Freude) und Sadness (Trauer) zu verdeutlichen. So können die EMOPIA Datensätze konkreten Emotionen zugeordnet werden.

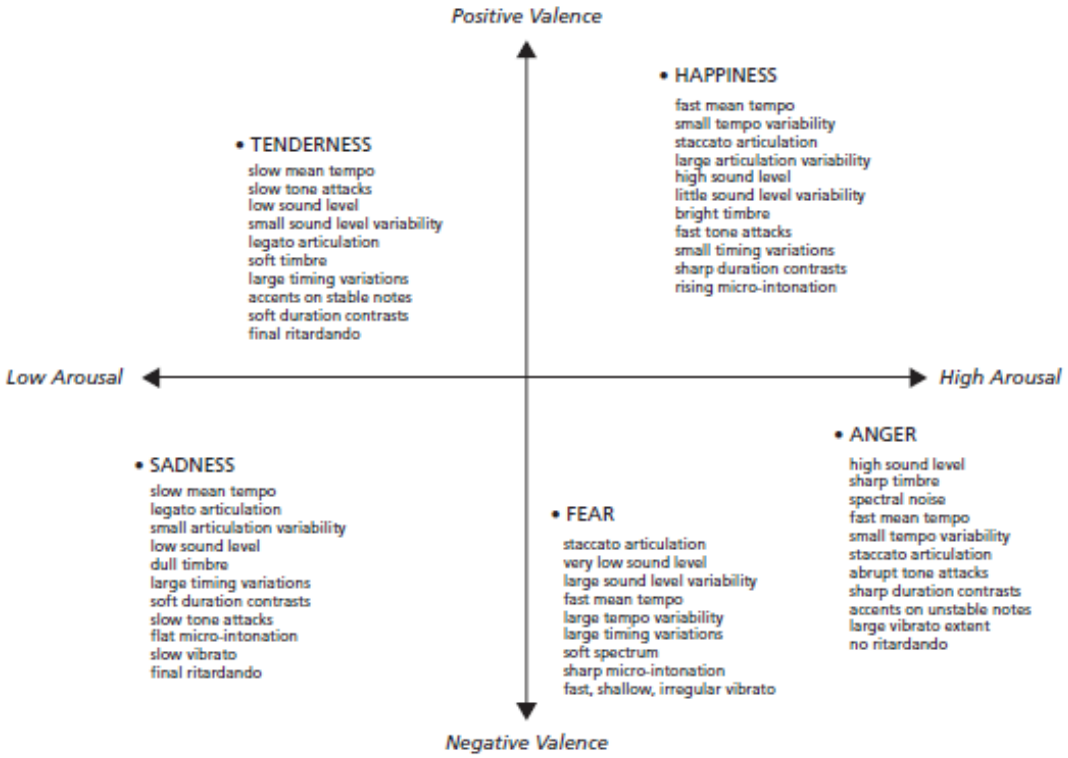



**Figure 7.3** Two-dimensional emotion space in music.
Reproduced from Juslin, P. N. (2010). *Expression And Communication of Emotion in Music Performance.* In P. N. Juslin & J. A. Sloboda (Eds.), Handbook of music and emotion: Theory and research. By permission of Oxford University Press.

*Abb. 4: Emotionen im zweidimensionalen Raum*

### 9.1.4 Features

Da Musikstücke bzw. MIDI-Aufnahmen üblicherweise nicht mit einer Beschreibung ihrer Valence- bzw. Arousal-Werte auftreten, müssen diese Eigenschaften bestimmt werden, um eine Emotion zuordnen zu können.

Die menschliche Wahrnehmung funktioniert anders als die Berechnungen einer Maschine: Während der Mensch sich auf bestimmte Muster verlässt und dadurch unbewusst eine bestimmte Emotion feststellen kann, muss ein Computer mit messbaren Werten arbeiten.

Diese Werte - wie zum Beispiel die Dichte von Noten über das Stück verteilt oder die Spanne zwischen den höchsten und tiefsten Tönen - werden Features genannt und können mit verschiedenen Tools aus MIDI-Aufnahmen exportiert werden. Wie diese Features zum anschließenden Erkennen von Emotionen eingesetzt werden, kann sich je nach Ansatz unterscheiden.

#### *9.1.4.1 Handcrafted Rules*

Zum einen gibt es die Bestimmung basierend auf „Handcrafted Rules" also handgefertigten Regeln, nach denen eine Entscheidung getroffen wird. Diese Entscheidungen basieren dabei grundsätzlich auf musiktheoretischer Grundlage.

Zu diesem Ansatz würde mit höchster Wahrscheinlichkeit ein Mensch, der basierend auf den Features einer MIDI-Datei eine Emotion bestimmen soll, greifen. Auch wenn nicht bewusst Features bestimmt werden, wertet ein Mensch in der Regel Eigenschaften wie Tempo, Tonart oder Betonung aus und trifft darauf basierend seine Entscheidung.

Dazu wird allerdings eine gewisse Erfahrung bzw. ein Level an Grundwissen vorausgesetzt. Es gibt zahlreiche Maßstäbe, an denen man eine Emotion oder Stimmung festmachen könnte.

Dreiklänge zählen zu den aussagekräftigsten Merkmalen. Dur-Dreiklänge haben grundsätzlich einen harmonischen Klang – ihnen wird daher eine allgemein positive Stimmung bzw. eine positive Valence zugeschrieben. Bei Moll-Dreiklängen, die grundsätzlich nicht harmonisch klingen ist es umgekehrt. Sie werden mit einer negativen Stimmung bzw. einer negativen Valence verbunden.

Auch Intervalle können Auskunft über die Stimmungslage geben. Beim ikonischen Thema aus „Jaws" wird beispielsweise eine kleine Sekund (zwei Töne, die nur einen Halbton auseinanderliegen) verwendet, um Spannung aufzubauen.

Eine große Terz (fünf Halbtöne Abstand) kennt man dagegen aus Stücken wie dem „Donauwalzer", wo eine heitere, positive Stimmung vermittelt wird. Ein besonderes Intervall stellt der „Tritonus des Teufels" dar, der in früheren Zeiten aufgrund seines unheimlichen Klanges gemieden wurde. Es handelt sich dabei allerdings lediglich um eine übermäßige Quart, also eine Konstellation mit sieben Halbtönen Abstand.

Auch die Dynamik, die Artikulation oder das Tempo beeinflussen stark, wie ein Stück wahrgenommen wird. Vortragsbezeichnungen, die in der Regel als Ergänzung auf dem Notenblatt zu finden sind, gehen dabei noch einen Schritt weiter und geben eine konkrete Stimmung vor. Diese italienischen Fachbegriffe umfassen dabei von „agitato" (aufgeregt, unruhig) bis zu „scherzando" (scherzhaft, scherzend) eine ganze Bandbreite an Stimmungen.

Während solche „Regeln" oder Richtlinien sehr gut auf Audioaufnahmen angewandt werden können, ist es schwierig diese Eigenschaften in den abstrakten MIDI-Features wiederzufinden. Für eine Maschine, die solche Eigenschaften nicht wie ein Mensch interpretieren kann, wird daher ein anderer Ansatz benötigt.

#### *9.1.4.2 Datengetriebener Ansatz*

Im Projekt kommen keine direkten handgefertigten Regeln zum Einsatz. Stattdessen werden die Features als numerische Werte exportiert und von einem Machine Learning-Modell weiterverarbeitet. Vorarbeit in Form von Korrelationsanalysen oder der Auswahl besonders relevanter Features kann dieses Ergebnis optimieren.

### 9.1.5 Features extrahieren

Features können mithilfe von geeigneten Tools aus sogenannten „Symbolic Files" gewonnen werden. Reguläre Audiodateien speichern eine digitale Annäherung der aufgenommenen Soundsignale, während Symbolic Files Informationen über die Aufnahme speichern. Dadurch ergeben sich Daten wie die Anzahl der gespielten Noten und deren Tonhöhe, welches Instrument welche Noten spielt und Start- und Endzeiten jeder Note. [17]

Aufgrund dieser Richtlinien gelten auch Notenblätter, die von einem Komponisten verfasst und von einem Musiker gelesen werden, als Symbolic Files. Alle Daten,

die benötigt werden, um das Musikstück zu reproduzieren, sind in dieser abstrakten Beschreibung enthalten und können vom Musiker interpretiert und umgesetzt werden.

Digitale Formate für Symbolic Files umfassen MIDI, Humdrum oder MusicXML.

Um Features aus solchen Dateien zu gewinnen, gibt es mehrere Tools. Zu den bekanntesten zählen dabei das Java-basierte jSymbolic[5] und die Python Library music21[6]. Beide stellen einen Excel- bzw. CSV-Export zur Verfügung, um die Features menschenlesbar anzuzeigen.

### 9.1.6 jSymbolic

Die Software Suite jMIR wird zur Gewinnung von Informationen aus der Musik, also Music Information Retrieval (MIR), eingesetzt. Sämtliche Software ist in Java implementiert, was gemeinsam mit dem Einsatzgebiet den Namen jMIR ergibt.

jSymbolic wiederum ist eine jMIR-Komponente, die dazu dient, Features aus symbolisch gespeicherten Daten zu extrahieren. Speziell geht es dabei um MIDI-Files.

Das Tool ist so umgesetzt, dass zur Verwendung keine speziellen Kenntnisse benötigt werden. Neben einer Anwendung für die Command Line gibt es auch eine GUI, die einfacher in der Anwendung ist.

Wie alle Komponenten von jMIR ist jSymbolic Open Source und frei verfügbar. [17]

### 9.1.7 Music21

Eine weitere Möglichkeit bietet das multifunktionale Open Source Toolkit Music21, das über Funktionen zur Extraktion von Features verfügt. Es bietet zahlreiche Funktionen zur Analyse, Transformation oder Suche von musikbezogenen Daten.

Neben einigen Standard-Extraction Tools, die von anderen Toolkits (z.B. von jSymbolic) bereitgestellt werden, verfügt Music21 auch über eigene, neue Tools und ermöglicht dem Anwender sogar eigene Extraktionsmethoden zu schreiben. [18]

[5] https://jmir.sourceforge.net/jSymbolic.html
[6] https://web.mit.edu/music21/

### 9.1.8 Extrahierte Features

Die Features, die durch die Extraktion ausgelesen werden können, lassen sich in verschiedene, grobe Kategorien einteilen. Bei jSymbolic sieht diese Einteilung folgendermaßen aus: [17]

- Instrumentation
- Textur
- Rhythmus
- Dynamik
- Statistiken zur Tonlage
- Melodie
- Akkorde

### 9.1.9 Auswahl von relevanten Features

Nicht alle Features sind gleich relevant für die Ermittlung von Valence und Arousal. Manche Features sagen annähernd nichts aus, während mit anderen sehr schnell bestimmt werden kann, in welchem Quadranten oder zumindest in welcher Hälfte des Graphs sich die analysierte Aufnahme einordnen lässt.

Eine gute Möglichkeit, um darzustellen, wie gut ein Feature zur Unterscheidung der Quadranten geeignet ist, sind Pair-Plots mit denen die Datensätze abgebildet werden können. Ein Pair-Plot stellt eine Matrix an Scatter-Plots dar, in dem alle Beziehungen zwischen den verschiedenen enthaltenen Variablen bzw. Features abgebildet werden können. Zur grafischen Darstellung eignen sich dabei allerdings nur die quantitativen Features, also jene, die einen numerischen Wert aufweisen. Kategorische Variablen, die beispielsweise als Wert entweder 0 oder 1 haben, liefern dagegen keine aussagekräftigen, grafischen Darstellungen. [19]

Gängige Tools zur Erstellung und Berechnung von Pair-Plots in Python sind beispielsweise Python Libraries Seaborn[7] und Matplotlib[8]. Bei beiden handelt es sich um Bibliotheken zur Visualisierung von Daten, besonders in Form von Plots und Grafen.

Im Projekt werden die Plots basierend auf einer Liste an ausgewählten Features mithilfe von Matplotlib generiert.

```
sns.pairplot(df, vars=features, hue='Label', palette=colors)
matplotlib.pyplot.show()
```

*Listing 1: Generieren und Anzeigen eines Pair Plots mit Matplotlib*

[7] https://seaborn.pydata.org/index.html
[8] https://matplotlib.org/

Verschiedene Features bilden die Unterschiede zwischen Valence und Arousal verschieden gut ab. Ein Pair-Plot, in dem eine gute Trennung zwischen zwei Teilmengen ersichtlich ist, ist hier in Abb. 5 abgebildet.

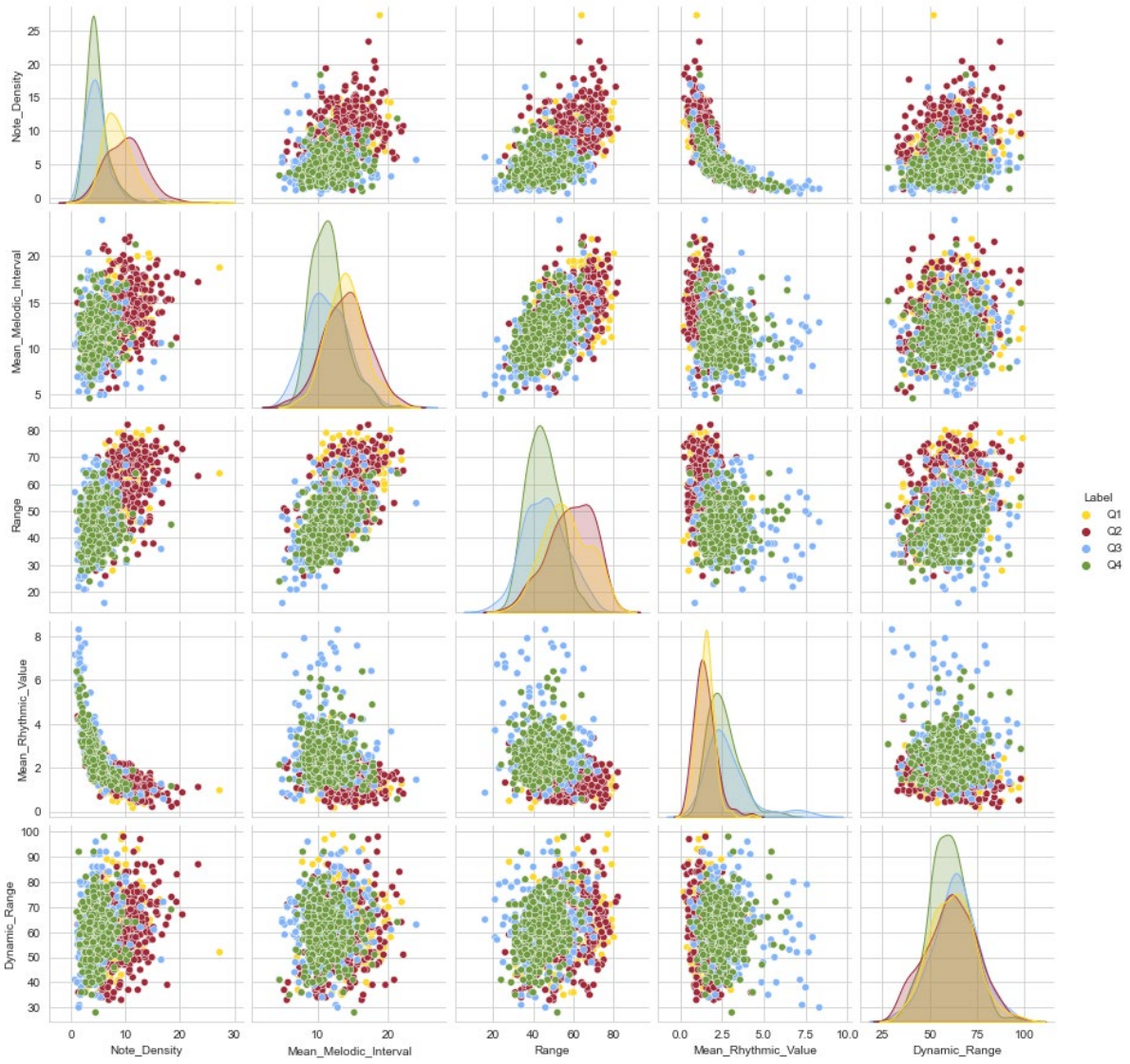



*Abb. 5: Pair-Plot mit fünf Features*

Wie Abb. 5 zeigt, kann im Pair-Plot klar abgelesen werden, dass Datensätze aus verschiedenen Quadranten unterschiedlich eingeordnet werden. Hierbei ist besonders interessant, dass sich Datensätze aus Quadrant 1 und 2 sehr stark überlappen, sich aber gut von den ebenfalls sehr deckungsgleichen Mengen aus Quadrant 3 und 4 abgrenzen.

Bei diesem Pair-Plot wurden bewusst Features gewählt, die sich gut zur Unterscheidung anhand des Arousals eigenen. In diesem Fall handelt es sich um folgende Features:

- Note Density
- Mean Melodic Interval
- Range
- Mean Rhythmic Value
- Dynamic Range

Die Dichte an gespielten Noten sagt viel darüber aus, wie „geladen" das Stück ist. Ein schnelles oder hektisches Stück weist eine größere Dichte an Anschlägen auf als ein sehr ruhiges, träges Stück. Das Mean Melodic Interval gibt Auskunft über das durchschnittliche Intervall in Halbtönen. Je nachdem um welches Intervall es sich dabei handelt, kann man auf eine grundlegende Stimmung schließen. Die Range gibt die Spanne zwischen dem höchsten und tiefsten vorkommenden Ton in Halbtönen an.

Der Mean Rhythmic Value gibt den durchschnittlichen Rhythmus in Viertelnoten an. Ein Wert von 0,5 würde in diesem Zusammenhang bedeuten, dass der Rhythmus der Musik ungefähr mit einer Achtelnote (halbe Viertelnote) zusammenstimmt. Daraus lässt sich wieder auf das Tempo schließen, was aktive, fröhliche oder aufgeregte Stücke von eher langsamen, ruhigen oder traurigen Stücken unterscheidet.

Die Dynamic Range gibt die Spanne in der Dynamik, also der Lautstärke über das Stück an. Hierbei wird der leiseste Ton vom lautesten Ton abgezogen und so die Differenz gebildet. Die Lautstärke, bzw. die daraus folgende Anschlagstärke, ist bei aufgeregten, fröhlichen Stücken meist um ein Vielfaches höher als bei sanften, traurigen Stücken und ist so ebenfalls ein guter Messwert zur Bestimmung des Arousals. [20]

In Abb. 6 ist ein Pair-Plot abgebildet, in dem Features verwendet wurden, die besser zur Bestimmung der Valence geeignet sind. Bei einigen dieser Features handelt es sich allerdings um kategorische Features, was eine Darstellung im Plot nicht aussagekräftig macht.

Bei den verwendeten Features handelt es sich um folgende:

- Dynamic Range
- Minor Major Melodic Third Ratio
- Minor Major Triad Ratio
- Similar Motion

Die Minor Major Melodic Third Ratio gibt das Verhältnis zwischen kleinen und großen Terzen im Sample an. Ein hoher Anteil an großen Terzen (niedriger Feature-Wert) wäre dabei als Hinweis auf eine positive Valence zu werten, während ein hoher Anteil an kleinen Terzen (hoher Feature-Wert) auf eine negative Valence hindeutet.

Bei der Minor Major Triad Ratio funktioniert es nach dem gleichen Prinzip, nur geht es hierbei um Dur- und Moll-Dreiklänge. Ein hoher Anteil an Dur-Dreilängen resultiert in einem niedrigen Feature-Wert und deutet auf eine positive Valence hin. Ein hoher Anteil an Moll-Dreiklängen produziert bei der Berechnung einen hohen Feature-Wert und stellt ein Anzeichen für eine negative Valence dar. Die Dreiklänge werden dabei nach ihrer Länge gewichtet, ein Dreiklang, der eine ganze Note andauert, hat also den gleichen Wert wie vier Dreiklänge, die über eine Viertelnote hinweg gehalten werden.

Diese Ratio-Features werden berechnet, indem der überwiegende Anteil durch den Rest geteilt wird. Sind beispielsweise in einem Sample überwiegend Dur-Dreiklänge enthalten, so wird die Anzahl bzw. die Wertigkeit dieser durch die Wertigkeit der enthaltenen Moll-Dreiklänge geteilt. Ein Mangel, der bei dieser Berechnungsart sofort ins Auge sticht, ist die verwendete Division. Es ist mathematisch nicht möglich durch 0 zu teilen. Ist daher eine dieser beiden Komponenten im Sample nicht enthalten, wird dem Feature standardmäßig der Wert 0 zugewiesen. Die vielversprechendsten Features zur Bestimmung der Valence werden somit gravierend entwertet.

Dieses Problem schlägt sich in der Darstellung des Modelle nieder, da es nahezu unmöglich ist, Mengen basierend auf diesen Features auseinanderzuhalten. In Abb. 6 ist der Pair-Plot zu den Valence-Features dargestellt. Es sind keine Punktemengen klar erkennbar, bis auf einige Ausreißer handelt es sich stets um eine nahezu deckungsgleiche Ansammlung an Datenpunkten.

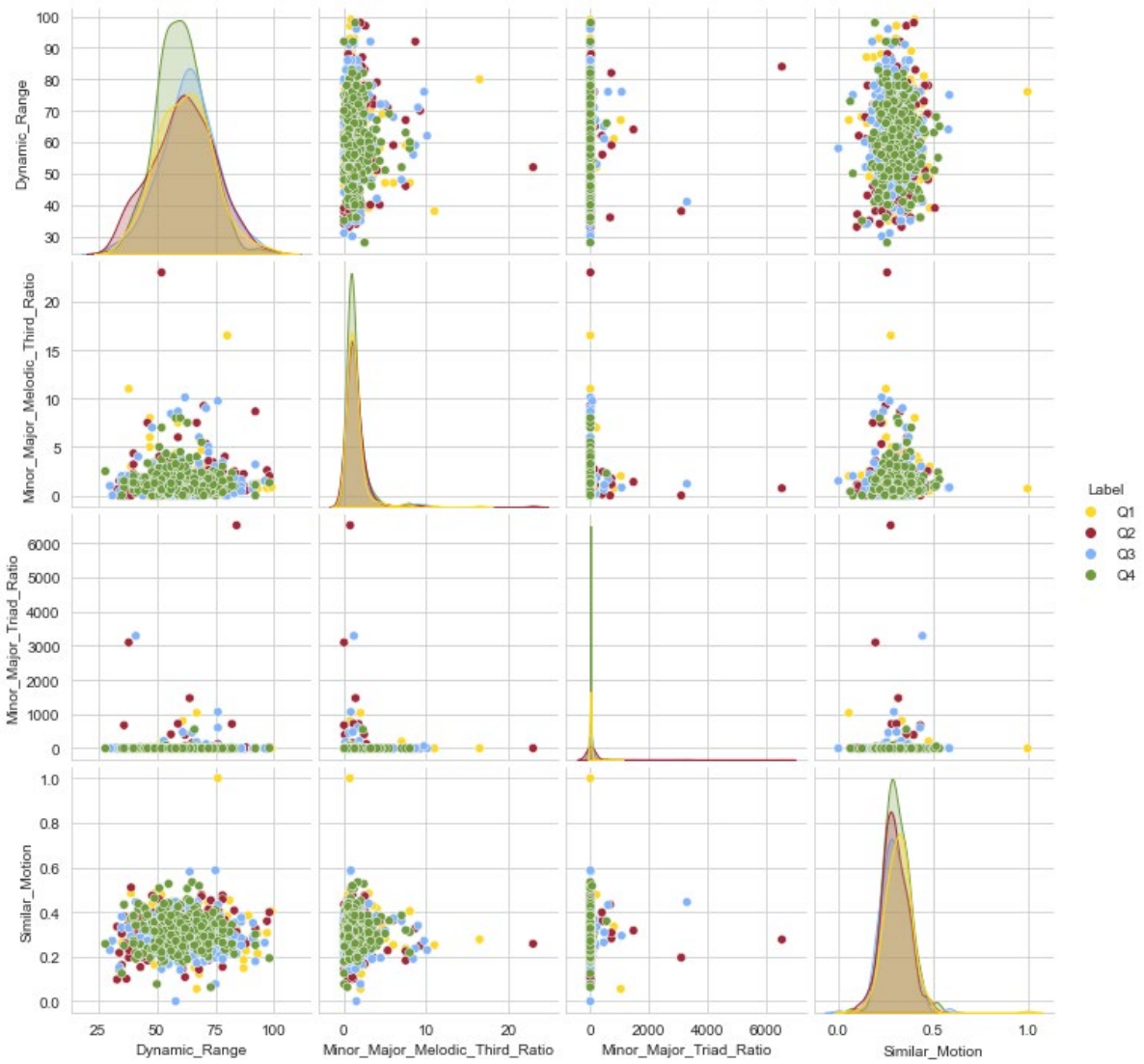



*Abb. 6: Pair-Plot mit vier Features*

Diese Funktionsweise bzw. Berechnungsmethode der jSymbolic Features trägt möglicherweise zu der anhaltenden Problematik bei, die Valence von MIDI-Files korrekt zu bewerten. Das Zurückgreifen auf den Standardwert 0 – ungeachtet dessen, ob ein eindeutiges Übergewicht vorliegt – schadet dem Feature. In den aussagekräftigsten Fällen - nämlich bei völliger Vorherrschaft von Major oder Minor in einer der beiden Kategorien - versagt die Berechnung, da der Standardwert zugewiesen wird. Somit ist es in solchen Fällen sehr schwierig, gute Anhaltspunkte für die Valence des Samples zu finden.

#### *9.1.9.1 Korrelationsanalyse*

Bestimmte Features korrelieren so stark, dass sie redundant sind, da eines der beiden Features keinen Mehrwert bringt. Mit einer Korrelationsanalyse kann festgestellt werden, wie groß der Zusammenhang ist und ob beide Features relevant sind.

Mit Hilfe von Excel kann eine Korrelationsanalyse zu ausgewählten Features durchgeführt und dargestellt werden. Je mehr Features dabei analysiert werden, desto umfangreicher gestaltet sich die entstehende Tabelle.

In diesem Beispiel werden das Mean Melodic Intervall, die Number of Common Pitches bzw. of Pitch Classes, der Amount of Staccato und die Range bewertet.

| | *Mean_Melodic_Interval* | *Number_of_Common_Pitches* | *Number_of_Common_Pitch_Classes* | *Amount_of_Staccato* | *Range* |
|---|---|---|---|---|---|
| Mean_Melodic_Interval | 1 | | | | |
| Number_of_Common_Pitches | -0.448818313 | 1 | | | |
| Number_of_Common_Pitch_Classes | -0.047572317 | 0.347272492 | 1 | | |
| Amount_of_Staccato | 0.106932417 | 0.014834507 | -0.001596373 | 1 | |
| Range | 0.60122918 | -0.481948616 | -0.060766285 | 0.211706939 | 1 |

*Abb. 7: Korrelationsanalyse mit fünf Features*

Eine farbcodierte Darstellung zeigt besser an, ob sich zwischen zwei Features ein Zusammenhang abzeichnet.

| | *Mean_Melodic_Interval* | *Number_of_Common_Pitches* | *Number_of_Common_Pitch_Classes* | *Amount_of_Staccato* | *Range* |
|---|---|---|---|---|---|
| Mean_Melodic_Interval | 1 | | | | |
| Number_of_Common_Pitches | -0.448818313 | 1 | | | |
| Number_of_Common_Pitch_Classes | -0.047572317 | 0.347272492 | 1 | | |
| Amount_of_Staccato | 0.106932417 | 0.014834507 | -0.001596373 | 1 | |
| Range | 0.60122918 | -0.481948616 | -0.060766285 | 0.211706939 | 1 |

*Abb. 8: Korrelationsanalyse mit fünf Features; farblich markiert*

Das Mean Melodic Interval zeigt das durchschnittliche Intervall in Halbtönen an. Dieses Feature kann also unter anderem als ein Richtwert für die tonale Spannweite des Stückes angesehen werden.

Die Range bezeichnet genau diese Spannweite – den Abstand zwischen dem höchsten und dem tiefsten Ton des Stückes. Es ist naheliegend, dass zwischen diesen beiden Features ein Zusammenhang herrscht. Und tatsächlich ist die Korrelation zwischen diesen Features am größten.

Entgegen mancher Erwartung lässt sich allerdings kein großartiger Zusammenhang zwischen der Anzahl an unterschiedlichen Tönen, der Anzahl unterschiedlicher Tonklassen oder dem Anteil an Staccato im Stück bilden.

Da das Mean Melodic Interval und die Range zwar in merkbarem Ausmaß korrelieren, jedoch beide für unterschiedliche Aspekte der Klassifizierung relevant sind, ist eine Streichung eines Features nicht sinnvoll.

Bei einer weiteren Analyse werden die Note Density, der Amount of Arpeggiation, die Number of Pitches und die Number of Common Pitches miteinander auf Zusammenhänge überprüft.

| | *Note_Density* | *Amount_of_Arpeggiation* | *Number_of_Pitches* | *Number_of_Common_Pitches* |
|---|---|---|---|---|
| Note_Density | 1 | | | |
| Amount_of_Arpeggiation | -0.04976894 | 1 | | |
| Number_of_Pitches | 0.58604646 | -0.197621278 | 1 | |
| Number_of_Common_Pitches | -0.25501572 | 0.261452919 | -0.627501452 | 1 |

*Abb. 9: Korrelationsanalyse mit vier Features*

In der kolorierten Version wird ersichtlich, dass zwischen der Note Density und der Number of Pitches eine gewisse Korrelation herrscht. Dass die Anzahl der im Stück vorkommenden Noten und die Dichte der Anschläge zusammenhängen ist schlüssig, da ein Wert den anderen zwangsläufig erhöht. In diesem Fall wäre eine Streichung zur Reduktion der Features grundsätzlich denkbar, allerdings durch die verhältnismäßig geringe Korrelation nicht gerechtfertigt.

| | *Note_Density* | *Amount_of_Arpeggiation* | *Number_of_Pitches* | *Number_of_Common_Pitches* |
|---|---|---|---|---|
| Note_Density | 1 | | | |
| Amount_of_Arpeggiation | -0.04976894 | 1 | | |
| Number_of_Pitches | 0.58604646 | -0.197621278 | 1 | |
| Number_of_Common_Pitches | -0.25501572 | 0.261452919 | -0.627501452 | 1 |

*Abb. 10: Korrelationsanalyse mit vier Features; farblich markiert*

Die Korrelationsanalyse kann daher im Prozess der Data Exploration angewandt werden, um redundante Features zu identifizieren und Zusammenhänge zwischen den Features besser zu verstehen.

### Durchführung

Um eine Korrelationsanalyse mit Excel durchzuführen, bedarf es einer Menge an normalisierten Daten. Die Datensätze können spaltenweise (nach Feature) aus einem jSymbolic Export entnommen und weiterverarbeitet werden.

Um die Daten zu standardisieren, müssen sowohl der Mittelwert (MITTELWERT) als auch die Standardabweichung (STABW.N) jeder Spalte ermittelt werden. Aus diesen Werten kann mithilfe der Funktion STANDARDISIERUNG ein normalisierter Wert errechnet werden.

Dieser normalisierte Wert wird zum Vergleich der verschiedenen Features benötigt, da diese sehr unterschiedliche Wertespannen aufweisen. Es wäre daher nicht möglich, einen Anzahlwert, der sich im dreistelligen Bereich bewegt mit einem Feature mit einem Wertebereich zwischen 1 und 0 zu vergleichen.

Die neu berechneten Werte können daraufhin per Hand systematisch verglichen werden, indem die Funktion KORELL angewandt wird. Einfacher geht dieser Vorgang mit dem Analysis ToolPak. Das Excel Add-In stellt Tools zur Durchführung komplexer statistischer Analysen bereit, unter anderem der Korrelationsanalyse. Bei der Anwendung kann ein Wertebereich zur Analyse sowie ein Ergebnisbereich ausgewählt werden, am Ende der Berechnung steht eine vollständige Tabelle.

Die Tabelle kann anschließend wie in den Abb. 8 und 10 erkennbar farblich gekennzeichnet und dementsprechend interpretiert werden.

#### *9.1.9.2 Krumhansl-Schmuckler*

Ein weiteres relevantes Feature bzw. Merkmal zur Klassifizierung der Datensätze stellt die Tonart dar. Wie bei den Dreiklängen unterscheiden sich die Tonarten in Moll- und Dur-Tonarten und weisen stets einen charakteristischen Klang auf. Wieder gilt, Moll klingt melancholischer als das heitere Gegenstück in Dur.

Es handelt sich dabei allerdings nicht um eine feste Regel, es gibt zahlreiche bekannte Musikstücke, die diesem Schema absolut nicht folgen. Dieser Mythos wird beispielsweise in Episode 65 von „The Daily Doug“[9] anhand bekannter Songs erklärt und widerlegt. Neben der Tonart spielen unzählige weitere Eigenschaften eines Musikstücks eine Rolle, was auch die hohe Anzahl an jSymbolic Features erklärt, die allesamt einen gewissen Einfluss auf die Stimmung und letztendliche Wahrnehmung einer Aufnahme haben.

Nichtsdestotrotz sind gewisse Features aussagekräftiger als andere und die Tonart ist ein guter Anhaltspunkt zur Bestimmung der Valence. Da jSymbolic allerdings nicht in der Lage ist, dieses Feature verlässlich zu extrahieren, muss dafür eine alternative Lösung gefunden werden.

Music21 besitzt eine Funktion, die den Krumhansl-Schmuckler Algorithmus anwendet, um die Tonart einer MIDI-Aufnahme zu bestimmen. Die Unterscheidung nach Dur und Moll ist nach Erhalt dieser Informationen einfach, da Dur-Tonarten

[9] https://www.youtube.com/watch?v=LgPiBx22gpc

mit einem Kreuz ♯ (sharp) und Moll-Tonarten mit einem Be ♭ (flat) gekennzeichnet werden. Die Funktion ermittelt beispielsweise „C#" (ausgesprochen wie die Programmiersprache) als die Tonart. In deutscher Schreibweise handelt es sich dabei einfach um C-Dur. Anhand des verwendeten Vorzeichens kann also festgestellt werden, ob es sich um Dur oder Moll handelt.

#### Algorithmus

Der Algorithmus an sich verwendet einen zwölfdimensionalen Vektor, der die jeweilige Länge aller zwölf Töne der chromatischen Tonleiter abbildet. Als Output wird ein weiterer Vektor berechnet, der die Wahrscheinlichkeit für jede mögliche Tonart anzeigt. Der Algorithmus unterscheidet nicht zwischen Tönen die enharmonisch verwechselt werden können. Die Tonart wird anhand der weiteren, erkannten Töne festgestellt. Sobald eine Tendenz zu einer Tonart vorliegt, wird überprüft welches Vorzeichen präsenter ist. Darauf basierend wird entschieden, um welche von zwei enharmonisch äquivalenten Tonarten es sich handelt. Kommen überwiegend Kreuze vor, wird beispielsweise die Dur-Variante bevorzugt. [21]

#### Anwendung

Obwohl mithilfe der Music21 Analyse-Funktionen Ergebnisse erzielt werden konnten, sind die Funktionen für eine finale Anwendung nicht einsatztauglich, da die Berechnungszeit auf einem handelsüblichen Endgerät eine Verarbeitung in Echtzeit unmöglich macht.

### 9.1.10 Das Valence/Arousal Problem

Wenn nun aussagekräftige Features gefunden werden konnten, steht bereits die nächste Herausforderung auf dem Weg zur Vorhersage bereit. Ein bekanntes Problem im Feld des Music Information Retrieval bzw. besonders im Gebiet der „Speech and Emotion Research" ist das „Valence-Arousal-Problem". Dabei unterscheidet ein Modell zwar sehr gut nach dem Arousal, aber kann nur schlecht nach der Valence unterscheiden. [22]

Daher kann mit hoher Genauigkeit festgestellt werden, ob ein Sample „Happy" oder „Angry" im Gegensatz zu „Sad" oder „Tender" ist. Die jeweilige Unterscheidung von „Happy" und „Angry" sowie zwischen „Sad" und „Tender" gestaltet sich dadurch allerdings sehr schwierig.

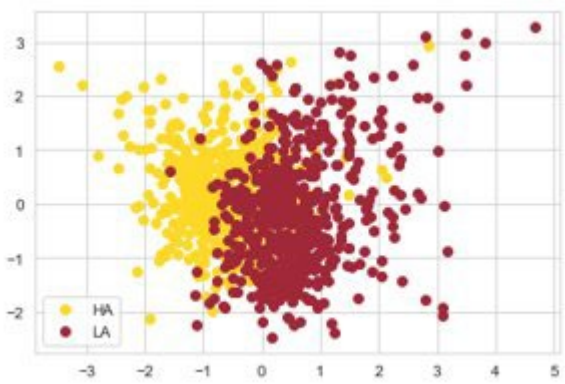



*Abb. 11: High/Low Arousal*

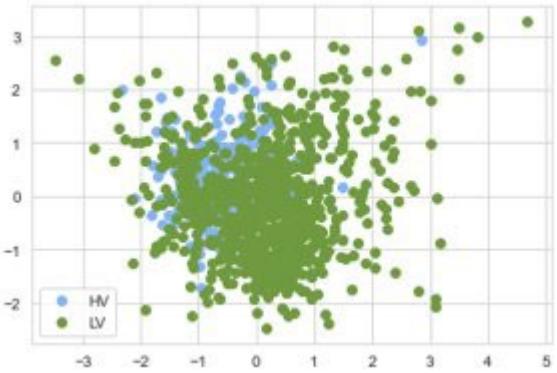



*Abb. 12: High/Low Valence*

In Abb. 11 sind die beiden Mengen klar voneinander unterscheidbar. Die gelben Punkte stellen Datensätze mit hohem Arousal (HA) dar, die roten stehen für Datensätze mit niedrigem Arousal (LA). Somit kann mit hoher Sicherheit eine Aussage getroffen werden, ob ein Datensatz über hohes oder niedriges Arousal verfügt. Darauf basierend kann mit hoher Treffsicherheit entschieden werden, ob ein Datensatz nun „Happy" und „Angry", oder eher „Tender" und „Sad" klingt.

In Abb. 12 dagegen, sind die beiden Punktewolken beinahe vollständig deckungsgleich. Zwischen den blauen Punkten mit hoher (HV) und den grünen mit niedriger Valence (LV) ist eine klare Unterscheidung nicht möglich. Es kann daher keine sichere Aussage getroffen werden, ob ein Datensatz über hohe oder niedrige Valence verfügt, wodurch auch nicht wirklich eingegrenzt werden kann, welche Emotion den Datensatz am besten beschreibt.

| | | *Actual Emotion* | | | |
|---|---|---|---|---|---|
| | | **Q1** | **Q2** | **Q3** | **Q4** |
| *Predicted* | **Q1** | 0.65 | 0.16 | 0.04 | 0.13 |
| | **Q2** | 0.23 | 0.57 | 0.13 | 0.05 |
| | **Q3** | 0.10 | 0.01 | 0.55 | 0.32 |
| | **Q4** | 0.07 | 0.02 | 0.16 | 0.74 |

*Tabelle 1: Confusion Matrix farblich markiert*

Die Tabelle 1 stellt eine normalisierte Confusion Matrix dar, die bei der Analyse verschiedener Machine Learning-Modelle zum Einsatz kommt. In dieser Tabelle wird die Confusion Matrix einer SVM abgebildet, wobei das Valence-Arousal Problem ebenfalls sehr gut beobachtet werden kann.

Der Anteil an richtig klassifizierten Datensätzen ist grün eingefärbt, der jeweils größte Anteil an falsch klassifizierten Datensätzen rot. Sofort sticht ins Auge, dass die Verwechslungen anhand einer Achse des Koordinatensystems besonders gravierend sind.

Q1 und Q2 (Happy und Angry) weisen beide ein hohes Arousal auf, Q3 und Q4 (Sad und Tender) beide ein niedriges. Zwischen diesen beiden „Lagern" kommt es nur selten zu Verwechslungen. Innerhalb dieser Paare ist die Unterscheidung allerdings sehr viel schwieriger. Das System kann zwischen der hohen Valence von Q1 und der niedrigen von Q2 wesentlich schlechter unterscheiden und ordnet somit mehr Datensätze dem falschen Quadranten zu.

Diese zentrale Schwierigkeit ist, wie bereits erwähnt, besonders in der Forschung zu Emotionen in der Sprache ein etabliertes Problem. Bezogen auf die menschliche Sprache liegt das Problem in der Messbarkeit der Valence. Die Absicht und damit gewissermaßen die Emotion wird oft mit sprachlichen Mitteln ausgedrückt, die für eine Maschine möglicherweise nicht als solche erkennbar sind. [22]

Oftmals stellt es schon für Menschen eine Schwierigkeit dar, Stilmittel, Redewendungen oder Sarkasmus richtig zu interpretieren. Wenn der Inhalt des Gesprochenen durch eine solche Linse bewertet werden muss, stoßen maschinelle Lösungen zwangsläufig an ihre Grenzen. Dieses Phänomen zeigt sich auch bei Musikstücken. Es ist wesentlich schwieriger die Valence zu bewerten, da diese sich auf weniger mathematisch messbare Parameter bezieht als das Arousal.

## 9.2 Aufzeichnung von eigenen Daten

### 9.2.1 Zahlen und Fakten zum Datenset

Insgesamt wurden über den Zeitraum mehrerer Tage und mit der Hilfe von 15 freiwilligen Testpersonen 68 Samples aufgenommen. Diese MIDI-Files sind gelabelt und gleichmäßig auf die Quadranten verteilt. Es sind daher für jeden Quadranten 17 Datensätze aufgenommen worden.

Das Datenset ist nicht umfangreich genug, um ein Modell eigenständig zu trainieren, da es zu wenige Datensätze enthält. Die Daten können allerdings in Kombination mit dem EMOPIA Datenset verwendet werden, da einheitliche Labels vorliegen. Die aufgezeichneten Daten entsprechen dabei einem Anteil von 6,6% des EMOPIA Datensets.

Zudem können aus den eigenen Daten wertvolle Schlüsse über relevante Features gezogen werden. Da die spontanen Aufnahmen ungeschulter Benutzer sehr stark von den professionellen Aufnahmen aus dem EMOPIA Datenset abweichen, wirft es allerdings auch Fragen zur Umsetzbarkeit des Projekts auf. Es stellt sich die Frage, ob die Features in solchen Aufnahmen stark genug ausgeprägt sind, um eine treffsichere Prediction zu ermöglichen.

### 9.2.2 Ablauf

Um einen geordneten Ablauf zu garantieren, wurde vor Beginn des Aufnahmeprozesses eine Informationsveranstaltung organisiert. Im Rahmen einer monatlichen Institutsbesprechung wurde das Projekt sämtlichen Kollegen vorgestellt und um Mithilfe gebeten.

Dieser Start stellte sich als eine gute Entscheidung heraus, da Interesse am Projekt geweckt wurde und somit einige Freiwillige zum Aufnehmen von MIDI-Files rekrutiert werden konnten.

Das Recording selbst folgt einer strukturierten Abfolge:

1. Erneute kurze Erklärung des Projekts an sich
2. Erklärung der vier Quadranten in Bezug auf die benötigten Aufnahmen
3. Aufnahme durch den freiwilligen Teilnehmer
4. Labelling der aufgenommen MIDI-Files

Es gibt zwei verschiedene Ansätze, um eine Aufnahme für einen bestimmten Quadranten zu erzielen:

Zum einen kann ein „gezielter" Ansatz gewählt werden, bei dem die Testperson einen Quadranten auswählt oder zugeteilt bekommt und daraufhin versucht eine Aufnahme zu produzieren, die der entsprechenden Emotion so gut wie möglich gerecht wird. Diese Vorgehensweise ist populär unter Testpersonen, die bereits Berührungspunkte mit Musik hatten. Durch Vorwissen im musikalischen Bereich ist es einfacher einen Anhaltspunkt zu finden. In der Regel wird das bewusste „Spielen" einer bestimmten Emotion auch dadurch erleichtert, dass eine Person, die mit Musik vertraut ist, sich den gewünschten Sound besser vorstellen kann.

Dem gegenüber steht ein „intuitiver" Ansatz, bei dem die Aufnahme gestartet wird, ohne dass die Testperson eine bestimmte Emotion anpeilt. Der resultierende Datensatz wird im Nachhinein bewertet und gelabelt. Dieser Ansatz wird meist von Testpersonen verfolgt, die keine Erfahrung mit dem Klavierspielen oder Musik im Allgemeinen und daher keinen Anhaltspunkt haben.

#### *9.2.2.1 Equipment*

Die Aufnahme der eingespielten Samples erfolgt mit einem MIDI-tauglichen Keyboard bzw. E-Piano. Über ein Kabel wird eine Verbindung zu einem Endgerät hergestellt, auf dem der MIDI-Stream mit dem MIDI-Editor aufgezeichnet werden kann.

Alternativ kann auch ein digitaler Emulator, wie die MIDI-Keyboard App verwendet werden, um einen Input zu erzeugen. Für Data Recording im großen Stil sind solche Lösungen allerdings nicht geeignet, da das Spielen so zusätzlich erschwert und in die Länge gezogen wird. Aufnahmen mit digitalen Hilfsmitteln eigenen sich besser für Testzwecke oder Demonstrationen und sind weitaus mobiler als ein vollständiges Keyboard-Setup.

### 9.2.3 Herausforderungen

Das Recording eigener Daten bringt einige Herausforderungen mit sich:

Der Aufnahmeprozess ist sehr zeitaufwendig, da für jeden Teilnehmer mehrere Aufnahmen benötigt werden. Teilweise müssen diese mehrmals aufgenommen werden, um ein verwertbares Ergebnis zu produzieren. Die daraus resultierenden Files müssen gelabelt und mitunter geschnitten werden. Da oftmals auch Erklärungen zum Prozess und dem Projekt an sich notwendig sind, kann nur eine begrenzte Anzahl von Teilnehmern abgearbeitet werden.

Die Aufnahme an sich fordert zusätzlich heraus, da es für Personen, die keine Erfahrung mit einem Klavier haben sehr schwierig sein kann brauchbare Aufnahmen zu produzieren. Es ist knifflig den vorgestellten, gewünschten Sound zu produzieren, wenn kein Wissen zur Handhabe des Instruments vorhanden ist. Zum einen fehlt die Vorstellungskraft, zum anderen sind die technischen Fertigkeiten nicht ausgeprägt genug, um die Vision auch umzusetzen.

Die starken Schwankungen des Könnens zwischen den einzelnen Testpersonen stellen eine weitere Problematik dar. Es ist mit so wenigen Datensätzen nicht möglich ein gleichmäßiges Datenset zusammenzustellen, da zwischen den verschiedenen Datensätzen so gravierende Unterschiede herrschen.

## 9.3 Training und Vergleich verschiedener Machine Learning-Modelle

Dieses Kapitel beschreibt welche einzelnen Modellklassen wie und warum trainiert worden sind und wie deren Performance evaluiert worden ist. Dabei gibt das Kapitel zusätzlich einen grundlegenden Überblick über das Vorgehen bei Maschine Learning-Thematiken im Allgemeinen. Am Ende des Kapitels werden die einzelnen Modelle der einzelnen Modellklassen miteinander verglichen und bewertet.

### 9.3.1 Supervised versus Unsupervised Learning

Supervised Learning bedeutet, dass jedem Eingabedatensatz x ein Ausgabewert y zugeordnet werden kann. Das Ziel des Supervised Learnings ist es, basierend auf einer großen Anzahl an Eingabe-Ausgabe-Paaren (x, y) ein Modell zu trainieren, welches Zusammenhänge zwischen den Eingabe-Ausgabe-Paaren (x, y) erkennt, sodass für einen dem Modell noch unbekannten Datensatz x das Modell den dazugehörigen Ausgabewert y finden kann. [23]

Im Gegensatz dazu beschreibt Unsupervised Learning den Fall, dass es zwar Eingabedatensätze x gibt, aber ihnen keine Ausgabewerte y zugeordnet werden können. Ziel von Unsupervised Learning ist es daher Muster, Zusammenhänge, Gruppierungen und Beziehungen zwischen den einzelnen Eingabedatensätzen zu finden. [23]

Anhand der Aufgabenstellung des Projekts lässt sich klar erkennen, dass es sich um ein Supervised Learning-Problem handelt, da Eingabedaten nach Emotionen klassifiziert werden sollten. Dahingehend sind im Projekt auch die Eingabedatensätze (MIDI-Dateien beziehungsweise Datensätze bestehend aus extrahierten Features) mit Ausgabewerten/Labels (Emotionen beziehungsweise Quadranten) gekennzeichnet. Somit muss hier keine neue Klassifizierung für die Eingabedaten beziehungsweise Zusammenhänge zwischen den Eingabedaten gefunden werden und dadurch gibt die Anwendung von Unsupervised Learning hier keinen Sinn.

### 9.3.2 Klassifikation

Im Zuge von Supervised Learning wird grundsätzlich zwischen Regression und Klassifikation unterschieden. Regression bedeutet, dass eine zu bestimmende Variable quantitative Werte, sprich einen nummerischen Wertebereich annehmen kann. Bei dem Erkennen von Emotionen handelt es sich um ein Klassifikations-problem, da Emotionen einen qualitativen Wertebereich annehmen können. [23] Qualitativ bedeutet, dass eine Variable keine numerischen Werte, sondern einen Wert aus K verschiedenen Klassen oder Kategorien annehmen kann. Beispiele dafür sind Farben (rot, gelb oder grün), das Geschlecht (weiblich, männlich oder divers) oder eben Emotionen (glücklich, traurig, ausgeglichen, wütend, verärgert etc.). Um das zu trainierende Modell zu vereinfachen und da im EMOPIA-Daten-set die beinhalteten MIDI-Dateien nach (vier) Quadranten (die wiederum Emotio-nen zugeordnet werden können) klassifiziert werden [24], erkennt CEPP nur vier verschiedene Emotionen (siehe Abb. 13):

- Happy (Q1)
- Angry (Q2)
- Sad (Q3)
- Tender (Q4)

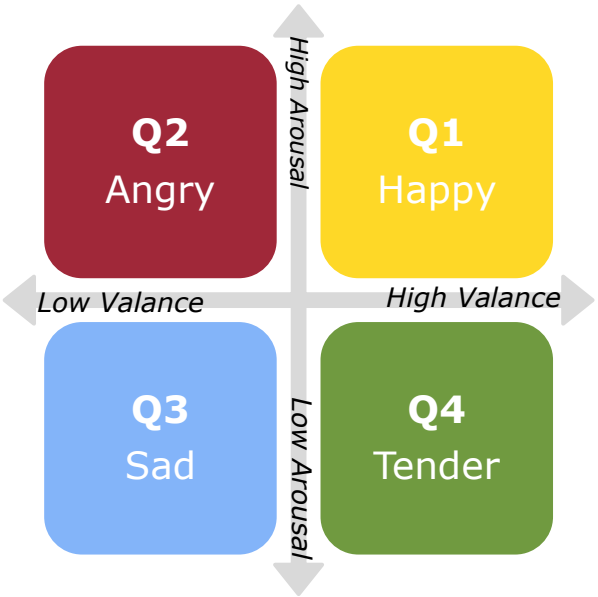



*Abb. 13: Zuordnung der Emotionen Happy, Angry, Sad und Tender zu Quadranten*

### 9.3.3 Versuchsaufbau

Um einen Vergleich der verschiedenen Modelle zu ermöglichen, ist ein Versuchsaufbau festgelegt worden. Als Ergebnis ist für jedes Modell ein Jupyter-Notebook angefertigt worden.

Der Versuchsaufbau selbst besteht aus folgenden aufeinanderfolgenden Schritten:

1. Import der Datensätze in das Programm
2. Partitionierung der Datensätze in Trainings- und Test-Datenset
3. Vorverarbeitung der Daten: Standardisierung der Features
4. Modelle unterschiedlicher Modellklassen und mit verschiedenen Parameter-Kombinationen trainieren und vergleichen (Grid-Search)
5. Trainieren eines finalen Modells
6. Evaluieren des Modells mit Hilfe der Berechnung verschiedener Vergleichswerte

#### *9.3.3.1 Import der Datensätze*

Die Datensätze, mit denen die Modelle trainiert werden bestehen aus den Features, die aus den MIDI-Dateien des Datensets mithilfe von jSymbolic extrahiert wurden. Das Datenset selbst umfasst sowohl die MIDI-Dateien des Emopia-Datensets als auch die im Zuge des Projektes selbst aufgenommenen Features. Die Features selbst werden jSymbolic als Datei im CSV-Format exportiert.

Der Import der Datensätze in das Programm erfolgt mit Hilfe der Python-Bibliothek pandas, welches die Daten als Data Frame importiert. Um von Scikit-learn verarbeitbar zu sein, muss der importierte Data Frame in zwei Teile, einem der die ganzen Eingabedaten X (Features) beinhaltet und einem anderen der die dazugehörigen Klassifikation Y beinhaltet, aufgeteilt werden und zusätzlich zu NumPy-Arrays konvertiert werden (siehe Listing 2).

```
import pandas as pd
import numpy as np

# Method for File Import
def load_dataset(csv_file):
    # load csv file
    dataset = pd.read_csv(csv_file, sep=',')

    # scikit-learn can't handle NaN
    dataset = dataset.replace(' NaN', 0)

    # split datasets in
    # X ... containing all the featuers (extract the features as numpy array
    #                                    (via .values))
    # Y ... containing all the labels
    X = dataset.loc[:, 'Number_of_Pitches':'Avg_Note_to_Note_Change'].values
    Y = dataset.loc[:, 'Label'].to_numpy()

    return X, Y
```

*Listing 2: Methode zum Import der Datensätze*

#### *9.3.3.2 Train-Test Split*

Ein Train-Test Split wird genutzt, um die Performance eines trainierten Machine Learning-Modells zu evaluieren. Als Performance eines Modells versteht man in diesem Fall die Richtigkeit der vom Modell vorausgesagten Werte auf dem Test-Set.

Grundsätzlich kann man ein Machine Learning-Modell evaluieren, indem man für Eingabedaten, die (im Falle von Klassifikation) fest einer Klasse zugeordnet sind, die Klasse vom Machine Learning-Modell voraussagen lässt. Nun kann man die tatsächliche mit der vom Modell vorausgesagten Klasse vergleichen, wobei diese Werte entweder übereinstimmen oder nicht, sprich die Voraussage des Modells richtig oder falsch gewesen ist. Wiederholt man diesen Vorgang mehrere Male, kann man abschätzen, wie gut ein Modell performt.

Problematisch ist dabei aber, wenn man zum Abschätzen der Performance die Daten verwendet, mit denen das Modell zuvor trainiert worden ist. Da das Modell diese Daten schon kennt, kann man somit nur abschätzen, wie gut es bereits gelernte Zuordnungen zwischen Eingabedaten und Ausgabeklassen reproduzieren kann. In diesem Zusammenhang spricht man auch von einem Training Error.

Um ein aussagekräftigeres Ergebnis über die Performance des Modells auf noch unbekannten Daten zu bekommen, kann man das zum Trainieren des Modells vorhandene Datenset in ein Trainings- und ein Test-Set unterteilen. Dabei wird das Modell nur mit den Daten des Trainings-Sets trainiert und zur Evaluierung werden die Daten des Test-Sets verwendet. In diesen Zusammenhang ist auch oft vom Test-Error die Rede.

Obwohl ein Train-Test-Split eine sehr gute Evaluation der Performance des Modells für zukünftige Eingabedaten bietet, wird der Train-Test Split vor allem dazu verwendet, einen Vergleich zwischen den trainierten Modellen verschiedener Modellklassen zu ermöglichen.

Zusätzlich ist der Train-Test-Split bei dem Versuchsaufbau dazu verwendet worden, mehrere Modelle einer Modellkasse mit verschiedenen Parameterkombinationen zu trainieren und zu evaluieren, damit die beste Parameterkombination gefunden werden kann. Häufig findet in diesem Zusammenhang aber auch S-fold Cross Validation in Verwendung. Vor dem Versuchsaufbau selbst sind zwar auch Versuche mit Cross Validation durchgeführt worden. Da aber die Ergebnisse der

einzelnen Parameter Kombinationen auch ohne Cross Validation sehr eindeutig sind, wird im Versuchsaufbau der Train-Test-Split auch zum Austesten der Parameterkombinationen herangezogen.

### S-fold Cross Validation

Beim Ansatz der S-fold Cross Validation wird das Trainingsdatenset in S gleichgroße Teile aufgeteilt. Danach werden S verschiedene Modelle mit den gleichen Parameterkombinationen trainiert, wobei immer ein anderer Teil der S Teile als Training-Set und der Rest als Test-Set verwendet wird. Am Ende wird die durchschnittliche Performance der S Durchläufe berechnet. Ein Nachteil besonders bei rechenintensiven Modellen ist, dass die Anzahl an zu trainierenden Modellen mit dem Faktor S multipliziert wird. [25]

### Umsetzung des Train-Test Split beim Versuchsaufbau

Eine Möglichkeit ist es, die zur Verfügung stehenden Daten randomisiert in ein Trainings- und ein Test-Set aufzuteilen. Dies hat jedoch den Nachteil, dass man nicht hundertprozentig garantieren kann, dass sowohl im Trainings-Set als auch im Test-Set alle (im Falle von CEPP) Emotionsklassen vorkommen und in etwa gleich verteilt sind. Eine weitere Möglichkeit, um einen solchen Split zu erzeugen, bietet die Stratified-Shuffle-Split Methode von Scikit-Learn [26].

Aus diesem Grund sind für den Versuchsaufbau die Datensätze manuell in zwei CSV-Dateien aufgeteilt worden. Somit wird auch garantiert, dass im Trainings- und im Test-Set das Verhältnis zwischen EMOPIA-MIDI-Aufnahmen und selbst aufgenommen MIDI-Dateien gleich ist. Dieser Aspekt ist dahingehend entscheidend, da es erheblich weniger selbst aufgenommene Midi-Files gibt. Jedoch sind selbst aufgenommene Midis aber um einiges repräsentativer für die Daten, basierend auf denen das Modell in Zukunft Vorhersagen treffen muss.

Ein weiterer Vorteil ist, dass der Versuchsaufbau dadurch repetitiv ist. Da die einzelnen Modelle immer mit denselben Daten trainiert werden, werden immer dieselben Ergebnisse geliefert.

Als Verhältnis zwischen Trainings und Test-Daten ist 75:25 gewählt worden wobei die Wahl dieses Verhältnisses stark von der Anzahl an verfügbaren Daten abhängig ist.

#### *9.3.3.3 Bias-Variance Tradeoff*

Das Ziel beim Trainieren eines Modells ist, wie schon im Abschnitt 9.3.3.2 Train-Test-Split beschrieben, dass man ein Modell findet, das sowohl die in den Trainingsdaten vorhanden Regeln lernt aber auch generalisiert, sodass es auch in der Lage ist, unbekannte Daten korrekt zu klassifizieren und nicht nur die bereits gelernten Trainingsdaten. Generalisiert ein Modell zu wenig, wird dies als Overfitting bezeichnet, zu Deutsch das Modell ist überangepasst an die Trainingsdaten. Das Gegenteil davon ist Underfitting, sprich das Modell generalisiert zu stark, sodass es nicht in der Lage ist, die einfachsten Regelmäßigkeiten in den Trainingsdaten festzustellen. Hier gilt es, die Balance zwischen Underfitting und Overfitting zu finden, wie auch Abb 14. veranschaulicht. [27]

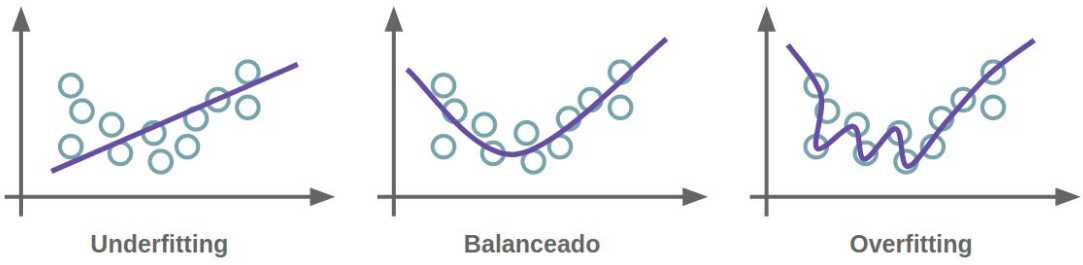



*Abb. 14: Grafische Darstellung von Under- und Overfitting (https://commons.wikimedia.org/wiki/File:Underfitting_e_overfitting.png)*

Im Zusammenhang dazu steht ein gängiges Problem im Feld des Maschine Learnings, das Bias-Variance-Tradeoff. Als Bias Error wird der Fehler bezeichnet, der durch fehlerhafte Annahmen im Lernalgorithmus entsteht. Ein hohes Bias führt dazu, dass ein Modell die relevanten Beziehungen zwischen Inputvariablen und Output-Klassen nicht erfassen kann, also zum Underfitting neigt. Unter Variance Error versteht man den Fehler, der durch zu hohe Sensitivität des Lernalgorithmus auf auch schon kleine Schwankungen in den Trainingsdaten entsteht. Ein Modell mit hoher Variance neigt daher eher zum Overfitting. [28]

Der Begriff Tradeoff wird verwendet, da es meist nicht möglich ist, ein Modell mit gleichzeitig niedrigem Bias und niedriger Variance zu finden und es daher einen Kompromiss zwischen Bias und Variance zu finden gibt.

#### *9.3.3.4 Vorverarbeitung der Daten*

Ein wichtiger Punkt ist die Vorverarbeitung der Features mit denen das Modell trainiert, getestet etc. wird, da viele der in Scikit-learn implementierten Machine Learning-Modellklassen meist bessere Ergebnisse mit standardisierten Daten liefern. Gründe dafür sind, dass die einzelnen Features sehr unterschiedliche Wertebereiche haben. Scikit-learn bietet dazu verschiedenste Scaler, Transformer und Normalizer. [29]

##### Standardisierung der Daten

Eine oft verwendete Form von Skalierung im Bereich Machine Learning ist die Standardisierung der einzelnen Features. Ein Wert x wird um Mittelwert μ aller Werte eines Features subtrahiert und danach durch die Standardabweichung σ der Feature-Werte dividiert:

$$Z = \frac{(x - \mu)}{\sigma}$$

Dadurch werden die Feature-Werte standardnormalverteilt mit einen Mittelwert von 0 und einer Standardabweichung von eins.

Eine Standardnormalverteilung wird im Versuchsaufbau auch auf die Trainings- und Testdaten angewendet. Dazu wird der `StandardScaler` von Scikit-learn verwendet wie im Listing 3 zu sehen ist. Hierbei ist wichtig, dass nur auf das Trainings-Set `fit_transform()` angewendet wird, da der Scaler dabei die Variablen Mittelwert und Standardabweichung der Features feststellt und speichert und diese danach skaliert. Das Test-Set muss mit `transform()` nur nach dem schon gespeicherten Mittelwert und der gespeicherten Standardabweichung skaliert werden und es darf keine Neuberechnung erfolgen, da sonst die Daten nicht zusammenpassen und das Modell dadurch möglicherweise nicht mehr gut bzw. richtig funktioniert. [30]

```
from sklearn.preprocessing import StandardScaler

# Normalise features in the training set
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
# Apply the same scaling and shifting operations as performed on the training
data
X_test = scaler.transform(X_test)
```

*Listing 3: Skalierung der Features mit dem StandardScaler von Scikit-learn*

### *9.3.3.5 Modelle vergleichen – Vergleichswerte und -Metriken*

Um die Performance unterschiedlicher Klassifikationsmodelle und -Modellklassen vergleichen zu können, muss zuerst definiert werden, welche Resultate es in Bezug auf eine bestimmte Klasse y bei der Vorhersage einer Klasse geben kann. Wenn die vorhergesagte Klasse der ‚wahren' Klasse y entspricht, so ist die Vorhersage wahr-positiv, zusätzlich dazu gibt es noch wahr-negativ, wenn eine andere Klasse vorausgesagt worden ist und die ‚wahre' Klasse sowieso nicht y ist. Wenn die ‚wahre' Klasse y ist, aber die vorausgesagte Klasse eine andere, ist von falsch-negativ die Rede. Ist aber die vorausgesagte Klasse y und die eigentliche Klasse eine andere spricht man von falsch-positiv. Eine Veranschaulichung bietet Abb. 15. [31]

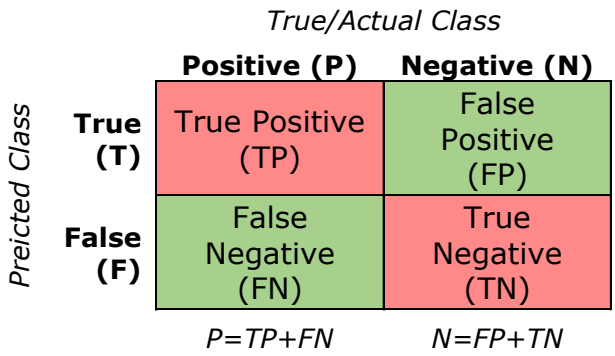



*Abb. 15: Binary Confusion Matrix*

Folgende Vergleichswerte und Metriken sind für den Versuchsaufbau herangezogen worden:

#### Confusion Matrix

Die vorher aufwendig beschriebenen Resultate lassen sich, wie Abb. 15 zeigt, mit Hilfe einer binären Confusion Matrix veranschaulichen. Eine Confusion Matrix, (auch als Error Matrix bekannt) ist somit im Bereich Klassifikation eine Möglichkeit die Performance eines Supervised Learning Algorithmus darzustellen. Bei einer Confusion Matrix mit mehreren Klassen wird in den Spalten die tatsächliche Klasse angeführt und in den Reihen die vorhausgesagte Klasse. Die einzelnen Kreuzungspunkte geben an, wie viele Datensätze, die einer tatsächlichen Klasse zugeordnet sind, als eine bestimmte Klasse vorhergesagt worden sind und umgekehrt. Abb. 16 veranschaulicht, wie eine Confusion Matrix bei der Auswertung eines Modells des Versuchsaufbaus aussehen könnte. Oft wird eine Confusion Matrix normalisiert, wie zusätzlich in Abb. 16 zu sehen ist.

*Confusion Matrix zur Klassfikation nach Emotionsquadranten*

| Predicted Emotion \ Actual Emotion | Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|---|
| **Q1** | 27 | 16 | 0 | 2 |
| **Q2** | 14 | 18 | 2 | 2 |
| **Q3** | 4 | 4 | 49 | 15 |
| **Q4** | 8 | 4 | 25 | 24 |

*Normalisierte Confusion Matrix*

| Predicted Emotion \ Actual Emotion | Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|---|
| **Q1** | 0.6 | 0.35 | 0.0 | 0.04 |
| **Q2** | 0.39 | 0.5 | 0.05 | 0.06 |
| **Q3** | 0.06 | 0.05 | 0.68 | 0.21 |
| **Q4** | 0.13 | 0.06 | 0.41 | 0.39 |

*Abb. 16: Confusion Matrizen zum Evaluieren trainierter Modelle im Versuchsaufbau*

### Accuracy

Die Accuracy gibt an wie viele Prozent aller vorhergesagten Klassifikationen eines Modells tatsächlich richtig sind [32]:

$$Accuracy = \frac{correct\ classifications}{all\ classifications}$$

Die Accuracy ist im Versuchsaufbau dazu verwendet worden, die verschiedenen Parameter einer Modellklasse zu optimieren.

### Recall

Der Recall einer Klasse ist darin definiert, wie viel Prozent der tatsächlichen Datensätze einer Klasse auch tatsächlich als solche vorausgesagt worden sind. [33] Ein Beispiel wäre wie viel Prozent der als „ Sad" gekennzeichneten MIDI-Dateien auch vom Modell als „Sad" klassifiziert worden sind. Folgende Formel kann dabei zur Berechnung des Recalls herangezogen werden:

$$Recall = \frac{TP}{TP + FN}$$

Da man den Recall nur für eine bestimmte Klasse berechnen kann, bietet Scikit-learn die Möglichkeit einen durchschnittlichen Recall über alle Klassen hinweg zu berechnen. Dazu gibt es unterschiedliche Berechnungsmethoden, wobei nur der

UAR, also der Unweighted Average Recall, für Vergleiche herangezogen worden ist. Dafür wird für jede Klasse einzeln der Recall berechnet und danach der ungewichtete Durchschnitt der Recalls berechnet. [34]

## Precision

Die Precision einer Klasse beschreibt wie viel Prozent aller als eine bestimmte Klasse vorhergesagten Datensätze auch der Klasse entsprechen. [33] Ein Beispiel dafür wäre, wie viele als „Happy" klassifizierte MIDI-Files auch wirklich „Happy" sind. Formel für die Berechnung der Precision für eine Klasse wird folgende verwendet:

$$Precision = \frac{TP}{TP + FP}$$

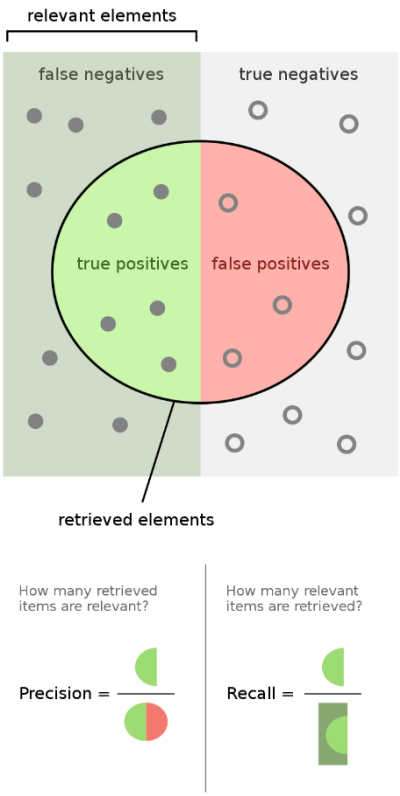



*Abb. 17: Veranschaulichung von Recall und Precision (https://en.wikipedia.org/wiki/Precision_and_recall)*

Eine Veranschaulichung von Recall und Precision bietet Abb. 17.

### Berechnung mit Scikit-learn

Listing 4 zeigt die Berechnungen der einzelnen Vergleichswerte und -indikatoren mit Hilfe von Scikit-learn. Dabei ist `y_pred` ein Array von vom Modell vorhergesagten Emotionen/Quadranten und `y_true` ein Array mit den dazugehörigen, tatsächlichen Werten.

```
# Train Model
clf.fit(X_train, Y_train.ravel())

# Let Model predict Classes for the Testdata
Y_pred = clf.predict(X_test)


# Evaluate Performance

# Confusion Matrix
cm = sklearn.metrics.confusion_matrix(y_true, y_pred,
                                      labels=['Q1', 'Q2', 'Q3', 'Q4'])
# Normalized Confusion Matrix
cm_n = sklearn.metrics.confusion_matrix(y_true, y_pred, normalize='true',
                                        labels=['Q1', 'Q2', 'Q3', 'Q4'])

# Accuracy
acc = sklearn.metrics.accuracy_score(y_true, y_pred)

# Unweighted Average Recall
UAR = sklearn.metrics.recall_score(y_true, y_pred, average='macro')

# Compute recall and precision score for each class (returns Array)
recall_result = sklearn.metrics.recall_score(y_true, y_pred, average=None,
                                             labels=['Q1', 'Q2', 'Q3', 'Q4'])

precision_result = sklearn.metrics.precision_score(y_true, y_pred,
                                   average=None, labels=['Q1', 'Q2', 'Q3', 'Q4'])
```

*Listing 4: Berechnung der Vergleichswerte mit Hilfe von Scikit-learn*

### 9.3.4 Trainieren von Modellen unterschiedlicher Modellklassen

Um das beste Modell zu finden sind verschiedene Modellklassen trainiert und die Parameter der Implementation optimiert worden. Für jede Modellklasse wurde als Ergebnis ein eigenes Jupyter Notebook erstellt.

#### *9.3.4.1 k-NN – k-nearest-neighbors algorithm*

Bei k-NN Klassifikation wird ein Datenpunkt dadurch klassifiziert, indem er mit seinen k nächsten Datenpunkten aus einem Set aus schon klassifizierten Datenpunkten verglichen wird. Dabei wird analysiert welche Klasse am häufigsten in den k nächsten Datenpunkten vorkommt und diese nimmt dann auch der vorherzusagende Datenpunkt an. [35] Abb. 18 bildet den k-NN-Algorithmus für den zwei-dimensionalen Raum ab.

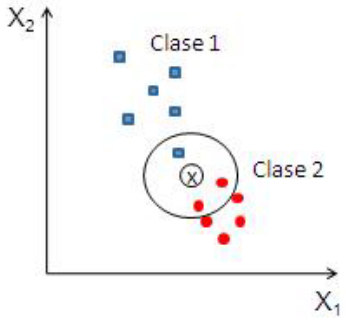



*Abb. 18: kNN-Algorithmus (https://commons.wikimedia.org/wiki/File:KNN_detec.JPG)*

#### k-NN in Scikit-learn

Scikit-learn stellt k-NN-Klassifikation mit der Klasse `sklearn.neighbors.KNeighborsClassifier()` zur Verfügung. Beim Erstellen eines KNeighborsClassifier (siehe Listing 5) können verschiedene Parameter gesetzt werden. Dabei ist der wichtigste Parameter `n_neighbors`. `n_neighbors` ist k gleichzusetzen und gibt an, wie viele Nachbarpunkte zur Klassifikation eines Wertes herangezogen werden. [36] Mit k lässt sich beeinflussen wie Under- oder Overfitting ein Modell ist. Bei Modellen mit einem niedrigen k beziehungsweise einem k von eins wird ziemlich sicher das Phänomen des Overfitting auftreten, da es bei einem k von eins alle bereits gelernten Datenpunkte richtig klassifizieren wird, denn dabei ist der nächste Datenpunkt genau der Datenpunkt, der vorausgesagt werden soll, jedoch bei unbekannten Datenpunkten gibt es diesen einen zuordenbaren Datenpunkt nicht.

```
from sklearn import neighbors
clf = neighbors.KNeighborsClassifier(n_neighbors=k)
```

*Listing 5: KNeighborsClassifier*

#### Ergebnisse

Die beste Performance nach Austesten verschiedener k's hat ein Modell mit einen k von 25 geliefert, für eine detaillierte Auswertung und Analyse der Performance siehe Kapitel 9.3.5 Vergleich der Modellklassen.

#### *9.3.4.2 SVM – Support Vector Machine*

Beim Training eines Modells basierend auf einem SVM-Algorithmus wird aus einer Reihe aus Trainingsdatensätzen, die jeweils einer Klasse zugeordnet sind, ein Modell erstellt, sodass zwischen den einzelnen Klassen eine möglichst breite Trennlinie entsteht. [37] Zu klassifizierende Datensätze werden je nachdem auf welcher Seite (im Falle von nur zwei Klassen siehe Abb. 19) beziehungsweise zwischen welchen Trennlinien sie eingeschlossen sind klassifiziert. Die Art Trennlinie wird auch als Kernel bezeichnet und dieser muss nicht zwingend linear sein, sondern kann auch eine Polynomfunktion, eine radiale Basisfunktion oder eine Sigmoidfunktion sein. [38] Abb. 19 zeigt auf der linken Seite eine zweidimensionale SVM mit Polynomfunktion als Kernel und rechts eine SVM mit linearem Kernel.

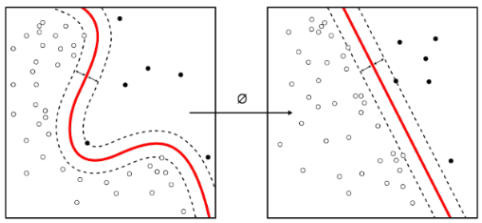


*Abb. 19: SVM mit Polynomfunktion als Kernel (rechts) und SVM mit linearen Kernel (links) (https://en.wikipedia.org/wiki/Support_vector_machine)*

##### SVM in Scikit-learn

Mit der Klasse `sklearn.svm.SVC()` stellt Scikit-learn eine Implementierung einer SVM für Klassifikationsprobleme zur Verfügung. Im Versuchsaufbau wurden folgende beim Erstellen einer SVM in Scikit-learn konfigurierbaren Parameter versucht zu optimieren:

- `kernel`: Dieser Parameter gibt den Kerneltyp an. Folgende Kernel sind getestet worden: ‚linear', ‚poly', ‚rbf', ‚sigmoid'
- `C`: C ist ein Regulierungsparameter, welcher positiv sein muss. Hierbei sind verschiedene Werte ausgetestet worden. [39]

```
from sklearn import svm
clf = svm.SVC(C=c, kernel=kernel)
```

*Listing 6: SVM mit Parametern*

##### Ergebnisse

Beim Optimieren der Parameter wurde festgestellt, dass eine SVM mit linearem Kernel und einer Complexity von C=0.005 die höchste Accuracy hat. Für eine detaillierte Auswertung der Performance siehe Kapitel 9.3.5 Vergleich der Modellklassen.

#### *9.3.4.3 Multi-Layer Perceptron (MLP)*

Ein Multi-Layer Perceptron ist ein neuronales Netzwerk, welches aus mehreren Schichten von Perzeptren besteht. Ein Perzeptron selbst kann man sich als Funktion vorstellen, die verschiedene Inputwerte mit verschiedenen Gewichtungen entgegennimmt und daraus einen Output erstellt, siehe Abb. 21.

Ein Multi-Layer Perceptron selbst besteht aus mindestens drei Schichten (Layer), wobei die erste Schicht der Input Layer ist, der die Features entgegennimmt und die letzte Schicht der Output Layer, der die endgültige Klasse bestimmt. Dazwischen befinden sich sogenannte Hidden Layers, die Perceptren beinhalten. Die Abb. 20 bildet dies grafisch ab. [40]

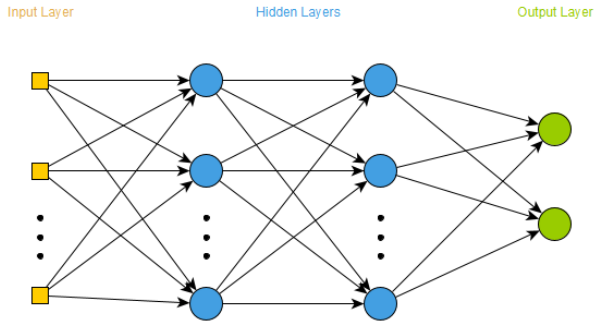



*Abb. 20: MLP*

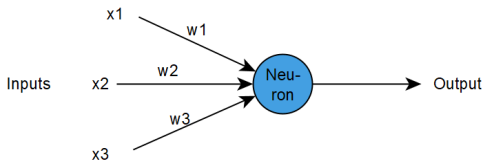



*Abb. 21: Neuron*

### MLP in scikit-learn

Mit `sklearn.neural_network.MLPClassifier()` kann man in Scikit-learn ein Multi-Layer Perceptron erstellen (siehe Listing). Dabei sind im Versuchsaufbau folgende Parameter optimiert bzw. konfiguriert worden:

- `hidden_layer_sizes`: Mit hidden_layer_sizes kann ein Array übergeben werden, das mit der Länge des Arrays die Anzahl der Hidden Layers bestimmt und die einzelnen Felder im Array die Anzahl der Neuronen des jeweiligen Layers
- `solver`: legt den Solver fest. Folgende Solver stehen zu Verfügung: ‘lbfgs’, ‘sgd’, ‘adam’
- `alpha`: Stärke des L2 regularisation terms.
- `random_state`: bestimmt den Zufallszahlengenerator, der benutzt wird. Legt man einen Integer als `random_state` fest, wird ermöglicht, dass das Ergebnis über mehrere Aufrufe hinweg reproduzierbar ist.

- `activation`: legt die Art der Aktivierungsfunktion für die Hidden Layers fest. Folgende Arten können festgelegt werden: ‘identity’, ‘logistic’, ‘tanh’, ‘relu’
- `max_iter`: legt die maximale Anzahl an Iterationen fest, wobei für den Versuchsaufbau, der eine maximale Anzahl von 1000 Iterationen festgelegt worden ist.

```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(solver=solver, alpha=alpha,
                    hidden_layer_sizes=hidden_layer_size,
                    random_state=42, activation=activation,
                    max_iter=1000)
```

*Listing 7: MLPClassifier*

### Ergebnisse

Die besten Ergebnisse beim Training lieferte ein Multi-Layer Perceptron mit einer Logistischen Sigmoid Funktion als Activation Function, einem Alpha von 0,001 einem ’lbgfs‘-Solver und drei Hidden Layers, wobei der erste 100, der zweite 50 und der dritte 33 Neuronen hat. Für eine genaue Analyse der Performance siehe Kapitel 9.3.2 Vergleich der Modellklassen.

#### *9.3.4.4 Random Forest*

Ein Random Forest besteht aus vielen Classification Trees. Bei solchen Classification Trees bzw. Entscheidungsbäumen handelt es sich um Algorithmen, die sowohl bei Klassifikations- als auch Regressionsproblemen eingesetzt werden können. Muss nun ein Random Forest einen Eingabedatensatz klassifizieren, werden jedem Classification Tree der Eingabedatensatz übergeben, sodass jeder eine eigenständige Klassifikation für die Eingabedaten voraussagt. Der Random Forest vergleicht nun die vorausgesagten Klassifikationen der einzelnen Classification Trees passierend auf welche Klassifikation am häufigsten vorausgesagt worden ist und diese ist dann sein endgültiges Ergebnis (siehe Abb. 22). [41]

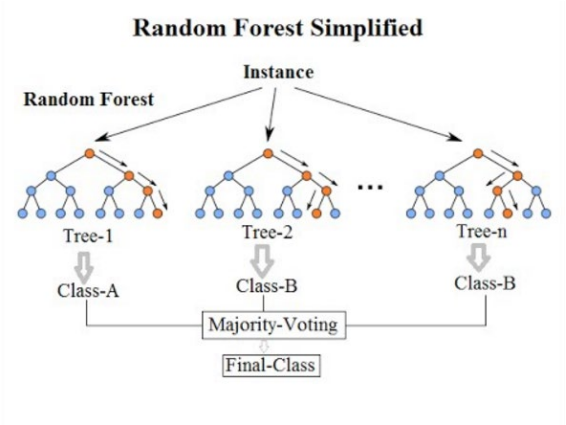



*Abb. 22: Random Forest (https://commons.wikimedia.org/wiki/File:Random_forest_diagram_complete.png)*

### Random Forests in Scikit-learn

Mit dem `sklearn.ensemble.RandomForestClassifier()` bietet sckit-learn (siehe Listing 8) die Möglichkeit ein Klassifikationsmodell basierend auf dem Konzept der Random Forests zu trainieren. Für den Versuchsaufbau sind dabei folgende Parameter angepasst worden:

- `n_estimators`: bestimmt die Anzahl an Trees des Random Forests.
- `random_state`: bestimmt den Zufallszahlengenerator, der benutzt wird. Legt man einen Integer als random_state fest, wird ermöglicht, dass das Ergebnis über mehrere Aufrufe hinweg reproduzierbar ist.

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state=42, n_estimators=estimator)
```

*Listing 8: RandomForestClassifier*

### Ergebnisse

Aus der Optimierung der Parameter hat resultiert, dass ein Random Forest mit 200 Decision Trees die besten Ergebnisse liefert. Kapitel 9.3.5 Vergleich der Modellklassen bietet einen Überblick über die Performance des Random Forest.

### 9.3.5 Vergleich der Modellklassen

Um die Modellklassen vergleichen zu können sind Performancewerte, wie Accuracy, Recall und Precision für die optimierten, mit den Trainingsdaten trainierten Modelle der einzelnen Modellklassen berechnet und in einer Tabelle veranschaulicht worden. Die Performancewerte geben die Performance auf den Testdaten an.

<table>
<tr><th></th><th>k-NN</th><th>SVM</th></tr>
<tr><td>Parameter</td><td>n_neighbors (k) = 25</td><td>kernel=`linear`<br>C=0.005</td></tr>
<tr><td colspan="3">Perfomance</td></tr>
<tr><td>Accuracy</td><td>56.4 %</td><td>63.8 %</td></tr>
<tr><td>UAR</td><td>55.5 %</td><td>63.3 %</td></tr>
<tr><td>Recall für Q1</td><td>83.5 %</td><td>65.6 %</td></tr>
<tr><td>Recall für Q2</td><td>36.2 %</td><td>57.9 %</td></tr>
<tr><td>Recall für Q3</td><td>23.5 %</td><td>55.8 %</td></tr>
<tr><td>Recall für Q4</td><td>79.0 %</td><td>74.0 %</td></tr>
<tr><td>Precision für Q1</td><td>51.3 %</td><td>60.2 %</td></tr>
<tr><td>Precision für Q2</td><td>86.2 %</td><td>74.0 %</td></tr>
<tr><td>Precision für Q3</td><td>59.2 %</td><td>60.3 %</td></tr>
<tr><td>Precision für Q4</td><td>53.3 %</td><td>63.1 %</td></tr>
<tr><td>Confusion Matrix</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>56</td><td>2</td><td>1</td><td>8</td></tr>
<tr><td>Q2</td><td>31</td><td>25</td><td>5</td><td>8</td></tr>
<tr><td>Q3</td><td>12</td><td>0</td><td>16</td><td>40</td></tr>
<tr><td>Q4</td><td>10</td><td>2</td><td>5</td><td>64</td></tr>
</table>
</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>44</td><td>11</td><td>3</td><td>9</td></tr>
<tr><td>Q2</td><td>16</td><td>40</td><td>9</td><td>4</td></tr>
<tr><td>Q3</td><td>7</td><td>1</td><td>38</td><td>22</td></tr>
<tr><td>Q4</td><td>6</td><td>2</td><td>13</td><td>60</td></tr>
</table>
</td></tr>
<tr><td>Normalisierte Confusion Matrix</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>0.83</td><td>0.02</td><td>0.01</td><td>0.11</td></tr>
<tr><td>Q2</td><td>0.44</td><td>0.36</td><td>0.07</td><td>0.11</td></tr>
<tr><td>Q3</td><td>0.17</td><td>0.0</td><td>0.23</td><td>0.58</td></tr>
<tr><td>Q4</td><td>0.12</td><td>0.02</td><td>0.06</td><td>0.79</td></tr>
</table>
</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>0.65</td><td>0.16</td><td>0.04</td><td>0.13</td></tr>
<tr><td>Q2</td><td>0.23</td><td>0.57</td><td>0.13</td><td>0.05</td></tr>
<tr><td>Q3</td><td>0.10</td><td>0.01</td><td>0.55</td><td>0.32</td></tr>
<tr><td>Q4</td><td>0.07</td><td>0.02</td><td>0.16</td><td>0.74</td></tr>
</table>
</td></tr>
</table>

*Tabelle 2: Ergebnisse kNN und SVM*

<table>
<tr><th></th><th>MLP</th><th>Random Forest</th></tr>
<tr><td>Parameter</td><td>activation='logistic'<br>alpha=0.001,<br>hidden_layer_sizes=[100, 50, 33]<br>solver='lbfgs'</td><td>n_estimators=200</td></tr>
<tr><td colspan="3">Perfomance</td></tr>
<tr><td>Accuracy</td><td>61.0 %</td><td>64.2 %</td></tr>
<tr><td>UAR</td><td>60.9 %</td><td>64.1 %</td></tr>
<tr><td>Recall für Q1</td><td>56.7 %</td><td>61.1 %</td></tr>
<tr><td>Recall für Q2</td><td>59.4 %</td><td>66.6 %</td></tr>
<tr><td>Recall für Q3</td><td>64.7 %</td><td>63.2 %</td></tr>
<tr><td>Recall für Q4</td><td>62.9 %</td><td>65.4 %</td></tr>
<tr><td>Precision für Q1</td><td>65.5 %</td><td>67.2 %</td></tr>
<tr><td>Precision für Q2</td><td>66.1 %</td><td>67.6 %</td></tr>
<tr><td>Precision für Q3</td><td>56.4 %</td><td>56.5 %</td></tr>
<tr><td>Precision für Q4</td><td>58.6 %</td><td>66.2 %</td></tr>
<tr><td>Confusion Matrix</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>38</td><td>15</td><td>5</td><td>9</td></tr>
<tr><td>Q2</td><td>17</td><td>41</td><td>5</td><td>6</td></tr>
<tr><td>Q3</td><td>0</td><td>3</td><td>44</td><td>21</td></tr>
<tr><td>Q4</td><td>3</td><td>3</td><td>24</td><td>51</td></tr>
</table>
</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>41</td><td>14</td><td>4</td><td>8</td></tr>
<tr><td>Q2</td><td>11</td><td>46</td><td>9</td><td>3</td></tr>
<tr><td>Q3</td><td>5</td><td>4</td><td>43</td><td>16</td></tr>
<tr><td>Q4</td><td>4</td><td>4</td><td>20</td><td>53</td></tr>
</table>
</td></tr>
<tr><td>Normalisierte Confusion Matrix</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>0.56</td><td>0.22</td><td>0.07</td><td>0.13</td></tr>
<tr><td>Q2</td><td>0.24</td><td>0.59</td><td>0.07</td><td>0.08</td></tr>
<tr><td>Q3</td><td>0.0</td><td>0.04</td><td>0.64</td><td>0.30</td></tr>
<tr><td>Q4</td><td>0.03</td><td>0.03</td><td>0.29</td><td>0.62</td></tr>
</table>
</td><td>
Actual Emotion<br>
Predicted
<table>
<tr><th></th><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr>
<tr><td>Q1</td><td>0.61</td><td>0.20</td><td>0.05</td><td>0.11</td></tr>
<tr><td>Q2</td><td>0.15</td><td>0.66</td><td>0.13</td><td>0.04</td></tr>
<tr><td>Q3</td><td>0.07</td><td>0.05</td><td>0.63</td><td>0.23</td></tr>
<tr><td>Q4</td><td>0.04</td><td>0.04</td><td>0.24</td><td>0.65</td></tr>
</table>
</td></tr>
</table>

*Tabelle 3: Ergebnisse MLP und Random Forest*

#### *9.3.5.1 Analyse*

Grundsätzlich lässt sich sagen, dass keines der Modelle wirklich sehr gut darin ist, Emotionen richtig zu klassifizieren. Dabei schneidet der Random Forest immer noch am besten ab mit einer Accuracy von 64,2% und sehr guten Werten, wenn es um Precision geht.

Die eindeutig schlechtesten Werte erzielte die k-NN mit einer Accuracy von 56,4%. Zwar hat sie sehr gute Recall-Werte für Q1 und Q4, aber der Recall für

Q2 und Q3 ist sehr schlecht. Bei einem Blick auf die Confusion Matrix der k-NN lässt sich erkennen, wieso das so ist, denn fast alle Datensätze werden entweder als Q1 oder Q4 klassifiziert. Somit klassifiziert zwar das Modell die Datensätze richtig als Q1 und Q4, aber auch die meisten Q2 und Q3 Datensätze werden als Q1 und Q4 missklassifiziert. Deswegen haben Q2 und Q3 einen sehr schlechten Recall, da insgesamt schon sehr wenige Datensätze als Q2 und Q3 klassifiziert werden.

Besser abgeschnitten hat das trainierte Multi-Layer Perceptron mit einer Accuracy von 61% und die trainierte Support Vector Machine mit einer Accuracy von 63,80%, womit die Accuracy in etwa 5% höher ist als bei der k-NN. Auffällig ist, dass die SVM mit 74% einen sehr hohen Recall für Q4 hat und im Gegensatz zur k-NN, welche einen noch höheren Recall für Q4 hat, auch eine akzeptablere Precision von 63,1%. Die beste Precision für eine Klasse hat auch die SVM mit einer 74% Precision von Q2.

Einen interessanten Einblick liefern auch die Confusion Matrices, hier kann man sehr gut erkennen, dass meist Q1 und Q2 miteinander vertauscht werden, aber Q1 und Q2 werden sehr selten als Q3 oder Q4 missklassifiziert. Umgekehrt gilt das auch für Q3 und Q4. Diese werden auch oft miteinander vertauscht, aber es gibt wenige Missklassifikationen mit Q1 oder Q2. Dieses Phänomen ist bei allen Modellklassen zu beobachten.

Um eine Erklärung dafür zu finden, gilt es einen Blick darauf zu werfen durch welche Eigenschaften sich die einzelnen Quadranten unterscheiden. So sind Musikstücke mit einem hohen Arousal immer entweder Q1 oder Q2 und Musikstücker mit einem niedrigen Arousal entweder Q3 oder Q4. Sowohl Q1 und Q2 als auch Q3 und Q4 lassen sich durch die Valance unterscheiden, wobei Q2 und Q3 eine niedrige und Q1 und Q4 eine hohe Valance besitzen. Daraus lässt sich schließen, dass die Modelle zwar leicht zwischen niedrigem und hohem Arousal unterscheiden können, aber schwer entscheiden können, ob die Feature-Werte eines bestimmten MIDI-Files nun auf eine hohe oder niedrige Valance hindeuten. Diese Missklassifikationen sind somit Resultat des schon in Kapitel 9.1.9 beschriebenen Valance/Arousal Problem.

### 9.3.6 Trainieren eines finalen Modells

Um entscheiden zu können welche Modellklasse nun für die Applikation verwendet wird, ist eine zusätzliche Analyse durchgeführt worden. Dabei ist die Performance auf den selbst aufgenommen Datensätzen des Test-Sets gemessen worden. Grund dafür ist, dass man so zusätzlich vielleicht ein wenig besser abschätzen kann, wie die einzelnen Modelle auf zukünftigen Input performen werden. Denn die selbstaufgenommen MIDI-Dateien sind den MIDI-Dateien ähnlicher, die die Modelle in Zukunft klassifizieren sollten, als die des EMOPIA-Datensets. Natürlich sind dabei die Analyse Ergebnisse mit großer Vorsicht zu behandeln, da das Test-Set nur 19 selbstaufgenommene Datensätze umfasst.

Das beste Ergebnis lieferte dabei das SVM-Modell mit einer Accuracy on 68,4%, dahinter das Multi-Layer Perceptron mit 63,2% und der Random Forest mit 52,6% Accuracy. Das k-NN-Modell schnitt auch hier am schlechtesten ab mit einer Accuracy von 52,6%.

Somit ist eine SVM mit linearem Kernel und einer Complexity von 0.005 nun mit allen Datensätzen, folglich die Datensätze des Trainings-Sets und des Test-Sets vereint, trainiert worden. Diese wird nun auch in der Applikation verwendet.

### 9.3.7 Transfer-Learning

Eine weitere Option beim Trainieren der Modelle wäre Transfer-Learning gewesen. Transfer-Learning beschreibt eine Methode im Bereich Machine Learning, wo ein Modell für eine Lernaufgabe benutzt wird und für eine andere Lernaufgabe als Einstiegspunkt wiederverwendet wird. [42] Grund dafür kann sein, dass man schon ein Machine Learning-Modell trainiert hat, welches auch schon in einen bestimmten Anwendungsbereich verwendet wird. Dieses Modell möchte man nun für eine ähnliche Aufgabe wiederverwenden und somit passt man es durch Transfer-Learning an die neue Aufgabe an. Weiters kann es auch bei dem Problem behilflich sein, dass man zwei Datensets hat, eines, das zwar sehr groß ist, aber die zukünftigen Daten nicht wirklich gut widerspiegelt und ein weiters, welches erheblich kleiner ist, aber dafür sehr gut die zukünftigen Daten widerspiegelt. Dabei würde das Modell zuerst mit dem großen Datenset trainiert werden und mit dem kleineren danach feinabgestimmt werden.

Im Falle des Projektes hätte das bedeutet, dass zum Trainieren der Modelle das EMOPIA-Datenset verwendet worden wäre und mit den selbstaufgenommen Datensätzen das trainierte Modell feinabgestimmt worden wäre.

# 9.4 CEPP-Anwendung

## 9.4.1 Prototyp

Der Prototyp für die CEPP-Anwendung besteht aus einigen Python-Modulen und PowerShell-Skripten, die auf lokal gespeicherte Daten und Executables zugreifen. Sofern ein bereits trainiertes Modell als Modul enthalten ist, kann die Anwendung unabhängig gestartet werden.

### *9.4.1.1 Aufbau*

Diese Abbildung verschafft einen genaueren Überblick über den Aufbau des Prototyps:

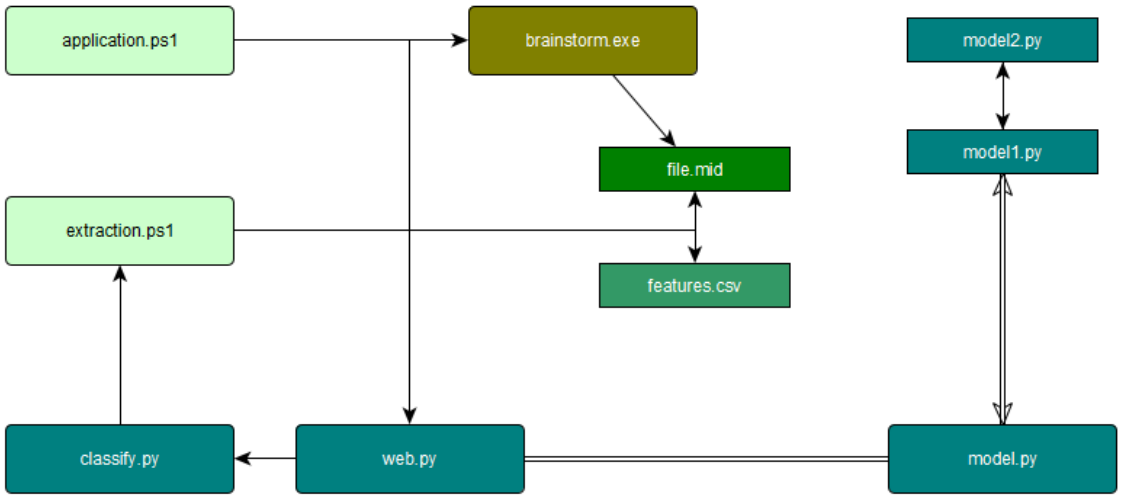



*Abb. 23: Aufbau der CEPP-App mit relevanten Bestandteilen und deren Zusammenwirken*

### *9.4.1.2 Enthaltene Komponenten*

| Komponente | Zweck |
|---|---|
| **application.ps1** | Start der Anwendung |
| **extraction.ps1** | Extrahiert Features aus dem aktuellen MIDI-File |
| **model.py** | Machine Learning-Modell |
| **web.py** | Grafische Darstellung und Aktualisierung |
| **classify.py** | Klassifiziert das aktuelle MIDI-File anhand der extrahierten Features |
| **brainstorm.exe** | MIDI-Utility zum automatisierten Aufnehmen von MIDI-Files |

*Tabelle 4: Relevante Komponenten in der Prototyp-Anwendung*

#### *9.4.1.3 Trainieren und Speichern des Modells*

Das Machine Learning-Modell stellt ein eigenständiges Python Modul dar. Es ist daher möglich, das Modell auszutauschen oder zu aktualisieren. Basierend auf dem Vergleich verschiedener Modelle[10] wird das Effizienteste ausgewählt.

Das Modell wird nicht bei jedem Start der Anwendung neu trainiert, da das Training zeit- und ressourcenaufwendig ist. Stattdessen wird das Modell einmal trainiert und in einer Datei abgespeichert. Diese Datei wird bei der Klassifizierung geladen und stellt das trainierte Modell zur Verfügung.

##### Speichern und Laden in Python

In Python gibt es mehrere Möglichkeiten ein Modell zu speichern.

##### pickle

Die pickle Library stellt die Objekt-Serialisierung für Python zur Verfügung. Dabei wird eine Python Objekthierarchie in einen Bytestream umgewandelt und somit serialisiert. Dieser Vorgang nennt sich „pickling". Der Umkehrvorgang, also die Deserialisierung des Bytestreams in eine Objekthierarchie wird „unpickling" genannt. [43]

##### joblib

Die in der Scikit-learn Dokumentation empfohlene Variante verwendet joblib. [44]

Joblib ist ein Ersatz für pickle und eignet sich besonders gut zum Speichern von Objekten, die große Datenmengen enthalten. Daher ist es für ein Machine Learning-Modell grundsätzlich besser geeignet. [45]

Grundsätzlich sollten nur Daten de-serialisiert werden, deren Ursprung bekannt ist. Da pickle nicht abgesichert ist – und joblib, das auf pickle basiert, ähnliche Mängel aufweist – ist bei der Verwendung Vorsicht geboten. Unachtsamer Umgang könnte dazu führen, dass bösartiger Code ausgeführt wird.

Werden beim Speichern und Laden des Modells bzw. Objekts verschiedene Sckit-learn-Versionen verwendet, könnte dies zu unerwarteten Ergebnissen oder Fehlern führen. Um ein korrektes Laden sicherzustellen, ist es daher ratsam Metadaten über das Objekt zu speichern. Dazu zählen – im Falle eines Machine Learning-Modells – beispielsweise die verwendeten Trainingsdaten, die Scikit-learn-

[10] 9.3 Training und Vergleich verschiedener Machine Learning-Modelle

Version inkl. Abhängigkeiten, der Python Source Code, mit dem das Modell generiert wurde und der Cross Validation Score aus dem Training. Dadurch kann überprüft werden, ob sich der Cross Validation Score durch das Speichern und Laden verändert hat, oder sich weiterhin in der erwarteten Spanne befindet.[44]

Da sich alle Python Module der Anwendung im gleichen Projekt befinden, gibt es keine Versionskonflikte und es werden nur selbst serialisierte, sichere Daten deserialisiert.

#### Umsetzung in der Anwendung

Im Anwendungsordner befindet sich der Unterordner data, in dem die Trainingsdaten für das Modell gespeichert sind. Wird das Modell oder die Daten ausgetauscht, muss das Modell einmal neu trainiert werden, sodass eine aktuelle Version gespeichert werden kann.

```
dump(clf, 'classifier.joblib')
```

*Listing 9: Speichern des Modells*

Die Datei classifier.joblib wird im Verzeichnis abgelegt und kann somit von anderen Python Modulen geladen und verwendet werden.

### *9.4.1.4 Klassifizieren von Inputdaten*

Bei Start der Anwendung wird ein Executable ausgeführt, welches die laufende Aufnahme von MIDI-Daten ermöglicht. Daraufhin werden in regelmäßigen Abständen die Features der neusten MIDI-Datei eingelesen, welche mithilfe des gespeicherten Modells klassifiziert werden.

#### Aufnahme von Daten

Um eine selbstständige, automatische Aufnahme und Speicherung von MIDI-Dateien zu ermöglichen, kommt die brainstorm.exe zum Einsatz. Das Executable ist Teil eines MIDI-Utility Pakets das von David G. Slomin entwickelt wurde.

Brainstorm ist nur eine von vielen Funktionen und wurde laut der Dokumentation als eine Art „Diktiergerät für MIDI" entwickelt. Die Audiofiles werden automatisch aufgenommen und mit einem Zeitstempel abgespeichert. [13]

#### Extrahieren von Features

Wie bereits in der Data Exploration Phase kommt jSymbolic zum Einsatz. Die Extraktion erfolgt automatisch durch ein PowerShell Skript, welches innerhalb des Python Moduls zur Klassifizierung aufgerufen wird.

Die Features werden in einer CSV-Datei gespeichert und zur Klassifizierung verwendet.

### Umsetzung in der Anwendung

Das Modul classify.py enthält alle nötigen Funktionalitäten oder löst diese aus. Dabei gibt es eine Methode, die regelmäßig neu aufgerufen wird und dabei ein aktuelles Ergebnis erzeugt.

Im ersten Schritt wird das Modell geladen:

```
loaded_model = load('classifier.joblib')
```
*Listing 10: Modell wird geladen*

Daraufhin wird das neueste MIDI-File der Brainstorm Aufnahmen ermittelt:

```
max_file = max(files, key=os.path.getctime)
```
*Listing 11: Code zum Ermitteln der neuesten MIDI-Datei*

Danach werden die Features für das MIDI-File extrahiert. Um mehrere Operationen zusammenzufassen, wird innerhalb des Python Codes ein PowerShell Skript aufgerufen.

```
os.system('cmd /c powershell .\extraction.ps1 ' + filename)
```
*Listing 12: Code zum Aufruf eines extraction.ps1*

Das genannte extraction.ps1 nimmt als Parameter den Dateinamen der gewünschten MIDI-Datei entgegen und extrahiert deren Features in ein CSV-File.

```
java -Xmx6g -jar jSymbolic2.jar -csv ../$filename ../feature_values.xml
../feature_descriptions.xml
```
*Listing 13: Extraktion von Features via PowerShell mit jSymbolic*

Die Features aus dieser Datei werden in einen pandas-Dataframe gespeichert und vom Modell zur Prediction verwendet.

```
y_predicted = loaded_model.predict(features)
```
*Listing 14: Code zum Treffen einer Vorhersage aus dem Modell*

Das vorhergesagte Label ist das Ergebnis der Klassifizierung und wird zur Darstellung weitergeleitet.

### 9.4.2 Version 1.0

Da der erste Prototyp sehr fehleranfällig in seiner Handhabung und nur schwer portierbar gewesen ist, ist dieser vollständig überarbeitet worden.

#### *9.4.2.1 Verbesserung am Aufbau*

Ein erster Schritt ist es gewesen, die einzelnen Aufgaben klarer zu strukturieren. In folgende Aufgaben wird dabei unterteilt:

1. Webapplikation
2. MIDI-Recording
3. Feature Extraction
4. Classification

In Abb. 24 ist zu sehen, wie diese interagieren und welche Daten dabei auch generiert werden. Die größte Änderung ist dabei gewesen, dass die Extraktion der Features aus dem MIDI-Input und die Klassifikation des MIDI-Inputs vollständig voneinander getrennt wurden.

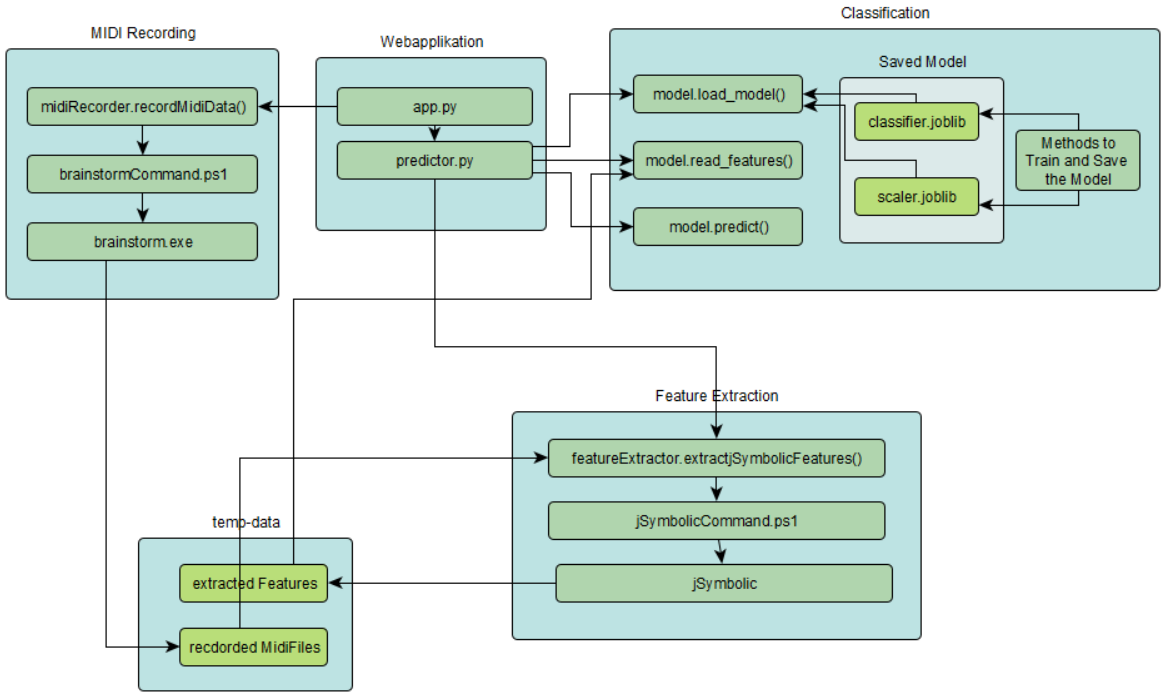



*Abb. 24: Aufbau der CEPP-Anwendung*

#### *9.4.2.2 Verbesserung des Ablaufes*

Beim Prototyp ist es nötig gewesen, die Aufnahme der MIDI-Dateien und die Webapplikation zur Visualisierung und Klassifizierung der aufgenommen MIDI-Dateien als zwei verschiedene Anwendungen einzeln zu starten. In Version 1.0 muss man nur noch die Webanwendung starten, da diese im Hintergrund auch die Aufnahme startet.

#### *9.4.2.3 MIDI-Recording als Hintergrundjob*

Wie schon beschrieben, ist es beim Prototyp der Applikation nötig, brainstorm.exe, welche für das Aufnehmen des MIDI-Inputs zuständig ist, und die Webapplikation, welche den MIDI-Input verarbeitet, eine Emotion vorhersagt und diese zur Anzeige bringt, jeweils einzeln zu starten.

Bei der Version 1.0 muss nun nur noch die Webapplikation gestartet werden, welche einen Hintergrundjob startet, welcher wiederum in einer Methode ausgelagert brainstorm.exe startet. Realisiert wird das mit Hilfe eines Flask APScheduler, einer Flask Erweiterung, die Support für den APScheduler (Adanced Python Scheduler) zu Flask hinzufügt. APScheduler steht für Advanced Python Scheduler und ist eine Python Library, mit der man Pyhton Code geplant später ausführen kann, sodass er entweder einmal oder periodisch ausgeführt wird. Im Programm wird dafür die Methode `add_job()` des erstellten APScheduler aufgerufen, welcher die Funktion, die brainstorm.exe aufruft, und ihre dazugehörigen Funktionsargumente übergeben werden (siehe Listing 15).

```
import os
from flask import Flask
from flask_apscheduler import APScheduler
import midiRecorder

CEPP_APP_DIR = os.environ['CEPP']
MIDI_OUTPUTDIR = CEPP_APP_DIR + "\\temp_data\\midis"

app = Flask(__name__)

scheduler = APScheduler()
# run background midi-recording job
scheduler.add_job(func=midiRecorder.recordMidiData, args=(MIDI_OUTPUTDIR, 0,
"cepp", 2), id='midi')
scheduler.init_app(app)
scheduler.start()
```

*Listing 15: Erstellung eines APSchedulers und hinzufügen eines Jobs*

#### *9.4.2.4 Verbesserung der UI und UX der Anwendung*

Verbesserung hinsichtlich der UI (User Interface) ist das Neudesign der Seite, wie in den Abb. 25 und Abb. 26 zu sehen ist. Um die UX (User Experience) zu verbessern, wurde ein Timer eingebaut, um anzuzeigen, wenn die Seite das nächste Mal neu geladen und somit der Prozess der Emotionsvorhersage erneut ausgelöst wird. Dies sollte dem Benutzer in dem Sinn helfen, damit er weiß, wann er mit seiner Aufnahme stoppen kann.

##### Timer

Der Timer selbst ist mithilfe von JavaScript realisiert worden, genau ist dafür die JavaScript-Funktion `setInterval(function, milliseconds)` verwendet worden. Mit `function` kann man der `setInterval()` die Funktion, die in einem bestimmten Intervall ausgeführt werden soll, in diesem Fall eine Funktion, die den Timer immer wieder updatet, übergeben. `milliseconds` gibt das Intervall in Millisekunden an, indem die übergebene Funktion ausgeführt wird. Da der Timer selbst sekündlich aktualisiert werden soll, wird `milliseconds` auf 1000 gesetzt.

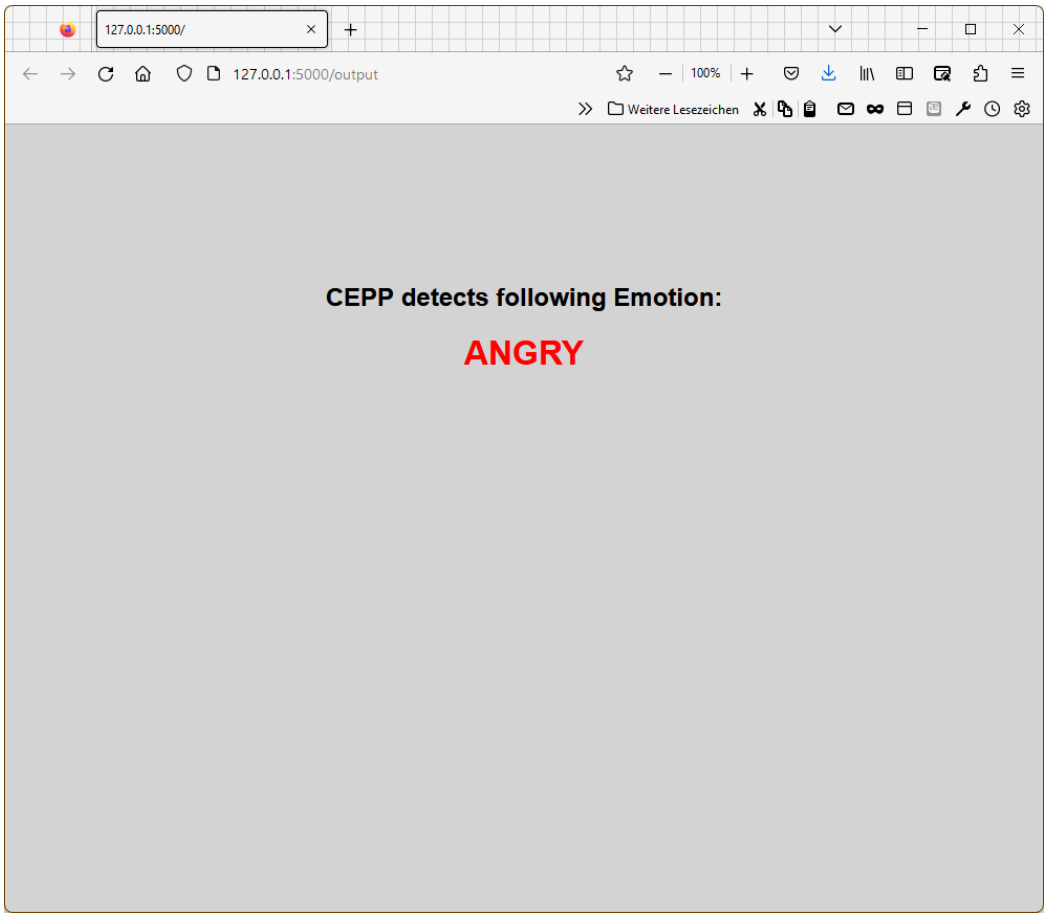



*Abb. 25: Weboberfläche Prototyp*

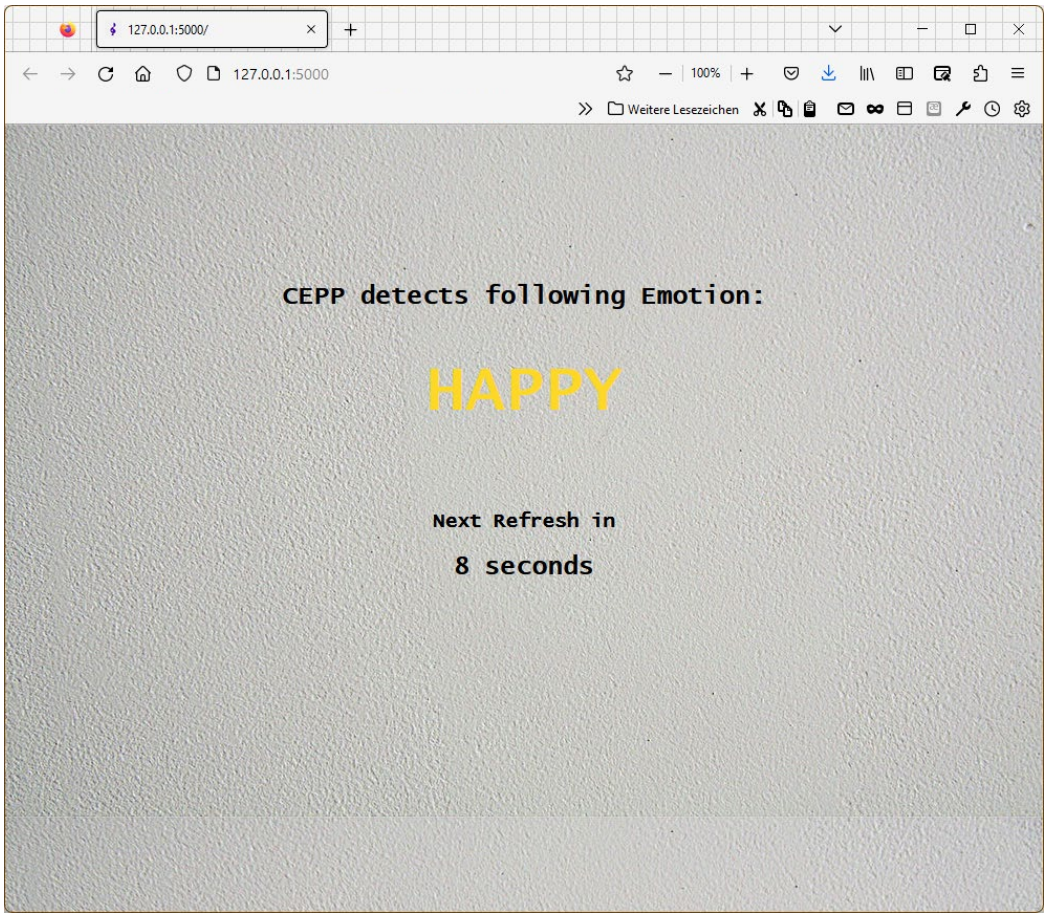



*Abb. 26: Weboberfläche Endanwendung*

# 10 Ergebnis

Das Ergebnis der Diplomarbeit besteht aus drei Teilen: dem CEPP-Datenset, einem Machine Learning-Modell zur Emotion Prediction von MIDI-Input und der CEPP-Applikation.

## 10.1 CEPP-Datenset

Das CEPP-Datenset umfasst 68 MIDI-Files, die jeweils einem Emotionsquadranten zuordenbar (Q1, Q2, Q3, Q4) und am Klavier aufgenommen worden sind. Die MIDI-Files sind in einzelnen Aufnahme-Sessions im Zuge des Projekts aufgenommen und von 15 unterschiedlichen Personen eingespielt worden, wobei unter den Personen erfahrene Klavierspieler und Laien gewesen sind. Mehr dazu ist im Kapitel 9.2 Aufzeichnung von eigenen Daten zu finden.

## 10.2 Machine Learning-Modelle

Wie in Kapitel 9.3 Training und Vergleich verschiedener Machine Learning-Modelle beschrieben, sind verschiedene Machine Learning-Modelle unterschiedlicher Modellklassen trainiert, analysiert und miteinander verglichen worden. Die Aufgabe der Modelle ist es aus MIDI-Dateien ausgewertete Features zu klassifizieren, sprich ihnen eine Emotion bzw. einen Emotionsquadranten zuzuordnen. Diese Modelle sind in Jupyter Notebooks erstellt worden und somit bilden diese Jupyter Notebooks ein zusätzliches Ergebnis der Diplomarbeit.

## 10.3 CEPP-Applikation

Ein weiteres Ergebnis der Diplomarbeit ist die CEPP-Applikation, mit der es möglich ist, den MIDI-Input eines an den Computer angeschlossenen Klaviers aufzunehmen. Für den aufgenommenen Input kann danach die (von der KI vorhergesagte) Emotion live angezeigt werden. Dafür verwendet die CEPP-Applikation eines der trainierten Modelle.

Auf Abb. 27 ist die Oberfläche der CEPP-Applikation nochmals abgebildet.


*Abb. 27: Screenshot der Webanwendung – CEPP detects Angry*

# 11 Resümee

Bisherige Erfahrung aus dem Projektunterricht hat dazu beigetragen, dass die Planungsphase selbstständig und sinnvoll durchgeführt werden konnte. Meetings mit dem Auftraggeber, das Erstellen eines Zeitplans oder die Definition von Zielen im Projekt stellten so keine Hindernisse dar.

Die Kommunikation mit dem Auftraggeber fand während des Praktikums größtenteils in englischer Sprache statt. Da das Institute of Computational Perception viele internationale Forscher beschäftigt, konnte so ein reger und lehrreicher Austausch stattfinden. In zahlreichen Gesprächen und Präsentationen kamen die erlernten Englischkenntnisse zum Einsatz und konnten dadurch noch zusätzlich ausgebaut werden. Der rege Kontakt mit Experten war für den theoretischen Teil des Projektes, mit dem viel Recherche verbunden war, äußerst förderlich.

Im Rahmen der Durchführung wurden zahlreiche neue Technologien kennengelernt und erarbeitet. Durch Unterstützung von Seiten der Projektbetreuung und des Auftraggebers sowie intensivem Selbststudium und gründlicher Recherche konnte so viel neues Wissen erlangt werden. Hauptsächlich wurden folgende Themen neu erlernt:

- Extraction von MIDI-Features mit jSymbolic und Music21
- Verschiedenste Machine Learning-Modelle
- Python und Jupyter

Besonders in der Anfangsphase wurde sehr viel Datenanalyse betrieben. Das Extrahieren, Vergleichen, Bewerten und letztendlich Verstehen der zahlreichen MIDI-Features bietet die Grundlage für das gesamte Projekt. Auch die Visualisierung mehrdimensionaler Datensets spielte hier eine wichtige Rolle. Das Arousal-Valence Problem trat ebenfalls bereits zu Beginn des Projekts auf und konnte durch Scatter-Plots besonders anschaulich dargestellt werden.

Die Evaluierung verschiedenster Machine Learning-Modelle stellte eine neue Herausforderung dar. Dabei wurden mehrere verschiedene Arten von Modellen mit unterschiedlichen Konfigurationen verglichen. Um zu verstehen welche Modelle gut funktionieren, warum sie genauere Ergebnisse liefern als andere und wie man sie verbessern kann, wurde auf viel Wissen aus der Data Exploration zurückgegriffen. Basierend auf diesen Auswertungen wurde für die Anwendung das beste Modell ausgewählt.

Die Anwendung selbst wurde in Python umgesetzt, eine Sprache, die ebenfalls für dieses Projekt erlernt werden musste. Die Benutzeroberfläche wurde dabei mit Flask realisiert, während die Logik des Programmes auf mehrere Python Module sowie PowerShell-Skripte aufgeteilt ist.

Ergebnis ist eine einfach zu bedienende Applikation, die in dieser Form neuartig ist. Man benötigt lediglich ein Kabel für den Verbindungsaufbau zu einem MIDI-fähigen Gerät und schon kann der damit generierte Input analysiert werden, nach dem Motto: Plug and Play. Auf dem Gebiet des Music Information Retrieval wird intensiv geforscht, und dieses Projekt ist nicht das einzige, das versucht Emotionen mit einem Computer festzustellen. Allerdings war eine derart kompakte, benutzerfreundliche und dennoch ausreichend präzise Lösung bisher nicht vorhanden.

Natürlich kann nur eine begrenze Präzision erreicht werden, und selbst dann kann das Programm nicht immer richtig liegen. Denn Emotionen sind subjektiv und oft ist es schon für Menschen schwierig, diese richtig einzuordnen. Maschinen sind zur Erkennung von Emotionen daher auf gute Daten angewiesen – sowohl beim Training als auch bei der Analyse. Und nicht jede Emotion kann mit bloßen Zahlen perfekt dargestellt werden.

Zum Abschluss lässt sich daher sagen: Informatik wird definiert als die systematische Verarbeitung von Informationen – als die Wissenschaft elektronischer Datenverarbeitung. Musik erscheint auf den ersten Blick vielleicht als ein ungewöhnliches und wenig technisches Thema für eine Diplomarbeit in diesem Gebiet, vor allem, wenn auch noch Emotionen ins Spiel kommen. Allerdings hat sich im Laufe dieser Arbeit gezeigt, dass gerade Musik unzählige Daten enthält, die nur darauf warten verarbeitet zu werden.

# 12 Planung und Realisierung

Im Zuge des Projektmanagement sind folgende Ergebnisse erstellt worden:

## 12.1 Projektstrukturplan

Für das Projekt wurde in der Projektplanungsphase ein Projektstrukturplan erstellt, welcher in Abb. 28 zu sehen ist.

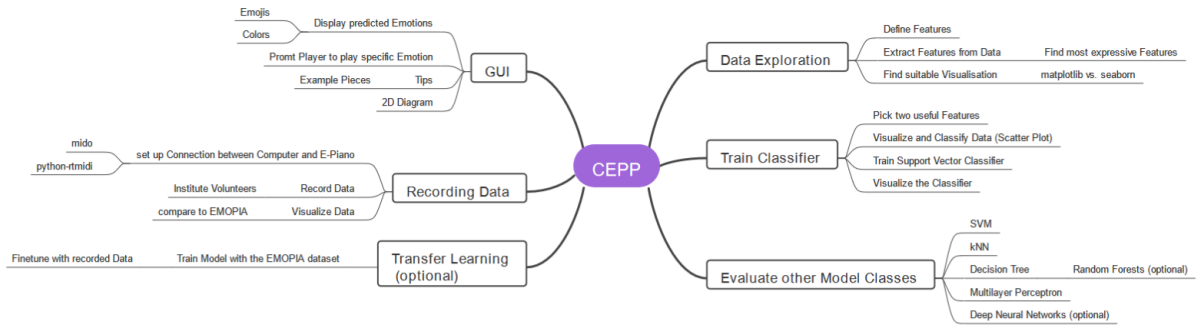



*Abb. 28: Projektstrukturplan in Form einer Mindmap. Gliederung nach Meilensteinen.*

## 12.2 Projektorganisation

Folgende IMV-Matrix stellt die Verantwortlichkeiten der einzelnen Projektmitglieder da:

| | Christina Brandstetter | Katharina Praher |
|---|---|---|
| **Projektmanagement** | M | V |
| **Feature Extraktion** | V | M |
| **Auswahl der Features** | V | I |
| **Aufnahme von Trainingsdaten am Klavier** | V | M |
| **Trainieren und Vergleichen verschiedener Machine Learning-Modelle** | M | V |
| **Erstellung einer Applikation für Demonstrationszwecke** | M | V |

*Tabelle 5: IMV-Matrix: I... Information, M... Mitarbeit, V... Verantwortlich*

## 12.3 Meilensteine und Projektzeitplan

Folgende Meilensteine wurden für die Diplomarbeit festgelegt und eingereicht:

- Datenerhebung und eigene Aufnahme von MIDI-Daten
- Modell mit festgelegten Eigenschaften trainieren
- Andere Modellklassen evaluieren
- Grafische Benutzeroberfläche und Visualisierung

Diese Meilensteine sind an einen Zeitplan gekoppelt. Der Zeitplan wurde mit Microsoft Project erstellt und stellt einen groben Plan dar, um den Überblick im Projekt zu wahren. Der Zeitplan kann im Anhang gefunden werden.

### 12.3.1 Weitere Aufteilung

Beim Projektstrukturplan wurde eine genauer aufgeschlüsselte Variante mit sechs Meilensteinen statt vier gewählt. Damit wurde auf einen Vorschlag des Auftraggebers eingegangen, um die Kommunikation zu erleichtern und schneller Zwischenergebnisse liefern zu können. Diese Meilensteine umfassen:

- Data Exploration
- Aufnahme von Daten
- Trainieren eines Classifiers basierend auf den extrahierten Features
- Evaluieren anderer Modellklassen
- OPTIONAL: Transfer Learning und Finetuning
- Grafisches User Interface

Fünf dieser Meilensteine finden sich beinahe unverändert bzw. leicht angepasst und zusammengefasst in den offiziellen Meilensteinen wieder. Der optionale Meilenstein zum Transfer-Learning ist dabei beim Vergleich der Modellklassen wiederzufinden.

### 12.3.2 Projektverlauf

Im Projektverlauf hat sich der Zeitplan als nützliche Richtlinie erwiesen, auch wenn nicht alle Meilensteine eingehalten werden konnten. Das hängt damit zusammen, dass schlichtweg die Erfahrung für Vorgänge im Forschungsumfeld fehlt und so keine präzise Schätzung getroffen werden konnte. Auch die Planung in Tagen ist nicht akkurat, da bestimmte wichtige Tasks keinen gesamten Arbeitstag in Anspruch nehmen, oder über viele Tage hinweg in kleinen Etappen behandelt werden.

Für die Meilensteine wurden folgende Termine gesetzt:

- 17.07.2022 - Datenerhebung und eigene Aufnahme von MIDI-Daten
- 24.07.2022 - Modell mit festgelegten Eigenschaften trainieren
- 07.08.2022 - Andere Modellklassen evaluieren
- 11.09.2022 - Grafische Benutzeroberfläche und Visualisierung

Die Überlegung hinter dieser Terminierung sah vor, dass das Praktikum an der JKU in den Sommerferien speziell zur Erarbeitung der Teile genutzt werden sollte, die besonders von der Anwesenheit am Institut profitieren konnten. Die Teilbereiche zur Datenerhebung und zum Data Recording sowie der Vergleich und das Training verschiedener Machine Learning-Modelle konnten durch die Expertise des Instituts unterstützt werden. Die ersten drei Meilensteine konnten daher auch eingehalten werden.

Der letzte Meilenstein umfasste die Anwendung an sich und damit vor allem die grafische Darstellung der Daten. Da in diesem Bereich kein besonderes Knowhow am Institut vorhanden war, wurde die Umsetzung bis zum Ende der Sommerferien geplant, um das Projekt innerhalb dieses Zeitraums abschließen zu können.

Der Termin konnte allerdings nicht eingehalten werden, da der Umfang der Applikation grob unterschätzt wurde. Letztendlich handelt es sich dabei nicht um eine bloße grafische Darstellung, sondern ein interaktives System, das Benutzereingaben in nahezu Echtzeit aufnimmt, analysiert und darstellt. Aufgrund dieser Diskrepanz zwischen Schätzung und tatsächlichem technischen Aufwand, konnte die Applikation erst mit Ende Dezember 2022 fertiggestellt werden.

### 12.3.3 Erkenntnisse

Ohne konkrete Erfahrung kann es schwierig sein eine genaue Schätzung vorzunehmen. Komponenten wie externe Teilnehmer, die für das Data-Recording hinzugezogen werden, sind nicht berechenbar und können Zeitverzug herstellen. Insgesamt konnte der Zeitplan daher nicht vollständig eingehalten werden, war aber hilfreich, um To-do-Listen und Wochenpläne zu erstellen. Dadurch konnte der weitere Projektverlauf stets effizient geplant werden und der Zeitplan konnte seinen primären Zweck erfüllen.

## 12.4 Funktionalität

Da die Aufgabenstellung der Diplomarbeit zu einem großen Teil aus der Erstellung einer Künstlichen Intelligenz, die den MIDI-Input eines Klaviers auswertet und basierend darauf versucht die gespielte Emotion zu erkennen, besteht, ist nur ein sehr kleiner Applikationsumfang gefordert. Die Funktionalität der Applikation ist dabei, dass sie den Input eines an den PC angeschlossen Klavier erfasst, diesen an die KI weitergibt und die von der KI vorhergesagte Emotion zur Anzeige bringt. Dieser einzige Anwendungsfall ist zusätzlich in Abb. 29 als UML Use-Case-Diagramm festgehalten worden.

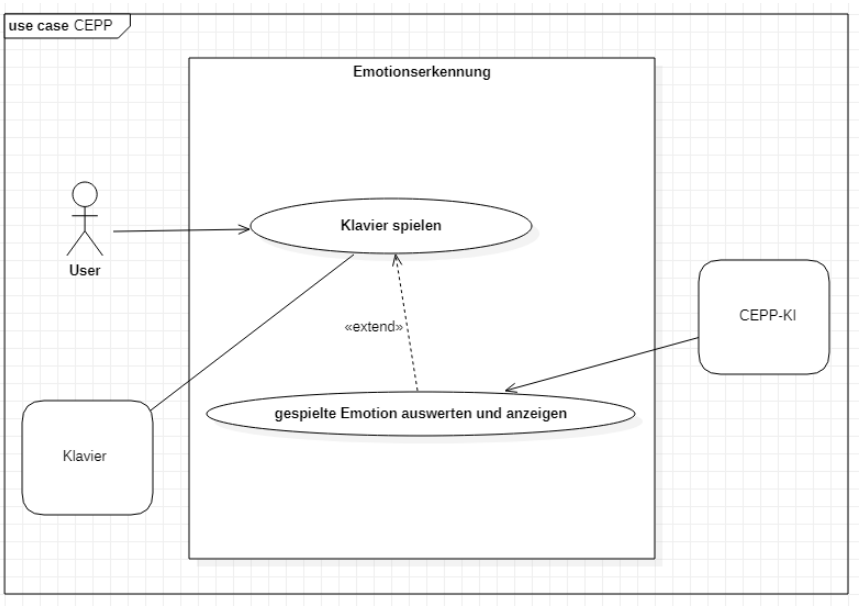



*Abb. 29: Use-Case Diagramm für die CEPP-Anwendung*

### 12.4.1 Entwurf der Funktionalität

Für die Anzeige der Emotionsvorhersage der KI sind zusätzlich zwei Wireframes entworfen worden, wie in Abb. 30 zu sehen ist. Zuerst wird während der Benutzer Klavier spielt und die Applikation die Eingabedaten entgegennimmt ein Ladebildschirm (1) angezeigt und sobald die KI ihre Vorhersage abgeschlossen hat, sollte die prognostizierte Emotion (2) angezeigt werden.

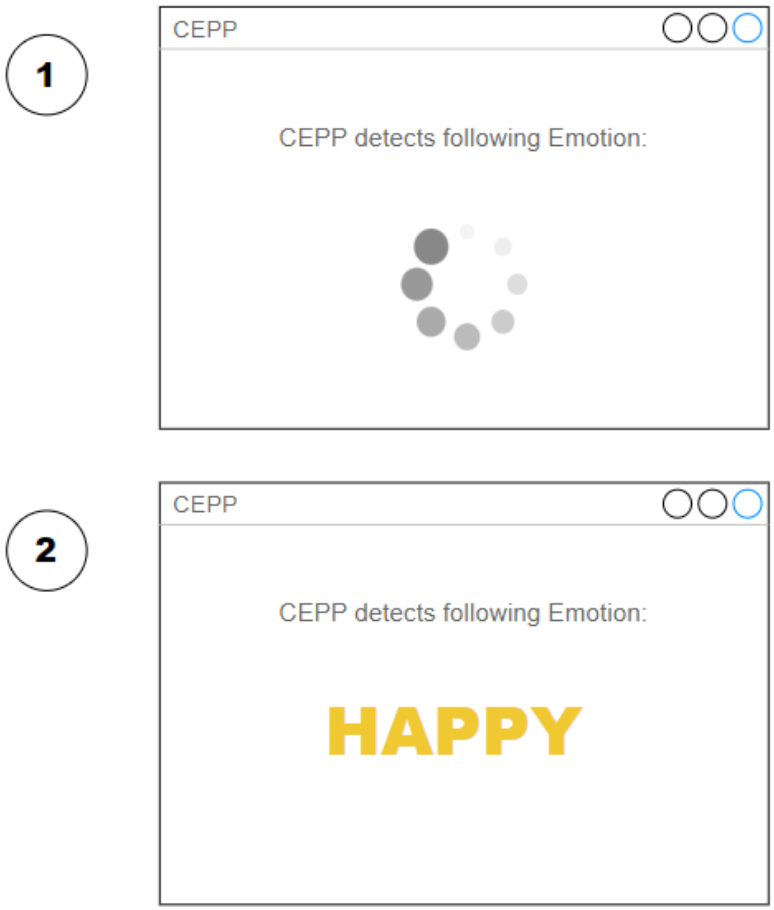



*Abb. 30: Wireframes – CEPP-Anwendung im Entwurf*

# 13 Aufgabenverteilung

Im folgenden Punkt ist festgehalten, wer welche Kapitel der Diplomarbeit verfasst hat. Teilweise wurden Kapitel von mehreren Personen verfasst, da der jeweilige Bereich in die Zuständigkeit mehrerer Teammitglieder fällt.

## 13.1 Christina Brandstetter

- 7 Einleitung
  - 7.1 Motivation
  - 7.2 Zielsetzungen
  - 7.3 Projektinhalt – Überblick
- 8 Theoretische- und fachpraktische Grundlagen
  - 8.1.6 jSymbolic
  - 8.3.2 Matplotlib
  - 8.3.3 Seaborn
  - 8.4.1 Div Slomins MIDI Utilities
  - 8.4.2 MidiEditor
- 9 Implementierung
  - 9.1 Daten Exploration
  - 9.2 Aufzeichnung von eigenen Daten
  - 9.4 CEPP-Anwendung
    - 9.4.1 Prototyp
- 12 Planung und Realisierung
  - 12.3 Meilensteine und Zeitplan
- 14 Literaturverzeichnis
- 15 Abbildungsverzeichnis
- 16 Listings
- 17 Tabellenverzeichnis
- 18 Stichwortverzeichnis
- 19 Anhang

## 13.2 Katharina Praher

- 7 Einleitung
  - 7.4 Projektumfeld
- 8 Theoretische und fachpraktische Grundlagen und Methoden
  - 8.1 Verwendete Technologien
    - 8.1.1 Python
    - 8.1.2 Project Jupyter
    - 8.1.3 Flask
    - 8.1.4 Conda
    - 8.1.5 Git und GitLab
  - 8.2 Verwendete Entwicklungssysteme
    - 8.2.1 PyCharm
    - 8.2.2 DataSpell
  - 8.3 Verwendete Bibliotheken und Plug-Ins
    - 8.3.1 pandas
    - 8.3.2 Matplotlib
    - 8.3.4 Scikit-learn
    - 8.3.5 Music21
    - 8.3.6 NumPy
  - 8.8 Verwendete Hardware
    - 8.8.1 Klavier
- 9 Implementierung
  - 9.3 Training und Vergleich verschiedener Machine Learning-Modelle
  - 9.4 CEPP-Applikation
    - 9.4.2 Version 1.0
- 10 Ergebnis
- 12 Planung und Realisierung
  - 12.2 Projektorganisation
  - 12.4 Funktionalität

# 14 Literaturverzeichnis

[1] G. van Rossum, „Python tutorial“, Amsterdam, Mai 1995.

[2] jupyter, „Architecture“, 2015.

[3] Wikipedia, „Flask (web framework)“, 7. November 2022. https://en.wikipedia.org/wiki/Flask_(web_framework) (zugegriffen 21. März 2023).

[4] S. Chacon und B. Straub, *Pro Git*. 2014. doi: 10.1007/978-1-4842-0076-6.

[5] Wikipedia, „GitLab“, 7. Februar 2023. https://de.wikipedia.org/wiki/GitLab (zugegriffen 21. März 2023).

[6] T. pandas development team, „pandas-dev/pandas: Pandas“. Zenodo, Februar 2020. doi: 10.5281/zenodo.3509134.

[7] J. D. Hunter, „Matplotlib: A 2D graphics environment“, *Comput Sci Eng*, Bd. 9, Nr. 3, S. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[8] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[9] M. Waskom, „seaborn: statistical data visualization“, *J Open Source Softw*, Bd. 6, Nr. 60, S. 3021, Apr. 2021, doi: 10.21105/joss.03021.

[10] F. Pedregosa *u. a.*, „Scikit-learn: Machine Learning in Python“, *Journal of Machine Learning Research*, Bd. 12, S. 2825–2830, 2011.

[11] M. S. Cuthbert und C. Ariza, „Music21: A toolkit for computer-aided musicology and symbolic music data“, in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, 2010.

[12] C. R. Harris *u. a.*, „Array programming with NumPy“, *Nature*, Bd. 585, Nr. 7825, S. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[13] D. G. Slomin, „Div’s MIDI Utilities“, 6. Oktober 2022.

[14] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, und Y.-H. Yang, „EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation“, Aug. 2021, [Online]. Verfügbar unter: http://arxiv.org/abs/2108.01374

[15] A. Birkett, „Valence, Arousal, and How To Kindle an Emotional Fire", *CXL*, 14. Juni 2021.

[16] N. Du *u. a.*, „Examining the effects of emotional valence and arousal on takeover performance in conditionally automated driving", *Transp Res Part C Emerg Technol*, Bd. 112, S. 78–87, März 2020, doi: 10.1016/J.TRC.2020.01.006.

[17] C. Mckay, „Automatic Music Classification with jMIR", 2010.

[18] M. S. Cuthbert, C. Ariza, und L. Friedland, „FEATURE EXTRACTION AND MACHINE LEARNING ON SYMBOLIC MUSIC USING THE music21 TOOLKIT Music and Theater Arts", 2011. [Online]. Verfügbar unter: http://mit.edu/music21.

[19] J. W. Emerson *u. a.*, „The Generalized Pairs Plot", 2011.

[20] C. Mckay, „Automatic Music Classification with jMIR", 2010.

[21] C. L. Krumhansl, „Cognitive Foundations of Musical Pitch (Oxford Psychology Series)".

[22] E. Parada-Cabaleiro, A. Batliner, M. Schmitt, M. Schedl, G. Costantini, und B. Schuller, „Perception and classification of emotions in nonsense speech: Humans versus machines", *PLoS One*, Bd. 18, Nr. 1, S. e0281079, Jän. 2023, doi: 10.1371/JOURNAL.PONE.0281079.

[23] T. Hastie, R. Tibshirani, G. James, und D. Witten, „An introduction to statistical learning (2nd ed.)", *Springer texts*, Bd. 102, 2021.

[24] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, und Y.-H. Yang, „EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation", Aug. 2021, [Online]. Verfügbar unter: http://arxiv.org/abs/2108.01374

[25] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[26] sklearn, „StratifiedShuffleSplit", 2023. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html#sklearn.model_selection.StratifiedShuffleSplit (zugegriffen 21. März 2023).

[27] „L09 Generalization".

[28] Wikipedia, „Bias–variance tradeoff", 21. Februar 2023. https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff (zugegriffen 21. März 2023).

[29] sklearn, „Preprocessing data", 2023. https://scikit-learn.org/stable/modules/preprocessing.html (zugegriffen 21. März 2023).

[30] sklearn, „StandardScaler", 2023. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn-preprocessing-standardscaler (zugegriffen 21. März 2023).

[31] A. Tharwat, „Classification assessment methods", *Applied Computing and Informatics*, Bd. 17, Nr. 1, 2018, doi: 10.1016/j.aci.2018.08.003.

[32] sklearn, „3.3.2.2. Accuracy score", 2023. https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score (zugegriffen 21. März 2023).

[33] C. J. Van Rijsbergen, „Information Retrieval, 2nd edition", *Butterworths*, 1979.

[34] sklearn, „recall_score", 2023. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score (zugegriffen 21. März 2023).

[35] T. M. Cover und P. E. Hart, „Nearest Neighbor Pattern Classification", *IEEE Trans Inf Theory*, Bd. 13, Nr. 1, 1967, doi: 10.1109/TIT.1967.1053964.

[36] sklearn, „KNeighborsClassifier", 2023. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier (zugegriffen 21. März 2023).

[37] C. Cortes und V. Vapnik, „Support-Vector Networks", *Mach Learn*, Bd. 20, Nr. 3, 1995, doi: 10.1023/A:1022627411411.

[38] B. E. Boser, I. M. Guyon, und V. N. Vapnik, „Training algorithm for optimal margin classifiers", in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 1992. doi: 10.1145/130385.130401.

[39] sklearn, „SVC“, 2023. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC (zugegriffen 21. März 2023).

[40] M. Bishop, „detworks for 1 4 Recognition“.

[41] L. Breiman und A. Cutler, „Random forests — Classification description: Random forests“, *https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm*, 2007.

[42] B. E. Boser, I. M. Guyon, und V. N. Vapnik, „Training algorithm for optimal margin classifiers“, in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 1992, S. 1–1. doi: 10.1145/130385.130401.

[43] „pickle - Python object serialization“, *Python documentation*, 14. Jänner 2023.

[44] scikit-learn developers, „Model Persistance“, 2022.

[45] Joblib developers, „Persistence“, *joblib documentation*, 2022.

# 15 Abbildungsverzeichnis

# 16 Listings

# 17 Tabellenverzeichnis

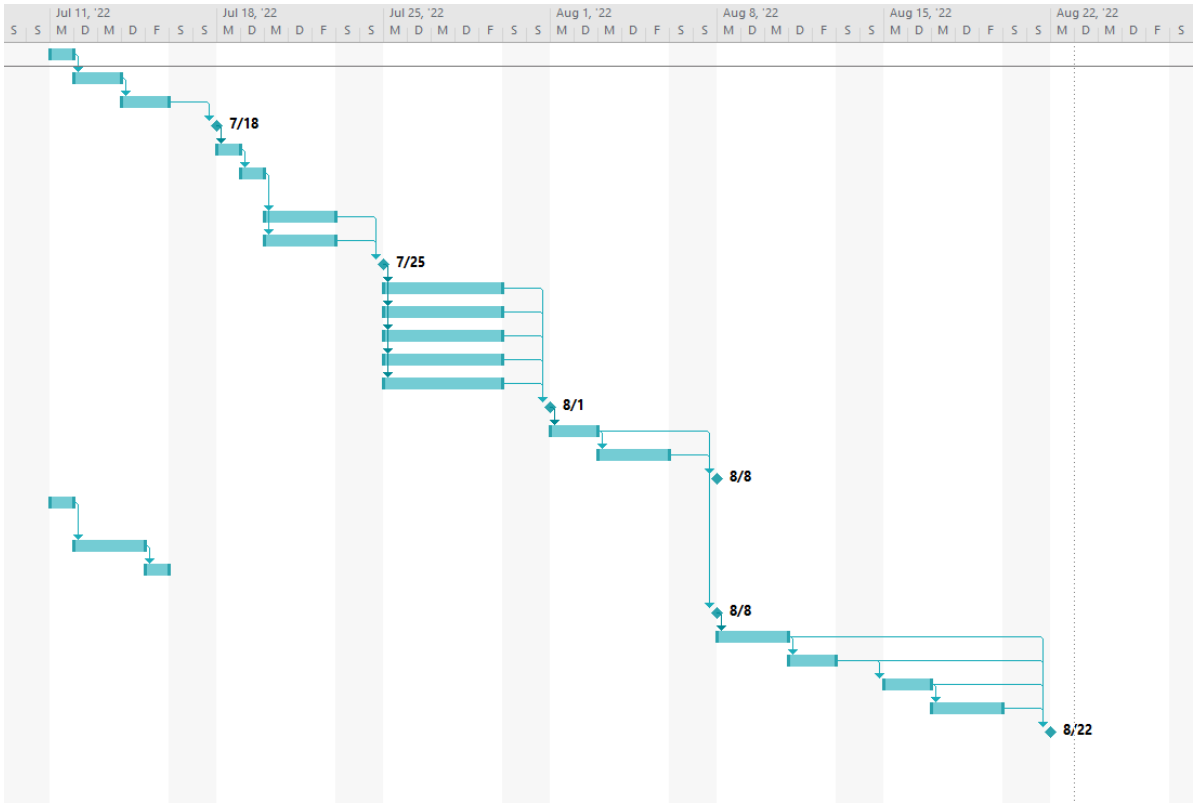
# 18 Stichwortverzeichnis

# 19 Anhang

Im Anhang sind weitere unterstützende Dokumente enthalten.

## 19.1 Zeitplan

Darstellung des Zeitplans auf einer Zeitachse:



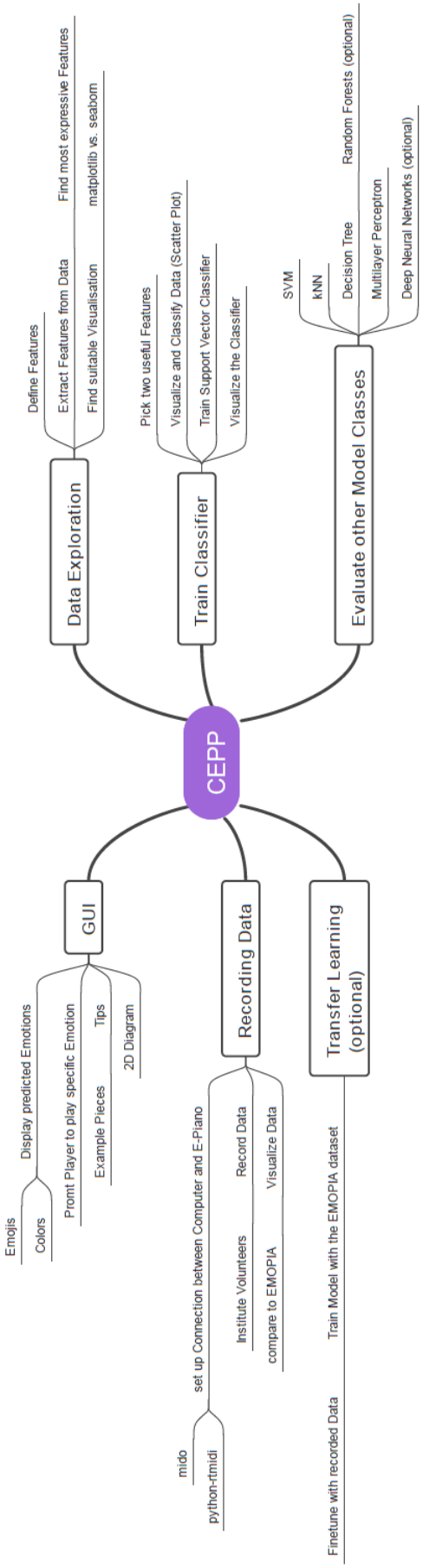

Manueller Vorgang

Meilenstein

Die einzelnen Vorgänge sind in der nachfolgenden Tabelle genauer beschrieben. Wie bereits im Kapitel zum Projektverlauf angemerkt konnte dieser Zeitplan nicht vollständig eingehalten werden, diente allerdings als Grundlage für die Planung aller anstehenden Tasks.

| Nr. | Vorg | Vorgangsname | Dauer | Anfang | Ende | Vorgänger |
|---|---|---|---|---|---|---|
| 1 | | Features festlegen | 1 Tag | Mon 7/11/22 | Mon 7/11/22 | |
| 2 | | Features aus Daten auslesen | 2 Tage | Die 7/12/22 | Mit 7/13/22 | 1 |
| 3 | | passende Visualisierung finden | 2 Tage | Don 7/14/22 | Fre 7/15/22 | 2 |
| 4 | | Data Exploration | 0 Tage | Mon 7/18/22 | Mon 7/18/22 | 3 |
| 5 | | Zwei Features auswählen | 1 Tag | Mon 7/18/22 | Mon 7/18/22 | 4 |
| 6 | | Daten anhand ausgewählter Features visualisieren | 1 Tag | Die 7/19/22 | Die 7/19/22 | 5 |
| 7 | | Support Vector Classifier trainieren | 3 Tage | Mit 7/20/22 | Fre 7/22/22 | 6 |
| 8 | | Classifier visualisieren | 3 Tage | Mit 7/20/22 | Fre 7/22/22 | 6 |
| 9 | | Train Classifier | 0 Tage | Mon 7/25/22 | Mon 7/25/22 | 7,8 |
| 10 | | SVM | 5 Tage | Mon 7/25/22 | Fre 7/29/22 | 9 |
| 11 | | kNN | 5 Tage | Mon 7/25/22 | Fre 7/29/22 | 9 |
| 12 | | Decision Tree | 5 Tage | Mon 7/25/22 | Fre 7/29/22 | 9 |
| 13 | | Multilayer Perceptron | 5 Tage | Mon 7/25/22 | Fre 7/29/22 | 9 |
| 14 | | Deep Neural Networks | 5 Tage | Mon 7/25/22 | Fre 7/29/22 | 9 |
| 15 | | Evaluate other Model Classes | 0 Tage | Mon 8/1/22 | Mon 8/1/22 | 10,11,12, 13,14 |
| 16 | | Mit EMOPIA Daten trainieren | 2 Tage | Mon 8/1/22 | Die 8/2/22 | 15 |
| 17 | | | 3 Tage | Mit 8/3/22 | Fre 8/5/22 | 16 |
| 18 | | Transfer Learning | 0 Tage | Mon 8/8/22 | Mon 8/8/22 | 16,17 |
| 19 | | Verbindung zwischen Computer und E-Piano herstellen | 1 Tag | Mon 7/11/22 | Mon 7/11/22 | |
| 20 | | Daten aufnehmen (Freiwillige) | 3 Tage | Die 7/12/22 | Don 7/14/22 | 19 |
| 21 | | Daten visualisieren (EMOPIA Vergleich) | 1 Tag | Fre 7/15/22 | Fre 7/15/22 | 20 |
| 22 | | Recording Data | 0 Tage | Mon 8/8/22 | Mon 8/8/22 | 17 |
| 23 | | Festgestellte Emotionen anzeigen | 3 Tage | Mon 8/8/22 | Mit 8/10/22 | 22 |
| 24 | | Herausfordern des Spielers | 2 Tage | Don 8/11/22 | Fre 8/12/22 | 23 |
| 25 | | Tipps anzeigen | 2 Tage | Mon 8/15/22 | Die 8/16/22 | 24 |
| 26 | | 2D Valence-Arousal Diagramm | 3 Tage | Mit 8/17/22 | Fre 8/19/22 | 25 |
| 27 | | GUI | 0 Tage | Mon 8/22/22 | Mon 8/22/22 | 23,24,25, 26 |

## 19.2 PSP


CEPP
Data Exploration
Define Features
Extract Features from Data
Find most expressive Features
Find suitable Visualisation
matplotlib vs. seaborn
Train Classifier
Pick two useful Features
Visualize and Classify Data (Scatter Plot)
Train Support Vector Classifier
Visualize the Classifier
Evaluate other Model Classes
SVM
kNN
Decision Tree
Random Forests (optional)
Multilayer Perceptron
Deep Neural Networks (optional)
GUI
Display predicted Emotions
Emojis
Colors
Promt Player to play specific Emotion
Example Pieces
Tips
2D Diagram
Recording Data
set up Connection between Computer and E-Piano
mido
python-rtmidi
Institute Volunteers
Record Data
compare to EMOPIA
Visualize Data
Transfer Learning (optional)
Train Model with the EMOPIA dataset
Finetune with recorded Data

## 19.3 Diplomarbeitsplakat