**DIPLOMARBEIT**

DermaAI: Intelligente Hautanalyse

Gesamtprojekt

Entwicklung von KI-Modellen zur Detektion und Klassifikation pigmentierter Hautläsionen und deren Integration in eine Mobile App

Diplomarbeitsnummer  
5AHINF-24/25-DA11

**Datenbank, Verwaltung der medizinischen Daten**

Jonas Maier 5AHINF Betreuer: Dipl.-Ing. Dr.

Gerhard Gaube

**Mobile Integration, Frontend**

Jonas Bogensberger 5AHINF Betreuer: MSC

Michael Prader

**KI-Modelle in Python, API und Appanbindung**

Daniel Jessner 5AHINF Betreuer: Dipl.-Ing. Dr.

Gerhard Gaube

Schuljahr 2024/25

Abgabevermerk:

Datum: TT.MM.JJJJ übernommen von:

# Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht haben.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, am TT.MM.JJJJ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Daniel Jessner

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jonas Maier

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jonas Bogensberger

# Allgemeines & Projektübersicht

*Kapitel verfasst von Daniel Jessner*

## Projektbeschreibung

Das Projekt „**DermaAI**“ stellt ein Abschlussprojekt für ein Projektteam der 5. AHINF an der HTL Saalfelden dar und hat als Ziel, ein System zu entwickeln, welches es einem Benutzer ermöglicht über sein Mobilgerät Bilder von Hautläsionen aufzunehmen und diese mithilfe von künstlicher Intelligenz auswerten zu lassen. Die Anwendung soll also dabei helfen eine erste Entscheidungsgrundlage zu bieten und eventuell bei Diagnosen helfen. Dazu eignet sich eine Dreiteilung in die Bereiche **Frontend** (Mobile Anwendung), **Backend** (Datenbank) und **Gateway** (KI-Komponente, Vermittler).

Wichtige Eckdaten:

* **Ausschreibung**

Das Projekt an sich wurde am **09.07.2024** von der Fachhochschule Joanneum ausgeschrieben und über die Lehrperson Gerhard Gaube dem Projektteam vorgestellt.

* **Projektstart**

Offizieller Projektstart war der **08.11.2024**. Dabei wurden sowohl eine erste Grundstruktur der Anwendung erstellt als auch das der Version Control zugrundeliegende Github-Repository eingerichtet. Projektplanung und Absprachen über Arbeitsteilung waren zu diesem Zeitpunkt bereits vollendet.

* **Projektende**

Das Projektende setzt sich grundlegend aus mehreren Zeitpunkten zusammen:

* + **.05.2025** Öffentliche Präsentation der Diplomarbeit
  + **21.05.2025** Schulinterne Verteidigung der Diplomarbeit
  + **.05.2025** Abgabe der gebundenen Diplomarbeit
  + **.05.2025** Abgabe der Diplomarbeitsvideos

Zum Zeitpunkt der Abgabe der Diplomarbeit besteht eine vollfunktionsfähige und dokumentierte erste Version der Anwendung, welche, so wie sie ist, vom Endnutzer genutzt werden kann.

## Projektteam und Schnittstellen

|  |  |  |  |
| --- | --- | --- | --- |
| **Rolle(n)** | **Name** | **Telefon** | **E-Mail** |
| Entwickler –  Frontend | Jonas Bogensberger | +43 676 3617230 | [jonas.bogensberger@htl-saalfelden.at](mailto:jonas.bogensberger@htl-saalfelden.at) |
| Entwickler –  Backend | Jonas Maier | +43 664 75087002 | [jonas.maier@htl-saalfelden.at](mailto:jonas.maier@htl-saalfelden.at) |
| Entwickler –  KI-Komponente | Daniel Jessner | +43 650 6349636 | [daniel.jessner@htl-saalfelden.at](mailto:daniel.jessner@htl-saalfelden.at) |

# Funktionale Anforderungen

*Kapitel verfasst von Daniel Jessner*

In diesem Kapitel werden die funktionalen Anforderungen des Projekts detailliert beschrieben. Dazu gehören die verschiedenen Use Cases sowie die spezifischen Anwendungsmöglichkeiten, die sowohl den Endnutzern als auch den Administratoren zur Verfügung stehen. Die dargestellten Anforderungen definieren die wesentlichen Funktionen des Systems und legen fest, welche Interaktionen zwischen den Nutzern und der Anwendung möglich sind.

Ein besonderer Fokus liegt dabei auf der Nutzerfreundlichkeit, der Effizienz der Systemprozesse sowie der klaren Abgrenzung der Berechtigungen zwischen den verschiedenen Nutzergruppen. Während Endnutzer beispielsweise bestimmte Kernfunktionen zur Nutzung des Systems erhalten, verfügen Administratoren über erweiterte Rechte zur Konfiguration, Verwaltung und Überwachung der Anwendung.

Die in diesem Kapitel beschriebenen funktionalen Anforderungen dienen als Grundlage für die Entwicklung und Implementierung des Systems und gewährleisten, dass alle notwendigen Features und Anwendungsfälle in der finalen Lösung berücksichtigt werden.

## Use Cases

Die folgenden Use Cases, also Interaktionsmöglichkeiten mit dem System, werden durch die Implementierung der Software abgedeckt. Sie beschreiben die verschiedenen Szenarien, in denen sowohl Anwender als auch Administratoren mit dem System arbeiten und bestimmte Funktionen ausführen können.

### Anwender

Endnutzer des Systems haben Zugriff auf eine Reihe von Funktionen, die ihnen die Nutzung der Software ermöglichen. Diese Use Cases beschreiben typische Interaktionen, die Anwender im Rahmen ihrer Nutzung durchführen können.

#### Registrierung eines neuen Benutzers

**Akteure:**  
Benutzer (Neukunde), Anwendung (Registrierungssystem), Server (Datenbank für Nutzerdaten)

**Auslöser/Trigger-Event:**  
Der Benutzer startet die Anwendung und möchte sich mit Nutzernamen registrieren.

**Kurzbeschreibung:**  
Ein neuer Benutzer legt ein Konto an, um Zugriff auf die Funktionen der Anwendung zu erhalten.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer öffnet die Anwendung und wählt „Registrieren“.
2. Der Benutzer gibt persönliche Daten (E-Mail, Passwort) ein.
3. Die Anwendung überprüft die Eingaben auf Korrektheit und Vollständigkeit.
4. Die Registrierungsdaten werden an den Server gesendet und dort gespeichert.
5. Der Benutzer erhält eine Bestätigung über die erfolgreiche Registrierung.

**Beschreibung alternativer Schritte:**

* Falls die E-Mail bereits existiert, erhält der Benutzer eine Fehlermeldung und muss eine andere wählen.
* Falls das Passwort nicht den Anforderungen entspricht, wird eine entsprechende Meldung angezeigt.

**Vor- und Nachbedingungen:**  
Vorbedingung: Der Benutzer hat keinen bestehenden Account.  
Nachbedingung: Der Benutzer kann sich mit seinen Zugangsdaten anmelden.

**Systemgrenzen:**  
Anwendung, Server, Gateway, Internetverbindung

#### Anmeldung eines bestehenden Benutzers

**Akteure:**  
Benutzer (bestehender Kunde), Anwendung (Login-System), Server (Datenbank für Nutzerdaten)

**Auslöser/Trigger-Event:**  
Der Benutzer startet die Anwendung und möchte sich anmelden.

**Kurzbeschreibung:**  
Ein registrierter Benutzer gibt seine Zugangsdaten ein, um Zugriff auf sein Konto und die Analysefunktionen zu erhalten.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer öffnet die Anwendung und wählt „Anmelden“.
2. Der Benutzer gibt seine E-Mail-Adresse und sein Passwort ein.
3. Die Anwendung sendet die Daten an den Server zur Überprüfung.
4. Falls die Anmeldedaten korrekt sind, wird der Benutzer eingeloggt und auf das Dashboard weitergeleitet.

**Beschreibung alternativer Schritte:**

* Falls die E-Mail oder das Passwort falsch ist, wird eine Fehlermeldung ausgegeben.
* Falls der Benutzer sein Passwort vergessen hat, kann er eine „Passwort zurücksetzen“-Funktion nutzen.

**Vor- und Nachbedingungen:**  
Vorbedingung: Der Benutzer muss ein bestehendes Konto besitzen.  
Nachbedingung: Der Benutzer ist erfolgreich eingeloggt und kann auf seine Daten zugreifen.

**Systemgrenzen:**  
Anwendung, Server, Gateway, Internetverbindung

#### Einblick in vergangene Analysen

**Akteure:**  
Benutzer, Anwendung, Server (Datenbank mit analysierten Bildern)

**Auslöser/Trigger-Event:**  
Der Benutzer möchte eine Übersicht über seine bisher analysierten Hautläsionen einsehen.

**Kurzbeschreibung:**  
Der Benutzer kann eine Liste aller von der KI analysierten Bilder abrufen und Details zu den Analysen einsehen.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer öffnet die Anwendung und navigiert zum Bereich „Analysehistorie“.
2. Die Anwendung sendet eine Anfrage an den Server, um die analysierten Bilder abzurufen.
3. Die Bilder und zugehörigen Diagnosedaten werden angezeigt.
4. Der Benutzer kann einzelne Bilder auswählen, um detaillierte Informationen zu erhalten.

**Beschreibung alternativer Schritte:**

* Falls keine Bilder gespeichert sind, wird eine Meldung angezeigt, dass noch keine Analysen durchgeführt wurden.

**Vor- und Nachbedingungen:**  
Vorbedingung: Der Benutzer hat bereits mindestens ein Bild analysiert.  
Nachbedingung: Der Benutzer kann frühere Analysen einsehen.

**Systemgrenzen:**  
Anwendung, Server, Gateway, Internetverbindung

#### Bild analysieren lassen

**Akteure:**  
Benutzer, Anwendung (Kamera & Analysemodul), KI-Komponente

**Auslöser/Trigger-Event:**  
Der Benutzer möchte ein Bild seiner Hautläsion aufnehmen und analysieren lassen.

**Kurzbeschreibung:**  
Der Benutzer kann mit der Kamera seines Geräts ein Bild aufnehmen und ein KI-Modell seiner Wahl zur Analyse nutzen. Falls das gewünschte Modell nicht verfügbar ist, wird eine entsprechende Mitteilung ausgegeben. Außerdem kann er die Ergebnisse speichern lassen.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer öffnet die Anwendung und wählt „Neue Analyse starten“.
2. Die Anwendung fragt die verfügbaren Modelle von der KI-Komponente ab.
3. Die Anwendung aktiviert die Kamera, und der Benutzer nimmt ein Foto auf.
4. Die Anwendung zeigt eine Vorschau des Bildes an.
5. Der Benutzer bestätigt das Bild oder nimmt ein neues auf.
6. Der Benutzer wählt ein KI-Modell für die Analyse aus.
7. Falls das gewählte Modell nicht verfügbar ist (z. B. nicht trainiert), erhält der Benutzer eine Mitteilung und kann ein alternatives Modell auswählen.
8. Die Anwendung sendet das Bild an den Server zur Analyse mit dem gewählten Modell.
9. Die KI analysiert das Bild und sendet das Ergebnis an die Anwendung.
10. Der Benutzer sieht die Analyseergebnisse auf seinem Bildschirm.
11. Der Benutzer wird gefragt, ob die Analyse in seiner Historie gespeichert werden soll.

**Beschreibung alternativer Schritte:**

* Falls das ausgewählte KI-Modell nicht verfügbar ist, wird eine Benachrichtigung ausgegeben, und der Benutzer kann ein anderes Modell auswählen.

**Vor- und Nachbedingungen:**

Vorbedingung: Der Benutzer hat Zugriff auf eine Kamera und die Anwendung verfügt mindestens ein trainiertes KI-Modell.

Nachbedingung: Die Analyseergebnisse werden auf dem Bildschirm angezeigt.

**Systemgrenzen:**  
Anwendung, Kamera, KI-Komponente, Internetverbindung

#### Automatische Bildzuschnitt-Funktion

**Akteure:**  
Benutzer, Anwendung (Bildverarbeitungssystem), Gateway, Server (Zuschnitt-Funktion)

**Auslöser/Trigger-Event:**  
Der Benutzer hat ein Bild aufgenommen und die automatische Zuschnitt-Funktion wird angeboten.

**Kurzbeschreibung:**  
Nach der Aufnahme kann die Anwendung das Bild automatisch zuschneiden, um die relevante Hautstelle optimal für die Analyse vorzubereiten.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer nimmt ein Bild auf.
2. Die Anwendung bietet an, das Bild automatisch zuzuschneiden.
3. Der Benutzer kann die automatische Anpassung akzeptieren oder das Bild manuell zuschneiden.
4. Falls akzeptiert, verarbeitet der Server das Bild und schneidet es entsprechend zu.
5. Der Benutzer kann nun das Ergebnis betrachten und es akzeptieren oder selbst zuschneiden.
6. Das zugeschnittene Bild wird zur Analyse an die KI gesendet.

**Beschreibung alternativer Schritte:**

* Falls der Benutzer das automatische Zuschneiden ablehnt, kann er das Bild manuell anpassen.
* Falls dem Benutzer das Ergebnis der Zuschneide-Funktion nicht gefällt, kann er das Bild manuell anpassen.

**Vor- und Nachbedingungen:**  
Vorbedingung: Der Benutzer hat ein Bild aufgenommen.  
Nachbedingung: Das Bild ist zugeschnitten und bereit zur Analyse.

**Systemgrenzen:**  
Anwendung, Kamera, Server, Gateway, Internetverbindung

#### Informationen zu verfügbaren KI-Modellen abrufen

**Akteure:**  
Benutzer, Anwendung (Modul zur Modellinformation)

**Auslöser/Trigger-Event:**  
Der Benutzer möchte sich über die verfügbaren KI-Modelle und deren Funktionalitäten informieren.

**Kurzbeschreibung:**  
Der Benutzer kann eine Übersicht aller KI-Modelle aufrufen und sich Details zu deren Funktionsweise, Training und Anwendungsgebieten anzeigen lassen.

**Beschreibung der einzelnen Schritte:**

1. Der Benutzer öffnet die Anwendung und navigiert zum Bereich „KI-Modelle“.
2. Die Anwendung ruft die Liste der verfügbaren Modelle vom Server ab.
3. Die Anwendung zeigt die verfügbaren Modelle mit einer kurzen Beschreibung an.

**Vor- und Nachbedingungen:**

Vorbedingung: Der Benutzer hat Zugriff auf die Anwendung und der Server enthält Informationen über die KI-Modelle.

Nachbedingung: Der Benutzer hat sich über die verfügbaren KI-Modelle informiert und kann fundierte Entscheidungen über deren Nutzung treffen.

**Systemgrenzen:**  
Anwendung, KI-Komponente, Internetverbindung

### Administrator

Administratoren verfügen über erweiterte Rechte innerhalb des Systems. Sie haben die Möglichkeit, Konfigurationen vorzunehmen, Nutzer zu verwalten und Systemprozesse zu überwachen. Die folgenden Use Cases beschreiben spezifische Interaktionen, die für Administratoren relevant sind.

#### KI-Modelle trainieren

**Akteure:**  
Administrator, Anwendung (Trainingsmodul), KI-Komponente, Datenbank

**Auslöser/Trigger-Event:**  
Der Administrator startet das Training eines spezifischen Modells oder aller Modelle.

**Kurzbeschreibung:**  
Der Administrator kann ein oder mehrere KI-Modelle trainieren. Die neuesten Datensätze werden aus der Datenbank abgerufen und zum Training verwendet. Das Training läuft auch bei Fehlern weiter, wobei fehlerhafte Modelle in einer Liste gespeichert und dem Administrator gemeldet werden.

**Beschreibung der einzelnen Schritte:**

1. Der Administrator öffnet die Anwendung und navigiert zur Modellverwaltung.
2. Er wählt aus, ob ein bestimmtes Modell oder alle Modelle trainiert werden sollen.
3. Der Administrator gibt die gewünschten Trainingsparameter ein:
   * **Resize Shape:** Zielgröße der Bilder für das Training
   * **Epochen:** Anzahl der Trainingsdurchläufe
4. Die Anwendung prüft, ob Benutzer berichtigt ist, Modelle zu trainieren.
5. Die Anwendung ruft die neuesten Datensätze aus der Datenbank ab.
6. Das Training startet für das ausgewählte Modell bzw. für alle Modelle.
7. Falls ein Modell nicht trainiert werden kann (z. B. aufgrund fehlerhafter Daten oder falscher Konfigurationen), wird dies in einer Fehlerliste gespeichert.
8. Das Training setzt mit dem nächsten Modell fort, ohne dass der Prozess abgebrochen wird.
9. Nach Abschluss des Trainings erhält der Administrator eine Übersicht über den Status der fehlerhaften Modelle (andere wurden erfolgreich trainiert).

**Beschreibung alternativer Schritte:**

* Falls der Administrator falsche Parameter eingibt (z. B. ungültiges Resize Shape oder Epochenzahl), wird eine Fehlermeldung ausgegeben, und er kann die Eingaben korrigieren.
* Falls der Benutzer nicht berechtigt ist, Modelle zu trainieren, wird eine Fehlermeldung angezeigt.

**Vor- und Nachbedingungen:**

Vorbedingung: Die Datenbank enthält Trainingsdaten.

Nachbedingung: Die gewählten Modelle wurden trainiert sowie gespeichert, und eine Fehlerliste wurde für nicht trainierbare Modelle erstellt.

**Systemgrenzen:**  
Anwendung, Server, Datenbank, Gateway, KI-Komponente, Internetverbindung

#### Classifier-Reports abrufen

**Akteure:**  
Administrator, Anwendung (Evaluierungsmodul), Server, KI-Komponente, Datenbank

**Auslöser/Trigger-Event:**  
Der Administrator möchte die Klassifikationsleistung eines spezifischen Modells oder aller Modelle anhand des aktuellen Datensatzes überprüfen.

**Kurzbeschreibung:**  
Der Administrator kann einen Klassifikationsbericht (Classifier Report) abrufen, um die Genauigkeit und Leistung eines oder aller Modelle zu analysieren. Dabei wird das Training ähnlich wie beim normalen Training durchgeführt, jedoch ohne Speicherung der trainierten Modelle. Stattdessen wird ein Report basierend auf 20 % Testdaten des gesamten Datenausmaßes ausgegeben.

**Beschreibung der einzelnen Schritte:**

1. Der Administrator öffnet die Anwendung und navigiert zur Modellbewertung.
2. Er wählt aus, ob ein spezifisches Modell oder alle Modelle analysiert werden sollen.
3. Die Anwendung prüft, ob der Benutzer berechtigt ist, Klassifikationsberichte erstellen zu lassen
4. Die Anwendung ruft die neuesten Datensätze aus der Datenbank ab.
5. Das Modell/die Modelle werden mit den aktuellen Daten getestet, ohne die trainierten Modelle zu speichern.
6. Die Anwendung verwendet 80 % der Daten für das Training und 20 % für die Evaluierung.
7. Nach der Evaluierung wird ein Klassifikationsbericht erstellt, der Informationen zu folgenden Metriken enthält:
   * **Präzision** (Precision)
   * **Trefferquote** (Recall)
   * **F1-Score**
   * **Genauigkeit** (Accuracy)
8. Der Report wird dem Administrator angezeigt und kann exportiert werden.

**Beschreibung alternativer Schritte:**

* Falls keine ausreichenden Daten für die Evaluierung vorhanden sind, wird eine Warnmeldung ausgegeben.
* Falls ein Modell fehlschlägt, wird dies in einer Fehlerliste gespeichert, und die Evaluierung fährt mit dem nächsten Modell fort.

**Vor- und Nachbedingungen:**

Vorbedingung: Die Anwendung hat Zugriff auf die neuesten Datensätze.

Nachbedingung: Der Administrator erhält einen detaillierten Report zur Modellleistung.

**Systemgrenzen:**  
Anwendung, Server, Datenbank, Gateway, KI-Komponente, Internetverbindung

# Nicht Funktionale Anforderungen

*Kapitel verfasst von Daniel Jessner*

Dieses Kapitel enthält die Anforderungen, die an die Anwendung gestellt werden, welche nicht die Funktionalität des Systems betreffen. Diese definieren, abgekapselt von jeglicher Implementierungslogik und verwendeten Methoden, die Rahmenbedingungen nicht nur für den Endnutzer und seine Erfahrungen mit der entwickelten Software sondern auch jene für Entwickler/Administratoren.

## Benutzerfreundlichkeit

Die mobile Anwendung sollte intuitiv gestaltet sein, um eine einfache Bedienung für Benutzer aller Erfahrungsstufen zu gewährleisten. Außerdem sollte die Reaktionszeit des Servers sowie die Latenz zwischen Frontend und Backend so gering wie möglich sein, um ein reibungsloses und ansprechendes Benutzererlebnis zu ermöglichen.

## Zuverlässigkeit

Es ist ein äußert wichtiger Punkt, dass die Verbindung zwischen Frontend und Backend zu allen Zeiten voll funktional ist und verlässlich Benutzeranfragen entgegennimmt. Weiters soll das System dahingehend robust sein, dass es angemessen auf Ausnahmesituationen reagiert, um einen kontinuierlichen Betrieb auch bei unvorhergesehenen Ereignissen sicherzustellen.

## Skalierbarkeit

Es ist gewünscht, das System so zu gestalten, dass es auch in der Zukunft liegende Erweiterungen und zusätzliche Änderung ohne großen Aufwand unterstützt. Die Software sollte die Funktionalität bereitstellen, dem Benutzer mehrere Algorithmen zur Erkennung der Hautläsionen zur Verfügung zu stellen, ohne dabei die Perfomance oder Stabilität des Systems zu beeinträchtigen.

## Sicherheit

Sowohl die Speicherung als auch die Übertragung der benutzerbezogenen Daten (Email, Passwort, Bilder) sollte verschlüsselt vonstattengehen. Dafür werden Passwörter nur unter Benutzung eines Hashs persistiert und Übertragungen mittels eines geeigneten Algorithmus verschlüsselt.

## Plattformkompatibilität

Wie auch die Grundanforderungen des Projektes vorgeben, sollte die Software derartig unabhängig sein, dass der Benutzer sowohl in einer Windows-Umgebung als auch auf mobilen Endgeräten (Android) einsteigen kann. Dabei ist die Responsivität der Software von großer Wichtigkeit, um eine konsistente Darstellung zu jeder Zeit zu gewährleisten.

## Dokumentation und Wartbarkeit

Das Projekt sollte ausführlich dokumentiert sein, einschließlich Anleitungen zur Installation, Konfiguration und Verwendung der Steuerungssoftware. Dazu gehören auch eine gute Strukturierung sowie Kommentierung des Quellcodes, um eine einfache Wartung und Weiterentwicklung des Systems zu ermöglichen.

# Projektplanung

*Kapitel verfasst von Daniel Jessner*

Die erfolgreiche Umsetzung eines Projekts erfordert eine strukturierte und detaillierte Planung, die sowohl technische als auch organisatorische Aspekte berücksichtigt. In diesem Kapitel werden die verschiedenen Planungsschritte erläutert, die für die Entwicklung des Projekts durchgeführt wurden.

## Recherche / Vorarbeit

Bevor mit der eigentlichen Implementierung des Projekts begonnen werden konnte, war eine umfassende Recherche und theoretische Vorarbeit erforderlich. Jeder Projektteilnehmer hat sich mit spezifischen Fachthemen auseinandergesetzt, um eine solide Grundlage für die Entwicklung zu schaffen. Diese Vorbereitungen umfassten sowohl technische als auch konzeptionelle Aspekte, darunter die Analyse relevanter Technologien, wissenschaftlicher Grundlagen sowie bestehender Lösungen und Frameworks.

Die Recherche diente nicht nur der Wissensaneignung, sondern auch der Identifikation möglicher Herausforderungen und der Evaluierung geeigneter Methoden und Werkzeuge. Dadurch konnte sichergestellt werden, dass fundierte Entscheidungen für die Umsetzung des Projekts getroffen wurden. In den folgenden Abschnitten werden die einzelnen Beiträge der Projektteilnehmer detailliert erläutert.

### Daniel Jessner (KI-Komponente, Gateway)

*Kapitel verfasst von Daniel Jessner*

#### Einführung

Bevor mit dem eigentlichen Training von KI-Modellen begonnen werden kann, ist eine gründliche Recherche und Vorarbeit essenziell. Eine fundierte theoretische Grundlage ermöglicht es, die geeigneten Algorithmen und Frameworks auszuwählen sowie potenzielle Herausforderungen frühzeitig zu erkennen. Zudem hilft die Recherche dabei, bewährte Methoden zur Datenaufbereitung, Modelloptimierung und Fehlervermeidung zu identifizieren. Da sich das Projektteam mit dieser Arbeit in ein ihnen unbekanntes Terrain wagt, ist dieser Schritt unerlässlich für den Erfolg des Projektes.

Die Vorarbeit umfasst mehrere wichtige Schritte, die die Qualität und Effizienz des Modelltrainings maßgeblich beeinflussen. Dazu gehören die Auswahl geeigneter Technologien, die Beschaffung und Bereinigung von Trainingsdaten (eigene Projekt-Komponente) sowie erste Tests mit Basisalgorithmen. Durch eine systematische Vorgehensweise wird sichergestellt, dass das spätere Modelltraining auf einer stabilen und gut vorbereiteten Grundlage aufbaut.

#### Theoretische Grundlagen – KI

Diese Projekt-Komponente beschäftigt sich zum größten Teil mit der Verwaltung und auch Entwicklung von KI-Modellen, um Eingabedaten anhand zuvor definierter Trainingsdaten zu klassifizieren. Dabei bedient sie sich sogenannten „**Classifiers**“, also KI-Modellen, welche ein solches Konzept realisieren. Diese Classifier sind definierte, oftmals komplexe Algorithmen und basieren auf zwei Ansätzen:

1. **Maschinelles Lernen** ist ein Bereich der Künstlichen Intelligenz, der es Computern ermöglicht, Muster in Daten zu erkennen und Vorhersagen zu treffen, ohne explizit programmiert zu sein. Dabei unterscheidet man zwischen verschiedenen Lernarten:

* **Überwachtes Lernen**: Das Modell wird mit gekennzeichneten Daten (Labels) trainiert, um eine bestimmte Ausgabe vorherzusagen (relevant für dieses Projekt).
* **Unüberwachtes Lernen**: Das Modell erkennt Muster in unmarkierten Daten, z. B. durch Clustering.
* **Bestärkendes Lernen**: Das Modell lernt durch Belohnungen aus Interaktionen mit seiner Umgebung.

2. **Deep Learning** ist eine spezielle Form des maschinellen Lernens, die auf künstlichen neuronalen Netzen basiert. Diese Netzwerke bestehen aus mehreren Schichten (Deep Neural Networks) und sind besonders gut für komplexe Mustererkennung in großen Datenmengen geeignet. Im Vergleich zu klassischen ML-Algorithmen können DL-Modelle automatisch relevante Merkmale aus den Daten extrahieren, was sie für Aufgaben wie Bild- und Spracherkennung besonders leistungsfähig macht. Dieser Ansatz des maschinellen Lernens wird von CNNs, also **Convolutional Neural Networks**, implementiert, worauf später noch genauer eingegangen wird.

#### Auswahl geeigneter Technologien

Grundsätzlich war von Anfang an klar, dass sich im Bezug auf verwendete Technologien sowie das Basiskonstrukt des Projektes an die Vorgaben des Tech-Stacks des Projektgebers (siehe Kapitel XY) gehalten wird. Dadurch fiel die Entscheidung auf relativ einfach auf folgende Punkte:

* **KI-Modelle**
  + Pytorch
  + Tensorflow Keras
  + Scikit-Learn
* **API**
  + FastAPI

Abgesehen von den Vorgaben sprechen noch einige weitere Aspekte für die Verwendung oben genannter Technologien:

**1. Python**

Python ist die bevorzugte Programmiersprache für Künstliche Intelligenz und maschinelles Lernen aufgrund seiner einfachen Syntax, umfangreichen Bibliotheken und starken Community-Unterstützung.

Die wichtigsten Vorteile sind:

* **Lesbarkeit und Benutzerfreundlichkeit**: Ermöglicht eine schnelle Entwicklung und erleichtert die Zusammenarbeit.
* **Breites Ökosystem an ML- und DL-Bibliotheken**: Bibliotheken wie **Scikit-Learn, TensorFlow und PyTorch** bieten leistungsfähige Tools für maschinelles Lernen und Deep Learning.
* **Effiziente Datenverarbeitung**: Python unterstützt leistungsfähige Bibliotheken wie **NumPy, Pandas und Matplotlib**, die für Datenanalyse und Visualisierung essenziell sind.
* **Gute Integration mit anderen Technologien**: Python kann leicht mit **C++, Java und cloudbasierten ML-Plattformen** integriert werden, was die Skalierbarkeit von KI-Modellen verbessert.

**2. KI-Bibliotheken**

Je nach Anwendungsfall eignen sich Frameworks unterschiedlich gut für zu bewältigende Aufgaben. Die KI-Komponente des DermaAI-Systems implementiert einen vielfältigen Mix von Modellen aus den drei verwendeten Bibliotheken.

|  |  |  |  |
| --- | --- | --- | --- |
| **Framework** | **Einsatzgebiet** | **Vorteile** | **Nachteile** |
| |  | | --- | | **Scikit-Learn** | | |  | | --- | | Klassische ML-Modelle (z. B. Entscheidungsbäume, SVMs, logistische Regression) | | Einfache Implementierung, gute Dokumentation, ideal für kleine bis mittelgroße Datensätze | Nicht für neuronale Netze oder große Datenmengen optimiert |
| |  | | --- | | **PyTorch** | | |  | | --- | | Deep Learning, flexible neuronale Netze (z. B. CNNs, RNNs, Transformer) | | |  | | --- | | Dynamische Berechnungsgrafen, einfaches Debugging, intuitive API | | Etwas weniger für großskalige Produktlösungen optimiert als TensorFlow |
| |  | | --- | | **TensorFlow** | | Skalierbares Deep Learning für Produktion, Cloud und Mobilgeräte | Hohe Effizienz, TensorFlow Serving für Deployment, GPU-Unterstützung | Komplexere API als PyTorch, steilere Lernkurve |

**-> Scikit-Learn** eignet sich besonders für klassische ML-Modelle wie Klassifikation, Regression oder Clustering.  
**-> PyTorch** ist ideal für forschungsorientierte Deep-Learning-Experimente, da es eine flexible und intuitive API bietet.  
**-> TensorFlow** ist besser für großskalige neuronale Netze in Produktionsumgebungen, insbesondere für Cloud- und Mobile-Deployments.

**3. FastAPI**

**FastAPI** ist ein modernes und leistungsstarkes Web-Framework für den Bau von APIs mit Python. Es hat einige herausragende Merkmale, die es besonders attraktiv für Entwickler machen, die APIs bauen möchten:

**Hohe Leistung**

* **FastAPI** ist eines der schnellsten Web-Frameworks in Python und bietet eine außergewöhnliche Leistung, die mit **Node.js** und **Go** konkurrieren kann. Die hohe Geschwindigkeit ist ein wichtiger Faktor, wenn es um die Entwicklung von APIs geht, da sie eine geringe Latenz und schnelle Antworten gewährleisten kann.
* Der Grund für die Geschwindigkeit ist, dass **FastAPI** auf **Starlette** basiert, einem schnellen und asynchronen Web-Framework. FastAPI verwendet die **ASGI**-Spezifikation, die eine hohe Performance bei gleichzeitigen Verbindungen ermöglicht.

**Automatische Dokumentation**

* FastAPI erstellt automatisch interaktive API-Dokumentationen für deine Endpunkte, und zwar direkt während der Entwicklung.
* Es verwendet **Swagger UI** und **ReDoc**, um eine benutzerfreundliche, visuelle API-Dokumentation bereitzustellen. Diese wird unter der URL /docs (Swagger UI) und /redoc (ReDoc) bereitgestellt.

**Vorteile**:

* + Entwicklern und API-Nutzern wird eine leicht verständliche, sofort zugängliche Dokumentation zur Verfügung gestellt.
  + Die Dokumentation ist immer aktuell, da sie auf der tatsächlichen Implementierung der API basiert.

**Einfach zu bedienen und flexibel**

* FastAPI nutzt Python-Typannotationen (zum Beispiel int, str, float, etc.), um Eingaben zu validieren, was die Entwicklung vereinfacht und weniger fehleranfällig macht. Das bedeutet, dass du keine separate Validierungslogik schreiben musst, sondern die Typen automatisch validiert werden.
* Durch die Verwendung von **Pydantic** für die Datenvalidierung können Modelle sehr einfach und effizient erstellt werden. Pydantic sorgt für automatische Datentypprüfung und Fehlerbehandlung.

**Asynchronität (Async / Await)**

* FastAPI ermöglicht es, asynchrone Endpunkte zu erstellen, die bei Bedarf sehr viele gleichzeitige Anfragen effizient verarbeiten können. Das bedeutet, dass du **async**/**await** verwenden kannst, um **asynchrone Programmierung** zu integrieren und z. B. auf **Datenbanken** oder **Web-APIs** effizient zuzugreifen.
* Diese asynchronen Endpunkte bieten eine deutlich bessere Leistung bei hoher Last und stellen sicher, dass deine API schnell bleibt, selbst wenn viele gleichzeitige Anfragen verarbeitet werden müssen.

**Sicherheit und Authentifizierung**

* FastAPI bietet Unterstützung für gängige **Authentifizierungsmethoden** wie **OAuth2**, **JWT (JSON Web Tokens)**, und **API-Schlüssel**.
* Es gibt eingebaute Mechanismen, um sicherheitsrelevante Operationen wie **CORS (Cross-Origin Resource Sharing)**, **CSRF (Cross-Site Request Forgery)** und **Request Validation** effizient zu implementieren.

**Einfacher Einstieg und gute Dokumentation**

* FastAPI hat eine **exzellente Dokumentation**, die für Einsteiger gut verständlich ist. Die Dokumentation führt Schritt für Schritt durch den Prozess der API-Entwicklung und hilft dabei, Best Practices zu etablieren.
* Die Nutzung von Python-Typen für **Datenvalidierung** und **Autovervollständigung** in modernen IDEs wie **VS Code** erleichtert das Schreiben von Code.

**Unterstützung für moderne Python-Funktionen**

* FastAPI nutzt moderne Python-Funktionen wie **Type Hints** und **Pydantic-Modelle**. Dadurch kannst du sicherstellen, dass deine API effizient und fehlerfrei bleibt.
* Es unterstützt **Python 3.6+**, einschließlich der neuesten Versionen, die die neuesten Funktionen und Verbesserungen bieten.

**Leichtgewichtig und skalierbar**

* FastAPI ist leichtgewichtig, sodass es keine unnötige Komplexität oder Overhead mit sich bringt, aber gleichzeitig auch in der Lage ist, komplexe Anwendungen und Skalierung zu bewältigen.
* FastAPI kann man für kleine Projekte bis hin zu großen, skalierbaren APIs verwenden.

**Open-Source und Community**

* FastAPI ist **Open-Source**, was bedeutet, dass du jederzeit den Quellcode einsehen und zur Weiterentwicklung beitragen kannst.
* Es gibt eine aktive Community, die regelmäßig neue Funktionen, Plugins und Erweiterungen entwickelt.

**Sehr gute Integration mit anderen Technologien**

* FastAPI lässt sich sehr gut mit anderen Tools und Frameworks kombinieren, wie zum Beispiel **SQLAlchemy** (für die Datenbankinteraktion), **Celery** (für Hintergrundaufgaben), **Redis** (für Caching und Messaging) und vielen weiteren.
* FastAPI kann man problemlos mit anderen modernen Python-Bibliotheken und Frameworks verwenden.

#### Installation KI-Modelle

Um im Anschluss die KI-Bibliotheken verwenden zu können, müssen diese zuvor installiert werden. Dazu wird ganz einfach der für Python entwickelte Package Manager „**PIP, packager installer python**“ verwendet, die Bibliotheken werden automatisch aus dem Internet geladen:

* **Scikit-Learn**

****

* **Pytorch**
  + **Ohne GPU-Untersützung (CPU-Only)**

****

* + **Mit GPU-Unterstützung**

****

* **TensorFlow**
  + **Ohne GPU-Untersützung (CPU-Only)**

****

* + **Mit GPU-Unterstützung**

****

**1. Scikit-Learn (scikit-learn)**

* **Was wird installiert?**
  + **Scikit-Learn** ist eine weit verbreitete Bibliothek für maschinelles Lernen in Python. Sie stellt eine Sammlung von Werkzeugen für viele gängige maschinelle Lernverfahren zur Verfügung, wie Klassifikation, Regression, Clustering und Dimensionalitätsreduktion.
* **Was wird durch die Installation bereitgestellt?**
  + Funktionen zur Modellbildung, wie **lineare Regression, Support Vector Machines (SVM), K-nearest Neighbors (KNN), Random Forest** und viele andere klassische ML-Algorithmen.
  + Funktionen zur **Datenvorverarbeitung** (z. B. Skalierung von Merkmalen, Umgang mit fehlenden Werten).
  + **Kreuzvalidierung** und **Hyperparameter-Tuning**.

**2. PyTorch (torch)**

* **Was wird installiert?**
  + **PyTorch** ist ein Deep-Learning-Framework, das es Entwicklern ermöglicht, neuronale Netze zu bauen und zu trainieren. Die Bibliothek enthält Funktionen zum Arbeiten mit **Tensors** (mehrdimensionalen Arrays) und zur **Automatischen Differenzierung** (für Backpropagation und Gradientenberechnung).
* **Was wird durch die Installation bereitgestellt?**
  + Grundlegende Funktionen für Deep Learning, einschließlich **Tensor-Operationen** (z. B. Addition, Multiplikation).
  + **Autograd**: Automatische Differenzierung, um Gradienten für das Training von neuronalen Netzen zu berechnen.
  + Unterstützung für **GPU-Computing** (durch CUDA) und **Datenparallele Verarbeitung**.
  + Flexibilität, um benutzerdefinierte **neuronale Netzwerke** zu erstellen und zu trainieren.

**3. torchvision**

* **Was wird installiert?**
  + **torchvision** ist eine Erweiterung von **PyTorch**, die speziell auf **Bildverarbeitung** fokussiert ist. Es enthält Tools und vortrainierte Modelle für Computer Vision Aufgaben wie Bildklassifikation, Objekterkennung und Segmentierung.
* **Was wird durch die Installation bereitgestellt?**
  + Vorverarbeitungsfunktionen für Bilder wie **Zuschneiden**, **Normalisierung** und **Skalierung**.
  + Vortrainierte **CNN-Modelle** (Convolutional Neural Networks), wie **ResNet**, **VGG** und **AlexNet**.
  + **Daten-Datasets** wie **CIFAR-10**, **ImageNet**, **COCO**.
  + Methoden zur **Datenaugmentation** (z. B. Bilddrehen, Spiegeln), um die Trainingsdaten zu erweitern.

**4. cudatoolkit**

* **Was wird installiert?**
  + **CUDA-Toolkit** ist eine Sammlung von Tools und Bibliotheken, die von **NVIDIA** entwickelt wurden, um die Leistung von GPU-Computing zu nutzen. Es stellt die Schnittstellen und Bibliotheken zur Verfügung, die erforderlich sind, damit Software wie **TensorFlow** oder **PyTorch** die **GPU** zur Berechnung von rechenintensiven Aufgaben (wie das Training von neuronalen Netzen) verwenden kann.
* **Was wird durch die Installation bereitgestellt?**
  + **CUDA-Bibliotheken** und Tools, die eine schnelle Verarbeitung durch die **NVIDIA-GPU** ermöglichen.
  + Optimierung und Beschleunigung von Deep-Learning-Modellen.
  + Verbindung zwischen der Software und der GPU für **parallelisierte Berechnungen**.
* **Hinweis**: Um GPU-Computing zu nutzen, muss man sicherstellen, dass die verwendete Grafikkarte und Treiber mit CUDA kompatibel sind.

**5. TensorFlow**

* **Was wird installiert?**
  + **TensorFlow** ist ein Open-Source-Deep-Learning-Framework von **Google**, das für maschinelles Lernen und neuronale Netzwerke entwickelt wurde. Es unterstützt sowohl **CPU** als auch **GPU** und eignet sich für **Forschung** und **Produktion**.
* **Was wird durch die Installation bereitgestellt?**
  + **Modelle und Tools** für maschinelles Lernen und Deep Learning, wie **Klassifikation**, **Regressionsmodelle**, **Textanalyse**, **Bildverarbeitung** und **Sprachverarbeitung**.
  + **TensorFlow Serving** und **TensorFlow Lite** für den **Produktionsbetrieb** (z. B. Bereitstellung auf Servern oder mobilen Geräten).
  + Optimierung der **Modell-Performance** (einschließlich GPU-Beschleunigung, **TensorFlow Extended (TFX)** für Produktionspipelines).

**6. tensorflow-gpu**

* **Was wird installiert?**
  + **tensorflow-gpu** ist die Version von **TensorFlow**, die **GPU-Unterstützung** integriert hat. Dies ermöglicht es, die **GPU** für das Training und die Berechnung von Deep-Learning-Modellen zu verwenden, was die Berechnungen deutlich beschleunigt.
* **Was wird durch die Installation bereitgestellt?**
  + **GPU-Unterstützung** für TensorFlow, was besonders bei großen Modellen und großen Datensätzen nützlich ist.
  + Die **CUDA**-Bibliotheken und Treiber werden mitinstalliert, sodass TensorFlow automatisch auf **NVIDIA-GPUs** zugreifen kann.
  + Schnelleres Training von Deep-Learning-Modellen im Vergleich zur **CPU-Version**, da die **GPU** für parallele Berechnungen verwendet wird.

#### Installation FastAPI

Um eine anständige Verbindung zum Frontend zu gewährleisten, wird FastAPI verwendet. Ebenso wie bei den Modellen wird dieses Modul über den Package Manager installiert:



Wichtig hierbei ist, dass FastAPI als Web-Framework auch einen ASGI-Server benötigt, um ausgeführt werden zu können. Uvicorn ist ein leichtgewichtiger ASGI-Server, der für FastAPI empfohlen wird:



**Erklärung ASGI:**

**ASGI (Asynchronous Server Gateway Interface)** ist eine Spezifikation für Web-Server und Python-Frameworks, die asynchrone Kommunikation unterstützt. Es ist die Weiterentwicklung des älteren WSGI (Web Server Gateway Interface), das die Grundlage für die meisten Python-Web-Frameworks bildet.

ASGI wurde entwickelt, um die Einschränkungen von WSGI zu überwinden und asynchrone Programmierung und parallele Verarbeitung von Anfragen zu ermöglichen. Dies ist besonders nützlich für Web-APIs, echte Echtzeit-Anwendungen wie Chat-Apps oder Spiele, sowie Anwendungen, die viele gleichzeitige Verbindungen benötigen.

#### Testexperiment

Am Anfang ist es wichtig, die Arbeit in diesem Bereich etwas kennenzulernen, ganz besonders wenn jemand noch nie zuvor damit in Berührung gekommen ist. Im Falle des DermaAI-Projektteams stellt die KI-Welt etwas Fremdes dar, daher wurde zu Beginn ein kleines Testexperiment mit Scikit-Learn implementiert, um den Umgang mit den Python-Bibliotheken zu verstehen.

Scikit-Learn stellt eine große Palette an unterschiedlichen KI-Algorithmen zur Verfügung:

Ein Bild, das Text, Screenshot, Grafikdesign, Design enthält.

Automatisch generierte Beschreibung

* + Es wird sich hier, wie schon erwähnt, auf die Classification-Algorithmen beschränkt

Anschließend wird ein Testskript in Python erstellt, um eine kleine Selektion der Scikit-Learn Algorithmen zu testen und deren Ergebnisse zu vergleichen (jegliche nicht definierte Funktionen sind aus dem Scikit-Learn-Framework oder anderen importiert):

**1. Classifier deklarieren**

Hier wird eine Liste von Klassifikatoren und ihre entsprechenden Namen erstellt. Diese Klassifikatoren werden später verwendet, um die trainierten Modelle zu evaluieren und die Entscheidungsgrenzen zu visualisieren.

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

KI-generierte Inhalte können fehlerhaft sein.

* + **names** enthält die Namen der verschiedenen Klassifizierer, die im Code verwendet werden.
  + **classifiers** enthält die Instanzen der verschiedenen Klassifizierer aus sklearn

Diese Algorithmen sind auch Teil der fertigen DermaAI-Anwendung und werden im Kapitel XY mit ihren Parametern genauer erklärt.

**2. Datensätze generieren**

In diesem Schritt werden die Datensätze erstellt, die für das Training und Testen der Classifier verwendet werden. Die **make\_classification**, **make\_moons** und **make\_circles** Funktionen erzeugen synthetische Datensätze für Tests.

1. X, y = make\_classification(

2. n\_features=2, n\_redundant=0, n\_informative=2, random\_state=1, n\_clusters\_per\_class=1)

3. rng = np.random.RandomState(2)

4. X += 2 \* rng.uniform(size=X.shape)

5. linearly\_separable = (X, y)

6.

7. datasets = [

8. make\_moons(noise=0.3, random\_state=0),

9. make\_circles(noise=0.2, factor=0.5, random\_state=1),

10. linearly\_separable,

11. ]

12.

* + **make\_classification** erzeugt einen Datensatz für eine Klassifikationsaufgabe mit 2 informativen Features.
  + **make\_moons** und **make\_circles** erzeugen jeweils Datensätze, die für Klassifikationen von halbmond- oder kreisförmigen Daten verwendet werden.
  + **linearly\_separable** ist ein benutzerdefinierter Datensatz, der eine lineare Trennbarkeit aufweist.

**3. Erstellen einer Plot-Figur**

Hier wird die Plot-Figur mit einer bestimmten Größe erstellt, die später alle Subplots für die Datensätze und die Ergebnisse der Klassifikatoren enthalten wird.

1. figure = plt.figure(figsize=(27, 9))

2. i = 1

* + **plt.figure(figsize=(27, 9))** legt eine Abbildung fest, die Platz für Subplots bietet.
  + **i** ist der Index für die Subplots, der später bei der Zuordnung der Plot-Positionen verwendet wird:

**4. Plotten der Datensätze**

In diesem Schritt wird jeder der drei Datensätze geplottet, wobei die Trainings- und Testdaten durch unterschiedliche Farben unterschieden werden:

1. for ds\_cnt, ds in enumerate(datasets):

2. # preprocess dataset, split into training and test part

3. X, y = ds

4. X\_train, X\_test, y\_train, y\_test = train\_test\_split(

5. X, y, test\_size=0.4, random\_state=42

6. )

7.

8. x\_min, x\_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5

9. y\_min, y\_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

10.

11. cm = plt.cm.RdBu

12. cm\_bright = ListedColormap(["#FF0000", "#0000FF"])

13. ax = plt.subplot(len(datasets), len(classifiers) + 1, i)

14. if ds\_cnt == 0:

15. ax.set\_title("Input data")

16. ax.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=cm\_bright, edgecolors="k")

17. ax.scatter(

18. X\_test[:, 0], X\_test[:, 1], c=y\_test, cmap=cm\_bright, alpha=0.6, edgecolors="k"

19. )

20. ax.set\_xlim(x\_min, x\_max)

21. ax.set\_ylim(y\_min, y\_max)

22. ax.set\_xticks(())

23. ax.set\_yticks(())

24. i += 1

* + **train\_test\_split** teilt die Daten in Trainings- und Testdaten (60% Training und 40% Test).
  + Die **scatter** Methode wird verwendet, um die Datenpunkte zu visualisieren.
  + **x\_min, x\_max, y\_min, y\_max** definieren den Bereich der x- und y-Achse, um sicherzustellen, dass alle Daten sichtbar sind.

**5. Trainieren und Auswerten der Klassifikatoren**

Für jeden Klassifikator wird ein Pipeline-Objekt erstellt, das den StandardScaler für die Datenvorverarbeitung und den jeweiligen Klassifikator beinhaltet. Der Klassifikator wird trainiert und auf den Testdaten evaluiert:

1. for name, clf in zip(names, classifiers):

2. ax = plt.subplot(len(datasets), len(classifiers) + 1, i)

3.

4. clf = make\_pipeline(StandardScaler(), clf)

5. clf.fit(X\_train, y\_train)

6. score = clf.score(X\_test, y\_test)

7. DecisionBoundaryDisplay.from\_estimator(

8. clf, X, cmap=cm, alpha=0.8, ax=ax, eps=0.5

9. )

10.

11. ax.scatter(

12. X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=cm\_bright, edgecolors="k"

13. )

14. ax.scatter(

15. X\_test[:, 0],

16. X\_test[:, 1],

17. c=y\_test,

18. cmap=cm\_bright,

19. edgecolors="k",

20. alpha=0.6,

21. )

22.

23. ax.set\_xlim(x\_min, x\_max)

24. ax.set\_ylim(y\_min, y\_max)

25. ax.set\_xticks(())

26. ax.set\_yticks(())

27. if ds\_cnt == 0:

28. ax.set\_title(name)

29. ax.text(

30. x\_max - 0.3,

31. y\_min + 0.3,

32. ("%.2f" % score).lstrip("0"),

33. size=15,

34. horizontalalignment="right",

35. )

36. i += 1

* + **make\_pipeline** erstellt eine Pipeline, die den **StandardScaler** zur Normalisierung der Eingabedaten und den Klassifikator selbst enthält.
  + **clf.fit(X\_train, y\_train)** trainiert den Klassifikator auf den Trainingsdaten.
  + **DecisionBoundaryDisplay.from\_estimator** zeigt die Entscheidungsgrenze des Klassifikators.
  + Die **ax.text** Methode zeigt die Genauigkeit des Klassifikators auf dem Plot an.

**6. Layout anpassen und Plot anzeigen**

1. plt.tight\_layout()

2. plt.show()

* + **plt.tight\_layout()** stellt sicher, dass alle Subplots ordentlich und ohne Überlappung angezeigt werden.
  + **plt.show()** zeigt den Plot auf dem Bildschirm an.

**Ergebnis:**

**Teil 1:**

**Ein Bild, das Screenshot, Kunst, Mosaik enthält.

KI-generierte Inhalte können fehlerhaft sein.**

**Teil 2:**

**Ein Bild, das Screenshot, Mosaik, Kunst, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.**

Das gezeigte Bild stellt eine Visualisierung verschiedener Klassifikationsalgorithmen dar, die auf drei verschiedene Datensätze angewendet wurden. Jeder Algorithmus versucht, eine Entscheidungsgrenze zu lernen, um die beiden Klassen (rot und blau) korrekt zu trennen.

**1. Aufbau der Visualisierung**

Die Darstellung besteht aus einer **Matrix von Plots**, wobei:

* **Jede Zeile** einen anderen Datensatz repräsentiert (von oben nach unten: Mondförmige Daten (make\_moons), Kreisförmige Daten (make\_circles), Linear trennbare Daten (make\_classification)).
* **Jede Spalte** einen anderen Klassifikationsalgorithmus repräsentiert (von links nach rechts: Unterschiedliche Methoden wie k-Nächste Nachbarn, SVMs, Entscheidungsbäume, neuronale Netze etc.).
* Das erste Bild in jeder Zeile zeigt den **rohen Datensatz**, also die ursprünglichen Punkte ohne eine Klassifikation.
* Die übrigen Bilder zeigen die **Entscheidungsgrenzen**, die durch die jeweiligen Algorithmen gelernt wurden. Die eingefärbten Regionen repräsentieren die Vorhersagen des jeweiligen Klassifikators:
  + **Rote Regionen** → Klassifikation als Klasse 1 (rot).
  + **Blaue Regionen** → Klassifikation als Klasse 2 (blau).
  + Die Zahlen in den Diagrammen repräsentieren die Genauigkeit (Accuracy) des jeweiligen Klassifikators auf dem Testdatensatz.

**2. Analyse der Algorithmen und Entscheidungsgrenzen**

**Nearest Neighbors (K-Nearest Neighbors, KNN)**

* **Charakteristik:** KNN basiert darauf, dass ein Punkt die gleiche Klasse wie seine **nächsten Nachbarn** annimmt.
* **Entscheidungsgrenze:** Sehr flexibel und passt sich stark an die Struktur der Daten an, daher eher **unregelmäßige Grenzen**.
* **Stärken:** Gute Leistung bei komplexen, nicht-linearen Datensätzen.
* **Schwächen:** Kann übermäßig an die Trainingsdaten angepasst sein (Overfitting), besonders bei verrauschten Daten.

**Linear SVM (Support Vector Machine mit linearem Kernel)**

* **Charakteristik:** Versucht, eine **lineare Trennlinie** zu finden, die die beiden Klassen optimal trennt.
* **Entscheidungsgrenze:** Gerade Linie – sehr gut für den linearen Datensatz (unterste Zeile), aber **ungeeignet für nicht-lineare Daten** (Mond- und Kreisförmige Daten).
* **Stärken:** Sehr gut für **linear separierbare Daten**.
* **Schwächen:** Funktioniert nicht für komplexe, nicht-lineare Strukturen.

**RBF SVM (Support Vector Machine mit radialem Basisfunktions-Kernel)**

* **Charakteristik:** Erweitert die lineare SVM, indem sie einen nicht-linearen Kernel verwendet, der die Daten in eine höhere Dimension transformiert.
* **Entscheidungsgrenze:** Sehr geschwungene, **komplexe Entscheidungsmuster**, die sich gut an nicht-lineare Strukturen anpassen.
* **Stärken:** Sehr leistungsfähig für nicht-lineare Probleme (wie hier für make\_moons und make\_circles).
* **Schwächen:** Erfordert sorgfältige Wahl der Hyperparameter (gamma, C).

**Gaussian Process Classifier (GPC)**

* **Charakteristik:** Probabilistischer Ansatz, der Wahrscheinlichkeitsverteilungen über Funktionen modelliert.
* **Entscheidungsgrenze:** Sehr **glatte, weiche Trennungen**, besonders sichtbar bei den kreisförmigen Daten.
* **Stärken:** Liefert nicht nur eine Klassifikation, sondern auch eine **Unsicherheitsabschätzung**.
* **Schwächen:** Hoher Rechenaufwand für große Datensätze.

**Decision Tree (Entscheidungsbaum)**

* **Charakteristik:** Hierarchische Struktur, die schrittweise die Daten in Bereiche unterteilt.
* **Entscheidungsgrenze:** **Stufenförmige, eckige Trennungen**, da Entscheidungsbäume durch einfache Regeln arbeiten.
* **Stärken:** Schnell zu trainieren, interpretiert leicht.
* **Schwächen:** Neigt zum Overfitting, besonders bei kleinen Datenmengen.

**Random Forest (Ensemble von Entscheidungsbäumen)**

* **Charakteristik:** Kombiniert mehrere Entscheidungsbäume, um eine robustere Klassifikation zu ermöglichen.
* **Entscheidungsgrenze:** Ähnlich wie Decision Trees, aber **glatter und stabiler**, weil mehrere Bäume kombiniert werden.
* **Stärken:** Reduziert Overfitting, robuste Performance.
* **Schwächen:** Weniger interpretierbar als ein einzelner Entscheidungsbaum.

**Neural Network (Mehrschichtige neuronale Netze, MLPClassifier)**

* **Charakteristik:** Mehrere Schichten von Neuronen lernen **hochgradig nicht-lineare Entscheidungsgrenzen**.
* **Entscheidungsgrenze:** Sehr **geschwungene, flexible Grenzen**, besonders sichtbar bei make\_moons und make\_circles.
* **Stärken:** Kann sehr komplexe Muster lernen.
* **Schwächen:** Erfordert viel Rechenleistung und Daten für effektives Training.

**AdaBoost (Adaptive Boosting)**

* **Charakteristik:** Ensemble-Methode, die schwache Klassifikatoren kombiniert, um eine starke Gesamtentscheidung zu treffen.
* **Entscheidungsgrenze:** Etwas eckige Trennung, aber deutlich robuster als ein einzelner Entscheidungsbaum.
* **Stärken:** Verbessert schwache Klassifikatoren und funktioniert gut auf moderaten Datensätzen.
* **Schwächen:** Sensitiv gegenüber Ausreißern.

**Naive Bayes (Gaussian Naive Bayes)**

* **Charakteristik:** Basierend auf **Bayes‘ Theorem**, nimmt an, dass Merkmale unabhängig sind.
* **Entscheidungsgrenze:** Glatte, oft ovale oder kreisförmige Trennung.
* **Stärken:** Schnell zu trainieren, funktioniert oft überraschend gut.
* **Schwächen:** Die Annahme der **Merkmalsunabhängigkeit** ist oft nicht realistisch.

**QDA (Quadratic Discriminant Analysis)**

* **Charakteristik:** Erweiterung von Naive Bayes, erlaubt **quadratische** Trennungen.
* **Entscheidungsgrenze:** Starke, geschwungene Muster (besonders sichtbar in der letzten Spalte).
* **Stärken:** Erlaubt nicht-lineare Trennungen.
* **Schwächen:** Funktioniert nicht gut, wenn die Daten nicht den zugrundeliegenden Modellannahmen entsprechen.

**3. Fazit und Interpretation der Ergebnisse**

* **Die besten Algorithmen für nicht-lineare Daten (make\_moons, make\_circles)** sind **RBF SVM, Gaussian Process, Random Forest und Neural Networks**, da sie geschwungene Entscheidungsgrenzen erzeugen.
* **Lineare SVM funktioniert nur für lineare Datensätze** und ist für make\_moons und make\_circles nicht geeignet.
* **Decision Trees und Random Forests neigen zu kantigen Entscheidungsgrenzen**, funktionieren aber trotzdem gut.
* **Probabilistische Methoden wie Naive Bayes und QDA** zeigen eher weiche, geschwungene Entscheidungsgrenzen, sind aber nicht immer die besten.

Die Wahl des besten Klassifikators hängt also stark von der Datenstruktur ab. Während lineare Modelle für einfache Trennungen gut sind, sind komplexe, nicht-lineare Modelle für anspruchsvollere Muster nötig.

### Jonas Maier (Datenbank, Backend)

*Kapitel verfasst von Jonas Maier*

### Jonas Bogensberger (Frontend)

*Kapitel verfasst von Jonas Bogensberger*

## Variantenbildung

Bei der Planung und Umsetzung des Projekts wurden verschiedene Architektur- und Technologievarianten in Betracht gezogen, um eine möglichst effiziente und benutzerfreundliche Lösung zu entwickeln. Dabei spielten Aspekte wie Skalierbarkeit, Performance, Wartbarkeit und Benutzerfreundlichkeit eine zentrale Rolle. Die Auswahl der finalen Variante erfolgte nach einer sorgfältigen Abwägung der Vor- und Nachteile jeder Option.

**Mögliche Varianten:**

* Verwendung einer zentralisierten Server-Architektur oder einer dezentralen Lösung
* Speicherung der Analyseergebnisse in einer relationalen Datenbank (MySQL, PostgreSQL) oder einer NoSQL-Datenbank (MongoDB)
* Einsatz einer lokalen KI-Verarbeitung auf dem Endgerät oder einer serverbasierten KI-Analyse
* Auswahl von vortrainierten KI-Modellen oder Implementierung eines eigenen, speziellen Trainingssystems
* Nutzung einer Webanwendung oder einer mobilen Applikation zur Interaktion mit dem System

Nach eingehender Analyse entschied sich das Projektteam für eine **zentralisierte Server-Architektur mit einer serverbasierten KI-Analyse**. Diese Wahl gewährleistet eine hohe Skalierbarkeit, da die KI-Modelle zentral verwaltet und aktualisiert werden können, ohne dass die Endgeräte der Nutzer über hohe Rechenkapazitäten verfügen müssen.

Für die Speicherung der Analyseergebnisse wurde eine **NoSQL-Datenbank (MongoDB)** bevorzugt. Diese bietet eine flexible Struktur, die sich gut für unstrukturierte oder sich dynamisch ändernde Daten eignet und ermöglicht eine effiziente Verarbeitung der Analyseergebnisse.

Zudem fiel die Entscheidung auf die Entwicklung einer **mobilen Applikation in Kotlin für Android**. Diese Wahl bietet eine optimierte Benutzererfahrung auf mobilen Geräten, ermöglicht den direkten Zugriff auf die Kamera für die Bilderfassung und erlaubt eine nahtlose Interaktion mit der KI-Analyse. Außerdem wurde dieser Punkt vom Projektgeber vorgegeben.

Diese Architektur gewährleistet eine effiziente, benutzerfreundliche und leistungsstarke Lösung, die sowohl die Vorteile einer mobilen App als auch einer serverbasierten KI-Verarbeitung vereint.

## Machbarkeitsstudie

Die Machbarkeitsstudie untersucht, ob das geplante Diplomprojekt innerhalb der gegebenen Rahmenbedingungen technisch und zeitlich realisierbar ist. Dabei wurden sowohl die technischen Voraussetzungen als auch die zur Verfügung stehende Arbeitszeit analysiert. Durch eine frühzeitige Einarbeitung in relevante Technologien konnten mögliche Herausforderungen identifiziert und Lösungsansätze erarbeitet werden.

**Technische Machbarkeit:**

* Die Entwicklung einer mobilen Anwendung in **Kotlin für Android** ist technisch realisierbar und wird durch eine Vielzahl von Entwicklungsressourcen und Bibliotheken unterstützt.
* Die **serverseitige KI-Analyse** kann mithilfe moderner Machine-Learning-Frameworks wie TensorFlow oder PyTorch umgesetzt werden.
* Die Speicherung der Analyseergebnisse in einer **MongoDB-Datenbank** ermöglicht eine flexible und effiziente Verwaltung der Daten.
* Die Anbindung der mobilen Anwendung an den Server erfolgt über standardisierte **REST- oder WebSocket-Schnittstellen**, was eine stabile Kommunikation sicherstellt.
* Die Implementierung verschiedener **KI-Modelle zur Analyse von Hautläsionen** ist möglich, wobei vortrainierte Modelle genutzt oder eigene Modelle trainiert werden können.
* Die Anwendung erfordert eine **stabile Internetverbindung**, um Bilder an den Server zu senden und Ergebnisse zurückzuerhalten.

**Zeitliche Machbarkeit:**

* Jeder Projektteilnehmer hat ca. **180 Stunden** für das Diplomprojekt zur Verfügung.
* Die Arbeit am Projekt erfolgt sowohl **innerhalb der Schulzeit** (Diplomarbeitsstunden) als auch **außerhalb der Schulzeiten** (individuelle und gemeinsame Arbeitstreffen).
* Die Entwicklung gliedert sich in mehrere Phasen:
  + **Vorbereitungsphase**: Einarbeitung in relevante Technologien, Recherche und Planung
  + **Entwicklungsphase**: Umsetzung der mobilen Anwendung, Server-Architektur und KI-Integration
  + **Testphase**: Überprüfung der Funktionalität, Optimierung und Fehlerbehebung
  + **Dokumentation**: Erstellung der Abschlussdokumentation und Vorbereitung auf die Präsentation
* Trotz potenzieller Herausforderungen bei der Entwicklung von KI-Modellen und der Server-Integration wird das Projektteam durch eine klare Aufgabenverteilung und regelmäßige Abstimmung die gesetzten Meilensteine erreichen.

Insgesamt ist das Projekt aus technischer und zeitlicher Sicht gut realisierbar. Die gewählten Technologien sind bewährt und ermöglichen eine effiziente Umsetzung. Durch die geplante Arbeitsverteilung und die Kombination aus Schul- und Freizeitstunden steht ausreichend Zeit zur Verfügung, um das Projekt erfolgreich abzuschließen.

## Allgemeine Planungsinformationen

Neben den bereits behandelten Aspekten der Projektplanung gibt es weitere zentrale Überlegungen, die für den erfolgreichen Ablauf des Projekts von Bedeutung sind. Dieses Kapitel befasst sich mit grundlegenden Planungsinformationen, die nicht in anderen Abschnitten detailliert behandelt wurden, jedoch maßgeblich zur Organisation und Umsetzung beitragen.

### Projektziele

*(Die Grundlegenden Projektziele sind aus den Vorgaben des Projektgebers abgeleitet.)*

Die Zielsetzung dieses Projekts besteht in der Entwicklung einer **leistungsfähigen Machine-Learning-Anwendung** zur **automatischen Klassifikation von Hautläsionen** anhand von Bildern. Dabei wird der **HAM10000-Datensatz** als Grundlage genutzt. Das Projekt umfasst nicht nur die Entwicklung eines geeigneten KI-Modells, sondern auch die vollständige Umsetzung des **Machine-Learning-Life-Cycles** – von der Datenvorbereitung über das Modelltraining bis hin zur Integration in eine mobile Anwendung.

Ein weiteres zentrales Ziel ist die **didaktische Aufbereitung** des Entwicklungsprozesses. Entlang der verschiedenen Phasen sollen **Jupyter Notebooks** erstellt werden, die eine verständliche Einführung in die einzelnen Schritte der Modellentwicklung ermöglichen. Dadurch wird das Projekt nicht nur zur praktischen Lösung eines medizinischen Problems, sondern auch zur wertvollen Lernressource für die Arbeit mit Machine Learning.

Die konkreten Projektziele lassen sich in fünf Kernbereiche unterteilen:

**1. Datenvorbereitung**

* Bereitstellung und Verwaltung der Bilddaten über eine strukturierte oder unstrukturierte **Datenbank**.
* **Datenbereinigung und -augmentation**, um die Qualität und Diversität des Trainingsdatensatzes zu optimieren.
* Optionale Ergänzung des **HAM10000-Datensatzes** durch weitere medizinische Bilddatensätze (z. B. ISIC Challenge).
* Aufteilung der Daten in **Trainings-, Validierungs- und Testdatensätze**, um eine effiziente Modellbewertung zu ermöglichen.

**2. Modellentwicklung**

* Auswahl und Implementierung geeigneter **Machine-Learning-Algorithmen**, insbesondere **Convolutional Neural Networks (CNNs)**.
* Untersuchung verschiedener Ansätze, darunter direkte Klassifikation oder ein **zweistufiger Prozess**, bei dem zunächst die Läsion erkannt und anschließend klassifiziert wird (Bounding Box, Semantic Segmentation etc.).
* **Training und Optimierung der Modelle** auf Basis des HAM10000-Datensatzes durch **Hyperparameter-Tuning**.

**3. Modellbewertung**

* Evaluierung der **Modellleistung** anhand relevanter Metriken wie **Accuracy, Precision, Recall, AUROC und F1-Score**.

**4. Deployment**

* Bereitstellung des trainierten Modells in einer **serverseitigen Umgebung**.
* Entwicklung einer **RESTful API**, um das Modell als Webservice zur Verfügung zu stellen.
* Optional: Implementierung des Modells direkt auf dem mobilen Gerät mit **TensorFlow Lite oder PyTorch Edge**, um die Abhängigkeit vom Server zu reduzieren.

**5. Integration in eine mobile Anwendung**

* Entwicklung einer **mobilen App in Kotlin**, die es Nutzern ermöglicht, **Bilder von Hautläsionen aufzunehmen und klassifizieren zu lassen**.
* Integration der **RESTful API** zur Bereitstellung der Klassifikationsergebnisse in Echtzeit.

Dieses Projekt kombiniert moderne **Machine-Learning-Technologien** mit einer benutzerfreundlichen **mobilen Anwendung**, um einen praktischen Beitrag zur medizinischen Diagnostik zu leisten. Gleichzeitig stellt die durchdachte Planung und Aufbereitung sicher, dass der gesamte Entwicklungsprozess **nachvollziehbar und wiederverwendbar** bleibt.

### Benötigte Ressourcen

Für die erfolgreiche Umsetzung des Projekts sind verschiedene Ressourcen erforderlich, die sowohl die technische als auch die organisatorische Realisierung unterstützen. Dieses Kapitel gibt einen Überblick über die benötigten Ressourcen, die in verschiedene Kategorien unterteilt sind: Hardware, Software, personelle Ressourcen und weitere unterstützende Materialien.

#### Hardware

Diese Diplomarbeit wird sowohl an den privaten Schullaptops diverser Hersteller des Projektteams als auch an Standrechnern zu Hause entwickelt und dokumentiert. Es sind keinerlei externe Hardware oder Mietserver von Nöten.

#### Software

Dadurch, dass im Projekt verschiedenste Technologien und Programmiersprachen verwendet werden, wird auch eine geeignete Softwareunterstützung zur Entwicklung in den jeweiligen Bereichen gefordert. Dazu gehören integrierte Entwicklungsumgebungen, Interpreter, Kompilierer oder auch Textverarbeitungsprogramme. Eine ausführliche Auflistung der verwendeten Programme/Tools ist im Kapitel **6.7 Softwareprogramme / Komponenten** zu finden.

#### Personelle Ressourcen

Das Projektteam arbeitet eigenständig und ohne jegliche Hilfe von außen an dem Projekt. Dafür stellt jedes Teammitglied um die 180 Stunden reine Arbeitszeit dem Projekt zur Verfügung.

#### Weitere Materialien

##### ISIC-Archive

Das **ISIC-Archive** (International Skin Imaging Collaboration) ist eine öffentlich zugängliche Datenbank mit dermatologischen Bildern, die zur Unterstützung der Forschung und Entwicklung im Bereich der Hautkrebs- und Hautläsionserkennung dient. Es enthält eine große Sammlung hochauflösender Bilder verschiedener Hauterkrankungen, darunter Melanome und andere Hautveränderungen. Die Datenbank wird häufig für das Training und die Validierung von KI-gestützten Diagnosemodellen genutzt und dient als Benchmark für medizinische Bildanalyse-Algorithmen.

Diese Datenbank wird als Datenquelle für Trainings- und Validierungsdaten der KI-Komponente verwendet.

### Entwicklungsmethodik

Für die Durchführung dieses Projekts wurde die **agile Entwicklungsmethodik** gewählt, um sicherzustellen, dass flexibel und effizient auf Anforderungen reagiert werden kann und der Projektfortschritt kontinuierlich überprüft wird. Der Fokus liegt auf iterativen Entwicklungszyklen, die eine schnelle Anpassung und Verbesserung des Produkts ermöglichen.

Im Speziellen wurde der Ansatz des **Scrum**-Frameworks verwendet, um das Projekt zu planen, zu steuern und zu überwachen. Scrum ermöglicht es, die Arbeit in überschaubare Iterationen, sogenannte **Sprints**, zu unterteilen. Jeder Sprint dauert in der Regel zwei Wochen und endet mit einer Überprüfung der erreichten Ziele. In der Planung des Projekts wurden alle Anforderungen und Aufgaben in **User Stories** umgewandelt, die dann geschätzt wurden. Die Schätzungen erfolgten sowohl in **Value Points** (für den Nutzen, den jede Story bringt) als auch in **Story Points** (für den Aufwand, der für die Umsetzung erforderlich ist).

Die Aufgaben und Anforderungen wurden im **Product Backlog** gesammelt, wobei für jeden Sprint eine Auswahl an Aufgaben ins **Sprint Backlog** übernommen wurde. Während der Sprintplanung wurde der Schwerpunkt auf die Umsetzung von Features gelegt, die den größten Mehrwert für das Projekt bringen.

Ein wesentlicher Bestandteil der Scrum-Methode ist die kontinuierliche Überprüfung des Fortschritts und das Einholen von Feedback. In diesem Projekt wurde der Fortschritt jedoch nicht nach jedem Sprint in einer Präsentation vor den Lehrpersonen präsentiert. Stattdessen wurden regelmäßige interne Besprechungen abgehalten, um den Fortschritt zu diskutieren und eventuelle Probleme zu identifizieren. Den Lehrpersonen wurde der Fortschritt zu festgelegten Zeitpunkten des Projekts präsentiert, um sicherzustellen, dass das Projekt in die richtige Richtung geht und alle Anforderungen eingehalten werden.

Die agile Methodik stellt sicher, dass das Projektteam jederzeit flexibel auf Änderungen oder neue Anforderungen reagieren kann, wodurch ein transparentes und anpassungsfähiges Projektumfeld geschaffen wird. Durch diesen iterativen Ansatz konnte die Qualität der Ergebnisse kontinuierlich verbessert und das Projekt effizient zum Erfolg geführt werden.

### Kommunikations- und Berichterstattungsstrategie

Im Rahmen dieses Projekts wird der Kontakt zu den Stakeholdern über die Lehrpersonen als zentrale Kommunikationsstelle gepflegt. Die Fortschritte des Projekts werden regelmäßig intern besprochen, und an festgelegten Meilensteinen werden die Lehrpersonen über den aktuellen Stand informiert. Die Berichterstattung erfolgt nicht in Form von regelmäßigen Präsentationen alle zwei Wochen, sondern fokussiert zu wichtigen Projektpunkten, bei denen eine detaillierte Rückmeldung erforderlich ist.

Die Erreichung der Sprintziele wird regelmäßig intern besprochen, wobei insbesondere darauf geachtet wird, welche User Stories im Sprint Backlog abgeschlossen wurden, welche noch offen sind und welche an den nächsten Sprint übergeben werden. Dabei wird auch die Sprint Velocity und der Projektverlauf mithilfe von Burndown-Charts und Säulendiagrammen visualisiert, um eine klare und nachvollziehbare Darstellung des Fortschritts zu gewährleisten.

**Live-Demo:**  
Ein weiteres Element der Kommunikationsstrategie ist die Bereitstellung einer Live-Demo. Diese ermöglicht den Stakeholdern, die neuesten Fortschritte des Projekts direkt zu erleben und gibt ihnen die Möglichkeit, Feedback zu geben. Die Live-Demo wird durch die Lehrperson organisiert, die auch den Kontakt zu den Stakeholdern koordiniert.

**Dokumentation:**  
Die gesamte Projektarbeit wird kontinuierlich in einer Dokumentation festgehalten, die laufend aktualisiert wird. Diese Dokumentation dient nicht nur als interner Bericht, sondern auch als Referenz für die Lehrpersonen und Stakeholder, um jederzeit einen detaillierten Überblick über den Projektverlauf zu erhalten. Sie gewährleistet, dass alle relevanten Informationen zugänglich sind und beantwortet mögliche Fragen.

### Projektrisiko(-bewertung)

Da es sich bei diesem Projekt um eine **Diplomarbeit** handelt, die im schulischen Kontext durchgeführt wird, ist das Risiko im Vergleich zu kommerziellen Projekten in gewisser Weise weniger gravierend. Dennoch gibt es auch hier Risiken, die berücksichtigt und im besten Fall frühzeitig gemanagt werden sollten. Die Risikobewertung erfolgt anhand potenzieller Faktoren, die den Verlauf und die erfolgreiche Umsetzung des Projekts beeinflussen könnten.

#### Mangelnde Verfügbarkeit von Ressourcen

Ein potenzielles Risiko besteht darin, dass möglicherweise nicht ausreichend Ressourcen wie Zeit, benötigte Software, Hardware oder Fachwissen zur Verfügung stehen. Besonders die **Zeit** stellt ein Risiko dar, da die Arbeit parallel zu schulischen Anforderungen durchgeführt wird. Die fehlende Verfügbarkeit von Experten oder spezifischen technischen Ressourcen könnte ebenfalls den Fortschritt verzögern.

**Ursachen:**  
Dieses Risiko kann verschiedene Ursachen haben, beispielsweise durch unvorhergesehene Verzögerungen, wie zusätzliche Zeitaufwände bei der Datensammlung, Modellierung oder dem Training der KI-Algorithmen. Auch technische Schwierigkeiten bei der Implementierung oder der Integration der Lösung in die mobile Anwendung können zusätzliche Ressourcen beanspruchen. Ein weiteres Risiko kann durch unerwartete Änderungen in den Projektanforderungen oder -spezifikationen entstehen.

**Auswirkungen:**  
Ein Mangel an Ressourcen könnte zu verschiedenen negativen Auswirkungen führen, darunter **Verzögerungen** im Zeitplan, eine **verminderte Qualität des Endprodukts** oder sogar eine **unvollständige Umsetzung** des Projekts. Dies könnte insbesondere in einer Diplomarbeit problematisch sein, da das Projekt rechtzeitig abgeschlossen und die Qualität des Endprodukts den Anforderungen entsprechen muss, um den akademischen Abschluss zu sichern.

**Risikobewertung:**  
Die Wahrscheinlichkeit dieses Risikos ist moderat, da es von mehreren Faktoren abhängt, darunter der Komplexität der Modellierung und der Implementierung der mobilen Anwendung, der Verfügbarkeit von benötigten Ressourcen und der Erfahrung des Projektteams. Die Projektarbeit könnte sich aufgrund von externen Faktoren wie Zeitmanagement oder unvorhergesehenen Schwierigkeiten bei der Implementierung verzögern, allerdings wurde durch die Festlegung von **realistischen Zeitrahmen** und **Kontrollen** das Risiko eingedämmt.

**Risikominderungsstrategien:**  
Zur Minimierung dieses Risikos können verschiedene Maßnahmen ergriffen werden:

* **Frühzeitige Planung** der benötigten Ressourcen, sowohl in Bezug auf Zeit als auch Materialien oder externe Unterstützung.
* **Realistische Zeitpläne** erstellen und regelmäßig überprüfen, um sicherzustellen, dass alle Aufgaben rechtzeitig abgeschlossen werden können.
* **Pufferzeiten** für unvorhergesehene technische oder logistische Probleme einplanen.
* Bei Bedarf auf **Zusatzressourcen** zurückgreifen, etwa zusätzliche Literaturquellen, technische Foren oder externe Unterstützung von Experten, etwa durch Online-Communities oder die Lehrpersonen als Ansprechpartner.

**Verantwortlichkeiten:**  
Die Projektverantwortlichen, also die **Projektteilnehmer** und die **Lehrpersonen**, sind dafür verantwortlich, das Risiko kontinuierlich zu überwachen und frühzeitig auf Anzeichen von Ressourcenmangel zu reagieren. Im Falle unerwarteter Herausforderungen oder Verzögerungen müssen geeignete Maßnahmen zur Risikominderung ergriffen und die Ressourcen gegebenenfalls angepasst werden.

## Projektumfeldanalyse

Die Projektumfeldanalyse bietet einen umfassenden Überblick über die verschiedenen Faktoren und Bedingungen, die den Erfolg dieses Projekts beeinflussen können. Sie berücksichtigt sowohl interne als auch externe Aspekte, die das Projekt vor, während und nach der Umsetzung beeinflussen könnten. Ziel dieser Analyse ist es, mögliche Chancen und Risiken zu identifizieren und die notwendigen Voraussetzungen für eine erfolgreiche Durchführung des Projekts zu schaffen.

**Zielsetzung des Projekts:**

Das Hauptziel des Projekts ist die Entwicklung einer mobilen Anwendung, die es den Nutzern ermöglicht, Bilder von Hautläsionen aufzunehmen und diese mithilfe eines Machine Learning Modells zu klassifizieren. Die Aufgabe ist in mehrere Hauptbereiche unterteilt:

* **Datenvorbereitung**
* **Modellentwicklung und -bewertung**
* **Integration in eine mobile Anwendung**

**Anforderungen an das Projekt:**

* **Funktionalität:** Die mobile Anwendung muss auf gängigen mobilen Endgeräten (insbesondere mit Android-Betriebssystem) lauffähig sein und eine benutzerfreundliche Oberfläche bieten. Sie muss in der Lage sein, Bilder zu erfassen und die Bilddaten an den Server zur Klassifikation zu senden.
* **Datenverarbeitung:** Die Anwendung benötigt eine zuverlässige Datenbank zur Speicherung der Bilddaten sowie der Klassifikationsergebnisse.
* **Zuverlässigkeit und Skalierbarkeit:** Das System muss skalierbar sein, um mit einer zunehmenden Menge von Nutzerdaten umgehen zu können. Es sollte auch in der Lage sein, Anfragen in Echtzeit zu verarbeiten.
* **Dokumentation:** Alle durchgeführten Schritte und Modellentwicklungen müssen dokumentiert werden, um eine nachvollziehbare Entwicklungsgeschichte sicherzustellen.

**Technologische Anforderungen:**

* **Mobile Entwicklung:** Die Anwendung wird mit Kotlin für Android entwickelt, um eine hohe Performance und nahtlose Integration auf mobilen Geräten zu gewährleisten.
* **Machine Learning Modell:** Ein tiefes neuronales Netzwerk (CNN) wird für die Klassifikation der Hautläsionen eingesetzt, trainiert auf einem umfangreichen Datensatz (HAM10000).
* **Backend und API:** Eine RESTful API wird benötigt, um das Machine Learning Modell in einer serverseitigen Umgebung bereitzustellen und die Kommunikation zwischen der mobilen Anwendung und dem Server zu ermöglichen.
* **Datenbank:** Eine NoSQL-Datenbank (wie MongoDB) wird zur Speicherung der Bilddaten und Klassifikationsergebnisse verwendet.

**Rahmenbedingungen:**

* **Projektzeitrahmen:** Das Projekt wird als Teil einer Diplomarbeit durchgeführt, und der Zeitrahmen ist durch das Ende des Schuljahres und die Abgabefrist der Diplomarbeit begrenzt.
* **Ressourcen:** Die notwendigen Ressourcen wie mobile Geräte (Android Smartphones), Computer für Modelltraining und Entwicklungsumgebungen werden durch die Schule bereitgestellt. Auch die Datensätze und Modellressourcen sind durch öffentliche Quellen zugänglich.
* **Expertise:** Das Projektteam besteht aus Schülern mit unterschiedlichen Kenntnissen in den Bereichen Machine Learning, Mobile Entwicklung und Datenbankmanagement. Eine kontinuierliche Unterstützung durch die Lehrperson ist für das Gelingen des Projekts von entscheidender Bedeutung.

**Externe Faktoren:**

* **Verfügbarkeit von Hardware und Software:** Die Nutzung von Open-Source-Bibliotheken für Machine Learning sowie die Verfügbarkeit der Datensätze über öffentliche Repositorien sind wesentliche Faktoren, die den Projekterfolg positiv beeinflussen. Zudem muss sichergestellt werden, dass die verwendete Software und die Entwicklungstools aktuell und kompatibel sind.
* **Technische Unterstützung:** Bei technischen Herausforderungen und der Implementierung können Online-Communities wie Stack Overflow oder spezialisierte Foren eine wertvolle Unterstützung bieten.
* **Risiken:** Unvorhergesehene technische Schwierigkeiten, wie z.B. Probleme bei der Modellintegration in die mobile Anwendung, können den Fortschritt verzögern. Ebenso können Engpässe bei der Ressourcenverfügbarkeit, wie z.B. mangelnde Zeit oder fehlende technische Geräte, das Projekt beeinflussen.

**Wettbewerbsfaktoren:**

* **Innovationspotenzial:** Es gibt verschiedene bestehende Projekte, die sich mit der Klassifikation von Hautläsionen befassen. Um sich von anderen Projekten abzuheben, wird in diesem Projekt die Integration des Machine Learning Modells in eine mobile App als innovativer Ansatz verfolgt. Diese Integration ermöglicht eine benutzerfreundliche und leicht zugängliche Lösung für die Hautläsionsklassifikation.
* **Benchmarking:** Ähnliche Projekte, die Machine Learning zur medizinischen Bildklassifikation einsetzen, können als Benchmarks für die Bewertung des eigenen Fortschritts und der angewandten Methoden dienen.

**Projektmanagement:**

* **Teamarbeit:** Eine klare Aufgabenverteilung und effektive Kommunikation sind entscheidend, um das Projekt termingerecht abzuschließen. Regelmäßige Teammeetings und der Austausch von Ideen und Fortschritten stellen sicher, dass alle Teammitglieder auf dem gleichen Stand sind.
* **Zeitplanung:** Die Meilensteine und Sprints des Projekts müssen klar definiert werden, um den Fortschritt zu überwachen und das Projekt innerhalb des vorgesehenen Zeitrahmens zu beenden.
* **Koordination mit der Lehrperson:** Der regelmäßige Austausch mit der Lehrperson stellt sicher, dass das Projekt auf dem richtigen Weg bleibt und wertvolles Feedback während der Entwicklung einfließt.

**Rechtliche Aspekte:**

* **Lizenzen:** Alle verwendeten Softwarebibliotheken und -tools müssen unter einer geeigneten Lizenz stehen, die den rechtlichen Anforderungen entspricht.
* **Datenschutz:** Da die mobile Anwendung Nutzerdaten (insbesondere Bilddaten) verarbeiten wird, müssen Datenschutzbestimmungen beachtet werden. Eventuell sind zusätzliche Maßnahmen wie eine Anonymisierung der Daten oder die Implementierung von Login-Systemen erforderlich, um die Privatsphäre der Nutzer zu gewährleisten.
* **Sicherheit:** Besonders bei der Übertragung von Bilddaten über Netzwerke müssen Sicherheitsaspekte wie Verschlüsselung berücksichtigt werden, um Missbrauch zu vermeiden.

Diese Projektumfeldanalyse stellt sicher, dass alle relevanten Faktoren für die erfolgreiche Durchführung des Projekts berücksichtigt werden. Sie dient als Grundlage für die Planung und Umsetzung und hilft dabei, potenzielle Risiken zu identifizieren und geeignete Maßnahmen zu ergreifen.

# Softwarearchitektur

*Kapitel verfasst von Daniel Jessner*

Ein fundiertes Verständnis der Softwarearchitektur ist unerlässlich, um die Funktionsweise eines Systems vollständig zu begreifen und eine effiziente Entwicklung sowie Wartung zu gewährleisten. In diesem Kapitel wird die Architektur der entwickelten Software detailliert vorgestellt. Es werden die verschiedenen Systemkomponenten und deren Interaktionen erläutert, um aufzuzeigen, wie sie zusammenarbeiten, um die angestrebte Funktionalität zu erreichen. Dabei wird besonders auf die Trennung der verschiedenen Softwareebenen und deren Kommunikation eingegangen, um ein ganzheitliches Bild der Systemstruktur zu vermitteln. Ein besonderer Fokus liegt auf der Strukturierung der KI-Komponente, der mobilen Anwendung sowie der dazugehörigen Backend-Architektur. Um diese komplexen Beziehungen klar und verständlich zu machen, werden die relevanten Architekturkomponenten durch Diagramme veranschaulicht, die die Zusammenhänge und Abläufe visuell verdeutlichen.

Das folgende Kapitel beleuchtet die grundlegenden Softwarekomponenten des Systems:

* **Mobile Anwendung (Android, Kotlin)**
* **Backend-Server (AdonisJS, Python)**
  + **Backend API**
* **KI-Komponente(Python, Tensorflow, Pytorch, Scikit-Learn)**
  + **Gateway-API (FastAPI)**
* **Datenbank (MongoDB)**

Durch diese detaillierte Darstellung wird ein klarer Überblick darüber gegeben, wie die einzelnen Teile des Systems zusammenspielen und miteinander kommunizieren, um die geplante Lösung zu realisieren.

## Aktivitätsdiagramme

Aktivitätsdiagramme sind ein wesentliches Werkzeug in der Softwaredokumentation, um Arbeitsabläufe und Prozesse innerhalb eines Systems grafisch darzustellen. Sie helfen dabei, die Sequenz von Aktivitäten und die logischen Abläufe in einer Anwendung zu visualisieren. Dieses Kapitel enthält die relevanten Aktivitätsdiagramme für das dokumentierte Projekt, die detailliert die verschiedenen Prozesse und deren Abläufe illustrieren.

## Sequenzdiagramme

Sequenzdiagramme sind genauso wie die Aktivitätsdiagramme ein wesentliches Werkzeug in der Softwaredokumentation zur Darstellung von Interaktionen zwischen Systemkomponenten im zeitlichen Verlauf. Dieses Kapitel enthält die relevanten Sequenzdiagramme für unser Projekt, die detailliert die Kommunikation und Abläufe zwischen den verschiedenen Systemelementen veranschaulichen.

Folgende Sequenzdiagramme wurden mit dem Tool von **sequencediagram.org** erstellt.

## Komponentendiagramm

Dieses Komponentendiagramm dient dazu, die strukturelle Organisation des Systems zu visualisieren und die Beziehungen zwischen seinen Komponenten aufzuzeigen. Es bietet einen ganzheitlichen Überblick über die einzelnen Bausteine des Systems und ihre Interaktionen.

**Beschreibung:**

Das Komponentendiagramm zeigt die verschiedenen technischen Komponenten des Systems sowie deren Kommunikationswege und Schnittstellen. Jede Komponente übernimmt eine spezifische Aufgabe und interagiert mit seinem Kommunikationspartner über definierte Schnittstellen. Das System besteht aus einem Client, einer Datenbank, einem Backend zur Datenverarbeitung und einem KI-Backend (Gateway) zur Modellierung und Vorhersage.

**Komponenten:**

**1. Client-Anwendung**

* **Technologie:** Kotlin (Android)
* **Beschreibung:** Die Client-Anwendung ist die Benutzeroberfläche des Systems. Sie ermöglicht es den Nutzern, mit dem System zu interagieren, Daten einzugeben und Ergebnisse abzurufen.
* **Schnittstelle:** Kommuniziert über **HTTP** mit dem Backend.

**2. Datenbank-Backend**

* **Technologie:** AdonisJS, Python
* **Beschreibung:** Das Database-Backend dient als Schnittstelle zur Speicherung und Bereitstellung von Benutzer- und Trainingsdaten für KI-Modelle. Es empfängt Daten vom Client und speichert diese in der MongoDB.
* **Schnittstellen:**
  + **HTTP:** Kommunikation mit dem Client über das Gateway.
  + **MongoDB:** Speicherung und Verwaltung von Benutzerdaten/Trainingsdaten.

**3. MongoDB (NoSQL-Datenbank)**

* **Technologie:** MongoDB
* **Beschreibung:** Diese NoSQL-Datenbank speichert alle relevanten Benutzerdaten sowie Trainingsdaten für das KI-Modell. Sie wird vom Datenbank-Backend verwaltet und dient als zentrale Datenquelle.
* **Schnittstellen:** Verknüpft mit dem Datenbank-Backend zur Speicherung und Bereitstellung von Daten.

**4. KI-Backend**

* **Technologie:** Python
* **Beschreibung:** Das KI-Backend ist für das Training von Modellen und das Erstellen von Vorhersagen verantwortlich. Es verarbeitet die Daten aus der Datenbank und stellt Analysen zur Verfügung. Außerdem dient es als Gateway zwischen Frontend und Datenbank-Backend.
* **Schnittstellen:**
  + **HTTP:** Kommunikation mit dem Client und Datenbank-Backend.
  + **PORT 8000:** API für Frontend
  + **PORT 6969:** Automatischer Bildzuschnitt (Resize Image).
  + **PORT 3333:** Datenbank-Backend

**Systemgrenzen:**

Das System umfasst interne Komponenten, die über definierte Schnittstellen miteinander kommunizieren. Die Client-Anwendung, das Datenbank-Backend und die Datenbank bilden zusammen das zentrale System, während das KI-Backend die Funktion zur Modellierung und Vorhersage darstellt und als Gateway dient.

Komponentendiagramm:

Ein Bild, das Text, Diagramm, Plan, technische Zeichnung enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Verteilungsdiagramm

Das Verteilungsdiagramm zeigt an, wie die einzelnen Teile der Software auf die Hardwarekomponenten verteilt sind und wie die Hardwarekomponenten miteinander verbunden sind. Sprich: Auf welchem Rechner läuft welche Software und wie sind diese übers Netzwerk miteinander verbunden. Es ist dem Komponentendiagramm sehr ähnlich.

**Komponenten:**

* **Device (Mobile) – Kotlin Android:**
  + Beschreibung: Die mobile Anwendung dient als Benutzer-Frontend, über das Nutzer mit dem System interagieren können. Sie sendet Anfragen über HTTP an das Backend.
* **AI-Backend – Python:**
  + Beschreibung: Diese Komponente verarbeitet KI-bezogene Anfragen, trainiert Modelle und führt Vorhersagen durch. Sie empfängt Daten über HTTP und gibt die verarbeiteten Ergebnisse zurück.
* **DB-Backend – AdonisJS, FastAPI:**
  + Beschreibung: Diese Schicht fungiert als Schnittstelle zur Datenbank und bietet APIs zur Speicherung und Bereitstellung von Daten. AdonisJS verwaltet Backend-Logik, während FastAPI für schnelle API-Anfragen genutzt wird.
* **DB-Server – MongoDB:**
  + Beschreibung: Die NoSQL-Datenbank speichert Benutzerdaten, Trainingsdaten und andere relevante Informationen, die von den Backend-Diensten benötigt werden.

**Interaktionen:**

* Die mobile Anwendung kommuniziert über HTTP (Port 8000) mit dem AI-Backend.
* Das AI-Backend kommuniziert über HTTP (Port 8000) mit dem DB-Backend, um notwendige Daten abzurufen oder zu speichern.
* Das DB-Backend verwaltet die Daten in der MongoDB-Datenbank und stellt sie dem AI-Backend zur Verfügung.

**Systemgrenzen:**  
Das Diagramm beschreibt die Verteilung der Software-Komponenten auf die jeweiligen Hardware-Ressourcen und veranschaulicht die Kommunikation zwischen diesen Komponenten im System.

Verteilungsdiagramm:

Ein Bild, das Text, Diagramm, Rechteck, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.

## C4-Modell

## Ablaufdiagramm(e)

## Softwareprogramme / Komponenten

Ein erfolgreiches Softwareprojekt erfordert eine durchdachte Auswahl und den gezielten Einsatz geeigneter Programme, Softwarekomponenten und Frameworks. In diesem Kapitel werden die für die Entwicklung des Projekts verwendeten Tools und Technologien angegeben. Dazu gehören sowohl die Entwicklungsumgebungen und unterstützenden Programme als auch die spezifischen Frameworks und Bibliotheken, die für die Umsetzung der Softwarearchitektur essenziell sind.

### Softwareprogramme

Im Laufe der Entwicklung wurden einige Tools/Programme verwendet, um das Projekt beziehungsweise seine Dokumentation zu bearbeiten. Diese sind hier aufgelistet:

|  |  |  |  |
| --- | --- | --- | --- |
| **Programm** | **Version** | **Hersteller** | **Beschreibung** |
| PyCharm | 2023.3.2 | JetBrains | Entwicklungsumgebung für Python |
|  |  |  |  |
|  |  |  |  |
| Github Desktop + git | 3.3.13 (x64) | Github, Inc. | Versionskontroll-Software zur Code-Sicherung |
| Microsoft Word | / | Microsoft | Textvearbeitungsprogramm |

### Softwarekomponenten

Die unten gelisteten Softwarekomponenten sind die Sprachen beziehungsweise Frameworks welche zur Realisierung des Projektes verwendet wurden.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SW-Komponent** | **Version** | **Hersteller** | **Bezugsquelle** | **SW-Lizenz** | **Beschreibung** |
| Python | 3.11.0 | Python Software Foundation | https://www.python.org | / | Programmiersprache |
| Tensorflow Keras |  |  |  |  |  |
| Pytorch |  |  |  |  |  |
| Scikit-Learn |  |  |  |  |  |

# Projektdurchführung

*Kapitel verfasst von Daniel Jessner*

## Daniel Jessner (KI-Komponente, Gateway)

*Kapitel verfasst von Daniel Jessner*

## Jonas Maier (Datenbank, Backend)

*Kapitel verfasst von Jonas Maier*

## Jonas Bogensberger (Frontend)

*Kapitel verfasst von Jonas Bogensberger*

# Projektabschluss

*Kapitel verfasst von Daniel Jessner*

## Installation / Software Deployment

# Literaturverzeichnis

*Kapitel verfasst von Daniel Jessner*

# Abbildungsverzeichnis

*Kapitel verfasst von Daniel Jessner*