Der TitelPflichtenheft und technische Spezifikation im   
Programmierprojekt

**Datenanalyse mit neuronalen Netzwerken**

Mitarbeiter:

Lukas Evers

Anderson Lewis Orock Soh Talla

Paul Schult

Versionshistorie

|  |  |  |
| --- | --- | --- |
| Datum | Bearbeiter | Durchgeführte Änderung |
| 20.10.20 | PS | Erstellung |
| 18.11 | LE | GUI-Mock-Up eingefügt  Use-Case-Diagramm eingefügt |
| 19.11 | PS | Akivitäten-, Kompenenten-, Klassendiagramm, interne Schnittstellen, Projektfeatures, Risiken, Test und Implementierung, Visionen und Ziele |
| 19.11 | LE | Abbildungs- und Tabellenbeschriftungen, Testsetup-Komponentdiagramme einfügen, GUI-Bilder erklärend beschriftet |

Inhaltsverzeichnis

[1 Visionen und Ziele 1](#_Toc5699678)

[2 Anforderungen an Ihr System 1](#_Toc5699679)

[2.1 Use-Cases 1](#_Toc5699680)

[2.2 Aktivitäten-Diagramme 1](#_Toc5699681)

[2.3 GUI 1](#_Toc5699682)

[3 Realisierung 2](#_Toc5699683)

[3.1 Allgemeines 2](#_Toc5699684)

[3.2 Interne Schnittstellen 2](#_Toc5699685)

[3.3 Visual-Studio-Projektsetup 2](#_Toc5699686)

[3.4 Externe Schnittstellen 3](#_Toc5699687)

[4 Test und Implementierungsphase 3](#_Toc5699688)

[5 Planung 3](#_Toc5699689)

[6 Lizenz 4](#_Toc5699690)

# Visionen und Ziele

Ziel des Projektes ist die Erstellung einer graphischen Desktop-Anwendung, die Mithilfe eines Modells zum maschinellen Lernen aus skalaren Daten typische Klassifikationsaufgaben bewältigen kann. Dabei sollen beliebige csv-Dateien eingelesen werden können. Der Nutzer kann so Trainings- und Testdaten einlesen, die zum Trainieren und Validieren des Machine Learning Modells verwendet werden. Die Anwendung zeigt dem Nutzer die Test Ergebnisse an und stellt umfangreiche statistische Bewertungen zur Verfügung. Sollte das Modell den Ansprüchen des Nutzers genügen, kann das Modell gespeichert bzw. für neue Datensätze verwendet werden.

Das Machine Learning Modell kann für Klassifikationsaufgaben verwendet werden.

Tabelle 1 Feature-ID, Zuständigkeit und Feature Beschreibung

|  |  |  |
| --- | --- | --- |
| Feature-ID | Zuständigkeit | Beschreibung |
| F.100 | Paul | Desktop-Anwendung |
| F.200 | Lewis | Datenverarbeitung |
| F.201 | Lewis | CSV Dateien einlesen |
| F.202 | Paul | Datenbereich und Zielspalte bestimmen |
| F.203 | Lewis | CSV Daten in internes Datenformat umwandeln |
| F.204 | Lewis | Ergebnis in CSV Datei ausgeben |
| F.300 | Lukas | Neuronales Netzwerk trainieren |
| F.400 | Lukas | Neuronales Netzwerk testen |
| F.401 | Lukas | Ergebnis-Sicherheit bestimmen |
| F.402 | Lukas | Neuronales Netzwerk speichern |
| F.500 | Lukas | Neuronales Netzwerk einsetzen |
| F.5z1 | Paul | Ergebnis auf dem Bildschirm ausgeben |
| F.600 | Paul | Ergebnis Auswerten |
| F.601 | Paul | Auswertung für binäre Klassifikation |
| F.602 | Paul | Auswertung für „multiclass“ Klassifikation |

# Anforderungen an Ihr System

## Use-Cases

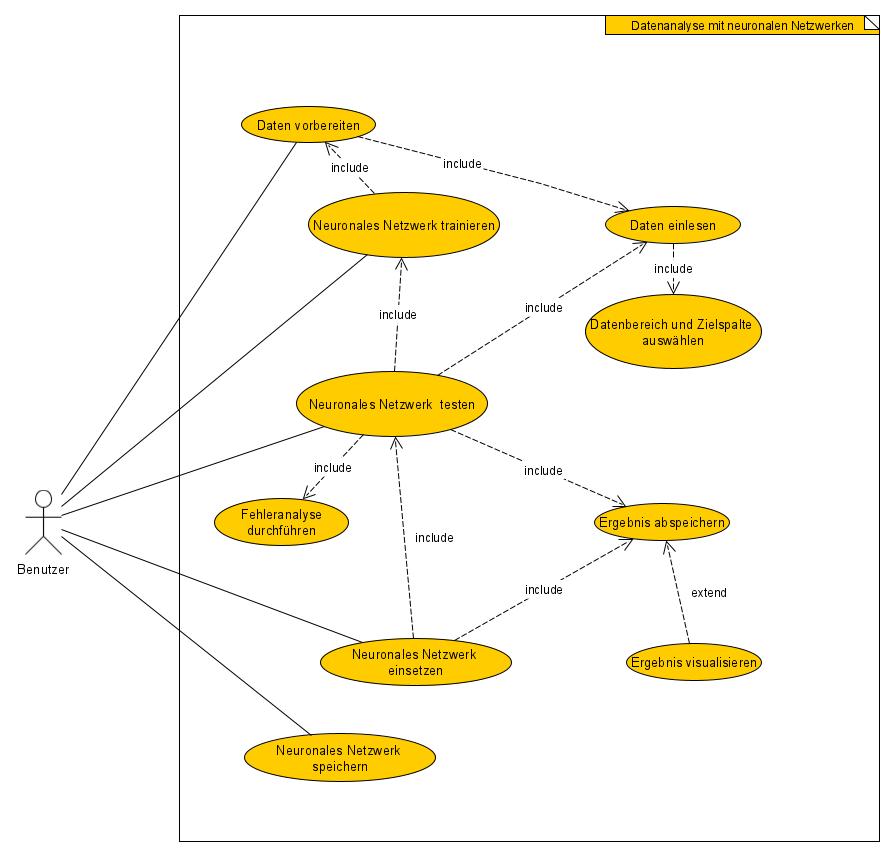


Abbildung 1 Use-Case-Diagramm

## Aktivitätendiagramm

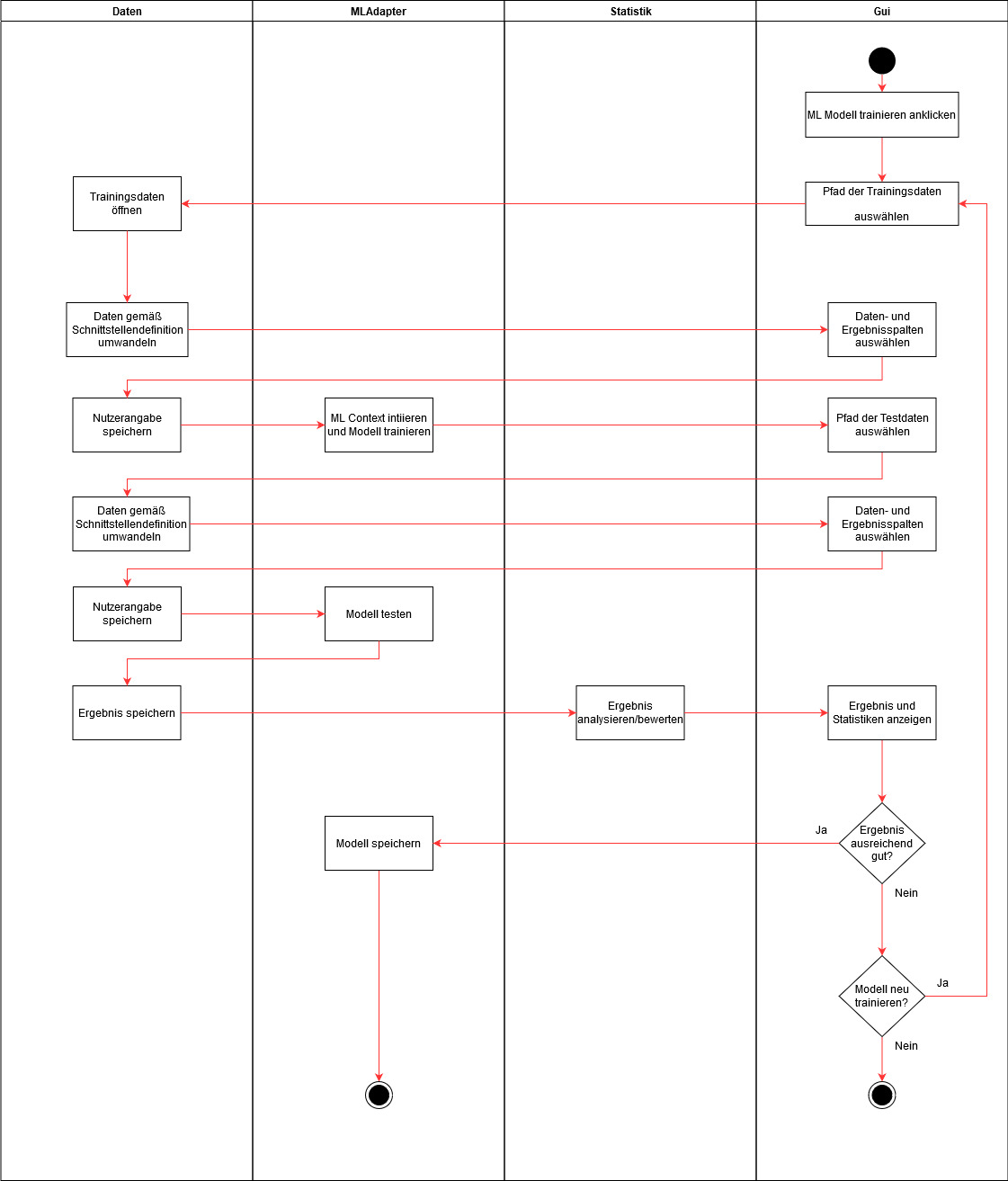


Abbildung 2 Aktivitätendiagramm

## Risiken

Risiken:

* Während Phase 2 und 3 auftauchende Komponentenübergreifende Änderungen am Projekt
* Erstes Projekt dieser Größe
* Fehlende Erfahrung mit externer Schnittstelle ML.NET und WPF
* Nicht alle Projektmitarbeiter führen das Projekt bis zum Ende durch
* Nichteinhaltung von Terminen
* Unterschiedliche Interpretationen und Wissensstände im Projekt

## GUI

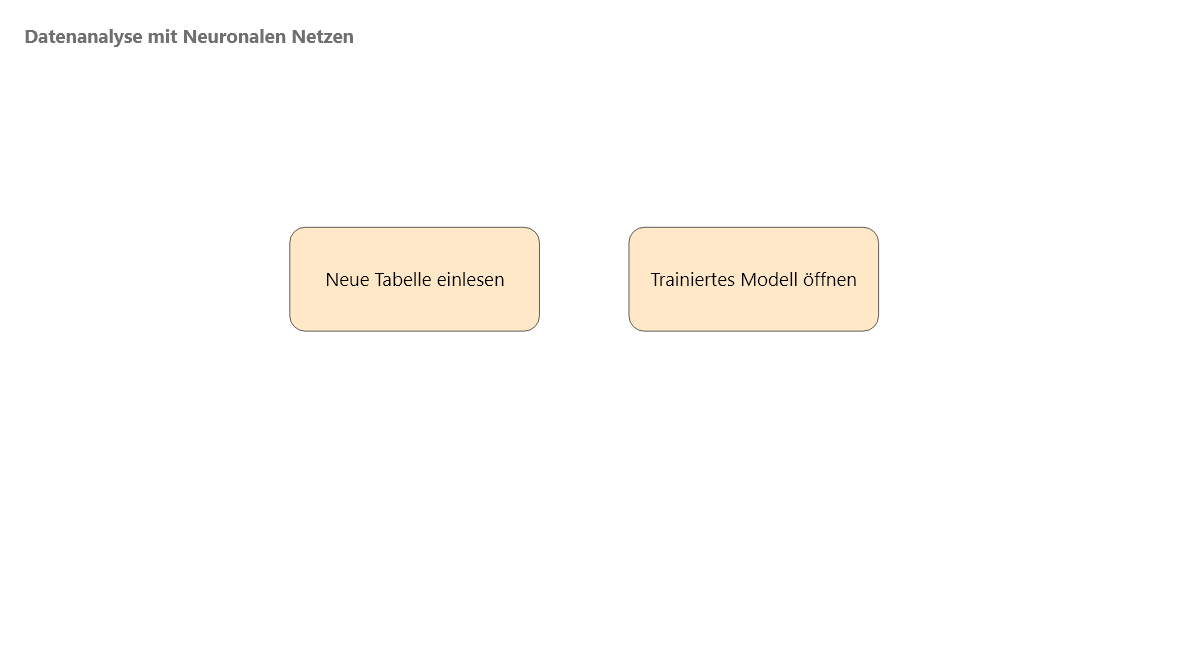


Abbildung 3: Homescreen, wenn noch kein Modell trainiert oder geladen ist

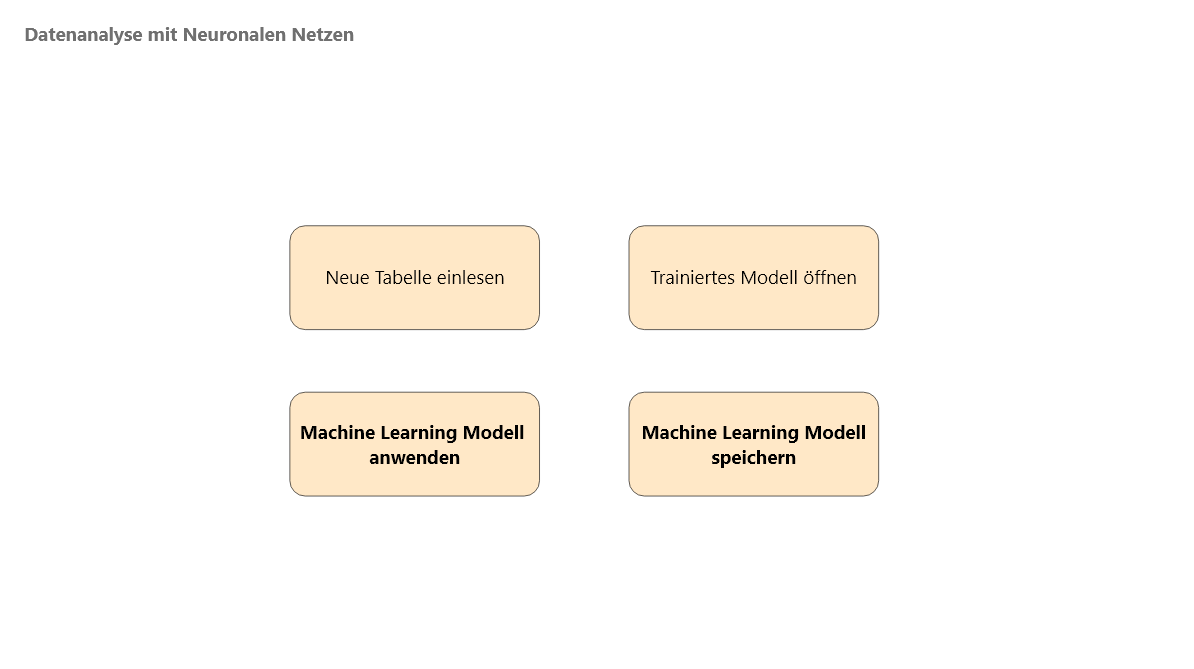


Abbildung 4: Homescreen, nachdem ein Model geladen oder trainiert wurde

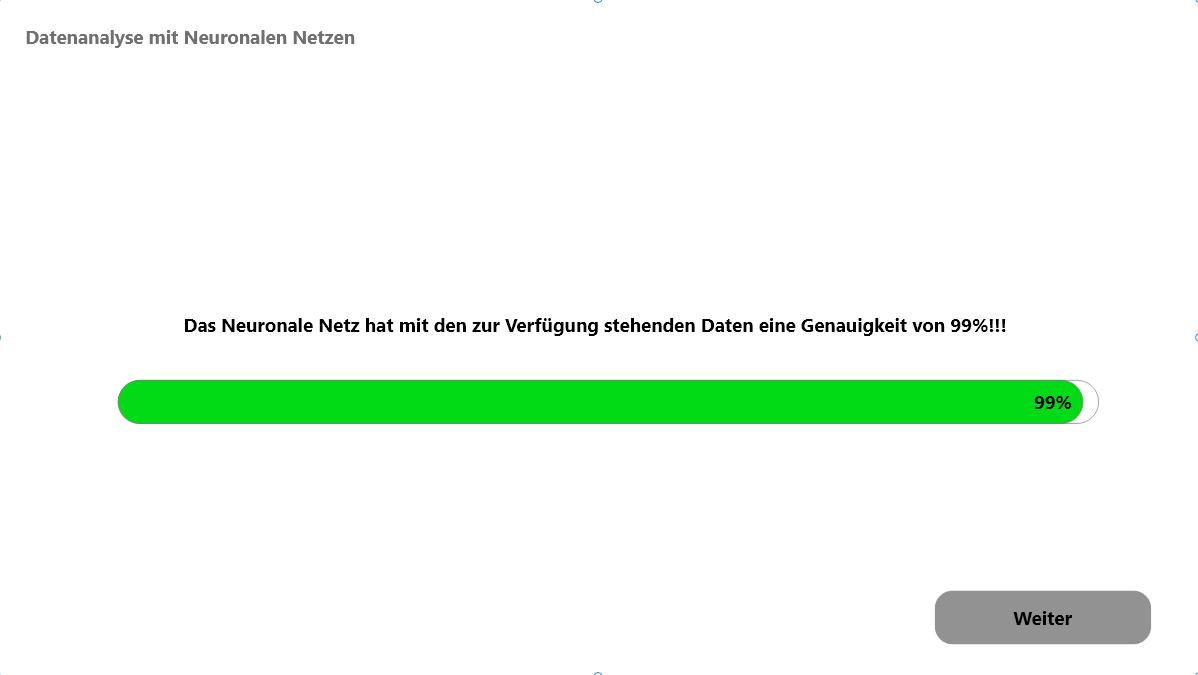


Abbildung 5: Ergebnisanzeige nach dem Trainieren des ML Modells, z.B. auch durch Verwechslungsmatrix

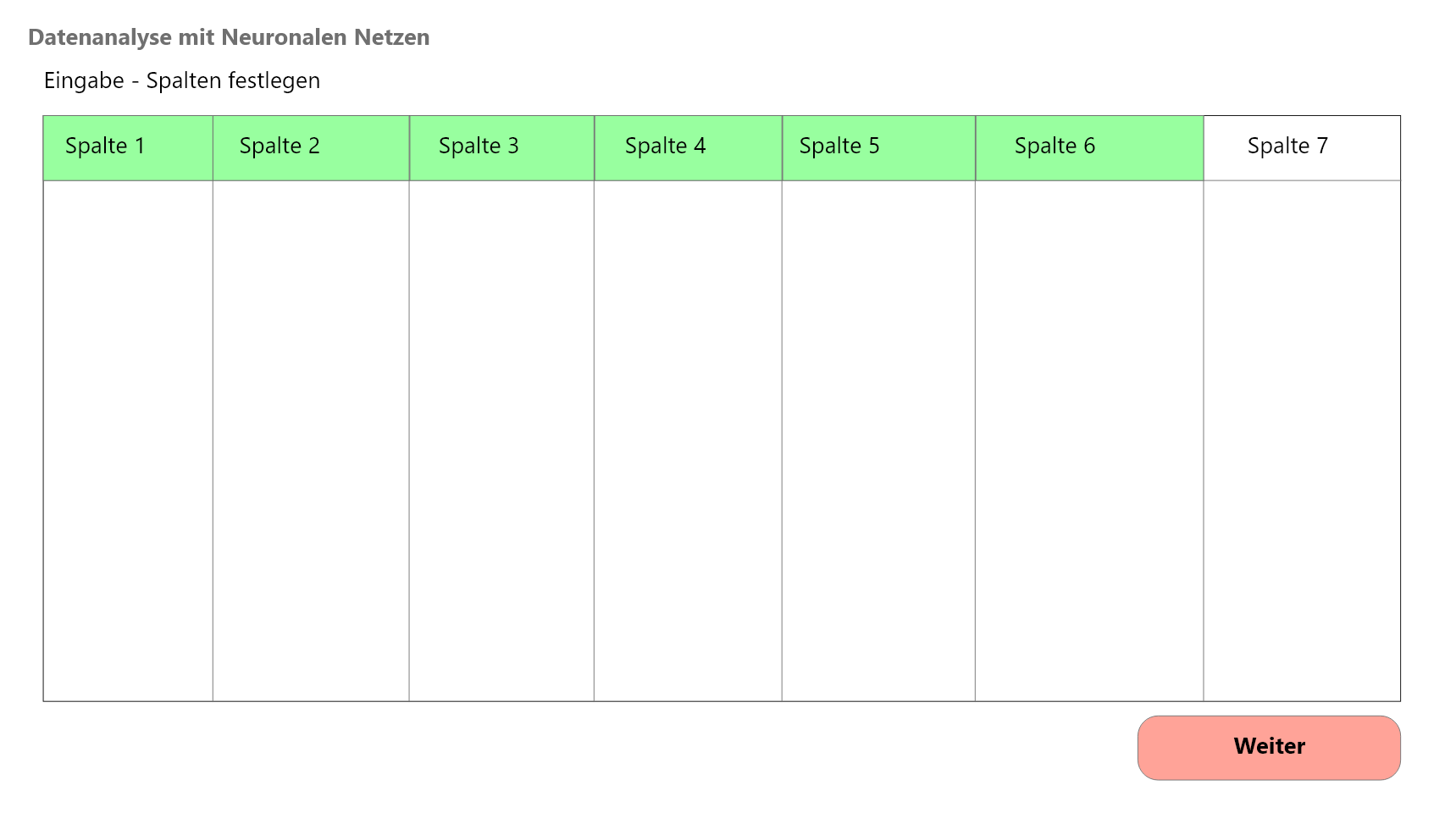


Abbildung 6: Festlegen der Eingabespalten, für Training und Verwendung des Modells

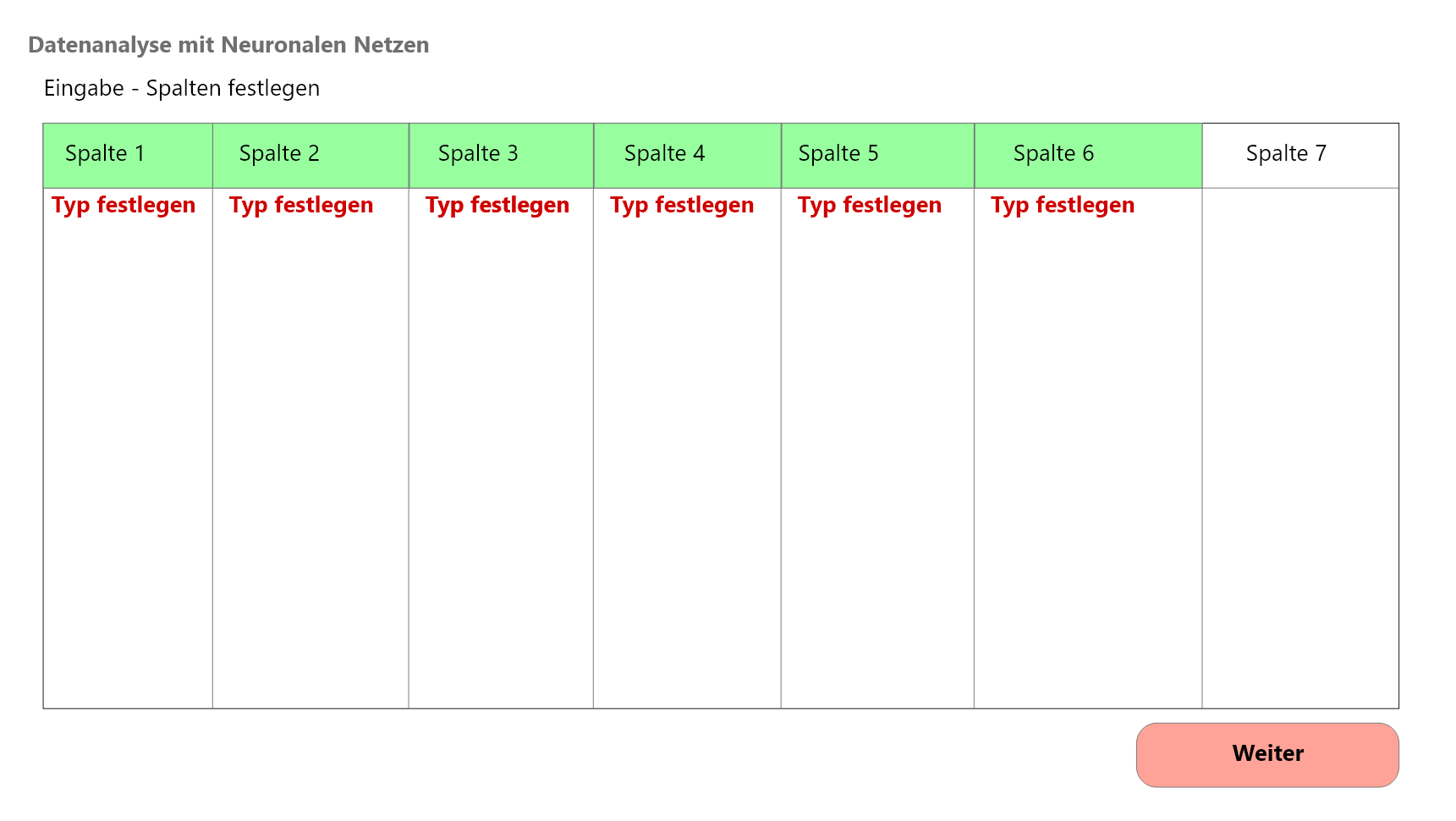


Abbildung 7: Festlegen in welchem Typ die Daten in den Eingabespalten vorliegen

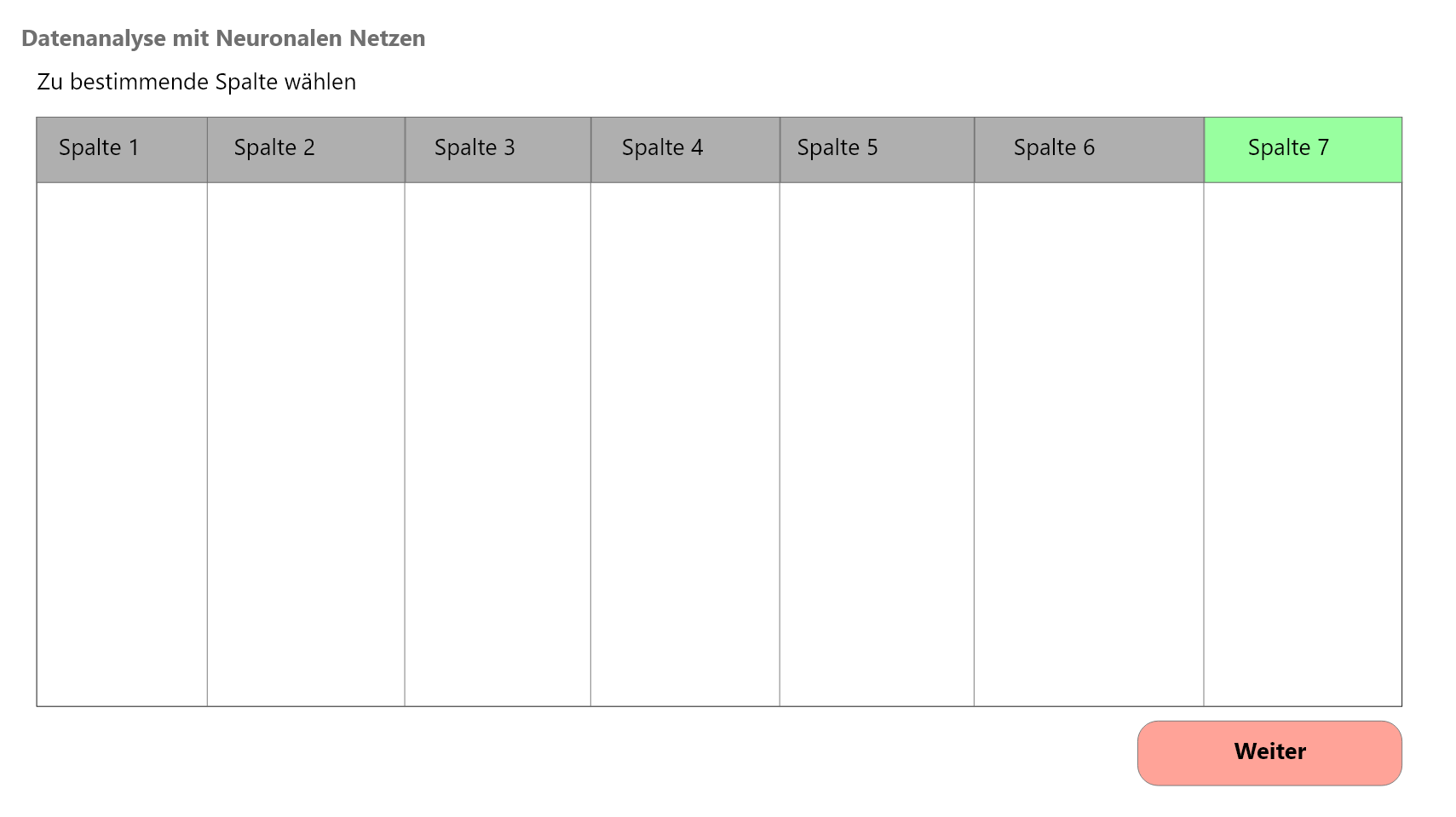


Abbildung 8: Vorherzusagende Spalte auswählen

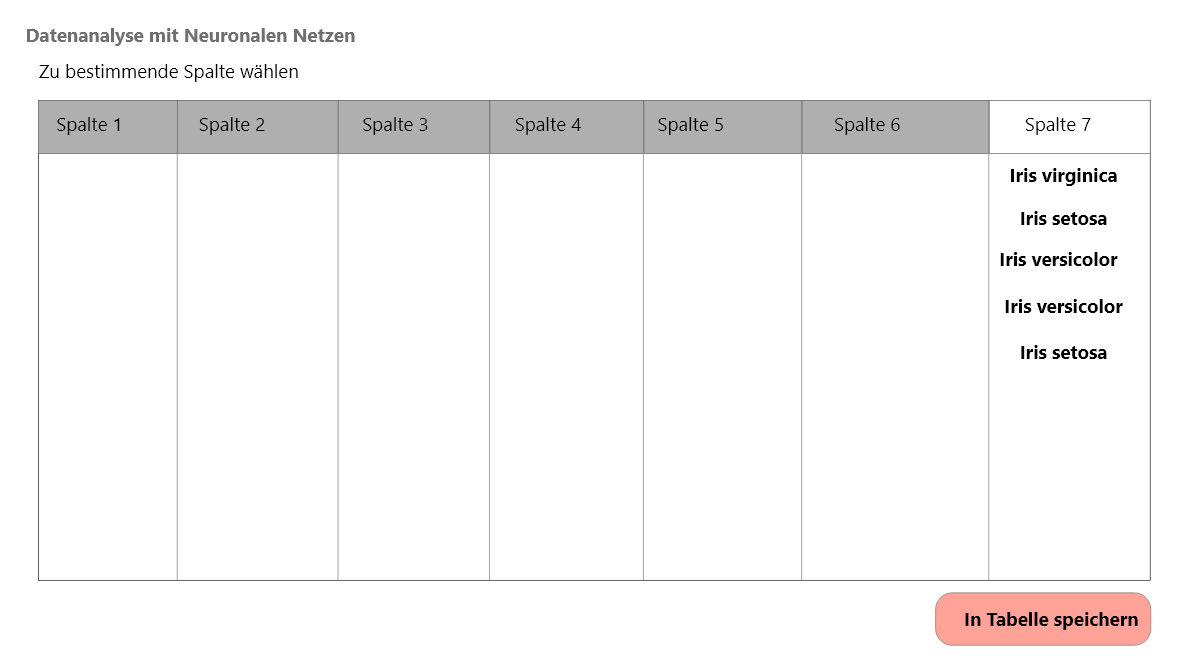


Abbildung 9: Anhand der Daten in den Eingabespalten wurden durch das zuvor trainierte Modell bestimmt, Option diese Daten zu exportieren

# Realisierung

## Allgemeines

Das Projekt wurde Mithilfe des Use-Case-Diagramms in unterschiedliche Komponenten aufgeteilt. Zu beachten ist, dass die Programmlogik größtenteils in der GUI enthalten ist. Aus dem Kompentendiagramm wurde anschließend das Klassendiagramm abgeleitet.

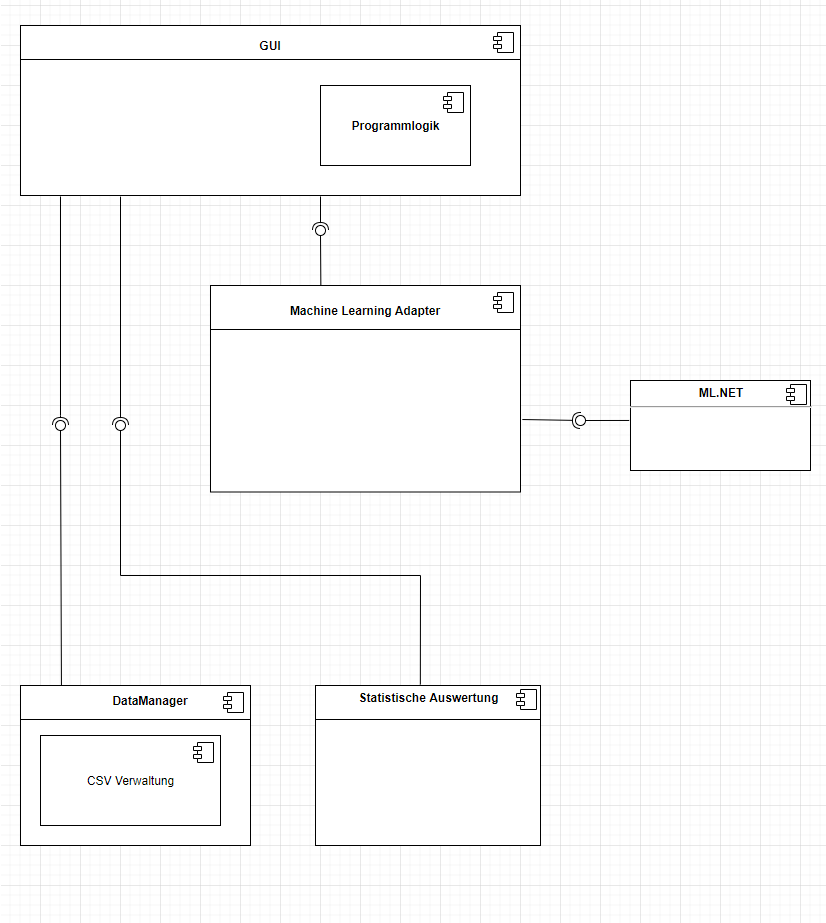


Abbildung 10: Komponenten Diagramm

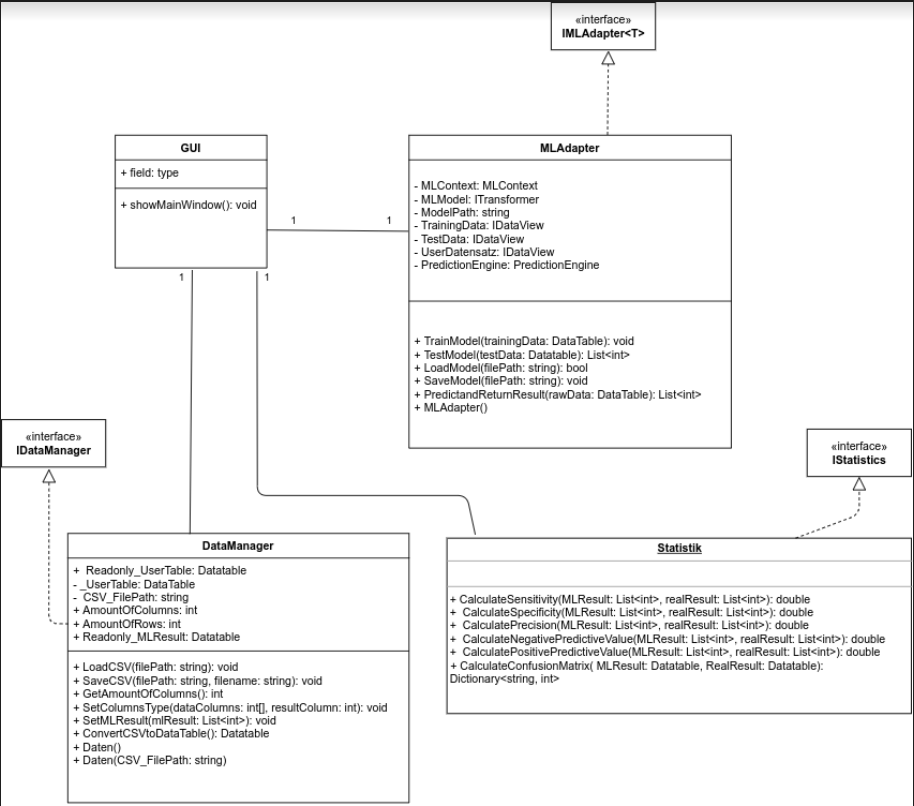


Abbildung 11: Klassendiagramm

Tabelle 2 Kompenenten und Zuständigkeit

|  |  |
| --- | --- |
| Komponente | Zuständigkeit |
| GUI | Paul |
| MLAdapter | Lukas |
| DataManager | Lewis |
| Statistics | Paul |

## Interne Schnittstellen

Die internen Schnittstellen wurden aus dem Klassendiagramm abgeleitet. Daraus ergeben die folgend aufgelisteten Schnittstellen.

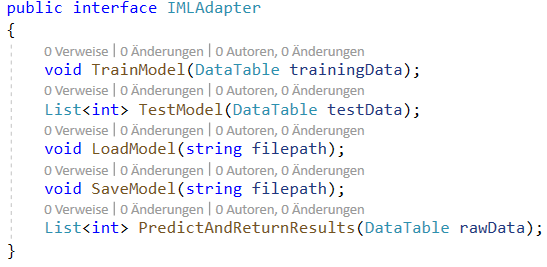


Abbildung 12: Interface IMLAdapter

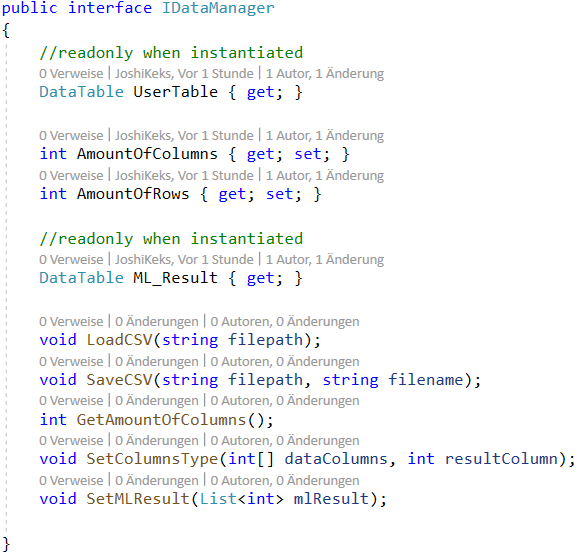


Abbildung 13: Interface IDataManager

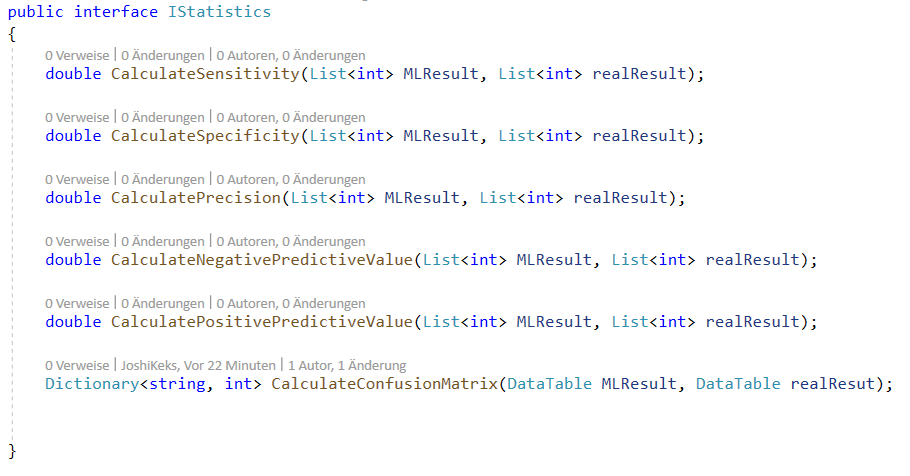


Abbildung 14: Interface IStatistics

## Visual-Studio-Projektsetup

Das Projekt wurde unter Berücksichtigung des Klassendiagramms in drei Teilprojekte gegliedert. Jedes Teilprojekt wird eigenständig programmiert. Hierfür wurden auf dem GIT Repository drei verschiedene Projekte erstellt. In jedem Projekt sind die eigene Komponente sowie alle über das CommonInterface verwendete Komponenten als Dummy enthalten. Projekte, die keine GUI implementieren haben eine Dummy GUI als Konsolenanwendung erhalten. Das vollständige Projektsetup ist in dem Projekt GIT Repository hinterlegt.

## Technologiestack

Das Projekt verwendet .NET Framework 4.7.2. Die grafische Benutzeroberfläche wird mit dem Windows Presentation Foundation Framework (WPF) realisiert.

Tabelle 3: Begründung der Technologiewahl

|  |  |
| --- | --- |
| Technologie | Begründung |
| C# 7 | Vorgegeben durch Studiengang/Modul |
| .NET Framework 4.7.2 | Für grafische Benutzeroberfläche besser als .NET Core |
| Windows Presentation Foundation Framework (WPF) | Moderne grafische Benutzeroberfläche, dynamischer und funktionsreicher als Windows Forms |
| ML.NET | Gute Dokumentation, leichter Einstieg, open Source |
| Visual Studio 2019 | Modernste IDE für .NET Entwicklung |

## Externe Schnittstellen

Tabelle 4: Externe Schnittstellen

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Realisiert in Adapter | Herausgeber |
| Microsoft.ML/ML.NET | Bibliothek | MLAdapter | Microsoft |
|  |  |  |  |
|  |  |  |  |

# Test und Implementierungsphase

In der Test- und Implementierungsphase ist jeder Entwickler allein verantwortlich für seine Komponenten und Dummies.

Jeder Programmierer hat seine Abhängigkeiten von anderen Komponenten mithilfe des Komponentendiagramms ermittelt. Auf dieser Basis wurden Dummy-Komponenten erstellt. Stellt eine Dummy-Komponente eine Schnittstelle bereit, so wurde diese dort implementiert. Zuerst als NotImplementedException um auch ohne vollständige Implementierung ein lauffähiges Programm zu erzeugen. Anschließend wurden diese simpel und mit vereinfachter Logik ausimplementiert. Beispielweise soll hier jeder mögliche Rückgabewert ausgeben werden (bzw. Corner-Cases).

Nutz eine Dummy-Komponente die eigene Komponente, so wurden Test-Aufrufe erzeugt, die die Funktionalität und Logik der Rückgabewerte überprüfbar macht.

Soweit möglich und absehbar, kann die jeweilige Komponente vollständig auf seine Funktionalität hin überprüft werden. Es wurden keine automatisierten Unit-Tests erstellt. Die Tests sind manuell auszuführen.

## GUI und Statisics

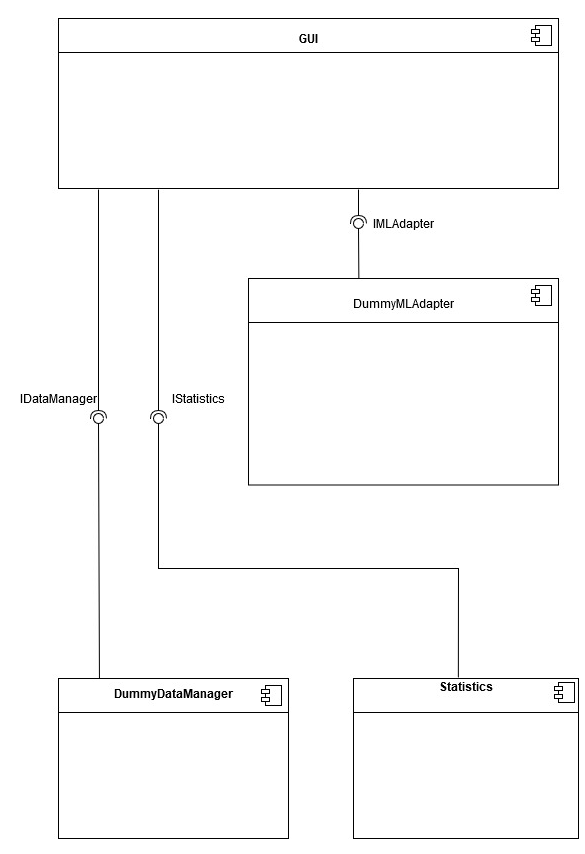


Abbildung 15 Komponentendiagramm aus Sicht von GUI und Statistics

## DataManager

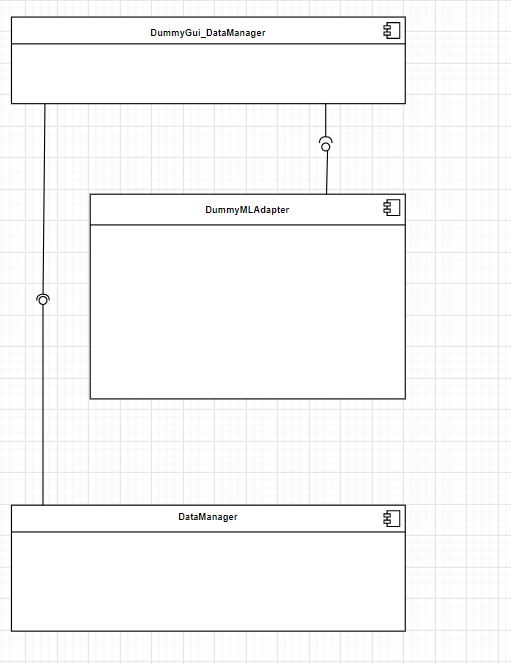


Abbildung 16: Komponenten Diagramm des DataManagers

## MLAdapter

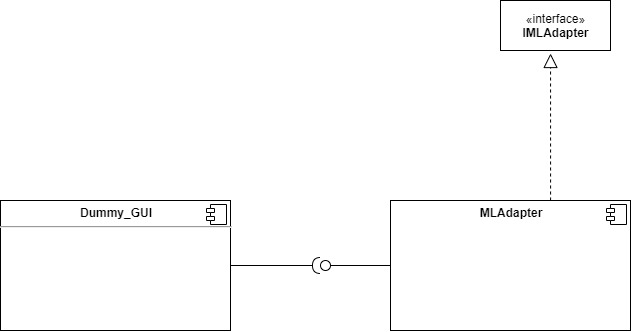


Abbildung 17: Testsetup des MLAdapters

## Einführung von Change-Requests

In der Risikoabwägung haben wir gesondert auf das nicht abschätzbare Risiko der Komponentenübergreifenden Änderungen des Projektes hingewiesen. Deswegen führen wir ein Change-Request-System ein. Im Git Hauptverzeichnis ist dafür eine Datei erstellt worden. Hier trägt jeder die seiner Meinung nach nötigen Änderungen mit Datum und Priorität ein. Bei dem nächsten internen Gruppenmeeting wird gemeinsam über das Change-Request entschieden. Im Optimalfall wird die Änderung noch während des Meetings umgesetzt. So stellen wir sicher, dass die Änderungen im Konsens umgesetzt wird und jeder die Auswirkung auf seine Komponente vollumfänglich versteht.

# Lizenz

Copyright (c) <2020 > < ML.Students (Anderson Lewis Orock Soh Talla, Lukas J. Evers, Paul J. Schult >

Jedem, der eine Kopie dieser Software und der zugehörigen Dokumentationsdateien (die "Software") erhält, wird hiermit kostenlos die Erlaubnis erteilt, ohne Einschränkung mit der Software zu handeln, einschließlich und ohne Einschränkung der Rechte zur Nutzung, zum Kopieren, Ändern, Zusammenführen, Veröffentlichen, Verteilen, Unterlizenzieren und/oder Verkaufen von Kopien der Software, und Personen, denen die Software zur Verfügung gestellt wird, dies unter den folgenden Bedingungen zu gestatten:

Der obige Urheberrechtshinweis und dieser Genehmigungshinweis müssen in allen Kopien oder wesentlichen Teilen der Software enthalten sein.

DIE SOFTWARE WIRD OHNE MÄNGELGEWÄHR UND OHNE JEGLICHE AUSDRÜCKLICHE ODER STILLSCHWEIGENDE GEWÄHRLEISTUNG, EINSCHLIEßLICH, ABER NICHT BESCHRÄNKT AUF DIE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT, DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK UND DER NICHTVERLETZUNG VON RECHTEN DRITTER, ZUR VERFÜGUNG GESTELLT. DIE AUTOREN ODER URHEBERRECHTSINHABER SIND IN KEINEM FALL HAFTBAR FÜR ANSPRÜCHE, SCHÄDEN ODER ANDERE VERPFLICHTUNGEN, OB IN EINER VERTRAGS- ODER HAFTUNGSKLAGE, EINER UNERLAUBTEN HANDLUNG ODER ANDERWEITIG, DIE SICH AUS, AUS ODER IN VERBINDUNG MIT DER SOFTWARE ODER DER NUTZUNG ODER ANDEREN GESCHÄFTEN MIT DER SOFTWARE ERGEBEN.