**Fachliches Konzept für das Projekt: KI-gestützte Vorhersage der Wahlpflichtfachnoten**

1 Einleitung und Zielsetzung

1.1 Hintergrund

Für Studierende stellt die Auswahl geeigneter Wahlpflichtfächer eine große Herausforderung und eine strategische Entscheidung dar, da diese ihren zukünftigen Berufsweg beeinflussen könnte. Mit diesem Projekt sollen Studierende befähigt werden, anhand ihrer fachlichen und persönlichen Präferenzen, Neigungen und Pflichtfachnoten eine einfache, datengetriebene Entscheidung zur Auswahl von vier geeigneten Wahlpflichtfächern zu treffen. Das Ergebnis dient als unverbindlicher Vorschlag und bietet den Studierenden lediglich eine Orientierungshilfe, um ihre Erfolgswahrscheinlichkeit in Wahlpflichtfächern besser einzuschätzen.

1.2 Zielsetzung

* **Prognosesystem:** Ein KI-gestütztes System soll Vorhersagen zu den für einen Studierenden[[1]](#footnote-1) geeigneten Wahlpflichtfächern geben.
* **Grafische Benutzeroberfläche (GUI):** Eine benutzerfreundliche Webanwendung wird entwickelt, um die Präferenzen, Matrikelnummer und das Geburtsdatum eines Studierenden einzugeben. Außerdem werden die Ergebnisse nach der Eingabe und KI-Prognose hier angezeigt.
* **Effizienz und Datenintegration:** Eine sichere, genaue und schnelle Datenverarbeitung von Noten- und Profildaten ist unabdingbar.

2 Anforderungsanalyse

Es gibt hier zwei Benutzersichten, nämlich die Administrationssicht und die Nutzersicht (Studierendensicht).

**Administrationssicht:**

1. Datenimport, Datenauswahl und Bereinigung:

* **Import von historischen Daten:** Ein Programm zum regelmäßigen Import neuer Notendaten – in der Regel einmal pro Semester – sowohl von Pflichtfächernoten als auch von Wahlpflichtfächernoten aus externen Dateien (z.B. CSV, Excel, JSON etc.), um die Datenbank zu ergänzen oder zu aktualisieren.
* **Datenauswahl:** Geeignete Daten, die eine wichtige Rolle beim Trainieren des Modells spielen, werden aus den verfügbaren Daten identifiziert und ausgewählt.
* **Datenbereinigung:** Um die Qualität des KI-Modells zu sichern, werden die Daten vor dem Trainieren des Modells bereinigt (Preprocessing). Dieser Prozess beinhaltet die Entfernung ungültiger Werte oder die Ersetzung von Null-Werten durch gültige Werte (Noten), sowohl für Pflichtfächer als auch für Wahlpflichtfächer.
* **Verwaltung historischer** **Daten:** Dabei geht es darum, die Daten von Studierenden aus vorherigen Semestern zu speichern und zu sichern, um ein wiederholtes Modelltraining zu ermöglichen.

1. Modelltraining und Verwaltung:

* **KI-Algorithmusauswahl:** Hier wird eine explorative Datenanalyse (EDA) durchgeführt, um einen geeigneten KI-Algorithmus, z.B. Random Forest Regression oder andere Regressionsmodelle, auswählen zu können.
* **Hyperparameter-Tuning**: Um die Vorhersagegenauigkeit und die Qualität des Modells optimal zu erhalten, spielen Machine-Learning-Methoden wie Cross-Validation und Grid Search eine zentrale und unverzichtbare Rolle. Diese dienen zur Auswahl der besten Parameter und somit des besten Modells.
* **Speicherung und Versionierung:** Trainierte Modelle werden gespeichert, um diese bei der Vorhersage zu nutzen. Zu Beginn jedes Semesters wird ein Modell trainiert, getestet, gespeichert und versioniert. Das Modell kann z.B. *Name\_des\_Modells\_WS202425.pkl* benannt werden.

1. Visualisierungen:

* **Datenvisualisierung und -analyse:** nalyse der Daten durch Diagramme und Statistiken, z. B. Box-Plot, um Datenausreißer und -verteilungen grafisch darzustellen.
* **Modellmetriken:** Die Qualität und Genauigkeit des Modells werden anhand von eingebauten Fehlermaßmethoden, wie z. B. R², Mean Absolute Error (MAE) oder Mean Squared Error (MSE), geprüft.

**Nutzersicht:**

1. Authentifizierung und Dateneingabe:

* **Login und Nutzervalidierung:** Studenten werden anhand ihrer eingegebenen Matrikelnummer und ihres Geburtsdatums authentifiziert. Dies stellt sicher, dass nur befugte Personen (immatrikulierte Studenten) Zugriff auf das KI-System haben.
* **Dateneingabevalidierung:** Die Eingaben der Studenten werden verifiziert, um die Integrität der Datenbank sicherzustellen und Schäden auszuschließen (z. B. SQL-Injection-Prävention).

1. Notenabfrage und -verarbeitung:

* **Pflichtfachnotenabruf:** Pflichtfachnoten werden anhand der gültigen Dateneingabe aus einer Datenbank abgerufen.
* **Präferenzeneingabe:** Der Student gibt seine persönlichen und fachlichen Präferenzen ein.
* **Vorhersageprozess:** Die aus der Datenbank abgerufenen Daten (Pflichtfachnoten) und Präferenzen werden als Input für das trainierte und validierte KI-Modell zur Berechnung der Wahlpflichtfächernoten verwendet.

1. Ergebnisdarstellung:

* **Tabelle und Grafiken:** Die Ausgabe des aus dem KI-Modell stammenden Ergebnisses wird in Tabellenform dargestellt. Dies wird durch Grafiken ergänzt.
* **Individualisierte Empfehlung:** uf Basis der Vorhersage werden dem Nutzer vier Wahlpflichtfächer empfohlen. Diese Empfehlung gilt als Vorschlag und ist nicht verpflichtend.

3. Technische Umsetzung und Architektur

3.1 Systemarchitektur

Die Systemarchitektur besteht aus den folgenden Bausteinen:

1. Frontend (Nutzersicht)

* **HTML:** Die Hypertext Markup Language (HTML) wird verwendet, um die Grundstruktur der Webseiten zu definieren.
* **CSS & Bootstrap:** Um die Benutzeroberfläche benutzerfreundlich und dynamisch zu gestalten, werden Cascading Style Sheets (CSS) und Bootstrap verwendet.

1. Backend (Flask API)

* **Flask-Server:** Um die Benutzeranfragen, Datenbankabfragen und das KI-Modell zentral zu verwalten, wird der Flask-Server (Python) genutzt.
* **REST-API:** Die REST-API wird verwendet, um eine Schnittstelle zu bilden, die Datenbank- und Vorhersageaufrufe zwischen Frontend und Backend organisiert.

1. Datenbank (SQLAlchemy):

* **Datenhaltung** **Eine SQL-Datenbank wird verwendet, um historische Nutzerdaten wie Wahlpflichtfächernoten und Pflichtfächernoten zu speichern.**
* **Datenbankoptimierung:** Um die Abfragegeschwindigkeit der Datenbank zu optimieren, werden Schritte wie die Indizierung der Matrikelnummer vorgenommen.

1. KI-Modell (Python scikit-learn):

* **Preprocessing-Schritte verwalten:** Preprocessing-Methoden wie MinMaxScaler, MaxAbsScaler, StandardScaler, IterativeImputer, KNNImputer etc. werden für eine weitere Nutzung bei neuer Dateneingabe gespeichert.
* **Modellverwaltung:** Trainierte und auf Richtigkeit und Genauigkeit getestete Modelle wie RandomForestRegressor, GradientBoostingRegressor, Support Vector Regressor, KNeighborsRegressor und AdaBoostRegressor werden mit ihren optimal ausgewählten Parametern (mithilfe von Methoden wie Grid Search, Cross Validation etc.) verwaltet.

3.2 Datenmodellierung

3.2.1 Tabellenstruktur:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Student*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *matrikelnummer* | Integer (PK) | Eine eindeutige Identifikationsnummer des Studenten | 5236028, 5594047, 5307180 |
| *semester\_id* | String (FK) | Erstes Immatrikulationssemester des Studenten (Verweis auf *semester\_id* in Tabelle *Semester)* | WS2425 |
| *datum\_geburt* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* | 2002-04-20 |
| *geburtsort* | String (FK) | Verweis auf *stadt\_id* in Tabelle *Stadt* | F |
| *studiengang\_id* | String (FK) | Verweis auf *studiengang\_id* in Tabelle *Studiengang* | WiInfo |
| *nutzerkennung* | String | Eine eindeutige Kennung bestehend aus vier Buchstaben und zwei zahlen | ACEG24 |
| *vorname* | String | Vorname des Studenten | Paul |
| *nachname* | String | Nachname des Studenten | Schmidt |
| *praeferenzen* | Text | Persönliche und fachliche Präferenzen | Programmierung,  Mathematik,  Malen, Kochen, Spielen |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Modul*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *modul\_id* | String (PK) | Eindeutige Identifikation des Moduls | GISH, KSW |
| *leiter\_id* | Integer (FK) | Verweis auf *leiter\_id* in Tabelle *Leiter* | 5523, 4164, 3221 |
| *ist\_pflichtfach* | Boolean | Gibt an, ob das Modul ein Pflichtfach ist | True, False |
| *modul\_name* | String | Name des Moduls | Datenbanken, Einführung in die Programmierung |
| *pruefungsart* | String | Art der Prüfung | Schriftlich, mündlich, Projektarbeit, Seminararbeit |
| *kreditpunkte* | Integer | Anzahl der Kreditpunkte des Moduls | 2, 3, 5, 6, 12 |
| *angebotshaeufigkeit* | String | Häufigkeit, mit der das Modul angeboten wird | WS, SS, JS |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Semester*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *semester\_id* | String (PK) | Eine eindeutige ID des Semesters | SS2024, WS2425 |
| *semester\_beginn* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* | 2024-04-01 |
| *semester\_ende* | Date (FK) | Verweis auf *datum* in Tabelle *Tag\_to* | 2024-03-31 |
| *semester\_abkuerzung* | String | Abkürzung des Semester-Namens | SS |
| *semester\_bezeichnung* | String | Textbeschreibung des Semesters | Sommersemester 2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Studiengang*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *studiengang\_id* | String (PK) | Eindeutige ID des Studiengangs | WiInfo |
| *leiter\_id* | Integer (FK) | Verweis auf *leiter\_id* in Tabelle *Leiter* | 2152 |
| *fachbereich\_id* | String (FK) | Verweis auf *fachbereich\_id* in Tabelle *Fachbereich* | MND |
| *bewerbungsfrist* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* | 2024-07-15 |
| *studiengang\_bezeichnung* | String | Bezeichnung des Studiengangs | Wirtschaftsinformatik |
| *abschluss* | String | Abschluss des Studiengangs | Bachelor, Master |
| *anzahl\_kreditpunkte* | Integer | Gesamtanzahl der Kreditpunkte für den Studienabschluss | 180, 210, 240 |
| *dauer* | Integer | Regelstudienzeit in Semestern | 3, 4, 6, 7, 8 |
| *Modulhand­­­buch* | String (URL) | Link zum Modulhandbuch | <https://modulhandbuch.xyz> |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Leiter*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *leiter\_id* | Integer (PK) | Eindeutige ID des Kursleiters | 2342 |
| *fachbereich\_id* | String (FK) | Verweis auf *fachbereich\_id* in Tabelle *Fachbereich* | MND |
| *buero\_id* | String (FK) | Verweis auf *buero\_id* in Tabelle *Buero* | B2 0.02 |
| *geburtsdatum* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* für das Geburtsdatum des Kursleiters | 1960-05-06 |
| *eintrittsdatum* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* für das Eintrittsdatum des Leiters | 2001-05-01 |
| *geburtsort* | String | Verweis auf *stadt\_id* in Tabelle *Stadt* | FD |
| *nutzerkennung* | String | Eindeutige Kennung des Leiters | ABCD89 |
| *vorname* | String | Vorname des Leiters | Peter |
| *nachname* | String | Nachname des Leiters | Hohmann |
| *telefonnummer* | String | Telefonnummer des Leiters | +49641 309 2431 |
| *raumort* | String | Stadt, in der sich das Büro befindet | Friedberg |
| *e\_mail\_Adresse* | String | E-Mail-Adresse des Leiters | [Peter.Hohmann@mni.thm.de](mailto:Peter.Hohmann@mni.thm.de) |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Fachbereich*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *fachbereich\_id* | String (PK) | Eindeutige ID des Fachbereichs | MND |
| *leiter\_id* | Integer (FK) | Verweis auf *leiter\_id* in Tabelle *Leiter* | 5523 |
| *fachbereich\_bezeichnung* | String | Bezeichnung des Fachbereichs | FB 13 – MND – Mathematik, Naturwissenschaften und Datenverarbeitung (Fb) |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Buero*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *buero\_id* | String (PK) | Eindeutige ID des Büros | B2 0.02 |
| *gebaeude\_id* | String (FK) | Verweis auf *gebaeude\_id* in Tabelle *Gebaeude* | B1 |
| *gebaeude\_name* | String | Bezeichnung des Gebäudes | B1 Gebäude |
| *groesse* | Integer | Fläche des Büros in Quadratmetern | 40 |
| *kapazitaet* | Integer | Maximale Kapazität des Büros | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Raum*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *raum\_id* | String (PK) | Eindeutige ID des Raumes | A4.0.19 |
| *gebaeude\_id* | String (FK) | Verweis auf *gebaeude\_id* in Tabelle *Gebaeude* | A1 |
| *raum\_nummer* | Integer | Interne Raumnummer | 19 |
| *flaeche* | Integer | Fläche des Raumes in Quadratmetern | 180 |
| *kapazitaet* | Integer | Maximale Anzahl an Personen, die der Raum fasst | 75 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Gebaeude*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *gebaeude\_id* | String (PK) | Eindeutige ID des Gebäudes | B1 |
| *stadt\_id* | String (FK) | Verweis auf *stadt\_id* in Tabelle *Stadt* | FFM |
| *baudatum* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* für Baudatum des Gebäudes | 1992-04-27 |
| *gebaeude\_bezeichnung* | String | Bezeichnung des Gebäudes | B1 Gebäude |
| *flaeche* | Integer | Fläche des Gebäudes in Quadratmetern | 2400 |
| *anzahl\_etagen* | Integer | Anzahl der Stockwerke des Gebäudes | 3 |
| *anzahl\_raeume* | Integer | Anzahl der verfügbaren Räume im Gebäude | 12 |
| *anzahl\_bueros* | Integer | Anzahl der verfügbaren Büros im Gebäude | 26 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Stadt*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *stadt\_id* | String (PK) | Eindeutige Kennung der Stadt | F |
| *land\_id* | String (FK) | Verweis auf *land\_id* in Tabelle *Land* | DE |
| *stadt\_bezeichnung* | String | Name der Stadt | Frankfurt am Main |
| *bevoelkerungsdichte* | Float | Bevölkerungsdichte in der Stadt (Einwohner pro km²) | 3124 |
| *anzahl\_bevoelkerung* | Integer | Einwohnerzahl der Stadt | 775790 |
| *flaeche* | Float | Fläche der Stadt in Quadratkilometern | 248.3 |
| *hoehe* | Integer | Höhe der Stadt über dem Meeresspiegel in Metern | 112 |
| *buerger\_meister* | String | Name des Bürgermeisters | Mike Joseph |
| *laengengrad* | Float | Geokoordinate (Längengrad) der Stadt | 8.682127 |
| *breitengrad* | Float | Geokoordinate (Breitengrad) der Stadt | 50.11092 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Land*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *land\_id* | String (PK) | Eindeutige Landes-ID | DE |
| *kontinent\_id* | String (FK) | Verweis auf die *kontinent\_id* in Tabelle *Kontinent* | EU |
| *staatsangehoerigkeit* | String | Nationalität der Bevölkerung | Deutsch |
| *land\_bezeichnung* | String | Name des Landes | Deutschland |
| *anzahl\_bevoelkerung* | Integer | Einwohnerzahl des Landes | 84480000 |
| *flaeche* | Float | Fläche des Landes in Quadratkilometern | 357592 |
| *hauptstadt* | String | Name der Hauptstadt des Landes | Berlin |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Kontinent*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *kontinent\_id* | String (PK) | Eindeutige ID des Kontinents | EU |
| *kontinent\_name* | String | Name des Kontinents | Europa |
| *anzahl\_bevoelkerung* | Integer | Einwohnerzahl des Kontinents in Millionen | 742,3 Millionen |
| *flaeche* | Float | Fläche des Kontinents in Quadratkilometern | 10530000 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Tag*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *datum* | Date (PK) | Datum (YYYY-MM-DD) | 2018-04-03 |
| *monat\_id* | String (FK) | Verweis auf *monat\_id* in Tabelle *Monat* | Apr |
| *wochentag\_bezeichnung* | String | Bezeichnung des Wochentags | Montag |
| *wochentag\_nummer* | Integer | Nummer des Wochentags | 1 |
| *anzahl\_stunden* | Integer | Anzahl der Stunden an diesem Tag | 24 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Monat*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *monat\_id* | String (PK) | Eindeutige Monats-ID | Jan, Feb, Dez |
| *quartal\_id* | String (FK) | Verweis auf *quartal\_id* in Tabelle *Quartal* | Q1, Q2, Q3, Q4 |
| *monat\_bezeichnung* | String | Name des Monats | Januar |
| *monat\_nummer* | Integer | Monat im Jahr (1-12) | 1, 2, 3, 12 |
| *anzahl\_tage* | Integer | Anzahl der Tage im Monat | 28, 29, 30, 31 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Quartal*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *quartal\_id* | String (PK) | Eindeutige ID für das Quartal | Q1, Q2 |
| *jahr\_id* | String (FK) | Verweis auf *jahr\_id* in Tabelle *Jahr* | FY2022 |
| *quartal\_bezeichnung* | String | Name des Quartals | Drittes Quartal |
| *quartal\_nummer* | Integer | Quartalnummer (1 bis 4) | 1, 2, 3, 4 |
| *startdatum* | Date | Startdatum des Quartals | 2024-04-01 |
| *enddatum* | Date | Enddatum des Quartals | 2024-03-31 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Jahr*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *jahr\_id* | String (PK) | Eindeutige Jahr-ID | FY2024 |
| *jahr\_name* | String | Bezeichnung des Jahres | 2024 |
| *startdatum* | Date | Beginn des Jahres | 2024-01-01 |
| *enddatum* | Date | Ende des Jahres | 2024-12-31 |
| *ist\_schaltjahr* | Boolean | Angabe, ob das Jahr ein Schaltjahr ist | True, True |
| *anzahl\_tage* | Integer | Anzahl der Tage im Jahr | 365 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Tag\_to*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *datum* | Date (PK) | Datum (YYYY-MM-DD) | 2018-04-03 |
| *monat\_id* | String (FK) | Verweis auf *monat\_id* in Tabelle *Monat* | Apr |
| *wochentag\_bezeichnung* | String | Bezeichnung des Wochentags | Montag |
| *wochentag\_nummer* | Integer | Nummer des Wochentags | 1 |
| *anzahl\_stunden* | Integer | Anzahl der Stunden an diesem Tag | 24 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Monat\_to*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *monat\_id* | String (PK) | Eindeutige Monats-ID | Jan |
| *quartal\_id* | String (FK) | Verweis auf *quartal\_id* in Tabelle *Quartal* | Q1 |
| *monat\_bezeichnung* | String | Name des Monats | Januar |
| *monat\_nummer* | Integer | Monat im Jahr (1-12) | 1, 2, 3, 12 |
| *anzahl\_tage* | Integer | Anzahl der Tage im Monat | 28, 29, 30, 31 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Quartal\_to*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *quartal\_id* | String (PK) | Eindeutige ID für das Quartal | Q1, Q2 |
| *jahr\_id* | String (FK) | Verweis auf *jahr\_id* in Tabelle *Jahr* | FY2022 |
| *quartal\_bezeichnung* | String | Name des Quartals | Drittes Quartal |
| *quartal\_nummer* | Integer | Quartalnummer (1 bis 4) | 1, 2, 3, 4 |
| *startdatum* | Date | Startdatum des Quartals | 2024-04-01 |
| *enddatum* | Date | Enddatum des Quartals | 2024-03-31 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Jahr\_to*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *jahr\_id* | String (PK) | Eindeutige Jahr-ID | FY2024 |
| *jahr\_name* | String | Bezeichnung des Jahres | 2024 |
| *startdatum* | Date | Beginn des Jahres | 2024-01-01 |
| *enddatum* | Date | Ende des Jahres | 2024-12-31 |
| *ist\_schaltjahr* | Boolean | Angabe, ob das Jahr ein Schaltjahr ist | True, True |
| *anzahl\_tage* | Integer | Anzahl der Tage im Jahr | 365 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pruefung*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *pruefung\_id* | Integer (PK) | Eindeutige ID der Prüfung | 1001 |
| *raum\_id* | String (FK) | Verweis auf *raum\_id* in Tabelle *Raum* | A4.0.19 |
| *datum\_pruefung* | Date (FK) | Verweis auf *datum* in Tabelle *Tag* | 2022-03-24 |
| *leiter\_id* | Integer (FK) | Verweis auf *leiter\_id* in Tabelle *Leiter* | 2342 |
| *modul\_id* | String (FK) | Verweis auf *modul\_id* in Tabelle *Modul* | GISH |
| *semester\_id* | String (FK) | Verweis auf *semester\_id* in Tabelle *Semester* | SS2024 |
| *uhrzeit\_beginn* | Time | Beginnzeit der Prüfung | 08:30 |
| *uhrzeit\_ende* | Time | Endzeit der Prüfung | 10:30 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Faktentabelle*** | | | |
| **Attribut** | **Typ** | **Beschreibung** | **Beispiel** |
| *matrikelnummer* | Integer (FK) | Verweis auf *matrikelnummer* in Tabelle *Student* | 5236028 |
| *pruefung\_id* | Integer (FK) | Verweis auf *pruefung\_id* in Tabelle *Pruefung* | 1001 |
| *noten* | Float | Note des Studenten in Prozent (0-100) | 88.5 |

3.2.2 Beziehungen:

* Die **Tabelle** *Student* ist über die Spalte *semester\_id* mit der **Tabelle** *Semester* durch die Spalte *semester\_id* verbunden. Ein Student ist somit einem Semester zugeordnet.
* Die **Tabelle** *Student* ist über die Spalte *datum\_geburt* mit der **Tabelle** *Tag* durch die Spalte *datum* verknüpft, was das Geburtsdatum des Studenten angibt.
* Die **Tabelle** *Student* ist über die Spalte *stadt\_id* mit der **Tabelle** *Stadt* durch die Spalte *stadt\_id* verknüpft, d.h. ein Student ist in einer bestimmten Stadt geboren.
* Die **Tabelle** *Student* ist über die Spalte *studiengang\_id* mit der **Tabelle Studiengang** durch die Spalte *studiengang\_id* verbunden, d.h. ein Student ist in einem Studiengang eingeschrieben.
* Die **Tabelle** *Modul* ist über die Spalte *leiter\_id* mit der **Tabelle** *Leiter* durch die Spalte *leiter\_id* verknüpft, d.h. ein Leiter gehört einem bestimmten Fachbereich an.
* Die **Tabelle** *Leiter* ist über die Spalte *fachbereich\_id* mit der **Tabelle** *Fachbereich* durch die Spalte *fachbereich\_id* verknüpft, d.h. ein Leiter gehört einem bestimmten Fachbereich an.
* Die **Tabelle** *Leiter* ist über die Spalte *stadt\_id* mit der **Tabelle** *Stadt* durch die Spalte *stadt\_id* verknüpft, d.h. ein leiter ist in einer bestimmten Stadt geboren.
* Die **Tabelle** *Leiter* ist über die Spalte *buero\_id* mit der **Tabelle** *Buero* durch die Spalte *buero\_id* verbunden, d.h. ein Leiter ist einem bestimmten Büro zugeordnet.
* Die **Tabelle** *Leiter* ist über die Spalte *geburtsdatum* mit der **Tabelle** *Tag* durch die Spalte *datum* verknüpft, was das Geburtsdatum des Leiters angibt.
* Die **Tabelle** *Leiter* ist über die Spalte *eintrittsdatum* mit der **Tabelle** *Tag\_to* durch die Spalte *datum* verknüpft, d.h. ein Leiter hat seine Tätigkeit an einem bestimmten Datum begonnen.
* Die **Tabelle** *Semester* ist über die Spalte *semester\_start* mit der **Tabelle** *Tag* durch die Spalte *datum* verknüpft, d.h. ein Semester beginnt an einem bestimmten Datum.
* Die **Tabelle** *Semester* ist über die Spalte *semester\_ende* mit der **Tabelle** *Tag\_to* durch die Spalte *datum* verknüpft, d.h. ein Semester endet an einem bestimmten Datum.
* Die **Tabelle** *Studiengang* ist über die Spalte *leiter\_id* mit der **Tabelle** *Leiter* durch die Spalte *leiter\_id* verbunden, d.h. ein Studiengang wird von einem Leiter geleitet.
* Die **Tabelle** *Studiengang* ist über die Spalte *fachbereich\_id* mit der **Tabelle Fachbereich** durch die Spalte *fachbereich\_id* verbunden, d.h. ein Studiengang gehört zu einem bestimmten Fachbereich.
* Die **Tabelle** *Studiengang* ist über die Spalte *bewerbungsfrist* mit der **Tabelle** *Tag* durch die Spalte *datum* verknüpft, d.h. ein Studiengang hat eine festgelegte Bewerbungsfrist.
* Die **Tabelle** *Fachbereich* ist über die Spalte *leiter\_id* mit der **Tabelle** *Leiter* durch die Spalte *leiter\_id* verknüpft, d.h. ein Fachbereich wird von einem Leiter geleitet.
* Die **Tabelle** *Buero* ist über die Spalte *gebaeude\_id* mit der **Tabelle** *Gebaeude* durch die Spalte *gebaeude\_id* verknüpft, d.h. ein Büro befindet sich in einem bestimmten Gebäude.
* Die **Tabelle** *Raum* ist über die Spalte *gebaeude\_id* mit der **Tabelle** *Gebaeude* durch die Spalte *gebaeude\_id* verknüpft, d.h. ein Raum befindet sich in einem bestimmten Gebäude.
* Die **Tabelle** *Gebaeude* ist über die Spalte *stadt\_id* mit der **Tabelle** *Stadt* durch die Spalte *stadt\_id* verknüpft, d.h. ein Gebäude befindet sich in einer bestimmten Stadt.
* Die **Tabelle** *Gebaeude* ist über die Spalte *baudatum* mit der **Tabelle** *Tag* durch die Spalte *datum* verknüpft, was das Baudatum des Gebäudes angibt.
* Die **Tabelle** *Stadt* ist über die Spalte *land\_id* mit der **Tabelle** *Land* durch die Spalte *land\_id* verknüpft, d.h. eine Stadt liegt in einem bestimmten Land.
* Die **Tabelle** *Land* ist über die Spalte *kontinent\_id* mit der **Tabelle** *Kontinent* durch die Spalte *kontinent\_id* verknüpft, d.h. ein Land liegt auf einem bestimmten Kontinent.
* Die **Tabelle** *Tag* ist über die Spalte *monat\_id* mit der **Tabelle** *Monat* durch die Spalte *monat\_id* verknüpft, d.h. ein Tag gehört zu einem bestimmten Monat.
* Die **Tabelle** *Monat* ist über die Spalte *quartal\_id* mit der **Tabelle** *Quartal* durch die Spalte *quartal\_id* verknüpft, d.h. ein Monat ist Teil eines bestimmten Quartals.
* Die **Tabelle** *Quartal* ist über die Spalte *jahr\_id* mit der **Tabelle** *Jahr* durch die Spalte *jahr\_id* verknüpft, d.h. ein Quartal gehört zu einem bestimmten Jahr.
* Die **Tabelle** *Tag\_to* ist über die Spalte *monat\_id* mit der **Tabelle** *Monat\_to* durch die Spalte *monat\_id* verknüpft, d.h. ein Tag gehört zu einem bestimmten Monat.
* Die **Tabelle** *Monat\_to* ist über die Spalte *quartal\_id* mit der **Tabelle** *Quartal\_to* durch die Spalte *quartal\_id* verknüpft, d.h. ein Monat ist Teil eines bestimmten Quartals.
* Die **Tabelle** *Quartal\_to* ist über die Spalte *jahr\_id* mit der **Tabelle** *Jahr\_to* durch die Spalte *jahr\_id* verknüpft, d.h. ein Quartal gehört zu einem bestimmten Jahr.
* Die **Tabelle** *Pruefung* ist über die Spalte *raum\_id* mit der **Tabelle** *Raum* durch die Spalte *raum\_id* verknüpft, d.h. eine Prüfung findet in einem bestimmten Raum statt.
* Die **Tabelle** *Pruefung* ist über die Spalte *datum\_pruefung* mit der **Tabelle Tag** durch die Spalte *datum* verknüpft, was das Datum der Prüfung angibt.
* Die **Tabelle** *Pruefung* ist über die Spalte *leiter\_id* mit der **Tabelle** *Leiter* durch die Spalte *leiter\_id* verknüpft, d.h. eine Prüfung wird von einem Leiter im Rahmen eines Moduls betreut.
* Die **Tabelle** *Pruefung* ist über die Spalte *modul\_id* mit der **Tabelle** *Modul* durch die Spalte *modul\_id* verknüpft, d.h. eine Prüfung gehört zu einem bestimmten Modul.
* Die **Tabelle** *Pruefung* ist über die Spalte *semester\_id* mit der **Tabelle** *Semester* durch die Spalte *semester\_id* verknüpft, d.h. eine Prüfung findet in einem bestimmten Semester statt.
* Die **Tabelle** *Faktentabelle* ist über die Spalte *Matrikelnummer* mit der **Tabelle** *Student* durch die Spalte *Matrikelnummer* verknüpft, d.h. eine Note wird einem bestimmten Studenten zugeordnet.
* Die **Tabelle** *Faktentabelle* ist über die Spalte *pruefung\_id* mit der **Tabelle** *Pruefung* durch die Spalte *pruefung\_id* verknüpft, d.h. eine Note wird in einer bestimmten Prüfung erzielt.

3.2.3 ER-Modell:

3.2.4 Python Code:

*from flask import Flask*

*from flask\_sqlalchemy import SQLAlchemy*

*app = Flask(\_\_name\_\_)*

*app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///university.db'*

*db = SQLAlchemy(app)*

*class Student(db.Model):*

*\_\_tablename\_\_ = 'student'*

*matrikelnummer = db.Column(db.Integer, primary\_key=True)*

*semester\_id = db.Column(db.String, db.ForeignKey('semester.semester\_id'))*

*datum\_geburt = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*staatsangehoerigkeit = db.Column(db.String, db.ForeignKey('land.staatsangehoerigkeit'))*

*studiengang\_id = db.Column(db.String, db.ForeignKey('studiengang.studiengang\_id'))*

*geburtsort = db.Column(db.String, db.ForeignKey('stadt.stadt\_bezeichnung'))*

*nutzerkennung = db.Column(db.String)*

*vorname = db.Column(db.String)*

*nachname = db.Column(db.String)*

*praeferenzen = db.Column(db.Text)*

*class Modul(db.Model):*

*\_\_tablename\_\_ = 'modul'*

*modul\_id = db.Column(db.String, primary\_key=True)*

*leiter\_id = db.Column(db.Integer, db.ForeignKey('leiter.leiter\_id'))*

*ist\_pflichtfach = db.Column(db.Boolean)*

*modul\_name = db.Column(db.String)*

*pruefungsart = db.Column(db.String)*

*kreditpunkte = db.Column(db.Integer)*

*angebotshaeufigkeit = db.Column(db.String)*

*class Semester(db.Model):*

*\_\_tablename\_\_ = 'semester'*

*semester\_id = db.Column(db.String, primary\_key=True)*

*semester\_beginn = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*semester\_ende = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*semester\_abkuerzung = db.Column(db.String)*

*semester\_bezeichnung = db.Column(db.String)*

*class Studiengang(db.Model):*

*\_\_tablename\_\_ = 'studiengang'*

*studiengang\_id = db.Column(db.String, primary\_key=True)*

*leiter\_id = db.Column(db.Integer, db.ForeignKey('leiter.leiter\_id'))*

*fachbereich\_id = db.Column(db.String, db.ForeignKey('fachbereich.fachbereich\_id'))*

*bewerbungsfrist = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*studiengang\_bezeichnung = db.Column(db.String)*

*abschluss = db.Column(db.String)*

*anzahl\_kreditpunkte = db.Column(db.Integer)*

*dauer = db.Column(db.Integer)*

*modulhandbuch = db.Column(db.String)*

*class Fachbereich(db.Model):*

*\_\_tablename\_\_ = 'fachbereich'*

*fachbereich\_id = db.Column(db.String, primary\_key=True)*

*leiter\_id = db.Column(db.Integer, db.ForeignKey('leiter.leiter\_id'))*

*fachbereich\_bezeichnung = db.Column(db.String)*

*class Leiter(db.Model):*

*\_\_tablename\_\_ = 'leiter'*

*leiter\_id = db.Column(db.Integer, primary\_key=True)*

*fachbereich\_id = db.Column(db.String, db.ForeignKey('fachbereich.fachbereich\_id'))*

*buero\_id = db.Column(db.String, db.ForeignKey('buero.buero\_id'))*

*geburtsdatum = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*eintrittsdatum = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*staatsangehoerigkeit = db.Column(db.String, db.ForeignKey('land.staatsangehoerigkeit'))*

*nutzerkennung = db.Column(db.String)*

*vorname = db.Column(db.String)*

*nachname = db.Column(db.String)*

*geburtsort = db.Column(db.String)*

*telefonnummer = db.Column(db.String)*

*raumort = db.Column(db.String)*

*e\_mail\_Adresse = db.Column(db.String)*

*class Buero(db.Model):*

*\_\_tablename\_\_ = 'buero'*

*buero\_id = db.Column(db.String, primary\_key=True)*

*gebaeude\_id = db.Column(db.String, db.ForeignKey('gebaeude.gebaeude\_id'))*

*gebaeude\_name = db.Column(db.String)*

*groesse = db.Column(db.Integer)*

*kapazitaet = db.Column(db.Integer)*

*class Raum(db.Model):*

*\_\_tablename\_\_ = 'raum'*

*raum\_id = db.Column(db.String, primary\_key=True)*

*gebaeude\_id = db.Column(db.String, db.ForeignKey('gebaeude.gebaeude\_id'))*

*raum\_nummer = db.Column(db.Integer)*

*flaeche = db.Column(db.Integer)*

*kapazitaet = db.Column(db.Integer)*

*class Gebaeude(db.Model):*

*\_\_tablename\_\_ = 'gebaeude'*

*gebaeude\_id = db.Column(db.String, primary\_key=True)*

*stadt\_id = db.Column(db.String, db.ForeignKey('stadt.stadt\_id'))*

*baudatum = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*gebaeude\_bezeichnung = db.Column(db.String)*

*flaeche = db.Column(db.Integer)*

*anzahl\_etagen = db.Column(db.Integer)*

*anzahl\_raeume = db.Column(db.Integer)*

*anzahl\_bueros = db.Column(db.Integer)*

*class Stadt(db.Model):*

*\_\_tablename\_\_ = 'stadt'*

*stadt\_id = db.Column(db.String, primary\_key=True)*

*land\_id = db.Column(db.String, db.ForeignKey('land.land\_id'))*

*stadt\_bezeichnung = db.Column(db.String)*

*bevoelkerungsdichte = db.Column(db.Float)*

*anzahl\_bevoelkerung = db.Column(db.Integer)*

*flaeche = db.Column(db.Float)*

*hoehe = db.Column(db.Integer)*

*buerger\_meister = db.Column(db.String)*

*laengengrad = db.Column(db.Float)*

*breitengrad = db.Column(db.Float)*

*class Land(db.Model):*

*\_\_tablename\_\_ = 'land'*

*land\_id = db.Column(db.String, primary\_key=True)*

*kontinent\_id = db.Column(db.String, db.ForeignKey('kontinent.kontinent\_id'))*

*staatsangehoerigkeit = db.Column(db.String)*

*land\_bezeichnung = db.Column(db.String)*

*anzahl\_bevoelkerung = db.Column(db.Integer)*

*flaeche = db.Column(db.Float)*

*hauptstadt = db.Column(db.String)*

*class Kontinent(db.Model):*

*\_\_tablename\_\_ = 'kontinent'*

*kontinent\_id = db.Column(db.String, primary\_key=True)*

*kontinent\_name = db.Column(db.String)*

*anzahl\_bevoelkerung = db.Column(db.Float)*

*flaeche = db.Column(db.Float)*

*class Tag(db.Model):*

*\_\_tablename\_\_ = 'tag'*

*datum = db.Column(db.Date, primary\_key=True)*

*monat\_id = db.Column(db.String, db.ForeignKey('monat.monat\_id'))*

*wochentag\_bezeichnung = db.Column(db.String)*

*wochentag\_nummer = db.Column(db.Integer)*

*anzahl\_stunden = db.Column(db.Integer)*

*class Monat(db.Model):*

*\_\_tablename\_\_ = 'monat'*

*monat\_id = db.Column(db.String, primary\_key=True)*

*quartal\_id = db.Column(db.String, db.ForeignKey('quartal.quartal\_id'))*

*monat\_bezeichnung = db.Column(db.String)*

*monat\_nummer = db.Column(db.Integer)*

*anzahl\_tage = db.Column(db.Integer)*

*class Quartal(db.Model):*

*\_\_tablename\_\_ = 'quartal'*

*quartal\_id = db.Column(db.String, primary\_key=True)*

*jahr\_id = db.Column(db.String, db.ForeignKey('jahr.jahr\_id'))*

*quartal\_bezeichnung = db.Column(db.String)*

*quartal\_nummer = db.Column(db.Integer)*

*startdatum = db.Column(db.Date)*

*enddatum = db.Column(db.Date)*

*class Jahr(db.Model):*

*\_\_tablename\_\_ = 'jahr'*

*jahr\_id = db.Column(db.String, primary\_key=True)*

*jahr\_name = db.Column(db.String)*

*startdatum = db.Column(db.Date)*

*enddatum = db.Column(db.Date)*

*ist\_schaltjahr = db.Column(db.Boolean)*

*anzahl\_tage = db.Column(db.Integer)*

*class Tag\_to(db.Model):*

*\_\_tablename\_\_ = 'tag\_to'*

*datum = db.Column(db.Date, primary\_key=True)*

*monat\_id = db.Column(db.String, db.ForeignKey('monat.monat\_id'))*

*wochentag\_bezeichnung = db.Column(db.String)*

*wochentag\_nummer = db.Column(db.Integer)*

*anzahl\_stunden = db.Column(db.Integer)*

*class Monat\_to(db.Model):*

*\_\_tablename\_\_ = 'monat\_to'*

*monat\_id = db.Column(db.String, primary\_key=True)*

*quartal\_id = db.Column(db.String, db.ForeignKey('quartal.quartal\_id'))*

*monat\_bezeichnung = db.Column(db.String)*

*monat\_nummer = db.Column(db.Integer)*

*anzahl\_tage = db.Column(db.Integer)*

*class Quartal\_to(db.Model):*

*\_\_tablename\_\_ = 'quartal\_to'*

*quartal\_id = db.Column(db.String, primary\_key=True)*

*jahr\_id = db.Column(db.String, db.ForeignKey('jahr.jahr\_id'))*

*quartal\_bezeichnung = db.Column(db.String)*

*quartal\_nummer = db.Column(db.Integer)*

*startdatum = db.Column(db.Date)*

*enddatum = db.Column(db.Date)*

*class Jahr\_to(db.Model):*

*\_\_tablename\_\_ = 'jahr\_to'*

*jahr\_id = db.Column(db.String, primary\_key=True)*

*jahr\_name = db.Column(db.String)*

*startdatum = db.Column(db.Date)*

*enddatum = db.Column(db.Date)*

*ist\_schaltjahr = db.Column(db.Boolean)*

*anzahl\_tage = db.Column(db.Integer)*

*class Pruefung(db.Model):*

*\_\_tablename\_\_ = 'pruefung'*

*pruefung\_id = db.Column(db.Integer, primary\_key=True)*

*raum\_id = db.Column(db.String, db.ForeignKey('raum.raum\_id'))*

*datum\_pruefung = db.Column(db.Date, db.ForeignKey('tag.datum'))*

*leiter\_id = db.Column(db.Integer, db.ForeignKey('leiter.leiter\_id'))*

*modul\_id = db.Column(db.String, db.ForeignKey('modul.modul\_id'))*

*semester\_id = db.Column(db.String, db.ForeignKey('semester.semester\_id'))*

*uhrzeit\_beginn = db.Column(db.Time)*

*uhrzeit\_ende = db.Column(db.Time)*

*class Faktentabelle(db.Model):*

*\_\_tablename\_\_ = 'faktentabelle'*

*matrikelnummer = db.Column(db.Integer, db.ForeignKey('student.matrikelnummer'), primary\_key=True)*

*pruefung\_id = db.Column(db.Integer, db.ForeignKey('pruefung.pruefung\_id'), primary\_key=True)*

*noten = db.Column(db.Float)*

*# Create all tables*

*with app.app\_context():*

*db.create\_all()*

4. Implementierung

4.1 Datenvorverarbeitung (Preprocessing)

4.1.1 Datensammlung und -auswahl

* **Datensammlung:** Wichtige Daten werden aus verschiedenen Datenbanken für das KI-System gesammelt.
* **Datenauswahl:** Wichtige Spalten werden ausgewählt.

4.1.2 Datenbereinigung und Transformation

* **Behandlung von Nullwerten:** Ungültige Noten (Pflichtfächer und Wahlpflichtfächer) und Nullwerte werden identifiziert und durch einen anderen Wert mithilfe von Imputationsmethoden wie z. B. KNNImputer oder IterativeImputer ersetzt.
* **Daten vereinheitlichen:** Noten werden mithilfe von Skalierungsmethoden wie z. B. StandardScaler oder MinMaxScaler standardisiert. Dies ermöglicht es, die Qualität des KI-Algorithmus zu verbessern.
* **Speicherung:** Sowohl mit Daten gefütterte Scaler als auch Imputer werden gespeichert, um eine erneute Nutzung bei neuen Daten zu ermöglichen.

4.1.3 Feature Engineering

* **Featuresauswahl:** Pflichtfachnoten und Präferenzen werden als Features ausgewählt, da sie stark mit den Wahlpflichtfächernoten korrelieren.
* **Targetsauswahl:** Wahlpflichtfächer­noten werden als Targets ausgewählt.

4.1.4 Modellverwaltung, -speicherung und Qualitätssicherung

* **Algorithmusauswahl:** Ein für die vorhandenen Daten und den Zweck geeigneter KI-Algorithmus wird ausgewählt.
* **Hyperparameteroptimierung:** Machine-Learning-Methoden wie Grid Search und Cross-Validation sichern eine optimale Auswahl der KI-Algorithmus-Parameter.
* **Modell-Speicherung:** Das beste Modell mit den besten Hyperparametern wird gespeichert, um es später bei der Vorhersage zu nutzen.
* **Modell-Versionierung:** Eine Version des Modells wird zu Beginn jedes Semesters gespeichert.

4.2 Ablauf der Vorhersageabfrage

4.2.1 Authentifizierung und Abruf der Pflichtnoten und Präferenzen

* **Matrikelnummereingabe:** Der Student gibt seine Matrikelnummer im System ein.
* **Geburtsdatumeingabe:** Der Student gibt sein Geburtsdatum im System ein.
* **Präferenzauswahl:** Der Student wählt anhand eines Fragebogens seine Präferenzen (Skala zwischen 1 und 5) aus.
* **Eingabevalidierung:** Das System validiert die Eingaben des Studenten, um eine SQL-Injection zu verhindern.

4.2.2 Vorhersageberechnung

* **Pflichtfachnotenabruf:** Pflichtfachnoten werden anhand der gültigen Nutzereingaben aus der Datenbank abgerufen.
* **Datenformatumwandlung:** Pflichtnoten und Präferenzen werden in ein für das KI-Modell geeignetes Format umgewandelt.
* **Datenübergabe:** Die Daten werden an das KI-Modell übergeben.
* **Vorhersage:** Das Modell berechnet die erwarteten Wahlpflichtfächernoten und speichert die Ergebnisse.

4.2.3 Ergebnisdarstellung

* **Visuelle Darstellung:** Die Ergebnisse werden dem Nutzer in Form einer Tabelle präsentiert.

5. Sicherheit und Datenschutz

5.1 Datensicherung und Zugriffsrechte

* **Backups-Erstellung:** Für die Datenbank werden in regelmäßigen Zeitabständen (in der Regel einmal pro Semester) Backups erstellt.
* **Zugriffsrechteverwaltung**: Der Zugriff auf die Datenbank erfolgt nur für Administratoren mit den entsprechenden Zugriffsrechten, die mit höchster Vertraulichkeit damit arbeiten.

5.2 Datenschutz und Anonymisierung

* **Datenanonymisierung:** Das Modelltraining erfolgt mit anonymisierten Daten.
* **Datenverschlüsselung:** Sensible Daten werden mit entsprechenden Verschlüsselungsmethoden gesichert.
* **Datenvalidierung:** Nutzereingaben werden im Voraus validiert, um eine Manipulation der Datenbank zu verhindern und Cyberangriffe zu vermeiden, z. B. SQL-Injection.

5.3 Zugriffsbeschränkung

* **Authentifizierung:** Alle Benutzer werden authentifiziert.
* **Zugriffsrechteverwaltung**: Die entsprechenden Zugriffsrechte für Nutzer werden definiert.
* **Datenbeschränkung:** Jeder Nutzer kann nur auf seine Daten zugreifen.

6 Wartung und Dokumentation

6.1 Wartungspläne

* **Modelltraining:** Semesterweise Modellneutrainierung auf Basis neuer Daten (Noten von Pflicht- und Wahlpflichtfächern)
* **Performance-Testing:** RegelmäßigesTesten der Performance des KI-Modells und der Datenbank
* **Performance-Optimierung:** Regelmäßiges Optimierung der Performance des KI-Modells und der Datenbank.

6.2 Dokumentation

* **Benutzerhandbuch:** Ein Ein Benutzerhandbuch wird erstellt, damit die Nutzer (Administratoren und Studenten) das System bedienen können.
* **Technische Dokumentation:** Eine umfassende technische und fachliche Dokumentation der Backend-, Frontend- und Datenbankarchitektur.
* **Fachliche Dokumentation:** Eine umfassende fachliche Dokumentation der Funktionsweise des KI-Modells.

1. Aus Gründen der besseren Lesbarkeit wird in dieser Seminararbeit auf die gleichzeitige Verwendung weiblicher und männlicher Sprachform verzichtet und von nun an lediglich die maskuline Form verwendet. [↑](#footnote-ref-1)