OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Ostbayerische Technische Hochschule Regensburg

Fakultät Informatik und Mathematik

Bachelorarbeit

Diabetes Mellitus Prognosen mit Hilfe einer Technik des Class-Imbalance-Learning

Vergleich zwischen traditionellen Modellen und einem Ensemble-Framework

Sandra Edigin

Matrikelnummer: 9222900

10. März 2025

Erstprüfer: Herr Prof. Brijnesh Jain

Zweitprüfer: Frau Dr. Stefanie Vogl

Studiengang: **Medizinische Informatik**

Inhaltsverzeichnis

Abkürzungsverzeichnis

AUC	Area Under the Curve
Acc.	Accuracy
CSV	Comma Separated Values
EMM	Ensemble-Modell Modified
IDE	Integrierte Entwicklungsumgebung
Kap.	Kapitel
KNN	K-Nearest Neighbor
MLP	Multilayer Perceptron
o. D.	Ohne Datum
Pre	Precision
Rec	Recall
ROC	Received Operation Characteristic
SVM	Support Vector Machine
Tab.	Tabelle

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In den letzten Jahren hat die Relevanz von maschinellen Lernalgorithmen im Gesundheitssystem erheblich zugenommen. Besonders die medizinische Diagnose stellt einen kritischen Aspekt im Gesundheitswesen dar. Beispielsweise erweist sich die Diagnose von Diabetikern oft als sehr aufwendig und komplex, da die Betroffenen mehrere Untersuchungen durchlaufen müssen. Für das medizinische Fachpersonal ist es herausfordernd, bei der Diagnose mehrere Faktoren im Auge zu behalten, infolgedessen es zu fehlerhaften Ergebnissen kommen kann. Diabetes Mellitus ist eine chronische Erkrankung, die sich durch einen erhöhten Fett- und Glucosegehalt im Blut charakterisiert. Weltweit sind schätzungsweise über 382 Millionen Menschen an Diabetes erkrankt, und die Zahl nimmt stetig zu (Ahsan, M. M. et al., 2022). Um die Auswirkungen der Krankheit auf die Gesundheit und Lebensqualität der Betroffenen zu minimieren, ist die Früherkennung und -behandlung von Diabetes von entscheidender Bedeutung. Komplexe Krankheiten zeichnen sich dadurch aus, dass sie einerseits selten auftreten und andererseits viele Faktoren zusammenwirken müssen, um ein Ereignis, in diesem Fall die Entstehung der Erkrankung, zu begünstigen. Seltene Ereignisse werden definiert als „*Vorkommnisse, die mit einer geringeren Frequenz auftreten als häufig auftretende Ereignisse*“ (Haixiang, G. et al., 2016). An dieser Stelle ist das Problem zu verorten. In der Medizin existiert eine größere Anzahl an Daten für „*gesunde“* Menschen bzw. Menschen, die nicht von einer bestimmten Erkrankung betroffen sind, im Vergleich zu denjenigen, die davon betroffen sind (Yu, H. et al., 2014). Infolgedessen ergeben sich einige Fragen: Welche Informationen aus diesen Datenmengen können das Auftreten einer Erkrankung herleiten und inwiefern ist die Klassifizierung dieser Krankheiten durch maschinelle Lernalgorithmen zuverlässig, unter Berücksichtigung des Umstands, dass Krankheiten im Vergleich seltener auftreten?

Diese Fragen sind berechtigt, da aufgrund der großen Anzahl an Daten zunehmend ein Klassifikationsproblem zustande kommt. Das Auftreten eines Ereignisses wird durch die Seltenheit seiner Vorkommnisse erschwert (Haixiang, G. et al., 2016). Das Resultat ist eine inhomogene Verteilung der Samples innerhalb der Klassen (***Class Imbalance***). Beispielsweise ist die Anzahl der Samples, die zu einer Klasse gehören, kleiner als die der anderen Klasse. Der Ursprung dieser Klassenungleichheit liegt in der Datenungleichheit begründet, im englischsprachigen Raum häufiger als ***imbalan-***

ced dataset bezeichnet. Dies ist insbesondere im medizinischen Bereich problematisch, da eine fehlerhafte Klassifikation von Krankheiten, wie beispielsweise Diabetes Mellitus, schwerwiegende Konsequenzen haben kann (Yu, H. et al., 2014 & Rasheed, K. et al., 2022).

Die Anhäufung von Fällen dieser Art hat in den letzten Jahren dazu geführt, dass sich zum einen das sogenannte ***Class Imbalanced Learning*** als eine wichtige Lernstrategie im maschinellen Lernen etabliert hat (Yu, H. et al., 2014), in dessen Rahmen die Lernmodelle mit Daten trainiert werden, die synthetisch angepasst worden sind. Zum anderen führte sie zur Entstehung von **Ensemble-Methoden**, die trotz der Datenungleichheit gute Prognosen liefern können (Zhang, L. et al., 2020 & Ganie, S. et al., 2023).

1.1 Zielsetzung der Arbeit

In der vorliegenden wissenschaftlichen Arbeit wird der Fokus auf die Untersuchung der Leistungen maschineller Lernalgorithmen für die Klassifizierung von Diabetes gelegt. In dieser Untersuchung werden mehrere Lernalgorithmen miteinander verglichen, um die optimale Lernstrategie für die Klassifizierung von Diabetes zu ermitteln. Die Auswahl der Algorithmen erfolgte auf Basis ihrer häufigen Verwendung bei der Klassifizierung von Diabetes (Ahsan, M. M. et al., 2022).

Als Ausgangspunkt dienen folgende Algorithmen:

- **K-Nearest Neighbor (KNN)**
- **Support Vector Machine (SVM)**
- **Multilayer Perceptron (MLP)**
- **Voting Classifier**

1.2 Forschungsfragen

Im Rahmen der Evaluierung der Modellleistung erfolgt die Beantwortung der vorliegenden Forschungsfragen:

1. **Worin besteht der Effekt von Datenvorbearbeitung?**
2. **Worin besteht der Effekt von Class-Imbalance-Techniken zur Bewältigung eines Datenungleichgewichts?**

3. **Wie performen die ausgewählten Modelle?**

Die Beantwortung dieser Forschungsfragen zielt darauf ab, ein tieferes Verständnis für die Stärken und Schwächen der verschiedenen maschinellen Lernalgorithmen sowie Class-Imbalance-Techniken und Ensemble-Methoden zum Zwecke der Klassifizierung von Diabetes zu erlangen.

2 Verwandte Arbeiten

Es existiert bereits eine Vielzahl von Methoden, die das Ziel verfolgen, Diabetes mellitus mittels maschineller Lernalgorithmen vorherzusagen (Ganie, S. et al., 2023). Die Klassifikation von Daten mit unausgewogenen Klassenverteilungen stellt eine Herausforderung im maschinellen Lernen dar. Traditionelle Lernmodelle neigen dazu, die Mehrheitsklasse zu bevorzugen. Die Konsequenz dessen ist eine unzureichende Leistung bei der Vorhersage der Minderheitsklasse. Um dieser Herausforderung zu begegnen, haben einige Forscher zur Klassifizierung von Krankheiten **Ensemble-Learning-Techniken** verwendet, um die Prognosefähigkeit der Modelle zu verbessern (Ganie, S. et al., 2023). Ein Beispiel für die erfolgreiche Anwendung von maschinellen Lernalgorithmen mit Ensemble-Methoden zur Klassifizierung von Diabetes mellitus ist die Studie von Khan et al. 2021 (Ganie, S. et al., 2023). In ihrer Studie verwendeten die Autoren verschiedene Klassifikatoren und verglichen diese miteinander. Zu den in der Studie angewendeten Klassifikatoren gehörten Modelle wie naive Bayes, hybride KNN, künstliche neuronale Netze sowie die Ensemble-Methode Gradient boosting. Die Analyse der Ergebnisse ergab, dass der Gradient-Boosting-Algorithmus die beste Leistung aufwies (Ganie, S. et al., 2023). Andere Forscher entwickelten ein ensemblebasiertes Framework, das sie *eDiaPredict* nannten. Hierbei handelt es sich um eine Kombination mehrerer Klassifikatoren, die Prognosen für die Diabetes-Erkrankung liefern. Die endgültige Vorhersage ist die Kombination aller Prognosen, die aggregiert werden. In ihrer Studie wurden verschiedene Machine-Learning-Algorithmen wie ***XGBoost***, ***Random Forest***, ***Support Vector Machine***, ***neuronale Netze*** und ***Decision Tree*** verwendet. Dabei wurden gute Ergebnisse erzielt (Ganie, S. et al., 2023).

Die Kombination mehrerer Modelle, wie die Ensemble-Lerntechniken, dient dem Zweck, die Gesamtleistung der Modelle zu optimieren. Diese Techniken haben sich als effektiv erwiesen, um die Vorhersagegenauigkeit zu erhöhen und die Robustheit der Modelle zu verbessern (Ganie, S. et al., 2023). Gleichzeitig bieten Class-Imbalance-Learning-Techniken Lösungen zur Bewältigung von unausgewogenen Datensätzen (Yu, H. et al., 2014). Diese sind Techniken, die einem Klassenungleichgewicht entgegenwirken. In der vorliegenden Arbeit wird mit Hilfe traditioneller Lernmodelle ein Ansatz präsentiert, der die Stärken von Ensemble-Methoden und Class Imbalance Learning vereint, um die Klassifikationsfähigkeit der Modelle weiter zu steigern. Die Zusam-

menfügung dieser Techniken bezweckt, ein Framework zu entwickeln, dass die Vorhersagegenauigkeit der Lernmodelle für die Minderheitsklasse erhöht und die Gesamtleistung des Modells bei der Klassifizierung verbessern.

3 Grundlagen des maschinellen Lernens

In diesem Kapitel werden die theoretischen Grundlagen präsentiert, die für ein tieferes Verständnis der im Rahmen verwendeten Konzepte und Überlegungen vonnöten sind. Im Folgenden werden in Kapitel 3.1 die grundlegenden Bausteine maschinellen Lernens erläutert, gefolgt von einer Erläuterung ihrer Lernarten in Kapitel 3.2 sowie den Modellvalidierungskriterien in Kapitel 3.3. Daraufhin wird in Kapitel 3.4. die Robustheit maschineller Modelle thematisiert, bevor die Vorgehensweise bei der Suche geeigneter Hyperparameter in Kapitel 3.5 beschrieben wird.

3.1 Definition des maschinellen Lernens

Der Terminus **Machine Learning** wurde erstmals 1959 von Arthur Samuel geprägt. Er definiert maschinelles Lernen als „*ein Feld des Lernens, das Computern die Fähigkeit verleiht, zu lernen, ohne explizit programmiert zu werden*“ (Samuel, A. L., 1959). Maschinelles Lernen befasst sich mit der Bildung von lernfähigen Systemen und Algorithmen. Dieses ist ein Teilbereich der künstlichen Intelligenz, in welchem Computeralgorithmen anhand von ausreichenden Daten Zusammenhänge lernen, um bestimmte Aufgaben zu lösen (Botsch, B., 2023) Die Vorgehensweise beim maschinellen Lernen ist in zwei Phasen unterteilt, die sich ähnlich wie bei der Vorbereitung auf eine Prüfung bei Menschen verhalten (Kelleher, D. J., 2019).

Die initiale Phase befasst sich mit der ***Modellbildung***, welche in diesem Kontext als Vorbereitung auf eine Prüfung interpretiert werden kann. Im Rahmen der Modellbildung, die zugleich als ***Training*** bezeichnet wird, werden geeignete Parameter erlernt, um optimale Algorithmen zu ermitteln.

Die zweite Phase entspricht der Inferenz bzw. der Schlussfolgerung oder dem ***Testing***, womit die Prüfung bezeichnet wird. Diese umfasst die Vorhersage des trainierten Modells für neue, dem Modell unbekannte Daten (Botsch, B., 2023) und die Evaluation des Modells, d. h., das Modell wird bewertet, um dessen Leistung zu prüfen (Kelleher, D. J., 2019).

Kelleher (2019) identifiziert drei wesentliche Bereiche, die als maßgeblich für maschinelles Lernen gelten. Die drei Sektionen lassen sich wie folgt benennen:

- **Datensätze**
- **Algorithmen**
- **Funktionen**

Datensätze sind erhobene Daten, die zumeist in einer Tabelle aufgefasst sind. Jede Zeile in der Tabelle entspricht einem Datensatz, während jede Spalte ein Feature, zu Deutsch *Merkmal*, darstellt, das mit dem Datensatz verknüpft ist (Kelleher, D. J., 2019). Merkmale sind Eigenschaften, welche ein Objekt innerhalb eines spezifischen Datensatzes charakterisieren. Sie sind die individuellen messbaren Eigenschaften der Daten, die als Eingabe für das maschinelle Lernmodell verwendet werden (siehe Abbildung 3.1).

Ein **Algorithmus** ist ein Prozess, welcher durch einen Computer ausgeführt werden kann, um während der Datenanalyse Muster aus den Daten zu erkennen. Das Lernmodell wird trainiert, einen Prozess zu finden, welcher die Korrelation zwischen einem Feature und einer Zielvariable (siehe Kap. 3.2) beschreibt. Das Ergebnis dieses Algorithmus ist eine **Funktion**, die insbesondere die Relation zwischen beiden Variablen am besten wiedergibt. Demnach kann eine Funktion als deterministische Zusammenführung bzw. als Modell aufgefasst werden (siehe Abbildung 3.1), die für jeden Eingabewert einen Ausgabewert liefert. Die Funktionen können sowohl arithmetischer (Subtraktion und Addition) als auch komplexer Natur sein, wie beispielsweise bei neuronalen Netzen (Kelleher, D. J., 2019).

3.2 Arten des maschinellen Lernens

Das maschinelle Lernen setzt sich zusammen aus verschiedenen Lernmethoden, die es ermöglichen, Strukturen und Muster aus Datensätzen zu extrahieren und zu interpretieren. In dieser Arbeit wird das Augenmerk auf die Modelle des ***Supervised Learning*** gelegt, das im Folgenden näher beschrieben wird.

Supervised Learning

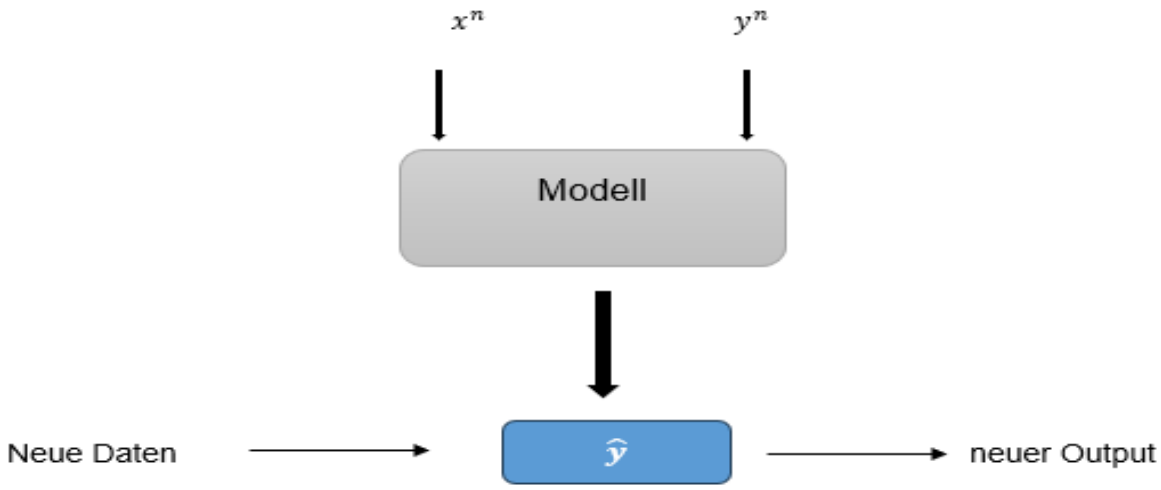

Abbildung 3 1: Lernende Algorithmen

Das Supervised Learning, zu Deutsch **überwachtes Lernen**, ist eine Form des maschinellen Lernens, bei der das Modell mit gelabelten Datensätzen trainiert wird, d. h., Daten, sowohl Eingabedaten (Features x^n) als auch die entsprechenden Entscheidungen (Labels y^n), auch **Zielvariable** genannt, zur Verfügung gestellt werden. Das Modell lernt im Rahmen des Trainings Muster und Struktur der Daten. Das Ziel besteht darin, wie in Abb. 3.1 dargestellt, dass das Modell neue Prognosen auf bislang *unbekannte neue Daten* tätigen kann (Botsch, B., 2023).

Im überwachten Lernen wird im Wesentlichen zwischen zwei Lernmodellen unterschieden: der ***Klassifikation*** und der ***Regression*** (Botsch, B., 2023).

3.2.1 Regression

Die Regression ist ein Lernmodell, bei der eine Korrelation zwischen unabhängiger Variable $\boldsymbol{x_{i=1\ldots n}}$ und Zielvariable $\boldsymbol{y_{i=1\ldots n}}$ beschrieben wird. Sie wird verwendet, wenn die Aufgabenstellung kontinuierliche Ausgabewerte erwartet, d. h., die Zielvariable $\boldsymbol{y}$ kann kontinuierliche nummerische Werte annehmen (Botsch, B., 2023).

Das weitere Vorgehen bei der Regression wird an dieser Stelle nicht weiter thematisiert. Die Beschreibung dient lediglich der vollständigen Vermittlung von Grundlagenwissen im Bereich der Regression in Rahmen des Supervised Learning. Für die praktische Arbeit sind diese Kenntnisse jedoch von untergeordneter Bedeutung.

3.2.2 Klassifikation

Die Klassifikation lässt sich als die differenzierbare Trennung von Merkmalen in gut unterscheidbare Klassen bzw. Gruppen definieren. Sie stellt eine Funktion dar, für deren Einsatz ML-Algorithmen vonnöten sind, die dazu befähigt sind, *Klassenproblemen* eine Klassenbezeichnung zuzuweisen. Die vier Hauptkategorien der Klassifikation von Problemen sind die *binäre Klassifizierung*, die *Mehrklassen-Klassifizierung*, die *Multi-Label-Klassifizierung* und die *unausgewogene Klassifizierung*. Die binäre Klassifizierung findet Anwendung bei Aufgaben, die zwei Klassenlabels aufweisen. Eine repräsentiert den Normalzustand, die andere den abweichenden Zustand. Im Gegensatz dazu umfasst die Mehrklassenklassifikation Aufgaben, die mehr als zwei Klassenlabels aufweisen und folglich, als *Klassifikationsaufgabe* bezeichnet werden (Hansot, J. et al., 2025). Sind die Merkmale *linear* separierbar, d. h., die Trennung der Klassen wird durch eine Trennlinie voneinander getrennt, werden lineare Lernalgorithmen eingesetzt, wohingegen Nichtlinearitäten dem entsprechenden Lernalgorithmus hinzugefügt werden müssen (Botsch, B., 2023).

Die Art der Klassifikation, ob es sich um eine **Entscheidungslinie** (Trennlinie)**, Hyperebene** oder Klassenbildung durch **Mehrheitsentscheidung** handelt, ist abhängig von dem verwendeten Klassifikator. Dies wird in Kapitel 4 genauer erläutert.

3.3 Modellvalidierung

Die Validierung ist ein wichtiges Konzept des maschinellen Lernens. Ohne diesen Schritt kann sich die Suche nach dem geeigneten Modellparameter und Hyperparameter und damit zugleich die Suche nach dem besten Modell, das eine spezifische Aufgabe bewältigen soll, als herausfordernd erweisen. Dennoch stellt sich die Frage, was genau das Grundprinzip der Aufteilung von Datensätzen ist, auf welche Weise die

Aufteilung der Daten erfolgen sollte, um einerseits eine möglichst repräsentative Abbildung der Realität zu gewährleisten (Jirakittayakorn, N. et al., 2024) und anderseits die Leistung des Modells für die Aufgabe zu optimieren.

Die Datensätze werden, um die repräsentative Darstellung der Realität zu ermöglichen, wie folgt in drei wesentliche Sets aufgeteilt (Jirakittayakorn, N. et al., 2024):

- **Trainingsdaten**
- **Validierungsdaten**
- **Testdaten**

3.3.1 Trainingsdatensätze

Die Trainingsdatensätze entsprechen 70 % des gesamten Datensätze und werden für die Bildung der Modellparameter genutzt. Diese Parameter definieren, in welchem Ausmaß das Modell die zugrunde liegenden Strukturen und Muster in den Daten erfasst hat (Jirakittayakorn, N. et al., 2024 & Trivedi, V. et al., 2024).

3.3.1 Validierungsdatensätze

Validierungsdaten sind jene Daten, welche für die Bewertung des Modells verwendet werden. Sie entsprechen 15 % des gesamten Datensatzes, der genutzt wird, um die Hyperparameter bzw. die Komplexität der geschätzten Funktion zu optimieren (Jirakittayakorn, N. et al., 2024). Die Hyperparameter werden im Gegensatz zu Modellparametern nicht vom Modell selbst erlernt, vielmehr werden sie vor dem Training von dem Anwender festgelegt (Fan, Z. et al., 2025).

3.3.1 Testdatensätze

Die Testdatensätze entsprechen den letzten 15 % der Daten. Sie werden für die endgültige Evaluation des Modells verwendet (Jirakittayakorn, N. et al., 2024).

3.3.2 K-Fold Cross-Validation

Nachdem das Modell mit den Trainingsdaten trainiert worden ist und entsprechende Modellparameter erlernt hat, wird das Gütemaß des Modells anhand der Validierungsdaten bewertet (Jirakittayakorn, N. et al., 2024). In der Regel folgen diese Schritte in der Modellbildungsphase, um während des Trainings die Hyperparameter zu optimieren.

Die Daten werden gemäß dem Algorithmus der **Kreuzvalidierung** (Cross-Validation), zufällig in Trainings- und Testdatensatz unterteilt. Der Trainingsteil wird verwendet, um das Modell zu trainieren, der Test- bzw. Validierungsdatensatz für die Bewertung des Modells genutzt (Jirakittayakorn, N. et al., 2024). Da dieser Vorgang mehrmals wiederholt wird, stellt die Kreuzvalidierung eine rechenintensive Methode dar. Es existieren verschiedene Kreuzvalidierungsmethoden, die je nach Datensatzgröße angewendet werden können (Botsch, B., 2023). In diesem kurzen Abschnitt wird die vorliegend verwendete Variation behandelt.

Die **K-Fold Cross-Validierung** ist eine spezielle Art der Kreuzvalidierung. Die Datensätze werden in k-gleich große Teilmengen (Fold) gesplittet. Das Modell wird K-mal trainiert und validiert. In jedem Durchlauf wird der k-te Fold als Validierungsdatensatz verwendet. Dieser Prozess wird K-mal wiederholt, bis alle Datensätze einmal als Validierungsdaten verwendet worden sind (Botsch, B., 2023).

3.4 Robustheit eines Modells

Wie bereits erwähnt, besteht das Ziel maschineller Lernalgorithmen darin, richtige Prognosen für neue, bislang unbekannte Daten zu treffen (Botsch, B., 2023). Unabhängig von der Art der Vorhersage, ob diese kategorisch oder kontinuierlich erfolgt (siehe Kapitel 3.2), liegt das Augenmerk auf der Fähigkeit dieser Modelle, spezifische Aufgaben zu bewältigen. Es gilt zu eruieren, wie gut diese Modelle performen und ob sie als zuverlässig zu betrachten sind. Wie bereits in Kapitel 3.3. erläutert, werden die Daten entsprechend aufgeteilt, um anschließend in der Testphase die Vorhersage- und Generalisierungsfähigkeit eines Modells zu bewerten. Eine gute Generalisierbarkeit geht mit der Vorhersagequalität des Modells einher. Dafür werden die Validierungs- und Testdatensätze verwendet, um zu überprüfen, ob ähnliche Ergebnisse wie bei den Trainingsdaten zu erreichen sind. Wie in Abb. 3.2 *(linkes Bild)* dargestellt, ist eine gute

Generalisierungsfähigkeit des Modells vorhanden, wenn die Komplexität des Modells ähnlich wie die des zu untersuchenden Problems ist (Botsch, B., 2023), sprich sie entsteht durch die Balance zwischen dem ***Bias***[1] und der ***Varianz***[2]. Besteht ein Ungleichgewicht zwischen diesen beiden Konzepten, kann es zu einer **Über-** oder **Unteranpassung** des Modells kommen.

3.4.1 Überanpassung

Eine **Überanpassung** liegt vor, wenn das Modell für die Trainingsdaten gute Ergebnisse liefert, für unbekannte Daten (Testdaten) hingegen schlechte. Im Kern bedeutet dies lediglich, dass die Varianz im Vergleich zum Bias zu hoch ist. Diesbezüglich wird von einem *zu komplexen Modell* gesprochen (siehe Abb. 3.2 *rechtes Bild*). Das Modell lernt zusätzlich aus den Trainingsdaten das Rauschen in diesen Datensätzen (Bisong, E., 2019 & Botsch, B., 2023). Die logische Konsequenz dessen ist eine mangelhafte Generalisierungsfähigkeit des Modells. Das Modell lernt keine Strukturen aus den Daten, sondern nur auswendig.

3.4.2 Unteranpassung

Eine **Unteranpassung** liegt vor, wenn das Modell unzureichend die zugrunde liegenden Strukturen der Daten gelernt hat (siehe Abb. 3.1 *mittleres Bild)*. Hierbei wird von einem *zu einfachen Modell* gesprochen. Dies ist auf ein zu hohes Bias zurückzuführen (Bisong, E., 2019 & Botsch, B., 2023).

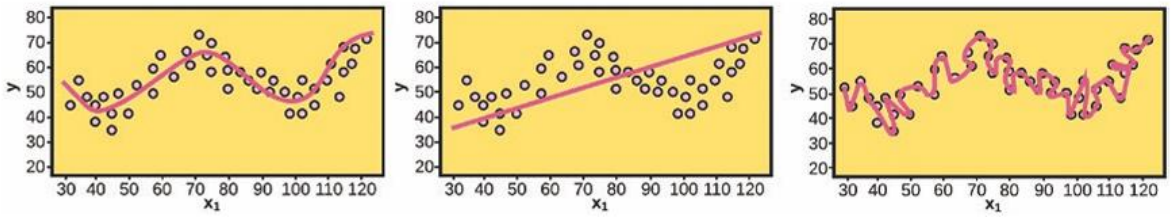

Abbildung 3 2: Modell-Komplexität (Bild von Bison, E., 2019)

[1] Beschreibt die durchschnittliche Abweichung der Vorhersage zu dem tatsächlichen Wert.
[2] Beschreibt die Flexibilität des Modells zu den unterschiedlichen Datensätzen.

3.5 Hyperparameter-Tuning

In Kapitel 3.3.1 wurde bereits darauf eingegangen, dass Hyperparameter zur Optimierung von Modellen eingesetzt werden. Die Auswahl, Art und Anzahl der Hyperparameter sind dabei abhängig von den spezifischen Aufgabenstellungen, dem Modell und dem verwendeten Datensatz. Dies macht die Suche nach geeigneten Hyperparametern zu einer herausfordernden Aufgabe, die oftmals nicht intuitiv ist.

Es existieren Algorithmen, welche die Suche nach den für das Modell optimalen Hyperparametern erleichtern (Fan, Z. et al., 2025). Diese werden im Folgenden beschrieben.

3.5.1 Grid Search

Der Grid Search stellt die am häufigsten verwendete Methode zur Bestimmung und Optimierung der Hyperparameter dar (Fan, Z. et al., 2025). Der Algorithmus basiert auf einer kartesischen Struktur, die eine Reihe von Konfigurationen umfasst, aus denen schließlich die als „besten“ eingestuften Hyperparameter selektiert werden. Ein Nachteil dieses Ansatzes ist jedoch, dass der Anwender sowohl die Grenzen für diesen Bereich festlegen muss als auch die Quantisierung des Gitters, d. h., die Bestimmung eines angemessenen Werts vornehmen muss (Ickenroth, T. et al., 2024). Ein weiterer Nachteil besteht darin, dass die Methode an „Curse of dimensionality“ (Fluch der Dimensionalität) leidet, da die Anzahl der Funktionsevaluationen mit der Anzahl der Hyperparameter exponentiell steigt (Fan, Z. et al., 2025).

3.5.2 Random Search

Der Random Search evaluiert die Hyperparameter im Vergleich zufällig, d. h., es wird eine gleichmäßige Abtastung des Hyperparameterraums anstelle einer kartesischen Struktur angewendet (Fan, Z. et al., 2025). Auch bei diesem Verfahren wird der Suchraum vom Benutzer festgelegt. Im Vergleich zum Grid Search bietet der Random Search den entscheidenden Vorteil eines stetig ansteigenden Wahrscheinlichkeitsanstiegs, wenn die „besten“ Hyperparameter gesucht werden (Ickenroth, T. et al., 2024).

4 Überblick der verwendeten Lernalgorithmen

Im vorliegenden Kapitel werden die im überwachten Lernen und im Rahmen dieser Arbeit verwendeten Klassifikatoren – **K-Nearest Neighbor**, **Support Vector Machine**, **Multilayer Perceptron** und **Voting Classifier** – beschrieben. Jedes Lernmodell wird anhand einer kurzen Beschreibung vorgestellt und hinsichtlich seiner Funktionsweise erläutert.

4.1 K-Nearest Neighbor (KNN)

Das **K-Nearest Neighbor** (KNN) ist ein einfaches und weit angewendetes Verfahren (Botsch, B., 2023), welches der Kategorie der *Shallow-Learner* (Burkov, A., 2019) zugeordnet wird. Shallow-Learner sind nach Burkov (2019) Lernmodelle, die ihre Modellparameter während des Trainings mit den Trainingsdaten lernen. Die Besonderheit dieses Verfahren ist, dass es die gesamte Datenmenge als Modell verwendet (Burkov, A., 2019). Die Funktionsweise von KNN basiert auf der Berechnung von Distanzen zwischen allen Datenpunkten und der Ermittlung des kürzesten Abstands zu seinen K-Nachbarn. Demnach berechnet KNN für neue, unbekannte Daten den kürzesten Abstand zu allen Datenpunkten und weist den neuen Datenpunkt seinen nächsten Nachbarn zu (Botsch, B., 2023 & Hansot, J. et al., 2025). Für die Distanzberechnung werden mathematische Funktionen herangezogen. Zu den häufigsten gehören unter anderem der **euklidische Abstand**, (Hansot, J. et al., 2025), der **Manhattan-Abstand** (Burkov, A., 2019) sowie der **Minkowski-Abstand** (Selle, S., 2024).

Euklidischer Abstand

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{M} x_p^{(i)} - x_p^{(j)}} \text{, mit } i \neq j$$

Manhattan-Abstand

$$d(x_i, x_j) = \sum_{p=1}^{M} \left| x_p^{(i)} - x_p^{(j)} \right| \text{, mit } i \neq j$$

Minkowski-Abstand

$$d(x_i, x_j) = \left(\sum_{\boldsymbol{p=1}}^{\boldsymbol{M}} \left| \boldsymbol{x_p^{(i)} - x_p^{(j)}} \right|^{\boldsymbol{p}} \right)^{\frac{\mathbf{1}}{\boldsymbol{P}}} \text{ mit } i \neq j$$

Dabei ist M die Anzahl der Dimensionen (Features), $x_p^{(i)}$ und $x_p^{(j)}$ sind die Koordinaten in der i -ten und j-ten Dimension.

Der Einsatz des KNN in dieser Arbeit ist neben der Verwendung für viele Klassifikationsaufgaben (Ahsan, M. M. et al., 2022) zudem ein schnelles Verfahren. Die zusätzlich zu den Distanz-Metriken verwendeten Hyperparameter umfassen die Anzahl der Nachbarn, die im Intervall [1, 3,5, 7, 9] gewählt wurden, sowie die Gewichte, die zwischen [distance, uniform] gewählt wird. D*istance* definiert für jeden Nachbarn dasselbe Gewicht und *uniform*, dass alle Nachbarn gleich wichtig erachtet werden sollen (KNeighborsClassifier, o. D.). Es sei jedoch angemerkt, dass das Verfahren aufgrund der Distanz Berechnung eine gewisse Empfindlichkeit gegenüber Ausreißern in den Daten aufweist, was eine Beeinflussung der Modellleistung zur Folge haben könnte.

4.2 Support Vector Machine (SVM)

Die **Support Vector Machine** (SVM) ist ein Lernalgorithmus, welcher sowohl in der Klassifikation als auch in der Regressionsanalyse Verwendung findet (Naganathan, G. S. et al., 2019). Die SVM erlernt während des Trainings Modellparameter, die eine optimale *Hyperebene*[3] in einem Feature-Raum definieren (Abdullah, D. M. et al., 2021). Ein Feature-Raum, ebenfalls als Merkmalsraum bekannt, ist ein mehrdimensionaler Raum, in dem jedes Merkmal eine Dimension repräsentiert. Die Abbildung 4.1 stellt die Trennung der Klassen mithilfe der Support-Vektoren (SVs) dar. Diese entsprechen den eingekreisten Vektoren, die auf den gestrichelten Linien liegen. Die SVs bilden den maximalen Abstand, der als Margin (Abdullah, D. M. et al., 2021) bezeichnet wird, zu der anderen Klasse.

[3] *Hyperebenen* sind Ebenen, die in beliebig vielen Dimensionen $\mathbb{R}^m$ definiert sind.

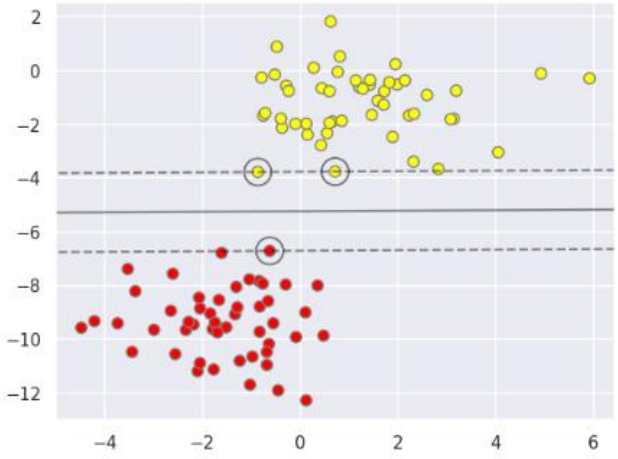

Abbildung 4 1: SVM-Klassifikation bei linearen Datensätzen

Die Art der Margin ist abhängig von der Größe der Regulierungsparameter C und der Wahl der Kernel-Funktion. Auch die Datensätze beeinflussen die Art der verwendeten SVM bzw. den geeigneten Kernel des Modells. Eine besondere Stärke dieses Modells besteht darin, dass es ebenfalls mit Datensätzen, die eine nicht-lineare Korrelation zueinander aufweisen, umgehen kann, mithin mit Daten, die nicht wie in Abb. 4.1 mittels einer Trennlinie separiert werden können. Hierfür wird sich nicht-linearer Kernel-Funktionen wie der Radialbasis-Funktion (RBF) bedient, welche die Daten in einen höherdimensionalen Feature-Raum transformiert (Abdullah, D. M. et al., 2021).

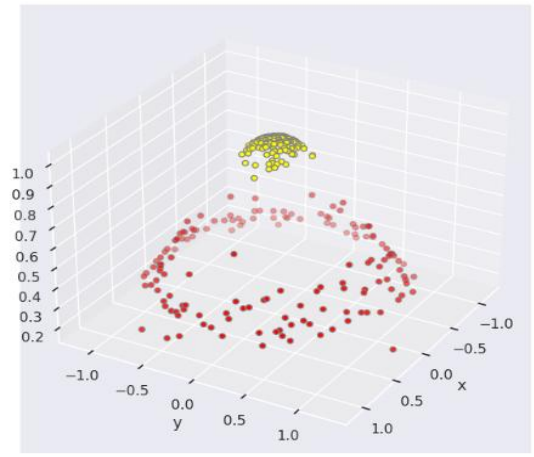

Abbildung 4 2: 3D-Darstellung nicht-linearer Datensätze

Die Mapping-Funktion transformiert die Eingabeparameter der Daten in eine höhere Dimension, sodass sie, wie in Abbildung 4.2 dargestellt, erneut linear trennbar sind und eine optimale Hyperebene definiert werden kann (Chen, Z. et al., 2007). Der SVM

ist demzufolge ideal für nicht-parametrische Datensätze geeignet, bei denen die zugrunde liegende Verteilung der Daten unspezifisch ist. Diese Fähigkeiten machen die SVM zu einem schnellen und robusten Modell (Naganathan, G. S. et al., 2019).

4.3 Multilayer Perceptron (MLP)

Das **Multilayer Perceptron** (MLP) ist ein Klassifikator und eine spezielle Form des neuronalen Netzes (NN), das ein Teilgebiet des maschinellen Lernens darstellt. Die Funktionsweise dieser neuronalen Netze ist dem menschlichen Gehirn nachempfunden, das aus Schichten von Neuronen besteht (Kreutzer et al., 2019).

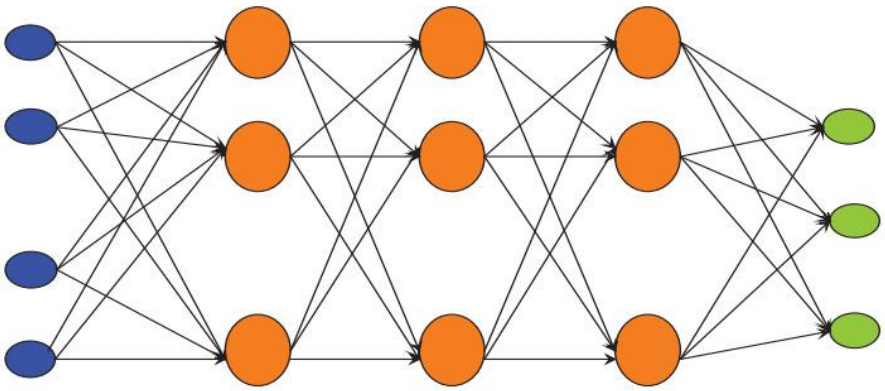

Abbildung 4 3: Struktur Multilayer Perceptron (Bild von Sreejith, S. et al., 2024)

Das MLP-Netzwerk ist in drei Schichten unterteilt (siehe Abbildung 4.3): die *Eingabeschicht* (Input Layer, blaue Neuronen), die *versteckte Schicht* (Hidden Layer, orange Neuronen) und die *Ausgabeschicht* (Output Layer, grüne Neuronen). Jeder Knoten einer Schicht ist vorwärtsgerichtet mit jedem Knoten der folgenden Schicht verbunden, zugleich als **Feed-Forward-Netz** bezeichnet. Die Abbildung 4.3 illustriert die Schichten und deren Kanten, jedoch nicht die zugehörigen Gewichte auf den Kanten (Naganathan, G. S. et al., 2017 & AlShourbaji, I. et al., 2022).

4.3.1 Eingabeschicht

In der Eingabeschicht werden die gewichteten Features mithilfe einer *linearen Aktivierungsfunktion*[4] an die verdeckte Schicht weitergeleitet. Dabei entspricht die Anzahl der

[4] Eine Aktivierungsfunktion bestimmt die Ausgabe der Neuronen in einer Schicht.

Eingabeneuronen (Eingabeknoten) der Dimension der Eingangsmerkmale (AlShourbaji, I. et al., 2022). In diesem Fall bedeutet linear, dass die Gewichte unverändert an die nächste Schicht weitergeleitet werden (Mothe R. et al., 2023).

4.3.2 Verborgene Schicht

Die versteckte Schicht transformiert die Ausgaben der vorherigen Schichten mithilfe der Gewichtsmatrix $\boldsymbol{w}$ und einer *nicht-linearen Aktivierungsfunktion* in eine nicht-lineare Ausgabe für die nächsten Schichten (AlShourbaji, I. et al., 2022). Der Vorteil einer nicht-linearen Transformation besteht darin, dass das Modell die zugrunde liegenden komplexen Strukturen der Merkmale lernen kann. Bei einer linearen Transformation wäre dies entweder nicht oder nur schwer möglich, da viele Informationen verloren gehen. Neben *linearen-* existieren gleichfalls einige nicht-lineare Aktivierungsfunktionen, darunter die ***Rectified Linear Unit*** (RELU), die ***Sigmoid***- oder die ***Tangent hyperbolic (Tanh)-Funktion*** (Mothe, R. et al., 2023).

Rectified Linear Unit (RELU)

Das RELU stellt die am häufigsten angewendete lineare Aktivierungsfunktion dar, da diese ausschließlich die positiven Gewichtete des neuronalen Netzwerkes unverändert an die nächste Schicht weiterleitet und eine Null-Konvertierung bei negativen Gewichten durchführt. Diese Null-Konvertierung hat zur Folge, dass nicht alle Neuronen gleichzeitig aktiviert werden (Mothe, R. et al., 2023).

$$\boldsymbol{f(r) = max(0, r)}$$

Sigmoid-Funktion

Die Sigmoid-Funktion, zugleich als logistische Funktion bekannt, unterdrückt die eingegebenen $\mathbb{R}$- Werte im Intervall zwischen 0 und 1. Werte, die stark positiv sind, werden bei der Ausgabe in der Nähe von 1 an die nächste Schicht weitergegeben, während niedrigere Werte in der Nähe von 0 an die weitere Schicht weitergegeben werden. (Mothe, R. et al., 2023).

$$\boldsymbol{f(x) = \frac{1}{1 + e^{-x}}}$$

Tangent hyperbolic (Tanh)-Funktion

Ein naher Verwandter der Sigmoid-Funktion ist die Tangens-hyperbolische Funktion. Der wesentliche Unterschied zur logistischen Funktion besteht in einer nullzentrierten Ausgabe. Dies bedeutet, dass die Tanh-Funktion $\mathbb{R}$ -Werte im Bereich zwischen -1 und 1 annimmt und der Mittelwert bei null liegt (Mothe, R. et al., 2023).

$$\boldsymbol{f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}}$$

Die Aktivierungsfunktionen beeinflussen den Gradienten eines neuronalen Netzwerkes (Mothe, R. et al., 2023). Diese sind entscheidend für das Training dieser Netzwerke, da sie bestimmen, wie die Gewichte während des **Backpropagation-Prozesses** aktualisiert werden (Naganathan, G. S. et al., 2017). Ein Gradient ist nichts anderes als eine mathematische Beschreibung des Anstiegs einer Funktion. In anderen Worten ist ein Gradient ein Vektor, der die partielle Ableitung $\nabla f \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$ einer Funktion $f(x, y)$, der die Richtung des steilsten Anstiegs der Funktion anzeigt. Sie ist ein wesentlicher Bestandteil der Optimierung von Feed-Forward-Netzwerken, da sie dazu dient, Fehler zu minimieren. Dabei werden die Gradienten der Fehlerfunktion in Bezug auf die Gewichte des neuronalen Netzwerkes berechnet. Die berechneten Fehler werden von den Neuronen der Ausgabeschicht an die Neuronen der Eingabeschacht propagiert und nachträglich angepasst, um Fehler zu minimieren und somit die Leistung des Modells zu verbessern (Naganathan, G. S. et al., 2017).

4.3.3 Ausgabeschicht

Die Ausgabeschicht berechnet einen Vektor aus Vorhersagewahrscheinlichkeiten für die Klassenzugehörigkeit eines Merkmals. Diese Schicht ist unmittelbar mit den entsprechenden Klassen vernetzt. Die Anzahl der Knoten in der Ausgabeschicht entspricht der Anzahl der Klassen (AlShourbaji, I. et al., 2022).

Neben den Aktivierungsfunktionen wurden im Rahmen des Experiments weitere Hyperparameter wie der Solver [*adam*, *lbfg*], die Größe der Hidden Layer im Intervall [(50, 50), (100, 100), (100, 50), (50, 100)], das Alpha und die Lernrate zwischen 0.001 und 0.01 sowie die maximale Anzahl der Iterationen auf 1000 gesetzt. Der Solver bestimmt das Optimierungsverhalten des Modells während dem Training. Das Alpha bestimmt

das Regularisierungsstärke die verwendet wird, um Überanpassung zu verhindern (MLPClassifier, o. D.).

4.4 Voting Classifier

Der Abstimmungsklassifikator ist ein ML-Modell, das durch Training auf einer Reihe von Modellen lernt und eine Ausgabe (Klasse) auf Grundlage der Klasse mit der höchsten Wahrscheinlichkeit vorhersagt. Hierbei existieren zwei Abstimmungsarten: *soft voting* und *hard voting* (Mehrheitsvotum). In dieser Arbeit liegt der Fokus auf dem *hard voting*, da dieser Ansatz bereits erfolgreich bei der Vorhersage von Diabetes Mellitus eingesetzt wurde (Hansot, J. et al., 2025).

Der Voting Classifier kombiniert konzeptionell unterschiedliche ML-Klassifikatoren, wählt daraus für eine Vorhersage denjenigen aus, welcher der höchsten Wahrscheinlichkeit für eine Klassenzugehörigkeit entspricht. Die Klassenzugehörigkeit gestaltet sich wie folgt: $y = mode\{C_1(x), \ldots, C_m(x)\}$ mit x = Eingabemerkmale, y = Vorhersage und $C(x)$ = Klassifikator. Die Verwendung von *hard voting* kann hilfreich sein, um die Schwächen einer Gruppe gleich wirksamer Modelle auszugleichen (Hansot, J. et al., 2025).

5 Evaluationsmetriken

In diesem Kapitel werden lediglich diejenigen Metriken beschrieben, die im Experiment verwendet wurden und sich mit dem binären Klassenproblem von Diabetes befassen, d. h., mit der Klassifikation des Vorhandenseins oder Nichtvorhandenseins von Diabetes.

Das Modell erlernt beim Training Parameter, zugleich als **Modelparameter** bezeichnet, die zur Bewältigung der Use-Cases erforderlich sind. Im Anschluss an das Training folgt die Prädiktion des Modells mit unbekannten Daten und die Bewertung der Leistung dieses Modells. Um die Leistung des Modells bewerten zu können, werden Qualitätskriterien herangezogen. Qualitätskriterien sind Kriterien, welche die Qualität eines Klassifikators beurteilen, und dessen Fähigkeit zur Bewältigung von **Klassifikationsaufgaben** evaluieren. Sie geben Aufschluss über die Leistung und die Eignung des

Modells für spezielle Aufgaben. Exemplarisch wird eine andere Evaluationsmetrik für Regressionen verwendet als für die Klassifikation (Bisong, E., 2019).

5.1 Confusion Matrix

Bei der Confusion Matrix bzw. Konfusionsmatrix handelt es sich um ein Qualitätsmaß, welches insbesondere, nicht ausschließlich, im Rahmen von binären Klassifikationsaufgaben Anwendung findet (Bisong, E., 2019). Per Default repräsentieren die unausgewogenen Klassen in der Konfusionsmatrix die *Minderheitsklasse*, dies entspricht der ***positiven*** Klasse, während die *Majoritätsklasse* die ***negative*** Klasse repräsentiert (Haixiang, G. et al., 2016)

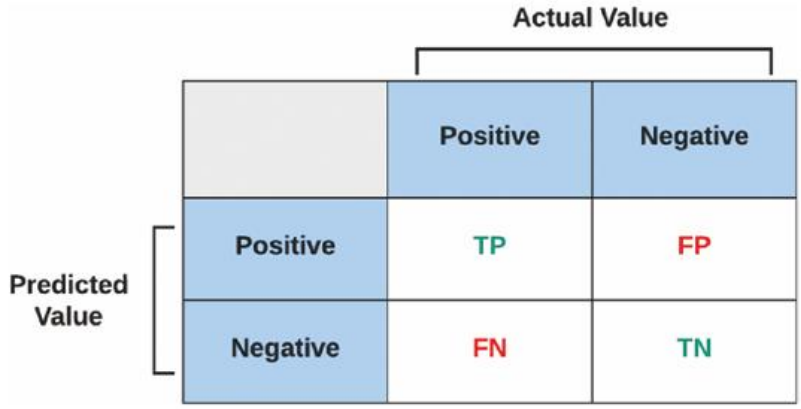

Abbildung 5 1: Konfusionsmatrix (Bild von Bisong, E., 2019)

Die Konfusionsmatrix, wie in Abb. 5.1 dargestellt, ähnelt einer Raster-Tabelle, die aus zwei Spalten und zwei Zeilen besteht. Die Spalten repräsentieren das angestrebte Ergebnis bzw. den wahren Werten, während die Zeilen den Prognosen des Modells entsprechen. Die vier Einträge der Tabelle repräsentieren ***True positive (TP), False positive (FP), False negative (FN)*** und ***True negative (TN)*** (Bisong, E., 2019)***.***

Der Wert ***TP*** (*True positive)* ist die Anzahl an Werten, die der Wahrheit entsprechen (‚*wahr*') und demnach durch das Modell korrekt vorhergesagt wurden. Umgekehrt gilt dies ebenso für den Wert ***TN*** *(True negative)*. Dieser entspricht der Anzahl an Werten, die das Modell richtig negiert hat. **FP** (*False Positive*) hingegen entspricht der Anzahl an Werten, die das Modell fälschlich als ‚*wahr*' klassifiziert hat, jedoch in der Wahrheit als ‚*falsch*' klassifiziert sind. Für **FN** (*False negative*) gilt das Umgekehrte, mithin die

Anzahl der Werte, die das Modell fälschlicherweise als ‚*falsch*' klassifiziert hat, die jedoch in Wahrheit ‚*wahr*' sind. Mittels dieser Werte lässt sich die Leistung des Modells beurteilen und in die nachfolgend erläuterten Eigenschaften zusammensetzen.

5.2 Accuracy

Die Treffgenauigkeit beschreibt die Korrektheit der Vorhersagen eines Modells. Im überwachten Lernen ist sie die meistverwendete Metrik, um die Leistung des Modells zu bewerten (Bisong, E., 2019).

Die Höhe der Treffgenauigkeit steht in direktem Zusammenhang mit der Qualität des Modells. Unter der Annahme einer unausgewogenen Klassenbildung lässt sich eine hohe Treffgenauigkeit nicht allein als Indikator für verlässliche Aussagen über das Modell interpretieren. Anders ausgedrückt bedeutet dies: Im Falle von Datenungleichheit bzw. Klassenungleichgewicht ist die alleinige Betrachtung der Accuracy unzureichend, da diese lediglich zuverlässige Ergebnisse hinsichtlich der dominanten Klasse liefert (Bisong, E., 2019).

$$acc = \frac{TP+TN}{TP+Fp+TN+FN}$$

5.3 Precision

Die Genauigkeit, auch Precision genannt, gibt ausschließlich die als positiv klassifizierten Werte an. Sie misst demnach die Genauigkeit des Modells für die Minderheitsklasse. Ein hoher Wert steht für eine hohe Genauigkeit (Bisong, E., 2019).

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

5.4 Recall

Der Recall, der zugleich mit der Sensitivität gleichgesetzt werden kann, misst die Fähigkeit des Modells, die tatsächlich positiv klassifizierten Werte korrekt zu identifizieren (AlShourbaji, I. et al., 2022).

$$Sensitivität = \frac{TP}{TP+FN}$$

5.5 Receiver Operating Characteristic (ROC) Area Under the Curve (AUC)

Die ROC in Kombination mit der AUC ist eine Metrik, die bei unausgeglichenen Daten verwendet wird. Diese Metrik hat die Eigenschaft, die Fähigkeit von Klassifikatoren zur Trennung von Klassen zu bewerten. Sie misst die True Positive Rates (TPR), ebenfalls bekannt als Sensitivität oder Recall, im Vergleich zu den False Positive Rates (FPR) und berechnet daraus die Fläche unter der Kurve (AUC). Die FPR misst den Anteil der True Negatives, die vom Modell fälschlicherweise als Positive (False Positives) erkannt werden. Je höher der AUC-Wert, desto besser kann das Modell zwischen den Klassen unterscheiden (Bisong, E., 2019).

6 Experiment

Dieses Kapitel befasst sich mit der praktischen Anwendung der in dieser Arbeit diskutierten Konzepte. Im Zuge dessen werden die Überlegungen sowie Versuche während des Experiments beschrieben.

Werkzeuge, die für das Experiment verwendet wurden, umfassen die Hardware, ein *HP-Envy* mit *Intel CORE i5* und *8 Gigabyte RAM* mit einem *64-Bit-Betriebssystem* und einem *x64-basierten Prozessor.* Die verwendete Programmiersprache ist *Python Version 3.12.3* in der IDE *Visual Studio Code* (VSC) unter Verwendung der *virtuellen Umgebung*. Zudem wurden die Bibliotheken, welche für das Experiment relevant sind, vor ihrer Anwendung installiert und importiert.

6.1 Beschreibung des Datensatzes

Für das Experiment wurde der medizinische Datensatz, PIMA-Diabetes-Datensatz, aus dem Repository für maschinelles Lernen aus der Universität von California genutzt (Ganie, S. et al., 2023). Dieser Datensatz wurde der *Kaggle-Community* als tabellarisierter (Comma Separated Values) CSV-Datei zur Verfügung gestellt.

]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

Abbildung 6 1: PIMA-Diabetes-Datensatz

Die Abbildung 6.1 stellt den PIMA-Datensatz, bestehend aus 768 Zeilen und 9 Spalten, dar. Die ersten 8 Spalten entsprechen den unabhängigen Variablen bzw. Features „***Schwangerschaften***, ***Glucose*** *(mmol/L)*, ***Blutdruck*** *(mmHg)*, ***Hautdichte*** *(mm)*, ***In-***

sulin**, **Body-Mass-Index** (BMI) (kg/m^2), **Diabetes-Stammbaumfunktion und das ***Alter***". Die letzte Spalte, die *abhängige Variable* „***Outcome***", entspricht den diskreten Targets bzw. Labels. Der Datensatz beschreibt ein Zwei-Klassen-Problem, mit *0* für *kein Diabetes* und *1* für *Diabetes*.

Features	Datatypes	NaN	Dublicates	Zeros Features	Procent Features
Pregnancies	768 non-null int64	0	False	Keine Angaben	Keine Angaben
Glucose	768 non-null int64	0	False	5	0,65
Blood Pressure	768 non-null int64	0	False	35	4,56
Skin Thickness	768 non-null int64	0	False	227	29,56
Insulin	768 non-null int64	0	False	374	48,70
BMI	768 non-null float64	0	False	11	1,43
Diabetes-Pedigree Function	768 non-null float64	0	False	Keine Angaben	Keine Angaben
Age	768 non-null int64	0	False	Keine Angaben	Keine Angaben

Tabelle 6 1: Darstellung der Dateneigenschaften

Datentypen der Features sind traditionelle Datentypen wie *integer* und *float*. Bis auf die Merkmale BMI und Diabetes-Stammfunktion, die den Datentyp *float* aufweisen, haben die anderen Features den Datentyp *integer.* Fehlende Werte sind in dem vorliegenden Datensatz nicht vorhanden, erkennbar an der *NaN-Spalte*. Des Weiteren existieren keine Duplikate in dem Datensatz. Erkennbar ist jedoch bei den Merkmalen *Hautdichte* und *Insulin* eine erhöhte Anzahl an Nullen in ihren Datensätzen. Diese sind aus physiologischer Sicht nicht möglich, da sie anderenfalls mit dem Leben nicht vereinbar wären. Diese Nullen repräsentieren demnach Füllwerte, die stellvertretend für nicht erhobene Werte dienen.

Features	Mittelwert	Median	Modus	Skewness	Kurtosis
Pregnancies	4	3	1	0,902	0,16
Glucose	120,89 mg/dl	117,00 mg/dl	99,00 mg/dl	0,174	0,641
Blood Pressure	69,11 mm/Hg	72,00 mm/Hg	70,00 mm/Hg	-1,844	5,180
Skin Thickness	20,54 mm	23,00 mm	0,00 mm	0,109	-0,520
Insulin	79,80 IE	30,50 IE	0,00 IE	2,272	7,214
BMI	31,99 kg/m2	32,00 kg/m2	67,10 kg/m2	-0,429	3,290
Diabetes-Pedigree Function	0,47	0,37	0,25	1,92	5,595
Age	33,24	29,00	22,00	1,13	0,643

Tabelle 6 2: Nummerische Darstellung der Datenverteilung einzelne Features

In der Tabelle 6.2 wird die Datenverteilung der einzelnen Features quantitativ dargestellt. Die Art der Datenverteilung wurde durch Verwendung von *skewness* und *kurtosis* analysiert, wobei diese mit den Informationen des *Mittelwerts*, des *Medians* und des *Modus* der einzelnen Features berechnet wurden. Bei Betrachtung des Merkmals *Insulin* in der Spalte *skew* ist ersichtlich, dass dieses, im Vergleich zu anderen Features, einen signifikant hohen positiven Wert von *2,272* aufweist. Dieser positive Wert deutet auf eine schiefe Verteilung der Daten hin. Bei einer Normalverteilung liegen der Mittelwert, der Median und der Modus nahe beieinander und sein *skew* liegt nahe bei *0* (Hatem, G. et al., 2022). Bei einer schiefen Verteilung der Daten, wie es bei dem Merkmal *Insulin* der Fall ist, sind der Mittelwert und der Median größer als der Modus, insbesondere ist der Mittelwert größer als der Median (siehe Tab. 6.2).

Die Datenverteilung für das entsprechende Merkmal zeigt sich überwiegend linksseitig, was zur Folge hat, dass sich ein *„rechter Schwanz*“ ergibt, wie in Abbildung 6.2 (links im linken Bild) erkennbar ist. Bei einer negativen *skew* sind sowohl der *Mittelwert* als auch der *Median* kleiner als der *Modus* und die Differenz zwischen *Mittelwert* und *Median* ist gering. Die Datenpunkte befinden sich überwiegend auf der rechten Seite und ziehen dadurch einen *„linken Schwanz“* nach sich. Die übrigen Merkmale *Hautdichte, BMI* und *Glucose* sind symmetrisch, da sich ihre Schiefe im Bereich *-0,5 > -1* und *0,5 < 1* befinden (Hatem, G. et al., 2022). Die Merkmale *Schwangerschaft, Diabetes-Stammfunktion* sowie *Alter* sind moderat schief, da sich ihre Schiefe im Bereich um den Wert *1* befindet, während die Schiefe des *Bluthochdrucks* im Bereich *-1* zu verorten ist (Hatem, G. et al., 2022).

Über die Kurtosis wurde auf das Vorkommen von Ausreißern analysiert. Features mit einer Kurtosis größer als *0* (siehe Tabelle 6.2) haben einen *„dichteren Schwanz“* (*leptokurtisch*), wohingegen Features mit Werten kleiner als *0* einen *„weniger dichten Schwanz“* (*platykurtisch)* aufweisen. Features mit einer Kurtosis näher bei *0* deuten auf eine Normalverteilung der Daten hin (*mesokurtisch*) (Hatem, G. et al., 2022).

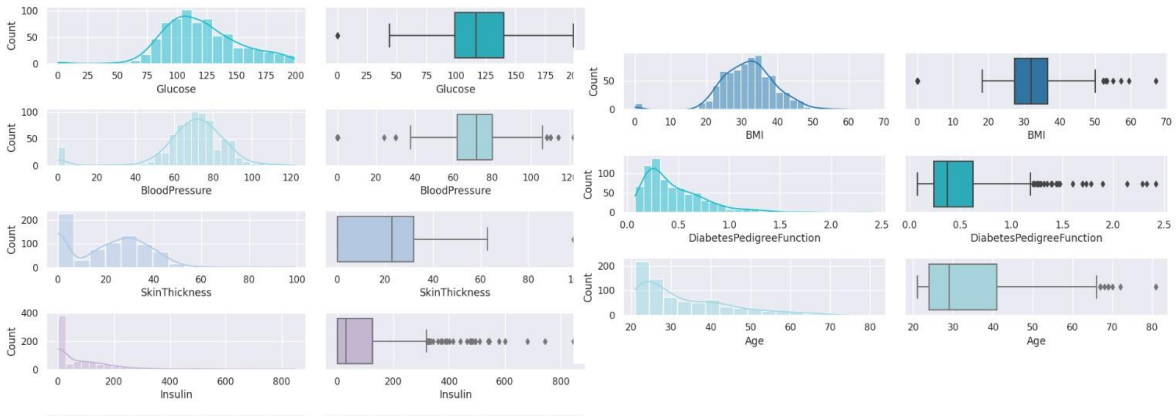

Abbildung 6 2: visuelle Darstellung der Ergebnisse Skew und kurtosis

Die *„Dichte des Schwanzes“* wird visuell als *schwarze Punkte* in der Boxplot-Grafik in der Abbildung 6.2 (rechts vom linken und rechten Bild) dargestellt. Die Abbildung 6.2 stellt *skewness* und *kurtosis* visuell gegenüber. Die x-Achse entspricht den Features in ihren jeweiligen Einheiten, die y-Achse der Anzahl der Datenpunkte. Die schwarzen Datenpunkte werden als Ausreißer identifiziert.

Insbesondere der Feature *Insulin* weist eine *leptokurtische* Verteilung auf. Gleiches gilt für die Features *Blutdruck, BMI, Diabetes-Stammfunktion, Glucose* und *Alter*, wohingegen die *Hautdichte* eine *platykurtische Verteilung* zeigt. Lediglich das Merkmal *Schwangerschaft* indiziert eine mögliche *mesokurtische Verteilung*.

Um auf mögliche Korrelationen zwischen den Features zu prüfen, wurde diese mit Hilfe der Korrelationsmatrix analysiert.

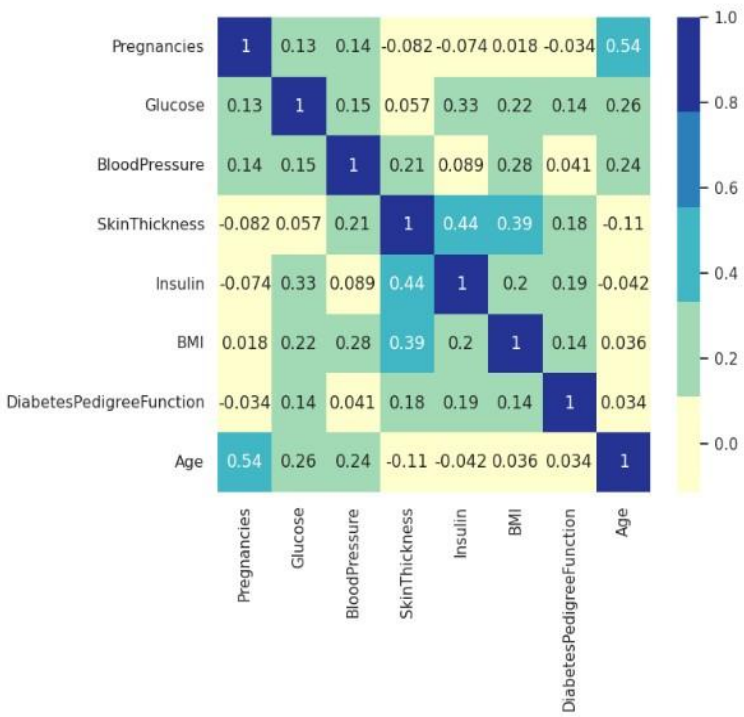

Abbildung 6 3: Korrelationsmatrix

Die Abbildung 6.3 veranschaulicht die Relation zwischen den Features zueinander und die Stärke dieser Relation. Rechts von der Korrelationsmatrix befindet sich die Korrelationsskala. Der Wert *1* auf der Diagonale weist auf eine starke Relation hin, was logisch ist, da die Features auf sich bezogen in einer starken Relation zueinanderstehen. Der Wert *0* hingegen bedeutet keine Relation. Negative und positive Werte kleiner 0 bzw. 1 weisen auf eine bestehende Korrelation hin, die in der Analyse der Daten jedoch geringfügig vorhanden ist. Folglich kann davon ausgegangen werden, dass zwischen den Merkmalen so gut wie keine für das Modell interfierende Korrelation besteht.

Im Rahmen der Datenanalyse wurde schließlich die Klassenverteilung anhand der verfügbaren Daten untersucht. Dieser Schritt wurde aufgrund der Tatsache unternommen, dass im medizinischen Kontext für die Klassifizierung einer Krankheit mehr Daten zur Verfügung stehen, die *keine* Erkrankung begünstigen, als solche, die eine Erkrankung begünstigen (Haixiang, G. et al., 2016 & Rasheed, K. et al., 2022).

Die Abbildung 6.4 veranschaulicht die bereits oben beschriebene Vermutung des Vorhandenseins eines Klassenungleichgewichts. Die Majoritätsklasse (dominante Klasse)

mit 500 Samples entspricht der Klasse *0*, die Minderheitsklasse mit über 268 Samples der Klasse *1*.

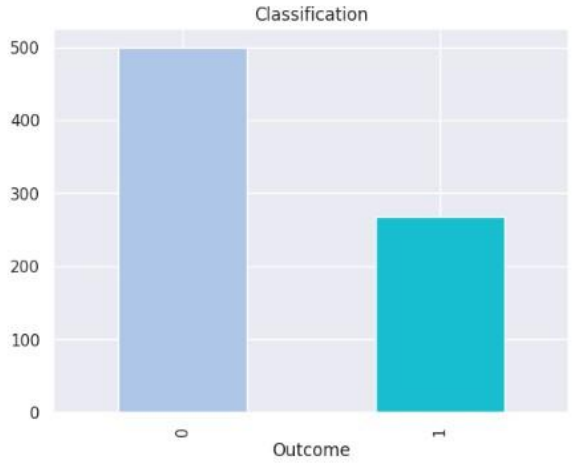

Abbildung 6 4: Darstellung der Class-Imbalance

Die angewandten Methoden zeigen, dass die Korrelation zwischen den einzelnen unabhängigen Variablen sehr gering ist, da alle Werte außer ihrer Diagonale nahe bei 0 liegen. Der Datensatz weist jedoch aufgrund fehlender Werte teilweise eine starke Asymmetrie auf (siehe Tabelle 6.1 und 6.2). Dies hat zur Folge, dass einige Datenpunkte in dem Datensatz fälschlicherweise als Ausreißer identifiziert werden (siehe Tab. 6.2). Zudem existiert ein Ungleichgewicht der Klassen. Auf Basis dieser Informationen und Erkenntnisse aus den Daten sind Vorkehrungen getroffen worden, um die Daten für die Modelle vorzubereiten. Dabei wurden zwei Bereiche des Datensatzes hinsichtlich der Verteilung der Features und der Klasse verarbeitet.

6.2 Datenvorbereitung

Um die Auswirkungen und die bestgeeignete Methode für die Verarbeitung und Bereinigung des Datensatzes zu ermitteln, wurden verschiedene Skalierer und Class-Imbalance-Techniken verwendet, darunter ein Standardisierer, zugleich als Z-Score (StandardScaler) bekannt, sowie ein Normalisierer (MinMaxScaler). Für das Klassenungleichgewicht wurde SMOTE (Syntetic Minority Oversampling Technique) eingesetzt. Der Z-Score skaliert die Daten mit dem Mittelwert bei *0* und der Standardabweichung bei *1*. Der MinMaxScaler normalisiert die Daten in der Weise, dass sie zwischen *0* und *1* liegen (Kasper, B., 2022). Um das vorhandene Klassenungleichgewicht zu reduzie-

ren, wurde SMOTE verwendet. SMOTE erzeugt künstliche Stichproben der Minderheitsklasse durch Interpolation zwischen vorhandenen Stichproben (Fernández, A. et al., 2018).

6.3 Methodischer Ansatz zur Framework-Erstellung

Das resultierende Framework folgt dem Konzept der Ensemble-Methode mit dem Voting Classifier und traditionellen Modellen KNN, SVM und MLP. Eine Besonderheit dieses Frameworks liegt in der expliziten Anwendung der Z-Score-Funktion und der Class-Imbalance-Technik KMeansSMOTE. Die Abbildung 6.5 veranschaulicht dieses Framework auf vereinfachte Weise.

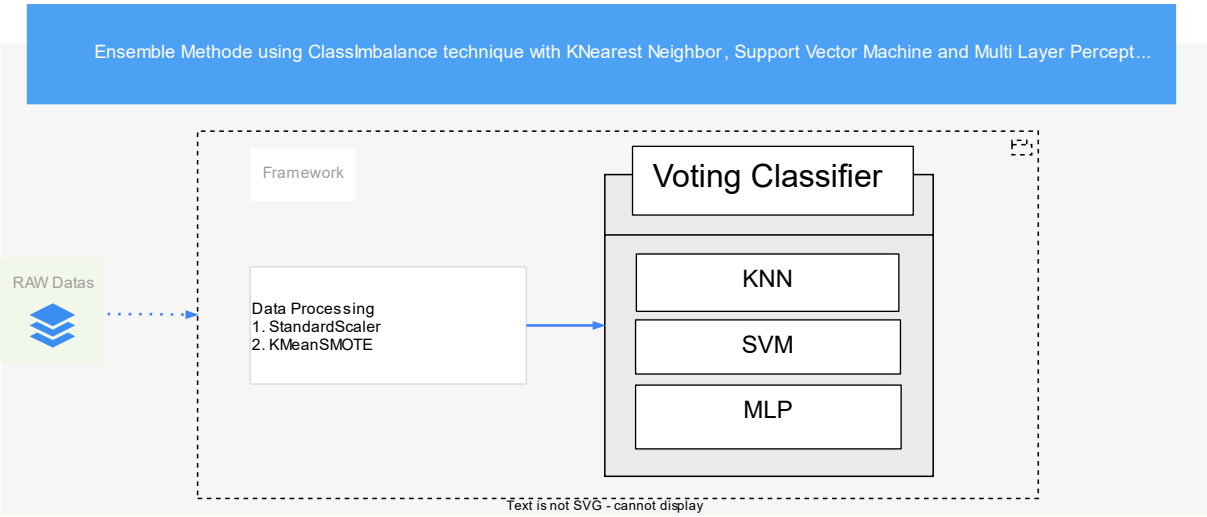

Abbildung 6 5: Graphische Darstellung des Ensemble Framework

Um ein Framework mit diesen besonderen Fähigkeiten zu entwickeln, müssen zunächst die Vorgaben für die Daten definiert werden, die für das Training und das Testen der Modelle verwendet werden sollen. Nach der Analyse der Daten erfolgte die Aufteilung in repräsentierbare Sets nach dem Schema 70/30 (Jirakittayakorn, N. et al., 2024 & Trivedi, V. et al., 2024). Um Verzerrungen vorzubeugen, wurde die Teilung der Daten randomisiert: Xtrain, Xtestfinal, Ytrain und Ytestfinal. Für die Modellbildung wurden ausschließlich der Xtrain und der Ytrain verwendet. Der Xtestfinal und Ytestfinal dienen der abschließenden Bewertung des Frameworks. Der Xtrain-Datensatz wurde erneut in xtrain, xvalidation, ytrain und yvalidation unterteilt.

Der xtrain-Datensatz wurde mittels der Z-Score-Funktion (StandardScaler) skaliert und die Klassenungleichverteilung mittels KMeansSMOTE ausgeglichen. Diese Techniken erwiesen sich als ideale Verfahren für die Modelle, da sich die einzelnen Leistungsfähigkeiten der Modelle KNN, SVM und MLP durch die Anwendung am ehesten verbesserten (siehe Kapitel 6.3). Für die Suche der optimalen Hyperparameter wurden sowohl die Grid- als auch die Random-Search-Methode angewendet. Für die Hyperparameter-Suche der Modelle KNN und SVM wurde die Grid-Search-Methode herangezogen, da der Grid-Search die Hyperparameter für diese Modelle schneller fand als der Random-Search. Die Anzahl der Kreuzvalidierung wurde auf 10 festgelegt, um die Robustheit der Modelle zu gewährleisten. Für das MLP-Modell wurde der Random Search für die Hyperparameter-Suche eingesetzt, da die Kombination der Parameter mit dem Grid-Search anderenfalls zu viel Zeit in Anspruch nehmen würde. Um dennoch die k-fache Kreuzvalidierung im Random Search widerzuspiegeln, wurde die Anzahl der Iterationen anhand der vorhandenen Parameter gewählt. Konkret bedeutet dies, dass die Anzahl der Hyperparameter gezählt und mit einer festgelegten Prozentzahl (25 %) multipliziert wurde. Diese Zahl repräsentierte die Anzahl der Iterationen. Es wurden alle Modelle bis auf den Voting Classifier vor der Zusammenführung trainiert und evaluiert, woraus sich die besten Modelle mit ihren optimalen Hyperparametern ergaben. Die Pipeline wurde unter Verwendung der Voting Classifier mit *hard voting* den entsprechenden Modellen KNN, SVM und MLP und deren optimalen Hyperparametern zusammengestellt. Der Evaluationsmetriken *accuracy*, *recall* und *precision* wurde sich bedient, um die Leistung des Ensemble-Modells zu bewerten.

6.4 Ergebnisse

In diesem Abschnitt werden die wesentlichen Ergebnisse der durchgeführten Experimente präsentiert. Die Ergebnisse sind maßgeblich für die Herleitung des Frameworks. Die einzelnen Modelle werden mit dem Ensemble-Modell verglichen. Der Fokus liegt hierbei auf der anschließenden Interpretation dieser Ergebnisse im Kontext der ursprünglichen Forschungsfragen. Es sei darauf hingewiesen, dass die Experimente nach mehrmaligem Durchlauf und Wiederholungen eine Änderung der Ergebnisse der ML-Modelle bewirken können. Diese Abweichungen können auf die zufällige Verteilung der Daten zurückzuführen sein.

Die Darstellung beginnt mit einer Übersicht der quantitativen und qualitativen Daten, die während der Evaluation gesammelt wurden.

6.4.1 Vergleich der Modelle und Interpretation der Ergebnisse

Die Ergebnisse zeigen den Effekt der Datenvorbereitung und eine Class-Imbalance-Technik sowie die Leistung der einzelne Modelle KNN, SVM und MLP im Vergleich zu dem Ensemble Framework.

In der Tabelle 6.3 wird die Leistung verschiedener Modelle (KNN, SVM und MLP) unter Verwendung von drei verschiedenen Datenverarbeitungstechniken, *ohne Vorbereitung der Daten, MinMax-Normalisierung* und *Z-Score-Skalierung*, untersucht.

Zunächst ist anzumerken, dass bis auf den SVM für die Modelle keine Hyperparameter definiert wurden. Der Hyperparameter *probability* bei SVM wurde auf *true* gesetzt. Die Entscheidung *Recall* und nicht andere Metriken für die Überprüfung der Leistung zu wählen, begründet sich durch die Datenungleichheit. Der *Recall* bewertet die Fähigkeit des Modells, die Minderheitsklasse vorherzusagen. Das Ergebnis zeigt den Effekt der Datenvorbereitung auf die Modelle.

Modelle	Nicht verarbeitet	MinMax	Z-Score
KNN	0,56	0,52	0,53
SVM	0,46	0,51	0,53
MLP	0,44	0,56	0,61

Tabelle 6 3: Modellleistung mit Recall

Der KNN erreicht bei *nicht verarbeiteten Daten* einen Recall-Wert von 0,56, was bedeutet, dass das Modell zu 56 % die tatsächlichen positive Fälle korrekt identifiziert. Bei der *MinMax-Normalisierung* reduziert sich die Leistung um 4 % auf 52 % und unter Verwendung der *Z-Score-Standardisierung* um 3% auf 53 %.

Die SVM erreicht bei unvorbereiteten Daten 46 % und erfährt bei der *MinMax-Normalisierung* 51 %, mithin eine Verbesserung um 5 %, und bei der *Z-Score-Standardisierung* eine Steigerung auf 53 %. Das MLP-Modell erreicht im Vergleich zu SVM und KNN einen schlechteren Recall-Wert von 0,44 bzw. 44 %. Dies bedeutet, dass das Modell lediglich in 44 % der Fälle die tatsächlich positive Klasse erkennt. Eine verhee-

rende Feststellung, wenn bedacht wird, dass eine Fehldiagnose schlimme Folgen haben kann. Durch die *Normalisierung* verbessert sich seine Leistung um 12 % auf 56 % und erfährt bezogen auf die *Standardisierung* sogar eine Steigerung um 17 % auf 61 %.

Das vorliegende Ergebnis deutet darauf hin, dass eine Verschlechterung des KNN-Modells durch die Normalisierung bzw. Standardisierung des Datensatzes nicht ausgeschlossen werden kann. Diese Abweichung kann auf verschiedene Faktoren zurückzuführen sein. Die Datenverteilung sowie die zufällige Aufteilung der Daten-Sets, mit denen das Modell trainiert wurde, können das Verhalten des Modells beeinflussen. Darüber hinaus ist zu berücksichtigen, dass das KNN-Modell aufgrund seiner Funktionsweise selbst zur Verzerrung beitragen kann. Aufgrund der hohen Empfindlichkeit des KNN-Modells gegenüber Wertebereichen (Kasper, B., 2022) ist es möglich, dass einige Merkmale des Datensatzes Ausreißer enthalten (siehe Abb. 6.2), die das KNN-Modell begünstigen. Dies kann dazu führen, dass die Ergebnisse verzerrt werden, da Ausreißer die Modellleistung beeinflussen. Bei der Berechnung der Distanzen zwischen den Punkten kann diese so berechnet werden, dass bestimmte Klassen bevorzugt werden, was wohlmöglich zu einer Erhöhung des Recall-Wertes führt.

Insgesamt bewirken sowohl die Normalisierung als auch die Standardisierung, als Datenvorbereitungstechniken, eine Leistungssteigerung, da beide Techniken die unterschiedlichen Datentypen (siehe Tabelle 6.2 & 6.3) im Datensatz aushebeln. Insbesondere die Z-Score-Standardisierung führt zur Erlangung der besten Ergebnisse bei SVM und MLP. Die Vorbereitung der Daten übt einen positiven Einfluss auf die Fähigkeiten der Modelle aus, da sie verhindert, dass die Modelle während des Trainings Verzerrungen zugunsten einer Klasse erfahren. Im Vergleich zum MinMaxScaler ist der StandardScaler effektiver bei der Reduzierung von Ausreißern, was zu einer zusätzlichen Steigerung der Modellleistung bei der Standardisierung der Daten führt (Kasper, B., 2022). Abschließend ist erkennbar, dass das MLP-Modell von der Vorbereitung der Daten am meisten profitiert, es zeigt die höchste Verbesserung des Recalls, was auf die Anpassungsfähigkeit und Flexibilität dieses Modells hinweist.

Modelle	Unbalancierte Daten	KMeansSMOTE
KNN	0,74	0,79
SVM	0,77	0,82
MLP	0,69	0,81

Tabelle 6 4: Modellleistung mit Class-Imbalance-Technik

Die Tabelle 6.4 illustriert die Treffgenauigkeit (Accuracy) der Modelle KNN, SVM und MLP bei der Vorhersage mit unbalancierten standardisieren Daten und Daten, die mit KMeansSMOTE augmentiert wurden. Die Ergebnisse in Tabelle 6.4 zeigen, dass das KNN für unbalancierte Daten eine Treffgenauigkeit von 0,74 aufweist. Dies bedeutet, dass das Modell 74 % der Fälle richtig vorhersagt. Die Anwendung von KMeansSMOTE resultiert in einer Leistungssteigerung um 5 % auf 79 %. Der SVM hat eine Treffgenauigkeit von 0,77, damit sagt das Modell 77 % der Fälle richtig voraus. Mit der KMeansSMOTE führt dies zu einer Steigerung auf 0,82 bzw. 82 % – eine Leistungssteigerung um 5 %. Im Gegensatz dazu zeigt das MLP-Modell bei unbalancierten Daten eine geringere Leistung von 69 %, wobei gleichzeitig eine Steigerung um 12 % auf 81 % durch die Anwendung von KMeansSMOTE zu verzeichnen ist.

Zusammenfassend lässt sich festhalten, dass die Anwendung von KMeansSMOTE die Leistung aller drei Modelle verbessert, was auf die Vorteile der synthetischen Erzeugung von Samples für die Minderheitsklasse hindeutet. Durch den Ausgleich der gegebenen Klassenungleichgewicht (siehe Abb. 6.4) wird die Varianz der Trainingsdaten erhöht, die die ML-Modelle dabei hilft, robuster und besser zu generalisieren. Insbesondere das MLP-Modell lässt eine signifikante Leistungssteigerung erkennen. Zudem zeigt sich, dass das SVM-Modell sowohl bei unausgewogenen Daten als auch bei synthetisch vergrößerten Daten die höchste Accuracy aufweist. Dies ist auf die Fähigkeit des Modells zurückzuführen, Regulierungsparameter-C zu verwenden, die es ermöglichen die Komplexität des Modells zu steuern und dadurch bei unausgewogenen Daten gut zu generalisieren, auch die Verwendung verschiedene Kernel-Funktionen ermöglichen das SVM, die optimale Entscheidungsgrenzen zu finden, die die Klassen besser trennen.

In der Gesamtbetrachtung lässt sich die Verwendung von KMeansSMOTE als Class-Imbalance-Technik als eine nützliche Methode zur Verbesserung der Modellleistung bei unausgewogenen Daten bzw. unausgeglichenen Klassen darstellen.

Clf	Acc.	Rec.	Pre.	Train Score	Test Score	SMOTE	Time
KNN	0,727	0,725	0,585	0,838	0,727	kmean	0,024
SVM	0,740	0,787	0,594	0,861	0,740	kmean	0,122
MLP	0,722	0,762	0,575	0,855	0,722	kmean	0,740
EMM	0,740	0,787	0,594	0,864	0,740	kmean	4,990

Tabelle 6 5: Leistungsvergleich traditionelle Clfs und Ensemble Framework

Tabelle 6.5 stellt die quantitativen Leistungen der Modelle **KNN**, **SVM** und **MLP** mit denen der Voting Classifier (**Ensemble-Methode modifiziert**), *mit Initialisierung* der Modelle KNN, SVM und MLP mit dem *besten Hyperparameter*, und unter Verwendung der Class-Imbalance-Technik KMeansSMOTE dar. Der Datensatz wurde vor der Anwendung skaliert.

Der **K-Nearest Neighbor** mit K-Means- und StandardScaler-Methoden weist einen Accuracy-Wert von 0,727 auf, darauf hindeutend, dass das Instanz-basierte Modell zu etwa 73 % aller Fälle die richtigen Klassen klassifiziert. Der Recall liegt bei 73 %, die Präzision bei 59 %, der Trainingswert bei 84 % und der Testwert bei 73 %. Ferner beträgt die gesamte Rechenzeit lediglich 0,024 Sekunden. Der KNN erzielt eine moderate Leistung mit einer akzeptablen Präzision und einem Recall. Die Präzision ist im Vergleich jedoch schlechter, das Verfahren ist in 41 % der Fälle nicht in der Lage, die Klassen richtig zu trennen, und die FP-Rate (False Positive Rate) ist weiterhin erhöht. Nichtsdestotrotz demonstriert das Verfahren insgesamt eine gute Anpassung an die Trainingsdaten und eine ähnliche Testgenauigkeit wie beim Training – ein Indiz für eine gute Generalisierbarkeit. Die Gesamtzeit für die Klassifizierung der Daten ist im Vergleich zu den übrigen Modellen sehr kurz, wodurch das Verfahren sich als ein schneller Lernalgorithmus erweist.

Die **Support Vector Machine** (SVM) mit K-Means- und StandardScaler-Methoden weist einen Accuracy-Wert von 74 %, einen Recall von 79 %, eine Präzision von 59 %, einen Trainingswert von 86 % und einen Testwert von 74 % auf. Die Gesamtlaufzeit beträgt 0,122 Sekunden. Im Vergleich zu KNN ist das SVM etwas langsamer, jedoch stets eines der schnellsten Modelle, da diese die Support-Vektoren für die Trennung der Klasse verwendet. Zudem zeigt das ML-Modell eine ähnliche Leistung wie das modifizierte Ensemble-Modell. Allerdings ist seine Präzision, wie im Falle von KNN,

ebenfalls schlechter als die übrigen Werte. In 41 % der Fälle trennt das Modell schlechter als erwartet, was ebenfalls auf eine erhöhte FP hindeutet. Es konnte beobachtet werden, dass sowohl die Trainings- als auch die Testwerte gut aufeinander abgestimmt sind und das Modell weder über- noch unterangepasst ist. Dies lässt die Hypothese zu, dass das Bias-Variance-Trade-Off gut abgestimmt ist.

Der **Multilayer Perceptron** (KMeansSMOTE & StandardScaler) weist eine moderate Leistung auf, die sich in einer akzeptablen Accuracy von 72 %, einem Recall von 76 % und einer Präzision von 56 % manifestiert. Sein Trainingswert beträgt 86 %, während der Testwert bei 72 % liegt, und dies über eine Gesamtzeit von 0,74 Sekunden. Es stellt sich heraus, dass das MLP im Vergleich zu den übrigen Modellen eine geringere Accuracy und einen niedrigeren Recall, jedoch eine relativ gute Generalisierungsfähigkeit aufweist, da sowohl der Trainings- als auch der Testwert aufeinander abgestimmt sind.

Das **Ensemble-Modell „Modifiziert“** (KMeansSMOTE & StandardScaler) weist eine gute Leistung auf. Accuracy (74 %), Recall (78 %), Trainings- (86 %) und Testwert (74 %) sind etwas höher als die Modelle KNN und MLP sowie nahezu identisch mit den Werten der SVM, wobei letztere lediglich um 0,007 % besser ist. Es ist jedoch anzumerken, dass das EMM ebenfalls in 41 % der Fälle nicht in der Lage ist, die Klassen korrekt zu separieren. Gleichzeitig weist es jedoch eine hohe Trainingsgenauigkeit auf, was auf eine verbesserte Anpassung der Modelle hindeutet. Jedoch ist zu beachten, dass die Gesamtlaufzeit des EMM mit 4,990 Sekunden länger ist als die der traditionellen Modelle.

6.4.2 Stärken und Schwächen des Ensemble-Modells

Die Analyse der vorliegenden Daten ergibt, dass das Ensemble-Modell mit der Technik des Class-Imbalance-Learnings KMeansSMOTE eine solide Leistung zeigt. Diese Technik führt zu einer Verbesserung der Recall-Werte und zu einer besseren Balance zwischen Precision und Recall. Der Grund dafür ist, dass durch das Klassengleichgewicht Verzerrungen gegenüber der Majoritätsklasse verhindert, werden können, da beide Klassen die gleiche Anzahl an Samples aufweisen. Somit können die Modelle besser trainiert werden. Die Analyse der Laufzeit insgesamt offenbart deutliche Unterschiede zwischen den Modellen, wobei das KNN-Modell die höchste Geschwindigkeit

aufweist. Das Ensemble-Modell ist im Vergleich am langsamsten, ebenso sind die Leistungen nicht signifikant höher als die anderen Modelle.

7 Fazit und Ausblick

In diesem Kapitel werden Schlussfolgerungen gezogen und ein Ausblick auf die Zukunft gegeben.

7.1 Fazit

Im Rahmen der vorliegenden Arbeit wurden verschiedene Modelle zur Klassifikation untersucht, wobei die Leistungsfähigkeit der Modelle in Bezug auf unbalancierte Datensätze evaluiert wurde. Die primären Ziele der Untersuchung bestanden in einer Bewertung der Modelle hinsichtlich ihrer Genauigkeit, Präzision, Recall und Laufzeit sowie die Untersuchung der Auswirkungen von Class-Imbalance-Techniken wie KMeansSMOTE.

Die Ergebnisse machen ersichtlich, dass Ensemble-Modelle, insbesondere in Kombination mit KMeansSMOTE und StandardScaler, eine verbesserte Leistung im Hinblick auf Recall und eine ausgewogene Balance zwischen Präzision und Recall erzielen konnten. Diese Technik verhindert Verzerrungen gegenüber der Mehrheitsklasse und verbessert folglich die Modellleistung. Das Ensemble-Modell erreicht mit dem SVM eine ähnliche Leistung, wobei das Ensemble-Modell eine erhöhte Präzision aufweist. Dies unterstreicht die Notwendigkeit, die gewählte Vorgehensweise an die spezifischen Anforderungen und Eigenschaften des Datensatzes anzupassen. Es wird deutlich, dass die Leistungen des Ensemble-Modells nicht signifikant über denen traditioneller Modelle liegen. Dies könnte auf den Umstand zurückzuführen sein, dass für die Erstellung des Frameworks dieselben Modelle mit ihren besten Hyperparameter verwendet werden, die zugleich für den Vergleich herangezogen werden. Der Voting Classifier des Typs *hard voting* wählt die Vorhersage desjenigen Modells aus, das die besten Ergebnisse erzielt. In diesem Fall war es der SVM. Ferner ist erkennbar, dass die Treffgenauigkeit der Modelle KNN, SVM und MLP in Tabelle 6.3 besser ausgefallen ist als in Tabelle 6.4 dargestellt. Diese Diskrepanz könnte auf dem Umstand beruhen, dass die Modelle aus der Tabelle 6.3 bei der Suche nach dem optimalen Hyperparameter mithilfe der Grid- oder Random-Search trainiert wurden. Dies erfolgte zudem mittels K-Fold Cross-Validation, dadurch konnte sich ihre Leistungen nach jedem Training und jeder Modellvalidierung verbessert haben.

Zusammenfassend lässt sich festhalten, dass die Anwendung der Class-Imbalance-Technik KMeansSMOTE eine positive Wirkung auf die Modelle hat, indem sie die Minderheitsklassen besser repräsentiert und dadurch die Gesamtleistung der Modelle verbessert. Die Kombination aus Class-Imbalance-Learning und Ensemble-Framework erweist sich als robuster, jedoch nicht signifikant leistungsfähiger als die einzelnen Modelle, da das Ensemble-Framework im Vergleich zu den einzelnen Modellen eine geringere Rechengeschwindigkeit aufweist.

7.2 Ausblick

Im Rahmen der vorliegenden Arbeit wurden verschiedene Erkenntnisse gewonnen, die westliche Beiträge zum Verständnis von Lernmodellen und das KMeansSMOTE liefern. Dieser Ausblick soll eine mögliche Erweiterung dieser Erkenntnisse aufzeigen. Zwei Aspekte werden hierfür beschrieben.

1. **Erweiterung der Modellvielfalt**

Die Modellvielfalt könnte erweitert werden, indem weitere Modelle wie Entscheidungsbäume, Random Forests und Gradient Boosting untersucht werden. Diese nutzen bereits Ensemble-Methoden, die bei der Vorhersage mit unbalancierten Daten gute Ergebnisse liefern.

2. **Erweiterung der Techniken der Datenvorverarbeitung**

Es könnten weitere Techniken der Datenvorverarbeitung und -erweiterung untersucht werden, beispielsweise Techniken des Feature Engineering wie das Hinzufügen von Polynomial Features, die eine Nicht-Linearität in den Datensatz einfügen können oder die Technik der Dimensionsreduktion wie die Hauptkomponenten-Analyse (PCA), die die Dimension des Merkmalsraums reduziert und irrelevante Merkmale entfernt. Beide Techniken können die Modellgenauigkeit erhöhen. Überdies könnte eine Vergrößerung des Datensatzes das Lernverhalten der Modelle positiv beeinflussen, da infolgedessen mehr Daten zum Trainieren und Validieren zur Verfügung stünden. Ferner wird angeregt, die Erforschung und Anwendung neuerer Class-Imbalance-Techniken oder Algorithmen, welche die Vorhersage eines Modells verbessern können, zu erweitern.

Die genannten Ansätze bieten vielversprechende Möglichkeiten, die Leistung der Modelle weiter zu verbessern und die Herausforderungen der unbalancierten Datensätze erfolgreich zu bewältigen.

Literaturverzeichnis

- *KNeighborsClassifier*. (o. D.). Scikit-learn.https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier
- *MLPClassifier.* (o. D.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- Ahsan, M. M., Luna, S. A., & Siddique, Z. (2022, March). *Machine-learning-based disease diagnosis: A comprehensive review. In Healthcare* (Vol. 10, No. 3, p. *5 of 30* 541). MDPI.
- AlShourbaji, I., Kachare, P., Zogaan, W., Muhammad, L. J., & Abualigah, L. (2022). *Learning Features Using an optimized Artificial Neural Network for Breast Cancer Diagnosis*.
- Zhang, L., Wang, Y., Niu, M., Wang, C., & Wang, Z. (2020). *Machine learning for characterizing risk of type 2 diabetes mellitus in a rural Chinese population: The Henan Rural Cohort Study*. *Scientific reports.*
- Khan, A., Qayum, H., Liaqat, R., Ahmed, F., Nawaz, A., & Younis, B. (2021). *Optimised prediction model for type 2 diabetes using gradient boosting algorithm*.
- Samuel, A. L. (1959). Machine learning. *The Technology Review.*
- Fan, Z. E., Lian, F., & Li, X. R. (2025*). Rethinking density ratio estimation based hyper-parameter optimization*. *Neural Networks*, *182*, 106917.
- Burkov, A. (2020). *Machine learning engineering* (Vol. 1). Montreal, QC, Canada: True Positive Incorporated.
- Botsch, B. (2023). *Maschinelles Lernen – Grundlagen und Anwendungen.* Heidelberg Springer
- Bisong, E. (2019). *Building Machine Learning and Deep Learning Models on Google Platform: A comprehensive Guide for Beginners.*
- Sreejith, S., Ajayan, J., Uma Reddy, N. V., Babu Devasenapati, S., & Shashank, R. (2024). *Analysis of COVID-19 CT Chest Image Classification using Dl4jMlp Classifier and Multilayer Perceptron in WEKA Environment*.
- Abdullah, D. M., & Abdulazeez, A. M. (2021). *Machine learning applications based on SVM classification a review. Qubahan Academic Journal.*

- Naganathan, G. S., & Babulal, C. K. (2019). *Optimization of support vector machine parameters for voltage stability margin assessment in the deregulated power system. Soft Computing.*
- Hansot, J., Wongsaroj, W., Sangsuwan, T., & Thong-un, N. (2025). *A low-cost autonomous portable poultry egg freshness machine using majority voting-based ensemble machine learning classifiers*, Smart Agricultural Technology, Volume 10, 100768, ISSN 2772-3755
- Chen, Z., Li, J., & Wei, L. (2007). *A multiple Kernel support vector machine scheme for feature selection and rule extraction from gene expression data of cancer tissue, Artificial Intelligence in Medicine.*
- Kreutzer, T. R. (2019). *Künstliche Intelligenz Verstehen.* Springer
- Selle, S. (2024). *Data Science Training – Supervised Learning. Ein praktischer Einstieg ins überwachte maschinelle Lernen.*
- Kelleher, D. J. (2019). *Deep Learning. The Massachusetts Institute of Technology.*
- Hatem, G., Zeidan, J., Moreira, C., & Goossens, M. (2022). *Normality Testing Methos and the importance of Skewness and Kurtosis in statistical analysis.*
- Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). *Learning from class-imbalanced data: Review of methods and applications, Expert Systems with Applications.*
- Rasheed, K., Qayyum, A., Ghaly, M., Al-Fuqaha, A., Razi, A., & Qadir, J. (2022). *Explainable, trustworthy, and ethical machine learning for healthcare: A survey, Computers in Biology and Medicine.*
- Trivedi, V., Mohseni, A., Lonardi, S., & Wheeldon, I. (2024). *Balanced Training Sets Improve Deep Learning-Based Prediction of CRISPR sgRNA Activity, ACS Synthetic Biology*.
- Jirakittayakorn, N., Wongsawat, Y., & Mitrirattanakul, S. (2024). *An enzyme-inspired specificity in deep learning model for sleep stage classification using multi-channel PSG signals input: Separating training approach and its performance on cross-dataset validation for generalizability.*
- Ickenroth, T., Breschi, V., Oomen, T., & Formetin, S. (2024). *Iterative Feedback Tuning with automated reference model selection.*

- Yu, H., & Ni, J. (2014). *An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data. IEEE/ACM transactions on computational biology and bioinformatics*, *11*(4).
- Kasper, B. (2022). *Getting started with Machine Learning (ML) and Support Vector Classifiers (SVC) – A systematic step-by-step approach: Test and Certification Body for Electrical Engineering at BG ETEM. Version 1.1.*
- Fernández, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018*). SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. Journal of artificial intelligence research.*

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Neutraubling, den 09.03.2025

Sandra Edigin