

**Dokumentation Projekt KI**

**k-nearest Neighbour**

an der Dualen Hochschule Baden-Württemberg Heidenheim

Studiengang Wirtschaftsinformatik

Fakultät Wirtschaft

Matrikelnummern: 2674711, 5089730, 5488044

Kurs: WWI2022/KI-UC

Dozent: Dr. Ralf Höchenberger

Abgabedatum: 06.09.2024

**Inhaltsverzeichnis**

[**1. Allgemeine Informationen 3**](#_py8dj4bvj0te)

[1.1 Beschreibung des Projekts 3](#_g8573dqn3ygp)

[1.2 Beschreibung des Datensatzes 3](#_4sx8go7kwl7p)

[1.3 Installationshinweise 4](#_5p8ib76ngza8)

[1.4. Verwendete Bibliotheken und Pakete 5](#_ocnwi9a068lj)

[**2. KNN-Modell zur Vorhersage von Überlebenden 6**](#_8j2j1moalaco)

[2.1 Erklärung des K-Nearest-Neighbors (KNN) Modells 6](#_n0xcjgo58klr)

[2.2 Vorbereitung des Datensatzes und Datenvorverarbeitung 6](#_ggz0czgj3gvt)

[2.3 Implementierungsschritte und Ergebnisse aus Titanic\_K\_N\_N.ipynb 7](#_jzxgyd8zm403)

[**3. Neuronales Netz als Alternative 11**](#_wasatpb697ez)

[3.1 Aufbau und Funktionsweise eines neuronalen Netzes 11](#_wdn4dbmvydvb)

[3.2 Entscheidungen und Begründungen für die Nutzung von neuronalen Netzen 12](#_ssqj5r8hdmjf)

[3.3 Implementierungsschritte aus Titanic\_Neuron\_Net.ipynb 12](#_75xsm29u5idj)

[**4. Random Forest als zweite Alternative 13**](#_4ex7ronuqxwm)

[4.1 Erklärung des Random Forest Modells 13](#_vhfkyghc6ufn)

[4.2 Entscheidungen und Begründungen für die Nutzung von Random Forest 14](#_mtu71234hvn4)

[4.3 Implementierungsschritte aus Titanic\_R\_D\_F.ipynb 14](#_tqpknas9u0g8)

[**5. Zusammenfassung 15**](#_x28k4gcgaasw)

# **1. Allgemeine Informationen**

## **1.1 Beschreibung des Projekts**

Der Zweck dieses Projekts ist es, mithilfe maschineller Lernmethoden vorherzusagen, welche Passagiere das Titanic-Unglück überleben und welche nicht. Die Aufgabe besteht darin, das K-Nearest-Neighbors (KNN)-Modell zu verwenden oder auch alternative Modelle wie neuronale Netze oder Random Forest zu analysieren, um ihre Eignung für diese Klassifikationsaufgabe zu bewerten. Das Ziel ist es, die Modelle zu vergleichen und herauszufinden, welches die höchste Vorhersagegenauigkeit liefert, um den Überlebensstatus der Passagiere basierend auf deren Merkmalen möglichst präzise zu bestimmen.

## **1.2 Beschreibung des Datensatzes**

Der Titanic-Datensatz enthält Informationen über die Passagiere, die auf der Titanic gereist sind, und ob sie das Unglück überlebt haben. Der gesamte Datensatz der Titanic besteht ursprünglich aus 1309 Passagieren und wurde in zwei Datensätze unterteilt: einen Trainingsdatensatz mit 891 Einträgen und einen Testdatensatz mit 418 Einträgen. Beide Datensätze sind strukturell identisch, mit der Ausnahme, dass im Testdatensatz die Spalte Survived fehlt, da er zur Validierung der Vorhersagen der Modelle verwendet wird. Die wesentlichen Spalten des Training-Datensatzes umfassen:

* PassengerId: Eindeutige ID des Passagiers
* Survived: Überlebensstatus (0 = Nicht überlebt, 1 = Überlebt)
* Pclass: Ticketklasse (1 = Erste Klasse, 2 = Zweite Klasse, 3 = Dritte Klasse)
* Name: Name des Passagiers
* Sex: Geschlecht des Passagiers
* Age: Alter des Passagiers
* SibSp: Anzahl der Geschwister und Ehepartner an Bord
* Parch: Anzahl der Eltern und Kinder an Bord
* Ticket: Ticketnummer des Passagiers
* Fare: Preis des Tickets
* Cabin: Kabinennummer
* Embarked: Hafen, an dem der Passagier zugestiegen ist (C = Cherbourg, Q = Queenstown, S = Southampton)

Ein wesentliches Merkmal des Trainingsdatensatzes ist, dass einige Daten unvollständig sind. Beispielsweise fehlen in der Age-Spalte 177 Einträge, sodass nur 714 von 891 Passagieren ein Alter zugeordnet werden kann. Besonders problematisch ist die Spalte Cabin, in der nur für 204 der 891 Passagiere eine Kabinennummer vorliegt. Auch die Spalte Embarked ist nicht vollständig, hier fehlen zwei Einträge.

Statistisch gesehen liegt das Durchschnittsalter der Passagiere bei etwa 29,7 Jahren, wobei das Alter von nur wenigen Monaten bis hin zu 80 Jahren variiert. Die meisten Passagiere reisten in der dritten Klasse, während der durchschnittliche Ticketpreis bei 32,20 Britischen Pfund lag und sich zwischen 0 und 512,33 Pfund bewegt. Von den 891 Passagieren überlebten etwa 38%, während 62% das Unglück nicht überstanden. Die Anzahl der mitreisenden Geschwister, Ehepartner sowie Eltern und Kinder variiert stark, wobei die Mehrheit der Passagiere allein reiste.

Der Trainingsdatensatz enthält sowohl numerische als auch kategorische und textliche Daten. Beispielsweise sind PassengerId, Age, SibSp, Parch und Fare numerische Werte, während Name, Sex, Cabin und Embarked kategorische oder textliche Daten darstellen. Diese Vielfalt erfordert eine sorgfältige Vorverarbeitung, insbesondere bei den fehlenden Werten und der Umwandlung der nicht-numerischen Daten in numerische Formate, um sie für maschinelle Lernmodelle nutzbar zu machen.

## **1.3 Installationshinweise**

Für die Ausführung sind einige Installationen und Voraussetzungen erforderlich. Zunächst muss Visual Studio auf dem System installiert sein, da es als Entwicklungsumgebung für die Ausführung und Verwaltung der Jupyter Notebooks dient. Zudem ist die Jupyter Notebook Extension notwendig, um die .ipynb-Dateien direkt in Visual Studio zu öffnen und auszuführen.

Die weiteren erforderlichen Python-Bibliotheken und Pakete werden direkt im Notebook als erster Ausführungsschritt installiert und müssen nur ausgeführt werden.

## **1.4. Verwendete Bibliotheken und Pakete**

Im Projekt wurden verschiedene Bibliotheken und Pakete verwendet, die jeweils spezifische Aufgaben übernehmen. Hier ist eine kurze Erklärung der wichtigsten Bibliotheken und ihrer Verwendung:

**Pandas** ist eine Bibliothek zur Datenmanipulation und Analyse. Sie wurde verwendet, um den Titanic-Datensatz zu laden, zu verarbeiten und zu analysieren. Insbesondere hilft Pandas bei der Handhabung von tabellarischen Daten.

**Scikit-learn** ist eine der am häufigsten verwendeten Bibliotheken für maschinelles Lernen. Sie bietet eine Vielzahl von Modellen und Werkzeugen für Klassifikation, Regression, Clustering, Modellbewertung und Hyperparameter-Tuning. In diesem Projekt wurde sie zur Implementierung und Optimierung des K-Nearest-Neighbors-Algorithmus sowie für das Modelltraining, die Datenaufteilung und die Berechnung der Genauigkeit sowie für das Random Forest-Modell verwendet.

**Matplotlib** ist eine Bibliothek zur Erstellung von Diagrammen und Visualisierungen. Sie wurde verwendet, um die Modellgenauigkeit in Abhängigkeit von verschiedenen K-Werten (Nachbarn) zu visualisieren und die Performance des Modells grafisch darzustellen.

**TensorFlow** ist eine umfassende Bibliothek für maschinelles Lernen und künstliche Intelligenz. Hier wurde sie verwendet, um mit Keras neuronale Netzwerke zu erstellen und zu trainieren.

Die **itertools**-Bibliothek bietet effiziente Tools für Iteratoren und wurde genutzt, um alle möglichen Kombinationen von Features für die Optimierung des KNN-Modells zu erstellen.

# **2. KNN-Modell zur Vorhersage von Überlebenden**

## **2.1 Erklärung des K-Nearest-Neighbors (KNN) Modells**

Das K-Nearest-Neighbors (KNN)-Modell ist ein einfaches und leicht verständliches Modell im Bereich des überwachten maschinellen Lernens. Es funktioniert, indem es für jeden Datenpunkt die K den nächsten Nachbarn in einem mehrdimensionalen Raum sucht und den Mehrheitswert der Nachbarn als Vorhersageklasse nimmt.

Für die Titanic-Daten schien KNN ein sinnvoller Ansatz zu sein, da es die Ähnlichkeit zwischen Passagieren nutzen kann, um deren Überlebenschancen vorherzusagen. KNN ist jedoch empfindlich gegenüber der Wahl der Features und der Anzahl der Nachbarn (K), weshalb eine Optimierung notwendig wurde.

## **2.2 Vorbereitung des Datensatzes und Datenvorverarbeitung**

Zu Beginn haben wir den Trainingsdatensatz eingelesen und mit der Datenvorverarbeitung begonnen. Wir haben uns zuerst entschieden, einige Features zu entfernen, die wenig oder keinen Einfluss auf das Überleben der Passagiere haben könnten, wie Cabin, Ticket und Name. Diese Spalten wurden entfernt, um den Fokus auf die relevanteren Merkmale zu legen, was dazu beiträgt, das Modell übersichtlicher und effizienter zu gestalten. Die Zielvariable Survived wurde ebenfalls separiert, da sie die Vorhersageklasse für das Modell ist.

Kategorische Variablen wie Sex und Embarked mussten in numerische Werte umgewandelt werden, da maschinelle Lernmodelle mit numerischen Daten arbeiten. Die Umwandlung wurde über ein Mapping durchgeführt (Sex: 0 für männlich und 1 für weiblich; Embarked: 0 für Q, 1 für S, 2 für C).

Um die Datenqualität zu sichern, haben wir fehlende Werte entfernt. Dies ist notwendig, da maschinelle Lernmodelle mit vollständigen Daten arbeiten müssen, um zuverlässige Vorhersagen zu treffen.

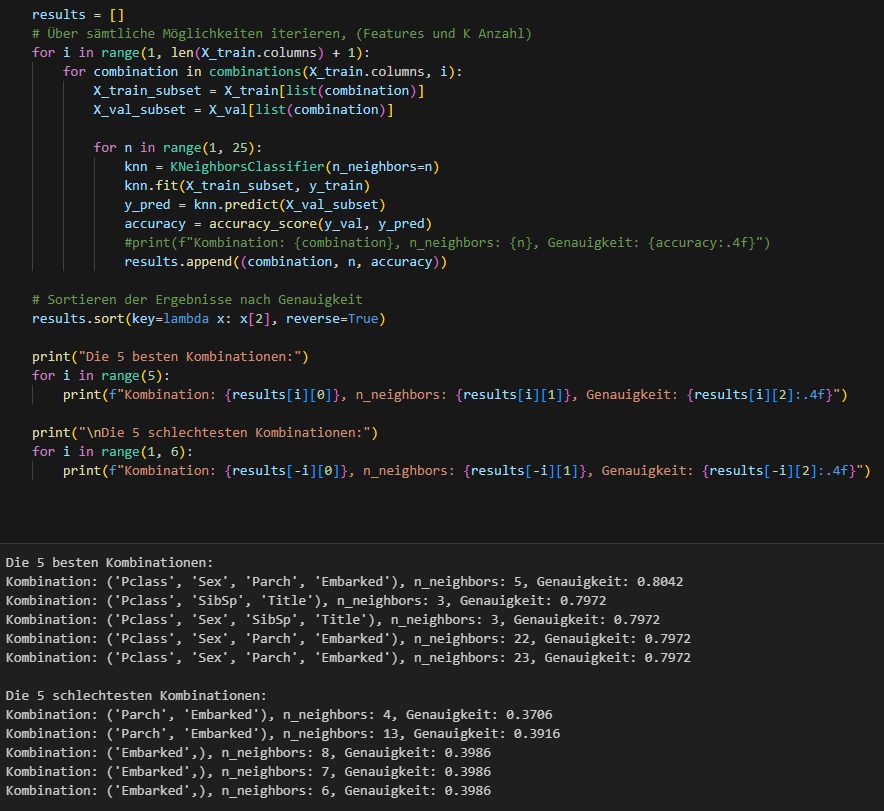
Danach haben wir den Datensatz in Trainings- und Testdaten aufgeteilt. Dies erfolgte im Verhältnis 80/20, wobei 80 % der Daten für das Training des Modells verwendet wurden und 20 % für die spätere Validierung und Genauigkeitsbewertung. Wir verwendeten train\_test\_split mit einem festen random\_state von 42, um eine Reproduzierbarkeit zu gewährleisten.

## **2.3 Implementierungsschritte und Ergebnisse aus Titanic\_K\_N\_N.ipynb**

**Erste Implementierung des KNN-Modells:**

Wir haben das KNN-Modell zunächst mit den Standardparametern implementiert, um eine Grundgenauigkeit zu erhalten. Es wurde das Modell mit den Trainingsdaten trainiert und auch auf diesem validiert. Die erreichte Genauigkeit lag bei 63,64 %. Dies zeigt, dass ohne weitere Optimierung das Modell nur mäßig gut funktioniert.

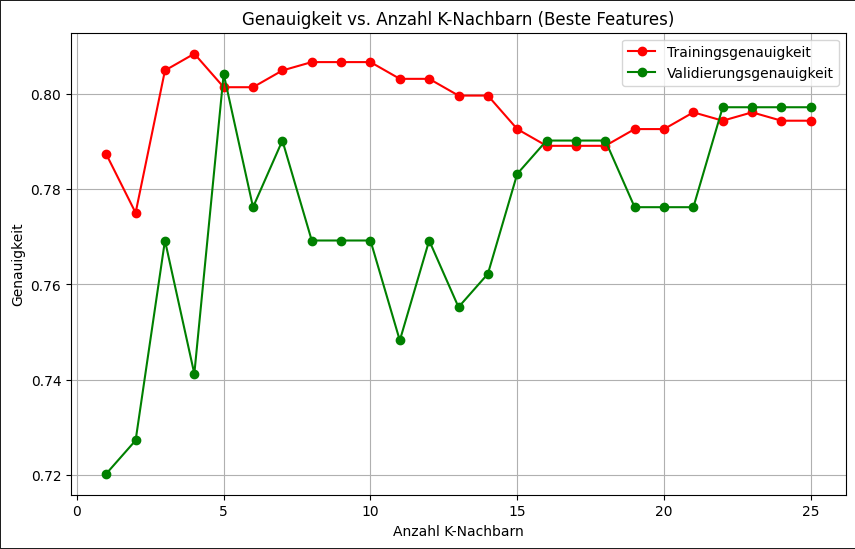
**Optimierung des KNN-Modells:**

Zuerst haben wir aus dem Namen die Titel extrahiert, um eine neue Informationsspalte zu generieren. Danach haben wir verschiedene Spalten entfernt, um zu ermitteln, wie sich die Genauigkeiten dabei ändern. Da es allerdings viele Kombinationen mit unterschiedlichen Anzahl K gab, haben wir uns entschieden, mittels eines Algorithmus die beste Kombination der Features mit der besten Anzahl K zu finden. Dazu haben wir alle möglichen Kombinationen der Features generiert und jede dieser Kombinationen mit verschiedenen K-Werten getestet. Die jeweils besten und schlechtesten 5 Ergebnisse haben wir dabei gespeichert.

Der iterative Prozess zur Optimierung zeigte, dass bestimmte Merkmalskombinationen und K-Werte deutlich bessere Ergebnisse erzielten. Beispielsweise erreichten die besten Kombinationen (Pclass, Sex, Parch, Embarked) mit n\_neighbors=5 eine Genauigkeit von etwa 80,42 % auf dem Validierungsset. Die Passagierklasse (Pclass) war eng mit dem sozialen und wirtschaftlichen Status der Passagiere verbunden. Passagiere der 1. Klasse hatten bessere Kabinen und mehr Zugang zu Rettungsbooten, während Passagiere der 3. Klasse oft in tieferen Decks untergebracht waren, was ihren Zugang zu Rettungsbooten erschwerte. Auch das Geschlecht (Sex) spielte eine zentrale Rolle, da Frauen und Kinder während der Evakuierung bevorzugt gerettet wurden. Frauen hatten dadurch eine deutlich höhere Überlebensrate als Männer, was das Geschlecht zu einem der stärksten Prädiktoren macht. Die Anzahl der Eltern oder Kinder an Bord (Parch) könnte ebenfalls die Reaktionen der Passagiere im Notfall beeinflusst haben, da Familien tendenziell zusammenblieben und dadurch höhere Überlebenschancen hatten. Schließlich gibt der Einschiffungshafen (Embarked) Aufschluss über den sozialen Hintergrund der Passagiere. Viele Passagiere, die in Queenstown (Q) einstiegen, kamen aus der Arbeiterklasse und reisten in der 3. Klasse, was ihre Überlebenschancen verringerte, während wohlhabende Passagiere aus Cherbourg (C) und Southampton (S) bessere Bedingungen und höhere Überlebenschancen hatten.

**Visualisierung der besten Anzahl an Nachbarn:**

Um die Auswirkungen von K auf die Modellgenauigkeit besser zu verstehen, haben wir eine grafische Darstellung der Trainings- und Validierungsgenauigkeit in Abhängigkeit von der Anzahl der Nachbarn erstellt. Es zeigt sich, dass die optimale Anzahl von Nachbarn bei 5 lag, was die beste Balance zwischen Trainings- und Validierungsgenauigkeit darstellte und bestätigt auch unseren oben erstellten Algorithmus..



**Hyperparameter-Tuning:**

Nach der manuellen Optimierung haben wir GridSearchCV verwendet, um eine automatisierte Hyperparameteroptimierung durchzuführen. Dabei wurden neben der Anzahl der Nachbarn auch verschiedene Gewichtungsstrategien (z.B. uniform, distance) und Distanzmetriken (z.B. euclidean, manhattan) getestet. Obwohl die beste Kombination eine Genauigkeit von 76,92 % auf dem Validierungsdatensatz erreichte, übertraf sie die vorherige manuelle Optimierung nicht. Dies zeigt, dass GridSearch nicht immer zu besseren Ergebnissen führt, insbesondere wenn bereits eine manuelle Optimierung erfolgreich war.

**Validierung des Modells auf dem Testdatensatz:**

Nach der Optimierung haben wir das Modell mit den besten Parametern auf den unabhängigen Testdatensatz angewendet. Die Testdaten wurden ebenfalls vorverarbeitet und auf die relevanten Features reduziert. Die Vorhersagen wurden in einer neuen Excel (titanic\_test\_results.csv) gespeichert und mit den tatsächlichen Überlebensdaten verglichen, die in einer separaten Datei (titanic\_validate.csv) gespeichert waren. Mit einer Genauigkeit von 77,64 % auf den Testdaten bestätigte das Modell seine solide Leistung, lag aber leicht unter der Validierungsgenauigkeit 80,42 %, was eine gewisse Varianz im Modell widerspiegelt.

**Abfragetool für einzelne Vorhersagen:**

Zum Schluss haben wir mithilfe von KI ein benutzerfreundliches Abfragetool erstellt, mit dem Benutzer die Überlebenswahrscheinlichkeit eines bestimmten Passagiers basierend auf dessen Passenger-ID abfragen können. Dieses Tool gibt neben der Vorhersage auch die tatsächlichen Überlebensdaten aus und zeigt an, ob die Vorhersage korrekt war. Dies veranschaulicht die Vorhersagekraft des Modells auf individueller Basis. Es kann allerdings bei bestimmten Passagieren vorkommen, dass aufgrund von fehlenden Werten nicht überprüft werden kann, ob die Vorhersage richtig oder falsch ist.

# **3. Neuronales Netz als Alternative**

## **3.1 Aufbau und Funktionsweise eines neuronalen Netzes**

Neuronale Netze sind eine leistungsfähige Methode des maschinellen Lernens, die durch ihre Struktur und Funktionsweise in der Lage sind, komplexe Muster in Daten zu erkennen. Ein neuronales Netz besteht aus Schichten von Neuronen, wobei jede Schicht spezifische Aufgaben übernimmt:

**Eingabeschicht**: Diese Schicht empfängt die Eingabedaten und leitet sie an die nächste Schicht weiter. Im Fall des Titanic-Datensatzes enthält die Eingabeschicht Merkmale wie Pclass, Sex, Age, SibSp, Parch und Embarked.  
**Verborgene Schichten:** Diese Schichten führen die eigentliche Berechnung durch. In unserem Modell haben wir zwei verborgene Schichten verwendet. Jede verborgene Schicht besteht aus einer bestimmten Anzahl von Neuronen, die mittels Aktivierungsfunktionen wie ReLU (Rectified Linear Unit) die Eingaben verarbeiten und nicht-lineare Transformationen durchführen.  
**Ausgabeschicht:** Diese Schicht gibt das Ergebnis der Berechnungen aus. Da unser Problem eine Klassifikationsaufgabe ist, haben wir eine Ausgabeschicht mit zwei Neuronen verwendet, die eine Softmax-Aktivierungsfunktion nutzen. Diese Funktion berechnet Wahrscheinlichkeiten für die beiden Klassen (Überlebt oder nicht Überlebt) und sorgt dafür, dass die Summe der Ausgaben gleich 1 ist.

## **3.2 Entscheidungen und Begründungen für die Nutzung von neuronalen Netzen**

Die Entscheidung, ein neuronales Netz als Alternative zum K-Nearest-Neighbors (KNN)-Modell zu verwenden, wurde aus mehreren Gründen getroffen. Neuronale Netze sind in der Lage, komplexe und nicht-lineare Beziehungen in den Daten besser zu modellieren. Da der Titanic-Datensatz eine Vielzahl von Merkmalen enthält, deren Interaktionen möglicherweise nicht linear sind, können neuronale Netze diese Komplexitäten effizienter erfassen als das KNN-Modell, das primär auf der Ähnlichkeit zwischen Datenpunkten basiert. Zudem bieten neuronale Netze die Möglichkeit, tiefere und komplexere Beziehungen zwischen den Merkmalen zu lernen, da sie durch ihre Schichtenstruktur die Interaktionen zwischen den Features umfassender berücksichtigen. Ein weiterer Vorteil liegt in der Skalierbarkeit neuronaler Netze, die durch zusätzliche Schichten und Neuronen an komplexere Datenmengen angepasst werden können, was das Potenzial bietet, das Modell weiter zu verfeinern und zu verbessern.

## **3.3 Implementierungsschritte aus Titanic\_Neuron\_Net.ipynb**

Zunächst haben wir die wieder relevanten Merkmale ausgewählt und irrelevante Spalten wie Cabin, Ticket und Name entfernt, um den Fokus auf die wichtigsten Informationen zu legen. Wir haben die kategorischen Variablen Sex und Embarked in numerische Werte umgewandelt, da dies für das neuronale Netz erforderlich ist. Fehlende Werte wurden eliminiert, da neuronale Netze vollständige Datensätze benötigen, um robuste Modelle zu trainieren.

Die Zielvariable Survived haben wir in eine kategorische Form umgewandelt, da unser neuronales Netz die Eingabedaten in eine von zwei Klassen einordnen muss. Danach teilten wir den Datensatz in Trainings- und Validierungsdaten auf, um die Leistung des Modells sowohl auf bekannten als auch auf unbekannten Daten zu bewerten.

Das neuronale Netz haben wir mithilfe des Sequential-Modells von Keras aufgebaut. Die erste verborgene Schicht bestand aus 32 Neuronen mit der ReLU-Aktivierungsfunktion, da wir der Überzeugung waren, dass eine ausreichende Anzahl an Neuronen die Fähigkeit des Modells erhöht, komplexe Muster zu erkennen. In der zweiten verborgenen Schicht nutzten wir 16 Neuronen, um die Modellkomplexität zu reduzieren und Überanpassung zu vermeiden. Die Ausgabeschicht umfasste 2 Neuronen mit der Softmax-Aktivierungsfunktion, um die Klassifikationswahrscheinlichkeiten für die beiden Klassen zu berechnen.

Für das Training entschieden wir uns für den Adam-Optimierer, da er durch seine adaptive Lernrate und Effizienz eine schnelle Konvergenz ermöglicht. Als Verlustfunktion wählten wir categorical\_crossentropy, da diese für Klassifikationsprobleme mit mehreren Klassen besonders geeignet ist. Unser Modell trainierten wir über 50 Epochen mit einer Batch-Größe von 32, um einen guten Kompromiss zwischen Speichernutzung und Rechenaufwand zu erzielen und die Stabilität des Lernprozesses sicherzustellen.

Die Ergebnisse des Trainings zeigten eine anfängliche niedrige Genauigkeit, die sich jedoch im Laufe der Epochen stetig verbesserte. Die finale Validierungsgenauigkeit von etwa 77,6 % deutet darauf hin, dass unser Modell die Überlebenswahrscheinlichkeit der Passagiere recht gut vorhersagen kann, obwohl noch Raum für weitere Verbesserungen besteht.

# **4. Random Forest als zweite Alternative**

## **4.1 Erklärung des Random Forest Modells**

Random Forest ist ein leistungsstarker und vielseitiger Klassifikationsalgorithmus, der auf der Idee basiert, viele Entscheidungsbäume zu kombinieren, um genauere und stabilere Vorhersagen zu treffen. Der Random Forest-Algorithmus funktioniert folgendermaßen:

**Bootstrapping:** Der Algorithmus erstellt mehrere Teilmengen des Trainingsdatensatzes durch zufällige Ziehung mit Zurücklegen. Diese Teilmengen werden als Trainingsdaten für die einzelnen Entscheidungsbäume verwendet.

**Erstellung der Entscheidungsbäume:** Für jede dieser Teilmengen wird ein Entscheidungsbaum trainiert. Während des Trainings wird für jede Entscheidung in einem Baum nur eine zufällige Teilmenge der Merkmale betrachtet. Dies sorgt für eine größere Vielfalt unter den Bäumen und hilft, Overfitting zu reduzieren.

**Aggregierung der Ergebnisse:** Nachdem alle Entscheidungsbäume trainiert sind, werden ihre Vorhersagen aggregiert, um die endgültige Entscheidung zu treffen. Bei Klassifikationsproblemen geschieht dies durch Mehrheitsabstimmung, bei der die Klasse gewählt wird, die die meisten Bäume als Vorhersage haben.

## **4.2 Entscheidungen und Begründungen für die Nutzung von Random Forest**

Die Wahl von Random Forest als Alternative zum K-Nearest-Neighbors (KNN)-Modell basiert auf mehreren Überlegungen. Random Forest zeichnet sich durch eine höhere Robustheit gegenüber Überanpassung aus, da es weniger anfällig für Überanpassung ist im Vergleich zu KNN. Während KNN stark von der Wahl der Parameter und der Auswahl der nächsten Nachbarn beeinflusst wird, nutzt Random Forest die Kombination vieler Entscheidungsbäume, um stabilere und verlässlichere Vorhersagen zu treffen.

Ein weiterer Vorteil von Random Forest ist seine Fähigkeit, große und komplexe Datensätze effizient zu verarbeiten. Im Gegensatz zu KNN, das auf die Berechnung der Ähnlichkeiten zwischen allen Datenpunkten angewiesen ist, erstellt Random Forest eine Vielzahl von Entscheidungsbäumen und aggregiert deren Ergebnisse. Diese Methode führt zu einem besseren Generalisierungsvermögen und einer besseren Handhabung großer Datenmengen.

Zusätzlich bietet Random Forest die Möglichkeit, die Bedeutung einzelner Merkmale zu bewerten. Diese Eigenschaft ist besonders nützlich für die Feature-Auswahl und -Interpretation, insbesondere wenn der Datensatz eine Vielzahl von Merkmalen enthält.

## **4.3 Implementierungsschritte aus Titanic\_R\_D\_F.ipynb**

Zunächst haben wir die CSV-Datei titanic\_training.csv in ein Pandas DataFrame geladen. Anschließend teilten wir den Datensatz in Trainings- und Validierungsdaten auf, wobei wir 20 % der Daten für die Validierung reservierten. Um die beste Anzahl an Entscheidungsbäumen (n\_estimators) für unser Random Forest-Modell zu bestimmen, implementierten wir eine Schleife, die die Genauigkeit des Modells für verschiedene Baumanzahlen von 10 bis 2110 testete.

Für jede Anzahl von Bäumen trainierten wir ein Random Forest-Modell und berechneten die Genauigkeit auf dem Validierungsdatensatz. Die beste Anzahl an Bäumen ermittelten wir durch den Vergleich der Genauigkeiten. Es stellte sich heraus, dass 610 Bäume die höchste Genauigkeit von 81,12 % erzielten.

Wir trainierten das finale Modell mit der besten Anzahl an Bäumen und überprüften die endgültige Genauigkeit auf dem Validierungsdatensatz erneut. Das finale Modell erreichte ebenfalls eine Genauigkeit von 81,12 %, was die robuste und zuverlässige Leistung unseres Random Forest-Modells für den Titanic-Datensatz bestätigte.

# **5. Zusammenfassung**

Im Rahmen des Projekts zur Vorhersage des Überlebens von Titanic-Passagieren wurden drei verschiedene maschinelle Lernmethoden untersucht, um die Vorhersagegenauigkeit zu bewerten und deren Eignung für die Klassifikationsaufgabe zu vergleichen: K-Nearest Neighbors (KNN), neuronale Netze und Random Forest.

Das KNN-Modell wurde zunächst mit den Standardparametern implementiert und erzielte eine Grundgenauigkeit von 63,64 %. Durch gezielte Optimierung der Feature-Kombinationen und der Anzahl der Nachbarn K konnte die Genauigkeit auf über 80 % gesteigert werden. Die Verwendung von GridSearchCV zur automatisierten Hyperparameteroptimierung ergab eine leicht geringere Genauigkeit als die manuelle Optimierung. Das Modell erreichte eine Testgenauigkeit von 77,64 %. Zusätzlich wurde ein Abfragetool entwickelt, das es ermöglichte, die Überlebenswahrscheinlichkeit einzelner Passagiere basierend auf deren Passenger-ID abzufragen, wodurch die Modellleistung auf individueller Ebene veranschaulicht wurde.

Für die Untersuchung der neuronalen Netze wurde ein Modell aufgebaut, das komplexe Muster in den Daten erkennen kann. Obwohl die Anfangsgenauigkeit des Modells niedrig war, konnte sie über die Trainingsphasen hinweg auf etwa 77,6 % verbessert werden.

Der Random Forest-Algorithmus zeigte sich als besonders leistungsfähig. Durch das Aggregieren vieler Entscheidungsbäume konnte eine hohe Genauigkeit erreicht werden. Die Implementierung ergab, dass 610 Entscheidungsbäume die beste Genauigkeit von 81,12 % lieferten. Random Forest bewährte sich durch seine Fähigkeit, große Datensätze zu verarbeiten und die Bedeutung einzelner Merkmale zu bewerten.

Zusammenfassend lässt sich sagen, dass der Random Forest-Algorithmus in diesem Projekt als das leistungsfähigste Modell hervorging, gefolgt von KNN und neuronalen Netzen.