exentra

Technische Hochschule Ingolstadt

Technische Hochschule Ingolstadt

Fakultät Elektrotechnik und Informatik

Studiengang Wirtschaftsinformatik

Bachelorarbeit

Anwendung von Künstlicher Intelligenz zur Vorhersage von Kennzahlen im Projektcontrolling

vorgelegt von

Marc Leon Stiglmayr
Matrikel-Nr.: 00080847

Ausgegeben am:	17.10.2019
Abgegeben am:	15.01.2020
Erstprüfer:	Prof. Dr. Melanie Kaiser
Zweitprüfer:	Prof. Dr. Jochen Rasch

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AC	Actual Cost
ACWP	Actual Cost of Work Performed
BAC	Budget at Completion
BCWP	Budgeted Cost of Work Performed
BCWS	Budgeted Cost of Work Scheduled
CPI	Cost Performance Index
CRISP-DM	Cross Industry Standard Process for Data Mining
CSV	Comma-separated values
DIN	Deutsche Institut für Normung
DM	Data Mining
EAC	Estimate at Completion
EV	Earned Value
EVM	Earned Value Management
KI	Künstliche Intelligenz
KDD	Knowledge Discovery in Databases
KNN	Künstliches Neuronales Netz
ML	Machine Learning
MSE	Mean-Squared-Error
PM	Projektmanagement
PMI	Project Management Institute
PSP	Projektstrukturplan
PV	Planned Value
ReLU	Rectified Linear Unit
SPI	Schedule Performance Index
VAC	Variance at Completion
WR	Work Remaining

1 Einführung

1.1 Motivation

Künstliche Intelligenz (KI) ist einer der meist verwendeten Begriffe, sobald es sich um zukunftsweisende Technologien handelt. Von intelligenten Maschinen, über Sprachassistenzsysteme, bis hin zu autonomen Fahrzeugen, KI erobert vermehrt unsere Alltagswelt und dessen Potential scheint in manchen Anwendungsgebieten erst entdeckt worden zu sein.

So ist nach einer Studie von Gartner [vgl. Gartner, 2019] in den letzten vier Jahren die Anzahl der Unternehmen, die künstliche Intelligenz im Einsatz haben um 270 Prozent gestiegen. Allein innerhalb eines Jahres hat sich der Einsatz verdreifacht. Ausgelöst wird dieser Anstieg laut Gartner durch Big Data und Technologieriesen wie Google, Facebook oder Amazon, die als Vorreiter im Bereich künstlicher Intelligenz angesehen werden und dabei hohe Gelder in dessen Forschung investieren. Google veröffentliche beispielsweise im Jahr 2015 eine für den internen Bedarf entwickelte Python-Bibliothek namens TensorFlow[1] als Open Source Software, welche auf maschinelles Lernen spezialisiert ist [vgl. Diedrich, 2015]. Die maschinellen Lernverfahren der KI bieten Möglichkeiten, aus großen Datenmengen komplexe Zusammenhänge automatisiert erkennen zu können und damit einen Ansatz, Big Data als Wettbewerbsvorteil zu nutzen [vgl. Fraunhofer-Allianz Big Data, 2017].

Ein Bereich, der bisher noch scheinbar unberührt von KI erscheint, stellt das Projektmanagement dar. Durch die Digitalisierung und den Einsatz effizienter Businesssoftware zur Unterstützung des Projektcontrollings stehen große Mengen an projektbezogenen Daten zur Verfügung, die in Relation zueinander Rückschlüsse auf den Erfolg eines Projektes liefern können. Dennoch überschreiten knapp die Hälfte aller Projekte ihren Budget- und Zeitrahmen und verschwenden dabei aufgrund der zu ungenauen Projektplanung und -steuerung 119 Millionen Dollar je Billion Dollar die investiert wurden, wie die Studie Global Project Management Survey von PMI zeigt [vgl. PMI, 2019b]. Die wachsenden Datenmengen und damit verbundene Komplexität stellt für Projektmanager eine Herausforderung dar, die hohe Informationsdichte mithilfe von klassischen Projektmanagement-Methodiken gewinnbringend in Projekten einzusetzen. Hierbei zeigt künstliche Intelligenz ihre Stärke. Vor allem im Projektcontrolling ist die Fähigkeit, komplexe Sachverhalte schnell zu erfassen um darin Muster zu erkennen, essenziell zur Überwachung und Steuerung eines Projektes. Durch die Mustererkennung können dabei Vorhersagen bezüglich Kosten- und Zeitverlauf eines Projektes getroffen werden, die zur Entscheidungsunterstützung im Projektmanagement beitragen. Experten sind sich einig, dass die einhergehende Disruption, welche die künstliche Intelligenz mit sich bringt, zu einer Veränderung im Projektmanagement führt und über die nächsten drei Jahre der Anteil der Projekte, die durch KI unterstützt werden, von 23 auf 37 Prozent steigen wird [vgl. PMI, 2019a].

Im Rahmen dieser Arbeit wird ein KI-Modell, in Form eines künstlichen neuronalen Netzes entwickelt, um dieses zur Vorhersage der Projektendkosten eines laufenden Projektes einzusetzen. Die Performance des KI-Modells wird dabei mit klassischen Projektcontrolling-Methoden, wie der Earned Value Management Prognose verglichen. Dabei soll gezeigt werden, ob künstliche Intelligenz einen Mehrwert zur Entscheidungsunterstützung in Projekten, bezogen auf klassische Verfahren des Projektcontrollings, liefert.

[1] https://www.tensorflow.org

1.2 Vorgehen und Zielsetzung

Zu Beginn werden die für diese Arbeit nötigen Grundlagen vorgestellt. Diese werden in einen wirtschaftlichen und einen technischen Teil untergliedert. Der wirtschaftliche Teil behandelt Grundlagen des Projektmanagements und definiert den Begriff des Projektcontrolling. Dabei wird Earned Value Management und dessen Kennzahlen als Projektmanagement-Methode zum Messen der Leistung eines Projektes vorgestellt. Der technische Teil setzt einen Fokus auf künstliche Intelligenz. Teilgebiete der KI, wie z.B. Machine Learning oder Deep Learning, werden definiert und zueinander abgegrenzt. Außerdem wird die Funktionsweise eines künstlichen neuronalen Netzes als Machine Learning Methode genauer erklärt.

Nach Vorstellung der Grundlagen, widmet sich Kapitel 4 der Analyse und Aufbereitung der in dieser Arbeit zugrunde liegenden Datenbasis, da dies einen essenziellen Schritt zur Erstellung eines KI-Modells darstellt. Hierbei stehen Daten aus mehr als 5000 Projekten zur Verfügung. Daraufhin wird in Kapitel 5 ein KI-Modell in Form eines künstlichen neuronalen Netzes implementiert, welches mit den aus Kapitel 4 aufbereiteten Daten trainiert wird. Dieses soll in der Lage sein die Endkosten eines Projektes vorherzusagen. Abbildung 1.1 veranschaulicht die zusammengefasste Vorgehensweise zur Erstellung des KI-Modells.

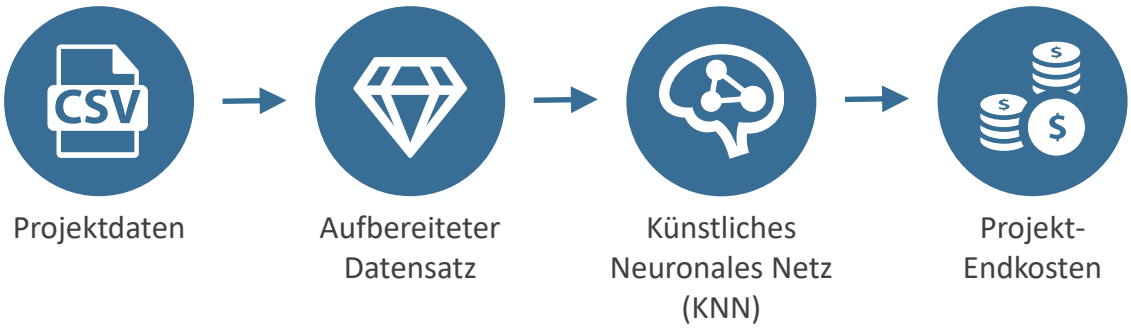

Abbildung 1.1: Vorgehen zur Prognose der Projektendkosten über ein künstliches neuronales Netz [Eigene Abbildung]

Nach Erstellung des KI-Modells, beschäftigt sich Kapitel 6 mit der Prognose der Projektendkosten nach dem Earned Value Management. Hierbei werden 5 Beispielprojekte ausgewählt, anhand derer die Vorhersage durchgeführt wird.

Abschließend findet ein Vergleich des erstellten KI-Modells zu der klassischen Earned Value Management Methode statt. Die Vorhersagen beider Modelle werden dabei nach den Kriterien der Genauigkeit und des Aufwandes miteinander verglichen.

Zielsetzung der Arbeit soll eine Veranschaulichung der Möglichkeiten von künstlicher Intelligenz im Projektcontrolling — am Beispiel der Vorhersage der Projektendkosten — sein. Dies soll durch einen Vergleich mit klassischen Projektcontrolling-Methodiken wie dem Earned Value Management erreicht werden.

2 Wirtschaftliche Grundlagen

2.1 Projektcontrolling

Um *Earned Value Management* (EVM) verstehen zu können, ist es zunächst wichtig die Begriffe Projektmanagement und Projektcontrolling vorzustellen.

2.1.1 Hinführung Projektmanagement

Laut Litke [vgl. Litke, 2007, S. 19] weisen Projekte einige Gemeinsamkeiten auf und haben dabei eine Vielzahl charakteristischer Eigenschaften. Dazu zählen unter anderem Merkmale wie eine klare Zielsetzung, definierte Start- und Endtermine, Neuartigkeit oder auch Komplexität eines Projektes. Das *Deutsche Institut für Normung* (DIN) definiert ein Projekt allgemein unter der DIN-Norm 69901 [DIN, 2009] als

> *„[...] ein Vorhaben, das im Wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, wie z.B.:*
>
> - *Zielvorgabe*
> - *zeitliche, finanzielle, personelle oder andere Bedingungen*
> - *Abgrenzungen gegenüber anderen Vorhaben und*
> - *projektspezifische Organisation.“*

Wird diese Definition nun im Kontext mit dem Begriff Management betrachtet, kann daraus bereits ein Verständnis des *Projektmanagement* (PM) abgeleitet werden, welches nach Möller und Dörrenberg [Möller und Dörrenberg, 2010, S. 4] wie folgt lautet:

> *„Gesamthafte, fachgebietsübergreifende, projektaufgabenbezogene Koordination der Planung (Projektvorbereitung) und Koordination der Durchführung (Projektsteuerung) von Projekten zur Erreichung der Projektziele Leistung, Termin, Kosten und Qualität (Produkt, Prozess, Projekt).“*

Die drei genannten Größen Termin (Zeit), Leistung und Kosten stehen dabei in einem besonderen Zusammenhang. Nicht nur dass sie den Erfolg eines Projektes maßgeblich bestimmen [vgl. Aichele, 2006, S. 25], sie sind zudem als Eckpunkte des sogenannten *Magischen Dreiecks* (Abb. 2.1) des Projektmanagement bekannt [vgl. Möller und Dörrenberg, 2010, S. 4].

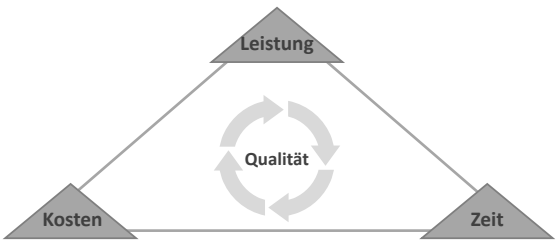

Abbildung 2.1: Das Magische Dreieck im Projektmanagement [Möller und Dörrenberg, 2010, S. 22]

Die drei Faktoren des Magischen Dreiecks stehen im gesamten Projektverlauf in einer wechselseitigen und konkurrierenden Beziehung zueinander. Die Beeinflussung einer Größe wirkt sich unmittelbar auf die übrigen Parameter aus. Um das Projektende bei Terminverzug einhalten zu können, muss entweder die Kapazität und somit die Kosten erhöht oder die Leistung reduziert werden [vgl. Möller und Dörrenberg, 2010, S. 22]. Ein Ziel des Projektmanagements liegt darin, die drei Faktoren des Magischen Dreiecks so zu beeinflussen, dass in kürzester Zeit und mit wenig Kosten ein Maximum an Leistung und Qualität erzeugt wird [vgl. Keßler und Winkelhofer, 2004, S. 55]. Im Prozess des Projektcontrollings spielt der Zusammenhang der drei Größen eine wichtige Rolle, da diese sowohl bei der Projektplanung als auch Projektsteuerung zu beachten sind [vgl. Möller und Dörrenberg, 2010, S. 4].

2.1.2 Definition Projektcontrolling

In dem vorherigen Abschnitt wurde erklärt wodurch Projekte charakterisiert werden und was unter Projektmanagement zu verstehen ist. In diesem Abschnitt werden der Begriff des Projektcontrollings und deren Werkzeuge genauer untersucht.

Das klassische Controlling ist ein Teilbereich des unternehmerischen Führungssystems. Die Hauptaufgabe ist dabei die Planung, Steuerung und Kontrolle aller Unternehmensbereiche [Weber, 2019]. Der Controller schafft über Analyse-, Planungs-, Steuerungs-, und Kontrollaktivitäten Transparenz und stellt der Geschäftsleitung wichtige Informationen zur Entscheidungsunterstützung und Verfolgung der geplanten Ziele zur Verfügung [vgl. Zirkler u. a., 2019, S. 24]. Setzt man nun das operative Controlling in Bezug zu Projekten, besteht das Projektcontrolling nach Wanner [vgl. Wanner, 2013b, S. 23] aus drei Prozessschritten: Der **Projektplanung**, **Projektüberwachung** und **Projektsteuerung**. Laut DIN 69901 [DIN, 2009] lässt sich Projektcontrolling dabei wie folgt definieren:

> *„Sicherung des Erreichens der Projektziele durch:*
>
> - *Soll-Ist-Vergleich*
> - *Feststellung der Abweichungen*
> - *Bewerten der Konsequenzen und Vorschlagen von Korrekturmaßnahmen*
> - *Mitwirkung bei der Maßnahmenplanung und Kontrolle der Durchführung.“*

Die aus der obigen Definition genannten Aktivitäten können dabei nach Wanner [vgl. Wanner, 2013b, S. 23] in den bereits vorgestellten Projektcontrolling Prozess eingeordnet werden (siehe Tabelle 2.1).

Projektüberwachung	**Projektsteuerung**
Soll-Ist-Vergleich	Maßnahmen definieren und planen
Feststellen von Abweichungen	Entscheidungen treffen
Abweichungen analysieren	Kontrolle der Durchführung

Tabelle 2.1: Einordnung Projektaktivitäten in Projektcontrolling Prozess

Die im Rahmen dieser Arbeit entwickelte Künstlichen Intelligenz unterstützt dabei in der Phase der Projektüberwachung und dient dazu die Steuerungsmöglichkeiten von Projekten zu fördern. Typische Instrumente zur Kosten-, Budget- und Leistungskontrolle sind z.B. der klassische Soll-Ist-Vergleich oder das Earned Value Management.

Im Rahmen dieser Arbeit wird die Earned Value Management Methode zur Vorhersage der Projektendkosten als Vergleich zu einer KI-Methode herangezogen, da diese in Kontrast zu einem einfachen Soll-Ist-Vergleich den Fertigstellungswert (Earned Value) berücksichtigt und damit eine genauere Analyse ermöglicht. Im folgenden Kapitel werden deshalb das Earned Value Management und dessen Kennzahlen genauer erklärt.

2.2 Earned Value Management

2.2.1 Definition

Earned Value Management (EVM) ist eine strukturierte Projektmanagement-Methode, um die Leistung eines Projektes zu messen. Es stellt Kennzahlen zur Verfügung um Analysen und Vorhersagen über die Kosten und Termine eines Projektes im Projektcontrolling zu ermöglichen und so auf Abweichungen während der Projektdauer frühzeitig reagieren zu können [vgl. Wanner, 2013a, S. 53].

EVM überwacht im Vergleich zum traditionellen Projektmanagement neben Budget (PLAN) und Ausgaben (IST) zusätzlich den Projektumfang bzw. die erledigte Arbeit und liefert damit eine höhere Transparenz des Projektstatus. Grundlage für eine genaue Bestimmung der Projektleistung liegt in der Projektplanung. Nur wenn detaillierte Planungsdaten vorhanden sind kann das EVM einen Mehrwert liefern. Zum Erfassen des Projektumfangs bietet sich vor allem die Anwendung eines detaillierten Projektstrukturplans (PSP) mit definierten Arbeitspaketen an, zu denen Projektmitarbeiter/innen laufend Leistung und Zeit erfassen. Diese Projektdaten sind essenziell zur Bestimmung des Projektfortschritts und der EVM-Kennzahlen [vgl. Wanner, 2013a, S. 77].

2.2.2 Kennzahlen

In diesem Kapitel werden die relevanten EVM-Kennzahlen nach Wanner [Wanner, 2013a] erklärt. Zunächst werden die Earned Value Management Basis-Kennzahlen und anschließend die darauf aufbauenden Leistungskennzahlen vorgestellt.

Basis-Kennzahlen

Das Earned Value Management besteht aus drei Basis-Kennzahlen — Planned Value (Planwert), Earned Value (Fertigstellungswert), Actual Cost (IST-Kosten) — welche die Grundlage zur Berechnung der EVM-Leistungskennzahlen bilden [vgl. Wanner, 2013a, S. 128 ff.]:

- **Planned Value** (PV) sind die aus Projektplanung ermittelten budgetierten Kosten der geplanten Leistung, zu jedem Zeitpunkt des Projektes. In dieser Arbeit bezieht sich der Planned Value immer auf die geplanten Gesamtkosten bei Fertigstellung und ist damit synonym zu dem Begriff Budget at Completion (BAC).
- **Earned Value** (EV) steht für den Wert der erreichten (*earned*) Leistung zu einem bestimmten Zeitpunkt, basierend auf dem geplanten Wert (Budget) für diese Arbeit. Um den Earned Value zu bestimmen gibt es unterschiedliche Verfahren, die im Rahmen dieser Arbeit jedoch nicht weiter behandelt werden.
- **Actual Cost** (AC) sind die im Projekt angefallenen (IST-)Kosten für die bis zu einem Stichtag geleistete Arbeit.

In der Literatur werden häufig die Synonyme *Budgeted Cost of Work Scheduled* (BCWS) für Planned Value, *Budgeted Cost of Work Performed* (BCWP) für Earned Value und *Actual Cost of Work Performed* (ACWP) für Actual Cost verwendet. Dies liegt daran, dass aufgrund der Norm *ANSI/EIA-784* im Jahre 1998 die Bezeichnungen der EVM-Basiskennzahlen auf PV, EV und AC geändert wurden [vgl. Wanner, 2013a, S. 128]. Im weiteren Verlauf der Arbeit werden PV, EV und AC als Bezeichnungen der Kennzahlen verwendet.

Leistungskennzahlen

Die Leistungskennzahlen unterstützen dabei frühzeitig Abweichungen der Kosten und Termine zu erkennen. Dadurch können rechtzeitig Lösungen gefunden und Maßnahmen getroffen werden, um den Projektstatus zu verbessern. Eine Übersicht der für diese Arbeit relevanten Leistungskennzahlen ist in Tabelle 2.2 aufgeführt [vgl. Wanner, 2013a, S. 134 ff.; Reichel, 2006].

Abk.	Kennzahl	Beschreibung
CV	**Cost Variance** $CV = EV - AC$	Abweichung der IST-Kosten zum Stichtag. < 0: Budget für erbrachte Leistung wurde überschritten.
SV	**Schedule Variance** $SV = EV - PV$	Abweichung der PLAN-Kosten zum Stichtag. < 0: Rückstand gegenüber den geplanten Werten.
SPI	**Schedule Performance Index** $SPI = \frac{EV}{PV}$	Misst die Zeiteffizienz eines Projektes. ≥ 1: Projektergebnisse wurden schneller als geplant erreicht. < 1: Projekt macht Fortschritte langsamer als geplant. Konvergiert gegen Projektende zum Wert 1, da dann EV gleich PV.
CPI	**Cost Performance Index** $CPI = \frac{EV}{AC}$	Misst die Kosteneffizienz eines Projektes. ≥ 1: Ressourcen wurden effizient eingesetzt. < 1: Budget wurde überzogen.
BAC	**Budget at Completion**	Geplante Gesamtkosten bei Fertigstellung.
EAC	**Estimate at Completion** $EAC = AC + \frac{BAC-EV}{CPI}$	Geschätzte Gesamtkosten bei Fertigstellung.
VAC	**Variance at Completion** $VAC = BAC - EAC$	Kostenabweichung bei Fertigstellung.

Tabelle 2.2: Übersicht der EVM-Leistungskennzahlen

Der **Cost Performance Index (CPI)** spielt im Earned Value Management eine besondere Rolle, da dieser die Kosteneffizienz eines Projektes darstellt und benötigt wird, um die Projekt-Endkosten (Estimate at Completion) zu prognostizieren. Der CPI steht in Kontrast zum EV, was bedeutet, dass bei einem CPI von 0.9 jeder ausgegebene Euro nur 90 Cent an Wert (Earned Value) realisiert hat. Im Vergleich dazu sind bei einem CPI von zwei die ausgegebenen Ressourcen das Doppelte an Leistung wert [vgl. Wanner, 2013a, S. 137].

Prognosen erweisen sich als wirkungsvolles Managementinstrument, da durch Schätzungen der zukünftige Projektverlauf in Bezug auf Kosten und Termine vorhergesagt werden kann. Um die Projektendkosten mithilfe des EVM vorhersagen zu können, werden zur Berechnung des **Estimate at Completion** mehrere Methoden zur Verfügung gestellt [vgl. Wanner, 2013a, S. 146]:

- **Optimistische** Methode (*Mathematical or Overrun to Date EAC*)

$$EAC = AC + \frac{BAC - EV}{1.0} \tag{2.1}$$

- **Realistische** Methode (*Low-End Cumulative CPI EAC*)

$$EAC = AC + \frac{BAC - EV}{CPI} \tag{2.2}$$

- **Pessimistische** Methode (*High-End Cumulative CPI x SPI EAC*)

$$EAC = AC + \frac{BAC - EV}{CPI \cdot SPI} \tag{2.3}$$

Grundlage zur Berechnung des Estimate at Completion bilden die AC, der EV, das BAC und je nach Berechnungsvariante der CPI oder SPI. Das BAC abzüglich des EV bilden dabei den Wert der verbleibenden Arbeit (*Work Remaining WR*) [vgl. Wanner, 2013a, S. 147 ff.]. Im Rahmen dieser Arbeit wird die *Realistische Methode* (2.2) zur Berechnung des EAC verwendet, da diese genaue Ergebnisse bezüglich der Vorhersage erzielt und die vorliegende Datenbasis alle hierfür benötigten Kennzahlen enthält. Prognosen bezüglich Terminen und zeitlichen Aspekten werden nicht behandelt.

Der Zusammenhang und das Verhalten zwischen den einzelnen EVM-Kennzahlen werden in Abb. 2.2 nochmals übersichtlich veranschaulicht.

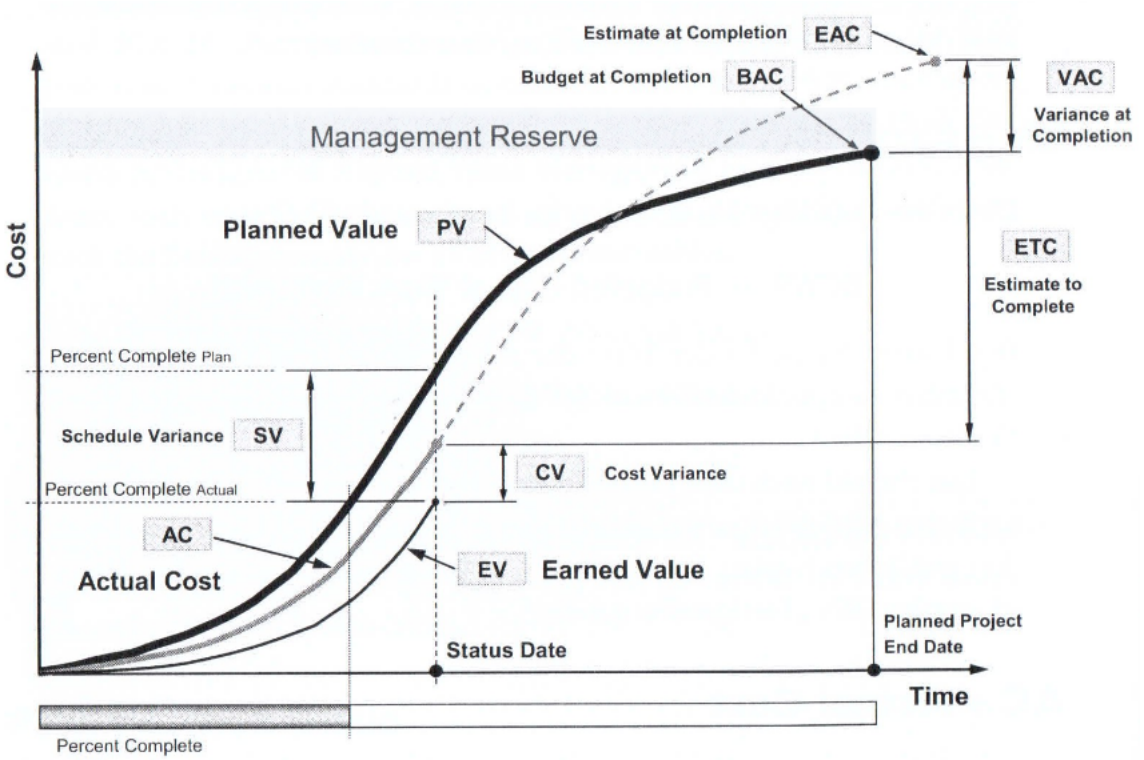

Abbildung 2.2: Zusammenhang der EVM-Kennzahlen [Wanner, 2013a, S. 130]

Die vorgestellten Kennzahlen können sowohl kumulativ oder als aktuelle Werte (monatsgenau) berechnet werden. In dieser Arbeit wird standardmäßig mit den kumulativen Werten gearbeitet, falls nichts weiter spezifiziert ist.

3 Technische Grundlagen

3.1 Künstliche Intelligenz

Der Begriff *Künstliche Intelligenz* bzw. KI (engl. artificial intelligence oder AI) fand seinen Ursprung im Jahre 1956, als John McCarthy einen zweimonatigen Workshop zur Untersuchung der KI am Dartmouth College initiierte. Beim Aufruf an eine ausgewählte Gruppe von Wissenschaftlern verwendete McCarthy dabei erstmals KI als Überbegriff für "Maschinen [...] die sich verhalten, als verfügten sie über Intelligenz" [Ertel, 2016, S. 1; vgl. Russell und Norvig, 2010, S. 17].

Da sich Intelligenz allgemein nur schwer definieren lässt, existiert bis heute noch keine einheitliche und exakte Definition einer KI [vgl. Kurbel, 1992, S. 1]. Lämmel und Cleve setzen beispielsweise die Intelligenz in Zusammenhang mit menschlichen Vorgehensweisen und definieren KI als:

> *„Teilgebiet der Informatik, welches versucht, menschliche Vorgehensweisen der Problemlösung auf Computern nachzubilden, um auf diesem Wege neue oder effizientere Aufgabenlösungen zu erreichen [Lämmel und Cleve, 2012, S. 13]."*

Die oben genannte Definition vereint dabei zwei unterschiedliche Herangehensweisen. Zum einen wird der Mensch als Vergleich herangezogen. Demnach stellt die KI eine Abbildung menschlicher Fähigkeiten, wie das Gedächtnis oder sein Lernverhalten, auf Maschinen dar. Der zweite Aspekt zielt auf die Assistenzfunktion der künstlichen Intelligenz ab. Die KI soll dabei dem Menschen bei Aufgabenbewältigung und Lösung komplexer Problemstellungen unterstützen [vgl. Lämmel und Cleve, 2012, S. 13].

Der britische Mathematiker Alan Turing lieferte 1950 mit dem *Turing-Test* [Turing, 1950] eine operative Definition von Intelligenz. Ein Computer absolviert den Intelligenztest dabei erfolgreich, wenn ein menschlicher Fragesteller, der einige schriftliche Fragen stellte, nicht unterscheiden kann ob die schriftlichen Antworten von einem Menschen oder einer Maschine stammen. Um den Test dabei erfolgreich bestehen zu können, muss ein Computer folgende Eigenschaften annehmen, welche gleichzeitig die Hauptthemengebiete der Künstlichen Intelligenz darstellen [vgl. Russell und Norvig, 2010, S. 2]:

- **Natural Language Processing** (NLP), zur Kommunikation, um die gestellten Fragen verarbeiten und Antworten liefern zu können.
- **Wissensrepräsentation**, um die Fragen inhaltlich korrekt interpretieren und speichern zu können.
- **Automatisches logisches Schließen**, um anhand der gespeicherten Informationen die richtigen Schlüsse zu ziehen damit die Fragen beantwortet werden können.
- **Machine Learning** (ML), um sich an neue Umstände anzupassen und allgemeine Muster erkennen zu können.

Berücksichtigt man dabei die physische Simulation einer Person, ergänzen sich die Themengebiete um **Bilderkennung** für die Objektwahrnehmung und der **Robotik** zur Manipulation und Bewegung von Objekten [vgl. Russell und Norvig, 2010, S. 3]. Da in dieser Arbeit Verfahren des Machine Learning zum Einsatz kommen wird nachfolgend näher auf dieses Teilgebiet eingegangen.

3.2 Machine Learning

Eines der wichtigsten Teilgebiete der KI stellt Machine Learning (kurz ML oder zu deutsch maschinelles Lernen genannt) dar. Machine Learning setzt Methoden des selbstständigen Lernens ein, um automatisiert Muster und Zusammenhänge in Daten zu erkennen. Basierend auf den erlernten Zusammenhängen werden diese Muster dann zur Entscheidungsfindung oder Vorhersage zukünftiger Daten eingesetzt [vgl. Murphy, 2012, S. 1].

Die wohl einfachste Form des Lernens ist das sogenannte *Supervised Learning* (Überwachtes Lernen). Beim überwachten Lernen wird ein Algorithmus mit Daten — bestehend aus Eingabewerten x und erwarteten Ausgabewerten y — trainiert, um dabei Relationen zu erkennen, damit y anhand von x vorhergesagt werden kann [vgl. Goodfellow u. a., 2018, S. 115]. Hierbei wird der Datensatz $D = \{(x_i, y_i)\}_{i=1}^{N}$ als **Trainingsdatensatz** bezeichnet, wobei N für die Anzahl der einzelnen Zeilen des Datensatzes steht [vgl. Murphy, 2012, S. 2]. Im einfachsten Fall ist jeder Eingabewert x_i ein n-dimensionaler Vektor, bestehend aus numerischen Werten, die beispielsweise im Falle eines Projektes die Mitarbeiteranzahl darstellen kann. Die Eingabewerte werden auch als **Merkmale** oder englisch Features bezeichnet [vgl. Murphy, 2012, S. 2].

Der Ausgabewert y_i — auch **Zielwert** oder Label genannt — kann ähnlich wie der Eingabewert relativ viele Formen je nach der zu lösenden Aufgabenstellung annehmen. Die bekanntesten Aufgaben stellen dabei die Klassifizierung und Regression dar. Bei der **Klassifizierung** ist y_i eine kategorische Variable aus einem Set von k Kategorien, $y_i \in \{1, ..., k\}$. Beispielsweise könnten die Geschlechter Männlich und Weiblich Kategorien darstellen. Ist y_i eine reelle Zahl — wie z.B. die Projektendkosten — so wird die Machine Learning Aufgabe als **Regression** bezeichnet [vgl. Murphy, 2012, S. 2; Goodfellow u. a., 2018, S. 109 f.]. Das Lernen wird hierbei als *überwacht* angesehen, da die Vorhersage des Modells $\hat{y}$, mit dem tatsächlichen y-Wert verglichen werden kann und dadurch der Lernprozess durch das Wissen des Zielwertes geführt wird [vgl. Hastie u. a., 2017, S. 2].

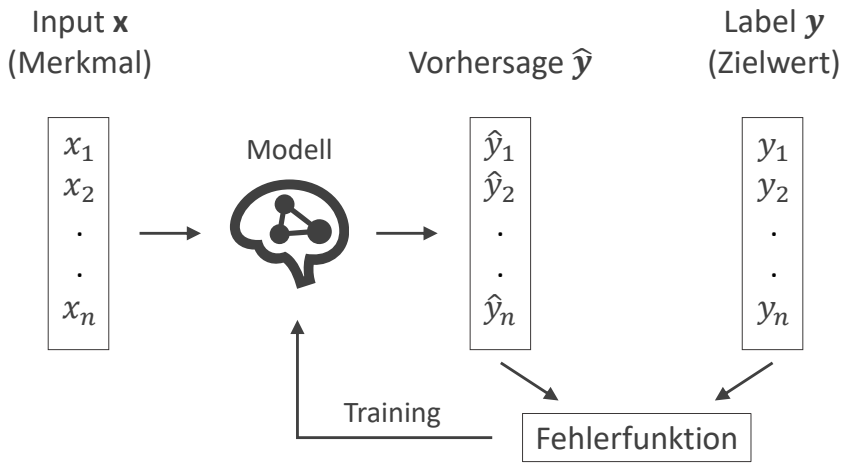

Abbildung 3.1: Trainingsphase des überwachten Lernen [Eigene Abbildung]

Abstrahiert kann also gesagt werden, dass bei dem Trainingsdatensatz D jeder y_i Wert durch eine unbekannte Funktion $y = f(x)$ generiert wird, während das Ziel des überwachten Lernens darin besteht eine Hypothesenfunktion h zu finden, welche die wahre Funktion f weitestgehend approximiert. Dies geschieht dabei durch Minimierung einer Fehlerfunktion, welche die

Abweichung zwischen y und der Vorhersage $\hat{y}$ darstellt (siehe Abb. 3.1). Die Genauigkeit einer solchen Hypothese wird mithilfe eines **Testdatensatzes** gemessen, dessen Elemente sich von dem Trainingsdatensatz unterscheiden. Die Funktion h generalisiert dabei gut, wenn sie gegenüber unbekannten Eingabedaten genaue Ergebnisse erzielt [vgl. Russell und Norvig, 2010, S. 695].

Eine weitere Form des Lernens ist das *Unsupervised Learning* (Unüberwachte Lernen). Beim unüberwachten Lernen ist im Vergleich zum überwachten Lernen, kein y-Wert gegeben. Der Trainingsdatensatz besteht dabei aus $D = \{x_i\}_{i=1}^{N}$, und hat als Ziel anhand der Merkmale interessante Muster zu erkennen, um die Daten zu klassifizieren [vgl. Murphy, 2012, S. 2]. Die Grenzen zwischen überwachten und unüberwachten Lernen sind oft fließend. Viele ML-Technologien können daher beide erfüllen [vgl. Goodfellow u. a., 2018, S. 116].

Das sogenannte *Reinforcement Learning* stellt die am wenigsten vorkommende Lernform dar. Hierbei lernt der Algorithmus über Rückmeldungen bezüglich seiner Handlungen und kann sich so optimal an seine Umgebung anpassen [vgl. Russell und Norvig, 2010, S. 830].

Der weitere Verlauf der Arbeit beschäftigt sich mit dem überwachten Lernen, da dieses im Rahmen der Vorhersage der Projektendkosten die verwendete Lernform darstellt.

3.3 Deep Learning

Das Deep Learning ist ein Unterbereich des Machine Learning (siehe Abbildung 3.2) und stellt ein leistungsstarkes Framework für überwachtes Lernen zur Verfügung [vgl. Goodfellow u. a., 2018, S. 184; Goodfellow u. a., 2018, S. 9].

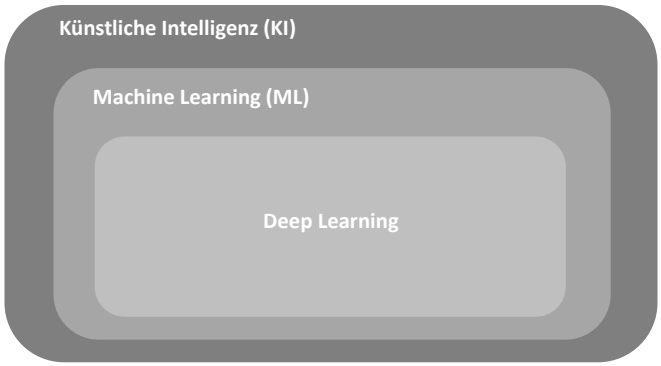

Abbildung 3.2: Venn-Diagramm Deep Learning [Eigene Abbildung]

Computern soll es dabei ermöglicht werden, aus Erfahrungen Wissen zu gewinnen und die Welt als eine verschachtelte Hierarchie von Konzepten zu verstehen. Jedes Konzept ist hierbei durch eine Beziehung zu einfacheren Konzepten definiert. Der Computer ist dadurch in der Lage komplexe Zusammenhänge zu erlernen, indem er mehrere einfachere Konzepte zu komplexeren zusammensetzt. Da ein solches Modell sich grafisch als Schichten darstellen lässt, die sich in die Tiefe erstrecken, wird diese Methode der künstlichen Intelligenz als Deep Learning (tiefergehendes Lernen) bezeichnet [vgl. Goodfellow u. a., 2018, S. 2]. Zur Anwendung von Deep Learning werden künstliche neuronale Netze verwendet, welche nachfolgend genauer erklärt werden.

3.4 Künstliche neuronale Netze

Ein künstliches neuronales Netz (KNN) ist ein Modell des Machine Learning, das Strukturen des menschlichen Gehirns abbildet [vgl. Haykin, 1999, S. 24]. Ein neuronales Netz implementiert den Lernprozess dabei über einfache Recheneinheiten, die als Neuronen bezeichnet werden. Typischerweise sind die Neuronen in Form von Schichten (*layer*) organisiert. Neuronen einzelner Schichten sind über gerichtete und gewichtete Kanten miteinander verbunden [vgl. Rojas, 1996, S. 121 f.; Deru und Ndiaye, 2019, S. 61 f.]. Abbildung 3.3 zeigt den Aufbau eines KNN.

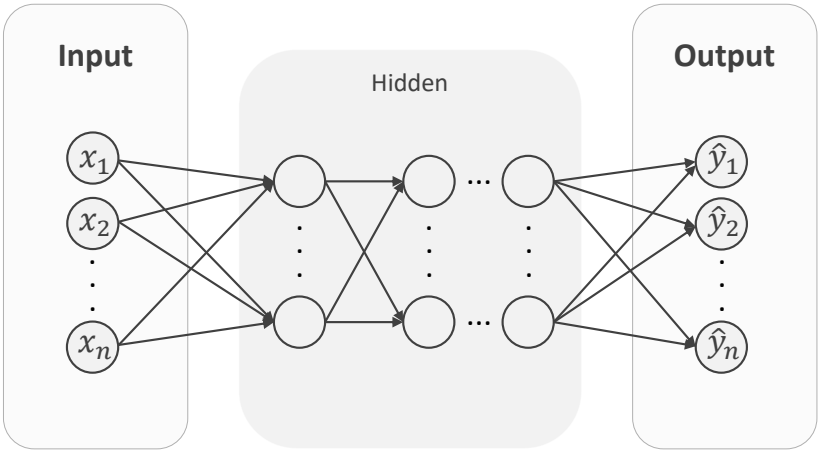

Abbildung 3.3: Aufbau eines künstlichen neuronalen Netzes [Eigene Abbildung]

Der sogenannte *Input-Layer* (Eingabeschicht) repräsentiert die numerischen Merkmalsvektoren für das Netzwerk und stellt somit im Machine Learning Kontext die x-Werte dar. Der *Output-Layer* (Ausgabeschicht) steht für die durch das Netzwerk berechnete Lösung und bildet die $\hat{y}$-Werte. Über die Verbindungen zwischen den Neuronen werden die x-Werte aus der Eingabeschicht entlang ein oder mehrerer *Hidden-Layer* (versteckte Schicht) bis zum Output-Layer propagiert und sind somit für die Weiterverarbeitung zuständig [vgl. Deru und Ndiaye, 2019, S. 61 f.]. Existieren hierbei mehrere Hidden-Layer, wird das neuronale Netzwerk auch als mehrschichtiges tiefes Netzwerk bezeichnet. Tiefe neuronale Netze zählen zu dem oben genannten Teilgebiet des Deep Learning [vgl. Goodfellow u. a., 2018, S. 185]. Die Anzahl der Hidden-Layer trägt dazu bei, aufgrund der zusätzlichen synaptischen Verbindungen und neuronalen Dimensionen, Informationen höherer Ordnung zu extrahieren [vgl. Haykin, 1999, S. 43 f.]. Da Neuronen einen essenziellen Teil des neuronalen Netzes darstellen, wird im folgenden Kapitel die Funktion eines einzelnen künstlichen Neurons genauer erklärt.

3.4.1 Das künstliche Neuron

Ein Neuron ist eine Recheneinheit, welche Informationen verarbeiten kann. Es ist die Grundlage zur Erstellung eines neuronalen Netzes, da es innerhalb des Netzes die Berechnungen ausführt. Die Darstellung in Abbildung 3.4 zeigt den Aufbau eines einzelnen Neurons nach Haykin [vgl. Haykin, 1999, S. 32 f.]. Das Modell eines Neuron besteht demnach aus drei Elementen:

1. **Gewichtete Eingangssignale**. Das Signal x_j an der Verbindung j des Neurons k wird mit dem Gewicht w_{kj} multipliziert. Das Gewicht beschreibt dabei die Intensität der Eingabe.

2. Eine **Summenfunktion**, welche die gewichteten Eingangssignale aufsummiert.

3. Die **Aktivierungsfunktion** φ, bestimmt den Aktivierungszustand bzw. die Stärke der Aktivierung eines Neuron. Hierbei gibt es mehrere Funktionen, wie z.B. die Schwellwertfunktion, Identitätsfunktion oder ReLU (*Rectified Linear Unit*). Die Schwellwertfunktion ist definiert als [vgl. Ertel, 2016, S. 269]:

$$\varphi(v) = \left\{ \begin{array}{ll} 0 & \text{falls } v < \theta \\ 1 & \text{sonst} \end{array} \right\}$$

wobei θ den Schwellenwert darstellt. Ein Neuron kann nach der Schwellwertfunktion den Zustand aktiv oder inaktiv einnehmen und gilt als aktiviert, wenn $v \geq \theta$.

Die wohl einfachste Form der Aktivierungsfunktion ist die Identitätsfunktion, welche definiert wird durch [vgl. Ertel, 2016, S. 269]:

$$\varphi(v) = v$$

Die Identitätsfunktion ist linear und aktiviert ein Neuron umso stärker, je größer v ist. Damit ist sie im Vergleich zu der binären Schwellfunktion — die ein Neuron entweder als aktiv oder inaktiv betrachtet — differenzierbar und gibt den Aktivierungsgrad an.

Die für moderne neuronale Netze empfohlene Aktivierungsfunktion stellt die ReLU-Funktion dar, die folgendermaßen definiert ist:

$$\varphi(v) = max\{0, v\}$$

Die ReLU ist nahezu linear. Sie besteht aus zwei linearen Abschnitten und stellt somit eine stückweise lineare Funktion dar. Dadurch erhält die ReLU viele Eigenschaften die zur Generalisierung von linearen Modellen beitragen [vgl. Goodfellow u. a., 2018, S. 192].

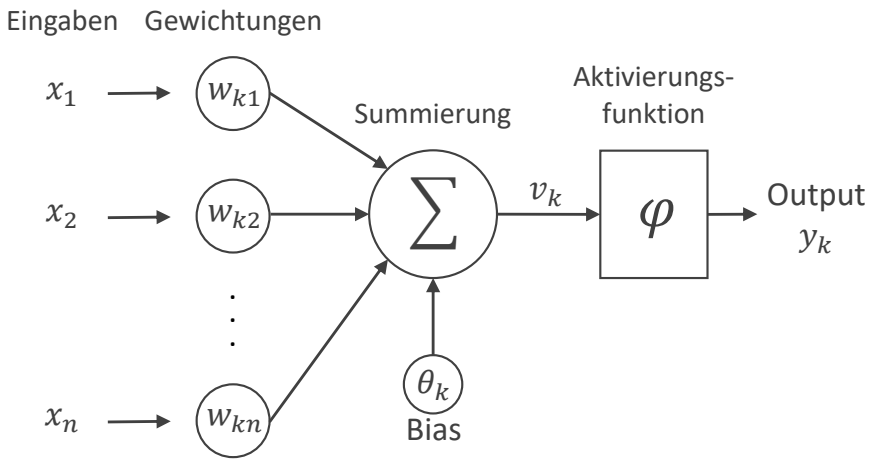

Abbildung 3.4: Ein künstliches Neuron [In Anlehnung an Haykin 1999, S. 33]

Die gewichtete Summe des Neurons k beinhaltet außerdem einen sogenannten Bias-Wert θ_k. Diese Konstante wird meistens mit -1 initialisiert und fungiert als eine Art Schwelle, die eine

feste Verschiebung der durch das Neuron geschaffenen Hyperebene erzeugt. Neben den Gewichten des Neurons wird der Bias-Wert im Lernprozess auch gelernt [vgl. Ertel, 2016, S. 204]. Mathematisch lässt sich ein Neuron also nach Haykin definieren als [Haykin, 1999, S. 33 f.]:

$$v_k = \sum_{j=1}^{n} w_{kj} \cdot x_j + \theta_k \tag{3.1}$$

und

$$y_k = \varphi(v_k) \tag{3.2}$$

wobei $x_1, x_2, ..., x_n$ die Eingabesignale sind und $w_{k1}, w_{k2}, ..., w_{kn}$ die Gewichte des Neuron k, für die jeweiligen Eingabesignale, darstellen. v_k ist die Linearkombination der Signale und Gewichte mit dem Grundwert θ_k. Die Aktivierungsfunktion φ nimmt als Eingabewert v_k entgegen und berechnet damit das Ausgabesignal y_k des Neurons k [vgl. Haykin, 1999, S. 32 ff.].

Werden mehrere Schichten von Neuronen hintereinander geschaltet, bilden diese ein neuronales Netz. Hierbei erhalten die Neuronen als Eingabesignale die Ausgabesignale der Neuronen aus der vorangegangenen Schicht. Das Ausgabesignal lässt sich damit über die Kantengewichte steuern. Werden diese Gewichte dahingegen optimiert, dass die durch das KNN berechnete Ausgabe $\hat{y}$ den tatsächlichen Zielwert y approximiert, findet ein Lernprozess statt. Dieser Lernprozess wird auch als *Backpropagation* Algorithmus bezeichnet.

3.4.2 Lernprozess

Die Aufgabe des Backpropagation Algorithmus liegt darin, durch Abstieg in der Gradientenrichtung das Minimum der Fehlerfunktion eines bestimmten Lernproblems zu suchen [vgl. Rojas, 1996, S. 149]. Im Rahmen dieser Arbeit wird als Fehlerfunktion der **mittlere quadratische Fehler** (MSE oder auch mean squared error genannt) verwendet (3.3). Diese Fehlerfunktion ist vor allem für Aufgaben der Regression gut geeignet [vgl. Goodfellow u. a., 2018, S. 148].

$$MSE = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}_i - y_i||^2 \tag{3.3}$$

Der MSE misst durch das Quadrat der Differenz zwischen dem geschätzten und tatsächlichen Zielwert die Annäherung der Vorhersage an den tatsächlichen Wert. Der Erwartungswert wird hierbei über m Trainingsdaten gebildet und sorgt damit für einen geringeren Generalisierungsfehler.

Das Lernproblem wird dabei gelöst, indem die Kombination derjenigen Gewichte eines Netzes, die den Berechnungsfehler bestmöglich minimieren, berechnet werden [vgl. Rojas, 1996, S. 149]. Die genauen mathematischen Vorgehensweisen des Backpropagation Algorithmus und des Gradientenabstiegverfahrens werden in dieser Arbeit nicht weiter behandelt.

Das Ziel des Lernprozesses ist es ein Modell zu schaffen, das gut generalisieren bzw. verallgemeinern kann. Mit generalisieren ist dabei gemeint, dass die berechneten Gewichte des Modells, basierend auf dessen Eingabewerte, korrekte Output-Werte für unbekannte Testdaten liefert, die das Modell während des Trainings nicht eingesehen hat. Passt sich ein Modell dabei zu stark an die Trainingsdaten an und lernt Muster die nur in den Trainingsdaten vorhanden sind, wird von einer Überanpassung des Modells gesprochen. Eine Unteranpassung ist demnach, wenn ein Modell mit einer zu geringen Datenmenge trainiert wird und dadurch kein allgemeines Input-Output-Mapping erstellen kann [vgl. Haykin, 1999, S. 227 f.].

4 Analyse und Aufbereitung der Datenbasis

In der Realität sind Rohdaten meist fehlerbehaftet und für Mensch und Maschine nur schwer verständlich. Damit ein Deep Learning Modell realistische Ergebnisse erzielen kann, ist es nötig, die Daten vorher zu analysieren und aufzubereiten. Dieser Prozess, der mit der Analyse und Aufbereitung der Datenbasis einhergeht, stellt einen essenziellen Schritt zur Erstellung eines künstlichen neuronalen Netzes dar und wird nachfolgend genauer erklärt.

4.1 Knowledge Discovery in Databases Prozess

Knowledge Discovery in Databases (KDD) lässt sich nach Fayyad definieren als *„[...] nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [Fayyad u. a., 1996, S. 40 f.].“* Mit Patterns sind dabei Muster gemeint, die eine Teilmenge des Datensatzes beschreiben. Diese Muster sollen gültig (*valid*), neuartig (*novel*), nützlich (*useful*) und verständlich (*understandable*) sein. Gültigkeit beschreibt hierbei die Eigenschaft, dass die Muster auch für unbekannte Daten angewendet werden können und somit das Modell gut generalisieren kann (siehe Kapitel 3.4.2) [vgl. Fayyad u. a., 1996, S. 41].

Die Mustererkennung in Daten wird häufig auch als Data Mining (DM) bezeichnet [vgl. Kotu und Deshpande, 2015, S.17]. Das einzelne Anwenden von DM-Methoden ist nach Fayyad jedoch sehr gefährlich, da dadurch schnell falsche und bedeutungslose Muster in den Daten erkannt werden. Fayyad definiert KDD deshalb nach obiger Definition als einen weiterführenden Prozess zum Data Mining, der neben dem Schritt zur Erkennung nützlicher Muster in Daten, zusätzliche Schritte zur Selektion, Aufbereitung, Transformation und Interpretation der Daten berücksichtigt. Diese weiterführenden Schritte dienen dazu, eine gute Datenqualität sicherzustellen, um damit vertrauenswürdige Informationen extrahieren zu können [vgl. Fayyad u. a., 1996, S. 39 ff.]. Obwohl DM nach Fayyad einen Teilschritt in dem KDD-Prozess darstellt, werden heutzutage aufgrund der engen Verwandtschaft die Begriffe Data Mining und Knowledge Discovery in Databases weitestgehend synonym verwendet [vgl. Piatetsky-Shapiro, 2007].

Abbildung 4.1: Phasen des CRISP-DM Data Mining Prozesses [Wikipedia, 2019]

Ein sehr beliebtes Modell des KDD-Prozesses ist das *Cross Industry Standard Process for Data Mining* (CRISP-DM) Modell, welches durch ein Konsortium aus mehreren Unternehmen entwickelt wurde und als branchenübergreifendes Standardmodell für das Data Mining gilt [vgl. Chapman u. a., 2000]. Wie aus Abbildung 4.1 zu ersehen ist, besteht ein Data Mining Projekt nach dem CRISP-DM-Modell grundsätzlich aus sechs generischen Phasen. Die Pfeile zeigen, dass Rückbeziehungen zwischen den Phasen möglich sind und der Ablauf damit nicht fest vorgegeben ist. Der äußere Kreis symbolisiert die zirkuläre Struktur von Data Mining Projekten und verdeutlicht, dass Data Mining ein iterativer Prozess ist, bei dem durch Erfahrungswerte aus vorangegangenen Projekten profitiert wird [vgl. Chapman u. a., 2000, S. 10 ff.]

Im Folgenden werden die sechs Phasen des CRISP-DM-Modells nach Chapman [Chapman u. a., 2000, S. 10 f.] genauer erklärt:

1. Geschäftsverständnis (Business Understanding)
Der Fokus in der initialen Phase liegt darin, die Projektziele und Anforderungen sowohl aus der Business-, als auch Data Mining Perspektive zu verstehen, damit daraus ein Projektplan erstellt werden kann. Dabei wird untersucht, welches Problem gelöst werden soll und was für Kosten und Risiken dabei entstehen können.

2. Datenverständnis (Data Understanding)
In diesem Schritt wird überprüft inwieweit die vorhandene Datenbasis zur Lösung des Problems geeignet ist. Die zur Verfügung stehenden Daten werden beschrieben und Zusammenhänge untersucht. Außerdem wird ein erster Blick auf die Datenqualität geworfen.

3. Datenvorbereitung (Data Preparation)
Die Datenvorbereitung stellt den aufwändigsten Schritt innerhalb des Prozesses dar. Ziel ist es dabei, die Daten so aufzubereiten, dass diese für Machine Learning Algorithmen verständlich sind und verarbeitet werden können. Dies geschieht durch Transformation, Auswahl, Reinigung und Normalisierung der Daten.

4. Modellierung (Modeling)
In der Modellierung werden Methoden des Data Mining angewendet um mit den transformierten und aufbereiteten Daten ein Modell zu trainieren. Mehrere Modellparameter werden dabei kalibriert um optimale Ergebnisse zu erzielen. Diese Phase stellt somit das eigentliche Data Mining dar.

5. Evaluierung (Evaluation)
Nach dem Training des in Schritt vier entwickelten Modells, werden in der Evaluierung die Resultate des trainierten Modells bewertet und entschieden, ob die Data Mining Resultate einen Mehrwert für das Business liefern und verwendet werden sollen.

6. Bereitstellung (Deployment)
Die letzte Phase des CRISP-DM beschäftigt sich mit der Bereitstellung des erarbeiteten Data Mining Modells, damit es für den Kunden nutzbar ist. Dies kann sowohl in Form eines einfachen Reports, als auch durch Implementierung des Data Mining Prozesses in die Unternehmenslandschaft gestaltet werden.

Die Vorgehensweise zur im Rahmen dieser Arbeit entwickelten KI für die Prognose der Projektendkosten im Projektcontrolling orientiert sich anhand des oben vorgestellten CRISP-DM Standards. In der Modellierungsphase werden dabei als DM-Modell Methodiken des Deep Learning eingesetzt, um ein künstliches neuronales Netz zu trainieren. Evaluiert wird das Modell letztendlich durch den Vergleich des Modells mit der klassischen EVM-Prognose.

Die Schritte Geschäftsverständnis (Business Understanding) und Bereitstellung (Deployment) werden nicht berücksichtigt, da die zu lösende Problemstellung der Vorhersage von Projektendkosten bereits definiert ist und die Bereitstellung des erarbeiteten Data Mining Modells den Umfang dieser Arbeit überschreiten würde.

Die folgenden Kapitel zur Entwicklung der künstlichen Intelligenz richten sich somit an den Phasen **Datenverständnis** (Kapitel 4.3), **Datenvorbereitung** (Kapitel 4.4), **Modellierung** (Kapitel 5) und **Evaluierung** (Kapitel 7) aus.

4.2 Werkzeuge zur Datenanalyse und Aufbereitung

Die Analyse und Aufbereitung der Daten wird mittels der Programmiersprache Python[1] durchgeführt. Python eignet sich vor allem für wissenschaftliche Berechnungen und die Analyse und Visualisierung großer Datensätze. Drittthersteller stellen dabei kostenlose Python-Bibliotheken zur Verfügung, die im KDD-Prozess unterstützen [vgl. VanderPlas, 2018, S. 14 ff.], wie z.B.:

- **IPython**[2] als Open Source Kommandozeileninterpreter und
  Jupyter[3] als interaktive Programmierumgebung gewährleisten einen effizienten Entwicklungsprozess.
- **NumPy**[4] ermöglicht eine einfache Handhabung mehrdimensionaler Arrays und stellt effizient implementierte Funktionen für wissenschaftliche Berechnungen zur Verfügung.
- **Pandas**[5] bietet die effiziente Speicherung sogenannter *DataFrame*-Objekten an, welche die Daten in Zeilen und Spalten orientierter Weise darstellen.
- **Matplotlib**[6] ist eine Bibliothek, um Daten über Diagramme visuell darstellen zu können.
- **Scikit-Learn**[7] stellt eine effiziente Implementierung der wichtigsten Machine Learning Algorithmen zur Verfügung und unterstützt bei der Datenvorbereitung.

Die oben genannten Programmpakete werden bei Bearbeitung der Problemstellung dieser Arbeit verwendet. Die erzielten Ergebnisse werden über Jupyter Notebooks bereitgestellt (siehe Anhang).

4.3 Datenverständnis

Für die Vorhersage der Projektendkosten wird ein großer Datensatz historischer Projekt- und Finanzdaten von 5.582 Projekten herangezogen. Die Daten wurden aus einer - von der Firma

[1] https://www.python.org/
[2] http://ipython.org/
[3] https://jupyter.org/
[4] https://numpy.org/
[5] https://pandas.pydata.org/
[6] https://matplotlib.org/
[7] https://scikit-learn.org/

exentra GmbH entwickelten - Projektmanagementsoftware als CSV (Comma-separated values) exportiert. Insgesamt umfasst dieser Export drei Tabellen: **Faktura**, **Bestand** und **Kosten**.

Die exportierten Daten sind als Zeitreihe zu interpretieren, da die Zeilen eine zeitliche Abfolge der Projektereignisse darstellen. Die Tabellen **Faktura** und **Bestand** spielen später bei der Berechnung des Earned Values (siehe Kapitel 6.1) eine wichtige Rolle, da diese Rückschlüsse auf den Fertigstellungswert eines Projektes schließen lassen. Die Zeilen der Faktura-Tabelle stellen dabei die in Rechnung gestellte Leistung dar, also die gebuchten Ausgangsrechnungen an den Kunden, während die Bestandstabelle die verbuchte Arbeit repräsentiert. Kapitel 6 erklärt den genaueren Zusammenhang zwischen den Faktura-, und Bestandsdaten zur Berechnung des Earned Values.

Die wichtigsten Daten stellt die **Kostentabelle** dar, deren Merkmale in Tabelle 4.1 abgebildet sind. Die Kostentabelle besteht aus 967.896 Zeilen an Projektbuchungen und wird benötigt zur Berechnung des Planned Value und der Actual Cost eines Projektes. Eine Buchung kann dabei entweder vom Typ Mitarbeiterzeit, Maschinenzeit oder Eingangsrechnung sein und ist genau einer Projektposition eines Projektes zugeordnet. Ein Projekt besitzt mindestens eine Projektposition und ist durch die Projektnummer (*pj_number*) eindeutig identifizierbar.

Merkmal	Beschreibung	Skala
date	Datum der Buchung	quantitativ, Intervall
costs	Die entstandenen Kosten der Buchung	quantitativ, Verhältnis
pj_task_id	Schlüssel der Projektposition	quantitativ, Intervall
pj_number	Projektnummer	quantitativ, Intervall
pj_customer_id	Kundennummer	quantitativ, Intervall
pj_task_planned_costs	Planned Value der Projektposition	quantitativ, Verhältnis
pj_task_budget	Für diese Projektposition vom Kunden freigegebene Budget.	quantitativ, Verhältnis
type	Buchungstyp	qualitativ, Nominal
pj_start_date	Projektstartdatum	quantitativ, Intervall
pj_order_date	Projektauftragsdatum	quantitativ, Intervall
pj_project_lead_id	Identifikationsschlüssel des Projektmanagers	quantitativ, Intervall
duration	Buchungsdauer	quantitativ, Verhältnis
employee_id	Identifikationsschlüssel der Arbeitskraft	quantitativ, Intervall
em_start_date	Datum des Unternehmenseintritts der Arbeitskraft	quantitativ, Intervall
em_team_id	Projektteam-Schlüssel der Arbeitskraft	quantitativ, Intervall
em_hourly_rate	Stundensatz der Arbeitskraft	quantitativ, Verhältnis
em_age	Alter der Arbeitskraft	quantitativ, Verhältnis
em_job_description	Stellenbezeichnung	qualitativ, Nominal

Tabelle 4.1: Merkmale der Kostentabelle

Handelt es sich bei der Buchung um eine Mitarbeiterbuchung sind zusätzliche Daten wie das Alter (*em_age*), die Stellenbezeichnung (*em_job_description*), der Projektteam-Schlüssel (*em_team_id*) oder der Stundensatz (*em_hourly_rate*) der Arbeitskraft verfügbar.

Zunächst wird die prozentuale Häufigkeitsverteilung der Buchungstypen (siehe Abb. 4.2) betrachtet. Dabei fällt auf, dass über 90% der Buchungen vom Typ Mitarbeiter sind, während Maschinen mit 0.7% einen deutlich geringeren Anteil ausmachen. Auf Projektebene hatten insgesamt 482 Projekte Maschinen im Einsatz, was ca. 8.6% der Gesamtmenge entspricht. Maschinenkosten sind dabei als sehr vorhersehbar einzustufen, da diese einen stark linearen Verlauf haben. Aufgrund des geringen Vorkommens und der Linearität der Maschinenbuchungen werden diese Projekte von der Projektmenge entfernt.

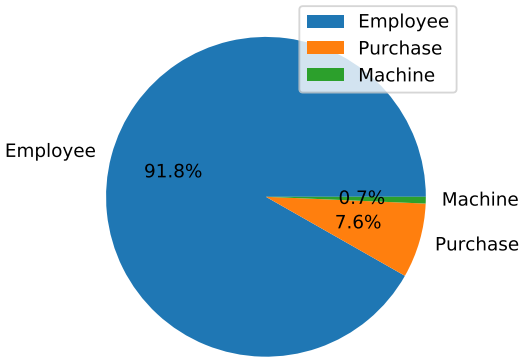

Abbildung 4.2: Prozentuale Häufigkeitsverteilung der Buchungstypen [Eigene Abbildung]

Als nächstes wird die Anzahl der Projektbuchungen B untersucht, da diese Auskunft über die Größe und den Verlauf eines Projektes liefert. Durchschnittlich besitzen Projekte bis zu 179 Buchungen. Die Spannweite der Buchungen reicht von einer bis hin zu einem Maximum von ca. 25.000 Buchungen pro Projekt. Dabei haben ungefähr 27% aller Projekte weniger als 10 und 0.9% mehr als 2500 Buchungen. Die Diskrepanz wird durch die Standardabweichung von 704 Buchungen unterstrichen. Aufgrund der sehr hohen Streuung werden deshalb alle Projekte außerhalb des Wertebereichs $10 \leq B \leq 2500$ ausgeschlossen. Die bereinigte Projektmenge führt damit zu einer typischen rechtsschiefen Häufigkeitsverteilung [vgl. Pflaumer u. a., 2017, S. 48], die in Abbildung 4.3 dargestellt ist. Dort wird deutlich, dass die Mehrheit aller Projekte eine geringe Anzahl an Projektbuchungen aufweist.

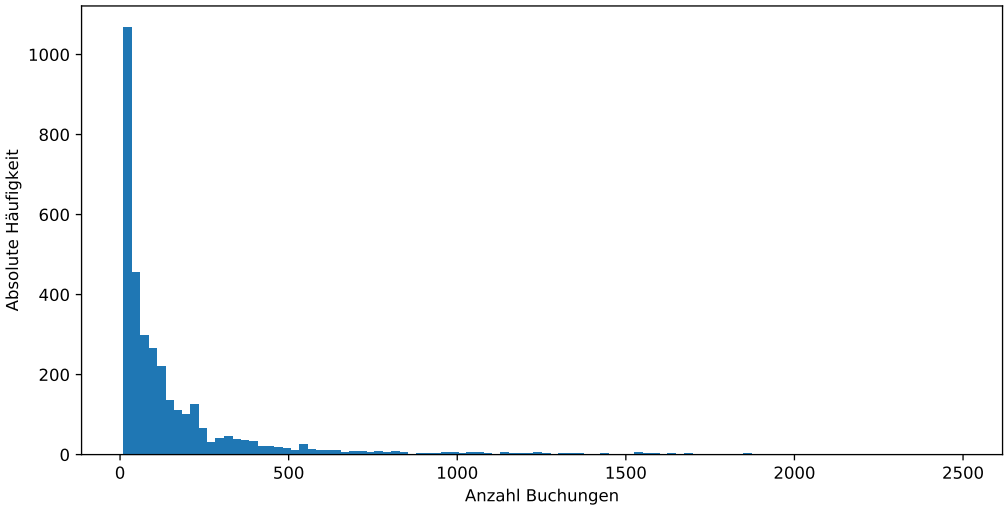

Abbildung 4.3: Häufigkeitsverteilung der Buchungen pro Projekt [Eigene Abbildung]

Nachdem nun ein besseres Verständnis der Daten vorliegt und bereits erste Datenstreuungen bereinigt wurden, können im nächsten Schritt die EVM-Basiskennzahlen *Actual Cost* (AC) und *Planned Value* (PV) berechnet werden. Jede Projektposition t — aus einer Menge von T Projektposition pro Projekt — pflegt Plankosten t_{plan} (*pj_task_planned_costs*) und ein freigegebenes Budget vom Kunden (*pj_task_budget*). Die summierten Plankosten aller Projektpositionen eines Projektes P ergeben dabei den Planned Value des Projektes in Euro:

$$P_{PV} = \sum_{t \in T} t_{plan}$$

Nach Berechnung des PV wird deutlich, dass bei rund 30% aller Projekte kein Planwert gepflegt wurde. Dies bestätigt die in der Praxis häufig festgestellte unvollständige Projektplanung [vgl. GPM, 2013].

Das quantitative Merkmal *costs* der Kostentabelle stellt die Kosten einer einzelnen Buchung b dar. Um die *Actual Cost* (AC) eines Projektes in Euro zu berechnen, werden die einzelnen Kosten der Projektbuchungen B eines Projektes aufsummiert:

$$P_{AC} = \sum_{b \in B} b_{costs}$$

Projekte, bei denen die Actual Cost (AC) negativ oder null sind, werden vom Datensatz entfernt. Außerdem werden Ausreißer, wie Projekte mit Actual Cost größer als 1.500.000 Euro, bereinigt, da diese sonst zu einer hohen Streuung der Daten führen. Abbildung 4.4 visualisiert den Zusammenhang der EVM-Basiskennzahlen. Übereinstimmt dabei der PV mit den AC, so wurde das Projekt laut Plan beendet. Abbildung 4.4 zeigt auf der linken Seite die Korrelation der Plan- und IST-Kosten als Streudiagramm. Hierbei wird die lineare Relation der Kennzahlen deutlich. Dennoch fällt auf, dass der PV und die AC einiger Projekte stark voneinander abweichen.

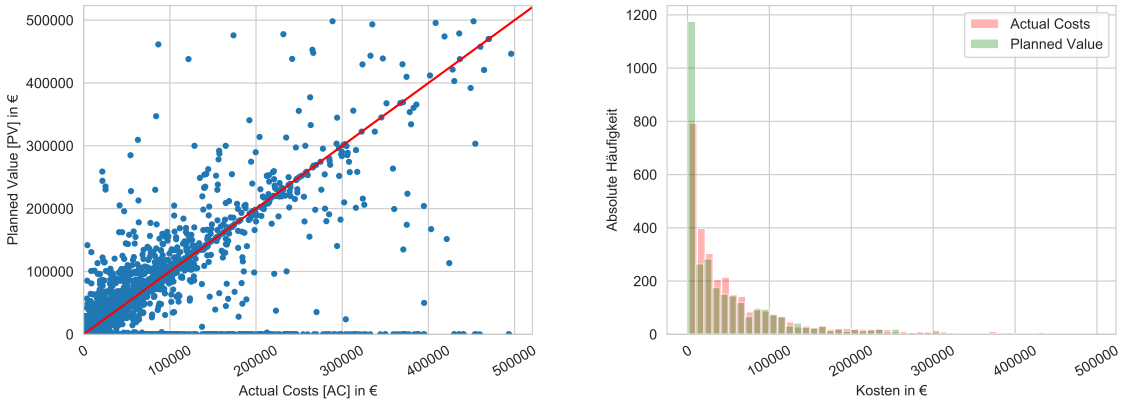

Abbildung 4.4: Korrelation der EVM-Kennzahlen [Eigene Abbildung]

Das Histogramm auf der rechten Seite (4.4) weist eine typische rechtsschiefe Häufigkeitsverteilung auf und veranschaulicht, dass Projekte mit geringem Planned Value, eher dazu neigen das Budget einzuhalten. Diese These wird von der Standish Group durchgeführten Studie *CHAOS Report* [The Standish Group International, 2015] unterstützt, worin festgestellt wird, dass kleinere Projekte vorwiegend erfolgreicher abschließen als größere.

Nun wurde ein erster Blick auf die Daten geworfen und einige Streuungen bereinigt. Außerdem fand die Berechnung der für das EVM relevanten Basiskennzahlen PV und AC statt, welche im späteren Verlauf der Arbeit benötigt werden. Im Folgenden werden die Daten aufbereitet, damit Data Mining Algorithmen angewendet werden können.

4.4 Datenvorbereitung

Datenbanken in der realen Welt sind aufgrund menschlicher Eingabefehler sehr anfällig für verrauschte, fehlende und inkonsistente Daten. Diese Datenfehler führen zu einer schlechten Datenqualität, was wiederum die Resultate der Data Mining Modelle beeinflusst [vgl. Han u. a., 2012, S. 83]. Die Datenvorbereitung - im ML auch als *Feature Engineering* bezeichnet [vgl. Schwaiger und Steinwendner, 2019, S. 281] - beschäftigt sich deshalb mit Techniken um die Daten in ein für Data Mining Algorithmen verständliches Schema zu transportieren und setzt sich dabei unter anderem aus Prozessen der Transformation, Bereinigung und Normalisierung von Daten zusammen [vgl. García u. a., 2016, S. 11; Han u. a., 2012, S. 83].
Die folgenden Abschnitte durchlaufen den Prozess der Datenvorbereitung. Besonderer Fokus wird dabei auf die Extraktion und Auswahl der Merkmale gelegt, da diese später ausschlaggebend für die Performance des künstlichen neuronalen Netzes sein werden.

4.4.1 Datentransformation

Damit Data Mining Algorithmen realistische Ergebnisse erzielen und effizient arbeiten können, sollten die aus Kapitel 4.3 vorgestellten Daten in ein für diese Algorithmen verständliches Schema konvertiert werden. Hierfür sind Teilschritte der Aggregation notwendig.

Bisher liegen die Daten als Zeitreihe von Projektbuchungen vor. Da ein zeitunabhängiges Modell zur Vorhersage der Projektendkosten trainiert werden soll, ist es nötig eine Datenreduktion durchzuführen und die bisherigen Daten auf Projektebene zu aggregieren. Das Zieldatenschema soll in den Zeilen dabei keine zeitlich abhängige Projektbuchung, sondern ein einzelnes Projekt repräsentieren, das als y-Wert die Actual Cost beinhaltet (siehe Abb. 4.5).

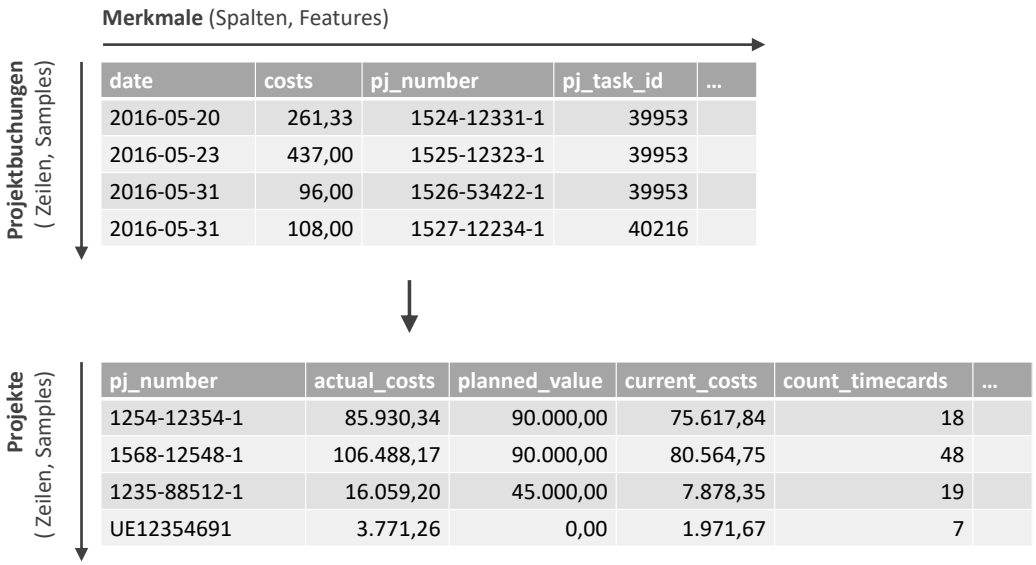

Abbildung 4.5: Aggregation der Daten auf Projektebene [Eigene Abbildung]

Die Merkmale des Projektes beziehen sich auf den Zeitpunkt t_i, der 50% der getätigten Buchungen dargestellt. Damit ist der Zeitpunkt gemeint, zu dem die Hälfte aller Projektbuchungen eines Projektes getätigt wurden, also der Median der Projektlaufzeit. Dieser eignet sich gut zur Prognose der Projektendkosten, da zu diesem Zeitpunkt bereits ausreichend Projektinformationen zur Vorhersage vorliegen und außerdem zeitlich noch die Möglichkeit besteht das Projekt über Maßnahmen zu steuern.

Bei genauerer Untersuchung der Daten können die in Tabelle 4.2 aufgezählten Merkmale festgestellt werden. Das Merkmal *current_costs* bezeichnet hierbei die zum Zeitpunkt t_i angefallenen Projektkosten.

Merkmal	**Beschreibung**	**Skala**
actual_cost	Actual Cost (y-Wert) des Projektes	quantitativ, Verhältnis
planned_value	Planned Value des Projektes	quantitativ, Verhältnis
current_costs	Die im Projekt angefallenen Kosten zum Zeitpunkt t_i	quantitativ, Verhältnis
pj_budget	Für das Projekt freigegebene Budget vom Kunden	quantitativ, Verhältnis
count_timecards	Anzahl der bis zum Zeitpunkt t_i getätigten Buchungen vom Typ Mitarbeiter.	quantitativ, Verhältnis
count_purchase	Anzahl der bis zum Zeitpunkt t_i getätigten Buchungen vom Typ Eingangsrechnung	quantitativ, Verhältnis
count_pj_tasks	Anzahl der Projektpositionen zum Zeitpunkt t_i	quantitativ, Verhältnis
pj_project_lead_id	Identifikationsschlüssel des Projektleiters	quantitativ, Intervall
pj_customer_id	Identifikationsschlüssel des Projektauftraggebers / Projektkunden	quantitativ, Intervall
days_pj_start_order	Anzahl Tage zwischen Projektstart und Projektangebot	quantitativ, Verhältnis
Anzahl Arbeitskräfte nach Stellenbezeichnung	Die zum Zeitpunkt t_i am Projekt beteiligten Arbeitskräfte aufgeschlüsselt nach Stellenbezeichnung	quantitativ, Verhältnis
mean_em_age	Altersdurchschnitt der am Projekt beteiligten Arbeitskräfte zum Zeitpunkt t_i	quantitativ, Verhältnis
mean_em_hourly_rate	Durchschnittlicher Stundensatz der am Projekt beteiligten Arbeitskräfte zum Zeitpunkt t_i	quantitativ, Verhältnis
sum_duration	Summe der Dauer aller Buchungen eines Projektes in Stunden zum Zeitpunkt t_i	quantitativ, Verhältnis

Tabelle 4.2: Aufbereitete Projektmerkmale

Die Merkmale aus Tabelle 4.2 wurden durch diverse Operationen der Summen- und Gruppenbildung, mittels der bereits vorgestellten Python Bibliotheken NumPy und Pandas, extrahiert.

4.4.2 Datenbereinigung

Nachdem nun die Merkmale für das neuronale Netz extrahiert und erstellt wurden, ist es nötig die Daten auf Inkonsistenz und fehlende Werte zu untersuchen.

Bei manchen Projekten ist kein Wert für die Kundennummer gepflegt (*pj_customer_id*). Diese Werte zu bereinigen ist schwer, deshalb wird zunächst anhand des Kundennamen untersucht, ob redundant gepflegte Kunden in den Daten vorhanden sind. Die redundanten Einträge werden dann auf eine Kundenummer zusammengefasst und die restlichen Projekte mit fehlender Kundenummer aus der Datenmenge entfernt. Weiterhin ist der Altersdurchschnitt der Arbeitskräfte (*mean_em_age*) bei einigen Projekten nicht gepflegt. Hierbei wird für die fehlenden Werte und Nullwerte der Mittelwert der Spalte als Wert herangezogen. Dasselbe Vorgehen wird zur Bereinigung der Spalten *days_pj_start_order* und *mean_em_hourly_rate* angewendet. Dort wird der Mittelwert der Anzahl der Tage zwischen Projektstart und Projektauftrag und des durchschnittlichen Stundensatzes aller Projekte für die fehlenden Werte eingesetzt. Einige Projekte bestehen ausschließlich aus Buchungen vom Typ *Purchase*. Bei solchen Projekten sind keine Mitarbeiterdaten verfügbar, weshalb die Spalten Anzahl der Arbeitskräfte nach Stellenbezeichnung, *mean_em_age*, *mean_em_hourly_rate* und *sum_duration* keine definierten Werte besitzen. Aus diesem Grund werden hier die fehlenden Daten mit 0 ersetzt.

4.4.3 Dimensionsreduktion

In den vorangegangenen Abschnitten wurden mehrere Merkmale aus der Datengrundlage erstellt. Bisher wurde jedoch noch nicht untersucht, ob all diese Merkmale tatsächlich nützlich für das spätere Modell sind. Generell gibt es nach Nguyen und Zeigermann [vgl. Nguyen und Zeigermann, 2018, S. 105 f.] drei Gründe, welche für eine Reduzierung der Merkmale sprechen:

1. Vermeidung von Überanpassung bei komplexen Modellen.
2. Reduzierung der Trainingszeit.
3. Bessere Interpretierbarkeit der Daten.

Vor allem der erste Aspekt spielt eine wichtige Rolle. Je höher die Anzahl der Merkmale desto größer ist die Dimensionalität des Merkmalsraums. Ist die Dimensionalität des Merkmalsraums sehr hoch, werden viele Daten benötigt um dessen Komplexität erfassen und darüber Aussagen machen zu können. Eine Überanpassung entsteht dabei sowohl wenn zu wenige Daten zur Verfügung stehen oder eine zu hohe Dimensionalität gewählt wurde [vgl. Nguyen und Zeigermann, 2018, S. 106].

Eine häufig verwendete Methode um die Beziehung zwischen zwei numerischen Merkmalen *A* und *B* zu verstehen, ist die Berechnung des Korrelationskoeffizienten r (auch Pearson Produkt-Moment-Korrelation genannt) (4.1) [Han u. a., 2012, S. 96]:

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B} \tag{4.1}$$

Dabei steht *n* für die Anzahl der Tupel und a_i, b_i für die Werte von *A* und *B* in Tupel *i*, wobei gilt $-1 \leq r_{A,B} \leq 1$. σ_A und σ_B bezeichnen die jeweiligen Standardabweichungen der Merkmale. Eine positive lineare Korrelation liegt vor, wenn $r_{A,B} > 0$, was bedeutet, dass bei Steigung der Werte von *A* auch die Werte von *B* steigen. Deren Korrelation wird dabei stärker,

je näher $r_{A,B}$ gegen eins strebt. Umgekehrt sind *A* und *B* negativ linear korreliert, sobald $r_{A,B} < 0$. Das ist dann der Fall, wenn der Wert eines Merkmals steigt, sobald der Wert des anderen Merkmals sinkt. *A* und *B* sind voneinander unabhängig, wenn $r_{A,B}$ gleich null ist.

Eine starke Korrelation ist ein Indikator dafür, dass die beiden Merkmale eine hohe Ähnlichkeit aufweisen und möglicherweise ähnliche Informationen tragen. Aus diesem Grund kann eines der beiden Merkmale meistens entfernt werden.

Abbildung 4.6 visualisiert über eine Heatmap die Korrelationskoeffizienten der Merkmale zueinander. Der Übersichtlichkeit halber sind die Anzahl der Stellenbezeichnungen als Merkmale nicht in der Heatmap aufgeführt. Die rötlichen Bereiche weisen auf eine eher höhere Korrelation zwischen den Merkmalen hin. Vor allem die Merkmale *pj_budget* und *planned_value* sind mit einer positiven linearen Korrelation von 0.94 sehr stark korreliert. Dieselbe Korrelation weisen außerdem die Attribute *sum_duration* und *count_timecards* auf. Aufgrund der hohen positiven Korrelation und damit starken Ähnlichkeit zwischen den Merkmalen, werden die Merkmale *count_timecards* und *pj_budget* von der Merkmalsmenge entfernt.

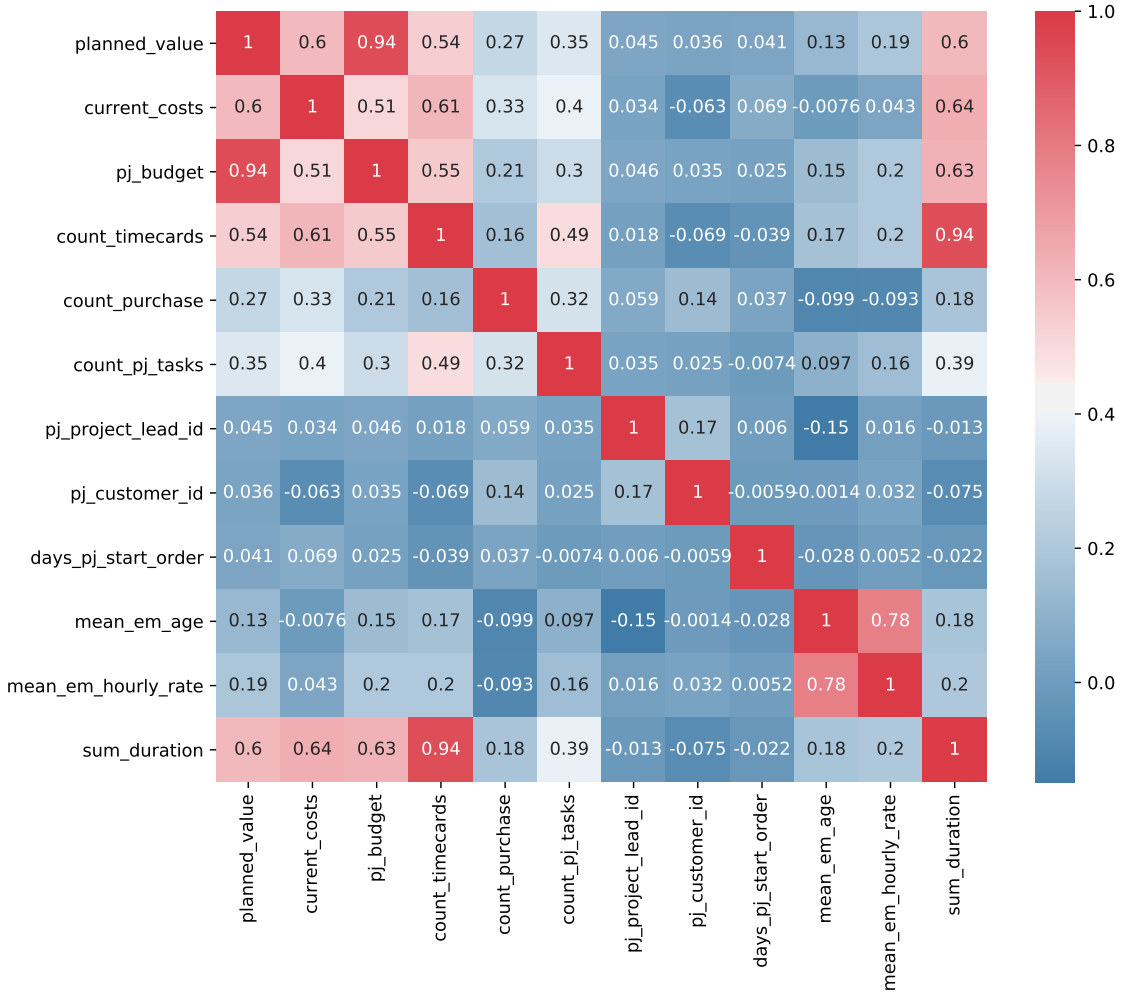

Abbildung 4.6: Korrelationsmatrix der Projektmerkmale als Heatmap Diagramm [Eigene Abbildung]

mean_em_hourly_rate und *mean_em_age* sind mit einem Korrelationskoeffizienten von 0.78 auch stärker zueinander korreliert. Diese Korrelation lässt sich dadurch erklären, dass ältere

und damit erfahrenere Mitarbeiter/innen tendenziell einen höheren Stundensatz im Vergleich zu jüngeren Angestellten haben. Aufgrund der positiven Korrelation der beiden Merkmale wird *mean_em_age* abschließend auch von der Merkmalsmenge entfernt.

4.4.4 Normalisierung

Merkmale haben häufig sehr unterschiedliche Wertebereiche. Beispielsweise unterscheiden sich der Planned Value in Euro (*planned_value*) um mehrere Zehnerpotenzen von dem Altersdurchschnitt der Arbeitskräfte (*mean_em_age*). Diese Größenunterschiede können bei Modellen dazu führen, dass numerisch kleinere Merkmale schwächer gewichtet werden als numerisch größere Merkmale [vgl. Runkler, 2010, S. 31]. Damit allen Merkmalen einheitliche Gewichte zugeordnet werden, ist es wichtig die Daten über Normalisierung (auch Standardisierung genannt) in einen gemeinsamen Wertebereich zu überführen. Hierfür existieren mehrere Methodiken, wobei im Folgenden zwei davon vorgestellt werden.

Min-Max-Normalisierung

Die Min-Max-Normalisierung führt eine lineare Transformation durch und normalisiert die Daten in einen neuen Wertebereich zwischen new_min_A und new_max_A. Dabei wird der Wert des Attributs A auf einen normalisierten Wert v_i' abgebildet, wobei min_A und max_A den minimalen und maximalen Wert des Attributs A repräsentieren. Folgende Formel (4.2) zeigt die Berechnung der Min-Max-Normalisierung [Han u. a., 2012, S. 114]:

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{4.2}$$

Im Rahmen der Arbeit wird die Min-Max-Normalisierung zur Skalierung der Merkmale bzw. x-Werte eingesetzt. Diese werden dabei auf einen neuen Wertebereich zwischen 0 und 1 transformiert, damit das neuronale Netz die Eingabedaten besser verarbeiten kann.

Quantile-Normalisierung

Die Quantile-Normalisierung ist eine Methode, welche Daten entsprechend einer gleichmäßigen Verteilung auf den Wertebereich zwischen 0 und 1 normalisiert. Diese Methode ist sehr robust gegenüber Datenausreißer [scikit learn, 2019]. Da die Verteilung der Projektendkosten viele Ausreißer aufweist, wird zur Skalierung der y-Werte, also der Zielwerte die durch das neuronale Netz vorhergesagt werden sollen, die Quantile-Normalisierung angewendet. Dadurch werden die Endkosten der einzelnen Projekte einheitlich und gleichmäßig auf den Wertebereich [0, 1] skaliert ohne dabei Ausreißer zu besitzen.

5 Implementierung und Training des KI-Modells

5.1 Vorstellung des KI-Frameworks

Die Implementierung einfacher neuronaler Netze ist zwar selbstständig möglich, nimmt jedoch sehr viel Zeit in Anspruch und schränkt die Flexibilität beim Testen ein. Um schnelle Ergebnisse erzielen zu können, bietet es sich an auf bereits vorhandene Frameworks zuzugreifen, welche Machine Learning Algorithmen auf effiziente Weise implementiert haben. In Rahmen dieser Arbeit werden zur Entwicklung des neuronalen Netzes die Bibliotheken TensorFlow als Low-Level-API und Keras als High-Level-API eingesetzt. Unter einer Low-Level-API ist dabei eine Bibliothek gemeint, die essenzielle Basisoperationen des Deep Learning implementiert und eine High-Level-API greift auf diese zu und stellt die Anwendung der Operationen vereinfacht zur Verfügung. Dieses Kapitel stellt die im Rahmen dieser Arbeit verwendeten Bibliotheken genauer vor.

5.1.1 TensorFlow

TensorFlow[1] ist eine Open-Source-Bibliothek welche Google für das maschinelle Lernen entwickelte. Ursprünglich wurde die Bibliothek für den internen Einsatz bei Google implementiert um Deep Learning Algorithmen für Produkte, wie der Suchmaschine oder Spracherkennung einzusetzen. 2015 wurde das Framework als Open-Source-Software frei zur Verfügung gestellt und verfügt seither über einen hohen Nutzungsgrad.

TensorFlow bildet die Berechnungen intern als Datenflussgraphen ab. Die Knoten des Graphen stehen dabei für numerische Operationen, wie der Addition oder Multiplikation, während entlang der Kanten die eigentlichen Daten fließen (*flow*). Daten werden im Kontext von TensorFlow auch als *Tensoren* bezeichnet. Ein *Tensor* repräsentiert ein n-dimensionales Array, was Tabellen oder Matrizen beliebiger Dimensionen darstellt. Im Wesentlichen orientiert sich die Architektur demnach stark an Implementierung maschineller Lernalgorithmen mit starkem Fokus auf Deep Learning Modelle [Tensorflow, 2019; vgl. Hope u. a., 2018, S. 5 ff.].

5.1.2 Keras

Keras[2] ist ein Deep Learning Framework für Python, das aufgrund der Bereitstellung von High-Level-Bausteinen die Definition und das Training von Deep Learning Modellen vereinfacht. Die Bibliothek bietet dabei selbst keine Low-Level-Operationen - wie die Bearbeitung von Tensoren - an, sondern greift auf eine hochoptimierte Tensorbibliothek zurück, die im Jargon von Keras auch als *Backend-Engine* bezeichnet wird. Keras unterstützt mehrere Backend-Engines wie z.B. TensorFlow, Theano[3] oder CNTK[4]. Dabei kann jeglicher mit Keras entwickelter Code ohne weiteren Änderungen von allen Backend-Engines ausgeführt werden, was zu einer sehr hohen Flexibilität in der Entwicklung führt. Keras lässt sich über TensorFlow außerdem auf der CPU oder der GPU ausführen [vgl. Chollet und Lorenzen, 2018, S. 91].

[1] https://www.tensorflow.org
[2] https://keras.io
[3] http://deeplearning.net/software/theano/
[4] https://docs.microsoft.com/de-de/cognitive-toolkit/

Im Verlauf der Arbeit wird standardmäßig TensorFlow als Backend-Engine mit Ausführung auf der CPU genutzt, da diese derzeit am weitesten verbreitet ist und einen hohen Reifegrad besitzt [vgl. Chollet und Lorenzen, 2018, S. 91].

5.2 Anwendung eines neuronalen Netzes zur Vorhersage der Projektendkosten

In diesem Kapitel wird ein neuronales Netz zur Berechnung der Endkosten eines Projektes erstellt, um die Ergebnisse später mit der EVM-Methode vergleichen zu können. Bevor das neuronale Netz trainiert werden kann, ist es nötig die Datenmenge in Trainings- und Testdaten aufzuteilen. Im Grundlagenkapitel (3.2) wurde bereits erläutert, dass die Trainingsdaten dazu dienen, das Modell zu trainieren und die Testdaten benötigt werden, um die Vorhersagegenauigkeit des Modells zu überprüfen. Im Rahmen dieser Arbeit werden 80% der Daten zu Trainingszwecken und die restlichen 20% als Testmenge verwendet.

Generell soll das neuronale Netz eine Regressionsaufgabe lösen, da die Projektendkosten eine stetige Zahl darstellen. Als Eingabedaten für den Input-Layer des neuronalen Netzes, werden die in Kapitel 4 erarbeiteten und normalisierten Merkmale verwendet. Nachfolgend wird ein tiefes neuronales Netz, bestehend aus insgesamt 3 Hidden-Layer und einem Output-Layer, implementiert. Der erste Hidden-Layer besteht aus 15 Neuronen und besitzt damit dieselbe Anzahl an Neuronen wie es Eingabesignale gibt. Die zweite Schicht setzt sich aus 10 Neuronen zusammen und die dritte Schicht umfasst 5 Neuronen. Da das Modell am Ende nur eine Kennzahl — nämlich die Endkosten eines Projektes — vorhersagen soll, besteht die letzte Schicht, der sogenannte Output-Layer, aus einem einzelnen Neuron. Der Output-Layer liefert den Vorhersagewert $\hat{y}$. Abbildung 5.1 visualisiert den Aufbau des neuronalen Netzes, wobei n die Anzahl der Neuronen in der jeweiligen Schicht darstellt.

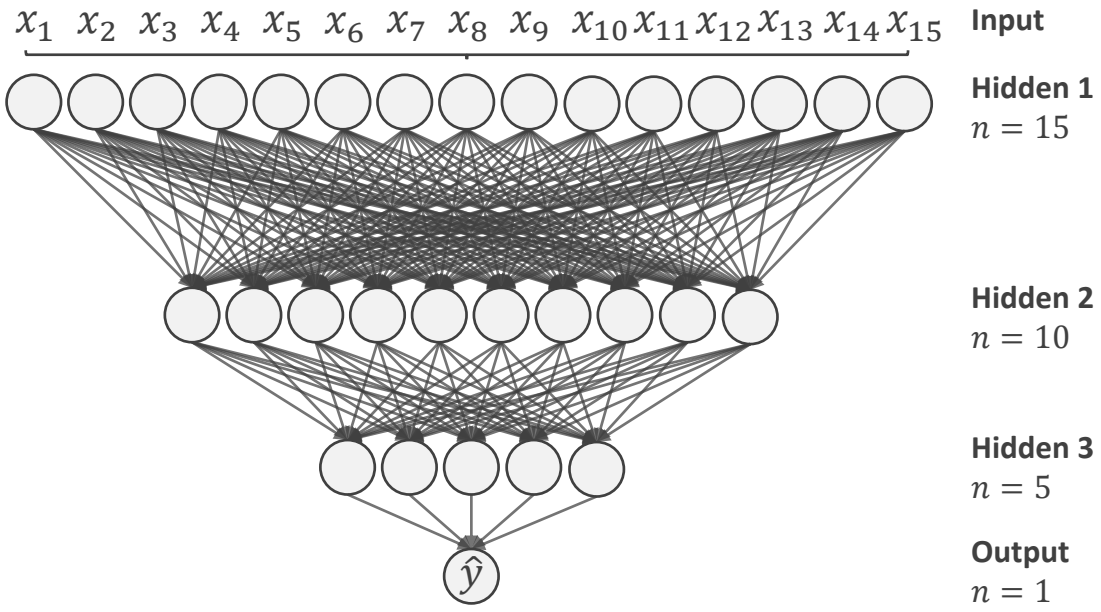

Abbildung 5.1: Aufbau des neuronalen Netzes zur Vorhersage der Projektendkosten [Eigene Abbildung]

Die vorgestellte Netzarchitektur wurde gewählt, da diese beim Test mit mehreren Modellen die besten Ergebnisse erzielte und damit einen guten Generalisierungsgrad besitzt. Das neuronale Netz wird mittels der in Abschnitt 5.1 vorgestellten Python Bibliothek Keras implementiert. Der hierfür erstellte Python-Code ist nachfolgend dargestellt und wird nun genauer erklärt (siehe Programmcode aus Abbildung 5.2).

```
1  from keras.models import Sequential
2  from keras.layers import Dense
3
4  model = Sequential()
5
6  model.add(Dense(len(X.keys()), activation="relu",
7        input_shape=[len(X.keys())]))
8  model.add(Dense(10, activation="relu"))
9  model.add(Dense(5, activation="relu"))
10 model.add(Dense(1))
11
12 model.compile(optimizer="adam", loss="mse")
13
14 history = model.fit(
15     X_train,
16     y_train,
17     epochs=200,
18     validation_split=0.20,
19     batch_size=10)
```

Abbildung 5.2: Ausschnitt eines Python Programmcode zur Erstellung eines neuronalen Netzes [Eigene Abbildung]

Keras bietet zur Definition eines Modells zwei Möglichkeiten an. Zum einen die Sequentiale Schnittstelle, die für linear gestapelte Layer gedacht ist und zum anderen die funktionale API, welche beliebige Architekturen ermöglicht und vor allem für gerichtete azyklische Graphen aus Layern gedacht ist [vgl. Chollet und Lorenzen, 2018, S. 92]. In dieser Arbeit wird die Sequentiale Klasse verwendet, da diese mit einem geringeren Implementierungsaufwand einhergeht und für den Anwendungsfall dieser Arbeit ausreichend ist (siehe Zuweisung der Modellschnittstelle in Zeile 4 des Programmcodes).

Die Zeilen 6-10 des oben abgebildeten Programmcodes beschreiben den zu Beginn bereits erläuterten Aufbau des 4-schichtigen neuronalen Netzes. Über die *add()*-Methode wird dem Sequentiellen Modell dabei ein Layer hinzugefügt. Der Layer ist in diesem Fall von dem Typ *Dense*. Dense Layer eignen sich vor allem sehr gut für die Verarbeitung von zwei-dimensionalen-Tensoren [vgl. Chollet und Lorenzen, 2018, S. 87]. Der erste Funktionsparameter der *Dense* Klasse gibt dabei die Anzahl der Neuronen, bzw. Ausgabeeinheiten einer Schicht an, während die Variable *activation* die Bezeichnung der zu verwendeten Aktivierungsfunktion definiert. Wird die Aktivierungsfunktion wie in Zeile 10 nicht angegeben, so nutzt die Dense Klasse automatisch die Identitätsfunktion. Da die Projektendkosten einen linearen Wert annehmen können, eignet sich für die Ausgabe in Zeile 10 die Verwendung der Identitätsfunktion. Bei den Hidden-Layer wurde *ReLU* als Aktivierungsfunktion gewählt, da diese vor allem für Regressionsaufgaben sehr effizient ist. In Zeile 6 wurde weiterhin ein Funktionsparameter namens *input_shape* definiert. Die *input_shape* beschreibt die Form (*shape*) des Eingabevektors für den Dense-Layer. Wie vorhin erwähnt, arbeitet ein Dense-Layer mit zwei-dimensionalen-Tensoren. Deshalb muss in diesem Fall als Eingabedaten ein zwei-dimensionaler-Vektor angegeben werden. Dieser setzt sich aus einem Array zusammen, welches pro Dateneintrag (in diesem Fall pro Projekt) ein Array mit den Merkmalsdaten des Projektes besitzt. Die *input_shape* ist nur

bei dem ersten Layer des Modells anzugeben. Das liegt daran, dass die nachfolgenden Layer miteinander verknüpft sind. Durch diese Verknüpfung ist es möglich die *input_shape* anhand der Anzahl an Ausgabeeinheiten der vorangegangenen Schicht zu berechnen.

In Zeile 12 des Python-Codes wird das erstellte Modell kompiliert. Dabei werden der Optimierer und die Fehlerfunktion des Modells definiert. Der verwendete Adam-Optimierer ermittelt anhand der Fehlerfunktion den zu verwendeten Lernalgorithmus zur Aktualisierung der Gewichte. Dieser implementiert eine bestimmte Variante des sogenannten stochastischen Gradientenabstiegsverfahrens [vgl. Chollet und Lorenzen, 2018, S. 88]. Der Adam-Optimierer wurde hierbei gewählt, da dieser bei den Testdurchläufen die besten Ergebnisse erzielt hatte. Als Fehlerfunktion wird der Mean-Squared-Error (MSE) verwendet.

Über die *fit()*-Methode wird das Modell in den Zeilen 14-19 trainiert. Dabei werden die x und deren zugehörigen y-Werte der Trainingsdaten angegeben. Außerdem wird das Modell 200 Epochen lang trainiert. In einer Epoche werden dabei alle Trainingsdaten einmal durchlaufen. Das Modell lernt und aktualisiert seine Gewichte dabei immer alle 10 Eingabedaten, was durch den Parameter *batch_size* definiert wird. *validation_split = 0.20* gibt an, dass in jeder Epoche 20% der Trainingsdaten für Validierungszwecke genutzt werden. Validierungsdaten sind nützlich um die Performance eines Modells gegenüber unbekannten Daten pro Epoche zu messen. Abbildung 5.3 visualisiert den Verlauf der Fehlerfunktion über alle Trainingsepochen hinweg. Der MSE der Validierungsdaten nimmt dabei vor allem zu Beginn der Trainingsphase stark ab und stagniert ab Epoche 75.

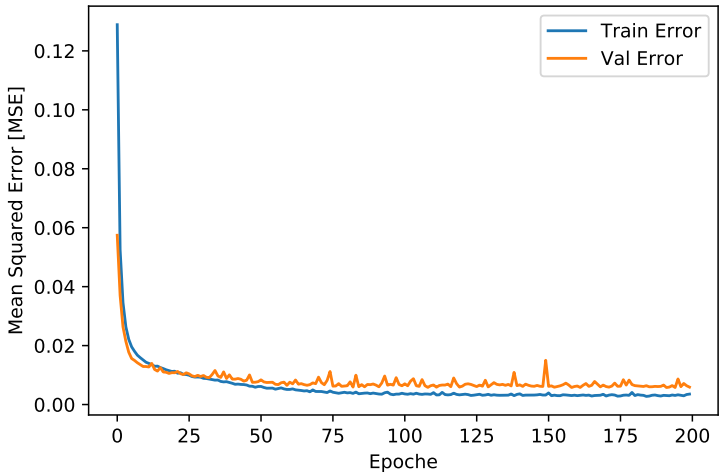

Abbildung 5.3: Verlauf der Fehlerfunktion während der Trainingsphase des neuronalen Netzes [Eigene Abbildung]

Die Fehlerwerte der Trainingsdaten sind minimal geringer als die der Validierungsdaten. Das liegt daran, dass sich der Algorithmus während dem Training stärker an die trainierten Daten angepasst hat.

Die Testdaten können nun herangezogen werden um die Vorhersagegenauigkeit des erstellten neuronalen Netzes zu testen. Hierfür werden die einzelnen Output-Werte $\hat{y}$ des neuronalen Netzes und dessen Zielwerte y in ihren ursprünglichen Wertebereich transformiert und deren Korrelation zueinander als Streudiagramm visualisiert (siehe Abbildung 5.4). Die Vorhersagen verhalten sich mit einem Korrelationskoeffizienten von 0.97 stark linear zu den tatsächlichen Werten. Das bedeutet, dass die Vorhersagewerte größtenteils nahe an den Zielwerten liegen.

Weiterhin ist zu erkennen, dass das neuronale Netz bei kleineren Projekten mit Projektendkosten bis zu 150.000 Euro vorwiegend präzise Vorhersagen trifft. Projekte mit Endkosten über 150.000 Euro weisen häufiger Ausreißer bei der Prognose vor. Das kann unter anderem daran liegen, dass die Trainingsdaten eine hohe Anzahl an Projekten mit Endkosten von unter 150.000 Euro beinhaltet und das Modell aus diesem Grund ein Muster bei größeren Projekten nicht präzise erkennen konnte. Generell ist es eher schwer die Projektendkosten für größere Projekte vorherzusagen, da solche Projekte meist von unterschiedlichen Faktoren abhängig sind und dadurch einen unvorhersehbaren Verlauf annehmen. Im Allgemeinen besitzt das Modell nach Abbildung 5.4 jedoch eine recht gute Performance und Vorhersagekraft mit einer leichten Streuung im oberen Wertebereich.

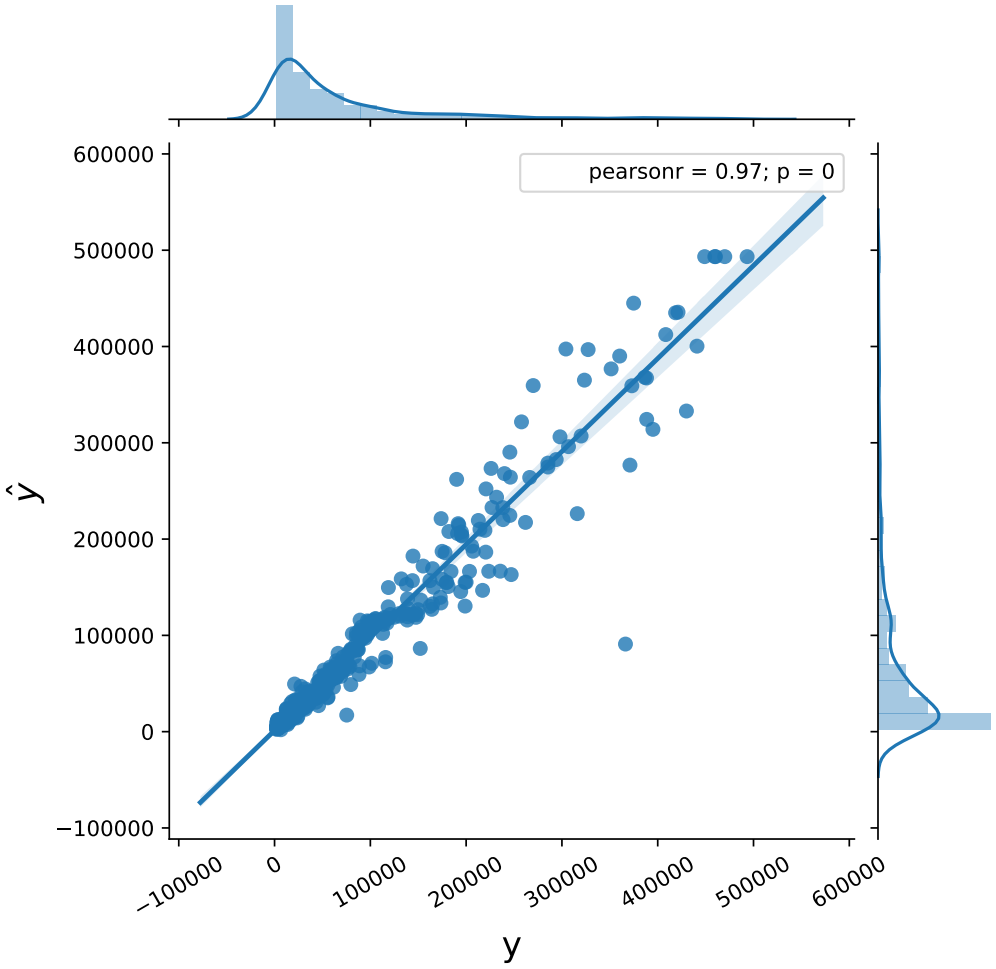

Abbildung 5.4: Korrelation der Vorhersage und der tatsächlichen Werte [Eigene Abbildung]

6 Vorhersage der Projektendkosten nach EVM

In diesem Kapitel werden 5 Beispielprojekte ausgewählt und hierfür die Estimate at Completion (EAC) nach dem bereits kennengelernten EVM-Verfahren manuell berechnet. Dazu ist im ersten Schritt die Berechnung des Earned Value nötig.

6.1 Bestimmung des Earned Value

In Kapitel 4.3 wurden die zur Verfügung stehenden Daten vorgestellt. Dabei wurde erklärt, dass die Fakturatabelle und Bestandstabelle bei der Berechnung des Earned Value eine wichtige Rolle spielen. Die Fakturatabelle repräsentiert die in Rechnung gestellte Leistung eines Projektes in Euro, bzw. die Ausgangsrechnungen an den Kunden. Diese Daten liefern Informationen darüber, welche Leistungen dem Kunden gegenüber in Rechnung gestellt wurden. Diese Daten hängen mit den Bestandsbuchungen der Bestandstabelle zusammen. Der Bestand steigt durch die im Projekt erbrachte Leistung und nimmt ab, wenn diese Leistung dem Kunden gegenüber in Rechnung gestellt wird. Der Earned Value, welcher den Fertigstellungswert eines Projektes repräsentiert, kann in diesem Fall also durch die Summe aus der Bestandsbuchung B und den kumulierten Fakturabuchungen F zum Zeitpunkt t berechnet werden, wobei n die Menge aller getätigten Fakturabuchungen bis zum Zeitpunkt t darstellt:

$$EV_t = \sum_{i=1}^{n} F_i + B_t \tag{6.1}$$

Generell besitzen manche Projekte in der Datenmenge schlecht gepflegte Projektdaten, was die Berechnung des Earned Value erschwert. Auch bei kleinen Projekten, die nur wenige Bestands- und Fakturabuchungen aufweisen, ist die Earned Value Berechnung schwer durchführbar. Bei solchen Projekten kann es sich anbieten, die Fakturadaten als Earned Value zu verwenden. Da die Fakturadaten alleine den Fertigstellungswert häufig jedoch nicht realitätsgetreu abbilden, wird diese Methodik zur Berechnung des EV nicht verwendet.

Im Folgenden Kapitel, werden 5 unterschiedliche Beispielprojekte ausgewählt und analysiert, anhand deren später die Projektendkosten vorhergesagt werden.

6.2 Projektauswahl

Im Rahmen dieser Arbeit werden aus der bestehenden Datengrundlage 5 Projekte ausgewählt, anhand derer die EVM-Prognose mit dem neuronalen Netz verglichen wird. Abbildung 6.1 zeigt die 5 selektierten Projekte und deren EVM-Basiskennzahlen mit den Faktura und Bestandsdaten. Die Projekte wurden dabei nach verschiedenen Kriterien wie der Projektgröße und Höhe der Projektendkosten selektiert. Weiterhin wurden gezielt Projekte ausgewählt, die von relativ gut gepflegten bis hin zu schlecht geplanten Projekten reichen. Die Actual Cost des Projekt 1 übersteigen beispielsweise den Planned Value um mehr als das 5-fache. Projekt 2, 3 und 4 weisen einen typischen Projektverlauf auf, da deren Actual Cost und Earned Value gegen den Planwert konvergieren und Projekt 5 wurde unsauber geplant, da kein Planned Value gepflegt wurde. Auffällig bei den Projekten ist außerdem, dass an vereinzelten Stellen die AC und der PV abnehmen, was eher untypisch für ein Projekt ist und ein Indiz für eine mangelnde Projektführung darstellt.

Abbildung 6.1: Liniendiagramme von 5 Beispielprojekten mit deren EVM-Kennzahlen [Eigene Abbildung]

6.3 Prognose der Projektendkosten

Nachdem nun 5 Projekte ausgewählt wurden, widmet sich dieser Abschnitt der Prognose der Projektendkosten nach EVM. Zur Prognose des EAC wird die in Kapitel 2.2.2 bereits behandelte Realistische Methode *(Low-End Cumulative CPI EAC)* eingesetzt. Außerdem wird die Vorhersage — analog wie bei der KI — zum Zeitpunkt t durchgeführt, der die Hälfte aller getätigten Kostenbuchungen eines Projektes darstellt.

Im ersten Schritt wird der CPI zum Zeitpunkt t berechnet, da dieser ein Bestandteil zur Berechnung des EAC ist. Der CPI_t setzt den EV_t in Relation zu den AC_t. Nach Berechnung des CPI_t wird deutlich, dass Projekt 1 und 3 einen $CPI_t < 1$ besitzen, was bedeutet, dass diese Projekte ihr Budget zum Zeitpunkt t überzogen haben. Projekt 2, 4 und 5 besitzen einen $CPI_t \geq 1$ und haben damit effizienter als geplant gearbeitet und Kosten eingespart. Die CV_t verdeutlicht diese Erkenntnisse und zeigt die Differenz der AC_t und des EV_t (siehe Tabelle 6.1).

	Projekt 1	**Projekt 2**	**Projekt 3**	**Projekt 4**	**Projekt 5**
AC_t	103.357,50 €	69.377,62 €	4.436,65 €	37.492,50 €	10.611,49 €
EV_t	96.545,10 €	82.299,60 €	3.860,17 €	49.702,50 €	26.007,76 €
CV_t	-6.812,39 €	12.921,98 €	-576,47 €	12.210,00 €	15.396,27 €
CPI_t	0.934088	1.186255	0.870064	1.325665	2.450905
BAC	28.050,00 €	156.534,24 €	7.252,00 €	88.842,00 €	0 €
EAC	30.029,25 €	131.956,57 €	8.335,01 €	67.016,92 €	0 €
AC	179.800,00 €	156.534,12 €	8.442,40 €	88.890,00 €	27.684,33 €
VAC	-1.979,25 €	24.577,66 €	-1.083,01 €	21.825,07 €	0 €
$\Delta\|EAC - AC\|$	**149.770,74 €**	**24.577,55 €**	**107,38 €**	**21.873,07 €**	**27.684,33 €**

Tabelle 6.1: EAC-Prognose der 5 Beispielprojekte

Werden nun die EAC berechnet, so fällt auf, dass Projekt 5 aufgrund des nicht gepflegten BAC eine Projektendkostenprognose von 0 Euro aufweist. Dabei wird deutlich, dass die EVM-Methode zur Vorhersage der Projektendkosten bei unzureichender Projektplanung fehlschlägt, da der Planwert zu Projektende als Grundlage der Prognose dient. Aus diesen Gründen kann für Projekt 5 keine Prognose nach der EVM-Methode durchgeführt werden.

Bei Vergleich der EAC mit den tatsächlichen Projektendkosten (AC), sticht Projekt 1 mit einer Kostenabweichung von 149.770,74 Euro hervor. Da die AC das BAC bei Projekt 1 deutlich überschreiten, ist die VAC des Projekt 1 mit einem Wert von -1.979,25 Euro signifikant geringer als die Differenz aus EAC und AC. Daran wird deutlich, dass hier die EAC gegen das BAC konvergiert.

Projekt 3 liefert eine sehr genaue Prognose. Hier liegt die Abweichung bei 107,38 Euro. Dabei nähert sich die Prognose eher den tatsächlichen statt den geplanten Projektendkosten an. Projekt 2 und 4 liefern eine eher durchschnittliche Vorhersage der Projektendkosten, obwohl die AC der Projekte zu Projektende nahe an dem BAC liegen.

Generell ist also festzustellen, dass die EVM-Methode zur Vorhersage der Projektendkosten eher durchschnittliche Ergebnisse erzielt. Zur Durchführung der EVM-Methode sollte dabei eine saubere Projektplanung gegeben sein, da sonst wie bei Projekt 5 keine Prognose möglich ist.

7 Vergleich der Modelle

Nachdem nun die Projektendkosten sowohl nach der Earned Value Management Methodik als auch mithilfe eines neuronalen Netzes prognostiziert wurden, widmet sich dieses Kapitel dem Vergleich beider Modelle nach den Kriterien Genauigkeit und Aufwand. Als Grundlage des Vergleichs werden die 5 Beispielprojekte aus Kapitel 6.2 herangezogen.

7.1 Nach Genauigkeit

Im ersten Schritt wird die Genauigkeit der beiden Methoden verglichen. Tabelle 7.1 liefert einen Überblick über die Modellgenauigkeit anhand der in Kapitel 6.2 vorgestellten Projekte, wobei EVM die Methode nach dem Earned Value Management und KNN das neuronale Netz darstellt.

	Projekt 1	**Projekt 2**	**Projekt 3**	**Projekt 4**	**Projekt 5**
AC	179.800,00 €	156.534,12 €	8.442,40 €	88.890,00 €	27.684,33 €
EAC_{EVM}	30.029,25 €	131.956,57 €	8.335,01 €	67.016,92 €	0 €
VAC_{EVM}	-1.979,25 €	24.577,66 €	-1.083,01 €	21.825,07 €	0 €
$\Delta\lvert EAC_{EVM} - AC\rvert$	**149.770,74 €**	**24.577,55 €**	**107,38 €**	**21.873,07 €**	**27.684,33 €**
EAC_{KNN}	119.764,94 €	165.760,66 €	8.608,89 €	89.091,28 €	27.654,85 €
VAC_{KNN}	-60.035,06 €	-9.226,42 €	-1.356,89 €	-249,28 €	-27.654,85 €
$\Delta\lvert EAC_{KNN} - AC\rvert$	**60.035,06 €**	**9.226,53 €**	**166,49 €**	**201,28 €**	**29,48 €**

Tabelle 7.1: Vergleich der EVM-Methode und des KI-Modells

Es ist zu erkennen, dass das neuronale Netz durchschnittlich bessere Vorhersagen als die EVM-Methode liefert. Projekt 4 und 5 werden sehr exakt von dem neuronalen Netz vorhergesagt, während die EVM-Berechnung eine Abweichung von ca. 20.000 Euro aufweist. Die Endkosten von Projekt 2 wurden durch das KNN eher durchschnittlich mit einer Abweichung von 9.226,53 Euro prognostiziert. Dennoch übertrifft die Vorhersage den EVM-Ansatz. Auch Projekt 1 wurde von dem neuronalen Netz genauer als von der EVM-Methode vorhergesagt. Trotzdem liegt auch hier eine eher große Abweichung von 60.035,06 Euro auf Seiten des neuronalen Netzes vor.

Das einzige der fünf Projekte, bei dem die Projektendkosten durch die EVM-Kennzahlen besser prognostiziert wurden, ist Projekt 3. Bei Projekt 3 liegen zwar die Vorhersagen beider Modelle sehr nahe an den tatsächlichen Endkosten, jedoch übertrifft das Modell nach EVM das neuronale Netz um ca. 60 Euro. Ursachen hierfür können sein, dass der Earned Value des Projektes den tatsächlichen Fertigungswert präzise dargestellt hat und das Projekt somit sauber gepflegt wurde.

Das neuronale Netz besitzt somit im Allgemeinen eine bessere Genauigkeit als die EVM-Methode. Diese erzielt bei sauber gepflegten Projekten zwar gute Ergebnisse, jedoch sind in der Realität die Projektdaten meist ungenau gepflegt. Beispielsweise weisen ca. 30% der Projekte aus der verwendeten Datenmenge einen fehlenden Planned Value auf. Dadurch ist eine Prognose nach EVM nicht mehr möglich. Hier zeigt das neuronale Netz seine Stärken, da

dieses durch Berücksichtigung mehrerer Projektdaten, selbst bei eher schlecht geplanten Projekten, in der Lage ist eine genaue Vorhersage zu treffen. Außerdem ist zu erwähnen, dass das neuronale Netz im Vergleich zur klassischen EVM-Methode keine Kenntnis über die Bestands- und Fakturadaten bzw. dem errechneten Earned Value hatte. Dennoch erzielt die KI ohne Berücksichtigung des Earned Value bessere Ergebnisse.

7.2 Nach Aufwand

Die Vorhersage der Projektendkosten geht bei beiden Modellen mit einem gewissen Aufwand und damit verbundenen Kosten einher. Um eine hohe Wirtschaftlichkeit zu gewährleisten, sollten deshalb beide Methoden bezüglich ihres Aufwands verglichen werden.

Zunächst wird die EVM-Methode untersucht. Hierbei entsteht der Hauptaufwand in der Berechnung der einzelnen EVM-Kennzahlen. Vor allem die Ermittlung des Earned Value kann zeitintensiv sein. Um diesen berechnen zu können, müssen Projekte eine saubere Planung aufweisen, was laut der in dieser Arbeit zugrunde liegenden Datenmenge häufig nicht der Fall ist (siehe Kapitel 6.1). Bei der EVM-Prognose ist der Aufwand außerdem projektbezogen. Das bedeutet, dass mit jedem zusätzlichen Projekt die Kennzahlen erneut ermittelt und berechnet werden müssen. Durch intelligente Excel-Tabellen können diese Berechnungen zwar weitestgehend automatisiert durchgeführt werden, jedoch stellt die Ermittlung der Kennzahlen dennoch einen erheblichen manuellen Aufwand dar.

Im Vergleich zur EVM-Methodik geht das neuronale Netz mit einem hohen initialen Aufwand für die Implementierung einher. Weiterhin müssen die Daten für den Machine Learning Algorithmus spezifisch aufbereitet werden, damit diese von dem Modell verarbeitet werden können. Kapitel 4 zeigt, dass eine solche Analyse und Aufbereitung der Datenbasis viel Zeit in Anspruch nehmen kann. Auch sind Experten im Bereich künstlicher Intelligenz notwendig um ein entsprechendes Machine Learning Modell zu erstellen. Dies kann zu zusätzlich anfallenden Kosten führen. Ein Vorteil des KI-Modells im Vergleich zu der EVM-Methode liegt darin, dass nach der Trainingsphase des Modells kein zusätzlicher projektspezifischer Implementierungs- oder Berechnungsaufwand zur Vorhersage der Projektendkosten notwendig ist.

Zusammenfassend ist zu sagen, dass die EVM-Methode für bereits gut gepflegte Projekte den geringeren Aufwand darstellt. Aufgrund des hohen initialen Aufwands des künstlichen neuronalen Netzes bietet sich diese Methode vor allem an, wenn für möglichst viele und unsauber gepflegte Projekte eine Vorhersage der Endkosten getroffen werden soll.

8 Fazit und Ausblick

Im Laufe dieser Arbeit wurde eine künstliche Intelligenz in Form eines neuronalen Netzes modelliert, um damit die Endkosten eines Projektes vorherzusagen. Hierfür sind projektbezogene Daten analysiert und aufbereitet worden. Diese aufbereiteten Daten dienten als Grundlage zur Erstellung und Validierung des Machine Learning Modells. Zum Testen der Vorhersagegenauigkeit wurden 5 Beispielprojekte ausgewählt, deren Projektendkosten zunächst nach der klassischen Earned Value Management Prognose geschätzt wurden. Die Ergebnisse der EVM-Methodik wurden daraufhin mit den Vorhersagen des trainierten KI-Modells verglichen. Dabei ist deutlich geworden, dass eine künstliche Intelligenz dazu in der Lage ist, selbst bei schlecht geplanten Projekten, durch Einbeziehung unterschiedlicher projektbezogener Faktoren, präzise Schätzungen zu berechnen. Die EVM-Prognose wurde hierbei in 4 von 5 Projekten von der KI übertroffen. Dies verdeutlicht die Stärke eines KI-Modells im Vergleich zu klassischen Projektcontrolling-Methodiken und zeigt, dass künstliche Intelligenz nützliche Anwendung im Projektcontrolling finden kann.

Die Arbeit stellt eine Grundlage für die Anwendung von künstlicher Intelligenz im Projektcontrolling dar. Über moderne Projektmanagement-Tools können nahezu unbegrenzt viele Projektdaten für den Einsatz von künstlicher Intelligenz genutzt werden. Durch die stetig wachsenden Datenmengen ist die künstliche Intelligenz außerdem in der Lage, sich durch einen kontinuierlichen Lernprozess weiterzuentwickeln.

In Kapitel 4.1 wurden die sechs Phasen des CRISP-DM-Modells behandelt. Dabei wurde erläutert, dass in dieser Arbeit Aspekte der Bereitstellungsphase des KI-Modells nicht behandelt werden, dennoch ist es denkbar solche Modelle als Entscheidungsunterstützung in bestehende Projektmanagement-Tools einzubinden. Ein intelligenter Assistent, in Form eines KI-Modells, eignet sich vor allem als Erweiterung zu den klassischen Projektcontrolling-Tools. Durch die Auswertung der bestehenden Projektdaten profitiert das Projektmanagement durch die allgemeine Mustererkennung und kann die Erkenntnisse in Relation mit klassischen Projektcontrolling-Verfahren zur Projektanalyse einsetzen.

Das im Rahmen dieser Arbeit erstellte Modell ist zudem weitestgehend unabhängig von zeitlichen Aspekten eines Projektes. Da Zeit als Faktor eine wesentliche Größe von Projekten ist, wäre es durchaus denkbar Zeitreihenanalysen in Bezug auf die Laufzeit und Kostenentwicklung eines Projektes zu untersuchen. Sogenannte rekurrente neuronale Netze — die eine bestimmte Art neuronaler Netze darstellen — eigenen sich besonders gut für zeitabhängige Modelle, da diese über eine Gedächtnisfunktion verfügen und dazu in der Lage sind Zeitreihen zu verarbeiten. Durch die Vorhersage des Projektverlaufs erhält der Projektmanager eine höhere Flexibilität und kann Steuerungsmaßnahmen präziser einsetzen.

Abschließend bleibt festzuhalten, dass künstliche Intelligenz einen deutlichen Mehrwert zur Unterstützung des Projektcontrolling liefern kann. Durch den Einsatz moderner Technologien kann dazu beigetragen werden, dass die Anzahl gescheiterter Projekte eines Unternehmens abnimmt.

Literaturverzeichnis

[Aichele 2006] Aichele, Christian: *Intelligentes Projektmanagement*. Stuttgart : Kohlhammer, 2006. – ISBN 3170190946

[Chapman u. a. 2000] Chapman, Pete ; Clinton, Julian ; Kerber, Randy ; Khabaza, Thomas ; Reinartz, Thomas ; Shearer, Colin ; Wirth, Rüdiger: *CRISP-DM 1.0: Step-by-step data mining guide*. 2000. – URL `https://www.the-modeling-agency.com/crisp-dm.pdf`. – Zugriffsdatum: 22.10.2019

[Chollet und Lorenzen 2018] Chollet, François ; Lorenzen, Knut: *Deep Learning mit Python und Keras: Das Praxis-Handbuch ; vom Entwickler der Keras-Bibliothek = Deep learning with Python*. 1. Auflage. Frechen and Ipswich, Massachusetts : mitp and EBSCO Industries, 2018 (Safari Tech Books Online). – URL `http://proquest.safaribooksonline.com/9783958458406`. – ISBN 3958458386

[Deru und Ndiaye 2019] Deru, Matthieu ; Ndiaye, Alassane: *Deep Learning mit TensorFlow, Keras und TensorFlow.js*. 1. Auflage. 2019 (Rheinwerk Computing). – ISBN 9783836265096

[Diedrich 2015] Diedrich, Oliver: *Google will das maschinelle Lernen voranbringen: TensorFlow, Google Software zum maschinellen Lernen, ist ab sofort als Open Source zur Implementierung eigener Anwendungen verfügbar*. 2015. – URL `https://www.heise.de/newsticker/meldung/Google-will-das-maschinelle-Lernen-voranbringen-2912530.html`. – Zugriffsdatum: 28.09.2019

[DIN 2009] DIN (Hrsg.): *DIN 69901-5: Projektmanagement - Projektmanagementsysteme*. Berlin : Beuth, 2009

[Ertel 2016] Ertel, Wolfgang: *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. 4., überarb. Aufl. 2016. Wiesbaden : Springer Fachmedien Wiesbaden, 2016 (Computational intelligence). – ISBN 9783658135492

[Fayyad u. a. 1996] Fayyad, Usama ; Piatetsky-Shapiro, Gregory ; Smyth, Padhraic: From Data Mining to Knowledge Discovery in Databases. In: *AI Magazine* 17 (1996), Nr. 3, S. 37. – URL `https://www.aaai.org/ojs/index.php/aimagazine/article/view/1230`

[Fraunhofer-Allianz Big Data 2017] Fraunhofer-Allianz Big Data: *Zukunftsmarkt Künstliche Intelligenz: Potenziale und Anwendungen*. 2017. – URL `https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/KI-Potenzialanalyse_2017.pdf`. – Zugriffsdatum: 28.09.2019

[García u. a. 2016] García, Salvador ; Luengo, Julián ; Herrera, Francisco: *Intelligent Systems Reference Library*. Bd. 72: *Data Preprocessing in Data Mining*. Softcover reprint of the original 1st ed. 2015. Cham : Springer International Publishing and Springer, 2016. – ISBN 3319377310

[Gartner 2019] GARTNER: *2019 CIO Survey: CIOs Have Awoken to the Importance of AI*. 2019. – URL https://www.gartner.com/en/newsroom/press-releases/2019-01-21-gartner-survey-shows-37-percent-of-organizations-have. – Zugriffsdatum: 21.09.2019

[Goodfellow u. a. 2018] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: *Deep Learning: Das umfassende Handbuch : Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze*. 1. Auflage. Frechen : mitp, 2018. – ISBN 9783958457027

[GPM 2013] GPM: *Misserfolgsfaktoren in der Projektarbeit*. 2013. – URL https://www.gpm-ipma.de/fileadmin/user_upload/GPM/Know-How/Misserfolgsfaktoren_final.pdf. – Zugriffsdatum: 12.11.2019

[Han u. a. 2012] HAN, Jiawei ; KAMBER, Micheline ; PEI, Jian: *Data mining: Concepts and techniques*. 3. ed. 2012 (The Morgan Kaufmann series in data management systems). – ISBN 9780123814791

[Hastie u. a. 2017] HASTIE, Trevor ; TIBSHIRANI, Robert ; FRIEDMAN, Jerome H.: *The elements of statistical learning: Data mining, inference, and prediction*. Second edition, corrected at 12th printing 2017. New York, NY : Springer, 2017 (Springer series in statistics). – ISBN 9780387848587

[Haykin 1999] HAYKIN, Simon S.: *Neural networks: A comprehensive foundation*. 2. ed., [Nachdr.], internat. ed. Upper Saddle River, NJ : Prentice Hall, 1999 (International edition). – ISBN 0139083855

[Hope u. a. 2018] HOPE, Tom ; RESHEFF, Yehezkel S. ; LIEDER, Itay: *Einführung in TensorFlow: Deep-Learning-Systeme programmieren, trainieren, skalieren und deployen*. 1. Auflage. Heidelberg : O'Reilly, 2018 (Safari Tech Books Online). – URL http://proquest.safaribooksonline.com/9781492067900. – ISBN 9783960090748

[Keßler und Winkelhofer 2004] KESSLER, Heinrich ; WINKELHOFER, Georg: *Projektmanagement: Leitfaden zur Steuerung und Führung von Projekten*. Vierte überarbeitete Auflage. Berlin, Heidelberg and s.l. : Springer Berlin Heidelberg, 2004. – URL http://dx.doi.org/10.1007/978-3-642-17025-6. – ISBN 9783642620843

[Kotu und Deshpande 2015] KOTU, Vijay ; DESHPANDE, Balachandre: *Predictive analytics and data mining: Concepts and practice with RapidMiner*. URL http://www.sciencedirect.com/science/book/9780128014608, 2015. – ISBN 9780128014608

[Kurbel 1992] KURBEL, Karl: *Entwicklung und Einsatz von Expertensystemen: Eine anwendungsorientierte Einführung in wissensbasierte Systeme*. 2., verb. Aufl. Berlin and Heidelberg and New York and London and Paris and Tokyo and Hong Kong and Barcelona and Budapest : Springer, 1992. – ISBN 3540552375

[Lämmel und Cleve 2012] LÄMMEL, Uwe ; CLEVE, Jürgen: *Künstliche Intelligenz*. 4., aktualisierte Auflage, neue Ausg. München : Hanser, Carl, 2012. – ISBN 3446428739

[scikit learn 2019] LEARN scikit: *Compare the effect of different scalers on data with outliers*. 2019. – URL https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html. – Zugriffsdatum: 17.11.2019

[Litke 2007] LITKE, Hans-Dieter: *Projektmanagement: Methoden, Techniken, Verhaltensweisen, evolutionäres Projektmanagement*. 5., erw. Aufl. München : Hanser, 2007. – ISBN 3446409971

[Möller und Dörrenberg 2010] Möller, Thor ; Dörrenberg, Florian: *Projektmanagement*. München u. a. : Oldenbourg, 2010 (WiSorium - Wirtschafts- und Sozialwissenschaftliches Repetitorium). – URL http://dx.doi.org/10.1524/9783486700602. – ISBN 978-3-486-70060-2

[Murphy 2012] Murphy, Kevin P.: *Machine learning: A probabilistic perspective*. 2012 (Adaptive computation and machine learning). – ISBN 9780262018029

[Nguyen und Zeigermann 2018] Nguyen, Chi N. ; Zeigermann, Oliver: *Machine Learning: Kurz & gut*. 1. Auflage. 2018 (O'Reillys Taschenbibliothek). – ISBN 3960090528

[Pflaumer u. a. 2017] Pflaumer, Peter ; Heine, Barbara ; Hartung, Joachim: *Statistik für Wirtschafts- und Sozialwissenschaften*. 4th ed. Berlin/Boston : De Gruyter, 2017 (Lehr- und Handbücher der Statistik). – URL https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4943311. – ISBN 9783486591163

[Piatetsky-Shapiro 2007] Piatetsky-Shapiro, Gregory: Data mining and knowledge discovery 1996 to 2005: overcoming the hype and moving from "university" to "business" and "analytics". In: *Data Mining and Knowledge Discovery* 15 (2007), Nr. 1, S. 99–105. – ISSN 1573-756X

[PMI 2019a] PMI: *AI Innovators: Cracking the Code on Project Performance*. 2019. – URL https://www.pmi.org/learning/thought-leadership/pulse/ai-innovators. – Zugriffsdatum: 29.09.2019

[PMI 2019b] PMI: *Pulse of the Profession 2019: 11th Global Project Management Survey*. 2019. – URL https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-all-comparison-reports-final.pdf?v=1f786811-3b88-443f-93c5-fc79188e6100. – Zugriffsdatum: 29.09.2019

[Reichel 2006] Reichel, Chance W.: *Earned value management systems (EVMS): "you too can do earned value management": Paper presented at PMI® Global Congress 2006*. 2006. – URL https://www.pmi.org/learning/library/earned-value-management-systems-analysis-8026. – Zugriffsdatum: 12.10.2019

[Rojas 1996] Rojas, Raúl: *Theorie der neuronalen Netze: Eine systematische Einführung*. 4., korrigierter Nachdr. Berlin : Springer, 1996 (Springer-Lehrbuch). – ISBN 9783540563532

[Runkler 2010] Runkler, Thomas A.: *Data Mining: Methoden und Algorithmen intelligenter Datenanalyse*. Wiesbaden : Vieweg + Teubner, 2010 (Computational intelligence). – URL http://dx.doi.org/10.1007/978-3-8348-9353-6. – ISBN 9783834808585

[Russell und Norvig 2010] Russell, Stuart J. ; Norvig, Peter: *Artificial intelligence: A modern approach*. 3. ed. Upper Saddle River, NJ : Prentice-Hall, 2010 (Prentice-Hall series in artificial intelligence). – ISBN 9780136042594

[Schwaiger und Steinwendner 2019] Schwaiger, Roland ; Steinwendner, Joachim: *Neuronale Netze programmieren mit Python*. 1. Auflage. Bonn : Rheinwerk Computing, 2019. – ISBN 3836261421

[Tensorflow 2019] Tensorflow: *TensorFlow tensors*. 2019. – URL https://www.tensorflow.org/guide/tensor. – Zugriffsdatum: 03.11.2019

[The Standish Group International 2015] The Standish Group International: *CHAOS REPORT 2015*. 2015. – URL https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf. – Zugriffsdatum: 12.11.2019

[Turing 1950] Turing, Alan M.: Computing Machinery and Intelligence. In: *Mind* 59 (1950), Nr. October, S. 433–460

[VanderPlas 2018] VanderPlas, Jake: *Data Science mit Python: Das Handbuch für den Einsatz von IPython, Jupyter, NumPy, Pandas, Matplotlib, Scikit-Learn*. 1. Auflage. Frechen : mitp, 2018. – ISBN 3958456952

[Wanner 2013a] Wanner, Roland: *Earned-value-Management: So machen Sie Ihr Projektcontrolling noch effektiver ; [Projektmanagement für Profis*. 3. Aufl. Leipzig : Amazon Distribution GmbH, 2013. – ISBN 9781484050965

[Wanner 2013b] Wanner, Roland: *Projektcontrolling: Projekte erfolgreich planen, überwachen und steuern*. 1. Aufl. Leipzig : Amazon Distribution GmbH, 2013. – ISBN 9781479142552

[Weber 2019] Weber, Jürgen: *Controlling: Definition*. 2019. – URL https://wirtschaftslexikon.gabler.de/definition/controlling-30235/version-370809. – Zugriffsdatum: 05.10.2019

[Wikipedia 2019] Wikipedia: *Cross-industry standard process for data mining*. 2019. – URL https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining. – Zugriffsdatum: 29.11.2019

[Zirkler u. a. 2019] Zirkler, Bernd ; Nobach, Kai ; Hofmann, Jonathan ; Behrens, Sabrina: *Projektcontrolling: Leitfaden für die betriebliche Praxis*. Wiesbaden : Springer Fachmedien Wiesbaden, 2019. – URL http://dx.doi.org/10.1007/978-3-658-23714-1. – ISBN 9783658237134

Anhang

Der Anhang dieser Arbeit wird digital auf dem beiliegenden USB-Stick bereitgestellt. Dieser umfasst sechs Python Jupyter Notebooks, die als PDF exportiert wurden. Die Jupyter Notebooks beinhalten Python-Programmcode und dessen zugehörige Konsolenausgabe.

Nachfolgend werden die einzelnen PDF-Dateien des Anhangs genannt und dessen Inhalt erklärt:

1. Kostentabelle (Analyse und Aufbereitung)
Python Jupyter Notebook zur Analyse und Aufbereitung der Kostentabelle.

2. Bestandstabelle (Analyse und Aufbereitung)
Python Jupyter Notebook zur Analyse und Aufbereitung der Bestandstabelle.

3. Fakturatabelle (Analyse und Aufbereitung)
Python Jupyter Notebook zur Analyse und Aufbereitung der Fakturatabelle.

4. Feature Engineering / Datentransformation für das neuronale Netz
Python Jupyter Notebook zur Erstellung der Merkmale für das neuronale Netz. Beinhaltet die Transformation und Bereinigung der Daten (siehe Kapitel 4.4).

5. KNN-Prognose
Python Jupyter Notebook zur Erstellung des künstlichen neuronalen Netzes. Dieses beinhaltet die Normalisierung der Daten (siehe Kapitel 4.4.4) und die Implementierung des KI-Modells zuzüglich dessen Analyse (siehe Kapitel 5.2). Außerdem werden die Projektendkosten für die 5 Beispielprojekte aus Kapitel 6.2 vorhergesagt, um die Ergebnisse in Kapitel 7 mit der EVM-Methode vergleichen zu können.

6. EVM-Prognose
Python Jupyter Notebook zur Berechnung der EVM-Kennzahlen anhand der 5 Beispielprojekte aus Kapitel 6.2 (siehe Kapitel 6).

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____