ML-basierte Prädiktion von Reisezielen auf Basis multivariater Parameter

Konzeption und Entwicklung

Fachhochschule Dortmund
Fachbereich Informatik

Thesis im Studiengang Wirtschaftsinformatik
zur Erlangung des Grades Master of Science (M. Sc)

Marius Golombeck
geboren am 19.02.1992

Betreuer: Dr. habil. Dipl.-Inf. Prof. Andreas Georg Harrer
Dortmund, 01.11.2020

DOI: 10.13140/RG.2.2.23262.25923

Danksagung

Ich bedanke mich bei Dr. habil. Dipl.-Inf. Prof. Andreas Georg Harrer für die Möglichkeit zur Anfertigung und die Betreuung dieser Arbeit. Weiterhin bedanke ich mich bei Robin Mundhenk, Henning Orlowski und Stefan Tübben für den hilfreichen Input bei der Konzeption, Entwicklung, Verschriftlichung und Korrektur dieser Arbeit. Außerdem danke ich meinem Arbeitgeber, der Dr. Ing. h.c. F. Porsche AG, für den Freiraum zur Erstellung. Abschließend danke ich meinen Eltern, die mir stets uneingeschränkt zur Seite stehen und mir dieses Studium erst ermöglicht haben.

Markenrechtlicher Hinweis

Die in dieser Thesis wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenzeichen usw. können auch ohne besondere Kennzeichnung geschützte Marken sein und als solche den gesetzlichen Bestimmungen unterliegen. Sämtliche in dieser Thesis abgedruckten Bildschirmabzüge unterliegen dem Urheberrecht des jeweiligen Herstellers.

Kurzfassung

Bei der Auswahl eines Reiseziels müssen stets grundlegende Entscheidungen getroffen werden. Welche Destinationen sind überhaupt interessant? Welche Aktivitäten werden beabsichtigt? Insbesondere einem unschlüssigen Nutzer kann eine präzise Formulierung dieser Randbedingungen schwerfallen. Wie kann dennoch die Findung eines relevanten Reiseziels erfolgen und wie kann ein Nutzer bei seiner Entscheidungsfindung maschinell unterstützt werden? Zur Lösung dieses Problems wird innerhalb dieser Thesis ein dualer Bildvergleich erdacht, in dem einem Nutzer eine Reihe generischer Reisebilder präsentiert werden. Diese Bilder sind mittels eines CNN auf eine Reihe möglicher Reisekategorien klassifiziert, sodass ein Nutzer sich bei der Bildauswahl implizit entscheidet, ob er z.B. an Sightseeing-Reisen oder an Wellness-Reisen interessiert ist, bzw. welche Reiseziele für ihn interessant sind. Die Reiseziele selbst werden auf Basis von Informationen zu POIs ausgewertet, wodurch auf die Ausprägung und Verteilung möglicher Reiseaktivitäten geschlossen werden kann. Die Nutzereingaben resultieren in einem Interessensprofil, welches möglichen Reisezielen gegenübergestellt wird. Auf Basis der so berechneten Übereinstimmungsquote werden dem Nutzer Reiseziele vorgeschlagen. Die Nutzerakzeptanz der Ergebnisse wird mit einer Zustimmungsquote von 70,37% gemessen.

Abstract

When choosing a travel destination, fundamental decisions must always be made. For example, which destinations are interesting? Which activities might be interesting to do? Especially an indecisive user may find it difficult to formulate these conditions precisely. How can a relevant travel destination still be found and how can a user be supported in his decision-making process by use of computerized means? To solve this problem, this thesis devises a dual image comparison, in which a user is presented with a series of generic travel images. These images are classified by CNN into several possible travel categories, which leads to a user implicitly deciding whether e.g. he is interested in sightseeing tours or wellness trips, or which destinations are of interest to him in general. The destinations themselves are evaluated based on information of POIs, which allow conclusions to be drawn about the characteristics and distribution of possible travel activities. The user input thus creates an interest profile, which is compared with possible destinations. Based on the calculated match rate, destinations are suggested to the user. The user approval rate of the displayed results is measured at 70.37%.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence (siehe KI)
API	Application Programming Interface
BCU	Binary Cross Entropy
bspw.	beispielsweise
bzgl.	bezüglich
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRISP-DM	Cross-industry standard process for data mining
CSP	Constraint Satisfaction Problem
CSV	Comma Separated Values
DV	Datenverarbeitung
DW	Data Warehouse
EVA	Eingabe-Verarbeitung-Ausgabe
GPU	Graphics Processing Unit
i.d.R.	in der Regel
insb.	insbesondere
IT	Informationstechnologie
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
(K)NN	(Künstliches) Neuronales Netz
ML	Machine Learning
o.g.	oben genannt(en)
POI	Point of Interest
ReLU	Rectified Linear Unit
RGB	Rot, grün, blau
u.U	unter Umständen
z.T.	zum Teil

1 Einleitung

1.1 Problemstellung

Eine Reiseplanung umfasst einen grundsätzlichen Planungsaufwand. Reisedatum und -ziel müssen spezifiziert, Unterkünfte und Transportmöglichkeiten eruiert, ein Budget bekannt und bspw. Wetterdaten geprüft werden, um eine möglichst erfolgreiche Urlaubsplanung vorzunehmen. Hierbei muss mindestens ein latentes Wissen über relevante Reiseziele vorhanden sein. Reiseanbieter bieten auf ihren digitalen Reiseplattformen (Bspw. Ab-in-den-Urlaub, TUI, Expedia, etc.) detaillierte Möglichkeiten zur Spezifikation einer Reise. Der Auswahlprozess erfolgt zumeist durch einen Filteralgorithmus, welcher bei fehlender Präzisierung eine erhebliche Anzahl an Ergebnissen zurückliefert, bspw. wenn ein Nutzer sein Reiseziel nicht näher einschränkt. Dies stellt für den Nutzer eine nicht effizient verarbeitbare Informationsmenge dar. Erst durch genaues spezifizieren von Reisedetails wird die Informationsmenge für den Nutzer auf eine verarbeitbare Menge reduziert und ihm eine Auswahl an, für ihn vermeintlich geeigneten, Reisezielen präsentiert.

Die Fragestellung, mit der sich dieser Problemstellung genährt wird, lautet daher: Kann die Findung von relevanten Reisezielen durch die in dieser Thesis angewandte Vorgehensweise mittels Machine Learning Verfahren vereinfacht werden?

1.2 Zielsetzung

Das Problem der Informationsüberfrachtung des Nutzers soll durch eine visuelle Auswahl von generischen Bildern umgangen werden. Diese werden auf Basis von Reisezielattributen ausgewählt (z.B. Strand, Mountainbike, See, Museum, Konzert, etc.). Der Nutzer soll eine Reihe dieser Bilder miteinander vergleichen und eine Auswahl treffen. Durch seine Auswahl soll ein Rückschluss auf seine Reiseinteressen getroffen werden, um ihm anschließend mögliche Reiseziele vorzuschlagen.

Im Rahmen dieser Thesis wird hierzu eine Machine Learning gestützte Anwendung zur Prädiktion von Reisezielen konzipiert und entwickelt. Zuerst werden Informationen über Reiseziele gesammelt und mittels Data Science Methoden verarbeitet. Mit Hilfe eines neuronalen Netzes werden anschließend generische Reisebilder klassifiziert. Diese werden dem Nutzer in einer Reihe von dualen Bildvergleichen präsentiert. Diese Auswahl wird interpretiert, um dem Nutzer als Ergebnis des Verfahrens relevante Reiseziele vorzuschlagen. Durch Details über die Auswahlergebnisse, wie z.B. die Übereinstimmungsquote seiner Auswahl mit möglichen Reisezielen ist der Nutzer in der Lage, einen Vergleich zu seinen Erwartungen zu ziehen und somit ein geeignetes Reiseziel abzuwägen.

1.3 Vorgehensweise

Die Thesis gliedert sich in ein drei Hauptkapitel, einen Grundlagenkapitel, den praktischen Entwicklungsteil und die anschließende Evaluation. Im Rahmen der Grundlagen werden in Kapitel 2 alle für das Verständnis der Entwicklung relevanten Themen eingeführt. Darüber hinaus wird das Konzept der Constraint Satisfaction Problems (CSP) thematisiert und auf allgemeiner (Kapitel 2.4), wie auf anwendungsbezogener (Kapitel 4.2) Ebene als alternative Entwicklungsmethodik bewertet. Die im Rahmen dieser Thesis entwickelte Software wird in Kapitel 3 dokumentiert. Insbesondere die Kernschritte der Entwicklung (Kapitel 3.4) werden ausführlich thematisiert. Abschließend erfolgt in Kapitel 4 eine Evaluation des Entwicklungsprozesses sowie die Durchführung eines Nutzertests und dessen Auswertung.

2 Grundlagen

Im Rahmen des Grundlagenkapitels werden grundlegende Konzepte, die in dieser Arbeit verwendet werden, vorgestellt. Hierzu gehören Methoden der Data Science (2.1), Grundlagen der Künstliche Intelligenz (2.2), Machine Learning (2.3) (insb. überwachtes Lernen (2.3.1) und Deep Learning (2.3.2)) sowie Grundlagen zu Constraint Satisfaction Problems (2.4).

2.1 Data Science

Das Fachgebiet der Data Science kombiniert domänenspezifische Kenntnisse aus der Informatik, Mathematik und Statistik sowie Wissen aus angrenzenden Bereichen der Informationstechnik (IT), mit dem Ziel Informationen aus Daten zu gewinnen [1–4]. Bratschler, Stadelmann und Stockinger definieren den Begriff Data Science als eine einzigartige Mischung aus Prinzipien und Methoden aus Analyse, Technik, Unternehmertum und Kommunikation. Dieser verfolge das Ziel, aus Daten einen wirtschaftlich nutzbaren Zweck zu ziehen [5]. Data Science wird i.d.R. als ein Konzept zur Bewältigung großer Datenmengen verwendet [4]. Hierzu kombinieren Data Scientists eine Reihe verschiedenartiger Fähigkeiten zur Datenverarbeitung [6]. Die Daten stammen aus unterschiedlichen Quellen und liegen zumeist in heterogenen Formaten vor [7]. Die Tätigkeiten können z.B. Datenakquise, -bereinigung, -aufbereitung, -analyse und -visualisierung umfassen [8]. Ziel ist es, kritische Informationen aus den Datensätzen zu extrahieren [8]. Das so erlangte Datenverständnis kann aus wirtschaftlicher Sicht genaue Vorhersagen und Einsichten ermöglichen, welche anschließend für Geschäftsentscheidungen verwendet werden können [8]. Während Data Science ein weitläufiges Fachgebiet beschreibt, stellt Data Mining die operative Technik dazu [5].

2.1.1 Data Mining

Ein Data Mining Prozess enthält problemunabhängige Schritte von der Problemformulierung bis zur Problemlösung [9]. Data Mining kann definiert werden als der Prozess, Informationen aus Datenbanken zu sammeln, die zuvor unverständlich und unbekannt waren, und diese Informationen dann (für relevante Geschäftsentscheidungen) nutzbar zu machen [Vgl. 10].

Ein allgemein anerkannter und weit verbreiteter Ansatz zur Abbildung generischer Data Mining Prozessschritte ist das Prozessmodell CRISP-DM [11]. Entlang dieses Prozessmodells werden verschiedenen Tätigkeiten zur Datenmanipulation in einem Data Mining Prozess erläutert.

2.1.1.1 CRISP-DM

CRISP-DM ist 1996 das Ergebnis der gemeinschaftlichen Entwicklung von sieben Mitarbeitern[1] der Unternehmen DaimlerChrysler[2], SPSS[3] und NCR [12]. Die drei Unternehmen engagieren sich bereits in verschiedener Art und Weise auf dem noch jungen und unstrukturierten Data Mining Markt [11]. Während DaimlerChrysler insbesondere für die praktische Anwendung von Data Mining in Geschäftsoperationen bekannt ist, bieten SPSS und NCR bereits frühe Data Mining bzw. Data Warehouse[4] Lösungen an [11, 13]. Zu dieser Zeit sorgt ein stetig steigendes Marktinteresse an Data Mining Dienstleistungen für die Idee zur Entwicklung eines nicht proprietären Standardprozessmodells für den Einsatz in der Industrie [12].

---

1 Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR), Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) und Rüdiger Wirth (DaimlerChrysler).

2 Damals Daimler-Benz.

3 Damals ISL.

4 Siehe Glossar.

CRISP-DM beinhaltet sechs Prozessphasen, die auf Basis der jeweils gewonnenen Erkenntnisse iterativ und z.T. richtungsungebunden durchlaufen werden (siehe Abbildung 1) [12].

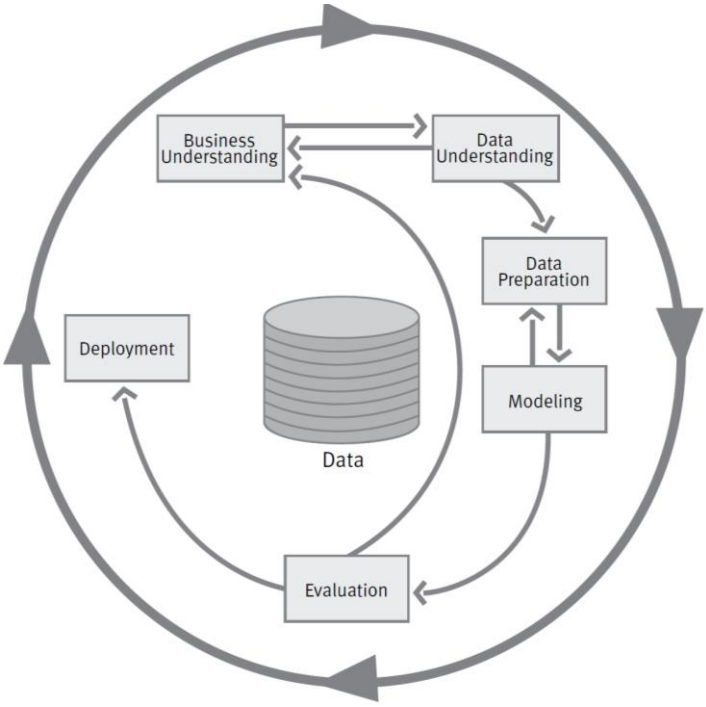


Abbildung 1 - CRISP-DM Prozessmodell [12]

Das Ergebnis der jeweiligen Phasen bestimmt den Startpunkt für die darauffolgende Phase [12]. Der äußere Kreislauf zeigt die kontinuierliche Natur eines Data Mining Projekts, in dem eine Wiederholung der Prozessschritte zur Optimierung der Ergebnisse durchgeführt wird [11].

Die Prozessschritte enthalten je nach Aufgabestellungen auf wirtschaftlicher, prozessualer, funktionaler und/oder technischer Ebene, sowie den damit verbundenen Daten, unterschiedliche Inhalte [12]. Je nach Granularität und Definition des Problemraums unterscheiden sich diese Schritte strukturell nur geringfügig, sodass der Data Science Prozess auf Basis von CRISP-DM generalisiert werden kann [11].

In Abbildung 2 sind die generischen Aufgaben (fett) und die Ergebnisse (kursiv) der CRISP-DM Prozessschritte dargestellt. Die in der praktischen Entwicklung im Rahmen dieser Thesis nicht realisierten Schritte sind transparent dargestellt.

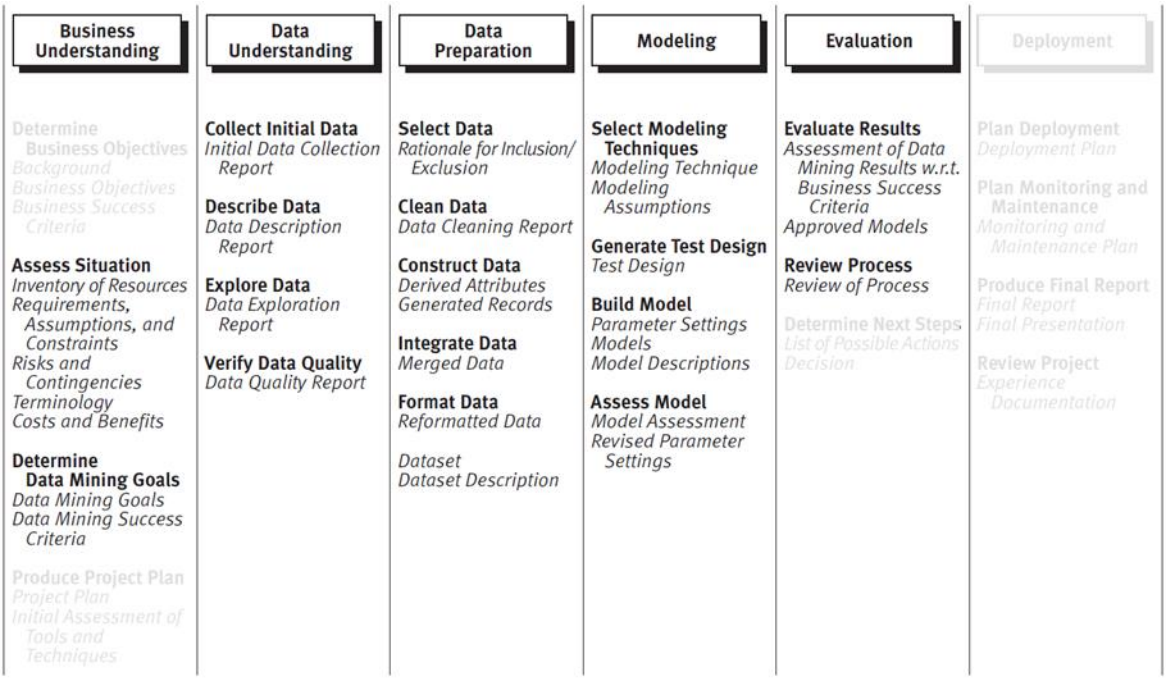


Abbildung 2 - Aufgaben und Ergebnisse der sechs CRISP-DM Prozessschritte [12]

Allgemein umfasst CRISP-DM sechs Prozessschritte, die entlang von Abbildung 1 und Abbildung 2 in den folgenden Kapiteln 2.1.1.1.1 bis 2.1.1.1.6 erläutert werden. Die Entwicklung und ihre Dokumentation orientieren sich an CRISP-DM.

2.1.1.1.1 Business Understanding

In der initialen Prozessphase gilt es die Projektziele und -anforderungen aus wirtschaftlicher Sicht zu bestimmen und zu verstehen. Das erlangte Wissen ist Grundlage für die Definition und Abgrenzung eines Data Mining Problems sowie für die Erstellung von vorbereitenden Maßnahmen zur Zielerreichung [12].

Um zu verstehen, welche Daten im Anschluss wie analysiert werden sollen, ist es für den Anwender unerlässlich, den geschäftlichen Hintergrund der Operation vollständig zu verstehen [11]. Die Phase des Geschäftsverständnisses umfasst mehrere Schritte, einschließlich der Festlegung von Geschäftszielen, Beurteilung der Situation, Festlegung der Data-Mining-Ziele und der Erstellung eines Projektplans [11].

2.1.1.1.2 Data Understanding

Die Data Understanding Phase beginnt mit einer Datenerhebung und/oder -akquisition. Durch einen explorative Analyse soll ein erstes Verständnis über die Daten geschaffen werden [11]. Weiterhin können so bereits früh Datenqualitätsprobleme identifiziert werden [11]. Die Erkenntnisse des Data Understanding hinsichtlich der benötigten Datenmanipulation fließen unmittelbar in die darauffolgende Phase ein [11].

2.1.1.1.3 Data Preparation

Die Phase der Data Preparation umfasst alle Aktivitäten zur Erstellung des endgültigen Datensatzes aus den anfänglichen Rohdaten [11]. Dies erfolgt auf Basis der Ergebnisse vorangegangener Prozessschritte des Business- und Data Understanding. Die Qualität vorangegangener Schritte ist essenziell für eine effiziente Verarbeitung der Daten [11].

2.1.1.1.4 Modeling

Im Rahmen des Modeling werden Modellierungstechniken ausgewählt und angewandt. Je nach Komplexität des Problems können bestimmte Techniken (z.B. neuronale Netze) besondere Anforderungen an die Form der Daten stellen [12]. Dies erfordert ggf. einen Rückschritt zur Datenvorbereitung. Modeling bedeutet hierbei nicht zwingend einen Schritt hin zu einer komplexen Machine Learning Anwendung, sondern kann auch die Erstellung verhältnismäßig simpler statistischer Analysen bedeuten[5] [11, 12].

2.1.1.1.5 Evaluation

Sobald ein oder mehrere Modelle erstellt sind, werden diese mit Hilfe „ungesehener" Daten[6] getestet, um sicherzustellen, dass sie eine

[5] z.B. Datengetriebene Auswertungsprozesse der Business Intelligence [13].

[6] Daten, welche nicht zur Modellentwicklung verwendet wurden.

Generalisierungsfähigkeit[7] besitzen [12]. Außerdem soll überprüft werden, dass alle wichtigen Geschäftsaspekte ausreichend berücksichtigt sind [12]. Das Endergebnis ist die Auswahl des Siegermodells bzw. ein Rückschritt zur weiteren Überarbeitung des Problems oder Neudefinition des Problemraums [11, 12].

2.1.1.1.6 Deployment

In der Deploymentphase wird eine Programmcode-Repräsentation des Modells implementiert, um neue, ungesehene Daten zu bewerten oder zu kategorisieren, sobald sie entstehen [12]. So soll ein Mechanismus für die Verwendung neuer Informationen zur Lösung des ursprünglichen Geschäftsproblems geschaffen werden [12]. Hierbei ist essenziell, dass die Code-Darstellung alle Schritte vor der Modellierung umfassen muss, sodass das Modell neue Rohdaten in der gleichen Weise behandelt, wie während der Modellentwicklung [11, 12].

2.2 Künstliche Intelligenz

In diesem Kapitel werden Grundlagen der künstlichen Intelligenz (KI) samt Begriffsdefinition und Funktionsweise eingeführt. Diese Grundlagen dienen dem besseren Verständnis der anschließenden Themengebiete Machine Learning (inkl. überwachtem Lernen) und Deep Learning.

In der deutschen Sprache existiert keine allgemeinsprachliche Begriffsdefinition. Eine englischsprachige Definition aus dem allgemeinen Sprachgebrauch ist jedoch im Oxford Dictionary aufgeführt [14]:

> *„THE THEORY AND DEVELOPMENT OF COMPUTER SYSTEMS ABLE TO PERFORM TASKS NORMALLY REQUIRING HUMAN INTELLIGENCE […]."*

Im Vergleich dazu, findet sich auch eine fachliche Definition im Dictionary of Computing [15]:

[7] Siehe Glossar: Generalisierung.

> *„THE DESIGN AND DEVELOPMENT OF COMPUTER PROGRAMS THAT ATTEMPT TO IMITATE HUMAN INTELLIGENCE AND DECISION-MAKING FUNCTIONS, PROVIDING BASIC REASONING AND OTHER HUMAN CHARACTERISTICS."*

Die Kombination der wesentlichen Elemente aus den o.g. englischsprachigen fachlichen und allgemeinsprachlichen ermöglicht die Bildung der folgenden deutschsprachigen Definition für künstliche Intelligenz:

> *„[KÜNSTLICHE INTELLIGENZ IST] DIE FÄHIGKEIT EINER MASCHINE / EINES COMPUTERPROGRAMMS, ABSTRAKTE UND VOR DEM HINTERGRUND EINES NATÜRLICHEN VORBILDS, INTELLIGENTE, EIGENSTÄNDIGE ENTSCHEIDUNGEN ZU TREFFEN UND DARAUS (ZWECKVOLLES) HANDELN ABZULEITEN."*

Zusätzlich kann Künstliche Intelligenz als Teilgebiet der Informatik beschrieben werden [15, 16]:

> *„[KÜNSTLICHE INTELLIGENZ IST] EIN TEILGEBIET DER INFORMATIK UND DES INGENIEURWESENS, WELCHES SICH MIT DEM (RECHNERGESTÜTZTEN) VERSTÄNDNIS VON INTELLIGENTEM (MASCHINEN-) VERHALTEN SOWIE DER ENTWICKLUNG VON DEMENTSPRECHENDEN ANWENDUNGEN AUSEINANDERSETZT."*

Die Funktionsweise einer künstlichen Intelligenz basiert grundsätzlich auf dem EVA-Prinzip[8] der Informationsverarbeitung mit seinen drei Schritten: Eingabe, Verarbeitung und Ausgabe [17]. Diese Schritte können für eine künstliche Intelligenz als die Kernfähigkeiten Sense, Comprehend und Act präziser beschrieben werden (siehe Abbildung 3).

[8] Siehe Glossar.

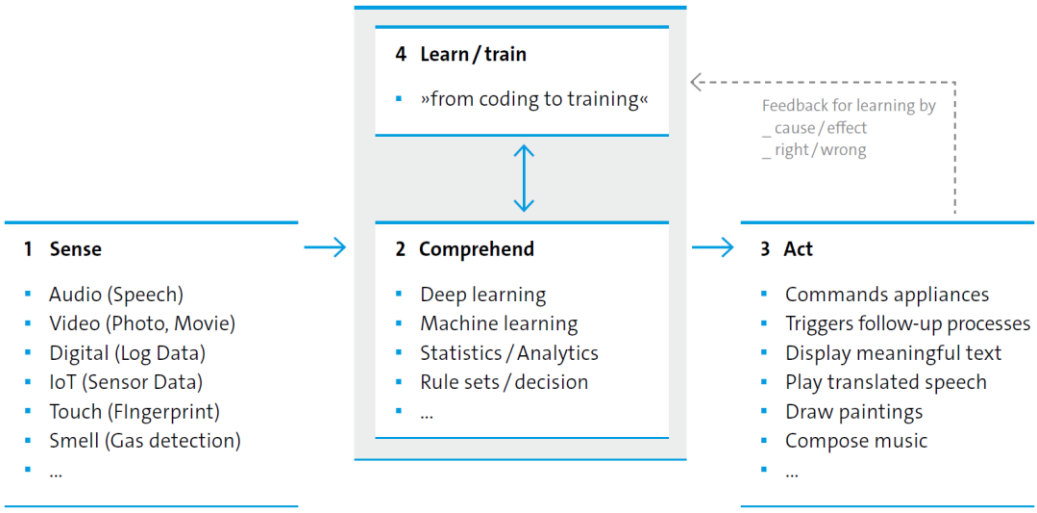


Abbildung 3 - Allgemeine Funktionsweise einer künstlichen Intelligenz [18]

Im ersten Schritt (Sense) erhält eine KI z.B. Audio-, Video-, oder Sensordaten. Diese werden im zweiten Schritt (Comprehend) mit Methoden der künstlichen Intelligenz, wie z.B. Machine Learning oder statistischen Verfahren verarbeitet [18]. Anschließend wird aus diesen Methoden eine Handlung (Act) abgeleitet. Diesem dritten Schritt schließt sich der iterative Rückschritt des Lernens (Learn / train) an, wodurch bereits erzielte Ergebnisse aus Schritt 3 in Schritt 2 zurückliefert werden. Lernfähigkeit ist ein grundsätzliches Merkmal von Systemen der künstlichen Intelligenz [19]. Die so erzielten Ergebnisse sind häufig besser wie beim Einsatz herkömmlicher Verfahren (siehe EVA-Prinzip), welche im Wesentlichen auf starren, klar definierten und fest programmierten Regelwerken basieren [Vgl. 19].

Im folgenden Kapitel wird Machine Learning als Teilmenge künstlicher Intelligenz vorgestellt und Kernfähigkeiten konkretisiert.

2.3 Machine Learning

Maschinelles Lernen ist eine Teilmenge der künstlichen Intelligenz, bei der Computeralgorithmen verwendet werden, um autonom aus Daten und Informationen zu lernen [17]. Beim maschinellen Lernen sind Computerprogramme in der Lage ihre Algorithmen selbstständig zu verändern und zu verbessern [20]. Der Begriff Machine Learning wird im Jahr 1952, Jahrzehnte vor der eigentlichen

ML-Hochphase durch den IBM-Wissenschaftler Arthur Samuel geprägt [21]. Er benutzt das Spiel Dame, um ein erstes selbstlernendes Programm zu erstellen. Samuel lässt das Computerprogramm gegen sich selbst antreten und wertet dabei gewinnbringende Spielzüge aus. Diese lässt er wiederum in die Spielstrategie einfließen und optimiert sie dadurch. Arthur Samuel gilt als der ursprüngliche Definitionsgeber des Begriffs Machine Learning. Seine dem Wortlaut entnommene Definition lautet [22, 23]:

> *„[MACHINE LEARNING IS THE] FIELD OF STUDY THAT GIVES COMPUTERS THE ABILITY TO LEARN WITHOUT BEING EXPLICITLY PROGRAMMED."*

Maschinelles Lernen ist also laut Samuel eine Computeranwendung, die automatisch aus Erfahrungen lernt, um sich dadurch zu verbessern, ohne explizit dafür programmiert zu sein.

Machine Learning kann wie folgt kategorisiert werden:

- Überwachtes Lernen (Supervised Machine Learning)
- *Unüberwachtes Lernen (Unsupervised Machine Learning)*
- *Halbüberwachtes Lernen (Semi-supervised Machine Learning)*
- *Verstärkungslernen (Reinforcement Machine Learning)*

Auf die Kategorie des überwachten Lernens wird im den folgenden Kapitel 2.3.1 näher eingegangen. Erläuterungen zu unüberwachtem Lernen, halbüberwachtem Lernen und Verstärkungslernen befinden sich im Glossar. In Kapitel 2.3.2 wird das Themengebiet des Deep Learning eingeführt. Hier wird speziell auf künstliche neuronale Netze zur Bildverarbeitung (Kapitel 2.3.2.1) sowie auf Convolutional Neuronal Networks (CNN) (Kapitel 2.3.2.1.1) eingegangen.

2.3.1 Überwachtes Lernen (Supervised Learning)

Überwachtes Lernen beschreibt ein maschinelles Lernkonzept zur Entwicklung eines Modells, bei dem anhand von bekannten Datensätzen Vorhersagen für unbekannte Daten getroffen werden (siehe Abbildung 4) [24].

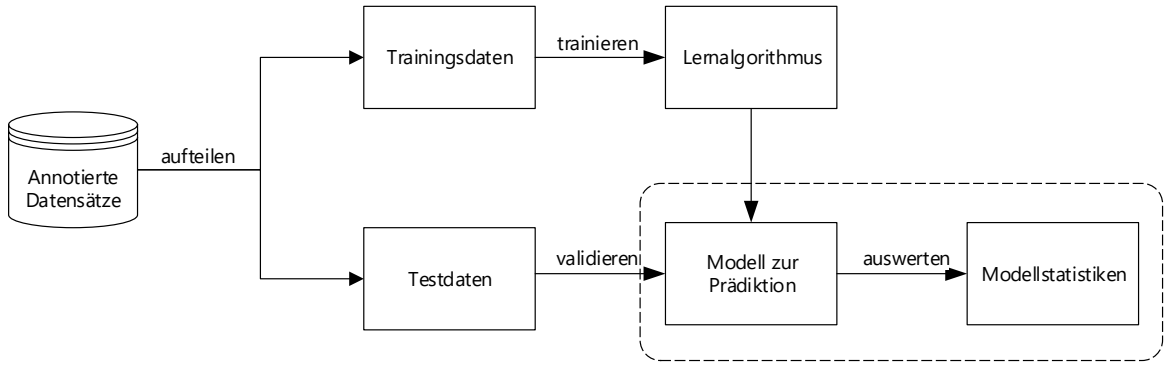


Abbildung 4 - Funktionsweise überwachten Lernens [adaptiert entnommen aus 25]

Hierzu benötigt ein überwachter Lernalgorithmus annotierte Eingabedaten, die eine Zielspalte mit Klassifizierern (z.B. "krank" bzw. "gesund") enthalten, auf die geschlossen werden soll [26, 27]. Mittels der sog. Trainingsdaten[9] und Testdaten[10] wird das Modell auf eine gewünschte Interpretation (und die damit verbundene Ausgabe) trainiert [28]. Das Ziel ist die Vorhersage unbekannter Daten [27, 28].

Überwachte Lernprobleme lassen sich in Klassifikation und Regression einteilen, welche in den folgenden Kapiteln 2.3.1.1 und 2.3.1.2 beschrieben werden [27].

2.3.1.1 Klassifikation

Klassifikation im Sinne von überwachten Lernalgorithmen bedeutet die Abbildung von Inputdaten auf bestimmte Klassen [17]. Ein Klassifikationsalgorithmus liefert diskrete Werte (wie z.B. „krank" bzw. „gesund"). Klassifikation lässt sich in binäre Klassifikation (binary classification) und mehrklassige Klassifikation (multi-class classification) unterteilen, auf die wiederum lineare bzw. nicht-lineare Verfahren angewendet werden können [27].

Eine **binäre Klassifikation** unterscheidet zwischen zwei Klassen. Sie findet einen praktischen Nutzen häufig in binären Problemen [24]. Ein Beispiel hierfür ist die Beantwortung der Ja/Nein-Frage: „Handelt es sich bei dieser E-Mail um Spam?". Hier werden E-Mails in die binären Kategorien „Spam" bzw. „Nicht-Spam"

[9] Siehe Glossar.

[10] Siehe Glossar.

klassifiziert [27]. Abbildung 5 zeigt visualisiert ein einfaches Beispiel binärer Klassen. Hierzu sind Attribut der Klassen 0 (rot) und 1 (blau) in einem Koordinatensystem abgebildet. Diese Werte können Rohdaten ebenso wie errechnete oder normalisierte Parameter repräsentieren, welche wiederum auf eine ursprüngliche Verteilung abgebildet werden können.

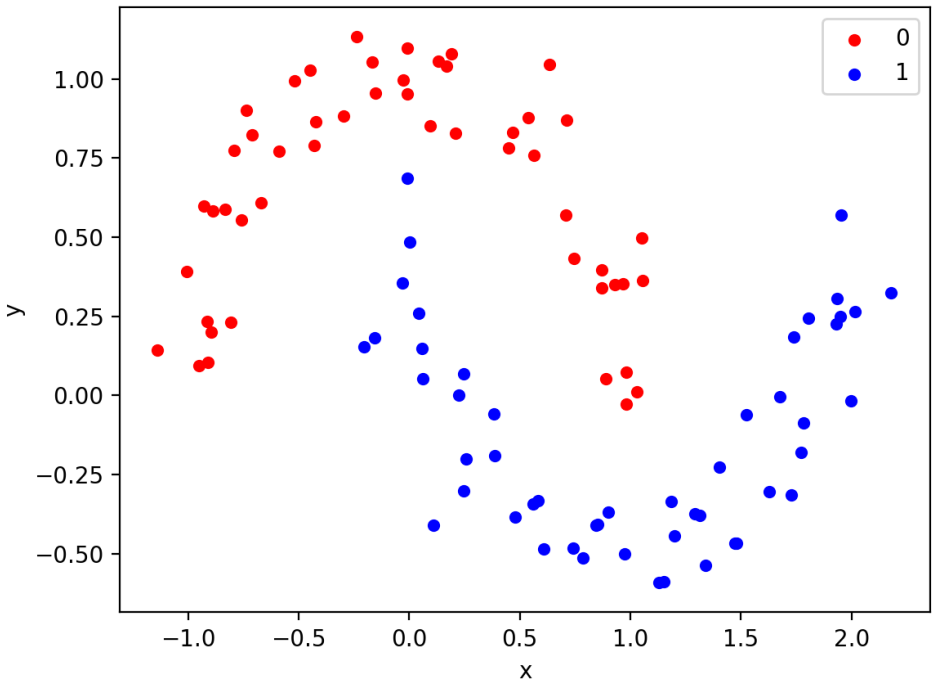


Abbildung 5 - Visualisierung von Werten der zwei Klassen 0 (rot) und 1 (blau)

Wird auf das Beispiel aus Abbildung 5 eine **lineare Klassifikation** angewendet, werden die Klassen 0 (rot) und 1 (blau) durch eine oder mehrere lineare Klassifikationsgrenzen getrennt bzw. klassifiziert (siehe Abbildung 6).

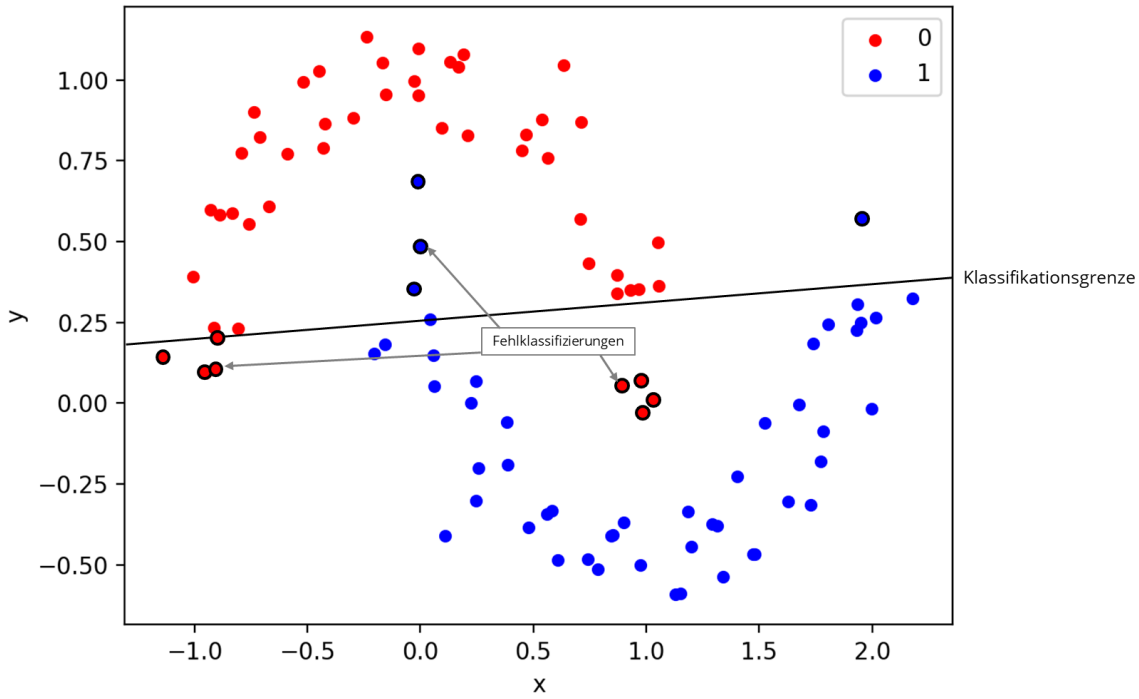


Abbildung 6 - Anwendung einer linearen Klassifikation auf die Klassen 0 (rot) und 1 (blau)

Die Klassifikationsgrenze trennt einen Großteil der Attribute korrekt, jedoch einen Teil inkorrekt. Die Klassifikation liefert somit keine optimalen Ergebnisse[11]. Ob diese fehlende Genauigkeit bei der Klassifikation von vermeintlichen Extremwerten hinzunehmen ist, hängt von der statistischen Relevanz bzw. vom Anwendungsfallszenario ab.

Im Gegensatz zur linearen Klassifikation werden die Klassen bei einer **nicht-linearen Klassifikation** durch eine oder mehrere nicht-lineare Funktionen getrennt. Wie bei einer linearen Klassifikation können auch bei einer nicht-linearer Klassifikation Fehlklassifikationen auftreten (siehe Abbildung 7).

[11] Für diesen Anwendungsfall wird davon ausgegangen, dass der Algorithmus an diesem Punkt verharrt und dies das optimale Ergebnis darstellt.

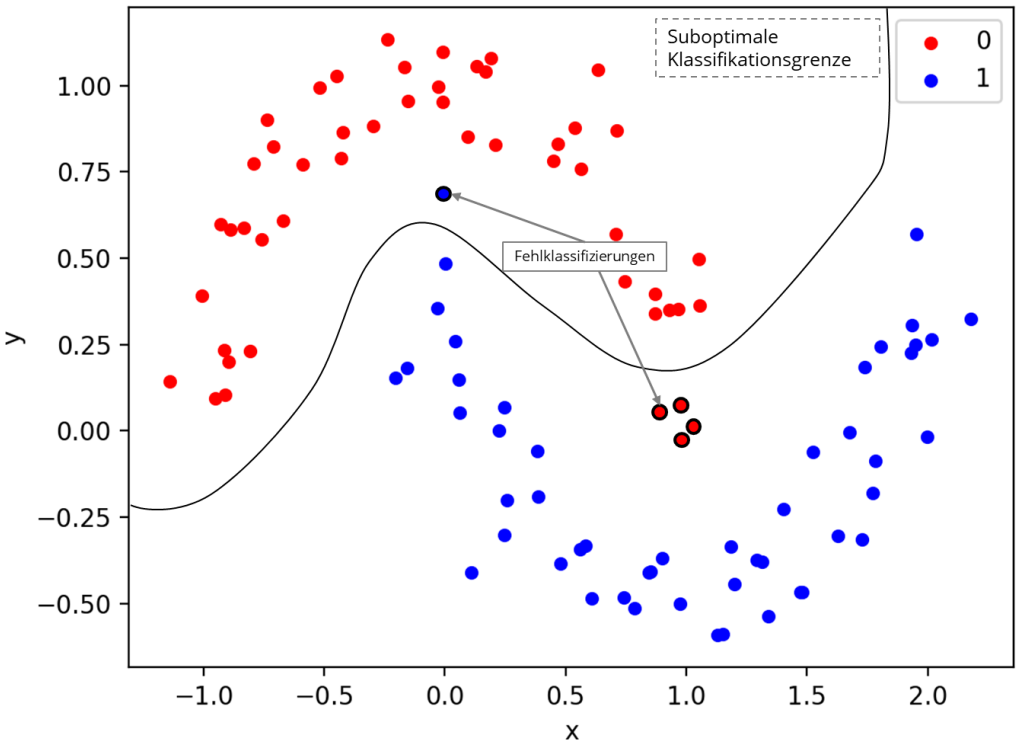


Abbildung 7 - Suboptimale nicht-lineare Klassifikation der Klassen 0 (rot) und 1 (blau)

Diese könnten z.B. durch zu frühes Halten einer Optimierung verursacht werden [29]. Durch eine optimale nicht-lineare Klassifikation kann das Problem optimal gelöst werden (siehe Abbildung 8).

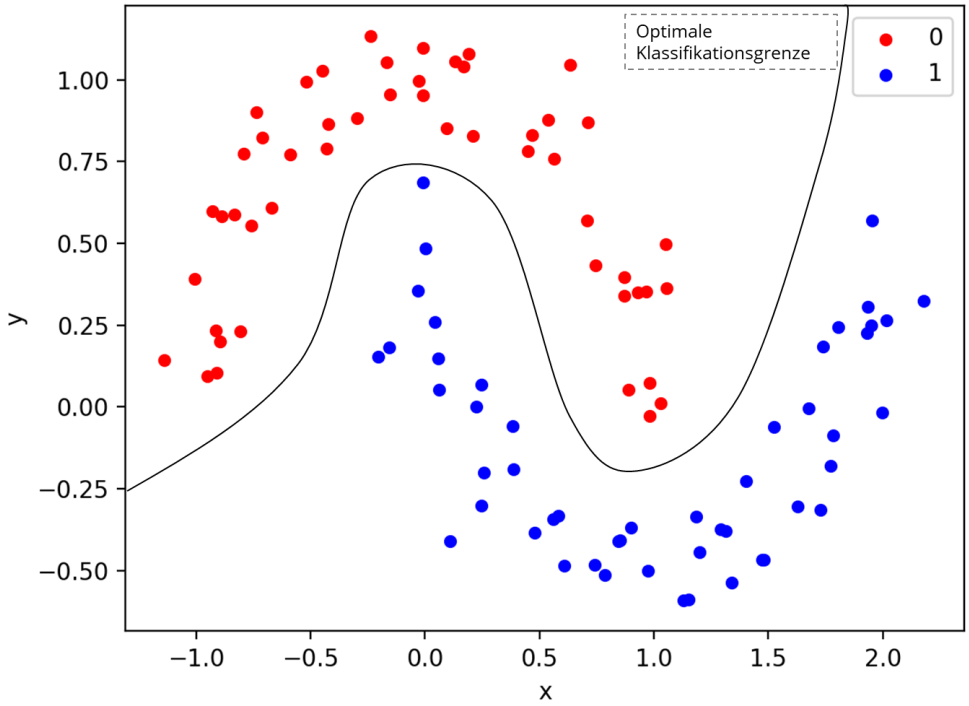


Abbildung 8 - Optimale nicht-lineare Klassifikation

Der Einsatz einer nicht-linearen Klassifikation hat im Vergleich zu einer linearen Klassifikation meist einen höheren Rechenaufwand zur Folge (bspw. mehr Schritte zur Bildung einer Klassifikationsgrenze und deren Annäherung). Dies wird jedoch durch die häufig besseren Klassifikationsergebnisse, insb. bei nicht-linear verteilten Werte, zur Folge [29].

Im Gegensatz zu einer binären Klassifikation wird bei einer **mehrklassigen Klassifikation** zwischen mehr als zwei Klassen unterschieden. Zur Erläuterung wird das bislang verwendete Beispiel auf drei Klassen erweitert (siehe Klasse 2 (grün), Abbildung 9).

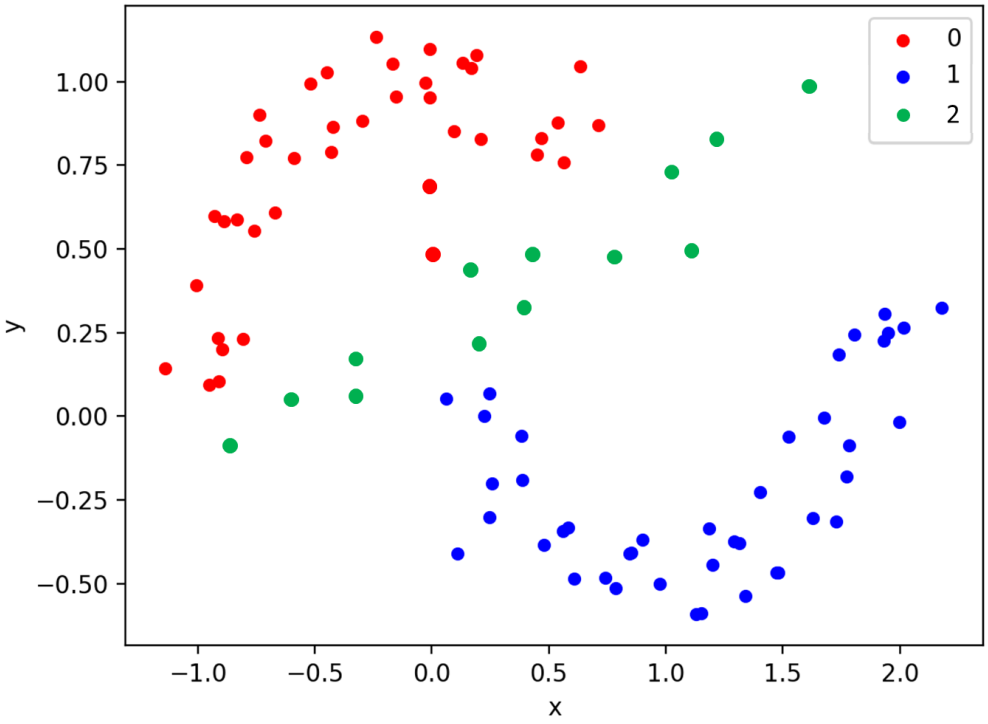


Abbildung 9 - Visualisierung von Werten der drei Klassen 0 (rot), 1 (blau) und 2 (grün)

Auch bei einer mehrklassigen Klassifikation können lineare, oder nicht-lineare, Klassifikationsverfahren eingesetzt werden [27]. Als Erweiterung der linearen Klassifikation binärer Klassen, lassen sich zur mehrklassigen Klassifikation auch mehrere lineare Klassifikationsgrenzen verwenden. Die drei Klassen aus Abbildung 9 lassen sich mit Hilfe von zwei Klassifikationsgrenzen $\{\alpha, \beta\}$ korrekt klassifizieren (siehe Abbildung 10). Das Problem lässt sich somit bereits mit einer linearen Klassifikation durch zwei Klassifikationsgrenze effizient lösen.

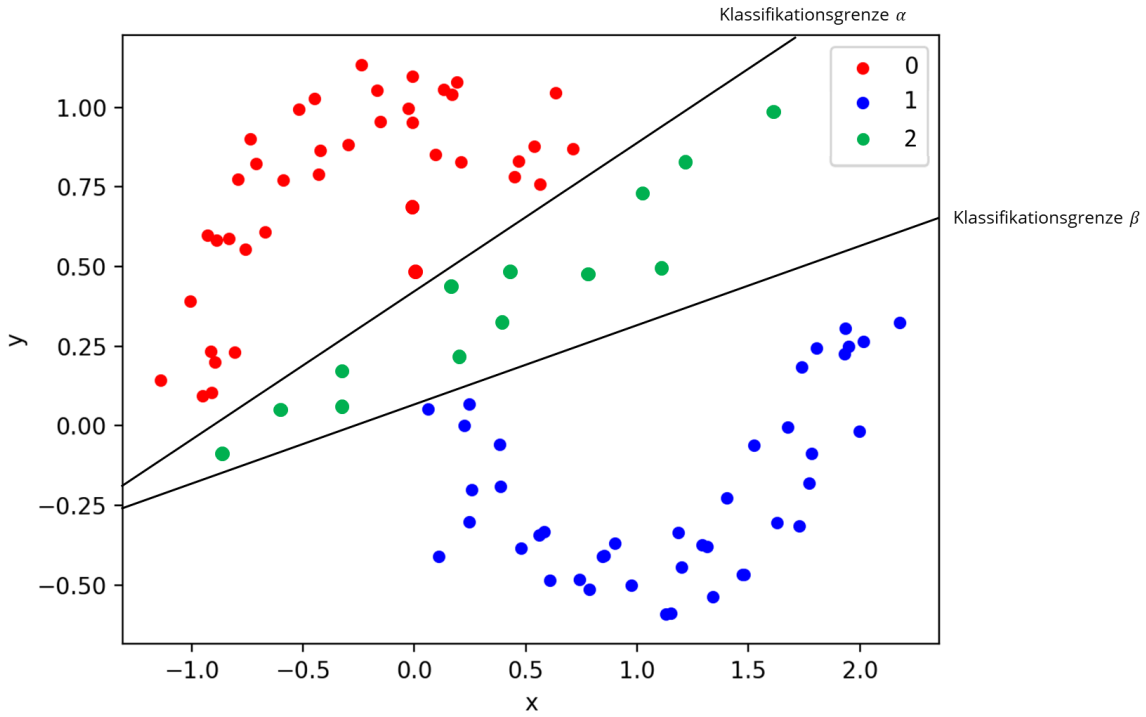


Abbildung 10 - Klassifikationsgrenzen zwischen drei Klassen

Klassifikationsprobleme beschränken sich nicht nur auf explizit numerische Probleme bzw. Werte, sondern können z.B. auch sprachlicher, akustischer oder visueller Natur sein. Ein Beispiel hierfür ist der praktische Teil dieser Thesis (siehe Kapitel 3), in der eine Bildklassifikation realisiert wird. Die Grundlagen hierfür werden im folgenden Kapitel 2.3.1.1.1 vermittelt.

2.3.1.1.1 Bildklassifikation

Bildklassifizierung zählt zu Problemen des überwachten Lernens. Hierbei wird ein Modell darauf trainiert, die Zugehörigkeit von Bildern zu bestimmten Zielklassen zu erkennen [30, 31]. Frühe Computervision-Modelle stützen sich auf unverarbeitete Pixeldaten als Input für das Modell [Vgl. 31]. Wie in Abbildung 11 gezeigt wird, bietet beispielsweise die Mittelwertbildung der Rohpixeldaten allein keine ausreichend stabile Darstellung, um die Variationen eines Objekts zu erfassen [31].

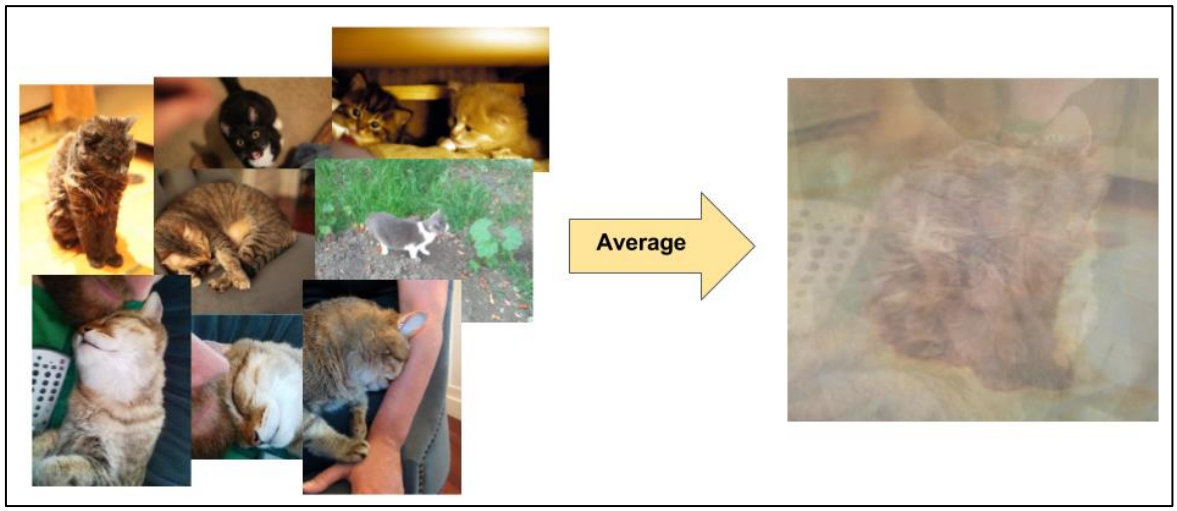


Abbildung 11 - Verschiedene Katzenbildern (links) zur Bildklassifikation mittels der Mittelwertbildung von Pixeldaten (rechts) [Vgl. 31]

Die Position des Objekts, der Hintergrund des Objekts, die Umgebungsbeleuchtung, der Kamerawinkel und der -fokus sind nur einige Beispiele für Faktoren, die Fluktuationen in den Rohpixeldaten erzeugen können [31]. Des Weiteren gehen analog zur Anzahl der Bilder durch die Überlappung und Vermischung Bildinformationen verloren [31]. Um Objekte flexibler zu modellieren, fügen moderne Computervision-Modelle neue, aus Pixeldaten abgeleitete, Funktionen zur Extraktion von sog. Bildfeatures[12] hinzu (siehe Kapitel 2.3.2.1.1) [31–33]. Voraussetzung für die korrekte Funktionsweise ist die richtige Wahl der zu extrahierenden Features [34]. Die Extraktion einer Vielzahl an möglichen Features kann ebenso negative Konsequenzen besitzen, wenn z.B. eine Fehlauswahl getroffen wird, oder die Optimierung der Features nicht (effizient) erreicht werden kann [31, 32].

2.3.1.2 Regression

Regression ist ein statistisches Verfahren, welches im Rahmen von überwachten Lernverfahren angewendet wird [28]. Im Gegensatz zur Klassifikation modelliert die Regression einen Zielvorhersagewert auf der Grundlage unabhängiger Variablen

[12] Siehe Glossar: Feature.

[26]. Sie wird hauptsächlich dazu verwendet, die Beziehung zwischen Variablen herauszufinden und Vorhersagen zu treffen [9]. Ein Regressionsalgorithmus liefert dazu numerische Werte, typischer Weise als Fließkommazahlen [28].

Eine **lineare Regression** stellt eine Beziehung zwischen der abhängigen Variablen Y und einer oder mehreren unabhängigen Variablen X_i her, indem eine sog. Regressionsgerade durch die Variablen gezogen wird (siehe Abbildung 12) [35].

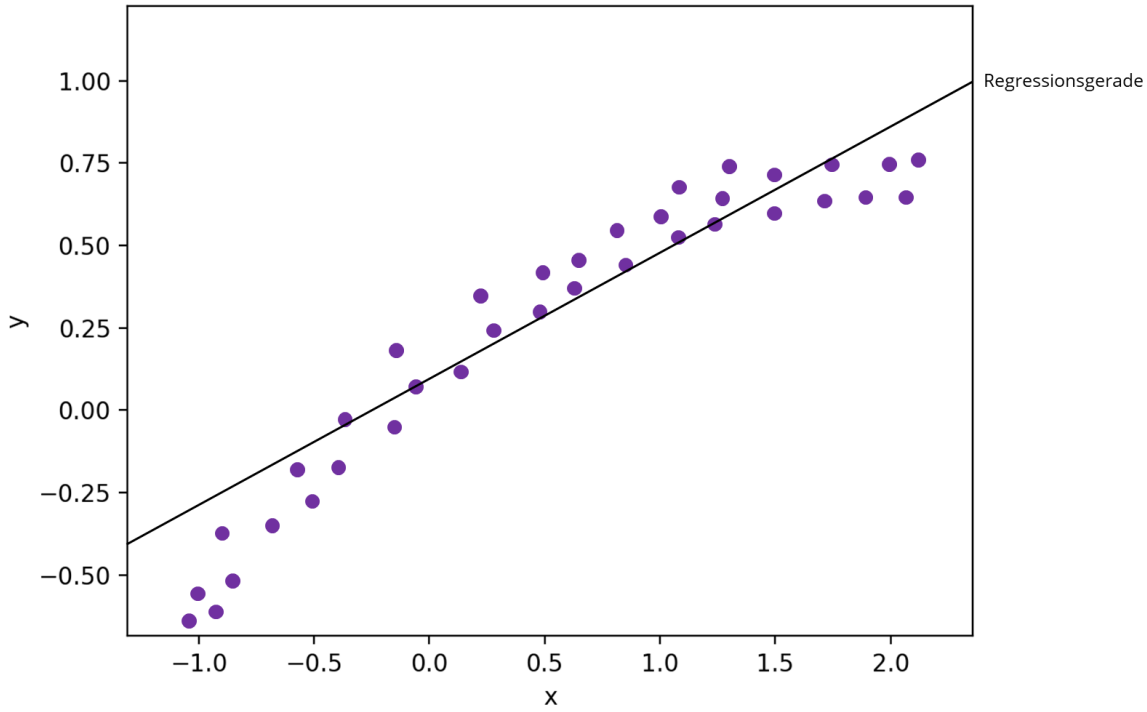


Abbildung 12 - Beispiel für eine lineare Regression

Zur Bestimmung der Regressionsgeraden können verschiedene mathematische Verfahren angewendet werden. Ein Verfahren hierzu ist die Methode der kleinsten Quadrate [35]. Hierbei wird die quadratische Summe der Abweichungen jedes Punktes zur Regressionsgeraden, den sog. Residuen, berechnet (siehe Abbildung 13) [35]. Die optimale Regressionsgerade verläuft dort, wo die Summe der Fehlerquadrate minimal ist (siehe Abbildung 13) [35].

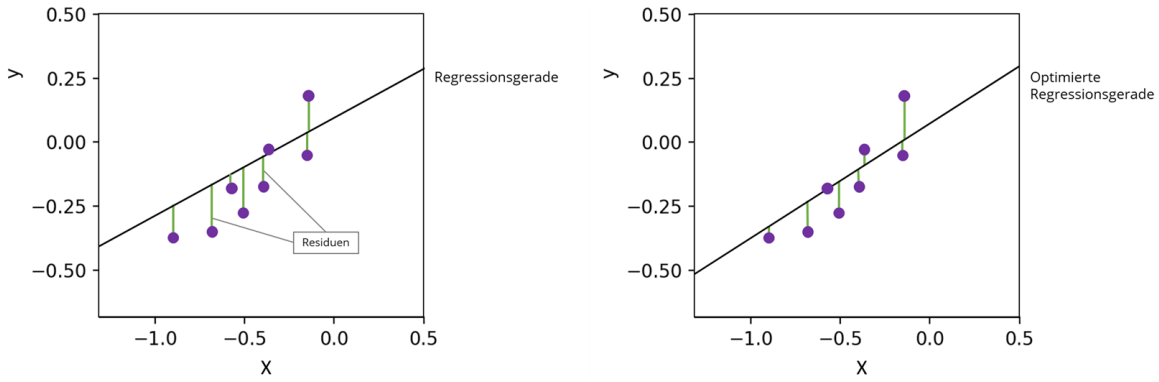


Abbildung 13 - Beispielhafte (links) und optimierte Regressionsgerade (rechts) mit Residuen

Wie bereits bei der Klassifikation gezeigt, können auch bei der Regression **nicht-lineare Funktionen** eingesetzt werden [36]. Typischerweise kommen hier Sigmoid-, Logarithmus- oder Polynomialfunktionen zum Einsatz [35]. In Abbildung 14 ist eine nicht-lineare Regression mittels einer Polynomialfunktion dargestellt. Die hierdurch entstehende Regressionsgrenze wird auch als Regressionsfunktion oder Regressionskurve bezeichnet [35].

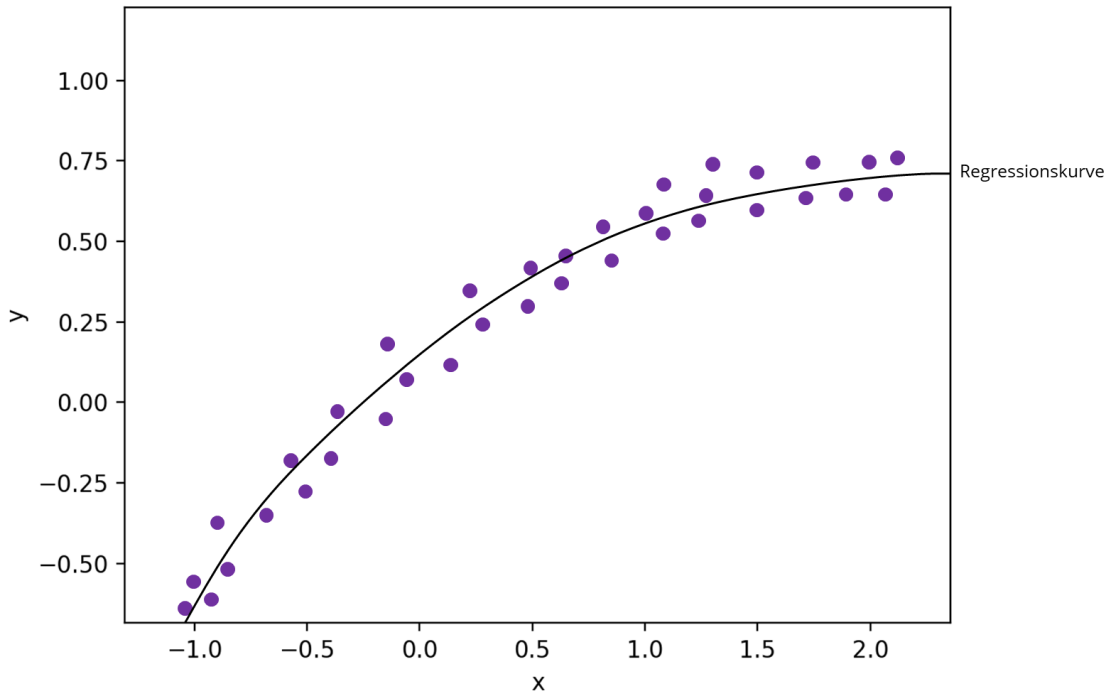


Abbildung 14 - Nicht-lineare Regression

Sowohl Regression als auch Klassifikation sind Vorhersagemodelle im Rahmen unüberwachter Lernverfahren [17]. Sie werden bspw. im Bereich der

Preisentwicklung, vorausschauenden Instandhaltung oder Bilderkennung eingesetzt [Vgl. 17]. Welches Vorhersagemodell gewählt wird ist dabei anwendungsabhängig. Während Vorhersagen über stetige Werte mittels Regression getroffen werden können (z.B. über die Einkommensentwicklung einer Person), erlaubt eine Klassifikation die Unterscheidung zwischen Klassen (bspw. über die Kreditwürdigkeit einer Person) [17].

Auch die Ergebnisse sind stets im Kontext des zu lösenden Problems zu betrachten. Die Komplexität eines Modells korreliert nicht zwingend mit seiner Effektivität bzw. Effizienz. Modelle, welche ähnlich gute Ergebnisse liefern, sich aber erheblich im Aufwand (Modellaufbau, Rechenleistung, etc.) unterscheiden, sind detailliert zu bewerten.

2.3.1.3 Entscheidungsbäume (Klassifikation / Regression)

Zur Umsetzung einer Klassifikation oder Regression können Baumstrukturen verwendet werden. Entscheidungsbäume sind Bäume, bei denen die Knoten (einschließlich der Wurzel) Eingabevariablen und die Blätter Entscheidungsergebnisse repräsentieren. Die Entscheidungsgrenzen werden durch die Pfade des Baumes repräsentiert (siehe Abbildung 15) [37, 38].

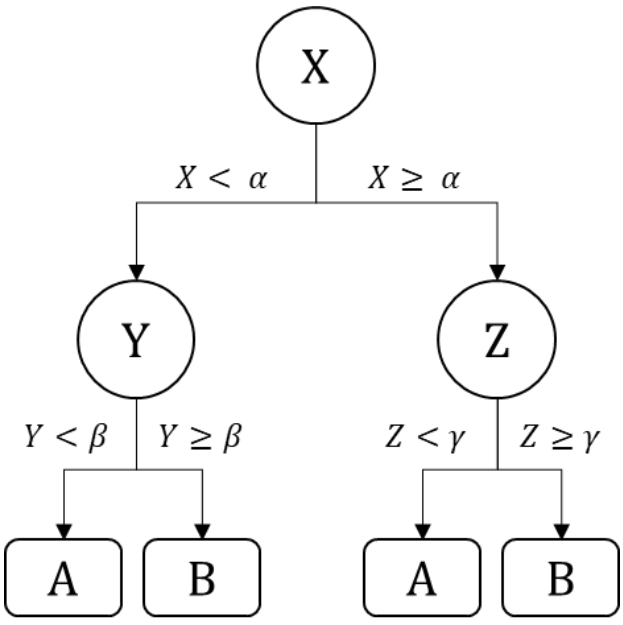


Abbildung 15 - Entscheidungsbaum

Der Entscheidungsbaum aus Abbildung 15 verfügt über die drei Knoten $\{X,\ Y,\ Z\}$, die Blätter $\{A,\ B\}$, und die Entscheidungsgrenzen $\{\alpha,\ \beta\}$. Die Blätter stellen je nach Anwendung einer Klassifikation bzw. einer Regression die Entscheidungsklassen bzw. numerische Werte dar. Entscheidungsbäume sind eine der frühsten und gleichzeitig bekanntesten Machine Learning Methoden und sind für vielfältige Klassifizierungszwecke anwendbar [37, 39]. Sie können sowohl zur Klassifikation als auch zur Regression verwendet werden und werden dementsprechend als Klassifikationsbäume und Regressionsbäume bezeichnet [37, 40]. Klassifikationsbäume werden angewendet, wenn auf ein diskretes Ergebnis (Klassen) geschlossen werden soll. Regressionsbäume werden indessen zur Vorhersage von numerischen Attributen verwendet.

2.3.2 Deep Learning

Unter Deep Learning wird eine Familie von maschinellen Lernalgorithmen gefasst, die auf künstlichen Neuronalen Netzen (KNN) basieren [17]. Dies sind algorithmische Netzstrukturen, welche sich in Grundzügen an der Funktionsweise biologischer Neuronen orientieren [17, 20].

2.3.2.1 Künstliche neuronale Netze

Die Ursprünge künstlicher neuronaler Netze gehen auf Warren McCulloch und Walter Pitts (1943) zurück [41]. Inspiriert von der Neurobiologie schlagen sie ein binäres grenzwertbasiertes Berechnungsmodell vor, welches sich an den rechnerischen Fähigkeiten realer Neuronen orientiert [42]. Damit verfolgen sie das Ziel, universelle Konzepte aus spezifischen Wahrnehmungen abzuleiten [43].

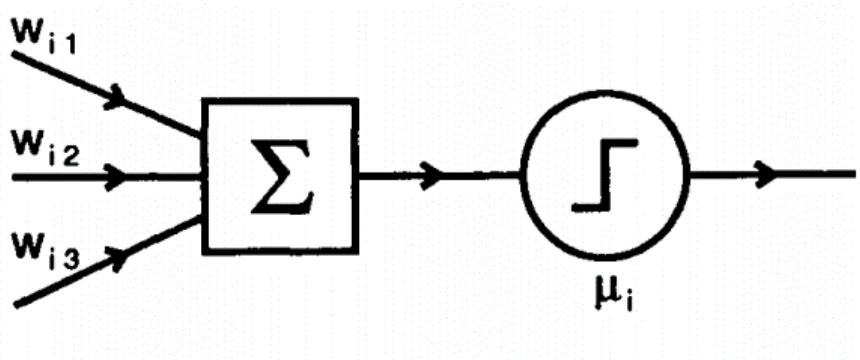


Abbildung 16 - Schematische Abbildung eines McCulloch-Pitts Neurons [43]

In Abbildung 16 ist ein McCulloch-Pitts Neuron dargestellt. Die Eingänge w_{ij} werden mittels gerichteten, ungewichteten Kanten in einer Summenfunktion zusammengeführt [42]. Zur Aktivierung des Neurons wird in diesem Fall die Heaviside Aktivierungsfunktion[13] verwendet [43].

$$H(\mu) \coloneqq \begin{cases} 0, for\ \mu < 0 \\ 1, for\ \mu \geq 0 \end{cases}$$

Formel 1 - Heaviside Funktion

Abhängig des Schwellenwerts μ_i wird das Neuron aktiviert und überträgt einen binären Wert [42, 43]. Diese Neuronen arbeiten ausschließlich mit binären Signalen, sodass sie auch nur in der Lage sind, binäre Ergebnisse zu erzeugen und über die Kanten binäre Daten zu übertragen [43]. Auf den ersten Blick scheint das McCulloch-Pitts Modell eingeschränkt, da nur binäre Informationen produziert und übertragen werden [43]. Diese Neuronendarstellung enthält jedoch alle notwendigen Funktionen zur Implementierung komplexerer Modelle [43]. Weitere Aktivierungsfunktionen, welche bspw. in modernen neuronalen Netzen Verwendung finden, werden in Kapitel 2.3.2.1.2 eingeführt.

Auf Basis von McCulloch-Pitts Neuronen entwickelt der Psychologe Frank Rosenblatt 1958 das klassische Perzeptron Modell [42, 44]. Die grundlegenden Änderungen in diesem Berechnungsmodell sind die Addition numerischer Gewichte

[13] Siehe Glossar.

sowie die Möglichkeit zur Abbildung verschiedener Verbindungsmuster [42, 44]. Der eigentliche Schritt des Lernens erfolgt durch die Anpassung der Gewichte [43].

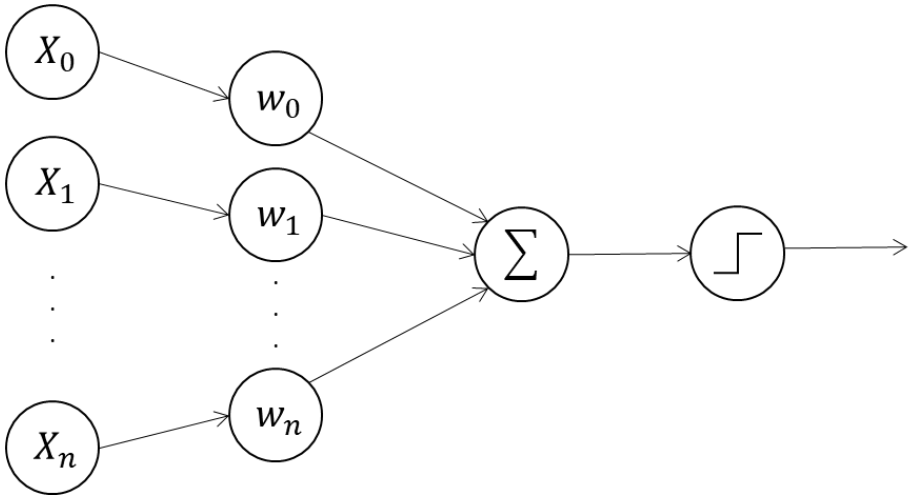


Abbildung 17 - Vereinfachte Abbildung eines Rosenblatt Perzeptron [Erstellt auf Basis von 44–46]

Wie in Abbildung 17 dargestellt, liegt der Unterschied in der Implementierung der Gewichtsknoten w_i. Minsky und Papert zeigen, dass diese Modelle ausschließlich in der Lage sind, linear-separierbare Probleme zu lösen [46]. Perzeptrone verfügen in ihrer Urform nach Rosenblatt bzw. Minsky und Papert nur über eine Schicht und werden daher als Einschichtperzeptron (Single Layer Perceptron) bezeichnet [44, 46]. Es gilt festzuhalten, dass einzelne Neuronen nicht in der Lage sind, komplexe Berechnungen durchzuführen [43]. Dies wird durch die burschikose Grundidee des Mehrschicht Perzeptron behoben, die auf Werbos (1974) sowie auf Rumelhart, McClelland und Hinton (1986) zurückgeht [43, 47, 48]. Mit Hilfe von Mehrschicht Perzeptronen (siehe Abbildung 18) werden Neuronen in Schichten hintereinander modelliert [20, 49].

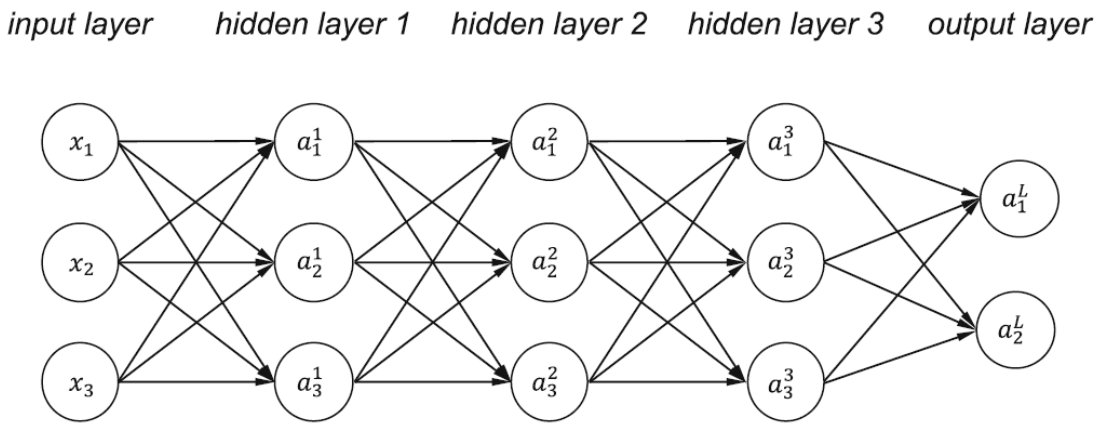


Abbildung 18 - Beispiel eines Mehrschicht Perzeptrons [Vgl. 45]

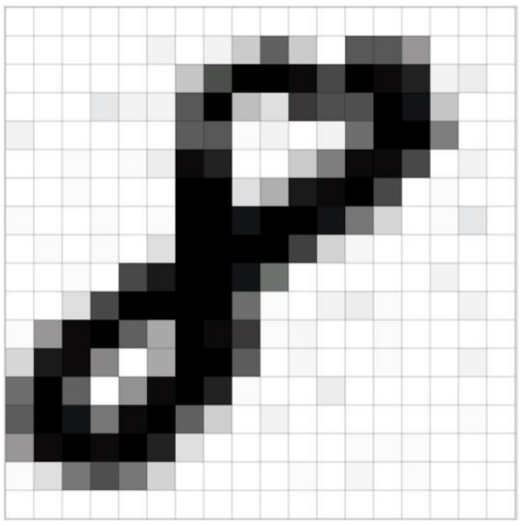
Diese Schichten umfassen stets eine Eingabeschicht (input layer), über welche Daten in das System eingelesen werden, sowie eine Ausgabeschicht (output layer), über welche die Ausgaben erfolgen [20, 49]. Dazwischen können sich versteckte Schichten (hidden layer) befinden, welche den eigentlichen Lernphasen entsprechen [20]. Die Bezeichnung hidden layer bezieht sich auf die Position der Schichten im Netzwerk zwischen Input- und Output Layer [45]. Die neuronalen Netze können Ebenen unterschiedlicher Komplexität aufspannen [49]. Zum Beispiel kann ein System in der ersten Ebene mit der der Identifizierung von relativ einfachen Mustern (z.B: Helligkeitsstufen von Pixeln bei Bildern) beginnen [49]. In der zweiten Ebene werden komplexere Features, wie z.B. Kanten erfasst. In der dritten Ebene folgt eine Erfassung detaillierter Formen und Objekte [Vgl. 49]. Durch diese aufeinanderfolgenden Schichten werden die internen Verknüpfungen des Netzwerks kontinuierlich optimiert, was dem Lernen des Modells entspricht.

Durch die fortschreitende Entwicklung auf dem Gebiet des Deep Learning werden eine Vielzahl unterschiedlicher Netzarchitekturen entwickelt. Die Grundidee bei einer Vielzahl dieser Architekturen geht weiterhin auf das Prinzip des McCulloch-Pitts Neurons, bzw. des Perzeptrons zurück [33, 45]. Eine Übersicht über verschiedene neuronale Netzwerktypen befindet sich im Anhang (siehe Abbildung 51). Im folgenden Kapitel 2.3.2.1.1 wird das in dieser Thesis verwendete CNN näher erläutert.

2.3.2.1.1 Convolutional Neural Network (CNN)

Ein Convolutional Neural Network (CNN) ist ein eine Deep Learning Netzwerkarchitektur, die primär zur Bildklassifikation verwendet wird [45, 50]. Das CNN unterscheidet Eingabebilder anhand verschiedener Bildfeatures (siehe Kapitel 2.3.1.1.1), die dementsprechend gewichtet werden [45, 50]. Die für den Einsatz eines CNN erforderliche Vorverarbeitung ist im Vergleich zu anderen Klassifikationsalgorithmen geringer [30, 50, 51]. Während bei primitiven Methoden zur Bildklassifikation die jeweiligen Features manuell entwickelt werden (siehe Kapitel 2.3.1.1.1), besitzen CNNs die Fähigkeit diese Features bei ausreichendem Training selbstständig zu erlernen. Zur korrekten Klassifikation sind in der Verarbeitung einige Schritte notwendig, die nachfolgend erläutert werden.

Vereinfacht gesprochen wird ein Bild wird von einem Computer als Matrix aus Pixelwerten „gesehen“. Eine monochrome Darstellung einer Acht in einer 18x18 Pixel Grafik kann bspw. als Array der Schwarzwerte aufgefasst werden (siehe Abbildung 19).



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 19 - Repräsentation des monochromen Bildes (links) als Matrix der Schwarzwerte (rechts)

Im Falle eine polychromen Grafik können die Kanäle (rot, grün, blau) getrennt voneinander verarbeitet werden, wie in der folgenden Abbildung 20 darstellt.

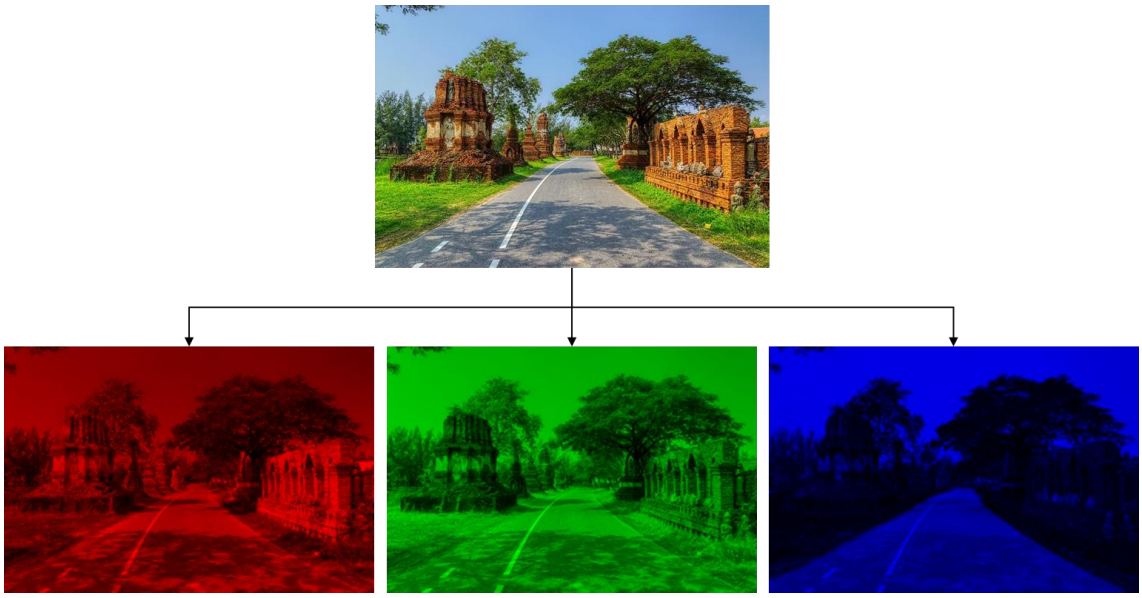


Abbildung 20 - Trennung eines Bildes in seine Kanäle (rot, grün, blau) zur Weiterverarbeitung [52]

Durch Auffassung eines Bildes als Array können verschiedene Manipulationen durchgeführt werden. Ein typisches CNN verwendet hierzu die Operationen Convolution[14] und Pooling, die als Schichten im Netzwerk eingesetzt werden.

Eine **Convolution** (Faltung) ist mathematisch betrachtet eine kombinierte Integration der Funktionen f und g, wobei die Funktionen übereinander verschoben werden [51, 53, 54].

$$(f \otimes g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Formel 2 - Convolutionsfunktion von f und g

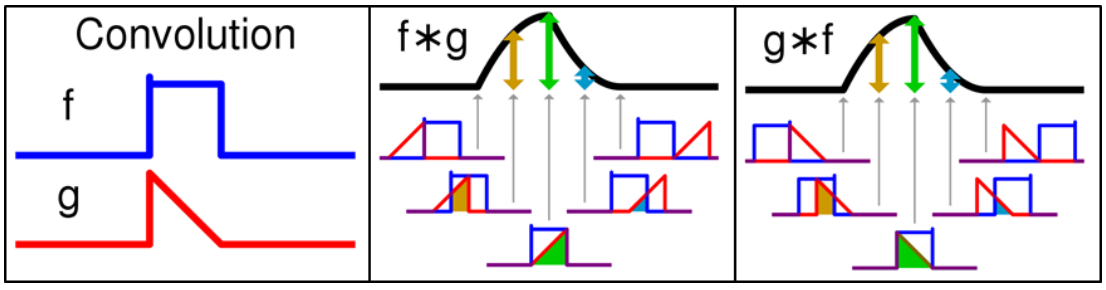


Abbildung 21 - Convolution der zwei Funktionen f und g [angepasst entnommen aus 54]

[14] Auch bekannt als Filter Layer oder Kernel Layer.

Convolutional Layer verwenden im Rahmen von neuronalen Netzen Faltungsoperationen von Matrizen zur Bildmanipulation (sog. convolutional operations) [28, 51]. Hierzu werden zwei Elemente benötigt. Einerseits die Faltungsoperation und andererseits (ein Ausschnitt) eine/r Eingabematrix[15] (siehe Abbildung 22) [28]. In Abbildung 22 wird eine 5x5 Eingabematrix und eine 3x3 Faltungsmatrix dargestellt. Zur Vereinfachung werden binäre Eingabewerte verwendet.

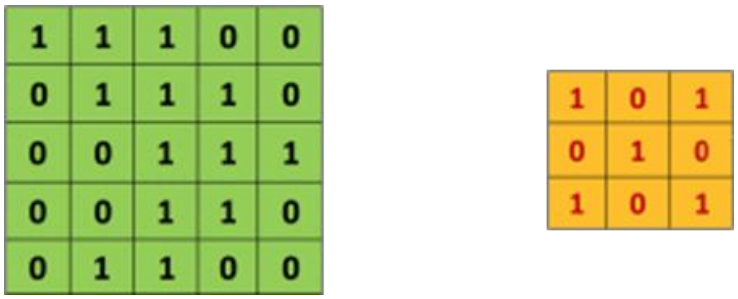


Abbildung 22 - Eingabematrix (5x5) und Faltungsmatrix (3x3) [55]

Die Faltungsmatrix wird erschöpfend über die Eingabematrix verschoben und erzeugt durch Addition der jeweiligen Werte, auf Basis einer logischen AND Verknüpfung, ein sog. Convolved Feature[16] (siehe Abbildung 23). Der Schritt der Verschiebung der Faltungsmatrix über die Eingabematrix wird als Stride bezeichnet [55].

[15] Auch bekannt als Filter oder Kernel.

[16] Auch als Feature Map bezeichnet.

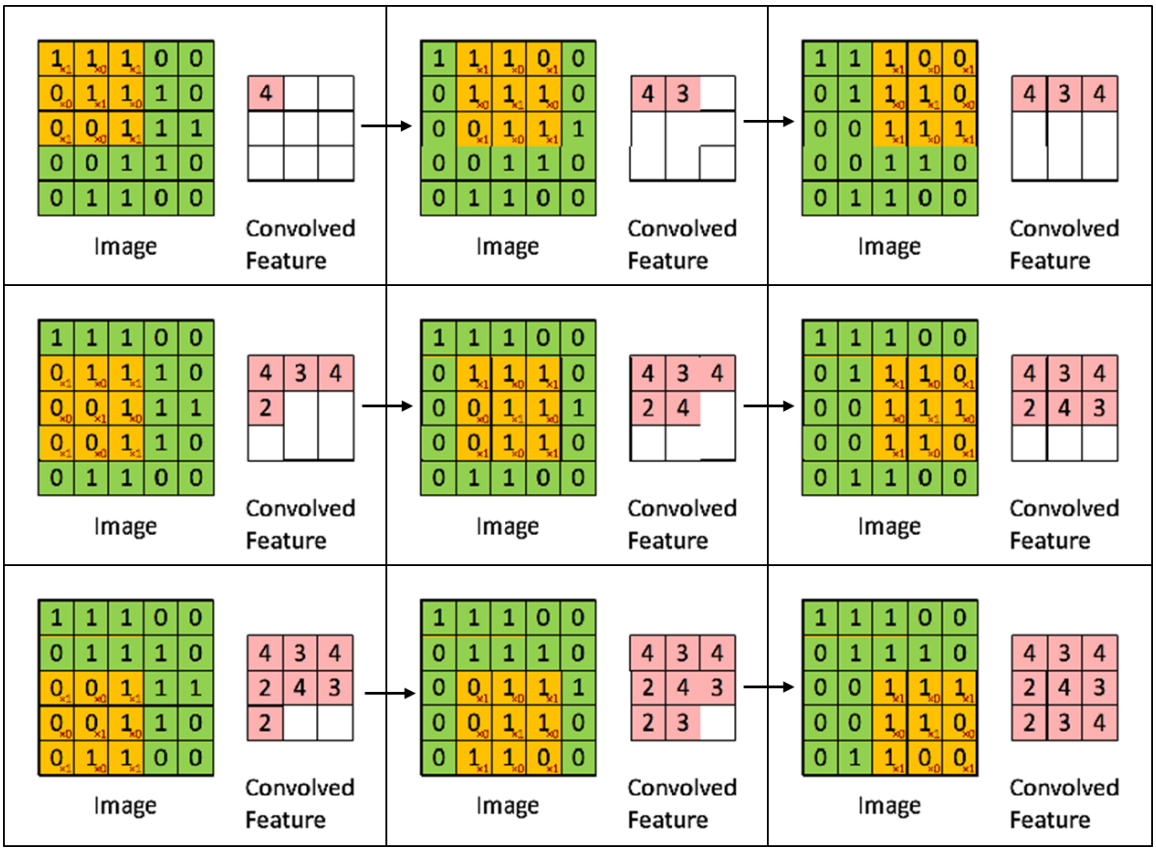


Abbildung 23 - Ablauf einer Convolution [55]

Das Ziel der Convolution ist die Extraktion von Features aus dem Eingabebild mittels verschiedener Filter [56]. In Tabelle 1 sind verschiedene Filteroperationen und ihre Effekte auf Bilder in CNNs abgebildet. Diese werden typischer Weise als 3 x 3 , 5 x 5 oder als 7 x 7 Matrizen angewendet [56].

Operation	**Filter (3x3)**	**Convolved Image**
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge Detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	

	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box Blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian Blur (approximated)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Tabelle 1 - Verschiedene Filteroperationen in CNNs [55]

Pooling beschreibt die Dimensionsreduktion einer Matrix, die in vorangegangenen Convolutional Layern erzeugt wird [28]. Pooling wird dazu verwendet, nur die relevantesten Signale an die nächsten Schichten weiterzugeben, um so eine abstraktere Repräsentation des Inhalts zu erreichen und die Parameteranzahl eines Netzes zu reduzieren [57]. MaxPooling beschreibt hierbei die Reduktion auf den Maximalwert im betrachteten Bereich [57]. Eine MaxPooling Operation teilt (wie die Convolution) eine Matrix in Teilbereiche auf und arbeitet diese in Strides ab (siehe Abbildung 24) [28].

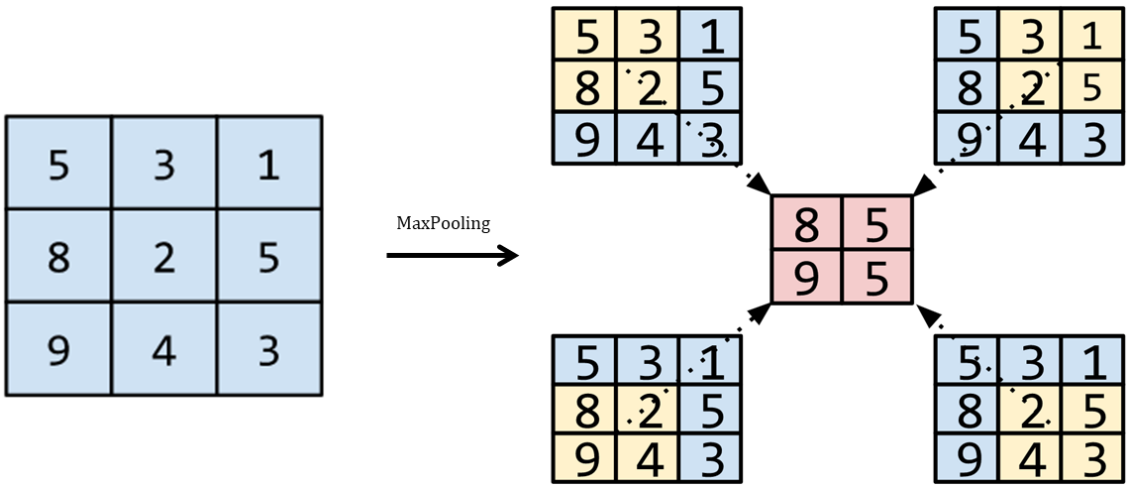


Abbildung 24 - Beispielhafte Anwendung einer MaxPooling Operation [Vgl. 28]

Wie in Abbildung 24 dargestellt arbeitet eine 2x2 Faltungsmatrix die 3x3 Eingabematrix in 1x1 Strides ab. So finden insgesamt vier Pooling Schritte statt.

Als zusätzliche Schicht, zwischen Convolution- und Pooling Schichten, kann in einem CNN zur Reduktion von Overfitting ein **Dropout** Layer eingesetzt werden [28, 58]. Hierbei wird ein bestimmter Prozentsatz zufälliger Einheiten im Netzwerk beim Training ausgelassen [58]. Der Dropout Layer bildet somit eine Form der Regularisierung[17] [58]. Je mehr Einheiten ausgelassen werden, desto stärker ist die Regularisierung [58].

Abschließend folgen ein oder mehrere **Dense** Layer [Vgl. 57]. Der Dense Layer (auch als **Fully Connected** Layer bezeichnet) ist ein mehrschichtiges Perzeptron [55]. Der Begriff Fully Connected impliziert, dass jedes Neuron in der vorherigen Schicht mit jedem Neuron auf der nächsten Schicht verbunden ist [Vgl. 55]. Während die Ausgabe der Convolutional Layer und Pooling Layer jeweils unterschiedliche Features des Eingabebildes darstellt, hat ein Dense Layer zum Zweck, diese Features, auf Basis eines Trainingsdatensatzes zur Bildklassifizierung des Eingabebildes zu nutzen [55]. Hierzu steht der finale Fully Connected Layer Output in einer 1:1 Relation zur Anzahl der Klassifizierer[18].

CNNs bestehen typischerweise aus einer Sequenz von jeweils zwei Convolutional Layern mit der gleichen Anzahl an Filtern, gefolgt von einem Pooling Layer, auf den wiederum zwei Convolutional Layer und ein Pooling Layer folgen [57]. Während die Größe des Inputs durch die Faltungen und Pooling immer weiter reduziert wird, nimmt die Anzahl der Filter zur Erkennung von übergeordneten Signalen zu [55, 57].

---

[17] Siehe Glossar.

[18] Beispielsweise werden zur Bildklassifikation im Rahmen dieser Thesis 9 Klassifizierer (und somit 9 Outputs des letzten Fully Connected Layer) verwendet.

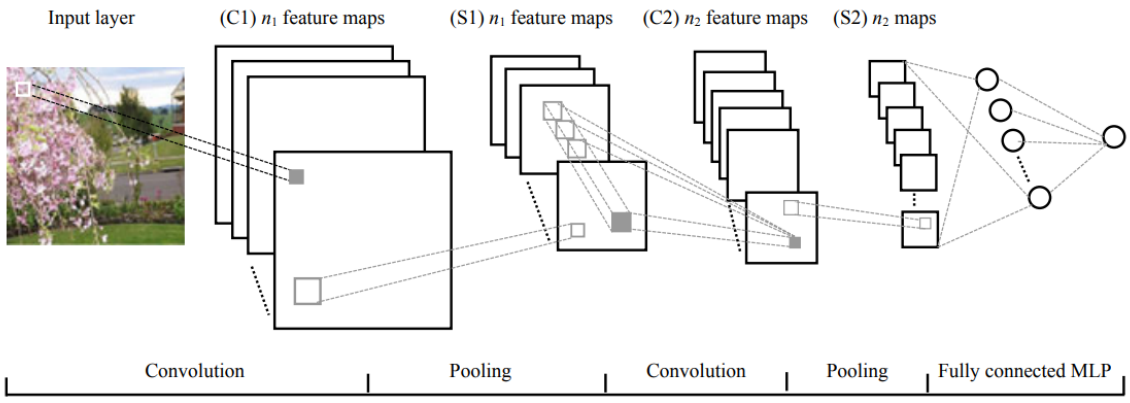


Abbildung 25 - Zusammenfassung eines beispielhaften CNN Modellaufbaus [Vgl. 59]

Abbildung 25 zeigt den kompletten Modellaufbau des hier beschriebenen CNN mit zwei sich jeweils abwechselnden Convolution und Pooling Schichten und einem Fully Connected Output Layer [59]. Eine allgemeingültige Vorgehensweise, in welcher Kombination die o.g. Schichten anzuwenden sind, existiert nicht. Hier gilt es anhand der zu lösenden Problemszenarien unterschiedliche Variationen zu erproben und deren Ergebnisse gegeneinander abzuwägen [23].

Zur Aktivierung der Neuronen in den o.g. Schichten können verschiedene Aktivierungsfunktionen verwendet werden. Die im Rahmen dieser Thesis verwendeten Funktionen ReLU und Softmax werden im folgenden Kapitel 2.3.2.1.2 eingeführt.

2.3.2.1.2 Aktivierungsfunktionen

Eine Aktivierungsfunktion empfängt die gewichtete Summe aller Eingaben aus der vorherigen Schicht, erzeugt einen (typischerweise nicht-linearen) Ausgabewert und gibt diesen an die nächste Schicht weiter [59]. Nachfolgend werden die in dieser Thesis verwendeten Funktionen ReLU und Softmax erklärt.

2.3.2.1.2.1 ReLU

Die Rectified Linear Unit Funktion (ReLU) ist eine nicht-lineare Aktivierungsfunktion, welche in mehrschichtigen bzw. tiefen neuronalen Netzen angewendet wird [55]. In

einem CNN werden die Ergebnisse jedes Layers typischerweise durch die ReLU Funktion aktiviert [Vgl. 55].

$$f(x) = \begin{cases} 0, wenn\ x < 0 \\ x, wenn\ x \geq 0 \end{cases}$$

Formel 3 - ReLU Funktion

Wie in Formel 3 zu erkennen, bildet die ReLU Funktion einen Eingabewert auf 0 ab, wenn dieser negativ ist und übernimmt ihn, wenn er positiv ist [59]. In einem CNN wird die ReLU Funktion als elementweise Operation angewendet und ersetzt alle negativen Pixelwerte in der Feature Map durch Nullen [55]. Der Einsatzzweck besteht darin, eine Nichtlinearität im CNN einzuführen, da auch die zu lernenden Eingabeparameter häufig nicht linear sind [55].

2.3.2.1.2.2 Softmax

Die Softmax-Funktion wandelt einen Vektor von reellen Zahlen in einen Vektor von Wahrscheinlichkeiten um [59]. Die Summe der so entstehenden Wahrscheinlichkeiten innerhalb des Vektors beträgt außerdem stets 1 [60].

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Formel 4 - Softmax Funktion

Die Softmax-Funktion wird bei mehrklassigen Klassifikationsproblemen eingesetzt, da die Ausgabe den Wahrscheinlichkeiten entspricht, mit denen auf die Klassen geschlossen wird.

2.3.2.1.3 Kostenfunktionen

Eine Kostenfunktion[19] misst die Leistung eines maschinellen Lernmodells für die gegebenen Daten [60]. Sie quantifiziert den Fehler zwischen vorhergesagten und

[19] Siehe engl. cost function, auch Verlustfunktion (loss function).

erwarteten Werten und stellt diesen als reelle Zahl dar. Abhängig vom Problemszenario kann die Kostenfunktion auf verschiedene Arten gebildet werden [60]. Der Zweck der Kostenfunktion ist entweder die Minimierung oder Maximierung eines Fehlers durch Anpassung des Modells [60, 61]. In dieser Arbeit findet die Binary Cross Entropy (BCE) Kostenfunktion Verwendung und wird daher nachfolgend eingeführt.

2.3.2.1.3.1 Binary Cross Entropy

Die Binary Cross Entropy (BCE) Kostenfunktion misst die Leistung eines Klassifikationsmodells, dessen Ergebnis ein Wahrscheinlichkeitswert ist [61].

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log\big(p(y_i)\big) + (1 - y_i) * \log\,(1 - p(y_i))$$

Formel 5 - Binary Cross Entropy Kostenfunktion

Hierzu werden die Parameter y für den Zielwert der jeweiligen Klasse und die zugehörige Wahrscheinlichkeit $p(y)$ verwendet [62, 63]. Diese stellt die vom Modell berechnete Wahrscheinlichkeit dar, mit welcher das zu klassifizierende Objekt zu einer Klasse gehört. Je stärker sich die BCE von y entfernt, desto geringer ist die Wahrscheinlichkeit einer Klassenzugehörigkeit [62, 64]. Die Funktion bestraft eine falsche Klassifikation umso mehr, desto höher die ermittelte Klassifikationswahrscheinlichkeit ist. Beispielsweise wird die fehlerhafte Klassifikation eines Bildes mit 99%iger Wahrscheinlichkeit deutlich stärker bestraft, als eine fehlerhafte Bildklassifikation mit 10% Wahrscheinlichkeit [65].

2.3.2.1.4 Optimierungsverfahren

Das Training in einem neuronalen Netz erfolgt durch die gezielte Veränderung der Verbindungsgewichte (siehe Kapitel 2.3.2.1) [65]. Dies wird mittels eines Optimierungsverfahrens realisiert [65]. Hierbei werden die vom Modell durchgeführten Prognosen auf Basis der Ergebnisse einer Kostenfunktion (siehe Kapitel 2.3.2.1.3) bewertet und die Gewichte angepasst. Die Effizienz dieses

Verfahrens variiert auf Basis der Anzahl von Neuronen bzw. Schichten in einem Netzwerk sowie der Eingaben selbst [61].

Im Rahmen dieser Thesis wird das Adaptive Moment Estimation (ADAM) Optimierungsverfahren verwendet [66]. Dieses Optimierungsmodell von Diederik P. Kingma und Jimmy Ba (2014) ist als derzeit führend anzusehen [45, 66]. Es ist rechentechnisch effizient, arbeitet gut mit großen Datensätzen und besitzt einen geringen Konfigurationsaufwand [60, 61, 66]. Die genaue Funktionsweise dieses Optimierungsverfahrens ist für das Verständnis dieser Thesis nicht erforderlich, weshalb an dieser Stelle auf die detaillierten Ausführungen von Kingma und Ba verwiesen wird [66].

2.3.2.1.5 Metriken

Zur Bewertung der Modellqualität eines maschinellen Lernmodells lassen sich verschiedene Metriken heranziehen oder berechnen. Im Folgenden wird explizit auf die Bewertungsmöglichkeiten eines binären Klassifikationsproblems im Rahmen eines neuronalen Netzes eingegangen.

Die Beurteilung einer Klassifikation kann mittels einer Confusion Matrix erfolgen. Sie beschreibt das Verhältnis aus Realität und Modellvorhersage und ermöglicht so eine detaillierte Darstellung von Kennzahlen zur Modellbeurteilung (siehe Abbildung 26) [29, 67].

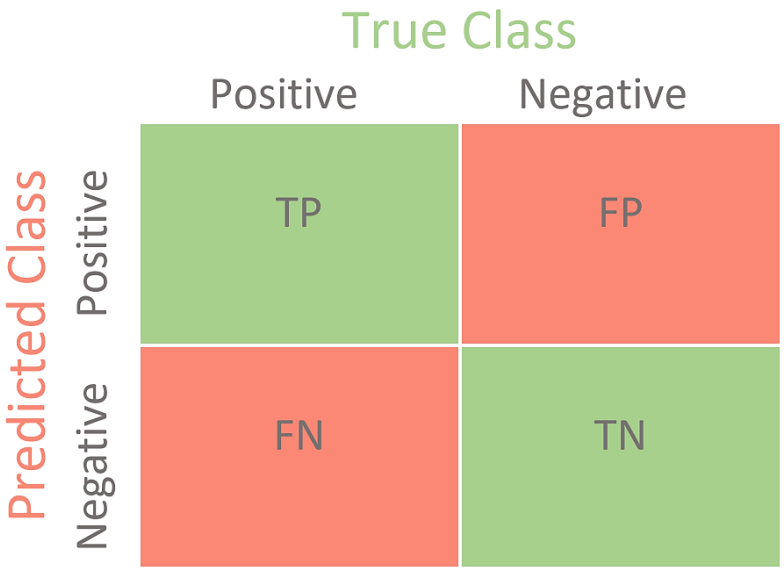


Abbildung 26 - Confusion Matrix [Vgl. 67]

Aus den verschiedenen Ausprägungen der Confusion Matrix lassen sich eine Vielzahl an Metriken ableiten. Eine dieser Metriken ist die Genauigkeit (Accuracy). Sie beschreibt den Anteil der insgesamt korrekten Vorhersagen und bildet somit eine intuitive Metrik [67].

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions} \Leftrightarrow \frac{TP + TN}{TP + FP + FN + TN}$$

Formel 6 - Accuracy / Accuracy für binäre Klassifikation

Sind die Klassen der Daten ungleich verteilt, kann das alleinige Betrachten der Accuracy jedoch zu einer Fehlinterpretation führen [68]. Wird z.B. ein Modell mit einem Datensatz bestehend aus zwei Klassen im Verhältnis 90 zu 10 trainiert, entspricht eine 90%ige Accuracy lediglich dem zufälligen Ziehen ohne Zurücklegen. Neben der Accuracy werden in dieser Thesis die Ergebnisse aus der Binary Cross Entropy Kostenfunktion (Loss) (siehe Kapitel 2.3.2.1.3.1) als Metrik herangezogen. Weitere Metriken wie F-Score, Recall oder Precision lassen sich ebenfalls aus den in Abbildung 26 dargestellten Klassen berechnen, werden im Rahmen dieser Thesis jedoch nicht verwendet und sind daher im Glossar erläutert.

2.4 Constraint Satisfaction Problems

Constraint Satisfaction Problem (CSP) bezeichnet eine Klasse von kombinatorischen Problemen, die mittels Randbedingungen (sog. Constraints) über eine Menge von Variablen formuliert werden [69, 70]. Ein Teil der in dieser Thesis gestellten Anforderungen können als CSPs aufgefasst werden [71]. Es bietet neben dem hier verwendeten ML-gestützten Verfahren einen weiteren Ansatz zur Umsetzung. Die unterschiedlichen Herangehensweisen bei CSP- oder ML-gestützen Verfahren werden in Kapitel 4.2 bewertet.

Ein CSP wird durch das Tripel (V, D, C) beschrieben, wobei gilt [69, 70]:

- $V = \{v_1, \dots, v_n\}$ beschreibt eine endliche Menge an Variablen mit zugehörigen Wertebereichen D (Domänen).
- $D = \{D_1, \dots, D_n\}\, mit\, \{v_1 : D_1, \dots, v_n : D_n\}$ beschreibt Domänen der jeweiligen Variablen und deren Zugehörigkeit.
- $C_j(V_j),\ j \in \{1, \dots, m\}$ stellt eine endliche Menge an Constraints dar, die eine Teilmenge der Variablen $V_j \in V$ in Relation setzt und deren gültige Wertekombinationen auf eine Teilmenge von $D_{J_1} \times \dots \times D_{J_k}$ beschränkt.

Bei dieser allgemeinen Definition müssen die Domänen der Variablen nicht unmittelbar als numerische Werte definiert sein [69]. Domänen, die nicht numerischer Natur sind, können ähnlich zu Kapitel 2.3.1.1.1 als solche repräsentiert werden und ermöglichen so eine Verarbeitung im Rahmen eines CSP [69, 70]. Die Lösung eines CSP erfolgt durch Finden einer konsistenten Wertebelegung aller Variablen, sodass deren Constraints erfüllt sind [69, 72]. Lösungen für CSPs werden in der Regel durch systematische Suche von möglichen Zuweisungen für Variablen gefunden [72, 73]. Algorithmen zur Propagierung von Einschränkungen treffen hierzu bestimmte Schlussfolgerungen (Inferenzen), basierend auf den Domänen und ihren jeweiligen validen Variablen [73]. Wenn ein Inferenzschritt dazu führt,

dass eine Variable nur noch eine leere Domäne besitzt[20], erfolgt ein Rückschritt in der Propagierung[21] und ein anderer Lösungszweig wird verfolgt [73, 74].

Bei der Implementierung von Algorithmen zur Lösung von CSPs (Constraint Solvern) und der Modellierung von Constraint-Problemen sind eine Vielzahl an konzeptuellen Entscheidungen zu treffen [73]. Als Beispiel gilt es festzulegen, welcher Grad an Konsistenz zu erzwingen ist, oder welche Datenstrukturen verwendet werden sollen, um eine Rückpropagierung zu ermöglichen [73]. Heute sind CSPs in vielen verschiedenen Bereichen der Informatik allgegenwärtig, von künstlicher Intelligenz und Datenbanksystemen über Schaltungsentwurf und Netzwerkoptimierung bis hin zur Theorie von Programmiersprachen [75–77]. Verschiedene Ansätze zur Involvierung maschineller Lernmethoden bieten weiterhin Optimierungsmöglichkeiten für CSPs (sog. Constraint Learning Methoden), indem sie die menschliche Komponente in der Konzeption ersetzen [69, 74, 78]. Das Ziel hierbei ist es z.B. die Heuristiken oder Lösungsstrategien automatisch anzupassen, um die Leistung der Solver zu verbessern. Aktuelle Fortschritte auf dem Gebiet des Deep Learning werden ebenso hierzu genutzt [77], wie z.B. der Ansatz CSP-cNN von Xu, Koenig und Kumar (2018) [77]. Sie wenden ein CNN an, um die Erfüllbarkeit von CSPs vorherzusagen. Mit einer Vorhersagegenauigkeit von >99.99% bei Zufallsdaten aus boolschen binären CSPs stellt diese Herangehensweise die erste effektive Anwendung von Deep Learning zur Lösung von CSP dar [77].

Die allgemeine Herangehensweise, bzw. Unterschiede in der Umsetzungen bei Machine Learning bzw. CSP Lösungsansätzen werden in der folgenden Tabelle 2 beschrieben.

ML-basierter Lösungsansatz	**CSP-basierter Lösungsansatz**

[20] An dieser Stelle wäre keine Variablenzuweisung im Rahmen der Domäne möglich.

[21] Die Propagierung erfolgt häufig via Tiefensuche mit jeweils einer Variable pro Schritt.

Datenvorverarbeitung	Überwachte Lernverfahren benötigen die Vorgabe von annotierten Klassen, auf deren Basis eine Ableitung für ungesehene Daten mittels Lernens erfolgt. Bei unüberwachten Lernverfahren ist keine Vorgabe von annotierten Daten notwendig. Der Algorithmus findet die Problemlösung selbstständig.	Genaue Festlegung von (numerischen) Variablen, Domänen und Constraints für alle Daten [79]. Alle Variablen, Domänen und Randbedingungen müssen bekannt sein, sonst ist keine eindeutige Lösung möglich[77].
Qualität	Die Ergebnisqualität von Machine Learning Verfahren ist stark von der Datenqualität (Eingabequalität) abhängig [27, 79]. Die prinzipielle Möglichkeit der Anwendung ist hiervon entkoppelt [27].	Ein CSP ist bei widersprüchlichen Eingabeparametern, bzw. nicht wohl-geformten Constraints nicht lösbar [77].
Laufzeit-verhalten	Das Laufzeitverhalten ist abhängig von den verwendeten Algorithmen und der Problem- bzw. Modelldefinition.	Bei baumstrukturierten CSPs kann die Lösung in linearer Zeit erfolgen [80].
Ab-straktion	Mathematische Vorgänge auf dem Weg der Problemlösung werden durch eine Fassade aus Programmcode maskiert.	Mathematische Zusammenhänge sind durch Analyse der Deklarationen unmittelbar nachvollziehbar.
Lösungsarten	Verschiedene Verfahren bilden die ermittelten mathematischen Zusammenhänge in unterschiedlichen Formen ab. Zum Beispiel: Neuronales Netz → Abbildung als Neuronen und Gewichtsvektoren (siehe Kapitel 2.3.2.1), Entscheidungsbäume → Abbildung als Baumstruktur (siehe Kapitel 2.3.1.3).	Ein CSP wird bspw. mit Hilfe eines Gleichungssystems gelöst, Deren Inhalte können auch als Graph dargestellt werden [80].
Optimierung	Während der Modelloptimierung werden die Gewichtsvektoren angepasst. Eine Anpassung der Gewichte hat eine Anpassung des Verhältnisses zwischen den Neuronen zur Folge [27].	Während der Propagierung werden die Domänen bzw. die Constraints nicht angepasst [80].
Um-setzung	Benötigt zur Lösung eine Repräsentation in Programmcode sowie ggf. geeignete Frameworks oder Bibliotheken zur Umsetzung.	Benötigt zur Lösung CSP-Solver bzw. Algorithmus, der Parameter des CSP zur Berechnung verwendet.

Tabelle 2 - Gegenüberstellung ML- versus CSP-basierte Lösungsansätze

Eine abschließende Bewertung zwischen der im Rahmen dieser Thesis angewendeten ML-basierten Lösung und einer möglichen CSP-basierten Lösung erfolgt in Kapitel 4.2 auf Basis der in Tabelle 2 verwendeten Kategorien.

3 Entwicklung

Im Rahmen dieser Thesis wird eine ML-gestützte Software zur Prädiktion von Reisezielen konzipiert und entwickelt. Der Entwicklungsprozess orientiert sich an CRISP-DM (siehe Kapitel 2.1.1.1). Dessen Inhalte sind detailliert in Kapitel 3.1 bis 3.4 beschrieben. Die Entwicklung erfolgt in der Programmiersprache Python unter Zuhilfenahme des Deep Learning Frameworks Keras [81, 82]. Die Darstellung erfolgt mittels des Frameworks Vue.js und wird in den Sprachen JavaScript bzw. TypeScript realisiert [83]. Bei Schritten der Datenmanipulation wird KNIME verwendet [84]. Zur inhaltlichen Beschreibung der Entwicklung werden Ausschnitte aus dem Programmcode verwendet, auf die entweder unmittelbar oder im Anhang Bezug genommen wird. Werden weitere Frameworks oder APIs verwendet, erfolgt an den jeweiligen Stellen eine Erläuterung.

3.1 Business Understanding

Es gilt einem Reisenden ein Reiseziel passend zu seinen Vorlieben zu empfehlen, ohne dass dieser eine detaillierte Spezifizierung seiner Reise durchführen muss. Durch den Einsatz von Machine Learning soll das Ergebnis optimiert werden. Das Hauptproblem wird bei den derzeit üblichen Verfahren als die Überforderung des Nutzers durch eine zu große Informationsmenge angesehen.

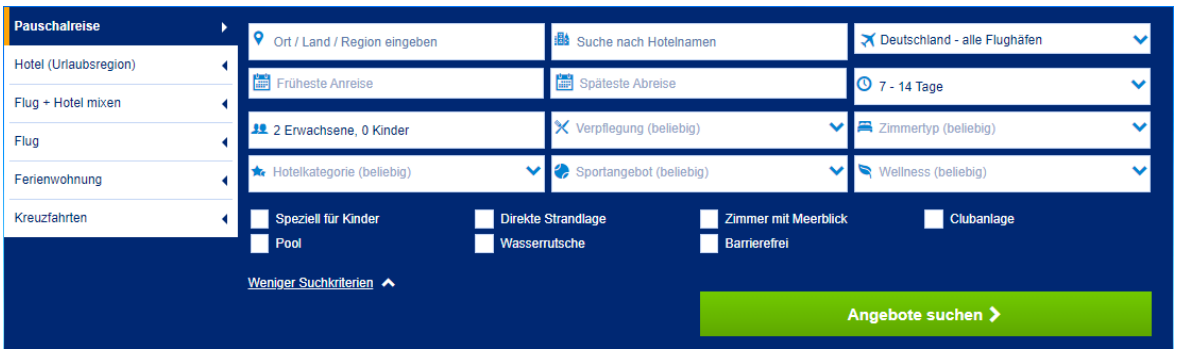


Abbildung 27 - Suchmaske für Pauschalreiseangebote auf der Plattform „Ab In den Urlaub“

Ein Nutzer setzt bei der Reiseplanung meist eine Vielzahl an Filtern zur Auswahl eines Reiseziels. Wie in Abbildung 27 dargestellt, müssen hierzu nicht nur Reiseziel, sondern auch Reisezeitpunkt, -teilnehmer, -dauer, etc. bekannt sein. Werden viele

Felder freigelassen, und z.B. nur als Abreiseflughafen „Deutschland - alle Flughäfen“, Reisedauer „7 - 14 Tage“ und Reiseteilnehmer „2 Erwachsene, 0 Kinder“ angegeben, wird eine Vielzahl von Reisezielen dargestellt (siehe Abbildung 28).

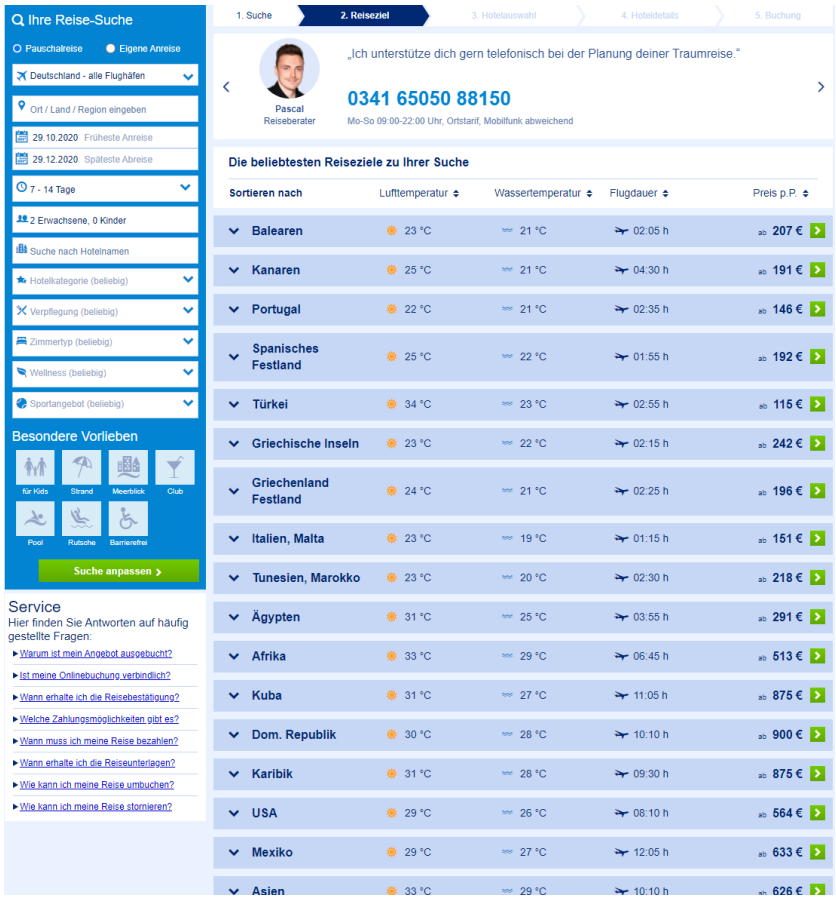


Abbildung 28 - Darstellung von Reisezielen auf „Ab in den Urlaub“ bei fehlender Präzisierung

Diese können im Nachgang typischerweise durch einen Filteralgorithmus weiter eingegrenzt werden. Erst durch eine genau Spezifikation seiner Reisewünsche wird die Informationsmenge für den Nutzer zu einer verarbeitbaren Menge reduziert, sodass ihm eine Auswahl an Reisezielen präsentiert werden kann, die vermeintlich geeignet ist.

Vereinfacht gesagt muss ein Nutzer bereits bei der Angebotsauswahl ein genaues Verständnis seiner Reiseabsichten besitzen, um verwertbare Angebote zu erhalten. Als Alternative ist bspw. eine persönliche Unterstützung durch Reiseberater oder -büros möglich, wie oben in Abbildung 28 dargestellt.

Als Entwicklungsziele werden grundsätzliche Ansätze zur Optimierung festgelegt. Zu diesen Optimierungsansätzen wird jeweils eine Vermutung und eine Folgerung konstruiert. Die Optimierungsansätze sollen im Rahmen dieser Thesis realisiert, Vermutungen bestätigt bzw. widerlegt und die daraus entstehenden Folgerungen geprüft werden. Die Tupel bestehend aus Optimierungsansätzen, deren Vermutungen und Folgerungen sind in der folgenden Tabelle 3 dargestellt.

Nr.	**Optimierungsansätze (O), Vermutungen und Folgerungen**
001	**Optimierungsansatz:** Nutzung von bestehenden Informationen zu Reisezielen zur Auswertung von Reisezieleigenschaften. **Vermutung:** Realistische qualitative und quantitative Auswertung durch Verwendung von Nutzerbewertungen. **Folgerung:** Schaffung einer Datenbasis als Fundament für alle folgenden Schritte.
002	**Optimierungsansatz:** Kategorisierung gleichartiger Reisezielattribute (z.B. Wellnessreisen, Aktivreisen, Kulturreisen). **Vermutung:** Vereinfachung der Reiseauswahl durch Zusammenfassen gleichartiger Eigenschaften als Kategorien. **Folgerung:** ▪ Effektive Reiseplanung durch einfache Anwendung ▪ Verringerung des Aufwands in der Entwicklung
003	**Optimierungsansatz:** Reisezielempfehlung gänzlich ohne detaillierte Spezifikation von Reiseattributen. **Vermutung:** Vereinfachung einer Reisezielempfehlung durch Einsparung einer detaillierten Spezifikation. **Folgerung:** ▪ Effektive Reiseplanung insb. für unentschlossene Nutzer ▪ Verringerung des Aufwands in der Entwicklung, Wartung wie auch in der Anwendung selbst ▪ Zeit- und Kostenersparnis sowie Verbesserung der Usability
004	**Optimierungsansatz:** Bildbasierte Auswahl anstatt textuelle, filterbasierter Auswahl.

	Vermutung: Möglichkeit der Nutzung von (unterbewussten) visuellen Assoziationen bzw. Emotionen, die in einer textuellen Auswahl nicht abzubilden sind. **Folgerung:** Möglichkeit für Nutzer eine detaillierte Reisezielauswahl im ersten Schritt einer Reiseplanung zu vermeiden.
005	**Optimierungsansatz:** Darstellung der Auswahlergebnisse sowie der Übereinstimmungsquote mit den Klassifikationsresultaten *(Abhängig von 004)*. **Vermutung:** Schaffung von Akzeptanz gegenüber einer maschinellen Entscheidung. **Folgerung:** Gegenüberstellung mit der intuitiven Vorstellung eines gewünschten Reiseziels unter Einbeziehung der vorbestimmten menschlichen Auffassung.
006	**Optimierungsansatz:** Angebot alternativer Reisezielempfehlungen basierend auf der Übereinstimmungsquote *(Abhängig von 004 und 005)*. **Vermutung:** Falls das Reiseziel mit der höchsten Übereinstimmungsquote dem Nutzer nicht zusagt, bieten die weitere Resultate eine alternative Auswahl. **Folgerung:** ▪ Vermeidung eines gänzlich neuen Durchlaufs der bildbasierten Auswahl ▪ Schaffung einer Möglichkeit zur Abwägung für den Nutzer ▪ Reduktion der Fehlerrelevanz bei suboptimalen Bildklassifikationen

Tabelle 3 - Katalog aus Optimierungsansätzen, Vermutungen und Folgerungen

Im Rahmen der Entwicklungsdokumentation wird jeweils zum Kapitelabschluss auf die Tupel aus Tabelle 3 über ihre zugehörige Nummerierung (z.B. O-001 für Optimierungsansatz 001) Bezug genommen.

3.2 Data Understanding

Für die initiale Datenakquise wird eine Quelle benötigt, die aktuelle Informationen zu Reisezielen beinhaltet. Reiseanbieter oder -agenturen, die sich nur auf eine Teilmenge möglicher Ziele beziehen, werden ausgeschlossen, da dies den Lösungsraum bereits bei der Akquise einschränken würde. Touristikplattformen, die eine große Anzahl von Nutzer-Erfahrungsberichten verwalten, werden bevorzugt, da für eine Auswertung eine signifikante Anzahl von Erfahrungsberichten benötigt

wird. Nach einer ersten Analyse von möglichen Touristikplattformen werden „Booking.com“, „Expedia“ und „TripAdvisor“ auf Grund ihrer hohen Verbreitung und Marktrelevanz ausgewählt [85, 86]. Bevor eine tiefergehende Analyse durchgeführt wird, erfolgt eine explorative Analyse der Funktions- und Informationsumfänge. Ziel ist es hierbei herauszufinden, welche Informationen über Reiseziele vorgehalten werden und insbesondere mit welcher Frequenz und Qualität Attribute über Reiseziele gepflegt werden.

Als einheitlicher Test werden auf allen Plattformen allgemeine Anfragen zu Hotelaufenthalten in Stuttgart durchgeführt [87–89]. Die Anfrage erfolgt am 14.08.2020 und bezieht sich auf eine einmalige Übernachtung vom 15.- zum 16.8.2020 für eine Person. Die Anzahl der verfügbaren Hotels ist bei allen Anbietern relativ ähnlich („Booking.com“: 295, „Expedia“: 243, „TripAdvisor“: 252). „Expedia“ zeigt jedoch nicht ausschließlich Hotels innerhalb der Stadtgrenzen von Stuttgart, sondern auch in näherer Umgebung an, obwohl dies nicht spezifiziert wird [89]. Dies wird anhand von weiteren Testdurchläufen (München, Hamburg) verifiziert (siehe Abbildung 53 im Anhang) [90, 91]. Die Ergebnisqualität wird daher als ungenügend eingestuft, da keine trennscharfe Abgrenzung auf ein Reiseziel erfolgen kann. Expedia wird daher ausgeschlossen. Durch die weitere Analyse stellt sich heraus, dass sowohl „Booking.com“ als auch „TripAdvisor“ umfangreiche Informationen zu Hotels verwalten und über eine Vielzahl an Nutzerbewertungen verfügen (siehe Abbildung 54 im Anhang) [92, 93]. Somit werden beide Anbieter als geeignet eingestuft. Bei der eingehenden Analyse der Hotelbewertungen fällt jedoch auf, dass diese Nutzerbewertungen sich stark auf die Bewertung des Hotels selbst und nur selten generell auf die Reiseziele beziehen (siehe Abbildung 55 im Anhang) [92, 93]. Somit wird gefolgert, dass basierend auf Hotelbewertungen keine allgemeingültigen Aussagen über Reisezielen getroffen werden können.

Als Alternative wird die Auswertung von sog. Points of Interests[22] (POIs) in Betracht gezogen, da zu vermuten ist, dass diese die Aktivitäten eines Reiseziels grundsätzlicher abbilden. „Booking.com“ bietet Aktivitäten nicht zu allen Reisezielen an, sondern hebt nur bestimmte Reiseziele hervor [94]. Informationen zu Aktivitäten, welche nicht durch „Booking.com“ aktiv hervorgehoben werden, sind nur in Verbindung mit einer Hotelabfrage abrufbar. Dies erschwert eine Auswertung und führt außerdem zu Redundanzen, wenn bspw. viele Hotels in der Nähe eines POIs liegen [93]. Weiterhin verfügt „Booking.com“ über keine detaillierten Informationen zu POIs, sondern nur allgemeine Parameter, wie bspw. die Bezeichnung oder Distanz zu einem POI [93]. Auf Basis dieser Einschränkungen wird „Booking.com“ ausgeschlossen.

Der dritte Anbieter „TripAdvisor“ verfügt über ebendiese Voraussetzungen. POIs verfügen über detaillierte Attribute, Informationen sowie Nutzerbewertungen und können für jedes Ziel abgerufen werden [95, 96]. Auf Basis der Analyseergebnisse wird „TripAdvisor“ ausgewählt und Möglichkeiten zur automatisierten Datenabfrage eruiert. „TripAdvisor“ stellt eine Programmierschnittstelle (API) zur Verfügung [97], welche jedoch primär auf den kommerziellen Einsatz durch Drittanbieter, z.B. zur Hotelbuchung, abzielt [98]. Auswertungen im Rahmen von Data Mining Zwecken, bzw. die Akquise von Massendaten sind untersagt und werden aktiv unterbunden [98]. Die Datenakquise ist somit nicht unmittelbar über eine Programmierschnittstelle möglich, sondern erfolgt mittels Data Scraping[23].

Da die Datenstruktur der auf „TripAdvisor“ gehaltenen Daten ständig Aktualisierungen und Änderungen unterliegt, ist zur Sicherstellung der korrekten Funktionsweise eine aktuelle Data Scraping Lösung zu verwenden. In diesem Fall wird der „tripadvisor-scraper“ von Maximilian Copelli (maxCopell) verwendet [99]. Die letzte Aktualisierung von Ende September 2020 ermöglicht die Abfrage auf die

[22] Siehe Glossar.

[23] Siehe Glossar.

aktuelle Version der „TripAdvisor“ Plattform. Die Web Scraping und Automatisierungsplattform Apify wird zur Datenabfrage verwendet [100]. Hierbei wird zusätzlich ein Proxy verwendet, der verhindert, dass Anfragen auf „TripAdvisor“ bei einer zu großen Anzahl blockiert werden [99, 100]. Um zu garantieren, dass nur aktuelle POIs abgefragt werden, wird der letzte Aktualisierungszeitpunkt einer Rezension auf den 01.01.2019 festgelegt. So wird die Abfrage geschlossener oder nicht mehr verfügbarer POIs vermieden (siehe Quellcode 1).

```
{
  „locationFullName“: „Stuttgart“,
  „lastReviewDate“: „2019-01-01“,
  „includeRestaurants“: false,
  „includeAttractions“: true,
  „includeHotels“: false,
  „includeReviews“: false,
  „proxyConfiguration“: {
    „useApifyProxy“: true
  }
}
```

Quellcode 1 - API Parameter für eine Anfrage zu Stuttgart mittels „tripadvisor-scraper“

Eine Anfrage für Stuttgart liefert 402 POIs zurück. Diese enthalten auf oberster Ebene 45 JSON Properties, welche sich weiter aufgliedern können. Die Darstellung eines Eintrags ist im Anhang unter Quellcode 11 und Abbildung 56 dargestellt. Weiterhin wird festgestellt, dass die o.g. POIs durch die Properties primary_category und subcategory annotiert sind (siehe Quellcode 11 / Abbildung 56 im Anhang). Hieraus wird der Ansatz abgeleitet, diese Kategorien zur späteren Bildklassifikation zu verwenden (siehe O-002). Durch die Datenakquise von POIs können somit deren Informationen zur Auswertung von Reisezieleigenschaften verwendet werden (O-001).

3.3 Data Preparation

Viele der Attribute aus Kapitel 3.2 sind für die Weiterverarbeitung irrelevant (siehe Quellcode 11 und Abbildung 56). Daher erfolgt als erster Schritt der Data Preparation eine Filterung auf die folgenden 9 Properties (siehe Abbildung 29).

Columns: 9	Column Type	Column Index
location_ids	List (Collection of: String)	0
names	List (Collection of: String)	1
latitudes	List (Collection of: String)	2
longitudes	List (Collection of: String)	3
num_reviewss	List (Collection of: String)	4
location_strings	List (Collection of: String)	5
primary_category	List (Collection of: String)	6
raw_rankings	List (Collection of: String)	7
subcategory	List (Collection of: String)	8

Abbildung 29 - Reduktion von 45 auf 9 Properties

Die Properties primary_category und subcategory werden durch Dimensions-reduktion zu primitiven Properties umgewandelt. Dies erfolgt durch eine JSON Path Operation in KNIME (siehe Abbildung 30).

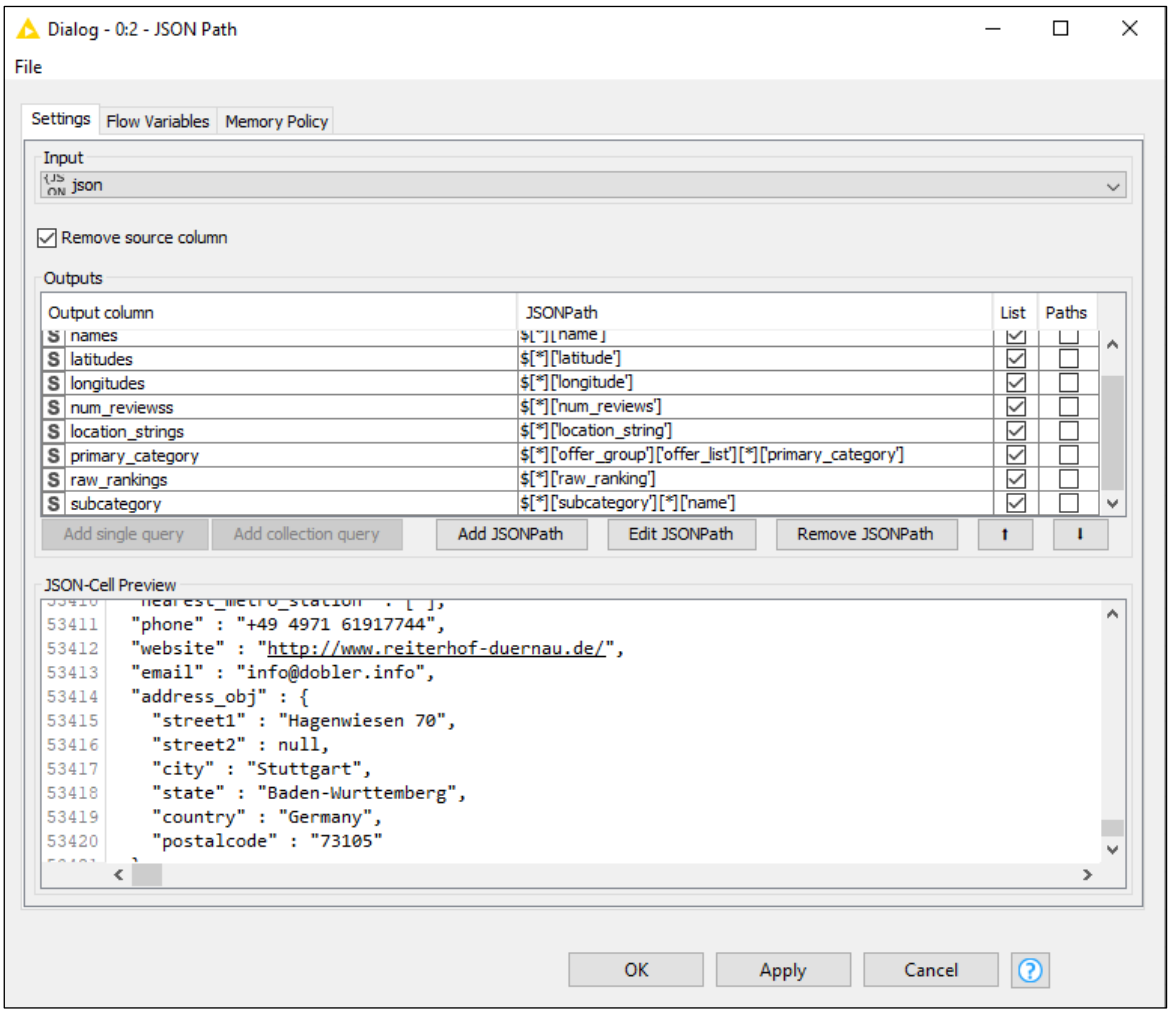


Abbildung 30 - JSON Path Operation in KNIME zur Transformation von Properties

Anschließend werden die Einträge auf Validität überprüft und auf primary_category bzw. subcategory gefiltert. Jeder Eintrag ist somit einer primary_category bzw. einer subcategory zugeordnet (siehe Abbildung 57 im Anhang). Die Einträge aus primary_category stellen dabei eine Untermenge von subcategory Einträgen dar. Dieser Schritt der Datenverarbeitung erfolgt für alle Reiseziele und ist grundsätzlich nicht auf eine bestimmte Anzahl an Reisezielen begrenzt. Im Rahmen dieser Thesis wird jedoch eine Auswahl von 29 globalen Reisezielen getroffen, um vor dem Hintergrund einer effizienten Verarbeitung bei limitiertem Aufwand ein möglichst breites Datenspektrum zu erhalten (siehe Abbildung 31).

Abbildung 31 - Übersicht der 29 globalen Reiseziele (durch rote Marker annotiert) [101]

Die Ergebnisse aller Reiseziele werden zu einer gesamtheitlichen Tabelle zusammengefasst. Hierzu werden die primary_category Properties in ihre elementaren Werte zerlegt, ihre Vorkommnisse gezählt und eine neue Tabelle mit der so entstehenden Anzahl von Werten als Spalten und den Reisezielen als Zeilen erstellt. Es gilt zu beachten, dass Reiseziele, welche zur Tabelle hinzugefügt werden, über noch nicht enthaltene subcategory Properties verfügen können. In diesem Fall werden sie als neue Spalte angefügt. Falls ein Wert bereits in der Tabelle vorhanden ist, erfolgt eine Einsortierung. Zur Realisierung wird über alle Werte iteriert und ein Dictionary aus einzigartigen Kategorien sowie ihrer jeweiligen Anzahl pro Reiseziel erstellt (siehe Quellcode 12 bis Quellcode 15 im Anhang). Die o.g. Funktionen produzieren schlussendlich eine CSV Datei mit 29 Reisezielen und insgesamt 193 unterschiedlichen Kategorien (siehe Tabelle 4).

Destination	skip-the-line tours	lunch cruises	dining experiences	pottery classes	shark diving	outdoor activities	airboat tours	photography tours	sights & landmarks	...
amsterdam	0	0	18	0	0	277	0	18	35	...
berlin	3	0	7	0	0	217	0	11	38	...
buenosaires	0	0	47	0	0	200	0	15	30	...
capetown	0	0	16	0	1	257	0	10	4	...

dubai	0	0	102	0	0	1409	0	16	113	...
dublin	0	0	4	0	0	114	0	4	40	...
hawaii	0	0	1	0	0	419	0	5	54	...
helsinki	0	0	1	0	0	81	0	4	16	...
istanbul	0	2	46	2	0	554	0	24	602	...
lasvegas	2	0	3	0	0	209	0	10	15	...
london	3	0	32	0	0	297	0	168	549	...
macau	0	0	3	0	0	9	0	2	103	...
madrid	3	0	24	0	0	163	0	20	31	...
maldives	0	0	1	0	0	220	0	8	3	...
mallorca	1	2	12	0	0	672	0	8	15	...
miami	0	0	1	0	0	219	3	2	0	...
moscow	0	0	6	0	0	604	0	26	2178	...
newyork	0	0	33	0	0	688	0	126	592	...
peking	1	0	18	0	0	99	0	5	246	...
prague	4	2	34	0	0	384	0	60	628	...
reykjavik	0	0	1	0	0	150	0	6	13	...
rio	2	1	6	0	0	376	0	11	27	...
rome	104	0	196	4	0	683	0	92	1884	...
sanfrancisco	1	0	6	0	0	173	0	9	18	...
santorin	0	0	7	0	0	75	0	17	0	...
seoul	0	0	18	0	0	112	0	10	45	...
singapore	0	0	15	2	0	205	0	11	31	...
stuttgart	0	0	2	0	0	16	0	0	69	...
sydney	0	11	6	1	0	279	0	26	12	...

Tabelle 4 - Reiseziele mit Kategorien und ihren jeweiligen Werteausprägungen (Auszug)

Die Wahrscheinlichkeitsverteilung dieser Kategorien kann zum besseren Verständnis von Problemen der Datenqualität bzw. zur Feststellung von Artefakten in den Daten visualisiert werden (siehe Abbildung 32). Hierzu wird ein Balkendiagamm und eine Wordcloud Darstellungsart verwendet (siehe Quellcode 15 bis Quellcode 17 im Anhang).

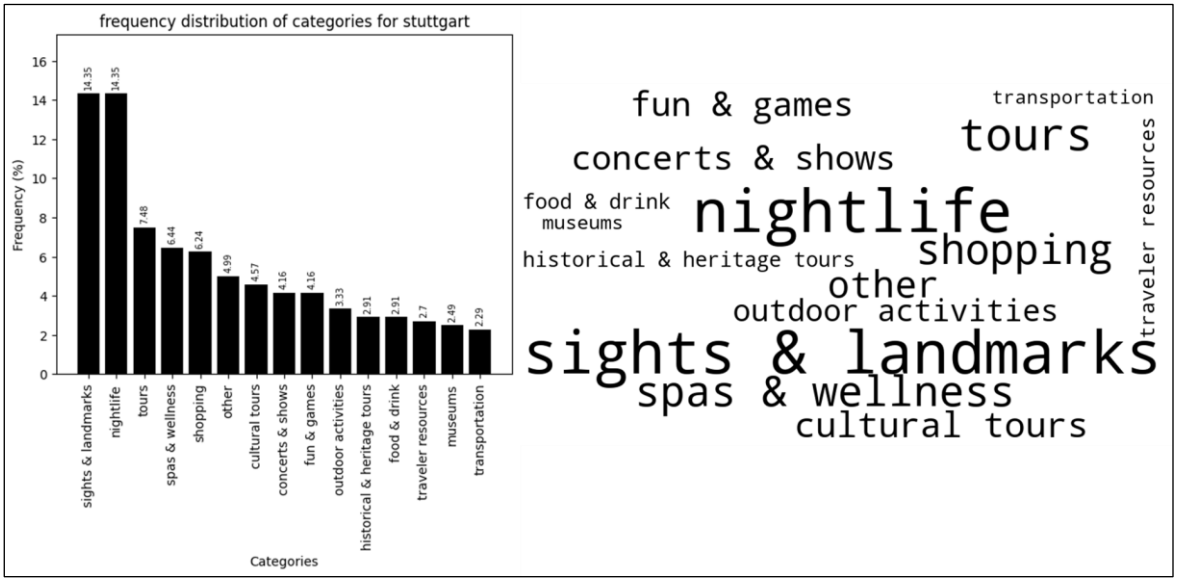


Abbildung 32 - Darstellung der Wahrscheinlichkeitsverteilung von Kategorien am Beispiel Stuttgart

Die Kategorie „sights & landmarks“ mit ca. 14.35% in der Verteilung führend. Außerdem ist zu erkennen, dass 3 der 15 Kategorien den Begriff „tours“ enthalten („tours“, „historical & heritage tours“, „cultural tours“). Weiterhin ist zu entnehmen, dass die Kategorien häufig aus Begriffskombinationen, wie „food & drink“ oder „spas & wellness“ bestehen oder keine weiteren Schlüsse zulassen (z.B. „other“). Durch eine Einsicht in die Wahrscheinlichkeits-verteilungen weiterer Reiseziele kann dies bestätigt werden. Eine detaillierte Evaluation dieser Probleme erfolgt in Kapitel 4. In Abbildung 33 ist der Stand der Entwicklung zum Abschluss der Data Preparation abgebildet.

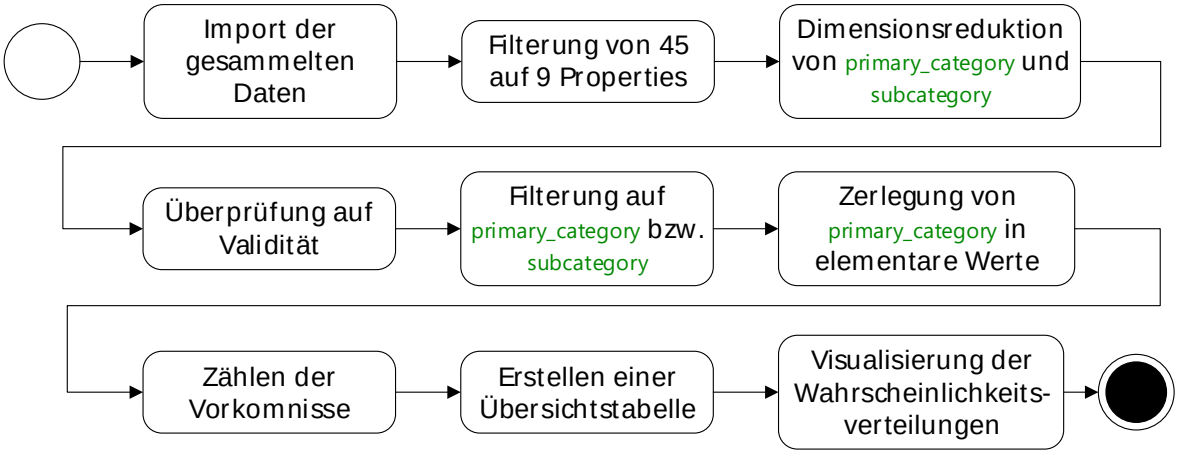


Abbildung 33 - Übersicht über Phase der Data Preparation

Die Schritte der Data Preparation lasse sich wie folgt zusammenfassen: Nach Durchführung der Datenakquise werden die gesammelten Daten (POI Data) mittels eines KNIME Prozesses weiterverarbeitet. Anschließend werden die Ausprägungen der jeweiligen Kategorien in eine Gesamttabelle überführt (siehe Optimierungsansätze: O-001, O-002). Zum besseren Verständnis werden auf dieser Basis statistische Berechnungen durchgeführt, welche mittels Balkendiagramm und Wordcloud visualisiert und analysiert werden.

3.4 Modeling

Der Abschnitt Modeling wird aus Gründen besserer Übersicht und Lesbarkeit in drei Abschnitte unterteilt. In Kapitel 3.4.1 wird die Datenverarbeitung von Reisezieldaten beschrieben (vorbereitet in Kapitel 3.3). Kapitel 3.4.2 enthält daraufhin den Prozess der Bildklassifikation mittels eines CNN. An dieses Kapitel schließt sich Kapitel 3.4.3 an, welches die Entwicklung des Frontends zur Realisierung eines dualen Bildvergleichs beinhaltet.

3.4.1 Datenverarbeitung von Reisezieldaten

Wie in Kapitel 3.3 bereits thematisiert, existieren einige Kategorien mit mangelnder Aussagekraft. Mittels einer allgemeinen Untersuchung zur Entropie aller Kategorien erfolgt eine tiefergehende Analyse. Die Ermittlung der Entropie[24] erfolgt mittels eines binären Regression Trees und im Rahmen eines KNIME Workflow realisiert (siehe Abbildung 34).

[24] Siehe Glossar.

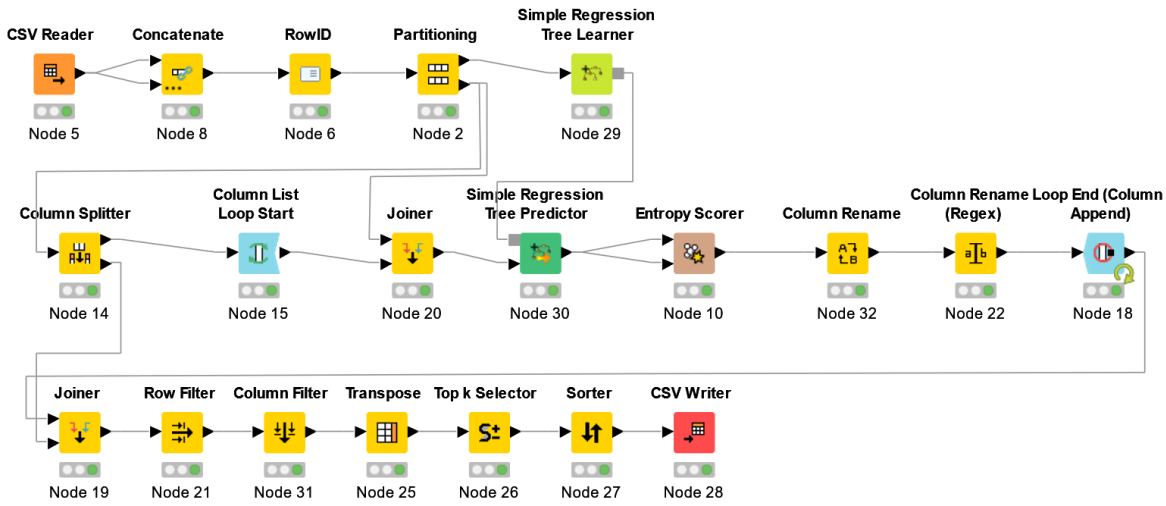


Abbildung 34 - KNIME Workflow zur Ermittlung der Entropie

Anfangs werden die Einträge der Eingangstabelle (results.csv) dupliziert und gleichmäßig in Trainings und Testmenge aufgeteilt (Siehe Knoten CSV Reader, Concatenate, Partitioning in Abbildung 34). Diese Datenmengen werden einem Regression Tree Learner bzw. -Predictor gleichmäßig zur Verarbeitung zugeführt. Anschließend wird für jedes Attribut die Entropie berechnet und (auf 0-1) normalisiert (Siehe Entropy Scorer in Abbildung 34). Zuletzt werden die Ergebnisse nach Entropie geordnet zusammengeführt und auf die 50 aussagekräftigsten Kategorien beschränkt (siehe Knoten Joiner, Row Filter, etc., bis zu CSV Writer in Abbildung 34). Ein Auszug des Endergebnisses ist in der folgenden Abbildung 35 dargestellt und zeigt die Kategorien mit zugehörigen Entropiewerten.

Row ID	D Overall
food & drink	0.755
nature & wildlife	0.754
tours	0.753
spas & wellness	0.753
outdoor activities	0.753
cultural tours	0.753
shopping	0.753
transportation	0.753
ports of call tours	0.752
other	0.749
boat tours & water sports	0.749
nightlife	0.746
historical & heritage tours	0.746
4wd, atv & off-road tours	0.746
museums	0.743
day trips	0.743
private sightseeing tours	0.743
fun & games	0.74
bus & minivan tours	0.74
classes & workshops	0.74
private transfers	0.74
nature & parks	0.737

Abbildung 35 - Kategorien mit zugehörigen Entropiewerten (Auszug der 22 höchsten Werte)

Nun erfolgt eine manuelle Verarbeitung von Kategorien mit mangelnder semantischer Aussagekraft zur Reisezielprädiktion. Wie in Abbildung 35 dargestellt, besitzen Kategorien wie „other“ zwar eine hohe Entropie, jedoch eine mangelnde semantische Aussagekraft. Um diesem Problem entgegenzuwirken wird ein manueller Filteransatz gewählt, da eine automatisierte Weiterverarbeitung (z.B. mit Clusteringverfahren) einen erheblichen Mehraufwand bedeuten würde. Weiterhin ist die Ergebnisqualität eines unüberwachten Verfahrens ist insbesondere bei den hier stark semantisch voneinander abhängigen Kategorien fraglich. Vereinfacht gesagt müsste ein unüberwachter Lernalgorithmus in der Lage sein, nicht-relevante Kategorien wie „other“ semantisch von Kategorien wie „food & drinks“ oder „nightlife“ abzugrenzen und optimal herauszufiltern. Durch die Auswertung von 29 verschiedenen, globalen Reisezielen kann außerdem vermutet werden, dass ein Großteil der auf „TripAdvisor“ verwendeten Kategorien bereits in den Daten repräsentiert werden. In Folge dessen korreliert eine Erweiterung der Reiseziele

nicht zwingend mit einer Erweiterung der Kategorien mit hohem Informationsgehalt, wodurch eine wiederholte Entropieanalyse von Nöten wäre.

Die 50 aussagekräftigsten Kategorien werden daher manuell in 13 Superkategorien zusammengefasst und gefiltert (siehe Abbildung 36).

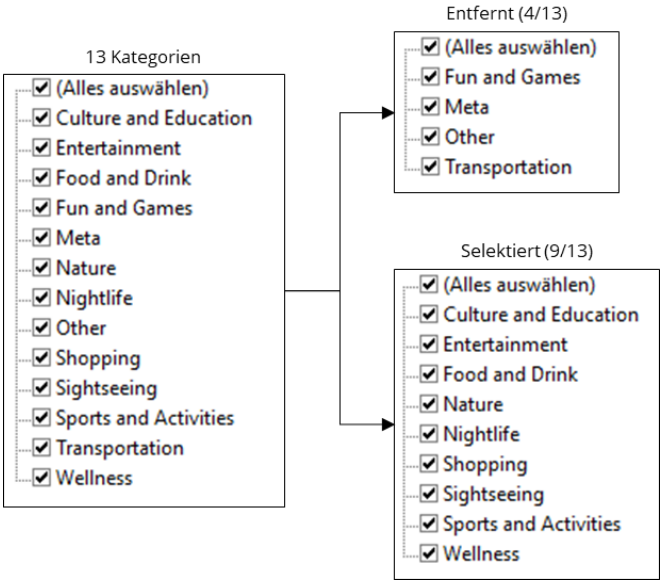


Abbildung 36 - Manuelle Zusammenfassung und Filterung von Superkategorien

Die Superkategorien „Fun and Games“, „Meta“, „Other“ und „Transportation“ werden auf Grund von fehlender Relevanz für die Reisezielprädiktion bzw. fehlende semantische Aussagekraft entfernt. Diese enthalten 23 von 50 Unterkategorien. Die verbleibenden 27 von 50 Unterkategorien werden in die 9 Superkategorien „Culture and Education“, „Entertainment“, „Food and Drink“, „Nature“, „Nightlife“, „Shopping”, „Sightseeing”, „Sports and Activities” und „Wellness” eingeteilt. Auf Basis dieser Superkategorien werden die Daten zusammengefasst. Dies wird durch eine Aggregation der jeweiligen Unterkategorien mittels KNIME Workflow realisiert (siehe Abbildung 59 im Anhang). Das Ergebnis ist eine Tabelle, bestehend aus den 9 Superkategorien und ihren jeweiligen Werteausprägungen pro Reiseziel (siehe Abbildung 37).

Row ID	culture...	enterta...	foodan...	nature	nightlife	shopping	sightse...	sportsa...	wellness
amsterdam	627	47	142	6	294	387	176	490	115
berlin	580	76	108	10	586	272	200	427	324
buenosaires	544	135	253	39	234	232	126	359	155
capetown	310	8	228	90	49	71	123	445	39
dubai	731	22	275	257	150	648	485	3209	449
dublin	372	55	81	20	284	182	62	153	75
hawaii	204	13	20	59	21	177	68	744	102
helsinki	98	9	16	27	78	114	85	119	41
istanbul	1780	185	466	190	697	1028	830	889	391
lasvegas	101	164	68	35	362	161	129	344	181
london	5209	423	546	153	1061	689	1652	591	232
macau	67	16	11	21	14	44	145	16	20
madrid	611	88	286	14	706	282	132	335	238
maldives	7	1	10	8	7	10	4	450	24
mallorca	181	24	116	50	400	129	38	1137	117
miami	50	10	39	7	129	38	12	386	88
moscow	2280	468	299	331	904	847	2482	892	708
newyork	3207	767	702	224	1486	2394	1534	1202	2016
peking	835	44	97	168	51	165	586	164	45
prague	1691	225	417	153	1128	789	888	716	272
reykjavik	77	6	23	39	28	76	90	282	21
rio	517	56	54	167	174	97	187	682	130
rome	8589	250	2095	188	817	1243	2916	1391	899
sanfrancisco	296	63	199	49	310	309	149	335	106
santorin	133	2	153	17	14	55	73	156	11
seoul	515	59	166	61	109	400	131	160	291
singapore	476	23	209	50	176	394	218	371	450
stuttgart	48	20	19	12	69	30	69	24	31
sydney	170	31	161	55	232	170	149	511	321

Abbildung 37 - Werteausprägungen der 29 Reiseziele für 9 Superkategorien

Zu diesem Zeitpunkt entsprechen die Werte der Anzahl POIs pro Kategorie und Reiseziel. Zur Transformation in eine einheitliche Skala werden die Werte zuerst auf (0 - 1) normalisiert und anschließend in Einheitsvektoren transformiert. Der abschließende Tabellenexport ist in Abbildung 38 dargestellt.

row ID	Culture...	Enterta...	Food a...	Nature	Nightlife	Shopping	Sightse...	Sports ...	Wellness
amsterdam	0.224	0.186	0.197	0	0.603	0.491	0.183	0.461	0.161
berlin	0.14	0.205	0.099	0.026	0.821	0.231	0.141	0.27	0.328
buenosaires	0.189	0.53	0.353	0.307	0.465	0.282	0.127	0.325	0.217
capetown	0.111	0.029	0.33	0.816	0.09	0.081	0.129	0.424	0.044
dubai	0.063	0.021	0.095	0.579	0.073	0.201	0.124	0.75	0.164
dublin	0.184	0.305	0.147	0.187	0.811	0.312	0.086	0.186	0.138
hawaii	0.078	0.053	0.016	0.553	0.032	0.238	0.075	0.773	0.154
helsinki	0.103	0.101	0.028	0.626	0.465	0.423	0.27	0.313	0.145
istanbul	0.201	0.233	0.212	0.55	0.453	0.415	0.276	0.266	0.184
lasvegas	0.03	0.578	0.076	0.243	0.652	0.172	0.117	0.279	0.23
london	0.44	0.4	0.187	0.329	0.518	0.207	0.411	0.131	0.08
macau	0.097	0.273	0.007	0.643	0.066	0.199	0.675	0	0.063
madrid	0.129	0.208	0.243	0.045	0.867	0.209	0.081	0.183	0.208
maldives	0	0	0	0.045	0	0	0	0.998	0.048
mallorca	0.043	0.064	0.108	0.288	0.565	0.106	0.025	0.746	0.112
miami	0.034	0.079	0.093	0.021	0.553	0.079	0.018	0.777	0.258
moscow	0.156	0.36	0.082	0.59	0.358	0.207	0.502	0.162	0.205
newyork	0.165	0.442	0.147	0.297	0.442	0.442	0.232	0.164	0.442
peking	0.173	0.101	0.075	0.895	0.053	0.117	0.359	0.083	0.03
prague	0.179	0.266	0.178	0.412	0.69	0.298	0.277	0.2	0.119
reykjavik	0.059	0.047	0.045	0.732	0.102	0.199	0.213	0.6	0.036
rio	0.105	0.127	0.037	0.876	0.2	0.065	0.111	0.369	0.105
rome	0.478	0.156	0.478	0.268	0.262	0.247	0.478	0.206	0.212
sanfrancisco	0.104	0.249	0.279	0.407	0.63	0.386	0.153	0.307	0.146
santorin	0.155	0.014	0.726	0.358	0.05	0.2	0.251	0.464	0
seoul	0.189	0.241	0.238	0.539	0.22	0.521	0.139	0.144	0.445
singapore	0.148	0.078	0.259	0.367	0.31	0.437	0.199	0.302	0.594
stuttgart	0.082	0.424	0.074	0.315	0.716	0.143	0.381	0.043	0.17
sydney	0.058	0.119	0.22	0.459	0.463	0.204	0.152	0.472	0.471

Abbildung 38 - Finales Exportergebnis der Datenverarbeitung von Reisezieldaten

Die Schritte der Datenverarbeitung von Reisezieldaten sind in Abbildung 39 zusammengefasst.

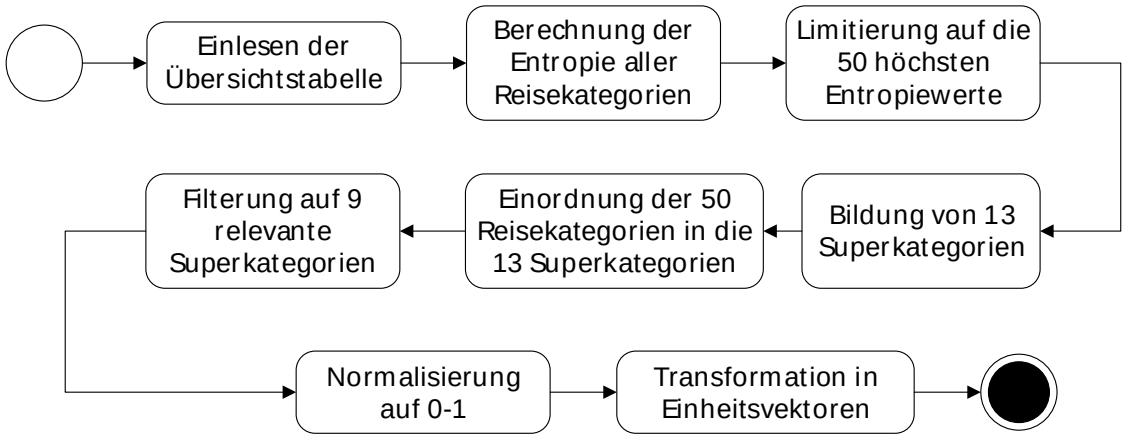


Abbildung 39 - Übersicht über die Datenverarbeitung von Reisezieldaten

Zuerst wird mit Hilfe eines Regression Tree die Entropie aller Kategorien berechnet und auf die 50 höchsten Entropiewerte limitiert. Diese werden anschließend in 13 Superkategorien zusammengefasst, welche folgend auf 9 relevante Superkategorien gefiltert werden. Diese enthalten nach wie vor die originalen Zahlenwerte und werden daher zuerst auf (0-1) normalisiert und anschließend in Einheitsvektoren transformiert. Die Repräsentation als normalisierte Vektoren wird zur Darstellung im Frontend genutzt und in Kapitel 3.4.3 weiter beschrieben.

3.4.2 CNN zur Bildklassifikation

Die Optimierungsansätze O-003 und O-004 (siehe Tabelle 3) formulieren eine bildbasierte Auswahl und eine Reisezielempfehlung ohne Spezifikation von Reiseattributen. Dies wird mittels eines CNN zur Bildklassifikation realisiert. Die Grundlagen dafür werden in Kapitel 2.3.2.1.1 beschrieben.

Zuerst werden geeignete Bilder für die Klassifikation akquiriert. Hierzu wird auf Basis vorangegangener Entwicklungserfahrung die frei verfügbare API der Foto-Community „Flickr“ verwendet [102]. Einstiegs wird eine explorative Analyse geeigneter Bilder durchgeführt. Hierzu wird eine Suche nach den Kategorien (z.B. „Nightlife“) auf Basis der „Flickr“-Bildnamen und -Tags durchgeführt. Es werden je

Kategorie die relevantesten[25] 25 Bilder exportiert und manuell auf ihre Eignung geprüft. Es wird festgestellt, dass in weitläufigen Kategorien (z.B. „Nightlife“, „Culture and Education“, oder „Wellness“) die Bedeutsamkeit der Bilder ungenügend ist. Beispiele sind in Abbildung 40 dargestellt.

Abbildung 40 - Beispiele für fehlende Bedeutsamkeit der Bilder („Nightlife“ links, „Culture and Education“ mittig, „Wellness“ rechts) [103–105]

Um die Bedeutsamkeit der Bilder zu maximieren, werden die Kategorien manuell mit zusätzlichen Tags ergänzt. Die finale Kombination der Abfragebegriffe ist in Quellcode 2 abgebildet.

```
1.  categories = ["Culture and Education", "Entertainment", "Food and Drink", "Nature", "Night-
    life", "Shopping",
2.              "Sightseeing", "Sports and Activities", "Wellness"]
3.  categories_tags = {
4.                  categories[0]: "cultural, Education, Museum",
5.                  categories[1]: "Show, Concert, Entertainment",
6.                  categories[2]: "Food, Drinks",
7.                  categories[3]: "Nature, Landscape, Environment",
8.                  categories[4]: "Nightlife, Party",
9.                  categories[5]: "Store, Shop, Purchase, Retail",
10.                 categories[6]: "Sights, Sightseeing, Monuments, Landmarks",
11.                 categories[7]: "Sport, Activities, Fitness",
12.                 categories[8]: „Spa, Wellness“
13. }
```

Quellcode 2 - Abfragebegriffe zur Bildakquise auf „Flickr“

[25] Flickr verwaltet die Relevanz von Bildern, die im Rahmen dieser Abfrage genutzt wird.

Mit Hilfe dieser Vorgehensweise kann die Bedeutsamkeit der Bilder für jede Kategorie erhöht werden. Nach erfolgreicher Prüfung werden statt 25 nun 1000 Bilder pro Kategorie akquiriert (siehe Quellcode 18).

Um dem Modell für die Verarbeitung in einem CNN eine eingehende Gewichtung zu übergeben, werden sie entsprechend ihrer Kategorien durch One-Hot Encoding[26] annotiert. Hierzu wird eine binäre Matrix aus Bildnamen und Kategorien aufgebaut, die auf Basis der jeweiligen Zugehörigkeit vollständig gefüllt wird (siehe Tabelle 5).

	Cultureandeducation	**entertainment**	**foodanddrink**	**...**
cultureandeducation_0.jpg	1	0	0	...
...	1	0	0	...
cultureandeducation_999.jpg	1	0	0	...
entertainment_*.jpg	0	1	0	...
foodanddrink_*.jpg	0	0	1	...
...	...	...	...	...

Tabelle 5 - One-Hot Encoding zur Festlegung der Startgewichte (Auszug)

Als Nächstes erfolgt der Aufbau des CNNs. Zuerst werden Informationen über die zu klassifizierenden Kategorien eingelesen. Hierbei wird überprüft, ob Bilder für eine Klassifikation vorhanden sind (siehe Quellcode 3).

```
1.  # Global Params
2.  img_x = 300
3.  img_y = 300
4.  main_folder = pathlib.Path("files/images")
5.
6.  # Check image availability
7.  img_count = len(list(main_folder.rglob("*.jpg")))
8.  print(f"Total .jpg filecount: {img_count}")
9.
10. # Get class name path and convert to an array of category names
11. class_names = [os.path.abspath(name) for name in os.listdir(main_folder)]
12. class_names = [path.rsplit("\\", 1)[-1] for path in class_names]
13. print(class_names)
```

Quellcode 3 - Festlegung globaler Parameter, Überprüfung der Verfügbarkeit von Bildern zur Klassifikation und Einlesen von Kategorien

[26] Siehe Glossar.

Die Bilder werden unmittelbar im Format 300 x 300 Pixel eingelesen, um die Effizienz der späteren Verarbeitung sicherzustellen [57, 59]. Die Bilder werden in ein Array konvertiert (Zeile 7, Quellcode 4) und der Farbraum reduziert (Zeile 8, Quellcode 4). Die Grundlagen hierzu werden ausführlich in Kapitel 2.3.2.1.1 beschrieben.

```
1.  train_image = []
2.  for sub_folder in main_folder.iterdir():
3.      print(f"current subfolder is... {sub_folder}")
4.      for file in sub_folder.iterdir():
5.          print(f"reading in... {file}")
6.          img = image.load_img(file, target_size=(img_x, img_y, 3))
7.          img = image.img_to_array(img)
8.          img = img / 255
9.          train_image.append(img)
10. X = np.array(train_image)
11. print(X.shape)
```

Quellcode 4 - Einlesen der zu klassifizierenden Bilder

Bei korrekter Funktionsweise enthalten die in Quellcode 3 und Quellcode 4 verwendeten Konsolenausgaben Informationen über die Anzahl der Bilder, die zu klassifizierenden Kategorien sowie deren Dimensionen (siehe Quellcode 5).

```
1.  Total .jpg filecount: 9000
2.  Classifiers: ['cultureandeducation', 'entertainment', 'foodanddrink', 'nature', 'nightlife', 'shop-
    ping', 'sightseeing', 'sportsandactivities', 'wellness']
3.  Image Information: (9000, 300, 300, 3)
```

Quellcode 5 - Informationen zu Eingangsdaten des Modells

Zur weiteren Verifikation der o.g. Ausgaben wird automatisch je ein Bild der neun zu klassifizierenden Kategorien angezeigt (siehe Abbildung 41 sowie Quellcode 19 im Anhang).

Abbildung 41 - Beispielsbilder für jede zu klassifizierende Kategorie [106–114]

Nun erfolgt das Einlesen der Startgewichte, die mittels One-Hot Encoding den jeweiligen Bildern zugeordnet sind (siehe Quellcode 6).

```
1.  # Read in one-hot-encoding training data
2.  multi_label_train = pd.read_csv('files/multi_label_train.csv', delimiter=";")
3.  Y = np.array(multi_label_train.drop(['ID'], axis=1))
```

Quellcode 6 - Einlesen der Startgewichte des Modells

Anschließend werden die Bilder in Trainings- und Testmenge aufgeteilt. Hierzu wird ein 80/20 Split (test_size=0.2) verwendet (siehe Quellcode 7).

```
# Split X/Y into train and test data based on test_size
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

Quellcode 7 - Aufteilung der Trainings- und Validierungsmenge

Jetzt erfolgt die eigentliche Definition der Modellschichten. Dies ist der wesentliche Schritt der Modellerstellung. Die Grundlagen hierzu werden in Kapitel 2.3.2.1.1 eingeführt. Im Modell werden vier sequenzielle Blöcke aus Convolutional Layern (Conv2D), MaxPooling Layern (MaxPooling2D) und Dropout Layern (Dropout) verwendet. Am Ende dieser vier Blöcke folgt ein Flatten Layer. Das Modell wird mit drei Dense Layern abgeschlossen.

```
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu", input_shape=(img_x, img_y, 3)))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(Dropout(0.22))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.22))
model.add(Conv2D(filters=256, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(8, 8)))
model.add(Dropout(0.22))
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(16, 16)))
model.add(Dropout(0.22))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(len(class_names), activation='softmax'))
```

Quellcode 8 - CNN Modelldefinition

Jeder Convolutional Layer besitzt Parameter zur Filtergröße, Kernelgröße und der zu nutzenden Aktivierungsfunktion. Der erste Convolutional Layer bildet den Eingabe Layer im Modell, weswegen dieser außerdem die Größe der zu verarbeitenden Bilder als Parameter erhält. Als Aktivierungsfunktion in den Convolutional Layern wird die ReLU Funktion verwendet (siehe 2.3.2.1.2.1). Die kernel_size ist auf eine 3x3 Faltungsmatrix festgelegt. Die Anzahl der Filter (filters) wird in jedem Schritt verdoppelt (64, 128, 256, 512). Hierdurch erfolgt eine schrittweise Verfeinerung der Features. Die Pooling Layer verfügen ebenfalls über einen

Parameter zur Größe der Faltungsmatrix (pool_size), welcher hier zu Beginn mit 1x1 angegeben und dann schrittweise erhöht wird (1x1, 2x2, 8x8, 16x16). Nach jedem Convolutional/Pooling Layer folgt ein Dropout Layer mit einer Dropout Rate von 0,22. Nach dem Flatten Layer, welcher die vorangehenden Layer auf einen einzelnen Vektor transformiert, folgen drei Dense Layer. Zuerst wird ein Dense Layer mit 512 Dimensionen und dann mit 64 Dimensionen verwendet. Beide verwenden die ReLU Aktivierungsfunktion. Schlussendlich erfolgt die Ausgabe durch einen Dense Layer mit 9 Einheiten. Dies entspricht der Anzahl der Klassen (len(class_names)). Als Aktivierungsfunktion wird hier nicht ReLU, sondern Softmax verwendet, da hierdurch eine direkte Ausgabe der Wahrscheinlichkeiten erfolgen kann, welche sich auf den Wert 1 aufsummieren (siehe 2.3.2.1.2.1).

Die Optimierung der Schichten und seiner Parameter wird entlang der Entwicklung und Verschriftlichung dieser Thesis getätigt und umfasst eine Vielzahl an Testdurchläufen. Änderungen des Modells sind stets mit neuen Rechendurchläufen und einer Vielzahl an Tests zur Verifikation der Modellqualität verbunden. Die hier dargestellte Modellarchitektur entspricht der höchsten Modellqualität und damit dem finalen Entwicklungsergebnis. Eine Zusammenfassung der Modelldefinition kann mit dem Befehl model.summary() ausgegeben werden. Sie umfasst die Bezeichnungen der jeweiligen Ebenen sowie deren Dimensionen und Anzahl der Features (siehe Quellcode 20 im Anhang).

Anschließend kann das Modell kompiliert werden. Hierbei werden die Kostenfunktion, Optimierungsfunktion und Metrik angegeben. Als Optimierungsfunktion (optimizer) wird Adam verwendet. Als Kostenfunktion (loss) wird die Binary Cross Entropy verwendet und als Metrik wird Accuracy verwendet (siehe Quellcode 9).

```
1. # Compile model.
2. model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Quellcode 9 - Kompilieren des Modells

Das kompilierte Modell wird anschließend trainiert. Hierzu werden dem Modell die Trainings- und Testdaten übergeben. Die Anzahl der Epochen wird festgelegt und die Anzahl der im Modell parallel zu propagierenden Datensätze (Batches) definiert. Mit Hilfe des Befehls verbose[27] kann außerdem der Umfang der Ausgaben gesteuert werden. Die Anzahl der Epochen ist in dieser Umsetzung als globale Variable definiert und auf 25 festgelegt. Die Batchanzahl ist auf 16 festgelegt. Mit der Ausführung des Befehls model.fit() erfolgt der Trainingsdurchlauf des Modells (siehe Quellcode 10).

```
1. # Train model using epochs from global variables
2. history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test),
   batch_size=batch_size, verbose=1)
```

Quellcode 10 - Training des Modells

Beim Training des Modells werden Ausgaben erzeugt, welche detaillierte Informationen zur Performance des Modells enthalten (siehe Quellcode 21 im Anhang). Die Daten über Fehler (loss) und Genauigkeit (accuracy) auf Trainings- und Testmenge[28] werden zur Analyse der Modellperformance verwendet. Um ein besseres Verständnis über die Modellergebnisse zu erhalten, kann die Performance des Modells visualisiert werden (siehe auch Quellcode 22).

[27] Während der Entwicklung und Modelloptimierung wird verbose=1 verwendet. Zur übersichtlicheren Darstellung der Konsolenausgabe wird verbose=2 verwendet.

[28] Die Testmenge ist durch das Präfix „val" gekennzeichnet.

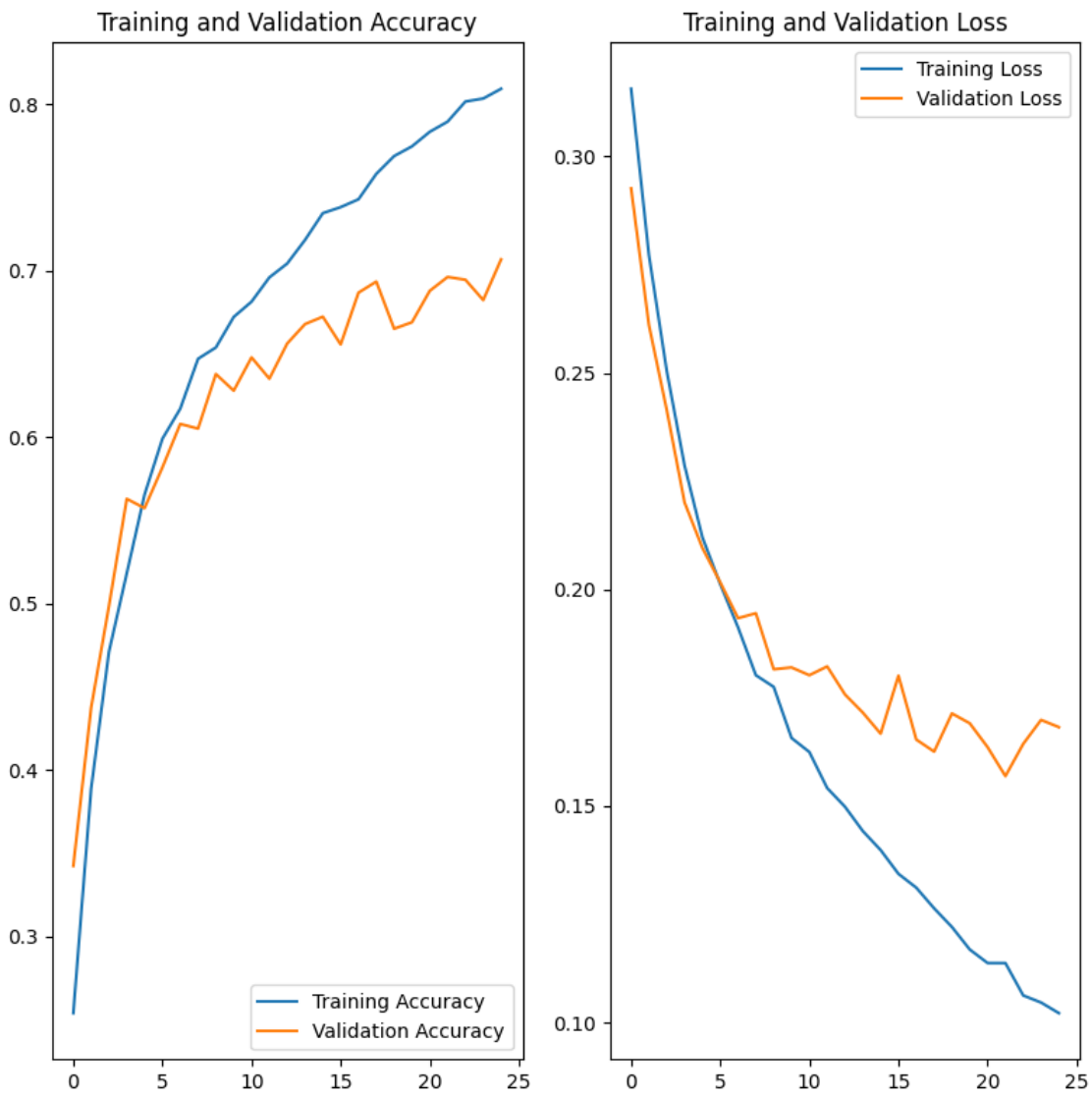


Abbildung 42 - Visualisierung der Modellergebnisse

Wie in Abbildung 42 zu erkennen, erreicht das Modell zum Ende eine Genauigkeit von 80.92% auf der Trainingsmenge und 70.67% auf der Testmenge mit einem Fehler von jeweils 10.22% bzw. 16.82%. Die Genauigkeit auf der Trainingsmenge verharrt ab ca. der 13. Epoche bei 65 - 71%. Der Fehler auf der Trainingsmenge nimmt stetig ab und erreicht am Abschluss des Trainings sein Optimum. Der Fehler auf der Validierungsmenge verringert sich stetig entlang des Fehlers auf der Trainingsmenge, flacht jedoch nach der 6. Epoche ab und verharrt zu Epoche 13 bei ca. 16 - 18%. Zur weiteren Verarbeitung werden die Prädiktionsergebnisse für jedes Bild als JSON gespeichert (siehe Quellcode 23 im Anhang).

Die Ergebnisse dieses relativ einfachen CNN werden als befriedigend bewertet. Optimierungsansätze zur Steigerung der Ergebnisqualität werden in Kapitel 4.1 thematisiert. Abschließend sind die beschriebenen Entwicklungsschritte in Abbildung 43 zusammengefasst.

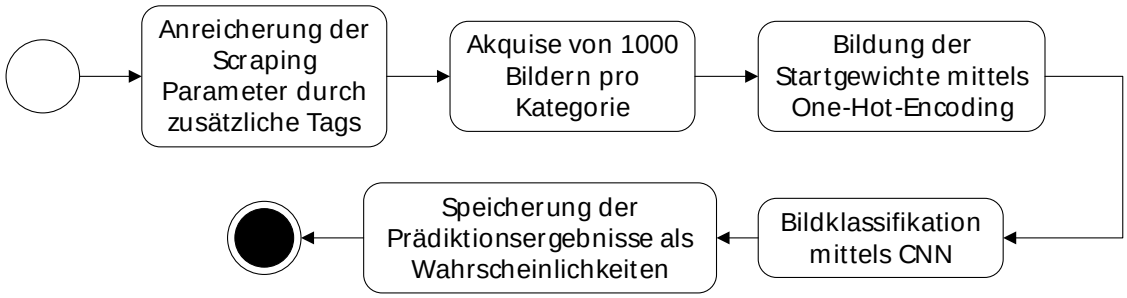


Abbildung 43 - Übersicht über die Schritte zur Bildklassifikation mittels eines CNN

Zur Erhöhung der Datenqualität werden zur Bildakquise die Suchbegriffe einer Abfrage auf „Flickr" durch verschiedene Tags angereichert. Die so akquirierten Bilder werden mit Hilfe mit Hilfe von One-Hot-Encoding startgewichtet. Anschließend werden sie durch ein CNN klassifiziert. Die Ergebnisse werden als Wahrscheinlichkeiten interpretiert und für jedes Bild abgespeichert.

Die im Rahmen dieses Kapitels getätigten Entwicklungsschritte sind den Optimierungsansätzen O-003 und O-004 zuträglich (siehe Tabelle 3, Kapitel 3.1). Die Ergebnisse aus Kapitel 3.4.1 und 3.4.2 werden zur Realisierung eines dualen Bildvergleichs im Frontend kombiniert, welches im folgenden Kapitel 3.4.3 beschrieben wird.

3.4.3 Frontend

Die Bildgegenüberstellung und Darstellung der Ergebnisse wird mit einem web-basierten Frontend durch den Einsatz von Vue.js realisiert. Hierbei werden zwei Ansichten umgesetzt. In der ersten Ansicht werden dem Nutzer zwei Bilder zur Auswahl gegenübergestellt. Möchte der Nutzer keine Auswahl treffen, kann er die Auswahl durch Druck auf einen Skip-Button überspringen und erhält zwei neue Bilder angezeigt (siehe Abbildung 44).

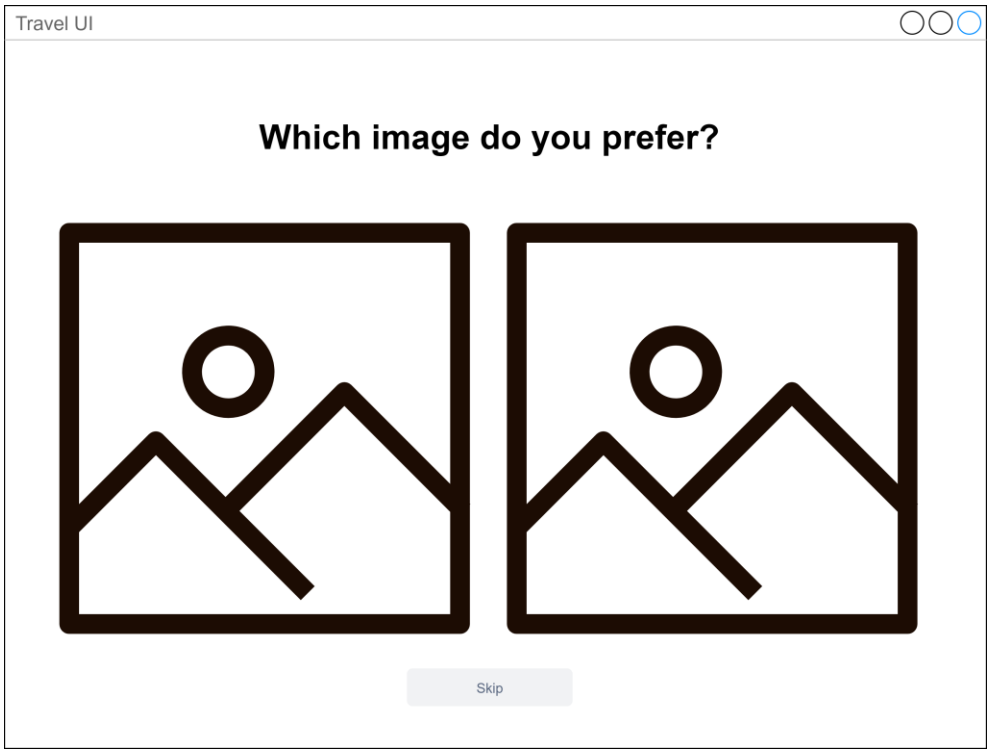


Abbildung 44 - Erste Ansicht: Darstellung des Bildvergleichs im Frontend (Mockup)

Nach eine Reihe von Vergleichen erhält der Nutzer eine Ergebnisdarstellung Die Anzahl der Vergleiche ist variabel und wird in diesem Fall 6 festgelegt.

Abbildung 45 - Zweite Ansicht: Darstellung der Ergebnisse im Frontend (Mockup)

Die Ergebnisdarstellung umfasst neben dem Ergebnis mit der höchsten Übereinstimmungsquote zusätzlich zwei Alternativresultate. Diese entsprechen der

zweit- bzw. drittbesten Übereinstimmungsquote (siehe Abbildung 45). Die Auswahlergebnisse werden zur Auswertung in Kapitel 4.3.1 protokolliert.

Der Nutzer besitzt abschließend die Möglichkeit die Auswahlergebnisse positiv oder negativ zu bewerten (siehe Buttons: „Yes, i agree“ und „No, i don’t agree“ in Abbildung 45). Dies wird zur Messung der Nutzerakzeptanz verwendet (siehe Kapitel 4.3.2). Außerdem besitzt er die Möglichkeit durch einen Klick auf den Button „Reset & Try again“ die Bildauswahl erneut zu beginnen.

Zur Darstellung der Ergebnisse werden repräsentative Bilder verwendet (z.B. Big Ben für London oder das Brandenburger Tor für Berlin). Die Berechnung der Übereinstimmung zwischen den in Kapitel 3.4.1 ermittelten Reisezieldaten und den Modellergebnissen aus Kapitel 3.4.2 erfolgt mittels der euklidischen Distanz[29]. Hierzu werden die jeweilig angesprochenen Prädiktionsergebnisse (z.B. foodanddrinks1.jpg) als Vektoren repräsentiert. In Abbildung 46 ist ein Schritt der Bildgegenüberstellung dargestellt.

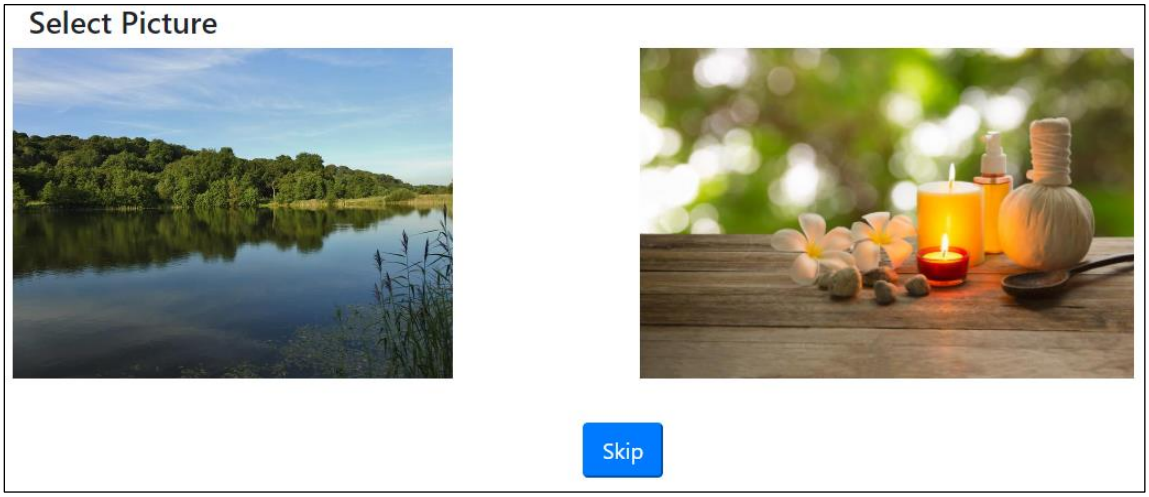


Abbildung 46 - Schritt einer Bildgegenüberstellung

Nach Ablauf der gesamten Bildauswahl werden die Ergebnisse aufsummiert und dem Nutzer angezeigt (siehe Abbildung 47).

[29] Siehe Glossar.

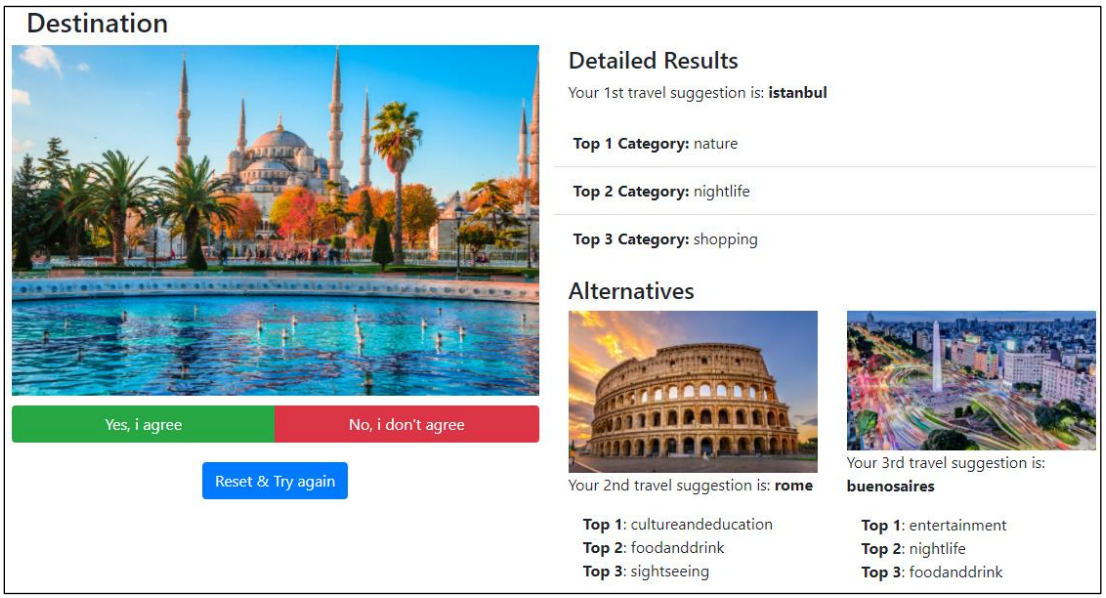


Abbildung 47 - Ergebnisdarstellung im Frontend

Die Schritte zur Darstellung im Frontend lassen sich wie folgt zusammenfassen: Zuerst erfolgt eine manuelle Akquise eines Beispielbilds je Reiseziel zur visuellen Darstellung der Ergebnisse. Anschließend erfolgt die eigentliche duale Bildgegenüberstellung (siehe O-004). Die Ergebnisse werden als euklidische Distanz zwischen den ausgewählten Bildern (in Summe) und dem korrelierenden Modellergebnis interpretiert, über alle ausgewählten Bilder aufsummiert und dem Nutzer angezeigt (siehe O-005). Außerdem erfolgt eine Darstellung alternativer Ergebnisse, wodurch O-006 realisiert wird.

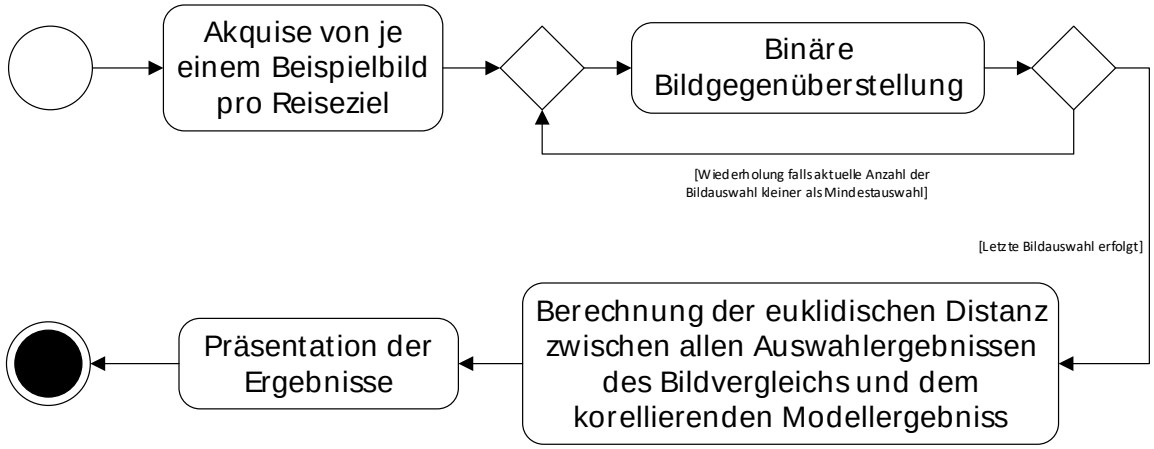


Abbildung 48 - Übersicht über die Frontendentwicklung

4 Evaluation

Die Evaluation gliedert sich in drei Kapitel. Eingehend werden die im Rahmen der Entwicklung auftretenden Probleme diskutiert (siehe 4.1). Auf Basis des allgemeinen Vergleichs zwischen ML und CSP aus Kapitel 2.4 werden anschließend mögliche alternative CSP Lösungsansätze aufgeführt (siehe Kapitel 4.2). Anschließend erfolgt eine Ergebnisevaluation (siehe Kapitel 4.3).

4.1 Evaluation der Entwicklung

In diesem Kapitel werden die bei der Entwicklung auftretenden Probleme definiert und weitergehend beschrieben. Falls keine Problemlösung im Rahmen der Entwicklung gefunden wird, erfolgt die Formulierung möglicher Lösungsansätze.

Problem 001	Die Ausgangsdaten (Kategorien) besitzen eine hohe Wahrscheinlichkeit von semantisch fragwürdigen Attributen (siehe Kapitel 3.2, 3.3).
Problem-beschreibung	Kategorien bestehen häufig aus Begriffskombinationen wie „food & drink“ oder „spas & wellness“. Weiterhin ermöglichen Kategorien z.T. keine weiteren Rückschlüsse zur Klassifikation (z.B. „other“). Diesen Kategorien kann eine mangelnde Aussagekraft unterstellt werden, da sie entweder zu allgemein, oder nichtssagend sind. Anhand der Beispiele New York, Prag, Kapstadt und Helsinki ist dies in Abbildung 49 ersichtlich.

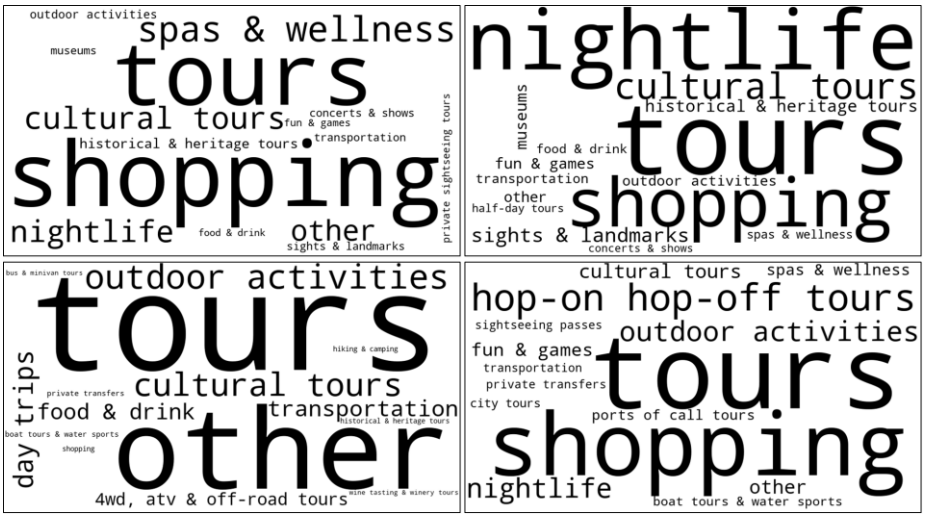


Abbildung 49 - Darstellung der Wahrscheinlichkeitsverteilungen von Kategorien als Wordcloud (oben links: New York, oben rechts: Prag, unten links: Kapstadt, unten rechts: Helsinki)

Erfolgte Lösung	Eine Lösung erfolgt durch die in Kapitel 3.3 eingehend durchgeführte Datenvorverarbeitung sowie die in Kapitel 3.4.1 durchgeführte Weiterverarbeitung.

Tabelle 6 - Problemdefinition, -beschreibung und Lösungsansatz 001

Problem 002	Die quantitative Werteausprägung für Kategorien ist selbst nach Zusammenfassung z.T. mangelhaft, sodass eine Datenasymmetrie entsteht (siehe Kapitel 3.4.1).
Problem-beschreibung	Werden z.B. die Werteausprägungen für „maldives" betrachtet, ist offensichtlich, dass durch die Normalisierung ein Großteil der Werte nahe 0 liegen bzw. 0 sind. Eine Ausnahme stellt die Kategorie „sportsandactivities" dar, bei welcher der höchste Wert konstatiert wird (siehe Tabelle 4). Dieses Problem entsteht durch die Asymmetrie in den Ausgangsdaten, in denen „maldives" eine Summe von 521 POIs aufweist, während bspw. New York 13.532 POIs zählt. Die geringe Informationsmenge der Kategorien in „maldives" führt dementsprechend zu einer asymmetrischen Gewichtung stark ausgeprägter Kategorien. Das Reiseziel „newyork" besitzt 767 POIs der Kategorie „entertainment", was relativ zur Gesamtanzahl der POIs in „newyork" nur 5,7% ausmacht, jedoch im Gesamtvergleich aller Reiseziele den höchsten Wert darstellt. In Folge dessen wird für „entertainment" in „newyork" ein verhältnismäßig hoher Wert von 0,442 durch Normalisierungsoperationen ermittelt (siehe Abbildung 37, Abbildung 38). Eine starke Asymmetrie in der Anzahl der POIs sowie deren Verhältnis untereinander kann verzerrte Ergebnisse hervorrufen, die eine Qualitätsminderung der Ergebnisse zur Folge haben können.
Lösungs-ansatz	Mangelnde Datenqualität und -quantität ist ein Grundsatzproblem der Datenakquise und -verarbeitung. Die verwendete Quelle „TripAdvisor" enthält teils eine mangelhafte Menge an Informationen für bestimmte Reiseziele. Zur Lösung können bspw. zusätzliche Quellen herangezogen werden, um die Daten zu augmentieren. Zur Vermeidung einer Datenasymmetrie könnte zusätzlich bspw. die kleinstmögliche Anzahl verfügbarer Daten für ein Reiseziel als Maßzahl verwendet werden. Auf diese Maßzahl können alle anderen Reiseziele fakturiert werden, sodass die entstehende Asymmetrie aufgehoben wird.

Tabelle 7 - Problemdefinition, -beschreibung und Lösungsansatz 002

Problem 003	Bilder, welche zur Klassifikation verwendet werden, sind nicht aussagekräftig genug, bzw. sind fehlklassifiziert (siehe Kapitel 3.4.2).
Erfolgte Lösung	Durch die zusätzliche Nutzung von Tags bei der Bildersuche wird die Qualität gesteigert.
Erfolgte Lösung	Durch Nutzung eines Skip-Buttons besitzt der Nutzer die Möglichkeit die aktuell präsentierten Bilder zu überspringen.
Lösungs-ansatz	Durch eine manuelle Überprüfung der Bilder können fehlerhafte, bzw. nicht aussagekräftige Bilder aussortiert werden. Dies ist bereits bei 1000 Bildern pro Kategorie jedoch mit großem Aufwand verbunden und wird im Rahmen dieser Thesis daher nicht durchgeführt.

Tabelle 8 - Problemdefinition, -beschreibung und Lösungsansatz 003

Problem 004	Die Modellparametrisierung und -Optimierung ist ein sehr weitläufiges Feld. Es existiert keine allgemeingültige Lösung für jeden Anwendungsfall, welche herausragende Ergebnisse liefert (siehe Kapitel 3.4.2).
Erfolgte Lösung	Die Wahl der Modellparameter und dessen Optimierung muss mit Sorgfalt und Bedacht erfolgen, da sie einen erheblichen Einfluss auf die Modellqualität besitzen. Die Optimierung der Schichten und Parameter dieses Modells wird entlang der Entwicklung und Verschriftlichung dieser Thesis durchgeführt und erstreckt sich über mehrere Wochen. Änderungen des Modells sind mit neuen Rechendurchläufen und einer Vielzahl an Tests zur Verifikation der Modellqualität verbunden.

Tabelle 9 - Problemdefinition, -beschreibung und Lösungsansatz 004

Problem 005	Der Zeitaufwand für das Training von Modellen ist ein erheblicher Faktor sowie stark hardwareabhängig.
Erfolgte Lösung	Moderne Frameworks, wie das hier verwendete Keras, unterstützen die Schnittstellen zur Berechnung auf GPUs (z.B: Nvidia CUDA) [115]. Selbst durch den Einsatz moderne Endbenutzerhardware, wie bspw. spezieller Grafikchips (im Rahmen dieser Thesis wurde eine GEFORCE RTX 2080 SUPER verwendet) ist es möglich die Geschwindigkeit im Vergleich zu CPU-basierten Rechenvorgängen erheblich zu steigern [115]. Die Berechnung einer Modellepoche dauert dennoch ca. 60-90s je nach Komplexität des hier verwendeten CNN, was einen gesamten Modelldurchlauf über 30 Minuten dauern lässt. Eine Steigerung der Modellkomplexität hat eine Verlangsamung zur Folge, wodurch ein erheblicher Aufwand auf die Optimierung von Modellen entfällt.

Tabelle 10 - Problemdefinition, -beschreibung und Lösungsansatz 005

Für die Probleme 001 sowie 003 bis 005 sind Lösungen erfolgt. Einzig Problem 002 besteht z.T. weiterhin. Für die in Problem 002 beschriebene mangelnde Datenqualität bzw. fehlende Aussagekraft der verwendeten Bilder bestehen jedoch Lösungsansätze (siehe Tabelle 7). Durch weitere Optimierung des Modells sowie durch die Verwendung der in Tabelle 6 bis Tabelle 10 aufgeführten Lösungsansätze lässt sich die Modellqualität weiter verbessern.

4.2 Bewertung der Herangehensweisen ML versus CSP

Der in Kapitel 2.4 aufgeführte allgemeine Vergleich zwischen ML- und CSP-basierten Lösungsansätzen wird auf Basis der Entwicklungserfahrungen bewertet.

Hierbei werden relevante Kategorien aus Tabelle 2 verwendet, um eine Bewertung der jeweiligen Lösungsansätze vorzunehmen.

Datenvorverarbeitung: Die Datensätze zur Kategorisierung der Reisebilder werden während des Ladeprozesses automatisch annotiert. Die Anlage und Verwaltung von Variablen, Domänen und Constraints bei einem CSP-basierten Lösungsansatz könnte hierzu analog erfolgen. Die Erweiterung der Daten oder Klassifizierer ist bei einem ML-gestützten Modell relativ einfach, da die bestehende Modellstruktur übernommen werden kann. Werden beispielsweise die Reiseziele erweitert oder die Kategorien ergänzt, erfordert dies zwar einen erneuten Berechnungsdurchlauf, die grundlegende Struktur bliebe dabei aber unberührt. Im Vergleich dazu, wäre bei einer CSP-gestützten Lösung eine Neudefinition nötig, da neue Variablen hinzugefügt, bzw. gelöscht werden und damit automatisch auch Constraints angepasst werden müssen. Der technische Aufwand zur adäquaten Implementierung überstiege vermutlich den Aufwand zur simplen Bildannotation im Rahmen eines ML-basierten Verfahrens.

Laufzeitverhalten: Zum Training des ML-Modells werden ca. 35 Minuten benötigt (siehe Tabelle 10). Die Kategorisierung erfolgt in Echtzeit. Eine Abschätzung über die Dauer bei einer äquivalenten Repräsentation eines CSP-Ansatz lässt sich nicht geben.

Abstraktion: Die Entwicklung des ML-Modells erfolgt in Programmcode. Die Modellkomponenten sowie Algorithmen entstammen dem Keras Framework. Eine Lösung auf Basis von Programmcode könnte ebenso mit einem CSP-Solver Framework (wie z.B. python-constraint) erfolgen [116].

Optimierung: Die Optimierung des Modells erfolgt mittels des Adam Optimierungsverfahrens (siehe 2.3.2.1.4). Eine Optimierung im eigentlichen Sinne erfolgt bei einer CSP-Lösung nicht, da die Domänen bzw. Constraints bei einer Propagierung nicht angepasst werden.

Umsetzung: Die Umsetzung erfolgt in Programmcode unter Zuhilfenahme des Keras Framework. Analog dazu kann auch eine Realisierung eines CSP-Lösungsansatzes in Programmcode erfolgen (siehe Abstraktion). Weiterhin existieren spezielle anwendungsorientierte Programmiersprachen aus dem Bereich der Logikprogrammierung, wie z.B. Prolog [117], welche zur Abbildung und Lösung von CSPs verwendet werden.

Die Entwicklung beweist, dass eine Problemlösung mittels eines ML-basierten Ansatzes theoretisch und praktisch möglich ist. Die gewonnenen Erkenntnisse widerlegen jedoch auch eine Anwendung von CSP-gestützten Lösungsverfahren grundsätzlich nicht. Die Struktur der Entwicklung muss hierzu jedoch auf eine optimale Verarbeitung mittels CSP-gestützten Lösungsverfahren angepasst werden. Um ein fundiertes Urteil über die detaillierte Anwendbarkeit einer CSP-gestützten Lösung zu treffen ist eine tatsächliche Implementierung notwendig, die den Rahmen dieser Thesis übersteigt.

4.3 Auswertung der Ergebnisse

Zur Evaluation der Aussagenqualität des dualen Bildvergleichs werden anonyme Nutzertests durchgeführt. 9 verschiedene Probanden nehmen am Test teil und absolvieren je 3 Testdurchläufe ($n = 27$). Die detaillierten Protokollergebnisse sind im Anhang unter Tabelle 13 zu finden. Diese werden hinsichtlich der folgenden Kernfragen ausgewertet:

- Wie sind die vorgeschlagenen Reiseziele verteilt?
    - Welches Reiseziel wird dem Nutzer am häufigsten vorgeschlagen?
    - Werden Reiseziele dem Nutzer u.U. gar nicht vorgeschlagen?
- Wie ist die Korrelation der vorgeschlagenen Reiseziele zu den Modellergebnissen zu bewerten?
    - Wie stark wirken sich insb. Probleme 002 und 003 aus Kapitel 4.1 aus?
    - Lassen sich aus den Nutzertests weitere Vermutungen bzgl. der Modellergebnisse ableiten?

- Wie hoch ist die Nutzerakzeptanz der Ergebnisse?
    - Wie ist die Aussagekraft der Nutzerakzeptanz zu bewerten?

Die Selektionsergebnisse der Bildauswahl sowie die Nutzerakzeptanz werden anonym protokolliert (siehe Kapitel 3.4.3). Ein Nutzer besitzt die Möglichkeit seine Akzeptanz durch Druck auf die Buttons: „Yes, i agree“ und „No, i don’t agree“ zum Ausdruck zu bringen (siehe Kapitel 3.4.3). Die Nutzerakzeptanz soll damit repräsentieren, wie häufig zumindest ein Reiseziel korrekt in den Ergebnissen abgebildet ist. In den folgenden zwei Kapiteln erfolgt die Auswertung der Reisezielprädiktionen (siehe Kapitel 4.3.1) sowie die Auswertung der Nutzerakzeptanz (siehe Kapitel 4.3.2).

4.3.1 Auswertung der Reisezielprädiktionen

In der folgenden Tabelle 11 ist die Auswertung der Reisezielprädiktionen abgebildet.

city	result_1	result_2	result_3	total	result_1%	total%
istanbul	6	2	5	13	22,22%	16,05%
rome	7	2	2	11	25,93%	13,58%
newyork	3	3	3	9	11,11%	11,11%
london	1	5	2	8	3,70%	9,88%
buenosaires	3	3	0	6	11,11%	7,41%
prague	2	2	2	6	7,41%	7,41%
seoul	0	1	3	4	0,00%	4,94%
singapore	2	1	0	3	7,41%	3,70%
berlin	0	2	1	3	0,00%	3,70%
moscow	0	0	3	3	0,00%	3,70%
rio	0	0	2	2	0,00%	2,47%
sanfrancisco	0	1	1	2	0,00%	2,47%
sydney	0	1	1	2	0,00%	2,47%
amsterdam	0	1	0	1	0,00%	1,23%
dublin	1	0	0	1	3,70%	1,23%
lasvegas	1	0	0	1	3,70%	1,23%
hawaii	1	0	0	1	3,70%	1,23%
macau	0	1	0	1	0,00%	1,23%
mallorca	0	1	0	1	0,00%	1,23%
miami	0	1	0	1	0,00%	1,23%
stuttgart	0	0	1	1	0,00%	1,23%
reykjavik	0	0	1	1	0,00%	1,23%
dubai	0	0	0	0	0,00%	0,00%
santorin	0	0	0	0	0,00%	0,00%
maldives	0	0	0	0	0,00%	0,00%

capetown	0	0	0	0	0,00%	0,00%
helsinki	0	0	0	0	0,00%	0,00%
madrid	0	0	0	0	0,00%	0,00%
peking	0	0	0	0	0,00%	0,00%

$\Sigma = 81$

Tabelle 11 - Auswertung der Reisezielprädiktionen

result_1 beschreibt wie häufig ein Reiseziel mit der höchsten Übereinstimmungsquote angezeigt wird. **result_2** und **result_3** beziehen sich jeweils auf die zweit- bzw. dritthöchste Übereinstimmungsquote. **result_1%** beschreibt das Verhältnis von **result_1** zur Gesamtanzahl der Testdurchläufe ($n = 27$) (bspw.: $istanbul = \frac{6}{27} = 22{,}22\%$). **Total%** beschreibt wie häufig ein Reiseziel dem Nutzer insgesamt vorgeschlagen wird (die Anzahl der Gesamtvorkommen in einer der drei Vorschläge) (bspw.: $istanbul = \frac{6+2+5}{3\times27} = 16{,}05\%$).

Die Auswertung der Nutzertests liefert folgende Ergebnisse: Rom ist mit $25{,}93\%$ das Reiseziel, welches am häufigsten mit der höchsten Übereinstimmungsquote vorgeschlagen wird. Rom ist darüber hinaus insgesamt auch in den Top-3 Vorschlägen mit $13{,}58\%$ vertreten. Istanbul ist das am zweithäufigsten zuerst vorgeschlagene Reiseziel ($22{,}22\%$) und befindet sich am häufigsten in den Top 3 ($16{,}05\%$). Im Gegensatz dazu werden Dubai, Santorini, die Malediven, Kapstadt, Helsinki, Madrid und Peking den Nutzern nicht vorgeschlagen.

Bei genauer Auswertung kann festgehalten werden, dass Istanbul und Rom im Verhältnis zu allen anderen Reisezielen signifikant mit der höchsten Übereinstimmungsquote vorgeschlagen werden. Dies könnte auf die, in Tabelle 7 (siehe Kapitel 4.1) angesprochene, Datenqualität und -quantität sowie auf die Gleichverteilung der Daten zurückzuführen sein (siehe blaue Kennzeichnung in Abbildung 50). Am Beispiel der Malediven (siehe gelbe Kennzeichnung in Abbildung 50) ist dies ebenfalls auf die in Tabelle 7 (Kapitel 4.1) angesprochene Auswirkung von asymmetrischen Ausgangsdaten und den daraus resultierenden fehlenden Informationen zurückzuführen.

S row ID	D Culture...	D Enterta...	D Food a...	D Nature	D Nightlife	D Shopping	D Sightse...	D Sports ...	D Wellness
amsterdam	0.224	0.186	0.197	0	0.603	0.491	0.183	0.461	0.161
berlin	0.14	0.205	0.099	0.026	0.821	0.231	0.141	0.27	0.328
buenosaires	0.189	0.53	0.353	0.307	0.465	0.282	0.127	0.325	0.217
capetown	0.111	0.029	0.33	0.816	0.09	0.081	0.129	0.424	0.044
dubai	0.063	0.021	0.095	0.579	0.073	0.201	0.124	0.75	0.164
dublin	0.184	0.305	0.147	0.187	0.811	0.312	0.086	0.186	0.138
hawaii	0.078	0.053	0.016	0.553	0.032	0.238	0.075	0.773	0.154
helsinki	0.103	0.101	0.028	0.626	0.465	0.423	0.27	0.313	0.145
istanbul	0.201	0.233	0.212	0.55	0.453	0.415	0.276	0.266	0.184
lasvegas	0.03	0.578	0.076	0.243	0.652	0.172	0.117	0.279	0.23
london	0.44	0.4	0.187	0.329	0.518	0.207	0.411	0.131	0.08
macau	0.097	0.273	0.007	0.643	0.066	0.199	0.675	0	0.063
madrid	0.129	0.208	0.243	0.045	0.867	0.209	0.081	0.183	0.208
maldives	0	0	0	0.045	0	0	0	0.998	0.048
mallorca	0.043	0.064	0.108	0.288	0.565	0.106	0.025	0.746	0.112
miami	0.034	0.079	0.093	0.021	0.553	0.079	0.018	0.777	0.258
moscow	0.156	0.36	0.082	0.59	0.358	0.207	0.502	0.162	0.205
newyork	0.165	0.442	0.147	0.297	0.442	0.442	0.232	0.164	0.442
peking	0.173	0.101	0.075	0.895	0.053	0.117	0.359	0.083	0.03
prague	0.179	0.266	0.178	0.412	0.69	0.298	0.277	0.2	0.119
reykjavik	0.059	0.047	0.045	0.732	0.102	0.199	0.213	0.6	0.036
rio	0.105	0.127	0.037	0.876	0.2	0.065	0.111	0.369	0.105
rome	0.478	0.156	0.478	0.268	0.262	0.247	0.478	0.206	0.212
sanfrancisco	0.104	0.249	0.279	0.407	0.63	0.386	0.153	0.307	0.146
santorin	0.155	0.014	0.726	0.358	0.05	0.2	0.251	0.464	0
seoul	0.189	0.241	0.238	0.539	0.22	0.521	0.139	0.144	0.445
singapore	0.148	0.078	0.259	0.367	0.31	0.437	0.199	0.302	0.594
stuttgart	0.082	0.424	0.074	0.315	0.716	0.143	0.381	0.043	0.17
sydney	0.058	0.119	0.22	0.459	0.463	0.204	0.152	0.472	0.471

Abbildung 50 - Hervorgehobene Werte für Istanbul und Rom (blau) sowie für die Malediven (gelb)

Istanbul oder Rom besitzen im Vergleich zu den nicht vorgeschlagenen Reisezielen relativ gleichverteilte Werteausprägungen in allen Reisekategorien („Food and Drinks“, etc.). Hieraus wird gefolgert, dass je häufiger ein Nutzer gleichverteilt Bilder mit einer starken Kategoriezugehörigkeit auswählt, desto eher fällt auch das Ergebnis auf ein Reiseziel, dessen Reisezielkategorien gleichverteilt sind.

4.3.2 Auswertung der Nutzerakzeptanz

Die Nutzerakzeptanz wird mit einer Zustimmungsquote von 70,37% gemessen (siehe Tabelle 12). Dies bedeutet, dass mit 70,37% die Nutzer zumindest eines der drei vorgeschlagenen Reiseziele als attraktiv ansehen.

	TRUE	FALSE
value	19	8
percentage	70,37%	29,63%

Tabelle 12 - Auswertung der Nutzerakzeptanz, ermittelt durch die Nutzerbewertung bei der Ergebnisdarstellung

Die alleinige Betrachtung dieser Ergebnisse ist jedoch nicht ausreichend, um eine definitive Aussage über die Ergebnisqualität zu treffen. Wird bspw. zufällig ohne Zurücklegen aus der Menge aller Reiseziele gezogen, liegt die Wahrscheinlichkeit ein Reiseziel in den Top 3 Vorschlägen angezeigt zu bekommen bei:

$$1 - \left(\frac{n-1}{n} \times \frac{n-2}{n-1} \times \frac{n-3}{n-2}\right) = \frac{3}{n}$$

Formel 7 - Berechnung der Wahrscheinlichkeit eines Reisezieles zur Darstellung in den Top 3 Ergebnissen bei n Reisezielen

$$f\ddot{u}r\ n = 29: \;\; 1 - \left(\frac{28}{29} \times \frac{27}{28} \times \frac{26}{27}\right) = \frac{3}{29} = 10{,}34\%$$

Formel 8 - Wahrscheinlichkeit eines Reisezieles zur Darstellung in den Top 3 Ergebnissen der hier verwendeten Gesamtmenge von 29 Reisezielen

Eine Nutzerakzeptanz, die deutlich über der Wahrscheinlichkeit eines zufälligen Ziehens ohne Zurücklegen liegt, entspricht isoliert betrachtet daher keinem ausreichendem Testergebnis. Beispielsweise wird vermutet, dass ein Nutzer den Testdurchlauf bereits voreingenommen startet. Dies könnte zur Folge haben, dass bekannte oder nicht (mehr) interessante Reiseziele, wie z.B. Stuttgart, Berlin oder London, bereits vor dem Testdurchlauf unterbewusst als irrelevant bewertet werden. Im Vergleich dazu könnte ein Nutzer eine Affinität gegenüber exotischen oder unbekannten Reisezielen besitzen und folglich bei einer Auswahlmöglichkeit von z.B. Istanbul, New York oder Buenos Aires eine hohe Akzeptanz besitzen.

Die Auswahlmöglichkeit von drei Ergebnissen, welche entlang der Übereinstimmungsquote ermittelt werden, ermöglicht dem Nutzer zusätzliche Auswahlmöglichkeiten, wodurch auch die Anzahl positiver Aussagen über die Ergebnisse gesteigert wird. Zur zusätzlichen Validierung der Ergebnisse könnte an dieser Stelle z.B. ein weiterer Test durchgeführt werden, in dem die Anzahl der vorgeschlagenen Reiseziele (z.B. 1, 3, 5) variiert wird.

Bei einer starken Divergenz zwischen Reisezielen könnte ein Nutzer weiterhin eher zur Auswahl eines latent relevanten Reiseziels neigen. Erhält bspw. ein Nutzer die

Ergebnisse Berlin, London und Sydney, wobei Berlin und London für ihn keine relevanten Reiseziele darstellen, könnte er im Umkehrschluss folgern, dass Sydney ein akzeptables Ergebnis ist. Die Entwicklung zielt, wie in Kapitel 1.1 und 1.2 beschrieben, explizit auf die Nutzung des Unterbewusstseins des Nutzers ab. Daher ist eine Vorprägung im Hinblick auf relevante Reiseziele (z.B. durch vorherige Reiseerfahrungen, Budgetgrenzen oder sonstige Vorlieben) nicht zu beseitigen, sondern explizit gewünscht. Diese (z.T. unterbewussten) Vorprägungen im Rahmen einer gewöhnlichen parameter- und textbasierten Reisezielauswahl zu verschriftlichen, könnte ebendiesen Nutzern besonders schwerfallen. Durch die Konfrontation der Prädiktionsergebnisse mit der Vorprägung des Nutzers wird explizit ein Diskurs angeregt, welcher schlussendlich in einer Auswahl eines Reiseziels mündet.

5 Abschluss

5.1 Zusammenfassung

Diese Thesis umfasst die Konzeption und Entwicklung einer ML-basierten Prädiktion von Reisezielen auf Basis multivariater Parameter. Einleitend erfolgt eine Informationsakquise von 29 globalen Reisezielen (Rom, Istanbul, New York, etc.). Diese Informationen werden in mehreren Schritten mittels Data Science Methoden weiterverarbeitet und die so gewonnenen Reisezielattribute in neun gleichartige Kategorien zusammengefasst. Diese werden anschließend zur Bildklassifikation mittels eines CNN verwendet. Die Ergebnisse aus Datenverarbeitung und Bildklassifikation sind die Basis für einen dualen Bildvergleich. Hierbei werden dem Nutzer generische Bilder aus den o.g. Reisezielkategorien präsentiert (z.B. ein Museum für die Kategorie „Culture & Education“, etc.). Durch die Klassifikation dieser Bilder kann auf Basis der Nutzerauswahl ein Rückschluss auf die Interessen des Nutzers gezogen werden. In Folgerung dessen erhält der Nutzer Reiseziele vorgeschlagen, welche eine bestmögliche Übereinstimmung mit seinen Interessen bieten (z.B. hat ein Nutzer hat ein großes Interesse an „Shopping“, wird ihm ein Reiseziel mit vielen Einkaufsmöglichkeiten empfohlen). Weiterhin erhält der Nutzer nach erfolgter Bildauswahl die Möglichkeit zur Bewertung der Qualität der Prädiktion. Die Bildauswahl, Prädiktionsergebnisse und Nutzerakzeptanz werden im Rahmen eines Nutzertests ausgewertet und diskutiert. Zusätzlich zur Konzeption, Entwicklung und Bewertung erfolgt ein allgemeiner sowie anwendungsorientierter Vergleich zwischen der hier verwendeten ML-basierten Lösung und einer CSP-basierten Herangehensweise.

5.2 Fazit

Die Konzeption und Entwicklung einer Softwarelösung zur ML-basierten Prädiktion von Reisezielen wird erfolgreich abgeschlossen. Die im Rahmen dieser Thesis aufgetretenen Probleme werden überwiegend beseitigt. Es verbleibt hauptsächlich

das Problem mangelnder Datenqualität und -quantität in den Ursprungsdaten. Hierfür werden jedoch umfangreiche Lösungsvorschläge gegeben. Weitere Probleme werden evaluiert und durch verschiedene Szenarien skizziert. Falls notwendig wird auch bereits ein Lösungsansatz vorgesehen. Die abschließende Evaluation der Ergebnisse zur Feststellung der Nutzerakzeptanz liefert mit über 70% Akzeptanzquote zudem ein positives Ergebnis. Die Anwendbarkeit eines ML-basierten Ansatzes zur Prädiktion von Reisezielen kann durch die Entwicklung und die Auswertung der Nutzerakzeptanz bestätigt werden.

5.3 Ausblick

Ein produktiver Einsatz der hier beschriebenen ML-basierten Prädiktionslösung benötigt weitere Optimierung und Automatisierung. Die Anwendung eines relativ einfachen neuronalen Netzes zur Bildklassifikation ermöglicht bereits gute Ergebnisse, die Qualität der Prädiktionsergebnisse lässt sich durch eine intensive Optimierung der Modellparameter weiter steigern. Auch während der Datenverarbeitung können manuell durchgeführte Schritte durch den Einsatz von Machine Learning Methoden (weiter) automatisiert werden.

Weiterhin liefert die Auswertung nur eine begrenzte Möglichkeit, um Aussagen über die Qualität der Prädiktion zu treffen. Um diese weiter zu verfeinern, können bspw. tiefergehende Nutzerstudien durchgeführt werden.

Zur Verbesserung des Prozesses und der Ergebnisqualität werden im Rahmen der Evaluation umfangreiche Ansätze aufgezeigt. Auf deren Basis sowie durch Nutzung bestehender Entwicklungsergebnisse kann eine Weiterentwicklung erfolgen, welche eine produktive Nutzung ermöglicht.

Glossar

Begriff	Erklärung
Accuracy	Eine Metrik zur Beschreibung der Modellgenauigkeit. Sie beschreibt den Anteil der korrekten Vorhersagen und stellt damit eine intuitive Metrik dar [67]: $Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions} \Leftrightarrow \frac{TP + TN}{TP + FP + FN + TN}$
Algorithmus	In der Informatik ist ein Algorithmus eine Berechnungsvorschrift zur Lösung einer Aufgabe [118].
AvgPooling	Extraktion des Durchschnittswerts aus den jeweiligen Strides (siehe Pooling).
Backpropagation	Der primäre Algorithmus zur Durchführung des Gradientenabstiegs in neuronalen Netzen. Zunächst werden die Ausgabewerte jedes Knotens in einem Vorwärtsdurchlauf berechnet (und zwischengespeichert). Dann wird die partielle Ableitung des Fehlers in Bezug auf jeden Parameter in einem Rückwärtsdurchlauf durch den Graphen berechnet [Vgl. 28]. Hierdurch werden die Gewichte während des Trainings immer weiter optimiert [59].
Binäre Klassifikation *Binary Classification*	Unterscheidung zwischen genau zwei Klassen zur Klassifikation [27].
Binary Cross Entropy *Kreuzentropieverlust*	Eine typische Verlustfunktion zur Messung der Modellperformance eines Klassifizierungsmodells, dessen Ergebnis ein Wahrscheinlichkeitswert zwischen 0 und 1 ist. Der Kreuzentropieverlust nimmt zu, wenn die vorhergesagte Wahrscheinlichkeit von der tatsächlichen Kennzeichnung abweicht [63].
Convolution	Eine Convolution (Faltung) ist mathematisch betrachtet eine kombinierte Integration der Funktionen f und g, wobei die Funktionen übereinander verschoben werden [52–54]. $(f \otimes g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$
Convolution Layer	Convolutional Layer verwenden Faltungsoperationen von Matrizen zur Bildmanipulation (sog. convolutional operations) [28, 51]. Hierzu werden zwei Elemente benötigt, einerseits die Faltungsoperation und andererseits (ein Ausschnitt) eine(r) Eingabematrix (auch bekannt als Filter oder Kernel) [28]. Das Ziel der Convolution ist die Extraktion von Features aus dem Eingabebild mittels verschiedener Filter [56].
Convolutional Neural Network (CNN)	Ein Convolutional Neural Network (CNN) ist ein eine Deep Learning Netzwerkarchitektur, die welche zur Bildklassifikation verwendet wird [45, 50]. Das Netzwerk unterscheidet Eingabebilder anhand

	verschiedener Aspekte im Bild (sog. Features), die dementsprechende Gewichtungen erhalten [45, 50].
CRISP-DM *Cross Industry Process for Data Mining*	Allgemein anerkanntes und weitverbreitetes iteratives Data Mining Prozessmodell mit den sechs Phasen Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation und Deployment [12].
Data Mining	Der Prozess, Informationen aus Datenbanken zu sammeln, die zuvor unverständlich und unbekannt waren, und diese Informationen dann (für relevante Geschäftsentscheidungen) nutzbar zu machen. Ein Data Mining Prozess enthält grundsätzlich problemunabhängige Schritte von der Problemformulierung bis zur Problemlösung.
Data Scraping	Mit Data Scraping ist in seiner allgemeinsten Form eine Technik gemeint, bei der ein Computerprogramm Daten aus dem Output eines anderen Programms extrahiert [Vgl. 119]. Data Scraping findet häufig beim Web-Scraping statt. Bei diesem Prozess wird eine Anwendung verwendet, um wertvolle Informationen aus einer Website zu extrahieren [Vgl. 119].
Data Warehouse	Eine Architektur zur Datenhaltung von Daten aus transaktionalen Systemen, operativen Datenspeichern und externen Datenquellen. Das Data Warehouse kombiniert Daten in einer Aggregationsform, sodass sie auf vordefinierte Geschäftsprozesse anwendbar sind, sich unternehmensweite Datenanalysen durchführen lassen und sie für Reportingzwecke genutzt werden können [120].
Deep Learning	Eine Familie von maschinellen Lernalgorithmen, welche auf künstlichen Neuronalen Netzen (KNN) basieren [17]. (Siehe Künstliches Neuronales Netz)
Entropie	Entropie ist ein informationstheoretisches Maß für den mittleren Informationsgehalt einer Nachricht[121]: $$H(p) = \sum_{i=1}^{n} -p_i \cdot \log_2 p_i$$
Entscheidungsbaum *Decision Tree*	Entscheidungsbäume folgen einem baumstrukturierten Klassifikationsschema, bei dem die Knoten die Eingabevariablen und die Blätter die Entscheidungsergebnisse repräsentieren [38].
Euklidische Distanz	Die euklidische Distanz berechnet den Abstand zwischen zwei reellwertigen Vektoren [28].
EVA-Prinzip	Information wird in einer Form […] in das Informationsverarbeitungssystem eingegeben und in einer anderen Form […] ausgegeben [122].
False Negative (FN)	Ein Beispiel, in dem das Modell fälschlicherweise die inkorrekte Klasse vorhersagt [28].

False Positive (FP)	Ein Beispiel, in dem das Modell fälschlicherweise die korrekte Klasse vorhersagt [28].
Feature	Individuell messbare Eigenschaft oder Charakteristik eines Objekts [33].
Feature Extraktion *Feature Extraction*	Abrufen von Features, die von einem unüberwachten Modell berechnet wurden, zur Verwendung in einem anderen Modell als Input [28].
F-Score	Der F-Score, auch F1-Score genannt, ist ein Maß für die Genauigkeit eines Modells [29]. Er wird verwendet, um binäre Klassifikationssysteme zu bewerten, die Beispiele in "positiv" oder "negativ" klassifizieren. Der F-Wert ist eine Möglichkeit, Precision und Recall des Modells zu kombinieren [36]. $$F1 = 2 \times \frac{precision\ x\ recall}{precision + recall} \Leftrightarrow \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$
Generalisierung	Das Ziel eines Modells ist es, möglichst genaue Vorhersagen über neue, ungesehene Daten zu treffen. Wenn ein Modell in der Lage ist, auf der Basis von Trainingsdaten, für neue, ungesehene Testdaten akkurate Vorhersagen zu treffen, gilt es als generalisierungsfähig [27].
Halbüberwachtes Lernen *Semi-Supervised Learning*	Halbüberwachte Lernalgorithmen verwenden ein Trainingsdatensatz mit annotierten und nicht annotierten Daten. Diese Methode ist besonders nützlich, wenn es schwierig ist, relevante Merkmale aus den Daten zu extrahieren, und die Kennzeichnung von Beispielen eine zeitintensive Aufgabe für Experten ist [25].
Heaviside Funktion	Die Heaviside Funktion gibt für negative Argumente eine 0 und für Elemente größer gleich 0 eine 1 zurück: $$H(\mu) \coloneqq \begin{cases} 0, for\ \mu < 0 \\ 1, for\ \mu \geq 0 \end{cases}$$
Informationsgehalt	Der Informationsgehalt einer Nachricht ist eine logarithmische Größe, die angibt, wie viel Information in dieser Nachricht übertragen wurde. Je unwahrscheinlicher eine Nachricht ist, desto höher ist ihr Informationsgehalt und umgekehrt [121]: $$I(p) = \log_2 p$$
Klassifikation *Classification*	Klassifikation im Sinne von überwachten Lernalgorithmen bedeutet die Abbildung von Inputdaten auf bestimmte Klassen [17]. Die Klassifikation lässt sich in die zwei Bereiche binäre Klassifikation (binary classification) und mehrklassige Klassifikation (multi-class classification) unterteilen [27].
Klassifikationsbaum *Classification Tree*	Ein Entscheidungsbaum, welcher zur Klassifikation verwendet wird, nennt sich Klassifikationsbaum [37, 40]. Klassifikationsbäume werden angewendet, wenn auf ein diskretes Ergebnis (Klassen) geschlossen werden soll.

Künstliche Intelligenz (KI) *Artificial Intelligence*	Die Fähigkeit einer Maschine / eines Computerprogramms, abstrakte und vor dem Hintergrund eines natürlichen Vorbilds, intelligente, eigenständige Entscheidungen zu treffen und daraus (zweckvolles) Handeln abzuleiten. Ein Teilgebiet der Informatik und des Ingenieurwesens, welches sich mit dem (rechnergestützten) Verständnis von intelligentem (Maschinen-) Verhalten sowie der Entwicklung von dementsprechenden Anwendungen auseinandersetzt.
Künstliches neuronales Netz *Artificial Neural Network*	Nervenzellen-ähnliche Netzstrukturen, die sich in Grundzügen an der Funktionsweise biologischer neuronaler Netze orientieren [123].
Lernalgorithmus	Ein Lernalgorithmus ist ein Algorithmus, der Beispieldaten (Lerndaten, oder Trainingsdaten) erhält und daraus ein verallgemeinertes Modell berechnet, das auch auf neue Daten anwendbar ist [118].
Lineare Klassifikation *Linear Classification*	Nutzung einer, oder mehrerer linearer Funktion als Klassifikationsgrenze [27].
Lineare Regression *Linear Regression*	Die lineare Regression stellt eine Beziehung zwischen der abhängigen Variablen Y und einer oder mehreren unabhängigen Variablen X_i her, indem eine sog. Regressionsgerade durch die Variablen gezogen wird [35].
Machine Learning *Maschinelles Lernen*	Maschinelles Lernen ist eine Computeranwendung, die automatisch aus Erfahrungen lernt, um sich so zu verbessern, ohne explizit dafür programmiert zu sein. Maschinelles Lernen ist eine Teilmenge der künstlichen Intelligenz, bei der Computeralgorithmen verwendet werden, um autonom aus Daten und Informationen zu lernen.
MaxPooling	Extraktion des Maximalwerts aus den jeweiligen Strides (siehe Pooling).
McCulloch-Pitts Neuron	Binäres grenzwertbasiertes Berechnungsmodell, welches sich an den rechnerischen Fähigkeiten realer Neuronen orientiert. Die Eingänge werden mittels gerichteten, ungewichteten Kanten in einer gewichteten Summenfunktion zusammengeführt [43]. Abhängig des Schwellenwerts μ_i wird das Neuron aktiviert und überträgt einen binären Wert [42, 43].
Mehrklassige Klassifikation *Multi-class Classification*	Im Gegensatz zu einer binären Klassifikation wird bei einer mehrklassigen Klassifikation wird zwischen mehr als zwei Klassen unterschieden [36].
Metrik	Maßzahl zur Bewertung der Modellqualität eines maschinellen Lernmodells.
Multi Layer Perceptron	Mit Hilfe von Mehrschicht Perzeptronen werden Neuronen in Schichten hintereinander modelliert [20, 49]. Diese Schichten umfassen

Mehrschicht Perzeptron	stets eine Eingabeschicht, über welche Daten in das System eingelesen werden, sowie eine Ausgabeschicht, über welche die Ausgaben erfolgen [20, 49]. Dazwischen können sich versteckte Schichten (hidden layer) befinden, welche den eigentlichen Lernphasen entsprechen [20].
Nicht-lineare Klassifikation *Non-Linear Classification*	Nutzung einer, oder mehrerer nicht-linearer Funktion als Klassifikationsgrenze [27].
Nicht-Lineare Regression *Non-Linear Regression*	Nutzung einer nicht-linearen Funktion zur Abbildung einer Regression [36].
One-Hot Encoding	One Hot Encoding ist eine gängige Methode zur Vorverarbeitung kategorialer Merkmale für maschinelles Lernen. Diese Art der Codierung erzeugt ein neues binäres Merkmal für jede mögliche Kategorie und weist dem Merkmal jedes Samples, das seiner ursprünglichen Kategorie entspricht, den Wert 1 zu [124]. Dies kann z.B. zur Startinitialisierung von Gewichten in einem CNN genutzt werden.
Overfitting	Overfitting beschreibt ein Modell, welches die Trainingsdaten zu stark abbildet und damit auf Testdaten nicht generalisierungsfähig ist. Im Anhang (Abbildung 52) ist mittig ein Beispiel für Overfitting beim Einsatz einer polynomialen Funktion hoher Ordnung bei einem Regressionsproblem skizziert.
Perzeptron *Perceptron*	Neuronenmodell mit numerischen Gewichte sowie der Möglichkeit zur Abbildung verschiedener Verbindungsmuster [43, 45]. Der Schritt des Lernens erfolgt durch die Anpassung der Gewichte [44].
Point of Interest (POI) *Sonderziel*	Ein Point of Interest (POI), sind Sonderziele, die ein Nutzer als nützlich oder interessant empfinden könnte[125]. Die meisten Verbraucher verwenden diesen Begriff, wenn sie sich auf Hotels, Campingplätze, Tankstellen oder andere in modernen Navigationssystemen verwendete Kategorien beziehen [Vgl. 125].
Pooling	Pooling beschreibt die Dimensionsreduktion einer Matrix, die in vorangegangenen Convolutional Layern erzeugt wird [28]. Siehe: AvgPooling, MaxPooling
Precision	Eine Metrik für Klassifizierungsmodelle. Präzision identifiziert die Häufigkeit, mit der ein Modell bei der Vorhersage der positiven Klasse korrekt ist [28]: $$Precision = \frac{TP}{TP + FP}$$

Recall

Eine Metrik für Klassifizierungsmodelle, die die folgende Frage beantwortet: Wie viele von allen möglichen positiven Kennzeichnungen hat das Modell korrekt identifiziert:

$$Recall = \frac{TP}{TP + FN}$$

Regression

Regression ist statistisches Verfahren, was im Rahmen von überwachten Lernverfahren angewendet wird [28]. Im Gegensatz zu der Klassifikation, modelliert die Regression einen Zielvorhersagewert auf der Grundlage unabhängiger Variablen [26].

Regressionbaum

Regression Tree

Ein Entscheidungsbaum, welcher zur Regression verwendet wird, nennt sich Regressionsbaum [37, 40]. Klassifikationsbäume werden angewendet, wenn auf ein diskretes Ergebnis (Klassen) geschlossen werden soll.

Regressionsbäume werden zur Vorhersage von numerischen Attributen verwendet.

Regularisierung

Änderung, an ein Lernalgorithmus, zur Reduktion des Generalisierungsfehlers, aber nicht des Trainingsfehlers [61].

ReLU

Die Rectified Linear Unit Funktion (ReLU) ist eine nicht-lineare Aktivierungsfunktion, welche in mehrschichtigen bzw. tiefen neuronalen Netzen angewendet wird [56]. Die ReLU Funktion bilet einen Eingabewert auf 0 ab, wenn er negativ ist. Wenn der Eingabewert positiv ist, wird er übernommen:

$$f(x) = \begin{cases} 0, wenn\ x < 0 \\ x, wenn\ x \geq 0 \end{cases}$$

Softmax

Die Softmax-Funktion wandelt einen Vektor von reellen Zahlen in einen Vektor von Wahrscheinlichkeiten um [59]. Es erfolgt stets eine Abbildung auf einen Wertebereich zwischen 0 und 1 [29]. Die Summe der so entstehenden Wahrscheinlichkeiten innerhalb des Vektors beträgt 1 [60]:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Stride

Der Schritt der Verschiebung einer Faltungsmatrix über die Eingabematrix wird als „Stride" bezeichnet [55].

Testdaten

Validation Data

Teil eines Datensatzes, unabhängig von den Trainingsdaten, mit dem das Modell auf Korrektheit überprüft (getestet) wird [28].

Im Englischen wird der Begriff *„Validation Data"* verwendet. *„Test Data"* beschreibt einen komplett disjunkten Datensatz, welcher zum Test des Modells verwendet wird, nachdem die Validierung erfolgt ist.

Trainingsdaten

Training Data

Teil eines Datensatzes, auf dem ein Modell trainiert wird [28].

True Negative (TN)	Ein Beispiel, in dem das Modell die inkorrekte Klasse korrekt vorhergesagt hat [28].
True Positive (TP)	Ein Beispiel, in dem das Modell die korrekte Klasse korrekt vorhergesagt hat [28].
Überwachtes Lernen *Supervised Learning*	Ein maschinelles Lernkonzept zur Entwicklung eines Modells, bei dem anhand von bekannten Datensätzen Vorhersagen für unbekannte Daten getroffen werden [41]. Dazu benötigt ein unüberwachter Lernalgorithmus (korrekt) annotierte Eingabedaten, die eine Zielspalte mit Klasseninformation enthalten, auf die eine Abbildung erfolgen soll [42, 43].
Underfitting	Der Term Underfitting beschreibt ein Modell, welches weder die Trainingsdaten korrekt modelliert, noch auf Testdaten generalisierungsfähig ist [17]. Meist ist dies durch ein Modell bedingt, welches zu simpel ist, um die Varianz der Daten zu beschreiben [27, 28]. Im Anhang (Abbildung 52) ist links ein Beispiel für Underfitting beim Einsatz einer Regressionsgerade skizziert.
Unüberwachtes Lernen *Unsupervised Learning*	Bei unüberwachten Lernverfahren ist im Gegensatz zum überwachten Lernen keine Vorgabe von annotierten Daten notwendig. Der Algorithmus findet die Problemlösung selbstständig.
Verstärkungslernen *Reinforcement Learning*	Bei dieser Art des maschinellen Lernens versucht eine KI, den optimalen Weg zur Zielerreichung zu finden, bzw. die Leistung bei einer bestimmten Aufgabe zu verbessern. Wenn der Algorithmus eine Handlung ausführt, die zum Ziel führt, erhält er eine Belohnung. Das Gesamtziel: Vorhersage des besten nächsten Schritts, der unternommen werden muss, um die größte endgültige Belohnung zu erhalten.

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Thesis selbständig angefertigt und mich keiner fremden Hilfe bedient sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Thesis hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 01.11.2020 Marius Golombeck

Erklärung

Mir ist bekannt, dass nach §156 StGB bzw. §161 StGB eine falsche Versicherung an Eides Statt bzw. eine fahrlässige falsche Versicherung an Eides Statt mit Freiheitsstrafe bis zu drei Jahren bzw. bis zu einem Jahr oder mit Geldstrafe bestraft werden kann.

Dortmund, den 01.11.2020 Marius Golombeck

Quellenverzeichnis

[1] Oracle, *Definition: Data Science.* [Online]. Verfügbar unter: https://www.oracle.com/data-science/what-is-data-science.html (Zugriff am: 1. November 2020).

[2] C. Hayashi, „What is Data Science ? Fundamental Concepts and a Heuristic Example“ in *Data Science, Classification, and Related Methods*, Tokyo, 1998, S. 40–51.

[3] B. J. Fry, „Computational Information Design“. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2004. [Online]. Verfügbar unter: https://benfry.com/phd/dissertation-110323c.pdf

[4] S. Dhanrajani, „Data Science - The new Monetization Model for Analytics“, 12. Jan. 2015. [Online]. Verfügbar unter: https://sameerdhanrajani.wordpress.com/2015/01/12/data-science-the-new-monetization-model-for-analytics-industry/. Zugriff am: 1. November 2020.

[5] M. Braschler, T. Stadelmann und K. Stockinger, *Applied Data Science: Lessons Learned for the Data-Driven Business,* 1. Aufl. Cham: Springer International Publishing; Imprint, Springer, 2019.

[6] Gartner, *Gartner Information Technology Glossary: Data Scientist.* [Online]. Verfügbar unter: https://www.gartner.com/en/information-technology/glossary/data-scientist (Zugriff am: 1. November 2020).

[7] C. Kozyrov, *What on earth is data science?* [Online]. Verfügbar unter: https://www.kdnuggets.com/2018/09/what-is-data-science.html (Zugriff am: 1. November 2020).

[8] D. Conway, „The Data Science Venn Diagram“, 30. Sep. 2010. [Online]. Verfügbar unter: http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram. Zugriff am: 1. November 2020.

[9] C. Shah, *A Hands-On Introduction to Data Science*. Cambridge University Press, 2020.

[10] L. Heiler, „Difference of Data Science, Machine Learning and Data Mining“, 20. März 2017. [Online]. Verfügbar unter: https://www.datasciencecentral.com/profiles/blogs/difference-of-data-science-machine-learning-and-data-mining. Zugriff am: 1. November 2020.

[11] C. Shearer, „The CRISP-DM Model: The New Blueprint for Data Mining“, *Journal of Data Warehousing*, Jg. 2000, Nr. 4, S. 13–22, 2000, Art. no. 4. [Online]. Verfügbar unter: https://mineracaodedados.files.wordpress.com/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf

[12] P. Chapman *et al.,* „CRISP-DM 1.0: Step-by-step data mining guide“, 2000.

[13] G. Shmueli, *Data mining for business intelligence*. John Wiley & Sons, 2016.

[14] Oxford Reference, "*artificial intelligence*". [Online]. Verfügbar unter: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095426960 (Zugriff am: 1. November 2020).

[15] A. &. C. B. Staff, Hg., *Dictionary of Computing,* 5. Aufl. London, Minneapolis: A & C Black; Consortium Book Sales & Distribution [distributor], 2008.

[16] Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence: A-N*. Wiley, 1987.

[17] V. Wittpahl, *Künstliche Intelligenz: Technologie | Anwendung | Gesellschaft*. Springer-Verlag GmbH Berlin Heidelberg, 2019.

[18] Accenture, Hg., „Why Artificial Intelligence is the Future of Growth“, 2018. [Online]. Verfügbar unter: https://www.accenture.com/t20170524t055435__w__/ca-en/_acnmedia/pdf-52/accenture-why-ai-is-the-future-of-growth.pdf. Zugriff am: 1. November 2020.

[19] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. und Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Hg., „Künstliche Intelligenz: Wirtschaftliche Bedeutung, gesellschaftliche Herausforderungen, menschliche Verantwortung“, 2017. [Online]. Verfügbar unter: https://www.dfki.de/fileadmin/user_upload/import/9744_171012-KI-Gipfelpapier-online.pdf. Zugriff am: 1. November 2020.

[20] A. W. Trask, *Grokking deep learning*. Shelter Island: Manning, 2019.

[21] B. Marr, *A Short History of Machine Learning: Every Manager Should Read.* [Online]. Verfügbar unter: https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#52ae8fc615e7 (Zugriff am: 1. November 2020).

[22] A. L. Samuel, „Some studies in machine learning using the game of checkers“, *IBM J. Res. & Dev.*, Jg. 44, Nr. 1.2, S. 206–226, 2000, doi: 10.1147/rd.441.0206.

[23] A. Géron, *Hands-on machine learning with Scikit-Learn & TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, Inc, 2019.

[24] Google, Hg., „Common ML Problems - What is Supervised Learning?“, 3. Sep. 2020. [Online]. Verfügbar unter: https://developers.google.com/machine-learning/problem-framing/cases. Zugriff am: 1. November 2020.

[25] Nvidia, Hg., „SuperVize Me: What’s the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?“, 2. Aug. 2018. [Online]. Verfügbar unter: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. Zugriff am: 1. November 2020.

[26] C. Friedrich, „Master Wahlpflichtmodul - 839 Maschinelles Lernen: Vorlesung 1 - Einführung“, 25. März 2019.

[27] A. C. Müller und S. Guido, *Introduction to machine learning with Python: A Guide for Data Scientists,* 1. Aufl. Bejing, Boston, Farnham, Sebastopol, Tokyo: Oreilly et Associates Inc, 2016.

[28] Google, Hg., „Machine Learning Glossary“, 11. Aug. 2020. [Online]. Verfügbar unter: https://developers.google.com/machine-learning/glossary. Zugriff am: 1. November 2020.

[29] D. Forsyth, *Applied Machine Learning,* 1. Aufl., 2019.

[30] F.-F. Li, R. Krishna und D. Xu, „CS231n: Convolutional Neural Networks for Visual Recognition“. Course Materials, 2020. [Online]. Verfügbar unter: https://cs231n.github.io/classification/

[31] Google, Hg., „ML Practicum: Image Classification“, 15. Sep. 2020. [Online]. Verfügbar unter: https://developers.google.com/machine-learning/practica/image-classification. Zugriff am: 1. November 2020.

[32] H. Liu, *Feature selection for knowledge discovery and data mining*. [Place of publication not identified]: Springer-Verlag New York, 2013.

[33] C. Bishop, *Pattern recognition and machine learning*. Springer Verlag.

[34] M. J. Canty, *Image Analysis, Classification and Change Detection in Remote Sensing: With Algorithms for Python, Fourth Edition*. CRC Press, 2019.

[35] S. Weisberg, *Applied linear regression,* 3. Aufl. Hoboken, NJ: Wiley, 2005.

[36] T. Hastie, R. Tibshirani, J. Friedman, R. Tibshirani und J. Friedman, *Elements of Statistical Learning, The*. New York, NY: Springer, 2001.

[37] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis und D. I. Fotiadis, „Machine learning applications in cancer prognosis and prediction“, *Computational and Structural Biotechnology Journal*, Jg. 13, S. 8–17, 2015, doi: 10.1016/j.csbj.2014.11.005.

[38] J. R. Quinlan, „Induction of decision trees“, *Mach Learn*, Jg. 1, Nr. 1, S. 81–106, 1986, doi: 10.1007/BF00116251.

[39] G. James und D. Witten, *An introduction to statistical learning: with applications in R*. New York [etc.], 2013.

[40] L. Breiman, *Classification and regression trees*. [Abingdon]: [Routledge], 1984.

[41] W. Ertel, Hg., *Grundkurs Künstliche Intelligenz*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016.

[42] J. A. Hertz, A. S. Krogh und R. G. Palmer, *Introduction to the theory of neural computation*. Reading, Massachusetts [i pozostałe]: Addison-Wesley Publishing Company, 1993.

[43] R. Rojas, *Neural networks: A systematic introduction*. Berlin [etc.]: Springer, op. 1996.

[44] F. Rosenblatt, „The perceptron: a probabilistic model for information storage and organization in the brain“ (eng), *Psychological review*, Jg. 65, Nr. 6, S. 386–408, 1958, doi: 10.1037/h0042519.

[45] Dong, H. D. D. Zhang und Chang, *Deep Reinforcement Learning,* 1. Aufl. [S.l.]: Springer Singapore, 2020.

[46] M. L. Minsky und S. A. Papert, *The Perceptrons*. Cambridge, Massachusetts, London.

[47] D. E. Rumelhart, G. E. Hinton und R. J. Williams, „Learning representations by back-propagating errors“, *Nature*, Jg. 323, Nr. 6088, S. 533–536, 1986, doi: 10.1038/323533a0.

[48] P. Werbos und P. John, „Beyond regression : new tools for prediction and analysis in the behavioral sciences“, 1974.

[49] J. Scherk, G. Pöchhacker-Tröscher und K. Wagner, „Künstliche Intelligenz - Artificial Intelligence“, Mai 2017. [Online]. Verfügbar unter: http://bmvit.gv.at/innovation/downloads/kuenstliche_intelligenz.pdf. Zugriff am: 1. November 2020.

[50] F. van Veen, „The neural network zoo“, 14. Sep. 2016. [Online]. Verfügbar unter: https://www.asimovinstitute.org/neural-network-zoo/. Zugriff am: 1. November 2020.

[51] Y. Lecun, L. Bottou, Y. Bengio und P. Haffner, „Gradient-based learning applied to document recognition“, *Proceedings of the IEEE*, Jg. 86, Nr. 11, S. 2278–2324, 1998, doi: 10.1109/5.726791.

[52] U. Schwarzbach, „Road in Muang Boran (Ancient City) in Samut Phrakan, Thailand: In den Daten zu finden als: cultureandeducation_3.jpg“, 21. Feb. 2016. [Online]. Verfügbar unter: https://www.flickr.com/photos/uwebkk/48960324397/. Zugriff am: 1. November 2020.

[53] E. W. Weisstein, „Convolution - MathWorld - A Wolfram Web Resource“, 13. Okt. 2020. [Online]. Verfügbar unter: https://mathworld.wolfram.com/Convolution.html. Zugriff am: 1. November 2020.

[54] C. M. Lee, *Visual comparison of convolution, cross-correlation and autocorrelation of two signals by CMG Lee.* [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Comparison_convolution_correlation.svg.

[55] U. Karn, *An Intuitive Explanation of Convolutional Neural Networks.* [Online]. Verfügbar unter: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[56] A. Geitgey, „Machine Learning is Fun“, 13. Juni 2016. [Online]. Verfügbar unter: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721. Zugriff am: 1. November 2020.

[57] R. Becker, *Convolutional Neural Networks - Aufbau, Funktion und Anwendungsgebiete*. [Online]. Verfügbar unter: https://jaai.de/convolutional-neural-networks-cnn-aufbau-funktion-und-anwendungsgebiete-1691/.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, Jg. 15, Nr. 56, S. 1929–1958, 2014. [Online]. Verfügbar unter: http://jmlr.org/papers/v15/srivastava14a.html

[59] Zhang, *Fundamentals of Image Data Mining*. Springer International Publishing, 2019.

[60] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. [Online]. Verfügbar unter: http://neuralnetworksanddeeplearning.com

[61] Y. Bengio, I. Goodfellow und A. Courville, *Deep learning*. Massachusetts: MIT Press, 2017.

[62] K. Krzyk, „Coding Deep Learning for Beginners — Linear Regression (Part 2): Cost Function", 8. Aug. 2018. [Online]. Verfügbar unter: https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f. Zugriff am: 1. November 2020.

[63] D. Godoy, „Understanding binary cross-entropy / log loss: a visual explanation", 21. Nov. 2018. [Online]. Verfügbar unter: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a. Zugriff am: 1. November 2020.

[64] C. Versloot, „How to use binary & categorical crossentropy with Keras?", 22. Okt. 2019. [Online]. Verfügbar unter: https://www.machinecurve.com/index.php/2019/10/22/how-to-use-binary-categorical-crossentropy-with-keras/. Zugriff am: 1. November 2020.

[65] H. Orlowski, „Implementierung eines neuronalen Netzes zur Prädiktion von Aktienkursbewegungen", 2019.

[66] D. P. Kingma und J. Ba, „Adam: A Method for Stochastic Optimization", 22. Dez. 2014. [Online]. Verfügbar unter: http://arxiv.org/pdf/1412.6980v9.

[67] A. Singh, „Classification Report: Precision, Recall, F1-Score, Accuracy", Aug. 2020. [Online]. Verfügbar unter: https://www.kaggle.com/getting-started/175898. Zugriff am: 1. November 2020.

[68] Google, Hg., „Classification: Accuracy", 10. Feb. 2020. [Online]. Verfügbar unter: https://developers.google.com/machine-learning/crash-course/classification/accuracy. Zugriff am: 1. November 2020.

[69] W. Runte, „YACS: Ein hybrides Framework für Constraint-Solver zur Unterstützung wissensbasierter Konfigurierung". Diplomarbeit, 2006. [Online]. Verfügbar unter: https://sys.cs.uos.de/woru/pub/diplom/html/Diplom.html

[70] E. Tsang, *Foundations of constraint satisfaction*. London: Academic Press, 1993.

[71] K. Ghédira, *Constraint satisfaction problems: CSP formalisms and techniques*. London, Hoboken, N.J.: ISTE; Wiley, 2013.

[72] E. Tsang, *Constraint Satisfaction Problem*. Academic Press, 1993.

[73] I. Gent, L. Kotthoff, I. Miguel und P. Nightingale, „Machine learning for constraint solver design - A case study for the alldifferent constraint", 25. Aug. 2020. [Online]. Verfügbar unter: https://arxiv.org/abs/1008.4326.

[74] Luc De Raedt, Siegfried Nijssen, Barry O'Sullivan und Pascal Van Hentenryck, „Constraint Programming meets Machine Learning and Data Mining (Dagstuhl Seminar 11201)", *Dagstuhl Reports*, Jg. 1, Nr. 5, S. 61–83, 2011, doi: 10.4230/DagRep.1.5.61.

[75] T. Wang, „Machine Learning for Constraint Programming“, KTH, School of Electrical Engineering and Computer Science (EECS), 2019.

[76] L. D. Raedt, A. Passerini und S. Teso, „Learning Constraints From Examples“ in *AAAI*, 2018.

[77] H. Xu, S. Koenig und T. K. S. Kumar, „Towards Effective Deep Learning for Constraint Satisfaction Problems“ in *Principles and Practice of Constraint Programming*, 2018, S. 588–597.

[78] Y. Xu, D. Stern und H. Samulowitz, „Learning Adaptation to Solve Constraint Satisfaction Problems“ in *LION 2009, Learning and Intelligent OptimizatioN*, 2009.

[79] M. Wolff, „Garbage In, Garbage Out: The Importance of Good Data“, 4. Juni 2019. [Online]. Verfügbar unter: https://medium.com/@marybrwolff/garbage-in-garbage-out-the-importance-of-good-data-ce1bb775468e. Zugriff am: 1. November 2020.

[80] M. Toussaint, „Introduction to Artificial Intelligence“, 4. Feb. 2019. [Online]. Verfügbar unter: https://www.user.tu-berlin.de/mtoussai/teaching/Lecture-AI.pdf. Zugriff am: 1. November 2020.

[81] Keras, „Keras Guide“. [Online]. Verfügbar unter: https://keras.io/guides/. Zugriff am: 1. November 2020.

[82] Python, „Python Documentation“. [Online]. Verfügbar unter: https://www.python.org/doc/. Zugriff am: 1. November 2020.

[83] Vue.JS, „Vue.JS Guide“. [Online]. Verfügbar unter: https://vuejs.org/v2/guide/. Zugriff am: 1. November 2020.

[84] KNIME, „KNIME Analytics Platform“. [Online]. Verfügbar unter: https://www.knime.com/knime-analytics-platform. Zugriff am: 1. November 2020.

[85] Statista, „Umfrage in den USA: Bei welchen dieser Anbieter haben Sie in den letzten 12 Monaten online (Website oder App) eine Unterkunft - Hotel oder Privatunterkunft - gebucht?“, 1. Mai 2020. [Online]. Verfügbar unter: https://de.statista.com/prognosen/1000433/umfrage-in-den-usa-zu-beliebten-buchungsportalen-fuer-unterkuenfte. Zugriff am: 1. Januar 2020.

[86] Statista, „Umfrage in Deutschland: Bei welchen dieser Anbieter haben Sie in den letzten 12 Monaten online (Website oder App) eine Unterkunft - Hotel oder Privatunterkunft - gebucht?“, 1. Mai 2020. [Online]. Verfügbar unter: https://de.statista.com/prognosen/999811/umfrage-in-deutschland-zu-beliebten-buchungsportalen-fuer-unterkuenfte. Zugriff am: 1. Januar 2020.

[87] Booking.com, „Booking.com - Hotels in Stuttgart“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.booking.com/searchresults.html?city=-1871728. Zugriff am: 1. November 2020.

[88] Tripadvisor, „Tripadvisor - Hotels in Stuttgart“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.tripadvisor.de/Hotels-g187291-Stuttgart_Baden_Wurttemberg-Hotels.html. Zugriff am: 1. November 2020.

[89] Expedia, „Expedia - Suchergebnisse für Hotels in Stuttgart (und Umgebung)“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.expedia.de/Hotel-Search?adults=2&d1=2020-11-05&d2=2020-11-06&destination=Stuttgart%20%28und%20Umgebung%29%2C%20Baden-W%C3%BCrttemberg%2C%20Deutschland&directFlights=false&endDate=2020-11-06&latLong=48.776993%2C9.17824&localDateFormat=dd.MM.yyyy&partialStay=false&regionId=6047258&semdtl=&sort=RECOMMENDED&startDate=2020-11-05&theme=&useRewards=false&userIntent. Zugriff am: 1. November 2020.

[90] Expedia, „Expedia - Suchergebnisse für Hotels in München (und Umgebung)“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.expedia.de/Hotel-Search?adults=2&d1=2020-11-

05&d2=2020-11-06&destination=M%C3%BCnchen%20%28und%20Umgebung%29%2C%20Bayern%2C%20Deutschland&directFlights=false&endDate=2020-11-06&latLong=48.1398%2C11.56005&localDateFormat=dd.MM.yyyy&partialStay=false&regionId=179896&semdtl=&sort=RECOMMENDED&startDate=2020-11-05&theme=&useRewards=false&userIntent. Zugriff am: 1. November 2020.

[91] Expedia, „Expedia - Suchergebnisse für Hotels in Hamburg (und Umgebung)“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.expedia.de/Hotel-Search?adults=2&d1=2020-11-05&d2=2020-11-06&destination=Hamburg%20%28und%20Umgebung%29%2C%20Hamburg%2C%20Deutschland&directFlights=false&endDate=2020-11-06&localDateFormat=dd.MM.yyyy&partialStay=false&regionId=180004&semdtl=&sort=RECOMMENDED&startDate=2020-11-05&theme=&useRewards=false&userIntent. Zugriff am: 1. November 2020.

[92] Tripadvisor, „Tripadvisor - Waldhotel Stuttgart“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.tripadvisor.de/Hotel_Review-g187291-d228307-Reviews-Waldhotel_Stuttgart-Stuttgart_Baden_Wurttemberg.html. Zugriff am: 1. November 2020.

[93] Booking.com, „Booking.com - Waldhotel Stuttgart“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.booking.com/hotel/de/waldhotel-stuttgart.de.html?aid=356980. Zugriff am: 1. November 2020.

[94] Booking.com, „Booking.com - Attractions“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.booking.com/attractions/index.en-gb.html?label=gen173nr-1DCAEoggI46AdIM1gEaDuIAQGYAQm4ARfIAQzYAQPoAQGIAgGoAgO4ArHWxvwFwAIB0gIkNjZkMDg1MDUtYTZkOC00NWI1LThhZTYtOWI5MDAyYTM1YTMx2AIE4AIB;sid=f4f51860593e0f5b2367b2ce74d109ec. Zugriff am: 1. November 2020.

[95] Tripadvisor, „Tripadvisor - Aktivitäten“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.tripadvisor.de/Search?q=Stuttgart&searchSessionId=31DC0958DA1527C59FF698A1F27C75BA1603379945184ssid&searchNearby=false&geo=4&sid=88163F307D98E73E8DFBE25568283946160337994927 8&blockRedirect=true&ssrc=A&rf=1. Zugriff am: 1. November 2020.

[96] Tripadvisor, „Tripadvisor - POIs in Stuttgart“, 1. Nov. 2020. [Online]. Verfügbar unter: https://www.tripadvisor.de/Attractions-g187291-Activities-Stuttgart_Baden_Wurttemberg.html. Zugriff am: 1. November 2020.

[97] Tripadvisor, „Tripadvisor - Content API“. [Online]. Verfügbar unter: http://developer-tripadvisor.com/content-api/. Zugriff am: 1. November 2020.

[98] Tripadvisor, „Tripadvisor - Request API Access“. [Online]. Verfügbar unter: http://developer-tripadvisor.com/content-api/request-api-access/. Zugriff am: 1. November 2020.

[99] M. Copelli, Hg., „tripadvisor-scraper“, 22. Sep. 2020. [Online]. Verfügbar unter: https://github.com/maxCopell/tripadvisor-scraper. Zugriff am: 1. November 2020.

[100] M. Copelli, Hg., „Apify - maxcopell/tripadvisor“, 22. Sep. 2020. [Online]. Verfügbar unter: https://apify.com/maxcopell/tripadvisor. Zugriff am: 1. November 2020.

[101] Easy Map Maker, „Übersicht über alle verwendeten Reiseziele“, 10. Okt. 2020. [Online]. Verfügbar unter: https://www.easymapmaker.com/map/44fd94bc7e4dfecbbf1d78d1752bb1f9. Zugriff am: 1. November 2020.

[102] Flickr, „Flickr - API Dokumentation“. [Online]. Verfügbar unter: https://www.flickr.com/services/api/. Zugriff am: 1. November 2020.

[103] Delfino Blu, „Natural Spa & wellness: In den Daten zu finden als: wellness_5.jpg“, 1. Apr. 2014. [Online]. Verfügbar unter: https://www.flickr.com/photos/delfinobluhotel/13949870410/. Zugriff am: 1. November 2020.

[104] J. Jeckson, „Best Nightlife Scottsdale: In den Daten zu finden als: nightlife_18.jpg“, 3. Juni 2017. [Online]. Verfügbar unter: https://www.flickr.com/photos/150123631@N05/35024436006/. Zugriff am: 1. November 2020.

[105] B. Wilson, „National Railway Museum, York, England: In den Daten zu finden als: cultureandeducation_193.jpg“, 23. Mai 2019. [Online]. Verfügbar unter: https://www.flickr.com/photos/billy_wilson/49917206413/. Zugriff am: 1. November 2020.

[106] iconicphotoservices, „Food & Drink: In den Daten zu finden als: foodanddrink_0.jpg“, 1. Feb. 2010. [Online]. Verfügbar unter: https://www.flickr.com/photos/iconicphotoservices/12942446503/. Zugriff am: 1. November 2020.

[107] R. Piyushgiri, „Colourful Corridor: In den Daten zu finden als: shopping_0.jpg“, 21. Aug. 2016. [Online]. Verfügbar unter: https://www.flickr.com/photos/p_revagar/29027654582/. Zugriff am: 1. November 2020.

[108] icemanphotos, „Spa and Wellness: In den Daten zu finden als: wellness_0.jpg“, 5. Jan. 2017. [Online]. Verfügbar unter: https://www.flickr.com/photos/icemanphotos/28042446309/. Zugriff am: 1. November 2020.

[109] Army Medicine, „Army doctor: Soldiers should avoid overtraining to prevent injury: In den Daten zu finden als: sportsandactivities_0.jpg“, 25. Aug. 2014. [Online]. Verfügbar unter: https://www.flickr.com/photos/armymedicine/15073002125/. Zugriff am: 1. November 2020.

[110] K. Young, „Transcendence: In den Daten zu finden als: sightseeing_0.jpg“, 1. März 2018. [Online]. Verfügbar unter: https://www.flickr.com/photos/katherineyoung/40556463671/. Zugriff am: 1. November 2020.

[111] G. Begin, „La Vallée du Diable: In den Daten zu finden als: nature_0.jpg“, 16. Juli 2016. [Online]. Verfügbar unter: https://www.flickr.com/photos/photos_nature/28413779220/. Zugriff am: 1. November 2020.

[112] R. Castro Labaca, „Nightlife & Party: In den Daten zu finden als: nightlife_0.jpg“, 4. Aug. 2016. [Online]. Verfügbar unter: https://www.flickr.com/photos/44713559@N05/31451381002/. Zugriff am: 1. November 2020.

[113] B. Onasill, „Toronto - Ontario - Canada - The Aga Khan Museum - Aga Khan Park: In den Daten zu finden als: cultureandeducation_0.jpg“, 17. Okt. 2018. [Online]. Verfügbar unter: https://www.flickr.com/photos/onasill/47940205991/. Zugriff am: 1. November 2020.

[114] S. Mohn, „Come Sail Away!: In den Daten zu finden als: entertainment_0.jpg“, 14. Jan. 2017. [Online]. Verfügbar unter: https://www.flickr.com/photos/123434475@N03/32129325140/. Zugriff am: 1. November 2020.

[115] Nvidia, „CUDA“. [Online]. Verfügbar unter: https://developer.nvidia.com/cuda-zone. Zugriff am: 1. November 2020.

[116] G. Niemeyer, „python-constraint 1.4.0“, 5. Nov. 2018. [Online]. Verfügbar unter: https://pypi.org/project/python-constraint/. Zugriff am: 1. Januar 2020.

[117] SWI Prolog, „SWI Prolog“, 27. Okt. 2020. [Online]. Verfügbar unter: https://www.swi-prolog.org/Download.html. Zugriff am: 1. Januar 2020.

[118] Frauenhof-Allianz Big Data, Hg., „Künstliche Intelligenz in Deutschland: Ein systematischer Katalog von Anwendungen des maschinellen Lernens“, Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme, Sankt Augustin, 7. Nov. 2018. [Online]. Verfügbar unter: https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Studie_KI_in_De_20181107.pdf. Zugriff am: 1. November 2020.

[119] Cloudflare, „Was ist Data Scraping?“. [Online]. Verfügbar unter: https://www.cloudflare.com/de-de/learning/bots/what-is-data-scraping/. Zugriff am: 1. November 2020.

[120] Gartner, *Gartner Information Technology Glossary: Data Warehouse.* [Online]. Verfügbar unter: https://www.gartner.com/en/information-technology/glossary/data-warehouse (Zugriff am: 1. November 2020).

[121] C. E. Shannon und W. Weaver, *The mathematical theory of communication,* 4. Aufl. Urbana: Univ. of Illinois Pr, 1969.

[122] W. Grafendorfer, *Einführung in die Datenverarbeitung für Informatiker*. Würzburg, Wien: Physica-Verlag, 1977.

[123] U. Helmich, „Digitaler Foliensatz "Bau und Funktion eines Neurons, Fallbeispiel Kniesehnenreflex"“, 2020. [Online]. Verfügbar unter: http://www.u-helmich.de/bio/neu/1/11/111/seite1112.html. Zugriff am: 1. November 2020.

[124] G. Novack, „Building a One Hot Encoding Layer with TensorFlow“, 7. Juni 2020. [Online]. Verfügbar unter: https://towardsdatascience.com/building-a-one-hot-encoding-layer-with-tensorflow-f907d686bf39. Zugriff am: 1. November 2020.

[125] Definitions.net, „point of interest“, STANDS4 LLC, 22. Okt. 2020. [Online]. Verfügbar unter: https://www.definitions.net/definition/point+of+interest. Zugriff am: 1. November 2020.

[126] F. van Veen und S. Leijnen, „A mostly complete chart of Neural Networks“. [Online]. Verfügbar unter: https://www.asimovinstitute.org/wp-content/uploads/2019/04/NeuralNetworkZo19High.png. Zugriff am: 1. November 2020.

Anhang

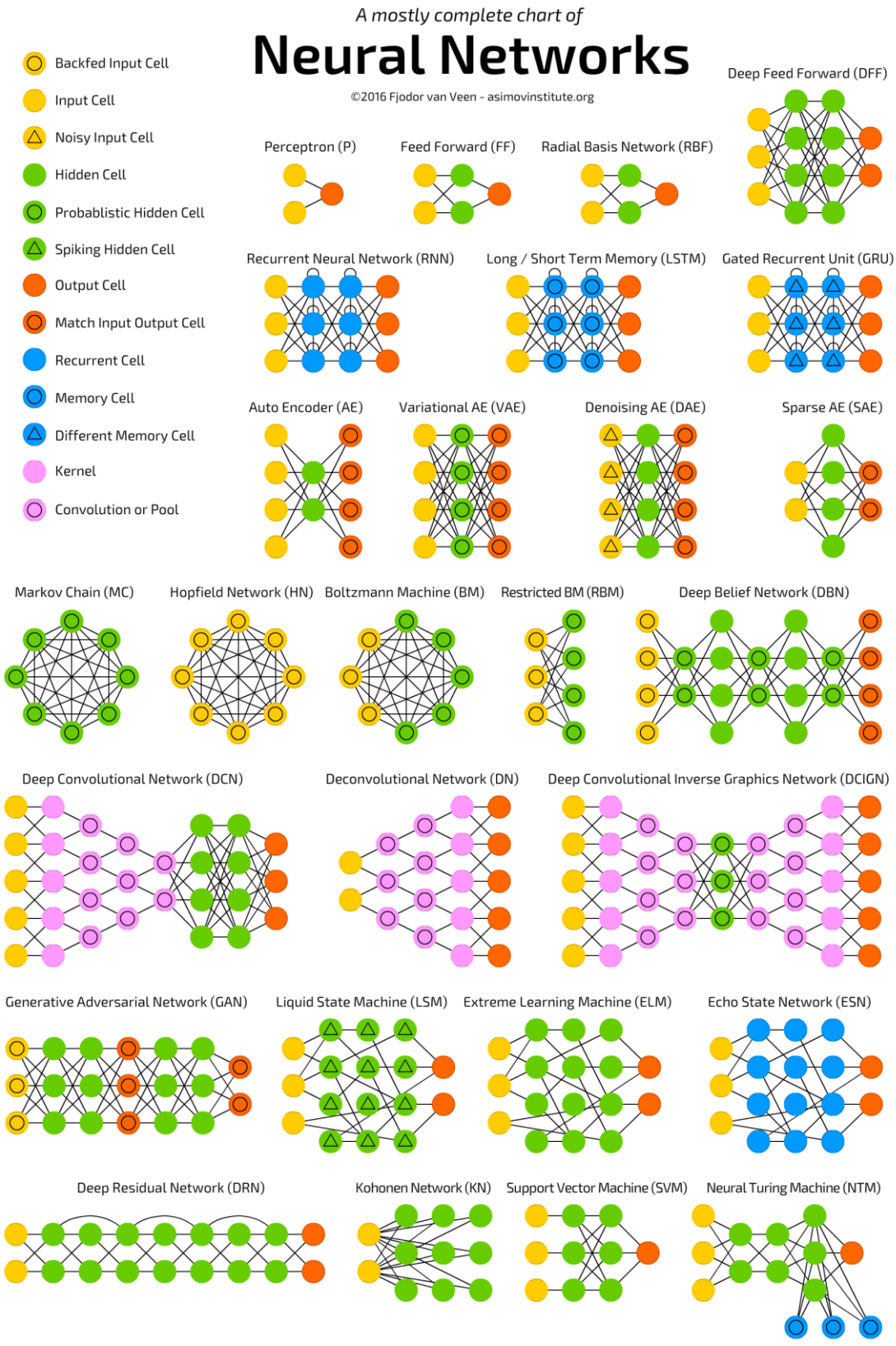


Abbildung 51 - Verschiedene Typen neuronaler Netzwerke [126]

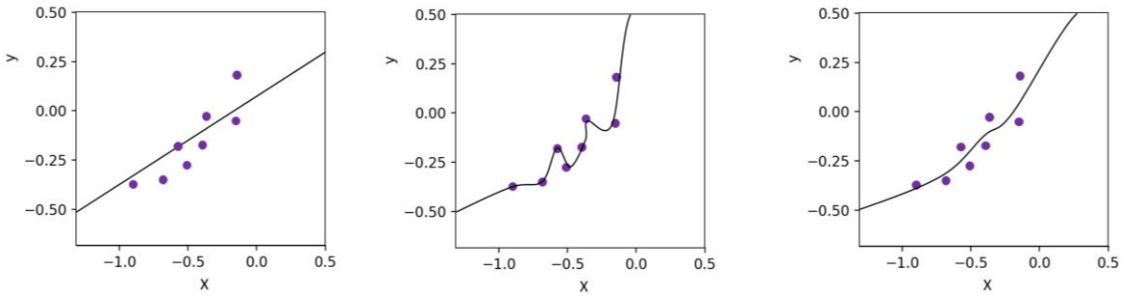


Abbildung 52 - Modellanpassung am Beispiel einer Regression: Underfitting (links), Overfitting (mittig) und ein optimales Fitting (rechts)

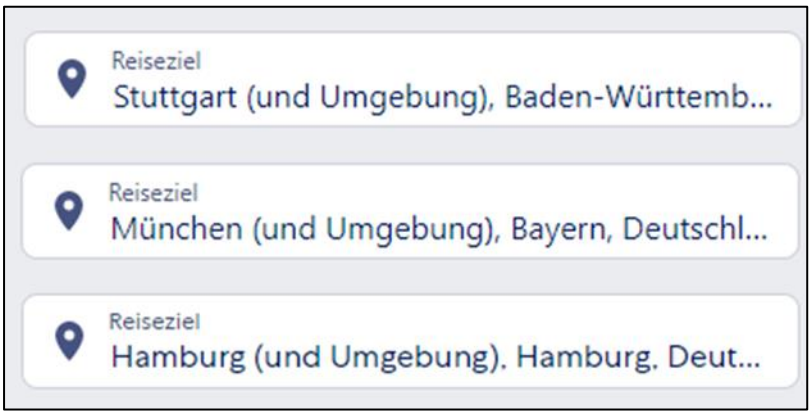


Abbildung 53 - Eine Abfrage auf „Expedia“ zu Stuttgart, Hamburg und München, die automatisch die jeweilige Umgebung mit einbezieht

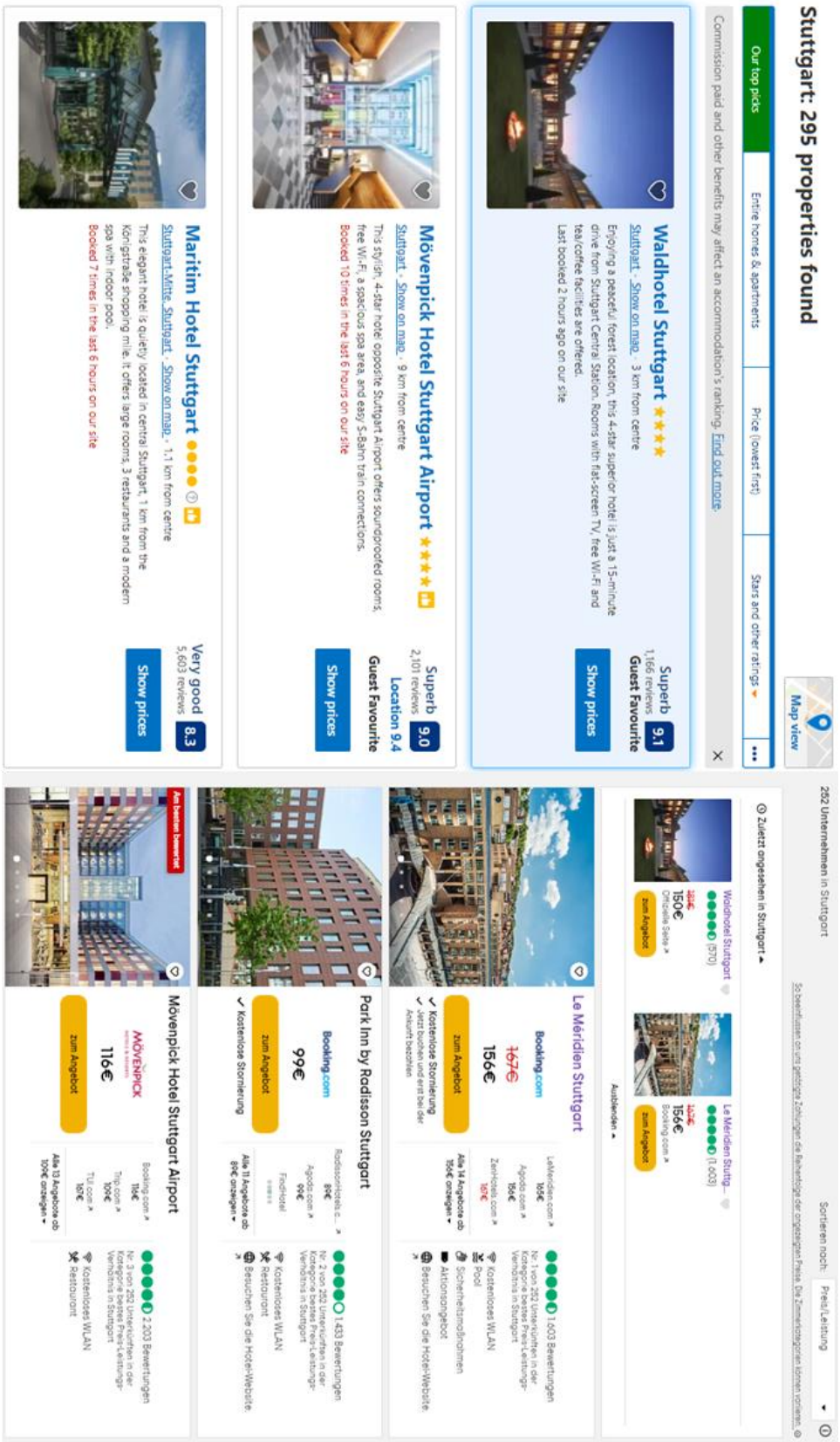


Abbildung 54 - Abfrage für Hotels in Stuttgart auf „Booking.com“ und „TripAdvisor“ mit jeweils drei Beispielhotels. Zu jedem Hotels sind die Bewertungen annotiert (z.B. 1166 für Waldhotel Stuttgart auf „Booking.com“)

Nähere Informationen

4,5 Ausgezeichnet
1.603 Bewertungen

Nr. 5 von 148 Hotels in Stuttgart

Lage
Sauberkeit
Service
Preis-Leistungs-Verhältnis

Travellers' Choice

Das Le Méridien steht für Design, Kunst, Kultur und lädt in absolut zentraler Stadtlage zu neuen Entdeckungen ein. Hier starten Gäste ihren Tag mitten im Herzen Stuttgarts – egal ob eine Jogging-Runde im Schlossgarten, ein Besuch in den beiden Automobilmuseen oder eine Tour zu den fußläufigen Wahrzeichen der Stadt – das Le Méridien Stuttgart vereint facettenreiche, kreative und lokale Inspiration im Bereich Lifestyle, Hotel und Business Meetings. 280 Hotelzimmer und 13 Suiten (30 bis 250 m²) bieten zeitgemäße Eleganz

Mehr lesen

Ausstattungen & Services des Unternehmens

- Parkhaus
- Kostenloses Highspeed-Internet (WLAN)
- Pool
- Fitnesscenter mit Trainingsraum
- Bar/Lounge
- Vermietung von Fahrrädern
- Babysitting
- Haustiere erlaubt (tierfreundlich)

Mehr anzeigen

Ausstattung der Zimmer

- Klimaanlage
- Kamin
- Zimmerservice
- Safe
- VIP-Zimmerausstattung
- Minibar
- Flachbildfernseher

Zimmertypen

- Nichtraucherzimmer
- Suiten

Weitere Zimmerinformationen anzeigen

Gut zu wissen

HOTEL-KLASSIFIZIERUNG

GESPROCHENE SPRACHEN
Deutsch

HOTELSTIL
Grün
Business

Hotel-Links

Besuchen Sie die Hotel-Website.
Aktionsangebot: Pauschalangebot
Hotelangebote

Abbildung 55 - Darstellung einer Nutzerbewertung des Hotels „Le Meridien“ in Stuttgart auf "TripAdvisor"

```
1.  [{
2.    "location_id": "2275103",
3.    "name": "Neckar Park",
4.    "latitude": "48.7973",
5.    "longitude": "9.21694",
6.    "num_reviews": "554",
7.    "timezone": "Europe/Berlin",
8.    "location_string": "Stuttgart, Baden-Wurttemberg",
9.    "photo": {...}
10.     },
11.     "is_blessed": true,
12.     "uploaded_date": "2016-05-07T05:19:25-0400",
13.     "caption": "Cannstatter Wasen",
14.     "id": "186854493",
15.     "helpful_votes": "8",
16.     "published_date": "2016-05-07T05:19:25-0400",
17.     "user": {
18.       "user_id": null,
19.       "member_id": "0",
20.       "type": "user"
21.     }
22.   },
23. ...
24.   "ranking_subcategory": "#3 of 195 things to do in Stuttgart",
25.   "subcategory_ranking": "#3 of 195 things to do in Stuttgart",
26.   "ranking": "#3 of 195 things to do in Stuttgart",
27.   "distance": null,
28.   "distance_string": null,
29.   "bearing": null,
30.   "rating": "4.5",
31.   "is_closed": false,
32.   "is_long_closed": false,
33.   "description": "",
34.   "web_url": "[...]",
35.   "write_review": "[...]",
36.   "ancestors": [...],
37.   "category": {
38.     "key": "attraction",
39.     "name": "Attraction"
40.   },
41.   "subcategory": [
42.     {
43.       "key": "57",
44.       "name": "Nature & Parks"
45.     }
46.   ],
47. ...
```

Quellcode 11 - Ausschnitt eines JSON Export einer API-Abfrage auf POIs in Stuttgart. Hier sind ist das Beispiel „Neckar Park“ abgebildet. Einige Stellen sind durch [...] oder ... verkürzt dargestellt.

Name	Value
address	"70372 Stuttgart, Baden-Wu…
address_obj	…
ancestors	…
api_detail_url	"https://api.tripadvisor.com/a…
awards	…
bearing	null
category	…
description	""
distance	null
distance_string	null
doubleclick_zone	"eu.germany.stuttgart"
is_candidate_for_contact_in…	false
is_closed	false
is_jfy_enabled	false
is_long_closed	false
latitude	"48.7973"
location_id	"2275103"
location_string	"Stuttgart, Baden-Wurttemb…
location_subtype	"none"
longitude	"9.21694"
name	"Neckar Park"
nearest_metro_station	…
num_reviews	"554"
parent_display_name	"Stuttgart"
photo	…
preferred_map_engine	"default"
ranking	"#3 of 195 things to do in St…
ranking_category	"attraction"
ranking_denominator	"195"
ranking_geo	"Stuttgart"
ranking_geo_id	"187291"
ranking_position	"3"
ranking_subcategory	"#3 of 195 things to do in St…
rating	"4.5"
raw_ranking	"4.111736297607422"
reviews	…
shopping_type	"none"
subcategory	…
subcategory_ranking	"#3 of 195 things to do in St…
subtype	…
tags	…
timezone	"Europe/Berlin"
website	"http://www.wasen.de"
web_url	"https://www.tripadvisor.com…
write_review	"https://www.tripadvisor.com…

Abbildung 56 - Attribute einer API-Abfrage auf POIs in Stuttgart. Hier sind ist das Beispiel „Neckar Park" abgebildet. Einige Stellen sind durch … verkürzt dargestellt.

Row ID	location...	names	latitudes	longitudes	num_re...	location_strings	primary...	raw_rankings	subcate...
Row0_1	2275103	Neckar Park	48.7973	9.21694	554	Stuttgart, Baden-Wurttemberg	Cultural Tours	4.111736297607422	Nature & Parks
Row0_2	6652123	Public Librar...	48.79016	9.183018	570	Stuttgart, Baden-Wurttemberg	Cultural Tours	4.164238929748535	Traveler Res...
Row0_3	4943582	Hohenpark ...	48.802185	9.171923	577	Stuttgart, Baden-Wurttemberg	Self-guided ...	4.072850227355957	Nature & Parks
Row0_4	8820363	Stuttgart Ch...	48.776966	9.178803	291	Stuttgart, Baden-Wurttemberg	Escape Games	4.098326683044434	Events
Row0_5	8820178	Cannstatter...	48.797546	9.216431	124	Stuttgart, Baden-Wurttemberg	4WD, ATV &...	3.6961510181427	Events
Row0_6	669189	Konigstrasse	48.780815	9.180469	1656	Stuttgart, Baden-Wurttemberg	Air Tours	3.769057035446167	Sights & Lan...
Row0_7	4081012	Rubble Hill	48.7653	9.131883	194	Stuttgart, Baden-Wurttemberg	4WD, ATV &...	4.004011154174805	Sights & Lan...
Row0_8	3258673	Killesbergturm	48.80392	9.16717	351	Stuttgart, Baden-Wurttemberg	Multi-day To...	4.033231735229492	Sights & Lan...
Row0_9	2281822	Mercedes B...	48.803284	9.238966	390	Stuttgart, Baden-Wurttemberg	Historical & ...	3.926877737045288	Sights & Lan...
Row0_10	7116780	Standseilba...	48.7518	9.145463	185	Stuttgart, Baden-Wurttemberg	Air Tours	4.048331260681152	Other
Row0_11	257371	Porsche Mus...	48.834465	9.151934	4595	Stuttgart, Baden-Wurttemberg	Water Tours	4.062851905822754	Museums
Row0_12	243380	Fernsehtur...	48.755753	9.190115	1149	Stuttgart, Baden-Wurttemberg	Water Tours	3.8201377391815...	Sights & Lan...
Row0_13	522407	Wilhelma Zo...	48.80533	9.20544	2257	Stuttgart, Baden-Wurttemberg	Transportati...	3.810035467147827	Zoos & Aqua...
Row0_14	243376	Landesmuse...	48.77709	9.179132	350	Stuttgart, Baden-Wurttemberg	Air Tours	3.8887951374053...	Nature & Parks
Row0_15	1850872	Grabkapelle ...	48.7821	9.268538	298	Stuttgart, Baden-Wurttemberg	Cultural Tours	4.0545878410339...	Outdoor Acti...
Row0_16	243381	Mercedes-B...	48.78837	9.23415	8156	Stuttgart, Baden-Wurttemberg	4WD, ATV &...	4.326462745666504	Sights & Lan...
Row0_17	669188	Solitude Palace	48.78686	9.084336	396	Stuttgart, Baden-Wurttemberg	Ports of Call...	3.822186231613159	Museums
Row0_18	243382	Staatsgalerie	48.78068	9.187137	583	Stuttgart, Baden-Wurttemberg	4WD, ATV &...	4.035303592681885	Sights & Lan...
Row0_19	2291761	Feuerseeplatz	48.77289	9.165439	261	Stuttgart, Baden-Wurttemberg	Multi-day To...	3.928828477859497	Sights & Lan...
Row0_20	311582	Palace Square	48.77798	9.18184	1642	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.9814164638519...	Museums
Row0_21	522630	Weissenhof ...	48.80076	9.17708	139	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.750913143157959	Sights & Lan...
Row0_22	8820232	Stuttgart Sp...	48.795883	9.219346	68	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.396502733230591	Sights & Lan...
Row0_23	2291780	Max-Eyth See	48.833237	9.210698	116	Stuttgart, Baden-Wurttemberg	Historical & ...	3.1910412311553...	Museums
Row0_24	2283944	Stuttgart St...	48.8052	9.189603	129	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.516503095626831	Sights & Lan...
Row0_25	2287121	Schweine-M...	48.78575	9.21986	291	Stuttgart, Baden-Wurttemberg	Historical & ...	3.3466989994049...	Sights & Lan...
Row0_26	598921	Schlossgarten	48.780933	9.182904	214	Stuttgart, Baden-Wurttemberg	Air Tours	3.5752127170562...	Sights & Lan...
Row0_27	2669011	Strotmanns ...	48.815296	9.213053	114	Stuttgart, Baden-Wurttemberg	Self-guided ...	3.733778953552246	Other
Row0_28	6784164	Johanneskir...	48.77349	9.164471	125	Stuttgart, Baden-Wurttemberg	Escape Games	3.775606632232666	Events
Row0_29	669171	Carl-Zeiss-Pl...	48.783104	9.186939	83	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.1018483638763...	Nature & Parks
Row0_30	669175	Kunstmuseu...	48.778202	9.177885	215	Stuttgart, Baden-Wurttemberg	Historical & ...	3.4202659130096...	Museums
Row0_31	1875301	Schillerplatz	48.77724	9.17845	197	Stuttgart, Baden-Wurttemberg	Historical & ...	3.7620136737823...	Museums
Row0_32	5400375	Weissenhof...	48.799816	9.177723	54	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.3360044956207...	Nature & Parks
Row0_33	2045925	Bahnhofsturm	48.78407	9.179034	233	Stuttgart, Baden-Wurttemberg	Historical & ...	3.4256389141082...	Concerts & S...
Row0_34	669186	New Castle ...	48.77796	9.181917	147	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.541975975036621	Sights & Lan...
Row0_35	8643460	Hohenheime...	48.70675	9.213819	57	Stuttgart, Baden-Wurttemberg	Historical & ...	3.435915231704712	Museums
Row0_36	519888	Collegiate C...	48.77655	9.177709	163	Stuttgart, Baden-Wurttemberg	Cultural Tours	3.6637213230133...	Museums
Row0_37	12154738	Flughafen S...	48.69291	9.20194	72	Stuttgart, Baden-Wurttemberg	Self-guided ...	3.2029402256011...	Nature & Parks
Row0_38	1870050	Mineralbad L...	48.7988	9.210235	254	Stuttgart, Baden-Wurttemberg	Escape Games	3.2418951988220...	Museums

Abbildung 57 - Ausschnitt der Tabelle aus POIs für das Reiseziel Stuttgart nach Filterung, Dimensionsreduktion und Validitätsprüfung

```
1.  def create_list_from_file(inputpath):
2.      """
3.      Creates a list from files in :param inputpath
4.      :return: list of files
5.      """
6.      print("gathering files...")
7.      country_list = []
8.      [country_list.append(file) for file in glob.glob1(inputpath, "*.csv")]
9.      return country_list
```

Abbildung 58 - Funktion create_list_from_file

```
1.  def create_category_dict(inputpath, file_list):
2.      """
3.      Creates a dictionary from file_list with the sum of all unique categories
4.      Used to create the indices for frequency distribution
5.      :param inputpath:
6.      :param file_list:
7.      :return:
8.      """
9.      print("parsing through files...")
10.     # iterate over files in file_list and save unique primary_categories in list
11.     category_list = []
12.     category_dict = {}
13.     for idx, val in enumerate(file_list):
14.         print("..." + file_list[idx])
15.         # read in CSV from file_list, limit to columns needed
16.         df = csv_processing(inputpath, file_list, idx)
17.
18.         # write all unique categories from df to list
19.         category_list.extend(df["primary_category"].unique())
20.
21.         # convert list with duplicates to dict with unique values
22.         category_dict = {value for value in category_list}
23.
24.     print("done.")
25.     return category_dict
```

Quellcode 12 - Funktion create_category_dict

```
1.   def csv_processing(inputpath, file_list, idx):
2.      """
3.      CSV processing prior to computing
4.      :param inputpath:
5.      :param file_list:
6.      :param idx:
7.      :return:
8.      """
9.      df = pd.read_csv("%s/%s" % (inputpath, file_list[idx]), usecols=["primary_category", "sub-
    category"])
10.
11.     # Fill column "primary_category" with "subcategory" if NaN.
12.     df["primary_category"] = df["primary_category"].fillna(df["subcategory"])
13.
14.     # convert all words to lowercase
15.     df["primary_category"] = df["primary_category"].str.lower()
16.
17.     # convert all words to lowercase
18.     df = df.drop("subcategory", 1)
19.
20.     return df
```

Quellcode 13 - Funktion csv_processing

```
1.  def count_value_occurences(inputpath, file_list, category_dict):
2.      """
3.      Creates frequency distribution for n most common (keyword extraction)
4.      Fills CSV with :param category_dict: as headers and freq_dist as values
5.      :param inputpath:
6.      :param file_list:
7.      :param category_dict:
8.      :return:
9.      """
10.
11.     # Write header to file
12.     with open("results.csv", "w", newline="\n") as outcsv:
13.         writer = csv.DictWriter(outcsv, fieldnames=category_dict)
14.         writer.writeheader()
15.         print("writing csv header...")
16.
17.     # Iterate through file_list and create frequency distribution
18.     for idx, val in enumerate(file_list):
19.         df = csv_processing(inputpath, file_list, idx)
20.         fdist = FreqDist(df["primary_category"])
21.         fdmc = fdist.most_common(len(fdist))
22.         fdmc_dict = (dict(fdmc))
23.
24.         # Write each fdmc_dict line to file
25.         with open("results.csv", "a", newline="\n") as output_file:
26.             writer = csv.DictWriter(output_file, category_dict)
27.             writer.writerow(fdmc_dict)
28.             print("writing data for %s..." % val)
29.
30.     # Create data frame from csv file
31.     df = pd.read_csv("results.csv")
32.
33.     # Replace NaN values with 0.0 and convert to int (thanks pandas)
34.     df.fillna(value=0, inplace=True)
35.     df = df.fillna(0.0).astype(int)
36.
37.     # Rename RowIndex to city
38.     df.index = [city.split("_")[0] for city in file_list]
39.
40.     # Write to file
41.     df.to_csv(r'results.csv', header=True)
```

Quellcode 14 - Funktion count_value_occurences

```
1.  def create_bar_plot(freq_subjects_as_list, freq_values_as_list_rounded, outputpath, target_lo-
    cation):
2.      """
3.      Create bar plot
4.      :param freq_subjects_as_list:
5.      :param freq_values_as_list_rounded:
6.      :param outputpath:
7.      :param target_location:
8.      :return:
9.      """
10.     fig, ax = plt.subplots()
11.
12.     # create bar plot from rounded values
13.     bars = ax.bar(np.arange(len(freq_values_as_list_rounded)), freq_values_as_list_rounded,
    color='black',
14.                   orientation="vertical")
15.
16.     # call autolabel to annotate
17.     autolabel(ax, bars)
18.
19.     # set x- and y-labels, title and make sure enough space is above the bars to annotate
20.     ax.set_ylim(top=freq_values_as_list_rounded[0] + 3)
21.     plt.xlabel("Categories")
22.     plt.ylabel("Frequency (%)")
23.     plt.title('frequency distribution of categories for %s' % target_location)
24.     plt.xticks(np.arange(len(freq_values_as_list_rounded)), freq_subjects_as_list, rotation=90)
25.
26.     # save
27.     plt.savefig("%s/%s_barplot.png" % (outputpath, target_location), bbox_inches='tight')
28.     plt.show()
```

Quellcode 15 - Funktion create_bar_plot

```
1.  def autolabel(ax, bars):
2.      """
3.      Attach a text label above each bar displaying its height
4.      :param ax:
5.      :param bars:
6.      :return:
7.      """
8.      for rect in bars:
9.          height = rect.get_height()
10.         ax.text(rect.get_x() + rect.get_width() / 2., height + 0.2,
11.                 height, rotation=90, fontsize=7,
12.                 ha='center', va='bottom')
```

Quellcode 16 - Funktion autolabel

```
1.  def create_wordcloud(freq_as_dict, outputpath, target_location):
2.      """
3.      Create a word cloud image (only lets you create from dictionary, not list...)
4.      :param freq_as_dict:
5.      :param outputpath:
6.      :param target_location:
7.      :return:
8.      """
9.      wordcloud = WordCloud(background_color="white", width=900, height=500,
    color_func=lambda *args, **kwargs: "black")
10.     wordcloud.generate_from_frequencies(freq_as_dict)
11.     wordcloud.to_file("%s/%s_wordcloud.png" % (outputpath, target_location))
12.     plt.imshow(wordcloud, interpolation='bilinear')
13.     plt.axis("off")
14.     plt.show()
```

Quellcode 17 - Funktion create_wordcloud

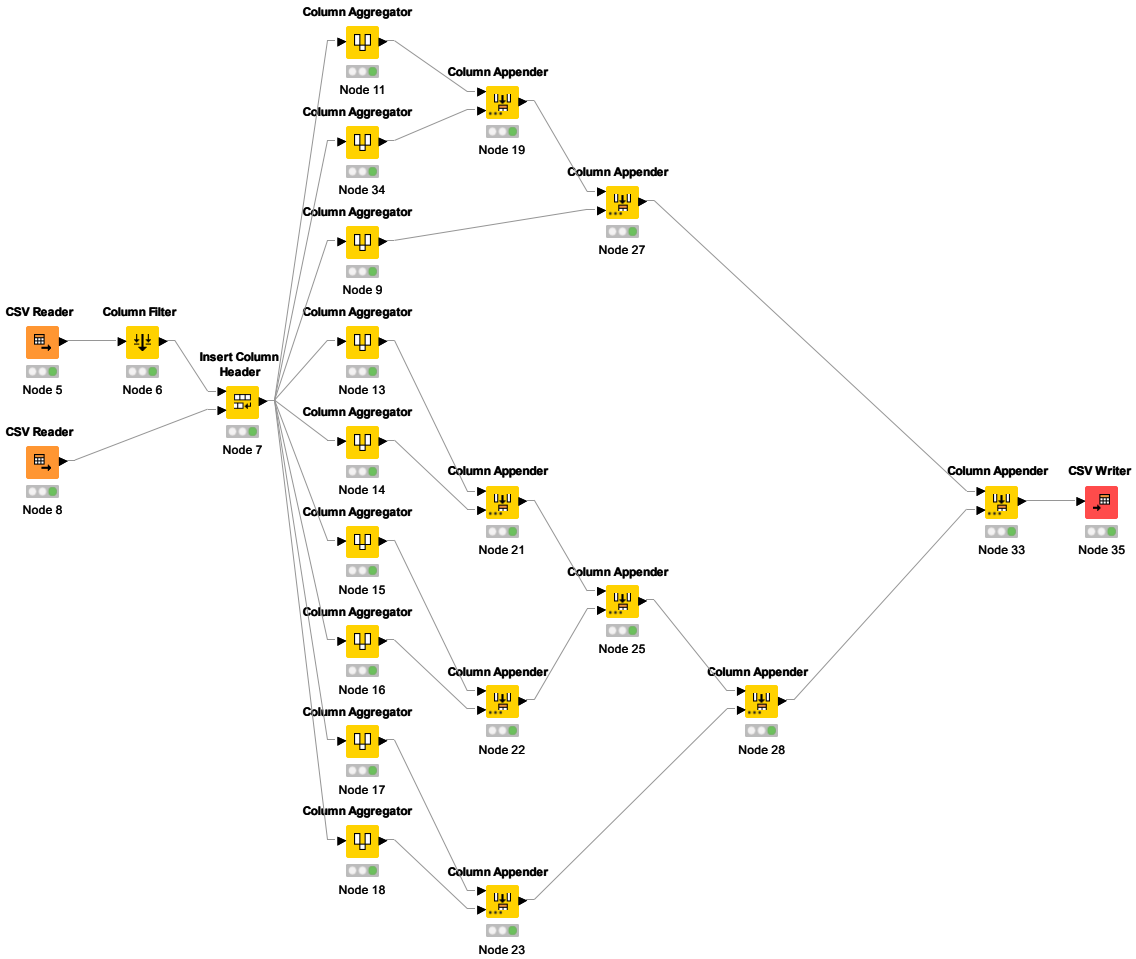


Abbildung 59 - Aggregation der Attribute passend zu den ausgewählten 9 Superkategorien

```
1.  for i, category in enumerate(categories):
2.      category_string = categories[i].replace(" ", "").lower()
3.
4.      photos = flickr.walk(mode="all", text=categories_tags[category], extras="url_c, views",
    sort="relevance")
5.      file_id = 0
6.      file_urls = []
7.
8.      print("/// Grabbing pictures for category: %s" % category)
9.      print("/// Category tags: %s" % categories_tags[category])
10.
11.     for j, photo in enumerate(photos):
12.         try:
13.             # Utilize "url_c" to get 800x600 sized pictures
14.             url = photo.get("url_c")
15.             file_urls.append(url)
16.
17.             request = urllib.request.urlretrieve(file_urls[j], "files/%s/%s_%i.jpg" % (category_string,
    category_string, file_id))
18.             request_printstring = f"Current filerequest: {request}"
19.             print(request_printstring)
20.
21.             image = Image.open("files/%s/%s_%i.jpg" % (category_string, category_string, file_id))
22.             image.save("files/%s/%s_%i.jpg" % (category_string, category_string, file_id))
23.
24.             file_id += 1
25.
26.         except TypeError:
27.             # In case url_c size is not available or another TypeError occurs, skip picture
28.             print("File Size Error, trying next picture")
29.
30.         # counter for images to grab for each category (n = file_id for each category)
31.         if file_id >= 1000:
32.             break
```

Quellcode 18 - Flickr Image Parsing Programmcode

```
1.  def display_sample_pictures(path, class_names, img_x, img_y):
2.      """
3.      Creates a 3x3 array of sample pictures.
4.      :param img_x: image width
5.      :param img_y: image height
6.      :param path: folder path of images
7.      :param class_names: name of classificator
8.      :return:
9.      """
10.     plt.figure(figsize=(10, 10))
11.     images = []
12.
13.     for sub_folder in path.iterdir():
14.         for file in sub_folder.iterdir():
15.             img = image.load_img(file, target_size=(img_x, img_y, 3))
16.             images.append(img)
17.             break
18.
19.     for i, item in enumerate(images):
20.         ax = plt.subplot(3, 3, i + 1)
21.         plt.imshow(item)
22.         plt.title(class_names[i])
23.     plt.show()
```

Quellcode 19 - Visualisierung eines Bildes passend zu jedem Klassifizierer

```
1.  Layer (type)                 Output Shape              Param #
2.  =================================================================
3.  conv2d (Conv2D)              (None, 298, 298, 64)      1792
4.  _________________________________________________________________
5.  max_pooling2d (MaxPooling2D) (None, 298, 298, 64)      0
6.  _________________________________________________________________
7.  dropout (Dropout)            (None, 298, 298, 64)      0
8.  _________________________________________________________________
9.  conv2d_1 (Conv2D)            (None, 296, 296, 128)     73856
10. _________________________________________________________________
11. max_pooling2d_1 (MaxPooling2 (None, 148, 148, 128)     0
12. _________________________________________________________________
13. dropout_1 (Dropout)          (None, 148, 148, 128)     0
14. _________________________________________________________________
15. conv2d_2 (Conv2D)            (None, 146, 146, 256)     295168
16. _________________________________________________________________
17. max_pooling2d_2 (MaxPooling2 (None, 18, 18, 256)       0
18. _________________________________________________________________
19. dropout_2 (Dropout)          (None, 18, 18, 256)       0
20. _________________________________________________________________
21. conv2d_3 (Conv2D)            (None, 16, 16, 512)       1180160
22. _________________________________________________________________
23. max_pooling2d_3 (MaxPooling2 (None, 1, 1, 512)         0
24. _________________________________________________________________
25. dropout_3 (Dropout)          (None, 1, 1, 512)         0
26. _________________________________________________________________
27. flatten (Flatten)            (None, 512)               0
28. _________________________________________________________________
29. dense (Dense)                (None, 512)               262656
30. _________________________________________________________________
31. dense_1 (Dense)              (None, 64)                32832
32. _________________________________________________________________
33. dense_2 (Dense)              (None, 9)                 585
34. =================================================================
35. Total params: 1,847,049
36. Trainable params: 1,847,049
37. Non-trainable params: 0
```

Quellcode 20 - Zusammenfassung des Modells mittels des Befehls model.summary()

```
Epoch 1/25
81s 179ms/step - loss: 0.3156 - accuracy: 0.2537 - val_loss: 0.2927 - val_accuracy: 0.3422
Epoch 2/25
79s 177ms/step - loss: 0.2773 - accuracy: 0.3885 - val_loss: 0.2611 - val_accuracy: 0.4372
Epoch 3/25
83s 184ms/step - loss: 0.2505 - accuracy: 0.4708 - val_loss: 0.2414 - val_accuracy: 0.4983
Epoch 4/25
83s 185ms/step - loss: 0.2286 - accuracy: 0.5181 - val_loss: 0.2201 - val_accuracy: 0.5628
Epoch 5/25
82s 183ms/step - loss: 0.2121 - accuracy: 0.5654 - val_loss: 0.2096 - val_accuracy: 0.5572
Epoch 6/25
95s 211ms/step - loss: 0.2013 - accuracy: 0.5989 - val_loss: 0.2017 - val_accuracy: 0.5817
Epoch 7/25
95s 212ms/step - loss: 0.1913 - accuracy: 0.6169 - val_loss: 0.1934 - val_accuracy: 0.6078
Epoch 8/25
81s 179ms/step - loss: 0.1802 - accuracy: 0.6469 - val_loss: 0.1945 - val_accuracy: 0.6050
Epoch 9/25
83s 183ms/step - loss: 0.1775 - accuracy: 0.6538 - val_loss: 0.1816 - val_accuracy: 0.6378
Epoch 10/25
81s 181ms/step - loss: 0.1657 - accuracy: 0.6721 - val_loss: 0.1820 - val_accuracy: 0.6278
Epoch 11/25
83s 184ms/step - loss: 0.1625 - accuracy: 0.6812 - val_loss: 0.1802 - val_accuracy: 0.6478
Epoch 12/25
84s 188ms/step - loss: 0.1542 - accuracy: 0.6958 - val_loss: 0.1822 - val_accuracy: 0.6350
Epoch 13/25
82s 183ms/step - loss: 0.1498 - accuracy: 0.7042 - val_loss: 0.1758 - val_accuracy: 0.6561
Epoch 14/25
82s 182ms/step - loss: 0.1442 - accuracy: 0.7183 - val_loss: 0.1716 - val_accuracy: 0.6678
Epoch 15/25
86s 191ms/step - loss: 0.1398 - accuracy: 0.7344 - val_loss: 0.1667 - val_accuracy: 0.6722
Epoch 16/25
90s 200ms/step - loss: 0.1343 - accuracy: 0.7381 - val_loss: 0.1801 - val_accuracy: 0.6556
Epoch 17/25
85s 189ms/step - loss: 0.1311 - accuracy: 0.7428 - val_loss: 0.1653 - val_accuracy: 0.6867
Epoch 18/25
85s 190ms/step - loss: 0.1264 - accuracy: 0.7581 - val_loss: 0.1626 - val_accuracy: 0.6933
Epoch 19/25
82s 182ms/step - loss: 0.1221 - accuracy: 0.7688 - val_loss: 0.1714 - val_accuracy: 0.6650
Epoch 20/25
86s 190ms/step - loss: 0.1169 - accuracy: 0.7746 - val_loss: 0.1691 - val_accuracy: 0.6689
Epoch 21/25
84s 186ms/step - loss: 0.1137 - accuracy: 0.7833 - val_loss: 0.1636 - val_accuracy: 0.6878
Epoch 22/25
83s 185ms/step - loss: 0.1137 - accuracy: 0.7894 - val_loss: 0.1569 - val_accuracy: 0.6961
Epoch 23/25
81s 181ms/step - loss: 0.1062 - accuracy: 0.8015 - val_loss: 0.1643 - val_accuracy: 0.6944
Epoch 24/25
80s 178ms/step - loss: 0.1046 - accuracy: 0.8033 - val_loss: 0.1699 - val_accuracy: 0.6822
Epoch 25/25
88s 196ms/step - loss: 0.1022 - accuracy: 0.8092 - val_loss: 0.1682 - val_accuracy: 0.7067
```

Quellcode 21 - Auswertung der Modellgenauigkeit (Konsolenausgabe)

```
1.  def result_graphs(history, epochs):
2.      """
3.      Creates accuracy and loss graphs for training and validation images
4.      :param history: model data used to draw graph
5.      :param epochs: epoch count used to plot x axis
6.      :return:
7.      """
8.
9.      acc = history.history['accuracy']
10.     val_acc = history.history['val_accuracy']
11.
12.     loss = history.history['loss']
13.     val_loss = history.history['val_loss']
14.
15.     epochs_range = range(epochs)
16.
17.     plt.figure(figsize=(8, 8))
18.     plt.subplot(1, 2, 1)
19.     plt.plot(epochs_range, acc, label='Training Accuracy')
20.     plt.plot(epochs_range, val_acc, label='Validation Accuracy')
21.     plt.legend(loc='lower right')
22.     plt.title('Training and Validation Accuracy')
23.
24.     plt.subplot(1, 2, 2)
25.     plt.plot(epochs_range, loss, label='Training Loss')
26.     plt.plot(epochs_range, val_loss, label='Validation Loss')
27.     plt.legend(loc='upper right')
28.     plt.title('Training and Validation Loss')
29.     plt.show()
```

Quellcode 22 - Visualisierung der Modellergebnisse

```
1.  # Create new CSV file with header by inserting "file_id" at [0] to class_names array
2.  csv_header = class_names.copy()
3.  csv_header.insert(0, "file_id")
4.
5.  # Set delimiter to ";", no idea why DictWriter doesnt accept delimiter param
6.  csv.register_dialect('pipes', delimiter=';')
7.
8.  # CSV file writer, uses csv_header array for file initialization
9.  with open("files/image_classification_results.csv", "w", newline="\n") as outcsv:
10.     writer = csv.DictWriter(outcsv, fieldnames=csv_header, dialect="pipes")
11.     writer.writeheader()
12.     print("writing csv header...")
13.
14. # Folder images_samples/ is a duplicate of images/ because file_permissions are blocked
    while the model is active...
15. sample_images_folder = pathlib.Path("files/images_samples/")
16.
17. # Iterate through folders to get classification results for all images
18. for file in sample_images_folder.iterdir():
19.     img = image.load_img(file, target_size=(img_x, img_y, 3))
20.     img = image.img_to_array(img)
21.     img = img / 255
22.
23.     predictions = model.predict(img.reshape(1, img_x, img_y, 3))
24.     score = nn.softmax(predictions[0])
25.     results_arr = [file.name]
26.
27.     for i, class_name in enumerate(class_names):
28.         results_arr.append(100 * np.max(score[i]))
29.
30.     with open("files/image_classification_results.csv", "a", newline="\n") as outcsv:
31.         writer = csv.writer(outcsv, delimiter=";")
32.         writer.writerow(results_arr)
33.         print(f"writing {results_arr} to file...")
34.
35. # Convert CSV to JSON for frontend readability
36. csv_file = pd.DataFrame(pd.read_csv("files/image_classification_results.csv", sep=";",
    header=0, index_col=False))
37. print("converting csv to json...")
38. csv_file.to_json("files/image_classification_results.json", orient="records", date_for-
    mat="epoch", double_precision=10,
39.                 force_ascii=True, date_unit="ms", default_handler=None)
```

Quellcode 23 - Export der Modellergebnisse

testid	choice_01	choice_02	choice_03	choice_04	choice_05	choice_06	result_1	result_2	result_3	agreed
1	shopping_732.jpg	entertainment_521.jpg	shopping_847.jpg	sportsandactivities_441.jpg	entertainment_360.jpg	culturededucation_537.jpg	rome	london	istanbul	TRUE
2	entertainment_448.jpg	nature_207.jpg	shopping_501.jpg	entertainment_437.jpg	nightlife_461.jpg	nature_306.jpg	rome	london	moscow	FALSE
3	sightseeing_727.jpg	entertainment_470.jpg	wellness_650.jpg	sightseeing_531.jpg	wellness_582.jpg	nature_325.jpg	newyork	istanbul	prague	TRUE
4	foodanddrink_764.jpg	wellness_876.jpg	wellness_514.jpg	entertainment_128.jpg	sightseeing_827.jpg	sightseeing_14.jpg	newyork	buenosaires	istanbul	TRUE
5	entertainment_310.jpg	foodanddrink_654.jpg	wellness_740.jpg	sightseeing_150.jpg	culturededucation_850.jpg	culturededucation_145.jpg	rome	amsterdam	london	FALSE
6	entertainment_815.jpg	nightlife_560.jpg	nightlife_909.jpg	culturededucation_327.jpg	sightseeing_905.jpg	entertainment_75.jpg	rome	london	istanbul	TRUE
7	wellness_534.jpg	shopping_126.jpg	shopping_378.jpg	shopping_469.jpg	nightlife_630.jpg	foodanddrink_294.jpg	rome	london	newyork	FALSE
8	culturededucation_351.jpg	nightlife_219.jpg	wellness_850.jpg	sightseeing_559.jpg	sightseeing_660.jpg	nature_388.jpg	rome	london	reykjavik	TRUE
9	culturededucation_981.jpg	nature_592.jpg	sportsandactivities_818.jpg	wellness_67.jpg	entertainment_941.jpg	wellness_530.jpg	istanbul	prague	rome	FALSE
10	culturededucation_129.jpg	nature_101.jpg	sightseeing_808.jpg	foodanddrink_473.jpg	nature_919.jpg	nature_66.jpg	istanbul	rome	moscow	TRUE
11	entertainment_837.jpg	entertainment_802.jpg	nightlife_272.jpg	nightlife_224.jpg	wellness_284.jpg	nature_256.jpg	newyork	buenosaires	istanbul	TRUE
12	sightseeing_959.jpg	nature_485.jpg	foodanddrink_461.jpg"	sportsandactivities_244.jpg	sightseeing_492.jpg	nature_686.jpg	hawaii	newyork	berlin	TRUE
13	sportsandactivities_635.jpg	shopping_95.jpg	wellness_361.jpg	nature_943.jpg	wellness_259.jpg	nature_571.jpg	singapore	miami	seoul	TRUE
14	culturededucation_15.jpg	shopping_523.jpg	wellness_873.jpg	shopping_977.jpg	shopping_980.jpg	entertainment_550.jpg	singapore	buenosaires	rome	TRUE
15	foodanddrink_384.jpg	sightseeing_531.jpg	culturededucation_530.jpg	shopping_517.jpg	shopping_62.jpg	wellness_153.jpg	london	newyork	istanbul	TRUE
16	wellness_588.jpg	sightseeing_23.jpg	shopping_737.jpg	sportsandactivities_841.jpg	nightlife_32.jpg	shopping_259.jpg	prague	berlin	rio	TRUE
17	culturededucation_984.jpg	nature_5.jpg	nightlife_462.jpg	entertainment_101.jpg	sightseeing_702.jpg	entertainment_950.jpg	dublin	prague	london	TRUE
18	foodanddrink_83.jpg	sightseeing_221.jpg	culturededucation_686.jpg	wellness_855.jpg	shopping_568.jpg	wellness_112.jpg	istanbul	newyork	seoul	FALSE
19	entertainment_368.jpg	nightlife_798.jpg	nature_980.jpg	shopping_990.jpg	nature_579.jpg	shopping_74.jpg	prague	mallorca	sanfrancisco	TRUE
20	nightlife_328.jpg	nightlife_877.jpg	nightlife_129.jpg	nightlife_379.jpg	nightlife_292.jpg	nature_914.jpg	buenosaires	sydney	rio	TRUE

21	sightsee-ing_861.jpg	well-ness_161.jpg	well-ness_253.jpg	night-life_318.jpg	well-ness_141.jpg	entertain-ment_938.jpg	istan-bul	singa-pore	sydney	TRUE
22	sportsan-dactivi-ties_266.jpg"	"na-ture_340.jpg"	"na-ture_143.jpg"	"sportsan-dactivi-ties_850.jpg"	"well-ness_990.jpg"	"well-ness_219.jpg"	lasve-gas	berlin	stutt-gart	FALSE
23	entertain-ment_885.jpg	night-life_293.jpg	night-life_850.jpg	night-life_5.jpg	foodandd-rink_586.jpg	entertain-ment_812.jpg	bueno-saires	san-fran-cisco	prague	TRUE
24	shop-ping_141.jpg	sightsee-ing_344.jpg	well-ness_796.jpg	culturean-deduca-tion_902.jpg	na-ture_749.jpg	well-ness_766.jpg	istan-bul	rome	seoul	TRUE
25	sightsee-ing_682.jpg	shop-ping_931.jpg	sightsee-ing_170.jpg	sightsee-ing_653.jpg	entertain-ment_237.jpg	culturean-deduca-tion_137.jpg	rome	macau	mosco w	FALSE
26	entertain-ment_849.jpg	na-ture_438.jpg	sportsan-dactivi-ties_366.jpg	night-life_931.jpg	entertain-ment_522.jpg	na-ture_987.jpg	bueno-saires	istan-bul	newyor k	FALSE
27	foodandd-rink_125.jpg	sightsee-ing_512.jpg	na-ture_161.jpg	shop-ping_987.jpg	well-ness_628.jpg	na-ture_344.jpg	istan-bul	seoul	newyor k	TRUE

Tabelle 13 - Detaillierte Ergebnisse der Nutzerauswahl