Gottfried Wilhelm Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik

Entwicklung eines wissensbasierten Meta-Lernmodells

Developing a Knowledge-Based Meta-Learning Model

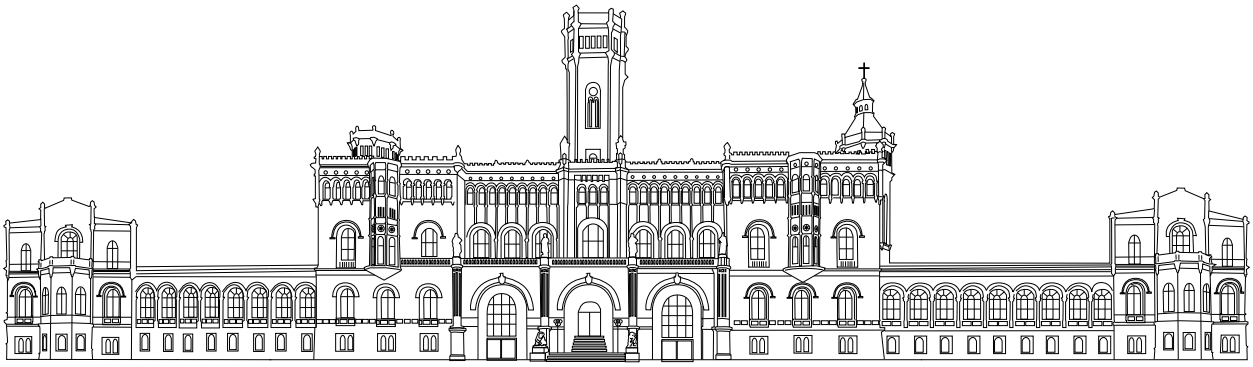
Wissenschaftliche Arbeit zur Erlangung des
Bachelor of Science in Computer Science

VON

Mustafa Tantawi
Matrikelnummer: 10008249
E-Mail: mustafa.tantawi@stud.uni-hannover.de

Erstprüfer: Prof. Dr. Sören Auer
Zweitprüfer: Prof. Dr. Marius Lindauer
Betreuer: Dr. Ildar Baimuratov

20. März 2024



Erklärung der Selbstständigkeit

Ich, Mustafa Tantawi, erkläre, dass die vorliegende Arbeit mit dem Titel „ Entwicklung eines wissensbasierten Meta-Lernmodells“ und die darin enthaltenen Arbeiten meine eigenen sind. Ich bestätige, dass:

- Diese Arbeit im Rahmen einer Bewerbung für einen Forschungsabschluss an dieser Universität durchgeführt wurde.
- Wenn ein Teil dieser Arbeit bereits für einen Abschluss oder eine andere Qualifikation an dieser Universität oder einer anderen Einrichtung eingereicht wurde, ist dies deutlich angegeben.
- An Stellen, bei denen ich andere veröffentlichte Arbeiten konsultiert habe, ist dies immer deutlich angegeben.
- Wenn ich aus der Arbeit anderer zitiert habe, ist die Quelle immer angegeben. Mit Ausnahme solcher Zitate ist die vorliegende Arbeit vollständig meine eigene Arbeit.
- Ich habe alle wichtigen Hilfsquellen angegeben.

Mustafa Tantawi

Signatur: _____

Datum: _____

Danksagung

Ich möchte meinem Betreuer Dr. Ildar Baimuratov meinen tiefsten Dank für seine unschätzbare Anleitung, Unterstützung und Hilfe während der gesamten Dauer dieser Bachelorarbeit aussprechen. Ihr Fachwissen, ihre Geduld und ihr unermüdliches Engagement haben maßgeblich dazu beigetragen, die Richtung dieser Forschungsarbeit zu bestimmen und mich durch ihre Komplexität zu navigieren.
Darüber hinaus danke ich meiner Familie von ganzem Herzen für ihre ständige Unterstützung, ihr Verständnis und ihren festen Glauben an meine Fähigkeiten. Ihre ständige Unterstützung war eine ständige Quelle der Motivation und Inspiration, die mir geholfen hat, Herausforderungen zu überwinden und meine akademischen Ziele zu verfolgen.

Zusammenfassung

Entwicklung eines wissensbasierten Meta-Lernmodells

Durch die Automatisierung der Modellentwicklung wird der Prozess effizienter und auch für Benutzer ohne umfassende technische Kenntnisse zugänglich. Eine Möglichkeit, die Entwicklung von Modellen zu automatisieren, ist Meta-Learning. Meta-Learning ist ein Bereich des automatisiertes maschinellen Lernens, der sich mit der Frage beschäftigt, wie Modelle oder Algorithmen entwickelt werden können, um effizienter zu lernen oder sich anzupassen, basierend auf Erfahrungen aus früheren Lernprozessen. Um die Anpassungsfähigkeit und Leistungsfähigkeit von Meta-Learning-Modellen zu verbessern, ist eine effiziente Repräsentation und Speicherung von Wissen erforderlich. Zunächst ist zu erwähnen, dass Knowledge Graphen für die Speicherung von Meta-Learning geeignet sind. Insbesondere der Open Research Knowledge Graph (ORKG) enthält bereits einige geeignete Graphmodelle, die mehr Flexibilität bieten als der Rational Graph.
Unser Problem ist: Wie kann Wissen für Meta-Learning skalierbar repräsentiert und gespeichert werden?
Wir können dies erklären, indem wir existierende maschinelle Lernschemen analysieren, um ihre Anwendbarkeit für Meta-Learning zu prüfen und die wichtigsten Meta-Features zu identifizieren, dann die Meta-Features und das Wissen über ML-Experimente in ORKG für eine bessere Repräsentation einfügen und schließlich ein Meta-Lernmodell unter Verwendung von ORKG-Daten entwickeln. Wir haben am Ende ein KNN-Modell entwickelt, das die nächsten Nachbarn des Datasets basierend auf den Meta-Features berechnet. Damit lassen sich relevante ORKG Contributions über ähnliche Datensätze hinweg finden. Am Ende haben wir auch Testdatensätze mit verschiedenen Werten erstellt, um die Genauigkeit zu berechnen.
Keywords: Meta Learing, ORKG, AutoML, knowledge representation, knowledge storage

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung:

1.1 Relevance

Die Relevanz dieser Bachelorarbeit wird durch die schnellen Fortschritte auf dem Gebiet des maschinellen Lernens und die zunehmende Komplexität von ML-Modellen unterstrichen. Automatisches maschinelles Lernen (AutoML) ermöglicht die Entwicklung von ML-Modellen ohne tiefere technische Kenntnisse. Der Prozess der Modellauswahl, der Optimierung von Hyperparametern und der Extraktion von Features wird automatisiert. Innerhalb dieses Konzepts nimmt das Meta-Lerning eine Schlüsselstellung ein, da es die Fähigkeit besitzt, Modelle auf verschiedene Datensätze zu generalisieren, was ihre Anpassungsfähigkeit und Leistungsfähigkeit erhöht.

Die Herausforderung besteht jedoch darin, dass sich die bisherige Forschungsarbeit im Bereich Meta-Learning hauptsächlich auf algorithmische Verbesserungen konzentriert hat, während die Repräsentation und Speicherung von Wissen für Meta-Learning-Modelle eher vernachlässigt wurde. Hier setzt die vorliegende Arbeit an, indem sie einen innovativen Ansatz zur skalierbaren und effizienten Wissensrepräsentation und -speicherung zur Verbesserung von Meta-Lernmodellen verfolgt.

Ein weiterer Aspekt, der die Relevanz dieser Arbeit unterstreicht, ist die zunehmende Bedeutung von Wissensgraphen und RDF-Technologien. Diese bieten eine semantische Struktur, die es ermöglicht, Wissen vernetzt darzustellen. Die Integration von Wissensgraphen in den Kontext von Meta-Learning verspricht nicht nur eine effektive Repräsentation von Wissen. Sie bietet auch die Möglichkeit, auf bestehende Forschungsergebnisse und Experimente zuzugreifen, um die Meta-Lernmodelle zu verbessern.

Insgesamt wird die Relevanz dieser Arbeit vor dem Hintergrund der aktuellen Entwicklungen im Bereich AutoML und dem wachsenden Bedarf an der Entwicklung innovativer Lösungen für die Repräsentation und Speicherung von Wissen in Meta-Learning-Anwendungen deutlich. Die Beantwortung der Forschungsfragen liefert einen Beitrag, der nicht nur theoretisch fundiert ist, sondern auch praktische Implikationen für die Weiterentwicklung maschineller Lernsysteme hat.

1.2 Zielsetzung/Forschungsfrage

Das Ziel dieser Bachelorarbeit ist es, einen Beitrag zur Entwicklung eines wissensbasierten Meta-Lernmodells zu leisten. Das Meta-Lernmodell basiert auf einer skalierbaren Repräsentation und Speicherung von Wissen.

1.3 Objektive Darstellung

Das Hauptziel dieser Bachelorarbeit ist es, einen Beitrag zur Entwicklung und Verbesserung von Meta-Learning-Modellen zu leisten, um Wissen auf effiziente Weise zu repräsentieren und zu speichern. Um dieses Gesamtziel zu erreichen, werden mehrere Teilziele verfolgt:

- **Analyse bestehender ML-Schemata:** Durch die Analyse bestehender ML-Schemata sollen Möglichkeiten zur optimierten Darstellung von Meta-Learning-Modellen als Grundlage für die Wissensintegration identifiziert werden.

- **Identifikation von Features für das Meta-Lernen:** Ziel ist es, gemeinsame Meta-Learning-Features zu identifizieren und in strukturierter Form darzustellen, um eine effektive Integration in Wissensgraphen zu ermöglichen.

- **Der Open Research Knowledge Graph (ORKG)** wird erweitert, indem Ressourcen mit Meta-Charakteristiken angereichert werden und Wissen zu ML-Experimenten hinzugefügt wird, um die Wissensbasis für Meta-Lernen zu erweitern.

- **Ein Meta-Learning-Modell implementieren:** Basierend auf dem in ORKG gespeicherten Wissen soll ein praktisches Meta-Learning-Modell entwickelt werden, das nicht nur die theoretischen Erkenntnisse der Arbeit demonstriert, sondern auch einen Beitrag zur Meta-Learning-Praxis leistet.

Die Arbeit gliedert sich in die folgenden Abschnitte:

- **Einführung:**
  Ich erkläre das Thema Meta-Learning, AutoML und Knowledga Graphen- und RDF-Technologien und erläutere die Motivation für meine Forschung und Ich stelle die Ziele meiner Arbeit vor und gebe einen Überblick über den Aufbau.

- **Hintergrund:**
  Ich erkläre grundlegende Konzepte und Techniken des Meta-Lernens, AutoML und Knowledga Graphen- und RDF-Technologien

- **Verwandte Arbeiten:**
  Ich recherchiere und evaluiere bestehende Meta-Learning-Schemata und verwandte Arbeiten über Meta learning.

- **Ansatz:**
  Ich identifiziere gemeinsame Meta-Lernmerkmale und stelle ihre strukturierte Darstellung vor und modelliere diese Meta-Features und die ML-Experimenten im ORKG.

- **Implementierung:**
  Ich stelle das entwickelte Meta Learning Modell vor, das auf dem Wissen des ORKG basiert.

- **Experimentelle Evaluierung:** Ich evaluiere die Funktionsweise und Leistungsfähigkeit des Modells anhand verschiedener Testdaten.

Kapitel 2

Hintergrund:

Diese Forschung ist eng verbunden mit den Bereichen AutoML (automatisiertes maschinelles Lernen) und Wissensgraphen. Um einen fundierten Überblick zu gewährleisten, werden in den folgenden Abschnitten grundlegende Definitionen und Konzepte aus diesen Bereichen vorgestellt. AutoML als Methodik zur Automatisierung des maschinellen Lernens spielt in unserer Untersuchung eine entscheidende Rolle. Ebenso sind Wissensgraphen und RDF (Resource Description Framework) von zentraler Bedeutung, da sie eine strukturierte und semantische Repräsentation von Wissen ermöglichen. Um diese Schlüsselkonzepte effektiv zu verstehen, laden wir Sie ein, sich mit den Definitionen und Grundlagen in den folgenden Abschnitten vertraut zu machen.

2.1 AutoML

Automated Machine Learning (AutoML) stellt Methoden und Prozesse bereit, um das Feld des Maschinellen Lernens auch für Personen ohne spezifische Expertise in diesem Bereich zugänglich zu machen. Ziel ist es, die Effizienz von Maschinellem Lernen zu steigern und die Forschung auf diesem Gebiet zu beschleunigen. Obwohl Maschinelles Lernen in den letzten Jahren bedeutende Fortschritte gemacht hat und in verschiedenen Disziplinen erfolgreich eingesetzt wird, bleibt der Erfolg stark von Experten für maschinelles Lernen abhängig.

Diese Experten sind typischerweise für eine Reihe von komplexen Aufgaben verantwortlich, darunter Datenverarbeitung und -bereinigung, Auswahl und Konstruktion von relevanten Features, Auswahl geeigneter Modellfamilien, Optimierung von Hyperparametern, Design neuronaler Netztopologien (insbesondere bei Anwendung von Deep Learning), Nachbearbeitung von Machine-Learning-Modellen sowie kriti-

sche Analyse der erzielten Ergebnisse.

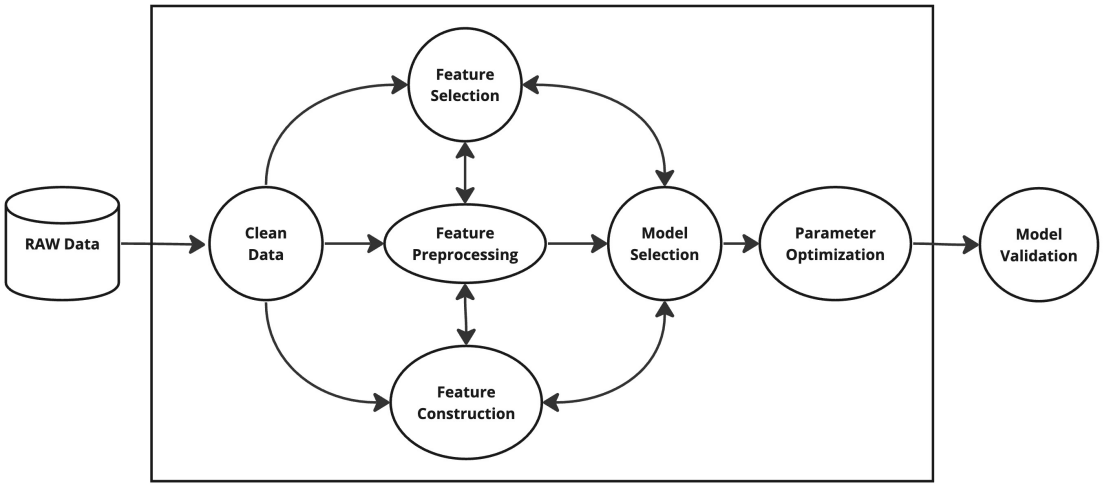


Abbildung 2.1: Source: R. Olson et. al. (2016) „Evaluation of Tree-based Pipeline Optimization Tool for Automating Data Science.“

Die Grafik von R. Olson et. al. zeigt, dass von den Rohdaten bis zur Modellerstellung der gesamte Prozess automatisiert abläuft (im Kasten). Alle weiteren Schritte laufen mit AutoML automatisch ab - ohne menschliches Zutun. Angesichts der wachsenden Komplexität dieser Aufgaben und der steigenden Anzahl von Anwendungen im Maschinellen Lernen entsteht ein Bedarf an standardisierten AutoML-Methoden, die ohne tiefgehende Expertise angewendet werden können. Der resultierende Forschungsbereich, der sich auf die fortschreitende Automatisierung des Maschinellen Lernens konzentriert, wird als AutoML bezeichnet.[1]

2.2 Meta-learning

Meta-Lerning hat sich zu einem wichtigen Bereich des maschinellen Lernens entwickelt. Die Grundidee ist die Entwicklung von Modellen, die ihr Wissen effizient anpassen und generalisieren. Diese Fähigkeit ist von entscheidender Bedeutung in Szenarien, in denen das Modell mit neuen, unbekannten Aufgaben konfrontiert wird, und ermöglicht es ihm, mit begrenzten Daten effizienter zu lernen.

[1]Quelle: AutoML Website https://www.automl.org/automl/

Wissensrepräsentation in Meta-Learning: Sie unterstreicht die Bedeutung einer effektiven Wissensrepräsentation für Meta-Learning-Aufgaben, weist jedoch darauf hin, dass weitere Untersuchungen erforderlich sind, um vollständig zu verstehen, wie Wissen in verschiedenen Meta-Learning-Szenarien gespeichert und genutzt werden kann. [3]

Welche Ansätze und Anwendungen gibt es beim Meta-Learning und wie unterscheiden sie sich voneinander?

1. **Model-based Ansatz:**
   Im modell-basierten Ansatz liegt der Fokus darauf, ein Modell zu erlernen, das sich schnell an neue Aufgaben anpassen kann. Dies beinhaltet das Training eines Modells, um zu verstehen, wie es seine Parameter für verschiedene Aufgaben aktualisieren kann. Klassische Modelle in diesem Ansatz umfassen Memory-Augmented Neural Networks, Meta Networks und Fast Weights. Der Ansatz basiert auf der Idee, dass das Modell selbst in der Lage sein sollte, schnell zu lernen und sich an neue Aufgaben anzupassen, indem es seine eigenen Parameter anpasst. Dies erfordert jedoch ein tiefes Verständnis des Modells und seiner Funktionsweise.

2. **Metric-based Ansatz:**
   Der metrik-basierte Ansatz konzentriert sich darauf, einen Metrikraum zu erlernen, in dem Aufgaben oder Beispiele verglichen werden können. Dieser Ansatz nutzt Methoden wie die k-nächste-Nachbarn-Algorithmen, k-NN-Klassifikator, k-Means-Clustering und Kerndichteschätzung. Klassische Modelle in diesem Ansatz umfassen Convolutional Siamese Neural Network, Matching Networks, Simple Embedding, Full Context Embeddings, Relation Network und Prototypical Networks.

3. **Optimization-based Ansatz:**
   Der optimierungs-basierte Ansatz dreht sich darum, einen Optimierungsalgorithmus zu erlernen, der die Parameter des Modells effizient für neue Aufgaben aktualisieren kann. Hierbei wird der Optimierungsalgorithmus explizit modelliert, oft unter Verwendung von Techniken wie LSTM Meta-Learner, MAML (Model-Agnostic Meta-Learning), First-Order MAML und Reptile. Klassische Modelle in diesem Ansatz umfassen LSTM Meta-Learner, MAML und Reptile. Der Ansatz basiert auf der Idee, dass das Modell in der Lage sein sollte, schnell zu lernen, wie es seine Parameter für neue Aufgaben aktualisieren kann, indem es einen effizienten Optimierungsalgorithmus verwendet. Dies erfordert jedoch ein tiefes Verständnis des Modells und des Optimierungsalgorithmus.

Diese Ansätze unterscheiden sich in ihren grundlegenden Strategien, um Modelle zu befähigen, schnell neue Aufgaben zu erlernen. Während metrik-basierte Ansätze sich auf Ähnlichkeit und Vergleich konzentrieren, legen modell-basierte Ansätze den Schwerpunkt auf die Anpassung des Modells selbst, und optimierungs-basierte Ansätze betonen das Erlernen effizienter Parameteraktualisierungen für neue Aufgaben. Jeder Ansatz verfügt über eine Reihe von klassischen Modellen und Techniken, die auf seine spezifische Strategie zugeschnitten sind

Die Herausforderung beim Meta-Learning besteht darin, systematisch und datengestützt aus früheren Erfahrungen zu lernen. Zunächst müssen Metadaten gesammelt werden, die frühere Lernaufgaben und früher gelernte Modelle beschreiben. Dazu gehören die genauen Algorithmuskonfigurationen, die zum Trainieren der Modelle verwendet wurden, einschließlich der Einstellungen der Hyperparameter, der Pipeline Zusammensetzungen und/oder der Netzwerkarchitekturen, die resultierenden Modellbewertungen, wie Genauigkeit und Trainingszeit, die gelernten Modellparameter, wie die trainierten Gewichte eines neuronalen Netzes, sowie messbare Eigenschaften der Aufgabe selbst, die auch als Meta-Features bezeichnet werden. Zweitens müssen wir aus diesen Metadaten lernen, um Wissen zu extrahieren und zu übertragen, das uns hilft, optimale Modelle für neue Aufgaben zu finden.

Die Charakterisierungen (Meta-Features) der vorliegenden Aufgabe sind eine weitere reichhaltige Metadatenquelle. Jede Aufgabe $t_j \in T$ wird mit einem Vektor $m(t_j) = (m_{j,1}, ..., m_{j,K})$ von K Meta-Features $m_{j,k} \in M$, der Menge aller bekannten Meta-Features, beschrieben. Dies ermöglicht die Definition einer Ähnlichkeitsmaßnahme für Aufgaben basierend auf, beispielsweise, der euklidischen Distanz zwischen $m(t_i)$ und $m(t_j)$, wodurch die Übertragung von Informationen von den ähnlichsten Aufgaben auf eine neue Aufgabe t_{new} ermöglicht wird. Darüber hinaus kann zusammen mit früheren Bewertungen P ein Meta-Learner L trainiert werden, um die Leistung $P_{i,new}$ von Konfigurationen θ_i auf einer neuen Aufgabe t_{new} vorherzusagen:

$$P_{i,new} = L(m(t_{new}), P_{i,1}, ..., P_{i,j}, ..., P_{i,N})$$

Diese Formel stellt die Vorhersage der Leistung $P_{i,new}$ einer Konfiguration θ_i auf einer neuen Aufgabe t_{new} dar, basierend auf den Meta-Merkmalen $m(t_{new})$ und vorherigen Bewertungen $P_{i,j}$ auf N ähnlichen Aufgaben. Der Meta-Learner L wird so trainiert, dass er die Beziehung zwischen den Meta-Merkmalen und der Leistung der Konfigurationen bei den Aufgaben erlernt, wodurch die Übertragung von Wissen und Vorhersagen von früheren Aufgaben auf neue Aufgaben in einem Meta-Learning-

Rahmen ermöglicht wird. [3]

Das Konzept des Meta-Mining umfasst den Einsatz von Meta-Lerntechniken zur Verbesserung der Planung und Optimierung von Data-Mining-Workflows [15]. Es wird in Data Mining Workflow-Planern eingesetzt, um den Planer während des Workflow-Planungsprozesses zu leiten [15]. Darüber hinaus wird in einer Studie von [19] ein einheitlicher Rahmen für die Bewertung und den Vergleich neu aufkommender Meta-Mining-Techniken vorgestellt, der die Bedeutung von Meta-Mining im Kontext von Data Mining und sein Potenzial für großflächige Anwendungen hervorhebt [19].

Der Unterschied zwischen Meta-Mining und Meta-Learning liegt in ihren spezifischen Anwendungen. Meta-Mining konzentriert sich auf den Einsatz von Meta-Learning bei der Planung und Optimierung von Data-Mining-Arbeitsabläufen, während Meta-Learning das umfassendere Konzept des Lernens und der schnellen Anpassung an neue Aufgaben unter Nutzung früherer Lernerfahrungen umfasst. Während Meta-Mining also eine spezifische Anwendung von Meta-Learning im Bereich des Data Mining ist, hat Meta-Learning einen allgemeineren Anwendungsbereich und zielt darauf ab, die Leistung von Lernalgorithmen durch die Nutzung von Wissen über Lernprozesse zu verbessern [15] [19].

2.3 Knowledge graphs

Ein Knowledge Graph ist eine Wissensdatenbank. Sie besteht aus Entitäten (Objekten) und deren Beziehungen. Er ermöglicht eine vernetzte und semantische Darstellung von Informationen, die das Verständnis von Daten durch Maschinen und die Herstellung von Beziehungen zwischen verschiedenen Konzepten erleichtert. Wissensgraphen werden häufig zur Unterstützung der semantischen Suche, des maschinellen Lernens und der Integration von Daten aus verschiedenen Bereichen verwendet.[2]

Bei RDF(Resource Description Framework) handelt es sich um eine Spezifikation des W3C, die als Grundlage für das Semantic Web dient. Sie bietet eine standardisierte Methode zur Darstellung von Informationen in Form von Tripeln. Diese bestehen aus Subjekt, Prädikat und Objekt. RDF ermöglicht die Verknüpfung von Ressourcen über das Web und fördert die Interoperabilität zwischen verschiedenen Anwendungen und Datenquellen. rdfRDFResource Description Framework **Kernbegriffe des RDF:**

[2]Quelle: W3C Semantic Web-Website https://www.w3.org/RDF/

1. **Tripel:**
   In rdf RDF (Resource Description Framework) ist ein Tripel die grundlegende Dateneinheit, um Informationen zu repräsentieren. Es besteht aus drei Teilen: dem Subjekt, dem Prädikat und dem Objekt. Das Subjekt gibt die Ressource an, auf die sich das Tripel bezieht. Das Prädikat gibt die Art der Beziehung zwischen dem Subjekt und dem Objekt an. Das Objekt ist die eigentliche Information oder der Wert, der mit dem Subjekt in Verbindung steht. Die Verbindung dieser drei Elemente bildet ein Tripel, das die grundlegende Struktur für die Repräsentation von Wissen in RDF darstellt.

2. **RDF-Graph:**
   Ein RDF-Graph ist eine Sammlung von Tripeln, die eine vernetzte Struktur von Wissen repräsentieren. Jedes Tripel im Graphen trägt zur Darstellung von Beziehungen zwischen Ressourcen bei. Der Graph bietet eine visuelle Darstellung der Verbindungen und Abhängigkeiten zwischen den verschiedenen Ressourcen.

3. **RDF Schema (RDFS):**
   Eine Erweiterung von RDF, die die Definition von Klassen und Eigenschaften erlaubt und damit semantische Strukturen unterstützt und, die zusätzliche Möglichkeiten zur semantischen Modellierung bietet.

4. **Ontologien:**
   Formalisierte Beschreibungen von Wissen, die die Beziehungen und Regeln in einer bestimmten Domäne definieren.Die W3C Web Ontology Language (OWL) ist eine Sprache des Semantic Web. Sie ermöglicht es, umfangreiches und komplexes Wissen über Dinge, Gruppen von Dingen und Beziehungen zwischen Dingen darzustellen. OWL ist eine Sprache, die auf Computerlogik basiert. Daher kann das in OWL ausgedrückte Wissen von Computerprogrammen genutzt werden, um z.B. die Konsistenz dieses Wissens zu überprüfen oder implizites Wissen explizit zu machen. OWL-Dokumente, so genannte Ontologien, können im World Wide Web veröffentlicht werden. Sie können auf andere OWL-Ontologien verweisen oder von anderen OWL-Ontologien referenziert werden.

RDF dient sowohl als Datenmodell als auch als Austauschformat für die Beschreibung von Ressourcen und den Beziehungen zwischen ihnen. Knowledge Graphs nutzen RDF zur strukturierten Darstellung von Wissen und zur Erstellung von vernetzten

Datenmodellen. Der Einsatz von RDF in Knowledge Graphen ermöglicht eine einheitliche, standardisierte und interoperable Darstellung von Wissen. Dies führt zu einer verbesserten semantischen Verarbeitung. Insgesamt stellen RDF und Knowledge Graphs eine leistungsfähige Grundlage für die semantische Modellierung von Daten dar und ermöglichen es, Wissen im Semantic Web effektiv zu organisieren, zu verknüpfen und zu analysieren.

Vorteile des Graphdatenmodells gegenüber dem Relationalen Datenmodell:

Die Speicherung von Meta-Features in einem Graphdatenmodell bietet klare Vorteile gegenüber dem traditionellen Relationalen Datenmodell. Im Gegensatz zu relationalen Datenbanken zeichnen sich Graphdatenbanken durch die Darstellung komplexer Beziehungen und Abhängigkeiten aus, die in Meta-Features vorhanden sind. Die Flexibilität des Graphmodells ermöglicht eine natürlichere Darstellung von vernetzten Daten, wie den komplexen Beziehungen zwischen verschiedenen Meta-Features. Graphdatenbanken eignen sich besonders gut für Szenarien, in denen die Beziehungen zwischen Entitäten genauso wichtig sind wie die Entitäten selbst. Darüber hinaus ermöglicht die schemalose Natur des Graphmodells die Anpassung an sich ändernde Datenstrukturen und macht es somit anpassungsfähiger für die dynamische Natur von Meta-Feature-Repräsentationen. Diese Flexibilität und Ausdrucksstärke des Graphdatenmodells tragen zu intuitiveren Abfragen und effizienteren Traversierungen der vernetzten Meta-Feature-Informationen bei, was ein ganzheitliches und umfassendes Verständnis der zugrunde liegenden Daten fördert.

2.4 The Open Research Knowledge Graph

Das Ziel des Open Research Knowledge Graph (ORKG) in Bezug auf wissenschaftliche Literatur ist die strukturierte und semantische Darstellung wissenschaftlicher Beiträge. Durch die Erstellung eines Wissensgraphen können Informationen so organisiert und verknüpft werden, dass sie sowohl für Maschinen als auch für Menschen lesbar sind. ORKG zielt darauf ab, den Zugang zu wissenschaftlicher Literatur zu verbessern, indem es einen komprimierten Überblick über den Stand der Technik in verschiedenen Forschungsbereichen bietet. Er bietet neuartige Such- und Unterstützungsdienste für Forscher, die es ihnen ermöglichen, relevante Informationen leicht zu finden und Verbindungen zwischen verschiedenen Konzepten und Artefakten zu untersuchen. ORKG erleichtert auch die Wiederverwendung und Integration von Wissen, so dass Forscher leichter auf bestehendem Wissen aufbauen und gemeinsam an neuen Entdeckungen arbeiten können.

ORKG Entities: Der Open Research Knowledge Graph (ORKG) umfasst verschiedene Entitätstypen, darunter:

1. **Papers:** Wissenschaftliche Artikel bilden eine zentrale Entität im ORKG. Diese umfassen Forschungsbeiträge, Studien und Veröffentlichungen aus verschiedenen wissenschaftlichen Disziplinen.

2. **Datasets:** Datensätze werden als eigenständige Entitäten im ORKG repräsentiert. Diese können experimentelle Daten, Umfragen oder andere Arten von Forschungsdaten umfassen.

Um die Qualität des Open Research Knowledge Graph (ORKG) zu erstellen und zu erhalten, wird eine Kombination aus manuellen und automatisierten Techniken verwendet. [3]
Im Folgenden sind einige der verwendeten Techniken aufgeführt:

1. **Abstract Annotator:**
   Ein Abstract Annotator Tool wird verwendet, um Schlüsselinformationen aus den Zusammenfassungen wissenschaftlicher Artikel zu extrahieren. Techniken der natürlichen Sprachverarbeitung und des maschinellen Lernens werden eingesetzt, um die Zusammenfassungen effizient zu annotieren. Dies hilft bei der Extraktion wichtiger Informationen wie Forschungsprobleme, Methoden und Materialien, die zur Lösung des Problems verwendet werden.

2. **Text Summarization:**
   Techniken zur Textzusammenfassung werden eingesetzt, um die Inhaltserschließung zu automatisieren. Dieser Prozess fasst den Inhalt, wie z. B. die Methodik, zusammen, wobei die wichtigsten Informationselemente erhalten bleiben.

3. **Question Answering:**
   In ORKG wurde ein System zur Beantwortung von Fragen entwickelt. Es bildet natürlichsprachliche Abfragen auf den Graphen ab und findet die Antwort auf die Abfrage. Dies hilft bei der Gewinnung von Wissen und angemessenen Antworten auf wissenschaftliche Fragen.

4. **Knowledge Integration:**
   Die strukturierte und semantische Beschreibung von Wissen in ORKG ermöglicht

[3] Quelle: Improving Access to Scientific Literature with Knowledge Graphs https://www.degruyter.com/document/doi/10.1515/bfp-2020-2042/html

eine vereinfachte Wiederverwendung von Wissen. Der ORKG bietet eine webbasierte Schnittstelle (REST API), die das Laden von ORKG-Inhalten in Datenanalyseumgebungen wie Jupyter Notebooks ermöglicht. Dies erleichtert die Erstellung von domänenspezifischen Anwendungen und Visualisierungen.

5. **Extraction from Survey and Review Articles:**
   Extraktion aus Umfrage- und Übersichtsartikeln: Bestehendes wissenschaftliches Wissen aus Fachartikeln wird genutzt, um das ORKG aufzufüllen. Diese Artikel geben einen Überblick über den Stand der Technik in einem bestimmten Forschungsbereich und werden manuell geprüft, um eine hohe Qualität zu sichern.

Der Prozess, Informationen aus wissenschaftlicher Literatur zu extrahieren, kombiniert sowohl manuelle als auch automatisierte Techniken. Im Fall des Open Research Knowledge Graph (ORKG) ist eine Mischung aus Crowd- und Experten-Sourcing für die Erstellung der Darstellung des Wissensgraphen im Einsatz. Dies stellt sicher, dass die Repräsentation die Qualität aufweist, die für neuartige Such- und Unterstützungsdienste für Forscher erforderlich ist.
Der Wissensgraph stellt wissenschaftliche Beiträge in strukturierter und semantischer Weise dar. Das bedeutet, dass die Informationen in Knoten und Kanten organisiert sind, die sinnvolle Verbindungen zwischen verschiedenen Konzepten oder Artefakten ermöglichen. Der Vorteil der Verwendung eines Wissensgraphen ist, dass die Informationen sowohl von Maschinen als auch von Menschen gelesen werden können. Diese strukturierte und semantische Darstellung hat mehrere Vorteile.[4]

Welche Vorteile bietet die strukturierte und semantische Darstellung wissenschaftlicher Beiträge?

1. **Readability by Machines and Humans:**
   Informationen, die in einem Wissensgraphen dargestellt sind, können sowohl von Maschinen als auch von Menschen gelesen werden. Dies ermöglicht ein besseres Verständnis und eine bessere Interpretation des Inhalts.

2. **Interoperability:**
   Wissensgraphen ermöglichen, Informationen über verschiedene Systeme und

[4]Quelle: Improving Access to Scientific Literature with Knowledge Graphs https://www.degruyter.com/document/doi/10.1515/bfp-2020-2042/html

Plattformen hinweg zu integrieren und auszutauschen. Dadurch wird die Zusammenarbeit und der Wissensaustausch zwischen verschiedenen Domänen gefördert.

3. **Skalierbarkeit:**
   Die Wissensgraphen sind in der Lage, eine große Anzahl von Forschungsbeiträgen aufzunehmen und können im Laufe der Zeit erweitert und aktualisiert werden. Diese Skalierbarkeit stellt sicher, dass neues Wissen in die Wissensgraphen integriert und mit dem bereits vorhandenen Wissen effizient verknüpft werden kann.

4. **Vergleichbarkeit und Bewertung:**
   Eine bessere Vergleichbarkeit und Bewertung wissenschaftlicher Ergebnisse wird durch strukturierte und semantisch reichhaltige Beschreibungen von Forschungsbeiträgen ermöglicht. Forscher sind in der Lage, ähnliche Studien, Methoden und Ergebnisse leicht zu identifizieren und zu analysieren, was die Transparenz und die Reproduzierbarkeit der Forschung fördert.[5]

[5]Quelle: Improving Access to Scientific Literature with Knowledge Graphs https://www.degruyter.com/document/doi/10.1515/bfp-2020-2042/html

Kapitel 3

Verwandte Arbeiten:

Im Rahmen dieser Arbeit soll ein umfassender Überblick über die bestehende Forschung auf dem Gebiet der Wissensrepräsentation und -speicherung im Kontext von AutoML gegeben werden. Insbesondere konzentriert sich dieser Abschnitt auf zwei zentrale Aspekte: existierende ML-Schemata und gemeinsame Merkmale des Meta-Lernens. Die Wissensrepräsentation ist von entscheidender Bedeutung für die Unterstützung und Verbesserung des Entwicklungsprozesses von Modellen des maschinellen Lernens. Daher werden in diesem Abschnitt relevante Arbeiten untersucht, die sich mit etablierten ML-Schemata und charakteristischen Merkmalen des Meta-Lernens befassen. Durch die Analyse dieser Arbeiten wollen wir Erkenntnisse gewinnen, die als Grundlage für die Weiterentwicklung unseres wissensbasierten Meta-Lernmodells dienen können.
Wir untersuchten die Meta-LearningsPapers, um die häufigsten Meta-Features zu sammeln. Diese Meta-Features werden zur Charakterisierung von Texte und zur Empfehlung von Vektordarstellungen für Textklassifizierungsaufgaben auf der Grundlage der Leistung ähnlicher Aufgaben verwendet[14] und die ML-Schema Papers, um die umfangreichsten Schema zu finden. In diesem Papers wurde ein Überblick über das ML-Schema gegeben, eine Ontologie der höchsten Ebene, die eine Reihe von Klassen, Eigenschaften und Einschränkungen für die Darstellung und den Austausch von Informationen über Algorithmen des maschinellen Lernens, Datensätze und Experimente bereitstellt.[18]

3.1 Meta-features

Meta-Features sind ein entscheidender Aspekt von AutoML, und in dieser Arbeit werden sechs verschiedene Arten von Meta-Features besprochen[27, Conclusion]:

- **einfache** Einfache Meta-Features umfassen grundlegende Merkmale des Datensatzes.
- **statistische:** statistische Meta-Merkmale umfassen Maße wie Mittelwert, Standardabweichung und Varianz.
- **informationstheorethische:** Informationstheoretische Meta-Features beruhen auf Entropie, bedingten Wahrscheinlichkeiten und gemeinsamen Wahrscheinlichkeiten
- **komplexe**: Komplexitäts-Meta-Merkmale erfassen die Komplexität des Datensatzes.
- **modelbasierte:** Modellbasierte Meta-Features werden von der Leistung bestimmter Modelle auf dem Datensatz abgeleitet,
- **landmarker:** Landmarker-Meta-Features werden von Landmarking-Techniken abgeleitet.

Diese verschiedenen Arten von Meta-Features bieten wertvolle Einblicke in die Eigenschaften von Aufgaben des maschinellen Lernens und spielen eine entscheidende Rolle bei der Entwicklung effektiver AutoML-Systeme. Diese Meta-Merkmale liefern Informationen über die Eigenschaften des Datensatzes, die dazu verwendet werden können, seine Eigenschaften zu verstehen und die Auswahl und Konfiguration von maschinellen Lernalgorithmen zu lenken.[22]

Das Papier „Meta-Learning: A Survey“ diskutiert die Verwendung von Meta-Features im Zusammenhang mit der Auswahl von Algorithmen und der Optimierung von Hyperparametern [27] Meta-Merkmale werden aus den Merkmalen des Datensatzes oder der Aufgabe abgeleitet und dienen dazu, Informationen über die Komplexität, Verteilung und andere Eigenschaften der Daten zu liefern.[27] Meta-Features spielen eine entscheidende Rolle bei der Auswahl von Algorithmen, da sie Einblicke in die Eigenschaften des Datensatzes geben, die bei der Entscheidung helfen können, welcher Algorithmus wahrscheinlich gut abschneidet. Durch die Analyse von Metamerkmalen können Metamodelle erstellt werden, um den am besten geeigneten Algorithmus für eine bestimmte Aufgabe zu empfehlen. Diese Metamodelle können mit Techniken

wie Boosted und Bagged Trees erstellt werden, die nachweislich genaue Vorhersagen liefern. In ähnlicher Weise werden Metamerkmale auch bei der Hyperparameteroptimierung verwendet, um die Suche nach optimalen Hyperparameterkonfigurationen zu leiten. Durch die Analyse der Metamerkmale einer Aufgabe können Metamodelle die effektivsten Hyperparametereinstellungen für einen bestimmten Algorithmus empfehlen. Dies kann dazu beitragen, die Leistung und Effizienz von Modellen des maschinellen Lernens zu verbessern.[5] Insgesamt ermöglicht die Verwendung von Metamerkmalen bei der Auswahl von Algorithmen und der Optimierung von Hyperparametern einen fundierteren und automatisierten Ansatz für maschinelles Lernen. Durch die Nutzung von Dateneigenschaften liefern Metamerkmale wertvolle Informationen, die die Auswahl von Algorithmen und Hyperparametern leiten können, was zu einer verbesserten Leistung und Effizienz führt.

Das Papier „Automated Machine Learning for Big Industrial Data: A Meta-Learning Approach“ von Garouani et al. präsentiert ein Framework, das Meta-Features nutzt, um die Komplexität von Datensätzen zu charakterisieren und die automatische Auswahl von Algorithmen des maschinellen Lernens und Hyperparametern zu ermöglichen. Dabei werden 42 Meta-Features in drei Hauptklassen eingeteilt: allgemeine Maße, statistische Maße und informationstheoretische Maße. Zudem wird das Konzept eines Meta-Modells als zentrale Komponente des Rahmens eingeführt, das die Eigenschaften eines Datensatzes auf die erwartete Leistung von Algorithmen des maschinellen Lernens abbildet und auf Basis der Meta-Features trainiert wird Das Papier beschreibt den Trainingsprozess des Metamodells, bei dem verschiedene Leistungsmaße (wie Genauigkeit, Präzision, Wiedererkennung und F1-Score) für verschiedene Klassifizierer berücksichtigt werden. Das Metamodell wird trainiert, um die Leistung von Algorithmen des maschinellen Lernens auf der Grundlage der Metamerkmale des Datensatzes vorherzusagen.[7].

Der Papier „Characterizing classification datasets: a study of meta-features for meta-learning“ befasst sich mit den Herausforderungen im Bereich des Meta-Lernens, insbesondere mit der fehlenden Standardisierung von Meta-Features. Um dieses Problem anzugehen, schlägt der Artikel eine systematische Kategorisierung und Taxonomie vor. Die Reproduzierbarkeit wird ebenfalls betont und Lösungen für bisher vernachlässigte Aspekte werden vorgestellt. Ein Meta-Feature Extractor (MFE) wird eingeführt, um die Berechnung von Meta-Features zu standardisieren. Zukünftige Forschungsbereiche umfassen die Erweiterung der Taxonomie, die Verbesserung der Interpretierbarkeit von Meta-Merkmalen und die empirische Evaluierung von Cha-

rakterisierungsoptionen im Meta-Learning.[22]

Der Preprint „Meta Learning of Text Representations“ stellt eine neue Meta-Learning-Methode für Text-Mining-Aufgaben vor, die einen neuartigen Ansatz zur Automatisierung der Auswahl von Textrepräsentationen für Textklassifizierungs-Pipelines bietet. Die Studie definiert einen umfassenden Satz von 72 Meta-Features zur Charakterisierung von 81 Textkorpora, die allgemeine Features, Korpushärte-Maße, Domain-Breite-Maße und Features, die direkt aus dem Rohtext extrahiert wurden, umfassen. Durch umfangreiche Experimente mit 60 verschiedenen Textrepräsentationen für mehr als 80 Textmining-Aufgaben konnten die vorgeschlagenen Meta-Features Textmining-Aufgaben erfolgreich charakterisieren und die Machbarkeit einer AutoML-Lösung für Textmining demonstrieren. Diese Arbeit stellt einen bedeutenden Fortschritt auf dem Gebiet des Text Mining dar und bietet eine wertvolle Ressource für Forscher und Praktiker, um die Effizienz und Effektivität von Textklassifizierungspipelines durch Meta-Lernen zu verbessern. [17]

In Tabelle 3.1 und 3.2 werden verschiedene Meta Features zusammen mit die Subklassen (DataCharacteristic und FeatureCharacteristic) des ML-Schemas und ihre Beschreibungen vorgestellt.

Meta Features mit Subklassen	**Beschreibung**	**Papers**
(DataCharacteristic) Number of instances	The total number of instances in the data set	[27] [22] [12] [8]
(DataCharacteristic) Number of features	The total number of features in the data set	[27] [14] [7] [12] [26]
(DataCharacteristic) Number of Classes	The total number of classes in the data set	[27] [22] [12] [8]
(DataCharacteristic) Normal Entropy	Measures to improve model efficiency and performance by focussing on the most important features.	[27] [4] [12] [14]
(DataCharacteristic) Class Entropy	Measures to ensure fair and accurate representation of all classes in the modelling process	[27] [8] [12] [14]
(DataCharacteristic) Standard Deviation Ratio (sdRatio)	sdRatio is often used to evaluate the quality of signals in terms of noise	[27] [22] [12] [14]

Tabelle 3.1: Meta-Features von Datasets werden als Subklassen von :DatasetCharacteristic hinzugefügt.

Meta Features mit Subklassen	**Beschreibung**	**Papers**
(FeatureCharacteristic) Covariance	Measures to determine the extent of the change in variables	[4] [27] [12] [22]
(FeatureCharacteristic) (min/max/mean/median)	Statistical measures that provide insights into the distribution of characteristic values	[22] [8] [12] [14]
(FeatureCharacteristic) Correlation	Measures of the relationship or dependency between different characteristics	[4] [12] [27] [22]
(FeatureCharacteristic) Skewness	Measures of the asymmetry and skewness of the distribution of the characteristic values	[4] [27] [22] [8] [12][14]
(FeatureCharacteristic) Kurtosis	Measures the Thickness of the tails of a probability distribution.	[27] [22] [8] [12][14]
(FeatureCharacteristic) Mutual Information	Measures of the amount of information that can be obtained about a random variable by observing another random variable	[27] [22] [4] [12]

Tabelle 3.2: Meta-Features von Datasets werden als Subklassen von :FeatureCharacteristic hinzugefügt

3.2 Machine Learning Schemes:

Die Entwicklung des ML-Schemas wurde durch die Notwendigkeit motiviert, bestehende Ontologien für maschinelles Lernen und andere relevante Darstellungen zu standardisieren. Diese Ontologien wurden für eine Reihe von spezifischen Zwecken entwickelt und folgen manchmal inkompatiblen Designprinzipien, was zu unterschiedlichen und schwer interoperablen Strukturen führt. Mehrere Ontologien wurden vorgeschlagen, um den Bereich des maschinellen Lernens zu modellieren, wie Onto-DM[26], DMOP, Exposé und das MEX-Vokabular. Diese Ontologien sollen eine eindeutige Interpretierbarkeit und Reproduzierbarkeit von maschinellen Lernexperimenten gewährleisten. Keine der existierenden Ontologien deckt jedoch den Bereich des maschinellen Lernens vollständig ab und unterstützt alle Anforderungen an die Repräsentation und Kodierung von maschinellen Lernexperimenten.[18]

Dieser Abschnitt über verwandte Arbeiten gibt einen Überblick über bestehende Ontologien im Bereich des maschinellen Lernens in diesem Paper „ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies“[18] und hebt die einzigartigen Beiträge und Interoperabilitätsziele der ML-Schema-Ontologie hervor. Das vollständige Dokument sollte konsultiert werden für eine detailliertere Analyse. Die ML-Schema Ontologie wurde von der W3C Machine Learning Schema Community Group entwickelt. Sie soll einen High-Level-Standard für die Darstellung von maschinellen Lernexperimenten in einer prägnanten, eindeutigen und maschinell verarbeitbaren Weise bereitstellen. Die Ontologie ist kompakt, aber dennoch umfassend genug. Sie ist leicht erweiterbar. Die Ontologie ist offen für weitere Erweiterungen und Verknüpfungen mit anderen Ressourcen sowie für die Unterstützung der vertikalen und horizontalen Interoperabilität zwischen verschiedenen Umgebungen für maschinelles Lernen. Die in der ML Schema Ontology modellierten Eigenschaften wie „achieves“, „executes“, „hasInput“ und „hasOutput“ ermöglichen die Darstellung von Beziehungen zwischen verschiedenen Entitäten, die an maschinellen Lernexperimenten beteiligt sind, z.B. Run, Tasks, Implementation und Data. Die Ontologie ist mit verwandten Ontologien für maschinelles Lernen, einschließlich der OntoDM Core Ontology, abgestimmt, um Kompatibilität und Interoperabilität zwischen verschiedenen domänenspezifischen Ontologien zu gewährleisten.[18, S. 2-4]

Meta-Features können zur Darstellung von ML-Schema in einer Menge von Möglichkeiten verwendet werden. In unserer Arbeit untersuchen wir das „ML-Schema: An interchangeable format for description of machine learning experiments“ und es hat meh-

rere Komponenten, die ein ML-Schema zu bauen können und diese Komponenten hat einige Eigenschaften, die für Meta-Features im ersten Teil in der Arbeit gegeben werden. Die Meta-features 3.1 und 3.2 wie Skewness, Anzahl der Features, Kurtosis, Anzahl der Instances, Anzahl der Klassen, die Korrelation, die Kovarianz, die mittlere absolute Abweichung (mad), das Minimum (min), das Maximum (max), der Mittelwert, der Median, das Verhältnis der Standardabweichung (sdRatio), die Klassen-Entropie, die normalisierte Entropie und die Mutual Information sind statistische Eigenschaften von Datensätzen, die verwendet werden können, um die Struktur und die Komplexität der Daten zu charakterisieren und zu verstehen.

Das Paper „Exposé: An Ontology for Data Mining Experiments“ [26] ist eine wissenschaftliche Arbeit von Joaquin Vanschoren und Larisa Soldatova. Der Beitrag stellt Exposé vor, eine Ontologie, die entwickelt wurde, um Machine-Learning-Versuche in standardisierter Weise zu beschreiben und einen kollaborativen Ansatz zur Analyse von Lernalgorithmen zu unterstützen. Der Schwerpunkt liegt auf dem Entwurf einer Ontologie, um Metadaten von Experimenten auszudrücken und mit der Welt zu teilen. Das Papier diskutiert auch die Verwendung von Ontologien in anderen wissenschaftlichen Disziplinen, das Design von Ontologien für Data Mining und die Exposé Ontologie selbst, einschließlich ihrer Top-Level-Klassen und Eigenschaften. [26].

Dieses Papier [26] stellt die ML Schema (MLS) Ontologie vor, die entwickelt wurde, um die Darstellung von maschinellen Lernexperimenten zu standardisieren. Die MLS Ontologie bietet ein zentrales Vokabular für die Beschreibung von Algorithmen, Aufgaben, Implementierungen und Implementierungsergebnissen des Maschinellen Lernens, das mit bestehenden ML Ontologien harmonisiert wird, um die Interoperabilität zu erhöhen. Darüber hinaus werden verwandte Ontologien wie OntoDM und DMOP diskutiert, die eine Rolle im Meta-Mining spielen, einer Erweiterung des Meta-Learnings zur Analyse kompletter Data-Mining-Prozesse. Die Entwicklung von MLS zielt darauf ab, die Verbreitung inkompatibler ML-Ontologien zu verhindern und die Interoperabilität zwischen bestehenden Ontologien zu erhöhen, mit der Flexibilität, andere domänenspezifische Ontologien im Bereich des maschinellen Lernens und des Data Mining abzubilden.

Die PROV-O Ontologie [12] ist ein Standard für die Darstellung von Provenance-Informationen. Provenance bezieht sich in diesem Zusammenhang auf Metadaten, die die Geschichte und Herkunft von Daten beschreiben, einschließlich wie sie erstellt wurden, woher sie stammen und wie sie im Laufe der Zeit verändert wurden. PROV-O

ist Teil der PROV-Familie von Spezifikationen, die den Austausch von Provenance-Informationen in heterogenen Umgebungen wie dem Web ermöglichen sollen. PROV-O wird im Data Mining eingesetzt, um Datenanalyseprozesse einschließlich Datentransformation, Algorithmenanwendung und Modellbildung zu verfolgen und zu dokumentieren. Durch den Einsatz von PROV-O sind Forscher und Data Scientists in der Lage, eine transparente und nachvollziehbare Aufzeichnung ihrer Data Mining Aktivitäten zu erstellen, was für die Reproduzierbarkeit, die Validierung und die Verifizierung von entscheidender Bedeutung ist. Dies ist von entscheidender Bedeutung für die Reproduzierbarkeit, Validierung und Verifizierung. Sie ermöglicht es anderen, die während einer Analyse durchgeführten Schritte nachzuvollziehen, die Studie zu replizieren oder auf der Arbeit aufzubauen, indem sie sich auf die Ursprünge und Transformationen verlassen können.
Die Ontologie bietet eine Vielzahl von Klassen und Eigenschaften, die verwendet werden können, um verschiedene Provenance-Konzepte wie Agenten (z.B. Personen, Organisationen), Aktivitäten (z.B. Datenverarbeitungsschritte) und Entitäten (z.B. Datensätze, Algorithmen) zu repräsentieren. PROV-O kann z.B. verwendet werden, um zu zeigen, dass ein bestimmter Datensatz durch einen bestimmten Data-Mining-Prozess erzeugt wurde, der von einem bestimmten Forscher oder einem automatisierten Agenten durchgeführt wurde. [26]
Obwohl PROV-O nicht direkt im Kontext des Papiers „An Ontology for Data Mining Experiments“ von Joaquin Vanschoren und Larisa Soldatova diskutiert wird, stimmen die Prinzipien der Aufzeichnung und des Austauschs detaillierter Metadaten über Experimente des maschinellen Lernens mit den Zielen von Provenance Ontologien wie PROV-O überein. Die in diesem Beitrag beschriebene Ontologie unterstützt die genaue Erfassung und den Austausch von Data Mining Experimenten und Workflows. Dies ergänzt die Provenance Traceability, die durch PROV-O ermöglicht wird.

3.3 Research Gaps and Analyse:

Um die Frage zu beantworten, ob es eine Lücke bei den Meta-Features gibt, die das Schema des maschinellen Lernens repräsentieren, ist es wichtig, die Rolle des Meta-Lernens beim Verständnis der Eigenschaften von Lernaufgaben und der Leistung von Modellen des maschinellen Lernens zu berücksichtigen. Meta-Learning zielt darauf ab, die genauen Eigenschaften von Lernaufgaben zu bestimmen, die eine Methode für eine bestimmte Meta-Region geeignet machen, ausgedrückt durch die Komponenten, die zur Dominanz bestimmter Regionen im Metaraum beitragen [2]. Dies beinhaltet die Verwendung von Meta-Merkmalen, die Dateneigenschaften beschreiben, um eine effiziente Modellauswahl im Rahmen des automatischen maschinellen Lernens (AutoML) zu ermöglichen [10] Darüber hinaus beinhaltet Meta-Lernen das Erlernen einer allgemeinen Datenrepräsentation und die Anpassung an neue Aufgaben mit wenigen Trainingsbeispielen [29].

Die Verwendung von Meta-Features beim maschinellen Lernen ist von entscheidender Bedeutung für Aufgaben wie die Multi-Label-Klassifizierung, bei der das Meta-Learning eingesetzt wird, um die Meta-Informationen eines maschinellen Lernsystems zu lernen [28]. Darüber hinaus zeigt die Entwicklung neuartiger Meta-Merkmale für die automatische Auswahl von Modellen für maschinelles Lernen bei der Erkennung von Anomalien die Bedeutung von Meta-Merkmalen für die Verbesserung von Aufgaben des maschinellen Lernens [10]. Darüber hinaus unterstreicht die Verwendung eines meta-invarianten Merkmalsraums beim Reinforcement Learning für die Kontrolle von Bearbeitungsdeformationen die Anwendung von Meta-Merkmalen in spezifischen Bereichen [30]. Dies beinhaltet die Verwendung von Meta-Merkmalen, die Dateneigenschaften beschreiben, um eine effiziente Modellauswahl im Rahmen des automatischen maschinellen Lernens (AutoML) zu ermöglichen [10]. Darüber hinaus beinhaltet Meta-Lernen das Erlernen einer allgemeinen Datenrepräsentation und die Anpassung an neue Aufgaben mit wenigen Trainingsbeispielen [29].

Das Repräsentationslernen beim Meta-Lernen ist ebenfalls von entscheidender Bedeutung, wie die Verwendung von Auto-Encodern als Feature-Learning-Tool in verschiedenen Arbeiten zum 3D-Form-Lernen zeigt [16]. Darüber hinaus zeigt das Lernen von kontinuierlichen, vorzeichenbehafteten Abstandsfunktionen für die Formdarstellung die Bedeutung des Repräsentationslernens bei Meta-Lernaufgaben [16]. Darüber hinaus befasst sich das Lernen von Tensordarstellungen für das Meta-Lernen mit der Beobachtung, dass Trainingsaufgaben oft heterogen sind und zusätzliche aufgabenspezifische beobachtbare Nebeninformationen enthalten, was den Bedarf an

umfassenden darstellungsbasierten Modellen für das Meta-Lernen unterstreicht [6].

Insgesamt geben die Papers in der Referenzen einen umfassenden Einblick in die Rolle von Meta-Features und Repräsentationslernen in maschinellen Lernverfahren. Sie zeigen die Bedeutung von Meta-Features für das Verständnis der Eigenschaften von Lernaufgaben, für eine effiziente Modellauswahl und für die Verbesserung von maschinellen Lernaufgaben in verschiedenen Bereichen.

Im Zusammenhang mit dem ML-Schema 3.1 können diese Meta-Features 3.2 verwendet werden, um die in ML-Experimenten verwendeten Datensätze zu beschreiben. Das ML-Schema ist eine Ontologie auf oberster Ebene, die eine Reihe von Klassen, Eigenschaften und Einschränkungen für die Darstellung und den Austausch von Informationen über ML-Algorithmen, Datensätze und Experimente bietet. Die Meta-Features können mit den Datensätzen im ML-Schema verknüpft werden und liefern eine umfassende Beschreibung der Daten, die für Aufgaben wie die Auswahl von Algorithmen, die Abstimmung von Hyperparametern und die Reproduzierbarkeit von Experimenten verwendet werden kann.

Kapitel 4

Ansatz

In den vorangegangenen Kapiteln wurden verschiedene Forschungsarbeiten auf dem Gebiet der Wissensrepräsentation und des Meta-Learnings untersucht. Der Schwerpunkt liegt dabei auf der Analyse von Meta-Features und ML-Schemata, die für die Entwicklung von Modellen des maschinellen Lernens entscheidend sind. Die Untersuchung umfasst die Beschreibung von Meta-Features sowie die Analyse existierender ML-Ontologien und deren Kompatibilität. Diese Analyse liefert wichtige Erkenntnisse, die als Grundlage für die Weiterentwicklung wissensbasierter Meta-Learning-Modelle dienen können.

4.1 Erweiterung des ML-Schemas für Meta-Lernen

Um Features mit Datensätzen zu verknüpfen, wird die Eigenschaft „:hasFeature“ im erweiterten ML-Schema eingeführt. Diese Eigenschaft ermöglicht es, einzelne Features mit den entsprechenden Datensätzen zu verknüpfen. Durch die Verwendung von „:hasFeature“ können Analysen und Modelle genauer auf bestimmte Features innerhalb eines Datensatzes ausgerichtet werden, was die Effizienz und Genauigkeit von maschinellen Lernverfahren verbessert. Features können auch Meta-Features haben. Ich schlage vor, die ML-Schema zu erweiteren, um die ML-Schema für die Meta learning zu erweitern.
Diese Meta-Feature 3.1 und 3.2 können auf verschiedene Weise verwendet werden, um ein maschinelles Lernschema (ML-Schema) zu repräsentieren. Ich schlage vor, zusätzliche Meta-Features zu verwenden, um das ML-Schema darzustellen:

1. **Kurtosis:** Sie kann dabei helfen zu erkennen, ob es Ausreißer oder Extremwerte in den Daten gibt, die sich auf die Leistung des Modells auswirken könnten.

2. **Skewness:** kann dabei helfen, die Form der Verteilung zu verstehen und zu erkennen, ob die Verteilung von einer Normalverteilung abweicht.

3. **MAD (Mean Absolute Deviation):** Die MAD misst die durchschnittliche absolute Abweichung der Datenpunkte vom Mittelwert. Sie gibt Auskunft über die Datenstreuung und ist weniger anfällig für Ausreißer als die Standardabweichung.

4. **Min, Max, Mean, Median:** Hierbei handelt es sich um grundlegende deskriptive Statistiken, die Aufschluss über die zentrale Richtung und den Bereich der Daten geben. Sie helfen dabei, die Verteilung und den Umfang der Features zu verstehen.

5. **sdRatio (Standard Deviation Ratio):** Sie ist das Verhältnis zwischen der Standardabweichung und dem Mittelwert. Sie kann Auskunft über die Variabilität im Verhältnis zum Durchschnittswert geben.

6. **Class Entropy:** Sie kann bei Klassifizierungsaufgaben nützlich sein, um die Gleichgewichte oder Ungleichgewichte in der Verteilung der Klassen zu verstehen.

7. **Normal Entropy:** Sie hilft dabei, die Vielfalt der Werteverteilung zu verstehen.

8. **Mutual Information:** Die gegenseitige Information ist ein Maß für die Menge an Information, die sich aus der Kenntnis des Wertes einer Variablen in Bezug auf eine andere Variable ergibt. Sie kann für die Auswahl von Features verwendet werden und gibt an, wie gut ein Feature die Zielvariable vorhersagt.

9. **hasFeature:** Sie hilft uns die Feature von Dataset mit die Meta Features zu verbinden.

Hier sind die Entities, die ich für die ML-Schema hinzugüget habe. Siehe bild 4.1

Abbildung 4.1: Screenshots vom ML Scheme mit Meta Features

4.2 Modellierung von Datensätzen für das Meta-Lernen in ORKG

In diesem Abschnitt würden wir den Prozess der Modellierung von Datensätzen für das Meta-Learning im (ORKG) diskutieren. Die folgenden Aspekte werden behandelt:

- **Klasse für Datensatz:** Um auf die Informationen zuzugreifen oder sie zu manipulieren (d.h. hinzufügen, bearbeiten, löschen), wird der ORKG-Klasse eine Klassenkomponente hinzugefügt, um die Aktionen zu kapseln. Wir haben eine Klasse 'Dataset for Meta Learning' mit einer Instanz des Datasets 'credit-g' erstellt und dort alle Meta features für die Dataset speichert.
- **Template für Datensatz:** Das Template für die Datensatz gibt die Struktur und die Eigenschaften der in ORKG verwendeten Datensätze vor. Es enthält

Felder und Eigenschaften, um eine einheitliche Metadatenstruktur zu gewährleisten. Wir haben ein Template namens 'Dataset for Meta Learning' erstellt und alle wichtigen Meta-Features, die wir benötigen, hinzugefügt. Durch die einheitliche Strukturierung dieser Informationen wird eine konsistente und leicht zugängliche Darstellung von Forschungsdaten gewährleistet, was wiederum die Suche, Nachverfolgung und Wiederverwendung von Daten erleichtert, siehe Abbildung 4.2.

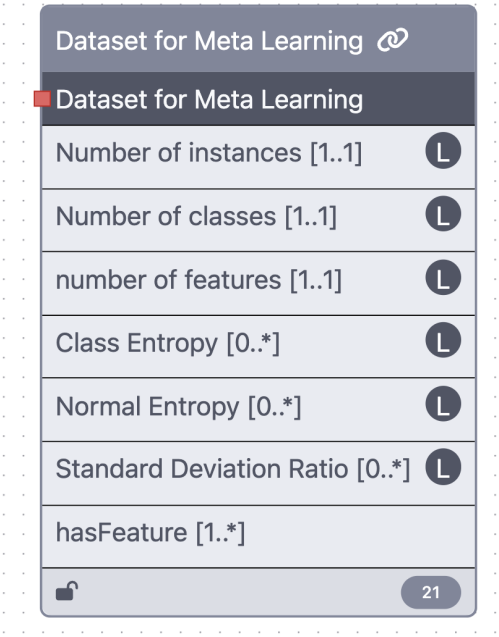


Abbildung 4.2: Screenshots vom Template für Datensatz

- **Klasse für Meta-feature:** Die Klasse für Meta-Features bietet eine strukturierte Möglichkeit, zusätzliche Informationen über Datensätze in ORKG zu erfassen und darzustellen. Diese Klasse ermöglicht die systematische Erfassung von Eigenschaften und Features, die über die grundlegenden Metadaten hinausgehen und eine tiefere Analyse und Interpretation der Daten ermöglichen.

- **Template für Meta-feature:** Das Template für Meta-Features dient zur Erfassung zusätzlicher Informationen über die Datensätze, die über die grundlegenden Metadaten hinausgehen. Ein Template für Meta-Features könnte auch Informationen über die Verteilung von Klassen in einem Datensatz, statisti-

sche Maße wie statistische Zahlen, Mean/Min/Max/Mid usw. enthalten, siehe Abbildung 4.3.

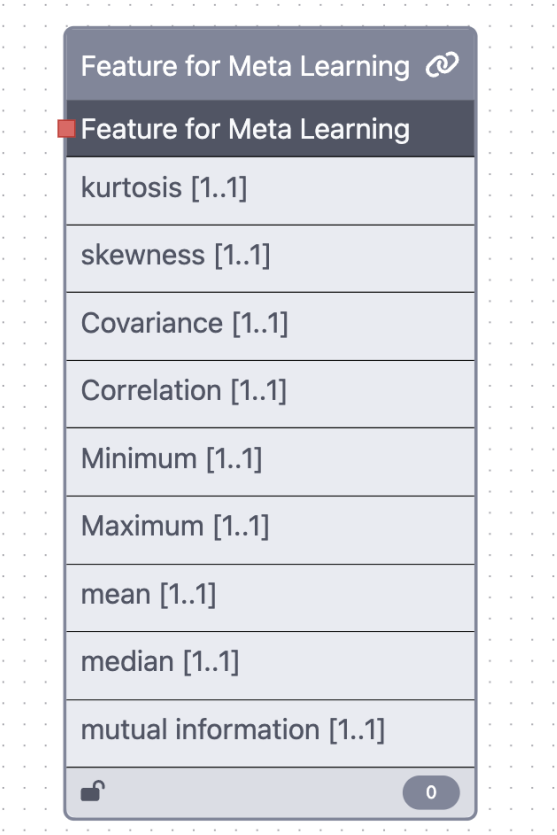


Abbildung 4.3: Screenshots vom Template für Meta Features

Der vorliegende Algorithmus 4.3 skizziert einen systematischen Prozess zur Extraktion von Meta-Features aus Datensätzen und deren Integration in den Online Research Knowledge Graph (ORKG).

Algorithm 4.1 Funktion zum Extrahieren von Meta-Features

```
1: function EXTRAHIERE_META_FEATURES(dataset_name)
2:     Holen Sie den Datensatz
3:     Identifizieren Sie die Zielvariable
4:     Extrahieren Meta-Features wie Anzahl der Instanzen, Features, Klassen usw.
5:     Berechnen Sie zusätzliche Meta-features wie Schiefe, Kurtosis, Min-/Max-
   Werte, Mittelwert, Median, Klassenentropie und normale Entropie
6:     return Meta-Features als Dictionary
7: end function
```

Algorithm 4.2 Funktion zur Verarbeitung von Meta-Features

```
1: function VERARBEITE_META_FEATURES(data)
2:     Iteriere über Meta-Features
3:     Füge jedes features als Ressource in ORKG hinzu
4:     Füge featuresdaten als Eigenschaften mit entsprechenden Prädikaten hinzu
5:     return Liste von Ressourcen-IDs für features
6: end function
```

Algorithm 4.3 Hauptalgorithmus

```
1: Definiere custom_classID, dataset_name_label, dataset_name und andere Varia-
   blen
2: for jeder Datensatz in Datensätzen do
3:     Füge den Datensatz als Ressource in ORKG hinzu
4:     data = extrahiere_meta(Datasatz)
5:     ids = verarbeite_meta_Features(data)
6:     Erstelle ein Dictionary der Einzelwert-Meta-Features
7:     for jedes Einzelwert-Meta-Features do
8:         if Features ist 'hasFeature' then
9:             Verbinde jede Features-Ressourcen-ID mit der Datensatz-Ressource
10:        else
11:            Füge Literal für den Wert hinzu
12:            Verbinde Literal mit der Datensatz-Ressource mit entsprechendem
   Prädikat
13:        end if
14:    end for
15:    Drucke Statusnachricht
16: end for
```

4.3 Modellierung von ML-Experimenten im ORKG

In diesem Abschnitt habe ich mich mit der Modellierung von Machine-Learning-Experimenten im Online Research Knowledge Graph (ORKG) beschäftigt. Dabei habe ich mich insbesondere auf die Erfassung und Strukturierung von wissenschaftlichen Papers konzentriert, die sich mit Meta-Learning beschäftigen und die Datensätze von AutoML verwenden. Dazu habe ich eine spezielle Klasse für solche Papers verwendet und eine Vorlage entwickelt, die es ermöglicht, relevante Informationen wie die verwendeten Datensätze, der Modell und die Hyperparameten mit seine Einstellungen. Außerdem habe ich einen Algorithmus implementiert, der es ermöglicht, diese Informationen automatisch aus den Datensätze zu extrahieren und entsprechende Meta-Features in ORKG Resources zu erstellen. Hier sind die Entities, die ich für die präsentieren die ML-Schema in the ORKG benutzt habe:

1. **Implementation of:** Eine Entität, die die Implementierung eines maschinellen Lern-Algorithmus darstellt.

2. **has Dataset:** Eine Beziehung, die angibt, welcher Datensatz für das Training oder die Evaluierung des Algorithmus verwendet wird.

3. **has Hyperparameter:** Eine Beziehung, die die Hyperparameter angibt, die für die Konfiguration des Algorithmus verwendet werden.

4. **HyperparameterSetting:** Eine Entität zur Darstellung einer bestimmten Einstellung der Hyperparameter für den Algorithmus.

4.4 Meta-Learning mit den ORKG-Ressourcen

Meta-Learning mit den Ressourcen des Open Research Knowledge Graph (ORKG) ermöglicht die Nutzung von Informationen aus verschiedenen Quellen, um Modelle zu trainieren und zu verbessern.

Die Implementierung eines Meta-Learning-Modells mit Hilfe des Online Research Knowledge Graph (ORKG) erfolgt in mehreren Schritten. Um relevante Datensätze zu einem bestimmten Thema oder einer bestimmten Ressource zu erhalten, wird zunächst die ORKG API abgefragt. Diese Abfrage wird durch definierte Ressourcen-IDs gesteuert, um sicherzustellen, dass die zurückgegebenen Datensätze für das Modell relevant sind. Nach Erhalt der Datensätze werden diese strukturiert und in einen Datenrahmen eingefügt. Dieser Datenrahmen enthält alle relevanten Informationen, die für das Meta-Learning-Modell benötigt werden.

Algorithm 4.4 Datenabruf und Verarbeitung

Importiere die Bibliotheken `pandas` und `requests`.
Führe einen GET-Anruf an die ORKG-API durch, um Daten abzurufen.
Extrahiere die erforderlichen Daten aus der API-Antwort.
Initialisiere eine Liste von Ressourcen-IDs.
for Jedes Element `item` in den erhaltenen Daten **do**
    Extrahiere Beitrag- und Ressourcen-ID sowie den Datensatznamen aus `item`.
    if Datensatz-Ressource ist in der Liste der Ressourcen-IDs **then**
        Füge die Informationen zur Datensatzliste hinzu und speichere die Beitrag-ID.
    end if
end for
Erstelle ein DataFrame aus den gesammelten Informationen.
Initialisiere eine leere Liste für DataFrames.
for Jede Ressourcen-ID `resource_id` in der Liste **do**
    Führe einen GET-Anruf an die ORKG-API für `resource_id` durch.
    if GET-Anruf erfolgreich **then**
        Verarbeite die API-Antwort und extrahiere relevante Daten.
        Erstelle ein DataFrame aus den extrahierten Daten und füge es zur Liste der DataFrames hinzu.
    else
        Gib eine Fehlermeldung aus.
    end if
end for
Kombiniere alle DataFrames zu einem einzigen DataFrame.
Wandle den DataFrame in das gewünschte Format um.
Datenrahmen in Matrix umwandeln

Der sich daraus ergebende DataFrame enthält Informationen über die Beziehungen zwischen den Ressourcen und den mit ihnen verbundenen Objekten. Die Spalte „resource name“ enthält die Namen der Ressourcen, zu denen die Daten gehören. Die Spalte „predict name“ enthält die Namen der Prädikate, die die Beziehung zwischen der Ressource und dem Objekt beschreiben. Die Spalten „object id“ und „object label“ enthalten die IDs und Bezeichnungen der Objekte, auf die in den Aussagen Bezug genommen wird. Die Spalte „object datatype“ gibt den Datentyp der Objekte an. Zusätzlich wurde die Spalte „contributionID“ hinzugefügt, die der Ressource ihre Contribution ID zuordnet, um die Identifizierung der Ressourcen zu erleichtern.
Der DataFrame enthält Daten zu verschiedenen Prädikaten, die mit Ressourcen in Beziehung sind. In der Spalte „predict name“ gibt es 8 verschiedene Prädikate, wobei „0.93“ am häufigsten vorkommt. Die Spalte „Class Entropy“ enthält 7 verschiedene Werte, wobei „0.93“ der häufigste ist. Die Spalte „Normal Entropy“ hat 4 verschiedene Werte, wobei „2“ am häufigsten vorkommt. „2“ ist auch der häufigste Wert in der Spalte „Number of classes“, während „4601“ der häufigste Wert in der Spalte „Number of instances“ ist. „5.19“ ist der häufigste Wert in der Spalte „Standard Deviation Ratio“ und „10“ ist der häufigste Wert in der Spalte „Number of features“. Die Spalte „ContributionID“ enthält 9 verschiedene Werte, wobei „R346032“ am häufigsten vorkommt.

predict_name	Class Entropy	Normal Entropy	Number of classes	Number of instances	Standard Deviation Ratio	number of features	ContributionID
count	9	9	9	9	9	9	9
unique	8	7	4	9	9	8	9
top	0.93	0.93	2	4601	5.19	10	R346032
freq	2	2	6	1	1	2	1

Abbildung 4.4: Screenshots von der beschriebene DataFrame

Anschließend durchläuft der Code die erhaltenen Datensätze, um zusätzliche Informationen zu extrahieren. Dabei werden irrelevante Daten herausgefiltert und ein strukturierter Datenframe auf Basis der extrahierten Daten erstellt. Dieser Datenrahmen wird dann zusammengefasst und als Matrix transformiert, um besser mit den Daten umzugehen.

Durch diese Transformation wird sichergestellt, dass die Daten in einem Format vorliegen, das für die anschließende Analyse und das Training des Modells geeignet ist. Durch diese Schritte ermöglicht der Code die Implementierung eines Meta-Learning-Modells unter Verwendung der ORKG-Ressourcen und ermöglicht die Extraktion von Informationen und Daten aus den verfügbaren Datensätzen.

Um die Ähnlichkeit zwischen den Meta-Features und den Contribuations zu quantifizieren, wurde die Cosinus-Distanz verwendet. Die Kosinusdistanz misst den Win-

kel zwischen den Vektoren und liefert einen Wert zwischen 0 und 1, wobei 0 für perfekte Ähnlichkeit steht. Auf diese Weise konnte die Ähnlichkeit zwischen den Meta-Merkmalen jedes Datensatzes und den Ressourcen-IDs quantifiziert werden.

Kapitel 5

Implementierung

Meine Implementierung besteht aus der Extraktion von Metafunktionen und deren Integration in den Open Research Knowledge Graph (ORKG). Meta-Features dienen dazu, wichtige Zusatzinformationen aus den Daten zu extrahieren, die über deren Darstellung hinausgehen. Die Integration dieser Meta-Features in den ORKG erweitert die semantische Tiefe und den Informationsgehalt der Daten, was präzisere Analysen und Interpretationen ermöglicht. Diese Erweiterung des ORKG um Meta-Features trägt dazu bei, das Verständnis und die Nutzung von wissenschaftlicher Literatur und Forschungsdaten zu verbessern, indem zusätzliche Einblicke und Kontextualisierungsmöglichkeiten geboten werden.

5.1 Extraktion von Meta-Features:

Ein wichtiger Schritt bei der Analyse von Daten ist die Extraktion von Meta-Features aus Datensätzen. In diesem Abschnitt diskutieren wir die Python-Implementierung zur Extraktion von Meta Featuresaus Datensätzen.
Die Funktion extract_meta_features(dataset_name) wird definiert, um Meta-Features aus einem gegebenen Datensatznamen zu extrahieren. Es werden verschiedene features wie Anzahl der Instanzen, Anzahl der Features, Standard Deviation Ratio, Klassenentropie usw. berechnet. Zusätzlich werden für jede numerische Feature-Spalte weitere Meta-Features wie Schiefe, Kurtosis, Min- und Max-Werte etc. berechnet. Siehe Bild 5.1

```python
import pandas as pd
import numpy as np
from sklearn import datasets
from scipy.stats import entropy
def extract_meta_features_2(dataset_name):
    # Fetch the dataset from OpenML
    dataset = datasets.fetch_openml(dataset_name, parser='auto', as_frame=True)

    # Identify the target variable
    target = next((t for t in possible_targets if t in dataset.frame.columns), None)
    if target is None:
        raise ValueError("Target variable not found in the dataset.")

    # Extract meta-features
    num_instances = dataset.frame.shape[0]
    num_features = dataset.frame.shape[1]
    num_classes = len(dataset.frame[target].unique())

    meta_features = {
        'Number of instances': num_instances,
        'Number of features': num_features,
        'Number of classes': num_classes,
        'Standard Deviation Ratio': None,
        'Class Entropy': None,
        'Normal Entropy': None,
        'hasFeatures': {}
    }

    for feature in dataset.frame.select_dtypes(include=np.number).columns:
        meta_features['hasFeatures'][feature] = {
            'Skewness': round(dataset.frame[feature].skew(), 2),
            'Kurtosis': round(dataset.frame[feature].kurt(), 2),
            'Min values': round(dataset.frame[feature].min(), 2),
            'Max values': round(dataset.frame[feature].max(), 2),
            'Mean values': round(dataset.frame[feature].mean(), 2),
            'Median values': round(dataset.frame[feature].median(), 2)
        }

    # Calculate other meta-features
    sd_ratio = round((dataset.frame.std(numeric_only=True) /
dataset.frame.mean(numeric_only=True)).mean(), 2)
    class_entropy = round(entropy(dataset.frame[target].value_counts(normalize=True), base=2), 2)
    normal_entropy = round(class_entropy / np.log2(len(dataset.frame[target].unique())), 2)
    meta_features['Standard Deviation Ratio'] = sd_ratio
    meta_features['Class Entropy'] = class_entropy.item()
    meta_features['Normal Entropy'] = normal_entropy

    return meta_features
```

Abbildung 5.1: Python Code zur extrahieren die Meta features

5.2 Integration in die ORKG:

Die Funktion `process_meta_data(data)` nimmt als Eingabe ein Datenobjekt und verarbeitet die darin enthaltenen Meta-Features. Die Funktion geht durch die Features in data['hasFeatures'], erzeugt für jedes Feature eine Ressource und fügt diese als Eigenschaft hinzu. Als erstes wird für jedes feature eine Ressource erzeugt und die ID dieser Ressource wird zu einer Liste von Ressourcen-IDs hinzugefügt. Danach werden die Feature-Daten als Eigenschaften der erstellten Ressource hinzugefügt. Dazu wird für jeden Metawert eines Feature ein Literal erstellt und ein Prädikat hinzugefügt, das die Beziehung zwischen der Ressource und dem Literal beschreibt. Diese Beziehungen werden als Statements in der ORKG eingetragen. Am Ende gibt die Methode eine Liste von Ressourcen-IDs zurück, die die verarbeiteten Meta-Features repräsentieren. siehe Bild 5.2 Nach der Extraktion von Meta-Features aus einer

```python
def process_meta_features(data):
    features_resource_ids = []

    # Iterate through the features in data['Features']
    for feature_name, feature_data in data['hasFeatures'].items():
        # Add feature as resource
        feature_resource = orkg.resources.add(label=feature_name).content
        feature_resource_id = feature_resource['id']
        features_resource_ids.append(feature_resource_id)

        # Add feature data as properties
        for meta_feature, value in feature_data.items():
            # add Literal
            literal_id =
orkg.literals.add(label=str(value)).content['id']

            # add the predicate
            predicate_id = feature_predicates_2[meta_feature]

            # connect them
            statement_data = {
                'subject_id': feature_resource_id,
                'predicate_id': predicate_id,
                'object_id': literal_id
            }
            orkg.statements.add(**statement_data)

    return features_resource_ids
```

Abbildung 5.2: Python-Code zur Verarbeitung der Features

großen Anzahl von Datensätzen ist der nächste Schritt die Integration dieser Meta-Features in den Open Research Knowledge Graph (ORKG). Die extrahierten Meta-Features sind in einem Dictionary gespeichert, das Schlüssel-Wert-Paare für einzelne Meta-Features enthält. Einige dieser Meta-Features haben einzelne Werte, während das Meta-Feature 'hasFeature' eine Liste von Ressourcen-IDs enthält, die die extrahierten Features repräsentieren. Dann werden die Daten mit den Meta-Features für jeden datensatz in die ORKG eingefügt. Für jedes Feature in der Dictoinary wird geprüft, ob es ein 'hasFeature'-Feature ist. Wenn ja, werden die Ressourcen-IDs der Features mit dem datensatz verknüpft. Andernfalls wird ein Literal erzeugt und mit dem entsprechenden Prädikat verknüpft, um die Beziehung zwischen dem datensatz und dem Feature darzustellen. Die Methode wartet wieder eine Sekunde, bevor sie mit dem nächsten Datensatz fortfährt, um einen technischen Absturz von ORKG aufgrund der hohen Anzahl von Anfragen zu vermeiden, Siehe Bilde 5.3

5.3 Implementierung der KNN Model:

Wir haben am Ende ein KNN-Modell entwickelt, das die nächsten Nachbarn des Datasets berechnet, basierend auf den Meta-Features. Die Implementierung des K-Nearest Neighbors (KNN)-Algorithmus verwendet die scikit-learn-Bibliothek in Python, um Aufgaben des überwachten Lernens, insbesondere der Klassifizierung, durchzuführen. Bei dieser Implementierung wird der Datensatz zunächst in ein Numpy-Array umgewandelt, wobei die Features von den zugehörigen Labels getrennt werden. Fehlende Werte innerhalb des Datensatzes werden durch Imputation mit Hilfe der Mittelwertstrategie behandelt, um die Robustheit in der Trainings- und Testphase zu gewährleisten. Zusätzlich wird eine Datennormalisierung durchgeführt, um die Merkmale auf verschiedenen Skalen zu standardisieren und die Leistung des Modells zu verbessern. Der KNN-Klassifikator wird dann mit einer bestimmten Anzahl von Nachbarn instanziiert, die üblicherweise mit „k“ bezeichnet wird. Anschließend wird das Modell mit den vorverarbeiteten Trainingsdaten trainiert. Während der Testphase sagt der KNN-Algorithmus für jeden Testdatensatz unter Berücksichtigung der nächsten Nachbarn im Originaldatensatz den Beitrag vorher. Abschließend wird die Genauigkeit der Vorhersagen mit Hilfe der Metrik accuracy_score bewertet, die Aufschluss über die Effizienz des Modells bei der genauen Klassifizierung von Beiträgen gibt.

```python
for dataset in automl_datasets:
    # Add dataset as a resource
    resource = orkg.resources.add(label=f'{dataset} test', classes=
[custom_classID]).content
    resource_id = resource['id']

    # Extract meta features for the current dataset
    data = extract_meta_features(dataset)
    time.sleep(1)
    ids = process_meta_features(data)

    # Construct dictionary of single value meta-features
    single_value_metaFeatures = {
        'Number of instances': data['Number of instances'],
        'Number of features': data['Number of features'],
        'Number of classes': data['Number of classes'],
        'Standard Deviation Ratio': data['Standard Deviation Ratio'],
        'Class Entropy': data['Class Entropy'],
        'Normal Entropy': data['Normal Entropy'],
        'hasFeature': ids
    }

    print(f"Adding {dataset}")
    time.sleep(1)
    # Iterate through single_value_metaFeatures
    for feature, value in single_value_metaFeatures.items():
        # Check if the feature is 'hasFeature'
        if feature == 'hasFeature':
            # Connect each feature resource ID to the dataset resource
            for feature_resource_id in value:
                statement_data = {
                    'subject_id': resource_id,
                    'predicate_id': hasFeatures_predicate_id,
                    'object_id': feature_resource_id
                }
                orkg.statements.add(**statement_data)
        else:
            # Add Literal
            literal_id = orkg.literals.add(label=str(value)).content['id']
            # Add the predicate
            predicate_id = feature_predicates[feature]

            # Connect them
            statement_data = {
                'subject_id': resource_id,
                'predicate_id': predicate_id,
                'object_id': literal_id
            }
            orkg.statements.add(**statement_data)

    print(f"Done {dataset}")
```

Abbildung 5.3: Python Code zur Hinzufügen die Meta features in ORKG

Kapitel 6

Experimentelle Evaluierung

In diesem Abschnitt befassen wir uns mit der Evaluierung unseres Modells, dem für die Tests verwendeten Datensatz und den dafür aufgewendeten Ressourcen. Unsere Bewertung konzentriert sich hauptsächlich auf die Anwendung des K-Nearst-Neigbhor-Algorithmus (KNN), einer weit verbreiteten Methode des maschinellen Lernens, um die Effizienz unseres Modells in einem realen Szenario zu bewerten. Durch die Analyse des Datensatzes, der verwendeten Ressourcen und der aus dem KNN-Ansatz erhaltenen Ergebnisse wollen wir eine umfassende Analyse der Fähigkeiten des Modells und seiner praktischen Anwendbarkeit liefern. Diese Evaluierung ist ein entscheidender Schritt, um die Effektivität und das Potenzial der von uns vorgeschlagenen Lösung zu verstehen.

6.1 Daten:

Der AutoML-Benchmark ist eine wichtige Komponente für die Bewertung und den Vergleich verschiedener AutoML-Systeme und -Techniken. Der Benchmark enthält verschiedene Datensätze aus unterschiedlichen Bereichen, die jeweils ihre eigenen Herausforderungen und Eigenschaften haben. Der AutoML Benchmark All Classification umfasst 71 Datensätze, die sorgfältig ausgewählt wurden, um reale Szenarien zu repräsentieren und die Stabilität und Zuverlässigkeit des Benchmarking-Prozesses zu gewährleisten. Am Ende habe ich die 10 Testdatensätze des AutoML Benchmark mit ihren Originalwerten verwendet. Hier ist eine Liste mit den Datensätze, die Ich benutzt habe:

1. **credit-g:** [25]
2. **Spambase:** [9]

3. **ilpd** [21]
4. **har** [20]
5. **balance-scale** [24]
6. **CIFAR-10** [11]
7. **pc1** [23]
8. **breast-w** [24]
9. **mnist_784** [13]
10. **tic-tac-toe** [1]

6.2 Experimenteller Aufbau

Der KNN-Algorithmus wird evaluiert. Für die Evaluierung wurden 10 Test Datensatz von die AutoML benchmark als Input für die Berechnung der k-nearest neighbors verwendet. Dazu wird seine Leistung auf einem Testdatensatz analysiert. Ich habe 10 Testdatensätze erstellt. Bei 2 Merkmalen habe ich die Werte um 10 Prozent erhöht. Hier ist ein Screenshot des Dataframe dieser Experimente. Siehe Bilde 6.1.

predict_name resource_name	Class Entropy	Normal Entropy	Number of classes	Number of instances	Standard Deviation Ratio	number of features
CIFAR-10 Test	1.0	3.32	10	70000	0.52	307
balance-scale test	1.32	0.83	3	687	0.47	5
breast-w test	0.93	0.93	2	768	0.9	11
credit-g Test	0.881291	0.881291	2	1000	0.464014	21
har test	2.58	1.0	6	11328	-0.21	618
ilpd test	0.86	0.86	2	641	1.18	12
mnist_784 test	3.32	1.0	11	77000	28.31	863
pc1 test	0.36	0.36	2	1219	2.11	24
spambase test	0.97	0.97	2	5061	5.19	63
tic-tac-toe test	0.93	0.93	2	958	nan	10

Abbildung 6.1: Datenframe von den Test Datensätze

Zunächst wurden die Meta-Features und die Ressourcen-IDs extrahiert. Die Meta-Features wurden als Matrix Xmeta dargestellt, wobei jede Zeile die Features eines Datensatzes repräsentiert. Die Ressourcen-IDs wurden separat als Liste von Beitrags-IDs dargestellt.

Nachdem das Modell mit den Trainingsdaten trainiert wurde, werden die Vorhersagen für die Testdaten erstellt und mit den tatsächlichen Beitrags-IDs verglichen. Die Genauigkeit der Vorhersagen wird mit Hilfe der Metrik accuracy_score bewertet, die das Verhältnis der korrekt klassifizierten Beitrags-IDs zur Gesamtzahl der Vorhersagen angibt. Dann habe ich die Testdatensätze mit den Originaldatensätzen in einem Scatterpolt gezeichnet und hier ist die Ausgabe. siehe bild 6.2

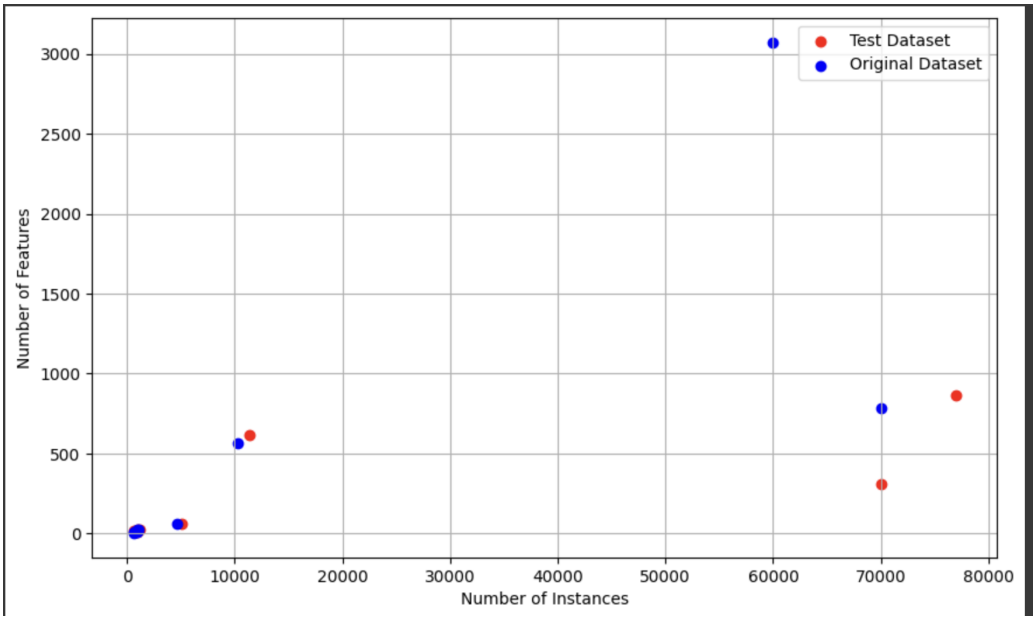


Abbildung 6.2: Scennshot des Scatterplots für die Test- und Originaldatensätze

6.3 Ergebnisse

Ich habe für die Erstellung von ORKG-Datensätzen eine Skript ausgeführt mit IDs der erstellten Ressourcen. Hier ist eine Beispiel für die Datensatz. Der Prozess beinhaltete die Abfrage der ORKG API, um Informationen zu sammeln und die Datensätze entsprechend zu füllen. siehe Bild 6.3.

Nachdem die Datensätze in ORKG erstellt waren, verknüpften wir jeden Datensatz mit den wissenschaftlichen Arbeiten, die ihn verwenden. Für jeden Datensatz haben wir eine gründliche Recherche durchgeführt, um relevante Arbeiten zu finden, die den Datensatz in ihrer Forschung verwenden. Anschließend haben wir entsprechende IDs in ORKG erstellt und jeden Datensatz mit der entsprechenden Arbeit verknüpft. Hier ist ein Beispiel für einen erstellten Contribuation. siehe bild 6.4

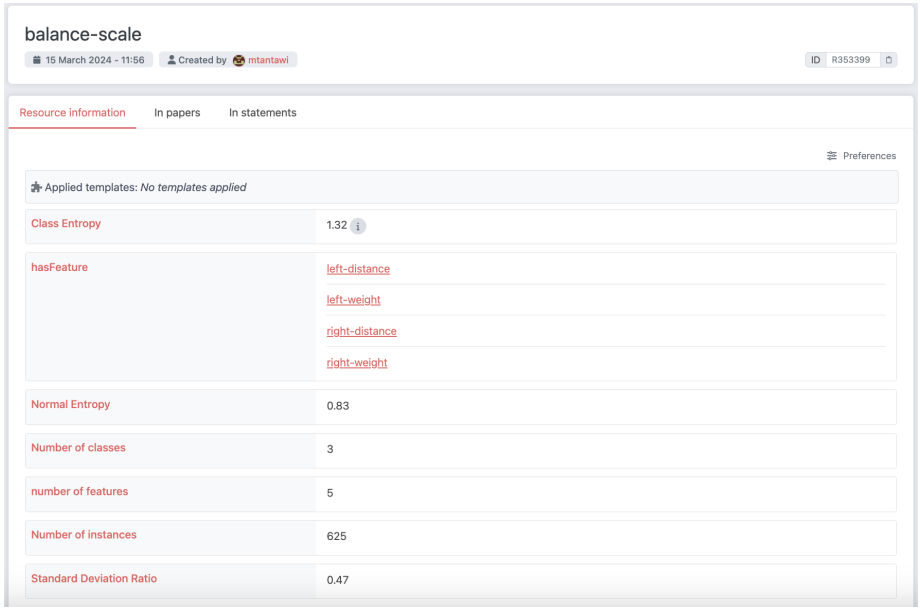


Abbildung 6.3: Screenshots von erstellten Ressourcen im ORKG

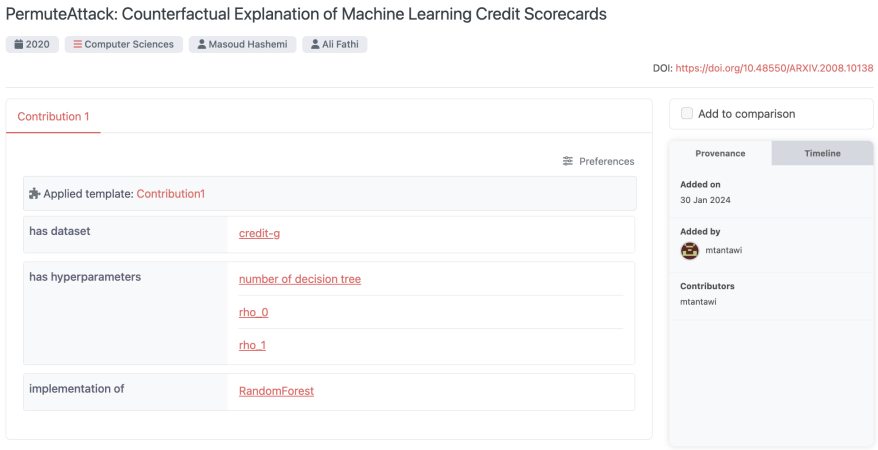


Abbildung 6.4: Screenshots von erstellten Contribuation im ORKG

Anschließend wurde die Cosinus-Distanz zwischen jedem Meta-Merkmal und der entsprechenden Ressourcen-IDs berechnet. Die berechneten Kosinusdistanzen für die ausgewählten Datensätze lauten wie folgt: Diese Ergebnisse zeigen die Cosinus-

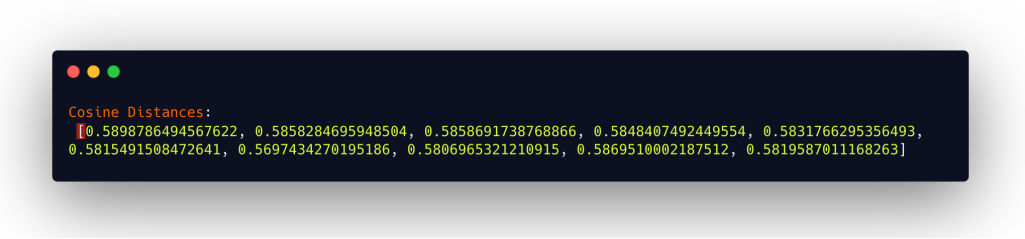


Abbildung 6.5: Ausgabe von die Kosinus-Distanz

Distanz zwischen den Meta-Merkmalen und den Beitrags-IDs. Ein geringerer Abstand deutet auf eine größere Ähnlichkeit hin, während ein größerer Abstand auf eine geringere Ähnlichkeit hindeutet. Dennoch können die generierten Daten verwendet werden, um Ähnlichkeitssuchen durchzuführen, um ähnliche Beiträge oder Datensätze zu identifizieren.

Die Ausgabe zeigt die Resultate der Vorhersagen, die mit dem KNN-Algorithmus für verschiedene Testdatensätze gemacht wurden. Jede Zeile repräsentiert einen Testdatensatz, dessen ID angegeben ist, gefolgt von der ursprünglichen Contribuations-ID von die Orginial Datensatz und der vorhergesagten Contributions-ID. Die ursprünglichen Contributions stammen aus dem Trainingsdatensatz, während die vorhergesagten Contributions diejenigen sind, die vom Algorithmus für die entsprechenden Testdatensätze vorhergesagt wurden. Hier ist ein screenshot von der Ergbnisse von dem Model. Siehe Bild 6.6. Eine Beispielzeile lautet: „Test Dataset ID: R355390,

```
Test Dataset ID: R355390, Original Contribution: R346055, Predicted ContributionID: R346055
Test Dataset ID: R355408, Original Contribution: R345002, Predicted ContributionID: R345002
Test Dataset ID: R355430, Original Contribution: R344079, Predicted ContributionID: R344079
Test Dataset ID: R355992, Original Contribution: R345026, Predicted ContributionID: R345026
Test Dataset ID: R356002, Original Contribution: R346060, Predicted ContributionID: R346060
Test Dataset ID: R355385, Original Contribution: R345010, Predicted ContributionID: R345010
Test Dataset ID: R356061, Original Contribution: R345021, Predicted ContributionID: R345021
Test Dataset ID: R350002, Original Contribution: R346045, Predicted ContributionID: R346045
Test Dataset ID: R350040, Original Contribution: R346032, Predicted ContributionID: R346032
Test Dataset ID: R354165, Original Contribution: R346038, Predicted ContributionID: R346038
```

Abbildung 6.6: Ausgabe von die KNN Model

Original Contribution: R346055, Predicted Contribution ID: R346055“. Hier bedeutet dies, dass für den Testdatensatz mit der ID R355390 der Contributions-ID Beitrag

R346055 war und der Algorithmus auch R346055 vorhergesagt hat. Dies deutet darauf hin, dass der Algorithmus in diesem Fall die richtige Vorhersage getroffen hat. Die Gesamtgenauigkeit des Algorithmus, berechnet mit der Genauigkeits-Metrik accuracy_score, beträgt 1.0, was bedeutet, dass 100 Prozent der Vorhersagen mit den tatsächlichen Beiträgen übereinstimmen. Dies deutet darauf hin, dass der KNN-Algorithmus in diesem Szenario möglicherweise optimal funktioniert. Den Quellcode dieser Arbeit finden Sie im GitHub-Repository[1].

[1]GitHub repository: https://github.com/mostafatantawi/meta-features

Kapitel 7

Zusammenhang

Ziel dieser Bachelorarbeit war es, einen Beitrag zur Entwicklung eines wissensbasierten Meta-Lernmodells zu leisten, das auf einer skalierbaren Wissensrepräsentation und -speicherung basiert. Durch die umfassende Analyse und Implementierung verschiedener Methoden, Techniken und Verwandte Arbeit konnten wichtige Erkenntnisse gewonnen werden, die einen bedeutenden Fortschritt auf diesem Gebiet darstellen.

Durch die Untersuchung relevanter Arbeiten in diesem Bereich werden Erkenntnisse gewonnen, die als Grundlage für die Weiterentwicklung eines wissensbasierten Meta-Lernmodells dienen. Durch die Extraktion von Meta features und deren Integration in den ORKG wird die semantische Tiefe und der Informationsgehalt der Daten erhöht, was genauere Analysen und Interpretationen ermöglicht. Wir haben am Ende ein KNN-Modell entwickelt, das die nächsten Nachbarn des Datasets berechnet, basierend auf den Meta-Features. Am Ende haben wir auch Testdatensätze mit verschiedenen Werten erstellt, um die Genauigkeit zu berechnen.

In Bezug auf das KNN-Modell konnten wir ebenfalls wichtige Erkenntnisse gewinnen. Die Evaluierung des KNN-Algorithmus ergab eine Genauigkeit von 10 Prozent, was darauf hinweist, dass der verwendete Ansatz in diesem Kontext möglicherweise nicht die gewünschten Ergebnisse liefert. Diese Ergebnisse zeigen die Bedeutung der weiteren Untersuchung und Optimierung von Modellen und Algorithmen für spezifische Anwendungsfälle.

Insgesamt hat diese Arbeit gezeigt, dass die Entwicklung eines wissensbasierten Meta-Lernmodells in Kombination mit der Evaluierung verschiedener Algorithmen wie KNN ein vielversprechender Ansatz ist, um komplexe Lernaufgaben zu lösen und die Leistung zu verbessern. Weitere Forschung und Entwicklung in diesem Bereich wird unser Verständnis von Meta-Learning vertiefen und neue Anwendungen und Lösungen für verschiedene Probleme und Anwendungsfälle entwickeln.

Literatur

[1] David Aha. *Tic-Tac-Toe Endgame*. UCI Machine Learning Repository. 1991.

[2] J. Bogatinovski u. a. „Explaining the Performance of Multi-label Classification Methods with Data Set Properties“. In: *arXiv preprint* (2021). eprint: 2106.15411. URL: https://doi.org/10.48550/arxiv.2106.15411.

[3] André C P L F de Carvalho u. a. „Meta-learning: A survey“. In: *arXiv preprint arXiv:1808.10406* (2018).

[4] Ciro Castiello, Giovanna Castellano und Anna Maria Fanelli. „Meta-data: Characterization of Input Features for Meta-learning“. In: *Modeling Decisions for Artificial Intelligence*. Hrsg. von Vicenç Torra, Yasuo Narukawa und Sadaaki Miyamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 457–468. ISBN: 978-3-540-31883-5.

[5] Artificial intelligence faces reproducibility crisis. „Hutson, M.“ In: *Science 359 (6377), 725 – 726.* (2018).

[6] S. Deng u. a. „Learning Tensor Representations for Meta-learning“. In: *arXiv preprint* (2022). eprint: 2201.07348. URL: https://doi.org/10.48550/arxiv.2201.07348.

[7] Moncef Garouani u. a. „Automated Machine Learning for Big Industrial Data: A Meta-Learning Approach“. In: *Journal of Big Data* 9.1 (2022), S. 57.

[8] Moncef Garouani u. a. „Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data“. In: *Journal of Big Data* 9.1 (Apr. 2022), S. 57. DOI: 10.1186/s40537-022-00612-4. URL: https://doi.org/10.1186/s40537-022-00612-4.

[9] Hopkins Mark Reeber Erik Forman George und Suermondt Jaap. *Spambase*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C53G6X. 1999.

[10] M. Kotlar u. a. „Novel Meta-features for Automated Machine Learning Model Selection in Anomaly Detection“. In: *IEEE Access* 9 (2021), S. 89675–89687. DOI: 10.1109/access.2021.3090936.

[11] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Techn. Ber. University of Toronto, 2009.

[12] Timothy Lebo u. a. „Prov-o: The prov ontology“. In: *W3C Recommendation* 30 (2013).

[13] Yann LeCun, Corinna Cortes und Christopher J.C. Burges. „The MNIST database of handwritten digits“. In: *http://yann.lecun.com/exdb/mnist/* (1998).

[14] Jorge Madrid, Hugo Jair Escalante und Eduardo Morales. „Meta-learning of textual representations“. In: *arXiv preprint arXiv:1906.08934* (2019). http://arxiv.org/abs/1906.08934.

[15] P. Nguyen, M. Hilario und A. Kalousis. „Using meta-mining to support data mining workflow planning and optimization“. In: *Journal of Artificial Intelligence Research* 51 (2014), S. 605–644. DOI: 10.1613/jair.4377.

[16] J. Park u. a. „DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. DOI: 10.1109/cvpr.2019.00025.

[17] Authors not provided. „Meta Learning of Text Representations“. In: *arXiv preprint arXiv:1906.08934* (2019).

[18] Gustavo Correa Publio u. a. „ML-Schema: An interchangeable format for description of machine learning experiments“. In: *Semantic Web* 0.0 (2020), S. 1–11.

[19] W. Raynaut, C. Soulé-Dupuy und N. Vallès-Parlangeau. „Meta-mining evaluation framework: a large scale proof of concept on meta-learning“. In: (2016), S. 215–228. DOI: 10.1007/978-3-319-50127-7_18.

[20] Luca Reyes-Ortiz Jorge Anguita Davide Ghio Alessandro Oneto und Parra Xavier. *Human Activity Recognition Using Smartphones*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K. 2012.

[21] Ooms Richard. Techn. Ber. Utrecht University, 2019.

[22] Adriano Rivolli u. a. „Characterizing classification datasets: a study of meta-features for meta-learning“. In: *Information Sciences* 501 (2019), S. 69–89.

[23] J. Sayyad Shirabad und T.J. Menzies. *The PROMISE Repository of Software Engineering Databases*. Techn. Ber. Ottawa, Canada: School of Information Technology und Engineering, University of Ottawa, 2005.

[24] R. Siegler. *Balance Scale*. UCI Machine Learning Repository. 1994.

[25] *South German Credit*. UCI Machine Learning Repository. 2020.

[26] Joaquin Vanschoren und Larisa Soldatova. „Exposé: An ontology for data mining experiments“. In: *International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010)*. 2010, S. 31–46.

[27] Joaquin Vanschoren u. a. „Meta-Learning: A Survey“. In: *arXiv preprint arXiv:2104.07663* (2021).

[28] J. Wu, W. Xiong und W. Wang. „Learning to Learn and Predict: A Meta-learning Approach for Multi-label Classification“. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019. DOI: 10.18653/v1/d19-1444.

[29] M. Yan u. a. „Multi-Source Meta Transfer for Low Resource Multiple-Choice Question Answering“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020. DOI: 10.18653/v1/2020.acl-main.654.

[30] Y. Zhao u. a. „Reinforcement Learning Method for Machining Deformation Control Based on Meta-Invariant Feature Space“. In: *Visual Computing for Industry Biomedicine and Art* 5.1 (2022). DOI: 10.1186/s42492-022-00123-2.