Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Prädiktive Modelle in YesWorkflow

Constanze Sudra

Studiengang:	Informatik
Prüfer/in:	PD Dr. rer. nat. habil. Holger Schwarz
Betreuer/in:	Dr. rer. nat. Peter Reimann
Beginn am:	27. April 2018
Beendet am:	23. November 2018

Kurzfassung

In der modernen Forschung sind Wissenschaftler zunehmend mit der komplexen Verarbeitung großer Datenmengen konfrontiert. Hierzu gibt es viele Systeme, die für die Definition und Ausführung solcher Workflows verwendet werden können. Sie bieten zahlreiche Vorteile, wie beispielsweise eine hohe Reproduzierbarkeit und eine Vereinfachung der Administration sowie der technischen Umsetzung. Da diese Systeme aber als komplex und die Migration der bestehenden Prozesse als zeitaufwendig angesehen werden, werden oftmals weiterhin vertraute Skriptsprachen für komplexe Datenverarbeitung und keine Workflowsysteme genutzt. Durch die Verwendung von YesWorkflow kann diese Lücke geschlossen werden und der Wissenschaftler weiterhin die vertrauten Skriptsprachen nutzen und dennoch von einigen der Vorteile von Workflowsystemen profitieren. Während von den Workflowsystemen bereits verschiedene Optimierungen (wie z.B. hinsichtlich der Laufzeit) unterstützt werden, wurde YesWorkflow diesbezüglich noch nicht erweitert. In dieser Arbeit wird ein Ablaufkonzept für eine YesWorkflow-Erweiterung erarbeitet, die Provenance-Informationen nutzt, um Optimierungen der verwendeten Skripte, basierend auf prädiktiven Modellen, zu unterstützen. Zum einen wird dazu untersucht, wie die bereits verfügbaren Informationen genutzt werden können. Zum anderen wird erörtert, ob durch zusätzliche Provenance-Informationen eine weitere Verbesserung erzielt werden kann. Der bestehende YesWorkflow-Prototyp wird im Rahmen dieser Arbeit dahingehend erweitert, dass zusätzliche Informationen erfasst und für Vorhersagen genutzt werden können. Dazu wird ein neuer Befehl eingefügt, der es erlaubt anhand von erfassten Provenance-Informationen beliebige Werte oder Metriken mittels verschiedener Verfahren vorherzusagen und dann für Optimierungen zu nutzen. Im Evaluationsteil der Arbeit wird anhand von zwei Skripten bestimmt wie groß die notwendige erfasste Trainingsdatenmenge für eine gewinnbringende Vorhersage sein muss und ob durch verschiedene Vorgehen bei der Vorhersage von Werten eine Verbesserung erzielt werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CSV comma-separated values. 49

NMAE negative median absolute error. 28

WfMS Workflow Management System. 15

YW YesWorkflow. 13

1 Einleitung

Bei YesWorkflow (YW) handelt es sich um ein System, mit dem beliebige Skripte durch Kommentare so erweitert werden können, dass einige der für Workflowsysteme üblichen Vorteile daraus resultieren [LKM]. In dieser Arbeit wird betrachtet, wie prädiktive Modelle in YW zur Optimierung von verwendeten Skripten eingesetzt werden können. Dazu wird betrachtet, welche Provenance-Informationen von YW bereits erfasst werden und welche zusätzlichen Nutzen erwarten lassen. In Form eines Prototyps wird eine beispielhafte Umsetzung für eine ausgewählte Optimierung erstellt und der Gewinn gegen die Menge an dazu notwendigen erfassten Informationen abgewägt.

In diesem Kapitel wird zunächst die Motivation der Arbeit weiter ausgeführt und dann die Zielsetzung erläutert.

1.1 Motivation

In der modernen Forschung sind Wissenschaftler zunehmend mit der komplexen Verarbeitung großer Datenmengen konfrontiert [DVJ+15]. Hierzu gibt es verschiedene Systeme, die für die Definition und Ausführung solcher Workflows verwendet werden können [LWMB09]. Aus ihrer Verwendung resultieren zahlreiche Vorteile für den Wissenschaftler, wie beispielsweise eine hohe Reproduzierbarkeit und eine Vereinfachung der Administration sowie der technischen Umsetzung [GSK+11; LWMB09]. Da diese Systeme aber als komplex und die Migration der bestehenden Prozesse als zeitaufwendig angesehen werden, werden oftmals weiterhin vertraute Skriptsprachen für die Umsetzung und keine Workflowsysteme genutzt [MSK+15].

Durch die Verwendung von YW kann diese Lücke geschlossen werden und der Wissenschaftler weiterhin die vertrauten Skriptsprachen nutzen und dennoch von einigen der Vorteile von Workflowsystemen profitieren [MBBL15b; MSK+15]. Dazu müssen lediglich die bestehenden Skripte durch Annotationen innerhalb von Kommentaren der jeweiligen Programmiersprache erweitert werden. Mithilfe des bestehenden YW-Prototypen kann dann auf eine graphische Repräsentation des Datenflusses sowie auf Provenance-Informationen zu erstellten und gelesenen Daten zugegriffen werden [MBBL15b].

Im Rahmen der Workflowsysteme werden bereits einige Ansätze zur Optimierung der Ausführung eines Workflows angeboten. Dies ist hinsichtlich der oftmals großen Datenmengen und aufwendigen Berechnungen für den Nutzer von großem Interesse. Beispielweise wird von dem Workflowsystem Kepler[1] eine verringerte Laufzeit durch sogenannte „Smart“-Reruns unterstützt [ABJ06]. Von YW werden bisher keine derartigen Optimierungen unterstützt.

[1] https://kepler-project.org/

Mit prädiktiven Modellen, die mit Provenance-Informationen trainiert werden können, kann man Vorhersagen erstellen wie sich ein Workflow in einer zukünftigen Ausführung verhalten wird. Diese Vorhersagen lassen sich dann für Optimierungen, des Workflows oder der gewählten Parameter, nutzen. Somit kann beispielsweise die Laufzeit verringert oder Ressourcen eingespart werden.

1.2 Zielsetzung und Gliederung der Arbeit

Die Zielsetzung dieser Arbeit ist eine Erweiterung des YW-Prototyps, welcher zusammen mit den weiteren für das Verständnis notwendigen Grundlagen in Kapitel 2 vorgestellt wird. Durch diese Erweiterung sollen Provenance-Informationen dazu genutzt werden, um Optimierungen der verwendeten Skripte, basierend auf prädiktiven Modellen, zu unterstützen. In Kapitel 3 werden zunächst ausgewählte Lösungen für ähnliche Problemstellungen vorgestellt und diskutiert. Im folgenden Kapitel 4 wird untersucht, wie prädiktive Modelle allgemein genutzt werden können, um Optimierungen von Workflows, mithilfe der erstellten Vorhersagen, zu unterstützen. Die Vorhersagen sollen zum einen auf den bereits durch YW bereitstehenden Provenance-Informationen basieren, was zunächst eine Analyse der verfügbaren Provenance-Informationen erfordert. Zum anderen wird erörtert ob durch zusätzliche Provenance-Informationen eine weitere Verbesserung erzielt werden kann. Dazu werden zwei unterschiedliche Vorgehensweisen für eine Vorhersage betrachtet, die sich auf ähnliche Problemstellungen übertragen lassen. Um die Modelle in YW einzubinden wird eine Erweiterung der YW-Kommentare und YW-Befehle in Kapitel 4 erläutert. Diese wurde prototypisch umgesetzt, was in Kapitel 5 beschrieben ist. Im 6. Kapitel dieser Arbeit werden die Ergebnisse der Evaluation präsentiert. Es wird anhand von zwei Skripten betrachtet wie viele Trainingsdaten für eine Vorhersage notwendig sind. In Kapitel 7 wird die Arbeit zusammenfassend betrachtet und Ansatzstellen für weitere Arbeiten vorgeschlagen.

2 Grundlagen

In diesem Kapitel werden die Grundlagen, welche für das Verständnis der Arbeit notwendig sind, dargestellt. Zunächst wird beschrieben, was ein Workflow und was Provenance-Informationen sind. In den folgenden Unterkapiteln werden dann der bestehende YW-Prototyp sowie die verfügbaren Provenance-Informationen erläutert. Im letzten Teil dieses Kapitels werden die relevanten Grundlagen zu Prädiktionen dargestellt.

2.1 Workflows

In diesem Abschnitt wird zunächst allgemein der Begriff des Workflows eingeführt. Dann werden Business sowie Scientific Workflows vorgestellt. Im letzten Abschnitt wird auf Scientific Workflow Management Systeme und ihre Vorteile eingegangen.

2.1.1 Allgemeine Definition

Im allgemeinen Sprachgebrauch wird der Begriff Workflow mit zwei Bedeutungen assoziiert.

Erstens mit der *„Abwicklung arbeitsteiliger Vorgänge bzw. Geschäftsprozesse in Unternehmen und Behörden mit dem Ziel größtmöglicher Effizienz“* und zweitens mit dem *„Arbeitsablauf bei Computerprogrammen“* [18].

Reimann et al. klassifizieren Workflows und unterscheiden dabei auf der ersten Ebene **Orchestration Workflows** und **Data-Intensive Workflows** [RSM11] wie folgt: Bei **Orchestration Workflows** steht das Zusammenspiel von Aktionen, bei **Data-Intensive Workflows** steht die Verarbeitung großer Datenmengen im Vordergrund. Business Workflows, die in Abschnitt 2.1.2 näher beschrieben werden, gehören zu der Klasse der **Orchestration Workflows**. Workflows, die unter dem Begriff Scientific Workflows bekannt sind, und in Abschnitt 2.1.3 beschrieben werden, gehören dagegen mehrheitlich zur Klasse der **Data-Intensive Workflows**.

2.1.2 Business Workflows

Unternehmen sind in der Regel in verschiedene Organisationseinheiten wie Bereiche, Abteilungen oder Projektteams strukturiert. Die für die Wertschöpfung notwendigen Vorgänge betreffen dabei oft mehr als nur eine dieser Organisationseinheiten. Eine Möglichkeit zur Erfassung, Formalisierung und Dokumentation dieser Vorgänge bieten Methoden aus dem Bereich der Geschäftsprozessmodellierung. Die Grundstruktur eines Geschäftsprozesses besteht dabei gemäß Schwickert et al. aus einer Kombination von Einsatzgütern, die in einer durch Regeln festgelegten Art und Weise zu einem definierten Arbeitsergebnis transformiert werden [SF96]. Die Abbildung von Geschäftsprozessen zu

Workflows mittels einer Modellierungssprache ermöglicht dabei die Ausführung dieser Vorgänge durch ein IT-System. Bereits 1996 definierte die Workflow Management Coalition, eine Organisation zur Verbreitung, Standardisierung und Nutzung von Workflows, einen (Business) Workflow als *„the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.“* [96]. Die für die Ausführung eines Workflows auf einem IT-System notwendige Umgebung wird als Workflow Management System (WfMS) bezeichnet.

2.1.3 Scientific Workflows

Während Business Workflows seit langem diskutiert und in Unternehmen produktiv eingesetzt werden, hat das Interesse der Wissenschaft am Thema Workflow (Scientific Workflow) in letzter Zeit ebenfalls zugenommen [TDGS07]. Im folgenden Abschnitt wird die Notwendigkeit zur Betrachtung von Scientific Workflows nach [DVJ+15] dargestellt.

Die Grundthese lautet hierbei, dass in der modernen Forschung die Validierung von Hypothesen oftmals auf die komplexe Verarbeitung großer Datenmengen angewiesen ist. Dies führt dazu, dass Wissenschaftler mit neuen Herausforderungen umgehen müssen. Beispielsweise mit einer verteilten Datenhaltung, inhomogenen Rechennetzwerken und über weite Entfernungen zusammenarbeitende Wissenschaftler. Als hiervon besonders betroffene Forschungsgebiete nennen Deelman et al. unter anderem die Astronomie, die Bioinformatik und die Klimaforschung.

Eine Darstellung der verschiedenen Phasen im Lebenszyklus eines Scientific Workflows, ausgehend von einer zu überprüfenden wissenschaftlichen Hypothese beschreiben Ludäscher et al. Diese sind laut [LWMB09] wie folgt definiert:

- Die erste Phase bezeichnet das **Workflow Design**. Im Fokus stehen dabei die Wiederverwendbarkeit bzw. Verfeinerung vorhandener Workflows und die Art und Weise der Bereitstellung von Daten.
- Die zweite Phase wird als **Workflow Vorbereitung** bezeichnet und ist durch die Auswahl, Anbindung und Parametrisierung von Datenquellen sowie die Planung von Rechenkapazitäten gekennzeichnet.
- Die dritte Phase wird als **Workflow Ausführung** bezeichnet. In dieser Phase werden die Eingabedaten verarbeitet und das Ergebnis erzeugt. Entscheidend ist hierbei die Überwachung des Fortschritts durch die Auswertung geeigneter Parameter.
- Die letzte Phase ist die **Post-Analyse**. Hier findet die Überprüfung und Bewertung des Ergebnisses statt, was gegebenenfalls zu einem Redesign und einer erneuten Ausführung des Workflows führen kann.

Den Nutzen von Scientific Workflows beschreiben Görlach et al. wie folgt [GSK+11]:

1. Scientific Workflows tragen zur Bereitstellung von Wissen über geeignete Dienste bei.
2. Die Verwendung von Workflows unterstützt die Beurteilung durch eine entsprechende Forschungsgemeinschaft.
3. Workflows ermöglichen den Umgang mit großen Datenmengen.

4. Workflows vereinfachen die Ausführung von Arbeitsschritten in verteilten und inhomogenen Umgebungen.
5. Die Automation eines Workflows erlaubt den Wissenschaftlern eine Fokussierung auf die eigentliche wissenschaftliche Problemstellung.
6. Workflows ermöglichen die parallele und automatische Durchführung von Simulationen.

2.1.4 Scientific Workflow Management Systeme und ihre Vorteile

Die Verwendung von Scientific Workflows bedingt die Existenz von Scientific Workflow Management Systemen. Ludäscher et al. diskutieren Konzepte und Funktionalitäten von Scientific Workflow Systemen anhand des bereits in Abschnitt 2.1.3 vorgestellten Lebenszyklus eines Scientific Workflow und verdeutlichen dies exemplarisch an der Umsetzung des Kepler Systems. Der folgenden Abschnitt greift einige der in [LWMB09] vorgestellten Ansätze heraus und erläutert diese:

Die Motivation zur Verwendung von Scientific Workflow Management Systemen sehen Ludäscher et al. vor allem darin Wissenschaftler von der Administration und Fragen der technischen Umsetzung, beispielsweise Fragen der Rechenumgebungen, zu entlasten. Die Wissenschaftler sollen sich thematisch auf ihr Fachgebiet fokussieren. Dies führt zu einem größeren Bedarf an Werkzeugen, mit denen man die verwendeten Methoden vereinfachen, Fehler vermeiden und sich wiederholende Aufgaben automatisieren kann. Mit ihnen können Abläufe, die sonst fehleranfällig oder sich wiederholend sind, aneinander gereiht werden. Dies spart dem Wissenschaftler Zeit, da er sich nicht mit komplexen Softwareproblemen und dem Datenmanagement befassen muss. Durch eine Optimierung der Workflows kann zusätzlich eine effizientere Nutzung der verfügbaren Ressourcen, beispielsweise durch Einsparung von Rechenzyklen, erreicht werden. Dabei stellen Ludäscher et al. folgende Funktionalitäten eines Scientific Workflow Management Systems dar, um mit den Anforderungen eines Scientific Workflows umzugehen: Zur besseren Unterstützung des Wissenschaftlers über den gesamten Life-Cycle eines Scientific Workflow ist eine **integrierte Workflow Umgebung** von großer Bedeutung. Die mehrfache Ausführung von Scientific Workflows mit unterschiedlichen Parametern wird durch eine **Workflow Vorbereitung und Ausführungsunterstützung** vereinfacht, beispielsweise durch die Bereitstellung von Dashboards. Die Verwendung von **datengetriebenen Rechenmodellen** vereinfachen die Arbeit mit verschiedenen Datenquellen und -formaten. Eine zunehmend größer werdende Bedeutung sehen Ludäscher et al. in dem Thema **Data Provenance**, vgl. hierzu auch Abschnitt 2.2. Als darauf aufbauender Aspekt wird die **Verwendung unterschiedlicher Berechnungs- und Provenance Modelle** gesehen. Als letzter Punkt wird die **Modellierung und das Design von Workflows** aufgelistet. Hierbei werden Aspekte wie die Verwendung von unterschiedlichen Berechnungsmodellen in Abhängigkeit der hierarchischen Einordnung eines Scientific Workflows genannt.

Ähnlich sehen Görlach et al. die Hauptanforderungen an ein Scientific Workflow Management System. Der folgende Abschnitt fasst die Betrachtung aus [GSK+11] zusammen. Als wichtigste Anforderung an ein WfMS sehen Görlach et al. die **Usability**. Die Wissenschaftler sind in der Regel keine Computerspezialisten und benötigen einfach zu bedienende Softwarewerkzeuge. Als zweite Anforderung wird die **Flexibilität** eines WfMSs betrachtet. Hier geht es vor allem darum den Workflow aufgrund von geänderten Rahmenbedingungen geeignet zu adaptieren. Als dritter Aspekt wird das **Monitoring** genannt, dass die Fähigkeiten zur Überwachung eines Workflows während der Ausführung beschreibt. Ein weiterer wichtiger Aspekt ist die **Reproduzierbarkeit**. Hierbei geht

es darum einen bereits ausgeführten Workflow in vollem Umfang oder in Teilen erneut auszuführen. Weitere Anforderungen sind die **Robustheit** der Anwendung sowie deren **Skalierbarkeit**. Der Begriff Skalierbarkeit umfasst hierbei auch die Möglichkeit zur dezentralen Ausführung eines Workflows in einer verteilten Umgebung. Aufgrund der Unterschiede zwischen Business Workflows und Scientific Workflows können die bereits in der Wirtschaft etablierten Systeme im Bereich der WfMSe nur partiell übertragen werden. Barga et al. erläutern dies an einem Beispiel [TDGS07, S. 9-16]. Während Business Workflows eine geringe Dynamik aufweisen, unterliegen Scientific Workflows eher häufig Änderungen. Gleichzeitig werden Business Workflows in der Regeln von IT-Experten aufgesetzt, während die Wissenschaftler Experten in ihrem jeweiligen Fachgebiet sind. Diese beiden Aspekte zusammen betrachtet führen zu unterschiedlichen Anforderungen an ein WfMS in der jeweiligen Domäne, insbesondere bezüglich der Aspekte Robustheit und der Usability.

2.2 Provenance

Provenance-Informationen sind alle Informationen, die den Entstehungsprozess eines Produktes beschreiben. In der allgemeinen Definition von Provenance bezieht sich Produkt sowohl auf ein physisch vorliegendes, als auch auf ein digitales Objekt [HDB17]. Als Beispiel für ein physisches Objekt wird in Cheney et al. ein Kunstwerk angeführt. Hierbei umfassen die Provenance-Informationen Meta-Daten zu dem Künstler, der es erstellt hat, den verwendeten Farben und ob das Werk später noch verändert wurde. Selbstverständlich sind dies alles Informationen, die beispielsweise für den Wert des Kunstwerks von großer Bedeutung sind [CFLV12]. Bezogen auf ein digitales Objekt, wie beispielsweise das Ergebnis einer Workflowausführung, kann es von Bedeutung sein zu wissen, wie lange die Ausführung gedauert hat und welche Daten erstellt und gelesen wurden. Hierdurch lässt sich beispielsweise ein Ergebnis auf mögliche Fehler überprüfen und die Reproduzierbarkeit sicherstellen [MSK+15].

Da für diese Arbeit ausschließlich Provenance-Informationen bezüglich der Ausführung von Skripten relevant sind, wird im Folgenden eine detaillierte Kategorisierung der Provenance-Informationen bei digitalen Objekten vorgestellt. Diese ist der Arbeit von Herschel et al. [HDB17], auf welcher der folgende Abschnitt basiert, entnommen und wird anhand von Abbildung 2.1 erläutert.

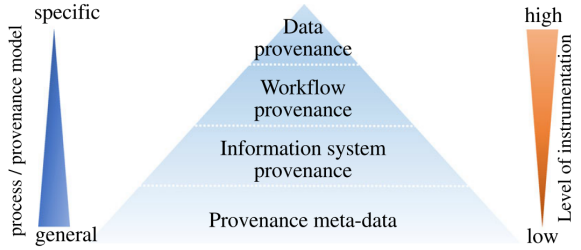


Abbildung 2.1: Provenance hierarchy. Abbildung aus [HDB17].

Die Kategorisierung kann, wie in Abbildung 2.1 zu sehen ist, als Pyramide, mit zur Spitze hin zunehmenden Einschränkungen, aufgefasst werden. Die Basis bildet hierbei **Provenance meta-data**, welche die allgemeinste Form der Provenance-Informationen darstellt. Durch diese Informationen kann ohne weitere Einschränkungen ein beliebiger Produktionsprozess in beliebiger

Form beschrieben werden. Dies ist auch mit dem breiteren Ende des blauen Pfeils, der auf der linken Seite der Abbildung zu sehen ist, dargestellt. Die nächste Stufe bildet die **Information system provenance**. Auf dieser Stufe ist der Entstehungsprozess auf Prozesse beschränkt, die digitale Daten innerhalb eines Informationssystems produzieren. Als nächste Stufe wird **Workflow provenance** dargestellt. Hier gilt die zusätzliche Einschränkung, dass es sich bei dem Produktionsprozess um einen Workflow handeln muss. Die Spitze der Pyramide bildet **Data provenance**, bei der Provenance-Informationen mit der Auflösung einzelner Daten erfasst werden. Dies trifft üblicherweise auf strukturierte Datenmodelle zu, für die dann deklarative Anfragesprachen genutzt werden können, was wiederum zu dem hohen **Level of instrumentation** führt. Dies wird durch das breite Ende des orangen Pfeils auf der rechten Seite der Abbildung dargestellt.

Data und Workflow Provenance sind im Rahmen dieser Arbeit von besonderer Bedeutung und können nach ihrer Granularität und Form noch weiter eingeordnet werden. In den Abschnitten 2.2.1 und 2.2.2, die auf Herschel et al. basieren, wird dies genauer erläutert.

2.2.1 Granularität

Die Granularität der gesammelten Provenance-Informationen wird auf einer Skala von grob bis fein eingestuft. Das eine Ende der Skala wird im Englischen als **coarse grained** und das andere als **fine grained** bezeichnet. Bei der Einordnung sind alle Zwischenstufen möglich.

- Provenance-Informationen werden als **coarse grained** bezeichnet, wenn es nicht möglich ist den Pfad einzelner Data-Items auch innerhalb der Module eines Workflows zu verfolgen. Eine formale Definition für coarse grained ist bei Herschel et al. zu finden:

  „[...] we qualify provenance as coarse-grained if the entire output O_m of a module $m \in M$ depends on its input I_m as a whole and the complete execution context C_m, and this for all modules in M. That is, we have $\forall m \in M : m \times I_m \times C_m \rightarrow O_m$.“ [HDB17]

  Dies bedeutet, dass keine Angaben zu einzelnen Daten $o \in O_m$ vorhanden sind.

- Provenance-Informationen werden als **fine grained** bezeichnet, wenn anhand dieser der vollständige Entwicklungsprozess jedes Data-Items innerhalb jedes Moduls eines Workflows nachvollzogen werden kann. Dies bedeutet das für jedes Data-Item bekannt ist, welche der Eingabeinformationen und welche der Kontextinformationen für die Erstellung des Data-Items relevant waren. Analog zu oben, ist bei Herschel et al. auch hierzu eine formale Definition angegeben:

  Fine grained provenance *„reflects that an item $o \in O_m$ produced by a Workflow module $m \in M$ may not depend on the complete input I_m and entire context C_m, but only on subsets $I'_m \subseteq I_m$ and $C'_m \subseteq C_m$. The finest-grained provenance is obtained when $\forall m \in M, \forall o \in O_m : m \times I'_m \times C'_m \rightarrow o$ and I'_m and C'_m are both minimal.“* [HDB17]

2.2.2 Form

Bezüglich ihrer Form werden Provenance-Informationen üblicherweise nach Prospektive Provenance, Retrospektive Provenance und Evolution Provenance unterschieden. Während es sich bei der Granularität um einen fließenden Übergang von fein zu grob handelt, so liegen hier Kategorien vor.

- **Prospektive Provenance** kann bereits vor einer Ausführung des Workflows anhand der Workflowbeschreibung und den gesetzten Parametern erfasst werden. Als prospektive Provenance-Informationen werden alle Informationen bezeichnet, welche die Struktur und den statischen Kontext eines Workflows erfassen.
- **Retrospektive Provenance** kann erst ab dem Beginn einer Workflowausführung erfasst werden und wird daher auch als Runtime Provenance bezeichnet [MBBL15b]. In diese Kategorie fallen Informationen zur Ausführung einzelner Teile des Workflows, der Laufzeitumgebung und der verwendeten sowie erstellten Ressourcen.
- **Evolution Provenance** erfasst alle Veränderungen der Eingaben zwischen mindestens zwei Workflowausführungen. Damit sind sowohl Änderungen an der Definition des Workflows, als auch an den Eingabedaten oder dem Ausführungskontext gemeint.

Bei dem für diese Arbeit verwendeten YW-Prototyp werden Provenance-Informationen auf einem eher groben Level erfasst. Prospektive Provenance-Informationen sind in Form eines Graphen verfügbar. Retrospektive Provenance-Informationen werden angeboten, jedoch wird keine Aufzeichnung zur Laufzeit durchgeführt. Evolution Provenance wird durch YW nicht unterstützt. Eine ausführliche Vorstellung und begründete Einordnung des YW-Prototyps befindet sich in Abschnitt 2.3.

2.3 YesWorkflow

Obwohl Scientific Workflow Management Systeme eine Vielzahl an Vorteilen bieten, die in Abschnitt 2.1.4 erläutert werden, so stellt die oft aufwendige Einarbeitung eine Barriere für Nutzer dar. Hingegen ist die Vertrautheit mit gängigen Skriptsprachen, wie z.B. R, Perl, MATLAB oder Python, groß und ermöglicht ein effizientes Arbeiten. Mithilfe von YW ist es möglich für beliebige Skripte Provenance-Informationen zu erfassen und zu nutzen [MBBL15b]. In den folgenden Abschnitten wird YW vorgestellt und die erfassten Informationen hinsichtlich Granularität und Form eingeordnet. Des Weiteren wird die Kommentar-Syntax von YW, soweit sie für diese Arbeit relevant ist, anhand von Beispielen, erläutert.

2.3.1 Grundgedanke und Vorteile von YesWorkflow

Wie in der Arbeit von McPhillips et al. [MSK+15], auf welcher dieser Abschnitt basiert, erläutert wird, können mithilfe von YW Provenance-Informationen beliebiger Skripte erfasst werden. Dazu ist keine Migration des Skripts, wie bei der Nutzung eines Scientific Workflow Management Systems, notwendig, sondern lediglich eine Erweiterung des bestehenden Programmcodes durch Kommentare, die einer definierten Syntax folgen. Diese Syntax muss durch den Nutzer erlernt werden, was jedoch

weitaus einfacher als die Einarbeitung in ein Scientific Workflow Management System ist. YW erkennt und wertet die Kommentare, welche als Annotationen bezeichnet werden, aus. Damit können Informationen, die sonst implizit in einem beliebigen Skript vorhanden sind, erfasst und graphisch dargestellt werden. Dies bedeutet konkret, dass die einzelnen Module und Datenflüsse innerhalb des Skripts erfasst werden können. Durch die YW-Annotationen werden sowohl prospektive als auch retrospektive Provenance-Informationen, was durch die YW-Recon-Erweiterung ermöglicht wird, erfasst.

2.3.2 Annotationssyntax

Basierend auf McPhillips et al. [MBBL15b] wird in diesem Abschnitt die Syntax, der für diese Arbeit relevanten YW-Annotationen vorgestellt und anhand von Beispielen erläutert. Um mit YW Provenance-Informationen bei einem Skript zu erfassen, werden die definierten YW-Annotationen, als Kommentare in der verwendeten Skriptsprache, eingefügt. In Abbildung 2.2 ist hierzu ein Ausschnitt aus einem Python-Skript zur Datensammlung bei Proteinkristallen, das um YW-Annotationen erweitert wurde, zu sehen.

Programm Block Ein Programm Block, der oft durch Programm oder Block abgekürzt wird, wird in YW durch eine @BEGIN- und eine @END-Annotation gekennzeichnet. Er deklariert einen Berechnungschritt innerhalb des Skripts, der als Eingabe Daten bekommt und als Ausgabe Daten erstellt. In Abbildung 2.2 sind zwei Programmblöcke zu sehen. Der erste beginnt in Zeile 1 und endet in Zeile 12. Der zweite beginnt in Zeile 14 und endet in Zeile 23. Ein Programm Block kann, beliebig geschachtelt, mehrere weitere Blöcke enthalten.

Port Mit Port wird eine Stelle im Programmcode gekennzeichnet, an der Daten für einen Programm Block oder einen Workflow gelesen oder geschrieben werden. Die Syntax für Daten, die verarbeitet werden sollen, ist hierbei ein @IN, wie es in Abbildung 2.2 bei Zeile 16 zu sehen ist. Für Parameter, die angeben wie der Block Daten verarbeitet, wird die @PARAM-Annotation, wie sie in Zeile 2 zu finden ist, verwendet. Ausgabewerte eines Blocks, zu sehen in Zeile 4, werden mit der @OUT-Annotation gekennzeichnet. Die @AS-Annotationen, zu sehen in Zeile 4 und 16, dienen dazu die Verbindung zwischen dem wissenschaftlichen Konzept (hier das Erstellen von raw_image-Dateien) und den Objekten im Programmcode herzustellen. In der graphischen Darstellung sind die so angelegten Pseudonyme zu sehen.

Channel Ein Channel bildet einen Pfad für Daten ab. Er stellt die Verbindung zwischen einer @IN- oder @PARAM- und einer @OUT-Annotation her. Ein solcher Pfad ist in Abbildung 2.2 von Zeile 4 zu Zeile 16 zu verfolgen. Anhand der Bezeichnungen werden durch YW Channels erkannt, die allerdings nicht explizit annotiert werden müssen.

YesWorkflow Recon-Erweiterung für retrospektive Provenance

Dieser Absatz basiert auf [MBBL15b]. In Abbildung 2.2 sind in Zeile 5, 17 und 18 @URI-Annotationen zu sehen. Sie ermöglichen das Erfassen von retrospektiven Provenance-Informationen und werden in der Recon-Erweiterung genutzt. Mithilfe dieser Annotationen ist es möglich den Zusammenhang zwischen den konzeptionellen Datenelementen und den tatsächlich erstellten oder gelesenen Dateien, und ihrem Speicherort, abzubilden. In der Praxis sind dies häufig die angestrebten Informationen. Beispielsweise sind die von einem Skript erstellten Dateien an sich von geringer Relevanz, während ihre Namensgebung, Anzahl oder ihre Verteilung innerhalb einer Ordnerstruktur einen hohen Informationsgehalt für den Nutzer hat. Mithilfe von YW lässt sich eine Zuordnung von erstellten Dateien zu Codeblöcken erfassen. In Abbildung 2.2 wird so beispielsweise in Zeile 5 definiert, wo und nach welcher Namenskonvention die raw_image-Dateien aus Zeile 4 angelegt und gespeichert werden. Da bei YW auf eine Provenance Aufzeichnung zur Laufzeit verzichtet wird, sind die retrospektiven Provenance-Informationen erst nach Abschluss der Ausführung eines Skripts verfügbar. Dies bietet den Vorteil, dass kein Overhead bei der Ausführungszeit entsteht. Der Nachteil ist allerdings, dass lediglich der Stand am Ende einer Ausführung erfasst werden kann.

```
1  # @BEGIN collect_data_set
2  # @PARAM cassette_id  @PARAM accepted_sample @PARAM num_images @PARAM energies
3  # @OUT sample_id @OUT energy @OUT frame_number
4  # @OUT raw_image_path @AS raw_image
5  # ... @URI file:run/raw/{cassette_id}/{sample_id}/e{energy}/image_{frame_number}.raw
6  run_log.write("Collecting data set for sample {0}".format(accepted_sample))
7  sample_id = accepted_sample
8  for energy, frame_number, intensity, raw_image_path in collect_next_image(
9                                  cassette_id, sample_id, num_images, energies,
10                                 "run/raw/{cassette_id}/{sample_id}/e{energy}/image_{frame_number:03d}.raw"):
11     run_log.write("Collecting image {0}".format(raw_image_path))
12 # @END collect_data_set
13
14 # @BEGIN transform_images
15 # @PARAM sample_id @PARAM energy @PARAM frame_number
16 # @IN raw_image_path @AS raw_image
17 # @IN calibration_image @URI file:calibration.img
18 # @OUT corrected_image @URI file:run/data/{sample_id}/{sample_id}_{energy}eV_{frame_number}.img
19 # @OUT corrected_image_path @OUT total_intensity @OUT pixel_count
20     corrected_image_path = "run/data/{0}/{0}_{1}eV_{2:03d}.img".format(sample_id, energy, frame_number)
21     (total_intensity, pixel_count) = transform_image(raw_image_path, corrected_image_path, "calibration.img")
22     run_log.write("Wrote transformed image {0}".format(corrected_image_path))
23 # @END transform_images
```

Abbildung 2.2: Mit YW-Annotationen versehener Ausschnitt aus einem Python-Skript zur Datensammlung bei Proteinkristallen [MBBL15b].

2.3.3 Graphische Darstellung

In diesem Abschnitt werden die von YW angebotenen graphischen Darstellungen, basierend auf der Arbeit von McPhillips et al. [MSK+15], vorgestellt. Zudem wird der Bezug zwischen dem annotierten Skript, das in Abbildung 2.2 zu sehen ist, und der graphischen Darstellung, zu sehen in Abbildung 2.3, hergestellt. Von YW wird Graphviz[1] genutzt, um die aus einem Skript erfassten Provenance-Informationen als gerichtete Graphen darzustellen.

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

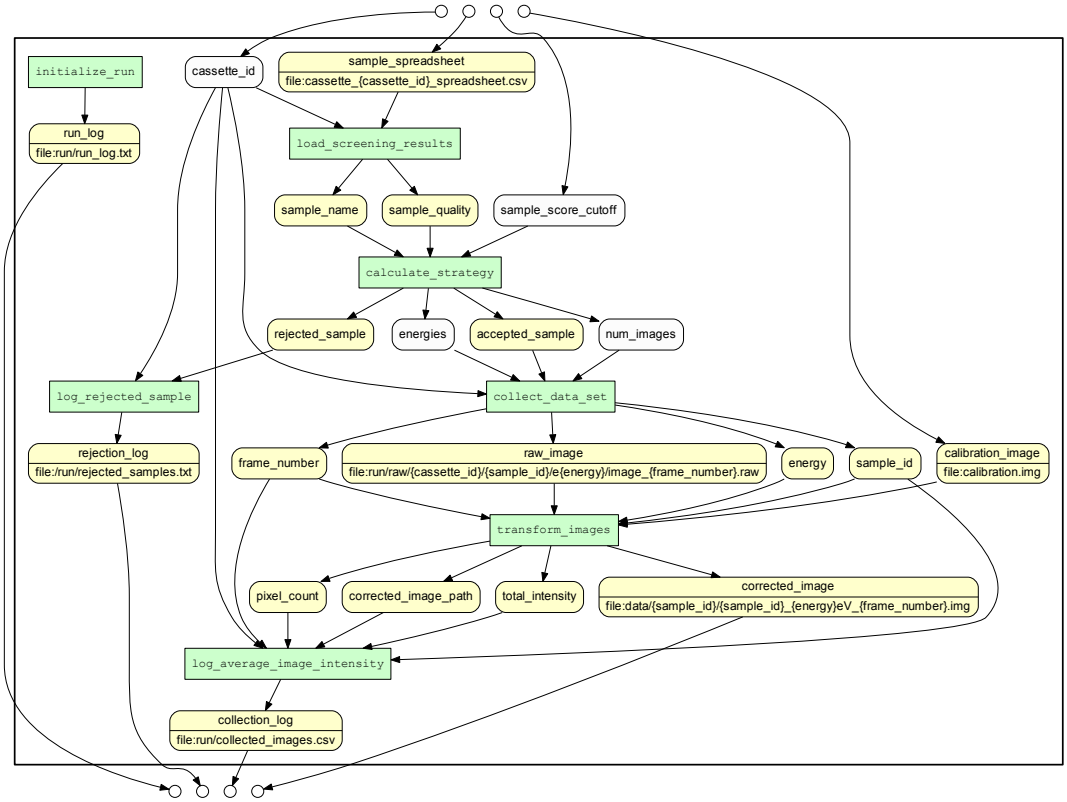


Abbildung 2.3: Kombinierte Darstellung der prospektiven Provenance-Informationen zu einem Python-Skript zur Datensammlung bei Proteinkristallen [MBBL15a].

Es werden drei Darstellungsvarianten angeboten:

- Bei der **prozessorientierten Workflowdarstellung** repräsentieren die Knoten Programmblöcke. Die Kanten stellen die Channels dar und die Bezeichnungen an den Kanten weisen auf Datenelemente hin.
- Bei der **datenorientierten Workflowdarstellung** werden die Channels als Knoten dargestellt und die Programmblöcke stehen als Bezeichnung über den Kanten.
- In der **kombinierten Workflowdarstellung** werden sowohl Programmblöcke als auch Daten als Knoten dargestellt. Da die kombinierte Darstellung für diese Arbeit am geeignetsten ist und in folgenden Beispielen Verwendung findet, ist hierzu ein Beispiel in Abbildung 2.3 zu sehen. Dieser Graph beinhaltet unter anderem auch den Teil des Skripts, der in Abbildung 2.2 zu sehen ist. Im Folgenden wird der Zusammenhang zwischen Programmcode und der Darstellung als Graph erläutert.

  In Abbildung 2.2 sind die beiden Programmblöcke collect_data_set und transform_images zu sehen. Diese werden in Abbildung 2.3 als **grüne Knoten** in der unteren Hälfte des Graphen dargestellt.

Der Parameter cassette_id (Zeile 2 in Abbildung 2.2) ist oben im Graphen als **weißer Knoten** zu sehen.
Die Ein- und Ausgabewerte, wie z.B. raw_image_path (Zeile 16 im Skript) und corrected_image_path (Zeile 19 im Skript) sind als **gelbe Knoten** in der unteren Hälfte des Graphen dargestellt.

2.3.4 Durch YesWorkflow verfügbare Provenance-Informationen und deren Einordnung

In den Abschnitten 2.2.1 und 2.2.2 werden zwei Dimensionen von Provenance-Informationen, Granularität und Form, erläutert. In diesen Abschnitt werden die von YW bereitgestellten Provenance-Informationen zusammengefasst und das System hinsichtlich Granularität und Form eingeordnet.

Anhand der Annotationen in einem Skript, kann YW bestimmen und in Graphdarstellungen anzeigen, welche Codeblöcke es gibt und in welcher Reihenfolge diese aufgerufen werden. Des Weiteren wird erfasst, welche Parameter oder Daten ein Codeblock liest beziehungsweise verwendet und welche er erstellt oder modifiziert. Durch diese Informationen lässt sich ableiten, welche Codeblöcke voneinander abhängig sind beziehungsweise Daten weiterverarbeiten. Hieraus ergibt sich hinsichtlich der Form der vorliegenden Provenance-Informationen, dass das Erfassen von prospektiven Provenance-Informationen von YW unterstützt wird.

Durch die Verwendung der Recon-Erweiterung sind zusätzlich Informationen verfügbar, welche Dateien nach der Ausführung des gesamten Skripts existieren. Auch die angelegte Ordnerstruktur wird durch YW erfasst. Hierbei ist anzumerken, dass nur der Endstand der Ordner und enthaltenen Dateien von YW erfasst wird. Des Weiteren werden auch keine Informationen zu verwendeten Ressourcen, der benötigten Zeit oder der Laufzeitumgebung erfasst. Dies bedeutet hinsichtlich der Form, dass die vorliegenden retrospektiven Provenance-Informationen als eingeschränkt einzustufen sind.

Die Granularität der erfassten Provenance-Informationen ist als eher grob einzustufen. Dies ist damit zu begründen, dass einzelne Data-Items nicht innerhalb eines Programmblocks verfolgbar sind. Dennoch ist bekannt, welche Teilmenge der Eingabeinformationen und welche Teilmenge der Kontextinformationen zu erstellten Dateien geführt haben.

Zusätzlich ist in diesem Abschnitt anzumerken, dass durch YW keine Werte von Parametern oder gesetzten Variablen erfasst werden. Dies lässt sich auch damit begründen, dass zur Laufzeit bisher generell keine Informationen erfasst werden.

2.3.5 YesWorkflow-Befehle

Basierend auf der Dokumentation zu YW werden in diesem Abschnitt die verfügbaren Befehle vorgestellt [LKM].

Der Aufruf von YW sieht folgendermaßen aus:
yw <command> [source file(s)] [-c <name=value>]...

Laut Dokumentation werden folgende drei Befehle unterstützt

- **EXTRACT:** In den übergebenen, mit YW-Annotationen versehenen, Skripten werden mithilfe des EXTRACT-Befehls alle YW Kommentare extrahiert
- **MODEL:** Durch diesen Befehl wird der Workflow-Modell erstellt. Bei der Ausführung des Befehls wird zunächst indirekt der EXTRACT-Befehl vor dem eigentlichen MODEL-Befehl ausgeführt.
- **GRAPH:** Der zugehörige Graph kann durch diesen Befehl erzeugt werden. Das Resultat wird im Graphviz DOT-Format angezeigt bzw. gespeichert. Hierzu werden zunächst indirekt der EXTRACT- und MODEL-Befehl ausgeführt.

Des Weiteren gibt es noch den Befehl **RECON**, der laut Beschreibung im Quellcode einen Lauf aus gespeicherten Daten und Protokolldateien rekonstruiert. Hierzu werden zunächst indirekt der EXTRACT- und MODEL-Befehl ausgeführt.

2.3.6 Der Proteinkristall-Workflow

In Abbildung 2.3 ist ein Workflow zu sehen der zur Datensammlung bei Proteinkristallen verwendet wird. Dieser wird im Rahmen der Arbeit von McPhillips et al. [MBBL15b] als Beispiel diskutiert. Auch in der hier vorliegenden Arbeit wird er bei der Evaluation in Kapitel 6 als Beispiel genutzt.

Der Workflow besteht aus sieben Blöcken: initialize_run, load_screening_results, calculate_strategy, log_rejected_sample, collect_data_set, transform_images und log_average_image_intensity.

Üblicherweise wählt der Nutzer vor der Ausführung Werte für cassette_id und sample_score_cutoff und hinterlegt eine sogenannte sample_spreadsheet-Datei sowie ein calibration_image. Mit diesen Werten wird der Workflow ausgeführt und erstellt dabei raw_image-Dateien. Des Weiteren werden log-Dateien erstellt, die für diese Arbeit keine Relevanz haben.

2.4 Prädiktive Modelle

In diesem Abschnitt werden verschiedene Verfahren, die zur Vorhersage genutzt werden können sowie ihr Funktionsprinzip vorgestellt.

2.4.1 Überblick

Das Nachschlagewerk Britannica formuliert das Einsatzgebiet von prädiktiven Modellen wie folgt: „*Predictive modeling is used when the goal is to estimate the value of a particular target attribute and there exist sample training data for which values of that attribute are known.*“ [Enc18].

Dementsprechend handelt es sich bei prädiktiven Modellen um statistische Methoden, die für die Erzeugung einer Vorhersage verwendet werden. In der Literatur sind zahlreiche Verfahren für die Realisierung von prädiktiven Modellen bekannt. Die nachfolgenden Abschnitte geben einen groben Überblick. Für weitere Informationen wird auf die entsprechende Literatur verwiesen.

Nach Kuhn et al. können die zugrundeliegenden Regressionsmodelle von prädiktiven Modellen in drei Klassen eingeteilt werden [KJ13]:

Bei der ersten Klasse handelt es sich um nicht lineare Regressionsmodelle. Hierzu zählen beispielsweise künstliche neuronale Netze. Die ursprüngliche Idee dieser Netze basiert auf der Nachbildung des biologischen Vorbilds, eines neuronalen Netzes [Mit97]. Exemplarische Anwendungen sind beispielsweise die Handschrifterkennung [LBD+89] und Spracherkennung [LWH90]. Ein weiteres Verfahren aus der Klasse der nicht linearen Regressionsmodelle sind Support Vector Machines. Der folgende Abschnitt fasst das grundlegende Konzept von Support Vector Machines zusammen, vgl. [SS01]: Ausgangspunkt ist die Repräsentation der Trainingsdaten als Vektoren. Die Idee einer Support Vector Machine besteht nun darin, Vektoren unterschiedlicher Klassen durch eine sogenannte Hyperebene im Vektorraum bestmöglich zu separieren. Hierbei kann eine Transformation in einen höher dimensionalen Raum notwendig sein. Eine exemplarische Anwendungen ist beispielsweise die Prädiktion von Zahlungsunfähigkeiten [ML05].

Bei der zweiten Klasse handelt es sich um Regressionsmodelle auf Basis von Entscheidungsbäumen und Regeln. Die grundlegende Idee bei Entscheidungsbäumen besteht darin, eine Zielfunktion durch eine Baumstruktur abzubilden. Die Blätter des Baumes stellen dabei die Klassen dar. Anwendungen von Entscheidungsbäumen finden sich beispielsweise in medizinischen Diagnosesystemen [Mit97].

Im Rahmen dieser Arbeit wird die dritte Klasse der linearen Regressionsmodelle näher betrachtet. Diese werden in Kapitel 2.4.2 erläutert.

2.4.2 Prädiktion auf Basis linearer Regressionsmodelle

Bei der Prädiktion mittels linearer Regression handelt es sich um ein gängiges Verfahren zur Schätzung numerischer Variablen [WFH11]. Dabei wird angenommen, dass sich eine abhängige Variable, das sogenannte Target, als Linearkombination einer oder mehrerer unabhängiger Variablen, auch Features genannt, abbilden lässt.

Für die Prädiktion der abhängigen Variable Y durch genau eine unabhängige Variable X gilt folgender Zusammenhang:

$$Y = \beta_0 + \beta_1 X \tag{2.1}$$

Hierbei sind β_0 und β_1 die Parameter des Prädiktionsmodells. Diese Parameter müssen i.d.R. anhand von Trainingsdaten ermittelt werden. Für den Fall einer Trainingsmenge mit $i = 1 \ldots n$ Datensätzen (x_i, y_i) gilt für die Regression mit genau einer unabhängigen Variable X:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{2.2}$$

Im Rahmen der Regressionsanalyse werden β_1 und β_2 als Regressionsparameter bezeichnet und so gewählt, dass die Trainingsdaten möglichst genau beschrieben werden, d.h. die Abweichungen ϵ_i möglichst klein sind.

Das dargestellte Verfahren kann für die Anwendung einer Prädiktion durch mehrere unabhängige Variablen entsprechend erweitert werden. Für die Prädiktion einer abhängigen Y Variable durch m unabhängige Variablen $X_1 \dots X_m$ gilt dabei:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_m X_m \tag{2.3}$$

Hierbei sind nun $\beta_1 \dots \beta_m$ die Parameter des Regressionsmodells. Für den Fall einer Trainingsmenge mit $i = 1 \dots n$ Datensätzen $((x_{1i}, \dots, x_{mi}), y_i)$ kann das Regressionsmodell wie folgt beschrieben werden:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{21} & \dots & x_{m1} \\ 1 & x_{12} & x_{22} & \dots & x_{m2} \\ \vdots & & & & \\ 1 & x_{1n} & x_{2n} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{2.4}$$

Unter der Voraussetzung $m < n$ können nun die Regressionskoeffizienten $\beta_1 \dots \beta_m$ beispielsweise mittels der Kleinstquadratmethode ermittelt werden.

2.4.3 Prädiktion auf Basis polynomieller Regressionsmodelle

Prädiktionen, denen eine polynomielle Regression zugrunde liegt gehen von der Annahme aus, dass sich der Zusammenhang zwischen einer abhängigen Variable Y und einer unabhängigen Variable X durch ein Polynom m-ten Grades beschreiben lässt [RA18]. Dies kann wie folgt dargestellt werden:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_m X^m \tag{2.5}$$

Hierbei sind nun $\beta_1 \dots \beta_m$ die Parameter des Regressionsmodells. Das zugehörige Regressionsmodell kann für eine Trainingsmenge mit $i = 1 \dots n$ Datensätzen (x_i, y_i) wie folgt dargestellt werden:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{2.6}$$

ersetzt man nun:

$$z_i = x^i \tag{2.7}$$

so ergibt sich folgende Gleichung:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & z_{11} & z_{21} & \dots & z_{m1} \\ 1 & z_{12} & z_{22} & \dots & z_{m2} \\ \vdots & & & & \\ 1 & z_{1n} & z_{2n} & \dots & z_{mn} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{2.8}$$

Dies entspricht dem Gleichungssystem einer linearen Regression mit m Parametern und kann beispielsweise durch die Kleinstquadratmethode gelöst werden, falls $m < n$. Zu beachten ist hierbei, dass die Voraussetzung der Unabhängigkeit der Variablen $Z_1 \dots Z_m$ weiter gilt.

2.4.4 Qualitätsbestimmung eines Modells

Laut [Chi18] ist es gängig, erstellte Modelle direkt mit vorliegenden Daten zu testen um ihre Qualität einzuschätzen. Dazu werden die vorliegenden Datensätze in Trainings- und Testdaten eingeteilt. Die Trainingsdaten werden, wie der Name bereits sagt, zum Training, also zum Erstellen der Modelle, die Testdaten, zum Testen der Modelle, verwendet. Die Abweichung zwischen den vorhergesagten Werten (y_n^{pred}) und den tatsächlichen Werten (y_n^{true}) können mittels verschiedener Maße bestimmt werden. Im Rahmen dieser Arbeit wird der negative median absolute error, der in Abschnitt 2.4.4 erläutert wird, verwendet.

Negative Median Absolute Error

Im Rahmen der Evaluation dieser Arbeit wird der negative median absolute error (NMAE) verwendet. Dieser ist für die Zielsetzung die Qualität einer Vorhersage einzuschätzen besonders geeignet, da ein absoluter Wert für den zu erwartenden Fehler bestimmt wird.

In [Chi18] ist folgende Definition für den median absolute error zu finden:

$$median\ absolute\ error = median(\left|y_1^{true} - y_1^{pred}\right|, \dots, \left|y_n^{true} - y_n^{pred}\right|) \tag{2.9}$$

Da für die Umsetzung, die Bibliothek scitkit-learn[2] verwendet wird, wird für diese Arbeit die Konvention, dass ein größerer Wert ein besseres Ergebnis darstellt als ein kleinerer Wert aus der Dokumentation übernommen. Daher wird der Wert des median absolute error in negierter Form verwendet.

[2] https://scikit-learn.org/stable/

k-fache Kreuzvalidierung zur Beurteilung

Um im Rahmen dieser Arbeit die vorliegende Datenmenge möglichst effizient zu nutzen, wird zum Testen der Modelle die k-fache Kreuzvalidierung genutzt. Sie stellt laut [RM18], auf welchem dieser Abschnitt basiert, ein robustes Verfahren zur Einschätzung der Leistung dar. Die vorliegenden Daten werden nicht nur in Trainings- und Testdaten, sondern in k Teilmengen aufgeteilt. k-1 Teilmengen werden zum Trainieren des Modells benutzt, während die k-te Teilmenge zum Testen verwendet wird. Dieses Verfahren wird k-mal wiederholt, sodass jede der k Teilmengen einmal als Testdatenmenge verwendet wird. Die so errechneten Gütewerte werden dann über alle k Modelle gemittelt.

Unter- und Überanpassung

Ist die Qualität der Vorhersage eines Modells für die Testdaten schlecht, so liegt dies laut [RM18] häufig an einer sogenannten Überanpassung (Overfitting) oder einer Unteranpassung (Underfitting). Die Unteranpassung rührt von einem zu einfachen Modell her, dass die Muster in den Trainingsdaten nicht ausreichend erkennt. Man spricht von einem großen Bias. Die Überanpassung rührt oft von zu vielen Parametern, die zu einem zu komplexen Modell für die Daten führen, her. Man spricht in diesem Fall von einer großen Varianz.

3 Verwandte Arbeiten

In diesem Kapitel wird ein Überblick über Arbeiten gegeben, die in Bezug zu der vorliegenden stehen. Da es, soweit die Recherche im Rahmen dieser Arbeit ging, keinen vergleichbaren Ansatz in direktem Bezug zu YW gibt, sind die hier vorgestellten Arbeiten jeweils in einem oder mehreren Punkten ähnlich, unterscheiden sich dann aber hinsichtlich des Anwendungsgebiets oder der Zielsetzung von dieser Arbeit.

Die erste Arbeit bietet einen allgemeinen Ansatz, wie wissenschaftliche Workflows vor einer Ausführung optimiert werden können. Im Rahmen der zweiten Arbeit werden anhand einfacher prädiktiver Modelle Vorhersagen für Ereignisse anhand eines erfassten Ereignisses erstellt und die Zeit für die Vorhersage optimiert. Die dritte Arbeit nutzt Provenance-Informationen, um eine Optimierung der Laufzeit bei einer Ausführung des Workflows zu erreichen. Im Rahmen der letzten vorgestellten Arbeit wurde ein System entwickelt, um in einem medizinischen Kontext Datenmenge und Berechnungsdauer für die Cloud Provisionierung vorhersagen. Hierbei wurde nur eine Vorhersage, jedoch keine Optimierung, angestrebt.

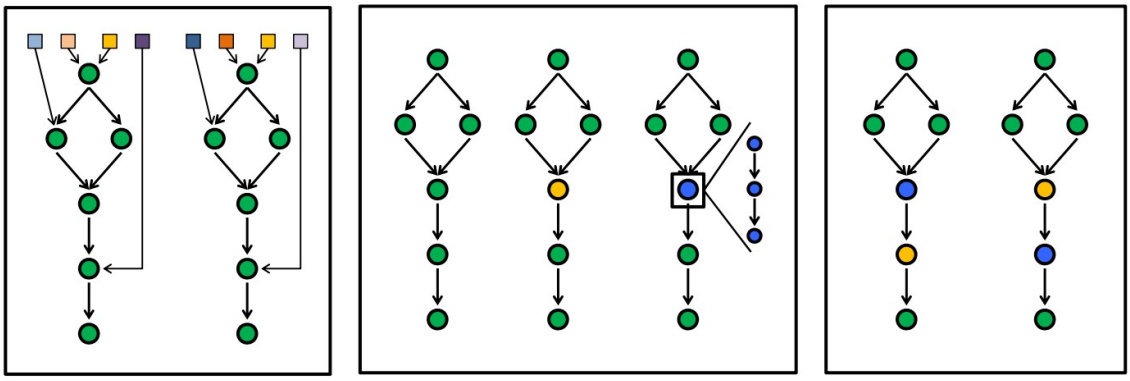
3.1 Allgemeiner Ansatz zur Optimierung wissenschaftlicher Workflows

In der Arbeit von S. Holl [Hol13], auf welcher dieser Abschnitt basiert, wird ein allgemeiner Ansatz für die automatisierte Optimierung von Scientific Workflows vorgestellt. Anhand von Erweiterungen für das WfMS Taverna[1] wird eine beispielhafte Umsetzung der Ziele präsentiert.

Im Rahmen der Arbeit wird dem in Abschnitt 2.1.3 vorgestellten Lebenszyklus für Scientific Workflows zusätzlich eine Optimierungsphase, zwischen Vorbereitung und Ausführung, hinzugefügt. In dieser Phase sollen Optimierungen stattfinden, bevor der gesamte Workflow ausgeführt wird.

Für diese Phase wird ein Konzept vorgestellt, dass aus drei Zielen besteht: Das erste Ziel besteht darin rechenintensive Schritte auszulagern. Im Rahmen der vorgestellten Umsetzung wird dies durch ein Plugin für Taverna, welches parallele Ausführungen auf hochperformanten Ressourcen erlaubt, realisiert. Ziel zwei ist es, eine Schnittstelle für Optimierungen bereitzustellen, die generisch ist und damit auf viele Anwendungsfälle übertragbar. Es werden drei Level für eine Workflowoptimierung aufgeführt, die in Abbildung 3.1 zu sehen sind.

[1] https://taverna.incubator.apache.org/



(a) Parameter Optimization (b) Component Optimization (c) Topology Optimization

Abbildung 3.1: Drei Level für die Optimierung von Workflows. Abbildung aus [Hol13].

Mit **Parameter Optimization** ist das Optimierungsproblem gemeint, die Kombination an Parametern zu finden, die das gewünschte, für die Fragestellung optimale, Ergebnis liefern. Üblicherweise bestimmen Wissenschaftler diese Informationen im Trial-and-Error-Verfahren oder orientieren sich an früheren Ausführungen. Beides ist sehr rechen- und zeitintensiv und führt nicht zwangsläufig zum optimalen Ergebnis. Mit **Component Optimization** ist das Optimierungsproblem gemeint, für einen Block des Workflows die optimale Implementierung zu finden. Dies kann auch, wie ganz rechts in Abbildung 3.1 (b) zu sehen, bedeuten, dass ein Block wiederum in ein untergeordneten Workflow weiter aufgeteilt wird. Mit **Topology Optimization** ist das Optimierungsproblem gemeint, die beste Reihenfolge für die Ausführung der Blöcke zu finden.

Taverna wurde dahingehend erweitert, dass beliebige formale Optimierungsalgorithmen für Probleme aller vorgestellten Level eingebunden werden können.

Das dritte Ziel ist es, den Optimierungsalgorithmus mit der Problemstellung zu verbinden. Hierzu wird im Rahmen des Prototyps eine API erstellt, sodass Entwickler sich nicht mit den Taverna-Eigenheiten beschäftigen müssen.

Somit wird in der Arbeit von S. Holl eine Methodik präsentiert, wie wissenschaftliche Workflows bereits vor der Ausführung, in der Optimierungsphase, auf verschiedenen Leveln optimiert werden können [Hol13]. In der hier vorliegenden Arbeit werden ebenfalls Optimierungen auf den drei von S. Holl betrachteten Leveln diskutiert. Hierbei ist jedoch die Zielsetzung prädiktive Modelle zu nutzen, um anhand gesammelter Informationen aus bereits durchgeführten Ausführungen Optimierungen zu erreichen.

3.2 Zeitoptimierte What-if Analyse

Der folgende Abschnitt basiert auf [KDTL17]. In der Arbeit von Ke et al. wird ein Provenance basiertes What-if Analyseverfahren für Data Mining Prozesse vorgestellt. Das Verfahren bezieht sich eigentlich auf einen wirtschaftlichen Kontext, lässt sich aber auf die hier vorliegende Arbeit übertragen. Mit den normalerweise verwendeten Data Mining Verfahren können die Verantwortlichen lediglich den Status eines Unternehmens in der Vergangenheit analysieren. What-if Verfahren

beschäftigen sich grundsätzlich mit der Frage, wie sich die Ausgabewerte verändern würden, wenn man die Eingaben ändert. In diesem Kontext werden in der Arbeit von Ke et al. die Zusammenhänge zwischen Eingaben und Ausgaben bestimmt. Der Nutzer kann nun Parameter verändern und bekommt eine Vorhersage, wie sich die Ausgaben damit in der Vergangenheit entwickelt hätten. Die Vorhersage basiert zum einen auf den bestimmten Zusammenhängen zwischen den erfassten Werten - dazu werden dann die notwendigen Berechnungsschritte neu ausgeführt. Zum anderen werden allerdings, wann immer es möglich ist, erfasste Werte für Teilschritte und Teilergebnisse verwendet. In der Evaluation konnte gezeigt werden, dass mit dem Verfahren, relevante Teilschritte zu bestimmen und nur diese neu zu berechnen, in bestimmten Kontexten eine Zeitersparnis erreicht werden kann.

Das von Ke et al. vorgestellte Verfahren, nur Teilschritte neu zu berechnen und erfasste Daten soweit es möglich ist wiederzuverwenden, kann auf die im Rahmen dieser Arbeit betrachtete Vorhersage von Ausgaben eines Workflows übertragen werden. Es wird in ähnlicher Form, in einem anderen Kontext, auch in Abschnitt 3.3 beschrieben. Während bei Ke et al. ein Erkenntnisgewinn anhand der erfassten Daten die Zielsetzung ist, wird im Rahmen der hier vorliegenden Arbeit eine Optimierung der durchgeführten Schritte oder gewählten Parameter angestrebt.

3.3 „Smart“-Reruns durch Provenance-Informationen

Ähnlich der in Abschnitt 3.2 vorgestellten Arbeit werden auch für die „Smart“-Reruns im Rahmen des WfMSs Kepler Teilergebnisse wiederverwendet. Das Vorgehen wird anhand von [ABJ06] erläutert:
Ziel ist hierbei eine Verringerung der Ausführungsdauer eines Workflows. Dazu werden, wenn der Nutzer Parameter ändert, nicht alle Schritte des Workflows ausgeführt. Anhand der erfassten Provenance-Informationen werden Abhängigkeiten bestimmt und nur die Teile des Workflows neu berechnet, die von den Änderungen betroffen sind. Auch werden bereits erfasste Teilergebnisse aus früheren Ausführungen direkt zurückgegeben, wenn sie zu der aktuellen Anfrage passen. Somit wird Redundanz vermieden und die Ausführung beschleunigt.

Die Zielsetzung bei der Arbeit von Altintas et al. ist, wie bei der vorliegenden Arbeit unter anderem auch betrachtet wird, eine Optimierung hinsichtlich der benötigten Ausführungsdauer eines Workflows. Allerdings wird diese hierbei im Kontext eines WfMSs betrachtet und dazu erfasste Ausführungen in direkter Weise für die Optimierung genutzt.

3.4 Vorhersagen für die Cloud Provisionierung

Die letzte der hier vorgestellten Arbeiten stellt eine Variante des Ensemble Learnings dar. Es werden mehrere unterschiedliche prädiktive Modelle bezüglich eines Workflows, basierend auf den verfügbaren Provenance-Informationen, erstellt. Um dem Nutzer eine möglichst präzise Vorhersage anzubieten, wird daraus dann die vielversprechendste Vorhersage, in Abhängigkeit der aktuell verfügbaren Informationen, ausgewählt.

Das System ist eine Anwendung in einem medizinischen Kontext. Der Ansatz lässt sich aber in verallgemeinerter Form auf weitere Bereiche übertragen. Im Folgenden wird das System mit dem, anhand von Provenance-Informationen zur Performance von Workflows, die Laufzeit und das entstehende Datenvolumen bei speziellen Algorithmen mithilfe prädiktiver Modelle vorhergesagt werden können, basierend auf [HWW16] vorgestellt.

Es wurden mithilfe von Armbändern mit Bewegungssensoren die Bewegungsdaten mehrerer Tage von über 100 000 Patienten auf 3 Achsen mit 100Hz erfasst. Die übliche Größe einer so erstellten Bewegungsdatei beträgt 800MB, was über 100 Millionen Zeilen Daten entspricht. In klinischen Studien werden diese Daten mittels fachspezifischer Algorithmen hinsichtlich der Form der Bewegung (z.B. Ruhen, leichte Anstrengung oder Rennen) oder Veränderungen durch Bewegung (beispielsweise auf das Schlafverhalten) ausgewertet. Da sich die Abläufe oft ähneln und wiederholen, werden diese Berechnungen oftmals als Workflows umgesetzt. Für den PAC1-Algorithmus wird beispielsweise angegeben, dass ein einzelner PC für die Berechnungen ungefähr sechs Monate Zeit bräuchte. Dies führt zur Notwendigkeit diese aufwendigen Berechnungen in einer Cloud-Umgebung durchzuführen. Hierbei ist es von großer Relevanz die Laufzeit und das Datenvolumen eines solchen Workflows vorherzusagen, um die Kosten und die benötigten Ressourcen realistisch abzuschätzen.

Um den Rechenaufwand bei der Vorhersage zu reduzieren werden keine Informationen über den verwendeten Algorithmus benötigt. Lediglich die Größe der Eingabedaten und die im Quelltext gewählten und gesetzten Parameter werden als Features für die Modelle genutzt. Da Laufzeit und erstellte Datenmenge durch ein Modell vorhergesagt werden soll, werden diese ebenfalls geloggt. Als Laufzeit wird hierbei der Zeitraum bezeichnet, der mit dem Beginn der Ausführung eines Codeblocks beginnt und mit seinem Abschluss endet. Für jeden Block wird die verarbeitete Datenmenge erfasst. Da im Rahmen der eingesetzten Cloud-Umgebung im Vorfeld nie bekannt ist, welche Maschine genutzt wird, werden diese Daten (z.B. CPU-Geschwindigkeit, Arbeitsspeicher usw.) nicht geloggt. Lediglich die Art der ausführenden Maschine wird für die Vorhersagen erfasst. Ein Workflow wird dann mehrfach ausgeführt, um die Provenance-Informationen für jeden Block zu erfassen.

Mit den gesammelten Informationen werden dann für jeden Block mehrere Modelle, die auch unterschiedliche Informationen nutzen können, erstellt. Diese Modelle werden dann wiederum aneinandergereiht. In [HWW16] werden drei grobe Kategorien für die Modellerstellung vorgestellt:

1. Lineare Regressionsmodelle werden dazu genutzt, um die Laufzeit abhängig von den oben genannten Ausführungsdaten abzuschätzen.
2. Wenn es keinen linearen Zusammenhang zwischen der Laufzeit und den erfassten Ausführungsdaten gibt, so können verschiedene nicht-lineare Modelle verwendet werden. Dies sind z.B. polynomielle Regressionsmodelle oder neuronale Netze. Bei den von Hiden et al. untersuchten Fällen trat dieser Zusammenhang niemals auf, wäre jedoch bei größeren Datenmengen zu erwarten.
3. Wenn es keine Korrelation zwischen der Laufzeit und den erfassten Ausführungsdaten gibt, so wird die Laufzeit als der Mittelwert der Ausführungszeit aller erfassten Durchläufe des Blocks abgeschätzt.

Anhand eines Testdatensatzes wird dann die Qualität der einzelnen Modelle evaluiert und für eine Vorhersage das beste genutzt. Ein erneutes Training und damit eine Verbesserung der Modelle, wenn beispielsweise zusätzliche Informationen vorliegen, wurde im Rahmen der Arbeit von Hiden et al.

nicht betrachtet. Um auch Fälle abzudecken in denen keines der erstellten Modelle eine ausreichende Qualität der Vorhersage bieten kann, wird in der Arbeit von Hiden et al. auch eine Alternativlösung angegeben [HWW16]. Dies tritt z.B. dann ein, wenn der Block nie zuvor ausgeführt wurde oder die Daten, die bis zum aktuellen Zeitpunkt gesammelt wurden, für eine Vorhersage nicht ausreichend sind.

1. Falls es kein passendes Modell für die aktuelle Version eines Blocks gibt, wird ein Modell aus allen vorhandenen Ausführungen, über verschiedene Versionen hinweg, erstellt und für die Vorhersage genutzt.
2. Wenn keinerlei Modelle für einen Block verfügbar sind, es aber mindestens einen geloggten Datensatz zu dem Block gibt, so werden alle verfügbaren Werte gemittelt und als Abschätzung verwendet.
3. Wenn keinerlei Daten für einen Block vorliegen, so wird die durchschnittliche Dauer aller erfassten Blöcke und die durchschnittliche erstellte Datenmenge aller erfassten Blöcke als Abschätzung verwendet.
4. Falls das System erst initialisiert wurde und keine Daten vorliegen, so wird auch keine Vorhersage angeboten.

Wie in [HWW16] angeben, ist es das Ziel dieser Alternativlösungen stets die zu dem Zeitpunkt beste mögliche Vorhersage anzubieten. Durch eine Markierung wird für den Nutzer kenntlich gemacht, wie die aktuelle Vorhersage erstellt wurde und die Qualität einzustufen ist.

Im Rahmen der Arbeit von Hiden et al. wurde die Qualität der erstellten Vorhersagen zum einen unter idealen Bedingungen und zum anderen in einer öffentlichen Cloud-Umgebung evaluiert [HWW16]. Zur Evaluation wurden zwei unterschiedliche Verfahren eingesetzt auf die an dieser Stelle nicht weiter eingegangen wird.

Es wurde ein Workflow verwendet, dessen Blöcke einfache Datenmanipulationen vornehmen. Es ist also eine hohe Korrelation zwischen Eingabedatenvolumen und Laufzeit zu erwarten. Dieser wurde 250 mal ausgeführt. Bei der idealisierten Umgebung konnte mit nahezu 100% Accuracy die erstellte Datenmenge für jeden Block vorhergesagt werden. Dazu wurde ein lineares Modells mit lediglich der Eingabedatenmenge als Feature verwendet. Ebenso war die Vorhersage der Laufzeit der Blöcke sehr gut. Für diese Vorhersage waren, zusätzlich zu der Eingabedatenmenge, noch die Parameter als weiteres Feature notwendig. In einigen Fällen war die Vorhersage auffallend schlecht, was die Autoren zum einem dadurch erklären, dass die Trainingsdatenmenge nicht die volle Vielfalt der Werte umfasste, und zum anderen durch mehr verfügbare Ressourcen zum Ende der Laufzeit. Dies rührt daher, dass in dem verwendeten Setup mehrere Workflows parallel ausgeführt werden und zum Ende der Laufzeit einige bereits abgeschlossen sind, was dann zu mehr Ressourcen für die noch nicht abgeschlossenen führt. Diese können dann schneller als erwartet ausgeführt werden. Als zweites Experiment wurde dasselbe Vorgehen in der Amazon EC2[2] Cloud-Umgebung ausgeführt. Die erstellten Modelle waren von vergleichbarer Qualität, allerdings waren die Abweichungen bei der Vorhersage größerer Eingabedatenmenge ungenauer, was die Autoren jedoch nicht weiter begründen.

[2] https://aws.amazon.com/de/ec2/

Im Rahmen der Evaluation wird noch eine weitere Vorhersage erstellt und ausgewertet. Hierbei wurde ein Algorithmus betrachtet, der zur Auswertung von Bewegungsdaten eingesetzt wird. Auch hier war die Vorhersagequalität vergleichbar. Die beobachteten Abweichungen erklären die Autoren durch äußere Umstände, z.B. durch gebündelten Upload der Daten und daraus resultierende Stoßzeiten bei der Verarbeitung.

Bei der hier betrachteten Arbeit ist die Zielsetzung eine Abschätzung der Datenmenge und Laufzeit in Hinblick auf die Cloud Provisionierung. Es wird keine Optimierung angestrebt. Die für die Vorhersage von Hiden et al. eingesetzten Methoden können mit der vorliegenden Arbeit verglichen werden [HWW16]. Allerdings ist hierbei die Zielsetzung, stets eine Vorhersage bereitzustellen, während im Rahmen der vorliegenden Arbeit untersucht wird, ab wann die Qualität einer Vorhersage für die betrachteten Fälle ausreichend ist.

4 Lösungsansatz

In diesem Kapitel wird zunächst erläutert, welche Vorhersagen bei Workflows für diese Arbeit von Interesse sind. Dann werden in allgemeiner Form die möglichen Ansatzpunkte für Optimierungen von Workflows vorgestellt. Es wird diskutiert, welche der Optimierungen mit den bereits verfügbaren Provenance-Informationen unterstützt werden können, und welche weiteren Informationen für Optimierungen genutzt werden könnten. Im nächsten Abschnitt wird dann der konkrete Ablauf einer solchen Optimierung mithilfe von Provenance-Informationen vorgestellt. Im letzten Abschnitt dieses Kapitels wird darauf eingegangen, wie sich dieses Konzept in YW integrieren und somit nutzen lässt und welche potenziellen Schwachstellen zu evaluieren sind.

4.1 Mögliche Vorhersagen bei einem Workflow

In diesem Abschnitt werden für diese Arbeit relevante Vorhersagen aufgeführt, die durch prädiktive Modelle in Bezug auf einen Workflow erstellt werden können. Die möglichen Vorhersagen werden dazu in zwei Kategorien eingeteilt:

1. Eigenschaften
2. Eingabedaten, Parameter und Ausgabedaten

4.1.1 Eigenschaften

In diese Kategorie fallen Eigenschaften eines Workflows, die direkt als Zahlenwerte messbar sind.

Dauer

Sowohl bei Workflows mit hohem Rechenaufwand, als auch bei Workflows, die oft durchgeführt werden, ist es aus Nutzersicht von besonderem Interesse die **Dauer einer Ausführung**, bei gewählten Parametern und Eingabedaten, vor der Ausführung abschätzen zu können oder im Idealfall zu kennen. Dies lässt sich wiederum in zwei Punkte untergliedern: Zum einen kann es von Interesse sein, die Laufzeit des gesamten Workflows vor einer Ausführung zu kennen. Zum anderen kann es aber auch relevant sein, zusätzlich die Laufzeit einzelner Blöcke eines Skripts zu kennen.

Datenvolumen

Falls das Skript Daten generiert kann es von Interesse sein das voraussichtlich **erstellte Datenvolumen**, also den nötigen Speicherplatz, abzuschätzen. Dies kann entweder in allgemeiner Form als Abhängigkeit von Variablen für die Eingabedaten und Parameter angegeben werden. Oder als konkreter Wert, wenn die Eingabedaten und Parameter bereits durch den Nutzer festgelegt wurden. Es kann sowohl nur das Datenvolumen, das am Ende der Ausführung besteht, von Interesse sein, als auch das gesamte zur Laufzeit generierte und eventuell wieder verworfene Datenvolumen. Wie im letzten Abschnitt auch können die Informationen zu dem ganzen Skript oder zu einzelnen Blöcken relevant sein.

Diese Vorhersage ist besonders in Hinblick auf eine Cloud Computing Umgebung relevant und ermöglicht hier eine zielgerichtete Allokation von Speicherressourcen. Dies ist sowohl aus Anbietersicht, der den Gewinn steigern und die Ressourcen optimal vergeben möchte, als auch aus Nutzersicht, der nur die Ressourcen bezahlen möchte, die er tatsächlich benötigt, von Interesse.

Rechenleistung und Arbeitsspeicher

Neben der Dauer einer Ausführung und dem erstellten Datenvolumen kann es auch von Interesse sein, die von einem Workflow zur Ausführung **benötigte Rechenleistung und den Arbeitsspeicher** vor der Ausführung zu kennen. Wie bei den beiden anderen Kategorien ist es auch hier, je nach Zielsetzung sinnvoll, diese Eigenschaft für den gesamten Workflow oder für einzelne Blöcke des Workflows zu kennen.

4.1.2 Eingabedaten, Parameter und Ausgabedaten

Während oben nur Eigenschaften des Workflows oder von Teilen des Workflows betrachtet werden, sollen in dieser Kategorie die konkreten Werte für Eingabedaten, Parameter und Ausgabedaten vorhergesagt werden. Dies kann grundsätzlich in zwei Richtungen geschehen: Ausgehend von gewählten Eingabedaten und Parametern werden die Ausgaben bestimmt. Oder, in die entgegengesetzte Richtung werden anhand von gewählten Ausgaben die Eingabedaten und Parameter bestimmt.

Konkrete Daten

Je nach Problemstellung und Zielsetzung könnten **konkrete Daten**, beispielsweise wie der Wert eines Parameters zu wählen ist um eine Zielsetzung zu erreichen, vorhergesagt werden.

In diese Kategorie fällt auch die möglichst genaue Vorhersage von Ausgabewerten bzw. Ausgabedaten in kürzerer Zeit als ihre Berechnung. Dies wird als Surrogatmodellierung bezeichnet. Das gelernte Modell, das genutzt wird um die Ergebnisse eines Workflows (der in diesem Kontext dann auch als Modell betrachtet wird) vorherzusagen, bezeichnet man in diesem Kontext als Surrogat für den Workflow.

Eigenschaften der Daten

Eine abstraktere Herangehensweise ist es **Eigenschaften der Eingabedaten, Eigenschaften der Parameter oder Eigenschaften der Ausgabedaten**, also Metriken, vorherzusagen.

Hier könnte beispielsweise auch vorhergesagt werden, wie viele Datenobjekte eines bestimmten Typs erstellt werden - also die Datenqualität als Eigenschaft. Damit könnte sichergestellt werden, dass nur relevante Daten erstellt werden. Die Anzahl an erstellten Datenobjekten gehört ebenso in diese Kategorie. Sie wird im Rahmen der Evaluation in Kapitel 6 weiter diskutiert.

4.2 Mögliche Optimierungen bei einem Workflow

In Abschnitt 4.1 sind Vorhersagen, die in Bezug auf einen Workflow von Interesse sein können, aufgeführt. In diesem Abschnitt wird darauf aufbauend erläutert, wie man einen Workflow durch Nutzung der Prädiktionen optimieren kann. Zusätzlich werden konkrete Beispiele, in welchem Kontext eine solche Optimierung von Interesse sein kann, genannt.

4.2.1 Optimierungen basierend auf der Vorhersage von Eigenschaften

Analog zu Abschnitt 4.1.1 wird die dort vorgestellte Kategorisierung auch für diesen Abschnitt übernommen.

Optimierungen basierend auf einer Vorhersage der Dauer

Wenn man die **Dauer einer Ausführung** abschätzen kann oder gar kennt, so bietet sich als Optimierungsziel eine **Zeitersparnis** an.

Betrachtet man an dieser Stelle eine Vorhersage der Ausführungsdauer des gesamten Workflows - die nur vorgenommen werden kann, wenn die Parameter und Eingabedaten festgelegt wurden - so kann eine Optimierung der Dauer einer Ausführung durch eine Anpassung der Eingaben durch den Nutzer erreicht werden. Möglicherweise wurden mehr Eingabedaten verwendet, als für die weitere Auswertung tatsächlich benötigt werden. Durch eine solche Vorhersage wird dem Nutzer die Auswahl sinnvoller Eingabedaten dann erleichtert. Auch eine parallele Ausführung, von voneinander unabhängigen Programmabschnitten, kann zu einer verringerten Laufzeit des erstellten Skripts führen.

Wenn Vorhersagen für die Ausführungsdauer einzelner Blöcke des Skripts verfügbar sind, so lassen sich potenzielle Schwachstellen identifizieren. Der Nutzer kann auf Programmabschnitte, die im Vergleich zu anderen viel Zeit in Anspruch nehmen, hingewiesen werden und diese dann eventuell verbessern. Dies könnte beispielsweise durch Veränderungen im Programmcode erreicht werden Möglicherweise kann auch die Reihenfolge, in der die Blöcke ausgeführt werden, so verändert werden, dass die Laufzeit verringert wird. Hierbei ist es erstrebenswert Programmabschnitte, welche die Datenmenge stark filtern, frühzeitig auszuführen, um so die Eingaben für die folgenden Programmblöcke zu verringern. Dies kann durch erfasste Abhängigkeiten, wie sie z.B. in YW angeboten werden, bestimmt werden.

Optimierungen basierend auf einer Vorhersage des Datenvolumens

Kann man das **erstellte Datenvolumen** abschätzen oder genau vorhersagen, so können auch hinsichtlich diesem Optimierungen am Workflow vorgenommen werden. Das Ziel hierbei ist entweder die möglichst genaue **Erstellung einer definierten Menge** oder, als Sonderfall davon, die **Verringerung der erstellten Menge**.

Wenn man den Fall betrachtet, dass man das erstellte Datenvolumen für den Workflow als Gesamtes kennt, so ließe sich auch hier eine Verringerung durch eine Anpassung der gewählten Eingaben und Parameter durch den Nutzer erreichen. Dasselbe gilt für die Erstellung einer definierten Menge. Die Erstellung einer definierten Menge kann sich beispielsweise aus einer konkreten Forschungsfrage ergeben, bei der ein Nutzer vorab weiß, wie viele Datensätze er für eine Auswertung benötigt.

Betrachtet man hingegen Vorhersagen für die einzelnen Blöcke, so lassen sich auch hier „Schwachstellen“ identifizieren. In diesem Fall sind hiermit Blöcke gemeint, die besonders viel Datenvolumen erstellen. Möglicherweise kann eine Verringerung des erstellten Volumens durch Anpassungen des Programmcodes durch den Nutzer erreicht werden. Oder es kann auch in diesem Fall die Reihenfolge der Ausführung der Blöcke so angepasst werden, dass das Datenvolumen frühzeitig gefiltert wird.

Optimierungen basierend auf einer Vorhersage von benötigter Rechenleistung und Arbeitsspeicher

Wenn eine Vorhersage der **benötigten Rechenleistung oder des Arbeitsspeichers** möglich ist, können auch hinsichtlich dieser Optimierungen am Workflow vorgenommen werden. Das angestrebte Ziel wäre in diesem Fall eine **Verringerung** der notwendigen Ressourcen oder der benötigten Zeit.

Gegebenenfalls kann ein Ansatz zur Problemlösung gewählt werden, der zwar eine höhere Rechendauer aufweist, dafür aber weniger Ressourcen insgesamt benötigt. Oder der mehr Ressourcen benötigt, dafür aber schneller hinsichtlich der benötigten Zeit ist. Dies könnte beispielsweise bedeuten, dass aufwendige Workflows nicht lokal, sondern in einer Cloud Umgebung ausgeführt werden, um mit mehr Ressourcen eine Zeitersparnis zu erreichen. Hat man beispielsweise verschiedene Implementierungen für einen Block zur Wahl, könnte so automatisch der effizienteste, hinsichtlich der Zielsetzung, ausgewählt werden. Auch kann eine solche Vorhersage genutzt werden, um einem Nutzer die Abschätzung der notwendigen Ressourcen zu erleichtern.

4.2.2 Optimierungen basierend auf der Vorhersage von Eingabedaten, Parametern und Ausgabedaten

In diesem Abschnitt werden mögliche Optimierungen analog zu der in Abschnitt 4.1.2 verwendeten Kategorisierung vorgestellt.

Optimierungen basierend auf einer Vorhersage konkreter Daten

Kann eine Vorhersage **konkreter Werte für Eingabedaten, Parameter oder Ausgabedaten** angeboten werden, so kann damit sowohl eine **Zeitersparnis** als auch eine **Ressourcenersparnis** erreicht werden. In diesem Fall ist mit Ressourcenersparnis sowohl Speicher, als auch Rechenleistung und Arbeitsspeicher gemeint.

Betrachtet man beispielsweise den Fall einer Vorhersage konkreter Parameter, so kann der k-means Algorithmus als Beispiel aus einem anderen Kontext betrachtet werden. Ziel des Algorithmus ist es Daten in zusammengehörige Gruppen aufzuteilen. Die Problemstellung ist es die optimale Menge an Gruppen, k, zu bestimmen, sodass sich die Daten innerhalb einer Gruppe ähneln und sich von den anderen unterscheiden. Üblicherweise werden mehrere Durchläufe mit unterschiedlichen k's ausgeführt. In diesem Beispiel würde die gewünschte Vorhersage das optimale k bestimmen oder zumindest den Bereich eingrenzen und somit die benötigte Zeit und die genannten Ressourcen verringern.

Optimierungen basierend auf einer Vorhersage von Eigenschaften von Daten

Lassen sich Vorhersagen für die **Eigenschaften von Eingabedaten, Eigenschaften von Parametern oder Eigenschaften von Ausgabedaten** erstellen, so kann damit sowohl eine **Erhöhung der Datenqualität**, eine **Zeitersparnis** als auch eine **Ressourcenersparnis** erreicht werden.

Betrachtet man auch hier den Fall, dass diese Vorhersage für Blöcke angeboten wird, so lassen sich damit Blöcke, die die Datenmenge stark filtern, bestimmen. Durch eine Optimierung der Reihenfolge lässt sich damit sowohl eine Zeitersparnis als auch eine Ressourcenersparnis erreichen.

Betrachtet man den Fall das eine Vorhersage für den ganzen Workflow angeboten wird, so lassen sich damit beispielsweise gewählte Eingabedaten und Parameter optimieren, sodass eine Zeit- und Ressourcenersparnis erreicht wird. Der Workflow wird also nur für Eingabedaten und Parameter ausgeführt, die auch zu relevanten Ergebnissen, also zu einer hohen Datenqualität, führen. Diese Erhöhung der Datenqualität stellt an sich bereits eine Optimierung dar. Dieses Szenario wird in Kapitel 6 weiter verfolgt.

4.3 Mögliche Nutzung der vorliegenden Informationen

In Abschnitt 2.3.4 werden die Provenance-Informationen, welche durch YW bereitgestellt werden, aufgeführt. In diesem Abschnitt werden nun die oben aufgeführten Optimierungen, die durch die bereitgestellten Informationen angestrebt werden können, diskutiert. Die Kategorisierung erfolgt wie in Abschnitt 4.2 nach den für eine Optimierung notwendigen Informationen.

4.3.1 Optimierungen basierend auf einer Vorhersage von Eigenschaften

Auch wenn von YW keine der oben genannten Informationen (Dauer, Datenvolumen, Rechenleistung und Arbeitsspeicher) in direkter Form erfasst wird, so können in dieser Kategorie dennoch Optimierungen mit den bereitgestellten Provenance-Informationen erreicht werden.

Mit YW kann anhand der verfügbaren prospektiven Provenance-Informationen bestimmt werden, ob es Parameter, Eingaben oder Blöcke gibt, die keinen Einfluss auf das Endergebnis haben. Diese können entfernt werden, was zu einer Optimierung der Dauer, des Datenvolumens und der notwendigen Ressourcen führen kann. Somit sind mit diesen Informationen Optimierungen hinsichtlich jeder der drei folgenden Unterkategorien möglich, weshalb sie an dieser Stelle und nicht erst in den jeweiligen folgenden Absätzen genannt werden.

Optimierungen basierend auf einer Vorhersage der Dauer

Bezüglich der Dauer einer Ausführung ist das Optimierungsziel eine **Zeitersparnis**. Grundsätzlich ist es hierzu notwendig die benötigte Zeit zu erfassen. Da YW keine zeitlichen Informationen bereitstellt, kann eine Prädiktion in diesem Sinne nicht angeboten werden.

Allerdings kann anhand von bereits verfügbaren Informationen entschieden werden, ob es Codeblöcke gibt, die voneinander unabhängig, also parallel, ausgeführt werden können. Dies bringt im Idealfall eine Zeitersparnis, im anderen Fall verändert sich die benötigte Zeit jedoch nicht.

Optimierungen basierend auf einer Vorhersage des Datenvolumens

In diesem Kontext wird die **Erstellung einer definierten Menge** oder eine **Verringerung der erstellten Menge** diskutiert.

Durch die erfassten Provenance-Informationen ist bekannt, was die Ein- und Ausgaben für einen Block sind. Damit können Programmabschnitte identifiziert werden, die viele Daten filtern bzw. nicht so viele Daten erstellen. Dadurch kann eine Optimierung der Reihenfolge und damit eine Verringerung des zwischenzeitlich erstellten Datenvolumens, sowie eine Zeitersparnis erreicht werden.

Allerdings wird anhand der Provenance-Informationen, welche bisher durch YW erfasst werden, lediglich der Stand zum Ende einer Ausführung erfasst und damit nur das zum Ausführungsende bestehende Datenvolumen oder die Anzahl der erzeugten Datenobjekte. Da jedoch keine Informationen zu den gewählten Eingaben und Parametern erfasst werden, kann hierzu kein Zusammenhang erschlossen werden.

Optimierungen basierend auf einer Vorhersage von Rechenleistung und Arbeitsspeicher

Von YW werden keine Systeminformationen erfasst, weshalb in dieser Unterkategorie auch keine Optimierungen mit den bereitgestellten Informationen angestrebt werden können.

4.3.2 Optimierungen basierend auf einer Vorhersage von Eingabedaten, Parametern und Ausgabedaten

Basierend auf dem Grundsatz von YW werden keine Informationen zur Laufzeit erfasst. Daher werden Werte, die Variablen zugewiesen werden durch YW bisher nicht erfasst. Mithilfe der Recon-Erweiterung und der @URI-Annotationen können aber Anfragen zu den erstellten oder gelesenen Daten beantwortet werden.

Optimierungen basierend auf einer Vorhersage konkreter Daten

Da zur Laufzeit zugewiesene Werte nicht erfasst werden, was auch vom Nutzer gewählte Parameter und Eingabedateien beinhaltet, kann hierzu keine Relation zu den erstellten Daten hergestellt werden.

Der Endzustand der erstellten Datenobjekte und die gelesenen Datenobjekte können anhand der @URI-Annotationen erfasst werden. Mithilfe der in Abschnitt 4.1.2 erwähnten Surrogatmodellierung könnte hierzu eine Vorhersage erstellt werden, die dann beispielsweise für eine Optimierung der gelesenen Datenobjekte verwendet werden kann.

Optimierungen basierend auf einer Vorhersage von Eigenschaften von Daten

In dieser Unterkategorie soll als Zielsetzung eine **Erhöhung der Datenqualität, Zeit- oder Ressourcenersparnis** erreicht werden. Anhand der @URI-Annotationen kann mithilfe der Recon-Erweiterung bestimmt werden, welche Dateien Eingaben und Ausgaben eines Workflows, beziehungsweise einzelner Schritte, waren. Mit dieser Information könnten Metriken zu den Dateien bereitgestellt und für Prädiktionen genutzt werden. Hiermit könnte beispielsweise eine Optimierung der Anzahl der Eingabedateien hinsichtlich der zu erwartenden Anzahl an Ausgabedateien durch einen Nutzer erfolgen. Wie bereits in Abschnitt 4.3.1 beschrieben wäre auch zu evaluieren, ob mit dieser Information eine optimierte Reihenfolge der Blöcke bestimmt werden kann.

4.4 Mögliche Nutzung weiterer Informationen

Im letzten Abschnitt wurde aufgeführt, welche der in Abschnitt 4.2 genannten Optimierungen unter Ausnutzung der bereits erfassten Provenance-Informationen unterstützt werden können. In diesem Abschnitt wird nun diskutiert, welche weiteren Provenance-Informationen erfasst werden können und welche Prädiktionen und damit Optimierungen dadurch unterstützt werden können.

Die Erfassung weiterer Informationen ist vor dem Hintergrund abzuwägen, dass dem Grundsatz von YW folgend eigentlich keine Provenance-Informationen zur Laufzeit erfasst werden sollen, um diese nicht zu erhöhen.

4.4.1 Erfassung von Eigenschaften

In diese Kategorie fällt die Erfassung von Dauer, Datenvolumen und Systeminformationen. Bisher wird lediglich das Datenvolumen durch die @URI-Annotationen in indirekter und eingeschränkter Form erfasst.

Am offensichtlichsten aus Nutzersicht erscheint die Erfassung der **Dauer** einer Ausführung. Entweder der einzelnen Blöcke, was allerdings ein Logging zur Laufzeit zwingend erforderlich macht. Oder der gesamten Laufzeit, was man mithilfe von Timestamps vor dem Beginn der Ausführung und nach ihrer Beendigung erfassen könnte. Um dies innerhalb von YW zu realisieren wäre allerdings eine Erweiterung von YW zur direkten Ausführung von Skripten notwendig. Das Problem beim Erfassen solcher Informationen ist allerdings, dass sie systemabhängig sind. Je nach verwendetem System kann die Dauer einer Ausführung stark variieren. Eine Lösung ist, nur das Verhältnis zwischen Dauer und Eingaben bzw. gewählten Parametern zu betrachten. Eine andere Lösung besteht darin, die Systemeigenschaften ebenfalls zu erfassen und in eine Vorhersage miteinzubeziehen.

In Bezug auf die Erfassung von Laufzeit ist anzumerken, dass hier nur eine Relation zwischen Eingabedaten, die über die @URI-Annotationen bestimmt werden können, und der Laufzeit möglich ist. Da von YW bisher die Werte, welche Parametern zugewiesen werden, nicht erfasst werden, kann hier auch kein Bezug hergestellt werden, der für eine Vorhersage nutzbar wäre. Wie bereits bei der Erfassung der Dauer beschrieben müsste hierfür YW dahingehend erweitert werden, dass Skripte auch innerhalb von YW ausgeführt werden können. Werte, die ein Nutzer bereits vor der Ausführung festlegt, könnten dann auch vor der eigentlichen Ausführung des Skripts durch YW erfasst werden. Werte, die Variablen während der Laufzeit zugewiesen werden, könnten zur Laufzeit des Skripts (beispielsweise durch manuelles Logging innerhalb des Skripts oder über die kombinierte Nutzung von YW und noWorkflow bei Python-Skripten) zusätzlich zu diesen erfasst werden.

Eine weiterer Ansatz besteht darin das **Datenvolumen** direkt zu erfassen. Bisher kann anhand der @URI-Annotationen das Datenvolumen, das den finalen Zustand der Ausführung darstellt, bestimmt werden. Möchte man auch Informationen zu Zwischenständen verfügbar machen, so ist ein Logging zur Laufzeit unumgänglich.

Hier verhält es sich allerdings wie oben: Da keine Parametern zugewiesenen Werte erfasst werden, kann hierzu auch keine Relation erstellt werden.

Die Erfassung von **Rechenleistung und Arbeitsspeicher** wird bisher durch YW nicht unterstützt. Auch in indirekter Form kann aus den erfassten Informationen hierzu kein Rückschluss gezogen werden.

Mit der Erfassung von benötigter Rechenleistung und Arbeitsspeicher, was zwangsläufig zur Laufzeit erfolgen müsste, könnte eine Relation zwischen Eingaben und diesen Parametern erstellt und für Vorhersagen genutzt werden. Noch gewinnbringender wäre es zusätzlich die vom Nutzer gewählten Werte für Parameter zu erfassen und hinsichtlich dieser einen Zusammenhang abzubilden.

4.4.2 Erfassung von Eingabedaten, Parametern und Ausgabedaten

In diese Kategorie fällt die zusätzliche Erfassung von konkreten Daten und Eigenschaften von Daten.

Das Erfassen von **konkreten Daten** meint in dem hier betrachteten Kontext, die zur Laufzeit genutzten Werte zu kennen. Zum einen kann man in dieser Unterkategorie die bereits im letzten Abschnitt erwähnte Erfassung von gewählten Werten für Parameter ansiedeln. Bei Parametern, welche der Nutzer festlegt, kann dies bereits vor der Laufzeit erfolgen. Um die Qualität einer Vorhersage zu erhöhen kann jedoch auch der Ansatz verfolgt werden, alle Werte die Variablen zugewiesen werden, teilweise dann auch zur Laufzeit, zu erfassen.

Für YW wurde diesbezüglich bereits eine Erweiterung unter Nutzung von noWorkflow[1] erstellt. NoWorkflow ist ein System zur Erfassung von Provenance-Informationen zu Python Skripten mit maximal feiner Granularität. Das Vorgehen wird detailliert in [ZCW+17] vorgestellt.

In weiteren Arbeiten wäre zu evaluieren, ob anhand von erfassten Aufzeichnungen mithilfe einer Feature Extraction bestimmt werden kann, welche Werte für eine Vorhersage besonders gewinnbringend sind und welche keine Vorteile bringen und damit auch nicht erfasst werden müssen.

In beiden Fällen können mit diesen Informationen zahlreiche Prädiktionen unterstützt werden.

Das Erfassen von **Eigenschaften der Daten** stellt eine Abstraktion der oben beschriebenen Erfassung konkreter Daten dar. Es erhöht die Möglichkeit Informationen bereits vor der Laufzeit zu erfassen. Beispielsweise kann die Größe eines Arrays, welches immer vier Felder enthält vor der Laufzeit erfasst werden. Die konkret darin gespeicherten Werte aber nicht. Analog zu oben können mit dem Erfassen von Eigenschaften zahlreiche Prädiktionen und damit Optimierungen unterstützt werden.

4.5 Ablauf einer Optimierung durch die Nutzung von Prädiktionen

In diesem Abschnitt wird anhand von Abbildung 4.1 der Ablauf einer Optimierung eines Workflows, unterstützt durch Prädiktionen, erläutert.

Lediglich die Optimierungen, die ausschließlich anhand der bereits angebotenen prospektiven Informationen realisierbar wären, könnten ohne Lernverfahren und damit ohne mehrfache Ausführungen eines Workflows umgesetzt werden. Darunter fällt beispielsweise das Erkennen von Blöcken, Parametern und Eingaben, die zwar ausgeführt werden, aber keinen Einfluss auf das Endergebnis haben. Dies wurde bereits in Abschnitt 4.3.1 erläutert. Ein weiteres Beispiel hierfür ist in der, in Kapitel 3, vorgestellten Arbeit von S. Holl zu finden. Hier werden Optimierungen der Graphen, die vor der Ausführung als prospektive Provenance-Informationen bereitstehen, betrachtet. Optimierungen anhand der prospektiven Provenance-Informationen sind jedoch nicht Gegenstand dieser Arbeit.

Das im Folgenden beschriebene Vorgehen ist im Bereich des maschinellen Lernens von großer Relevanz und wird auf die hier vorliegende Problemstellung übertragen.

[1] https://github.com/gems-uff/noWorkflow

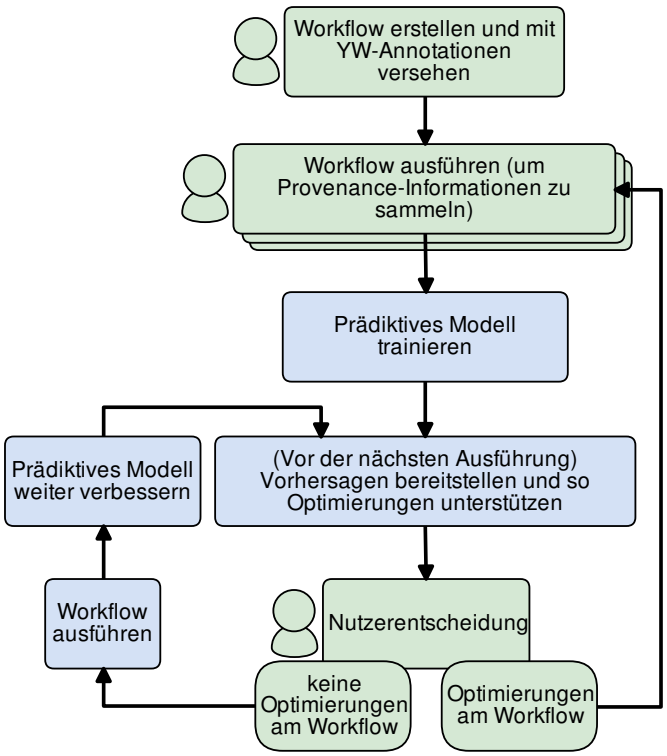


Abbildung 4.1: Ablaufkonzept der angestrebten YW-Erweiterung. In grün sind Schritte dargestellt, die eine Nutzerinteraktion erfordern. Rechenschritte von YW sind blau dargestellt.

In Abbildung 4.1 ist der Optimierungsprozess eines Workflows, der in beliebiger Sprache erstellt wurde, dargestellt. Der gesamte Ablauf ist in YW zu integrieren. Schritte, die eine Nutzerinteraktion benötigen, sind grün markiert und zusätzlich mit einem Symbol für den Nutzer gekennzeichnet. Rechenschritte von YW sind blau dargestellt.

Zu Beginn, in der Abbildung ganz oben zu sehen, erstellt der Nutzer das Skript in beliebiger Sprache und versieht es mit YW-Annotationen, wodurch eine Erfassung der Provenance-Informationen ermöglicht wird.

Der folgende Schritt stellt das Ausführen des Workflows dar, um Provenance-Informationen, die eine Ausführung benötigen, zu erfassen. Die erfassten Provenance-Informationen werden als Trainingsdaten für ein prädiktives Modell benötigt. Dieser Schritt ist notwendig, da, wie oben erläutert wurde, im Rahmen dieser Arbeit keine Optimierungen betrachtet werden, die lediglich prospektive Provenance-Informationen nutzen. Der in der Darstellung verwendete Kasten für die Ausführung des Workflows ist mehrfach hintereinander dargestellt, da für ein Lernverfahren möglichst viele Trainingsdaten, also erfasste Informationen aus Ausführungen, notwendig sind. Diese Ausführungen müssen nicht vom selben Nutzer stammen, sondern können ebenso, wenn das Skript global verfügbar ist, von unterschiedlichen Anwendern stammen. Nach den wiederholten Ausführungen des Workflows können die so erstellten Daten als Trainingsdaten für ein prädiktives Modell verwendet werden. Möchte man zusätzlich die Qualität des Modells bewerten, können die Daten auch in Test- und Trainingsdaten aufgeteilt werden und so dem Nutzer ein Feedback über

die zu erwartende Qualität der Vorhersage angezeigt werden. Trainiert man statt einem mehrere Modelle kann über Testdaten die Qualität einer Vorhersage für die verfügbare Trainingsdatenmenge bestimmt werden und damit das beste Modell für die Vorhersage genutzt werden.

Das Training eines solchen Modells stellt den nächsten Schritt des hier dargestellten Ablaufs dar (an dieser Stelle könnten parallel auch mehrere Modelle trainiert und das beste ausgewählt werden). Für diesen Schritt können verschiedene Verfahren gewählt werden. In Abschnitt 4.5 wird dies näher erläutert.

Mithilfe des so verfügbaren Modells können nun, vor einer erneuten Ausführung des Workflows, Vorhersagen berechnet und der Nutzer so auf potenzielle Probleme und Optimierungen hingewiesen werden.

Der folgende Schritt besteht darin, dass der Nutzer entscheidet, ob er den Workflow mit den gewählten Parametern und Eingabedaten und ohne Optimierungen am Quelltext ausführt. Diese wäre beispielsweise der Fall, wenn die Menge an voraussichtlich erstellten Daten dem Wissenschaftler für die Bearbeitung seiner Forschungsfrage nützt.

Dieser Fall ist in der Abbildung unten links zu verfolgen. Der Nutzer entscheidet sich gegen Optimierungen am Workflow, führt ihn mit den gewählten Parametern und Daten aus und erstellt somit zusätzliche Datensätze die für das bestehende Modell, zur weiteren Verbesserung, verwendet werden können. Dieses verbesserte Modell kann wiederum vor einer erneuten Ausführung für Vorhersagen verwendet werden. Im Rahmen dieser Variante ist es auch möglich, dass der Nutzer durch die Vorhersage auf ungünstig gewählte Parameter oder Eingabedaten hinsichtlich seiner Zielsetzung aufmerksam wurde. Korrigiert er diese Auswahl, nimmt er damit ebenfalls keine Optimierungen am Workflow vor. Es kann also auch in diesem Fall das bestehende Modell weiter verbessert werden.

Die unten rechts dargestellte Alternative besteht darin, dass der Nutzer Verbesserungen am Workflow, basierend auf den Erkenntnissen aus der Vorhersage, vornimmt oder falls möglich automatisch vornehmen lässt. Dies führt dazu, dass das bereits erstellte Modell für eine nächste Ausführung nicht mehr verwendet werden kann. Eventuell könnte an dieser Stelle durch Ansätze wie das sogenannte „Transfer Learning“ eine Verbesserung erzielt werden. Hierbei werden die erlernten Erkenntnisse aus früheren Versionen des Workflows verwendet, um Vorhersagen für die aktuelle Version zu erstellen. Im Rahmen dieser Arbeit wird dies allerdings nicht weiter erläutert, da hier zunächst das grundlegende Vorgehen betrachtet wird.

Der Ablauf muss dann ab dem zweiten Schritt erneut durchlaufen werden, was durch den Pfeil am rechten Rand der Abbildung verbildlicht ist. Es müssen also erneut mehrere Durchläufe zur Datensammlung und die anschließende Erstellung eines Modells stattfinden, bevor eine Vorhersage möglich ist.

Die notwendige Anzahl an durchgeführten Durchläufen um eine ausreichende Menge an Trainingsdaten zu sammeln, ist sowohl Gegenstand des Umsetzungs- als auch des Evaluationsteils dieser Arbeit. Hier wird die Problemstellung, dass der Lernalgorithmus eine möglichst große Menge an Trainingsdatensätzen, die nur durch zahlreiche Ausführung des Skripts erfasst werden können, benötigt, weiter betrachtet. Ein Nutzer hat kein Interesse an zahlreichen Ausführungen des Workflows, mit unterschiedlichen Daten und Parametern, die mit einem Zeitaufwand verbunden sind und nur für den Lernalgorithmus verwendet werden. Im Rahmen dieser Ausführungen würde ein Wissenschaftler eventuell bereits alle notwendigen Daten, die zur Klärung seiner Forschungsfrage

notwendig sind, generieren. Er hätte dann kein Interesse mehr an Vorhersagen oder Optimierungen. Erfasst man die Provenance-Informationen allerdings nicht lokal, könnten andere Wissenschaftler von den „nebenher“ gesammelten Informationen profitieren und ihre Ausführungen optimieren.

Nutzbare Lernverfahren

Grundsätzlich gibt es im Bereich des maschinellen Lernens eine Vielzahl an Verfahren unterschiedlicher Komplexität und mit unterschiedlichen Vor- und Nachteilen, die für eine Prädiktion genutzt werden können. Ein kurzer Überblick ist in Abschnitt 2.4 zu finden. Auch eine Kombination verschiedener Verfahren, also ein Ensemble Learning, könnte von Vorteil sein.

Im Rahmen dieser Arbeit wird beispielhaft die Vorhersage der von einen Workflow erstellten Dateien betrachtet. Hierfür wird die Auswahl der Verfahren auf eine lineare und polynomielle Regression unterschiedlichen Grades beschränkt. Regressionsverfahren sind generell gut geeignet um numerische Eingaben zu verwenden und mit diesen Modelle zu trainieren um wiederum Zahlenwerte vorherzusagen. Die Verfahren sind gut nachvollziehbar, was besonders für die folgende Evaluation von großem Nutzen ist. Nachteilig ist, dass komplexere Verfahren möglicherweise bessere Ergebnisse erzielen. Dies zu Evaluieren würde allerdings den Rahmen dieser Arbeit übersteigen.

4.6 Einbindung in YesWorkflow

Aus den Abschnitten 4.3 und 4.4 geht hervor, dass eine Erfassung von Eingabedaten, Parametern und Ausgabedaten, über die bisher verfügbaren Provenance-Informationen hinaus, den größten Nutzen hinsichtlich von Prädiktionen und damit angestrebten Optimierungen erwarten lässt.

Daher wird eine Erweiterung von YW angestrebt, mit der sich entweder Metriken von zugewiesenen Werten oder die zugewiesen Werte selbst erfassen lassen. Diese sollen dann als Trainingsdaten für prädiktive Modelle genutzt werden können und so Vorhersagen und daraus folgende Optimierungen erlauben. Die Vorhersagen sollen über eine Erweiterung der Befehlssyntax angefragt werden können.

Da es einer der Grundsätze von YW ist keine Informationen zur Laufzeit zu erfassen, wird der Fokus auf Eigenschaften von Eingabedateien und vom Nutzer gesetzten Parametern gerichtet.

4.6.1 Erweiterung der verfügbaren Kommentare

In Abschnitt 2.3.2 werden die verfügbaren YW-Kommentare, welche für das Erfassen der Provenance-Informationen genutzt werden, vorgestellt. Da im Rahmen dieser Arbeit zusätzliche Provenance-Informationen erfasst werden sollen, ist eine Erweiterung der YW-Kommentare notwendig. Im Rahmen weiterer Arbeiten ist eine zusätzliche Anpassung der bestehenden Visualisierung anzustreben.

@LOC-Annotation

Als erster Schritt muss angegeben werden, an welcher Stelle die zu erfassenden Informationen gespeichert werden sollen. Dazu wird die @LOC-Annotation eingeführt. Auf die @LOC-Annotation muss stets die Angabe eines Speicherorts mithilfe einer @URI-Annotation erfolgen. Für jede Version des Workflows muss diese Angabe eindeutig sein, um die erfassten Daten einem Stand des Skripts zuordnen zu können. Für jede Ausführung des Workflows wird dann an dem definierten Ort ein Datensatz oder mehrere Datensätze an Informationen geschrieben. In einem größeren Kontext, mit mehreren Nutzern, sollten die Daten an einem für alle Nutzer zugänglichen Ort abgelegt werden und die ACID-Eigenschaften sichergestellt werden. Zum aktuellen Stand ist die Konvention, dass die @LOC-Annotation genau ein mal pro Skript an beliebiger Stelle verwendet wird. An dieser Stelle ist aber auch auf einen größeren Kontext zu verweisen, in welchem mehrere Speicherorte von Vorteil sein können, oder eine zentrale Datenbank, in der alle Informationen gesammelt werden.

Daher wurde bewusst eine zusätzliche Annotation eingeführt, da in einem größeren Kontext hier ein Speicherort anzugeben ist auf den alle Nutzer zugreifen können, um eine möglichst große Menge an Trainingsdaten zu erfassen.

Im Rahmen des in Kapitel 5 vorgestellten Prototypen wird hier der Speicherort und Dateiname einer comma-separated values (CSV)-Datei angegeben. Es wird pro Version des Workflows eine Datei angelegt. Die Kopfzeile der CSV-Datei enthält die Variablenbezeichnungen aus den unten beschriebenen @DATA-Annotationen, in der Reihenfolge wie sie beim Skriptdurchlauf auftreten. Für jede Ausführung des Workflows werden mehrere Zeilen mit Informationen geschrieben.

@DATA-Annotation

Als zweiter Schritt muss spezifiziert werden, welche Informationen an der durch @LOC definierten Position erfasst werden sollen. Dazu wird die @DATA-Annotation eingeführt. Diese Annotation muss sich stets auf eine bestehende @IN-,@OUT- oder @PARAM-Annotation beziehen. Dies bedeutet konkret, dass auf @DATA stets eine Variablenbezeichnung folgen muss, die in den YW-Annotationen desselben Blocks vorkommt. Die Variablenbezeichnung wird dann dazu genutzt, um die erfassten Werte an der korrekten Postion innerhalb des in @LOC definierten Speicherorts abzulegen. In einem größeren Kontext würde der Variablenname den Key für das Feld in einer Datenbank darstellten. Und mithilfe weiterer Lernverfahren könnten beliebige Datentypen, daher auch @DATA, erfasst und vorhergesagt werden.

Eine einfachere Schreibweise wäre es eine bestehenden @IN-, @OUT- oder @PARAM-Annotation durch eine @DATA-Annotation zu ergänzen. Um die Übersichtlichkeit und Einfachheit von YW zu wahren, wurde allerdings für die ausführliche Schreibweise entschieden.

In dem, im Rahmen dieser Arbeit umgesetzten Prototypen, wird mit der Variablenbezeichnungen die Kopfzeile der CSV-Datei geschrieben. Das Schreiben der entsprechenden Informationen wurde im Rahmen des Prototyps manuell umgesetzt, da die dadurch entstehende Laufzeiterhöhung keinen Einfluss auf das Ergebnis hat. Zusätzlich wurden manuell weitere Informationen zur Laufzeit erfasst, um sie für die spätere Evaluation nutzen zu können.

In weiteren Arbeiten ist die Frage zu klären wie sich das Schreiben der so definierten Informationen automatisieren lässt. Zudem sollte die Annotation so eingeschränkt werden, dass nur Werte deklariert werden können, die nicht zur Laufzeit erfasst werden müssen.

4.6.2 Erweiterung der verfügbaren Befehle

In Abschnitt 2.3.5 sind die bisher nutzbaren YW-Befehle aufgelistet. Um die angestrebte Erweiterung einzubinden sind an dieser Stelle zwei neue Befehle notwendig.

TRAIN-Befehl

Der TRAIN-Befehl wird genutzt um anzugeben was die Eingabewerte und was die Ausgabe einer Vorhersage sein soll. Bei der Ausführung des Befehls werden dann entsprechende Modelle trainiert. Wo sich die Trainingsdaten dazu finden kann über die @LOC-Annotation des Skripts bestimmt werden. Hierbei ist es eine Voraussetzung, dass eine ausreichende Menge an Durchläufen des Skripts bereits erfasst wurde. Ist die Anzahl nicht ausreichend hoch, so ist mit einer schlechten Qualität der Vorhersagen zu rechnen. Der explizite Aufruf des Trainings führt zu einer Verbesserung der Laufzeit bei der Verwendung des im Folgenden beschriebenen PREDICT-Befehls.

PREDICT-Befehl

Der PREDICT-Befehl kann genutzt werden, um mit Variablen, denen bereits vor der Ausführung Werte zugewiesen wurden und die mit @DATA annotiert wurden, Vorhersagen für eine andere durch @DATA annotierte Variable zu erstellen. Dazu müssen die entsprechenden Modelle bereits durch den TRAIN-Befehl gelernt worden sein, sodass lediglich der angefragte Wert von dem Modell zurückgegeben wird. In einer Erweiterung könnten hierbei Methoden des Ensemble Learnings genutzt werden um die Qualität des bestimmten Wertes zu erhöhen.

Da im Rahmen des umgesetzten Prototypen keine hohe Laufzeit für das Training der Modelle zu erwarten ist und der Fokus der Arbeit auch nicht auf dieser Erweiterung liegt, wird der TRAIN-Befehl im Rahmen des Prototypen nicht umgesetzt. Durch Ausführen des PREDICT-Befehls, bei welchen die Eingaben explizit angegeben werden müssen, werden die Modelle auch trainiert und der angefragte Wert zurückgegeben. Dies ist detaillierter in Kapitel 5 beschrieben.

Mit dem PREDICT-Befehl kann grundsätzlich eine Vorhersage in zwei Richtungen erfolgen: Zum einen können damit Ausgaben des Workflows oder Metriken anhand der Eingaben und Parameter erfolgen. Zum anderen können damit Eingaben und Parameter, oder Metriken davon, anhand der gewünschten Ausgaben vorhergesagt werden.

4.6.3 Vorschlag für eine Visualisierung der Erweiterung

Für die hier vorgeschlagenen Erweiterungen des bestehenden YW-Prototypen können auch in der graphischen Darstellung Erweiterungen vorgenommen werden. Dies bezieht sich zunächst auf die Graphviz-Darstellungen, in welcher mit @DATA annotierte Werte farbig hervorgehoben werden könnten. Zusätzlich könnte bei diesen Knoten dann angegeben werden, wo die Informationen erfasst werden, also die @LOC-Annotation.

Im Rahmen einer Erweiterung, die weit über diese Arbeit hinausgeht, wäre dann auch eine graphische Oberfläche erstrebenswert, in welcher man beispielsweise anhand einer Darstellung des Graphen Werte für mit @DATA annotierte Variablen auswählen kann. In einer solchen Darstellung könnten dann die Vorhersagen angezeigt werden und auch eine Ausführung des Workflows mit den gewählten Eingaben und Parametern gestartet werden.

4.6.4 Problemstellung Trainingsdatenmenge

In den Abschnitten 4.6.1 und 4.6.2 wird erläutert wie zusätzliche Provenance-Informationen erfasst und durch einen Aufruf innerhalb von YW für eine Vorhersage genutzt werden können. Es ist offensichtlich, dass eine solche Erweiterung für einen Nutzer gewinnbringend ist. Allerdings wird vorausgesetzt das eine hinreichende Menge an Trainingsdaten aus durchgeführten Ausführungen des Workflows erfasst wurde. Wie auch bereits in Abschnitt 4.5 beschrieben, ist diese Menge der entscheidende Punkt für die Qualität und damit für den Nutzen eines damit erstellten Modells. Um abschätzen zu können ab welcher Menge an verfügbaren Trainingsdaten eine sinnvolle Vorhersage möglich ist, wurden hierzu Skripte erstellt. Diese werden in Kapitel 5 vorgestellt. In Kapitel 6 wird dann anhand der Skripte die notwendige Menge an Trainingsdaten abgeschätzt.

5 Umsetzung

Im Rahmen dieser Arbeit wurde prototypisch eine Erweiterung für YW erstellt, mit der zusätzliche Provenance-Informationen erfasst werden können. Ebenso wurde YW dahingehend erweitert, dass basierend auf erfassten Provenance-Informationen Voraussagen für beliebige, in früheren Durchläufen erfasste, Werte erstellt werden können. Mithilfe dieser Prädiktionen wird der Nutzer bei Optimierungen unterstützt. Dazu werden eine lineare Regression, sowie polynomielle Regressionen unterschiedlichen Grades angeboten. Die Umsetzung wurde so generisch gestaltet, dass die Einbindung weiterer Lernverfahren unterstützt wird.

Des Weiteren wurden zwei Skripte erstellt, mit deren Hilfe evaluiert wird, wie viele Datensätze als Trainingsdaten verfügbar sein müssen, um eine Prädiktion von guter Qualität zu ermöglichen. Dazu wurden zwei unterschiedliche Vorgehensweisen betrachtet.

In diesem Kapitel wird zunächst die Umsetzung der YW-Erweiterung vorgestellt. Danach werden die zur Evaluation verwendeten Skripte vorgestellt. Der Programmcode ist unter https://github.com/Constanze-S/yw-prototypes zu finden.

5.1 Umsetzung der YesWorkflow-Erweiterung

Da die von YW bisher erfassten Provenance-Informationen für die angestrebten Vorhersagen nicht ausreichen, wurde die bestehende Annotationssyntax sowie die verfügbaren Befehle erweitert.

5.1.1 Erweiterung der Annotationssyntax

Die bisher in YW verfügbaren Annotationen unterstützen keine Erfassung von Werten oder Metriken von Werten, die einer Variablen oder einem Parameter zur Laufzeit zugewiesen werden. Lediglich Informationen zum Endstand von erstellten Dateien sind nach einer Ausführung durch die Recon-Erweiterung verfügbar. Aus diesem Grund werden im Folgenden die beiden Annotationen @LOC und @DATA eingeführt:

@LOC-Annotation

Um die zugewiesenen Werte von Variablen zu erfassen, wurde zunächst die @LOC-Annotation eingeführt. Diese Annotation gibt den Speicherort an, an welchem die Informationen, die als Trainingsdaten verfügbar sein sollen, gespeichert werden. In Abschnitt 4.6.1 ist eine konzeptionelle Beschreibung der Annotation zu finden.

Im Rahmen des Prototypen erwartet YW die Werte in einer CSV-Datei. Hierbei wird sowohl ein absoluter als auch relativer Pfad unterstützt. Bei relativen Pfaden ist zu beachten, dass diese bei der Ausführung von YW relativ zum Arbeitsverzeichnis von YW zu sehen sind. Der eigentliche Pfad wird über eine @URI-Annotation angegeben. Die Kombination aus @LOC- und @URI-Annotation kann in dem mit YW-Annotationen versehenen Skript an beliebiger Stelle hinzugefügt werden. Eine vollständige Annotation in einem Python-Skript sieht beispielsweise folgendermaßen aus:
@LOC Trainingsdaten @URI file:c:/Beispiel/Beispiel_Trainingsdaten.csv

@DATA-Annotation

Analog zu der @LOC-Annotation wird auch die @DATA-Annotation in Abschnitt 4.6.1 konzeptionell beschrieben. Die neu eingeführte @DATA-Annotation gibt an, welche Werte in der CSV-Datei erfasst werden sollen.

Dazu muss auf eine @DATA-Annotation ein Variablenname folgen, der auch im selben Block durch eine @PARAM-, @IN- oder eine @OUT-Annotation deklariert wurde. Im Rahmen des Prototypen können an dieser Stelle beliebige Werte erfasst werden.

Betrachtet man allerdings ein reales Szenario, so dürfen nur Werte erlaubt werden, die vor dem Beginn einer Ausführung feststehen oder die nach ihrer Beendigung bestimmt werden können. Diese sind beispielsweise ausgewählte Eingabedateien, manuell gewählte Parameter oder die Anzahl an erstellten Dateien. Die @DATA-Annotation ist nur dann sinnvoll, wenn im selben Skript auch eine @LOC-Annotation verwendet wird, da diese den Speicherort für die durch die @DATA-Annotation deklarierten Werte angibt.

Der aktuelle Stand des Prototypen sieht vor, dass der Nutzer das Schreiben der Werte in die oben genannte CSV-Datei selbst in dem mit YW-Annotationen versehenden Skript implementiert. Langfristig ist anzustreben, dass der Prototyp dahingehend erweitert wird, dass nur Werte erfasst werden können, die vor oder nach der Ausführung feststehen und dass diese anhand der Annotation automatisch erfasst werden können. Dies würde jedoch den Umfang dieser Arbeit übersteigen. Werte, die zum Endzeitpunkt feststehen, könnten dann beispielsweise über die Recon-Erweiterung erfasst werden. Im Rahmen des Prototyps muss auch für diese Werte das Schreiben in dem mit YW-Annotationen versehenden Skript explizit implementiert sein.

Beide Annotationen, also sowohl die @LOC- als auch die @DATA-Annotation, müssen, analog zu den bestehenden YW-Kommentaren, innerhalb eines Kommentars der jeweiligen Programmiersprache erfolgen.

Durch die manuelle Anpassung der Workflow-Skripte durch den Nutzer können die für eine Vorhersage notwendigen Trainingsdaten erfasst werden. Des Weiteren können mithilfe dieser Annotationen die relevanten Daten für die Vorhersage durch YW automatisiert ausgelesen und Eingabedaten plausibilisiert werden, was in den folgenden Abschnitten näher erläutert wird.

Zu beachten ist, dass im Rahmen des Prototyps bei der Ausführung des beispielhaft verwendeten annotierten Workflows noch weitere Werte erfasst werden, da diese für die spätere Evaluation benötigt werden.

5.1.2 Erweiterung der YesWorkflow-Befehle

Im letzten Abschnitt ist beschrieben, wie die bestehenden Annotationen so erweitert wurden, dass Trainingsdaten erfasst werden können. Damit ein Nutzer diese Informationen im Rahmen von YW für eine Vorhersage verwenden kann, wird im Rahmen des Prototyps ein neuer YW-Befehl definiert.

PREDICT-Befehl

Der PREDICT-Befehl, der konzeptionell in Abschnitt 4.6.2 beschrieben wird, kann dazu genutzt werden einen gewünschten Wert durch eine lineare oder eine polynomielle Regression dritten, vierten oder fünften Grades vorhersagen zu lassen. Da hierbei kurze Laufzeiten zu erwarten sind, wurde im Rahmen des Prototyps kein expliziter TRAIN-Befehl, wie konzeptionell beschrieben, umgesetzt. Die eigentliche Regression wird mithilfe eines Python-Skripts ausgeführt, welches in Kapitel 5.1.3 genauer beschrieben wird. Der PREDICT-Befehl kann, analog zu den bereits in YW verfügbaren Befehlen, die in Abschnitt 2.3.5 erläutert werden, auf die folgende Art und Weise aufgerufen werden:

yw PREDICT [source file(s)] [-c <name=value>] ...

Hierbei ist als „source file“ das mit YW-Annotationen versehene Skript anzugeben. Die Übergabe von mehreren Skripten auf einmal, wird zum aktuellen Zeitpunkt beim PREDICT-Befehl nicht unterstützt.

Als Konfigurationsparameter ([-c <name=value>]) stehen die folgenden Optionen bereit:

- **predict.input** Namen der Variablen (die als Features genutzt werden sollen) und für sie ausgewählte Werte, mit denen eine Vorhersage erstellt werden soll. Um eine Vorhersage zu erhalten, müssen den Variablen Werte zugewiesen werden. Die Übergabe der Variablen bzw. der für sie gewählten Werte geschieht deshalb als Key/-Value-Paare auf folgende Art: (Variable1,Wert1);(Variable2;Wert2) . . .
- **predict.ouput** Name der Variablen bzw. des Wertes, der vorhergesagt werden soll.
- **predict.model** Hier kann der Nutzer zwischen verschiedenen Lernverfahren wählen. Aktuell werden folgende Verfahren zum Training des Modells unterstützt: lineare Regression (predict.model=linear), polynomielle Regression Grad 3 (predict.model=poly3), polynomielle Regression Grad 4 (predict.model=poly4) und polynomielle Regression Grad 5 (predict.model=poly5).
- **predict.verbose** Mithilfe dieses Konfigurationsparameters kann ein erweitertes Logging an- oder ausgeschaltet werden.
- **predict.python** Für die Ausführung der Regressionen mithilfe eines Python-Skripts wird eine Python-Installation benötigt. Sollte Python innerhalb des Systems nicht bekannt sein, kann über diesen Konfigurationsparameter der vollständige Pfad zu einer Python-Installation angegeben werden.

Exemplarischer Aufruf

predict C:/Example/example.py -c **predict.input**=(number_of_lines_in_sample_spreadsheet,20); (sample_score_cutoff,45) -c **predict.output**=number_of_raw_image_files -c **predict.python**= C:/Example/Python -c **predict.model**=linear -c **predict.verbose**=on

Dieser Aufruf nutzt die Datei C:/Example/example.py als Trainingsdaten, um ein lineares Regressionsmodell (predict.model=linear) zu trainieren.
Das Modell kann anhand von Werten für number_of_lines_in_sample_spreadsheet und sample_score_cutoff einen Wert für number_of_raw_image_files vorhersagen.
Es wird der Wert für number_of_raw_image_files zurückgegeben, den das Modell für number_of_lines_in_sample_spreadsheet = 20 und sample_score_cutoff = 45 berechnet.
Zur Demonstration wurde hier noch der Pfad für die Python-Installation (predict.python=C:/Example/Python) angegeben.
Das erweiterte Logging ist aktiviert (predict.verbose=on).

Anpassung von YesWorkflowCLI

Für die Erweiterung von YW um den PREDICT-Befehl wurde die Klasse YesWorkflowCLI angepasst. Die folgenden beiden Schritte werden ausgeführt, wenn der PREDICT-Befehl erkannt wird:

1. Extract Mithilfe des EXTRACT-Befehls werden alle Kommentare aus dem mit YW-Annotationen versehenen Skript extrahiert und weiterverarbeitet. Die Klasse DefaultExtractor wurde dahingehend angepasst, dass nun auch @LOC und @DATA-Annotationen berücksichtigt werden. Im Paket org.yesWorkflow.annotations gibt es dazu die Klassen Log und Data. Wie bei den bisherigen Annotationen auch, werden bei der Verarbeitung durch den DefaultExtractor Informationen über die Annotation in der verwendeten In-Memory-Datenbank gespeichert. Dazu zählen unter anderen Tag (z.B. LOC) und Value (z.B. Trainingsdaten). Bei einer @LOC-Annotation gibt es stets einen zugehörigen @URI Datensatz, der ebenfalls in der Datenbank gespeichert wird. Auch dieser enthält unter anderem Tag (z.B. URI) und Value (z.B. c:/Beispiel/Beispiel_Trainingsdaten.csv). Nach der Verarbeitung des mit den YW-Annotationen versehenen Skripts durch den DefaultExtractor sind Informationen über alle im Skript enthaltenen Annotationen in der Datenbank gespeichert.

2. Predict Anschließend erfolgt die eigentliche Vorhersage. Dazu wurde das Paket org.yesWorkflow.predict angelegt. In diesem Paket sind die folgenden Klassen bzw. Interfaces enthalten:

- DefaultPredict
- LoggingLevel
- Predict
- RegressionModel
- StreamConsumer

Im Folgenden wird nun das Zusammenspiel der einzelnen Klassen erläutert. Bei den Klassen LoggingLevel und RegressionModel handelt es sich um sogenannte Enums (Aufzählungstypen). Mithilfe dieser Enums werden die Optionen **predict.model** und **predict.verbose** verarbeitet und auf die verschiedenen Regressionsmodelle bzw. Logging-Level gemappt.

Die Methoden des Interfaces Predict sind in Listing 5.1 zu finden. Es gibt zwei Methoden mit dem Namen **configure**, aber mit unterschiedlichen Eingabeparametern. Für die eine Methode wird hier ein einzelner Konfigurationsparameter übergeben und für die andere werden mehrere Konfigurationsparameter übergeben. Mithilfe dieser Methoden können die vom Nutzer übergebenen Optionen verarbeitet werden. Die Methode **predict** führt die eigentliche Vorhersage aus. Die Methode **getPrediction** liefert den Wert der Vorhersage.

```
1  package org.yesworkflow.predict;
2
3  import java.util.Map;
4
5  import org.yesworkflow.YWStage;
6  import org.yesworkflow.config.Configurable;
7
8  public interface Predict extends YWStage, Configurable {
9          Predict configure(String key, Object value) throws Exception;
10
11         Predict configure(Map<String, Object> config) throws Exception;
12
13         Predict predict() throws Exception;
14
15         Double getPrediction();
16 }
```

Listing 5.1: Übersicht über das Interface Predict.

Die Klasse DefaultPredict implementiert das Interface Predict. Im Folgenden wird nun erläutert, wie die Vorhersage durchgeführt wird. Bei der Übernahme der Konfigurationsparameter des Nutzers wird geprüft, ob die Voraussetzungen (z.B. unterstütztes Modell für die Regression und korrektes Format des Parameters **predict.input**) erfüllt sind. Die Methode **predict** prüft zuerst mithilfe der Daten in der In-Memory-Datenbank, ob durch das mit YW-Annotationen versehene Skript alle Voraussetzungen erfüllt sind. Dazu zählen unter anderem:

- Die Konfigurationsparameter **predict.input** und **predict.output** müssen übergeben worden sein.
- Es wurde genau ein mit YW-Annotationen versehenes Skript übergeben.
- Es gibt genau eine @LOC-Annotation.
- Für jede, bei den Konfigurationsparametern **predict.input** und **predict.output**, angegebenen Input- oder Output-Variablen gibt es eine zugehörige @DATA-Annotation im übergebenen Skript.
- Jede @DATA-Annotation bezieht sich auf eine bestehende @IN-, @OUT- oder @PARAM-Annotation.
- Die CSV-Datei mit den Trainingsdaten ist vorhanden (Angabe entweder über einen absoluten Pfad oder relativ zum Arbeitsverzeichnis von YW).

Sind alle Voraussetzungen erfüllt, dann kann die eigentliche Vorhersage gestartet werden. Dazu wird die Klasse ProcessBuilder [1] verwendet. Mithilfe dieser Klasse können Prozesse auf der Kommandozeile gestartet und Ausgaben bzw. Fehlermeldungen gelesen werden. Im Rahmen des Prototyps wird hiermit das eigentliche Python-Skript gestartet, welches die Vorhersage berechnet. Dieses wird in Abschnitt 5.1.3 erläutert. Da bei den angebotenen Regressionen keine längere Laufzeit zu erwarten ist, ist die prototypische Umsetzung so implementiert, dass der Vorgang abgebrochen wird, wenn nach 60 Sekunden noch immer kein Ergebnis berechnet wurde. Dies fängt den Fall ab, dass bei der Erstellung der Modelle ein Fehler passiert und das Python-Skript z.B. in einer Endlosschleife hängt. Diese Zeitspanne kann in der Implementierung beliebig angepasst oder aufgehoben werden, wenn aufwendigere Verfahren genutzt werden sollen.

Mithilfe der Klasse StreamConsumer werden die Ausgaben des Python-Scripts für die Vorhersage gelesen und analysiert. Sobald das Muster **Result=** erkannt wird, wird der nachfolgende Zahlenwert extrahiert und als Vorhersage gespeichert. Zudem wird der berechnete Wert auf der Konsole bzw. Kommandozeile ausgegeben und dem Nutzer so sichtbar gemacht.

Für den im Rahmen dieser Arbeit beschriebenen Prototypen muss das Java Runtime Environment 1.8[2] oder höher verwendet werden. Des Weiteren wird der PREDICT-Befehl aktuell nur unter Windows unterstützt. Ein Anpassung an andere Betriebssysteme ist mit überschaubaren Aufwand möglich, da nur der Aufruf des Python-Skript auf der Kommandozeile an das jeweilige Betriebssystem angepasst werden muss.

5.1.3 Skript zum Training des Modells und zur Bestimmung des vorherzusagenden Werts

Wie in Abschnitt 5.1.2 beschrieben, wird mithilfe des PREDICT-Befehls bzw. der Methode **predict** der Klasse DefaultPredict ein Skript aufgerufen, mithilfe dessen ein Modell anhand der gespeicherten Trainingsdaten erstellt wird. Das Skript ist in Python 3.6.5[3] implementiert und nutzt zusätzlich die Bibliothek scikit-learn.

Das Skript erwartet die folgenden Übergabeparameter:

- Argument 1: Namen der Variablen und zugeordnete Werte, die als Features für die Vorhersage verwendet werden sollen (entspricht dem Konfigurationsparameter **predict.input** von YW)
- Argument 2: Name der Variablen bzw. des Wertes, der vorhergesagt werden soll (entspricht dem Konfigurationsparameter **predict.output** von YW)
- Argument 3: Absoluter Pfad zu den Trainingsdaten
- Argument 4: Modelltyp für die Regression (also lineare Regression oder polynomielle Regression Grad 3, 4 oder 5)

[1] https://docs.oracle.com/javase/8/docs/api/java/lang/ProcessBuilder.html.
[2] https://www.oracle.com/java/
[3] https://www.python.org/.

Zunächst wird ein Modell des entsprechenden Typs (Argument 4) trainiert. Dazu werden anhand von Argument 1 die Spalten der CSV-Datei bestimmt, die als Eingabewerte dienen (Features). Mithilfe von Argument 2 wird die Spalte mit den Zielwerten bestimmt (Targets). Mit dem so erstellten Modell wird dann eine Vorhersage für die Ausgabe, die aus den in Argument 1 übergebenen Werten entsteht, berechnet. Der berechnete Wert wird auf der Kommandozeile ausgegeben und kann so, wie in Abschnitt 4.2 beschrieben, für Optimierungen genutzt werden.

5.2 Umsetzung von zwei Skripten zur Abschätzung der notwendigen Trainingsdatenmenge

Um mithilfe von Prädiktionen die Optimierungen, welche in Kapitel 4 dieser Arbeit diskutiert werden, zu unterstützen oder durchzuführen, ist eine hinreichend große Menge an erfassten Informationen von Workflowausführungen, die als Trainingsdaten verwendet werden können, notwendig. Wenn es sich um einen häufig genutzten Workflow handelt und bereits eine ausreichenden Menge an Provenance-Informationen gesammelt wurde, so können diese Optimierungen direkt angeboten werden. Allerdings sind Optimierungen während des Erstellungsprozesses eines Workflows, wenn er noch nicht so häufig ausgeführt wurde, von besonders großem Interesse. An dieser Stelle ist die Frage zu erörtern, wie viele Durchläufe eines Workflows erfasst werden müssen, um die gesammelten Provenance-Informationen für Vorhersagen von ausreichender Qualität nutzen zu können. Um dies anhand eines Beispiels tiefgehender zu betrachten, wurde der bereits in Kapitel 2 verwendete Workflow gewählt.

In Abbildung 5.1 ist das Skript, das Daten zu Proteinkristallen sammelt, nochmals zu sehen. Mit Python und der Bibliothek scikit-learn wurden zwei Skripte erstellt, die es ermöglichen die Qualität einer Vorhersage in Abhängigkeit der vorhandenen Trainingsdatenmenge zu evaluieren. Beispielhaft wird im Rahmen der Evaluation eine Vorhersage der Anzahl der erstellten raw_image Dateien diskutiert. Für das Training der Modelle wird eine lineare und polynomielle Regressionen verwendet und die Qualität der Vorhersage anhand des NMAE, welcher in Abschnitt 2.4.4 erläutert wird, bewertet. Zielsetzung war es hierbei zu bestimmen, wie viele Trainingsdatensätze notwendig sind um eine Vorhersage von angemessener Qualität zu erstellen.

Um die Qualität einer Vorhersage in Abhängigkeit der vorhandenen Menge an Trainingsdaten einzustufen, müssen diese in einem ersten Schritt erstellt werden. Dazu wurde das Proteinkristall-Skript, um die in Abschnitt 5.1.1 erläuterte @LOC- und @DATA-Annotation ergänzt. Des Weiteren wurde das Skript so erweitert, dass die zu erfassenden Werte in eine CSV-Datei geschrieben werden. Anschließend wurde es 1000 mal mit variierenden Eingaben ausgeführt.

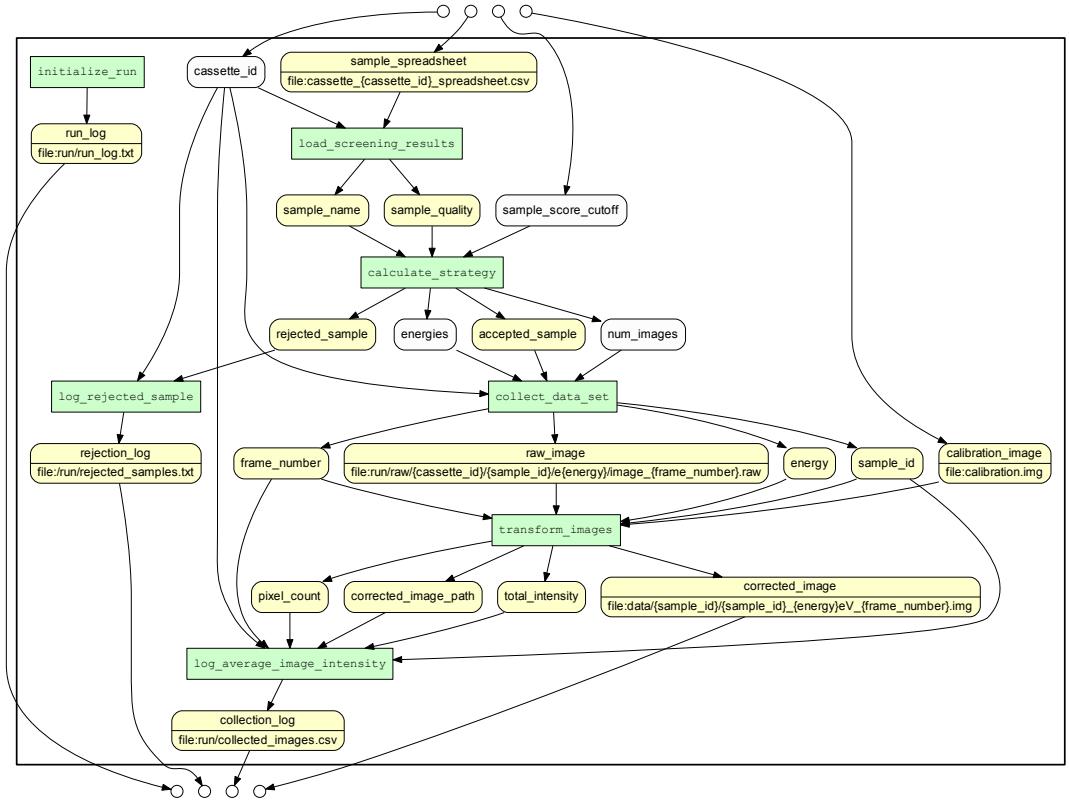


Abbildung 5.1: Kombinierte Darstellung der prospektiven Provenance-Informationen zu einem Python-Skript zur Datensammlung bei Proteinkristallen [MBBL15a].

5.2.1 Simulation von Nutzereingaben

Abbildung 5.1 zeigt den durch YW erstellten Graphen in der kombinierten Darstellung. Er zeigt graphisch die Abfolge der annotierten Blöcke in dem Skript zur Datensammlung bei Proteinkristallen und ist von oben nach unten zu lesen.

Als erster ausgeführter Block ist links oben **initialize_run** zu sehen. Aus dem Programmcode ist zu entnehmen, dass dieser Block lediglich ein Verzeichnis und eine Logdatei initialisiert. Er hat also keine Auswirkungen auf die Anzahl der erstellten Dateien zu Raw-Images und wird daher im Folgenden außer Acht gelassen.

Als nächster Block ist im Ablauf **load_screening_results** zu sehen. Für diesen Block müssen der Parameter **cassette_id** und die Datei **sample_spreadsheet** definiert sein.

Der Parameter **cassette_id** wird, wie beim Betrachten des Programmcodes offensichtlich wird, lediglich für die eindeutige Bezeichnung der zu verwendenden **sample_spreadsheet**-Datei benötigt. Dadurch ist er auch in der @URI-Annotation in dem Knoten von **sample_spreadsheet** zu finden. Da der Parameter keinen Einfluss auf die im folgenden betrachtete Datenmenge hat, wird **cassette_id**

lediglich aufsteigend nummeriert, um die **sample_spreadsheet-Dateien** eindeutig zu bezeichnen. Dies bedeutet konkret, dass der Parameter als q0, q1,...,q999 gesetzt wird, um 1000 Ausführungen zu simulieren.

Bei der durch **cassette_id** eindeutig spezifizierten Datei (**sample_spreadsheet**) handelt es sich um eine CSV-Datei, die zwei Spalten und beliebig viele Zeilen enthält. Die erste Spalte wird als **id** betitelt und enthält Werte der Form „DRT“ gefolgt von einer Zahl zwischen 100 und 999 (beispielsweise DRT110, DRT325). Die zweite Spalte trägt den Titel **score** und enthält Werte zwischen 1 und 100. Bei einer Ausführung kann diese Eingabedatei sowohl beliebige Werte enthalten, als auch von beliebiger Größe, d.h. von beliebiger Zeilenanzahl, sein.

Zu Beginn der Ausführung des Workflows wird mithilfe des Parameters **cassette_id**, der vom Nutzer gewählt werden muss, eine **sample_spreadsheet-Datei** für die Ausführung des ersten Blocks, **load_screening_results**, ausgewählt. Da die sample_spreadsheet-Datei beliebige Werte und beliebig viele Zeilen enthalten kann, wurden 1000 solcher Dateien mit einer zufällig gewählten Zeilenanzahl zwischen 1 und 25 Zeilen erstellt. Die in den 1000 Dateien enthaltenen Werte für **id** und **score** wurden ebenfalls zufällig, nach den oben genannten Kriterien, variiert. Benannt wurden die Dateien nach der angegebenen Namenskonvention **cassette_{cassette_id}_spreadsheet.csv**, wobei **cassette_id** aufsteigend als q0, q1, ..., q999 gewählt wurde.

Die so generierten 1000 **sample_spreadsheet-Dateien** wurden zusammen mit den bekannten Werten für **cassette_id** als Eingabe für somit 1000 Ausführungen des Workflows verwendet. Während der Durchläufe wurden relevante Provenance-Informationen, die im nächsten Abschnitt näher erläutert sind, in einer CSV-Datei erfasst und somit 1000 Ausführungen des Workflows, mit unterschiedlichen Eingabedateien, simuliert. Die so gesammelten Daten können für das Training, sowie das Testen, verschiedener Modelle genutzt werden.

5.2.2 Erfasste Provenance-Informationen

Im letzten Abschnitt ist erläutert, wie 1000 Eingabedateien mit zufälligen Werten für den Proteinkristall-Workflow generiert wurden. Der Workflow wurde mit jeder der Dateien ausgeführt und dabei die ausgewählten Provenance-Informationen in eine CSV-Datei geschrieben. Ein Ausschnitt aus der erstellten CSV-Datei ist in Abbildung 5.2 zu sehen. Eine Zeile der Datei entspricht einem Sample, das durch sample_name und die cassette_id spezifiziert wird. In dem Skript wurden dazu folgende numerische Parameter, die in Abbildung 5.3 hellblau markiert sind, mit der @DATA-Annotation versehen:

- Der Parameter **cassette_id** wird im Ablauf des Skripts für die eindeutige Zuordnung einer sample_spreadsheet-Datei, welche die Eingabedatei des Skripts darstellt, verwendet.
- Da die Eingabedatei, sample_spreadsheet, zeilenweise von dem Workflow bearbeitet wird, wird nicht ihr Wert, sondern die Anzahl der Zeilen, **number of lines in sample_spreadsheet**, erfasst. Dies wird zwar nicht durch die prototypische Umsetzung der @DATA-Annotation angeboten, lässt sich aber leicht umsetzen, da sich diese Metrik der Datei einfach erfassen lässt. Die Anzahl der Zeilen hat direkten Einfluss auf die Anzahl der erstellten raw_image-Dateien und ist daher von Relevanz.

- Jede der gelesenen Zeilen aus der geladenen sample_spreadsheet-Datei enthält einen **sample_name**, der für das Anlegen der Ordnerstruktur beim Erstellen der Dateien verwendet wird.
- Ebenso enthält jede der gelesenen Zeilen eine **sample_quality**, von der direkt abhängt, wie viele raw_image-Dateien erstellt werden.
- Der Wert für **sample_score_cutoff** entscheidet ebenfalls direkt, wie viele raw_image-Dateien erstellt werden.
- Abhängig von dem Größenverhältnis zwischen sample_quality und sample_score_cutoff kann es sein, dass für ein Sample keine Dateien erstellt werden. Ist dies der Fall, dann wird **rejected_sample** als 1.0 geloggt und accepted_sample als 0.0. Analog kann auch der Fall eintreten, dass rejected_sample als 0.0 und accepted_sample als 1.0 geloggt wird. Hier wird also eine Metrik und nicht der enthaltene Wert erfasst.
- Die Länge der Liste energies, **size of energies**, stellt einen Skalierungsfaktor für die Anzahl der erstellten raw_image-Dateien dar. Es wird die Länge der Liste erfasst und nicht ihr Inhalt, was ebenso eine Metrik ist.
- Ein weiterer erfasster Wert ist **num_images**, welcher, falls eine Sample bearbeitet wird, stets dessen sample_quality-Wert um zwei addiert ist. Falls für ein Sample keine Dateien erstellt werden, so wird dieser als 0.0 geloggt.
- Mit **number of raw_image-files** wird erfasst wieviele raw_image-Dateien erstellt wurden.

Ebenfalls aufgezeichnet wurde wie lange die Bearbeitung jeder Zeile einer sample_spreadsheet-Datei, also jedes bearbeitete Sample, in Anspruch genommen hat. Anhand der erfassten Informationen konnte dann noch bestimmt werden, wie viele raw_image-Dateien von jeder sample_spreadsheet-Datei erstellt wurden.

In Abbildung 5.2 ist ein Ausschnitt aus der so erstellten CSV-Datei zu sehen. Die so verfügbaren, gesammelten Provenance-Informationen simulieren 1000 Ausführungen des Workflows durch einen Nutzer. Um den Inhalt der Datei, was ohne Kopfzeile insgesamt 13147 Zeilen sind, als Trainings- beziehungsweise Testdaten für Vorhersagemodelle nutzen zu können, wurde nachträglich noch die Reihenfolge der Zeilen zufällig vertauscht.

cassette_id	number of lines in sample_spreadsheet	sample_name	sample_quality	sample_score_ cutoff	rejected_ sample	size of energies	accepted_ sample	num_images	number of raw_image-files	duration in s	number of raw_image-files (whole spreadsheet)
q0	13	DRT926	15	23	1	0	0	0	0	0,000816822	2633
q0	13	DRT704	100	23	0	4	1	102	408	4,546191454	2633
q0	13	DRT643	80	23	0	3	1	82	246	4,199771881	2633
q0	13	DRT136	76	23	0	3	1	78	234	4,21275115	2633
q0	13	DRT404	1	23	1	0	0	0	0	0,000499487	2633
q0	13	DRT303	10	23	1	0	0	0	0	0,00049901	2633
q0	13	DRT462	72	23	0	3	1	74	222	2,835956573	2633
q0	13	DRT615	70	23	0	3	1	72	216	2,525454521	2633
q0	13	DRT268	96	23	0	4	1	98	392	7,378679752	2633
q0	13	DRT420	36	23	0	1	1	38	38	0,246104956	2633
q0	13	DRT139	88	23	0	3	1	90	270	2,18300271	2633
q0	13	DRT193	71	23	0	3	1	73	219	2,729127884	2633
q0	13	DRT629	95	23	0	4	1	97	388	2,42261982	2633
q1	3	DRT719	21	19	0	1	1	23	23	0,18170929	2337
q1	3	DRT318	96	19	0	5	1	98	490	2,65674448	2337
...											
q999	24	DRT853	69	46	0	1	1	71	71	0,213658094	1606
q999	24	DRT827	46	46	0	1	1	48	48	0,161239624	1606
q999	24	DRT828	94	46	0	2	1	96	192	0,809202909	1606

Abbildung 5.2: Alle ermittelten Provenance-Informationen aus 1000 sample_spreadsheet-Dateien. Ausschnitt aus der erstellten CSV-Datei vor dem zufälligen Vertauschen der Zeilenabfolge.

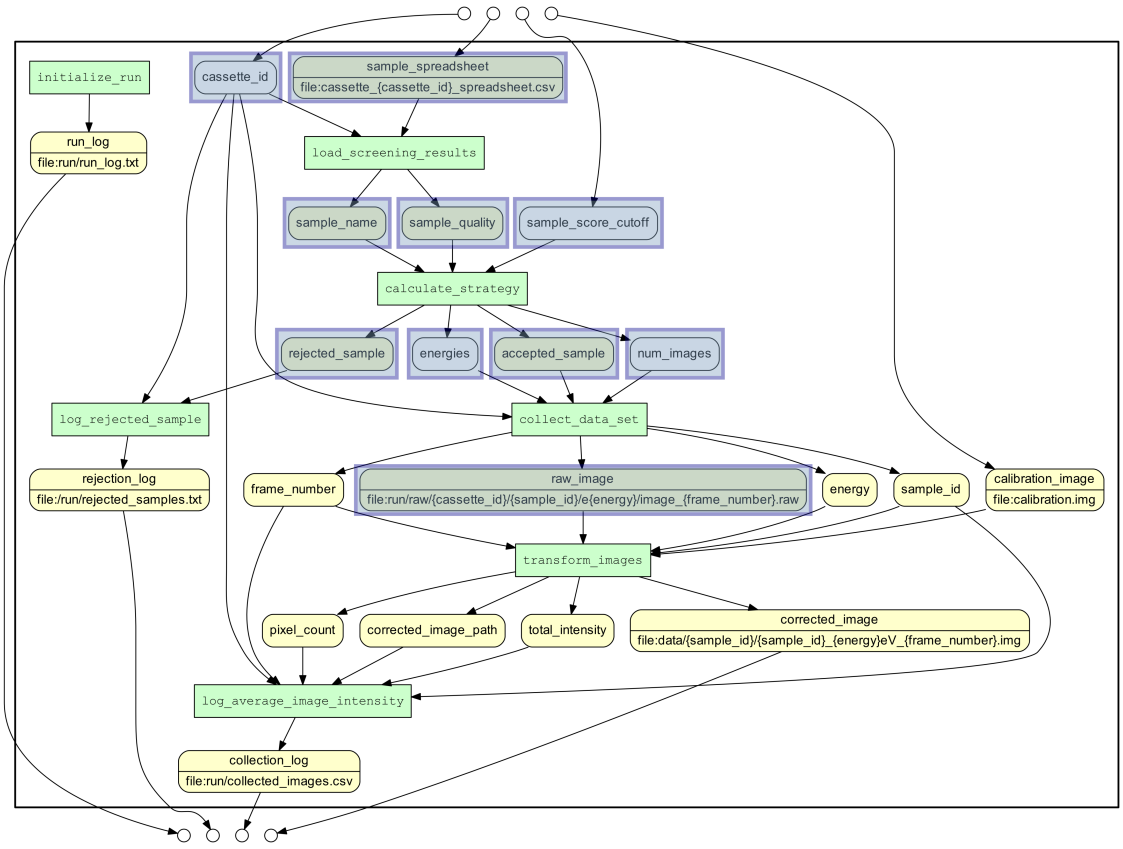


Abbildung 5.3: Erfasste Provenance-Informationen. Darstellung des Graphen aus [MBBL15a].

5.2.3 Prädiktion der Anzahl von raw_image-Dateien mittels zweier verschiedener Ansätze

Durch die in Abschnitt 5.2.1 beschriebene Simulation von 1000 Ausführungen des Workflows, bei denen die oben genannten Informationen erfasst wurden, können diese Daten nun für das Trainieren und die Evaluation verschiedener Modelle genutzt werden. Primär soll im Rahmen der Evaluation im folgenden Kapitel untersucht werden, wie viele Durchläufe des gewählten Workflows bereits erfasst sein müssen, um die Anzahl der entstehenden Dateien, also die Anzahl der **raw_image-Dateien**, vor einer weiteren Ausführung möglichst präzise anhand verschiedener Eingaben vorhersagen zu können.

Dazu bieten sich zwei grundsätzlich unterschiedliche Vorgehensweisen an, für die jeweils ein Skript zur Evaluation erstellt wurde.

Prädiktion durch ein Modell für beliebig viele Blöcke

Hierzu wurde in Python 3.6.5 ein Skript entwickelt, dass als Eingabe die CSV-Datei mit den Datensätzen liest. Diese werden sowohl zum Training, als auch zum Testen verschiedener Modelle genutzt. Der Nutzer kann aus der CSV-Datei wählen, welche Spalten als Features und welche Spalte als Target verwendet werden soll. Je nachdem, wie der Nutzer hier wählt, wird ein Modell erstellt, das eine Vorhersage über entsprechend viele Blöcke hinweg erstellt.

Ein Beispiel hierfür ist in Abbildung 5.4 zu sehen. Diese nimmt Bezug auf den Graphen aus Abbildung 5.3. Es werden die Werte für **sample_quality** und **sample_score_cutoff** genutzt, um eine Vorhersage für die Anzahl der erstellten **raw_image-Dateien** zu erstellen. Dabei umfasst das Modell die zwei Blöcke **calculate_strategy** und **collect_data_set**. Der hierfür gestaltete Prototyp kann jedoch in gleicher Weise auch für die Vorhersage anderer Werte genutzt werden

Für die Vorhersage wurden eine lineares Regressionsmodell sowie polynomielle Regressionsmodelle von Grad drei, vier und fünf umgesetzt. Dies wurde so generisch gestaltet, dass sich das Skript leicht um weitere Verfahren ergänzen lässt.

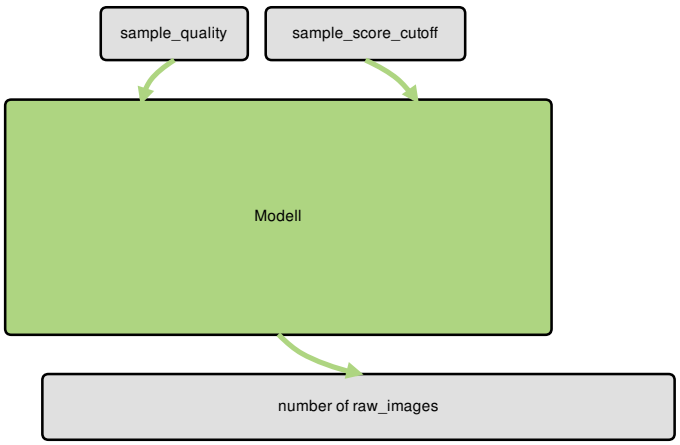


Abbildung 5.4: Vorhersage über zwei Blöcke mit einem Modell.

Ebenso kann durch den Nutzer bestimmt werden, in welchen Schritten der genutzte Teil der Trainingsdaten erhöht wird. Dies bedeutet konkret, dass zunächst beispielsweise nur 10 Datensätze, also nur die ersten 10 Zeilen aus der CSV-Datei, verwendet werden.

Anhand dieser Zeilen wird für die gewählte Kombination aus Features und einem Target zunächst ein lineares Regressionsmodell zur Vorhersage trainiert und getestet.

Da, um die Daten möglichst effizient zu nutzen, eine 5-fache Kreuzvalidierung eingebunden wurde, führt dies dazu, dass anhand von acht Datensätzen das Modell trainiert wird. Die beiden übrigen werden dann zum Testen verwendet. Im nächsten Schritt werden dann andere acht Datensätze zum Trainieren verwendet und wiederum mit den übrigen zwei getestet. Dies wird insgesamt fünf mal wiederholt, bis mit jedem der 10 Datensätze einmal getestet wurde. Für die Beurteilung der Qualität wird der NMAE für jedes Modells bestimmt. Die beim Testen erreichten Werte werden dann gemittelt, sodass für jede eingesetzte Datenmenge ein gemittelter Wert der bestimmten NMAE Werte bereitsteht.

Anhand der Abweichung zwischen den fünf vorliegenden Testwerten für jede betrachtete Datenmenge kann bestimmt werden, wie stark die Qualität der Modelle verglichen untereinander variiert.
Bei geringen Datenmengen schwanken die hier bestimmten Werte stark. Ist die für das Modell gewählte Trainingsdatenmenge zufällig repräsentativ für die zugehörige Testdatenmenge, kann ein passendes Modell trainiert werden. Der NMAE-Wert ist größer, was bedeutet dass das Modell besser gepasst hat. Wird im nächsten der fünf Durchläufe eine nicht repräsentative Trainingsdatenmenge verwendet, wird der NMAE für das Modell kleiner. Das Modell hat also Vorhersagen mit größerer Abweichung für die Testdatensätze erstellt. Da insgesamt fünf Modelle für jede eingesetzte Datenmenge erstellt und evaluiert werden, existieren jeweils fünf solche NMAE pro Datenmenge. Bei größeren Datenmengen nähern sich diese immer weiter an. Dies liegt daran, dass ein großer Testdatensatz meist repräsentativer ist als ein kleiner und somit „bessere“ Modelle von konstanterer Qualität trainiert werden können.

Die ersten 10 Datensätze werden nun nochmals verwendet, um mit dem gleichen Vorgehen ein polynomielles Regressionsmodell vom Grad drei, vier und fünf zu trainieren und zu testen.

Im nächsten Durchlauf wird die Menge der eingesetzten Daten, wie vom Nutzer angegeben, erhöht. Dies kann beispielsweise bedeuten, dass nun 20 Datensätze verwendet werden um die Modelle zu trainieren und zu evaluieren.

In einem letzten Durchlauf könnte der Nutzer beispielsweise auswählen, dass alle Datensätze verwendet werden. Dies bedeutet dann, dass je 80% Datensätze zum Trainieren verwendet werden und die übrigen 20% zum Testen der Modelle.

An dieser Stelle soll explizit darauf hingewiesen werden, dass eine Ausführung des Workflows die Bearbeitung einer **sample_spreadsheet-Datei** umfasst. Im Mittel enthalten diese Dateien 13,15 Samples, was bedeutet das bei einem Durchlauf des Workflows etwa 13 Samples bearbeitet werden. Je nach gewählten Eingabeparametern ist es sinnvoll die erstellte Anzahl an Dateien pro Sample oder pro **sample_spreadsheet-Datei** zu betrachten. Auf dies wird aber an den entsprechenden Stellen nochmal hingewiesen.

Das errechnete gemittelte Qualitätsmaß, NMAE, wird dann in Diagrammen in Abhängigkeit zu den eingesetzten Datensätze dargestellt. Zusätzlich können die Koeffizienten, welche die Regressionsmodelle erstellt haben, analysiert werden.

Prädiktion durch aneinandergereihte Modelle

Der zweite Ansatz wurde nur für eine ausgewählte Konfiguration betrachtet. Hierzu wurde wiederum ein Skript in Python erstellt, das dem obigen ähnlich ist.

Abbildung 5.5 zeigt die selbe Vorhersage wie Abbildung 5.4, jedoch basierend auf einem anderen Vorgehen.

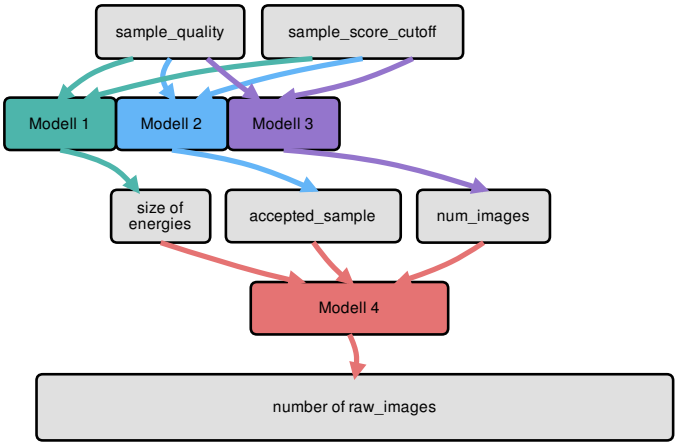


Abbildung 5.5: Vorhersage über zwei Blöcke mit vier Modellen. Die Vorhersagen für den ersten Block werden als Eingaben für den zweiten Block verwendet.

Für diese Herangehensweise werden die Features **sample_quality** und **sample_score_cutoff** verwendet, um mittels drei Modellen eine Vorhersage für **size_of_energies**, **accepted_sample** und **num_images** zu erstellen. Die hierfür vorhergesagten Werte werden wiederum als Eingabe für Modell 4 verwendet, welches eine Vorhersage für **number of raw_images** erstellt. Somit erstellt jedes Modell eine Vorhersage über einen Block.

Die Umsetzung ist im Wesentlichen analog zu dem in Abschnitt 5.2.3 vorgestellten Skript für ein Modell für beliebig viele Blöcke. Statt eines Modells werden hier aber vier Modelle eingesetzt. Der Nutzer kann in gleicher Weise, wie bei dem anderen Skript, bestimmen, wie viele Trainingsdaten verwendet werden sollen und der Fehler wird anhand des NMAE bestimmt. Um die Ergebnisse vergleichen zu können, wird auch hier eine 5-fache Kreuzvalidierung vorgenommen.

6 Evaluation

Durch die in Kapitel 5 erläuterte Erweiterung von YW ist es möglich anhand von zusätzlichen Provenance-Informationen beliebige numerische Parameter für einen Workflow vorhersagen zu lassen und damit Optimierungen verschiedener Art zu unterstützen. Dass die Nutzung von Vorhersagen in diesem Zusammenhang für einen Nutzer Vorteile bringt, geht bereits aus den aufgeführten Optimierungen in Kapitel 4 hervor. Allerdings sind diese nur von Vorteil, wenn die Qualität der Vorhersagen ausreichend ist. Hierfür ist, wie bereits in Abschnitt 4.5 beschrieben, eine ausreichende Menge an Trainingsdaten, die nur aus abgeschlossenen Ausführungen stammen können, notwendig. Die Fragestellung, wie viele Trainingsdaten verfügbar sein müssen, um eine Vorhersage von ausreichender Qualität anbieten zu können, ist also für diese Arbeit von zentraler Bedeutung.

In diesem Kapitel wird nun beispielhaft, anhand des Proteinkristall-Workflows, der in Abschnitt 2.3.6 vorgestellt wurde, mithilfe der in Kapitel 5 vorgestellten Skripte zur Evaluation die für eine Vorhersage notwendige Trainingsdatenmenge diskutiert.

6.1 Rahmenbedingungen

Bei dem in dieser Arbeit beispielhaft verwendeten Proteinkristall-Workflow, der als Ausgabe verschiedene Dateien erstellt, ist offensichtlich, dass sich die Anzahl der erstellten **raw_image-Dateien** nach dem **collect_data_set-Block** nicht weiter verändert. Aus diesem Grund wurde die Anzahl der erstellten Dateien auch an dieser Stelle des Workflows in der durch die @LOC-Annotation spezifizierten CSV-Datei, welche die Trainingsdaten enthält, erfasst.

In Abbildung 6.1 ist der beispielhaft verwendete Proteinkristall-Workflow nochmal zu sehen. Die Stelle, an welcher **raw_image-Dateien** erstellt werden, ist mit einer blauen Umrandung markiert. Ausgehend von dieser Stelle im Workflow können die Abhängigkeiten, die Einfluss auf die Erstellung haben, anhand der blauen Pfeile nach oben, das heißt entgegen der Ausführungsreihenfolge des Workflows, verfolgt werden. Bei den grün umrandeten Elementen, **cassette_id**, **sample_spreadsheet** und **sample_score_cutoff**, handelt es sich um globale Eingaben für den Workflow. Diese stehen bereits vor dem Beginn der Ausführung fest und sind daher außerhalb des Kastens in der Abbildung verankert. Sie, oder einfache Eigenschaften von ihnen, könnten daher auch außerhalb der Laufzeit automatisch erfasst werden. Bei der Nutzung eines prädiktiven Modells in einem realen Szenario müsste anhand dieser Eingaben eine Vorhersage erstellt werden. In den orange umrandeten Elementen finden sich die Einflussgrößen, die keine Eingabe für den Workflow darstellen. Ihnen werden erst zur Laufzeit Werte zugewiesen. Sie zu erfassen kann zu einer Verbesserung der Vorhersagen führen, widerspricht aber dem Grundprinzip von YW keine Informationen zur Laufzeit zu erfassen. Um jedoch den möglichen Nutzen solcher, zur Laufzeit erfasster, Informationen abschätzen zu können, wurden auch sie im Rahmen dieser Evaluation verwendet. Generell wurden alle farbig umrandeten Informationen, oder Metriken davon, zusammen mit der Ausführungsdauer, in der in Abschnitt 5.2.2 erläuterten CSV-Datei, zusammen mit der Anzahl der erstellten raw_image-Dateien, erfasst.

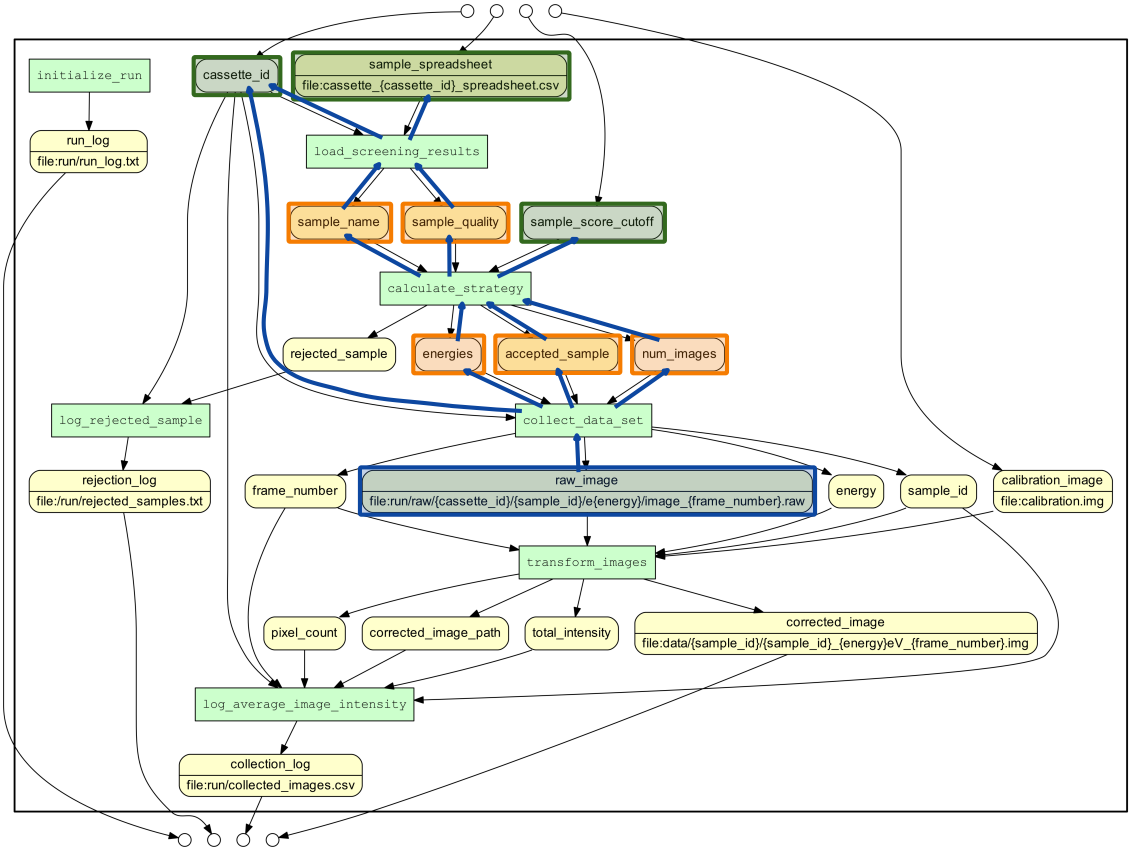


Abbildung 6.1: Abhängigkeiten der erstellten raw_image-Dateien. Grafik ursprünglich aus [MBBL15a].

Zusätzlich wurden anhand der CSV-Datei noch weitere Metriken, die in dieser Evaluation verwendet werden, mithilfe von Microsoft Excel und kurzen Python-Skripten bestimmt. Alle im Folgenden genannten Werte wurden auf zwei Nachkommastellen gerundet, um die Übersichtlichkeit zu bewahren:

- Im Durchschnitt bestanden die generierten **sample_spreadsheet-Dateien** aus 13,15 Zeilen.
- Pro Zeile einer **sample_spreadsheet-Datei**, also für jedes Sample, wurden durchschnittlich 137,18 **raw_image-Dateien** erstellt.
- Durchschnittlich wurden pro **sample_spreadsheet-Datei** 1803,46 Dateien erstellt.
- Im Mittel dauerte die Bearbeitung einer **sample_spreadsheet-Datei** 7,84 Sekunden auf dem verwendeten Rechner (Intel Core i5, 2,6 GHz, 16GB RAM). (In dieser Zeiterfassung ist jegliches Logging enthalten, was aber keinen Einfluss auf die Auswertung hat, da die Werte dennoch untereinander vergleichbar sind.)
- Die Bearbeitung eines Samples dauerte auf dem verwendeten Rechner durchschnittlich 0,6 Sekunden.

Ab wann die Qualität einer Vorhersage als ausreichend einzustufen ist, hängt vom konkreten Anwendungsfall ab und wird jeweils in den einzelnen Abschnitten bewertet. Auch muss im Rahmen dieser Diskussion immer betrachtet werden, dass es sich hierbei lediglich um eine theoretische Abschätzung handelt. Die Eingabedateien für die Trainingsdaten wurden anhand von Zufallszahlen generiert und der Zeitaufwand den ein Nutzer tatsächlich hätte, um sie erstellen, kann nicht verallgemeinert werden.

In den folgenden Abschnitten wird zunächst untersucht, ob eine nutzbare Vorhersage für die Anzahl der erstellten raw_image-Dateien anhand der globalen Eingaben möglich ist. Dann wird betrachtet, ob die Qualität besser und die notwendige Trainingsdatenmenge geringer wird, wenn man weitere Features zur Verfügung hat. Diese könnte man zwar mit einem Logging zur Laufzeit erfassen und somit als Trainingsdaten bereitstellen, allerdings wären in einem realen Szenario die Werte als Eingabe für ein Vorhersagemodell nicht verfügbar. Deshalb wird in Abschnitt 6.3 erörtert, ob es gewinnbringend ist, diese Zwischenergebnisse anhand der globalen Eingaben vorherzusagen.

6.2 Vorhersagen durch ein Modell für beliebig viele Blöcke

In diesem Abschnitt werden Vorhersagen von Modellen betrachtet, die nach dem in Abschnitt 5.2.3 beschriebenen Vorgehen für ein Modell für beliebig viele Blöcke erstellt wurden.

6.2.1 Vorhersage der Anzahl der erstellten Dateien ohne zur Laufzeit erfasste Informationen

Dem Grundsatz von YW folgend, wird zunächst die Qualität einer Vorhersage betrachtet, die nur Provenance-Informationen nutzt, die nicht zur Laufzeit erfasst werden müssen. Dies bedeutet das lediglich die Eingaben cassette_id, sample_spreadsheet und sample_score_cutoff und die Ausgabe, die erstellten raw_image-Dateien für eine Prädiktion verwendet werden können. Da im Rahmen der eingesetzten Regressionen nur numerische Parameter in Betracht kommen, findet cassette_id keine Verwendung.

Da cassette_id keinen Einfluss auf die Anzahl der erstellten Dateien hat, was sich auch daran zeigt, dass der Wert zufällig gewählt werden konnte, würde die Verwendung hier auch keinen Nutzen bringen. Lediglich in einem allgemeinen Fall, in welchem der Programmcode nicht manuell analysiert werden könnte, würde man alle Eingaben des Workflows für eine Vorhersage verwenden. Dazu wäre allerdings auch andere Verfahren zur Vorhersage notwendig.

Von der sample_spreadsheet-Datei wird die Anzahl der Zeilen als Metrik verwendet. Bei sample_score_cutoff handelt es sich um einen numerischen Wert, ebenso bei der Anzahl der erstellten raw_image-Dateien.

Im Rahmen eines vorstellbaren Nutzungsszenarios würde ein Nutzer die Werte für die Eingabeparameter festlegen. Durch die Wahl von cassette_id wird eine sample_spreadsheet-Datei selektiert, deren Zeilenanzahl leicht zu bestimmen ist. Für sample_score_cutoff würde er einen beliebigen Wert wählen. Anhand dieser beiden Werte sollte dann die Anzahl der erstellten raw_image-Dateien vorhergesagt werden.

Da ein Nutzer in diesem Szenario kein Interesse an der Information hat, wie viele Dateien pro Zeile der sample_spreadsheet-Datei erstellt werden, sondern lediglich wissen möchte wie viele Dateien insgesamt für eine gewählte sample_spreadsheet-Datei erstellt werden, muss die erstellte CSV-Datei mit den Trainingsdaten dahingehend angepasst werden. In der Datei ist nämlich nur erfasst wie viele Dateien pro Zeile jeder sample_spreadsheet-Datei, also pro Sample, erstellt werden. Deshalb werden Zeilen, die für dieses Szenario doppelte Informationen beinhalten, entfernt. Dies bedeutet konkret, dass für jede erfasste sample_spreadsheet-Datei eine Zeile in der für diese Evaluation verwendeten CSV-Datei enthalten ist. Da insgesamt, wie in Abschnitt 5.2.1 beschrieben wurde, 1000 sample_spreadsheet-Dateien generiert wurden, beinhaltet die Datei 1000 Zeilen. Eine Ausschnitt aus der Datei ist in Abbildung 6.2 zu sehen. Spalten, die für dieses Nutzungsszenario ohne Relevanz sind, sind grau unterlegt dargestellt. Jede dieser Zeilen enthält die Anzahl der Zeilen der verwendeten sample_spreadsheet-Datei, den sample_score_cutoff Wert und die Anzahl der daraus erstellen Dateien. Mithilfe dieser Daten kann das beschriebene Nutzungsszenario nun evaluiert werden.

cassette_id	number of lines in sample_spreadsheet	sample_name	sample_quality	sample_score_ cutoff	rejected_ sample	size of energies	accepted_ sample	num_images	number of raw_image-files	duration in sec	number of raw_image-files (whole spreadsheet)
q0	13	DRT926	15	23	1	0	0	0	0	0,000816822	2633
q1	3	DRT719	21	19	0	1	1	23	23	0,18170929	2337
q2	17	DRT760	15	2	0	5	1	17	85	0,676916122	4504
q3	2	DRT764	23	30	1	0	0	0	0	0,000499487	0
q4	23	DRT796	53	13	0	4	1	55	220	0,809203148	3545
...											
q998	18	DRT940	97	34	0	2	1	99	198	0,890075445	1935
q999	24	DRT306	29	46	1	0	0	0	0	0,000498295	1606

Abbildung 6.2: Alle ermittelten Provenance-Informationen aus 1000 sample_spreadsheet-Dateien nach der Duplikateliminierung. Spalten, die für diese Vorhersage nicht relevant sind, sind grau unterlegt. Ausschnitt aus der erstellten CSV-Datei vor dem zufälligen Vertauschen der Zeilenabfolge.

Hierzu werden für das in Abschnitt 5.2.3 vorgestellte Skript, für eine Vorhersage über beliebig viele Blöcke, die Anzahl der Zeilen der sample_spreadsheet-Datei und der Wert für sample_score_cutoff als Eingaben gewählt. Es werden mithilfe der im letzten Abschnitt beschriebenen CSV-Datei dann Modelle trainiert, die die Anzahl der raw_image-Dateien vorhersagen. Insgesamt liegen hierfür nun 1000 Datensätze zum Trainieren und Testen vor.

In Abbildung 6.3 ist auf der x-Achse die Anzahl der verwendeten Datensätze abgetragen. **Hierbei ist zu beachten, dass 80% dieser Daten für das Training der Modelle und 20% für das Testen der Modelle genutzt wurden.** Auf der y-Achse ist der NMAE, der in Abschnitt 2.4.4 erläutert wird, zu sehen. Alle noch folgenden verwendeten Diagramme sind analog aufgebaut und daher wird ihr allgemeiner Aufbau nur an dieser Stelle erläutert. Der jeweils gezeigte Ausschnitt wird so angepasst, dass nur die relevanten Werte zu erkennen sind. In Abbildung 6.3 sind die Ergebnisse der Verwendung einer linearen Regression und polynomieller Regressionsmodelle vom Grad drei, vier und fünf bei unterschiedlich vielen Datensätzen zu sehen. Auch dies wird in weiteren Diagrammen der Fragestellung angepasst.

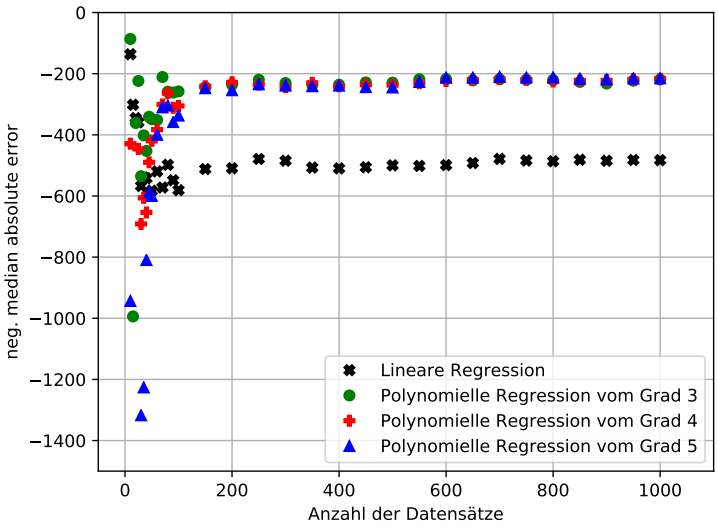


Abbildung 6.3: Der NMAE von vier Modellen zur Vorhersage der Anzahl an pro sample_spreadsheet erstellten raw_image-Dateien, anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Alle gemessenen Werte sind zu sehen.

Abbildung 6.3 soll für den Leser als Überblick dienen. Da alle gemessenen Werte zu sehen sind, überlagern sie sich. Es ist aber zu erkennen, dass bei einer geringen eingesetzten Datenmenge (<150) kaum Schlüsse möglich sind. Bei wenigen Datensätzen ist bei allen Verfahren eine Über- bzw. Unteranpassung zu beobachten, weshalb die Werte stark variieren und keine verlässliche Vorhersage möglich ist. Da es für die Ergebnisse dieser Arbeit keinen Unterschied macht, ob es zu einer Über- oder einer Unteranpassung kommt, wurde dies auch nicht weiter untersucht. Durch das Erhöhen der verwendeten Datensätze ist bei allen hier dargestellten Verfahren eine Konvergenz zu erkennen. Dann ist auch eine Vorhersage mit abschätzbarem Fehler möglich.

Lineare Regression

Um dem Leser zunächst den direkten Vergleich mit Abbildung 6.3 zu ermöglichen, wurde für Abbildung 6.4 derselbe Bildausschnitt gewählt. Es fällt im Vergleich zu Abbildung 6.3 auf, dass die Über- bzw. Unteranpassung, bei der hier betrachteten linearen Regression, deutlich schwächer ist, als bei den anderen drei Verfahren.

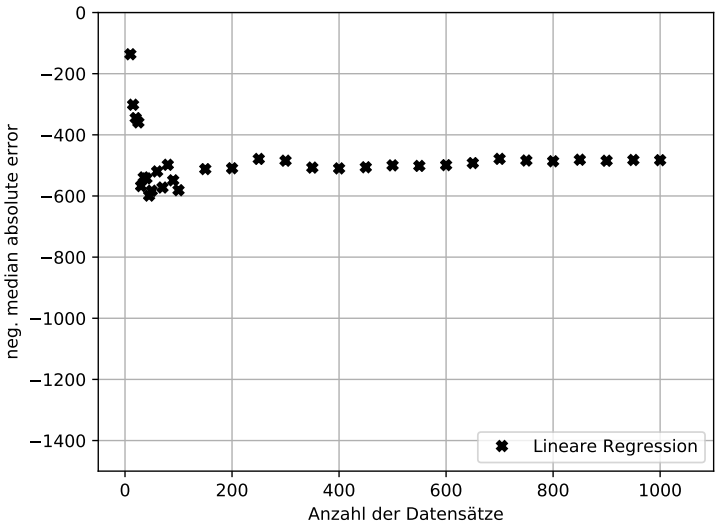


Abbildung 6.4: Der NMAE eines linearen Regressionsmodells zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Alle gemessenen Werte sind zu sehen.

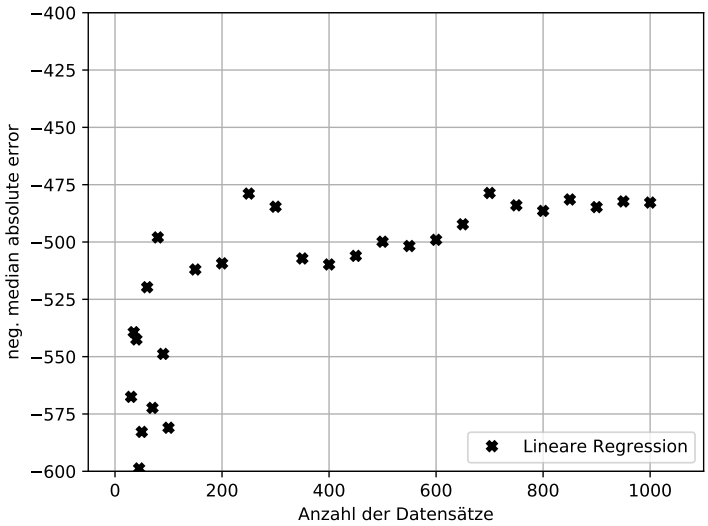


Abbildung 6.5: Der NMAE eines linearen Regressionsmodells zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Für Abbildung 6.5 wurde der Bildausschnitt so angepasst, dass die erreichten Werte für den NMAE besser erfasst werden können. Bei weniger als 600 verwendeten Datensätzen schwanken die Fehlerwerte stark. Da es sich bei den dargestellten Fehlerwerten um gemittelte Werte aus fünf Modellen handelt, kann hier keine Abschätzung vorgenommen werden wie groß der zu erwartende Fehler ist, wenn man eines der Modelle für eine Prädiktion nutzt. Eine Vorhersage, basierend auf dieser Trainingsdatenmenge wäre also nicht nutzbar. Ab 600 verwendeten Datensätzen bewegt sich der bestimmte Fehler zwischen -475 und -500. Die Modelle weisen alle ähnliche Fehlerraten auf. Bei 1000 verwendeten Datensätzen beträgt der gemittelte Fehler -482.

Durchschnittlich werden pro sample_spreadsheet-Datei 1803 Dateien erstellt. Wenn die Vorhersage hierbei um 482 Dateien im Mittel falsch ist, so entspricht dies einer Abweichung von 27%. Je nach Anwendungsszenario kann die Genauigkeit einer solchen Aussage zwar von Nutzen sein, dennoch müsste der Ablauf dafür bereits 800 mal in unveränderter Version ausgeführt worden sein.

Es ist zu erwarten, dass ein Wissenschaftler, wenn er den Workflow 800 mal ausführt, seine Forschungsfrage vollständig bearbeitet hat. Für ihn wäre eine Vorhersage dann also nutzlos. Sammelt man die Provenance-Informationen aber an einem zentralen, für viele Wissenschaftler zugreifbaren Ort, und handelt es ich um einen sehr häufig genutzten Workflow, so könnte man durchaus 800 Ausführungen erfassen.

Polynomielle Regressionen

In den Abbildungen 6.6, 6.7 und 6.8 sind die Ergebnisse der polynomiellen Regressionen analog zu Abbildung 6.4 zu sehen. Es ist zu erkennen, dass die Über- bzw. Unteranpassung bei den polynomiellen Regressionen bei geringen Datenmengen stärker ausgeprägt ist als bei der linearen Regression.

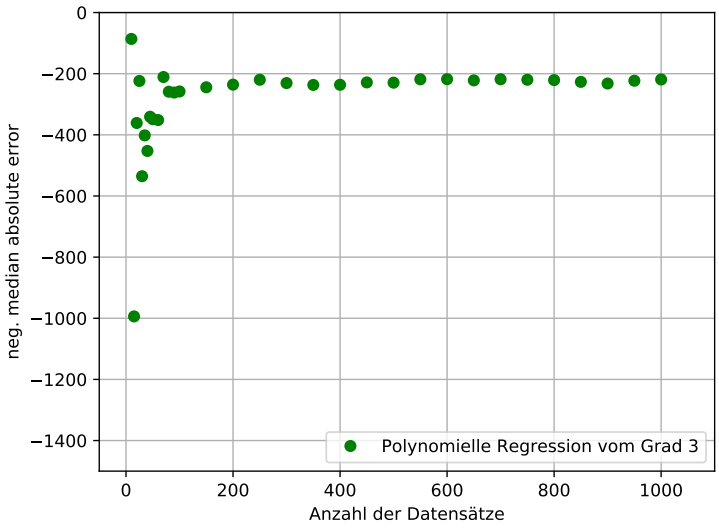


Abbildung 6.6: Der NMAE eines polynomiellen Regressionsmodells dritten Grades zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Alle gemessenen Werte sind zu sehen.

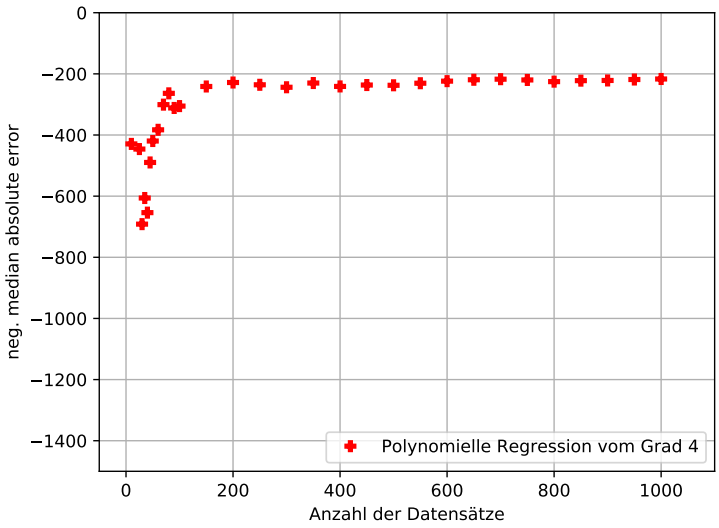


Abbildung 6.7: Der NMAE eines polynomiellen Regressionsmodells vierten Grades zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Alle gemessenen Werte sind zu sehen.

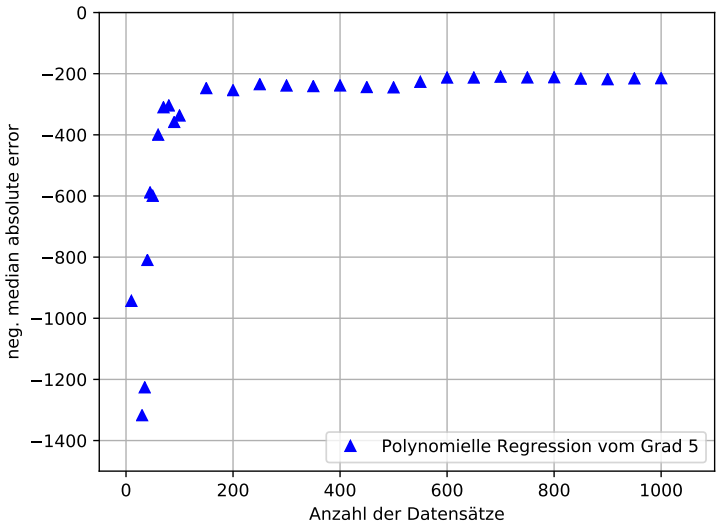


Abbildung 6.8: Der NMAE eines polynomiellen Regressionsmodells fünften Grades zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Alle gemessenen Werte sind zu sehen.

In Abbildung 6.9 sind die Ergebnisse der polynomiellen Regressionen genauer zu sehen. Es ist zu erkennen, dass mit einer Menge von 1000 Datensätzen und der Verwendung einer polynomiellen Regression fünften Grades ein Fehlerwert von etwa -215 erreicht wird. Anhand des Verlaufs der Fehlerwerte, bei zunehmender Menge an Datensätzen, lässt sich abschätzen, dass die Vorhersage auch mit noch mehr Datensätzen nicht signifikant besser wird. Dies ist zwar eine deutliche Verbesserung der zu erwartenden Vorhersagequalität im Vergleich zur Nutzung einer linearen Regression, dennoch entspricht dies einer Abweichung von 12%. Zusätzlich werden dafür mindestens 480 Datensätze aus erfassten Durchläufen benötigt, da erst ab etwa 600 Datensätzen (480 Trainingsdatensätze + 120 Testdatensätze) die Werte in Abbildung 6.9 nicht mehr so stark schwanken.

Dies lässt sich auch aus den Fehlerwerten für die fünf trainierten und getesteten Modelle ablesen. Bei einer Verwendung von 600 Datensätzen zum Trainieren und Testen und einer polynomiellen Regression dritten Grades sind die Fehlerwerte der fünf trainierten Modelle: -239.99, -199.93, -229.35, -199.71, -220.38. Der zu erwartende Fehler schwankt also um 40,28 zwischen den Modellen. Bei einer Verwendung von 700 Datensätzen: -232.06, -222.80, -222.74, -218.80, -194.80. Hier schwankt der Fehler um 37,26. Um dem Leser an dieser Stelle noch einen Vergleich anzubieten, werden noch die Fehlerwerte bei 1000 Datensätzen angegeben: -219.46, -237.25, -200.56, -217.09, -220.68. Hier beläuft sich die Schwankung zwischen den Modellen also nur noch auf 20,12.

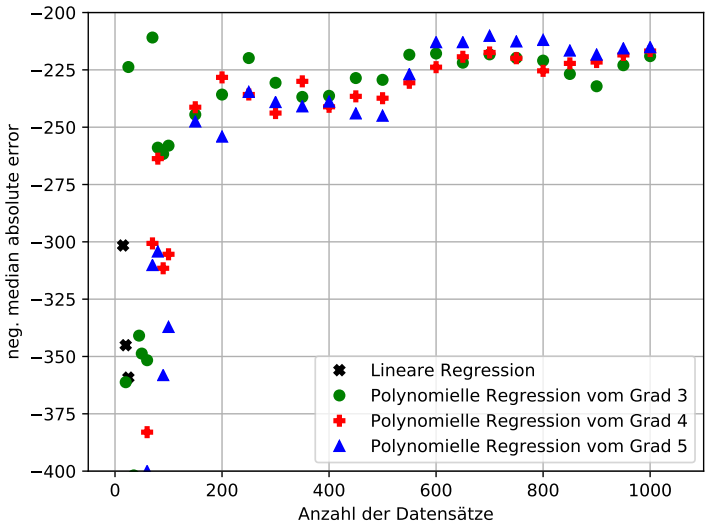


Abbildung 6.9: Der NMAE von vier verschiedenen Verfahren zur Vorhersage der Anzahl an erstellten raw_image-Dateien pro sample_spreadsheet anhand der Anzahl der Zeilen einer sample_spreadsheet-Datei und dem Wert für sample_score_cutoff. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

480 Durchläufe erscheint zunächst als unverhältnismäßig große Anzahl. Wenn es sich jedoch um einen Workflow handelt, der zahlreiche Nutzer hat und häufig genutzt wird, so können die Daten durchaus bereitgestellt werden. Je nach Anwendungsszenario (z.B. wenn die Ausführung eines Durchlaufs sehr lange dauert) können zukünftige Läufe damit optimiert werden und es ergibt sich ein Mehrwert für alle Nutzer. Handelt es sich allerdings um einen gelegentlich genutzten Workflow, so ist zu erwarten, dass 480 Durchläufe zu erfassen als unrealistisch angesehen werden muss.

6.2.2 Vorhersage der Anzahl der erstellten Dateien unter Ausnutzung von Informationen die sich aus der detaillierten Analyse von Eingabeobjekten ergeben oder zur Laufzeit erfasst werden müssen

Im letzten Abschnitt wird erläutert, dass die Qualität der Vorhersage, wie viele raw_image-Dateien pro sample_spreadsheet-Datei erstellt werden, für entsprechende Anwendungsszenarien ausreichend sein kann. Allerdings müssen für eine solche Vorhersage mindestens 480 Durchläufe des Workflows bereits erfasst worden sein. In diesem Abschnitt wird erörtert, ob durch zusätzliche Informationen, die vor der Laufzeit anhand einer inhaltlichen Analyse der Eingabeobjekte bestimmt werden können oder zur Laufzeit erfasst werden müssen, diese Zahl verringert werden kann.

Beim Betrachten des Workflows in Abbildung 6.1 kann anhand der blauen Pfeile, entgegen der Reihenfolge der Ausführung des Workflows, verfolgt werden, welche Größen Einfluss auf die Anzahl der erstellten raw_image-Dateien haben. Aus dem ersten ausgeführten Block resultieren sample_name und sample_quality, die auch Einflussfaktoren für die erstellte Anzahl sind. Bei sample_name handelt es sich um einen String, weshalb der Wert nicht als Variable in einer

Regression in Frage kommt. Bei sample_quality handelt es sich um eine Zahl, die, wie aus dem Programmcode leicht entnommen werden kann, lediglich aus der sample_spreadsheet-Datei für jedes Sample, also jede Zeile, gelesen wird.

Der Block load_screening_results dient nur dem Einlesen der sample_spreadsheet-Datei. Für diesen Workflow hat sich diese Erkenntnis aus der Analyse des Programmcodes ergeben, was aufgrund des trivialen Workflows möglich war. Wenn man allerdings davon ausgeht, dass langfristig komplexere Workflows eingesetzt werden und die Modelle automatisch generiert werden sollen, müsste diese Tatsache auch automatisch bestimmt werden. Damit könnten die Informationen vor der Laufzeit anhand der sample_spreadsheet-Datei erfasst werden. Im allgemeinen Fall muss jedoch davon ausgegangen werden, dass Informationen, die keine globalen Eingaben sind, als erst zur Laufzeit bekannt zu behandeln sind.

Zunächst soll aber an dieser Stelle bestimmt werden, ob diese Informationen überhaupt zu einer Verringerung der notwendigen Trainingsdatenmenge führen. Für das in Abschnitt 5.2.3 vorgestellte Skript werden also die Anzahl der Zeilen der sample_spreadsheet-Datei, die enthaltenen Werte für sample_quality und der Wert für sample_score_cutoff als Eingaben gewählt. Da durch sample_quality nun einzelne Zeilen, also Samples, aus der sample_spreadsheet-Datei als unabhängige Variable verwendet werden, muss auch die Vorhersage für die von den Zeilen erstellten raw_image-Dateien erfolgen (und nicht wie in Abschnitt 6.2.1 für die pro sample_spreadsheet-Datei erstellen raw_image-Dateien). Daher kann die CSV-Datei in unveränderter Form, wie sie in Abschnitt 5.2.2 beschrieben wurde, verwendet werden. Es werden dann Modelle trainiert, die die Anzahl der raw_image-Dateien pro Sample vorhersagen.

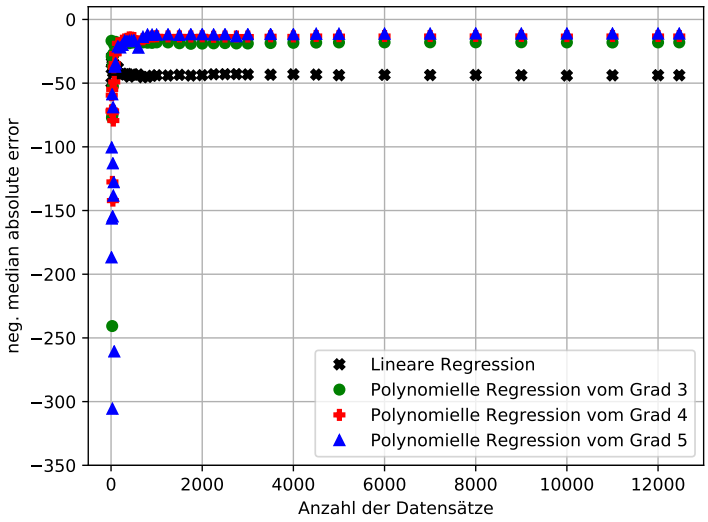


Abbildung 6.10: Der NMAE der Vorhersagen durch verschiedene Regressionsverfahren. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand der Zeilenanzahl in der sample_spreadsheet-Datei, der sample_quality und des Werts für sample_score_cutoff vorhergesagt. Alle gemessenen Werte sind zu sehen.

In Abbildung 6.10 sind die Werte für den NMAE in Abhängigkeit der Anzahl der verwendeten Datensätze zu sehen. Als unabhängige Variablen für die Regressionen wurden die Anzahl der Zeilen der sample_spreadsheet-Datei, die enthaltenen Werte für sample_quality und der Wert für sample_score_cutoff genutzt. Vorhergesagt wurde die Anzahl der erstellten raw_image-Dateien. Wie bereits in Abschnitt 6.2.1 ist bei einer zu geringen Menge an Trainingsdaten eine Über- bzw. Unteranpassung aller Modelle in dieser Überblicksvisualisierung zu erkennen. Ebenso kann bei allen Verfahren eine Konvergenz festgestellt werden. Insgesamt betrachtet hat sich die Qualität der Vorhersagen maßgeblich durch die zusätzlich verwendeten Informationen verbessert. Die Werte für den NMAE belaufen sich hier bei genügend vorliegenden Trainingsdaten zwischen -10 und -45. Dies kann auch Abbildung 6.11 entnommen werden.

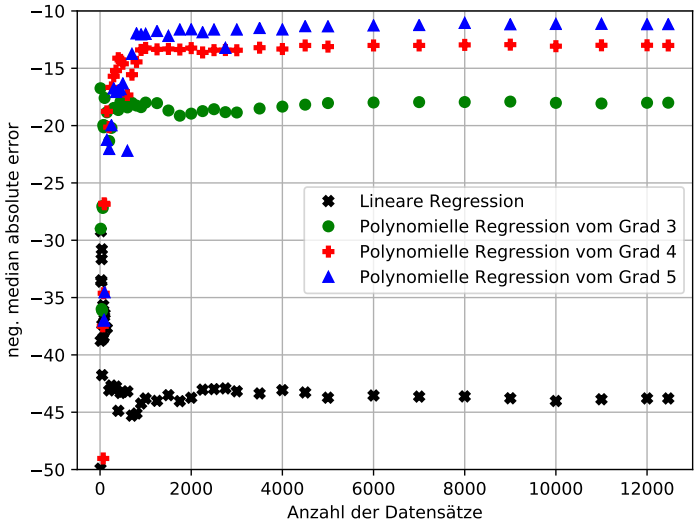


Abbildung 6.11: Der NMAE der Vorhersagen durch verschiedene Regressionsverfahren. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand der Zeilenanzahl in der sample_spreadsheet-Datei, der sample_quality und des Werts für sample_score_cutoff vorhergesagt. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

In Abbildung 6.11 ist zu sehen, dass bei einer ausreichenden Menge an Trainingsdaten, durch die Nutzung einer polynomiellen Regression vom Grad 5, das beste Ergebnis erreicht werden kann. Grad 4 führt zu einem ähnlich guten Ergebnis. Durch die polynomielle Regression dritten Grades wird eine schlechtere Vorhersage getroffen. Die lineare Regression ist zwar besser als in Abschnitt 6.2.1, jedoch deutlich schlechter als die polynomiellen Regressionen. Da jedoch, wie in Abschnitt 6.2.1 bereits festgestellt wird, je nach Anwendungsszenario nicht die Qualität der Vorhersage, sondern die notwendige Trainingsdatenmenge der kritische Faktor ist, wird dies im Folgenden weiter betrachtet.

Lineare Regression

Wie in Abbildung 6.12 zu erkennen ist, schwankt der Fehlerwert ab 900 verwendeten Datensätzen bei der linearen Regression nur noch zwischen -40 und -45. Das heißt, dass ab 720 Trainingsdatensätzen eine Vorhersage der Anzahl der erstellten Dateien pro Zeile einer sample_spreadsheet-Datei mit einer durchschnittlichen Abweichung von maximal 45 Dateien getroffen werden kann. Da pro Zeile einer sample_spreadsheet-Datei durchschnittlich 137,18 Dateien erstellt wurden, entspricht dies einer Abweichung von etwa 33%. Da in jeder sample_spreadsheet-Datei durchschnittlich 13,15 Zeilen, also Samples, enthalten waren, entsprechen 720 Trainingsdatensätze etwa 55 Durchläufen des Workflows (mit durchschnittlichen sample_spreadsheet-Dateien).

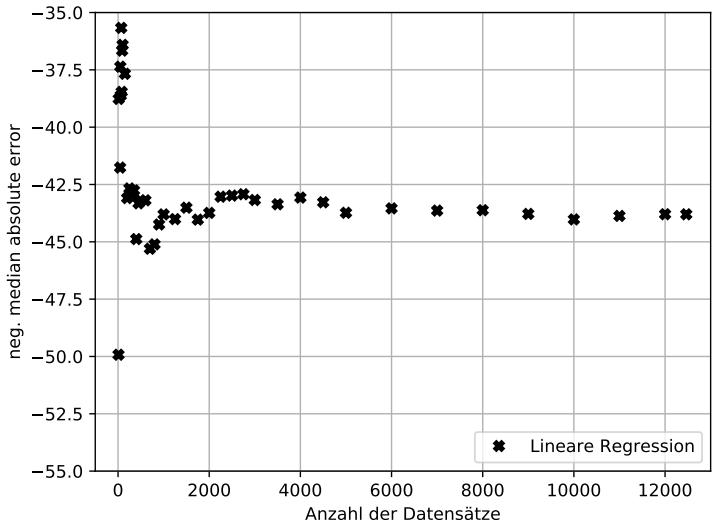


Abbildung 6.12: Der NMAE des linearen Regressionsmodells. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand der Zeilenanzahl in der sample_spreadsheet-Datei, der sample_quality und des Werts für sample_score_cutoff vorhergesagt. Alle gemessenen Werte sind zu sehen.

Anhand des Programmcodes des Proteinkristall-Workflows kann man darauf schließen, dass die Anzahl der Zeilen in der sample_spreadsheet-Datei keinen Einfluss darauf haben kann, wie viele Dateien für jede Zeile generiert werden. Dies lässt sich auch an denn zugeordneten Koeffizienten für die einzelnen Features in den Regressionsgleichungen erkennen. Die Koeffizienten werden hier beispielhaft für Verwendung von 4000 Datensätzen angegeben:

- Koeffizient für number_of_lines_in_sample_spreadsheet: 0.09
- Koeffizient für sample_quality: 3.80
- Koeffizient für sample_score_cutoff: -5.13

Die Gewichtung für number_of_lines_in_sample_spreadsheet ist nahezu 0, während sample_score_cutoff am stärksten gewichtet wird.

Polynomielle Regressionen

Beim Betrachten der polynomiellen Regressionen in Abbildung 6.11 lässt sich feststellen, dass mit Grad 5, bei einer großen Menge an verfügbaren Trainingsdaten, die Fehlerrate bei der Vorhersage am geringsten ist. Die polynomiellen Modelle vom Grad 3 weisen die größte Fehlerrate auf.

Wie bereits in Abbildung 6.11 zu erkennen ist, scheint ein Modell dritten Grades mit der geringsten Menge an verfügbaren Daten bereits gleichbleibende Ergebnisse zu liefern. Bereits ab 2000 verwendeten Datensätzen schwankt der Fehler nur noch zwischen -17 und -19. Dies kann anhand von Abbildung 6.13 nachvollzogen werden.

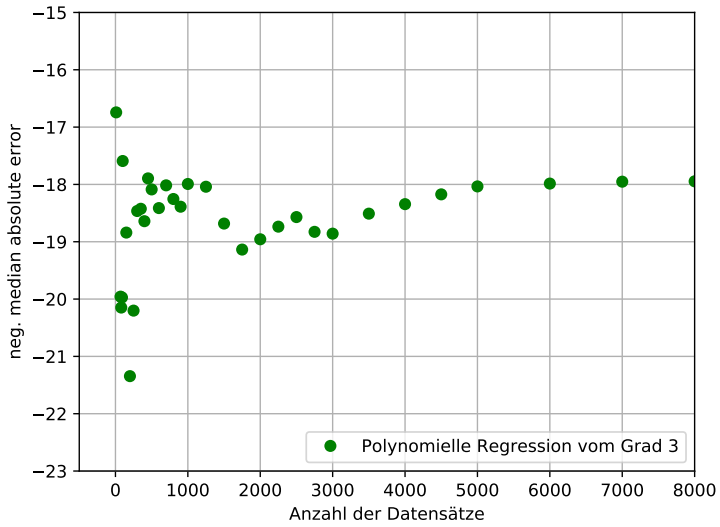


Abbildung 6.13: Der NMAE des polynomiellen Regressionsmodells dritten Grades. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand der Zeilenanzahl in der sample_spreadsheet-Datei, der sample_quality und des Werts für sample_score_cutoff vorhergesagt. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Der erreichte durchschnittliche Fehler von maximal -19 entspricht einer Abweichung von 14%. Dazu müssen 1600 Trainingsdatensätze verfügbar sein. Bei durchschnittlichen sample_spreadsheet-Dateien, die 13,15 Zeilen enthalten, entspricht dies 122 Ausführungen des Workflows.

Auf die beiden weiteren polynomiellen Regressionen wird an dieser Stelle nicht weiter eingegangen, da sie die notwendige Menge an Trainingsdaten nicht zu verringern scheinen. Lediglich der errechnete Fehler lässt sich auf 8% mit der Verwendung einer polynomiellen Regression fünften Grades vermindern.

6.2.3 Vorhersage der Anzahl der erstellten Dateien unter Ausnutzung aller zur Laufzeit erfassbaren Informationen

In einem letzten Schritt wird nun diskutiert, ob die für das Training der Modelle notwendige Trainingsdatenmenge verringert werden kann, wenn alle Informationen, die zur Laufzeit erfassbar sind, für eine Vorhersage genutzt werden. Betrachtet man dazu nochmals Abbildung 6.1, so ergeben sich als unabhängige Variablen, die zu der Anzahl der erstellen raw_image-Dateien beitragen, die Folgenden:

- **Cassette_id**, was kein numerischer Wert ist und daher hier keine Verwendung findet.
- Von der gewählten **sample_spreadsheet-Datei** wird wiederum die Zeilenanzahl verwendet.
- Bei **sample_name** handelt es sich um einen String, der nicht verwendet werden kann.
- **Sample_quality** kann direkt verwendet werden.
- **Sample_score_cutoff** steht bereits vor der Laufzeit fest und kann auch direkt verwendet werden.
- Bei **energies** handelt es sich um ein Array, dessen Länge hier erfasst und verwendet wird.
- **Accepted_sample** wurde als „1“ erfasst, wenn für das betrachtete Sample Dateien erstellt wurden.
- **Num_images** kann direkt verwendet werden.

Da sample_quality als Eingabe für die Regressionen verwendet werden soll, bedeutet dies, dass eine Vorhersage nur für die Anzahl der erstellten Dateien pro Zeile, also pro Sample, einer sample_spreadsheet-Datei Sinn macht.

Zunächst wird überprüft, ob mit diesen Informationen eine Verbesserung erreicht werden kann. In einem realen Szenario müssten sie für eine Vorhersage allerdings einem Nutzer (was nicht eintreten kann) oder YW vor der Ausführung des Workflows bekannt sein.

Hierzu ist in Abbildung 6.14 der bestimmte NMAE für die vier im Rahmen dieser Arbeit getesteten Verfahren zu sehen. Wie bereits in den anderen Schaubildern ist auch hier eine Über- bzw. Unteranpassung bei einer zu geringen Menge an Trainingsdaten zu erkennen. Ebenso kann leicht abgelesen werden, dass bei einer ausreichenden Menge an Trainingsdaten kaum noch eine Abweichung zwischen Vorhersage und der dann tatsächlich erstellten Anzahl zu erwarten ist. Um die dazu notwendige Datenmenge abschätzen zu können, ist in Abbildung 6.15 ein entsprechender Ausschnitt zu sehen.

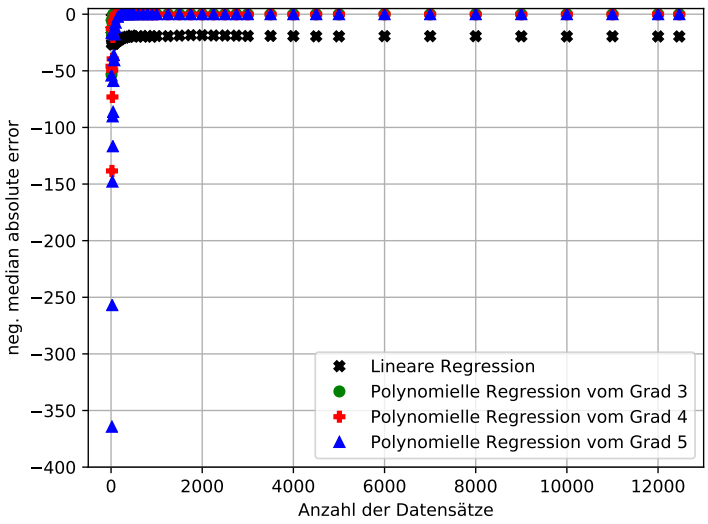


Abbildung 6.14: Der NMAE verschiedener Regressionsmodelle. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand aller erfassbaren Informationen vorhergesagt. Alle gemessenen Werte sind zu sehen.

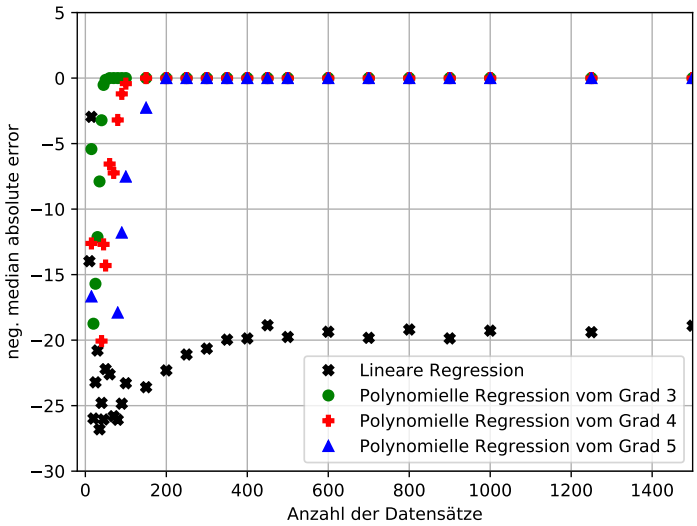


Abbildung 6.15: Der NMAE verschiedener Regressionsmodelle. Es wird die Anzahl an erstellten raw_image-Dateien pro Sample anhand aller erfassbaren Informationen vorhergesagt. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Bei der Verwendung der linearen Regression ist stets mit einem größeren Fehler als bei den polynomiellen Verfahren zu rechnen. Wie in Abbildung 6.15 zu erkennen ist, kann hier bereits ab 60 verwendeten Datensätzen mit der polynomiellen Regression dritten Grades eine Vorhersage

ohne zu erwartende Abweichung getroffen werden. Dazu sind, wie bereits erläutert, 48 bearbeitete Zeilen, also Samples aus sample_spreadsheet-Dateien, notwendig. Auf dem verwendeten Rechner dauerte die Bearbeitung eines Samples 0,6 Sekunden. Die hier gemessene Zeit beinhaltet auch sämtliches zusätzliches Logging. Das heißt, dass diese Informationen in unter 30 Sekunden erstellt werden könnten. In diesem Fall ist sowohl die Qualität einer Vorhersage als auch die notwendige Trainingsdatenmenge sehr gut.

Ein solche Vorhersage, wie sie hier theoretisch diskutiert wird, hat keinen direkten Nutzen für einen Anwender. Wenn dieser den Programmcode so gut kennt, dass er all diese Parameter vor der Ausführung angeben kann, so ist vermutlich keine Vorhersage der Anzahl der erstellten Dateien mehr notwendig. Ansonsten sind ihm die Parameter nicht bekannt, so dass er sie auch nicht für eine Regression verwenden kann.

6.2.4 Auswertung der Vorhersagen eines Modells über beliebig viele Blöcke

Im Rahmen der Beobachtung aus Abschnitt 6.2.3 muss erörtert werden, ob die hierfür benötigten Informationen auf anderem Wege, als durch eine Nutzereingabe, zu erreichen sind. Wie bereits in Abschnitt 6.2.1 beschrieben, kann lediglich davon ausgegangen werden, dass der Nutzer anhand der gewählten sample_spreadsheet-Datei und des gewählten Werts für sample_score_cutoff wissen möchte, wie viele Dateien voraussichtlich erstellt werden.

Zusätzlich kann man noch den Fall betrachten, dass der Nutzer den Inhalt der sample_spreadsheet-Datei kennt und Werte aus der Datei oder Metriken dazu als Eingabe für die Regression verwenden kann. Im allgemeinen Fall muss aber davon ausgegangen werden, dass nur globale Eingaben des Workflows oder oberflächliche Metriken davon verwendet werden können.

	Vorhersage über drei Blöcke (Abschnitt 6.2.1)	**Vorhersage über zwei Blöcke (Abschnitt 6.2.2)**	**Vorhersage über einen Block (Abschnitt 6.2.3)**
Geeignetstes Verfahren hinsichtlich erwartetem Fehler und notwendiger Datenmenge	polynomielle Regression fünften Grades	polynomielle Regression dritten/fünften Grades	polynomielle Regression dritten Grades
Erwartete Abweichung der vorhergesagten Dateianzahl (bester Wert)	12%	8%	0%
Notwendige Anzahl an erfassten, bearbeiteten Samples (stabile Vorhersage)	6312 (entspricht 480 durchschnittlichen sample_spreadsheet-Dateien)	480	48

Tabelle 6.1: Überblick über die notwendige Trainingsdatenmenge bei einer Vorhersage über unterschiedlich viele Blöcke.

Betrachtet man Abbildung 6.1 nun nochmals, so fällt auf, dass mit der Verwendung aller verfügbaren Informationen in Abschnitt 6.2.3 nur eine Vorhersage für den letzten betrachteten Codeblock, collect_data_set, getroffen wird. Im Abschnitt 6.2.2 wurde dagegen eine Vorhersage über zwei

Blöcke, calculate_strategy und collect_data_set, erstellt. In Abschnitt 6.2.1 wurde über drei Blöcke vorhergesagt. Allerdings ist hier anzumerken, dass load_screening_results lediglich die Daten einliest.

Analog zu dieser Beobachtung konnte in den Abschnitten auch gezeigt werden, dass eine Vorhersage über eine kurze Distanz bessere Ergebnisse erzielt und eine geringere Menge an Trainingsdaten benötigt. Dies kann zusammenfassend Tabelle 6.1 entnommen werden.

Im Folgenden wird evaluiert, ob ähnlich der in Abschnitt 3.4 vorgestellten Arbeit von Hiden et al., hierzu einzelne Modelle für jeden Block trainiert werden können.

6.3 Vorhersagen durch die Kombination von Modellen

Aus den Ergebnissen des letzten Abschnitts ist ableitbar, dass eine vergleichende Betrachtung zwischen einer Vorhersage über mehrere Blöcke hinweg und das Aneinanderreihen von Modellen für einzelne Blöcke vielversprechend erscheint.

Um die Qualität der zwei unterschiedlichen Vorgehensweisen an dem Proteinkristall-Workflow exemplarisch zu betrachten, wird dazu eine Vorhersage der Anzahl der pro Sample erstellten raw_image-Dateien anhand der Werte für sample_quality und sample_score_cutoff gewählt. Der Ausschnitt des Workflows, welcher dieser Vorhersage entspricht, beinhaltet zwei Blöcke, calculate_strategy und collect_data_set. Er ist in Abbildung 6.16 farbig gekennzeichnet. Relevante Abhängigkeiten sind anhand der blauen Pfeile zu erkennen.

Die Wahl des in Abbildung 6.16 markierten Ausschnitts lässt sich damit begründen, dass es sich bei load_screening_results um einen Programmblock handelt, der lediglich Nutzereingaben einliest. Diese Informationen könnten anhand der vom Nutzer gewählten sample_spreadsheet-Datei in anderer Weise, ohne eine Vorhersage, bereitgestellt werden.

Sample_name kann in direkter Form nicht verwendet werden, da es sich um keinen numerischen Wert handelt. An dieser Stelle würde die Verwendung einer Metrik (beispielsweise der Anzahl) auch keinen Sinn machen, da anhand eines einzelnen Wertes für sample_quality auch nur die voraussichtlich erstellte Dateianzahl für ein Sample vorhergesagt werden kann. Formal betrachtet wäre, wenn man die Anzahl von sample_name verwenden möchte, dieser Wert sowohl in den Trainingsdaten als auch bei der Vorhersage mit der erstellten Modell immer 1 und damit nutzlos.

Die beiden hier betrachteten Vorgehensweisen wurden bereits in Abschnitt 5.2.3 im Rahmen der erstellten Skripte ausführlich erläutert. Die grundsätzliche Idee wird aber zu Beginn des jeweiligen Abschnitts wiederholt.

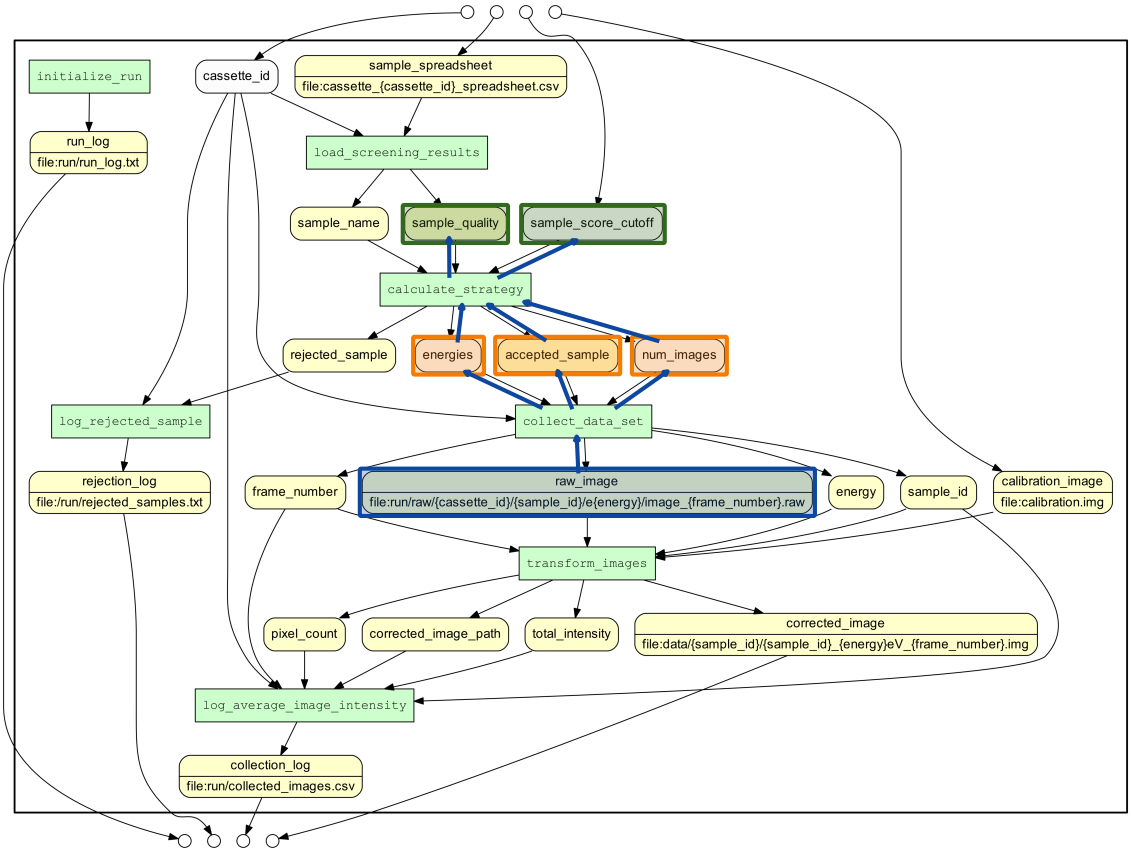


Abbildung 6.16: Für diesen Abschnitt betrachteter Ausschnitt des Workflows. Grafik ursprünglich aus [MBBL15a].

6.3.1 Erreichter Fehlerwert bei Verwendung eines Modells für beide Blöcke

In Abbildung 5.4 ist das in den Abschnitten 6.2.1 bis 6.2.3 verwendete Vorgehen schematisch dargestellt zu sehen.

Mit diesem Verfahren wird nun ein Modell generiert, das über zwei Blöcke hinweg eine Vorhersage erstellt. Dazu werden sample_quality und sample_score_cutoff verwendet, um die Anzahl der pro Sample erstellten raw_image-Dateien vorherzusagen. Anhand des Prototyps zur Evaluation wird die Qualität einer solchen Vorhersage bestimmt.

Eine Überblickvisualisierung hierzu ist in Abbildung 6.17 zu sehen. Bei wenigen Datensätzen ist, wie bereits mehrfach beschrieben, eine Über- bzw. Unteranpassung aller Modelle zu beobachten. Bei einer ausreichenden Datenmenge schwankt der zu erwartende Fehler dann je nach verwendetem Verfahren zwischen -9 und -45.

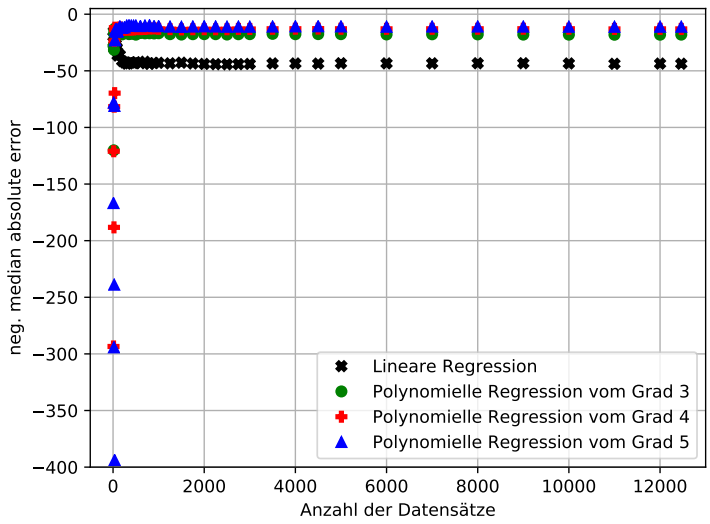


Abbildung 6.17: Der NMAE der Vorhersage der Anzahl der pro Sample erstellten raw_image-Dateien aufgrund der Werte für sample_quality und sample_score_cutoff mit einem Modell. Alle gemessenen Werte sind zu sehen.

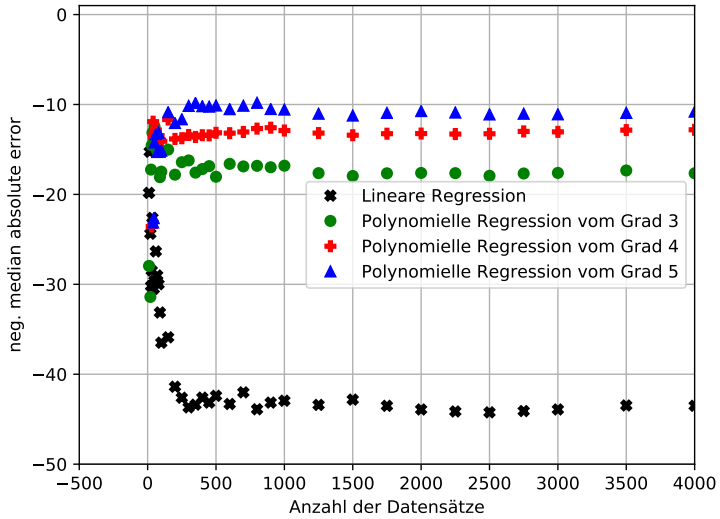


Abbildung 6.18: Der NMAE der Vorhersage der Anzahl der pro Sample erstellten raw_image-Dateien aufgrund der Werte für sample_quality und sample_score_cutoff mit einem Modell. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

In Abbildung 6.18 wurde der Bildausschnitt so angepasst, dass zu erkennen ist, das ab 500 verwendeten Datensätzen der Fehlerwert bei der polynomiellen Regression fünften Grades kaum noch schwankt und für eine Vorhersage ein Fehler zwischen -9 und -11 zu erwarten ist. Da pro Sample durchschnittlich 137,18 Dateien erstellt wurden, entspricht dies einer Abweichung von maximal 8%. Dies scheint das hier am besten geeignete Verfahren für eine Vorhersage zu sein.

6.3.2 Erreichter Fehlerwert bei der Verwendung aneinandergereihter Modelle

Zum Vergleich wird nun eine Vorhersage für die gleichen Features und Targets vorgenommen. Allerdings werden anstelle eines Modells für zwei Blöcke, Modelle für jeden Block trainiert.

Die Idee ist hierbei, aus den Werten für sample_quality und sample_score_cutoff mit drei verschiedenen Modellen Werte für size_of_energies, accepted_sample und num_images vorherzusagen. Mit diesen Werten soll dann wiederum mit einem vierten Modell ein Wert für die Anzahl der pro Sample voraussichtlich erstellen raw_image-Dateien vorhergesagt werden. Das Verfahren ist schematisch in Abbildung 5.5 zu sehen.

Zunächst muss hierfür bestimmt werden, welches der eingesetzten Lernverfahren sich für Modell 1, 2, 3 und 4 am besten eignet. Dazu wird wieder das in Abschnitt 5.2.3 vorgestellte Skript zur Vorhersage über beliebig viele Blöcke genutzt.

Modell 1

Mit Modell 1 soll, wie Abbildung 5.5 zu entnehmen ist, eine Vorhersage für size of energies, anhand von sample_quality und sample_score_cutoff, erstellt werden. Die Ergebnisse einer solchen Vorhersage sind in Abbildung 6.19 zu sehen. Alle eingesetzten Verfahren führen zu einem sehr geringen Fehlerwert, wobei zu erkennen ist, dass für dieses Modell die polynomielle Regression fünften Grades den geringsten Fehler erwarten lässt (ab etwa 40 Datensätzen wird hier ein NMAE zwischen -0,4 und -0,2 erreicht).

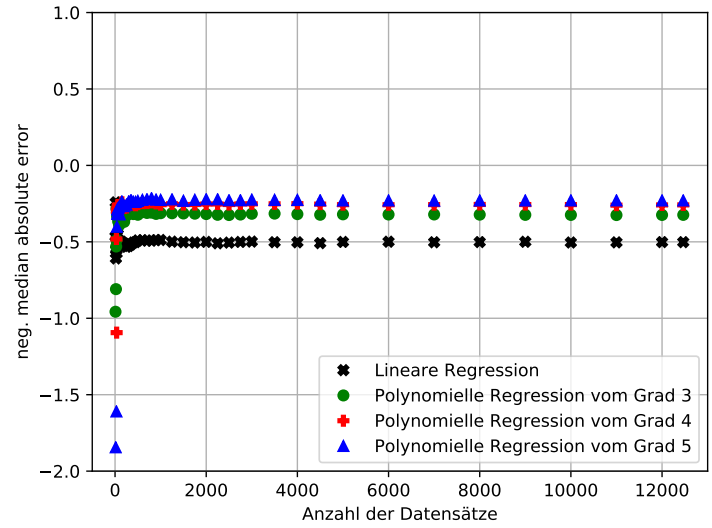


Abbildung 6.19: Der NMAE einer Vorhersage von size_of_energies anhand von sample_quality und sample_score_cutoff. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Modell 2

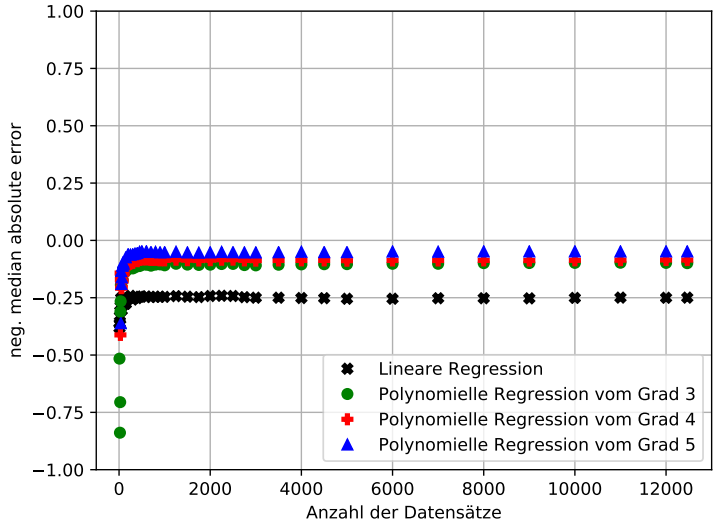


Abbildung 6.20: Der NMAE einer Vorhersage von accepted_sample anhand von sample_quality und sample_score_cutoff. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Mit Modell 2 soll, wie Abbildung 5.5 zu entnehmen ist, eine Vorhersage für accepted_sample, anhand von sample_quality und sample_score_cutoff erstellt werden. Die Ergebnisse einer solchen Vorhersage sind in Abbildung 6.20 zu sehen. Alle eingesetzten Verfahren führen zu einem sehr geringen Fehlerwert, wobei zu erkennen ist, dass für dieses Modell die polynomielle Regression fünften Grades den geringsten Fehler erwarten lässt (ab etwa 45 Datensätzen wird ein NMAE zwischen -0,2 und -0,05 erreicht).

Modell 3

Mit Modell 3 soll, wie Abbildung 5.5 zu entnehmen ist, eine Vorhersage für num_images, anhand von sample_quality und sample_score_cutoff erstellt werden. Die Ergebnisse einer solchen Vorhersage sind in Abbildung 6.21 zu sehen. Alle eingesetzen Verfahren führen zu einem sehr geringen Fehlerwert, wobei zu erkennen ist, dass für dieses Modell die polynomielle Regression fünften Grades den geringsten Fehler erwarten lässt (ab etwa 250 Datensätzen wird ein NMAE zwischen -2 und -1 erreicht).

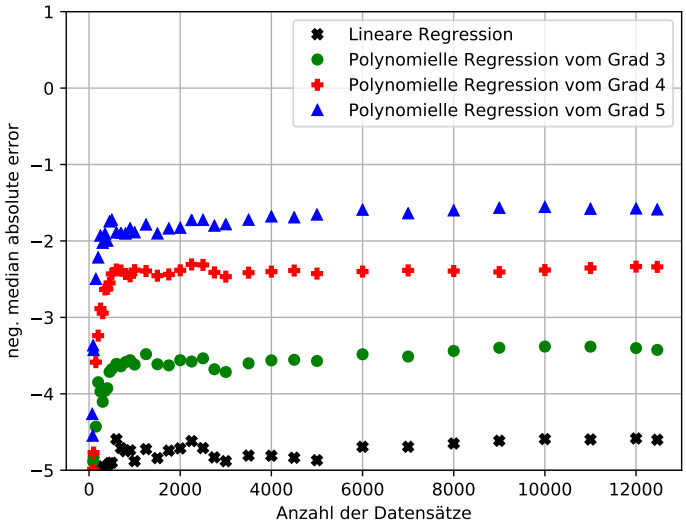


Abbildung 6.21: Der NMAE einer Vorhersage von num_images anhand von sample_quality und sample_score_cutoff. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Modell 4

Mit Modell 4 soll, wie Abbildung 5.5 zu entnehmen ist, eine Vorhersage für die Anzahl der erstellten raw_image Dateien, anhand von size of energies, accepted_sample und num_images, erstellt werden. Die Ergebnisse einer solchen Vorhersage sind in Abbildung 6.22 zu sehen. Hier ist auch zu sehen, dass für diese Vorhersage eine lineare Regression deutlich schlechter geeignet scheint, als die polynomiellen Verfahren (ab etwa 35 Datensätzen kann eine Vorhersage ohne Fehler getroffen werden).

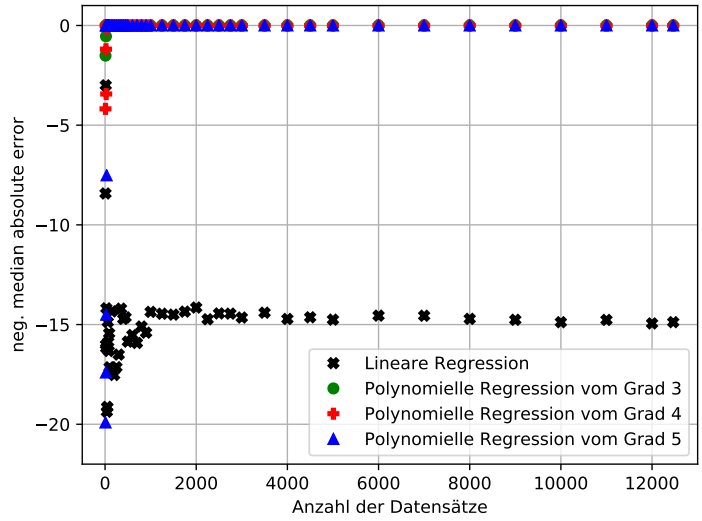


Abbildung 6.22: Der NMAE einer Vorhersage der Anzahl an pro Sample erstellten raw_image-Dateien anhand von size_of_energies, accepted_sample und num_images. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

In Abbildung 6.23 wurde der Bildausschnitt so angepasst, dass zu erkennen ist, dass alle polynomiellen Verfahren Vorhersagen von ähnlicher Qualität erstellen.

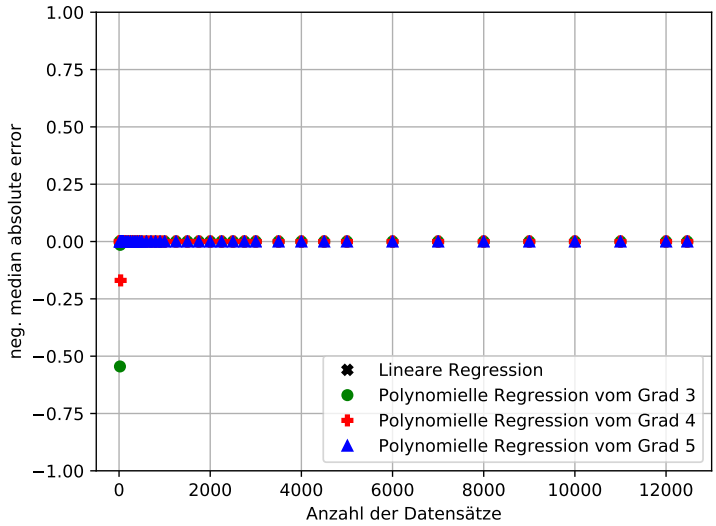


Abbildung 6.23: Der NMAE einer Vorhersage der Anzahl an pro Sample erstellten raw_image-Dateien anhand von size_of_energies, accepted_sample und num_images. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Anhand der so ermittelten optimalen Verfahren wurde ein Skript erstellt, dass die einzelnen Modelle mit den genannten Verfahren trainiert. Da bei Modell 4 alle polynomiellen Verfahren in gleicherweise geeignet scheinen, wird für alle Modelle polynomielle Regression fünften Grades verwendet.

6.3.3 Auswertung der aneinandergereihten Modelle

Wie bereits in Abschnitt 5.2.3 beschrieben ist, wurde ein Skript erstellt, welches, wie in Abbildung 5.5 dargestellt, drei Modelle für den ersten Block und ein Modell für den zweiten Block trainiert. Für alle Modelle wurde polynomielle Regression fünften Grades als Verfahren gewählt, da diese, wie bereits erörtert, die besten zu erwartenden Ergebnisse liefern. Die erstellten Ergebnisse aus den Modellen eins bis drei werden hierbei als Eingabe für Modell vier verwendet. Analog zu dem in Abschnitt 6.3.1 verwendeten Prototypen wird mittels k-facher Kreuzvalidierung anhand der CSV-Datei der NMAE in Abhängigkeit zur verwendeten Datenmenge berechnet. In dem Diagramm, das in Abbildung 6.24 zu sehen ist, kann die Qualität einer Vorhersage und die dafür nötige Datenmenge abgelesen werden.

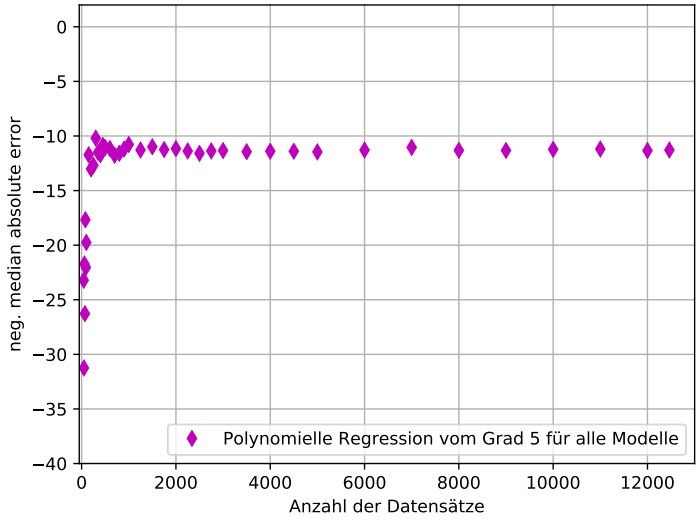


Abbildung 6.24: Der NMAE einer Vorhersage der Anzahl an erstellten raw_image-Dateien pro Sample anhand von sample_quality und sample_score_cutoff bei der Verwendung von 4 Modellen. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Es fällt auf, dass die Qualität einer Vorhersage bei einer ausreichenden Datenmenge ähnlich der zu erwartenden Qualität bei der Verwendung eines einzelnen Modells liegt. Wie in Abbildung 6.25 detaillierter betrachtet werden kann, ist ab 1750 Datensätzen mit einem Fehler von etwa -11 bis -12 zu rechnen. Dies ist auf die Verknüpfung der Fehlerwerte für die Modelle eins bis drei zurückzuführen. Diese erreichten einzeln betrachtet Fehlerwerte zwischen -2 und -0,05 bei 250 Datensätzen, während bei Modell vier ab 35 Datensätzen mit keinem Fehler mehr zu rechnen war. Bei der in Abbildung 6.18

dargestellten Verwendung eines Modells, das beide Blöcke vorhersagt, wurde ab 500 Datensätzen ein Fehlerwert von -9 bis -11 erreicht. Damit sind die zu erwartenden Abweichungen zwischen Vorhersage und dann tatsächlich erstellter Anzahl an Dateien vergleichbar.

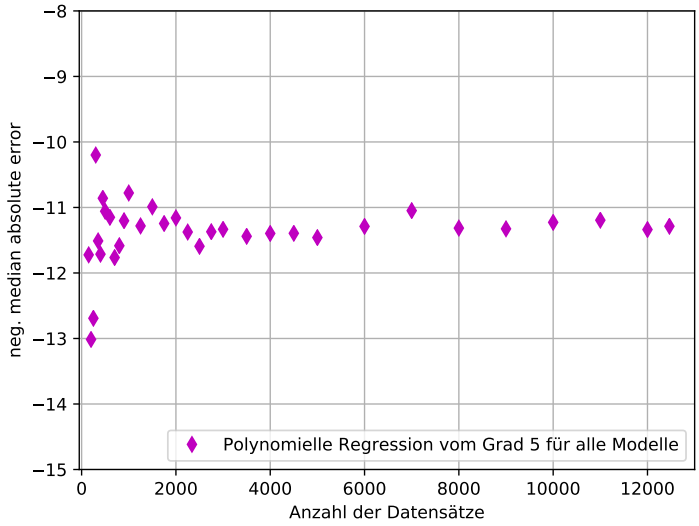


Abbildung 6.25: Der NMAE einer Vorhersage der Anzahl an erstellten raw_image-Dateien pro Sample anhand von sample_quality und sample_score_cutoff bei der Verwendung von 4 Modellen. Der Bildausschnitt wurde der Fragestellung entsprechend angepasst.

Anzumerken ist an dieser Stelle, dass bei diesem Verfahren die beobachtete Über- bzw. Unteranpassung bei geringen Datenmengen deutlich stärker ausgeprägt ist, als bei den anderen diskutierten Diagrammen. Deswegen kann hierzu auch kein Diagramm eingefügt werden, in welchem alle Werte gleichzeitig erkennbar sind (bei beispielsweise 20 verwendeten Datensätzen beträgt der zu erwartende Fehler -334463037).

Anhand des hier vorgestellten Beispiels konnte keine Verbesserung durch die Verwendung von aneinandergereihten Modellen erreicht werden. Es wäre zu untersuchen wie sich die Fehler der einzelnen Modelle aufsummieren und ob mit der Methode im günstigen Fall eine Verbesserung erreicht werden kann. In den hier betrachteten Fall wurden etwa viermal so viele Trainingsdatensätze, wie bei der Verwendung eines Modells, benötigt, um eine Vorhersage von konstanter Qualität zu erhalten. In weiteren Szenarien könnte es aber auch von Interesse sein, die errechneten Zwischenergebnisse zu kennen und deswegen diese Methode zu nutzen.

Die grundlegende Methode hierzu lässt sich verallgemeinern und damit auf andere, komplexere, Workflows übertragen. Indem man den Pfad einer Informationen anhand des von YW erstellten Graphen zurückverfolgt, könnte man beteiligte Blöcke und Variablen identifizieren.

Um redundante und wertlose Informationen auszuschließen und so die Menge an Features zu optimieren, könnte ein Feature Extraction Ansatz (z.B. Principal Component Analysis) verfolgt werden. Bei einer solchen Dimensionsreduktion wird die Menge der ursprünglichen Features auf einen kleinere Menge an Features abgebildet ohne dabei Informationen der ursprünglichen Features zu verlieren.

Für diese lassen sich dann wiederum Modelle zur Vorhersage trainieren, die aneinander gereiht werden können. Damit könnte man anhand der Eingaben die von einem Nutzer stammen, beliebige weitere Vorhersagen unterstützen. Allerdings wäre es hierzu notwendig eine große Menge an Informationen zur Laufzeit zu erfassen, da für alle Zwischenergebnisse auch Informationen vorliegen müssen. Dies ist jedoch kein Bestandteil dieser Arbeit und könnte in weiteren Arbeiten betrachtet werden.

6.4 Zusammenfassende Betrachtung der erstellten YesWorkflow-Erweiterung

In diesem letzten Evaluationsabschnitt soll noch kurz auf die erstellte YW-Erweiterung eingegangen werden. Mithilfe der erstellten Erweiterung können aufgrund beliebiger Werte vor der Ausführung eines Workflows Vorhersagen angeboten werden. Der in diesem Kapitel beschriebene Anwendungsfall, dass ein Nutzer anhand der Zeilenanzahl einer gewählten sample_spreadsheet-Datei und eines Werts für sample_score_cutoff wissen möchte, wie viele raw_image-Dateien voraussichtlich erstellt werden, kann damit umgesetzt werden. Es wird, wenn ausreichend Trainingsdaten vorhanden sind, ein Ergebnis mit maximal 12% Abweichung bereitgestellt. Da dazu allerdings der Workflow bereits mit mindestens 6312 Samples ausgeführt worden sein muss, ist dies aus Nutzersicht nur attraktiv, wenn diese Daten bereits vorliegen.

7 Schlussbetrachtung

In diesem Kapitel wird ein kurzer Überblick über das Vorgehen und die Ergebnisse der Arbeit gegeben. Der umgesetzte Ansatz wird betrachtet und weitere Ideen, für zukünftige Arbeiten, werden erläutert.

7.1 Zusammenfassung

In Rahmen dieser Arbeit wurden zunächst Vorhersagen, die bezüglich einer Workflowausführung bereitgestellt werden können, erläutert. Darauf aufbauend wurde diskutiert, welche Optimierungen damit unterstützt werden können. Es wurde in Bezug auf das betrachte YW-System diskutiert, welche der erfassten Provenance-Informationen hierfür genutzt werden können und welchen Gewinn eine Erfassung zusätzlicher Informationen erwarten lässt. Für eine konstruktive Nutzung von Vorhersagen zur Optimierung eines Workflows wurde ein Ablaufkonzept entwickelt.

Da die Erfassung von Werten, die Variablen zur Laufzeit zugewiesen werden, als für eine Vorhersage am gewinnbringendsten erachtet wurde, wurde der bestehende Prototyp dahingehend erweitert, dass diese Informationen, wenn sie manuell erfasst wurden für Vorhersagen genutzt werden können. Die bestehenden YW-Befehle wurden im Rahmen des Prototypen um einen weiteren Befehl ergänzt, der es erlaubt anhand von erfassten Provenance-Informationen beliebige Werte oder Metriken mittels verschiedener Verfahren vorherzusagen und dann für Optimierungen nutzen zu können.

Da der Gewinn einer Vorhersage und einer daraus resultierenden Optimierung außer Frage steht, wurde im Rahmen der Evaluation betrachtet, wie groß die verfügbare Trainingsdatenmenge für eine verlässliche Vorhersage sein muss. Dazu wurden zwei Skripte, die unterschiedliche Vorgehen zur Vorhersage implementieren, erstellt.

Mit dem Ersten wurde evaluiert wie sich die zu erwartenden Fehler bei der Vorhersage über beliebig viele Blöcke eines Workflows hinweg verhalten. Für ein konkretes Beispiel wurde unter anderem untersucht, ob sich mit den globalen Eingaben des Workflows die voraussichtlich erstellte Anzahl an Dateien vorhersagen lässt, was einem realen Nutzungsszenario entspricht. Dazu wären 480 bereits durchgeführte Durchläufe des Workflow notwendig, um eine Vorhersage mit 12% Abweichung bereitzustellen.

Mit dem Zweiten wurde untersucht, ob sich für dieses Nutzungsszenario eine Verbesserung erzielen lässt, indem für jeden Block Vorhersagen erstellt und diese dann aneinandergereiht werden. Hierbei wurde die etwa gleiche Qualität für eine Vorhersage erreicht, allerdings war die dafür benötigte Menge an Trainingsdaten etwa viermal so groß.

7.2 Weitere Arbeiten

In diesem Abschnitt werden mögliche Erweiterungen und weitere Untersuchungen zu der bisherigen Arbeit erläutert.

- Im Rahmen der prototypischen Erweiterung wurde keine Anpassung der graphischen Repräsentation vorgenommen. In Abschnitt 4.6.3 wurde vorgestellt, wie eine solche Erweiterung in der einfachsten Form aussehen könnte. Um aber die Nutzbarkeit des Systems zu verbessern wäre langfristig eine graphische Benutzeroberfläche, die beispielsweise auf der Graphdarstellung basiert, anzustreben.
- In dieser Arbeit wurden als Verfahren zur Vorhersage ausschließlich Regressionsverfahren betrachtet. Dies hat den Vorteil, dass die Ergebnisse leicht nachvollziehbar sind und die Laufzeit gering ist. Allerdings gibt es komplexere Verfahren, mit denen sich möglicherweise bessere Ergebnisse, hinsichtlich der Vorhersagequalität oder der notwendigen Trainingsdatenmenge, erzielen lassen. Hierzu könnten weitere Verfahren evaluiert werden. Die Einbindung wurde so generisch gehalten, dass hier leicht eine Erweiterung um zusätzliche Verfahren möglich ist. Auch der Einsatz einer Ensemble Learning Methode, ähnlich der Arbeit [HWW16], wäre denkbar.
- Bei der Verwendung komplexerer Verfahren ist mit einer höheren Laufzeit für das Trainieren der Modelle sowie für die Vorhersage zu rechnen. Durch die Integration des in Abschnitt 4.6.2 erläuterten TRAIN-Befehls in den Prototypen, könnte das Training dazu von der eigentlichen Vorhersage getrennt werden. Des Weiteren könnten durch komplexere Verfahren auch nicht-numerische Werte erfasst bzw. vorhergesagt werden.
- In der prototypischen Umsetzung muss der Nutzer das Schreiben der zusätzlich zu erfassenden Daten explizit implementieren. In weiteren Arbeiten wäre ein Ansatz zu entwickeln, wie diese Daten automatisch zum richtigen Zeitpunkt erfasst werden könnten.
- In weiteren Arbeiten wäre zu untersuchen, ob die vorgeschlagene Methode zur Bestimmung relevanter Features durch einen Feature Extraction Ansatz gewinnbringend ergänzt werden kann. Der Aufwand bei der Erfassung von Trainingsdaten könnte damit verringert und die Erstellung der Modelle sowie der Vorhersagen beschleunigt werden.

Literaturverzeichnis

[18] *Duden*. 2018. URL: https://www.duden.de/rechtschreibung/Workflow (zitiert auf S. 15).

[96] *The Workflow Management Coalition Specification*. 1996. URL: http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html (zitiert auf S. 16).

[ABJ06] I. Altintas, O. Barney, E. Jaeger-Frank. „Provenance Collection Support in the Kepler Scientific Workflow System“. In: *Provenance and Annotation of Data*. Springer Berlin Heidelberg, 2006, S. 118–132 (zitiert auf S. 13, 33).

[CFLV12] J. Cheney, A. Finkelstein, B. Ludaescher, S. Vansummeren. „Principles of Provenance (Dagstuhl Seminar 12091)“. In: *Dagstuhl Reports* 2.2 (2012). Hrsg. von J. Cheney, A. Finkelstein, B. Ludaescher, S. Vansummeren, S. 84–113. URL: http://drops.dagstuhl.de/opus/volltexte/2012/3507 (zitiert auf S. 18).

[Chi18] O. Z. Chi Nhan Nguyen. *Machine Learning - kurz & gut*. Dpunkt.Verlag GmbH, 25. Apr. 2018. 183 S. ISBN: 3960090528 (zitiert auf S. 28).

[DVJ+15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, K. Wenger. „Pegasus, a workflow management system for science automation“. In: *Future Generation Computer Systems* 46 (Mai 2015), S. 17–35. URL: https://www.sciencedirect.com/science/article/pii/S0167739X14002015?via%3Dihub (zitiert auf S. 13, 16).

[Enc18] i. Encyclopædia Britannica. *Data Mining - Predictive Modelling*. 2018. URL: https://www.britannica.com/technology/data-mining (besucht am 17. 11. 2018) (zitiert auf S. 25).

[GSK+11] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, M. Reiter. „Conventional Workflow Technology for Scientific Simulation“. In: *Guide to e-Science*. Springer London, 2011. Kap. 11, S. 323–352. URL: https://www.researchgate.net/publication/225825948_Conventional_Workflow_Technology_for_Scientific_Simulation (zitiert auf S. 13, 16, 17).

[HDB17] M. Herschel, R. Diestelkämper, H. Ben Lahmar. „A Survey on Provenance: What for? What Form? What from?“ In: *The VLDB Journal* 26.6 (Dez. 2017), S. 881–906 (zitiert auf S. 18, 19).

[Hol13] S. Holl. „Automated Optimization Methods for Scientific Workflows in e-Science Infrastructures“. Diss. Rheinische Friedrich-Wilhelms-Universität Bonn, 2013 (zitiert auf S. 31, 32).

[HWW16] H. Hiden, S. Woodman, P. Watson. „Prediction of workflow execution time using provenance traces: Practical applications in medical data processing“. In: (Okt. 2016), S. 21–30. URL: https://ieeexplore.ieee.org/document/7870882 (zitiert auf S. 34–36, 96).

[KDTL17] J. Ke, H. Dong, C. Tan, Y. Liang. „PBWA: A Provenance-Based What-If Analysis Approach for Data Mining Processes“. In: *Chinese Journal of Electronics* 26.5 (Sep. 2017), S. 986–992 (zitiert auf S. 32).

[KJ13] M. Kuhn, K. Johnson. *Applied Predictive Modeling*. SpringerLink : Bücher. Springer New York, 2013. ISBN: 9781461468493 (zitiert auf S. 25).

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. „Backpropagation Applied to Handwritten Zip Code Recognition“. In: *Neural Comput.* 1.4 (Dez. 1989), S. 541–551 (zitiert auf S. 26).

[LKM] B. Ludaescher, T. Kolisnik, T. McPhillips. *YesWorkflow Github Repository*. URL: https://github.com/yesworkflow-org (zitiert auf S. 13, 24).

[LWH90] K. J. Lang, A. Waibel, G. E. Hinton. „A time-delay neural network architecture for isolated word recognition“. In: *Neural Networks* 3.1 (1990), S. 23–43 (zitiert auf S. 26).

[LWMB09] B. Ludäscher, M. Weske, T. Mcphillips, S. Bowers. „Scientific Workflows: Business As Usual?“ In: *Proceedings of the 7th International Conference on Business Process Management - Volume 5701*. BPM 2009. Ulm, Germany: Springer-Verlag New York, Inc., 2009. ISBN: 978-3-642-03847-1 (zitiert auf S. 13, 16, 17).

[MBBL15a] T. M. McPhillips, S. Bowers, K. Belhajjame, B. Ludäscher. *Slides: Retrospective Provenance Without a Runtime Provenance Recorder*. 2015. URL: https://github.com/yesworkflow-org/yw-tapp-15-recon/blob/master/TAPP15-presentation/YW-recon-TaPP15-Edinburgh.pdf (zitiert auf S. 23, 60, 63, 68, 85).

[MBBL15b] T. McPhillips, S. Bowers, K. Belhajjame, B. Ludäscher. „Retrospective Provenance Without a Runtime Provenance Recorder“. In: *Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance*. TaPP'15. Edinburgh, Scotland: USENIX Association, 2015, S. 1–1. URL: https://www.usenix.org/system/files/tapp15-mcphillips.pdf (zitiert auf S. 13, 20–22, 25).

[Mit97] T. M. Mitchell. *Machine Learning*. 1. Aufl. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072 (zitiert auf S. 26).

[ML05] J. H. Min, Y.-C. Lee. „Bankruptcy Prediction Using Support Vector Machine with Optimal Choice of Kernel Function Parameters“. In: *Expert Syst. Appl.* 28.4 (Mai 2005), S. 603–614 (zitiert auf S. 26).

[MSK+15] T. M. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. C. Dey, J. Freire, D. N. Huntzinger, C. Jones, D. Koop, P. Missier, M. Schildhauer, C. R. Schwalm, Y. Wei, J. Cheney, M. Bieda, B. Ludäscher. „YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts“. In: *CoRR* (2015) (zitiert auf S. 13, 18, 20, 22).

[RA18] H. Riedwyl, M. Ambühl. *Statistische Auswertungen mit Regressionsprogrammen: Lineare Regression und Verwandtes - Multivariate Statistik - Planung und Auswertung von Versuchen*. Lehr- und Handbücher der Statistik. De Gruyter, 2018. ISBN: 9783486805543 (zitiert auf S. 27).

[RM18] S. Raschka, V. Mirjalili. *Machine Learning mit Python und Scikit-learn und TensorFlow: das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics*. Hrsg. von K. Lorenzen. 2., aktualisierte und erweiterte Auflage. Frechen: mitp, 2018, 577 Seiten. ISBN: 978-3-95845-733-1 (zitiert auf S. 29).

[RSM11] P. Reimann, H. Schwarz, B. Mitschang. „Design, Implementation, and Evaluation of a Tight Integration of Database and Workflow Engines“. Englisch. In: *Journal of Information and Data Management* 2.3 (Oktober 2011), S. 353–368. URL: https://seer.ufmg.br/index.php/jidm/article/view/150/91 (zitiert auf S. 15).

[SF96] A. C. Schwickert, K. Fischer. *Der Geschäftsprozess als formaler Prozess : Definition, Eigenschaften und Arten*. ger. 1996. URL: http://geb.uni-giessen.de/geb/volltexte/2004/1703 (zitiert auf S. 15).

[SS01] B. Scholkopf, A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001. ISBN: 0262194759 (zitiert auf S. 26).

[TDGS07] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer Publishing Company, Incorporated, 2007. ISBN: 1849966192, 9781849966191 (zitiert auf S. 16, 18).

[WFH11] I. H. Witten, E. Frank, M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 0123748569, 9780123748560 (zitiert auf S. 26).

[ZCW+17] Q. Zhang, Y. Cao, Q. Wang, D. Vu, P. Thavasimani, T. McPhillips, P. Missier, P. Slaughter, C. Jones, M. Jones, B. Ludascher. „Revealing the Detailed Lineage of Script Outputs using Hybrid Provenance“. In: 2017 (zitiert auf S. 45).

Alle URLs wurden zuletzt am 20. 11. 2018 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift