

Vorhersage von E/A-Leistung im Hochleistungsrechnen unter der Verwendung von neuronalen Netzen

— Bachelorarbeit —

Arbeitsbereich Wissenschaftliches Rechnen

Fachbereich Informatik

Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Vorgelegt von:	Jan Fabian Schmid
E-Mail-Adresse:	2schmid@informatik.uni-hamburg.de
Matrikelnummer:	6440383
Studiengang:	Computing in Science - SP. Physik
Erstgutachter:	Dr. Julian Kunkel
Zweitgutachter:	Prof. Dr. Thomas Ludwig
Betreuer:	Dr. Julian Kunkel

Hamburg, den 17.12.2015

Abstract

Inhaltsverzeichnis

1 Einleitung	5
1.1 Motivation	5
1.2 Problemstellung	5
1.3 Ziele der Thesis	6
1.4 Strukturierung	6
2 Hintergrund	7
2.1 Ein-/Ausgabe	7
2.2 Hochleistungsrechnen	8
2.3 Maschinelles Lernen	9
2.4 Künstliche Neuronale Netze	11
3 Verwandte Arbeiten	15
3.1 Leistungsvorhersage von Ein-/Ausgabe	15
3.2 Leistungsvorhersage mit Maschinellem Lernen	15
3.3 Neuronale Netze im Hochleistungsrechnen	15
3.4 Kategorisierung (woanders hin?)	15
3.5 Modellierungssansatz gegenüber Regressionsansatz	16
3.6 xxx	16
3.7 Fourier-Assisted Machine Learning	17
3.8 Predicting Performance of Non-Contiguous I/O with Machine Learning	17
4 Gestaltung der Analyse	18
4.1 Modell der Ein-/Ausgabe	18
4.2 Leistungs- und Ausreißervorhersage	18
4.3 Attribut Selektion und Generierung	18
4.4 Modellklassen	18
4.5 Leistungsmetriken	18
5 Implementierung	19
5.1 Untersuchte Modelle	19
5.2 Bagging zur Leistungssteigerung	19
5.3 Anwendung? und Parameter	19
5.4 Verwendete Programmiersprache und Bibliotheken	19
5.5 Aufbau der Benchmark-Tests	19
5.6 Testsystem, Mistral	19

6 Evaluierung	20
6.1 Exploration der Daten	20
6.2 Analyse der Fehlerklassen	25
6.3 Leistungsvorhersage	25
6.3.1 Analyse der trivialen Modelle	25
6.3.2 Analyse der höheren Modelle	39
6.3.3 Betrachtung der neuronalen Netze	39
6.4 Ensemble Lernen	39
7 Fazit	48
Literaturverzeichnis	49
Abbildungsverzeichnis	51
Tabellenverzeichnis	53
Listingverzeichnis	54
Anhänge	55
A Anhangskapitel	56

1. Einleitung

Im folgenden wird zunächst kurz dargelegt mit welcher Problemstellung sich diese Thesis befasst, welches Ziel verfolgt wird, und wie dieser Text im Weiteren aufgebaut sein wird.

1.1. Motivation

Hochleistungsrechnen ist in der Wissenschaft ein Thema mit zunehmender Signifikanz, viele komplexere Fragestellungen, insbesondere in den Naturwissenschaften und der Informatik, können z.B. nur in einer effizienten Weise durch Analyse einer Simulation eines Modells gelöst werden. Der extensive Rechenaufwand solcher Simulationen erfordert, dass Wissenschaftler nicht am eigenen Computer rechnen, sondern hierfür die Dienste eines Hochleistungszentrum in Anspruch nehmen. Die Entwicklung der Computer-Hardware in den vergangenen Jahrzehnten drängte die Hochleistungszentren dazu für den gewünschten Leistungsgewinn in stark parallelisierte Systeme zu investieren. Sodass, statt einzelner sehr schneller Prozessoren heutzutage viele Tausend Prozessoren vernetzt arbeiten. Diese horizontale Leistungssteigerung am Hochleistungsrechner umgeht die technischen Flaschenhälse, welche die Leistung eines einzelnen Prozessors beschränkt, die zur Verfügung stehende Leistung wird dadurch allerdings schwieriger nutzbar. Einerseits liegt dies am großen technischen Aufwand, der zur Vernetzung der Recheneinheiten notwendig ist, andererseits liegt es an der komplexen Programmierung der Software, welche die Parallelität des Rechners berücksichtigt. Insbesondere ist es auch bei der Ein-/Ausgabe (E/A) von erforderlichen Dateien und Ergebnissen des Programms wichtig, dass sie parallel durchgeführt wird. Um die Wissenschaftler beim Programmieren zu unterstützen, gibt es hilfreiche Tools zur Fehlerdiagnostik, Leistungsanalyse, Visualisierung des Programms und der Ergebnisse, sowie zum Parallelisieren von Programmcode. Wünschenswert ist es dabei, wenn diese Tools die Optimierungen möglichst selbstständig durchführen können, sodass der Wissenschaftler sich auf die Funktionalität seines Programms konzentrieren kann, statt sich mit Leistungsoptimierung abzulenken.

1.2. Problemstellung

Im Bereich der Leistungsanalyse stellt sich unter anderem die Problematik der effizienten Ausnutzung der verschiedenen Puffer-Speicher (Caches), wie Arbeitsspeicher, und die direkt auf dem Prozessor liegenden Caches. Autonome Tools zur Optimierung der E/A können versuchen die Dauer von Dateizugriffen vorherzusagen (**warum?**). Diese Arbeit versucht dies mit dem Hilfsmittel neuronaler Netze zu erreichen.

1.3. Ziele der Thesis

Main goal:

A neural network that is reliable in predicting performance of HPC-IO with sufficient quality

Subgoals:

Knowing which data can be provided by SIOX

Identify interesting (data mining) features, that can be derived from the available data
Having an understanding of what kind of neural network is suitable for the task

A measure for adequate quality of an IO-performance predictor

Benchmark of different predictors using different approaches

Implementation of a predictor module for SIOX for online evaluation of application performances

1.4. Strukturierung

Nachdem in diesem Kapitel die Metaebene der Thesis behandelt wird, soll das zweite Kapitel alle nötigen Hintergrundinformationen zum Verstehen der Thematik und der hier angewandten Ansätze liefern. Das dritte Kapitel beschäftigt sich mit bereits vorhandenen Ansätzen und Arbeiten zur Problemstellung. Im vierten Kapitel werden die Konzepte, sowie die Implementierung, der verschiedenen Modelle beschrieben, die als potenzielle Lösungen des Problems entwickelt und getestet wurden. Die Evaluierung der verschiedenen Lösungsansätze wird daraufhin im fünften Kapitel vorgenommen. Im sechsten Kapitel wird untersucht, welche Lösungsansätze für welche Anwendungsfälle am besten geeignet sind, um dann mit diesen Erkenntnissen im siebten Kapitel ein Fazit der Arbeit zu ziehen.

2. Hintergrund

2.1. Ein-/Ausgabe

Wie werden Dateien abgelegt, wie greift man darauf zu. Wie funktioniert Caching? Read ahead, Write ahead? **Was soll hier so hin?**

Als Ein-/Ausgabe (E/A) bezeichnet man jedweden Austausch von Informationen eines Informationssystems mit der Außenwelt (Wiki). Durch Eingaben erhält der Rechner auszuführende Befehle, die Programme und Funktionen mit denen er etwas ausführen soll, sowie die Daten, die verarbeitet werden sollen. Eine Ausgabe des Rechners gibt dem Nutzer Informationen zum inneren Zustand des Systems und er erhält die Ergebnisse seiner Eingaben. Im Kontext dieser Arbeit handelt es sich bei Ein-/Ausgaben um Dateien, die vom Computersystem von einem Datenträger eingelesen und wieder darauf geschrieben werden. Die in den Testsystemen verwendeten Datenträger sind Festplattenlaufwerke, bei diesen werden Informationen durch magnetische Polarisierung von Speicherzellen auf Magnetscheiben gespeichert und durch Abtastung dieser Magnetisierung mit einem Lesekopf ausgelesen. (**Bild einer Festplatte**) Festplatten sind in Datenblöcke (auch Sektoren) unterteilt, diese bilden die kleinste Einheit, die von dem Medium gelesen bzw. darauf geschrieben werden kann. Durch eine eindeutige Adressierung dieser Sektoren kann der Schreib-/Lesekopf durch Aus- und Einfahren, sowie einer Drehung der Magnetscheibe, direkt auf den gewünschten Datenblock zugreifen. Aufgrund des vergleichsweise geringen Durchsatzes, und insbesondere wegen der großen Latenz bei der Durchführung von Festplattenaufrufen, sind zwischen Festplatte und den tatsächlichen Recheneinheiten im Prozessor mehrere Schichten von Speichern zwischengeschaltet. Von der Festplatte gelesene Daten befinden sich zunächst auf dem Arbeitsspeicher und werden dann in die auf dem Prozessor liegenden Pufferspeicher (Caches) geladen. In den verschiedenen Cache-Ebenen geschieht vergleichbares, die Ebenen gehen von kleinen, sehr schnellen zu größeren, langsameren Speichern über, üblich sind hier zwei oder drei Level. Die Zugriffszeit auf eine Datei ist durch diese Struktur stark davon abhängig in welcher Speicherebene es einen Treffer zu den gesuchten Speicheradressen gibt. Wenn die Datei bereits vollständig im Level 1 Cache liegt, ist sie schon nach wenigen Prozessorzyklen geladen, wenn sie aus dem Arbeitsspeicher geholt werden muss, braucht es mehrere hundert Prozessorzyklen. Das lesen von der Festplatte dauert hingegen einige Millionen Zyklen (**Link? duartes.org**). Im wesentlichen kann also unterschieden werden, ob eine Datei "gecached", sich also im Arbeitsspeicher oder Prozessor-Cache befindet, oder noch von der Festplatte geladen werden muss. Verschiedene Caching-Strategien erlauben eine noch effizientere Nutzung der Zwischenspeicher. Ein Beispiel ist das Einschalten der Read-Ahead-Einstellung, dadurch werden weitere Sektoren im Cache zwischengespeichert,

die sich in der unmittelbaren physischen Umgebung auf dem Speichermedium befinden. Falls ein Programm fortlaufend über einen großen Datenbereich arbeitet, weil dort beispielsweise direkt hintereinander Bilddateien eines Fotoalbums befinden, das gerade im Präsentationsmodus gezeigt wird, so ist der Zugriff auf ein weiteres Bild für das E/A-System nicht mehr 'überraschend'. Statt erst in dem Moment des E/A-Aufrufs des nächsten Bildes die erforderlichen Datenblöcke von der Festplatte zu lesen, befinden sich diese nun bereits in einem der vorgeschalteten Zwischenspeicher. Diese Caching-Strategie macht offensichtlich nur Sinn, wenn ein solches sequentielles Zugriffsverhalten eines Programms stattfindet. Wenn aufeinanderfolgende Zugriffe in unterschiedlichen Bereichen der Festplatte Datenblöcke anfordern, würde bei dieser Strategie immerzu zusätzlicher Aufwand für das lesen der umgebenden Daten betrieben, ohne davon einen Nutzen zu haben. Eine weitere Caching-Strategie ist das Wechseln von Write-Through zu Write-Back. Beim Write-Through werden Schreibbefehle, die zunächst nur direkt im Cache umgesetzt werden, direkt in den Arbeitsspeicher übernommen. Cache und Arbeitsspeicher sind so immer in einem widerspruchsfreien Zustand, sie enthalten den gleichen Zustand der Daten, sodass auch nach plötzlicher Löschung des Caches keine Informationen verloren wurden. Diese Sicherheit ist beim Write-Back nicht gegeben, da der geänderte Zustand von Daten zunächst nur im Cache bekannt bleibt. Das Ausschreiben der Änderungen im Arbeitsspeicher geschieht erst in einem günstigen Moment, wenn beispielsweise ansonsten gerade wenige E/A-Zugriffe geschehen, oder wenn ein Zugriff einer anderen Instanz auf die noch nicht geänderten Daten im Arbeitsspeicher stattfindet. Entscheidend für die beste Auswahl der Cache-Strategien sind jeweils die vorherrschenden Bedingungen im System, sowie dessen Benutzungsweise und die gestellten Anforderungen.

2.2. Hochleistungsrechnen

Wofür HPC? c Was zeichnet HPC aus? c Ein-/Ausgabe im HPC c Besondere herausforderungen

Man spricht von Hochleistungsrechnen, wenn der Rechenaufwand eines Programms außerhalb dessen liegt, was ein einzelner Desktop-Computer in vertretbarer Zeit bearbeiten kann. Die im Hochleistungsrechnen verwendeten Computer werden als Superrechner bezeichnet (wiki Superrechner), hierbei handelt es sich heutzutage üblicherweise um Rechnerverbünde (englisch: Cluster) in denen große Anzahlen Prozessoren und Speichermedien zusammengeschaltet werden. Notwendig wird Hochleistungsrechnen in der Forschung für die Simulation von numerischen Modellen aus verschiedensten Bereichen, beispielsweise zur Mehrkörpersimulation in der Astronomie, für Strömungssimulationen oder zur Berechnung von Klimaprognosen. Wichtige Themen im Hochleistungsrechnen sind die Ausnutzung der zur Verfügung stehenden Leistung, das Erkennen und Beheben von Fehlern des genutzten Programmcodes, die Bereitstellung der Rechen- und Speicherkapazitäten, sowie die Energieeffizienz von Hard- und Software. Um den Superrechner gut ausnutzen zu können ist eine effiziente Ein-/Ausgabe bei vielen Anwendungen von großer Wichtigkeit, da die Menge der anfallenden Datenmengen wesentlich stärker ansteigt, als die Geschwindigkeit der Verbindungen zwischen den verschiedenen Speichermedi-

en und -orten. So steigt beispielsweise die Rechenleistung und die Speicherkapazität beim dem Superrechner des Deutschen Klimarechenzentrum (DKRZ) gegenüber dem Vorgänger wesentlich stärker an, als der mögliche Datendurchsatz (**Verweis**). Die in 2.1 beschriebene Ein-/Ausgabe erweitert sich im Rechnerverbund zur parallelen E/A, dies bedeutet einerseits, dass eine Datei von mehreren Prozessen zeitgleich gelesen und bearbeitet werden kann, und andererseits, dass eine Datei über mehrere Festplatten verteilt sein kann. Diese Parallelität hat einen wesentlichen Einfluss auf die Aufgabe der E/A-Leistungsvorhersage, denn statt nur den Aufwand der Arbeitsschritte auf einer einzelnen Festplatte abzuschätzen, müssen hier die verstrickten Zusammenhänge zwischen Netzwerken von Festplatten und Rechnern, den jeweiligen Auslastungen der Komponenten, sowie Priorisierungen bestimmter Aufgaben und Instanzen.

Dies woanders hin? Die Erfassung dieser Informationen wäre sehr aufwendig, sodass dies zur Zeit nicht möglich ist. Die Genauigkeit einer Vorhersage von E/A-Leistung eines parallelen Systems sollte daher systematisch ungenauer sein, als die zu einem seriellen System.

2.3. Maschinelles Lernen

Wozu c Verschiedene Klassen, un -/ supervised c kmeans regresson trees neuronale netze
-> verweis Bagging

Maschinelles Lernen gehört in die Bereiche künstliche Intelligenz und automatisierte Wissensgenerierung. Verfahren dieser Disziplin versuchen durch intelligentes lernen von Mustern Vorhersagen und Entscheidungen, wie die Zuordnung zu einer bestimmten Klasse, zu treffen. Intelligent bedeutet hierbei (beispielweise), dass vorgegebene Informationen nicht schlicht auswendig gelernt und wiedergegeben werden, sondern von diesen Informationen abstrahiert wird, um Gesetzmäßigkeiten innerhalb der Trainingsdaten zu erkennen. Ein Verfahren oder Algorithmus des maschinellen Lernens erstellt ein Modell, welches Aussagen über die Datenmenge trifft, die zu Verfügung stand und eventuell Vorhersagen über Daten trifft, die ihm beim Lernen nicht bekannt waren. Zu unterscheiden sind Verfahren des überwachten und des unüberwachten Lernens. Während beim überwachten Lernen die idealen Ausgaben zu den Eingabedaten vorgegeben werden, wird beim unüberwachten Lernen kein bestimmtes Ergebnis erwartet, stattdessen muss der Algorithmus versuchen den Informationen inhärente Abhängigkeiten und Zusammenhänge zu erkennen. Je nach Art der behandelten Daten sind die Modelle von Verfahren, die Vorhersagen über neue Daten treffen sollen, Klassifikationsmodelle oder Regressionsmodelle. Das Klassifikationsmodell ordnet Datenpunkte einer Klasse von Daten mit ähnlichen Eigenschaften zu, die verschiedenen Klassen müssen beim Erstellen des Modells bekannt sein und Beispiele als Datenpunkte dieser Klasse vorgegeben werden. Es wird also eine qualitative Aussage anhand dieser Einordnung in Gruppen über die einzelnen Datenpunkte getroffen. Dagegen wird beim Regressionsmodell eine quantitative Aussage über die Zusammenhänge zwischen den Attributen der Daten getroffen. Das Modell sagt einen bestimmten Wert für ein Attribut der Datenpunkte voraus. Die Trainingsdaten sind die Informationen, die dem Algorithmus bekannt sind, von diesen versucht er zu

lernen. Falls ein Testdatensatz vorliegt, kann damit anschließend die berechnete Ausgabe zu bisher unbekannten Daten bewertet werden. Sowohl Trainingsdaten, sowie Testdaten enthalten gemessene Werte zu den Attributen, von denen gelernt werden soll (Eingabewerte). Ein Attribut ist eine messbare Größe des Systems, das untersucht wird. Dies könnte beispielsweise bei einem Datensatz über Blumen die Farbe der Blütenblätter sein. Ein Datenpunkt beschreibt eine Instanz des untersuchten Systems (z.B. ein exemplar der Blume), dazu enthält er einen gemessenen Wert zu jedem Attribut. Die Datenpunkte der Trainingsdaten beim überwachten Lernen, enthalten auch die Lösungen "(die gesuchten Ausgabewerte) zu den Attributen, dessen Werte der Algorithmus vorhersagen soll. Diese Informationen werden dann beim Test vorenthalten, sodass durch den Vergleich zwischen tatsächlicher und vorhergesagter Lösung Rückschlüsse auf die Qualität der Vorhersagen gezogen werden können. Zur Anwendung eines Verfahrens des maschinellen Lernens müssen zunächst Kriterien bzw. Leistungsmetriken eingeführt werden, anhand derer die Qualität der Vorhersagen und Entscheidungen gemessen werden können. Einerseits zum Vergleich verschiedener Ansätze, aber insbesondere für den Lernprozess des Algorithmus selbst, damit dieser sozusagen aus seinen Fehlern und Erfolgen lernen kann. Einfache Metriken wären beispielsweise für einen Klassifizierungsalgorithmus der Anteil falsch zugeordneter Datenpunkte. Oft ist es notwendig die zur Verfügung stehenden Daten aufzubereiten, bevor ein maschinelles Lernalgorithmus effizient und korrekt Informationen aus diesen ableiten kann. So weist Alpaydin [Alp10] (Seite 13-15) auf fehlerhafte Daten hin, die durch zufälligen Messfehler oder eine systematisch inkorrekte Messung entstanden sind. Zudem schreibt er, dass Teile der Daten überflüssig sein können, da sie redundant sind oder keine relevanten Informationen enthalten. Problematisch sind auch widersprüchliche (engl. inconsistent) Datenpunkte, hierbei stellen mehrere Datenpunkte dem maschinellen Entscheider die selben Eingabewerte zur Verfügung, sodass dieser sie nicht unterscheiden kann, sie haben jedoch unterschiedliche Ausgabewerte [Alp10] (Seite 14). Mit all diesen Problemen muss, unter Beachtung der Eigenschaften des vorliegenden Datensatzes, bei der Aufbereitung der Daten sinnvoll umgegangen werden. So können Ausreißer bei den Daten aussortiert werden, da diese eventuell durch eine Fehlmessung entstanden sind. Widersprüchliche Datenpunkte können zusammengefasst werden, indem sie zusammen einen neuen Datenpunkt mit eindeutigen Ausgabewerten bilden (dies könnten die Mittelwerte sein).

Unüberwachtes Lernen findet bei der Clusteranalyse statt, Kantardzic beschreibt Clusteranalyse folgendermaßen: Cluster analysis is the formal study of methods and algorithms for natural grouping, or clustering, of objects according to measured or perceived intrinsic characteristics or similarities." [Kan11] (Seite 250). Ein einfaches und anschauliches Beispiel sind Punkte im zweidimensionalen Raum, die hinsichtlich ihrer Position gruppiert werden. (**BILDER SELBER MACHEN HIERZU**) Ein einfacher Clustering-Algorithmus ist der k-Means-Algorithmus. Bei diesem muss zunächst die Anzahl Cluster k festgelegt werden. Die k Mittelwerte der Cluster mit zufälligen Werten initialisiert. Dann wird jeder Datenpunkt dem Cluster zugeordnet, dessen Mittelpunkt seinem am dichtesten ist. Danach werden die Mittelpunkte der Cluster anhand der ihnen zugeordneten Punkte mit einer beliebigen Metrik berechnet und gesetzt. Das Zuordnen der Punkte und Berechnen der Mittelpunkte wird nun solange wiederholt, bis sich keine Änderung der Zuordnung

mehr ergibt. Als Endergebnis eines Cluster-Algorithmus erhält man eine Gruppierung der Datenpunkte in Mengen mit möglichst niedriger Varianz innerhalb der Menge und möglichst großer Varianz zwischen den Mengen.

Die Aggregierung der Vorhersagen gesuchter Ausgabewerte von überwachten Lernalgorithmen ist eine recht simple Möglichkeit ein zuverlässigeres und exakteres Endergebnis zu erhalten. Verschiedene Modelle im Bereich des Ensemble Lernens verfolgen diesen Ansatz. Ein Ensemble ist dabei eine Menge von Vorhersage-Modellen. Nach Kantardzic [Kan11] (Seite 236) kann ein solches Ensemble eine bessere Vorhersage treffen, als eines der in ihm enthaltenen einzelnen Modelle, wenn das Ensemble aus voneinander unabhängigen Modellen besteht, die mit einer Wahrscheinlichkeit größer als 0,5 das richtige Ergebnis vorhersagen. Wobei er sich hier auf Klassifizierende Modelle bezieht, bei einem Regressionsmodell kann eine solche Wahrscheinlichkeit für die exakt richtige Ausgabe nicht erreicht werden, hier würde man eventuell von der Wahrscheinlichkeit sprechen, dass die Vorhersage in einem bestimmten Fehlerrahmen liegt. An einem einfachen Beispiel verdeutlicht Kantardzic [Kan11] (Seite 237) weshalb seine Behauptung gilt. Angenommen man hat 15 voneinander unabhängige Modelle, jeweils mit einer Fehlerrate $\epsilon = 0,3$, das Ensemble bestehend aus diesen 15 Modellen sagt die Klasse für einen Datenpunkt voraus, für die die Mehrheit der Modelle stimmt. Entsprechend liegt die Vorhersage des Ensembles falsch, wenn mehr als die Hälfte der Modelle eine falsche Vorhersage gemacht haben. Somit ergibt sich als Fehlerrate des Ensembles: Der Fehler der einzelnen Modelle von 0,3 konnte also um ein wesentliches Stück verkleinert werden.

$$\epsilon_{ensemble} = \sum 15, i = 8 \binom{n}{k} \epsilon^i (1 - \epsilon)^{15 - i} = 0,05$$

(generell , statt . bei Zahlen)

2.4. Künstliche Neuronale Netze

Data Mining kurz allg. zu lösende Problemklasse lineare Serperabilität, -> Mächtigkeit von NN RNN Vor- und Nachteile

Bei künstlichen neuronalen Netzen, im Folgenden oft nur neuronale Netze genannt, handelt es sich um eine Methode aus dem Bereich des maschinellen Lernens zum Approximieren einer unbekannten Funktion, die der Relation zwischen zwei Datenmengen zugrunde liegt. Die Methode ist inspiriert von biologischen neuronalen Netzen, wie sie im Gehirn vorkommen. Dafür verwenden sie einen statistischen Ansatz, ihre Lösung für das Problem wird zunächst zufällig im Lösungsraum angelegt und dann mit Hilfe eines Gradientenverfahrens optimiert.

Rojas [Roj96] vergleicht neuronale Netze mit einer Black Box, also einem System mit beobachtbarer Ein- und Ausgabe, aber unbekannter innerer Verarbeitung der Informationen. Zur Verwendung eines neuronalen Netzes gibt man dem Netz eine Menge von Eingabevektoren $E \in \mathbb{R}^n$ mit jeweils zugehörigem Ausgabevektor $A \in \mathbb{R}^m$ vor und dieses versucht eine passende Funktion $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ zu finden. Dementsprechend handelt es

sich hierbei um überwachtes Lernen, da eine gewünschte ideale Ausgabe vorgegeben wird.

Ein feedforward-Netz neuronales Netz besteht aus einer Eingabeschicht, einer beliebigen Anzahl verborgener Schichten und einer Ausgabeschicht (siehe 2.1), wobei die Verbindungen aus jeder Schicht jeweils nur in die nächsthöhere Schicht gehen. Die Eingabeschicht besteht meiner vorherigen Definition entsprechend aus n Stellen, an denen die Werte eines Eingabevektors stehen. Jeder Eingabewert wird dann an jedes Neuron in der ersten verborgenen Schicht weitergegeben und dort verrechnet. Die Ergebnisse der Neuronen der verborgenen Schicht werden dann an die nächste Schicht gegeben und so weiter, bis die Ergebnisse der Ausgabeschicht als Ausgabevektor interpretiert werden.

Rekurrente Netze haben die gleiche Struktur, doch es können auch Verbindungen zu zurückliegenden Schichten vorkommen, dadurch ist die Berechnung mehr deterministisch bestimmt. Es müssen Berechnungs- und Determinierungsregeln festgelegt werden. Ein

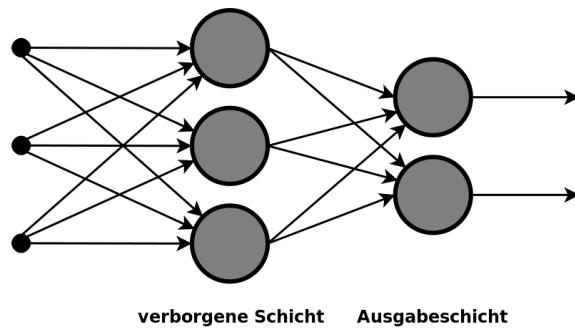


Abbildung 2.1.: Zweischichtiges Netz (wiki)

jedes Neuron rechnet die hineinkommenden Eingabewerte mit einer zur Übertragungskante zugehörigen Gewichtung mit einer Übertragungsfunktion zusammen, die Anzahl der Eingaben ist hierbei unbegrenzt ("unlimited fan-in property" [Roj96]). Die Übertragungsfunktion kann hierbei schlicht die Summe aller gewichteten Eingaben sein. Der errechnete Wert wird als Netzeingabe an die Aktivierungsfunktion gegeben.

Die Aktivierungsfunktion berechnet eventuell mit einem Schwellwert die Aktivierung des Neurons, welche dann an alle verbundenen Neuronen der nächsten Schicht gegeben wird. Die Aktivierungsfunktion kann beispielsweise eine simple Stufenfunktion sein, die allen Netzeingaben kleiner des Schwellwertes eine Null und allen Eingaben größer gleich des Schwellwertes eine Eins zuweist.

Ein Neuron mit der gewichteten Summe aller Eingaben als Übertragungsfunktion und einer Stufenfunktion mit Schwellwert als Aktivierungsfunktion wird von Rojas Perzeptron bezeichnet [Roj96] (Seite 60). Ein feedforward-Netzwerk aus Perzeptrons, in dem jedes Neuron mit allen Neuronen der folgenden Schicht verbunden ist, wird als multilayer perceptron (kurz MLP) bezeichnet. Ein neuronales Netz lernt die Abbildung zwischen den vorgegebenen Paaren von Eingabe- und Ausgabedaten durch Anpassung der Gewichte an

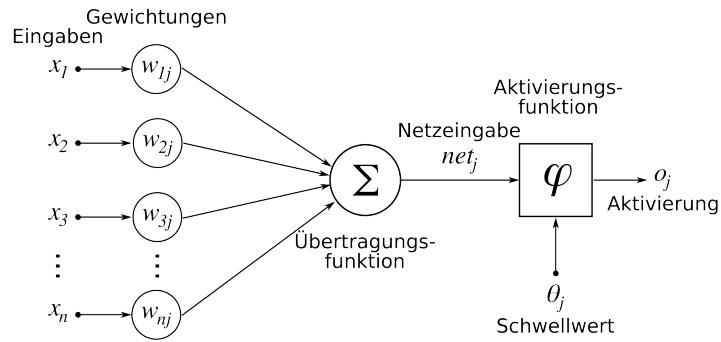


Abbildung 2.2.: Schema eines künstlichen Neurons (wiki)

den Kanten, nachdem es mit zufällig initialisierten Kantengewichten gestartet ist. Diese Anpassung geschieht beim MLP durch Fehlerrückführung (engl. backpropagation). Dabei wird der mittlere quadratische Fehler der berechneten Ausgabe gegenüber der vorgegebenen Ausgabe ermittelt und daraufhin unter Rücksichtnahme auf eine Lernrate mit Hilfe des Gradientenverfahrens minimiert. Vor der Anwendung eines künstlichen neuronalen Netzwerks für ein Problem stellt sich die Frage, ob dieses für das Problem geeignet ist. Dazu gibt es einige interessante mathematische Beweise. Ein einzelnes Perzeptron kann alle linear separierbaren logischen Funktionen exakt approximieren [Roj96] (Seite 62-63). Wobei lineare Separierbarkeit definiert ist als: Zwei Mengen von Punkten A und B .. (auf deutsch?)

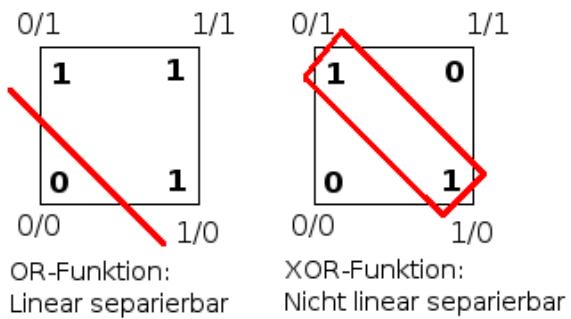


Abbildung 2.3.: Lineare Separierbarkeit von Funktionen (wiki)

Diese Beschränkung gilt allerdings nicht für ein Netzwerk von Neuronen. Bereits ein zweilagiges Netzwerk aus noch simpleren McCulloch-Pitts-Zellen kann jede beliebige logische Funktion berechnen [Roj96] (Seite 37). Für MLPs hat Cybenko [Cyd89] bewiesen, dass sie beliebige kontinuierliche Funktionen auf einer kompakten Teilmenge des euklidischen Raums \mathbb{R}^n approximieren kann. Der entsprechende Satz hierzu ist das "universal approximation theorem".

Zusammenfassung: 2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.

3. Verwandte Arbeiten

Um weniger Umschreiben zu müssen führe ich zunächst zwei Kategorien von Lösungsansätzen ein. Danach erwähne ich kurz einige Arbeiten, dessen Themen sich mit dem dieser Arbeit überschneiden, und danach gehe ich noch etwas detaillierter auf Veröffentlichungen mit hoher Ähnlichkeit und Relevanz ein. Mini-Gliederung was jetzt kommt. Einleitung ist OPTIONAL!!!! In Sektion X steht, in Y steht.

3.1. Leistungsvorhersage von Ein-/Ausgabe

3.2. Leistungsvorhersage mit Maschinellem Lernen

Artificial neural networks for modelling and control of non-linear systems https://books.google.de/books?hl=de&lr=&id=tmTTBwAAQBAJ&oi=fnd&pg=PR9&dq=%22neural+networks%22+storage&ots=KyYA14xY1w&sig=yAQG0zn41xAHccDNaUWYd3L_ywI

3.3. Neuronale Netze im Hochleistungsrechnen

Viele Projekte verwenden Neuronale Netze für die Modellierung vom Gehirn.... <http://worldcomp.org/p2012/PDP3003.pdf>

Machine Learning for Machines: Data-Driven Performance Tuning at Runtime Using Sparse Coding SJ Tarsa - 2015 - dash.harvard.edu ... 83 5.2 Workload Models and Data Collection 85 5.3 Predicting Storage Performance with CART 87 5.3.1 Applying CART to Multi-Tenant Scenarios 87 5.3.2 Variation-Tolerant CART Using Workload Labels

Predicting disk drive failure at a central processing facility using an evolving disk drive failure prediction algorithm

ABER sie werden nur unzureichend eingesetzt für I/O und deswegen braucht es die Arbeit.

10-20 andere literatur

3.4. Kategorisierung (woanders hin?)

Das Problem, die Zugriffszeiten auf Festplatten vorherzusagen, kann im wesentlichen durch zwei verschiedene Ansätze gelöst werden.

Zum einen kann man versuchen ein Modell des Festplattensystems zu erstellen, indem

Hardwaredetails, wie die Rotationsgeschwindigkeit der Platte, die Reaktionszeit und Geschwindigkeit des Lesekopfs, sowie das Zusammenspiel der Komponenten, bekannt sind oder entsprechende Parameter werden durch gezielte Untersuchung approximiert. (**stimmt das? Referenz?**) Mit diesem möglichst exakten Modell können dann Zugriffe simuliert und so deren Aufwand vorhergesagt werden.

Der zweite Ansatz ist auch der dieser Arbeit, es wird von dem eigentlichen Festplattensystem abstrahiert und stattdessen ein mathematisches Modell gesucht, das aus gemessenen Leistungswerten von Festplattenzugriffen die Werte von neuen Zugriffen ableitet. Ich unterscheide daher im Folgenden zwischen dem Modellierungssansatz (in der Fachliteratur etwa als analytic device modeling und simulation modeling bezeichnet), bei dem versucht wird das Festplattensystem nachzubilden und dem Interpolationsansatz, bei dem ein numerisches Modell entwickelt wird. Da hier ohne Wissen über die inneren Zustände des Systems modelliert wird, wird dieser Ansatz auch black-box modeling bezeichnet. (Quelle? Fourier assisted ml)

3.5. Modellierungssansatz gegenüber Regressionsansatz

Die Nachteile von Modellen mit Modellierungssansatz liegen insbesondere daran, dass sie aufwendig zu konfigurieren sind "In fact, one of us (Oldfield) spent several months configuring DiskSim to model an existing device" [CMW⁺13] (S.1) und naturgemäß schnell veralteten, da sie jeweils an spezielle Hardware angepasst sind. Der Vorteil dagegen ist, dass sie bei korrekter Konfiguration sehr präzise sind, wie z.B. hier gezeigt [RW94].

Der Interpolationsansatz ist in der Anwendung einfacher und flexibler, da es sich automatisch an das System anpasst. Dafür erwartet man aufgrund der fehlenden analytischen Einsicht ins System eine etwas schlechtere Präzision.

Für die Anwendung im HPC-Bereich spielt der analytische Ansatz eine untergeordnete Rolle, da hier unterschiedliche Festplattensysteme zusammenarbeiten und stark mit der Netzwerkarchitektur verstrickt sind, sodass eine entsprechende Analyse des Systems zu aufwendig wird. Furthermore, [...] building an accurate model or simulator using white box method cannot be a general solution in serving a variety of very different workloads" [ZLZ⁺10] (S.2, Zeile 20-24).

3.6. xxx

Im HPC Bereich ist die Leistungsanalyse generell ein wichtiger Punkt, so wird beispielsweise das Scheduling-Algorithmen vom Dateisystem simuliert [LFC⁺], hier wird DiskSim [BSSG08] zur Vorhersage der Festplattenzugriffszeiten genutzt, dabei nutzt DiskSim einen Modellierungssansatz (analytical simulation). Eine weitere Arbeit, in der ein Modellierungsansatz genutzt wird stammt von Lebrecht et al. [LDK09].

Bei Arbeiten, in denen ein Interpolationsansatz genutzt wird, werden verschiedene Data-Mining bzw. stochastischen Methoden angewandt, beispielsweise eine Kombination aus regression trees und support vector machines [DLZC12], bagging classification und re-

gression trees [ZLZ⁺10]. Verschiedene statistische Methoden werden von Kelly et al. untersucht [KCGK04].

3.7. Fourier-Assisted Machine Learning

Adam Crume et al. [CMW⁺13] gehen davon aus, dass der entscheidende Faktor bei der Vorhersage von Zugriffszeiten in der Erkennung von periodischen Mustern liegt. Diese Annahme ist für eine einzelne Festplatte gerechtfertigt, da man die Zugriffzeit grob in zwei Teile aufteilen kann, zum einen die Bewegung des Lesekopfs auf die richtige Spur und die Bewegung des Kopfes entlang der Spur zum richtigen Punkt, auch wenn diese beiden Bewegungen in der Realität überlappen. Jede Festplattenspur hat entsprechend des Radius eine andere Periodendauer.

Durch eine Fourier Analyse finden sie die Hauptfrequenzen heraus und können diese dann nutzen, um mit einem neuronalen Netz Vorhersagen zu treffen. Eine Schwierigkeit in dieser Arbeit ist unter anderem, dass durch die große Anzahl Perioden nur ein kleiner Ausschnitt der Festplatte in seiner Gesamtheit, also alle möglichen Paare von Ausgangs- und Endspuren, untersucht werden kann. Die Priorität solcher Frequenzen für die exakte Leistungsvorhersage, ist im komplexen HPC-E/A-System zunächst einmal nicht zu erwarten, müsste allerdings untersucht werden.

3.8. Predicting Performance of Non-Contiguous I/O with Machine Learning

Direkt aus dem Bereich des Hochleistungsrechnens stammt die Arbeit von Kunkel et. al [KZB15]. Hier wird versucht mit Hilfe von decision trees die performantesten Parameter für nicht zusammenhängende Zugriffe auf Dateien durch ROMIO, einer Implementierung von MPI-2 I/O, zu finden. Dabei sagen sie mit den decision trees die Performance für die verschiedenen Parameter auf den Daten voraus, sodass sie anhand dieser Vorhersagen die besten Parameter finden. (**habe ich das richtig verstanden?**) Mit dieser Methode haben sie es geschafft zufriedenstellende Ergebnisse zu erzielen. Die Simplizität von decision trees, beispielsweise können diese nur Entscheidungen als eine Aneinanderreihung von linearen Separationen des Werteraums treffen, lässt vermuten, dass mit Hilfe von komplexeren Data Mining - Methoden, wie neuronalen Netzen noch bessere Ergebnisse erzielt werden könnten. Hier findet sich ein potenzielles Anwendungsgebiet der Ergebnisse dieser Bachelorarbeit.

Zusammenfassung: 2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.

4. Gestaltung der Analyse

test

4.1. Modell der Ein-/Ausgabe

Zeit zum Bewegen des Lesekopfes + Zeit zum rotieren + lese-/schreibzeit + Aufrufprozedur mit festem Beitrag; $t = t_{\text{Network}} + t_{\text{HDD}} + t_{\text{Mem}}$
mit $t_{\text{HDD}}(\text{deltaOffset}, \text{sizeInBlocks}) = t_{\text{Bus}}(\text{size}) + f(\text{deltaOffset})$
track-to-track-seek-time, rotational time Welche Abhängigkeiten gibt es, oder könnte es geben? Z.B. Umso größer die Size, desto länger die Lese-/Schreibzeit, umso größer Offset, desto größer die Seekzeit

4.2. Leistungs- und Ausreißervorhersage

Die Beiden Dinge, die ich versuche vorherzusagen. Warum? Unterschiede. **Macht es Sinn zu versuchen Ausreißer vorherzusagen?**

4.3. Attribut Selektion und Generierung

Korrelationen untersuchen, neues Attribute generieren (wesentlich für die verschiedenen Modelle)

4.4. Modellklassen

Zeitreihenanalyse, Aggregierung, Hybride

4.5. Leistungsmetriken

Welche Leistungsmerkmale werden warum untersucht?

Zusammenfassung: 2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.

5. Implementierung

test

5.1. Untersuchte Modelle

jeweils Beschreiben, wie die Daten hierfür vorbereitet werden mussten, was die Idee davon ist (warum macht es Sinn, dies zu testen), (wie soll die Performance hier untersucht werden?, Design?)

5.2. Bagging zur Leistungssteigerung

Konzept und Anwendung

5.3. Anwendung? und Parameter

Wie wird mit Ausreißern umgegangen, wie werden sie definiert, Cross-Validation, Anzahl berechneter Netze pro Modell, threshold, Größe der Netze, lern- und fehler-funktionen,

5.4. Verwendete Programmiersprache und Bibliotheken

5.5. Aufbau der Benchmark-Tests

5.6. Testsystem, Mistral

Speziell die Hardware von Mistral, von hier kommen die Benchmarks Zur Evaluation verschieben: Testsystem

Zusammenfassung: 2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.

6. Evaluierung

test

6.1. Exploration der Daten

Zunächst werde ich die vier Datensätze genauer betrachten. Dazu eignet es sich einige Metainformationen zu sammeln. In der Tabelle 6.1 sind für alle vier Datensätze jeweils für die drei Attribute Dauer, Größe und Delta-Abstand der minimale Wert, der Wert des ersten Quartils, der Median, das arithmetische Mittel, der Wert des dritten Quartils, der maximale Wert und die Korrelation zu Dauer. Alle Datensätze bestehen aus 200.000 Messungen.

Das Attribut OpTyp kann nur sinnvoll über der Vereinigung von cached-off0-seq-R und cached-off0-seq-W bzw. cached-off0-rnd-R und cached-off0-rnd-W betrachtet werden.

6.2

Nachdem nun ein grobes Verständnis für die vorliegenden Daten erlangt worden ist, folgt eine Betrachtung der tatsächlichen Messungen in Zeitreihe. Um den wesentlichen Teil der Punkte besser erkennen zu können, wurden in den Graphen die obersten 1%, also die langsamsten Datenpunkte, abgeschnitten.

Wenn die Messpunkte nach der Laufzeit sortiert sind, kann die Verteilung Daten besser betrachtet werden. Die logarithmische Skalierung der Y-Achse entzerrt die Punkte.

Als Ausreißer werden für alle **Attribut-Sets** die Messungen mit den 10% kürzesten und längsten Laufzeiten behandelt. Die Verteilung der Ausreißer ist in 6.3 erkennen. Am Verhalten der Ausreißer lässt sich gut unterscheiden, ob es sich um den sequentiellen oder den randomisierten Datensatz handelt. Bei cached-off0-seq befinden sich die Ausreißer gerade an den oberen und unteren Enden einer Ansammlung von Punkten, da alle Punkte innerhalb der Ansammlung zu einem Attribut-Set gehören muss sich gerade dieses Bild ergeben. Dagegen sind die roten Punkte bei cached-off0-rnd unregelmäßiger verteilt. Da die Ausreißer als zweites in den Graphen gezeichnet wurden, überdecken sie die grauen und erscheinen daher präsenter. (**Sind die Ausreißer hier eher ein Zeichen dafür, dass das System in den Häufungspunkten von Ausreißern besonders schnell oder langsam agiert hat, oder ist dies durch die Art des Benchmarks bedingt?**)

Bei der Betrachtung der Metainformationen wurde festgestellt, dass die Zugriffsgröße sehr stark mit der Laufzeit eines Messpunktes korreliert. Um eine Vorstellung davon zu entwickeln, wie diese beiden Größen miteinander zusammenhängen, habe ich die

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelati
Dauer	6.2e-06	6.9e-06	3.6e-05	4.9e-04	6.2e-04	3.0e-02	1.000
Größe	1.0e+00	2.0e+02	4.1e+04	5.8e+05	5.2e+05	1.7e+07	0.970
Delta-Abstand	-1.1e+10	1.6e+01	1.6e+04	8.6e+03	5.2e+05	1.7e+07	0.019

(a) cached-off0-seq-R

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelati
Dauer	7.6e-06	8.3e-06	9.1e-06	3.0e-04	5.7e-05	3.0e-02	1.000
Größe	1.0e+00	1.6e+01	6.4e+02	3.1e+05	2.9e+04	1.7e+07	0.984
Delta-Abstand	-1.1e+10	1.6e+01	2.6e+02	5.2e+03	1.6e+04	1.7e+07	0.023

(b) cached-off0-seq-W

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelati
Dauer	6.6e-06	2.6e-05	9.8e-04	5.6e-03	1.0e-02	2.9e-01	1.000
Größe	1.0e+00	6.4e+01	1.2e+04	8.6e+05	6.6e+05	8.4e+06	0.299
Delta-Abstand	-1.1e+10	-3.2e+09	-9.8e+06	-7.8e+05	3.1e+09	1.1e+10	0.045

(c) cached-off0-rnd-R

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelati
Dauer	1.1e-05	3.7e-04	4.6e-04	3.2e-03	2.2e-03	1.4e+00	1.00000
Größe	1.0e+00	7.2e+03	4.1e+04	1.3e+06	1.3e+06	8.4e+06	0.14976
Delta-Abstand	-1.1e+10	-3.2e+09	-9.8e+06	-1.2e+06	3.1e+09	1.1e+10	-0.00033

(d) cached-off0-rnd-W

Tabelle 6.1.: Metainformationen über die Datensätze

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelation
OpTyp	1	1	1.5	1.5	2	2	-0.077

(a) cached-off0-seq

Attribut	Min.	1. Quartil	Median	Arith. Mittel	3. Quartil	Max.	Korrelation
OpTyp	1	1	1.5	1.5	2	2	-0.075

(b) cached-off0-rnd

Tabelle 6.2.: Metainformationen über OpTyp

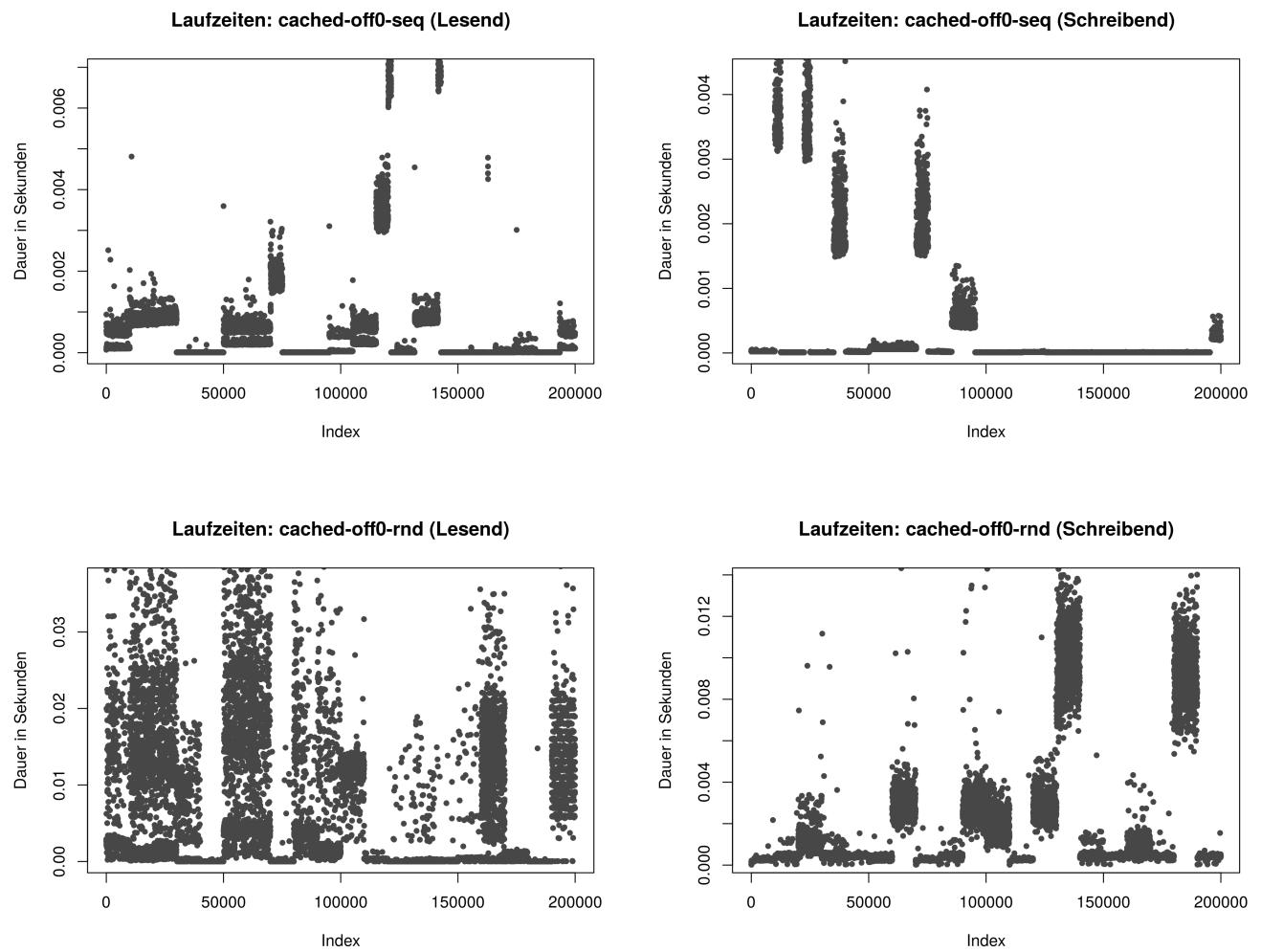


Abbildung 6.1.: Messungen der Laufzeiten als Zeitreihe dargestellt

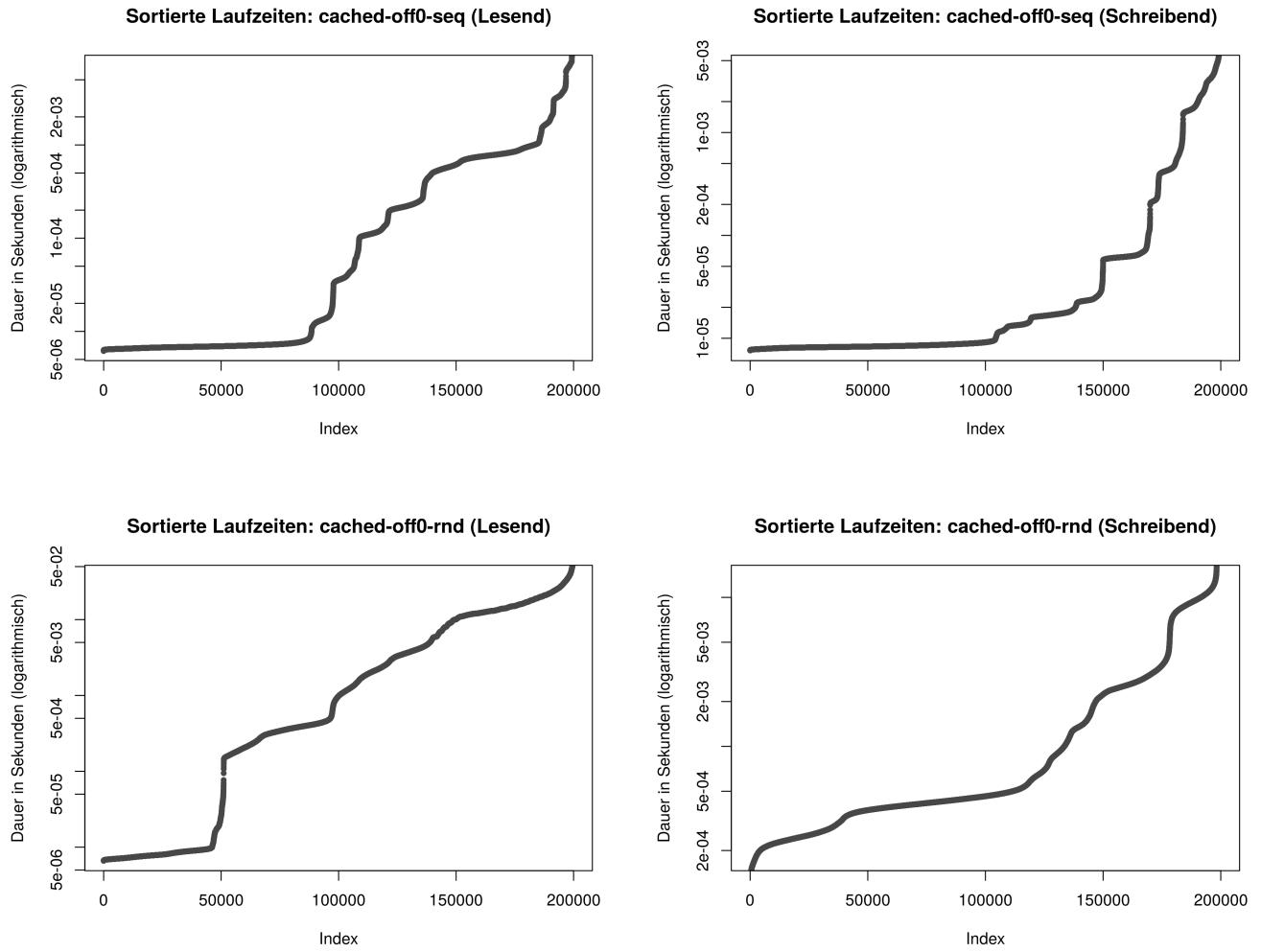


Abbildung 6.2.: Messungen der Laufzeiten sortiert dargestellt (logarithmische Y-Achse)

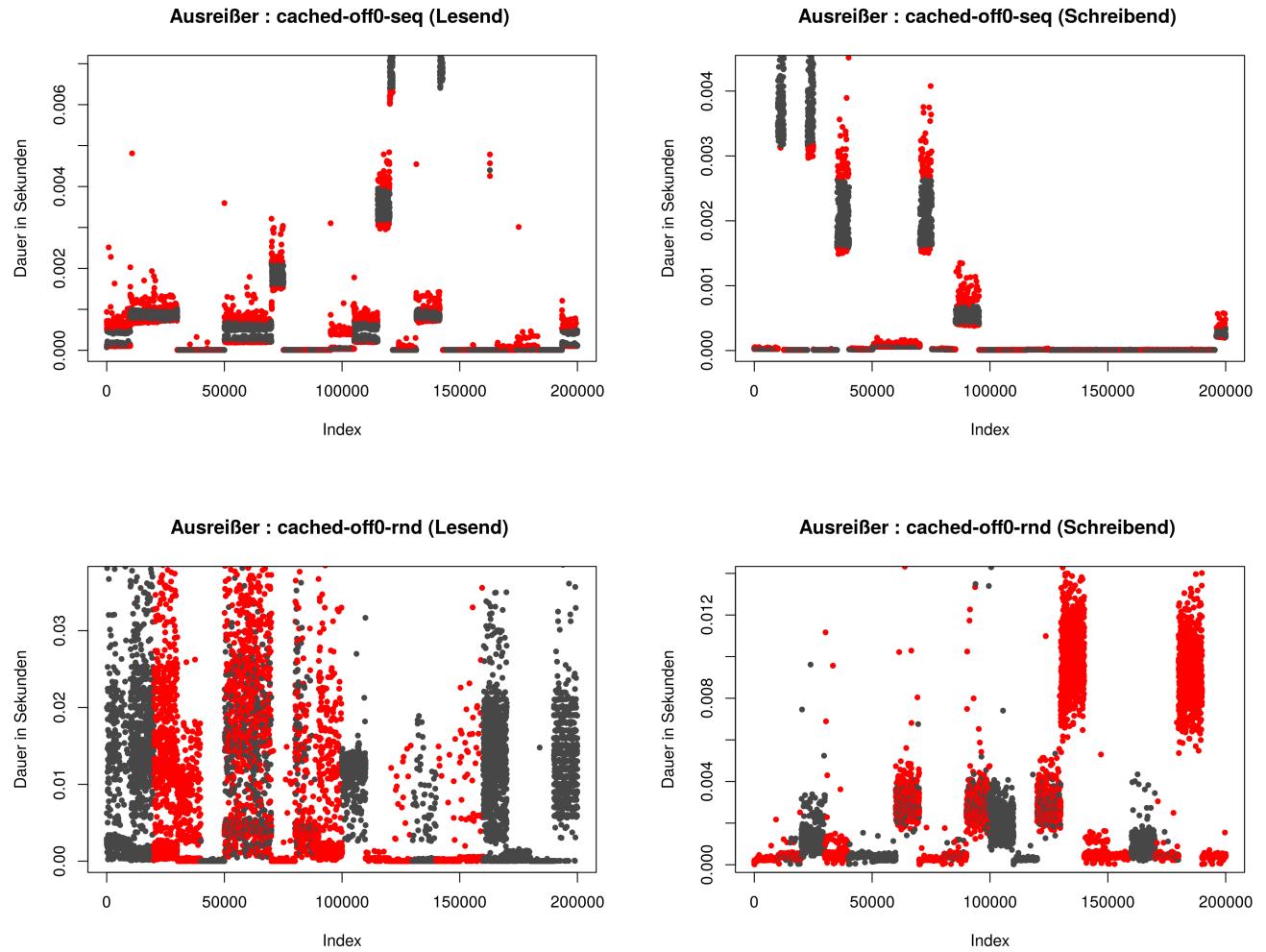


Abbildung 6.3.: Ausreißer sind in rot dargestellt, sie überdecken die grauen Punkte

Messdaten nach der Zugriffsgröße sortiert und ihre Laufzeit aufgetragen. 6.4 Die Korrelation der beiden Größen ist deutlich erkennbar, die Laufzeiten nehmen im Mittel zu. Doch es ist auch zu erkennen, insbesondere bei cached-off0-rnd-R, dass die Laufzeiten aufgrund der Streuung von weiteren Faktoren abhängen müssen. Die Verteilung der Messpunkte einer Zugriffsgröße in zwei bis drei deutlich zu unterscheidende Gruppen deutete darauf hin, dass Messungen innerhalb einer vom E/A-System Gruppe ähnlich behandelt wurden.

Um die unterschiedlichen Laufzeiten innerhalb einer Zugriffsgröße zu untersuchen, habe ich für die Größen 1KB, 16384KB und 2097152KB (diese Zugriffsgrößen kommen in allen vier Datensätzen vor) alle Messungen betrachtet.

Um die Messdaten noch detaillierter darzustellen, müssen kleine Ausschnitte herausgegriffen werden. Zunächst betrachte ich die ersten 250 Messungen. 6.8 Auf den ersten Blick scheint bei cached-off0-seq-W eine gewisse Periodizität ersichtlich (etwa alle 45 Messungen gibt es einen größeren Sprung, alternierend sind die Laufzeiten jeweils etwas langsamer und schneller). Nach genau 123 Punkten scheint sich das Muster zu wiederholen, dort befindet sich der zweitgrößte Messwert. Wenn man nun als erstes simples Modell eine Fortführung der augenscheinlichen Periodizität der ersten 123 Messpunkten betrachtet, so erkannt man doch, dass der Verlauf recht unregelmäßig ist. 6.9

Bei den anderen Graphen kann keine so simple Periodizität vermutet werden.

Eine weitere Detailbetrachtung mache ich bei den Messungen 30.001 bis 30.250. 6.10 Hier scheint eine Periodizität bei cached-off0-seq-R vorhanden zu sein. Und wenn eine Überlappung wie zuvor durchgeführt wird (diesmal nach den ersten 129 Messungen), so erkennt man, dass dieses simple Modell die Ausreißer für diesen kleinen Ausschnitt exakt vorhersagen kann. 6.11

Im Allgemeinen kann dies jedoch offensichtlich nicht funktionieren. Doch unsere Annahme **ABSCHNITT** einer gewissen Periodizität in der Leistung des E/A-Systems ist scheinbar gerechtfertigt. Ein Modell das versucht diese auszunutzen muss eine komplexere Methode als schlichtes Übertragen vorheriger Leistungswerte haben, ansonsten kann es wohl nur in äußerst eingeschränktem Maße korrekte Leistung vorhersagen.

6.2. Analyse der Fehlerklassen

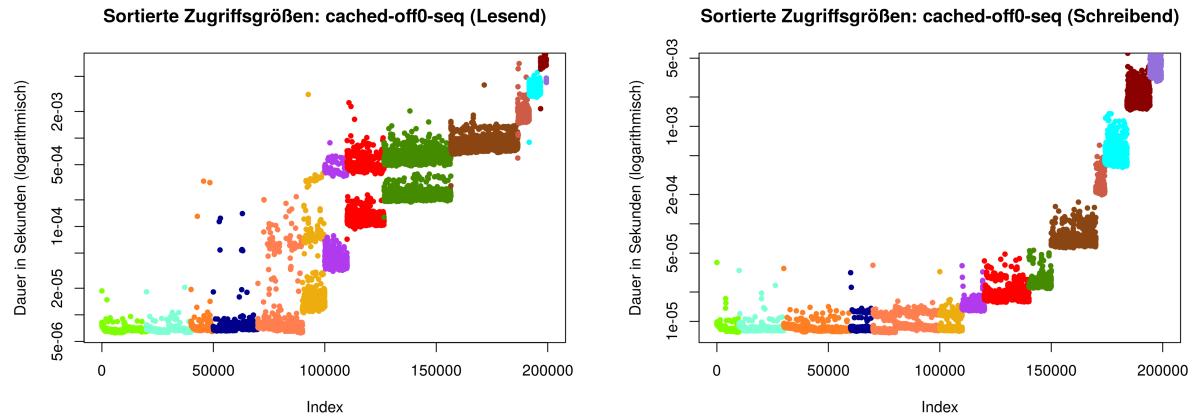
6.3. Leistungsvorhersage

Zu jedem Modell schreiben, welche Attribute und Parameter sich als geeignet herausgestellt haben. Ergebnisse anhand von Graphen veranschaulichen.

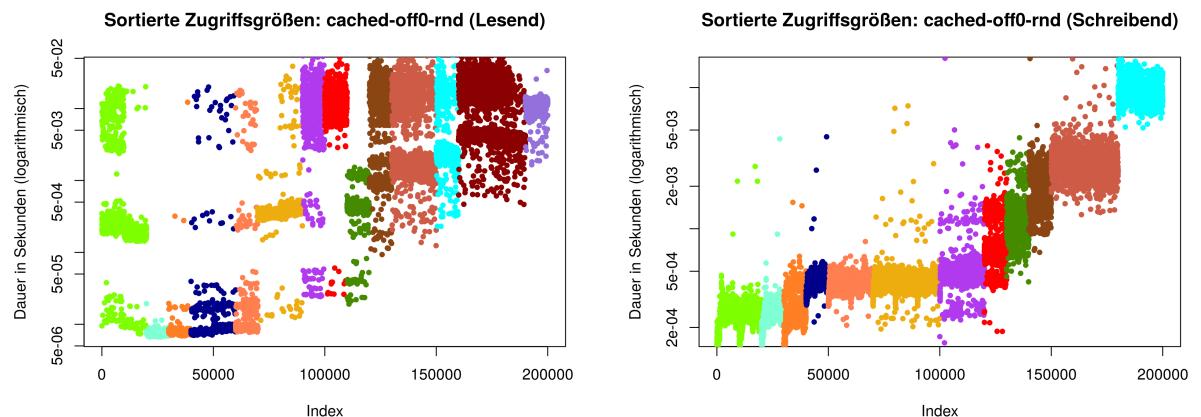
Ergebnisse für verschiedene Fälle vergleichen, seq,rnd,mistral vs pc?
Jeweils schreiben, ob bagging etwas bringt.

6.3.1. Analyse der trivialen Modelle

Die Ergebnisse der in **ABSCHNITT** beschriebenen trivialen Modelle sind in 6.3 zu sehen. Eine positive Beobachtung in den Ergebnissen auf cached-off0-seq ist zunächst, dass



- (a) Von links nach rechts in KB: 1, 4, 16, 256, 1024, 16384, 65536, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216
- (b) Von links nach rechts in KB: 1, 4, 16, 64, 256, 1024, 4096, 8192, 16384, 65536, 262144, 524288, 1048576, 2097152, 4194304, 16777216



- (c) Von links nach rechts in KB: 1, 4, 16, 64, 256, 1024, 4096, 8192, 16384, 65536, 262144, 524288, 1048576, 2097152, 8388608
- (d) Von links nach rechts in KB: 1, 4, 1024, 4096, 8192, 16384, 65536, 262144, 524288, 1048576, 2097152, 8388608

Abbildung 6.4.: Messungen der Laufzeiten nach Zugriffsgröße sortiert dargestellt (logarithmische Y-Achse)

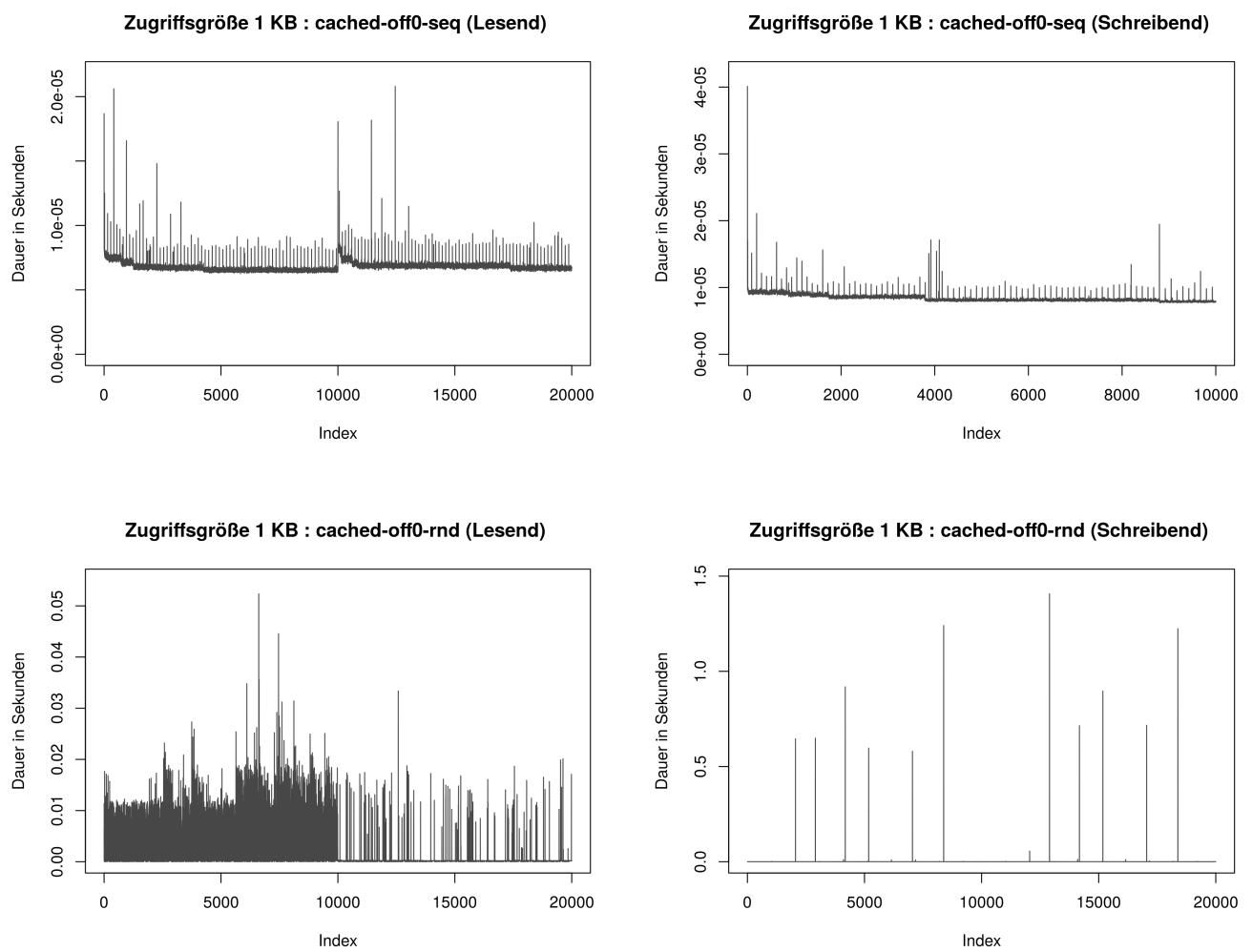


Abbildung 6.5.: Detailbetrachtung aller Messungen mit Zugriffsgröße 1KB

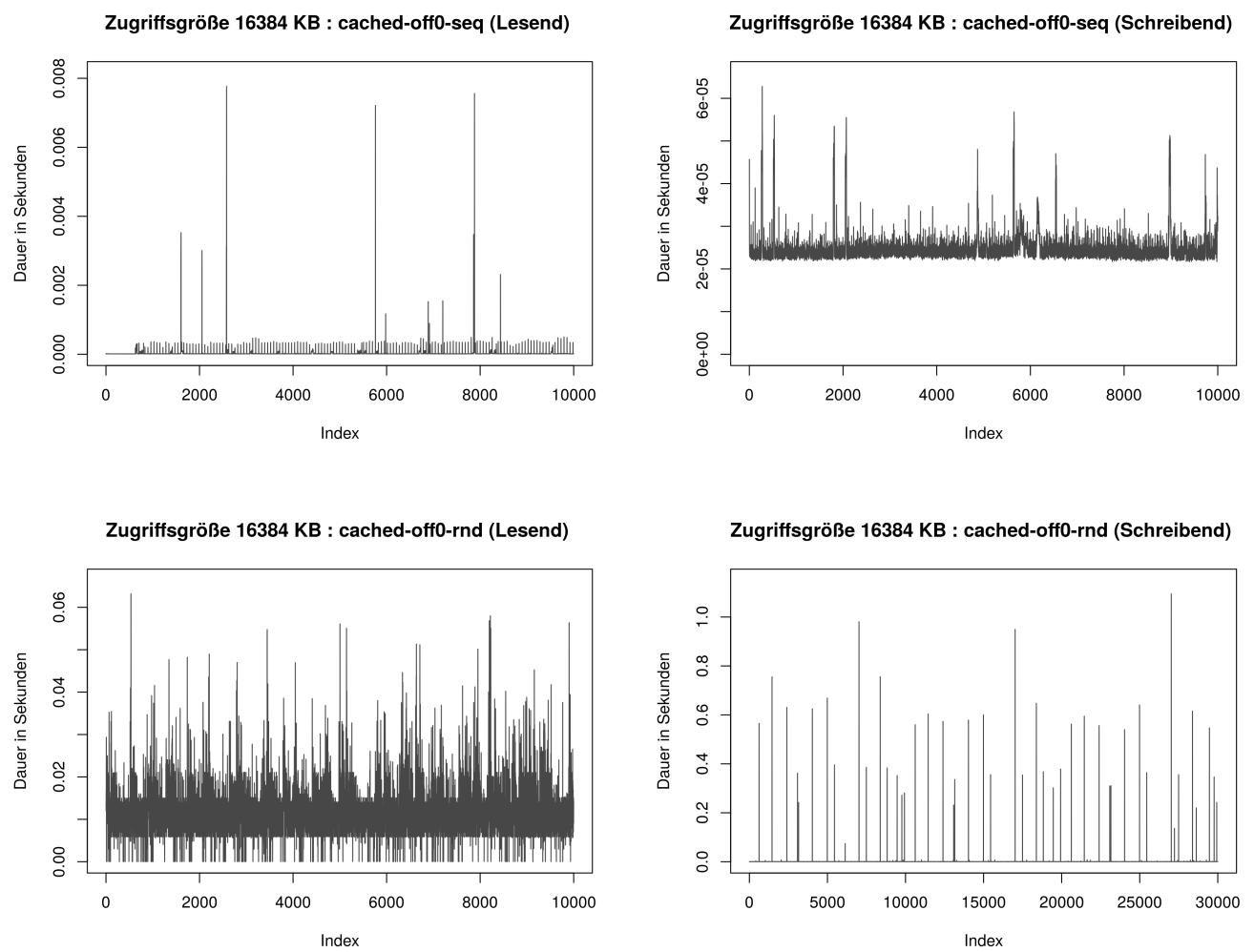


Abbildung 6.6.: Detailbetrachtung aller Messungen mit Zugriffsgröße 16384KB

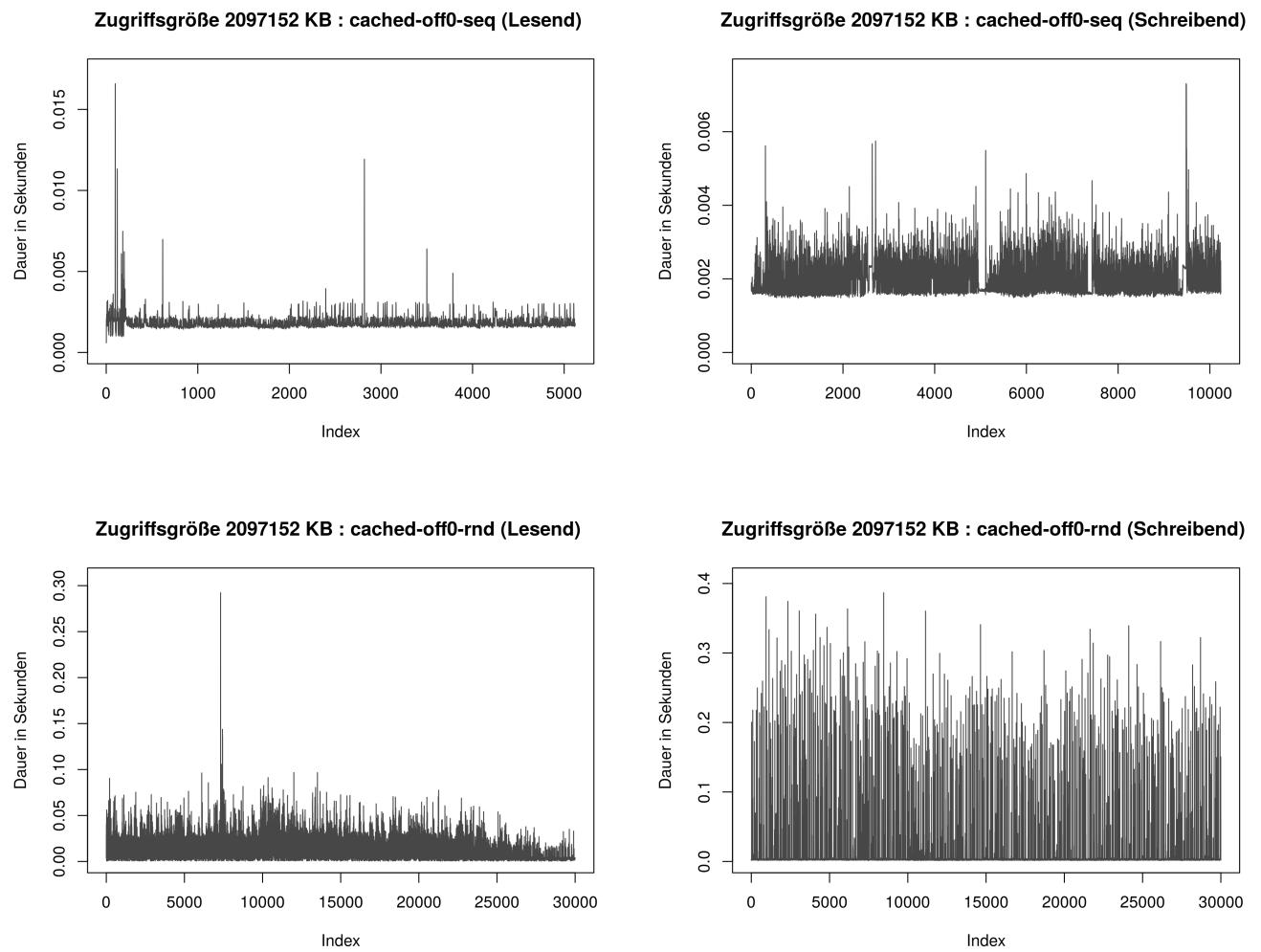


Abbildung 6.7.: Detailbetrachtung aller Messungen mit Zugriffsgröße 2097152KB

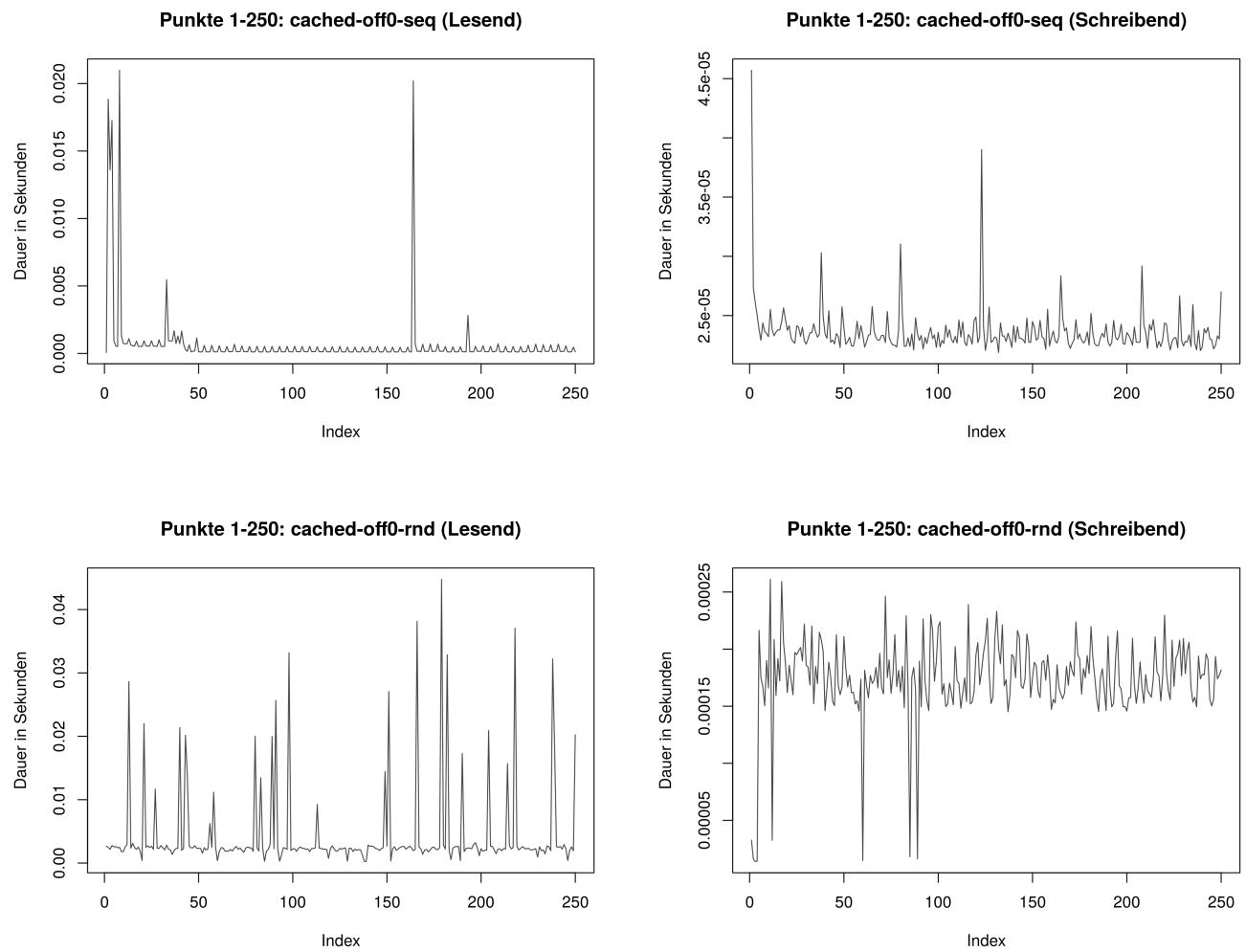


Abbildung 6.8.: Detailbetrachtung der ersten 250 Messungen

Punkte 1-250: cached-off0-seq (Schreibend)

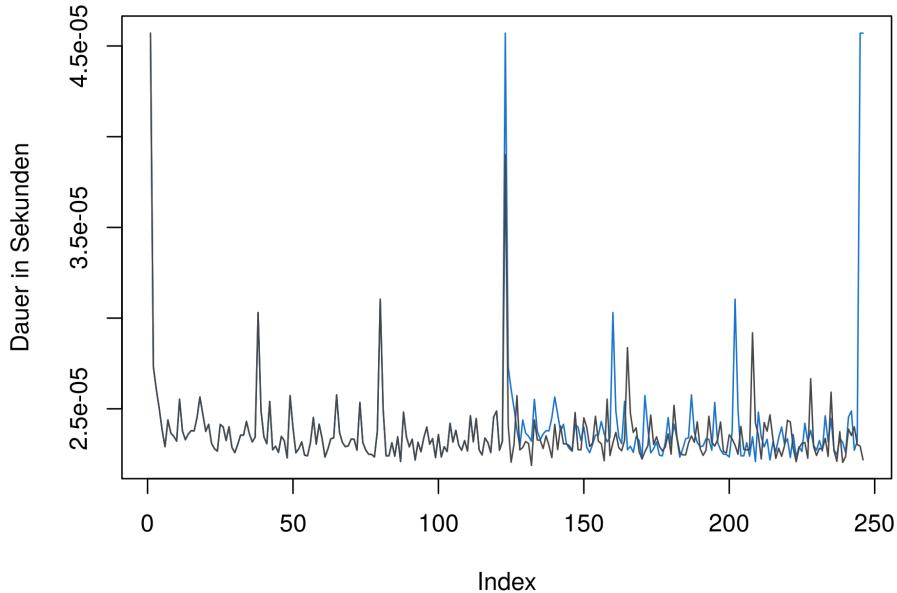


Abbildung 6.9.: Wiederholung der ersten Werte, als erstes einfaches Modell

alle Metriken einen ähnlichen Trend der Modelle aufzeichnen. Das gilt im Wesentlichen auch für cached-off0-rnd, jedoch hat das Modell Median agg einen sehr hohen maximalen Fehler. Dies führt entsprechend zu einem vergleichsweise hohen RMQA-Wert, obwohl der relative arithmetische Fehler noch recht gering ist.

Wie erwartet (**ABSCHNITT**) sind die Vorhersagen auf dem randomisierten Datensatz schwieriger zu machen, sodass die Fehlerwerte höher sind. Dies gilt insbesondere für die Modelle, die auf linearer Regression basieren, hier sind die Werte 30 – 140 mal höher. Eine Ausnahme ist Median agg LinReg-Fehlerklasse", das Modell kann eine weiterhin sehr gute mittlere Vorhersage treffen, nur die Ausreißer nach oben scheinen zugenommen zu haben, sodass RMax und RMQA höher als bei sequentiellen Datensatz sind.

Allgemein sind die Ergebnisse bereits überraschend gut. Ein durchschnittlicher Fehler von 5% auf cached-off0-seq bzw. 7 – 12% auf cached-off0-rnd durch die Modelle, die Fehlerklassen ausnutzen, deutet darauf hin, dass die Vorgänge im E/A-System im Wesentlichen korrekt repräsentiert werden. Ob über eine größere Messreihe ein geringerer durchschnittlicher Fehler als 5% überhaupt erreicht werden kann, ist fraglich, da das Messrauschen durch unvorhersehbare Ereignisse sowohl auf System-, als auch Bauteilebene, in diesen Bereichen eine zu große Bedeutung zukommt.

Die Modelle mit linearer Regression führen auf cached-off0-seq auch bereits zu zufriedenstellenden Ergebnissen, während sie auf cached-off0-rnd kaum sinnvolle Vorhersagen machen zu scheinen. Die Modelle lassen sich für die Datensätze jeweils in drei Gruppen (gut, mittel, schlecht) unterteilen. Auf beiden Datensätzen führen die beiden Modelle

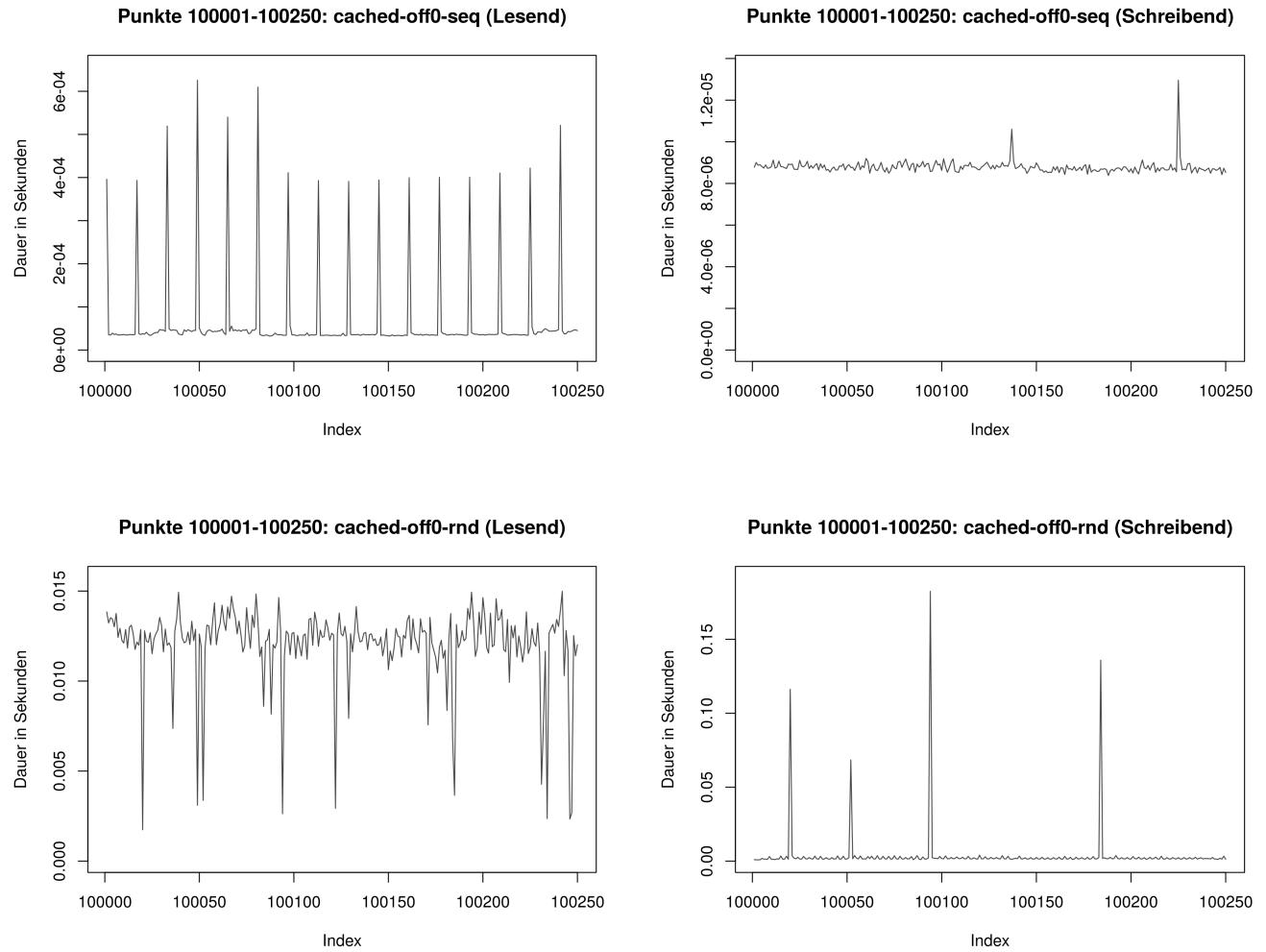


Abbildung 6.10.: Detailbetrachtung der Messungen 100001 bis 100250

Punkte 100001-100250: cached-off0-seq (Lesend)

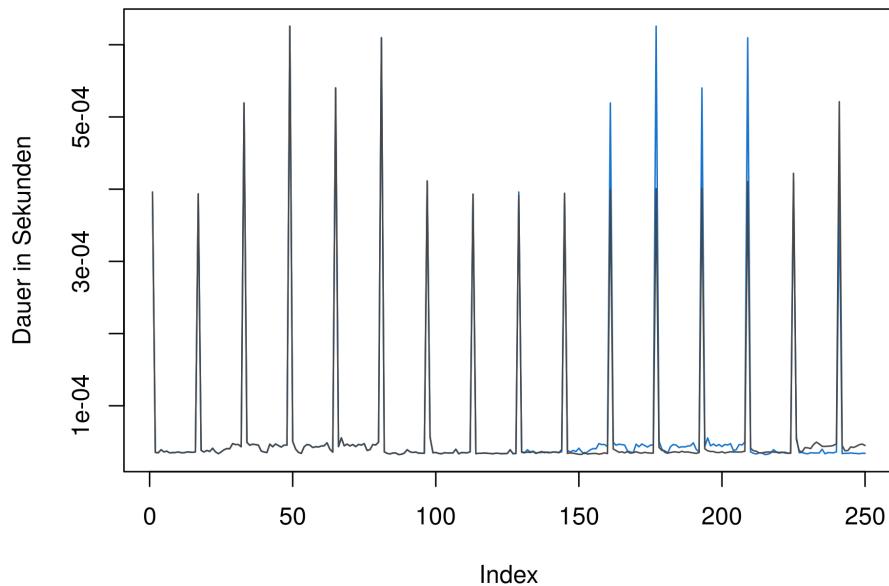


Abbildung 6.11.: Wiederholung der ersten Werte, als erstes einfaches Modell

mit Fehlerklasse zu einer guten Vorhersage. Median agg" hat jeweils eine mittel gute Leistung, während "Durchschnitt" jeweils schlecht ist. Interessant ist, dass Modelle mit linearer Regression auf dem sequentiellen Datensatz eine zufriedenstellende mittel gute Vorhersage macht, während die Vorhersage auf dem randomisierten Messungen kaum besser als des Durchschnitts-Modell ist.

Das Modell, das immer die Durchschnittsdauer vorhersagt ist als absolute untere Grenze für angewendete Modelle zu verstehen, da hier praktisch keine Expertise über die Daten eingeht. Einem Modell mit schlechterer Leistung, als das Durchschnitts-Modell müsste unterstellt werden zufällige Vorhersagen zu machen. Tatsächlich erreicht dieses Modell die schlechtesten Ergebnisse in beiden Testfällen. Die Ansätze der rechenintensiven halten diesem Anspruch also stand und sind somit gerechtfertigt.

Aufgrund des geringen Nutzen der Attribute Abstand und OpTyp für die lineare Regression wird im Folgenden nur das Modell über die Zugriffsgröße betrachtet.

MAF → Mittlerer arithmetischer absoluter Fehler

RMAF → Relativer mittlerer arithmetischer absoluter Fehler

RMQA → Relative mittlere quadratische absolute Abweichung

Q3 → oberes Quartil des absoluten relativen Fehlers

Max → Maximaler relativer absoluter Fehler

Um genauer zu verstehen, wie die trivialen Modelle die Messdaten annähern, ist es

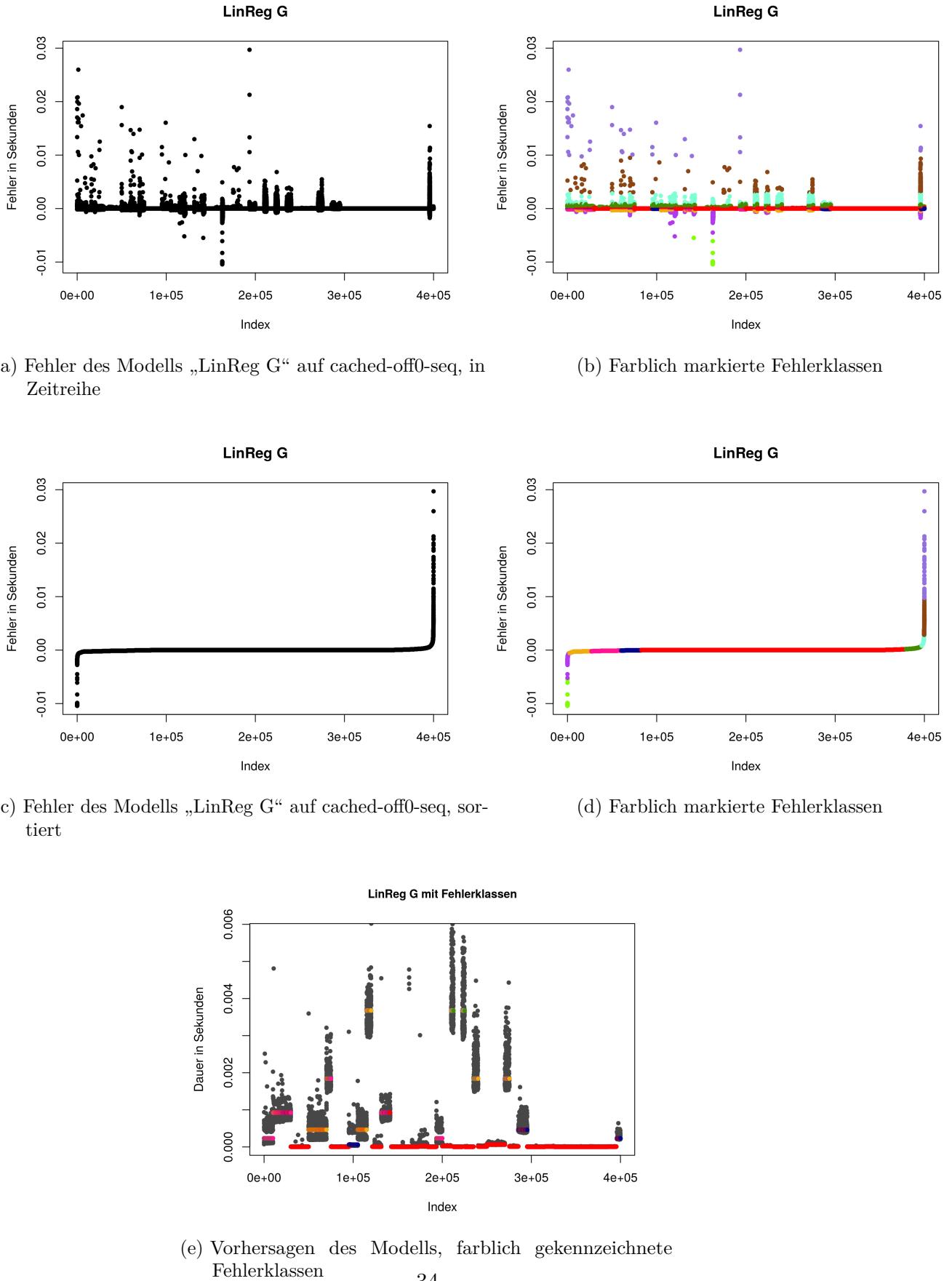


Abbildung 6.12.: Verwendung des K-Means Algorithmus zur Bestimmung der Fehlerklassen auf cached-off0-rnd mit der Vorhersage von „LinReg G“

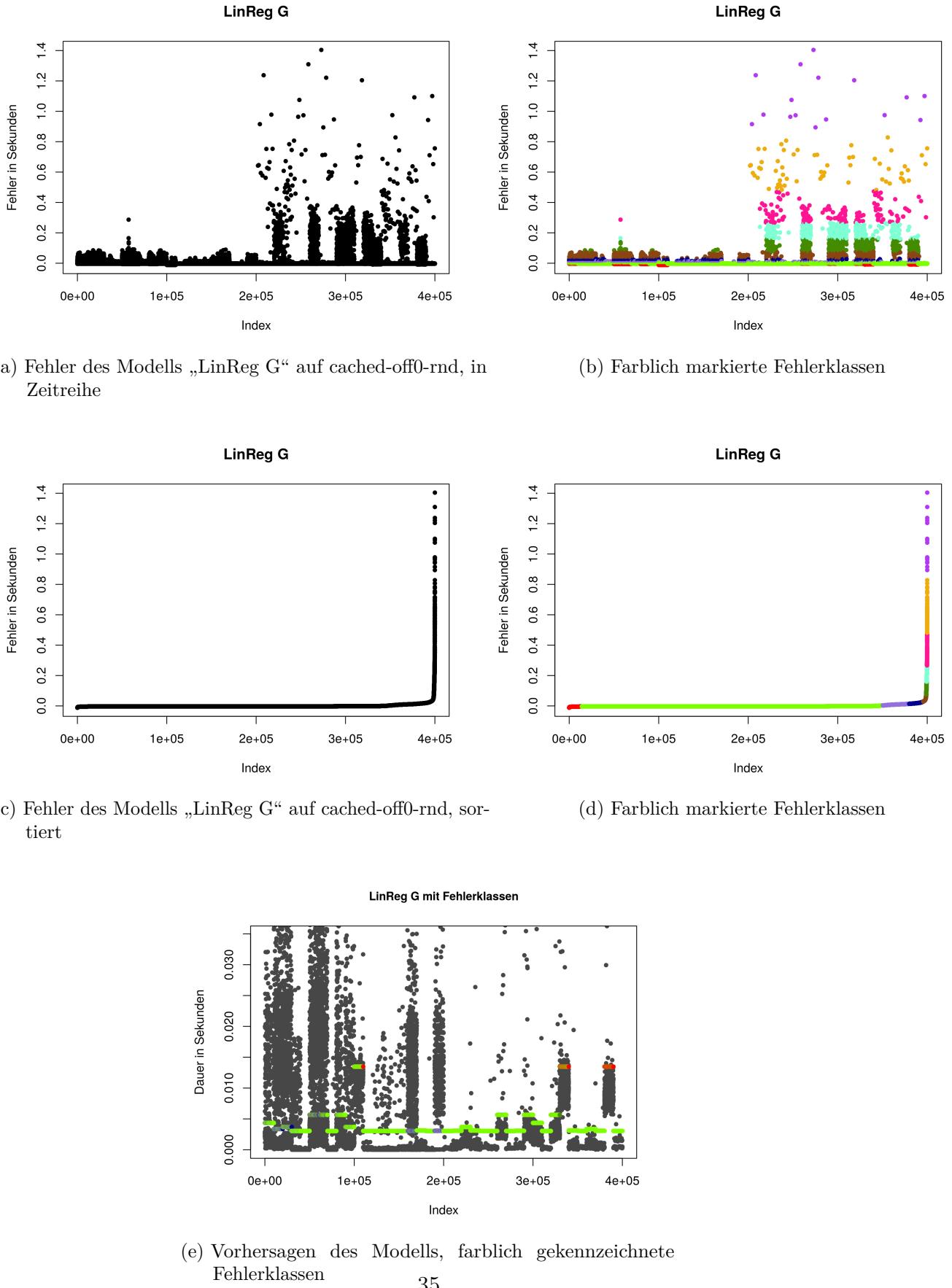


Abbildung 6.13.: Verwendung des K-Means Algorithmus zur Bestimmung der Fehlerklassen auf cached-off0-rnd mit der Vorhersage von „LinReg G“

Modell	MAF (s)	RMAF (%)	RMQA (%)	RQ3 (%)	RMax (%)
Median agg LinReg-Fehlerklasse	1.5e-05	5.2	10	5.8	96
Median agg Tupel1-Fehlerklasse	1.7e-05	5.5	10	6.1	113
Median agg	6.3e-05	12.2	28	9.7	224
LinReg G	7.3e-05	35.2	43	42.2	310
LinReg GA	7.3e-05	35.1	48	41.8	5908
LinReg GAO	8.5e-05	189.8	243	277.2	6021
Durchschnitt	5.6e-04	2744.6	3534	4647.2	6115

(a) cached-off0-seq

Modell	MAF (s)	RMAF (%)	RMQA (%)	RQ3 (%)	RMax (%)
Median agg LinReg-Fehlerklasse	0.00013	7.4	56	6.5	1972
Median agg Tupel1-Fehlerklasse	0.00027	12.3	111	6.8	5654
Median agg	0.00084	39.9	2291	8.9	1359978
LinReg G	0.00470	5019.1	13188	999.0	46070
LinReg GA	0.00470	5018.7	13215	984.7	51343
LinReg GAO	0.00435	7048.9	19261	754.4	72017
Durchschnitt	0.00569	7742.3	20352	1588.3	71083

(b) cached-off0-rnd

Tabelle 6.3.: Ergebnisse der trivialen Modelle

notwendig sich die vorhergesagten Werte gegenüber den tatsächlichen Werten zu betrachten. Wie bei der Exploration der Daten sind dabei einerseits die Vorhersagen in der Reihenfolge in der sie getroffen worden, und andererseits die sortierte Betrachtung von Nutzen. Auf den Graphen zu cached-off0-seq 6.14 lässt sich recht gut die Qualität der Modelle anhand der Stärke der Differenzierung erkennen. Während bei der Durchschnittsdauer gar nicht differenziert wird, können die Modelle Lineare Regression nach Größe und Median aggregierte Daten erfolgreich die größeren Gruppen von Messwerten unterscheiden. Die beiden Modelle mit Fehlerklassen können zudem innerhalb einer Punktgruppe weitere Differenzierungen machen und erreichen entsprechend die besten Fehlerwerte.

Nach der Sortierung nach Laufzeit 6.15 kann dieses Verhalten noch ein wenig deutlicher erkannt werden. Bei den beiden Modellen mit mittlere Leistung sind die Unterscheidungen, die das Modell macht, an den horizontalen Stufen erkennbar. Nach zu Hilfenahme der Fehlerklassen können feinstufigere Vorhersagen gemacht werden.

Grob ergibt sich auch ein ähnliches Bild auf dem cached-off0-rnd Datensatz. Umso größer die Verteilung der Vorhersagen, desto besser das Modell (6.16). Die Probleme der linearen Regression werden hier sehr gut sichtbar, da einer Größe genau ein Funktionswert zugewiesen wird, ist die Abstraktionsmöglichkeit für den randomisierten Datensatz nicht mehr ausreichend. In 6.17 sieht man sehr deutlich die Konzentration der vorhergesagten Werte auf einen recht kleinen Wertebereich. Dieser Umstand führt dazu, dass die Vorhersage im Vergleich zu der auf cached-off0-seq wesentlich ungenauer ist und wie oben

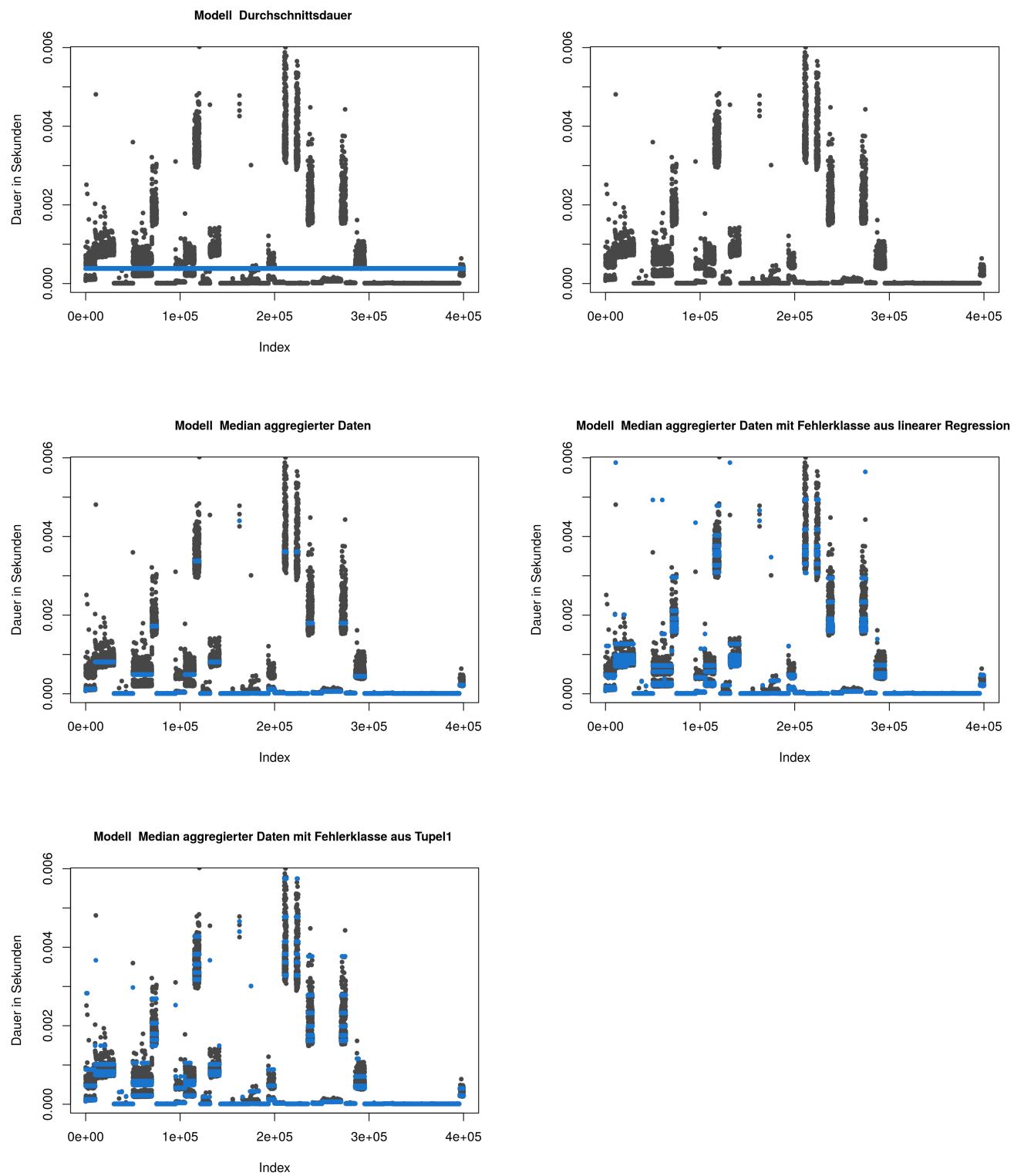


Abbildung 6.14.: Triviale Modelle auf cached-off0-seq als Zeitreihe, in Blau die vom Modell vorhergesagten Werte

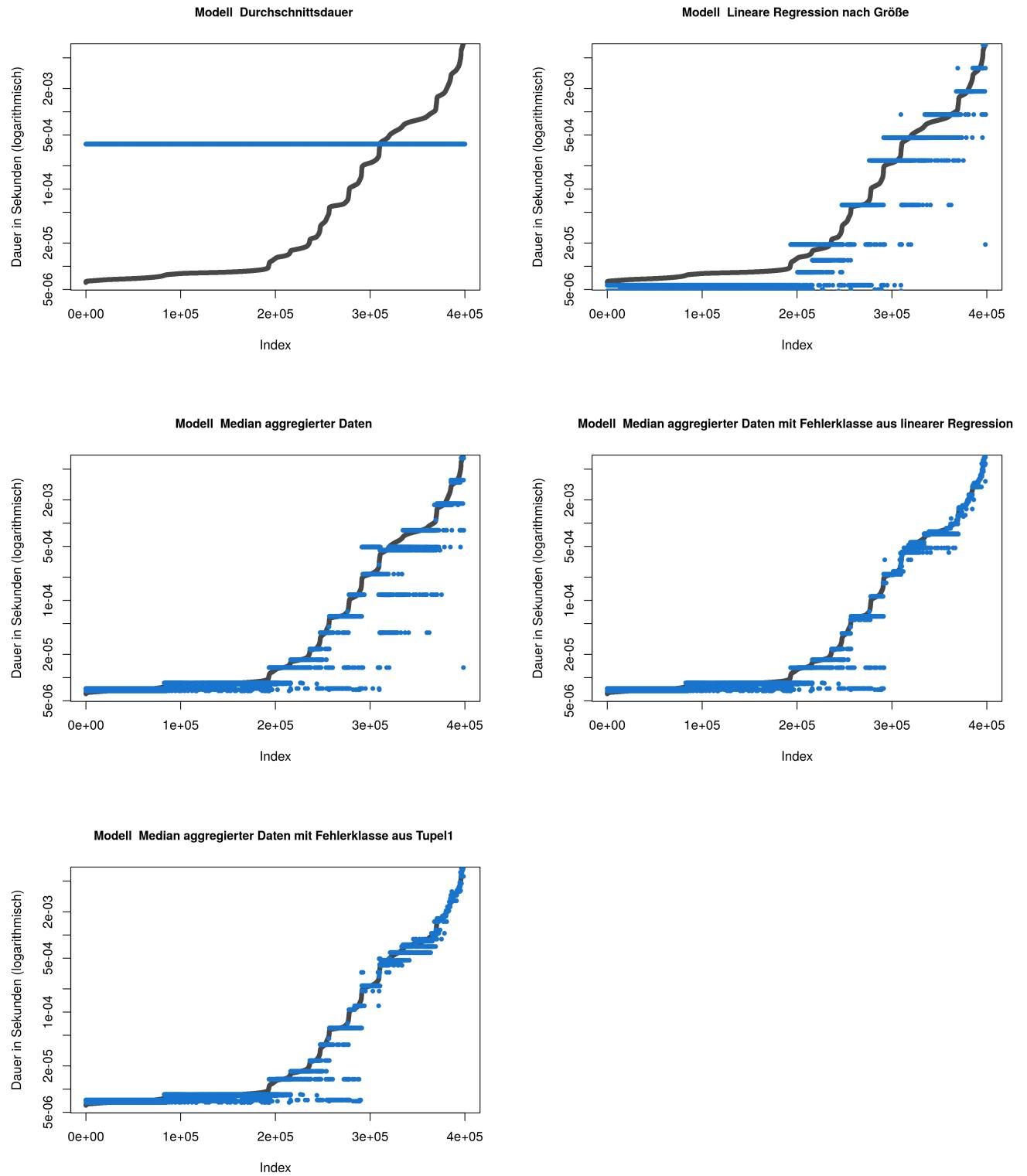


Abbildung 6.15.: Triviale Modelle auf cached-off0-seq nach Laufzeit sortiert (logarithmische Y-Achse)

bereits angemerkt eher mit dem Modell "Durchschnittsdaueräuf einer Stufe steht. Um die Leistungsunterschiede zwischen Median aggünd den Fehlerklassen-Modellen zu erkennen versagt die Zeitreihe im wesentlichen. Im sortierten Graphen ist dagegen klar zu sehen, dass Median aggéine stärkere Streuung bei der Vorhersage der langsameren Datenpunkte aufweist.

Auch jetzt kann durch eine Betrachtung spezifischerer Ausschnitte der Modell-Prognosen untersucht werden, wie die unterschiedliche Qualität der Vorhersagen zu Stande kommt. In den Graphen in 6.18 ist die Dichte der Laufzeiten für jeweils ein spezifisches Attribut-Set dargestellt. Der angegebene Fehler ist der relative arithmetische Fehler des Modells auf dem Set. Es gibt jeweils N Messungen mit Attributen aus diesem Set.

In 6.18 a) ist die Vorhersage vom Modell „Durchschnittsdauer“ mit dem geringsten RMAF-Wert, der erreicht wird, zu sehen. Die Vorhersage ist nicht sehr genau, der vom Modell geschätzte Wert ist für weit weniger als 10% der Messungen korrekt. Die Vorhersage mit dem geringsten Fehler von „LinReg G“ ist wesentlich besser. Interessanterweise ist der vorhergesagte Wert näher am Maximum der Dichte, als am genaueren Mittelwert des Sets. Die Wahl für den Median gegenüber dem arithmetischen Mittel für das Modell „Median agg“ kommt in 6.18 c) zu tragen. Würde das Modell das arith. Mittel vorhersagen, wäre die Vorhersage für dieses Set exakt an der Stelle der schwarzen Markierung. Dadurch wäre der Wert des Fehlermaßes RMAF noch geringer, denn durch die Ausläufer der Messungen dieses Sets zu höheren Laufzeiten wird das arith. Mittel im Graph nach rechts gezogen. Stattdessen wird nun durch den Median ein Großteil der Messungen genauer vorhergesagt, nämlich die Messungen zu den Aufrufen mit der typischen Laufzeit, während die Ausreißer vernachlässigt werden.

Interessanterweise erreichen die beiden Modelle mit Fehlerklassen auf dem selben Set ihren geringsten Fehler mit etwa 1,5%, dabei entspricht die Vorhersage bei dem Modell mit Tupel1-Fehlerklassen (6.18 e)) exakt dem arith. Mittel, während sie bei dem mit LinReg-Fehlerklassen direkt daneben liegt (6.18 d)).

Eine weiterhin bemerkenswerte Erscheinung ist bei 6.18 f) zu sehen. Diese Vorhersage ist die ungenaueste (nach RMAF-Wert), die das Modell „Median agg“ auf cached-off0-seq getroffen hat, dabei ist der Vorhergesagte Wert in direkter Umgebung des „idealen“ arith. Mittel. Einem Modell, dass für alle Messungen eines Sets den selben Wert vorhersagt kann dementsprechend auf diesen Daten nicht immer einen kleinen Fehler erreichen. Eine Unterscheidung von Messungen innerhalb eines Sets kann allerdings nur durch eine Betrachtung der zeitlichen Abhängigkeit des E/A-Aufrufs von den vorherigen gelingen, es sei denn es wären weitere Systeminformationen bekannt.

6.3.2. Analyse der höheren Modelle

6.3.3. Betrachtung der neuronalen Netze

6.4. Ensemble Lernen

Qualität einzelner Netze

statistische Abhängigkeit mehrerer identisch gelernter Netze

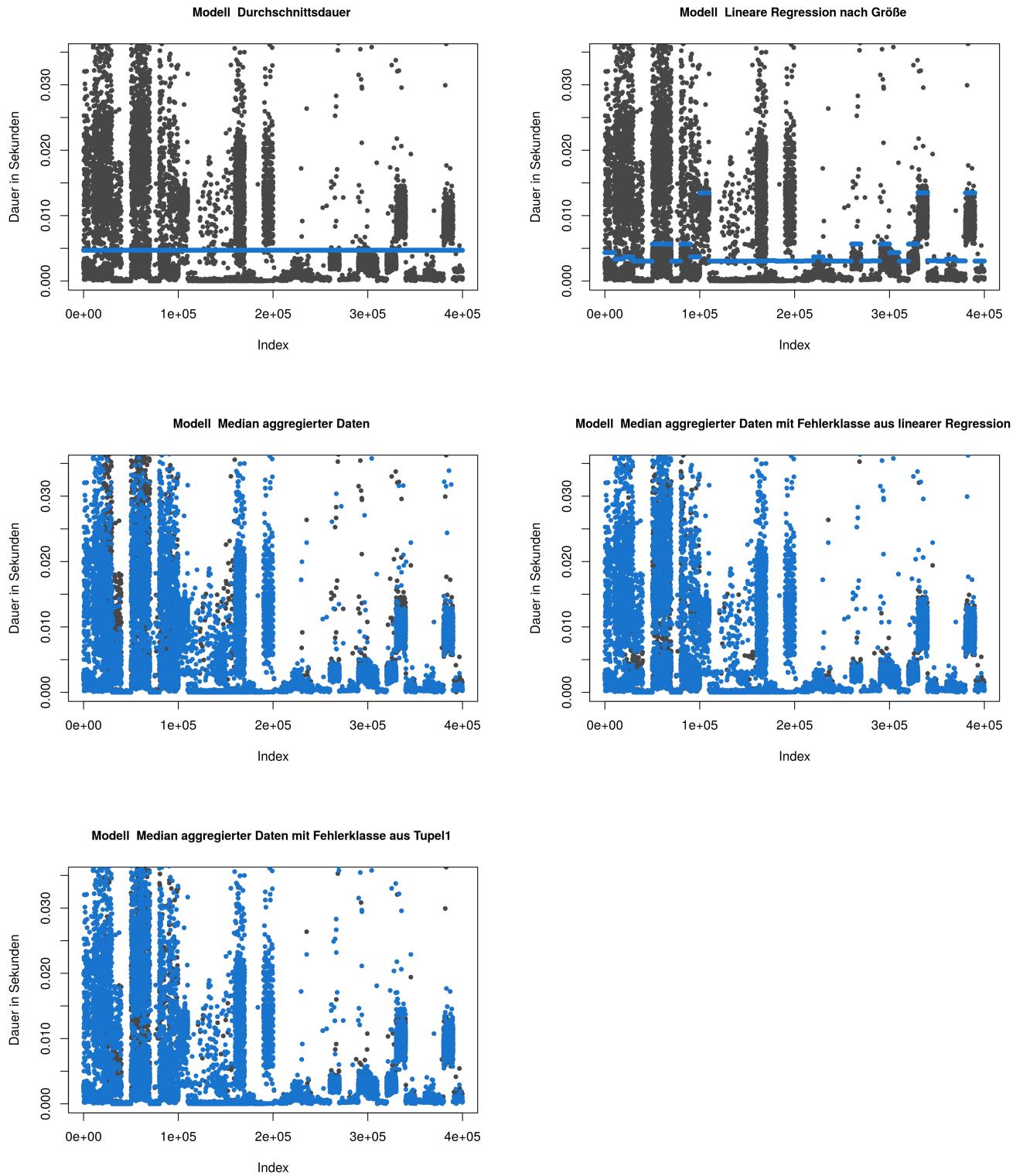


Abbildung 6.16.: Triviale Modelle auf cached-off0-rnd als Zeitreihe, in Blau die vom Modell vorhergesagten Werte

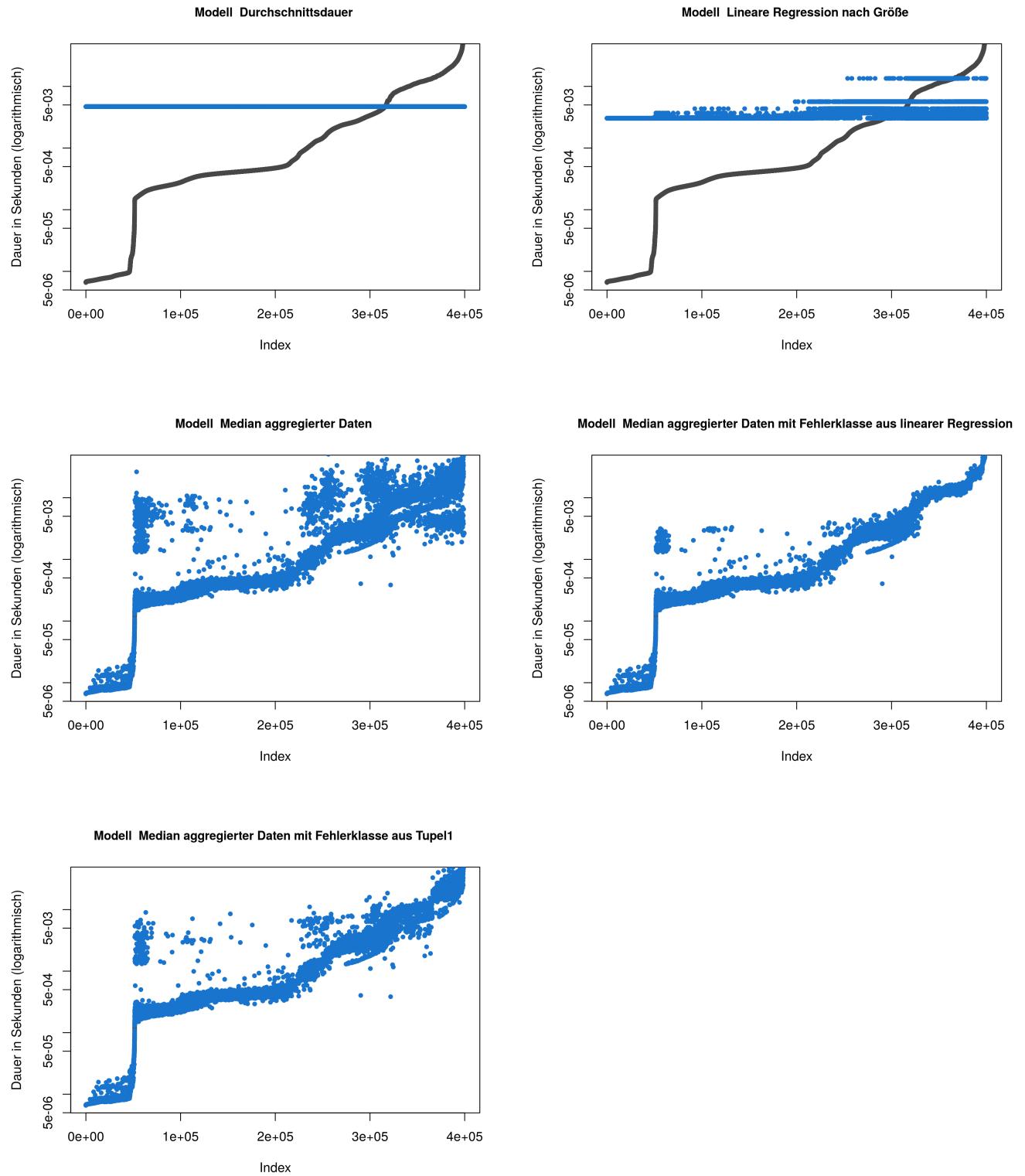


Abbildung 6.17.: Triviale Modelle auf cached-off0-rnd nach Laufzeit sortiert (logarithmische Y-Achse)

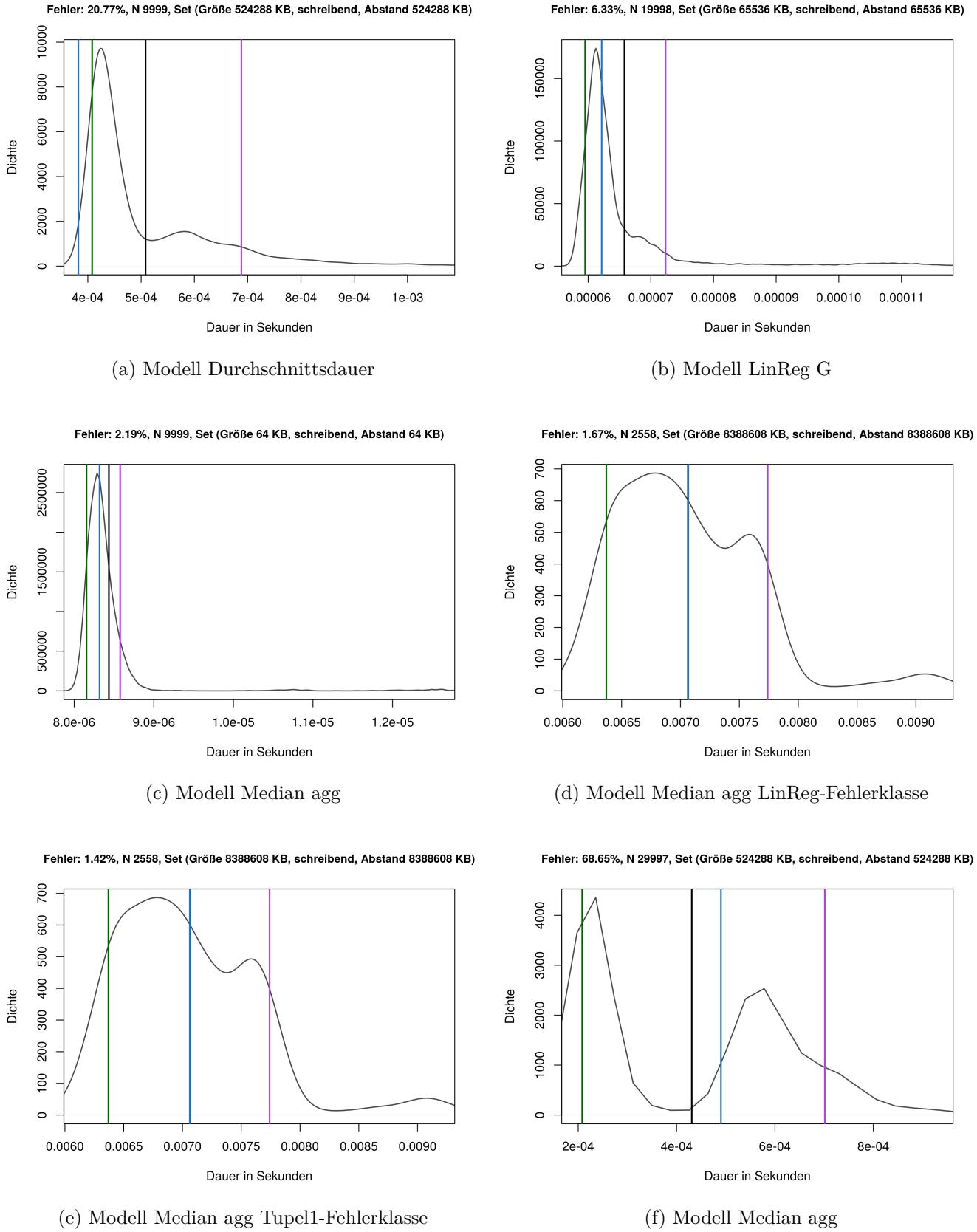


Abbildung 6.18.: Triviale Modelle auf cached-off0-seq, 90% der Werte sind größer als die grüne, 10% sind größer als die pinke Linie, die mittlere Dauer entspricht der schwarzen Linie, die blaue Linie ist die mittlere Vorhersage des Modells für dieses Set

Modell	MAF (s)	RMAF (%)	RMQA (%)	RQ3 (%)	RMax (%)
Tupel1 LinReg-Fehlerklasse	2.1e-05	6.6	11	8.1	382
Tupel1	6.7e-05	13.8	23	16.1	483

(a) cached-off0-seq

Modell	MAF (s)	RMAF (%)	RMQA (%)	RQ3 (%)	RMax (%)
Tupel1	0.0014	44	108	44	3525
Tupel1 LinReg-Fehlerklasse	0.0025	93	1096	46	69533

(b) cached-off0-rnd

Tabelle 6.4.: Ergebnisse der Modelle

Modell	verdeckte Schichten	Neuronen	Iterationen	Trainingsdauer
Tupel1	12	13	3292	1421
Tupel1 LinReg-Fehlerklasse	6	29	2806	818

(a) cached-off0-seq

Modell	verdeckte Schichten	Neuronen	Iterationen	Trainingsdauer
Tupel1	12	13	3292	1421
Tupel1 LinReg-Fehlerklasse	6	29	2806	818

(b) cached-off0-rnd

Tabelle 6.5.: Informationen über die erfolgreichsten Neuronalen Netze

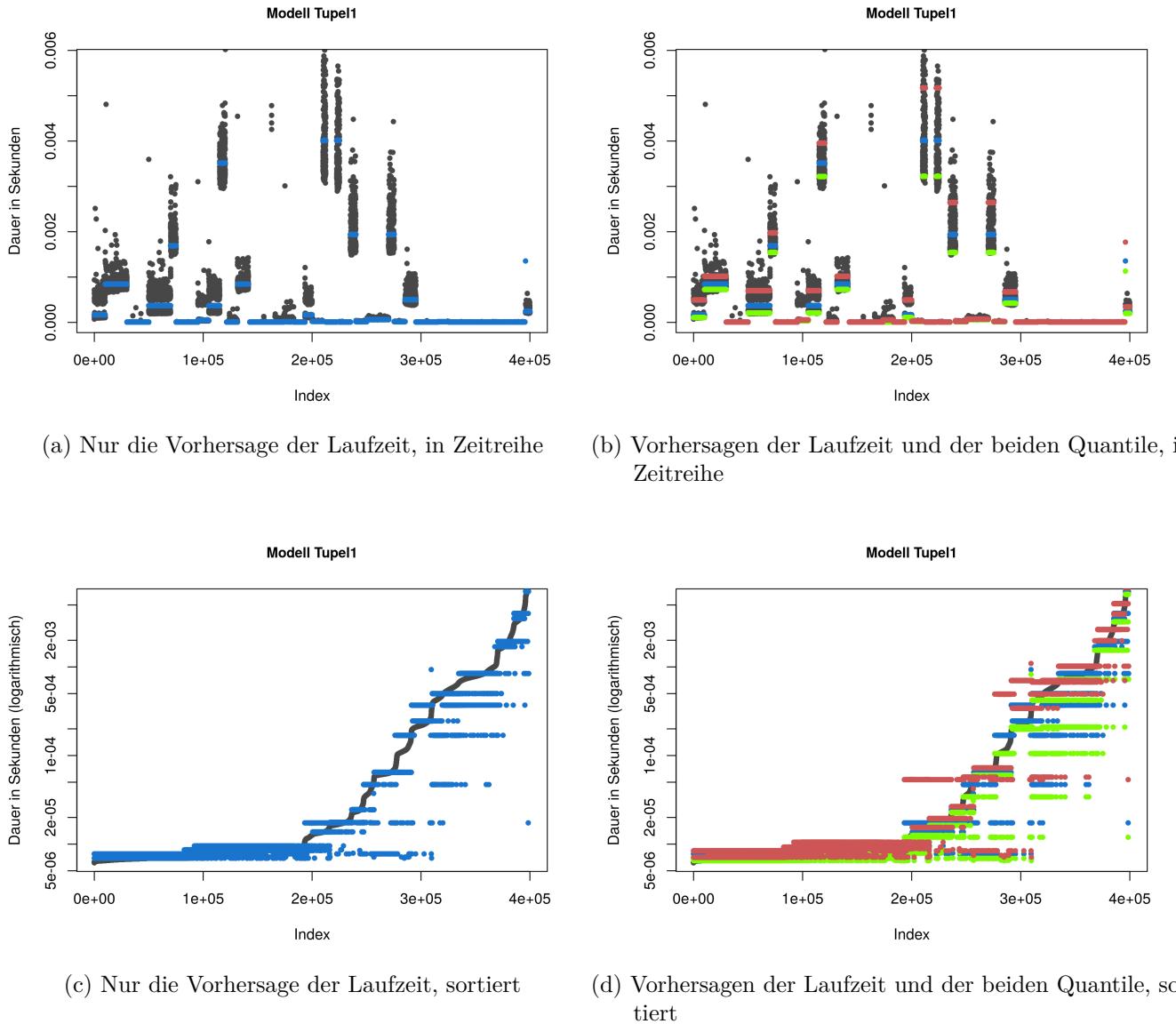


Abbildung 6.19.: Betrachtung der Vorhersagen von „Tupel1“ auf cached-off0-seq. In blau Vorhersage der Laufzeit, rot/grün Vorhersage des Quantil 0.9/0.1

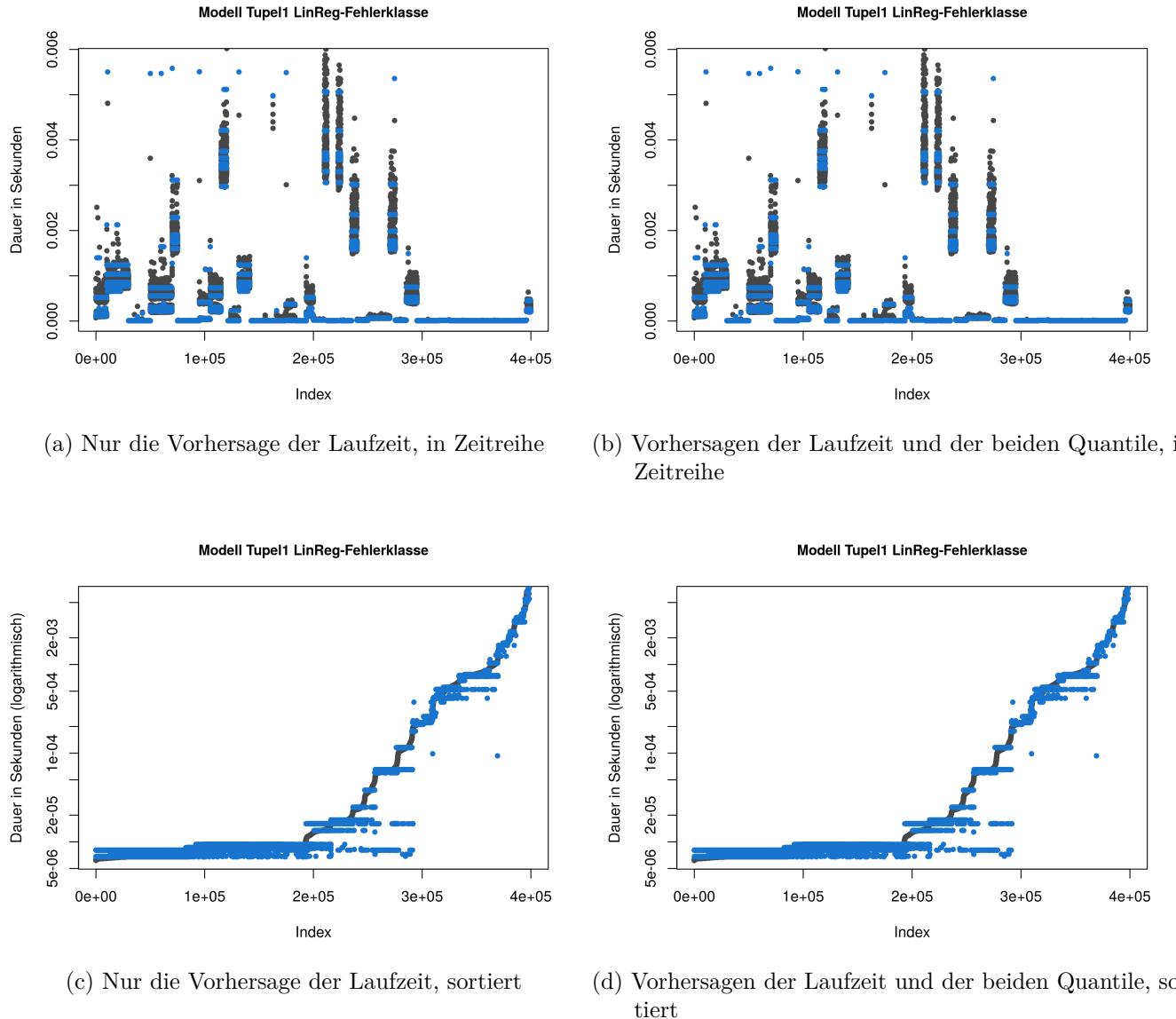


Abbildung 6.20.: Betrachtung der Vorhersagen von „Tupel1 LinReg-Fehlerklassen“ auf cached-off0-seq. In blau Vorhersage des Modells, rot bzw. grün Vorhersage des Quantil 0.9 bzw. 0.1

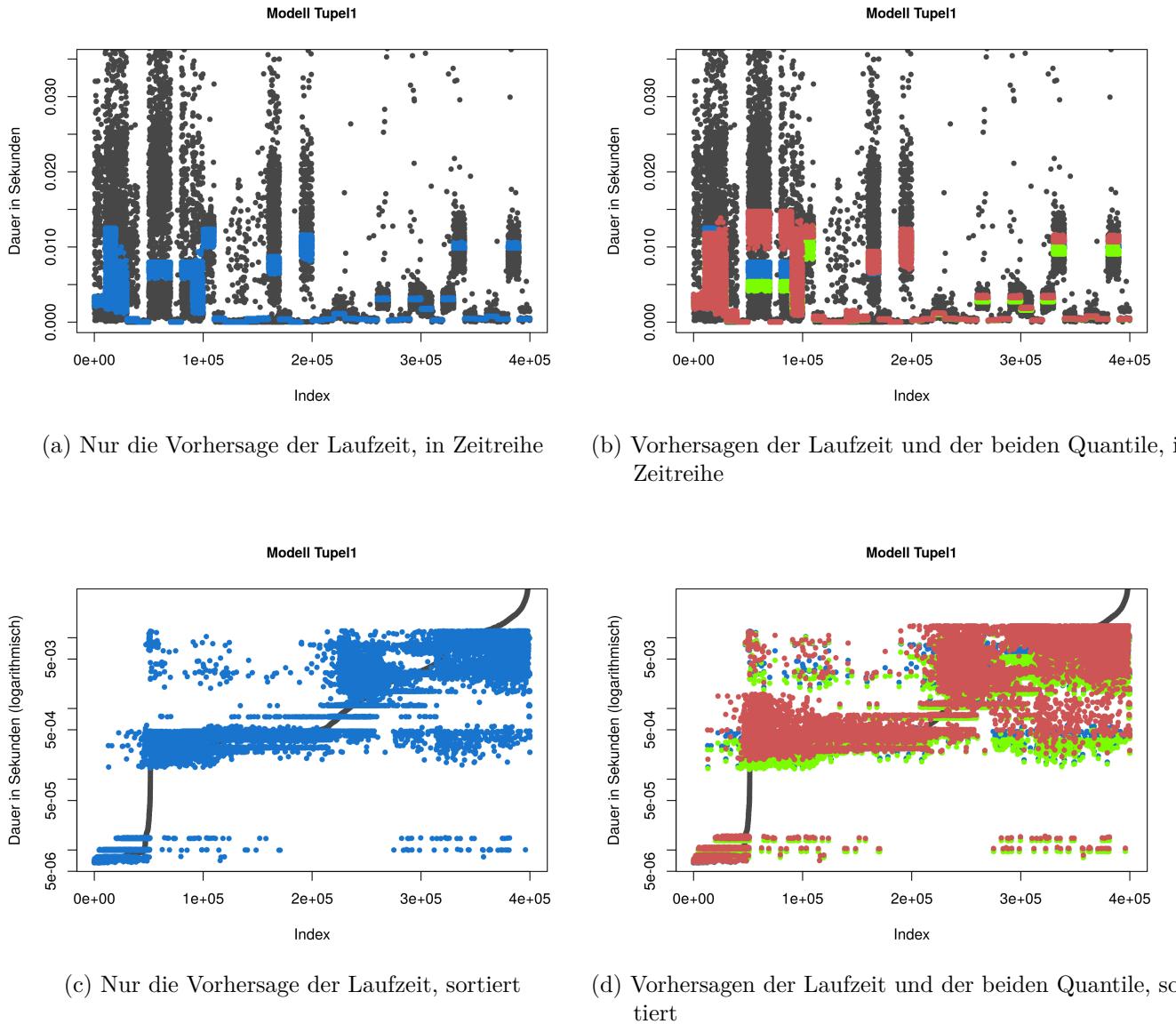


Abbildung 6.21.: Betrachtung der Vorhersagen von „Tupel1“ auf cached-off0-rnd. In blau Vorhersage der Laufzeit, rot/grün Vorhersage des Quantil 0.9/0.1

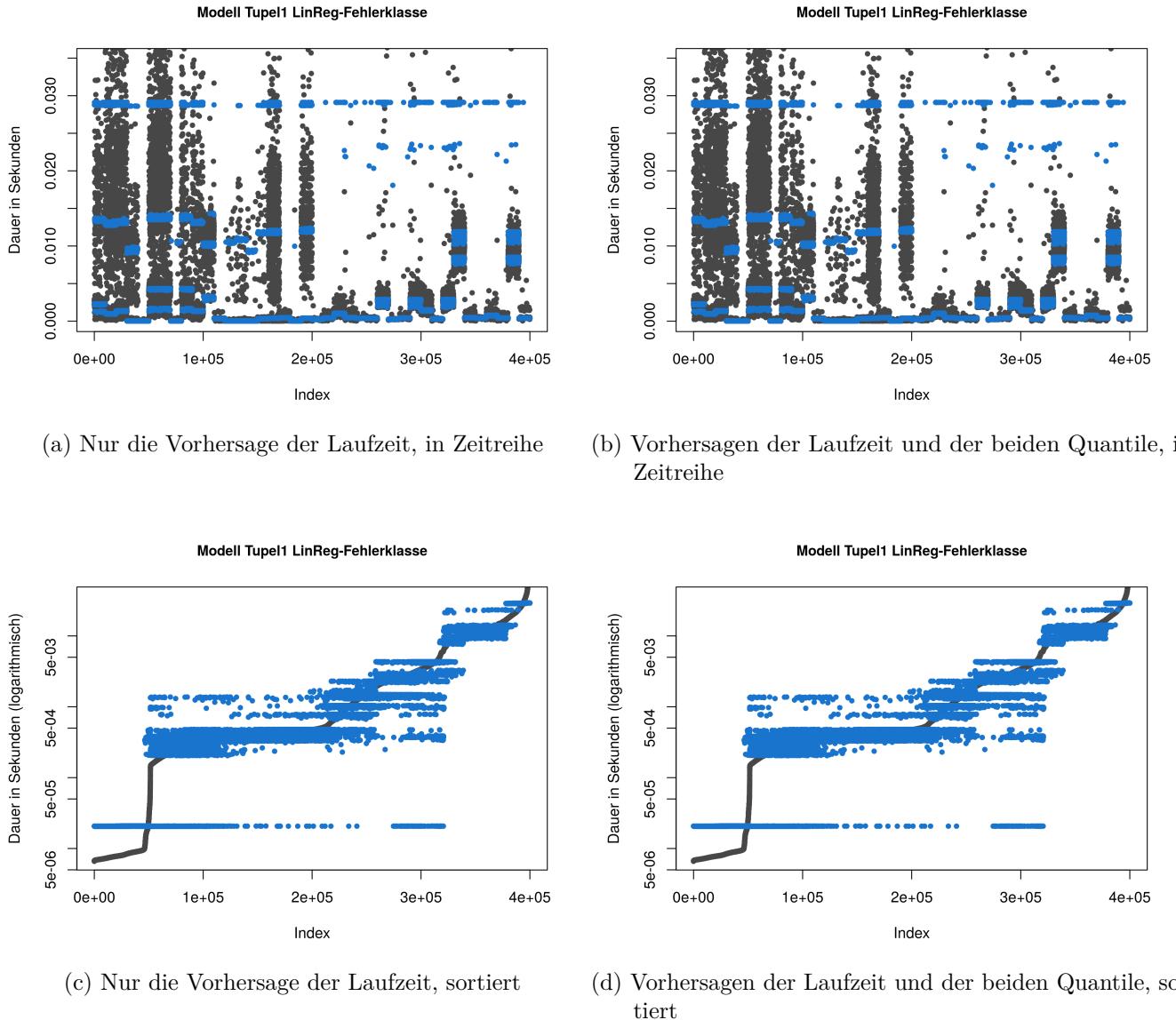


Abbildung 6.22.: Betrachtung der Vorhersagen von „Tupel1 LinReg-Fehlerklassen“ auf cached-off0-rnd. In blau Vorhersage des Modells, rot bzw. grün Vorhersage des Quantil 0.9 bzw. 0.1

Zusammenfassung: 2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.

7. Fazit

Welche Modelle sind für welchen Fall erfolgsversprechend? Eignen sich Neuronale Netze zum Vorhersagen von E/A-Leistung im HPC?

Wo könnten die Ergebnisse eingesetzt werden?

Was müsste als nächstes getan werden?

Literaturverzeichnis

- [Alp10] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.
- [BSSG08] John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. The DiskSim Simulation Environment Version 4.0 Reference Manual, 2008.
- [CMW⁺13] Adam Crume, Carlos Maltzahn, Lee Ward, Thomas Kroeger, Matthew Curry, and Ron Oldfield. Fourier-assisted Machine Learning of Hard Disk Drive Access Time Models. In *Proceedings of the 8th Parallel Data Storage Workshop*, PDSW '13, pages 45–51, New York, NY, USA, 2013. ACM.
- [Cyb89] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.
- [DLZC12] Chengjun Dai, Guiquan Liu, Lei Zhang, and Enhong Chen. Storage Device Performance Prediction with Hybrid Regression Models. In *Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, PDCAT '12, pages 556–559, Washington, DC, USA, 2012. IEEE Computer Society.
- [Kan11] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.
- [KCGK04] Terence Kelly, Ira Cohen, Moises Goldszmidt, and Kimberly Keeton. Inducing models of black-box storage arrays. Technical report, 2004.
- [KZB15] Julian Kunkel, Michaela Zimmer, and Eugen Betke. Using Machine Learning to Predict the Performance of Non-Contiguous I/O, 07 2015.
- [LDK09] AS Lebrecht, NJ Dingle, and WJ Knottenbelt. A Performance Model of Zoned Disk Drives with I/O Request Reordering. pages 97–106. IEEE COMPUTER SOC, 2009.
- [LFC⁺] Yonggang Liu, Renato Figueiredo, Dulcardo Clavijo, Yiqi Xu, and Ming Zhao. Towards Simulation of Parallel File System Scheduling Algorithms with TODO.
- [Roj96] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

- [RW94] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27:17–28, 1994.
- [ZLZ⁺10] Lei Zhang, Guiquan Liu, Xuechen Zhang, Song Jiang, and Enhong Chen. Storage Device Performance Prediction with Selective Bagging Classification and Regression Tree. In *Network and Parallel Computing, IFIP International Conference, NPC 2010, Zhengzhou, China, September 13-15, 2010. Proceedings*, pages 121–133, 2010.

Abbildungsverzeichnis

2.1	Zweischichtiges Netz (wiki)	12
2.2	Schema eines künstlichen Neurons (wiki)	13
2.3	Lineare Separierbarkeit von Funktionen (wiki)	13
6.1	Messungen der Laufzeiten als Zeitreihe dargestellt	22
6.2	Messungen der Laufzeiten sortiert dargestellt (logarithmische Y-Achse)	23
6.3	Ausreißer sind in rot dargestellt, sie überdecken die grauen Punkte	24
6.4	Messungen der Laufzeiten nach Zugriffsgröße sortiert dargestellt (logarithmische Y-Achse)	26
6.5	Detailbetrachtung aller Messungen mit Zugriffsgröße 1KB	27
6.6	Detailbetrachtung aller Messungen mit Zugriffsgröße 16384KB	28
6.7	Detailbetrachtung aller Messungen mit Zugriffsgröße 2097152KB	29
6.8	Detailbetrachtung der ersten 250 Messungen	30
6.9	Wiederholung der ersten Werte, als erstes einfaches Modell	31
6.10	Detailbetrachtung der Messungen 100001 bis 100250	32
6.11	Wiederholung der ersten Werte, als erstes einfaches Modell	33
6.12	Verwendung des K-Means Algorithmus zur Bestimmung der Fehlerklassen auf cached-off0-rnd mit der Vorhersage von „LinReg G“	34
6.13	Verwendung des K-Means Algorithmus zur Bestimmung der Fehlerklassen auf cached-off0-rnd mit der Vorhersage von „LinReg G“	35
6.14	Triviale Modelle auf cached-off0-seq als Zeitreihe, in Blau die vom Modell vorhergesagten Werte	37
6.15	Triviale Modelle auf cached-off0-seq nach Laufzeit sortiert (logarithmische Y-Achse)	38
6.16	Triviale Modelle auf cached-off0-rnd als Zeitreihe, in Blau die vom Modell vorhergesagten Werte	40
6.17	Triviale Modelle auf cached-off0-rnd nach Laufzeit sortiert (logarithmische Y-Achse)	41
6.18	Triviale Modelle auf cached-off0-seq, 90% der Werte sind größer als die grüne, 10% sind größer als die pinke Linie, die mittlere Dauer entspricht der schwarzen Linie, die blaue Linie ist die mittlere Vorhersage des Modells für dieses Set	42
6.19	Betrachtung der Vorhersagen von „Tupel1“ auf cached-off0-seq. In blau Vorhersage der Laufzeit, rot/grün Vorhersage des Quantil 0.9/0.1	44

6.20	Betrachtung der Vorhersagen von „Tupel1 LinReg-Fehlerklassen“ auf cached-off0-seq. In blau Vorhersage des Modells, rot bzw. grün Vorhersage des Quantil 0.9 bzw. 0.1	45
6.21	Betrachtung der Vorhersagen von „Tupel1“ auf cached-off0-rnd. In blau Vorhersage der Laufzeit, rot/grün Vorhersage des Quantil 0.9/0.1	46

Tabellenverzeichnis

6.1	Metainformationen über die Datensätze	21
6.2	Metainformationen über OpTyp	21
6.3	Ergebnisse der trivialen Modelle	36
6.4	Ergebnisse der Modelle	43
6.5	Informationen über die erfolgreichsten Neuronalen Netze	43

Listingverzeichnis

Anhänge

A. Anhangskapitel

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor
invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et
accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata
sanctus est Lorem ipsum dolor sit amet.

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen, als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internetquellen – benutzt habe, die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Optional: Ich bin mit der Einstellung der Bachelor-Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik einverstanden.

Hamburg, den 01.01.2012