HAW HAMBURG

Bachelorarbeit

Luca Leo Gebhardt

Erkennung von verfügbarkeitsgefährdendem Nutzerverhalten im dCache-System

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Luca Leo Gebhardt

Erkennung von verfügbarkeitsgefährdendem Nutzerverhalten im dCache-System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Jan Sudeikat
Zweitgutachter: Prof. Dr. Michael Köhler-Bußmeier

Eingereicht am: 12. Oktober 2023

Luca Leo Gebhardt

Thema der Arbeit

Erkennung von verfügbarkeitsgefährdendem Nutzerverhalten im dCache-System

Stichworte

dCache, Data Mining, KDD, logistische Regression, MAPE-Loop

Kurzzusammenfassung

Das Ziel der vorliegenden Arbeit ist es in den Transfer-Events und pool-info-Logs des dCache-Speichersystems Indikatoren zu identifizieren, die auf ein verfügbarkeitsgefährdendes Nutzerverhalten durch parallele Dateizugriffe hindeuten. Weiterhin soll mit diesen Indikatoren ein Klassifikations-Modell auf Basis der logistischen Regression entwickelt werden, welches in ein Software-Tool eingebettet wird, welches im Anschluss evaluiert wird. Um Indikatoren in den Events und Log-Daten zu identifizieren wurde der KDD-Prozess genutzt, wodurch strukturiert Merkmale in den Daten ausgewählt und extrahiert werden konnten. Mit diesen Daten konnte ein Modell trainiert und getestet werden. Das Software-Tool ist auf Basis eines MAPE-Loops entwickelt worden, in welchen das Modell eingebettet wurde. Mittels verschiedener Evaluationsmetriken ist das Modell mit Referenzwerten und einem Baseline-Klassifikator verglichen worden. Zur Evaluation ist das Software-Tool in den Betrieb integriert worden und hat die Zugriffe der echten Nutzer überwacht. Zur reproduzierbaren Evaluation wurden verschiedene Zugriffsmuster simuliert. Diese Arbeit konnte zeigen, dass sich aus den Transfer-Events Indikatoren extrahieren lassen und diese auch erfolgreich zum Training eines Modells verwendet werden können. Weiterhin konnte dieses in ein funktionierendes Software-Tool integriert werden, welches getestet wurde. Die Ergebnisse der Evaluation zeigen, dass das Modell im Betrieb verfügbarkeitsgefährdende Zugriffe erkennt und meldet. Im Zuge der Evaluation ist ein weiterer Indikator ermittelt worden, welcher genutzt werden kann, um möglicherweise die Anzahl an Falschmeldungen zu verringern. Die Simulation der Zugriffe hat nicht zum erwarteten Ergebnis geführt, es konnten aber mögliche Ursachen identifiziert werden.

Luca Leo Gebhardt

Title of Thesis

Detection of availability-threatening user behaviour in the dCache system

Keywords

dCache, Data Mining, KDD, logistic regression, MAPE-Loop

Abstract

The objective of the present work is to identify indicators of user behavior potentially threatening availability through parallel file accesses in the transfer events and pool-info logs of the dCache storage system. Additionally, a classification model based on logistic regression is developed using these indicators, embedded into a software tool, and evaluated. The Knowledge Discovery in Databases (KDD) process was employed to systematically select and extract features from the events and log data. With these data, a model was trained and tested. The software tool was developed based on a Monitoring, Analyze, Planning, Execution Loop (MAPE-Loop), incorporating the model. The model was compared with reference values and a baseline classifier using various evaluation metrics. For evaluation, the software tool was integrated into operations, monitoring real user accesses. Different access patterns were simulated for reproducible evaluation. This study demonstrated that indicators can be extracted from transfer events and successfully used for model training. Furthermore, the model was integrated into a functional software tool and successfully tested. Evaluation results indicate that the model detects and reports availability-threatening accesses during operations. An additional indicator was identified during evaluation, which could potentially reduce false positives. Simulating accesses did not yield the expected results, but potential causes were identified.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

CERN Conseil européen pour la recherche nucléaire.

CRISP-DM Cross-Industry Standard Process for Data Mining.

CSV Comma-separated Value.

DESY Deutsche Elektronen Synchrotron.

DTO Data Transfer Object.

GB Gigabyte.

GROGU storaGe Responsiveness Overwatching Guarding Utility.

JSON JavaScript Object Notation.

KDD Knowledge Discovery in Databases.

KNN k-Nearest Neighbour.

LHC Large Hadron Collider.

MAPE-Loop Monitoring, Analyze, Planning, Execution Loop.

ms Millisekunden.

NAF National Analysis Facility.

NFS Network File System.

PR-Kurve Precision-Recall-Kurve.

RNN rekurrente neuronale Netze.

SVM Support Vector Machine.

UML Unified Modeling Language.

1 Einleitung

Die zunehmende Komplexität physikalischer Forschungsprojekte erzeugt eine immer stärker wachsende Menge an Daten. Einige der weltweit größten Forschungsexperimente werden am Large Hadron Collider (LHC) beim Conseil européen pour la recherche nucléaire (CERN) durchgeführt [4][5]. So erzeugen die Experimente des LHC eine jährliche Datenmenge von 30 Petabyte [9]. Um Analysen auf dieser enormen Datenmenge zu ermöglichen haben sich Rechenzentren zum LHC Computing Grid zusammengeschlossen. Die LHC-Daten werden an verschiedenen Orten weltweit gespeichert und analysiert. Das Deutsche Elektronen Synchrotron (DESY) beteiligt sich als Rechen- und Speicher-Einrichtung für die Experimente A Toroidal LHC Apparatus (ATLAS), Compact Muon Solenoid (CMS) und LHC beauty (LHCb) [9]. Rechenintensive Analysen werden auf dem batch-basierten Rechenkomplex National Analysis Facility (NAF) durchgeführt [11]. Zur Datenspeicherung setzt das DESY das dCache-Speichersystem ein. Das dCache-Speichersystem hat sich als eine Lösung für die effiziente Verwaltung großer Datenmengen etabliert und wird von zahlreichen Einrichtungen weltweit zur Speicherung großer Datenmengen genutzt [6].

Bei komplexen Analysen auf Daten, die im dCache-Speichersystem gespeichert sind, kann es zu Problemen kommen, die die Verfügbarkeit des dCache-Speichersystems beeinträchtigen. Eine Problemursache sind parallele Lesezugriffe auf einzelne Dateien innerhalb kurzer Zeit. In dieser Arbeit soll eine Lösung zur Erkennung dieses Problem entwickelt und evaluiert werden.

1.1 Problemstellung

Auf einem physikalischen Server der NAF befinden sich Slots, die von verschiedenen Forschungsgruppen verwendet werden. Über das Network File System (NFS)-Protokoll wird das dCache-Speichersystem auf einen Host der NAF eingebunden [7]. Verschiedene Slots greifen über den gleichen NFS-Client auf das dCache-Speichersystem zu. Wenn Slots auf

verschiedenen physikalischen Hosts parallel auf eine Datei zugreifen, führt dies dazu, dass ein dCache-Pool viele Anfragen gleichzeitig bearbeiten muss. Dies resultiert in einer langen Antwortzeit und blockiert den Zugriff anderer Slots auf das dCache-Speichersystem, die sich auf dem gleichen physikalischen Server befinden. Wenn ein solcher Zugriff auf Dateien stattfindet, folgt ein Abbruch der Verbindung zwischen dem NFS-Client und Server bis die angeforderten Dateien geladen sind, wodurch andere Slots keinen Zugriff mehr auf das dCache-Speichersystem haben. In einigen Fällen führte ein solches Verhalten zum erzwungenen Neustart des NFS-Clients und einem Ausfall für den entsprechenden Zeitraum.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es herauszufinden, ob es Indikatoren gibt, die darauf hinweisen, dass es zu einem verfügbarkeitsgefährdenden Verhalten der Nutzer durch parallele Zugriffe auf Dateien innerhalb kurzer Zeit kommt. Dafür werden auf Basis historischer Meldungen der dCache-Administratoren Zeiträume eingegrenzt und es wird die Code-Basis geschaffen, um die Transfer- und pool-info-Daten zu verarbeiten.

Außerdem wird untersucht, ob mit den Indikatoren und Methoden des Data Mining ein Modell entwickelt werden kann, welches Zeiträume entsprechend klassifiziert. Das Software-Tool wird im Anschluss mit verschiedenen Methoden evaluiert.

Weiterhin soll ein Software-Tool entwickelt werden, welches Daten im Betrieb verarbeiten kann und im Falle eines verfügbarkeitsgefährdenden Verhaltens auf eine Datei die dCache-Administratoren benachrichtigt.

Darüber hinaus soll das Software-Tool im Live-Betrieb sowie durch eine Zugriffs-Simulation evaluiert werden, um zu überprüfen, ob das Software-Tool sowie das Modell im Betrieb eingesetzt werden können.

1.3 Gliederung der Arbeit

Zu Beginn ist in Kapitel 2 ein Überblick über die Funktionsweise des dCache-Speichersystems gegeben und wie die Transfer-Events und Log-Nachrichten strukturiert sind.

Außerdem erläutert dieses Kapitel, wie mittels des KDD-Prozesses Wissen aus Daten gewonnen wird und wie die logistische Regression für Klassifikationsaufgaben eingesetzt werden kann. Zum Abschluss des Kapitels wird der MAPE-Loop vorgestellt, auf dem das Software-Tool storaGe Responsiveness Overwatching Guarding Utility (GROGU) basiert, sowie die Evaluationsmetriken und Verfahren zur Evaluation des Modells. Kapitel 3 erklärt wie der KDD-Prozess benutzt wird, um ein Modell aus den Daten zu entwickeln. Darüber hinaus beschreibt das Kapitel den Aufbau des Tools GROGU und wie es im Betrieb verwendet werden kann. Anschließend wird in Kapitel 4 die Aussagekraft des Modells evaluiert und mit einem Baseline-Modell verglichen. Weiterhin wird die simulierte Evaluation und Live-Evaluation des Software-Tool vorgestellt. Es wird erläutert, wie Zugriffe simuliert werden können und wie die Bedeutung der Resultate der Live-Evaluation einzuschätzen ist. Abschließend wird in Kapitel 5 die Arbeit zusammengefasst und diskutiert. Weiterhin werden Einschränkungen und Alternativen der Arbeit und Empfehlungen für die weitere Forschung gegeben.

2 Grundlagen

Im folgenden Kapitel werden die Funktionsweise und Komponenten des dCache-Speichersystems erläutert. Darüber hinaus wird erläutert wie die Transfer-Events und pool-info-Logs generiert und persistiert werden. Außerdem wird das Konzept des KDD-Prozesses vorgestellt, welcher genutzt wird, um aus Daten Wissen zu extrahieren. Bestandteil dieses Prozesses wird die logistische Regression sein, welche erläutert wird. Zur Evaluation werden verschiedene Metriken genutzt, die in diesem Kapitel erläutert werden sollen. Der MAPE-Loop wird zum Abschluss vorgestellt. Dieser stellt die Grundlage für das Software-Tool dar.

2.1 dCache-Speichersystem

Das dCache-Speichersystem ist ein System, welches dem Speichern und Abrufen von großen Datenmengen dient. Die Daten werden über einen einzelnen virtuellen Verzeichnisbaum zur Verfügung gestellt [6]. Der Zugriff auf die Daten kann über diverse Protokolle wie beispielsweise NFS erfolgen [7]. Eine minimale dCache-Instanz besteht aus den folgenden Komponenten [6]:

Komponente	**Beschreibung**
door	Protokoll-spezifischer Zugriffspunkt der Protokoll-Instruktionen in eine Sequenz von dCache-internen Nachrichten konvertiert.
namespace	Bietet eine hierarchische Dateisystem-Sicht auf die gespeicherten Daten an.
pool	Beinhaltet Daten, die über einen eindeutigen Identifier identifizierbar sind. Der Identifier wird der Datei bei der Erstellung zugewiesen.
poolmanager	Verwaltet alle Pools im System. Schreib- oder Lese-Anfragen werden vom poolmanager verarbeitet.
zookeeper [3]	Verwaltung von Konfigurations- und Layout-Einstellungen.

Tabelle 2.1: Beschreibung der Komponenten einer minimalen dCache-Instanz [6]

Wie in Abbildung 2.1 dargestellt, geht ein Zugriff über ein beliebiges Zugriffsprotokoll wie NFS an die entsprechende `Door`. Diese leitet den Zugriff weiter an die `Core Domain`, welche von verschiedenen Services Metainformationen einsammelt. So stellt sie beim `Namespace Manager` eine Anfrage, wie die interne ID für die Datei lautet und beim `Pool Manager` eine Anfrage nach dem Pool für die entsprechende Datei. Diese Informationen werden an den dCache-Client zurückgeleitet. Mit den gelieferten Informationen kann der Client auf die angeforderte Datei auf dem korrekten `Pool` zugreifen. Dieser Ablauf geschieht ohne Eingreifen des Nutzers.

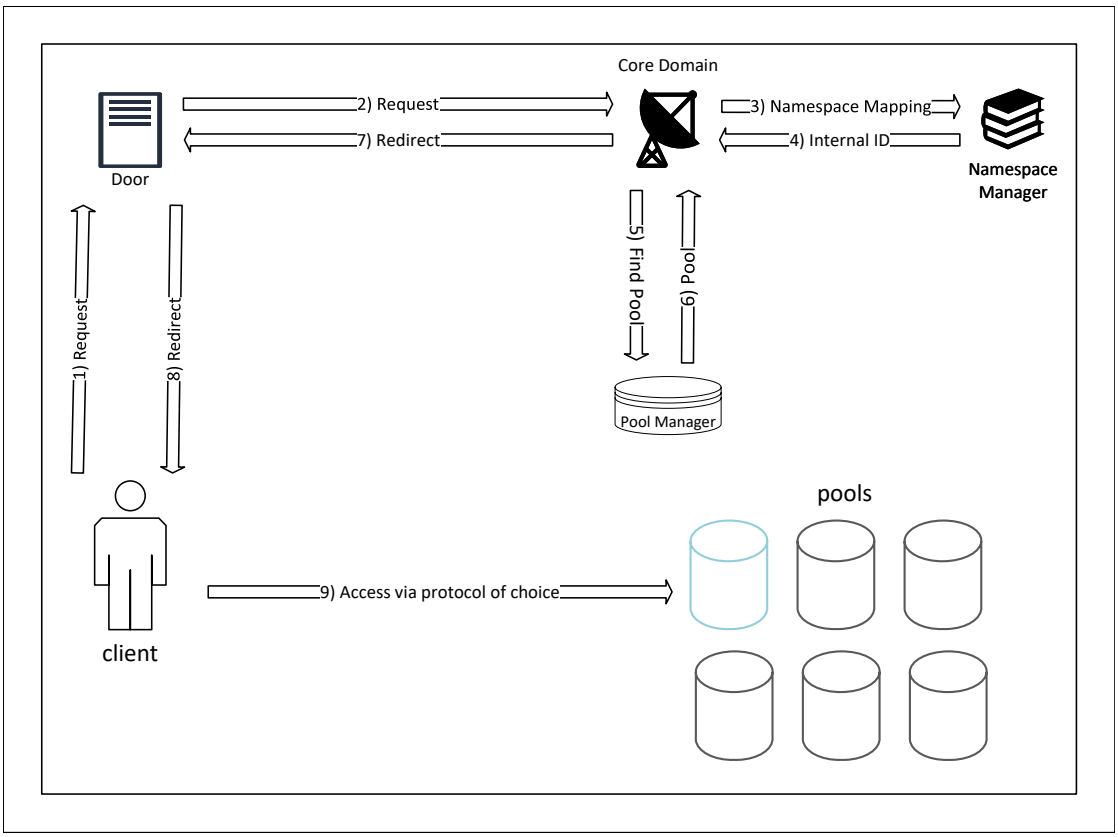

Abbildung 2.1: Dateizugriff im dCache-Speichersystem

Wenn eine Datei über ein Protokoll angefragt wird, wird ein sogenannter *Mover* gestartet. Die Anzahl der Mover gibt zu einem bestimmten Zeitpunkt an, wie viele Dateien aktuell über ein bestimmtes Zugriffsprotokoll angefragt werden.

2.1.1 Apache Kafka

Apache Kafka ist eine Event Streaming Plattform und dient dem Übertragen und Speichern von Daten in Form von Events. Komponenten von Apache Kafka sind insbesondere:

- Producer: Anwendungen, die Daten produzieren und diese in Form von Events an den Apache Kafka Broker senden.
- Consumer: Anwendungen die Events konsumieren und verarbeiten.
- Broker: Zentrale Komponente, die Events entgegennimmt, speichert und an Consumer verteilt.

Events werden über Topics organisiert. Über diese Topics werden die Events vom Producer verschickt und vom Consumer empfangen. Wie in Abbildung 2.2 abgebildet, registrieren (subscribe) sich Consumer für ein Topic beim Broker. Sobald ein Event von einem Producer über dieses Topic verschickt wird, verteilt der Broker dieses Event an alle Consumer, die sich für das Topic registriert haben [1].

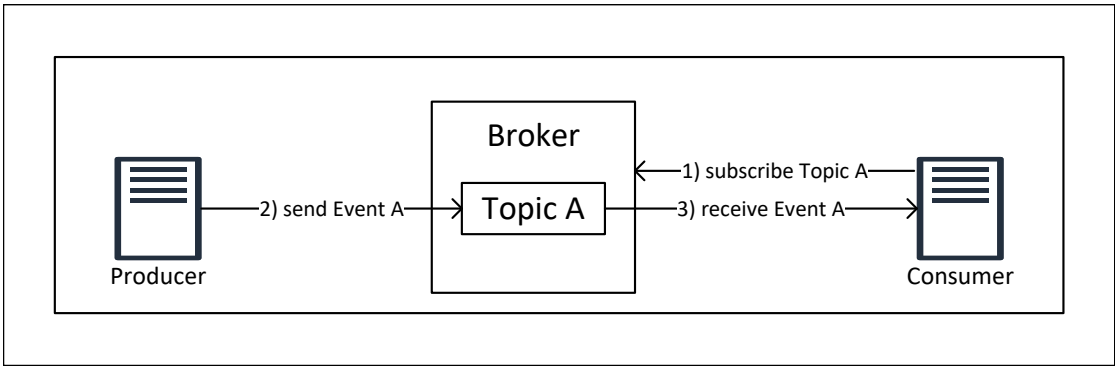

Abbildung 2.2: Arbeitsweise von Apache Kafka

2.1.2 Log-Nachrichten im dCache

Das dCache-Speichersystem verfügt über Monitoring-Funktionalitäten, welche Log-Nachrichten über Aktivitäten und den Zustand der Pools generieren und abspeichern. In diesem Abschnitt soll ein besonderer Fokus auf die Transfer-Events gelegt werden, die für jede Transaktion generiert werden, wenn auf Dateien zugegriffen wird.

Wenn eine Operation, wie ein Lesezugriff, im dCache-Speichersystem auf einer Datei durchgeführt wird, wird für diese Operation ein Transfer-Event generiert und gespeichert. In diesem Eintrag steht eine Vielzahl von Informationen über den Dateizugriff, wie z.B. Zeitpunkt des Zugriffs, Dateipfad, User-ID, gelesene Bytes, etc. Die Einträge variieren je nach Operation und verwendetem Zugriffsprotokoll [8]. Um die Transfer-Events zu persistieren und sie zur weiteren Verarbeitung bereitzustellen, werden die Events im Betrieb beim Deutsche Elektronen Synchrotron (DESY) mittels Apache Kafka verteilt.

Die Transfer-Events werden vom dCache-Speichersystem an einen Apache Kafka Broker geschickt. Dieser verteilt die Transfer-Events an Consumer, wie in Abbildung 2.3 dargestellt ist. Einer der Consumer ist ein Logstash-Prozess, der die Transfer-Events mit weiteren Informationen erweitert und zu einem JavaScript Object Notation (JSON)-Objekt umwandelt. Ein weiterer Consumer wird in dieser Arbeit entwickelt und in Kapitel 3.4 vorgestellt. Ein Beispiel für ein Transfer-Event ist in Kapitel A.1 aufgeführt. Der logstash-Consumer legt die Events in seinem lokalen Speicher ab, von welchem sie im Billing[1]-Archiv persistiert werden [12].

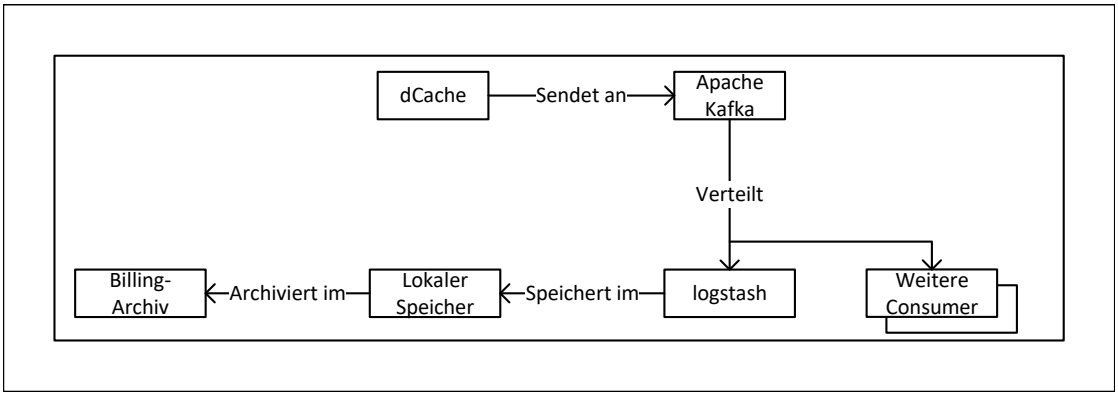

Abbildung 2.3: Schematischer Workflow zum Verteilen und Archivieren der Transfer-Events

Im Billing-Archiv sind die Transfer-Events tageweise in Dateien in einer Ordnerstruktur wie *{experiment}/{jahr}/{monat}/billing-{jahr}-{monat}-{tag}.json* gespeichert. In einer solchen Datei sind alle Transfer-Events eines Tages für das entsprechende Experiment gespeichert.

[1] Billing ist eine alternative Bezeichnung für die Transfer-Events

pool-info-Daten

Neben den Transfer-Events werden beim DESY pool-info-Daten generiert. Diese werden nicht standardmäßig von dCache generiert, sondern nur in den vom DESY verwendeten dCache-Instanzen erzeugt. Diese pool-info-Daten werden auch im JSON-Format erstellt und ähnlich wie in Abbildung 2.3 über Apache Kafka versandt und von Consumern verarbeitet. Die *pool-info-Daten* werden alle 15 Minuten von jedem dCache-Pool generiert und enthalten pool-spezifische Informationen, wie die aktuelle Speicherauslastung oder die aktuelle Anzahl an Movern. Sie werden genutzt, um den aktuellen Zustand der dCache-Pools zu erfassen.

2.2 Data Mining

Wie in Kapitel 1.2 beschrieben, sollen aus den Transfer-Events und pool-info-Daten Muster extrahiert werden, um parallele und verfügbarkeitsgefährdende Dateizugriffe auf Dateien zu identifizieren. Um in der Menge an Daten Muster zu identifizieren, eignen sich die Methoden und Vorgehensweisen des Data Minings. Data Mining dient dazu, Wissen aus Daten zu extrahieren, wobei Wissen als interessantes Muster, welches allgemeingültig ist, definiert ist [26]. In diesem Kapitel werden die Schritte des KDD-Verfahrens näher erläutert und es wird das Klassifikationsverfahren *logistische Regression* [18] vorgestellt mit welchem ein Klassifikations-Modell erstellt wird.

2.2.1 KDD

Das Verfahren KDD beschreibt einen Prozess, bei dem mittels Data Mining Methoden Wissen in Daten gefunden werden kann. Vor der Wissengewinnung sind die ersten drei Schritte des KDD-Prozesses die Selection, das Pre-Processing und die Transformation. Die Schritte sind vorbereitende Schritte zur Aufbereitung und Transformation von Daten [17].

Selection

Wie in Abbildung 2.4 zu erkennen ist, startet der KDD-Prozess mit Daten bzw. Rohdaten. Aus diesen Daten muss ein Dataset erstellt werden. Im Schritt Selection wird

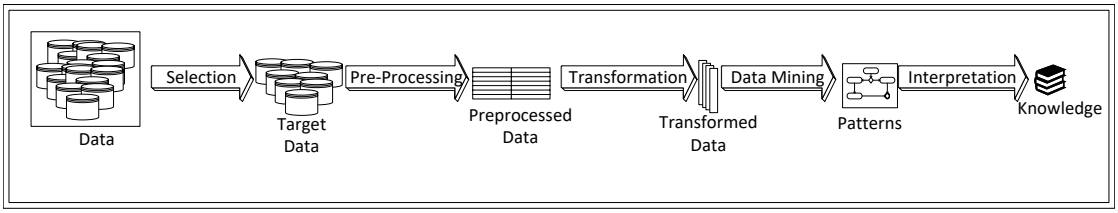

Abbildung 2.4: Überblick über die Schritte des KDD-Prozesses

eine Menge aus den zur Verfügung stehenden Daten ausgewählt, die betrachtet werden sollen. Wenn in den Daten verschiedene Variablen zur Verfügung stehen, kann in der Selection eine Teilmenge der Variablen ausgewählt werden, welche analysiert werden. Die Selection liefert als Resultat Zieldaten zurück, die im nächsten Schritt weiterverarbeitet werden [17]. Welche Daten ausgewählt werden, hängt vom Anwendungsfall und dem Wissen über die Daten ab. Im Fall dieser Arbeit haben die Administratoren des dCache-Speichersystems das meiste Wissen über die Interpretation der Transfer-Events und pool-info-Daten, weshalb dieses Wissen für die Auswahl der Zieldaten relevant ist.

Pre-Processing

Der Pre-Processing-Schritt dient dazu, die Daten des Datasets zu bereinigen oder zu erweitern. Systematische Effekte können dazu führen, dass das Ergebnis verfälscht wird. Oder es gibt Werte, die sich signifikant vom Mittel der Datenpunkte unterscheiden, sogenannte Ausreißer. Diese Effekte und Werte können verschiedene Ursachen haben. So kann es bei der manuellen Dateneingabe zu Fehlern kommen, Daten werden bei der Übertragung falsch interpretiert oder konvertiert oder ein Messinstrument ist falsch kalibriert. Neben den Ausreißern und systematischen Effekten können auch ungültige Daten vorliegen. Diese befinden sich beispielsweise außerhalb eines gültigen Wertebereiches oder in Datensätzen fehlen Merkmale komplett [26].

Auf die verschiedenen Probleme kann unterschiedlich reagiert werden. Im Folgenden wird ein exemplarischer Auszug von Handlungsempfehlungen vorgestellt [26]:

- **Fehlerwert:** Für alle ungültigen oder fehlenden Werte eines Datensatzes wird ein Fehlerwert eingesetzt. Wenn alle validen Werte größer 0 sind, könnte ein Wert kleiner als 0 als Fehlerwert interpretiert werden.

- **Korrektur oder Schätzung**: Bei ungültigen Daten kann der Wert ersetzt, bei fehlenden Daten geschätzt werden. Als Ersatz können beispielsweise Mittelwert, Median, Maximal- oder der Minimalwert der Variable über alle korrekten Daten gebildet und eingesetzt werden.
- **Entfernen vom Datensatz:** Wenn mindestens eine Variable eines Datensatzes fehlt oder ungültig ist, wird der gesamte Datensatz entfernt.

Transformation

Der Transformations-Schritt dient dem Umwandeln der Daten in eine Form, in der sie vom Data Mining Verfahren verwendet werden können. Beispiele für solche Transformationen sind die Aggregation und die Standardisierung von Daten. Die Aggregation von Daten wird verwendet, wenn man eine Menge von Daten zusammenfassen möchte, die ansonsten in mehreren Datensätzen dargestellt sind. Die Art der Aggregation hängt von den vorliegenden Daten ab und davon, was die daraus resultierende Kenngröße ausdrücken soll.

Unter Umständen haben die Daten unterschiedliche Wertebereich oder Einheiten. Dies kann dazu führen, dass Analyseresultate verfälscht werden. Um Merkmale miteinander vergleichbarer zu machen, können die Merkmale standardisiert werden und in einen anderen Wertebereich transformiert werden. Es gibt eine Vielzahl an Transformationen. In dieser Arbeit wird die $\mu - \sigma-$Transformation, auch z-Standardisierung genannt, verwendet [26]. Die z-Standardisierung überführt die Daten einer Variable in einen Wertebereich, in dem der Mittelwert μ den Wert null hat und eine Standardabweichung von eins genau ein σ vom Mittelwert entfernt ist. Um den standardisierten Wert zu berechnen, wird für die betrachtete Dimension i eines Datensatzes D der Wert σ_i und μ_i gebildet. Dadurch kann für einen Datenpunkt k der Dimension i der neue Wert z_s durch Gleichung 2.1 berechnet werden.

$$z_s = \frac{k - \mu_i}{\sigma_i}, k \in D_i \tag{2.1}$$

Data Mining

Im vorletzten Schritt, dem Data Mining, werden aus den transformierten Daten durch Anwendung von Data Analysis Verfahren Modelle entwickelt [17]. Die Verfahren unter-

scheiden sich in ihrem Ziel und in den Daten, auf denen sie angewendet werden können. Die Klasse von Verfahren, die in dieser Arbeit von Relevanz sind, sind Klassifikationsverfahren. Diese Verfahren sind *supervised*, das bedeutet die Daten sind mit einer Klassenzugehörigkeit versehen. Ziel ist es, anhand von Trainingsdaten ein Modell zu entwickeln, welches für einen neuen Datensatz eine Klassenzugehörigkeit ermittelt [25]. Verfahren die *unsupervised* sind, lernen die Muster von ungelabelten Daten. Ein Beispiel dafür sind Clustering-Verfahren, welche Daten Gruppen von Daten zuordnen. [18]. In Kapitel 2.2.2 wird die *logistische Regression* beschrieben, die in dieser Arbeit als Klassifikationsverfahren verwendet wird.

Interpretation

Wie in Abbildung 2.4 zu sehen, ist der letzte Schritt des KDD-Prozess die Interpretation bzw. die Evaluation. Dieser letzte Schritt dient dazu die Muster, die mit Data Mining Methoden entdeckt worden sind, zu bewerten und ggf. den Prozess zu wiederholen. Neben der Interpretation des numerischen Ergebnisses kann die Visualisierung der Daten Bestandteil dieses Schrittes sein. Zur Bewertung der Ergebnisse bzw. der visualisierten Ergebnisse ist häufig ein Verständnis der betrachteten Daten und der Domäne notwendig [17][26].

2.2.2 Logistische Regression

Das in dieser Arbeit verwendete Data Mining Verfahren ist die logistische Regression. Die logistische Regression ist ein Klassifikationsverfahren, das mit Wahrscheinlichkeiten arbeitet und aus der *linearen Regression* heraus abgeleitet ist. Die lineare Regression nähert eine n-dimensionale Funktion bestmöglich an gegebene Datenpunkte an. Die anzunähernde Funktion hat die Form:

$$y(X) = b_0 + b_1 x_1 + ... + b_n x_n \tag{2.2}$$

Mit der Methode der kleinsten Quadrate wird die beste Anpassung an die Daten geliefert. Zur genaueren Funktionsweise der linearen Regression sei auf die Literatur in [20] verwiesen. Die Funktion bildet einen Datenpunkt auf ein Intervall von $[-\infty, \infty]$ ab. Um für einen Datenpunkt Wahrscheinlichkeiten abzubilden, muss die Zielmenge auf das Intervall

$[0, 1]$ eingeschränkt werden. Um dies zu erreichen, kann die lineare Regressionsgleichung in die logistische Funktion transformiert werden und man erhält:

$$p(y) = \frac{1}{1 - e^{-y}} = \frac{e^{y}}{1 + e^{y}} = \frac{e^{b_0 + b_1 x_1 + \ldots + b_n x_n}}{1 + e^{b_0 + b_1 x_1 + \ldots + b_n x_n}} \tag{2.3}$$

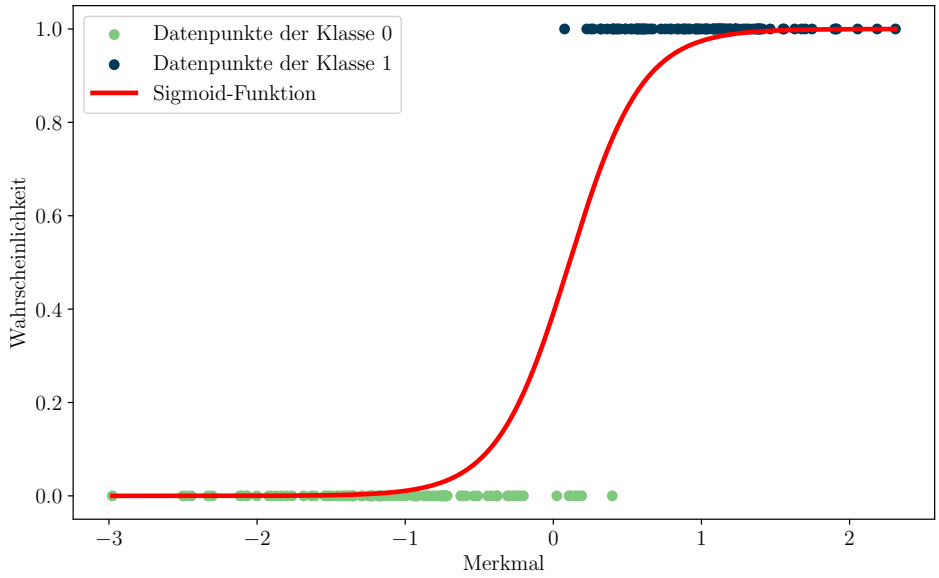

Abbildung 2.5: Exemplarische Darstellung einer logistischen Regression mit einem Merkmal

Eine beispielhafte Darstellung im zwei-dimensionalen Koordinatensystem ist in Abbildung 2.5 gegeben. Die in dieser Abbildung dargestellten Datenpunkte sind einer der binären Klassen 0 oder 1 zugeordnet. Mit entsprechenden Trainings-Datenpunkten werden die Koeffizienten b_0 und b_1 trainiert. Eine Veränderung von b_0 bewirkt eine Verschiebung entlang der x-Achse und eine Veränderung von b_1 führt zu einem schnelleren bzw. langsameren Anstieg der logistischen Funktion [22]. Die Funktion p wird mit der *Maximum Likelihood* Methode trainiert, welche in [18, S. 137] beschrieben ist. Nach dem Training des Klassifikators kann mit der Funktion die Wahrscheinlichkeit bestimmt werden, dass ein gegebener Datenpunkt zu Klasse 1 gehört. Mit der entsprechenden Wahrscheinlichkeit

kann daraufhin mit einer Funktion K die Klassenzugehörigkeit bestimmt werden:

$$K(y,s) = \begin{cases} 1 & p(y) > s \\ 0 & p(y) \leq s \end{cases} \tag{2.4}$$

Der Parameter s ist in der verwendeten `scikit-learn`-Bibliothek auf $0,5$ gesetzt, aber wenn man beispielsweise False-Negatives komplett ausschließen möchte, kann man den Parameter s entsprechend anpassen, damit die Klassenzuteilung erst ab einer anderen Wahrscheinlichkeit stattfindet.

2.2.3 Evaluationsmetriken

Für die Trainings- und Testdaten[2] eines Klassifikators kann eine Confusion Matrix erstellt werden. Diese Matrix besteht aus 4 Feldern, wie in Tabelle 2.2 dargestellt ist.

- **True Positive:** Anzahl der Datensätze, die das Label Positive haben und für die dieses Label vorhergesagt wurde.
- **True Negative:** Anzahl der Datensätze, die das Label Negative haben und für die dieses Label vorhergesagt wurde.
- **False Positive** Anzahl der Datensätze, die das Label Negative haben, für die aber das Label Positive vorhergesagt wurde.
- **False Negative**: Anzahl der Datensätze, die das Label Positive haben, für die aber das Label Negative vorhergesagt wurde.

	Vorhergesagte Klasse	
Echte Klasse	Positiv	Negativ
Positiv	True Positive (TP)	False Negative (FN)
Negativ	False Positive (FP)	True Negative (TN)

Tabelle 2.2: Aufbau einer Confusion Matrix

Aus den vier Komponenten der Confusion Matrix lassen sich weitere Kenngrößen ableiten, welche zur Evaluation und dem Vergleich von Klassifikatoren dienen.

[2] Das Dataset wird in Daten zum Trainieren und zum Testen aufgeteilt.

Die **Accuracy** (Genauigkeit) gibt an, wie viele Datensätze richtig klassifiziert werden. Die Berechnung der Accuracy ist in Gleichung 2.5 dargestellt.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Ein hoher Accuarcy-Wert sagt aus, dass viele Datensätze dem richtigen Label zugeordnet worden sind. Dies kann bei Datasets, die eine hohe Klassen-Ungleichheit haben, dazu führen, dass ein simpler Klassifikator, der allen Datensätzen das Klassenlabel der überwiegenden Klasse zuweist. Dies führt zu einem hohen Accuracy-Wert besitzt, aber es wird nicht klassifiziert [16, S. 876].

Zwei weitere Metriken, die zur Evaluation von Klassifizierern verwendet werden, sind die **Precision** und der **Recall**, auch Sensitivität genannt [16, S. 878]. Der Recall ergibt sich aus Gleichung 2.6 und sagt aus, welcher Anteil positiver Datensätze richtig klassifiziert wurde. Diese Metrik ist insbesondere dann wichtig, wenn negativ klassifizierte positiv Datensätze schwerere Auswirkungen haben, als positiv klassifizierte negativ Datensätze.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Die Precision gibt an, wie hoch der Anteil an wirklich positiven Datensätzen innerhalb der positiv klassifizierten Datensätze ist. Sie wird wie in Gleichung 2.7 berechnet. Zwischen beiden Zielgrößen besteht ein Zielkonflikt, da mit steigender Precision ein niedriger Recall akzeptiert wird und umgekehrt. Beide Werte lassen sich in einer Precision-Recall-Kurve (PR-Kurve) auftragen [18, S. 86-92].

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

Eine PR-Kurve, wie in Abbildung 2.6, gibt das Verhältnis zwischen der Precision und dem Recall bei verschiedenen Schwellwerten, in der Funktion K, an der Klassifikation an. Anhand dieser Kurve kann der Schwellwert gewählt werden, um das beste Ergebnis zwischen Recall und Precision zu erhalten oder um einen der beiden Werte zu maximieren und für die andere Kenngröße den höchsten Wert zu erhalten. Der markierte Punkt gibt den höchsten Recall mit dem bestmöglichen Precision-Wert an.

Zum Vergleich des verwendeten Klassifikations-Modell wird ein Baseline-Modell benötigt. Mit dem Baseline-Modell kann der Fortschritt des Klassifikations-Modells mit ei-

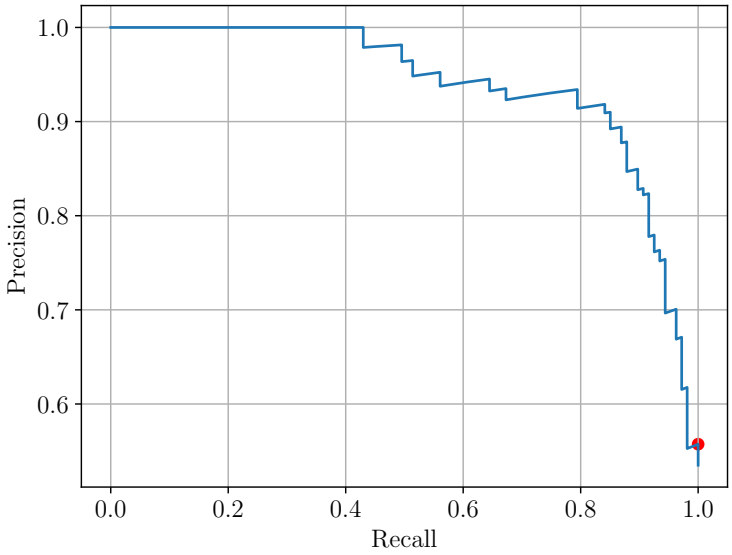

Abbildung 2.6: Beispielhafte PR-Kurve mit markiertem maximierten Recall-und höchstmöglichen Precision-Wert

nem einfacheren Modell verglichen werden [13]. Für das bestehende Problem besteht kein existierendes Modell, weshalb Klassifizierer verwendet werden, die die Inputs ignorieren.

Verschiedene solcher Klassifizierer werden in der `scikit-learn`-Bibliothek als `DummyClassifier` angeboten. Über diese Bibliothek können verschiedene Strategien zum Verhalten des Klassifizierers ausgewählt werden. Für diese Arbeit wird die Strategie `constant` genutzt um einen hohen Recall zu ermöglichen und die daraus resultierende Precision mit der des Modells vergleichen zu können. Dies soll validieren, dass das Modell besser klassifiziert als ein simpler Input, der alle Intervalle als positiv klassifiziert.

2.3 MAPE-Loop

Um den trainierten und getesteten Klassifizierer nutzen zu können, wird dieser in ein Software-Tool eingebunden. Das Software-Tool ist als MAPE-Loop aufgebaut. MAPE-Loops werden zur Implementierung von selbstadaptiven Systemen (self-adaptive systems) verwendet, welche ihr eigenes Verhalten auswerten und anpassen, wenn es nicht dem gewünschten Verhalten entspricht. Adaptivitäts-Eigenschaften werden auch als *self-*

* Eigenschaften bezeichnet, welche unterschiedliche Merkmale eines Systems beschrieben und auf verschiedenen Hierarchiestufen angeordnet sind [27, S. 2-6]. Die zur Implementierung von adaptiven Systemen eingesetzten MAPE-Loops bestehen aus vier Schritten (Abbildung 2.7), welche nacheinander durchlaufen werden [27, S. 9].

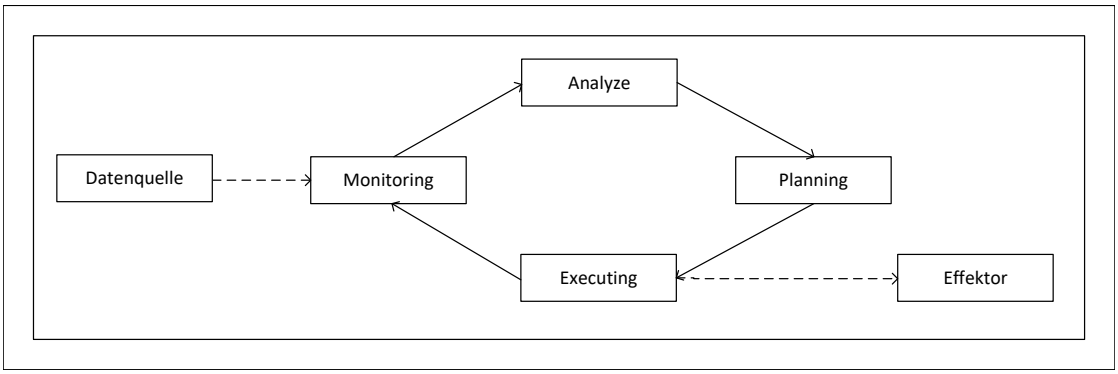

Abbildung 2.7: Darstellung des MAPE-Loops

Monitoring-Phase
Die Monitoring-Phase dient dem Sammeln und dem Konvertieren von Daten aus Datenquellen. Die zusammengestellten Daten werden der Analyze-Phase zur Datenanalyse gegeben.

Analyze-Phase
Die Analyze-Phase analysiert die gesammelten Daten. Für die Analyse der Daten können beispielsweise Data Mining Techniken oder andere Modell-basierte Verfahren verwendet werden.

Planning-Phase
Die Planning-Phase entscheidet was ausgeführt werden muss und wie es ausgeführt wird. Dafür werden verschiedene Möglichkeiten verglichen und die Möglichkeit mit dem besten Ausgang wird verwendet.

Executing-Phase
Die von der vorherigen Phase ausgewählten Aktionen werden von der Executing-Phase ausgeführt. Dies beinhaltet das Ausführen definierter Workflows und dem Konvertieren von Aktionen zu dem was von den Effektoren benötigt wird.

3 Methodik

Im anschließenden Kapitel soll die Methodik vorgestellt werden, mit der Muster in Transfer-Events und pool-info-Daten identifiziert werden, die auf ein verfügbarkeitsgefährdendes Nutzerverhalten hindeuten. Dieses Kapitel stellt zu Beginn den bisherigen Prozess der dCache-Administratoren vor. Im Anschluss wird mittels der logistischen Regression mit den ermittelten Kenngrößen ein Modell entwickelt. Zum Abschluss wird die Architektur des für diese Arbeit entwickelten Software-Tools `GROGU` vorgestellt, welches im Betrieb verwendet wird, um verfügbarkeitsgefährdende Zugriffe festzustellen und dies zu melden.

3.1 Lösungsansatz

In dieser Arbeit wird mit dem KDD-Verfahren gearbeitet, welches in Kapitel 2.2.1 vorgestellt wurde. Neben dem KDD-Verfahren existieren noch weitere Verfahren, um Wissen aus Daten zu gewinnen. Ein anderes populäres Verfahren ist das Cross-Industry Standard Process for Data Mining (CRISP-DM)-Verfahren. Bei diesem handelt es sich um einen Standard zum Data Mining, bestehend aus sechs Phasen. CRISP-DM und KDD weisen in ihren Phasen und dem Vorgehen Ähnlichkeiten auf. Wie in [14] dargestellt ist, handelt es sich bei CRISP-DM um eine Implementierung von KDD und die Schritte des CRISP-DM-Vorgehens können auf Schritte des KDD-Verfahrens abgebildet werden. Aufgrund dieser Ähnlichkeit wurde, wurde das generellere KDD-Verfahren als Vorgehensmodell gewählt.

Im KDD-Schritt Data Mining können verschiedene Verfahren zur Klassifikation verwendet werden. Ein Verfahren sind rekurrente neuronale Netze (RNN) [19, S. 165,185]. Dies sind neuronale Netze, die sich für den Einsatz von Zeitreihen eignen. Zeitreihen bzw. diskrete Zeitreihen sind Daten, die in regelmäßigen Intervallen über ein System gewonnen werden und mit einem Zeitpunkt versehen sind. Die vorliegenden Daten werden in

15-minütigen Intervallen verarbeitet, die durch RNNs verarbeitet werden könnten. Neben den RNNs existieren noch weitere klassische Verfahren zur Klassifikation, wie die Support Vector Machine (SVM) oder k-Nearest Neighbour (KNN)-Verfahren [18]. Es wird dennoch die logistische Regression verwendet, da es sich bei dieser um ein einfach anzuwendendes Verfahren handelt, welches in der Vergangenheit für ähnliche binäre Aufgaben wie die Klassifikation von Spam eingesetzt wurde [21]. Durch die Verwendung der logistischen Regression wird in dieser Arbeit gezeigt, ob sich dCache-Daten für diese Problemstellung eignen und ob es möglich ist überhaupt eine Klassifikation durchzuführen. Weitere Forschung ist notwendig um einen optimalen Klassifikator zu identifizieren.

Weiterhin werden die Administratoren im Falle eines verfügbarkeitsgefährdenden Zugriffes benachrichtigt, um entsprechende Schritte zu unternehmen. Obwohl das Software-Tool als MAPE-Loop modelliert wird, welcher ein selbst-adaptives Vorgehen vorsieht, wird in dieser Arbeit auf ein solches Anpassen verzichtet und es werden für weitere Handlungen die Administratoren benachrichtigt. Damit werden die letzten beiden Schritte des MAPE-Loops anders umgesetzt als in der Literatur vorgesehen. Aufgrund von internen Regularien des DESY's kann keine selbst-adaptive Anpassung durchgeführt werden. In Kapitel 5.2.3 werden Alternativen vorgestellt, wie auf verfügbarkeitsgefährdendes Nutzerverhalten reagiert werden kann, das durch parallele Dateizugriffe ausgelöst wird. Es wurde sich für den MAPE-Loop als Architekturstil entschieden, um die Grundlage für eine Software zu legen, welche die Möglichkeit hat, um selbst-adaptive Pläne ergänzt zu werden. Durch die gewonnene Flexibilität und Erweiterbarkeit des Software-Tools kann selbst-adaptives Verhalten in künftigen Veröffentlichungen des Tools erweitert werden.

3.2 Manueller Detektionsprozess

Im bisherigen Prozess zum Erkennen von verfügbarkeitsgefährdendem Nutzerverhalten prüfen die dCache-Administratoren manuell verschiedene Metriken und leiten aus diesen die Ursache ab. Ein Administrator überprüft regelmäßig die Metrik in Abbildung 3.1 und entscheidet, ob es eine ungewöhnlich hohe Anzahl an Nachrichten gibt, in denen gemeldet wird, dass ein NFS-Client keine Verbindung mehr zu einem NFS-Server hat. Ab wann ein Zeitraum als verfügbarkeitsgefährdend gilt, ist nicht genau festgelegt und basiert auf der Erfahrung der Administratoren. Laut deren Aussage gelten einige tausend Meldungen bereits als kritisch und erfordern weitere Maßnahmen. Im Laufe dieser Arbeit werden mehrere hunderttausend Meldungen beobachtet.

Eine Unterbrechung kann verschiedene Ursachen haben. Gängige Ursachen sind:

- Paralleler Zugriff auf Dateien innerhalb kurzer Zeit.
- Verteilung der Dateien: Unterschiedliche Dateien, die gleichzeitig, aber nicht ungewöhnlich häufig gelesen werden, befinden sich auf dem gleichen Pool, wodurch es zu langen Ladezeiten kommt.
- Ausfall eines dCache-Pools: Der Ausfall eines dCache-Pools kann verschiedene Ursachen, wie Hardware-Ausfälle oder Netzwerk-Fehler, haben.

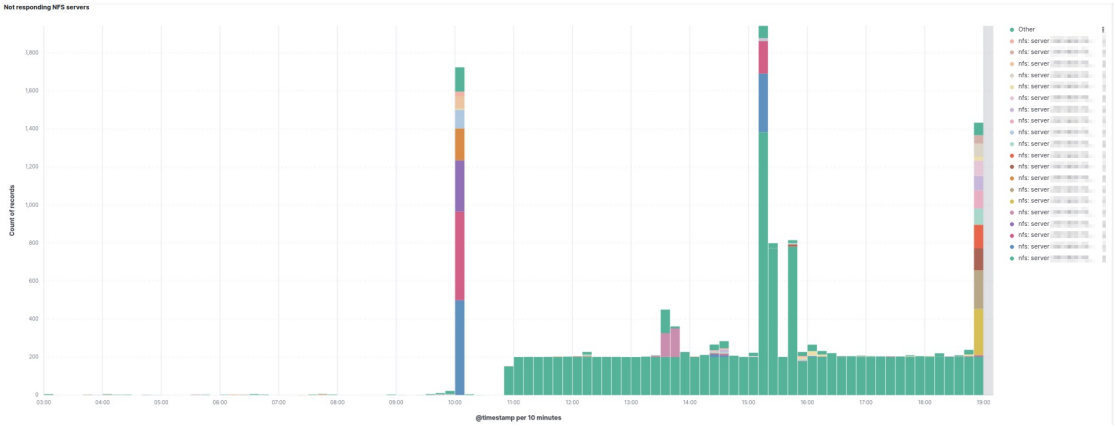

Abbildung 3.1: Anzahl an Einträgen in System-Logdateien, die auf Verbindungsprobleme zwischen NFS-Client und Server hindeuten, werden regelmäßig in einem Kibana-Dashboard überprüft

Wenn ungewöhnlich viele Nachrichten zu Verbindungsproblemen erkannt werden, werden verschiedene Kenngrößen betrachtet, wie die Anzahl an Transfers pro Datei oder die Gesamtanzahl an Transfers. Einige dieser Kenngrößen werden im folgenden KDD-Prozess verwendet. Anhand dieser Kenngrößen versuchen die Administratoren die Ursache zu finden. Sobald die Ursache ermittelt ist und ein Nutzer als Ursache für die Verbindungsprobleme ausgemacht werden konnte, wird dieser kontaktiert. Weiterführende Maßnahmen, wie das Sperren eines Nutzers, sind die Ausnahme und müssen von Seiten der Administratoren begründet werden.

Dieses Vorgehen birgt verschiedene Schwachstellen gegenüber einem automatisierten Verfahren. Einerseits müssen diese Metriken kontinuierlich von einem Administrator ausgewertet werden, da jeder Zeitraum potenziell schädlich ist. Vorkommnisse, die zu ungewöhnlichen Uhrzeiten auftreten, wie am Wochenende oder am späten Abend, werden oft

erst Stunden oder gar Tage später entdeckt. Außerdem kann die Abwesenheit eines Administrators dazu führen, dass Events gar nicht erkannt werden. Darüber hinaus bindet ein regelmäßiges Überprüfen der Metriken zeitliche Ressourcen. Daher soll dieser manuelle Prozess durch ein automatisiertes Verfahren ersetzt werden.

3.3 KDD

Um aus den Transfer-Events und den pool-info-Logs Wissen zu extrahieren, wird der in Kapitel 2.2.1 beschriebene KDD-Prozess verwendet. Dafür werden die Schritte auf Transfer-Events und pool-info-Logs angewendet und mit den vorverarbeiteten Daten ein Modell basierend auf der logistischen Regression entwickelt.

3.3.1 Umsetzung des KDD-Prozesses

Um den KDD-Prozess umzusetzen, wird in erster Linie auf die Daten im Billing-Archiv zugegriffen. Die Dateien, in denen die Transfer-Events tageweise abgelegt sind, können einige Gigabyte (GB) an Daten beinhalten. Um die Menge an Daten analysieren zu können, werden die ersten drei Schritte des KDD-Prozesses mittels Apache Spark durchgeführt [2]. Dies ist eine Open-Source Engine zur Datenverarbeitung von großen Datenmengen, welche auf der Recheninfrastruktur des DESYs betrieben wird. Um diese zu nutzen, wird ein Jupyter Notebook verwendet, in dem in Python die Schritte programmiert sind. Die aggregierten Daten werden in einer Comma-separated Value (CSV)-Datei gespeichert. Die Erstellung des Modells wird auf Basis der Daten lokal durchgeführt.

3.3.2 Selektion

Als Datengrundlage dienen Transfer- und pool-info-Logs der dCache-Instanzen des DESYs. Der Aufbau dieser Daten ist in Kapitel 2.1.2 näher erläutert. Das Ziel ist es, ein Dataset zu erstellen, welches vom folgenden KDD-Schritt verarbeitet werden kann. In den Transfer-Daten befinden sich neben den `transfer`-Einträgen auch andere Events. Diese werden nicht betrachtet und werden aus den Daten entfernt. Außerdem werden alle Einträge entfernt, die eine Schreib-Operation darstellen. Damit werden nur `transfer`-Events, die eine Lese-Operation, darstellen betrachtet. Darüber hinaus werden die Einträge entfernt, in welchen ein anderes Protokoll als NFS verwendet wurde. Der Verlust

dieser Merkmale und Datensätze wird in Kauf genommen, da dadurch die zu verarbeitende Datenmenge verkleinert wird und für das betrachtete Problem die aktuell beste Grundlage bildet.

Auswahl der betrachteten Zeiträume

Wie in Kapitel 2.1.2 beschrieben ist, sind im Billing-Archiv die vergangenen Transfer-Events und pool-info-Daten persistiert. Um mit diesen Daten ein Modell zu trainieren, müssen sie mit einem *Klassen-Label* versehen werden. Es werden in dieser Klassifikationsaufgabe zwei Klassen-Labels verwendet:

- Klasse 0: In diesem Zeitraum kam es zu keinem Ausfall, der durch parallele Dateizugriffe ausgelöst wurde.
- Klasse 1: Es kam zu verfügbarkeitsgefährdendem Verhalten aufgrund von parallelen Dateizugriffen.

Die Daten werden in 15-minütige Intervalle unterteilt. Diesen Intervallen wird ein entsprechendes Label zugewiesen. Um einem solchen Zeitraum ein Label zuzuweisen, werden interne Tools der DESY-Administratoren ausgewertet. Wenn ein Administrator eine erhöhte Anzahl an Verbindungsproblemen zwischen NFS-Clients und -Servern feststellt, wird es in dem Tool gemeldet. Anhand der darauf folgenden Meldungen kann der Vorfall einer der in Kapitel 3.2 erläuterten Ursachen zugeordnet werden. Der Zeitraum wird als *Klasse 1*-Datenpunkt gespeichert. Mittels dieser Methode konnten 158 Datenpunkte im Zeitraum von April 2023 bis Juni 2023 ermittelt werden. Diese Datenpunkte gehören zu Transfer-Events und pool-info-Daten der Experimente CMS oder ATLAS.

Zur Generierung von Klasse 0 Daten werden 1000 Zeitpunkte aus dem gleichen Zeitraum ausgewählt, die nicht in der Menge der Klasse 1 Zeiträume liegen. Diesen Zeiträumen wird zufällig eines der beiden Projekte zugewiesen. Die Auswahl der Zeiträume zusammen mit den Labels und den entsprechenden Projekten ist in einer CSV-Datei abgelegt. Es sind 1000 Repräsentanten ausgewählt, da für 1000 Zeitpunkte noch in annehmbarer Zeit die Werte berechnet werden können. Bei 33 Datensätzen der Klasse 0 waren keine Daten vorhanden, weshalb diese entfernt worden sind.

Das vorliegende Dataset besteht damit zum Großteil aus Datensätzen, die zur Klasse 0 gehören. Die Datensätze machen damit nur rund 15% im Dataset aus. Bei einem

solchen Ungleichgewicht zwischen den Klassen kann man das vorliegende Dataset als *Imbalanced Dataset* bezeichnen [16]. Dies muss bei der Auswahl der Evaluationsmetriken berücksichtigt werden.

Auswahl der Felder in den vorliegenden Daten

Zur Erkennung des Nutzerverhaltens werden verschiedene Kenngrößen aus den Transfer-Events verwendet und aggregiert. Aus der Menge der verwendbaren Felder wird im Folgenden erläutert, welche Einträge in den Tranfer-Events in Betracht gezogen werden um sie zu aggregieren und in eine Klassifikation mit einzubeziehen. Als Grundlage für die Auswahl dient das Wissen der dCache-Administratoren beim DESY. Die meisten Kenngrößen werden im aktuell manuellen Prozess, der in Kapitel 3.2 beschrieben ist, verwendet.

Parameter	**Beschreibung**
nfs_mover	Für jeden Transfer wird ein `nfs_mover` gestartet. Eine hohe Anzahl an NFS-Movern zu einem Zeitpunkt indiziert, dass viele Dateien im dCache-System angefordert wurden.
billingPath	Der Pfad der Datei, die im Transfer angefordert wird.
uid	Der User-Identifier des Nutzers, der eine Datei in einem Transfer anfordert.
fileSize	Die Größe der angeforderten Datei in Byte.
transfersize	Die Anzahl an übertragenen Byte.
readBandwidth	Wie viele Byte pro Sekunde gelesen wurden.
transferTime	Dauer des Transfers in Millisekunden (ms).

Tabelle 3.1: Beschreibung der ausgewählten Parameter aus den Transfer-Events und pool-info-Daten

Die in Tabelle 3.1 beschrieben Parameter sind die ausgewählten Einträge aus den Transfer-Logs, welche in Abstimmung mit den dCache-Administratoren als gewinnbringende Ausgangsbasis für eine weitere Analyse verwendet werden können. Einzig der Parameter `nfs_mover` wird nicht aus den Transfer-Events gewonnen, sondern aus den pool-info-Daten entnommen. Darüber hinaus wird der Parameter `uid` nicht zur Klassifikation, sondern zur Zuordnung von Zugriffen zu einem Nutzer verwendet.

3.3.3 Preprocessing der ausgewählten Daten

Um die ausgewählten Daten in die gewünschte Form zu transformieren bzw. aggregieren zu können, müssen bestimmte Werte behandelt werden. Dafür sind in Kapitel 2.2.1 verschiedene Verfahren aufgeführt, mit denen man mit entsprechenden Werten umgehen kann. In diesem Abschnitt wird beschrieben, wie entsprechende Werte für die unterschiedlichen Felder in den Transfer-Events und pool-info-Daten behandelt werden.

Für die Variablen `billingPath`, `fileSize`, `transferSize`, und `transferTime` gab es in keinem betrachteten Transfer-Event fehlende oder ungültige Werte. Daher wird für diese Einträge kein Pre-processing durchgeführt und sie werden im Folgenden nicht näher erläutert. Einige Merkmale müssen erst aus einem vorliegenden JSON-Objekt extrahiert werden. Dies ist in dem entsprechenden Schritt beschrieben.

nfs_mover

Die `nfs_mover` sind in einem JSON-Objekt mit Movern der anderen Protokolle gespeichert, wie in Listing 1 dargestellt ist. Die Anzahl an `nfs_movern` ergibt sich aus dem Eintrag `active` innerhalb des `nfs-q`-Objekts.

```
"queues": {
  "nfs-q": {
    "active": 0,
    "queued": 0
  }
}
```

Listing 1: Beispielhafter Eintrag der `NFS_mover` in einem pool-info-Datensatz

In einigen pool-info-Datensätzen fehlt der Eintrag `nfs-q` oder der Eintrag `active`. Für diese Fälle ist die Anzahl der `NFS_mover` mit 0 ersetzt.

uid

In einem Transfer-Event ist die `uid` Bestandteil des `UidPrincipal`-String, welcher wiederum im `subject`-Array enthalten ist (siehe Listing 2).

```
"subject": [
    "UidPrincipal[42]",
    "GidPrincipal[4242,primary]",
    "GidPrincipal[4242]",
    "Origin[127.0.0.1]"
]
```

Listing 2: `UidPrincipal`-String enthaltende `uid`

Um die `uid` zu erhalten muss geprüft werden, ob ein String im `subject` enthalten ist, der mit `UidPrincipal[` beginnt. Im positiven Fall wird die einzige Nummer aus diesem String selektiert. Sollte es keinen String geben, der mit `UidPrincipal[` beginnt, wird für diesen Datensatz der Wert -1 eingefügt.

readBandwidth

Die `readBandwidth` kann in einigen Tranfer-Events nicht vorhanden sein, wenn kein Byte gelesen wurde. In diesem Fall wird der Wert für `readBandwidth` auf null gesetzt.

3.3.4 Transformation der vorverarbeiteten Daten

Der folgende Teil erläutert die Transformation der ausgewählten und vorverarbeiteten Variablen. Dafür wird dargestellt, welche Variablen wie aggregiert werden. Abschließend wird erläutert, wie die transformierten Variablen standardisiert werden, um sie vergleichen zu können. Die Transformationen basieren zum einen auf bisherigen Variablen, die von den dCache-Administratoren betrachtet werden, oder sie werden als möglicherweise gewinnbringend betrachtet. Wie in [17, S. 30] beschrieben, können zwischen den KDD-Schritten Schleifen auftreten, wodurch innerhalb des *Transformation*-Schrittes in den *Pre-Processing*-Schritt gewechselt werden kann. Fast alle Variablen, die im Data Mining Schritt verwendet werden, müssen aus den bereits bestehenden Daten aggregiert werden. Ausschließlich die Anzahl an `nfs_mover` existiert bereits. Die transformierten Daten werden mit ihren Labels, ihren Zeiträumen und dem Experiment als Informationen in einer CSV-Datei gespeichert. Diese Datei wird als Eingabe zur Erstellung des Modells verwendet.

Anzahl aller Transfers

Um zu erkennen, ob eine Datei innerhalb eines Zeitraums mehrfach gelesen wurde, ist eine erste Variable die Anzahl aller Transfers. Dafür wird werden alle Transfers gezählt die für den Zeitraum vorhanden sind. Hinter dieser Variable steckt die Idee, dass wenn ein Nutzer eine Datei mehrfach liest, die Anzahl der Transfers für diese Datei steigt, aber auch, dass er andere Dateien oft liest, wodurch die Gesamtanzahl der Transfers ansteigt.

Die meistgelesene Datei

Für die folgende Transformation wird aus den Daten der `billingPath` benötigt. Nach diesem werden die Transfers sortiert und anschließend gezählt. Damit soll die Anzahl an Transfers pro angefragter Datei bestimmt werden. Aus dieser Zählung wird die Datei ermittelt, die in dem betrachteten Intervall am häufigsten gelesen wurde. Vorteil dieser Variable ist, dass explizit auf das Problem eingegangen wird, dass eine Datei häufig gelesen wird.

Analog dazu wird außerdem die Variable *Summe der Transfers der fünf am meisten gelesenen Dateien* betrachtet. Dafür wird nicht nur die Datei verwendet, die am meisten gelesen wurde, sondern es werden die Transfers der fünf am häufigsten gelesenen Dateien aufsummiert und als Kenngröße erstellt.

Aufsummierte Transferrate

In jedem Transfer-Event ist in der Variable `transfersize` enthalten, wie viele Bytes aus der im `billingPath` enthaltenen Datei übertragen wurden. Einerseits sagt eine hohe Transferrate aus, dass viele Daten übertragen wurden. Andererseits ist eine niedrige Transferrate kombiniert mit der *Anzahl aller Transfers* ein Indikator dafür, dass viele Transfers durchgeführt wurden, aber nicht viele Bytes aus den Dateien gelesen wurden. In Kapitel 1.1 ist darauf hingewiesen, dass ein Muster in der Vergangenheit war, dass einige Nutzer viele unterschiedliche Teile einer Datei lesen. Dies führt einerseits zu vielen Lesezugriffen auf einer Datei, als auch zu vielen Transfers, aber nicht zu einer hohen Transferrate. Um diesen Zusammenhang zu nutzen, wird die *Aufsummierte Transferrate* als Variable erstellt.

Median des Verhältnisses zwischen `transfersize` und `fileSize`

Um zu erkennen, ob das Dokument fragmentiert gelesen wird, wird das Verhältnis zwischen der `transfersize` und der `fileSize` gebildet. Ein niedrieger Wert weist darauf hin, dass in dem entsprechenden Transfer-Event ein kleiner Teil der Datei gelesen wurde. Wenn ein Intervall im Median einen niedrigen Wert aufweist, kann dies ein Indikator dafür sein, dass zusätzlich zu vielen Transfers unterschiedliche Teile der Datei angefordert werden und dies nicht aus dem dCache-Cache gelesen werden kann.

Standardisierung der Variablen

Die bisher vorgestellten Variablen werden über ein festes Intervall von 15 Minuten erstellt. Diese Werte werden für die Experimente CMS und ATLAS separat erstellt. Um die Intervalle vergleichen zu können, werden die CMS und ATLAS getrennt voneinander mittels der in Kapitel 2.2.1 vorgestellten z-Standardisierung standardisiert. Dafür werden die Zeiträume in zwei Mengen für CMS und für ATLAS geteilt. Für diese Mengen wird die z-Standardisierung durchgeführt und es wird eine gemeinsame standardisierte Menge von Daten gebildet. Die unterschiedliche Standardisierung wird durchgeführt, da CMS und ATLAS für die Variablen unterschiedlich hohe Mittelwerte in den betrachteten Daten aufweisen.

3.3.5 Data Mining

Anschließend soll auf den vorletzten Schritt des KDD-Prozesses, das Data Mining, eingegangen werden. Das zur Klassifikation verwendete Verfahren ist die logistische Regression (siehe Kapitel 2.2.2). Es wird die logistische Regression verwendet, da es sich bei dieser um ein etabliertes Verfahren handelt, welches häufig Einsatz findet, wenn zwischen zwei Zuständen unterschieden werden soll [15]. Beispiele für entsprechende Fragestellungen wie die Erkennung von E-Mail Spam oder die Korrektheit einer Steuererklärung [21][20] oder wie im vorliegenden Problem, ob ein verfügbarkeitsgefährdender Zugriff vorliegt oder nicht. Für das Training des Modells muss der Datensatz in einen Satz Trainingsdaten und einen Satz Testdaten unterteilt werden. 80% werden als Trainingsdaten verwendet und 20% als Testdaten [18, S. 30]. Die Daten werden aus der CSV-Datei eingelesen und aufgeteilt. Es wird die Python-Bibliothek `scikit-learn` verwendet, mit welcher ein Modell erstellt wird.

Bewertung des Klassifikationsmodells

Zur Bewertung des Klassifikationsmodells werden die in Kapitel 2.2.3 vorgestellten Metriken Precision und Recall verwendet. Aufgrund des Ungleichgewichts in den Klassen wird auf eine Verwendung der Accuracy verzichtet. Insbesondere der Recall ist für diese Arbeit eine wichtige Metrik, da mit ihr eine Aussage, darüber getroffen werden kann, wie hoch der Anteil an negativ klassifizierten positiv Datensätzen ist. Für das verwendete Dataset würde ein niedriger Recall-Wert bedeuteten, dass viele Ereignisse, bei denen verfügbarkeitsgefährdende Dateizugriffe stattfinden, als negativ eingestuft werden. Daher muss der Schwellwert s aus der Funktion K in Kapitel 2.2.2 so gewählt sein, dass der Recall-Wert maximiert wird und der Precision-Wert bei einem maximalen Recall-Wert so hoch wie möglich ist.

Um dieses Ziel zu erreichen, muss s so gewählt werden, dass aus allen möglichen Schwellwerten die Schwellwerte gewählt werden, bei denen der Recall-Wert eins ist. Aus dieser Teilmenge wird der Schwellwert mit dem höchsten Precision-Wert ausgewählt. In einer beispielhaften PR-Kurve entspricht das dem rot markierten Punkt in Abbildung 3.2. An der PR-Kurve kann man den bestmöglichen Punkt ablesen. Anhand der PR-Kurve wird

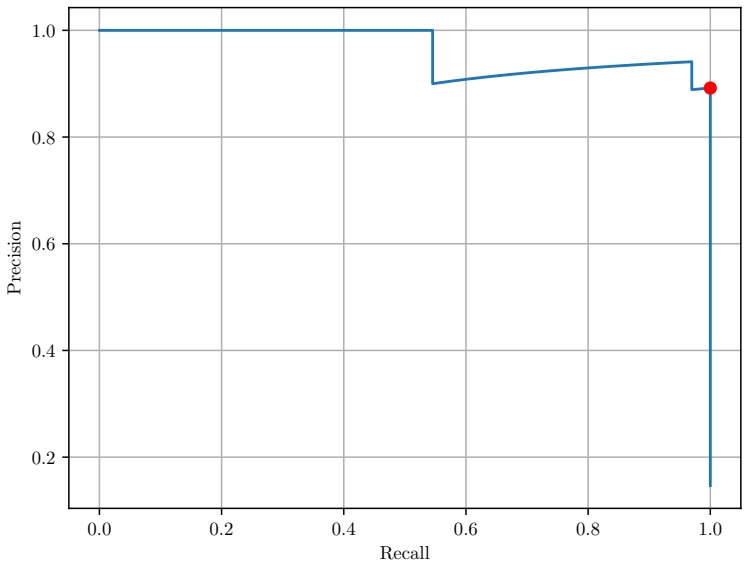

Abbildung 3.2: PR-Kurve des ausgewählten Modells mit markiertem maximierten Recall- und höchstmöglichen Precision-Wert

bei der folgenden Feature Selection der Punkt bestimmt, bei dem ein Recall von eins vorliegt und dabei die höchste Precision hat.

Feature Selection

Aus der Menge der Features muss das Feature *Anzahl der NFS Mover* direkt entfernt werden, da die `NFS_mover` in den betrachteten Daten ungewöhnlich stark schwanken. Dies ist laut Aussage der Administratoren auf alte, nicht mehr aktive Mover zurückzuführen. Die Schwankungen waren ein Bug, der in der dCache-Software ausgebessert wurde. Dennoch können die NFS-Mover nicht weiter betrachtet werden, könnten aber in Zukunft weiterhin eine interessante Kenngröße darstellen.

Aus den fünf zur Verfügung stehenden Features soll eine Teilmenge ausgewählt werden, mit der eine bestmögliche Klassifikation erreicht wird. Dazu muss eine Feature Selection, eine Auswahl der zur Verfügung stehenden Features, durchgeführt werden. Zwei Ansätze zur Auswahl von Features sind die forward selection und die backward elimination [23]. Bei der forward selection wird mit der leeren Menge als Feature Set begonnen und es werden sukzessive Features hinzugefügt. Bei der backward elimination wird mit allen Features begonnen und es werden allmählich Features entfernt. Die Güte eines Feature Sets wird mittels eines Kriteriums oder einer Funktion bestimmt, z.B. Höhe der Precision. Die Anzahl aller Teilmengen ist 2^n mit n als Anzahl der Features. Bei kleinen n können trotz des exponentiellen Wachstums alle Möglichkeiten probiert werden [23, S.283,284]. Zur Auswahl der Features werden in dieser Arbeit der Recall und die Precision verwendet. Es werden 32 Klassifizierer trainiert mit allen Variationen an Features. Mittels forward selection wird die Größe der genutzten Features schrittweise vergrößert.

Die Ergebnisse sind in Kapitel A.2 dargestellt. Die Klassifikatoren sind nach der Precision und dem Recall sortiert, sollte eine Menge an Features keinen Recall von 1 zulassen. Der Parameter s stellt den Schwellwert der Funktion K des Klassifikators dar. Wie sich in den Daten in Kapitel A.2 erkennen lässt, hat die Feature Auswahl aus *Transfer of File that was requested the most* und *Ratio between transfersize and filesize* die höchste Precision von 0,89 und einem Recall von 1. Diese Features entsprechen den Merkmalen *Die meistgelesene Datei* in Kapitel 3.3.4 und *Median des Verhältnis zwischen transfersize und filesize* in Kapitel 3.3.4. Alle weiteren Variation verfügen über eine schlechtere Precision oder einen schlechteren Recall.

Verwendetes Klassifikationsmodell

Mit den Ergebnissen der Feature Selection ist ein Modell ausgewählt, welches die zwei Features *Die meistgelesene Datei* und den *Median des Verhältnisses zwischen transferSize und fileSize* benötigt. Als Schwellwert für die Zuordnung in eine der Klassen ist eine Wahrscheinlichkeit von $0,14$ ausgewählt. Auf den Trainingsdaten hat das Modell folgende Werte für die Evaluationsmetriken:

- Precision: 0.89
- Recall: 1

Die Confusion Matrix für die klassifizierten Testdaten ist:

	Vorhergesagte Klasse	
Echte Klasse	Positiv	Negativ
Positiv	33	0
Negativ	4	188

Tabelle 3.2: Confusion Matrix des verwendeten Modells

Wie in der Confusion Matrix in Tabelle 3.2 beschrieben ist, hat das Modell für die Testdaten keinen Datensatz fälschlicherweise der Negativ-Klasse zugeordnet. 33 Datensätze sind fälschlicherweise als positiv eingestuft. Die Interpretation der Ergebnisse sowie die Evaluation des Klassifikators werden in Kapitel 4 dargestellt.

3.4 Software zur dCache-Überwachung

Das Modell verwendet historische Daten aus dem Billing-Archiv um trainiert und evaluiert zu werden. Um eine Klassifizierung von Zeiträumen im Betrieb zu erreichen, müssen Tranfer-Events erhalten und transformiert werden, um dem Modell übergeben zu werden und auf dem entsprechenden Resultat eine Aktion auszuwählen und durchzuführen. Für den Betrieb wird das Software-Tool `GROGU` auf Basis des MAPE-Loops entwickelt, welches Tranfers-Events sammelt und transformiert, dem Klassifikator übergibt, eine Aktion auswählt und ausführt. Wie die Schritte des KDD-Prozesses wird das Tool auch in Python umgesetzt. Die vier Schritte des MAPE-Loop werden innerhalb einer `while-true`-Schleife ausgeführt. Für jeden Schritt ist eine Interface-Klasse definiert. Die Ergebnisse einer jeden Phase werden in einem Data Transfer Object (DTO) gespeichert

(siehe Listing 3). Die Verwendung von definierten Schnittstellen führt dazu, dass sich Implementierungen der Phasen einfach austauschen lassen.

```
while True:
    monitoring_data: monitoring.get_data()
    analyze_result: analyze.analyze(monitoring_data)
    plan_result: plan.execute(analyze_result)
    executor.execute(plan_result, monitoring_data)
```

Listing 3: Umsetzung des MAPE-Loop in Python

Das `DataForClassifierAndExecutor`-DTO, das als Output der Monitoring-Phase zurückgegeben und in der Variable `monitoring_data` gespeichert wird, beinhaltet Attribute, die in der Analyze- und Execution-Phase benötigt werden.

Name	**Beschreibung**
filename	Name der Datei, die am meisten Zugriffe verzeichnet hat.
most_reads_on_one_file	Meisten Zugriffe auf eine Datei.
median_ratio_transfersize_filesize	Median des Verhältnis von `transfersize` und `filesize`
user_id	ID des Nutzers, der die meisten Zugriffe auf die Datei `filename` hat.
pool	dCache-Pool, auf dem die Datei liegt.
timestart	Start des Intervalls.
timeend	Ende des Intervalls.
accesses_by_user	Anzahl der Zugriffe des Nutzers `user_id` auf die Datei.

Tabelle 3.3: Attribute des `DataForClassifierAndExecutor`-DTO

Die Attribute `most_reads_on_one_file` und `median_ratio_transfersize_filesize` werden von der Analyze-Phase verwendet, um eine Entscheidung durch den Klassifikator zu treffen. Alle anderen Attribute werden von der Executor-Phase zur Ausführung benötigt. Die Rückgabewerte der Analyze-Phase stellen eine binäre Entscheidung dar, ob ein verfügbarkeitsgefährdender Zugriff stattfand. Mit dieser Entscheidung entscheidet die Planning-Phase, wie auf den Output der Analyze-Phase reagiert wird. Die Entscheidungsfindung der Planning-Phase ist in Kapitel 3.4 beschrieben.

Monitoring-Phase

Um aktuelle Tranfer-Events zu erhalten, wird der Kafka-Broker verwendet, welcher auch genutzt wird, um Transfer-Events im dCache-Archiv zu persistieren (siehe Kapitel 2.1.2). Dafür enthält die Monitoring-Phase einen Kafka-Consumer, welcher das Topic für das Experiment abonniert hat, welches aktuell beobachtet werden soll. In der Monitoring-Phase werden die Transfer-Events entgegengenommen und es werden folgende Eigenschaften des Billing-Eintrags überprüft:

- Protokoll: NFS4
- Nachrichtentyp: transfer
- Zugriffsart: read
- Pool-to-Pool: false

Nur wenn die obigen Eigenschaften mit dem Billing-Eintrag übereinstimmen, wird dieser gespeichert.

```
def get_data(self) -> DataForClassifierAndExecutor:
    self.new_results_are_available.wait()
    self._evaluate_past_time_slot()
    self.new_results_are_available.clear()
    return self.past_timeslot
```

Listing 4: `get_data()`-Methode der verwendeten Monitoring-Klasse

Wie in Listing 4 dargestellt ist, wird der Python Synchronisierungs-Mechanismus `Event` verwendet, um den Zeitpunkt zu steuern, an dem gesammelte Daten zurückgegeben werden können. Bei Erzeugung des Monitoring-Objektes wird ein `scheduler`-Objekt gestartet, welches in einem eigenen Thread läuft und das `Event`-Objekt alle 15 Minuten freigibt. Nach der Freigabe berechnet die Methode `_evaluate_past_time_slot()` alle relevanten Transfer-Events und speichert diese in dem Attribut `past_timeslot`, welches an den MAPE-Loop zurückgeliefert wird. Die zurückgelieferten Transfer-Events werden für den vergangenen Zeitraum aus einem internen Buffer gelöscht. Dadurch werden feste Zeiträume betrachtet, die vom Start-Zeitpunkt bis zum End-Zeitpunkt reichen. Eine Alternative dazu wäre ein Vorgehen, welches auf dem *Sliding Window* basiert. Die Funktionsweise der Alternative wird in Kapitel 5.2.4 besprochen.

Analyze-Phase

Die Analyze-Phase des MAPE-Loops dient dazu für das vergangene Intervall zu entscheiden, ob ein gefährdender Zugriff stattfand. Dafür wird das Modell verwendet, das in Kapitel 3.3.5 beschrieben ist. Die Variablen, die der Klassifikator benötigt, sind im DTO `DataForClassifierAndExecutor` als Attribute `median_ratio_transfersize_filesize` und `most_reads_on_one_file` gespeichert. Als Ergebnis wird ein Enum vom Typen `AnalyzeResult` zurückgegeben, welches die Status `NO_MULTIPLE_FILE_ACCESS_DETECTED` und `MULTIPLE_FILE_ACCESS_DETECTED` annehmen kann. Auf Grundlage dieser Analyse kann die Planning-Phase eine Entscheidung darüber treffen, wie gehandelt wird.

In der aktuellen Implementierung kann das Software-Tool nur für die Experimente CMS und ATLAS verwendet werden. Für welches Experiment das Tool verwendet wird, muss beim Start angegeben werden, da durch die Standardisierung der Merkmale das entsprechende Standardisierungs-Objekt verwendet wird. Diese Standardisierung wird in der Analyze-Phase durchgeführt.

Planning-Phase

Die Planning-Phase trifft auf Grundlage des Ergebnisses der Analyze-Phase eine Entscheidung darüber, wie reagiert werden soll. Die Planning-Phase wird mit einer State Machine modelliert. Das Ziel der verwendeten Planning-Phase ist es, Administratoren über einen verfügbarkeitsgefährdenden Dateizugriff zu informieren oder das Ereignis als Log-Nachricht auszugeben. Die konkrete Umsetzung der Benachrichtigung ist in der Execute-Phase umgesetzt. Neben der Alarmierung der Administratoren stehen noch weitere Möglichkeiten zur Verfügung, welche in Kapitel 5 näher erläutert werden. Die möglichen Ergebnisse der Analyze-Phase dienen als Eingabealphabet. Die State Machine verfügt über zwei Zustände:

- **NoMultipleFileAccessDetected:** Dies ist der Startzustand. Er repräsentiert den Zustand, dass in einem Zeitraum kein verfügbarkeitsgefährdender, paralleler Dateizugriff stattfand.

- **MultipleFileAccessDetected:** Es wurde ein verfügbarkeitsgefährdender, paralleler Dateizugriff erkannt. Es wird eine Aktion unternommen, um auf das Ereignis zu reagieren.

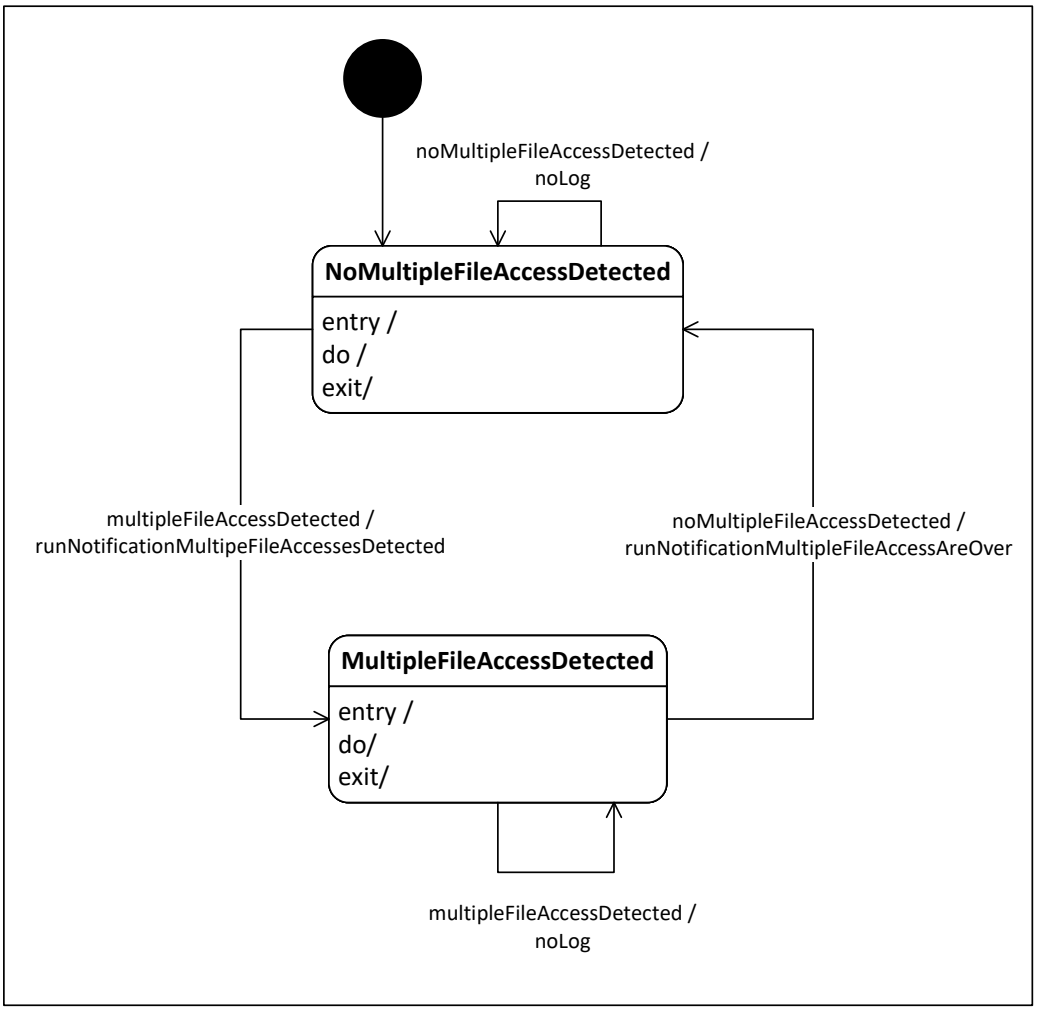

Abbildung 3.3: UML-Darstellung der State Machine, die in der Planning-Phase verwendet wird

Eine Unified Modeling Language (UML) Beschreibung der State Machine ist in Abbildung 3.3 dargestellt. Wenn mehrfache Dateizugriffe hintereinander detektiert werden, bleibt die State Machine im Zustand `MultipleFileAccessDetected`. Dadurch werden die Administratoren nicht erneut benachrichtigt. Erst wenn in einem Intervall kein verfügbarkeitsgefährdender Dateizugriff erkannt wurde, reagiert die State Machine.

Die Aktionen an den Transitionen werden als Enum `PlanningResult` an den MAPE-Loop zurückgegeben. Der MAPE-Loop übergibt diese Informationen an die Execution-Phase. Die Ausgaben der State-Machine werden im folgenden Kapitel vorgestellt.

Execution-Phase

Das Ergebnis der Planning-Phase wird von der verwendeten `Execution`-Instanz ausgeführt. Die verwendete Execution-Phase wird genutzt um Nachrichten zu generieren und diese sichtbar zu machen. Dafür existieren drei mögliche Aktionen:

- **noLog**: Es wird keine Nachricht generiert und verschickt.
- **runNotificationMultipeFileAccessesDetected:** Es muss eine Nachricht generiert werden, aus der hervorgeht, dass ein verfügbarkeitsgefährdender Dateizugriff im vergangenen Zeitraum stattfand. Diese soll die User-ID, den Dateinamen, die Anzahl der Zugriffe, den betroffenen Pool, den betroffenen Zeitraum sowie die betrachteten Features beinhalten.
- **runNotificationMultipleFileAccessAreOver**: Generiert eine Nachricht, dass kein verfügbarkeitsgefährdender Dateizugriff festgestellt wurde.

Die Aktionen werden zusammen mit den benötigten Informationen aus der Monitoring-Phase an die `Execution`-Instanz übergeben. Abhängig von der verwendeten `Execution`-Klasse werden Nachrichten unterschiedlich generiert und versandt. Für diese Arbeit gibt es zwei `Execution`-Klassen, die für verschiedene Zwecke geeignet sind.

Die Klasse `ConsoleLogExecutor` dient dem Testen und dem Debuggen des Tools. Sie generiert einfache Nachrichten auf der Konsole, wenn es benötigt wird. Für den produktiven Einsatz ist diese Klasse nicht geeignet, da ein kontinuierliches Überwachen von Nachrichten notwendig ist.

Für den produktiven Einsatz ist die Klasse `EMailNotificationExecutor` gedacht. Diese generiert E-Mail Nachrichten, die an die Administratoren versandt wird. Es wird eine E-Mail verschickt, wenn ein verfügbarkeitsgefährdender Zugriff erkannt wurde. Wenn dieser als beendet betrachtet wird, wird eine weitere verschickt.

In beiden `Executor`-Klassen werden die Informationen aus der Monitoring-Phase benötigt, da diese wichtige Informationen über den Zugriff enthält. Dazu zählen die Anzahl der Zugriffe sowie der Nutzer, der die meisten Zugriffe durchgeführt hat. Dies ist für die weitere Arbeit relevant, da der Kontakt zu dem entsprechenden Nutzer gesucht werden muss. Damit der Nutzer seine verwendeten Programme anpassen kann, benötigt dieser die Datei, auf die am häufigsten zugegriffen wurde, damit der Nutzer ein schnelles Debugging seines eigenen Programmes betreiben kann. Weiterhin sei darauf hingewiesen, dass andere `Executor`-Instanzen, welche im zukünftigen Betrieb verwendet werden, nicht alle Informationen oder andere Informationen aus der Monitoring-Phase benötigen. In diesem Fall muss das DTO der Monitoring-Phase angepasst werden.

4 Evaluation

Nach der Darstellung der Methodik im letzten Kapitel, evaluiert dieses Kapitel das Software-Tool sowie das Klassifikationsmodell. Dafür werden die Metriken des Test-Datensatzes mit gängigen Ergebnissen in der Literatur und die Evaluationsmetriken des Modells mit einem Baseline-Klassifizierer verglichen. Darüber hinaus wird das vorgestellte Software-Tool `GROGU` getestet. Dafür werden parallele Zugriffe auf eine Datei über die NAF simuliert. Ziel ist es, herauszufinden, ob das Software-Tool im Betrieb den MAPE-Loop korrekt durchführt und Zugriffe korrekt erkannt werden. Außerdem werden Live-Evaluationen des Software-Tools durchgeführt.

4.1 Evaluation des Modells

Zur Evaluation des Modells werden der Recall sowie die Precision mit Referenzwerten von logistischen Modellen aus der Literatur [24] verglichen. Bei den verwendeten Referenzwerten handelt es sich um Datensätze, welche verschiedene Eigenschaften mit Hinblick auf Balance im Dataset, Datensatzgröße und Fachdomäne aufweisen. Darüber hinaus wird das Modell mit einem Baseline-Klassifikator aus der `scikit-learn`-Bibliothek verglichen.

4.1.1 Vergleich mit Baseline-Modell

Um das verwendete Modell zu vergleichen, wird ein `DummyClassifier` verwendet. Dieser ist in Kapitel 2.2.3 beschrieben. Es wird die Klassifikationsstrategie `constant` genutzt. Diese ignoriert sämtliche Eingaben und liefert immer Klasse 1 zurück. Dadurch wird jeder Zeitraum als verfügbarkeitsgefährdend eingestuft. Als Vergleichswerte dienen der Recall und die Precision. Der `DummyClassifier` wird mit den gleichen Testwerten getestet wie das Modell. Die Werte sind in Tabelle 4.1 dargestellt. Das Klassifikations-Modell als auch der `DummyClassifier` weisen einen Recall von 1 auf. Die Precision

Klassifizierer	Precision	Recall
DummyClassifier	0,14	1
Logistisches Modell	0,89	1

Tabelle 4.1: Evaluationsmetriken des `DummyClassifier` und des entwickelten Modells

des Klassifikationsmodell ist die Gleiche wie in Kapitel 3.3.5, da der gleiche Trainingsdatensatz verwendet wird. Der `DummyClassifier` weist eine Precision von 0,14 auf und das logistische Regressionsmodell eine Precision von 0,89. Das entwickelte Modell klassifiziert deutlich besser als ein statischer Klassifikator, der alle Zeiträume als verfügbarkeitsgefährdend einstuft. Es wurde dieser einfache Klasifikator verwendet, weil er das Verhalten der Administratoren gut beschreibt, da ein kontinuierliches Überwachen der Metriken diesen Umstand gut abbildet. Wie in Kapitel 3.1 beschrieben, kann das entwickelte Modell als Baseline für weitere Modelle dienen um diese zu beurteilen.

4.1.2 Vergleich mit Referenzdaten

Da der Recall des verwendeten Modells mit eins angegeben ist, liegt ein besonderer Fokus auf der Precision und ob diese trotz des hohen Recall-Wertes vergleichbar mit dem Precision-Wert anderer Modelle ist. Wie in Abbildung 4.1 zu erkennen, ist der Recall-Wert des Modells höher als die Vergleichswerte.

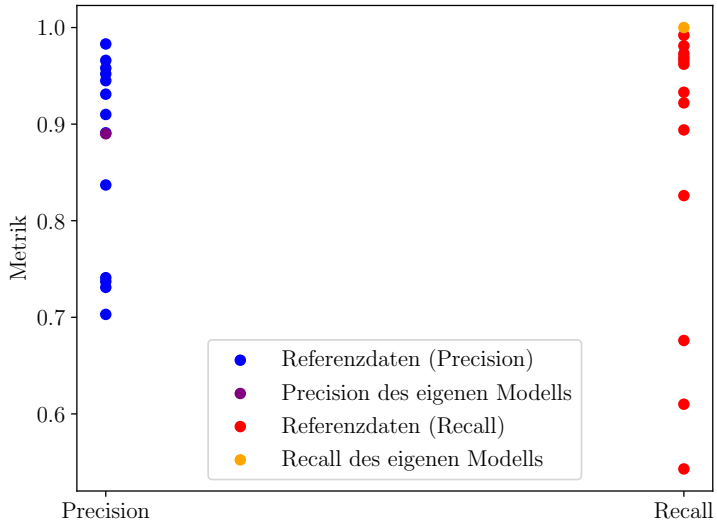

Abbildung 4.1: Vergleich des Klassifizierungs-Modell mit Referenzdaten

Ein Großteil der Vergleichswerte von Precision und Recall befindet sich nahe des Maximums. Der Precision-Wert des Modells ist mit $0,89$ im oberen Mittel im Vergleich zu den Vergleichswerten. Trotz des hohen Recall-Wertes hat das verwendete Modell noch einen Precision-Wert der sich mit Vergleichswerten messen lässt. Das verwendete Modell ist daher mit anderen logistischen Klassifizierungsmodellen vergleichbar. Wie in Kapitel 3.1 beschrieben, können andere nicht betrachtete Modelle verwendet werden, um die Klassifizierung durchzuführen. Der Einsatz von anderen Verfahren für das betrachtete Problem und eine anschließende Evaluation erfordert weitere Forschung.

4.2 Evaluation des Softwaretools

Zur Evaluation des Software-Tools wird dieses durch gezielte Zugriffe, die das Nutzerverhalten simulieren sollen sowie eine Live-Evaluation für die Experimente von CMS und ATLAS, getestet. Durch simuliertes Nutzerverhalten kann das Software-Tool unabhängig von zufälligem Nutzerverhalten getestet werden. Durch die Live-Evaluation wird das Software-Tool im Betrieb evaluiert. Durch diese Evaluation wird geprüft, ob es sich im tatsächlichem Betrieb mit natürlichem Nutzerverhalten wie erwartet verhält.

4.2.1 Simulierte Zugriffe

Zur Simulation von Zugriffen werden innerhalb des *DESY*-Namespace 30 Dateien hinterlegt, welche eine Größe von ca. einem GB bis zu mehreren GB haben. Dies sind die Zieldateien, aus denen gelesen wird. Ein Python-Programm simuliert den Zugriff von Nutzern. Die zu lesende Dateigröße ergibt sich aus der dem Verhältnis zwischen `transferSize` und `fileSize`. Aus allen positiven Datensätzen wird das arithmetische Mittel der Kenngröße gebildet, welche bei $0,11$ liegt. Das Python-Programm wählt einen zufälligen Startpunkt innerhalb der Datei und liest 11% der Dateigröße. Die Zugriffe werden über parallel laufende Jobs auf der NAF durchgeführt. Die durchschnittliche Anzahl an Dateizugriffen der Positivdatensätzen liegt bei 108 Zugriffen. Daher werden 108 parallele Jobs gestartet, welche eine der Testdateien lesen. Die Erwartung des Tests ist es, dass kurz nach dem Abschicken der Jobs diese in Teilen durchlaufen und es daraufhin zu einer Verlangsamung der Ausführungszeit kommt, da der dCache-Pool die Zugriffe nicht schnell genug beantworten kann. Daraufhin sollte das Software-Tool einen verfügbarkeitsgefährdenden Zugriff melden und im Überwachungstool der Administratoren sollten

die Meldungen von NFS-Clients, die die Server nicht mehr erreichen, zunehmen. Das Software-Tool hat beim Kafka-Broker das Topic *billing-desy* abonniert, um die entsprechenden Transfer-Events zu erhalten.

Nach dem Absenden der Jobs an die NAF werden die Jobs parallel ausgeführt und nach ca. 70 Jobs brauchen die verbleibenden Jobs mehrere Stunden zur Fertigstellung des Lesevorgangs. Das Software-Tool erkennt einen verfügbarkeitsgefährdenden Zugriff und meldet dies entsprechend. Die Anzahl an gemeldeten NFS-Clients ohne Verbindung zu ihrem Server nimmt nur minimal zu. Diese minimale Zunahme ist nicht ausreichend, um Auswirkungen auf die Verfügbarkeit zu haben. Eine Schlussfolgerung ist, dass ein mehrfacher Zugriff auf eine Datei nicht ausreicht, um das Verhalten der Nutzer zu simulieren.

Parallele Zugriffe auf verschiedene Dateien

Eine Möglichkeit das Nutzerverhalten besser zu simulieren ist es, die Zugriffe auf die meist angefragtesten Dateien zu betrachten. Für die verfügbarkeitsgefährdeten Intervalle werden die Lese-Zugriffe auf die 20. nachgefragtesten Dateien ermittelt. In Abbildung 4.2 kann man erkennen, dass wenn ein Dateizugriff auf eine oft gelesene Datei stattfindet auch häufig auf andere Dateien ähnlich häufig zugegriffen wird. Diese wichtige Erkenntnis über das Nutzerverhalten wird bei den simulierten Zugriffen verwendet, um das gewünschte Nutzerverhalten zu erreichen. Dafür wird auf 20 Dateien zugegriffen. Die Anzahl der Zugriffe ergibt sich aus dem in Abbildung 4.2 dargestellten Median. Für jede Datei werden entsprechend viele NAF-Jobs gestartet.

Leider führt dieses Vorgehen nicht zum gewünschten Resultat. Es werden für den betreffenden Zeitraum lediglich 40 Meldungen über unterbrochene NFS-Client-Server-Verbindungen gemeldet. Eine Beobachtung der Administratoren ist, dass die Anzahl der Jobs auf der NAF ein Faktor ist, welcher untersucht werden kann. Jeder Nutzer darf maximal 5000 Jobs auf der NAF starten. Um das Limit auszureizen, werden so viele Jobs wie möglich gestartet. Dieses Vorgehen führt zu 700 Meldungen, was deutlich mehr Meldungen sind, als bei den vorherigen Vorgehen, aber nicht mit der beobachteten Anzahl von mehreren zehntausend Meldungen zu vergleichen, die die Nutzer hervorrufen können.

Ein letzter Versuch, dass Nutzerverhalten herbeizuführen ist es, die Erkenntnisse aus Kapitel 4.2.2 in den simulierten Zugriffen zu verwenden. Dafür werden mindestens $2,5$

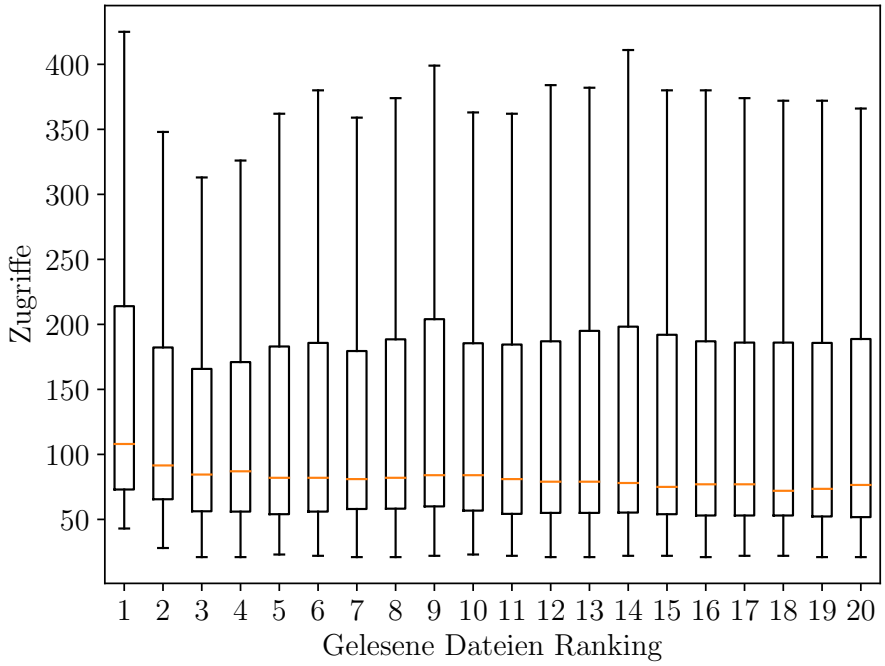

Abbildung 4.2: Boxplot der Zugriffe auf die 20. am häufigsten gelesenen Dateien der Klasse 1 Zeiträume

GB aus den Dateien gelesen. Die gelesenen Byte sind möglicherweise ein Faktor, der verfügbarkeitsgefährdende Zugriffe von nicht verfügbarkeitsgefährdenden unterscheidet. Leider führt auch dieses Vorgehen nicht zum gewünschten Ergebnis.

Es kann in dieser Arbeit nicht das genaue Zugriffsverhalten der Nutzer, welches verfügbarkeitsgefährdende Auswirkungen hat, nachgestellt werden. Um das genaue Zugriffsverhalten zu verstehen, ist weitere Forschung nötig.

4.2.2 Live-Evaluation

Abschließend wird das Software-Tool in mehreren Praxis-Tests evaluiert. Dafür werden die Transfer-Events aus den Experimenten CMS und ATLAS dem Tool zur Evaluation übergeben. Die betreffenden Zeiträume werden mit den dCache-Administratoren zusammen ausgewertet um die Ergebnisse des Tools einzuordnen. Es werden drei Durchläufe von jeweils sechs Stunden Länge durchgeführt.

1. Durchlauf

Die erste Evaluation fand am 31.08.2023 von 10:00 bis 16:00 statt. In Abbildung 4.3 ist der Verlauf der Evaluation mit unterbrochenen Verbindungen zwischen NFS-Servern zu ihren Clients sowie dem Ergebnis des Software-Tools dargestellt. Das Tool hat für den Zeitraum zwischen 10:45 bis 11:00 das erste Mal einen verfügbarkeitsgefährdenden Zugriff gemeldet. Kurz darauf steigt die Anzahl an Verbindungsabbrüchen rapide auf über 100.000 Meldungen an. In den folgenden Zeiträumen bis 11:30 werden zahlreiche Abbrüche gemeldet. Bis 11:45 erfasst das Tool verfügbarkeitsgefährdende Zugriffe. Ab 11:45 nimmt die Anzahl der Verbindungsabbrüche ab und das Tool klassifiziert die Zeiträume ab 11:45 bis 14:00 korrekterweise als nicht verfügbarkeitsgefährdend. Für den Zeitraum von 13:45 bis 14:45 meldet das Tool durchgehend verfügbarkeitsgefährdende Zugriffe und auch die Anzahl an Meldungen zu Verbindungsabbrüchen steigt auf mehrere hunderttausend Meldungen an. Die von dem Software-Tool klassifizierten Zeiträume sind auch Zeiträume, die von den Administratoren als verfügbarkeitsgefährdend eingestuft werden. Im Zeitraum von 10:45 bis 11:30, den das Software-Tool als verfügbarkeitsgefährdend

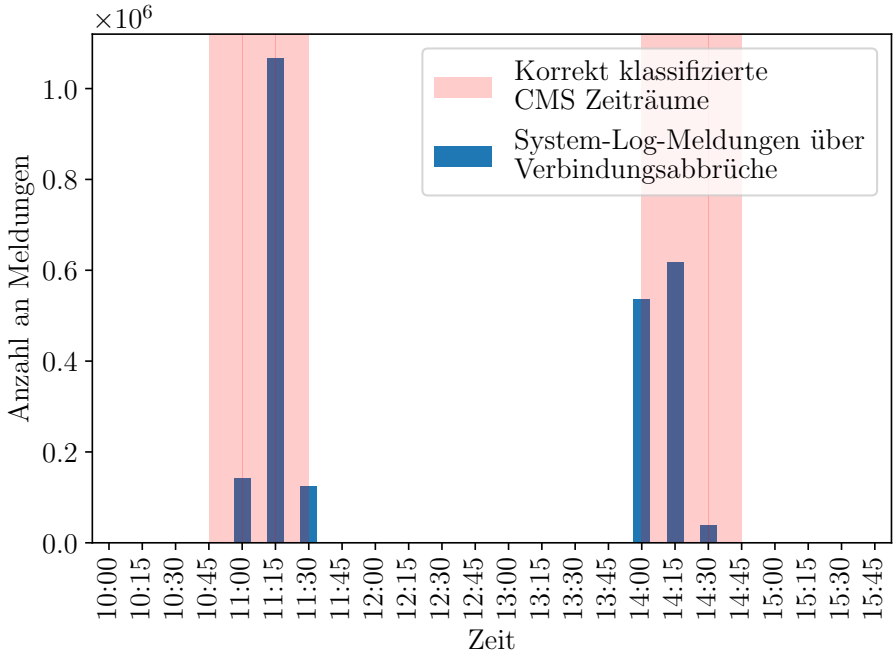

Abbildung 4.3: Betrachteter Zeitraum der ersten Live-Evaluation mit der Anzahl an gemeldeten NFS-Verbindungsproblemen sowie dem Ergebnis des Software-Tools

eingestuft hat, wurde im ersten Slot von 10:45 bis 11:15 Nutzer A als Ursache festgestellt. Für den dritten Slot wurde Nutzer B als Ursache festgestellt. Beide greifen auf die gleiche Datei F zu. Zeitgleich um 10:55 starten beide Nutzer insgesamt über 5000 Jobs auf der NAF. Das Verhalten, dass Nutzer zeitgleich zahlreiche Jobs starten und Zugriffe auf die gleiche Datei durchführen, ist bisher unbekannt und ist bei der Konzeption des Software-Tools nicht mit eingeflossen. Der Umstand, dass Nutzer zeitgleich Zugriffe durchführen unterstützt die Forderung aus Kapitel 4.2.1, dass weitere Arbeiten sich mit den Zugriffsmustern von Nutzer im dCache-System befassen müssen. Für den zweiten Zeitraum von 14:00 bis 14:45 ist Nutzer B als Ursache für die Ausfälle bestimmt.

2. Durchlauf

Die zweite Live-Evaluation dauerte sechs Stunden und ist am 09.09.2023 von 9:00 bis 15:00 durchgeführt worden. In dem Zeitraum gab es in den Tools der Administratoren keine auffällig vielen Ausfälle von NFS-Client-Server-Verbindungen. Auf der NAF konnte eine hohe Jobanzahl beobachtet werden.

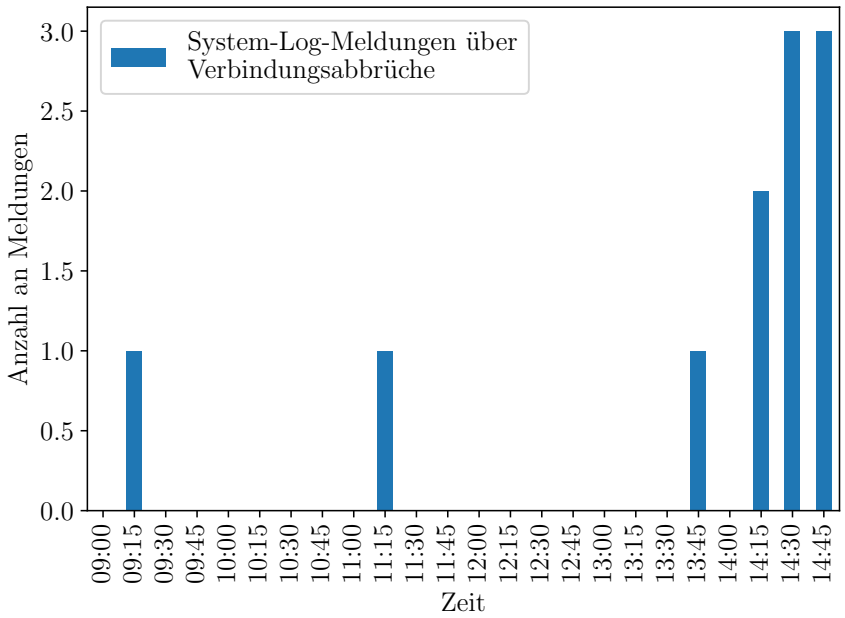

Abbildung 4.4: Betrachteter Zeitraum der zweiten Live-Evaluation mit der Anzahl an gemeldeten NFS-Verbindungsproblemen sowie dem Ergebnis des Software-Tools des 2. Durchlaufs

Wie in Abbildung 4.4 dargestellt ist, gab es trotz der hohen Jobanzahl keine verfügbarkeitsgefährdenden Dateizugriffe auf das dCache-System. Das Software-Tool hat für den gesamten Beobachtungszeitraum keinen gefährdenden Zugriff erkannt. Der erste Test stellt damit einen erfolgreichen Durchlauf dar, in welchem alle Zeiträume für beide Services korrekterweise als negativ klassifiziert wurden.

3. Durchlauf

Der dritte Durchlauf wurde am 11.09.2023 von 18:00 bis 00:00 am 12.09.2023 durchgeführt. Für das ATLAS-Experiment sind drei Zeiträume als verfügbarkeitsgefährdend eingestuft. In allen drei Zeiträumen konnten über hundert Lesezugriffe eines einzelnen Nutzers auf eine Datei beobachtet werden. Wie in Abbildung 4.5 zu erkennen ist, konnte für diesen Zeitraum keine Anzahl an unterbrochenen NFS Verbindungen zwischen dem Server und Client beobachtet werden. Für das Experiment CMS wurde für diesen

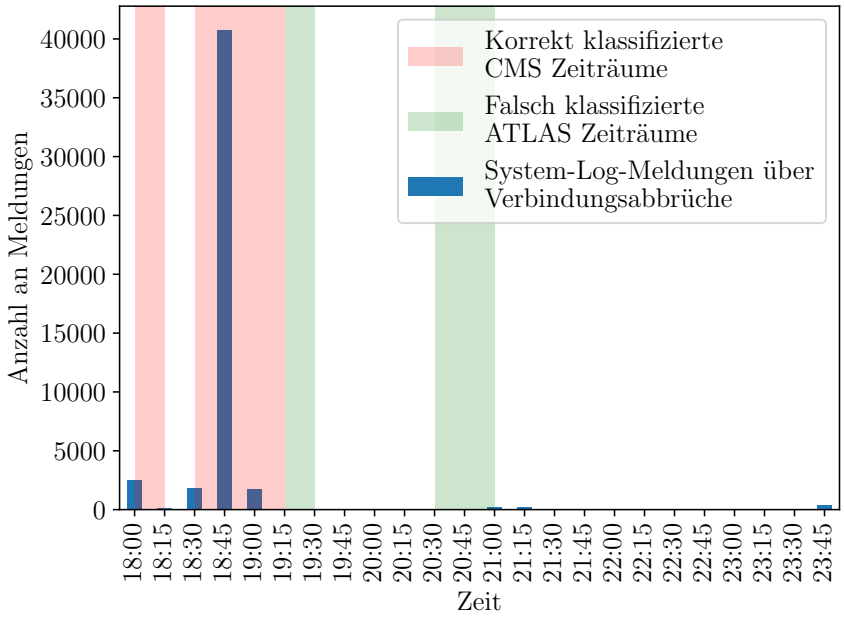

Abbildung 4.5: Betrachteter Zeitraum der ersten Live-Evaluation mit der Anzahl an gemeldeten NFS-Verbindungsproblemen sowie dem Ergebnis des Software-Tools des 3. Durchlaufs

Durchlauf für den Zeitraum 18:00-18:15 und von 18:45-19:15 eine verfügbarkeitsgefährdende Meldung vom Software-Tool gemacht. In dem detektierten Zeitraum wurde der

gleiche Nutzer A wie im ersten Durchlauf identifiziert. Darüber hinaus kann man erkennen, dass im korrekt detektierten Zeitraum die Anzahl an unterbrochenen NFS-Client-Server-Verbindungen stark zunimmt. Diese Zunahme ist von den Administratoren als verfügbarkeitsgefährdend eingestuft. Im dritten Durchlauf konnte das Software-Tool erfolgreich Intervalle korrekterweise als verfügbarkeitsgefährdend einordnen, hat aber auch drei Zeiträume als fälschlicherweise als verfügbarkeitsgefährdend eingestuft.

Vergleich der positiv klassifizierten Zeiträume

Im ersten und dritten Durchlauf konnten erfolgreich Zeiträume identifiziert werden, in denen verfügbarkeitsgefährdende Zugriffe beobachtet wurden. Im dritten Durchlauf sind Zeiträume fälschlicherweise als verfügbarkeitsgefährdend eingestuft worden. Eine Analyse der `transfersize` der Zeiträume im ersten und dritten Durchlauf ist ein Erklärungsansatz für die False Positives. In Abbildung 4.6 sind die höchsten `transfersizes` der 20 am meisten nachgefragtesten Dateien im Vergleich zur Anzahl der Transfers, die die `transfersize` beinhaltet haben für die korrekt klassifiziert und die fälschlicherweise als positiv klassifizierten Zeiträume dargestellt.

Man kann erkennen, dass für die korrekt klassifizierten Zeiträume Dateiausschnitte in Höhe von mehreren GB angefordert werden. Dafür werden diese Ausschnitte in einem Zeitraum aber nicht häufig angefordert. Die False Positive klassifizierten Zeiträume weisen eine Verteilung auf, die entgegengesetzt zu den korrekt klassifizierten Zeiträumen ist. Die größten `transfersizes` sind gering im Vergleich mit den korrekt klassifizierten Datenpunkten. Außerdem werden viele der `transfersizes` häufiger gelesen. Wenn die gleichen Dateiausschnitte häufiger angefragt werden, können diese aus dem dCache-Cache gelesen werden und müssen nicht von der Festplatte angefordert werden. Das Lesen der gleichen Dateiausschnitte vom Cache führt nicht zu dem untersuchten Problem. Aus der Länge der `transfersize` kann aber nicht auf die angeforderten Dateiausschnitte geschlossen werden. Die höchste `transfersize` in einem Intervall ist in den betrachteten Zeiträumen ein Merkmal anhand dessen man die Gruppen voneinander trennen kann.

Die höchste `transfersize` der am meisten angeforderten Datei sowie die Anzahl an `transfersize` der 20 am meisten angeforderten Datei scheinen Merkmale zu sein, in denen sich die True Positive sowie die False Positive klassifizierten Zeiträume unterscheiden. In diesem Abschnitt sind nur die Zeiträume der Evaluation betrachtet worden. Da-

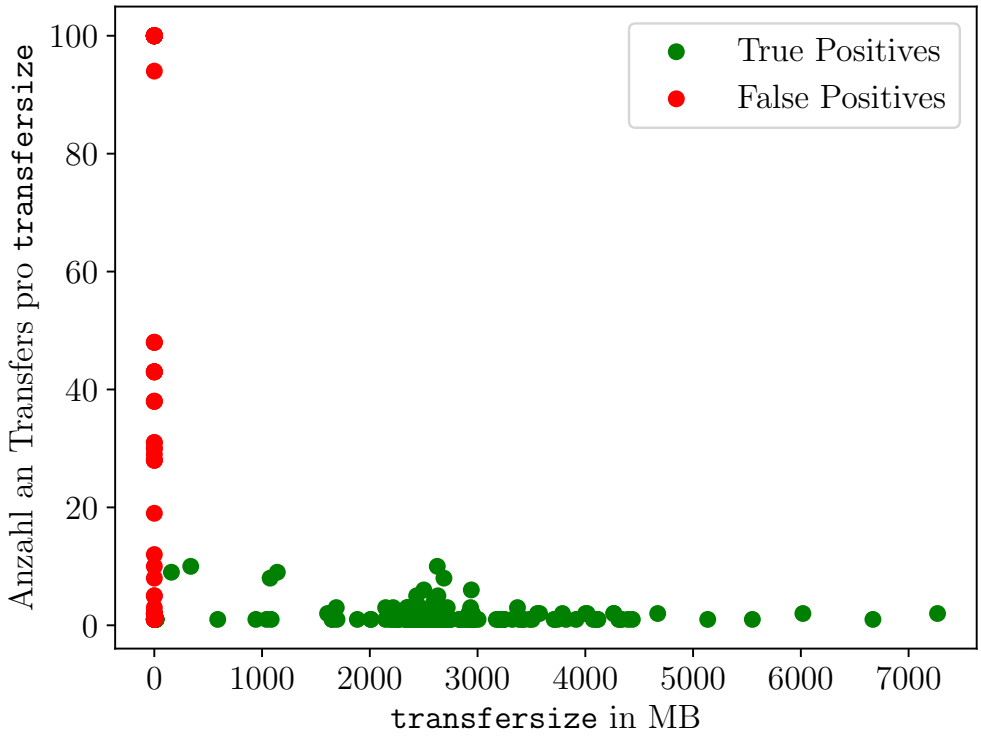

Abbildung 4.6: `transfersize` der 20. am häufigsten gelesenen Dateien sowie die Anzahl der Transfers dieser für die True Positive und False Positive Zeiträume aus dem ersten und dritten Durchlauf

her ist hier weitere Forschung nötig, ob die `transfersize` und die hier beschriebenen Features sich eignen, um die Anzahl an False Positives zu reduzieren.

5 Diskussion

Im folgenden Kapitel werden die Ergebnisse dieser Arbeit interpretiert und diskutiert. Anschließend wird ein Ausblick auf künftige Forschung gegeben. Die vorliegende Arbeit hat untersucht, ob mit dCache-Daten verfügbarkeitsgefährdendes Nutzerverhalten, das auf parallele Dateizugriffe zurückzuführen ist, erkannt werden kann. Dafür wurde mit dem KDD-Prozess ein logistisches Regressionsmodell auf Grundlage von Transfer-Events entwickelt. Weiterhin wurde untersucht, wie das Modell in ein Software-Tool integriert werden kann. Dafür wurde das Tool `GROGU` auf Basis eines MAPE-Loop entwickelt, welches im Betrieb Transfer-Events verarbeiten und bewerten sowie die dCache-Administratoren benachrichtigen kann. Abschließend wurde das Modell mit Referenzdaten und einem Baseline-Klassifikator evaluiert und das Software-Tool konnte in mehreren Live-Evaluationen überprüft werden.

5.1 Interpretation

Diese Arbeit hat die Frage untersucht, ob verfügbarkeitsgefährdenes Nutzerverhalten erkannt werden kann. Die Ergebnisse zeigen, dass dies möglich ist. Durch die Umsetzung des Klassifikators wurde ein Modell entwickelt, welches in Kapitel 3.3.5 vorgestellt sowie in Kapitel 4 evaluiert wurde. Weiterhin wurde das Ziel erreicht, dass ein Software-Tool auf Basis eines MAPE-Loop entwickelt wurde, welches genutzt werden konnte, um das Modell zu testen. Weiterhin kann dieses Tool beim DESY im produktiven Einsatz verwendet werden, wodurch es Arbeitszeit der Administratoren erspart. Darüber hinaus konnte in Kapitel 4.2.1 gezeigt werden, dass ein mehrfacher Zugriff auf eine einzelne Datei keinen verfügbarkeitsgefährdenden Zugriff darstellt, aber mehrfache Zugriffe auf einzelne Dateien selten vorkommen. Stattdessen führen Nutzer bei einem mehrfachen Zugriff auf eine Datei meist mehrfache Zugriffe auf mehrere Dateien durch. Über das genaue Zugriffsverhalten kann in dieser Arbeit keine Aussage getroffen werden. Nichtsdestotrotz konnte

am Ende der Live-Evaluation ein interessantes Merkmal identifiziert werden, welches möglicherweise gewinnbringend in den Klassifikator integriert werden kann.

5.2 Alternativen und Beschränkungen der Arbeit

Der in dieser Arbeit betrachtete Fall des Nutzerverhaltens im dCache-Speichersystem sowie die verwendeten Techniken zur Identifikation und Behebung der Ursache stellen nicht den vollen Umfang an Handlungsmöglichkeiten dar. Aufgrund von zeitlichen Einschränkungen sowie Rahmenbedingungen konnten nicht alle Möglichkeiten evaluiert werden. Diese interessanten Möglichkeiten und Alternativen werden im Folgenden erläutert und diskutiert.

5.2.1 Weiteres verfügbarkeitsgefährdendes Verhalten

Wie in Kapitel 3.2 erläutert wurde, kann es verschiedene Ursachen dafür geben, dass ein NFS-Client keine Verbindung mehr zum entsprechenden NFS-Server hat. Eine der Ursachen ist, dass Dateien, auf die häufig zusammen zugegriffen wird, auf dem gleichen dCache-Pool liegen. Der daraus resultierende Verbindungverlust ist nicht auf ein Fehlverhalten der Nutzer zurückzuführen, sondern auf eine ungünstige Verteilung von Dateien. Daher wurde dieser in dieser Arbeit nicht betrachtet.

5.2.2 Vergleich von Klassifizierungsverfahren

In dieser Arbeit wurde ausschließlich die logistische Regression als Klassifizierungsverfahren vorgestellt. Der Fokus der Arbeit war es, Indikatoren zu identifizieren, die auf ein verfügbarkeitsgefährdendes Nutzerverhalten hinweisen und ein Software-Tool zu entwickeln. Aufgrund von zeitlichen Gründen musste daher auf einen Vergleich von Klassifizierungsmethoden, die in Kapitel 3.1 vorgestellt wurden, verzichtet werden.

5.2.3 Alternative Handlungsstrategien

Die in dieser Arbeit verwendete E-Mail-Benachrichtigung für Administratoren ist eine einfache Art auf das Verhalten der Nutzer zu reagieren. Dennoch existieren Alternativen, wie auf das Verhalten reagiert werden kann. Diese können im Betrieb nicht verwendet

werden, da dies nach Aussage der Administratoren gegen interne Regelungen des DESY verstoßen würde. Neben der Benachrichtigung ist eine weitere Möglichkeit Nutzern, deren Verhalten als verfügbarkeitsgefährdend eingestuft wurde, Zugriff auf das dCache-Speichersystem zu verweigern. Dadurch hätten die Jobs auf der NAF der entsprechenden Nutzer keinen Zugriff mehr auf das dCache-System. Darüber hinaus besteht die Möglichkeit, die Anzahl an Jobs betreffender Nutzer auf der NAF zu reduzieren. Durch eine Reduzierung würden sich die Zugriffe auf das dCache-System und möglicherweise auf die betreffenden Dateien reduzieren.

Die vorgestellten Alternativen sind Handlungen die von den dCache-Administratoren manuell ausgeführt werden können, stellen aber die Handlungen mit den weitreichendsten Konsequenzen dar. Der Regelfall ist, dass die dCache-Administratoren mit den Nutzern Kontakt aufnehmen und darüber die Zugriffe reduzieren. Die vorgeschlagenen Alternativen wären aber Handlungsoptionen, die selbst-adaptive Handlungen darstellen könnten. Falls die Beschränkungen seitens des DESYs wegfallen würden, könnte das Software-Tool, um diese Fähigkeiten erweitert werden.

5.2.4 Sliding Window

In der vorgestellten Monitoring-Phase werden Intervalle mit festgesetzten Start- und End-Zeiträumen analysiert. Eine weitere Möglichkeit Intervalle zu definieren ist die *Sliding Window* Technik. Dabei würde das Intervall von 15 Minuten über die Zeit um einen Zeitraum Δt weitergeschoben werden. Die betrachteten Intervalle würden sich überlappen und wären nicht wie in der aktuellen Implementierung abgeschlossene Intervalle. Vorteil dieser Technik wäre, dass verfügbarkeitsgefährdende Zugriffe schneller erkannt werden würden, da nicht der gesamte Zeitraum verstrichen wäre, sondern nach einen Δt könnte das Software-Tool entsprechend reagieren. Nachteil wäre ein höherer Aufwand, da die gespeicherten Daten nach jedem Δt entfernt und wieder hinzugefügt werden müssten. Die *Sliding Window* Technik könnte aber zu einer schnelleren Detektion führen, weshalb sie in einer späteren Version des Software-Tool implementiert werden könnte.

5.2.5 Weitere Experimente

Die dCache-Namesräume für die Experimente ATLAS und CMS weisen das untersuchte Problem am häufigsten auf. Nichtsdestotrotz existieren weitere dCache-Namespaces beim

DESY für die, das beschriebene Problem nicht untersucht worden ist. Dies liegt an der geringen Anzahl an gemeldeten Zeiträumen. Das vorgestellte Software-Tool sollte aber bei einer entsprechenden Anzahl an gemeldeten Zeiträumen auch auf andere Experimente bzw. Namensräume übertragbar sein.

5.3 Weitere Forschung

Ein weiteres Forschungsfeld ist die Auswahl des besten Klassifizierungsverfahren. Wie in Kapitel 3.1 erwähnt, sind insbesondere Verfahren, die zur Klassifizierung von Zeitreihen geeignet sind, Verfahren, die möglicherweise für das untersuchte Problem eingesetzt werden können. Unter Verwendung des hier vorgestellten Datensatz sowie dem entwickelten Modell als Baseline könnten verschiedene Verfahren untersucht werden, um das bestmögliche Modell zu entwickeln.

Darüber hinaus ist die Analyse von Zugriffsmustern im dCache-System ein Forschungsgegenstand, welchem weitere Forschung gewidmet werden sollte. Die Analyse von Nutzerverhalten kann helfen Dateien zu identifizieren, auf welche innerhalb kurzer Zeiträume zusammen zugegriffen wird und welche sich auf den gleichen Pools befinden. Durch eine entsprechende Verteilung dieser Dateien auf unterschiedliche Nodes kann die Verfügbarkeit des dCache-Systems erhöht werden.

Neben den Zugriffsmustern ist weitere Forschung nötig, um den Einfluss der `transfersize` zu bestimmen. In Kapitel 4.2.2 wurde gezeigt, dass die `transfersize` möglicherweise ein Merkmal ist, welches die Anzahl an False Positives reduzieren kann. Insbesondere die Höhe der maximalen `transfersize` scheint ein guter Indikator zu sein. Um die Wirksamkeit dieses Merkmals zu bestimmen, sollte diese für alle Datensätze erhoben werden und mit den anderen Merkmalen systematisch verglichen werden.

6 Fazit

Im Verlauf dieser Arbeit sind die Transfer-Events und pool-info-Daten des dCache-Speichersystems untersucht worden, um Indikatoren auf ein verfügbarkeitsgefährdendes Verhalten durch parallele Zugriffe auf Dateien zu identifizieren. Dafür wurden Zeiträume, in denen ein solches Verhalten auftrat identifiziert. Im Anschluss sind mittels der Kenntnisse der dCache-Administratoren Metriken aus den Einträgen der Daten entwickelt worden. Diese sind unter Verwendung des KDD-Prozesses strukturiert aus den historischen Daten erhoben worden. Mittels der Daten ist ein Modell auf Basis der logistischen Regression erstellt und evaluiert worden. Das Modell ist in das Software-Tool `GROGU` eingebunden worden. Das Modell ist mit Referenzwerten sowie einem Baseline-Klassifikator evaluiert worden. Darüber hinaus wurde versucht das gesamte Software-Tool mit simulierten Zugriffen zu testen. Zum Abschluss konnte das Software-Tool in mehreren Live-Evaluationen im Betrieb erfolgreich getestet werden.

Die unterschiedlichen Ziele, die an diese Arbeit gestellt worden sind, konnten größtenteils erfüllt werden. Es konnten Indikatoren in den Transfer-Events identifiziert werden, welche auf ein verfügbarkeitsgefährdendes Verhalten hindeuten. Mit diesen Indikatoren konnte ein Modell trainiert werden, welches 15 minütige Intervalle klassifizieren kann. Dieses Modell weist einen Recall von 1 und eines Precision von 0,89 auf den Testdaten auf. Insbesondere die Precision weist im Vergleich zu Referenzwerten einen Wert auf, der im oberen Mittel liegt. Weiterhin war es ein Ziel, ein Software-Tool auf Basis eines MAPE-Loop zu entwickeln und zu evaluieren. Beides konnte erfolgreich umgesetzt werden.

Das Ziel das Nutzerverhalten zu simulieren, um das Sofware-Tool gezielt zu testen, konnte leider nicht erfüllt werden, da man nicht in der Lage war das schädliche Nutzerverhalten nachzubilden. Darüber hinaus konnte in der Evaluation festgestellt werden, dass die `transfersize` ein Merkmal darstellt, welches zur Reduzierung der False Positives verwendet werden könnte. Diese Hypothese wurde erst im Rahmen der Evaluation aufge-

deckt, weshalb das entsprechende Merkmal im Modell zur Klassifizierung nicht genutzt werden kann.

Ausblick

Mit der erfolgreichen Entwicklung und der Evaluation des Klassifikationsmodells sowie der Einbettung in ein Software-Tool kann das Resultat dieser Arbeit von den dCache-Administratoren betrieben werden. Zur Verbesserung des Modells muss das bestmögliche Klassifikationsverfahren ermittelt werden. Als Baseline-Klassifikator kann das Modell dieser Arbeit sowie der hier erstellte Datensatz dienen. Durch die Modularisierung des Software-Tools kann ein Verfahren, welches bessere Werte in den Evaluationsmetriken aufweist, einfach eingebaut werden.

Zur Verbesserung des Klassifikations-Modells empfehle ich, dass die größtmögliche `transfersize` einer Datei innerhalb eines Intervalls als Merkmal untersucht wird. In Kapitel 4.2.2 ist dieses Merkmal identifiziert worden, um die Anzahl an False Positives zu minimieren und es hat möglicherweise das Potential ein zukünftiges Modell robuster gegen False Positive Meldungen zu machen.

Literaturverzeichnis

[1] : *Apache Kafka*. – URL https://kafka.apache.org/documentation/. – Zugriffsdatum: 2023-07-28

[2] : *Apache Spark*. – URL https://spark.apache.org/docs/latest/. – Zugriffsdatum: 2023-10-11

[3] : *Apache ZooKeeper*. – URL https://zookeeper.apache.org/. – Zugriffsdatum: 2023-06-06

[4] : *ATLAS Projekt*. – URL https://portal.research.lu.se/en/equipments/a-toroidal-lhc-apparatus-atlas-at-the-cern-large-hadron-collider. – Zugriffsdatum: 2023-06-08

[5] : *CMS Projekt*. – URL https://home.cern/science/experiments/cms. – Zugriffsdatum: 2023-06-08

[6] : *dCache.org Book Introduction*. – URL https://www.dcache.org/manuals/Book-9.0/intro.shtml. – Zugriffsdatum: 2023-06-06

[7] : *dCache.org Protocols Supported by dCache*. – URL https://www.dcache.org/manuals/Book-8.2/intro.shtml#protocols-supported-by-dcache. – Zugriffsdatum: 2023-05-24

[8] : *dCache.org The Billing Service*. – URL https://www.dcache.org/manuals/Book-8.2/config-billing.shtml. – Zugriffsdatum: 2023-05-24

[9] : *LHC Grid Computing*. – URL https://www.desy.de/research/facilities__projects/tier_2/index_eng.html. – Zugriffsdatum: 2023-07-29

[10] : *Logstash*. – URL https://www.elastic.co/de/logstash. – Zugriffsdatum: 2023-07-28

[11] : *NAF*. – URL https://www.desy.de/forschung/anlagen__projekte/naf/index_ger.html. – Zugriffsdatum: 2023-07-29

[12] : *Scientist approach to dCache monitoring - 15th International dCache Workshop*. – URL https://indico.desy.de/event/29564/contributions/104598/attachments/66124/81847/dCache-Workshop-June2021.pdf. – Zugriffsdatum: 2023-07-28

[13] Arp, Daniel ; Quiring, Erwin ; Pendlebury, Feargus ; Warnecke, Alexander ; Pierazzi, Fabio ; Wressnegger, Christian ; Cavallaro, Lorenzo ; Rieck, Konrad: *Dos and Don'ts of Machine Learning in Computer Security*. 2021

[14] Azevedo, Ana ; Santos, Manuel: KDD, semma and CRISP-DM: A parallel overview, 01 2008, S. 182–185

[15] Backhaus, Klaus ; Erichson, Bernd ; Gensler, Sonja ; Weiber, Rolf ; Weiber, Thomas: *Logistische Regression*. S. 287–379. In: *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. Wiesbaden : Springer Fachmedien Wiesbaden, 2023. – URL https://doi.org/10.1007/978-3-658-40465-9_5. – ISBN 978-3-658-40465-9

[16] Chawla, Nitesh V.: *Data Mining for Imbalanced Datasets: An Overview*. S. 875–886. In: Maimon, Oded (Hrsg.) ; Rokach, Lior (Hrsg.): *Data Mining and Knowledge Discovery Handbook*. Boston, MA : Springer US, 2010. – URL https://doi.org/10.1007/978-0-387-09823-4_45. – ISBN 978-0-387-09823-4

[17] Fayyad, Usama ; Piatetsky-Shapiro, Gregory ; Smyth, Padhraic: The KDD process for extracting useful knowledge from volumes of data. In: *Communications of the ACM* 39 (1996), Nr. 11, S. 27–34

[18] Géron, A. ; Rother, K. ; Demmig, T.: *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. dpunkt, 2020. – URL https://books.google.de/books?id=hUZqzQEACAAJ. – ISBN 9783960091240

[19] Hirschle, Jochen: *Machine Learning für Zeitreihen: Einstieg in Regressions-, ARIMA-und Deep Learning-Verfahren mit Python. Inkl. E-Book*. Carl Hanser Verlag GmbH Co KG, 2020

[20] Hude, Marlis von der: *Regressionsanalyse – Ein klassisches Verfahren der Statistik*. S. 107–123. In: *Predictive Analytics und Data Mining : Eine Einführung mit R*.

Wiesbaden : Springer Fachmedien Wiesbaden, 2020. – URL https://doi.org/10.1007/978-3-658-30153-8_9. – ISBN 978-3-658-30153-8

[21] Jindal, Nitin ; Liu, Bing: Review Spam Detection. In: *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA : Association for Computing Machinery, 2007 (WWW '07), S. 1189–1190. – URL https://doi.org/10.1145/1242572.1242759. – ISBN 9781595936547

[22] Kalisch, Markus ; Meier, Lukas: *Das logistische Regressionsmodell*. S. 13–29. In: *Logistische Regression: Eine anwendungsorientierte Einführung mit R*. Wiesbaden : Springer Fachmedien Wiesbaden, 2021. – URL https://doi.org/10.1007/978-3-658-34225-8_3. – ISBN 978-3-658-34225-8

[23] Kohavi, Ron ; John, George H.: Wrappers for feature subset selection. In: *Artificial Intelligence* 97 (1997), Nr. 1, S. 273–324. – URL https://www.sciencedirect.com/science/article/pii/S000437029700043X. – Relevance. – ISSN 0004-3702

[24] Musa, Abdallah B.: Comparative study on classification performance between support vector machine and logistic regression. 4, Nr. 1, S. 13–24. – URL https://doi.org/10.1007/s13042-012-0068-x. – ISSN 1868-808X

[25] Neelamegam, S ; Ramaraj, E: Classification algorithm in data mining: An overview. In: *International Journal of P2P Network Trends and Technology (IJPTT)* 4 (2013), Nr. 8, S. 369–374

[26] Runkler, Thomas A.: *Data Mining: Modelle und Algorithmen intelligenter Datenanalyse*. Springer Vieweg, 2015 (Computational Intelligence). – ISBN 978-3-8348-1694-8

[27] Salehie, Mazeiar ; Tahvildari, Ladan: Self-Adaptive Software: Landscape and Research Challenges. In: *ACM Trans. Auton. Adapt. Syst.* 4 (2009), may, Nr. 2. – URL https://doi.org/10.1145/1516533.1516538. – ISSN 1556-4665

A Anhang

A.1 Beispiel für ein Transfer-Event im JSON-Format


```
{
    "@timestamp": "1970-01-01T00:00:00.000Z",
    "@version": "1",
    "billingPath": "/path/to/file",
    "cellDomain": "dcache-desy01-01Domain",
    "cellName": "dcache-desy01-01",
    "cellType": "pool",
    "date": "1970-01-01T00:00:00.000+01:00",
    "fileSize": 42,
    "initiator": "door:nfs4-afs@dcache-door-desy01_nfs4wnDomain
       :xx+xx:1",
    "isP2p": false,
    "isWrite": "read",
    "meanReadBandwidth": 4.2,
    "meanWriteBandwidth": 0.0,
    "msgType": "transfer",
    "pnfsid": "42",
    "protocolInfo": {
        "host": "192.168.0.1",
        "port": 739,
        "protocol": "NFS4",
        "versionMajor": 4,
        "versionMinor": 1
    },
    "queuingTime": 0,
    "readActive": "PT0.042",
```

```
    "readIdle": "PT0.42",
    "session": "pool:dcache-desy01-01@dcache-desy01-01Domain:42
       -42",
    "status": {
        "code": 0,
        "msg": ""
    },
    "storageInfo": "desy:gebhardt@dcache",
    "subject": [
        "UidPrincipal[4242]",
        "GidPrincipal[1997,primary]",
        "GidPrincipal[1997]",
        "Origin[192.168.0.1]"
    ],
    "tags": [
        "desy"
    ],
    "transferPath": "/",
    "transferSize": 4,
    "transferTime": 1,
    "version": "1.0",
    "writeActive": "PT0.42",
    "writeIdle": "PT0.24"
}
```

A.2 Evaluationsmetriken aller möglichen Feature-Variationen

Verwendete Features	precision	recall	s
'Transfer of Files that was requested the most'; 'Ratio between transfersize and filesize'	0,891892	1,000000	0,145178
'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Ratio between transfersize and filesize'	0,868421	1,000000	0,137932
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Ratio between transfersize and filesize'	0,846154	1,000000	0,132205
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Ratio between transfersize and filesize'	0,846154	1,000000	0,130230
'Transfer of Files that was requested the most'; 'Sum of transfersize'	0,825000	1,000000	0,128343
'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Sum of transfersize'	0,825000	1,000000	0,126848
'Transfer of Files that was requested the most'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,825000	1,000000	0,129472
'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,825000	1,000000	0,126962
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Sum of transfersize'	0,785714	1,000000	0,126255
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Sum of transfersize'	0,785714	1,000000	0,125639
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,785714	1,000000	0,125506
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,785714	1,000000	0,124504

Verwendete Features	precision	recall	s
'Transfers of the five most requested files'; 'Ratio between transfersize and filesize'	0,727273	0,969697	0,075529
'Number of transfers'; 'Transfers of the five most requested files'; 'Ratio between transfersize and filesize'	0,680851	0,969697	0,072944
'Number of transfers'; 'Transfers of the five most requested files'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,666667	0,969697	0,068933
'Number of transfers'; 'Transfers of the five most requested files'; 'Sum of transfersize'	0,653061	0,969697	0,069247
'Transfers of the five most requested files'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,653061	0,969697	0,068498
'Transfers of the five most requested files'; 'Sum of transfersize'	0,640000	0,969697	0,068656
'Number of transfers'; 'Sum of transfersize'	0,196203	0,939394	0,079004
'Number of transfers'; 'Transfer of Files that was requested the most'	0,194118	1,000000	0,013206
'Transfers of the five most requested files'	0,192982	1,000000	0,023583
'Number of transfers'; 'Transfers of the five most requested files'	0,192982	1,000000	0,023375
'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'	0,192982	1,000000	0,013338
'Number of transfers'; 'Transfer of Files that was requested the most'; 'Transfers of the five most requested files'	0,192982	1,000000	0,013135
'Transfer of Files that was requested the most'	0,190751	1,000000	0,013588
'Sum of transfersize'	0,189655	1,000000	0,080027
'Number of transfers'	0,181287	0,939394	0,088539
'Number of transfers'; 'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,150000	1,000000	0,032429
'Sum of transfersize'; 'Ratio between transfersize and filesize'	0,148649	1,000000	0,020421
'Ratio between transfersize and filesize'	0,146667	1,000000	0,012992
'Number of transfers'; 'Ratio between transfersize and filesize'	0,146667	1,000000	0,029499

Glossar

ATLAS steht für A Toroidal LHC Apparatus und ist das weltweit größte teilchenphysikalische Experiment der Welt[4] .

CMS steht für Compact Muon Solenoid und ist ein Experiment am CERN[5].

Dataset Ein Dataset ist eine Menge an Daten, die für einen bestimmten Zusammenhang erhoben wurden.

DESY Das Deutsche Elektronen Synchrotron (DESY) ist ein Forschungszentrum der Helmholtz-Gemeinschaft.

Jupyter Web-basierte Anwendung zum Erstellen von interaktiven Programmen und wissenschaftlichen Berechnungen.

LHCb Das Large Hadron Collider beauty Experiment ist ein Experiment am LHC. .

Logstash ist eine Datenverarbeitungspipeline zur Integration , Transformation und dem Versand von Daten aus verschiedenen Quellen zu unterschiedlichen Empfängern[10].

Python Python ist eine universelle, interpretierte Programmiersprache, welche über zahlreiche Bibliotheken zur Datenanalyse verfügt.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original