Nutzung bahnbezogener Sensordaten zur Vorhersage von Wartungszyklen

Machine Learning zur Vorhersage von Schäden an Lokomotiven

Diplomarbeit von Sebastian Kauschke

Mai 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Nutzung bahnbezogener Sensordaten zur Vorhersage von Wartungszyklen

Vorgelegte Diplomarbeit von Sebastian Kauschke aus Darmstadt

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Frederik Janssen, Dr. Immanuel Schweizer

Tag der Einreichung:

Erklärung zur Diplomarbeit

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 06.05.2014

_____

(Sebastian Kauschke)

Kurzfassung

Auf den Zügen der DB Schenker Rail AG werden kontinuierliche Diagnosedaten-Aufzeichnungen vorgenommen. Firmenintern wird nach Möglichkeiten gesucht, diese Daten zur Verbesserung der Flottenverfügbarkeit und zur Einsparung von Kosten zu nutzen. Ein mögliches Einsatzgebiet ist die Vorhersage von Schäden an den Lokomotiven.

In der vorliegenden Diplomarbeit werden Möglichkeiten und Methoden zur *Condition based Maintenance* bei Lokomotiven der DB Schenker Rail AG entwickelt und mittels der *WEKA* Suite evaluiert.

Die technischen Gegebenheiten werden erläutert und zwei Vorgehensweisen zur Vorhersage von Schäden werden vorgestellt. Eine basiert auf Diagnosedaten, welche durch die Lokomotive generiert werden, die zweite auf analogen Sensormesswerten der Fahrmotoren. Technische Engpässe werden aufgezeigt und Lösungsvorschläge erarbeitet.

Die Resultate zeigen, dass eine Vorhersage bestimmter Schäden im Bereich von Tagen bis zu mehreren Wochen möglich ist, sofern eine adäquate Datenqualität und -quantität zugrunde liegt, was derzeit noch nicht der Fall ist.

Abstract

Trains of DB Schenker Rail AG create a continuous logfile of diagnostics data. Within the company methods to use this data in order to increase train availability and reduce costs are researched. A possible field of application would be the prediction of train failure.

In this Diploma thesis possibilities and methods regarding a *Condition Based Maintenance* system for trains of the DB Schenker Rail AG are developed and evaluated via the *WEKA* Suite.

The technical background is explained and two workflows for a prediction of imminent hardware failure are shown. One of those methods exploits the train's internal logging of diagnostics data, the other is based on analog sensor measurements within the train's engines. Technical difficulties are explained and solutions are derived.

The results show that a prediction of certain system failures is possible starting from days up to a few weeks ahead of the failure. This requires a level of data-quality and quantity that is currently not available.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listing-Verzeichnis

1. Einleitung

Der Begriff *Condition based maintenance* (Zustandsbasierte Wartung) bezeichnet Wartungsvorgänge, die erst durchgeführt werden, wenn die Notwendigkeit dazu besteht. Die Wartung findet dann statt, wenn indiziert ist, dass Bauteile bald ausfallen werden oder ihre Leistung sich verschlechtert.
Um eine solche Indikation leisten zu können, muss die Voraussetzung, dass Diagnosesysteme mit Zustands- und Fehlerprotokollierung vorhanden sind, erfüllt sein. Im Idealfall liefern diese Systeme Echtzeitdaten und können sie auch in Echtzeit auswerten.

Die Anwendungsgebiete sind vielfältig und umfassen alle Industriezweige, auch den des Schienenverkehrs.

Die DB Schenker Rail AG ist ein Tochterunternehmen der Deutschen Bahn, welches das Geschäftsfeld Schienengüterverkehr abdeckt.
Mit einer Flottengröße von über 3500 Lokomotiven und jährlich fast 400 Millionen Tonnen beförderten Gütern ist es unabdingbar, eine hohe Verfügbarkeit der Lokomotiven bereitstellen zu können.
Innerhalb der DB Schenker Rail AG wird seit 2012 das Projekt "TechLok" durchgeführt. In dem Projekt geht es darum, die auf neueren Lokomotiven (welche ab Mitte der 1990er Jahre gebaut wurden) vorhandenen Diagnosesysteme zu nutzen.
Ziel des Projektes ist, mit diesen Systemen den technische Zustand der Lokomotive – Fehlermeldungen, Zustandsvariablen und analoge Messwerte – zu ermitteln, um daraus bevorstehende Schäden an Komponenten der Lokomotive abzuleiten. Auf Basis dieser Daten sollen Maßnahmen für die Instandhaltung, den Betrieb und den Einsatz neuer Komponenten ergriffen werden. Langfristig ist die Einführung der *Condition Based Maintenance* geplant, die durch die Auswertung der Daten erst ermöglicht wird. Zum Zeitpunkt der Verfassung dieser Arbeit werden die vorhandenen Daten lediglich in der Werkstatt ausgelesen und nur selten zu Analysezwecken genutzt.

Durch *Condition Based Maintenance* können individuelle Wartungen vorgenommen werden und regelmäßige Wartungsintervalle entfallen. Das spart Zeit und Kosten und erhöht die Flottenverfügbarkeit.

1.1. Anforderungen und Ziele

Im Rahmen dieser Diplomarbeit werden Konzepte und Methoden erarbeitet, um die vorhandenen Daten für eine Schadensvorhersage zu nutzen. Es wird gezeigt, welche weiteren Daten noch benötigt werden, um dem Ziel der *Condition Based Maintenance* näher zu kommen.

Das Ziel dieser Diplomarbeit ist eine prototypische Machbarkeitsstudie zum Thema *Condition Based Maintenance* für Lokomotiven bei DB Schenker Rail.

Durch *Machine-Learning*-Methoden soll die Erkennung und Vorhersage von Ausfällen anhand historischer Diagnosedaten ermöglicht werden. Dazu werden die betrachteten Problemstellungen als Klassifizierungsprobleme modelliert und mittels der *WEKA*-Suite evaluiert.

Zu den Aufgaben für diese Machbarkeitsstudie zählen:

- Analyse des aktuellen Standes der Technik
- Gewinnung und Bereitstellung der benötigten Rohdaten
- Spezifikation und prototypische Implementierung geeigneter Vorhersage-Verfahren
- Anwendung verschiedener *Machine-Learning* und *Data-Mining* Methoden und Evaluation der Ergebnisse
- Vorschlag zur Integration der Methoden in die bestehende Software-Infrastruktur

2. Stand der Technik

In diesem Kapitel wird eine Bestandaufnahme der Situation bei DB Schenker Rail, eine grundlegende Einführung in das *Machine Learning* und ein Überblick über verwandte Forschungsarbeiten gegeben.

2.1. *Condition based maintenance* (Zustandsbasierte Wartung)

Im Konzern der Deutschen Bahn werden unterschiedliche Lokomotiven-Baureihen von verschiedenen Herstellern eingesetzt. Die Lokomotiven werden von der Bahn erworben, betrieben und gewartet. Im Durchschnitt ist eine Lokomotive 30 Jahre im Einsatz. Die Techniker der Bahn verfügen somit über technische Kenntnisse und Erfahrungswerte im Langzeitbetrieb, die über jene der Herstellerfirmen hinausgehen.

Abbildung 1 - Baureihe 185 - Foto von Thomas Wolf, www.foto-tw.de
Download-Datum: 28.04.2014

Seit Mitte der 1990er Jahre werden Lokomotiven mit On-Board Diagnosesystemen ausgestattet. Diese Systeme zeichnen während der Fahrt Ereignisse, Zustände, Zustandsveränderungen, analoge Messwerte und Fehlermeldungen der Komponenten auf. Ereignisse umfassen sowohl Betriebsabläufe, die von außen gesteuert werden, zum Beispiel den Wechsel der Fahrtrichtung, als auch Vorgänge, die durch interne Komponenten der Lokomotive ausgelöst werden. Als analoge Messwerten werden diverse Temperaturen, Stromspannungen und Geschwindigkeiten gemessen. Es werden insgesamt über 1000 verschiedene Messwerte mit unterschiedlicher Abtastfrequenz aufgezeichnet.

Ereignisse und Meldungen werden dem Lokführer auf der Multi-Funktions-Anzeige (MFA) im Führerstand der Lokomotive angezeigt. Er kann bei technischen Problemen Telefonkontakt zu einer Hotline aufnehmen, und dort weitere Detail-Informationen über einen Vorfall schildern. Im besten Fall kann er mit Hilfe des Hotline-Personals das Problem lösen, und die Lokomotive kann weiter betrieben werden. Im schlimmsten Fall ist dies nicht möglich. Dann kann es passieren, dass die Lokomotive aus dem Betrieb genommen und zur Reparatur der Werkstatt zugeführt werden muss.

Die anfallenden Diagnosedaten werden auf der Lokomotive gespeichert. Aufgrund der veralteten Technik bestehen Limitierungen in der Gesamtmenge der Daten, die vorgehalten werden können. Es ist keine Rechenkapazität verfügbar, um diese Daten direkt weiter zu verarbeiten. Die Weiterverarbeitung kann also nicht auf den vorhandenen Geräten der Lokomotive durchgeführt werden.

Eine Diagnosemeldung besteht aus einigen Basiswerten (Zeitstempel, Diagnosecode, etc.) sowie einigen spezifischen analogen Messwerten und Zustandsvariablen. Diese variieren je nach Art der Meldung und enthalten zum Beispiel Motortemperaturen und Getriebetemperaturen bei Meldungen, welche die Fahrmotoren betreffen.

Die gespeicherten Daten werden bei der nächsten regelmäßigen Kontrolle in der Werkstatt ausgelesen und zur Weiterverarbeitung abgelegt. Sie werden dann gegebenenfalls zu Diagnosezwecken betrachtet und zur genaueren Beschreibung von Werkstattaufträgen verwendet. Dies ist eine komplett manuelle Tätigkeit, die aufwändiges Durchsuchen der Daten durch einen Techniker voraussetzt.
Eine alternative Auslesevariante besteht. Mittels einer Direktverbindung per Modem ist das Auslesen der Daten von der Lokomotive möglich. Diese Methode setzt detaillierte technische Kenntnisse und die entsprechende Befugnis voraus und kann nur von wenigen Personen aus dem technischen Management angewandt werden.

Die aufgezeichneten Diagnosedaten variieren zwischen den einzelnen Baureihen der Lokomotiven. Zwischen verschiedenen Softwareversionen einer Baureihe sind Unterschiede zu erkennen. Da jeder Hersteller eine eigene proprietäre Software verwendet, sind hier nochmals größere Varianzen vorhanden.
Es gibt keine Standards oder Konventionen bezüglich der Schnittstellen und des Formats, in dem die Daten vorliegen.

Ein weiteres Problem stellt die zeitliche Diskontinuität der Daten dar. In den aktuell eingesetzten Aufzeichnungsarten werden ausschließlich bei einem Systemereignis Datensätze mit den dazugehörigen Sensormesswerten gespeichert. Um einen zeitlichen Verlauf auswerten zu können, wäre eine kontinuierliche Aufzeichnung einiger vordefinierter Messwerte in einem festen zeitlichen Abstand nötig. Es existieren bereits Hersteller- und Drittherstellerlösungen, die zum Verfassungszeitpunkt dieser Arbeit prototypisch auf einigen Lokomotiven eingesetzt werden und im Stande sind diese Aufgabe zu erfüllen.

2.2. Betrachtung der Diagnosedaten im Rohformat

Um ein besseres Verständnis des Formats der Diagnosedaten zu erhalten, folgt eine Betrachtung der Rohdaten aus der Baureihe 185 von Bombardier.

Die Daten liegen in kommaseparierten Dateien vor und werden in dieser Form aus der Lokomotive exportiert. Diese Dateien enthalten zusätzlich zu den Diagnosedaten einen Dateikopf, der Basisinformationen der Lokomotive zum Auslesezeitpunkt enthält. In jeder Datei sind nur Daten einer Lokomotive enthalten. So kommen hunderte Dateien für die gesamte Flotte zusammen.

Listing 1 - Auszug einer Diagnosedaten-Datei

```
"Auslesung vom: 02.10.12 02:24:52 1349137492"
"Loeschung vom: 18.09.12 07:47:57 1347947277"
```

```
"Initialisierung vom: 30.01.09 08:13:15 1233299595"
"Kilometerstand: 1209015"
"Fuellstand: 2914 von 10000"
"Aenderungsnummer: DB-Version=0238"
"DCPU-Version: dcpu-5.2.0.2"
"Display-Version: (nicht verfuegbar)"
"FZG-Version: 17F"
"Eigene Baunummer: 185090"
"Dateiversion: 0001"

"baunr, tkmw, stat, fcode, lnummer, kdatum, gdatum, kumfdat, gumfdat, kglon, kglat, kdirti, gglon, gglat,
    gdirti"

185090, T, 0, 16378, 1, 1347947279, 1347952718, 165D120085000000005000000337F0BA3800042,
    00000000030000000000000002E050A00800002, 821E700C, CC5BA42F, 0000, 021E742F, 4C5BA812, 1410

185090, T, 0, 3984, 1, 1347947279, 1347947331, 165D120050008500100000CA0000000051230553,
    165D120050008400100000CA0000000051231553, 821E700C, CC5BA42F, 0000, 821E701E, CC5BA429, 0000

185090, T, 0, 3985, 1, 1347947279, 1347947331, 165D120050008500100000CA0000000051230553,
    165D120050008400100000CA0000000051231553, 821E700C, CC5BA42F, 0000, 821E701E, CC5BA429, 0000
```

In Listing 1 ist zu sehen, dass auf den Dateikopf eine Spalten-Beschreibung der Datensätze („baunr, tkmw, stat,...“) folgt. Jede weitere Zeile definiert das Auftreten eines Diagnosecodes mit der Code-Nummer ("fcode"). Dieser besitzt Informationen über die Lokomotive, der er zugehört ("Baunr": Baunummer), Zeitpunkt des Auftretens ("kdatum": Kommt-Datum) und Endzeitpunkt ("gdatum": Geht-Datum). Diese Zeitpunkte grenzen den Zeitraum ein, in dem der Diagnosecode aktiv war. In den meisten Fällen treten nur punktuelle Meldungen auf. Es gibt jedoch auch Meldungen, die über einen Zeitraum aktiv bleiben und einen abweichenden Endzeitpunkt haben. Solch eine Meldung könnte beispielsweise eine Warnmeldung über durchdrehende Räder aufgrund von Glatteis sein.

Für Komm- und Gehzeitpunkt sind jeweils die GPS-Koordinaten hinterlegt („kglon“, „kglat“, „kgdirti“, „gglon“, „gglat“, „ggdirti“). Diese Attribute sind nicht immer befüllt. Sie liegen in einer speziellen Codierung vor und müssen erst in ein Fließkomma-Format konvertiert werden.

Die sogenannten Umfelddaten beinhalten Zusatzinformationen, die sich je nach Diagnosecode unterscheiden. Sie liegen jeweils für den Kommt- und Geht-Zeitpunkt vor und sind in einem 40-Zeichen String kodiert. Die Decodierung ist in Abhängigkeit vom Diagnosecode und eingesetzter Software-Version auf der Lokomotive unterschiedlich. Die Umfelddaten können Text-, Zahlen- und Boolesche Werte enthalten.
Zu jeder Software-Version einer Lokomotive liegt ein Decodierungs-Schlüssel vor, der für alle Diagnosecodes Informationen bereitstellt, welche Messwerte in den Umfelddaten enthalten sind und wie diese decodiert werden. Die Decodierung dieser Werte erfolgt über Bitverschiebungen und Rechenoperationen auf den Zeichen des Umfelddaten-Strings.

Auf die Decodierung wird hier nicht näher eingegangen, sie erläutert sich am besten über den Quellcode der Import-Scripts in den Anhängen (siehe Kapitel 7).

2.3. *Machine Learning*

In diesem Kapitel wird eine Einführung in die Methoden des *Machine Learning* und deren Anwendungsgebiete erfolgen. Es bietet dem fachfremden Leser einen Überblick und ermöglicht

ihm, den weiteren Kapiteln folgen zu können, ohne die genauen technischen Details kennen zu müssen.

Machine Learning und *Data Mining* sind zwei Begriffe, die oftmals fälschlicherweise als Synonyme verwendet werden.
Machine Learning bezeichnet die Generierung von Wissen aus Erfahrung, welches aus Beispielen gelernt und verallgemeinert wurde. Data Mining bezeichnet die Anwendung mathematischer Methoden auf einen Datenbestand, mit dem Ziel der Mustererkennung.

Die grundsätzlichen Methoden für *Data Mining* und *Machine Learning* existieren seit Jahrzehnten. Durch die Entwicklung in den letzten Jahren hin zum "*ubiquitous computing*", dem überall vernetzt und online-Seins durch Smartphones und Tablets, steigen die aufgezeichneten Datenmengen stetig. Die Aufzeichnung der Daten ist aber nur ein Bruchteil des Aufwandes. Es stellt sich die Frage „Wie kann aus all diesen Datensätzen echtes, nützliches Wissen generiert werden?“.

Ein mögliches Ergebnis dieser Frage kann der Vorschlag "Personen, die X gekauft haben, kauften auch Y" sein, der in Online-Shops häufig gezeigt wird. Eine solche Empfehlung kann für den Shop-Inhaber einen weiteren Verkauf bedeuten, er hat sein Geschäft damit direkt verbessert. Natürlich nur dann, wenn die Empfehlung auch Sinn macht.

Machine Learning und *Data Mining* werden benutzt, wenn große Datenmengen vorliegen, die vom Menschen nicht mehr überblickt und verarbeitet werden können. Hierbei kann es sich um langjährige Aufzeichnungen von Wetterdaten oder Verkäufen aus einem Online-Shop handeln. Vorhandene Daten werden analysiert und Informationen und Wissen aus diesen Daten gewonnen. Dabei handelt es sich um Informationen, die vorher aufgrund der schieren Menge nicht direkt sichtbar waren. *Machine Learning* stellt einen großen Werkzeugkasten zur Datenaufbereitung und -verarbeitung bereit und verwendet diesen, um nützliche Information zu generieren. *Data Mining* hingegen zielt auf die Entdeckung vorhandenen Wissens ab (Witten, Frank, & Hall, 2011).

Eine der wichtigsten Fähigkeiten von *Machine Learning* Verfahren besteht darin, vorhandene Datenaufzeichnungen zu studieren und diese "Erfahrung" zu nutzen, um neue Datensätze zu verarbeiten. Ein Anwendungsfall für diese Art von Vorgehen ist die Klassifizierung. Anhand einer Menge von Attributen eines Datensatzes (einer Instanz) wird entschieden, welcher Klasse er angehört. Attribute können in verschiedenen Formen vorkommen: Sie können nominal, numerisch oder boolesch sein. Numerische Attribute sind Ganz- oder Fließkommawerte. Nominale Attribute sind Werte mit diskreten Bezeichnungen. Boolesche-Werte beinhalten wahr/falsch Aussagen.

Ein Beispiel aus der Lehre ist der in der *WEKA*-Suite enthaltene "*Weather*" Datensatz (Tabelle 1). In diesem Datensatz sind verschiedene nominale Attribute der Wetterlage (*„Outlook“, „Temperature“, „Humidity“, „Windy“*) und die Entscheidung, ob es sinnvoll ist bei diesem Wetter nach draußen zum Fußballspielen zu gehen (die sogenannte Klasse) enthalten. Das Attribut „Play“ zeigt die Klasse an, und ist für jede Instanz dieses Datensatzes befüllt.

Tabelle 1 - "*Weather*"-Datensatz

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

2.3.1. Klassifizierer

Ein Klassifizierer erhält eine bestimmte Menge von Instanzen, wie in Tabelle 1 gezeigt, und erzeugt ein Programm, das für neue Datensätze, die ohne das *Play*-Attribut geliefert werden, berechnen kann, ob *Play=yes* oder *Play=no* gesetzt wird. Dies nennt sich *Supervised-Learning* Ansatz. Beim *Supervised-Learning* ist Voraussetzung, dass eine gewisse Menge von Instanzen vorhanden ist, anhand derer der Klassifizierer die korrekte Einordnung lernen kann. Dabei handelt es sich um die sogenannte Trainingsphase, in der ein Modell gelernt wird. Die Instanzen bestehen in diesem Fall aus realen Aufzeichnungen und sind manuell klassifiziert worden. Sie stellen die Trainingsdaten dar. Attribute, die zum Training genutzt werden oder zur Klassifizierung dienen, bezeichnet man auch als Features.

Im Gegensatz zum *Supervised-Learning* stehen *Unsupervised*-Verfahren. Ein Beispiel hierfür ist das sogenannte *Clustering*. Beim *Clustering* wird versucht, in unklassifizierten Mengen von Instanzen zusammenhängende Gruppen zu finden. Instanzen, die sich ähnlich sind, werden zu einem Cluster zusammengefügt.

Um eine Klassifizierung zu ermöglichen, gibt es verschiedene technische Ansätze. Im Folgenden wird kurz auf die wichtigsten Arten eingegangen.

Regelbasierte Klassifizierer

Bei regelbasierten Klassifizierern wird in der Lernphase ein Regelwerk aufgebaut, anhand dessen dann die Klassifizierung vorgenommen wird. Das Regelwerk besteht aus einzelnen Regeln, die jeweils in der Form "Wenn X, dann Y" anhand eines oder mehrerer Attribute eine Entscheidung treffen. Ein Regelwerk kann aus beliebig vielen solcher Regeln bestehen, die nacheinander abgearbeitet werden, bis eine Bedingung komplett erfüllt und die Entscheidung gefällt wird.

Solch ein Regelwerk für den *Weather*-Datensatz ist in Listing 2 dargestellt.

Listing 2 Beispiel: Regelwerk für Klassifizierung (Erstellt mit WEKA, JRip Klassifizierer)

```
IF (humidity = high) and (outlook = sunny) THEN play=no
IF (outlook = rainy) and (windy = TRUE) THEN play=no
play=yes
```

Hier zeigt sich, dass die letzte Anweisung keine *IF*-Bedingung mehr enthält. Sie wird dann ausgeführt, wenn keine der vorherigen Regeln greift. Dies wird auch als „*Default-Regel*“ bezeichnet.

Viele regelbasierte Klassifizierer basieren auf dem sogenannten *Separate and Conquer* Prinzip. Hierbei wird zuerst versucht, für jede Klasse ein Regelwerk zu finden, das alle Instanzen abdeckt und sie von den anderen Klassen separiert (*Separate*). Im nächsten Schritt (*Conquer*) werden rekursiv weitere Regeln erzeugt, die die übrigen Trainings-Instanzen abdecken.

Oftmals werden Regelwerke erzeugt, die zwar die Trainingsdaten gut abdecken, für Verwendung auf anderen Daten aber zu spezifisch sind. Sie werden als *overfitted* bezeichnet. Um *Overfitting* zu verhindern werden, *Pruning*-Verfahren eingesetzt (*Pruning* (engl.): Der Beschnitt von Geäst). Diese Verfahren passen die Regelwerke an und versuchen sie allgemeiner zu halten und dabei gleichzeitig die Erkennungsgenauigkeit nicht zu verschlechtern. *Pruning* kann bereits während des Lernvorgangs eingesetzt werden (*Pre-Pruning*), aber auch nach dem Lernvorgang (*Post-Pruning*).

Als Gegenteil zum *Overfitting* gibt es das *Underfitting*, bei dem ein Regelwerk nicht präzise genug ist. Als Resultat von *Underfitting* werden viele Instanzen einer Klasse zugeordnet, der sie nicht angehören. Es fehlt die nötige Genauigkeit in der Unterscheidung.

Beispiele für verbreitete Regelwerks-Algorithmen sind RIPPER (*Repeated Incremental Pruning to Produce Error Reduction*) (Cohen, 1995) und 1R (Holte, 1993). Einen Überblick über die Hintergründe des *Separate and Conquer* Prinzips und *Pruning* Algorithmen bieten (Fürnkranz, Separate-and-Conquer Rule Learning, 1999) und (Fürnkranz, Pruning Algorithms for Rule Learning, 1997).

Baumbasierte Klassifizierer

Eine weitere Methode einen Klassifizierer aufzubauen stellen die baumbasierten Klassifizierer dar. Sie erzeugen in der Trainingsphase einen Entscheidungsbaum, der aus einer Wurzel, Verzweigungen und Blättern besteht. Die Klassifizierung startet an der Wurzel. An der Wurzel und an jeder Verzweigung wird eines der Attribute geprüft, und – basierend auf dem Ergebnis – zur nächsten Verzweigung oder zu einem Blatt weitergeleitet.

In den Blättern sind die Ergebnis-Klassen enthalten. In dem *Weather*-Beispiel kann dies wie in Abbildung 2 dargestellt aussehen.

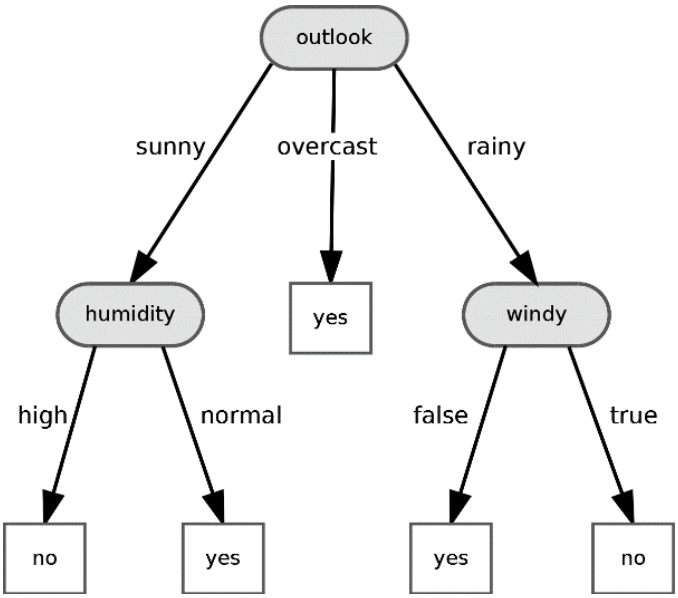


Abbildung 2 - Entscheidungsbaum generiert in WEKA mittels des Iterative Dichotomiser 3 (ID3) Verfahren

Das Ergebnis weist Parallelen zu einem Regelwerk auf und es gilt: Entscheidungsbäume lassen sich immer in ein Regelwerk überführen (Witten, Frank, & Hall, 2011).

Bei der Erzeugung eines Entscheidungsbaumes werden andere Verfahren angewandt als bei den regelbasierten Klassifizierern.

Ein Entscheidungsbaum kann rekursiv erzeugt werden. Zuerst wird ein Attribut an der Wurzel platziert und von dort aus wird für jeden möglichen Wert dieses Attributes eine Verzweigungsmöglichkeit eingeführt.
Um herauszufinden, welches Attribut sich am besten für die Wurzel eignet, werden alle Klassen aller Instanzen betrachtet, die nach der Verzweigung noch in die einzelnen Unterbäume fallen.

Die optimale Auswahl ist das Attribut, das - in diesem Schritt des Baumes - die höchste Entropie hat. Entropie ist ein Maß für die Menge an Information, die in dieser Verzweigung steckt (vgl. Abbildung 3). Um einen flachen Entscheidungsbaum zu erhalten, verzweigt er an der Stelle, die die höchste Entropie im aktuellen Teilbaum besitzt. In diesem Fall (Wurzel) ist es das Attribut "*outlook*" (siehe Abbildung 3).

Dieses Verfahren lässt sich rekursiv über die jeweiligen Verzweigungen fortsetzen, bis alle Blätter in dem Baum erzeugt sind.

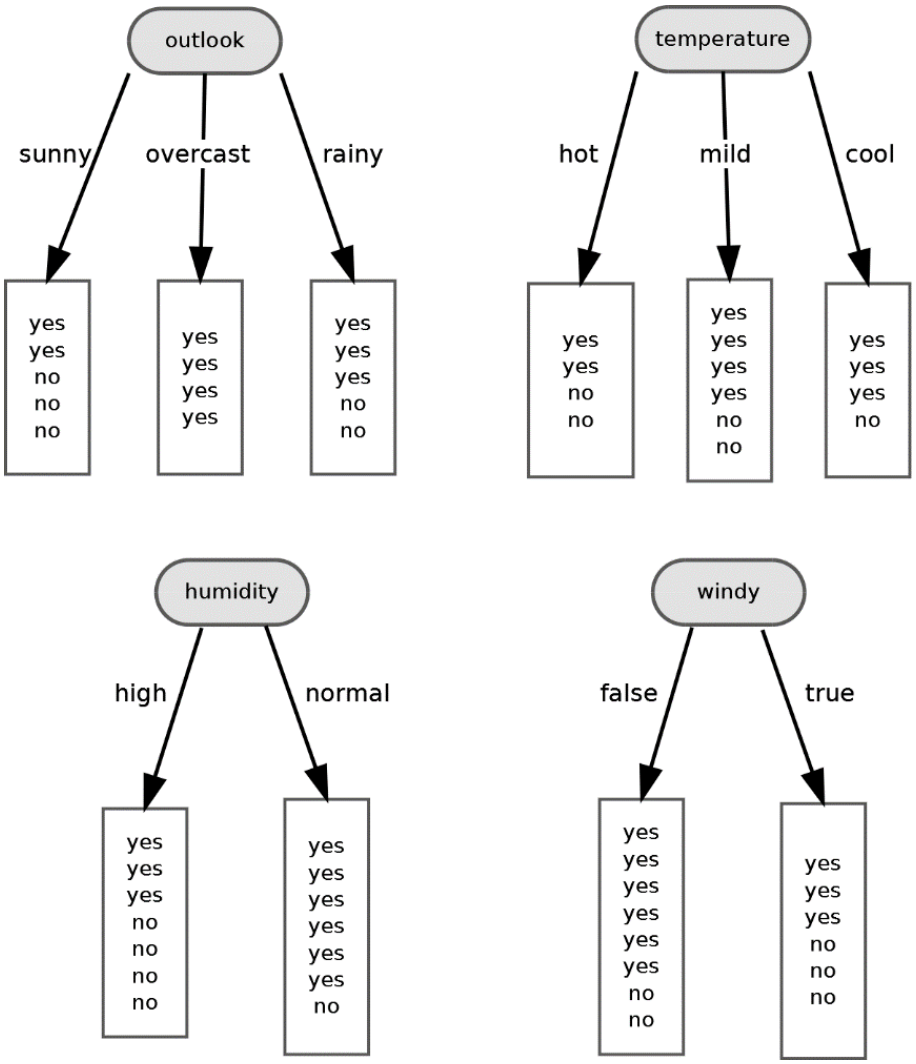


Abbildung 3 - Attributauswahl bei der Entscheidungsbaum-Erstellung (Witten, Frank, & Hall, 2011)

Der ID3-Algorithmus (Quinlan, Induction of decision trees, 1986) erzeugt einen Baum nach diesem Schema, das Resultat lässt sich in Abbildung 2 begutachten. Bei Entscheidungsbäumen werden ebenfalls *Pruning*-Verfahren angewandt, um *Overfitting* zu verhindern (Witten, Frank, & Hall, 2011).

Weitere bekannte Algorithmen für Entscheidungsbäume sind *Random Forest* (Breiman, Random Forests, 2001) und C4.5 (Quinlan, Programs for Machine Learning, 1993).

Support Vector Maschinen (SVM)

Support Vector Maschinen wurden erstmals von Vapnik und Chervonenkis entwickelt (Vapnik & Chervonenkis, 1974).

Sie basieren auf der Idee, dass sich eine binär klassifizierte Menge von Instanzen in einem n-dimensionalen Raum durch eine Hyperebene trennen lassen. Diese Hyperebene wird so in den Raum hineingelegt, dass der Abstand zu den nächstliegenden Punkten aus beiden Klassen maximal ist (Abbildung 4).

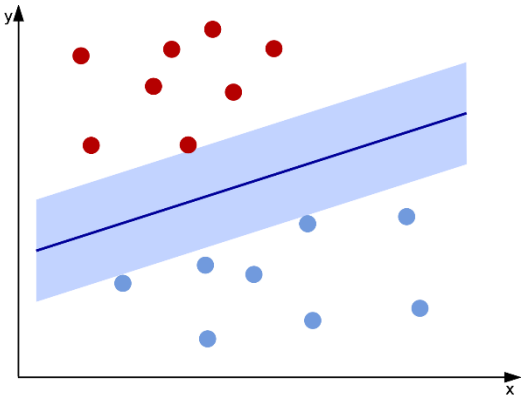


Abbildung 4 - Linear separierbare Punktemenge im zweidimensionalen Raum

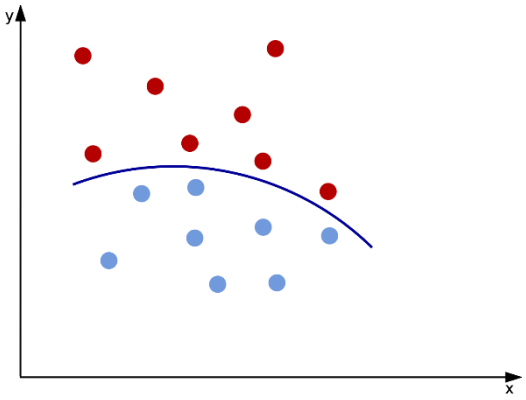


Abbildung 5 - Nicht linear separierbare Menge im zweidimensionalen Raum

Anhand der Punkte, die den geringsten Abstand zu der Hyperebene haben, wird diese mittels Stützvektoren (*Support Vectors*) beschrieben.

Dieses Vorgehen funktioniert nur dann, wenn die Objekte sich linear separieren lassen. Da dies bei realen Datensätzen nur selten der Fall ist, nutzen *Support Vector* Maschinen folgende Methode: Wenn die Objekte nicht linear trennbar sind (Abbildung 5), werden sie in einen höherdimensionalen Raum überführt.
Für jede beliebige Menge von Objekten lässt sich ein ausreichend hochdimensionaler Raum finden, in dem sie durch eine Hyperebene linear separiert werden können. Dieser Vorgang ist komplex und benötigt, je nach Dimensionalität des Raumes, hohe Rechenleistung. Der sogenannte Kernel-Trick vereinfacht diesen Vorgang. Die Anwendung geeigneter Kernel-Funktionen ermöglicht die Hin- und Rücktransformation, ohne sie wirklich auszuführen (Schölkopf & Smolar, 2001).

Durch Ausreißer und Überlappung der Klassen kann es passieren, dass eine lineare Trennung erheblich erschwert wird. Für diesen Fall wird erlaubt, dass eine gewisse Anzahl an Verletzungen des Optimierungsproblems vorkommen darf. Diese Verletzungen müssen aber so klein wie möglich sein und stellen ein weiteres Optimierungsproblem dar.

Die Thematik der *Support Vector* Maschinen gestaltet sich sehr komplex und wird unter anderem auch bei (Christianini & Shawe-Taylor, 2000) behandelt.

Neuronale Netze

Neuronale Netze (engl: *neural networks*) sind von der Biologie inspirierte Ansammlungen von künstlichen Neuronen, die miteinander verknüpft sind.
Ähnlich dem Prinzip natürlicher Neuronen sendet ein künstliches Neuron (Abbildung 6) ein Ausgangssignal (es „feuert“), wenn die lineare Kombination seiner gewichteten Eingänge eine gewisse Schwelle, dargestellt durch eine Schwellenfunktion $g()$, überschreitet (Russel & Norvig, 2010).

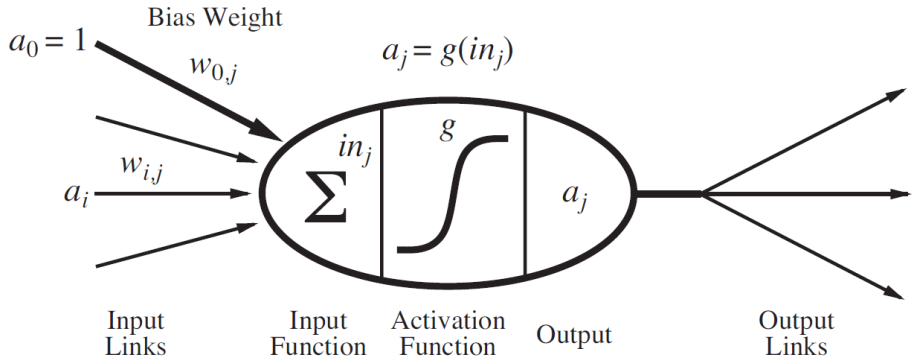


Abbildung 6 - Künstliches Neuron (Russel & Norvig 2010, S. 728)

Werden mehrere dieser künstlichen Neuronen durch sogenannte *Links* verbunden, ist das Resultat ein neuronales Netzwerk. Jeder *Link* erhält eine Gewichtung, die die Stärke und das Vorzeichen (ein *Link* kann auch negativen Einfluss haben) bestimmt und führt von einem Neuron zum nächsten.
Dies wird *Feed-Forward*-Netz genannt. *Links* können auch rekursiv auf das Ausgangsneuron zurückführen und dort zu einem Eingangsneuron werden.

Umfangreiche Erklärungen zum Thema Neuronale Netze finden sich in (Russel & Norvig, 2010) und (Bishop, 1995).

2.3.2. Genauigkeit und Kennzahlen

Es ist äußerst unwahrscheinlich, dass die zuvor erklärten Methoden für jede mögliche Instanz eine korrekte Vorhersage treffen. Dies kann verschiedene Ursachen haben. Die Trainingsdaten können Ausreißer (*Outliers*) enthalten, die entweder fehlklassifiziert wurden oder sich durch Zufall oder Fehlmessungen von den anderen Vertretern dieser Klasse unterscheiden. Sie können unter Umständen das Training beeinflussen. Bei einem Klassifizierer, der diese *Outliers* gelernt hat, kann es zu Fehlklassifizierungen aufgrund von *Overfitting* kommen. Es empfiehlt sich daher *Outlier* nicht in die Trainingsmenge aufzunehmen oder mit anderen Methoden zu vermeiden, dass sie zu einem *Overfitting* beitragen. *Pruning* wurde als Beispiel für eine solche Maßnahme bereits erwähnt.

Daten aus einem realen Umfeld sind meistens nicht optimal, sie enthalten Fehlmessungen oder zufällige Werte, und erschweren die Klassifizierung.

Wie gut ein Klassifizierer arbeitet, lässt sich durch die im Folgenden erklärten Kennwerte identifizieren.

Zuerst ist hier die *Accuracy*, die Treffergenauigkeit, zu erwähnen. Die Treffergenauigkeit errechnet sich wie folgt:

$$Treffergenauigkeit = \frac{Anzahl\ korrekt\ klassifizierter\ Instanzen}{Anzahl\ vorhandener\ Instanzen} \quad (1)$$

Hieraus lässt sich die Fehlerrate herleiten:

$$Fehlerrate = 1 - Treffergenauigkeit \quad (2)$$

Klassifizierer arbeiten unter anderem auf der Basis der Maximierung der Treffergenauigkeit. Durch das Setzen verschiedener Parameter ist ein gewisses Maß an manuellem Einfluss auf die Modellbildung und somit auf die Treffergenauigkeit möglich.

Abseits der Treffergenauigkeit gibt es noch weitere Kennzahlen.
Annahme: Es gibt einen Klassifizierer der zwischen "Wahr" und "Falsch" unterscheiden soll.

Wird ein Datensatz, der "Wahr" ist klassifiziert und erhält das Label "Wahr", ist dies ein *True-Positive* oder *Hit*.

Würde ein Datensatz, der "Falsch" ist als "Wahr" klassifiziert werden, ist dies ein *False-Positive*.

Komplementär zu diesen Werten existieren die Werte *True Negative* und *False Negative*. Ein *False Negative* wird oft auch als *Miss* bezeichnet.

Precision und *Recall* sind Kennzahlen auf Basis dieser Einstufungen und wie folgt definiert:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)$$

Diese beiden Werte können neben der Treffergenauigkeit in bestimmten Anwendungsfällen eine große Wichtigkeit erlangen. Dazu im Folgenden ein kurzes Beispiel:

Annahme: Es gibt einen Datensatz mit 100.000 Instanzen.

500 dieser Instanzen gehören in die Klasse "Wahr", und der Rest in die Klasse "Falsch". Ein Klassifizierer, der alle Instanzen pauschal als "Falsch" bezeichnen würde, erreicht hier eine Treffergenauigkeit von 99,5 %.

Die hohe Treffergenauigkeit ist in diesem Fall jedoch bedeutungslos, da bei dem Problem auf den *Recall*-Wert geachtet werden muss, der bei 0 liegt.
Der *Recall*-Wert wird auch als Sensitivität bezeichnet. Die beschriebene Klassifizierung besitzt keinerlei Sensitivität, da sie immer das gleiche Resultat liefert, unabhängig von der Instanz.

Eine wichtiges Werkzeug zur Beurteilung von Klassifizierern ist die *Receiver-Operator-Characteristic* (kurz: ROC) aus der Signaltheorie (Swets, 1964).
Die ROC stellt eine Kurve dar (Abbildung 7), die die Fehlerrate mit der Trefferrate in Korrelation setzt. Auf der X-Achse wird die Wahrscheinlichkeit eines *False-Positives* aufgetragen, auf der Y-Achse die Wahrscheinlichkeit eines *True-Positives*. Für jeden Klassifizierer lässt sich eine solche ROC-Kurve ermitteln. Die ROC-Kurve eines perfekten Klassifizierers verläuft vertikal vom Koordinatenursprung zum Punkt [0,1], die eines Klassifizieres der zufällige Ergebnisse liefert eine Diagonale vom Koordinatenursprung zum Punkt [1,1]. Eine Kurve unterhalb dieser Diagonalen impliziert, dass das Signal nicht erfasst wurde.

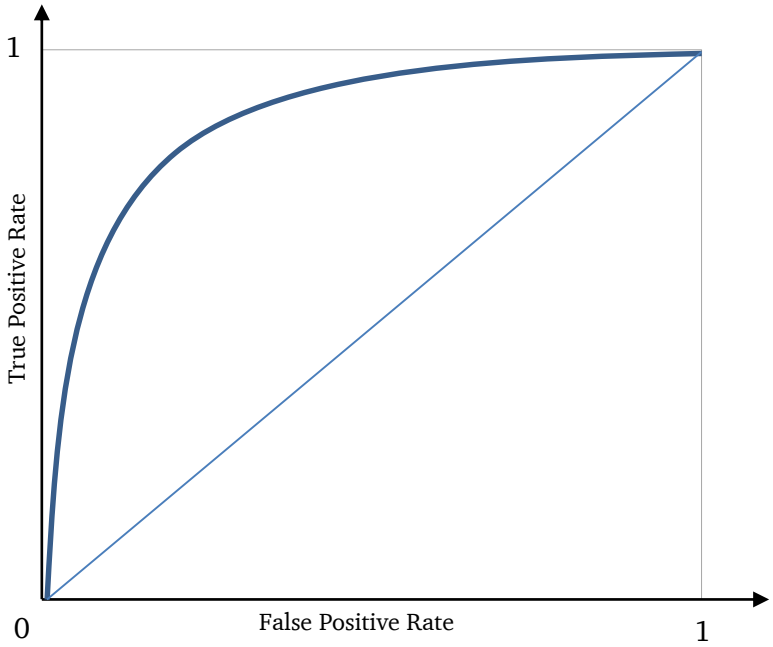


Abbildung 7 - Receiver-Operator-Charakteristik

Als wichtige Metrik basierend auf der ROC resultiert der sogenannte *AUC*-Wert (*Area under Curve*), der Flächeninhalt unter der Kurve. Bei einer perfekten Erkennung ist der Flächeninhalt 1, bei einer zufälligen Erkennung 0,5 (entsprechend dem Flächeninhalt unter der Diagonalen).

Generell ist es von der Problemstellung abhängig, welche Kennzahlen und Metriken aussagekräftig sind.
Es kann beispielsweise einen erheblichen Unterschied machen, *Misses* zu tolerieren. Dies richtet sich stark nach den damit verbundenen Kosten. Eine verpasste Warnung auf Totalschaden (*Miss*) eines wichtigen Maschinenbauteils kann um ein vielfaches teurer sein als ein *False Positive*, bei dem die Maschine voreilig mit Verdacht auf Schaden in die Werkstatt eingeliefert wurde.

Es muss demnach für jedes Problem separat bestimmt werden, welches die relevanten Metriken sind.

2.3.3. Training und Validierung

Um die in Kapitel 2.3.1 erwähnten Probleme *Overfitting* und *Underfitting* zu vermeiden, muss beim Training und der Validierung eine Vorgehensweise gewählt werden, die diese Probleme berücksichtigt.
In realen Szenarien sind die Datenmengen für ein *Supervised Learning* begrenzt. Trainiert der Klassifizierer das Modell mit den gleichen Daten, mit denen auch die Validierung ausgeführt wird, kann dies – je nach Art des Klassifizierers – zu *Overfitting* führen. Er wird dann neue Daten aus dem realen Einsatzgebiet nicht mit der Treffergenauigkeit klassifizieren können, wie es anhand der Validierung erwartet wurde.
Es gilt also sicherzustellen, dass die Validierung nicht auf denselben Daten stattfindet, mit denen trainiert wurde.
Die folgenden beiden Methoden helfen bei der Lösung dieser Problematik anhand eines fiktiven Datensatzes D.

Aufteilung in Trainings- und Testdaten

Der Datensatz D wird in zwei disjunkte Teildatensätze D_{train} und $D_{validation}$ geteilt. Die Teilmenge zur Validierung wird üblicherweise kleiner als die Trainingsmenge gewählt. Ein häufig genutztes Verhältnis (Witten, Frank, & Hall, 2011) ist 70:30 (Training:Validierung).
Bei dieser Art der Aufteilung kann es passieren, dass die resultierenden Teildatensätze nicht repräsentativ für die Gesamtpopulation sind. Um diesem Umstand entgegenzuwirken, gibt es einige Verbesserungsmethoden (Witten, Frank, & Hall, 2011), die hier nicht näher betrachtet werden.

Cross-Validation

Eine beliebte Methode zur Validierung ist die *Cross Validation* bzw. Kreuzvalidierung (Witten, Frank, & Hall, 2011).
Hierbei wird der Datensatz D in n gleichgroße, disjunkte Teilmengen (sogenannte „*Folds*“) aufgeteilt. Dies wird als n-fache Kreuzvalidierung bezeichnet.
Für jede Teilmenge D_i mit $i \in \{1,..,n\}$ wird eine Komplementärmenge $D \setminus D_i$ erzeugt. Auf dieser Komplementärmenge wird das Training ausgeführt, und mit D_i validiert. Der Vorgang wird für alle n Teilmengen durchgeführt. Die Genauigkeit wird als Mittel aus den n Durchläufen berechnet.
In dieser Diplomarbeit wird ausschließlich die Kreuzvalidierung mit n =5 (*5-Fold Cross Validation*) genutzt. Die Wahl des Wertes von n ist dabei nicht ausschlaggebend, Werte zwischen 5 und 20 sind gängige Parametrisierungen (Witten, Frank, & Hall, 2011, S. 152ff).

2.3.4. *WEKA*-Suite (*Waikato Environment for Knowledge Analysis*)

WEKA ist ein von der Universät Waikato (Neuseeland) entwickelte Sammlung von verschiedenen *Machine Learning* Instrumenten.

Es beinhaltet eine graphische Oberfläche zur Datenanalyse, Klassifizierung und zum *Clustering*. In *WEKA* sind umfangreiche Sammlungen von Algorithmen verfügbar und können beispielsweise zur Klassifizierung eingesetzt und mit den integrierten Evaluationsmethoden ausgewertet werden. Für diese Zwecke greift *WEKA* auf das eigenentwickelte Dateiformat ARFF (Attribute Relation File Format) zurück.

Des Weiteren verfügt *WEKA* über *Client-Server* Systeme zum verteilten Rechnen, um große Datenmengen bewältigen zu können.

WEKA wird in Java entwickelt und kann als Open-Source Software genutzt und unter den Bedingungen der GPL verwendet werden.

Weitere Informationen und die Software selbst können auf der Webseite der Universität von Waikato unter http://www.cs.waikato.ac.nz/ml/weka/index.html bezogen werden.

WEKA wird in dieser Diplomarbeit zur Durchführung der Versuchsreihen und zur Evaluation eingesetzt.

2.4. Verwandte Arbeiten

Im Folgenden werden einige Arbeiten anderer Forscher präsentiert, die bei der Anfertigung dieser Diplomarbeit zur Findung und Ausarbeitung der Methoden grundlegende Ideen und Informationen lieferten.

Diese Arbeiten beschäftigen sich mit Ausfällen und Schäden von mehr oder weniger komplexen Systemen. Es bestehen viele Parallelen, weswegen einige der dort beschriebenen Methoden in angepassten Varianten übernommen werden konnten, und andere als Inspiration für abgewandelte Implementierungen dienten.

Diese Diplomarbeit unterscheidet sich im Wesentlichen dadurch, dass ein breiteres Spektrum an Anwendungsfällen vorliegt. Sowohl eine sehr maschinennahe als auch eine abstrakte Betrachtung der komplexen Systemlandschaft einer Lokomotive sind möglich und führen jeweils zu Resultaten.

„A Survey of Online Failure Prediction Methods“

In dieser Studie (Salfner, Lenk, & Malek, 2010) werden Methoden aus dem Bereich „*Online failure prediction*“ vorgestellt. Eine Taxonomie wird aufgebaut, welche die verschiedenen Teilbereiche und Vorgehensweisen in Kategorien und Unterkategorien einordnet. Die Studie bietet einen guten Überblick und Referenzen zur Vertiefung der Thematik.

„Wind Turbine Gearbox Condition Monitoring with AAKR and Moving Window Statistic Methods“

Bei dieser Arbeit (Guo & Bai, 2011) im Bereich der Windkraftwerke wird die Methode der autoassoziativen Kernel Regression (AAKR) eingesetzt, um das Normalverhalten eines Windturbinen-Getriebes zu modellieren.
Hierfür werden Temperaturvektoren (Momentaufzeichnungen der Sensoren) genutzt, um eine Matrix aus Vektoren zu erstellen. Diese Matrix deckt den gesamten Bereich ab, in dem ein funktionierendes Getriebe arbeitet.
Getriebeausfälle kündigen sich an, wenn das Residuum zwischen Modell und aktuell gemessener Temperatur eine signifikante Größe überschreitet.
Eine *Moving-Window* Methode wird genutzt, um zeitabhängige Mittelwerte für Residuen und Standardabweichung zu errechnen.

In dieser Arbeit werden Echtdaten mit generierten Daten vermischt, um Getriebeschäden zu simulieren und die Methodik evaluieren zu können.

„*Predicting Computer System Failures Using Support Vector Machines*“

In diesem Paper (Fulp, Fink, & Haack, 2008) beschreiben die Autoren einen Ansatz zur Vorhersage von Festplatten-Ausfällen anhand von Logfiles. Logfiles zeigen Statusänderungen auf einem System an. Ist eine Meldung im Logfile nicht ausreichend, um eine Ausfall-Vorhersage zu ermöglichen, so kann eine Sequenz oder ein Muster im Auftreten der Meldungen jedoch auf einen Ausfall hinweisen.

Mit einer *Sliding-Window* Methode werden Subsequenzen beobachtet, anhand derer eine SVM trainiert wird. Die SVM ordnet diesen Subsequenzen dann eine Klasse (*fail* oder *non-fail*) zu. Die Forscher konnten mit dieser Methode Festplatten-Ausfälle mit bis zu 73 % Treffergenauigkeit und zwei Tagen Vorlauf vorhersagen.

„*Data mining for prediction of aircraft component replacement*“

Dieses Paper (Létourneau, Famili, & Matwin, 1999) kommt aus dem Luftfahrtbereich. Die Autoren beschäftigen sich mit der Problematik Daten der vielen Einzelsysteme eines Flugzeuges und der im Flugbetrieb eingesetzten stationären Systeme zusammenzuführen, weiterzuverarbeiten und damit Probleme an Flugzeugkomponenten vorherzusagen.
In dem Ansatz werden *Data Mining* und *Machine Learning* Techniken verwendet, um komplexe historische Daten zum Erlernen von Modellen für die Vorhersage zu nutzen.
Automatisierte Selektion relevanter Daten, automatisches Labeling von Instanzen und Evaluationsmethoden zur Bewertung der Resultate werden vorgestellt.

„*Entwicklung und Demonstration eines integrierten Systems zur Zustandsüberwachung von Gasturbinen*“

In dieser Dissertation (Lipowsky, 2010) widmet sich der Autor der Zustandsüberwachung von Gasturbinen. Er erörtert die unterschiedliche Handhabung von graduell auftretenden Alterserscheinungen im Gegensatz zu spontan auftretenden Einzelereignissen und entwickelt ein integriertes System, welches beide Arten von Ereignissen detektieren und diskriminieren kann.

Die Problematik der Diagnose von Einzelereignissen löst er mit einer kombinatorischen *Least-Squares*-Lösung; die der graduellen Verschlechterung hingegen mit Verfahren der nichtlinearen Optimierung.

2.5. Systemarchitektur der Condition based maintenance

In Abbildung 8 sind der Datenfluss und die durchgeführten Implementierungsarbeiten im Rahmen dieser Diplomarbeit aufgezeigt. Die vorhandenen Daten werden durch Export aus den Bestands-Systemen und Import in eine neue Datenbank aufbereitet und für die Weiterverarbeitung zugängig gemacht. Mit der neuen Datenbank kann sichergestellt werden, dass die Originalsysteme (UDG und SAP) mit ihrer komplexen Business-Logik und ihren Sicherheitsmechanismen unangetastet bleiben. Die Daten sind so ohne Eingriff in betriebliche Abläufe verfügbar.

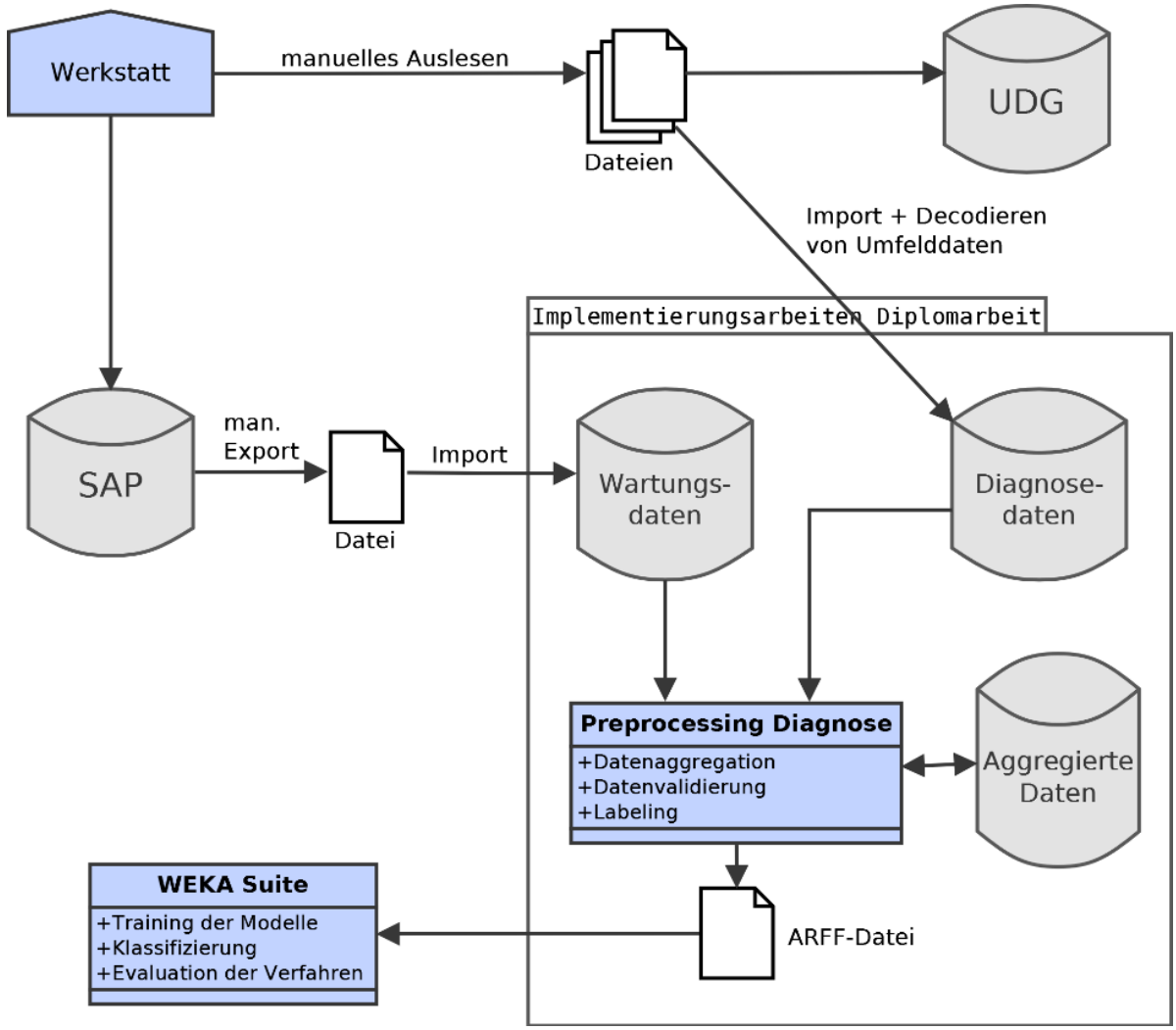


Abbildung 8 - Architektur Diagnose (Ist-Zustand)

Zu den Implementierungsarbeiten dieser Diplomarbeit gehören nicht nur die vorbereitenden Schritte, sondern auch die Weiterverarbeitung der Daten bis hin zur Evaluation der Ergebnisse. Auf die genauen Vorgänge wird in den entsprechenden Kapiteln eingegangen.

3. Methodik

In dieser Arbeit werden zwei Versuchsreihen mit unterschiedlichen Herangehensweisen und Daten von verschiedenen Lokomotiv-Baureihen durchgeführt.
Beide Verfahren arbeiten mit aufgezeichneten Diagnosedaten aus dem Flottenbestand der DB Schenker Rail AG, und zielen darauf ab, Schäden an den Lokomotiven rechtzeitig vorherzusagen.

Die zugrundeliegenden Versuchsaufbauten und Hypothesen werden in diesem Kapitel vorgestellt. Im darauffolgenden Kapitel „Versuchsdurchführung“ werden die Datenlage und Bearbeitungsschritte beschrieben. Im Kapitel „Ergebnisse“ folgen die konkreten Versuchsresultate und deren Interpretation.

3.1. Versuchsreihe 1 - Vorhersage von Schadfällen durch Diagnosecode-Häufigkeit

Um einen langfristigen Nutzen aus der Speicherung der Diagnosedaten zu beziehen, wird ein Verfahren erarbeitet, welches Schäden anhand der Häufigkeit des Auftretens der Diagnosemeldungen vorhersagen soll.

3.1.1. Arbeitshypothese

Die mit dieser Versuchsreihe zu untersuchende Arbeitshypothese lautet:

Hypothese H1: *Schäden an Bauteilen kündigen sich durch eine Zunahme von bestimmten Diagnosemeldungen vor dem Ausfall des Bauteils an. Durch Ermittlung der Häufigkeit der aufgetretenen Diagnosecodes in einem bestimmten Zeitfenster vor dem Ausfall lassen sich Schäden vorhersagen.*

Es wird angenommen, dass bei einem normalen Tagesverlauf die betrachteten Diagnosemeldungen in einer gewissen Häufigkeit auftreten, die Summe sich um einen bestimmten Mittelwert bewegt. Ein solcher Verlauf könnte wie Abbildung 9 aussehen (fiktiv). *T* bezeichnet den aktuellen Tag, *T-n* den n-ten Vortag.

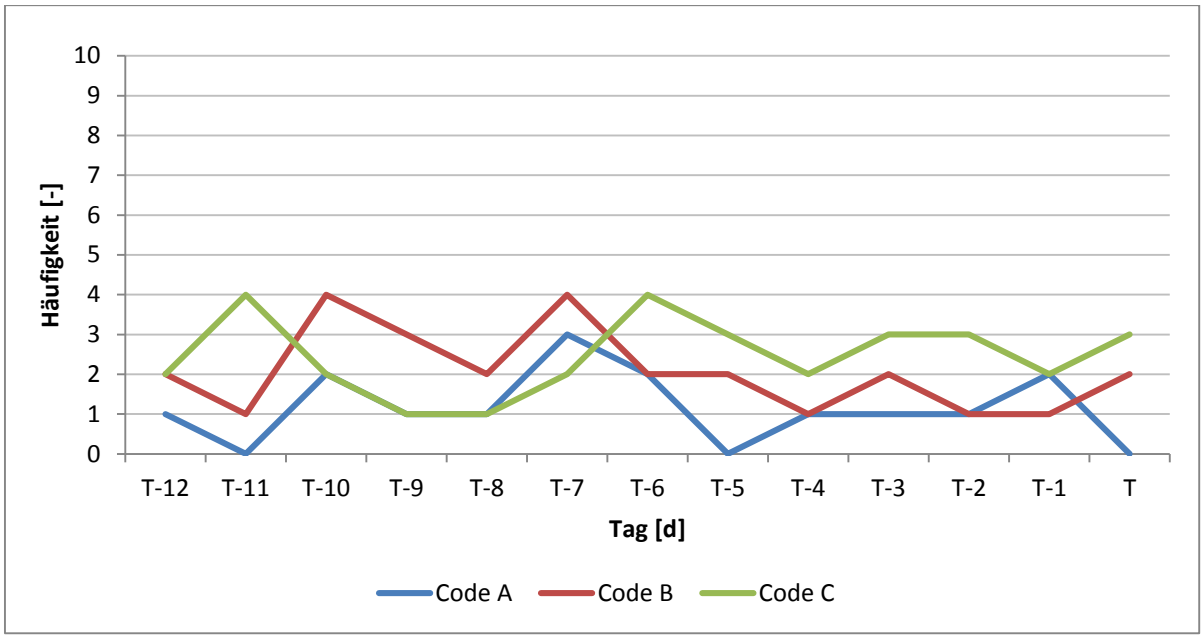


Abbildung 9 - Beispiel für stochastischen Verlauf von Diagnosecode-Auftreten

Zudem wird angenommen, dass bei einem bevorstehenden Schaden bestimmte Diagnosemeldungen häufiger auftreten und den Schaden vorankündigen (Abbildung 10).

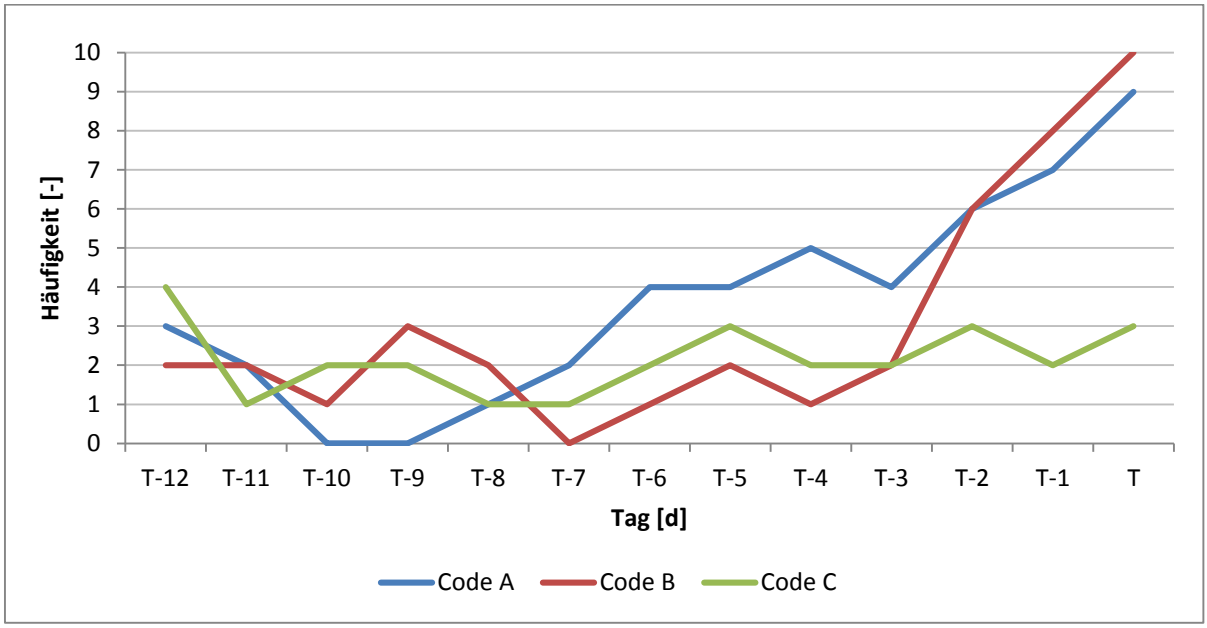


Abbildung 10 - Beispiel für Verlauf von Diagnosecode-Auftreten im bevorstehenden Schadfall, wie von der Arbeitshypothese angenommen

Damit wird eine Methode geschaffen, die mittels *Machine Learning* Verfahren die Vorhersage und Bestimmung der Schäden ermöglicht. Dabei ist insbesondere die Frage interessant, in welchem Zeitraum die Daten betrachtet werden müssen und wie viele Tage im Voraus eine Warnung stattfinden kann.
Die Ergebnisse ermöglichen eine zuverlässige Erkennungsrate und eine frühzeitige Warnung, so dass defekte Lokomotiven noch vor dem Ausfall der Werkstatt zugeführt werden können.

Die Ermittlung der Warnungs- und Betrachtungszeiträume erfolgt experimentell. Verschiedene Methoden zur Verarbeitung der Daten im Betrachtungszeitraum werden eingesetzt. Für die Klassifikation werden Baum- und Regelbasierte Klassifizierer, sowie *Support-Vector* Maschinen-Verfahren eingesetzt.

Alternativ zu einem Verfahren, welches auf Zeiteinheiten basiert, könnte die Betrachtung auch in Hinsicht auf den Kilometerstand der Lokomotive durchgeführt werden. In diesem Fall ist eine zeitliche Einteilung allerdings günstiger, da der aktuelle Kilometerstand nicht zu jedem Datensatz vorliegt.

Um die Hypothese zu untermauern und einen groben Eindruck von den Zeiträumen zu erhalten, die für die Betrachtung zielführend sind, wurde eine Voruntersuchung durchgeführt. Die Ergebnisse der Voruntersuchung befinden sich im Anhang „Voruntersuchung Schadfälle 185er.pdf“.
Dabei wurden für einige ausgewählte Schäden die Auftrittshäufigkeiten bestimmter Diagnosecodes betrachtet. Hierzu wurden Histogramme der Diagnosecodes in verschieden großen Zeiträumen erstellt.
Es zeigt sich, dass ein Betrachtungszeitraum von mehreren Tagen nötig ist, um erkennbare Muster zu erhalten. Als Folgerung hieraus wird die feinste Granularität für eine Zählung der Häufigkeiten auf einen Tag festgelegt.

Ein ähnliches Problem in der Vorhersage von Festplattendefekten wurde von Fulp, Fink und Haack (Fulp, Fink, & Haack, 2008) mit vergleichbaren Methoden gelöst. Bei dem in dieser Diplomarbeit untersuchten Fall ist die Datendichte geringer und die Vorhersagezeit größer.

3.1.2. Datenverarbeitung von den Quellen bis zur Evaluation

Für diese Versuchsreihe liegen zusätzlich zu den Diagnosedaten auch Daten aus den Werkstätten vor. Darin sind alle Tätigkeiten der Werkstätten verzeichnet. Die Werkstattdaten dienen zur Ermittlung geeigneter Schadensfälle.

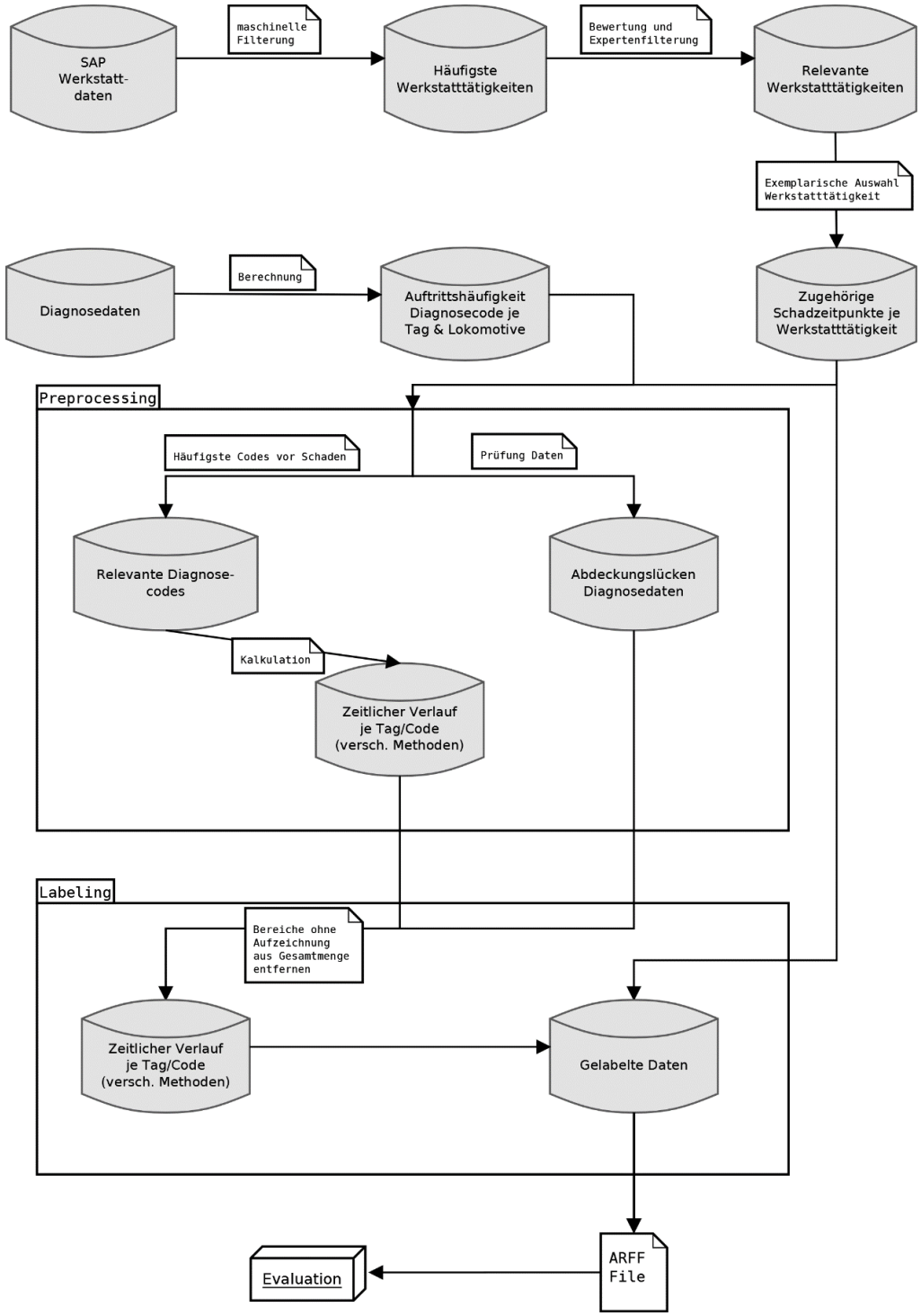


Abbildung 11 - Verarbeitungsfluss Diagnose- und Werkstattdaten bis zur Evaluierung

Der Ablauf der Datenverarbeitung aus beiden Quellen gestaltet sich wie in Abbildung 11. Verglichen mit Abbildung 8 stellt dies eine konkrete Variante der Implementierungsarbeiten dar.

In der Versuchsdurchführung werden die einzelnen Schritte dieses Prozesses genauer beschrieben.

3.2. Versuchsreihe 2 – Zentralschraubenbruch

In dieser Versuchsreihe wird eine Schadensvorhersage basierend auf Temperaturmesswerten der Fahrmotoren einer Lokomotive erarbeitet.

Bei den Zügen der 189er Baureihe von Siemens tritt in unregelmäßigen Abständen ein Bruch der Zentralschraube eines Fahrmotors (FM) auf.
In Abbildung 12 ist eine Antriebsachse dargestellt. Lokomotiven der Baureihe 189 besitzen vier Antriebsachsen, aufgeteilt auf zwei Drehgestelle. An jeder dieser Achsen greift ein FM an. Die FM dieser Baureihe sind Elektromotoren mit einer Leistung von 1633 kW und einer Masse von 2700 kg. Diese werden im Folgenden mit FM1 bis FM4 bezeichnet.
Von der Motorwelle wird die Kraft auf ein Ritzel übertragen. Die Zentralschraube stellt eine schlüssige Verbindung zwischen Motor und Ritzel her. Von dort erfolgt eine weitere Kraftübertragung auf das Getriebe bis hin zur Radsatzwelle.

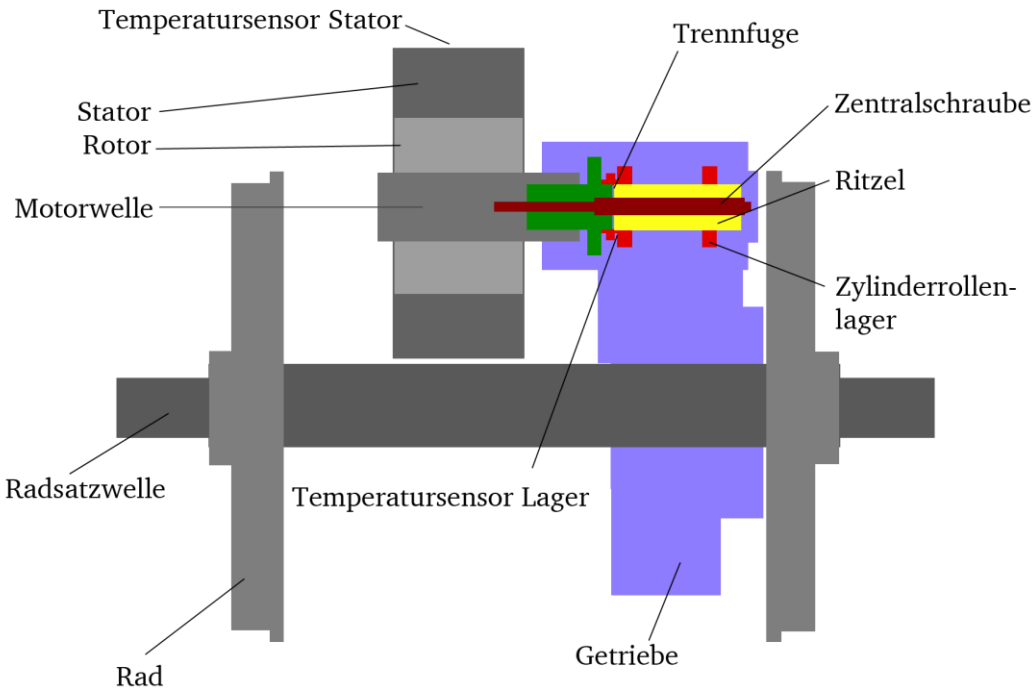


Abbildung 12 - Schematische Skizze des Antriebsstrangs der Siemens 189er Baureihe

Aufgrund von Scherkräften bei bestimmten Fahrsituationen kommt es in seltenen Fällen zu einem Bruch der Zentralschraube an der Trennfuge. Das abgebrochene Teil fällt dadurch in den laufenden FM hinein, wo es für gewöhnlich eine vollständige Zerstörung des FM verursacht.

Die Zentralschrauben sind auf die komplette Lebensdauer einer Lokomotive ausgelegt. Es wird keine regelmäßige Prüfung der Schrauben durchgeführt.
Für eine Überprüfung muss die Schraube mit einem Ultraschallgerät auf Risse untersucht werden. Dies ist eine aufwendige und kostspielige Prüfung, die auch zu Irrtümern führen

kann. Die Lokomotive ist für mehrere Tage in der Werkstatt.
Diese Art der Untersuchung findet nie ohne konkreten Verdacht auf Bruchgefahr statt. Der Bruch kündigt sich jedoch nicht spürbar für Lokführer oder Werkstattpersonal an, was eine Untersuchung auf Verdacht hin erschwert.

Die Einführung einer regelmäßigen Überprüfung ist aufgrund der hohen Kosten und der niedrigen Auftrittswahrscheinlichkeit nicht vorgesehen. Die Brüche treten – bei einer Flottengröße von 90 Lokomotiven dieser Baureihe – lediglich vier bis fünf mal pro Jahr auf.
Es wird vermutet, dass ein Zentralschraubenbruch schleichend über einen Zeitraum von mehreren Wochen oder Monaten erfolgt. Es liegen hierzu aber keine genaueren Daten vor, da, falls ein schleichender Bruch frühzeitig erkannt werden sollte, direkt ein Austausch der Schraube erfolgt.

Sollte der Fall des Zentralschraubenbruchs eintreten, wird durch den Austausch des FM und damit zusammenhängenden Kosten ein Schaden in Höhe mehrerer hunderttausend Euro verursacht.

Um Kosten und Zeit zu sparen, wird ein Verfahren gesucht, welches eine rechtzeitige Erkennung eines Zentralschraubenbruchs ermöglicht.

Mit einem ähnlichen Problem bei Getrieben von Windkraftwerken befassten sich auch Guo und Bai (Guo & Bai, 2011).

3.2.1. Arbeitshypothese

Als Basis für die Entwicklung eines Prozesses zur Vorhersage des Zentralschraubenbruchs wird folgende Hypothese statuiert.

Hypothese H2. *Ein Zentralschraubenbruch kündigt sich in einem zeitlichen Fenster von mehreren Wochen durch noch unbekannte intraindividuelle Abweichungen der FM-Temperaturen einer Antriebsachse und ebenfalls noch unbekannte interindividuelle Abweichung der Temperaturen der vier FM einer Lokomotive an.*

Die im Folgenden beschriebenen Methoden und Abläufe nutzen diese Hypothese und versuchen sie zu validieren.

3.2.2. Temperaturmessung am Motor

Bei den FM einer Lokomotive werden an zwei Stellen Temperaturen gemessen. Am Stator und an der meistbelasteten Stelle des Zylinderrollenlagers (siehe Abbildung 12).
Die Belastungen der einzelnen FM sind im Normalbetrieb beinahe identisch, woraus folgt, dass die Temperaturen der vier Sensoren am Lager und die Temperaturen der vier Sensoren am Stator nahe beieinander liegen sollten.
Die Temperaturen selbst sind abhängig von der Betriebsdauer und den Außentemperaturen und können größere Schwankungen aufweisen. Die Temperaturen der vier FM sollten untereinander allerdings nur geringfügige Abweichungen aufweisen, die durch unterschiedliche Wärmeabfuhr bedingt sind.

3.2.3. Datenfluss-Diagramm

In Abbildung 13 ist der Verarbeitungsfluss für diesen Versuchsaufbau dargestellt. In der Versuchsdurchführung werden diese Schritte näher erläutert. Sie stellen eine Konkretisierung der Implementierungsarbeiten aus Abbildung 8 dar.

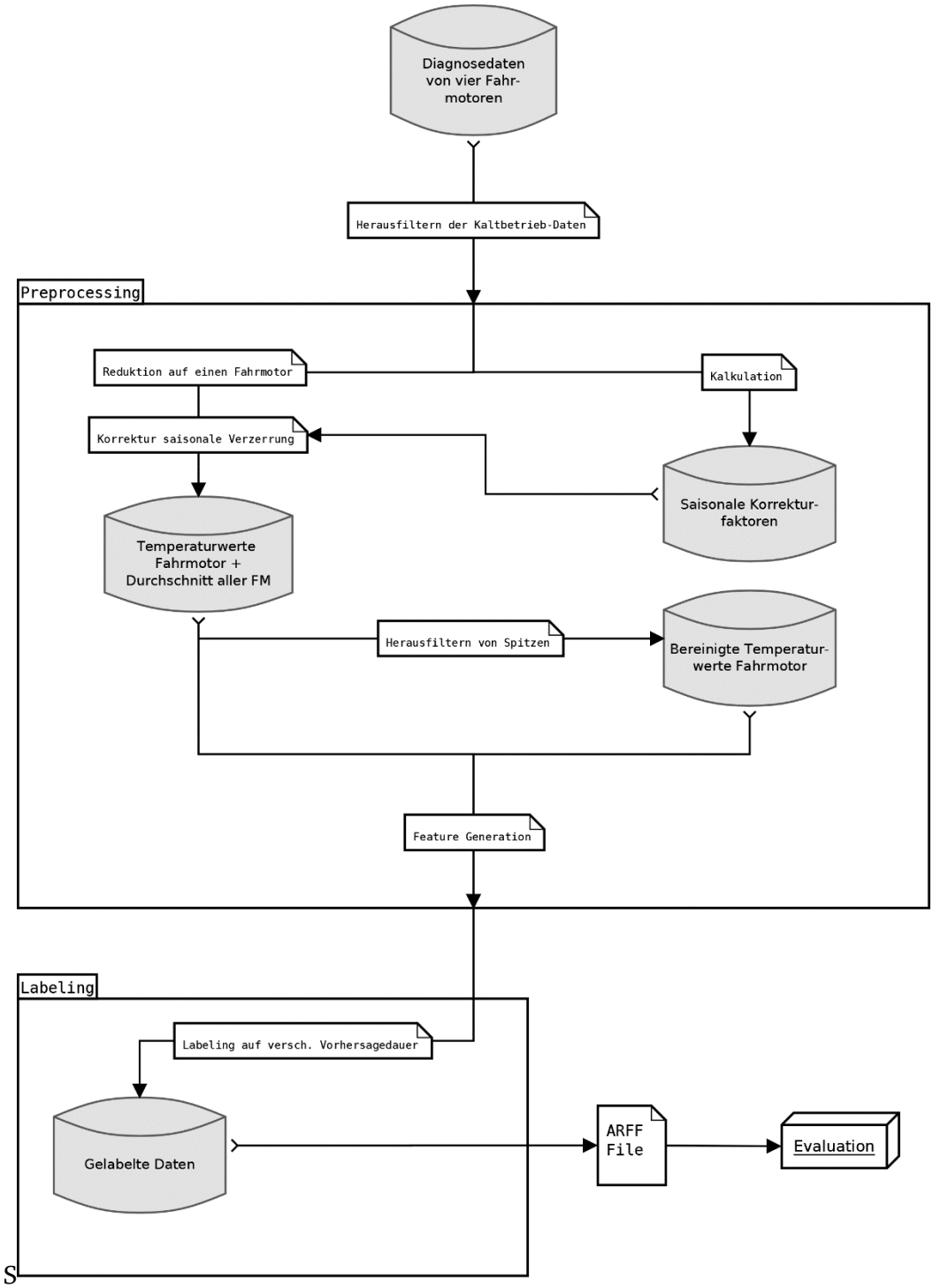


Abbildung 13 - Verarbeitungsfluss Versuchsreihe 2: Zentralschraubenbruch

4. Versuchsdurchführung

In diesem Kapitel werden für Versuchsreihe 1 und 2 alle Schritte der Versuchsdurchführung beschrieben.

4.1. Versuchsreihe 1

Im Folgenden werden die für diese Versuchsreihe vorliegenden Daten, die Auswahl geeigneter Schadensfälle für eine Vorhersage und die Schritte des Preprocessing und Labelings erläutert.

4.1.1. Voraussetzungen

Für diese Versuchsreihe liegen Diagnosedaten-Aufzeichnungen der Baureihe 185 des Herstellers Bombardier vor. In diesen Daten sind, in einer Form ähnlich der eines Logfiles in Computersystemen, zeitlich aufeinanderfolgende Diagnosemeldungen enthalten. Diagnosemeldungen enthalten sowohl Protokollmeldungen als auch Fehlermeldungen von den einzelnen Systemen und Subsystemen, die auf einer Lokomotive verbaut sind. Es gibt bei dieser Baureihe 37 Hauptsysteme mit insgesamt 70 Subsystemen.

Zum Verfassungszeitpunkt dieser Arbeit sind ca. 14,5 Monate an Diagnosedaten der 185er Baureihe, und zwar vom 01.06.2012 bis zum 19.08.2013, verfügbar. In diesem Zeitraum waren 187 Lokomotiven der Baureihe 185 in Betrieb. Die Diagnosedaten sind lückenhaft und für keine Lokomotive über den kompletten Zeitraum vollständig. Im besten Fall sind für eine Lokomotive an 55 % der Tage des Betrachtungszeitraumes Daten verfügbar. Bei einigen Lokomotiven weniger als 10 %. Die exakten Werte befinden sich im Anhang „Vollständigkeitsanalyse Diagnosedaten 185er.xlsx“.

Die Lücken in den Datenbeständen haben verschiedene Ursachen: Eine Lokomotive kann mehrere Tage stillstehen, zum Beispiel wenn Sie nicht benötigt wird oder auf eine Reparatur wartet. In dieser Zeit wird nicht aufgezeichnet.
Es kann passieren, dass aufgezeichnete Daten verloren gehen. Bedingt dadurch, dass die Diagnosedaten manuell vom Werkstatt-Team ausgelesen und abgespeichert werden, geschehen Fehler (Unachtsamkeit, Vergessen), was den Verlust der Daten zur Folge hat.

4.1.2. Schadbild-Auswahl

Um eine sinnvolle Auswahl an vorherzusagenden Schäden zu treffen, werden die historischen Werkstattdaten einbezogen. Diese Daten befinden sich in einem SAP System (siehe Abbildung 11, SAP Werkstattdaten) und sind nur für speziell berechtigte Mitarbeiter von DB Schenker Rail zugängig. Sie beinhalten alle Wartungs- und Reparaturtätigkeiten je Lokomotive, sowie Zeitpunkt der Meldung des Schadens und Informationen zur Bearbeitungsdauer. Der Meldezeitpunkt eines Schadens ist der Tag, an dem der Schaden auftritt. Für die Analyse wird ein Auszug dieser Daten für den gewählten Auswertungszeitraum genutzt.

Im ersten Schritt werden die Wartungsarbeiten gefiltert. Die Wartungstätigkeiten sind mit einem Tätigkeits-Code bezeichnet. Unter diese Tätigkeits-Codes fallen Servicetätigkeiten der Werkstatt, Reinigungstätigkeiten und der Tausch von Verbrauchsmitteln auf der Lokomotive, die nicht relevant für die Betrachtung sind. Die Tätigkeitsgruppen sind als Kategorien und

Unterkategorien in einem zweistelligen Code definiert. Der erste Buchstabe bezeichnet die Kategorie, der zweite die Unterkategorie:

Tabelle 2 - Tätigkeitsgruppen (Werkstatt)

Kategorie-Bezeichnung	
B	Bautätigkeiten
C	Spanende Fertigung
D	Daten / Software
I	Instandsetzung
O	Oberflächenbehandlung
P	Prüfen
S	Servicetätigkeiten
W	Wartung

Beispiel: Der Code „**BE**“ bezeichnet einen Vorgang der Kategorie „**B**autätigkeiten – **E**rsetzen“. Eine genaue Definition aller Tätigkeitscodes befindet sich im Anhang „Übersicht EFK Tätigkeiten.xlsx“.

Als nächster Schritt wird eine grobe Vorauswahl durch einen Werkstattexperten getroffen. Er schränkt die Auswahl ein.
In unserer Betrachtung konzentrieren wir uns auf Tätigkeitscodes der Kategorien „B“ und „I“, da diese nicht-reguläre Tätigkeiten wie Austausch oder Reparaturen enthalten. Die genauen Tätigkeitscodes, die betrachtet werden, lauten wie folgt: "BE", "BT", "BU", "BW", "BX", "BY", "DT", "IE", "IF", "IG", "IL", "IO", "IP", "IS", "IT", "IV".

Eine Liste aller Tätigkeiten dieser Kategorien – nach Häufigkeit des Auftretens sortiert – findet sich im Anhang „Liste aller Wartungstätigkeiten sortiert nach Summe.xlsx“.

Mit Hilfe des Werkstattexperten wird eine weitere Filterung vorgenommen.
Zuerst werden alle Tätigkeiten als „relevant“ markiert, die nach Ansicht des Werkstattexperten für Vorhersagezwecke interessant sind (siehe Abbildung 11, Relevante Werkstatttätigkeiten). Das bedeutet, diese Tätigkeiten hängen mit Austausch oder Reparatur von Komponenten der Lokomotive zusammen, die an Diagnosemeldung erzeugende Systeme angekoppelt sind (sich potentiell vorhersagen lassen).
Der Tausch eines Scheibenwischers ist zum Beispiel ein häufiger Vorgang, aber in diesem Fall nicht relevant. Das Einstellen der Empfangsantenne hingegen ist relevant, da die Empfangsantenne mit dem „LZB“-System (Linienförmige Zugbeeinflussung) verbunden ist, welches Diagnosemeldungen erzeugt. Hier kann unter Umständen ein Ausfall vorhergesagt werden.

In einem weiteren Filterungsschritt werden diejenigen Wartungstätigkeiten markiert, bei denen es – nach Ansicht des Experten – wahrscheinlich ist, dass sie sich durch einen Anstieg von bestimmten Diagnosemeldungen erkennen lassen. Im selben Schritt nimmt der Experte eine Einschränkung vor, von welchen Systemen und Subsystemen für diese Art von Schaden die Diagnosemeldungen betrachtet werden sollen.

Die Betrachtung wird auf diese Systeme eingegrenzt. Diese Eingrenzung ist wichtig, da durch die schiere Menge der vorhandenen Diagnosecodes die Attributauswahl erheblich erschwert ist. Eine Attributauswahl ließe sich auch auf Basis der Gesamtmenge aller Diagnosecodes treffen. Aufgrund der Anzahl der Codes wurde diese Vorgehensweise allerdings verworfen und eine manuelle Vorauswahl als sinnvoll erachtet.

Eine Liste aller Tätigkeiten – nach Häufigkeit sortiert und mit Markierungen für die zu betrachtenden Tätigkeitcodes versehen – befindet sich in „Gefilterte Liste Tätigkeiten.xlsx“.

Nach dem Auswahlprozess ergeben sich die Schadfälle wie in Tabelle 3 dargestellt.

Tabelle 3 - Häufigste Wartungsarbeiten nach Filterung durch Experte (Zeitraum: 14,5 Monate)

Gruppe	Code	Bezeichnung	Anzahl	System
EBGCBCT	IS	Antriebsteuergerät (ASG) instand setzen	142	System ASG1 und ASG2
EBJDZCB	IS	LZB instand setzen	126	System LZB
EBJDBEF	IE	LZB, Empfangsantenne einstellen	93	System LZB

Diese erfüllen die Auswahlkriterien und sind im Betrachtungszeitraum am häufigsten aufgetreten.

4.1.3. Preprocessing

In diesem Abschnitt werden die einzelnen Schritte erklärt, die zur *Feature*-Erstellung und dem anschließenden *Labeling* dienen. Diese Schritte stellen die Basis für die Klassifizierer bereit und sind ausschlaggebend für das spätere Ergebnis.

Feature-Selektion

Um eine Klassifizierung zu ermöglichen, wird eine Auswahl an zu beobachtenden Attributen vorgenommen. Ein Attribut ist in diesem Fall die Häufigkeit des Auftretens eines bestimmten Diagnosecodes in einem Zeitintervall.
Es gibt 6908 verschiedene Diagnosecodes. Je nach Softwareversion, die auf einer Lokomotive installiert ist, kann diese Menge leicht variieren. Neuere Versionen enthalten mehr Codes als ältere, in der Praxis kommen aber nur selten neue hinzu. Die Bedeutung der Codes bleibt über alle Versionen hinweg gleich, eine Vergleichbarkeit ist daher gegeben. Von diesen 6908 Diagnosecodes sind im kompletten Betrachtungszeitraum 1990 verschiedene Codes zu beobachten. Insgesamt liegen über 3,7 Millionen Einträge für den Betrachtungszeitraum vor. Der am häufigsten auftretende Code kommt mehr als 200.000 mal vor.

Mit der Experteninformation aus dem vorigen Abschnitt werden in der *Preprocessing*-Phase für jede betrachtete Schadvariante die 30 am häufigsten auftretenden Diagnosecodes der zugehörigen Systeme ermittelt Diese Attribute werden als *Features* für die Klassifizierung in den folgenden Schritten verwendet (siehe Abbildung 11, Relevante Diagnosecodes).

Feature-Generation

Um eine Vorhersage und Erkennung zu ermöglich, wird ein Klassifizierer genutzt und mit *Features* aus den bekannten Daten trainiert.

Klassifizierer arbeiten auf einem Vektor von *Features*. Es können beliebig viele *Features* in diesem Vektor enthalten sein.

Beispiel:

Sei *D* die Menge aller Diagnosecodes.
Sei *D'* die Menge aller betrachteten Diagnosecodes. D besitzt die Kardinalität n.
Sei $C_i \in D'$ mit $i \in (1...n)$ die Menge aller *Features*.

$D' = [C_1, C_2, C_3]$ = [Code A, Code B, Code C] mit n = 3

Zu jedem Tag x lässt sich für die betrachteten Diagnosecodes ein Attributsvektor A erstellen. Dieser enthält die Summe, wie oft die Diagnosecodes $C_i \in D'$ an diesem Tag aufgetreten sind. Ein solcher Vektor für Tag *T-7* aus Abbildung 10 lautet wie folgt:

$$A_x = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \tag{5}$$

Um einen zeitlichen Verlauf zu untersuchen, wird mehr als eine Momentaufnahme benötigt. Dazu werden in einem Zeitraum von v Tagen die Werte mittels verschiedener Funktionen aggregiert (siehe Abbildung 11, Zeitlicher Verlauf je Tag/Code). Die Aggregationsfunktionen bilden verschiedene Eigenschaften des zeitlichen Verlaufs wie folgend beschrieben ab.

Definition 4.1.3.1. v ist die Anzahl der vorangehenden Tage, die betrachtet werden.

Die Funktion $mittel(x_1, \dots, x_n)$ berechnet den Durschnitt (Mittelwert) einer Zahlenreihe.

$$mittel(x_1, \dots, x_n) = \frac{1}{n} * \sum_{i=1}^{n} x_i \tag{6}$$

Die Funktion *AVG(x,v)* berechnet einen Vektor aller Durchschnittswerte
$A_{x,i}$ mit $i \in 1..n \; und \; x \in \{(x-v), \dots, x\}$

$$AVG(x, v) = \begin{pmatrix} mittel(A_{x-v,1}, A_{x-v+1,1}, \dots, A_{x,1}) \\ mittel(A_{x-v,2}, A_{x-v+1,2}, \dots, A_{x,2}) \\ \dots \\ mittel(A_{x-v,n}, A_{x-v+1,n}, \dots, A_{x,n}) \end{pmatrix} \tag{7}$$

Die Funktion $var(x_1, \dots, x_n)$ berechnet die Varianz.

$$var(x_1, \dots, x_n) = \frac{1}{n} * \sum_{i=1}^{n} (x_i - \sigma)^2 \tag{8}$$

Wobei σ den Mittelwert der Population $x_1 \dots x_n$ darstellt.

Die Funktion *VARIANCE(x,v)* berechnet einen Vektor aller Varianzen aller
$A_{x,i}$ mit $i \in 1..n \; und \; x \in \{(x-v), \dots, x\}$]

$$VARIANCE(x,v) = \begin{pmatrix} v(A_{x-v,1}, A_{x-v+1,1}, \dots, A_{x,1}) \\ v(A_{x-v,2}, A_{x-v+1,2}, \dots, A_{x,2}) \\ \dots \\ v(A_{x-v,n}, A_{x-v+1,n}, \dots, A_{x,n}) \end{pmatrix} \quad (9)$$

Mit Hilfe dieser Funktionen lassen sich für jeden Zeitpunkt x ein Vektor $V_{x,v}$ mit einer der beiden Aggregationsmethoden *AVG()* oder *VARIANCE()* berechnen.

Ein Vektor $V_{x,v}(10, 5)$ beinhaltet also eine Aggregation der folgenden Werte:

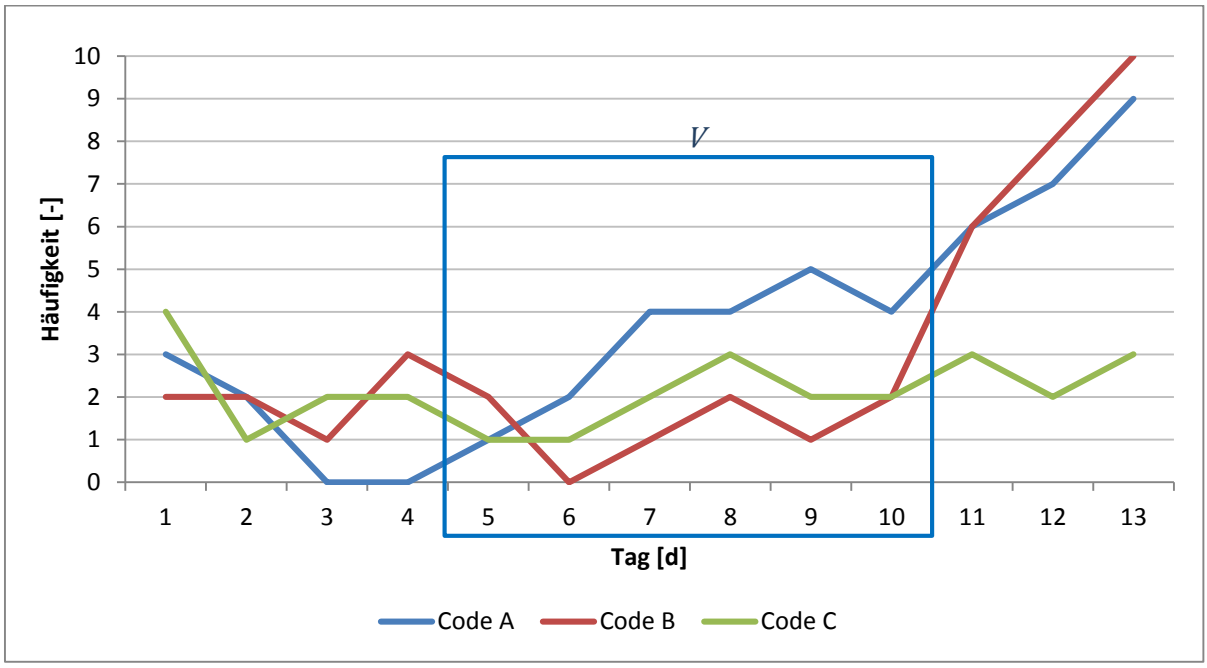


Abbildung 14 - Beispiel für Aggregationszeitraum von V mit v=5 und x=10

Bei der Erstellung dieser aggregierten Vektoren stellt sich folgendes Problem dar:
Durch Aufzeichnungslücken kann es dazu kommen, dass Werte in dem Bereich des Vektors fehlen. Es lässt sich aber nicht mit Sicherheit entscheiden, ob sie wirklich fehlen oder ob der Diagnosecode an dem Tag nicht auftrat.

Folgende Vorgehensweise wurde gewählt: Werte, die nicht existieren, werden von der Berechnung ausgeschlossen, also nicht als 0 mit einberechnet. Diese Vorgehensweise erleichtert die Berechnungen, welche auf einer SQL-Datenbank mit den dort vorhandenen Funktionen zur Bildung des Durchschnitts und der Varianz ausgeführt werden. Ein Kontrollversuch zeigte keinen signifikanten Einfluss auf das Klassifikationsergebnisses.

4.1.4. *Labeling*

Die selektierten Schadfälle werden als jeweils eigenes Klassifizierungsproblem betrachtet.

Aufgrund obiger Annahmen ergibt sich folgendes Lernschema für den Klassifizierer:
Für jeden Tag *x* und jede Lokomotive wird ein Attributsvektor $V_{x,v}$ – je nach gewählter Aggregationsmethode (*AVG* oder *VARIANCE*) – berechnet. Der Attributsvektor beinhaltet die aggregierten Werte vom Zeitpunkt *x* rückwirkend für einen Zeitraum von *v* vorangehenden Tagen.

Für alle Schadfälle gibt es einen Warnungszeitraum W. Dieser Warnungszeitraum endet mit dem Tag des Schadens (Schadzeitpunkt S) und hat die Länge w (siehe Abbildung 15).

Folgende Bedingungen gelten:

1. $W_{Start} = S\text{-}w$
2. $W_{Ende} = S$
3. $w >= 0$
4. $v >= 0$

Es gilt, die Werte für w und v so zu wählen, dass eine zuverlässige Erkennung und rechtzeitige Vorhersage getroffen werden kann. Geeignete Größen für diese Werte herauszufinden ist Bestandteil des Untersuchungsergebnisses.

Das *Labeling* der Daten wird folgendermaßen vorgenommen:

1. Für alle Schadzeitpunkte: Datensätze im Warnungszeitraum W werden als „warnung“ *gelabelt*.
2. Alle anderen Datensätze werden als „normal“ *gelabelt*.

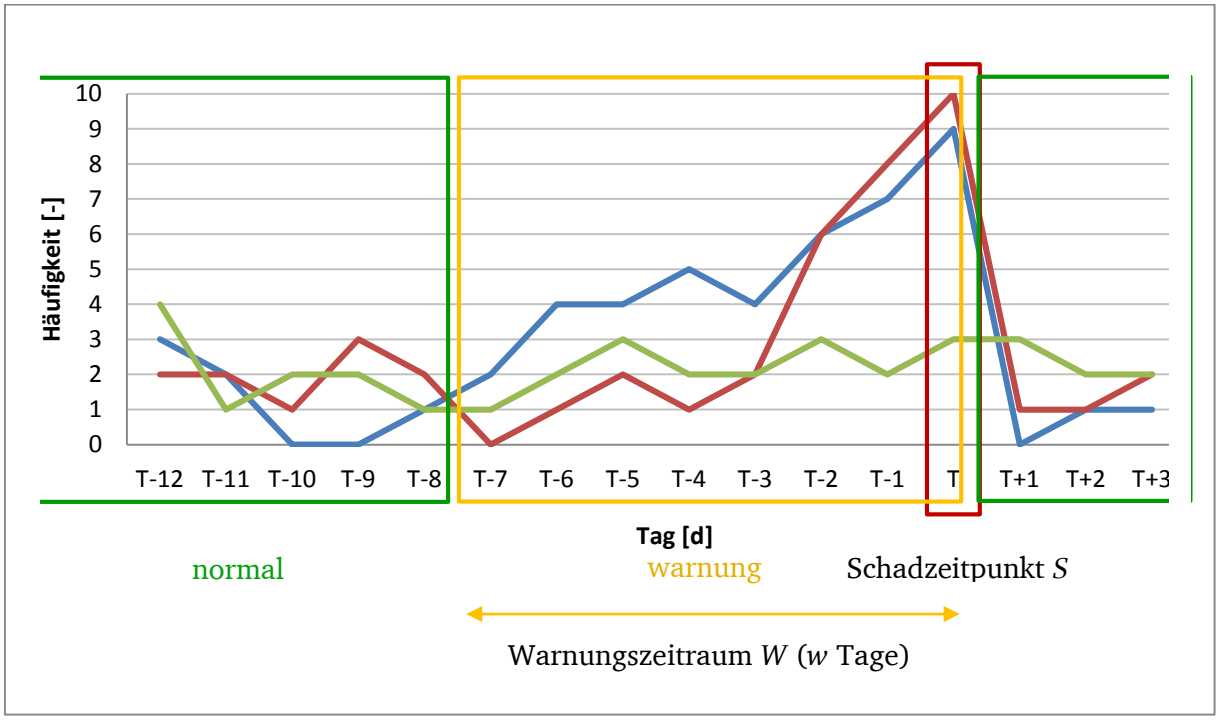


Abbildung 15 - Labeling-Zeiträume

Für jede Kombination von v und w wird ein separater Datensatz im ARFF-Format erzeugt.

Die Vektoren mit je 30 Attributen (plus Klassenattribut für das Training) werden im ARFF-Format wie folgt dargestellt (Beispiel):

```
1.25,1.1667,1.3333,1.3333,1.6667,1.6667,1,1,1,1,1,0,0,1,0,1,0,1,0,0,0,1,0,0,0,1.5,0,0,0,0,normal
1.25,1.1667,1.3333,1.3333,1.5,1.5,1,1,1,1,1.3333,0,0,1.3333,0,1,0,1,0,0,0,1,0,0,0,1.5,0,0,0,0,warnung
```

Die einzelnen Werte sind die errechneten Durchschnittswerte der Auftrittshäufigkeiten von 30 Tagen. Die erste Zeile ist ein „normal“ *gelabelter* Datensatz, die zweite ein mit „warnung“ *gelabelter*.

4.1.5. Nachfilterung der Diagnosedaten und Schadzeitpunkte

Eingangs wurde bereits erwähnt, dass die Datenbestände lückenhaft sind. Das bedeutet, dass für manche Lokomotiven in bestimmten Zeiträumen keine Daten aufgezeichnet wurden. Für einen speziellen Diagnosecode an einem bestimmten Tag ist es schwierig zu sagen, ob nichts aufgezeichnet wurde oder ob der Code an diesem Tag nicht auftrat. In beiden Fällen ist kein Eintrag für diesen Code vorhanden.
Daher gilt folgende Annahme: Wenn an einem Tag x zu keinem Code eine Aufzeichnung für die betrachtete Lokomotive vorliegt, so gilt dieser Tag als nicht aufgezeichnet. Nicht aufgezeichnete Tage werden als „invalide“ markiert und in der Auswertung nicht berücksichtigt. Wird dieser Schritt ausgelassen, werden diese Tage als Nullvektoren in den Lernvorgang eingefügt, verfälschen die Trainingsmenge und verschlechtern das Klassifikationsergebnis.

Das Problem überträgt sich auch auf die Schadzeitpunkte. Bei ca. 50 % - 75 % aller Schadzeitpunkte liegen in den vorangehenden Tagen keine Aufzeichnungen vor. Dies verfälscht die Erstellung des Attributsvektors erheblich, daher wird für jeden Schadzeitpunkt geprüft, ob im Bereich des Aggregationszeitraumes für diesen Tag mindestens 50 % der Tage aufgezeichnet wurden. Ist dies nicht der Fall, werden alle Tagesdaten in diesem Bereich als „invalide“ markiert und von der Klassifikation ausgeschlossen (siehe Abbildung 11, Abdeckungslücken Diagnosedaten). Dies stellt sicher, dass keine unvollständigen Daten zum Trainieren der „warnung“-Klasse verwendet werden.

Als Resultat der 50 % Grenze werden, je nach Größe des v-Wertes, unterschiedlich viele der ursprünglichen Schadzeitpunkte verworfen.

4.1.6. Training des Klassifizierers und Validierung

Mit den *gelabelten* Attributsvektoren $V_{x,v}$ wird nun der Klassifizierer trainiert. Es werden verschiedenartige Klassifizierer mit dem *WEKA*-Tool getestet und in der Auswertung berücksichtigt.

Die besten Werte für v und w werden experimentell ermittelt. Für initiale Richtwerte wird die Voruntersuchung (Anhang „Voruntersuchung Schadfälle 185er.pdf“) herangezogen.

Zur Verifizierung und Auswertung wird *5-Fold Cross-Validation* eingesetzt. Die *5-Fold* Variante wird genutzt, da das Training des *SMO*-Klassifizierers lange dauert und größere Werte für n daher schwierig zu berechnen sind.

Bei der Validierung muss beachtet werden, dass die meisten Datensätze mit „normal“ *gelabelt* sein werden und ein großes Ungleichgewicht der Klassen vorliegt.

Der *Accuracy*-Wert ist also wenig aussagekräftig, da ein Klassifizierer der alle Daten als „normal“ klassifiziert, *Accuracy* Werte über 98 % erzielt.
Stattdessen werden die *Precision* und *Recall*-Werte, sowie der *AUC*-Wert der Warnungs-Klasse betrachtet. Diese geben einen besseren Eindruck davon, wie präzise der Klassifizierer vorgeht (siehe auch (Salfner, Lenk, & Malek, 2010), S. 10ff.).

Um verschiedene Arten von Klassifizierern abzudecken, werden baum- und regelbasierte Verfahren sowie *Support-Vector*-Maschinen verwendet:

JRip
JRip ist die in Java implementierte Variante des *RIPPER*-Algorithmus. *RIPPER* ist ein regelbasierter Lernalgorithmus, der *Pruning* zur Generalisierung verwendet (Cohen, 1995).

WEKA-Parametrisierung von *JRip*:

- *Folds*: 3; Menge der Daten, die für das *Pruning* benutzt werden.
- *minNo*: 2; Das Minimalgewicht der Instanzen in einer Regel
- *Optimizations*: 2; Anzahl der durchgeführten Optimierungsläufe

J48
J48 ist die in Java implementierte Variante des C4.5-Algorithmus. C4.5 erzeugt einen Entscheidungsbaum und kann *Pruning* zur Generalisierung verwenden (Quinlan, Programs for Machine Learning, 1993).

WEKA-Parametrierung von *J48*:

- *confidenceFactor*: 0,25; Der Konfidenz-Faktor, der für *Pruning* verwendet wird.
- *minNumObj*: 2; Minimale Anzahl Instanzen je Blatt
- *subtreeRaising*: True: *Subtree-Raising* darf bei *Pruning*-Operationen verwendet werden

RandomForest
RandomForest basiert auf dem Prinzip des *RandomTree*. Beim *RandomTree*-Verfahren wird ein Entscheidungsbaum erzeugt, der an jedem Knoten *K* zufällige Attribute auswählt und anhand dieser entscheidet. Es wird kein *Pruning* vorgenommen. Bei *RandomForest* wird ein Wald aus mehreren *RandomTree* Instanzen erzeugt. Jeder dieser Bäume darf eine Klassifizierungs-Entscheidung treffen. Die am häufigsten gewählte wird als Resultat zurückgeliefert (Breiman, Random Forests, 2001).

WEKA-Parametrisierung von *RandomForest*:

- *maxDepth*: unlimited; Keine Maximale Tiefe ist vorgegeben
- *numFeatures*: unlimited: Keine Limitierung, wie viele Attribute an einem Knoten genutzt werden können
- *numTrees*: 10; Es werden 10 Bäume konstruiert

SMO (*Sequential minimal optimization*)
SMO ist ein *Support-Vector* Verfahren und verwendet sequenzielle Minimaloptimierung, um einen Klassifizierer zu trainieren (Platt, 1998)

WEKA-Parametrisierung von *SMO*:

- *C*: 300; Der Komplexitätsparameter
- *Tolerance*: 0,001; Der Toleranzparameter
- *Epsilon*: 1E-12; Der Parameter für Rundungsfehler
- *Kernel*: PolyKernel (Cache Size: 250007; E=5)

Grundsätzlich wird bei allen Verfahren mit den „Standardwerten“ aus *WEKA* gearbeitet. Alleinig bei *SMO* werden die Parameter modifiziert, da mit der Standardparametrisierung sehr niedrige Klassifikationsleistungen erzielt werden. *Support-Vector*-Maschinen erreichen oftmals

erst durch individuelle Parametrisierung ihren maximalen Wirkungsgrad. In diesem Fall ist aufgrund des enormen Ungleichgewichts der beiden Klassen eine Anpassung nötig.

4.1.7. Versuche

Im Folgenden werden die technischen Hintergründe zu den Versuchen erklärt und die Parametrisierung erläutert.

Zu den drei Schadbildern wird jeweils eine separate Auswertung vorgenommen. Bei dieser wird eine Vorhersage für das jeweilige Schadbild durchgeführt.

Abschließend werden zwei *Multiclass*-Versuche durchgeführt, die in einer Auswertung zwei bzw. drei Schadbilder vorhersagen.

Versuch 1.1 – Defektes ASG

Bei diesem Versuch werden Schäden des Typs „ASG instand setzen“ betrachtet.
Das ASG ist der Zentralrechner für die Stromrichter und nimmt die Aufteilung der Motorleistung der Fahrmotoren vor. Fällt das ASG aus, ist eine häufige Konsequenz, dass einer der beiden Stromrichter sich abschaltet. Dies bedeutet eine Halbierung der Antriebskraft. Je nach Strecke und Gewicht des Zuges ist eine Weiterfahrt nicht möglich und der Zug muss der Werkstatt zugeführt werden. Daraus resultieren erhebliche Verspätungen und Folgeprobleme. Das ASG kann nicht in allen Werkstätten repariert werden. Die Lokomotive muss in eine spezielle Werkstatt gebracht werden, was eine lange Überführungsdauer und Ausfallzeiten bedeutet.

Zu diesem Schadbild gibt es 142 aufgezeichnete Vorkommnisse. Nach Herausfiltern der Vorkommnisse mit vorangehenden Aufzeichnungslücken reduziert sich diese Zahl (Tabelle 4).
Die relevanten Diagnosecodes kommen aus den Systemen ASG1 und ASG2 (Antriebssteuergerät 1 und 2).

Es wurden folgende Kombinationen von v und w mit je zwei Aggregationsmethoden betrachtet:

Tabelle 4 - v und w-Werte für Versuch 1.1

v	w	Schäden (nach Filterung)	Methoden
30	10	40	AVG|VARIANCE
10	5	46	AVG|VARIANCE
5	3	49	AVG|VARIANCE

Versuch 1.2 – Defekt LZB

Bei diesem Versuch werden Schäden des Typs „LZB instand setzen“ betrachtet.
Das LZB-System (linienförmige Zugbeeinflussung) dient der Zugsicherung und ermöglicht unter anderem, dass Züge mit einer Geschwindigkeit von mehr als 160 km/h fahren dürfen. Bei Ausfall des LZB entstehen betriebliche Einschränkungen. Es muss mit dem PZB-System (punktförmige Zugbeeinflussung) weitergefahren werden. Fahrten ohne LZB sind auf vielen Strecken nicht zulässig, der Zug muss deswegen manuell vom Lokführer, der im telefonischen Kontakt mit dem Fahrdienstleiter steht, weitergefahren werden. Dieser Vorgang hat eine lang-

same Weiterfahrt und häufiges Stehenbleiben zufolge. Wie das ASG kann das LZB-System nur in bestimmten Werkstätten gewartet werden.

Es wurden 126 Vorkommnisse aufgezeichnete. Diese Zahl reduziert sich nach der Filterung der Vorkommnisse mit vorangehenden Aufzeichnungslücken (Tabelle 5).
Die relevanten Diagnosecodes kommen aus dem System ZSG (Zentrales Steuergerät).

Es wurden folgende Kombinationen von *v* und *w* mit je zwei Aggregationsmethoden betrachtet:

Tabelle 5 - v und w-Werte für Versuch 1.2

v	w	Schäden (nach Filterung)	Methoden
30	10	89	AVG|VARIANCE
10	5	87	AVG|VARIANCE
5	3	85	AVG|VARIANCE

Versuch 1.3 – Defekte LZB-Antenne

Bei diesem Versuch werden Schäden des Typs „LZB Empfangsantenne einstellen“ betrachtet. Bei einer defekten Empfangsantenne muss das LZB-System abgeschaltet werden. Die gleichen Konsequenzen wie in Versuch 1.2 treten auf.

Zu diesem Schadbild gibt es 93 Aufzeichnungen. Die Anzahl reduziert sich, nachdem die Aufzeichnungslücken herausgefiltert sind.
Die für diese Betrachtung relevanten Diagnosecodes kommen aus dem ZSG.

Es wurden folgende Kombinationen von *v* und *w* mit je zwei Aggregationsmethoden betrachtet:

Tabelle 6 - v und w-Werte für Versuch 1.3

v	w	Schäden (nach Filterung)	Methoden
30	10	20	AVG|VARIANCE
10	5	17	AVG|VARIANCE
5	3	17	AVG|VARIANCE

Versuch 1.4 – Defektes ASG und LZB

Bei diesem Versuch werden Schäden der Versuche 1.1 und 1.2 simultan betrachtet. Das Ziel ist festzustellen, ob eine Diskrimination in der Vorhersage zweier Schadbilder in einem Datensatz möglich ist.
Hierzu wurde ein gemischtes Attributset mit Diagnosecodes aus allen relevanten Systemen genutzt. Die Schadfälle basieren auf verschiedenen Systemen, die Diagnosecodes überschneiden sich daher nicht.

Das *Labeling* der Daten wurde verändert: Es wurden zwei Warnungsklassen „warnung1“ für die Schäden der Art „Antriebssteuergerät instand setzen“ und „warnung2“ für „LZB instand setzen“ genutzt.

Die relevanten Diagnosecodes kommen aus den Systemen ZSG, ASG1 und ASG2.

Es wird nur eine Kombination von v und w mit je zwei Aggregationsmethoden betrachtet. Die Kombination $v=30$ und $w=10$ hat sich in den vorherigen Versuchen als Parametrisierung mit der höchsten Erkennungsrate erwiesen, daher wird nur diese Kombination betrachtet.

Tabelle 7 - v und w-Werte für Versuch 1.4

v	w	Schäden (nach Filterung)	Methoden
30	10	40 (1.1) + 37 (1.2)	AVG

Versuch 1.5 – Defektes ASG, LZB und Antenne

Bei diesem Versuch werden Schäden der Versuche 1.1, 1.2 und 1.3 gleichzeitig betrachtet. Dies fügt dem Versuch 1.4 noch ein zusätzliches Schadbild hinzu, und soll zeigen, ob eine Diskrimination in der Vorhersage, insbesondere zwischen den Schäden aus den Versuchen 1.2 und 1.3, stattfindet.
Hierzu wurde ein gemischtes Attributset mit Attributen aus allen betroffenen Systemen genutzt. Die relevanten Diagnosecodes kommen aus den Systemen ZSG, ASG1 und ASG2.

Die Schadfälle von 1.2 und 1.3 basieren auf den gleichen Systemen, die relevanten Diagnosecodes überschneiden sich daher teilweise.

Das *Labeling* der Daten wurde verändert: Es wurden drei Warnungsklassen „warnung1“, „warnung2“ und „warnung3“ für die drei Schadtypen genutzt.

Es wird nur eine Kombination aus Vorhersagezeitraum und Aggregationszeitraum untersucht. Die Parametrisierung mit $w=10$ und $v=30$ lieferte in den vorigen Versuchen die besten Ergebnisse.

Tabelle 8 - v und w-Werte für Versuch 1.5

v	w	Schäden (nach Filterung)	Methode
30	10	40 (1.1) + 37 (1.2) + 20 (1.3)	AVG

4.2. Versuchsreihe 2

Dieser Abschnitt beschreibt die Voraussetzungen und Verarbeitungsschritte, die zur Betrachtung des Problems „Zentralschraubenbruch“ dienen.

4.2.1. Voraussetzungen

Für die Baureihe 189 liegen zum Zeitpunkt der Verfassung dieser Arbeit für den Zeitraum von April 2011 bis Oktober 2013 Diagnosedaten vor.

Bei den Diagnosedatensätzen der Siemens-Lokomotive werden analoge Messwerte mitgeliefert. Ähnlich der Diagnosedaten der Bombardier Lokomotive aus Versuchsreihe 1 ist es aber so, dass bei den verschiedenen Diagnosecodes jeweils unterschiedliche analoge Werte geliefert werden. In diesem Falle liegt das Augenmerk auf den Temperaturwerten der FM. Diese werden nur bei einigen Diagnosemeldungen erfasst und abgespeichert.

Aus einer initialen Menge von über 570.000 Diagnosemeldungen bleiben somit ungefähr 18.000 Meldungen, bei denen die FM-Temperaturen vorliegen. Im Verhältnis zum Zeitraum von 18 Monaten und einer Gesamtmenge von 59 Lokomotiven ist die Menge dieser Daten sehr gering.

Eine genauere Untersuchung der Datendichte kommt zu folgendem Ergebnis:

Im Durchschnitt sind je Lokomotive an 40 Tagen des 18-monatigen Zeitraumes Daten vorhanden.
Im Hinblick auf die Gesamtspanne an Tagen, bei denen für die jeweiligen Lokomotiven Daten vorhanden sind, sind diese 40 Tage im Durchschnitt zu 60 % befüllt.

Die Menge an Zentralschraubenschäden in diesem Zeitraum, zu denen Sensordatenaufzeichnungen der betroffenen Lokomotiven vorhanden sind, beläuft sich auf vier.

Für die betroffenen vier Lokomotiven sind die Abdeckungswerte wie folgt:

Tabelle 9 – Diagnosedaten-Vollständigkeit der schadhaften Lokomotiven

Baunr	Erster Eintrag	Letzter Eintrag	Aufgezeichnete Tage	Tage Mit Daten
189009	05.12.2012 15:52	01.04.2013 23:51	118	61
189049	02.10.2012 22:24	24.10.2013 05:43	388	106
189057	06.08.2013 18:10	03.09.2013 13:39	29	23
189086	28.06.2012 09:47	02.08.2012 12:55	36	26

Die genauen Werte für alle Lokomotiven befinden sich im Anhang „Versuchsreihe2 – Vollständigkeitsanalyse der Diagnosedaten“.

Die Schäden dieser vier Lokomotiven traten an verschiedenen FM auf. Bei der Lokomotive 189009 und 1890057 trat der Bruch an FM2, bei 189086 an FM1 und bei 189049 an FM3 auf.

4.2.2. Ziel der Untersuchung

Aufgrund der Tatsache, dass nur vier exemplarische Schadfälle für den Zentralschraubenbruch vorliegen, sind die Voraussetzungen für eine repräsentative Vorhersagetechnik anhand eines trainierten Klassifizierers schlecht.
Das Ziel dieser Untersuchung konzentriert sich darauf, Techniken und Methoden im Bereich des *Preprocessing* zu finden, die später, wenn eine bessere Datenlage vorhanden ist, angewandt werden können, um einen verlässliches Verfahren mit *Machine Learning* Methoden aufzubauen.
Am Ende der Versuchsreihe wird eine exemplarische Klassifizierung vorgenommen. Die Er-

gebnisse können jedoch nur einen vagen Ausblick auf das Potential des Verfahrens bieten und keinesfalls als Validierung dienen.

4.2.3. Preprocessing

Nach dem Import der Daten in eine *MySQL* Tabelle liegen sie wie in Tabelle 10 dargestellt vor. Alle Temperaturwerte sind stets als Fließkommazahlen dargestellt und wurden in Grad Celsius aufgezeichnet.

Tabelle 10 - Rohdatenformat Baureihe 189 (Ausschnitt)

baunr	fcode	kdatum	FM1 (°C)	FM2 (°C)	FM3 (°C)	FM4 (°C)	LFM1 (°C)	LFM2 (°C)	LFM3 (°C)	LFM4 (°C)
189081	1866	1302692280	16,4	15,8	14,7	14,7	16,5	15,4	14,5	14,7
189081	2678	1302692280	16,4	15,8	14,7	14,7	16,5	15,4	14,5	14,7
189081	1866	1302692280	16,4	15,8	14,7	14,7	16,5	15,4	14,5	14,7
189081	2678	1302692280	16,4	15,8	14,7	14,7	16,5	15,4	14,5	14,7
189050	2682	1305115620	25,6	25,9	25,6	25,3	26,3	26,3	25,3	24,8

Diese Form wird im Folgenden auch als Rohdaten bezeichnet.

Neben der eindeutigen Zugnummer „baunr“ und dem Diagnosecode „fcode“ sowie dem Auftrittszeitpunkt „kdatum“ als Unix Timestamp finden sich hier die Temperaturwerte der vier FM. Für jeden FM stehen die Werte für den Motor selbst (FM1-FM4) und für den Temperatursensor am Lager (LFM1-LFM4) bereit.

Beim Importvorgang stellt sich heraus, dass in den Daten einige Datensätze mit einem Zeitstempel von 1996 vorhanden sind. Diese werden nicht importiert. Auch Datensätze, bei denen alle vier FM-Temperaturwerte 0 °C angeben, werden nicht importiert.

Voranalyse der Daten

Zuerst werden die vorliegenden Diagnosedaten genauer betrachtet. Insbesondere ist interessant, in welchen Regionen sich die Temperaturen im Normalfall bewegen, was die Standardabweichungen sind und wie der Datenbestand beschaffen ist.

In Tabelle 11 ist dargestellt, wie hoch die Durchschnittstemperaturen der einzelnen FM sind. Hier wurde eine Mittelwert-Bildung über alle vorhandenen Datensätze, abzüglich derer die von defekten Lokomotiven stammen, vorgenommen.

Tabelle 11 – Durchschnittstemperaturen der FM – alle Datensätze

FM1	FM2	FM3	FM4
37,96 °C	37,60 °C	38,23 °C	37,76 °C

Bei diesen Durchschnittstemperaturen fällt auf, dass sie für Motoren, die mehrere tausend Kilowatt Leistung haben, sehr niedrig sind. Ein genauerer Blick in die Rohdaten (siehe Abbil-

dung 16) zeigt, dass viele aufgezeichnete Temperaturen unterhalb von 30 °C liegen. Wahrscheinlich wurden diese aufgezeichnet, als die Lokomotive noch nicht warm gelaufen war.

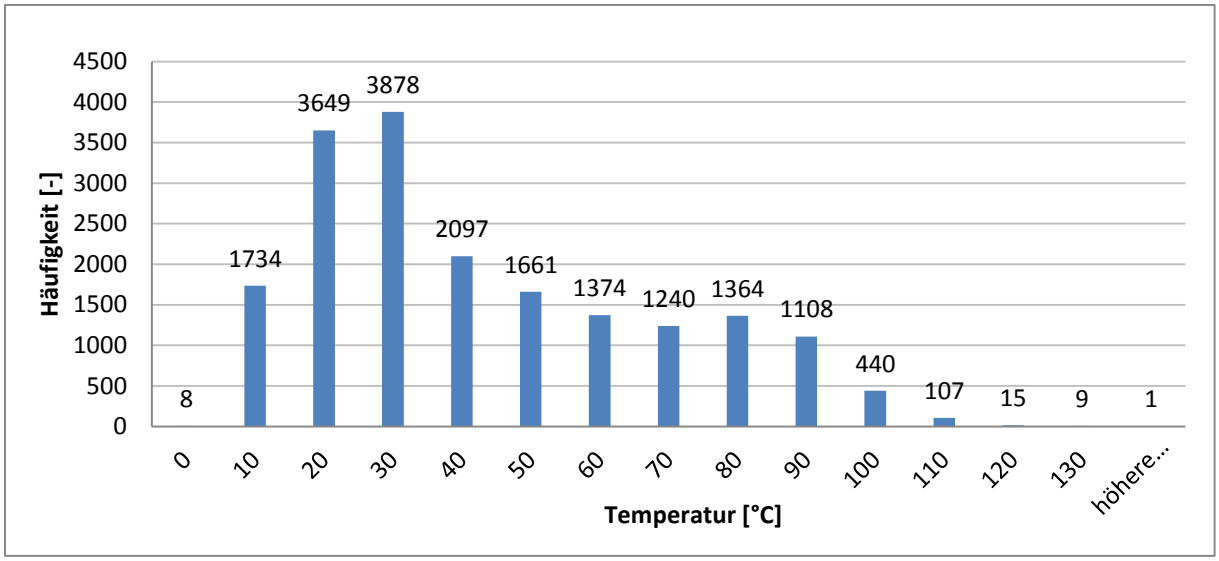


Abbildung 16 - Histogramm FM-Temperatur – alle Datensätze

Die Instanzen aus dem Kaltbetrieb zu betrachten ist nicht zielführend, da sie das Gesamtbild verzerren.
Daher werden für die Durchschnittstemperatur nur Aufzeichnungen, bei denen alle Motoren (FM1-FM4) über 40 °C liegen, betrachtet. Die 40 °C als Schwellwert wurden in Absprache mit einem Werkstattexperten gewählt. Alle darunter liegenden Werte gelten als nicht warmgelaufen und werden ignoriert. In Tabelle 12 zeigt sich das so errechnete Ergebnis.
Dieser Wert wurde als Kompromiss gewählt, um sowohl die Situation im Winter als auch im Sommer annäherungsweise adäquat abzubilden. Die Außentemperatur spielt bei der Motortemperatur natürlich eine Rolle. Wie sie sich exakt auf die Motortemperatur auswirkt ist aber unbekannt. Da keine Aufzeichnungen über die Außentemperaturen vorliegen, kann dies nicht genauer analysiert werden und diese grobe Trennung wird vorgenommen. Sollten zu einem späteren Zeitpunkt genauere Aufzeichnungen über die Fahrmotortemperaturen vorliegen, bietet es sich an diesen Schwellwert automatisiert zu berechnen oder eine andere Methode zur Ausfilterung der Kaltbetriebsdaten anzuwenden.

Tabelle 12 – Durchschnittstemperaturen der FM - Warmzustand

FM1	FM2	FM3	FM4
68,29 °C	67,52 °C	68,43 °C	67,89 °C

Die Werte unterscheiden sich deutlich von denen in Tabelle 11.
Als Resultat der Filterung bleiben von den ursprünglichen 17160 Datensätzen noch 5831 übrig. Die Datenquantität hat sich somit um etwa zwei Drittel verringert.

Als mittlere Durchschnittstemperatur ergibt sich für die Fahrmotoren:

$$avg(FM) = \frac{(FM1 + FM2 + FM3 + FM4)}{4} = 68{,}03\ °C \tag{10}$$

Um einen Eindruck zu bekommen, in welchen Regionen sich die Temperaturen bei einem warm gelaufenen Motor bewegen, wird die Standardabweichung zu den Temperaturwerten aus Tabelle 12 berechnet.

Tabelle 13 - Standardabweichung der Fahrmotortemperaturen

Std(FM1)	Std(FM2)	Std(FM3)	Std(FM4)
16,60 °C	16,32 °C	16,57 °C	16,54 °C

Die Standardabweichung der FM-Temperaturen liegt bei ungefähr 16,5 °C.

Zusätzlich zu den Temperaturen für die FM selbst werden noch die Temperaturen für die Lager errechnet.

Tabelle 14 - Durchschnittstemperaturen Motorlager

LFM1	LFM2	LFM3	LFM4
49,77 °C	50,19 °C	45,48 °C	49,48 °C

Hier lässt sich beobachten, dass die Lagertemperatur von Motorlager 3 um ca. 5 °C niedriger ist als die der übrigen.
Der Mittelwert der Durchschnittstemperaturen der Lager liegt ungefähr 20 °C unter dem Mittelwert der Fahrmotoren.

$$avg(LFM) = \frac{(LFM1 + LFM2 + LFM3 + LFM4)}{4} = 48{,}73\ °C \tag{11}$$

Bei den Standardabweichungen der Lagertemperaturen ergibt sich wieder ein homogenes Bild, mit einer leichten Abweichung bei LFM3.

Tabelle 15 - Standardabweichung der Lagertemperaturen

LFM1	LFM2	LFM3	LFM4
9,29 °C	9,29 °C	10,49 °C	9,13 °C

Bei FM3 und dem zugehörigen Lager ist eine Temperaturabweichung zu erkennen. Die Abweichung beträgt weniger als 10 % des Gesamtwertes, und ist geringer als die Standardabweichung. Daher wird der Temperaturunterschied dieses Motors für die weiteren Betrachtungen ignoriert.

Idealisierung der Datenmenge

Wie in Abschnitt 4.2.1 beschrieben liegen nur vier konkrete Fälle von Zentralschraubenbruch vor, die im Aufzeichnungszeitraum stattfanden. Die Tatsache, dass diese vier Fälle an verschiedenen FM auftraten, verschlechtert die Lernsituation zusätzlich.

Anhand der Ergebnisse von Tabelle 12 wird folgende Annahme getroffen:
Alle vier FM verhalten sich im Betrieb ähnlich, da sie der gleichen Umgebungstemperatur und der gleichen Belastung ausgesetzt sind.

Dass dies nicht vollkommen der Wahrheit entspricht wurde in Kapitel 4.2.2 anhand der Temperaturunterschiede der Motoren gezeigt. Da die Abweichung nicht signifikant ist, wird sie für die weiteren Betrachtungen ignoriert.

Um eine breitere Basis für die weitergehenden Schritte zu haben, wird eine Idealisierung der Fahrmotordaten vorgenommen (siehe Abbildung 13, Reduktion auf einen Fahrmotor). Von jedem Fahrzeug werden die Daten eines seiner FM behalten und die anderen verworfen.
Bei den Fahrzeugen mit Defekt werden die Daten des defekten Motors behalten, bei den übrigen Fahrzeugen die Daten eines zufällig gewählten Motors.
Damit erfolgt eine virtuelle Reduktion auf einen FM, für den vier Schadfälle vorliegen.

Dieses Vorgehen vereinfacht die Gesamtdatenmenge und erhöht die Anzahl nutzbarer Schadbeispiele. Auch wenn dies in einer realen Applikation nicht zielführend ist, da die Differenzierungsmöglichkeiten zwischen den einzelnen FM verloren gehen, ist die Anwendung in diesem Fall nützlich, um die Datenbasis zu verbreitern. Die Methoden und *Features*, die anhand eines virtuellen FM erarbeitet werden, lassen sich später auf vier separate Motoren anwenden. Dort müssen auch Temperaturunterschiede, wie hier bei FM3 beobachtet, berücksichtigt werden.

Um den Informationsverlust durch die Reduktion gering zu halten, werden zusätzlich zu den Temperaturwerten des einen virtuellen FM die Durchschnittstemperaturen aller vier Motoren und Lager beibehalten.

Folgende Werte sind nach diesem Schritt verfügbar:

- Temperatur des virtuellen FM
- Temperatur des virtuellen FM-Lagers
- Temperaturdurchschnitt aller vier FM
- Temperaturdurchschnitt aller vier FM-Lager

Saisonale Korrekturwerte

Die Umgebungstemperatur beeinflusst die Leistungsfähigkeit einer Lokomotive.
Seien es eingefrorene Gleise im Winter oder überhitzte Kompressoren im Sommer; die Jahreszeiten sorgen für Schwankungen und Besonderheiten, die sich auf alle Komponenten einer Lokomotive auswirken. Die FM sind keine Ausnahme.
Um die Daten für die spätere Verwendung mit *Machine Learning* Methoden vorzubereiten, werden saisonale Korrekturfaktoren auf Monatsbasis für die Temperaturen errechnet. Da keine Daten über die Außentemperatur vorliegen, werden die aufgezeichneten FM-Temperaturen verwendet, um einen relativen Korrekturfaktor zu errechnen (siehe Abbildung 13, Saisonale Korrekturfaktoren).

Mit Hilfe der absoluten Durchschnittstemperatur und den monatlichen Durchschnittstemperaturen werden Korrekturfaktoren ($f_{korr}(M)$) errechnet, um alle Temperaturen saisonal bereinigen zu können (Tabelle 16).

Sei $avg_{Monat}(M)$ die Durchschnittstemperatur des Monats *M*.
Sei $avg_{absolut}$ die Durchschnittstemperatur aller Datensätze.

Dann gilt:

$$f_{korr}(M) = \frac{avg_{Monat}(M)}{avg_{absolut}} \quad mit\ M \in \{1..12\} \tag{12}$$

Diese Korrekturfaktoren werden sowohl für die FM als auch für die Lagertemperaturen errechnet.
In der Tabelle ist erkennbar, dass die Temperaturen der FM selbst geringen Schwankungen unterlegen sind, wohingegen die Lagertemperatur größere Schwankungen aufweist.

Tabelle 16 - Durchschnittstemperaturen je Monat / Korrekturfaktoren

Monat	Avg(FM)	Avg(LFM)	Korrektur-Faktor FM	Korrektur-Faktor LFM	Anzahl
1	67,79 °C	46,21 °C	1,00	0,95	529
2	64,85 °C	45,30 °C	0,95	0,93	537
3	66,92 °C	46,86 °C	0,98	0,96	325
4	69,75 °C	51,29 °C	1,03	1,05	494
5	69,58 °C	49,42 °C	1,02	1,01	699
6	69,31 °C	50,12 °C	1,02	1,03	503
7	70,83 °C	52,95 °C	1,04	1,09	520
8	69,76 °C	51,39 °C	1,03	1,05	618
9	70,57 °C	51,91 °C	1,04	1,07	362
10	70,90 °C	47,56 °C	1,04	0,98	286
11	67,63 °C	43,90 °C	0,99	0,90	438
12	67,91 °C	44,00 °C	1,00	0,90	520

Mit diesen Korrekturfaktoren kann ein saisonal bereinigter Datensatz generiert werden.

Ausfilterung abweichender Daten

Um den Klassifizierer zu schärfen und ihm beim Training eine breitere Basis für das Erlernen der Lokomotiven ohne Schäden zu geben, wird eine weitere Filterung auf den idealisierten Daten vorgenommen (siehe Abbildung 13, Herausfiltern von Spitzen).

Vorgehensweise: Bei allen Instanzen wird geprüft, ob die FM-Temperatur innerhalb der 2-fachen Standardabweichung um die absolute Durchschnittstemperatur herum liegt. Instanzen, die nicht in diesen Grenzen liegen, werden als Ausreißer behandelt und herausgefiltert.

Die Standardabweichung der Fahrmotortemperaturen wird anhand der idealisierten Datenmenge neu ermittelt.

$Std(FM) = 16{,}786\ °C$

Es wird eine Bandpass-Filterung anhand der FM-Temperaturen vorgenommen.
In diesem Fall werden alle Instanzen mit einer FM-Temperatur außerhalb des Bereichs von 40 °C bis 101,6 °C verworfen. Die Untergrenze ergibt sich dadurch, dass nur Daten aus dem Warmbetrieb genutzt werden sollen.

4.2.4. *Feature* Generation

Als *Features* für die Klassifizierung werden die folgend beschriebenen Attribute verwendet. Zu den Temperaturwerten kommen einige Temperaturverhältnisse hinzu. Diese sollen es dem Klassifizierer leichter machen, intraindividuelle Temperaturunregelmäßigkeiten zu erkennen.

Für die Erstellung aussagekräftiger *Features* stehen die Werte aus Kapitel 4.2.3 zur Verfügung. Diese vier Werte werden direkt als *Feature* genutzt.

Zusätzlich werden noch die folgend beschriebenen *Features* errechnet.

1. Das Verhältnis der FM-Temperatur zur Lagertemperatur.

$$ratio(FM, LFM) = \frac{FM}{LFM} \tag{13}$$

   Aus den Betrachtungen der Histogramme lässt sich schließen, dass die FM-Temperatur um einen gewissen Faktor höher ist als die Temperatur des Lagers. Um Veränderungen in diesem Verhältnis bemerken zu können, wird es als *Feature* genutzt.

2. Das Verhältnis der FM-Temperatur zum Durchschnitt aller FM.

$$ratio(FM, allFM) = \frac{FM}{allFMavg} \tag{14}$$

   Sollte die Temperatur des FM von den übrigen abweichen, schlägt sich das in diesem Attribut nieder.

3. Das Verhältnis der Lager-Temperatur zum Durchschnitt aller Lager.

$$ratio(LFM, allLFM) = \frac{LFM}{allLFMavg} \tag{15}$$

   Unterschiede der Lagertemperatur im Vergleich zu den anderen Lagern werden durch dieses Attribut abgebildet.

4.2.5. Labeling

Bei Guo und Bai (Guo & Bai, 2011) wird eine Matrix von Zustandsvektoren konstruiert, um den Normalbetrieb einer Windturbine zu modellieren. Weicht ein Zustandsvektor von dieser Matrix um einen bestimmten Wert ab, wird dies als eine Warnung für einen bevorstehenden Defekt angesehen.
Ein ähnliches Prinzip wird in dieser Diplomarbeit mittels *Machine-Learning* Methoden implementiert. Anstatt eine Matrix mit ausgewählten Werten zu konstruieren, wird es dem Klassifizierer überlassen, diese Werte zu erlernen.
Es wird ein ähnliches *Labeling* wie in Versuchsreihe 1 (Kapitel 4.1.4) eingesetzt.

Ausgehend von der Hypothese H2 aus Kapitel 3.2.1 ergibt sich Abbildung 17, welche einen Schadverlauf an einem Fahrmotor darstellt.

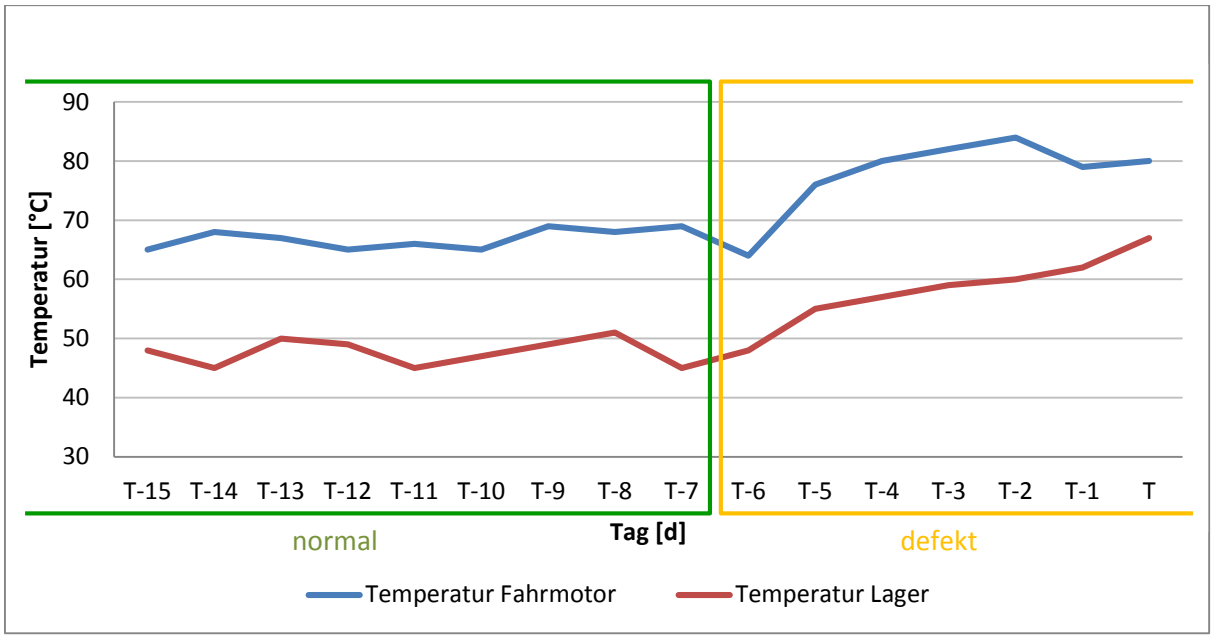


Abbildung 17 - Exemplarischer Schadfall, Temperaturverlauf

T bezeichnet den Tag des Zentralschraubenbruchs, *T-n* den *n*-ten Vortag.

Das *Labeling* wird folgendermaßen durchgeführt:

Für alle Lokomotiven, bei denen Defekte auftraten, für jeden Defekt *T*:

- Alle Datensätze an den Tagen *T-v bis T* werden für diese Lokomotive mit „defekt“ *gelabelt*.
- Alle übrigen Datensätze werden mit „normal“ *gelabelt*.

Für alle übrigen Lokomotiven:

- Alle Datensätze werden mit „normal“ *gelabelt*.

Der optimale Wert für *v* muss experimentell ermittelt werden. Dies ist Bestandteil des Untersuchungsergebnisses.

4.2.6. Versuchsaufbau

Bei dieser Versuchsreihe wird aufgezeigt, welche Effekte die Filterungs- und Optimierungsverfahren des *Preprocessings* haben. Als Basis wird die idealisierte Datenmenge verwendet. Sie beinhaltet bereits die initiale Ausfilterung aller Sensorwerte, die im Kaltbetrieb aufgezeichnet wurden.

Die *Preprocessing*-Schritte werden in drei Stufen aufeinander aufgebaut. Für jede dieser Stufen wird ein separates ARFF-Datenset erzeugt. Eine Evaluierung des Trainings- und Klassifizierungsergebnisses findet auf diesem Datenset statt.

Die Evaluierung erfolgt als *5-Fold Cross-Validation* mit verschiedenen Arten von Klassifizierern. Es werden baum- und regelbasierte Verfahren sowie *Support-Vector*-Maschinen verwendet. Eine genauere Beschreibung der Algorithmen und deren Parametrisierung befindet sich in Kapitel 4.1.6.

4.2.7. Versuche

Im Folgenden werden die durchgeführten Versuche beschrieben.

Aufgrund der ungleichen Verteilung der Klassen erfolgen zur Evaluation der Leistungsfähigkeit eine Betrachtung der *Precision*- und *Recall*-Werte der Minoritätenklasse, sowie des *AUC*-Wertes.
Die Gesamt-Trefferquote (*Accuracy*) dagegen ist nicht aussagekräftig, durch die Ungleichheit der Klassenverteilung liegt sie immer sehr nahe an 100 %.

Das Schadbild „Zentralschraubenbruch“ impliziert folgendes:

- Wenn es zum Schadfall kommt, sind die Reparaturkosten überdurchschnittlich hoch (Laut einer Experten-Schätzung ca. 200.000 - 300.000 €)
- Die Kosten für eine Überprüfung der Schraube sind im Verhältnis zu einer normalen Wartung sehr hoch, aber im Verhältnis zu den Reparaturkosten vernachlässigbar

Da keine genauen Daten über die Reparaturkosten und die Überprüfungskosten (inklusive der impliziten Kosten für Ausfallzeit, Werkstattpersonal, etc.) vorliegen, kann keine genaue Kosten-Nutzen Rechnung vorgenommen werden.

Es ist empfehlenswert, die Lokomotiven nicht unnötig oft zu einer Überprüfung zur Werkstatt zu schicken. Aufgrund der enormen Mehrkosten im Schadfall ist eine unnötige Überprüfung allerdings weniger kostenintensiv als ein aufgetretener Schaden.
Für die Auswertung bedeutet dies, dass dem *Recall*-Wert die größte Bedeutung beigemessen wird. Er besagt, wie hoch der Anteil der wiedererkannten Instanzen liegt. Die *Precision* sollte gleichzeitig hohe Werte aufweisen, damit die benötigte Differenzierung gewährleistet ist.

Versuch 2.1 – Idealisierte Datenmenge

Bei der idealisierten Datenmenge sind die Aufzeichnungen des Kaltbetriebs (Temperaturen unter 40 °C) herausgefiltert und auf einen virtuellen FM – im Gegensatz zu den realen vier FM – reduziert.

Es werden drei Vorhersagezeiträume getestet: 5, 10 und 20 Tage. Die folgenden ARFF-Datensätze resultieren daraus (Tabelle 17).

Tabelle 17 - ARFF-Dateien Versuch 2.1

Datei	Anzahl Instanzen gesamt	Anzahl Instanzen mit Label „normal“	Anzahl Instanzen mit Label „defekt“	Verhältnis „defekt“/“normal
Job0010-zsb_5.arff	6585	6450	135	0,02
Job0010-zsb_10.arff	6585	6386	199	0,031
Job0010-zsb_20.arff	6585	6288	297	0,047

Versuch 2.2 – Idealisierte Datenmenge (saisonbereinigt)

Bei dieser Datenmenge sind die Korrekturfaktoren für die saisonale Bereinigung mit eingerechnet.

Es werden drei Vorhersagezeiträume getestet: 5, 10 und 20 Tage. Die in Tabelle 18 gezeigten-Datensätze resultieren daraus.

Tabelle 18 - ARFF-Dateien Versuch 2.2

Datei	Anzahl Instanzen gesamt	Anzahl Instanzen mit Label „normal“	Anzahl Instanzen mit Label „defekt“	Verhältnis „defekt“/“normal“
Job0011-zsb_deseas_5.arff	6585	6450	135	0,021
Job0011-zsb_deseas_10.arff	6585	6386	199	0,031
Job0011-zsb_deseas_20.arff	6585	6288	297	0,047

Versuch 2.3 – Idealisierte Datenmenge (saisonbereinigt + *Outlier* entfernt)

Bei dieser Datenmenge sind die Korrekturfaktoren für die saisonale Bereinigung mit eingerechnet und *Outlier* wurden entfernt.

Das Training des Klassifizierers wird auf dieser Datenmenge ausgeführt, die Evaluation auf den Datensätzen von Versuch 2.2. Die Trainings-*Folds* und die Test-*Folds* werden so generiert, dass eine *5-Fold Cross-Validation* möglich ist.

Es werden die Vorhersagezeiträume 5, 10 und 20 Tage getestet. Die resultierenden ARFF-Datensätze sind in Tabelle 19 aufgelistet.

Tabelle 19 - ARFF-Dateien Versuch 2.3

Datei	Anzahl Instanzen gesamt	Anzahl Instanzen mit Label „normal“	Anzahl Instanzen mit Label „defekt“	Verhältnis „defekt“/“normal“
Job0012-zsb_deseas_filter_5.arff	6455	6320	135	0,021
Job0012-zsb_deaseas_filter_10.arff	6455	6256	199	0,031
Job0012-zsb_deseas_filter_20.arff	6455	6158	297	0,048

Versuch 2.4 – Rohdaten mit Multi-*labeling*

Als letzter Versuch dieser Reihe wird das *Labeling* direkt mit den Rohdaten ausgeführt. In den Rohdaten sind die Temperaturwerte für alle vier FM und zugehörigen Lager vorhanden.
Die Idealisierung, Berücksichtigung saisonaler Einflüsse und die *Outlier*-Filterung werden nicht angewandt.
Es werden nur Daten aus dem Warmbetrieb verwendet.

Die *Feature*-Generierung wird wie in Kapitel 4.2.4 beschrieben durchgeführt. Es werden *Features* für alle vier FM generiert. Dies erhöht die Anzahl an Attributen auf 22.

Die Klassennamen der „defekt“-Klassen werden angepasst: „fm1“, „fm2“ und „fm3“ zeigen an, welcher der Motoren den Defekt hatte. Da es zweimal Schäden an Fahrmotor 1 gab, ist die Anzahl dieser Instanzen höher als die der übrigen (siehe Tabelle 20).

Es wird eine Vorhersagedauer von 10 Tagen getestet.

Tabelle 20 - ARFF-Dateien Versuch 2.4

Datei	Anzahl Instanzen gesamt	Anzahl mit Label „fm1“	Anzahl mit Label „fm2“	Anzahl mit Label „fm3“
Job0013-zsb_raw10.arff	6585	140	34	25

5. Ergebnisse

Dieses Kapitel beinhaltet die Ergebnisse und Evaluation der beiden Versuchsreihen.

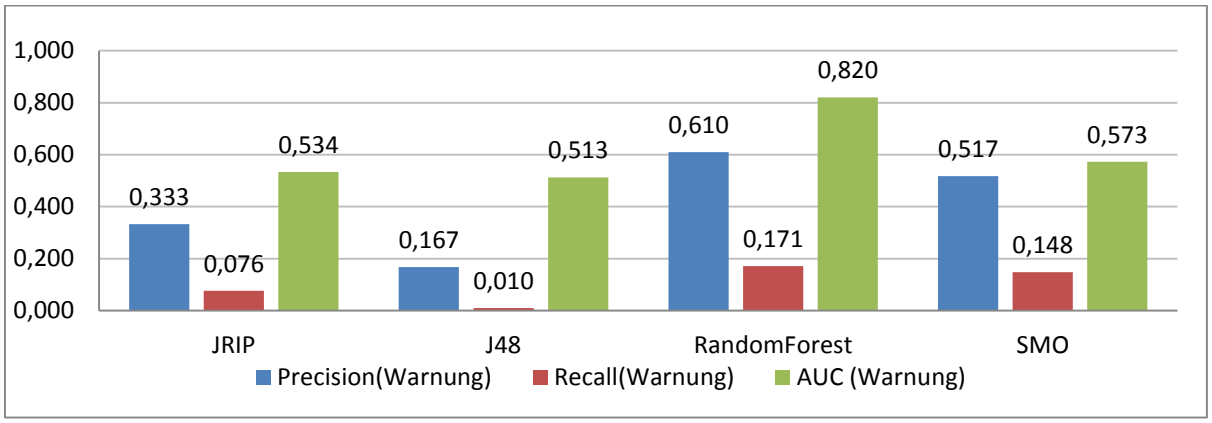
5.1. Ergebnisse Versuchsreihe 1

Im Folgenden werden für die Versuche der Versuchsreihe 1 die Ergebnisse dargestellt. Sie sind nach der Aggregationsmethode (*AVG* oder *VARIANCE*) unterteilt.

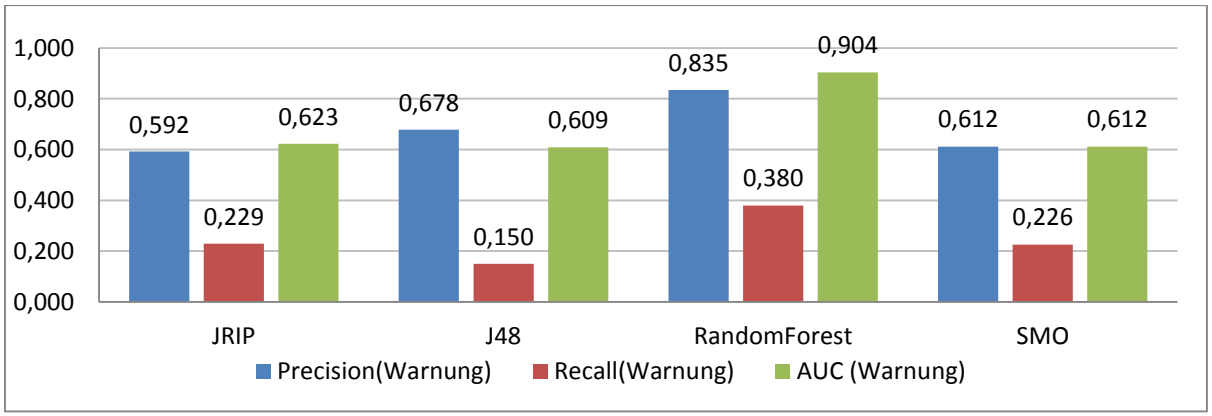
5.1.1. Versuch 1.1 – Methode AVG

In Abbildung 18 werden die Ergebnisse der AVG-Methode dargestellt und danach erläutert.

Teil-Abbildung 18 a - *v*=5; *w*=3; Methode: AVG



Teil-Abbildung 18 b - *v*=10; *w*=5; Methode: AVG



Teil-Abbildung 18 c - v=30; w=10; Methode: AVG

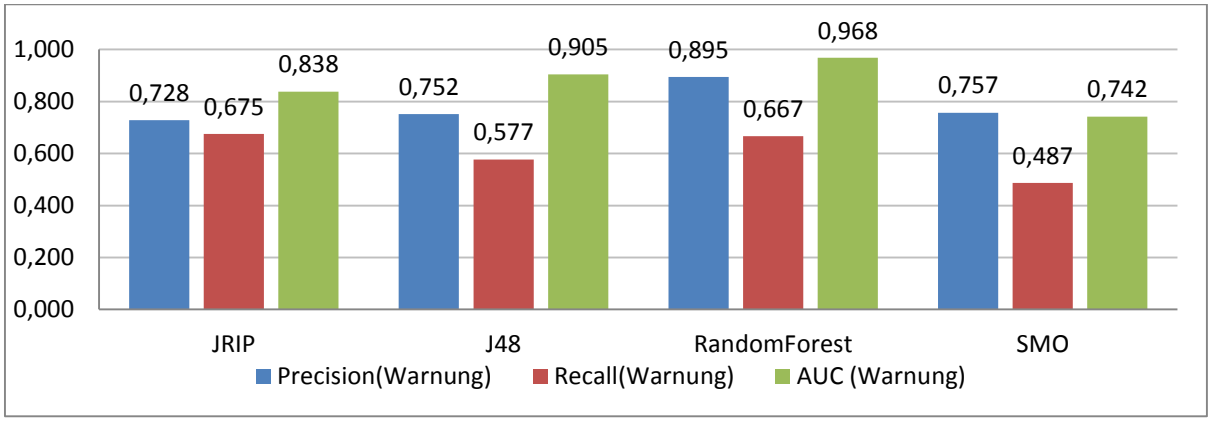


Abbildung 18 - Ergebnisse Versuch 1.1 - Methode AVG

In den Ergebnissen (Abbildung 18) zeigt sich, dass der größte gewählte Zeitraum (v=30, w=10, Teil-Abbildung 18 c) das beste Ergebnis bei allen Algorithmen liefert. *RandomForest* hat hier mit 98,5 % die höchste *Precision*, eine *AUC* von 0,698 und einen *Recall*-Wert von 66,7 %.

JRip und *J48* liefern ebenfalls gute Ergebnisse. *JRip* kann den insgesamt höchsten *Recall*-Wert von 67,5 % erzielen, die Präzision liegt dabei mit 72,8 % unter der von *RandomForest*. Bei *J48* ist der *Recall*-Wert leicht unter denen der Konkurrenten (57,7 %), die *Precision* (75,2 %) und die *AUC* (0,905) liegen zwischen *JRip* und *RandomForest*.

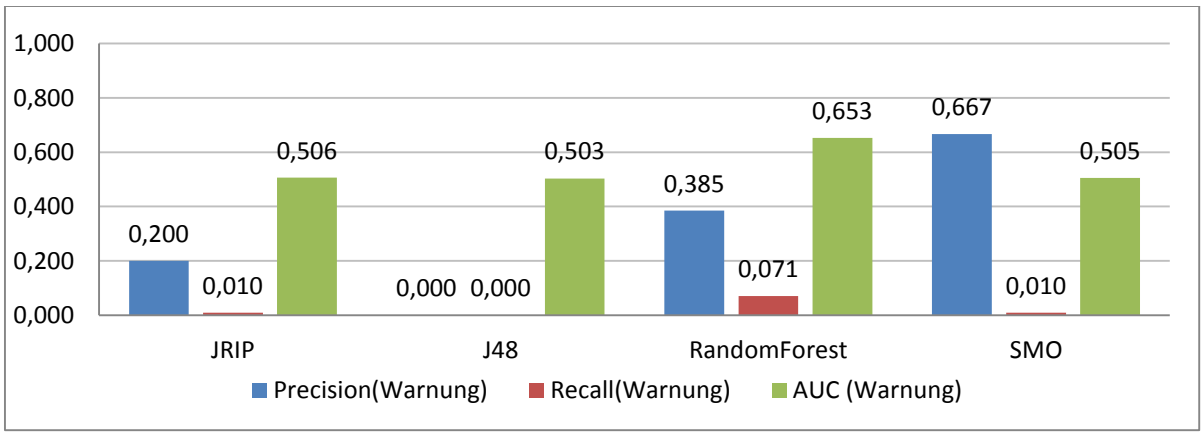
Die *Support-Vector* Maschine *SMO* kann mit der AVG-Methode trotz akzeptabler *Precision* von 75,7 % nur mit einem *Recall* von 48,7 % aufwarten und liegt in allen drei Bereichen zurück.

In den kürzeren Zeiträumen verschlechtern sich alle Algorithmen erheblich im *Recall*-Wert. Selbst *RandomForest* kann bei einer Vorhersagedauer von 5 Tagen (Teil-Abbildung 18 a) nur noch 17,1 % Recall leisten. Die *Precision* und *AUC* sinken ebenfalls, allerdings in kleineren Schritten. Bei 5 Tagen Vorhersagedauer ist der *Recall* von *J48*, *JRip* und *SMO* so niedrig, dass die *AUC*-Werte auf nahezu 0,5 sinken. Es ist keine Erkennung mehr vorhanden.

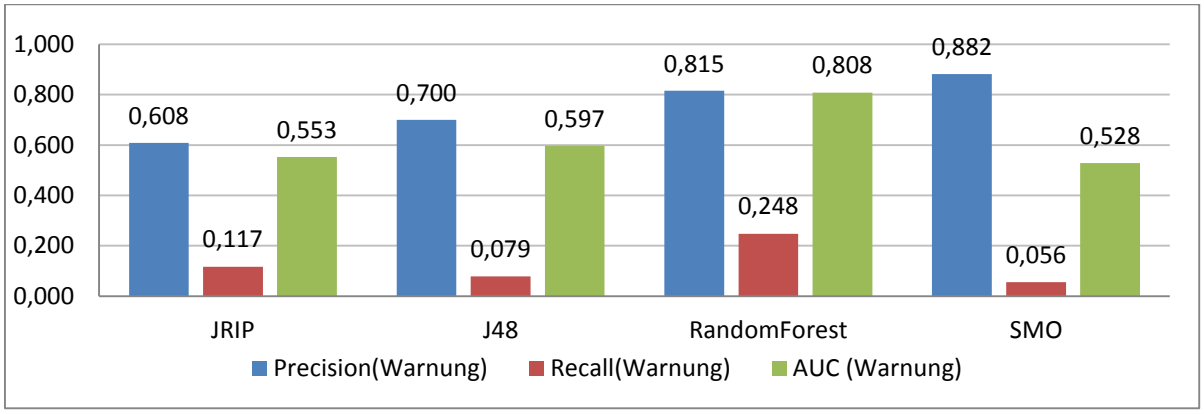
5.1.2. Versuch 1.1 – Methode VARIANCE

In Abbildung 19 werden die Ergebnisse der VARIANCE-Methode dargestellt.

Teil-Abbildung 19 a - v=5; w=3; Methode: VARIANCE



Teil-Abbildung 19 b - *v*=10; *w*=5; Methode: VARIANCE



Teil-Abbildung 19 c - *v*=30; *w*=10; Methode: VARIANCE

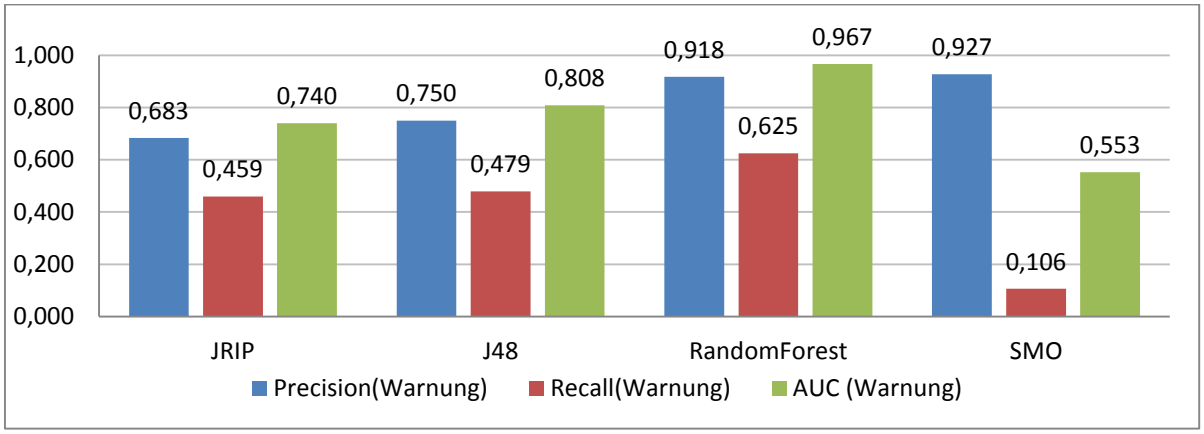


Abbildung 19 - Ergebnisse Versuch 1.1 – Methode VARIANCE

Bei der VARIANCE-Methode liefert ebenfalls der größte gewählte Zeitraum ($v=30$, $w=10$, Teil-Abbildung 19 c) das beste Ergebnis bei allen Algorithmen.
SMO hat die höchste *Precision* (92,7 %), aber die niedrigsten Werte bei *Recall* (10,6 %) und *AUC* (0,553).
Die insgesamt besten Ergebnisse erreicht *RandomForest*. Bei der *Precision* ist nur wenig Abfall gegenüber der von *SMO* zu beobachten (91,8 % zu 92,7 %), *Recall* (62,5 %) und *AUC* (0,967) stellen die Höchstwerte im ganzen Feld dar.

Die anderen Klassifizierer liefern erneut gute Ergebnisse im Zeitraum $v=30$; $w=10$.
J48 liegt hierbei vor *JRip*, mit einer *AUC* von 0,808 und mittleren Werten für *Recall* (47,9 %) und *Precision* (75 %). *JRip* kann eine *AUC* von 0,74, *Recall* 45,9 % und *Precision* von 68,3 % erreichen.

Mit kürzerer Dauer der Vorhersage- und Aggregationszeiträume verschlechtern sich bei allen Klassifizierern die Ergebnisse. Der Abfall des *Recall*-Wertes ist schon bei einer Vorhersagedauer von 5 Tagen (Teil-Abbildung 19 b) für alle Algorithmen erheblich, bei der Vorhersagedauer von 3 Tagen (Teil-Abbildung 19 a) liegt er bei allen Klassifizierern bei 0 oder nur wenig darüber. Der *AUC* Wert von *J48*, *JRip* und *SMO* nahe 0,5 verdeutlicht, dass diese keine Erkennung mehr durchführen können.

5.1.3. Versuch 1.1 – Vergleich der Methoden

Die beiden Methoden AVG und VARIANCE liefern mit den größten gewählten Zeiträumen die besten Ergebnisse.

Mit der AVG-Methode sind – unabhängig vom Klassifizierer – die *Recall*-Werte höher als mit der VARIANCE-Methode.

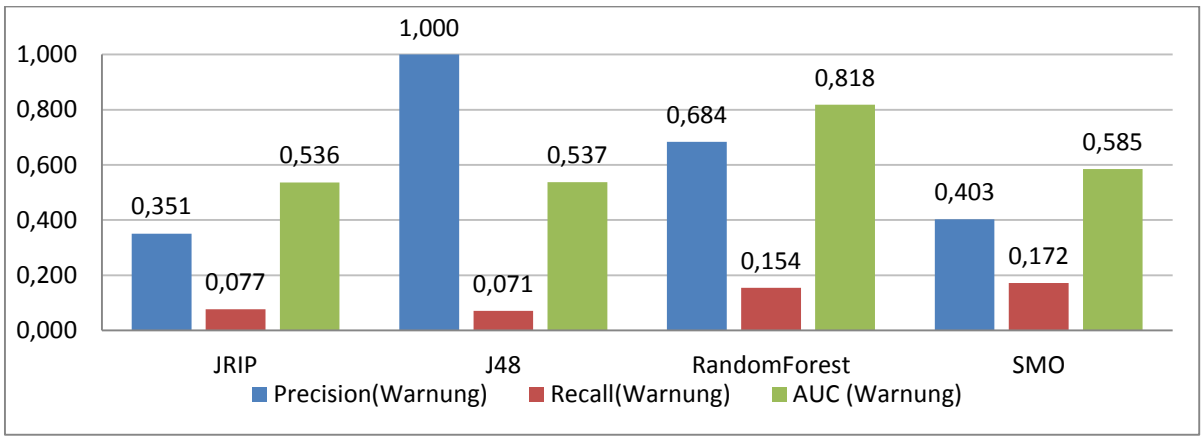
Im Gegensatz dazu ist die *Precision* bei allen Klassifizierern außer *JRip* mit der VARIANCE-Methode ein wenig höher. Bei den kleineren Zeiträumen stellt sich die AVG-Methode als die leistungsfähigere heraus, die VARIANCE-Methode liefert dort zu geringe *Recall*-Werte.

Auffällig ist, dass *SMO* mit dem VARIANCE-Verfahren durchgängig sehr schlechte *Recall*-Ergebnisse liefert. Die Vermutung liegt nahe, dass der *SMO*-Algorithmus mit den Varianzen als Attributen nicht zurechtkommt, da die resultierenden Attributwerte klein sind und nahe beieinander liegen.

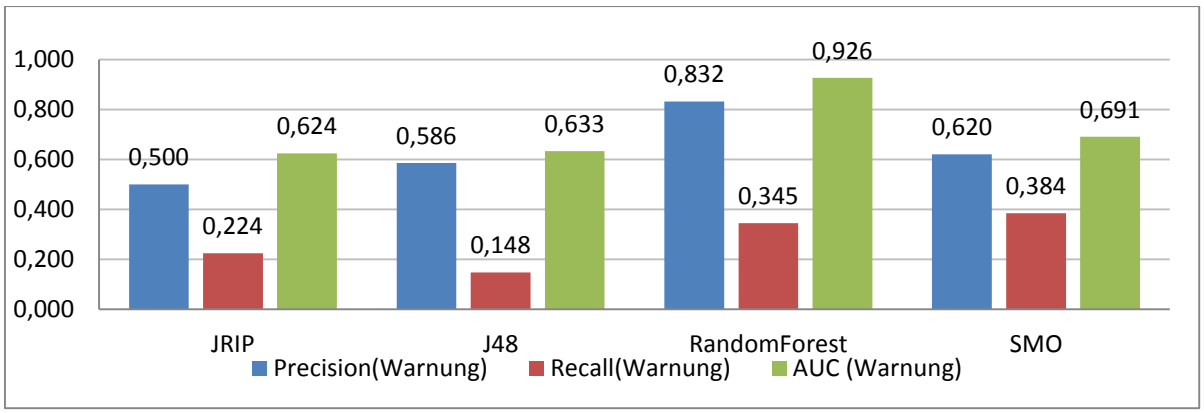
5.1.4. Versuch 1.2 – Methode AVG

In diesem Abschnitt werden die Ergebnisse der AVG-Methode dargestellt (Abbildung 20) und erläutert.

Teil-Abbildung 20 a - *v=5; w=3*; Methode: AVG



Teil-Abbildung 20 b - *v=10; w=5*; Methode: AVG



Teil-Abbildung 20 c - *v=30; w=10*; Methode: AVG

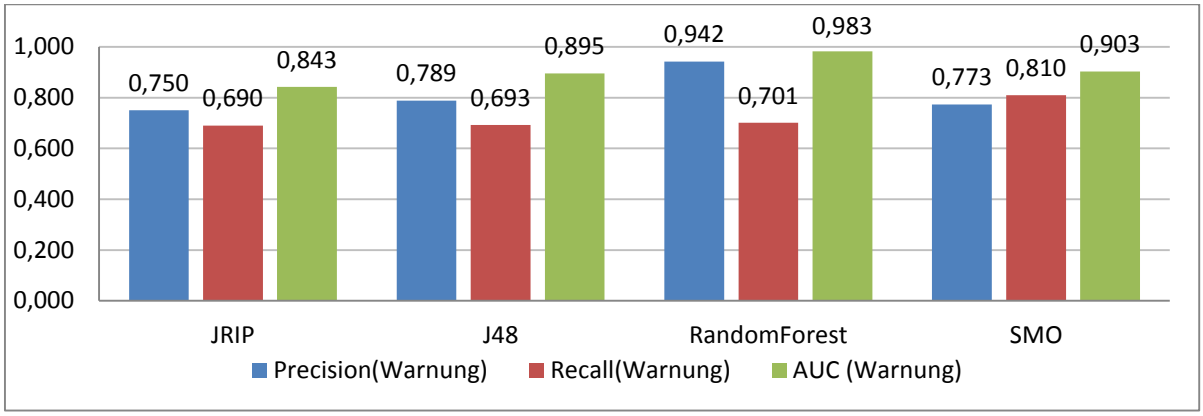


Abbildung 20 - Versuch 1.2 - Methode: AVG

Die besten Leistungen ergeben sich mit dem längsten Vorhersage- und Aggregationszeitraum (Teil-Abbildung 20 c).

SMO kann in diesem Zeitraum den höchsten *Recall*-Wert von 81 % bei einer *AUC* von 0,903 und *Precision* von 77,3 % verzeichnen. *RandomForest* erreicht einen *Recall*-Wert von 70 % und kann mit der sehr hohen *Precision* (94,2 %) und *AUC* (0,983) insgesamt sehr gute Ergebnisse vorweisen. Die Ergebnisse von *JRip* und *J48* ähneln denen von *RandomForest*. Bei *JRip* beträgt die *AUC* 0,843, der *Recall* 69 % und die *Precision* 75 %. *J48* hat eine *AUC* von 0,895, einen *Recall* von 69,3 % und eine *Precision* von 78,9 %.

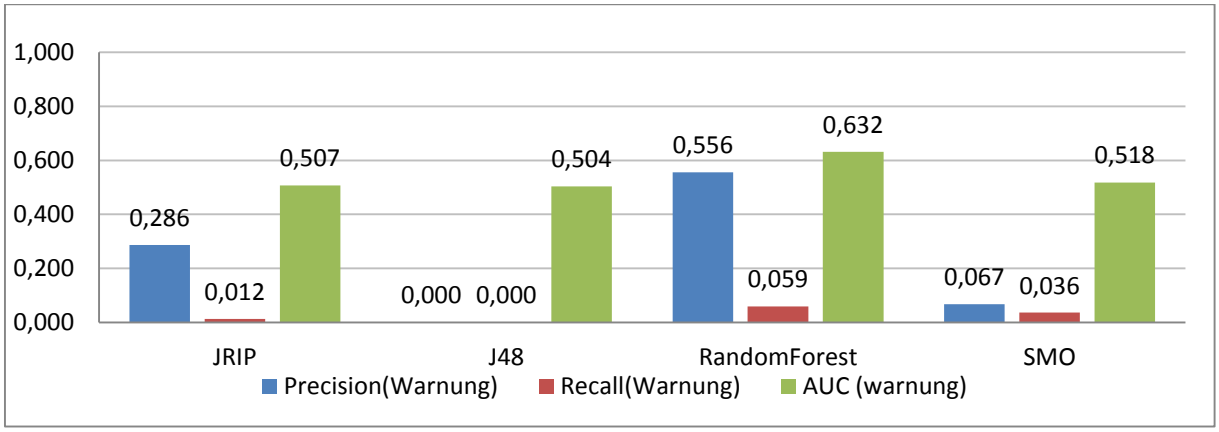
SMO kann den besten *Recall*-Wert erreichen, was eine wichtige Qualität ist, wenn es darum geht, möglichst viele Schäden rechtzeitig zu erkennen, auch wenn dies mit einer höheren *False-Positive* Rate einhergeht.

Bei den kürzeren Vorhersage- und Aggregationszeiträumen (Teil-Abbildung 20 a und b) sinken die Werte bei allen Klassifizierern ab, insbesondere der *Recall*-Wert verschlechtert sich enorm, je kürzer der Zeitraum ist. Für *J48*, *JRip* und *SMO* sinkt er (Teil-Abbildung 20 a) so weit ab, dass kaum noch Erkennung vorhanden ist.

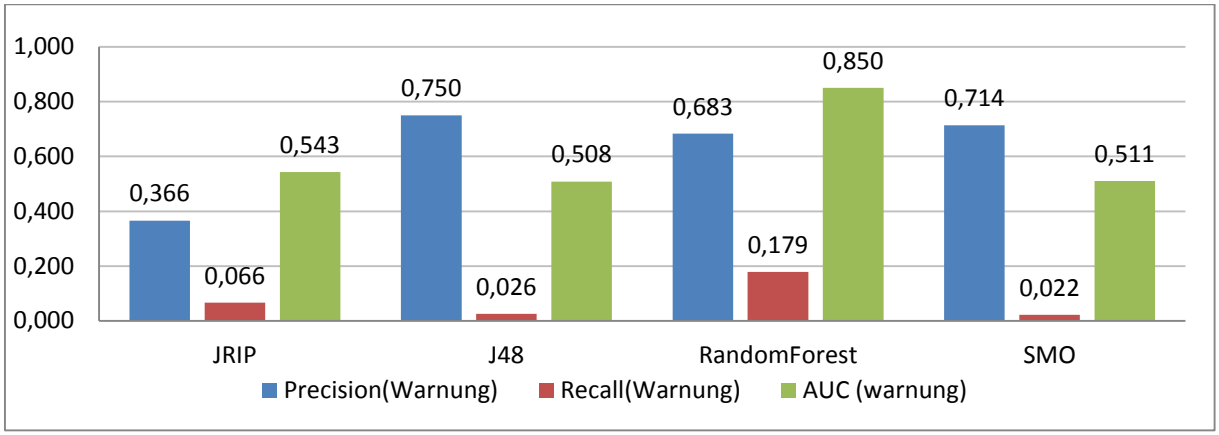
5.1.5. Versuch 1.2 – Methode VARIANCE

In Abbildung 21 werden die Ergebnisse der VARIANCE-Methode dargestellt.

Teil-Abbildung 21 a - *v=5; w=3*; Methode: VARIANCE



Teil-Abbildung 21 b - *v=10; w=5*; Methode: VARIANCE



Teil-Abbildung 21 c - *v=30; w=10*; Methode: VARIANCE

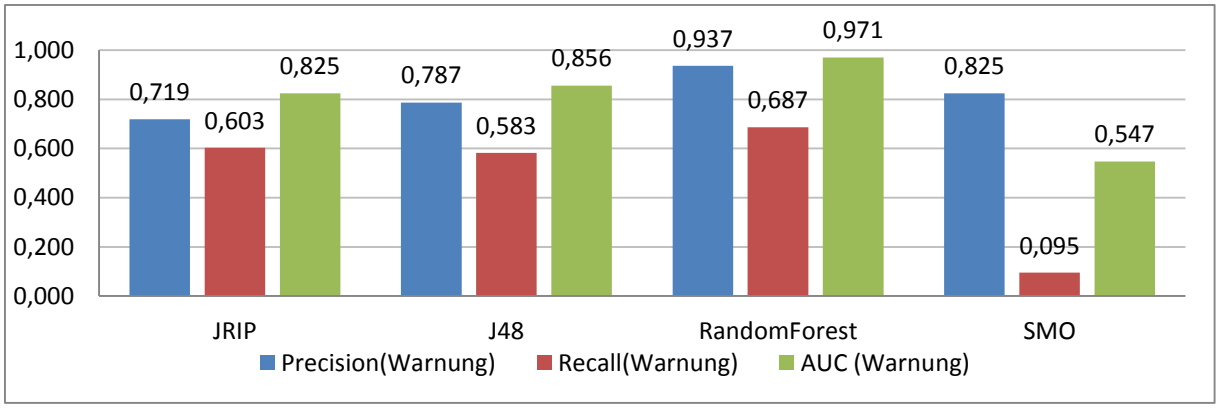


Abbildung 21 - Versuch 1.2 - Methode: VARIANCE

Die besten Leistungen ergeben sich beim längsten Vorhersage und Aggregationszeitraum (Teil-Abbildung 21 c).

RandomForest erreicht in diesem Zeitraum einen *Recall*-Wert von 68,7 % und kann mit der höchsten *Precision* (93,7 %) und *AUC* (0,971) insgesamt die besten Ergebnisse vorweisen.

SMO zeigt mit der VARIANCE-Methode nur sehr niedrige *Recall*-Leistungen (9,5 %) und somit das schlechteste Ergebnis der vier Klassifizierer.

Bei *JRip* beträgt die *AUC* 0,825, der *Recall* 60,3 % und die *Precision* 71,9 %. Er liegt damit im Mittelfeld.
J48 bewegt sich im gleichen Bereich mit einer *AUC* von 0,856, *Recall* 58,3 % und *Precision* 85,6 %.

Bei den kürzeren Vorhersage- und Aggregationszeiträumen verschlechtern sich alle Klassifizierer im *Recall*-Wert enorm. Schon im mittleren Zeitraum (Teil-Abbildung 21 b) liegen *J48*, *JRip* und *SMO* sehr niedrig, es erfolgt nahezu keine Erkennung mehr. *RandomForest* liegt hier noch etwas höher (17,9 % *Recall*). Im kurzen Zeitraum (Teil-Abbildung 21 a) kann auch *RandomForest* keine gute Erkennung mehr liefern.

5.1.6. Versuch 1.2 – Vergleich der Methoden

Die VARIANCE-Methode liefert bei diesem Versuch durchgängig schlechtere Ergebnisse als die AVG-Methode.

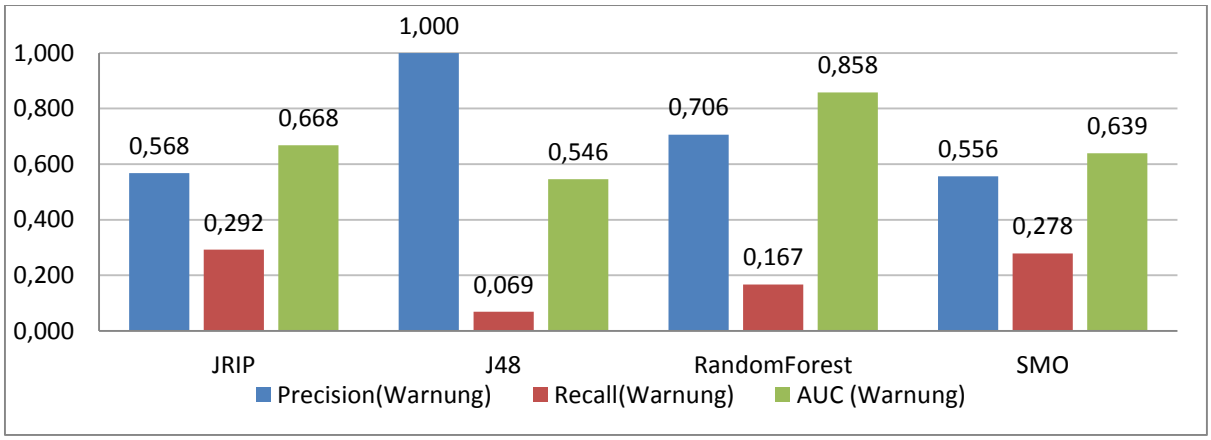
Insbesondere der extreme Abfall der Leistungen bezüglich des *Recall*-Wertes bei kleineren Vorhersage und Aggregationsdauern ist erwähnenswert. Hier kann die AVG-Methode deutlich bessere Werte erzielen.

Die AVG-Methode besitzt insgesamt für diesen Versuch die besseren Vorhersageleistungen, unabhängig von Klassifizierer und Vorhersagedauer. Die besten Ergebnisse werden im längsten Aggregations- und Vorhersagezeitraum erreicht.

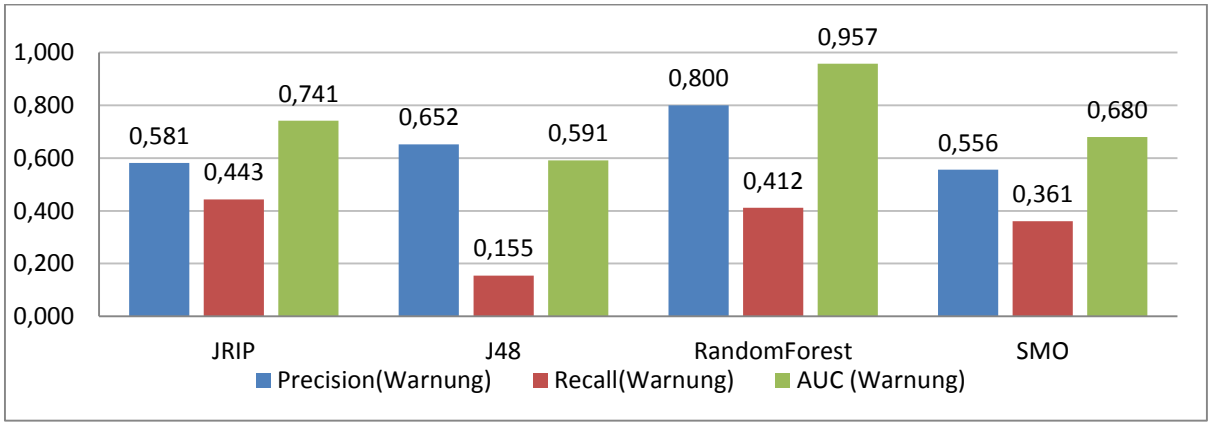
5.1.7. Versuch 1.3 – Methode AVG

In diesem Abschnitt werden die Ergebnisse der AVG-Methode dargestellt (Abbildung 22) und erläutert.

Teil-Abbildung 22 a - *v=5; w=3*; Methode: AVG



Teil-Abbildung 22 b - *v=10; w=5*; Methode: AVG



Teil-Abbildung 22 c - *v=30; w=10*; Methode: AVG

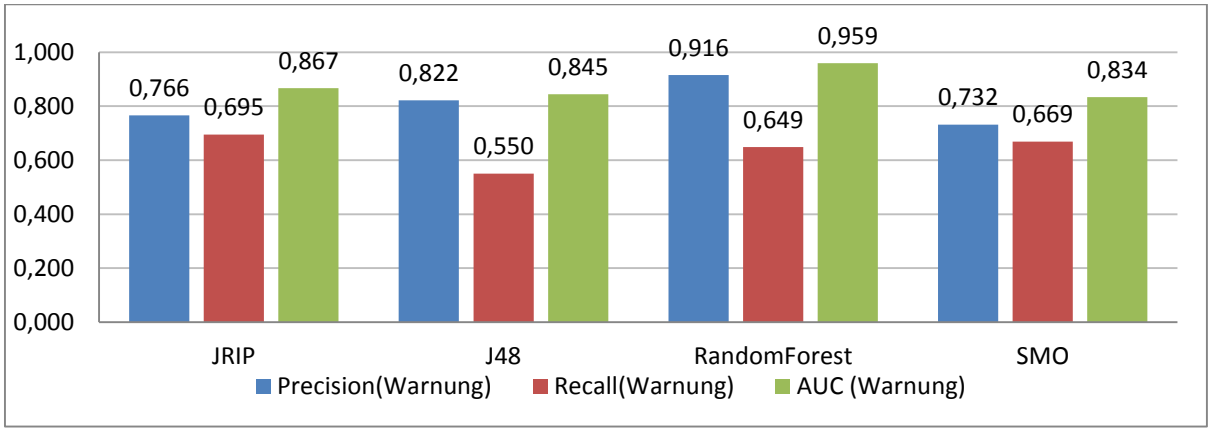


Abbildung 22 - Versuch 1.3 - Methode: AVG

In diesem Versuch ergeben sich bei Benutzung der AVG-Methode die besten Ergebnisse bei der Vorhersagedauer von 10 Tagen (Teil-Abbildung 22 c).

JRip zeigt bei dieser Dauer das höchste *Recall*-Ergebnis (69,5 %) und gleichzeitig hohe *Precision* (76,6 %) und *AUC* (0,867).

RandomForest kann diesen *Recall*-Wert nicht ganz erreichen (64,3 %), besitzt dafür die beste *Precision* (91,6 %) und *AUC* (0,959).

J48 liegt nahe bei *JRip*, kann aber nur 55 % *Recall* erreichen. Die *Precision* ist mit 82,2 % dafür etwas höher.

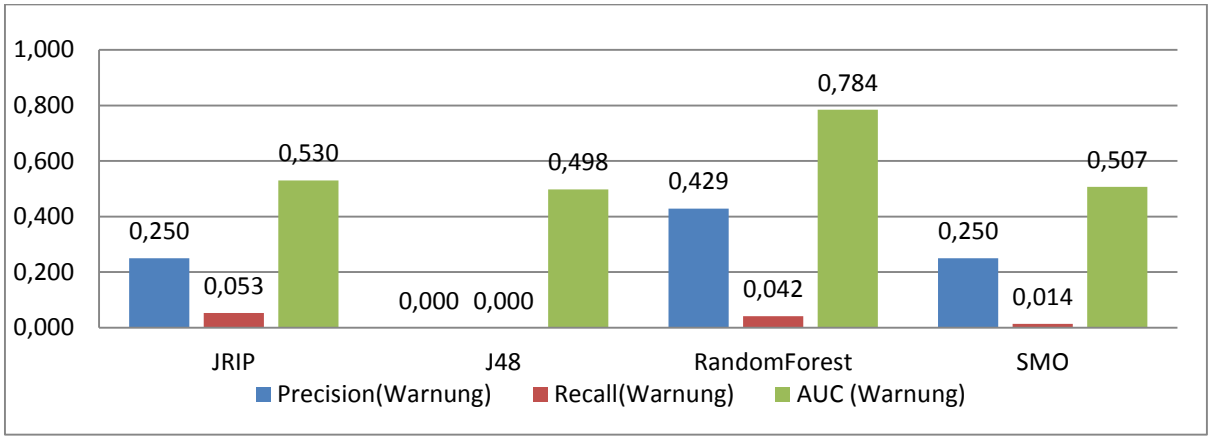
SMO zeigt einen hohen *Recall* Wert (66,9 %), *Precision* (73,2 %) und *AUC* (0,834) fallen im Vergleich mit den anderen niedriger aus.

Mit abnehmender Vorhersagedauer verschlechtern sich alle Klassifizierer. Wie schon in den vorigen Versuchen ist ein enormer Rückgang in den *Recall*-Werten zu beobachten.
Im kürzesten Zeitraum (Teil-Abbildung 22 a) sinkt der *Recall*-Wert bei *J48* auf 6,9 % ab. Die dabei erreichte *Precision* von 100 % hat bei diesem niedrigen Wert keine Aussagekraft mehr.

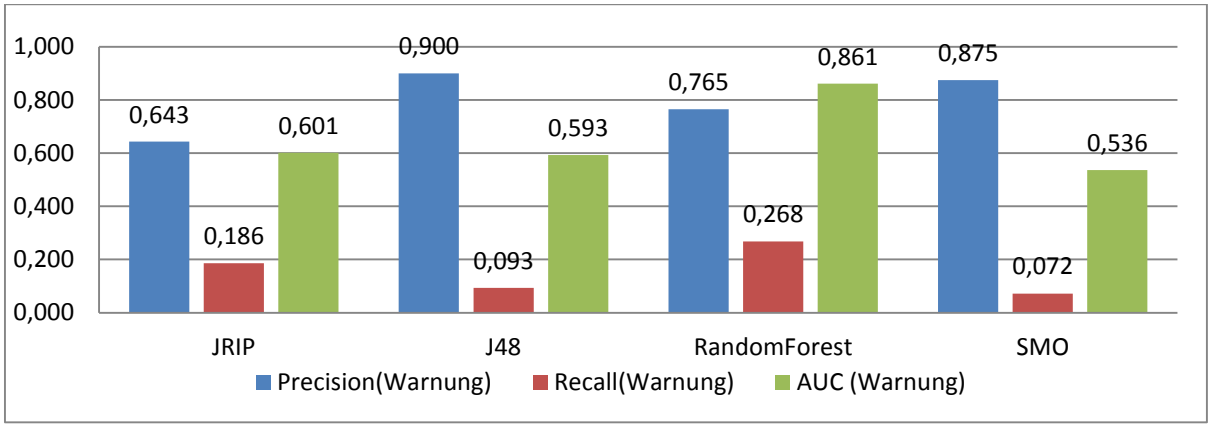
5.1.8. Versuch 1.3 – Methode VARIANCE

In diesem Abschnitt werden die Ergebnisse der VARIANCE-Methode dargestellt (Abbildung 23) und erläutert.

Teil-Abbildung 23 a - *v=5; w=3*; Methode: VARIANCE



Teil-Abbildung 23 b - *v=10; w=5*; Methode: VARIANCE



Teil-Abbildung 23 c - *v=30; w=10*; Methode: VARIANCE

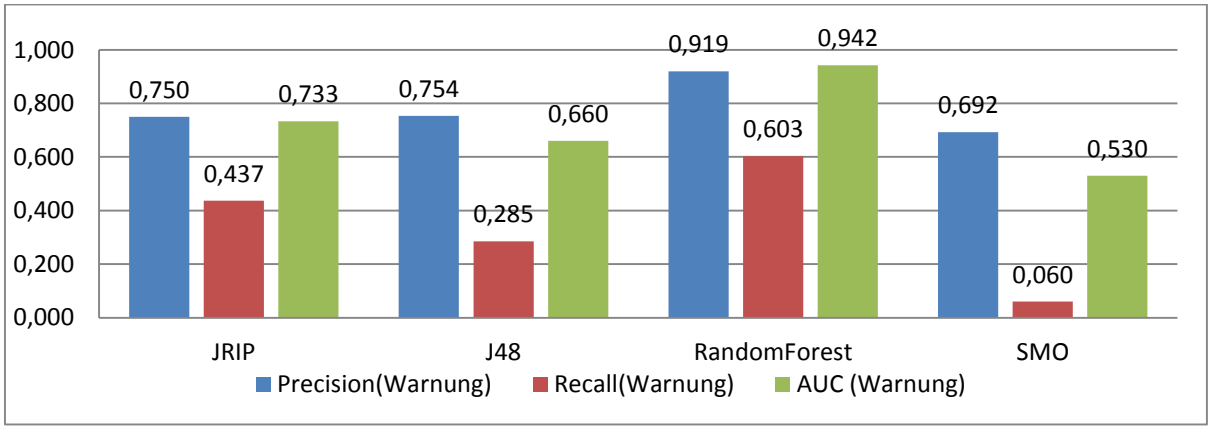


Abbildung 23 - Versuch 1.3 - Methode: VARIANCE

Die längeren Zeiträume zeigen gegenüber den kürzeren bessere Ergebnisse (Teil-Abbildung 23 c).

Im längsten Zeitraum überzeugt *RandomForest* in der Präzision (91,9 %) bei einem *Recall*-Wert von 60 % und erreicht somit das beste Ergebnis.
JRip (*AUC*: 0,733) liegt im Mittelfeld mit einem *Recall* von 43,7 %. *J48* (Recall: 28,5 %) und *SMO* (*Recall*: 6 %) können keine guten Ergebnisse erzielen.

Mit abnehmender Vorhersagedauer verschlechtern sich alle Klassifizierer. Ein enormer Rückgang in den *Recall*-Werten ist zu beobachten.
Im kürzesten Zeitraum (Teil-Abbildung 23 a) sinkt der *Recall*-Wert bei allen Klassifizierern auf Werte nahe 0 ab. Es können kaum noch Warnungen erkannt werden.

5.1.9. Versuch 1.3 – Vergleich der Methoden

Vergleicht man die beiden Methoden AVG und VARIANCE, zeigt sich, dass der *RandomForest* Klassifizierer mit beiden Methoden nahezu gleiche Ergebnisse liefert.
Die anderen Klassifizierer können jeweils bessere Ergebnisse mit der AVG-Methode erzielen.

Die Ergebnisse der AVG-Methode verschlechtern sich bei kleineren Vorhersagezeiträumen nicht so schnell wie die der VARIANCE-Methode.

5.1.10. Versuch 1.4

In diesem Kapitel werden die Ergebnisse von Versuch 1.4 dargestellt (Abbildung 24) und evaluiert.

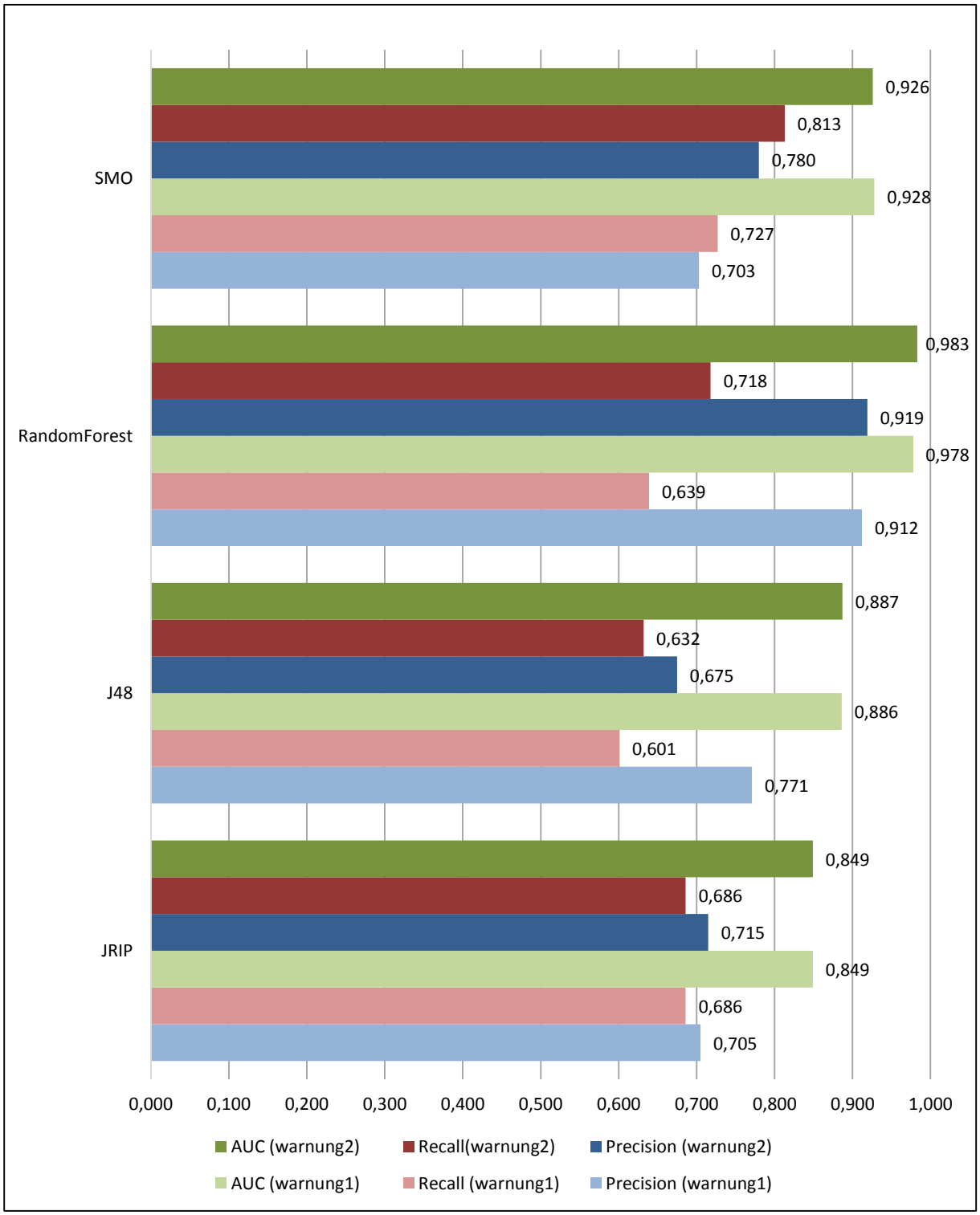


Abbildung 24 - Ergebnisse Versuch 1.4

Im Hinblick auf die *AUC*-Werte zeigt der *RandomForest*-Algorithmus das beste Ergebnis. Er kann für die Klasse „warnung1“ einen Wert von 0,983 und für „warnung2“ 0,978 erreichen. Bei der *Precision* zeigt *RandomForest* für beide Fehlerklassen die höchsten Werte (91,9 % bzw. 91,2 %).
SMO kann im Bereich des *Recalls* für beide Fehlerklassen die höchsten Werte verzeichnen und liefert 81,3 % bzw. 72,7 %.

J48 und *JRip* liefern ebenfalls gute Ergebnisse, diese liegen aber unterhalb der Ergebnisse von *SMO* und *RandomForest*.

In der Konfusionsmatrix von *RandomForest* (Tabelle 21) ist erkennbar, dass eine gute Diskrimination zwischen den beiden Schadtypen erfolgt und ein hoher Anteil der „warnung1“ und „warnung2“ Instanzen als solche erkannt werden.

Tabelle 21 - Ergebnis Versuch 1.4 - Konfusionsmatrix *RandomForest*

		klassifiziert als		
		normal	**warnung1**	**warnung2**
Klasse	**normal**	**19937**	20	21
	warnung1	122	**218**	1
	warnung2	97	1	**250**

Der *SMO*-Klassifizierer (Tabelle 22) zeigt ebenfalls eine gute Diskrimination und erkennt sogar einen höheren Anteil der Warnungen. Er stuft aber – verglichen mit *RandomForest* – eine größere Menge Instanzen fälschlicherweise als Warnung ein.

Tabelle 22 - Ergebnis Versuch 1.4 - Konfusionsmatrix *SMO*

		klassifiziert als		
		normal	**warnung1**	**warnung2**
Klasse	**normal**	**19798**	101	79
	warnung1	92	**248**	1
	warnung2	61	4	**283**

5.1.11. Versuch 1.5

Im Folgenden werden die Ergebnisse des Versuchs 1.5 (Abbildung 25) dargestellt und diskutiert.

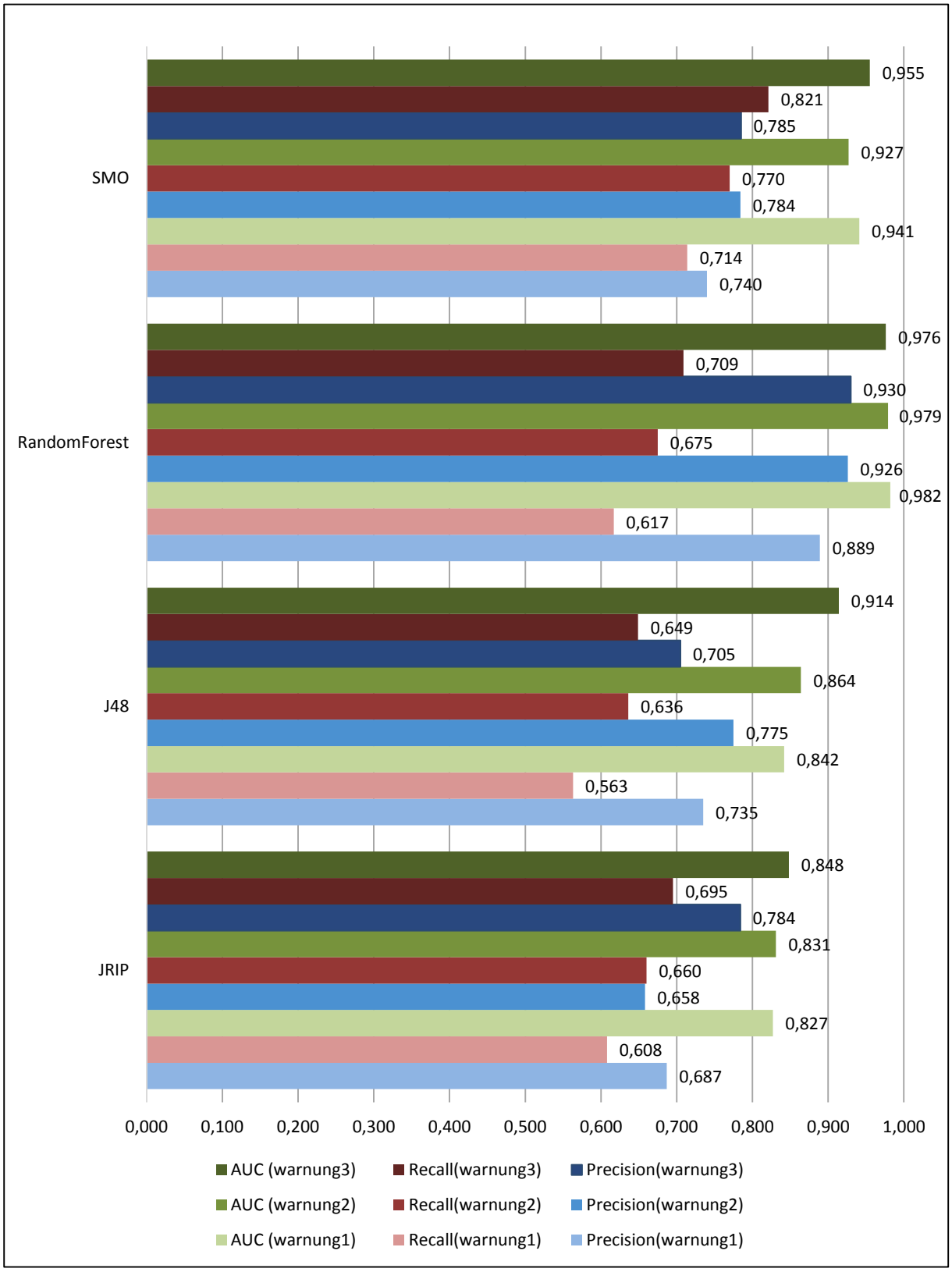


Abbildung 25 - Ergebnisse Versuch 1.5

RandomForest zeigt bei diesem Versuch die höchsten Werte sowohl für die *AUC* als auch die *Precision* aller drei Warnungs-Klassen. Bei den *Recall*-Werten kann *SMO* bessere Resultate liefern. *J48* und *JRip* zeigen auch sehr gute Ergebnisse, können aber nicht die Werte von *SMO* und *RandomForest* erreichen.

Bei der Betrachtung der Konfusionsmatrix des *RandomForest* Klassifizierers (Tabelle 23) zeigt sich eine gute Diskrimination der verschiedenen Warnungs-Klassen, insbesondere auch zwischen den Schäden von Versuch 1.2 und Versuch 1.3, die auf den gleichen Attributen basieren.

Tabelle 23 - Versuch 1.5 – Konfusionsmatrix *RandomForest*

		klassifiziert als			
		normal	**warnung1**	**warnung2**	**warnung3**
Klasse	**normal**	**19753**	25	16	7
	warnung1	128	**209**	2	0
	warnung2	107	1	**226**	1
	warnung3	44	0	0	**107**

In der Konfusionsmatrix von *SMO* (Tabelle 24) ist die Diskrimination nicht ganz so ausgeprägt, dafür werden mehr Instanzen der Fehlerklassen korrekt wiedererkannt.

Tabelle 24- Versuch 1.5 - Konfusionsmatrix *SMO*

		klassifiziert als			
		normal	**warnung1**	**warnung2**	**warnung3**
Klasse	**normal**	**19626**	80	65	30
	warnung1	89	**242**	6	2
	warnung2	72	3	**258**	2
	warnung3	25	2	0	**124**

Die gute Diskrimination impliziert, dass durch verschiedene Schäden unterschiedliche Vorkommens-Muster in den Diagnosedaten hinterlassen werden, welche von den Klassifizierern erkannt werden.

5.1.12. Versuch 1.6

Bei den Versuchen 1.1 - 1.3 lieferte der größte betrachtete Zeitraum durchgehend die besten Klassifizierungsergebnisse. Um festzustellen, ob die Ergebnisse noch weiter verbessert werden können, wird eine erweiterte Untersuchung des Verhaltens in Bezug auf die beiden Parametergrößen *v* und *w* vorgenommen. Als Grundlage wird die Datenbasis von Versuch 1.1 genutzt.

Eine Ergebnismatrix für verschiedene Kombinationen der *v* und *w*-Parameter wird erzeugt. Die AVG-Methode und der *RandomForest* Klassifizierer werden verwendet, da diese in den vorangehenden Versuchen die besten Ergebnisse lieferten. Der *AUC*-Wert wird als Vergleichswert genutzt. Als Evaluierungsmethode wird *5-Fold Cross-Validation* verwendet.

Es zeigt sich in Tabelle 25, dass mit zunehmenden Werten für *v* und *w* bessere Ergebnisse erzielt werden können. Das höchste Ergebnis mit einem *AUC* Wert von 0,999 liefert der Klassifizierer bei $v=80$ und $w=50$. Die Farbcodierung stellt die niedrigsten Werte mit roten Farbtönen und die höchsten Werte mit grünen Farbtönen dar.

Tabelle 25 - Ergebnismatrix Versuch 1.6

	v=100	*v=90*	*v=80*	*v=70*	*v=60*	*v=50*	*v=40*	*v=30*	*v=20*
w=50	0,991	0,992	0,999	0,997	0,992	0,99	0,982	0,978	0,966
w=40	0,995	0,995	0,996	0,997	0,989	0,989	0,984	0,979	0,97
w=30	0,994	0,99	0,993	0,996	0,995	0,988	0,991	0,978	0,973
w=20	0,984	0,992	0,995	0,995	0,99	0,99	0,988	0,976	0,972
w=10	0,982	0,989	0,992	0,992	0,99	0,972	0,981	0,968	0,967

In der Konfusionsmatrix (Tabelle 26) zeigt sich das Klassifikationsergebnis bei dieser Parametrisierung.

Tabelle 26 - Konfusionsmatrix Versuch 1.6

		klassifiziert als	
		normal	**warnung**
Klasse	**normal**	**15936**	12
	warnung	34	**534**

Die *Precision* für die Warnung-Klasse liegt bei 97,8 %, der *Recall* bei 94 %, was eine weitere Steigerung zu den Resultaten aus Versuch 1.1 darstellt.

Dieser Versuch kann für die anderen Schad-Arten unterschiedliche Ergebnisse liefern.

5.1.13. Komplexität der Modelle

Bevor die abschließende Bewertung stattfindet, wird die Komplexität der trainierten Modelle betrachtet. Anhand eines Regelwerkes oder eines Entscheidungsbaumes können Rückschlüsse über die Ursache des Problems gezogen werden, wenn die resultierende Komplexität des Baumes oder des Regelwerks überschaubar bleibt.
Der Datensatz mit der besten Erkennungsperformance ist aus Versuch 1.6 mit $v=40$ und $w=30$. Auf Basis dieses Datensatzes werden die Komplexität von *JRip* und *J48* betrachtet. *RandomForest* und *SMO* sind aufgrund ihrer komplexen Algorithmen bzw. den daraus resultierenden Ergebnissen nur schwer in Hinsicht auf ihre Komplexität zu beurteilen.

Datensatz: Job0001-v40-w30.arff

***JRip* (Standardparametrisierung)**
Mit *JRip* werden 35 Regeln mit bis zu 9 *if*-Entscheidungen je Regel erzeugt.

***J48* (Standardparametrisierung)**
J48 erzeugt einen Baum mit 157 Blättern und 313 Verzweigungen.

Beide Ergebnisse weisen solch ein Maß an Komplexität auf, so dass bei Betrachtung durch einen Menschen nicht unmittelbar erkannt werden kann, wie sich die gelernten Muster und Korrelationen zusammensetzen.

5.1.14. Zusammenfassung und Bewertung

Die Versuchsreihe zeigt, dass eine Vorhersage von Schäden anhand der Häufigkeit des Auftretens der Diagnosecodes möglich ist. Durch eine optimierte Parametrisierung lassen sich gute Ergebnisse erzielen, die weit über den derzeitig verfügbaren Möglichkeiten liegen. Die eingangs statuierte Hypothese zeigt sich demnach als bestätigt.

Die hier gewählte Vorgehensweise ist aufwändig. Sie benötigt große Datenbanken zur Zwischenspeicherung der Diagnosedaten. Idealerweise müssten die Diagnosedaten in möglichst kurzen Zeitintervallen den Weg von der Lokomotive in diese Datenbanken finden. Dies ist derzeit nicht der Fall.

Der manuelle Aufwand für diese Vorgehensweise ist groß. Für jeden Schadfall muss ein Modell gelernt werden. Kombinierte Modelle sind zwar möglich (siehe Versuch 1.4 und 1.5), der Aufwand hierfür wird aber nur unwesentlich reduziert und die Kombination wirkt sich negativ auf die Erkennungsgenauigkeit aus. Eine optimale Parametrisierung muss manuell für jedes Modell ermittelt werden.
Es ist ungewiss, ob ein einmal gelerntes Modell Bestand hat oder in regelmäßigen Abständen neu trainiert werden muss. Ebenfalls ist ungewiss, ob sich ein Modell auf eine andere Lokomotiv-Baureihe übertragen lässt. Hier vervielfacht sich der Aufwand, aber es ist nicht ausgeschlossen, dass sich ein Teil davon für einen späteren Livebetrieb automatisieren lässt.

Im derzeitigen Entwicklungsstand ist die Datenqualität der Diagnosedaten schlecht. Aufgrund der Diskontinuitäten könnten sich bei einer zukünftigen Verbesserung der Datenkontinuität Veränderungen der Vorhersageleistung einstellen, bedingt durch Effekte, die sich zum jetzigen Zeitpunkt nicht vorhersehen lassen. Diese Veränderungen könnten sowohl positiv als auch negativ ausfallen. In beiden Fällen wird eine Anpassung der Parametrisierung der Modelle nötig sein, um eine optimale Leistung zu erzielen.

Zusammengefasst lässt sich sagen, dass die hier vorgestellte Methode viel Potential hat. Gepaart mit regelmäßigen und kontinuierlichen Diagnosedaten-Aufzeichnungen bietet sie eine Möglichkeit zur frühzeitigen Erkennung von Schäden. Sie kann damit dazu dienen, die Kosten zu reduzieren und Ausfälle zu minimieren.

5.1.15. Verbesserungsmöglichkeiten

Aufgrund der Versuchsergebnisse – vor allem denen von Versuch 1.6 – ergibt sich die Idee, eine weitere Warnstufe einzuführen. Neben der normalen Warnung könnte eine „kritische Warnung“ dazu dienen, den Zeitpunkt des Schadens näher einzugrenzen und zur dringenden Handlung aufzurufen. Gerade bei den Ergebnissen mit einem Warnzeitraum von über 30 Tagen wäre dies wichtig, um Abstumpfung der verantwortlichen Personen gegenüber den Warnmeldungen zu vermeiden. Gleichzeitig können damit unnötige Werkstattbesuche verhindert werden.

Die Klassifizierer können ebenfalls optimiert werden. In den Versuchen werden die Standardeinstellungen genutzt (*SMO* ausgenommen). Auch *Bagging* (Breiman, Bagging predictors, 1996) oder *Boosting*-Methoden (Freund & Schapire, 1996) können noch eingesetzt werden. *Bagging* und *Boosting* sind Meta-Klassifizierer. Sie kombinieren die Ergebnisse mehrerer anderer Klassifizierer und erzeugen mittels einer gewichteten Bewertung ein optimiertes Ergebnis.

Durch bessere Attributselektion für die verschiedenen Schadfälle könnte eine Steigerung der Detektionsleistung erzielt werden. In diesem Versuchsaufbau fand eine manuelle Vorfilterung der Attribute (Diagnosecodes) statt. Im Idealfall werden die Attribute maschinell auf ihre Aussagekraft hin untersucht und danach ausgewählt. Hier können zuvor unbekannte Korrelationen zustande kommen. Aus Performancegründen wurde dies in den hier durchgeführten Versuchsreihen nicht durchgeführt.

Eine *Feature*-Erzeugung mit komplexeren Methoden, zum Beispiel Extrapolations- oder Regressionsmethoden, können unter Umständen bessere Ergebnisse liefern.

Das hier angewandte Verfahren basiert auf Zeitintervallen. Ein Verfahren welches auf dem Kilometerstand der Lokomotive basiert, könnte die Ergebnisse noch verbessern. Die gefahrenen Kilometer wären hierbei Grundlage der Aggregation, nicht die verstrichene Zeit. Es hätte den entscheidenden Vorteil, dass Tage an denen die Lokomotiven stillstehen oder nur wenig bewegt werden, nicht verfälschend in den Trainingsprozess eingreifen.

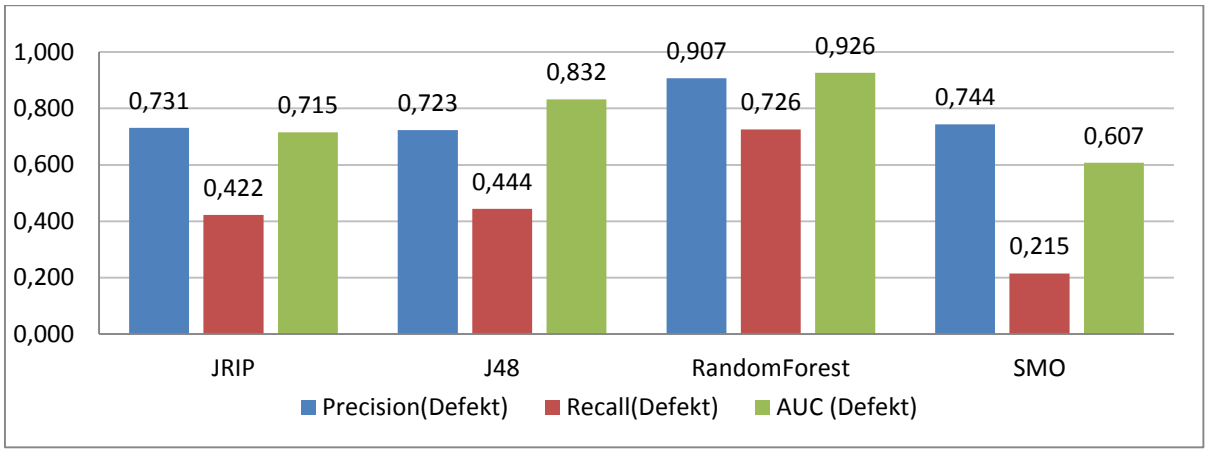
5.2. Ergebnisse Versuchsreihe 2

Dieses Kapitel stellt die Ergebnisse der Versuchsreihe 2 dar und vergleicht die Klassifikationsergebnisse der Versuche miteinander.

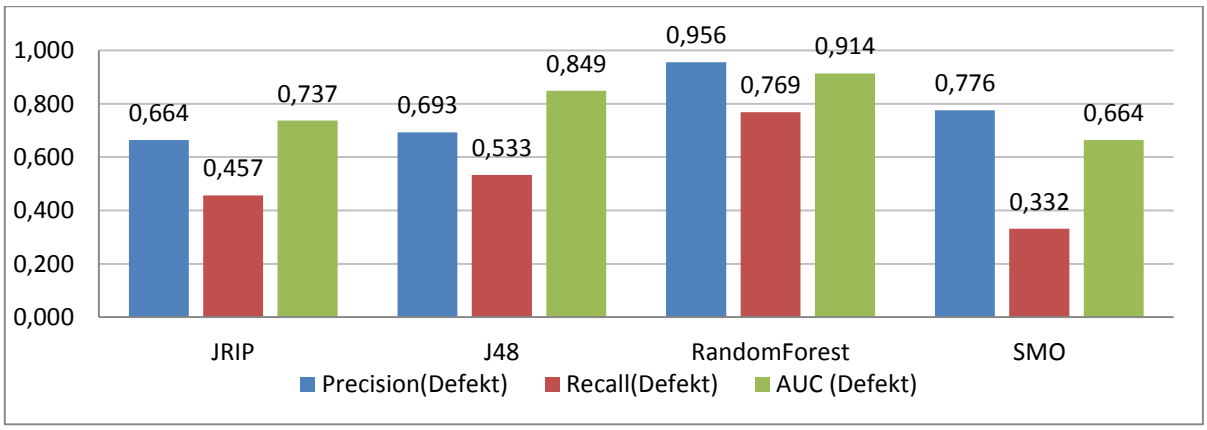
5.2.1. Versuch 2.1

In Abbildung 26 sind die Ergebnisse des Versuchs dargestellt.

Teil-Abbildung 26 a – Vorhersagedauer 5 Tage



Teil-Abbildung 26 b – Vorhersagedauer 10 Tage



Teil-Abbildung 26 c – Vorhersagedauer 20 Tage

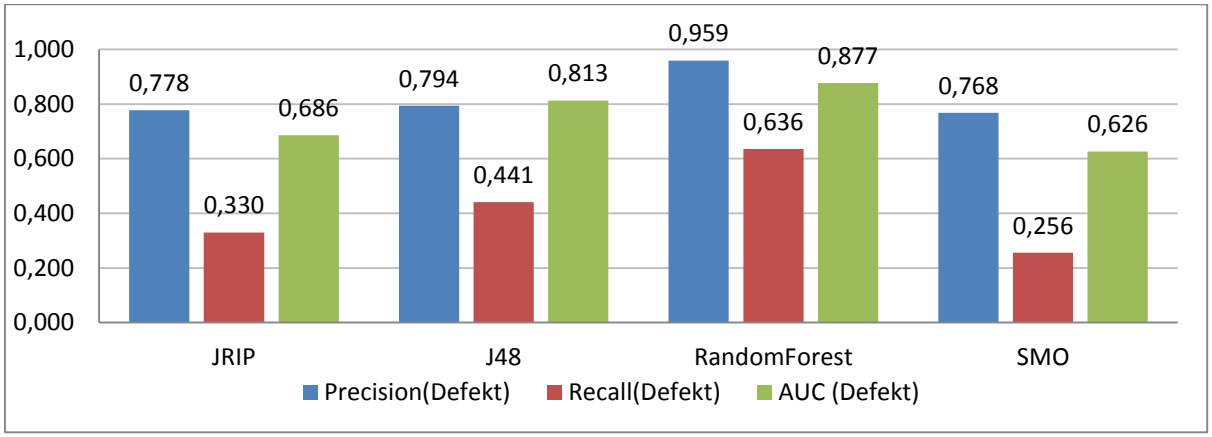


Abbildung 26 - Ergebnisse Versuch 2.1

Anhand der *AUC*-Werte beurteilt liefern die Klassifizierer mit einer Vorhersagedauer von 5 und 10 Tagen die besten Ergebnisse. Im Gegensatz zu Versuchsreihe 1 ist keine generelle Verschlechterung bei abnehmenden Vorhersagezeiträumen zu beobachten.

RandomForest liegt vor den anderen Methoden und erzielt einen *Recall*-Wert von 72,6 % bei 5 Tagen Vorhersagedauer. Eine genauere Betrachtung der Ergebnisse von *RandomForest* (10 Tage Vorhersage) zeigt, dass sehr wenige „normale“ Instanzen fälschlicherweise als „defekt“ klassifiziert wurden (Tabelle 27), was sich durch eine sehr gute *Precision* von 95,6 % zeigt.

Tabelle 27 - Versuch 2.1 - Konfusionsmatrix *RandomForest*, 10 Tage Vorhersage

		klassifiziert als	
		normal	**defekt**
Klasse	**normal**	**6379**	7
	defekt	46	**153**

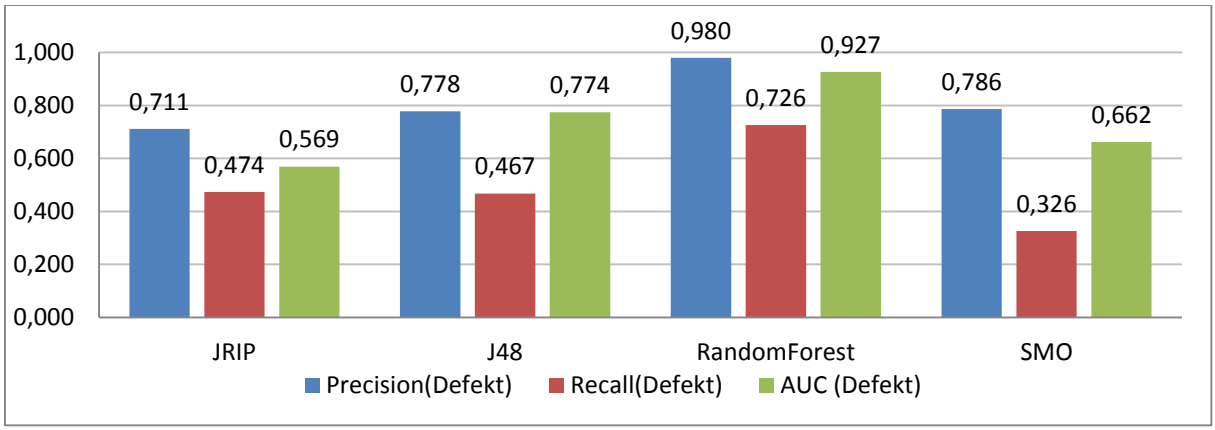
Die anderen Klassifizierer liegen – vor allem im *Recall*-Wert – hinter *RandomForest* zurück.

J48 und –*JRip* weisen eine höhere *Precision* im 5-Tage Zeitraum verglichen mit dem 10-Tage Zeitraum auf, bei den *Recall*-Werten ist es umgekehrt. *SMO* erzielt das beste Gesamtergebnis bei 10 Tagen Vorhersagedauer, kann aber maximal einen *Recall*-Wert von 33,2 % erzielen und schneidet damit diesbezüglich von allen Klassifizierern am schlechtesten ab.

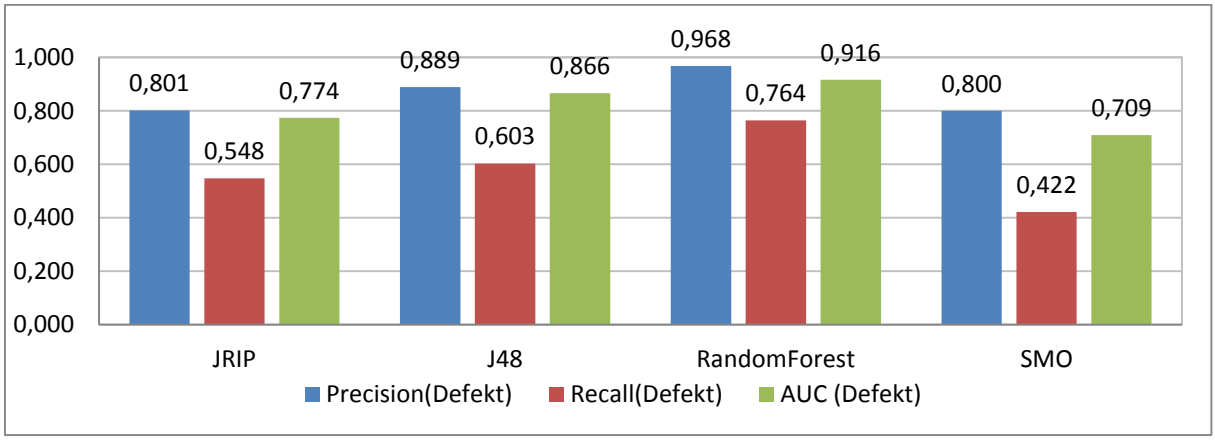
5.2.2. Versuch 2.2

Die Ergebnisse von Versuch 2.2 sind in Abbildung 27 dargestellt.

Teil-Abbildung 27 a – Vorhersagedauer 5 Tage



Teil-Abbildung 27 b – Vorhersagedauer 10 Tage



Teil-Abbildung 27 c – Vorhersagedauer 20 Tage

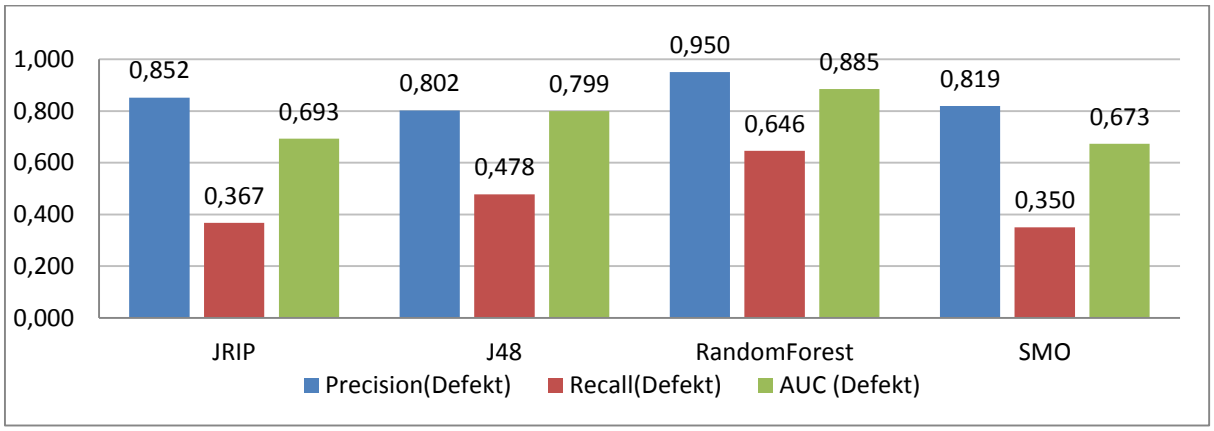


Abbildung 27 - Ergebnisse Versuch 2.2

RandomForest kann bei den Vorhersagedauern von 5 und 10 Tagen die beste Leistung verzeichnen. Bei 5 Tagen liegen die Höchstwerte für *Precision* (98 %) und *AUC* (0,927) vor, bei 10 Tagen wird der maximale *Recall*-Wert von 76,4 % erreicht.

Durch die Berücksichtigung der saisonalen Schwankungen in den Messwerten können die Ergebnisse von *Recall* und *Precision* gesteigert werden (Abbildung 28). Dies erscheint logisch, führt diese Berücksichtigung schließlich dazu, dass die Temperaturwerte der Motoren näher zusammenrücken und die Abweichungen durch die defekten FM stärker herausstechen. *J48* und *JRIP* zeigen deutliche Verbesserungen verglichen mit Versuch 2.1, *RandomForest* und *SMO* können durch diesen *Preprocessing*-Schritt keine Verbesserung erzielen.

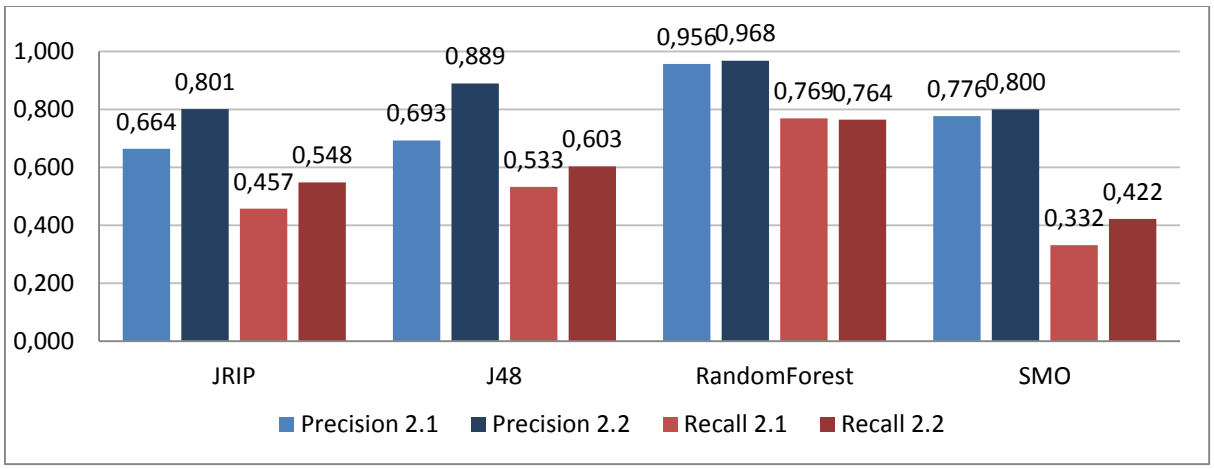
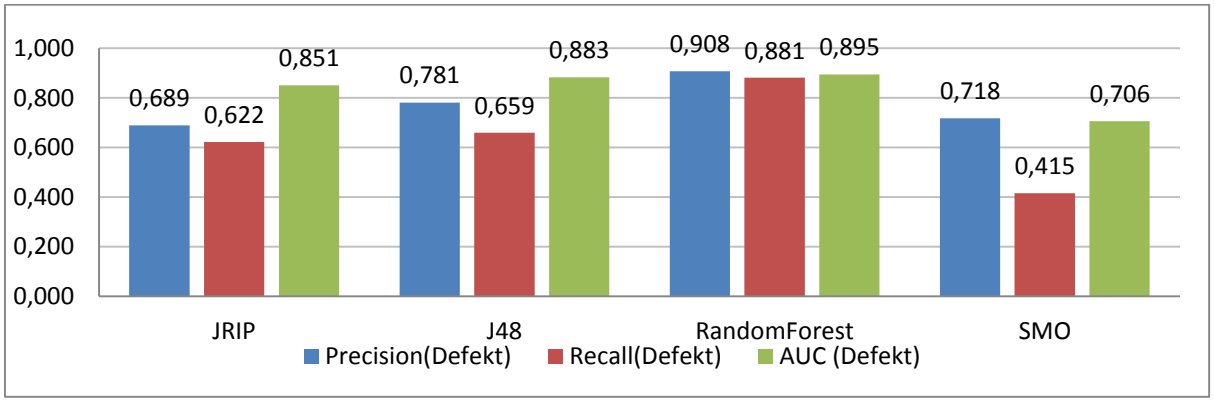


Abbildung 28 - Vergleich 2.1, 2.2 (10 Tage Vorhersagedauer)

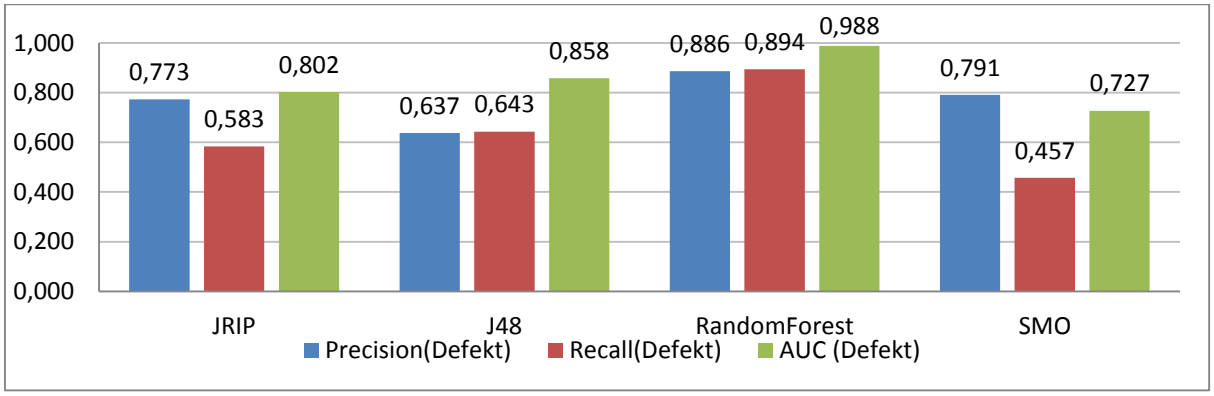
5.2.3. Versuch 2.3

Abbildung 30 zeigt die Ergebnisse aus Versuch 2.3.

Teil-Abbildung 29 a – Vorhersagedauer 5 Tage



Teil-Abbildung 29 b – Vorhersagedauer 10 Tage



Teil-Abbildung 29 c – Vorhersagedauer 20 Tage

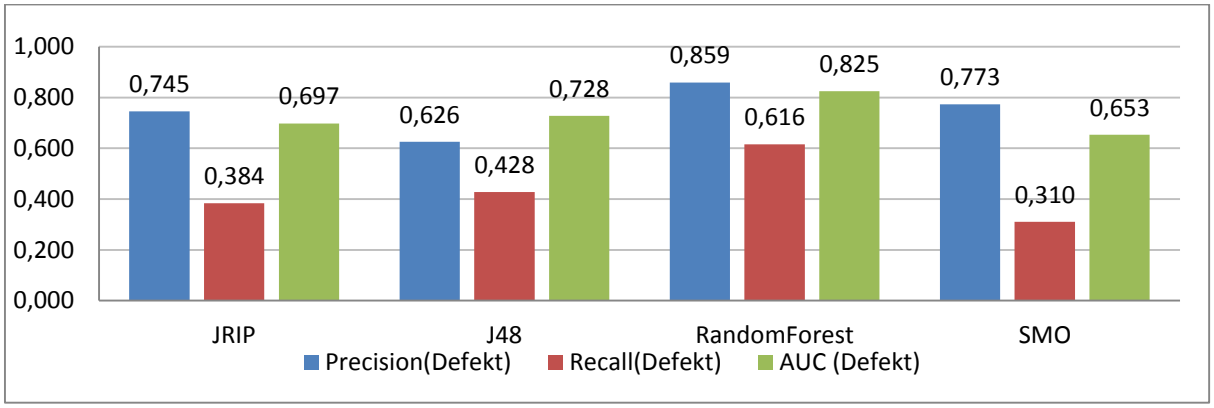


Abbildung 29 - Ergebnisse Versuch 2.3

Bei der Vorhersagedauer von 5 Tagen können alle Klassifizierer, verglichen mit Versuch 2.2, deutliche Steigerungen im *Recall*-Wert verzeichnen.

RandomForest leistet 88,1 % *Recall* bei einer gleichzeitigen *Precision* von 90,8 %.

Bei *RandomForest* steigert sich der *Recall*-Wert bei 10 Tagen Vorhersagedauer auf 89,4 %, was den Maximalwert der gesamten Versuchsreihe darstellt.

JRip und *J48* zeigen die besten Ergebnisse bei einem Zeitraum von 5 Tagen. *SMO* hingegen hat die höchsten Werte bei 10 Tagen Vorhersagedauer.

Die Klassifizierer, die auf den *Outlier*-bereinigten Datensätzen trainiert wurden, können nur geringfügige weitere Verbesserungen aufweisen. Bei allen Algorithmen geht die Präzision im Vergleich zu Versuch 2.2 zurück, der *Recall*-Wert hingegen kann sich nochmals verbessern (Abbildung 30 und Abbildung 31).

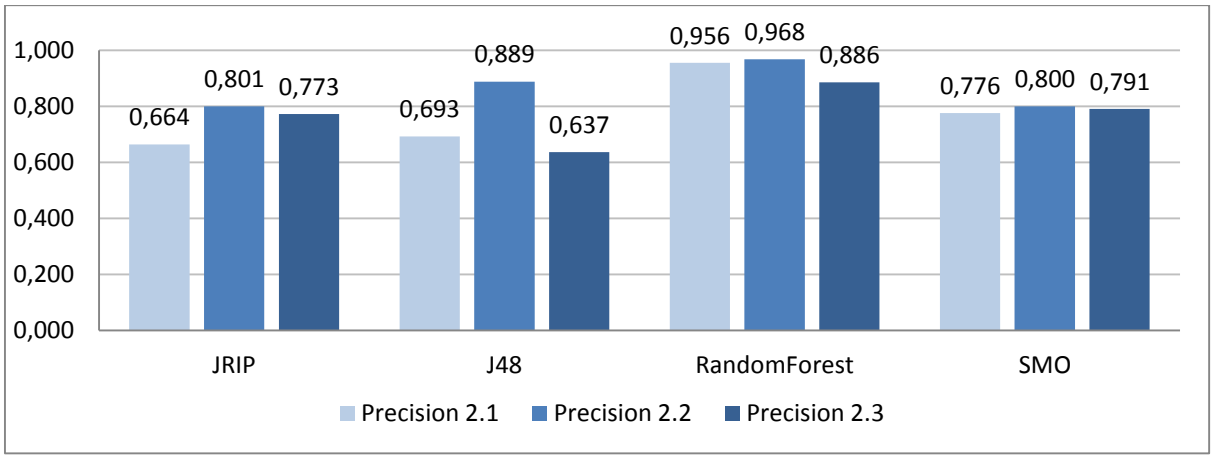


Abbildung 30 - *Precision* - Vergleich 2.1, 2.2, 2.3

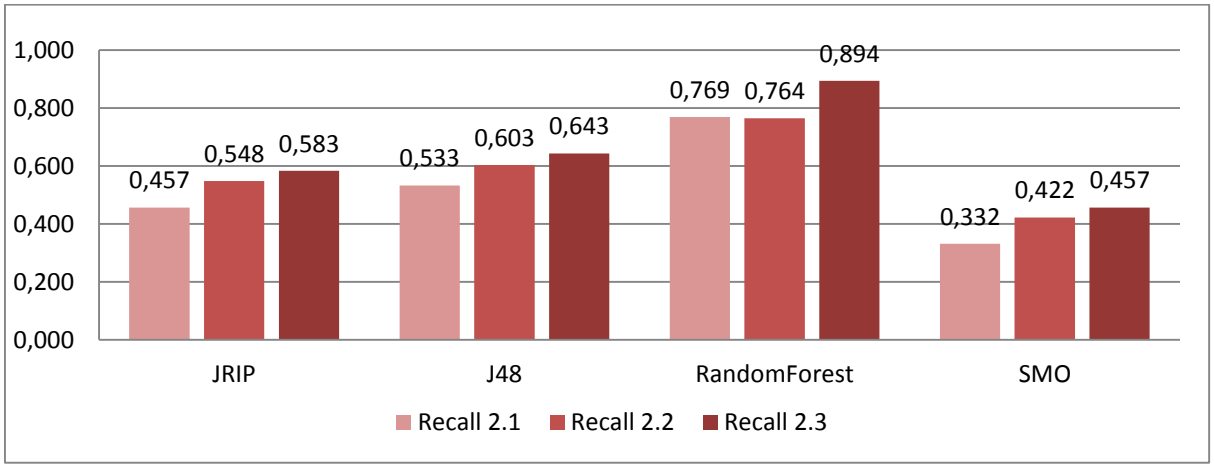


Abbildung 31 - *Recall* - Vergleich 2.1, 2.2, 2.3

Im Falle des *RandomForest* Algorithmus erreicht der *Recall*-Wert 89,4 %, was, bei gleichzeitig hohem *Precision*-Wert, zu einem *AUC*-Wert von 0,988 führt. In der Konfusionsmatrix stellt sich dieses Ergebnis wie in Tabelle 28 dar.

Tabelle 28 - Versuch 2.3, Konfusionsmatrix *RandomForest*

		klassifiziert als	
		normal	**defekt**
Klasse	**normal**	**6363**	23
	defekt	21	**178**

5.2.4. Versuch 2.4

Aufgrund der Übersichtlichkeit sind in diesem Kapitel nur die Ergebnisse des *RandomForest* Algorithmus‘ dargestellt. Im Anhang „Auswertung Versuchsreihe 2.xlsx“ sind die vollständigen Ergebnisse vorhanden.

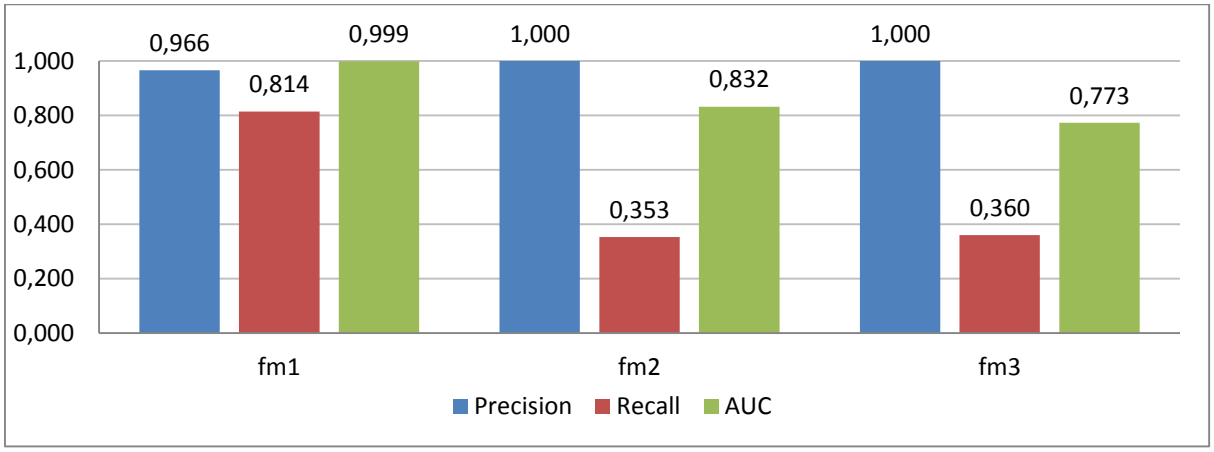


Abbildung 32 - Versuch 2.4 – Ergebnisse *RandomForest*, Vorhersagezeitraum 10 Tage

Das *Labeling* von drei Fehlerklassen liefert, trotz weniger vorhandener Schadfälle, gute Klassifikationsergebnisse.

Die Fehlerklasse für FM1 kann gut klassifiziert werden. Dies zeigt sich im *Recall*-Wert von 81,4 % bei einer *Precision* von 96,6 %. Das lässt sich darauf zurückführen, dass für diese Art von Schaden eine größere Menge Instanzen vorhanden ist.

Die übrigen Fehlerklassen werden nicht so gut erkannt, es werden nur *Recall*-Werte von 35,3 % bzw. 36 % erreicht. Die Diskrimination zwischen den einzelnen Schadklassen ist hoch, ein Blick in die Konfusionsmatrix (Tabelle 29) verdeutlicht dies.

Tabelle 29 - Versuch 2.4 - *RandomForest*, Konfusionsmatrix

		klassifiziert als			
		normal	**fm1**	**fm2**	**fm3**
Klasse	**normal**	**6382**	4	0	0
	fm1	26	**114**	0	0
	fm2	22	0	**12**	0
	fm3	16	0	0	**9**

Die Trennung der Fehlerklassen untereinander ist fehlerfrei, was darauf schließen lässt, dass für die verschiedenen Schäden jeweils andere *Features* ausschlaggebend sind. Einige Instanzen der fehlerbehafteten Motoren werden als „normal“ klassifiziert, der *Recall*-Wert ist unzureichend. Die *Precision* hingegen ist hoch, so dass nur vier Instanzen der Klasse „normal“ fälschlicherweise als Schadwarnung klassifiziert werden.

5.2.5. Weitere Betrachtungen der Ergebnisse

Anhand der Ergebnisse wird im Folgenden eine Betrachtung der resultierenden Modellkomplexität vorgenommen. Darauf folgt eine Evaluation der Attribute, die zeigen soll, welche Attribute den größten Informationsgehalt für die Klassifizierer liefern.

Komplexität der Modelle

Zur Betrachtung der Komplexität der trainierten Modelle wird ein Datensatz aus Versuch 2.3 verwendet (Warnzeitraum 10 Tage). In diesem Versuch lieferten alle Algorithmen gute Ergebnisse.

Die Modelle von *JRIP* und *J48* werden betrachtet, die Modelle von *SMO* und *RandomForest* sind zu komplex für solch eine Betrachtung.

Datensatz: Job0012-zsb_deseas_filter_10.arff

JRip (Standardparametrisierung)

JRip erzeugt 12 Regeln mit bis zu 6 *if*-Entscheidungen je Regel.

J48 (Standardparametrisierung)

J48 erzeugt einen Baum mit 45 Blättern und 89 Verzweigungen.

Die erzeugten Regelwerke bzw. Bäume sind in überschaubarer Größe.
Bei genauerer Betrachtung des Regelwerkes von *JRip* lässt sich erkennen, dass ein Attribut in fast allen Regeln zum Einsatz kommt: Das Verhältnis der Temperatur des defekten Motors zu den übrigen Motoren.

Evaluation der Attribute

Um herauszufinden, welche Attribute die ausschlaggebenden sind und um Rückschlüsse auf die technischen Ursachen zu finden, wird eine Evaluation der Attribute durchgeführt.

Hierfür wird mit dem *RandomForest* Algorithmus auf dem Datensatz zsb_deseas_filter_10.arff ein iterativer Greedy-Algorithmus mit den Attributen durchgeführt.

Anfangs sind keine Attribute zum Evaluieren vorhanden. In jedem Schritt des Algorithmus‘ wird ein neues Attribut zur Evaluation hinzugefügt. Es wird immer dasjenige Attribut hinzugefügt, welches für den aktuellen Schritt den größten *AUC*-Gewinn liefert.

Die Ergebnisse befinden sich in Tabelle 31. Zur Erinnerung noch einmal die Erklärung der Attribute (Tabelle 30).

Tabelle 30 - Attribute - Erklärung

Attribut	Erklärung
FM	Temperatur des Fahrmotors
LFM	Temperatur des Lagers
allFMavg	Durchschnittstemperatur aller Fahrmotoren

allLFMavg	Durchschnittstemperatur aller Lager
FM_LFM_ratio	Verhältnis FM zu LFM
FM_allFM_ratio	Verhältnis FM zu allen FM
LFM_allLFM_ratio	Verhältnis LFM zu allen LFM

Tabelle 31 - Greedy-Algorithmus: Attributevaluation

Attribute	**AUC**	**Prec(defekt)**	**Recall(defekt)**	**Hit Rate**	
ALLE	0,928	0,967	0,739	99,117	Vergleich

Iteration 1

FM	0,887	0,591	0,553	97,4438	
LFM	0,878	0,561	0,462	97,227	
allFMavg	0,861	0,612	0,618	97,6143	*
allLFMavg	0,853	0,561	0,558	97,2889	
FM_LFM_ratio	0,826	0,363	0,146	96,5763	
FM_allFM_ratio	0,84	0,347	0,085	96,6847	
LFM_allLFM_ratio	0,801	0,667	0,03	96,9636	

Iteration 2: allFMavg

FM	0,907	0,843	0,704	98,6832	
LFM	0,911	0,867	0,688	98,7142	
allLFMavg	0,898	0,824	0,658	98,5128	
FM_LFM_ratio	0,897	0,882	0,678	98,7297	
FM_allFM_ratio	0,914	0,847	0,724	98,742	*
LFM_allLFM_ratio	0,903	0,869	0,668	98,6677	

Iteration 3: allFMavg, FM_allFM_ratio

FM	0,91	0,908	0,698	98,8536	
LFM	0,925	0,924	0,729	98,9775	
allLFMavg	0,925	0,923	0,724	98,962	
FM_LFM_ratio	0,927	0,916	0,714	98,9156	
LFM_allLFM_ratio	0,92	0,954	0,734	99,0705	*

Iteration 4: allFMavg, FM_allFM_ratio, LFM_allLFM_ratio

FM	0,917	0,973	0,729	99,1015	
LFM	0,928	0,959	0,709	99,0085	
allLFMavg	0,92	0,973	0,734	99,117	*
FM_LFM_ratio	0,925	0,967	0,729	99,086	

Iteration 5: allFMavg, FM_allFM_ratio, LFM_allLFM_ratio, allLFMavg

FM	0,935	0,979	0,719	99,086	*
LFM	0,922	0,941	0,719	98,993	
FM_LFM_ratio	0,928	0,967	0,729	99,086	

Iteration 6: allFMavg, FM_allFM_ratio, LFM_allLFM_ratio, allLFMavg, FM

LFM	0,917	0,98	0,724	99,1015	
FM_LFM_ratio	0,935	0,967	0,744	99,1325	*
ALLE	0,928	0,967	0,739	99,117	Vergleich

Der * zeigt für jede Iteration die beste Auswahl an. Die Werte sind farblich so hinterlegt, dass die niedrigsten Werte rot und die höchsten Werte grün sind. Abstufungen dazwischen werden über einen Farbverlauf angezeigt.

Es zeigt sich, dass die Temperaturen der Fahrmotoren und das Verhältnis des einen Fahrmotors zu den übrigen die ersten wichtigen Attribute sind.
Auf dieser Basis werden Lagertemperatur und Verhältnisse hinzugenommen. Nach der vierten Iteration ist schon eine nahezu so gute Erkennung wie mit allen Attributen möglich.
Nach der sechsten Iteration ist das erzielte Ergebnis sogar besser als das Ergebnis mit allen Attributen. Das letzte Attribut (LFM) sorgt demnach für keine weitere Verbesserung mehr.

Interpretation

Das Zusammenspiel von Basis-Temperatur und Verhältnissen der einzelnen Motoren und Lagern liefert die benötigte Information.
In Iteration 1 ist sichtbar, dass die Verhältnisse (FM_allFM_ratio, FM_LFM_ratio, LFM_allLFM_ratio) nur geringe *Recall*-Werte erzeugen. Sie sind also nur in Verbindung mit anderen Attributen ausschlaggebend. Dies zeigt sich in den Iterationen zwei und drei, hier werden Verhältnis-Attribute gewählt.

Das *Feature*, welches am Ende keine Verbesserung mehr bringen kann, ist die Lager-Temperatur (LFM). Diese ist implizit auch in den Attributen FM_LFM_ratio und LFM_allLFM_ratio vorhanden, weshalb nicht behauptet werden kann, die Lagertemperatur hätte keinen Einfluss auf das Ergebnis.

5.2.6. Zusammenfassung und Bewertung

Die Versuchsreihe zeigt, dass die Vorhersage eines Schadens aufgrund analoger Messwerte möglich ist. Auch wenn für diese Versuche nur wenige Daten in geringer zeitlicher Auflösung vorlagen, sind die Ergebnisse vielversprechend. Es muss aber ausdrücklich gewarnt werden, dass die Ergebnisse mit besseren Daten anders aussehen können, und die Erkennungsraten sowohl geringer als auch höher ausfallen könnten.

Die Anwendungsmöglichkeiten sind breitgefächert: alle Systeme, die mit analogen (und digitalen) Sensoren versehen sind, können mit dieser Verfahrensweise überwacht werden.

Der Aufwand ist groß, es muss für jeden möglichen Schaden anhand von historischen Daten ein Modell zur Schaderkennung gelernt werden. Je nach Beschaffenheit des überwachten Systems müssen aus vorhandenen Messwerten zusätzliche *Features* generiert werden. Die Daten müssen zuerst über einen längeren Zeitraum angesammelt werden, damit einige Beispiele von Ausfällen aufgezeichnet werden können. Hier ist mindestens ein Jahr Vorlaufzeit nötig.

Zugriff auf umfassendes Detailwissen über die Funktionsweise der Lokomotive ist unabdingbar, Werkstattexperten müssen in das Projekt eingebunden werden.

Kombinierte Schad-Erkennungsmodelle oder zumindest Kombinationen von ähnlichen Typen scheinen möglich, siehe Versuch 2.4. Dies kann den Gesamtaufwand verringern, verschlechtert aber unter Umständen die Erkennungsgenauigkeit für den einzelnen Schadtypen. Es muss im Einzelfall untersucht werden, wie sich die Vor- und Nachteile auswirken und was als sinnvoll betrachtet wird.

5.2.7. Verbesserungsmöglichkeiten

Um die Möglichkeiten der aufgezeigten Methoden voll auszuschöpfen, müssen die Datenqualität und -quantität erhöht werden.

Eine regelmäßige Aufzeichnung der analogen Messwerte wäre sinnvoll. Eine Frequenz von vier Aufzeichnungen pro Stunde ist empfehlenswert. Diese Frequenz ermöglicht eine umfassende Datenanalyse, ohne zugleich die erzeugten Datenmengen derart zu steigern, dass kostspielige Änderungen der Infrastruktur im Zug und der Datenübertragung vorgenommen werden müssen.

Die meisten wichtigen und teuren Bauteile in einem Zug, für die ein solches Monitoring interessant ist, sind auf eine lange Betriebsdauer ausgelegt. Sie unterliegen langanhaltenden Alterungs- und Abnutzungsprozessen. Eine feingranulare Aufzeichnung – zum Beispiel im Sekundentakt – wird für diese Prozesse wahrscheinlich keinen Vorteil bieten.

Annahme: Alle 15 Minuten wird ein Datensatz mit 60 Temperaturwerten aus verschiedenen Systemen aufgezeichnet. Die Temperaturwerte sind alle als „*Float*“ vorhanden, benötigen also 2 Byte Speicherplatz.

Bei einer Siemens-Lokomotive der Baureihe 185 sind Diagnosemeldungen als ASCII-Strings codiert. Sie haben eine Maximallänge von 140 Zeichen. In diesen 140 Zeichen ist ein Substring mit 40 Zeichen Länge für Umfelddaten enthalten. In den 40 Zeichen lassen sich 20 Temperaturwerte (je 2 Byte) encodieren.
Es werden also drei Diagnosemeldungen benötigt, um alle 60 Temperaturwerte zu übertragen.

Pro Tag und Lokomotive benötigt dies bei einer Frequenz von 15 Minuten ein Speicher- und Übertragungsvolumen von

$$140\ \frac{\text{Byte}}{Meldung} * 3\ \frac{Meldungen}{15\ \text{min}} * 24\ \frac{\text{h}}{\text{d}} = \mathbf{40320}\ \frac{\mathbf{Byte}}{\mathbf{d}}$$

Es handelt sich damit um sehr kleine Datenmengen, die zusätzlich übertragen werden müssen.

Prinzipiell lässt sich nicht auszuschließen, dass es Anwendungsfälle für eine feinere Abtastrate geben kann. Die Möglichkeit, diese realisieren zu können sollte optional sein.
Die Handlungsmöglichkeiten für spontane Ausfälle sind jedoch beschränkt. Eine Sekundengenaue Auswertung direkt auf der Lokomotive ist nicht vorgesehen, und würde zusätzliche Hardware benötigen.

Es müssen in regelmäßigen Abständen Sensorwerte aufgezeichnet werden. Es kann durchaus sein, dass Sensorwerte auf einen Schaden hinweisen, obwohl sie, oberflächlich betrachtet, technisch und logisch nichts mit dem defekten Bauteil zu tun haben. Solche Verbindungen sind nicht leicht erkennbar, *Machine-Learning* Verfahren können die Verbindung aber herstellen und bieten im Nachhinein auch logische Erklärungen für etwas, das vorher vielleicht nur als Unregelmäßigkeit aufgefallen ist.

Eine weitere Verbesserungsmöglichkeit ist die Optimierung der Algorithmen. Durch verbesserte Parametrisierung und Benutzung von Verfahren wie *Boosting* und *Bagging* kann die Leistung erhöht werden.

6. Zusammenfassung und Ausblick

Bei der Betrachtung der Ergebnisse aus beiden Versuchsreihen fallen die guten Vorhersageresultate ins Auge. Gerade in Hinblick auf die zugrundeliegende Datenqualität ist dies erstaunlich.

Die Vermutung, dass sich mit höherer Datenqualität ein besseres Ergebnis erreichen lässt, liegt nahe, lässt sich aber nicht mit absoluter Sicherheit statuieren. Zu diesem Zeitpunkt wird davon ausgegangen, dass zumindest ähnlich gute Ergebnisse möglich sind.
Weitere Möglichkeiten in Bezug auf die Erkennungsgenauigkeit sind zu erwarten. Da meistens schleichende Prozesse für einen Ausfall verantwortlich sind, ist es sinnvoll, mehrere Stufen der Dringlichkeit in das Warnsystem zu implementieren. Eine „Warnung“ vor der „Dringenden Warnung“ würde den beteiligten Personen, die mit dem System arbeiten müssen, zusätzliche Informationen zum Handlungsbedarf bieten.

6.1. Datenqualität und Quantität

In Versuchsreihe 1 liegt der Hauptkritikpunkt an der Datenqualität darin begründet, dass die Diagnosedaten keinen zuverlässigen Pfad zwischen Lokomotive und zentraler Datenbank haben.
Derzeit ist kein System vorhanden, welches die Daten in zeitlich geregelten Abständen aus der Lokomotive in die Datenbank überspielt. Die manuell gepflegten Abläufe sind fehleranfällig und finden nicht regelmäßig statt, was in Datenverlust resultiert und wodurch Vorhersagemöglichkeiten in diesen Zeitabschnitten unmöglich werden. Es bietet sich an, eine Mobilfunkgestützte Übertragungslösung zu verwenden. Diese kann häufig und regelmäßig Daten in die Datenbank übertragen. Erste Schritte zur Erweiterung der Infrastruktur zur Realisierung einer solchen Lösung wurden Anfang 2014 eingeleitet, der Umsetzungszeitraum ist jedoch nicht bekannt.

Die Qualität der Werkstattaufzeichnungen muss ebenfalls adressiert werden. Bei einem Verfahren, welches auf historischen Daten und realen Schäden lernt, ist es wichtig, dass alle Schadfälle korrekt aufgezeichnet werden. Im hier genutzten Verfahren wurde nicht unterschieden, ob ein Bauteil tatsächlich defekt war, oder ob es auf Verdacht oder aus anderen Gründen ausgetauscht wurde. Beim Training der Modelle muss sichergestellt werden, dass nur tatsächlich defekte Bauteile als solche gelernt werden, sonst wird das Lernergebnis beeinträchtigt.

Bei Versuchsreihe 2 gestalten sich die Probleme schwieriger. Zusätzlich zur Problematik des Datenverlustes gestaltet sich die Sachlage so, dass zu wenige Datensätze mit Analogwerten aufgezeichnet werden.
Zur Erinnerung: Jeder Diagnosemeldung sind bestimmte analoge Messwerte beigefügt. Es ist also nicht unüblich, dass manche Temperaturwerte über mehrere Stunden hinweg nicht aufgezeichnet werden, da keine passende Diagnosemeldung auftritt.

Um dieses Problem zu umgehen, muss eine neue Diagnosemeldung generiert werden. Diese kann eine Protokoll-Meldung sein und muss in regelmäßigen Abständen gefeuert werden. In den Umfelddaten dieser Diagnosemeldung müssen die gewünschten Temperaturwerte mitgeliefert werden. Dies erhöht das Datenvolumen, welches von der Lokomotive transferiert wer-

den muss. In Kapitel 5.2.7 wurde gezeigt, dass die zusätzlich zu übertragenden Datenmengen nur geringe Ausmaße annehmen.

Zusammengefasst lässt sich sagen, dass eine konsistente und stetige Übertragung aller Daten von allen Lokomotiven vorliegen muss. Dies ist die Voraussetzung dafür, dass ein Vorhersagesystem überhaupt funktionieren kann.

6.2. Daten-Historie

Alle hier vorgestellten Verfahren basieren auf historischen Daten.
Wenn die Voraussetzung der konsistenten Übertragung erfüllt ist, müssen über einige Monate Daten aufgezeichnet werden, um mit der Modellbildung – basierend auf den neuen, verbesserten Daten – beginnen zu können.

Bei häufig auftretenden Schäden können wenige Monate ausreichen, um eine ausreichend große Basis an Beispielen zu schaffen. Bei seltenen Schäden wie dem Zentralschraubenbruch müssen längere Vorlaufzeiten eingeplant werden, um genug Beispiele aufzeichnen zu können.

Durch die fortlaufende Aufzeichnung können alle entstehenden Modelle in der Zukunft regelmäßig angepasst und verbessert werden. Dies ist ein iterativer Vorgang, der erst stoppt, wenn Schäden einer bestimmten Art immer rechtzeitig erkannt werden und nicht mehr auftreten.

6.3. Anbindung verschiedener Lokomotiven und Hersteller

Es ist nicht zu erwarten, dass Schadmodelle von einer Baureihe auf andere übertragbar sind. Hierfür sind die technischen Unterschiede zwischen den einzelnen Lokomotiven zu groß. Unter Umständen lassen sich die Vorhersagemodelle zwischen zwei Baureihen eines Herstellers teilweise wiederverwenden, wenn gleiche Systemkomponenten verwendet werden. Um dies zu verifizieren, müsste aber zuerst eine gesonderte Untersuchung stattfinden.

Eine zusätzliche Hürde besteht im Moment dadurch, dass die Übertragungsformate der Daten nicht einheitlich sind. Dies kann zwar in der Hard- und Software der Lokomotive nicht verändert werden, spätestens vor der UMTS-Übertragung zur Datenbank sollten aber alle Daten in ein Meta-Format konvertiert werden, welches zumindest die Daten in eine gleiche Form bringt (XML, JSON oder ähnliche, flexible Formate bieten sich an).

Der Vorteil hiervon wäre, dass die Anbindung an die Datenbank über diese vereinheitlichte Schnittstelle stattfindet und die Eigenheiten der Herstellerformate umgangen werden.

Ein entscheidender Nachteil ist die resultierende Größe. Bei einer Konvertierung aus einem binären Format in ein Metaformat steigt die Größe erheblich an. Ob dies ein ernsthaftes Übertragungsproblem verursachen würde, muss noch untersucht werden.

Als Alternative hierzu kann die Übertragung weiterhin im herstellereigenen Format erfolgen und die Konvertierung in das Metaformat erst serverseitig erfolgen.

6.4. Übertragungsablauf und Integration der Vorhersagelogik

Als Vorschlag für eine Integration in die bestehende und geplante Systemlandschaft dient Abbildung 33.

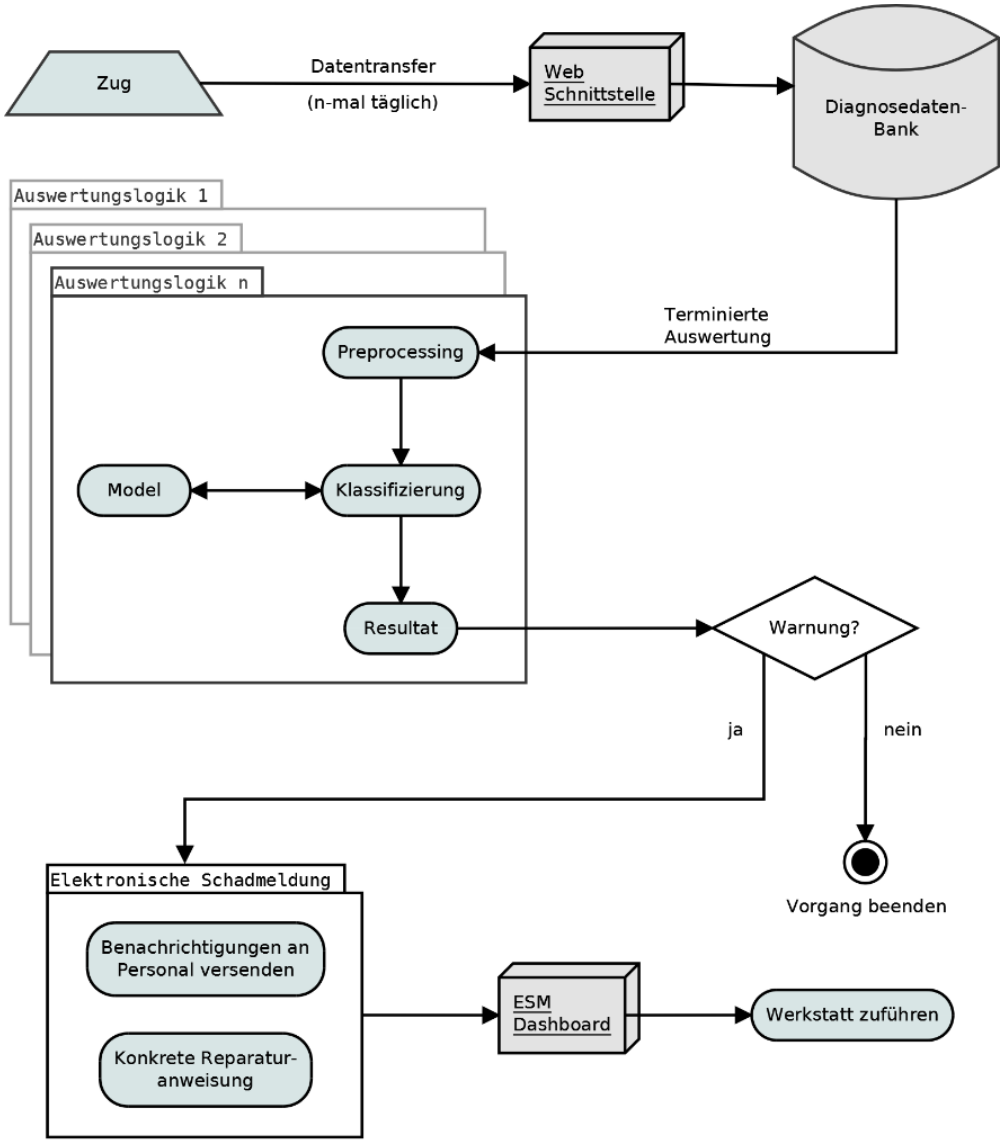


Abbildung 33 - Einbettung der Vorhersagelogik - Sollzustand

Vom Zug werden – im Idealfall – mehrfach täglich die aufgezeichneten Diagnosedaten an die Web-Schnittstelle gesendet und von dort in die Datenbank abgelegt. Dies sorgt dafür, dass die Diagnosedatenbank von allen Fahrzeugen immer den aktuellen Stand der Diagnosedaten beinhaltet.

In regelmäßigen Abständen werden die Prozesse zur Auswertung der potentiellen Schäden gestartet. Für jeden Schadtyp besteht eine separate Auswertungslogik.
Liefert eine der Auswertungslogiken eine Warnung für einen bestimmten Schadtyp, so wird eine Warnung in das System zur Elektronischen Schadmeldung (ESM) eingespeist.

Das System zur ESM befindet sich derzeit in der Entwicklung. Es soll eine Vereinheitlichung der Schadmeldungen vom Lokführer bis zur Werkstatt bewirken. Die Lokführer erfassen mittels eines Tablets ihre Meldungen. Im Gegensatz zur bisherigen „11-Punkte Meldung“ soll dieses System mit Hilfe vorgegebener Auswahlmöglichkeiten und Standardisierung eine Vergleichbarkeit der Schadmeldungen schaffen und gleichzeitig dem Werkstattteam bessere Informationen über die möglichen Quellen des Schadens bieten.
Dieses System ist so ausgelegt, dass es ein Dashboard enthält, in dem ein Operator die neusten Meldungen sieht und nacheinander abarbeitet. Es ersetzt den bisherigen Schadmeldungs-Mechanismus, der telefonisch erfolgte.

Es bietet sich an, die automatische Schadvorhersage an dieses System anzubinden, und die generierten Schadwarnungen im ESM-Dashboard in einer gesonderten Kategorie anzuzeigen. So verbleibt die letztendliche Entscheidung, eine Lokomotive der Werkstatt zuzuführen, beim Operator. Dieser kann anhand der Vorhersageergebnisse die Fehlerursache einschränken und die Werkstatt mit gezielten Anweisungen zur Fehlerbehebung beliefern. Er kann anhand der Dringlichkeit der Schadmeldung festlegen, ob eine Lokomotive sofort der Werkstatt zugeführt wird, oder ob die nächste reguläre Wartung ausreicht.

7. Anhang: Dokumentation des Quellcodes

Dieses Kapitel soll einen Überblick über die implementierten Programme und Scripts geben. Sowohl Java als auch *PHP* wurden genutzt, um die Daten zu importieren beziehungsweise weiterzuverarbeiten. Als Datenbank wurde *MySQL* verwendet.

Zur Ausführung des Javacodes ist das *Java* JDK 1.7 nötig. Genauere Informationen zu den benötigten Packages können dem Netbeans-Projekt im Anhang entnommen werden.

Die *PHP*-Scripte benötigen *PHP* 5.3 oder neuer. *MySQL* in Version 5.5 wurde verwendet. Als Hilfsprogramm für die Datenbank-Erstellung und Verwaltung wurde die Freeware „HeidiSQL“ in Version 8.3 verwendet (www.heidisql.com).

7.1. *MySQL* Datenbank

Die Definition der Datenbank und all ihrer Tabellen kann im Anhang "database structure dump.sql" eingesehen werden. Mit Hilfe dieser Datei kann die Struktur der Datenbank auf einem anderen *MySQL*-Server wiederhergestellt werden.

Tabelle "rawdata"

Dies ist die wichtigste Tabelle für die Versuchsreihe 1. In diese Tabelle werden die Diagnosedaten von den Lokomotiven der Baureihe 185 importiert.

- id - ID des Datensatzes,
- baunr - Baunummer der Lokomotive,
- tkmw - "TKMW"-Wert
- stat - Status-Wert
- fcode - Codenummer des Diagnosecodes
- lnummer - Laufende Nummer
- kdatum - Kommt-Datum (als Timestamp)
- gdatum - Geht-Datum (als Timestamp)
- kumfdat - Umfelddaten "Kommt"-Zeitpunkt
- gumfdat - Umfelddaten "Geht"-Zeitpunkt
- klon - Geokoord.: Longitude (Kommt-Zeitpunkt)
- klat - Geokoord.: Latitude (Kommt-Zeitpunkt)
- glon - Geokoord.: Longitude (Geht-Zeitpunkt)
- glat - Geokoord.: Latitude (Geht-Zeitpunkt)
- version - Software Version der Lokomotive
- lasterase - Letzter Löschzeitpunkt der Lokomotiv-Importdatei
- umfeld - Decodierte Umfelddaten des "Kommt"-Zeitpunktes als JSON-String

Die Umfeld-Daten des "Kommt"-Zeitpunktes werden beim Importvorgang decodiert und sind direkt einsehbar. Die Geokoordinaten sind oftmals nicht vorhanden (kein GPS-Empfang). Die Laufende Nummer zählt hoch, wie oft ein Code innerhalb einer Import-Datei auftritt. Sie wird aber nicht weiter genutzt, ähnlich der Werte „TKMW“ und „Stat“.
Anhand der Software-Version wird entschieden, wie die Umfeld-Daten decodiert werden müssen.

Tabelle „eventcountdaily“

In dieser Tabelle wird für jede Lokomotive zu jedem Diagnosecode die Auftrittshäufigkeit für alle einzelnen Tage berechnet. Sie dient als Hilfstabelle für spätere Berechnungen, die auf diesen Häufigkeiten basieren (Versuchsreihe 01). Die Tage sind hier nummeriert, beginnend vom 01.06.2012 aufwärts (Tag „0“).

- id – ID des Datensatzes
- time – *Timestamp* des zugehörigen Tages
- day – Nummer des zugehörigen Tages (beginnend am 01.06.2012)
- baunr – Bezeichnung der Lokomotive
- code – Code-Nummer
- amount – Auftrittshäufigkeit des Codes an diesem Tag

Tabelle „stoerfile“

Diese Tabelle beinhaltet alle existierenden Fehlercodes, zugeordnet zur jeweiligen Software-Version der 185er Baureihe. Anhand dieser Tabelle wird beim Importvorgang die zugehörige Dekodierungsanweisung für die Umfelddaten ermittelt.

Tabelle „fehlercodes“

In dieser Tabelle sind alle vorhandenen Fehlercodes der 185er Baureihe mit ihrer Erklärung und den zugehörigen Systemen und Subsystemen aufgeführt. Es ist eine simplifizierte Variante der Tabelle „stoerfile“, die die Softwareversion nicht beachtet. Da die Codebeschreibungen durch die Softwareversionen konsistent sind, stellt dies kein Problem dar. Diese Tabelle dient zur einfacheren Ermittlung der Systeme und Subsysteme, ohne dass die wesentlich größere Tabelle „stoerfile“ benutzt werden muss.

Tabelle „wartungen“

Alle Wartungszeitpunkte im Zeitraum Juni 2012 bis August 2013 der Baureihe 185 werden in diese Tabelle importiert.

***SQL-View* „baunr“**

Enthält alle distinkten Fahrzeugnummern der Baureihe 185.

***SQL-View* „day“**

Enthält alle Tage, an denen Einträge zu mindestens einer Lokomotive der Baureihe 185 vorhanden sind.

***SQL-View* „baunrdays“**

Enthält für alle Fahrzeugnummern der Baureihe 185 alle Tage, an denen Einträge für dieses Fahrzeug vorhanden sind. Dies dient bei der Auswertung dazu, herauszufinden, ob an einem Tag Aufzeichnungslücken für eine Lokomotive vorhanden sind.

***SQL-View* „dailyeventcount“**

Hier werden die Gesamtevents je Tag und Code – unabhängig von der Lokomotive – aufsummiert (Baureihe 185).

***SQL-View* „eventstotalperday“**

Enthält für jede Fahrzeugnummer der Baureihe 185 und für jeden Tag die Summe aller Einträge, die für dieses Fahrzeug vorhanden sind.

Tabelle „rawdata189“

Diese Tabelle enthält die Rohdaten der 189er Baureihe, die für die Versuchsreihe 2 benötigt werden.

- Id: ID des Datensatzes
- Baunr: Fahrzeugnummer der Lokomotive
- Fcode: Code-Nummer des Diagnosecodes
- Kdatum: Kommt-Datum des Fehlers
- FM1: Temperatur Fahrmotor 1
- FM2: Temperatur Fahrmotor 2
- FM3: Temperatur Fahrmotor 3
- FM4: Temperatur Fahrmotor 4
- LFM1: Temperatur Lager 1
- LFM2: Temperatur Lager 2
- LFM3: Temperatur Lager 3
- LFM4: Temperatur Lager 4
- Day: Nummer des Tages

***SQL-View* „189seasonalcorrection“**

Errechnet die Durchschnittstemperatur je Kalendermonat, und den Korrekturfaktor zur absoluten Durchschnittstemperatur. Dies wird genutzt, um später saisonal bereinigte Temperaturen errechnen zu können.

Tabelle „189reduced“

Beinhaltet die idealisierten Fahrmotordaten (nur ein Fahrmotor), die aus den Rohdaten errechnet wurden. Dabei sind die Datensätze, die im Kaltbetrieb aufgezeichnet wurden, herausgefiltert.

***SQL-View* „189reduced_deseason“**

Beinhaltet die gleichen Daten wie „189reduced“ in saisonal korrigierter Form.

Tabelle „189reduced_filtered_deseason“

In dieser Tabelle sind zusätzlich die Outlier aus „189reduced_deseason“ ausgefiltert.

Tabelle 189labeled

Beinhaltet die gelabelten Daten der verschiedenen Versuchsaufbauten. Von hier aus werden die Daten in ein ARFF-File geschrieben.

7.2. Java *Packages*

Im Folgenden werden die verschiedenen Java-Packages kurz beschrieben. Dies soll einen groben Überblick über den vorhandenen Programmcode darstellen. Eine exakte Funktionsbeschreibung ergibt sich aus der Javadoc-Dokumentation im Programmcode selbst.

7.2.1. *Package* „de.dbschenker"

Dieses Package enthält alle Haupt-Klassen, die zur Ausführung der verschiedenen implementierten Programme dienen.

Programm „CreateDailyAmounts.java"

Gehört zu Versuchsreihe 1. Dieses Programm dient dazu, die Berechnung der Tagessummen anhand der Rohdaten zu starten. Es geht davon aus das die „rawdata" Tabelle sinnvoll befüllt ist, leert die vorhandene „eventcountdaily" Tabelle und befüllt diese erneut.

Programm „TechLokConverter"

Dieses Programm dient bei einer Voruntersuchung dazu, eine CSV-Datei in eine geeignete ARFF-Datei zu konvertieren.

Programm „Versuchsreihe01CountCodeSums"

In diesem Programm werden die ARFF-Dateien für die Versuchsreihe 1 erzeugt. Es läuft sehr lange und benötigt viel freies RAM, 8GB werden empfohlen.

Programm „Versuchsreihe02FMunification"

Berechnet den idealisierten Datensatz aus den Rohdaten der 189er Baureihe und speichert alle Instanzen in Tabelle „189reduced"

Programm „Versuchsreihe02Filtering"

Führ die Outlier-Filterung durch und speichert die gefilterten Werte in „189reduced_filtered_deseason"

Programm „Versuchsreihe02Zentralschraubenbruch"

Berechnet, labelt und speichert die ARFF-Dateien die zur Evaluation der Versuche benötigt werden.

7.2.2. *Package* „de.dbschenker.arffconverter"

In diesem Package befinden sich diverse Klassen zur Verarbeitung von CSV-Dateien oder anderen Quellen in ARFF-Dateien. Durch Erweiterungen der Klassen ist es möglich, die Attributtypen der Quelle anzupassen und ein gewisses Maß an Filterung und Preprocessing zu betreiben.

7.2.3. *Package* „de.dbschenker.database"

Enthält eine Klasse, die die Verbindung zum Datenbankserver herstellt.

7.2.4. *Package* „de.dbschenker.jobs“

Enthält weitere Klassen, die für die einzelnen Versuchsreihen Berechnungen, Labeling und Erzeugung der ARFF-Dateien ausführen.

7.2.5. *Package* „de.dbschenker.utils“

Enthält Utility Klassen und Datentypen, die von den verschiedenen Jobs benötigt werden.

7.3. PHP Scripte

PHP-Scripte wurden zum Importieren der zur Verfügung gestellten Exporte in die *MySQL*-Datenbank genutzt. Da diese Importe einmalig ausgeführt werden ist die Performance nicht ausschlaggebend und können einige Zeit in Anspruch nehmen.

Script „umfelddatenmethoden.php“

In diesem Script sind einige Hilfsfunktionen lokalisiert, die zur Decodierung der Umfelddaten der Baureihe 185 dienen. Wird von den anderen Scripts benötigt.
Die Decodierungs-Informationen für die jeweiligen Software-Versionen werden vorausgesetzt. Sie müssen im Unterordner „umfelddaten“ unter folgenden Dateinamen vorhanden sein:

- umfeld40.csv
- umfeld60.csv
- umfeld295.csv

Das Mapping, für welche Software-Version welche Decodierungs-Anweisungen benutzt werden, befindet sich im Programmcode.

Script „importstoerfile.php“

Dieses Script importiert die Dateien mit allen Code-Informationen in die Datenbank. Ein Mapping, für welche Software-Version welche Datei importiert wird befindet sich im Programmcode. Betrifft Baureihe 185.

Script „importallfiles.php“

Dieses Script importiert alle Dateien der 185er Baureihe. Die Dateien müssen im Unterordner „importdaten185“ vorliegen.

Script „decodeAllUmfelddaten.php“

Decodiert alle Umfelddaten in der Tabelle „rawdata“. Die Umfelddaten werden nicht direkt beim Import decodiert.

Script „importall189.php“

Importiert die Rohdaten der 189er Baureihe aus dem Unterverzeichnis „importdaten189“.

8. Glossar

1R - Regelbasierter Lernalgorithmus.

ASG – Antriebssteuergerät.

AUC - *Area Under Curve*. Der Flächeninhalt der sich unterhalb der ROC-Kurve befindet. Der bestmögliche Wert ist eins.

Bagging - Meta-Klassifizierer, der die Ergebnisse mehrerer einfacher Klassifizierer kombiniert.

Boosting - Meta-Klassifizierer, der die Ergebnisse mehrerer einfacher Klassifizierer kombiniert.

C4.5 - Baumbasierter Lernalgorithmus.

ESM - Elektronische Schadmeldung. Sofware-System zur Erfassung von Schäden an der Lokomotive. Bildet einen kompletten *Workflow* bis zur Erledigung durch die Werkstatt ab.

False Negative - Ein vorhandenes Signal wird fälschlicherweise nicht erkannt.

False Positive - Ein nichtvorhandenes Signal wird fälschlicherweise als Signal erkannt.

FM – Fahrmotor.

Fold - Teilmenge eines Datensatzes.

GPL - *Gnu Public License*. Software-Lizenz für *Open-Source Software*.

GPS - *Global Positioning System*. Ein System zur globalen Lokalisierung basierend auf Satelliten. Die Genauigkeit liegt bei ungefähr 10 Meter.

Greedy-Algorithmus - Ein Verfahren in der Informatik, bei ein Vorgehen Schrittweise ausgeführt wird, indem in jedem Schritt der bestmögliche Folgeschritt ausgewählt wird.

Hit - siehe *True Positive*.

ID3 - Iterative Dichotomiser 3 (Lernalgorithmus) **.**

J48 - In Java implementierte Variante von C4.5.

JRip - In Java implementierte Variante von *RIPPER*.

JSON - JavaScript *Object Notation*. Menschenlesbares, kompaktes Datenformat.

LFM - Abkürzung: Lager Fahrmotor.

LZB - Linienförmige Zugbeeinflussung. Dieses Zugbeeinflussungssystem ermöglicht eine genaue Lokalisierung des Zuges und sorgt unter anderem dafür, dass Züge schneller als 160 km/h und in geringerem Abstand zueinander fahren können. Es ergänzt die an den Gleisen vorhandenen Signale der Zugsteuerung.

MFA - Multi-Funktions-Anzeige. Ein Universaldisplay im Führerstand der Lok, welches Betriebsinformationen, Meldungen und Fehler anzeigt.

Miss - siehe *False Negative*.

MySQL - Frei verfügbare SQL-Datenbank.

Overfitting - Spezifische Anpassung eines Klassifizierers unter Vernachlässigung der Allgemeinheit.

PHP - *PHP Hypertext Preprocessor*. Programmiersprache, die hauptsächlich in der Webentwicklung verwendet wird.

PZB - Punktförmige Zugbeeinflussung. Ein maschinell gesteuertes Zugbeeinflussungssystem. Seine Hauptaufgabe ist die Verhinderung der Vorbeifahrt an Haltesignalen.

RandomForest - Baumbasierte Lernalgorithmus, der mehrere *RandomTree* Instanzen kombiniert.

RandomTree - Baumbasierter Lernalgorithmus.

RIPPER - Regelbasierter Lernalgorithmus.

ROC - Receiver Operator Charakteristik. Eine Methode der Signaltheorie, die *False Positives* und *True Positives* miteinander in Zusammenhang setzt, und daraus eine Kurve für die Empfindlichkeit eines Signalinterpreters erzeugt.

SMO - *Sequential Minimum Optimization*. Klassifizierer der auf dem Support-Vector Maschinen Prinzip funktioniert.

SQL - *Structured Query Language*, Datenbanksprache bei relationalen Datenbanken.

SVM - *Support Vector Machine*.

True Positive - Ein vorhandenes Signal wird korrekt erkannt.

True Negative - Ein nichtvorhandenes Signal wird als nichtvorhanden erkannt.

Underfitting - Unzureichend genaue Anpassung eines Klassifizierers.

WEKA - Waikato Environment for Knowledge Analysis. Softwarepaket der Universität Waikato.

XML - *Extensible Markup Language*. Auszeichnungssprache für hierarchische Datenstrukturen.

ZSG - Zentrales Steuergerät.

9. Literaturverzeichnis

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* New York: Oxford University Press, Inc.

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*, (S. 123-140).

Breiman, L. (2001). Random Forests. *Machine Learning 45* (S. 5-32). Kluwer Academic Publishers.

Christianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press.

Cohen, W. W. (1995). Fast Effective Rule Induction. *Twelfth International Conference on Machine Learning*, (S. 115-123).

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning*, (S. 148-156). San Francisco.

Fulp, E. W., Fink, G. A., & Haack, J. N. (2008). Predicting Computer System Failures Using Support Vector Machines. *WASL'08 Proceedings of the First USENIX conference on Analysis of system logs*.

Fürnkranz, J. (1997). Pruning Algorithms for Rule Learning. *Machine Learning 27*, 139-172.

Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review 13*, 3-54.

Guo, P., & Bai, N. (2011). Wind Turbine Gearbox Condition Monitoring with AAKR and. *Energies 2011, 4*, 2077 - 2093.

Holte, R. C. (1993). Very simple classification rules perfom well on most commonly used datasets. *Machine Learning 11*, S. 63-91.

Létourneau, S., Famili, F., & Matwin, S. (12 1999). Data Mining For Prediction of Aircraft Component Replacement. *IEEE Intelligent Systems Jr., Special Issue on Data Mining*, S. 59-66.

Lipowsky, H. (2010). *Entwicklung und Demonstration eines integrierten Systems zur Zustandüberwachung von Gasturbinen.* Stuttgart: Institut für Luftfahrtantriebe.

Platt, J. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* Microsoft Research.

Quinlan, R. (1986). Induction of decision trees. *Machine Learning 1*, (S. 81-106).

Quinlan, R. (1993). *Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann Publishers.

Russel, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach, 3rd Edition.* Upper Saddle River, NJ, USA: Prentice Hall Press.

Salfner, F., Lenk, M., & Malek, M. (01. 03 2010). A Survey of Online Failure Prediction Methods. *ACM Computing Surveys (CSUR) Article 10*.

Schölkopf, B., & Smolar, A. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* Cambridge, MA, USA: The MIT Press.

Swets, J. (1964). *Signal detection and recognition by human observers.* New York: Wiley.

Vapnik, V. N., & Chervonenkis, A. J. (1974). *Theory of Pattern Recognition.*

Witten, I. H., Frank, E., & Hall, M. (2011). *Data Mining - Practical Machine Learning Tools and Techniques.* Burlington: Morgen Kaufmann Publishers.

10. Anhänge

Alle Anhänge befinden sich auf der beiliegenden CD. Sie beinhaltet alle in diesem Dokument referenzierten Anhänge, die vollständigen Quellcodes und die Datensätze der Versuchsreihen.