Die Auswirkung der Reihenfolge von Mess- und Simulationsdaten auf das Ergebnis der Kreuzvalidierung in KDD Prozessen

Christof Küstner, Thilo Breitsprecher und Sandro Wartzack
Lehrstuhl für Konstruktionstechnik;
Universität Erlangen Nürnberg

1 Einleitung

Die Wissensentdeckung in Datenbanken (engl. Knowledge Discovery in Databases; KDD) stellt einen rechnerunterstützten Prozess dar, Wissen aus komplexen Daten zu extrahieren [2]. Durch den verstärkten Einsatz rechnerunterstützter Methoden und Werkzeuge (z. B. FEM, MKS, CFD, usw.) in frühen Phasen des Produktentwicklungsprozesses (PEP), steigt das Volumen an entwicklungsrelevanten Daten im PEP stetig an. Sowohl Messdaten als auch Simulationsergebnisse enthalten konstruktionsrelevantes Wissen, auf das der Produktentwickler ohne Rechnerunterstützung nicht in vollem Umfang zugreifen kann. Hierdurch bleibt in der Entwicklung ein Potential unangetastet, das eigentlich beinahe kostenneutral zur Verfügung steht. In einer Zeit, in der Wissen zu einer der wichtigsten Produktionsfaktoren zählt, kann dieses nicht ausgeschöpfte Potential zu einem Wettbewerbsnachteil führen [5].

Die KDD Methodik kann u. a. dazu eingesetzt werden, den Informationsgewinn von Simulationen bzw. Simulationsstudien für komplexe Produkte zu erhöhen. Ebenso kann durch den gezielten Einsatz die Anzahl von Simulationsiterationen reduziert werden. KELLERMEYER, WALTER und BREITSPRECHER zeigen diesen Effekt erfolgreich [1,3,7]. Sie erzeugen mittels des KDD Prozesses Vorhersagemodelle, um mit diesem entwicklungsrelevantem Wissen das Pro-

duktverhalten abzuschätzen. Zur Erzeugung der Vorhersagemodelle setzt KDD im Kern maschinelles Lernen ein. Hierbei ist die Kenntnis der Performanz des Vorhersagemodells ein wichtiges Qualitätsmerkmal, da dieses sehr anschaulich als „Wissensgüte" verglichen werden kann. Die Performanz definiert die Vorhersagegüte bzw. –genauigkeit des Vorhersagemodells und bildet dadurch z. B. die Basis des Vergleichs von verschiedenen Vorhersagemodellen. Die Performanz muss aus diesem Grund dem Anspruch der Robustheit gerecht werden um als Vergleichsbasis zu dienen, d. h. die Performanz muss eine von allen Einflussgrößen unabhängige Größe darstellen.

Diese Vorgehensweise findet auch im intereuropäischen Verbundprojekt EUREKA-ALARM, bestehend aus REpower, Novicos, ESM, ZF Wind Power, LMS, KU Leuven und KTmfk, eine Anwendung. ALARM ist ein Akronym und steht für Assistenzsystem für die lärmreduzierte Auslegung rotierender Maschinen. Am Beispiel einer modernen Windenergieanlage soll der Entwicklungsingenieur durch kontextsensitive Wissensbereitstellung von entwicklungsrelevantem Wissen unterstützt werden, so dass das Optimierungspotential im Hinblick auf das akustische Verhalten schon in frühen Phasen des PEP erkannt wird (siehe Bild 1) [4].

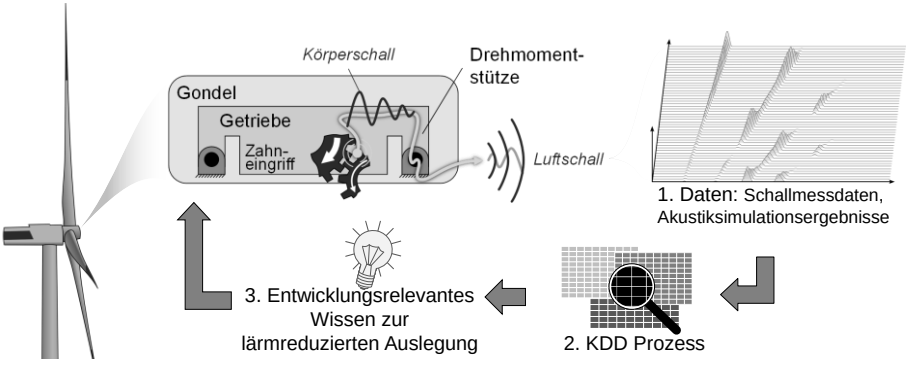

Bild 1: Wissensakquisitionsszenario im Projekt EUREKA-ALARM

Das entwicklungsrelevante Wissen wird im Projekt EUREKA-ALARM durch KDD aus Schallmessungen und Akustiksimulationsergebnissen erhoben. Hierbei werden u. a. Produktmerkmale moderner Windenergieanlagen deren Schallemission gegenüber gestellt. Die vorgestellte Problemstellung ist im Verlauf des Projektes EUREKA-ALARM bei der Wissensakquisition mit dem KDD Werkzeugs RapidMiner® aus Schallemissionsmessungen und Simulati-

onsergebnissen aufgetreten. Es zeigte sich, dass diese Problemstellung ein grundlegendes Problem von KDD Prozessen zur Wissensakquisition ist.

2 Grundlagen

Bevor auf die Problemstellung näher eingegangen wird, werden wichtige Grundlagen, die zum Verständnis der Problematik beitragen, erläutert.

2.1 Der KDD Prozess

Ein Teilschritt im KDD Prozess bedarf der Verwendung von Data Mining Methoden. Nach FAYYAD ist Data Mining die Anwendung von speziellen Algorithmen aus dem Bereich des maschinellen Lernens, um Muster oder Zusammenhänge in Daten, z. B. aus Akustiksimulationen oder experimentellen Messungen von modernen Windenergieanlagen, aufzudecken [2]. Hierbei können *Beschreibungsprobleme* (Clusteranalysen) oder *Prognoseprobleme* (Klassifkation oder Regression) vorliegen. Beschreibungsprobleme teilen die vorliegenden Eingangsdaten so in Gruppen (Cluster) ein, dass die Datentupel eines Clusters möglichst ähnlich sind. Bei Prognoseproblemen wird basierend auf einem Eingangsdatensatz, bestehend aus m Simulationen oder experimentellen Beobachtungen (Datentupel), ein Vorhersagemodell für neue, nicht im Eingangsdatensatz enthaltene Tupel aus Attributen und Zielgrößen erstellt (siehe Bild 2). Die Problemstellung ist somit induktiver Natur. Das Vorhersagemodell ist in jedem Fall nur eine bestmögliche Annäherung an das wirkliche Systemverhalten und verlangt die Angabe einer Performanz (Vorhersagegüte), wie gut die Zielgröße durch das Vorhersagemodell abgebildet ist. Das Vorhersagemodell kann in Form von Regressionsgleichungen, Modellbäumen, künstlichen neuronalen Netzen, Regeln oder Mischformen (z. B. Regressionsbäume) abgespeichert sein [8].

	Attribut 1	Attribut 2	Attribut 3	…	Zielgröße 1	…
Datentupel 1:	5	3,2	4	…	2,3	…
Datentupel 2:	8	5,6	2	…	6,5	…
…	…	…	…	…	…	…
Datentupel m:	1	6,67	150	…	10,8	…
	4	10	2	…	?	?

Prognose

Bild 2: Darstellung eines Eingangsdatensatzes und eines Prognoseproblems

2.2 Validation der Vorhersagemodelle

Die Performanz eines Vorhersagemodells beschreibt die Vorhersagegüte bzw. -genauigkeit des Vorhersagemodells. Die Performanz eines Vorhersagemodells kann durch spezielles Testen des erzeugten bzw. trainierten Vorhersagemodells abgeschätzt werden. Hierbei wird beim Trainieren des Vorhersagemodells gezielt ein Teil der Datentupel dem Trainingsprozess vorenthalten, die im Anschluss als Testdaten verwendet werden. Nachdem das Vorhersagemodell mit den Trainingsdaten trainiert wurde, werden für jeden Testdatentupel die Attribute in das Vorhersagemodell eingesetzt und die vorhergesagten Zielgrößen mit den bekannten Zielgrößen der Testdatentupel verglichen. Die Performanz sagt aus, wie gut das Vorhersagemodell die Zielgrößen der Testdatentupel auf Basis der Trainingsdaten vorhersagen kann. Diese so ermittelte Performanz stellt eine Abschätzung der Prognosefähigkeit der betrachteten Zielgrößen für unbekannte Datentupel dar.

Ziel eines KDD Prozesses ist es ein Vorhersagemodell zu generieren, das einerseits die Eingangsdaten und andererseits zukünftige Beobachtungen gut abbildet. Prognostiziert ein Vorhersagemodell die Eingangsdaten sehr gut, zeigt jedoch bei neuen Daten bzw. Beobachtungen eine schlechte Performanz, wird dies *Overfitting* genannt (dt. Überanpassung des Vorhersagemodells auf die Eingangsdaten). Beschreibt das Vorhersagemodell die Eingangsdaten zu ungenau, wird dies *Underfitting* genannt (dt. Unteranpassung des Vorhersagemodells). Das Underfitting zeigt sich ebenfalls in einer schlechten Performanz.

Zur Vermeidung des Overfittings werden verschiedene Validationsmethoden eingesetzt [8]:

- Cross-validation (dt. Kreuzvalidierung oder n-fache Kreuzvalidierung)
- Leave-One-Out Cross-Validation (Kreuzvalidierung mit Anzahl Datentupel gleich Anzahl der Gruppen n in der Kreuzvalidierung)
- The Bootstrap (dt. Stiefelriemen; Testdateneinteilung durch Ziehen mit Zurücklegen)

Die verbreitetste und anerkannteste Validationsmethode stellt die Kreuzvalidierung dar [6]. Die n-fache Kreuzvalidierung teilt alle Datentupel des Eingangsdatensatzes zufällig in n gleich große Gruppen. Danach verwendet die Kreuzvalidierung alle Gruppen bis auf einen für das Training $\left(\frac{n-1}{n}\right)$ und

die verbleibende Gruppe $\left(\frac{1}{n}\right)$ für die Performanzberechnung. Der Training-Test-Prozess wird n-Mal durchgeführt, wobei bei jeder Iteration sukzessive eine andere Gruppe ausgelassen und zur Berechnung der Performanz verwendet wird (engl. folds). Für n wird sehr oft ein Wert von 10 angenommen, da dieser sich als bewährter Kompromiss zwischen Rechenzeit und statistischer Genauigkeit gezeigt hat [8].

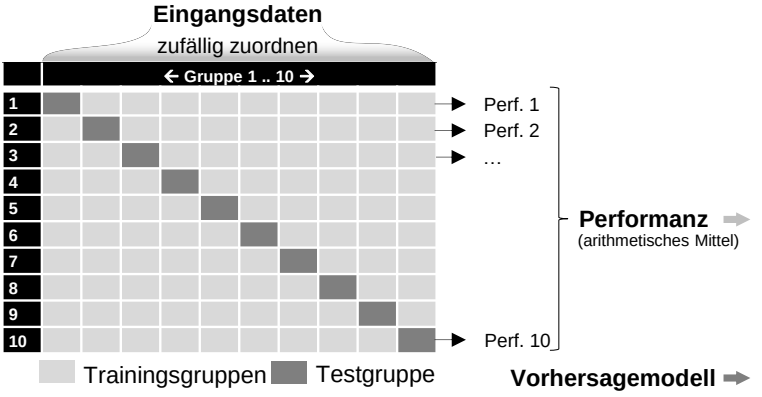

Bild 3: Die 10-fache Kreuzvalidierung

Bei $= 10$ möglichst gleich großen, zufällig aufgeteilten Gruppen wird in der ersten Iteration Gruppe $1 - 9$ für das Training und Gruppe 10 für das Testen verwendet. In der zweiten Iteration wird Gruppe $1 - 8$ und Gruppe 10 für das Training, die Gruppe 9 für das Testen verwendet, u. s. w.. Sind die 10 Iterationen durchlaufen wird aus den 10 Performanzen das arithmetische Mittel gebildet. Ein $n + 1$ ter (hier: 11 ter) Durchlauf trainiert erneut das Vorhersagemodell, diesmal werden jedoch alle 10 Gruppen berücksichtigt, um alle Datentupel für die Modellbildung einzuschließen (siehe Bild 3). Hierdurch bleiben keine zur Verfügung stehende Informationen unberücksichtigt. Das Ergebnis einer Kreuzvalidierung ist schließlich das Vorhersagemodell, das aus allen Eingangsdaten gebildet wurde, sowie das arithmetische Mittel aus allen 10 Performanzen. Dieser Kennwert stellt einen guten Schätzer der Performanz des Vorhersagemodells für neue Daten dar.

2.3 Rechnererzeugte Zufallszahlen

Der Computer ist nicht in der Lage echte Zufallszahlen zu generieren. Jede Programmiersprache stellt Algorithmen zur Verfügung Zufallszahlen zu erzeugen. Der Algorithmus errechnet auf Basis eines Startwertes, dem sog.

Randomseed (dt. Zufallssaat) die Zufallszahl. Ist der Randomseed von zwei unabhängigen Zufallszahlenberechnungen gleich, so sind die erzeugten Zufallszahlen in Reihenfolge und Quantität identisch. Aus diesem Grund wird auch von Pseudozufallszahlen gesprochen. Um bessere Zufallszahlen zu erzeugen, muss der Randomseed stetig verändert werden, so dass nicht direkt nachvollzogen werden kann, welchen Wert der Randomseed zu Beginn der Zufallszahlberechnung hat. Die Zeit bietet sich hier als geeignete Basis zur Zufallszahlenberechnung an. Im Computer steht als Zeitmaß zum Beispiel der UNIX-Timestamp (dt. Unixzeit) zur Verfügung. Dieser repräsentiert die vergangenen Sekunden als ganze Zahl seit dem 01.01.1970 00:00 Uhr UTC (siehe Bild 4). Eine zweite Möglichkeit stellt die Uptime (dt. Betriebszeit) als Referenzzeit dar, die die vergangene Zeit seit dem letzten Reboot des Computers zählt. Wird diese Zeit als Randomseed verwendet, ändert sich die Basis vor jeder relevanten Zufallszahlenberechnung. Auf diese Weise besitzt die Zufallszahlenberechnung stets eine unterschiedliche „Ausgangsbasis", wodurch eine schwer vorhersagbare Zufallszahl berechnet wird.

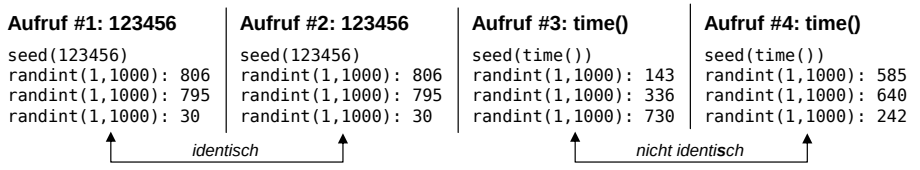

Bild 4: Auswirkungen des Randomseeds auf die Zufallszahlenberechnung

3 Problemstellung

Ziel des KDD Prozesses besteht u. a. darin, eine möglichst robuste Aussage über die Performanz des berechneten Vorhersagemodells zu treffen. Eine Performanz kann nur berechnet werden, wenn ein Vorhersagemodell mit Datentupel getestet wird, die zum Lernen nicht zur Verfügung standen. Aus diesem Grund wird die Kreuzvalidierung eingesetzt (siehe Kapitel 2.2). Die 10-fache Kreuzvalidierung ordnet hierbei die einzelnen Datentupel des Eingangsdatensatzes *zufällig* in zehn möglichst gleich große Gruppen. Hierbei muss der Aspekt der Zufallszahlengenerierung beachtet werden, die bei der zufälligen Zuordnung der Datentupel von Relevanz ist: Wird der Randomseed zu Beginn des Prozesses auf einen konstanten Wert festgelegt, sind alle nachfolgenden generierten Zufallszahlen stets reproduzierbar, indem der Randomseed erneut auf diesen Wert gesetzt wird (siehe Bild 4; Aufruf #1 und #2). Die so generierten Zufallszahlen sind in Quantität und Reihenfolge identisch.

Wird ein konstanter Randomseed zur zufälligen Verteilung der Datentupel in zehn gleich große Gruppen verwendet, so wird jedes Datentupel im ersten Durchlauf der Kreuzvalidierung einer echt zufälligen Gruppe zugeordnet, in den folgenden Durchläufen ist die Aufteilung jedoch bekannt und somit nicht mehr zufällig, d. h. durch jeden Neustart der Kreuzvalidierung wird der Randomseed auf den vorherigen, konstanten Wert zurückgesetzt. Die Verteilung der einzelnen Datentupel in Gruppen erfolgt in jedem Durchlauf somit nach demselben Zuordnungsschlüssel.

In Bild 5 ist ein einfacher KDD Prozess im KDD Werkzeug RapidMiner® modelliert. Zuerst werden die Daten mittels einem Datenimportoperator in den Prozess geladen und anschließend die Performanz in einem Kreuzvalidierungsoperator berechnet. Im vorliegenden Fall handelt es sich bei der Performanz um den relativen Fehler zwischen vorhergesagter und realer Zielgröße als leicht zu interpretierenden Performanzkennwert. Je kleiner der relative Fehler ist, desto besser ist die Performanz. Zudem muss im KDD Werkzeug RapidMiner® zwischen globalem und lokalem Randomseed differenziert werden. Der globale Randomseed wird von den Operatoren verwendet, in denen nicht explizit ein lokaler Randomseed gesetzt wird. Standmäßig ist der globale Randomseed im KDD Werkzeug RapidMiner® auf den konstanten Wert 2001 gesetzt. In den Operatoren, die Zufallszahlen benötigen ist standardmäßig kein lokaler Randomseed gesetzt, weshalb diese auf den globalen Randomseed zurückgreifen.

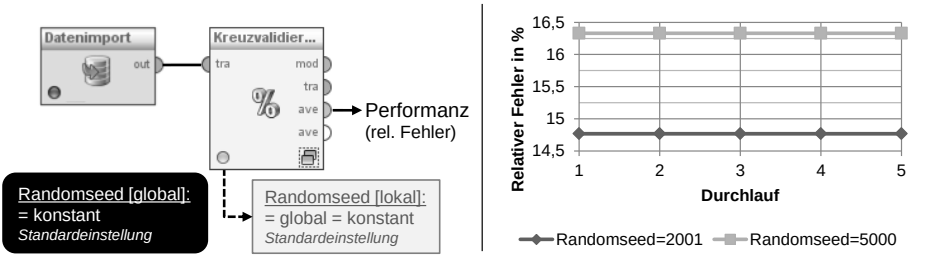

Bild 5: Ein einfacher KDD Prozess mit konstantem Randomseed

Das Bild 5 zeigt, dass nach jedem Neustart des KDD Prozesses der Performanzwert identisch zum vorherigen Durchlauf ist (Fall: „Randomseed=2001" in Bild 5). Im ersten Moment wirkt die Performanz somit robust, denn der Performanzwert ist reproduzierbar. Wird der globale Randomseed nun verändert (Fall: „Randomseed =5000" in Bild 5), so ist zu erkennen, dass sich die Performanz auch verändert. Die naheliegende Lösung zur Berechnung einer robusten Performanz wäre, den Randomseed global

konstant zu halten, was sich als fatal zeigt, wenn sich die Auswirkung der Veränderung des globalen Randomseeds bewusst gemacht wird: Die Veränderung des Randomseeds verändert die Verteilung der Datentupel in Test- und Trainingsgruppen. Der gleiche Effekt könnte erzielt werden, wenn die Reihenfolge der Eingangsdaten bei bekanntem Zuordnungsschlüssel verändert wird, denn auch dann würden die einzelnen Datentupel unterschiedlichen Test- und Trainingsgruppen zugeordnet. Dieser Zusammenhang erklärt die unterschiedliche Performanz, wenn sich der lokale Randomseed des Kreuzvalidierungsoperators verändert, oder wenn bei gleichem Randomseed die Reihenfolge der Eingangsdaten verändert wird. Der letztere Fall tritt z. B. in Erscheinung, wenn neue Akustiksimulationsergebnisse oder Schallmessungen den Eingangsdatensatz um ein oder mehrere Datentupel erweitern, oder diese z. B. durch Umsortieren neu angeordnet werden.

Bei der Veränderung der Eingangsdaten oder des globalen Randomseeds ist somit eine hohe Varianz der Performanz zu erwarten. Generell beschreibt die Literatur, dass eine Kreuzvalidierung statistisch abgesichert werden soll. Zur statistischen Absicherung wird eine zehnfach ausgeführte 10-fache Kreuzvalidierung mit anschließender Mittelwertbildung empfohlen [8].

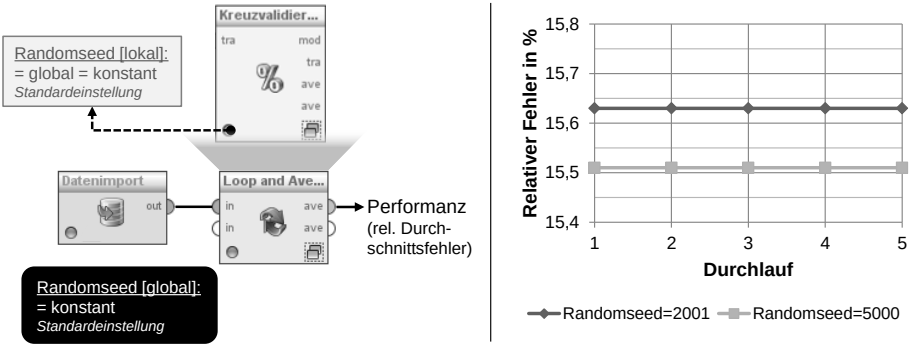

Bild 6: Zehnfache 10-fach Kreuzvalidierung mit konstantem Randomseed

In Bild 6 wird der einfache KDD Prozess aus Bild 5 um den „Loop and Average" Operator erweitert. Dieser ermöglicht die Wiederholung (loop) aller inneren Subprozesse und die anschließende Ausgabe des arithmetischen Mittels (average) eines beliebigen im Innern erzeugten Wertes. Er garantiert somit die statistische Absicherung der Kreuzvalidierung, indem er die Kreuzvalidierung mehrfach ausgeführt und anschließend aus den Einzelperformanzen ein arithmetisches Mittel bildet und ausgibt.

Auch in diesem KDD Prozess ist nach jedem Neustart der Performanzwert identisch zum vorherigen Durchlauf (Fall: „Randomseed=2001" in Bild 6). Ebenso verändert sich die Performanz, falls der globale Randomseed verändert wird (Fall: „Randomseed=5000" in Bild 6). Es kann also davon gesprochen werden, dass die Reihenfolge der Eingangsdaten einen Einfluss auf das Ergebnis der Kreuzvalidierung in KDD Prozessen hat. Im Folgenden Kapitel wird für diese Problematik ein Lösungsansatz diskutiert.

4 Lösungsansatz

Möchte der spätere Anwender von Vorhersagemodellen deren Performanz vergleichen, muss er sich auf eine statistisch robuste und dadurch reproduzierbare Kreuzvalidierung verlassen. Wie im Kapitel 3 gezeigt wurde, beeinflusst die Reihenfolge von Eingangsdaten die Performanz als ein Ergebnis des KDD Prozesses jedoch nachhaltig. Für eine robuste Kreuzvalidierung muss die dabei durchgeführte Aufteilung des Eingangsdatensatzes statistisch abgesichert werden. Da der Eingangsdatensatz an sich nicht verändert werden kann und eine Durchmischung der einzelnen Datentupel vor dem KDD Prozess (außerhalb des KDD Werkzeugs) sehr ineffektiv ist, muss durch einen günstig gewählten Randomseed die benötigte statistische Sicherheit erreicht werden. Die Festlegung des Randomseed kann für einen einfachen Kreuzvalidierungsprozess innerhalb des KDD Werkzeugs RapidMiner® an verschiedenen Stellen erfolgen (siehe Kapitel 3).

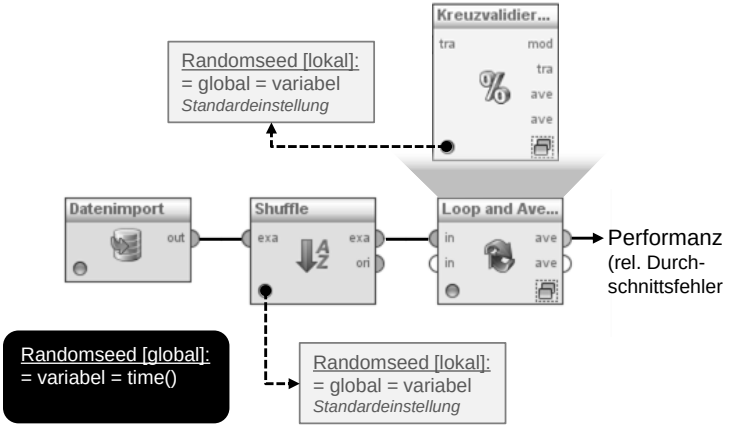

Bild 7: Um variable Randomseeds erweiterter KDD Prozess.

Die in diesem Beitrag vorgeschlagene Lösung erweitert den im Kapitel 3 gezeigten KDD Prozess durch gezielten Einsatz von variablen Randomseeds

(siehe Bild 7). Der globale Randomseed wird durch die Systemzeit bestimmt, wodurch sich bei jedem Neustart des Prozesses andere Pseudozufallszahlen ergeben. Nach dem Datenimport werden die Eingangsdaten durchmischt (Shuffle), um den Einfluss einer veränderten Datentupelreihenfolge abzubilden. Das Durchmischen greift auf den globalen Randomseed zurück und erzeugt somit bei jedem Durchlauf eine andere Anordnung der Datentupel in den Eingangsdaten. Innerhalb des „Loop and Average" Operators wird eine 10-fache Kreuzvalidierung durchlaufen. Auch die Kreuzvalidierung greift für die Einteilung in Test- und Trainingsdaten auf den globalen Randomseed des KDD Prozesses zurück.

Zur Verdeutlichung der Wirksamkeit des Lösungsansatzes wird der gezeigte KDD Prozess mehrmals durchlaufen und vor jedem Neustart die Anzahl der zu durchlaufenden Loops auf 1, 2, 3, 4, 5, 10, 20, 50, 100 und 1000 gesetzt (entspricht zehn Variationen). Um die statistische Auswertung anschaulicher zu gestalten wird der Prozess mit jeder Loop Einstellung 25 Mal wiederholt. Dadurch ergeben sich insgesamt 250 unterschiedliche Performanzen.

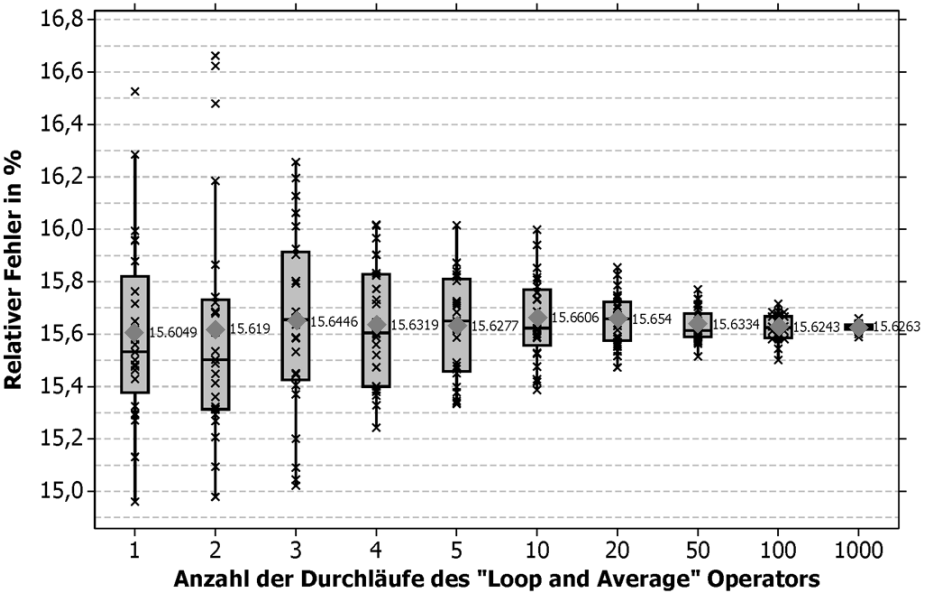

Bild 8: Boxplot der Performanzen mit steigender Anzahl von Loopdurchläufen

Für die statistische Auswertung wird auf einen sogenannten Boxplot zurückgegriffen (siehe Bild 8). Innerhalb jeder Box befinden sich die mittleren

50% aller Einzelwerte (Symbol ×) jedes Durchlaufs, d. h. je 12 bis 13 Einzelwerte. Ein Querstrich innerhalb jeder Box kennzeichnet die Lage des Median. Die Mittelwerte sind durch das Symbol ◆ gekennzeichnet und jeweils rechts der Box vermerkt. Für jede Anzahl an Loops liegt der Mittelwert des relativen Fehlers bei 15,6%, was aufgrund der 25-fachen Wiederholung nachvollziehbar ist. Wichtig ist hierbei jedoch anzumerken, dass sehr häufig zur Berechnung der Performanz im KDD Prozess fehlerhafterweise der Versuch mit einem Datensatz nicht so häufig wiederholt, sondern nur einmal durchgeführt wird (linke Box, 1 Loop). Dies entspricht einer einfachen 10-fach Kreuzvalidierung. Das vorliegende Beispiel zeigt, dass der relative Fehler dabei sowohl 14,9% (Minimum, unterer Whisker) als auch 16,5% (Maximum, oberer Whisker) betragen kann. Der Anwender darf also nicht von einer robusten Performanz ausgehen. Diese Streubreite reduziert sich mit steigender Anzahl an durchlaufenden Loops, bis bei einem zehnfachen Loop die Länge der Box der Hälfte der ursprünglichen Box entspricht. Bei 1000 Loops beläuft sich dieser Wert dann nur noch auf einen geringen Bruchteil von $1/22$. Es kann also gezeigt werden, dass der Einfluss der Reihenfolge der Eingangsdatensatzes durch geschickt eingesetzte Zufallszahlen, sowie durch eine mehrfache Wiederholung der Kreuzvalidierung minimiert und die Robustheit eines Performanzkennwertes erhöht werden kann.

5 Diskussion und Ausblick

Es konnte gezeigt werden, dass durch einen richtigen Einsatz von Zufallszahlen in Kombination mit einer statistischen Absicherung, der Einfluss der Reihenfolge der Eingangsdaten auf das Ergebnis eines KDD Prozesses reduziert werden kann. Ein Kompromiss zwischen Rechenzeit und robuster Performanz zeigt sich bei einer 10 bis 20-fachen Kreuzvalidierung und anschließender Mittelwertbildung. Die robuste Performanz zu einem Vorhersagemodell ist vor allem dann notwendig, wenn verschiedene Vorhersagemodelle miteinander verglichen werden. Generell werden verschiedene Vorhersagemodelltypen trainiert (z. B. Regressionsgleichungen, künstlichen neuronalen Netzen, u. s. w.) und das passende Vorhersagemodell Anhand der Performanz ausgewählt. Dies ist notwendig, da nicht jedes Vorhersagemodell zu jedem wirklichen Systemverhalten passt, z. B. ist ggf. für Schallmessungen ein anderer Vorhersagemodelltyp als für Akustiksimulationsergebnisse notwendig. Wesentlich wichtiger ist die robuste Performanz bei Optimierungsprozessen, die zur Reduzierung des Underfittings eingesetzt werden. Treten hierbei in einem Optimierungsschritt zwei Varianten eines Vorhersagemodells mit ihren Performanzen gegeneinander an, so kann eine hohe Varianz einer Performanz dazu führen, dass die eigentlich bessere Vorhersagemodellvariante nicht mehr berücksichtigt wird.

Danksagung

Dieses Forschungs- und Entwicklungsprojekt wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) im Rahmenkonzept „Forschung für die Produktion von morgen" gefördert und vom Projektträger Karlsruhe (PTKA) betreut. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Literatur

[1] Breitsprecher, T. et al.: "Acquisition of heuristic knowledge for the prediction of the frictional behavior of surface structures created by self-excited tool vibrations", Key Engineering Materials, Vol. 504–506, S. 963–968, 2012.

[2] Fayyad, U. et al.: "From Data Mining to Knowledge Discovery in Databases", AI Magazine, Vol. 17, S. 37–54, 1996.

[3] Kellermeyer, M. et al.: "Numerische Untersuchung der Einflüsse von Streuungen auf Versagenskriterien von Composite Strukturen", In: 23. DfX-Symposium 2012, 2012.

[4] Küstner, C. et al.: "Design for noise reduction – The architecture of an engineering assistance system for the development of noise-reduced rotating systems", In: ICED 13 – 19th International Conference on Engineering Design, Seoul, 2013.

[5] Mohr, H.: "Wissensnetze heute", In: Beyrer, K. et al. (Hrsg.): Das Netz. Sinn und Sinnlichkeit vernetzter Systeme, Heidelberg: Braus, S. 125–129, 2002.

[6] Prekopcsák, Z. et al.: "Cross-validation: the illusion of reliable performance estimation", In: RCOMM RapidMiner Community Meeting and Converence, 2010.

[7] Walter, M. et al.l: "Tolerance analysis of mechanism taking into account the interactions between deviations using meta-models", In Proceedings of 9th Norddesign Conference 2012, Aalborg, 2012.

[8] Witten, I. H. et al.: "Data mining: Practical machine learning tools and techniques", 3. Auflage, Burlington: Morgan Kaufmann, 2011.