Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften, Bereich EuI

Semesterarbeit

Data Mining Semesterprojekt Dokumentation

Eingereicht am: 15. Juni 2021

Christoph Werner
geboren am 16. November 1993
Email: christoph.werner@stud.hs-wismar.de

Josef Prothmann
geboren am 16. Dezember 1998
Email: j.prothmann@stud.hs-wismar.de

Dozent: Prof. Dr. rer. nat. Jürgen Cleve

Inhaltsverzeichnis

1 Grundlagen

Das Data Mining ist ein immer weiter polarisierendes Fachgebiet der Informatik, welches zunehmend Anwendung in Problemstellung der Forschung und der Industrie findet. Das klassische Data Mining befasst sich mit der automatischen Verarbeitung von nicht trivialen Daten und beschäftigt sich mit den Themen der Datenvorverarbeitung sowie mit Anwendungsfällen von Analysealgorithmen [**?**]. Im Rahmen des Studienmoduls „Wissenextraktion / Data Mining“ wurde ein Projekt bearbeitet, welches anhand eines gegebenen Datensatz der Data Science Plattform Kaggle.com, die kennengelernten Vorverarbeitungsmechanismen sowie zwei Ausgewählte Analyseverfahren aus dem Modul anwendet.

1.1 Datenvorverarbeitung

Die Datenvorverarbeitung ist eines der ersten Schritte, welche in einem Data Mining Projekt angewendet werden[1]. Das Ziel ist es die Rohdaten so aufzubereiten, dass Analysealgorithmen einfach angewendet werden können und bestmögliche Ergebnisse liefern. Damit die Daten entsprechen aufbereitet werden können, müssen zunächst die Daten verstanden werden (Data Understanding). In der nächsten Sektion wird auf die Rohdaten eingegangen, welche für das Projekt zur Verfügung standen.

[1] Nach der Datenbeschaffung von der Datenquelle

1.1.1 Daten

Der Datensatz besteht aus acht Spalten und 1000 Zeilen. Die Tabelle liefert Testergebnissen in Mathe, Lesen und Schreiben anhand von verschiedenen Ausgangskriterien. Unterteilt wird in Geschlecht, ethnische Herkunft, Bildung der Eltern, welcher Typ Mittagessen eingenommen wurde und ob sich auf den Test vorbereitet wurde.

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
2	female	group B	bachelor's degree	standard	none	72	72	74
3	female	group C	some college	standard	completed	69	90	88
4	female	group B	master's degree	standard	none	90	95	93
5	male	group A	associate's degree	free/reduced	none	47	57	44

Abbildung 1.1: Auszug aus dem Rohdatensatz

1.1.2 Vorverarbeitung

Im nächsten Schritt werden die Daten in eine, für die Algortihmen einfacher zu verarbeitende Form gebracht. Umgestetzt wurde das mit dem folgendem Workflow aus Abbildung 1.2.

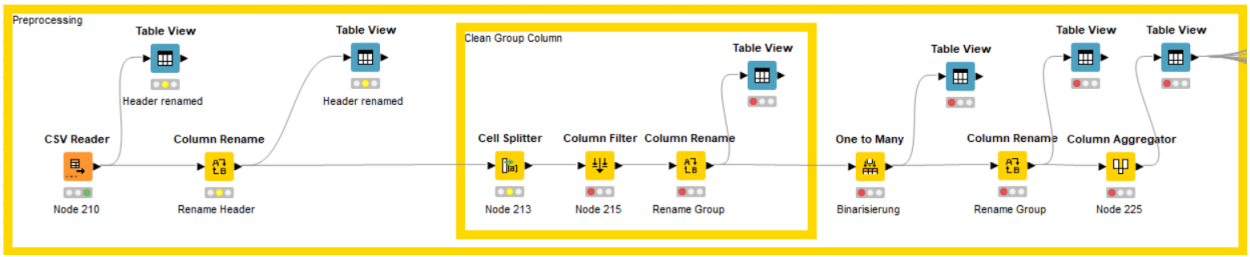


Abbildung 1.2: Datenvorverarbeitung in KNIME

Folgende Verarbeitungsschritte wurden angewendet:

- Umformulieren der Tabellenheader
- Aufteilen der Group-Spalte und ausschneiden des Wortes Group
- Durchschnittsbildung über die drei Testergebnisse
- Binarisieren der Spalten mit nominalen Datentypen

Nach der Vorverarbeitung ist folgende Tabelle aus Abbildung 1.3 entsanden.

gender	parental education	lunch	preparation	math	reading	writing	group	bachelor	college	master	associate	high school	some high school	standard	nonstandard	none	completed	B	C	A	D	E	Mean
female	bachelor's degree	standard	none	72	72	74	B	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	72.66666666666666
female	some college	standard	completed	69	90	88	C	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	82.33333333333333
female	master's degree	standard	none	90	95	93	B	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	92.66666666666666
male	associate's degree	free/reduced	none	47	57	44	A	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	49.33333333333333
male	some college	standard	none	76	78	75	C	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	76.33333333333333
female	associate's degree	standard	none	71	83	78	B	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	77.33333333333333

Abbildung 1.3: Datensatz nach der Vorverarbeitung

Das Ergebnis der Vorverarbeitung ist die Eingangstabelle für die weiteren Analyse Algorithmen. Im Folgendem wird auf zwei Analysekonzepte aus der Vorlesung eingegangen: Das Konzept des Entscheidungsbaum und das der Cluster.

1.2 Knime Knoten für Entscheidungsbäume

Knime selbst bietet einen 'Decision Tree Learner' Knoten und die Extension von Weka biete eine Vielzahl von neuen Knoten für Entscheidungsbäume, welche im Folgenden kurz angesprochen seien.

1.2.1 Decision Tree Learner (Knime)

Laut Beschreibung auf der Homepage von Knime muss das Zielattribut für diesen Knoten nominal sein. Die anderen zur Entscheidungsfindung verwendeten Attribute können entweder nominal oder numerisch sein. Der Algorithmus stellt zwei Qualitätsmaße für die Split-Berechnung zur Verfügung: den Gini-Index und das Gain-Ratio. Weiterhin gibt es eine Post Pruning-Methode, um die Baumgröße zu reduzieren und die Vorhersagegenauigkeit zu erhöhen. Die Pruning-Methode basiert auf dem Prinzip der minimalen Beschreibungslänge. [**?**]

1.2.2 SimpleCart (Weka 3.7)

Ein bedeutendes Merkmal des CART-Algorithmus ist, dass nur Binärbäume erzeugt werden können. CARTs zeichnen sich dadurch aus, dass sie die Daten in Bezug auf die Klassifikation optimal trennen. Dies wird durch einen Attributschwellwert erreicht. Bei den errechneten Entscheidungsbäumen gilt: Je höher der Informationsgehalt eines Attributs in Bezug auf die Zielgröße, desto weiter oben im Baum findet sich dieses Attribut. [**?**]

1.2.3 J48 (Weka 3.7)

Diesem Knoten liegt der C4.5 Algorithmus von J. Ross Quinlan zu Grunde.

Grundlegend verhalent sich dieser Algorithmus und der CART-Algorithmus ähnlich. Der Hauptunterschied besteht darin, dass bei C4.5 keine binäre Aufteilung erfolgen muss, sondern eine beliebige Anzahl Verzweigungen eingebaut werden können. Der Baum wird dadurch deutlich breiter und ist außerdem meist weniger tief als der korrespondierende CART-Baum. Ein weiterer Unterschied zeigt sich beim sogenannten Pruning, beim Stutzen des Baumes. CART erzeugt einige Subtrees und testet diese mit neuen, vorher noch nicht klassifizierten Daten. C4.5 beschneidet den Baum ohne Beachtung der gegebenen Datenbasis. [**?**]

1.2.4 NBTree (Weka 3.7)

Es wurde bereits gezeigt, dass Naive-Bayes-Induktionsalgorithmen bei vielen Klassifizierungsaufgaben erstaunlich genau sind, selbst wenn die bedingte Unabhängigkeitsannahme, auf der sie basieren, verletzt ist. Die meisten Studien wurden jedoch mit kleinen Datenbanken durchgeführt. NBTree einen Hybridalgorithmus aus Entscheidungsbaum-Klassifikatoren und Naive-Bayes-Klassifikatoren. Die Knoten des Entscheidungsbaums enthalten inivariate Splits wie reguläre Entscheidungsbäume, aber die Blätter enthalten Naive-Bayes'sche Klassifikatoren. Der Ansatz behält die Interpretierbarkeit von Naive-Bayes und Entscheidungsbäume, während er zu Klassifikatoren führt, die häufig beide Konstituenten übertreffen, insbesondere in größeren Datenbanken. [**?**]

1.2.5 REPTree (Weka 3.7)

Dieser Knoten stellt einen schnellen Entscheidungsbaum-Lerner. Erstellt wird hierbei eine Entscheidungs-/Regressionsbaum unter Verwendung von Informationsgewinn/Varianz. Dieser wird anschließend unter Verwendung von fehlerreduziertem Pruning (mit Backfitting) bereinigt. Außerdem werden Werte für numerische Attribute nur einmal sortiert. Fehlende Werte werden behandelt, indem die entsprechenden Instanzen in Stücke aufgeteilt werden (d.h. wie beim C4.5 Algorithmus).

1.2.6 LMT (Weka 3.7)

Baumінduktionsverfahren und lineare Modelle sind beliebte Techniken für überwachte Lernaufgaben, sowohl für die Vorhersage von nominalen Klassen als auch von numerischen Werte. Für die Vorhersage numerischer Größen wurde an der Kombination dieser beiden beiden Verfahren zu „Modellbäumen“ gearbeitet, d.h. zu Bäumen, die lineare Regressionsfunktionen an den Blättern enthalten. LMT ist ein Algorithmus, der diese Idee für Klassifikationsprobleme adaptiert, wobei die logistische Regression anstelle der linearen Regression verwendet wird. Es wird einen stufenweisen Anpassungsprozess verwendet, um die logistischen Regressionsmodelle zu konstruieren, welche die relevante Attribute in den Daten auf natürliche Weise auswählen kann. [**?**]

1.2.7 DecisionStump (Weka 3.7)

Ein Entscheidungsstumpf ist ein maschinelles Lernmodell, das aus einem einstufigen Entscheidungsbaum besteht. Das heißt, es ist ein Entscheidungsbaum mit einem internen Knoten (der Wurzel), der direkt mit den Endknoten (seinen Blättern) verbunden ist. Ein Entscheidungsstumpf macht eine Vorhersage, die auf dem Wert nur eines einzigen Eingabe-Features basiert. Manchmal werden sie auch als „1-rules“ bezeichnet. Bei kontinuierlichen Merkmalen wird in der Regel ein Schwellenwert gewählt, und der Stumpf enthält zwei Blätter - für Werte unterhalb und oberhalb des Schwellenwerts. In seltenen Fällen können jedoch auch mehrere Schwellenwerte gewählt werden, und der Stumpf enthält dann drei oder mehr Blätter. [**?**]

Entscheidungsstümpfe werden oft als Komponenten (als „schwache Lerner“ oder „Basislerner“ bezeichnet) in Ensemble-Techniken des maschinellen Lernens wie Bagging und Boosting verwendet. Ein Algorithmus zur Gesichtserkennung nach Viola-Jones verwendet z.B. AdaBoost mit Entscheidungsstümpfen als schwache Lerner. [**?**]

1.2.8 J48Graft (Weka 3.7)

J48graft basiert auf J48, dem der C4.5 Algortihmus zugrunde liegt. J48graft nutzt den C4.5++ Algortihmus, welcher sich den Ansatz ‘all-tests-but-one-partition’ (ATBOP) zu Nutzen macht um den C4.5 Algortihmus zu verbessern.

Beim Entscheidungsbaum-Grafting werden Knoten zu einem vorhandenen Entscheidungsbaum hinzugefügt, um den Vorhersagefehler zu reduzieren. Der C4.5++ Algortihmus behält die Fehlerreduktion des ursprünglichen Grafting-Algorithmus bei und reduziert gleichzeitig die Rechenzeit und die Komplexität des abgeleiteten Baums drastisch. Bias/Varianz-Analysen zeigen, dass das ursprüngliche Transplantationsverfahren in erster Linie durch Varianzreduktion funktionierte, während das neue Technik sowohl die Verzerrung als auch die Varianz reduziert. [**?**]

1.2.9 BFTree (Weka 3.7)

Entscheidungsbäume sind potenziell leistungsfähige Prädiktoren und stellen die Struktur eines Datensatzes dar. Standard-Entscheidungsbaum-Lerner, wie z.B. C4.5, expandieren Knoten, während bei Best-First-Entscheidungsbaumlernern der „beste“ Knoten zuerst expandiert wird. Der „beste“ Knoten ist der Knoten, dessen Aufteilung zu einer maximalen Reduktion der Unreinheit (z.B. Gini-Index) unter allen für die Aufteilung verfügbaren Knoten führt. Der resultierende Baum ist derselbe, wenn er vollständig gewachsen ist, nur die Reihenfolge, in der er aufgebaut wird, ist unterschiedlich. In der Praxis werden einige Zweige eines vollständig expandierten Baums die zugrundeliegende Information in der Domäne nicht widerspiegeln. Dieses Problem wird als Overfitting bezeichnet und wird hauptsächlich durch verrauschte Daten verursacht. Pruning ist notwendig, um eine Überanpassung der Trainingsdaten zu vermeiden, und verwirft die Teile, die nicht vorhersagend für zukünftige Daten sind. Die Best-First-Knotenerweiterung ermöglicht die Untersuchung neuer Pruning-Techniken, indem die Anzahl der durchgeführten Expansionen auf Basis von Kreuzvalidierung verglichen werden. [**?**]

1.2.10 RandomTree (Weka 3.7)

Dieser Knime-Knoten ist zum Konstruieren eines Baums geeignet, der an jedem seiner Knoten K zufällig ausgewählte Attribute berücksichtigt. Der Algorithmus führt kein Pruning durch. Verfügt jedoch über eine Option, die eine Schätzung der Klassenwahrscheinlichkeiten auf der Basis eines Hold-out-Sets (Backfitting) ermöglicht.

1.2.11 RandomForest (Weka 3.7)

Random Forests sind eine Kombination von Baumprädiktoren, wobei jeder Baum von den Werten eines Zufallsvektors abhängt, der unabhängig und mit der gleichen Verteilung für alle Bäume im 'Wald' genutzt wird. Der Generalisierungsfehler für Wälder konvergiert gegen einen Grenzwert, wenn die Anzahl der Bäume im Wald groß wird. Der Generalisierungsfehler eines Waldes von Baumklassifikatoren hängt von der Stärke der einzelnen Bäume im Wald und der Korrelation zwischen ihnen ab. Die Verwendung einer zufälligen Auswahl von Merkmalen zur Aufteilung jedes Knotens führt zu Fehlerraten, die mit denen von Adaboost (Freund und Schapire) vergleichbar, aber robuster gegenüber Rauschen sind. Interne Schätzungen überwachen Fehler, Stärke und Korrelation, und diese werden verwendet, um die Reaktion auf die Erhöhung der Anzahl der bei der Aufteilung verwendeten Features zu zeigen. Interne Schätzungen werden auch verwendet, um die Wichtigkeit von Variablen zu messen. Diese Ideen sind auch auf Regression anwendbar. [**?**]

1.2.12 FT (Weka 3.7)

Funktionale Bäume erlauben eine enge Integration verschiedener Repräsentationssprachen für verallgemeinernde Beispiele. Es ist hervorzuheben, dass es Standardalgorithmen für topologische Transformationen auf Entscheidungsbäumen gibt, die hier nicht weiter besprochen, jedoch von Gama behandelt werden. [**?**]

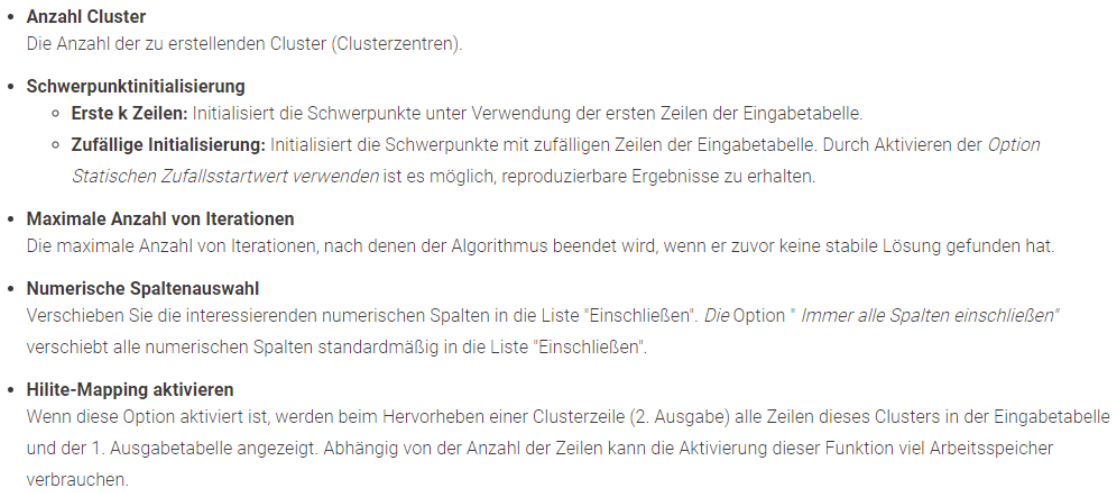
1.3 Clusteranalysen

In KNIME wird von der Auslieferungsversion diverse Knoten zur Clusteranalyse angeboten. Im Folgenden werden drei Clusterkonzepte vorgestellt und im nächsten Kapitel die Implementierung dargestellt.

1.3.1 K Means - Algorithmus

Der K Means Knoten in KNIME berechnet für eine angegebene Anzahl an Cluster (keine dynamische Clusteranzahl) die Clusterzentren. Die Cluster werden iterativ berechnet und der Algorithmus stoppt, wenn entweder ein Limit an maximalen Iterationen erreicht wird oder die Cluster sich nicht mehr verändern. In dem gegebenen Datensatz wird der Algorithmus durch sich selbst beendet und nicht durch

eine maximale Anzahl an Iterationen. Der Output des Knoten ist ist die Tabelle mit einer angehängten Spalte, wo die Tabellenreihe einem Cluster zugeordnet wurde. Der Clusteralgorithmus verwendet zum Berechnen der Clusterzugehörigkeit die euklidische Distanz.



- **Anzahl Cluster**
  Die Anzahl der zu erstellenden Cluster (Clusterzentren).
- **Schwerpunktinitialisierung**
  - **Erste k Zeilen:** Initialisiert die Schwerpunkte unter Verwendung der ersten Zeilen der Eingabetabelle.
  - **Zufällige Initialisierung:** Initialisiert die Schwerpunkte mit zufälligen Zeilen der Eingabetabelle. Durch Aktivieren der *Option Statischen Zufallsstartwert verwenden* ist es möglich, reproduzierbare Ergebnisse zu erhalten.
- **Maximale Anzahl von Iterationen**
  Die maximale Anzahl von Iterationen, nach denen der Algorithmus beendet wird, wenn er zuvor keine stabile Lösung gefunden hat.
- **Numerische Spaltenauswahl**
  Verschieben Sie die interessierenden numerischen Spalten in die Liste "Einschließen". *Die* Option " *Immer alle Spalten einschließen*" verschiebt alle numerischen Spalten standardmäßig in die Liste "Einschließen".
- **Hilite-Mapping aktivieren**
  Wenn diese Option aktiviert ist, werden beim Hervorheben einer Clusterzeile (2. Ausgabe) alle Zeilen dieses Clusters in der Eingabetabelle und der 1. Ausgabetabelle angezeigt. Abhängig von der Anzahl der Zeilen kann die Aktivierung dieser Funktion viel Arbeitsspeicher verbrauchen.

Abbildung 1.4: Knotenoptionen K-Means -Quelle: knime.com

1.3.2 Dichtebasiertes Clustern

Für das Dichtebasierte Clustern in KNIME kann auf den Standard-Knoten **DBSCAN** zurückgegriffen werden. DBSCAN definiert drei Arten von Punkten

- Core Punkte (haben eine minimale Anzahl an Nachbarn in einer bestimmten Entfernung)
- Border Punkte (sind innerhalb der Distanz der Core Points besitzen aber weniger Nachbarn)
- Rausch Punkte (weder Core-, noch Border Punkte)

Cluster werden gebildet, indem Core Punkte miteinander verbunden werden. Alle Punkte die innerhalb der Kernpunkte-Distanz liegen werden als ein Cluster gesehen. Alle Punkte ausserhalb werden als Rausch-Punkte gesehen.

- **Epsilon**
  Die Entfernung, innerhalb derer die Anzahl der Punkte gezählt werden soll, um zu bestimmen, welche Punkte *Kernpunkte sind* .
- **Mindestpunktzahl**
  Die minimale Anzahl von Punkten innerhalb von **eps** , um zu bestimmen, welche Punkte Kernpunkte innerhalb eines Clusters sind.
- **Daten in den Speicher laden**
  Da DBScan über eine quadratische Laufzeit verfügt, kann die Leistung durch Laden des gesamten Datasets in den Arbeitsspeicher verbessert werden.

Abbildung 1.5: Knotenoptionen DBSCAN -Quelle: knime.com

1.3.3 Hierarchisches Clustern

Das Hierarchische Clustern ist ein agglomerativer (bottom-up) Algorithmus-Knoten von KNIME. Dieser Knoten stellt verschiedene Distanzmaße zur Verfügung um die Cluster zuberechnen. Folgende beispiel Distanzmaße werden bereitgestellt.

- Euklidische Distanz
- Manhatten Distanz
- Mirkowski Distanz
- ... und weitere

Folgende Konfigurationen sind in diesem Knoten auszuwählen.

- **Nummernausgabecluster**
  Welche Hierarchieebene für die Ausgabespalte verwendet werden soll.
- **Distanzfunktion**
  Welches Entfernungsmaß für die Entfernung zwischen Punkten verwendet werden soll.
- **Verknüpfungstyp**
  Welche Methode zum Messen des Abstands zwischen Punkten verwendet werden soll (wie oben beschrieben)
- **Distanz-Cache**
  Das Caching der Abstände zwischen den Datenpunkten verbessert die Leistung insbesondere bei hochdimensionalen Datensätzen drastisch. Es benötigt jedoch viel Speicher, sodass Sie es für große Datensätze ausschalten können.

Abbildung 1.6: Knotenoptionen Hierachical Clusterer -Quelle: knime.com

1.3.4 Silhouette Coefficient

Der Silhouette Coefficient wird zu Fehleranalyse in den Cluster-Workflow eingebaut. Er stellt eine nützliche Metrik zur Berechnung der Cluster-Leistung dar. Standardmäßig verwendet der Koeffizient die euklidische Distanz zum Berechnnen der Zeilenabständen, ist aber um andere Abstandsmaße erweiterbar.

Nachdem alle Knoten, welche in diesem Projekt verwendet werden beschrieben worden sind, wird im Folgendem auf die Implementierung dieser Knoten in KNIME eingegangen.

2 KNIME Implementierung

2.1 Entscheidungsbäume

In Abbildung 2.1 ist zunächst der gesamte Workflow mit Metaknoten dargestellt. Unten befindet sich die Datenvorverarbeitung, oben die in Abbildung 2.2 dargestellte Eingabe der Variablen für Aufteilung der Trainingsmenge und die vorherzusagende Spalte.

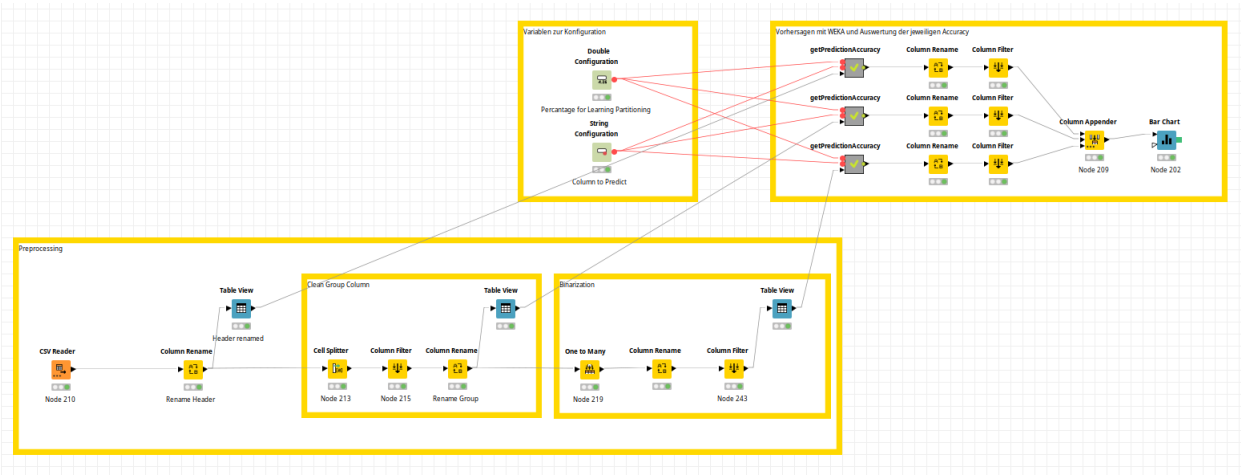


Abbildung 2.1: Aufbau des Workflows

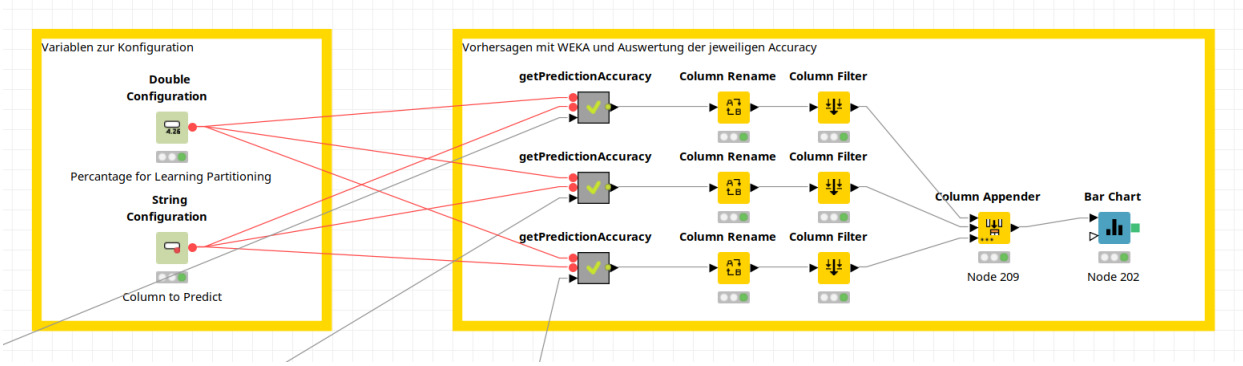


Abbildung 2.2: Ausschnitt des Workflows (Variableneingabe und Vorhersage)

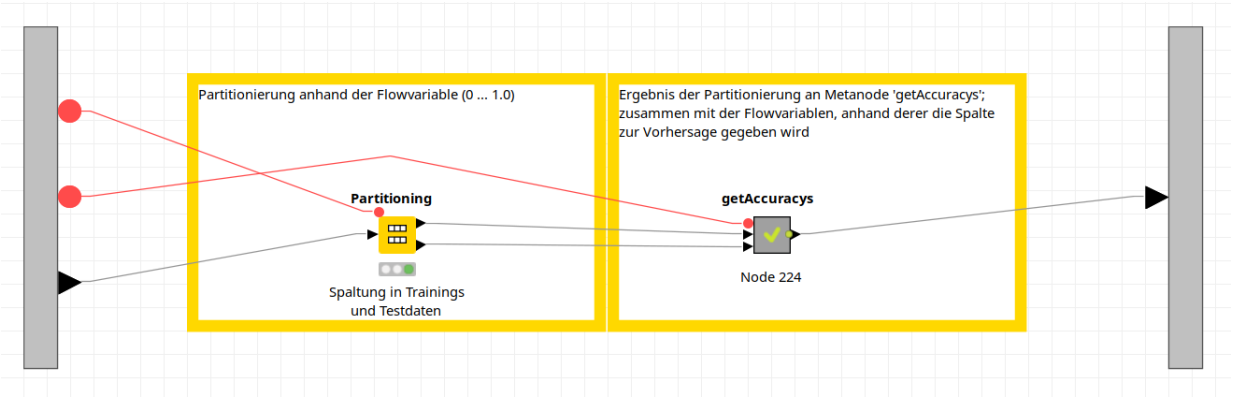


Abbildung 2.3: Inhalt des Metanodes ‘getPredictionAccuracy’

Hierbei wird dem Metaknoten ‘getPredictionAccuracy’ dreimal der verschieden aufbereitete Datensatz übergeben. Von ihm ausgegeben werden, die Werte des ‘Accuracy’ Feldes für jeden in Kapitel 1.2 genannten Entscheidungsbaum-Knoten. Die Trainingsmengen sind hierbei immer gleich (vgl. Abblidung 2.3). Zusammengefügt und visualisiert werden die Ergebnisse in einem Barchart.

Der in Abbildung 2.3 gezeigte Metaknoten ‘getAccuracys’ ist in Abbildung 2.5 genauer dargestellt. Er beinhaltet für jeden in Kapitel 1.2 genannten Entscheidungsbaum einen eigenen Metaknoten dessen Inhalt analog zu dem in Abbildung 2.4 ist. Übergeben werden hierbei die Trainings und die Testmenge, sowie der Name der vorherzusagende Spalte. Der Knoten ‘Scorer’ erzeugt unteranderem den zu vergleichenden Wert ‘Accuracy’, welcher durch einen Spalten- und Reihenfilter extrahiert und umbenannt wird.

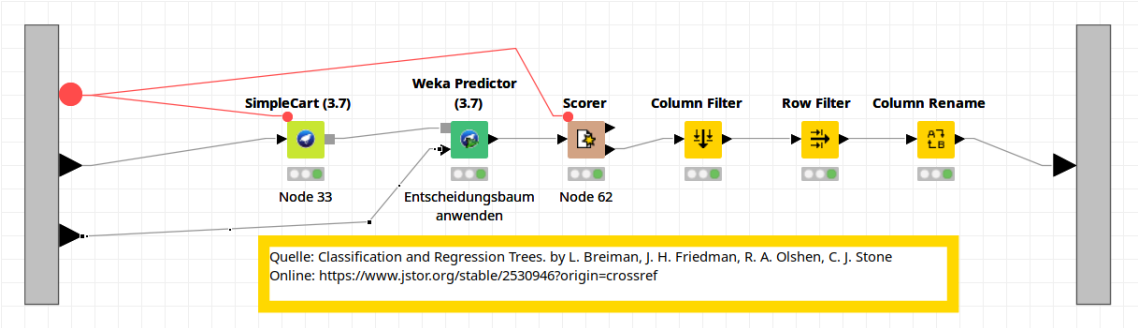


Abbildung 2.4: Inhalt des Metanodes ‘getAccuracyWekaSimpleCart’

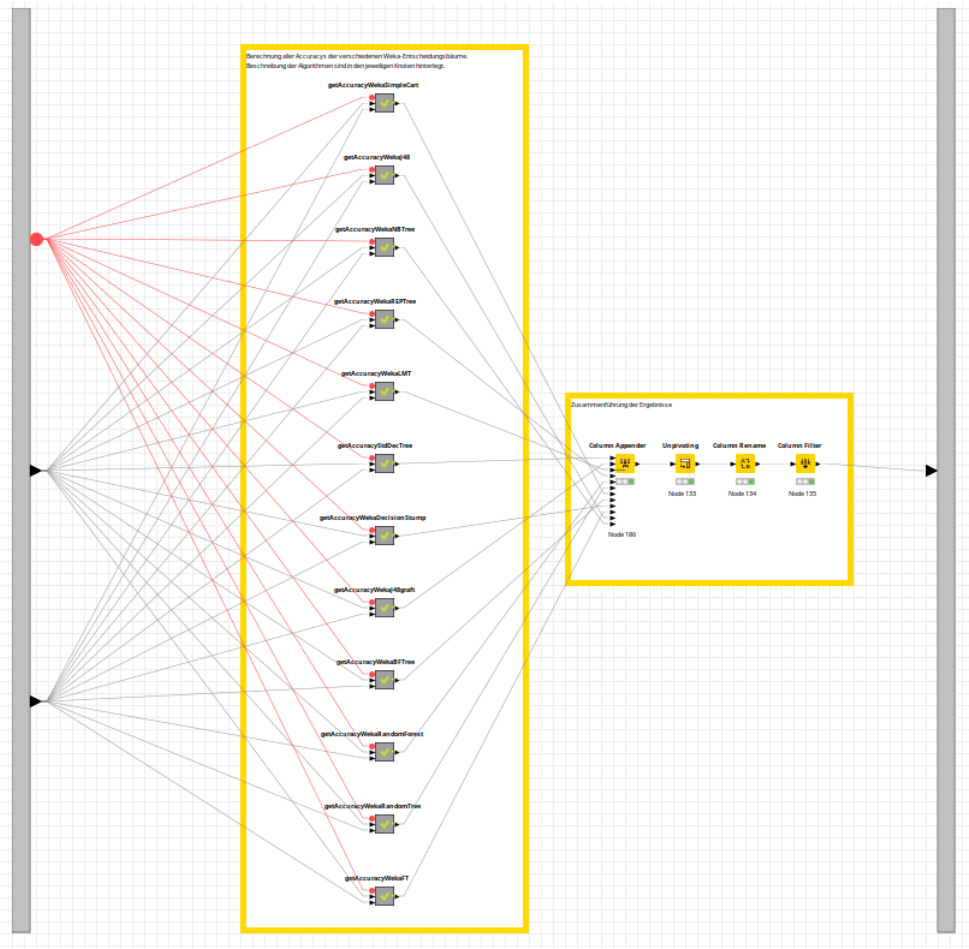


Abbildung 2.5: Inhalt des Metanodes 'getAccuracys'

2.2 Clusteranalyse

2.2.1 kMeans Cluster Workflow

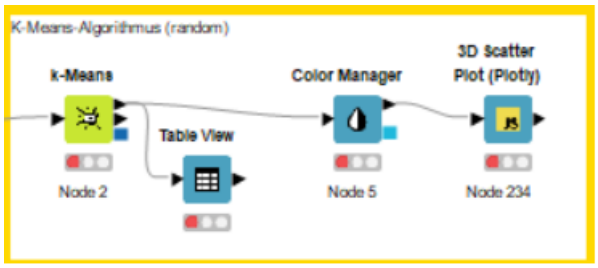


Abbildung 2.6: K Means Algorithmus Implementierung in KNIME

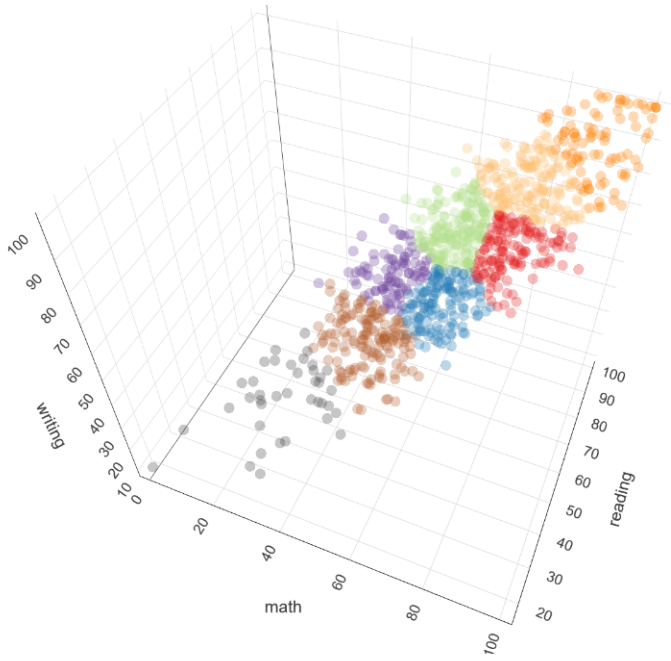


Abbildung 2.7: K Means Algorithmus Implementierung in KNIME

2.2.2 Dichtebasierters Cluster Workflow

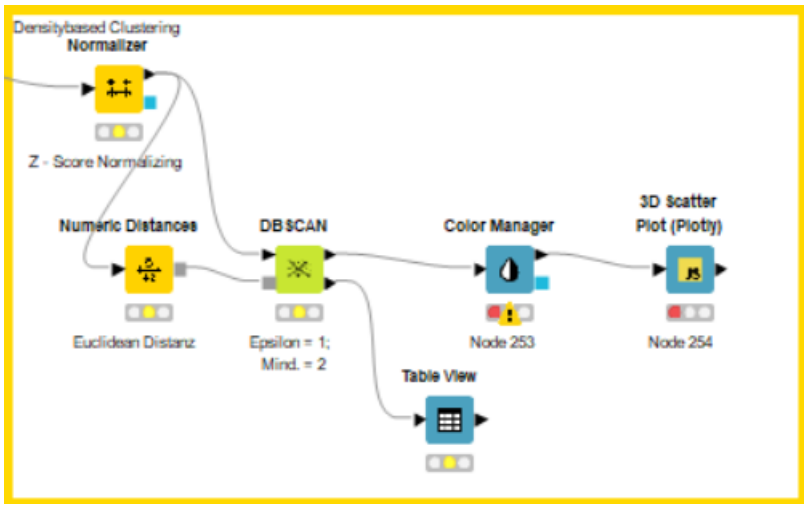


Abbildung 2.8: K Means Algorithmus Implementierung in KNIME

2.2.3 Hierarchisches Cluster Workflow

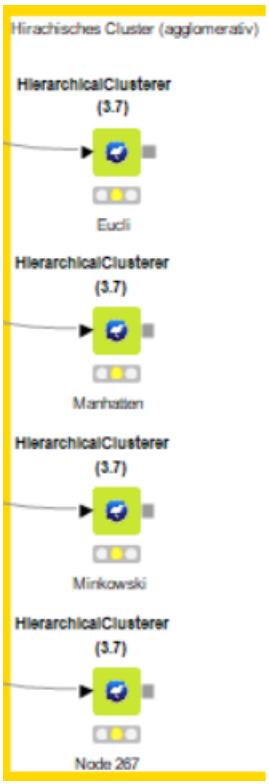


Abbildung 2.9: K Means Algorithmus Implementierung in KNIME

Abbildungsverzeichnis