Ausarbeitung des Studienprojektes

Aufbau und Implementierung eines Data-Mining-Analyseprozesses mit KNIME Analytics Platform

Business Intelligence 2

Sommersemester 2019

Prof. Dr. Hubert Kempter

Autor: Marco Hetzel

Matr. 10056756

hetzel.marco@fh-swf.de

Fachhochschule Südwestfalen

07.07.2019

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabenstellung

Dieses Studienprojekt hat zur Aufgabe sich mit der Materie des Data Minings und der gesamten Datenverarbeitungskette mit Hilfe einer fiktive Problemsituation auseinanderzusetzen. Dabei sollen nicht nur die Anbindung der Daten, deren Aufbereitung, Bereinigung sowie Filterung Anwendung finden, sondern auch explizit ein für ETL-Prozesse geeignetes Programm zur Lösung der Aufgabe genutzt und kennengelernt werden. Anhand den eingesetzten Data Mining Techniken und Methoden soll dann eine unternehmerische Interpretation der Analyseergebnisse stattfinden.

Zunächst wird bei den theoretischen Grundlagen der Datenanalyseprozess näher erläutert. Dabei wird vor allem auf den KDD-Prozess eingegangen, der bereits 1996 in einem Paper von FAYYAD et al. veröffentlicht wurde. Seit dieser Veröffentlichung hat sich technisch enorm viel getan, so dass wir dieses Studienprojekt mithilfe der Open Source Software „KNIME Analytics Platform[1]“ mit wesentlich weniger Aufwand umsetzen können. Auch auf Data Mining explizit wird später eingegangen.

Die Umsetzung des Studienprojektes wird in Kapitel 3 näher erläutert. Dabei wird zum Einen auf den iterativ-inkrementiellen Vorgang bei der Erkenntnisgewinnung mit Knime zum Anderen auf die durch die Aufgabenstellung vorgegebene Phasen-Orientierte Modellierung eingegangen. Die Phasen unterstützen dabei bei der Grundstrukturierung des Modell-Aufbaus. Hier werden die gefundenen Problemstellen und wie sie mit Knime gelöst wurden, näher beleuchtet. Dieses Kapitel soll somit zeigen, wie die Lösung der Aufgabe umgesetzt wurde.

Zum Abschluss wird beleuchtet „*was*“ als Lösung durch das Data Mining herausgefunden wurde. Die als Grundlage der Analyse zur Verfügung gestellten Daten der fiktiven Firm AWS (Adventure Works Cycles) bestehen zum einen aus Vergangenheitsdaten , die auch einen Kauf entsprechend belegen und zum anderen aus einer Liste mit Personendaten, die es zu klassifizieren gilt (im Weiteren „Zieldaten“ genannt).

[1] KNIME Analytics Platform - www.knime.com

Die Modelle generieren am Ende eine Mailing-Liste mit einer „Wahrscheinlichkeit der Richtigkeit der Vorhersage“[2].

1.2 Zielsetzung

Ziel ist es ein Modell aufzubauen, welches dazu dient, mögliche Neukunden aus den Zieldaten zu identifizieren. Als Ergebnis soll das erstellte Modell am Ende eine Liste aller potentiellen Kunden mit Kontaktinformationen ausgeben, welche für Marketingzwecke weiter genutzt werden können.

[2] Aufgabenbeschreibung Phase VI

Kapitel 2

Theoretische Grundlagen

2.1 Ablauf einer Datenanalyse mittels KDD

„*Knowledge Discovery in Databases is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*“ [FPSS96]

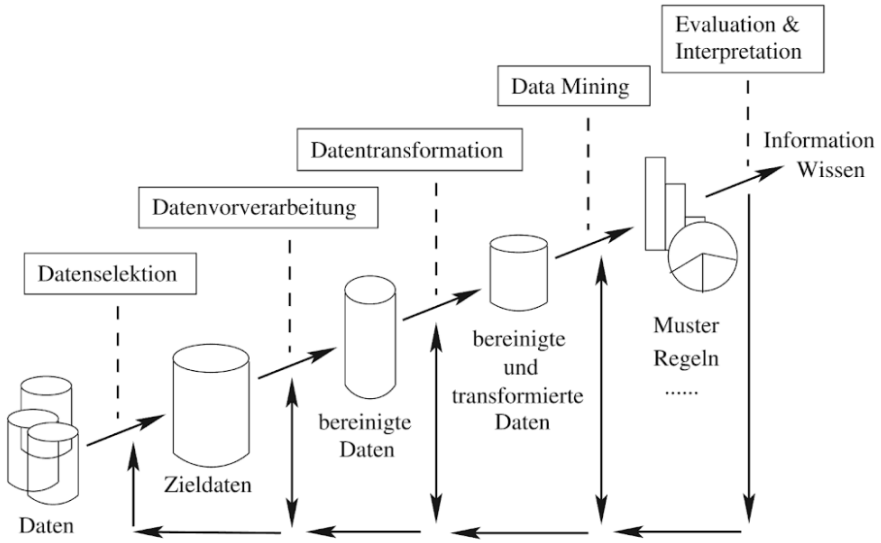


Abbildung 2.1: Ablauf KDD-Prozess nach FAYYAD et al. 1996

Das obige Zitat bringt in einem Satz die Bedeutung des Prozesses der Daten-Analyse in Form von KDD wieder. Abbildung 2.2 stellt diesen 1996 vorgestellten Prozess in grafischer Form dar. Die Datenselektion ist grundsätzlich für die Anpassung und Anbindung der Daten verantwortlich. Hier werden Datenquellen jeglicher Art und Form an das zur Analyse genutzte System angebunden. In dieser Studienarbeit ist die Datenquelle z.B. eine Excel-Datei. Sind die Daten erstmal geladen, so beginnt man im ersten Schritt mit der Vorverarbeitung in der Form, dass die Daten bereinigt und fehlende Daten entsprechend behandelt werden. In der Datentransformation geht es darum Merkmalsausprägungen in ein dem Modell zweckdienliches Format zu überführen. So kann es z.B. sein, dass für den Naive Bayes metrische Werte durch Intervalle in Nominale Werte überführt werden müssen.

Das eigentliche Data Mining findet im Anschluss statt. Hier werden bestimmte Algorithmen angewandt (Naive Bayes ist ein erwähntes Beispiel), die dazu dienen, Muster zu erkennen und anhand dieser ein (abstraktes) Modell von den Daten aufzubauen, das später zur Vorhersage genutzt werden kann. Im letzten Schritt der Evaluation&Interpretation ist die Prüfung des Modells mittels zurückgehaltenen Testdaten, sowie der Interpretation der Ergebnisse im Vordergrund.

Wichtig ist allerdings auch, dass von jedem Prozess-Schritt beliebige Schritte zurückgegangen werden kann. Wenn z.B. bei einer Datenanalyse im Schritt der Transformation auffällig wird, dass eine Information fehlt, die allerdings in einer Datenquelle aufrufbar wäre, so kann wieder zur Datenselektion zurückgesprungen werden.

2.2 Data Mining

In der Industrie Data Mining mehr als nur ein Prozess des „Schürfens“ in Daten. Das „Cross Industry Standard Process for Data Mining“-Modell, kurz CRISP-DM-Modell, bewährt sich in der Industrie zudem durch seine weitere Sichtweise. Hier besteht der Prozess zu der reinen Datenanbindung zuerst aus dem Verständnis über die Business-Domäne und damit verbunden dem Verständnis der Daten.

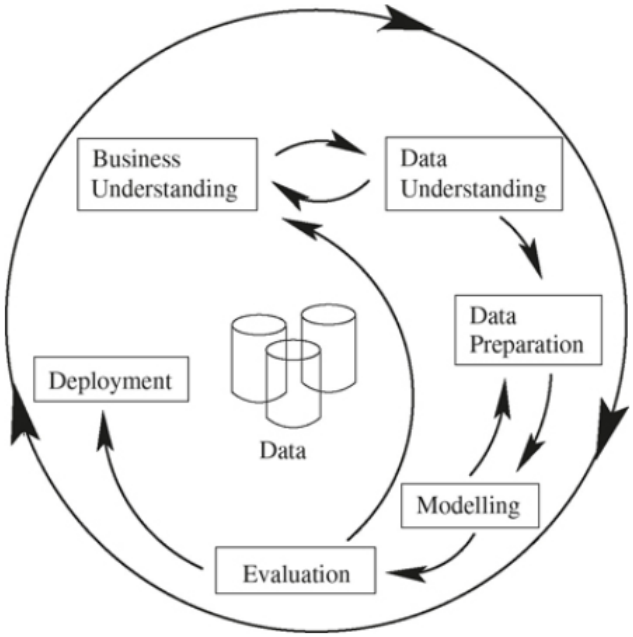


Abbildung 2.2: Ablauf CRISP-DM

CRISP-DM wurde von mehreren Unternehmen gemeinsam entwickelt und geht von

6 Etappen aus. In der Ersten Etappe ist das Verständnis der Aufgabe im Vordergrund. Hier werden Kriterien, die für den Erfolg der Aufgabe stehen, sowie Bedingungen ausgearbeitet, unter denen die Aufgabe ablaufen kann/darf/soll. Sind diese Rahmenbedingungen ausgearbeitet, beschäftigt man sich bei der zweiten Etappe mit den Daten. Bei diesem Projekt z.B. wurde nach mehreren Schritten immer wieder eine Analyse der Daten durchgeführt, wobei im Rahmen der Datenvorverarbeitung die meisten Analysen stattgefunden haben. Bei der zweiten Etappe geht es allerdings auch um eine Prüfung der Qualität und Aussagefähigkeit der Daten. Die dritte Etappe der Datenvorbereitung vereint den zweiten und dritten Schritt des KDD-Prozesses. Genauso ist die vierte und fünfte Etappe der Modellbildung (Data Mining) und der Evaluation mit den KDD-Prozessschritten des Data Minings und der Evaluation&Interpretation vergleichbar. In der sechsten Etappe geht beim CRISP-DM wieder weiter. Hier ist der Einsatz im Unternehmen und die damit Verbundenen Konsequenzen im Mittelpunkt. Es geht hierbei um Change Management im Unternehmen und dem Anwenden des neuen Wissens, das durch Erkenntnisse aus dem jeweils Projektabhängig durchgeführten Data Minings entstanden ist. (Zusammenfassung S.6ff aus [CL16])

2.3 Analyse-Tool KNIME

Die erste Version von KNIME - dem Konstanz Information Miner - wurde 2006 veröffentlicht. Diese Software ist unter Java entwickelt und läuft damit auf jedem Betriebssystem auf dem Java lauffähig ist.

Knime ist in Form der Analytics Platform kostenlos verfügbar. Durch die Drag&Drop-Funktionsweise ist es leicht zu erlernen und bringt schon nach der Installation einen sehr großen „Baustein“-Vorrat mit. Diese Bausteine werden in Knime (sowie in dieser Ausarbeitung) als „Nodes“ bezeichnet. Diese Nodes bilden jeweils eine modular implementierte Lösung für eine Aufgabe ab. So hat z.B. das „Excel Reader (XLS)“-Node die Aufgabe, ein Arbeitsblatt einer Exceldatei in ein Knime lesbares Format zu einzulesen. Bei diesem Node kann dann per Rechtsklick die Konfiguration entsprechend angepasst werden. Sehr hilfreich kann die integrierte Hilfe in Form des „Node Description“-Windows sein. Hier werden viele Informationen über das ausgewählte Node aufgeführt, sowie Informationen welche Eingangs- bzw. Ausgangsdatenpunkte erwartet bzw. gefordert werden. Der Excel-Node hat z.B. nur einen Ausgangsdatenpunkt, der wiederum mit anderen Nodes (auch parallel) verbunden werden kann.

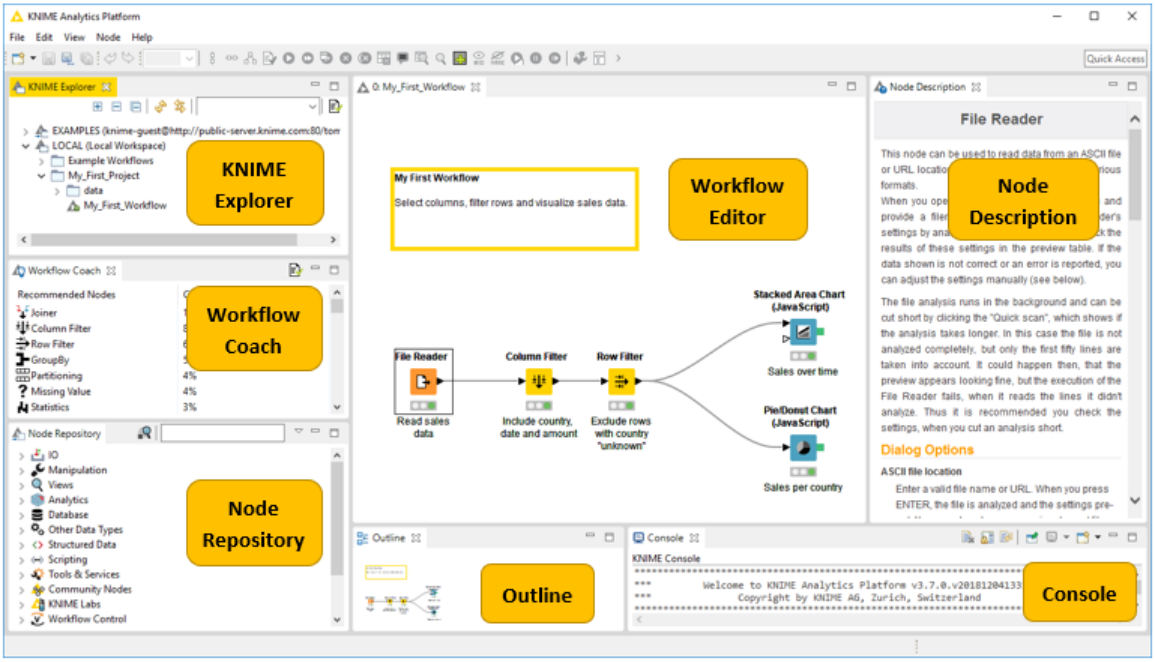


Abbildung 2.3: Knime Analytics Platform Programmaufbau

Knime ist ein sehr übersichtlich gehaltenes Programm wie in Abbildung 2.3 ersichtlich ist. Besonders erwähnenswert ist: Software-Entwickler, die bereits mit Eclipse arbeiten durften, finden sich hier auch schnell „Zuhause“, da Knime auf Eclipse aufbaut.

Da Knime komplett Open Source ist, kann ein Entwickler sich den Quellcode von Knime frei laden und so an Knime eigene Ergänzungen umsetzen. Auch ohne sich mit Knime-Source direkt auseinander setzen zu müssen, ist es möglich eigene Nodes zu entwickeln, die auf die eigenen noch nicht gedeckten Bedürfnisse eingehen.

Knime ist durch seine stetig wachsende Community und zeigt, wie die Open Source-Erfolgsgeschichte vermittelt[1], weltweit bereits in über 50 Ländern in den unterschiedlichsten Branchen aber auch in Regierungen präsenz.

[1]Siehe knime.com/knime-open-source-story

Kapitel 3

Umsetzung in KNIME® und Diskussion der Ergebnisse

3.1 Wahl der iterativ-inkrementellen Umsetzung

3.1.1 Beschreibung der Umsetzungsweise

Als Prozess zur Umsetzung ist die iterativ-inkrementelle Form gewählt worden, da diese Möglichkeiten der Verbesserung in den Vordergrund stellt. Dabei ist es wichtig, dass zunächst dafür gesorgt wird relativ schnell ein kleines „Inkrement“ zu erstellen, welches grob den ersten Mehrwert bringen kann. Im agilen Projektmanagement spricht man hier auch vom „MVP (minimum viable product), dem minimal lebensfähigen Produkt“. Im Weiteren werden dann iterativ weitere Modelle aufgebaut, in denen das durch die vergangenen Iterationen entstandene Wissen eingebaut werden. Wege, die als ziel-fördernd oder korrekt bewertet wurden, werden weiter gegangen und aus Wegen, die als nicht ziel-fördernd bewertet wurden, wird das dadurch entstandene Wissen zur Verbesserung genutzt.

Um das Projekt nun in Knime umsetzen zu können, bedeutete diese Herangehensweise zunächst Wissen anzueignen, welches notwendig ist um mit Knime überhaupt arbeiten zu können. Dieses Inkrement forderte kaum Planung, da in den Projekt-Ressourcen, von Knime.com und von etlichen weiterführenden Online-Quellen bereits reichlich Material auffindbar war. Dennoch gab es damit die Herausforderung, die Quellen klein zu halten um nicht zu viel Zeit in das Kennenlernen des Tools zu setzen. Somit wurde die Wissensaneignung auf die kommenden Iterationen ausgebreitet und es wurde zunächst mit dem kleinsten Inkrement in der ersten Iteration - im Folgenden weiter als „Version“ bezeichnet - gestartet: Das erste lauffähige Modell aufzubauen, das lediglich Daten laden und die Merkmale in ihre Datentypen umwandeln kann.

3.1.2 Beschreibung der Versionen

Eine Version entspricht hier einem abgeschlossenen bzw. für sich abgespeicherten Knime-Projekt (hier auch „Knime-Modell“ genannt). Da nach jeder Version entweder neue Knime-Nodes oder neues Wissen angeeignet wurde, wird hier neben dem Ziel hinter der Version auch die einzelnen Knime-Nodes, die neu eingesetzt wurden genannt. Da die Beschreibung der Knime-Nodes die Dimensionen dieser Ausarbeitung sprengen würde, wird hiermit der versierte Leser angehalten, die genannten Knime-Nodes über die Anwendung Knime selbst[1], eine online-Ressource wie z.B. KNIME Hub[2] oder NodePit.com[3] zu suchen und an dortiger Stelle weiterführend zu betrachten.

Zwischen den Versionen 1 und 5 wurden das Laden der Daten, die erste Datentypisierung der einzelnen geforderten Merkmale bis hin zu den gezielten drei Modellen, dem Decision Tree, dem Clustering mittels k-Means und dem Naive Bayes angewandt. In Version 2 war bereits das Ziel gesetzt den Decision Tree über die einzelnen Merkmale und deren aktuellen Typisierung abbilden zu lassen. Diskrete Merkmale haben hier jede Merkmalsausprägung für sich genommen einen Ast im Entscheidungsbaum abbilden lassen, während das Alter als kontinuierlichen Datentyp per binär-Entscheidung (z.B. entweder < 42 oder $>= 42$) aufgebaut wurde. In Version 4 wurde der Naive Bayes, wie auch der Decision Tree bereits versucht über die Zieldaten, die es zu klassifizieren gilt, laufen zu lassen, wobei das Hindernis erkannt wurde, dass dort nicht alle Merkmale zur Verfügung standen. Zum einen waren nicht alle Merkmale und manche Merkmale nicht in der geforderten Qualität enthalten. So waren die geforderten Merkmale „GeographyKey“, „CommuteDistance“ und „Region“ überhaupt nicht und das Merkmal Alter lediglich in Form des Geburtsdatums zur eigenen Neuberechnung in den Daten gegeben. Außerdem unterschieden sich die Spalten EnglishEducation und EnglishOccupation in der Bezeichnung und die Merkmalsausprägungen des Merkmals Education waren bereits nach 10 Zeichen abgeschnitten und daher nicht mehr mit den Modell-Daten vergleichbar. Um später die Modelle gesichert zu haben, wurde mit Hilfe

[1] Die Anwendung Knime bietet im Fenster „Node Description“ direkt bei Auswahl eines Nodes gute erste Hilfe an.

[2] hub.knime.com - Offizielle Platform von Knime zum Finden von Lösungsmöglichkeiten und Hilfereferenzen. Sehr gut auch der Verweis bei jedem Node zur jeweiligen e-Learning-Einheit in der der Node im Video vorgestellt wird.

[3] NodePit.com - Die KNIME User+Developer Group Dresden gibt mit dieser Website die Möglichkeit nach Nodes, die auch nicht zum Standardrepertoire von KNIME gehören, zu suchen und damit auch noch weiter führende Ideen zu entdecken.

des „Model Writer“-Nodes jedes der drei Modelle in eine eigene Datei gespeichert. Nun, da in Version 4 durch die Vielzahl an Nodes doch ein Übersichtsverlust drohte, wurde in Version 5 damit dagegen gesteuert, dass durch den Einsatz der Funktion „Encapsulate into Wrapped Metanode“ die jeweilig zusammengehörigen Nodes als „Gruppierung“ zusammengefasst wurden:

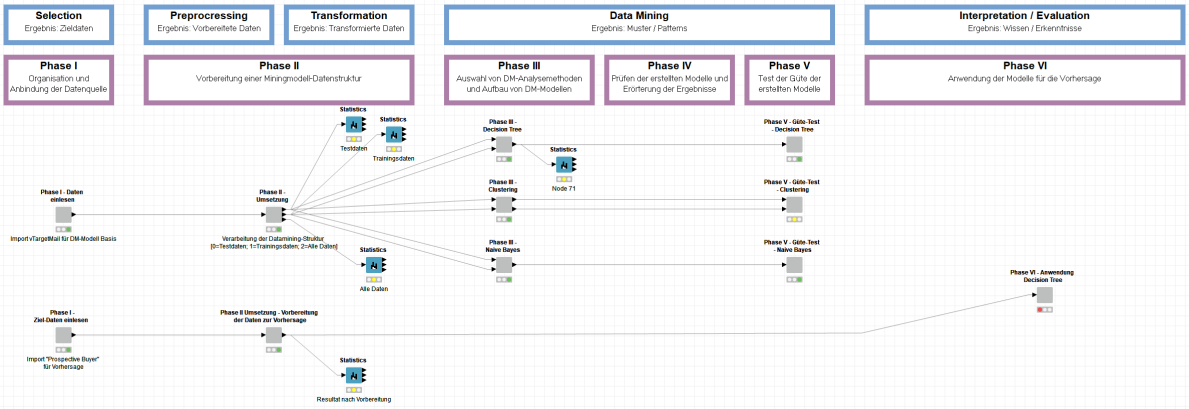


Abbildung 3.1: Überblick Version 5

Zwischen den Versionen 6 und 9 wurde die Datenvorbereitung noch weiter ausgefeilt indem Klassen unterteilt und auch nominale Daten in ordinale transformiert wurden. Letzteres geht bei den Merkmalen Education und Occupation nur dadurch, dass man von Beginn an sagt, dass man eine einfache schulische Ausbildung mit einer geringeren Kennzahl interpretiert, als z.B. einen Hochschulabschluss. Dieses Verfahren kann nicht bei allen nominalen Daten genutzt werden, da im Allgemeinen z.b. bei Farben keine Wertung definierbar ist. Das Merkmal Education kann hier transformiert werden, da idR. die Qualität der Ausbildung deutliche Unterschiede aufweisen kann und so eine höhere Qualität auch mit einer höheren Zahl bewertet werden kann. In Version 7 und 8 wurden neben Datenfluss-Steuerung auch noch weitere Tests an den Merkmalen und der Nutzbarkeit von noch nicht bedachten Merkmalen unternommen. So konnte z.B. entdeckt werden, dass die Daten zur Modell-Bildung alle samt nicht von „echten“ Kunden stammen, sondern von AWC, dem fiktiven Auftraggeber selbst: Die Idee war, dass AWC auch den B2B-Sektor als Markt nutzt und somit Spezialangebote an Firmen senden könnte, denen besonders viele Käufer angehören. Die Information, zu welcher Firma ein Kunde gehört, könnte durch die Mail-Adresse herausgefiltert werden - wenn man öffentliche Webmail-Dienste von der Analyse zuvor ausschließt. Leider war ausnahmslos bei allen Datensätzen die URL „adventure-works.com“ womit sich die neue

Frage gibt, wie zuverlässig die Daten zur Modellierung für echte Kunden sind, da es sich hierbei wohl um Mitarbeiter der Firma handelt - sofern AWC nicht zusätzlich einen Webmail-Dienst betreibt oder allen Kunden, die ihre Daten hergeben, einen eigenen Mail-Account bereitstellt. Diese Information wird für die Weitere Analyse und Bearbeitung nicht weiter als relevant angesehen. Version 9 hat als Implementierung den ersten Schritt unternommen um das Clustering-Modell zur Vorhersage zu nutzen. Dabei wird jedem Cluster über die Analyse des „BikeBuyer"-Merkmals den Modus der Merkmalsausprägung (1 = Käufer, 0 = kein Käufer) zugewiesen und dank dem Mittelwert kann hier direkt dem Cluster ein „Wahrscheinlichkeitswert" zugewiesen werden. Sind z.b. 80 von 100 Datensätzen bei Cluster 1 als BikeBuyer=1 bekannt, so ist die Wahrscheinlichkeit, dass die in den Zieldaten dem Cluster 1 zugewiesenen Personen auch Käufer werden, bei 80 %.

Die Version 10 kann insofern als herausragend bezeichnet werden, als dass die weitere Behandlung davon den Rahmen dieser Studienarbeit sprängen würde. Hier ist die Methode des Baggings für Decision Tree und Boosting für Naive Bayes geprüft worden. Bagging wie auch Boosting sind Methoden aus dem s.g. Ensemble Learning, bei dem ein Zusammenschluss von einer Vielzahl von Modellen (der gleichen Art) auf unterschiedliche Teilmengen der Trainingsmenge trainiert werden, wobei sich diese Mengen auch überschneiden können. Boosting ist kurz gesagt ein erweitertes Verfahren, bei welchem die falsch-positiv- und falsch-negativ-klassifizierten Daten erneut klassifiziert werden und so versucht wird die Fehlerrate zu minimieren und auch weitere neue Erkenntnisse zu gewinnen. Bei beiden Methoden entscheidet am Ende die „Mehrheit" (also das Ensemble) über den Vorhersage-Ausgang. (S. 222ff [Ras17])

In den Versionen 11 bis 15 wurde mit Hilfe von Flowvariablen, die Knime zur Parametrisierung des Workflows bereitstellt, ergänzt, so dass z.B. die Modelle pro Version gespeichert (und wieder verwendet) werden können. Die Modelle der Version 14 bekommen z.B. den Prefix „v14" an die Datei gehängt. Außerdem wurde zur besseren Auffinden der optimalen Parameter das s.g. „Parameter Optimization Loop Start" bzw. -End genutzt und eine Statistik über die Einzelnen Einstellungen durchgeführt. Da das berechnen von Clustern einige Zeit in Anspruch genommen hat, wurde so über Nacht ein paar hundert Cluster berechnet und es konnte leicht festgestellt werden, dass sich mit der Anzahl der Cluster, die Genauigkeit der Vorhersage steigern lässt.

Da der Loop nur über numerische Werte iterieren kann, musste hier für den Decision Tree eine Tabelle mit den übrigen Merkmalen zur Auswertung herangezogen werden (Siehe Abbildung 3.2). Was der Loop bei Clustering und bei Naive Bayes für optimal-Entscheidungen beigetragen hat, wird in Phase III weiter erläutert.

	A	L	M	N	O	P	Q
1	**Einsatz des Parameter Optimizat**						
3	**Versuch Nr.**	**11**	**12**	**13**	**14**	**15**	**16**
4	Quality measure	Gain	Gain	Gain	Gain	Gain	Gain
5	Pruning method	MDL	MDL	No	No	No	No
6	Reduce Error pruning	Yes	No	Yes	No	No	Yes
7	Average Split point	No	Yes	Yes	Yes	No	No
8	Min number records (bei max. Accuracy)	5	10	8	11	28	7
9	Genauigkeit (max):	74,00%	75,20%	75,40%	**75,50%**	74,50%	74,60%
10	Ähnlichkeit / Genauigkeit wie						
11	Min number records:	Genauigkeit:	Genauigkeit:	Genauigkeit:	Genauigkeit:	Genauigkeit:	Genauigkeit:
16	6	73,90%	0,749	74,50%	74,10%	72,30%	72,90%
17	7	73,60%	0,749	75,20%	75,00%	74,30%	**74,60%**
18	8	73,60%	0,749	**75,40%**	75,20%	74,20%	74,20%
19	9	73,60%	0,751	74,90%	74,80%	73,70%	73,70%
20	10	73,70%	**0,752**	75,30%	75,20%	74,10%	74,10%
21	11	73,60%	0,751	75,00%	**75,50%**	74,30%	73,90%
22	12	73,30%	0,749	73,90%	74,40%	73,10%	72,70%

Abbildung 3.2: Auswertungsergebnis Decision Tree

Version 16 wurde wieder für eine weitere Idee genutzt, mit deren Hilfe die Modelle insgesamt eine noch höhere Genauigkeit erzielen konnten (82-92,3 %). Da dies jedoch darauf beruhte, dass einfach alle Merkmale in ein Merkmal konkateniert und anschließend diese „neuen“ Merkmalsausprägungen über eine Gruppierung des Merkmals „BikeBuyer“ zu 100 % „BikeBuyer“=Y und N sonst klassifiziert wurde, was schließlich auf der Annahme beruht, dass die Käufer, die eindeutig klassifizierbar sind, tatsächlich auch repräsentativ für Käufer stehen können, ist diese Vorangehensweise nicht korrekt umsetzbar, da hier zu wenig Daten pro gefundene Ausprägung existieren, dass diese Annahme bekräftigt werden könnte. Sollte ein solches Expertenwissen tatsächlich eingetragen werden können, hätte dies neben der Erhöhung der korrekt-positiv und korrekt-negativ Vorhergesagten Daten zudem zur Folge, dass es weniger Daten aus der Zieldatenmenge als „Käufer“ klassifiziert (Hier wurden ca. 250 von rund 2000 Datensätzen als mögliche Käufer identifiziert. Da wir hier kein echtes Expertenwissen haben, verstärkt das Modell lediglich seine Genauigkeit auf die Quelldaten und bildet damit ein zu maschinell erstelltes genaues Bild der Datenmenge. Hier spielt die Genauigkeit der Maschine gegen die Individualität der Menschen. Die Idee wurde nicht weiter verfolgt.

Version 17 und 18 dienten letztendlich dem Abschluss des Modells. Dabei war das Augenmerk auf die Umsetzungsqualität, die durch Hinweise und Warnungen, die Knime mitteilt, gefördert werden konnte. So war z.B. die Warnung „*WARN Naive Bayes Learner 0:176:11 The following columns will possibly be skipped due to too many values: Age [Binned] (count: 15)*“ der Hinweis, dass der Naive Bayes Learner die Einstellung hatte, keine Merkmale zu nutzen, die mehr als 15 unterschiedliche Merkmalsausprägungen aufweisen (Einstellung beim Naive Bayes Learner) und dass das Merkmal „Age [Binned]“ insgesamt 15 unterschiedliche Merkmalsausprägungen besitzt. Im weiteren wurde noch mehr Zeit in die Analyse der Daten investiert, wobei durch Korrelations- und Verteilungsanalysen einige Punkte Interesse geweckt haben. Näheres dazu in Phase II.

Es folgt die Auflistung der einzelnen Modellversionen mit einer Kurzbeschreibung noch einmal zur **Übersicht des Projektverlaufs**:

V001: Excel laden und Spalten in nominale, ordinale und metrische Daten umwandeln.

V002: Decision Tree als erstes Modell und Bewertung durch Scorer.

V003: Clustering als zweites Modell und Zieldaten laden.

V004: Naive Bayes als drittes Modell und Modelle als Datei exportieren.

V005: Mehr Übersicht durch Wrapped Metanodes.

V006: Klasseneinteilung durch automatisches und manuelles numerisches „Binning“.

V007: Case Switches zur Datenfluss-Steuerung und Normalisierung der Daten.

V008: Dimensionsreduktion durch Ausschluss fehlender Merkmale.

V009: Logik-Verknüpfung beim Clustering zur Nutzung als Vorhersagemodell.

V010: Experimente mit Boosting und Bagging.

V011: Einsatz von erstem Parameter Optimization Loop bei Decision Tree.

V012: Modell-Aufbau vereinfachen, Case Switches entfernen.

V013: Modell-Versionierung durch Flowvariablen und Entscheidungsbaum komplett binär.

V014: Flowvariablen nutzen ⇒ Dateien unter KNIME://local/ im Workspace direkt speichern.

V015: Umsetzung der Güte-Prüfung und Analyse der Übereinstimmung der Vor-

hersage.

V016: Experiment mit „Expertenwissen“.

V017: Knime Warnungen und Hinweise prüfen und behandeln.

V018: Trennung nach Modellbildung und Modellanwendung.

3.2 Beschreibung der Phasen in der Umsetzung

Die Beschreibung der Phasen findet nun aufgrund des Modells der Version 18 statt. Dieses wurde auch am 29.06. an der TAE als finales Modell präsentiert.

3.2.1 Phase I - Organisation und Anbindung der Datenquelle

In der letzten Version der Umsetzung wurden alle oben genannten „Wrapped Metanodes“ in einfache Metanodes umgewandelt. Dies hatte zwei Vorteile: Zum einen wird noch deutlicher Visuell zurückgemeldet, dass ein gekapselter Workflow komplett durchgelaufen ist, zum anderen wird die Nutzung der Flow-Variablen damit ohne weitere Einstellungen ermöglicht.

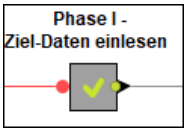


Abbildung 3.3: Korrekt ausgeführtes Metanode

Das Knime-Node „Excel Reader (XLS)“ gibt in den Einstellungen die Möglichkeit, die Excel-Datei und das Arbeitsblatt einzustellen, was geladen werden soll. Diese Einstellung wurde als Knime Workflow Variable global definiert und sowohl für die Quelldaten als auch für die Zieldaten konfiguriert.

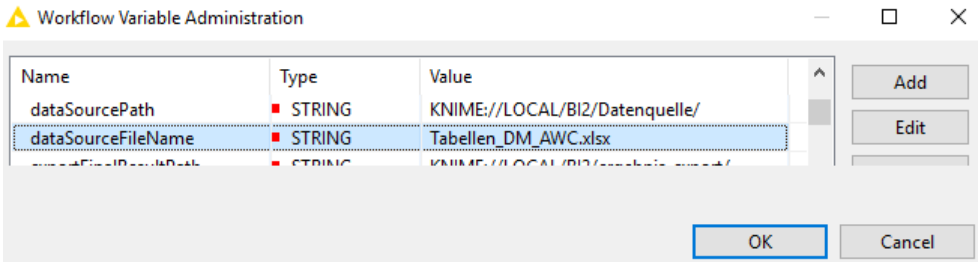


Abbildung 3.4: Fensterausschnitt Workflow Variablen Administratonsfenster

Um die Merkmale sowohl von den Quelldaten korrekt identifizieren zu können, wird der „Column Rename“-Node in dieser Phase genutzt um alle Spalten der Tabelle korrekt zu benennen.

3.2.2 Phase II - Vorbereitung einer Miningmodell-Datenstruktur

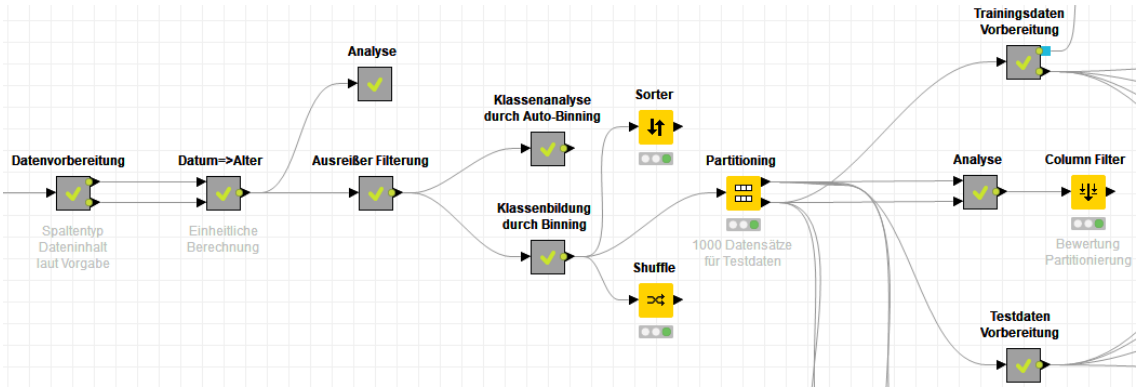


Abbildung 3.5: Ausschnitt Knime Workflow zu Phase II

Der Aufwand, die Daten mittels einem iterativem Prozess vorzubereiten, liegt nach Erfahrungen von [CL16] bei rund 80 %. Es ist enorm wichtig schon beim Vorbereiten der Daten (Datenvorverarbeitung + Datentransformation im KDD-Prozess) die notwendige Zeit zu investieren, da sonst das Prinzip GIGO gilt: garbage in, garbage out. (Siehe [CL16] S. 204)

Datenvorbereitung

In diesem ersten Schritt werden die Datensätze in ihrer Dimension reduziert indem durch den „Column Filter“-Node nur die für das Data Mining relevante Spalten selektiert werden. Zudem werden durch einige weiter Vorgänge die Daten in ihre Datentypen übersetzt, da beim Auslesen aus Excel die Werte alle als String ausgelesen wurden. Ein weitere Maßnahme ist es mittels „Row Filter“-Node die Zeile zu entfernen, die nur den String „NULL“ in jeder Spalte beinhaltete. Bei der statistischen Prüfung mittels „Statistic“-Node konnten die weiteren Merkmalsausprägungen über die Funktion „Occurences Table“ analysiert werden. Wären noch weitere unangebrachte Werte vorhanden wie z.B. Lücken, also fehlende Werte, so könnten diese mittels „Missing Value“-Node weiter behandelt werden.

Im Anschluss an die erste Filterung wurde mittels „String Replace (Dictionary)“-Node einige Merkmale mit Boolean-Datentypen übersetzt um einheitliche Merkmalsausprägungen zu erhalten. So wurden 1/0 übersetzt in Y/N und auch das Merkmal MaritalStatus wurde übersetzt in Verheiratet:=Y/N. Knime gibt dabei die Möglichkeit, dass mehrerer Werte in einen Wert überführt werden kann, so dass es genügte eine Dictionary-File in Form einer CSV anzulegen, welche alle Spalten bediente: BikeBuyer, HouseOwnerFlag, MaritalStatus und z.b. „Y,1,M“ die Übersetzung von entweder „1“ oder „M“ in den Wert „Y“ durchführte.

Als weiterer Punkt, der allerdings in einem eigenen Metanode behandelt wurde, galt es ein konkrete Datenbasis für die Merkmalsausprägung „Alter“ zu finden, da in den Zieldaten die Angabe nicht numerischer Form vorlag, sondern im Datumsformat. Es wurde festgestellt, dass der 24.04.2019 als Datum eine Übereinstimmung aller als Merkmal „Age“ eingetragenen Werte ergab, so dass dieses Datum auch für die Alter-Berechnung bei den Zieldaten einsetzbar galt.

Analyse

Knime bietet eine Vielzahl an Nodes zur Visualisierung oder schnellen Analyse von Daten. Als erster Schritt der Übersicht kann z.B. mittels „Sunburst Chart (JavaScript)“-Node interaktiv über die einzelnen Merkmalsausprägungen per Maus die gesamte Verteilung betrachtet werden.

Sehr interessant ist auch die Analyse von möglichen Korrelationen. Hier kann z.B. das Modell vereinfacht werden, wenn Merkmale korrelieren, kann dies schon mal eine erste Information sein. Mit dem „Linear Correlation“-Node kann dies in Knime bereits grafisch aufbereitet dargestellt und analysiert werden. Die in Abbildung 3.6 abgebildete Korrelation-Matrix zeigt z.b. eine rund 48 %-ige Korrelation zwischen dem Merkmal „Age“ und „TotalChildren“.

Eine weitere Untersuchung gilt der Verteilung der Merkmalsausprägungen bei den Möglichen Gruppierungen von des Merkmals „Bikebuyer“. Wenn alle Merkmale mittels GroupBy zusammengefasst werden, kann am Ende in dieser Merkmalskette die damit abhängige Verteilung untersucht werden. Da dies in der Präsentation nicht betrachtet wurde, wird in dieser Form nur die Ergebnisse in Tabellarisch-Grafischer Form wiedergegeben. Als Fazit kann genannt werden, dass die Käufer tendenziell keine Kinder mehr Zuhause haben und 1 oder weniger Autos besitzen, dafür aber im Mittel ein höheres Einkommen, als Nichtkäufer, oder die „gemischte“ Gruppierung (YN).

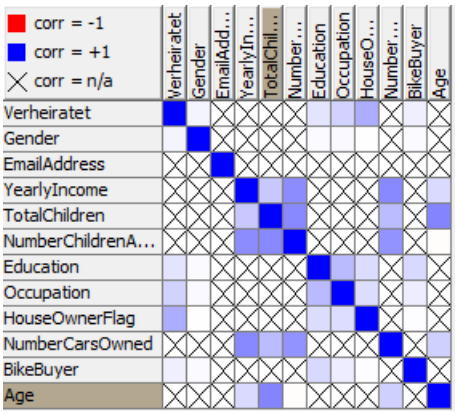
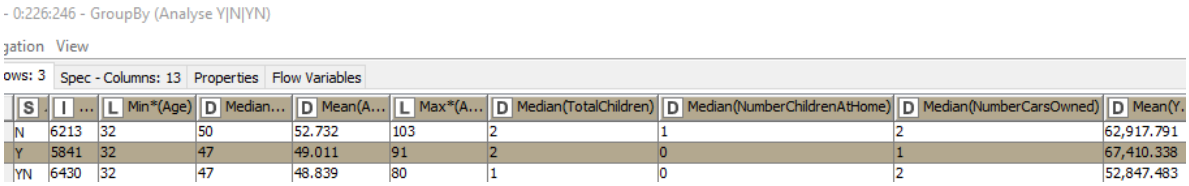


Abbildung 3.6: Korrelation-Matrix



S	I ...	L Min*(Age)	D Median...	D Mean(A...	L Max*(A...	D Median(TotalChildren)	D Median(NumberChildrenAtHome)	D Median(NumberCarsOwned)	D Mean(Y.
N	6213	32	50	52.732	103	2	1	2	62,917.791
Y	5841	32	47	49.011	91	2	0	1	67,410.338
YN	6430	32	47	48.839	80	1	0	2	52,847.483

Abbildung 3.7: BikeBuyer-Analyse per GroupBy-Node

Ein weiterer wichtiger Punkt in der Analyse ist das Finden von Ausreißern. Knime bietet dazu den „Numeric Outliners“-Node an um numerische Ausreißer berechnen zu lassen. Abbildung 3.8 zeigt die Funde mit der Standard-Einstellung von Knime. Offen bleibt die Frage, wie mit den gefundenen Ausreißern umgegangen werden soll.

Row ID	S Outlier column	I Member count	I Outlier count	D Lower bound	D Upper bound
Row2	YearlyIncome	18484	309	-30,000	130,000
Row8	NumberCarsOwned	18484	1261	-0.5	3.5
Row9	Age	18484	82	14.5	82.5

Abbildung 3.8: Numeric Outliners

Nicht weiter eingegangen wird auf die Analyse der Werteverteilung der einzelnen Merkmale. Allein durch die Ausreißer ist hier bereits eine mögliche Aussage durchführbar, dies gibt allerdings nicht wieder, ob es z.B. eine Normalverteilung im Merkmal „YearlyIncomme“ gibt (Hier konnte festgestellt werden, dass es eine deutliche Lücke in den Datensätzen bei der Ausprägung „50.000“ gibt.).

Ausreißer Filterung

Wie mit Ausreißern zu verfahren ist, bleibt immer ein wichtiges Diskussionsthema. Zum einen können Ausreißer aufgrund Falschangaben oder aber auch technischen

Schwierigkeiten passieren (z.B. System Englisch, statt Deutsch, dann wird Komma als Tausender-Trennzeichen interpretiert). Da im Kontext der Daten die Information über die Herkunft und Beschaffungsweise der Daten fehlt, kann höchstens auf die Annahme gesetzt werden, dass eine zu geringe Erscheinung einer Merkmalsausprägung im Verhältnis der möglichen Ausprägungen ungünstig für ein verallgemeinerungsfähigendes Modell sein kann. Daher wurde experimentiert und bestätigt, dass der Naive Bayes bessere Werte beim „Scorer"-Node liefern kann, wenn in den Merkmalen „Age" und „NumberCarsOwned" die gefundenen Ausreißer entfernt werden. In Knime wurde dies durch einen „Row Filter"-Node umgesetzt.

Klasseneinteilung

Mit einer Klasseneinteilung sollen Werte zusammengelegt werden, die entweder innerhalb gleich-breitem Wertebereich liegen oder die möglichst gleiche Anzahl an Elementen beinhaltet. Mit dem „Auto-Binner"-Node kann in Knime erstmal eine der Einstellungen gewählt und das Ergebnis über ein „GroupBy"-Node geprüft werden. Da beim Auto-Binner nur Bereiche eingeschlossen werden, die in den Trainingsdatensätzen existieren, kann dies bei Anwendung auf die Zieldaten zu fehlenden Daten führen, weshalb die Bereiche durch „Numeric-Binner"-Node manuell nachgebaut wurden. Beim Merkmal „YearlyIncome" wurde dabei fast jede Merkmalsausprägung bis „80.000" in eine eigene Bin gefasst, da die Werte schon sehr den Anschein haben, dass sie in ihrer Quelle bereits aggregiert bzw. zusammengefasst wurden. Eine eigene Bin für Werte jenseits „110.000" sorgt allerdings zudem als eine weitere Möglichkeit in der Behandlung von Ausreißern.

Analyse der Partitionierung

Damit die Aufteilung der Daten in Trainings- und Testdaten gewisse Ähnlichkeit besitzen ist es dienlich diese aufgeteilten Daten zu analysieren und entsprechend zu bewerten. Dies kann sehr einfach getestet werden, wenn die Daten vor der Partitionierung sortiert und beim Partitionieren einfach die ersten Datensätze genutzt werden. Als Berechnungskriterium des Ähnlichkeitsmaßes wurde entschieden die Euklidische Distanz zwischen den Trainingsdaten und den Testdaten zu berechnen, was durch Einsatz von „GroupBy"-Node, einem „Joiner"-Node und mehreren „Java Snippet"-Nodes ermöglicht wurde. Im weiteren wurden „Statistics"-Nodes eingesetzt um die Merkmalsausprägungs-

verteilung zu überblicken.

Weitere Vorbereitung für Decision Tree und k-Means

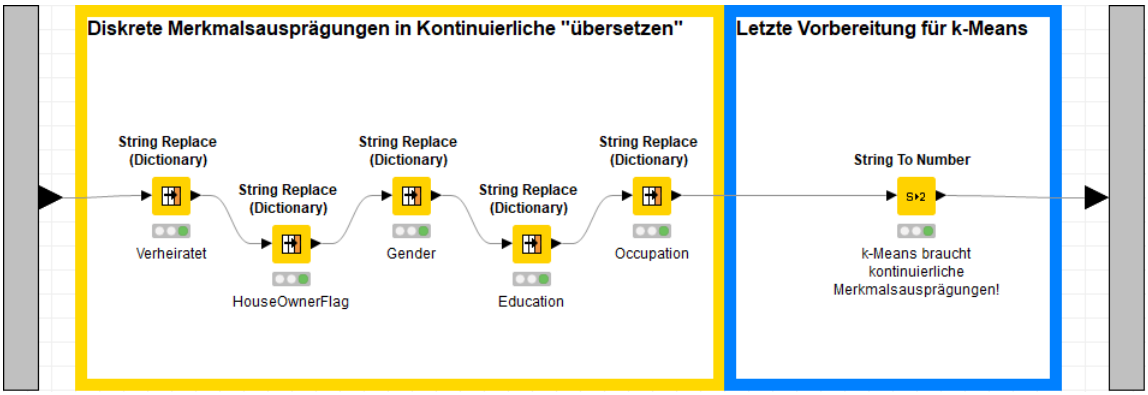


Abbildung 3.9: Letzter Schritt der Transformation von nominalen in ordinale Werte

In Abbildung 3.9 sieht man den letzten Schritt der Datenvorbereitung. Hier wird eine weitere Transformation der Daten vom Modell k-Mean gefordert. Dieser kann nur mit ordinalen oder metrischen Daten funktionieren, weshalb, wie bei der Versionierung (Siehe 3.1.2) oben beschrieben, alle Daten in numerische Daten übersetzt werden müssen. Um alle Daten in metrische umzuwandeln wurde zudem der „Normalizer“-Node eingesetzt(nicht auf dem Teilausschnitt enthalten), so dass alle Merkmalsausprägungen aller Daten zwischen 0 und 1 liegen. Insbesondere war dann interessant, dass der Decision Tree in Knime wohl in dieser Konstellation auch bessere Ergebnisse lieferte.

3.2.3 Phase III - Auswahl von DM-Analysemethoden und Aufbau von DM-Modellen

Als Modelle werden nun nacheinander Decision Tree, k-Means und Naive Bayes im Knime-Kontext vorgestellt. Dabei werden die in Abbildung 3.10 abgebildeten Metanodes „Optimizer-Loop“ erst später in Unterabschnitt 3.2.4 näher beleuchtet.

Decision Tree

Der Entscheidungsbaum (Abbildung 3.11) ist eine der einfachsten Möglichkeiten Regeln abzuleiten, da die Abbildung der Entscheidungen durch den Baum in grafischer Form möglich ist.

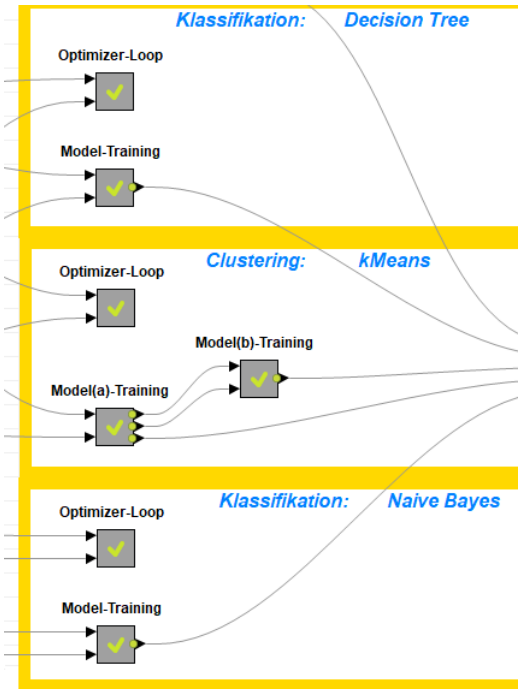


Abbildung 3.10: Ausschnitt Knime Workflow zu Phase III

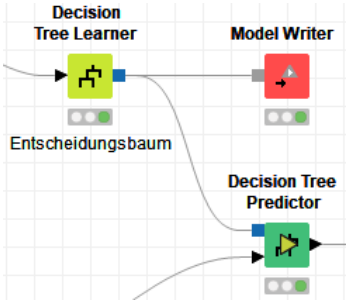


Abbildung 3.11: Decision Tree Workflow in Knime

In Abbildung 3.12 kann z.B. die folgende Regel abgeleitet werden:

Education<=0,125 UND Occupation>0,75
DANN Vorhersage(BIKEBUYER=Y) := 76,5%

Hierbei ist allerdings auffällig, dass eine sehr geringe Repräsentation in der Gesamtmenge besteht. In dem dargestellten Entscheidungsbaum wurden 16.109 Datensätze behandelt. Davon wurden am Ende anhand 13 von 17 Datensätzen die Vorhersage mit Y bewertet.

Je tiefer ein Entscheidungsbaum gehen kann, um so genauer hat er die Datenmenge abgebildet und damit quasi auswendig gelernt. Dieses Auswendiglernen bezeichnet

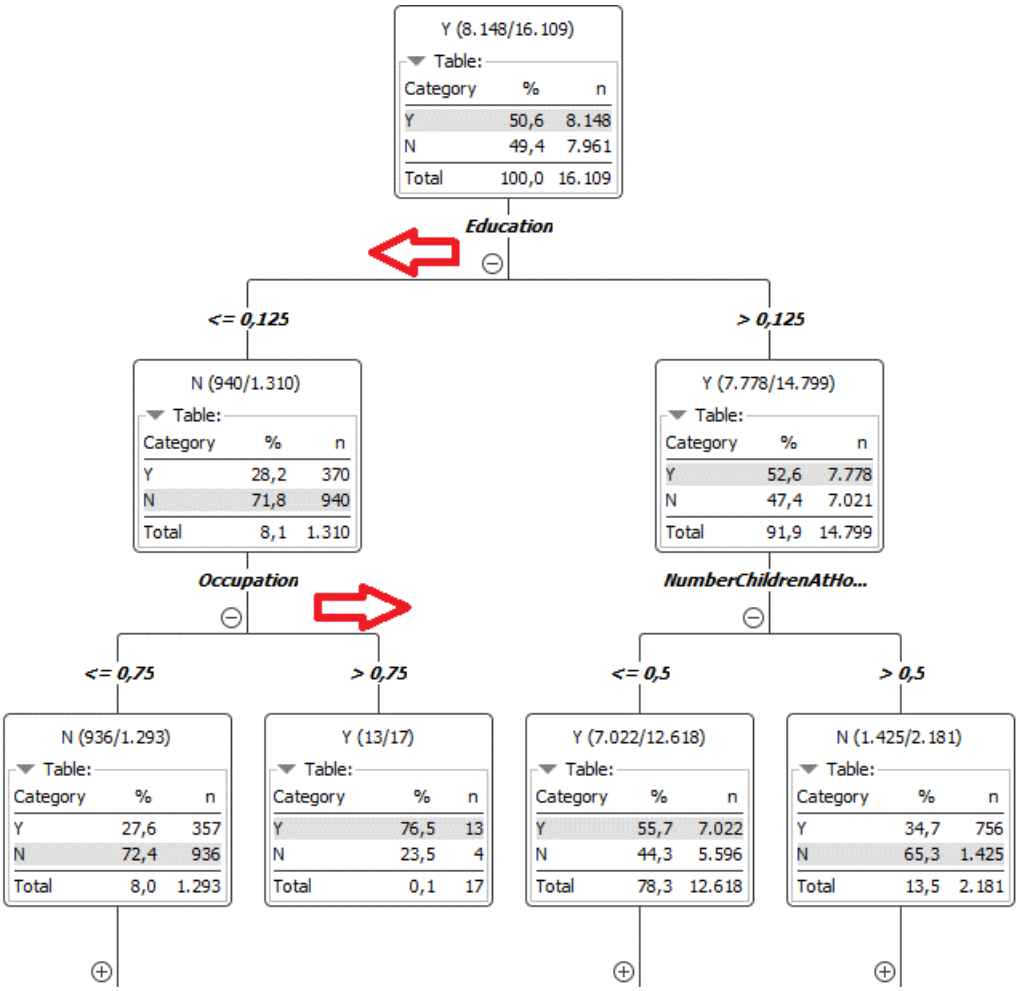


Abbildung 3.12: Ausschnitt Decision Tree Abbild

man als *Overfitting*. Dem kann durch s.g. *Pruning* entgegengewirkt werden. Dabei wird zum einen eine Mindestanzahl an Datensätzen vorgegeben, ab der dann in der Baumgenerierung abgebrochen wird. Zum anderen kann auch zunächst der gesamte Baum generiert und im Anschluss ein Blatt als Aggregation eingesetzt werden. (Aus Kapitel 5.2.7 [CL16]) In unserem Beispielbild wurde das Abbruchkriterium auf 11 Datensätze gelegt, was die Darstellung als korrekt definiert.

Der in Knime implementierte Decision Tree basiert auf dem 1993 von R. Quinlan veröffentlichen C4.5-Algorithmus und wurde seither mit höherer Genauigkeit, neuen Qualitätsmerkmalen und verbesserter Abbruchkriterien, wenn keine weitere Qualitätsverbesserung mehr berechenbar wird. (Frei Übersetzt von [Kni19])

Der C4.5 Algorithmus kann nach [CL16] (Kapitel 5.2.6) metrische Merkmalsausprägungen nicht behandeln und überführt diese automatisch in Intervalle, wodurch sie zu ordinalen Merkmalsausprägungen werden. Genau genommen ist der im Knime-Modell

entstandene Baum ein binärer Baum, in dem die einzelnen Merkmale auch mehrfach hintereinander verschachtelt kommen können.

Naive Bayes

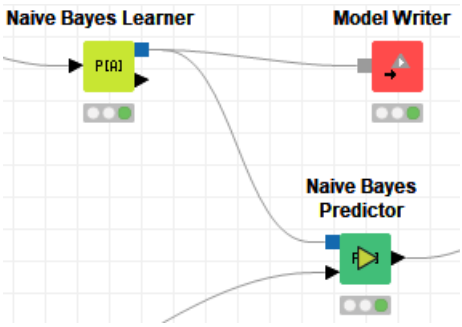


Abbildung 3.13: Naive Bayes in Knime

In Abbildung 3.13 ist die Modell-Generierung mittels Naive Bayes aus Knime dargestellt. Der Naive Bayes basiert auf dem aus der Statistik bekannten Satz von Bayes. In Knime ist der Algorithmus bereits voll implementiert und das Ergebnis kann auch anschließend ausgelesen und auf interessante Informationen geprüft werden.

Abbildung 3.14 zeigt einen Ausschnitt des Naive Bayes Ergebnisses. Das Merkmal Education wird in seine Merkmalsausprägungen unterteilt und die Wahrscheinlichkeit wird nun in Form einer Tabelle nachvollziehbar.

P(Education | class=?)

Class/Education	Bachelors	Graduate Degree	High School	Partial College	Partial High School
N	1864	1314	1638	2205	940
Y	2691	1581	1184	2322	370
Rate:	28%	18%	18%	28%	8%

Abbildung 3.14: Naive Bayes Auszug Ergebnis in Knime

k-Means Standardverfahren

Das Clusteringverfahren k-Means kann in Knime eingesetzt werden um eine Clusteranalyse durchzuführen und so z.B. Gruppierungen erkannt werden, die gewisse Eigenschaften gemeinsam haben.

Direkt bei der Modell-Generierung gibt es allerdings lediglich die Möglichkeit die Anzahl der zu generierten Cluster festzulegen. Da die Erstellung der s.g. *Centroiden*

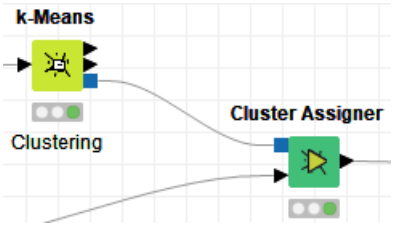


Abbildung 3.15: k-Means in Knime

(Zentrum/Schwerpunkt des Clusters) durch Zufallsdefinition geschieht, kann hier kein Einfluss darauf gesetzt werden, wie die Cluster entstehen.

Ziel des Clustering-Verfahrens ist das technisch unterstützte Auffinden von Gruppierungen, die, sollten sie interessante Informationen aufweisen, für weitere Zwecke genutzt werden können. Findet man z.B. raus, dass es einen großen Cluster mit jungen Erwachsenen, die in einer Stadt wohnen, die für ihre Universität sehr bekannt ist, kann als Schlussfolgerung angenommen werden, dass hier die Generierung eines Angebotes für Studenten sehr sinnvoll sein kann.

k-Means zur Vorhersage

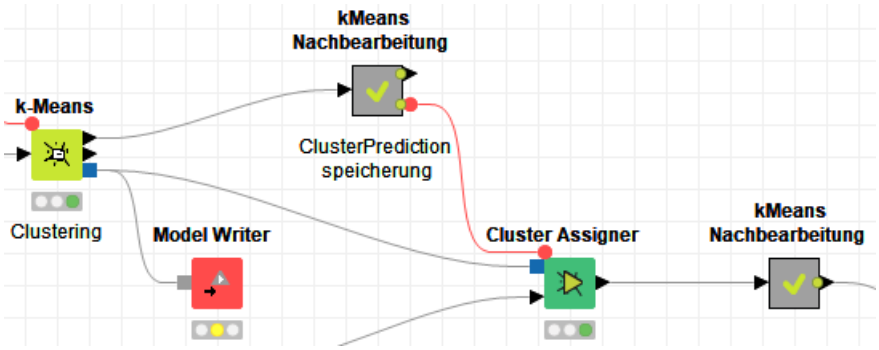


Abbildung 3.16: k-Means zur Vorhersage in Knime

Oben haben wir k-Means bereits zur Untersuchung von Gruppierungen kennengelernt. Da hier ähnlich dem Decision Tree auch quasi eine Zusammensetzung der einzelnen Merkmale stattfindet und sogar durch die Berechnung der Nähe zu einer Ausprägung die Zusammengehörigkeit noch breiter aufgeteilt ist (da nicht nur, wie beim Decision Tree, wenige einzelne Merkmale von Relevanz sein können - bei einem Cluster finden alle Merkmale Beachtung) kann gefolgert werden, dass die Cluster-Analyse zur Klassifizierung und damit zur Vorhersage genutzt werden kann. Da Knime dafür allerdings kein Verfahren bereitstellt, war die Herausforderung die in Knime verfügbaren

und bekannten Nodes so zu nutzen, dass jedem Cluster eine Zuweisung durchgeführt werden kann.

Diese Zuweisung konnte über die Auswertung einer Gruppierung über den „Group By“-Node realisiert werden. Es wird prinzipiell das „BikeBuyer“-Merkmal einmal als Modus und einmal als Mittelwert genutzt. Wenn dann z.b. der Cluster c insgesamt 75 von 100 Datensätzen eine 1 als Merkmalsausprägung von „BikeBuyer“ über die Methoden aggregiert, führt das zu einer 1 als Vorhersage-Ergebnis (Modus) des Clusters c und zu einem Mittelwert von 0.75, was als „Wahrscheinlichkeit“ genutzt werden kann. Abbildung 3.17 zeigt die angewandten Nodes und beispielhaft ein Auszug aus der Cluster-Tabelle, die nach dem „GroupBy“-Node als Ergebnis resultiert.

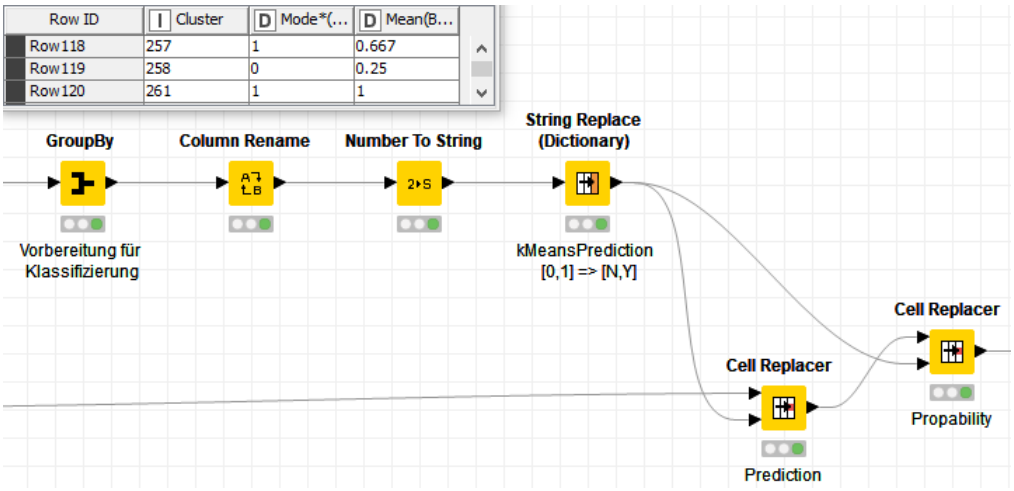


Abbildung 3.17: k-Means per Gruppierung zur Vorhersage

3.2.4 Phase IV - Prüfen der erstellten Modelle und Erörterung der Ergebnisse

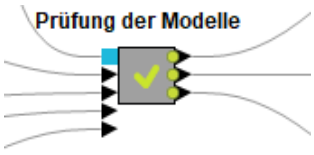


Abbildung 3.18: Knime Metanode zu Phase IV

Die Prüfung der Modelle kann über eine Vielzahl an Möglichkeiten in Knime durchgeführt werden. Da die Vielfalt der Möglichkeiten so groß und für die Einzelnen Modelle

doch unterschiedlich ausfällt, ist auch hier die Möglichkeit genutzt worden, alle Analysen durch Metanodes zu packen. Abbildung 3.19 zeigt hier in einem roten Kasten z.B. das erste und eines der wichtigsten Kriterien bei der Prüfung: Fehlende Werte.

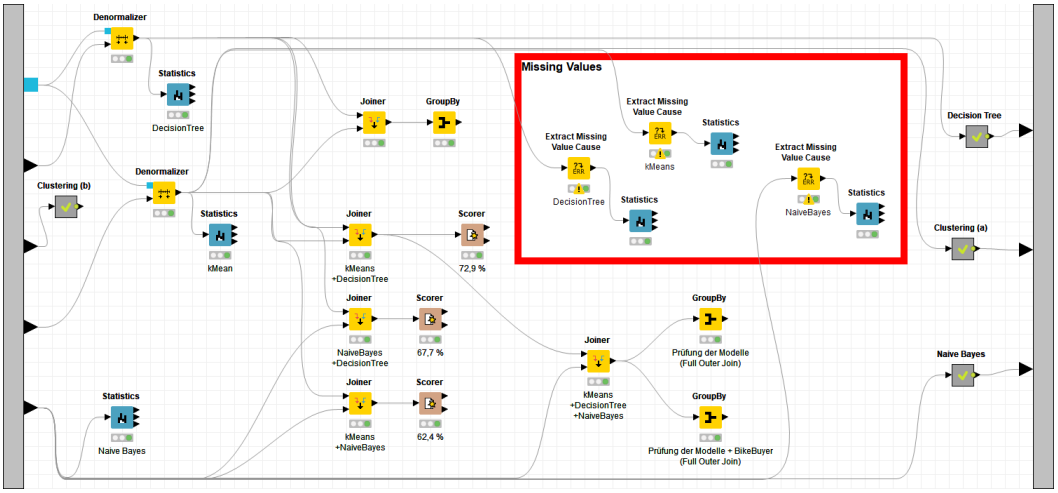


Abbildung 3.19: Ausschnitt Knime Workflow zu Phase IV

Im Laufe des Prozesses sollte es nie zu fehlenden Werten kommen, weil dies sonst eine Lücke im Prozess aufzeigen würde. Generell kam es bei einer Version tatsächlich zu genau diesem Problem, wobei sich schnell herausstellte, dass es an der Definition der Cluster gelegen hat. Zu beginn wurde die Definition der ClusterPrediction über eine Datei festgelegt, in der die Cluster quasi übersetzt wurden. So war z.B. der Ausdruck „Cluster_1“ zunächst per „Column Replacer (Dictionary)“-Node in den Ausdruck „1“ übersetzt worden und anschließend die 1 mit „String to Number“-Node in eine Zahl umgewandelt. Dabei waren lediglich die ersten 10 Cluster in der Dictionary-File zur „Übersetzung“ eingetragen, was die Umwandlung von String to Number von Clustern die > 10 waren, unmöglich machte. Letztendlich konnte diese Übersetzung auch per „Java Snippet (simple)“-Node korrekt umgesetzt und ohne weitere Lücken durchgeführt werden.

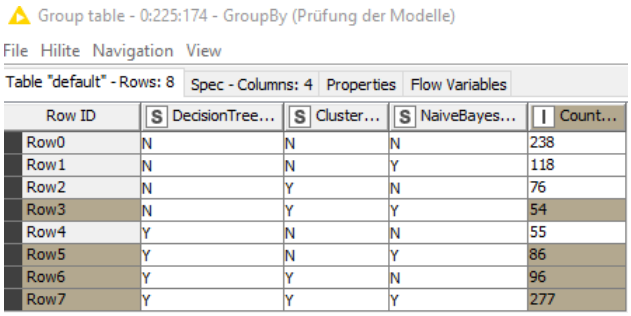
Einige weitere Inhalte von Abbildung 3.19 werden nun genauer beleuchtet.

Vergleich der Ergebnisse

Die Gruppierung aller Ergebnisse ist schnell per „Joiner“-Node und anschließendem „GroupBy“-Node realisiert. Sehr leicht lässt sich hier ablesen, dass von 1000 Testdaten insgesamt $277 + 238 = 515$, also 51,5 % der Ergebnisse, eindeutig von allen Modellen klassifiziert wurde. Dabei sind 277 insgesamt eindeutig als Käufer klassifiziert worden.

In Unterabschnitt 3.1.2 haben wir das Ensemble-Lerning kurz angesprochen. Dabei geht es prinzipiell darum, dass ein Mehrheitsentscheid entscheidet. Wenn wir bei 3 Modellen somit definieren, dass mindestens 2 Modelle die Vorhersage mit einem Ja bestimmt haben müssen, so ist das Ergebnis komplett abgedeckt, da es kein „Unentschieden“ bei einem Modell für sich geben kann und so immer entweder 2 ein Ja oder 2 ein Nein als Mehrheitsentscheid preisgeben.

Wenn wir allerdings nur daran interessiert sind, wie viele von den 1000 Datensätzen als mögliche Käufer klassifiziert werden, kommen wir auf $54 + 86 + 96 + 277 = 513$ Datensätze und somit 51,3 %.


Group table - 0:225:174 - GroupBy (Prüfung der Modelle)

File Hilite Navigation View

Table "default" - Rows: 8 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	DecisionTree...	Cluster...	NaiveBayes...	Count...
Row0	N	N	N	238
Row1	N	N	Y	118
Row2	N	Y	N	76
Row3	N	Y	Y	54
Row4	Y	N	N	55
Row5	Y	N	Y	86
Row6	Y	Y	N	96
Row7	Y	Y	Y	277

Abbildung 3.20: GroupBy Ergebnis über alle Modelle

Die zusätzliche Prüfung, ob die bekannte Information aus BikeBuyer auch „getroffen“ wurde in der eindeutigen bzw. der Ensemble-Vorhersage kommt auf folgendes Ergebnis:

Eindeutig: $260 + 219 = 479$ Datensätze $\Rightarrow$ **47,9 % korrekt klassifiziert**

Bei der Beachtung, dass eine Mehrheit von mindestens 2 Modellen ein Ja oder ein Nein gesetzt wird, führt zu folgendem Ergebnis:

Tabelle 3.1: Übersicht BikeBuyer vs. Ensemble-Methode

BikeBuyer	**Ensemble Ergebnis**	**Anzahl**	**% Gesamt**
Ja	Ja	$40 + 26 + 91 + 260 = 417$	41,7 %
Ja	Nein	$19 + 8 + 51 + 18 = 96$	9,6 %
Nein	Ja	$14 + 60 + 5 + 17 = 96$	9,6 %
Nein	Nein	$219 + 110 + 25 + 37 = 391$	39,1 %
-	-	Summe: $417 + 96 + 96 + 391 = 1000$	100 %
-	-	**Korrekt:** $417 + 391 = 808$	**80,8** %

Als weitere Prüfung steht nun die Definition an, dass bereits eine einzige positive Bewertung zur Klassifizierung führt, also dass bereits 1 Modell mit einem Ja, als Ergebnis ein Ja verursacht:

Tabelle 3.2: Übersicht BikeBuyer vs. 1-Modell-Gewinnt-Methode

BikeBuyer	**1M-Ergebnis**	**Anzahl**	**% Gesamt**
Ja	Ja	$8+51+40+18+26+91+260 = 494$	49,4 %
Ja	Nein	19	1,9 %
Nein	Ja	$110+25+14+37+60+5+17 = 268$	26,8 %
Nein	Nein	$219 = 219$	21,9 %
-	-	Summe: $494 + 19 + 268 + 219 = 1000$	100 %
-	-	**Korrekt:** $494 + 219 = 713$	**71,3** %

Somit wird deutlich, dass die Zusammenführung der Modelle zusammen mit der Ensemble-Lerning-Methode am Ende durchaus Sinn ergibt. Auch wenn wir hier nur 3 Modelle einsetzen, die der Vorhersage dienen, liegt der Nutzen in der Ensemble-Lerning-Methode hinter der Idee des Mehrheitsentscheids. So werden die einzelnen Klassifizierer zu einem Meta-Klassifizierer zusammengefasst, welcher dadurch „über eine bessere Verallgemeinerungsfähigkeigt verfügt als jeder einzelne Klassifizierer.“ [Ras17]

Würden wir hierbei nur die korrekten Ergebnisse nutzen wollen, die durch unsere gewählte Taktik umsetzbar wäre, hätten wir bei Einstimmigkeit 260, bei einfacher Mehrheit 417 und bei einer einzigen positiven Stimme 491. Halten wir die falsch-positiven Ergebnisse dazu, so sind es bei Einstimmigkeit keine, bei einfacher Mehrheit 96 und bei einer einzigen positiven Stimme 268. Wenn wir von der Einstimmigkeit als Basis ausgehen, haben wir für einen einfachen Mehrheitsentscheid insgesamt 157 Ergebnisse mehr zzgl. 96 möglichen falsch-positiven Ergebnissen, die dann insgesamt zu $\frac{96}{417+96} = 18,7\%$ fälschlicherweise klassifizierten Ergebnissen führen würde. Gleiches bei einer einfach positiv geltenden Stimme sind es mit Basis der Einstimmigkeit ganze 231 mehr Ergebnisse bei 268 falsch-positiven Ergebnissen: $\frac{268}{491+268} = 35,3\%$ falsch-positiv klassifizierten Ergebnissen. Wenn wir dabei berücksichtigen, dass es nur 8,7 %-Punkte unterschied ist, hat die Nutzung der Ensemble-Methode auch in dieser Betrachtung seinen Mehrgewinn gezeigt.

BikeBuyer	DecisionTree...	Cluster...	NaiveBayes...	Count...
N	N	N	N	219
N	N	N	Y	110
N	N	Y	N	25
N	N	Y	Y	14
N	Y	N	N	37
N	Y	N	Y	60
N	Y	Y	N	5
N	Y	Y	Y	17
Y	N	N	N	19
Y	N	N	Y	8
Y	N	Y	N	51
Y	N	Y	Y	40
Y	Y	N	N	18
Y	Y	N	Y	26
Y	Y	Y	N	91
Y	Y	Y	Y	260

Abbildung 3.21: GroupBy Ergebnis über alle Modelle mit BikeBuyer

Prüfung der Decision Tree-Ergebnisse

Zur allgemeinen Bewertung der Modelle wird die Anwendung des „ROC-Curve“-Nodes eingesetzt.

„Die Receiver-Operating-Characteristic-Diagramme (Grenzwertoptimierungskuven [...]) sind ein nützliches Tool, um Klassifizierungsmodelle anhand ihrer Leistung hinsichtlich der Falsch-Positiv- und Richtig-Positiv-Rate auszuwählen. [...] Die Fläche unter der ROC-Kurve, die so genannte ROC AUC (Receiver Operator Characteristic Area Under Curve) dient zur Beschreibung der Leistung eines Klassifizierungsmodells.“ (S. 197 [Ras17]).

Die ROC-Curve des Decision Tree ist mit Abbildung 3.22 aufgeführt. Aus ihr ist erkennbar, dass den größten Einfluss das Merkmal „Education“ auf die richtig-positiv-klassifizierten Merkmalausprägungen hat (blaue Linie).

Resultat Clustering

Beim Clustering besteht die Schwierigkeit bei der gegebenen Datenmenge darin, wirklich konkrete Gruppierungen/Muster erkennen zu können. Die menschlich erlernte Realitätsbefinden beschränkt sich meist auf 4 Dimensionen, wenn wir zu den 3 räumlichen noch die Zeit mit berücksichtigen, oder bei der grafischen Darstellung zu dem 3-D-Modell noch durch Farben eine weitere Dimension darstellbar machen. Bei unserem Modell nutzen wir insgesamt 10 Merkmale (von „BikeBuyer“ als weitere einmal abgesehen), was das Vorstellungsvermögen doch sehr herausfordert.

Mit Knime lässt sich diese Herausforderung durch den s.g. Scatter Plot in 2-D-Matrix-Darstellung dennoch analysieren. Der Aufbau Abbildung 3.23 zeigt den Work-

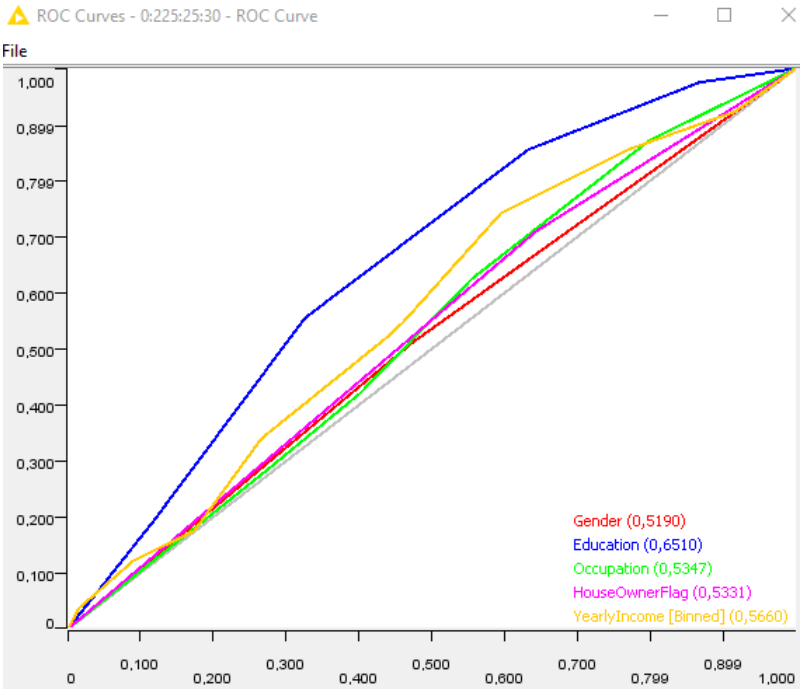


Abbildung 3.22: ROC-Curve für Decision Tree Prediction

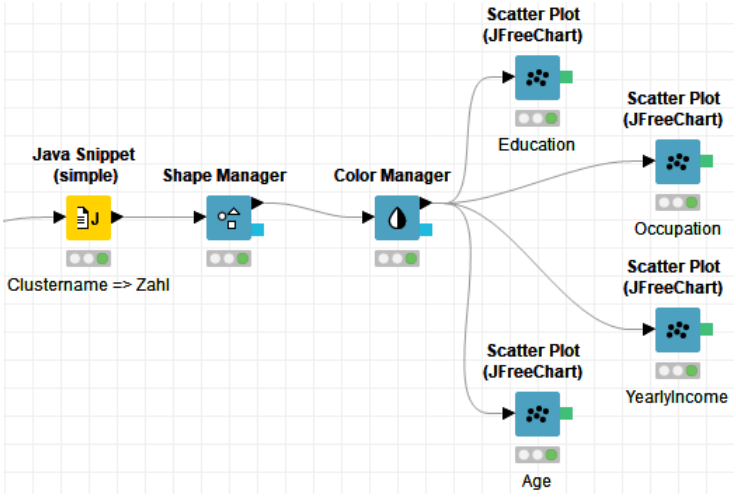


Abbildung 3.23: Ausschnitt Cluster-Analyse

flow für die Erstellung von Scatter Plot Diagrammen. Dabei kann z.B. für die Dimension „BikeBuyer“ die Symbole + und x für Ja bzw. Nein im „Shape Manager“-Node und die Cluster farblich per „Color Manager“-Node definiert werden.

Abbildung 3.24 und Abbildung 3.27 zeigen beispielhaft die Ansicht für 3 Cluster bzw. 7 Cluster. Hier haben wir somit 4 Dimensionen in einem 2-dimensionalen Bild untergebracht. Die Farben sind von Knime speziell für Farbenblinde vordefiniert (Im

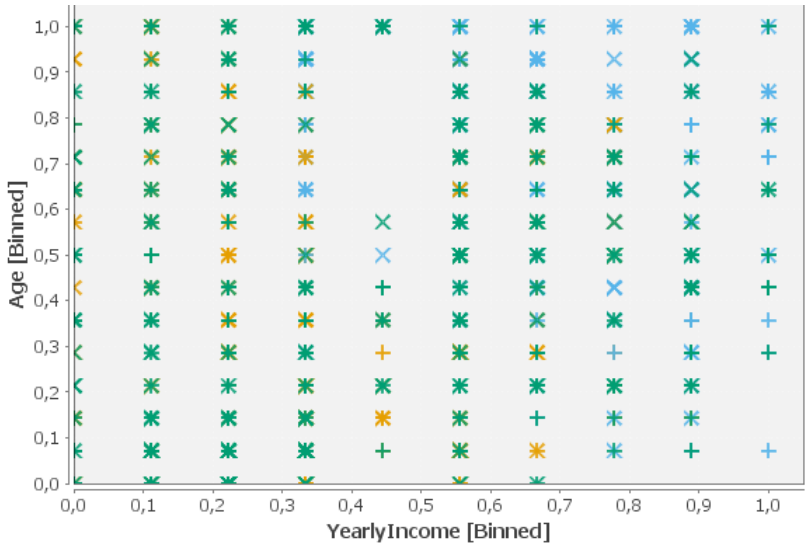


Abbildung 3.24: Scatter Plott (JFreeChart)-Resultat bei 3 Clustern

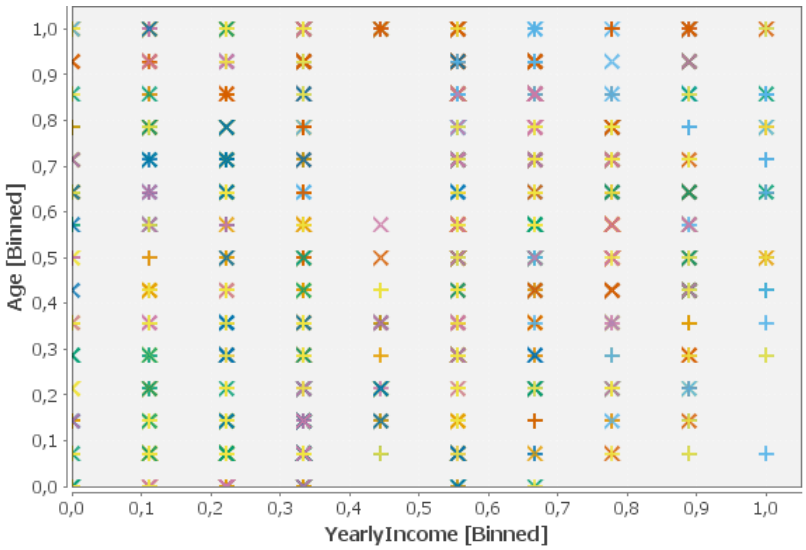


Abbildung 3.25: Scatter Plott (JFreeChart)-Resultat bei 7 Clustern

„Color Manager“-Node auswählbar), können allerdings bei steigender Cluster-Anzahl hier keinen Mehrwert bringen, da irgendwann die Farben ausgehen, die von einem ungeübten Menschen eindeutig unterscheidbar sind.

Bei der Vorhersage mit k-Means werden 1190 Cluster genutzt um eine hohe „Trefferquote“ zu erzielen. Aufgrund der hohen Anzahl wurde hierfür keine grafische Aufbereitung durchgeführt. Stattdessen wurde mittels „Optimization-Loop“ mehrere verschiedene Einstellungen untersucht und per Excel festgehalten. Zum einen kann die Anzahl der Cluster festgelegt werden. Zum anderen können aber auch die Daten in

einer anderen Form über den „Numeric Binning“-Node in Knime entsprechend klassifiziert werden. Letzteres wurde jedoch nicht in die tiefe verfolgt.

Vielmehr wurde das Verhältnis zwischen Anzahl Cluster und Ergebnisse nach der Modell-Anwendung näher untersucht (Abbildung 3.26). Je mehr Cluster definiert werden sollen umso höher wird die Anzahl von „leeren“ Clustern. Dies ist per einfachem Group By in der Ergebnismenge nach dem Clustering über alle Cluster feststellbar. Da über diese Gruppierung das Vorhersage-Modell abgeleitet wird, ist es verständlich wieso später bei der Modell-Anwendung Lücken entstehen können. So kann per Optimization Loop nun die Anzahl Cluster erhöht werden und der Punkt gesucht werden, ab welcher Cluster-Anzahl die Anzahl fehlenden Daten in der Ergebnismenge > 1 wird.

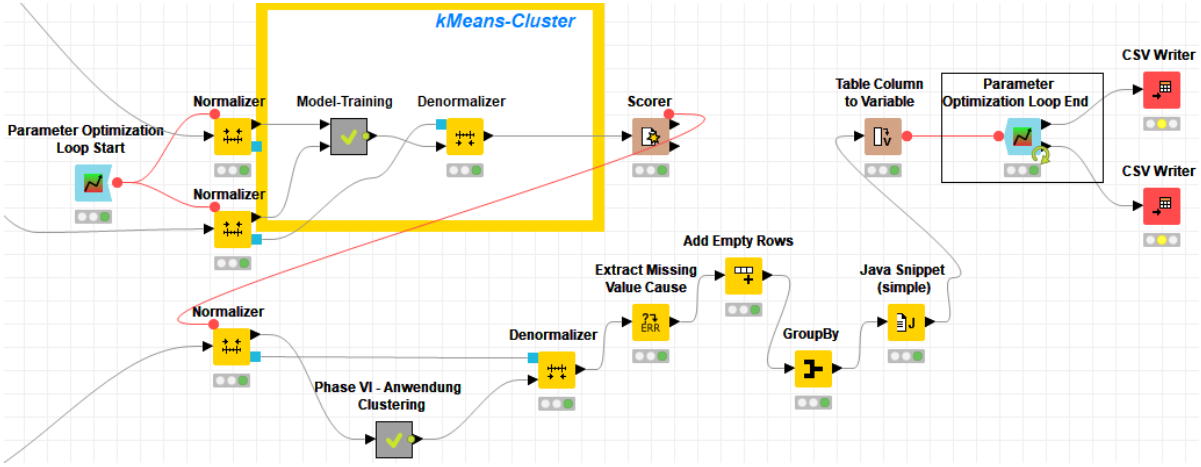


Abbildung 3.26: Clusteranzahl - MissingValue-Anzahl Analyse Aufbau

Bei dieser Aufstellung gab es zunächst die Herausforderung den Einstiegspunkt zu finden, ab welcher Clusteranzahl überhaupt erstmal 1 fehlender Datensatz besteht. Dies konnte in Knime allein mit „Extract Missing Value Cause“-Node nicht bewerkstelligt werden, da der „Parameter Optimization Loop End“-Node bei nicht vorhandenen Daten (was der Fall ist, wenn keine Missing Values existieren) seine Funktion einstellt und mit einer Fehlermeldung abbricht. Dem konnte mit einem „Add Empty Rows“-Node entgegengewirkt werden, der immer eine Zeile einfügt, die dann bei dem „GroupBy“-Node immer vorhanden ist. Mittels „Java Snippet (simple)“-Node kann dann diese selbst hinzugefügte Zeile wieder entfernt werden, so dass der Parameter Optimization Loop eine 0 als Ergebnis bekommen kann.

Abbildung 3.27 zeigt beispielhaft ein Auszug des Ergebnisses eines Durchlaufs von 1170 bis 1220 Cluster in 10er-Schritten.

All parameters - 0:48 - Parameter Optimizati...

File Hilite Navigation View

Table "default" - Rows: 6 | Spec - Columns: 2 | Properties | Flow Va

Row ID	NumberOfClusters	Objective value
Row0	1170	1
Row1	1180	1
Row2	1190	1
Row3	1200	5
Row4	1210	5
Row5	1220	5

Abbildung 3.27: Clusteranzahl - MissingValue-Anzahl Analyse Teilergebnis

Bei der Prüfung der Vorhersage durch ein „ROC-Curve“-Node (Abbildung 3.28) sticht das Merkmal „Education“ auch bei k-Means hervor.

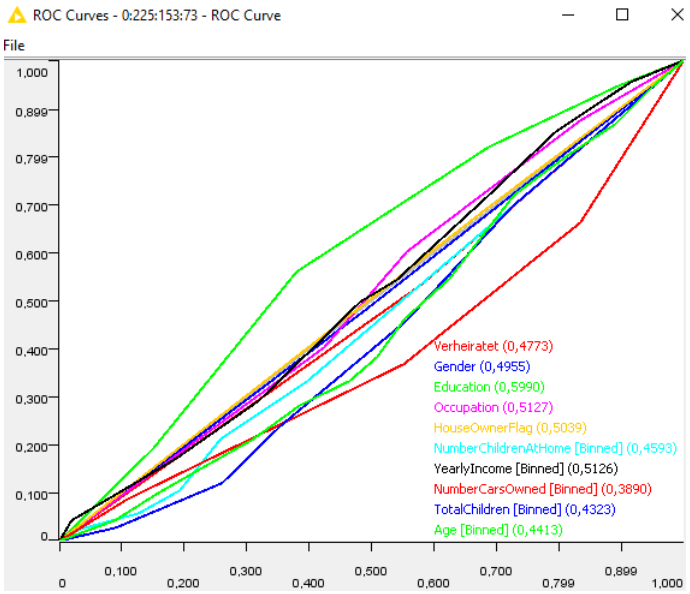


Abbildung 3.28: ROC-Curve für k-Means Prediction

Prüfung der Naive Bayes-Ergebnisse

In Abbildung 3.29 wird gleich wie auch in den anderen Modellen das Merkmal „Education“ hervorgehoben.

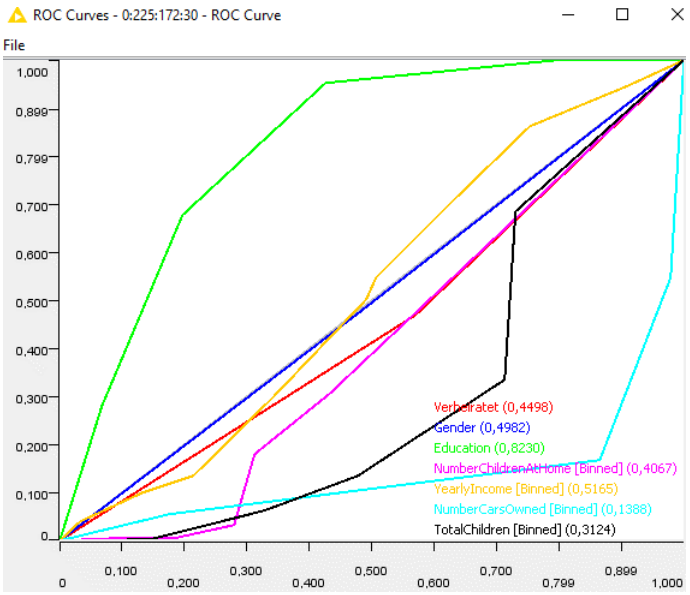


Abbildung 3.29: ROC-Curve für Naive Bayes Prediction

3.2.5 Phase V - Test der Güte der erstellten Modelle

Bei allen Modellen wurde mit einem Optimization Loop das Optimum in einer bestimmten Einstellungs-Konstellation durchgeführt. Im Allgemeinen wurde allerdings überall der „Scorer“-Node eingesetzt, so dass hierdurch eine Vergleichbarkeit möglich wird. Hier soll nun das Ergebnis aus den Güte-Prüfungen aufgestellt werden.

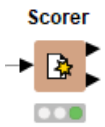


Abbildung 3.30: Scorer-Node aus Knime zur Bewertung der Güte des Modells

Abbildung 3.30 zeigt beispielhaft den „Scorer“-Node aus Knime. Dieser hat als Ergebnis eine Matrix über die Klassifizierung und der Verteilung der Testdaten im Hinblick auf das Merkmal „BikeBuyer“, anhand welchem die Korrekt-Positiv bzw. Korrekt-Negativ-Klassifizierung bewertet wird. Der Aufbau der Ergebnisse wird anhand dem

Decision Tree kurz erläutert.

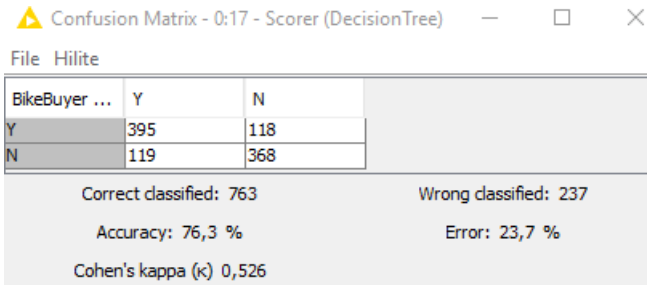


Abbildung 3.31: Scorer-Ergebnis für Decision Tree-Modell

Abbildung 3.31 zeigt die Ergebnisse des Decision Trees. Dabei ist in den Zeilen die BikeBuyer-Zugehörigkeit ausgewiesen und in den Spalten die Zugehörigkeit der Vorhersage. In der Fußzeile wird dann zunächst die Anzahl der korrekt klassifizierten und der falsch klassifizierten aufgeführt, sowie der Quotient aus korrekt-positiv durch Gesamt in Prozent als „Genauigkeit“ (Accuracy) aufgezeigt. Cohen's kappa zeigt, wie zuverlässig die Vorhersage ist, wobei nicht eindeutig erklärt werden kann, welche Werte zu welcher Gewichtung führen kann. Da der Wertebereich maximal bis 1 geht (was gleichzeitig die beste Bewertung darstellt) kann dennoch gesagt werden, je größer, desto besser. [HJS19]

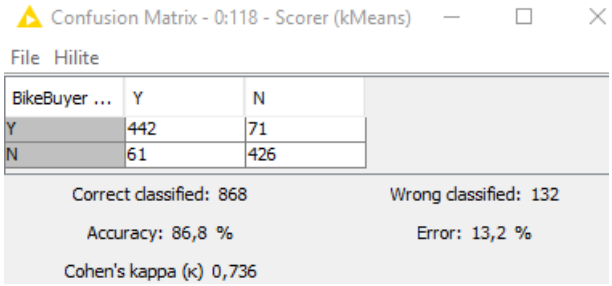


Abbildung 3.32: Scorer-Ergebnis für k-Mean-Modell

Die Bewertung des k-Mean kann in Abbildung 3.32 Beachtung finden. Da der k-Mean mit jeden weiteren Clustern die Trainingsdaten auch immer „besser“ lernt, kann hier die Genauigkeit für Clusteranzahl = Anzahl Gruppierte Datensätze auf 94 % heraufgesetzt werden. (Gruppierte Datensätze bedeutet hier, dass bei mehrere Vorkommen der gleichen Ausprägungskombinationen nur das 1. Vorkommen gerechnet wird, was mit Hilfe eines „GroupBy“-Nodes sehr einfach umsetzbar ist.) Wie oben über Abbil-

dung 3.26 beschrieben ist für den k-Means ein Optimum gesucht worden, so dass die nicht klassifizierbaren Datensätze bei maximaler Anzahl an Cluster minimiert werden.

Im weiteren kann hier das Bedenken der Wirkungsweise sehr wertvoll sein, dass k-Means seine Cluster anfangs zufällig verteilt und damit auch nach jedem Modell-Bildungs-Prozess selbst bei gleich bleibender Datenmenge und Clusteranzahl eine andere Bewertung berechnet wird.

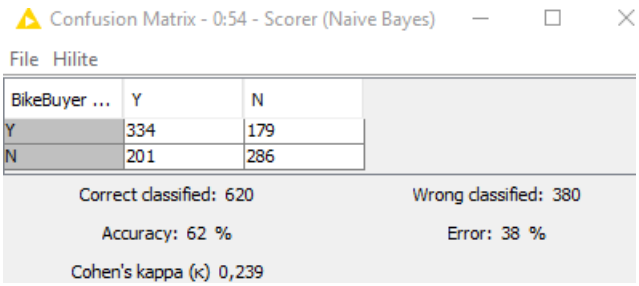


Abbildung 3.33: Scorer-Ergebnis für Naive Bayes-Modell

Abbildung 3.33 stellt die Bewertung des Naive Bayes dar. Hier ist zu sagen, dass der Naive Bayes eine starke Schwankung zwischen 60 und 65 % aufweisen kann, je nach Zufallsauswahl beim erneuten Partitionieren der Daten.

3.2.6 Phase VI - Anwendung der Modelle für die Vorhersage

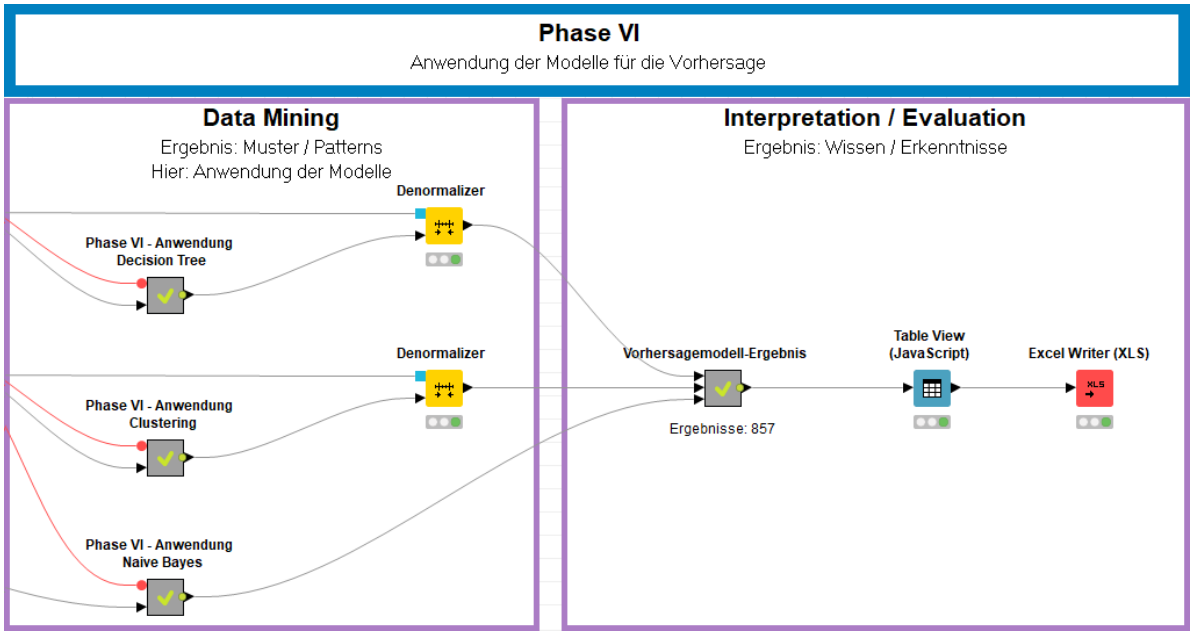


Abbildung 3.34: Ausschnitt Knime Workflow zu Phase VI

In der letzten Phase wurde vergleichbar mit den in den Phasen 1-2 vorgestellten Methoden die Zieldaten geladen, vorbereitet und transformiert. Anschließend werden diese Zieldaten in der vorbereiteten Mining Datenstruktur mit den Modellen bearbeitet und jedes Modell definiert seine Vorhersage per eigene Spalte in dem hiernach entstandenen Ergebnis.

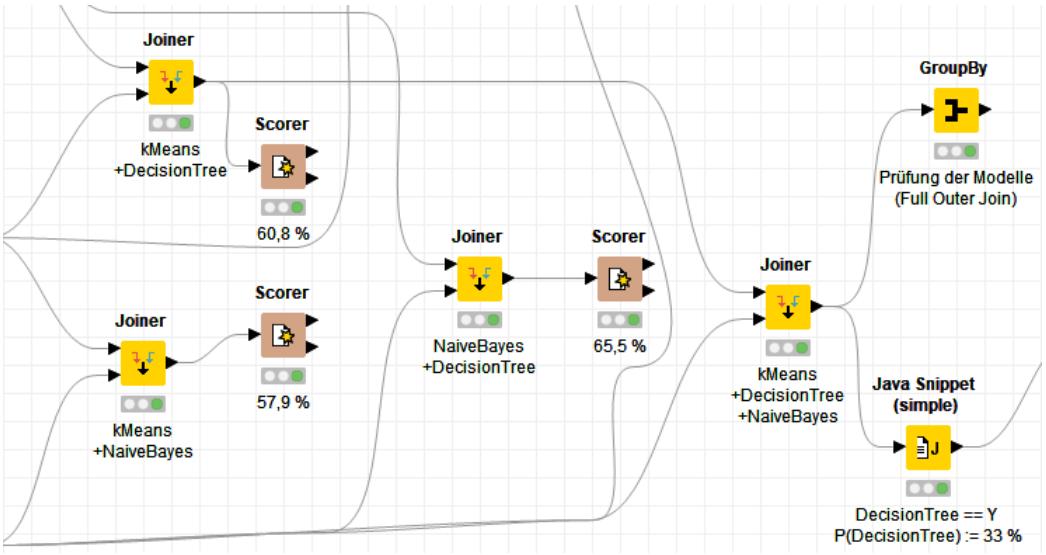


Abbildung 3.35: Aufbau des Ensembles zur Vorhersage (Teil 1/2)

Abbildung 3.35 zeigt den ersten Part des Ensemble-Prozesses, dass alle Tabellen zu Einer per „Joiner“-Node zusammengefasst werden. Dabei werden aus der ersten Tabelle alle Merkmale übernommen, aus den folgenden jeweils nur die zur Berechnung der vorhersage relevanten Vorhersage-Spalte.

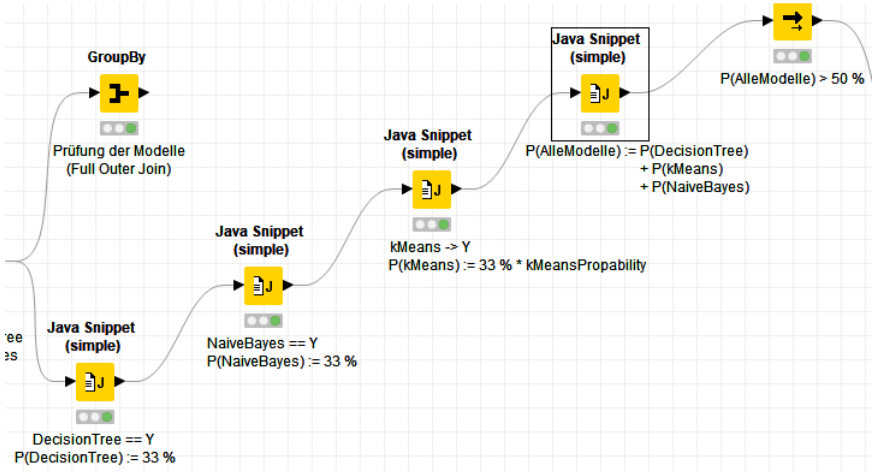


Abbildung 3.36: Aufbau des Ensembles zur Vorhersage (Teil 2/2)

Abbildung 3.36 zeigt den weiteren Schritt, nach dem jede Vorhersage per „Java Snippet (simple)“-Node alle „Y“-Werte in metrische „0, 33“ übersetzt werden - außer beim k-Means. Hier haben wir durch die Berechnung des Mittelwertes einen Wahrscheinlichkeitswert, der mitberechnet werden kann, so dass auch „krumme“ Ergebnisse (statt nur 33,3, 66,6 oder 100) entstehen können.

Filtered - 0:128:164 - Rule-based Row Filter (P(AlleModelle) > 50 %)

File Hilite Navigation View

Table "default" - Rows: 938 | Spec - Columns: 19 | Properties | Flow Variables

Row ID	DecisiontreePrediction	ClusterPrediction	BayesPrediction	Alle Modelle über 50 %	Firstname	LastName	EmailAddress	Company
6.0	33.333	19.048	0	52.381	Angel	Bell	abell@thephone-company.com	thephone-company.com
7.0	33.333	26.087	33.333	92.754	Anna	Bennett	abennett@alpineskihouse.com	alpineskihouse.com
8.0	33.333	33.333	0	66.667	Alyssa	Bennett	abennett@cpandl.com	cpandl.com
9.0	33.333	28.571	33.333	95.238	Arturo	Bhat	abhat@adatum.com	adatum.com
11.0	33.333	18.519	0	51.852	Abigail	Bryant	abryant@northwindtraders.c...	northwindtraders.com
15.0	33.333	33.333	33.333	100	Adam	Carter	acarter@margiestravel.com	margiestravel.com
18.0	33.333	17.391	33.333	84.058	Alicia	Chande	achande@adatum.com	adatum.com
19.0	33.333	30	33.333	96.667	April	Chande	achande@cohovineyard.com	cohovineyard.com

Abbildung 3.37: Ergebnis des Ensembles

Als Ergebnis kann in Abbildung 3.37 erkannt werden, dass in der Spalte „Alle Modelle über 50 %“ alle Werte berechnet zusammengefasst sind.

	A	B	C	D	E	F
1	row ID	Firstname	LastName	EmailAddress	Company	Alle Modelle über 50 %
2	6.0	Angel	Bell	abell@thephone-company.com	thephone-company.com	52,38095238
3	7.0	Anna	Bennett	abennett@alpineskihouse.com	alpineskihouse.com	92,75362319
4	8.0	Alyssa	Bennett	abennett@cpandl.com	cpandl.com	66,66666667
5	9.0	Arturo	Bhat	abhat@adatum.com	adatum.com	95,23809524
6	11.0	Abigail	Bryant	abryant@northwindtraders.com	northwindtraders.com	51,85185185
7	15.0	Adam	Carter	acarter@margiestravel.com	margiestravel.com	100
8	18.0	Alicia	Chande	achande@adatum.com	adatum.com	84,05797101
9	19.0	April	Chande	achande@cohovineyard.com	cohovineyard.com	96,66666667

Abbildung 3.38: Ergebnis für Marketing-Abteilung

In Abbildung 3.38 ist das Resultat in Excel aufgeführt. Denkbar wäre auch, dass mit der Marketing-Abteilung Rücksprache gehalten wird, wie weiter vorgegangen werden soll. Zudem könnte die Idee geklärt werden, ob mit der Firmen-Information aus den Mail-Adressen etwas umgesetzt werden könnte.

Kapitel 4

Schlussbetrachtung

Kritisch kann bei der abschließenden Berechnung der Wahrscheinlichkeit erwähnt werden, dass diese Art der Zusammenfassung keinerlei Gewichtung berücksichtigt. Zumindest beim k-Means kann einfach noch eine weitere Information mitgeliefert werden: Die Anzahl an Datensätzen, die durch ein Cluster repräsentiert wird. Bei den anderen Modellen ist allerdings die Information, welche Wahrscheinlichkeit und welche Gewichtung dahinter steht, nicht ersichtlich. Die Frage wäre hier jedoch: Ist das überhaupt notwendig?

Die Modelle sind für sich gesehen gekapselte Algorithmen die durch Erkenntnisse erweitert und dennoch sehr einfach anwendbar in Knime implementiert wurden. Für Nicht-Entwickler ist es fast wie ein erweitertes Excel, mit dem man, anstatt einzelne Zellen und Funktionen dahinter, frei verschiebbare Nodes und damit verbundene Methoden (die nicht unbedingt eine „Rückmeldung“ mittels Datenausgang geben müssen) einfach per Drag&Drop benutzen kann. Man selbst braucht nur die Theorie dahinter verstehen um zu wissen, was man da genau macht, die Implementierung der Algorithmen wurde jedoch schon einmal umgesetzt. Aus Entwickler-Sicht ist dies bei Knime auch sehr attraktiv, da sehr schnell Daten analysiert werden können, ohne eine Zeile programmieren zu müssen.

Was die Zukunft angeht - angenommen es handle sich hier nicht um ein fiktives Projekt - da wäre auf Anhieb denkbar, dass Daten (auch vordefinierte Marketing-Texte) mittels Schnittstelle direkt aus einer Marketing-Quelle wöchentlich oder täglich in Knime importiert, klassifiziert und anschließend anhand dem Ergebnis sofort - auch aus Knime heraus - an alle möglicherweise zukünftigen Kunden mittels Mail gesendet werden. Durch den „Table Row To Variable Loop Start“-Node, den „Send Email“-Node und den „Variable Loop End“-Node kann so Zeile für Zeile eine automatisch generierte Mail verfasst werden.

Wenn man hierbei den Aspekt bedenkt, dass früher Briefe, oder auch Mails, die persönlich definiert sein sollten, einen hohen Zeit- oder Mitarbeiteraufwand bedeuteten - mal von der Serienbrief-Option abgesehen - ist über Data Mining ein weiterer Vorteil ersichtlich: Es werden nicht mehr Mails als vom DM-Modell vorgeschlagen erstellt und die

Anzahl an potentiellen Mails wird auf diejenigen Beschränkt, denen eine gewisse Wahrscheinlichkeit zugewiesen werden kann, dass die gesendeten Marketing-Informationen auch vom potentiellen Kunden als interessant gesehen werden. Das wiederum hat zur Folge, dass die Kunden mehr Nachrichten erhalten, die von Interesse sein können, als solche, die dann eine negative Stimmung bei der Person erzeugen. Diese Stimmung kann auf das Unternehmen durch eine negative Bewertung zurückfallen.

Fazit: Diese Studienarbeit hat mir einen enormen Einblick in die Welt des Data Minings verschafft, meinen Horizont in der Datenanalyse erweitert und mich darauf vorbereitet, die Möglichkeiten auch für das Unternehmen in dem ich tätig bin mitzunehmen. In einem Stahlwerk stehen Marketing-Maßnahmen nicht im Mittelpunkt, die Methoden lassen sich allerdings gut für das Vorhersagen von Maschinenausfällen im 24/7-Betrieb einsetzen. Dies kann einen sehr hohen Mehrwert geben, wenn es darum geht, die Wartung so spät wie nötig, aber aus Sicht des Ausfalls, so früh wie möglich durchzuführen und damit durch vorausschauende Planung u.U. mehrere 100.000 € einzusparen - allein durch den Ausfall einer Maschine, der nicht (ungeplant) stattfindet!

Anhang A

Verzeichnisse
Abkürzungsverzeichnis

AWC Adventure Works Cycles

Bagging Von engl. Bootstrap aggregating

CRISP-DM Cross Industry Standard Process for Data Mining

DM Data Mining

ETL Extract, Transform, Load

KDD Knowledge Discovery in Databases

KNIME Konstanz Information Miner

ROC-Curve Receiver Operating Characteristics-Curve, Grenzwertoptimierungskurve

ROC AUC Receiver Operator Characteristic Area Under Curve

URL Uniform Resource Locator - Hier genauer: Internet-Adresse

Abbildungsverzeichnis

Literaturverzeichnis

[CL16] Cleve, Jürgen und Uwe Lämmel: *Data Mining*. DeGruyter Oldenbourg, München, 2. Auflage, 2016.

[FPSS96] Fayyad, Usama, Gregory Piatetsky-Shapiro und Padhraic Smyth: *Knowledge Discovery and Data Mining: Towards a Unifying Framework*. KDD-96 Proceedings. AAAI, Seiten 82–88, 1996.

[HJS19] Hammann, Marcus, Janina Jördens und Horst Schecker: *Cohens Kappa*. https://www.springer.com/cda/content/document/cda_downloaddocument/Cohens+Kappa.pdf%3FSGWID=0-0-45-1426183-p175274210, 05.07.2019.

[Kni19] Knime, AG: *Decision Tree | KNIME*. https://www.knime.com/knime-introductory-course/chapter6/section3/decision-tree, 05.07.2019.

[Ras17] Raschka, Sebastian: *Machine Learning mit Python: Das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning*. mitp Professional. MITP, Frechen, 1. Auflage, 2017.

Anhang B

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Offenburg, 07.07.2019
Ort, Datum

Marco Hetzel