Okay, genau. Ich hatte euch gequält mit Datenaufbereitung. Unser Case war Teleflow und die hatten Daten, die mir mein PowerPoint nicht zeigen will. Doch.

Wir hatten als Ausgangspunkt Verträge von Telekom-Kunden und wir hatten dazu noch Informationen über Tickets, die diese Kunden erstellt haben und über sozusagen den Leistungsumfang, den sie in Anspruch nehmen, indem sie zusätzliche Services dazu gebucht haben. Und dann haben wir einen riesen Workflow gebaut hier in Tableau Prep und hatten ja erst mal das Ziel,

uns nochmal klarzumachen, was ist eigentlich die Formalisierung der Aufgabe? Erinnert euch, Formalisierung heißt, wir müssen wissen, was sind die Instanzen? Also sprich, im Endeffekt die Zeilen von unserer Ausgabedatei. Was wollen wir vorher sagen? Was ist unser Klassenattribut? Und was gibt es noch für hoffentlich hilfreiche Attribute, die wir entweder schon haben in den Daten oder, das wäre natürlich der langweilige Fall, also der spannendere, die wir noch konstruieren können und wollen. Und

Es ging los mit den Instanzen. Da habe ich mit euch argumentiert, dass es vielleicht gut wäre, weil ja die Kunden jedes Jahr kündigen können, dass wir jedes Vertragsjahr als Instanz nehmen und dann jeweils gucken, wurde gekündigt oder nicht am Ende des Jahres. Initial hatten wir einfach nur den Vertrag über die gesamte bisherige Dauer. Das heißt, die ersten Jahre waren eben meistens nicht gekündigt worden und daraus auch zu lernen,

Kann interessant sein. Das heißt, wir hatten ja erstmal so ein paar Schritte, wo wir das Enddatum des Vertrags noch konstruiert haben, weil in der Ursprungsvariante, wenn man hier guckt, da gab es entweder ein Datum, zu dem die Kündigung empfangen wurde oder eben nicht. Was bedeutet, dass die noch liefen? Das haben wir also in ein Enddatum umgewandelt. Man sieht immer hier in Tableau Prep auch, was man gemacht hat. Also unter

Änderungen und man sieht auch hier oben Symbole, die einem zum Beispiel zeigen, dass hier, hier sieht man ein Filtersymbol, da haben wir irgendwas gefiltert, nur so zur Erinnerung. Und dann haben wir die Zeilen hier aufgesplittet. Zwischen dem Start- und dem Enddatum haben wir jahresweise Zeilen eingefügt. Kommen die Erinnerungen zurück so im Großen und Ganzen?

Kann ich schneller machen? Genau, dann haben wir hier, ihr müsst ja nicht wieder alles. Nur, dass wir uns nochmal vor Augen führen, was wir da so gelernt haben. Also wir haben zum Beispiel gelernt, dass man Zeilen aufsplitten kann. Wir haben dann an diversen Stellen die Funktionalität berechnetes Feld erstellen genutzt und haben neue Spalten damit kreiert, also berechnetes Feld erstellen.

bedeutet immer, dass wir quasi eine neue Spalte erstellen. Da kann man ihren Namen geben und dann kann man die Formel definieren. Ich habe gesagt, stellt euch vor, wie wenn ihr in Excel über eine leere Spalte eine Überschrift macht, in die erste Zeile darunter eine Formel und die dann so runterziehen. Das haben wir auch genutzt, um das Klassenattribut zu konstruieren und auch später noch ein, zweimal. Wir haben dann

Auch gelernt, wie man noch Daten aus anderen Dateien hinzunehmen kann. Über Joins haben wir noch diskutiert. Was ist ein Left Join? Warum brauchen wir das? Und was auch in beiden Fällen der Fall war, also wenn ihr euch anschaut, was wir hier haben, wir haben in beiden Fällen pro Vertrag mehrere Zeilen. Bei Services, also ein Kunde kann mehrere Dienste dazu gebucht haben, sowohl als auch bei Tickets. Natürlich kann man auch mehrere Tickets anbieten.

pro Vertrag haben. Und wie sind wir damit umgegangen? Also man kann natürlich das einerseits aggregieren, wir haben dann einfach gezählt, wie viele Tickets jemand in einem Jahr hatte. Da muss man ein bisschen gucken, dass man das mit den Jahren richtig sortiert, schon hier bei dem Join als Bedingungen eingeben und dann einfach zählen. Also das entspricht dieses Summenzeichen so einer Art in SQL Group By mit Summenfunktion kombiniert.

Man kann aber auch manchmal bei den Services zum Beispiel die Information, also wenn ich aggregiere, verliere ich Informationen über die einzelnen Tickets, zum Beispiel wann die waren, genau innerhalb des Jahres. Wenn ich alles konservieren will, alle Informationen, habe ich trotzdem das Problem, wenn ich mehrere Tickets habe und ein Vertragsjahr als Instanz, sorry, mehrere Services, dass ich, ja,

wenn ich den Join mache, mehrere Zeilen bekommen und die müssen wieder weg. Und da haben wir pivotiert. Erinnert ihr euch an pivotieren? Also wir haben quasi für jeden Service eine Spalte erzeugt. Also hier unten sieht man Streaming Movies, Tag Support, Multiple Lines. Wir haben viele neue Spalten. Ich glaube, das Attribut hatte sieben Werte. Das heißt, wir haben sieben neue Spalten bekommen und dann eins oder null Spalten.

als Werte verwendet, um anzuzeigen, ob jemand dieses Service gebucht hat oder nicht. Ja, das war es ja eigentlich schon. Wollt ihr über das Quiz sprechen? Ja, Marc. Ja,

Also lassen wir uns einfach die ganze Frage durchsprechen, wenn es okay ist. Es ist ja ein bisschen ähnlich zu der Diskussion, die wir letzte Woche noch am Ende hatten, wo wir gesehen haben, dass es manchmal blöd ist, wenn man Zeilen löscht.

insbesondere wenn man beim Klassenattribut eine starke Unbalanciertheit hat und dann gerade die interessanten Zeilen, wo zum Beispiel Kunden auf das Angebot reagiert haben, gerade die verliert. Hier sieht man, das ist ein bisschen ähnlich. Hier geht es jetzt also um Versicherungsfälle, also Schadensmeldungen. Und die Frage, die vorausgesagt werden soll, ist, ist es nötig, einen Experten oder eine Expertin zu involvieren bei dem Fall?

Und ihr seht, also die meiste Zeit No, sind glaube ich wieder 10 Zeilen und es gibt 3 Zeilen, wo es ein Yes hat. Das heißt, es sind wieder die wertvollen Zeilen, die wir nicht verlieren wollen, die seltenen. Und damit geht es eigentlich auch los, die zwei ersten Optionen, wo man nochmal überlegt, wenn man jetzt diese Zeilen löscht, also entfernt von Instanzen, bedeutet Zeilen löschen, wo Werte fehlen, also da, wo irgendwo diese roten Felder sind,

dann passiert ja wieder, ähnlich wie bei dem Beispiel letzte Woche, das Problem, dass wir jetzt zwei von diesen Schadensfällen löschen, die einen Experten involviert haben. Das heißt, wenn wir das tun und dann mit den restlichen sieben Zeilen oder wie viel Sinn das Modell trainieren, dann wird es öfter Nein sagen, weil es einfach mehr Fälle gesehen hat, wo es keinen Experten hatte. So weit klar? Also, Zweite kann ja dann nicht sein.

Injury Code, das ist das hier. Wie geht man da dran? Also, wenn man jetzt beurteilen will, ob der Injury Code hilfreich ist, gerade bei solchen Problemen, wo es nicht balanciert ist, würde ich immer auf die Yes-Fälle zuerst schauen. Also die drei Zeilen, wo Yes hinten steht und was ist deren Injury Code? Und das fällt es auf, oder? Das beginnt immer mit zwei. Also es ist irgendwie ein Injury Code, wenn ich den als Dezimalzahl interpretiere, ist immer zwischen zwei und drei.

Also hier 2,34 oder 2,34, 2,45, 2,42. Und die anderen fangen nicht mit 2 an, also da wo es ein No hat in der letzten Zeile. Das heißt, das ist eigentlich ein sehr wertvolles Attribut. Das ist auf keinen Fall

unproblematisch, das zu entfernen. So, jetzt sieht man auch Patient Age, da haben wir leider beim Alter zwei fehlende Werte. Und die zwei Optionen hier...

drehen sich darum, ob man das irgendwie aus einer anderen Spalte ableiten kann. Also Imputation hatten wir das genannt. Sorry, das sind die zwei. Vielleicht machen wir erst noch das. Können wir aus Patient Age das Klassenattribut vorhersagen? Auch da würde ich wieder auf die Yes-Fälle gucken. Also bei den Yes-Fällen fehlt eben zweimal das Alter. Das heißt, wir haben nur einen Fall, wo ein Experte involviert war, wo wir das Alter kennen. Also das Alter hier

wo das Alter nicht fehlt, 38, ja, daraus können wir ja nichts ableiten. Also wir können jetzt nicht daraus schließen, dass die Yes-Fälle immer mittelalte Patienten waren oder so. Also tatsächlich können wir das nicht sagen. Und jetzt noch die zweite Frage, hier kann man es irgendwie aus anderen Spalten ableiten. Einmal die Idee vielleicht aus dem Geschlecht, also ich meine,

Bevor wir jetzt überhaupt checken, das macht überhaupt keinen Sinn, oder? Dass man das Alter aus dem Geschlecht vorher sagt, das macht keinen Sinn. Andererseits kann man doch wahrscheinlich annehmen, dass jemand, der älter ist, schon mehr Behandlungen hatte. Und deswegen, also wenn ihr das in den Daten hier ein bisschen überprüft, dann seht ihr auch die Patienten, die älter sind, zum Beispiel hier 62 oder 61, die haben auch elf respektive neun Behandlungen und die jüngeren tendenziell weniger. Also das

gibt eine gewisse Korrelation und das könnte uns helfen, da was zu schätzen. Der letzte Punkt macht keinen Sinn, weil es keine Instanzen gibt, also keine Zeilen, wo mehr als ein Wert gibt. Jetzt die Frage, ob ich deine Frage beantwortet habe, Marc. Zwischenritt. Die ist ein bisschen tricky, die Frage, deswegen habe ich jetzt ein bisschen Zeit genommen. Möchtet ihr noch andere Fragen diskutieren? Also bei dem Quiz letzte Woche, da wart ihr auch nicht so...

Ich habe nicht so viel gewollt und am Freitag wollten Sie dann alles erklärt haben. Ich könnte von mir aus mal überlegen, was ich noch wichtig finde. Aber ihr müsst mir schon sagen. Ich würde jetzt zum Beispiel zwei und drei überspringen. Das hier ist vielleicht noch interessant. Ein Spital hat Daten gesammelt. Was ist das Resultat einer Diskretisierung? Also da muss man sich erstmal überlegen, was war das nochmal? Diskretisierung.

Diskretisierung bedeutet, ich habe eine ursprünglich numerische Variable und dann bilde ich Intervalle. Und man sieht ja Intervalle hier bei der Spalte Alter. So ist das in dem Fall. Und, ah nein, eigentlich wollte ich 5. 5 ist noch spannend. Da sehen wir, all diese 5 Attribute scheinen numerisch zu sein, denn in der Spalte stehen jeweils Zahlen. Und jetzt kann man sich aber überlegen, sind das wirklich numerische Attribute? Ich hatte euch ja auch letztes Mal gezeigt,

Vielleicht tue ich es nochmal in Orange zum Beispiel. Wenn ihr hier Daten ladet und sagt, ja, okay, Customer ID ist eigentlich gar nicht numerisch, ist kategorisch, könnt ihr das hier ändern. Und dann wird es einfach nicht als Zahl behandelt. Das ist die Frage, für welche dieser...

Attribute macht es Sinn. Man muss gar nicht hochscrollen zu den Daten. Man kann überlegen, was diese Attribute bedeuten. Also Labortests. Das hilft es doch, wenn man hochscrollt. Dann sieht man, dass damit wahrscheinlich die Anzahl an Labortests gemeint ist, die jemand bekommt. Ist wirklich eine Zahl, oder? Der Code für die Diagnose. Also tatsächlich auch numerisch hier. Also sowas wie 4242.

Sieht aus wie eine Zahl, aber es ist keine Zahl. Wenn ihr darüber nachdenkt, wir sollten nicht damit rechnen, wir sollten Diagnose-Codes nicht addieren, multiplizieren oder was typischerweise passiert,

wenn wir zum Beispiel einen Entscheidungsbaum bauen, dann sagt der vielleicht, wenn die Diagnose kleiner als 4000 ist, dann das und das. Und darauf wollen wir uns nicht verlassen, dass das irgendwas bedeutet, dass die Diagnose kleiner als 4000 ist. Das ist möglich, aber ich würde mich nicht darauf verlassen wollen. Ich würde es lieber als kategorisch behandeln.

Da habe ich hier geklickt. Ja, da oben habe ich geklickt. Genau, hier wollte ich klicken. Sollte also nicht numerisch behandelt werden. Notfalleinweisung. Also wir können erstmal weitermachen. Tage im Spital ist echt eine Zahl, oder? Also wie viele Tage war jemand im Spital und wie viele Behandlungen hat er oder sie bekommen, ist auch eine Zahl. Notfalleinweisung ist hier mit 1 oder 0 kodiert. Wo ist es hier? Man hätte aber auch Ja oder Nein schreiben können, wenn man sich so überlegt. Also es ist eigentlich ein binäres Attribut.

würde ich auch, also ist es nicht schlimm, in Orange könnt ihr es auch einfach als numerisch lassen, da passiert nichts Schlimmes, dann sagt er einfach größer 0, wenn er Ja meint. Da passiert nichts, aber nur so für sich im Kopf klar zu haben, das ist eigentlich nichts numerisches, auch wenn es so aussieht. Nummer 6 war klar für euch, soll ich was dazu sagen? Ich sehe auch, wer es macht, also ich weiß schon, manche haben es nicht gemacht, oder einige,

Es hilft, wenn ihr es vorher macht, auch weil die Fragen, die ihr dazu haben werdet, die kommen natürlich erst, wenn ihr es probiert. Und auch jetzt, wenn ich es euch erkläre, dann versteht ihr es vielleicht, aber ihr versteht noch mehr und besser, wenn ihr es selber probiert. Soll ich es noch erklären oder nicht? Sagt es doch ruhig. Es passiert nichts Schlimmes jetzt. Es ist für euch. Also, was ist da passiert?

Was man sehen kann, also das sind Patienten, jede Zeile ein Patient und man sieht also, welche Medikamente die, also wahrscheinlich ist es eine Kombination aus Patient und Zeitraum, welche Medikamente jemand in einen bestimmten Zeitraum genommen hat. Und was mal auffällt, wenn man genau guckt, es gibt mehr als eine Eins pro Zeile. Also manche Patienten, zum Beispiel in der ersten Zeile, hat jemand, ist ja logisch, es gibt Patienten, die nehmen ganz viele Medikamente jeden Tag.

Und wenn der Zeitraum größer ist, dann sowieso. Auf jeden Fall spricht das eigentlich gegen die erste Option hier. Also ist One-Hot kodiert, würde ich bedeuten, es gibt pro Zeile nur eine Eins. Ja, wenn man jetzt drüber nachdenkt, das ist so ähnlich wie der Leistungsumfang beim Telekom. Da haben wir pivotiert, weil wir genau das Problem hatten, dass ein Kunde oder zu einem Vertrag mehrere Services dazugehören konnten. Und hier kann eben zu einem Patienten mehrere Medikamente

eingenommen worden sein das heißt wir haben hier eigentlich sowas pivotiert ist und dann ist noch relativ viel nonsens hier unten also wenn du jetzt zum beispiel

Bleiben wir bei dem Beispiel, dem medizinischen. Sagen wir mal, du hast eine Primärdiagnose. Da kann also nur ein Wert zugeordnet werden zu jedem Patienten. Und manche Algorithmen mögen aber keine kategorialen Attribute. Also wenn Primärdiagnose, was weiß ich, schwarzer Hautkrebs ist, dann...

ist das kein Wert, den zum Beispiel neuronale Netze fressen. Du musst es dann one-hot kodieren. Das heißt, du nimmst alle möglichen Primärdiagnosen als Spalten, aber pro Patient, weil es eben nur die eine Primärdiagnose gibt, gibt es nur eine 1 und alle Rest 0. Und hier ist es anders, weil es eben N zu M ist. Also ein Patient kann mehrere Medikamente nehmen, dann wird das pivotiert. Noch weitere Zweifel oder Fragen?

Gut, lassen wir es hinter uns. Heute etwas Neues. Neue Chance. Ich weiß noch. Ja, unbedingt. Also bei der Übung, die wir letzte Woche gemacht haben. Die Instanzen, die haben wir im Jahr umgebracht. Also die Verträge. Ja, die Verträge. Also sozusagen, man muss es nicht unbedingt machen. Aber ich finde, es macht Sinn, weil

Sie haben ja Einjahresverträge in der Firma und immer zum Ende des Jahres hättest du die Möglichkeit, aus dem Vertrag rauszugehen. Und wenn du es nicht tust in einem Jahr, also wenn du einfach mehrere Jahre dabei geblieben bist, es war jetzt nicht unbedingt eine total bewusste Entscheidung wahrscheinlich, aber trotzdem hast du entschieden, den Vertrag weiterzuführen. Jedes Jahr eigentlich. Der wurde zwar stillschweigend verlängert, aber du hast nichts getan, um das zu verhindern und hättest die Chance gehabt.

Und das bedeutet ja, dass in dem Jahr eigentlich im Prinzip alles gut war für dich oder du zu faul warst, diesen Wechsel zu vollziehen, was auch immer. Jedenfalls ist das auch was, woraus wir was lernen können. Es ist sozusagen auch schwierig, wenn du es nicht tust, dann vergleichst du oder in deinen Trainingsdaten hast du dann lauter Verträge. Manche davon haben gerade erst angefangen und manche dauern schon viele Jahre. Und diese Tatsache, dass jemand da schon viele Jahre trainiert,

immer wieder die Entscheidung getroffen hat, nicht zu kündigen, die wird dann nicht mehr so richtig. Ja, also wie gesagt, du musst es nicht unbedingt so machen. Ich würde es so machen, ja. Das ist irgendwie fairer. Jetzt will der hier meine Folien nicht anzeigen. Mal schauen. Naja, das wird schon werden.

Im schlimmsten Fall muss ich meinen Rechner mal starten. Na ja. Na ja. Oh ja. Sorry. Mal checken, ob da Sachen sind, die nicht da sein sollen. Ja.

Ja, sorry, hatte ich vergessen. Danke für den Hinweis. Also nochmal. Genau, ihr seht, wie schnell wir uns entlang von diesem Zyklus hier bewegen. Wir sind jetzt schon bei Modeling. Ich hatte ja gesagt, unser Ziel ist nicht, dass ihr am Ende die ganze Mathematik hinter den Algorithmen versteht. Wir werden trotzdem ein, zwei Algorithmen ein bisschen näher anschauen, damit wir es ein bisschen entzaubern. Also die Magie, die man dahinter vermutet.

Und es hilft auch zu verstehen, ich hatte ja gesagt, was wir können wollen, ist einerseits die richtigen Algorithmen für bestimmte Kontexte auswählen können und auch bestimmte Parameter. Wir müssen nicht alle Parameter von allen Algorithmen am Ende kennen, aber so ein paar Mechanismen, die sich bei vielen Algorithmen wiederholen, kennen. Und dazu hilft es auch, wenn man ein bisschen Einblick hat, wie es funktioniert. Also hier gibt es so ein paar berühmte. Wir werden uns...

Die logistische Regression müssen wir anschauen, die Entscheidungsbäume, Random Forest, also beziehungsweise, ja, Random Forest ist so eine Art Meta-Verfahren. Wir werden stattdessen XGBoost anschauen, beziehungsweise Gradient Boosting. Nicht im Detail, nur die Idee und Canerous Neighbor. Wir fangen aber an mit was ganz Einfachem. Also das ist die einfachste Idee, auf die man kommen kann, wenn man einen Datensatz hat, wie zum Beispiel diesen Wintercheck-Datensatz von

Swiss Bikes, erinnert ihr euch? Der hatte zehn Zeilen und vier davon war Response gleich Yes und sechs davon Response gleich No. Die aller dümmste Baseline, die ich mir denken kann, ist ein Algorithmus, der einfach zählt, wie oft Yes, wie oft No und dann das vorhersagt, was häufiger auftritt und zwar immer. Also für jeden neuen Kunden

das sechsmal No war, dass die Mehrheit sozusagen würde jetzt dieser Constant-Algorithmus, den es in Orange gibt, der anderswo ein bisschen anders vielleicht heißt, der sagt dann für diese Daten, die wir da hatten, immer No. Das ist immerhin in 60% der Fälle auf diesen Trainingsdaten korrekt und wahrscheinlich auch auf weiteren Daten, die wir als Testdaten nutzen später. Diese Baseline ist manchmal gar nicht so leicht zu schlagen, aber sie ist in den meisten Fällen

aus wirtschaftlichen Gesichtspunkten völlig nutzlos. Also wenn ihr euch überlegt, wir wollen Leute finden, die unseren Wintercheck in Anspruch nehmen und wir haben ein Modell, was immer sagt, versuch es gar nicht erst, dann haben wir nichts gewonnen. Also machen wir es ein bisschen komplexer. Ich nenne es auch Baseline. Es ist auch relativ dumm, was da passiert, aber schon ein bisschen intelligenter als vorher.

Wir können jetzt diesen Pseudocode hier durchgehen. Wir können es auch lassen vielleicht. Ja, so ganz grob vielleicht. Also es geht darum, wirklich am Ende sich für ein Attribut zu entscheiden und aus den Werten dieses Attributs Regeln zu bauen. Also wenn das Attribut kategorial ist, dann habe ich eine begrenzte Anzahl von Werten des Attributs, die ich alle durchgehe.

Wenn es numerisch ist, dann muss ich es erst diskretisieren. Das ist aber, ja, also den Algorithmus One Rule, den gibt es in Orange nicht, aber wir können ihn simulieren. Wir werden ihn mittels eines Entscheidungsbaums simulieren und der entscheidet auch selber, wie diese numerischen Attribute diskretisiert werden. Wir gehen also mal davon aus, so der Einfachheit halber, dass das Diskretisieren schon passiert ist und wir haben also ein

kategoriales Attribut. Oder alle Attribute sind kategorial. Was machen wir? Also wir nehmen ein Attribut, also letztlich tun wir das für alle Attribute und am Ende gucken wir, was am besten funktioniert. Und für das Attribut gucken wir uns an, was hat es für Werte. Also lass mal gucken hier, hatte ich doch ein Beispiel. Zum Beispiel bei Kreditkartenbetrug habe ich vielleicht ein Attribut Kartentyp. Und das hat drei Werte. Standardkarte, Goldkarte, Premiumkarte. Und letztlich

Das Modell wird sowas sein hier. Also so viele Werte, wie das Attribut hat, so viele Regeln umfasst mein Modell. In dem Fall hat es drei Werte, also drei Regeln. Und diese Regeln fangen immer an mit, wenn Kartentyp ist gleich und dann der Wert des Attributs. Und dann kommt die Vorhersage, ob es Betrug ist oder nicht. Und das ist auch wieder nur dummes Auszählen.

Also ich mache im Prinzip das Gleiche wie bei Constant. Ich zähle einfach, was kommt häufiger vor, ja oder nein, in der Betrugsspalte. Nur eben mit der Einschränkung, dass ich nur die Zeilen berücksichtige, bei denen ein Kartentyp Standard ist, zum Beispiel. So, jetzt gucken wir mal, ob ihr gut zugehört habt. Also am Ende gucke ich mir dann diese Regeln an und auf meinen Trainingsdaten kann ich dann zählen, wie oft...

Diese Regeln Fehler machen. Also ich zähle ja aus, was ist am häufigsten der Fall. Das heißt ja aber nicht, dass es immer der Fall ist. Das heißt, es macht Fehler. Und ich kann für jedes Attribut dann den Gesamtfehler berechnen. Also einfach die Anzahl Fälle, wo es daneben geht. Und dann wähle ich das Attribut, wo es am wenigsten solche Fehler gibt. Jetzt machen wir das einfach mal. Einmal von Hand. Also ihr habt euch überrascht. Also was müsst ihr machen? Wir können es auch aufteilen.

Eine Hälfte macht das Attribut Monate seit letzter Reparatur. Also ihr seht, ich habe das schon für euch schön so diskretisiert, dass beide Attribute nur zwei Werte haben. Das heißt, für jedes Attribut wird es nur zwei Regeln geben. Man kann auch schon mal anfangen, die hinzuschreiben. Dann wird es für euch leichter. Also, wenn ihr in der Gruppe seid, die das erste Attribut hier checkt,

Dennis, willst du was sagen? Nein, ich wollte nicht, dass du schon verletzt hast. Ich weiß nicht, ob das stimmt. Kann sein. Also, wenn ich kürz mal einfach als Monate ab, ihr wisst, was gemeint ist, also Monate seit letzter Reparatur, kleiner gleich 20, dann Response gleich 20.

Dennis meint nein. Bildet euch eure eigene Meinung dazu. Genau, also das ist die Regel, die wir für den einen Wert dieses Attributs kreieren und wir müssen dann auch noch eine Regel für den anderen Wert. Monate größer 20, dann Response gleich. Mal schauen und lasst uns noch eine Spalte machen hier. Wähler.

Also hier könnt ihr gucken, wie oft geht es schief. Jetzt teilen wir einfach mal hier durch. Hier auf der Wandseite macht ihr Monate. Und hier auf der Fensterseite macht ihr Anzahl Reparaturen in den letzten drei Jahren. Ja, so.

Funktioniert genau analog. Es sind auch zwei Regeln. Ist klar, was die Grundlage ist? Am Ende tragen wir es zusammen und dann haben wir ein Modell. ...

Ja, ich habe mich dann mal überrascht. Ich

Oder andere Malen, wenn man malen kann. Oder Scheiße. Oder so.

Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so. Oder so.

Das ist ein ganz wichtiger Punkt, der uns hier in der Politik und in der Gesellschaft sehr interessant ist. Das ist ein ganz wichtiger Punkt, der uns hier in der Politik und in der Gesellschaft sehr interessant ist.

Vielen Dank.

Vielen Dank.

Ja, stimmt. Ja, ja. Ja, ja.

Ja, das heißt, da wurde es weniger, als sie alles können. Ja, das ist gut. Dann gibt es eine innere Rückmeldung. Ja, das ist gut.

...

... ...

Das wäre dann schon entscheidend.

Ja, das ist gut.

... ...

Jetzt fehlen ja hier noch ein paar Einschläge.

Also nehmen wir mal hier die Bandleite. Wollt ihr anfangen mit eurem Monate-Attribut? Wenn Monate kleiner gleich 20, was sagt das Modell dann? Sag mir doch mal deinen Namen. War das eigentlich das, was der Dennis gesagt hat? Das war andersrum, ja.

Und bei größeren 20? Also, ich habe gerade mit Jeremy, oder? Wir haben gerade diskutiert, es kann natürlich auch mal passieren, dass beide Male jetzt rauskommen oder beide Male nur, insbesondere wenn eins sehr viel häufiger ist als das andere, dann ist das auch nicht besser als der Constant-Algorithmus, aber hier habe ich es natürlich so kontrolliert, dass es wirklich diskriminiert

Wie viele Fehler machen wir? Also kleiner gleich 20 oder machen wir erst größer 20? Bei größer 20 haben wir immer No, oder? Also wenn du hier guckst, größer 20 No, größer 20 No. Also Fehler 0, oder? Jetzt haben wir es sogar schon verraten. Bei größer 20. Was für einen Fehler habt ihr bei kleiner gleich 20? Nein.

Also wenn ich hier die Summe bilde, ein Summenzeichen, kennt ihr die Summenzeichen? Sehr gut. Dann habe ich also Gesamtfehler 2 für dieses Modell. Jetzt habe ich alternativ von der Fensterseite ein Modell. Wenn die Anzahl der Faktoren kleiner 2 ist, was sagt euer Modell? Nein. Hast du das auch gemacht? Ihr habt alles gesummt. Also was noch? Und wenn es größer gleich 2 ist, merkt es

Wie sind die Fehler? Fehler? 0 und 1. Also bei kleiner 2 sagt es immer No.

Ja. Oder sagt es No und das ist immer richtig. Und dann gibt es diesen einen Fall bei größer gleich 2, wo der Kunde trotzdem nicht reagiert hat. Ja, genau. Das wäre eine Fehler. Am Ende entscheiden wir uns für das Modell mit dem kleineren Fehler logischerweise. Das heißt, das hier ist unser Klassifikationenmodell.

Und jetzt habe ich zu euch gesagt, wir können das in Orange simulieren. Jetzt schauen wir mal, ob es da rauskommt. Ich habe die Daten hier schon geladen. Ich weiß gerade nicht. Ich glaube, sie sind nicht auf Moodle. Aber lasst uns nochmal checken. Wir hatten die ja auch auf der Folie. Wirklich nur

diese 10 Zeilen. Also im Prinzip diese Daten, die wir jetzt auf dieser Folie hatten, nur natürlich, dass dabei ein paar Attribute gefehlt haben. Die kann ich jetzt einfach gleich beim Select Columns mal rausnehmen.

Und hier ist also die Monate sind noch nicht diskretisiert und die Repairs auch nicht. Aber das macht jetzt, habe ich versprochen, der Algorithmus für uns. Ja, das hat er sich wieder gemerkt, aber falsch. Also Last Spike und Reach nehme ich raus. Also.

Ich brauche zum Simulieren jetzt ein Tree und ich simuliere oder ich erlaube diesem Tree einfach nur die Tiefe 1. Dann wird er automatisch nur ein Attribut zur Unterscheidung nehmen und dann hat er zwei Äste. Der eine ist kleiner 2, der andere größer 2. Also wenn ich Binary Tree mache, dann sowieso. Das heißt, ich gehe hier auf App 1. Ich glaube, es war so. Und dann nehme ich den Tree Viewer. Schauen wir mal, was passiert.

Okay, ich muss doch auf 2 gehen. Ich brauche die Wurzel und dann die jeweils entscheidenden Blattnoten. Jetzt habe ich irgendwie einen Vorführeffekt. Wir machen mal das hier weg. Vielleicht hilft das noch. So.

Genau, also er macht das Gleiche. Sorry, fand ich nicht hilfreich, da noch so große Zahlen drin stehen zu haben. Er entscheidet sich auch für die Anzahl der Reparaturen, weil das den kleineren Fehler hat. Also in dem Entscheidungsbaum, wir werden es gleich in den Folien auch sehen, passiert etwas ganz Ähnliches, wie das, was wir jetzt gerade auf dem Hand gemacht haben. Und zwar immer wieder. Also er fängt sozusagen oben an, den Baum zu bauen.

sucht das beste Attribut aus, im Prinzip wie so ein One-Rule-Klassifikator und geht dann einfach noch weiter runter und macht es nochmal und nochmal, bis es perfekt ist sozusagen, bis der Fehler verschwindet. Und ja, also ihr seht, hier ist größer 1, was das gleiche ist wie größer gleich 2 und kleiner gleich 1, was das gleiche ist wie kleiner 2, also es ist exakt das gleiche Modell. Habt ihr Fragen? Super. Dann können wir weitermachen.

Hier hatte ich das versteckt. Das ist nochmal das Ergebnis, Anzahl Fehler in der letzten Spalte und dann jetzt nicht mehr, das könnte ich noch hervorheben eigentlich, dass die untere Zeile das Modell ist, für das wir uns entscheiden. Genau, jetzt habe ich gesagt, der Entscheidungsbaum, der setzt das sozusagen fort aufs Horis und wir lernen gleich, wie man den baut.

Vielleicht verstehen wir erstmal, das ist jetzt also ein Baum, der diese Trainingsdaten perfekt abbildet. Wir werden später lernen, dass das nicht unbedingt eine gute Idee ist, aber der macht keine Fehler auf diesen Daten. Und also zunächst mal, wie wendet man das an auf die Daten oder auf neue Daten? Also wenn das der gleiche Baum wie gerade eben ist.

Und das ist ein neuer Kunde, den wir noch nicht kennen. Wir wissen nicht, ob dieser Kunde oder diese Kundin reagieren wird oder nicht. Also hinten das Klassenattribut ist ein Fragezeichen. Ja, und der Baum sagt uns sozusagen, schau erst auf das Attribut Repairs. Der Wert dafür ist 2. Das heißt, wir gehen in diesen Zweig des Baumes. Und dann sagt es uns, okay, schau als nächstes auf die Fahrradkategorie.

Also dieser Neukunde hat ein Tracking-Rad. Wenn ich das mit den möglichen Werten hier vergleiche, dann lande ich auf der linken Seite. Also das ist ja kein Single-Speed-Rad. Und dann, ja, muss ich das hier irgendwo anders hinschieben, sagt der Baum also, es wird ja eine Response geben. Und ihr erinnert euch vielleicht, oder wir können es hier noch sehen, bei größer gleich 2 gibt es eben einen Fehler. Und da gab es einen Kunden, der hatte...

Ich glaube, zwei Reparaturen oder sogar mehr. Hat aber nicht geantwortet. Und das war dann wahrscheinlich einer, der hatte ein Single-Speed-Rad. Das fangen wir damit auch noch ab. Jetzt, wie lernen wir sowas? Schauen wir uns noch an, dann machen wir Pause. Wie gesagt, man spult diesen Algorithmus, den wir gerade von Hand gemacht haben, einfach mehrfach ab. Also man fängt an, da oben mit der Gesamtmenge und man guckt sich immer an,

habe ich schon die perfekte Unterscheidung. Am Anfang habe ich sechsmal No und viermal Yes. Das ist noch nicht perfekt. Wenn ich nur zehnmal No hätte, dann hätte ich schon mein Modell fertig. Ich würde einfach immer No sagen. Aber insofern muss ich jetzt weitermachen. Okay, jetzt steht hier eine Menge Text. Wir befinden uns jetzt noch in dem ersten Fall hier. Und zwar, dieses DT bezeichnet immer diese Menge von Instanzen, die

den aktuellen Zweig unseres Baums erreicht haben. Also wir sind noch ganz am Anfang. Ich sage, dt ist unsere Ausgangsmenge, also alle. Und da gibt es noch zwei verschiedene Klassen in allen Instanzen. Und wir wählen jetzt ein Attribut. Der Algorithmus sagt hier nicht, wie wir das wählen. Ich verrate euch eine Möglichkeit, das zu tun, ist so, wie wir es gerade gemacht haben. Das heißt, so einen Fehler zu berechnen. Es gibt noch andere Maße außer Fehler,

die ein bisschen komplexer sind. Aber stellt es euch einfach mal so vor. Und in dem Fall hatten wir jetzt ausgerechnet, dass das Anzahl Reparaturen Attribut den geringsten Fehler verursacht. Das heißt, ach komm jetzt, das gibt es doch nicht. Ich werde wahnsinnig. Also gut, man kann genug sehen, glaube ich. Das hat den geringsten Fehler. Und jetzt gucken wir hin. Also wenn die Anzahl Reparaturen kleiner 2 ist, dann hatten wir vorhin gesehen, oder sehen wir immer noch,

Da macht das keinen Fehler. Und es gibt fünf Hunden, die weniger als zwei Reparaturen hatten. Bei denen war immer No. Also die haben nicht reagiert. Und wenn das wirklich so ein reiner Knoten ist, dann machen wir den zu einem Blattknoten. Das heißt, an der Stelle sagen wir No vorher. Fertig, dann machen wir nicht weiter. Und jetzt sozusagen der gelbe Kasten hier wieder zu unserem DT. Und da...

Also dieser war ja auch GT. Das ist der zweite Fall hier. Alle gehören zur gleichen Klasse. In dem Fall No. Dann fertig. Auf der rechten Seite sind wir immer noch in dem Fall, dass es sowohl No als auch Yes hat. Und wir machen dementsprechend weiter. Das heißt, wir wählen das nächste Attribut. Und in dem Fall haben wir jetzt die Fahrradkategorie gewählt.

Ich glaube, in dem Fall ist es schon fast egal, welches Attribut man wählt, denn man wird es mit jedem Attribut schaffen, null Fehler zu machen. Also hier sind ja auch wieder nur fünf Kunden drin und ich muss nur ein Attribut finden, was die... Also wir können tatsächlich nochmal gucken, welche fünf Kunden das sind. Größer gleich zwei. Also hier haben wir Yes, hier haben wir Yes, hier haben wir Yes. Und da ist das No.

Und da sehen wir eben dieser Kunde, der nicht reagiert hat, obwohl er oder sie mehr als zwei Reparaturen hatten, hat ein Simul-Speed-Rat. Und das ist dann das, was wir hier hernehmen, um die Unterscheidung zu machen. Ich habe vorhin gesagt, es ist nicht unbedingt eine gute Idee, so weit diesen Baum wachsen zu lassen, bis er perfekt auf der Trainingsmenge funktioniert. Auch deswegen, weil wir jetzt zum Beispiel hier wirklich nur aus einem Kunden irgendwie ableiten, was vorher zu sagen ist.

Und ob das dann wirklich immer stimmt, wenn man das jetzt in Kombination nimmt. Und das ist vermutlich relativ robust. Also die Aussage, dass Leute, die mehr Reparaturen hatten, eher dazu neigen, auf den Wintertag machen zu lassen. Ob es dann wirklich die Fahrradkategorie ist, die den

letztlichen Unterschied macht, das kann auch sein, dass es einfach in diesen Daten zufällig war. Genau, ich hatte jetzt gesagt, das Ziel ist es, so ein Maß zu haben, was irgendwie misst, wie gut ein Attribut

unterscheidet. Wir hatten jetzt den Fehler genommen, den Gesamtfehler und man kann auch andere Maße nehmen. Letztlich ist es so, das Maß soll leisten, dass wenn sowas rauskommt, dass es bestraft wird. Wenn sowas rauskommt, wird es belohnt. Genau, jetzt haben wir fast die Pause erreicht. Ja, was wir jetzt vielleicht besser verstehen in diesem Dialog hier, ich hatte ja gerade gesagt, dass mit dem Single-Speed-Fahrrad

Finde ich kritisch, weil da nur ein Kunde im Blatt war. Also das wurde nur aus einem Kunden gelernt, die Vorhersage in dem Blattknoten. Und hier kann ich zum Beispiel einstellen, wie viele Instanzen mindestens in einem Blattknoten landen müssen, damit der überhaupt verwendet werden darf. Und wenn die Zahl unterschritten wird, dann wird dieser Blattknoten gar nicht erst angelegt, dann wird einfach vorher abgeschnitten sozusagen.

Und natürlich, das kennen wir schon, dass wir einfach die Tiefe einschränken. Das sind einfach zwei unterschiedliche Mechanismen, mit denen wir arbeiten können. Man nennt das Pruning. Also Pruning heißt sozusagen, dass man etwas zurückschneidet. Bäume oder Sträucher. Und dadurch eigentlich letztlich die robuster macht. Okay. In Orange, jetzt ist wirklich die letzte Folie, ja. Ähm,

Da haben wir ja gesehen, wie so ein Baum aussieht. Das ist jetzt wieder das Beispiel aus unserer ersten Sitzung, wo wir das Escape Game gemacht haben. Und ihr seht nochmal diese Zahlen. Und da kann man wieder sehen, dass von den Trainingsdaten, aus denen dieser Baum gelernt wurde,

zum Beispiel 229 Instanzen in diesem Blattknoten landen. Also die Vorhersage hier, yes, basiert auf 229 Instanzen und denen 134 yes waren. Das ist sogar auch mal grafisch dargestellt. Also yes ist rot und no ist blau.

Und wenn man anfängt, und das machen wir dann nachher auch, solche Bäume zu interpretieren, dann ist natürlich einerseits wichtig, wie verlässlich ist das? Also je kleiner der Unterschied zwischen diesen zwei Zahlen, desto verlässlicher ist es. Und andererseits wird es aber auch irgendwie darauf ankommen, zu gucken, also oft guckt man bei solchen Sachen eher auf die Yes-Fälle zuerst und dann guckt man auch meistens noch

Wo sind die Zahlen denn groß? Also wenn es jetzt mehrere Yes-Knoten hätte, dann würde ich wahrscheinlich erstmal auf den gucken, wo viele Kunden drin sind, weil das auch mehr Erfolg verspricht dann für meine Kampagne. Okay? Fragen? Zweifel? Unklarheiten? Alles Bestens. Dann würde ich sagen, machen wir bis 25.000. Machen wir weiter. Wir haben noch ein paar Fragen.

Algorithmen vor uns. Also, unser nächster Algorithmus ist hier so grafisch. Das ist manchmal irre für, aber ich mache es trotzdem, weil es wirklich die einfachste Art ist, es zu erklären.

Der K-Nearest Neighbor wird auch Lazy Classifier genannt, weil er gar kein Modell berechnet. Er macht sozusagen alles zu der Zeit, wo die Vorhersage gemacht werden soll. Also man gibt ihm eine Trainingsmenge, speichert er irgendwo ab und dann passiert alles erst, wenn ich einen roten Punkt hier, also eine neue Instanz habe, die klassifiziert werden soll. Also jetzt...

Stellen wir uns mal was darunter vor, was man hier sieht. Nehmen wir mal an, hatten wir das? Doch, hatten wir, die Kreditvergabe. Und das ist jetzt zweidimensional gezeichnet. Das scheinen zwei

numerische Attribute zu sein, durch die hier jeder Kreditantrag beschrieben wird. Nehmen wir mal an, die X-Achse ist das Einkommen und die Y-Achse ist Geld.

Der Schuldenstand, das ist jetzt nicht so realistisch, wenn man es schaut, die Plus- und Minuswerte, aber egal. Also Plus heißt, hat den Kredit zurückgezahlt, Minus heißt, hat den Kredit nicht zurückgezahlt. Wir wollen jetzt für den roten Punkt vorhersagen, was passieren wird. Man hat ja so eine Intuition, wenn man sich das Bild anschaut. Also dieser neue Punkt wird jetzt auch beschrieben durch Einkommen und den Schuldenstand und befindet sich jetzt grafisch

in einer Wolke von Plus-Instanzen. Das heißt, wir würden wahrscheinlich auch raten, dass dieser Kunde oder diese Kundin den Kredit zurückzahlt. Und das ist genau die Intuition auch hinter K-Nearest Neighbor. Also man legt eine Zahl fest für K. In diesem Fall hier war es wohl drei. Und bestimmt dann für die neue Instanz die K, beziehungsweise in dem Fall drei nächsten Nachbarn.

Nächste Nachbarn heißt, ich brauche so etwas wie ein Ähnlichkeitsmaß. Oder hier, was wir intuitiv in unserem Kopf machen, wenn wir so etwas in 2D sehen, wir haben ein Distanzmaß. Also wir können einfach unser Lineal nehmen und gucken, welches sind die drei Punkte, die den kleinsten Abstand zu diesem Punkt haben. Und natürlich können wir aus einem Abstandsmaß mathematisch auch ein Ähnlichkeitsmaß machen, indem wir es irgendwie umdrehen. Was daran irreführend ist, ist, dass...

Das, was für uns intuitiv klar ist, nämlich, dass diese drei Plus- Instanzen die nächsten Nachbarn sind, das muss ein Computer berechnen. Also der muss wirklich den neuen Punkt mit allen anderen vergleichen. Insbesondere wird es nötig, wenn es mehr als zwei Attribute gibt. Das ist nicht von vornherein klar, wo die nächsten sind. Man muss wirklich zu jedem Punkt der Trainingsmenge diese Distanz oder diese Ähnlichkeit berechnen und dann weiß man, welches die drei nächsten Nachbarn sind. Und das ist

werden wir auch nachher sehen, etwas, was dann doch gewisse Zeit in Anspruch nehmen kann, also für Echtzeit-Klassifikationen ist der Gainer's Neighbor manchmal nicht geeignet, wenn es viele Ähnlichste anhat. Jetzt gucken wir mal. Okay, ich hatte jetzt, jetzt habe ich hier Schulden und Sicherheiten, Sicherheiten, Einkommen, egal, ist irgendwie ähnlich. Jedenfalls wollte ich das nicht zeigen. Machen wir das mal anders. Habt ihr es euch gemerkt? Ja.

Aber es ist wahrscheinlich auch auf Moodle, ne? Schaut nicht auf Moodle. Steht es da drin? Ja, wahrscheinlich. Man schaut es nicht an. Wenn es drin steht, auch gut und ihr es angeguckt habt, was passiert hier eigentlich? Okay, dann können wir uns immer noch überlegen, warum. Also, jetzt mal nur nach vorne schauen. Ihr habt jetzt also hier diesen neuen Punkt, neuer Kreditantrag.

Und jetzt ist die Frage, wenn K gleich 1 ist, was passiert dann? Wir nehmen mal erst die erste Zeile, ungewichtet. Was gewichtet bedeutet, können wir gleich überlegen. Also was entscheidet der Cane-Earish-Neighbor? Wird der Kunde das zurückzahlen oder nicht? Das hast du jetzt gut abgelesen. Also warum? Das ist der nächste Nachbar, oder? Also K ist 1, das heißt, wir suchen nur einen nächsten Nachbar.

Gewichtet bedeutet, dass man, wenn man mehrere Nachbarn sucht, eine Abstimmung macht. Man macht immer eine Abstimmung bei gewichtet und ungewichtet. Aber bei gewichtet bedeutet es, dass die Stimme von Instanzen, die näher dran sind oder ähnlicher sind, mehr zählt. Also man gewichtet die Stimme mit der Ähnlichkeit. In dem Fall macht das natürlich keinen Unterschied, weil man nur einen hat. Wenn man jetzt zwei nächste Nachbarn sucht,

Was passiert dann bei ungewichtet? Welches ist der zweite, der zweitnächste Nachbar? Der hier, oder? Das heißt, ich habe viele zwei nächste Nachbarn. Was kommt raus? Sehr gut. Genau, also es ist

Du hast es auch noch nicht verstanden. Genau. Es ist nicht unbedingt eine gute Idee, K als gerade Zahl zu wählen, weil genau sowas passieren kann. Das will man natürlich nicht. Gut, manchmal ist es vielleicht gar nicht so schlecht, wenn man sagt, ich weiß nicht. Weil die Situation, dass man einen dafür, einen dagegen hat, ja wirklich, dieser Punkt ist ja auch so gewählt. Da ist gerade dazwischen eingestreut. Das ist so ein bisschen grenzhaft. Wenn ich eine Gewichtung habe,

Was wird dann vorher gesagt und warum? Also ich gewichte jetzt die zwei nächsten Nachbarn mit ihrer Ähnlichkeit. Jetzt machen wir k gleich 3 und dann ist wahrscheinlich dieser hier, oder? Der drittnächste. Das heißt, wir haben dann 2 mal Plus, 1 mal Minus und dann gewinnt Plus.

Wenn wir es gewichten, was passiert dann? Wenn ihr auf die Folie guckt, steht da ein Fragezeichen. Was bedeutet das? Ja, also wir müssten die genaue Distanz berechnen und das vielleicht auch in eine Ähnlichkeit umwandeln. Und dann ist es möglich, dass die Summe aus den Ähnlichkeiten von den beiden hier größer ist als die Ähnlichkeit von dem. Es ist aber auch möglich, dass die Ähnlichkeit von dem größer als die Summe ist. Also das hängt davon ab.

wie genau die Zahlen sind. Deswegen können wir das jetzt nicht so entscheiden. Und das hängt auch ein bisschen von dem Distanzmaß oder Ähnlichkeitsmaß ab, was wir verwenden. So, jetzt sieht es wieder so aus wie vorher. Muss ich nachher daran denken, dass ich das wieder rausnehme für Freitag. Gut. Also, Canaris Neighbor hat so ein paar Nachteile. Erstens mal, ja, das ist sowas, was man ausprobieren muss. Also das ist eigentlich der einzige Parameter, den man bei Canaris Neighbor in der Hand hat, den man verstellen kann.

Das K, also ihr habt es wahrscheinlich schon gesehen, in Orange gibt es den Algorithmus auch. Hier ist mal so ein Extrembeispiel, also wenn ich K zu klein wähle, das war jetzt nicht unbedingt das Beispiel hier, wir wissen nicht, ob es Minus oder Plus ist, aber es kann auch mal sein, wenn der Punkt jetzt noch ein bisschen weiter hier wäre,

und ich hätte hier noch irgendwo ein Minus oder auch hier ein Minus, kann ja auch mal sein, das wäre dann so eine Art ein Noise Point oder ein Ausreißer, der dann, also nehmen wir an, ich habe hier ein Minus, dann würde ich trotzdem vermuten, wenn ich einen neuen Punkt irgendwo hier habe, wenn ich dann K gleich 1 wähle, dann wird der als Minus klassifiziert, weil er diesem Ausreißer am nächsten ist, aber hier ist ja eigentlich so die Plusregion.

erwarten, dass eigentlich die meisten Punkte, die jetzt hier irgendwo liegen, also als positiv klassifiziert werden. Und hier sieht man eben, wenn man es zu groß wählt, dann überschreitet man manchmal so Entscheidungsgrenzen. Also wenn man den Punkt nimmt, dann würde man ja auch eigentlich erwarten, dass der seine Schulden zurückzahlt sozusagen oder dass er als Plus klassifiziert wird. Und

Wenn ich aber so viele Neighbors nehme, dann wird er am Ende als Minus klassifiziert. Es gibt auch nichts, was ich interpretieren kann. Also ich kann ein bisschen erklären, warum etwas vorhergesagt wird, indem ich zeige, welches die nächsten Nachbarn von dem neuen Punkt sind, aber ich kann kein Modell interpretieren. Ich kann nicht sagen, was das Modell im Allgemeinen tut. Okay, noch ein Algorithmus, die logistische Regression. Eine wichtige Annahme der logistischen Regression ist,

ist, dass die Attribute voneinander unabhängig sind. Wir werden nachher sehen, dass auch zum Beispiel der Fall ist bei dem Naive Bayes Klassifikator, der ist ein bisschen ähnlich. Die logistische Regression, also wenn ihr euch dann irgendwann später mit neuronalen Netzen beschäftigt,

dann werden wir sehen, dass es sozusagen die Keimzelle des neuronalen Netzes, also die Formel, die hier steht, beschreibt, was ein Neuron in einem neuronalen Netz macht. Und zwar wird ein Output berechnet als gewichtete Summe von Attributwerten, also von Inputs. Und also die x i hier sind die Attributwerte und die w i sind die Gewichte sozusagen. Und die Gewichte werden gelernt. Also

Wir machen das gleich an dem Beispiel von Swiss Bikes. Also da sind die XI dann wieder so Sachen wie, wie viele Reparaturen hat jemand gehabt und welches Gewicht das in der Entscheidung haben soll. Also wie sehr das, in dem Fall wird es wahrscheinlich einen positiven Einfluss auf die Response gleich Yes haben, wie sehr das das beeinflussen soll, das wird gelernt aus den Daten. Und ja, dann setzt man das noch in so eine Funktion ein, da passiert letztlich nicht viel.

Wenn das hier größer als 0 ist, dann kommt bei dieser Funktion 1 raus. Und wenn das hier kleiner als 0 ist, dann kommt 0 raus. So im Großen und Ganzen zumindest. Und der Vorteil hiervon, das liegt ja zwischen 0 und 1, das ist erstmal schön und kann auch dann als Wahrscheinlichkeit interpretiert werden. Also als Wahrscheinlichkeit, dass ein Kunde antworten wird. Also

Ist es eine Transformation von dieser gewichteten Summe? Eigentlich bräuchten wir die für die Entscheidung selbst nicht. So, hier ist das Beispiel. Sollen wir das mal ausprobieren? Jetzt muss ich mal gerade gucken, wie ich das gemacht habe. Also, das sind die gleichen Daten. Vielleicht nehmen wir diesmal wieder alle Falten. Zumindest diese hier noch. Und dann die logistische Regression. Ich glaube...

Ich hatte hier auf Lasso gestellt. Was das ist, lernen wir später noch. Und dann kann ich hier hinten einfach mal ein Data Table anhängen und dann sagt er mir, er will mir in diesem Data Table die Koeffizienten zeigen. Also die Koeffizienten der logistischen Regression sind, wenn ich diese schöne Formel hier nehme, sind diese WIs. Und es wird gleich hoffentlich so aussehen wie hier, dass ich hier die Attribute habe.

Und dann hier diese Koeffizienten in der anderen Spalte. Lass mal schauen. Vielleicht kommt das raus, vielleicht auch was anderes. Doch, ich glaube, das ist genau das, oder? Außer, dass die Reihenfolge der Attribute anders ist, ist es dasselbe. Was ihr seht, ist auch, der hat hier mit der Bike-Category was gemacht, das Orange? Der hat da irgendwas umtransformiert. Ja, da fehlt noch das Wort. Ja,

Im Coding war besser, da fehlten noch zwei Wörter vorne dran. Ja, genau. Also man hat ja Last Byte Board, das kann ja nur eins sein. Also ich meine, wir können nochmal... Nee, können wir nicht. Aber was die Zahlen hier bedeuten, ist... Fangen wir mal so an. Wenn die Zahl 0 ist, dann heißt es, dieses...

Das heißt, die einzigen zwei Attribute, die es benutzt, sind, wie viele Reparaturen hat jemand in den letzten drei Jahren gehabt und wie lange ist es her. Anna, du wolltest was fragen? Ich wollte fragen, wie ich das genau weiß, welche Zahl jetzt eine Gewichtung ist, in welcher Form ein X-Rechte-Logum.

Also sozusagen, wenn du eine logistische Regression hier in Orange lernst, dann siehst du ja hier, dass die Koeffizienten in diesem Data Table gespießen werden. Das heißt, alle Zahlen, die du da siehst, sind WIs, also Koeffizienten. Und die werden dann multipliziert mit den XIs für jede Instanz, für jeden Kunden steht da was anderes. Also ich

mach das mal gleich vorher und ich mache die animation dann kommt so nach wie man ja genau wir können jetzt sozusagen hier so ein beispiel kunden

mal daneben halten. Also dieser Beispielkunde hatte vor sieben Monaten seine letzte Reparatur und hatte aber auch nur diese eine in den letzten drei Jahren. Genau, also was ist deine Frage? Also die Anzahl Monate sieben wird jetzt mit diesem Gewicht multipliziert. Und der gute Punkt hier ist,

ihr seht, dass es eine negative Zahl ist, das ist irgendwie logisch. Also je größer diese Summe wird, also wenn die Summe größer 0 ist, habe ich vorhin gesagt, dann sagt das Modell ja, wenn sie kleiner 0 ist, sagt es nein. Das heißt, die Tatsache, dass es negativ ist, bedeutet einfach, je länger es her ist, also je mehr Monate seit der letzten Reparatur, desto kleiner wird die gewichtete Summe. Das macht Sinn. Leute, die

Noch vor nicht so langer Zeit, ihr Fahrrad bei der Reparatur haben, sind eher solche, die vielleicht eben eher affin sind, solche Angebote anzunehmen. Also man kann es natürlich immer so rum und so rum sich erklären, aber ich finde, es macht Sinn.

Und jetzt, was passiert, ist einfach nur, ich muss 7 mal minus 0,15 plus 1,4 rechnen. Und dann kommt da was Positives raus. Dann weiß ich schon, das Modell wird Ja sagen. Kann es dann noch in diese Funktion einsetzen? Dann kann ich es auch als Wahrscheinlichkeit interpretieren. Das heißt, das Modell sagt, mit ungefähr 58% Wahrscheinlichkeit bringt dieser Kunde sein Fahrrad zum Wintercheck. Und somit sagt das Modell Ja. Jetzt kann ich so ein bisschen...

umschrauben und sagen, wenn die Reparatur zehn Monate her ist, also ich habe jetzt einfach hier die Werte ein bisschen manipuliert, dann beim Aufsummieren kommt hier was Negatives raus, entsprechend wird die Wahrscheinlichkeit kleiner ein halb, nur mit 46% Wahrscheinlichkeit bringt der Kunde laut Modell sein Fahrrad zur Reparatur und somit sagt nein. Ist so klar ungefähr? Also wir haben jetzt nicht verstanden, wie dieser Algorithmus funktioniert,

Also wie diese Gewichte konkret gelernt werden. Aber wir haben zumindest verstanden, was gelernt wird und wie wir das interpretieren. Was genau war dein Sprengstil? Ja, okay. Also es ist ein Verfahren, das nennt sich Gradient Descent. Ich glaube, es führt zu weit, wenn wir das jetzt machen. Du kannst das mal googeln. Okay, jetzt habe ich das, glaube ich, alles schon gesagt, oder? Ja.

Was man machen kann ist, man kann diese Attributwerte hier, also die 7 zum Beispiel, also Months since last repair, liegt irgendwo zwischen, weiß ich nicht, 0 und, ja, was weiß ich, wie weit der Datensatz zurückgeht. Also vielleicht ist die letzte Reparatur maximal 36 oder 40 Monate her, dann liegt es zwischen 0 und 40 und ich könnte das jetzt auch normalisieren. Sollen wir das mal ausprobieren?

Das habe ich jetzt nicht vorher getestet. Mal schauen, was passiert. Das ist weg. Das machen wir pre-process. Also wir könnten mal sagen, wir wollen es ins Intervall 0,1 normalisieren. Das klingt doch gut. Dann können wir mal schnell noch ein Data Table machen. Das heißt zum Beispiel, wenn einer... Was war das Maximum in den Daten? Weiß ich nicht mehr. Aber ihr seht, dass jetzt alles zwischen 0 und 1 liegt. Können wir uns nochmal anschauen.

Also maximal war es 36 Monate her. Das ist der größte Wert, oder? Das heißt, in der vierten Zeile müsste jetzt eine 1 stehen. Und zum Beispiel hier in der vorletzten haben wir sicher was, was 0,9 irgendwas ist. Ja, hier ist die 1 und da ist der sehr hohe Wert in der vorletzten Zeile. Jetzt können wir das alles wieder wegmachen. Und wenn wir das jetzt hier dranhängen, so vielleicht, dann kommen andere Koeffizienten raus. Sogar ein ganz anderes Modell.

Das ist jetzt schade. Aber jetzt können wir diese Koeffizienten wirklich auch als Wichtigkeit des Attributs interpretieren. Das ist jetzt wirklich ein bisschen seltsam. Ihr seht, das ist nicht immer alles ganz stabil. Das Problem ist wahrscheinlich die One-Hot-Encoding. Das ist entweder 0 oder 1. Die anderen Sachen sind dann kleiner geworden und verlieren an Einfluss.

Okay, also im Allgemeinen, das ist sozusagen der Punkt, bevor ich euch jetzt verliere, solltet ihr jetzt nicht schließen, weil der Koeffizient bei Repairs last three years größer ist als der andere, dass das Attribut wichtiger ist. Man muss immer bedenken, mit was wird es multipliziert.

Die Zahl hier ist vermutlich im Allgemeinen größer als die Zahl hier. Also hier gibt es vielleicht Leute, die haben 36 Monate lang keine Reparatur gehabt, aber keinen Fall 36 Reparaturen haben. Versteht ihr, was ich meine? Also das muss man dann entweder mal normalisieren oder man hat das im Hinterkopf, was ist der Value Range von den Attributen. Wir können jetzt auch noch ein Nomogramm anhängen.

Dann wird das auch nochmal schön plastisch und da können wir natürlich auch jetzt, dann lassen wir das normalisieren mal wieder weg, da können wir jetzt zum Beispiel auch unseren Testkunden einfach mal reingeben und gucken, ob das Gleiche passiert. Also ihr seht hier Region Last Byte Board, das hat einen Einfluss von 0, das kann man hier auch sehen und jetzt kann ich sozusagen den Kunden einstellen, der eine Reparatur in den letzten Jahren,

drei Jahren hatte. Ich muss das hier ziemlich gut abpassen, dass ich genau auf die 1 komme. Ich würde es gerne von Hand eingeben, das ist nicht so einfach. So, 1. Ne, doch. Und hatte das vor sieben Monaten. Ich muss in die Richtung gehen. Und unten seht ihr, wie sich die... Warte mal, ich habe was falsch gemacht. Also ich muss auf den Value achten unten. Na komm schon. Ich mache es mal so ungefähr her.

Kriegt es nicht hin. Naja, egal. Ungefähr sieben Monate her. Und hier der Value muss eins sein. Und dann seht ihr unten die Wahrscheinlichkeit, wenn wir das hier mal zur Seite tun, liegt ungefähr bei 58%. Naja, gut 56, weil ich es nicht ganz genau eingestellt bekommen habe. Und jetzt könnte ich auch noch die Months since last repair auf 10 stellen. Dann würdet ihr sehen, dass ungefähr 46% rauskommt. Also man kann hier sozusagen, indem man die blauen Punkte rumschiebt,

die Vorhersage für neue Kunden, also für zu klassifizierende Instanzen simulieren. Und allgemein sieht man natürlich auch sozusagen hier, Monthly Slots Repair ist das wichtigste Attribut, der sortiert hier nach Wichtigkeit.

Und wenn es mehr wird, Repairs last three years, wenn es mehr wird, dann steigt auch die Wahrscheinlichkeit, dass das Modell Ja sagt, wohingegen es hier bei dem zweiten Attribut umgekehrt ist. Also je weiter ich nach rechts gehe, desto kleiner wird der Wert. Also je weniger lang es her ist, desto wahrscheinlicher kommt eine Antwort vom Kunden laut Modell. Okay, soweit klar? Verwirrung am Maximum angekommen? Also

Merkt euch einfach, wenn der Koeffizient positiv ist, dann spricht das eher dafür, dass ein höherer Wert des Attributs zu einem Jahr, zu einer Jahrvorhersage führt, ein negatives Umgekehr. Und die Koeffizienten anzuschauen ist hilfreich. Zum Beispiel sieht man, welche Null sind, die irgendwie nichts beitragen zur Klassifikation. Okay, das Nomogramm haben wir auch angeschaut. Ich hatte schon gesagt, die Attribute werden als unabhängig behandelt. Das heißt,

Wir werden keine solchen multivariaten Muster lernen. Wir hatten schon in unserer allerersten Sitzung zusammen hier ein multivariates Muster. Erinnert ihr euch noch? Also wenn der Delivery

Truck etwas zum Small Business bringt, dann gibt es eine Rückgabe der Bestellung. Das war unser multivariates Muster. So etwas gibt es bei logistischer Regression nicht. Jedes Attribut trägt etwas bei, aber es ist nicht möglich, dass bestimmte Kombinationen von Attributwerten

was bestimmtes auslösen. Und das hatte ich schon erwähnt, also das, was da passiert, bei dieser Berechnung der Gewichtsumme und dann bei dem Einsetzen in diese Funktion, ist genau das, was in einem Neuron, in einem neuronalen Netz passiert. Also man stellt das oft auch grafisch dar, man hat hier sozusagen die Inputs von anderen Neuronen und das Neuron selbst nimmt diese Inputs und summiert die auf mit Gewichten und produziert dann selbst zum Aufbruch.

Genau. Letzter Algorithmus, über den ich noch kurz sprechen will, ist Gradient Boosting. Gradient Boosting, also ich habe vorhin gesagt, Cane Nearest Neighbor fällt in eine Familie, die Familie der Lazy Learners. Die logistische Regression fällt in die Familie der sogenannten linearen Lernalgorithmen, also wo man sozusagen so eine Art gerade, also wo man eben

gewichtete Summe bildet, um zu unterscheiden. Und Gradient Boosting fällt in die Familie der sogenannten Meta-Algorithmen. Und Meta-Algorithmen bedeutet, dass mehrere in diesem Fall, vielleicht habt ihr schon mal zum Beispiel von XGBoost gehört, ist eine Ausprägung, die besonders erfolgreich ist,

Und da werden verschiedene Bäume kombiniert. Also ich könnte auch verschiedene logistische Regressionen kombinieren, aber Bäume sind irgendwie beliebt beim Gradient Boosting. Bäume gelten als schwache Algorithmen in dem Sinne, dass sie oft nicht besonders gut performen. Und wenn ich mehrere Bäume kombiniere, können aber erstaunliche Sachen rauskommen. Also mir geht das so ganz grob, das ist jetzt wirklich nicht sehr genau erklärt.

man lernt also erstmal irgendwie einen Baum. Und der macht Fehler und man schaut, was sind das für Fehler und trainiert dann ein weiteres Modell darauf, diese Fehler zu korrigieren beziehungsweise zu vermeiden. Und das macht man so weiter und die Vorhersagen dieser Modelle werden dann am Ende kombiniert, wobei das erste Modell den größten Einfluss hat dabei. Wir werden später noch

ein Problem kennenlernen, was häufig auftritt, das Problem des Overfitting. Und manche dieser Meta-Algorithmen haben Probleme mit Overfitting, Gradient Boosting vermeidet das ganz gut. Das kann natürlich aber auch passieren. Und wenn es um Parameter geht, also bei der logistischen Regression habe ich Parameter ausgelassen, beziehungsweise ich habe es in Orange schnell gemacht, ohne darüber viel zu reden.

Also hier kann ich zum Beispiel auch wieder die Tiefe der Bäume angeben, die da gelernt werden sollen und aber auch die Anzahl. Also wie lange soll er das machen? Ein weiteres Modell trainieren, was die Fehler der vorherigen Bäume hat, versucht zu korrigieren. Der hört dann irgendwann auf, wenn ich ihm sage, lernen nicht mehr als vier Bäume, dann lernt er nur vier. Das kann sehr genau sein, es ist aber nicht mehr interpretierbar in den meisten Fällen. Kann man nicht lesen jetzt, aber ich sage es euch.

Wir wollen gleich eine Übung machen und zwar natürlich mit Orange, sonst macht es keinen Spaß. Und in Orange, wenn ihr da hingeht, dann habt ihr die Algorithmen, die wir gerade kennengelernt haben und noch ein paar mehr zur Auswahl. Also insbesondere, wenn ihr hier auf Model geht, dann seht ihr hier so ein paar Sachen. Vielleicht nochmal kurz kommentieren. Also ihr seht zum Beispiel Gradient Boosting, Random Forest,

Oder AgaBoost, das sind Algorithmen, die gehören eben in diese Familie der Meta-Algorithmen, die also mehrere Modelle miteinander kombinieren. Also der Random Forest, wie der Name schon sagt, kombiniert auch mehrere Bäume. Dann haben wir den Entscheidungsbaum, den kennen wir auch schon. Hier ist unsere absolute Baseline, Constant, haben wir auch schon gesehen. Es gibt auch hier noch sowas wie Regellernler, die werden wir in dieser Tabelle auch gleich sehen. Hier ist unser K-Nearest Neighbor. Ihr seht hier neuronale Netze, die habe ich jetzt...

Nicht wirklich vorgestellt, nur gesagt, die logistische Regression ist die Keimzelle der neuronalen Netze. Die ist hier übrigens. Ich hatte noch erwähnt, dass Naive Bayes existiert und eigentlich ein bisschen ähnlich ist im Verhalten wie die logistische Regression insbesondere. In dem Sinne, dass es die Attribute als unabhängig voneinander behandelt. Und die Tabelle, was nützt die euch? Nützt die euch irgendwas? Die kriege ich nochmal hier in den...

Präsentationsmodus. Also, was seht ihr in der Tabelle? Ihr seht Zeilen und Spalten und Sternchen. Die Spalten entsprechen jetzt so ungefähr den Algorithmen-Familien, die ich gerade erwähnt habe. Also da gibt es Rule-Learners zum Beispiel ganz rechts. Da gibt es in Orange diesen CN2-Rule-Learner als einzigen Vertreter.

Es gibt den Tree in Orange als einzigen Vertreter hier von Decision Trees. In anderen Tools habt ihr auch teilweise mehr Vertreter dieser Familien drin. Aber das hier ist sozusagen ein bisschen generalisiert. Dann gilt es, was hier ein Sternchen ist für alle Vertreter der Familie.

neuronale Netze, da habt ihr natürlich auch sehr viele Möglichkeiten, die zu konfigurieren und verändern die Aussehen. Naive Base, ich habe oben dran noch geschrieben, logistische Regression, weil das ungefähr auch umhaut, was da am Sternchen vergeben wurde. Hier haben wir den Can-Use-Main-Bug. Support-Vektor-Maschinen habe ich noch nicht erwähnt. Da erkläre ich nichts dazu. Ist relativ komplex von der Mathematik und auch hat einige Parameter, die man im Griff haben muss. Gibt es auch in Orange. Mal gucken.

Na komm, wo ist er? Hier, SVM, Support Vector Machine. Also, die Sachen gibt es eigentlich alle in Orange und was bedeuten die Zeilen? Die Zeilen sind Kriterien, die eine Rolle spielen können bei der Auswahl von Algorithmen, je nachdem, wie es aussieht in eurem Kontext, der gegeben ist.

Das oberste ist sozusagen, wie performt so ein Algorithmus im Allgemeinen? Also ihr seht zum Beispiel vier Sterne für Support Vector Machine. Das ist relativ alt hier. Das Paper, aus dem die Tabelle stammt, also Neural Networks, würden heutzutage sicher auch vier Punkte oder vier Sterne haben. Damals waren die noch nicht so tief, wie sie heute sind.

Aber Naive Bayes zum Beispiel performt im Allgemeinen schlechter als alle anderen. Wenn ich sage im Allgemeinen, dann lohnt es sich aber trotzdem immer für euer gegebenes Dataset zu testen, welches der beste Algorithmus ist. Es ist nicht unbedingt immer die Reihenfolge. Insofern würde ich die erste Zeile mal gar nicht so sehr in den Blick nehmen. Die anderen, die sind so ein bisschen verlässlicher. Also,

Speed of Classification zum Beispiel. Da hatte ich schon was erklärt. Ich hatte erklärt zum Beispiel, der Canerous Neighbor, der hat vier Sterne bei Speed of Learning, weil der nämlich nichts lernt. Der lernt kein Modell, habe ich gesagt, sondern der macht alles zur Laufzeit.

Und zwar, wenn eine neue Instanz kommt, wird die mit allen Trainingsinstanzen verglichen. Und das dauert. Das heißt, deswegen hat er nur einen Stern für Speed of Classification. Also Speed of Classification, da brauche ich vier Sterne, wenn ich wirklich Real-Time-Anwendungen habe. Also

wenn ich in Echtzeit Vorhersagen machen will. Ihr seht, der KNN ist der einzige, der da wirklich so aus der Reihe tanzt. Aber natürlich kann auch hier dieses eine Sternchen für SVM

praktische Auswirkungen haben. Also wenn ihr nicht viele Daten habt, das kann dann schon mal ein paar Minuten oder auch mal eine halbe Stunde dauern. Und wenn wir gleich eine Übung machen, dann ist das vielleicht zu lang. Wer weiß. Dann gibt es hier so ein paar Attribute, die haben was damit zu tun, wie robust die Algorithmen reagieren auf

sagen wir mal reale Daten, also Daten, die gewisse Probleme aufweisen, zum Beispiel fehlende Werte oder Attribute, die irrelevant sind oder die miteinander korrelieren, also redundant sind. Oder Noise und Irrelevant Attribute hätte ich jetzt ähnlich gesehen. Und da seht ihr, das verhält sich sehr unterschiedlich, zum Beispiel

ist die Frage, wenn ihr jetzt nicht vorher ein wahnsinnig ausgefeiltes Feature Engineering gemacht habt und ihr habt aber einen Algorithmus, der vier Sterne für Tolerance to Irrelevant Attributes hat, zum Beispiel die Support Vector Machine, dann müsst ihr euch ja keine Gedanken machen. Ihr schmeißt einfach

der Support Vector Maschine die Daten vor die Füße und die wird selbst rausfinden, welches die wichtigen Attribute sind. Das ist sozusagen dann eingebaute Feature Selection. Ihr seht, dass der Tree auch ganz gut ist. Das macht auch Sinn. Er entscheidet zum Beispiel, welches Attribut soll als oberstes verwendet werden, weil das Attribut den geringsten Fehler verursacht. Und wenn man dann sagt, hör irgendwann auf, dann werden automatisch unwichtige Attribute einfach nicht berücksichtigt. Wir haben es auch gesehen bei der logistischen Regression zum Beispiel.

Gut, die kommt hier nicht so gut weg. Ich hatte noch einen Trick dabei, damit viele Nullen aufgetaucht sind. Wie gesagt, sprechen wir noch drüber. Dann kommt hier, da möchte ich noch darauf hinweisen, Explanation Ability, also wie einfach oder schwierig ist es zu verstehen, was das Modell tut, also wie es entscheidet. Also ihr seht zum Beispiel vier Sterne beim Decision Tree. Ich kann mir den Tree angucken und versuchen zu verstehen, was er macht. Bei Cane nearest neighbor habe ich gesagt, ich

Habe kein Modell, was ich anschauen kann. Immerhin kann ich irgendwie so eine Art Erklärung geben, warum etwas auf diese oder jene Art klassifiziert wurde, indem ich mir anschaue, welches die nearest neighbors waren, die da zurate gezogen wurden. Neural Network oder Support Vector Machine, keine Chance.

Und letztens noch hier, was ist damit gemeint? Model Parameter Handling. Damit ist gemeint, wie viele Parameter gibt es und wie schwierig ist es herauszufinden, was der beste Wert für diese Parameter ist. Also zum Beispiel sind die ein Stern pro Sport Vector Machine. Sport Vector Machine hat mindestens drei Parameter, die wichtig sind und es gibt eigentlich keine generelle Aussage darüber, wie man die am besten wählt. Das heißt, man muss das wirklich tunen. Das kann mühsam sein.

Also es gibt Tools dafür, sogenanntes AutoML, wo das ein bisschen einem abgenommen wird. Aber wenn ihr jetzt mit Orange erstmal experimentiert und die Support Vector Maschine wählt, dann werdet ihr diese Parameter sehen. Ihr werdet nicht wissen, welchen Wert ihr wählen sollt. Und die Performance wird davon abhängen, welchen Wert ihr wählt. Das heißt, da seid ihr einfach erstmal ein bisschen verloren. Und da, wo es vier Sterne hat, da könnt ihr nichts falsch machen.

Es gibt einfach zum Beispiel dabei in Naive Bay gar keine Parameter, die verändert werden. Okay, soweit klar. Jetzt ist das hier am Ende. Und das bedeutet, dass wir jetzt die große Übung machen. Die wird wahrscheinlich eine längere Zeit brauchen. Wollen wir vorher nochmal Pause machen? Ja, okay. Ja gut, dann bis wieder 25, so wie gerade eben.

Okay.

Was machen wir jetzt? Es gibt eine Übung, die findet ihr hier in dem Verzeichnis. Sie besteht aus zwei Dingen. Ein PDF und ein Datensatz. Also, Datensatz ist klar. Runterladen und mit dem File Widget in Orange aufmachen und dann geht's weiter. Aber lasst uns kurz vorher, bevor ihr das macht und ich euch schon direkt verliere, noch kurz gucken, worum es geht. Also,

die Firma fixiert. Die haben eine Software und sie haben Kunden im Gesundheitswesen und die haben ein Helpdesk und mit den Kunden ein Service-Level-Agreement getroffen. Und dieses Service-Level-Agreement verspricht allen Kunden, dass sie ein Ticket immer in 10 Tagen bearbeiten oder lösen werden. Und

Was ihr sehen werdet, wenn ihr die Daten anfangs analysieren oder anzuschauen, 25 Prozent der Tickets werden leider nicht in zehn Tagen geschlossen. Und natürlich ist das Management darüber unglücklich und möchte verstehen, warum das so ist. Das ist das eine. Und wenn möglich, auch SLA-Verstöße, also Tickets, die zu lange brauchen werden, womöglich schon frühzeitig vorherzusagen. Also am besten

dann, wenn das Ticket eröffnet wird von Kunden. Ja, und hier gibt es so ein paar Hinweise, was ihr tun sollt. Am Ende, vielleicht schauen wir es uns von hinten an, ihr seht auch, wenn ihr hier auf Moodle guckt, dass es da die Möglichkeit, beziehungsweise für euch die Pflicht gibt, am Ende eine Präsentation hochzuladen. Also am Ende wollt ihr dem Management erklären, was das Problem ist.

und sozusagen die Muster. Es geht wieder um Muster finden, so wie beim ersten Mal. Welches sind die Gründe für die schlechte ETLA-Performance? So, was steht hier? Überlegt, ob ihr alle Attribute braucht. Also ihr könnt mit einem Select Column Widget wieder Sachen rauskicken, die ihr vielleicht für irrelevant haltet. Ihr könnt auch nochmal überlegen, ob die Attribute alle den richtigen Typ haben, also ob

insbesondere alle Attribute, die von Orange offensichtlich als numerisch behandelt werden, ob das wirklich Zahlen sein sollen, mit denen man vielleicht rechnet oder irgendwie kleiner, größer Unterscheidungen macht, ob das Sinn macht, dann, also im File Widget habe ich euch ja gezeigt, könnt ihr das ändern, wenn nötig, dann könntet ihr, also ich habe euch ja diese Tabelle gezeigt und irgendwie habe ich mir vielleicht was dabei gedacht, ihr könntet die nehmen. Ihr könnt auch einfach rumprobieren. Ich sage nicht, dass das falsch ist.

Am Ende müsst ihr euch für einen Klassifikator entscheiden. Nicht mal das müsst ihr. Ihr könnt eigentlich machen, was ihr wollt. Aber ihr müsst mindestens einen Klassifikator auswählen. Ihr könnt dann, wahrscheinlich wollt ihr auch, bevor ihr dem Management irgendwas erzählt, zumindest grob ein Gefühl dafür haben, wie gut dieser Klassifikator seinen Job macht. Und jetzt ist es in Orange gar nicht so einfach. Ich zeige euch das vielleicht einmal mit diesen Swiss Bikes da an.

Das ist ein bisschen absurd, weil wir dann nur 10 Datensätze haben, aber ich versuche es trotzdem. Wenn ihr hier Select Columns gemacht habt und ihr könnt das auch parallel laufen lassen, also ich kann jetzt hier ein Test & Score Widget anhängen und parallel kann ich hier ein Tree trainieren. Dieser Tree wird dann auf allen 10 Datensätzen trainiert.

trainiert werden. Dieses Test-and-Score-Widget tut etwas für euch. Es evaluiert, indem es die Daten nimmt und zerhackt in eine Trainingsmenge und eine Testmenge, im einfachsten Fall. Also ihr könnt zum Beispiel hier einfach mal Random Sampling einstellen, dann passiert das. Und wenn ihr hier 66% nehmt, dann werden zwei Drittel der Daten zum Trainieren benutzt, also in diesem

wirklich ein Spielzeug-Datensatz sind, also dann vermutlich sieben Kunden zum Training genutzt werden und auch drei Kunden wird dann evaluiert. Und der Witz ist, so ein Test-and-Score-Widget hat einerseits den gesamten Datensatz als Input und auf dem Datensatz muss auch die Target-Variable festgelegt sein, das ist ja alles schon passiert hier im Select-Column. Der zweite Input ist ein Algorithmus und der wird von links dran gehängt. Das ist das, also zum Beispiel nehme ich jetzt irgendwie meinen Tree-

Und das ist jetzt nicht der gleiche wie der hier. Der hier wird auf allen Daten trainiert. Und wenn ich ihn hier links dran hänge, die zehn Daten, die zehn Kunden fließen hier rein. Die werden dann, also sieben Kunden werden benutzt, um diesen Tree hier zu trainieren. Dann ist der trainiert. Und der trainierte Tree wird dann auf die drei übrigen Kunden, die als Testmenge zur Seite gelegt worden sind, angewendet.

Und dann wird berechnet, wie gut der Algorithmus ist. Und der Einfachheit halber würde ich jetzt mal vorschlagen, wir schauen einfach hier auf diesen Spalte CA. CA steht für Classification Accuracy. Das ist eine Metrik, die schnell erklärt ist. Die misst einfach, wie viel Prozent der Fälle sagt der Baum das Richtige vorher. Also die richtige Response. Ihr seht jetzt hier 50 Prozent. Das kann man noch optimieren. Aber es ist auch logisch, weil

wir nur auf diesen Datensätzen trainiert haben, das ist ein bisschen wenig. In den Fixit-Daten habt ihr deutlich mehr Datensätze, also ungefähr 95.000. Da solltet ihr verlässlichere Performance-Zahlen rausbekommen. Hier habe ich es nochmal aufgezeichnet. Die Daten fließen hier rein und ihr hängt dann den Learner von links da dran. Und dann kriegt ihr die Zahlen und dann macht ihr die Präsentation. Das ist doch nicht...

Also macht einfach Gruppen, ich würde sagen so drei bis vier Leute, setzt euch wieder auf den Tisch rum, wenn ihr zu viert seid, dann auf jeden Fall, wenn ihr zu dritt seid, bleibt meinetwegen nebeneinander, aber dass ihr wirklich reden könnt miteinander und dann geht's los.

Ich schaue, wie es euch geht und was ich helfen kann. Also ich weiß nicht, wie lange wir brauchen, aber ich schätze mal, das dauert immer eine Weile, so eine dreiviertel Stunde vielleicht. Dann machen wir noch mal kurz Pause und danach hören wir dann eure Präsentationen an, zumindest ausgewählte Präsentationen. Ach so, halt, wir anfangen noch eins, ein letztes. Ich habe das vorbereitet hier auf Moodle, dass ihr es hochladet. Hier könnt ihr eure Präsentationen hochladen und dann...

Schaue ich mir die vielleicht schon mal ein bisschen an und überlege mir in der Pause, welche wir am liebsten uns zu Gemüte führen. Okay, los geht's. Okay.

Also folgendes. Und zwar haben wir herausgefunden, wenn es eine Anfrage ist zu einem Problem, das Hardware, Software oder System betrifft,

haben folgende aufgemischte Personen Mühe damit, offenbar. Und der Vorschlag wäre, mit den Personen zu klären, was für eine Anfrage es ist, also was für ein Anfrageticket es ist und wieso es durch eine Zögerung führt. Und wenn es ein Problem ist, also wie Hardware-Angelegenheiten haben,

Also Angelegenheiten von Issues. Issues, aber auch Request. Wir haben da noch für die, die sich vom Management interessieren, unsere Decision Tree aufgezeigt. Und zwar, wenn es Hardware-Probleme sind, dann folgen die Personen wie Chris Indy, Fred, Tim, Lisa oder Will.

Hier sind sowohl Issues als auch Mitbrüche. Ja, es wird halt zweimal drüber. Dann machen wir eine Schulung. Dann können wir Hardware-Schulungen und Trainings anbieten. Und dann noch unten die Notiz, dass es provisorisch ist und man noch mehr Faktoren gibt. Also, jetzt gab es ganz viele Sachen, die in dieser Übung passiert sind.

Und wir haben einfach keine Zeit mehr jetzt, aber wir werden das natürlich einfach so stehen lassen. Nächste Woche gehen wir nochmal ein, zwei weitere Präsentationen durch oder gucken, was wir dem Management vielleicht noch für andere Messages mitgeben können.

Und dann möchte ich auch nochmal mit euch wirklich die einzelnen Schritte durchgehen. Also, was habt ihr für Attribute aussortiert und warum? Gab es vielleicht irgendwo die Notwendigkeit, den Typ von der Variablen zu ändern? Warum habt ihr euch am Ende alle für ein Decision Tree entschieden? All diese Dinge schauen wir uns nächste Woche natürlich nochmal das Detail an. Aber jetzt erstmal danke. Sehr gerne.

Aufs Management wird es wohl wollen. Okay, also jetzt haben wir ein bisschen überzogen. Danke für eure Geduld und wir sehen uns. Ihr kriegt eine E-Mail. Wir sehen uns nächsten Mittwoch. Sehr schön war das.