Themenblock Datenbeschaffung und -aufbereitung: 04_03Aufbereitung_FeatureEngineering

Weitere Techniken des Feature Engineerings

Erarbeitet von
Dr. Ann-Kathrin Selker

Lernziele

- Du kannst erklären, was Feature Engineering alles beinhalten kann
- Du kannst Diskretisierung, Normalisierung und Feature Construction anhand von Beispielen durchführen

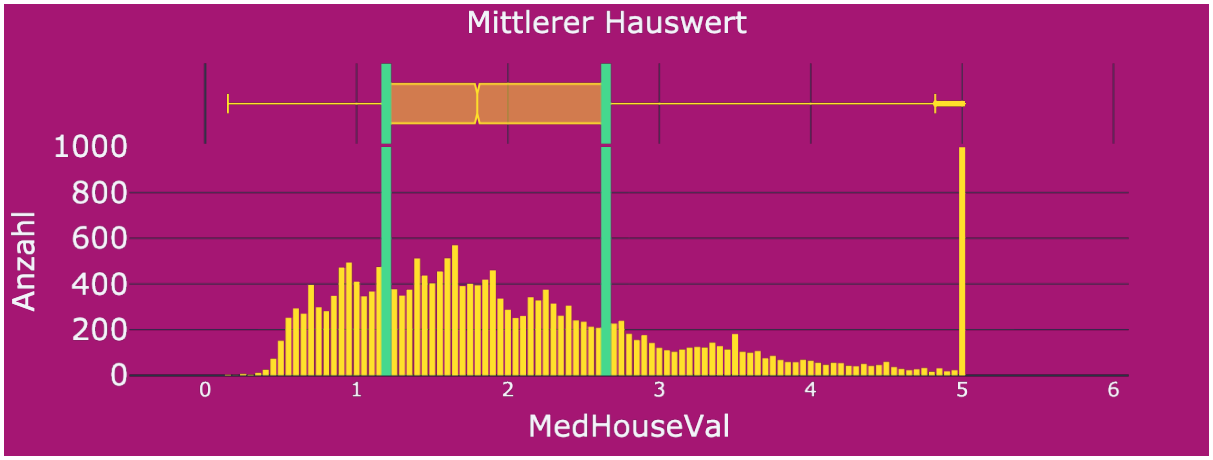
Projekt ai4all „KI für Alle"
HeiCAD –Heine Center for Artificial Intelligence and Data Science

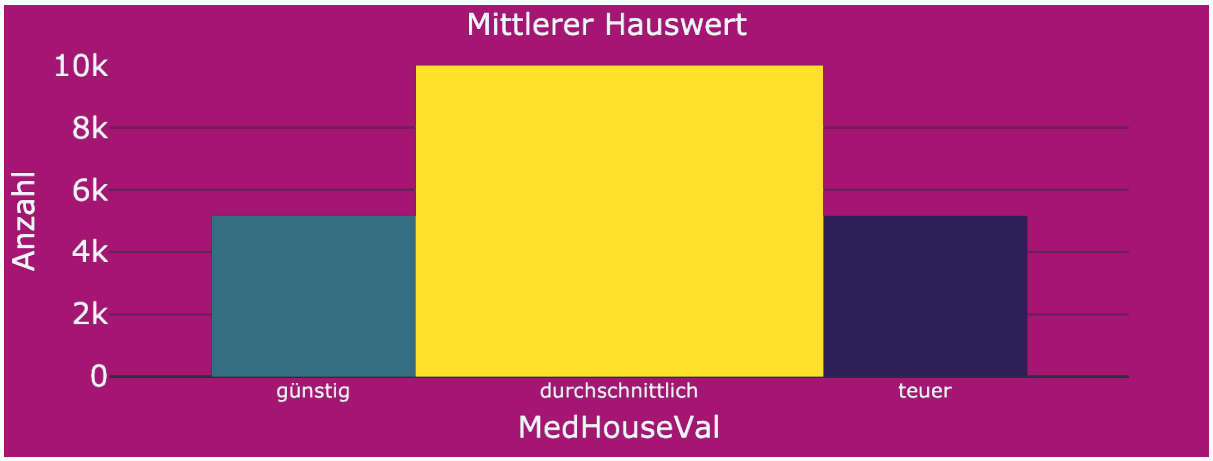
Inhalt

Um unsere Daten später effizient nutzen zu können, müssen sie passend aufbereitet werden. Du hast bereits einige Aufbereitungsschritte gesehen, zum Beispiel das Finden und Eliminieren von Fehlern in den Daten und die Behandlung von fehlenden Werten und Ausreißern. Doch welche Schritte gehören eigentlich noch dazu?

Diskretisierung

Damit du deine Daten für die gewünschte Aufgabe verwenden kannst, müssen sie die richtige Form haben. Um zum Beispiel Klassifikationsalgorithmen anwenden zu können, brauchst du, wie der Name schon sagt, Klassen. Gucken wir uns als Beispiel doch einmal den Datensatz California Housing an. Die Zielvariable ist hier der Hauspreis, der im Grunde beliebige Zahlenwerte annehmen kann. Um diese beliebigen Werte auf Klassen aufzuteilen, können wir die Werte einfach nach gewissen Kriterien gruppieren und so die mögliche Anzahl an verschiedenen Werten reduzieren. Dies nennt sich Histogrammanalyse. Wir erstellen drei Klassen: günstige Häuser, durchschnittliche Häuser und teure Häuser. Ein Histogramm ist übrigens ein Diagramm, bei dem ein Feature zusammen mit einer Aggregierungsfunktion geplottet wird, z. B. der Häufigkeit, in der der jeweilige Wert im Datensatz vorkommt. Die dabei entstehenden Säulen müssen im Gegensatz zum Säulendiagramm keine einheitliche Breite haben. Wenn wir die Klassen so erstellen, dass günstige Häuser unter dem 25 %-Quantil liegen, teure Häuser über dem 75 %-Quantil liegen und durchschnittliche Häuser dem Rest entsprechen, erhalten wir diese Aufteilung.

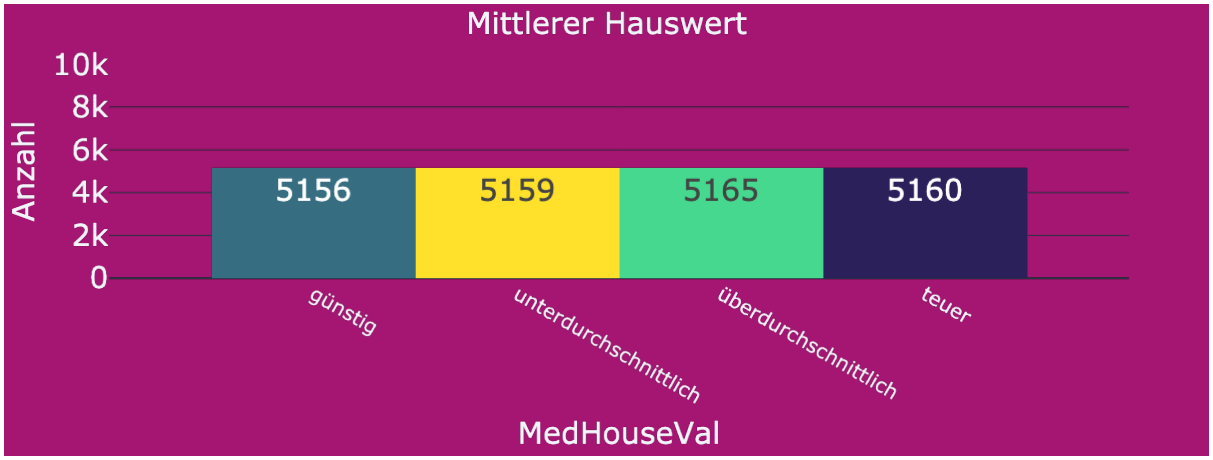




Damit können wir dann ein Klassifikationsmodell trainieren, um uns vorherzusagen, ob ein bestimmtes Haus günstig, durchschnittlich oder teuer sein wird.

Die mathematische Bezeichnung für Werte, die in Klassen vorliegen, lautet „diskret", daher nennt sich die Umwandlung von Daten in diskrete Werte auch „Diskretisierung". Je weniger Klassen entstehen, desto mehr Informationen gehen bei der Diskretisierung verloren. Allerdings führt die Datenreduktion auch zu einem Geschwindigkeitsgewinn beim Training und kann daher auch zu diesem Zweck eingesetzt werden.

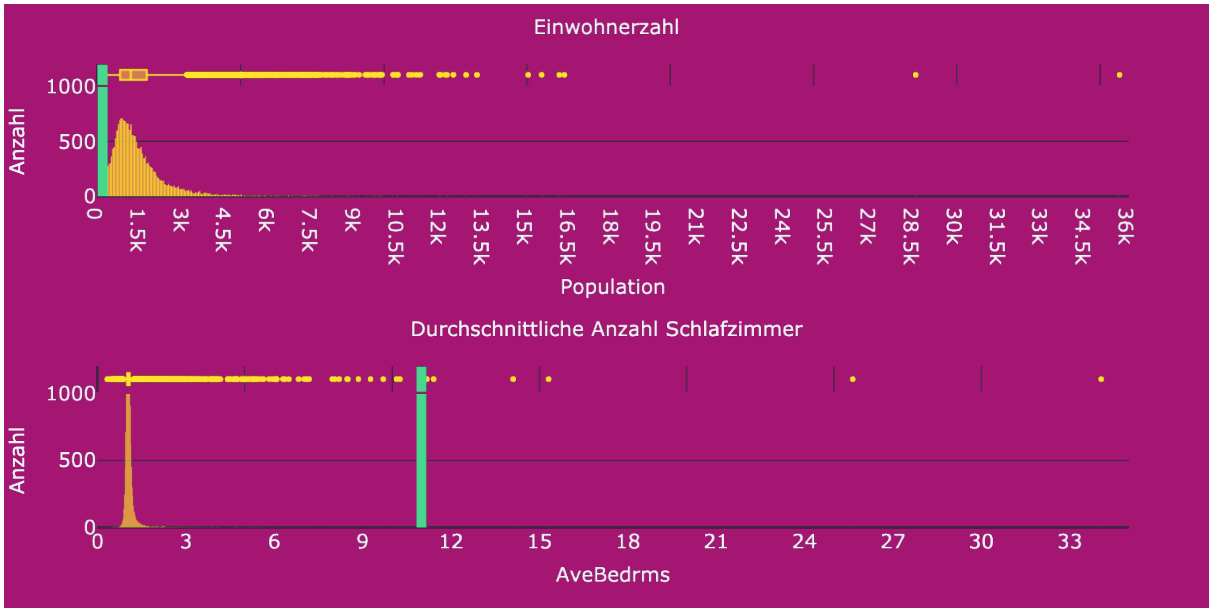
Nicht immer ist es wichtig, welche Klassen bei der Diskretisierung genau entstehen. Falls die genaue Grenze zwischen den entstehenden Klassen nicht entscheidend ist, ist es besser für die Performance von Machine-Learning-Modellen, wenn alle entstandenen Klassen etwa gleich viele Beobachtungen enthalten.



Normalisierung

Nach der Ausreißerbehandlung müssen Daten häufig noch normalisiert werden. Unter Normalisierung oder Bereichsanpassung verstehen wir das Anpassen der Daten auf eine bestimmte Skala, sodass die Werte verschiedener Features direkt miteinander vergleichbar sind. Dies ist bei metrischen Features notwendig. Betrachte zum Beispiel einmal den Datensatz California Housing. Die Werte bei der Einwohnerzahl einer Gegend reichen von 3

bis 35 682, wohingegen sich die Werte bei „Durchschnittliche Anzahl Schlafzimmer“ einer Gegend zwischen 0,33 und 34,07 bewegen. Beide Features nehmen einmal den Wert 11 an, aber trotzdem bedeutet diese 11 ja für beide Features nicht dasselbe.



Wir führen jetzt eine Min-Max-Normalisierung mit der folgenden Formel durch:

$$x\_norm = (x - x\_min) / (x\_max - x\_min)$$

Dabei bezeichnet x den aktuell betrachteten Wert eines Features, x_min den minimalen und x_max den maximalen Wert dieses Features. Der normalisierte Wert x_norm liegt dann immer zwischen 0 und 1. Dadurch kann man auch den Mittelwert und die Varianz zweier Features besser miteinander vergleichen.

$$x1\_norm = (11 - 3) / (35682 - 3) = 8 / 35\,679 \cong 0{,}0002$$
$$x2\_norm = (11 - 0{,}33) / (34{,}07 - 0{,}33) = \cong 0{,}316$$

Nach Einsetzen der Werte erhalten wir das folgende Ergebnis. Bei „Einwohnerzahl” handelt es sich bei der 11 um einen normalisierten Wert von 0,0002, während es sich bei „Durchschnittliche Anzahl Schlafzimmer” bei 11 um einen normalisierten Wert von 0,316 handelt. Dadurch können wir beide Werte jetzt besser vergleichen: Im Kontext vom Feature Einwohnerzahl handelt es sich bei der 11 um einen extrem kleinen Wert, wohingegen 11 im Kontext von „Durchschnittliche Anzahl Schlafzimmer” zwar immer noch zu den kleineren Werten gehört, aber bei weitem nicht so extrem.

Je nach Anwendung benötigst du unterschiedliche Normalisierungsverfahren. Die Normalisierung ist deshalb besonders wichtig, weil einige Machine Learning Verfahren sehr anfällig sind für fehlende Bereichsanpassungen. Dies fällt besonders auf bei Verfahren, die auf Distanzmaßen basieren wie zum Beispiel k-nearest neighbors (k-NN). Dadurch erhalten Features mit einem großen Bereich und damit in sich größeren Distanzen ein stärkeres

Gewicht. Beim diskretisierten Datensatz California Housing führt das Normalisieren vor k-NN zum Beispiel zu einer Verbesserung der Metrik F1 (F1-Score) von etwa 0,46 auf etwa 0,73, was sicher insbesondere an der Normalisierung der Einwohnerzahlen liegt.

Feature Construction

Nicht immer sind bereits alle Informationen schon so in unseren Features enthalten, dass ein Machine-Learning-Modell damit umgehen kann. Hier müssen wir manchmal selber Hand anlegen und Features schaffen, die die fehlenden Informationen widerspiegeln. Betrachte dazu einmal den Datensatz der Stadt New York über die Schwimmbadbesuche der Stadt. Für jedes Datum und jedes Schwimmbad ist hier aufgetragen, wie viele Personen das jeweilige Schwimmbad an dem Tag besucht haben.

Quelle [1]

Ziel eines Machine-Learning-Modells könnte dann sein, für die jeweiligen Schwimmbäder die Besucher*innenzahlen vorherzusagen.

Der Datensatz enthält nur das jeweilige Datum. Besonders für Einrichtungen wie Schwimmbäder ist für die Vorhersage der Besucher*innenzahlen aber nicht das genaue Datum wichtig, sondern stattdessen die Information, ob es sich um Wochenenden, Feiertage oder Schulferien handelt. Wir können also ein neues binäres Feature hinzufügen, dass 1 („wahr") ist, wenn es sich bei dem Datum um einen Tag am Wochenende, an einem Feiertag und/oder in den Schulferien handelt, und 0 („falsch") sonst.

So geben wir dem Modell zusätzliche relevante Informationen für die Vorhersage. Diese Vorgehensweise bietet sich bei Datumsangaben immer an, da Machine-Learning-Modelle nur schlecht Logik und daher auch Muster in diesen Datumsangaben finden können.

Abschluss

Wichtig: Hier werden Annahmen über die Daten gemacht. Diese Annahmen können dazu führen, dass du den Daten Biases hinzufügst. Daher solltest du dies immer hinterher überprüfen und entsprechend reagieren! Feature Engineering ist extrem zeitaufwändig und wird einen Großteil deiner Arbeit ausmachen. Trotzdem solltest du schon mit kleinen Maßnahmen große Erfolge bei der Performance deiner Machine-Learning-Modelle sehen können.

Quellen

Quelle [1] Outdoor Swimming Pool Attendance (24. Mai 2023). Department of Parks and Recreation (DPR) NYC, via https://data.cityofnewyork.us /City-Government/Outdoor-Swimming-Pool-Attendance /jvwx-xnsr/about_data

Weiterführendes Material

https://www.kaggle.com/learn/feature-engineering/course

https://www.geeksforgeeks.org/what-is-feature-engineering/

Disclaimer

Transkript zu dem Video „04 Datenbeschaffung und -aufbereitung: Weitere Techniken des Feature Engineerings“, Ann-Kathrin Selker.